

HEIDELBERG UNIVERSITY

DEPARTMENT OF PHYSICS AND ASTRONOMY

BOOK IN PROGRESS

An Introduction to Computational Physics

Leonard Storcks

February 22, 2024

Abstract

Computation - quickly crunching billions of numbers - can give us a new perspective and understanding of physical systems. This book aims to equip the reader with fundamental knowledge of numerics, computation and statistics as well as tools to tackle problems from physics (and many other disciplines).

Contents

1	Introduction	1
I	Basics of Numerical Computation	2
2	Digital Representation of Numbers	3
2.1	Integer Arithmetic	3
2.1.1	Unsigned integers	3
2.1.2	Two's complement for negative numbers	6
2.1.3	Integer types in C	7
2.1.4	Byte Ordering in Storage: Big and Little Endian	7
2.1.5	Properties and Caveats of Integer Arithmetic	8
2.1.6	Can there be integer-overflow in python?	9
2.2	Floating Point Arithmetic	9
2.2.1	IEEE 754 Floating Point Standard	10
2.2.2	Only a finite set of floating point numbers can be represented exactly	12
2.2.3	Machine Precision is finite	13
2.2.4	Rounding and Pitfalls of Floating Point Arithmetic	13
2.2.5	Rewriting Expressions to Avoid Cancellation I	14
2.2.6	Rewriting Expressions to Avoid Cancellation II	16
2.2.7	Accumulation of Round-off Errors	16
2.2.8	Higher Precision	16
2.3	A more general view on sources of numerical error	17
2.4	Backward error, forward error and condition number	17
2.4.1	Conditioning	17
II	Simulation Methods	19
3	Integration of ordinary differential equations	19
3.1	Notes on ODEs	19
3.1.1	Converting to a first order system	19
3.1.2	Existence and uniqueness of an ODE solution for an initial value problem - Picard-Lindelöf and Lipschitz condition	20
3.2	Introduction of Numerical Integration at the hand of the two-body problem	20
3.2.1	The two-body problem	21
3.2.2	Integrals of Motion	21

3.2.3	Kepler Orbits are Conic Sections	22
3.2.4	Connection of the Runge-Lenz vector to the eccentricity of a conic section	25
3.2.5	Rescaling to Dimensionless variables	25
3.2.6	Solving the two-body problem using explicit (aka forward) Euler	25
3.2.7	Probing the accuracy of an integration scheme - energy error of explicit Euler	26
3.3	Explicit Euler and it's shortcomings	27
3.3.1	Explicit Euler is only first order accurate truncation error	27
3.3.2	Explicit Euler has stability issues	27
3.4	Introduction of the Problem of Stiffness and Implicit Euler to the help	29
3.4.1	Introducing stiffness at the hand of a simple example	29
3.4.2	A <i>definition</i> of stiffness	31
3.4.3	Implicit Euler to the help	31
3.4.4	Region of absolute stability of the Implicit Euler method	32
3.4.5	Implicit Euler for stiff linear ODEs	33
3.4.6	But how can we approach non-linear ODEs using the Implicit Euler method?	34
3.5	Construction of higher-order methods	36
3.5.1	Meaning of going to higher order	36
3.5.2	Approaches to constructing a higher order method	37
3.5.3	Construction by Taylor expansion	37
3.5.4	Runge-Kutta (RK) Integration schemes I: General Idea	38
3.5.5	Runge-Kutta (RK) Integration schemes II: Derivation of the general RK scheme	39
3.5.6	Runge-Kutta (RK) Integration schemes III: General m-substep RK method	41
3.5.6.1	Butcher-Tableau for visualizing the RK coefficients	41
3.5.7	Runge-Kutta (RK) Integration schemes IV: Taylor expansion to identify RK parameters for 2nd order schemes	43
3.5.7.1	Comparison of coefficients	43
3.5.7.2	Resulting integration formula with free parameter q	44
3.5.7.3	Different integration schemes based on the choice of q	44
3.5.8	Runge-Kutta (RK) Integration schemes V: Classical 4th order RK scheme (RK-4)	45
3.6	Adaptive Step Sizes	46
3.6.1	Step halving and doubling method	47
3.6.2	Note on the Local accuracy	48

3.6.3	When does doubling make sense?	48
3.6.4	Adaptively choosing ϵ_0	49
3.6.5	Continuous time step adjustment	49
3.6.5.1	Continuous adaptive time step control scheme	49
3.6.5.2	Embedded Runge-Kutta schemes for cheaper error estimates	50
3.7	The problem of conserved quantities Symplectic Integrators	50
3.7.1	Hamiltonian Systems and Symplecticity	50
3.7.1.1	Poisson brackets and constants of motion (first integrals) . .	51
3.7.1.2	Canonical transformations	51
3.7.1.3	Definition of symplectic transformations	52
3.7.2	Runge-Kutta methods do not conserve energy and are not symplectic	52
3.7.3	Symplectic integrators to the help	55
3.7.4	Verlet Scheme	56
3.7.4.1	Velocity Verlet algorithm	57
3.7.5	The Leapfrog Method	58
3.7.5.1	Connection between Leapfrog and Velocity Verlet	59
3.7.5.2	Kick-drift-kick and Drift-kick-drift Leapfrog formulations to have velocity and position information at the same time . .	59
3.7.5.3	Advantages of the Leapfrog scheme	61
3.7.5.4	Leapfrog is symmetric (time reversible)	61
3.7.5.5	Symplecticity of the leapfrog scheme I: Intuition and Meaning	62
3.7.5.6	Symplecticity of the leapfrog scheme II: Proof	63
3.8	Extrapolation method: Bulirsch-Stoer algorithm	66
3.8.1	Basic integration method second order method with $\mathcal{O}(h^2)$; midpoint rule \rightarrow modified midpoint rule	67
3.8.1.1	Modified midpoint rule	68
3.8.1.2	Combining modified midpoint calculations with different h ; advantage of modified midpoint	68
3.8.1.3	What extrapolation nodes to choose? - how to increase n (or rather decrease h)	69
3.8.1.4	How to extrapolate from multiple $F(h_n)$ to the limit $h \rightarrow 0$?	69
3.9	Predictor-corrector methods	70
3.9.1	One-step predictor-corrector method: RK2 and $P(EC)^k$	71
3.9.2	4th order Adams-Bashforth-Moulton	71
3.10	Shooting adapting parameters until boundary conditions are fulfilled	72
3.10.1	Remark on ODE solutions in phase space	72
3.10.2	Examplary Shooting Problem	72
3.10.3	Shooting	73

4 Simulation of Physical Systems - from Quantum Mechanics to Fluid Dynamics	74
4.1 Different levels of modelling from Quantum Mechanics to Kinetic Gas theory	74
4.2 From a classical particle description to the Boltzmann equation	75
4.3 Emergence of irreversibility in the Boltzmann equation	76
5 Basic Fluid Dynamics	78
5.1 Basic notes on fluid description - the fluid from the view of a parcel - macroscopic fluid view	78
5.1.1 When is a fluid description valid?	78
5.1.1.1 Connection between temperature and random movement .	78
5.1.1.2 Continuum Hypothesis	78
5.1.2 Example: the plasma in the intercluster medium can be considered a fluid*	79
5.1.2.1 Mean free path in a model of colliding spheres	79
5.1.2.2 Collisional cross-section of an electron in a plasma and first approximation of λ_{mfp}	80
5.1.2.3 A better approximation to the mean free path in an ionized plasma	80
5.1.3 Fluid description based on fluid parcels	81
5.1.3.1 Eulerian and Lagrangian fluid dynamics	81
5.1.3.2 Continuity equation	82
5.1.3.3 Incompressible fluids	82
5.1.3.4 Equation of motion of a fluid parcel, general path towards Navier-Stokes	83
5.2 Basic Gas Dynamics	84
5.2.1 Distribution function and Boltzmann equation	84
5.2.2 Retrieving information from the Boltzmann equation	85
5.2.3 Mass conservation continuity equation (1st moment)	85
5.2.3.1 Derivation of the continuity equation*	86
5.2.4 Momentum conservation Navier Stokes equation (2nd moment) . . .	86
5.2.4.1 Notes on the derivation of the Navier-Stokes equation	87
5.2.4.2 Interpretation and viscous stress tensor for a Newtonian fluid	87
5.2.5 Energy conservation (3rd moment)	88
5.2.5.1 Notes on the conductive heat flux	89
5.2.5.2 Evolution equation for the total specific energy $e = e_{th} + \underline{v}^2/2$	89
5.2.6 Entropy conservation	90
5.3 Euler Equation and Navier-Stokes equation	90

5.3.1	Euler Equations	90
5.3.2	Navier-Stokes equation	92
5.3.2.1	Simplification of the Navier-Stokes equations for incompressible fluids ($\nabla \cdot \underline{v} = 0$)	93
5.3.2.2	Characterizing flow Reynolds number	93
5.4	Shocks	94
5.4.1	Propagation of disturbances 1: Speed of sound	95
5.4.2	Characteristics of Perturbations	96
5.4.3	Formation of a shock	97
5.4.3.1	Formation as a pressure driven compressive disturbance . . .	97
5.4.3.2	Causes for shocks	97
5.4.4	Collisional and collisionless shocks shock front	97
5.4.5	Properties at fluid discontinuities	98
5.4.5.1	(Rankine-Hugoniot) Jump conditions I: Assumptions	98
5.4.5.2	(Rankine-Hugoniot) Jump conditions II: Jump condition from the continuity equation	99
5.4.5.3	(Rankine-Hugoniot) Jump conditions III: Jump condition from the momentum equation	99
5.4.5.4	(Rankine-Hugoniot) Jump conditions IV: Jump condition from the energy equation	100
5.4.5.5	Types of discontinuities: contact discontinuity vs. shock . . .	100
5.4.6	Characterizing the Shock strength - Mach number	101
5.4.6.1	Occurrence of the Mach number in the continuity equation .	101
5.4.6.2	Rewriting the Rankine-Hugoniot jump conditions in terms of \mathcal{M}_1 - relating pre- and post-shock quantities	102
5.4.6.3	Conversion of kinetic to thermal energy in the shock	102
5.4.6.4	Conservation of energy in the shock	102
5.4.6.5	Connection between pre- and post-shock Mach number . . .	104
5.4.6.6	Shock adiabatic curve*	104
5.4.6.7	Oblique shocks	105
5.5	Fluid instabilities	106
5.5.1	Stability of a shear flow	106
5.5.2	Rayleigh-Taylor instability	106
5.5.3	Kelvin-Helmholtz instability	107
5.5.4	Further instabilities	107
5.6	Turbulence	107
5.6.1	Subsonic (incompressible) turbulence, low Mach numbers rotational modes	108

5.6.2	How to quantify turbulence? - Reynolds number	109
5.6.2.1	Reynolds number as the ratio between advection and dissipation timescale	109
5.6.3	Supersonic turbulence, shocks $\mathcal{M} \gg 1$ rotational and compressive modes	109
5.6.4	Schematic concept of turbulence	110
5.6.5	Kolmogorov scales of turbulence	110
5.6.5.1	Dissipation scale - smallest scale to be resolved in a simulation	110
5.6.6	Scaling of the eddy velocity and vorticity in the inertial range	110
5.6.7	Power spectrum of Kolmogorov turbulence	111
5.6.7.1	Derivation of the energy spectrum of Kolmogorov turbulence	111
6	Eulerian Hydrodynamics Solving PDEs	112
6.1	Introductory notes on PDEs	112
6.2	Types of PDEs	113
6.2.1	Classification of linear 2nd order PDEs in analogy with conic sections	113
6.2.1.1	Derivation homogeneous solutions are conic section in k -space	113
6.2.1.2	Classification into elliptic, parabolic, hyperbolic	114
6.2.1.3	Qualitative differences on the types of PDEs	114
6.2.2	Typical examples and classification of homogeneous 2nd order PDEs .	115
6.2.3	Classification of linear 2nd order PDEs with more unknowns	115
6.2.4	Linear systems of 1st order homogeneous PDEs	116
6.2.4.1	Extension to conservation laws	116
6.3	Solution schemes for PDEs	117
6.4	Advection - Keep information flow in the physical system in mind	119
6.4.0.1	Analytic solution to the advection equation	119
6.4.0.2	Simple but wrong approach we need to consider the flow of information	119
6.4.0.3	Directional splitting / upwind scheme to the rescue	120
6.4.0.4	Where does the smoothing in the upwind scheme come from?	122
6.4.0.5	What is the maximum timestep we can take? Courant-Friedrichs-Lowy (CFL) criterion	122
6.4.0.6	Hyperbolic conservation laws changing upwind direction . .	123
6.4.0.7	What if identifying the local characteristics is very difficult?	124
6.5	Intermezzo: CFL like criterion and connection to stiffness in a reaction diffusion system	124
6.6	Riemann problem Riemann solvers	127
6.6.1	Structure of the solution of the Euler-Riemann-Problem	128

6.6.1.1	Characteristics of the three waves	128
6.6.1.2	Example Riemann-Problem situation	128
6.6.1.3	Properties of shock, contact discontinuity and rarefaction wave	129
6.7	Finite volume discretization Reducing a hyperbolic conservation law to a Riemann problem Godunov scheme	131
6.7.1	Problem solve a hyperbolic conservation law PDE	131
6.7.2	Deriving a finite volume scheme where only Riemann problems are left to solve	131
6.7.2.1	Caveats of the Godunov scheme	133
6.7.3	Godunov's method and Riemann solver reconstruct - evolve - average (REA)	133
6.8	Approximate Riemann solvers HLL solver	134
6.8.1	1D Riemann problem to solve	134
6.8.2	Basic HLL assumptions and problem statement	134
6.8.3	Deriving the solution of the Riemann problem in the HLL scheme . .	135
6.8.3.1	Derivation of the middle state u^{HLL} at $t = T$	135
6.8.3.2	Deriving the intercell flux $f^{HLL} = f^*$	136
6.8.4	Final HLL solution	136
6.8.5	Mind that the extreme velocities can point into the same direction .	137
6.8.6	Godunov scheme with HLL solver	137
6.8.7	Pointers to extensions of the HLL scheme	138
6.8.8	Ansätze for the maximum wave velocities S_L and S_R	139
6.9	Extension of Eulerian hydrodynamics to multiple dimensions	139
6.9.1	Dimensional splitting Ansatz	140
6.9.1.1	1st order ansatz	141
6.9.1.2	2nd order accurate in 2D examples	141
6.9.2	2nd order accurate in 3D example	141
6.9.3	Unsplit schemes	142
6.10	Extensions for high-order accuracy	143
6.10.1	What even is a schemes order?	143
6.10.2	2nd order extension to Godunov's scheme by changing the reconstruction step from piecewise-constant to linear	143
6.10.2.1	How to estimate the time derivatives $(\partial_t \underline{U})_i$? MUSCL-Hancock scheme	145
6.10.3	Idea and discussion of even higher order methods	145
6.10.3.1	Discussion of higher order methods	146
6.11	Flux / slope limiters adaptively switching between a high and low order method	147

6.11.1	Possibly advantageous properties of the flux limiter	149
7	Smoothed Particle Hydrodynamics - Lagrangian Particle Method	150
7.1	Lagrangian fluid equations (i.e. as material derivatives)	151
7.1.1	Continuity equation	151
7.1.2	Navier-Stokes equation Conservation law of Linear Momentum . . .	151
7.1.3	Energy equation	151
7.2	A simple SPH fluid simulator*	152
7.3	Smooth then discretize - smoothing kernels and their usage	153
7.3.1	Properties of the smoothing approach for calculating derivatives of the smoothed fluid quantities	153
7.3.2	Discrete formulation of the smoothing	154
7.3.3	Why a kernel with compact support is preferred?	154
7.3.4	How to make the smoothing length h variable in space to account for variations in the density? sampling procedure in SPH - scatter and gather approach	157
7.3.4.1	How to choose h_i ?	158
7.4	SPH continuity equation and equations of motion	160
7.4.1	SPH continuity equation	160
7.4.2	Gradients in SPH	161
7.4.3	SPH Euler equation The central ingredient to making our simple fluid simulator work	162
7.4.4	Including further accelerations	162
7.5	Artificial Viscosity	163
7.5.1	Viscous Pressure	163
7.5.2	Adding the artificial viscosity to the equation of motion	163
7.5.3	Shear-Flow-Balsara correction	164
7.5.4	Further viscosity switches	164
7.6	SPH energy equation with artificial viscosity	165
7.7	SPH Entropy equation	165
7.8	Maximum timestep - CFL criterion	166
7.9	Notes on boundary modeling*	166
7.10	Reversibility in the context of viscosity-free, weakly-compressible SPH* . . .	166
7.11	Notes on the conservative formulation using Lagrange multipliers	169
7.12	Further improvements	169
7.13	Advantages and Disadvantages of SPH	169
7.14	Outlook: Machine-learning enhanced multiscale-physics SPH simulation . .	170

8 Finite Element Methods	172
8.1 Finite element methods for linear PDEs	172
8.1.1 The solution is represented by weighted base functions on nodes within finite elements	172
8.1.1.1 Example 1D linear reconstruction	173
8.1.2 From the PDE to an algebraic equation for the expansion coefficients ϕ_1, \dots, ϕ_n	174
8.1.2.1 Inserting the finite element approximation into the PDE yields a residuum	174
8.1.2.2 Finding the expansion coefficients by minimizing the residual in some sense	175
8.1.2.3 A linear system for ϕ_1, \dots, ϕ_N in the Galerkin scheme	176
8.1.2.4 Example Application of Galerkin FEM	176
8.2 Discontinuous Galerkin Method	178
8.2.1 Problem we want to tackle Euler equations	178
8.2.2 Steps in formulating the Discontinuous Galerkin (DG) scheme	179
8.2.3 Subdivision and Representation modal vs nodal	179
8.2.3.1 Example for an orthogonal polynomial basis: Legendre polynomials	181
8.2.4 Solving for the weights	182
8.2.4.1 What even is Gauss-Legendre quadrature?*	182
8.2.5 Finding initial weights - just apply the determination of weights to the initial state	183
8.2.6 Evolution equation for the weights	184
8.2.7 Efficiency of DG and refinement schemes	184
9 Diffusion	186
9.1 Thermodynamic Basics of Diffusion	186
9.1.1 Mean squared velocity and mean squared relative velocity	186
9.1.2 Mean free path and relaxation time	186
9.1.3 Random Walk spreading Gaussian distribution in space	187
9.2 Diffusion equation	188
9.2.1 Derivation of the diffusion equation Fick's law from the microscopic consideration	188
9.2.2 Analytical Solution to the diffusion equation via Fourier transform	189
9.2.2.1 Solution for an initial delta peak in the density over space	189
9.3 Numerical solutions	190
9.3.1 Forward in time, central in space	190

9.3.1.1	Discretized diffusion equation	190
9.3.1.2	Explicit scheme for performing a time step	190
9.3.1.3	Stability of the central in space, forward in time scheme for diffusion	191
9.3.2	Backward in time, central in space	192
9.3.2.1	Discretized implicit scheme	192
9.3.2.2	Matrix equation	192
9.3.3	Crank-Nicolson method	192
9.4	Flux-limited diffusion (<i>tempered</i>)	193
9.5	Diffusion in three dimensions	194
10	Solving Linear Equations with Iterative Solvers and the Multigrid Technique	195
10.1	Motivational Example 1: From the Poisson equation we can get a possibly big linear system	195
10.2	Poisson equation and solving a tridiagonal system	196
10.2.1	1D heat Diffusion equation with Dirichlet boundaries in matrix form	196
10.2.2	Forward elimination backward substitution method for solving a tridiagonal system	197
10.3	Classical Exact Solution: LU-decomposition*	198
10.3.1	Solving the linear system for \underline{x} when we already know the LU-decomposition	198
10.3.2	Calculating the LU-decomposition in $\mathcal{O}(N^3)$	199
10.4	Jacobi iteration a splitting method	200
10.4.1	When does the Jacobi iteration converge?	200
10.4.1.1	Derivation of the convergence criterion	201
10.4.2	Example Jacobi Step	201
10.5	Gauss-Seidel iteration better splitting method	201
10.5.1	Motivational Example	202
10.5.2	Gauss-Seidel update	202
10.5.3	The problem of parallelization in Gauss-Seidel and red-black ordering	203
10.6	Relaxation problem Poisson equation in red-black ordering	204
10.6.1	Formulating an elliptic equation as an equilibrium of a relaxation problem	204
10.6.2	Red-black ordering for the 2D Poisson equation	205
10.7	Multigrid technique	205
10.7.1	Getting finer and coarser prolongation and restriction	206
10.7.1.1	Coarse-to-fine interpolation called prolongation	207
10.7.1.2	Fine-to-coarse restriction	208

10.7.1.3 Relation of restriction and prolongation	208
10.7.1.4 Short-hand stencil notation	210
10.7.2 Multigrid V-cycle	210
10.7.2.1 Initial definitions	211
10.7.2.2 Coarse-grid correction scheme	211
10.7.2.3 V-cycle	212
10.7.2.4 Full multigrid method	213
10.8 Krylov subspace methods	214
10.8.1 Motivation for the need of Krylov subspace methods in the context of non-linear root-finding of big systems*	215
10.8.2 Motivation solving a system of linear equations using a gradient method	215
10.8.3 Krylov subspace and construction of iterative methods for solving $\underline{\underline{A}}\underline{x} = \underline{b}$	218
10.8.4 Conjugate gradients method	219
11 Fourier methods	220
11.1 Convolution problems solving Poisson's equation using Fourier methods .	220
11.1.0.1 The solution to the Poisson equation is a convolution	220
11.1.0.2 A convolution in real space turns into a multiplication in Fourier space	221
11.1.0.3 For periodic boundaries, the Fourier transform turns from an integral to a sum	221
11.1.0.4 Solution to the Poisson equation in Fourier space	222
11.2 Discrete Fourier Transform (DFT)	223
11.2.0.1 The spatial discretization (and periodicity) limits the number of \underline{k} vectors that lead to possibly different $\hat{p}_{\underline{k}}$	223
11.2.0.2 Resulting discrete Fourier transform	224
11.2.0.3 Different conventions for the wave vector \underline{k} and Nyquist fre- quency	224
11.2.0.4 Plancherel's theorem	226
11.2.0.5 Normalization factor	226
11.2.1 Computational Complexity and Fast Fourier Transform (FFT)	227
11.3 DFT storage conventions	227
11.4 DFT and Linear and Cyclic Convolution	231
11.5 Non-periodic problems with <i>zero-padding</i> in 2D	237
11.6 Power spectra and correlation functions	238
11.6.1 Definition of the power spectrum	238
11.6.2 Definition of the correlation function	239

11.6.3	Connection by Fourier Transform	239
11.6.4	Power function and variance of a smoothed field	239
11.7	Projections in Fourier space	240
11.7.1	Fundamental theorem of vector analysis Helmholtz decomposition .	240
11.7.2	Proof in Fourier space	240
11.7.3	Applications of the Helmholtz decomposition	241
11.7.4	Sidenote: Importance of dimensionality	241
12	Collisionless particle systems	242
12.1	Introduction of collisionless systems in the context of fluid modeling	242
12.2	Structure of the following considerations	244
12.3	General N-particle ensembles one-, two-, three, ..., particle distribution BBKGY chain	244
12.4	Uncorrelated (collisionless) systems multiplication closure to the BBGKY chain	245
12.4.1	General Continuity equation for probability in phase space	246
12.4.2	Liouville equation for the general evolution of phase space probability	246
12.4.3	Vlasov equation for collisionless systems	246
12.4.4	Accelerations in collisionless systems - including gravity into the Vlasov equation	247
12.5	When is a gravitational system collisionless?	247
12.5.1	Relaxation time in a gravitational system	248
12.5.2	Crossing time	248
12.5.3	Change in perpendicular velocity when crossing the system	249
12.5.3.1	Size of one small angle deflections based on the impact parameter	249
12.5.3.2	Mean squared deflection for many small deflections	250
12.5.3.3	Many small deflections are more important than few large ones	251
12.5.3.4	Finally the calculation of $\langle(\Delta v_{\perp})^2\rangle$	251
12.5.3.5	How to choose b_{min} and b_{max} in the Coulomb logarithm? . .	252
12.5.4	Examples of astrophysical relaxation times	252
12.6	N-body models of collisionless systems	253
12.6.1	The softening length ϵ	253
13	Force calculations tree algorithms and particle mesh technique	255
13.1	Calculating the forces Direct summation	255
13.2	Overview on faster, approximate force calculation schemes	255
13.3	Faster method I Multipole expansion tree algorithms	256

13.3.1	Deriving the multipole expansion	257
13.3.2	Multipole expansion	258
13.3.3	Hierarchical grouping baseline for smart force calculations	258
13.3.3.1	Constructing the tree Barnes and Hut oct-tree	259
13.3.4	Tree walk - force calculation with adaptive group resolution	261
13.3.4.1	Tree walk algorithm	262
13.3.4.2	Derivation of the cost of the tree based force calculation $\mathcal{O}(N \log N)$	263
13.3.4.3	Expected typical force errors in a monopole approximation $(\Delta F_{\text{tot}}) \propto \theta_c^7$	263
13.4	Faster method II Particle mesh technique for efficiently computing long-ranged forces	264
13.4.1	Schematic particle mesh algorithm	264
13.4.2	Mass / charge assignment of particles to mesh cells	264
13.4.2.1	Assignment scheme I Nearest grid point (NGP) assignment δ -shape function	266
13.4.2.2	Assignment scheme II Cloud in cell (CIC) assignment top-hat shape function	267
13.4.2.3	Assignment scheme III Triangular shaped cloud (TSC) assignment triangular shape function	268
13.4.2.4	Comparing the assignment schemes in terms of continuity .	268
13.4.3	Solving the Poisson equation for the potential on the mesh based on the meshed density	269
13.4.4	Calculating the force field on the mesh	269
13.4.4.1	On the choice of the order of the finite difference scheme for the force calculation	270
13.4.5	Interpolating the force from the mesh to the particles	270
13.4.5.1	We assign forces to the particles' positions using the same assignment kernel used to assign mass of the particles to the grid points	270
13.4.5.2	Proof that for the same assignment kernel in the density and force assignment there is <u>no self-force occurring</u>	271
13.4.5.3	Proof that for the same assignment kernel in the density and force assignment, the forces between particle pairs are pairwise antisymmetric	273
13.5	Outlook - combining the particle mesh and tree method	273

14.1	Random Number Generation and Sampling	275
14.1.1	An intuitive introduction to sampling	275
14.1.2	Random Number Generators - Base of all sampling methods: Sampling from the uniform distribution is <i>easy</i>	275
14.1.2.1	Advantages of Pseudo-Random Number Generators over True Random Number Generators	276
14.1.2.2	Desirable properties of good pseudo-random number generators	276
14.1.2.3	A simple class of pseudo-random number generators: Linear Congruential Generators	276
14.1.2.4	A first improvement Combining multiple LCGs	278
14.1.2.5	Lagged Fibonacci Generators	278
14.1.2.6	Roughly evenly spaced sampling - blue noise	279
14.1.3	Inverse Transform Method sampling from distributions with algebraically invertible cumulative distribution functions (CDFs)	279
14.1.3.1	Alternative derivation from the transformation between probability distributions*	280
14.1.3.2	Example: Inverse Transform Method applied to the standard Laplace distribution*	281
14.1.3.3	Sampling from a Gaussian using exact inversion Box-Muller trick	281
14.1.3.4	Sampling from a discrete distribution*	283
14.1.4	Acceptance-rejection method	284
14.1.4.1	Example: Sampling uniformly from a sphere	285
14.1.4.2	Example II: Sampling from a conditioned Gamma distribution*	286
14.2	Monte Carlo Estimation	288
14.2.1	A basic intuition for estimation	288
14.2.2	Monte Carlo Estimation	289
14.2.3	Distribution and Error of the Monte Carlo Estimator	289
14.3	Monte Carlo Integration	290
14.3.1	Intuition for Monte Carlo Integration	290
14.3.2	Connection to the Monte Carlo Estimator	291
14.3.3	Intuition from calculating the area of a shape	291
14.3.4	Error in Monte Carlo Integration - scaling with $\frac{1}{\sqrt{N}}$	291
14.3.5	Distribution of \hat{I}_N and derivation of the central limit theorem*	292
14.3.6	Illustrative Example of Monte Carlo Integration	294
14.3.7	Comparison to other techniques - when to use Monte Carlo integration?	295
14.3.8	Reducing the variance of the Monte Carlo Estimator	297
14.3.8.1	Antithetic Estimators*	297

14.3.8.2	Importance Sampling	298
14.4	Sampling with a stochastic process	302
14.4.1	Markov Process and Markov chain Monte Carlo Markov Chain (MCMC)	302
14.4.1.1	Characterizing property of the Markov process Memory-lessness	302
14.4.1.2	Transitioning the probability distribution	303
14.4.1.3	Properties demanded of the update step f stochastic process has equilibrium distribution ergodicity	303
14.4.1.4	Nice consequences of the demanded properties of the update step	303
14.4.1.5	Proof that updating the probability distribution converges to its fixed-point equilibrium distribution p_{eq}	304
14.4.1.6	Detailed balance Common condition for the update steps .	304
14.4.2	Metropolis Hastings algorithm - simple and generic construction of the transition f	305
14.4.2.1	The Metropolis Hastings Algorithm fulfills detailed balance .	305
14.4.2.2	Metropolic update for a symmetric proposal operator the Hastings ratio simplifies	306
14.4.3	Example: Stochastic Sampling from a Gaussian	306
14.4.3.1	Choosing a symmetric proposal function	307
14.4.3.2	Metropolis Hastings Algorithm for sampling from a Gaussian	307
14.5	Monte Carlo Simulations of Thermodynamic Systems	308
14.5.1	Calculating the ensemble average $\langle A \rangle$ using importance sampling powered by a stochastic process	309
14.5.1.1	Sampling the states ϕ_i using the Metropolis-Hastings algorithm	309
14.5.2	Example thermodynamic model: Ising Model	310
14.5.2.1	Exemplary parameter of interest: Critical temperature under which spontaneous magnetization occurs	310
14.5.2.2	Metropolis algorithm for simulating the spin configuration on the lattice (and so the partition function and magnetization, ...)	311
14.5.2.3	More thermodynamic properties that might be of interest* .	312
14.6	Monte Carlo Markov Chains in Parameter Estimation	312
14.6.1	Maximum Likelihood Estimation	312
14.6.1.1	Estimating parameters based on the posterior	313
14.6.2	Monte Carlo Markov Chain that samples the posterior as an equilibrium distribution	315
14.6.3	Caveat of MCMC we never know when the chain is long enough .	315

15 Parallelization Techniques	317
15.1 Why do we need to parallelize?	317
15.2 Hardware perspective and parallelism	317
15.2.1 We should write code reducing memory access row and column major storing convention for matrices	317
15.2.2 General computer architectures in increasing complexity*	318
15.2.3 CPU vs GPU	319
15.2.3.1 When and how to use GPUs	320
15.2.4 Vector cores	320
15.2.5 Hyperthreading	320
15.3 Types and challenges of concurrency and parallelism	321
15.3.1 Shared memory and message passing concurrency	321
15.3.2 Challenges of concurrency	322
15.4 Shared memory parallelization (with OpenMP)	322
15.4.1 Simplest parallelization - parallel for loop of independent iterations .	322
15.4.2 OpenMP's fork-join model	322
15.4.3 Race conditions	323
15.4.3.1 Race conditions in the context of non-atomic operations .	323
15.4.4 Using the same variable where it is not intended	324
15.5 Message passing concurrency enabling distributed memory parallelization (with MPI)	325
15.5.1 Architecture of an MPI program	326
15.5.2 MPI program structure and basic communication	326
15.5.2.1 Point-to-point communication	326
15.5.2.2 Collective communication	327
15.5.2.3 More complex communication	328
15.5.3 Pinning - distribution of processes on cores and nodes	328
15.5.4 Notes on reading and writing	328
15.6 Parallel computing for physical simulations	329
15.6.1 Parallelized hydrodynamics	329
15.6.1.1 Adaptive grid e.g. for the simulation if interstellar gas . . .	331
15.6.1.2 Inclusion of long-range forces long range forces vs. parallelization	332
15.7 Scaling of the processing time with increasing parallelism and Ahmdal's law	333
15.8 Examples of parallel algorithms	334
15.9 Notes on scheduling	334

III Computational Statistics and Data Analysis	334
16 Basic probability theory	335
16.1 Basic definitions	336
16.1.1 Sample space Ω	336
16.1.2 Set of events E defined on the sample space	336
16.1.3 Probability measure P	336
16.1.3.1 Examples for the probabilities of events discrete and continuous situations	337
16.2 Basic probability laws	337
16.2.1 Conditional and joint probability	337
16.2.2 Bayes rule	338
16.2.3 Probability of the union of events	338
16.2.4 Independent events	339
16.2.5 Example for throwing one dice (once)	339
16.2.6 Probability of the complement	340
16.2.7 Law of total probability marginalization	340
16.3 Random Variables (RVs)	340
16.3.1 Examples for random variables	341
16.3.2 Continuous and discrete Random variables	341
16.3.3 Expectation values	341
16.3.3.1 Properties of the expectation value	341
16.3.3.2 Conditional expectation	342
16.3.3.3 Law of total expectation	343
16.3.4 Variance, Covariance and Correlations	343
16.3.4.1 Variance	343
16.3.4.2 Covariance and correlation	344
16.3.5 Sample mean and variance	346
16.4 Univariate probability distributions	346
16.4.1 Discrete probability distributions	346
16.4.1.1 Discrete uniform distribution	346
16.4.1.2 Bernoulli distribution probability distribution in a binary setting	347
16.4.1.3 Binomial distribution probability of obtaining m heads in N coin tosses	347
16.4.1.4 Geometric distribution probability of the first success at trial k	349

16.4.1.5	Hypergeometric distribution k successes in r draws without replacement	350
16.4.1.6	Poisson distribution probability of k events that have a mean rate occurring in a fixed interval	352
16.4.2	Continuous probability distributions	354
16.4.2.1	Cumulative distribution function	354
16.4.2.2	Probability density function	354
16.4.2.3	Quantiles	355
16.4.2.4	Uniform distribution	356
16.4.2.5	Gaussian (normal) distribution	356
16.4.2.6	Beta distribution	357
16.4.2.7	Gamma distribution	358
16.4.2.8	Exponential distribution	360
16.4.2.9	Exponential family distributions and conjugate priors	360
16.5	Tschebysheff's theorem	363
16.5.1	Proof of Tschebysheff's theorem	364
16.6	Moment generating functions	364
16.6.1	Calculating moments from the moment generating function	365
16.6.2	Multidemensional moment generating function	365
16.6.3	Example: Moment generating function of the Poisson distribution	365
16.7	Multivariate probability distributions	366
16.7.1	Multi-categorical distribution	366
16.7.1.1	Encoding the category by 1-hot encoding	366
16.7.1.2	Categorical distribution	367
16.7.1.3	Expectation, variance and covariance of the categorical distribution	368
16.7.2	Multinomial distribution	368
16.7.2.1	Expectation, variance and covariance of the multinomial distribution	368
16.7.3	Multivariate Gaussian distribution	368
17 Statistical Inference	372	
17.1	Introduction to Inference	372
17.1.1	Statistics	373
17.1.2	General Questions we ask ourselves in statistical modeling	373
17.2	Examples of Statistical Models - baseline of statistical modeling	374
17.2.1	General form of the models	374
17.2.2	Example I: One- and two-factor univariate analysis; discrete RVs*	374

17.2.3 Example II: Linear regression model	375
17.2.4 Example III: Generalized linear model	375
17.2.5 Example IV: Autoregressive model	376
17.3 Estimation of Model Parameters	376
17.3.1 Properties of estimators	377
17.3.1.1 Bias of an estimator	377
17.3.1.2 Consistency of an estimator	377
17.3.1.3 Examples for bias and consistency at the hand of the mean .	378
17.3.1.4 The naive sample variance estimation is a biased but consistent estimator	378
17.3.1.5 Sampling distribution	379
17.3.1.6 Standard error of an estimator	379
17.3.1.7 Sufficient statistic	380
17.3.1.8 Efficiency of an estimator	380
17.3.1.9 Precision of an estimator	381
17.3.1.10 What characterizes a good estimator?	381
17.3.2 Approaches to Statistical Parameter Estimation	381
17.3.2.1 Least squared error	382
17.3.2.2 Maximum likelihood estimation (MLE)	383
17.3.2.3 Bayesian inference	385
17.4 Hypothesis Testing	389
17.4.1 Basic terms of statistical hypothesis testing	389
17.4.1.1 Around the hypothesis	389
17.4.1.2 Decision-making	390
17.4.1.3 Validity of the decision	390
17.4.1.4 Statements on tests	391
17.4.2 Basic Null Hypothesis Significance Testing Procedure	391
17.4.2.1 Type I and type II error	393
17.4.2.2 Bayesian perspective on hypothesis testing	394
17.4.3 Overview on specific test procedures	394
17.4.4 Hypothesis and statistics for typical tests	395
17.4.5 Simple example for an application case of hypothesis testing	396
17.4.6 Exact tests	397
17.4.6.1 Sign test - testing the direction of change	397
17.4.6.2 Mann-Whitney U test (aka Wilcoxon rank-sum test) - no strict pairing necessary	398
17.4.7 Asymptotic test	399
17.4.7.1 Central limit theorem	399

17.4.7.2	Important distributions in testing I: χ^2 distribution	400
17.4.7.3	Important distributions in testing II: t -distribution	401
17.4.7.4	The Chi-Square Test - test on counts	401
17.4.7.5	Students t -test	402
17.4.7.6	Important distributions in testing III: F -distribution	404
17.4.7.7	The F -test - test on variances	404
17.4.7.8	Likelihood ratio test principle - comparing models by likelihood	405
17.5	Bootstrap methods	407
17.5.1	Caveats of Bootstrap*	408
17.5.2	Bootstrap Confidence Intervals*	409
17.5.3	Standard Normal Bootstrap Confidence Interval	409
17.5.3.1	Percentile Bootstrap Confidence Interval*	410
17.5.4	Bootstrap Hypothesis Testing	410
17.5.4.1	1-sample test as an introductory example	411
17.5.4.2	Calculation of p values in Bootstrap Hypothesis Testing . . .	413
17.5.4.3	Bootstrap Hypothesis Test on the Difference of Means	413
17.5.4.4	Bootstrap Hypothesis Test - Are two samples from different distributions?	414
17.6	Multiple testing problem significance by chance	415
18	Numerical Methods for Parameter Estimation	416
18.1	Overview on strategies for parameter estimation	416
18.2	Overview on numerical methods for minimizing the loss (in the MLE setting)	417
18.3	Gradient Descent	418
18.3.1	Improvements to standard gradient descent	418
18.4	Newton-Raphson	419
18.4.1	Standard Newton iteration for root finding	419
18.4.2	Newton's method in optimization	420
18.4.2.1	Advantages of Newton optimization	421
18.4.2.2	Disadvantages of Newton optimization	421
18.5	Numerical Parameter Estimates in a Bayesian Setting	423
18.5.1	Maximum A Posteriori (MAP) Estimation	423
18.5.2	A-posteriori Expectation	423
19	Regression	425
19.1	Multiple Linear Regression the standard	426
19.1.1	Model setup what we model how under which assumptions	426
19.1.1.1	Assumptions	427

19.1.1.2 Matrix notation	428
19.1.2 Two views on finding an estimator for $\underline{\beta}^*$	428
19.1.3 Finding the model parameters minimizing the sum of squared errors .	429
19.1.4 How is $\hat{\underline{\beta}}$ distributed? - normally	430
19.1.4.1 Intuition for the dependence of the distribution of $\hat{\underline{\beta}}$ on the Distribution of the explanatory variables	431
19.1.4.2 Exact form of the normal distribution of $\hat{\underline{\beta}}$	431
19.1.4.3 Distribution of $\hat{\underline{\beta}}$	433
19.1.4.4 Confidence intervals for $\underline{\beta}$ - where is the true $\underline{\beta}^*$?	434
19.1.4.5 Usage in hypothesis testing	435
19.2 General linear model	435
19.2.1 Purely categorical predictors	435
19.2.2 Testing if categories have the same mean	436
19.3 Multivariate linear model multiple outcomes	437
19.3.1 Multivariate model	437
19.3.2 Except for possible correlations in the error terms, it's just a concate- nated model	437
19.3.3 The error matrix E can contain correlations	438
19.3.4 Distribution of $\hat{\underline{B}}$	439
19.3.5 Hypothesis testing in the multivariate case - considering parameters jointly	439
19.4 Regression Models Suitable for Non-Linear Relationships	440
19.4.1 Regression with Feature Engineering	442
19.4.2 Spline Regression	443
19.4.3 Local Linear Regression	445
19.4.3.1 Model and parameter estimates in local linear regression . .	445
19.4.3.2 Choosing the bandwidth λ - the hyperparameter of local lin- ear regression	446
19.4.4 Generalized Linear Models (GLM)*	447
19.4.4.1 Repetition on Linear Regression	447
19.4.4.2 Generalized Linear Models	448
19.4.4.3 Finding the parameters $\underline{\beta}$ for a GLM	449
19.4.4.4 Canonical Link Functions for a Bernoulli GLM - Logistic Regression	449
19.4.4.5 Complementary Log-Log Link Function	449
19.4.5 Gaussian Process Regression*	451
19.4.5.1 Weight-space view - generalization of linear regression . . .	451

20 Bias-Variance Tradeoff and dealing with model complexity	453
20.1 Bias-Variance-Tradeoff	453
20.1.1 Model Bias and Variance	453
20.1.2 Bias-Variance Tradeoff	454
20.2 Measuring generalization: Train and Test error and their relation to Bias and Variance	455
20.2.1 Bias-Varianc-Decomposition of the (Mean-Squared) prediction error	455
20.2.1.1 Definition of the prediction error	455
20.2.2 Bias-Varianc-Decomposition of the prediction error	455
20.2.3 Derivation of the Bias-Varianc-Decomposition	456
20.2.4 Striking a balance between Bias and Variance	457
20.2.4.1 Effect of more training data on the balance	457
20.3 Finding good hyperparameters partitioning, cross-validation and analytical model selectors	458
20.3.1 Choosing hyperparameters if we are rich in training data - partitioning (train-test-validation-split)	458
20.3.2 Choosing hyperparameters if we are poorer in training data - K-folds Cross-Validation	459
20.3.2.1 K-Fold for estimating the prediction error	459
20.3.2.2 K-Fold for choosing hyperparameters	460
20.3.3 Analytical Model Selectors	461
20.4 Breakpoint of Bias-Variance Tradeoff - double descent	462
20.5 Curse of Dimensionality in Modeling*	463
20.5.1 Curse of Dimensionality in Modeling / Sampling	463
20.5.2 Example for Curse of Dimensionality in number of possible models*	464
20.6 Overview on dealing with complexity	465
20.7 Feature selection - best subset selection	465
20.8 Regularization I General idea and pre-step	465
20.8.1 Pre-step to regularization - centering, standardizing	465
20.8.1.1 Centering and standardizing (z-scoring)	466
20.9 Bayesian perspective on Regularization	466
20.10 Regularization II Regularized regression	467
20.10.1 Ridge Regression	467
20.10.1.1 Influence of λ and Bias-Variance Tradeoff	468
20.10.1.2 Bayesian perspective on Ridge regression	468
20.10.1.3 Discussion of the ridge regression	469
20.10.2 On choosing the penalty	469
20.10.3 Using the L_0 norm (under $0^0 = 0$)	470

20.10.4 Lasso Regression	471
20.10.4.1 Penalty in the Lasso - L_1 norm	471
20.10.4.2 Why does Lasso drive coefficients to exactly zero? Soft-thresholding of the coefficients in Lasso regression	472
20.10.4.3 Influence of λ on the coefficients in Lasso regression	473
20.10.5 Why does Lasso regression lead to feature selection (<i>induces sparsity</i>) and Ridge regression does not?	476
20.10.5.1 Constraint perspective and regularized regression	476
20.10.5.2 Intuition for the difference in feature selection	476
20.11 Preview: Fighting overfitting and improving performance in Neural Networks: Dropout	477
21 Classification	478
21.1 Overview on approaches to classification	478
21.2 Discriminant analysis	479
21.2.1 Linear discriminant analysis (LDA) and Mahalanobis distance	481
21.2.1.1 The LDA decision surface is linear	482
21.2.2 Estimation of the prior, mean and covariance in discriminant analysis	483
21.2.2.1 Advantages of Discriminant Analysis	484
21.3 Linear Classifiers	484
21.4 Support Vector Machines (SVM)	485
21.4.1 Basic Idea of SVMs - maximum margin classifier	485
21.4.2 The margins - intuition for linearly separable classes	486
21.4.3 Finding β and β_0 specifying the maximum-margin hyperplane	487
21.4.3.1 Formulating the optimization problem	487
21.4.3.2 Reformulating into constraint minimization	487
21.4.3.3 Intermezzo: Primal and Dual Problems in Quadratic Programming*	489
21.4.3.4 Reformulating into a dual problem	489
21.4.4 Relaxing the constraint of linear separability - soft-margin SVM	491
21.4.5 Kernelized SVM	492
21.4.5.1 Kernel trick for high-D computations	493
21.4.5.2 Kernel SVM is where the dual formulation shines	494
21.4.6 Multiclass SVM	496
21.5 K-Nearest Neighbor classifier	496
21.5.1 Dependence of the classification on k - balance between over- and underfitting	496
21.5.2 Advantages and Disadvantages of kNN	497
21.6 Logistic classification	497

21.6.1 GLM perspective on logistic regression	498
21.6.2 Linear classifier perspective on logistic regression	499
21.6.3 Finding the parameters of $\mu(x), \beta$ by MLE	501
21.6.3.1 Understanding the terms in the negative log-loss as penalties	501
21.6.3.2 For linearly separable data, $\ \beta\ \rightarrow \infty$ minimizes the loss .	502
21.6.3.3 The decision boundary in logistic regression is linear	502
21.6.4 Multinomial Logistic Regression	505
21.6.4.1 Perspective of a multicategorical probability model	505
21.6.4.2 Derivation of the class probabilities soft(arg)max	506
21.6.4.3 An engineering style approach to multinomial logistic classification	506
21.6.4.4 Perceptron for binary classification	507
21.7 A primer on neural networks in the context of classification	508
21.7.1 Neural Network Loss for Classification	510
21.7.1.1 Intermezzo: Information measures for probability distributions	511
21.7.1.2 Introduction of Cross-Entropy	511
21.7.1.3 Relation between cross-entropy and the negative log-likelihood - they're the same	512
22 A Primer on Neural Networks*	513
22.1 The final layer of a neural network for classification and regression	513
22.2 Adding hidden layers - path to a neural network	514
22.2.1 On the choice of activation function	515
22.2.2 Neural Network are universal function approximators	516
22.2.3 Neural Networks and Algorithmic modeling	516
22.3 Training of a neural network	517
22.3.1 Problem statement of training NNs	517
22.3.2 Stochastic Gradient Descent for minimizing the loss	517
22.3.2.1 Normal gradient descent	518
22.3.2.2 Stochastic gradient descent	518
22.3.3 How to compute the gradients? backpropagation	518
22.3.3.1 Gradient we want to compute	518
22.3.3.2 Why can't we use finite differencing?	519
22.3.3.3 Backpropagation to the help	519
22.3.3.4 Intermezzo: Automatic differentiation	520
22.3.3.5 Error backpropagation in a neural network	522
22.3.3.6 Problem in training deep networks: Exploding and vanishing gradients	525

22.4 Outlook	526
23 Dimensionality Reduction	527
23.1 Principal Component Analysis (PCA)	529
23.1.1 Deriving the principal components from finding the direction with maximum variance	529
23.1.1.1 The principal components and dimensionality reduction . . .	530
23.2 Nonlinear Dimensionality Reduction I: Nonlinear PCA	531
23.2.1 Kernel PCA	534
23.2.1.1 Finding a formulation where the projection to the principal components is only expressed in terms of a kernel function .	534
23.2.1.2 Kernel expression for mean centered projected features . .	535
23.3 Nonlinear Dimensionality Reduction II: Further techniques	536
23.3.1 Linear discriminant analysis for dimensionality reduction of classified data	536
23.3.2 Retaining (local) distance information or neighbor structure	536
23.3.2.1 Global preservation of distance information	537
23.3.2.2 Local preservation of distance information	537
23.3.2.3 Isomap - distance preservation along geodesics	538
23.3.2.4 Locally Linear Embedding (LLE)	538
23.3.2.5 t-SNE (t-distributed stochastic neighbor embedding)	539
23.3.3 Autoencoders	539
24 Latent Variable Models	540
24.1 The Cocktail Party Problem - Blind Source Separation	540
24.2 Independent Component Analysis (ICA)	542
24.2.1 Intuition	542
24.2.2 Formalization and aim	544
24.2.3 Pre-step: Whitening of the observed data	544
24.2.4 Minimizing dependence of the sources	545
24.3 Factor Analysis	547
24.3.1 Going deeper than relating observed data - in search of the unobserved	547
24.3.2 Setting of Factor Analysis	547
24.3.3 Deriving the parameters in Factor Analysis by Maximum Likelihood .	548
24.4 Generative Latent Models	549
24.4.1 Introduction and problem of the intractable marginal likelihood of the data	549
24.4.2 Evidence Lower Bound (ELBO)	550

24.4.2.1	Proof of the ELBO	551
24.4.2.2	Gap between evidence and ELBO	553
24.4.3	Expectation-Maximization (EM) algorithm	554
24.4.4	Application of the EM algorithm: Gaussian Mixture Models (GMM)	555
24.4.5	Variational Inference	558
24.4.5.1	Recapitulation on parameter estimation parameters vs. latent variables	558
24.4.5.2	The general problem of inference: Sampling vs. Variational approach	559
24.4.5.3	Variational inference	561
24.4.5.4	Mean-Field Algorithm	561
24.4.6	Variational Autoencoder (VAE)	561
24.4.6.1	Back to the problem of likelihood optimization under latent variables	561
24.4.6.2	The Variational Autoencoder	562
24.4.6.3	Gradient of the ELBO	563
References		569

1 Introduction

Add / refer to example applications

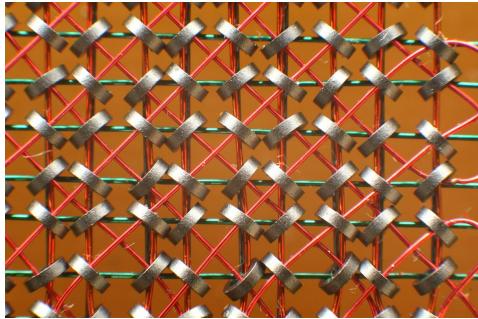
Computational physics encompasses

- simulation as a new paradigm to approach physical problems and validate theories by comparing simulated results to experiments
- statistics and data analysis to make sense of experimental or simulated data
- scientific machine learning to incorporate physical knowledge into Machine Learning and Machine Learning into simulations

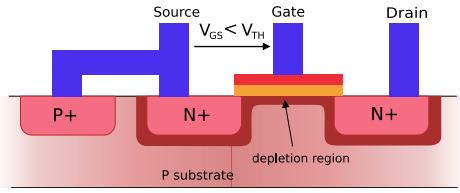
This books aim is to give an introduction to Computational Physics with the presented concepts often also being of great use in other fields (e.g. solving partial differential equations computationally is not only greatly important for physics but for instance also for solving the Black-Scholes equation in finance).

Content included mainly encompasses the lectures

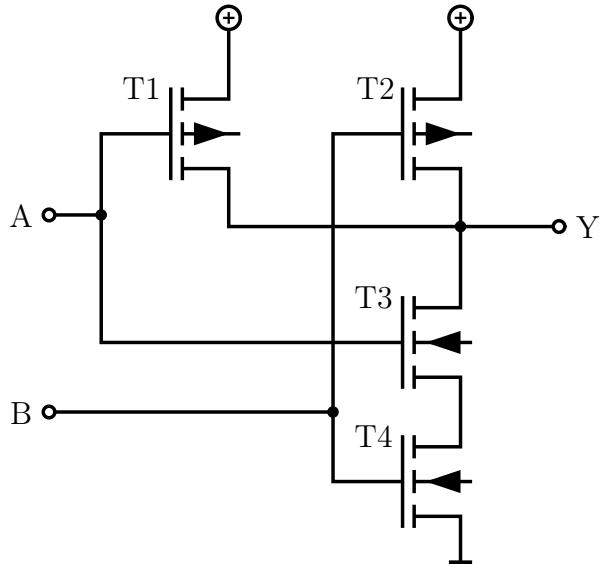
- Fundamentals of Simulation Methods (Ralf Klessen and Philip Girichidis, 2023)
- Computational Statistics and Data Analysis (Daniel Durstewitz, 2023)



(a) Historic Magnetic Core Storage. Bits are stored as the direction of magnetic flux.



(b) Field-effect transistor (more specifically MOSFET schematic). Power-efficiently switching currents is at the heart of modern computing. Based on a Floating-Gate or Charge-Trapping mechanism and the tunnel effect, storage can be realized via stored charges.



(c) NAND-Gate. Bit operations lie at the core of computations.

Figure 1: Basics of Computation.

Part I

Basics of Numerical Computation

How we write what algorithms is informed by how we represent data. While there exist exotic approaches like analog computers that can e.g. handle integration elegantly (Ulmann, 2020) or quantum computing which might e.g. at some point accelerate machine learning (Biamonte et al., 2017), standard binary data representation is vastly prevailing.

Binary data can be very efficiently and reliably stored and operated on (see figure 1), but the respective chosen representations might come with caveats.

2 Digital Representation of Numbers

In C, `int x = 100 * 200 * 300 * 400;` will surprisingly yield -1894967296 (for a 32-bit integer representation) (*overflow*), `float f1 = (2.3 + 1e20) - 1e20;` yields 0.0 (*round-off-error*) but `float f2 = 2.3 + (1e20 - 1e20);` correctly gives 2.3 .

We want to understand and mitigate such caveats of arithmetic on computers, where we want quick calculations while also using as little storage as possible - simulations can quickly become big in storage (e.g. lots of particles).

Further details can be found in Higham, 2002 and Bryant and O'Hallaron, 2011.

2.1 Integer Arithmetic

In C, integers ($\in \mathbb{Z}$) are stored as fixed-size bit-sequences. The respective size dictates a range. C provides both unsigned and signed integer types, with negative numbers in the signed case represented using *two's-complements*.

2.1.1 Unsigned integers

For a bit-vector $\underline{x} = [x_{\omega-1}, x_{\omega-2}, \dots, x_0]$ the unsigned conversion is

$$B2U_\omega := \sum_{i=0}^{\omega-1} x_i 2^i \quad (1)$$

(illustrated in figure 2) and the range is $0, \dots, 2^\omega - 1$. In case of overflow in operations, the overflow is truncated in the most significant bits (see figure 3). As $B2U_\omega[x_{\omega-1}, x_{\omega-2}, \dots, x_0] \bmod 2^k = B2U_\omega[x_{k-1}, x_{k-2}, \dots, x_0]$ we effectively store arithmetic result $\bmod 2^\omega$.

$x_{\omega-1}$ is called most significant bit (MSB) and x_0 least significant bit (LSB).

Problem: When the result of an arithmetic operation exceeds the range of an integer type, unexpected results occur (overflow) (and also underflow in the signed case)

unsigned char in C

$$= 2^7 + 2^5 + 2^3$$

$$= 168$$

Figure 2: Example of an unsigned char in C.

Why does unsigned char $x = 168 + 96$ represent 8?

\underline{x}_A with $B2U_8(\underline{x}_A) = 168$

1	0	1	0	1	0	0	0
---	---	---	---	---	---	---	---

\underline{x}_B with $B2U_8(\underline{x}_B) = 96$

$+$	0	1	1	0	0	0	0
$B2U_8($	0	0	0	0	1	0	0

$= 8$

$(B2U_8(\underline{x}_A) + B2U_8(\underline{x}_B)) \bmod 2^8$

Figure 3: Example of unsigned char overflow.

2.1.2 Two's complement for negative numbers

Note that addition of integers by bitwise addition with carry-on is very fast.

Why can't we just let the MSB encode the sign?: A representation of the form

$$B2S_\omega(\underline{x}) := (-1)^{x_{\omega-1}} \cdot \sum_{i=0}^{\omega-2} x_i 2^i \quad (2)$$

(sign magnitude) has the disadvantage, that normal bitwise addition with carry-on does not work. Also, zero is encoded twice, as ± 0 .

Idea of the two's-complement: The MSB flags the sign in $\underline{x} = [x_{\omega-1}, x_{\omega-2}, \dots, x_0]$ by having a weighting factor $-2^{\omega-1}$

$$B2T_\omega(\underline{x}) := -x_{\omega-1} 2^{\omega-1} + \sum_{i=0}^{\omega-2} x_i 2^i \quad (3)$$

Carry-on to the ω -th bit in addition is again ignored. As we really just have to add positive numbers in bits x_0 to $x_{\omega-2}$ with correct sign-switch by carry-on, no special handling is necessary (see figure 4). The range is $-2^{\omega-1} \dots 2^{\omega-1} - 1$.

From an unsigned int to the two's complement and vice versa we can get by

1. invert all bits
2. add $+1$ to the result¹

If we want \underline{x} with $B2T_\omega(\underline{x}) = -u$ then the bits following the MSB must encode k with $-u = -2^{\omega-1} + k$, so the encoded unsigned number must be $B2T_\omega(\underline{x}) = \underbrace{2^{\omega-1}}_{\text{sign bit}} + k = 2^\omega - u - 1$

a *two's complement*.

¹These rules intuitively follow from the constraint, that bitwise addition with carry-on of $-u$ and u should result in an all-zero bitvector. Adding a bitvector to its inverted self results in an all-1 bitvector, adding one more then results in all zeros, as the last carry-on is discarded.

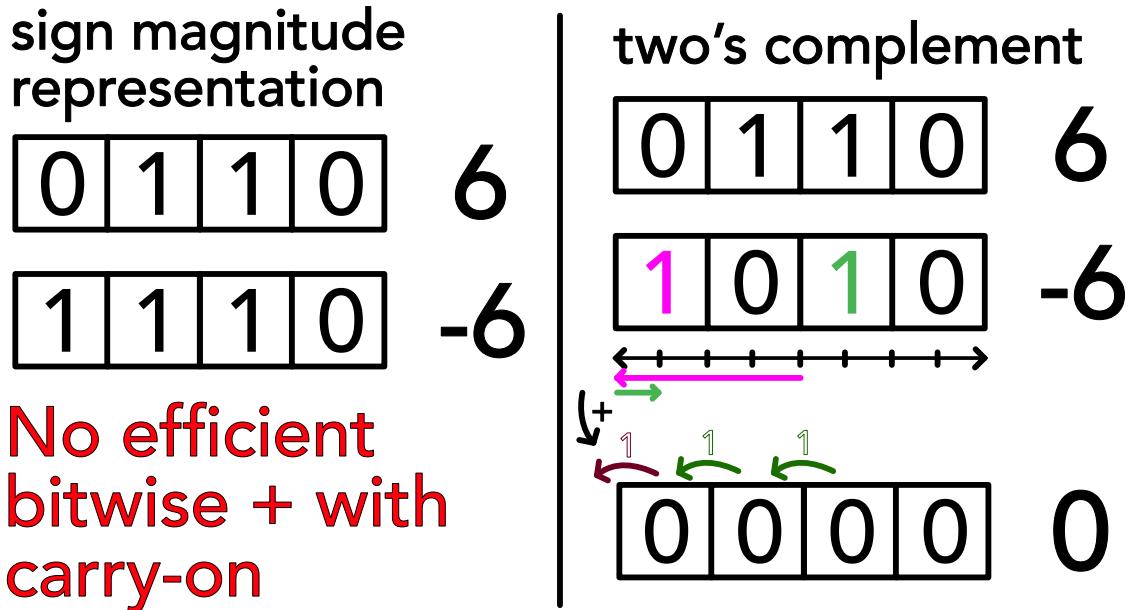


Figure 4: Illustration of the defining feature of the two's complement - addition is simple.

2.1.3 Integer types in C

Some common integer types with respective minimum sizes are given in table 1.

Type	Minimum Size ω
<code>char</code>	8 bits
<code>short</code>	16 bits
<code>int</code>	16 bits
<code>long</code>	32 bits
<code>long long</code>	64 bits

Table 1: Common integer types in C, signed ranges are $-2^{\omega-1} \dots 2^{\omega-1} - 1$, unsigned ranges are $0 \dots 2^\omega - 1$.

Problem: Using unsigned can come with unexpected results, when being cast from a negative number. E.g. `unsigned int x = -1;` will result in $2^{32} - 1$ in a 32-bit system (the two's complement is read as if an unsigned representation). More deviously, $-1 < 0U$ will evaluate to false, as all integers in a comparison are cast to unsigned, when one of them is unsigned.

2.1.4 Byte Ordering in Storage: Big and LittleEndian

Bytes of e.g. a multi-byte integer can be stored from most significant byte to least significant byte (big endian) or vice versa (little endian), see figure 5.

byte ordering

indicates hex
 consider int $x = \overbrace{0x01}^{\text{byte}} \overbrace{23}^{\text{byte}} \overbrace{45}^{\text{byte}} \overbrace{67}^{\text{byte}} = 0 \times 16^7 + 1 \times 16^6 + \dots = 19088743$ with pointer $\&x = 0x100$

big endian (most significant byte first)

adress	0x100	0x101	0x102	0x103		
value	...	MSB 01_{16} 0000 0001	23_{16} 0010 0011	45_{16} 0100 0101	67_{16} 0110 0111	LSB ...

little endian (least significant byte first)

adress	0x100	0x101	0x102	0x103		
value	...	67_{16} 0110 0111	45_{16} 0100 0101	23_{16} 0010 0011	MSB 01_{16} 0000 0001	...

ARM chips can operate with both, iOS and Android use little endian.

Figure 5: Big and Little Endian.

2.1.5 Properties and Caveats of Integer Arithmetic

Typical problems in integer arithmetic are overflow, integer division, the modulo operation and implicit type conversion.

Overflow: The respective integer ranges have finite ranges, overflow in arithmetic operations is cut-off. We must choose a type with sufficient range - mind that choosing types with too large of a footprint (e.g. always long) wastes storage and compute.

Example I: In `char c = 100 * 4; // range -128 to 127` the result is -112 as 400 in bits is 000110010000 where a cut-off to one byte means 10010000 so $-2^7 + 2^4 = -128 + 16 = -112$.

Example II: For `int a = -1 * pow(2,31); int b = 10;` we have $a < b$ but not $a - b < 0$ (for char the behavior is a bit different as up to int there is implicit type conversion.)

Integer division: All decimal places are truncated, so

$$5/3 = 1; -5/3 = -1; 5/-3 = -1$$

Modulo operation: The modulus in C is defined as $n \% m = n - (n/m)m$ so

$$5\%3 = 2; -5\%3 = -2; 5\%-3 = -2$$

Implicit type conversion: In C, the type can implicitly be converted up to unsigned int, which can avoid overflow, as illustrated in table 2.

implicit type conversion up to int can be helpful	mind its only up to int	explicit type conversion
<pre> 1 char a,b; 2 a = 100; 3 b = 4; 4 int c = a * b; // ↵ 400 </pre>	<pre> 1 int a = 2e9; 2 int b = 3; 3 long long c = a * ↵ b; 4 printf("%llu\n", ↵ c); // ↵ 1705032704 </pre> <p>As $2^{32} - 1 \approx 4 \cdot 10^9 < 6 \cdot 10^9 < 2^{33} - 1$ (unsigned ranges) we overflow to the 33rd-bit, which is cut-off, giving the unsigned result of $6 \cdot 10^9 - 2^{32} = 1705032704$.</p>	<pre> 1 int a = 2e9; 2 int b = 3; 3 long long c = ↵ ((long long) a) ↵ * ((long long) ↵ b); 4 printf("%llu\n", ↵ c); // ↵ 6000000000 </pre>

Table 2: Type conversion and its caveats

2.1.6 Can there be integer-overflow in python?

Note that in python3, integers are implemented as “long” integer objects of arbitrary size, overflows are impossible (at the cost of speed; mind that e.g. numpy is based on C code).

2.2 Floating Point Arithmetic

In the following, we will encode rational numbers in the form $V = x \cdot 2^y$, with $x, y \in \mathbb{Z}$. Similar to the decimal notation

$$d_m d_{m-1} \dots d_0.d_{-1} d_{-2} \dots d_{-n} = \sum_{i=-n}^m d_i 10^i \quad (4)$$

we can write in binary notation

$$b_m b_{m-1} \dots b_0.b_{-1} b_{-2} \dots b_{-n} = \sum_{i=-n}^m b_i 2^i, \quad 0.01_2 = 0.25_{10} \quad (5)$$

or generally in base β in scientific notation

$$(-1)^s \cdot \underbrace{b_0.b_{-1} b_{-2} \dots b_{-n}}_{\text{mantissa } M} \cdot \beta^e = \beta^e \cdot \sum_{i=-n}^0 b_i 2^i, \quad \begin{aligned} &\text{exponent } e, \\ &\text{precision (number of digits) } p = n + 1, \quad \text{sign-bit } s \in \{0, 1\} \end{aligned} \quad (6)$$

Note that in this notation (in the form $V = x \cdot 2^y$) we cannot exactly represent e.g. 0.1 or 0.2.

$$0.1_{10} = 1.10011[0011] \dots_2 \cdot 2^{-4} \quad (7)$$

Disastrous historic example: In the first Gulf War (more specifically on 25th February 1991), a Patriot missile defense battery failed to intercept an incoming Iraqi Scud missile, because it used an internal clock counting up in tenths of a second represented by a 23-bit sequence. Future missile positions are predicted by extrapolating from past position with constant velocity. The Patriot system mixed both this inaccurate internal clock and a more accurate one, leading to the failed interception and 28 deaths among soldiers.

2.2.1 IEEE 754 Floating Point Standard

Based on the representation in scientific notation, we can store a floating point number in a bit-vector with

- a sign bit s
- an exponent e stored as an unsigned integer $E = e + b$ with bias b
- a mantissa M

where for single precision (32-bit)

- the exponent is stored in 8 bits, $b = 127$, $e_{min} = -126$, $e_{max} = 127$ with $E = 255$ and $E = 0$ reserved for special cases
- the mantissa is stored in 23 bits, with the first bit being implicitly 1 (**normalization**), which can always be assumed by appropriately choosing the exponent (floating point representations are not unique), so we have $p = 23(+1)$ bits of precision encoding an integer M but need a special representation for 0

where based on the exponent we differentiate between

- **normalized values for $1 \leq E \leq 254$** with value of the floating point number

$$V = (-1)^s \cdot \left(1 + \frac{M}{2^p}\right) \cdot 2^{E-b}, \quad \begin{array}{l} \text{sign bit } s \\ \text{biased exponent } E = e + b, \quad \text{integer representation of mantissa } M \\ \text{precision } p = \#\text{mantissa bits} + 1 \text{ implicit bit} \end{array} \quad (8)$$

- **denormalized values for $E = 0$** with value of the floating point number

$$V = (-1)^s \cdot \frac{M}{2^p} \cdot 2^{-b+1}, \quad \text{for } M = 0, V = \pm 0 \text{ depending on } s \quad (9)$$

The denormalized numbers start just below the normalized ones (by the factor 2^{-b+1} with $+1$ as we do not have the implicit leading 1 here) and now no normalization (implicit starting 1-bit) is assumed. This allows to approach zero with gradually decreasing precision (and even spacing) (smaller numbers occupy fewer digits as of the leading zeros) and ensures that for $x \neq y$, $x - y$ is non-zero, so $\frac{1}{x-y}$ is safe for $x \neq y$.

- **$\pm\infty$ for $E = 255$ and $M = 0$**
- **NaN (Not a Number) for $E = 255$ and $M \neq 0$**

Why is the exponent stored in a biased way, not two's complement?: In a two's complement representation of the exponent to compare two numbers, we have to compare the exponents and mantissas separately and the exponent-comparison is a bit more complicated than comparing biased representations. In the biased exponent representation we can just compare the bitvectors of exponent followed by mantissa interpreted as integers (also mind the sign).

The cases are illustrated in figure 6, with a specific example in figure 7.

1. Normalized



2. Denormalized



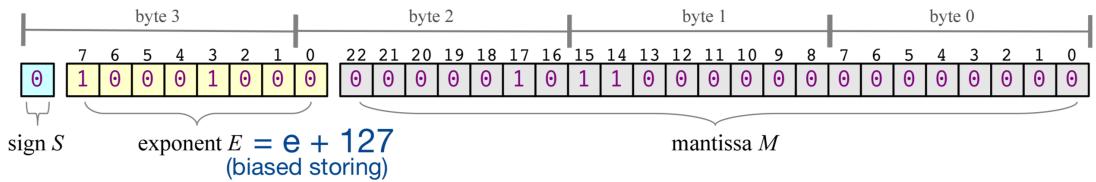
3a. Infinity



3b. NaN



Figure 6: Cases of the IEEE 754 Floating Point Standard.

Figure 7: 523 can be written as $1.000001011_2 \cdot 2^9$ so $e = 9$, $E = e + 127 = 136$. As of the normalization, the leading 1 in the mantissa is implicit. The number is given as a single precision float in big endian.

2.2.2 Only a finite set of floating point numbers can be represented exactly

With 32 bits, 2^{32} states can be encoded (and mind that here e.g. NaN has multiple representations). In any case, the number of exactly representable numbers is finite, above 0 starting at

$$V_{\text{denorm},\min} = \frac{1}{2^p} \cdot 2^{-b+1} \underset{\text{single}}{\approx} 1.4 \cdot 10^{-45} \quad (10)$$

p.

with the smallest normalized number being

$$V_{\text{norm},\min} = (1+0) \cdot 2^{-b} \underset{\text{single}}{\approx} 1.2 \cdot 10^{-38} \quad (11)$$

p.

and the largest normalized number being

$$V_{\text{norm,max}} = \left(1 + \frac{2^p - 1}{2^p}\right) \cdot 2^{E_{\max} - b} \underset{\substack{\text{single} \\ p.}}{\approx} 3.4 \cdot 10^{38} \quad (12)$$

2.2.3 Machine Precision is finite

The smallest increment in the mantissa, in $1 + \frac{M}{2^p}$, is the **machine precision**

$$\epsilon_{\text{mach}} = \frac{1}{2^p} \underset{\substack{\text{single} \\ p.}}{\approx} 1.2 \cdot 10^{-7} \quad (13)$$

Consider two floating point numbers $V_1, V_2 > 0$ next to each other

$$V_1 = \left(1 + \frac{M}{2^p}\right) \cdot 2^{E-b}, \quad V_2 = \left(1 + \frac{M+1}{2^p}\right) \cdot 2^{E-b} \quad (14)$$

so their relative difference is bound by

$$\frac{V_2 - V_1}{V_2} = \frac{\frac{1}{2^p} \cdot 2^{E-b}}{\left(1 + \frac{M+1}{2^p}\right) \cdot 2^{E-b}} \leq \epsilon_{\text{mach}} \quad (15)$$

2.2.4 Rounding and Pitfalls of Floating Point Arithmetic

In the IEEE standard, results of addition, subtraction, multiplication and division must equal to one where the arithmetic operations are assumed to be exact and then there is rounding to the nearest representable number (computation is done at higher (typically double or higher) precision). Therefore (mind the section before)

$$\text{relative error } \frac{|x - \hat{x}|}{|x|} \leq \epsilon_{\text{mach}}, \quad \text{number } x, \text{ number on machine } \hat{x} \quad (16)$$

with the common pitfalls

- **Limitation of machine precision:** $a + b = a$ typically for $|b| < \epsilon_{\text{mach}}|a|$, i.e. when b cannot be resolved by the mantissa of a , e.g. $(1 + 0.5\epsilon) - 0.5\epsilon$ in floating point arithmetic yields 0.999999999999999 not 1.
- **Associativity is not guaranteed:** $(a+b)+c \underset{\text{i.A.}}{\neq} a+(b+c)$, e.g. $(2.3+1e20)-1e20$ yields 0.0 but $2.3 + (1e20 - 1e20)$ yields 2.3

- **Problems of representability:** As e.g. 0.1 is not exactly representable in base $\beta = 2$, $x/10 \neq 0.1 \cdot x$ in general while $x/2.0 = 0.5 \cdot x$ is exact. The compiler may automatically choose the multiplication variant as multiplication is faster than division.
- **Cancellation:** For $x = a - b$ subtractive cancellation causes relative errors already present in \hat{a} and \hat{b} to be (relatively) amplified, when a and b are of similar size. Significant digits are lost and the relative error explodes. Consider e.g. $a = 1.75682, b = 1.75471$ with $\hat{a} = 1.76$ and $\hat{b} = 1.75$. While \hat{a}, \hat{b} have small relative errors (3 digits precision²) Note that here 0.123 and 0.127 agree in two significant digits, where one intuitively might say this should be rather 1.), $a - b = 0.00211$ and $\hat{a} - \hat{b} = 0.01$ has a large relative error (no precise digit).
- **NaN and Inf:** All calculations including NaN yield NaN, calculations with inf mostly inf (except of course $1/\inf = 0, \dots$)

where we should

- **Rewrite calculations so that errors do not amplify:** For $x = 10^8, y = 10^5, z = -1 - 10^5$ we have $xy + xz = -1.0066 \cdot 10^8$ (problem in resolving the -1 in xz) but $x(y + z) = -1.0 \cdot 10^8$.
- **Compare floats based on a maximum relative error:** Instead of $x == y$ we should use e.g. $|x - y| \leq \epsilon_{\text{mach}} \cdot \max(|x|, |y|)$.
- **As avoid overflow to inf and nan:** E.g. in `float x = 1e20; float y = x * x; float z = y / x` z will be inf.

2.2.5 Rewriting Expressions to Avoid Cancellation I

Consider $f(x) = \frac{1-\cos x}{x^2}$. For $x_e = 10^{-4}$, we have (when we represent 6 figures)

$$c := \cos x_e, \quad \hat{c} = 0.999999, \quad 1 - \hat{c} = 10^{-6}, \quad \text{while } 1 - c \approx 5 \cdot 10^{-9} \quad (18)$$

$1 - c$ has only one significant digit, the relative error is enormously amplified and we get

$$\frac{1 - \hat{c}}{x_e^2} = 100, \quad \text{while in reality for } x \neq 0 : 0 \leq f(x) \leq \frac{1}{2} \quad (19)$$

² \hat{x} is said to approximate x to the r-th digit, if the absolute error is at most $\frac{1}{2}$ in the r-th digit, so

$$\text{largest integer } s \text{ so that } 10^s < |x|, \quad |x - \hat{x}| < 1/2 \cdot 10^{s-r+1} \quad (17)$$

To avoid cancellation we can rewrite using $\cos x = \cos^2 \frac{x}{2} - \sin^2 \frac{x}{2} = 1 - 2 \sin^2 \frac{x}{2}$ to

$$f(x) = \frac{1}{2} \left(\frac{\sin \frac{x}{2}}{\frac{x}{2}} \right)^2 \quad (20)$$

A comparison of both versions in python can be found in figure 8, at some point (roughly below 10^{-8}), $\cos x$ is too close to 1 to be resolved by the mantissa, so the total result is 0, above that the magnified error is visible.

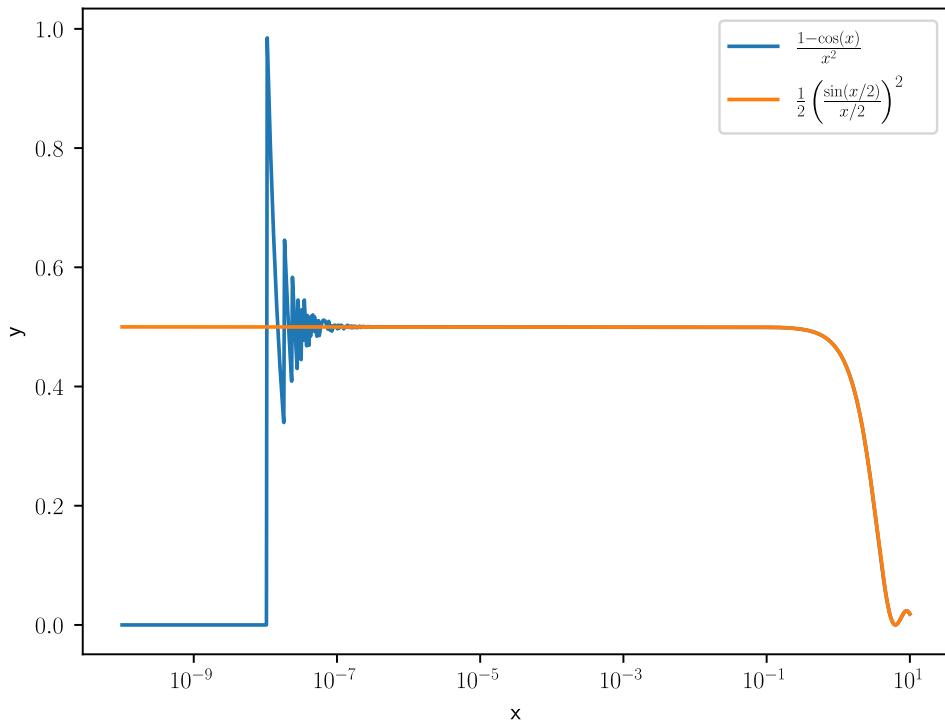


Figure 8: Comparison of the mathematically equivalent expressions $f(x) = \frac{1-\cos x}{x^2}$ and $f(x) = \frac{1}{2} \left(\frac{\sin \frac{x}{2}}{\frac{x}{2}} \right)^2$ in python.

2.2.6 Rewriting Expressions to Avoid Cancellation II

Consider the following expressions for the sample variance of $\{x_i\}_{i=1}^N$

$$\begin{aligned} \text{two-pass formula: } \bar{x} &= \frac{1}{N} \sum_{i=1}^{N-1} x_i, \quad \sigma_N^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 \\ \text{one-pass formula: } \sigma_N^2 &= \frac{1}{N-1} \left(\sum_{i=1}^N x_i^2 - \frac{1}{N} \left(\sum_{i=1}^N x_i \right)^2 \right) \\ \text{as } \sigma^2 &= E[(x - \bar{x})^2] = E[x^2] - E[x]^2 \end{aligned} \quad (21)$$

While for the one-pass formula, we can calculate all necessary sums in one pass through the data, it suffers heavily from cancellation: For $\{10000, 10001, 10002\}$, the two-pass formula in single precision correctly gives 1.0 while the one-pass formula yields 0.0 (cancellation) (there are better one-pass formulas though).

2.2.7 Accumulation of Round-off Errors

Consider the sum

$$\sum_{k=1}^{\infty} k^{-2} = \frac{\pi^2}{6} \quad (22)$$

which we want to approximate by finitely many summands. If we sum the terms just as the formula suggests from large to small, at some point, the small changes will not be resolved anymore - so better sum up from small to large. Summing up in single precision for $N = 10^7$ terms, one gets

$$\begin{aligned} \text{big to small: } &1.644725323, \quad \text{small to big: } 1.644933939, \\ &\text{exact (till 9th digit): } 1.644934058 \end{aligned} \quad (23)$$

As expected the big-to-small summation is too small.

2.2.8 Higher Precision

The above pitfalls are less severe in higher precision. For instance in double precision (64-bit)

$$\begin{aligned} p = 52(+1) \text{ mantissa bits, } \quad 11 \text{ exponent bits, with } e_{min} = -1022, e_{max} = 1023, \\ \text{smallest and largest repr. numbers } f_{min} \simeq 2.2 \cdot 10^{-308}, f_{max} \simeq 1.8 \cdot 10^{308}, \\ |e_{min}| < |e_{max}| \rightarrow \frac{1}{f_{min}} < f_{max} \end{aligned} \quad (24)$$

with machine precision $\epsilon_{\text{mach}} \approx 2.2 \cdot 10^{-16}$. There is even quad-double precision (128-bit) (not supported on hardware though and therefore relatively slow as it has to be emulated). Packages for nearly arbitrary precision also exist.

2.3 A more general view on sources of numerical error

In numerical computation, the typical error sources are

- rounding
- data uncertainty
- truncation (of terms in numerical schemata, e.g. in the approximation of a function by its Taylor series)

where we have now discussed rounding errors and their effects to some extent.

2.4 Backward error, forward error and condition number

Consider we approximate $y = f(x)$ as \hat{y} in an arithmetic of limited precision, with $f : \mathbb{R} \rightarrow \mathbb{R}$.

- **Forward error:** The absolute or relative error between \hat{y} and y is the forward error (living in the output space)
- **Backward error:** The backward error is the smallest Δx (in the input space) so that $f(x + \Delta x) = \hat{y}$, so the smallest perturbation where the exact function gives our approximate result.

If this Δx is sufficiently small, e.g. as small as the uncertainty in the data in the first place, we speak of backward stability. A weaker formulation is the mixed forward-backward error

$$\hat{y} + \Delta y = f(x + \Delta x), \quad |\Delta y| \leq \epsilon |y|, \quad |\Delta x| \leq \eta |x| \quad (25)$$

In the context of rounding errors, we call the algorithm numerically stable if \hat{y} is almost the right answer for almost the right data (ϵ, η small).

2.4.1 Conditioning

Backward and forward error are connected by the conditioning of a problem, the sensitivity of the solution to perturbations in the data. Assuming $\hat{y} = f(x + \Delta x)$ and f differentiable, we have

$$\hat{y} - y = f(x + \Delta x) - f(x) = f'(x)\Delta x + \mathcal{O}((\Delta x)^2) \quad (26)$$

so the relative error is

$$\frac{\hat{y} - y}{y} = \frac{f'(x)\Delta x}{f(x)} + \mathcal{O}((\Delta x)^2) \quad (27)$$

leading to the relative condition number

$$\kappa(x) \underset{\text{i.A.}}{\sim} \lim_{\epsilon \rightarrow 0^+} \sup_{\|\Delta x\| \leq \epsilon} \frac{\|y - \hat{y}\|/\|y\|}{\|\Delta x\|/\|x\|} = \left| \frac{x f'(x)}{f(x)} \right| \quad (28)$$

for small Δx measuring the relative change in the output over a relative change in the input. As a rule of thumb

$$\text{forward error} \lesssim \text{condition number} \cdot \text{backward error} \quad (29)$$

so ill-conditioned problems can have large forward errors.

Application on Matrices

Consider the linear system $\underline{\underline{A}}\underline{y} = \underline{x}$, $\underline{y} = \underline{\underline{A}}^{-1}\underline{x}$, $\hat{\underline{y}} = \underline{\underline{A}}^{-1}(\underline{x} + \underline{\Delta x})$. The condition number follows as

$$\begin{aligned} \kappa(\underline{\underline{A}}) &= \max_{\underline{x}, \underline{\Delta x} \neq 0} \frac{\|\underline{\underline{A}}^{-1}\underline{x} - \underline{\underline{A}}^{-1}(\underline{x} + \underline{\Delta x})\|/\|\underline{\underline{A}}^{-1}\underline{x}\|}{\|\underline{\Delta x}\|/\|\underline{x}\|} \\ &= \max_{\underline{\Delta x} \neq 0} \frac{\|\underline{\underline{A}}^{-1}\underline{\Delta x}\|}{\|\underline{\Delta x}\|} \max_{\underline{x} \neq 0} \frac{\|\underline{x}\|}{\|\underline{\underline{A}}^{-1}\underline{x}\|} \\ &= \max_{\underline{\Delta x} \neq 0} \frac{\|\underline{\underline{A}}^{-1}\underline{\Delta x}\|}{\|\underline{\Delta x}\|} \max_{\underline{y} \neq 0} \frac{\|\underline{\underline{A}}\underline{y}\|}{\|\underline{y}\|} \\ &= \|\underline{\underline{A}}^{-1}\| \cdot \|\underline{\underline{A}}\| \end{aligned} \quad (30)$$

where we used the definition of the matrix norm $\|\underline{\underline{A}}\| = \max_{\underline{x} \neq 0} \frac{\|\underline{\underline{A}}\underline{x}\|}{\|\underline{x}\|}$. For large condition numbers, small perturbations in the input \underline{x} lead to large changes in the solution \underline{y} .

Part II

Simulation Methods

The dynamical evolution of physical systems is described using differential equations. Numerical methods for solving differential equations and the rise of computers have allowed for accurate modeling of complex dynamical systems that could hardly be approached by analytical means even under the usage of perturbation theory (compare Moser, 1978).

Oftentimes, we face an initial value problem (IVP) where from initial values from the functions to solve and values for their derivatives as necessary, the evolution is sought to be calculated. In a boundary value problem a differential equation is given together with a set of additional constraints (e.g. Sturm-Liouville problems).

3 Integration of ordinary differential equations

Our aim is solving an ordinary differential equation (ODE) $\partial_t \underline{y} = \underline{f}(\underline{y})$ with initial values $\underline{y}(t = t_0) = \underline{y}_0$. Notice that $\underline{f} = \underline{f}(\underline{y}, t)$ can be handled by augmenting $\tilde{\underline{y}} = \begin{pmatrix} \underline{y} \\ t \end{pmatrix}$ and $\tilde{\underline{f}}(\tilde{\underline{y}}) = \begin{pmatrix} \underline{f}(\underline{y}) \\ 1 \end{pmatrix}$.

3.1 Notes on ODEs

3.1.1 Converting to a first order system

Ordinary differential equations only contain derivatives with respect to one variable. Note, however, that higher order derivatives with respect to that variable can occur. We can get to the form $\partial_t \underline{y} = \underline{f}(\underline{y})$ by converting to a coupled first order system.

Consider the n-th order ODE

$$\partial_t^n y(t) = f(y(t), \partial_t y(t), \dots, \partial_t^{n-1} y(t), t), \quad f : U \subset \mathbb{R} \times \mathbb{K}^n \rightarrow \mathbb{K} \quad (31)$$

for instance a pendulum with damping

$$\partial_t^2 \phi = -\omega_0^2 \sin \phi - \gamma \partial_t \phi, \quad \gamma, \omega_0 \in \mathbb{R} \quad (32)$$

Now we define the variables

$$u_m = \partial_t^m y(t), \quad m \in \{0, \dots, n-1\} \quad (33)$$

leading to the coupled first order system

$$\partial_t \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-2} \\ u_n \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ f(t, u_0, u_1, \dots, u_{n-1}) \end{pmatrix} \rightarrow \partial_t \underline{u} = \underline{f}(t, \underline{u}) \quad (34)$$

Using ϕ for the angle and $\omega = \partial_t \phi$ for the angular velocity, we can write the pendulum as

$$\partial_t \begin{pmatrix} \phi \\ \omega \end{pmatrix} = \begin{pmatrix} \omega \\ -\omega_0^2 \sin \phi - \gamma \omega \end{pmatrix} \quad (35)$$

3.1.2 Existence and uniqueness of an ODE solution for an initial value problem - Picard-Lindelöf and Lipschitz condition

For the initial value problem $\partial_t \underline{y} = \underline{f}(\underline{y}), \underline{y}(t_0) = \underline{y}_0$ to have a unique solution in the vicinity of (\underline{y}_0, t_0) , i.e. for the change around that point, to uniquely determine the development, this change must be *well-behaved*, \underline{f} must be *Lipschitz-continuous*.

$$\forall (\underline{y}, t), (\underline{z}, t) \text{ in the vicinity of } (\underline{y}_0, t_0) : \|\underline{f}(\underline{y}, t) - \underline{f}(\underline{z}, t)\| \leq \lambda \|\underline{y} - \underline{z}\| \quad (36)$$

with $\lambda > 0$ and $\|\cdot\|$ being an arbitrary vector norm. The slope of the line connecting two close-by evaluations of \underline{f} must be bounded by λ . This is guaranteed for \underline{f} being continuous and sufficiently often differentiable with bounded derivatives and more so \underline{f} analytic.

3.2 Introduction of Numerical Integration at the hand of the two-body problem

Our aim is computationally modelling the interaction of two-bodies. This lends itself well as an example, as stepping the system forward in time is easy to imagine visually, an analytic solution exists to which we might compare numerical solution and it guides us to the problem of conserved quantities and symplectic integrators.

3.2.1 The two-body problem

For the two-body problem (illustrated in figure 9) the equations of motion are

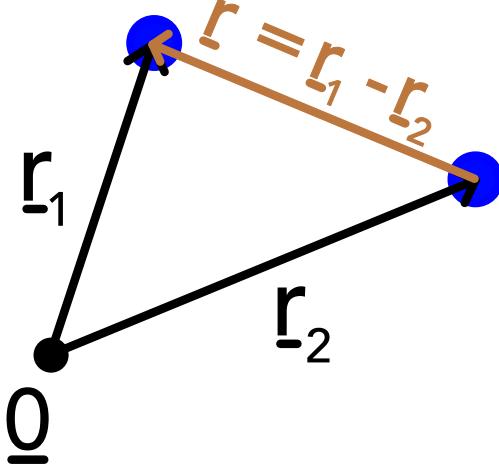


Figure 9: Illustration of the two-body problem.

$$\begin{aligned} m_1 \partial_t^2 \underline{r}_1 &= -G \frac{m_1 m_2}{|\underline{r}|^3} \underline{r} \\ m_2 \partial_t^2 \underline{r}_2 &= +G \frac{m_1 m_2}{|\underline{r}|^3} \underline{r} \end{aligned} \tag{37}$$

for $\underline{r} = \underline{r}_1 - \underline{r}_2$. Subtracting both yields

$$\partial_t^2 \underline{r} = -G \frac{M}{|\underline{r}|^3} \underline{r} \tag{38}$$

with $M = m_1 + m_2$. Which is equivalent to the equation of motion of a single body of mass $\mu = \frac{m_1 m_2}{M}$ in a potential $U(r) = -G \frac{m_1 m_2}{r} = -G \frac{M\mu}{r}$.

We can write this as the first order system

$$\partial_t \begin{pmatrix} \underline{r} \\ \underline{v} \end{pmatrix} = \begin{pmatrix} \underline{v} \\ -G \frac{M}{|\underline{r}|^3} \underline{r} \end{pmatrix} \tag{39}$$

3.2.2 Integrals of Motion

The following quantities are conserved along the trajectories of m_1 and m_2 and are therefore useful sanity checks for simulations.

- Total energy

$$E = T + U = \frac{\mu}{2} \underline{v}^2 - \frac{GM}{r} \mu \tag{40}$$

- Angular momentum (\underline{L} perpendicular to the orbital plane)

$$\underline{L} = \underline{r} \times \underline{p} = \underline{r} \times \mu \underline{v} \quad (41)$$

- Laplace-Runge-Lenz vector (here in its dimensionless form, the eccentricity vector)

$$\underline{e} = \frac{\underline{v} \times \underline{j}}{GM} - \hat{\underline{e}}_r, \quad \text{specific angular momentum } \underline{j} = \frac{\underline{L}}{\mu} \quad \text{eccentricity } e = \|\underline{e}\| \quad (42)$$

Note: The 1-body Kepler problem has 6 degrees of freedom (phase-space coordinates), of which one cannot be conserved, as nothing should be able to tell us the initial time of our motion. Therefore, only 5 quantities can be conserved and the Laplace-Runge-Lenz vector indeed only adds 1 more conserved degree of freedom (taking E and \underline{L} as primary conserved quantities, \underline{e} only has one degree of freedom).

Additional notes on the Laplace-Runge Lenz vector

The Lenz vector is conserved in all $\frac{1}{r}$ -potentials, like the gravitational or Coulomb potential, for instance in the Hydrogen atom (but not for multi-electron atoms). Kepler-orbits are conic sections and the Laplace-Runge-Lenz vector is illustrated in figure 10.

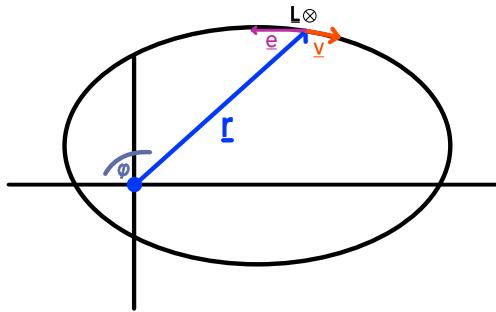


Figure 10: Illustration of the Laplace-Runge-Lenz vector.

From our pictorial evidence, we see that \underline{e} is points along the semi-major axis. Note here we have drawn that $\underline{r} = \underline{r}_1 - \underline{r}_2$ follows a conic section. Likewise m_1 and m_2 move on conic sections with respect to the center of mass, $\underline{0} = \underbrace{\frac{1}{M} (m_1 \underline{r}_1 + m_2 \underline{r}_2)}$ leading to $\underline{r}_1 = \frac{m_2}{M} \underline{r}$ and $\underline{r}_2 = -\frac{m_1}{M} \underline{r}$.

3.2.3 Kepler Orbits are Conic Sections

Depending on the total energy E , we have

- $E < 0 \rightarrow$ (closed) elliptic orbit

- $E = 0 \rightarrow$ parabolic orbit
- $E > 0 \rightarrow$ hyperbolic orbit

This dependence of the orbit form on the energy can be seen from writing

$$E = \frac{\mu}{2}(\partial_t r)^2 + U(r), \quad U(r) = -\frac{\alpha}{r}\mu, \quad \text{here } \alpha = GM \quad (43)$$

and using polar coordinates as the movement takes place on a planar surface

$$(\partial_t r)^2 = (\partial_t r)^2 + r^2(\partial_t \phi)^2 \quad (44)$$

with $\partial_t \phi$ expressed via the conserved angular momentum

$$\text{const. } = l = I\omega = \mu r^2 \partial_t \phi \Rightarrow \partial_t \phi = \frac{l}{\mu r^2} \quad (45)$$

so

$$\begin{aligned} E &= \frac{\mu}{2}(\partial_t r)^2 + U(r) \\ &= \frac{\mu}{2}(\partial_t r)^2 + \frac{\mu}{2}r^2(\partial_t \phi)^2 + U(r) \\ &= \frac{\mu}{2}(\partial_t r)^2 + \underbrace{\frac{l^2}{2\mu r^2}}_{U_{eff}} + U(r) \end{aligned} \quad (46)$$

Note here that we have expressed the total energy as the sum of a kinetic part stemming from a change in distance between the two bodies (which we have already related to the individual positions) and an effective potential U_{eff} . Where the vertical line of constant energy intersects the effective potential, $\partial_t r$ must be zero so such a point must be a point of reversal of movement (see figure 11 and 12).

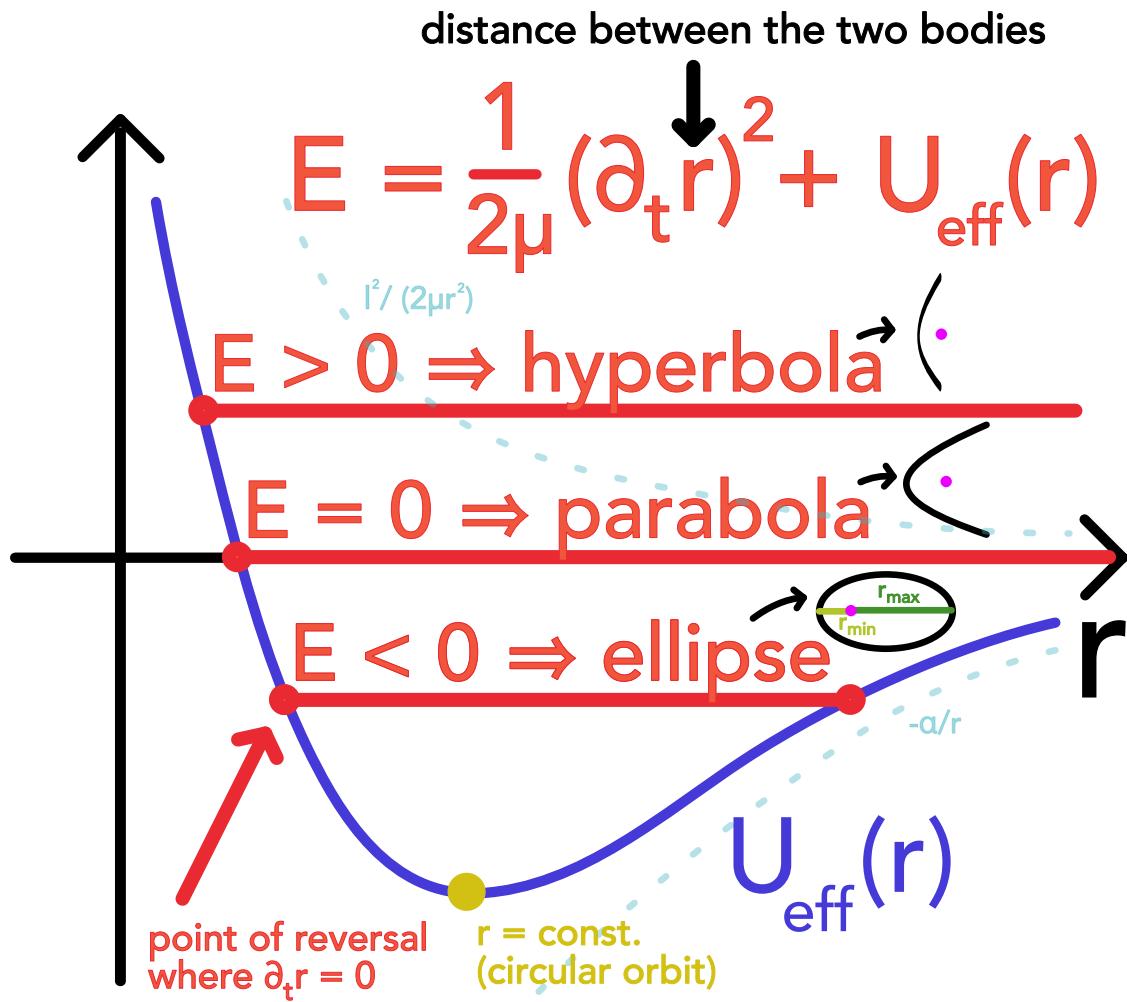


Figure 11: Illustration of the effective potential U_{eff} and the resulting orbits.

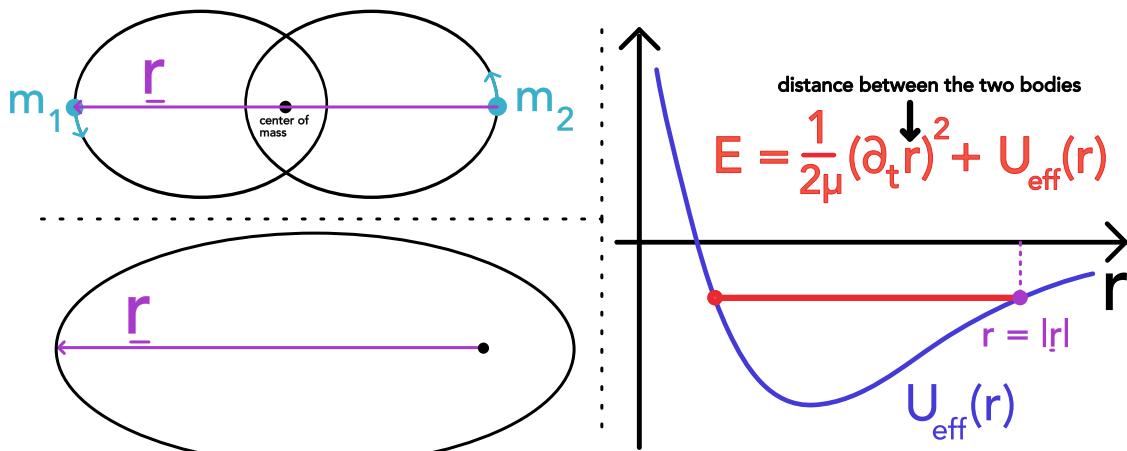


Figure 12: Connection between the different views on the two-body problem.

3.2.4 Connection of the Runge-Lenz vector to the eccentricity of a conic section

From multiplying \underline{e} with \underline{r} we obtain

$$\begin{aligned} \underline{e} \cdot \underline{r} &= er \cos \varphi = \frac{(\underline{v} \times \underline{j}) \cdot \underline{r}}{GM} - r \underset{\underline{a} \cdot (\underline{b} \times \underline{c}) = \underline{b} \cdot (\underline{c} \times \underline{a}) = \underline{c} \cdot (\underline{a} \times \underline{b})}{=} \frac{(\underline{r} \times \underline{v}) \cdot \underline{j}}{GM} - r = \frac{\underline{j}^2}{GM} - r \\ &\rightarrow r(\varphi) = \frac{\underline{j}^2/GM}{1 + \epsilon \cos \varphi} \end{aligned} \quad (47)$$

3.2.5 Rescaling to Dimensionless variables

While the relative precision with which a number is stored on a computer is \sim the machine precision, so independent of magnitude, we rescale our variables so that they predominantly fall into the range $[-1, 1]$ so that different variables are on the same scale (so also their absolute precisions) (also making the problem statement more general).

$$\underline{r} \rightarrow \underline{s} := \frac{\underline{r}}{R_0}, \quad \text{characteristic length } R_0 \text{ e.g. initial separation} \quad (48)$$

$$\underline{v} \rightarrow \underline{w} := \frac{\underline{v}}{v_0}, \quad \text{characteristic velocity } v_0 = \left(\frac{GM}{R_0} \right)^{1/2} \quad (49)$$

v_0 is the velocity, a body circling one with mass M at distance R_0 would have ($F_{zp} = F_G$).

$$t \rightarrow \tau := \frac{t}{T_0}, \quad \text{characteristic time } T_0 = \frac{R_0}{v_0} = \left(\frac{R_0^3}{GM} \right)^{1/2} \quad (50)$$

With this we can write the equation of motion as

$$\frac{d\underline{s}}{d\tau} = \underline{w}, \quad \frac{d\underline{w}}{d\tau} = -\frac{\underline{s}}{|\underline{s}|^3} \quad (51)$$

3.2.6 Solving the two-body problem using explicit (aka forward) Euler

Let us discretize the derivatives with a simple difference quotient, where we probe the current slope by comparing the current position to the one a small time-step in the past or future.

$$\frac{ds^{(n)}}{d\tau} = \frac{\underline{s}^{(n)} - \underline{s}^{(n-1)}}{h} + \mathcal{O}(h)(\text{backwards}) \quad \text{or} \quad \frac{ds^{(n-1)}}{d\tau} = \frac{\underline{s}^{(n)} - \underline{s}^{(n-1)}}{h} + \mathcal{O}(h)(\text{forward}) \quad (52)$$

where $h = \tau^{(n)} - \tau^{(n-1)}$ is the step-size and the *forward* formulation gives an explicit scheme for \underline{s}_n (only depending on already known values)

$$\begin{aligned}\underline{s}^{(n)} &= \underline{s}^{(n-1)} + h \frac{d\underline{s}^{(n-1)}}{d\tau} + \mathcal{O}(h^2) = \underline{s}^{(n-1)} + h \underline{w}^{(n-1)} + \mathcal{O}(h^2) \\ \underline{w}^{(n)} &= \underline{w}^{(n-1)} + h \frac{d\underline{w}^{(n-1)}}{d\tau} + \mathcal{O}(h^2) = \underline{w}^{(n-1)} - h \frac{\underline{s}^{(n-1)}}{|\underline{s}^{(n-1)}|^3} + \mathcal{O}(h^2)\end{aligned}\quad (53)$$

(explicit Euler)

and the *backward* formulation gives an implicit scheme for \underline{s}_n (*implicit* as depending on the yet unknown $\underline{w}^{(n)}$).

$$\begin{aligned}\underline{s}^{(n)} &= \underline{s}^{(n-1)} + h \frac{d\underline{s}^{(n)}}{d\tau} + \mathcal{O}(h^2) = \underline{s}^{(n-1)} + h \underline{w}^{(n)} + \mathcal{O}(h^2) \\ \underline{w}^{(n)} &= \underline{w}^{(n-1)} + h \frac{d\underline{w}^{(n)}}{d\tau} + \mathcal{O}(h^2) = \underline{w}^{(n-1)} - h \frac{\underline{s}^{(n)}}{|\underline{s}^{(n)}|^3} + \mathcal{O}(h^2)\end{aligned}\quad (54)$$

This is also very clear just from first order Taylor expansion.

3.2.7 Probing the accuracy of an integration scheme - energy error of explicit Euler

We probe the accuracy, by checking on the conserved quantities (now dimensionless)

$$\begin{aligned}\text{total energy } E^{(n)} &= \frac{(\underline{w}^{(n)})^2}{2} + \frac{1}{\underline{s}^{(n)}}, & \text{angular momentum } \underline{j}^{(n)} &= \underline{s}^{(n)} \times \underline{w}^{(n)} \\ \text{Laplace - Runge - Lenz vector } \underline{e}^{(n)} &= \underline{w}^{(n)} \times (\underline{s}^{(n)} \times \underline{w}^{(n)}) - \underline{s}^{(n)}\end{aligned}\quad (55)$$

Wanted behavior: In a good integration scheme, the truncation errors (from the Taylor expansion) and rounding errors should be small or at least not accumulate without bound.

We calculate relative errors with respect to the initial values, e.g.

$$\epsilon^{(n)}(h) = \frac{|E^{(n)} - E^{(0)}|}{|E^{(0)}|}$$

Rough error estimation for explicit Euler: Each step has an error of $\mathcal{O}(h^2)$, an orbit takes $\sim \frac{T_0}{\Delta t} = \frac{1}{h}$ steps so we expect an error of $\mathcal{O}(h)$ per orbit, more on the problem of applying *non-symplectic* schemes onto *symplectic* problems follow later.

3.3 Explicit Euler and it's shortcomings

The simplest method for solving an ODE is the Explicit Euler method

$$\underline{y}^{(n+1)} = \underline{y}^{(n)} + f(\underline{y}^{(n)}) \Delta t, \quad \text{where } \underline{y}^{(0)} = \underline{y}_0$$

which is explicit as the computation of $\underline{y}^{(n+1)}$ only depends on already known states.

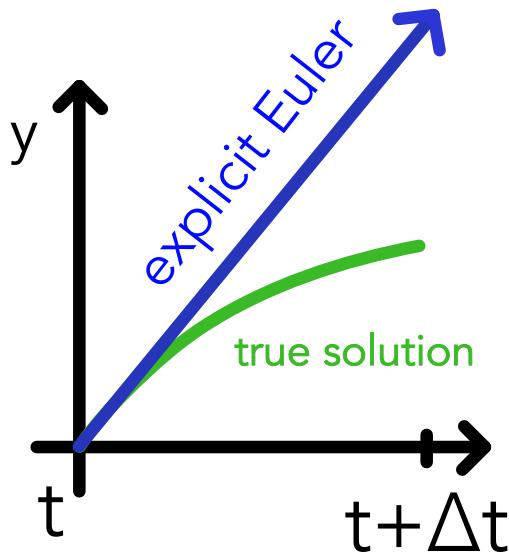


Figure 13: Illustration of one time step in the Explicit Euler scheme.

As illustrated in Figure 13 in every step we step forward along the current derivative $f(\underline{y}^{(n)})$.

3.3.1 Explicit Euler is only first order accurate | truncation error

A simple error approximation follows from Taylor expansion

$$\underline{y}(t + \Delta t) = \underline{y}(t) + \Delta t f(t) + \mathcal{O}_s(\Delta t^2)$$

In each step we make an error $\mathcal{O}_s(\Delta t^2)$ so over some timespan T where we need $N_S = \frac{T}{\Delta t}$ steps we accumulate the error $N_S \mathcal{O}_s(\Delta t^2) = \mathcal{O}_T(\Delta t)$. We therefore call Explicit Euler first order accurate.

Note: For a global error scaling with $\mathcal{O}_T(\Delta t^n)$ (n-th order accurate scheme), the local truncation error (of the Taylor expansion) must be $\mathcal{O}_s(\Delta t^{n+1})$.

3.3.2 Explicit Euler has stability issues

Stability analysis is a broad field, and the interested reader can find details in chapter IV.3 of Hairer and Wanner, 1996. For now, consider the ODE $\partial_t y = \alpha y, \text{Re}(\alpha) < 0, y(0) = y_0$

with the solution $y(t) = y_0 e^{\alpha t}$.

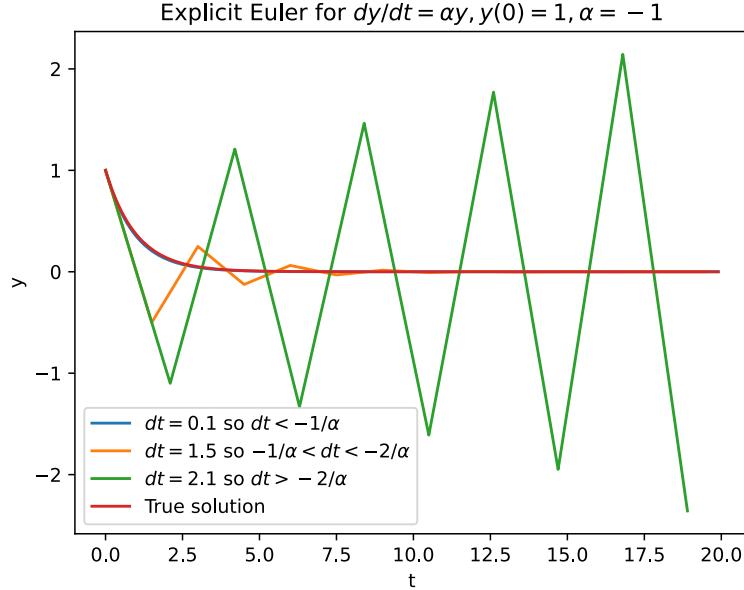


Figure 14: Linear stability of the Explicit Euler scheme.

The results of applying Explicit Euler for different step sizes Δt are shown in figure 14. At a small step size the correct solution is obtained, for a larger step size the numerical solution becomes oscillatory and for even larger step sizes it diverges. We can quantitatively explain this behavior by looking at the Euler steps

$$\begin{aligned} y^{(n+1)} &= y^{(n)} + \alpha y^{(n)} \Delta t \\ &= y^{(n)} (1 + \alpha \Delta t) \\ &= y^{(0)} (1 + \alpha \Delta t)^{n+1} \end{aligned}$$

- $\Delta t < -\frac{1}{\alpha} \rightarrow$ we observe monotonous decrease (ok)
- $-\frac{1}{\alpha} < \Delta t < -\frac{2}{\alpha} \rightarrow$ oscillation (regarding the sign) but still decrease in the absolute value (problematic)
- $-\frac{2}{\alpha} < \Delta t \rightarrow$ an increasing, oscillating solution (very bad)

The growth factor $R(\alpha \Delta t) = 1 + \alpha \Delta t$ in $y^{(n+1)} = R(\alpha \Delta t) y^{(n)}$ is called stability function and

$$\mathcal{D} := \{z \in \mathbb{C} : |R(z)| \leq 1\} \quad \text{so} \quad D_{Euler} = \{z = \alpha \Delta t \in \mathbb{C} : |1 + z| \leq 1\}$$

is called region of absolute stability or linear stability domain. D_{Euler} is a finite region of absolute stability in form of a circle on the left of the complex plane (see figure 15).

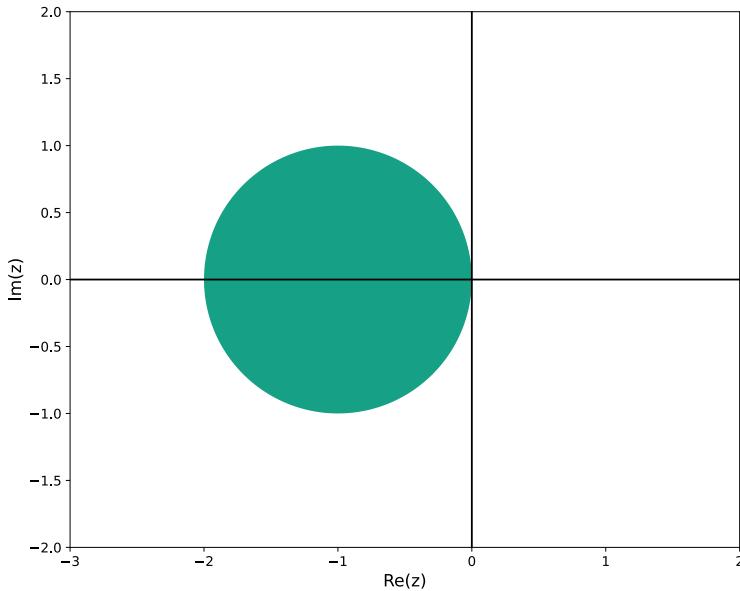


Figure 15: Region of absolute stability of the Explicit Euler method.

Problem: While in this example the stability constraint is easy to fulfill (we get a good solution for a reasonably large step-size), in problems with different timescales, with explicit Euler we must resolve the fastest one, even if its completely negligible (*stiff problems*).

More on stability, A-stable, L-stable, ...

3.4 Introduction of the Problem of Stiffness and Implicit Euler to the help

3.4.1 Introducing stiffness at the hand of a simple example

Consider the following ODE system (following Press et al., 2007, chapter 17.5)

$$\begin{aligned}\partial_t y_1 &= 998y_1 + 1998y_2 \\ \partial_t y_2 &= -999y_1 - 1999y_2\end{aligned}$$

with initial conditions $y_1(0) = 1$ and $y_2(0) = 0$. The system can be represented in matrix form as

$$\partial_t \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \underline{\underline{A}} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad \underline{\underline{A}} = \begin{pmatrix} 998 & 1998 \\ -999 & -1999 \end{pmatrix}$$

The eigenvalues of $\underline{\underline{A}}$ are $\lambda_1 = -1$ and $\lambda_2 = -1000$. The eigenvectors are $e_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ and $e_2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$. The solution of the system is then

$$\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \exp\left(\underline{\underline{A}}t\right) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \exp(-1t) + \begin{pmatrix} -1 \\ 1 \end{pmatrix} \exp(-1000t)$$

Let us now apply the Explicit Euler method to this system for different time-steps Δt . The result is shown in figure 16.

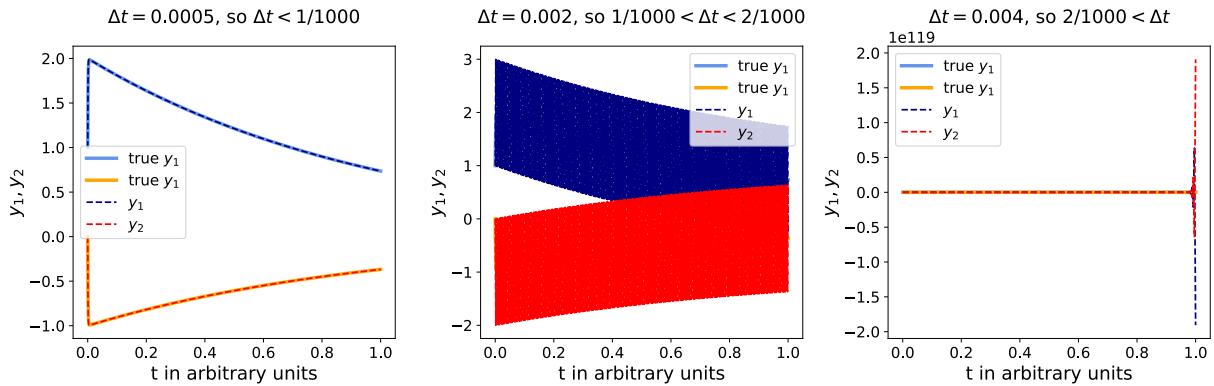


Figure 16: Numerical solution to the linear system $\partial_t \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \underline{\underline{A}} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ with $\underline{\underline{A}} = \begin{pmatrix} 998 & 1998 \\ -999 & -1999 \end{pmatrix}$ and $y_1(0) = 1, y_2(0) = 0$ using the Explicit Euler method for different time-steps Δt . The left panel shows the solution for $\Delta t = 0.0005$, the central one for $\Delta t = 0.002$ and the right one for $\Delta t = 0.004$.

Let us think back to the linear stability analysis of the Explicit Euler scheme for $\partial_t y = \alpha y, Re(\alpha) < 0, y(0) = y_0$. We had obtained

- $\Delta t < -\frac{1}{\alpha} \rightarrow$ we observe monotonous decrease (ok)
- $-\frac{1}{\alpha} < \Delta t < -\frac{2}{\alpha} \rightarrow$ oscillation (regarding the sign) but still decrease in the absolute value (problematic)
- $-\frac{2}{\alpha} < \Delta t \rightarrow$ an increasing, oscillating solution (very bad)

The same result holds in principle for our linear system - but with α replaced by the eigenvalue of largest magnitude of $\underline{\underline{A}}$, here $\lambda_2 = -1000$ (for the proof see Press et al., 2007, chapter 17.5).

As we move away from the origin, the fastest decreasing term $\propto \exp(-\lambda_2 t)$ in the true solution is completely negligible. However, in the explicit scheme it still sets the timescale

that has to be resolved for a stable solution.

In the setting of $\partial_t \underline{y} = \alpha \underline{y}$, $\text{Re}(\alpha) < 0$ the stability constraint for Δt is not too problematic because the resulting step-size is reasonable compared to the timescale of the problem. In the case of an ODE with different timescales in the solution, however, we are often interested in the timescale of the slowest processes but in the explicit scheme we still need to resolve the fastest timescale which quickly becomes infeasible. This is the problem of stiffness and can - in such a linear setting with all negative eigenvalues of $\underline{\underline{A}}$ - be characterized by the stiffness ratio

$$\text{stiffness ratio} := \frac{\max_{\text{eigenvalues } \lambda_i \text{ of } \underline{\underline{A}}} |\text{Re } \lambda_i|}{\min_{\text{eigenvalues } \lambda_i \text{ of } \underline{\underline{A}}} |\text{Re } \lambda_i|} = \frac{\lambda_2}{\lambda_1} = 1000$$

A large stiffness ratio indicates that an explicit scheme like the Explicit Euler method would be very inefficient for following the slowest process.

3.4.2 A *definition* of stiffness

As discussed in Lambert, 1991 a hard mathematical definition of stiffness is difficult and we therefore resort to the broad practical definition (Lambert, 1991, chapter 6)

»If a numerical method with a finite region of absolute stability, applied to a system with any initial conditions, is forced to use in a certain interval of integration a step length which is excessively small in relation to the smoothness of the exact solution in that interval, then the system is said to be stiff in that interval.«

An example for a numerical method with a finite region of absolute stability is the Explicit Euler method (see figure 15). In the example above, in spite of the fact that the solution is very smooth and the term $\propto \exp(-\lambda_2 t)$ is quickly negligible, we have to use excessively small steps.

3.4.3 Implicit Euler to the help

At the core of dealing with stiffness are implicit methods, the simplest representative being Implicit Euler.

An Implicit Euler step for solving $\partial_t \underline{y} = \underline{f}(\underline{y})$ is given by

$$\underline{y}^{(n+1)} = \underline{y}^{(n)} + \underline{f}(\underline{y}^{(n+1)}) \Delta t \quad \text{where} \quad \underline{y}^{(0)} = \underline{y}_0$$

which is an implicit equation as f is evaluated at the new time step $y^{(n+1)}$.

Intuition behind implicit Euler: We can write the implicit Euler step as $\underline{y}^{(n+1)} - f(\underline{y}^{(n+1)}) \Delta t = \underline{y}^{(n)}$, so which is the point where when I sit on it and shoot back with the corresponding slope, I get back to where I am coming from. This is illustrated in figure 17.

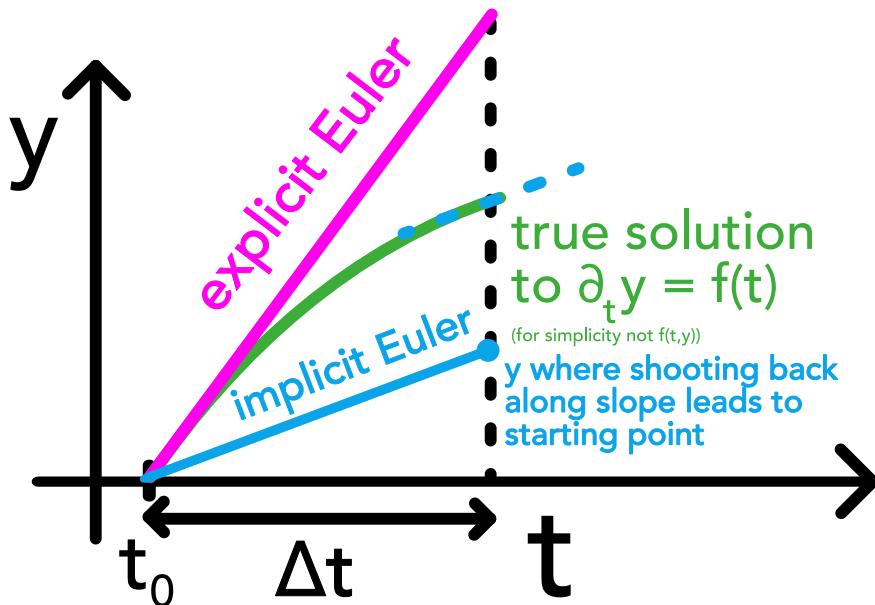


Figure 17: Illustration of the implicit Euler step.

Note: Implicit Euler is often referred to as backward Euler and the explicit Euler as forward Euler.

Problem: Note that implicit Euler is also a first order accurate scheme.

3.4.4 Region of absolute stability of the Implicit Euler method

As for the Explicit Euler method, we perform a linear stability analysis of the Implicit Euler method for $\partial_t y = \alpha y$, $Re(\alpha) < 0$, $y(0) = y_0$. We obtain

$$y^{(n+1)} = y^{(n)} + \alpha y^{(n+1)} \Delta t \quad \Rightarrow \quad y^{(n+1)} = \frac{1}{1 - \alpha \Delta t} y^{(n)}$$

which decreases for any $\Delta t > 0$ (illustrated in figure 18a). For large time-steps, the result is inaccurate (Implicit Euler is a first order scheme) but the solution remains stable. As of the stability function $R(z) = \frac{1}{1-z}$ the region of absolute stability is given by

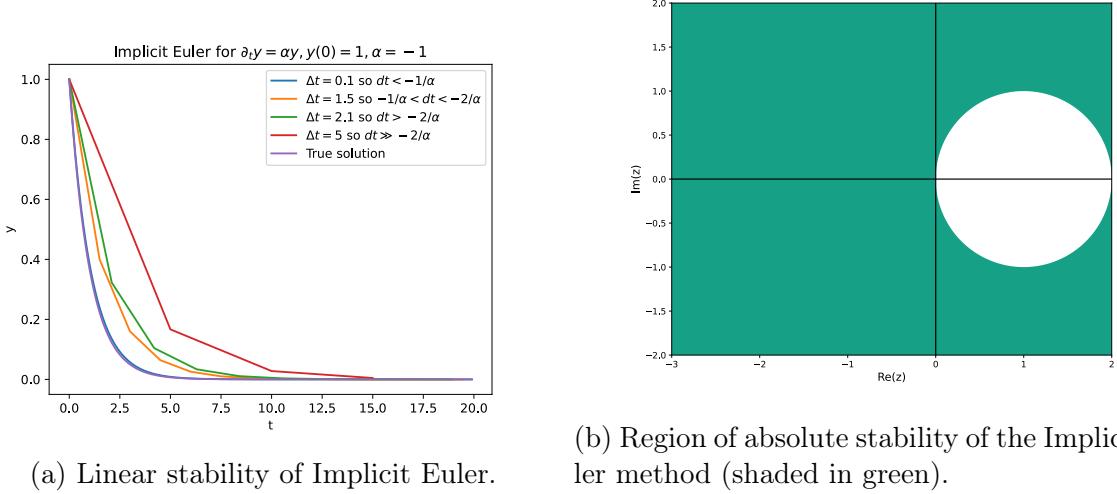


Figure 18: Stability of the Implicit Euler scheme.

$$\mathcal{D}_{\text{implicit euler}} = \{z \in \mathbb{C} \mid |R(z)| < 1\} = \{z \in \mathbb{C} \mid |1 - z| > 1\}$$

which is illustrated in figure 18b. The whole left half plane is included in the region of absolute stability and the method is therefore unconditionally stable.

3.4.5 Implicit Euler for stiff linear ODEs

As Implicit Euler is unconditionally stable, the fast oscillating terms resulting from

$$\partial_t \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \underline{\underline{A}} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad \underline{\underline{A}} = \begin{pmatrix} 998 & 1998 \\ -999 & -1999 \end{pmatrix}$$

with initial conditions $y_1(0) = 1$ and $y_2(0) = 0$ are no problem as illustrated in figure 19, where in spite of the relatively large time-step a good approximation of the solution is obtained.

The implicit step for such a linear system $\partial_t \underline{\underline{y}} = \underline{\underline{A}} \underline{\underline{y}}$ is

$$\underline{\underline{y}}^{(n+1)} = \underline{\underline{y}}^{(n)} + \underline{\underline{A}} \underline{\underline{y}}^{(n+1)} \Delta t \quad \Rightarrow \quad \left(\underline{\underline{I}} - \underline{\underline{A}} \Delta t \right) \underline{\underline{y}}^{(n+1)} = \underline{\underline{y}}^{(n)}$$

which means that to make a step we have to solve a linear system which is usually done by matrix decomposition (like LU decomposition).

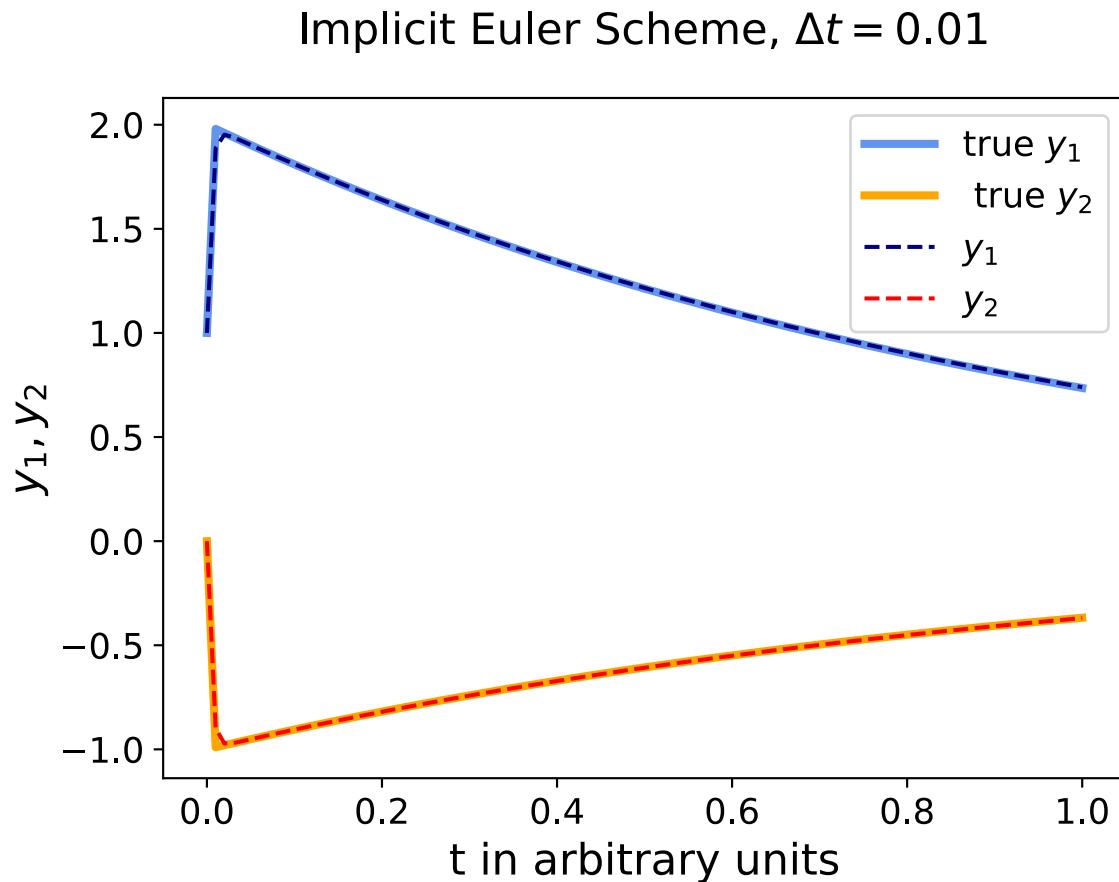


Figure 19: The same problem as in figure 16 is now approached using the Implicit Euler method and a relatively large time-step of $\Delta t = 0.01$.

3.4.6 But how can we approach non-linear ODEs using the Implicit Euler method?

To perform an implicit step

$$\underline{y}^{(n+1)} = \underline{y}^{(n)} + \underline{f}(\underline{y}^{(n+1)}) \Delta t$$

for a non-linear system $\partial_t \underline{y} = \underline{f}(\underline{y})$ like the Davis-Skodje equation

$$\begin{aligned}\dot{y}_1(t) &= -y_1(t) \\ \dot{y}_2(t) &= -\gamma y_2(t) + \frac{(\gamma - 1)y_1(t) + \gamma y_1^2(t)}{(1 + y_1(t))^2}\end{aligned}$$

where γ is a measure for the stiffness (see Heiter, 2012, chapter 2.4) we reformulate the implicit step as a root-finding problem

$$\begin{aligned} \underline{0} &= \underline{y}^{(n+1)} - \underline{y}^{(n)} - \Delta t \underline{f}(\underline{y}^{(n+1)}), \quad \underline{g}(\underline{\xi}) := \underline{\xi} - \underline{y}^{(n)} - \Delta t \underline{f}(\underline{\xi}) \\ &\rightarrow \underline{0} = \underline{g}(\underline{\xi}) \Leftrightarrow \underline{\xi} = \underline{y}^{(n+1)} \end{aligned}$$

where each of those time-steps is solved using Newton's method (or quasi-Newton)

$$\begin{aligned} \underline{\xi}_{k+1} &= \underline{\xi}_k - \underline{\underline{J}}_{\underline{g}}^{-1}(\underline{\xi}_k) \underline{g}(\underline{\xi}_k), \quad \underline{\underline{J}}_{\underline{g}} = \underline{\underline{I}} - \Delta t \underline{\gamma}(\underline{\xi}_k) \underline{\underline{J}}_{\underline{f}} \\ \underline{\xi}_0 &= \underline{y}^{(n)}, \quad \underline{\xi}_m \rightarrow \underline{y}^{(n+1)} \quad \text{for } m \rightarrow \infty \end{aligned}$$

where $\underline{\underline{J}}_{\underline{f}}$ is the Jacobian of \underline{f} . In Quasi-Newton the Jacobian is only recalculated once per time-step in the Euler method

$$\underline{\xi}_{k+1} = \underline{\xi}_k - \underline{\underline{J}}_{\underline{g}}^{-1}(\underline{\xi}_0) \underline{g}(\underline{\xi}_k)$$

For the Davis-Skodje problem mentioned above some Implicit Euler steps are drawn into the stream plot of the equation in figure 20. Here, one can also see the intuition behind Implicit Euler steps: One searches a point where the derivative is such that shooting back with this slope leads back to the point we are coming from, as

$$\underline{y}^{(n)} = \underline{y}^{(n+1)} - \underline{f}(\underline{y}^{(n+1)}) \Delta t$$

The steps of the Newton iteration done for each Implicit Euler step can most intuitively be understood in the formulation as the linear equation

$$\underline{b} := \underline{g}(\underline{\xi}_k) = \underline{\underline{J}}_{\underline{g}}(\underline{\xi}_k - \underline{\xi}_{k+1}) = \underline{\underline{J}}_{\underline{g}} \underline{a}, \quad \underline{a} := \underline{\xi}_k - \underline{\xi}_{k+1}$$

which is also the equation solved on the computer using matrix decomposition. $\underline{\underline{J}}_{\underline{g}} \underline{a}$ is the directional derivative of \underline{g} in the direction of \underline{a} and in a step of the Newton iteration we search for a step \underline{a} that gets us from $\underline{0}$ to \underline{b} in other words $\underline{\xi}_{k+1} = \underline{\xi}_k - \underline{a}$.

Problem: While the Implicit Euler method is unconditionally stable, performing the implicit step for non-linear ODEs requires solving a non-linear equation with some root-finding algorithm, which can be even more costly than doing small explicit steps if no proper care (e.g. smart forward differentiation in the root finding) is taken.

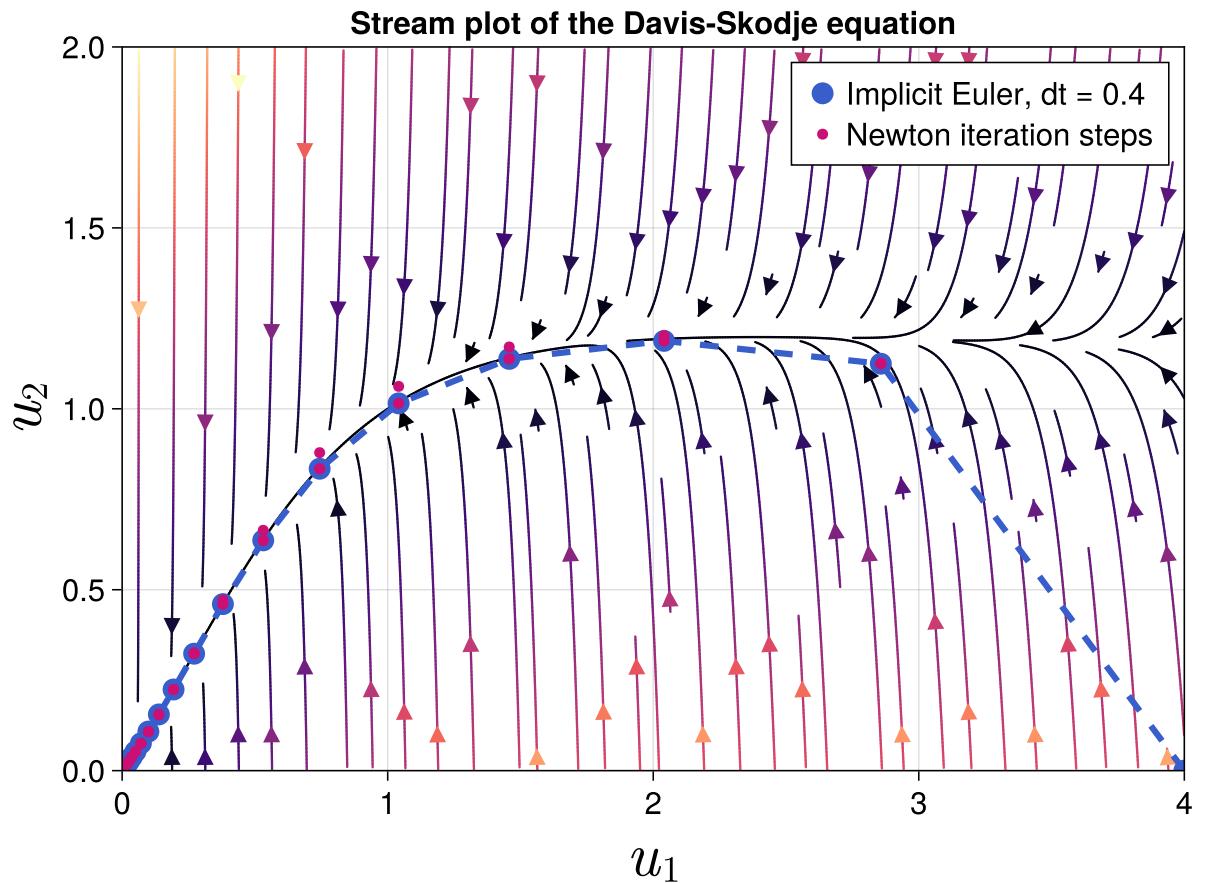


Figure 20: Stream plot of the Davis-Skodje equation with some Implicit Euler steps also drawn. The direction of the Implicit Euler steps is from right (starting at $(4, 0)$) to left.

3.5 Construction of higher-order methods

The two basic kinds of numerical methods for ODEs are

- one-step methods using one starting value at each step (in a step we want to get from $y^{(n)}$ to $y^{(n+1)}$)
- multistep methods using e.g. also $y^{(n-1)}, y^{(n-2)}$ to get from $y^{(n)}$ to $y^{(n+1)}$

In this subsection, we essentially only deal with one-step methods.

3.5.1 Meaning of going to higher order

Let us first understand, why we would want a higher order method. We still want to solve $\underline{f} = \underline{f}(y, t)$ with $\underline{y}(t^{(0)}) = \underline{y}^{(0)}$ and our scheme approximates $\underline{y}(t^{(0)} + h)$ as $\underline{y}^{(1)}$. The scheme has order p if

$$\|\underline{y}(t^{(0)} + h) - \underline{y}^{(1)}\| \leq K h^{p+1} \quad (56)$$

for sufficiently smooth problems, so if the Taylor series for the exact solution $\underline{y}(t^{(0)} + h)$ and for $\underline{y}^{(1)}$ coincide up to (and including) h^p .

Computational advantage: Our basic unit of cost are function evaluations of $f(\underline{y}, t)$, where explicit Euler takes one function evaluation per step and has $p = 1$. Now imagine we can construct a second order scheme ($p = 2$) with some constant number of function evaluations per step. While halving the step-size still doubles the integration cost over an interval for $p = 2$ it quarters the error, so at some point the higher order scheme will be advantageous.

3.5.2 Approaches to constructing a higher order method

Higher order schemes can either be constructed using Taylor expansion so higher order derivatives (can be costly) or by the weighted combination of simple derivatives of multiple points and clever substeps.

3.5.3 Construction by Taylor expansion

The most obvious way to get a higher order truncation error, is to build a scheme based on higher order Taylor expansion.

We start by expanding $y(t)$ around some time t .

$$y(t+h) = y(t) + h \partial_t y|_t + \frac{h^2}{2} \partial_t^2 y|_t + \frac{h^3}{6} \partial_t^3 y|_t + \mathcal{O}(h^4) \quad (57)$$

The expansion up to the first order is just our explicit Euler scheme. As of our problem statement (1st order ODE)

$$\partial_t y = f(y, t), \quad y(t^{(0)}) = y^{(0)}, \quad t^{(n)} = t^{(0)} + hn, \quad \text{stepsize } h \quad (58)$$

(with the solutions defining 2D trajectories $(t, y(t))$ and we approximate $(t^{(n)}, y^{(n)})$), we can express the higher order derivatives in the Taylor expansion as (chain rule)

$$\partial_t^k y|_t = \left(\frac{d}{dt} \right)^{k-1} f \Big|_{y(t), t} =: f^{(k-1)}(y, t) \quad (59)$$

$$f^{(k)}(y, t) = \partial_t f^{(k-1)}(y, t) + (\partial_t y) \partial_y f^{(k-1)}(y, t) = \partial_t f^{(k-1)}(y, t) + f \partial_y f^{(k-1)}(y, t)$$

Problem: The higher order derivatives have to be calculated recursively e.g. with forward differentiation at the ground level which is complicated and slow.

Problem: In high-order Taylor expansion, the individual terms can be numerically problematic, e.g. in

$$\cos(x)|_{x=0} \approx 1 - \frac{x^2}{2} + \frac{x^4}{4!} + \dots \quad (60)$$

all polynomial terms diverge while the infinite sum is bound in $[-1, 1]$ as expected to the cosine.

3.5.4 Runge-Kutta (RK) Integration schemes I: General Idea

Consider instead of a 1st order ODE problem $\partial_t y = f(y, t)$ we had a quadrature problem $\partial_t y = f(t)$. Then a step would be

$$y(t+h) = y(t) + \int_t^{t+h} f(t') dt' \quad (61)$$

where we could approximate, e.g. using the trapezoidal rule

$$y^{(n+1)} = y^{(n)} + h \frac{f^{(n+1)} + f^{(n)}}{2}, \quad f(n) = f(t^{(0)} + hn) \quad (62)$$

We can't just apply this to the ODE $\partial_t y = f(y, t)$, because to calculate $f^{(n+1)}$ there we need $y^{(n+1)}$ which is what we are searching for. But what if we would approximate $y^{(n+1)}$ for $f^{(n+1)}$ with an Euler step? We would then have

$$\begin{aligned} k_1 &= f(y^{(n)}, t^{(n)}) \\ k_2 &= f(y^{(n)} + hk_1, t^{(n)} + h) \\ y^{(n+1)} &= y^{(n)} + \frac{h}{2} (k_1 + k_2) + \mathcal{O}(h^3) \end{aligned} \quad (63)$$

where the central advantage roughly is, that k_2 which includes the Euler approximation of $\mathcal{O}(h^2)$ is multiplied by h in the expression for $y^{(n+1)}$ so the error becomes less important. This already is the $RK2$ scheme. The more general idea is illustrated in figure 21.

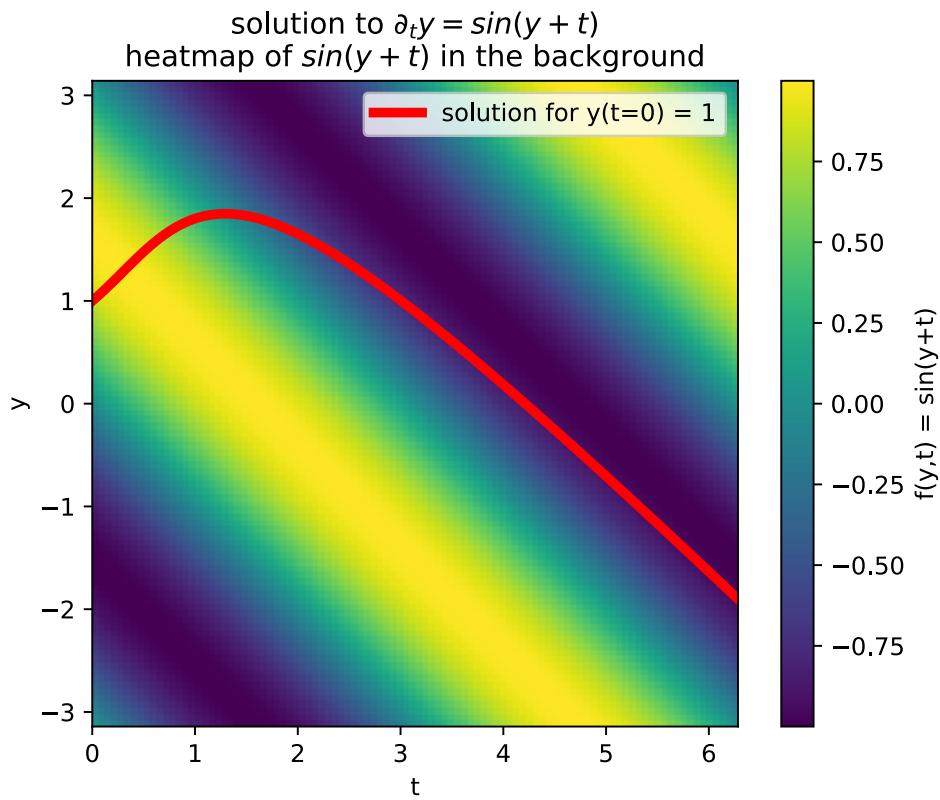


Figure 21: For a step in t the corresponding step in y equals the integration of $f(y, t)$ over the correct path. The correct path, however, is unknown. So we approximate points along the path to get a better approximation of the step as an approximation of the integral over $f(y, t)$.

3.5.5 Runge-Kutta (RK) Integration schemes II: Derivation of the general RK scheme

Using clever substeps, we want to construct an accurate and cost-efficient scheme, to solve the initial value problem $\partial_t \underline{y} = f(\underline{y}, t)$ with $\underline{y}(t^{(0)}) = \underline{y}^{(0)}$. We make the ansatz

$$\begin{aligned}
 \underline{y}(t+h) &= \underline{y}(t) + [\underline{y}(t+h) - \underline{y}(t)] \\
 &= \underline{y}(t) + \int_t^{t+h} \partial_t \underline{y}|_{t'} dt' \\
 &= \underline{y}(t) + h \int_0^1 \partial_t \underline{y}|_{t+\tau h} d\tau \\
 &= \underline{y}(t) + h \int_0^1 f(\underline{y}(t+\tau h), t+\tau h) d\tau
 \end{aligned} \tag{64}$$

which will later allow us to extrapolate \underline{y} according to the ODE. We approximate the integral by a quadrature rule of the form

$$\int_0^1 g(\tau) d\tau \approx \sum_{i=1}^m \beta_i g(\gamma_i), \quad \text{RK weights } \beta_i, \quad \text{RK nodes } \gamma_i$$

$$\sum_{i=1}^m \beta_i = 1 \quad \rightarrow \quad \text{correct integration of unity } g \equiv 1, \quad \text{scheme order } m \quad (65)$$

Application to the ansatz (64) yields

$$\underline{y}(t+h) \approx \underline{y}(t) + h \sum_{i=1}^m \beta_i \underline{f}(\underline{y}(t + \gamma_i h), t + \gamma_i h) \quad (66)$$

Problem: We don't know $\underline{y}(t + \gamma_i h)$ yet.

Idea: Use an analogous quadrature rule to get approximations to $\underline{y}(t + \gamma_i h)$ with the γ_i as nodes again - quadrature in quadrature.

$$\underline{y}(t + \gamma_i h) = \underline{y}(t) + h \int_0^{\gamma_i} \partial_t \underline{y} \Big|_{t+\tau h} d\tau \approx \underline{y}(t) + h \sum_{l=1}^m \alpha_{i,l} \partial_t \underline{y} \Big|_{t+\gamma_l h}$$

$$\sum_{i=1}^m = \gamma_i \quad \rightarrow \quad \text{correct integration of unity } g \equiv 1, \quad i = 1, \dots, m \quad (67)$$

We define

$$\underline{k}_l := \partial_t \underline{y} \Big|_{t+\gamma_l h} = \underline{f}(\underline{y}(t + \gamma_l h), t + \gamma_l h) \quad (68)$$

But what have we won, we still need the $\underline{y}(t + \gamma_l h)$, right? By setting $\alpha_{i,l} = 0$ for $l \leq i$, we gain an explicit scheme where $\underline{y}(t + \gamma_l h)$ is constructed only based on previous substeps.

1. Starting with $\underline{y}^{(0)} = \underline{y}(t^{(0)})$ and $\underline{k}_1 = \underline{f}(\underline{y}^{(0)}, t^{(0)})$ we approximate $\underline{y}(t^{(0)} + \gamma_1 h)$ from which we calculate $\underline{k}_1 = \underline{f}(\underline{y}(t^{(0)} + \gamma_1 h), t^{(0)} + \gamma_1 h)$, then based on k_0, k_1 we approximate $\underline{y}(t^{(0)} + \gamma_2 h)$ and thus k_3 and so on.
2. Based on $\underline{k}_1, \dots, \underline{k}_m$, we approximate $\underline{y}^{(1)} = \underline{y}^{(0)} + h \sum_{i=1}^m \beta_i \underline{k}_i$
3. ...

3.5.6 Runge-Kutta (RK) Integration schemes III: General m-substep RK method

In general, we have obtained

$$\begin{aligned}\underline{y}^{(n+1)} &= \underline{y}^{(n)} + h \sum_{i=1}^m \beta_i \underline{k}_i \\ \underline{k}_i &= f \left(\left(\underline{y}^{(n)} + h \sum_{l=1}^{m-1} \alpha_{i,l} \underline{k}_l \right), t_n + \gamma_i h \right), \quad i = 1, \dots, m \\ \sum_{l=1}^m \alpha_{i,l} &= \gamma_i, \quad \alpha_{i,l} = 0 \text{ for } l \geq i \rightarrow \text{ explicit method}\end{aligned}\tag{69}$$

where for $\alpha_{i,l} = 0$ for $l \geq i$, \underline{k}_i only depends on $\underline{k}_l, l < i$.

3.5.6.1 Butcher-Tableau for visualizing the RK coefficients

The general Butcher-Tableau is given in figure 22, for an explicit scheme, the α 's form a lower left triangular matrix ($\alpha_{i,l} = 0$ for $l \geq i$, \underline{k}_i).

Examples for butcher tableaus of explicit RK methods are given in table 3.

Explicit Euler (1st order)	Implicit Euler (1st order)	RK2 (2nd order)	RK4 (4th order)
$\begin{array}{c c} 0 & \\ \hline & 1 \end{array}$	$\begin{array}{c c} 0 & 1 \\ \hline & 1 \end{array}$	$\begin{array}{c cc} 0 & & \\ \hline 1 & 1 & \\ & \frac{1}{2} & \frac{1}{2} \end{array}$	$\begin{array}{c cccc} 0 & & & & \\ \hline \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$

Table 3: Butcher-Tableau for explicit RK methods.

Butcher-Tableau

RK-nodes

γ_1	$\alpha_{1,1} \cdots \alpha_{1,m}$
\vdots	\vdots
γ_m	$\alpha_{m,1} \cdots \alpha_{m,m}$
<hr/>	
$\beta_1 \cdots \beta_m \xrightarrow{+1}$	
RK-weights	

weights used to integrate up to $y(t + \gamma_1 h)$ to find k_1

Figure 22: Butcher-Tableau for the general m-substep RK method.

3.5.7 Runge-Kutta (RK) Integration schemes IV: Taylor expansion to identify RK parameters for 2nd order schemes

3.5.7.1 Comparison of coefficients

We want to find appropriate α 's and β 's such that the RK scheme follows the **Taylor expansion**

$$\begin{aligned}
 y(t^{(n+1)}) &= y^{(n+1)} \\
 &= y^{(n)} + h\partial_t y + \frac{h^2}{2}\partial_t^2 y + \mathcal{O}(h^3) \\
 &= y^{(n)} + hf + \frac{h^2}{2}\frac{d}{dt}f + \mathcal{O}(h^3) \\
 &= y^{(n)} + hf + \frac{h^2}{2}((\partial_y f)\partial_t y + \partial_t f) + \mathcal{O}(h^3) \\
 &\boxed{= y^{(n)} + hf + \frac{h^2}{2}(\textcolor{blue}{f}\partial_y f + \partial_t f) + \mathcal{O}(h^3)}
 \end{aligned} \tag{70}$$

where if not specified differently, the evaluation is at (y_n, t_n) , so that we know that the error per step is $\mathcal{O}(h^3)$.

Now let us bring the explicit RK ansatz for $m = 2$ (\rightarrow only $\alpha_{2,1} \neq 0$ so $\gamma_1 = 0$ and $\gamma_2 = \alpha_{2,1}$) into the form of the Taylor expansion. We start with

$$\begin{aligned}
 y^{(n+1)} &= y^{(n)} + h(\beta_1 \textcolor{blue}{k}_1 + \beta_2 \textcolor{teal}{k}_2) \\
 &= y^{(n)} + h(\beta_1 \textcolor{blue}{f} + \beta_2 \textcolor{teal}{f}((y^{(n)} + h\alpha_{2,1}f), t^{(n)} + \gamma_2 h))
 \end{aligned} \tag{71}$$

and first order expand $\textcolor{teal}{k}_2$ to

$$\textcolor{teal}{f}((y^{(n)} + h\alpha_{2,1}f), t^{(n)} + \gamma_2 h) = f + h\alpha_{2,1}f\partial_y f + \gamma_2 h\partial_t f \tag{72}$$

Using $\gamma_2 = \alpha_{2,1}$ yields a form that allows comparison of coefficients

$$\begin{aligned}
 y^{(n+1)} &= y^{(n)} + h \cdot (\beta_1 f + \beta_2 (\textcolor{teal}{f} + h\alpha_{2,1}f\partial_y f + \alpha_{2,1}h\partial_t f)) \\
 &\boxed{= y^{(n)} + hf \cdot (\beta_1 + \beta_2) + h^2\beta_2\alpha_{2,1}(f\partial_y f + \partial_t f)}
 \end{aligned} \tag{73}$$

Comparing the boxed equations yields two equations for three variables $(\beta_1, \beta_2, \alpha_{2,1})$

$$\beta_1 + \beta_2 = 1, \quad \beta_2\alpha_{2,1} = \frac{1}{2} \quad \rightarrow \quad \text{choose } \beta_2 = q \tag{74}$$

so

$$\beta_1 = 1 - q, \quad \beta_2 = q, \quad \alpha_{2,1} = \frac{1}{2q} \tag{75}$$

3.5.7.2 Resulting integration formula with free parameter q

We get the integration formula

$$t^{(n+1)} = t^{(n)} + h, \quad y^{(n+1)} = y^{(n)} + h \left[(1 - q)f + qf \left(\left(y^{(n)} + \frac{h}{2q} f \right), t^{(n)} + \frac{h}{2q} \right) \right] \quad (76)$$

3.5.7.3 Different integration schemes based on the choice of q

Depending on q we can yield different schemes, with examples shown in table 4.

Only based on two function evaluations of f , the midpoint rule and RK2 are already second order accurate.

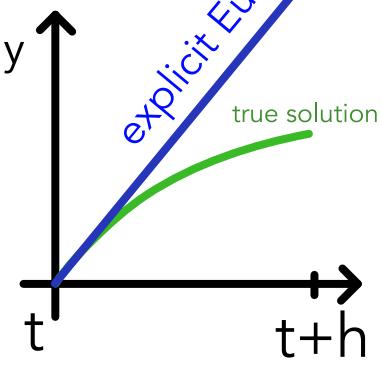
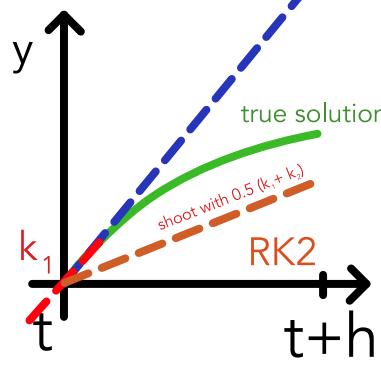
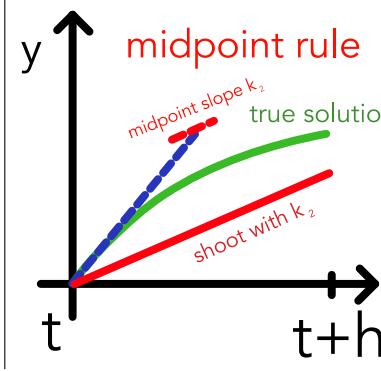
$q = 0$	$q = \frac{1}{2}$	$q = 1$
Forward Euler	2nd order Runge-Kutta (RK2) (aka Heun-method)	Midpoint-Rule
$k_1 = f(y^{(n)}, t^{(n)})$ $y^{(n+1)} = y^{(n)} + hk_1 + \mathcal{O}(h^2)$	$k_1 = f(y^{(n)}, t^{(n)})$ $k_2 = f(y^{(n)} + hk_1, t^{(n)} + h)$ $y^{(n+1)} = y^{(n)} + \frac{h}{2}(k_1 + k_2) + \mathcal{O}(h^3)$	$k_1 = f(y^{(n)}, t^{(n)})$ $k_2 = f(y^{(n)} + \frac{h}{2}k_1, t^{(n)} + \frac{h}{2})$ $y^{(n+1)} = y^{(n)} + hk_2 + \mathcal{O}(h^3)$
Shoot along a tangent from the starting point.	<ul style="list-style-type: none"> Using k_1 Euler-approximate $y(t+h)$ Find k_2 using this approximation Shoot with the mean of k_1 and k_2 	Approximate y at the midpoint of the interval and use the slope there for shooting across the whole interval.
		

Table 4: Different schemes based on the choice of q .

3.5.8 Runge-Kutta (RK) Integration schemes V: Classical 4th order RK scheme (RK-4)

$$\begin{aligned}
 \underline{k}_1 &= f(\underline{y}^{(n)}, t^{(n)}), \quad \underline{k}_2 = f(\underline{y}^{(n)} + \frac{h}{2}\underline{k}_1, t^{(n)} + \frac{h}{2}), \\
 \underline{k}_3 &= f(\underline{y}^{(n)} + \frac{h}{2}\underline{k}_2, t^{(n)} + \frac{h}{2}), \quad \underline{k}_4 = f(\underline{y}^{(n)} + hk_3, t^{(n)} + h) \\
 \underline{y}^{(n+1)} &= \underline{y}^{(n)} + \frac{h}{6} (\underline{k}_1 + 2\underline{k}_2 + 2\underline{k}_3 + \underline{k}_4) + \mathcal{O}(h^5)
 \end{aligned} \tag{77}$$

Note: Here we need 4 function evaluations of f per step. Depending on the situation, lower order schemes with smaller stepsize might be more efficient. Choosing an appropriate step-size might be very important.

An example application of RK4 with the corresponding evaluation points of k_2, k_3, k_4 is shown in figure 23.

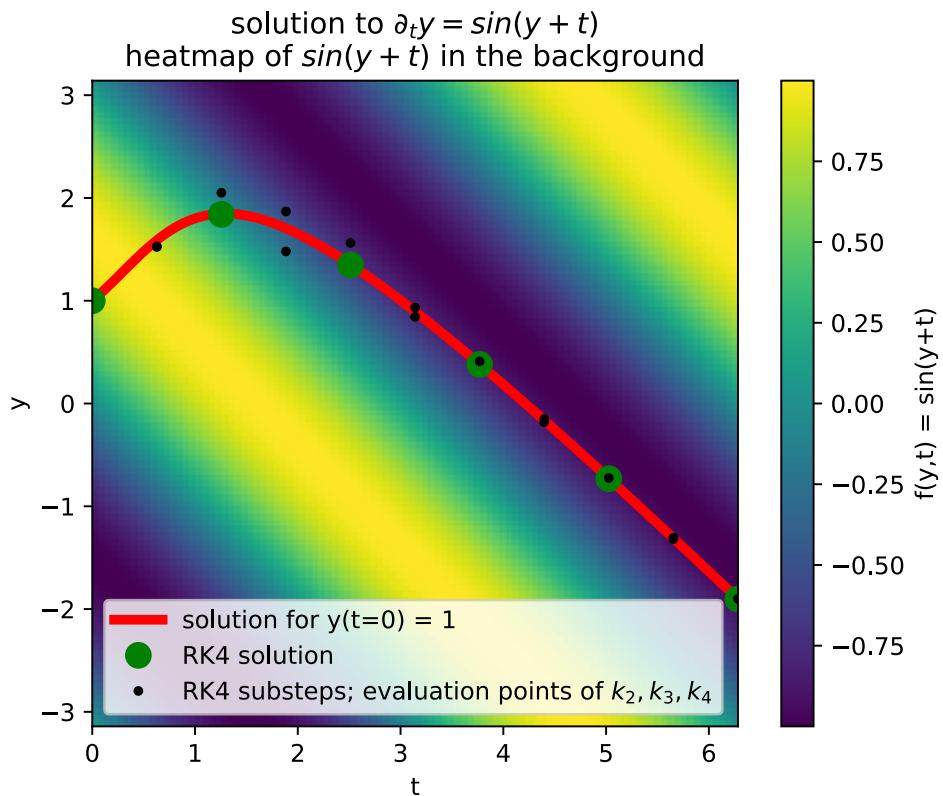


Figure 23: RK4 applied to the problem $\partial_t y = f(y, t)$ with $f(y, t) = \sin(y + t)$ and $y(0) = 1$. The evaluation points of k_2, k_3, k_4 are marked in black.

3.6 Adaptive Step Sizes

Problem: While in one region of a problem a relatively large step-size might suffice in others we might need a very small one. Using the large step size everywhere does not work but taking the small one everywhere is a waste of compute.

Idea: Take steps of adaptive size, striking a balance between accuracy, stability and efficiency. The adaptation is based on the estimation of a local integration error and a user-specified wanted upper bound on it. For an example see figure 24.

When we simulate e.g. hydrogen in space we might even want to use different step-sizes in different local regions of the problem (smaller timesteps in denser regions).

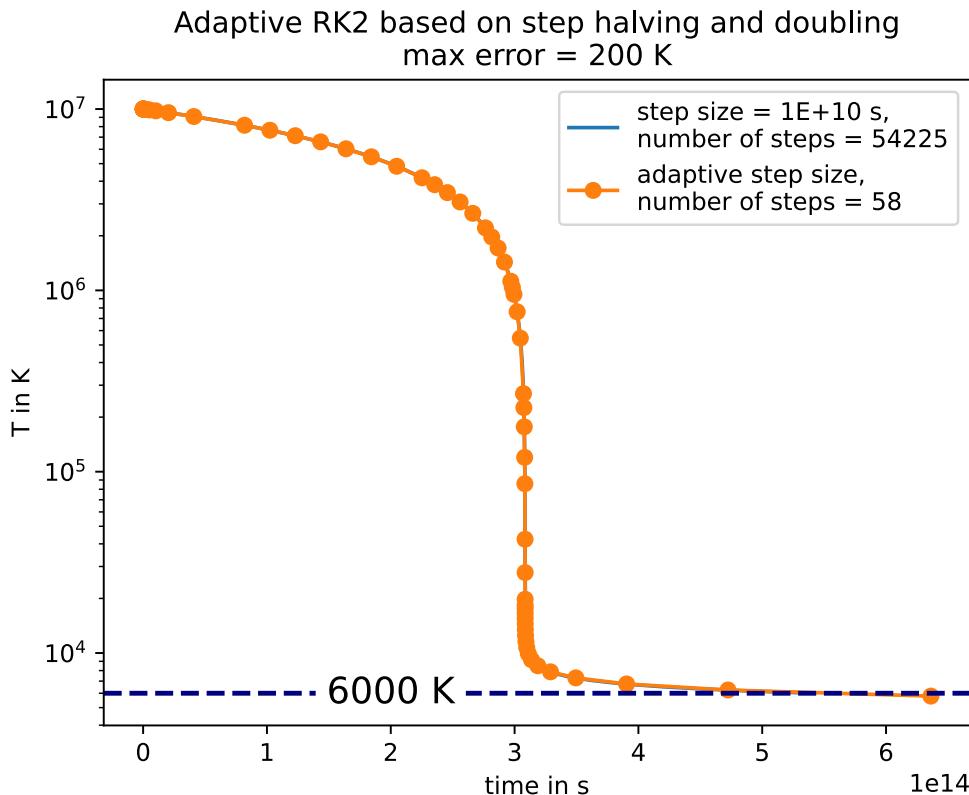


Figure 24: Example of adaptive step sizes.

3.6.1 Step halving and doubling method

We can estimate the local integration error by

- performing one step of size h to obtain y_a
- performing two steps of size $\frac{h}{2}$ to obtain y_b

from the same starting point and then comparing the results.

With this, the **halving and doubling scheme** given the user-specified local upper error bound ϵ_0 is

1. Calculate y_a, y_b and $\epsilon = |y_a - y_b|$
2. If $\epsilon > \epsilon_0$ discard the step and try again with $h' = \frac{h}{2}$
3. If $\epsilon \ll \epsilon_0$, keep y_b and use $h' = 2h$ for the next step (doubling)
4. Else if $\epsilon < \epsilon_0$, keep y_b and retain h for the next step

Advantage of the halving and doubling scheme in spatio-temporal simulations:

Consider some hydrogen simulation. In a halving-doubling scheme we can use different step sizes in different spatial regions and still have results in sync - for every point on the coarse time-grid we will also have a result from the finer grids in time, see figure 25

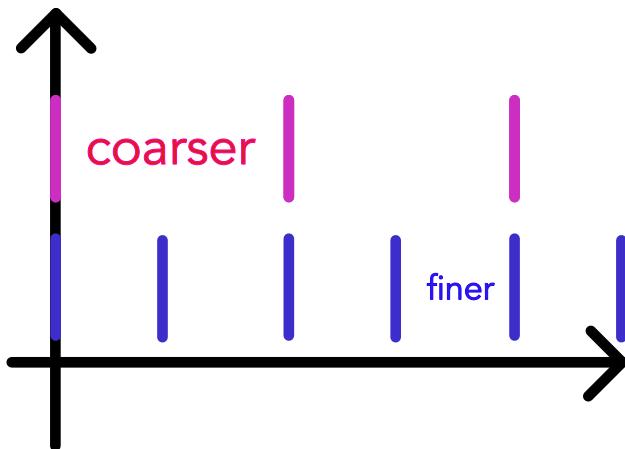


Figure 25: Different step sizes in time but still results in sync in the halving-doubling scheme.

Note: We said we want to double h for the next step for $\epsilon \ll \epsilon_0$. A more concise criterion is even if we double we would expect an error lower than the user-specified upper bound. For this we first make a remark on the local accuracy.

3.6.2 Note on the Local accuracy

As a p -th order scheme is locally accurate to the $p+1$ -th order, we can write the local errors as

$$\begin{aligned} y_a - y(t^{(0)} + h) &= \alpha h^{p+1} + \mathcal{O}(h^{p+2}) \\ y_b - y(t^{(0)} + h) &= 2\alpha \left(\frac{h}{2}\right)^{p+1} + \mathcal{O}(h^{p+2}) \end{aligned} \quad (78)$$

yielding an error estimate of

$$\epsilon = |y_a - y_b| = \alpha h^{p+1} (1 - 2^{-p}) \quad (79)$$

3.6.3 When does doubling make sense?

For a doubled time-step, we expect the error ($h \rightarrow 2h$ in (79)) to be

$$\epsilon' = \alpha(2h)^{p+1} (1 - 2^{-p}) = 2^{p+1} \epsilon \quad (80)$$

which we still want to be smaller than ϵ_0 so we double if

$$\epsilon' = 2^{p+1}\epsilon < \epsilon_0 \quad (81)$$

so when the expected error after doubling is below our error bound.

3.6.4 Adaptively choosing ϵ_0

Problem: The smaller the time-step the more steps we need to cover a certain timespan, the more error can accumulate.

Idea: Set ϵ_0 adaptively, taking into account how many timesteps one would need to cover the total integration time with the current h , so

$$\epsilon_0 = \frac{h}{T} \epsilon_0^{\text{global}}, \quad \text{total integration time } T, \quad \text{prescribed global error bound } \epsilon_0^{\text{global}} \quad (82)$$

3.6.5 Continuous time step adjustment

While the halving-doubling is nicely suited for being able to use different step-sizes in different spatial regions of a simulation, often we want a more flexible continuous adjustment.

Idea: For the next step use a step-size h^{new} such that we would assume this step-size to just hit the error bound in our current step (with some safety factor.)

The next timestep is scaled according to the current error, such that for h^{desired} we have $\epsilon' = \epsilon_0$ so

$$\begin{aligned} \epsilon_0 = \epsilon' &= \alpha \cdot (h^{\text{desired}})^{p+1} \cdot (1 - 2^{-p}) = \left(\frac{h^{\text{desired}}}{h} \right)^{p+1} \epsilon \\ &\rightarrow h^{\text{desired}} = h \left(\frac{\epsilon_0}{\epsilon} \right)^{\frac{1}{p+1}}, \quad \text{error } \epsilon \text{ if step with size } h \text{ is taken} \end{aligned} \quad (83)$$

Note that we have used the error formula for the halving-doubling scheme but more generally assuming $\mathcal{O}(\epsilon) = h^{p+1}$ we get the same result for h^{desired} from

$$\frac{\epsilon_0}{\epsilon} = \left(\frac{h^{\text{desired}}}{h} \right)^{p+1} \quad (84)$$

3.6.5.1 Continuous adaptive time step control scheme

We get to the scheme

1. Advance the system by a step h and estimate the error of the step, we assume a p -th

order scheme with $\mathcal{O}(\epsilon) = h^{p+1}$

2. Calculate the new step size as

$$h^{\text{new}} = \beta h \left(\frac{\epsilon_0}{\epsilon} \right)^{\frac{1}{p+1}} \quad (85)$$

with some safety factor $\beta \sim 0.9$

3. If $\epsilon < \epsilon_0$ accept the step, otherwise discard it and repeat with the new step-size

But is there a more efficient way to estimate ϵ than the halving-doubling scheme?

3.6.5.2 Embedded Runge-Kutta schemes for cheaper error estimates

In embedded Runge-Kutta schemes like Runge-Kutta-Fehlberg (e.g. RKF45), based on the same function evaluations and respectively k_i schemes of different order are constructed and from the difference of their results, the error is estimated.

For instance RK45 is illustrated in figure 26.

RK nodes	$k_1 = f(y^{(n)}, t^{(n)})$	$k_2 = f(y^{(n)} + \frac{1}{4} h k_1, t^{(n)} + \frac{1}{4} h)$	$k_3 = f(y^{(n)} + \frac{3}{32} h k_1 + \frac{9}{32} h k_2, t^{(n)} + \frac{3}{8} h)$	$k_4 = f(y^{(n)} + \frac{1932}{2197} h k_1 - \frac{7200}{2197} h k_2 + \frac{7296}{2197} h k_3)$	$k_5 = f(y^{(n)} + \frac{439}{216} h k_1 - 8 h k_2 + \frac{3680}{513} h k_3 - \frac{845}{4104} h k_4)$	$k_6 = f(y^{(n)} + -\frac{8}{27} h k_1 + 2 h k_2 - \frac{3544}{2565} h k_3 + \frac{1859}{4104} h k_4 - \frac{11}{40} h k_5)$
0	$k_1 = f(y^{(n)}, t^{(n)})$					
$\frac{1}{4}$		$k_2 = f(y^{(n)} + \frac{1}{4} h k_1, t^{(n)} + \frac{1}{4} h)$				
$\frac{3}{8}$			$k_3 = f(y^{(n)} + \frac{3}{32} h k_1 + \frac{9}{32} h k_2, t^{(n)} + \frac{3}{8} h)$			
$\frac{12}{13}$				$k_4 = f(y^{(n)} + \frac{1932}{2197} h k_1 - \frac{7200}{2197} h k_2 + \frac{7296}{2197} h k_3)$		
1					$k_5 = f(y^{(n)} + \frac{439}{216} h k_1 - 8 h k_2 + \frac{3680}{513} h k_3 - \frac{845}{4104} h k_4)$	
$\frac{1}{2}$						$k_6 = f(y^{(n)} + -\frac{8}{27} h k_1 + 2 h k_2 - \frac{3544}{2565} h k_3 + \frac{1859}{4104} h k_4 - \frac{11}{40} h k_5)$
						$\epsilon = y_A^{(n+1)} - y_B^{(n+1)} $

Figure 26: Embedded Runge-Kutta scheme RK45.

3.7 The problem of conserved quantities | Symplectic Integrators

3.7.1 Hamiltonian Systems and Symplecticity

Evolutions of classical physical systems can very generally be stated in terms of equations of motion derived from a real-valued, smooth Hamiltonian $H(\underline{p}, \underline{q})$

$$\begin{aligned}\partial_t \underline{p} &= -\underline{\nabla}_q H(\underline{p}, \underline{q}) \\ \partial_t \underline{q} &= \underline{\nabla}_{\underline{p}} H(\underline{p}, \underline{q})\end{aligned}\quad \text{or} \quad \partial_t \underline{y} = \underline{J}^{-1} \underline{\nabla} H(\underline{y}), \underline{y} = \begin{pmatrix} \underline{p} \\ \underline{q} \end{pmatrix}, \quad \underline{J} = \begin{pmatrix} 0 & \underline{1} \\ -\underline{1} & 0 \end{pmatrix} \in \mathbb{R}^{2d} \quad (86)$$

where $\underline{p} \in \mathbb{R}^d$ is the generalized momentum and $\underline{q} \in \mathbb{R}^d$ are the generalized coordinates. The Hamiltonian can be followed as the legendre transform of the Lagrangian.

Hamiltonian systems

- conserve energy, i. e. $H(\underline{p}, \underline{q})$ is conserved along a trajectory if the Hamiltonian does not explicitly depend on time
- show *symplecticity*, which is most intuitively understood as area conservation in phase space Hairer, Wanner, and Lubich, 2006, phase space volumina spanned by trajectories remain constant (see the following examples) / the phase space ditribution function is constant along trajectories

3.7.1.1 Poisson brackets and constants of motion (first integrals)

The Poisson brackets are

$$\{f, g\} = \{f, g\}_{qp} = \sum_{k=1}^d \frac{\partial f}{\partial q_k} \frac{\partial g}{\partial p_k} - \frac{\partial f}{\partial p_k} \frac{\partial g}{\partial q_k} \quad (87)$$

and we can find the **total derivation of a variable $F(p, q, t)$ along physical trajectories** as

$$\frac{dF(p(t), q(t), t)}{dt} = \{F, H\} + \partial_t F \quad (88)$$

so F is a constant of motion if $\{F, H\} = 0$ and $\partial_t F = 0$

3.7.1.2 Canonical transformations

Consider a coordinate transform in phase space

$$(p, q, t) \rightarrow (P, Q, t), \quad Q_i = Q_i(p, q, t), \quad P_i = P_i(p, q, t), \quad i = 1, \dots, d \quad (89)$$

and the corresponding transformation of the Hamiltonian

$$H(p, q, t) \rightarrow \tilde{H}(P, Q, t) \quad (90)$$

Then if the coordinate transformation is a **canonical transformations** it necessarily leaves the Hamiltonian equations of movement invariant.

$$\begin{aligned}\partial_t \underline{P} &= -\underline{\nabla}_Q \tilde{H}(\underline{P}, \underline{Q}) \\ \partial_t \underline{Q} &= \underline{\nabla}_P \tilde{H}(\underline{P}, \underline{Q})\end{aligned}\tag{91}$$

Closely connected are the Hamilton-Jacobi equations.

3.7.1.3 Definition of symplectic transformations

Linear maps: A linear map in 2D $F : \mathbb{R}^{2d} \rightarrow \mathbb{R}^{2d}$ is called symplectic if $\forall \underline{\xi}, \underline{\eta} \in \mathbb{R}^{2d} : \omega(F\underline{\xi}, F\underline{\eta}) = \omega(\underline{\xi}, \underline{\eta})$, where ω gives the area of the parallelogram spanned by both vectors.

Differentiable maps: $g : U \rightarrow \mathbb{R}^{2d}$ is symplectic if its Jacobian matrix $J_{\underline{\underline{g}}}$ is everywhere symplectic, so $\omega(J_{\underline{\underline{g}}} \underline{\xi}, J_{\underline{\underline{g}}} \underline{\eta}) = \omega(\underline{\xi}, \underline{\eta})$.

Connection to canonical transformations: In Hamiltonian systems, symplectic transformations are canonical transformations.

Compositions: Compositions of symplectic transformations are symplectic again.

Poincare's recurrence theorem: The time-evolution generated by a Hamiltonian in phase space is a symplectic transformation.

Idea: If the Hamiltonian evolution is symplectic, maybe its advantageous if our integrator is symplectic too.

3.7.2 Runge-Kutta methods do not conserve energy and are not symplectic

Let us now apply RK2 to the two-body-problem reduced to a dimensionless one-body-problem as described by

$$\frac{ds}{d\tau} = \underline{w}, \quad \frac{dw}{d\tau} = -\frac{\underline{s}}{\underline{s}^3}$$

where \underline{s} is the position and \underline{w} the velocity. The numerical solution obtained using RK2 can be seen in figure 27. One can see that neither the energy nor the angular momentum nor the Runge-Lenz vector are conserved. Every step in the scheme is erroneous, and those errors add up leading to our quantities of interest not being conserved.

Let us look at another physical problem, the pendulum, as it lends itself well to phase space visualization. We see in figure 28 that the phase space area, the area spanned by the test points as they propagate in time, is not preserved as it should be for Hamiltonian flow.

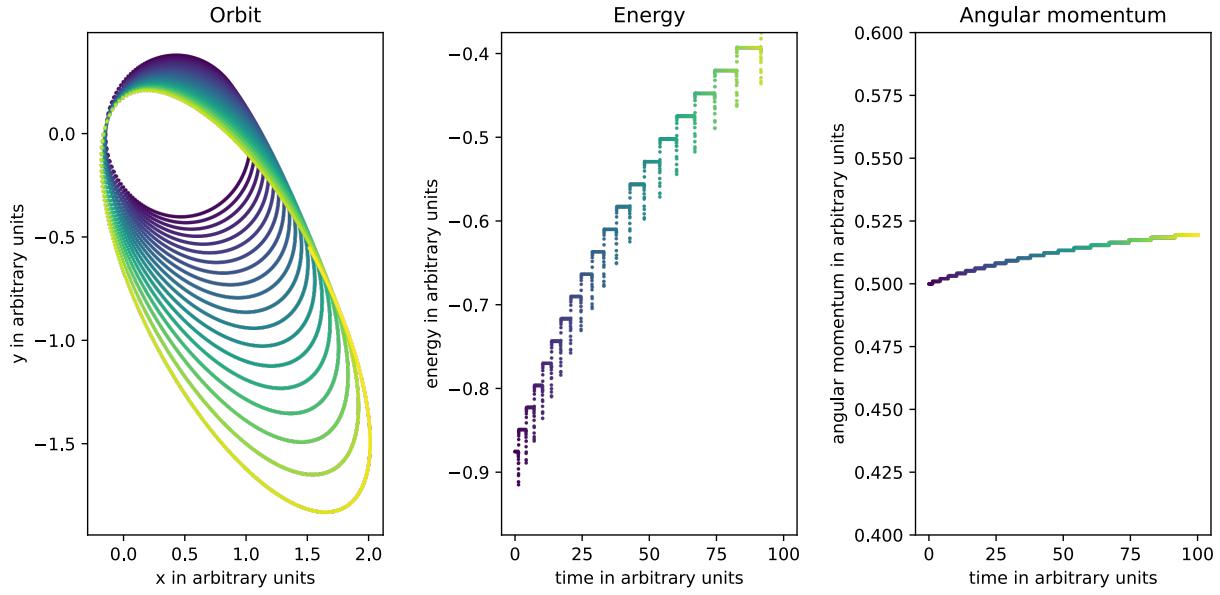


Figure 27: Numerical solution of the two-body-problem using RK2. The left panel shows the orbit, the central one the energy and the right one the angular momentum. Time is also encoded in the form of color, going from a dark blue to yellow.

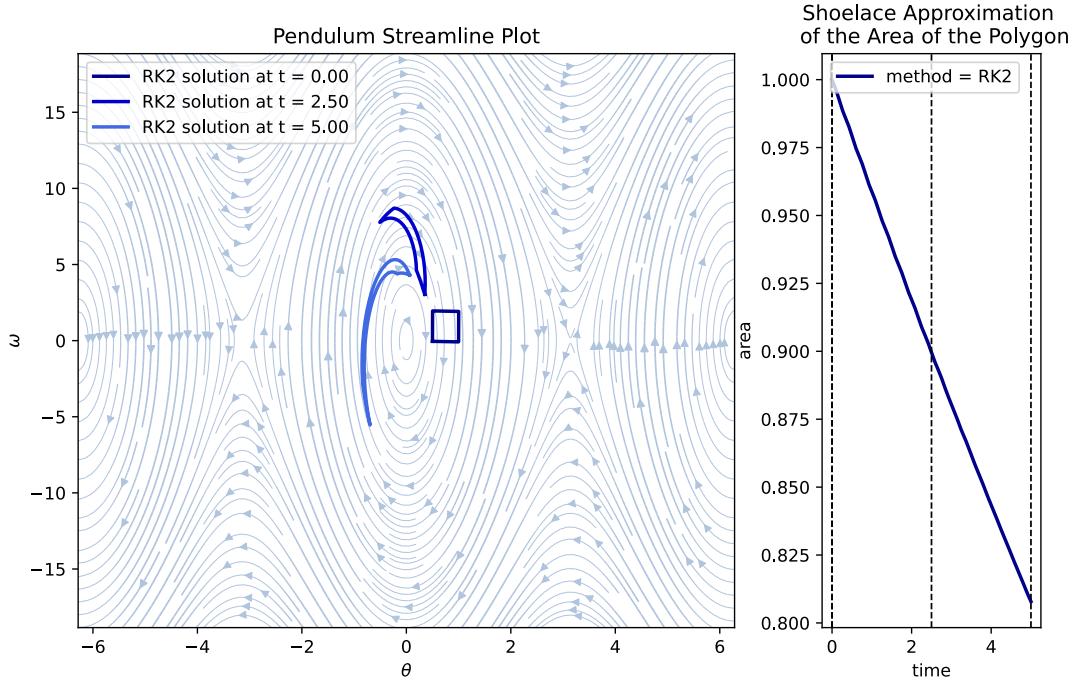


Figure 28: The left panel shows solutions to the pendulum problem at different points in time for different initial values as obtained by the RK2 scheme. The initial values are chosen so that they initially span a square in phase space. The right panel shows the phase space area spanned by the solutions as a function of time.

We need to switch to a whole other class of integrators.

3.7.3 Symplectic integrators to the help

This leads us to geometric integrators which are numerical methods preserving geometric properties of the exact flow of ODEs (Hairer, Wanner, and Lubich, 2006). More specifically for Hamiltonian systems we use **Symplectic Integrators** which produce a flow in phase space that is symplectic just as the flow of the exact solution (Hairer, Wanner, and Lubich, 2006, chapter VI).

Symplectic integrators

- are *structure preserving*
- nearly conserve properties of Hamiltonian systems, e.g. *first integrals* (variables $F(p, q)$ constant along the motion as dictated by the Hamiltonian)
- conserve phase-space (as prescribed by the Liouville theorem)
- more generally conserve all the Poincare invariants

It turns out that preserving symplecticity and energy at the same time is very difficult (Hairer, 2006). However, symplectic integrators still have good energy conservation properties without much long-term drifts. The general idea behind the connection between symplecticity and energy conservation is that geometric properties of the integrator (e.g. phase space conservation) translate into structure preservation on the level of modified equations (Hairer, Wanner, and Lubich, 2006, preface and chapters X through XII).

Problem: Symplectic integrators do not come without caveats:

- Using adaptive time-steps and keeping symplecticity is a problem.
- Non-conservative forces (those which cannot be described by a potential) like radiation forces (depending on velocity not only position) might be important, e.g. for debris in space and particles in planetary rings but then the central requirement of a Hamiltonian system breaks down
- The propagation of floating-point errors might be non-optimal

So the rebound N-body simulation package actually uses IAS15 (implicit integrator with adaptive time-stepping, 15th order) as the default (Rein and Spiegel, 2014).

Note: The default integrator in the N-body-simulator rebound is IAS15, a non-symplectic integrator

3.7.4 Verlet Scheme

Consider we want to solve the Newtonian equation of movement

$$\partial_t^2 \underline{s} = \underline{a}(\underline{s}) \quad (92)$$

- a classical physical problem with a particle at position \underline{s} with velocity \underline{v} and an acceleration \underline{a} that depends only on the position.

Note: In the following we will introduce the Störmer-Verlet, velocity Verlet and Leapfrog scheme which differ in their formulations but are essentially the same. In molecular dynamics it is mostly called Verlet method, in the context of partial differential equations of wave propagation leapfrog method (Hairer, Lubich, et al., 2003).

We can derive a two-step scheme based on a Taylor expansion forward by Δt and backward by $-\Delta t$

$$\begin{aligned} \underline{s}(t + \Delta t) &= \underline{s}(t) + \underline{v}(t)\Delta t + \frac{1}{2}\underline{a}(t)\Delta t^2 + \frac{1}{6}\underline{b}(t)\Delta t^3 + \mathcal{O}(\Delta t^4) \\ \underline{s}(t - \Delta t) &= \underline{s}(t) - \underline{v}(t)\Delta t + \frac{1}{2}\underline{a}(t)\Delta t^2 - \frac{1}{6}\underline{b}(t)\Delta t^3 + \mathcal{O}(\Delta t^4) \end{aligned} \quad (93)$$

with

$$\begin{aligned} \text{position } \underline{s}, \quad \text{velocity } \underline{v}, \quad \text{acceleration } \underline{a} &= -\frac{1}{m} \nabla V(\underline{s}) \\ \text{some potential } V, \quad \text{jerk } \underline{b} &= \frac{d\underline{a}}{dt} \end{aligned} \quad (94)$$

Adding both equations yields a scheme 4th order accurate in time

$$\underline{s}(t + \Delta t) = 2\underline{s}(t) - \underline{s}(t - \Delta t) + \underline{a}(t)\Delta t^2 + \mathcal{O}(\Delta t^4) \quad (95)$$

where terms with odd powers of Δt were eliminated. This is a two-step scheme as for $\underline{s}(t + \Delta t)$, $\underline{s}(t)$ and $\underline{s}(t - \Delta t)$ are used (sometimes this form is called Störmer-Verlet).

Problem: What if we are also interested in calculating the velocity \underline{v} ? What if \underline{a} also depends on \underline{v} as in the Lorentz and we need \underline{v} to calculate \underline{a} ?

From subtracting the equations 93, we can find

$$\underline{v}(t) = \frac{\underline{s}(t + \Delta t) - \underline{s}(t - \Delta t)}{2\Delta t} - \frac{1}{6}\underline{b}(t)\Delta t^2 + \mathcal{O}(\Delta t^3) \quad (96)$$

but this is problematic as

- only accurate to the third order
- an implicit equation as we do not know the future $\underline{s}(t + \Delta t)$ at the current timepoint
- we would need to know the jerk at the current timestep which could itself depend on $\underline{v}(t)$, we could ignore the jerk, but then the accuracy is only $\mathcal{O}(\Delta t^2)$

3.7.4.1 Velocity Verlet algorithm

Idea: Starting again from the Taylor expansion, we omit the jerk and extrapolate the velocity $\underline{v}(t + \Delta t)$ using the average of the accelerations at t and $t + \Delta t$

$$\begin{aligned}\underline{s}(t + \Delta t) &= \underline{s}(t) + \underline{v}(t)\Delta t + \mathcal{O}(\Delta t)^3 \\ \underline{v}(t + \Delta t) &= \underline{v}(t) + \frac{\underline{a}(t) + \underline{a}(t + \Delta t)}{2} + \mathcal{O}(\Delta t)^2\end{aligned}\tag{97}$$

Note: Again, we use $\underline{a}(t + \Delta t)$ to get $\underline{v}(t + \Delta t)$ which is not possible if \underline{a} depends on v . If we would take only $\underline{a}(t)$ instead of the average, we would just have explicit Euler.

In steps we can write

1. calculate the new position: $\underline{s}(t + \Delta t) = \underline{s}(t) + \underline{v}(t)\Delta t + \frac{1}{2}\underline{a}(t)\Delta t^2$
2. update the acceleration: $\underline{a}(t + \Delta t) = -\frac{1}{m} \nabla V|_{\underline{s}(t+\Delta t)}$
3. calculate the new velocity: $\underline{v}(t + \Delta t) = \underline{v}(t) + \frac{\underline{a}(t) + \underline{a}(t + \Delta t)}{2}\Delta t$
4. update the time: $t \rightarrow t + \Delta t$

Introducing half timesteps, we can write the Verlet scheme as a combination of 1st order steps

1. $\underline{v}(t + \frac{1}{2}\Delta t) = \underline{v}(t) + \frac{1}{2}\underline{a}(t)\Delta t$
2. $\underline{s}(t + \Delta t) = \underline{s}(t) + \underline{v}(t + \frac{1}{2}\Delta t)\Delta t$
(note that plugging (1) into (2) yields the previous step (1))
3. $\underline{a}(t + \Delta t) = -\frac{1}{m} \nabla V|_{\underline{s}(t+\Delta t)}$
4. $\underline{v}(t + \Delta t) = \underline{v}(t + \frac{1}{2}\Delta t) + \frac{1}{2}\underline{a}(t + \Delta t)\Delta t$
(note that plugging (1) into (4) yields the previous step (3))
5. $t \rightarrow t + \Delta t$

This is illustrated in figure 29. Here the last step can be called *implicit* as it depends on $\underline{a}(t + \Delta t)$, so a result at the same time as it is supposed to deliver a result at - we have a semi-implicit Euler scheme.

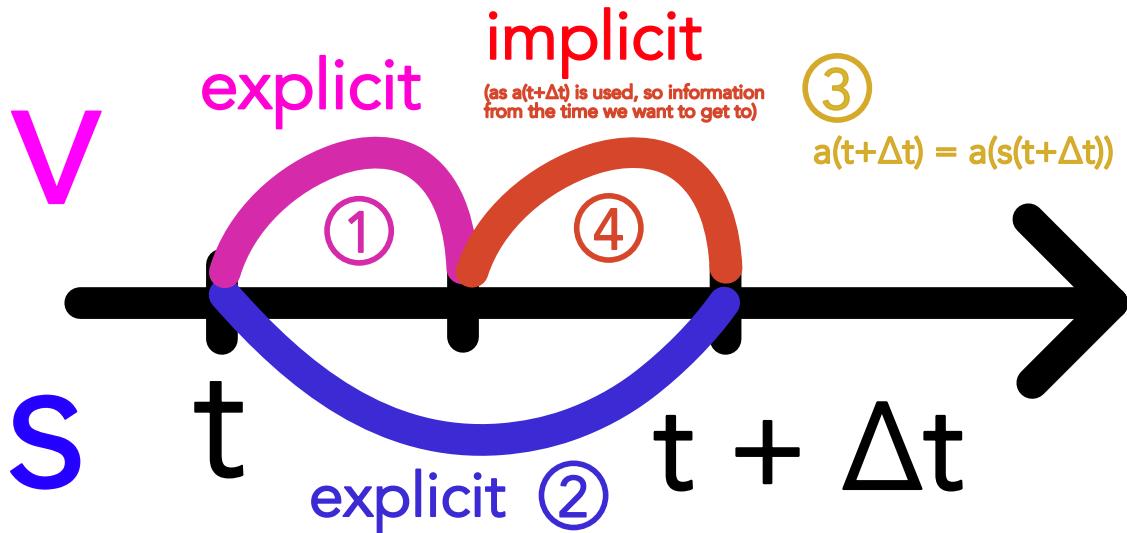


Figure 29: Illustration of the velocity Verlet scheme.

3.7.5 The Leapfrog Method

We will now introduce the Leapfrog scheme and at its hand what it means for a method to be symplectic.

The leapfrog scheme can then be written as

$$\begin{aligned}\underline{s}(t + \frac{1}{2}\Delta t) &= \underline{s}(t - \frac{1}{2}\Delta t) + \underline{v}(t)\Delta t + \mathcal{O}(\Delta t^3) \\ \underline{v}(t + \Delta t) &= \underline{v}(t) + \underline{a}(t + \frac{1}{2}\Delta t)\Delta t + \mathcal{O}(\Delta t^3)\end{aligned}\tag{98}$$

The position and velocity are updated at alternating half time steps as illustrated in figure 30, just as the name suggests (the velocity is "leaping" over the position and vice versa (*Bockspringen*)).

Note: To start the scheme or to save the position and velocity at the same time, a half step needs to be performed, e.g. using half standard explicit Euler.

Leapfrog

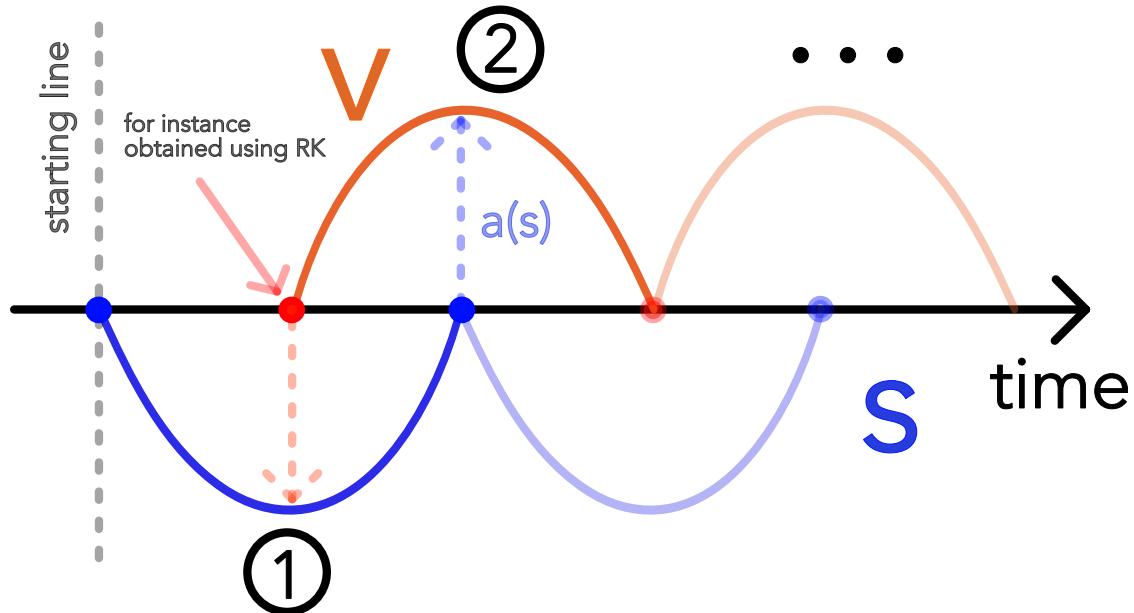


Figure 30: Illustration of the leapfrog scheme.

3.7.5.1 Connection between Leapfrog and Velocity Verlet

We can combine steps 4, 5, 1 in the velocity Verlet scheme in the half-step formulation to get

$$\begin{aligned}\underline{s}(t + \Delta t) &= \underline{s}(t) + \underline{v} \left(t + \frac{1}{2} \Delta t \right) \Delta t \\ \underline{v}(t + \frac{3}{2} \Delta t) &= \underline{v}(t + \frac{1}{2} \Delta t) + \underline{a}(t + \Delta t) \Delta t\end{aligned}\tag{99}$$

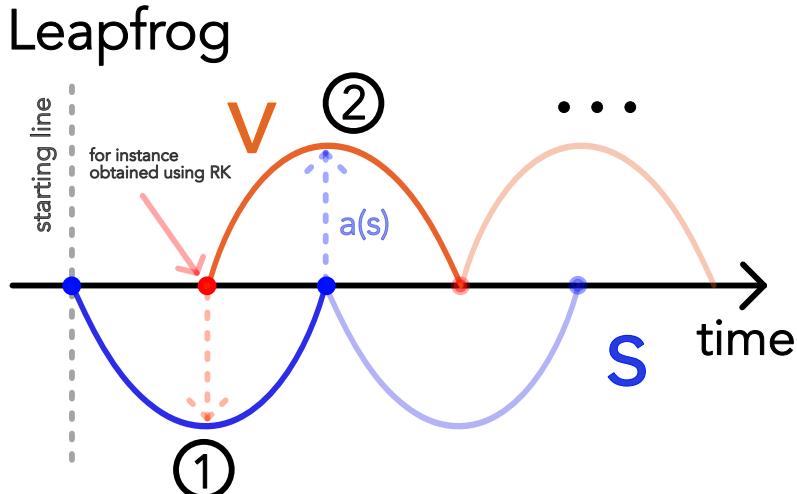
which if we shift everything by $\frac{1}{2} \Delta t$ yields the aforementioned leapfrog scheme. Phrased differently velocity Verlet is leapfrog with Euler-kickoff built-in.

3.7.5.2 Kick-drift-kick and Drift-kick-drift Leapfrog formulations to have velocity and position information at the same time

Problem: One problem of the Leapfrog scheme in this form is that velocity and position information are not available at the same time. This is for instance problematic if we want to calculate the energy which depends on both or if we want to change the step-size adaptively without destroying the interlacement (see figure 31).

Idea: We rearrange and split the interlaced integration by a full time step into a half step in first variable, full step in second variable, half step in first variable.

There are two possible schemes.



Problem: Where to adapt timestep without damaging interlace?

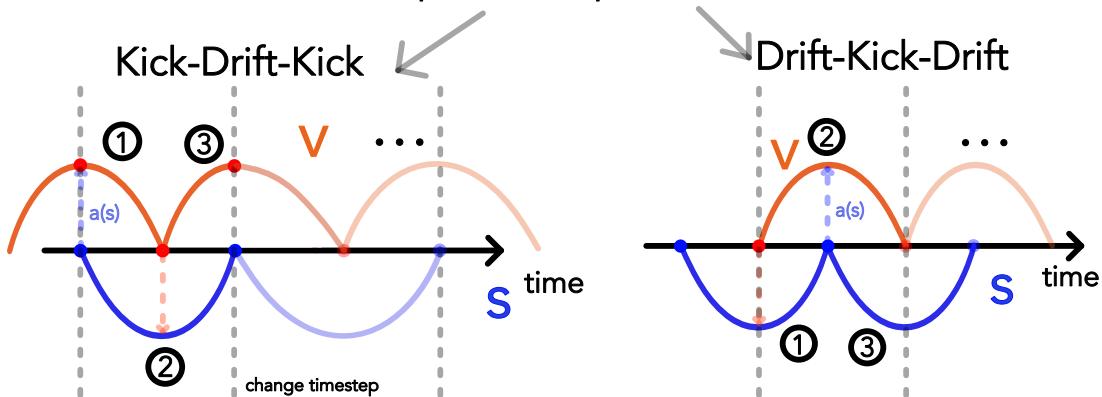


Figure 31: Illustration of the problem of changing the step-size in the leapfrog scheme and the solutions - kick-drift-kick and drift-kick-drift.

In the **kick-drift-kick formulation**, positions are stored at full, velocities at half time steps (equivalent to the velocity Verlet).

$$\begin{aligned}
 \underline{v}(t + \frac{1}{2}\Delta t) &= \underline{v}(t) + \underline{a}(t) \frac{\Delta t}{2} && \text{kick, half-step in first variable} \\
 \underline{s}(t + \Delta t) &= \underline{s}(t) + \underline{v}(t + \frac{1}{2}\Delta t)\Delta t && \text{drift, full-step in second variable} \\
 \underline{v}(t + \Delta t) &= \underline{v}(t + \frac{1}{2}\Delta t) + \underline{a}(t + \Delta t) \frac{\Delta t}{2} && \text{kick, half-step in first variable} \\
 \end{aligned} \tag{100}$$

possibly adapt step-size here

$$t \rightarrow t + \Delta t$$

where a *drift* is a change of the position with constant velocity and a *kick* is a change of the velocity with constant acceleration.

In the **drift-kick-drift formulation**, velocities are stored at full, positions at half time steps.

$$\begin{aligned}
 \underline{s}(t + \frac{1}{2}\Delta t) &= \underline{s}(t) + \underline{v}(t) \frac{\Delta t}{2} && \text{drift, half-step in first variable} \\
 \underline{v}(t + \Delta t) &= \underline{v}(t) + \underline{a}(t + \frac{1}{2}\Delta t)\Delta t && \text{kick, full-step in first variable} \\
 \underline{s}(t + \Delta t) &= \underline{s}(t + \frac{1}{2}\Delta t) + \underline{v}(t + \Delta t) \frac{\Delta t}{2} && \text{drift, half-step in first variable} \\
 &\text{possibly adapt step-size here} \\
 t &\rightarrow t + \Delta t
 \end{aligned} \tag{101}$$

3.7.5.3 Advantages of the Leapfrog scheme

The leapfrog scheme is second order accurate, symmetric (time reversible), symplectic, has good energy conservation properties and is time reversible (proofs in Springel et al., 2023, chapter 2.8).

Second order accuracy may be surprising as we seemingly only Taylor approximate up to the first order - but note the interlacement and connection to velocity Verlet.

3.7.5.4 Leapfrog is symmetric (time reversible)

In terms of the one-step map $\Phi_h : (\underline{s}_n, \underline{v}_n) \rightarrow (\underline{s}_{n+1}, \underline{v}_{n+1})$ symmetry means that $\Phi_{-h}^{-1} = \Phi_h$ so going one step forward and then back leads us back to where we came from, $n \rightleftharpoons n+1$.

We integrate from $(\underline{s}_n, \underline{v}_{n-\frac{1}{2}})$ to $(\underline{s}_{n+1}, \underline{v}_{n+\frac{1}{2}})$ and return.

$$\begin{aligned}
 \underline{s}_{\text{fin}} &= \underline{s}_{n+1} - \underline{v}_{n+\frac{1}{2}} \Delta t = \underline{s}_n + \underline{v}_{n+\frac{1}{2}} \Delta t - \underline{v}_{n+\frac{1}{2}} \Delta t = \underline{s}_n \\
 \underline{v}_{\text{fin}} &= \underline{v}_{n+\frac{1}{2}} - \underline{a}_n \Delta t = \underline{v}_{n+\frac{1}{2}} + \underline{a}_n \Delta t - \underline{a}_n \Delta t = \underline{v}_{n-\frac{1}{2}}
 \end{aligned} \tag{102}$$

The flow is also reversible - inverting the direction of initial velocity does not change the solution trajectory, it just inverts the direction (if the acceleration only depends on the position).

So leapfrog is symmetric (time-reversible), different from e.g. explicit Euler (see figure 32).

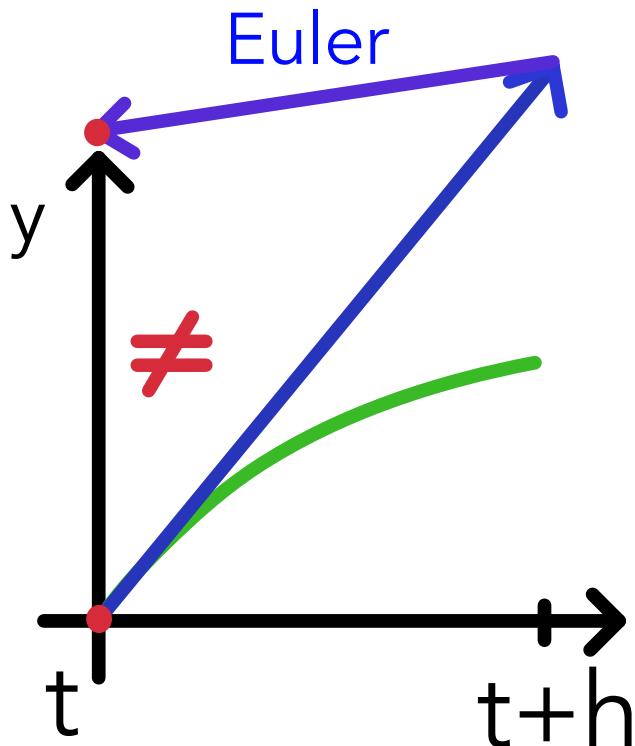


Figure 32: Explicit Euler is not symmetric (time-reversible).

Advantages of symmetric methods: Symmetric methods applied to *integrable and near-integrable reversible systems* share similar properties to symplectic methods applied to *(near)-integrable Hamiltonian systems*: linear error growth and long-time near-conservation of first integrals. For a non-reversible system, a symmetric but non-symplectic method (e.g. Lobatto IIIB) will have no good conservation properties though.^a

^aA system of differential equations

$$\partial_t \underline{u} = \underline{f}(\underline{u}, \underline{v})$$

$$\partial_t \underline{v} = \underline{g}(\underline{v}, \underline{v})$$

is reversible if

$$\underline{f}(\underline{u}, -\underline{v}) = -\underline{f}(\underline{u}, \underline{v})$$

$$\underline{g}(\underline{v}, -\underline{v}) = \underline{g}(\underline{v}, \underline{v})$$

Note: Not every symplectic method is symmetric. For instance symplectic Euler is symplectic but not symmetric.

3.7.5.5 Symplecticity of the leapfrog scheme I: Intuition and Meaning

Symplecticity (so area conservation in phase space) is illustrated at the hand of the pendulum in figure 33 and energy conservation at the hand of the two-body-problem in figure 34. The area in phase space does not change at all while the energy shows small fluctuations but

no overall drift (compare Hairer, Lubich, et al., 2003, theorem 5.5). The angular momentum is exactly conserved in the leapfrog solution to the two-body-problem (details on the conservation of specific *quadratic first integrals* can be found in Hairer, Lubich, et al., 2003, theorem 3.5). As visible in the changing orientation of the orbit in figure 34 the leapfrog scheme does not preserve the orientation of the Runge-Lenz vector.

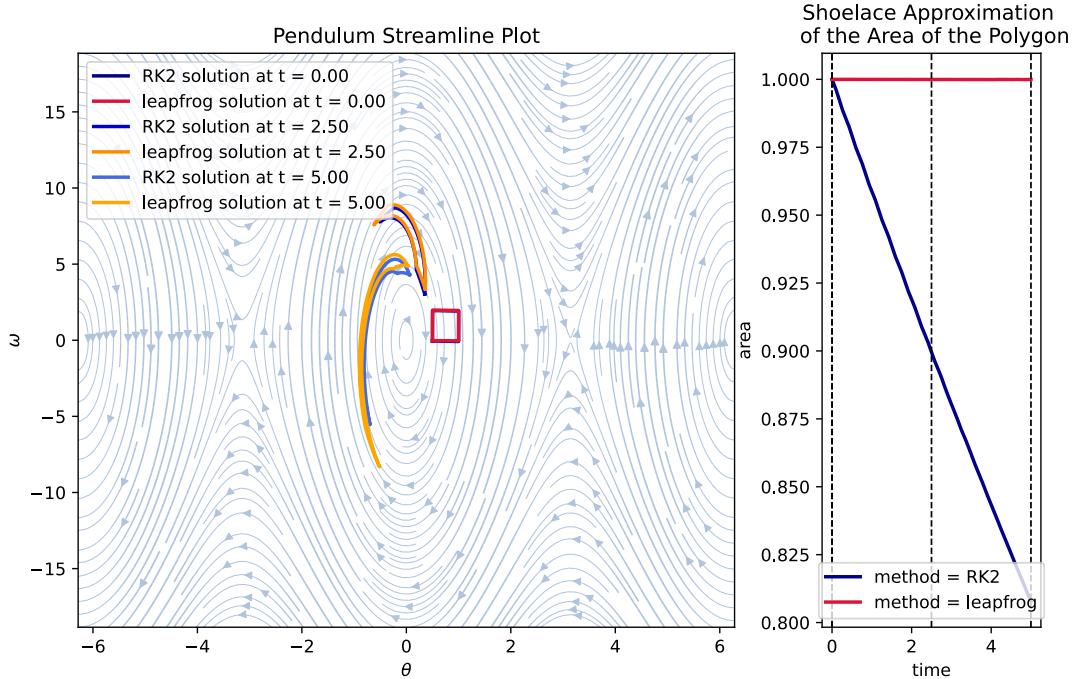


Figure 33: The same situation as in figure 28 is shown but now with the result of the leapfrog method added. The leapfrog scheme preserves the area in phase space while the RK2 scheme does not.

3.7.5.6 Symplecticity of the leapfrog scheme II: Proof

Consider the separable Hamiltonian

$$\begin{aligned} H(q, p) &= H_{\text{kin}}(p) + H_{\text{kin}}(q) \\ &\stackrel{\text{here}}{=} \underbrace{\frac{p^2}{2m}} + U(q) \end{aligned} \tag{103}$$

Procedure: We solve both parts of the Hamiltonian separately (operator splitting) and construct Leapfrog as the concatenation of these solutions (proving symplecticity) and then calculate an error Hamiltonian.

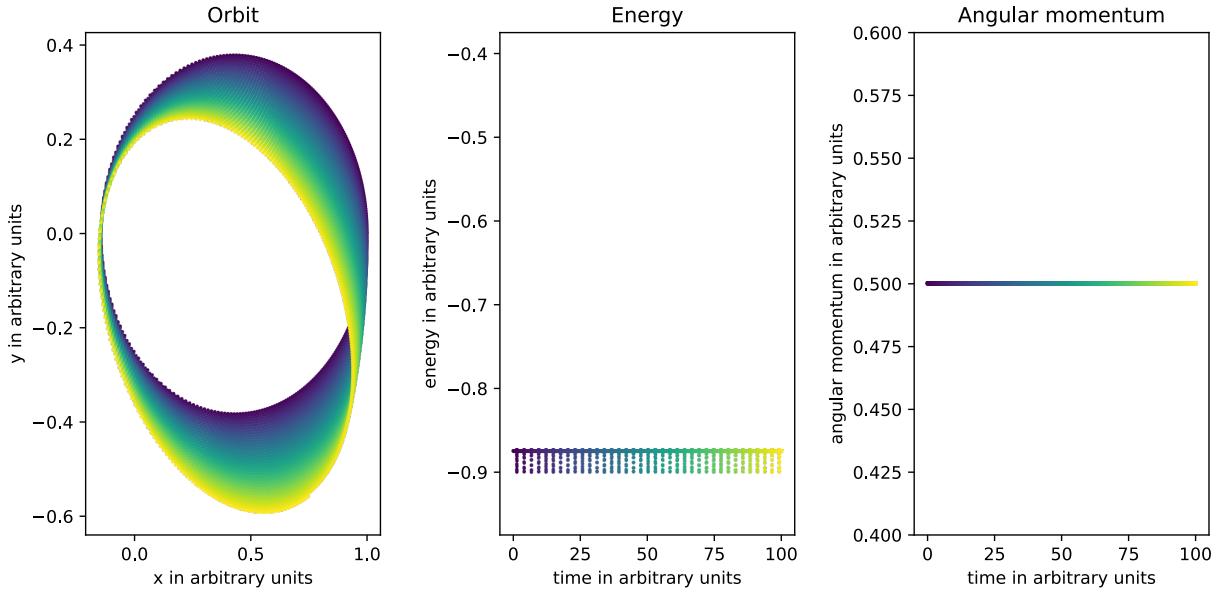


Figure 34: Numerical solution of the two-body-problem using the leapfrog method in the kick-drift-kick scheme. The left panel shows the orbit, the central one the energy and the right one the angular momentum. Time is also encoded in the form of color, going from a dark blue to yellow.

Operator splitting From the Hamiltonian equations of the kinetic and potential part, we find update steps.

For the kinetic part H_{kin} we find

$$\left. \begin{array}{l} \partial_t q = \partial_p H_{\text{kin}} = \frac{p}{m} \\ \partial_t p = -\partial_q H_{\text{kin}} = 0 \end{array} \right\} \rightarrow \left\{ \begin{array}{l} q^{(n+1)} = q^{(n)} + \frac{p^{(n)}}{m} \Delta t \\ p^{(n+1)} = p^{(n)} \end{array} \right. \quad (104)$$

and for the potential part

$$\left. \begin{array}{l} \partial_t q = \partial_p H_{\text{pot}} = 0 \\ \partial_t p = -\partial_q H_{\text{pot}} = -\partial_q U \end{array} \right\} \rightarrow \left\{ \begin{array}{l} q^{(n+1)} = q^{(n)} \\ p^{(n+1)} = p^{(n)} - \partial_q U \Delta t \end{array} \right. \quad (105)$$

Note: Independent of Δt the schemes found are exact; and symplectic as brought forward by Hamiltonians.

Constructing leapfrog Let $\phi_{\Delta t}(H)$ describe making a step Δt governed by H in phase-space (a time evolution operator). Then leapfrog (kick-drift-kick version) is given as

$$\phi_{\Delta t}(H) = \phi_{\frac{\Delta t}{2}}(H_{\text{pot}}) \odot \phi_{\frac{\Delta t}{2}}(H_{\text{kin}}) \odot \phi_{\Delta t}(H_{\text{pot}}) \quad (106)$$

so Leapfrog is symplectic as a concatenation of symplectic operators, so

- there is no secular (long-lasting, non-oscillatory) drift in e.g. the Energy of e.g. the Kepler orbits
- the longer the timespan to simulate, the more it makes sense to use leapfrog over Runge Kutta schemes (e.g. the explicit Euler scheme always overshoots the orbits leading to increasing total energy and unbound states)

Note: »Yes, symplectic integrators do not exactly conserve energy. It is a common misconception that they do. What symplectic integrators actually do is solve for a trajectory which rests on a symplectic manifold that is perturbed from the true solution's manifold by the truncation error. This means that symplectic integrators do not experience (very much) longtime drift, but their orbit is not exactly the same as the true solution in phase space, and thus you will see differences in energy that tend to look periodic. There is a small drift which grows linearly and is related to floating-point error, but this drift is much less than standard methods. This is why symplectic methods are recommended for longtime integration.« (Rackauckas, Sciemon, et al., 2022)

Error Hamiltonian Indeed, it turns out that the leapfrog scheme exactly solves the modified Hamiltonian

$$H_{\text{leap}} = H + H_{\text{err}}, \quad H_{\text{err}} \propto \frac{\Delta t^2}{12} \left\{ \{H_{\text{kin}}, H_{\text{pot}}\}, H_{\text{kin}} + \frac{1}{2} H_{\text{pot}} \right\} + \mathcal{O}(\Delta t^3) \quad (107)$$

where H_{kin} and H_{pot} are the kinetic and potential part of the original Hamiltonian (Springel et al., 2023, chapter 2.8). The curly brackets denote the Poisson bracket.

Notes on the derivation of the Error Hamiltonian The time evolution of a phase space function $F(p, q)$ under the flow generated by a Hamiltonian H fulfills $-\partial_t F = -\{H, F\} = -\hat{H}F$ and similar to the time-development operator in Quantum Mechanics, we can write

$$F(t) = \exp\left(\hat{H}t\right)F(0) \quad (108)$$

and for the leapfrog scheme with $H_{\text{leap}} = H + H_{\text{err}}$, $H = H_{\text{kin}} + H_{\text{pot}}$ we have

$$\exp((H + H_{\text{err}})\Delta t) = \exp\left(H_{\text{pot}} \frac{\Delta t}{2}\right) \exp(H_{\text{kin}} \Delta t) \exp\left(H_{\text{pot}} \frac{\Delta t}{2}\right) \quad (109)$$

where using the Baker-Campbell-Hausdorff formula lets us find H_{err} .

3.8 Extrapolation method: Bulirsch-Stoer algorithm

Our goal is still to obtain highly accurate and cheap solutions to ODEs. The ingredients of Bulirsch-Stoer are

1. we integrate across a whole interval with length H multiple times with a sequence of decreasing substep-sizes h_j each yielding a final result $F(h_j)$
2. we extrapolate $F(h)$ to $h = 0$ asking ourselves *What would be the solution, if we had taken infinitely many, infinitely small steps?*, for instance using polynomial extrapolation or Richardson extrapolation³ with rational functions

with the method being most-appropriate for differential equations containing smooth (cheap to evaluate) functions. An illustration is given in figure 35.

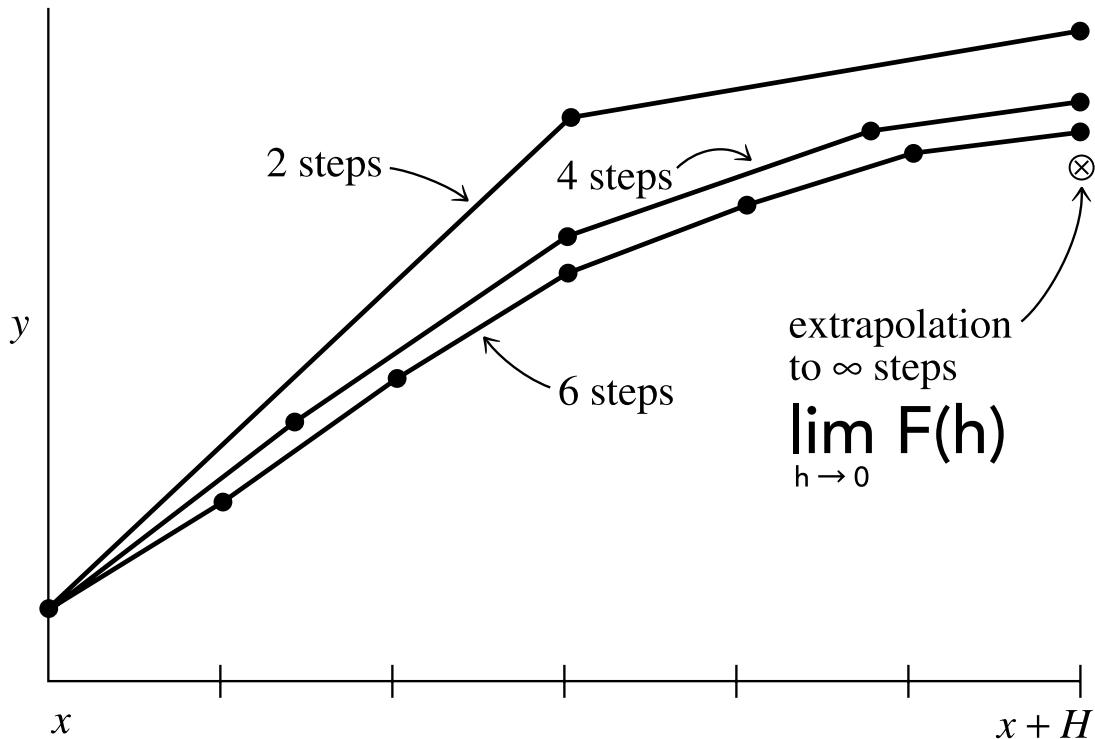


Figure 35: Illustration of the Bulirsch-Stoer algorithm.

While the idea of this method is very beautiful, its usage is disputed, with e.g. W. Van Snyder writing: extrapolation methods are almost always substantially inferior to Runge-Kutta, Taylor's series, or multistep methods.

A single step taking us from x to $x + H$ (large distance H) consists of many substeps using the modified midpoint rule.

³A sequence acceleration method to improve the convergence of a sequence $F^* = \lim_{h \rightarrow 0} F(h)$

3.8.1 Basic integration method | second order method with $\mathcal{O}(h^2)$; midpoint rule → modified midpoint rule

The midpoint rule is given by

$$\begin{aligned} k_1 &= f(y_i, x_i) \\ k_2 &= f\left(y_i + \frac{h}{2}k_1, x_i + \frac{h}{2}\right) \\ x_{i+1} &= x_i + h \\ y_{i+1} &= y_i + hk_2 + \mathcal{O}(h^3) \end{aligned} \tag{110}$$

as illustrated in figure 36.

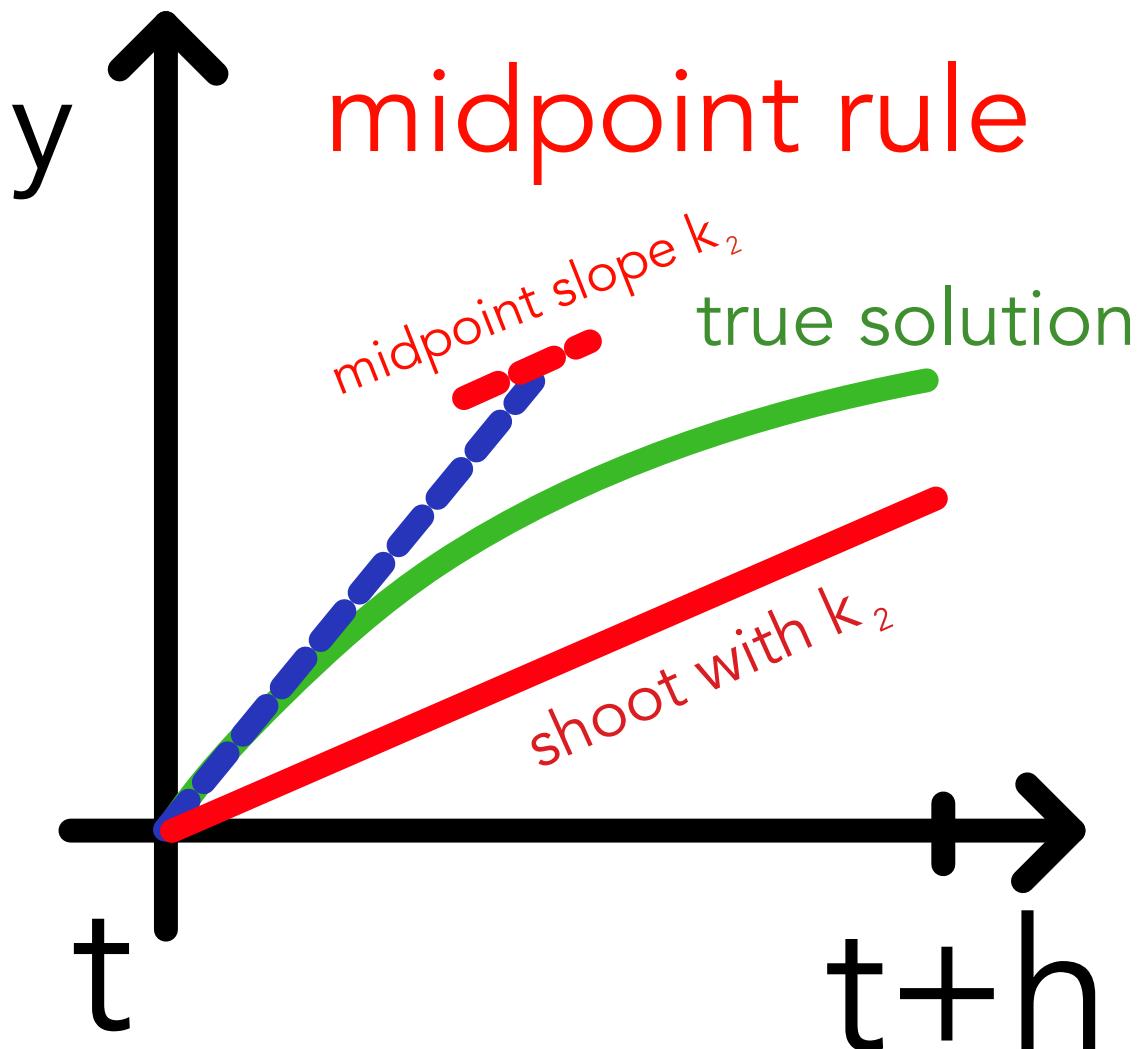


Figure 36: Illustration of the midpoint rule.

3.8.1.1 Modified midpoint rule

We advance from x to $x + H$ using n substeps of size $h = \frac{H}{n}$. Except for the first and last step, we advance using the midpoint rule.

Advantage of the midpoint rule over RK2: We only need one evaluation of f per step h instead of two.

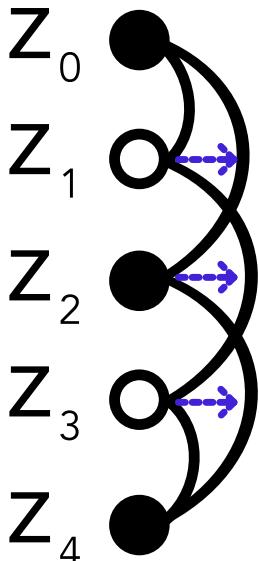


Figure 37:
Modified mid-
point rule.

$$\begin{aligned}
 z_0 &= y(x) \\
 z_1 &= z_0 + hf(z_0, x) \text{ not midpoint} \\
 z_2 &= z_0 + 2hf(z_1, x + h) \text{ midpoint with stepsize } 2h \\
 z_3 &= z_1 + 2hf(z_2, x + 2h) \text{ midpoint with stepsize } 2h \\
 &\vdots \\
 z_m &= z_{m-2} + 2hf(z_{m-1}, x + (m-1)h) \\
 z_{m+1} &= z_{m-1} + 2hf(z_m, x + mh), \quad m = 4, \dots, n-1 \\
 &\vdots \\
 y(x + H) &\approx y_n = \frac{1}{2} \left[z_n + \underbrace{z_{n-1} + hf(z_n, x + H)}_{\text{euler Step from } z_{n-1}} \right]
 \end{aligned}$$

(111)

3.8.1.2 Combining modified midpoint calculations with different h ; advantage of modified midpoint

Remember that the midpoint rule is of 2nd order so the error when covering an interval H with multiple steps is $\mathcal{O}(h^2)$. Central to the advantage of the modified midpoint rule is (not proven here)

$$y(x + H) - y_n = \sum_{i=1}^{\infty} \alpha_i h^{2i} \quad (112)$$

so the error between the true $y(x + H)$ and our numerical result y_n expressed as a power series in h only contains even powers of h . **We can therefore combine results with different step-sizes to gain two orders at a time.**

Example Let us combine a version with n steps h and one with $2n$ steps $2h$.

$$\begin{aligned} n : \quad & y(x + H) - y_n = \alpha_1 h^2 + \alpha_2 h^4 \\ 2n : \quad & y(x + H) - y_{2n} = \alpha_1 \left(\frac{h}{2}\right)^2 + \alpha_2 \left(\frac{h}{2}\right)^4 \end{aligned} \quad (113)$$

from which we obtain

$$y(x + H) = \frac{4y_{2n} - y_n}{3} + \mathcal{O}(h^4) \quad (114)$$

which is 4th order accurate like RK4 but at less cost (function evaluations).

3.8.1.3 What extrapolation nodes to choose? - how to increase n (or rather decrease h)

Let us remember

$$F(h_n) = y_n \quad \text{with } n = \frac{H}{h} \quad (115)$$

We cross the large interval H using multiple substeps multiple times with decreasing substep size. Each iteration delivers a result $F(h_n)$ and we have already seen that such results can be combined smartly - and extrapolation to $h \rightarrow 0$ is even better. But what evaluation nodes for $F(h)$ should we choose?

$$\begin{aligned} \text{Romberg : } & n = [2, 4, 8, 16, \dots, 1024] \\ \text{Bulirsch : } & n = [2, 4, 6, 8, 12, 16, \dots, 96] \\ \text{Deufelhard : } & n = [2, 4, 6, 8, 10, \dots, 24] \end{aligned} \quad (116)$$

3.8.1.4 How to extrapolate from multiple $F(h_n)$ to the limit $h \rightarrow 0$?

One option is polynomial interpolation (two points define a line, ..., n points define a polynomial of (max) degree $n - 1$), so we do polynomial regression and evaluate it at zero.

There is also the approach to use rational functions, which can also capture poles and divergence regions between the interpolation points (which polynomials will never do), so

$$R_{m+1}(x) = \frac{P_\mu(x)}{Q_\nu(x)} = \frac{p_0 + p_1 x + p_2 x^2 + \dots + p_\mu x^\mu}{q_0 + q_1 x + q_2 x^2 + \dots + q_v x^v}, \quad m + 1 = v + \mu + 1 \quad (117)$$

Hairer, Wanner, and Nørsett, 1993 write: »Many authors in the sixties claimed that it is better to use rational functions instead of polynomials. Later numerical experiments (Deuflhard 1983) showed that rational extrapolation is nearly never more advantageous than polynomial extrapolation.«. One reason I can imagine is that the more complex rational model is more unstable and harder to appropriately fit.

3.9 Predictor-corrector methods

Multistep idea: While a one-step method only uses the differential value and the ODE itself in a step, multistep methods also include information from previous steps (e.g $x, x-h, x-2h$) to obtain better estimates for the next step ($x+h$). The method is e.g. kicked off using Runge-Kutta steps.

Let us start by writing an advance in an ODE $\partial_x y = f(y(x), x)$ as

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} f(x', y(x')) dx' \quad (118)$$

the problem is that different from an integration problem, we would need to know $y(x)$ to use this formula to calculate $y(x)$ or rather y_{n+1} - so what have we won?

Consider we are in the multistep setting and already have approximations y_n, y_{n-1}, \dots at x_n, x_{n-1}, \dots . We can then approximate $f(x, y)$ by a polynomial passing through those points and yield

$$y_{n+1} = y_n + h \cdot (\beta_0 f(x_{n+1}, y_{n+1}) + \beta_1 f(x_n, y_n) + \beta_2 f(x_{n-1}, y_{n-1}) + \dots) \quad (119)$$

But wait - we used y_{n+1} in the RHS which we do not know. We could get an explicit scheme by setting $\beta_0 = 0$ but the formula screams fixpoint iteration (as in Picard iteration) - **predictor-corrector method**.

1. **predictor step:** obtain a good estimate for y_{n+1}
2. **corrector step(s):** plugging the result of the predictor step into eq. 119 gives an improved estimate for y_{n+1}

For correction to make sense, the first prediction must be sufficiently good.

3.9.1 One-step predictor-corrector method: RK2 and $P(EC)^k$

Indeed standard RK2 is a predictor-corrector method

$$\begin{aligned} k_1 &= f(y_n, x_n) \\ k_2 &= f\left(\underbrace{y_n + hk_1}_{\text{predictor}}, x_n + h\right) \\ y_{n+1} &= \underbrace{y_n + \frac{h}{2}(k_1 + k_2)}_{\text{corrector}} + \mathcal{O}(h^3) \end{aligned} \quad (120)$$

Different notation and $P(EC)^k$ method We write the predictor P step as

$$\tilde{y}_{n+1,[0]}^P = y_n + hf(y_n, x_n) \quad (121)$$

and can then write the evaluation / corrector step EC as

$$\tilde{y}_{n+1,[1]}^{EC} = y_n + \frac{h}{2} [f(y_n, x_n) + f(\tilde{y}_{n+1,[0]}^P, x_{n+1})] \quad (122)$$

which we can repeat

$$\begin{aligned} \tilde{y}_{n+1,[2]}^{EC} &= y_n + \frac{h}{2} [f(y_n, x_n) + f(\tilde{y}_{n+1,[1]}^{EC}, x_{n+1})] \\ \tilde{y}_{n+1,[k]}^{EC} &= y_n + \frac{h}{2} [f(y_n, x_n) + f(\tilde{y}_{n+1,[k-1]}^{EC}, x_{n+1})] \end{aligned} \quad (123)$$

until convergence $|\tilde{y}_{n+1,[k]}^{EC} - \tilde{y}_{n+1,[k-1]}^{EC}| \leq \epsilon_0$ (some error tolerance ϵ_0) where our final approximation is $y_{n+1} = \tilde{y}_{n+1,[k]}^{EC}$. This is the $P(EC)^k$ method.

3.9.2 4th order Adams-Bashforth-Moulton

Here we used the multistep principle as introduced above. In the 4th order Adams-Bachforth-Moulton approach we use three previous timesteps. It is a 4th order method.

The predictor step has weights designed so it gives the correct result for cubic polynomials

$$\tilde{y}_{n+1} = y_n + \frac{h}{24} [55f(y_n, x_n) - 59f(y_{n-1}, x_{n-1}) + 37f(y_{n-2}, x_{n-2}) - 9f(y_{n-3}, x_{n-3})] + \mathcal{O}(h^5) \quad (124)$$

the evaluation / corrector step EC is

$$y_{n+1} = y_n + \frac{h}{24} [9f(\tilde{y}_{n+1}, x_{n+1}) + 19f(y_n, x_n) - 5f(y_{n-1}, x_{n-1}) + f(y_{n-2}, x_{n-2})] + \mathcal{O}(h^5) \quad (125)$$

containing \tilde{y}_{n+1} (*implicit*) and can be repeated for higher accuracy (plug in y_{n+1} instead of \tilde{y}_{n+1} in the next EC step). We start the scheme with three RK steps.

3.10 Shooting | adapting parameters until boundary conditions are fulfilled

3.10.1 Remark on ODE solutions in phase space

There are infinitely many solutions to an ODE but only one unique to a initial value problem. Those solutions are streamlines through phase space that

- do not start / end
- do not cross

3.10.2 Exemplary Shooting Problem

Consider the motion of a projectile from a canon, given by

$$\partial_t^2 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ -g \end{pmatrix} - b \partial_t \begin{pmatrix} x \\ y \end{pmatrix} \quad (126)$$

where the canon sits at $(0, 1)$ and shoots with a give velocity v_{canon} at an angle α to the horizontal. Our aim is hitting a target at $(x_{\text{target}}, y_{\text{target}})$ (boundary condition). This is illustrated in figure 38.

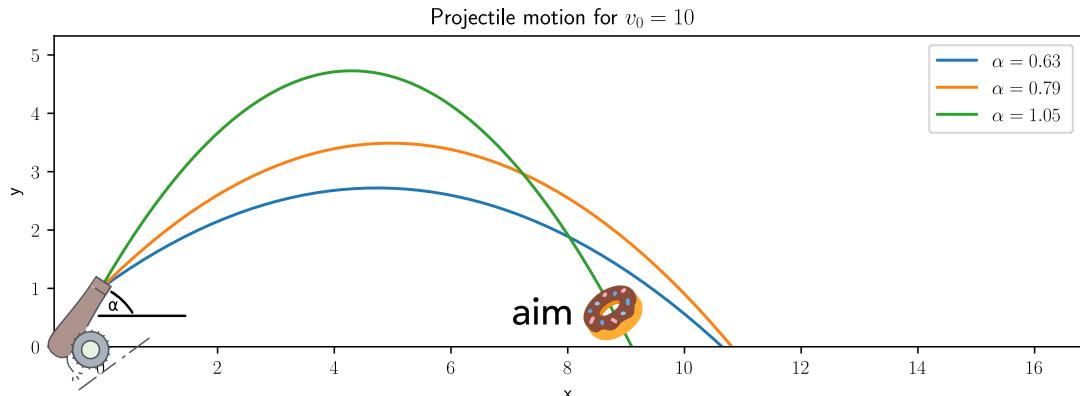


Figure 38: Illustration of the shooting problem. Trajectories gained by converting the system to first order ($q = (x, y, v_x, v_y)$, $f = (v_x, v_y, -bv_x, -g - bv_y)$) and then applying RK2.

3.10.3 Shooting

In the shooting method a boundary value problem is reduced to an initial value problem. We solve the initial value problem for different choices of parameters until the given boundary conditions are satisfied – we shoot trajectories in different directions from one boundary until we find one that hits the other boundary condition. For a neutron star we could know the central density and density at the surface and seek some process parameters.

4 Simulation of Physical Systems - from Quantum Mechanics to Fluid Dynamics

4.1 Different levels of modelling from Quantum Mechanics to Kinetic Gas theory

Our aim is simulating real-world physical systems with two of our fundamental tools being modeling and model order reduction. When we model a fluid, we will probably not model every particle let alone the underlying quantum mechanics⁴.

Starting off with Quantum Mechanics Consider we would model the whole wave function of a system of particles $\Phi(\underline{x}_1, \dots, \underline{x}_N)$ (giving the joint probability $|\Phi|^2$ of finding all particles at $\underline{x}_1, \dots, \underline{x}_N$ with single particle probabilities being the margins) with possibly more degrees of freedom (like spin) and evolve it using e.g. the Schrödinger equation with particle interaction being represented in the potential of the Hamiltonian. If we would discretize each dimension with 1000 cells, the simulation would have 1000^{3N} cells - quickly infeasible (we model a too large probability space). First reduction steps could be

- use symmetries, e.g. particles of a type are indistinguishable reducing the degrees of freedom
- in given potentials approximate the particle wave as a sum of standing waves and evolve the coefficients
- ...

Molecular dynamics Quantum mechanical simulation is infeasible (and unnecessary) for larger systems. A first step is to decouple the electrons and protons (Born-Oppenheimer approximation) as the electrons move on much quicker timescales and describe the atoms as localized particles in phase-space. Based on the positions of the nuclei forming a fixed potential, we can calculate the lowest energy wave function of the electrons (which will occur in *no time* for the protons), from there forces on the nuclei with which we move them and so on. The next step is to model the interatomic interactions using inter-atomic potentials (e.g. Morse, Lenard-Jones) with the potential parameters for instance adapted to match quantum mechanical simulations or for a system as a whole to portray correct behavior.

⁴Except for instance when we want to analyse shocks in Warm Dense Matter in Neutron Stars or inertial confinement fusion based on Quantum hydrodynamics (Graziani et al., 2022)

Kinetic Gas theory Let us assume, the "free flight" between interactions is much larger than the interaction time. We can model *molecule-particles* with some effective interaction radius which only exhibit collisions. Note that while when modeling the potentials, at a sufficiently large collision parameter, particles do a kind of slingshot maneuver (attractive potential), which can never be modelled in a pure collision system. But this can be fine - with the right parameters our system can work globally in spite of local differences.

4.2 From a classical particle description to the Boltzmann equation

Consider we start of with classical point-like particles in phase space (e.g. the molecules) $\{\underline{x}_i, \underline{p}_i\}_{i=1}^N$ with a force term

$$\frac{d}{dt} \underline{p}_i = \underbrace{\underline{f}(\underline{x}_i(t), t)}_{\text{e.g.}} = q_i \cdot \left(\underline{E}_m(\underline{x}_i(t), t) + \frac{1}{m_i} \underline{p}_i(t) \times \underline{B}_m(\underline{x}_i(t), t) \right) \quad (127)$$

for
ions

where the forces depend on the particle positions and momenta themselves (e.g. via Maxwell's equations where the particle positions inform the charge density and thus the electric field and the velocities the currents and thus the magnetic field).

For as many particles as in a fluid (a mol has $\sim 6 \cdot 10^{23}$ particles and e.g. water has roughly 18 grams per mol) this is still unfeasible for simulation. It turns out to be smarter to model a phase space density. The exact classical phase space density is

$$\mathcal{F}(\underline{x}, \underline{p}, t) = \sum_{i=1}^N \delta(\underline{x} - \underline{x}_i) \delta(\underline{p} - \underline{p}_i) \quad (128)$$

Phase space conservation (Liouville theorem) (previously visualized for the pendulum) means that

$$\frac{d}{dt} \mathcal{F}(\underline{x}, \underline{u}, t) = \partial_t \mathcal{F} + \underline{v} \nabla_{\underline{x}} \mathcal{F} + \underline{\mathcal{A}} \nabla_{\underline{u}} \mathcal{F} = 0 \quad (129)$$

with the local acceleration $\underline{\mathcal{A}}$ following from the local force and mass. Let us use a mean phase space density and acceleration instead.

$$\begin{aligned} \mathcal{F}(\underline{x}, \underline{u}, t) &= f(\underline{x}, \underline{p}, t) + \delta \mathcal{F}(\underline{x}, \underline{u}, t), & f(\underline{x}, \underline{u}, t) &= \langle \mathcal{F}(\underline{x}, \underline{u}, t) \rangle \\ \underline{\mathcal{A}}(\underline{x}, \underline{u}, t) &= \underline{a}(\underline{x}, \underline{u}, t) + \delta \underline{\mathcal{A}}(\underline{x}, \underline{u}, t), & \underline{a}(\underline{x}, \underline{u}, t) &= \langle \underline{\mathcal{A}}(\underline{x}, \underline{u}, t) \rangle \end{aligned} \quad (130)$$

which we have also done for the acceleration, separating a mean effect on a *fluid parcel* from the direct particle-particle interactions. With this we get

$$\partial_t f + \underline{v} \cdot \nabla_{\underline{x}} f + \underline{a} \cdot \nabla_{\underline{u}} f = -\langle \delta \underline{\mathcal{A}} \cdot \nabla_{\underline{u}} \delta \mathcal{F} \rangle =: \frac{df}{dt} \Big|_c \quad (131)$$

which is the Boltzmann equation, where we identified the local fluctuations with collisions $\frac{df}{dt} \Big|_c$ from a kinetic gas theory perspective.

4.3 Emergence of irreversibility in the Boltzmann equation

Consider a classical simulation of colliding spheres (interaction time \ll free flight time), where we start out with the particles concentrated at the center of our box. As of their thermal motion, the particles will spread out and fill the box. Now consider we start with this end state and reverse all velocities - the particles will clump up - anti-dissipation. Such things can happen microscopically, they are simply very unlikely.

Something unlikely microscopically is virtually impossible macroscopically.

But based on the Boltzmann equation, this will never happen - the Boltzmann equation is *irreversible* and will never show anti-dissipation.

»The derivation of the Boltzmann equation (BE) from the Hamiltonian equations of motion of a hard spheres gas is a key topic on irreversibility (Sklar 1993, p.32; Uffink 2007, Section 4). Although the Hamiltonian equations of motion are invariant under time reversal, the BE is not. Moreover, this equation allows us to derive the H-theorem, which states that a function H monotonically decreases with time, and thus, that the minus-H function increases, in agreement with the second law of thermodynamics. The derivation of the BE thus raises the question of irreversibility, since this equation exhibits irreversibility even though the microscopic description of the gas is based on reversible equations.« from Ardourel, 2017 where the emergence of irreversibility is discussed in detail.

We can try to understand the emergence of irreversibility based on going from the exact information of the positions and momenta of the particles to an averaged phase space density:

»In other words, when the particles are described by the Boltzmann equation, our knowledge is incomplete, since the positions and momenta of all the particles remain unknown, in contrast to the description by means of Hamilton canonical equations or Liouville equation. And this lack of knowledge makes the evolution of the one-particle distribution function f irreversible. The irreversibility is then explicitly expressed by the collision integral in the Boltzmann equation. The second law of thermodynamics thus emerges from completely reversible dynamics when our description is incomplete (not seeing all positions and momenta of the particles).« from Kincl and Pavelka, 2023

5 Basic Fluid Dynamics

Fluids (gases or liquids⁵) react to tangential (aka shear) stress with flow, a deformation rate depending on the viscosity, as opposed to solids which deform. Many systems from galaxies to lab plasmas can - on the right scale - be successfully modelled as fluids.

5.1 Basic notes on fluid description - the fluid from the view of a parcel - macroscopic fluid view

Physical systems can be described on different level: from a wave function in quantum physics, over a collection of point-like particles in classical physics to a **continuous fluid**.

5.1.1 When is a fluid description valid?

We want to describe the fluid in terms of macroscopic position and time dependent quantities like: mass density ρ , temperature T , velocity $\underline{u} = \underline{v} + \underline{w}$ where $\underline{v} = \langle \underline{u} \rangle$ is the mean (bulk) velocity of the local fluid element and \underline{w} is the random velocity defining the temperature.

5.1.1.1 Connection between temperature and random movement

As of the equipartition theorem, each exitable degree of freedom adds $\frac{k_B T}{2}$ to the internal energy, so

$$\left\langle \frac{1}{2} m w_i^2 \right\rangle = \frac{1}{2} k_B T \quad \rightarrow \quad \langle \|\underline{w}\|^2 \rangle = \frac{3k_B T}{m} \quad (132)$$

5.1.1.2 Continuum Hypothesis

Q: When does it make sense to describe a physical system by a continuous fluid? A: When we can construct a volume small compared to the region of the fluid but big compared to the molecular free path. When we consider a fluid in terms of **parcels** with constant mass and particle number

$$\text{parcel mass } m_p = \int_V \rho dV, \quad \text{parcel volume } V, \quad \text{parcel density } \rho \quad (133)$$

we postulate that every fluid quantity (bulk quantity of the parcel) tends to a limit as the size of the volume approaches zero, before molecular fluctuation kicks in (**continuum hypothesis**), see figure 39.

⁵In gases, time between interactions is so much longer than time of interactions that they can often be described by kinetic gas theory. At higher temperatures there are more collisions and the gas is more viscous while in liquids at higher temperatures *bonds are easier to break* so the liquid is less viscous.

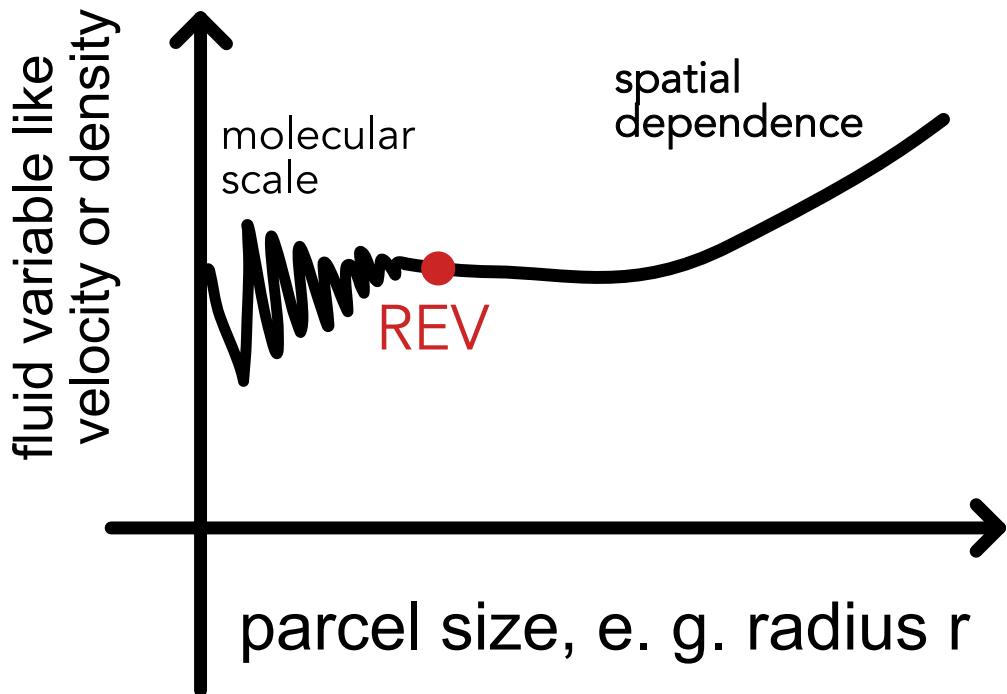


Figure 39: Continuum hypothesis; REV = representative elementary volume

A fluid description is thus justified if

$$\text{Knudsen number } Kn = \frac{\text{mean free path of particles } \lambda_{mfp}}{\text{characteristic system length } L} \ll 1 \quad (134)$$

5.1.2 Example: the plasma in the intercluster medium can be considered a fluid*

5.1.2.1 Mean free path in a model of colliding spheres

Consider a particle has a collisional cross-section of σ (in m^2) and moves with root mean square velocity \bar{v} in a medium with number density n (in m^{-3}). Note that the mean relative velocity between particles is larger than the mean velocity per particle

$$\langle v_{rel}^2 \rangle = \langle (v - v')^2 \rangle = \langle v^2 \rangle - \underbrace{2\langle vv' \rangle}_{= 0} + \langle v'^2 \rangle = 2\langle v'^2 \rangle = 2\bar{v}^2 \quad \rightarrow \quad v_{rel} = \sqrt{2}\bar{v} \quad (135)$$

as
uncor-
related

A particle thus in a time τ probes the volume $V = \sigma v_{rel} \tau$, so interacts with $N = nV$ particles. We are interested in the time till one interaction so the case $N = 1$, so $\tau = \frac{1}{n v_{rel} \sigma}$.

The path the particle itself has moved during that time is

$$\lambda_{mfp} = \tau \bar{v} = \frac{1}{\sqrt{2n}\sigma} \quad (136)$$

Note: The electrons though are much more mobile than the protons ($\frac{m_e}{m_p} \approx \frac{1}{1836}$, $m_e \approx 10^{-30}$ kg) so for electron-ion collisions the simpler $\lambda_{mfp} = \frac{1}{n\sigma}$ (assuming stationary nuclei) is probably better.

5.1.2.2 Collisional cross-section of an electron in a plasma and first approximation of λ_{mfp}

We approximate $\sigma = \pi r_{eff,e}^2$. We can approximate this radius by the distance where the electrostatic potential (in the electron-ion or electron-electron interaction) is only as relevant as the thermal energy, so (Z for the nucleic charge, relevant for the ion-electron interaction)

$$\frac{Ze^2}{r_{eff,e}} \sim m_e w_e^2 \sim k_B T_e \quad (137)$$

We can therefore approximate an electron mean free path

$$\begin{aligned} \lambda_{mfp,e} &\simeq \frac{1}{n\sigma} \sim \frac{1}{n\pi r_{eff,e}^2} \sim \frac{1}{\pi n} \left(\frac{k_B T_e}{Ze^2} \right) \\ &\sim 1.5 \times 10^{22} \left(\frac{n}{10^{-3} \text{ cm}^{-3}} \right)^{-1} \left(\frac{k_B T_e}{1 \text{ keV}} \right) \text{ cm} \end{aligned} \quad (138)$$

where we assumed $Z = 1$.

Note: Most *collisions* in (strongly ionized) plasmas are Coulomb interactions with very small deflections (so collision parameters mostly larger than $r_{eff,e}$). Therefore, we define the collision rate and collisional cross-section based on a total deflection of $\frac{\pi}{2}$.

5.1.2.3 A better approximation to the mean free path in an ionized plasma

More careful considerations take into account the smaller deflections. The integral over $\frac{1}{b}$ (impact parameter b) yields the Coulomb logarithm

$$\log \Lambda = \log \frac{b_{max}}{b_{min}} \quad (139)$$

where for b_{max} we take the Debye length

$$\lambda_{D\xi} = \left(\frac{\epsilon_0 k_B T_\xi}{n_\xi e^2} \right)^{\frac{1}{2}}, \quad \xi = \text{ion, electron, in plasma } n_i \approx n_e \quad (140)$$

(where the potential of a *Überschussladung* drops to $\frac{1}{e}\phi_0$ (e here Euler's number)) and for b_{min} e.g. $\pi r_{eff,e}$. One finally gets

$$\lambda_{mfp,e} = \frac{\bar{v}_e}{\nu_{ei}} \approx 64\pi\lambda_D \frac{\Lambda}{\log \Lambda} \propto \frac{T_e^2}{n_e} \gg \lambda_D \quad (141)$$

with ν_{ei} being the electron-ion collision rate. Examples of mean free paths are given in table 5.

Plasma	Solar Wind ($n \sim 1 \text{ cm}^{-3}$, $T = \text{some eV}$)	Ionosphere	Gas molecule in air under standard conditions
mean free path λ_{mfp}	1 AU	1 km	68 nm

Table 5: Mean free paths

The plasma in the intercluster medium can be considered a fluid. The solar wind can hardly be considered a fluid on a reasonable scale.

5.1.3 Fluid description based on fluid parcels

The modes of motion in a fluid are

- motion of macroscopic fluid parcels
- random walk between fluid parcels (diffusion, very slow)

The most important fluid equation, the **Navier Stokes equation** falls out of the Boltzmann equation but can also be *understood* as an equation of movement for a fluid parcel.

- mass conservation \iff continuity equation
- mass conservation + momentum conservation \iff Navier Stokes equation

5.1.3.1 Eulerian and Lagrangian fluid dynamics

Lagrangian and Eulerian view are presented in table 6.

The rate of change in the Eulerian view is the **material derivative**, the rate of change of a physical quantity (like temperature) of a material element that is subjected to a macroscopic

Lagrangian description	Eulerian description
Coordinate system comoves with the fluid element (we sit on the fluid parcel)	Coordinate system is fixed in space
The temporal evolution of fluid quantities is described along the trajectory of the movement of a fluid parcel	The temporal evolution of fluid quantities is described at a fixed locations over accounting volumes
e.g. measurement on a weather balloon following the wind flow	e.g. measurement by ground stations

Table 6: Lagrangian and Eulerian fluid dynamics

velocity field \underline{v} . Let $\Psi_L(t)$ describe the evolution of a fluid quantity along the parcel motion (Lagrangian) and $\Psi_E(\underline{x}, t)$ the evolution of a fluid quantity at a fixed location (Eulerian). Then the material derivative is

$$D_t \Psi_L = \frac{D\Psi_L}{Dt} = \partial_t \Psi_L = \underbrace{\partial_t \Psi_E}_{\substack{\text{local} \\ \text{change}}} + \underbrace{\underline{v} \cdot \nabla_{\underline{x}} \Psi_E}_{\substack{\text{convective} \\ \text{change}}} = D_t \Psi_E \quad (142)$$

(derive $\Psi(\underline{x}(t), t)$ with respect to time and use the chain rule). Generally

$$D_t = \partial_t + \underline{v} \cdot \nabla \quad (143)$$

5.1.3.2 Continuity equation

As of mass conservation, the change of mass m_V in a volume V can only be due to flux $\underline{j} = \rho \underline{v}$ through the surface ∂V .

$$\partial_t m_V = \partial_t \int_V \rho dV = - \int_{\partial V} \underline{j} \cdot d\underline{A} \underset{\text{Gauss}}{=} - \int_V \nabla \cdot \underline{j} dV \rightarrow \boxed{\partial_t \rho + \nabla \cdot (\rho \underline{v}) = 0} \quad (144)$$

which is the continuity equation.

5.1.3.3 Incompressible fluids

An incompressible fluid is one, where the density of a fluid parcel never changes with time $\frac{d\rho}{dt} = 0$ (there can still be gradients in the density). From the Lagrangian form of the continuity equation

$$0 = \frac{d\rho}{dt} = \partial_t \rho + \underline{v} \cdot \nabla \rho \underset{\substack{\text{cont.} \\ \text{eq.}}}{=} \boxed{\nabla \cdot \underline{v} = 0} \quad (145)$$

For $\nabla \cdot \underline{v} < 0$ we have a sink, for $\nabla \cdot \underline{v} > 0$ a source. The divergence of the velocity field of incompressible fluids vanishes (divergence free flow) (e. g. rotational flow) Note in a 3D flow horizontal convergence can be balanced by vertical divergence.

Interpretation of $\nabla \cdot \underline{v} = 0$ A divergence free flow does not mean that the velocity does not change over space as one can easily see from flow along a narrowing tube. Consider a tube with flow tangential to it.

$$\begin{aligned} \nabla \cdot \underline{v} = 0 &\xrightarrow{\text{integrate over tube}} 0 = \int_V \nabla \cdot \underline{v} dV \stackrel{\text{Gauss}}{=} \int_{\partial V} \underline{v} d\underline{A} \\ &\xrightarrow{\text{only opposing flux through } A_1 \text{ and } A_2} v_1 A_1 - v_2 A_2 = 0 \\ &\rightarrow v_1 A_1 = v_2 A_2 \end{aligned} \quad (146)$$

5.1.3.4 Equation of motion of a fluid parcel, general path towards Navier-Stokes

Consider a fluid parcel with constant mass m . The equation of motion is

$$\underline{F} = \frac{d\underline{p}}{dt} = m \frac{d\underline{V}}{dt} = \rho \underline{V} \frac{d\underline{v}}{dt}, \quad \underline{a} = \frac{\underline{F}}{\rho \underline{V}} = \frac{\underline{f}}{\rho} = \frac{d\underline{v}}{dt} \quad (147)$$

Using the material derivative, we can write

$$\frac{d\underline{v}}{dt} = \partial_t \underline{v} + \underbrace{\underline{v} \cdot \nabla \underline{v}}_{\substack{\text{non-linear} \\ \text{term}}} = \underline{a} = \frac{\underline{f}}{\rho} \quad (148)$$

advection
→ chaotic behavior

where one can find (leading to a simplified **Navier Stokes equation**)

$$\underline{f} = \underbrace{\underline{f}_g}_{\substack{\text{gravi.} \\ \text{force}}} + \underbrace{\underline{f}_p}_{\substack{\text{pressure} \\ \text{force}}} + \underbrace{\underline{f}_f}_{\substack{\text{friction} \\ \text{force}}} = \rho \underline{g} - \nabla p + \rho \nu \nabla^2 \underline{v} \quad (149)$$

where the viscous friction \underline{f}_v (viscosity ν) is an approximation for an incompressible isotropic Newtonian fluid. While pressure is a force per area, only when there are different pressures acting on the sides of a fluid parcel (a gradient), there is net movement. The friction term can be understood as diffusion of momentum when there is a velocity gradient (which only leads to a local change when $\nabla^2 \underline{v} \neq 0$, otherwise there is the same momentum diffusion in and out of the parcel).

5.2 Basic Gas Dynamics

Aim: Our aim is to derive equations for the evolution of variables of the fluid, like density, velocity and temperature. For instance how do perturbation (e.g. by a jet) propagate through the fluid?

5.2.1 Distribution function and Boltzmann equation

Idea: Let us derive the fluid equations based on the Boltzmann equation for the distribution function.

The distribution function $f(\underline{x}, \underline{u}, t)$ is defined so that

$$f(\underline{x}, \underline{u}, t) d^3x d^3u \quad (150)$$

is (depending on the normalization) the probability of finding a particle in the phase space volume $d^3x d^3u$ around $(\underline{x}, \underline{u})$ at time t or the expected number of particles therein, so in total

$$N = \int \int f(\underline{x}, \underline{u}, t) d^3x d^3u, \quad \text{total number of particles } N \quad (151)$$

Phase space conservation (Liouville theorem) (particles are neither created nor destroyed) is captured in the Boltzmann equation

$$\frac{d}{dt} f(\underline{x}, \underline{u}, t) = \partial_t f + \dot{\underline{x}} \nabla_{\underline{x}} f + \dot{\underline{u}} \nabla_{\underline{u}} f = \left. \frac{df}{dt} \right|_c, \quad \text{with } \dot{\underline{x}} = \underline{u}, \dot{\underline{u}} = g \quad (152)$$

where $\left. \frac{df}{dt} \right|_c$ is the change in f due to *collisions*.

Note: One can understand this in the sense of discontinuous motion (instantaneous velocity changes) kicking a particle out of a phase space volume. More aligned with our previous derivation, we can see this as a simplification of the correlation term between forces and accelerations (also capturing the particle interaction) (see subsection 4.2).

In the case of a sufficiently large collision term $\left. \frac{df}{dt} \right|_c$ (in the fluid limit $\lambda_m f p \ll L$) $f(\underline{u})$ becomes Maxwellian. Near the sun for instance Coulomb collisions are incapable of isotropizing the velocity distribution of the ions in the solar wind (cigar shape).

5.2.2 Retrieving information from the Boltzmann equation

By taking the moments of the distribution function over velocity space

$$M_n(\underline{x}, t) = \int \underline{v}^n f(\underline{x}, \underline{u}, t) d^3 u \quad (153)$$

we find out on the quantities of interest in space as

- density (0th moment)

$$\begin{aligned} n &= \int f(\underline{x}, \underline{u}, t) d^3 u, \quad \rho = nm, \quad \text{particle mass } m \\ \rightarrow \text{mass weighted average } \langle q \rangle(\underline{x}, t) &= \frac{1}{\rho(\underline{x})} \int q(\underline{x}, \underline{u}, t) f(\underline{x}, \underline{u}, t) d^3 u \end{aligned} \quad (154)$$

- mean velocity (1st moment)
- pressure tensor as of the fluctuations of the velocity around the mean (\rightarrow temperature in case of a Maxwell distribution) (2nd moment)
- heat tensor (3rd moment)

The development of those quantities in time is given by the balance (/conservation) equations obtained from taking the appropriate moments of the Boltzmann equation (mass, momentum and energy are conserved).

form of balance equations: ∂_t conserved quantity + ∇ corrsp. flux = source term (155)

5.2.3 Mass conservation | continuity equation (1st moment)

By following the steps

- multiply the Boltzmann equation by m and integrate over $d^3 u$
- using Gauss divergence theorem and assuming $f \rightarrow 0$ for $u \rightarrow \infty$
- local mass conservation $\int m \frac{df}{dt} \Big|_c d^3 u = 0$ (collisions only shift discontinuously on velocity space)

one follows the mass continuity equation

$$\partial_t \rho + \nabla \cdot (\rho \cdot \underline{v}) = 0, \quad \underline{u} = \underline{v} + \underline{w}, \quad \underline{v} = \langle \underline{u} \rangle \quad (156)$$

By taking the volume integral ($d^3 x$) we find $\frac{dm}{dt} = 0$, i.e. mass conservation.

5.2.3.1 Derivation of the continuity equation*

From

$$\underbrace{\int m \partial_t f d^3 u}_{\partial_t \rho(\underline{x}, t)} + \underbrace{\nabla_{\underline{x}} \int m \underline{u} f d^3 u}_{\text{using } r, u \text{ indep.}; \underline{u} = \nabla_{\underline{x}}(\rho \underline{v})} + \underbrace{\int m \nabla_{\underline{u}}(\underline{g} f) d^3 u}_{\text{assume acc. } \underline{g} \text{ indep. } \underline{u}} = \underbrace{\int m \frac{df}{dt} \Big|_c d^3 u}_{= 0 \text{ (local particle conserv.)}} \quad (157)$$

where the third term also vanishes (apply Gauss, assume $f \rightarrow 0$ (e.g. Maxwellian $\exp(-u^2)$) for $u \rightarrow \infty$), we get the result from above.

5.2.4 Momentum conservation | Navier Stokes equation (2nd moment)

By following the steps

- multiply Boltzmann equation with momentum ($m\underline{u}$) and integrate over $d^3 u$
- using that collisions conserve momentum
- using the continuity equation

one obtains the Navier-Stokes equation

$$\partial_t(\rho \underline{v}) + \nabla \cdot (\rho \underline{v} \underline{v}^T + P \underline{1} - \underline{\underline{\Pi}}) = \rho \underline{g} \quad (158)$$

which using the continuity equation becomes

$$\partial_t \underline{v} + (\underline{v} \cdot \nabla) \underline{v} = \underline{g} - \frac{1}{\rho} \nabla P + \frac{1}{\rho} \nabla \cdot \underline{\underline{\Pi}} \quad (159)$$

with

$$(\text{isotropic}) \text{ pressure } P = \frac{1}{3} \rho \langle \|\underline{w}\|^2 \rangle$$

anisotropic viscous stress tensor $\underline{\underline{\Pi}}_{ij} = P \delta_{ij} - p \langle w_i w_j \rangle$ (symmetric and traceless)

acceleration \underline{g} from external sources

(160)

Viscosity opposes shearing motion and interpenetration (diffusion of momentum on shear).

Taking the volume integral over the Navier-Stokes equation and applying Gauss theorem, we see that in the absence of an external force ($\underline{g} = 0$), the momentum is conserved.

5.2.4.1 Notes on the derivation of the Navier-Stokes equation

Take a look at the terms in

$$\partial_t \int m\underline{u} f d^3 u + \underline{\nabla}_x \int m\underline{u} \underline{u}^T f d^3 u + \int m\underline{g} \underline{u}^T \underline{\nabla}_x f du^3 = \int m\underline{u} \frac{\partial f}{\partial t} \Big|_c d^3 u \quad (161)$$

- first term: $\partial_t \int m\underline{u} f d^3 u = \partial_t(\rho\underline{v})$ by definition of \underline{v}
- second term: $\underline{\nabla}_x \int m\underline{u} \underline{u}^T f d^3 u = \underline{\nabla}_x (\rho \langle \underline{u} \underline{u}^T \rangle)$ using $\underline{u} = \underline{v} + \underline{w}$ this becomes $\underline{\nabla}_x (\rho \underline{v} \underline{v}^T) + 2\underline{\nabla}_x (\rho \underline{v} \langle \underline{w} \rangle^T) + \underline{\nabla}_x (\rho \langle \underline{w} \underline{w}^T \rangle) = \underline{\nabla}_x (\rho \underline{v} \underline{v}^T) + \underline{\nabla}_x (\rho \langle \underline{w} \underline{w}^T \rangle)$ as $\langle \underline{w} \rangle = 0$.
- third term: assuming \underline{g} does not depend on \underline{u} , $\underline{g} \int m\underline{u}^T \underline{\nabla}_x f du^3 = -\rho \underline{g}$ as by the chain rule $\underline{u}^T \underline{\nabla}_x f = \underline{\nabla}_x (\underline{u}^T f) - f \underline{\nabla}_x \underline{u}^T$ where the integral over the first term vanishes by the same reasoning as before
- the right-hand side vanishes, because collisions conserve momentum

Finally, $(\rho \langle \underline{w} \underline{w}^T \rangle)$ (so the correlation matrix of the random fluctuations (?)) is split into an isotropic contribution from pressure P and an anisotropic viscous stress tensor $\underline{\underline{\Pi}}$. The continuity equation can be used to get (first apply chain rule) $\partial_t(\rho\underline{v}) + \underline{\nabla}_x (\rho \underline{v} \underline{v}^T) = \rho (\partial_t \underline{v} + (\underline{v} \cdot \underline{\nabla}) \underline{v})$

5.2.4.2 Interpretation and viscous stress tensor for a Newtonian fluid

Our result

$$\frac{d\underline{v}}{dt} = \partial_t \underline{v} + (\underline{v} \cdot \underline{\nabla}) \underline{v} = \underline{g} - \frac{1}{\rho} \underline{\nabla} P + \frac{1}{\rho} \underline{\nabla} \cdot \underline{\underline{\Pi}} \quad (162)$$

already looks similar to our intuitive result. We have

- some acceleration from external forces \underline{g} (e.g. gravity)
- pressure force, where on the pressure we now also have a microscopic understanding $P = \frac{1}{3}\rho \langle \|\underline{w}\|^2 \rangle$
- some viscous force as described by $\frac{1}{\rho} \underline{\nabla} \cdot \underline{\underline{\Pi}}$

Intuition for viscosity and momentum diffusion: Viscosity is an every-day phenomenon - but how can we understand it microscopically? Consider a fluid with bulk velocity \underline{v} only in x -direction. Now imagine v_x increases with z (maybe because I'm pulling a plate on top of my container). The molecules constantly bump into each other, and it is much more probable that a particle from higher up z will give x -momentum to one from lower down in a collision than vice versa. Therefore x -momentum diffuses down. However if the x -velocity increases linearly with z , $v_x(z)$ will not change, as there is as much momentum transport in as out of a given slice along x at some height z . Therefore the simplest diffusion equation for momentum is $\rho \partial_t v_x(z) = \nu \rho \partial_z^2 v_x$ (for a positive curvature more momentum diffuses down from the top then goes out of the bottom). Speaking in terms of the fluctuations w from the bulk velocity \underline{v} , in the above scenario we expect particles with high *random* downward velocity $-w_z$ to be faster than their surrounding (high w_x), so we expect $-\langle w_z w_x \rangle > 0$.

With the above intuition, it should make sense that

$$\Pi_{ij} = P\delta_{ij} - p\langle w_i w_j \rangle \quad (163)$$

and that we might make the ansatz of Π_{ij} being linear in $\frac{\partial v_i}{\partial x_j}$

$$\begin{aligned} \Pi_{ij} &= \eta D_{ij} + \xi \delta_{ij} (\nabla \cdot \underline{v}) \\ D_{ij} &= \frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3} \delta_{ij} (\nabla \cdot \underline{v}) \end{aligned} \quad (164)$$

with D_{ij} being the deformation tensor that vanishes for uniform expansion or contraction, η and ξ are the coefficients of shear and bulk viscosity respectively, with units $\text{g cm}^{-1} \text{s}^{-1}$.

- ηD_{ij} represents the resistance to shearing motion
- $\xi \delta_{ij} (\nabla \cdot \underline{v})$ represents the resistance in volume, remember that for $\nabla \cdot \underline{v} = 0$ we have an incompressible fluid

5.2.5 Energy conservation (3rd moment)

By the following steps

- multiply the Boltzmann equation by $m\underline{u}^2$ and integrate over $d^3 u$
- use that collisions conserve energy
- use the continuity equation

one follows

$$\begin{aligned}
\rho \frac{de_{th}}{dt} &\equiv \partial_t(\rho e_{th}) + \underline{\nabla} \cdot (\rho e_{th} \cdot \underline{v}) = -P \underline{\nabla} \cdot \underline{v} + \Psi - \underline{\nabla} \cdot \underline{Q} \\
\text{specific internal energy } e_{th} &\equiv \frac{1}{2} \langle ||\underline{w}||^2 \rangle \\
\text{viscous dissipation rate (bulk motion } &\rightarrow \text{ internal energy) } \Psi \equiv \sum_{i,j=1}^N \Pi_{ij} \frac{\partial v_i}{\partial x_j} \\
\text{conducting heat flux } \underline{Q} &= \frac{1}{2} \rho \langle \underline{w} || \underline{w} ||^2 \rangle
\end{aligned} \tag{165}$$

5.2.5.1 Notes on the conductive heat flux

There is only conductive heat flux \underline{Q} if \underline{w} is asymmetric around the bulk motion (hotter (*faster* in the sense of $||\underline{w}||$) particles drift relative to cold ones). The conductive flux is in most cases produced by a temperature gradient

$$\begin{aligned}
\underline{Q} &= -\xi \underline{\nabla} T, \quad \text{with } \chi \simeq 6 \times 10^{-7} T^{\frac{5}{2}} \text{ erg s}^{-1} \text{ cm}^{-1} \text{ K}^{-1} \\
\text{proper diffusion coefficient } \kappa &= \frac{\chi T}{P} \text{ in cm}^2 \text{ s}^{-1}
\end{aligned} \tag{166}$$

(by collisions and random movement, wiggling spreads, stochastically along the gradient of T). As always, there is only change in a volume if there is different in- and outflux, so if $\underline{\nabla} \cdot \underline{Q} \neq 0$.

5.2.5.2 Evolution equation for the total specific energy $e = e_{th} + \underline{v}^2/2$

We are interested in

$$\frac{de}{dt} = \frac{de_{th}}{dt} + \frac{d}{dt} \left(\frac{1}{2} \underline{v}^2 \right) \tag{167}$$

where we already know $\frac{de_{th}}{dt}$. So let us consider

$$\begin{aligned}
\partial_t \left(\frac{\rho \underline{v}^2}{2} \right) &= \frac{\underline{v}^2}{2} \partial_t \rho + \rho \underline{v} \partial_t \underline{v} \\
&= \frac{\underline{v}^2}{2} (-\underline{\nabla}(\rho \underline{v})) + \rho \underline{v} \left(-(\underline{v} \cdot \underline{\nabla}) \underline{v} + \underline{g} - \frac{1}{\rho} \underline{\nabla} P + \frac{1}{\rho} \underline{\nabla} \cdot \underline{\Pi} \right)
\end{aligned} \tag{168}$$

where we used the continuity equation and [Navier-Stokes equation](#). Based on

$$(\underline{v} \cdot \underline{\nabla}) \underline{v} \equiv \frac{1}{2} \underline{\nabla} \underline{v}^2, \quad \rho \underline{v} \frac{1}{2} \underline{\nabla} \underline{v}^2 + \frac{\underline{v}^2}{2} \underline{\nabla} \rho \underline{v} = \underline{\nabla} \cdot \left(\frac{1}{2} \rho \underline{v}^2 \underline{v} \right) \tag{169}$$

this becomes

$$\frac{d}{dt} \left(\frac{\rho v^2}{2} \right) = \partial_t \left(\frac{\rho v^2}{2} \right) + \underline{v} \cdot \left(\frac{1}{2} \rho \underline{v}^2 \underline{v} \right) = \rho \underline{v} \cdot \underline{g} - \underline{v} \cdot \nabla P + \underline{v} \cdot (\nabla \cdot \underline{\Pi}) \quad (170)$$

so we finally get the evolution equation for the total specific energy $e = e_{th} + \underline{v}^2/2$

$$\partial_t(\rho e) + \nabla \cdot [(\rho e + P)\underline{v} - \underline{\Pi} \cdot \underline{v} + \underline{Q}] = \rho \underline{v} \cdot \underline{g} \quad (171)$$

where taking the volume integral and applying Gauss theorem shows energy conservation if $\underline{g} = 0$.

5.2.6 Entropy conservation

Note: Heat conduction and viscous friction change the entropy and entropy is conserved if those processes are absent.

From the first law of thermodynamics in its specific form (specific volume $\tilde{V} = 1/\rho$)

$$de_{th} = dw + dq = -Pd\tilde{V} + Tds = \frac{P}{\rho^2} d\rho + Tds \quad (172)$$

we find (*divide by dt* and insert the continuity and energy equation)

$$\rho T \frac{ds}{dt} = -\nabla \cdot \underline{Q} + \Psi \quad (173)$$

proving the statement in the note.

5.3 Euler Equation and Navier-Stokes equation

Ideal gas dynamics, where we assume internal friction and heat conduction to be absent, are described by the Euler equations. Those assumptions are well justified for gas flow in astrophysics, which are often of extremely low density.

If viscosity is relevant, we use the hydrodynamical equations including viscosity, the Navier-Stokes equation.

We will later discuss fluid instabilities in the absence of viscosity.

5.3.1 Euler Equations

Assuming (valid for non-dense media like air)

- no thermal conductivity
- no internal friction
- no external forces

the balance equations obtained from the moments of the Boltzmann equation simplify to the Euler equations (table 7).

Continuity equation	Balance of momentum	Balance of energy
$\partial_t \rho + \nabla \cdot (\rho \underline{v}) = 0 \quad (174)$	$\partial_t (\rho \underline{v}) + \nabla \cdot (\rho \underline{v} \underline{v}^T + P \underline{\underline{1}}) = 0 \quad (175)$	$\begin{aligned} \partial_t (\rho e) + \nabla \cdot [(\rho e + P) \underline{v}] \\ = 0 \text{ with } e = e_{th} + \frac{1}{2} \underline{v}^2 \\ \frac{\text{total energy}}{\text{unit mass}} \end{aligned} \quad (176)$

Table 7: Euler equations

The equations form a set of hyperbolic conservation laws (all continuity equations, one for mass, one for momentum, one for energy).

Hierarchy and closure The equations are in a hierarchy that in one equations occur quantities following from the next higher one (velocity in the continuity equation, ...) - we need a further closure relation making the energy balance unnecessary, for an ideal gas this is

$$P = (\gamma - 1) \rho e_{th}, \quad \gamma = \text{specific heat ratio} = \frac{c_p}{c_v}, \quad \text{for monoatomic gas } \gamma = \frac{f+2}{f} = \frac{5}{3} \quad (177)$$

Intuition: Let us consider the x -component of the momentum equation to get a better understanding (the others are analogous). We have

$$\partial_t(\rho v_x) + \underbrace{\partial_x P}_{\text{pressure}} + \underbrace{\partial_x(v_x \cdot (\rho v_x)) + \partial_y(v_y \cdot (\rho v_x)) + \partial_z(v_z \cdot (\rho v_x))}_{\text{advection of } x\text{-momentum from all directions}} = 0 \quad (178)$$

gra-
 di-
 ent
 force

so x -momentum (density) at a given position (Eulerian) changes due to pressure gradients, and because of the advection of x -momentum from all directions to the given position.
Density, pressure and energy advect with the flow in the Eulerian view.

5.3.2 Navier-Stokes equation

In real fluids, viscosity transforms relative motion into heat. For

- vanishing conductivity
- vanishing external forces

we have (table 8)

Continuity equation	Balance of momentum	Balance of energy
$\partial_t \rho + \nabla \cdot (\rho \underline{v}) = 0 \quad (179)$	$\begin{aligned} \partial_t(\rho \underline{v}) + \\ \nabla \cdot (\rho \underline{v} \underline{v}^T + \underline{\underline{\Pi}}) &= \nabla \cdot \underline{\underline{\Pi}} \end{aligned} \quad (180)$	$\begin{aligned} \partial_t(\rho e) \\ + \nabla \cdot [(\rho e + P) \underline{v}] \\ = \nabla \cdot (\underline{\underline{\Pi}} \cdot \underline{v}) \end{aligned} \quad (181)$

Table 8: Navier stokes equations

$\underline{\underline{\Pi}}$ is the viscous stress tensor, a material property. For a vanishing stress tensor, we recover the Euler equations. To first order $\underline{\underline{\Pi}}$ must be linear in the velocity derivatives, where the most general rank-2 tensor of this type can be written as

$$\underline{\underline{\Pi}} = \eta \left[\nabla \underline{v}^T + (\nabla \underline{v}^T)^T - \frac{2}{3} (\nabla \cdot \underline{v}) \underline{\underline{1}} \right] + \xi (\nabla \cdot \underline{v}) \underline{\underline{1}}$$

η scales the traceless part, describes shear viscosity (182)
 ξ scales the trace, describes bulk viscosity
 possibly η, ξ depend on ρ, T, \dots

5.3.2.1 Simplification of the Navier-Stokes equations for incompressible fluids ($\underline{\nabla} \cdot \underline{v} = 0$)

In the case $\underline{\nabla} \cdot \underline{v} = 0$ only shear forces matter (no bulk compression), and we have

$$\frac{1}{\eta}(\underline{\nabla} \cdot \underline{\Pi})_x = (\underline{\nabla} \cdot [\underline{\nabla} \underline{v}^T + (\underline{\nabla} \underline{v}^T)^T])_x = \underline{\nabla}^2 v_x \quad (183)$$

(hint: write the component out and use $\partial_x^2 v_x + \partial_y \partial_x v_y + \partial_z \partial_x v_z = \partial_x (\partial_x v_x + \partial_y v_y + \partial_z v_z) = 0$).

Introducing the kinematic viscosity $\nu \equiv \frac{\eta}{\rho}$, we get

$$\frac{d\underline{v}}{dt} = \partial_t \underline{v} + (\underline{v} \cdot \underline{\nabla}) \underline{v} = -\frac{\underline{\nabla} P}{\rho} + \nu \underline{\nabla}^2 \underline{v} \quad (184)$$

so the simplified expression from the beginning. The bulk motion responds to pressure gradients and viscous forces.

5.3.2.2 Characterizing flow | Reynolds number

Consider a flow problem with characteristic length scale L_0 , velocity V_0 and density scale ρ_0 . Let us define dimensionless fluid variables and operators

$$\begin{aligned} \hat{\underline{v}} &= \frac{\underline{v}}{V_0}, & \hat{x} &= \frac{x}{L_0}, & \hat{P} &= \frac{P}{\rho_0 V_0^2} \\ T_0 &= \frac{L_0}{V_0}, & \hat{t} &= \frac{t}{T_0}, & \hat{\rho} &= \frac{\rho}{\rho_0}, & \hat{\underline{\nabla}} &= L_0 \underline{\nabla} \end{aligned} \quad (185)$$

where for the pressure mind that pressure is also an energy density and the kinetic energy density is $\frac{1}{2}\rho_0 V_0^2$.

Plugging this the incompressible Navier-Stokes equation (184) we get

$$\frac{d\hat{\underline{v}}}{d\hat{t}} = -\frac{\hat{\underline{\nabla}} \hat{P}}{\hat{\rho}} + \frac{\nu}{L_0 V_0} \hat{\underline{\nabla}}^2 \hat{\underline{v}} \quad (186)$$

involving the Reynolds number

$$Re \equiv \frac{L_0 V_0}{\nu} \quad (187)$$

We can understand the Reynolds number as the ratio between the chaotic advective term in the Navier-Stokes equation and the frictional term, so

$$Re = \frac{\text{advective term}}{\text{frictional term}} = \frac{|(\underline{v} \cdot \underline{\nabla}) \underline{v}|}{\nu \underline{\nabla}^2 \underline{v}} \quad (188)$$

Note: Connection between the Reynolds number and turbulence: The quadratic (in the velocity) term $(\underline{v} \cdot \nabla)\underline{v}$ generates turbulence (deterministic chaos) while the viscous term $\nu \nabla^2 \underline{v}$ destroys it via dissipation. In terms of energy

$$Re \sim \frac{V_0 L_0}{\nu} = \frac{\rho L^3 U^2}{\mu L^2 U} = \frac{2E_{kin}}{W_{friction}}, \quad \text{as } W_{friction} \sim F_{friction} \cdot L_0 = V \rho \nu \nabla^2 \underline{v} \cdot L_0 = \mu V_0 L_0^2 \quad (189)$$

(with the kinematic viscosity $\nu = \frac{\mu}{\rho}$ in $m^2 s^{-1}$). In general, the higher the Reynolds number the more turbulence, for

$$Re > R_c \sim 10^3 \quad (190)$$

we have turbulent flow (figure 40). For $Re \sim 1$ viscosity dominates, for $Re \rightarrow \infty$ we approach an ideal gas.

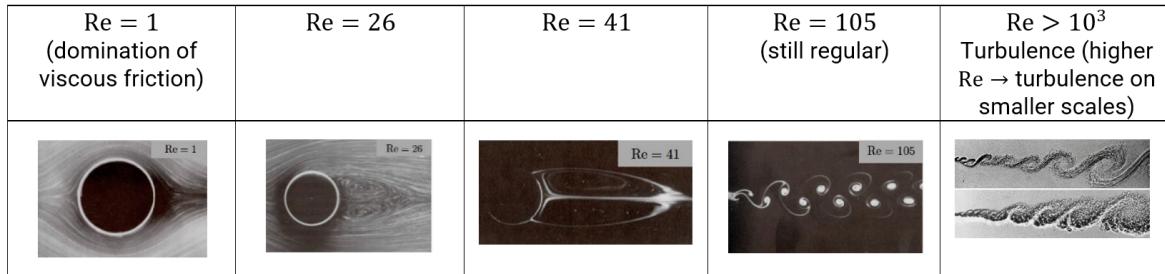


Figure 40: Reynolds number

5.4 Shocks

In hydrodynamical flows, shock waves can develop, where the fluid variables

- density ρ
- velocity \underline{v}
- temperature T
- specific entropy s

jump by finite amounts. In the frame of the Euler equations these are true mathematical discontinuities, while exhibiting a finite width in the Navier Stokes equations. A shock wave

- propagates faster than the signal speed for compressible waves c_s
- produces and irreversible change to the fluid change (increase in entropy)

A shock wave is a region of small thickness over which the properties of the flow change rapidly.

5.4.1 Propagation of disturbances 1: Speed of sound

Microscopic Intuition: Consider a higher density region in a fluid. Probabalistically there is more momentum in the direction of lower density, so (by collisions) the higher density will spread out. The characteristic speed of on which density information propagates is related to the jiggling of the particles, i.e. their themperature. This characteristic speed of sound is roughly derived in the following.

Soundwaves are messengers, carrying density and pressure fluctuations. Imagine you're driving in fog towards a traffic jam - if you're so quick no messenger can quickly enough reach you, you will have a shock.

Consider the Euler equation for momentum without internal forces or friction in 1D in a steady state with constant flux $j = \rho v$, so $0 = d(\rho v) \rightarrow \rho dv = -v d\rho$. We get

$$\frac{dv}{dt} = -\frac{1}{\rho} \frac{dP}{dx} \quad \rightarrow \quad dP = -(\rho dv) \frac{dx}{dt} = (vd\rho)v \quad \rightarrow \quad v^2 \equiv c_s^2 = \frac{dP}{d\rho} \quad (191)$$

(note is relative to the bulk motion, so in the local rest frame of the flow, more later).

Now we use that for an adiabatic process, $P\rho^{-\gamma} = \text{const.}$ (and $T\rho^{1-\gamma} = \text{const.}$) so taking the logarithm and differentiating with respect to ρ yields

$$\frac{dP}{d\rho} = \frac{\gamma p}{\rho} \quad (192)$$

so using the ideal gas law

$$P = nk_B T = \frac{\rho}{m_p \mu} k_B T \quad \text{with } \mu \text{ being a mean molecular weight} \quad (193)$$

we get

$$c_s^2 = \frac{\gamma k_B T}{m_p \mu} \quad (194)$$

so based on $T\rho^{1-\gamma} = \text{const.}$ we can write

$$c_s^2 \propto \rho^{\gamma-1} \quad (195)$$

5.4.2 Characteristics of Perturbations

Idea: Our aim is to find the characteristics, lines in the (x, t) plane along which perturbations propagate.

Let us start with the continuity equation in 1D

$$\text{cont.: } \frac{1}{\rho} \left(\frac{\partial \rho}{\partial t} + v \frac{\partial \rho}{\partial x} \right) = 0 \quad (196)$$

With $c_s^2 = \frac{\partial P}{\partial \rho}$ we can write the Euler equation for momentum as

$$\text{Euler: } \frac{\partial v}{\partial t} + v \frac{\partial v}{\partial x} = -\frac{1}{\rho} \frac{\partial P}{\partial x} = -\frac{c_s^2}{\rho} \frac{\partial \rho}{\partial x} \quad (197)$$

Based on $c_s^2 \propto \rho^{\gamma-1}$, we can replace $\partial \rho$ in those equations using

$$\frac{d\rho}{\rho} = \frac{2}{\gamma-1} \frac{dc_s}{c_s} \quad (198)$$

With this replace $\partial \rho$ in the continuity and Euler equation. From adding and subtracting the Euler and continuity equation, we get

$$\begin{aligned} [\partial_t + (v + c_s) \partial_x] \left(u + \frac{2}{\gamma-1} c_s \right) &= 0 \\ [\partial_t + (v - c_s) \partial_x] \left(u - \frac{2}{\gamma-1} c_s \right) &= 0 \end{aligned} \quad (199)$$

Defining

$$\begin{aligned} \xi_+ \equiv u + \frac{2}{\gamma-1} c_s &\rightarrow \frac{d}{dt} \xi_+(x(t), t) = [\partial_t + (\partial_t x) \partial_x] \xi_+(x(t), t) = 0 \text{ for } \partial_t x = v + c_s \\ \xi_- \equiv u - \frac{2}{\gamma-1} c_s &\rightarrow \frac{d}{dt} \xi_-(x(t), t) = [\partial_t + (\partial_t x) \partial_x] \xi_-(x(t), t) = 0 \text{ for } \partial_t x = v - c_s \end{aligned} \quad (200)$$

where we applied the *method of characteristics*.

From this we can see that (*as expected*) in a fluid with bulk motion v , perturbations propagate along characteristics with velocity $v \pm c_s$.

These characteristic equations are the same no matter the amplitude of the perturbations.

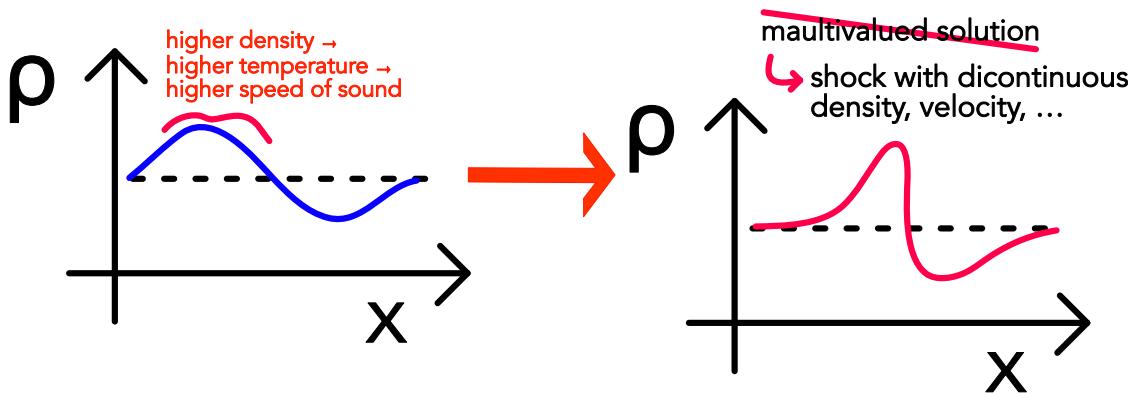


Figure 41: Formation of a shock

5.4.3 Formation of a shock

5.4.3.1 Formation as a pressure driven compressive disturbance

Consider a fluid with base density ρ_0 . For small perturbation in density, we can to first order use

$$c_s = \left(\frac{\partial P}{\partial \rho} \right)^{\frac{1}{2}} \simeq \frac{\gamma p_0}{\rho_0} \quad (201)$$

But now consider a larger perturbation. Adiabatically the temperature scales with $T \propto \rho^{\gamma-1}$, and as c_s^2 scales linearly with the temperature, we have

$$c_s^2 \propto T \propto \rho^{\gamma-1} \quad (202)$$

Therefore, the “waves crest overtakes the valley” (figure 41) but as the hydrodynamic equations don’t allow for multivalued solutions, we get a shock with discontinuities in ρ , v , T and s . For an isothermal gas $c_s = \text{const.}$ but steepening can happen nonetheless as of the non-linearity of the Euler / Navier-Stokes equations (?).

5.4.3.2 Causes for shocks

- supersonic compressible disturbance
- supersonic collision of two streams of fluids
- non-linear interaction of subsonic compressible modes (nonlinear wave interaction)

5.4.4 Collisional and collisionless shocks | shock front

The »discontinuous« change normally happens over a scale proportional to the *effective* mean free path λ_{eff} .

- Collisional shocks: Coulomb-collisions determine λ_{eff}

- Collisionless plasma like solar wind: λ_{eff} is reduced by electromagnetic viscosities
 $\lambda_{eff} \ll \lambda_{coulomb}$

The shock front or transition layer is of the scale of λ_{eff} . Here, viscous effects are important - they dissipate kinetic energy, generating heat and entropy. Outside this layer viscous effects are small on scales $L \gg \lambda_{eff} \rightarrow \underline{\underline{\Pi}} = \underline{\underline{0}}$.

Note: This scale violates the assumptions under which the Navier-Stokes equations are derived from the Boltzmann equation. »It would be more than 50 years before computer simulation and laboratory experiments would show that physical shocks are measured to be twice the width predicted by theory, validating Becker's assertion that something beyond the Navier-Stokes description is needed.« (Margolin and Lloyd-Ronning, 2023)

5.4.5 Properties at fluid discontinuities

External gravitational forces and conductive heat flux are way slower than the transition time of fluid discontinuities and can thus be neglected.

We consider the propagating fluid discontinuity in its rest frame (i.e. upstream is ahead of the shock).

We distinguish two types of fluid discontinuities

- shocks characterized by mass flux through their interface
- contact discontinuities without such mass flux

5.4.5.1 (Rankine-Hugoniot) Jump conditions I: Assumptions

Relative to the shock, fluid moves from upstream to downstream and we would like to relate upstream conditions ρ_1, v_1, T_1 to downstream conditions ρ_2, v_2, T_2 by *jump conditions*, see figure 42.

We assume

- the velocity to be perpendicular to the surface of the discontinuity (we later generalize)
- a steady state $\partial_t = 0$
- a 1D situation $\underline{\nabla} \rightarrow \partial_x$

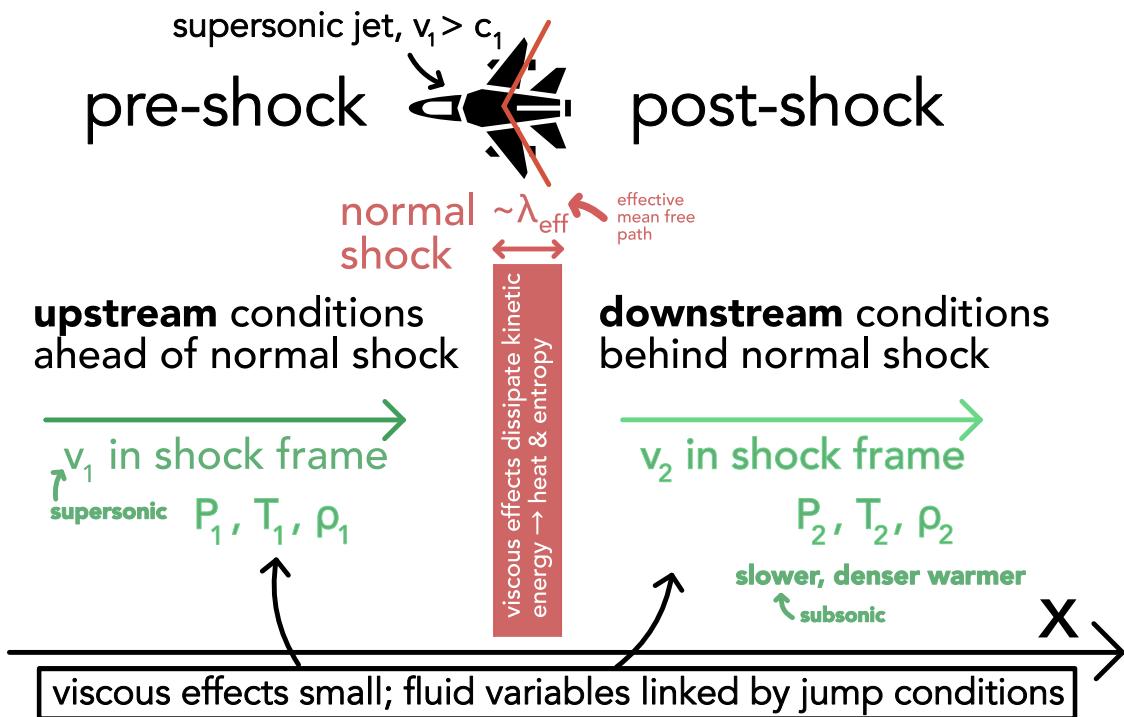


Figure 42: Normal shock

5.4.5.2 (Rankine-Hugoniot) Jump conditions II: Jump condition from the continuity equation

From the above assumption and the continuity equation, we have the following

$$\frac{d}{dx}(\rho v) = 0 \rightarrow \rho v = \text{const.} \rightarrow \rho_1 v_1 = \rho_2 v_2 = j, \quad \text{mass flux } j$$

v_1, v_2 measured in frame of the discontinuity (203)

notation for up-downstream-difference: $[\rho v] = 0$

The mass flux is constant across the discontinuity.

5.4.5.3 (Rankine-Hugoniot) Jump conditions III: Jump condition from the momentum equation

One obtains (for the pre and post shock zones)

$$\frac{d}{dx}(\rho v^2 + P) = 0 \rightarrow [\rho v^2 + P] = 0 \quad \text{span style="float: right;">(204)}$$

Note: We consider the difference in the pre- and post-shock-zones, where viscosity effects can be neglected ($\xi, \eta = 0$) and also $\frac{dv}{dx} = 0$. Note that as the transition zone is typically of a scale λ_{mfp} , we would have to resort to kinetic theory (or plasma particle in cell-codes) there anyways.

5.4.5.4 (Rankine-Hugoniot) Jump conditions IV: Jump condition from the energy equation

We obtain

$$\begin{aligned}
 0 &= \frac{d}{dx} ((\rho e + P)v) = \frac{d}{dx} \left(\rho v \left(e_{th} + \frac{v^2}{2} + \frac{P}{\rho} \right) \right) \\
 &= \rho v \frac{d}{dx} \left(\left(e_{th} + \frac{v^2}{2} + \frac{P}{\rho} \right) \right) + \frac{d(\rho v)}{dx} \left(\left(e_{th} + \frac{v^2}{2} + \frac{P}{\rho} \right) \right) \underset{[\rho v]=0}{=} \rho v \frac{d}{dx} \left(\left(e_{th} + \frac{v^2}{2} + \frac{P}{\rho} \right) \right) \\
 &\rightarrow \boxed{\left[\frac{v^2}{2} + e_{th} + \frac{P}{\rho} \right]} = 0
 \end{aligned} \tag{205}$$

Marked in the boxes are the Rankine-Hugoniot Jump conditions.

We can plug in the closure $e_{th,i} = P_i / (\rho_i \cdot (\gamma_i - 1))$ into the energy jump condition where theoretically γ_i can be different in the pre- and post-shock zones, e.g. when molecules are dissociated.

5.4.5.5 Types of discontinuities: contact discontinuity vs. shock

The continuity of mass flux allows two scenarios

- **tangential discontinuity:** $\rho_1 v_1 = \rho_2 v_2 = 0$ so as $\rho_1, \rho_2 \neq 0$ in general, $v_1 = v_2 = 0$ so $[P] = 0$ as of $[\rho v^2 + P] = 0$. The pre- and post-shock zones move with the same velocity as the shock, there is no mass-flux through the shock. If the tangential velocities are also continuous ($[v_y] = [v_z] = 0$) (which we do not consider by our current assumptions), this is called a **contact discontinuity**. The density may jump but as of $[P] = 0$, T then has to do the *opposite* jump. A contact discontinuity is a surface separating two fluids with different physical properties.
- **shock:** for $\rho_1 v_1 = \rho_2 v_2 \neq 0$ we have mass flux and thus a shock. Shock waves do propagate with respect to the fluid because of the mass flux in the normal.

5.4.6 Characterizing the Shock strength - Mach number

Note: In the shock's frame of reference, the unshocked material is moving at speed v_1 and the shocked material at speed v_2 .

The (pre-shock) Mach number is the ratio between the upstream (with respect to the shock) velocity to the upstream sound speed, characterizing the strength of a shock ($\rho_1 v_1 \neq 0$) (in case of a jet causing the upstream velocity, the jet's velocity is used⁶)

$$\mathcal{M}_1 \equiv \frac{v_1}{c_{s,1}} \underset{c_s^2 = \frac{\gamma P}{\rho}}{=} \sqrt{\frac{\rho_1 v_1^2}{\gamma P_1}} \underset{P = \frac{\rho}{m} k_B T}{=} \sqrt{\frac{mv_1^2}{\gamma k_B T_1}} \quad (206)$$

we can analogously define a post-shock Mach number \mathcal{M}_2 .

$$\mathcal{M}_2 \equiv \frac{v_2}{c_{s,2}} \quad (207)$$

Equivalently, the Mach number is

- ratio of ram pressure ρv^2 (pressure as of fluid's bulk motion, not the thermal motion) to thermal pressure (for \mathcal{M}_1 in the pre-shock zone)
- and (as pressure is also an energy density) the kinetic thermal energy density

Note: Below $\mathcal{M} = 1$ there is no shock, the flow is subsonic. A shock occurs, when supersonic flow (e.g. Solar Wind) encounters an obstacle forcing a change in velocity (e.g. the Earth's magnetosphere \rightarrow bow shock).

5.4.6.1 Occurrence of the Mach number in the continuity equation

Rewriting $\partial_t \rho + \nabla \cdot (\rho \underline{v}) = 0$ using $\frac{D}{Dt} = D_t = \partial_t + \underline{v} \cdot \nabla$ to $-\frac{1}{\rho} \frac{D\rho}{Dt} = \nabla \cdot \underline{v}$ and using the adiabatic $dP = c_s^2 d\rho$ we get

$$-\frac{1}{\rho c_s^2} \frac{D\rho}{dt} = \nabla \cdot \underline{v} \quad \xrightarrow{\text{dimensionless form}} \quad -\mathcal{M}^2 \frac{1}{\hat{\rho}} \frac{D\hat{\rho}}{D\hat{t}} = \hat{\nabla} \cdot \hat{\underline{v}} \quad (208)$$

So in the limit $\mathcal{M} \rightarrow 0$ we have incompressible flow.

⁶If a jet is flying sufficiently fast, some of its energy goes into compressing the air in front. If the jet itself moves faster than c_s , the information speed in the air, shock waves form as of those compressions (they cannot spread sufficiently fast for there not to be a shock).

5.4.6.2 Rewriting the Rankine-Hugoniot jump conditions in terms of \mathcal{M}_1 - relating pre- and post-shock quantities

One can rewrite the jump conditions in terms of the Mach number \mathcal{M}_1 (assume $\gamma_1 = \gamma_2 = \gamma$ (here without proof)

$$\begin{aligned}\frac{\rho_2}{\rho_1} &= \frac{v_1}{v_2} = \frac{(\gamma + 1)\mathcal{M}_1^2}{(\gamma - 1)\mathcal{M}_1^2 + 2} \xrightarrow{\gamma=1} \mathcal{M}_1^2 \\ \frac{P_2}{P_1} &= \frac{\rho_2 k_B T_2}{\rho_1 k_B T_1} = \frac{2\gamma\mathcal{M}_1^2 - (\gamma - 1)}{\gamma + 1} \xrightarrow{\gamma=1} \mathcal{M}_1^2 \\ \frac{T_2}{T_1} &= \frac{[(\gamma - 1)\mathcal{M}_1^2 + 2][2\gamma\mathcal{M}_1^2 - (\gamma - 1)]}{(\gamma + 1)^2\mathcal{M}_1^2} \xrightarrow{\gamma=1} 1\end{aligned}\quad (209)$$

from which we for a strong shock ($\mathcal{M}_1 \gg 1$)⁷ can find (use $\gamma = 5/3$ for an ideal non-relativistic gas)

$$\begin{aligned}\frac{\rho_2}{\rho_1} &= \frac{v_1}{v_2} \approx \frac{\gamma + 1}{\gamma - 1} = 4, \\ P_2 &\approx \frac{2\gamma}{\gamma + 1} \mathcal{M}_1^2 P_1 = \frac{2}{\gamma + 1} \rho_1 v_1^2 = \frac{3}{4} \rho_1 v_1^2, \\ k_B T_2 &\approx \frac{2\gamma(\gamma - 1)}{(\gamma + 1)^2} k_B T_1 \mathcal{M}_1^2 = \frac{2(\gamma - 1)}{(\gamma + 1)^2} m v_1^2 = \frac{3}{16} m v_1^2,\end{aligned}\quad (210)$$

In the shock frame, the post-shock medium is slower, denser, has higher-pressure and is warmer, see figure 43.

5.4.6.3 Conversion of kinetic to thermal energy in the shock

Based on the above relations for $\mathcal{M}_1 \gg 1, \gamma = 5/3$ we can write

$$\begin{aligned}\text{post-shock specific kinetic energy: } \frac{1}{2} v_2^2 &\approx \frac{1}{16} \frac{1}{2} v_1^2 \\ \text{post-shock specific thermal energy: } \frac{3}{2} \frac{k_B T_2}{m} &\approx \frac{9}{32} v_1^2 = \frac{9}{16} \frac{1}{2} v_1^2\end{aligned}\quad (211)$$

We find that in the shock frame, roughly half of the pre-shock kinetic energy ($\frac{9}{16}$) is converted to thermal energy.

5.4.6.4 Conservation of energy in the shock

In the pre-shock flow (for a strong shock) we can neglect the thermal energy, so $e_1 = \frac{v_1^2}{2}$ (specific energy per particle).

⁷In a strong shock $v_1^2 \gg c_1^2$, so the thermal pressure of the unshocked gas is negligible to its ram pressure.

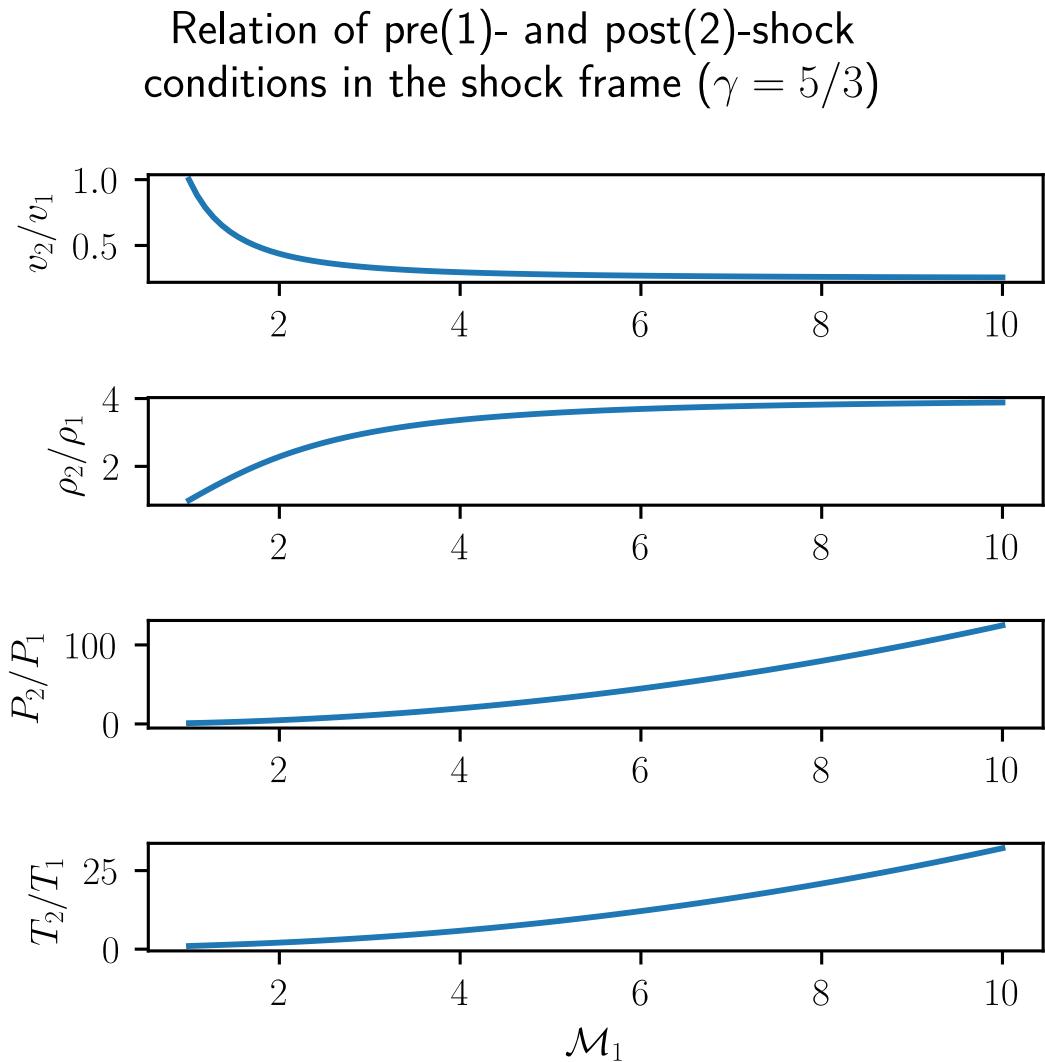


Figure 43: Relation of pre- and post-shock quantities (shock $\rightleftharpoons \mathcal{M}_1 = \frac{v_1}{c_1} > 1$). v_1 is the velocity of the upstream (pre-shock) fluid with respect to the shock.

In

$$\frac{1}{2}v_2^2 + \frac{3}{2}\frac{k_B T_2}{m} \approx \frac{10}{16}\frac{v_1^2}{2} \quad (212)$$

(in the shock rest frame) one is missing the pdV work, which is done by the shock to compress the post-shock gas and amounts to $k_B T_2 \approx \frac{6}{16}\frac{1}{2}v_1^2$.

Note: The sum of enthalpy (thermal energy + pdV work) and kinetic energy is conserved in adiabatic flow (even when non-adiabatic processes like shocks occur between the two sections). Enthalpy plays the same role in a flowing system that internal energy takes in a non-flowing one, taking care of the energy associated with flow work in / out of the control volume.

Note: There are also radiative shocks, where in the transition through the shock energy is radiated away.

Note: In the rest frame of the post-shock gas there is no PdV term.

5.4.6.5 Connection between pre- and post-shock Mach number

We can write the post-shock Mach number as

$$\mathcal{M}_2 = \frac{v_2}{c_2} = \frac{v_1}{c_1} \frac{v_2}{v_1} \frac{c_1}{c_2} \underset{c^2 \propto T}{=} \mathcal{M}_1 \frac{v_2}{v_1} \left(\frac{T_1}{T_2} \right)^{\frac{1}{2}} \quad (213)$$

Plugging in the jump condition for T_2/T_1 , in the strong shock limit we get

$$\mathcal{M}_2 = \left(\frac{\gamma - 1}{2\gamma} \right)^{\frac{1}{2}} \underset{\gamma=5/3}{\approx} 0.45 \quad (214)$$

Summary: Supersonic gas is slowed down (to subsonic), compressed (density, pressure and temperature increase) by a shock.

5.4.6.6 Shock adiabatic curve*

Note: In shocks, the post-shock entropy is increased with respect to the pre-shock entropy
- the shock shifts the gas to a higher adiabatic curve.

The shock is a non-adiabatic process ($\delta Q \neq 0 \rightarrow dS = \frac{\delta Q}{T} \neq 0$). Based on the first law of thermodynamics $\delta Q = dE + PdV$, the ideal gas law and $dE = \nu c_V dT$ (number of mols ν), we can one can find for an ideal polytropic gas that $s = c_V \ln \left(\frac{p}{\rho^\gamma} \right)$, so here

$$s_2 - s_1 = c_V \ln \left(\frac{p_2}{p_1} \left(\frac{\rho_1}{\rho_2} \right)^\gamma \right) = c_V \ln \left(\frac{K_2}{K_1} \right) \quad (215)$$

The shock shifts the gas to a higher adiabatic curve $K = P\rho^{-\gamma}$. Note that one finds using the jump conditions, that $s_2 - s_1 > 0$ only for $\mathcal{M}_1 > 1$, so there is only a shock for $\mathcal{M}_1 > 1$.

Based on $j = \rho_1 v_1 = \rho_2 v_2$ and $[\rho v^2 + P] = 0$ the slope of the shock adiabatic curve in a PV-diagram is

$$\frac{j^2}{m} = \frac{P_2 - P_1}{V_1 - V_2} \quad (216)$$

$P = \text{const.} \times \rho^\gamma$ on either side of the shock (where we assume equilibrium), but with different

constants.

Note: The jump conditions are *reversible* - if we interchange post- and pre-shock flow conditions, so $v_1 < v_2$, then density, pressure and temperature would decrease across the shock and so the entropy, excluding a shock with deceleration.

5.4.6.7 Oblique shocks

The fluid might not impact the shock perpendicular to the shock front, but at an oblique angle. **Result:** The shock deflects the flow away from the shock's normal direction (towards the shock's surface), the final velocity may remain supersonic. Only the velocity component normal to the shock front changes, V_t is continuous across the shock (see figure 44).

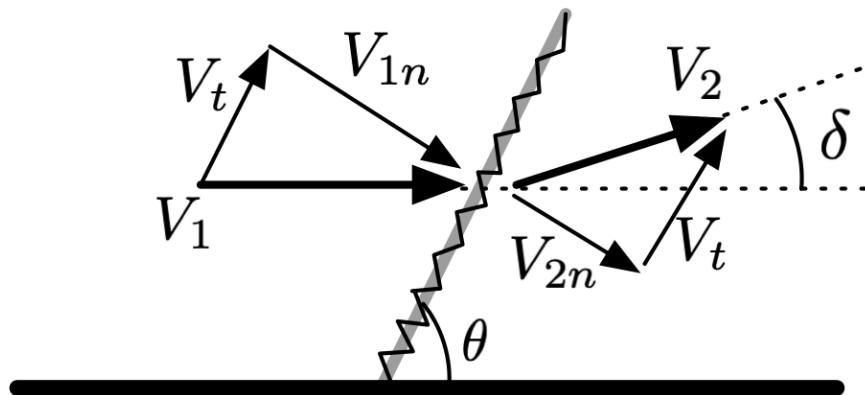


Figure 44: Oblique shock

Derivation - oblique jump conditions: Let \underline{n} (here = \hat{e}_x) be the shock normal, so $v_{\parallel} = \underline{v} \cdot \underline{n}$ is the component of the fluid velocity \underline{v} parallel and v_{\perp} the one perpendicular to \underline{n} . The Navier-Stokes equation describes conservation of momentum in form of a continuity equation and as such contains a momentum current. This current across the shock, $\rho\underline{v}(\underline{v} \cdot \underline{n})$, is continuous across the shock, yielding

$$\begin{aligned} [\rho v_x^2 + P] &= 0 \\ [\rho v_x v_y] &= 0 \\ [\rho v_x v_z] &= 0 \end{aligned} \tag{217}$$

As we are dealing with a shock with momentum flux $[\rho v_x] \neq 0$, we get

$$[v_y] = 0, \quad [v_z] = 0 \tag{218}$$

so as stated, the tangential velocities are continuous across the shock, $v_{1,\perp} = v_{2,\perp} = v_{\perp}$ and

for $v_{2,\parallel}$ we have $v_{2,\parallel} = v_{1,\parallel} \frac{\rho_1}{\rho_2}$ ($\rho_2 > \rho_1$). So if the component parallel to the shock normal gets smaller and the one perpendicular remains the same, we are deflected away from the shock normal.

5.5 Fluid instabilities

Instabilities are the rapid growth of small perturbations, tapping into a source of free energy.

5.5.1 Stability of a shear flow

We consider two flows counterpropagating side by side (see figure 45). Using perturbation theory, the stability of the flow can be analyzed - if the dispersion relation for a perturbation yields a positive imaginary part (not just yeal oscillation) such modes grow exponentially - the flow is unstable.

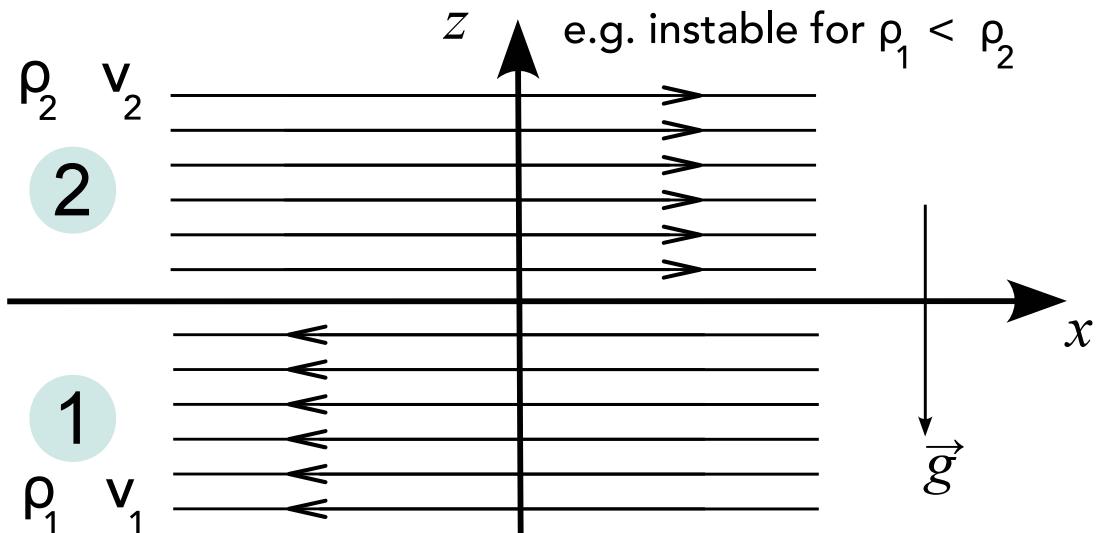


Figure 45: Shear flow

5.5.2 Rayleigh-Taylor instability

Here we consider a fluid at rest, i. e. $v_1 = v_2 = 0$. If the denser fluid lies on top, there are unstable solution - Rayleigh-Taylor instability. The instability is driven by the buoyancy of the lighter fluid or rather the release of potential energy with respect to the external force g (see figure 46). If the denser fluid is on the bottom, the interface is stable and will only oscillate if perturbed. There is also the Rayleigh-Taylor instability in plasmas.

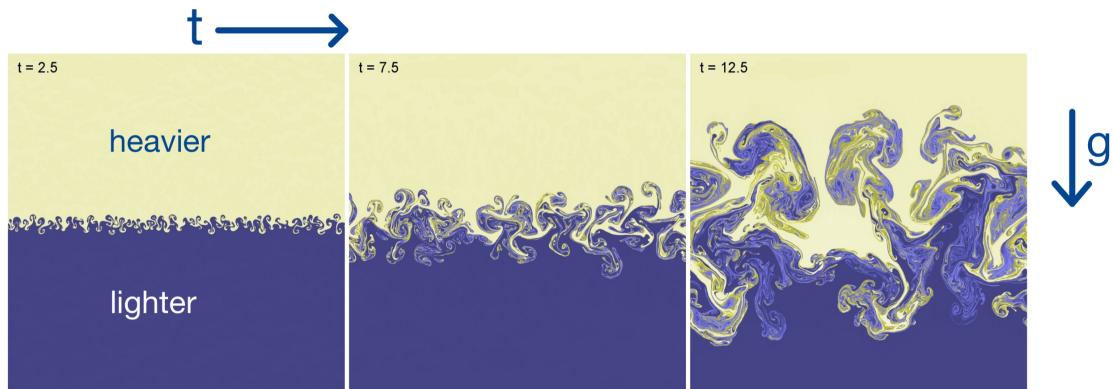


Figure 46: Rayleigh-Taylor instability

5.5.3 Kelvin-Helmholtz instability

Consider the case without the gravitational field $\underline{g} = 0$. In an ideal gas, small wave like perturbances will grow into large waves (with largest growth for large k , so small wavelengths) - Kelvin-Helmholtz-Billows, which subsequently roll up to vortex-like structures. Sharp velocity gradients are unstable - we can create turbulence. Some modes can be stabilized against the instability if we have the gravitational field, the heavy part is on the bottom (otherwise Rayleigh-Taylor instability) and the velocity difference $(v_2 - v_1)^2$ is sufficiently small.

5.5.4 Further instabilities

- **Richtmyer-Meshkov instability:** at suddenly accelerated interfaces
- **Jeans-instability:** in self-gravitating fluids, where denser regions can grow and collapse under their own attraction
- **Thermal instability:** ...

5.6 Turbulence

»Big whirls have little whirls that feed on their velocity, and little whirls have lesser whirls and so on to viscosity.« - Lewis Fry Richardson

Both laminar and turbulent flow are solutions to the deterministic Navier-Stokes equations, however turbulent flow is chaotic, laminar not (see table 9 and figure 47).

Laminar flow	Turbulent flow
Fluid flows in parallel layers with no disruption between those layers	Unsteady, chaotic flow with varying velocity and pressure in position and time
similar conditions → similar solutions	infinitesimal difference in conditions → vastly different solutions (deterministic chaos ⁸)

Table 9: Laminar vs. turbulent flow

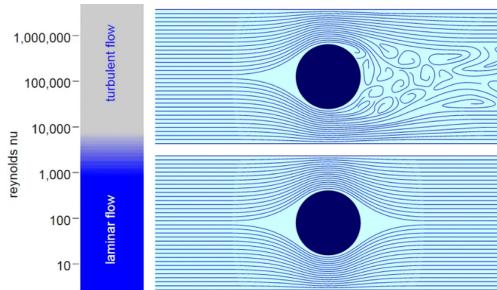


Figure 47: Laminar vs. turbulent flow

5.6.1 Subsonic (incompressible) turbulence, low Mach numbers | rotational modes

For subsonic turbulence, the information speed is far higher than the transport speed limiting the naturally occurring compressions (no shocks), so we can assume the fluid to be incompressible, $\nabla \cdot \underline{v} = 0$.

In Fourier space the nabla operator becomes $\nabla \rightarrow \underline{k}$, so $\underline{k} \cdot \underline{v} = 0$, so there are no longitudinal disturbances (aka soundwaves) but only solenoidal, i.e. source free motion, so shear flow and rotational turbulence (swirling eddies as for instance produces by the Kelvin-Helmholtz instability).

$\underline{v} \cdot \underline{v} = 0$ also implies subsonic flow as supersonic velocities would cause shocks coming with compression of the post-shock fluid.

Incompressible turbulence is described by Kolmogorov's theory of incompressible turbulence.

Incompressible flow is governed by the Navier-Stokes equation for an incompressible fluid, so

$$\partial_t \underline{v} + \overbrace{(\underline{v} \cdot \nabla) \underline{v}}^{\text{advective transport}} = \underline{g} - \underbrace{\frac{1}{\rho} \nabla P}_{\text{pressure force}} + \overbrace{\nu \nabla^2 \underline{v}}^{\text{viscous dissipation}} \quad (219)$$

5.6.2 How to quantify turbulence? - Reynolds number

Let us come back to the Reynolds number characterizing the ratio of the advective to the friction term in the Navier-Stokes equation

$$Re = \frac{\text{advective term}}{\text{frictional term}} = \frac{|(\underline{v} \cdot \nabla) \underline{v}|}{\nu \underline{\nabla^2} \underline{v}} = \frac{V_0 L_0}{\nu}$$

with V_0 = characteristic velocity, L_0 = characteristic length scale, (220)

$$\nu = \text{kinematic viscosity} = \frac{\eta}{\rho} \sim \lambda_{mfp} v_{th}$$

with λ_{mfp} = mean free path, v_{th} = thermal velocity

Although counterintuitive, the viscosity increases with larger mean free path, intuitively as shear stress information is transported over larger distances.

In this view a high Reynolds number means that turbulence is generated faster by the chaotic advective term than is destroyed via dissipation.

For approximately $Re > 3.5 \cdot 10^3$ turbulence is expected, the interstellar medium has $Re \sim 10^8$. Oceans (viscosity of water $\sim 10^{-6} \text{ m}^2 \text{ s}^{-1}$) and atmosphere are always turbulent, except for boundary layers (with small characteristic length scale).

5.6.2.1 Reynolds number as the ratio between advection and dissipation timescale

Most simply by dimensional analysis, one can yield (ν in units of $\frac{\text{m}^2}{\text{s}}$, a diffusion coefficient)

$$t_{adv} = \frac{L_0}{V_0}, \quad t_{dis} = \frac{L_0^2}{\nu} \quad \rightarrow \quad Re = \frac{t_{dis}}{t_{adv}} = \frac{L_0 V_0}{\nu} \sim \frac{L_0 V_0}{\lambda_{mfp} v_{th}} \quad (221)$$

where a high Reynolds number means that advection is faster than dissipation, thus dissipation cannot stabilize the growth of turbulence sufficiently and we have turbulent flow. In the equation we can also see that the Reynolds number is the product of the macroscopic-to-microscopic length and velocity scales.

5.6.3 Supersonic turbulence, shocks $\mathcal{M} \gg 1$ | rotational and compressive modes

Depending on the dimensionality, we have

- 1D: 1 compressive mode
- 2D: 1 compressive mode, 1 solenoidal mode
- 3D: 1 compressive mode, 2 solenoidal modes

5.6.4 Schematic concept of turbulence

In 3D

- **injection range** energy is injected on macroscopic scales, typical scale L , velocity v
- **inertial range** large eddies break up into smaller eddies and energy is transferred to smaller scales, vorticity ($\zeta = \nabla \times \underline{v}$) is conserved and no energy is dissipated
- **dissipation range** at the microscopic viscous scale (λ_{visc} , roughly the mean free path λ_{mfp}) energy is dissipated into viscous heat

In 2D the energy flow is reverted from small to large (inverse cascade).

5.6.5 Kolmogorov scales of turbulence

In the following consider

$$\begin{aligned} & \text{largest eddy scale: } L_S, \quad \text{dissipation scale: } L_k \\ & \text{rate of energy dissipation on small scales, energy flow in the inertial range: } \epsilon \\ & \text{some eddy size: } \lambda, \quad \text{velocity on that scale: } v_\lambda \\ & \text{fluid viscosity: } \nu \text{ in } \frac{m^2}{s} \end{aligned} \tag{222}$$

5.6.5.1 Dissipation scale - smallest scale to be resolved in a simulation

Assume high Reynolds number and start at the small scale. At the small scale, turbulent motions are statistically isotropic (different from the large scales L_S) and Kolmogorov postulates the statistics to be universally determined by ν and ϵ (in m^2 / s^3). From dimensional analysis (physical units), one can then get

$$L_K \sim \left(\frac{\nu^3}{\epsilon} \right)^{\frac{1}{4}} \tag{223}$$

Note: This is the smallest scale to be resolved in a classical simulation, for an airplane with chord length 2 m this is $\mathcal{O}(10^{-6})$ m - quite a problem for simulations.

5.6.6 Scaling of the eddy velocity and vorticity in the inertial range

Energy flow ϵ must be constant in the inertial range (otherwise accumulation). We approximate the energy flow by the kinetic energy of an eddy divided by its characteristic time

scale

$$\epsilon \approx \left(\frac{v_\lambda^2}{2} \right) \left(\frac{v_\lambda}{\lambda} \right) \approx \frac{v_\lambda^3}{\lambda} \underset{\epsilon=\text{const.}}{\approx} \frac{v_{L_S}^3}{L_S} \quad (224)$$

therefore, we get

$$v_\lambda \approx v_{L_S} \left(\frac{\lambda}{L_S} \right)^{\frac{1}{3}} \quad (225)$$

Note: The largest eddies have highest velocities.

But as the size scales down quicker than the velocity, smallest eddies have the highest vorticity

$$|\zeta_\lambda| \approx \frac{v_\lambda}{\lambda} \approx \frac{v_{L_S}}{(\lambda^2 L_S)^{\frac{1}{3}}} \quad (226)$$

Note: As the vorticity increases with decreasing scale but overall vorticity is approximately constant (\sim conservation of angular momentum) on smaller scales less volume is filled with turbulent eddies.

5.6.7 Power spectrum of Kolmogorov turbulence

The dissipation is reflected by the energy spectrum of Kolmogorov turbulence, see figure 48.

The constant energy transfer through the cascade is described by

$$\begin{aligned} &\text{incompressible fluid, subsonic: } E(k) \propto k^{-\frac{5}{3}} \\ &\text{compressible, shock-dominated: } E(k) \propto k^{-2} \end{aligned} \quad (227)$$

Note: Including the the dissipation range, based on Kolmogorov's assumption that the statistics for small scale motion are universal, one might make the ansatz $E(k) = C\epsilon^{\frac{2}{3}}k^{-\frac{5}{3}}f_\eta(k\eta)$ with $f_\eta(x) = 1$ for $x \ll 1$ and $f_\eta(x) \rightarrow 0$ for $x \rightarrow \infty$.

5.6.7.1 Derivation of the energy spectrum of Kolmogorov turbulence

See Springel et al., 2023.

Maybe add derivation of Kolmogorov spectrum.

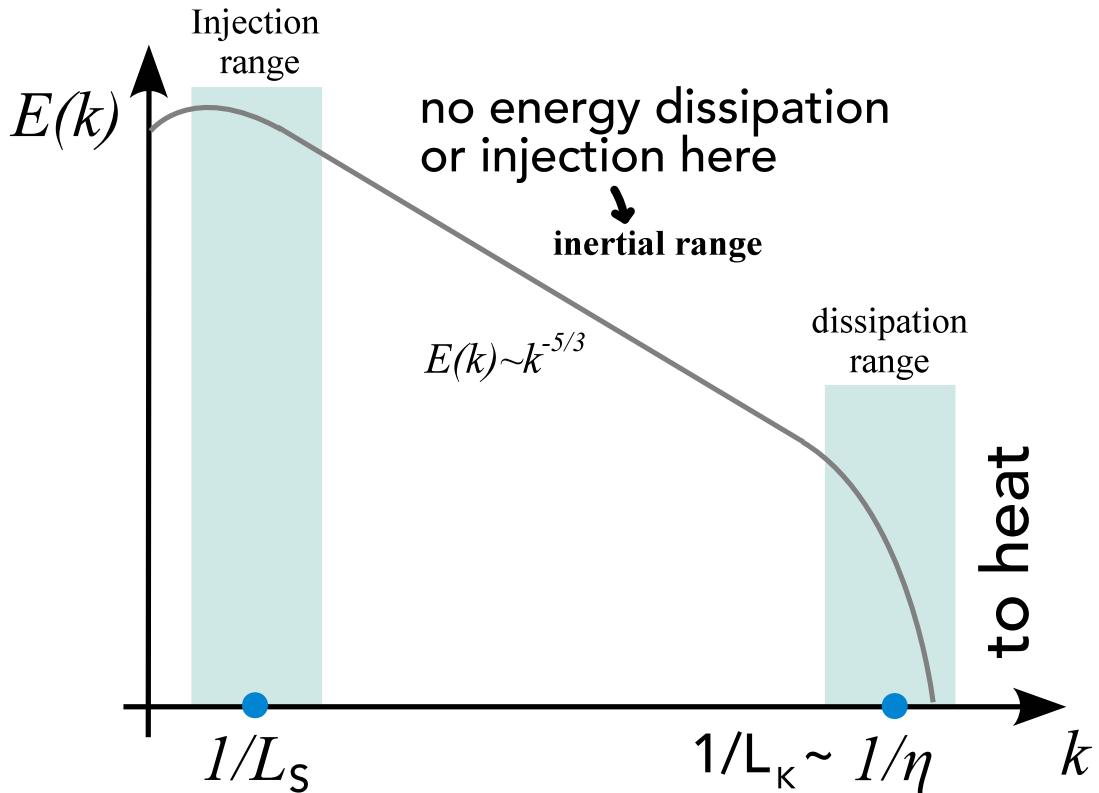


Figure 48: Kolmogorov energy spectrum

6 Eulerian Hydrodynamics | Solving PDEs

As our overarching aim is simulating physical systems (e.g. how do perturbations evolve in a fluid) and most physical systems are described by partial differential equations (PDEs) (like the Euler equations), we turn our heads towards numerically solving PDEs. We search for functions complying to $\mathcal{P}[u] = 0$ (and boundary conditions) with \mathcal{P} being a differential operator.

Note: This section focuses on solvers following fluid variables on a fixed grid - eulerian hydrodynamics.

6.1 Introductory notes on PDEs

Partial differential equations (like Euler and Navier Stokes equation, Maxwell's equation, ...) describe relations between partial derivatives of a dependent variable (e.g. fluid density) with respect to several independent variables (e.g. time and position), like

$$\partial_t u = \partial_x^2 u, \quad \text{dependent variable } u(x, t), \quad \text{independent variables } x, t \quad (228)$$

(a kind of 1D diffusion equation).

Note: There is no general approach for solving PDEs.

6.2 Types of PDEs

PDEs are classified by

- **Order of the PDE:** Order of the highest occurring derivative
- **Linearity:** If the dependent variable and all its derivatives only occur linearly (no $\sqrt{\partial_x u}$), the PDE is linear and if u_1 and u_2 are solutions to the PDE, $c_1 u_1 + c_2 u_2$ are as well (superposition)
- **Homogeneity:** The PDE is homogeneous, if all terms contain the dependent variable or its derivatives, so if there is no source term

6.2.1 Classification of linear 2nd order PDEs in analogy with conic sections

PDEs can be distinguished into different types which give clues about appropriate solution strategies as well as appropriate initial and boundary conditions and the smoothness of the solution.

Consider a 2nd order linear PDE with two independent variables, so generally

$$a \frac{\partial^2 u}{\partial x^2} + b \frac{\partial^2 u}{\partial x \partial y} + c \frac{\partial^2 u}{\partial y^2} + d \frac{\partial u}{\partial x} + e \frac{\partial u}{\partial y} + f u = g, \quad a, b, c \text{ not all zero} \quad (229)$$

6.2.1.1 Derivation | homogeneous solutions are conic section in k -space

The unknown function u is expanded into plane waves (Fourier transformation)

$$u(\underline{x}) = \frac{1}{(2\pi)^2} \int \hat{u}(\underline{k}) \exp(-i\underline{k} \cdot \underline{x}) d^2 k, \quad \underline{k} =: \begin{pmatrix} k \\ l \end{pmatrix} \quad (230)$$

Plugging this into the PDE yields

$$-\frac{1}{(2\pi)^2} \int (\cancel{a k^2} + \cancel{b k l} + \cancel{c l^2} + \cancel{i d k} + \cancel{i d k} - \cancel{f}) \hat{u}(\underline{k}) \exp(-i\underline{k} \cdot \underline{x}) d^2 k = g \quad (231)$$

For the homogeneous solution ($g = 0$) this must be zero, so

$$\underline{k}^T \underline{\Delta} \underline{k} + i \begin{pmatrix} d \\ e \end{pmatrix}^T \underline{k} - f = 0, \quad \underline{\Delta} = \begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix} \quad (232)$$

which is the matrix representation of a conic section in \underline{k} -space (why?).

6.2.1.2 Classification into elliptic, parabolic, hyperbolic

$$\text{The PDE is } \begin{cases} \text{hyperbolic if } D > 0 \\ \text{parabolic if } D = 0, & \Delta = \begin{vmatrix} a & b/2 \\ b/2 & c \end{vmatrix}, \quad D = -4\Delta = b^2 - 4ac \end{cases} \quad (233)$$

6.2.1.3 Qualitative differences on the types of PDEs

- Elliptic

- often describe static problems (no time dependence) like the Poisson equation ($\underline{\nabla}^2 \phi = -\frac{\rho}{\epsilon}$)
- as smooth as coefficients allow in the interior region where the equation and solutions are defined (independent of the smoothness of the boundary conditions)

- Parabolic

- often of second order, describing slowly changing processes like diffusion, becoming smoother with time
- the problem is described using an initial state $u(x, t_0)$ as well as boundary conditions
- all parabolic PDEs can be transformed into a form analogous to the heat equation by change of independent variables

- Hyperbolic

- typically describe dynamical processes in physics
- initial conditions are specified by $u(x, t_0)$, $\frac{\partial u}{\partial x}(x, t_0)$ and higher derivatives as necessary as well as boundary conditions
- disturbances have finite propagation speed
- solutions can develop steep regions and real discontinuities

Boundary types: Fixed function values on the boundaries = Dirichlet boundary; fixed value of the derivative = van Neumann boundary.

Equation	Classification
Laplace equation $\nabla^2 u = \partial_x^2 u + \partial_y^2 u = 0 \quad (a = 1, b = 0, c = 1) \quad (234)$	Elliptic
1D Heat conduction equation (diffusion equation) $\partial_t u - \lambda^2 \partial_x^2 u = 0 \quad (a = 0, b = 0, d = 1, c = -\lambda^2) \quad (235)$	Parabolic
1D Wave equation $\partial_t^2 u - c_s^2 \partial_x^2 u = 0 \quad (a = 1, b = 0, c = -c_s^2) \quad (236)$	Hyperbolic

Table 10: Typical examples and classification of homogeneous 2nd order PDEs

6.2.2 Typical examples and classification of homogeneous 2nd order PDEs

Note: The classification scheme introduced is for 2nd order PDEs with two variables, it is not made e.g. for the advection equation $\partial_t u + v \partial_x u = 0$ which is first order and hyperbolic.

6.2.3 Classification of linear 2nd order PDEs with more unknowns

For the general form

$$\sum_{i,j=1}^n A_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^n b_i \frac{\partial u}{\partial x_i} + cu + d = 0, \quad \underline{A} \text{ with entries } A_{ij} \quad (237)$$

regarding the eigenvalues of \underline{A} (following from $\underline{A}\underline{v} = \lambda\underline{v} \rightarrow \det(\underline{A} - \lambda\underline{I}) = 0$) it holds that

$$\text{The PDE is } \begin{cases} \text{hyperbolic if one negative, rest positive or one positive, rest negative} \\ \text{parabolic if one zero, others all positive or all negative} \\ \text{elliptic if all positive or all negative} \end{cases} \quad (238)$$

Note that this classification is not exhaustive, the occurrence of multiple different signs is sometimes called ultra-hyperbolic.

Task to the reader: Check that for $n = 2$ this classification scheme is equivalent to the previous one.

6.2.4 Linear systems of 1st order homogeneous PDEs

Question: When is a 1st order homogeneous PDE hyperbolic?

A first order homogeneous PDE is of the form

$$\partial_t \underline{u}_i + \sum_{j=1}^n A_{ij} \partial_{x_j} u_i = 0, \quad i = 1, \dots, n \quad (239)$$

with short-hand notation

$$\partial_t \underline{u} + (\underline{\underline{A}} \nabla_x) \underline{u} = 0, \quad \underline{\underline{A}} \text{ with entries } A_{ij}, \quad \nabla_x = \text{diag}(\partial_{x_1}, \dots, \partial_{x_n}) \quad (240)$$

where **the system is hyperbolic if $\underline{\underline{A}}$ has real eigenvalues and is diagonalizable.**

6.2.4.1 Extension to conservation laws

We extend this to conservation laws

$$\partial_t \underline{u} + \nabla_x \cdot \underline{\underline{F}}(\underline{u}) = \partial_t \underline{u} + (\nabla_{\underline{u}} \underline{\underline{F}}(\underline{u})) \nabla_x \underline{u} = 0 \quad (241)$$

with conserved variable \underline{u} and flux matrix $\underline{\underline{F}}(\underline{u})$. If $\nabla_{\underline{u}} \underline{\underline{F}}(\underline{u})$ is diagonalizable with real eigenvalues, the system is hyperbolic.

Consider e.g. the Navier-Stokes equation

$$\partial_t (\rho \underline{v}) + \nabla \cdot (\rho \underline{v} \underline{v}^T + \underline{\underline{P}} \underline{\underline{I}} - \underline{\underline{\Pi}}) = \rho \underline{g} \quad (242)$$

where $\nabla \cdot$ here is a matrix divergence (a vector) with entries as expected for such a matrix vector multiplication.

Note: The classification introduced has its limits: What type the Navier-Stokes equation is seems different to tell, often *hyperbolic* is used as *advection-dominated* (the advection equation is hyperbolic) and *parabolic* as *diffusion-dominated* (the diffusion equation is parabolic) and then the Navier Stokes equation can be either depending on the Reynolds number.

Solving an elliptic problem by solving a hyperbolic problem to steady state*:
Consider for instance the hyperbolic Poisson equation

$$\underline{\nabla}^2 \phi = 4\pi G\rho \quad (243)$$

from which hyperbolic gravity equations can be obtained as

$$\frac{\partial}{\partial \tau} \begin{bmatrix} \phi \\ q_1 \\ q_2 \end{bmatrix} + \frac{\partial}{\partial x} \begin{bmatrix} -q_1 \\ -\phi/T_r \\ 0 \end{bmatrix} + \frac{\partial}{\partial y} \begin{bmatrix} -q_2 \\ 0 \\ -\phi/T_r \end{bmatrix} = \begin{bmatrix} -4\pi G\rho \\ -q_1/T_r \\ -q_2/T_r \end{bmatrix}, \quad \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} \approx \underline{\nabla} \phi \quad (244)$$

pseudotime τ , relaxation time T_r

which can then be used in the (compressible) Euler equations

$$\frac{\partial}{\partial t} \begin{bmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ E \end{bmatrix} + \frac{\partial}{\partial x} \begin{bmatrix} \rho v_1 \\ \rho v_1^2 + p \\ \rho v_1 v_2 \\ (E+p)v_1 \end{bmatrix} + \frac{\partial}{\partial y} \begin{bmatrix} \rho v_2 \\ \rho v_1 v_2 \\ \rho v_2^2 + p \\ (E+p)v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -\rho \partial_x \phi \\ -\rho \partial_y \phi \\ -(v \cdot \underline{\nabla} \phi) \rho \end{bmatrix} \quad (245)$$

$p = (\gamma - 1) \left(E - \frac{1}{2} \rho (v_1^2 + v_2^2) \right)$

to evolve a self-gravitating fluid only using a hyperbolic solver (e.g. a purely hyperbolic discontinuous Galerkin approach as in Schlottke-Lakemper et al., 2021).

6.3 Solution schemes for PDEs

There is no general approach but multiple general methods for different types or even only a certain PDE. Common methods are

- **Finite difference methods:** Differential operators are approximated by finite difference operators, usually on a regular (Cartesian) mesh
- **Finite volume methods:** Useful for hyperbolic conservation laws. We consider quantities averaged over finite volumes around mesh cells where divergence terms in a PDE turn into fluxes through the cells surface (\rightarrow conservative method, fluxes are not lost). Exact expressions for the average value of the solution over some volumes are calculated and from these averages solutions within the cells can be reconstructed. This contrasts with finite difference methods where derivatives are approximated based on nodal values and finite element methods where local approximations are made using local data and stitched together to a global solution.

- **Spectral methods:** The PDE is converted into an algebraic form (e.g. by Fourier transform), the solution is represented by a linear combination of functions (e.g. the plane waves from the inverse Fourier transform).
- **Method of lines:** All derivatives but one are approximated by finite differences, leading to an ODE system, where we can use ODE solvers. **Time-dependent problems:** Consider a time-dependent problem on a 1D grid, $x_i, i = 1, \dots, N$. Each point yields an ODE for the time evolution of the solution at that point, dependent on e.g. the solution on neighboring points $\rightarrow N$ coupled ODEs.
- **Finite element methods:** The simulation domain is divided into cells (elements, arbitrary shape, unstructured mesh) and the solution on each cell approximated in form of a simple (polynomial) function. The solutions from the cells are linearly combined and we solve for the coefficients.

Example for the method of lines: Consider the 1D heat equation

$$\partial_t u - \lambda^2 \partial_x^2 u = 0 \quad (246)$$

How to discretize the 2nd order derivative $\partial_x^2 u$?: Second order Taylor expansion yields

$$\begin{aligned} u(x + \Delta x) &= u(x) + \Delta x \partial_x u(x) + \frac{1}{2} \Delta x^2 \partial_x^2 u(x) + \mathcal{O}(\Delta x^3) \\ u(x - \Delta x) &= u(x) - \Delta x \partial_x u(x) + \frac{1}{2} \Delta x^2 \partial_x^2 u(x) + \mathcal{O}(\Delta x^3) \end{aligned} \quad (247)$$

where adding both yields

$$\partial_x^2 u(x) \approx \frac{u(x + \Delta x) - 2u(x) + u(x - \Delta x)}{\Delta x^2} \quad (248)$$

We therefore get

$$\partial_t u_i - \lambda^2 \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = 0, \quad i = 1, \dots, N \quad (249)$$

which we could for instance approach using the Euler method. The grid is illustrated in fig. 49.

Problem: This scheme might not be stable.

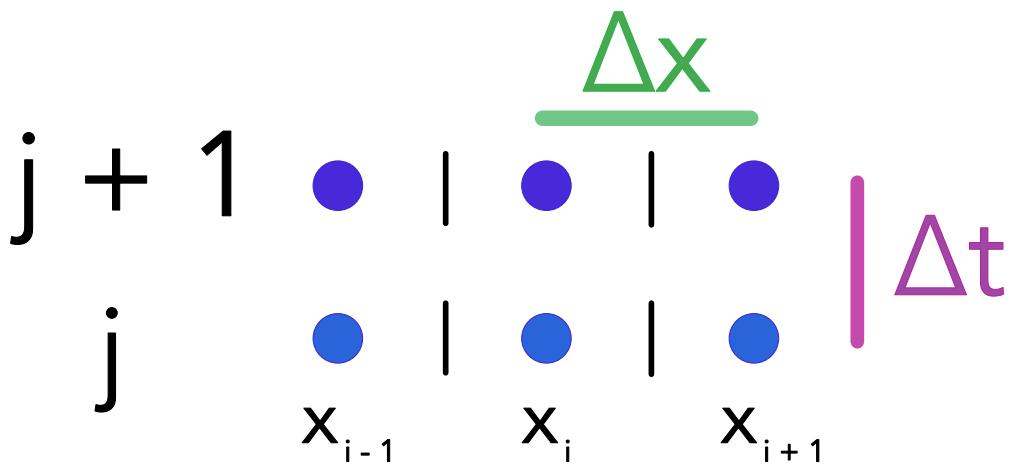


Figure 49: 1D grid

6.4 Advection - Keep information flow in the physical system in mind

Consider the first-order hyperbolic advection equation

$$\partial_t u + v \partial_x u = 0, \text{ we seek } u(x, t), \quad v \text{ constant parameter} \quad (250)$$

which is hyperbolic as of the real and *diagonizable* coefficient matrix (a scalar).

6.4.0.1 Analytic solution to the advection equation

For any function $q(x)$, the function $u(x, t) = q(x - vt)$ is a solution to the advection equation ($\partial_t u = -v \partial_x q$) (see fig. 50).

Interpreting $u(x, t = 0) = q(x)$ as an initial condition, the solution at a later time is just a shifted (by vt along x) copy of $q(x)$.

While for the advection problem we know the analytic solution, it helps us uncover the basic cavets of numerically solving PDEs.

6.4.0.2 Simple but wrong approach | we need to consider the flow of information

Let us replace the derivative in space by a central difference with respect to the neighboring grid points

$$\frac{\partial u_i}{\partial t} + v \frac{u_{i+1} - u_{i-1}}{2h} = 0 \quad (251)$$

and step in time using explicit Euler

$$u_i^{(n+1)} = u_i^{(n)} - v \frac{u_{i+1}^{(n)} - u_{i-1}^{(n)}}{2h} \Delta t \quad (252)$$

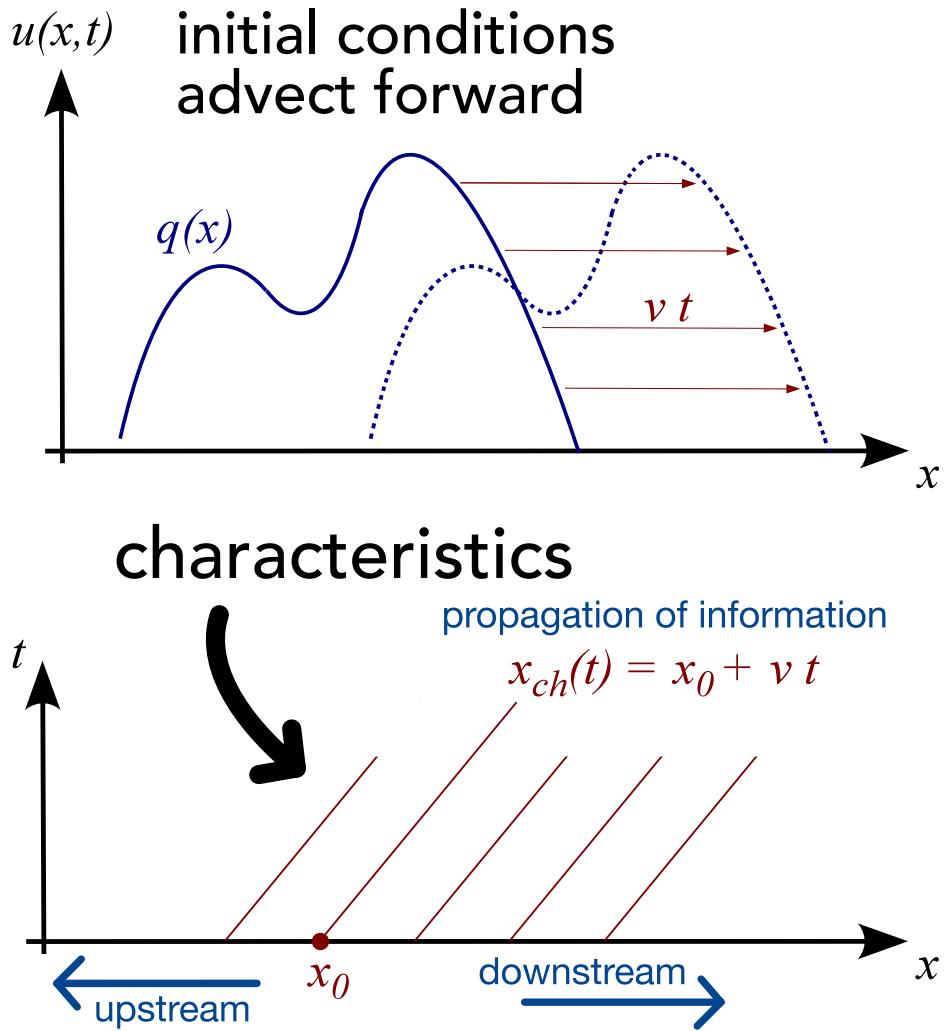


Figure 50: Advection

Problem: This is violently unstable (illustrated in 51), as (based on the characteristics) information should only travel downstream but in the central differencing, we use upstream information $u_{i+1}^{(n)}$

6.4.0.3 Directional splitting / upwind scheme to the rescue

Let us only use downstream information, so (mind the sign of v)

$$\begin{aligned} v > 0 : u_i^{(n+1)} &= u_i^{(n)} - v \frac{u_i^{(n)} - u_{i-1}^{(n)}}{h} \Delta t \\ v < 0 : u_i^{(n+1)} &= u_i^{(n)} - v \frac{u_{i+1}^{(n)} - u_i^{(n)}}{h} \Delta t \end{aligned} \quad (253)$$

An example application is given in fig. 52.

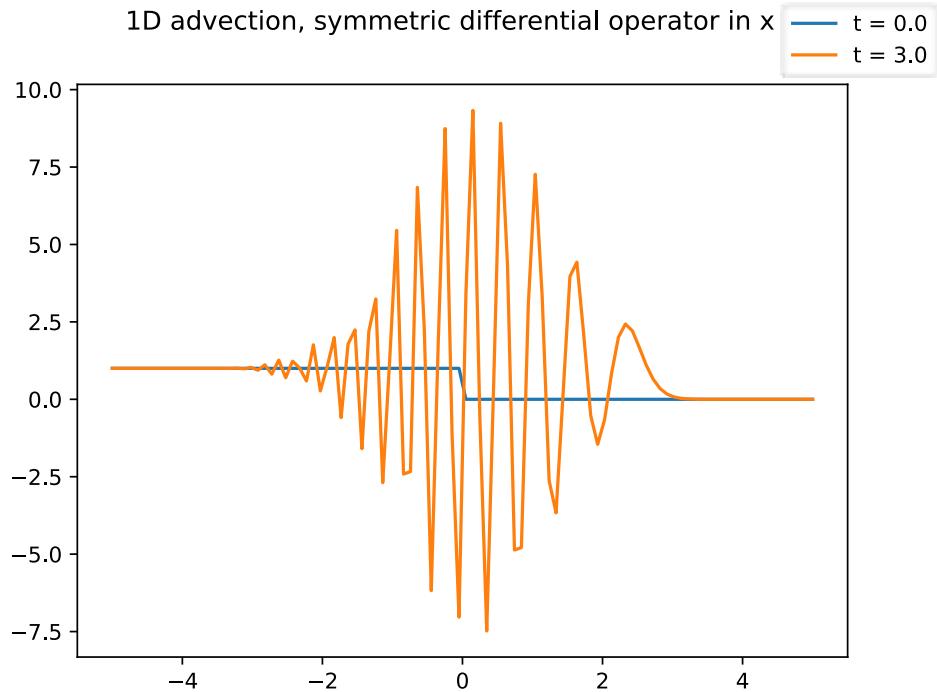


Figure 51: Advection with central differencing, violently unstable

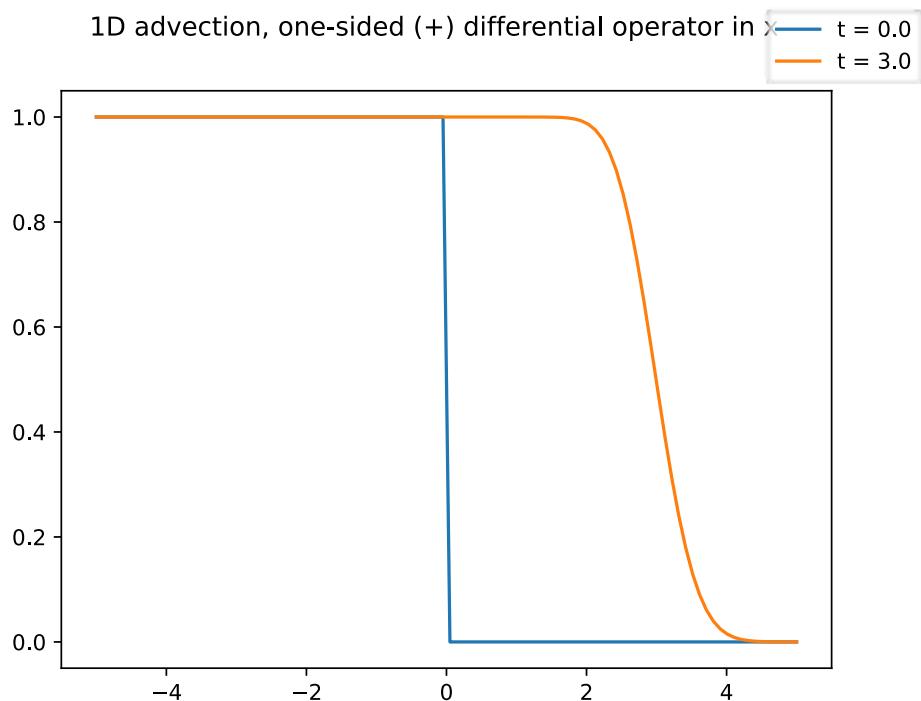


Figure 52: Advection with upwind scheme

Problem: The solution is smeared out (smoothed).

6.4.0.4 Where does the smoothing in the upwind scheme come from?

Our numerical algorithm (that lead to stability) introduced numerical diffusion as a byproduct.

Using

$$\frac{u_i - u_{i-1}}{h} = \frac{u_{i+1} - u_{i-1}}{2h} - \frac{u_{i+1} - 2u_{i+1} + u_{i-1}}{2h} \quad (254)$$

we can rewrite the upwind scheme for $v > 0$ as

$$0 = \partial_t u_i + v \frac{u_i - u_{i-1}}{h} = \partial_t u_i + v \frac{u_{i+1} - u_{i-1}}{2h} - \underbrace{\frac{vh}{2}}_D \underbrace{\frac{u_{i+1} - 2u_{i+1} + u_{i-1}}{h^2}}_{\text{discretization of } \partial_x^2 u} \quad (255)$$

$$\text{so } \partial_t u_i + v \frac{u_i - u_{i-1}}{h} = D \frac{u_{i+1} - 2u_{i+1} + u_{i-1}}{h^2}$$

which is the central difference version of a advection-diffusion equation

$$\partial_t u + v \partial_x u = D \partial_x^2 u \quad (256)$$

where the diffusion term smears out the solution.

The numerical diffusion term $D = \frac{vh}{2}$ is small for

- fine grids ($h \rightarrow 0$)
- small velocities ($v \rightarrow 0$), stronger advection \rightarrow stronger diffusion

The diffusion term dampens all post-shock oscillations / oscillations connected to steep gradient and can thus also be useful for stabilization.

6.4.0.5 What is the maximum timestep we can take? | Courant-Friedrichs-Lowy (CFL) criterion

Consider the advection problem. Information travels with velocity v . Consider you would take a timestep $\Delta t > \frac{h}{v}$. Then in the upwind scheme, we would not only need to consider u_{i-1} (assuming $v > 0$) but also u_{i-2} , which we do not do - leading to catastrophic instability, as illustrated in fig. 53.

The CFL criterion therefore reads

$$\Delta t \leq \frac{h}{v} \quad (257)$$

a necessary but not sufficient condition for the stability of explicit methods regarding hyperbolic conservation laws (here for the advection case, might generally take different

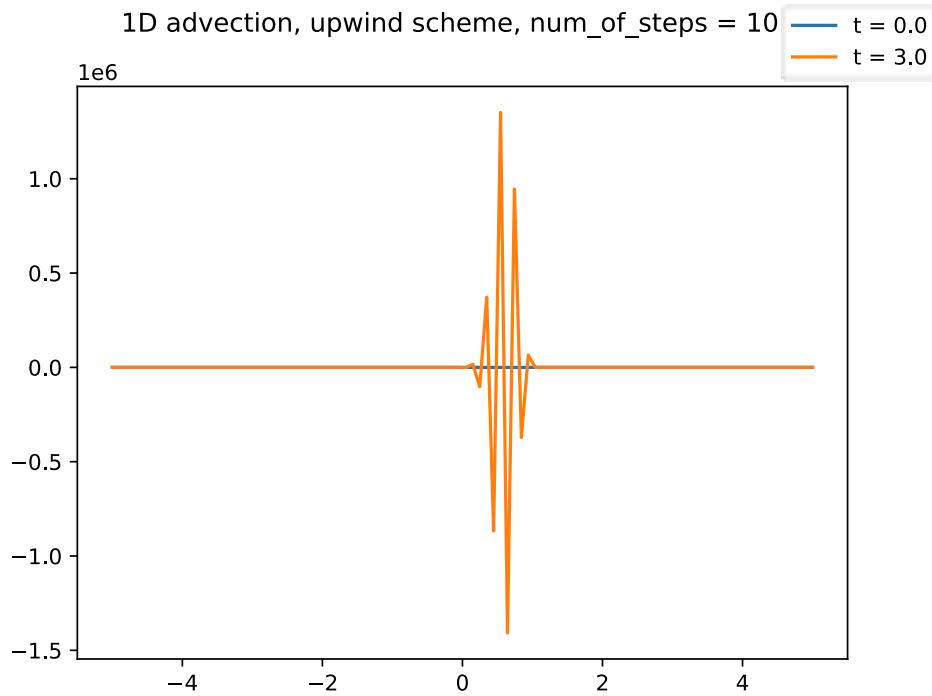


Figure 53: Advection with upwind scheme, violating the CFL criterion

forms)

Note: Integrating hyperbolic conservation laws in time needs sufficiently small integration steps, as there is a finite speed of information travel in such hyperbolic problems.

6.4.0.6 Hyperbolic conservation laws | changing upwind direction

Consider the continuity equation

$$\partial_t \rho + \nabla \cdot \underline{F} = 0, \quad \text{mass flux } \underline{F} = \rho \underline{v} \quad (258)$$

While this is essentially an advection problem, \underline{v} can vary over space, $\underline{v} = \underline{v}(x)$.

In a naive discretization of space and time

$$\frac{\rho_i^{(n+1)} - \rho_i^{(n)}}{\Delta t} + \frac{F_{i+1}^{(n)} - F_{i-1}^{(n)}}{2\Delta x} = 0 \rightarrow \rho_i^{(n+1)} = \rho_i^{(n)} + \frac{\Delta t}{2\Delta x} (F_{i+1}^{(n)} - F_{i-1}^{(n)}) \quad (259)$$

the solution is highly unstable as of not accounting for the direction of flow of information (here flow of mass). We need to choose the correct discretization depending on the direction of the characteristic / sign of mass flux.

6.4.0.7 What if identifying the local characteristics is very difficult?

In general (non-linear PDE) situations, information about the local solution and local characteristics is obtained using Riemann solvers.

6.5 Intermezzo: CFL like criterion and connection to stiffness in a reaction diffusion system

Introduction to the example problem - the Brusselator

As our example, we use a simplified *Brusselator* as introduced in Hairer, Wanner, and Nørsett, 1993, chapter I.16 and further discussed in Hairer and Wanner, 1996, chapter IV.1 (originally introduced in Lefever and Nicolis, 1971).

Some details on the Brusselator and all of our implementations regarding numerical methods for solving it can be found in the [accompanying Julia notebook](#).

Here, it is sufficient to know that we consider a non-linear chemical-reaction-diffusion partial differential equation in one dimension of the form

$$\begin{aligned}\frac{\partial u}{\partial t} &= A + u^2 v - (B + 1)u + \alpha \frac{\partial^2 u}{\partial x^2} \\ \frac{\partial v}{\partial t} &= Bu - u^2 v + \alpha \frac{\partial^2 v}{\partial x^2}\end{aligned}$$

where $u(x, t)$ and $v(x, t)$ are the concentrations of chemical substances, α is a diffusion constant and A and B are fixed concentrations of other substances.

From discretizing the differentiation in space (i.e. using the method of lines for approaching this partial differential equations) we follow (with $x_i = \frac{i}{N+1} (1 \leq i \leq N)$, $\Delta x = \frac{1}{N+1}$, $A = 1$, $B = 3$, $\alpha = \frac{1}{50}$)

$$\begin{aligned}u'_i &= 1 + u_i^2 v_i - 4u_i + \frac{\alpha}{(\Delta x)^2} (u_{i-1} - 2u_i + u_{i+1}), \\ v'_i &= 3u_i - u_i^2 v_i + \frac{\alpha}{(\Delta x)^2} (v_{i-1} - 2v_i + v_{i+1}) \\ u_0(t) &= u_{N+1}(t) = 1, \quad v_0(t) = v_{N+1}(t) = 3 \\ u_i(0) &= 1 + \sin(2\pi x_i), \quad v_i(0) = 3, \quad i = 1, \dots, N.\end{aligned}\tag{260}$$

where some boundary conditions and initial conditions have been chosen. The constant boundary values are enforced using so-called ghost-cells in the implementation.

We compactly write the differential equation system as

$$\underline{y} = \underline{f}(\underline{y}), \quad \underline{y} = \begin{pmatrix} u_0 \\ \vdots \\ u_{N+1} \\ v_0 \\ \vdots \\ v_{N+1} \end{pmatrix}$$

where $\underline{f} : \mathbb{R}^{2N} \rightarrow \mathbb{R}^{2N}$ follows from equation 260.

The occurrence of stiffness

Let us start by applying the Explicit Euler method to the simplified Brusselator with $N = 40$ grid points and a step size $dt = 0.01$. The result is shown in figure 54a and is in agreement with the literature results (Hairer and Wanner, 1996, chapter IV.1).

But if we increase N to $N = 400$, i.e. we decrease the spacing between the grid points $\Delta x = \frac{1}{N+1}$, the Explicit Euler scheme yields a diverging result (see figure 54b).

We can get back to a stable solution by decreasing the step size but notice that we have to use a much smaller step size, e.g. $dt = 0.0001$ (see figure 54c), in spite of the solution still being very smooth.

In fact even a more sophisticated explicit method like the Tsitouras 5/4 Runge-Kutta method from the DifferentialEquations.jl package (Rackauckas and Nie, 2017) will use excessively many steps (e.g. to cover a time interval of length 10 (dimensionless as of our problem formulation) using the default settings 220047 evaluations of \underline{f} are used). The problem of stiffness as described in section 3.4.2 has occurred.

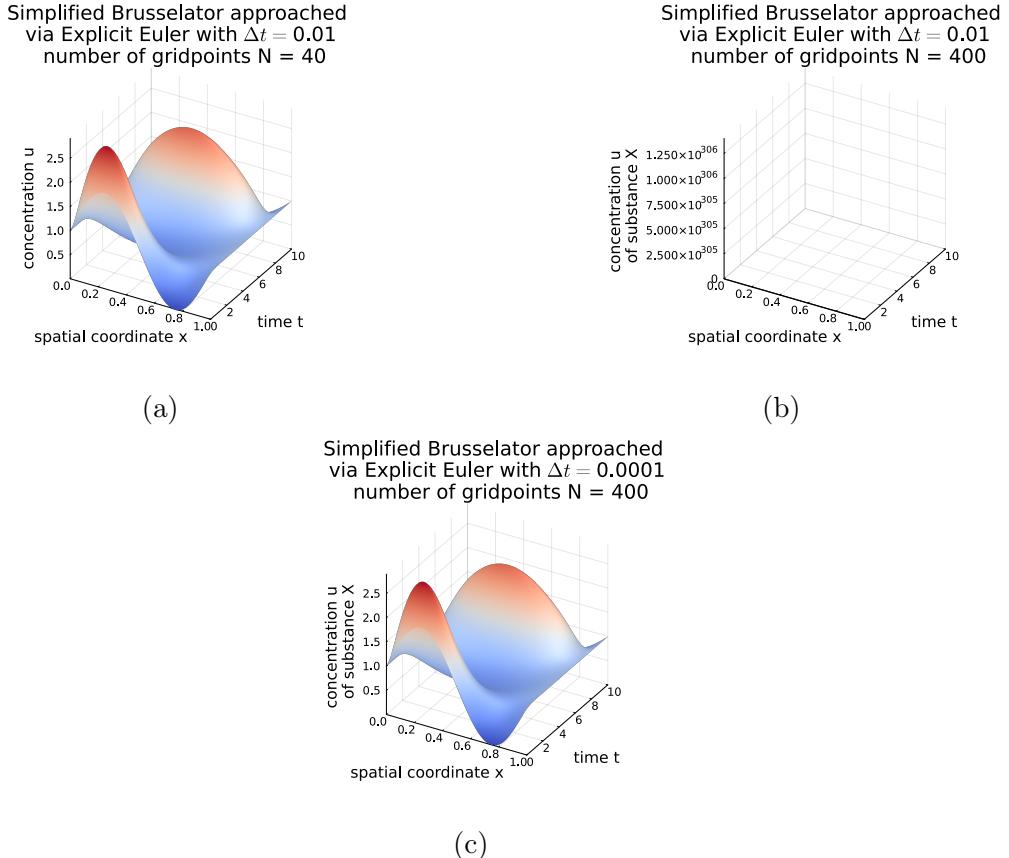


Figure 54: Numerical solutions to a simplified Brusselator using the Explicit Euler method with different numbers N of grid points and time-steps Δt .

Understanding stiffness in a diffusive context

The transport process at play is diffusion. For a diffusive process we know the spreading of some concentration to follow $\sigma = \sqrt{2\alpha t}$. Now in each Euler step we do, only neighboring cells have an effect on each other (compare the discretized ODE we introduced in the beginning). Therefore - in the style of a Courant-Friedrichs-Levy criterion (Courant et al., 1928) - we can propose the stability constraint

$$\Delta x > \sigma(\Delta t) = \sqrt{2\alpha t}$$

so $\Delta t < \frac{\Delta x^2}{2\alpha}$. If we want to double N (cut in half Δx) we need $\mathcal{O}(N^2)$ more time-steps with the complexity of a function evaluation scaling with $\mathcal{O}(N)$ resulting in a $\mathcal{O}(N^3)$ scaling - calculations quickly become unfeasible.

Let us note that in the simplified Brusselator at hand it is the diffusive term causing stiffness, but in a more complex model the chemical reactions could be an additional factor of stiffness (see e.g. Chou et al., 2007).

6.6 Riemann problem | Riemann solvers

Consider a hyperbolic system. At time $t = 0$ we start out with two piecewise constant states (in the fluid variables) meeting at a plane. The Riemann problem is to determine the subsequent evolution.

Note: One reason the Riemann problem is important is that when we discretize a fluid into cells with constant values, we effectively have Riemann problems in-between.

Consider the Riemann problem for the Euler equations (ideal gas dynamics). The two constant states can be uniquely described by

$$\underline{U}_L = \begin{pmatrix} \rho_L \\ P_L \\ \underline{v}_L \end{pmatrix}, \quad \underline{U}_R = \begin{pmatrix} \rho_R \\ P_R \\ \underline{v}_R \end{pmatrix}, \quad \text{mass density } \rho, \quad \text{pressure } P, \quad \text{velocity } \underline{v} \quad (261)$$

alternative primitive variables: density, momentum density, energy density

with a hydrodynamic discontinuity in between (illustrated in 55). This can be solved analytically (but not be written down explicitly, requires numerical root-finding for an implicit equation).

The shock tube (for $\underline{v}_L = \underline{v}_R = 0$) is a common test for Riemann solvers. There is no smooth information transport across the shock (but many collisions) and hydrodynamics breaks down.

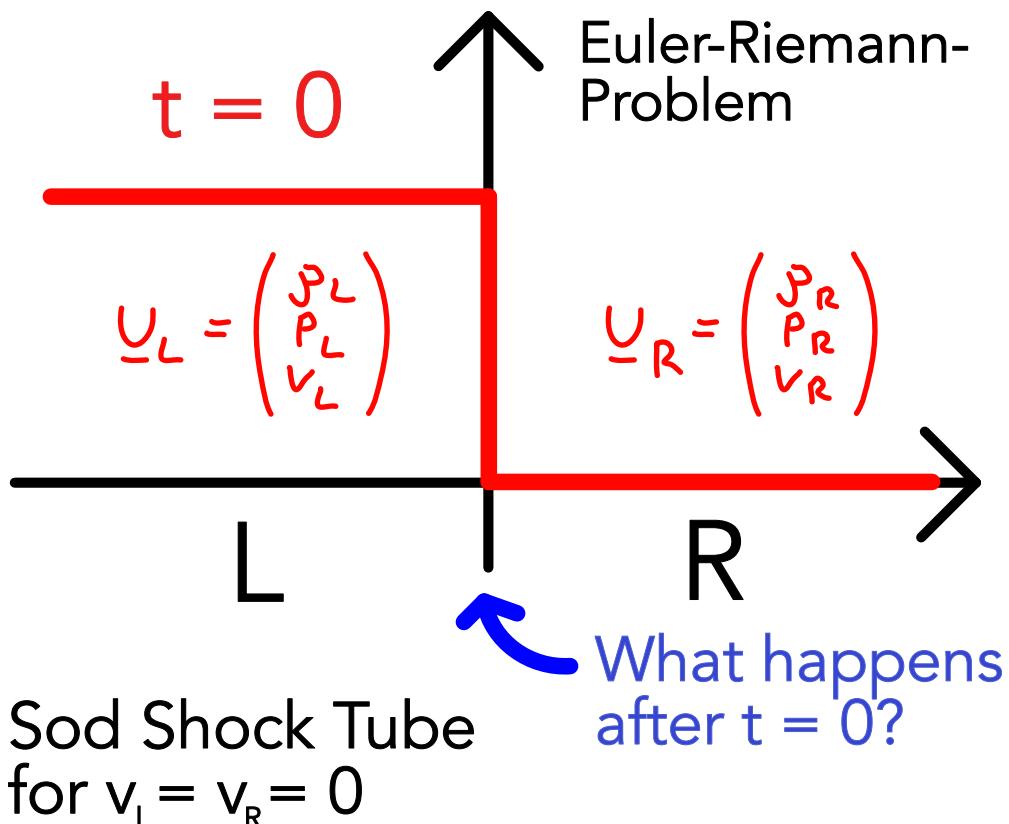


Figure 55: Riemann problem

6.6.1 Structure of the solution of the Euler-Riemann-Problem

In general, the solution to an Euler-Riemann problem always contains three waves

- **contact wave / discontinuity**: a middle wave marking the boundary between the original fluid phases
 - on either side of the contact wave there can either be a **shock** or **rarefaction wave** (rather rarefaction fan with continuously changing variables). Shock or rarefaction on both sides is also possible.

where all three waves propagate with constant speed. At $x = 0$ the fluid quantities (ρ, P, v) (in the region containing the interface) are constant in time for $t > 0$.

6.6.1.1 Characteristics of the three waves

The characteristics are shown in fig. 56.

6.6.1.2 Example Riemann-Problem situation

An example with marked rarefaction fan, contact discontinuity and shock is shown in fig. 57.

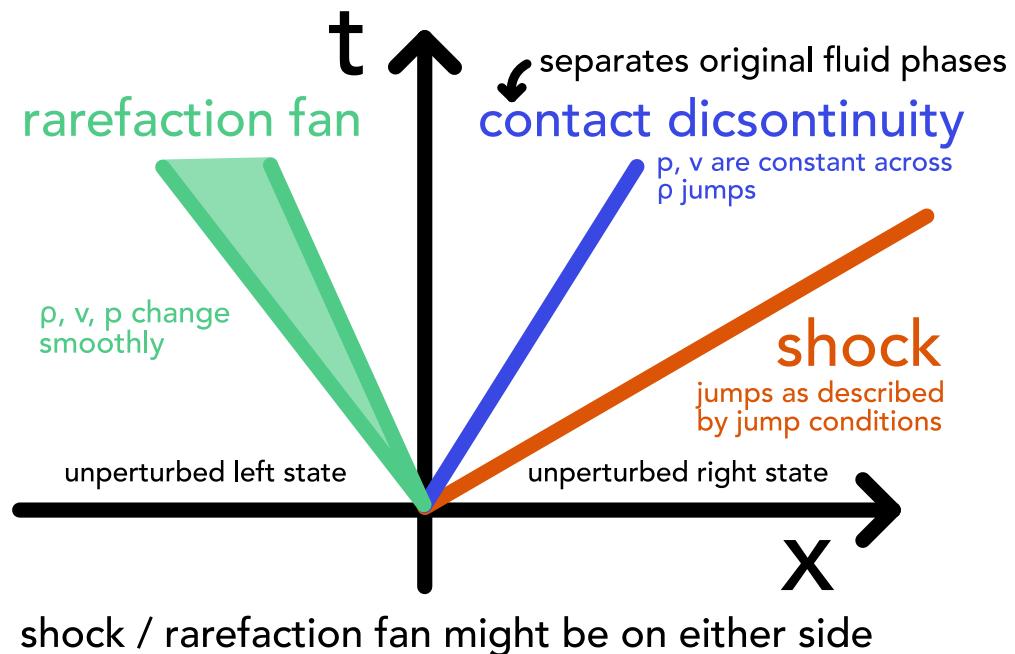


Figure 56: Characteristics of the three waves

6.6.1.3 Properties of shock, contact discontinuity and rarefaction wave

- Shock: Normal velocity, pressure, density, entropy change discontinuously. In the rest frame of the shock fast upstream fluid is converted to slow downstream one. The fluid is compressed and kinetic energy turned into heat (addition of entropy).
- Contact discontinuity: Traces the original separating plane between the two originally separated fluid phases.
 - constant across the contact: pressure, normal velocity
 - can jump: density, entropy, temperature
- Rarefaction wave: smooth transition between two states, no discontinuities

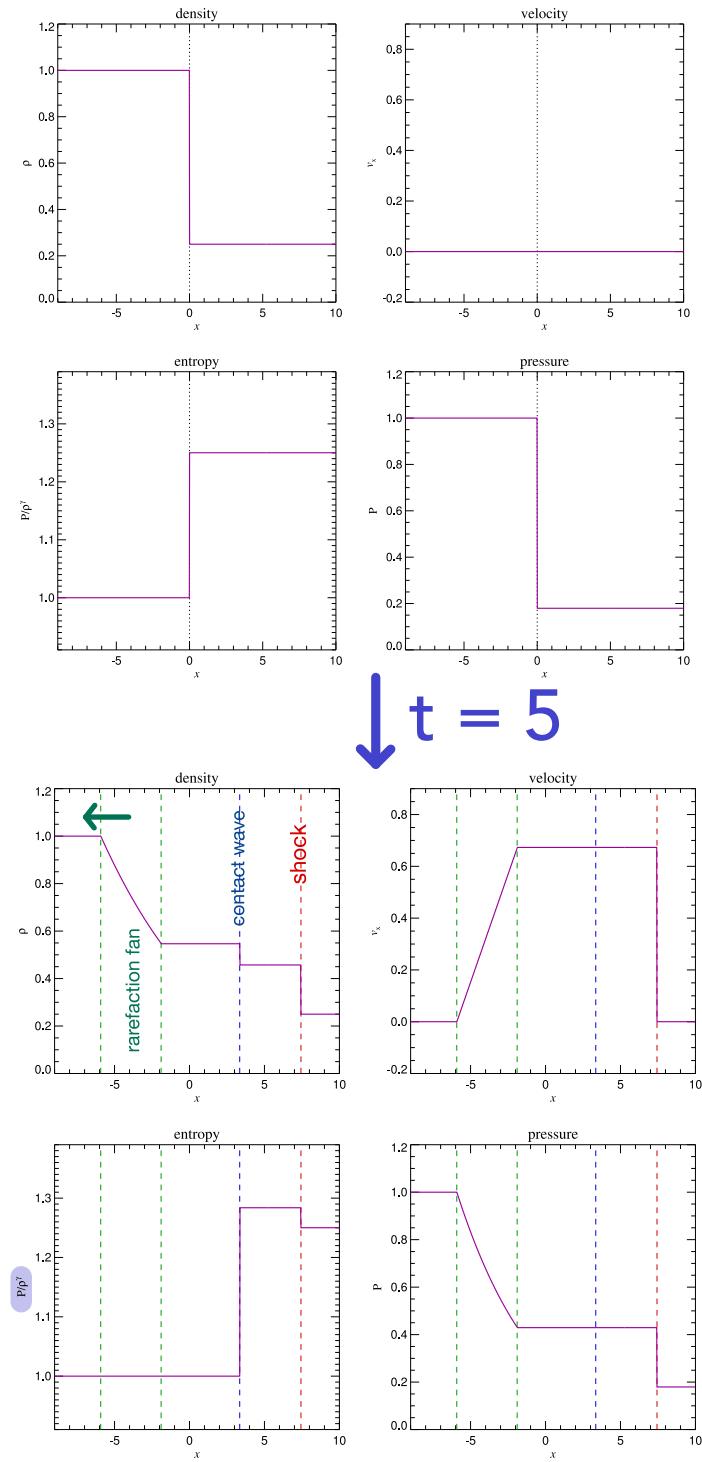


Figure 57: Example Riemann-Problem situation

6.7 Finite volume discretization | Reducing a hyperbolic conservation law to a Riemann problem | Godunov scheme

Idea: If we discretize the fluid into finite volumes and assume constant fluid variables on them, at each cell interface we have a Riemann problem. We are now more concerned with cell boundaries than centers. In the - conservative, finite volume - Godunov method, exact or approximate Riemann problems are solved at the boundaries and no flux is lost.

6.7.1 Problem | solve a hyperbolic conservation law PDE

We want to solve the conservation law

$$\partial_t \underline{U} + \partial_{\underline{x}} \cdot \underline{\underline{F}}(\underline{U}) = 0, \quad \text{state vector } \underline{U}, \quad \text{flux matrix } \underline{\underline{F}} \quad (262)$$

for instance for the Euler equations with

$$\underline{U} = \begin{pmatrix} \rho \\ \rho v \\ \rho e \end{pmatrix}, \quad \underline{\underline{F}} = \begin{pmatrix} \rho v \\ \rho v v^T + P \mathbf{1} \\ (\rho e + P) v \end{pmatrix}, \quad e = e_{th} + \frac{v^2}{2}, \quad P = (\gamma - 1)\rho e_{th} \text{ closure} \quad (263)$$

6.7.2 Deriving a finite volume scheme where only Riemann problems are left to solve

In a finite volume method, the state of the cell is an average over the fluid quantities over the cell

$$\underline{U}_i = \frac{1}{V_i} \int_{\text{cell } i} \underline{U}(\underline{x}) dV \quad (264)$$

Aim: We want to derive an update scheme for the cell averages \underline{U}_i (the vector of the fluid variables), where no intercell flux is lost.

We derive the update scheme in 1D, so $\underline{\underline{F}}$ is a vector ($\underline{v}\underline{v}^T$ is a scalar) (see figure 58).

Step 1: Integrate the conservation law over a cell and timestep and recognize the average defined in 264.

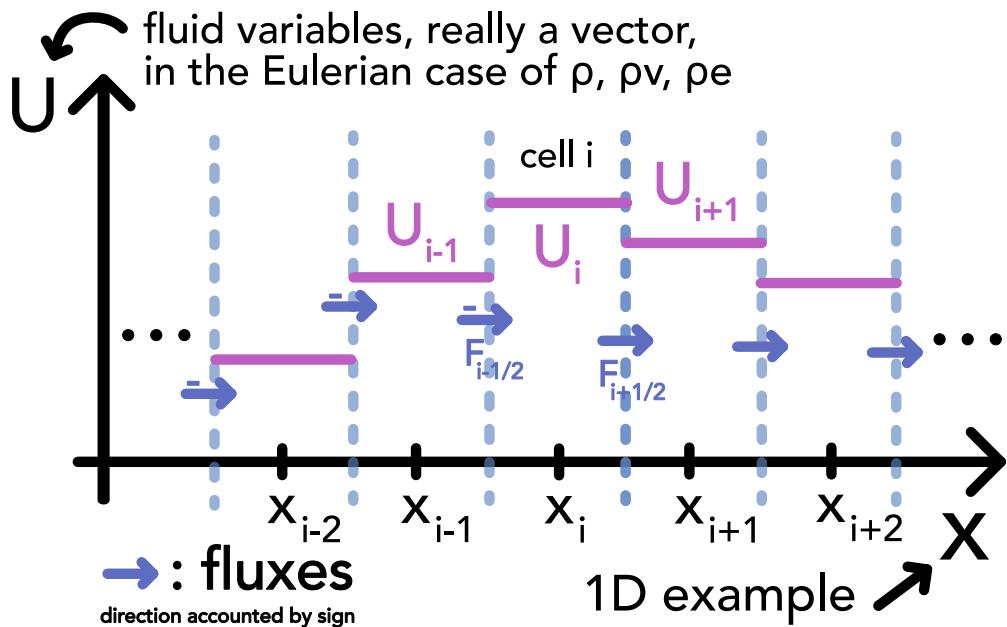


Figure 58: Finite volume scheme

$$\begin{aligned}
 & \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \int_{t_n}^{t_{n+1}} \left(\frac{\partial \underline{U}}{\partial t} + \frac{\partial \underline{F}}{\partial x} \right) dt dx = 0 \\
 & \underbrace{\quad}_{\text{carry out simple integrals}} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} [\underline{U}(x, t_{n+1}) - \underline{U}(x, t_n)] dx + \int_{t_n}^{t_{n+1}} [\underline{F}(x_{i+\frac{1}{2}}, t) - \underline{F}(x_{i-\frac{1}{2}}, t)] dt \\
 & \underbrace{\quad}_{\text{recognize avg}} \Delta x \left[\underline{U}_i^{(n+1)} - \underline{U}_i^{(n)} \right] + \int_{t_n}^{t_{n+1}} [\underline{F}(x_{i+\frac{1}{2}}, t) - \underline{F}(x_{i-\frac{1}{2}}, t)] dt = 0
 \end{aligned} \tag{265}$$

Step 2: In the frame of discrete cell averages, between any two cells we essentially have a Riemann problem from which solution we can follow the flux between the cells.

$\underline{F}(x_{i+\frac{1}{2}}, t)$ for $t > t_n$ = solution of Riemann problem with left state $\underline{U}_i^{(n)}$ and right state $\underline{U}_{i+1}^{(n)}$

Note: At the cell interface, the solution of the Riemann problem is constant in time, so

$$\underline{F}(x_{i+\frac{1}{2}}, t) = \underline{F}_{i+\frac{1}{2}}^* = \underline{F}_{\text{Riemann}}(\underline{U}_i^{(n)}, \underline{U}_{i+1}^{(n)}) \text{ from Riemann solution at interface} \tag{267}$$

Step 3: As the solution at the interface is constant, without approximation, we can

write the Godunov scheme (based on eq. 265) as

$$\underline{U}_i^{(n+1)} = \underline{U}_i^{(n)} + \frac{\Delta t}{\Delta x} \left[\underbrace{\underline{F}_{i-\frac{1}{2}}^*}_{\text{flux from left into cell}} - \underbrace{\underline{F}_{i+\frac{1}{2}}^*}_{\text{out on the right}} \right] \quad (268)$$

This defined an update scheme for the fluid variables on the cells (with some appropriate initial and boundary conditions). $\underline{F}_{i-\frac{1}{2}}^*$ and $\underline{F}_{i+\frac{1}{2}}^*$ are black-boxes for now - to find the fluxes we need to solve the Riemann problem, later e.g. done with the approximate HLL solver.

6.7.2.1 Caveats of the Godunov scheme

CFL needs to be obeyed: We can only assume the Riemann problems at the interfaces to be independent, if the timestep is short enough, so that no information has travelled from one interface to the other \rightarrow CFL: $\frac{\Delta x}{\Delta t} \leq c_{max}$.

U is not piecewise constant in reality: Even if we start out with piecewise constant \underline{U} in reality, this will change. Therefore the flux we calculate between the cells is also only an approximation.

6.7.3 Godunov's method and Riemann solver | reconstruct - evolve - average (REA)

Godunov's method can be seen as a REA scheme of a hydrodynamical system discretized on a mesh

1. Reconstruct: A global solution is constructed from the cell averaged quantities, simples approach: piecewise-constant
2. Evolve: The reconstructed state is evolved by Δt (mind the CFL criterion), in the Godunov scheme based on the intercell Riemann problems
3. Average: $\underline{U}_i^{(n+1)}$ is calculatted from the evolved state, in the Godunov scheme, evolving and averaging are combined as we directly calculate the new average based on accounting the fluxes entering and leaving the cell (an implicit average over the new state)

But how can the Riemann problem giving us the fluxes be solved?

6.8 Approximate Riemann solvers | HLL solver

6.8.1 1D Riemann problem to solve

Let us again formulate the Riemann problem in 1D. Given the conservation law

$$\partial_t u(x) + \partial_x f(u) = 0 \quad (269)$$

with piecewise initial values

$$u(x, t=0) = \begin{cases} u_L & \text{for } x < 0 \\ u_R & \text{for } x \geq 0 \end{cases}, \quad \text{cell interface at } x = 0 \quad (270)$$

we want to solve for our fluid quantity u . The characteristics, i.e. where information from point $x = 0$ (discontinuity) can travel in some time, are shown in figure 59.

$\vec{f}_L = \vec{f}_R = \vec{f}$
aim: find flux f^* of conserved quantity u
 $\partial_x u + \partial_t f(u) = 0$ through boundary at $x = 0$

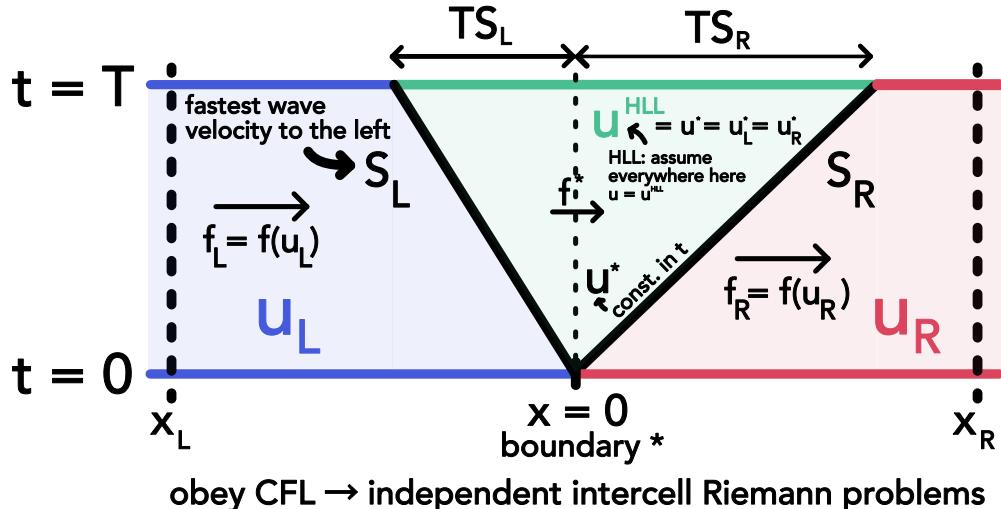


Figure 59: Characteristics of the Riemann problem and HLL approach.

6.8.2 Basic HLL assumptions and problem statement

In the HLL scheme we assume to know

- the fastest moving wave velocity to the left S_L and right S_R
- the left and right state u_L and u_R and therefore the fluxes f_L and f_R

Based on the CFL criterion (no interaction with other cells), we can assume the following quantities to be constant in time $t \in [0, T]$:

$$u_L = u(x_L, t), \quad u_R = u(x_R, t), \quad f_L = f(u_L), \quad f_R = f(u_R) \quad (271)$$

and we make the simplifying assumption that

$$\forall x \in [S_R t, S_L t] : u(x, t) = u^{HLL} = u_L^* = u_R^* = u^* \underset{\text{at interface}}{\underset{\curvearrowleft}{=}} \text{const.}, \quad f^{HLL} = f^* = f_L^* = f_R^* \quad (272)$$

Aim: Find expressions for u^* and f^* .

6.8.3 Deriving the solution of the Riemann problem in the HLL scheme

Idea: u is conserved, so the spatial integral over u at some point in time is the same as at another point in time plus / minus the in- and outcoming fluxes during that time-interval. Based and the flux balance in our whole region $[x_L, x_R]$ we can first find an expression for u^{HLL} and then based on the same balance in the left and right region $[S_R T, 0]$ and $[0, S_L T]$ we can find an expression for f^{HLL} .

6.8.3.1 Derivation of the middle state u^{HLL} at $t = T$

Remember, we assume the middle state u^{HLL} hold for the whole interval $[S_R T, S_L T]$, where information could have spread.

We can therefore write

$$u^{HLL} = \frac{1}{T \cdot (S_R - S_L)} \int_{TS_R}^{TS_L} u(x, T) dx \quad (273)$$

we can find this integral from considering the spatial integral over the whole domain at time T

$$\begin{aligned} \int_{x_L}^{x_R} u(x, T) dx &= (TS_L - x_L) u_L + \int_{TS_R}^{TS_L} u(x, T) dx + (x_R - TS_R) u_R \\ &\underset{u \text{ conserved}}{=} \int_{x_L}^{x_R} u(x, 0) dx + \underbrace{\int_0^T f(u_L) dt}_{\text{flux in from left}} - \underbrace{\int_0^T f(u_R) dt}_{\text{flux out to right}} \quad (274) \\ &= u_R x_R - u_L x_L + f_L T - f_R T \end{aligned}$$

We therefore find an expression for our integral and thus for u^{HLL} :

$$u^{HLL} = \frac{S_R u_R - S_L u_L + f_L - f_R}{S_R - S_L} \quad (275)$$

Which is a pretty simple balance we could have seen directly.

6.8.3.2 Deriving the intercell flux $f^{HLL} = f^*$

We consider the integral over u in the left region where the information can have propagated to, so $x \in [S_R T, 0]$ and in the right region $x \in [0, S_L T]$.

Idea: As before the integral of u over space at some time t must be the same as at an earlier time plus / minus the fluxes in / out since then.

For the left ($x \in [S_R T, 0]$) we have

$$\begin{aligned} -T S_L u^{HLL} &= \int_{TS_L}^0 u(x, T) dx \\ &= \underbrace{-T S_L u_L}_{\text{at } t=0} + \underbrace{T \cdot (f_L - f_L^*)}_{\text{fluxes since then}} \end{aligned} \quad (276)$$

For the right ($x \in [0, S_L T]$) we have

$$\begin{aligned} T S_R u^{HLL} &= \int_0^{TS_R} u(x, T) dx \\ &= \underbrace{T S_R u_R}_{\text{at } t=0} + \underbrace{T \cdot (f_R^* - f_R)}_{\text{fluxes since then}} \end{aligned} \quad (277)$$

The fluxes at the interface must be equal, $f_L^* = f_R^* = f^{HLL}$, so

$$f^{HLL} = f_L + S_L (u^{HLL} - u_L) = f_R + S_R (u^{HLL} - u_R) \quad (278)$$

6.8.4 Final HLL solution

There are three states after the step T in the HLL scheme

$$u(x, T) = \begin{cases} u_L & \text{for } x < S_L t \\ u^{HLL} & \text{for } S_L t \leq x \leq S_R t \\ u_R & \text{for } x > S_R t \end{cases} \quad (279)$$

and the interface flux is

$$f^{HLL} = \frac{S_R f_R - S_L f_L + S_L S_R (u_R - u_L)}{S_R - S_L}$$

maximum velocity to the left S_L , maximum velocity to the right S_R (280)

initial state on the left u_L , initial state on the right u_R

$$f_L = f(u_L), \quad f_R = f(u_R)$$

6.8.5 Mind that the extreme velocities can point into the same direction

Both extreme velocities can be to the left, both to the right, or on to the left and one to the right. For instance in figure 60 characteristics for advection are drawn.

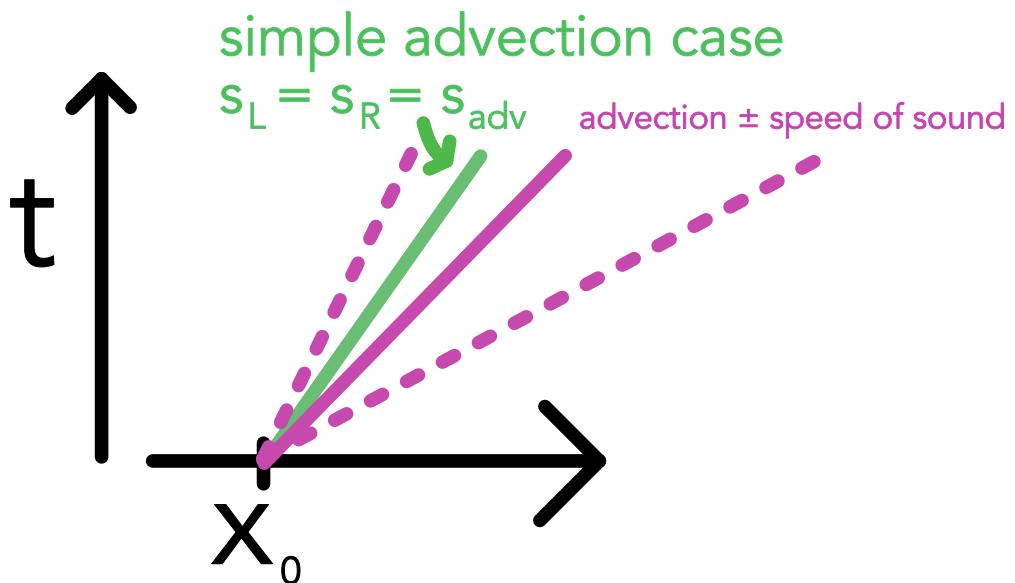


Figure 60: Advection characteristics, once the simple advection case with only the advection speed and once with information travelling from the advected state to the left and right with the speed of sound.

6.8.6 Godunov scheme with HLL solver

For the Godunov scheme

$$U_i^{(n+1)} = U_i^{(n)} + \frac{\Delta t}{\Delta x} \left[F_{i-\frac{1}{2}}^* - F_{i+\frac{1}{2}}^* \right] \quad (281)$$

the flux we choose depends on the orientation of S_L and S_R (developing the *Lichtkegel*). This is illustrated for unidirectional information flow to the right in figure 61.

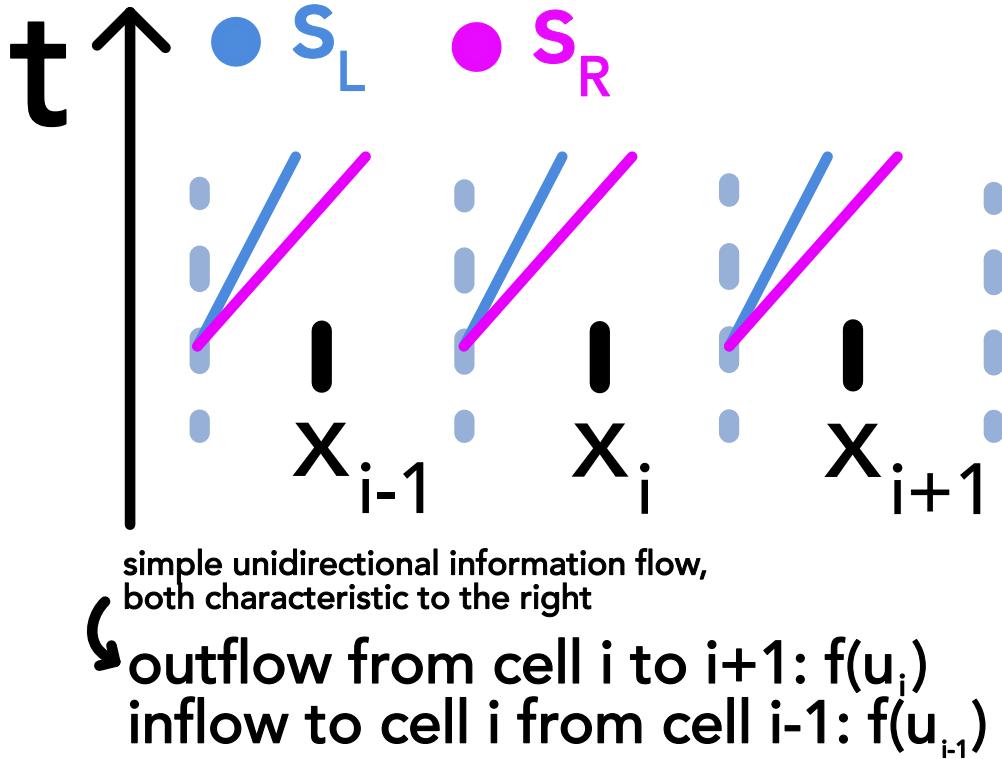


Figure 61: Godunov scheme with HLL solver for unidirectional information flow to the right.

We get

$$F_{i+\frac{1}{2}}^* = \begin{cases} F_i & \text{for } 0 < S_L \\ F_{i+\frac{1}{2}}^{HLL} & \text{for } S_L \leq 0 \leq S_R , \quad F_{i-\frac{1}{2}}^* \text{ from } i \rightarrow i-1 \\ F_{i+1} & \text{for } 0 > S_R \end{cases} \quad (282)$$

with

$$F_{i+\frac{1}{2}}^{HLL} = \frac{S_R F_{i+1} - S_L F_i + S_L S_R (U_{i+1} - U_i)}{S_R - S_L} \quad (283)$$

where we can combine this expression and the cases above to

$$F_{i+\frac{1}{2}}^{HLL} = \frac{S_R^+ F_{i+1} - S_L^- F_i + S_L^- S_R^+ (U_{i+1} - U_i)}{S_R^+ - S_L^-}, \quad S_R^+ = \max(0, S_R), \quad S_L^- = \min(0, S_L) \quad (284)$$

6.8.7 Pointers to extensions of the HLL scheme

- in HLLC an additional velocity between S_L and S_R is considered
- HLLD used for magnetohydrodynamics (MHD)

6.8.8 Ansätze for the maximum wave velocities S_L and S_R

Consider the gas velocity v as given by part of the state vector and sound speed c_s (given by the problem, here denoted a_L and a_R), then possible estimates are

- $S_L = v_L - a_L, S_R = v_R + a_R$
- $S_L = \min(v_L - a_L, v_R - a_R), S_R = \max(v_R + a_R, v_R + a_R)$
- Roe average, where we weigh dense areas as more important to the communication (leading to less smearing but **instability**) of information.

$$\begin{aligned} S_L &= \tilde{u} - \tilde{a} \\ S_R &= \tilde{u} + \tilde{a} \\ \tilde{u} &= \frac{\sqrt{\rho_L} u_L + \sqrt{\rho_R} u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\ \tilde{a} &= \left[(\gamma - 1) \left(\tilde{H} - \frac{1}{2} \tilde{u}^2 \right) \right]^{\frac{1}{2}} \text{ with the enthalpy} \\ H &= (e + P)/\rho \text{ and} \\ \tilde{H} &= \frac{\sqrt{\rho_L} H_L + \sqrt{\rho_R} H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \end{aligned} \tag{285}$$

6.9 Extension of Eulerian hydrodynamics to multiple dimensions

Based on our previous results, we can simulate the 1D conservation law

$$\partial_t \underline{U} + \partial_x \underline{F}(\underline{U}) = 0 \tag{286}$$

e.g. for an isothermal gas (so constant sound speed c_s) with

$$\underline{U} = \begin{pmatrix} \rho \\ \rho v_x \end{pmatrix}, \quad \underline{F} = \begin{pmatrix} \rho v_x \\ \rho v_x^2 + P \end{pmatrix}, \quad P = c_s^2 \rho \tag{287}$$

Let us formulate the Euler equations for a 3D fluid (see eq. 263) by separating the flux by direction

$$\partial_t \underline{U} + \partial_x \underline{F} + \partial_y \underline{G} + \partial_z \underline{H} = 0 \tag{288}$$

with

\underline{F} : flux vector along \hat{e}_x , \underline{G} : flux vector along \hat{e}_y , \underline{H} : flux vector along \hat{e}_z

$$\underline{F} = \begin{pmatrix} \rho u \\ \rho u^2 + P \\ \rho u v \\ \rho u w \\ u(\rho e + P) \end{pmatrix}, \quad \underline{G} = \begin{pmatrix} \rho v \\ p u v \\ \rho v^2 + P \\ \rho v w \\ v(\rho e + P) \end{pmatrix}, \quad \underline{U} = \begin{pmatrix} \rho w \\ p u w \\ \rho v w \\ \rho w^2 + P \\ w(\rho e + P) \end{pmatrix} \quad (289)$$

state vector $\underline{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{pmatrix}$

and

$$\begin{aligned} \text{total specific energy per uni mass } e &= e_{th} + \frac{1}{2} (u^2 + v^2 + w^2) \\ \text{pressure } P &= (\gamma - 1)\rho e_{th}, \quad \text{thermal energy per unit mass } e_{th} \\ u &: \text{velocity in } \hat{e}_x \text{ direction, } v : \text{velocity in } \hat{e}_y \text{ direction, } w : \text{velocity in } \hat{e}_z \text{ direction} \end{aligned} \quad (290)$$

6.9.1 Dimensional splitting Ansatz

Idea: Separately update dimensions (using our 1D solver) and combine.

From eq. 288 we make the following separation ansatz

$$\partial_t \underline{U} + \partial_x \underline{F} = 0, \quad \partial_t \underline{U} + \partial_y \underline{G} = 0, \quad \partial_t \underline{U} + \partial_z \underline{H} = 0 \quad (291)$$

Note: While the state vector and fluxes are still $\in \mathbb{R}^5$ (the velocities in the other directions appear), we effectively have *augmented* 1D problems (the flux along x is not as directly coupled to w as it is to u)

To forward our state in 3D we have to sequence multiple augmented 1D steps (sweeps). For 2D this is illustrated in figure 62.

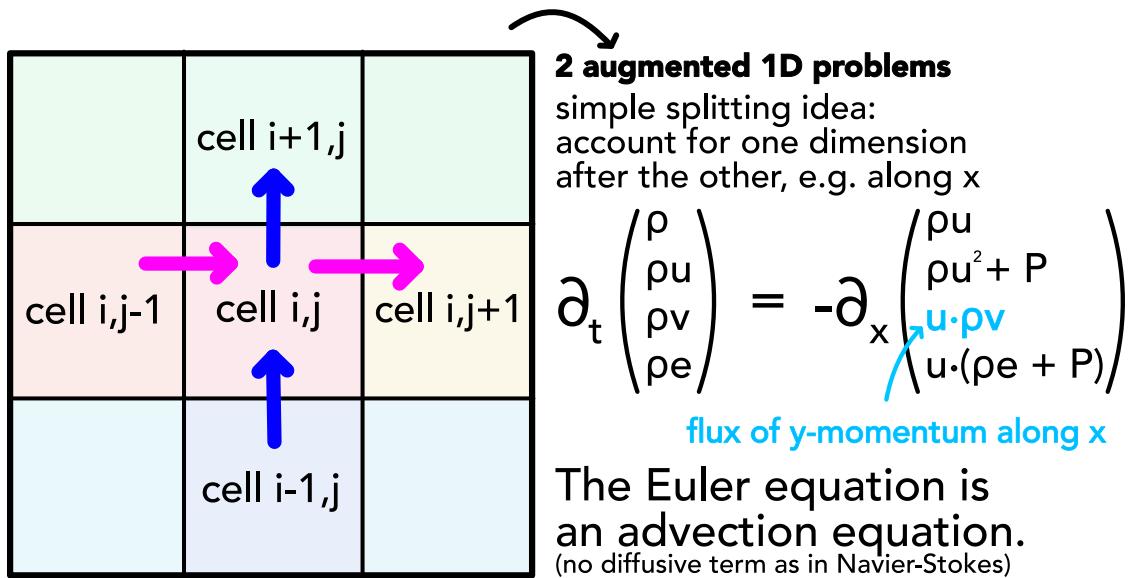


Figure 62: Splitting Ansatz for 2D

6.9.1.1 1st order ansatz

Assuming we already have a method to advance in one dimension then

$$\underline{U}^{(n+1)} = \mathcal{X}(\Delta t) \mathcal{Y}(\Delta t) \mathcal{Z}(\Delta t) \underline{U}^{(n)}, \quad \text{time evolution operators } \mathcal{X}, \mathcal{Y}, \mathcal{Z} \quad (292)$$

is a dimensionally split update scheme, which is exact for linear advection but not so for any higher order problem (first order reduction) as the steps in the dimensions are done separately.

6.9.1.2 2nd order accurate in 2D examples

$$\begin{aligned} \underline{U}^{(n+1)} &= \frac{1}{2} [\mathcal{X}(\Delta t) \mathcal{Y}(\Delta t) + \mathcal{Y}(\Delta t) \mathcal{X}(\Delta t)] \underline{U}^{(n)} \\ \text{or } \underline{U}^{(n+1)} &= X\left(\frac{\Delta t}{2}\right) \mathcal{Y}(\Delta t) \mathcal{X}\left(\frac{\Delta t}{2}\right) \underline{U}^{(n)} \end{aligned} \quad (293)$$

6.9.2 2nd order accurate in 3D example

$$\underline{U}^{(n+1)} = x\left(\frac{\Delta t}{2}\right) \mathcal{Y}\left(\frac{\Delta t}{2}\right) z(\Delta t) \mathcal{Y}\left(\frac{\Delta t}{2}\right) \mathcal{X}\left(\frac{\Delta t}{2}\right) \underline{U}^{(n)} \quad (294)$$

where the 2nd order is based on the alternating reverse order application of the time evolution operators.

6.9.3 Unsplit schemes

Consider rectangular 2D cells. In a dimensionally split scheme we would make updates to a cell based on the information flow in only one direction and then the other based on the changed situation. In an unsplit scheme we apply both fluxes simultaneously.

In the case of rectangular cells in 2D, we have

$$\underline{U}_{i,j}^{(n+1)} = \underline{U}_{i,j}^{(n)} + \frac{\Delta t}{\Delta x} \left(\underline{F}_{i-\frac{1}{2},j} - \underline{F}_{i+\frac{1}{2},j} \right) + \frac{\Delta t}{\Delta y} \left(\underline{G}_{i,j-\frac{1}{2}} - \underline{G}_{i,j+\frac{1}{2}} \right) \quad (295)$$

In the situation of an unstructured mesh, we generally have

$$\underline{U}^{(n+1)} = \underline{U}^{(n)} - \frac{\Delta t}{V} \int \underline{F} \cdot d\underline{S} \text{ integral over cell surface, } d\underline{S} \text{ is the surface element vector, pointing outward} \quad (296)$$

which makes sense intuitively (the change coming from the borders distributes over the cells).

The situations are illustrated in figure 63.

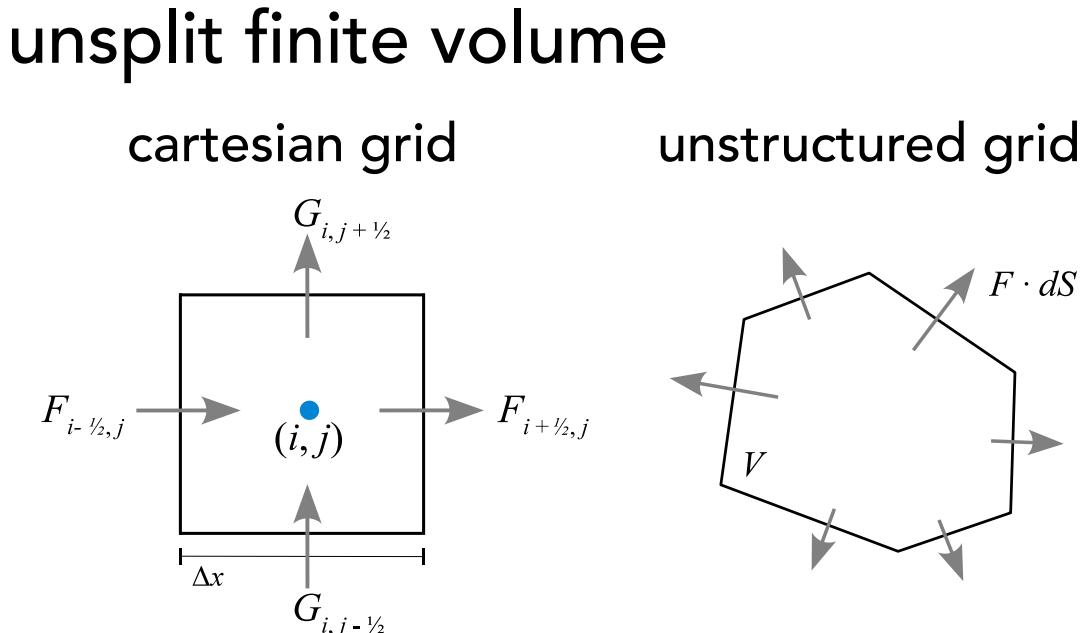


Figure 63: Unsplitting Ansatz Grids

6.10 Extensions for high-order accuracy

6.10.1 What even is a schemes order?

Consider our numerical solution ρ_i sits on a grid of N points x_i , $i = 1, \dots, N$. Let $\rho(x)$ be the true solution. Then based on the mean L1 error

$$L1 = \frac{1}{N} \sum_{i=1}^N |\rho_i - \rho(x_i)| \quad (297)$$

we call a method

- first order accurate if $L1 \propto \Delta x \propto N^{-1}$ with $\Delta x = \frac{L}{N}$
- second order accurate if $L1 \propto \Delta x^2 \propto N^{-1}$
- ...

In the 2nd order accurate scheme, doubling the number of cells will quarter our error.

6.10.2 2nd order extension to Godunov's scheme by changing the reconstruction step from piecewise-constant to linear

Godunov's theorem states: Linear numerical schemes for solving partial differential equations (PDE's), having the property of not generating new extrema (monotone scheme), can be at most first-order accurate^a. Bram van Leer (and Vladimir P. Kolgan^b) first succeeded in **circumventing this** by nonlinearly limiting the second order term as a function of the non-smoothness of the numerical solution (sustaining monotonicity) (a non-linear technique even for a linear equation, see flux limiters later on) (so we have (at least) 2nd order accuracy where the flow is smooth) (e.g. van Leer, 1979).

^aIn Godunov's scheme the dissipation is just strong enough to damp shorter waves before they get too much out of step and show up as oscillations on top of the larger features.

^bAlready in 1972, he developed a scheme 2nd order in space (using the (*crude*) minmod limiter) but only first order in time, using Forward Euler. Sadly, he succumbed to lung cancer in 1978, at the age of 37.

1. Estimate the fluid variables gradients, e.g. $\partial_x \rho$ (e.g. based on the averages on the neighboring cells)
2. Slope limit these gradients as otherwise we could introduce quite extreme values of the fluid variables at the cell boundaries, especially in case real fluid discontinuities are present

3. Estimate the values of the fluid variables at one interface by linear extrapolation

$$\rho_{i+\frac{1}{2}}^L = \rho_i + \frac{\Delta x}{2} (\partial_x \rho)_i, \quad \rho_{i+\frac{1}{2}}^R = \rho_{i+1} - \frac{\Delta x}{2} (\partial_x \rho)_{i+1} \quad (298)$$

See figure 64 for an illustration.

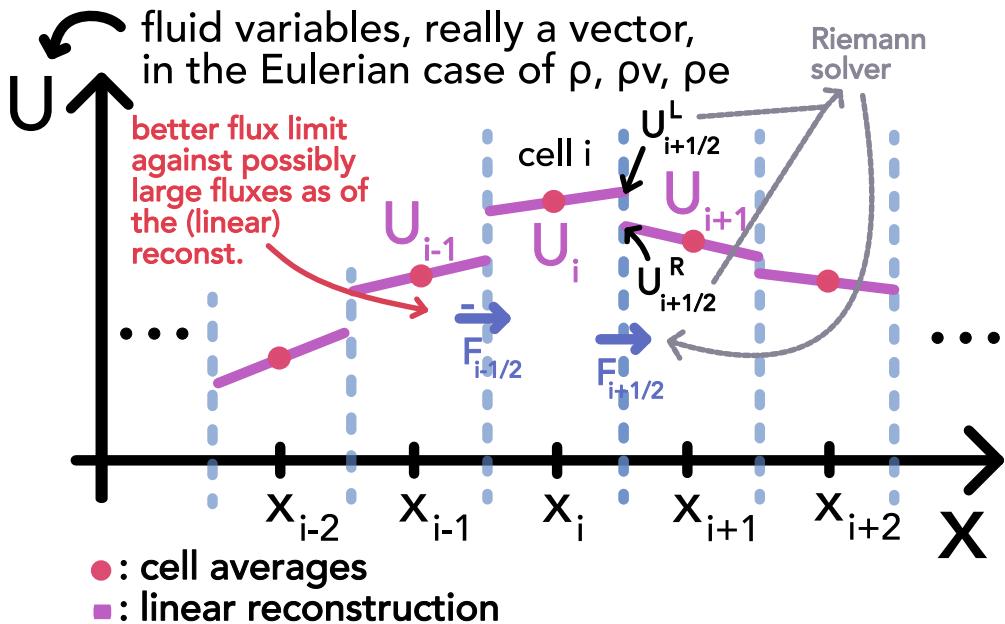


Figure 64: Linear extrapolation of the fluid variables to the cell interface.

4. Based on the extrapolated fluid variable values at the interface we apply our Riemann solver to find the flux and update the cell averages (although we do not have the piecewise-constant Riemann situation)

Problem: Above we have done a linear extrapolation to the boundary $\rho_{i+\frac{1}{2}}^L$ based on the gradient at the center $(\partial_x \rho)_i$. Note, however, that over our timestep Δt , ρ_i changes, so also our value at the boundary.

$$d\rho(x, t) = (\partial_x \rho)dx + (\partial_t \rho)dt \quad (299)$$

For more stability (and 2nd order accuracy), we do the extrapolation based on the gradient but also include the effect of $(\partial_t \rho)_i$ up until half a timestep (as a proxy to the situation throughout the timestep).

$$\begin{aligned} \rho_{i+\frac{1}{2}}^L &= \rho_i + (\partial_x \rho)_i \frac{\Delta x}{2} + (\partial_t \rho)_i \frac{\Delta t}{2}, & \rho_{i+\frac{1}{2}}^R &= \rho_{i+1} - (\partial_x \rho)_{i+1} \frac{\Delta x}{2} + (\partial_t \rho)_{i+1} \frac{\Delta t}{2} \\ \rightarrow \text{flux estimate } F_{\text{Riemann}} \left(\underline{U}_{i+\frac{1}{2}}^L, \underline{U}_{i+\frac{1}{2}}^R \right) && \end{aligned} \quad (300)$$

So for the entire state \underline{U} we have

$$\underline{U}_{i+\frac{1}{2}}^L = \underline{U}_i + (\partial_x \underline{U})_i \frac{\Delta x}{2} + (\partial_t \underline{U})_i \frac{\Delta t}{2}, \quad \underline{U}_{i+\frac{1}{2}}^R = \underline{U}_{i+1} - (\partial_x \underline{U})_{i+1} \frac{\Delta x}{2} + (\partial_t \underline{U})_{i+1} \frac{\Delta t}{2}$$

$(\partial_x \underline{U})_i$ calculated from finite difference approach + slope limiting

(301)

Idea of the MUSCL-Hancock scheme: Cell averages of the primitive fluid quantities are used to predict the values at the cell boundaries as $t + \frac{\Delta t}{2}$ and then use these prediction to calculate the fluxes and with them the primitive fluid quantities at $t + \Delta t$.

6.10.2.1 How to estimate the time derivatives $(\partial_t \underline{U})_i$? | MUSCL-Hancock scheme

Let us use the Euler equation in 1D (x is a scalar)

$$\partial_t \underline{U} + \partial_x F(\underline{U}) \underset{\text{quasi-linear form}}{=} \partial_t \underline{U} + \underline{\underline{J}}_U(F) \cdot \partial_x \underline{U} = 0 \rightarrow \boxed{\partial_t \underline{U} = -\underline{\underline{J}}_U(F) \cdot \partial_x \underline{U}}$$

(302)

with Jacobian matrix $\underline{\underline{J}}_U(F)$ of $F(\underline{U})$ with respect to \underline{U} , $\underline{\underline{J}}_U(F) \Big|_{\underline{U}=\underline{U}} =: \underline{\underline{A}}(\underline{U})$

We can therefore estimate the derivative in time based on our estimation $\partial_x \underline{U}$ of the derivative in space, yielding the **MUSCL-Hancock scheme** (Monotonic Upstream Scheme for Conservation Laws)

$$\underline{U}_{i+\frac{1}{2}}^L = \underline{U}_i + \left[\frac{\Delta x}{2} \underline{\underline{1}} - \underline{\underline{A}}(\underline{U}) \frac{\Delta t}{2} \right] (\partial_x \underline{U})_i$$

(303)

$$\underline{U}_{i+\frac{1}{2}}^R = \underline{U}_{i+1} + \left[-\frac{\Delta x}{2} \underline{\underline{1}} - \underline{\underline{A}}(\underline{U}) \frac{\Delta t}{2} \right] (\partial_x \underline{U})_{i+1}$$

- a 2nd order accurate extension of Godunov's scheme.

6.10.3 Idea and discussion of even higher order methods

We can

- use higher order polynomial reconstruction as in piecewise parabolic methods (also mind information transport by characteristic waves), see figure 65 (high order methods tend to create post shock oscillations → add dissipation mechanism / some flattening⁹ to PPM)

⁹Which essentially means that locally we go back to lower order.

- even higher order polynomials are used in methods like ENO and WENO (find polynomials based on values from multiple cells (*larger stencil*))

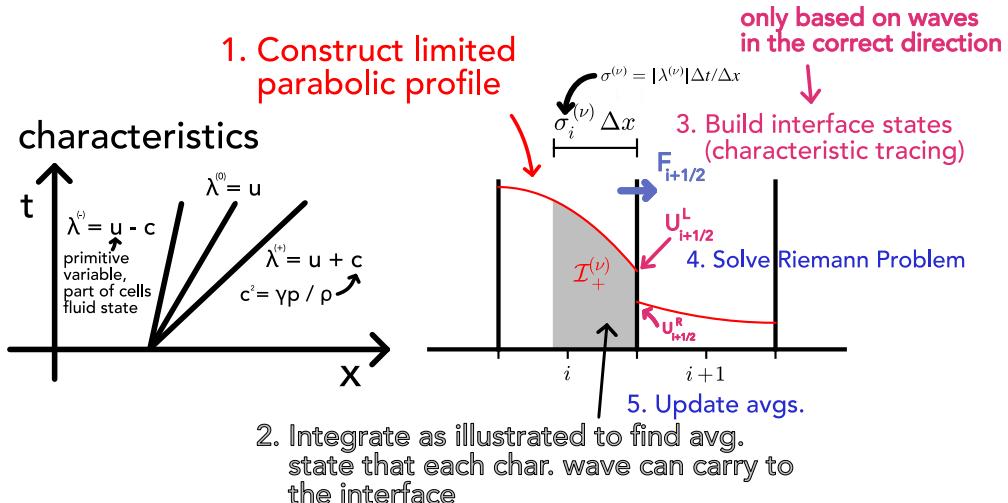


Figure 65: Piecewise parabolic reconstruction.

Note: Independent of the order, we only **store** one value (per variable) per cell in finite volume methods. In **finite element methods**, per cell polynomial representations of the state vector are stored (discontinuities are harder to capture though).

6.10.3.1 Discussion of higher order methods

Advantages and disadvantages of higher order methods can be found in table 11, those of lower order methods in table 12.

Pro higher	Con higher
<ul style="list-style-type: none"> sharp solution more accurate (sharper solutions alone can be inaccurate, e.g. when the bulk position is inaccurate) 	<ul style="list-style-type: none"> more expensive strong oscillations at discontinuities (principle illustrated in figure 66) crash at high Mach numbers

Table 11: Advantages and disadvantages of higher order methods.

Why do higher order schemes produce oscillations near discontinuities?

(here linear piecewise reconstruction)

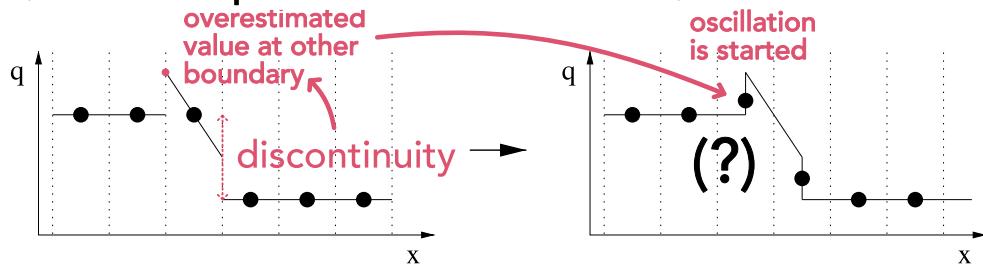


Figure 66: Illustration of the principle of oscillations at discontinuities in higher order methods.

Pro lower	Con lower
<ul style="list-style-type: none"> stable for complex flows no oscillations at discontinuities do not crash at high Mach numbers 	<ul style="list-style-type: none"> smear out solutions slower convergence to accurate solutions less accurate

Table 12: Advantages and disadvantages of lower order methods.

6.11 Flux / slope limiters | adaptively switching between a high and low order method

Problem: A system can contain very boring and strongly dynamic parts. It is very difficult to choose which solver is best for the global problem for all timesteps.

Idea: Dynamically switch between orders and solvers. Use 2nd order where possible and 1st order where necessary (e.g. for discontinuities).

Let us for simplicity consider a 1D problem with a single state variable u , e.g. the viscous Burgers equation

$$\partial_t u + \partial_x \left(\frac{1}{2} u^2 - \nu \partial_x u \right) = 0 \quad (304)$$

(which can be analytically solved using the Cole-Hopf transform $u = -2\nu \frac{1}{\phi} \partial_x \phi^{10}$).

¹⁰This is nice if we want to test different flux limiters against a ground-truth result.

Let $\mathcal{F}_{i+\frac{1}{2}}^H$ be a high-order flux computation and $\mathcal{F}_{i+\frac{1}{2}}^L$ be a low-order flux computation. We use the flux

$$\begin{aligned} F_{i+\frac{1}{2}}^{(n)} &= \mathcal{F}_{i+\frac{1}{2}}^{L,(n)}(U_i, U_{i+1}) + \phi_{i+\frac{1}{2}}^{(n)}(r) \cdot \left(\mathcal{F}_{i+\frac{1}{2}}^{H,(n)}(U_i, U_{i+1}) - \mathcal{F}_{i+\frac{1}{2}}^{L,(n)}(U_i, U_{i+1}) \right) \\ &\text{high order for } \phi_{i+\frac{1}{2}}^{(n)}(r) = 1, \quad \text{low order for } \phi_{i+\frac{1}{2}}^{(n)}(r) = 0 \end{aligned} \quad (305)$$

with

$$\text{flux limiter } \phi_{i+\frac{1}{2}}^{(n)}(r) \text{ based on the ratio } r = \frac{U_i - U_{i-1}}{U_{i+1} - U_i}$$

large if the jump of interest between U_i and U_{i+1} is large compared to the jump between U_{i-1} and U_i

(306)

with an exemplary flux limiter being

$$\phi_{\minmod} = \max(0, \min(1, r)), \quad r > 1 \rightarrow \phi_{\minmod} = 1 \rightarrow \text{high order} \quad (307)$$

(illustrated in figure 67) where it makes sense that for a big r we use the higher order method: If the jump between i and $i + 1$ is small compared to the jump between $i - 1$ and i , we can assume that the solution is smooth and use the higher order method (no discontinuity expected).

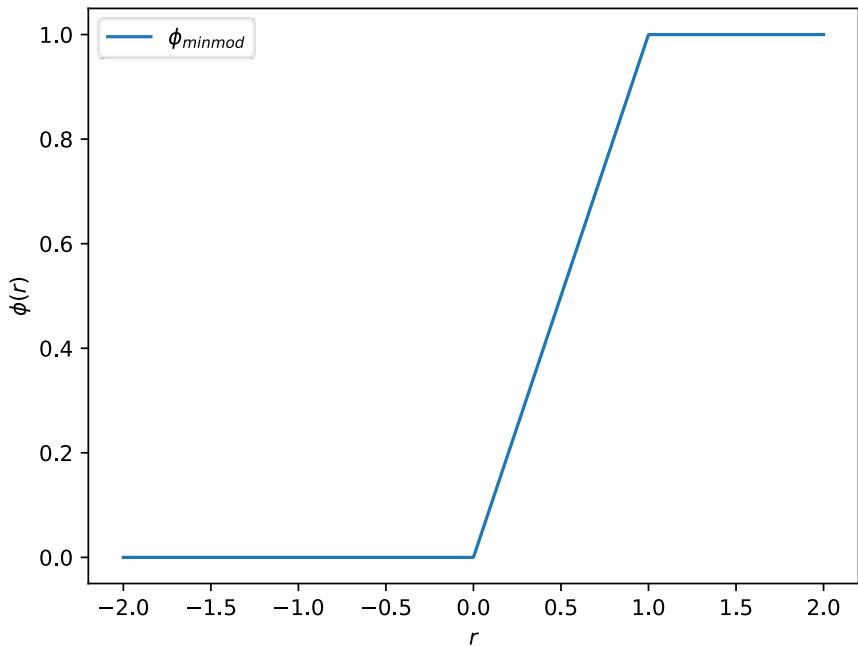


Figure 67: Minmod flux limiter.

Why is ϕ called a flux limiter? Higher order schemes can have larger slopes and predict larger fluxes, taking the lower order will has less of this problem.

6.11.1 Possibly advantageous properties of the flux limiter

It is often advantageous for the limiters to be symmetric with the property

$$\frac{\phi(r)}{r} = \phi\left(\frac{1}{r}\right) \quad (308)$$

so that the switch works the same for forward and backward facing gradients (mind the definition of r) (?).

Another property is based on the total variation

$$\text{TV}(\underline{U}) = \sum_i |U_{i+1} - U_i| \quad (309)$$

(where here \underline{U} is the vector over the grid points not fluid variables), called total variation diminishing (TVD) property

$$\text{TV}(\underline{U}^{(n+1)}) \leq \text{TV}(\underline{U}^{(n)}) \quad (310)$$

which means no oscillation will appear but real oscillations in the system will also not grow but rather smeared out. Minmod is TVD.

7 Smoothed Particle Hydrodynamics - Lagrangian Particle Method

Idea: In Smoothed Particle Hydrodynamics (SPH) we approximately solve the fluid equations numerically by replacing the fluid with a set of particles, we call *SPH-particles*^a. The equations of motion and properties of those particles are followed from the Lagrangian form of the continuity equations for the fluid. We then forward these particles in time using e.g. the leapfrog or semi-implicit Euler scheme.

^aCharacterized by their position and velocity. Additionally, hydrodynamic variables, e.g. ρ_i, T_i , are derived at the particles positions \underline{x}_i . Not to be confused with the real particles making up the fluid.

From this it makes sense that the main ingredients to the baseline scheme will be

- formulate the fluid equations in their Lagrangian¹¹ form
- formulate an algorithm to update the SPH-particles positions and velocities based on the Lagrangian fluid equations
- find expressions for the quantities used in the update-steps which goes hand in hand with finding how to get from the SPH-particle-perspective to continuous fluid quantities

Such a mesh-free scheme has **some key advantages**

- This particle representation of SPH has great conservation features - energy, linear momentum, angular momentum mass and specific thermodynamic entropy (more later; if we do not add artificial viscosity) are conserved (as we follow particles there is no loss of mass etc.).
- No advection errors, scheme is fully Galilean invariant (unlike mesh-based Eulerian techniques)
- As of the Lagrangian character, the local resolution of SPH follows the mass flow automatically → adaptive resolution, good for problems with vastly different densities

¹¹Co-moving with the flow rather than fixed as in the Eulerian perspective.

7.1 Lagrangian fluid equations (i.e. as material derivatives)

7.1.1 Continuity equation

Written as a material derivative, the continuity equation is

$$D_t \rho = -\rho (\underline{\nabla} \cdot \underline{v}), \quad \text{fluid mass density } \rho, \quad \text{fluid velocity } \underline{v} \quad (311)$$

which is zero for an incompressible fluid.

7.1.2 Navier-Stokes equation | Conservation law of Linear Momentum

A natural extension of Newtons 2nd law to continua is (including internal and external forces)

$$\rho D_t \underline{v} = \sum \text{forces} = -\underline{\nabla} P + \underline{\nabla} \cdot \underline{\underline{\Pi}} + \rho \underline{g} \quad (312)$$

stress tensor $\underline{\underline{\Pi}}$, pressure P , external accelerations \underline{g}

where a general approach to $\underline{\underline{\Pi}}$ is the Cauchy-Stress tensor for compressible flow

$$\underline{\underline{\Pi}} = \left\{ \mu \left[\underline{\nabla} \underline{v}^T + (\underline{\nabla} \underline{v}^T)^T - \frac{2}{3} (\underline{\nabla} \cdot \underline{v}) \underline{\underline{1}} \right] + \zeta (\underline{\nabla} \cdot \underline{v}) \underline{\underline{1}} \right\} \quad (313)$$

shear viscosity μ , bulk viscosity ζ

where for incompressible flow one yields

$$\rho D_t \underline{v} = -\underline{\nabla} P + \mu \underline{\nabla}^2 \underline{v} + \rho \underline{g} \quad (314)$$

Note: In an incompressible setting, the pressure P can be interpreted as a Lagrange multiplier which has to be chosen such that incompressibility really holds. Otherwise P is determined by a state equation, e.g. very basic $P(\rho) = k \left(\frac{\rho}{\rho_0} - 1 \right)$, $k > 0$ (variation of the ideal gas equation) or isothermal $P(\rho) = P_0 + c_0^2(\rho_0 - \rho)$, as previously discussed.

7.1.3 Energy equation

While we can forward the system only based on the Navier-Stokes equation and closure, let us write down the energy equation.

Let ϵ be the energy per volume so the energy per mass is $e = \frac{\epsilon}{\rho}$.

From basic thermodynamics, we can write for the internal energy U

$$dU = dQ + dW = dQ - PdV = TdS - PdV, \quad U = \epsilon V = eM$$

volume $V = \frac{M}{\rho} \rightarrow dV = -\frac{M}{\rho^2}d\rho$, total mass M , entropy S , pressure P

(315)

using this we find

$$\begin{aligned} \frac{de}{dt} &= \partial_t e + (\underline{v} \cdot \nabla) e = \frac{1}{M} \frac{dU}{dt} = \frac{1}{M} T \frac{dS}{dt} - \frac{1}{M} P \frac{dV}{dt} \\ &= T \frac{ds}{dt} + \frac{P}{\rho^2} \frac{d\rho}{dt} \underset{\text{continuity eq.}}{=} T \frac{ds}{dt} - \frac{P}{\rho} (\nabla \cdot \underline{v}) \end{aligned}$$
(316)

7.2 A simple SPH fluid simulator*

Based on the Navier-Stokes equation in Lagrangian form we can construct a simple fluid simulator, here using semi-implicit aka symplectic Euler¹². Pressure and viscosity forces are calculated separately. We still have to mind the CFL criterion (more on this later).

```

1   for sph_particle_i in sph_particles:
2       # reconstruct density  $\rho_i$  at  $\underline{x}_i$ 
3   for sph_particle_i in sph_particles:
4       ### viscous-force calculation in case of a viscous fluid
5       # / to make shocks resolvable using artificial viscosity
6       # compute  $\underline{a}_i^{viscosity}$  in the incomp. case =  $\nu \nabla^2 \underline{v}_i$ 
7       #  $\underline{v}_i^* = \underline{v}_i + \Delta t (\underline{a}_i^{viscosity} + \underline{g})$ , external accelerations  $\underline{g}$ 
8
9       ### pressure force calculation
10      # compute  $\underline{a}_i^{pressure} = -\frac{1}{\rho_i} \nabla P$ 
11
12      ### forward the particles, here using symplectic Euler
13      #  $\underline{v}_i(t + \Delta t) = \underline{v}_i^* + \Delta t \cdot \underline{a}_i^{pressure}$ 
14      #  $\underline{x}_i(t + \Delta t) = \underline{x}_i + \Delta t \cdot \underline{v}_i(t + \Delta t)$ 

```

Code-Snippet 1: Simple SPH fluid simulator. We need two loops, as to calculate e.g. the pressure force on one SPH-particle, the densities at positions of other SPH-particles are necessary.

To code up our simulator we have to answer

- How do we construct the density from the SPH-particles positions \underline{x}_i ?
- How do we calculate the gradients over fluid variables, so ∇P ?
- How do we handle viscosity?

¹²The PySPH package for instance uses a second order predictor-corrector method.

7.3 Smooth then discretize - smoothing kernels and their usage

Fluid quantities like the density are estimated through a kernel summation interpolant. Start by replacing a general fluid quantity $F(\underline{r})$ with a smoothed version by convoluting it with a smoothing kernel

$$F(\underline{r}) \rightarrow F_S(\underline{r}) \equiv \langle F(\underline{r}) \rangle = \int F(\underline{r}') W(\underline{r} - \underline{r}', h) d^3 \underline{r}', \quad \text{smoothing width } h \quad (317)$$

kernel W with $\int W(\underline{r}', h) d^3 \underline{r}' = 1$, $\langle F(\underline{r}) \rangle \xrightarrow[h \rightarrow 0]{} F(\underline{r})$, i.e. $W(\underline{r}, h) \xrightarrow[h \rightarrow 0]{} \delta(\underline{r})$

where the kernel has to be normed to not modify e.g. the total mass, and also differentiable (so that our fluid quantities are smooth). Typically, a spherical kernel is used

$$W(\underline{r}, h) = W(r, h) \quad (318)$$

which could be a Gaussian - but a Kernel with finite support, for instance a cubic spline, is better (more on that later).

7.3.1 Properties of the smoothing | approach for calculating derivatives of the smoothed fluid quantities

The smoothed version is 2nd order accurate with respect to the smoothing length (no first order correction as of the symmetry of the kernel)

$$\langle F(\underline{r}) \rangle = F(\underline{r}) + \mathcal{O}(h^2) \quad (319)$$

We can also find

$$\begin{aligned} \langle F(\underline{r}) + G(\underline{r}) \rangle &= \langle F(\underline{r}) \rangle + \langle G(\underline{r}) \rangle \\ \langle F(\underline{r}) \cdot G(\underline{r}) \rangle &= \langle F(\underline{r}) \rangle \cdot \langle G(\underline{r}) \rangle + \mathcal{O}(h^2) \\ \boxed{\frac{d}{dt} \langle F(\underline{r}) \rangle = \left\langle \frac{dF(\underline{r})}{dt} \right\rangle} \\ \underbrace{\nabla \langle F(\underline{r}) \rangle}_{\substack{= \\ \text{Kernel with compact support}}} &= \langle \nabla F(\underline{r}) \rangle \end{aligned} \quad (320)$$

where the main result that will allow us to make the fluid equation algebraic is

$$\langle \nabla F(\underline{r}) \rangle = \nabla \langle F(\underline{r}) \rangle = \int F(\underline{r}') \nabla W(|\underline{r} - \underline{r}'|, h) d^3 \underline{r}' \quad (321)$$

so we weight $F(\underline{r}')$ using the gradient of the kernel which can be pre-computed.

7.3.2 Discrete formulation of the smoothing

We now introduce the SPH-particles at positions \underline{r}_i where the fluid variable has value $F_i = F(\underline{r}_i)$. We assign a mass m_i to those particles. Together with the density $\rho_i = \rho(\underline{r}_i)$ we can write

$$\Delta r_i^3 \sim \frac{m_i}{\rho_i} \quad (322)$$

With this, assuming that those SPH-particles densely sample the space of interest, we can write the smoothed fluid quantity as

$$F_s(\underline{r}) \equiv \langle F(\underline{r}) \rangle \simeq \sum_{j=1}^{N_i} \frac{m_j}{\rho_j} F_j W(\underline{r} - \underline{r}_j, h), \quad \text{number of neighbors } N_i \quad (323)$$

Note: While this is similar to Monte-Carlo integration, the evaluation points are our SPH-particles which here turns out to be favorable over random sampling, as the distances between the particles tend to equilibrate due to pressure forces (making the interpolation smaller - still resulting noise is a problem of SPH).

$F_s(\underline{r})$ is defined everywhere and differentiable as the kernel is differentiable.

For the density we can write

$$\rho_s(\underline{r}) = \sum_{j=1}^{N_i} m_j W(\underline{r} - \underline{r}_j, h) \quad (324)$$

The smoothing length h should at least be larger than the particle distance.

7.3.3 Why a kernel with compact support is preferred?

Consider we are interested in the density ρ_i at \underline{x}_i . Note that the density at any point is based on the overlap of multiple smoothing kernels (see eq. 324).

Consider a Gaussian kernel

$$W_{\text{Gaussian}}(\|\underline{r}_i - \underline{r}_j\|, h) = \frac{1}{(\pi h^2)^{\frac{d}{2}}} \exp(-q^2), \quad q := \frac{\|\underline{r}_i - \underline{r}_j\|}{h}, \quad \text{dimension } d \quad (325)$$

then as of the infinite support, for the density ρ_i we would have to sum over all particles, and the density calculation at all SPH-particles $i = 1, \dots, N$ would be $\mathcal{O}(N^2)$.

Now for a cubic spline Kernel

$$\begin{aligned} W(q) &= \sigma_3 \left[1 - \frac{3}{2}q^2 \left(1 - \frac{q}{2} \right) \right], && \text{for } 0 \leq q \leq 1 \\ &= \frac{\sigma_3}{4}(2-q)^3, && \text{for } 1 < q \leq 2 \\ &= 0, && \text{for } q > 2 \end{aligned} \quad (326)$$

with normalization

$$d = 1 : \sigma_3 = \frac{2}{3h}, \quad d = 2 : \sigma_3 = \frac{10}{7\pi h^2}, \quad d = 3 : \sigma_3 = \frac{1}{\pi h^3} \quad (327)$$

the density calculation is only $\mathcal{O}(N_{ngb}N)$ with N_{ngb} being the average number of neighbors considered depending on the choice of h .

Note: The support size of SPH-particles is usually chosen to be $2h$ so that $W_{\text{Gaussian}}(\|\underline{r}_i - \underline{r}'\|, h) = 0$ for $\|\underline{r}_i - \underline{r}'\| > 2h$, see figure 68.

illustration for the case of constant smoothing length h

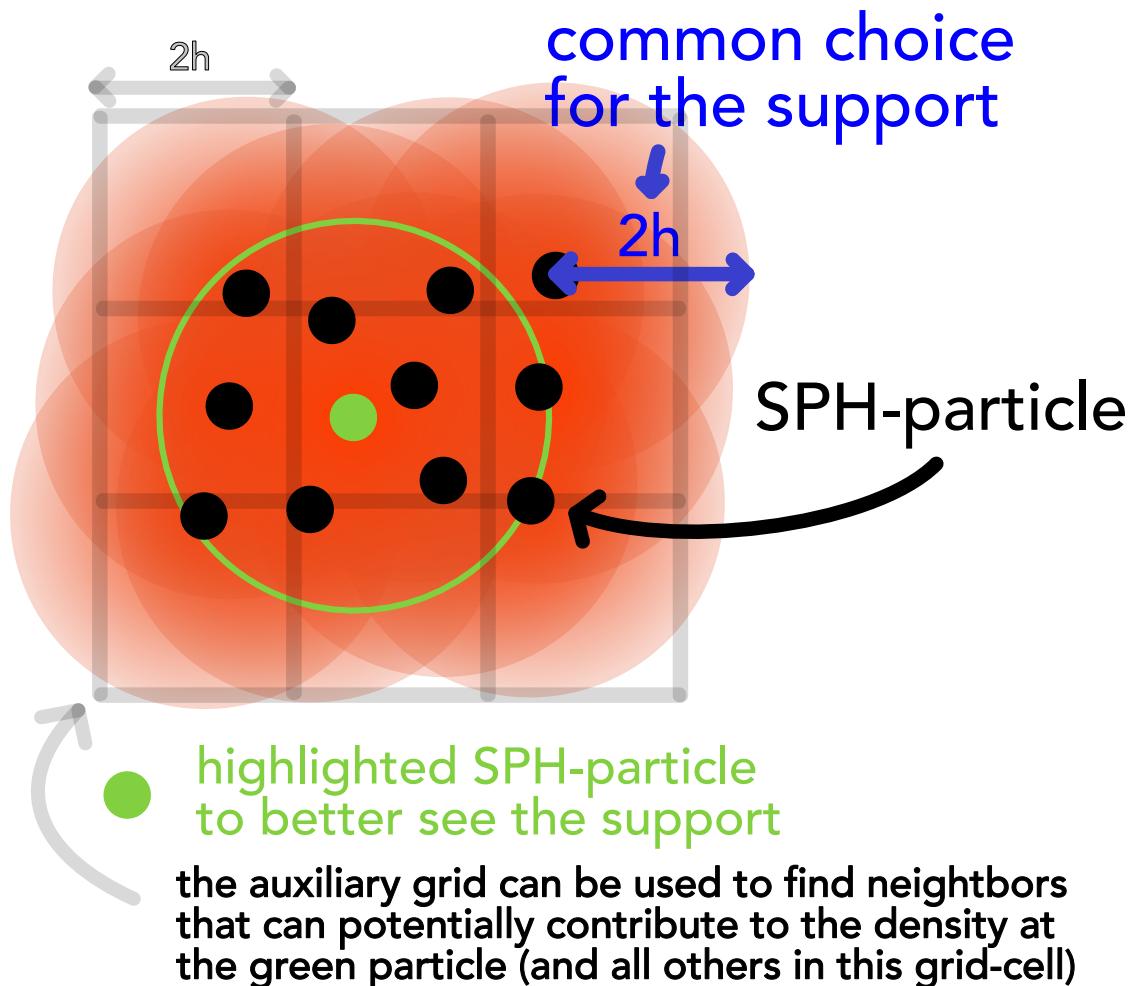


Figure 68: SPH illustration for fixed h .

7.3.4 How to make the smoothing length h variable in space to account for variations in the density? | sampling procedure in SPH - scatter and gather approach

Problem: Choosing a good overall smoothing length h is a problem in any algorithm using kernels (for instance also in Kernel Density Estimation). If h is chosen too large, we lose the details of the density distribution, if h is chosen to small, we get not a smooth density distribution but one with peaks at the positions of our SPH-particles. It makes more sense to use an adaptive smoothing length which is smaller in regions where SPH-particles are denser (to still resolve details there) and larger where they are more rarefied, so that in such regions the overall density would still make sense for a - in reality - continuous fluid, see figure 69

Idea: Use a variable smoothing length h .

There are two general approaches for introducing a variable smoothing length ρ_i into the calculation of our fluid quantities - the scatter and gather approach, as shown in table 13 and illustrated in figure 70.

Symmetry problem: Consider the effect of a variable smoothing length on how SPH-particles affect each other. In the schemes above this is not necessarily symmetric, leading to a force asymmetry and Newton's third law being broken (no conservation of total angular momentum).

We therefore have to symmetrize the equations of motion in $h_i = h(\underline{r}_i)$ and $h_j = h(\underline{r}_j)$. We make forces antisymmetric by substituting e.g.

$$h_{ij} = \frac{h_i + h_j}{2}, \quad \text{or geometric mean} \quad h_{ij} = \sqrt{h_i h_j} \quad (330)$$

for h_i and h_j in the force calculations. Therefore, a symmetric approach to the density is

$$\rho_s(\underline{r}_i) = \sum_{j=1}^{N_i} m_j W(r_{ij}, h_{ij}), r_{ij} = |\underline{r}_i - \underline{r}_j| \quad (331)$$

Note: As h is a function of \underline{r} in the gather scheme, we must account for this in

$$\underline{\nabla}W(|\underline{r} - \underline{r}'|, h) = \underline{\nabla}W(|\underline{r} - \underline{r}'|, h)|_{h=const.} + (\underline{\nabla}h)\partial_h W(|\underline{r} - \underline{r}'|, h)|_{h=const.} \quad (332)$$

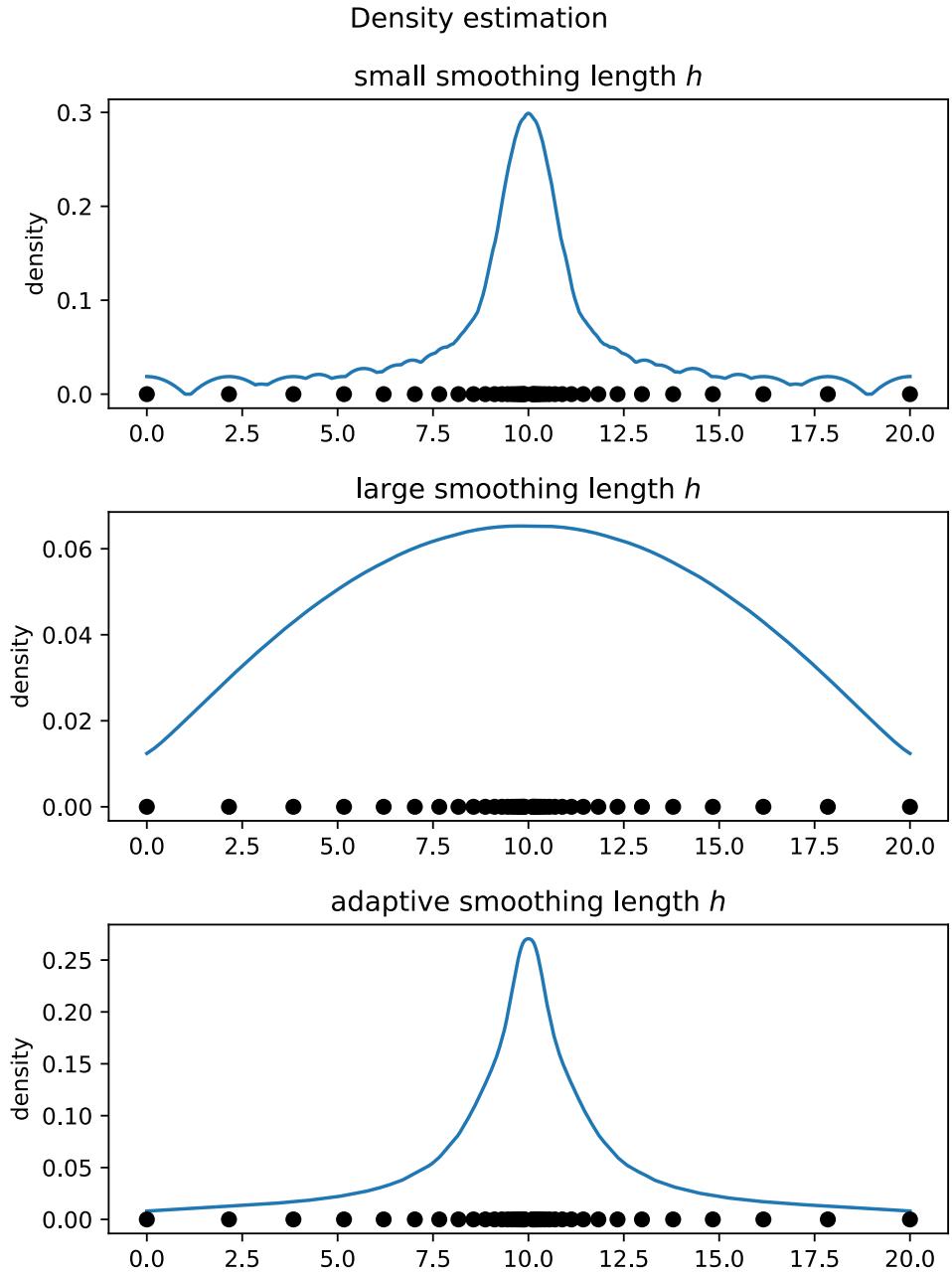


Figure 69: Density estimation

7.3.4.1 How to choose h_i ?

We adjust h_i so that we always consider $50 \lesssim N_{ngb} \lesssim 500$ neighbors.

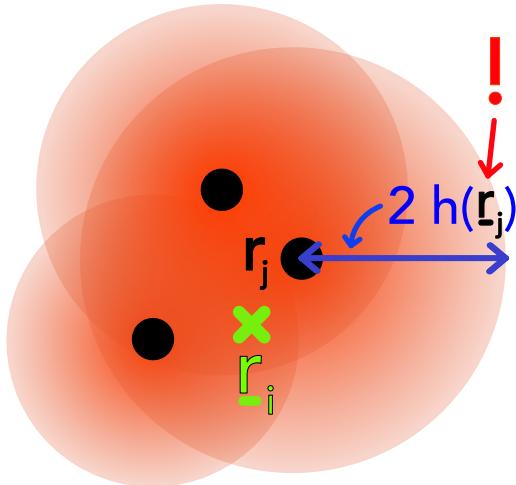
One choice is to choose h to keep N_{ngb} constant. Another is to scale h according to the density (higher for lower densities), e.g.

$$h_i = h_0 \left(\frac{\rho_0}{\rho_i} \right)^3, \quad \frac{dh}{dt} = -\frac{1}{3} \underbrace{\frac{h}{\rho} \frac{d\rho}{dt}}_{\text{continuity eq.}} = \frac{1}{3} h \nabla \cdot \underline{v}, \quad \text{dimension } d \quad (333)$$

scatter

each SPH-particle is assigned a smoothing length, the density at any point is calculated from the overlap of all surrounding density distributions

scatter

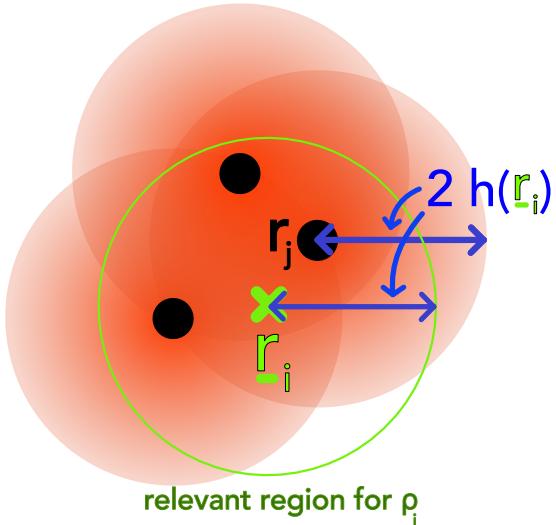


r_i : point of interest

gather

The observer at a position \underline{r} has its smoothing length h and assigns it to all SPH-particles in the relevant region. Assuming kernels with a compact support $2h$ this relevant region - starting from the observer - has radius $2h$.

gather



r_i : point of interest

$$\langle F(\underline{r}) \rangle = \int_{N_i} F(\underline{r}') W(\underline{r} - \underline{r}', h(\underline{r}')) d^3 \underline{r}'$$

$$\rho_s(\underline{r}_i) = \sum_{j=1}^{N_i} m_j W(r_{ij}, h_j), r_{ij} = |\underline{r}_i - \underline{r}_j| \quad (328)$$

$$\langle F(\underline{r}) \rangle = \int_{N_i} F(\underline{r}') W(\underline{r} - \underline{r}', h(\underline{r})) d^3 \underline{r}'$$

$$\rho_s(\underline{r}_i) = \sum_{j=1}^{N_i} m_j W(r_{ij}, h_i), r_{ij} = |\underline{r}_i - \underline{r}_j| \quad (329)$$

The total mass is conserved in the scatter approach, $\int \rho_s d^3 \underline{r} = \sum m_i$

The total mass is not conserved in the gather approach, the error is only $\mathcal{O}(h^2)$ though.

Table 13: Scatter and gather approach for introducing a variable smoothing length h .

so that the mass ρh^3 is constant.

The neighbors can be found using a tree structure for partitioning space.

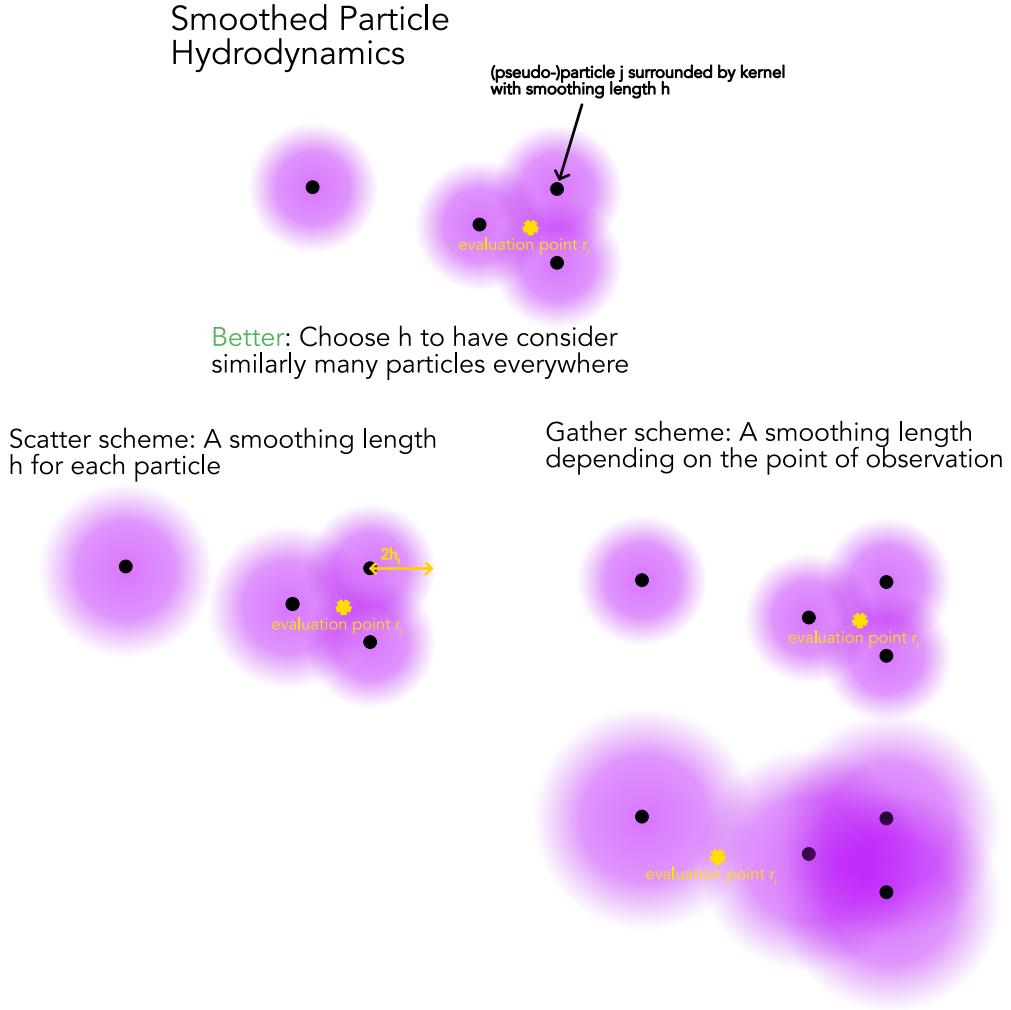


Figure 70: Scatter and gather approach for introducing a variable smoothing length h .

7.4 SPH continuity equation and equations of motion

We now know how to construct the densities ρ_i at the SPH-particles positions in one loop. Those densities are required for subsequent calculation of pressure forces etc. in a second loop.

7.4.1 SPH continuity equation

While better not used in forwarding the system, the SPH continuity equation can still be of interest.

Directly applying eq. 321 to the continuity equation in Lagrangian form, so $D_t \rho = -\rho \nabla \cdot \underline{v}$, turns out to be extremely sensitive to the particle distribution, we can rewrite

$$\rho \nabla \cdot \underline{v} = \nabla \cdot (\rho \underline{v}) - \underline{v} \cdot \nabla \rho \quad (334)$$

and applying eq. 321 and 323 we find for the i -th SPH-particle

$$\langle \underline{\nabla} \underline{v} \rangle_i = \frac{1}{\rho_i} [\langle \underline{\nabla}_i \cdot (\rho \underline{v})_i \rangle - \underline{v}_i \cdot \langle \underline{\nabla} \rho \rangle_i] = \frac{1}{\rho_i} \sum_{j=1}^{N_i} m_j (\underline{v}_j - \underline{v}_i) \underline{\nabla}_i W(r_{ij}, h_{ij}) \quad (335)$$

(which has the advantage that as it should be for all equal velocities $\underline{v}_i = \underline{v}_j$ we have $\langle \underline{\nabla} \underline{v} \rangle_i = 0$).

and thus the SPH continuity equation

$$\frac{d\rho_i}{dt} \simeq \sum_{j=1}^{N_i} m_j \underline{v}_{ij} \underline{\nabla}_i W(r_{ij}, h_{ij}), \quad \underline{v}_{ij} = \underline{v}_i - \underline{v}_j \quad (336)$$

7.4.2 Gradients in SPH

Note: There are multiple ways to obtain the SPH equations of motion. One is to find an expression for gradients and apply it to the Euler equations, another a variational approach, which directly guarantees certain conservation laws^a

^aThere we start from the Lagrangian for inviscid (zero-viscosity) flow and then derive the Lagrangian equations of motion. The symmetries of the Lagrangian and absence of explicit time dependence directly give us energy conservation, momentum conservation from the translational invariance and momentum conservation from the rotational invariance.

As with the density, it is better not to directly apply eq. 321 but better to use the identities

$$\begin{aligned} \underline{\nabla} \cdot F(\underline{r}) &= \frac{1}{\rho} [\underline{\nabla} \cdot (\rho F(\underline{r})) - F(\underline{r}) \underline{\nabla} \cdot \rho] \\ \underline{\nabla} \cdot F(\underline{r}) &= \rho \left[\underline{\nabla} \cdot \left(\frac{F(\underline{r})}{\rho} \right) + \frac{F(\underline{r})}{\rho^2} \underline{\nabla} \cdot \rho \right] \end{aligned} \quad (337)$$

to obtain

$$\langle \underline{\nabla} \cdot F(\underline{r}) \rangle_i = \frac{1}{\rho_i} \sum_{j=1}^{N_i} m_j [F(\underline{r}_j) - F(\underline{r}_i)] \cdot \underline{\nabla}_i W_{ij}, \quad W_{ij} = W(r_{ij}, h_{ij}) \quad (338)$$

and the symmetric form

$$\langle \underline{\nabla} \cdot F(\underline{r}) \rangle_i = \rho_i \left[\sum_{j=1}^{N_i} m_j \left[\frac{F(\underline{r}_j)}{\rho_j^2} + \frac{F(\underline{r}_i)}{\rho_i^2} \right] \cdot \underline{\nabla}_i W_{ij} \right] \quad (339)$$

7.4.3 SPH Euler equation | The central ingredient to making our simple fluid simulator work

With the gradient-expression, we can tackle the Euler equation in convective (Lagrangian) from (Navier-Stokes without stress tensor / external forces), in other words we can find the **acceleration as of pressure** $\underline{a}_i^{\text{pressure}}$.

$$\underline{a}_i^{\text{pressure}} = -\frac{\nabla P}{\rho} \quad (340)$$

so in the (anti)symmetric form

$$\boxed{\underline{a}_i^{\text{pressure}} = -\sum_{j=1}^{N_i} m_j \left[\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2} \right] \cdot \nabla_i W_{ij}} \quad (341)$$

This is the acceleration we need to get our simple fluid simulator working - P follows from the equation of state.

Note: Artificial viscosity will have to be added to allow for the treatment of shocks.

The equation is antisymmetric in i and j so momentum is conserved locally and globally (also follows from the variational method).

Other symmetrizations

$$\text{e.g. } \frac{1}{2} \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \leftrightarrow \frac{\sqrt{P_i P_j}}{\rho_i \rho_j} \quad (342)$$

and additional correction factors are possible.

7.4.4 Including further accelerations

Further accelerations like self gravity can be calculated as usual

$$\underline{g}_i = -\nabla \Phi_i = -G \sum_{j=1}^{N_i} m_j \frac{\underline{r}_{ij}}{r_{ij}^3} \quad (343)$$

7.5 Artificial Viscosity

The SPH equations formulated above keep the specific thermodynamic entropy A_i strictly constant.

The Euler equations, however, can produce true discontinuities in form of shock waves or contact discontinuities¹³ where the specific entropy increases at the shock front - which our current SPH scheme will never display. We must introduce artificial viscosity for the necessary dissipation processes producing heat and entropy to be possible. The artificial viscosity dampening the motion of particles broadens the shock layer into a differentially resolvable form. Also, without artificial viscosity, particles would be able to interpenetrate. We want the viscosity to only be active at shocks and not disturb our ideal gas behavior.

7.5.1 Viscous Pressure

Based on the discretized estimate $\langle \nabla \cdot \underline{v} \rangle_i$, viscosity can be added in form of the following pressure (von-Neumann-Richtmyer-Landshoff)

$$p_i^{AV} = \begin{cases} \underbrace{-\alpha_i^{AV} \rho_i c_i h_i (\nabla \cdot \underline{v})_i}_{\text{combined shear and bulk viscosity, dampens post shock osci.}} + \underbrace{\beta_i^{AV} \rho_i h_i^2 (\nabla \cdot \underline{v})_i^2}_{\text{Richtmyer viscosity, prevent interpenetration in high Mach number shocks}} & \text{if } (\nabla \cdot \underline{v})_i < 0, \\ 0 & \text{otherwise} \end{cases} \quad (344)$$

density ρ , sound speed c , smoothing length h , parameters α, β

7.5.2 Adding the artificial viscosity to the equation of motion

To our SPH Euler equation, we can add the viscous force as

$$\begin{aligned} \underline{a}_i^{\text{visc}} &= \frac{d \underline{v}_i}{dt} \Big|_{\text{visc}} = - \sum_{j=1}^{N_i} m_j \Pi_{ij} \cdot \nabla_i \overline{W_{ij}} \\ \overline{W_{ij}} &= \frac{1}{2} W_{ij} \left(\frac{h_i + h_j}{2} \right), \quad \Pi_{ij} \text{ should be symmetric} \rightarrow \text{antisymmetric force} \end{aligned} \quad (345)$$

where keeping the force antisymmetric retains conservation of linear and angular momentum.

¹³At the shock front the differential form of the Euler equation breaks down and the integral form leading to the Rankine-Hugoniot jump conditions has to be used.

For the viscous stress tensor we can model

$$\text{one possibility } \Pi_{ij} = \begin{cases} [-\alpha c_{ij} \mu_{ij} + \beta \mu_{ij}^2] / \rho_{ij} & \text{if } \underline{v}_{ij} \cdot \underline{r}_{ij} < 0 \\ 0 & \text{otherwise} \end{cases}, \quad \mu_{ij} = \frac{h_{ij} \underline{v}_{ij} \cdot \underline{r}_{ij}}{\left| \underline{r}_{ij} \right|^2 + \epsilon h_{ij}^2}$$

mean sound speed $c_{ij} = \frac{c_i + c_j}{2}$, singularity protection $\epsilon \simeq 0.01$
 viscosity strength regulated by $\alpha \simeq 0.5$ to 1 , $\beta \simeq 2\alpha$

(346)

This form of a viscous force is a combination of a bulk and von-Neumann-Richtmyer viscosity and only acts if two particles (rapidly) approach ¹⁴ each other. It is Galilean invariant and vanishes for rigid body rotation.

The total momentum equation then is

$$\frac{d\underline{v}_i}{dt} = - \sum_{j=1}^{N_i} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij} \right) \nabla_i W(r_{ij}, h_{ij}) - \nabla \phi_i \quad (347)$$

In order to conserve total energy, work done against the viscous force has to show up as heat (discussed later).

7.5.3 Shear-Flow-Balsara correction

Problem: In the above we will also add high viscosity to shear flows (which we do not want), where particles also quickly approach each other (move adjacently with different velocities)

Idea: Such shear flows are marked by high vorticity $\omega = \nabla \times \underline{v}$, so for high vorticity we can crank down the artificial viscosity.

$$\tilde{\mu}_{ij} = \mu_{ij} \cdot \frac{f_i + f_j}{2}, \quad f_i = \quad (348)$$

7.5.4 Further viscosity switches

To reduce viscosity far away from shocks many other switches have been proposed, e.g. by making the viscosity strength parameters α variable in time and adopting $\beta = 2\alpha$.

¹⁴ $\underline{v}_{ij} \cdot \underline{r}_{ij} < 0$, and μ_{ij} measure how strongly two particles approach each other.

Idea: We only want high Π_{ij} when there is high compression, $\underline{\nabla} \cdot \underline{v}$ strongly negative, which we can use as a switch after which we let α_i decay exponentially.

7.6 SPH energy equation with artificial viscosity

The work done against the viscous force can be accounted in terms of energy or entropy.

Let us start with the hydrodynamic energy equation

$$\frac{d\epsilon}{dt} = \frac{\partial\epsilon}{\partial t} + \underline{v} \cdot \underline{\nabla}\epsilon = \frac{ds}{dt} - \frac{p}{\rho} \underline{\nabla} \cdot \underline{v} \quad (349)$$

from which for adiabatic systems with $\frac{ds}{dt} = 0$ and with eq. 338 we find

$$\frac{d\epsilon_i}{dt} = \frac{p_i}{\rho_i^2} \sum_{j=1}^{N_i} m_j \underline{v}_{ij} \cdot \underline{\nabla}_i W(r_{ij}, h_{ij}) \quad (350)$$

(where here we do not take the symmetric form as it can lead to unphysical solutions like negative internal energy).

To this we add a dissipation term due to artificial viscosity and incorporate heating and cooling sources into a function Γ_i to obtain

$$\frac{d\epsilon_i}{dt} = \underbrace{\frac{p_i}{\rho_i^2} \sum_{j=1}^{N_i} m_j \underline{v}_{ij} \cdot \underline{\nabla}_i W_{ij}}_{\text{from hydrodynamic energy equation}} + \underbrace{\frac{1}{2} \sum_{j=1}^{N_i} m_j \Pi_{ij} \underline{v}_{ij} \cdot \underline{\nabla}_i W_{ij}}_{\text{dissipation from art. viscosity}} + \underbrace{\Gamma_i}_{\text{heating and cooling}} \quad (351)$$

7.7 SPH Entropy equation

Alternatively to the energy equation, one can integrate an equation for the specific thermodynamic entropy A_i . The entropic function is defined by

$$P = A(s)\rho^\gamma, \quad \text{adiabatic index } \gamma \quad (352)$$

from which the internal energy follows as

$$\epsilon = \frac{P}{(\gamma - 1)\rho} = \frac{A(s)}{\gamma - 1} \rho^{\gamma-1} \quad (353)$$

The SPH entropy equation can be derived as

$$\frac{dA_i}{dt} = -\frac{\gamma-1}{\rho_i}\Gamma_i + \frac{1}{2}\frac{\gamma-1}{\rho_i^{\gamma-1}}\sum_{j=1}^{N_i} m_j \Pi_{ij} \underline{v}_{ij} \cdot \nabla_i W(r_{ij}, h_{ij}) \quad (354)$$

From this using eq. 353 we can retrieve the internal energy and the temperature T_i (proportional to ϵ_i).

7.8 Maximum timestep - CFL criterion

A possible criterion is

$$\Delta t_i^{CFL} = 0.3 \frac{h_i}{h |(\nabla \cdot \underline{v})_i| + c_i + 1.2 (\alpha_i c_i + \beta_i h_i |\min((\nabla \cdot \underline{v})_i, 0)|)} \quad (355)$$

sound speed c_i , viscosity strength α_i, β_i

7.9 Notes on boundary modeling*

The two basic options are

- use fixed dummy particles (however those lead to the violation of the conservation of energy)
- use a fluid-solid force (e.g. inspired by the Lenard-Jones potential)

7.10 Reversibility in the context of viscosity-free, weakly-compressible SPH*

Note that a fluid rising up to a dam form again after a dam-break and anti-dissipative or *clumping-up* so anti-pressure behavior are very different. The advantage of our SPH particles representing a fluid so acting on e.g. pressure unlike normal particles, the main advantage of rediscretizing based on the fluid equations, is of course sustained.

The SPH equations (based on the Euler fluid equations) lead us to a numerically time-reversible scheme¹⁵ for the evolution of the positions and velocities of our SPH-particles - we must take care though and

- use a reversible, symplectic method like leapfrog
- calculate the density from the current positions of the particles not the evolution equation avoiding accumulation of density errors and making our SPH evolution symplectic.

¹⁵Solving backwards in time while recovering the initial conditions.

- use fixed-point over floating-point arithmetic, to avoid floating point errors violating reversibility

Note: The SPH-particles (without added viscosity) obey reversible Hamiltonian dynamics.

Indeed, for the N particle distribution function $f_N(t, \underline{r}_1, \underline{p}_1, \dots, \underline{r}_N, \underline{p}_N)$ the Liouville entropy

$$S^{\text{Liouville}}(f_N) = -\frac{k_B}{N!} \int d\underline{r}_1 \int d\underline{p}_1 \cdots \int d\underline{r}_N \int d\underline{p}_N f_N \ln(h^{3N} f_N) \quad (356)$$

remains - in theory and in the reversible simulation (Kincl and Pavelka, 2023) - constant.

However, the Boltzmann entropy - which is obtained by maximizing the Liouville entropy under the constraint of the one-particle distribution $f(t, \underline{r}, \underline{p})$ ¹⁶

$$S^{\text{Boltzmann}}(f) = -k_B \int d\underline{r} d\underline{p} f (\ln(h^2 f) - 1) \quad (357)$$

grows in the simulation of the dam break experiment in Kincl and Pavelka, 2023 (with velocity distribution becoming Maxwellian).

So forgetting the exact positions and momenta of our SPH particles, the one-particle density they describe increases in Boltzmann entropy, the 2nd law of thermodynamics emerges (statistically).

In the words from Kincl and Pavelka, 2023: »In summary, if we see all the positions and momenta in the SPH simulation, we can not see the second law of thermodynamics. Indeed, the simulation is reversible and the Liouville entropy remains constant. However, when we only focus on the one-particle distribution function, we can see the growth of Boltzmann entropy and thus irreversible behavior.«.

Note: Since we have a discrete (as of our discretization), deterministic, reversible system which can only exist in finitely many states, it is recurrent, thus will after long enough time come back to its initial state (unlikely though as of the high-dimensional phase space).

Note: The situation here is without any viscosity, so that if we invert the velocities and come back to our previous state (see figure 71) this is not anti-dissipation and does not happen when the system is dissipative (with viscosity). (?)

¹⁶Normalized to the number of particles.

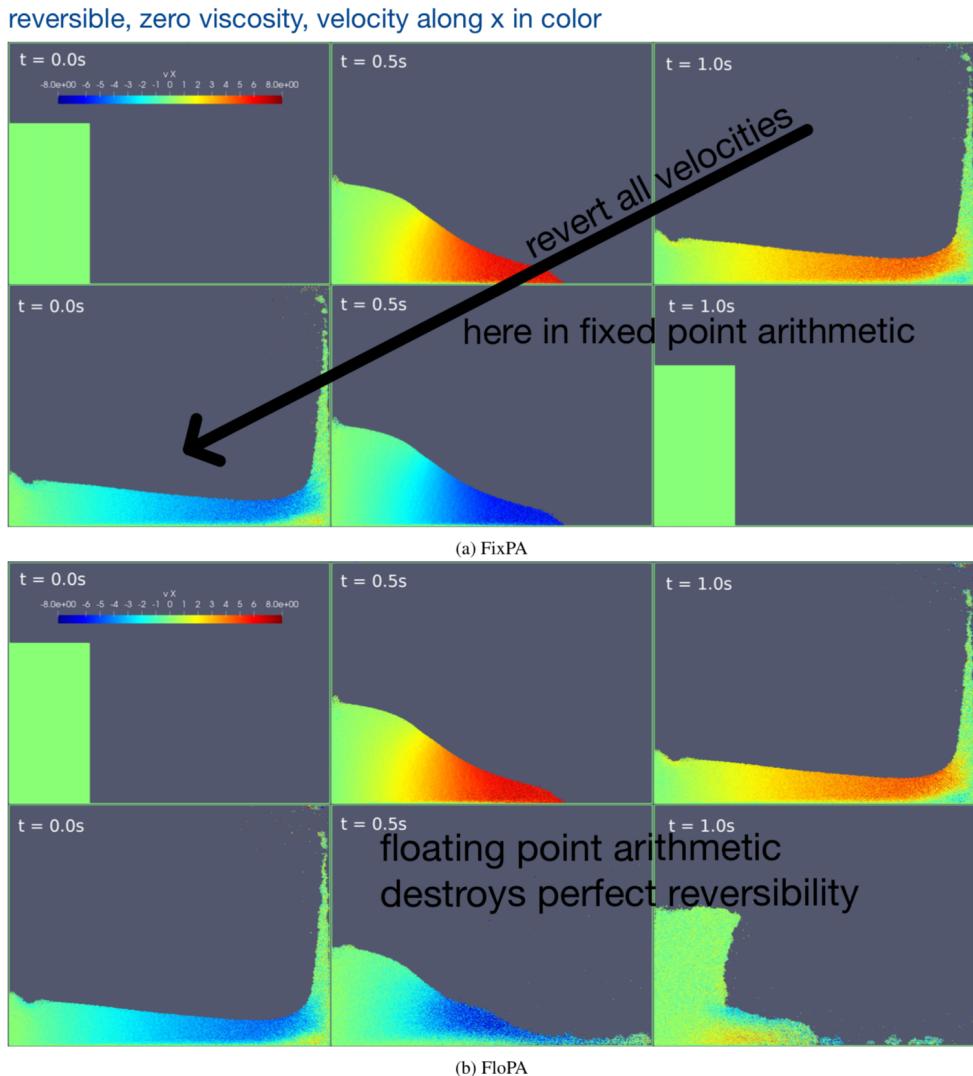


Figure 71: Reversible SPH simulation without viscosity at the example of a breaking dam. In floating-point arithmetic this reversible behavior is broken. Being in an initial state leading back to the *dam* is very unlikely so even slight deviations will make us not go back to the dam.

Note the different levels at play: The SPH-particles are artificial in that they describe the fluid not the fluid particles. So while the SPH-particles start out resting, there is still temperature and pressure. However, in the dam experiment (fig. 71) (and generally) the SPH particles themselves thermalize to a Boltzmann distribution (they start out in non-equilibrium), fitting to the equilibrium Boltzmann entropy (although we do not add any heat, Boltzmann entropy of the SPH-particles (**based on their phase space distribution**) increases as we go from non-equilibrium to equilibrium). Note that this is not the same as the specific thermodynamic entropy of the fluid, as previously described.

7.11 Notes on the conservative formulation using Lagrange multipliers

Starting with the Lagrangian for inviscid, compressible flow

$$\mathcal{L} = \int \rho \left\{ \frac{1}{2} v^2 - u(\rho, s) \right\} d^3 r \quad (358)$$

the SPH equation with $s = \text{const.}$ follows from the Lagrange equation

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \underline{v}} - \frac{\partial \mathcal{L}}{\partial \underline{r}} = 0 \quad (359)$$

(for variable smoothing length h under the constraint (Lagrange multiplier) $\rho_j h_j^3 = \text{const.}$) as

$$\frac{d\underline{v}_i}{dt} = - \sum_{j=1}^{N_i} m_j \left\{ \frac{1}{f_i} \frac{p_i}{\rho_i^2} \nabla_i W(r_{ij}, h_i) + \frac{1}{f_j} \frac{p_j}{\rho_j^2} \nabla_i W(r_{ij}, h_j) \right\}, \quad f_i = \left[1 + \frac{h_i}{3\rho_i} \frac{\partial \rho_i}{\partial h_i} \right] \quad (360)$$

From the symmetries of the Lagrangian it follows that energy, specific thermodynamic entropy, linear and angular momentum are conserved.

7.12 Further improvements

Ideas for improvement are

- it may be more physical to move particles with a smoothed flow velocity
- ...

7.13 Advantages and Disadvantages of SPH

Advantages and disadvantages of SPH are summarized in table 14.

Advantages	Disadvantages
<ul style="list-style-type: none"> • versatile and simple • avoids numerical diffusion issues arising when advection is not aligned with the mesh • automatically adaptive resolution, can cover large ranges in density and space • excellent conservation properties (energy, linear and angular momentum, not guaranteed in Eulerian codes) • inherent mass conservation • Galilean invariant and free from advection errors • can deal with complicated geometries as mesh-free • simple and transparent codes • very robust • as a particle scheme it is good for describing the transition from gaseous to stellar dynamical systems (e.g. formation of stellar clusters) 	<ul style="list-style-type: none"> • limited accuracy in multi-D flow • The number of neighboring particles (based on the neighbors positions we calculate the density and then equations of motion at the particle of interest) in SPH is far greater than the number of neighboring nodes in mesh based methods • low density regions are more poorly resolved • poorly handles shocks • noise as of discrete sums over limited neighbor set • jitter develops • fluid instabilities across contact discontinuities are problematic such as Kelvin helmholtz instabilities • artificial viscosity limits the Reynolds number which can be reached • low convergence rate ¹⁷ • approximation of boundary conditions can be difficult • formation of voids • free surfaces can be problematic (density underestimated there) • magnetic fields are hard to handle (problems with stability and $\nabla \cdot \underline{B} = 0$ requirement)

Table 14: Advantages and disadvantages of SPH.

7.14 Outlook: Machine-learning enhanced multiscale-physics SPH simulation

Consider a SPH simulation of the Milky Way. In the N-body/SPH simulation individual SPH particles represent a clump of dark matter, gas or a group of stars.

It is desirable to use particles with as small mass as possible, the current state of the art is $10^3 M_{\odot}$ ^a. Hirashima et al., 2023 push for a model with resolution $1M_{\odot}$ for galaxy formation and evolution.

^a $1M_{\odot}$ is the mass of the sun.

Problem: Such high resolution galaxy evolution simulations consider multi-scale physics - a tiny fraction of short timescale regions (e.g. Supernovae) becomes the bottleneck for large scale parallel computation due to **Ahmdals law** (discussed later).

Idea: Replace the direct simulation of short time-step regions with a *surrogate* machine learning model (that can directly do a large physically adequate time-step) that learns from supernova simulations in turbulent gas clouds.

Problem: The machine learning model (e.g. U-Net) needs a voxel (grid) representation. Density temperature and 3D velocity can e.g. be represented as 5 scalar fields on a 3D grid.

Idea: Based on the SPH particles we can construct a 3D grid representation that we can feed into our ML model (e.g. U-Net) in the respective regions. After doing the larger timestep with U-Net, we get back from the Eulerian voxel / field representation to SPH-particles by Gibbs-Sampling^a (a Markov Chain Monte Carlo method).

^aA technique for generating samples of the marginals based on the conditional distributions in a multivariate setting. Consider e.g. we have a probability distribution for particles in 3D, $p(x, y, z)$. Imagine that one can easily sample from the conditionals but not easily from the known joint distribution. **Gibbs sampling**:

1. Choose an initial state (x_0, y_0, z_0)
2. Do an iteration with $t = 1, 2, \dots, N$:
 - (a) Sample $x^{(t)}$ from $p(x|y^{(t-1)}, z^{(t-1)})$
 - (b) Sample $y^{(t)}$ from $p(y|x^{(t)}, z^{(t-1)})$
 - (c) Sample $z^{(t)}$ from $p(z|x^{(t)}, y^{(t)})$

Where the conditionals just follow from the joint distribution $p(x, y, z)$ by plugging in the calculated values for the other variables. A burn in of ~ 1000 iterations is often used. **Note that a standard global Metropolis Hastings algorithm can get us (multivariate) samples from the total joint distribution directly, not by sampling from the marginals separately.** The rough rationale behind Gibbs sampling is that it might be easier to propose updates to one variable at a time than to all simultaneously as in such a Metropolis Hastings algorithm (Gibbs sampling is also a special case of a Metropolis Hastings algorithm).

8 Finite Element Methods

Finite element methods (FEM) are a class of methods for solving PDEs. Advantages include

- can work with flexible geometries
- handle geometrically intricate boundary conditions
- spatially adapt the resolution

Generally, numerical schemes need to represent a problem's solution by finitely many numbers, and then manipulate those numbers as true to the problem as possible. In finite volume methods we represent the solution as averages on cells which we update based on devised fluxes. In Smoothed Particle Hydrodynamics, we represent the fluid by a finite set of artificial *fluid particles* and update their positions and velocities according to the Lagrangian form of the fluid equations.

Idea of FEMs: In FEMs the solution domain is structured into finite elements with nodes on which base functions sit. The partial differential equation (or rather a weak form^a of it) turns into equations for the (finitely many) weights of those basis functions.

^aWeak here means, that the differential equation must not hold strictly locally, but for instance an integrated residual is to be zero, not the residual everywhere itself.

8.1 Finite element methods for linear PDEs

8.1.1 The solution is represented by weighted base functions on nodes within finite elements

The central ideas are

- **Division of space:** The space on which the solution sits is divided into smaller regions called elements (e.g. segments in 1D, rectangles in 2D, cubes, octahedra, ... in 3D). Every element contains a certain number of points called nodes.
- **Elementwise solution approximation:** We element-wise approximate the solution with a set of linearly independent (not necessarily orthogonal) basis functions evaluated on nodes.

On an element we could for instance use a polynomial basis

$$\text{a line } \phi(x) = a_0 + a_1x, \quad \text{or a parabula } \phi(x) = a_0 + a_1x + a_2x^2 \quad (361)$$

or Legendre polynomials.

Note: We need the same number of nodes as coefficients so that the values ϕ_i on the nodes fully specify our polynomial.

Better yet, we could use shape function N , so that our node values themselves are the coefficients we model.

So for n node values ϕ_1, \dots, ϕ_n (called *expansion coefficients*), we can write the solution on the k -th element as

$$\begin{aligned} \phi^{(k)}(x) &= \phi_1^{(k)} N_1^{(k)}(x) + \phi_2^{(k)} N_2^{(k)}(x) + \cdots + \phi_n^{(k)} N_n^{(k)}(x) \\ &\text{shape functions } N_i^{(k)}, \text{ zero outside k-th element} \end{aligned} \quad (362)$$

All elements use the same base function forms, but for each specific element the sum over its base functions is only non-zero in the respective element. We can therefore also write the total solution as

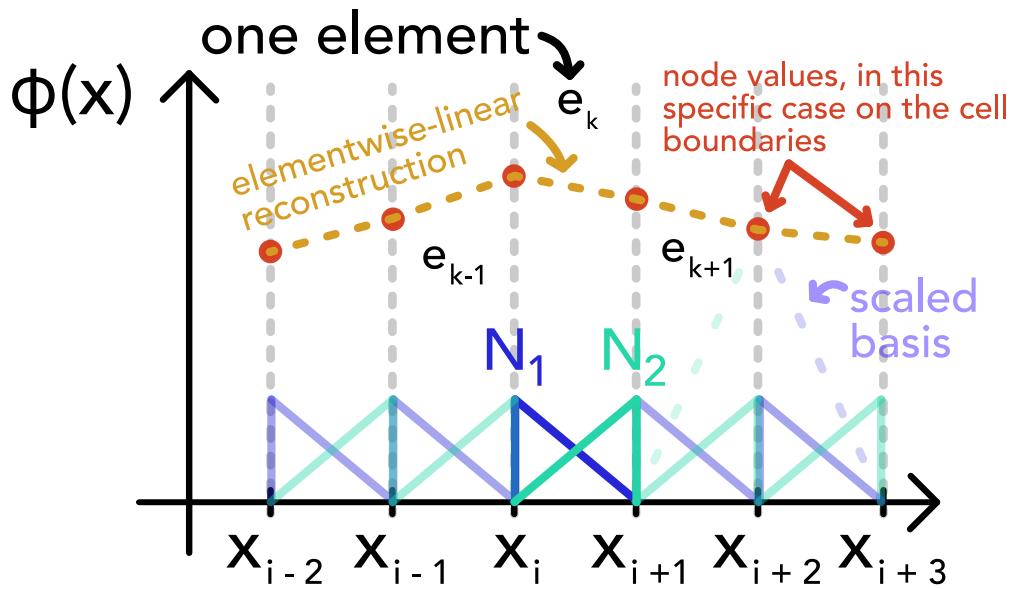
$$\phi(x) = \phi_1 N_1(x) + \phi_2 N_2(x) + \cdots + \phi_N N_N(x), \quad \text{in total } \mathcal{N} \text{ nodes} \quad (363)$$

Note: While we use 1D notation here, we can expand to $\phi(\underline{x}) = \sum_{i=1}^n \phi_i N_i(\underline{x})$. Time-dependency can be included as time-dependency of the weights aka expansion coefficients $\phi_i = \phi_i(t)$. But first we will consider constant coefficients, so a stationary (elliptic) problem.

Based on this the next step is turning the PDE into an algebraic equation for the expansion coefficients ϕ_i .

8.1.1.1 Example 1D linear reconstruction

A linear 1D example can be seen in figure 72.



when the x_i are used as nodes (so the boundaries of the elements) the reconstruction is continuous with the effective base functions being triangles over 2 cells

Figure 72: 1D linear reconstruction

8.1.2 From the PDE to an algebraic equation for the expansion coefficients ϕ_1, \dots, ϕ_n

For now consider a stationary example (no time dependence of ϕ_j in one spatial dimension.)

In the following we find an algebraic expression (here even a system of linear equations) usually done for the expansion coefficients on one element. Afterwards the equations for all elements have to be assembled (to an overall linear system) in a further step. If the system has \mathcal{N} total nodes, the final linear equation is described by an $\mathcal{N} \times \mathcal{N}$ system. See Lewis et al., 2004.

We might sometimes directly find such a system for all expansion coefficients and we will focus on that.

8.1.2.1 Inserting the finite element approximation into the PDE yields a residuum

Consider a linear PDE of the form

$$L\hat{\phi} + \hat{s} = 0, \quad \text{linear differential operator } \hat{L}, \text{ source term } \hat{s}, \text{ solution } \hat{\phi} \quad (364)$$

\hat{L} being linear means $\hat{L}(a\phi + b\psi) = a\hat{L}\phi + b\hat{L}\psi$ for any functions ϕ, ψ and constants a, b .

We now plug in the finite element approximation $\phi(x)$ and source function $s(x)$

$$L\phi + s = R^{(k)}(x; \phi_1, \dots, \phi_n) = L \left(\sum_{j=1}^n \phi_j N_j(x) \right) + s \underset{L \text{ linear}}{\sim} \sum_{j=1}^n \phi_j L N_j(x) + s \quad (365)$$

superscript k to indicate this is the residual over the k -th element

As only the shape functions depend on x , we only have to apply L to them. As of our approximation we have a generally non-zero residual.

8.1.2.2 Finding the expansion coefficients by minimizing the residual in some sense

We want to choose expansion coefficients minimizing the residual in some sense.

Ritz method: Here we require the integral over the residual to vanish

$$\int_{\text{domain}} R dx = 0 \quad (366)$$

where here we consider the whole problem domain, to readily find the whole system of linear equations (but this might also be just an element). Therefore

$$R = R(x; \{\phi_i\}_{i=1}^N) \quad (367)$$

Weighted residual method Compared to the Ritz method weighting functions $w_i(x)$ are introduced

$$\int_{\text{domain}} w_i(x) R(x; \{\phi_i\}_{i=1}^N) dx = 0, \quad i = 1, \dots, N \quad (368)$$

which leads to

- collocation method: Residuum is required to vanish at N points x_i inside the domain,
 $w_i = \delta(x - x_i)$
- least-square method: $w_i = \partial_{\phi_i} R \rightarrow \int_{\text{domain}} (\partial_{\phi_i} R) R dx = \frac{1}{2} \partial_{\phi_i} \int_{\text{domain}} R^2 dx = 0$
- Galerkin method: Choose basis functions themselves as weights, so $w_i(x) = N_i(x)$, so

$$\int_{\text{domain}} N_i(x) R(x; \{\phi_i\}_{i=1}^N) dx = 0, \quad i = 1, \dots, N \quad (369)$$

The general Galerkin principle is to multiply an equation by arbitrary test functions and describe the unknown field with the same set of basis functions.

8.1.2.3 A linear system for ϕ_1, \dots, ϕ_N in the Galerkin scheme

Based on

$$\begin{aligned} \forall i \in 1, \dots, \mathcal{N} : \\ 0 &= \int_{\text{domain}} N_i(x) \cdot R \left(\sum_j \phi_j N_j(x) \right) dx \\ &= \int_{\text{domain}} N_i(x) \cdot \left(L \left(\sum_j \phi_j N_j(x) \right) + s \right) dx \\ &\stackrel{L \text{ linear}}{=} \sum_j \phi_j \underbrace{\int_{\text{domain}} N_i(x) L(N_j(x)) dx}_{A_{ij}} - \underbrace{\int_{\text{domain}} -N_i(x) s dx}_{b_i} \\ &\rightarrow \sum_j \phi_j A_{ij} = b_i \end{aligned} \quad (370)$$

we can turn the PDE into a linear system for the expansion coefficients ϕ_i , compactly

$$\underline{\underline{A}} \underline{\phi} = \underline{b}, \quad \underline{\phi} = \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_N \end{pmatrix}, \quad \text{vector of source elements } \underline{b} \quad (371)$$

$\underline{\underline{A}}$ very sparse because of the localization of the expansion elements

FEMs are only good for linear PDEs (only if L is linear do we get a linear system for the coefficients) (sometimes non-linear PDEs can be linearized though)

Dynamical systems where the ϕ_i might change in time will follow shortly.

8.1.2.4 Example Application of Galerkin FEM

Consider the 1D Poisson equation (really an ODE)

$$\partial_x^2 \phi(x) = 4\pi G \rho(x), \quad \text{van Neumann boundary conditions } \partial_x \phi|_{x_L} = \partial_x \phi|_{x_R} = 0 \quad (372)$$

Aim: From a given density distribution ρ we want to find the field ϕ .

We formulate the basis functions as triangular basis functions spanning two elements, as shown in figure 72. In this figure you can also see that one can formulate this in terms of basis functions only sitting on one element, which is more true to the previous introduction.

$$S_i(x) = \begin{cases} \frac{x-x_{i-1}}{\Delta x} & \text{for } x \in [x_{i-1}, x_i] \\ \frac{x_{i+1}-x}{\Delta x} & \text{for } x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise.} \end{cases} \quad (373)$$

We have linear elements between \mathcal{N} points, Δx apart.

The finite element approximation is

$$\phi = \sum \phi_i S_i(x) \quad (374)$$

we calculate the residual as

$$R = \partial_x^2 \phi(x) - 4\pi G \rho(x) \quad (375)$$

so using the Galerkin weighting we get

$$\begin{aligned} \forall i = 1, \dots, \mathcal{N} : \int_{x_L}^{x_R} S_i(x) (\partial_x^2 \phi(x) - 4\pi G \rho(x)) dx &= 0 \\ \rightarrow \int_{x_L}^{x_R} S_i(x) \partial_x^2 \phi(x) dx &= \underbrace{\int_{x_L}^{x_R} S_i(x) 4\pi G \rho(x) dx}_{:= -b_i \text{ as } \rho, S_i \text{ known}} \end{aligned} \quad (376)$$

The LHS can be rewritten by integration over parts (mind $\partial_x \phi|_{x_L} = \partial_x \phi|_{x_R} = 0$)

$$\int_{x_L}^{x_R} S_i(x) \partial_x^2 \phi(x) dx = - \int_{x_L}^{x_R} \partial_x S_i(x) \partial_x \phi(x) dx \quad (377)$$

so in total we get the linear equation

$$\int_{x_L}^{x_R} \partial_x S_i(x) \partial_x \sum \phi_j S_j(x) dx = \sum \phi_j \underbrace{\int_{x_L}^{x_R} \partial_x S_i(x) \partial_x S_j(x) dx}_{A_{ij}} = \sum \phi_j A_{ij} = b_i \quad (378)$$

we retrieve a linear equation with (use the definition of $S_i(x)$)

$$A_{ij} = \begin{cases} \frac{2}{\Delta x} & \text{for } i = j \\ -\frac{1}{\Delta x} & \text{for } i = j \pm 1 \\ 0 & \text{otherwise} \end{cases} \quad (379)$$

which would also follow from

$$\partial_x^2 \phi_i = \frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2} \quad (380)$$

8.2 Discontinuous Galerkin Method

Aim: We want to find the solution to a hyperbolic differential equation system. The solution is often characterized by discontinuities and shocks. Me might be interested in complex geometries, for which Finite Element Methods are very suitable.

Problem: Usual Finite Element Methods (FEMs) are piecewise polynomial and continuous - shocks are often smeared out.

Idea: Combine the advantages of Finite Element Methods and Finite Volume Schemes. Discontinuous Galerkin is a FEM - the problem domain is subdivided into a grid of a finite number of elements. We use a piecewise polynomial solution which can be discontinuous across cell interfaces, where we use the methods from finite volume to compute the intercell fluxes - so conservation laws are baked in.

Continuous and discontinuous Galerkin are illustrated in figure 73.

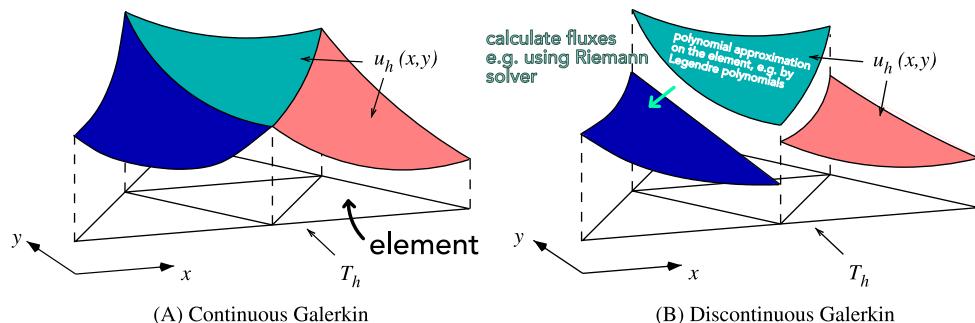


Figure 73: Continuous and discontinuous Galerkin

8.2.1 Problem we want to tackle | Euler equations

Consider the 3D formulation of the Euler equations we know from dimensional splitting

$$\partial_t \underline{u}(\underline{x}) + \sum_{\alpha=1}^3 \partial_{x_\alpha} f_\alpha(\underline{u}) = 0, \quad \text{state vector } \underline{u} = \begin{pmatrix} \rho \\ \rho \underline{v} \\ \rho e \end{pmatrix} \quad (381)$$

specific energy $e = \rho e_{th} + \frac{1}{2} \rho \underline{v}^2$ with specific internal energy e_{th}

with flux vectors

$$\underline{f}_1 = \begin{pmatrix} \rho v_1 \\ \rho v_1^2 + P \\ \rho v_1 v_2 \\ \rho v_1 v_3 \\ v_1(\rho e + P) \end{pmatrix}, \quad \underline{f}_2 = \begin{pmatrix} \rho v_2 \\ p v_1 v_2 \\ \rho v_2^2 + P \\ \rho v_2 v_3 \\ v_2(\rho e + P) \end{pmatrix}, \quad \underline{f}_3 = \begin{pmatrix} \rho v_3 \\ p v_1 v_3 \\ \rho v_2 v_3 \\ \rho v_3^2 + P \\ v_3(\rho e + P) \end{pmatrix} \quad (382)$$

ideal gas closure $P = \rho e_{th}(\gamma - 1)$

Aim: From a given initial state $\underline{u}(\underline{x}, t = 0) = u(\underline{x}, 0)$ we want to find the subsequent evolution of the fluid.

8.2.2 Steps in formulating the Dicontinuous Galerkin (DG) scheme

1. Subdivide the space into elements aka cells
2. Represent the fluid state on a cell using a polynomial basis (e.g. Legendre polynomials) with weights evolving in time; *nodal* vs *modal*
3. Find a general formula for the weights in the *modal* variant
4. Find the initial weights from specific *nodal* starting values in the *modal* scheme
5. Find an evolution equation for the weights

8.2.3 Subdivision and Representation | modal vs nodal

The fluid state is represented as a polynomial approximation (for the respective fluid variables) on the element (non overlapping elements with discontinuities in-between) - but how?

A typical DG cell is illustrated in figure 74.

There are two possible representations of the solution

- **nodal:** We store and operate on fluid state vectors at chosen positions within the cell. In the Legrange interpolation with Lagrange polynomials of degree k , so $l_j(x) = \prod_{\substack{0 \leq m \leq k \\ m \neq j}} \frac{x - x_m}{x_j - x_m}$, for one fluid variable the node values are also the expansion coefficients, $u = \sum_{j=0}^{N(k)} u_j l_j(x)$. The positions of the nodes in the cell are chosen smartly regarding quadrature (integration) rules.
- **modal:** We store and operate on weights of usually orthogonal polynomials (usually

one cell

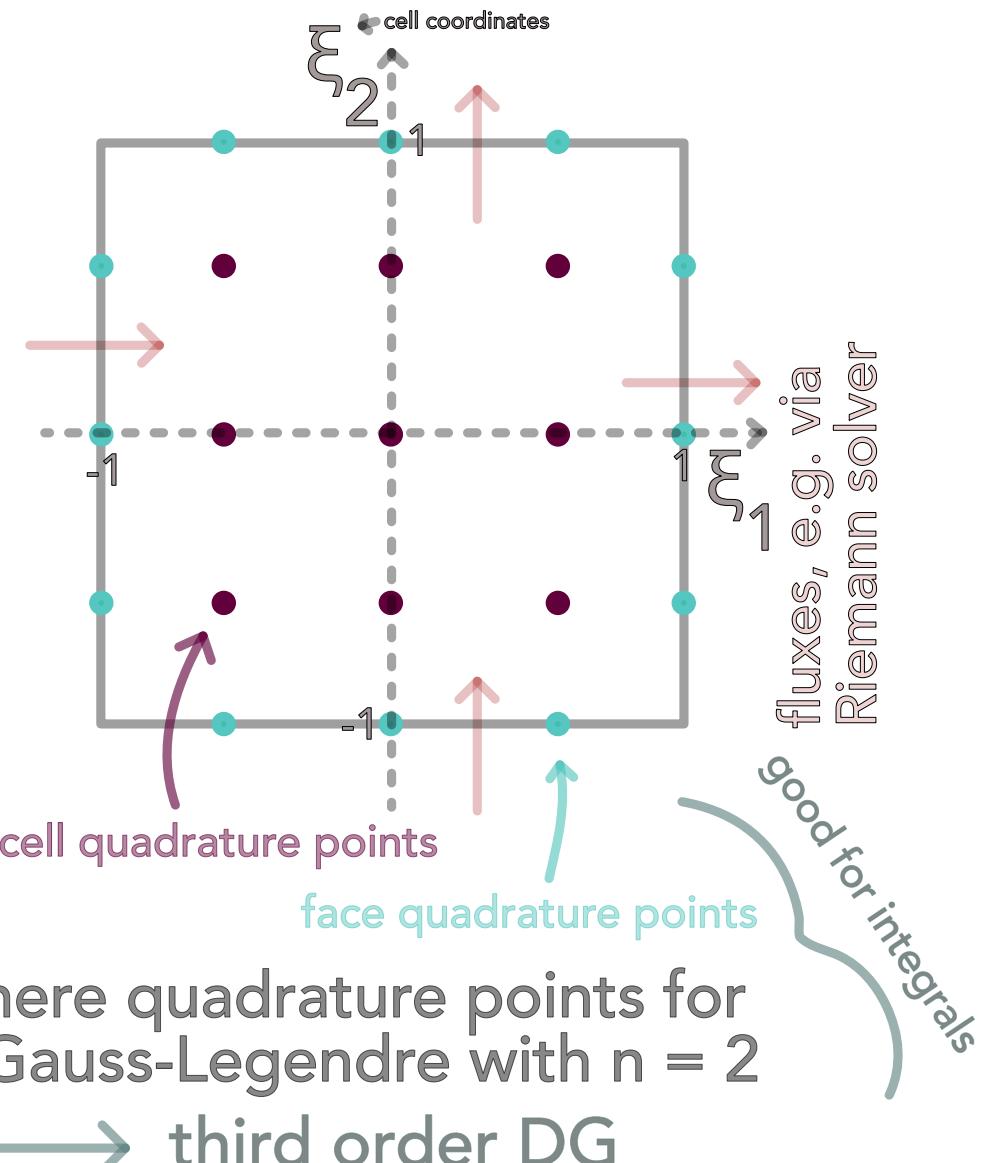


Figure 74: Typical DG cell

Legendre). Integrals are still evaluated using quadrature rules. Initial weights have to be calculated based on nodal values.

We opt for **modal**.

The solution in the interior of cell K is given by a linear combination of $N(k)$ orthogonal and normalized basis functions $\phi_l^{(K)}$ (maximum degree of the basis functions is k).

$$\underline{u}^{(K)}(\underline{x}, t) = \sum_{l=1}^{N(k)} \omega_l^{(K)}(t) \phi_l^{(K)}(\underline{x}) \quad (383)$$

where

- the space-time dependence of the fluid state was split into time dependent weights and space dependent basis functions
- the cell's state is completely given by the $N(k)$ weights

8.2.3.1 Example for an orthogonal polynomial basis: Legendre polynomials

We can combine Legendre polynomials to 3D base functions with degree up to k in the following manner

$$\begin{aligned} \{\phi_l(\xi)\}_{l=1}^{N(k)} &= \left\{ \tilde{P}_u(\xi_1) \tilde{P}_v(\xi_2) \tilde{P}_w(\xi_3) \mid u, v, w \in \mathbb{N}_0 \wedge u + v + w \leq k \right\} \\ &\text{scaled Legendre polynomials } \tilde{P}_n(\xi) = \sqrt{2n+1} P_n(\xi) \\ &\text{"special orthog. " } \int_{-1}^1 P_i(\xi) P_j(\xi) d\xi = \begin{cases} 0 & \text{if } i \neq j \\ 2 & \text{if } i = j \end{cases} \end{aligned} \quad (384)$$

For a polynomial basis with maximum degree k we have

$$N(k) = \sum_{u=0}^k \sum_{v=0}^{k-u} \sum_{w=0}^{k-u-v} 1 \quad (385)$$

basis polynomials.

The first Legendre polynomials are

$$P_0(\xi) = 1, P_1(\xi) = \xi, P_2(\xi) = \frac{1}{2} (3\xi^2 - 1) \quad (386)$$

Note: The span of $P_0(x), P_1(x), \dots, P_n(x)$ is the same as of $1, x, x^2, \dots, x^n$. So

$$\forall \text{polynomial } P \text{ of degree } < n : \int_{-1}^1 P(x) P_n(x) dx = 0 \quad (387)$$

8.2.4 Solving for the weights

Note: The main application of finding weights is finding the weights of the initial state. Going from weights to evaluations is easy.

Let's say we would know $\underline{u}^{(K)}(\underline{x}, t)$ - then we can determine the weights based on the orthogonality and normalization properties of the basis functions as

$$\underline{\omega}_j^K(t) = \frac{1}{|K|} \int_K \underline{u}^{(K)} \phi_j^{(K)} dV, \quad j = 1, \dots, N(k) \quad (388)$$

with $|K|$ volume of cell K

We hereby choose $\phi_1^{(K)} = 1$ so that $\omega_1^{(K)}$ is the cell's average of the state vector $\underline{u}^{(K)}$ (constant term). $\phi_j^{(K)}, j \geq 2$ are higher order basis functions, with $w_j^{(K)}$ being the higher order moments of the state vector $\underline{u}^{(K)}$.

Scaled variable on cell: On a cube, we can define the basis functions in terms of a scaled variable $\underline{\xi}$.

$$\phi_l(\underline{\xi}) : [-1, 1]^3 \rightarrow \mathbb{R}, \quad \underline{\xi} = \frac{2}{\Delta x} (\underline{x} - \underline{x}^{(K)})$$

cell's center $\underline{x}^{(K)}$, cell's sidelength Δx (389)

Idea: Let us approximate this integral by a quadrature rule.

First write the integral equation for the weights in the reference frame of a cubic cell with sidelength 2

$$\underline{\omega}_j^{(K)}(t) = \frac{1}{8} \int_{[-1,1]^3} \underline{u}^{(K)}(\underline{\xi}, t) \phi_j^{(K)}(\underline{\xi}) d^3 \underline{\xi}, \quad j = 1, \dots, N(k) \quad (390)$$

We then apply Gauss-Legendre quadrature with $(k+1)^3$ quadrature points, so

$$\underline{\omega}_j^{(K)}(t) \approx \frac{1}{8} \sum_{q=1}^{(k+1)^3} \underline{u}^{(K)}(\underline{\xi}_q^{3D}, t) \phi_j^{(K)}(\underline{\xi}_q^{3D}) w_q^{3D}, \quad j = 1, \dots, N(k)$$

positions of quadrature nodes in cell's reference frame $\underline{\xi}_q^{3D}$, quadrature weights w_q^{3D} (391)

8.2.4.1 What even is Gauss-Legendre quadrature?*

It is intuitive that an integral can be approximated by the mean of evaluation points. However, it turns out, that one can exactly integrate polynomials of degree $2n - 1$ with n smart evaluation points and weights. One such method is called Gauss-Legendre quadrature.

Claim: The approximation

$$\int_{-1}^1 f(\xi) d\xi \approx \sum_{q=1}^n f(\xi_q^{1D}) w_q^{1D} \quad (392)$$

roots ξ_q^{1D} of $P_n(\xi)$, weights $w_q^{1D} = \frac{2}{\left(1 - (\xi_q^{1D})^2\right) (P'_n(\xi_q^{1D}))^2}$

for $f : [-1, 1] \rightarrow \mathbb{R}$ is exact for polynomials of degree $2n - 1$.

Proof: Let $f = P(x)$ have degree $\leq 2n - 1$. Then we can write $P(x) = Q(x)P_{n+1}(x) + R(x)$ with $Q(x), R(x)$ polynomials of degree $\leq n$ ($P_{n+1}(x)$ is the $(n+1)$ -th Legendre polynomial). Then

$$\begin{aligned} \int_{-1}^1 P(x) dx &= \underbrace{\int_{-1}^1 Q(x)P_{n+1}(x) dx}_{=0 \text{ as } Q \text{ can be written in base } \{P_0, \dots, P_n\} \perp P_{n+1}} + \int_{-1}^1 R(x) dx \\ &= \int_{-1}^1 R dx \\ &= \underbrace{\sum_{q=1}^n R(\xi_q^{1D}) w_q^{1D}}_{\text{exact as } R \text{ is a polynomial of degree } \leq n} \\ &= \sum_{q=1}^n \left(R(\xi_q^{1D}) + \underbrace{P_{n+1}(\xi_q^{1D}) \cdot Q(\xi_q^{1D})}_{=0 \text{ as } \xi_q^{1D} \text{ roots of } P_{n+1}} \right) w_q^{1D} = \sum_{q=1}^n P(\xi_q^{1D}) w_q^{1D} \end{aligned} \quad (393)$$

Formulation in higher dimensions: We generalize to $f : [-1, 1]^2 \rightarrow \mathbb{R}$ by

$$\int_{-1}^1 \int_{-1}^1 f(\xi_1, \xi_2) d\xi_1 d\xi_2 \approx \sum_{q=1}^n \sum_{r=1}^n f(\xi_q^{1D}, \xi_r^{1D}) w_q^{1D} w_r^{1D} = \sum_{q=1}^{n^2} f(\xi_q^{2D}) w_q^{2D} \quad (394)$$

8.2.5 Finding initial weights - just apply the determination of weights to the initial state

The initial state $\underline{u}(\underline{x}, t=0)$ is best represented by weights $\underline{\omega}_j^{(K)}(t=0)$, such that

$$w_{l,i}^{(K)}(t=0) = \underset{\underline{\omega}_{j,i}^{(K)}(t=0)}{\operatorname{argmin}} \int_K \left(u_i^{(K)}(\underline{x}, t=0) - u_i(\underline{x}, t=0) \right)^2 d^3 \underline{x}, \quad i \text{ over state vector components} \quad (395)$$

which just leads us to the previous projection (eq. 388) and solution in eq. 391 at $t = 0$, where our initial state must be known on the quadrature nodes.

8.2.6 Evolution equation for the weights

We derive a DG scheme on cell K .

Weak form of the Euler equations: A weak formulation of the Euler equations for the polynomial approximation $\underline{u}^{(K)}$ on cell K is found by multiplying the Euler equations with the basis function $\phi_j^{(K)}$ and integrating over the cell K .

$$\int_K \left[\partial_t \underline{u}^{(K)} + \sum_{\alpha=1}^3 \partial_{x_\alpha} f_\alpha \right] \phi_j^{(K)} dV = 0 \quad (396)$$

Integrating by parts and applying Gauss theorem, we get

$\frac{d}{dt} \underbrace{\int_K \underline{u}^{(K)} \phi_j^K dV}_{\underline{w}_j^{(K)} \cdot K }$	$+ \sum_{\alpha=1}^3 \underbrace{\int_{\partial K} f_\alpha n_\alpha \phi_j^K dS}_{\begin{array}{l} \text{evaluate using} \\ \text{Gauss-Quad,} \\ \text{unknown flux} \\ \text{across discont.} \\ \text{via Riemann} \\ \text{solver} \end{array}}$	$- \sum_{\alpha=1}^3 \underbrace{\int_K f_\alpha \frac{\partial \phi_j^K}{\partial x_\alpha} dV}_{\begin{array}{l} \text{evaluate via} \\ \text{Gauss-Quad.,} \\ \text{interior flux} \\ \text{known from} \\ \text{state variable} \\ \text{approx.} \end{array}} = 0$
--	---	---

$\text{normal vector } \underline{n} = \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} \text{ on } \partial K$

(397)

- we have found a system of coupled ODEs for the weights which can be solved for instance using RK schemes. Remember to transform to the cell coordinates $\left(d\xi^3 = \left(\frac{2}{\Delta x^{(K)}} \right)^2 dV \right)$ (from eq. 389).

Note on strong shocks: In case there are strong shock waves, limiting-schemes damping oscillations (e.g. by setting higher expansion coefficients in cells adjacent to the detected discontinuity to zero) are used.

8.2.7 Efficiency of DG and refinement schemes

Accuracy can be increased by

- p-refinement: higher order scheme (polynomials up to higher order degree in the basis set)
- h-refinement: finer grid with smaller spacing

In both cases, the number of weights increases. For isentropic vortex flow, one finds that p-refinement is much more efficient (better solution at less CPU time) than h-refinement.

more on nodal vs modal, more details from script

9 Diffusion

On a micro scale diffusion is a random walk with the mean free path as the step size. Intuitive example: Imagine a region with high density. It is more likely that a particle within this region takes a step out of it than a particle outside stepping in (as of the difference in density). Physically, the entropy cannot decrease.

9.1 Thermodynamic Basics of Diffusion

How quickly do thermalized, randomly moving particles spread?

9.1.1 Mean squared velocity and mean squared relative velocity

Assume the particles follow a Maxwell-Boltzmann distribution (we assume only one particle species with mass m and temperature T)

$$f(\underline{v})d^3v = \left(\frac{m}{2\pi k_B T}\right)^{\frac{3}{2}} \exp\left(\frac{-\frac{1}{2}mv^2}{k_B T}\right) d^3v \quad (398)$$

$$f(v)dv = 4\pi v^2 \left(\frac{m}{2\pi k_B T}\right)^{\frac{3}{2}} \exp\left(\frac{-\frac{1}{2}mv^2}{k_B T}\right) dv, \quad v = |\underline{v}|$$

Based on the equipartition theorem, the mean squared velocity is

$$\langle \frac{1}{2}m\underline{v}^2 \rangle = \frac{3}{2}k_B T \quad \rightarrow \quad \langle \underline{v}^2 \rangle = \frac{3k_B T}{m}, \quad v := \sqrt{\langle \underline{v}^2 \rangle} \quad (399)$$

so the mean squared relative velocity is higher (avg here over velocity space)

$$\langle \underline{v}_{rel}^2 \rangle = \langle (\underline{v} - \underline{v}')^2 \rangle = \langle \underline{v}^2 \rangle + \langle \underline{v}'^2 \rangle - \underbrace{2\langle \underline{v} \cdot \underline{v}' \rangle}_{=0 \text{ avg. over uncorrelated}} = \langle \underline{v}^2 \rangle = \frac{6k_B T}{m} \quad (400)$$

where we shorthand use $v_{rel} = \sqrt{\langle \underline{v}_{rel}^2 \rangle}$.

9.1.2 Mean free path and relaxation time

Consider a fluid with particle density n (in m^{-3}) of particles with effective radius r so effective cross-section $\sigma = \pi r^2$. In sec. 5.1.2.1, we already devised (under particle movement with

the root-mean-square velocity)

$$\begin{aligned} \text{particle moving through stationary particles: } \lambda_{mfp} &= \frac{1}{n\sigma} \\ \text{relative particle movement: } \lambda_{mfp} &= \frac{1}{\sqrt{2}n\sigma} \end{aligned} \quad (401)$$

and the relaxation time (mean time until a particle collides with another)

$$t_{rel} = \frac{\lambda_{mfp}}{v} = \frac{1}{n\pi r^2} \sqrt{\frac{m}{6k_B T}} \quad (402)$$

Springel et al., 2023 - I would say wrongly - uses $\lambda_{mfp} = \frac{1}{n\sigma}$ and with this defines $t_{scat} = \frac{\lambda_{mfp}}{v}$ and $t_{rel} = \frac{\lambda_{mfp}}{v_{rel}}$, which is equivalent to our t_{rel} with our λ_{mfp} .

9.1.3 Random Walk | spreading Gaussian distribution in space

Central Limit Theorem: Let $w(s)ds$ be the probability of taking a step with length between s and $s+ds$ with mean $\langle s \rangle$ and variance σ_s^2 . We make N steps, so the final position along one dimension is

$$x = s_1 + s_2 + s_3 + \cdots + s_N \quad (403)$$

By the rules of the mean and variance, one finds

$$\begin{aligned} \langle x \rangle &= N\langle s \rangle, \quad \langle s \rangle = \int_{-\infty}^{\infty} s \cdot w(s) ds \\ (\Delta x)^2 &= \sigma_x^2 = \langle (x - \langle x \rangle)^2 \rangle = N\sigma_s^2 \end{aligned} \quad (404)$$

And the **Central Limit Theorem** states

$$P(x) = \frac{1}{\sqrt{2\pi}\Delta x} \exp\left(-\frac{(x - \langle x \rangle)^2}{2(\Delta x)^2}\right) \quad (405)$$

if the moments $\langle s^n \rangle = \int_{-\infty}^{\infty} s^n w(s) ds$ are finite. The basis reasoning is that one can show that for added random variables their sum converges to one distribution, and as added random variables are distributed according to the convolution of their distributions and the convolution of two Gaussians is a Gaussian, the sum of many random variables is a Gaussian.

Consider particles starting out at the origin with equal probability of going left or right with λ_{mfp} . We have

$$w(s) = \frac{1}{2}\delta(s - \lambda_{mfp}) + \frac{1}{2}\delta(s + \lambda_{mfp}), \quad \langle s \rangle = 0, \quad \sigma_s^2 = \lambda_{mfp}^2 \quad (406)$$

so therefore, estimating $N = \frac{t}{t_{rel}}$

$$\langle x \rangle = 0, \quad \langle x^2 \rangle = N\sigma_s^2 = \frac{\lambda_{mfp}^2}{t_{rel}} t \rightarrow \boxed{\sqrt{\langle x^2 \rangle} \propto \sqrt{t}} \quad (407)$$

So the distribution flattens and spreads in time. As $\langle (\Delta x)^2 \rangle$ is time dependent, so is $p(x, t)$.

9.2 Diffusion equation

9.2.1 Derivation of the diffusion equation | Fick's law from the microscopic consideration

Consider the evolution of a system from $t \rightarrow t + \Delta t$. For a change Δt let Δx be the according change of a particle position, distributed according to $p(\Delta x, \Delta t)$. We can therefore write

$$\begin{aligned} n(x, t + \Delta t) &= \langle n(x - \Delta x, t) \rangle \underset{\text{Taylor}}{=} n(x, t) - \partial_x n \cdot \langle \Delta x \rangle + \frac{1}{2} \partial_x^2 n \cdot \langle \Delta x^2 \rangle + \dots \\ &\underset{\text{vanishing uneven moments}}{=} n(x, t) + \frac{1}{2} \partial_x^2 n \cdot \langle \Delta x^2 \rangle + \mathcal{O}(\Delta x^4) \end{aligned} \quad (408)$$

Taylor expansion with respect to time yields

$$n(x, t + \Delta t) = n(x, t) + \partial_t n \cdot \Delta t \quad (409)$$

So combined, we get (ignoring the higher order terms, so it's really an approximation)

$$\boxed{\partial_t n(x, t) = \frac{\Delta x^2}{2\Delta t} \partial_x^2 n(x, t) = D \partial_x^2 n(x, t)} \quad (410)$$

with

$$\text{diffusion coefficient } D = \frac{\Delta x^2}{2\Delta t} \propto \frac{\lambda_{mfp}^2}{t_{rel}} \quad (411)$$

We can rewrite this as a conservation equation (introducing the flux J) (Fick's law of diffusion)

$$\partial_t n(x, t) = -\partial_x J(x, t), \quad J(x, t) = -D \partial_x n(x, t) \quad (412)$$

where the same principle applies to heat, energy and momentum diffusion.

9.2.2 Analytical Solution to the diffusion equation via Fourier transform

Using the Fourier transform in space in the convention

$$n(x, t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{n}(k, t) \exp(-ikx) dk, \quad \hat{n}(k, t) = \int_{-\infty}^{\infty} n(x, t) \exp(ikx) dx \quad (413)$$

we yield an ODE in Fourier space with trivial solution

$$\partial_t \hat{n}(k, t) = -Dk^2 \hat{n}(k, t) \rightarrow \hat{n}(k, t) = \hat{n}(k, t_0) \exp(-Dk^2(t - t_0)) \quad (414)$$

9.2.2.1 Solution for an initial delta peak in the density over space

Consider

$$n(x, t_0) = n_0 \delta(x - x_0), \quad \hat{n}(x, t_0) = n_0 \int_{-\infty}^{\infty} \delta(x - x_0) \exp(ikx) dx = n_0 \exp(ikx_0) \quad (415)$$

yielding the solution in Fourier space

$$\rightarrow \hat{n}(k, t) = \hat{n}(k, t_0) \exp(-Dk^2(t - t_0)) = n_0 \exp(ikx_0) \exp(-Dk^2(t - t_0)) \quad (416)$$

so with the inverse Fourier transform (completing the square and using that a Gaussian distribution is normed)

$$\text{a Gaussian } n(x, t) = \frac{n_0}{\sqrt{4\pi D(t - t_0)}} \exp\left(-\frac{(x - x_0)^2}{4D(t - t_0)}\right) \quad (417)$$

Conservation: As the Gaussian is normalized, n_0 stays constant irrespective of time implying particle number conservation (or energy conservation for the diffusion of heat (heat conduction))

Problem of particle propagation: As the solution is a Gaussian with infinite wings, there must have been infinitely fast particle transport starting from the initial situation - unphysical. **Reason - approximation in diffusion equation:** Our diffusion equation is based on a random walk with well-behaved steps but in finding the diffusion equation we have done a truncated Taylor expansion, leaving away $\mathcal{O}(\Delta x^4)$. In numerical schemes we will use limited flux.

Infinite Signal Speeds: In contrast to advection with a given signal speed the diffusion equation and also Poisson equation (e.g. for Gravity) have infinite domains of influence - an event at (x, t) depends on the full domain. In the case of diffusion we will later introduce limited flux aka tempered diffusion.

9.3 Numerical solutions

Let us change the notation $n \rightarrow u$ to be consistent with previously introduced schemes. We indicate with the convention $u_{\text{space}}^{(\text{time})}$.

In the following we discuss multiple possible discretizations.

The general grid in space and time is illustrated in figure 75.

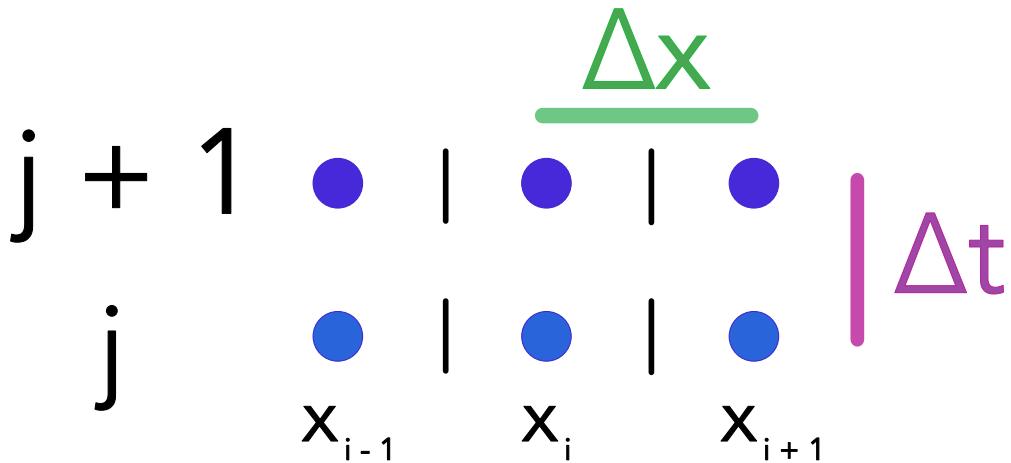


Figure 75: Grid in space and time for the diffusion equation.

9.3.1 Forward in time, central in space

9.3.1.1 Discretized diffusion equation

Based on the central approximation of $\partial_x^2 u$ (from Taylor in both directions, see eq. 248) at the current time j and forward difference in time, we get

$$\frac{u_i^{(j+1)} - u_i^{(j)}}{\Delta t} = D \frac{u_{i+1}^{(j)} - 2u_i^{(j)} + u_{i-1}^{(j)}}{\Delta x^2} \quad (418)$$

9.3.1.2 Explicit scheme for performing a time step

Defining $\alpha = \frac{D\Delta t}{\Delta x^2}$, we get

$$u_i^{(j+1)} = u_i^{(j)} + \alpha \left(u_{i+1}^{(j)} - 2u_i^{(j)} + u_{i-1}^{(j)} \right) \quad (419)$$

an explicit scheme for forwarding. The information flow is illustrated in figure 76.

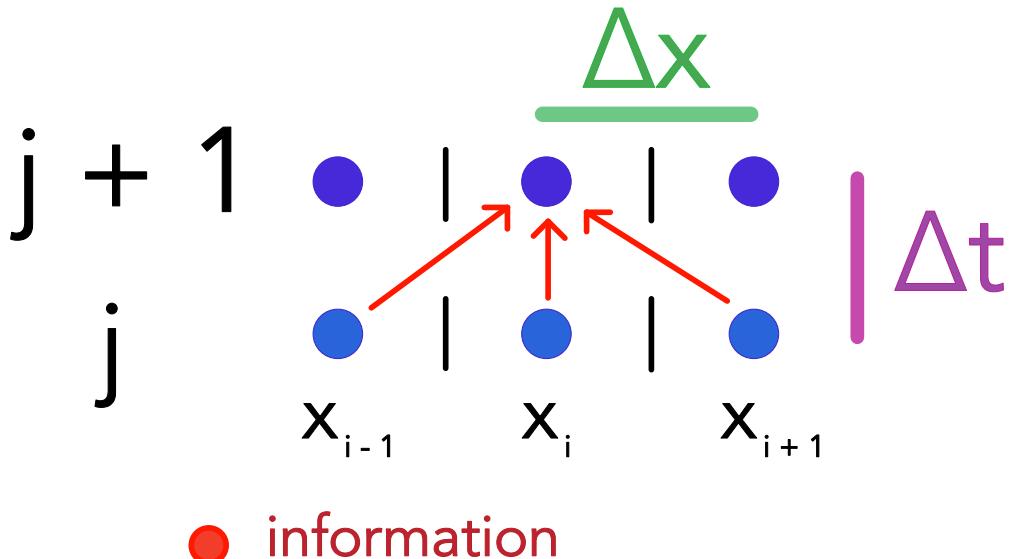


Figure 76: Information flow for the central in space, forward in time scheme.

9.3.1.3 Stability of the central in space, forward in time scheme for diffusion

Let us make a simple stability analysis considering $u_{i+1}^{(j)} = u_{i-1}^{(j)} = 0$. In such a scenario it does not make sense for there to be a sign change from $u_i^{(j)}$ to $u_i^{(j+1)}$. This requires (just plug conditions in)

$$\alpha \leq \frac{1}{2} \leftrightarrow \Delta t \leq \frac{1}{2} \frac{\Delta x^2}{D} \quad (420)$$

Physical intuition and connection to CFL: In diffusion, density spreads with $\Delta x = \sqrt{2D\Delta t}$, so when we impose a CFL criterion - we only use information from the left and right grid points next to the grid point of interest, so Δt must not be so large, that further information would in reality be necessary - we get the same criterion as above.

Problem of scaling the resolution: If we want to double the spatial resolution, we must quadruple the resolution in time, i. e. in total the 1D simulation is 8x more expensive (in 3D double spatial resolution means 8x more points in a certain volume so 32x more cost in total).

9.3.2 Backward in time, central in space

9.3.2.1 Discretized implicit scheme

We now evaluate the spatial derivative at time $j + 1$ instead of j .

$$\frac{u_i^{(j+1)} - u_i^{(j)}}{\Delta t} = D \frac{u_{i+1}^{(j+1)} - 2u_i^{(j+1)} + u_{i-1}^{(j+1)}}{\Delta x^2} \quad (421)$$

This can be rewritten as

$$-\alpha u_{i+1}^{(j+1)} + (1 + 2\alpha)u_i^{(j+1)} - \alpha u_{i-1}^{(j+1)} = u_i^{(j)} \quad (422)$$

- a linear system of equations for $u_i^{(j+1)}$.

9.3.2.2 Matrix equation

The exact matrix depends on the boundary conditions we choose. For periodic boundary conditions¹⁸ we get

$$\underline{\underline{A}} \underline{\underline{u}}^{(j+1)} = \underline{\underline{u}}^{(j)}, \quad \underline{\underline{u}}^{(j)} = \begin{pmatrix} u_1^{(j)} \\ u_2^{(j)} \\ \vdots \\ u_N^{(j)} \end{pmatrix}$$

$$\underline{\underline{A}} = \begin{pmatrix} 1 + 2\alpha & -\alpha & 0 & \dots & 0 & -\alpha \\ -\alpha & 1 + 2\alpha & -\alpha & 0 & \dots & \\ 0 & -\alpha & 1 + 2\alpha & -\alpha & 0 & \dots \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ -\alpha & 0 & \dots & & 0 & -\alpha & 1 + 2\alpha \end{pmatrix} \quad (423)$$

which we can solve for $\underline{\underline{u}}^{(j+1)} = \underline{\underline{A}}^{-1} \underline{\underline{u}}^{(j)}$.

Problem: This is unconditionally stable but only first order in time $\mathcal{O}(\Delta x^2, \Delta t)$.

9.3.3 Crank-Nicolson method

Idea: Evaluate the spatial derivative at $j + \frac{1}{2}$, constructed as an average of j and $j + 1$.

¹⁸The point before the first point is the last point and the point after the last point is the first point.

$$\frac{u_i^{(j+1)} - u_i^{(j)}}{\Delta t} = D \frac{u_{i+1}^{(j+\frac{1}{2})} - 2u_i^{(j+\frac{1}{2})} + u_{i-1}^{(j+\frac{1}{2})}}{\Delta x^2}, \quad \text{use avg. } u_i^{(j+\frac{1}{2})} = \frac{u_i^{(j+1)} + u_i^{(j)}}{2} \quad (424)$$

yielding

$$-\alpha u_{i+1}^{(j+1)} + (1 + 2\alpha)u_i^{(j+1)} - \alpha u_{i-1}^{(j+1)} = \alpha u_{i+1}^{(j)} + (1 - 2\alpha)u_i^{(j)} + \alpha u_{i-1}^{(j)} \quad (425)$$

So again a matrix problem over the whole spatial domain

$$\underline{\underline{A}} \underline{u}^{(j+1)} = \underline{b}, \quad \underline{\underline{A}} \text{ as before, } b_i = \alpha u_{i+1}^{(j)} + (1 - 2\alpha)u_i^{(j)} + \alpha u_{i-1}^{(j)} \quad (426)$$

where again we can solve for $\underline{u}^{(j+1)} = \underline{\underline{A}}^{-1} \underline{b}$. Note for non-periodic boundary conditions e.g. Dirichlet boundaries (fixed), $\underline{\underline{A}}$ is tridiagonal, so very easy to solve.

Advantage of Crank-Nicolson: $\mathcal{O}(\Delta x^2, \Delta t^2)$ (2nd order in space and time) and unconditionally stable.

9.4 Flux-limited diffusion (*tempered*)

As of the approximation in the diffusion equation infinitely large signal speeds (unphysical) can result. Instead of including higher order terms in the Taylor expansion (\rightarrow hyperbolic system with finite information speed) we limit the speed by hand using a limited diffusion flux.

Idea: Express the flux as a product of density and velocity and limit this velocity akin to Special Relativity (based on a momentum consideration), e.g. to the speed of sound.

Let us start with the diffusion equation

$$\partial_t u = -\partial_x J, \quad J = -D \partial_x u \quad (427)$$

We make the Ansatz

$$J(x, t) = u(x, t) \cdot v, \quad \text{characteristic velocity } v \quad (428)$$

and define a *negative specific diffusion momentum*, really just the negative of the not-yet-limited velocity v .

$$-v = -\frac{J}{u} = \frac{D \partial_x u}{u} \equiv R \quad (429)$$

Now we ask ourselves: What if R was a relativistic momentum? What velocity \tilde{v} would have brought forth this relativistic momentum will naturally be limited to be smaller than c . So drawing from $p = \gamma mv, v < c$ we write

$$R = -\frac{\tilde{v}}{\sqrt{1 - \frac{\tilde{v}^2}{c^2}}} \leftrightarrow \tilde{v} = -\frac{R}{\sqrt{1 + \frac{R^2}{c^2}}} \quad (430)$$

so we use the altered flux

$$\tilde{J} = u\tilde{v}, \quad \partial_t u = \partial_x \tilde{J} \quad (431)$$

9.5 Diffusion in three dimensions

In 3D, we have

$$\partial_t u(\underline{x}, t) = -\underline{\nabla} \cdot \underline{J}(\underline{x}, t), \quad \text{flux vector } \underline{J}(\underline{x}, t) = -\underline{\underline{D}} \cdot \underline{\nabla} u(\underline{x}, t), \quad \text{diffusion matrix } \underline{\underline{D}} \quad (432)$$

For a magnetized plasma, movement is mostly along the magnetic field lines (diffusion along the lines is dampened by collisions, diffusion perpendicular can only exist as of collisions).

$$\underline{\underline{B}} = B \hat{\underline{e}}_z \rightarrow \underline{\underline{D}} = \begin{pmatrix} D_{\perp} & 0 & 0 \\ 0 & D_{\perp} & 0 \\ 0 & 0 & D_{\parallel} \end{pmatrix} \quad (433)$$

typically in MHD: $D_{\perp} \ll D_{\parallel} \rightarrow \partial_t \underline{u}(\underline{x}, t) = \underline{\nabla} \cdot D_{\parallel} \hat{\underline{e}}_z (\hat{\underline{e}}_z \underline{\nabla} \underline{u}(\underline{x}, t))$

10 Solving Linear Equations with Iterative Solvers and the Multigrid Technique

Relevance of solving linear equations for simulations: Implicit schemes lead to linear systems of equations which we have to solve, examples already covered are implicit Euler applied to a linear ODE in sec. 3.4.5, or to a non-linear one (the linear equation then follows from doing the implicit step by Newton-Raphson where the Jacobian is to be inverted (sec. 3.4.6)), or just recently in the diffusion equation.

Another example of a linear system we care about is solving a discretized Poisson equation.

Direct inversion or LU or QU decomposition of our linear system (the matrix describing it) are often too costly, so iterative methods - methods where we construct a sequence of approximate solutions that hopefully converge to the exact solution - are used.

In the multigrid-technique, differential equations are solved using a hierarchy of discretizations.

10.1 Motivational Example 1: From the Poisson equation we can get a possibly big linear system

Let us discretize the 1D Poisson equation

$$\partial_x^2 \phi = 4\pi G \rho \rightarrow (\partial_x^2 \phi) \approx \frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{h^2} \approx 4\pi G \rho_i \quad (434)$$

$$i = 1, \dots, N, \quad \text{spacing } h$$

We write this as a matrix equation

$$\underline{\underline{A}} \underline{\phi} = \underline{b}, \quad \underline{\phi} = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{pmatrix}, \quad \underline{b} = 4\pi G \underline{\rho}, \quad \underline{\rho} = \begin{pmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_N \end{pmatrix} \quad (435)$$

with in the case of periodic boundary conditions

$$\underline{\underline{A}} = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & \dots & 0 & 1 \\ 1 & -2 & 1 & 0 & \dots & \\ 0 & 1 & -2 & 1 & 0 & \dots \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & & 1 & -2 & 1 \\ 1 & 0 & 0 & \dots & 1 & -2 \end{pmatrix} \quad (436)$$

Problem: LU-decomposition or Gauss elimination with pivoting are generally $\mathcal{O}(N^3)$ (for such a sparse matrix we can do better though). Imagine a discretization in 3D, then for 1000 grid points per dimension $\underline{\underline{A}}$ would be $10^9 \times 10^9$ (mind the discretization in 3D is not *as* simple as in 1D).

10.2 Poisson equation and solving a tridiagonal system

10.2.1 1D heat Diffusion equation with Dirichlet boundaries in matrix form

Consider simple heat diffusion with constant heating rate ϵ .

$$-D\partial_x^2 T = \epsilon \quad (437)$$

so a 1D Poisson equation with constant source term ϵ .

Discretized we get as before

$$\frac{T_{i+1} - 2T_i + T_{i-1}}{\delta^2} \approx -\frac{\epsilon}{D} \quad (438)$$

So for a state vector

$$\underline{T} = \begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_N \end{pmatrix} \quad (439)$$

with Dirichlet boundary conditions $T_1 = T_N := T_0$ we get the matrix equation

$$\underline{\underline{A}}\underline{T} = \underline{b}, \quad \underline{b} = \begin{pmatrix} T_0 \\ -\frac{\delta^2 \epsilon}{D} \\ \vdots \\ -\frac{\delta^2 \epsilon}{D} \\ T_0 \end{pmatrix}, \quad \underline{\underline{A}} = \frac{1}{h^2} \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & -2 & 1 & 0 & \dots & \\ 0 & 1 & -2 & 1 & 0 & \dots \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & & 1 & -2 & 1 \\ 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix} \quad (440)$$

So for a 1D Poisson problem with Dirichlet (constant) boundary conditions we get a tridiagonal matrix.

10.2.2 Forward elimination backward substitution method for solving a tridiagonal system

Consider the general tridiagonal matrix

$$\underline{\underline{A}} = \begin{pmatrix} d_1 & u_1 & 0 & \dots & 0 & 0 \\ l_1 & d_2 & u_2 & 0 & \dots & \\ 0 & l_2 & d_3 & u_3 & 0 & \dots \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & & l_{N-2} & d_{N-1} & u_{N-1} \\ 0 & 0 & \dots & 0 & l_{N-1} & d_N \end{pmatrix} \quad (441)$$

where we want to solve

$$\underline{\underline{A}}\underline{x} = \underline{b} \quad (442)$$

for \underline{x} . We use elementary operations $\underline{\underline{E}}_i$ to add multiples of rows in $\underline{\underline{A}}$ to other rows in $\underline{\underline{A}}$.

1. (Forward elimination) Using those operations, we successively (top-down) eliminate the lower diagonal \underline{l} of $\underline{\underline{A}}$.

$$\underline{\underline{E}}_1 \underline{\underline{E}}_2 \dots \underline{\underline{E}}_{N-1} \underline{\underline{A}} \underline{x} = \begin{pmatrix} d_1 & u_1 & 0 & \dots & 0 & 0 \\ 0 & \tilde{d}_2 & u_2 & 0 & \dots & \\ 0 & 0 & \tilde{d}_3 & u_3 & 0 & \dots \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & & 0 & \tilde{d}_{N-1} & u_{N-1} \\ 0 & 0 & \dots & 0 & 0 & \tilde{d}_N \end{pmatrix} \underline{x} = \underline{\underline{E}}_1 \underline{\underline{E}}_2 \dots \underline{\underline{E}}_{N-1} \underline{b} \quad (443)$$

with $\tilde{d}_2 = d_2 - \frac{l_1}{d_1}u_1$ and $\tilde{d}_i = d_i - \frac{l_{i-1}}{d_{i-1}}u_{i-1}$ for $i = 3, \dots, N$.

2. (Backward substitution) We can now solve the system starting from the last row.

10.3 Classical Exact Solution: LU-decomposition*

Why is it not a good idea to numerically invert the matrix?: While $\underline{\underline{A}} \in \mathbb{R}^{N \times N}$ might often be sparse (e.g. a sparse Jacobian), $\underline{\underline{A}}^{-1}$ is dense in general and our storage might not be able to handle N^2 terms of a dense inverse. This is not a problem of stability - the inverse can be computed as numerically stable as other methods are. Matrix inversion can even be done in $\mathcal{O}(N^{\log_2 7})$ (Strassen algorithm, *Gaussian elimination is not optimal*).

Let us discuss LU-decomposition (also called LR-decomposition).

Consider $\underline{\underline{L}}, \underline{\underline{R}} \in \mathbb{R}^{N \times N}$ and $\underline{\underline{A}} \in \mathbb{R}^{N \times N}$ invertible with

$$\underline{\underline{A}} = \underline{\underline{L}}\underline{\underline{R}}, \quad \text{lower-left triangular matrix } \underline{\underline{L}}, \text{ upper-right triangular matrix } \underline{\underline{R}} \quad (444)$$

Note: More generally, $\underline{\underline{P}}\underline{\underline{A}} = \underline{\underline{L}}\underline{\underline{R}}$ with $\underline{\underline{P}}$ a permutation matrix is possible for every invertible matrix $\underline{\underline{A}}$. Otherwise the decomposition is not always possible, e.g. for $a_{11} = 0 \rightarrow l_{11}$ or $l_{11} = 0$, so $\underline{\underline{L}}$ or $\underline{\underline{R}}$ would be singular (not full rank), so $\underline{\underline{A}}$ would not be invertible (contradiction). So $\underline{\underline{P}}$ is used to switch rows of $\underline{\underline{A}}$.

10.3.1 Solving the linear system for \underline{x} when we already know the LU-decomposition

We can split

$$\underline{\underline{A}}\underline{x} = \underline{\underline{L}}\underline{\underline{R}}\underline{x} \leftrightarrow \underline{\underline{L}}\underline{y} = \underline{b}, \quad \underline{\underline{R}}\underline{x} = \underline{y} \quad (445)$$

so into solving two triangular systems.

1. (Forward substitution) Solve $\underline{\underline{L}}\underline{y} = \underline{b}$ for \underline{y} . We can write the system as

$$\begin{pmatrix} l_{11} & & \\ \underline{l}_{*1} & \underline{\underline{L}}_{**} & \end{pmatrix} \begin{pmatrix} y_1 \\ \underline{y}_* \end{pmatrix} = \begin{pmatrix} b_1 \\ \underline{b}_* \end{pmatrix} \Rightarrow l_{11}y_1 = b_1, \quad \underline{l}_{*1}y_1 + \underline{\underline{L}}_{**}\underline{y}_* = \underline{b}_* \quad (446)$$

from which we can solve for $y_1 = \frac{b_1}{l_{11}}$. We can then construct a new triangular system for \underline{y}_*

$$\underline{\underline{L}}_{**}\underline{y}_* = \underline{b}_* - \underline{l}_{*1}y_1 \quad (447)$$

so \underline{y} can be solved recursively, yielding the explicit formula

$$y_i = \frac{1}{l_{ii}} \left(b_i - \sum_{k=1}^{i-1} l_{ik}y_k \right) \quad (448)$$

taking $\mathcal{O}(N^2)$ operations.

2. (Backward substitution) Then $\underline{\underline{R}}\underline{x} = \underline{\underline{y}}$ for \underline{x} . The logic is the same as for forward substitution, but bottom-up. We can write the system as

$$\begin{pmatrix} R_{**} & R_{*n} \\ r_{nn} & \end{pmatrix} \begin{pmatrix} \underline{x}_* \\ x_1 \end{pmatrix} = \begin{pmatrix} \underline{y}_* \\ y_n \end{pmatrix} \quad (449)$$

yielding the explicit formula

$$a_i = \frac{1}{r_{ii}} \left(y_i - \sum_{k=i+1}^N r_{ik} a_k \right) \quad (450)$$

also taking $\mathcal{O}(N^2)$ operations.

10.3.2 Calculating the LU-decomposition in $\mathcal{O}(N^3)$

From

$$\begin{pmatrix} a_{11} & \underline{\underline{A}}_{1*}^T \\ \underline{\underline{A}}_{*1} & \underline{\underline{\underline{A}}}^{**} \end{pmatrix} = \begin{pmatrix} l_{11} & \\ \underline{\underline{L}}_{*1} & \underline{\underline{\underline{L}}}^{**} \end{pmatrix} \begin{pmatrix} r_{11} & \underline{\underline{R}}_{1*}^T \\ & \underline{\underline{\underline{R}}}^{**} \end{pmatrix}, \quad \text{set diagonal of } \underline{\underline{L}} \text{ to 1} \rightarrow \text{unique decomp} \quad (451)$$

we can devise an $\mathcal{O}(N^3)$ algorithm to calculate $\underline{\underline{L}}$ and $\underline{\underline{R}}$, see code 2.

```

1      # in-place decomposition of A into L and R, the upper-right part of
2      # A is overwritten by R,
3      # the lower-left part of A is overwritten by L without diagonal (set
4      # to 1)
5      function lu_decomposition!(A::Matrix{T}) where T <: Number
6          N = size(A, 1)
7          for i in 1:N
8              for j in i+1:N
9                  A[j, i] /= A[i, i]
10                 for k in i+1:N
11                     A[j, k] -= A[j, i] * A[i, k]
12                 end
13             end
14         end
15         # one can then extract L = tril(A, -1) + I
16         # and U = triu(A) via the LinearAlgebra package

```

Code-Snippet 2: LU-decomposition of a matrix $\underline{\underline{A}} \in \mathbb{R}^{N \times N}$ in $\mathcal{O}(N^3)$.

Note: Blocked versions can be parallelized, QR-decomposition retains the condition number of $\underline{\underline{A}}$.

Problem: LU-decomposition is $\mathcal{O}(N^3)$.

10.4 Jacobi iteration | a splitting method

Aim: We want quicker than $\mathcal{O}(N^3)$, approximate solutions to linear systems.

Consider the linear system

$$\underline{\underline{A}}\underline{x} = \underline{b}, \quad \underline{\underline{A}} \in \mathbb{R}^{N \times N}, \underline{x}, \underline{b} \in \mathbb{R}^N \quad (452)$$

For Jacobi iteration, we start with the trivial decomposition

$$\begin{aligned} \underline{\underline{A}} &= \underline{\underline{D}} - (\underline{\underline{L}} + \underline{\underline{U}}), & \text{diagonal part } \underline{\underline{D}}, & \text{negative left-below-diagonal part } \underline{\underline{L}} \\ && \text{negative right-above-diagonal part } \underline{\underline{U}} \end{aligned} \quad (453)$$

so

$$\underline{\underline{A}}\underline{x} = \left[\underline{\underline{D}} - (\underline{\underline{L}} + \underline{\underline{U}}) \right] \underline{x} = \underline{b} \rightarrow \underline{x} = \underline{\underline{D}}^{-1}\underline{b} + \underline{\underline{D}}^{-1}(\underline{\underline{L}} + \underline{\underline{U}})\underline{x} \quad (454)$$

so \underline{x} is a fixed point of the RHS, leading to fixed-point, here Jacobi iteration

$$\underline{x}^{(n+1)} = \underline{\underline{D}}^{-1}\underline{b} + \underline{\underline{D}}^{-1}(\underline{\underline{L}} + \underline{\underline{U}})\underline{x}^{(n)}, \quad (\underline{\underline{D}}^{-1})_{ii} = \frac{1}{A_{ii}} \quad (455)$$

10.4.1 When does the Jacobi iteration converge?

The Jacobi iteration converges if and only if all eigenvalues λ_i of the convergence matrix

$$\underline{\underline{M}} := \underline{\underline{D}}^{-1}(\underline{\underline{L}} + \underline{\underline{U}}) \quad (456)$$

are smaller than one.

$$\text{spectral radius } \rho_s(\underline{\underline{M}}) \equiv \max_i |\lambda_i| < 1 \quad (457)$$

The smaller the spectral radius, the faster the convergence.

10.4.1.1 Derivation of the convergence criterion

Consider the error at step $n + 1$

$$\begin{aligned} \underline{e}^{(n+1)} &= \underline{x}_{\text{exact}} - \underline{x}^{(n+1)} \\ &\stackrel{\text{Jacobi-Iteration}}{=} \underline{x}_{\text{exact}} - \left(\underline{\underline{D}}^{-1} \underline{b} + \underline{\underline{M}} \underline{x}^{(n)} \right) \\ &\stackrel{\underline{x}_{\text{exact}} = \underline{\underline{D}}^{-1} \underline{b} + \underline{\underline{M}} \underline{x}_{\text{exact}}}{=} \underline{\underline{M}} (\underline{x}_{\text{exact}} - \underline{x}^{(n)}) = \underline{\underline{M}} \underline{e}^{(n)} \end{aligned} \quad (458)$$

so

$$\underline{e}^{(n)} = \underline{\underline{M}}^n \underline{e}^{(0)} \quad (459)$$

which scales with $\rho_s^n(\underline{\underline{M}})$ for n sufficiently large (follows from decomposing $\underline{e}^{(0)}$ into eigenvectors of $\underline{\underline{M}}$) so the convergence criterion follows.

Problem: The convergence is usually slow, better use Gauss-Seidel iteration.

10.4.2 Example Jacobi Step

Consider

$$\underline{\underline{A}} \underline{x} = \begin{pmatrix} 10 & -5 & 0 \\ -5 & 10 & -5 \\ 0 & -5 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \underline{b} = \begin{pmatrix} 0 \\ 0 \\ 15 \end{pmatrix} \quad (460)$$

then

$$\underline{\underline{D}} = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 5 \end{pmatrix}, \quad \underline{\underline{L}} = \begin{pmatrix} 0 & 0 & 0 \\ 5 & 0 & 0 \\ 0 & 5 & 0 \end{pmatrix}, \quad \underline{\underline{R}} = \begin{pmatrix} 0 & 5 & 0 \\ 0 & 0 & 5 \\ 0 & 0 & 0 \end{pmatrix} \quad (461)$$

so

$$\underline{x}^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \rightarrow \underline{x}^{(1)} = \begin{pmatrix} \frac{1}{10} b_0 + \frac{1}{10} \cdot 5x_2^{(0)} \\ \frac{1}{10} b_1 + \frac{1}{10} \cdot (5x_1^{(0)} + 5x_3^{(0)}) \\ \frac{1}{5} b_1 + \frac{1}{5} \cdot 5x_2^{(0)} \end{pmatrix} \quad (462)$$

10.5 Gauss-Seidel iteration | better splitting method

Idea: In each step $\underline{x}^{(n)} \rightarrow \underline{x}^{(n+1)}$ ($\underline{x} \in \mathbb{R}^N$) consists of N scalar updates. If we do these N scalar updates iteratively (not in parallel) we can use the results from previous updates.

10.5.1 Motivational Example

For instance in the example above

$$\underline{x}^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \rightarrow \underline{x}^{(1)} = \begin{pmatrix} \frac{1}{10}b_0 + \frac{1}{10} \cdot 5x_2^{(0)} \\ \frac{1}{10}b_1 + \frac{1}{10} \cdot (5x_1^{(1)} + 5x_3^{(0)}) \\ \frac{1}{5}b_1 + \frac{1}{5} \cdot 5x_2^1 \end{pmatrix} \quad (463)$$

10.5.2 Gauss-Seidel update

Let us devise a different fix-point equation

$$(\underline{\underline{D}} - \underline{\underline{L}})\underline{x} = \underline{\underline{U}}\underline{x} + \underline{b} \rightarrow \underline{x} = (\underline{\underline{D}} - \underline{\underline{L}})^{-1}\underline{\underline{U}}\underline{x} + (\underline{\underline{D}} - \underline{\underline{L}})^{-1}\underline{b} \quad (464)$$

from which we follow the fix-point iteration

$$\underline{x}^{(n+1)} = (\underline{\underline{D}} - \underline{\underline{L}})^{-1}\underline{\underline{U}}\underline{x}^{(n)} + (\underline{\underline{D}} - \underline{\underline{L}})^{-1}\underline{b} \quad (465)$$

Problem: We cannot easily compute $(\underline{\underline{D}} - \underline{\underline{L}})^{-1}$.

Therefore, multiply with $(\underline{\underline{D}} - \underline{\underline{L}})$.

$$(\underline{\underline{D}} - \underline{\underline{L}})\underline{x}^{(n+1)} = \underline{\underline{U}}\underline{x}^{(n)} + \underline{b} \rightarrow \underline{x}^{(n+1)} = \underline{\underline{D}}^{-1}\underline{\underline{L}}\underline{x}^{(n+1)} + \underline{\underline{D}}^{-1}\underline{\underline{U}}\underline{x}^{(n)} + \underline{\underline{D}}^{-1}\underline{b} \quad (466)$$

But isn't this implicit now with $\underline{x}^{(n+1)}$ on both sides?: $\underline{\underline{L}}$ is a lower diagonal matrix (diagonal and everything above zero), so in $\underline{\underline{L}}\underline{x}^{(n+1)}$ we always only need values we have already calculated if we solve consecutively (\rightarrow problem for parallelization). This is illustrated in fig. 77.

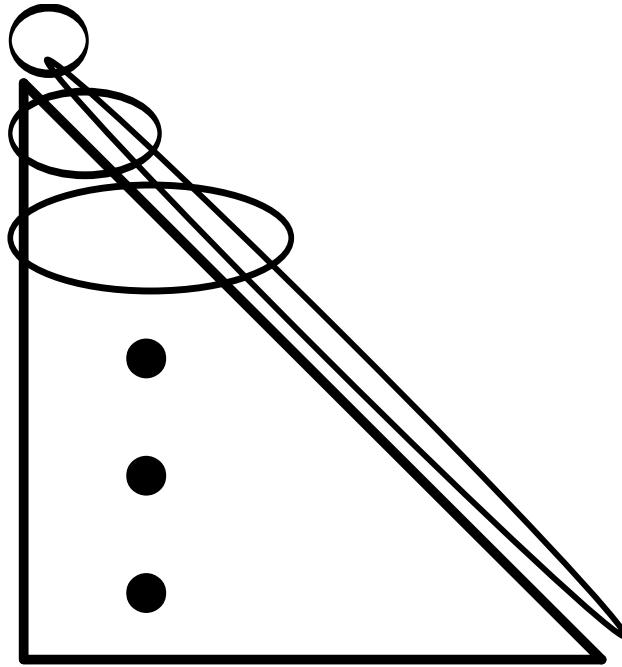


Figure 77: As of $\underline{L}\underline{x}^{(n+1)}$ to calculate $x_i^{(n+1)}$ we only need $x_0^{(n+1)}, \dots, x_{i-1}^{(n+1)}$ which we have already calculated.

Advantage of Gauss-Seidel: Convergence is sped up (often by a factor of 2) compared to Jacobi iteration. Convergence is guaranteed if $\underline{\underline{A}}$ is strictly diagonally dominant $|A_{ii}| > \sum_{j \neq i} |A_{ij}| \quad \forall i$ or symmetric and positive definite.

10.5.3 The problem of parallelization in Gauss-Seidel and red-black ordering

Problem: In Gauss-Seidel, the equations must be solved in sequential order - this cannot be parallelized. A general problem of using information from the same step is that the overall result is order dependent (which element is selected first).

Idea: Do not do a scheme with sequential dependence (where information is used as soon as available) but if possible e.g. rather split the N updates into two groups where all updates within a group are independent, but the updates of the second group depend on the ones from first.

For instance for the 2d-Poisson equation, a *red-black ordering* scheme can be derived, see figure 78 (details follow).

2d-Poisson in red-black ordering

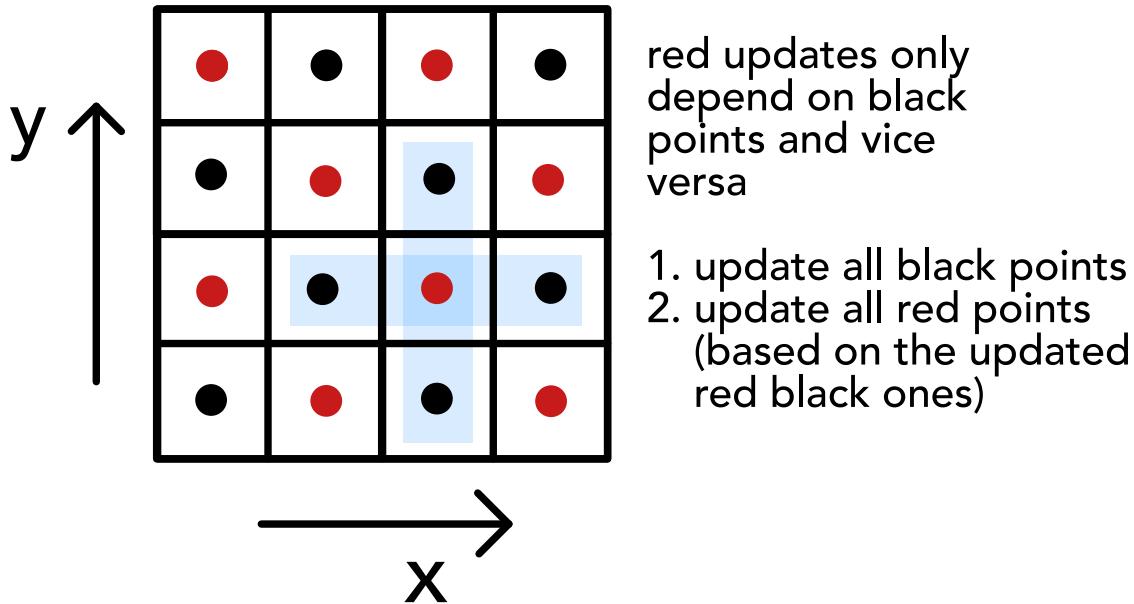


Figure 78: Red-black ordering for the 2D Poisson equation.

10.6 Relaxation problem | Poisson equation in red-black ordering

We can physically understand why applying an iterative scheme to a stationary (elliptic) problem makes sense in terms of relaxation of a dynamical system.

10.6.1 Formulating an elliptic equation as an equilibrium of a relaxation problem

Consider the elliptic equation

$$L\underline{x} = \underline{b} \quad (467)$$

for instance for the gravitational 2D-Poisson problem $(\partial_x^2 + \partial_y^2)\phi = 4\pi G\rho$.

The elliptic problem can be written as the equilibrium of the relaxation problem

$$\frac{1}{K} \partial_t \underline{x} = L\underline{x} - \underline{b}, \quad \text{for } \partial_t \underline{x} = 0 \text{ for } t \rightarrow \infty, \quad \text{some factor } K \text{ in } \frac{\text{m}^2}{\text{s}} \quad (468)$$

10.6.2 Red-black ordering for the 2D Poisson equation

Consider for instance the 2D Poisson equation, formulated as a relaxation problem

$$K \frac{\phi_{i,j}^{(n+1)} - \phi_{i,j}^{(n)}}{\Delta t} = \frac{\phi_{i+1,j}^{(n)} - 2\phi_{i,j}^{(n)} + \phi_{i-1,j}^{(n)}}{\Delta x^2} + \frac{\phi_{i,j+1}^{(n)} - 2\phi_{i,j}^{(n)} + \phi_{i,j-1}^{(n)}}{\Delta x^2} - 4\pi G \rho_{ij} \quad (469)$$

so

$$\phi_{i,j}^{(n+1)} = \frac{K \Delta t}{\Delta x^2} \left(\phi_{i,j}^{(n+1)} + \phi_{i-1,j}^{(n)} + \phi_{i,j+1}^{(n)} + \phi_{i,j-1}^{(n)} \right) + \underbrace{\left(1 - 4 \frac{K \Delta t}{\Delta x^2} \right)}_{=0 \text{ by choice } \Delta t = \frac{1}{4} \frac{K}{\Delta x^2}} \phi_{i,j}^{(n)} - 4\pi G \rho_{ij} \Delta t \quad (470)$$

where the time-step choice is akin to the CFL criterion in diffusion.

The discretized Poisson-relaxation update lends itself naturally to a red-black ordering scheme.

10.7 Multigrid technique

We have now gained an understanding of iteratively solving a linear equation as a relaxation problem.

Note: In each update step information travels as given by the stencil, for the Poisson equation only between nearest neighbors. This also limits the largest possible *timestep* to the CFL criterion.

Problem: As only cells in the stencil (often only neighboring cells) communicate per step in Jacobi and Gauss-Seidel iteration, we have to make a compromise between speed of convergence and resolution:

- On a coarse grid, information travels quicker through space (in less steps) but the resolution is bad
- On a fine grid, resolution is good, but long range interactions take lots of computational steps and long-wavelength errors (in the error vector \underline{e}) die out only very slowly

Idea: Start on a coarse grid, where information travels quickly, long range correlations are taken care of and we have fast convergence, then get the information onto a fine grid and resolve the details.

1. But how can we map from a coarser to a finer grid and vice versa?
2. How can we solve $\underline{A}\underline{x} = \underline{b}$ on the coarser grid?

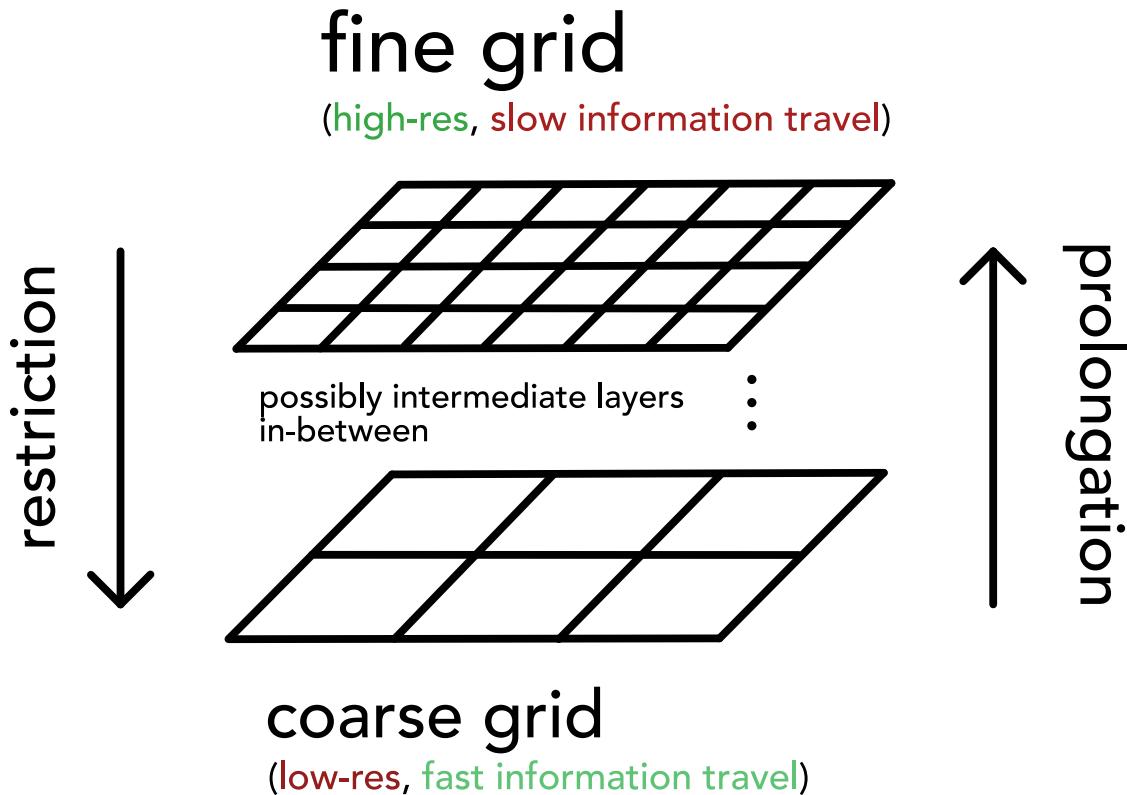


Figure 79: Grids of different resolution.

Note: Coarse-to-fine is called prolongation, fine-to-coarse restriction (see figure 79).

10.7.1 Getting finer and coarser | prolongation and restriction

Note: In the following we will only consider the case of a 1D grid (not 2D as illustrated before).

Consider meshes

1. Let $\Omega^{(h)}$ denote a 1D mesh with N cells $i = 1, \dots, N$ with spacing h .
2. Let $\Omega^{(2h)}$ denote a 1D mesh with $N/2$ cells $i = 1, \dots, N/2$ with spacing $2h$.

and let

1. $\underline{x}_i^{(h)} \in \mathbb{R}^N$ denote the solution on $\Omega^{(h)}$, so $x_i^{(h)}$ is the solution on cell i of $\Omega^{(h)}$ (e.g. a density $\rho_i^{(h)}$).

2. $\underline{x}_i^{(2h)} \in \mathbb{R}^{N/2}$ denote the solution on $\Omega^{(2h)}$, so $x_i^{(2h)}$ is the solution on cell i of $\Omega^{(2h)}$.

Restriction and prolongation in 1D are illustrated in figure 80.

Prolongation and restriction are linear operators written as matrices $I_{\text{from this spacing}}^{\text{to this spacing}}$.

Restriction Prolongation

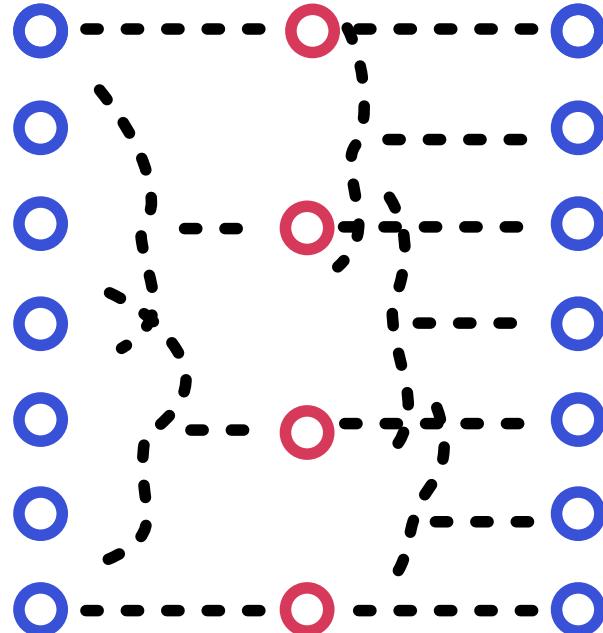


Figure 80: Illustration of restriction and prolongation.

10.7.1.1 Coarse-to-fine | interpolation called prolongation

We map from coarse to fine using

$$\underline{\underline{I}}_{=2h}^h \underline{x}^{(2h)} = \underline{x}^{(h)} \quad (471)$$

for instance by

$$\underline{\underline{I}}_{=2h}^h \in \mathbb{R}^{N \times \frac{N}{2}} : \text{for } 0 \leq i < \frac{N}{2} : x_{2i}^{(h)} = x_i^{(2h)}, \quad x_{2i+1}^{(h)} = \frac{1}{2} (x_i^{(2h)} + x_{i+1}^{(2h)}) \quad (472)$$

as shown in figure 80 and furthermore in figure 81.

prolongation

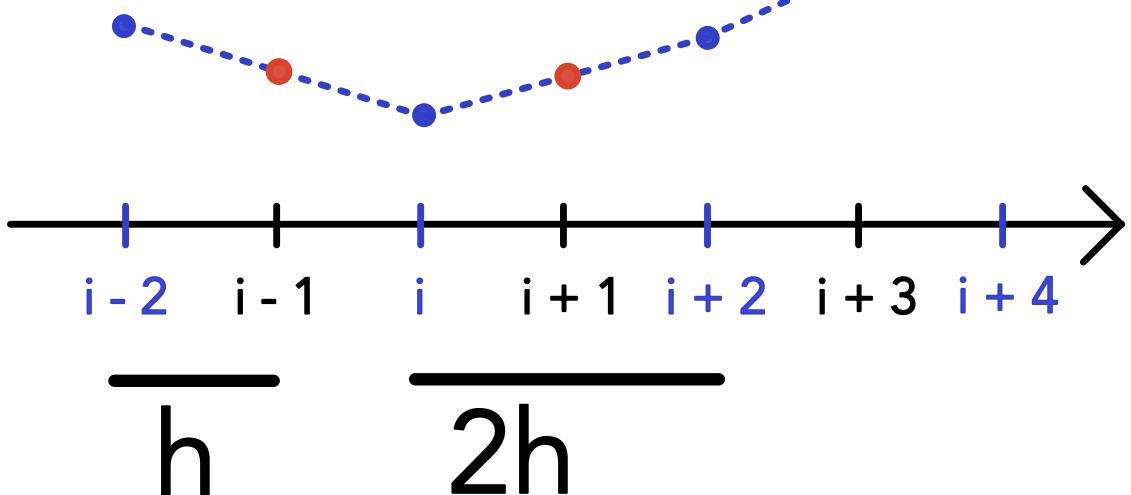


Figure 81: 1D-Prolongation from coarse to fine grid.

10.7.1.2 Fine-to-coarse | restriction

We now convert

$$\underline{\underline{I}}_h^{2h} \underline{x}^{(h)} = \underline{x}^{(2h)} \quad (473)$$

with a simple example being

$$\underline{\underline{I}}_h^{2h} \in \mathbb{R}^{\frac{N}{2} \times N} : \text{for } 0 \leq i < \frac{N}{2} : x_i^{(2h)} = \frac{x_{2i-1}^{(h)} + 2x_{2i}^{(h)} + x_{2i+1}^{(h)}}{4} \quad (474)$$

10.7.1.3 Relation of restriction and prolongation

Prolongation and restriction matrices are oftentimes constructed to be scaled transposes of each other, so

$$\underline{\underline{I}}_h^{2h} = c[\underline{\underline{I}}_h^h]^T \quad (475)$$

Consider for instance

$$\begin{aligned}
 \text{prolongation: } I_{=2h}^h &= \frac{1}{2} \begin{pmatrix} 1 & & & \\ 2 & & & \\ 1 & 1 & & \\ & 2 & & \\ 1 & 1 & & \\ & 2 & & \\ & & 1 & \end{pmatrix} \\
 \text{restriction: } I_{=h}^{2h} &= \frac{1}{2} \left(I_{=2h}^h \right)^T = \frac{1}{4} \begin{pmatrix} 1 & 2 & 1 & & & \\ & 1 & 2 & 1 & & \\ & & 1 & 2 & 1 & \\ & & & 1 & 2 & 1 \end{pmatrix}
 \end{aligned} \tag{476}$$

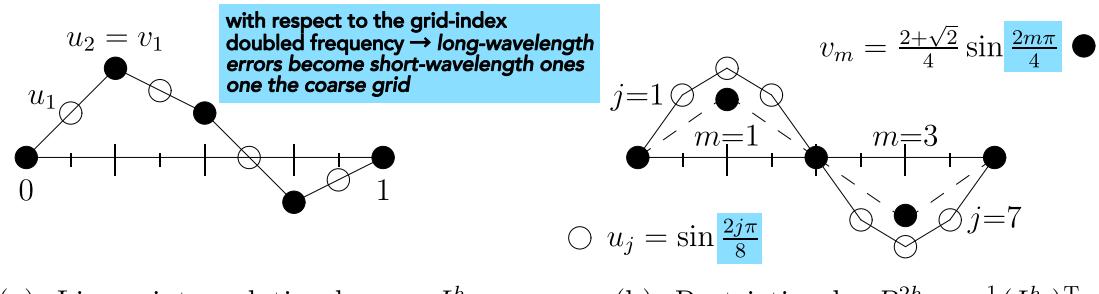
so the application of **prolongation** means

$$\frac{1}{2} \begin{pmatrix} 1 & & & \\ 2 & & & \\ 1 & 1 & & \\ & 2 & & \\ 1 & 1 & & \\ & 2 & & \\ & & 1 & \end{pmatrix} \begin{pmatrix} x_1^{(2h)} \\ x_1^{(2h)} \\ x_1^{(2h)}/2 + x_2^{(2h)}/2 \\ x_2^{(2h)} \\ x_2^{(2h)}/2 + x_3^{(2h)}/2 \\ x_3^{(2h)} \\ x_3^{(2h)}/2 \end{pmatrix} = \begin{pmatrix} x_1^{(h)} \\ x_2^{(h)} \\ x_3^{(h)} \\ x_4^{(h)} \\ x_5^{(h)} \\ x_6^{(h)} \\ x_7^{(h)} \end{pmatrix} \tag{477}$$

and **restriction**

$$\frac{1}{4} \begin{pmatrix} 1 & 2 & 1 & & & \\ & 1 & 2 & 1 & & \\ & & 1 & 2 & 1 & \end{pmatrix} \begin{pmatrix} x_1^{(h)} \\ x_2^{(h)} \\ x_3^{(h)} \\ x_4^{(h)} \\ x_5^{(h)} \\ x_6^{(h)} \\ x_7^{(h)} \end{pmatrix} = \begin{pmatrix} x_1^{(2h)} \\ x_2^{(2h)} \\ x_3^{(2h)} \end{pmatrix} \tag{478}$$

Both are illustrated in figure 82, also illustrating that with respect to the grid index, low-frequency errors on the fine grid have double (so higher) frequency on the coarse grid.



(a) Linear interpolation by $u = I_{2h}^h v$ (b) Restriction by $R_h^{2h} u = \frac{1}{2}(I_{2h}^h)^T u$

Figure 82: Prolongation and restriction of a sine wave (figure by Gilbert Strang).

Note: Taking prolongation and restriction as transposes of each others breaks at the boundaries, if we do not as in figure 82 have zeroes on the boundaries.

10.7.1.4 Short-hand stencil notation

Short-hand notations for prolongation and restriction are given in table 15.

Prolongation	Restriction
<p>We can write prolongation as</p> <p>1D prolongation: $I_{=2h}^h : \left[\begin{array}{ccc} \frac{1}{2} & 1 & \frac{1}{2} \end{array} \right]$ (479)</p> <p>meaning that every coarse point is added to three fine points with these respective weights (see previous example).</p>	<p>We can write restriction as</p> <p>1D restriction: $I_h^{2h} : \left[\begin{array}{ccc} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{array} \right]$ (480)</p> <p>- the coarse point is the weighted sum of the three fine points.</p>

Table 15: Short-hand notations for prolongation and restriction.

Similar short-hand notations can be devised for 2D and 3D.

10.7.2 Multigrid V-cycle

Our aim is still to solve $\underline{\underline{A}}x = \underline{b}$ smartly, by - in the view of a relaxation problem - doing the rough large-scale work quickly on a coarse and the fine details on a fine grid afterwards. We still have to answer

- How can such a procedure look like?
- How do we convert $\underline{\underline{A}}$ to coarser grids?

10.7.2.1 Initial definitions

The error is defined as

$$\underline{e} \equiv \underline{x}_{\text{exact}} - \tilde{\underline{x}}, \quad \text{current approximate solution } \tilde{\underline{x}} \quad (481)$$

Note: The residual is the error in the solution, not \underline{x} . The error and residual are related linearly by $\underline{\underline{A}}$.

$$\underline{\underline{A}}\underline{x}_{\text{exact}} = \underline{b} \quad \rightarrow \quad \underline{\underline{A}}(\underline{e} + \tilde{\underline{x}}) = \underline{b} \quad \rightarrow \quad \underline{\underline{A}}\underline{e} = \underline{r} \quad (482)$$

Idea: Consider we have an initial guess $\tilde{\underline{x}}$ on the fine grid. We can easily calculate the residual \underline{r} . We can then restrict \underline{r} to a coarse grid, solve for the error there ($\underline{\underline{A}}\underline{e} = \underline{r}$) (given we can transform $\underline{\underline{A}}$ to the coarse grid) and then prolong the error to the fine grid and make the correction $\underline{x}_{\text{exact}} = \tilde{\underline{x}} + \underline{e}$.

Let us formalize this idea.

10.7.2.2 Coarse-grid correction scheme

We use the error calculated on the coarse grid to correct on the fine grid.

Let us start with a (fine-grid) guess $\tilde{\underline{x}}^{(h)}$ for the problem

$$\underline{\underline{A}}^{(h)}\underline{x}^{(h)} = \underline{b}^{(h)} \quad (483)$$

and we want to obtain an improvement $\underline{x}'^{(h)} = CG(\underline{\underline{A}}^{(h)}, \underline{b}^{(h)})$.

CG consists of the following steps

1. Perform one iterative *relaxation* step on the current grid h , e.g. Jacobi or Gauss-Seidel
2. Compute the residual

$$\underline{r}^{(h)} = \underline{b} - \underline{\underline{A}}\tilde{\underline{x}}^{(h)} \quad (484)$$

3. Restrict the residual to the coarser mesh

$$\underline{r}_{\frac{1}{2}h}^{(2h)} = I_{\frac{1}{2}h}^{2h}\underline{r}^{(h)} \quad (485)$$

4. Solve for the error on the coarser mesh

$$\underline{\underline{A}}^{(2h)}\underline{e}^{(2h)} = \underline{r}^{(2h)}, \quad \text{starting guess } \tilde{\underline{e}}^{(2h)} = \underline{0} \quad (486)$$

5. Correct $\tilde{x}^{(h)}$ on the finer mesh using the prolonged error $\underline{e}^{(h)} = \underline{\underline{I}}_{=2h}^h \underline{e}^{(h)}$

$$\underline{\tilde{x}}'^{(h)} = \underline{\tilde{x}}^{(h)} + \underline{e}^{(h)} \quad (487)$$

6. Further iterative *relaxation* step on the fine mesh

How to solve for the error on the coarse mesh?: Step 4 can be performed by recursively calling $CG(\underline{\tilde{x}}^{(2^ih)}, \underline{b}^{(2^ih)})$, $i \geq 1$ until a coarseness is reached where one can easily exactly solve or solve by multiple relaxation steps.

How to find $\underline{\underline{A}}^{(2h)}$ on the coarse mesh?: Generally in the **Galerkin coarse grid approximation** one defines

$$\underline{\underline{A}}^{(2h)} = \underline{\underline{I}}_{=h}^{2h} \underline{\underline{A}}^{(h)} \underline{\underline{I}}_{=2h}^h \quad (488)$$

which is an additional matrix operation that might enlarge the stencil (and so the computational cost), depending on the interpolation operator.

When $\underline{\underline{A}}$ was brought forth by formulating a discretization in matrix form, we can use the same discrete equations as one the fine grid (**direct method**). In other words, the same stencil (*Schablone*) is used to go over the points and collect for the linear relations, e.g. for the 2D-Poisson - no matter the coarseness - we would construct $\underline{\underline{A}}$ based on eq. 470.

10.7.2.3 V-cycle

The 6-step iteration process with a recursion call in step 4 leads to a cycle as illustrated in figure 83.

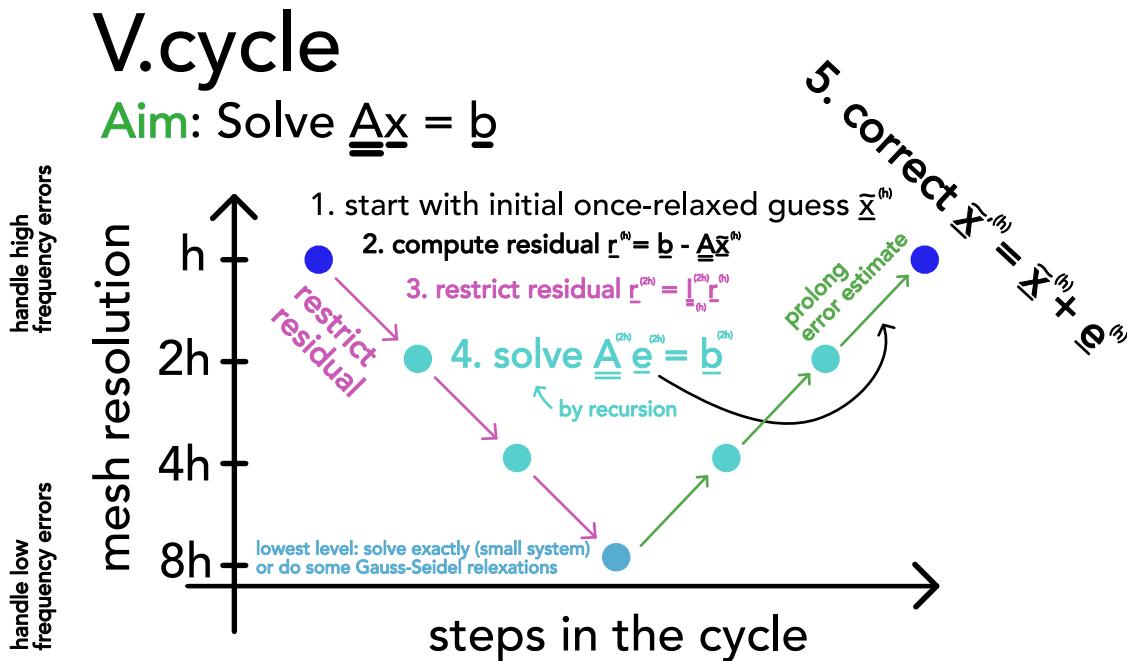


Figure 83: V-cycle.

The V-cycle for the Poisson equation has the same cost as FFT-based methods but requires less thread communication when parallelized.

Computational costs

computational cost per V-cycle: $\mathcal{O}(N_{\text{grid}})$

computational cost until convergence to machine error (mult. cycles) $\mathcal{O}(N_{\text{grid}} \log N_{\text{grid}})$
with N_{grid} being the number of grid-cells on the fine grid

(489)

10.7.2.4 Full multigrid method

Problem: How to make a good initial guess $\tilde{x}^{(h)}$ (if not available e.g. by taking the result from the last time-step in a simulation)?

Idea: First solve on the coarsest grid, then interpolate up to get a good initial guess. Based on this idea, the full multigrid cycle is constructed.

The multigrid cycle is illustrated in figure 84.

mesh resolution

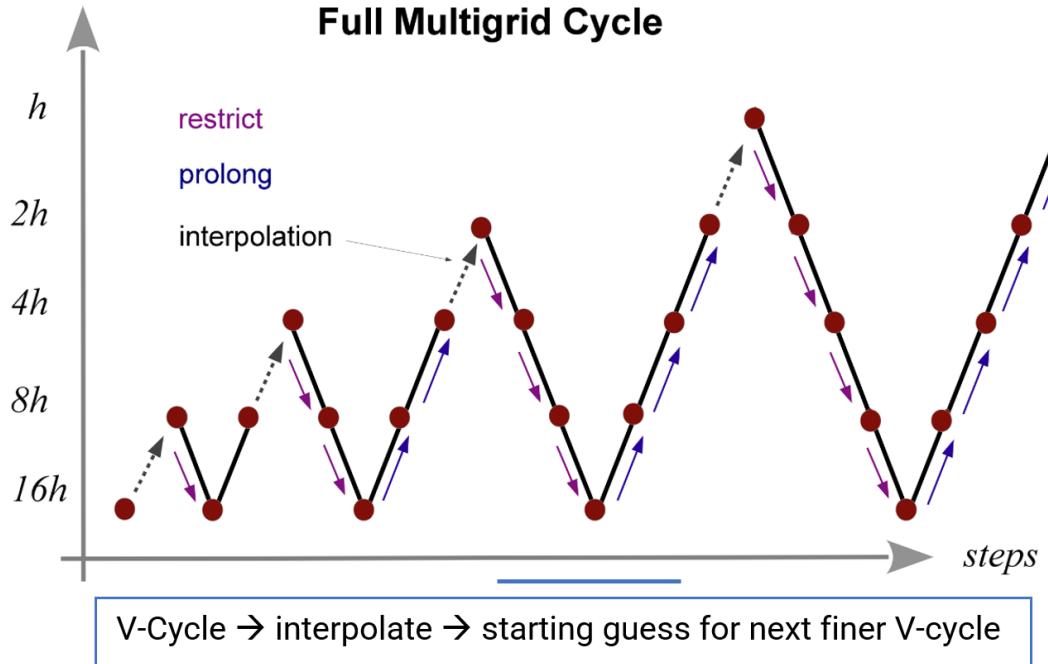


Figure 84: Full multigrid cycle.

The steps are

1. Initialize the righthand side on all grid levels $\underline{b}^{(h)}, \underline{b}^{(2h)}, \underline{b}^{(H)}$.
2. Solve $A^{(H)} \underline{x}^{(H)} = \underline{b}^{(H)}$ exactly on the coarsest level H .
3. Obtain an initial guess for a finer grid by interpolating the solution from the next coarser level below, i.e. $\underline{\tilde{x}}^{(h)} = \underline{\underline{I}}_{=2h}^h \underline{\tilde{x}}^{(2h)}$
4. Solve the problem in the finer level using this starting guess $\underline{\tilde{x}}^{(h)}$ and one v-cycle.
5. repeat step 3 until finest level is reached

Computational cost of one multigrid cycle: As the V-cycle it has $\mathcal{O}(N_{\text{grid}})$ as N_{grid} is large compared to the cost of the V-cycle hierarchy.

10.8 Krylow subspace methods

Our aim still is to solve a linear system $\underline{\underline{A}} \underline{\underline{x}} = \underline{\underline{b}}$.

Problem: LU-decomposition has cubic runtime ($\sim \frac{2}{3}n^3$) and the previously discussed iterative methods might also not always be optimal.

10.8.1 Motivation for the need of Krylow subspace methods in the context of non-linear root-finding of big systems*

Consider a non-linear equation

$$\underline{0} = g(\xi) \quad (490)$$

which we want to solve for ξ .¹⁹ A common root-finding approach is Newton-Iteration²⁰

$$\underline{\xi}_{k+1} = \underline{\xi}_k - \underline{\underline{J}}_g^{-1}(\underline{\xi}_k)g(\underline{\xi}_k), \quad \text{Jacobian } \underline{\underline{J}}_g \quad (492)$$

where at the lowest level we have to solve the linear equation

$$\underline{b} := g(\underline{\xi}_k) = \underline{\underline{J}}_g(\underline{\xi}_k - \underline{\xi}_{k+1}) = \underline{\underline{J}}_g \underline{a} \quad (493)$$

for \underline{a} .

Problem: But what if the resulting Jacobian is too large to be stored?

Idea: A directional derivative (a Jacobian vector product^a) can be calculated in one (forward) automatic differentiation pass - cheaply and storage efficient. So what if we could solve $\underline{\underline{J}}\underline{a} = \underline{b}$ only based on Jacobian vector products?

^a

$$\begin{aligned} \underline{\underline{J}}\underline{v} &= (\nabla \cdot g)\underline{v} = \lim_{h \rightarrow 0} \frac{g(\underline{x} + \underline{v}h) - g(\underline{x})}{h} \\ \left(\text{as } (\underline{\underline{J}}\underline{v})_i &= \sum_{j=1}^n (\underline{\underline{J}}_g)_{ij} v_j = \sum_{j=1}^n \frac{\partial g_i}{\partial x_j} v_j = ((\nabla \cdot g)\underline{v})_i \right) \end{aligned} \quad (494)$$

10.8.2 Motivation | solving a system of linear equations using a gradient method

Consider $\underline{\underline{A}} \in \mathbb{R}^{N \times N}$ symmetric ($\underline{\underline{A}}^T = \underline{\underline{A}}$) and positive definite, so $\underline{x}^T \underline{\underline{A}} \underline{x} > 0 \forall \underline{x} \in \mathbb{K}^N \setminus \{\underline{0}\}$ (only positive eigenvalues). Then the following holds

$$\underline{\underline{A}}\underline{x} = \underline{b} \leftrightarrow \text{quadratic form } f(x) := \frac{1}{2}\underline{x}^T \underline{\underline{A}} \underline{x} - \underline{x}^T \underline{b} \text{ is minimal} \quad (495)$$

¹⁹For instance an implicit Euler step can be formulated as such a root finding problem.

$$\begin{aligned} \text{implicit step } \underline{y}^{(n+1)} &= \underline{y}^{(n)} + \Delta t f(\underline{y}^{(n+1)}) \\ \rightarrow \text{root-finding-problem } \underline{0} &= \underline{y}^{(n+1)} - \underline{y}^{(n)} - \Delta t f(\underline{y}^{(n+1)}) =: g(\underline{\xi}) \end{aligned} \quad (491)$$

²⁰Which has quadratic convergence - as the method converges on the root, the difference between the root and the approximation is squared in each step.

(intuitively as the quadratic form for a positive definite matrix if parabolic) where we can now apply standard gradient descent (**steepest descent method**)

$$\underline{x}^{(n+1)} = \underline{x}^{(n)} + \alpha^{(n)} \underline{p}^{(n)}, \quad \underline{p}^{(n)} = -\underline{\nabla} f(\underline{x}^{(n)}) = -(b - \underline{A}\underline{x}^{(n)}) \quad (496)$$

so we take steps along the steepest descent with some *learning rate* $\alpha^{(n)}$

Note: Quadratic forms for different kinds of matrices are illustrated in figure 85. For an indefinite matrix the steepest descent method or the later-introduces conjugate gradient method will fail, because the solution is a saddle point. For the more general case of even non-symmetric matrices, GMRES can be used.

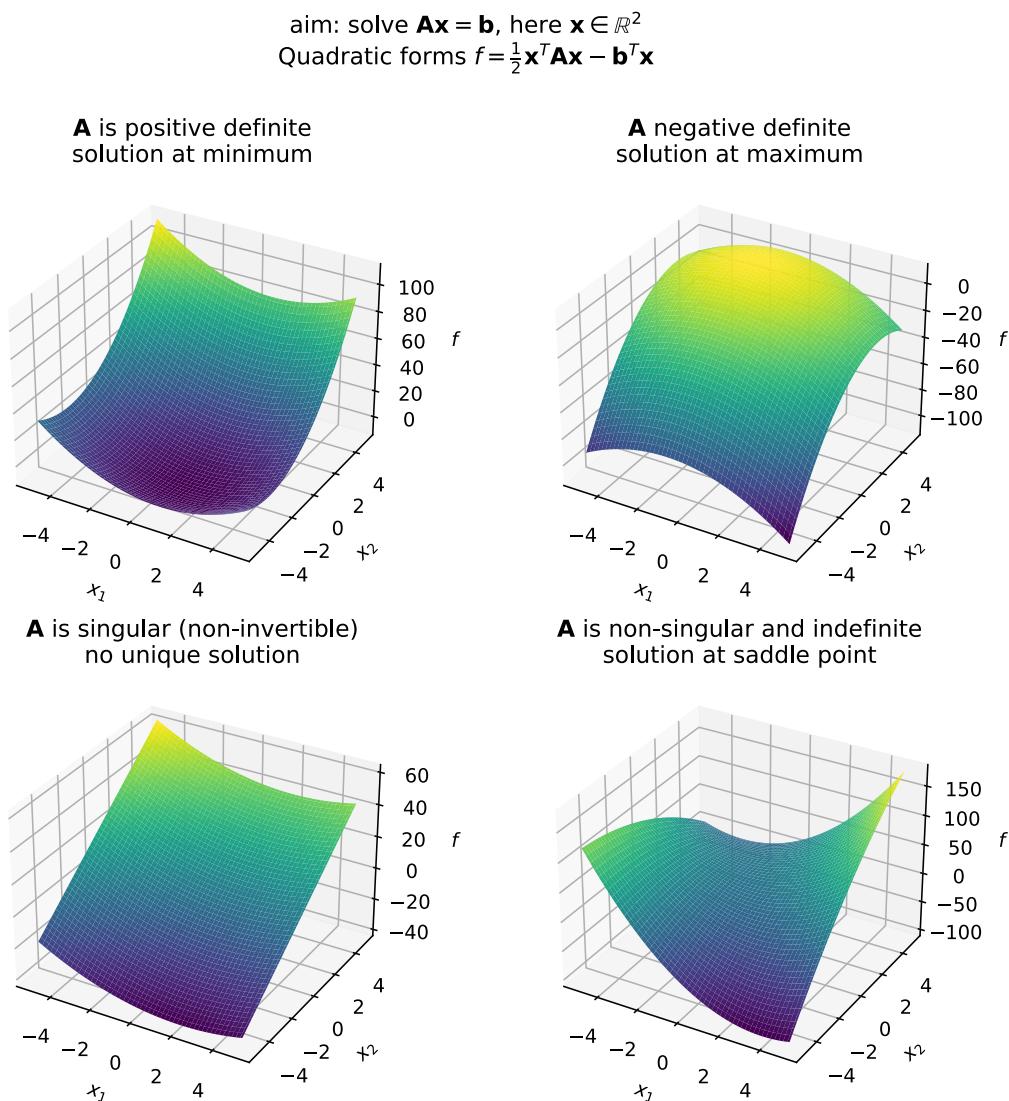


Figure 85: Quadratic forms for different kinds of matrices.

Problem: Steepest descent often takes steps in directions already taken. Steepest descent is good if our parabolic quadratic form f is nearly circular but the more elliptic f is, the more problematic steepest descent becomes, see figure 86.

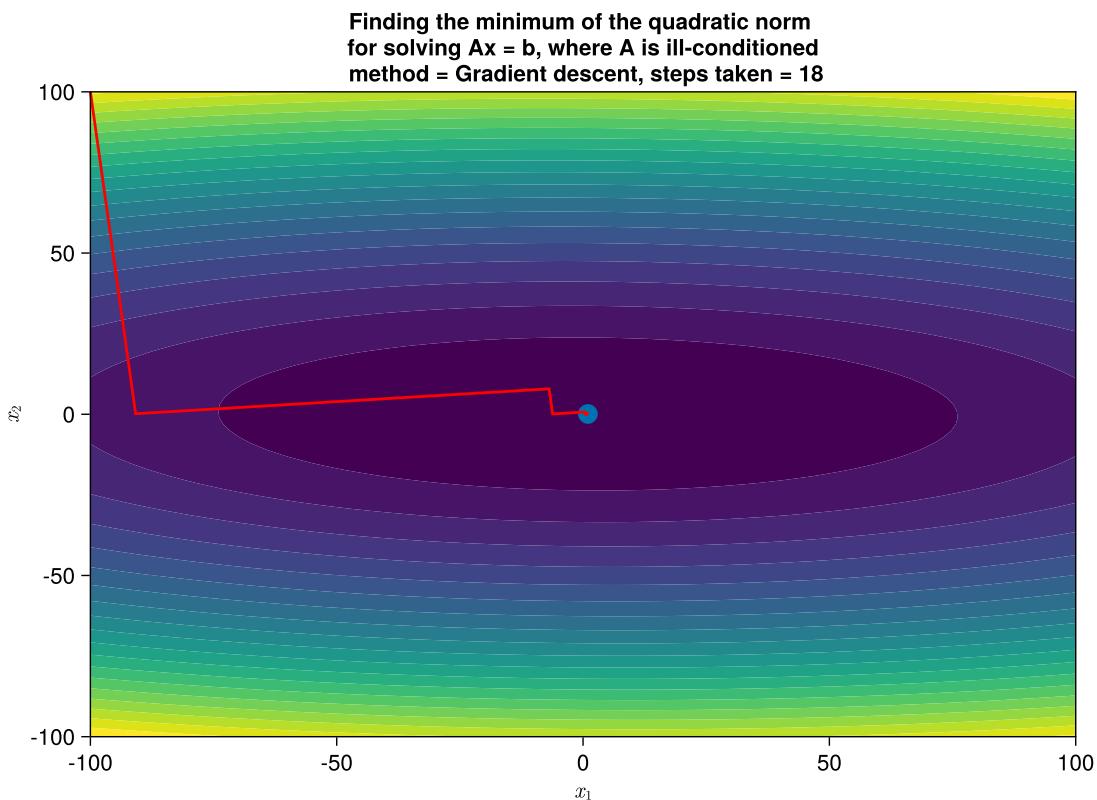


Figure 86: Steepest descent for solving a linear system $\underline{Ax} = \underline{b}$ with positive definite matrix A .

Idea: Is there maybe a smarter way to choose directions $\underline{p}^{(n)}$ so that we do not go into the same direction multiple times and still find the solution?

We will introduce the conjugate gradient method, which takes at most N (size of the linear system) steps to converge to the solution, see figure 87.

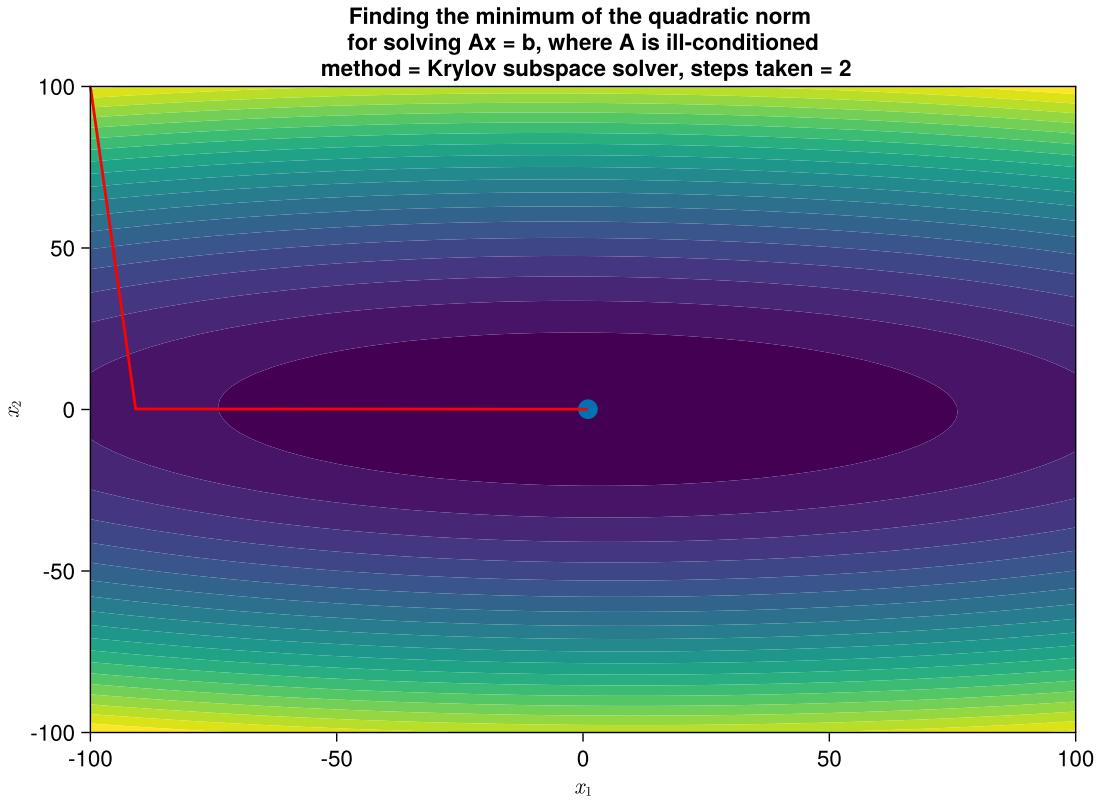


Figure 87: Conjugate gradient method for solving a linear system $\underline{A}\underline{x} = \underline{b}$ with positive definite matrix \underline{A} .

10.8.3 Krylov subspace and construction of iterative methods for solving $\underline{A}\underline{x} = \underline{b}$

The Krylov subspace of order r is based on

$$\underline{b}, \underline{A}\underline{b}, \underline{A}^2\underline{b}, \dots, \underline{A}^{r-1}\underline{b} \quad (497)$$

forming a linear vector space

$$\mathcal{K}_r := \mathcal{K}_r(\underline{A}, \underline{b}) = \text{span}(\underline{b}, \underline{A}\underline{b}, \underline{A}^2\underline{b}, \dots, \underline{A}^{r-1}\underline{b}) \quad (498)$$

note that

- $\underline{A}^i\underline{b}$ can be calculated by i matrix-vector products (very efficient)
- $\underline{A}^i\underline{b}$ are linearly independent for each $i < N$, the dimension of \mathcal{K}_r increases with r up to N , the dimension of \underline{A}

How can we smartly choose $\underline{x}^{(n)} \in \mathcal{K}_n$ in an iterative scheme for solving $\underline{A}\underline{x} = \underline{b}$?

- so that the residual $\underline{r}^{(n)} = \underline{b} - \underline{A}\underline{x}^{(n)}$ is orthogonal to \mathcal{K}_n (so $\in \mathcal{K}_{n+1}$) (**conjugate**

gradients)

- so that the residual has minimum norm for $\underline{x}^{(n)} \in \mathcal{K}_n$
- ...

Note: The best basis for the Krylov subspace \mathcal{K}_r is orthonormal and can be computed using Arnoldi's method (essentially Gram-Schmidt). Since $\underline{r}^{(n)} \perp \mathcal{K}_n$ it will be a scaled version of the last orthonormal base vector, one needs to get from \mathcal{K}_n to \mathcal{K}_{n+1}

10.8.4 Conjugate gradients method

The sequence

$$\underline{x}^{(n+1)} = \underline{x}^{(n)} + \alpha^{(n)} \underline{p}^{(n)}, \quad \text{residual } \underline{r}^{(n)} = \underline{b} - \underline{\underline{A}}\underline{x}^{(n)} \quad (499)$$

with (without proof)

$$\underline{p}^{(n)} = \underline{r}^{(n)} + \frac{\underline{r}^{(n)} \cdot \underline{r}^{(n)}}{\underline{r}^{(n-1)} \cdot \underline{r}^{(n-1)}} \underline{p}^{(n-1)} \quad (500)$$

and *learning rate*²¹

$$\alpha^{(n)} = -\frac{(\underline{p}^{(n)})^T \underline{r}^{(n)}}{(\underline{p}^{(n)})^T \underline{\underline{A}} \underline{p}^{(n)}} \quad (501)$$

solves $\underline{\underline{A}}\underline{x} = \underline{b}$, $\underline{\underline{A}} \in \mathbb{R}^{N \times N}$, $\underline{\underline{A}}$ symmetric and positive definite, in at most N steps, so $\underline{r}^{(N)} = \underline{0}$ (here stated without proof).

The directions taken are constructed so that

- the residuals are orthogonal, $\underline{r}^{(i)} \cdot \underline{r}^{(j)} = 0$ for $i \neq j$
- the step-directions are conjugate, $(\underline{p}^{(j)})^T \underline{\underline{A}} \underline{p}^{(j)} = 0$ for $i \neq j$
- the residuals and step-directions are mutually orthogonal, $\underline{p}^{(i)} \cdot \underline{r}^{(j)} = 0$ for $i \neq j$

²¹This follows simply from $\partial_x f|_{\underline{x}^{(n)} + \alpha^{(n)} \underline{p}^{(n)}} = 0$, where f is the quadratic form of $\underline{\underline{A}}\underline{x} = \underline{b}$.

11 Fourier methods

Derivatives in normal space turn into multiplications in Fourier space - we can convert a PDE into an algebraic equation by Fourier transform. Other applications of Fourier methods are

1. calculate correlation functions
2. projection of vector operators
3. diagonalize circulant matrices
4. image smoothing

11.1 Convolution problems | solving Poisson's equation using Fourier methods

We want to solve Poisson's equation

$$\text{gravitational } \nabla^2 \phi = 4\pi G\rho, \quad \text{electrostatics } \nabla^2 \phi = -4\pi\rho \quad (502)$$

for a given density distribution ρ (charge density in electrostatics) (here in CGS units).

Idea: The solution to the Poisson equation is generally a convolution^a, which by the convolution theorem becomes a multiplication in Fourier space - a very fast operation $\mathcal{O}(N)$. So the main cost is the Fourier transform, which can be done in $\mathcal{O}(N \log N)$.

^aA convolution of a Greens function and the density.

11.1.0.1 The solution to the Poisson equation is a convolution

For a point source q at the origin, the density distribution is $\rho(\underline{x}) = q\delta(\underline{x})$ and we know the potential to be $\phi = \frac{q}{|\underline{x}|}$, so in the Poisson equation $\nabla^2 \frac{1}{|\underline{x}|} = -4\pi\delta(\underline{x})$.

As of the linearity of Poisson's equation, the solution to a more complex charge distribution is

$$\phi(\underline{x}) = \sum_{i=1}^N \frac{q_i}{|\underline{x} - \underline{x}_i|} \quad \underbrace{\rightarrow}_{\text{continuous limit}} \quad \phi(\underline{x}) = \phi(\underline{x}) = \int \frac{\rho(\underline{x}')}{|\underline{x} - \underline{x}'|} d\underline{x}' \quad (503)$$

or in the gravitational case

$$\phi(\underline{x}) = -G \int \frac{\rho(\underline{x}')}{|\underline{x} - \underline{x}'|} d\underline{x}' = \int g(\underline{x} - \underline{x}') \rho(\underline{x}') d\underline{x}' = g \star \rho$$

Greens function of gravity $g(\underline{x}) = -\frac{G}{|\underline{x}|}$

(504)

- the solution for the potential is the convolution of the density and the Greens function g .

11.1.0.2 A convolution in real space turns into a multiplication in Fourier space

The Fourier transform of the convolution of two functions is equal to the product of the individual Fourier transforms of those functions (**convolution theorem**).

$$\mathcal{F}(f \star g) = \mathcal{F}(f) \cdot \mathcal{F}(g), \quad \text{functions } f, g, \quad \text{Fourier transform } \mathcal{F} \quad (505)$$

We can turn a convolution in real space into a simple point-by-point multiplication in Fourier space. Applied to the Poisson problem, we get

$$\phi = \mathcal{F}^{-1}[\mathcal{F}(g) \cdot \mathcal{F}(\rho)], \quad \text{in Fourier space } \mathcal{F}(\phi) =: \hat{\phi}(\underline{k}) = \hat{g}(\underline{k}) \cdot \hat{\rho}(\underline{k}) \quad (506)$$

11.1.0.3 For periodic boundaries, the Fourier transform turns from an integral to a sum

Assume our space to be a box of size L in all dimensions and the boundary conditions to be periodic.

As of the periodic boundary conditions, the set of possible k -vectors is discrete, as illustrated in figure 88.

$$\rho(\underline{x}) = \sum_{\underline{k}} \rho_{\underline{k}} \exp(i \underline{k} \cdot \underline{x})$$

periodic boundary conditions $\rightarrow \underline{k} \in \frac{2\pi}{L} \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix}, \quad n_1, n_2, n_3 \in \mathbb{Z}$

as of $\rho \in \mathbb{R} (\rho^* = \rho) \rightarrow \hat{\rho}_{\underline{k}} = \hat{\rho}_{-\underline{k}}^*$

(507)

We have an infinite but discrete grid in reciprocal space.

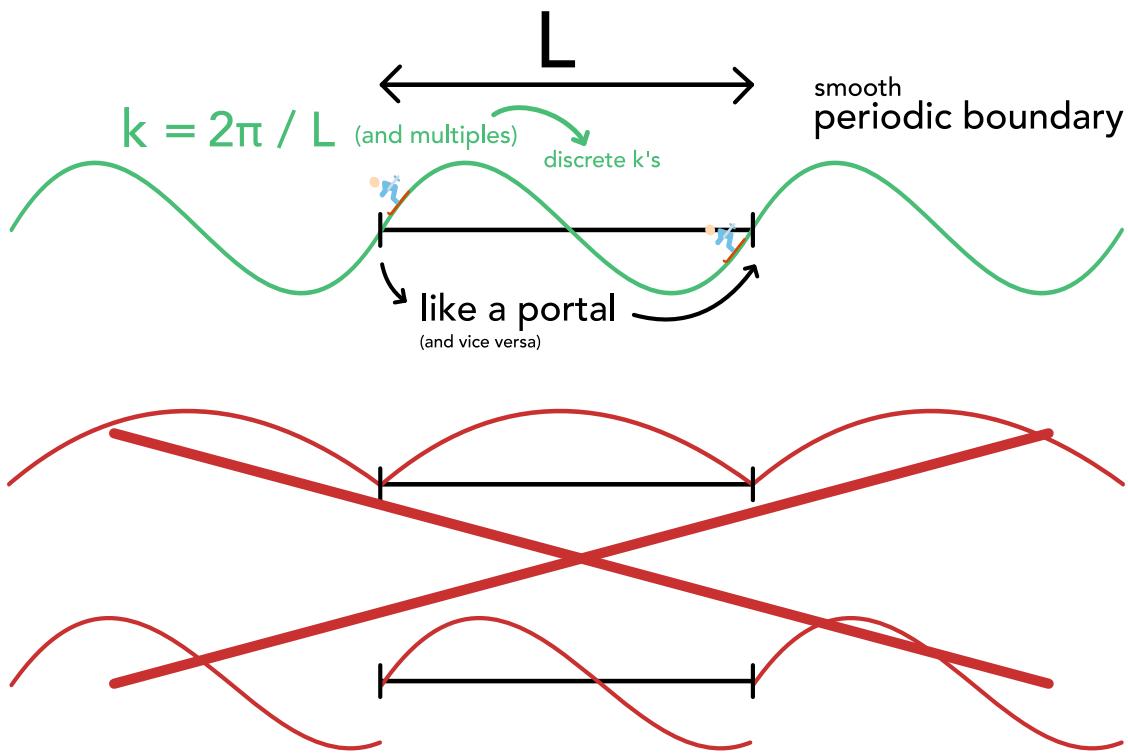


Figure 88: The set of possible k -vectors is discrete for periodic boundary conditions.

The Fourier transform over the periodic density in real space is

$$\hat{\rho}_{\underline{k}} = \frac{1}{L^3} \int_V \rho(\underline{x}) \exp(-i\underline{k} \cdot \underline{x}) d\underline{x}, \quad \text{normalization } \frac{1}{L^3} \quad (508)$$

General properties of the periodic Fourier series:

$$\begin{aligned} &\text{orthogonality } \frac{1}{L^3} \int_V \exp(-i(\underline{k} - \underline{k}') \cdot \underline{x}) d\underline{x} = \delta_{\underline{k}, \underline{k}'}, \\ &\text{closure } \frac{1}{L^3} \sum_{\underline{k}} \exp(i\underline{k} \cdot \underline{x}) = \delta(\underline{x}) \end{aligned} \quad (509)$$

11.1.0.4 Solution to the Poisson equation in Fourier space

Replacing the density and potential by their corresponding Fourier series in the Poisson equation - assuming periodic boundaries - yields

$$\begin{aligned} \nabla^2 \sum_{\underline{k}} \phi_{\underline{k}} \exp(i\underline{k} \cdot \underline{x}) &= 4\pi G \sum_{\underline{k}} \hat{\rho}_{\underline{k}} \exp(i\underline{k} \cdot \underline{x}) \\ \rightarrow \phi_{\underline{k}} &= -\frac{4\pi G}{\underline{k}^2} \hat{\rho}_{\underline{k}} = \underline{g}_{\underline{k}} \cdot \hat{\rho}_{\underline{k}} \end{aligned} \quad (510)$$

where $\mathcal{G}_{\underline{k}}$ is the Greens function of the Poisson equation in periodic fourier space.

11.2 Discrete Fourier Transform (DFT)

Periodicity in space led to discrete wave vectors \underline{k} , discretization in space will limit the different \underline{k} necessary to fully represent a density distribution, leading to a discrete Fourier and inverse Fourier transform by summation.

On a computer, the density and potential in normal space are also discretized to the following positions

$$\underline{x}_{\underline{p}} = \frac{L}{N} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \frac{L}{N} \underline{p}, \quad \text{box length } L, \quad \text{points per dimension } N, \quad (511)$$

$$p_1, p_2, p_3 \in \{0, 1, \dots, N - 1\}, \quad \rho_{\underline{p}} = \rho(\underline{x}_{\underline{p}})$$

Using $d\underline{x} \rightarrow \left(\frac{L}{N}\right)^3$ the integral for calculating the Fourier transform of the density (eq. 508) turns into the sum

$$\hat{\rho}_{\underline{k}} = \frac{1}{N^3} \sum_{\underline{p}} \rho_{\underline{p}} \exp\left(-i\underline{k} \cdot \underline{x}_{\underline{p}}\right) \quad (512)$$

11.2.0.1 The spatial discretization (and periodicity) limits the number of \underline{k} vectors that lead to possibly different $\hat{\rho}_{\underline{k}}$

$$\underline{k} \in \frac{2\pi}{L} \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} \xrightarrow{\text{discretization of space}} \underline{k} = \frac{2\pi}{L} \begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix} = \frac{2\pi}{L} \underline{l}, \quad l_1, l_2, l_3 \in \{0, 1, \dots, N - 1\} \quad (513)$$

Why does a longer range of l_1, l_2, l_3 not introduce additional modes?: Consider the inverse Fourier transform

$$\rho(\underline{x}) = \sum_{\underline{l}} \hat{\rho}_{\underline{l}} \exp(i \underline{k}_{\underline{l}} \cdot \underline{x}_{\underline{p}}) \quad (514)$$

with $\underline{x}_{\underline{p}} = \frac{L}{N} \underline{p}$. Consider now we would introduce a further \underline{k} to the sum with e.g.

$$\underline{k} = \frac{2\pi}{L} \begin{pmatrix} l_1 + N \\ l_2 \\ l_3 \end{pmatrix} \quad (515)$$

then it will indeed not add a **new** mode to the sum 514, as

$$\exp\left(i\left(\frac{2\pi}{L}l_1 + \frac{2\pi N}{L}\right)\frac{L}{N}p_1\right) = \exp\left(i\frac{2\pi}{L}l_1\frac{L}{N}p_1\right) \underbrace{\exp\left(i\frac{2\pi N}{L}\frac{L}{N}p_1\right)}_{=1} \quad (516)$$

A finite number of wave vectors is sufficient to describe a density distribution on a discretized grid - higher *frequencies* are aliased to lower ones.

11.2.0.2 Resulting discrete Fourier transform

$$\text{Fourier space } \hat{\rho}_{\underline{l}} = \frac{1}{N^3} \sum_{\underline{p}} \rho_{\underline{p}} \exp\left(-i \frac{2\pi}{N} \underline{l} \cdot \underline{p}\right)$$

$$\text{Real space } \rho_{\underline{p}} = \sum_{\underline{l}} \hat{\rho}_{\underline{l}} \exp\left(i \frac{2\pi}{N} \underline{l} \cdot \underline{p}\right)$$

$$\underline{l}, \underline{p} \in \begin{pmatrix} \{0, 1, \dots, N-1\} \\ \{0, 1, \dots, N-1\} \\ \{0, 1, \dots, N-1\} \end{pmatrix}$$

(517)

where (in 3D) we invertibly and linearly map N^3 values $\rho_{\underline{p}}$ to N^3 values $\hat{\rho}_{\underline{l}}$.

11.2.0.3 Different conventions for the wave vector \underline{k} and Nyquist frequency

The wave vectors \underline{k} (in the general context of DFT also called *frequencies*) are conventionally defined as

$$\underline{k} = \frac{2\pi}{L} \begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix} = \frac{2\pi}{L} \underline{l}, \quad l_1, l_2, l_3 \in \left\{-\frac{N}{2}, \dots, -1, 0, 1, \dots, \frac{N}{2}-1\right\} \quad (518)$$

Note: This does not make a difference compared to $l_1, l_2, l_3 \in 0, 1, \dots, N - 1$ as shifts by $\frac{2\pi N}{L}$ in an entry of \underline{k} do not lead to new unique modes.

This shift makes the occurrence of positive and negative wave vectors explicit and the \underline{k} -vectors are now arranged quasi-symmetrically around $\underline{k} = (0, 0, 0)^T$, as illustrated in figure 89.

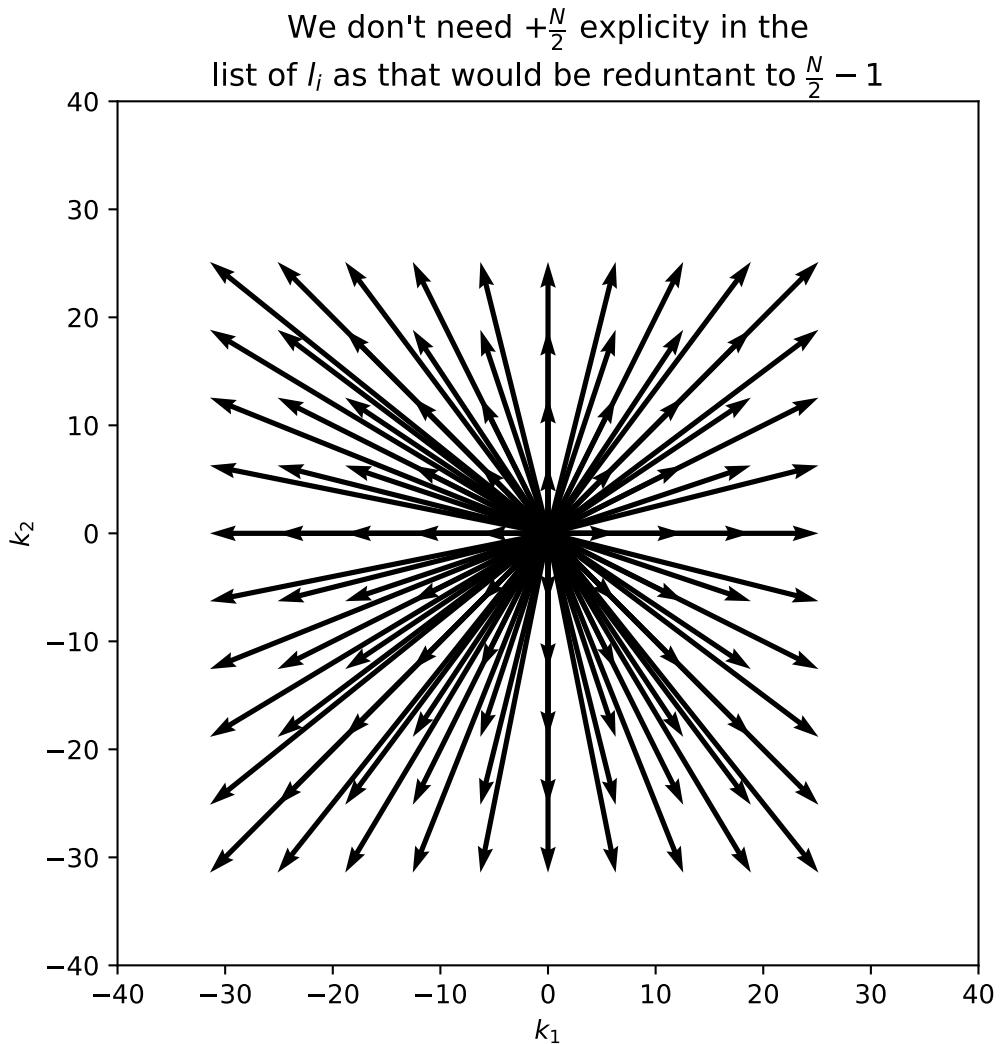


Figure 89: The wave vectors \underline{k} are conventionally defined as $\underline{k} = \frac{2\pi}{L} \begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix} = \frac{2\pi}{L} \underline{l}$ with $l_1, l_2, l_3 \in \left\{-\frac{N}{2}, \dots, -1, 0, 1, \dots, \frac{N}{2} - 1\right\}$.

Higher frequencies than the

$$\text{Nyquist frequency } k_{\max} = \frac{N}{2} \frac{2\pi}{L} \quad (519)$$

cannot be represented unambiguously on a grid with N points per dimension, instead they are aliased to lower frequencies, as illustrated in figure 90.

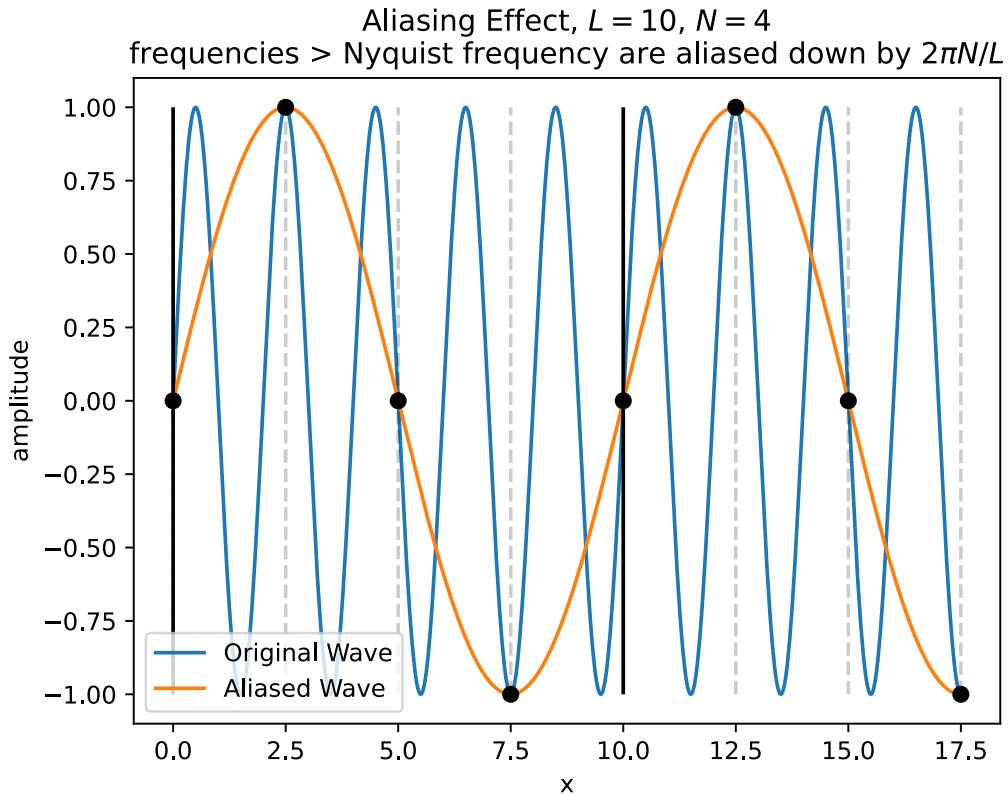


Figure 90: Higher frequencies than the Nyquist frequency $k_{\max} = \frac{N}{2} \frac{2\pi}{L}$ cannot be represented unambiguously on a grid with N points per dimension, instead they are aliased to lower frequencies.

11.2.0.4 Plancherel's theorem

Plancherel's theorem makes the following relation

$$\sum_{\underline{p}} |\rho_{\underline{p}}|^2 = N^3 \sum_l |\hat{\rho}_l|^2 \quad (520)$$

- a function and its Fourier transform have the same L2 norm. We can therefore decide if we want to calculate this L2 norm in real or Fourier space.

11.2.0.5 Normalization factor

The $\frac{1}{N^3}$ normalization is set arbitrarily (could also be split between both equations), often omitted in numerical packages.

Synthesis	Analysis
Given $\hat{z}_0, \dots, \hat{z}_{N-1} \in \mathbb{Z}$, we search for the $z_i = \sum_{j=0}^{N-1} \hat{z}_j \omega_N^{ij} \quad \forall i \in [0 : N-1] \quad (521)$	Given $z_0, \dots, z_{N-1} \in \mathbb{Z}$, we search for the $\hat{z}_j = \sum_{i=0}^{N-1} z_i \omega_N^{*ij} \quad \forall i \in [0 : N-1] \quad (522)$

Table 16: Synthesis and analysis equations for the Fourier transform. ω_N are the square roots of unity (as $\omega_N^N = 1$), $\omega_N = \exp\left(\frac{2\pi\imath}{N}\right)$, where \imath is the imaginary unit (to not confuse it with the index i). $\omega_N^{ij} = \exp\left(\frac{2\pi\imath}{N}ij\right)$. This can also be written as the matrix equation $\underline{z} = \underline{\underline{F}}\hat{\underline{z}}$ with the Fourier matrix $\underline{\underline{F}}$, where $F_{ij} = \omega_N^{ij}$.

Commonly the form of table 16 is used.

Note: In this case $\mathcal{F}^{-1}(\mathcal{F}(\underline{z})) = N^3 \underline{z}$

Higher dimensional transforms are just the cartesian products of one-dimensional transforms.

11.2.1 Computational Complexity and Fast Fourier Transform (FFT)

In the naive approach, to calculate all z_0, \dots, z_{N-1} (with each calculation being $\mathcal{O}(N)$) we would need $\mathcal{O}(N^2)$ operations.

FFT-idea in a nutshell: The central ideas are to *is divide and conquer* and smartly use the unit square root. Let N be a power of 2 and $N = 2m$. Then $\omega_N^2 = \omega_m$. We can then split the sum

$$z_i = \sum_{j=0}^{N-1} \hat{z}_j \omega_n^{ij} = \underbrace{\sum_{k=0}^{m-1} \hat{z}_{2k} \omega_m^{ik}}_{:= x_i} + \omega_N^i \underbrace{\sum_{k=0}^{m-1} \hat{z}_{2k+1} \omega_m^{ik}}_{:= y_i}, \quad \text{using } \omega_N^{i+1} = \omega_n \omega_N^i \quad (523)$$

and find

$$z_i = x_i + \omega_N^i y_i, \quad z_{i+m} = x_i - \omega_N^i y_i \quad (524)$$

so by recursion we can construct the z_i in $\mathcal{O}(N \log N)$ operations.

11.3 DFT storage conventions

Let us get back to the frame of the DFT in the form

$$\text{Fourier space } \hat{\rho}_{\underline{l}} = \frac{1}{N^3} \sum_{\underline{p}} \rho_{\underline{p}} \exp \left(-i \frac{2\pi}{N} \underline{l} \cdot \underline{p} \right)$$

$$\text{Real space } \rho_{\underline{p}} = \sum_{\underline{l}} \hat{\rho}_{\underline{l}} \exp \left(i \frac{2\pi}{N} \underline{l} \cdot \underline{p} \right)$$

$$\underline{l} \in \begin{pmatrix} \left\{ -\frac{N}{2}, \dots, -1, 0, 1, \dots, \frac{N}{2} - 1 \right\} \\ \left\{ -\frac{N}{2}, \dots, -1, 0, 1, \dots, \frac{N}{2} - 1 \right\} \\ \left\{ -\frac{N}{2}, \dots, -1, 0, 1, \dots, \frac{N}{2} - 1 \right\} \end{pmatrix}, \quad \underline{p} \in \begin{pmatrix} \{0, 1, \dots, N - 1\} \\ \{0, 1, \dots, N - 1\} \\ \{0, 1, \dots, N - 1\} \end{pmatrix}$$

(525)

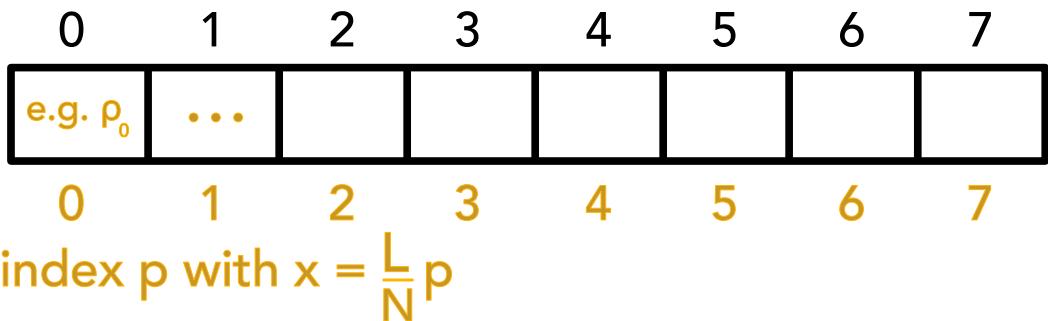
Both the original data and the Fourier transform are stored as simple arrays (1D Fourier). Negative *frequencies* are stored *backwards*. A 1D illustration is given in figure 91. A 2D illustration is given in figure 92. One can calculate

$$\# \text{ independent numbers in 2D grid} = 2 \left(\frac{N}{2} - 1 \right)^2 \cdot 2 + 4 \left(\frac{N}{2} - 1 \right) \cdot 2 + 4 = N^2 \quad (526)$$

DFT storage convention

real space array $N = 8$

array index



Fourier transformed array

array index

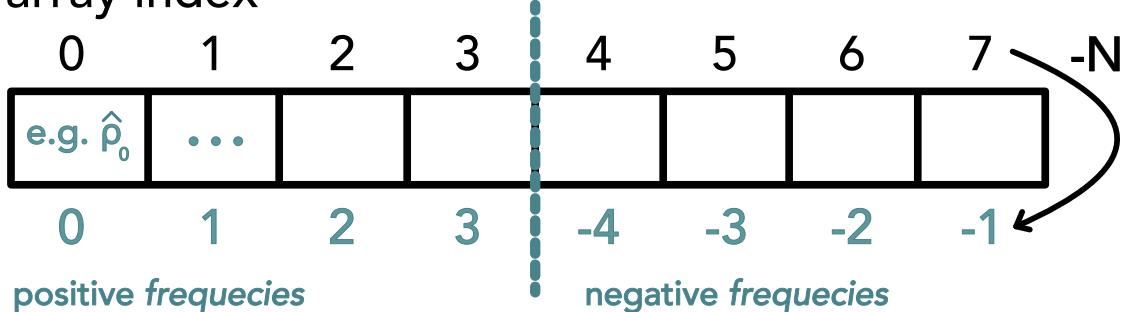


Figure 91: Illustration of the storage of the original data and the Fourier transform in 1D.

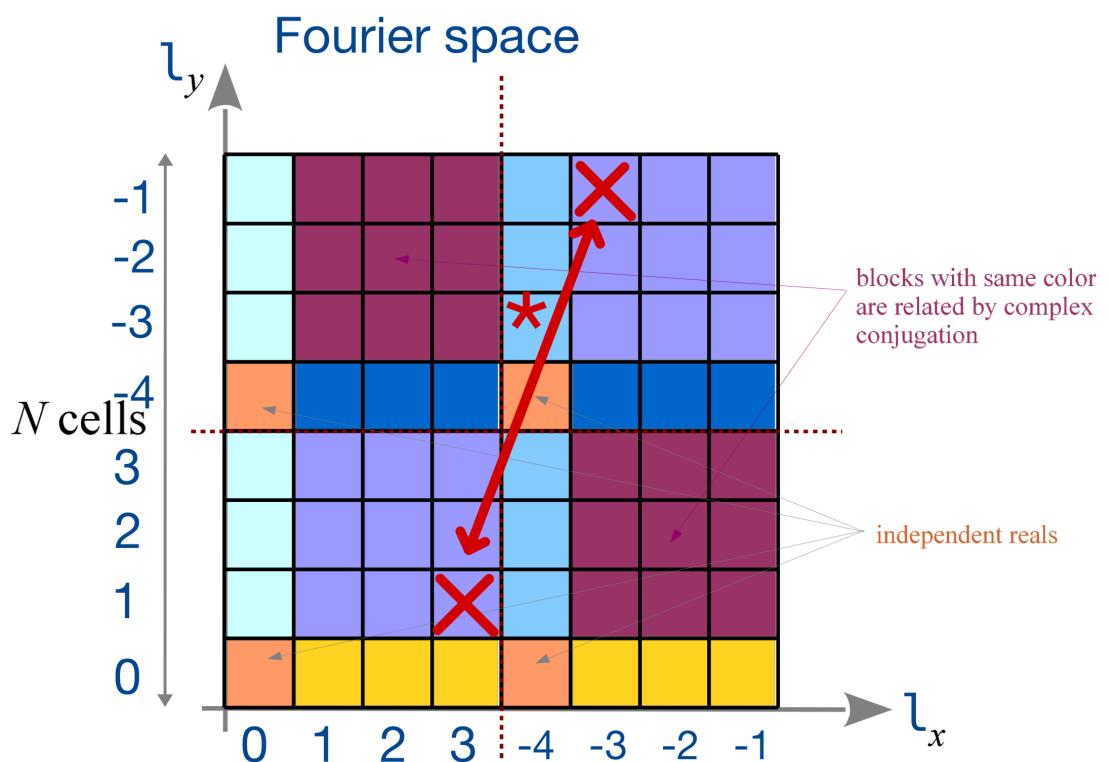


Figure 92: 2D fourier space storage convention. Note that if the data in real space is real $\rho_{\underline{l}} \in \mathbb{R}$, then $\hat{\rho}_{\underline{l}} = \hat{\rho}_{-\underline{l}}^*$. Because of aliasing $\hat{\rho}_{-\frac{N}{2}} = \hat{\rho}_{\frac{N}{2}} = \hat{\rho}_{-\frac{N}{2}}^*$ so $\in \mathbb{R}$, marked in orange in the illustration. We have (a complex number consists of two independent real numbers) the same number of independent numbers as in real space.

11.4 DFT and Linear and Cyclic Convolution

Let us step back to the convolution of two functions - e.g. for solving the Poisson equation or for smoothing an image. The convolution is defined as

$$(f \star g)(t) = \int_{\mathbb{R}} f(\tau)g(t - \tau)d\tau \quad (527)$$

so naturally for discrete f, g , we define

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m] \quad (528)$$

The basic intuition for convolutions is that one function is flipped and slides over the other as a window, as illustrated in figure 93.

Note: When one convolves using $\text{ifft}(\text{fft}(f) \cdot \text{fft}(g))$ this is actually the cyclic convolution

$$(f * g_N)[n] = \sum_{m=0}^{N-1} f[m]g_N[n - m] \quad (529)$$

assuming f, g are periodic with period N . The wanted result from a linear convolution is illustrated in figure 94, what we get from the cyclic convolution is illustrated in figure 95. Code for the cyclic convolution is given in code 3.

Idea: If we zero-pad sufficiently, the cyclic convolution becomes a linear convolution. For two vectors \underline{f} of length N and \underline{g} of length M we append zeros to both so they are both of length $N + M - 1$ (the length of the linear convolution), see figure 96 and code 4.

```
1      # cyclic convolution in real space
2      function DirectCyclicConv1D(f,g)
3          N = length(f)
4          Conv = zeros(N)
5
6          for n in 1:N, m in 1:N
7              if n-m+1 > 0
8                  Conv[n] = Conv[n] + f[m] * g[n-m+1]
9              else
10                  # make g periodic
11                  Conv[n] = Conv[n] + f[m] * g[N+(n-m+1)]
12              end
13          end
14
15          return Conv
16
17      end
18
19      # cyclic convolution in Fourier space
20      FastCyclicConv1D(f,g) = ifft(fft(f).*fft(g))
```

Code-Snippet 3: Cyclic convolution in Julia.

Convolution

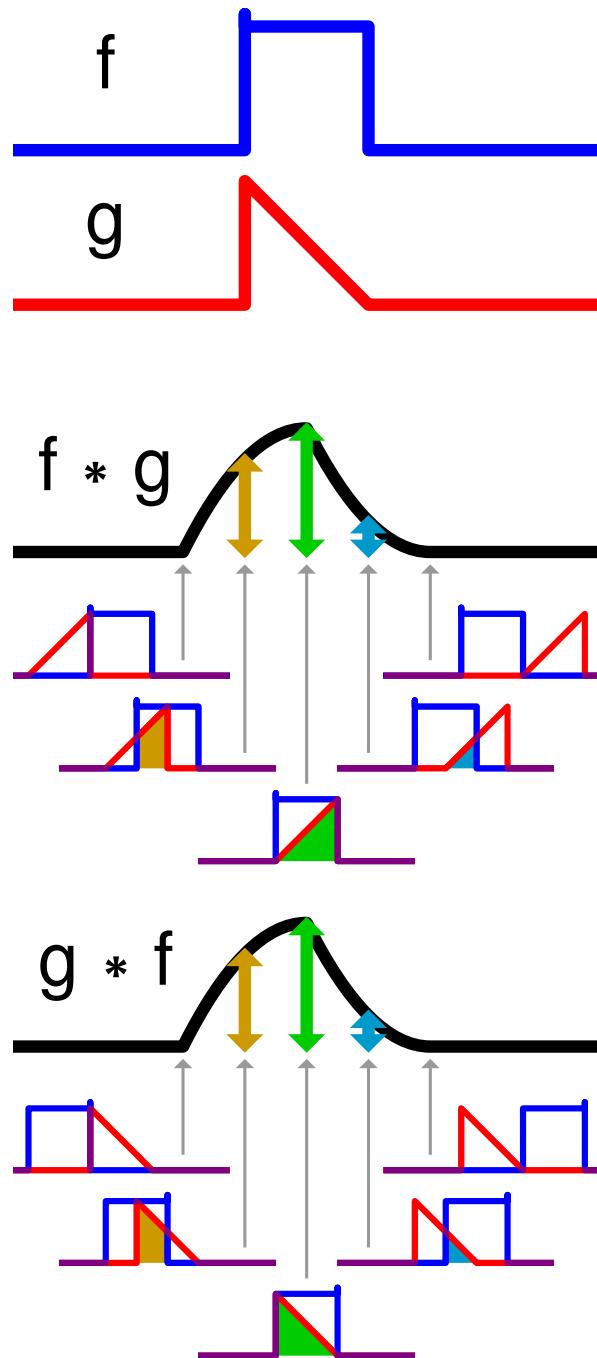


Figure 93: Basic intuition for convolutions.

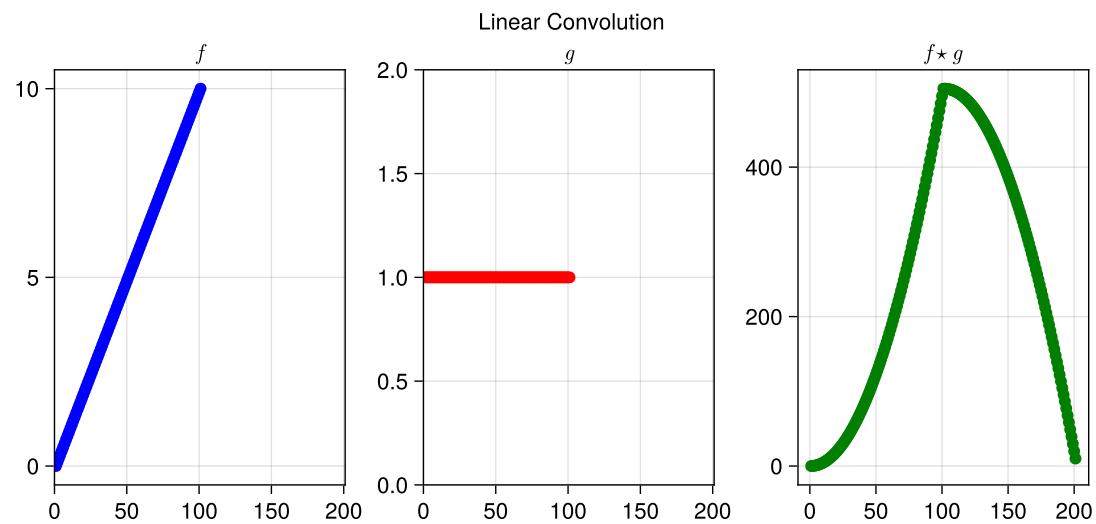


Figure 94: Example of a linear convolution.

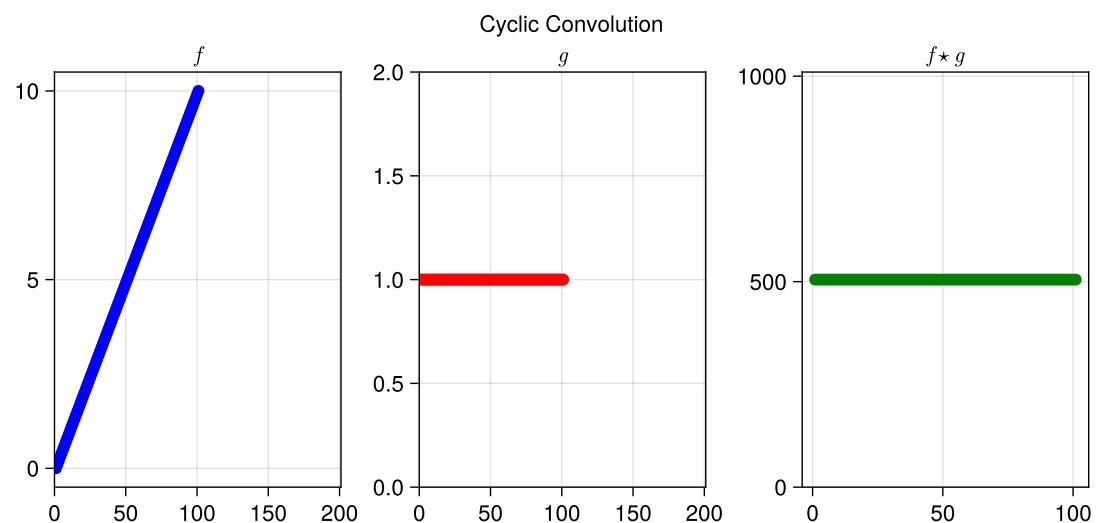


Figure 95: Example of a cyclic convolution.

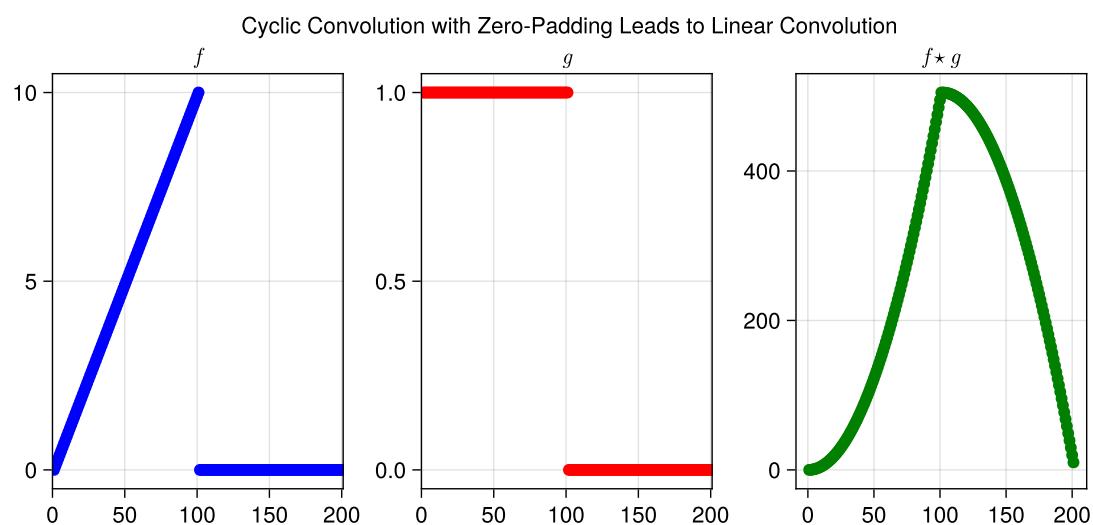


Figure 96: Illustration of the cyclic convolution becoming a linear convolution by zero-padding.

```

1      # simple linear convolution in real space
2      function conv1d(f, g)
3          n = length(f)
4          m = length(g)
5          y = zeros(n + m - 1)
6          for i in 1:n
7              for j in 1:m
8                  y[i + j - 1] += f[i] * g[j]
9              end
10         end
11         return y
12     end
13
14     # zero padded linear convolution in real space
15     # note that this zero padded version is equivalent
16     # to the cyclic convolution with zero padding
17     function DirectLinearConvolution(f,g)
18         N = length(f)
19         M = length(g)
20
21         g_pad = [ g; zeros(N-1) ]      # length N+M-1
22         f_pad = [ f; zeros(M-1) ]      # length N+M-1
23
24         Conv = zeros(N+M-1)
25         for n=1:N+M-1
26             for m=1:N+M-1
27                 if n-m+1 > 0
28                     Conv[n] = Conv[n] + f_pad[m] * g_pad[n-m+1]
29                 end
30                 # n+1 <= m
31             end
32         end
33         return Conv
34     end
35
36     # fast linear convolution in Fourier space
37     function FastLinearConvolution(f,g)
38         N = length(f)
39         M = length(g)
40         f_pad = [ f; zeros(M-1) ]
41         g_pad = [ g; zeros(N-1) ]
42         return FastCyclicConv1D( f_pad, g_pad )
43     end

```

Code-Snippet 4: Linear convolution in Julia.

11.5 Non-periodic problems with *zero-padding* in 2D

Problem: DFT / FFT are defined for periodic problems, a convolution based on them is a cyclic convolution.

Idea: As before we can zero-pad. *The cost, however, rises ($\times 4$ in 2D).*

For solving the Poisson equation for non-periodic density distributions, we

1. set up the mesh so that the density distribution only lives in one quarter - rest zeroes (see figure 97).
2. periodically extend the Green's function over the whole grid (so that the distance used is the one to the nearest periodic image)

$$g_{N-i,j} = g_{i,N-j} = g_{N-i,N-j} = g_{i,j}, \quad 0 \leq i, j \leq \frac{N}{2} \quad (530)$$

which is symmetric around the origin (when replicating the mesh in all directions).

Why does the Greens function have to be extended?

3. calculate $\phi = g \star \rho$, in real space this would be

$$\phi_{\underline{p}} = \sum_{\underline{n}} g_{\underline{p}-\underline{n}} \rho_{\underline{n}}, \quad g, \rho \text{ periodic in the index } N \quad (531)$$

The zero padding is large enough so that if *the Green's function slides over the density, there is no cross talk between periodic images of the density distribution*. We use the fast cyclic convolution in Fourier space.

Zero-padding → linear (non-cyclic) convolutions

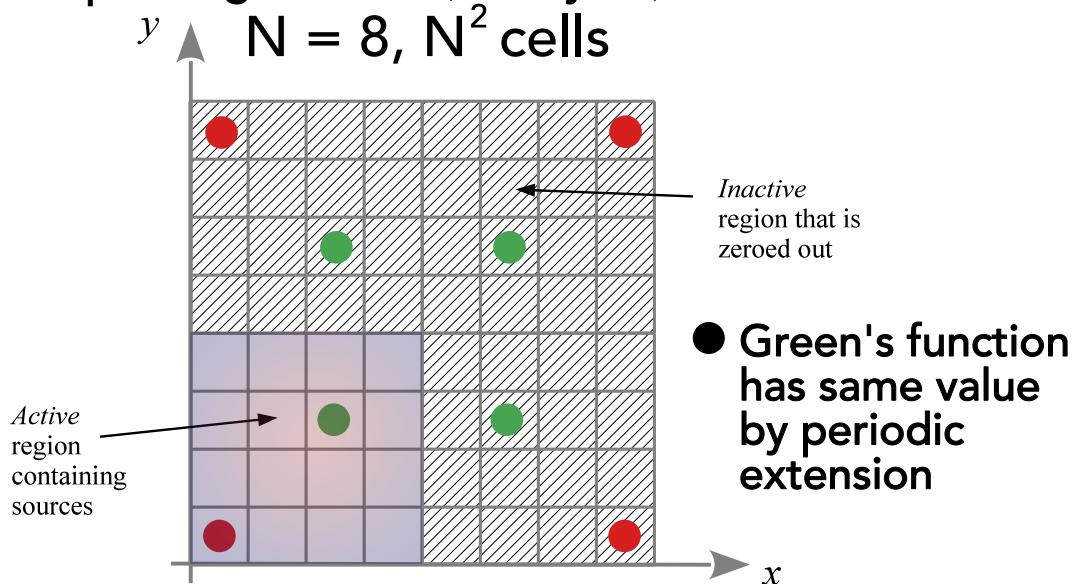


Figure 97: Illustration of the zero padding in 2D.

11.6 Power spectra and correlation functions

Power spectra and correlation functions are deeply connected via the Fourier transform. Assume the field ρ to be isotropic and homogeneous (mean or variance do not change under rotations and translations).

11.6.1 Definition of the power spectrum

Consider the fourier transform (decomposing the scalar field ρ into plane waves)

$$\rho(\underline{x}) = \frac{1}{(2\pi)^3} \int \hat{\rho}(\underline{k}) e^{-i\underline{k} \cdot \underline{x}} d^3 k, \quad \hat{\rho}(\underline{k}) = \int \rho(\underline{x}) e^{i\underline{k} \cdot \underline{x}} d^3 \underline{x} \quad (532)$$

the variance in Fourier space defines the power spectrum $P(k)$

$$\langle \hat{\rho}(\underline{k}) \hat{\rho}^*(\underline{k}') \rangle \equiv (2\pi)^3 P(k) \delta(\underline{k} - \underline{k}'), \quad k = |\underline{k}|, \quad (\text{isotropic}) \text{ power spectrum } P \quad (533)$$

- modes with different wave vector are uncorrelated (ensured by Delta distribution) to ensure homogeneity. To ensure isotropy, the power spectrum P cannot depend on the direction of \underline{k} .

Usually the energy spectral density for a measurement $x(t)$ over time is just $\bar{S}_{xx}(f) \triangleq |\hat{x}(f)|^2$ where f is a frequency (Fourier transform in time) and the general signal processing understanding of energy as $E \triangleq \int_{-\infty}^{\infty} |x(t)|^2 dt$ is used (which based on Plancherel's theorem $\int_{-\infty}^{\infty} |x(t)|^2 dt = \int_{-\infty}^{\infty} |\hat{x}(f)|^2 df$ makes the formula for the *spectral* energy density (in a small frequency interval) plausible).

11.6.2 Definition of the correlation function

The auto-correlation function

$$\xi(\underline{y}) = \langle \rho(\underline{x})\rho(\underline{x} + \underline{y}) \rangle \quad (534)$$

average over all positions \underline{x} and orientations of \underline{y}

measures the coherence of the fluctuating field ρ at all points separated by \underline{y} (can also not depend on the direction of \underline{y} as of isotropy). For $\underline{y} = 0$ we have the variance.

11.6.3 Connection by Fourier Transform

Correlation function and power spectrum are Fourier transforms of each other (can be seen by inserting the Fourier transforms into the correlation function), hence carry an equivalent amount of information.

From this we can find the variance of ρ in real space (at $\underline{y} = 0$) to be

$$\sigma^2 = \frac{4\pi}{(2\pi)^3} \int P(k)k^2 dk \quad (535)$$

11.6.4 Power function and variance of a smoothed field

Consider the smoothed field

$$\bar{\rho}(\underline{x}) = \int \rho(\underline{y})W_R(||\underline{x} - \underline{y}||) d^3\underline{y}, \quad \text{window with compact support } R \quad (536)$$

which is a convolution. As of the convolution theorem $\widehat{f \star g} = \hat{f}\hat{g}$, we have

$$\hat{\rho}(\underline{k}) = \hat{\rho}(\underline{k})\hat{W}_R(\underline{k}) \quad (537)$$

so the power spectrum (as of eq. 533) of the smoothed field is

$$\bar{P}(k) = P(k)\hat{W}_R^2(k) \quad (538)$$

which can be used to calculate a variance of the smoothed field using eq. 535.

11.7 Projections in Fourier space

Splitting a vector field into a source-free rotational and a curl-free part can be useful (e.g. for cleaning divergence from a numerically obtained magnetic field) - this can be done and proven in Fourier space.

One can proof the Helmholtz decomposition (fundamental theorem of vector analysis) using Fourier transform.

11.7.1 Fundamental theorem of vector analysis | Helmholtz decomposition

Let $\underline{F}(\underline{x})$ be a vector field with compact support and derivatives that vanish at infinity.

Then \underline{F} can be uniquely decomposed into a curl-free (longitudinal, purely radial in Fourier space) and a divergence-free part

$$\begin{aligned}\underline{F}(\underline{x}) &= \underline{F}_\perp(\underline{x}) + \underline{F}_\parallel(\underline{x}) \\ \nabla \cdot \underline{F}_\perp &= 0, \quad \nabla \times \underline{F}_\parallel = 0\end{aligned}\tag{539}$$

11.7.2 Proof in Fourier space

Consider a decomposition of the Fourier transform $\hat{\underline{F}}(\underline{k})$ of $\underline{F}(\underline{x})$ into a part parallel and perpendicular to \underline{k} (the wave vector)

$$\begin{aligned}\hat{\underline{F}}(\underline{k}) &= \hat{\underline{F}}_\perp(\underline{k}) + \hat{\underline{F}}_\parallel(\underline{k}) \\ \hat{\underline{F}}_\parallel(\underline{k}) &= \hat{\underline{e}}_k \left(\hat{\underline{e}}_k \cdot \hat{\underline{F}}(\underline{k}) \right), \quad \hat{\underline{e}}_k = \frac{\underline{k}}{\|\underline{k}\|} \\ \hat{\underline{F}}_\perp(\underline{k}) &= \hat{\underline{F}}(\underline{k}) - \hat{\underline{F}}_\parallel(\underline{k})\end{aligned}\tag{540}$$

In Fourier space divergence and curl are simple to calculate (they become scalar products and cross products with \underline{k} respectively).

$$\underline{k} \cdot \hat{\underline{F}}_\perp(\underline{k}) = 0, \quad \underline{k} \times \hat{\underline{F}}_\parallel(\underline{k}) = 0\tag{541}$$

from which by inverse Fourier transform the Helmholtz decomposition follows.

11.7.3 Applications of the Helmholtz decomposition

- Retain $\nabla \cdot \underline{B} = 0$ in simulations by projecting out the radial component (the one with divergence in real space) in Fourier space ($\hat{\underline{B}}(\underline{k}) \rightarrow \hat{\underline{B}}(\underline{k}) - \hat{\underline{e}}_k \hat{\underline{e}}_k \cdot \hat{\underline{B}}(\underline{k})$, then apply inverse Fourier transform to get the source-canceled field).
- Analyze turbulent flow for shocks
- Stability analysis in astrophysics: Based on determining how much compressive motion and how much solenoidal motion is present, stability can be analyzed. Solenoidal motion stabilizes, compressive motion increases density so astrophysically the cooling rate of the system ($\propto \rho^2$) and finally leads to collapse. This can be studied by Fourier transform (problem: spatial information is non-local in Fourier space), and also based on

$$\partial_t \rho = -\rho \nabla \cdot \underline{v}, \quad \text{solenoidal } \nabla \cdot \underline{v} = 0 \rightarrow \partial_t \rho = 0 \quad (542)$$

which can be seen as a peak in the probability distribution of the density which is broader for compressive motion.

11.7.4 Sidenote: Importance of dimensionality

Depending on the dimensionality, we have

- 1D: one compressible mode
- 2D: one compressible, one source-free (solenoidal, non-compressible) mode
- 3D: one compressible, two source-free modes

In a tube with small width we mostly have compressible modes, in shallow water we can have vortices with spin perpendicular to the surface.

12 Collisionless particle systems

12.1 Introduction of collisionless systems in the context of fluid modeling

Let us widen our view to collisionless systems (*collisionless »fluids«*) and recapitulate on fluid dynamics to avoid confusion.

Important fluid concepts are

- **Boltzmann:** starting from the view of lots of interacting particles one can derive the Boltzmann equation (and we will repeat on this) in the collisionless case ($\left.\frac{df}{dt}\right|_{\text{coll}} = 0$) called Vlasov equation
- **Navier-Stokes:** based on the moments of the Boltzmann equation one can derive a continuity, momentum and energy equation - Navier-Stokes equation for a collisional and Jeans equation for a *collisionless* fluid
- **Euler:** in case viscosity and conductivity can be ignored, the Navier stokes equations can be reduced to the Euler equations
- **Fluid variables:** *fluid elements* can be characterized by six hydrodynamical variables: density ρ , fluid velocity \underline{u} (of a fluid element, not to be confused with the velocity of individual particles), pressure P and specific internal energy ϵ
- **Closure:** The Euler equations give 5 relations of the fluid variables - one continuum equation, three momentum equations (vector-equation), one energy equation; we need a further constitutive relation - almost always an equation of state $P = P(\rho, \epsilon)$
- **Perspectives:** The Eulerian view is fixed in space, in the Lagrangian view, the description co-moves with the fluid flow

where we have explored some solvers for the Navier-Stokes / Euler equation with respective closure

- **Eulerian methods:** Based on a mesh / accounting volumes, the evolution of the fluid variables is e.g. calculated based on intercell fluxes
- **Lagrangian methods:** In Smoothed Particle Hydrodynamics we introduce SPH-particles (macroscopic, describing the fluid) forwarded based on the Euler / Navier Stokes equation

Collisionless fluids are very different from our standard collisional fluids in that

- their behavior is not collision dominated, if one would want to use a description based on fluid variables, one would at least have to use a local anisotropic pressure described via a stress tensor - as there are no frequent collisions that would act to isotropize the local pressure
- no equation of state exists for a collisionless fluid, so one can never close the set of fluid equations, unless one makes a number of simplifying assumptions
- no fluid element can be defined for a collisionless fluid (one in which the continuum hypothesis would hold) - the macroscopic fluid perspective for deriving fluid laws cannot be used

Let us give examples for collisional and collisionless systems

- Systems that can be described as (collisional) fluid
 - stars - to good approximation the equation of state of a star is that of an ideal gas
 - giant gaseous planets
 - planet atmospheres
 - ...
- Collisionless systems
 - Galaxies (stellar component) - two stars in a galaxy - to a good approximation - will never collide
 - Dark matter (halos) - assumed to be collisionless

Note: Complex systems often have collisional and collisionless components: In galaxies the gaseous component can be described with classic hydrodynamic equations (enough collisions to isotropize the local pressure), but stellar and dark matter lack collisions, where one could use a local anisotropic pressure (extending classical hydrodynamic simulations) but N-body simulations have shown to be more stable. For instance Mitchell et al., 2012 write »For simulations that deal only with dark matter or stellar systems, the conventional N-body technique is fast, memory efficient and relatively simple to implement. However when extending simulations to include the effects of gas physics, mesh codes are at a distinct disadvantage compared to Smooth Particle Hydrodynamics (SPH) codes. Whereas implementing the N-body approach into SPH codes is fairly trivial, the particle-mesh technique used in mesh codes to couple collisionless stars and dark matter to the gas on the mesh has a series of significant scientific and technical limitations. These include spurious entropy generation resulting from discreteness effects [...]« and introduce a method to use »collisionless Boltzmann moment equations as a means to model the collisionless material as a fluid on the mesh«.

Model reduction for collisionless systems: While we cannot directly apply the solvers discussed so far to collisionless systems, central ideas remain important: For instance to represent the system based on artificial / fiducial particles with larger masses in very high-N systems.^a For instance in the large dark-matter-only N-body *millenium* simulation (Springel, 2005), in a cube with sidelength $2 \cdot 10^9$ lightyears, $2160^3 \approx 10^{10}$ fiducial particles were used on which 10^{18} solar masses were equally distributed.

^aA low N system would be the planets in the solar system, a medium-N system stellar clusters with $N \sim 10^2$ stars and high N systems could be globular clusters.

12.2 Structure of the following considerations

In the following we will consider

- From a phase space view, what is a collisionless system and how can it be described?
- What systems can be assumed to be collisionless?
- N-body model of collisionless systems

12.3 General N-particle ensembles | one-, two-, three, ..., particle distribution | BBKGY chain

The exact particle distribution of N particles is given by (exact Klimontovich-Dupree representation)

$$F(\underline{x}, \underline{v}, t) = \sum_{i=1}^N \delta(\underline{x} - \underline{x}_i(t)) \cdot \delta(\underline{v} - \underline{v}_i(t)) \quad (543)$$

Let p be the N particle phase space probability at time t

$$p(\underline{x}_1, \dots, \underline{x}_N, \underline{v}_1, \dots, \underline{v}_N) d\underline{x}_1 \cdots d\underline{x}_N d\underline{v}_1 \cdots d\underline{v}_N \quad (544)$$

The mean phase space density (ensemble-averaged one particle distribution) is

$$\begin{aligned} f_1(\underline{x}, \underline{v}, t) &= \langle F(\underline{x}, \underline{v}, t) \rangle = \int F(\underline{x}, \underline{v}, t) \cdot p d\underline{x}_1 \cdots d\underline{x}_N d\underline{v}_1 \cdots d\underline{v}_N \\ &\stackrel{\text{plug in } F \rightarrow N \text{ terms}}{=} N \int p(\underline{x}, \underline{x}_2, \dots, \underline{x}_N, \underline{v}, \underline{v}_2, \dots, \underline{v}_N) d\underline{x}_2 \cdots d\underline{x}_N d\underline{v}_2 \cdots d\underline{v}_N \end{aligned} \quad (545)$$

with $f_1(\underline{x}, \underline{v}, t)$ being the mean number of particles in the phase space volume $d\underline{x}d\underline{v}$ around $(\underline{x}, \underline{v})$. The form is quite natural, if we want the probability of one particle being at a certain phase-space coordinate, we have to integrate out the others (marginalize) (as we know from quantum mechanics).

Ensemble-averaged two-particle distribution: What do we expect the mean of the product of the number of particles at two phase-space coordinates $(\underline{x}, \underline{v})$ and $(\underline{x}', \underline{v}')$ to be?

$$\begin{aligned} f_2(\underline{x}, \underline{v}, \underline{x}', \underline{v}', t) &= \langle F(\underline{x}, \underline{v}, t) F(\underline{x}', \underline{v}', t) \rangle \\ &= N \cdot (N - 1) \int p(\underline{x}, \underline{x}', \underline{x}_3, \dots, \underline{x}_N, \underline{v}, \underline{v}', \underline{v}_3, \dots, \underline{v}_N) d\underline{x}_3 \cdots d\underline{x}_N d\underline{v}_3 \cdots d\underline{v}_N \end{aligned} \quad (546)$$

f_1, f_2, f_3, \dots is the so called BBGKY chain. The Boltzmann equation is e.g. a model for f_1 .

12.4 Uncorrelated (collisionless) systems | multiplication closure to the BBGKY chain

Consider the simplest closure to the BBGKY hierarchy

$$f_2(\underline{x}, \underline{v}, \underline{x}', \underline{v}', t) = f_1(\underline{x}, \underline{v}, t) f_1(\underline{x}', \underline{v}', t) \quad (547)$$

i.e. we assume the particles to be uncorrelated, i.e. a particle at $\underline{x}', \underline{v}'$ does not effect one at $\underline{x}, \underline{v}$.

Note: This is akin to $P(A, B) = P(A) \cdot P(B)$ for independent events A, B .

Still particles are effected by the global effect of all others. For instance electrons in a plasma can be assumed uncorrelated.

12.4.1 General Continuity equation for probability in phase space

Let $\underline{w} = (\underline{x}_1, \dots, \underline{x}_N, \underline{v}_1, \dots, \underline{v}_N)$ be the phase space state, so $p = p(\underline{w})$. As the particles themselves, probability is conserved as captured in the continuity equation

$$\begin{aligned}\partial_t p + \nabla_{\underline{w}} \cdot (p \cdot \dot{\underline{w}}) &= 0 \\ \partial_t p + p \nabla_{\underline{w}} \cdot \dot{\underline{w}} + \dot{\underline{w}} \nabla_{\underline{w}} p &= 0\end{aligned}\tag{548}$$

so in terms of the particle velocities and positions (apply the chain rule)

$$\frac{\partial p}{\partial t} + \sum_i \left(p \frac{\partial \dot{\underline{x}}_i}{\partial \underline{x}_i} + \frac{\partial p}{\partial \underline{x}_i} \cdot \dot{\underline{x}}_i + p \frac{\partial \dot{\underline{v}}_i}{\partial \underline{v}_i} + \frac{\partial p}{\partial \underline{v}_i} \cdot \dot{\underline{v}}_i \right) = 0\tag{549}$$

12.4.2 Liouville equation for the general evolution of phase space probability

Based on the Hamiltonian equations²² one gets $\partial_{\underline{x}} \dot{\underline{x}} = -\partial_{\underline{v}} \dot{\underline{v}}$ and so

$$\frac{d\rho}{dt} = \partial_t p + \sum_i \left(\underline{v}_i \cdot \frac{\partial p}{\partial \underline{x}_i} + \underline{a}_i \cdot \frac{\partial p}{\partial \underline{v}_i} \right) = 0, \quad \underline{a}_i = \underline{v}_i = \frac{\underline{F}_i}{m_i} \text{ (Liouville's eqn)}\tag{550}$$

12.4.3 Vlasov equation for collisionless systems

In the collisionless / uncorrelated limit, one obtains the Vlasov equation for $f := f_1$ (multiply Liouville's equation by F and integrate)

$$\frac{\partial f}{\partial t} + \underline{v} \cdot \frac{\partial f}{\partial \underline{x}} + \underline{a} \cdot \frac{\partial f}{\partial \underline{v}} = 0\tag{551}$$

- the phase space density along trajectories is constant in the collisionless case. We can also understand the Vlasov equation based on the Boltzmann equation in eq. 131, where we assume the fluctuations in accelerations and change of phase space density to be decoupled.

²² $\dot{\underline{x}} = \partial_t \underline{x} = \partial_p H, \quad \dot{\underline{p}} = -\partial_{\underline{x}} H$

12.4.4 Accelerations in collisionless systems - including gravity into the Vlasov equation

Collective effects like self gravity can still be described in collisionless systems. Based on the density

$$\rho(\underline{x}, t) = m \int f(\underline{x}, \underline{v}, t) d\underline{v} \quad (552)$$

we can calculate the gravitational potential via Poisson's equation and thus an acceleration

$$\underline{\nabla}^2 \phi = 4\pi G \rho \rightarrow \underline{a} = -\underline{\nabla}_{\underline{x}} \phi \quad (553)$$

so the Vlasov equation

$$\frac{\partial f}{\partial t} + \underline{v} \cdot \frac{\partial f}{\partial \underline{x}} - \underline{\nabla}_{\underline{x}} \phi \cdot \frac{\partial f}{\partial \underline{v}} = 0 \quad (554)$$

As of the Vlasov equation we do not describe single particles anymore but rather directly model the phase space density. For discretization we later reintroduce particles - but not the physical ones rather macro-particles sampling the phase space in a Monte-Carlo fashion (Monte-Carlo comes later).

12.5 When is a gravitational system collisionless?

Idea: In every system collisions happen, but if the time of our simulation is shorter than the timescale on which collisions have a meaningful impact t_{relax} we can assume the system to be collisionless. **We thus view a system as collisionless if**

$$t_{\text{relax}} \gg t_{\text{of interest for the simulation}}, \quad \text{final result } t_{\text{relax}} = \frac{N}{8 \log N} t_{\text{cross}} \quad (555)$$

number of particles in the system N , typical time to cross the system t_{cross}

Where we will derive and explain t_{relax} in the following. We can already see the surprising result: **For two gravitational systems with the same mass and size, the one with more smaller particles has a longer relaxation time so acts *more collisionless* as a whole.**

$$\text{overall potential more averaged} \gg \text{more frequent encounters} \quad (556)$$

12.5.1 Relaxation time in a gravitational system

But what is the relaxation time t_{relax} ? Consider a system of size R with N particles of (average) mass m . Consider a particle moving at speed v through it. Assume we would know the mean squared perpendicular velocity $\langle(\Delta v_{\perp})^2\rangle$ a particle would accumulate as of collisions by crossing the whole system and the time t_{cross} to cross the system.

We can then reasonably define the relaxation time as

$$t_{\text{relax}} \equiv \frac{v^2}{\langle(\Delta v_{\perp})^2\rangle/t_{\text{cross}}} \quad (557)$$

so the time at which perturbations have added up to a perpendicular velocity of the order of the velocity the particles move with - where collisions can surely not be neglected anymore.

Our next steps therefore are

- find an expression for t_{cross}
- find an expression for $\langle(\Delta v_{\perp})^2\rangle$

12.5.2 Crossing time

Consider a system of size R with N particles. It takes the particles roughly

$$t_{\text{cross}} = \frac{R}{v} \sim \frac{R^{\frac{3}{2}}}{\sqrt{GM}} \quad (558)$$

to cross the system, where we used

$$v^2 \simeq \frac{GM}{R}, \quad M = Nm, (\text{avg.}) \text{ particle mass } m \quad (559)$$

for the typical speed v of a field star is roughly that of a particle in a circular orbit at the edge of the galaxy.

Note: t_{cross} depends on the total mass and size of the system but not the individual masses directly.

12.5.3 Change in perpendicular velocity when crossing the system

For the change in perpendicular velocity, we can approximate

$$\langle(\Delta v_{\perp})^2\rangle \approx v^2 \langle\theta^2\rangle$$

mean squared deflection angle over the whole system $\langle\theta^2\rangle$

typical velocity of a particle v

(560)

We will

- calculate the deflection in one interactions
- calculate $\langle\theta^2\rangle$ assuming small angle interactions
- justify that most of the total deflection stems from many small deflections, justifying using $\langle\theta^2\rangle$ under the assumption of small angle deflections

12.5.3.1 Size of one small angle deflections based on the impact parameter

Consider the deflection scenario sketched in figure 98.

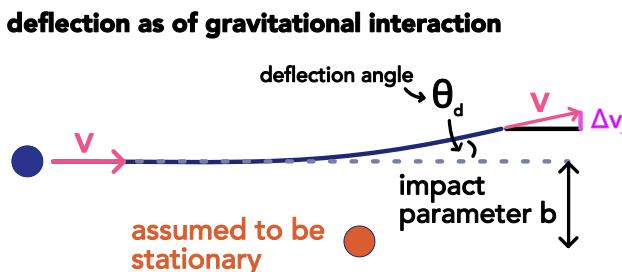


Figure 98: Gravitational deflection

In the small angle (here small deflection) approximation, we get

$$\theta_d \approx \sin \theta_d \approx \frac{\Delta v_{\perp}}{v}$$
(561)

where the change in perpendicular velocity as of the gravitational field of the other particle is calculated as if the particle would have moved in a straight line (small deflection, Born approximation)

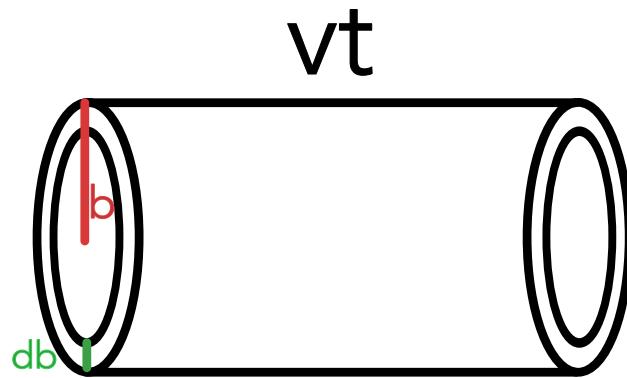


Figure 99: Impact cylinder

$$\begin{aligned} \Delta p_{\perp} = m \Delta v_{\perp} &= -m \underbrace{\int_{-\infty}^{\infty} \nabla_{\perp} \phi dt}_{\text{deflection as}} \approx -m \underbrace{\int_{-\infty}^{\infty} \partial_b \left(\frac{Gm}{\sqrt{b^2 + v^2 t^2}} \right) dt}_{\substack{\text{Born} \\ \text{approximation}}} \\ &= \dots = \frac{2Gm^2}{bv} \end{aligned} \quad (562)$$

so the deflection angle is

$$\theta_d \approx \frac{b_0}{b} \text{ with critical impact parameter } b_0 = \frac{2Gm}{v^2} \ll b \text{ as of our small angle approximation} \quad (563)$$

where we would have $b = b_0$ for $\Delta v_{\perp} = v$.

12.5.3.2 Mean squared deflection for many small deflections

Over many interaction we accumulate

$$\langle \theta^2 \rangle = \sum_{\text{all encounters}} \theta_d^2 = \sum_{\text{all encounters}} \left(\frac{b_0}{b} \right)^2, \quad \langle \theta \rangle = 0 \text{ as of symmetry} \quad (564)$$

and write

$$\langle \theta^2 \rangle = \int_{b_{\min}}^{b_{\max}} \theta_d^2 dN, \quad \text{number of encounters } dN = n vt 2\pi b db \text{ with } b \in [b, b + db] \quad (565)$$

(for dN see figure 99) so (plug in and integrate)

$$\langle \theta^2 \rangle = n 2\pi b_0^2 v t \ln \Lambda, \quad \text{Coulomb logarithm } \ln \Lambda = \ln \frac{b_{\max}}{b_{\min}} \quad (566)$$

12.5.3.3 Many small deflections are more important than few large ones

Note: In the following we will compare deflection frequencies. The frequency in the small angle case is the frequency with which small deflections add up to e.g. 1 or $\frac{\pi}{2}$, so that a higher frequency means that small deflections are overall more important (otherwise small deflections have a higher *frequency* without calculation).

Based on our previous result for **small deflections** we can calculate the time t until $\langle \theta^2 \rangle \approx 1$ (equivalent to one deflection with $b = b_0$), to get a deflection frequency

$$\nu_d \equiv \frac{1}{t_d} = n 2\pi b_0^2 v \ln \Lambda \quad (567)$$

Let us continue with the **mean deflection frequency based on large deflections**. Large deflections are those for $b \gtrsim b_0$. The mean deflection frequency can be calculated as

$$\nu_{d, \text{single}} \equiv \frac{1}{t_{d, \text{single}}} = \frac{v}{\lambda_{mfp}} = n \underbrace{\sigma}_{\approx \pi b_0^2} v \approx n \pi b_0^2 v \quad (568)$$

cross section σ , mean free path λ_{mfp} , number density n

so we can relate

$$\frac{\nu_d}{\nu_{d, \text{single}}} \ln \Lambda, \quad \text{typically } 20 \leq \ln \Lambda \leq 30 \quad (569)$$

so the deflection frequency based on lots of small deflections adding up is larger than the one based on single big scattering event which based on our definition of the deflection frequency means that small angle deflections are more important. This is also reflected in

$$\frac{\sigma_{\text{many small deflections adding to } \frac{\pi}{2}}}{\sigma_{\text{one } \frac{\pi}{2} \text{ deflection}}} = 8 \ln \Lambda \quad (570)$$

12.5.3.4 Finally the calculation of $\langle (\Delta v_{\perp})^2 \rangle$

We can now use the result for many small deflections

$$\langle \theta^2 \rangle = n 2\pi b_0^2 v t \ln \Lambda \quad (571)$$

in

$$\langle(\Delta v_{\perp})^2\rangle \approx v^2\langle\theta^2\rangle = v^2n2\pi b_0^2vt\ln\Lambda \underset{N=vt\pi R^2n}{=} = 2N\frac{v b_0^2}{R} \ln\Lambda \quad (572)$$

justified by the higher meaning of many small deflections.

12.5.3.5 How to choose b_{min} and b_{max} in the Coulomb logarithm?

We choose $b_{max} = R$ (the system size) and $b_{min} \approx b_0 = \frac{2Gm}{v^2}$ (the critical impact parameter) (as large deflections with $b \lesssim b_0$ are rare).

Using the typical velocity $v^2 \approx \frac{GM}{R}$ and $N = \frac{M}{m}$ we can write

$$\log\lambda \approx \log\frac{N}{2} \approx \log N \quad (573)$$

Which gets us to the initially stated result for the relaxation time.

12.5.4 Examples of astrophysical relaxation times

As a reference for determining if collisionless or not dependent on the relaxation time, we use the age of the universe $t_{age} = \frac{1}{H_0} \sim 10$ Gyr. So while here the globular star cluster is not collisionless, on a more usual timescale it is. The relaxation times are in table 17 - as previously noted systems of lots of small particles or more collisionless.

System	Number of bodies \sim	Crossing time \sim	Relaxation time $\approx \frac{N}{8\ln N}t_{cross} \sim$	Collisionless over age of universe $t_{relax} \gg t_{age}$
Globular star clusters	10^5	$0.5Myr$	$0.5Gyr$	No
Stars in typical galaxy	10^{11}	$\frac{1}{100H_0}$	$5 \cdot 10^6 t_{age}$	Yes
Dark matter in galaxy	10^{77}	$\frac{1}{10H_0}$	$10^{73} t_{age}$	Absolutely

Table 17: Relaxation times

12.6 N-body models of collisionless systems

Idea for modelling a collisionless system - N-body simulation: Use the standard gravitational equations of motion (not fluid equations) but for fiducial heavier macro particles than in the original system.

We introduce non-physical macro particles, to discretize the collisionless fluid described by the Poisson-Vlasov system. We use far fewer macro particles than particles in the real system - the macro particles are heavier (/ have more charge). The macro-particles follow the equations

$$\ddot{\underline{x}} = -\nabla \phi(\underline{x}_i), \quad \phi(\underline{x}) = -G \sum_{j=1}^N \frac{m_j}{\left[(\underline{x} - \underline{x}_j)^2 + \epsilon^2 \right]^{\frac{1}{2}}} \quad (574)$$

softening length ϵ

Validity-note: The real system we model is composed of much more particles than our N and is collisionless (remember $t_{\text{relax}} = \frac{N}{8 \log N} t_{\text{cross}}$). For our model to be valid, it also has to be collisionless, so with our lower N we still need to fulfill $t_{\text{relax}} \gg t_{\text{of interest}}$ for the simulation (and to have a sufficiently smooth gravitational potential (/ well described compared to the real *smooth-potential* system)).

Assume that our fiducial particles sufficiently well create the gravitational potential of the real system. Then a fiducial particle at any point, will have the same acceleration as a real particle there (the force is higher but also the inertial, cancelling to give the same acceleration) - so the fiducial particles will follow valid real-particle trajectories.

Problem: We only retrieve one (noisy) realization of the one-point function f_1 by one N -particle simulation

Idea: We can combine multiple simulation results for ensemble averages. The details of f_1 are critical. For instance, for multiple crossings of a shock front we would have a high energy tail in the velocity distribution.

12.6.1 The softening length ϵ

There is a softening length in the denominator that reduces the potential for distances very close to other particles. This is especially important if we choose large macro-particles - the

softening length gives a smallest impact parameter on a macro scale.

- This avoids large angle scattering (as strong potential interaction → strong deflection)
- **Avoid numerically expensive singularities:** Without softening, there would be singularities in the potentials, which would cause high numerical effort when integrating the orbits (as of the large numbers)
- **Avoid bound particle pairs:** When gravitational particles can come very close to each other they can form highly interactive / correlated pairs - and we want a collisionless system not highly correlated particles. Bounded pairs are avoided if

$$\langle v^2 \rangle \gg \frac{Gm}{\epsilon} \quad (575)$$

The softening length introduces a smallest resolved scale / smallest trustworthy scale. We must make a compromise between spatial resolution, computational cost and the points discussed above.

13 Force calculations | tree algorithms and particle mesh technique

We have discretized our N-body system using fiducial macro-particles. The equations of motion are given by

$$\ddot{\underline{x}} = -\nabla \phi(\underline{x}_i), \quad \phi(\underline{x}) = -G \sum_{j=1}^N \frac{m_j}{\left[(\underline{x} - \underline{x}_j)^2 + \epsilon^2 \right]^{\frac{1}{2}}} \quad (576)$$

Integration in time: Given the forces on the particles, we can integrate this ordinary differential equation in time. Classically a symplectic method like leapfrog would be used. But modern N-body-packages like Rebound use high-order adaptive-step-size integrators. See section 3.7.3 for details.

But how can we calculate the forces on the particles given the particle positions?

13.1 Calculating the forces | Direct summation

We can simply and exactly calculate

$$\ddot{\underline{x}}_i = -G \sum_{j=1}^N \frac{m_j}{\left[(\underline{x}_i - \underline{x}_j)^2 + \epsilon^2 \right]^{\frac{3}{2}}} (\underline{x}_i - \underline{x}_j) \quad (577)$$

Problem: For the force on each of the N particles, $N - 1$ terms have to be evaluated and summed - so in total we have $\mathcal{O}(N^2)$ operations. For t_{relax} to be large (so the simulated time can be large), N must be large. But even for $N = 10^{10}$ assuming we can to $N = 10^6$ in one month of computer time, we'd need to run our computer $\sim 10^7$ years.

13.2 Overview on faster, approximate force calculation schemes

- **tree algorithms** - distant groups do not need to be resolved in every detail to calculate their respective forces - some or even one term of the multipole expansion can be used. Generally we can construct hierarchical multipole methods where a tree is a method to partition space based on the particle distribution.
- **particle mesh methods** - from the particle distribution, a mass density on a grid is calculated and using

- **fourier transform based methods** to calculate the potential from the Poisson equation and the density distribution (the necessary convolution is a multiplication in Fourier space and FFT is fast ($\mathcal{O}(N \log N)$))
- **iterative solvers** relaxation methods to solve the Poisson equation

the forces on gridpoints can be calculated and interpolated to the particles.

We can often combine the different force calculation approaches and use direct summation on small scales. To speed things up we use GPUs that are very fast for easy parallel operations like (block) matrix multiplication.

13.3 Faster method I | Multipole expansion | tree algorithms

Idea: In tree algorithms the force on one particle is calculated based on multipole expansions of groups in the *far-field* rather than individual particles (see figure 100), while the *near-field* is calculated exactly.

Consider the simple example where we want to calculate the force (and with this the movement) of a single particle based on the interaction with a distant group of particles. If the group spans only a small angle of the particles view-field, the force acting on the particle can be approximated by the one along the axis to the groups center of mass, so

$$\underline{a}_{\text{our particle}} \approx -GM \frac{\underline{r}_{\text{our particle}} - \underline{s}}{\|\underline{r}_{\text{our particle}} - \underline{s}\|^3} \quad (578)$$

group's mass M , group's center of mass \underline{s}

Further details of the group can be resolved by higher order terms of the multipole expansion.

In the following we need to answer

- How do we resolve the potential of a group in less detail? (the simple monopole term in the force above will often be enough) When will this be accurate?
- How can derive a scheme, so that for the force on each particle, not all others have to be considered, but groups under small opening angle are only resolved as a whole (e.g. by the monopole term)?

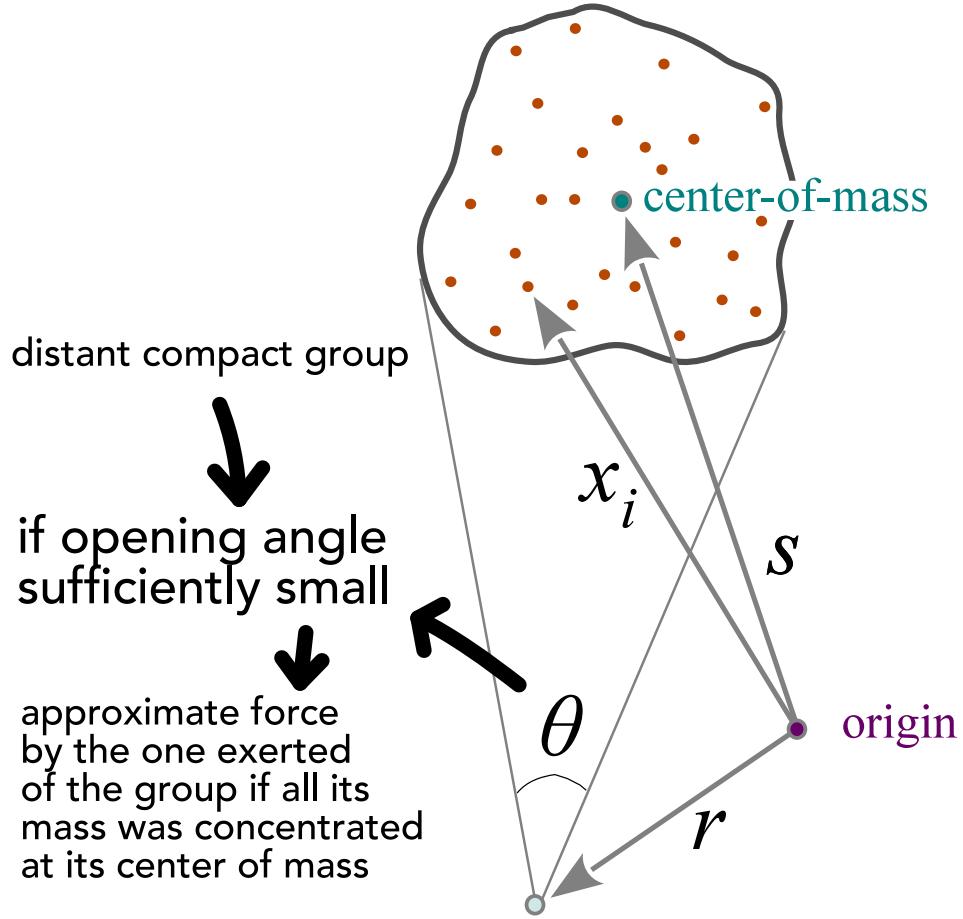


Figure 100: If the opening angle of a group is small enough, the force can be approximated by the monopole term at its center of mass.

The exact potential of the group measured at \underline{r} is

$$\phi(\underline{r}) = -G \sum_i \frac{m_i}{\|\underline{r} - \underline{x}_i\|} \quad (579)$$

13.3.1 Deriving the multipole expansion

Let us rewrite the potential of the group as

$$\phi(\underline{r}) = -G \sum_i \frac{m_i}{\|\underline{r} - \underline{s} + \underline{s} - \underline{x}_i\|}, \quad \text{group center of mass } \underline{s} = \frac{1}{M} \sum_i m_i \underline{x}_i \quad (580)$$

and define $\underline{y} := \underline{r} - \underline{s}$ and $\underline{l} = \underline{s} - \underline{x}_i$. We Taylor expand with respect to \underline{l} around $\underline{l} = 0$.

Multivariate Taylor expansion:

$$f(\underline{x} + \underline{h}) = f(\underline{x}) + \underline{h} \cdot \nabla f(\underline{x}) + \frac{1}{2} \underline{h}^T \underline{\underline{H}}_f \Big|_{\underline{x}} \underline{h} + \mathcal{O}(\underline{h}^3) \quad (581)$$

where $\underline{\underline{H}}_f$ is the Hessian matrix of f , the Jacobian of the gradient of f .

with this we get

$$f(\underline{y} + \underline{l}) = \frac{1}{\|\underline{y} + \underline{s} - \underline{x}_i\|} = \frac{1}{\|\underline{y}\|} + \frac{\underline{y} \cdot \underline{l}}{\|\underline{y}\|^3} + \frac{1}{2} \underbrace{\frac{\underline{l}^T [3\underline{y}\underline{y}^T - \underline{y}^2 \mathbf{1}] \underline{l}}{\|\underline{y}\|^5}}_{= \frac{\underline{y}^T [3\underline{l}\underline{l}^T - \underline{l}^2 \mathbf{1}] \underline{y}}{\|\underline{y}\|^5}} + \dots \quad (582)$$

The numerators summed over all particles in the group lead to the *moments*

$$\begin{aligned} \text{monopole: } M &= \sum_i m_i, & \text{dipole : } \underline{D} &= \sum_i m_i (\underline{s} - \underline{x}_i), \\ \text{quadrupole: } Q_{ij} &= \sum_k m_k \left[3 (\underline{s} - \underline{x}_i) (\underline{s} - \underline{x}_i)^T - (\underline{s} - \underline{x}_i)^2 \delta_{ij} \right] \end{aligned} \quad (583)$$

Note: The gravitational dipole moment (asymmetry of the mass distribution around the center of mass) vanishes as of the definition of the center of mass.

13.3.2 Multipole expansion

Up to the quadrupole term, the potential of the group can be written as

$$\phi(\underline{r}) = -G \left(\frac{M}{\|\underline{y}\|} + \frac{1}{2} \frac{\underline{y}^T \underline{Q} \underline{y}}{\|\underline{y}\|^5} \right), \quad \underline{y} = \underline{r} - \underline{s}, \quad \text{center of mass } \underline{s} \quad (584)$$

which we expect to be accurate if

$$\theta \underset{\text{small angle}}{\approx} \frac{\langle \|x_i - \underline{s}\| \rangle}{\|\underline{y}\|} \approx \frac{l}{y} \ll 1, \quad \text{radius } l \text{ of the group} \quad (585)$$

so in other words if the distance of our point of interest to the center of mass of the group $\underline{y} = \underline{r} - \underline{s}$ is much larger than the radius of the group l .

13.3.3 Hierarchical grouping | baseline for smart force calculations

Let us split the particles into hierarchical groups (so a tree structure with groups and subgroups ...), which will later allow us to resolve details in the force on a particle as

necessary. For all groups we calculate the multipole moments we want to consider (e.g. only the monopole, i.e. the total mass) and the center of mass.

13.3.3.1 Cosntructing the tree | Barnes and Hut oct-tree

The Barnes and Hut algorithm goes as

1. start with a cube containing all particles
2. subdivide the cube into 8 sub-cubes
3. if a cube contains more than one particle, to back to step 2 (recursion)
4. Empty sub-cubes are not stored

we therefore grow a tree with particles as leaves to purity.

In 2D we have a quad-tree (illustrated in figure 101), in 1D a binary tree.

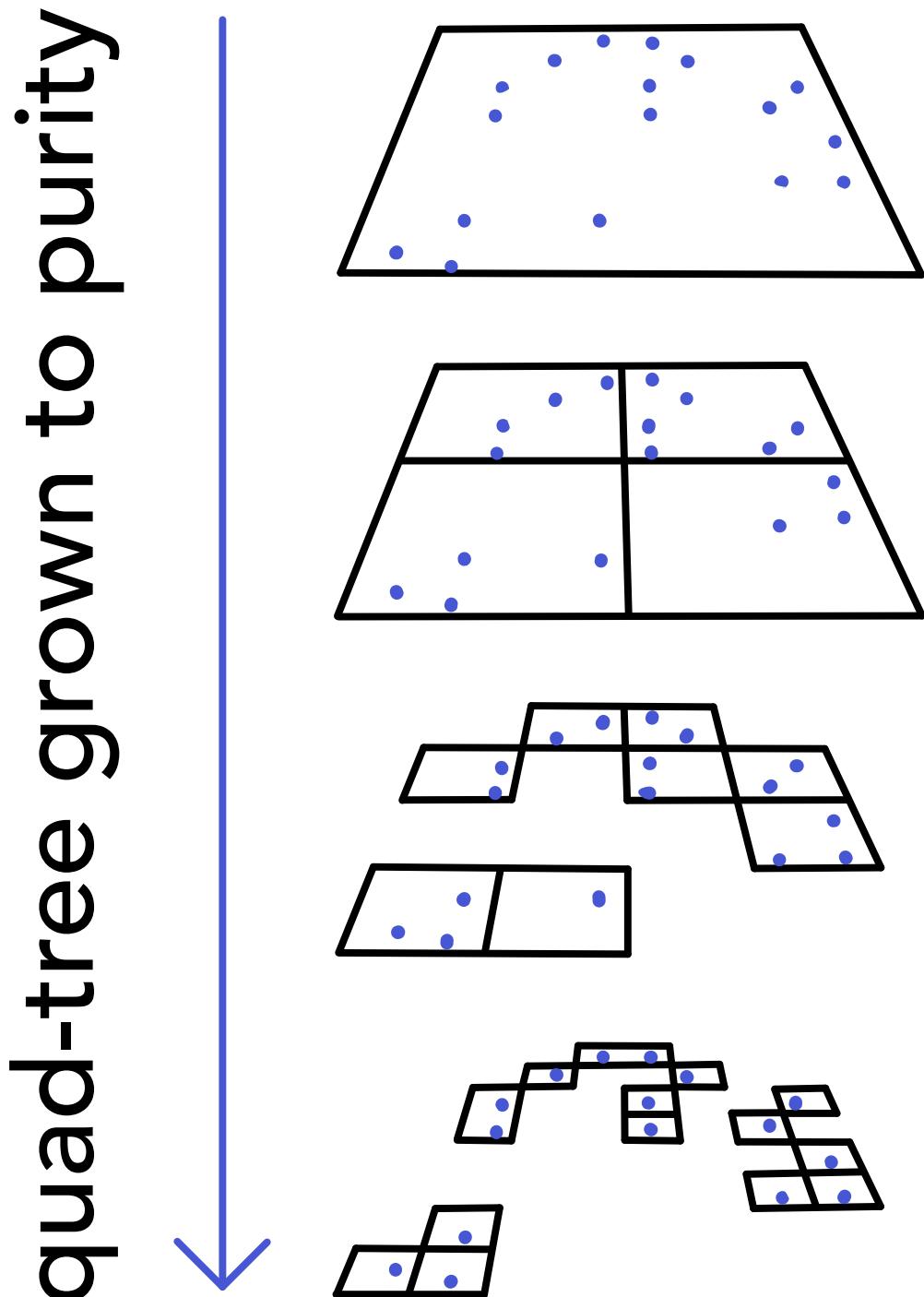


Figure 101: The Barnes and Hut algorithm constructs a tree with particles as leaves.

- **Advantage - auto-adaptation:** There is more refinement in dense areas and less in less dense areas.
- **Disadvantage - things can go deep:** The deepest level depends on the smallest separation of two particles - things go very deep if two particles are very close to each other and do not naturally fall onto some border.

Tree construction and calculation of the multipole moments

1. Build the tree by sequentially adding particles. Splits into sub-nodes are done, until the particle can be placed inside an empty sub-node.
2. Recursively compute the multipole moments
 - Does the node have sub-nodes?
 - **Yes:** Compute the multipole moments of the sub-nodes, then calculate the node's-moments based on them. For the monopole, we just have to sum the masses of the subnodes go get the monopole moment, the center of mass is the weighted average of the subnodes' centers of mass.
 - **No:** The moments should be trivial (monopole: just the particle mass, center of mass is the particle position).

Alternative grouping to the Barnes and Hut algorithm - kd-trees Alternatively, binary subdivisions along the axis can be done to do splits as illustrated in figure 102. The splitting can e.g. be done to balance fraction of mass. **Advantage:** The depth of the tree can easily be controlled; **disadvantage:** we have to store all the splitting planes - complicated bookkeeping.

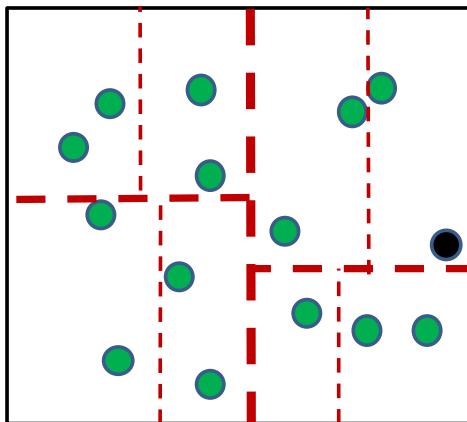


Figure 102: The kd-tree - binary subdivisions along the axes.

13.3.4 Tree walk - force calculation with adaptive group resolution

We want to calculate the force on each of our N particles, without considering all $N - 1$ other particles (in total $\mathcal{O}(N^2)$ operations).

Based on the hierarchical grouping and calculated multipole moments, we can reach $\mathcal{O}(N \log N)$ (derivation follows). The idea is to for each particle calculate an approximate force acting on it using the tree-walk algorithm.

13.3.4.1 Tree walk algorithm

For each of the N particles, to calculate the force we starting at the root of the tree check for the opening angles of the sub-groups, as illustrated in figure 103.

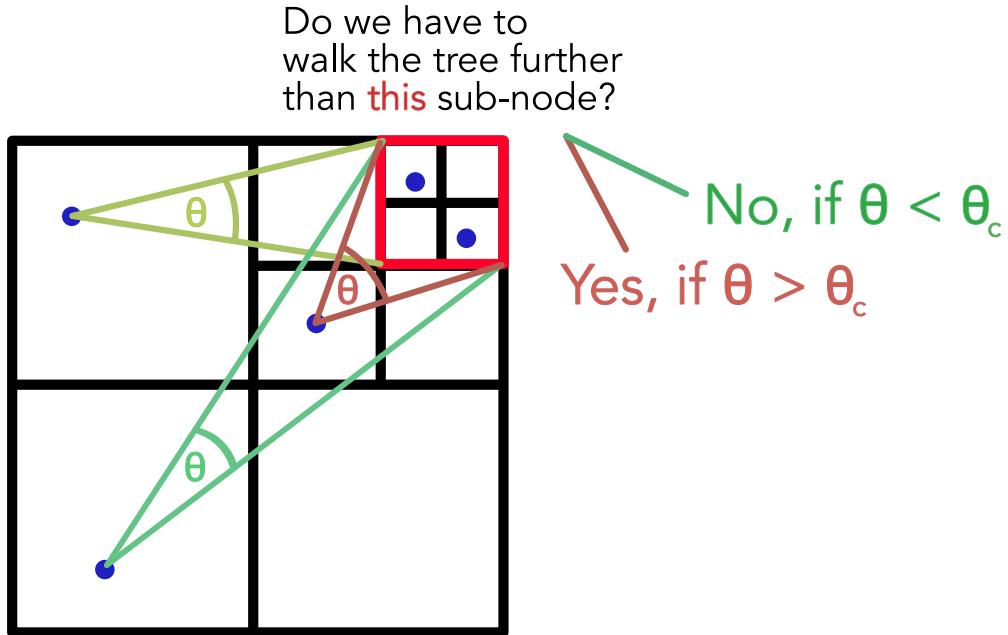


Figure 103: When do we walk further down the tree?

For each node consider the opening angle θ

- $\theta < \theta_c$ (smaller than critical angle): calculate the force from this node at our point of interest based on the multipole (or simply monopole) expansion. This is added to the total force at the point of interest.
- $\theta > \theta_c$ (larger than critical angle): open the subnodes of this node recurse

Error control by θ_c : The smaller θ_c the more expensive but also exact the force calculation. For $\theta_c = 0$ we are back at direct summation.

Fast multipole methods In our current method we calculate an approximate force for each of our N particles and then forward them based on this. In a further approximation, we can use, that the potential of a distant group field by nearby particles will be similar - this is cheaper but also harder to parallelize.

13.3.4.2 Derivation of the cost of the tree based force calculation | $\mathcal{O}(N \log N)$

Consider N homogeneously distributed particles in a sphere with radius R . Now consider a particle at the center of the sphere. At a distance r from it, the node-size which we have so far resolved in the force calculation can be estimated as

$$l^3(r), \quad l(r) \approx \theta_c r \quad (586)$$

So taking the mean particle distance

$$d = \left[\frac{\left(\frac{4\pi}{3}\right) R^3}{N} \right]^{1/3} \quad (587)$$

as the minimum distance at which other particles have to be considered, we can estimate

$$\begin{aligned} & \# \text{ nodes to be considered in the force calculation} \\ \text{for one particle} & \approx \int_d^R \frac{4\pi r^2 dr}{l^3(r)} = \frac{4\pi}{\theta_c^3} \ln \frac{R}{d} \propto \frac{\ln N}{\theta_c^3} \end{aligned} \quad (588)$$

As we have N particles with $\sim \log N$ interactions each, the total cost of the force calculation is $\mathcal{O}(N \log N)$, much better than the $\mathcal{O}(N^2)$ of direct summation.

13.3.4.3 Expected typical force errors in a monopole approximation | $(\Delta F_{\text{tot}}) \propto \theta_c^7$

We estimate the force error when resolving a group only based on the monopole (so as if all mass was concentrated at the center of mass) by the difference to the situation where all mass would be concentrated at the groups edge (group radius l).

$$\Delta F_{\text{node}} \sim |F_c - F_x| = \left| \frac{GM_{\text{node}}}{r^2} - \frac{GM_{\text{node}}}{r^2 + l^2} \right| \underset{\frac{1}{1+x} \approx 1-x \ll 1}{\approx} \frac{GM_{\text{node}}}{r^2} \frac{l^2}{r^2} = \frac{GM_{\text{node}}}{r^2} \theta_c^2 \quad (589)$$

plugging in $M_{\text{node}} = \frac{M}{N_{\text{nodes}}}$ with $N_{\text{nodes}} \propto \theta_c^3$ (as previously found), and summing over all nodes (akin to how variance adds up in the random walk), we get

total squared error in the force calculation for one particle based on monopole approximations:

$$\Delta F_{\text{tot}} \sim N_{\text{nodes}} \Delta F_{\text{node}} \propto \theta_c^7 \rightarrow |F_{\text{tot}}| \propto \theta_c^{3.5} \quad (590)$$

so roughly inversely proportional to the computational cost for this particle (as $N_{\text{nodes}} \propto$

θ_c^{-3}).

The smaller the opening angle, the higher the cost, the smaller the error.

13.4 Faster method II | Particle mesh technique for efficiently computing long-ranged forces

Consider for instance a plasma. Coulomb interactions are short-ranged, occurring on the scale of the Debye length (as of the shielding effect of charge) while gravity is long-ranged.

Problem: Direct summation is $\mathcal{O}(N^2)$ and all interactions are considered. However, for the short ranged interactions of a particle, only a few others in the vicinity are important and the long range interactions can safely be approximated.

Idea: Split the potential into one part describing short-ranged interactions and one part describing long-ranged interactions.

$$V = V^{\text{short}} + V^{\text{long}} \quad (591)$$

Calculate V^{short} using direct summation and V^{long} using the **particle mesh technique**.

The central idea of the particle mesh method is to use an auxiliary mesh on which the potential can be quickly calculated based on the methods discussed for the Poisson equation (multigrid relaxation or Fourier techniques).

13.4.1 Schematic particle mesh algorithm

1. Construct the density field ρ on the mesh based on the particle positions
2. Compute the potential on the mesh by solving the Poisson equation
3. Calculate the force field (on the mesh) from the discrete differentiation of the potential
4. Interpolate the forces onto the particles, move the particles appropriately and go back to step 1

We will now discuss each step.

13.4.2 Mass / charge assignment of particles to mesh cells

Setup Consider N particles $m_i, \underline{x}_i, i = 1, \dots, N$. We assume a cubical domain with side-length L and N_g grid cells per dimension so a uniform spacing $h = \frac{L}{N_g}$. The cell centers are

\underline{x}_p , integer index $\underline{p} = (p_x, p_y, p_z)^T, 0 \leq p_i < N_g$.

Question How should we assign the mass of the particles to the mesh cells? An intuition is given in figure 104.

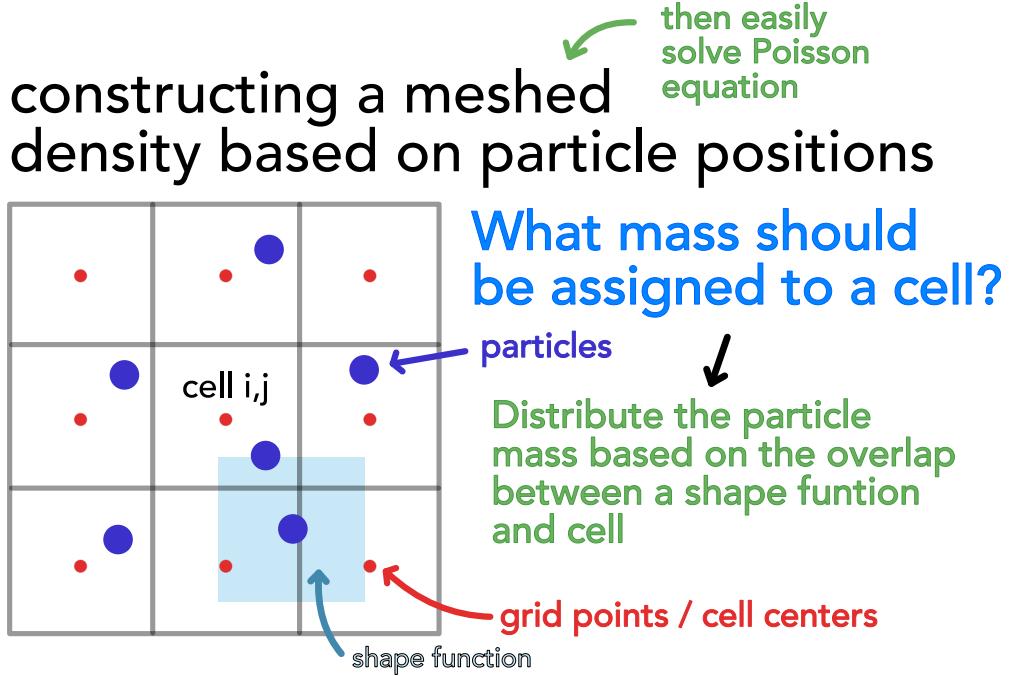


Figure 104: Mass assignment to the mesh cells.

Particle description Particles are not considered as point masses but described by normalized shape functions $S(\underline{x})$.

$$\int S(\underline{x}) d\underline{x} = 1 \quad (592)$$

Mass assignment Mass is assigned based on the overlap of a particles shape function with a mesh-cell, we assign mass of the particle to the mesh cell. The overlap over the shape function of particle i with cell with index \underline{p} is

$$W_{\underline{p}}(\underline{x}_i) = \int_{\text{cell } \underline{p} \text{ so } x_p - \frac{h}{2} \text{ to } x_p + \frac{h}{2} \text{ in all dims.}} S(\underline{x}_i - \underline{x}) d\underline{x} \quad (593)$$

Using

$$\Pi(\underline{x}) = \begin{cases} 1 & \text{if } \|\underline{x}\| < \frac{1}{2} \\ 0 & \text{else} \end{cases} \quad (594)$$

we can rewrite this overlap $W_{\underline{p}}(\underline{x}_i)$ as a convolution of Π and S .

$$W_{\underline{p}}(\underline{x}_i) = \int \Pi \left(\frac{\underline{x} - \underline{x}_p}{h} \right) S(\underline{x}_i - \underline{x}) d\underline{x} \quad (595)$$

Total density of the cell with index-vector \underline{p}

$$\rho_{\underline{p}} = \frac{1}{h^3} \sum_{i=1}^N m_i W_{\underline{p}}(\underline{x}_i) \quad (596)$$

But what shape function - so what assignment scheme - should we choose?

13.4.2.1 Assignment scheme I | Nearest grid point (NGP) assignment | δ -shape function

Using $\delta(\underline{x}_i - \underline{x})$ as the shape functions

$$W_{\underline{p}}(\underline{x}_i) = \Pi \left(\frac{\underline{x}_i - \underline{x}_p}{h} \right) \quad (597)$$

so we fully assign the mass of the particle to the nearest grid point, as illustrated in figure 105.

nearest grid point assignment

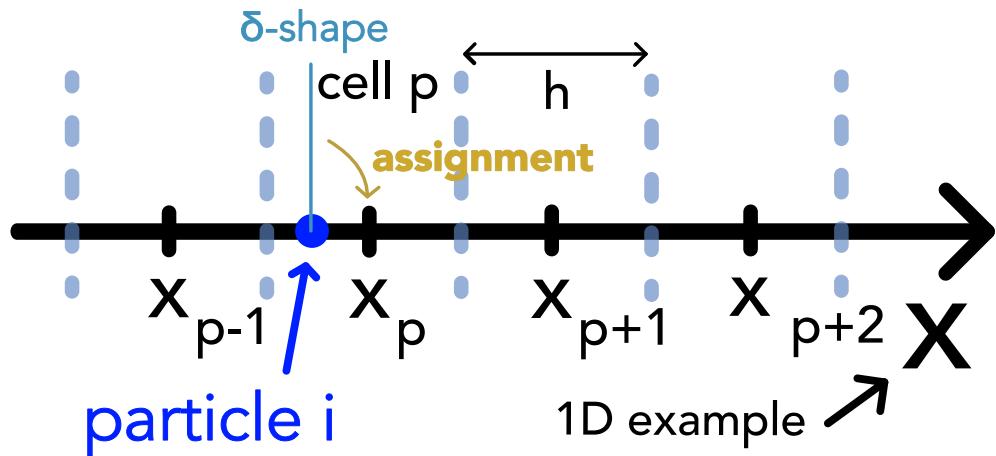


Figure 105: Nearest grid point assignment.

Problem: Mind that our particles are fiducial macro-particles making such an assignment (which would be reasonable if our particles would represent the physical ones) problematic.

13.4.2.2 Assignment scheme II | Cloud in cell (CIC) assignment | top-hat shape function

Here we use a cubical cloud shape (a top-hat)

$$S(\underline{x}) = \frac{1}{h^3} \Pi\left(\frac{\underline{x}}{h}\right), \quad \frac{1}{h^3} \text{ normalization in 3D} \quad (598)$$

of the same size as the mesh cells themselves (so at perfect overlap, all mass is assigned to the corresponding mesh cell and at max there is overlap with $2^3 = 8$ cells (in 3D)).

$$W_p(\underline{x}_i) = \int \Pi\left(\frac{\underline{x} - \underline{x}_p}{h}\right) \frac{1}{h^3} \Pi\left(\frac{\underline{x}_i - \underline{x}}{h}\right) d\underline{x} \quad (599)$$

The 1D case is illustrated in figure 106 (the 2D case in figure 104)

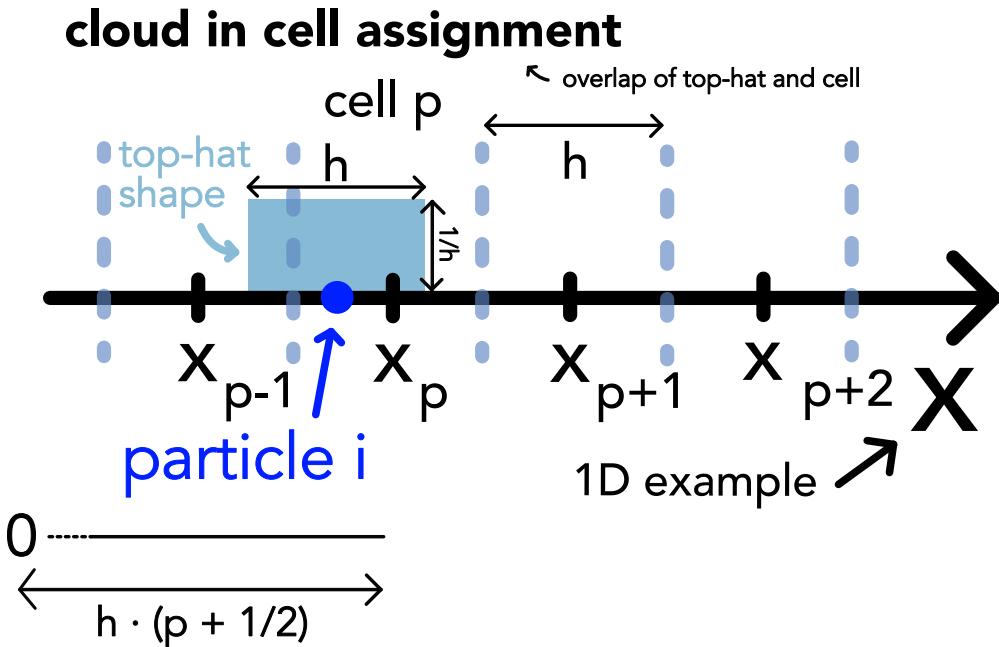


Figure 106: Cloud in cell assignment.

How to calculate the overlap practically? Consider we assign a floating point index consistent to how it is assigned to the cells, $x_i = (p_i + \frac{1}{2})h$, so $p_i = \frac{x_i}{h} - \frac{1}{2}$. The floored $\lfloor p_i \rfloor$ is the index of the cell closest to the left of that particle. The overlap with this cell is

$$W_{\lfloor p_i \rfloor}(\underline{x}_i) = 1 - (p_i - \lfloor p_i \rfloor) = 1 - p^*, \quad W_{\lfloor p_i \rfloor + 1}(\underline{x}_i) = p^* \quad (600)$$

(a short plausibility check is that for $p_i = \lfloor p_i \rfloor$ we have full overlap). Higher dimensions follow the same logic (but with more splits).

13.4.2.3 Assignment scheme III | Triangular shaped cloud (TSC) assignment | triangular shape function

We now use a triangular shape with a maximal overlap of 3^d cells (in d dimensions). A 1D illustration is given in figure 107.

triangular shaped cloud assignment

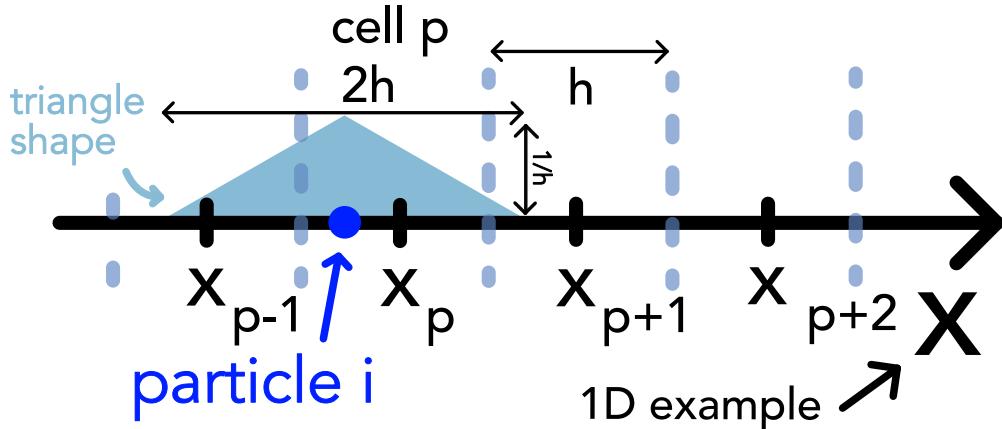


Figure 107: Triangular shaped cloud assignment.

The triangular shape can be obtained by the convolution of two top-hats, so we get

$$\begin{aligned} W_p(x_i) &= \int \Pi\left(\frac{x - x_p}{h}\right) \frac{1}{h^3} \Pi\left(\frac{x_i - x - x'}{h}\right) \frac{1}{h^3} \Pi\left(\frac{x'}{h}\right) dx dx' \\ &= \frac{1}{h^6} \int \Pi\left(\frac{x - x_p}{h}\right) \Pi\left(\frac{x_i - x}{h}\right) \Pi\left(\frac{x' - x}{h}\right) dx dx' \end{aligned} \quad (601)$$

13.4.2.4 Comparing the assignment schemes in terms of continuity

Let us consider the continuity of the force field.

- **Nearest grid point (NGP) assignment:** Density and hence force jump discontinuously when a particle crosses the cell boundary - at best piecewise constant force law
- **Cloud-in-cell (CIC) assignment:** Produces piecewise linear and continuous force but has jumps on the derivative of the force. As of the overlap information on where the particle is in the cell is stored
- **Triangular shaped cloud (TSC) assignment:** Here, the first derivative of the force is continuous

Name	Cloud shape $S(x)$	# of cells used	assignment function shape
NGP	$\delta(x)$	1^d	Π
CIC	$\frac{1}{h^d} \Pi\left(\frac{x}{h}\right)$	2^d	$\Pi \star \Pi$
TSC	$\frac{1}{h^d} \Pi\left(\frac{x}{h}\right) \star \frac{1}{h^d} \Pi\left(\frac{x}{h}\right)$	3^d	$\Pi \star \Pi \star \Pi$

Table 18: Overview over the assignment schemes.

The higher the order of the assignment scheme, the smoother one can make the force but the higher the computational cost (as the mass of a particle has to be distributed over more cells; more overhead in parallelization).

An overview over the assignment schemes is given in table 18.

13.4.3 Solving the Poisson equation for the potential on the mesh based on the meshed density

To solve the Poisson equation

$$\underline{\nabla^2} \phi = 4\pi G \rho \quad (602)$$

we have already discussed two methods

- Iterative solvers (relaxation methods): The solution is found based on an iterative relaxation in real space (the discretized Poisson equation is a linear system), e.g. by Jacobi or Gauss-Seidel iteration with possible speed-up by multigrid methods (making use of faster information travel / faster reduction of longwave errors on coarser grids and refinement on finer grids).
- Fourier transform based methods: The analytical solution to the potential is the convolution of the density with the Green's function of the Poisson equation. Based on the convolution theorem the convolution turns into a multiplication in Fourier space, which combined with the fast Fourier transform (FFT) is very fast ($\mathcal{O}(N \log N)$).

Assume now the potential is calculated on the mesh. **How do we calculate the force on the mesh?**

13.4.4 Calculating the force field on the mesh

Generally, the acceleration of a particle follows from the potential by

$$\underline{a} = -\underline{\nabla} \phi \quad (603)$$

which on the mesh can be done by **finite differencing**, e.g. the central difference

$$a_x(i, j, k) = -\frac{\phi(i+1, j, k) - \phi(i-1, j, k)}{2h} + \mathcal{O}(h^2), \quad \text{cell index } \underline{p} = (i, j, k) \quad (604)$$

Using a larger stencil, higher order schemes can be constructed (based on Taylor expansion from which also the truncation error can be estimated), e.g. by the 4-point stencil

$$a_x(i, j, k) = -\frac{1}{2h} \left\{ \frac{4}{3}[\Phi(i+1, j, k) - \Phi(i-1, j, k)] - \frac{1}{6}[\Phi(i+2, j, k) - \Phi(i-2, j, k)] \right\} + \mathcal{O}(h^4) \quad (605)$$

a_y and a_z follow analogously.

13.4.4.1 On the choice of the order of the finite difference scheme for the force calculation

The higher the order, **the higher the accuracy**, **the higher the computational cost**.

Note: In many collisionless systems, other errors inherent to the simulation scheme will be the bottleneck of accuracy so that the second order scheme is sufficient.

13.4.5 Interpolating the force from the mesh to the particles

We now have the forces on the mesh point but we want to know the forces on the particles.

13.4.5.1 We assign forces to the particles' positions using the same assignment kernel used to assign mass of the particles to the grid points

Idea: The mesh-nodes *give back* acceleration to a particle by the same ratio they have received mass from the particle.

Remember the mass assignment from particles to mesh cells

$$\rho_{\underline{p}} = \frac{1}{h^3} \sum_{i=1}^N m_i W_p(\underline{x}_i) = \frac{1}{h^3} \sum_{i=1}^N m_i W(\underline{x}_i - \underline{x}_{\underline{p}}), \quad W_p(\underline{x}_i) = \int \Pi \left(\frac{\underline{x} - \underline{x}_{\underline{p}}}{h} \right) S(\underline{x}_i - \underline{x}) d\underline{x} \quad (606)$$

The **force interpolation** on a mass m at coordinate \underline{x} is based on the acceleration field $\{a_{\underline{p}}\}$ on the mesh, so

$$\underline{F}(\underline{x}) = m \sum_{\underline{p}} \underline{a}_{\underline{p}} W(\underline{x} - \underline{x}_{\underline{p}}) \quad (607)$$

The assignment Kernel to interpolate the forces from the mesh to the particles must be the same that was used to assign mass from the particles to the mesh cells to

- have a vanishing self-force (a particle alone on the mesh, should not start moving by ghost forces)
- have force-asymmetry (pair-wise antisymmetric forces between particle pairs, Newton's third law)

Proofs of these properties follow.

13.4.5.2 Proof that for the same assignment kernel in the density and force assignment there is no self-force occurring

What is a self-force?: A self-force is a force, the particle would feel even if it was alone in the system - the particle would accelerate by itself violating conservation of momentum.

Consider the case of only one particle in the system. The force on this particle at \underline{x}_i is

$$\underline{F}(\underline{x}_i) = m_i \sum_{\underline{p}} \underline{a}_{\underline{p}} W_f(\underline{x}_i - \underline{x}_{\underline{p}}), \quad \text{density assignment for } N = 1 : \rho_{\underline{p}} = m_i W_d(\underline{x}_i - \underline{x}_{\underline{p}}) \quad (608)$$

where W_f denotes a general force assignment kernel. We will see that for $W_f = W_d$ there is no self-force as of a symmetry argument.

Deriving an exact expression for $\underline{a}_{\underline{p}}$ on the grid Here we do not use the finite difference but an exact expression via the Green's function of the Poisson equation. The Greens function for the Laplace equation

$$\underline{\nabla}^2 \phi = 4\pi G \rho \quad (609)$$

fulfills

$$\underline{\nabla}^2 G(\underline{x} - \underline{x}_{\underline{p}}) = 4\pi \delta(\underline{x} - \underline{x}_{\underline{p}}) \quad (610)$$

Using this we can write the density as a combination of lots of point masses

$$\rho(\underline{x}) = \int_{-\infty}^{+\infty} \rho\left(\frac{\underline{x}}{\underline{p}}\right) \delta\left(\underline{x} - \underline{x}_{\underline{p}}\right) d\underline{x}_{\underline{p}} \quad (611)$$

we can as previously find that the potential is the convolution of the density with the Green's function

$$\begin{aligned} \underline{\nabla}^2 \phi &= 4\pi G \int_{-\infty}^{+\infty} \rho\left(\frac{\underline{x}}{\underline{p}'}\right) \delta\left(\underline{x} - \underline{x}_{\underline{p}'}\right) d\underline{x}_{\underline{p}'} \\ &= \int_{-\infty}^{+\infty} \rho\left(\frac{\underline{x}}{\underline{p}'}\right) \underline{\nabla}^2 g\left(\underline{x} - \underline{x}_{\underline{p}'}\right) d\underline{x}_{\underline{p}'} = \underline{\nabla}^2 \int_{-\infty}^{+\infty} \rho\left(\frac{\underline{x}}{\underline{p}'}\right) g\left(\underline{x} - \underline{x}_{\underline{p}'}\right) d\underline{x}_{\underline{p}'} \\ &\xrightarrow{\text{check boundary conditions}} \phi = \int_{-\infty}^{+\infty} \rho\left(\frac{\underline{x}}{\underline{p}'}\right) g\left(\underline{x} - \underline{x}_{\underline{p}'}\right) d\underline{x}_{\underline{p}'} \end{aligned} \quad (612)$$

Discretizing to our grid (sum over all grid points) we get

$$\boxed{a_{\underline{p}} = -\underline{\nabla} \phi_{\underline{p}} = \sum_{\underline{p}'} d(\underline{p}, \underline{p}') h^3 \rho_{\underline{p}'}, \quad \text{mass in mesh cell: } h^3 \rho_{\underline{p}'}} \quad (613)$$

$$\text{derivative of the Green's function: } d(\underline{p}, \underline{p}') = -4\pi G \underline{\nabla} g\left(\underline{x}_{\underline{p}} - \underline{x}_{\underline{p}'}\right)$$

where the derivative of the Greens function is antisymmetric $d(\underline{p}, \underline{p}') = -d(\underline{p}', \underline{p})$ so that this exact expression for the force is antisymmetric.

A symmetry argument for the absence of self-force Let us plug $a_{\underline{p}}$ into the force expression

$$\begin{aligned} F_{\text{self}}(\underline{x}_i) &= m_i \sum_{\underline{p}} W_f\left(\underline{x}_i - \underline{x}_{\underline{p}}\right) a_{\underline{p}} \\ &= m_i^2 \sum_{\substack{\underline{p}, \underline{p}' \\ d(\underline{p}, \underline{p}') = -d(\underline{p}', \underline{p})}} \underbrace{d(\underline{p}, \underline{p}')}_{d(\underline{p}, \underline{p}') = -d(\underline{p}', \underline{p})} \underbrace{W_f\left(\underline{x}_i - \underline{x}_{\underline{p}}\right) W_d\left(\underline{x}_i - \underline{x}_{\underline{p}'}\right)}_{:= \mathcal{W}_{\underline{x}_i}(\underline{x}_{\underline{p}}, \underline{x}_{\underline{p}'}) \text{ with}} \\ &\quad \mathcal{W}_{\underline{x}_i}(\underline{x}_{\underline{p}}, \underline{x}_{\underline{p}'}) = \mathcal{W}_{\underline{x}_i}(\underline{x}_{\underline{p}'}, \underline{x}_{\underline{p}}) \text{ for } W_f = W_d \\ &= 0 \text{ for } W_f = W_d \end{aligned} \quad (614)$$

as the sum over this in total antisymmetric expression vanishes.

13.4.5.3 Proof that for the same assignment kernel in the density and force assignment, the forces between particle pairs are pair-wise antisymmetric

As of Newton's third law, two particles should exert opposite but equal in magnitude forces on each other otherwise conservation of momentum would be violated.

Consider a system of two particles 1 and 2. As of the vanishing self-force the force exerted on particle 1 only follows from the mass brought onto the mesh by particle 2 and vice versa.

For the force on particle 1 we have

$$\begin{aligned}\underline{F}_{12} &= m_1 \underline{a}(\underline{x}_1) \\ &= m_1 \sum_{\underline{p}} \underline{a}_{\underline{p}} W_f \left(\underline{x}_i - \underline{x}_{\underline{p}} \right) \\ &= m_1 m_2 \sum_{\underline{p}, \underline{p}'} \underline{d}(\underline{p}, \underline{p}') W_f \left(\underline{x}_1 - \underline{x}_{\underline{p}} \right) W_d \left(\underline{x}_2 - \underline{x}_{\underline{p}'} \right)\end{aligned}\tag{615}$$

So for $W_f = W_d$ we can see that swapping the indices of the particles as well as \underline{p} and \underline{p}' and using the antisymmetry of the derivative of the Green's function yields

$$\underline{F}_{12} = -\underline{F}_{21} \quad \rightarrow \quad \underline{F}_{12} + \underline{F}_{21} = 0\tag{616}$$

13.5 Outlook - combining the particle mesh and tree method

The particle mesh method has the elegant property of providing periodic boundary conditions (when the Poisson equation is solved in Fourier space) and natural force softening ensuring collisionless behavior by default.

However the natural force softening on the grid-scale means that the force resolution is rather poor.

The tree method on the other hand has high force resolution but does not provide periodic boundary conditions.

To carry out large-scale N-body simulations to study the formation of large scale structures in the universe, one might want to combine the particle mesh and (oct)-tree method (Bagla, 2002),

The main idea is to split the gravitational potential in Fourier space

$$\begin{aligned}\varphi_k &= -\frac{4\pi G \varrho_k}{k^2} \\ &= \underbrace{-\frac{4\pi G \varrho_k}{k^2} \exp(-k^2 r_s^2)}_{\text{long range part } \phi_k^l} - \underbrace{\frac{4\pi G \varrho_k}{k^2} (1 - \exp(-k^2 r_s^2))}_{\text{short range part } \phi_k^s}\end{aligned}\quad (617)$$

where the longrange potential (and from this force on our particles) is calculated in Fourier space as in the Particle Mesh method and the short force is calculated in real space as

$$\underline{f}^s(r) = -\frac{Gmr}{r^3} \left(\operatorname{erfc}\left(\frac{r}{2r_s}\right) + \frac{r}{r_s\sqrt{\pi}} \exp\left(-\frac{r^2}{4r_s^2}\right) \right) \quad (618)$$

14 Random Number Generation and Monte Carlo Techniques

In the following we will discuss

- How can (pseudo-) random numbers (following a given distribution) be generated?
- How can those random numbers be useful in estimation (e.g. of integrals)? - Monte Carlo techniques
- How can we sample from very complicated distributions based on a stochastic process and how can this be useful in calculating partition functions and parameter estimation?
- Monte Carlo Markov Chain

14.1 Random Number Generation and Sampling

14.1.1 An intuitive introduction to sampling

Probability distributions are ubiquitous. Consider the air molecules around you - what we feel as a temperature stems from the microscopic movement of those molecules, some moving slower, some quicker, some in one, some in another direction. Actually (under ideal conditions), if I was to measure those velocities of many molecules only along one axis and do a normalized histogram of my measurements, I would see a Gaussian emerge.

Now we want to do the opposite:

Sampling: Given a probability distribution $f(x)$, we want to generate measurements x_i such that in the limit of lots of measurements their normalized histogram resembles the probability distribution $f(x)$.

For instance to simulate my previous velocity measurement. Sampling has numerous applications, e.g. in smartly approximating integrals (Monte Carlo integration).

14.1.2 Random Number Generators - Base of all sampling methods: Sampling from the uniform distribution is *easy*

The standard continuous uniform distribution is given by the probability density function (PDF)

$$f(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases} \quad (619)$$

We would now like to generate pseudo-random numbers $U \sim \mathcal{U}(0, 1)$. To do so, **pseudo-random number generators** usually deterministically generate a sequence of integers which is then converted to a float in $[0, 1]$.

14.1.2.1 Advantages of Pseudo-Random Number Generators over True Random Number Generators

While we can also use true random numbers (e.g. based on radioactive decay or lava lamps), pseudo-random number generators have the advantages

- they are deterministic, so the number sequence is repeatable → reproducibility, debugging
- they are usually faster
- the distribution quality is possibly better
- the distribution is not dependent on environmental factors

14.1.2.2 Desirable properties of good pseudo-random number generators

- **Repeatability:** Same seed → same sequence
- **Randomness:** the random numbers should
 - be uniformly and homogeneously distributed in $[0, 1]$
 - be independent of each other (no correlation, not fully possible)
- **Efficiency:** Fast and memory efficient generation
- **Portability:** Same results across different machines
- **Long period:** The sequence of random numbers should at least not repeat for sufficiently long
- **Insensitivity to seed:** The seed should not change the characteristics of the random number distribution

14.1.2.3 A simple class of pseudo-random number generators: Linear Congruential Generators

Linear congruential generators generally follow the form

$$\begin{aligned} v_{n+1} &= (av_n + b) \mod m \in [0, m - 1], \quad \text{seed } v_0 \in \mathbb{N} \\ \text{pseudo-random number } u_n &= \frac{v_n}{m} \in [0, 1) \quad \text{parameters } a, b, m \in \mathbb{N} \end{aligned} \tag{620}$$

Note: As of the modulo operation, the period is at most m , where the cycle length depends on the parameters and seed, see figure 108.

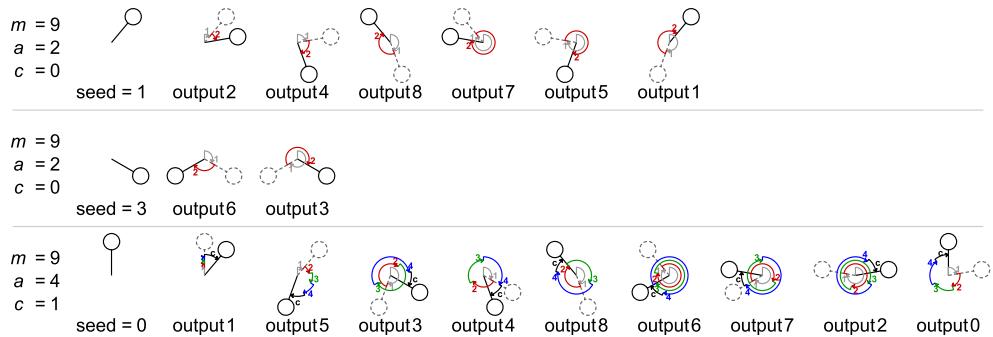


Figure 108: The cycle length of a linear congruential generator depends on the parameters and seed.

Examples of LCGs are ANSI-C, RAND, drand48 and NAG (NAG with period of 2^{48}).

While LCGs are fast and require minimum storage, but they show regular patterns. For instance if you generate lots of random numbers and from them take bunches of k and plot them in a k -dimensional space, you will see they lie on at most $(k! \cdot m)^{1/k}$ parallel $k - 1$ -dimensional hyperplanes. Also, if m is chosen as a power of 2 (as in the subsequently discussed RANDU), the least significant bits are not very random (the least significant one has a period of at most 2).

Let us present RANDU, an infamous, simple linear congruential generator (LCG), given by

$$v_{n+1} = (65539 \cdot v_n) \bmod 2^{31}, \quad \text{seed } v_0, \quad \text{pseudo-random number } u_n = \frac{v_n}{2^{31}} \quad (621)$$

where based on an integer sequence uniformly distributed numbers u_n are generated. Now RANDU is not infamous because it is especially good, but rather because it is especially bad in the sense that generated numbers are closely related, as illustrated in figure 109.

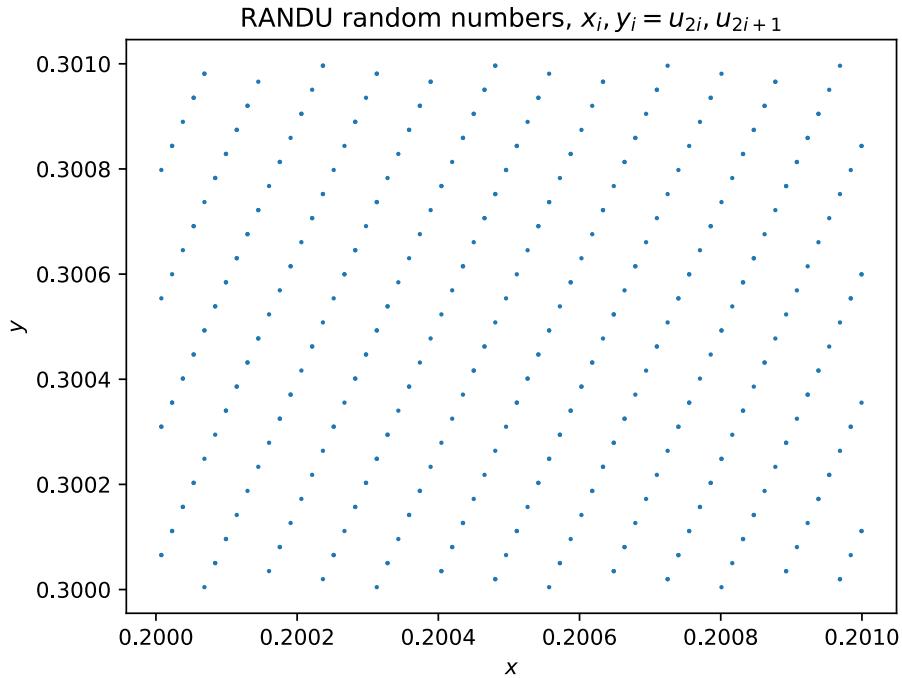


Figure 109: Consecutive numbers generated by RANDU plotted in 2D lie on parallel lines.

A more in-depth discussion about the problems of RANDU and linear congruential generators in general can be found in Press et al., 2007, chapter 7, where more advanced methods are also discussed. In R for instance the default number generator is the Mersenne-Twister (Matsumoto and Nishimura, 1998) according to the Comprehensive R Archive Network.

14.1.2.4 A first improvement | Combining multiple LCGs

$$\left. \begin{array}{l} X_{i+1} = (40014X_i) \bmod 2147483563 \\ Y_{i+1} = (40692Y_i) \bmod 2147483399 \\ Z_{i+1} = (X_i + Y_i) \bmod 2147483563 \end{array} \right\} \rightarrow \text{then map } Z_i \text{ to floating point number} \quad (622)$$

14.1.2.5 Lagged Fibonacci Generators

Inspired by the Fibonacci series, we combine *lagged numbers*, i.e. numbers earlier in the series by offsets p and q

$$v_i = (v_{i-p} \odot v_{i-q}) \bmod m \quad (623)$$

where \odot is some arithmetic operation like addition or bitwise XOR. For instance the Mersenne Twister is based on such a method and has period $2^{19937} - 1$.

14.1.2.6 Roughly evenly spaced sampling - blue noise

Randomly sampled points in 2D will not be evenly spread - if we want more even sampling, *blue noise* generated for instance by Fast Poisson Disk sampling (sampled points are at least r apart, $\mathcal{O}(N)$, N sampled points) can be used.

Blue noise and usual randomly sampled numbers are shown in table 19. For instance the photoreceptors on our retina are laid out in a blue noise fashion.

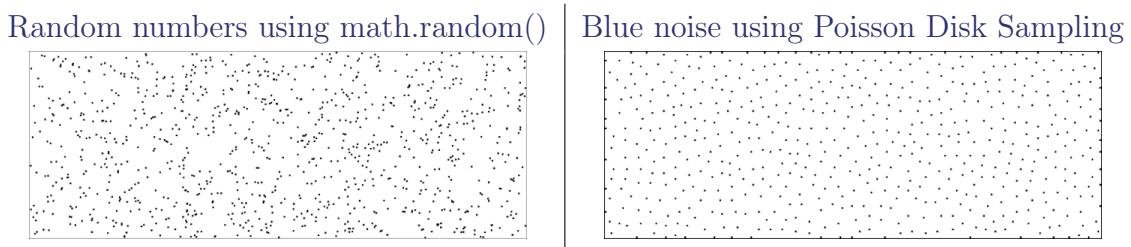


Table 19: Randomly sampled points in 2D (left) and blue noise (right).

14.1.3 Inverse Transform Method | sampling from distributions with algebraically invertible cumulative distribution functions (CDFs)

Consider we want to sample from a distribution with PDF $f(x)$ and cumulative distribution function (CDF) $F(x) = \int_{-\infty}^x f(x')dx'$. Given a uniformly distributed variable $U \sim \mathcal{U}(0, 1)$, $F^{-1}(U)$ is distributed according to f , as

$$P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x), \quad x \in \mathbb{R} \quad (624)$$

where we used that as of $f(x) \geq 0$, $F(x)$ is monotonically increasing, so its application keeps the inequality intact.

This is the foundation of the inverse transform method which is implemented in R in code-snippet 5 and illustrated at the hand of sampling from a Gaussian in figure 110. Note that the inverse transform method would actually not be used on a Gaussian as the inverse of its CDF has no closed-form expression and e.g. the Box-Muller method (Box and Muller, 1958) would be used.

Problem: Not all CDFs are algebraically invertible.

Idea: Numerical inversion is just swapping the axes - but we have discrete inverted point with different spacing. One idea is to linearly interpolate in the to find F^{-1} at the uniformly distributed points.

```

1  inverse_transform_method <- function(n, F_inv) {
2      # Draw n samples from a distribution with CDF F
3      # given its inverse F_inv.
4      u <- runif(n) # draw n samples from U(0,1)
5      return(F_inv(u)) # apply the inverse transform
6  }

```

Code-Snippet 5: Inverse Transform Method in R

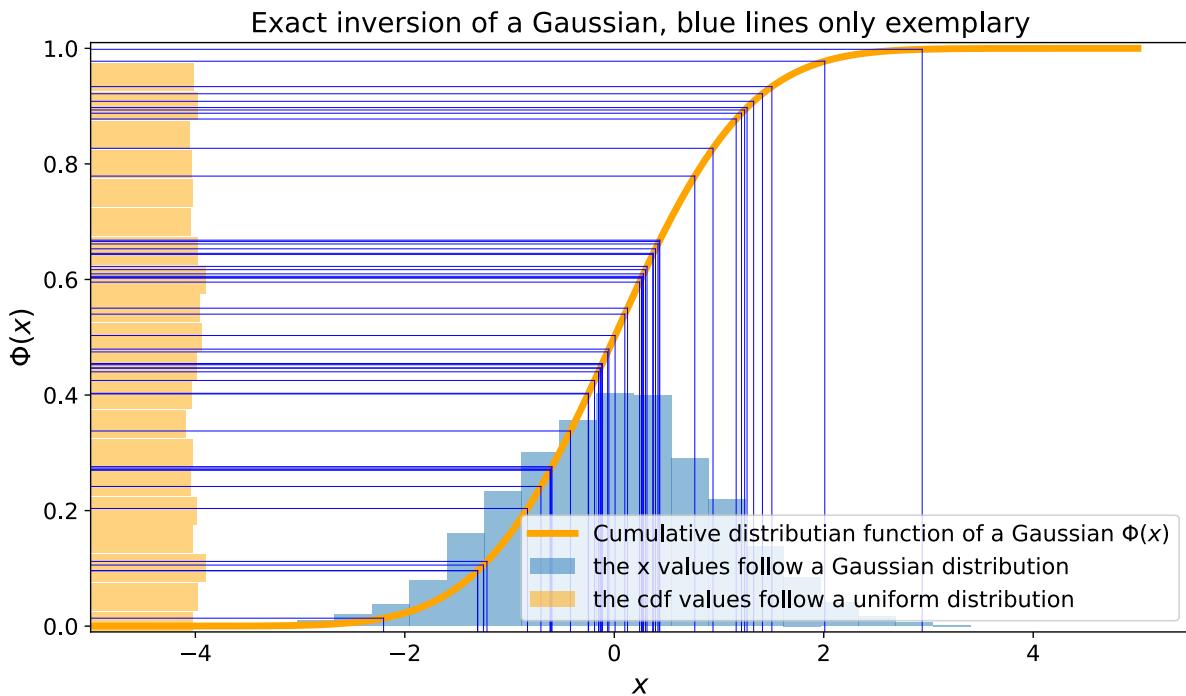


Figure 110: Illustration of the Inverse Transform Method

14.1.3.1 Alternative derivation from the transformation between probability distributions*

Consider probability distributions $f_1(x)$, $f_2(y)$ with $y = y(x)$. Conservation of total probability implies

$$f_1(x)dx = f_2(y)dy \quad \rightarrow \quad f_2(y) = f_1(x) \left| \frac{dx}{dy} \right| \quad (625)$$

so the cumulative distribution functions are equal (?)

$$F_1(x) = \int_{-\infty}^x F_1(\tilde{x})d\tilde{x} = \int_{-\infty}^y f_2(\tilde{y})d\tilde{y} = F_2(y(x)) \quad (626)$$

so taking $f_1(x)$ as the uniform distribution, $F_1(x) = x$ and we get to the same result.

14.1.3.2 Example: Inverse Transform Method applied to the standard Laplace distribution*

The standard Laplace distribution is given by the PDF

$$f(x) = \frac{1}{2}e^{-|x|}, \quad x \in \mathbb{R} \quad (627)$$

with the CDF following from $F(x) = \int_{-\infty}^x f(x')dx'$ to

$$F(x) = \begin{cases} \frac{1}{2}e^x & \text{if } x \leq 0 \\ 1 - \frac{1}{2}e^{-x} & \text{if } x > 0 \end{cases} \quad (628)$$

with the inverse CDF

$$F^{-1}(u) = \begin{cases} \ln(2u) & \text{if } u \leq \frac{1}{2} \\ -\ln(2(1-u)) & \text{if } u > \frac{1}{2} \end{cases} \quad (629)$$

where $\frac{1}{2}$ as the transitioning point between the pieces follows intuitively from the symmetry of the PDF around $x = 0$.

The results are illustrated in figure 111.

14.1.3.3 Sampling from a Gaussian using exact inversion | Box-Muller trick

We want to sample from the Gaussian

$$f(y) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{y^2}{2}\right) \quad (630)$$

Problem: The CDF is the error function, which has no closed-form inverse.

Idea: In the Box-Muller trick, a 2D Gaussian in polar coordinates is inverted.

Consider the 2D Gaussian

$$f(x, y) = \frac{1}{2\pi} \exp\left(-\frac{x^2 + y^2}{2}\right) \quad (631)$$

note that we can also write this in polar coordinates $x = r \cos(\theta)$, $y = r \sin(\theta)$ ($r^2 = x^2 + y^2$) as the product of an independent radial and angular probability distribution

$$f(x, y) dx dy = \cancel{f(\phi)} d\phi \cdot f(r) dr = \frac{1}{2\pi} \exp\left(-\frac{r^2}{2}\right) r dr d\phi \quad (632)$$

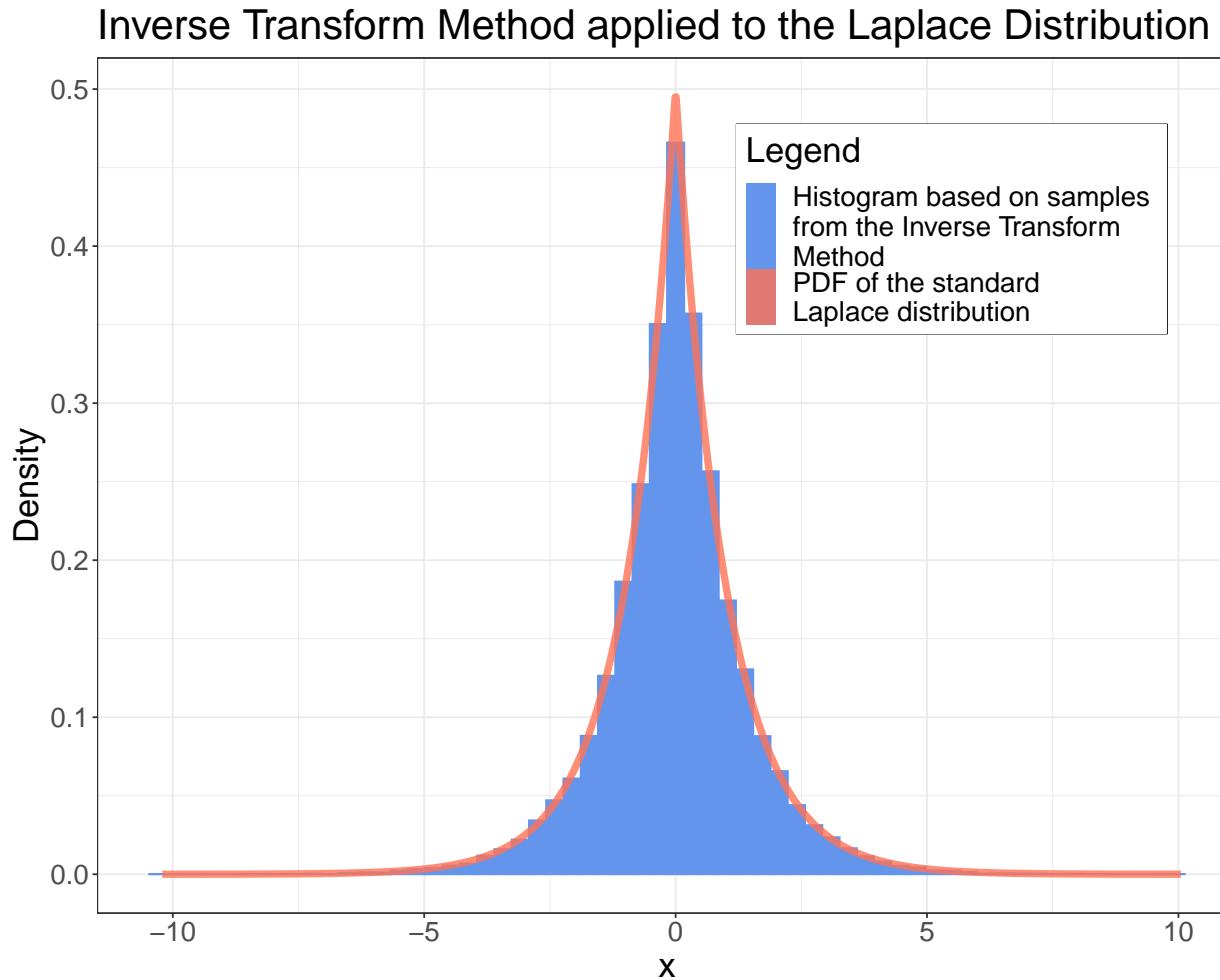


Figure 111: Illustration of the Inverse Transform Method applied to the standard Laplace distribution

where based on random numbers $u_1, u_2 \sim \mathcal{U}(0, 1)$ we can sample both r and ϕ by exact inversion.

$$\begin{aligned}
 u_1 &= \frac{1}{2\pi}\phi \quad \rightarrow \quad \phi = 2\pi u_1 \\
 u_2 &= \int_0^r \exp\left(-\frac{r'^2}{2}\right) dr' = 1 - \exp\left(-\frac{r^2}{2}\right) \quad \rightarrow \quad r = \sqrt{-2 \ln(1 - u_2)}
 \end{aligned} \tag{633}$$

As the 2D Gaussian is just a product of 1D Gaussians, from r and ϕ we can then get two normally distributed numbers x and y .

Box-Buller trick: Given two uniformly distributed random numbers $u_1, u_2 \sim \mathcal{U}(0, 1)$, we can generate two normally distributed random numbers $x, y \sim \mathcal{N}(0, 1)$ by

$$\phi = 2\pi u_1, \quad r = \sqrt{-2 \ln(1 - u_2)} \quad \rightarrow \quad x = r \cos(\phi), \quad y = r \sin(\phi) \quad (634)$$

14.1.3.4 Sampling from a discrete distribution*

Consider a random variable X with the following probability mass function

$$P(X = x) = \begin{cases} 0.1 & \text{if } x = 0 \\ 0.15 & \text{if } x = 1 \\ 0.25 & \text{if } x = 2 \\ 0.3 & \text{if } x = 3 \\ 0.2 & \text{if } x = 4 \\ 0 & \text{otherwise} \end{cases} \quad (635)$$

Intuitively sampling according to such a discrete distribution is simple: For uniform draws between 0 and 1, each section $a \leq x < b$ within has a probability $b - a$. Therefore, we only have to associate each x with a section of length $P(X = x)$.

The sections we are interested in are directly given by the cumulative distribution function, as illustrated in figure 112.

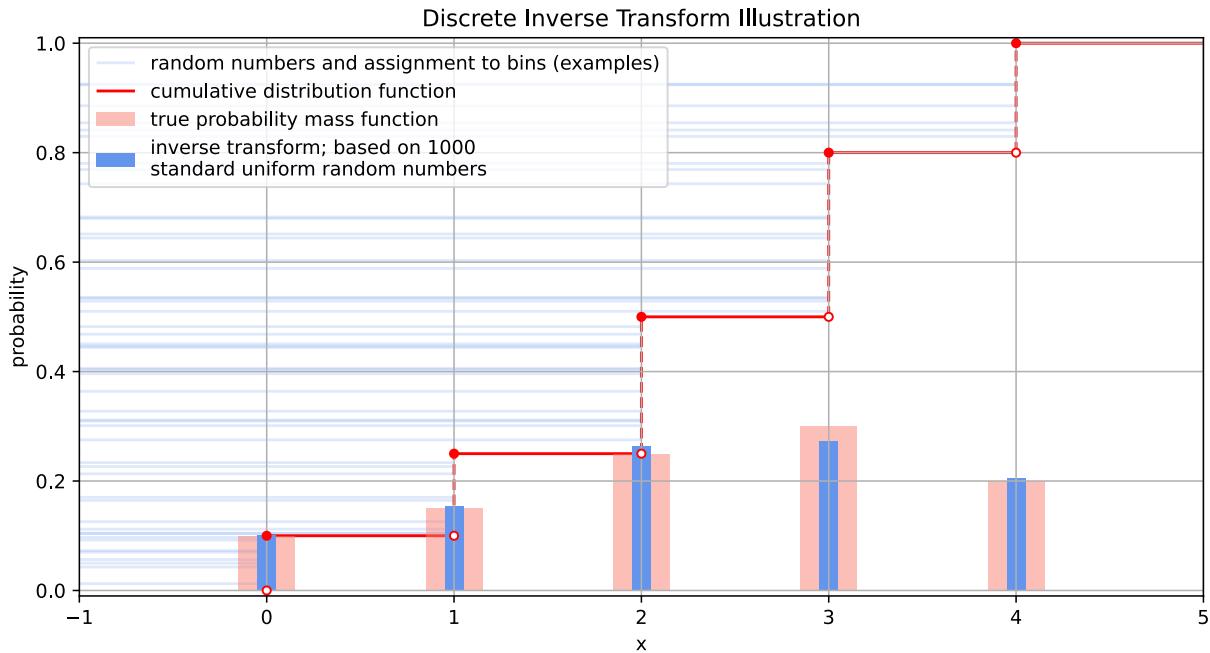


Figure 112: Illustration of the Inverse Transform Method applied to the standard Laplace distribution

14.1.4 Acceptance-rejection method

Let's say we want to sample from a complex distribution $p(x)$ and we can sample from a simple distribution $f(x)$ with $p(x) \leq Cf(x)$, some constant C .

We can then draw samples from $p(x)$ using the Acceptance-Rejection method, consisting of the following steps:

1. Generate a trial value x from $f(x)$ (e. g. using exact inversion)
2. Generate a value y from a uniform distribution $0 \leq y < Cf(x)$
3. If $y \leq p(x)$ return x as the sample value
4. Else, reject the trial value x and draw again

An illustration is provided in figure 113.

Intuition: The logic behind this is very clear if we just choose $f(x) = \text{const.}$ i. e. we uniformly sample x . This, however, would be very inefficient, so we use $Cf(x)$ that lies on top $p(x)$ as closely as possible.

Proof that we get the correct distribution: The probability dq to get (accept) a certain x within dx is

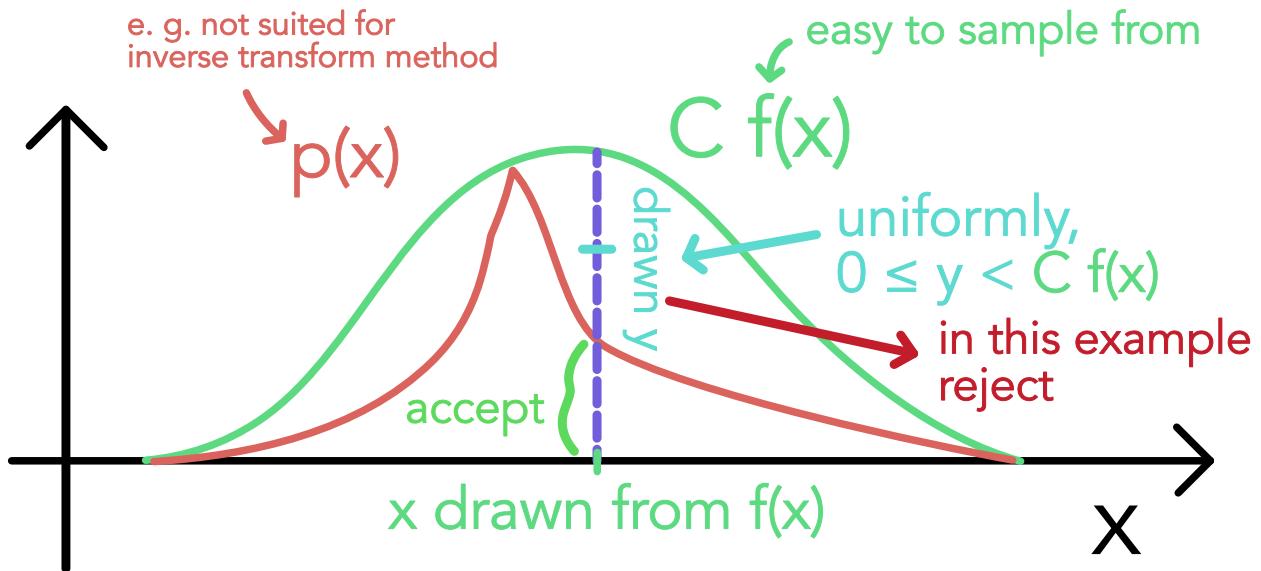


Figure 113: Illustration of the Acceptance-Rejection Method

$$dq = \underbrace{f(x)dx}_{\substack{\text{from} \\ \text{step 1}}} \underbrace{\frac{p(x)}{Cf(x)}}_{\substack{\text{step} \\ \text{2 to 4}}} = \frac{1}{C}p(x)dx \propto p(x)dx \quad (636)$$

Advantages:

- works in any dimension
- $p(x)$ does not have to be normalized (in contrast to exact inversion)

Disadvantages: Note that this is very inefficient if the rejection area is large and efficiency rapidly decreases in higher dimensions (then one might try to construct samples through a stochastic process).

14.1.4.1 Example: Sampling uniformly from a sphere

Aim: Uniformly sample points from the surface of a sphere.

In 3D we can use exact inversion to sample from the surface. The surface element in spherical coordinates is

$$dS = r^2 \sin(\theta) d\theta d\phi = r^2 d\cos(\theta) d\phi \quad (637)$$

(sign?) so $d\cos(\theta)d\phi$ and $d\phi$ are uniform over their range. So we can use

$$\cos \theta = 1 - 2u_1, \quad \phi = 2\pi u_2, \quad u_1, u_2 \sim \mathcal{U}(0, 1) \quad (638)$$

so we can sample from the surface of a sphere by

$$\begin{aligned} x &= r \sin(\theta) \cos(\phi) \\ y &= r \sin(\theta) \sin(\phi) \\ z &= r \cos(\theta) \end{aligned} \quad (639)$$

Problem: Generalization to higher dimensions is not straightforward.

Idea: Use the acceptance-rejection method.

- Uniformly sample $u_1, u_2, u_3 \sim \mathcal{U}(0, 1)$
- If $r_d = u_1^2 + u_2^2 + u_3^2 \leq 1$, return (u_1, u_2, u_3) , else reject and draw again
- Project the points to the surface of the unit sphere by

$$x = \frac{u_1}{r} y = \frac{u_2}{r} z = \frac{u_3}{r} \quad (640)$$

which can easily be extended to higher dimensions.

14.1.4.2 Example II: Sampling from a conditioned Gamma distribution*

We would like to sample from $\text{Gamma}(2, 1)$ conditional on the random variable being greater than 5, i.e.

$$p(x) = \begin{cases} \frac{x \exp(-x)}{6 \exp(-5)} & \text{if } x \geq 5 \\ 0 & \text{otherwise} \end{cases} \quad (641)$$

(normalized). We use the acceptance-rejection method with $f(x)$ being an exponential distribution with $\lambda = 0.5$ conditional on the random variable being greater than 5, i.e.

$$f(x) = \begin{cases} \frac{1}{2 \exp(-\frac{5}{2})} \exp(-\frac{x}{2}) & \text{if } x \geq 5 \\ 0 & \text{otherwise} \end{cases} \quad (642)$$

which we have normalized. We can draw from $f(x)$ using the inverse transform method, where we can e.g. efficiently take care of the conditioning by drawing uniformly between $\int_0^5 \frac{1}{2} \exp(-\frac{x}{2}) dx = 1 - \exp(-\frac{5}{2})$ and 1 in the inverse transform method.

The result is illustrated in figure 114.

As for both the exponential and the gamma distribution of consideration for $x \geq 5$ the PDFs

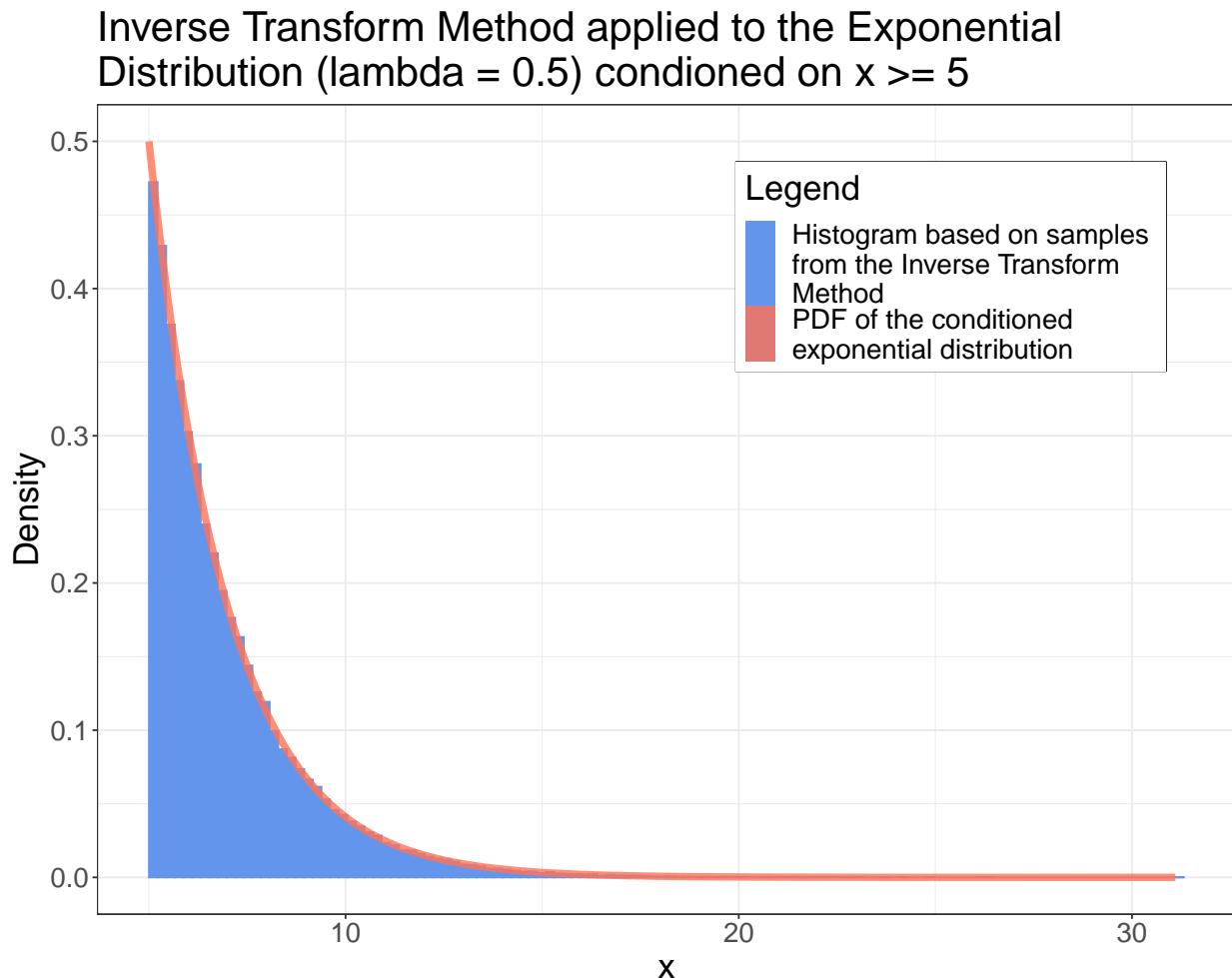


Figure 114: Application of the Inverse Transform Method to sample from a conditioned Exponential distribution

are monotonically decreasing, for C in the acceptance-rejection method, we can use

$$C = \frac{p(5)}{f(5)} = \frac{5}{3} \quad (643)$$

The results from applying the acceptance-rejection method are illustrated in figure 115.

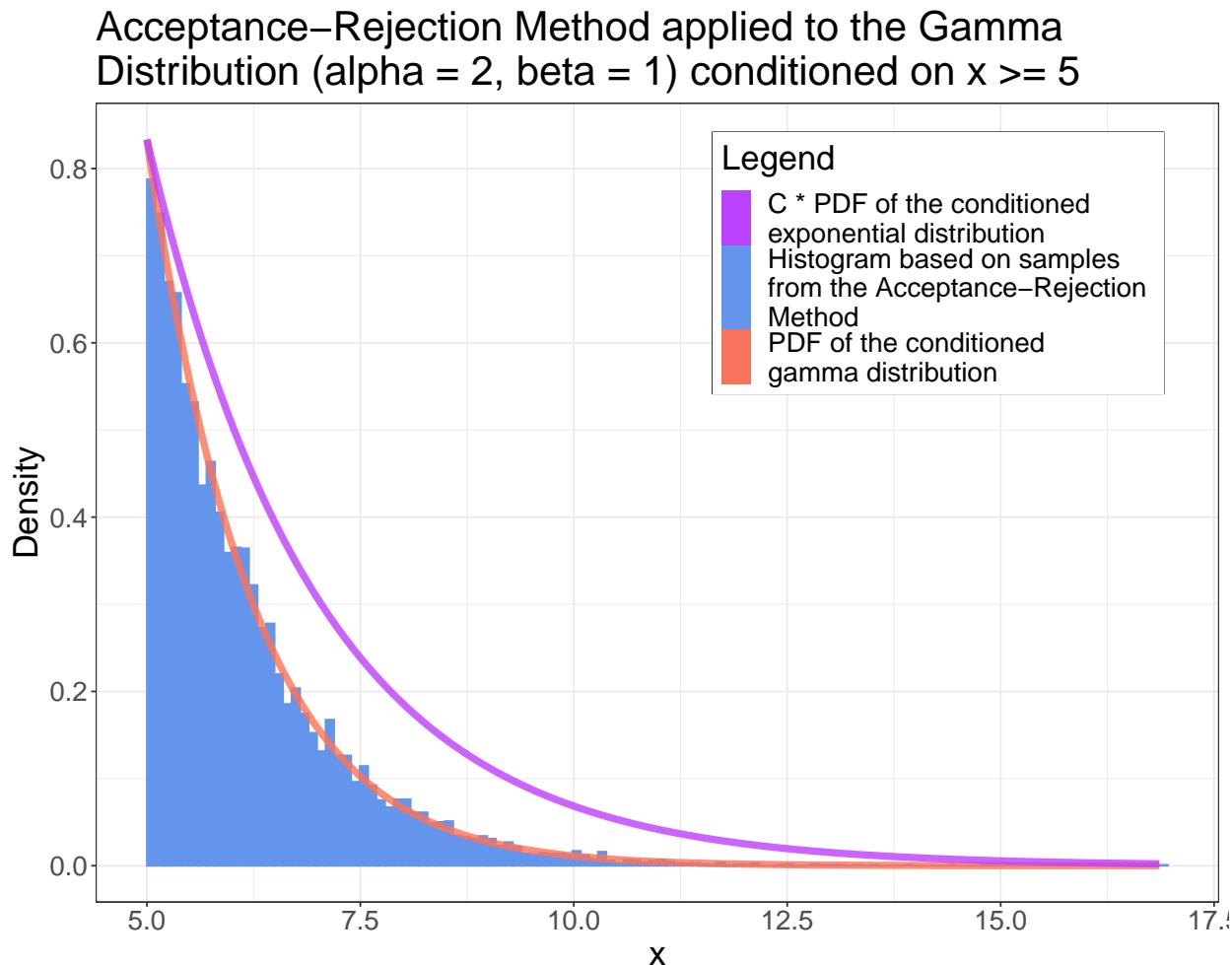


Figure 115: Application of the Acceptance-Rejection Method to sample from a conditioned Gamma distribution

14.2 Monte Carlo Estimation

14.2.1 A basic intuition for estimation

We generally draw information about reality from samples. Consider for instance we measure the energy deposited by energetic muons in a thin silicon layer. We might measure (for different muons of the same energy) $X = [0.5 \text{ MeV}, 0.6 \text{ MeV}, 1.3 \text{ MeV}]$ and we might say that in the mean the energy deposit might in general be 0.8 MeV (without knowing the true distribution, from which we sampled; here quite intricate, see Particle-Data-Group et al., 2020, figure 34.7).

We have just declared our sample estimator to be at least an approximation for the population parameter - which might not be a good idea, especially for such a small sample size.

14.2.2 Monte Carlo Estimation

Monte Carlo Estimation is concerned with estimating statistics by approximating their expectations. So for a random variable with probability distribution $f(x)$ (continuous or discrete) the expectation of a function g , so

$$\theta := E[g(X)] = \begin{cases} \sum_{k=1}^{\infty} g(x_k) \cdot f(x_k) & \text{if } X \text{ is discrete} \\ \int_{-\infty}^{\infty} g(x) \cdot f(x) dx & \text{if } X \text{ is continuous} \end{cases} \quad (644)$$

is approximated by a sample mean over (X_1, \dots, X_m) (m independent realizations from the distribution of X)

$$\hat{\theta} := \frac{1}{m} \sum_{i=1}^m g(X_i) \quad (645)$$

which is an unbiased estimator for θ if the expectation exists and converges almost surely to θ in the limit of a large sample size (by the strong law of large numbers).

14.2.3 Distribution and Error of the Monte Carlo Estimator

Assume that $\text{Var}[g(X)] = \sigma^2 < \infty$. Then by the basic properties of the variance, we can write

$$\text{Var}[\hat{\theta}_m] = \frac{1}{m^2} \sum_{i=1}^m \text{Var}[g(X_i)] = \frac{\sigma^2}{m} \quad (646)$$

Notice that for the distribution of our sample mean (an addition of m i.i.d random variables) the central limit theorem applies, so

$$\frac{\hat{\theta}_m - \theta}{\sigma/\sqrt{m}} \xrightarrow[m \rightarrow \infty]{d} Z \sim \mathcal{N}(0, 1) \quad (647)$$

We would like to use the above to specify confidence intervals for our estimator $\hat{\theta}_m$. Note, however, that σ is unknown to us. Luckily, the above still holds in the limit of a large sample size when using the estimator

$$S_n^2 := \frac{1}{m-1} \sum_{i=1}^m (g(X_i) - \hat{\theta}_m)^2 \quad (648)$$

or rather the square root S_n . Notice that while S_n is different from S_n^2 , not an unbiased estimator, it converges in probability to σ which is sufficient to apply Slutsky's theorem yielding

$$\frac{\hat{\theta}_m - \theta}{S_n/\sqrt{m}} \xrightarrow[m \rightarrow \infty]{d} Z \sim \mathcal{N}(0, 1) \quad (649)$$

14.3 Monte Carlo Integration

Classical numerical methods for integration (e.g. Simpson's rule) become infeasible for high-dimensional integrals, essentially, as the errors scale with the number of function-evaluation-points per dimension (so in high dimensions, we need to perform lots of function evaluations).

14.3.1 Intuition for Monte Carlo Integration

Consider we want to calculate the integral

$$I := \int_V g(\underline{x}) d^d \underline{x} \quad (650)$$

where V is a d -dimensional volume and g is a function $g : V \rightarrow \mathbb{R}$. Then it makes intuitive sense to approximate this integral by the mean of the function g on the domain of interest, as found as the mean of g evaluated at N random points \underline{x}_i in V , multiplied by the volume of V (also denoted by V)

$$\hat{I}_N := \frac{V}{N} \sum_{i=1}^N g(\underline{x}_i) \xrightarrow[N \rightarrow \infty]{a.s.} I \quad (651)$$

Monte Carlo Integration: Let V for simplicity be the hypercube $[0, 1]^d$ (by change of variables, the domain of interest can be mapped to this hypercube). Then Monte Carlo integration to approximate $I = \int_{[0,1]^d} g(\underline{x}) d^d \underline{x}, \underline{x} \in \mathbb{R}^d$ is given by

- Generate N random vectors \underline{x} where each component is drawn uniformly from $[0, 1]$ (so we need $N \times d$ random numbers)
- Evaluate

$$\hat{I}_N = V \frac{1}{N} \sum_{i=1}^N g(\underline{x}_i), \quad \text{here } V = 1 \quad (652)$$

Scaling of the error: The error of the Monte Carlo integrator scales with $1/\sqrt{N}$, independent of the number of dimensions of the integration d .

14.3.2 Connection to the Monte Carlo Estimator

We can also obtain the result from above from the Monte Carlo Estimator (eq. 644) by using the uniform distribution

$$f(\underline{x}) = \begin{cases} \frac{1}{V} & \text{if } \underline{x} \in V \\ 0 & \text{else} \end{cases} \quad (653)$$

and rewriting the integral as

$$I = \int_V g(\underline{x}) d^d \underline{x} = V \int_V g(\underline{x}) \cdot \frac{1}{V} d^d \underline{x} = V \int_V g(\underline{x}) \cdot f(\underline{x}) d^d \underline{x} \quad (654)$$

14.3.3 Intuition from calculating the area of a shape

Consider an irregular shape with area A_{shape} within a rectangle with area A_{rect} . The probability that a point drawn uniformly from the rectangle lies within the shape is

$$p_{\text{shape}} = \frac{A_{\text{shape}}}{A_{\text{rect}}} \quad (655)$$

so we can approximate

$$A_{\text{shape}} \approx A_{\text{rect}} \frac{\# \text{ hits in shape}}{\# \text{ hits in rectangle}} \quad (656)$$

which is the same as the Monte Carlo Integration introduced, where $g(\underline{x}) = 1_{\text{shape}}(\underline{x})$ and $V = A_{\text{rect}}$.

14.3.4 Error in Monte Carlo Integration - scaling with $\frac{1}{\sqrt{N}}$

Based on the connection to the Monte Carlo Estimator, the variance of our estimator for the integral is given by

$$\text{Var}[\hat{I}_N] = \frac{V^2}{N} \text{Var}[g(x)], \quad \text{estimate Var}[g(x)] \text{ by } S_{g;N}^2 := \frac{1}{N-1} \sum_{i=1}^N \left(g(x_i) - \frac{\hat{I}_N}{V} \right)^2 \quad (657)$$

So the standard error of our Monte Carlo estimator

$$\hat{I}_N = \frac{V}{N} \sum_{i=1}^N g(\underline{x}_i) \quad (658)$$

that is if we calculate \hat{I}_N multiple times with different random samples, the standard deviation of these different estimates is

$$\sigma_{\hat{I}_N} = V \sqrt{\frac{\sigma_g^2}{N}} \propto \frac{1}{\sqrt{N}} \quad (659)$$

which is illustrated in figure 116. This follows directly from the basic properties of the variance

$$\text{for } X, Y \text{ independent} \quad \text{Var}[X + Y] = \text{Var}[X] + \text{Var}[Y], \quad \text{Var}[aX] = a^2 \text{Var}[X] \quad (660)$$

with

$$\text{Var}[\hat{I}_N] = \text{Var} \left[\frac{V}{N} \sum_{i=1}^N g(\underline{x}_i) \right] = \frac{V^2}{N^2} \sum_{i=1}^N \text{Var}[g(\underline{x}_i)] \underset{\underline{x}_i \text{ i.i.d.}}{=} \frac{V^2}{N} \text{Var}[g(\underline{x})] \quad (661)$$

14.3.5 Distribution of \hat{I}_N and derivation of the central limit theorem*

The Monte Carlo Estimator is

$$\hat{I}_N = \frac{V}{N} \sum_{i=1}^N g(\underline{x}_i) = \frac{1}{N} \sum_{i=1}^N y_i = \sum_{i=1}^N s_i, \quad y_i = Vg(\underline{x}_i), \quad s_i = \frac{y_i}{N} \quad (662)$$

by basic properties of variance and mean

$$\begin{aligned} \langle \hat{I}_N \rangle &= N \langle s \rangle = \langle y \rangle \\ \text{Var}[\hat{I}_N] &= N \text{Var}[s] = \frac{1}{N} \text{Var}[y] = \frac{V^2}{N} \text{Var}[g(\underline{x})] \end{aligned} \quad (663)$$

Note: $p(y)$ is the distribution of y over V along the y -axis with

$$\langle y \rangle = \int y p(y) dy = \int_V g(\underline{x}) d^d \underline{x}, \quad \int p(y) dy = 1 \quad (664)$$

$p(y)$ might be any distribution.

Applying the Central Limit Theorem (eq. 405) to the Monte Carlo Estimator (assuming

the moments of $p(y)$ exist and are finite), we obtain

$$\begin{aligned} P_N(\hat{I}_N) &= \frac{1}{\sqrt{2\pi \text{Var}[\hat{I}_N]}} \exp\left(-\frac{(\hat{I}_N - \langle \hat{I}_N \rangle)^2}{2 \text{Var}[\hat{I}_N]}\right) \\ &= \frac{\sqrt{N}}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{N(\hat{I}_N - \langle y \rangle)^2}{2\sigma_y^2}\right) \end{aligned} \quad (665)$$

independent of the shape of $p(y)$ for N sufficiently large.

Derivation of the Central Limit Theorem by Fourier Transform The central limit theorem can be derived by considering P_N in Fourier space, where the central ingredient will simply be

$$\exp x = \lim_{N \rightarrow \infty} \left(1 + \frac{x}{N}\right)^N \quad (666)$$

which is also approximately true for large N .

Distribution of the sum of independent random variables: Consider independent variables X, Y with PDFs f_X and f_Y . The PDF of the sum $Z = X + Y$ is naturally the convolution

$$f_Z(z) = \int_{-\infty}^{\infty} \underbrace{f_X(x)f_Y(z-x)}_{x+(z-x)=z} dx \quad (667)$$

Another way to write this is

$$f_Z(z) = \int f_X(x)f_Y(y)\delta(z - (x + y)) dx dy \quad (668)$$

which one can easily see is equivalent to the convolution and also makes intuitive sense - the δ -function ensures that the sum of x and y is z .

Analogously to the two-variable case, we write

$$P_N(\hat{I}_N) = \int \delta\left(\hat{I}_N - \sum_i \frac{y_i}{N}\right) p(y_1)p(y_2)\cdots p(y_N) dy_1 dy_2 \cdots dy_N \quad (669)$$

Based on the Fourier transform and expansion of $p(y)$

$$\begin{aligned} \hat{p}(k) &= \int p(y) \exp(ik(y - \langle y \rangle)) dy \\ &\stackrel{\text{Series Expansion}}{=} \int p(y) \left[1 + ik(y - \langle y \rangle) - \frac{k^2}{2}(y - \langle y \rangle)^2 + \dots\right] dy \\ &= 1 - \frac{k^2}{2}\sigma_y^2 + \dots \end{aligned} \quad (670)$$

we find the Fourier transform of P_N to be

$$\begin{aligned}
 \hat{P}_N(k) &= \int P_N(\hat{I}_N) \exp(ik(I_N - \langle I_N \rangle)) dI_N \\
 &\stackrel{\langle \hat{I}_N \rangle = \langle y \rangle}{=} \int p(y_1) \cdots p(y_N) \exp\left(i\frac{k}{N}(y_1 - \langle y_1 \rangle + y_2 - \langle y_2 \rangle + \dots + y_N - \langle y_N \rangle)\right) dy_1 \cdots dy_N \\
 &= \left[\hat{p}\left(\frac{k}{N}\right) \right]^N \\
 &\stackrel{\text{series expansion of } \hat{p}}{=} \left(1 - \frac{k^2 \sigma_y^2}{2N^2}\right)^N \\
 &\stackrel{N \text{ large}}{\approx} \exp\left(-\frac{k^2 \sigma_y^2}{2N}\right)
 \end{aligned} \tag{671}$$

where the inverse Fourier transform of $\hat{P}_N(k)$

$$P_N(\hat{I}_N) = \frac{1}{2\pi} \int \exp(-ik(I_N - \langle I_N \rangle)) \hat{P}_N(k) dk \tag{672}$$

yields the previously stated result for $P_N(\hat{I}_N)$ ²³.

14.3.6 Illustrative Example of Monte Carlo Integration

Consider the integral

$$I = \int_0^1 g(x) dx, \quad g(x) = \exp\left(-\frac{x^2}{2}\right) \tag{673}$$

Our Monte Carlo estimator for this integral is given by

$$\hat{I}_N = \frac{1}{N} \sum_{i=1}^N g(x_i), \quad x_i \sim \mathcal{U}(0, 1) \tag{674}$$

which is illustrated in figure 116. The higher the number of sampled points N the lower the variance of our estimator; the lower the variance of g along the y axis, the lower the variance of our estimator.

²³Insert the result for $\hat{P}_N(k)$, complete the square and use the normalization of the normal distribution or the Gaussian integral $\int \exp(-\alpha x^2) dx = \sqrt{\frac{\pi}{\alpha}}$.

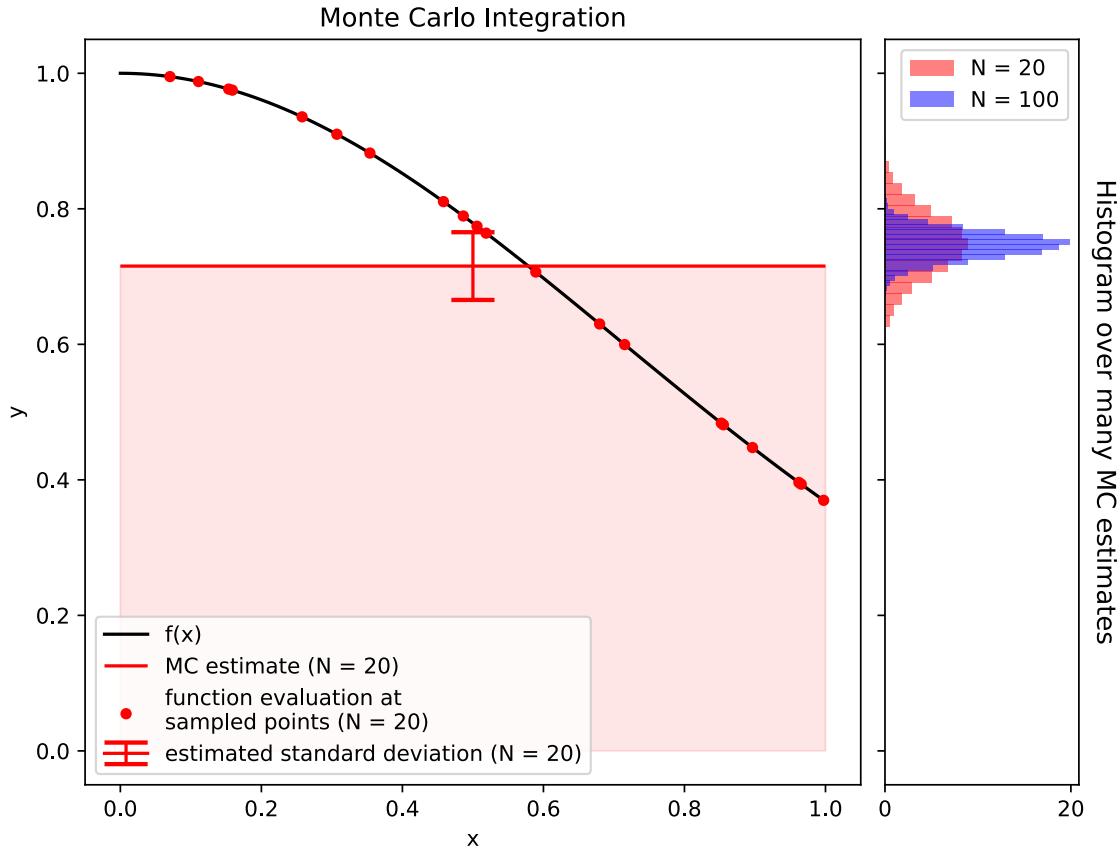


Figure 116: Illustration of Monte Carlo Integration for $I = \int_0^1 \exp\left(-\frac{x^2}{2}\right) dx$

14.3.7 Comparison to other techniques - when to use Monte Carlo integration?

Error of standard methods In a standard integration scheme like the midpoint rule or Simpsons rule²⁴ each dimension is divided into n regularly space points, so that we have $N = n^d$ points in total. The error scales with

$$\text{midpoint or trapezoidal rule } \propto \frac{1}{n^2}, \quad \text{Simpson's rule } \propto \frac{1}{n^4} = \frac{1}{N^{\frac{4}{d}}} \quad (675)$$

In these *regular* methods, adding more points to the integration (increasing N) helps less and less the higher we go in dimension - e.g. for Simpson's rule $\text{err} \propto \frac{1}{N^{\frac{4}{d}}}$. For a standard method if we want 10 points per dimension with $d = 10$ we already need 10^{10} function evaluations - infeasible (e.g. high-dimensional integrations in Machine Learning).

²⁴ $\int_a^b f(x) dx \approx \frac{b-a}{6} [f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)]$ which can be derived by approximating the function f by a quadratic polynomial and integrating this polynomial.

On the other hand in Monte Carlo integration the error scales with $\frac{1}{N^{\frac{1}{d}}}$ independent of the dimensionality d .

Starting at what dimension of the integral d will MC integration have a more favorable error scaling compared to Simpson's rule? Setting $\frac{1}{N^{\frac{1}{d}}} = \frac{1}{N^{\frac{1}{2}}}$ yields that starting at $d = 8$ Monte Carlo Integration's error will reduce more quickly if we increase N the number of points (/ function evaluations) used for the integration.

Intuition for the better scaling One intuition for this advantage of Monte Carlo is that the higher the dimension, the more evenly pairs of random points are spread with respect to their pair-wise distance - normally known as the curse of dimensionality, illustrated in figure 117.

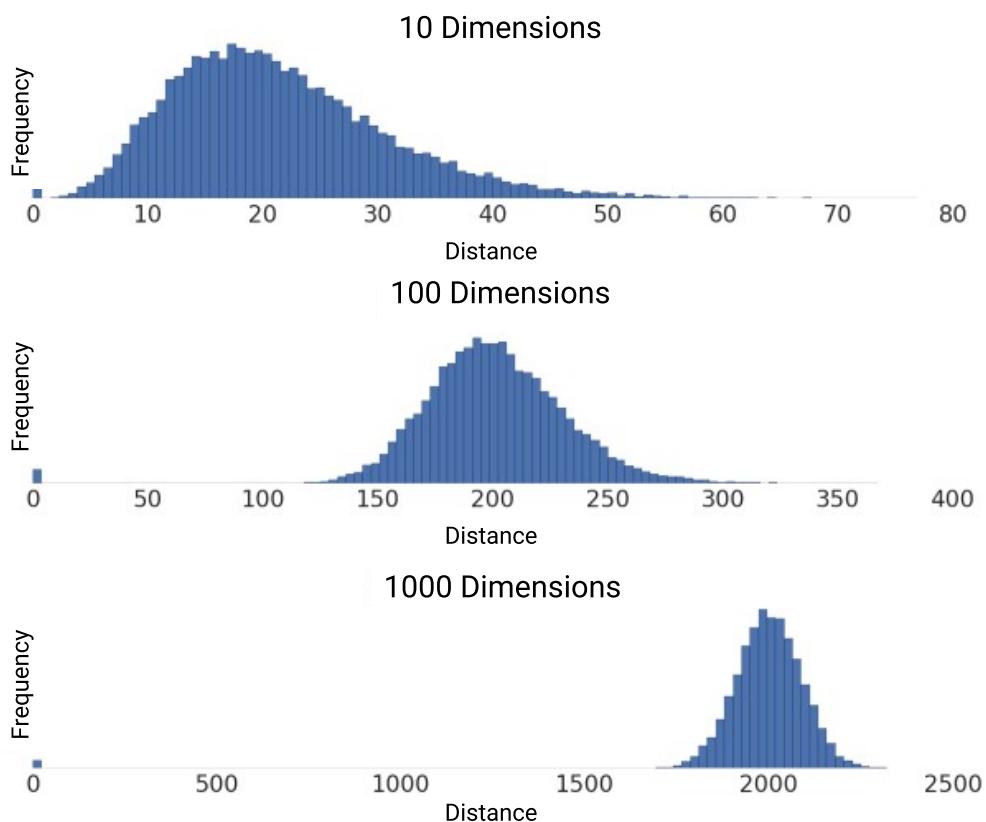


Figure 117: In higher dimensions, pairwise distances between random points show a sharper distribution around the mean distance (a more even distribution in space)

14.3.8 Reducing the variance of the Monte Carlo Estimator

14.3.8.1 Antithetic Estimators*

Two random variables with the same distribution on the same probability space are called *antithetic*, if their covariance is negative. We can use such antithetic variables to reduce the variance of a Monte Carlo estimator.

This motivates the following ansatz

$$\begin{aligned} I &:= \int_0^1 g(x)dx, \quad \hat{I}_{\text{anti}} = \frac{\hat{I}_{N/2} + \tilde{I}_{N/2}}{2} \\ \hat{I}_{N/2} &= \frac{1}{N/2} \sum_{i=1}^{N/2} g(u_i), \quad \tilde{I}_{N/2} = \frac{1}{N/2} \sum_{i=1}^{N/2} g(1 - u_i), \quad u_i \sim \mathcal{U}(0, 1) \end{aligned} \tag{676}$$

with

$$\text{Var}[\hat{I}_{\text{anti}}] = \frac{1}{4} \left(\text{Var}[\hat{I}_{N/2}] + \text{Var}[\tilde{I}_{N/2}] + 2 \text{Cov}[\hat{I}_{N/2}, \tilde{I}_{N/2}] \right) \underset{\text{see eq. 657}}{=} \text{Var}[\hat{I}_N] - \frac{1}{2} \text{Cov}[\hat{I}_{N/2}, \tilde{I}_{N/2}] \tag{677}$$

where for g continuous and monotonic, $\text{Cov}[\hat{I}_{N/2}, \tilde{I}_{N/2}]$ is negative, as U and $1 - U$ with $U \sim \mathcal{U}(0, 1)$ are negatively correlated. So we can reduce the variance compared to an estimator \hat{I}_N using the same number of function evaluation and only half the number of random numbers.

Let us give an intuition why using antithetic evaluation points makes sense for estimating an integral of a monotonic function from 0 to 1. Using uniform antithetic variables we create pairs of points $(x_i, 1 - x_i)$, which are symmetric around $x = 0.5$ and as of the monotony of the function g we want to integrate, we obtain a better balance between sampling points where g is large and where g is small, reducing the variance of the estimate.

Example of using Antithetic Estimators

Consider the integral

$$I = \int_0^2 \frac{1}{1+x^2} dx \underset{\text{exact solution}}{=} \arctan(2) \approx 1.107149 \tag{678}$$

Estimates based on the standard Monte Carlo estimator and the antithetic estimator are given in table 20 and illustrated in figure 118. A relative reduction in standard deviation of roughly 80% is achieved using the antithetic estimator.

Estimator	Estimate	Standard Deviation	95% Confidence Interval
Standard Monte Carlo	1.096	0.017	[1.0627 1.128]
Antithetic Monte Carlo	1.106	0.0033	[1.099 1.112]

Table 20: Rounded estimates for the integral $\int_0^2 \frac{1}{1+x^2} dx$ using the standard Monte Carlo estimator and the antithetic Monte Carlo estimator for $N = 100$ samples with seed 1234

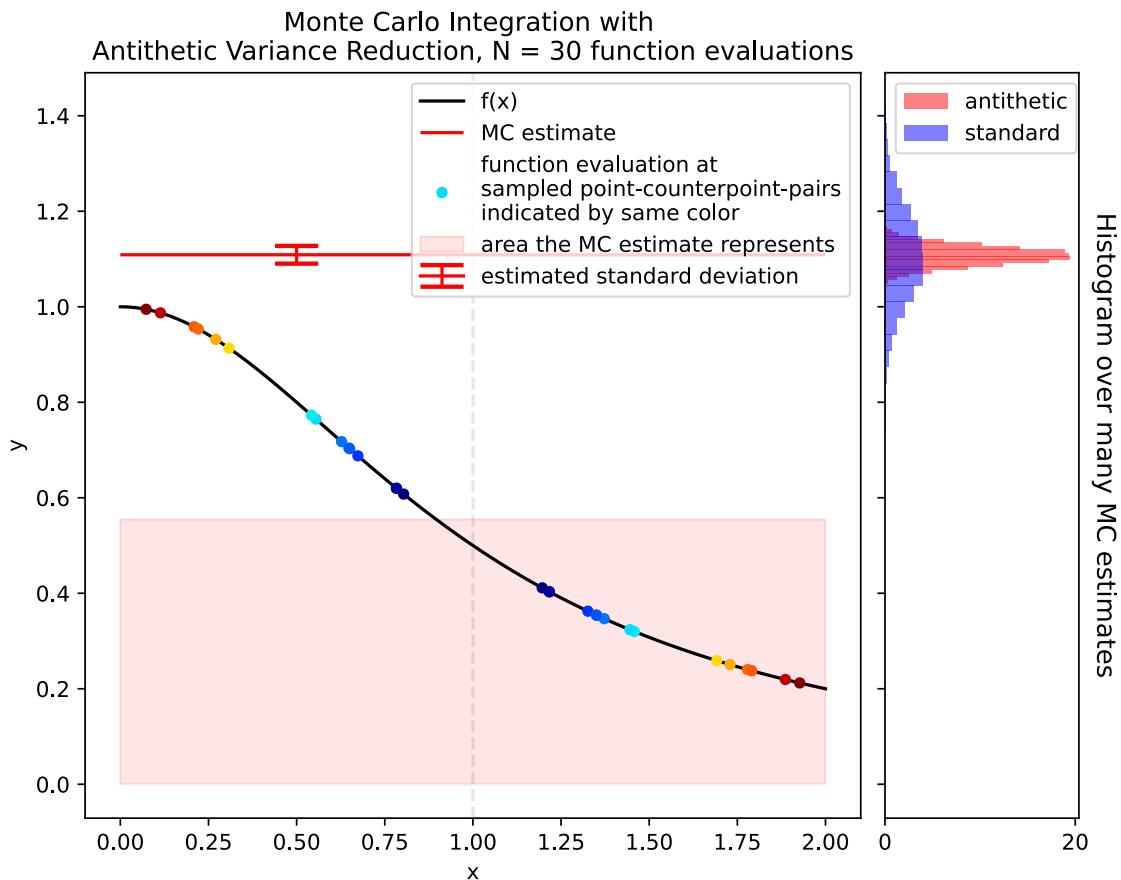


Figure 118: Illustration of Monte Carlo Integration with Antithetic Variables. Notice the even distribution of points around $\frac{a+b}{2} = 1$ by construction yielding an advantage over the standard Monte Carlo estimator.

14.3.8.2 Importance Sampling

Problem: Consider we have a sharply peaked function $g(x)$ over the domain of integration. In this case using evaluation points sampled from the uniform will be inefficient.

Importance sampling idea: Preferably choose points where the integrand is large. This is done by sampling from a distribution $f(x)$ similarly shaped as $g(x)$ but from which we can easily sample (e.g. by direct inversion^a or the rejection method).

^aNote that for the inverse transform we need the inverted CDF, so we need to integrate the PDF. So choosing $f(x)$ itself as $g(x)$ is non-sense as we would need to integrate $f(x)$ to obtain the CDF.

The higher the variance of the output of the function $g(x)$ over the domain of integration, the higher the variance in our Monte Carlo Estimate for our integral.

$$I := \int_V g(x)dx = \int_V \frac{g(x)}{f(x)} f(x)dx, \quad I_N = \frac{1}{N} \sum_{i=1}^N \frac{g(x_i)}{f(x_i)}$$

sample $\{x_i\}_{i=1,\dots,N}$ drawn from $f(x)$ limited to V

(679)

Let us as an example tackle the integral

$$I = \int_0^1 \frac{4}{1+x^2} dx \underset{\substack{= \\ \text{exact solution}}}{\sim} \pi$$
(680)

using the importance sampling function $f(x) = \frac{4-2x}{3}$ on $0 \leq x \leq 1$. We can sample from $f(x)$ using the inverse transform method using the CDF $F(x) = \frac{4x-x^2}{3}$ (with $F(1) = 1$ as we want) so $F^{-1}(u) = 2 - \sqrt{4 - 3u}$. We can then simply use equation 679 to obtain an estimate.

The importance sampling approach compared to the standard Monte Carlo approach is illustrated in figure 119 and 120. There it is easy to see how effectively integrating a flatter function is to our advantage.

Advantage of Importance Sampling: As we can see in figure 119 the flatter $h(x) := \frac{g(x)}{f(x)}$ (best $h \propto g$ as achievable in lattice Monte Carlo) (mind the different naming in the figure) the sharper the probability distribution $p(h)$ (a limited range of h values with higher probabilities), the smaller the Monte Carlo error $\sigma_{I_N} = \sqrt{\frac{\sigma_h^2}{N}}$.

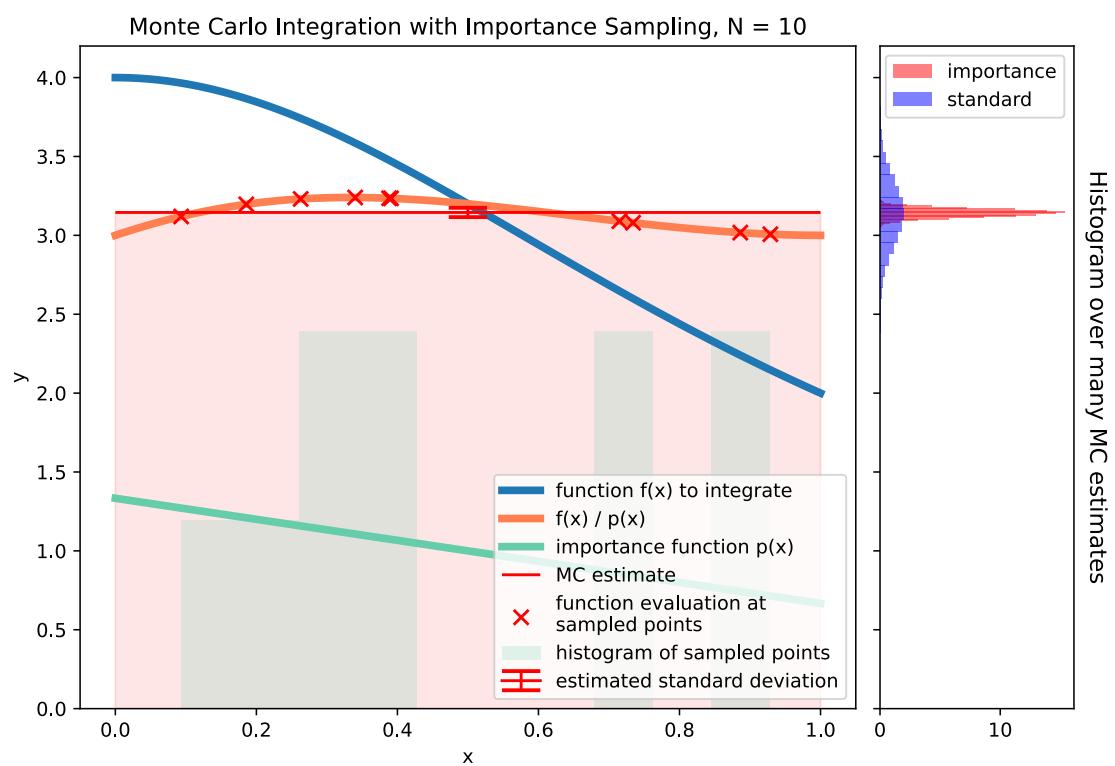


Figure 119: Illustration of Monte Carlo Integration with Importance Sampling for $N = 10$ samples.

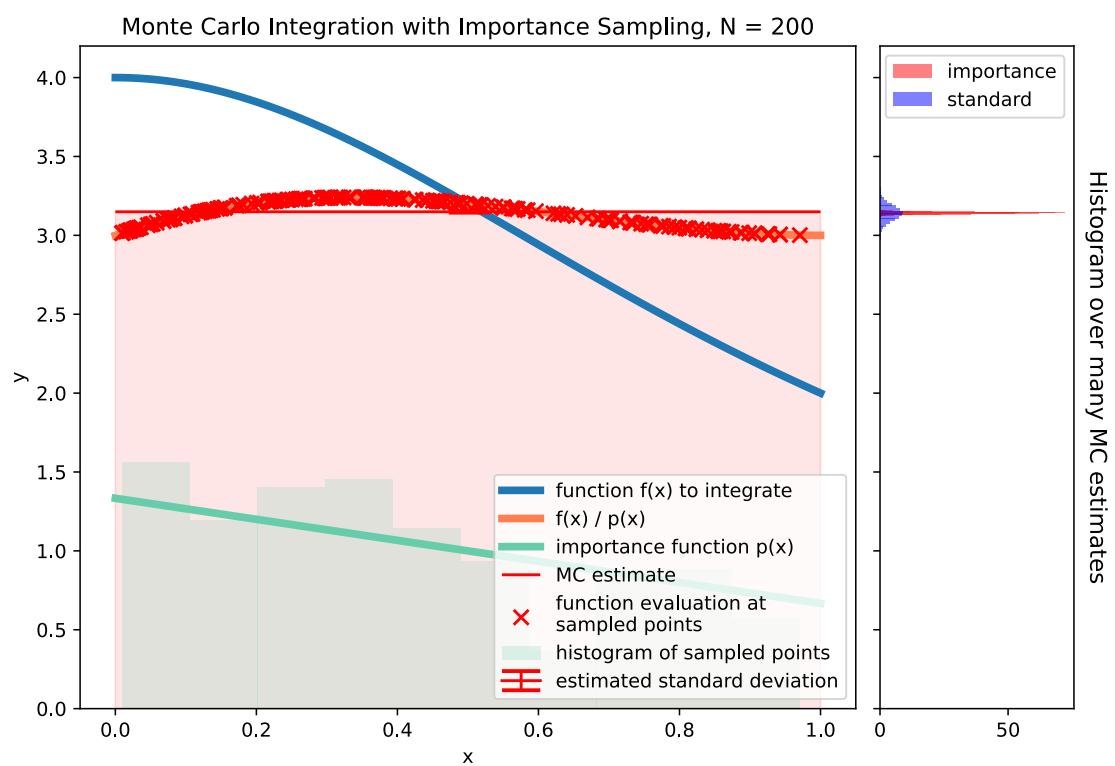


Figure 120: Illustration of Monte Carlo Integration with Importance Sampling for $N = 200$ samples.

14.4 Sampling with a stochastic process

Let us recapitulate

- We first introduced how one can sample uniformly distributed random numbers, based on generating an integer sequence (e.g. the infamously bad RANDU)
- We then introduced how one can - based on the capability of sampling from the uniform distribution - sample from other distributions. If the Cumulative Distribution Function (CDF) of a distribution is invertible, we can use the inverse transform method if not for instance the rejection method can be used.
- We have seen how uniform random numbers can be used in standard Monte Carlo integration and how the variance of the Monte Carlo estimate can be reduced by importance sampling - based on our ability to sample from a given distribution.

Problem: What can we do if neither direct inversion nor the rejection method are viable for sampling from a distribution $p(\underline{x})$, e.g. if p is complicated and \underline{x} very high-dimensional?

In the following we will construct a stochastic process, a sequence of numbers where the next one is chosen stochastically, so that if we at some point take a sequence of those numbers, they are a sample of the distribution $p(\underline{x})$, so if we take a sufficiently large sequence, its histogram will approximate the distribution $p(\underline{x})$ - **the process has $p(\underline{x}) = p_{eq}(\underline{x})$ as its equilibrium distribution.**

14.4.1 Markov Process and Markov chain | Monte Carlo Markov Chain (MCMC)

Under conditions detailed later, a discrete sequence of states

$$x_1 \xrightarrow{f} x_2 \xrightarrow{f} x_3 \xrightarrow{f} \dots \xrightarrow{f} x_n, \quad \text{Monte-Carlo operator } f \quad (681)$$

is called **Markov chain generated by a Markov process**.

We will first discuss characterizing properties of Markov processes and then introduce an algorithm the Metropolis Hastings algorithm fulfilling these.

14.4.1.1 Characterizing property of the Markov process | Memorylessness

The probabilistic Monte Carlo steps f from one state of the system to the next (one number to the next) are taken with a transition probability W_f

$$W_f(x \rightarrow x') = W_f(x'|x) \quad (682)$$

which only depends on the current state - no information on the history is used.

This conditional transition property has the natural properties

$$\int W_f(x \rightarrow x') dx' = 1, \quad W_f(x \rightarrow x') \geq 0 \quad (683)$$

Memorylessness also means that if you imagine two frogs hopping on the Markov chain, once they meet, they will be coupled (in love) forever. *Ergodicity* means that they will eventually meet in finite time.

14.4.1.2 Transitioning the probability distribution

The probability of a value x' depending on the prior distribution $p(x)$ is

$$p(x) \xrightarrow{f} p'(x') = \int p(x) W_f(x \rightarrow x') dx \quad (684)$$

Idea: Imagine this process would have an equilibrium, fixed-point distribution $p_{eq}(x)$, then subsequent applications of f will yield us a sequence of numbers that are a sample of $p_{eq}(x)$.

14.4.1.3 Properties demanded of the update step f | stochastic process has equilibrium distribution | ergodicity

1. Equilibrium distribution: There is an equilibrium distribution $p_{eq}(x)$ that is preserved by the update step (once reached, no other distribution comes forth), i.e.

$$p_{eq}(x) = \int p_{eq}(x) W_f(x \rightarrow x') dx \quad (685)$$

so $p_{eq}(x)$ is a fixed point of the update step f .

2. Ergodicity - entire domain must be reachable: Starting from any state x by repeatedly applying f we must be able to get arbitrarily close to any other state x' .

14.4.1.4 Nice consequences of the demanded properties of the update step

- Ensemble approaches equilibrium distribution: Any ensemble of states (a set of numbers) approaches the equilibrium distribution $p_{eq}(x)$ if f is applied sufficiently often.
- Collection of states from a chain approaches equilibrium distribution: The collection of states x_1, \dots, x_n in a single Markov chain under action of f approaches p_{eq} as the number of steps n goes to infinity.

We therefore have two principal ways of sampling: **ensemble sampling** (we apply f repeatedly to a set of states, which at some point are samples from p_{eq}) and **chain sampling** (starting from one initial state samples are taken as sequences from the chain).

14.4.1.5 Proof that updating the probability distribution converges to its fixed-point equilibrium distribution p_{eq}

We proof this by first showing that each update step f will bring us closer to the equilibrium distribution and secondly that there is only one fixed-point distribution p_{eq} .

$p'(x')$ is closer to $p_{eq}(x')$ than $p(x)$ as

$$\begin{aligned}
 \|p' - p_{eq}\| &\equiv \int |p'(x') - p_{eq}(x')| dx' \\
 &\stackrel{\text{plug in } p'(x'), p_{eq}(x')}{=} \int \left| \int (p(x) - p_{eq}(x)) W_f(x \rightarrow x') dx \right| dx' \\
 &\stackrel{\text{triangle inequality}}{\leq} \int \int |p(x) - p_{eq}(x)| W_f(x \rightarrow x') dx dx' \\
 &\stackrel{W_f \text{ normed}}{=} \int |p(x) - p_{eq}(x)| dx \\
 &= \|p - p_{eq}\|
 \end{aligned} \tag{686}$$

so $\|p' - p_{eq}\| \leq \|p - p_{eq}\|$.

Why can't there be two fix-point distributions p_{eq}, p'_{eq} with $\|p_{eq} - p'_{eq}\| > 0$? As of ergodicity all states are reachable from any other state. As of memorylessness the transition only depends on the current state. If there were two fixed-point distributions, as of the above properties, nothing would prevent the system from transitioning from one fixed-point distribution to the other - and they therefore would not be fixed-point distributions in the first place. So it is really $\|p' - p_{eq}\| < \|p - p_{eq}\|$.

Note: Ergodicity is more easily followed by f in practice than one might imagine. A Markov chain is ergodic if and only if it has at most one recurrent class^a and is aperiodic.

^aState i is recurrent, if for all states j for which there is a non-zero probability to get to i the same is true the other way around. A set of recurrent states is a recurrent class.

14.4.1.6 Detailed balance | Common condition for the update steps

A stronger version of p_{eq} must be a fix-point of the update step is

$$p_{eq}(x)W_f(x \rightarrow x') = p_{eq}(x')W_f(x' \rightarrow x) \rightarrow p_{eq} \text{ is fix-point under } f \tag{687}$$

where the looser fix-point condition (eq. 685) follows by integrating over x .

Therefore a Markov Chain fulfilling detailed balance has p_{eq} as equilibrium. In total we want W_f to fulfill detailed balance and ergodicity.

But how do we choose W_f ?

14.4.2 Metropolis Hastings algorithm - simple and generic construction of the transition f

Starting from a current state x we want to go to the next one x' . Let p be the distribution we want to sample from. How do we choose the next state?

1. x' is proposed with the proposal probability $q(x \rightarrow x')$ (fairly arbitrary, must be ergodic though)

A simple symmetric update step could be

$$q(x \rightarrow x') : x' = x + e, \quad e \text{ distributed symmetrically around } 0 \quad (688)$$

e.g. $e \sim \mathcal{N}(0, \sigma^2)$ so $q(x \rightarrow x') = \mathcal{N}(x - x', \sigma^2)$

2. Calculate the Hastings ratio

$$r = \min \left(1, \frac{p(x') q(x' \rightarrow x)}{p(x) q(x \rightarrow x')} \right) \in [0, 1] \quad (689)$$

distribution p we want to sample from.

This is the only place where the probability we want to sample from, p , comes into play. Note that the normalization of p cancels out, so we do not need to normalize p in the Metropolis Hastings algorithm.

3. Accept x' with probability r , i.e. draw a random number $u \in [0, 1)$ and

- (a) $r \geq u$: accept x' as the next state
- (b) $r < u$: reject x' and add x again to the Markov Chain

14.4.2.1 The Metropolis Hastings Algorithm fulfills detailed balance

We want to show detailed balance, so

$$p_{eq}(x) W_f(x \rightarrow x') = p_{eq}(x') W_f(x' \rightarrow x) \quad \rightarrow \quad p_{eq} \text{ is fix-point under } f \quad (690)$$

For this let us start by calculating $W_f(x \rightarrow x')$ assuming $r < 1$ ($r = 1$ follows analogously)

$$W_f(x \rightarrow x') = q(x \rightarrow x') \frac{p(x') q(x' \rightarrow x)}{p(x) q(x \rightarrow x')} = q(x' \rightarrow x) \frac{p(x')}{p(x)} \quad (691)$$

Notice that based on the definition of the Hastings ratio, for $r < 1$ we have $r' = 1$, as

$$\text{assumption } r < 1 \rightarrow \frac{p(x') q(x' \rightarrow x)}{p(x) q(x \rightarrow x')} < 1 \rightarrow r' = \min\left(1, \frac{p(x) q(x \rightarrow x')}{p(x') q(x' \rightarrow x)}\right) = 1 \quad (692)$$

So for $W_f(x' \rightarrow x)$ we get

$$W_f(x' \rightarrow x) = q(x' \rightarrow x) \cdot 1 \quad (693)$$

which together with the expression for $W_f(x \rightarrow x')$ we obtain detailed balance.

14.4.2.2 Metropolis update | for a symmetric proposal operator the Hastings ratio simplifies

Consider the case of a symmetric proposal operator (e.g. the one already introduced)

$$q(x' \rightarrow x) = q(x \rightarrow x') \quad (694)$$

so the Hastings ratio (acceptance probability) becomes

$$r = \min\left(1, \frac{p(x')}{p(x)}\right) \quad (695)$$

Note: For a symmetric proposal operator we always accept the new state if it has higher probability than the current one. Moves to states with lower probability are still possible.

14.4.3 Example: Stochastic Sampling from a Gaussian

Aim: We want to sample from a Gaussian

$$p(x) \propto \exp\left(-\frac{x^2}{2}\right) \quad (696)$$

using the Metropolis algorithm. We do not need the normalization in the Metropolis algorithm.

14.4.3.1 Choosing a symmetric proposal function

Consider the symmetric proposal function

$$q(x \rightarrow x') : x' = x + \frac{2u - 1}{10}, \quad u \text{ drawn uniformly from } [0, 1) \quad (697)$$

14.4.3.2 Metropolis Hastings Algorithm for sampling from a Gaussian

1. start with some x with $p(x) > 0$
2. Draw u and calculate the proposal x'
3. Compute $r = \min\left(1, \frac{p(x')}{p(x)}\right) = \min\left(1, \frac{\exp\left(-\frac{x'^2}{2}\right)}{\exp\left(-\frac{x^2}{2}\right)}\right)$ (the normalization cancels out)
4. Draw another random number u' from $[0, 1)$
 - (a) accept x' as new state in the chain if $u' \leq r$
 - (b) otherwise add x again to the chain
5. Repeat until the chain is sufficiently long

Note:

1. the higher the rejection rate, the higher the repetition rate
2. subsequent entries are highly correlated \rightarrow the sequence is not random (ensemble sampling might be advantageous)
3. for $N \rightarrow \infty$ the total distribution of x_1, \dots, x_N will be perfectly Gaussian

The distribution of the whole sequence is illustrated in figure 121.

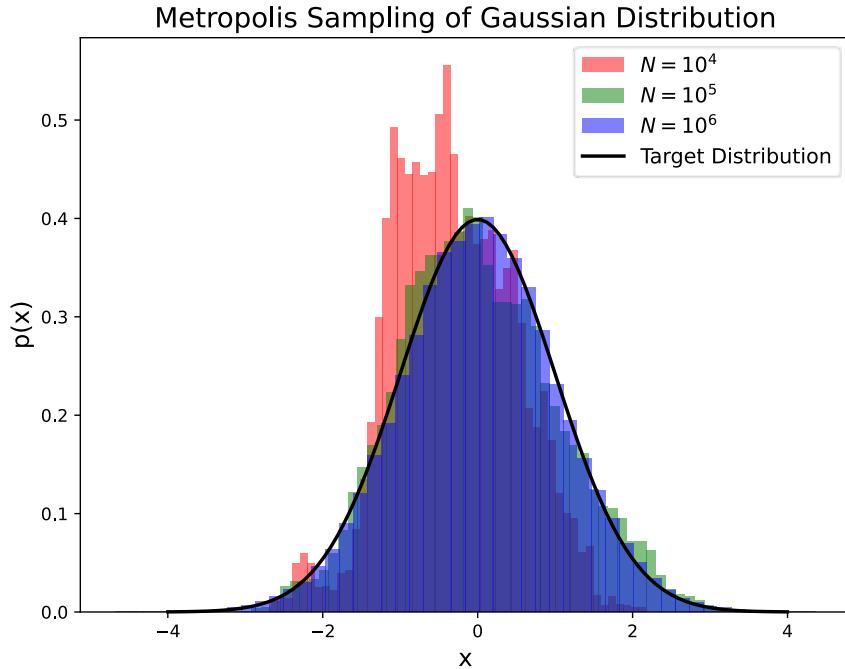


Figure 121: Marcov Chain Monte Carlo for sampling from a Gaussian; different chain lengths N .

14.5 Monte Carlo Simulations of Thermodynamic Systems

Consider a thermodynamic system at an externally given temperature T with a given Hamiltonian $H(\underline{\phi})$ (\rightarrow canonical ensemble), with

$$\text{configuration space coordinates } \underline{\phi} = \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_N \end{pmatrix} \quad (698)$$

usually very high dimensional.

Aim: We are interested in ensemble averages of physical quantities for instance the mean energy $\langle E \rangle$.

Remember that states in the canonical ensemble have relative probabilities given by the Boltzmann factor

$$\exp\left(-\frac{H(\underline{\phi})}{k_B T}\right) \quad (699)$$

(higher energy states are more unlikely at lower temperatures).

Thermodynamic ensemble averages consequently follow from

$$\langle A \rangle = \frac{1}{Z} \int A(\underline{\phi}) \exp\left(-\frac{H(\underline{\phi})}{k_B T}\right) [d\underline{\phi}]$$

canonical partition function $Z = \int \exp\left(-\frac{H(\underline{\phi})}{k_B T}\right) [d\underline{\phi}]$

$$[d\underline{\phi}] = d\underline{\phi}_1 d\underline{\phi}_2 \dots d\underline{\phi}_N$$
(700)

Nearly all thermodynamic quantities like the entropy or free energy can be derived from the partition function.

for instance the mean energy

$$\langle E \rangle = \frac{1}{Z} \int H(\underline{\phi}) \exp\left(-\frac{H(\underline{\phi})}{kT}\right) [d\underline{\phi}]$$
(701)

Problem: Directly calculating this ensemble average would require to calculate Z and this integral - both infeasibly high-dimensional.

14.5.1 Calculating the ensemble average $\langle A \rangle$ using importance sampling powered by a stochastic process

Idea: Imagine we would sample states $\underline{\phi}_i$ distributed according to the Boltzmann factor $\exp\left(-\frac{H(\underline{\phi})}{k_B T}\right)$ and approximate $\langle A \rangle$ by the average of the $A(\underline{\phi}_i)$. This is exactly Monte Carlo integration with importance sampling. And if we use MCMC we do not even need the normalization Z .

14.5.1.1 Sampling the states $\underline{\phi}_i$ using the Metropolis-Hastings algorithm

We want to sample $\underline{\phi}_i$ such that $p(\underline{\phi}) \propto \exp\left(-\frac{H(\underline{\phi})}{k_B T}\right)$.

For a symmetric proposal operator

$$q(\underline{\phi}' \rightarrow \underline{\phi}) = q(\underline{\phi} \rightarrow \underline{\phi}')$$
(702)

the Hastings ratio is

$$r = \min \left(1, \frac{p(\underline{\phi}')}{p(\underline{\phi})} \right) = \min \left(1, \exp \left(-\frac{H(\underline{\phi}') - H(\underline{\phi})}{kT} \right) \right)$$
(703)

Alternative direct specification of $W(\underline{\phi} \rightarrow \underline{\phi}')$ - Gibbs sampler: One can use

$$W_f(\underline{\phi} \rightarrow \underline{\phi}') = C \exp\left(-\frac{H(\underline{\phi}')}{kT}\right) \quad (704)$$

not dependent on the current state $\underline{\phi}$.

14.5.2 Example thermodynamic model: Ising Model

More details on the following can be found in Gould et al., 1996, chapter 17.

Consider a discrete spin model where on each lattice point \underline{x} the spin can either be up or down, $s_{\underline{x}} = \pm 1$ (*Ising Model*). A magnetic field \underline{B} and a temperature $T = \frac{1}{\beta}$ are externally given.

In this setting one obtains for the canonical partition function

$$Z = \sum_{\{s_{\underline{x}}\}} \exp \left[-\beta \left(\frac{1}{2} \underbrace{\sum_{\langle \underline{x}, \underline{y} \rangle} (1 - s_{\underline{x}} s_{\underline{y}})}_{\text{sum over pairs of nearest neighbors}} + B \sum_{\underline{x}} s_{\underline{x}} \right) \right] \quad (705)$$

14.5.2.1 Exemplary parameter of interest: Critical temperature under which spontaneous magnetization occurs

Consider the case without an external magnetic field, $B = 0$. In more than one dimension ($d > 1$) such a system will - under a critical temperature (the curie temperature) $T_c = \frac{1}{\beta_c}$ - show spontaneous magnetization (such as ferromagnetic materials such as iron and nickel do).

For temperatures $> T_c$ the magnetization vanishes, thus T_c separates the disordered phase with $T > T_c$ from the ordered one $T < T_c$.

Problem: While for $d = 2$ one can calculate $\beta_c = \log(1 + \sqrt{2})$, in higher dimensions we do not have such a formula.

Idea: Consider the mean magnetization

$$M = \frac{1}{V} \sum_i s_i \quad (706)$$

then it will show a characteristic temperature dependence, as shown in figure 122. So if we can numerically simulate the system at different temperatures, we can e.g. from the mean magnetization curve find T_c .

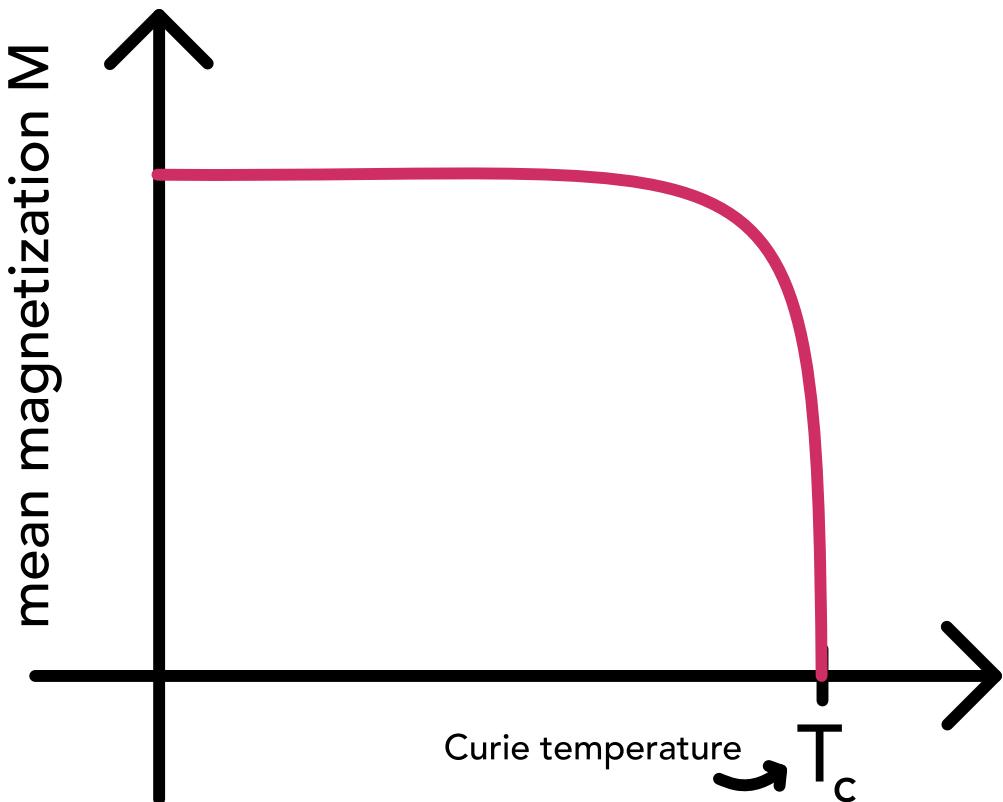


Figure 122: Temperature dependence of the mean magnetization for the infinite lattice 2D Ising model.

14.5.2.2 Metropolis algorithm for simulating the spin configuration on the lattice (and so the partition function and magnetization, ...)

1. Select a single lattice site \underline{x} randomly or in red-black ordering
2. Propose and update $s_{\underline{x}} \rightarrow s'_{\underline{x}}$ to the opposite spin direction
3. Compute the local energy functional $E_{\underline{x}}$ (all terms in the interaction energy involving the selected spin)

4. Accept the new state / reuse the old one based on the Hastings ratio

$$r = \min(1, \exp(-\beta \Delta E_{\underline{x}})), \quad \Delta E_{\underline{x}} = E_{\underline{x}}(s'_{\underline{x}}) - E_{\underline{x}}(s_{\underline{x}}) \quad (707)$$

Where the update is always accepted, if the change led to a lower energy (but also energy increasing updates can be accepted, its just less likely)

5. After a long MC chain has been generated, thermal equilibrium states at the prescribed temperature will eventually be reached

Alternatively a **heat-bath update** can be performed, where the new spin direction is based on

$$p(s_{\underline{x}}) = \frac{\exp(-\beta E_{\underline{x}}(s_{\underline{x}} = +1))}{\exp(-\beta E_{\underline{x}}(s_{\underline{x}} = +1)) + \exp(-\beta E_{\underline{x}}(s_{\underline{x}} = -1))} \quad (708)$$

where we draw a random number u uniformly from $[0, 1]$ and pick

$$s_{\underline{x}} = \begin{cases} 1 & \text{if } p(s_{\underline{x}}) > u \\ 0 & \text{else} \end{cases} \quad (709)$$

14.5.2.3 More thermodynamic properties that might be of interest*

$$\begin{aligned} \text{specific heat } C_V &= \frac{1}{V} \frac{\partial E}{\partial T} = \langle E^2 \rangle - \langle E \rangle^2 \\ \text{Magnetic susceptibility } \chi_M &= \frac{1}{V} \frac{\partial M}{\partial T} = \langle M^2 \rangle - \langle M \rangle^2 \end{aligned} \quad (710)$$

14.6 Monte Carlo Markov Chains in Parameter Estimation

Consider we have collected data $\{x_i\}_{i=1}^N$. We want to model this data using $f(\underline{x}|\underline{\mu})$ - a parametric model with parameters $\underline{\mu}$. The model could for instance be a Gaussian with the parameters being the mean and standard deviation.

How can we find parameters of a model from data?

The two principal ways of obtaining parameter estimates are maximizing the *likelihood* and maximizing the *posterior*.

14.6.1 Maximum Likelihood Estimation

The probability of our data given the model and its parameters (so assuming the model and parameters to be true) is called likelihood.

Under certain parameters $\underline{\mu}$ and a given sample $\underline{x} = \{x_i\}_{i=1}^N$ assuming the entries of the sample to be independent and identically distributed (i.i.d), we get the easy to calculate likelihood

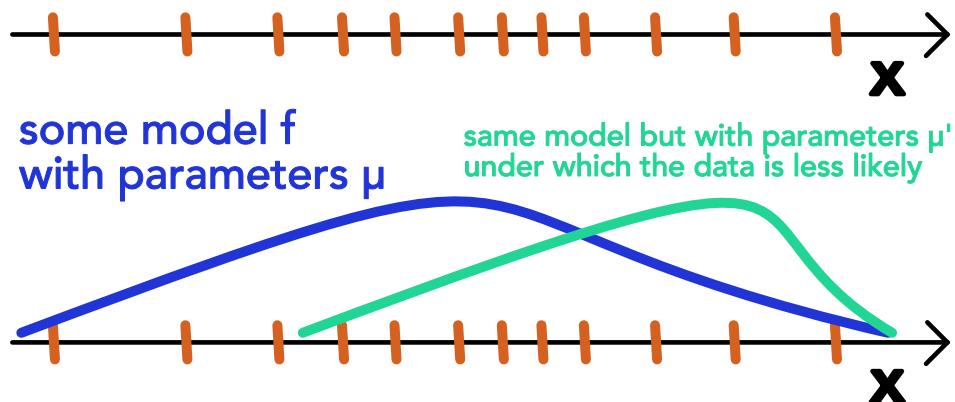
$$\mathcal{L}(\underline{x}|\underline{\mu}) = \mathcal{L}_{\underline{\mu}}(\{x_i\}_{i=1}^N) \underset{x_i \text{ i.i.d.}}{=} \prod_{i=1}^N f(x_i|\underline{\mu}) \quad (711)$$

and therefore the maximum likelihood estimation

$$\hat{\mu}_{\text{MLE}} = \underset{\mu}{\operatorname{argmax}} \mathcal{L}(\underline{x}|\underline{\mu}) \quad (712)$$

where one usually finds the parameters by minimizing the negative log-likelihood numerically, e.g. using gradient descent. See figure 123 for an illustration.

maximum likelihood estimation given data



MLE: find parameters for model under which the data is most likely

Figure 123: Maximum Likelihood Estimation

14.6.1.1 Estimating parameters based on the posterior

Consider we would know the probability distribution of the parameters $\underline{\mu}$ given the data \underline{x} , so $\Pi(\underline{\mu}|\underline{x})$.

Bayes theorem: We can calculate the conditional probability of an unobserved variable using Bayes theorem

$$\text{posterior} = \Pi(\underline{\mu}|\underline{x}) = \frac{\mathcal{L}(\underline{x}|\underline{\mu})g(\underline{\mu})}{h(\underline{x})} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}} \quad (713)$$

posterior \propto likelihood · prior

where Π, p, g, h denote probability distributions, possibly of different form (for conjugate priors, prior and posterior have the same form).

- $g(\underline{\mu})$ is the prior encoding knowledge already present about the parameters μ even before the data is taken. This can be constraints (e.g. not allowing negative values) of knowledge from prior experiments. If we know nothing, we choose a flat distribution as our prior. E.g. in classification where the question becomes *what is the class given the data*, the prior of a class $\underline{\mu}$ is usually taken as the abundance of this class in our classified data.
- $\mathcal{L}(\underline{x}|\underline{\mu})$ is the likelihood, the probability of obtaining the data given the parameters μ , easy to calculate
- $\Pi(\underline{\mu}|\underline{x})$ is the posterior, encoding the information we have gained about the parameters given our sample \underline{x}
- $h(\underline{x}) = \int p(\underline{x}, \underline{\mu}) d\underline{\mu} = \int \mathcal{L}(\underline{x}|\underline{\mu})g(\underline{\mu}) d\underline{\mu}$ is called model evidence giving the probability distribution function of the data

If we know $\Pi(\underline{\mu}|\underline{x})$ we can e.g. take the parameters with the largest posterior, as our estimate, $\hat{\underline{\mu}}_{MAP} = \operatorname{argmax}_{\underline{\mu}} \Pi(\underline{\mu}|\underline{x}) = \operatorname{argmax}_{\underline{\mu}} \mathcal{L}(\underline{x}|\underline{\mu})g(\underline{\mu})$ (the normalization (posterior) $h(\underline{x})$ is the same no matter μ , so it can be left out in the argmax), or the expectation value $\hat{\underline{\mu}}_{MMSE} = \int \underline{\mu} \Pi(\underline{\mu}|\underline{x}) d\underline{\mu}$.

Aim: We want to calculate integrals of the form

$$I = E_{\Pi}(u(\underline{\mu})) = \int u(\underline{\mu}) \Pi(\underline{\mu}|\underline{x}) d\underline{\mu} = \frac{1}{h(\underline{x})} \int u(\underline{\mu}) \mathcal{L}(\underline{x}|\underline{\mu})g(\underline{\mu}) d\underline{\mu} \quad (714)$$

which for the identity $u(\underline{\mu}) = \underline{\mu}$ is just the minimum mean squared error (MMSE) $\hat{\underline{\mu}}_{MMSE}$.

Problem: The integral for I as well as that for $h(\underline{x})$ will usually not be approachable analytically or with the other usual means.

Idea: When we can sample parameters $\{\underline{\mu}_i\}_{i=1}^N$ from the posterior without using $h(\underline{x})$, then we can approximately calculate expectations over the posterior distribution (like $\hat{\underline{\mu}}_{MMSE}$) simply by the Monte Carlo Estimator

$$\hat{I}_N = \sum_{i=1}^N u(\underline{\mu}_i) \quad (715)$$

14.6.2 Monte Carlo Markov Chain that samples the posterior as an equilibrium distribution

Problem: We want to calculate integrals involving the posterior $\Pi(\underline{\mu}|\underline{x})$ where the likelihood and prior are unproblematic but the evidence $h(\underline{x})$ is a complicated integral - just like the integrals we want to calculate.

So we seek a method to sample from the posterior $\Pi(\underline{\mu}|\underline{x})$ without knowing the evidence $h(\underline{x})$.

Idea: Via the Markov Monte Carlo chain we can sample from $\Pi(\underline{\mu}|\underline{x})$ without knowing the evidence $h(\underline{x})$ as it cancels out in the Hastings ratio.

Procedure

- Adopt a proposal function $q(\underline{\mu} \rightarrow \underline{\mu}') = q(\underline{\mu}'|\underline{\mu})$ only depending on the current point in the chain (not the previous history, MCMC is memoryless)
- Compute the acceptance probability (Hastings ratio)

$$r = \min \left(1, \frac{\Pi(\underline{\mu}' | \underline{x}) q(\underline{\mu} | \underline{\mu}')}{\Pi(\underline{\mu} | \underline{x}) q(\underline{\mu}' | \underline{\mu})} \right) = \min \left(1, \frac{\mathcal{L}(\underline{x} | \underline{\mu}') g(\underline{\mu}') q(\underline{\mu} | \underline{\mu}')}{\mathcal{L}(\underline{x} | \underline{\mu}) g(\underline{\mu}) q(\underline{\mu}' | \underline{\mu})} \right) \quad (716)$$

where the evidence $h(\underline{x})$ drops out. We keep the proposal $\underline{\mu}'$ with this probability and re-use the old point otherwise as before. [The normalizations of prior and likelihood also drop out.](#)

Based on our chain of $\underline{\mu}_i$ (or analogous ensemble sampling), we can calculate the expectation values over the posterior (eq. 715).

14.6.3 Caveat of MCMC | we never know when the chain is long enough

It is difficult to estimate, when the chain is long enough, that the results we can obtain from it (also for instance regarding the thermodynamic quantities in our Ising model) are good,

there are some heuristics on convergence though.

We cut off the burn-in time and as in a sequence of $\underline{\mu}$'s sequential one are not independent, we thin out, e.g. only retain every n-th $\underline{\mu}$ in the chain.

15 Parallelization Techniques

Consider we can decompose a problem into order-independent or partially-ordered components of computation (*concurrency*) then we can tap into the power of multiple cores by parallel execution (*parallelism*).

15.1 Why do we need to parallelize?

The most basic reason why we need parallelism is that we cannot arbitrarily scale up sequential operations, the speed of a single core cannot be scaled arbitrarily (very basically because logic gates have finite switching times and otherwise problems of metastable states occur (oscillation between states), because storage access has limited speed, because cooling becomes very difficult, ...) - **we need to simultaneously work on parts of a problem.**

Note: To use parallelization the algorithm of interest must parallelize well, and necessary communication between threads must be fast.

Further scenarios where parallelization is advantageous are

- In a big simulation, the data we want to process might not fit onto the memory of a single device.
- Different tasks might benefit from different hardware (e.g. complex instructions on the CPU, lots of simple instructions on the GPU).

15.2 Hardware perspective and parallelism

Computation is carried out on *cores* which on the CPU receive the data to process from a cache (where e.g. two cores might share a cache) which in turn comes from a (random access) memory. The smallest unit of storage transferred to a cache is a cache line, e.g. of **64 bytes** (a double precision float has **64 bits** so 8 bytes).

15.2.1 We should write code reducing memory access | row and column major storing convention for matrices

The two possibilities for sequentially storing a matrix are *row major* and *column major*, as illustrated in figure 124.

As we transport full cache lines from the main storage, we have to optimize our access according to the storage convention. E.g. in column major one would do a matrix vector multiplication by addition of the scaled column vectors.

Row-major order

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Red arrows indicate the row-major traversal path: $a_{11} \rightarrow a_{12} \rightarrow a_{13} \rightarrow a_{21} \rightarrow a_{22} \rightarrow a_{23} \rightarrow a_{31} \rightarrow a_{32} \rightarrow a_{33}$.

Column-major order

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Cyan arrows indicate the column-major traversal path: $a_{11} \rightarrow a_{21} \rightarrow a_{31} \rightarrow a_{12} \rightarrow a_{22} \rightarrow a_{32} \rightarrow a_{13} \rightarrow a_{23} \rightarrow a_{33}$.

Figure 124: Row and column major storing convention for matrices.

15.2.2 General computer architectures in increasing complexity*

Architectures from serial to clusters can be found in table 21.

Architecture	Characteristics
Serial computer	The central processing unit (CPU) executes a single stream (of load, compute, store operations), one operation at a time
Multi core nodes	Multiple cores are connected to a single memory and can execute operations simultaneously
Multi-socket computing nodes	Each multi-core CPU preferably accesses its memory bank but all cores can access the full memory (but not everything at the same speed)
Compute Clusters	Many computing nodes are connected via a communication network (Infiniband better than Ethernet)

Table 21: General computer architectures in increasing complexity.

15.2.3 CPU vs GPU

GPUs are specialized for executing many floating point operations in parallel. CPUs and GPUs are used complementarily to have the speed in floating point operations of the GPU and the complex instructions of the CPU.

CPUs and GPUs use Arithmetic Logical Units (ALUs) for the basic operations.

CPUs and GPUs are compared in table 22 and figure 125.

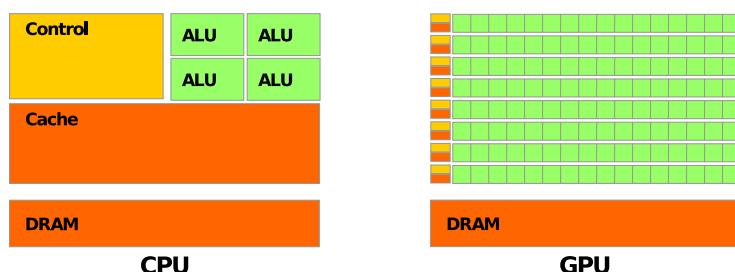


Figure 125: GPUs are build for parallel operations over many ALUs, CPUs for more complex operations over fewer ALUs.

CPU	GPU
Optimized for serial operations <ul style="list-style-type: none"> • few ALUs (4 to 16) • high clock speed (~ 3 GHz) 	Optimized for parallel operations <ul style="list-style-type: none"> • many ALUs (1000s) • lower clock speed (~ 1.5 to 2 GHz)
<ul style="list-style-type: none"> • each core runs its own code • complex control logic \rightarrow general purpose • shallow instruction pipelines²⁵ 	<ul style="list-style-type: none"> • many cores run the same code • simple control logic \rightarrow specialized for parallel operations • deep instruction pipelines
<ul style="list-style-type: none"> • low compute density (?) • large but slow memory • faster data transfer 	<ul style="list-style-type: none"> • high compute density • high computation per mean access • slower data transfer

Table 22: Comparison of CPUs and GPUs.

15.2.3.1 When and how to use GPUs

As data transfer to the GPU is slower

- copy data rarely / copy little data
- run the same code on all cores
- run many instructions

e.g. for

- N^2 force terms in direct summation
- (blocked) matrix multiplication
- ...

15.2.4 Vector cores

Many single compute cores can carry out vector instructions, i.e. apply one operation like addition to multiple elements (a vector).

15.2.5 Hyperthreading

At some point speed is more limited by the slow data transfer from memory to the cores than the speed of the cores. In hyperthreading, waiting time is efficiently used - we overload the computation core with execution streams and switch rapidly between these hyperthreads so

we can use waiting time efficiently. This way we effectively have more “virtual cores” than physical ones.

15.3 Types and challenges of concurrency and parallelism

15.3.1 Shared memory and message passing concurrency

The basic question is, how processes can be coordinated. Shared memory and message passing concurrency are compared in table 23.

Based on message passing concurrency we can have distributed memory parallelization, where our program can run on multiple computers.

Shared memory	Message passing
<ul style="list-style-type: none"> processes coordinate by reading and writing to shared memory locations locks are used to manage concurrent access to memory (or on a higher level a sophisticated type system as Rusts ownership system avoiding race conditions automatically) 	<ul style="list-style-type: none"> processes coordinate by sending and receiving messages through channels channel operations need to be managed process results can e.g. be brought together at barriers
<ul style="list-style-type: none"> OpenMP in C Threads in Rust 	<ul style="list-style-type: none"> MPI in C Go-routines combined with channels in Go, <i>sharing memory by communicating instead of communicating by sharing memory</i>

Table 23: Comparison of shared memory and message passing concurrency.

15.3.2 Challenges of concurrency

- **Race conditions in shared memory concurrency:** if different threads can modify the same variable at the same time, bugs can quickly arise or even memory corruption
- **Deadlocks:** being stuck forever acquiring a lock in shared memory concurrency (cyclic waiting) or waiting for a message that never arrives or never being able to send to a channel in message passing concurrency
- **Livelock:** repeating the same interaction without doing useful work

Note that for instance in the message passing model, by adequate types of channels (protocols specifying the usage of the channel (e.g. first send, then receive)) where for a *linearly typed channel* every capability must be used exactly once and only using dual pairs of channels (e.g. dual of the previous example: first receive then send) deadlock freedom can be guaranteed.

15.4 Shared memory parallelization (with OpenMP)

In shared memory parallelization, work is distributed to threads with access to the same memory.

Shared memory parallelization is limited to one node / CPU.

In C/C++/Fortran, OpenMP is a language / compiler extension for shared memory parallelization.

15.4.1 Simplest parallelization - parallel for loop of independent iterations

If order does not matter, we can simply parallelize a for loop as done in code 6.

```

1  #pragma omp parallel for
2  for (int i = 0; i < N; i++) {
3      some_expensive_computation(i);
4 }
```

Code-Snippet 6: Loop parallelization with OpenMP.

15.4.2 OpenMP's fork-join model

Sections of the code are run in parallel (starting from the main thread) and joined again. This is illustrated in figure 126.

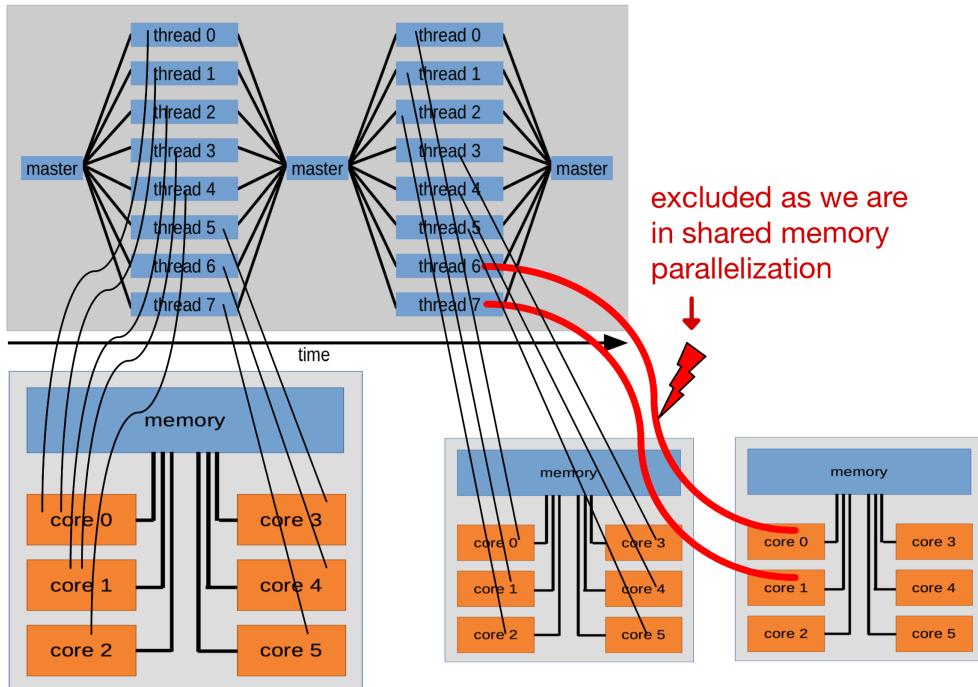


Figure 126: OpenMP's fork-join model.

15.4.3 Race conditions

If different threads can modify the same variable at the same time, bugs can quickly arise.

15.4.3.1 Race conditions in the context of non-atomic operations

An operation like `a = a + 1` is not atomic, but actually three operations:

- read the value of `a`
- add 1
- write the new value to `a`

In a setting where one thread might read before another thread has written, we get into trouble, as illustrated in go in code 7 and in C in code 8.

To avoid races, critical areas can be defined, that can only be accessed by one thread at a time, all others have to wait. This also adds a bottleneck to parallelization. A more fine-tuned approach is to lock memory, not code, where there can be multiple reads at the same time, but writes exclude all other operations.

```

1   func add_one(x *int) {
2       *x = *x + 1 // really three instructions
3           // 1. dereference pointer, 2. add 1, 3. assign to x
4   }
5
6   func main() {
7       x := 0
8       for i := 0; i < 1000; i++ {
9           go add_one(&x)
10      }
11      time.Sleep(2 * time.Second)
12      println(x) // e.g. 990
13  }

```

Code-Snippet 7: Parallel counter in Go. As a solution one could use a lock, or an addition worker with an input channel on which numbers are sent, and an output channel on which channels are sent on which the current value of the counter is sent back.

```

1   int count = 0;
2   // faulty addition
3   #pragma omp parallel for
4   for (int i = 0; i < 1000; i++) {
5       for (int j = 0; j < 1000; j++) {
6           count++;
7       }
8   }
9   // correct e.g. by defining a critical area, so
10  // ...
11  #pragma omp critical
12  {
13      count++;
14  }
15  // ...
16  // or by the annotation
17  #pragma omp parallel for reduction(:count)
18  // loop as before

```

Code-Snippet 8: Parallel counter in C.

15.4.4 Using the same variable where it is not intended

Problem: In code 9, j is used in all loops over i, so the inner loops interfere.

Solution: Give each thread its own j.

Another example in go is given in code 10.

```

1  #pragma omp parallel for private(j)
2  // without private(j) the inner loops interfere
3  for (int i = 0; i < N; i++) {
4      for (int j = 0; j < N; j++) {
5          // do something
6      }
7  }
```

Code-Snippet 9: Problem with using the same variable where it is not intended.

```

1  var wg sync.WaitGroup
2  wg.Add(5)
3  for i := 0; i < 5; i++ {
4      go func() {
5          println(i) // race on i!
6          // when goroutine reads i
7          // its already incremented
8          wg.Done()
9      }()
10 }
11 wg.Wait()
12 // output e.g.: 2 5 5 5 5
13 // solution, pass i as argument
```

Code-Snippet 10: Races can be devious.

15.5 Message passing concurrency enabling distributed memory parallelization (with MPI)

In message passing concurrency threads communicate by messages rather than by access to the same memory. In a message passing concurrency setting, different processes can be on different nodes and be executed individually with their own (part of) memory.

Example: All nodes might run the same code for a physical simulation but with different data (e.g. different parts of the grid) (single program, multiple data (SPMD)).

Note: OpenMP and MPI can be used together, e.g. OpenMP on each node and MPI for communication between nodes.

15.5.1 Architecture of an MPI program

The basic modes of communication in MPI are individual communication between two processes and collective communication. At barriers, results are brought together, the scheme is illustrated in figure 127.

A different approach is using channels onto which messages are sent and drawn from where the threads are shared between the channels (as in Go). We can for instance realize a worker pool where there are different worker threads drawing from a channel on which tasks are sent. Or pipelines ...

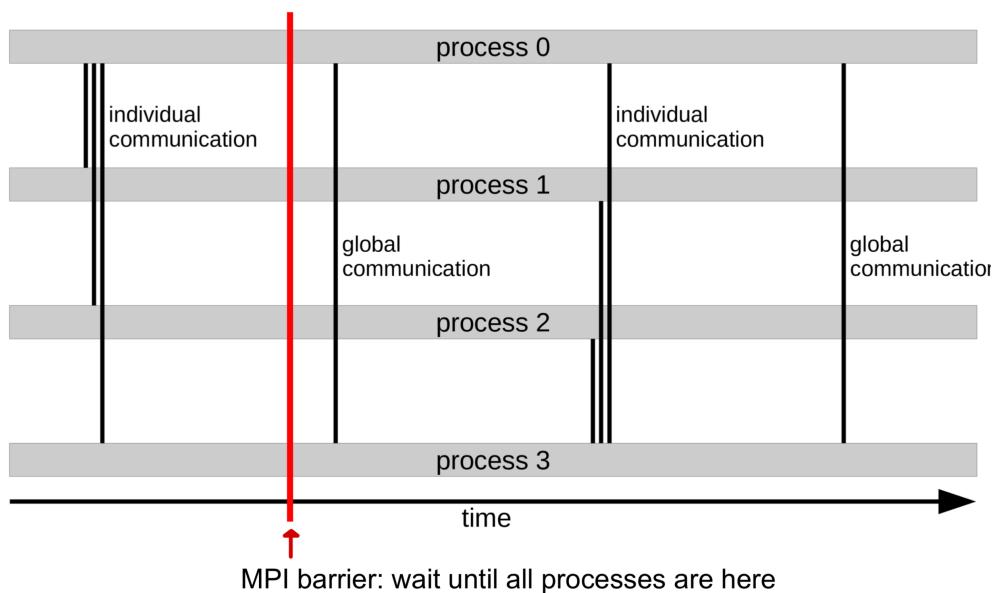


Figure 127: Architecture of an MPI program.

15.5.2 MPI program structure and basic communication

The basic idea is to start multiple instances of a program, where different instances can have different behavior based on their uniquely assigned rank. A typical MPI program is shown in code 11.

15.5.2.1 Point-to-point communication

The basic point-to-point communication is illustrated in code 12.

If we are not careful, we can easily introduce deadlocks where a process waits for Godot, i.e. a message that never arrives (typically cyclic waiting).

```

1   #include <mpi.h>
2   int main(int argc, char **argv) {
3       // initialize MPI
4       MPI_Init(&argc, &argv);
5
6       // we get the total number of processes
7       int size;
8       MPI_Comm_size(MPI_COMM_WORLD, &size);
9
10      // get the rank of the current process
11      int rank;
12      MPI_Comm_rank(MPI_COMM_WORLD, &rank);
13      // do something based on the rank
14      // ...
15
16      // finalize MPI
17      MPI_Finalize();
18  }
```

Code-Snippet 11: Basic structure of an MPI program.

```

1  // ...
2  if (rank == 0) {
3      int data = 42;
4      // MPI_Send(data, count, datatype, destination, tag,
5      //            communicator)
6      MPI_Send(&data, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
7  } else if (rank == 1) {
8      int data;
9      // MPI_Recv(data, count, datatype, source, tag, communicator,
10      //            status)
11      MPI_Recv(&data, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
12      //            MPI_STATUS_IGNORE);
13      printf("Received %d\n", data);
14  }
15  // ...
```

Code-Snippet 12: Basic point-to-point communication in MPI.

15.5.2.2 Collective communication

In the example of code 13, the process with rank 0 sends a message to all other processes.

```

1  // ...
2  if (rank == 0) {
3      int data = 42;
4      // MPI_Bcast(buffer, count, datatype, root, communicator)
5      MPI_Bcast(&data, 1, MPI_INT, 0, MPI_COMM_WORLD);
6  } else {
7      int data;
8      MPI_Bcast(&data, 1, MPI_INT, 0, MPI_COMM_WORLD);
9      printf("Received %d\n", data);
10 }
11 // ...

```

Code-Snippet 13: Basic collective communication in MPI.

15.5.2.3 More complex communication

More complex communication patterns are possible by barriers, and operations like gather and reduce.

In Go we can use contexts carrying deadlines, cancellation signals, and other request-scoped values across API boundaries and between processes. From an initial context, further contexts can be derived, e.g. when I send a complex math problem to a server it might split the problem and send out derived requests - a request tree is formed, where cancelling a context cancels all derived contexts, e.g. when I cancel the complex math task, all derived tasks also have to be cancelled.

15.5.3 Pinning - distribution of processes on cores and nodes

How do we assign on which core and node a process runs? - How do we pin processes?

Imagine we have four processes on four cores of one node, then they have to share the memory of the node. It can be advantageous to use less of the available cores of a node and more nodes if memory is a bottleneck.

15.5.4 Notes on reading and writing

While having lots of processes simultaneously read data is not a problem, **multiple parallel writing accesses** are problematic. Possible strategies are

- split the writing based on the domain the respective process is concerned with
- let one MPI process collect all data and write

An organized approach would be a pipeline, starting with data-reading and distribution to workers in the next step and collection and writing thereafter.

15.6 Parallel computing for physical simulations

How can we parallelize e.g. a hydrodynamics simulation?

15.6.1 Parallelized hydrodynamics

Idea: Split the domain between processes, each handling one spatial area.

For instance for simulating a magnetized plasma, based on the equations of magnetohydrodynamics

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \underline{v}) &= 0 \\ \frac{\partial \rho \underline{v}}{\partial t} + \nabla \cdot [\rho \underline{v} \underline{v}^T + \underbrace{\left(P_{\text{th}} + \frac{\|\underline{B}\|^2}{8\pi} \right) I - \frac{\underline{B} \underline{B}^T}{4\pi}}_{P_{\text{tot}}}] &= \rho \underline{g} \\ \frac{\partial e}{\partial t} + \nabla \cdot \left[\left(u + \frac{\rho \|\underline{v}\|^2}{2} + \frac{\|\underline{B}\|^2}{8\pi} + \frac{P_{\text{th}}}{\rho} \right) \underline{v} - \frac{\underline{B}(\underline{v} \cdot \underline{B})}{4\pi} \right] &= \rho \underline{v} \cdot \underline{g} \\ \frac{\partial \underline{B}}{\partial t} - \nabla \times (\underline{v} \times \underline{B}) &= 0 \end{aligned} \quad (717)$$

where we as a baseline use the discretizations on a grid (Eulerian perspective)

$$\begin{aligned} \partial_t y &\rightarrow \frac{y_i^{(n+1)} - y_i^{(n)}}{\Delta t} \\ \partial_x^2 y &\rightarrow \frac{y_{i+1}^{(n)} - 2y_i^{(n)} + y_{i-1}^{(n)}}{\Delta x^2} \end{aligned} \quad (718)$$

We can also split the domain of the Lagrangian SPH simulation, so each process handles a part of the particles.

Problem: In each time step, information from neighboring subdomains is necessary

- in the Eulerian case, the fluxes / for the three-point stencil of the second derivative approximation the values of the neighboring grid points at the edges
- for SPH the position information of all particles which have an effect on the density at the positions of the particles in the subdomain possibly all particles where the interpolation kernel can reach into the subdomain

Idea: We use guard cells (/guard space) (aka locally stored ghost cells) on which we copy the necessary information from the neighboring subdomains, see figure 128.

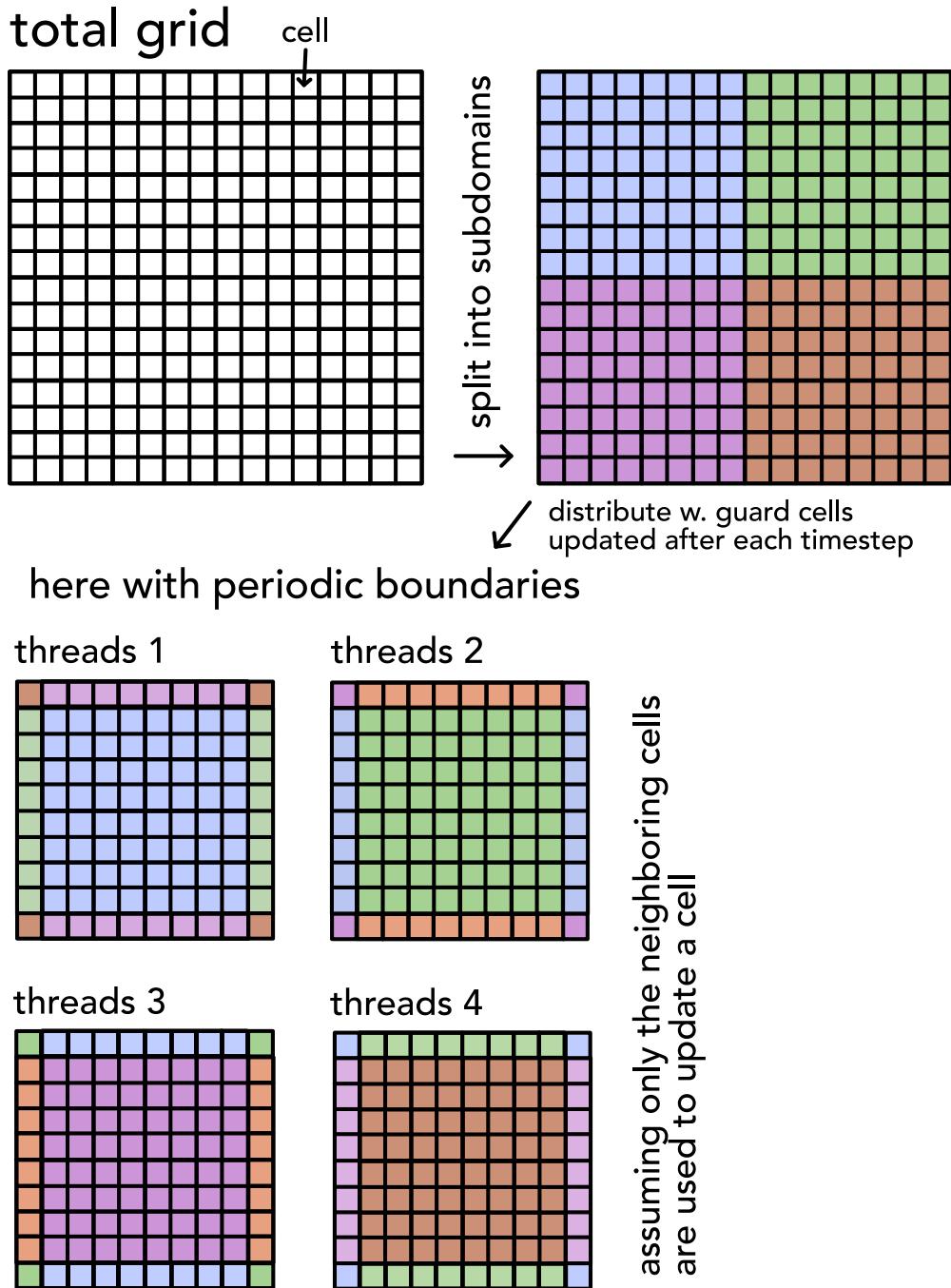


Figure 128: Guard cells for parallelized hydrodynamics.

Note: It is usually better for each process to have a local copy of the neighboring cells and not to communicate information as necessary as this has a higher communication overhead (higher number of communications). For the processes not to interfere all the time, we collect all neighboring values in each process at the start of each step, see table 24.

Communicate when necessary	Limit communication using locally stored ghost cells
<pre> 1 for time in times: 2 for x in xs: 3 for y in ys: 4 if at_boundary(x, y): 5 neighbors = 6 → communicate... 7 else: 8 neighbors = ... 9 density[x, y] = ... 10 momentum[x, y] = ... 11 energy[x, y] = ... </pre>	<pre> 1 for time in times: 2 get_ghost_cells() 3 for x in xs: 4 for y in ys: 5 density[x, y] = ... 6 momentum[x, y] = ... 7 energy[x, y] = ... 8 communicate... </pre>

Table 24: Different modes of communication in parallelized hydrodynamics.

15.6.1.1 Adaptive grid e.g. for the simulation of interstellar gas

Problem: Many physical systems are multi-scale with densities orders of magnitude apart and process on vastly different scales in space and time^a - we cannot afford to have a fine grid and small time steps everywhere.

^aE.g. in the interstellar medium cold dense gas forming clouds and stars and hot gas escaping galaxies.

Idea: Use an adaptive grid with a higher resolution where necessary, e.g. in denser regions, as illustrated in figure 129 and 130. As the system evolves, the grid is adapted (refined and derefined where necessary).

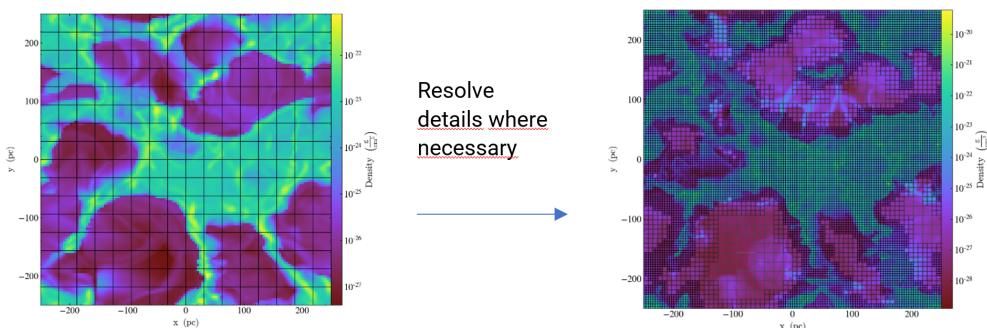


Figure 129: Grid refinement for interstellar medium.

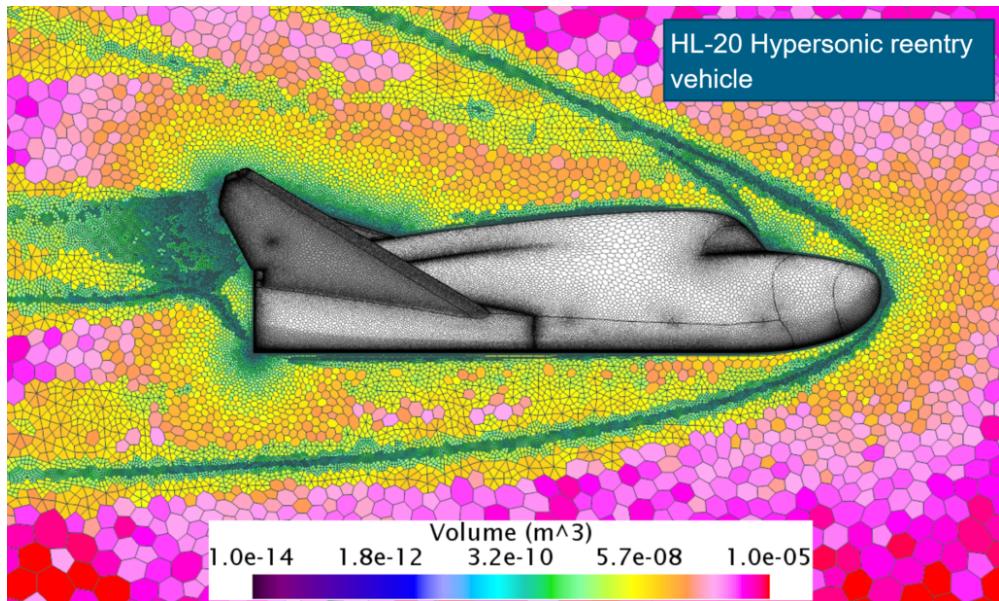


Figure 130: Adaptive mesh for the simulation of a supersonic jet.

How to distribute the simulation onto processes?: A simple approach is to give all processes the same number of cells with domains chosen to limit communication (/ minimize to overall borderlength) and cell-size chosen e.g. to have the roughly the same number of particles (equal memory distribution over cells). **Problems:**

- with increasing distribution increases the necessary communication, giving us a limit until which parallelization makes sense (beyond this further parallelization does not yield better performance)
- in the simple *memory balanced* approach, we neglect that different cells can have different costs (see physical load balancing below)

Local timesteps | physics load balancing: Small cells interact on smaller time scales so they require smaller timesteps (as before with the CFL criterion), so more iterations to cover the same time as for larger cells are necessary - so the workload for smaller cells is higher. **Decompose the domain according to work for more balanced tasks.**

15.6.1.2 Inclusion of long-range forces | long range forces vs. parallelization

In hydrodynamics interactions are short-range, information travels *one cell at a time*, a cell might be updated only based on its neighbors. Therefore, if we divide the domain into subdomains to distribute them onto processes, **a padding of only one (communicated) guard cell is sufficient for pure hydrodynamics** (for a *three-point-stencil*).

But gravity is an instantaneous long-range interaction:

- all cells are interdependent → all processes need to communicate (not only those handling neighboring regions)
- so practically every process while only updating its subdomain would need to have all particles in memory

Idea: While each process also needs distant information it must not be highly resolved - we can reduce communication by all processes working on a shared tree (possibly managed by one MPI rank / *arced RWLock in Rust*). For a distant group under small opening angle a single process must only retrieve the respective center-of-mass force, limiting communication.

So we can do (magneto)-hydrodynamics based on local information transfer and reduce the communication of (self)-gravitation via a shared tree structure.

15.7 Scaling of the processing time with increasing parallelism and Ahmdal's law

Concurrency is the partial-decomposability of a problem. The maximum speed-up gained from parallelization (the ratio of execution time with and without parallelization = t_1/t_N) is limited by the fraction of the problem, that is not decomposable.

Strong scaling Consider a fixed problem size. When using multiple processes the ideal speedup would be $\propto N_{proc}$ as $t_{N_{proc}} = t_1/N_{proc}$. The non-parallelizable part gives an upper limit for the speedup (**Ahmdal's law**)

$$\text{speedup} = \frac{1}{s + \frac{p}{N_{proc}}}, \quad \begin{matrix} \text{proportion of execution time of parallelizable part } p \\ \text{proportion of execution time of serial part } s \end{matrix} \quad (719)$$

The higher the non parallelizable part, the worse the scaling, see figure ???. Strong scaling is measured by how the computational time increases with the number of processes for a constant job size.

Weak scaling Here we consider the amount of work done for a scaled problem size. In Gustafson's law it is assumed that the parallel part scales linearly with the processes, while the serial part does not increase, speedup = $s + pN$. Weak scaling is tested by increasing job size and number of processes.

Note: In real application the overhead of parallelization increases with problem size, reducing the speedup.

15.8 Examples of parallel algorithms

- sorting numbers in parallel
 - even-odd-sort: while in (stupid) bubble sort we go through the list and consecutively swap, so $[72964] \rightarrow [27649]$ after one run (and we might need N of them), we can also make this independent by considering only consecutive pairs in each step, so $[72, 96, 4] \rightarrow [27, 69, 4]$ (can be done in parallel), then $[2, 76, 94] \rightarrow [2, 67, 49]$
 - parallel merge sort: sort chunks of the list in parallel, then merge
- parallel FFT: In a 3D FFT problem, we can split the domain into 2D slices and start by performing FFTs along their axis and after a transposed composition a final FFT along the missing axis, for details see Jagode, 2006.

15.9 Notes on scheduling

Problem: If tasks (a sequence of commands executed by a thread) need totally different computation times, the slowest task is the bottleneck for subsequent merging. For instance in a Milky Way SPH simulation where we would want to do time-steps on the order of 10^5 years, the bottleneck are cells with supernovae for which we would need to do multiple time-steps on the order of 10^3 years.

Task based scheduling: One approach is to estimate the cost of tasks and distribute work wisely accordingly. Another is work having a worker pool with work stealing. Work is distributed to workers but if one finishes early, it can steal work from another worker and not just wait, which might be more or less feasible depending on the problem.

Async: Asynchronous runtimes can allow us to write code similar as in a non-parallel setting while managing the distribution of work - we specify when we need what results (async / await).

Part III

Computational Statistics and Data Analysis

Simulation is concerned with computing the evolution of a system based on governing laws.

Statistics starts with data. It is concerned with finding information in data, like distributions of variables or relationships between them.

Two Cultures of Statistical Modeling: Consider independent variables \underline{x} and dependent variables \underline{y} . How nature associates one with the other (e.g. CO₂ with temperature) is generally a black box. The two cultures of statistical modeling (Breiman, 2001)^a can be seen as

- data-modeling culture: a simple interpretable stochastic model is assumed for the inside of the black box
- algorithmic modeling culture: more flexible models like decision trees or neural nets are used to predict \underline{y} from \underline{x}

An addition to this dichotomy is what we already discussed

- mechanistic modeling: derive a model from scientific theories

Of course those three cultures are extremes in model space, hybrids are possible.

^aLeo Breiman is probably best known for his work on Random Forests and Bagging. »A friend of mine, a prominent statistician from the Berkeley Statistics Department, visited me in Los Angeles in the late 1970s. After I described the decision tree method to him, his first question was, “What’s the model for the data?”«

16 Basic probability theory

Scientists measure things, for instance the (adult) length of a species of fish. Very natural questions might be

- How probable are certain (ranges) of lengths? - so how is the fish-length *distributed*
- What is the average length? And given we have measured 8 fish, how sure can we be that our average is a good proxy to the whole population average? So if I measured 8 other random fishes, how much would their average length differ? Things probably get better the more fish we measure ...

- How is the fish length related to other measurements we did? For instance the enrichment of nutrients in the water?

Let us start with basic definitions and distributions.

16.1 Basic definitions

A probability model is given by $\{\Omega, E, P\}$ with

- sample space Ω , e.g. all possible fish lengths
- events E , e.g. a length measured $> 5 \text{ cm}$
- probability P of an event

16.1.1 Sample space Ω

This is the set of fundamental outcomes in our experiment

- for rolling 1 dice: $\{1, \dots, 6\}$
- for rolling 2 dice: $\{(1, 1), (1, 2), \dots, (6, 6)\}$

16.1.2 Set of events E defined on the sample space

Examples for events could be

- for rolling 1 dice: $A_1 := \{2, 4, 6\}$ (even result), $A_2 := \{1, 3, 5\}$ (odd result)
- for our fish lengths: $A := \{l : l \geq 6 \text{ cm}\}$.

more precisely, E needs to be a σ -algebra that

- E contains the sample space Ω
- E is closed under complementation: $A \in E \rightarrow \bar{A} \in E$
- E is closed under countable unions: if $\{A_i\}_{i=1}^K \in E$ then $\bigcup_{i=1}^K A_i \in E$

The smallest set of events is $\{\emptyset, \Omega\}$, the smallest one containing a subset $A \subseteq \Omega$ is $\{\emptyset, \Omega, A, \bar{A}\}$.

16.1.3 Probability measure P

A probability measure maps events to a probability, so $P : E \rightarrow [0, 1]$ with

- $P(A) \geq 0 \forall A \in E$
- $P(\Omega) = 1$

- for any $A_i, A_j \in E$ such that $A_i \cap A_j = \emptyset$ (are disjoint)

$$P\left(\bigcup_{i=1}^K A_i\right) = \sum_{i=1}^K P(A_i) \quad (720)$$

(probabilities of disjoint sets just add)

16.1.3.1 Examples for the probabilities of events | discrete and continuous situations

- for 1 dice: $A_1 = \{2, 4, 6\}, A_2 = \{1, 3, 5\} \rightarrow P(A_1) = P(A_2) = \frac{1}{2}$
- for 2 dice (where order matters): $\Omega = \{(c_1, c_2) | c_1, c_2 \in \{1, \dots, 6\}\}, |\Omega| = 36$, e.g.
 - $- A = \{c_1 = c_2\} \rightarrow P(A) = \frac{6}{36} = \frac{1}{6}$
 - $- B = \{c_1 + c_2 \geq 10\} = \{(5, 6), (6, 5), (6, 6)\} \rightarrow P(B) = \frac{1}{12}$
- continuous situation: $x \in \mathbb{R}$, e.g.
 - $- A = x \leq b, P(A) = P(x \leq b) = \int_{-\infty}^b f(x) dx$ where f is the probability density function (PDF)

16.2 Basic probability laws

Let $A, B \in E$ with E a σ -algebra.

16.2.1 Conditional and joint probability

The probability of A when B has already occurred is called conditional probability $P_B(A) = P(A|B)$.

Naturally the intersection of two sets of events $A \cap B$ has the *joint probability* (»path-rule«)

$$P(A, B) := P(A \cap B) = P(B)P(A|B) = P(A)P(B|A) \quad \rightarrow \quad P(A|B) = \frac{P(A, B)}{P(B)} \quad (721)$$

so the general joint probability is

$$P(A_1, A_2, \dots, A_K) = P(A_1)P(A_2|A_1)P(A_3|A_1, A_2) \cdots \cdots P(A_K|A_1, \dots, A_{K-1}) \quad (722)$$

The path rule is illustrated in figure 131.

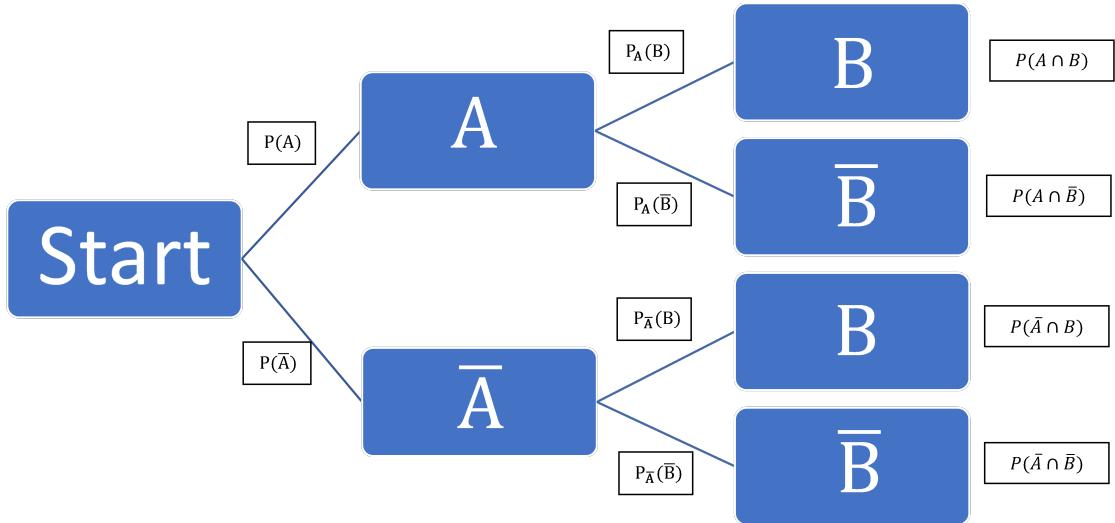


Figure 131: Path rule for joint probability

16.2.2 Bayes rule

From eq. 721 we can follow Bayes rule

$$P(H | E) = \frac{P(E | H) \cdot P(H)}{P(E)}, \quad P(E | H) = \frac{P(H | E) \cdot P(E)}{P(H)} \quad (723)$$

with

- **hypothesis H** : a hypothesis which probability may be affected by evidence E
- **prior probability $P(H)$** : the probability of the hypothesis before the evidence is observed
- **likelihood $P(E | H)$** : the probability of the evidence given that the hypothesis is true
- **probability of evidence $P(E)$** : (aka marginal likelihood) the probability of obtaining the evidence generally. When we are likely to obtain the evidence independent of the hypothesis, it does not confidently support any hypothesis
- **posterior probability $P(H | E)$** : the probability of the hypothesis after the evidence is observed

16.2.3 Probability of the union of events

Additive laws are shown in table 25.

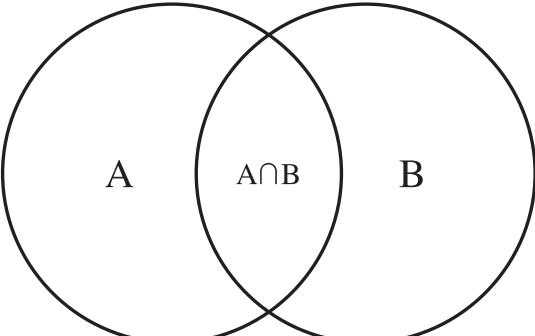
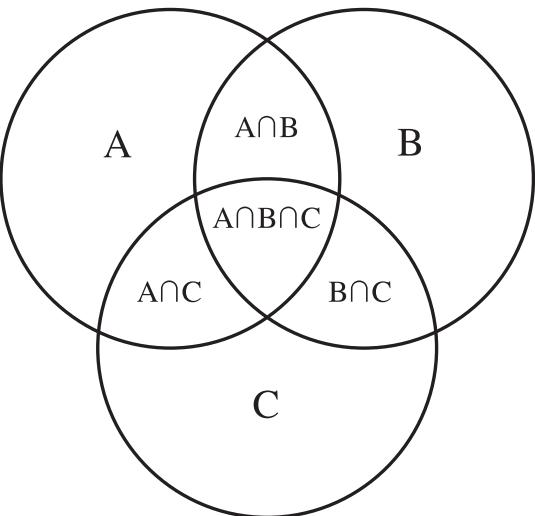
<p>Venn diagram</p> 	
$P(A \cup B) = P(A) + P(B) - P(A, B) \quad (724)$	$\begin{aligned} P(A \cup B \cup C) \\ = P(A) + P(B) + P(C) \\ - P(A, B) - P(A, C) - P(B, C) \\ + P(A, B, C) \end{aligned} \quad (725)$

Table 25: Additive laws

16.2.4 Independent events

Two events A and B are independent if and only if

$$P(A, B) = P(A)P(B) \quad (726)$$

so their joint probability is the product of the marginals, which follows if

$$P(A|B) = P(A) \quad \text{or} \quad P(B|A) = P(B) \quad (727)$$

so for the occurrence of event A it does not matter if B has already occurred and vice versa.

16.2.5 Example for throwing one dice (once)

For a dice $\Omega = \{1, \dots, 6\}$. Consider events $A = \{1, 3, 5\}$ and $B = \{2, 4, 6\}$ and $C = \{1, 2\}$.

- We can readily see that A excludes B and vice versa (fully dependent), $P(A) = P(B) \frac{1}{2}$ and $P(A|B) = 0$ so $P(A, B) = P(B)P(A|B) \neq P(A) \cdot P(B)$.
- A and C as well as B and C are independent - having obtained a 1 or 2 the probability

that it is a 1, 3 or 5 is $\frac{1}{2}$ just as for obtaining a 1, 3 or 5 in the first place.

16.2.6 Probability of the complement

$$P(\bar{A}) = 1 - P(A) \quad (728)$$

The complement can be very useful: Consider from integers up to N you draw N numbers (repetitions allowed), so

$$\Omega_0 := \{1, 2, \dots, N\}, \quad P(\omega_i \in \Omega_0) = \frac{1}{N} \quad (729)$$

Question: What is the probability $P(A)$ that at least two numbers are the same? - The probability that all numbers are different is the fraction of possible different draws $N!$ divided by all possible arrangements N^N , so $P(A) = 1 - \frac{N!}{N^N}$.

16.2.7 Law of total probability | marginalization

Given the joint $P(A_i \cap B) = P(A_i, B); A_i, B \subseteq \Omega$ how can we obtain $P(B)$?

If the A_i are a pairwise disjoint partition of Ω so

$$\bigcup_{i=1}^K A_i = \Omega, \quad \forall i \neq j : A_i \cap A_j = \emptyset \quad (730)$$

then we can marginalize out the A_i by

$$P(B) = \sum_{i=1}^K P(A_i) \cdot P(B | A_i) = \sum_{i=1}^K P(A_i, B) \quad (731)$$

Note: Therefore, we can write the probability of the evidence in Bayes law as

$$P(E) = \sum_{i=1}^K P(H_i, E) = \sum_{i=1}^K P(H_i) \cdot P(E | H_i) \quad (732)$$

-marginalized likelihood.

16.3 Random Variables (RVs)

Strictly, a random variable X is neither random nor a variable.

A random variable X is a mapping or function $X : \Omega \rightarrow F$ from possible outcomes (from a sample space, e.g. $\Omega = \{\text{Head, Tail}\}$ for a coin) to a measurable space F (e.g. $\{-1, 1\}$, 1 for heads, -1 for tails).

16.3.1 Examples for random variables

- for two dice with $\Omega = \{(c_1, c_2) | c_1, c_2 \in \{1, \dots, 6\}\}$, e.g. $X = c_1 + c_2$ is a random variable
- the length of a randomly measured fish
- for 8 randomly measured fishes, the mean length is also a random variable - **estimators themselves are random variables** and hence have a distribution, a mean, a variance, and so on with a particular realization of this random number called *estimate*
- ...

16.3.2 Continuous and discrete Random variables

A random variable X comes with its probability distribution.

Discrete random variables have a probability mass function (PMF) $P(X = X_i) = p(x_i)$ while continuous random variables have a probability density function $f(x)$ where formally any specific x has probability zero but ranges have probabilities, cumulative density (CDF) $P(X < x_0) = \int_{-\infty}^{x_0} f(x) dx$. PMF and PDF are normed.

16.3.3 Expectation values

The expectation value is the first moment of the probability mass or density function.

$$\begin{aligned} E[X] &\underset{X \text{ discrete}}{=} \sum_{i=1}^N P(X = x_i) x_i \\ &\underset{X \text{ continuous}}{=} \int_D f(x) x dx \end{aligned} \tag{733}$$

16.3.3.1 Properties of the expectation value

- expectation value of $c = \text{const.} \rightarrow E[c] = c$
- linearity of the expectation value: Let $\{x_i\}$ be a set of RVs and $\{c_i\}$ of constants, then

$$E \left[\sum c_i x_i \right] = \sum c_i E[x_i] \tag{734}$$

Proof: For two variables, this follows from

$$\begin{aligned} E[c_1x_1 + c_2x_2] &= \sum_{x_1} \sum_{x_2} (c_1x_1 + c_2x_2) P(x_1, x_2) \\ &= c_1 \sum_{x_1} x_1 \sum_{x_2} P(x_1, x_2) + c_2 \sum_{x_2} x_2 \sum_{x_1} P(x_1, x_2) \\ &\stackrel{\text{marginalization}}{=} c_1 E[x_1] + c_2 E[x_2] \end{aligned} \quad (735)$$

- Expectation over a function $g(x)$

$$E[g(x_1, \dots, x_N)] = \sum_{x_1} \dots \sum_{x_N} p(x_1, \dots, x_N) g(x_1, \dots, x_N) \quad (736)$$

- multiplicativity in case of independence: for X, Y independent

$$p(x_1, x_2) = p(x_1)p(x_2) \rightarrow E[g_1(x_1)g_2(x_2)] = E[g_1(x_1)]E[g_2(x_2)] \quad (737)$$

16.3.3.2 Conditional expectation

What is the expected value of a random variable X if $Y = y$ has already occurred?

$$E[X | Y = y] = \sum_x x P(X = x | Y = y) = \sum_x x \frac{P(X = x, Y = y)}{P(Y = y)} \quad (738)$$

so in the continuous case

$$E[X | Y = y] = \int_{-\infty}^{\infty} f(x|y) x dx \quad (739)$$

Note: Subscripts in expectation values are used ambiguously, $E_X[h(X, Y)]$ can mean the conditional expectation

$$E_X[h(X, Y = y)] = E[h(X, Y = y) | X] = \sum_y h(x, y) p(h(x, y) | x) \quad (740)$$

or a marginal density used for averaging

$$E_X[h(X, Y = y)] = \sum_x h(x, y) p_X(x) \quad (741)$$

16.3.3.3 Law of total expectation

Let us over the conditional expectation

$$E_{X|Y}[X | Y] \text{ is a random variable depending on } Y \quad (742)$$

take the expectation over Y

$$E_Y [E_{X|Y}[X | Y]] = E[X] \quad (743)$$

following from

$$\begin{aligned} E_Y [E_{X|Y}[X | Y]] &= \textcolor{teal}{E_Y} \left[\sum_x x P[X = x | Y] \right] \\ &= \sum_y \left[\sum_x x P[X = x | Y = y] \right] \cdot P(Y = y) \\ &\stackrel{P(A|B)=\frac{P(A,B)}{P(B)}}{=} \sum_y \left[\sum_x x P[X = x, Y = y] \right] \\ &= \sum_x x \sum_y P[X = x, Y = y] \\ &\stackrel{\substack{y \text{ is marginalized out}}}{=} \sum_x x P[X = x] = E[X] \end{aligned} \quad (744)$$

16.3.4 Variance, Covariance and Correlations

16.3.4.1 Variance

The variance is the expected squared deviation from the mean, so

$$\sigma^2 = \text{Var}(X) = E[(X - \mu)^2] = \sum_{i=0}^{\infty} p(X = x_i) (x_i - \mu)^2 = \text{Cov}(X, X), \quad \mu = E[X] \quad (745)$$

For calculating the variance the following relations

$$\text{Var}(X) = E[(X - E[X])^2] = E[X^2 - 2XE[X] + E[X]^2] \stackrel{E \text{ linear}}{=} E[X^2] - E[X]^2 \quad (746)$$

and

$$a, b = \text{ const } : \quad \text{Var}[ax + b] = a^2 \text{Var}[x] \quad (747)$$

might be useful.

16.3.4.2 Covariance and correlation

Covariance is a measure of joint variability (a linear relationship) between two random variables.

$$\text{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])] \quad (748)$$

so in the discrete case

$$\text{Cov}(X, Y) = \sum_x \sum_y (x - \mu_x)(y - \mu_y) p(x, y) \quad (749)$$

and in the continuous case

$$\text{Cov}(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \mu_x)(y - \mu_y) f(x, y) dx dy \quad (750)$$

Correlation coefficient To make covariances comparable it would be nice if they were bounded. The correlation coefficient

$$\text{corr}(X, Y) = \rho(X, Y) := \frac{\text{Cov}[X, Y]}{\sigma_X \sigma_Y} \in [-1, 1] \quad (751)$$

fulfills this, where the boundedness can be proven based on

$$\rho^2(X, Y) \underset{\substack{\text{task for} \\ \text{the reader}}}{<} 1 \quad (752)$$

Note: While independent variables are uncorrelated, the return must not be true, see figure 132.

Note: The expressions for the covariance and variance give us the formulas in case we have knowledge over the complete population - but if we only have a sample - e.g. our 8 fish - the formulas can be different. For instance using the population formula for the variance applied to a sample will underestimate the population variance this is the stray of the sample around the sample mean not around the population mean.

Properties of the covariance

- $\text{Cov}(X, Y) = E[XY] - E[X]E[Y]$
- Independent random variables are not correlated: For X, Y independent, so $E[XY] = E[X]E[Y] \rightarrow \text{Cov}(X, Y) = 0$. The reverse is not true, e.g. for $X \sim \mathcal{U}_{[-1,1]}$ and $Y = X^2$ we have $\text{Cov}(X, Y) = 0$ but X, Y are not independent.

The Correlation Coefficient Measures a Linear Relationship

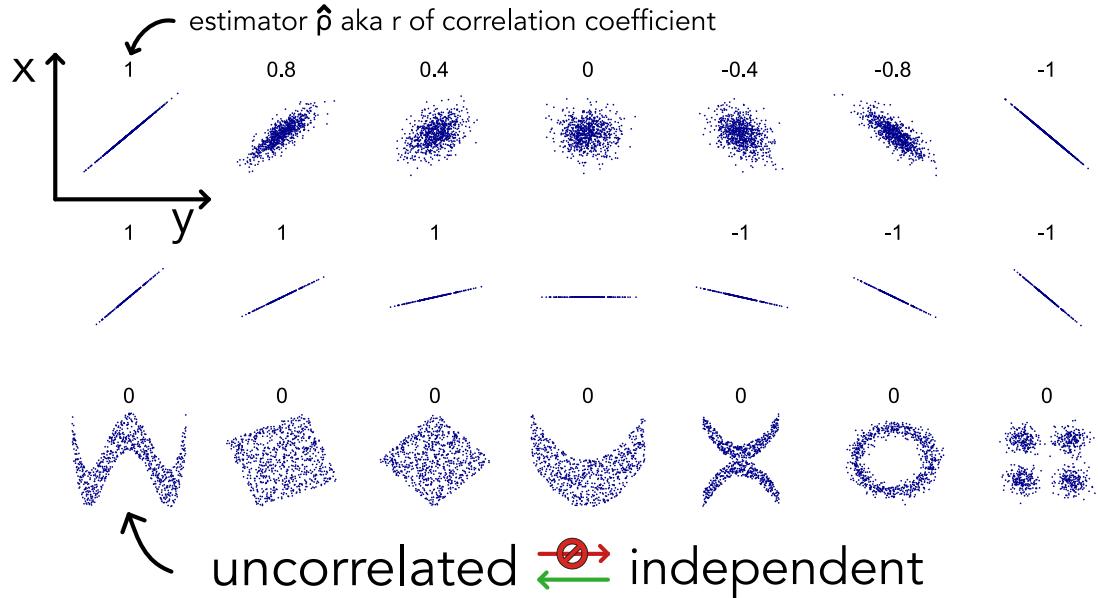


Figure 132: Correlation coefficient

- $\text{Var}(X+Y) = \text{Var}(X)+\text{Var}(Y)+2 \text{Cov}(X, Y)$, so for independent variables $\text{Var}(\sum X_i) = \sum \text{Var}(X_i)$
- $\text{Cov}(aX + b, cY + d) = ac \text{Cov}(X, Y)$ for constants a, b, c, d
- $\text{Cov}(X + Z, Y) = \text{Cov}(X, Y) + \text{Cov}(Z, Y)$
- $\text{Cov}(X, Y) = \text{Cov}(Y, X)$

Covariance-matrix for multivariate distributions Consider random variables $\{x_1, \dots, x_N\}$. The pairwise covariances can be collected into a covariance matrix

$$\sigma_{ij}^2 := \text{Cov}(x_i, x_j), \quad \text{often denoted as } \underline{\underline{\Sigma}} \in \mathbb{R}^{N \times N} \quad (753)$$

which is

- symmetric $\underline{\underline{\Sigma}} = \underline{\underline{\Sigma}}^T$ as of $\text{Cov}(x_i, x_j) = \text{Cov}(x_j, x_i)$
- has only positive real eigenvalues (\rightarrow positive semi-definite $\underline{x}^T \underline{\underline{\Sigma}} \underline{x} \geq 0$) as $\sigma_{ii}^2 = \text{Var}(x_i) \geq 0$

16.3.5 Sample mean and variance

Consider we collected a random sample²⁶ $\{x_i\}_{i=1}^N$ from a much larger population of size N_{pop} . Sample mean and variance are

$$\bar{x} := \frac{1}{N} \sum_{i=1}^N x_i, \quad s_x^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2$$

$$\lim_{N \rightarrow \infty} \bar{x} = E[x] =: \mu, \quad \lim_{N \rightarrow \infty} s_x^2 = \text{Var}(x)$$
(754)

weak law of large numbers: $\lim_{N \rightarrow \infty} P(|\bar{x} - \mu| > \epsilon) = 0$ but as noted before especially this variance might not be a good proxy for the population variance.

16.4 Univariate probability distributions

A probability distribution assigns probabilities to all outcomes of a random variable (or rather a probability density in the continuous case). E.g. $\frac{1}{6}$ for each side of a dice.

16.4.1 Discrete probability distributions

In the discrete setting, our experiment (e.g. dice-throw) has a countable number of outcomes, X is discrete and its distribution given by the probability mass function (PMF) $p(X = x) = p(x)$, with

$$(\text{positive}) \quad \forall x : p(x) \geq 0, \quad (\text{normed}) \quad \sum_{x_i} p(x_i) = 1$$
(755)

The **cumulative distribution function** (CDF) is defined as (assuming x to be ordinal)

$$F(x) = P(X \leq x) = \sum_{x_i \leq x} p(x_i)$$
(756)

16.4.1.1 Discrete uniform distribution

The same probability is assigned to all of the N possible outcomes.

$$X : \Omega \rightarrow \{1, \dots, N\}, \quad p(x = k) = \frac{1}{N}, \quad k = 1, \dots, N$$
(757)

²⁶Drawing such a sample has probability $\binom{1}{N_{\text{pop}}/N}$

16.4.1.2 Bernoulli distribution | probability distribution in a binary setting

A single draw has the probability distribution

$$P(X = x) = \text{Bern}(x; \mu) = \begin{cases} 1 - \mu & \text{if } x = 0 \\ \mu & \text{if } x = 1 \end{cases} = \mu^x(1 - \mu)^{1-x} \text{ parameter } \mu \quad (758)$$

The expectation value is

$$E[X] = 1 \cdot P(X = 1) + 0 \cdot P(X = 0) = 1 \cdot \mu + 0 \cdot (1 - \mu) = \mu \quad (759)$$

and the variance

$$\text{Var}[X] = E[X^2] - E[X]^2 = 1^2\mu - \mu^2 = \mu \cdot (1 - \mu) \quad (760)$$

16.4.1.3 Binomial distribution | probability of obtaining m heads in N coin tosses

Consider N identical and independent (i.i.d.) repititions of a Bernoulli experiment. The occurence of $x = 1$ in m of N draws has the probability

$$P(X = k) = \text{Bin}(X = m; N, \mu) = \binom{N}{m} \mu^m (1 - \mu)^{N-m} \text{ parameters } N, \mu, \quad X : \Omega \rightarrow \{0, \dots, N\} \text{ is the nu} \quad (761)$$

(plotted in 133) which is the product of the probability of a specific ordering with m heads ($\mu^m(1 - \mu)^{N-m}$, as i.i.d. Bernoulli experiments) and the possible orderings of m heads in N experiments given by the binomial coefficient

$$\# \text{ possible paths with } m \text{ heads in } N = \binom{N}{m} = \frac{N!}{m!(N-m)!} = \binom{N}{m-m} \quad (762)$$

The bernoulli distribution is a special case of the binomial distribution for $N = 1$.

Distributed according to notation: For a random variable to follow a distribution is indicated by \sim as in $X \sim \text{Bin}(N, \mu)$

Expectation value and variance of the binomial distribution As we are considering N i.i.d. Bernoulli experiments, the expectation value and variance of the binomial distribution are the same as for the Bernoulli distribution, just multiplied by N .

Binomial Distributions, probability of k successes in N trials
given probability of success μ and number of trials N

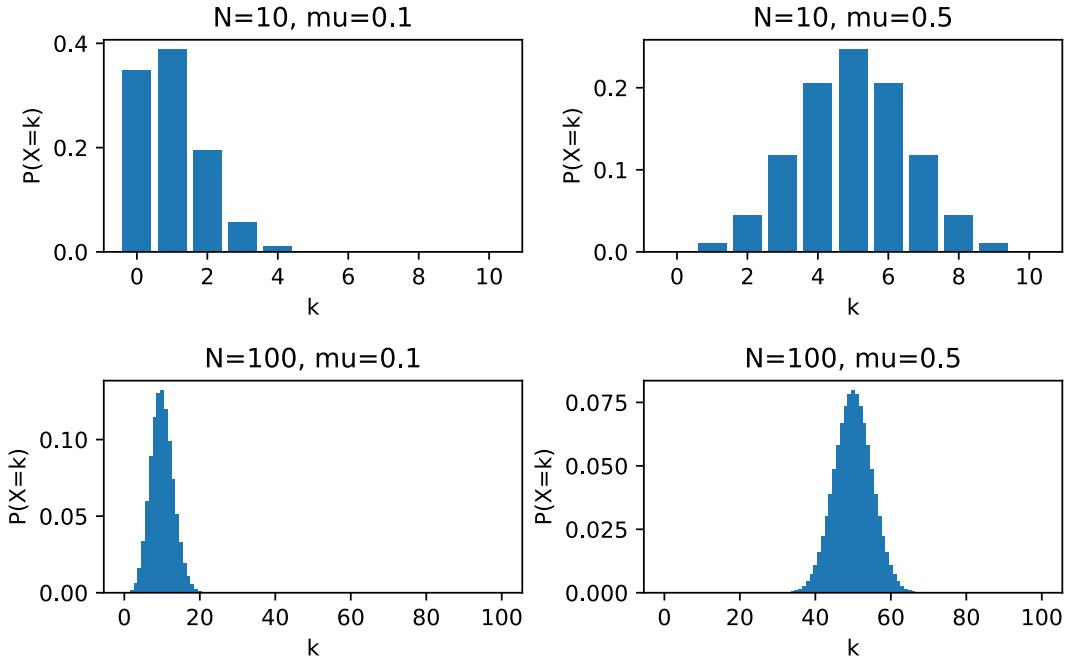


Figure 133: Binomial distribution

$$\begin{aligned}
 E[X] &\equiv \sum_{m=0}^N m \text{Bin}(X = m; N, \mu) = N\mu \\
 \text{Var}(X) &\equiv \sum_{m=0}^N (m - E[X])^2 \text{Bin}(m; N, \mu) = N\mu(1 - \mu)
 \end{aligned} \tag{763}$$

Cumulative distribution function of the binomial distribution Up to K successes in N trials have the probability

$$F(K) = p(X \leq K) = \sum_{m=0}^K \binom{N}{m} \mu^m (1 - \mu)^{N-m}, \quad F(N) = 1 \tag{764}$$

where the normalization of the binomial distribution ($F(N) = 1$) follows from the binomial theorem

$$(p + q)^N = \sum_{k=0}^N \binom{N}{k} p^k q^{N-k} \tag{765}$$

16.4.1.4 Geometric distribution | probability of the first success at trial k

Consider we are (i.i.d.) drawing from a Bernoulli distribution (e.g. coin toss) until the first success. The probability of $A_k = (\underbrace{0, \dots, 0}_{\times k-1}, 1)$ is

$$P(A_k) = P(X = k) = (1 - \mu)^{k-1} \mu \quad (766)$$

The probability mass function is plotted in figure 134.

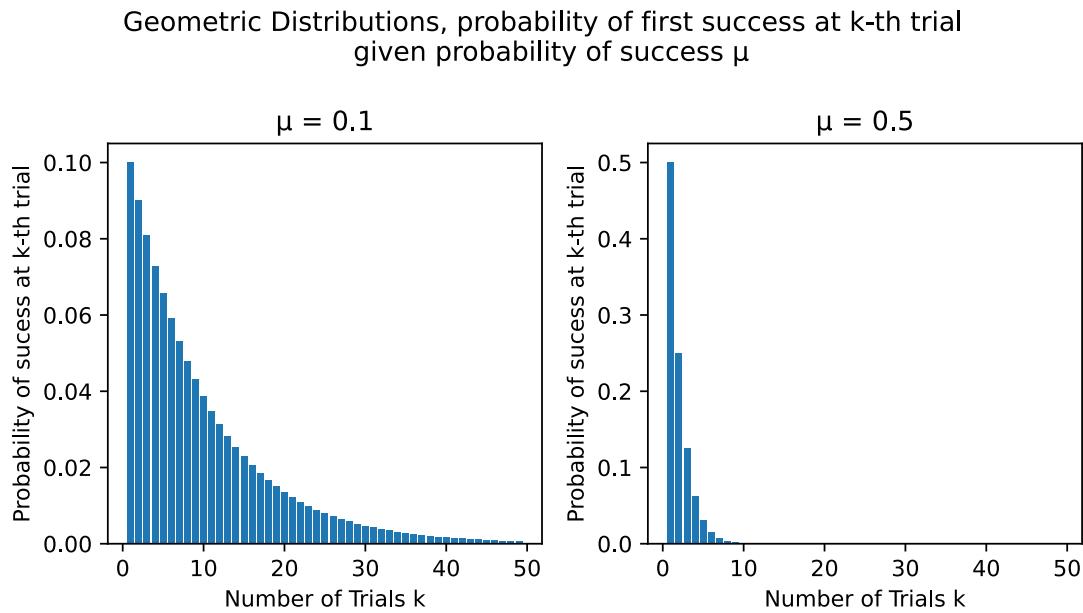


Figure 134: Geometric distribution

Expectation value and variance of the geometric distribution

$$E[X] = \frac{1}{\mu}, \quad \text{Var}[X] = \frac{1 - \mu}{\mu^2} \quad (767)$$

Cumulative distribution function of the geometric distribution The probability that the first success at latest occurs at trial k is the complement of the probability that no success occurs at all, so

$$F(k) = P(X \leq k) = 1 - (1 - \mu)^k \quad (768)$$

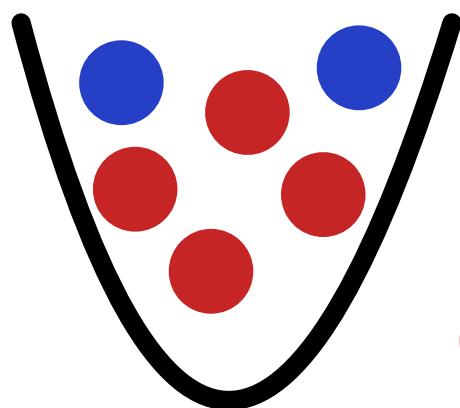
which also follows from

$$\begin{aligned}
 F(k) := P(X \leq k) &= \mu \sum_{i=1}^k (1-\mu)^{i-1} = \\
 &\quad \sum_{k=1}^n ar^{k-1} = \begin{cases} \text{geom. s.} & r \neq 1 : a \frac{1-r^n}{1-r} \\ & r = 1 : an \end{cases} \\
 &\mu \frac{1 - (1-\mu)^k}{1 - (1-\mu)} = 1 - (1-\mu)^k
 \end{aligned} \tag{769}$$

16.4.1.5 Hypergeometric distribution | k successes in r draws without replacement

For the binomial distribution the identical coin flip is repeated over and over again. But what if we are drawing from an urn with balls of two colors (binary) without replacement (see figure 135) where e. g. drawing one color very often really reduces the probability in further rounds.

Drawing from Urn without replacement



N items
1: ball is blue (n)
0: ball is red (N-n)

r balls are drawn, probability that k of them are blue: hypergeometric dist.

Figure 135: Drawing from an urn without replacement.

Consider we draw r balls in total, in the urn are n blue and $N - n$ red balls. The probability

of drawing k blue balls (with $k \leq r, n, N$) is

$$P(X = k) = \frac{\# \text{ paths where } k \text{ blue and } r - k \text{ red are drawn}}{\# \text{ paths where } r \text{ balls are drawn}} = \frac{\binom{n}{k} \binom{N-n}{r-k}}{\binom{N}{r}} \quad (770)$$

plotted in figure 136 (which is of course normalized, $\sum_{k=0}^r P(X = k) = 1$ (*task for the reader*)).

Hypergeometric Distributions, probability of k blue balls in r draws
given total number of balls in the urn $N = 20$, number of blue balls n , and number of draws r

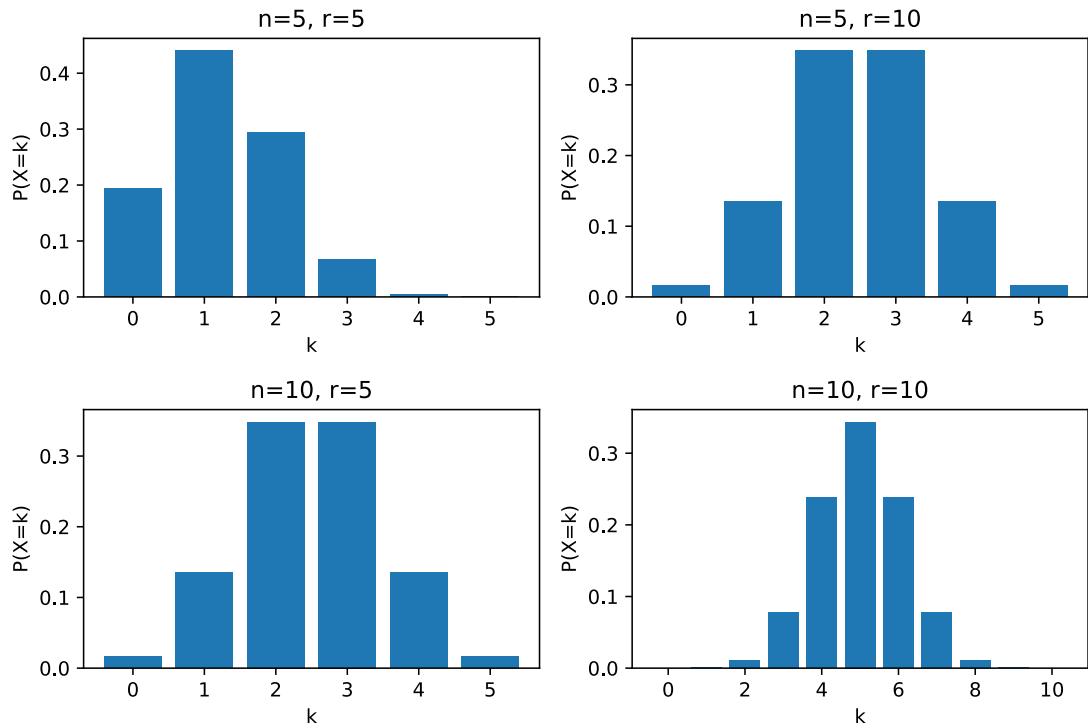


Figure 136: Hypergeometric distribution

Expectation value and variance of the hypergeometric distribution

$$E[X] = r \frac{n}{N}, \quad \text{Var}[X] = r \frac{n}{N} \frac{N-n}{N} \frac{N-r}{N-1} \quad (771)$$

Note: For large total number of balls N , large number of blue balls n and small number of balls drawn r the change in the number of blue balls as of previous draws is negligible so that we are back to the binomial case, which for large N is well approximated by the normal distribution, see figure 137.

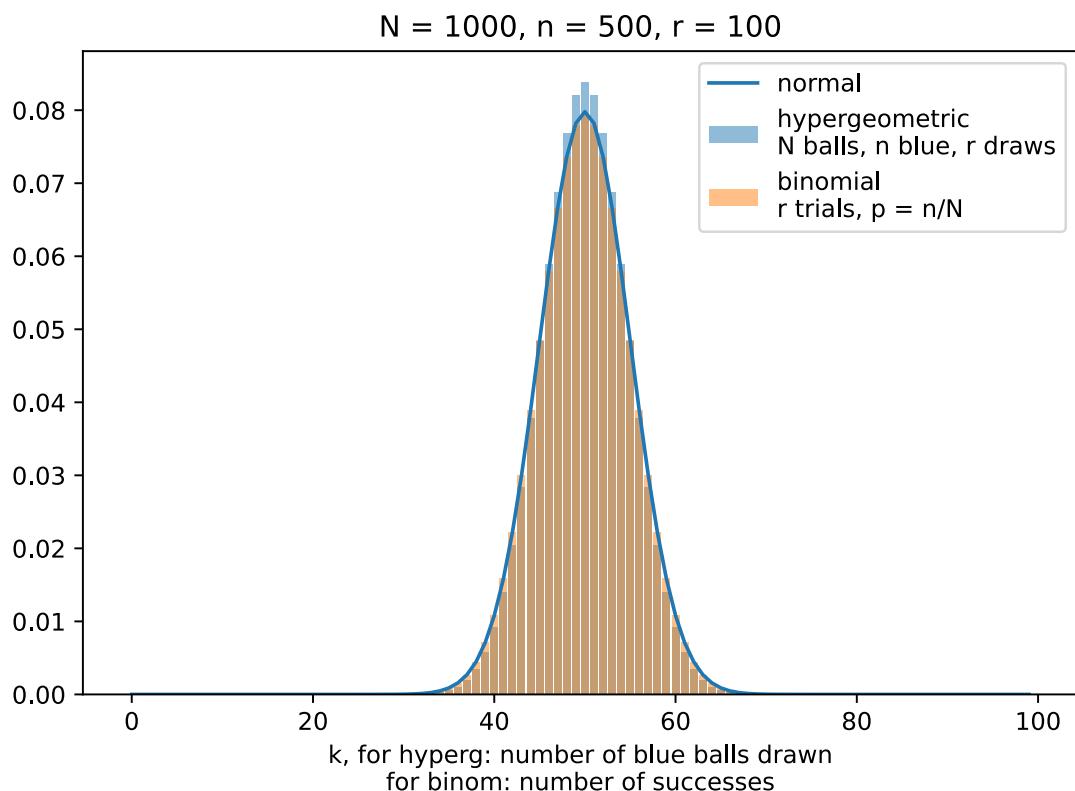


Figure 137: From hypergeometric to binomial to normal distribution.

16.4.1.6 Poisson distribution | probability of k events that have a mean rate occurring in a fixed interval

Consider events happening at a constant mean rate and independently of the time of the last occurrence. Let T be our time window and the probability of an occurrence (a success) in a small timestep Δt be μ . The expected number of occurrences in T is

$$\text{rate parameter } \lambda = N\mu = \text{const.}, \quad N = \frac{T}{\Delta t} \quad (772)$$

where we want our overall rate in T to be independent of Δt (and thus N). Naturally the probability of success μ in Δt has to therefore decrease as Δt goes down, so we set

$$\mu = \frac{\lambda}{N} \quad (773)$$

In the limit $\Delta t \rightarrow 0$ we yield the Poisson distribution from the Binomial distribution

$$P(X = k) = \text{Pois}(X = k; \lambda) = \frac{\lambda^k}{k!} \exp(-\lambda) \quad (774)$$

λ is the mean number of occurrences in T

So given the mean number of occurrences in a time interval λ what is the probability of any given number of occurrences?

Recapitulation - Binomial coefficient: Consider an urn with N different balls from which we draw k *without replacement and without order in our sample playing a role*. How many different such samples are there? Naturally in the first draw there are N possibilities, in the second $N - 1$ and so forth, but as we do not consider order we have to divide by $k!$, so naturally

$$\binom{N}{k} = \frac{\prod_{j=0}^{k-1} (N-j)}{k!} = \frac{N!}{k!(N-k)!} \quad (775)$$

Another possibility to directly see the RHS result, is to think of the possibilities to order N balls of which k are blue and $N - k$ are red, where order within the color-groups do not matter.

Obtaining the Poisson distribution as the limit of the Binomial one Using the above definition and that the *limit of a product is the product of the limits (if they exist)*, we get

$$\begin{aligned} & \lim_{\substack{N \rightarrow \infty \\ \mu \rightarrow 0}} \binom{N}{k} \mu^k (1-\mu)^{N-k} = \lim_{N \rightarrow \infty} \frac{\prod_{j=0}^{k-1} (N-j)}{k!} \left(\frac{\lambda}{N}\right)^k \left(1 - \left(\frac{\lambda}{N}\right)\right)^{N-k} \\ &= \frac{\lambda^k}{k!} \lim_{N \rightarrow \infty} \frac{\prod_{j=0}^{k-1} (N-j)}{N^k} \left(1 - \frac{\lambda}{N}\right)^{N-k} = \frac{\lambda^k}{k!} \lim_{N \rightarrow \infty} \underbrace{\prod_{j=0}^{k-1} \left(1 - \frac{j}{N}\right)}_{\rightarrow 1, N \rightarrow \infty} \underbrace{\left(1 - \frac{\lambda}{N}\right)^{-k}}_{\rightarrow 1, N \rightarrow \infty} \left(1 - \frac{\lambda}{N}\right)^N \\ &= \frac{\lambda^k}{k!} \lim_{N \rightarrow \infty} \left(1 - \frac{\lambda}{N}\right)^N = \frac{\lambda^k}{k!} \exp(-\lambda) \end{aligned} \quad (776)$$

where we used $\exp(x) = \lim_{n \rightarrow \infty} (1 + \frac{x}{n})^n$.

Expectation value and variance of the Poisson distribution One finds

$$E[X] = \lambda = \text{Var}[X] \quad (777)$$

for the expectation value following from

$$\begin{aligned}
 E[X] &= \sum_{k=0}^{\infty} k P(X = k) = \sum_{k=0}^{\infty} k \frac{\lambda^k}{k!} \exp(-\lambda) \\
 &= \sum_{k=1}^{\infty} k \frac{\lambda^k}{k!} \exp(-\lambda) = \lambda \sum_{k=1}^{\infty} \frac{\lambda^{k-1}}{(k-1)!} \exp(-\lambda) \\
 &= \lambda \underbrace{\sum_{k=0}^{\infty} \frac{\lambda^k}{k!} \exp(-\lambda)}_{=1 \text{ as Poisson normed}} = \lambda
 \end{aligned} \tag{778}$$

16.4.2 Continuous probability distributions

For a continuous random variable X the sample space has infinitely many outcomes. It makes no sense to give probabilities to single events, although a PDF is defined, but rather ranges of outcomes.

16.4.2.1 Cumulative distribution function

The CDF is

$$F(x) = p(X \leq x) = \int_{-\infty}^x f(x') dx' \tag{779}$$

where $f(x)$ is the probability density function (PDF) of X . (F is continuously differentiable except for finitely many points.)

Properties of the CDF

$$\begin{aligned}
 F(-\infty) &:= \lim_{x \rightarrow -\infty} F(x) = 0, & F(\infty) &:= \lim_{x \rightarrow \infty} F(x) = 1 \\
 \text{monotonic so } x_1 < x_2 \rightarrow F(x_1) &\leq F(x_2)
 \end{aligned} \tag{780}$$

Probabilities of intervals A range $x_0 \leq x \leq x_1$ has the probability

$$P(x_0 \leq X \leq x_1) = F(x_1) - F(x_0) = \int_{x_0}^{x_1} f(x) dx \geq 0 \text{ for } x_0 \leq x_1 \tag{781}$$

16.4.2.2 Probability density function

From the CDF, we can define the PDF as

$$f(x) = \frac{dF(x)}{dx} \tag{782}$$

with

$$f(x) \geq 0, \quad \int_{-\infty}^{\infty} f(x) dx = 1 \quad (783)$$

PDF and CDF for the normal distribution are plotted in figure 138.

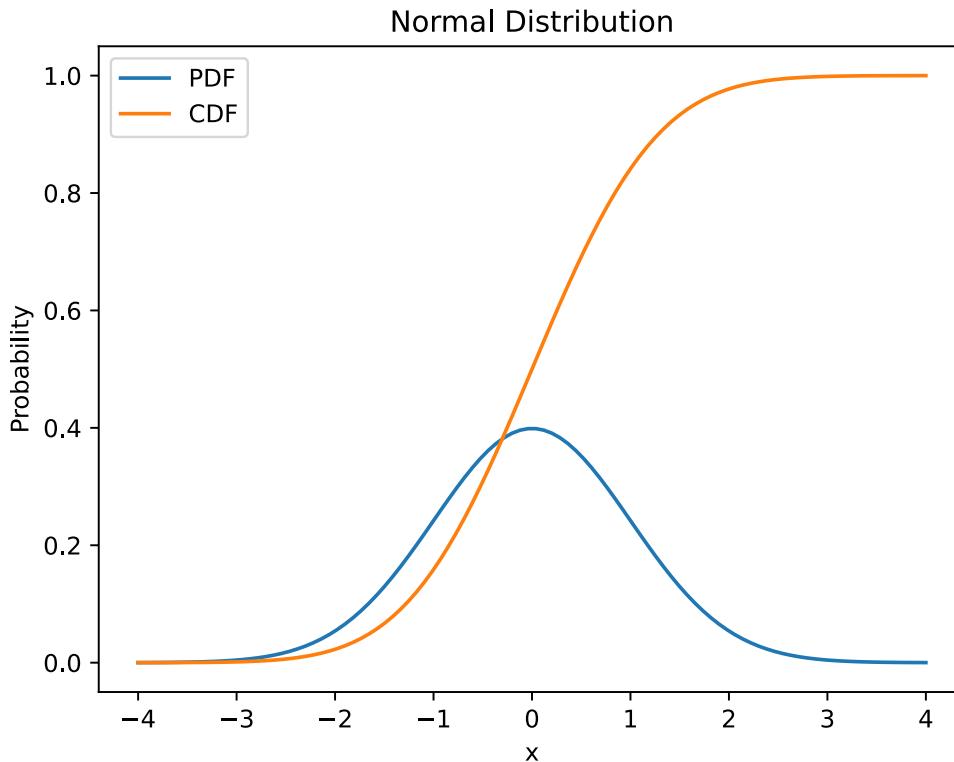


Figure 138: Normal distribution

16.4.2.3 Quantiles

A quantile is an x -value for which the CDF is equal to a given probability. For instance the 0.5-quantile is the median.

$$\phi_q : F(\phi_q) = P(X \leq \phi_q) = q \quad (784)$$

Quantile-quantile plot for testing whether an empirical distribution follows a theoretical one: Consider we are given measurements $[x_1, \dots, x_N]$. In the sorted list a given $x = \phi$ is the quantile ϕ_q^{emp} to the proportion q that it and all measurements to its left are to N . We can then evaluate the theoretical quantile $\phi_q^{\text{theo}} = F_{\text{theo}}^{-1}(q)$ at these proportions q and plot them against each other. If the empirical distribution follows the theoretical one the points should lie on the bisecting line ($\phi_q^{\text{emp}} = \phi_q^{\text{theo}}$). This is illustrated in figure 139.

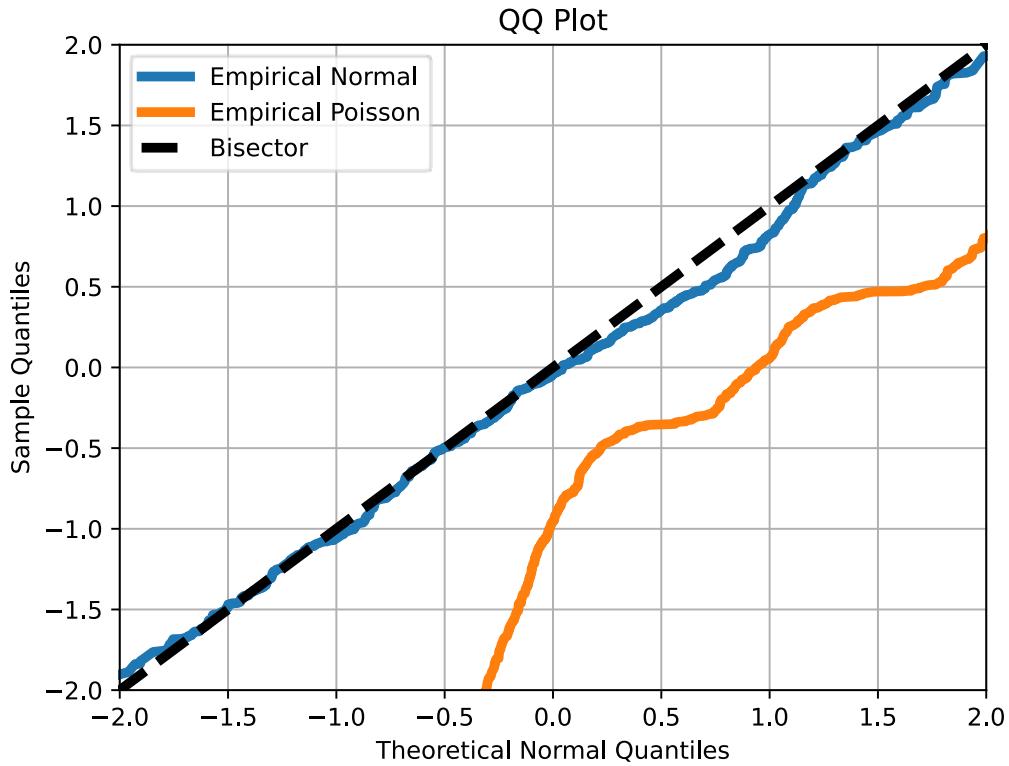


Figure 139: Quantile-quantile plot

16.4.2.4 Uniform distribution

The uniform distribution is given by

$$f(x) := \begin{cases} \frac{1}{\theta_2 - \theta_1} & \text{for } x \in [\theta_1, \theta_2], \theta_1 < \theta_2, \\ 0 & \text{else} \end{cases} \quad \text{parameters } \theta_1, \theta_2 \quad (785)$$

with

$$E[x] = \frac{\theta_1 + \theta_2}{2}, \quad \text{Var}(x) = \frac{(\theta_2 - \theta_1)^2}{12} \quad (786)$$

We have already discussed that sampling from any distribution numerically is based on sampling from the uniform distribution.

16.4.2.5 Gaussian (normal) distribution

The Gaussian in one dimensions is given by

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right), \quad \text{parameters } \sigma > 0, -\infty < \mu < \infty \quad (787)$$

$E[x] = \mu$ (center of distribution) , $\text{Var}(x) = \sigma^2$ (standard deviation) σ

The Gaussian is symmetric around μ which is its mean, mode (unique maximum) and median.

$$\mu = \underset{x}{\operatorname{argmax}} f(x) = \text{median } \phi_{\frac{1}{2}}, \text{ where } F\left(\phi_{\frac{1}{2}}\right) = \frac{1}{2} \quad (788)$$

Importance of the Gaussian: Many distributions converge do a Gaussian (like the Gaussian) and the distribution of the sum of many independent random variables converges to a Gaussian by the central limit theorem.

Proof that μ is the expectation value of the Gaussian By the smart reparametrization

$$z = \frac{x - \mu}{\sigma} \rightarrow x = \mu + \sigma z \rightarrow E[x] = \mu + \sigma E[z] \quad (789)$$

and

$$E[z] = \int_{-\infty}^{\infty} z \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right) dz \underset{\text{uneven function}}{=} 0 \rightarrow E[x] = \mu \quad (790)$$

Using the same *trick* by drawing z from a Normal distribution with $\mu = 0$ and $\sigma = 1$ we can draw from any Gaussian distribution ($x = \mu + \sigma z$).

16.4.2.6 Beta distribution

Motivation for conjugate priors: Consider we have some success-failure data (k successes in N trials) which we assume to be Bernoulli distributed. We can reasonably estimate the success- probability (the parameter of the Bernoulli distribution) μ as $\hat{m}\mu = \frac{k}{N}$. When we do multiple such experiments we will get different estimates for $\hat{\mu}$. What distribution could describe the distribution of μ (so how likely obtaining a certain μ is)? Even better, what distribution could that be so that with subsequent experiments, by a Bayesian update

$$\text{posterior probability } p(\mu | \text{evidence}) \propto \text{likelihood } p(\text{evidence} | \mu) \times \text{prior } p(\mu) \quad (791)$$

the prior and posterior have the same functional form? So a Bayesian update is an update to the parameters of that distribution?

The Beta distribution is given by

$$\begin{aligned} f(x) &= \text{Beta}(x; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1}, \quad \Gamma(\alpha) := \int_0^\infty x^{\alpha-1} \exp(-x) dx \\ &\Gamma(z+1) = z\Gamma(z), \quad \Gamma(n) = (n-1)!, \quad n \in \mathbb{N}, \quad \Gamma(1) = 1 \\ &\alpha, \beta > 0, \quad x \in [0, 1] \end{aligned} \quad (792)$$

and is **the conjugate prior of the**

- Bernoulli distribution
- Binomial distribution
- Negative binomial distribution
- Geometric distribution

Different beta distributions are plotted in figure 140.

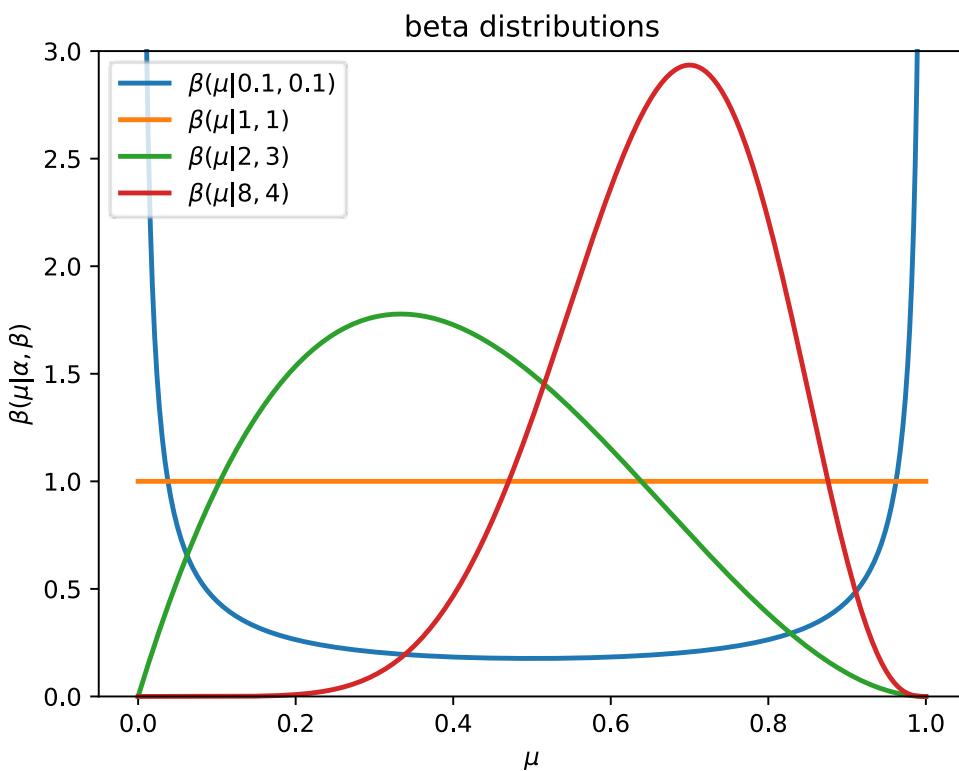


Figure 140: Beta distribution

Expectation value and variance of the Beta distribution

$$E[x] = \frac{\alpha}{\alpha + \beta}, \quad \text{Var}[x] = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)} \quad (793)$$

16.4.2.7 Gamma distribution

The Gamma distribution is a family²⁷ of distributions for non-negative real random variables. It is often used to model waiting times.

²⁷In the sense that other distributions, here e.g. the exponential distribution are special cases of it.

$$f(x) = \Gamma(x; \alpha, \beta) = \begin{cases} \frac{x^{\alpha-1} \exp(-\frac{x}{\beta})}{\beta^\alpha \Gamma(\alpha)}, & 0 \leq x < \infty \\ 0, & \text{else} \end{cases} \quad (794)$$

shape parameter α , rate parameter β

For instance in a sequence of events with each having a waiting time described by an exponential distribution with rate β the waiting time for the n -th event follows $\Gamma(x; n, \beta)$.

Different gamma distributions are plotted in figure 141.

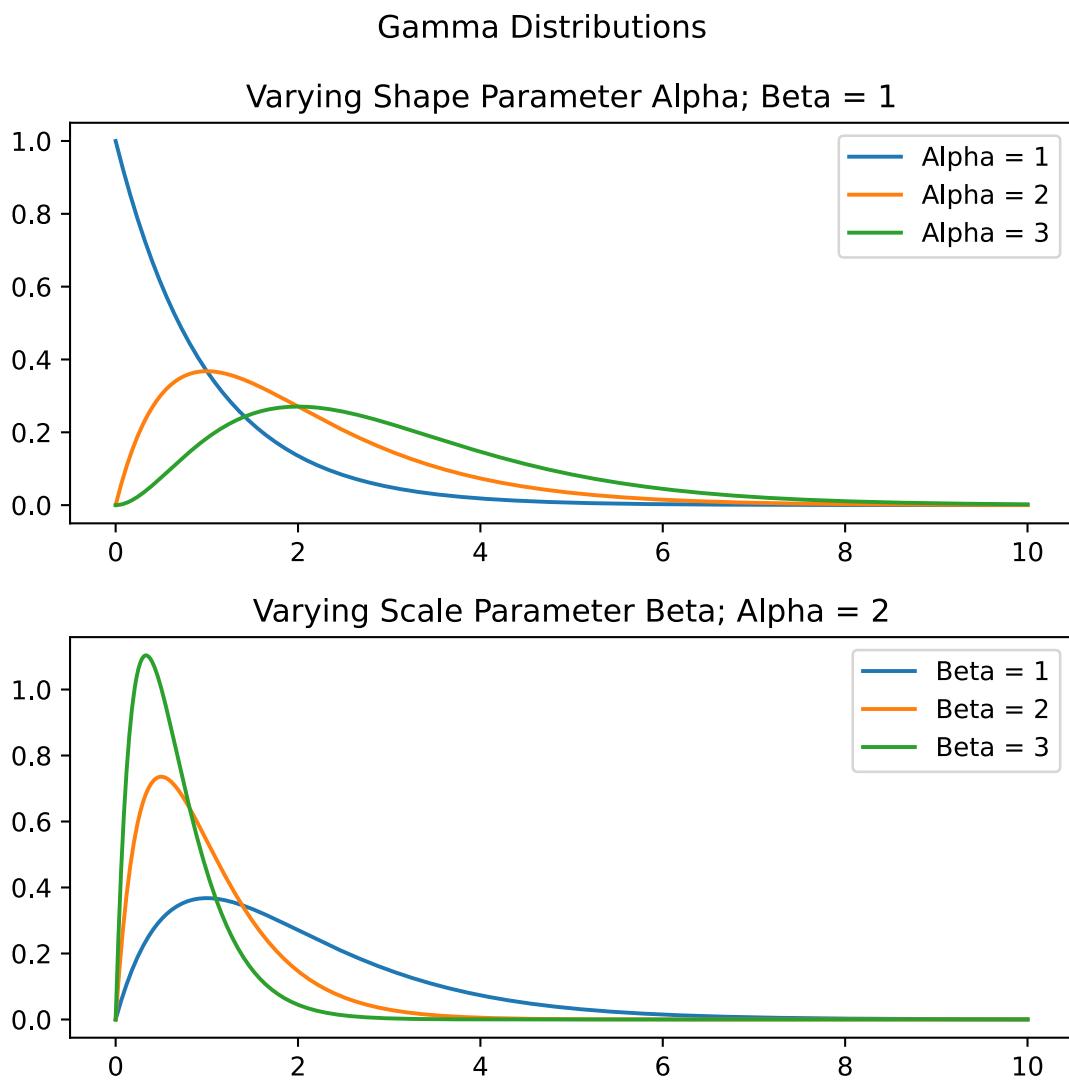


Figure 141: Gamma distribution

Expectation value and variance of the Gamma distribution

$$E[x] = \alpha\beta, \quad \text{Var}[x] = \alpha\beta^2 \quad (795)$$

where the result for the expectation value follows from

$$E[x] = \int_0^\infty x \frac{x^{\alpha-1} \exp\left(-\frac{x}{\beta}\right)}{\beta^\alpha \Gamma(\alpha)} dx \underset{\Gamma(z+1)=z\Gamma(z)}{=} \alpha \beta \int_0^\infty \underbrace{\frac{x^\alpha \exp\left(-\frac{x}{\beta}\right)}{\beta^{\alpha+1} \Gamma(\alpha+1)}}_{=\Gamma(x;\alpha+1,\beta)} dx \underset{\text{Gamma normed}}{=} \alpha \beta \quad (796)$$

16.4.2.8 Exponential distribution

For $\alpha = 1$, the Gamma distribution becomes the exponential distribution

$$f(x) = \begin{cases} \frac{1}{\beta} \exp\left(-\frac{x}{\beta}\right), & 0 \leq x < \infty \\ 0, & \text{else} \end{cases}, \quad \text{so } E[x] = \beta = \sqrt{\text{Var}(x)} \quad (797)$$

Usage - time until: The time until radioactive decay is described by the exponential distribution. In the discrete case of Bernoulli trials, the geometric distribution describes the number of trials until the first success. The exponential distribution is the continuous analog of the geometric distribution with the base process being a Poisson process.

Memorylessness That we already waited t_0 for an event, does not make its soon occurrence more likely (just as throwing ten heads in a row will not make tail more likely in the next throw (geometric distribution is also memoryless)). The probability of after waiting t_0 to then furthermore wait t_1 is the same as waiting t_1 from the start, so

$$P(X > (t_0 + t_1) \mid X > t_0) = P(X > t_1) \quad (798)$$

as

$$\begin{aligned} P(X > t_1) &= \int_{t_1}^\infty \frac{1}{\beta} \exp\left(-\frac{x}{\beta}\right) dx = e^{-\frac{t_1}{\beta}} \\ P(X > (t_0 + t_1) \mid X > t_0) &= \frac{P(X > (t_0 + t_1))}{P(X > t_0)} = \frac{e^{-\frac{t_0+t_1}{\beta}}}{e^{-\frac{t_0}{\beta}}} = e^{-\frac{t_1}{\beta}} \end{aligned} \quad (799)$$

16.4.2.9 Exponential family distributions and conjugate priors

We will now introduce a broad family of distributions which the previously introduced distributions are special cases of.

Based on the exponential family, we can derive very general results - e.g. as we can derive general expressions of the moments of distributions following the exponential family. In generalized linear models, for instance, the distribution of which we model the mean is from the exponential family.

For a random variable x and so-called natural parameters $\underline{\eta}$ the exponential family is (in its canonical form) given by

$$\begin{aligned} p(x \mid \underline{\eta}) &= \exp(\underline{\eta}^T \underline{T}(x) - h(\underline{\eta}) + g(x)) = \tilde{h}(\underline{\eta}) \tilde{g}(x) \exp(\underline{\eta}^T \underline{T}(x)) \\ h(\underline{\eta}) &= -\log(\tilde{h}(\underline{\eta})), \quad g(x) = \log(\tilde{g}(x)) \end{aligned} \quad (800)$$

link function $h(\underline{\eta})$, sufficient statistic $\underline{T}(x)$

What is a sufficient statistic? The statistic $\underline{T}(x)$ is sufficient for underlying parameter(s) $\underline{\eta}$ if it contains all the information in the data about $\underline{\eta}$, so the conditional probability of the data given the statistic $\underline{T}(x)$ does not depend on $\underline{\eta}$. *Example:* Sample mean $T(X) = \frac{1}{N} \sum_i x_i$ for a Gaussian distribution with known variance.

Advantages of the exponential family:

- distributions from the exponential family have conjugate priors
- the moments of the distributions follow from a general formula
- very flexible with small model size: the size of \underline{T} is fixed in the sense that it does not grow with the number of data points (as e.g. in a KNN model) but still arbitrary amounts of i.i.d. data can be modeled

Conjugate priors For any likelihood from the exponential family $p(x \mid \underline{\eta})$ there is a conjugate prior (often also from the exponential family) $p(\underline{\eta})$ so that the posterior as given by Bayes rule

$$p(\underline{\eta} \mid x) = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}} = \frac{p(x \mid \underline{\eta}) p(\underline{\eta})}{p(x)} = \frac{p(x \mid \underline{\eta}) p(\underline{\eta})}{\int p(x \mid \underline{\eta}') p(\underline{\eta}') d\underline{\eta}'} \quad (801)$$

is from the same family as the prior - *posterior and prior are conjugate*. Some likelihoods and their conjugate priors can be found in table 26.

Example: For a binomial likelihood (which is from the exponential family) the conjugate

Likelihood	Model parameters	Conjugate prior distribution
Bernoulli	μ	Beta
Binomial	μ	Beta
Normal with known variance σ^2	μ	Normal
Exponential	λ	Gamma

Table 26: Likelihoods and their conjugate priors

prior is the beta distribution. From the likelihood and prior

$$\text{binom.likelihood } P(k | \mu) = \text{Bin}(k; N, \mu) = \binom{N}{k} \mu^k (1 - \mu)^{N-k} \quad (802)$$

$$\text{prior } P(\mu) = \text{Beta}(\mu; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \mu^{\alpha-1} (1 - \mu)^{\beta-1}$$

one finds by Bayes rule

$$\text{posterior } p(\mu | k) = \text{Beta}(\mu; \alpha + k, \beta + N - k) \quad (803)$$

so we have updated the distribution of μ as of the new evidence k .

Moments of the exponential family in terms of the sufficient statistic \underline{T} based on the link $h(\underline{\eta})$ Based on the normalization $\int p(x | \underline{\eta}) dx = 1$ we find

$$0 = \partial_{\underline{\eta}} \int p(x | \underline{\eta}) dx = \int (\underline{T}(x) - \partial_{\underline{\eta}} h(\underline{\eta})) p(x | \underline{\eta}) dx = E[\underline{T}(x)] - \partial_{\underline{\eta}} h(\underline{\eta}) \quad (804)$$

The expectation value of the sufficient statistic $\underline{T}(x)$ is the gradient of the link function $h(\underline{\eta})$.

$$E[\underline{T}(x)] = \partial_{\underline{\eta}} h(\underline{\eta}), \quad E[T_j] = \partial_{\underline{\eta}_j} h(\underline{\eta}) \quad (805)$$

and the covariance matrix of the sufficient statistic is the Hessian of the link function

$$\text{Cov}[T_i, T_j] = \frac{\partial^2 h(\underline{\eta})}{\partial \underline{\eta}_i \partial \underline{\eta}_j} \quad (806)$$

Writing the Bernoulli distribution in the from of the exponential family We want to bring the Bernoulli distribution

$$p(x|\mu) = \text{Bern}(x; \mu) = \mu^x (1 - \mu)^{1-x}, \quad x \in \{0, 1\} \quad (807)$$

to the form

$$p(x|\eta) = \exp(\eta T(x) - h(\eta) + g(x)) \quad (808)$$

We start with

$$\begin{aligned} p(x) &= \exp(\log(\mu^x(1-\mu)^{1-x})) \\ &= \exp(x \log(\mu) + (1-x) \log(1-\mu)) \\ &= \exp\left(\underbrace{x}_{:=T(x)} \underbrace{\log\left(\frac{\mu}{1-\mu}\right)}_{:=\eta} + \log(1-\mu)\right) \\ &\stackrel{\mu=\frac{1}{1+\exp(-\eta)}}{=} \exp\left(T(x)\eta + \log\left(1 - \frac{1}{1+\exp(-\eta)}\right)\right) \\ &= \exp\left(T(x)\eta - \underbrace{\log(1+\exp(\eta))}_{:=h(x)}\right) \end{aligned} \quad (809)$$

With this we have found the link function $h(\eta) = \log(1 + \exp(\eta))$ and the sufficient statistic $T(x) = x$. And can thus calculate the expectation value of the sufficient statistic as

$$E[T(x)] = E[x] = \partial_\eta h(\eta) = \frac{\exp(\eta)}{1 + \exp(\eta)} = \mu \quad (810)$$

16.5 Tschebysheff's theorem

Tschebysheff's theorem is a general result for the probability of a random variable to deviate from its mean more than k standard deviations. For any distribution with finite mean μ and variance σ^2

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2} \quad (811)$$

This is illustrated for a Gaussian distribution in figure 142.

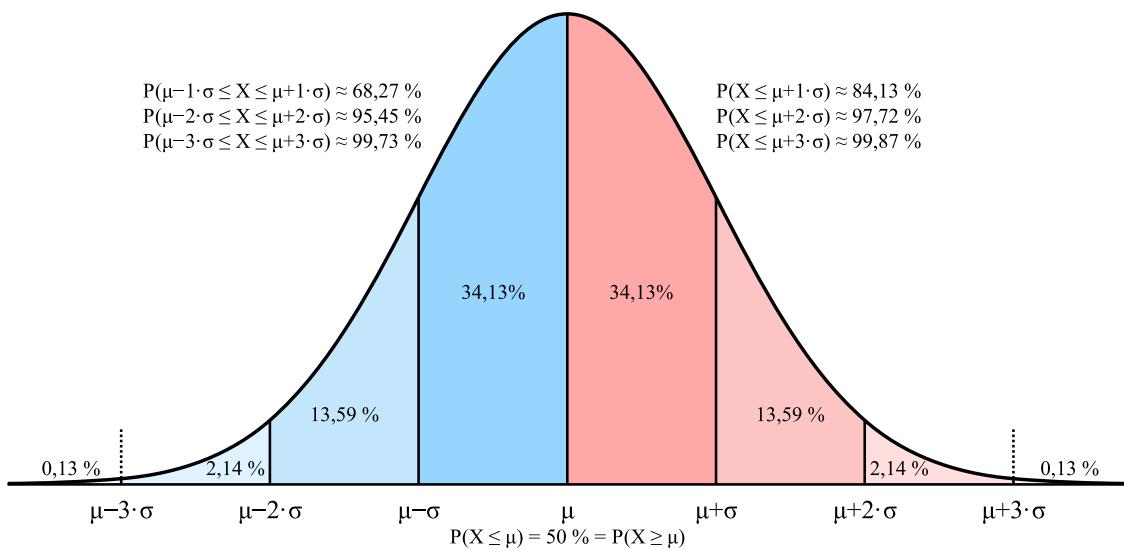


Figure 142: For the normal distribution $P(|X - \mu| \geq 2\sigma) \approx 1 - 0.9545 \approx 0.05 \leq \frac{1}{2^2}$ - so here Tschebysheff's theorem is not very tight.

16.5.1 Proof of Tschebysheff's theorem

$$\begin{aligned}
 (k\sigma)^2 \cdot P(|X - \mu| \geq k\sigma) &= \underbrace{\int_{-\infty}^{\mu-k\sigma} (k\sigma)^2 f(x) dx}_{\text{note here } (k\sigma)^2 \leq (x-\mu)^2} + \int_{\mu+k\sigma}^{\infty} (k\sigma)^2 f(x) dx \\
 &\leq \int_{-\infty}^{\mu-k\sigma} (x - \mu)^2 f(x) dx + \underbrace{\int_{\mu-k\sigma}^{\mu+k\sigma} (x - \mu)^2 f(x) dx}_{\geq 0} + \int_{\mu+k\sigma}^{\infty} (x - \mu)^2 f(x) dx \\
 &= \sigma^2
 \end{aligned} \tag{812}$$

16.6 Moment generating functions

To characterize and compare distributions, we use their moments²⁸

$$m_n := E[X^n] = \int x^n f(x) dx \tag{813}$$

where the mean is the first moment. The **moment generating function** is

$$\boxed{M_X(t) := E[\exp(tx)]} = \int \exp(tx) f(x) dx \tag{814}$$

which is defined if

$$\exists h > 0 : \forall t \in (-h, h) : M_X(t) < \infty \tag{815}$$

²⁸Which under certain conditions can completely and uniquely specify a distribution.

so for instance not for the Cauchy distribution (which does not even have a finite mean).

From Taylor expansion to the moments Consider the series expansion

$$\exp(tX) = 1 + tX + \frac{t^2 X^2}{2!} + \frac{t^3 X^3}{3!} + \cdots + \frac{t^n X^n}{n!} + \dots \quad (816)$$

so as of the linearity of the expectation value

$$E[\exp(tX)] = 1 + tE[X] + \frac{t^2 E[X^2]}{2!} + \frac{t^3 E[X^3]}{3!} + \cdots + \frac{t^n E[X^n]}{n!} + \dots \quad (817)$$

Now consider $\partial_t M_X(t)|_{t=0}$ - it is just $E[X]$.

16.6.1 Calculating moments from the moment generating function

The n -th moment of X is the n -th derivative of the moment generating function at $t = 0$

$$m_n := E[X^n] = \left. \frac{d^n M_X(t)}{dt^n} \right|_{t=0} \quad (818)$$

16.6.2 Multidimensional moment generating function

For the random vector

$$\underline{X} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{pmatrix} \quad (819)$$

the moment generating function is

$$M_{\underline{X}}(\underline{t}) = E[\exp(\underline{t}^T \underline{X})] \quad (820)$$

16.6.3 Example: Moment generating function of the Poisson distribution

The (discrete) Poisson equation is

$$P(k; \lambda) = \frac{\lambda^k}{k!} \exp(-\lambda) \quad (821)$$

so the moment generating function is (using $\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}$)

$$\begin{aligned} E[\exp(tk)] &= \sum_{k=0}^{\infty} \exp(tk) \frac{\lambda^k}{k!} \exp(-\lambda) \\ &= \exp(-\lambda) \sum_{k=0}^{\infty} \frac{(\exp(t)\lambda)^k}{k!} \\ &= \exp(-\lambda) \exp(\exp(t)\lambda) \\ &= \exp(\lambda(\exp(t) - 1)) \end{aligned} \tag{822}$$

16.7 Multivariate probability distributions

Let us consider the joint probability distribution of multiple RVs

$$p(x_1, x_2, \dots, x_k) = p(X_1 = x_1, X_2 = x_2, \dots, X_k = x_k) \tag{823}$$

so the probability of a certain combination of outcomes of the RVs. The normalization is (integrals in the continuous case)

$$\sum_{x_1} \cdots \sum_{x_k} p(x_1, x_2, \dots, x_k) = 1 \tag{824}$$

The joint cumulative distribution function is

$$F_{X_1, \dots, X_k}(x_1, x_2, \dots, x_k) = p(X_1 \leq x_1, X_2 \leq x_2, \dots, X_k \leq x_k) \tag{825}$$

which is monotonically non-decreasing in all its arguments, bound by 0 and 1 and has the limits

$$\lim_{x_1, \dots, x_n \rightarrow +\infty} F_{X_1 \dots X_n}(x_1, \dots, x_n) = 1 \quad \text{and} \quad \lim_{x_i \rightarrow -\infty} F_{X_1 \dots X_n}(x_1, \dots, x_n) = 0, \text{ for all } i. \tag{826}$$

16.7.1 Multi-categorical distribution

Consider a setting of multiple possible classes not just failure and success, e.g. the probability of a randomly chosen farm animal being some species.

16.7.1.1 Encoding the category by 1-hot encoding

Consider we have k (exclusive) categories. The naive encoding would be to assign a number to each category, $y = 1$ for cow, $y = 2$ for donkey, 3 for sheep, etc. This has two major

disadvantages:

- it implies an ordering and distance, in this space cow and sheep are farther apart than cow and donkey, which makes no sense
- moments of the distribution, e.g. the expected value, are nonsensical

We better use **1-hot encoding**, where k categories are encoded in a k -dimensional vector with a 1 at the position of the active category and 0 elsewhere. So for an observation \underline{y}_i (a single data-point), we have

$$[\underline{y}_i]_k = \begin{cases} 1 & \text{if } x_i \text{ belongs to class } k \\ 0 & \text{else} \end{cases}, \quad (827)$$

so an observation of class k has the vector

$$k : \underline{y}_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow \text{k-th position} \quad (828)$$

where 1-hot (class exclusivity) means that for any observation

$$\sum_k [\underline{y}_i]_k = 1 \quad (829)$$

16.7.1.2 Categorical distribution

For K classes with probabilities μ_k , so

$$\sum_{k=1}^K \mu_k = 1, \quad \mu_k \geq 0, \quad \underline{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_K \end{pmatrix} \quad (830)$$

and the categorical distribution can be written as

$$p(\underline{y} \mid \underline{\mu}) = \prod_{k=1}^K \mu_k^{y_k}, \quad y_k \in \{0, 1\} \text{ so that } p(y = k \mid \underline{\mu}) = \mu_k \quad (831)$$

So for $K = 2$ we are back at the Bernoulli distribution.

16.7.1.3 Expectation, variance and covariance of the categorical distribution

For many observations $\{\underline{y}_i\}_{i=1}^N$ we expect for the categories (entries of \underline{y}_i)

$$\begin{aligned} E[y_k] &= \mu_k, \quad \text{Var}(y_k) = \mu_k(1 - \mu_k) \\ \text{Cov}(y_k, y_l) &= \underbrace{E[y_k y_l]}_{=0 \text{ for } k \neq l \text{ as 1-hot}} - E[y_k] E[y_l] = -\mu_k \mu_l \text{ for } k \neq l \\ \underline{E}[y] &= \underline{\mu}, \quad \text{covariance matrix } \underline{\Sigma} \in \mathbb{R}^{K \times K} \end{aligned} \tag{832}$$

16.7.2 Multinomial distribution

Consider we cannot only draw from two possibilities (like success and failure) but from K possibilities. For example, consider 3 blue, 2 red and 5 green balls in an urn and we draw 20 with replacement. What is the probability for drawing 10 green, 5 blue and 5 red balls?

The multinomial is given by

$$\begin{aligned} p(x_1, \dots, x_k) &= \frac{N!}{x_1! \dots x_k!} \prod_{i=1}^k \mu_i^{x_i}, \quad \text{count } x_i \text{ of occurrences of class } i, \quad \sum_{i=1}^k x_i = N \\ &\text{in } N \text{ trials, permutations } x_1! \dots x_k! \text{ within the classes} \\ &\text{parameters } N, \mu_i, \quad \sum \mu_i = 1, \quad \mu_i \geq 0 \end{aligned} \tag{833}$$

16.7.2.1 Expectation, variance and covariance of the multinomial distribution

$$E[x_i] = N\mu_i, \quad \text{Var}[x_i] = N\mu_i(1 - \mu_i), \quad \text{Cov}[x_i, x_j] = -N\mu_i\mu_j \text{ for } i \neq j \tag{834}$$

16.7.3 Multivariate Gaussian distribution

Consider we have recorded multiple continuous features like temperature, humidity and pressure. For a feature vector

$$\underline{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix} \tag{835}$$

with mean

$$\underline{\mu} = E[\underline{x}] = \begin{pmatrix} E[x_1] \\ \vdots \\ E[x_p] \end{pmatrix} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_p \end{pmatrix} \in \mathbb{R}^p \quad (836)$$

and covariance matrix²⁹

$$\text{Cov}(\underline{X}) = \underline{\underline{\Sigma}} = \underbrace{E[(\underline{X} - \underline{\mu})(\underline{X} - \underline{\mu})^T]}_{\text{applied elementwise}} = E[\underline{X}\underline{X}^T] - \underline{\mu}\underline{\mu}^T \in \mathbb{R}^{p \times p} \quad (837)$$

so with the elements

$$\Sigma_{ij} = \text{Cov}(x_i, x_j) = E[(x_i - \mu_i)(x_j - \mu_j)] \quad (838)$$

to follow a multivariate Gaussian distribution, $\underline{X} \sim \mathcal{N}(\underline{\mu}, \underline{\underline{\Sigma}})$, means

$$\phi_{\underline{\mu}, \underline{\underline{\Sigma}}}(\underline{x}) = (2\pi)^{-\frac{p}{2}} (\det \underline{\underline{\Sigma}})^{-\frac{1}{2}} \exp \left(-\frac{1}{2} (\underline{x} - \underline{\mu})^T \underline{\underline{\Sigma}}^{-1} (\underline{x} - \underline{\mu}) \right) \quad (839)$$

For the case of two features, the Gaussian distribution is illustrated in figure 143 and 144.

²⁹One can also define the precision matrix $\underline{\Lambda} = \underline{\underline{\Sigma}}^{-1}$ (\sim more variation \rightarrow less precision).

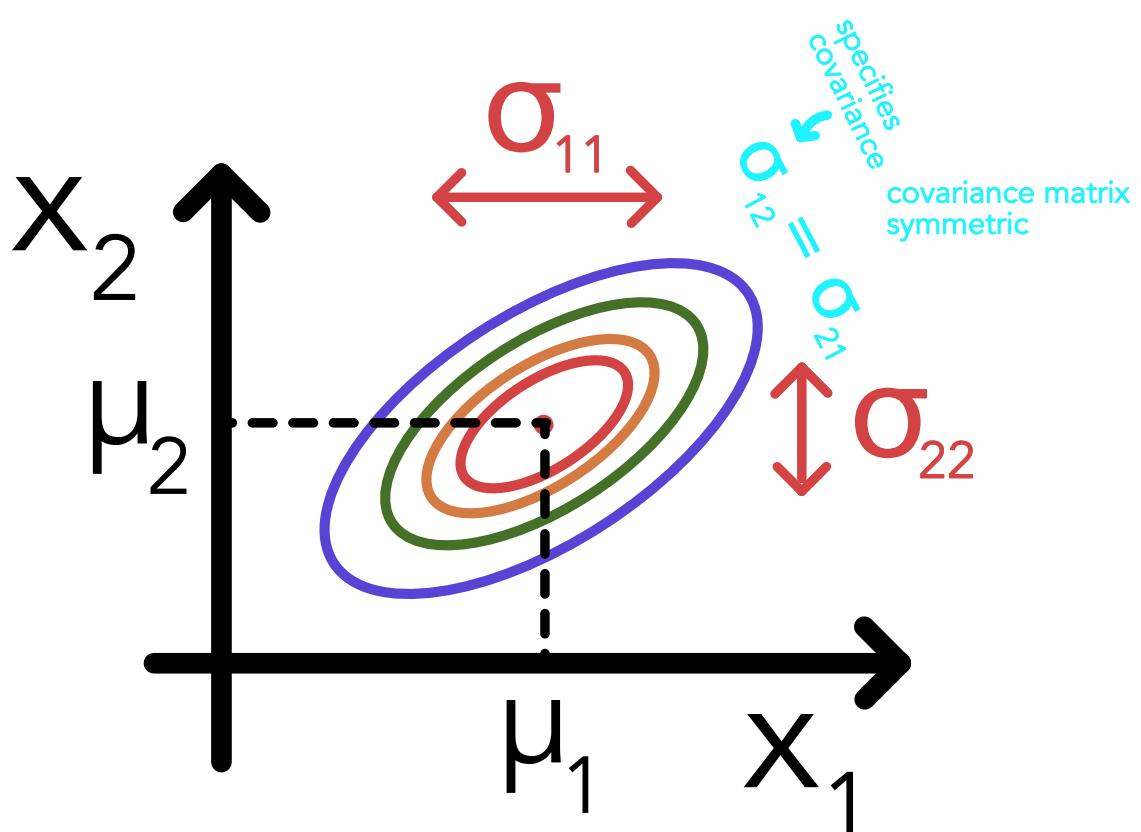


Figure 143: Gaussian distribution in 2D

Bivariate normal distributions

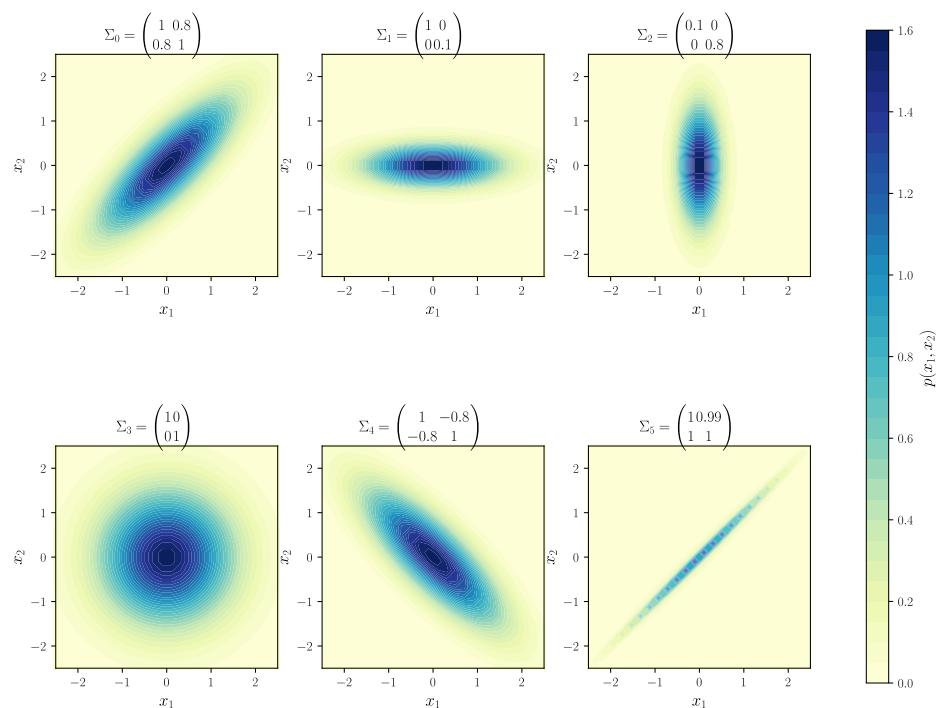


Figure 144: Gaussian distribution in 2D

17 Statistical Inference

17.1 Introduction to Inference

We assume the black box of nature to be some model³⁰ F_θ where from a sample

$$X_N = \{x_1, \dots, x_N\}, \quad \text{sample size } N \quad (840)$$

we would like to draw conclusions on a population

$$\text{population size } K \gg N \quad (841)$$

and estimate parameters $\hat{\theta}$, see figure 145.

Goal in statistical inference: Find out about the underlying principles that generated the data by formulating models and inferring their parameters.

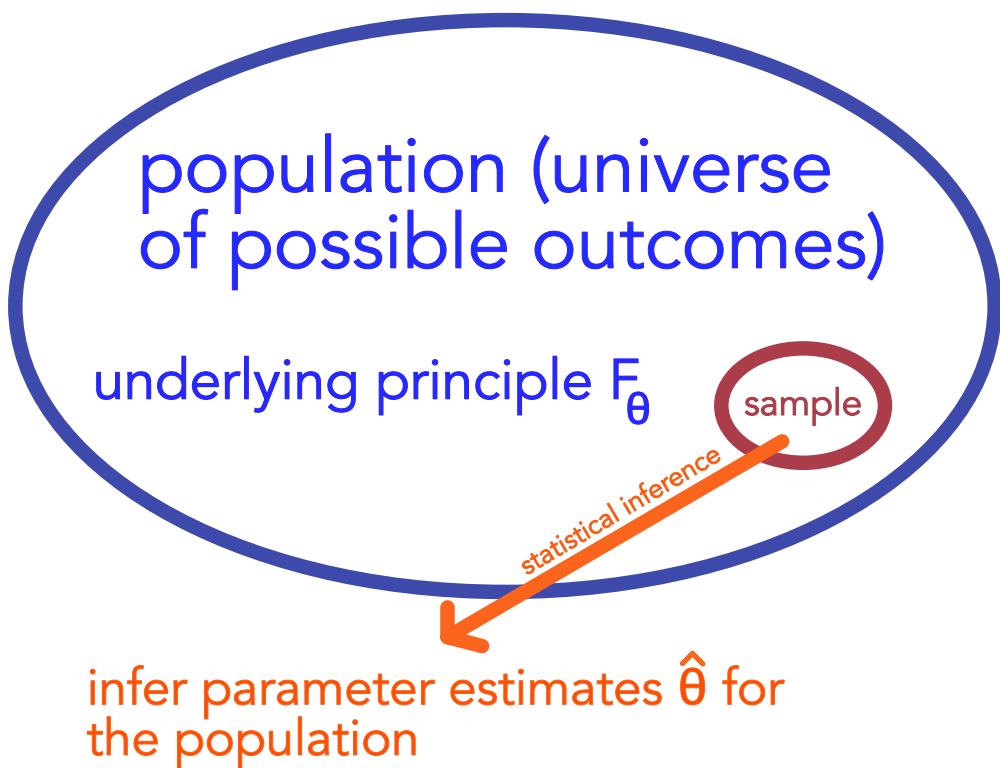


Figure 145: Statistical inference

Each sample from the population has the probability

³⁰So we are more in the data modeling culture here.

$$p(x_N) = \frac{1}{\binom{K}{N}} \quad (842)$$

17.1.1 Statistics

Any quantity computed from values in a sample is called statistic

$$T(x), \quad \text{e.g. } \bar{x}_N = \frac{1}{N} \sum_{i=1}^N x_i, s_N^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x}_N)^2 \quad (843)$$

which is done

- for estimating a population parameter (e.g. the mean)
- or evaluating a hypothesis (does the calculated statistic on the sample conform to its assumed statistic under the hypothesis?)

We want those statistics to be maximally informative (sufficient) for the population parameters, and in the limit of the sample size $\rightarrow \infty$ to equal the population parameters, e.g.

$$\lim_{N \rightarrow \infty} \bar{x} = E[x] =: \mu, \quad \lim_{N \rightarrow \infty} s_n^2 = \text{Var}(x) \quad (844)$$

more formally, the *weak law of large numbers* state for the mean

$$\bar{x}_N \xrightarrow{P} \mu \text{ meaning } \forall \epsilon > 0 : \lim_{N \rightarrow \infty} p(|\bar{x}_N - \mu| > \epsilon) = 0, \quad \text{probability } p \quad (845)$$

and the *strong law of large numbers*

$$\bar{x}_N \xrightarrow{a.s.} \mu \text{ meaning } p\left(\lim_{N \rightarrow \infty} \bar{x}_N = \mu\right) = 1 \quad (846)$$

where a. s. stands for almost surely.

17.1.2 General Questions we ask ourselves in statistical modeling

Consider we want to carry out an experiment, e.g. we measure lifetimes for a certain disease.

- What model can we use for our data? Or should we better take an *algorithmic approach* in the first place?
- What are good statistics, i.e. what quantities should we compute from our sample?
- How large should a sample be?

- What is a good estimate for a population parameter θ ? How can we obtain an estimate $\hat{\theta}$?
- How accurate do we assume this estimate to be?

17.2 Examples of Statistical Models - baseline of statistical modeling

We formulate a supposedly underlying model of our data with population parameters (denoted in Greek letters) where from test statistics (Roman letters) we want to infer parameter estimates (Greek letters with hats) or test hypothesis on them.

How can a statistical model look like?

17.2.1 General form of the models

A model has the general form

$$\text{response } y_i(\text{features } \underline{x}_i) = \text{structural part}_\theta(\underline{x}_i) + \text{random part} \epsilon_i \quad (847)$$

where the random part - under the assumption of many small additive errors - is usually modelled by

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2) \quad (848)$$

17.2.2 Example I: One- and two-factor univariate analysis; discrete RVs*

In the following we consider simple models where discrete characteristics (e.g. a disease is present) are modeled by constant terms across samples.

One-factor: The virus load of patient i medicated with drug j , x_{ij} could be described by

$$x_{ij} = \mu + \tau_j + \epsilon_{ij}, \quad \text{random part } \epsilon_{ij} \sim \mathcal{N}(0, \sigma^2) \text{ i.i.d. so } \forall i, j \neq k, l : E[\epsilon_{ij}\epsilon_{il}] = 0 \quad (849)$$

mean contribution μ , drug specific effect τ_j (assume no patient specificity)

$\epsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$ is a distributional assumption.

Two-factor: Adding e.g. a gender (k) effect β_k and cross effect $\alpha\beta_{jk}$ (present when the gender effect varies between drugs)

$$x_{ijk} = \mu + \alpha_j + \beta_k + \alpha\beta_{jk} + \epsilon_{ijk} \quad (850)$$

17.2.3 Example II: Linear regression model

Consider a dataset

$$D = \{\text{features, response}\}_{i=1}^N, \quad i = 1, \dots, N \quad (851)$$

with a linear model being

$$y(\underline{x}) = \underline{\beta}^* \cdot \underline{x} + \epsilon(x), \quad \text{true parameter } \underline{\beta}^* = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \quad (852)$$

augmented independent variables $\begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_p \end{pmatrix}$

so if there is only one feature x_i , we have

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots + \epsilon_i \quad (853)$$

Note: A linear model is linear in the parameters, not necessarily the independent variables.
For instance

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots + \epsilon_i \quad (854)$$

is a linear model (polynomial regression).

Note: How to check if a linear model breaks down?: Often plots of the residuals are done. If they show a trend, i.e. they are not i.i.d. with zero mean, our model is probably not applicable.

17.2.4 Example III: Generalized linear model

Here we generally assume the data was brought forth from a distribution from the exponential family

$$p(y|\theta) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)\right), \quad (855)$$

with real functions a, b, c , and parameters θ, ϕ

which mean we model.

$$E[y_i | x_i] = \mu = g^{-1}(\eta_i) = \partial_\theta b, \quad \text{linear predictor } \eta_i = \underline{\beta}^T \underline{x}_i \quad (856)$$

Note: The notation is slightly different from before. For a *canonical link* we have $\theta = \eta$ as before.

17.2.5 Example IV: Autoregressive model

In the model for a time-series, e.g. the development

$$x_t = \alpha x_{t-1} + \epsilon_t \quad (857)$$

the x_t are of course on i.i.d.

17.3 Estimation of Model Parameters

Consider we now have our model F_θ . For instance, we could assume a linear model for the yearly temperature mean on earth. We might be interested in

- make point estimates for the parameters, e.g. the temperature increases by 0.06 K per decade
- the estimated deviation of our estimator over many samples (see figure 146) (the *sampling distribution*), called standard error, so **an error approximation for our estimator**
- based on the standard error and distributional assumptions on the estimator, interval estimates for the parameters, e.g. we are 95% certain that the temperature increases by 0.05 K to 0.07 K per decade
- hypothesis tests about model parameters (or other statistics from the data), for instance to test if there even is a linear trend in the mean temperature, e.g. with *null-hypothesis* (no difference, devils advocate): there is no global warming, $H_0 : \mu = 0, H_1 : \mu \neq 0$

We will first consider what a good estimator is and then discuss strategies for estimation.

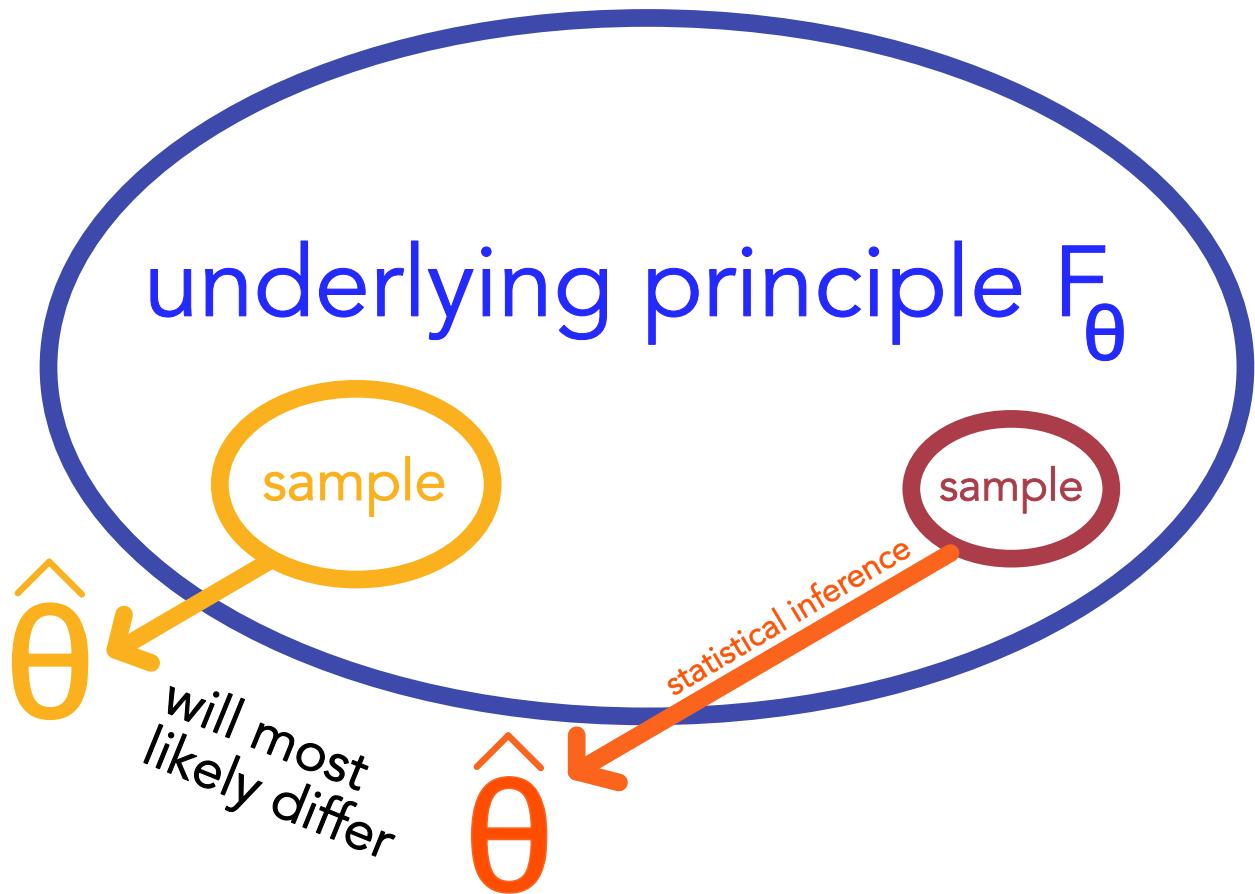


Figure 146: Different sample, different estimate - the estimator itself is a random variable

17.3.1 Properties of estimators

What makes a good estimator?

17.3.1.1 Bias of an estimator

If the expected value of our estimator over many samples

$$E[\hat{\theta}_N] = \theta^* + c, \quad \text{true parameter } \theta^*, \text{ bias } c \quad (858)$$

does not equal the population parameter, it is called biased, for $c = 0$ unbiased. Good estimators are unbiased.

17.3.1.2 Consistency of an estimator

In the limit of an infinitely large sample, the estimator should in probability converge to the true parameters

$$\forall \epsilon : \lim_{N \rightarrow \infty} p(|\hat{\theta}_N - \theta^*| < \epsilon) = 1 \quad (859)$$

Estimator	Bias	Consistency
$\hat{\mu}_I = x_i, \text{ random } i$	Unbiased, as $E[\hat{\mu}_I] = \mu$.	Inconsistent, $N \rightarrow \infty$ as sample size makes no difference as we pick a random value.
$\hat{\mu}_{II} = \frac{1}{N} \sum_{i=1}^N x_i + \frac{1}{N}$	Biased as $E[\hat{\mu}_I] = \mu + E\left[\frac{1}{N}\right]$	Consistent as for $N \rightarrow \infty$ $\frac{1}{N} \rightarrow 0$ and thus $\hat{\mu}_{II} \rightarrow \mu$
$\hat{\mu}_{III} = \frac{1}{N} \sum_{i=1}^N x_i$	Unbiased	Consistent

Table 27: Bias and consistency at the hand of estimators for the mean

17.3.1.3 Examples for bias and consistency at the hand of the mean

Aim: We want to estimate the expectation $E[X]$ from an i.i.d. sample $\{x_i\}_{i=1}^N$. For the total population, we know

$$E[X] = \mu = \frac{1}{K} \sum_{i=1}^K x_i, \quad \text{size of the total population } K \quad (860)$$

We consider different estimators in table 27.

17.3.1.4 The naive sample variance estimation is a biased but consistent estimator

The population variance is

$$\sigma^2 = \frac{1}{K} \sum_{i=1}^K (x_i - \bar{x})^2 \quad (861)$$

The analogue sample estimator for the population variance

$$s_x^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 \quad (862)$$

however is biased as of (proof later)

$$E[s_x^2] = \frac{N-1}{N} \sigma^2 \quad (863)$$

so we use the *Bessel-correction*

$$\hat{\sigma}_{\text{unbiased}}^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2 \quad (864)$$

Our upscaled correction $\hat{\sigma}_{\text{unbiased}}^2$ takes into account that \bar{x} is the sample estimate not the population parameter from which the values from the sample naturally deviate less than from the true mean.

While $\hat{\sigma}_{\text{unbiased}}^2$ is unbiased, $\hat{\sigma} = \sqrt{\hat{\sigma}_{\text{unbiased}}^2}$ generally is not. This can be followed from Jenssens inequality

$$\begin{aligned} X \text{ a random variable, } \phi \text{ a convex function} &\rightarrow \phi(E[X]) \leq E[\phi(X)] \\ \phi \text{ a concave function} &\rightarrow \phi(E[X]) \geq E[\phi(X)] \end{aligned} \quad (865)$$

by

$$\begin{aligned} \text{assume we use } \hat{\sigma} = \sqrt{\frac{1}{N-1} s_x^2} \text{ as our estimator, then} \\ E[\hat{\sigma}] = E \left[\sqrt{\frac{1}{N-1} s_x^2} \right] \stackrel{\text{Jenssens ineq.}}{\leq} \sqrt{E \left[\frac{1}{N-1} s_x^2 \right]} = \sigma \end{aligned} \quad (866)$$

17.3.1.5 Sampling distribution

Consider we repeatedly draw samples of size N and estimate $\hat{\theta}_N$. The distribution of those estimates is called sampling distribution, denoted $F_N(\hat{\theta})$.

17.3.1.6 Standard error of an estimator

Consider an estimator $\hat{\theta}_N$ with sampling distribution $F_N(\hat{\theta})$. We then call the standard deviation of the sampling distribution standard error.

$$\text{variance in } \hat{\theta}_N \rightarrow \text{SE}_{\hat{\theta}}(N) := \sqrt{E[(\hat{\theta}_N - E[\hat{\theta}_N])^2]} \quad (867)$$

we can also define the **mean squared error**, accounting for bias and variance in $\hat{\theta}_N$

$$\text{MSE}(\hat{\theta}_N) = \sqrt{E[(\hat{\theta}_N - \theta^*)^2]}, \quad \text{true parameter } \theta^* \quad (868)$$

where for an unbiased estimator ($E[\hat{\theta}_N] = \theta^*$) so $\text{SE}_{\hat{\theta}}(N) = \text{MSE}(\hat{\theta}_N)$.

Standard error of the means

$$SEM = SE_{\hat{\mu}}(N) = \sqrt{\text{Var}(\bar{x}_N)} = \sqrt{\text{Var}\left(\frac{1}{N} \sum_{i=1}^N x_i\right)}$$

$\text{Var}(aX+b)=a^2 \text{Var}(X)$

$$\stackrel{\sum_i \text{Var}(X_i)}{=} \stackrel{x_i \text{ independent}}{=} \frac{1}{N} \sqrt{\sum_{i=1}^N \text{Var}(x_i)} = \frac{1}{N} \sqrt{N\sigma} = \frac{\sigma}{\sqrt{N}}$$
(869)

General estimation of standard errors by bootstrapping: If for the statistic of interest there is no explicit formula for the standard error and we only have one sample at hand, we can estimate the standard error by the standard deviation of resampled samples from our sample (by choosing randomly from our sample with replacement). More on this later :)

17.3.1.7 Sufficient statistic

A statistic or set of statistics $t(\underline{X})$ is sufficient if it contains all information there is in a sample about a population parameter θ , so

$$p(\underline{X}|t(\underline{X}), \theta) = p(\underline{X}|t(\underline{X})) \quad (870)$$

so the likelihood of the sample is independent of the population parameter given the sufficient statistic.

The smallest set of sufficient statistics is called minimally sufficient, e.g. for a Gaussian these are the sample mean μ and variance $\frac{N}{N-1} s_x^2$.

17.3.1.8 Efficiency of an estimator

Consider $\hat{\theta}_{\text{opt}}$ to be the estimator with the smallest variance (smallest standard error). The efficiency of any other estimator $\hat{\theta}_k$ is then

$$\text{Eff}_{\hat{\theta}_k} = \frac{\text{Var}(\hat{\theta}_{\text{opt}})}{\text{Var}(\hat{\theta}_k)} \in [0, 1] \quad (871)$$

Note: The maximum likelihood estimator (the estimator under which the observed data is most probable) is an efficient estimator.

17.3.1.9 Precision of an estimator

We can define the precision of an estimator as

$$\text{Prec}(\hat{\theta}_N) = \frac{1}{\text{SE}_{\hat{\theta}}(N)} \quad (872)$$

- the less the estimator varies over many samples, the more precise it is. For an unbiased estimator, the precision is fundamentally limited by the *Fisher information*³¹, so

$$\text{Var}(\hat{\theta}_N) \geq \frac{1}{\mathcal{I}(\theta)} \quad (874)$$

(Cramer-Rao bound).

17.3.1.10 What characterizes a good estimator?

A good estimator is

- unbiased (its expected value is the population one)
- consistent (in the limit of a large sample we get the population parameter)
- efficient, so low standard error (it should not vary a lot between different samples)
- minimally sufficient (it should extract all information from the sample with regards to the population parameter of interest)

17.3.2 Approaches to Statistical Parameter Estimation

Question: What are the best model parameters for a given model which we can deduce from a sample?

Principal ways of parameter estimation are

- Least squared error estimation (LSE), minimize the (mean) squared error between the response predicted by the model under the estimated parameters and the sample
- maximum likelihood estimation (MLE), choose parameters making the sample most likely

³¹The partial derivative of the logarithm of the likelihood $p(\underline{X}|\theta)$ with respect to θ is called score. The Fisher information is the variance of the score, so

$$\mathcal{I}(\theta) = E \left[\left(\frac{\partial}{\partial \theta} \log p(\underline{X}|\theta) \right)^2 | \theta \right] = \int \left(\frac{\partial}{\partial \theta} \log p(\underline{X}|\theta) \right)^2 p(\underline{X}|\theta) d\underline{X} \quad (873)$$

- Bayesian inference (BI), estimate the whole posterior of the parameters $p(\underline{\theta} | \{\underline{x}_i\}_{i=1}^N)$
- Maximum a posteriori (MAP), take the parameters maximizing the posterior $p(\underline{\theta} | \{\underline{x}_i\}_{i=1}^N)$

17.3.2.1 Least squared error

For the general model

$$y_i = f_{\underline{\theta}}(\underline{x}_i) + \epsilon_i \quad (875)$$

in LSE, we calculate the sum of squares loss on the sample $\{\underline{x}_i, y_i\}_{i=1}^N$ as

$$\text{SSQ}(\underline{\theta}) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2, \quad \text{model estimate } \hat{y}_i = E[y_i | x_i] = E[f_{\underline{\theta}}(\underline{x}_i)] + \underbrace{E[\epsilon_i]}_{=0} = f_{\underline{\theta}}(\underline{x}_i) \quad (876)$$

where here $\underline{\theta}$ denote some model parameters. As an estimate for the true parameters $\underline{\theta}^*$ we can use

$$\hat{\underline{\theta}}_{LSE} = \underset{\underline{\theta}}{\operatorname{argmin}} \text{SSQ}(\underline{\theta}) \quad (877)$$

Least squared error estimation for linear regression Let us write the sample $\{\underline{x}_i, y_i\}_{i=1}^N$ in the form

$$\underline{Y} \in \mathbb{R}^N, \quad \underline{\underline{X}} \in \mathbb{R}^{N \times p} \quad (878)$$

where \underline{Y} is called response or measurement and $\underline{\underline{X}}$ is the feature matrix (independent variables).

We can then write the regression problem as

$$\begin{aligned} \underline{Y} &= f_{\beta}(\underline{\underline{X}}) + \underline{\epsilon} = \underline{\underline{X}}\beta + \underline{\epsilon}, \quad \text{residuals } \underline{\epsilon} \\ \underline{\beta} &= \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \text{feature matrix is augmented by prepended column of 1s} \end{aligned} \quad (879)$$

\underline{Y} is a random variable by virtue of the residual $\underline{\epsilon}$, $\epsilon_i \sim N(0, \sigma^2)$ i.i.d.

We can calculate the sum of squares residual (a scalar) as

$$\text{SSQ}(\underline{\beta}) = \|\underline{Y} - \underline{\underline{X}}\beta\|_2^2 = (\underline{Y} - \underline{\underline{X}}\beta)(\underline{Y} - \underline{\underline{X}}\beta)^T = \|\underline{\epsilon}\|_2^2 \quad (880)$$

which is nicely convex, so we can find a unique minimum by

$$\begin{aligned}\frac{\partial \text{SSQ}}{\partial \underline{\beta}} &= -2\underline{\underline{X}}^T(\underline{Y} - \underline{\underline{X}}\underline{\beta})_! = 0 \\ \rightarrow \underline{X}^T \underline{Y} &= \underline{\underline{X}}^T \underline{\underline{X}}\underline{\beta} \rightarrow \hat{\underline{\beta}} = (\underline{\underline{X}}^T \underline{\underline{X}})^{-1} \underline{\underline{X}}^T \underline{Y} \text{ (normal equation)}\end{aligned}\quad (881)$$

Note: For this to apply $\underline{\underline{X}}^T \underline{\underline{X}}$ must be invertible, so $\underline{\underline{X}}$ must have rank p . This is e.g. not the case for the overparametrized $p > N$. $\underline{\underline{X}}^T \underline{\underline{X}}$ measures covariance, assuming the columns of $\underline{\underline{X}}$ to be centered, it is the sample covariance matrix (without a scaling factor), $\underline{\Sigma}_{XX} = \frac{1}{N-1} \underline{\underline{X}}^T \underline{\underline{X}} \in \mathbb{R}^{p \times p}$.

1D case: For $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$ one finds from $\partial_{\beta_0} \text{SSQ} = 0$ and $\partial_{\beta_1} \text{SSQ} = 0$

$$\begin{aligned}\hat{\beta}_0 &= \bar{y} - \hat{\beta}_1 \bar{x} \text{ interset as expected} \\ \beta_1 &= \frac{\frac{1}{N-1} \sum (x_i - \bar{x})(y_i - \bar{y})}{\frac{1}{N-1} \sum (x_i - \bar{x})^2} = \frac{\widehat{\text{Cov}}(x, y)}{\widehat{\text{Var}}(x)}\end{aligned}\quad (882)$$

17.3.2.2 Maximum likelihood estimation (MLE)

The likelihood of a sample $\{\underline{x}_i\}_{i=1}^N$ of N i.i.d. data points under a model $F_{\underline{\theta}}$ which assigns each data point a probability $p(\underline{x}_i | \underline{\theta})$ is

$$\mathcal{L}_X(\underline{\theta}) = \prod_{i=1}^N p(\underline{x}_i | \underline{\theta}) \quad (883)$$

For e.g. linear regression this model is hidden in the assumption of the residuals being i.i.d. Gaussian with variance σ^2 , where

$$p(y_i | \underline{x}_i, \underline{\beta}) = \mathcal{N}(\underline{\beta}^T \underline{x}_i, \sigma^2) \quad (884)$$

In the generalized linear setting, this will not be a normal distribution but a distribution from the exponential family and the mean will be modelled by a link function (its inverse).

Idea: Model parameters under which the sample is likely are probably good.

$$\hat{\underline{\theta}}_{MLE} = \underset{\underline{\theta}}{\operatorname{argmax}} \mathcal{L}_X(\underline{\theta}) \quad (885)$$

Problem: The product of many small probabilities can numerically be problematic (underflow)

Idea: We can alternatively maximize the log-likelihood or minimize the negative log-likelihood

$$\mathcal{LL}_X(\underline{\theta}) = \log \mathcal{L}_X(\underline{\theta}) = \sum_{i=1}^N \log p(x_i | \underline{\theta}) \quad (886)$$

so

$$\hat{\underline{\theta}}_{MLE} = \operatorname{argmax}_{\underline{\theta}} \mathcal{LL}_X(\underline{\theta}) \quad (887)$$

MLE for linear regression - same result as LSE Again consider a dataset $\{\underline{x}_i, y_i\}_{i=1}^N$ with a linear model $y_i = \underline{\beta}^T \underline{x}_i + \epsilon_i$ and i.i.d. residuals $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. The likelihood of the sample under some model parameters $\underline{\beta}$ is then

$$\begin{aligned} \mathcal{L}_X(\underline{\beta}) &= \prod_{i=1}^N p(y_i | \underline{\beta}) = \prod_{i=1}^N N(y_i - \underline{\beta}^T \underline{x}_i, \sigma^2) = \prod_{i=1}^N \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \underline{\beta}^T \underline{x}_i)^2}{2\sigma^2}\right) \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \exp\left(-\frac{\sum_{i=1}^N (y_i - \underline{\beta}^T \underline{x}_i)^2}{2\sigma^2}\right) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \exp\left(-\frac{SSQ(\underline{\beta})}{2\sigma^2}\right) \quad (888) \\ \hat{\underline{\beta}}_{MLE} &= \operatorname{argmax}_{\underline{\beta}} \mathcal{L}(\underline{\beta}) \Leftrightarrow \hat{\underline{\beta}}_{MLE} = \operatorname{argmin}_{\underline{\beta}} SSQ(\underline{\beta}) = \hat{\underline{\beta}}_{LSE} \end{aligned}$$

MLE for μ and σ^2 in a Gaussian model Consider a sample $\{x_i\}_{i=1}^N$ from a Gaussian distribution. What could be estimates for the mean μ and variance σ^2 ?

From the univariate Gaussian distribution

$$p(x_i | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right) \quad (889)$$

we calculate the log-likelihood $\mathcal{LL}_X(\mu, \sigma^2)$ and find

$$\begin{aligned} \partial_\mu \mathcal{LL}_X|_{\mu=\hat{\mu}_{MLE}} &= 0 \quad \rightarrow \quad \hat{\mu}_{MLE} = \frac{1}{N} \sum_{i=1}^N x_i \\ \partial_{\sigma^2} \mathcal{LL}_X|_{\sigma^2=\hat{\sigma}_{MLE}^2} &= 0 \quad \rightarrow \quad \hat{\sigma}_{MLE}^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2 \end{aligned} \quad (890)$$

(where we have carried out the calculations as if the other parameter was known). For $\mu = \hat{\mu}_{MLE}$, $\hat{\sigma}_{MLE}^2$ is the biased estimator for the population variance. Given the model

estimate $\hat{y} = \underline{\beta}^T \underline{x}$, the **MLE for the variance in linear regression** is

$$\hat{\sigma}_{MLE}^2 = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 = \frac{1}{N} \sum_{i=1}^N \epsilon_i^2 \quad (891)$$

17.3.2.3 Bayesian inference

Consider we have measured a sample $\{\underline{x}_i\}_{i=1}^N$ and want to estimate the model parameters $\underline{\theta}$.

By Bayes theorem we can calculate the full distribution of the parameters given the sample and prior knowledge $p(\underline{\theta})$. **From there we can e.g. take the parameters most likely given the data (MAP) or the expected value of the parameters,** The prior (essentially a *bias* or *regularization*) is especially useful when the sample size is small - we can update our previous knowledge with new evidence - the sample.

$$\begin{aligned} \text{posterior} &= p(\underline{\theta} | \underline{x}_1, \underline{x}_2, \dots, \underline{x}_N) = \frac{p(\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N | \underline{\theta}) p_{\underline{\alpha}}(\underline{\theta})}{h(\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}} \\ &= \frac{p(\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N | \underline{\theta}) p_{\underline{\alpha}}(\underline{\theta})}{\int p(\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N | \underline{\theta}') p_{\underline{\alpha}}(\underline{\theta}') d\underline{\theta}'}, \quad \text{prior } p_{\underline{\alpha}} \text{ with hyper-parameters } \alpha \end{aligned} \quad (892)$$

Updating our knowledge on the distribution of the model parameters given the data - one experiment at a time Consider we start out with the prior $p(\underline{\theta}) = p_{\underline{\alpha}}(\underline{\theta})$ with $\underline{\alpha}$ being the hyperparameters of the prior.

Example: For instance if our likelihood was a Binomial distribution (so the samples would consist of counts of successes in N trials) and we would model the only parameter of the Binomial distribution, the success probability μ , we could use a Beta prior, so that posterior and prior would be of the same distributional form (conjugacy). If we start out without knowledge, we could use a uniform prior, so a Beta distribution with $\alpha = \beta = 1$.

Consider X is new data, the update to the prior is

$$p^{(n)}(\underline{\theta}|X) = \frac{p(X|\underline{\theta})p^{(n-1)}(\underline{\theta})}{h(X)} \quad (893)$$

MAP as a special case of Bayesian inference Maximizing the posterior yields the MAP estimate - the mode of the posterior distribution

$$\hat{\underline{\theta}}_{\text{MAP}} = \underset{\underline{\theta}}{\operatorname{argmax}} p(\underline{\theta} | \underline{x}_1, \underline{x}_2, \dots, \underline{x}_N) \stackrel{\text{Bayes}}{=} \underset{\underline{\theta}}{\operatorname{argmax}} \frac{p(\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N | \underline{\theta}) p_{\alpha}(\underline{\theta})}{h(\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N)} \quad (894)$$

which based on

- i.i.d. assumption
- seeing that the evidence $h(\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N)$ is independent of $\underline{\theta}$

simplifies to

$$\begin{aligned} \hat{\underline{\theta}}_{\text{MAP}} &= \underset{\underline{\theta}}{\operatorname{argmax}} \frac{p_{\alpha}(\underline{\theta}) \prod_{i=1}^N p(\underline{x}_i | \underline{\theta})}{h(\underline{x}_1, \underline{x}_2, \dots, \underline{x}_N)} \\ &= \underset{\underline{\theta}}{\operatorname{argmax}} \left(p_{\alpha}(\underline{\theta}) \prod_{i=1}^N p(\underline{x}_i | \underline{\theta}) \right) \\ &= \underset{\underline{\theta}}{\operatorname{argmax}} \left(\log p_{\alpha}(\underline{\theta}) + \sum_{i=1}^N \log p(\underline{x}_i | \underline{\theta}) \right) \end{aligned} \quad (895)$$

Pros and cons of Bayesian inference See table 28.

Note on the subjectivity of the prior: In fact the entire model is subjective - both likelihood and prior. In the famous George Box quote: »all models are wrong, but some are useful« - acknowledging that statistical models always fall short of the complexities of reality but can still be useful.

Bayesian inference for linear regression with normally distributed residuals Consider we have a dataset $\{\underline{x}_i, y_i\}_{i=1}^N$ and a linear model

$$y_i = \underline{\beta}^T \underline{x}_i + \epsilon_i \quad (896)$$

with i.i.d. residuals $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

Question: What is the posterior distribution of the model parameters $\underline{\beta}$ and σ^2 given the sample?

Let us start with writing down the posterior (not denoting the fixed \underline{X} (the collection of all

³³In the sense that the model is not prejudice-free but has previous knowledge.

³⁴This is a bias-variance trade-off: the stronger the bias, the less flexible the model but the less it will also vary for different samples.

Pros	Cons
<ul style="list-style-type: none"> Using previous knowledge: The prior is a <i>bias</i>³² regularizing the model parameters (in the sense that a low data setting will not bring forth unrealistic parameters just by chance) Uncertainty quantification: The full posterior distribution can be used to quantify the uncertainty in the model parameters (a flat posterior means we have little information about the parameters, a peaked one means we have a lot of information) Hypothesis testing: Based on the full posterior distribution, we can test hypotheses about the model parameters Updating: The prior can be updated with new evidence 	<ul style="list-style-type: none"> Subjectivity of the prior: Danger if prior is not reliable or incorrect³³ Computational cost: The full posterior distribution is often hard to calculate, for instance the evidence integral is often intractable³⁴

Table 28: Pros and cons of Bayesian inference

\underline{x}_i) explicitly)

$$\begin{aligned}
 \text{posterior} &= p(\underline{\beta}, \sigma^2 | \underline{y}) \\
 &= \frac{p(\underline{y} | \underline{\beta}, \sigma^2) p(\underline{\beta}, \sigma^2)}{h(\underline{y})} \\
 &= \frac{p(\underline{y} | \underline{\beta}, \sigma^2) p(\sigma^2) \cdot p(\underline{\beta} | \sigma^2)}{h(\underline{y})} = \frac{\text{likelihood} \cdot \text{priors}}{\text{evidence}}
 \end{aligned} \tag{897}$$

The likelihood is normal as we assumed normal residuals, so

$$p(\underline{y} | \underline{\beta}, \sigma^2) = (2\pi)^{-N/2} \left| \sigma^2 \underline{\underline{1}} \right|^{-1/2} \exp \left(-\frac{1}{2} (\underline{y} - \underline{\underline{X}} \underline{\beta})^T (\sigma^{-2} \underline{\underline{1}}) (\underline{y} - \underline{\underline{X}} \underline{\beta}) \right) \tag{898}$$

Note: For a Gaussian likelihood (without σ^2 or μ being known a priori) the prior $p(\underline{\beta} | \sigma^2)$ can be chosen to be Gaussian as well, and the prior $p(\sigma^2)$ can be chosen to be an inverse Gamma distribution (the total prior is their product), so that the posterior is of the same distributional form as the prior (conjugacy).

The prior for $\underline{\beta}$ is the Gaussian

$$\text{Gaussian prior on } \underline{\beta} : p(\underline{\beta} | \sigma^2) = (2\pi)^{-p/2} \left| \sigma^{-2} \underline{\underline{\Lambda}}_0 \right|^{1/2} \exp \left(-\frac{1}{2} (\underline{\beta} - \underline{\beta}_0)^T (\sigma^{-2} \underline{\underline{\Lambda}}_0) (\underline{\beta} - \underline{\beta}_0) \right) \quad (899)$$

with

- $\underline{\beta}_0$ being the mean of the prior of $\underline{\beta}$
- $\underline{\underline{\Lambda}}_0$ being the precision matrix (inverse covariance matrix) of the prior of $\underline{\beta}$
- σ^2 being only a parameter here, convenient for later calculations

The prior for σ^2 is the inverse Gamma distribution

$$\text{Inverse - Gamma prior on } \sigma^2 : p(\sigma^2) = \frac{\kappa^\alpha \sigma^{2(-\alpha-1)}}{\Gamma(\alpha)} e^{-\kappa/\sigma^2} \quad (900)$$

Where the priors have been chosen so that the posterior

$$\begin{aligned} \text{posterior} &= \frac{\text{likelihood} \cdot \text{priors}}{\text{evidence}} \\ &= \frac{\text{gaussian} \cdot \text{gaussian} \cdot \text{inverse gamma}}{\text{evidence}} \\ &= \text{gaussian} \cdot \text{inverse gamma} \end{aligned} \quad (901)$$

is of the same distributional form as the prior (conjugacy) (the Gaussian of the likelihood and the Gaussian of the prior for $\underline{\beta}$ can be combined to one Gaussian).

Update rules for the parameters from prior to posterior Without proof, let us specify

- the Gaussian distribution of $\underline{\beta}$ is updated by

$$\underline{\beta}_N = (\underline{\underline{X}}^T \underline{\underline{X}} + \underline{\underline{\Lambda}}_0)^{-1} (\underline{\underline{X}}^T \underline{y} + \underline{\underline{\Lambda}}_0 \underline{\beta}_0), \quad \text{precision matrix } \underline{\underline{\Lambda}}_N = \underline{\underline{X}}^T \underline{\underline{X}} + \underline{\underline{\Lambda}}_0 \quad (902)$$

- the inverse Gamma distribution of σ^2 is updated by

$$\alpha_N = \alpha_0 + \frac{N}{2}, \quad \kappa_N = \kappa_0 + \frac{1}{2} \left(\underline{y}^T \underline{y} + \underline{\beta}_0^T \underline{\underline{\Lambda}}_0 \underline{\beta}_0 - \underline{\beta}_N^T \underline{\underline{\Lambda}}_N \underline{\beta}_N \right) \quad (903)$$

with the expectation of σ^2 being $E[\sigma^2 | \underline{y}, \underline{\underline{X}}] = \frac{\kappa_N}{\alpha_N - 1}$

For each subsequent update with new data X , the posterior of the parameters becomes the prior for the next update.

17.4 Hypothesis Testing

Aim of hypothesis testing³⁵: We want to decide whether some data sufficiently supports a *hypothesis*.

What is a hypothesis?: A hypothesis is a statement about population (a) parameter(s) (not about a sample). We usually formulate a devil's advocate null hypothesis H_0 (no difference, as in H_0 : the drug we test had no effect, H_0 : there is no global warming, ...), under which we devise the distribution of a statistic and under this distribution check how likely it is to observe the statistic following from our sample. We are cautious and want to make sure that we have to reject the no-difference case to make our statement (e.g. H_1 : the drug has an effect).

Example setting: Consider we have a dataset $D = \{x_i, y_i\}_{i=1}^N$ and assume a linear model

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2) \quad (904)$$

anthropogenic CO₂ in the atmosphere x_i , global temperature y_i

with the devil's advocate $H_0 : \beta_1 = 0$ (climate change is a hoax). Neglecting H_0 means there is an impact of CO₂ on temperature, a *one-sided* can give clues about the direction of the impact.

Problem: Null hypothesis significance testing is set up in a way that we crunch all our data into a single **accept-reject decision** with a *p-value*, where one who lacks integrity might *p-hack* on. Bayesian statistics on the other hand gives a full distribution over the parameters, but here a problem lies in the subjectivity of the prior.

17.4.1 Basic terms of statistical hypothesis testing

Let us first introduce basic terms of hypothesis testing.

17.4.1.1 Around the hypothesis

See table 29.

³⁵More specifically null hypothesis significance testing.

Term	Short explanation
Statistical hypothesis	Statement about the parameters of a model describing a population, e.g. the parameters in a linear model
Test statistic T	Generally any value calculated from a sample
Simple hypothesis	Hypothesis specifying the population distribution completely
Composite hypothesis	Hypothesis not specifying the population distribution completely.
Null hypothesis H_0	The statement tested in the statistical significance test, usually a statement of <i>no-effect</i> or <i>no difference</i> , e. g. COVID19 measures had no effect on the virus's reproduction rate.
Alternative hypothesis H_1	Against the null hypothesis H_0 (mutually exclusive statements, often exhaustive) the alternative hypothesis is tested and accepted when the null hypothesis is rejected; usually consistent with the research hypothesis; e. g.: COVID19 measures had an effect on the virus's reproduction rate.

Table 29: Basic terms around the hypothesis

Term	Short explanation
Rejection region / critical region	Set of values of the test statistic t for which H_0 is rejected (this region is derived under H_0)
Acceptance region	Set of t -values where H_0 is accepted
Critical value	Boundary / boundaries of the acceptance region

Table 30: Basic terms around the decision-making in hypothesis tests

17.4.1.2 Decision-making

See table 32.

17.4.1.3 Validity of the decision

See table 32.

Term	Short explanation
Significance level α	Probability of rejecting the null hypothesis H_0 given it is true (type 1 error). α is a parameter we choose and set the rejection region(s) accordingly.
p -value	Based on the distribution of the test statistic devised under H_0 it is the probability of observing a value at least as extreme as t in our sample. p is used to check if we are in the rejection region.
Statistical significance	If $p \leq \alpha$ we reject H_0 at the α significance level. So the chance that this sample was brought forth under the <i>no-difference</i> assumption is less than α .
Type 1 error, false positive (<i>rejection is positive</i>)	Rejecting H_0 when it is true
Size of the test	Probability of a type 1 error, α
Power of a test, true positive	Probability of correctly rejecting H_0 when it is false, it is the probability of the rejection region under the <i>true</i> distribution

Table 31: Basic terms around the decision-making in hypothesis tests

Term	Short explanation
Exact test	The derivation of the distribution of the test statistic only depends on H_0 being true. So if H_0 is true, we will by chance reject on average <i>exactly</i> the proportion α of tests.
Most powerful test	The test with the highest power for a given size α

Table 32: Basic terms around tests

17.4.1.4 Statements on tests

See table ??.

17.4.2 Basic Null Hypothesis Significance Testing Procedure

Idea: Based on a null hypothesis come up with a test statistic which distribution can be derived under H_0 and check if the observed statistic can reasonably be assumed to have been brought forth under H_0 .

1. We want to test a statement about model parameters for a population. We therefore

state the null-hypothesis H_0 and the alternative hypothesis H_1 in terms of these.

$$\begin{aligned} \text{e.g. the two-sided } H_1 : \beta_1 \neq 0, & \quad H_0 : \beta_1 = 0 \\ \text{e.g. the one-sided } H_1 : \beta_1 > 0, & \quad H_0 : \beta_1 \leq 0 \end{aligned} \tag{905}$$

Note: For the one-sided test we in effect mostly also use the *no-difference* $H_0 : \beta_1 = 0$ (the most *extreme* version of $H_0 : \beta_1 \leq 0$) to devise the test statistic and then based on *where* we reject^a can make a 1-sided statement.

^aOn the left or right side of the distribution. The test statistic is 1-dimensional, while our statement might be concerned with multiple parameters.

2. Define a test statistic T which can be calculated from our sample (and for which we can find a distribution)

$$\text{e.g. } t = \hat{\beta}_1^{\text{MLE}} \quad \text{for the linear model} \tag{906}$$

3. Derive the distribution of the test statistic under H_0 (and possibly further assumptions) (or construct it empirically via *bootstrapping*)

$$\text{e.g. } t \sim \mathcal{N}(0, \sigma^2 (\underline{\underline{X}}^T \underline{\underline{X}})^{-1}) \quad \text{for the linear model, e.g. with } \beta_0 \text{ marginalized out} \tag{907}$$

4. Define a significance level α (e.g. $\alpha = 0.05$)
5. Partition the distribution of the test statistic T into rejection and acceptance region. The rejection region is chosen to have probability α under H_0 .
6. Compute the observed value t_{obs} of the test statistic T from the sample
7. If t_{obs} is in the rejection region, reject H_0 at the α significance level, otherwise accept H_0 .

Steps 5 to 7 can alternatively be formulated as

1. Compute the *p*-value of the observed test statistic t_{obs} , i.e. the probability of obtaining a sample at least as extreme as the observed sample under H_0 .
2. Reject H_0 in favor of H_1 if $p \leq \alpha$.

The testing procedure is illustrated in figure 147.

Goal: Based on some sample decide on statement on population parameters.

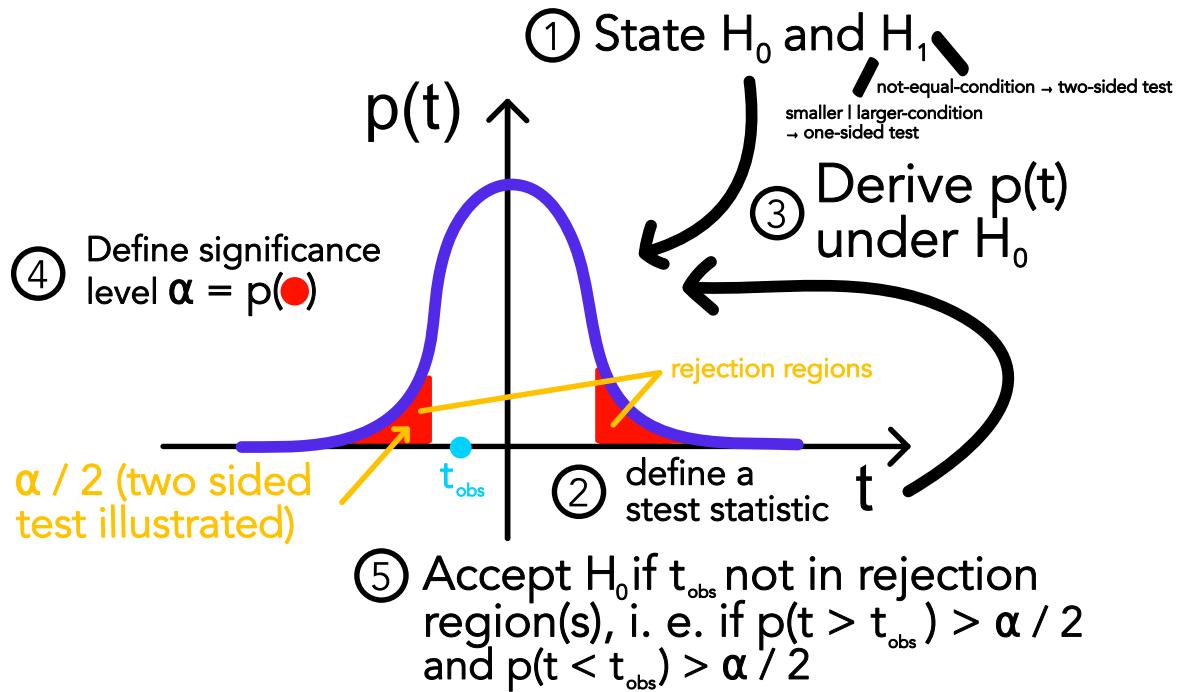


Figure 147: Illustration of the hypothesis testing procedure

Rejection in a two-sided test: In a two-sided test we construct equally likely rejection regions on both sides of the distribution of the test statistic. Therefore, we reject H_0 if

$$p(t \geq t_{\text{obs}} | H_0) \leq \frac{\alpha}{2}, \quad p(t \leq t_{\text{obs}} | H_0) \leq \frac{\alpha}{2} \quad (908)$$

Formally the rejection region is given by (let $F(t)$ be the CDF of $p(t)$)

$$R = \left[-\infty, F^{-1}\left(\frac{\alpha}{2}\right) \right] \cup \left[F^{-1}\left(1 - \frac{\alpha}{2}\right), \infty \right] \quad (909)$$

17.4.2.1 Type I and type II error

- **Type I error:** Rejecting H_0 when it is true, by design the probability of a type I error is α
- **Type II error:** Not rejecting H_0 when it is false, the probability of a type II error is β based on the true distribution of the test statistic

The errors are illustrated in figure 148.

The most powerful test - the one with the highest probability of correctly rejecting H_0 at a

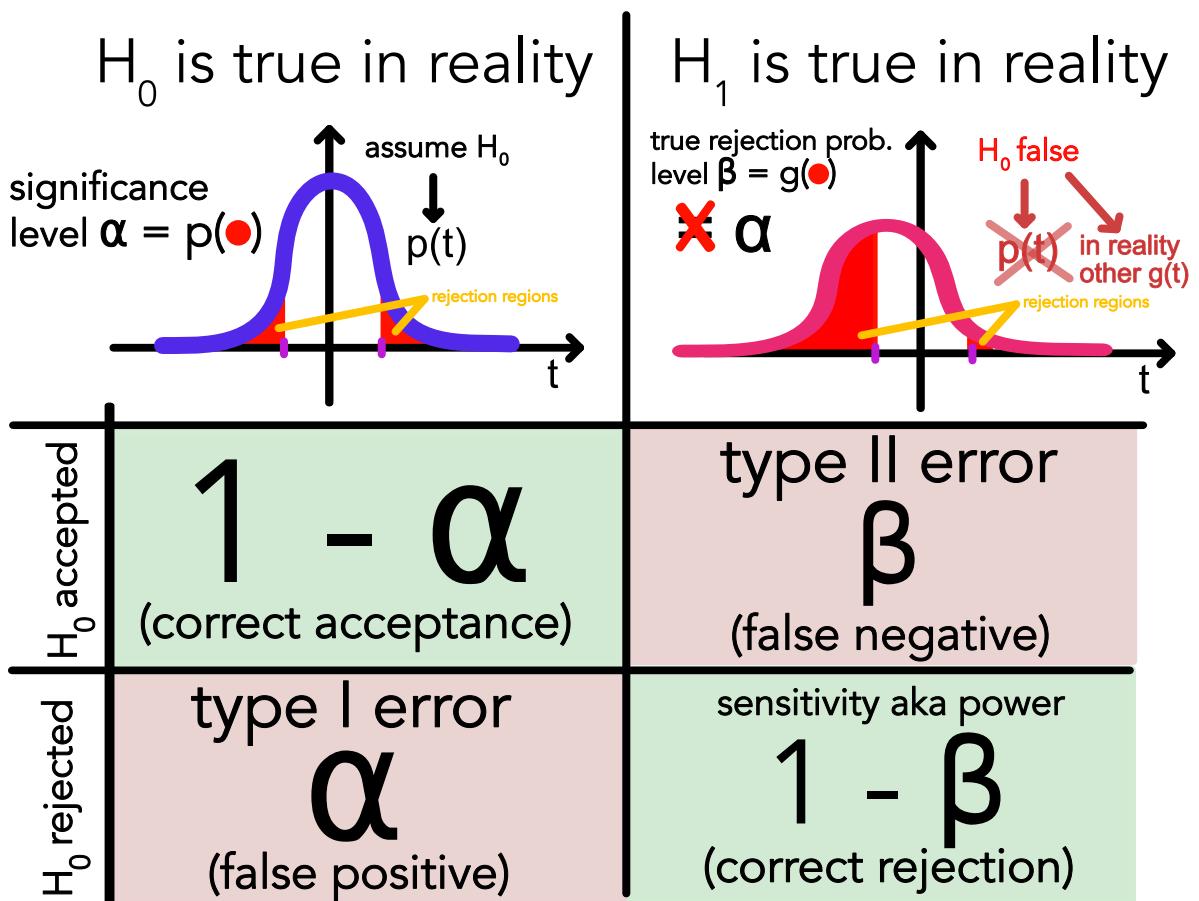


Figure 148: Truth table of hypothesis testing.

given α - is chosen based on the Neyman-Pearson lemma³⁶.

17.4.2.2 Bayesian perspective on hypothesis testing

In Bayesian statistic we have the full posterior of the parameter of interest, so we can directly make statements about the parameters.

17.4.3 Overview on specific test procedures

- **Exact tests:** The underlying probability distribution is exactly known under H_0 (e.g. the binomial distribution for a coin flip). No further parameters need to be estimated → *non-parametric*.
- **Asymptotic tests:** Based on the central limit theorem, the distribution of the test statistic is derived under H_0 . Parameters are estimated from the data as necessary.
- **Bootstrapping:** The distribution of the test statistic is derived empirically (empirical

³⁶Roughly it says that a likelihood ratio test is the way to go for testing simple hypotheses against each other.

distribution function) by resampling the data.

17.4.4 Hypothesis and statistics for typical tests

Consider table 33.

Test For	Null Hypothesis (H_0)	Test Statistic	Distribution	Use When
Population mean (μ) if variance σ^2 is known	$\mu = \mu_0$	$\frac{(\bar{x} - \mu_0)}{\sigma/\sqrt{n}}$	Z	Normal distribution or $n > 30$; σ known
Population mean (μ) if variance σ^2 must be estimated	$\mu = \mu_0$	$\frac{(\bar{x} - \mu_0)}{s/\sqrt{n}}$	t_{n-1}	$n < 30$, and/or σ unknown
Difference of two means ($\mu_1 - \mu_2$) with σ_1^2, σ_2^2 known	$\mu_1 - \mu_2 = 0$ (Test for $= c$ by replacing the 0 accordingly.)	$\frac{(\bar{x}_1 - \bar{x}_2) - 0}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$	Z	Both normal distributions, or $n_1, n_2 \geq 30$; σ_1, σ_2 known
Difference of two means ($\mu_1 - \mu_2$) with σ_1^2, σ_2^2 estimated	$\mu_1 - \mu_2 = 0$	$\frac{(\bar{x}_1 - \bar{x}_2) - 0}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$	t distribution with $df =$ the smaller of $n_1 - 1$ and $n_2 - 1$	$n_1, n_2 < 30$; and/or σ_1, σ_2 unknown
Mean difference μ_d (of paired data $\{x_i^{(1)}, x_i^{(2)}\}_{i=1}^N, d_i = x_i^{(1)} - x_i^{(2)}$)	$\mu_d = 0$	$\frac{(\bar{d} - \mu_d)}{s_d/\sqrt{n}}$	t_{n-1}	$n < 30$ pairs of data and/or σ_d unknown

Table 33: Hypothesis and test statistic for typical tests

17.4.5 Simple example for an application case of hypothesis testing

Consider survival times of patients with breast cancer (sample 1) and stomach cancer (sample 2).

$$\begin{aligned} D^{(1)} &= (107, 353, 1764, 667, 990, 78, 667, 44, 9, 27), \quad n_1 = 10 \\ D^{(2)} &= (374, 253, 812, 246, 95, 367, 251, 309, 594, 826, 593, 97), \quad n_2 = 12 \end{aligned} \quad (910)$$

We will work with the logged lifetimes $X^{(1)}, X^{(2)}, x_j^{(k)} = \log d_j^{(k)}$ (see e.g. **over04**) and present kernel density estimates of both the original and log-transformed data in figure 149. Note that in the sample 1 we have an outlier at 1764 which strongly pulls the mean of the non-logged data to higher values. The log-transforms crunches down especially the outlier, shifting the mean to lower values.

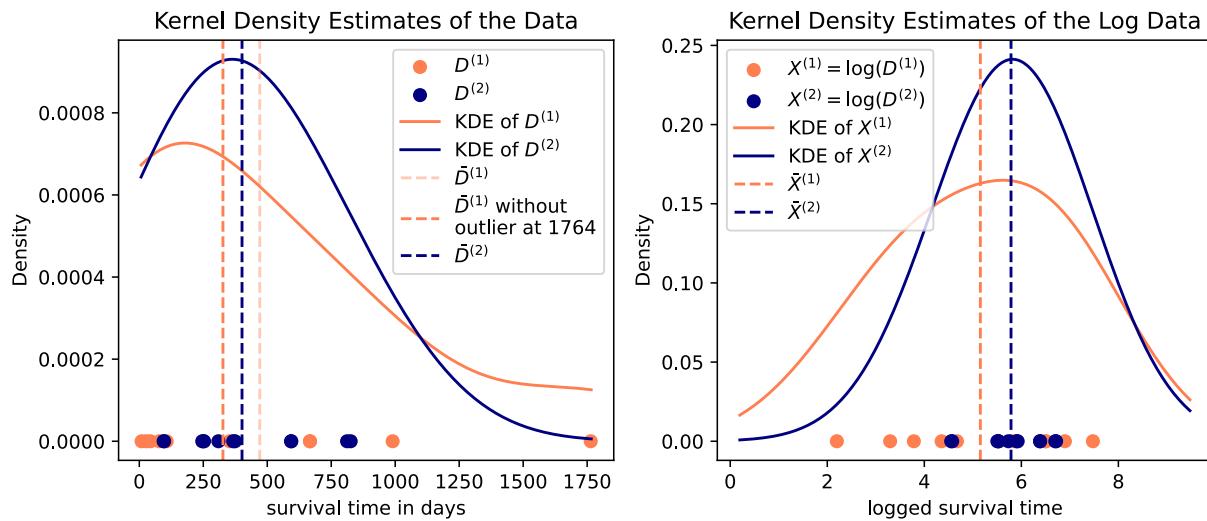


Figure 149: Kernel Density Estimates of the original and log-transformed data.

Question: Do patients with stomach cancer live significantly longer than patients with breast cancer?

As $\bar{X}^{(1)}$ is smaller than $\bar{X}^{(2)}$ (see figure 149), we will consider $\mu_2 - \mu_1$.

Null hypothesis: Therefore our *no-difference hypothesis* is: $H_0 : \mu_2 - \mu_1 \leq c$, some constant c .

Intuition: While $\bar{X}^{(2)} - \bar{X}^{(1)} > 0.25$ we cannot reject H_0 fundamentally as of the large stray in the distributions (see figure 149 again).

Test statistic: The basic reasoning here is that we want an under H_0 centered test distribution, which takes into consideration that the scale on which means are to be compared is

given by the standard error of the means (see again figure 149, if the distributions would be more peaked, maybe H_0 could be rejected).

$$t_{\text{obs}} = \frac{\bar{X}_{\text{sample}}^{(2)} - \bar{X}_{\text{sample}}^{(1)} - c}{S_p \sqrt{\frac{1}{n^{(1)}} + \frac{1}{n^{(2)}}}}$$

$$S_p = \sqrt{\frac{(n^{(1)} - 1) (S^{(1)})^2 + (n^{(2)} - 1) (S^{(2)})^2}{n^{(1)} + n^{(2)} - 2}} \quad (911)$$

The next step would be to calculate (derive / bootstrap) the distribution of the test statistic and see where the observed value falls. But before we get to this, we consider *exact tests*.

17.4.6 Exact tests

Here based on H_0 the distribution of the test statistic is exactly known.

17.4.6.1 Sign test - testing the direction of change

Consider **paired** observations $\{x_i \in X, y_i \in Y\}_{i=1}^N$, e.g. Covid19 infection rates before (x_i) and after (y_i) some restriction measures across multiple countries $i = 1, \dots, N$.

Sign testing procedure

1. We want to test if observations in X are larger than those in Y (e.g. Covid-measures decreased infection rates) (or vice versa), ties are ignored / disregarded.
2. Consider the statistic $T_i = \text{sign}(x_i - y_i)$ only based on the sign of the difference. With this

$$H_0 : E[T] = 0, \quad H_1 : E[T] > 0 \quad (912)$$

or equivalently $H_0 : p(T = +1) = 0.5, \quad H_1 : p(T = +1) > 0.5$. Consider the test statistic

$$k_0 = \sum_{i=1}^N T_i + 1 \quad (913)$$

counting the number of positive signs.

3. Under H_0 we follow $k_0 \sim \text{Binomial}(N, 0.5)$
4. We choose a significance level $\alpha = 0.05$
5. We want to reject H_0 on the right side so towards $H_1 : E[T] > 0$. Therefore the p -value

of observing a k more extreme than k_0 is (under H_0)

$$p(k \geq k_0 | H_0) = \sum_{i=k_0}^N \binom{N}{i} \left(\frac{1}{2}\right)^k \left(\frac{1}{2}\right)^{N-k} = \sum_{i=k_0}^N \binom{N}{i} \left(\frac{1}{2}\right)^N \quad (914)$$

6. We accept H_1 if $p \leq \alpha$

Note: In the one-sided scenario the »we neglect the null hypothesis and therefore accept the alternative« is not a nice formulation, if H_0 and H_1 are not exhaustive.

Problem of confounding factors: Note that we strictly only make statements about data. E.g. the viral load decreased between measurements. If we want to make a statement like giving the drug caused the decrease, we need to control for confounding factors. For instance the reason for the change in viral load could also be that one measurement was in the morning, the other in the evening.

Pro and con - sign test

- **Pro:** as in general in exact tests, no further assumptions about the distribution of the statistic are done, except that H_0 holds
- **Con:**
 - only sign information is used, not how large the differences are, not powerful, high β
 - other relations like correlations between x_i and y_i are not considered

17.4.6.2 Mann-Whitney U test (aka Wilcoxon rank-sum test) - no strict pairing necessary

Aim: Test if two independent samples come from the same distribution. It is an alternative to the *unpaired two-sample t-test*. **Example Question:** Do marks of boys and girls in a test differ? **Concept:** The test is done base on ranks-sums. Order all observations from both samples in ascending order and assign ranks to them³⁷. Then sum the ranks of the observations from each sample respectively. One can find an exact distribution of the test statistic under H_0 .

³⁷For ties both get the same rank halfway between the next lower and higher rank.

17.4.7 Asymptotic test

17.4.7.1 Central limit theorem

Let $X_i, i = 1, \dots, N$ be independent random variables with variance σ^2 and finite mean $\mu := E[X_i] < \infty$. Then

$$\lim_{N \rightarrow \infty} \frac{\frac{1}{N} \sum_i (X_i - \mu)}{\sigma / \sqrt{N}} \sim \mathcal{N}(0, 1) \quad (915)$$

so when we repeatedly sample from some distribution (e.g. an exponential one) and calculate the mean of those samples, then those means will be normally distributed, as e.g. shown in figure 150.

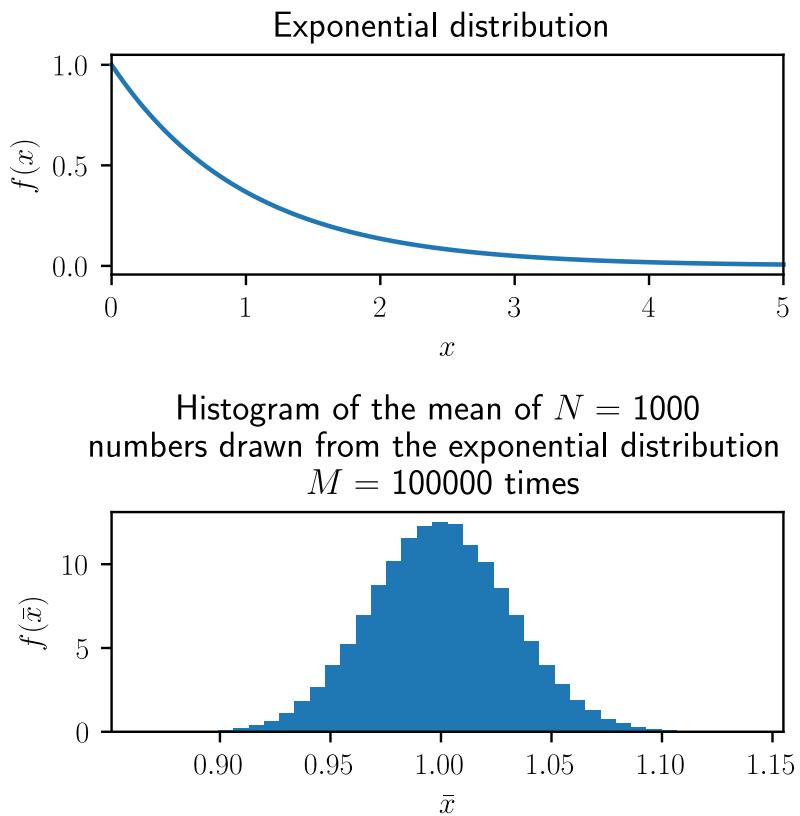


Figure 150: Central limit theorem for the exponential distribution.

Note: If μ and σ are unknown, this - in the large N limit - still holds true when the estimates of μ and σ are used, even though the estimate of σ is (potentially) biased, by *Slutskys theorem* (for which it's enough that the estimate of σ converges in probability to σ). For smaller N we have to be more careful.

Usage in testing: Being able to estimate the sampling distribution of a mean just from a sample is very useful when we want to compare means.

In the following we will introduce some important asymptotic distributions and asymptotic tests.

17.4.7.2 Important distributions in testing I: χ^2 distribution

The sum of squares of a normally distributed variable $Z_i \sim \mathcal{N}(0, 1)$ ($\{Z_i\}_{i=1}^N$ i.i.d.) follows

$$\sum_{i=1}^K Z_i^2 \sim \chi^2(k) \quad (916)$$

where N is the number of degrees of freedom (in the sample not the parameters), here the number of independent random variables. The χ^2 distribution is shown in figure 151.

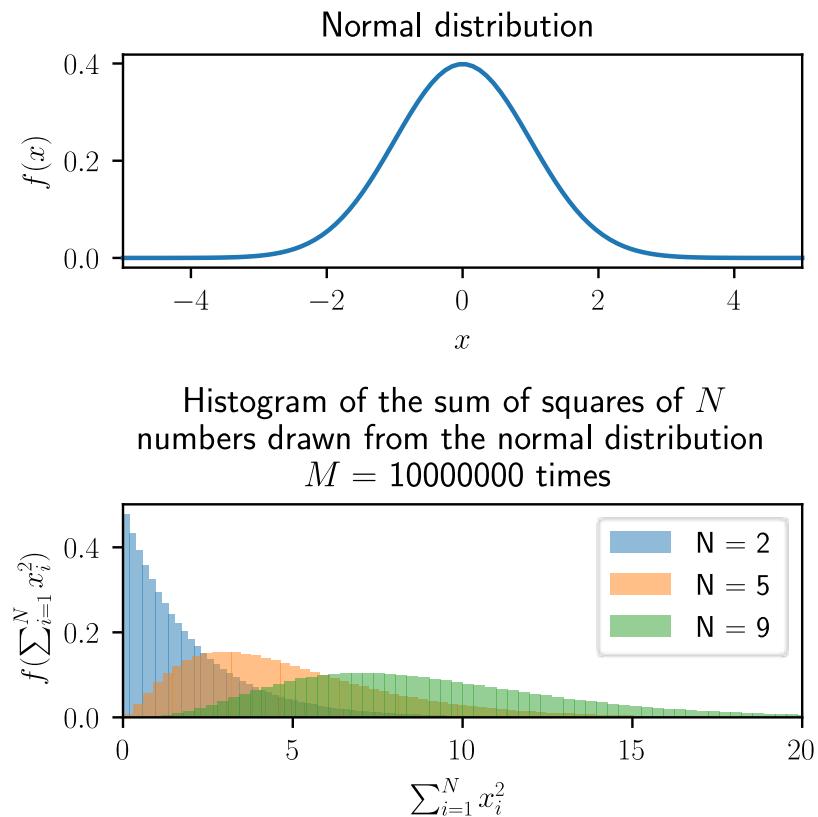


Figure 151: The χ^2 distribution for different degrees of freedom (numbers drawn per sample). For large N we should approach a normal distribution again.

17.4.7.3 Important distributions in testing II: t -distribution

The t_N distribution is defined on the ratio

$$t_N = \frac{z}{\sqrt{\chi_N^2/N}}, \quad z \sim \mathcal{N}(0, 1) \quad (917)$$

where z and χ_N^2 are independent. The t -distribution is shown in figure 152.

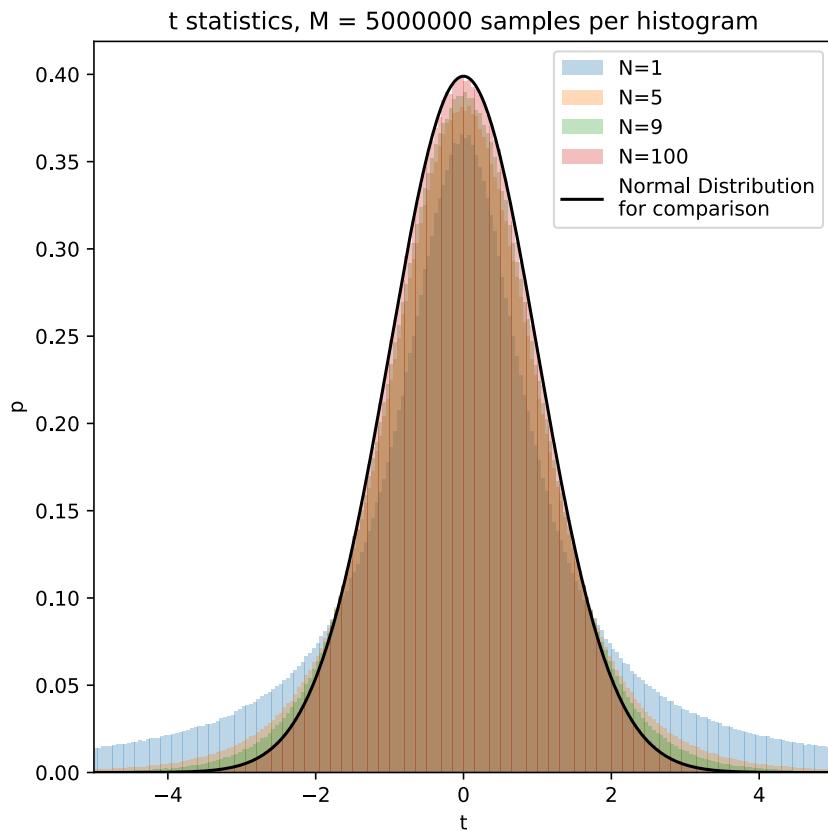


Figure 152: The t -distribution for different degrees of freedom (numbers drawn per sample). For large N we should approach a normal distribution again.

17.4.7.4 The Chi-Square Test - test on counts

Setting Consider an experiment of counts of categories, i.e. when tossing balls at boxes assuming non-changing probabilities of the categories (boxes) and independent throw.

Hypothesis We make a hypothesis for all the probabilities of the categories $p_i, i = 1, \dots, k$. For n total throws, we expect (binomial distribution) $E(n_i) = np_i$ hits per box in n throws (counts per category).

Test statistic The test statistic is the sum of squares of the differences between the observed and expected counts, normalized by the expected counts.

$$X^2 = \sum_{i=1}^k \frac{[n_i - E(n_i)]^2}{E(n_i)} = \sum_{i=1}^k \frac{[n_i - np_i]^2}{np_i} \underset{N \text{ large}}{\sim} \chi_{df}^2 \underset{N \text{ large, approx.}}{\sim} \chi_{k-1}^2 \quad (918)$$

Why does this follow a χ^2 distribution with $k - 1$ degrees of freedom?: For large n , n_i will be approximately normally distributed, while n and p_i are just constants. While we sum over k categories, we have the constraint that the n_i sum up to n , so one degree of freedom is lost (equivalently $\sum p_i = 1$).

Given e.g. tabulated data of the χ^2 distribution we can, based on the observed X^2 value, calculate the p -value and decide if we reject H_0 .

17.4.7.5 Students t -test

We finally come back to tests on means, as introduced for the example of the survival times of breast and stomach cancer patients (a *two-sample t-test*).

Aim in t-tests:

- 1-sample t-test: Test a sample might come from a distribution with mean μ_0
- 2-sample t-test: Test whether two distributions come from a distribution with the same mean, e.g. H_0 : breast and stomach cancer patients have the same survival times.

Note: Formulations where we test on a certain difference between means follow analogously.

One-sample t-test Consider we have measured $\{x_i\}, i = 1, \dots, N$ and want to test against a known means μ_0 .

$$H_0 : \mu = \mu_0, \quad H_1 : \mu \neq \mu_0, \quad \bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad (919)$$

We use the test statistic

$$t = \frac{\bar{x} - \mu_0}{\hat{\sigma}_{\bar{x}}} \sim t_{N-1}, \quad \text{std. of mean } \hat{\sigma}_{\bar{x}} = \frac{s}{\sqrt{N}}$$

(920)

corrected (but biased) sample std. $s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$

Note: While for large N we have a normal distribution by Slutkys theorem, for lower N we have to mind that $\hat{\sigma}_{\bar{x}}$ is an estimate.

The distribution according to t_{N-1} can be seen from

$$t = \frac{\bar{x} - \mu_0}{\hat{\sigma}_{\bar{x}}} = \frac{(\bar{x} - \mu_0) / (\sigma / \sqrt{N})}{\hat{\sigma}_{\bar{x}} / (\sigma / \sqrt{N})} = \frac{\overbrace{(\bar{x} - \mu_0) / (\sigma / \sqrt{N})}^{\sim \mathcal{N}(0,1)}}{\sqrt{\underbrace{\frac{1}{N-1} \sum_{i=1}^N \left(\frac{x_i - \bar{x}}{\sigma} \right)^2}_{\sim \sqrt{\chi_{N-1}^2 / (N-1)}}}} \sim t_{N-1} \quad (921)$$

Why is it $N - 1$? The reasoning for why it is t_{N-1} here is the same as for the reasoning of $N - 1$ in the Bessel correction. While we have N independent observations in the sample we only have $N - 1$ independent residuals, $x_i - \bar{x}$ as they must sum up to 1 (by definition of the sample mean).

Two-sample t-test Consider we have two independent samples X_1 and X_2 of (possibly different) sizes N_1 and N_2 with means

$$\bar{x}_1 = \frac{1}{N_1} \sum_{i=1}^{N_1} x_{1i}, \quad \bar{x}_2 = \frac{1}{N_2} \sum_{i=1}^{N_2} x_{2i} \quad (922)$$

We want to test if the means **of the populations they come from** μ_1, μ_2 are significantly different (or e.g. larger than a threshold).

$$H_0 : \mu_1 = \mu_2, \quad H_1 : \mu_1 \neq \mu_2 \quad (923)$$

For unequal sample sizes and similar variances, we get

$$t = \frac{(\bar{x}_1 - \mu_1) - (\bar{x}_2 - \mu_2)}{\hat{\sigma}_{\bar{x}_1 - \bar{x}_2}} \underset{H_0: \mu_1 = \mu_2}{\underset{\hat{\sigma}_{\text{pool}}}{\approx}} \frac{\bar{x}_1 - \bar{x}_2}{\hat{\sigma}_{\text{pool}} \sqrt{\frac{1}{N_1} + \frac{1}{N_2}}} \quad (924)$$

where the pooled standard deviation $\hat{\sigma}_{\text{pool}}$ is given by

$$\hat{\sigma}_{\text{pool}} = \sqrt{\frac{(N_1 - 1) \hat{\sigma}_1^2 + (N_2 - 1) \hat{\sigma}_2^2}{N_1 + N_2 - 2}}, \quad \left(\begin{array}{l} \text{makes sense for} \\ \frac{1}{2} < \frac{\hat{\sigma}_1^2}{\hat{\sigma}_2^2} < 2 \end{array} \right) \quad (925)$$

where $\hat{\sigma}_1^2, \hat{\sigma}_2^2$ are the unbiased estimates of the population variances.

Why this complicated denominator and t-distribution? The idea is that means have to be compared on the scale of their standard errors, as illustrated in figure 153.

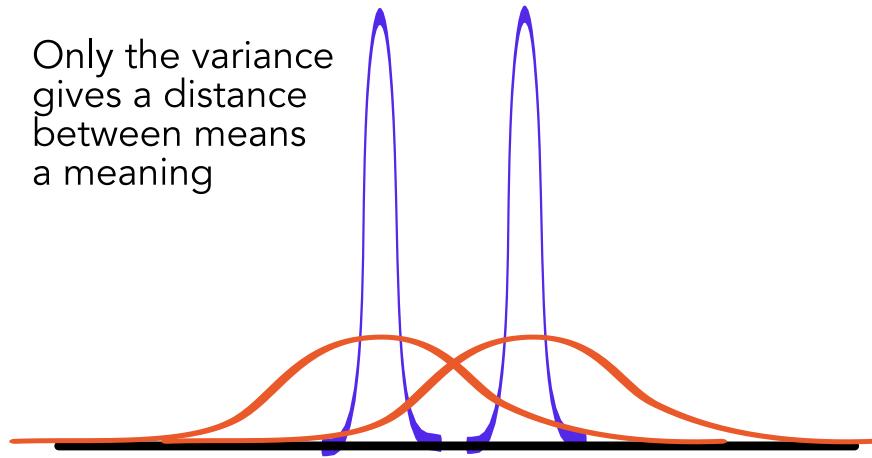


Figure 153: It is not the absolute difference between means that counts.

17.4.7.6 Important distributions in testing III: F-distribution

Let $\chi_1^2 \sim \chi_{N_1}^2$ and $\chi_2^2 \sim \chi_{N_2}^2$ be independent. Then the ratio

$$F = \frac{\chi_1^2/N_1}{\chi_2^2/N_2} \sim F_{N_1, N_2} \quad (926)$$

of these two χ^2 distributed random variables divided by their degrees of freedom follows the F-distribution.

17.4.7.7 The F-test - test on variances

Assuming two independent samples X_1 and X_2 from two populations (e.g. test scores of boys and girls), which we assume to come from populations with normal distribution with means μ_1, μ_2 and variances σ_1^2, σ_2^2 .

Based on the samples

$$\{x_{1i}\}_{i=1}^{N_1}, \quad \{x_{2i}\}_{i=1}^{N_2} \quad (927)$$

we want to test if the population variances can reasonably be assumed to be equal, i.e. $H_0 : \sigma_1^2 = \sigma_2^2$.

The test statistic is given by

$$F = \frac{s_1^2}{s_2^2} \sim F_{N_1-1, N_2-1}, \quad s_1^2 = \frac{1}{N_1-1} \sum_{i=1}^{N_1} (x_{1i} - \bar{x}_1)^2, \quad s_2^2 = \frac{1}{N_2-1} \sum_{i=1}^{N_2} (x_{2i} - \bar{x}_2)^2 \quad (928)$$

Example of comparing two models based on explained variance Consider data

$$D = \{y_i, \underline{x}_i\}, \quad \underline{x}_i = \begin{pmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{pmatrix} \quad (929)$$

on which we try a full and reduced model

$$\begin{aligned} \text{full model (I)}: y_i &= \beta_0 + \sum_{j=1}^p \beta_j^{(I)} x_{ij} + \epsilon_i \\ \text{reduced model (II)}: y_i &= \beta_0 + \sum_{j=1}^q \beta_j^{(II)} x_{ij} + \epsilon_i, \quad q < p \end{aligned} \quad (930)$$

and make the null hypothesis

$$H_0: \beta_{q+1} = \dots = \beta_p = 0 \quad (931)$$

i.e. that the small model is sufficient for linearly explaining the data.

Now consider the model losses

$$\text{SSQ}(\underline{\beta}^{(I)}) = \sum_{i=1}^N \left(y_i - \hat{y}_i^{(I)} \right)^2, \quad \text{SSQ}(\underline{\beta}^{(II)}) = \sum_{i=1}^N \left(y_i - \hat{y}_i^{(II)} \right)^2 \quad (932)$$

Note: By choice of the residuals, $y_i - \hat{y}_i^{(I,II)} \sim \mathcal{N}(0, \sigma^2)$, so the sums of squares of the residuals $\text{SSQ}(\underline{\beta}^{(I,II)})$ follow χ^2 distributions.

Therefore the test statistic comparing the sum of squares of the residuals of the two models is given by

$$\mathcal{F} = \frac{\left(\text{SSQ}(\underline{\beta}^{(II)}) - \text{SSQ}(\underline{\beta}^{(I)}) \right) / (p - q)}{\text{SSQ}(\underline{\beta}^{(I)}) / (N - p - 1)} \sim F_{p-q, N-p-1} \quad (933)$$

$\text{SSQ}(II) - \text{SSQ}(I) \geq 0$ as the smaller model has the higher loss

17.4.7.8 Likelihood ratio test principle - comparing models by likelihood

Aim of the likelihood ratio test: We want to compare a full to a restricted model, to see, if the full more complex model should be used or if the simpler one is just fine.

So consider a model $M_{\underline{\theta}}$ with parameters $\underline{\theta} \in \Omega$ (the parameter space). The restricted model $M_{\underline{\theta}_0}$ has parameters from a limited parameter space $\Omega_0 \subset \Omega$, constrained by the null hypothesis H_0 .

Test statistic We compare the likelihood of the sample under the full and restricted model.

$$\lambda = \frac{\mathcal{L}_X(\hat{\underline{\theta}}_{\text{MLE}} \in \Omega_0)}{\mathcal{L}_X(\hat{\underline{\theta}}_{\text{MLE}} \in \Omega)} \in [0, 1] \quad (934)$$

If the null hypothesis is true, then the limitation of Ω has no effect on the likelihood

$$\text{if } H_0 : \hat{\underline{\theta}}_{\text{MLE}} \in \Omega_0 \text{ in general} \rightarrow \lambda = 1 \quad (935)$$

As we usually want a test statistic distributed around 0, we use the log-likelihood ratio

$$D := -2 \log \lambda \underset{\text{for sample size } \geq 10 \text{ as of CLT}}{\sim} \chi^2_{k-k_0}$$

k = number of parameters in the full model, k_0 = number of parameters in the restricted model

$$df = k - k_0 \# \text{parameters fixed by the null hypothesis} \quad (936)$$

Example: For instance we could have the baseline linear model

$$M_{\Omega} : y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i, \quad \beta_1, \beta_2 \in \Omega \quad (937)$$

and the null hypothesis $H_0 : \beta_2 = 0$. Then

$$M_{\Omega_0} : y_i = \beta_0 + \beta_1 x_i + \epsilon_i \rightarrow df = 1 \quad (938)$$

Note: To really approximately have a χ^2 distribution, it might make sense to bring the baseline observation to a more Gaussian form, e.g. by Box-Cox transformation, generally

$$z_i = \begin{cases} \frac{x_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \ln y_i & \text{if } \lambda = 0 \end{cases} \quad (939)$$

with λ being a parameter. Where for $\lambda = 0$ we have the log-transformation which can sometimes help to unskew distributions (stabilize the variance) (it was log-normal previously) (might also have the opposite effect).

17.5 Bootstrap methods

Uses of Bootstrap include:

- estimating the standard error of a statistic
- estimating the bias of a statistic
- estimating confidence intervals for a statistic
- estimating the distribution of a statistic for hypothesis testing

At the heart of bootstrap lies random sampling with replacement to better assess statistical estimates (assign measures of accuracy³⁸) - we use a sample as an estimate of a population and bootstrap samples from the sample as estimates of the sampling distribution. Consider we have observed a sample $X_{\text{sample}} = (x_1, \dots, x_n)$ and are interested in a population parameter θ which we can estimate as $\hat{\theta}$ from a sample. The non-parametric bootstrap method is given by

1. $\forall b \in \{1, \dots, B\}$ (where B is the number of bootstrap samples): Generate a bootstrap sample $X^{(b)} = (x_1^{(b)}, \dots, x_n^{(b)})$ by sampling from X with replacement and calculate $\hat{\theta}^{(b)}$ on the bootstrap sample.
2. This yields an empirical distribution of estimates $\hat{\theta}^{(1)}, \dots, \hat{\theta}^{(B)}$ from which we can calculate the mean $\hat{\theta}^*$ and standard error $\text{SE}(\hat{\theta}^*)$.

with the bootstrap estimate and the estimated standard error given by

$$\begin{aligned}\hat{\theta}^* &= \frac{1}{B} \sum_{b=1}^B \hat{\theta}^{(b)} \\ \hat{\text{SE}}(\hat{\theta}) &= \text{SE}(\hat{\theta}^*) = \sqrt{\frac{1}{B-1} \sum_{b=1}^B (\hat{\theta}^{(b)} - \hat{\theta}^*)^2}\end{aligned}\tag{940}$$

The bootstrap estimate of the bias $\hat{\text{bias}}(\hat{\theta}) = E[\hat{\theta}] - \theta$ of the estimator $\hat{\theta}$ is given by

$$\hat{\text{bias}}(\hat{\theta}) = \hat{\theta}^* - \hat{\theta}\tag{941}$$

³⁸For the mean e.g. a measure of accuracy is the standard error for which we have the explicit formula $\text{SE} = \sqrt{\frac{s^2}{n}}$, $\bar{X}_{\text{sample}} = \frac{1}{n} \sum_{i=1}^n x_i$, $s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{X}_{\text{sample}})^2$, $X_{\text{sample}} = (x_1, \dots, x_n)$. For most other statistics (e.g. the median) we do not have such an explicit formula. See Efron and Tibshirani, 1994, chapter 1.

Note: By the bias correction we introduce a new source of variance (the variance of the bias) which we do not account for in the standard error. Using further bootstrapping we can estimate the standard error of \bar{t} , but if $\text{bias}(\hat{t})$ is small compared to $\hat{\text{SE}}(\hat{t})$ it is safer to use \hat{t} than \bar{t} (see Efron and Tibshirani, 1994, chapter 10.6, p. 138).

17.5.1 Caveats of Bootstrap*

Beware, that bootstrap is not magic - it will not retrieve information not present in a sample or fix a bad sample. Also, the bootstrap sample is generally centered at the observed statistic, not the population parameter (e.g. \bar{X} not μ) - using bootstrap we will not improve on \bar{X} (Hesterberg, 2015, section 2.3 (also see there for exceptions)). And if there is bias (as estimated above), the bootstrap estimate is indeed not a bias-corrected estimate - if $\hat{\theta}^*$ is greater than $\hat{\theta}$, the bias corrected estimate $\bar{\theta} = \hat{\theta} - \text{bias}$ should be less than $\hat{\theta}$, and this kind of bias correction also has pitfalls³⁹ (Efron and Tibshirani, 1994, chapter 10.6, p. 138). Simply put, if from resampling a sample we get a positive bias then to get to the unbiased one we have to go "two steps back", $\hat{\theta} = \hat{\theta}^* - 2 \cdot (\hat{\theta}^* - \hat{\theta}) = 2\hat{\theta} - \hat{\theta}^*$ (eq. 10.41 in Efron and Tibshirani, 1994).

In figure 154 we illustrate that as the sample carries distributional information so do the bootstrap samples. In figure 155 we illustrate the difference between the sample estimate, bootstrap estimate and corrected estimate for a biased estimator, the standard deviation. Here the bias comes from the fact that in the calculation of the standard deviation we also estimate the mean from which the points in the sample naturally deviate less than from the true mean, so the standard deviation is (with the biased estimator) underestimated.

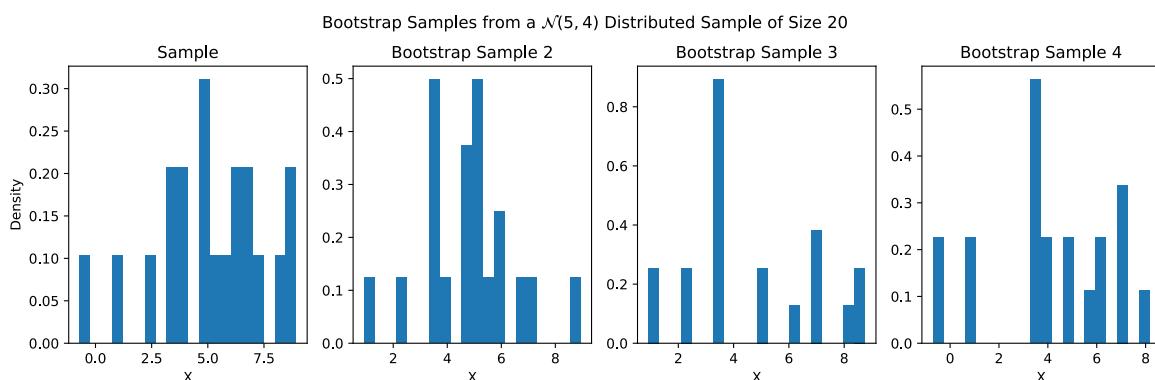


Figure 154: Bootstrap Samples from a sample.

³⁹ $\bar{\theta}$ might have a much larger standard error than $\hat{\theta}$.

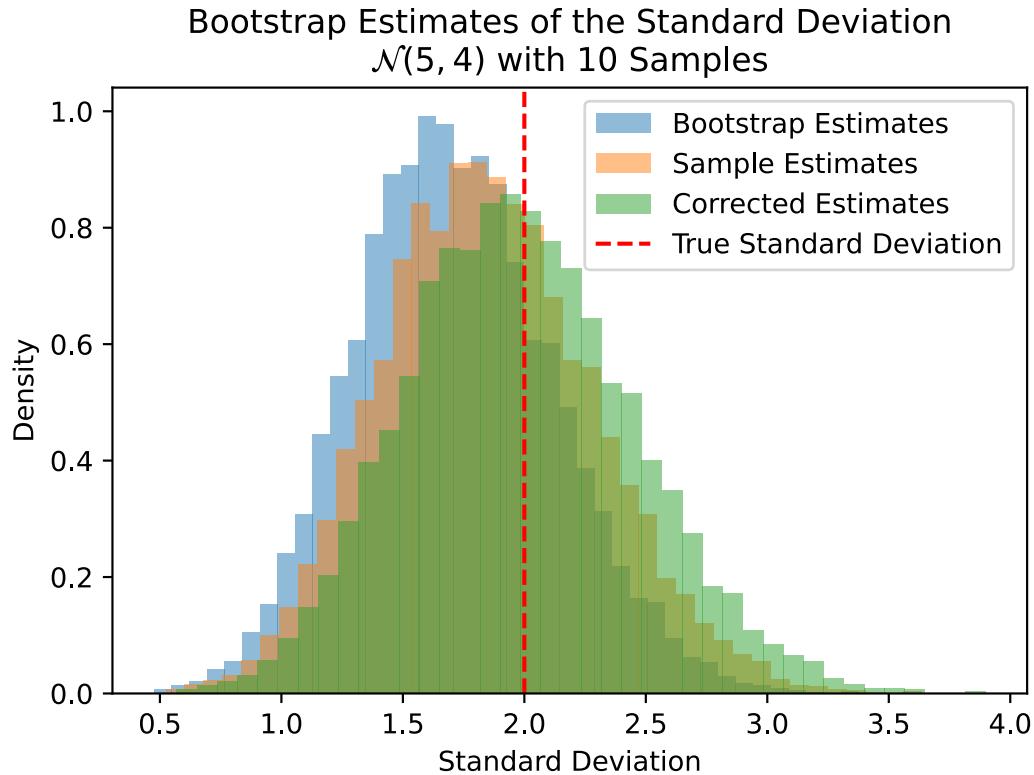


Figure 155: Illustration of the difference between the sample estimate, bootstrap estimate and corrected estimate for the biased estimate of the standard deviation.

17.5.2 Bootstrap Confidence Intervals*

Note in general that bootstrap distributions tend to be narrow on average, so the bootstrap confidence intervals often under-cover (Hesterberg, 2015).

Note: This also carries over to testing. When a bootstrap distribution is used as an alternative to a t -distribution it is usually more narrow, so the bootstrap test rejects H_0 more often than the t -test.

17.5.3 Standard Normal Bootstrap Confidence Interval

If $\hat{\theta}$ is a sample mean, then for large sample sizes, we can apply the central limit theorem

$$\frac{\hat{\theta} - \theta}{\text{SE}(\hat{\theta})} \xrightarrow{a} N(0, 1) \quad (942)$$

so based on the bootstrap estimate of the standard error $\hat{\text{SE}}(\hat{\theta})$ we can calculate a $1 - \alpha$ -confidence interval for θ as

$$\hat{\theta} \pm z_{1-\alpha/2} \hat{SE}(\hat{\theta}), \quad z_{1-\alpha/2} \text{ is the } 1 - \alpha/2 \text{ quantile of } N(0, 1) \quad (943)$$

assuming that $\hat{\theta}$ is approximately normally distributed, $\hat{\theta}$ is unbiased and $\hat{SE}(\hat{\theta})$ is a good estimate of $SE(\hat{\theta})$.

17.5.3.1 Percentile Bootstrap Confidence Interval*

Here, we use quantiles based on the empirical distribution of the bootstrap samples, so a $1 - \alpha$ -confidence interval for θ is given by

$$\begin{aligned} & [\hat{\theta}_{\alpha/2}^*, \hat{\theta}_{1-\alpha/2}^*], \quad \hat{\theta}_{\alpha/2}^* \text{ is the } \alpha/2 \text{ quantile of } \hat{\theta}^*, \\ & \text{with } \hat{\theta}_{1-\alpha/2}^*, \hat{\theta}_{1-\alpha/2}^* \text{ being the quantiles of the empirical distribution } \left\{ \hat{\theta}^{(b)} \right\}_{b=1,\dots,B} \end{aligned} \quad (944)$$

17.5.4 Bootstrap Hypothesis Testing

Consider we want to test a null hypothesis H_0 and we have a test statistic T where observed values of T under H_0 can be calculated for a sample as $t_{\text{obs}}(X_{\text{sample}})$. In a setting where we make assumptions on the distributions of the test statistic we would test based on where our observed value falls in the distribution of the test statistic under H_0 .

Problem: What if the sampling distribution of the test statistic is not known? What if assumptions of asymptotic tests are not met?

Intermezzo - general empirical distribution function: Consider we have observed data $X = (x_1, \dots, x_n)$ from some underlying population distribution F . The distribution information of F lies in how often similar values of x_i occur in the sample X . We therefore estimate the cumulative distribution function F by the empirical distribution function \hat{F}_n as

$$\hat{F}_n(x) = EDX(x) = \frac{\#\{x_i \leq x\}}{N} \quad (945)$$

which resembles the true distribution in the large N limit.

Idea: While we only have one sample at hand, by resampling we can generate multiple *samples* and based on those samples **conditioned on** H_0 find an empirical distribution of the test statistic under H_0 . See figure 156a.

Note: The re-samples / empirical distribution must be constructed / adapted to follow H_0 . So we want to use the distributional information of the sample to generate new samples that are consistent with H_0 . If we would just directly calculate our test statistic on resampled samples and then our observed test statistic by the same formula - why should we expect to ever reject the null hypothesis?

In table 34, the different kinds of tests with respect how the test-statistic distribution is constructed are compared.

	Exact test	Parametric test	Nonparametric bootstrap test
Knowledge on the distribution function F of the test statistic	$F_\alpha(T)$ (/ the parameters α) is known exactly based on H_0	In $F_{\hat{\alpha}}(T)$ some parameters have to be estimated, e.g. the STD $\hat{\sigma}$	The distribution \hat{F} itself is estimated using the data constrained to H_0

Table 34: Comparison of different kinds of tests.

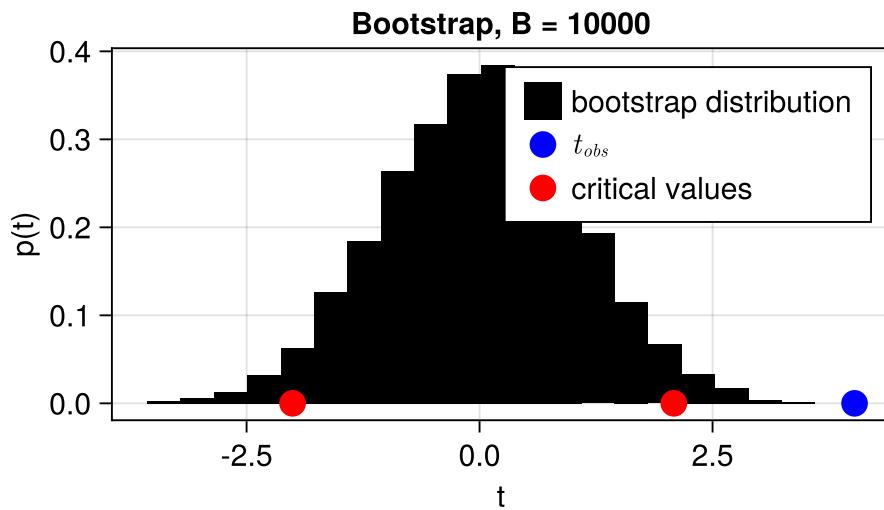
Note that there is also

- **Parametric bootstrap test:** The sample is assumed to come from a known distribution with unknown parameters. These parameters are estimated from the sample and then the distribution is used to generate bootstrap samples and to construct the *empirical distribution of the test statistic under H_0* .
- **Semiparametric bootstrap test:** E.g. do the usual bootstrap resampling but add noise (e.g. $\sim \mathcal{N}(\sigma^2)$)

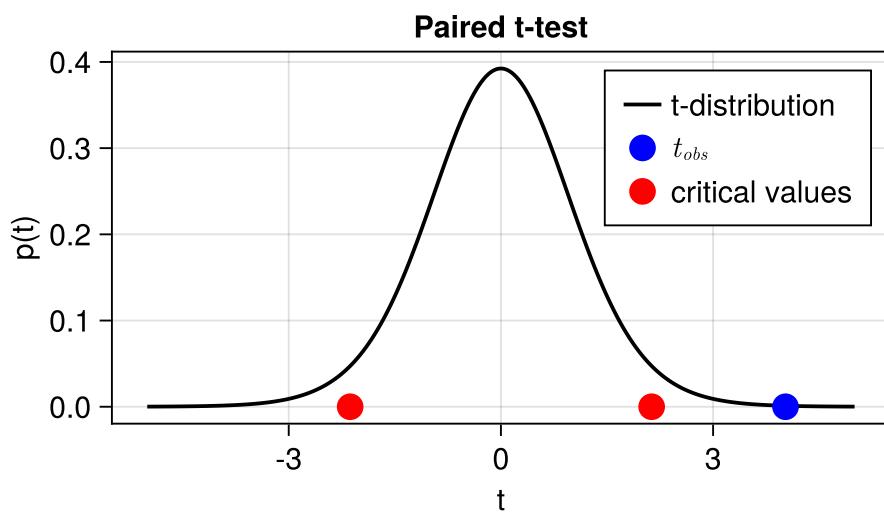
17.5.4.1 1-sample test as an introductory example

Consider for instance for a sample $X_{\text{sample}} = (x_1, \dots, x_n)$ we want to test the null hypothesis $H_0 : \mu = \mu_0$ vs $H_1 : \mu \neq \mu_0$ (one-sample test). We then base our bootstrap samples on $z_i = x_i - \bar{X}_{\text{sample}} + \mu_0$ and use the test statistic $t^{(b)} = t^*(Z^{(b)}) = \frac{\bar{Z}^{(b)} - \mu_0}{s_Z^{(b)}/\sqrt{n}}$ where $Z^{(b)} = (z_1^{(b)}, \dots, z_n^{(b)})$ and $s_Z^{(b)} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (z_i^{(b)} - \bar{Z}^{(b)})^2}$ and $t_{\text{obs}} = t^*(X_{\text{sample}})$.

Here we assume that as the mean varies, the distributions are just translated versions of each other (translation family) where if our x_i are lifetimes it might make sense to use logged lifetimes as they are more likely to satisfy a translation or normal family assumption (Efron and Tibshirani, 1994, chapter 16.4).



(a) (Two-sided) Bootstrap Hypothesis Testing based on empirical distribution from bootstrap samples.



(b) (Two-sided) Standard Hypothesis test with distributional assumption.

Figure 156: Bootstrap vs Distributional Assumption Hypothesis Testing.

17.5.4.2 Calculation of p values in Bootstrap Hypothesis Testing

Practically, we calculate estimates of the p values (where for $p < \alpha$ we reject H_0) as

$$\begin{aligned}\hat{p}_{\text{right}}^* &= \frac{1}{B} \sum_{b=1}^B \mathbb{1}_{[t^{(b)} \geq t_{\text{obs}}]} \\ \hat{p}_{\text{left}}^* &= \frac{1}{B} \sum_{b=1}^B \mathbb{1}_{[t^{(b)} \leq t_{\text{obs}}]} \\ \hat{p}_{\text{two-sided}}^* &= \frac{1}{B} \left[\min \left\{ \sum_{b=1}^B \mathbb{1}_{[t^{(b)} \geq t_{\text{obs}}]}, \sum_{b=1}^B \mathbb{1}_{[t^{(b)} \leq t_{\text{obs}}]} \right\} \right]\end{aligned}\quad (946)$$

17.5.4.3 Bootstrap Hypothesis Test on the Difference of Means

Consider we have two samples $X^{(1)} = (x_1^{(1)}, \dots, x_n^{(1)})$ and $X^{(2)} = (x_1^{(2)}, \dots, x_n^{(2)})$. And we want to test the null-hypothesis

$$H_0 : \mu^{(2)} - \mu^{(1)} \leq c \quad \text{vs} \quad H_1 : \mu^{(2)} - \mu^{(1)} > c \quad (947)$$

where $c \in \mathbb{R}$. Assume the true means are $\mu^{(1)}$ and $\mu^{(2)}$. We can generate a bootstrap distribution under H_0 by sampling from

$$\begin{aligned}z_i^{(1)} &= x_i^{(1)} - \bar{X}_{\text{sample}}^{(1)} - \frac{c}{2} \quad \text{and} \quad z_i^{(2)} = x_i^{(2)} - \bar{X}_{\text{sample}}^{(2)} + \frac{c}{2} \\ Z^{(1)} &= (z_1^{(1)}, \dots, z_n^{(1)}) \quad \text{and} \quad Z^{(2)} = (z_1^{(2)}, \dots, z_n^{(2)})\end{aligned}\quad (948)$$

and using the test statistic

$$\begin{aligned}t_{\text{obs}} &= \frac{\bar{X}_{\text{sample}}^{(2)} - \bar{X}_{\text{sample}}^{(1)} - c}{S_p \sqrt{\frac{1}{n^{(1)}} + \frac{1}{n^{(2)}}}} \\ S_p &= \sqrt{\frac{(n^{(1)} - 1)(S^{(1)})^2 + (n^{(2)} - 1)(S^{(2)})^2}{n^{(1)} + n^{(2)} - 2}}\end{aligned}\quad (949)$$

with analogous calculations of $t^{(b)}$ on the bootstrap samples (the basic reasoning behind the denominator is that the scale on which means can be compared is the standard deviation, without which a statement like the means are 1 apart is meaningless (very different statement for standard deviations of the order of 10 or 10^{-2})).

17.5.4.4 Bootstrap Hypothesis Test - Are two samples from different distributions?

Consider we have samples

$$\begin{aligned} X &= \{x_1, \dots, x_{N_X}\}, \quad X \sim F_X \\ Y &= \{y_1, \dots, y_{N_Y}\}, \quad Y \sim F_Y \end{aligned} \quad (950)$$

drawn from unknown distributions F_X, F_Y . Our aim is to test

$$H_0 : F_X = F_Y \quad \text{vs} \quad H_1 : F_X \neq F_Y \quad (951)$$

For testing if two samples possibly come from different distributions (e.g. test results of girls and boys), the testing procedure is

1. Formulate H_0 , so $H_0 : F_X = F_Y$.
2. Specify the α -level and test statistic T , which if $N_X = N_Y = N$ could be

$$T(X, Y) = \frac{\bar{\Delta}}{\sigma_\Delta / \sqrt{N}}, \quad \delta_i = x_i - y_i \quad (952)$$

3. **Bootstrapping:** From the union $Z = X \cup Y$ we make draw replacement N_b samples of sizes N_X and N_Y respectively with replacement. For $b = 1, \dots, N_b$
 - (a) Draw samples with repetition from Z

$$\begin{aligned} N_X \text{ draws with repetition from } Z \rightarrow \tilde{X}_b &= \{\tilde{x}_1, \dots, \tilde{x}_{N_X}\} \\ N_Y \text{ draws with repetition from } Z \rightarrow \tilde{Y}_b &= \{\tilde{y}_1, \dots, \tilde{y}_{N_Y}\} \\ \text{e.g. } \tilde{X}_b &= \{x_1, y_2, x_1, y_5, \dots\}, \tilde{Y}_b = \{y_1, y_4, x_3, y_8, \dots\} \end{aligned} \quad (953)$$

H_0 (*what if they were from the same distribution*) is built-in as we draw from the union of the samples.

- (b) Calculate the test statistic $t_b = T(\tilde{X}_b, \tilde{Y}_b)$.
4. Step 3 yields a set of test statistics t_1, \dots, t_{N_b} on which *histogram* check if $t_{\text{obs}} = T(X, Y)$ falls into the rejection region of H_0 . We reject H_0 if

$$\begin{aligned} p_{EDF}(t \geq t_{\text{observed}}) &\approx \frac{\# \text{ of bootstrap } t \geq t_{\text{observed}}}{\text{bootstrap drawings } N_b} < \frac{\alpha}{2} \\ \text{or } p_{EDF}(t \leq t_{\text{observed}}) &\approx \frac{\# \text{ of bootstrap } t \leq t_{\text{observed}}}{\text{bootstrap drawings } N_b} < \frac{\alpha}{2} \end{aligned} \quad (954)$$

See figure 156a for an illustration.

17.6 Multiple testing problem | significance by chance

Problem: Consider we are testing 100 independent null hypotheses (e.g. benefits of alcohol on 100 health indicators) at $\alpha = 0.05$. Then - only by chance - on average 5 of them will lead to the wrong rejection of H_0 (type I error) (so to wrong significant results, e.g. alcohol is good for red body cell production). So we are likely to run into a type I error.

Idea: We set a confidence level for the whole family of tests, and scale down the α -level for each test the more tests we do, so we scale down the confidence that a single test is significant (/ will generalize to independent data).

We want to control the **confidence in the family of tests which is quantified by the family-wise error rate (FWER)**. The FWER is the probability of obtaining at least one significant result just by chance (so when all null hypotheses are true).

$$\hat{\alpha} = p(\#\{\text{accept } H_1 \mid H_0 \text{ is true}\} \geq 1) = 1 - (1 - \alpha)^K \quad (955)$$

total number K of hypotheses tested with significance level α per test

FWER control: Decrease α with $1/K$ The so-called Bonferroni correction is

$$\alpha_{\text{per test}} = \frac{\alpha_{\text{family}}}{K}, \quad \text{e.g. } \alpha_{\text{family}} = 0.05 \quad (956)$$

Note: This correction comes at the cost of increasing the probability for false negatives (type II error) (so accepting the *no-difference* statement when we really have a significant result), so decreasing the statistical power of the test. Holm-Bonferroni is more powerful.

Alternatively one can also **control the false discovery rate** (the proportion of wrongly rejected null hypotheses (*false-discoveries*) to all rejected hypothesis (*discoveries*)). See e.g. Benjamin-Hochberg procedure.

18 Numerical Methods for Parameter Estimation

18.1 Overview on strategies for parameter estimation

Consider we have a model $F_{\underline{\theta}}$ with parameters $\underline{\theta}$ which we want to estimate based on

- a sample $X = \{\underline{x}_i\}_{i=1}^N$ in the setting where we model a distribution
- a sample $D = \{y_i, \underline{x}_i\}_{i=1}^N$ in the setting where we model a relation between independent (fixed) variables \underline{x}_i and dependent (random) variables y_i , so the probability model is $p(y_i|\underline{x}_i, \underline{\theta})$, and model $y_i = f_{\underline{\theta}}(\underline{x}_i) + \epsilon_i$, random part ϵ_i .

Maximum Likelihood Estimation (MLE) A common approach to estimate $\underline{\theta}$ is to maximize the likelihood function

$$\mathcal{L}_X(\underline{\theta}) = \prod_{i=1}^N p(\underline{x}_i|\underline{\theta}), \quad \mathcal{L}_D(\underline{\theta}) = \prod_{i=1}^N p(y_i|\underline{x}_i, \underline{\theta}) \quad (957)$$

which we in any case formulate as minimizing the loss L_{MLE}

$$\hat{\underline{\theta}} = \underset{\underline{\theta}}{\operatorname{argmin}} L_{\text{MLE}}(\underline{\theta}) = \underset{\underline{\theta}}{\operatorname{argmin}} -\mathcal{L}\mathcal{L}(\underline{\theta}) = \underset{\underline{\theta}}{\operatorname{argmin}} -\log \mathcal{L}(\underline{\theta}) \quad (958)$$

(equivalence to maximizing the likelihood follows as the logarithm is strictly monotonous, so it leaves order relations intact, and by the sign we go to minimization).

This is illustrated in figure 157.

Least Squares Estimation (LSE) In the case of a relation between independent and dependent variables, we can minimize

$$\hat{\underline{\theta}} = \underset{\underline{\theta}}{\operatorname{argmin}} L_{\text{SSQ}}(\underline{\theta}) = \underset{\underline{\theta}}{\operatorname{argmin}} \sum_{i=1}^N (y_i - f_{\underline{\theta}}(\underline{x}_i))^2 \quad (959)$$

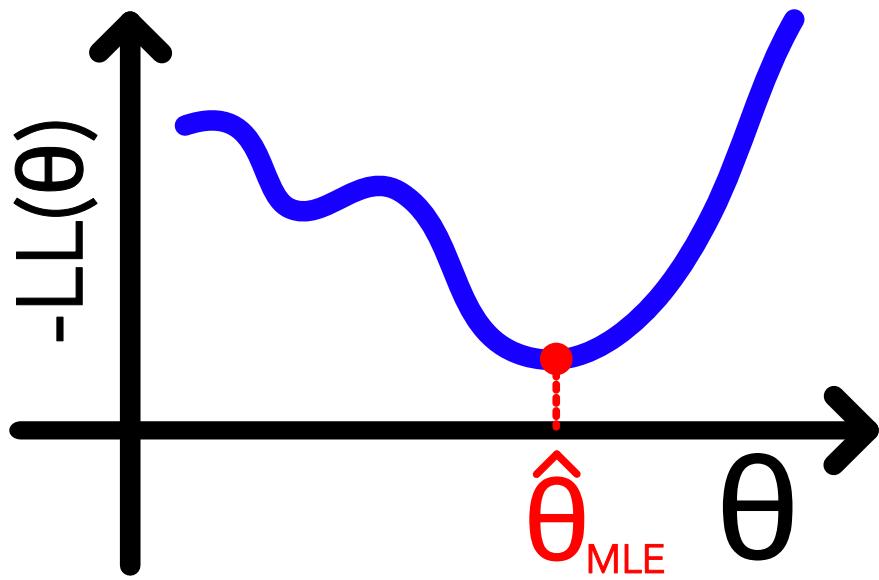


Figure 157: Illustration of the optimization problem for MLE.

Note: While in Bayesian statistics (where compared to the frequentist approach we use information from a prior^a) one could use the MAP, $\hat{\theta} = \operatorname{argmax}_{\theta} p(\theta|X)$, which we could also write as the minimization of a loss, this is not usual. One more commonly uses the expectation of the posterior distribution as an estimator if necessary (which has lower a-posteriori variance than the MAP). More later.

^aIf the prior assigns the same probability to all parameters, we are back at MLE.

18.2 Overview on numerical methods for minimizing the loss (in the MLE setting)

The basic methods are to either numerically minimize the loss function directly, e.g.

- gradient descent

or to apply root-finding methods to the gradient of the loss function, in the MLE setting the gradient of the log-likelihood, **the score**

$$\underline{s}(\underline{\theta}) = \nabla_{\underline{\theta}} \log \mathcal{L}(\underline{\theta}) \quad (960)$$

with methods like

- Newton-Raphson

- Fisher scoring⁴⁰

18.3 Gradient Descent

Here we just walk down the gradient of the loss function, starting at some initial guess $\underline{\theta}_0$, so

$$\begin{aligned}\hat{\underline{\theta}}^{(n+1)} &= \hat{\underline{\theta}}^{(n)} - \gamma \nabla_{\underline{\theta}} L \Big|_{\underline{\theta}=\hat{\underline{\theta}}^{(n)}}, \quad \text{learning rate } \gamma \\ \text{until } &\left| L\left(\hat{\underline{\theta}}^{(n+1)}\right) - L\left(\hat{\underline{\theta}}^{(n)}\right) \right| < \epsilon \text{ or } \# \text{ steps} > N_{\max}\end{aligned}\tag{962}$$

where ϵ is a small number and N_{\max} is the maximum number of steps, set by the user. Alternatively, one can stop when $\|\underline{\theta}^{(n+1)} - \underline{\theta}^{(n)}\| < \epsilon$.

Problems of standard gradient descent:

- finds local minima
- how to choose the learning rate
- gets stuck on plateaus

18.3.1 Improvements to standard gradient descent

- to combat local minima, we can start from multiple initial conditions and take the best final result
- we prefer an approximate global optimum over an exact local one → use stochasticity, (e.g. injected noise → jump out of local but not global minima); simulated annealing (probabilistically move to some state or stay in the current one, even locally suboptimal steps are allowed)
- **stochastic gradient descent:** consider we want to minimize an additive function

$$Q(\underline{\theta}) = \frac{1}{N} \sum_{i=1}^N Q_i(\underline{\theta}), \quad \text{e.g. } L(\underline{\theta}) = -\mathcal{LL}(\underline{\theta}) = \sum_{i=1}^N -\log p(x_i | \underline{\theta})\tag{963}$$

⁴⁰Like Newton-Raphson but with the Fisher information matrix in place of the Hessian.

$$\underline{\underline{I}}(\underline{\theta}) = E(\underline{s}\underline{s}^T) = -E\left(\frac{\partial^2 \log \mathcal{L}}{\partial \underline{\theta} \partial \underline{\theta}^T}\right)\tag{961}$$

(the negative expectation of the Hessian of the log-likelihood) (which is not a function of any particular observation as the random variable X was averaged out).

We do the gradient descent steps by

$$\hat{\underline{\theta}}^{(n+1)} = \hat{\underline{\theta}}^{(n)} - \frac{\gamma}{k} \nabla_{\underline{\theta}} \sum_{\substack{\text{random subset} \\ \text{of size } k}} Q_k(\underline{\theta}) \quad (964)$$

so we only consider the loss following from a subset of data points - which can avoid local minima (noise in the data itself is used)

- adaptive learning rate γ (reduce for steep gradients) (e.g. AdaGrad, RMSProp, Adam)

18.4 Newton-Raphson

18.4.1 Standard Newton iteration for root finding

Consider we want to find the root of a function $g(\xi)$. We linearly approximate g around the current guess ξ_k and solve for the root.

$$\begin{aligned} g(\xi_{k+1}) &\approx g(\xi_k) + (\xi_{k+1} - \xi_k) \partial_\xi g|_{\xi=\xi_k} = 0 \\ \rightarrow \xi_{k+1} &= \xi_k - \frac{g(\xi_k)}{\partial_\xi g|_{\xi=\xi_k}} \end{aligned} \quad (965)$$

so in the multivariate case, $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$, we have

$$\xi_{k+1} = \xi_k - \underline{J}_g^{-1} \left(\xi_k \right) \underline{g} \left(\xi_k \right), \quad \underline{J}\underline{g} = \begin{pmatrix} - & \nabla_{\underline{\xi}} g_1 & - \\ \vdots & \vdots & \vdots \\ - & \nabla_{\underline{\xi}} g_n & - \end{pmatrix} \quad (966)$$

Which is illustrated in figure 158.

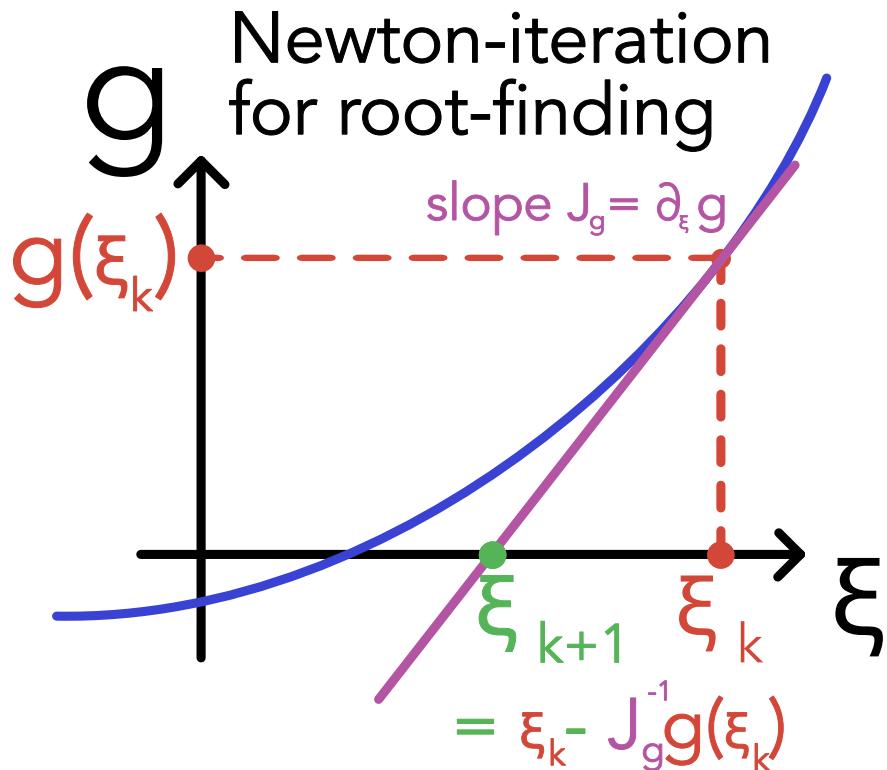


Figure 158: Illustration of the Newton-Raphson method.

18.4.2 Newton's method in optimization

We now want to find critical values ξ_{crit} of a function g , where

$$\partial_{\xi} g|_{\xi_{\text{crit}}} = 0 \quad (967)$$

These can be minima, maxima or saddle points.

Using Newton iteration, we get

$$\xi_{k+1} = \xi_k - \frac{\partial_{\xi} g|_{\xi=\xi_k}}{\partial_{\xi}^2 g|_{\xi=\xi_k}} \quad (968)$$

Intuition of Newton-optimization: At ξ_k , a parabola is fitted to the function g and we move the the extremum (in higher dimension this can be a saddle point) of it.

In higher dimensions $g : \mathbb{R}^n \rightarrow \mathbb{R}$, we get

$$\underline{\xi}_{k+1} = \underline{\xi}_k - \underline{\underline{H}}_g^{-1}(\underline{\xi}_k) \underline{\nabla}_{\underline{\xi}} g(\underline{\xi}_k) \quad (969)$$

with the numerically costly Hessian

$$\underline{\underline{H}}_g = \underline{\underline{\nabla}}_{\xi} g \quad (970)$$

best analytically found or calculated based on Automatic Differentiation (AD).

In our loss-optimization we simply have $g(\xi) = L(\theta)$.

18.4.2.1 Advantages of Newton optimization

- In the convex / concave setting the usage of curvature information by Newton iteration leads to faster convergence than gradient descent, as illustrated in figure 159.
- step-size is automatically adaptive
 - *small Hessian* → automatically larger steps
 - *large Hessian* → automatically smaller steps

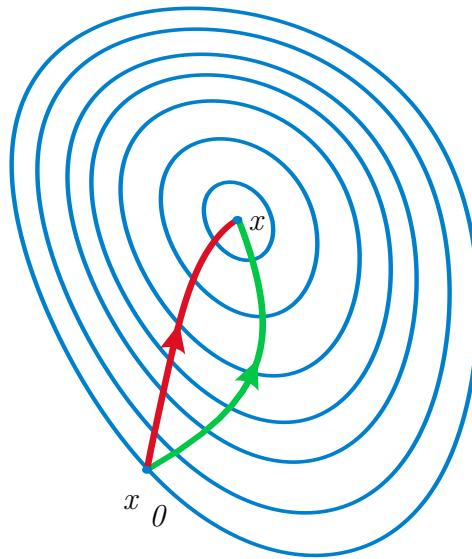


Figure 159: Newton optimization (green), gradient descent (red).

18.4.2.2 Disadvantages of Newton optimization

- strictly only applies to convex / concave functions g
- at the inflection point of a non-convex function, the Hessian is indefinite so in the 1D case $\partial_{\xi}^2 g = 0$ and Newton optimization breaks down
- it searches critical points, not minima, if $\partial_{\xi}^2 g < 0$ and $\partial_{\xi} g > 0$, we go uphill, see figure 160.

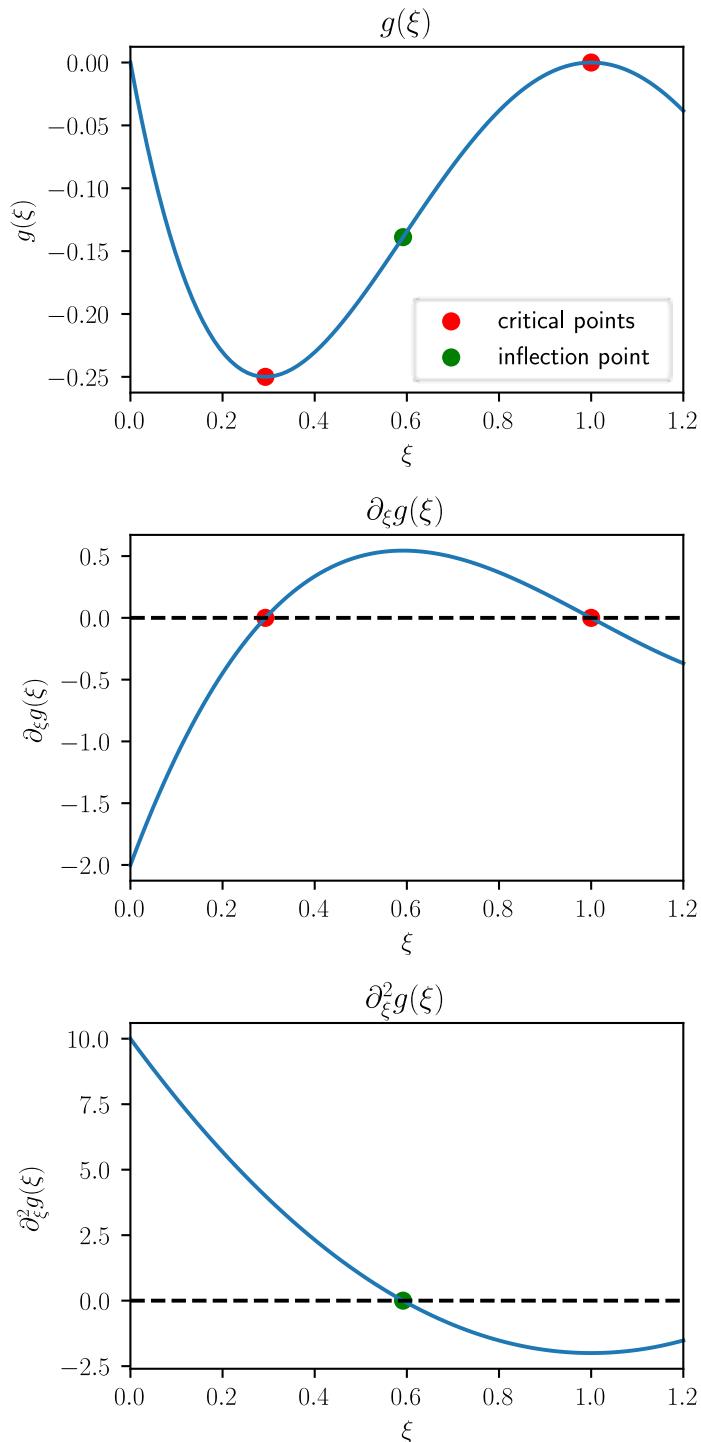


Figure 160: Newton optimization breaks down at the inflection point and goes uphill right of it.

18.5 Numerical Parameter Estimates in a Bayesian Setting

Consider we have collected data $\{x_i\}_{i=1}^N$. We want to model this data using $f(\underline{x}|\underline{\mu})$ - a parametric model with parameters $\underline{\mu}$. The model could for instance be a Gaussian with the parameters being the mean and standard deviation.

In Bayesian statistic we assume we somehow have prior knowledge in form of a distribution $g(\underline{\mu})$ over the parameters.

$$\text{posterior} = \Pi(\underline{\mu}|\underline{x}) = \frac{\mathcal{L}(\underline{x}|\underline{\mu})g(\underline{\mu})}{h(\underline{x})} = \frac{\mathcal{L}(\underline{x}|\underline{\mu})g(\underline{\mu})}{\int \mathcal{L}(\underline{x}|\underline{\mu})g(\underline{\mu}) d\underline{\mu}} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}} \quad (971)$$

posterior \propto likelihood · prior

18.5.1 Maximum A Posteriori (MAP) Estimation

The MAP estimator is the mode of the posterior distribution

$$\hat{\underline{\mu}}_{MAP} = \underset{\underline{\mu}}{\operatorname{argmax}} \Pi(\underline{\mu}|\underline{x}) = \underset{\underline{\mu}}{\operatorname{argmax}} \mathcal{L}(\underline{x}|\underline{\mu})g(\underline{\theta}) \quad (972)$$

which for $g(\underline{\mu}) \sim \text{Beta}(1, 1)$ (uniform, no prior knowledge) is equivalent to MLE.

Problem: In e.g. a bimodal distribution, the mode is uncharacteristic of the majority of the distribution (imagine a *Gaussian with a random peak at one of the tails*). From this it makes sense to expect this estimator to have a relatively high variance (low precision).

18.5.2 A-posteriori Expectation

Bayes estimator: In general, a bayes estimator is given by

$$\hat{\underline{\mu}}_{\text{Bayes}} = \underset{\tilde{\underline{\mu}}}{\operatorname{argmin}} E(\ell(\tilde{\underline{\mu}}, \underline{\mu})|\underline{x}) = \int \ell(\tilde{\underline{\mu}}, \underline{\mu}) \Pi(\underline{\mu}|\underline{x}) d\underline{\mu} \quad (973)$$

For $\ell(\tilde{\underline{\mu}}, \underline{\mu}) = (\tilde{\underline{\mu}} - \underline{\mu})^2$, the Bayes estimator is the expectation of the posterior distribution

$$\hat{\underline{\mu}}_{\text{PLSQ}} = \int \underline{\mu} \Pi(\underline{\mu}|\underline{x}) d\underline{\mu} \quad (974)$$

which can be derived from

$$\partial_{\tilde{\underline{\mu}}} E(\ell(\tilde{\underline{\mu}}, \underline{\mu})|\underline{x}) \Big|_{\tilde{\underline{\mu}}=\hat{\underline{\mu}}_{\text{PLSQ}}} = 0 \quad (975)$$

$\hat{\underline{\mu}}_{\text{PLSQ}}$ is the point around which the posterior distribution has the lowest expected squared deviation (a kind of inertia).

Problem: But how to calculate this expected value? The integrals, that for the expectation and that for the evidence are often intractable (very high-dimensional).

Idea: By Markov-Chain Monte Carlo, we can sample from the posterior and calculate

$$\hat{\underline{\mu}}_{\text{PLSQ}} \approx \frac{1}{N} \sum_{i=1}^N \underline{\mu}_i \quad (976)$$

samples from the posterior $\underline{\mu}_i \sim \Pi(\underline{\mu} | \underline{x})$

where for the MCMC sampling the evidence drops out. See section 14.6.

19 Regression

Aim of regression: In Regression analysis, based on a sample, we are trying to find the relationship between a dependent (continuous) variable y (*outcome, response*) and one or more independent variables collected in the vector \underline{x} (*features, predictors, regressors, explanatory variables*).

$$D = \{y_i, \underline{x}_i\}_{i=1}^n, \quad \text{model } E[y_i | \underline{x}_i] = f_{\underline{\theta}}(\underline{x}_i) \quad (977)$$

parameters $\underline{\theta}$, continuous variable y

Possible aims in regression analysis are

- based on the estimated model parameters from a sample, predict the outcome for new observations
- understand the relationship between the outcome and the predictors

Primer on machine learning: In Machine Learning (ML) we want to find meaningful patterns in data. We can roughly divide ML into

- **Supervised learning:** We have input-output pairs (aka labeled data, with the outcome variable being the labeling), with the basic aim being labeling (aka making a prediction of the output for) new (unseen) data
 - **Regression:** here the continuous y_i is the outcome variable, our labeling (e.g. given environmental factors \underline{x} we want to predict the temperature)
 - **Classification:** here the y_i is a categorical (nominal) variable, and we model category probabilities, e.g. for \underline{x} being some representation of an image we want to assign it to one of some categories (e.g. elephant, cow, ...)
- **Unsupervised learning:** We have only input data, and we want to find patterns in it
 - **Clustering:** We want to find groups of similar observations (e.g. K-means, agglomerative clustering, ...)
 - **Dimensionality reduction:** We want to find a lower-dimensional representation of the data
- **Reinforcement learning:** We want to find a policy for an agent to maximize some reward

Note: While regression and classification differ in that the outcome variable is continuous in the former and categorical in the latter, they are also connected. For instance in the Generalized Linear Model (GLM) framework, we can also model class probabilities, e.g. for logistic regression, which really is a classification model.

19.1 Multiple Linear Regression | the standard

19.1.1 Model setup | what we model how under which assumptions

Goals: Consider we have measured

$$\text{features } \underline{x}_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix}, \text{e.g. trace gas concentrations} \quad (978)$$

outcome y_i , e.g. temperature

and we want to

- predict the temperature at given trace gas concentrations
- find which trace gas concentrations are important for the temperature (important explanatory variables)

Assumed underlying model: We assume that the data $D = \{y_i, \underline{x}_i\}_{i=1}^n$ is generated by the model (data modelling culture)

$$y(\underline{x}) = \underline{\beta}^* \cdot \underline{x} + \epsilon(x), \quad \text{true parameter } \underline{\beta}^* = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \quad (979)$$

augmented independent variables $\begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_p \end{pmatrix}$

The random variables are y and ϵ . The β_i are non-random but unobservable.

Note: The model is linear in the parameters but not necessarily in the original features - we can e.g. get to polynomial regression, by expanding the feature-vector by powers of the original features (*feature engineering*).

19.1.1.1 Assumptions

We make the general assumption that

1. \underline{x} is deterministic and exact, there is no error in the explanatory variables (which is not true for experimental data)⁴¹

When we add the **Gauss-Markov assumptions**

2. $E[\epsilon(\underline{x})] = 0$, so a linear model is applicable (the error is not systematically biased)
3. $Var[\epsilon(\underline{x})] = \sigma^2 = \text{const.} < \infty$ independent of \underline{x} (homoscedasticity), otherwise high variance regions would induce instability (see figure 161)
4. $Cov[\epsilon(\underline{x}_i), \epsilon(\underline{x}_j)] = 0$ for $i \neq j$, so the errors are uncorrelated

then the best **linear unbiased estimator** (BLUE) of β^* is the **Ordinary Least Squares** (OLS) estimator $\hat{\beta}$ (by the **Gauss-Markov theorem**). Best in the sense that it has the **lowest sampling variance** among all linear unbiased estimators.

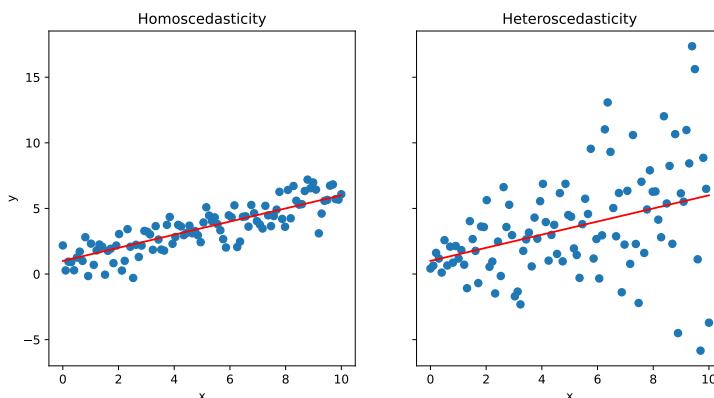


Figure 161: homoscedasticity vs heteroscedasticity

Finally, if we additionally assume

6. $\epsilon(\underline{x})$ to be independent and identically (i.i.d.) normally distributed

then the sampling distribution of the OLS (ordinary least squares) estimate $\hat{\beta}$ will also be normal (see also later).

⁴¹If the experimental errors in \underline{x} are to be considered, resort to orthogonal distance regression.

Generally OLS linear regression is an easy, robust benchmark for more complex models. In situations with little data a bias in form of a regularization (we can also formulate as a prior) can help.

19.1.1.2 Matrix notation

For the whole sample $\{\underline{x}_i, y_i\}_{i=1}^N$ we can write

$$\underline{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_N \end{pmatrix}, \quad \underline{\underline{X}} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \cdots & x_{Np} \end{bmatrix} \in \mathbb{R}^{N \times p} \quad (980)$$

N observations, explanatory variable with p features

$\underline{\underline{X}}$ with 1 column to allow for constant offset β_0

so we can write the model as

$$\hat{y} = \underline{\underline{X}} \hat{\beta} \quad (981)$$

19.1.2 Two views on finding an estimator for $\underline{\beta}^*$

Consider the true model to be

$$\underline{y} = \underline{\underline{X}} \underline{\beta}^* + \underline{\epsilon}, \quad \underline{\epsilon} \sim \mathcal{N}\left(0, \sigma^2 \underline{\underline{I}}_{N \times N}\right), \quad \text{some true parameters } \underline{\beta}^*, \quad \epsilon_i \text{ i.i.d.} \quad (982)$$

$\underline{\underline{X}}$ is deterministic and y_i are distributed normally around a modeled mean. The basic intuitions for finding an estimator $\hat{\beta}$ for $\underline{\beta}^*$ are

- **Least squares:** Imagine springs between data points and model (along y), minimizing the overall energy
- **Maximum likelihood:** Imagine the data points as the result of a stochastic process, we want to find the model parameters that make the data most likely

both lead to the same estimator $\hat{\beta}$. The two intuitions are illustrated in figure 162.

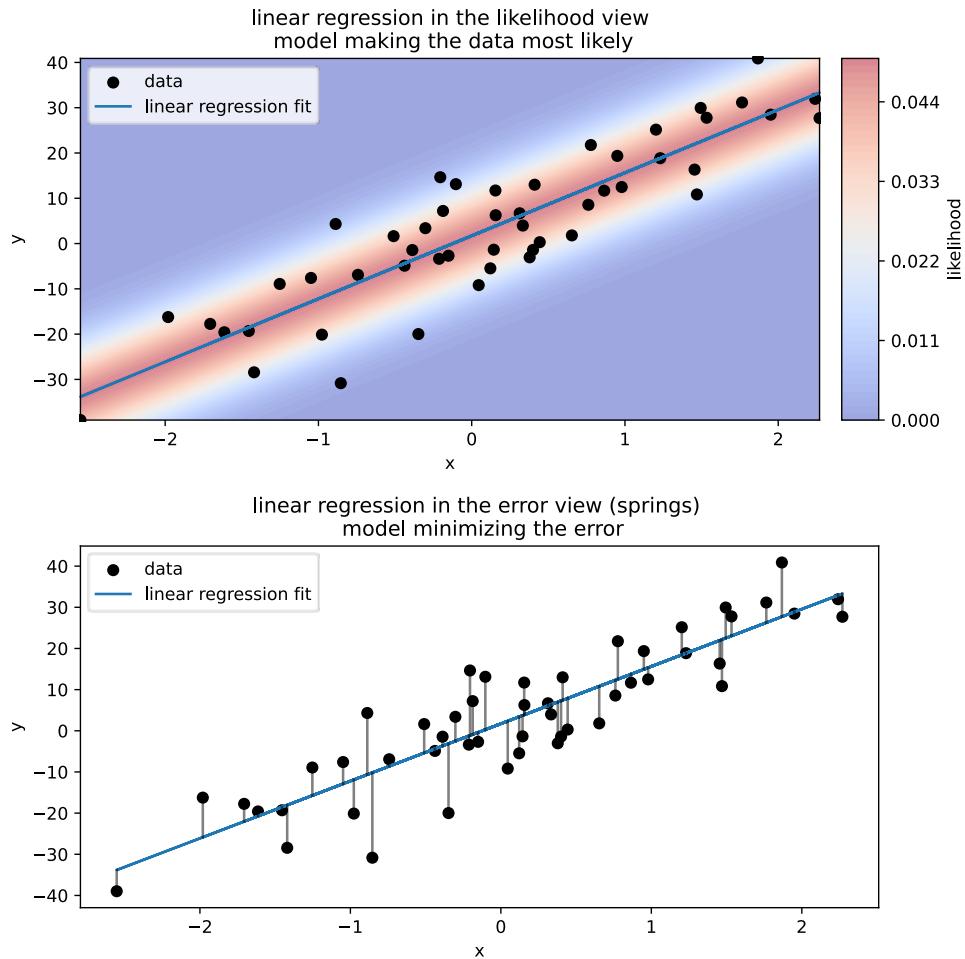


Figure 162: Two views on finding an estimator for $\underline{\beta}^*$

19.1.3 Finding the model parameters minimizing the sum of squared errors

We can calculate the sum of squares residual (a scalar) as

$$\text{SSQ}(\underline{\beta}) = \|\underline{Y} - \underline{\underline{X}}\underline{\beta}\|_2^2 = (\underline{Y} - \underline{\underline{X}}\underline{\beta})(\underline{Y} - \underline{\underline{X}}\underline{\beta})^T = \|\underline{\epsilon}\|_2^2 \quad (983)$$

which is nicely convex, so we can find a unique minimum by

$$\begin{aligned} \frac{\partial \text{SSQ}}{\partial \underline{\beta}} &= -2\underline{\underline{X}}^T(\underline{Y} - \underline{\underline{X}}\underline{\beta}) = 0 \\ \rightarrow \underline{\underline{X}}^T \underline{Y} &= \underline{\underline{X}}^T \underline{\underline{X}}\underline{\beta} \rightarrow \hat{\underline{\beta}} = (\underline{\underline{X}}^T \underline{\underline{X}})^{-1} \underline{\underline{X}}^T \underline{Y} \quad (\text{normal equation}) \end{aligned} \quad (984)$$

On the side, note

- the expression for $\hat{\underline{\beta}}$ can be read as the cross-covariance of \underline{Y} and $\underline{\underline{X}}$, normalized by

the auto-covariance of $\underline{\underline{X}}$

- $(\underline{\underline{X}}^T \underline{\underline{X}})^{-1} \underline{\underline{X}}^T$ is the **Moore-Penrose pseudoinverse** of $\underline{\underline{X}}$

Note: This only holds for $\underline{\underline{X}}^T \underline{\underline{X}}$ invertible, so not if features are linearly dependent or if $N < p$.

Estimator for the overparametrized regime*: If $N < p$ (more features than data points in the sample), we can exactly fit our data at hand, so it makes sense to use the optimization with Lagrange multipliers

$$\hat{\underline{\beta}}_{\text{over}} = \underset{\underline{\beta}, \lambda}{\operatorname{argmin}} L(\underline{\beta}, \lambda) = \underset{\underline{\beta}, \lambda}{\operatorname{argmin}} \|\underline{\beta}\|_2^2 + \lambda^T (\underline{Y} - \underline{\underline{X}} \underline{\beta}) \quad (985)$$

leading to the estimator

$$\hat{\underline{\beta}}_{\text{over}} = \underline{\underline{X}}^T (\underline{\underline{X}} \underline{\underline{X}}^T)^{-1} \underline{Y}, \quad \text{Gram-Matrix } \underline{\underline{X}} \underline{\underline{X}}^T \in \mathbb{R}^{N \times N} \quad (986)$$

as of

$$\begin{aligned} \nabla_{\underline{\beta}} \mathcal{L}(\underline{\beta}, \lambda) &= \vec{0} = 2\hat{\underline{\beta}} - \underline{\underline{X}}^T \underline{\lambda} \Rightarrow \hat{\underline{\beta}}_{\text{over}} = \frac{1}{2} \underline{\underline{X}}^T \underline{\lambda} \\ \nabla_{\underline{\lambda}} \mathcal{L}(\underline{\beta}, \lambda) &= \vec{0} = \underline{Y} - \underline{\underline{X}} \hat{\underline{\beta}}_{\text{over}} \Rightarrow \underline{Y} = \frac{1}{2} \underline{\underline{X}} \underline{\underline{X}}^T \underline{\lambda} \\ &\Rightarrow \underline{\lambda} = 2 \left(\underline{\underline{X}} \underline{\underline{X}}^T \right)^{-1} \underline{Y} \\ &\Rightarrow \hat{\underline{\beta}}_{\text{over}} = \underline{\underline{X}}^T \left(\underline{\underline{X}} \underline{\underline{X}}^T \right)^{-1} \underline{Y} \end{aligned} \quad (987)$$

While in the underparametrized regime, the moment (aka scattering) matrix $\underline{\underline{X}}^T \underline{\underline{X}}$ is invertible (if no collinearities), in the overparametrized regime, the Gram matrix $\underline{\underline{X}} \underline{\underline{X}}^T$ is invertible.

19.1.4 How is $\hat{\underline{\beta}}$ distributed? - normally

We consider the underlying situation to be Consider the true model to be

$$\underline{y} = \underline{\underline{X}} \underline{\beta}^* + \underline{\epsilon}, \quad \underline{\epsilon} \sim \mathcal{N} \left(0, \sigma^2 \mathbf{I}_{N \times N} \right), \quad \text{some true parameters } \underline{\beta}^*, \quad \epsilon_i \text{ i.i.d.} \quad (988)$$

and our estimate is

$$\hat{\underline{\beta}} = \left(\underline{\underline{X}}^T \underline{\underline{X}} \right)^{-1} \underline{\underline{X}}^T \underline{Y} \quad (989)$$

$\underline{\underline{X}}$ is deterministic (fixed). Consider for the same $\underline{\underline{X}}$ we would have different realizations (samples) of \underline{Y} , brought forth from the true model as of different random parts $\underline{\epsilon}$.

Naturally, we expect to get different estimates $\hat{\beta}$ for such different samples.

Question: For given fixed \underline{X} , how is the distribution of $\hat{\beta}$ based on different realizations of \underline{Y} ?

Form of the distribution - normal: Assuming $\underline{\epsilon} \sim \mathcal{N}\left(0, \sigma^2 \mathbf{I}_{N \times N}\right)$, $\underline{y} \sim \mathcal{N}\left(\underline{X}\underline{\beta}^*, \sigma^2 \mathbf{I}_{N \times N}\right)$, so as $\hat{\beta}$ is a linear transformation of \underline{Y} (remember \underline{X} deterministic), it is also normally distributed.

19.1.4.1 Intuition for the dependence of the distribution of $\hat{\beta}$ on the Distribution of the explanatory variables

How much normally wiggling at \underline{Y} translates into wiggling at $\hat{\beta}$ (so a spread in the distribution of $\hat{\beta}$) depends in the distribution of the explanatory variables, as illustrated in figure 163.

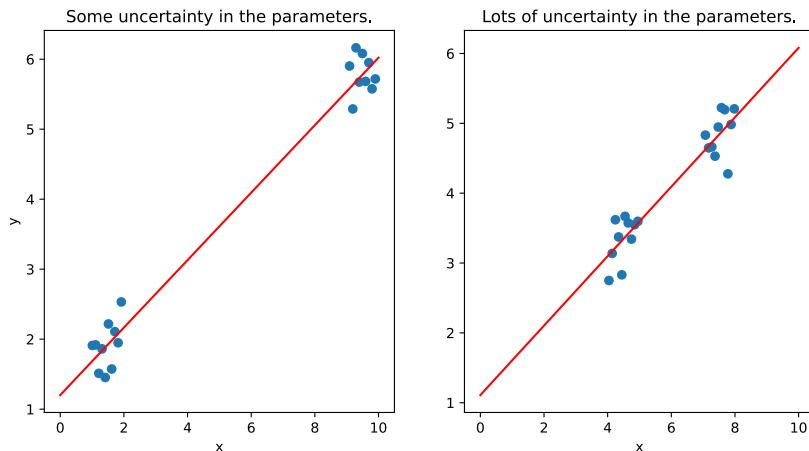


Figure 163: In the 1D setting, the less spread the explanatory variable, the more their wiggling translates into wiggling at $\hat{\beta}$.

An example in 2D (only the independent variables plotted) is given in figure 164. The more the data is spread along a direction the less is $\hat{\beta}$ sensitive to wiggling in this direction, the narrower the distribution of $\hat{\beta}$ in this direction.

19.1.4.2 Exact form of the normal distribution of $\hat{\beta}$

Expectation value of $\hat{\beta}$ | it is an unbiased estimator We can see that $\hat{\beta}$ is an unbiased estimator from

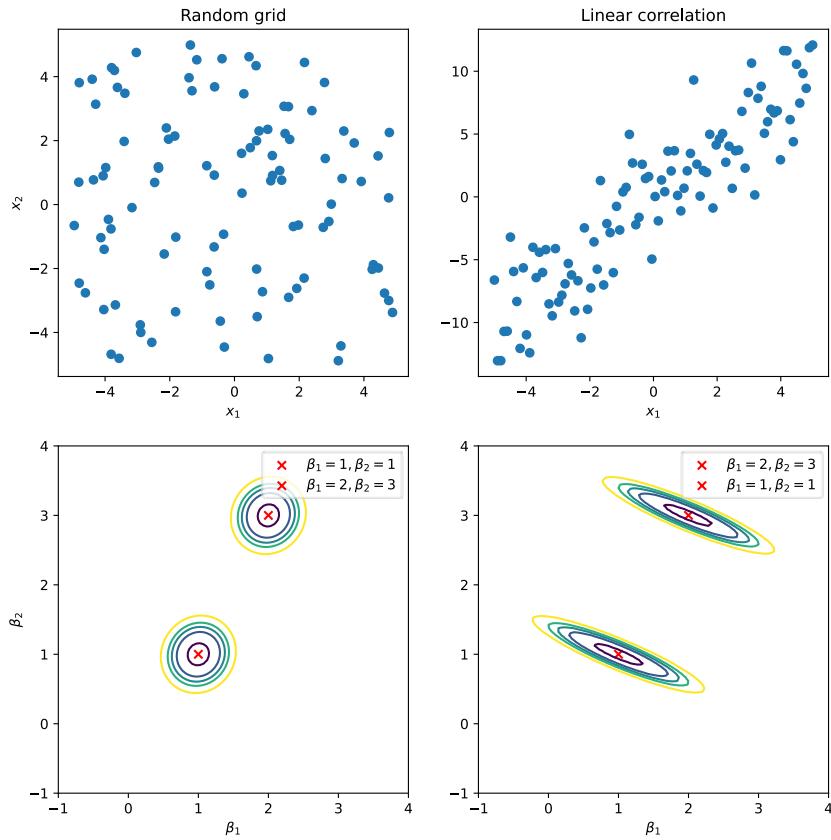


Figure 164: Depending on the distribution of $\underline{\underline{X}}$, here $\in \mathbb{R}^{N \times 2}$, the spread of $\hat{\underline{\beta}}$ differs, illustrated are the isocontours of the sum of squares expression, $\hat{\underline{\beta}}$ on a given isocontour are equally likely.

$$\begin{aligned}
E[\hat{\underline{\beta}}] &= E \left[\left(\underline{\underline{X}}^T \underline{\underline{X}} \right)^{-1} \underline{\underline{X}}^T \underline{\textcolor{teal}{Y}} \right] \\
&= E \left[\left(\underline{\underline{X}}^T \underline{\underline{X}} \right)^{-1} \underline{\underline{X}}^T \left(\underline{\underline{X}} \underline{\beta^*} + \underline{\epsilon} \right) \right] \\
&= E \left[\left(\underline{\underline{X}}^T \underline{\underline{X}} \right)^{-1} \left(\underline{\underline{X}}^T \underline{\underline{X}} \right) \underline{\beta^*} \right] + \textcolor{red}{E} \left[\left(\underline{\underline{X}}^T \underline{\underline{X}} \right)^{-1} \underline{\underline{X}}^T \underline{\epsilon} \right] \\
&= \underset{\substack{\underline{\underline{X}} \text{ exact} \\ \underline{\epsilon} \sim \mathcal{N}(0, \sigma^2 \underline{\underline{I}}_{N \times N})}}{E[\underline{\beta^*}]} \\
&\quad \underline{\beta^*} \underset{\text{exact}}{=} \underline{\beta^*}
\end{aligned} \tag{990}$$

where $\underline{\beta^*}$ is the true parameter vector.

Calculation of the covariance of $\hat{\underline{\beta}}$ We calculate

$$\begin{aligned}
 \text{Cov}_\epsilon(\hat{\underline{\beta}}) &= E \left(\left(\hat{\underline{\beta}} - E(\hat{\underline{\beta}}) \right) \left(\hat{\underline{\beta}} - E(\hat{\underline{\beta}}) \right)^T \right) \in \mathbb{R}^{p \times p} \\
 &= \text{Cov}_\epsilon \left(\left(\underline{\underline{X}}^T \underline{\underline{X}} \right)^{-1} \underline{\underline{X}}^T \underline{\underline{Y}} \right) \\
 &= \text{Cov}_\epsilon \left(\left(\underline{\underline{X}}^T \underline{\underline{X}} \right)^{-1} \underline{\underline{X}}^T \left(\underline{\underline{X}} \underline{\underline{\beta}}^* + \underline{\underline{\epsilon}} \right) \right) \\
 &\stackrel{\underline{\beta}^* \text{ deterministic} \rightarrow \text{no cov.}}{=} \text{Cov}_\epsilon \left(\underbrace{\left(\underline{\underline{X}}^T \underline{\underline{X}} \right)^{-1}}_{:= A} \underline{\underline{X}}^T \underline{\underline{\epsilon}} \right)
 \end{aligned} \tag{991}$$

As $\underline{\underline{A}}$ is just a constant matrix

$$\text{Cov}_\epsilon \left(\underline{\underline{A}} \underline{\underline{\epsilon}} \right) = \underline{\underline{A}} \text{Cov}_\epsilon(\underline{\underline{\epsilon}}) \underline{\underline{A}}^T \tag{992}$$

and as $\underline{\underline{\epsilon}} \sim \mathcal{N} \left(0, \sigma^2 \mathbf{1}_{N \times N} \right)$ i.i.d., we have

$$\text{Cov}_\epsilon(\underline{\underline{\epsilon}}) = \sigma^2 \mathbf{1}_{N \times N} \tag{993}$$

so

$$\boxed{\text{Cov}_\epsilon(\hat{\underline{\beta}}) = \sigma^2 \left(\underline{\underline{X}}^T \underline{\underline{X}} \right)^{-1}} \tag{994}$$

so wiggle \times inverse scattering. This is illustrated in 165.

Note: The expression for the covariance is true for any distribution of $\underline{\underline{\epsilon}}$ with $E[\underline{\underline{\epsilon}}] = 0$ and $\text{Cov}[\underline{\underline{\epsilon}}] = \sigma^2 \mathbf{1}_{N \times N}$.

19.1.4.3 Distribution of $\hat{\underline{\beta}}$

So we have

$$\hat{\underline{\beta}} \sim \mathcal{N} \left(\underline{\underline{\beta}}^*, \sigma^2 \left(\underline{\underline{X}}^T \underline{\underline{X}} \right)^{-1} \right) \tag{995}$$

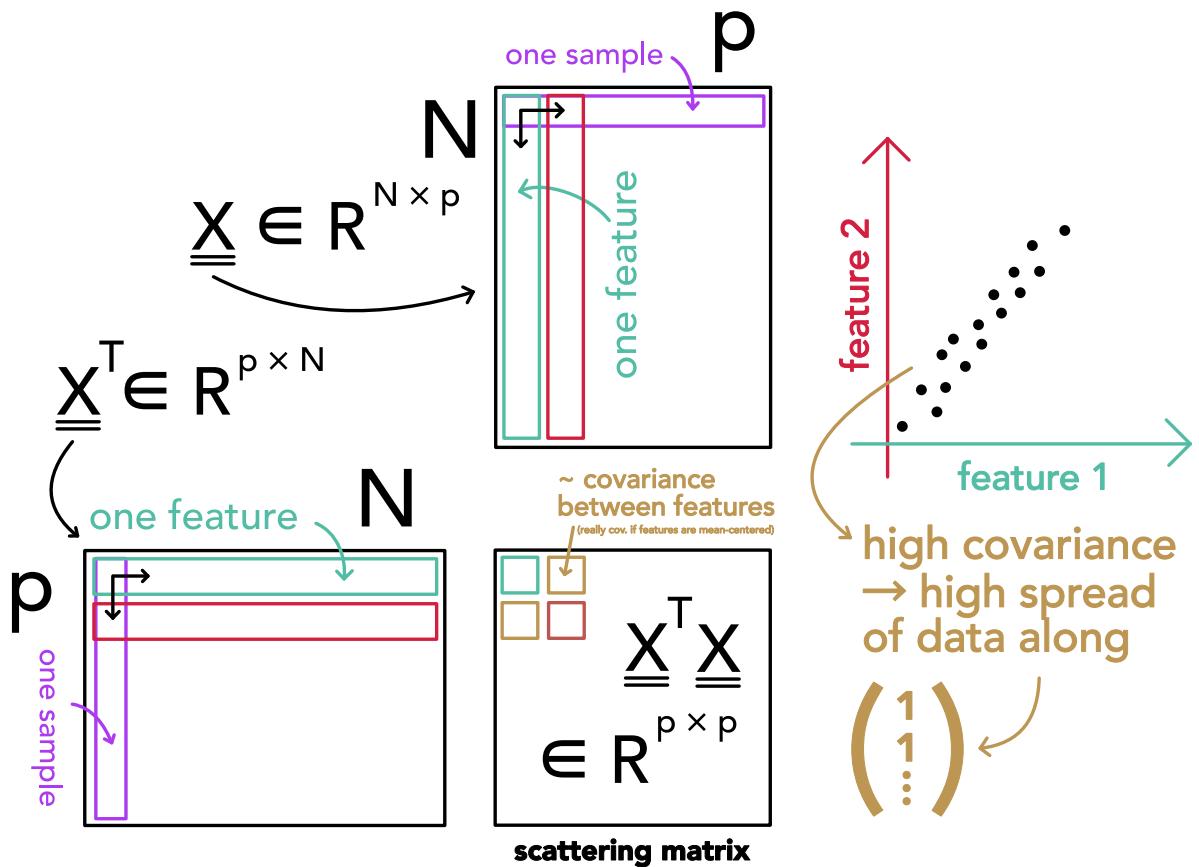


Figure 165: On how the scattering matrix measures *scattering*.

19.1.4.4 Confidence intervals for $\hat{\beta}_i$ - where is the true β_i^* ?

The estimates $\hat{\beta}_i$ have sampling uncertainty. Consider the limit where we draw lots of samples. Then in $1 - \alpha$ of those samples, the true β_i^* will be in the interval

$$\text{CI}_{1-\alpha}^{\beta_i} = \left(\hat{\beta}_i - z_{\alpha/2} \cdot \text{SE}(\hat{\beta}_i), \hat{\beta}_i + z_{\alpha/2} \cdot \text{SE}(\hat{\beta}_i) \right)$$

with $z_{\alpha/2}$ the $\alpha/2$ quantile of the standard normal distribution (996)

$$\text{and } \text{SE}(\hat{\beta}_i) = \sqrt{\sigma^2 \left((\underline{\underline{X}}^T \underline{\underline{X}})^{-1} \right)_{ii}}$$

19.1.4.5 Usage in hypothesis testing

Consider we want to test $H_0 : \beta_i = 0$ vs $H_1 : \beta_i \neq 0$. As $\hat{\beta}$ is normally distributed, we can use the t -test

$$\begin{aligned} t &= \frac{\hat{\beta}_i}{\text{SE}(\hat{\beta}_i)} \sim t_{N-p-1} \\ \text{SE}(\hat{\beta}_i) &= \sqrt{\hat{\sigma}^2 \left((\underline{\underline{X}}^T \underline{\underline{X}})^{-1} \right)_{ii}} \end{aligned} \quad (997)$$

here with estimate $\hat{\sigma}^2 = \frac{1}{N-p} \|\underline{Y} - \underline{\underline{X}}\hat{\beta}\|_2^2$

as of this estimation $t \sim t_{N-p-1}$ not normal

19.2 General linear model

A general linear model is a conventional linear regression model (with a continuous response variable) given **continuous and / or categorical predictors**. As before $y_i \sim \mathcal{N}(\underline{x}_i^T \underline{\beta}, \sigma^2)$.

An example could be that in for purely categorical \underline{x} , we have classes

$$\text{cow} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \text{manatee} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \text{elephant} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (998)$$

and we have measured weights, so a very small sample could be

$$\left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, 525.5 \text{ kg} \right\}, \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, 918.3 \text{ kg} \right\}, \left\{ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, 5485.6 \text{ kg} \right\}, \left\{ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, 480.1 \text{ kg} \right\} \quad (999)$$

so

$$\underline{y} = \begin{pmatrix} 525.5 \\ 918.3 \\ 5485.6 \\ 480.1 \end{pmatrix}, \quad \underline{\underline{X}} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (1000)$$

where we might want to add further predictors (e.g. age (continuous), gender (categorical), ...) to better predict the weight.

19.2.1 Purely categorical predictors

For $\underline{\underline{X}} \in \{0, 1\}^{N \times p}$, purely categorical (p categories) in 1-hot encoding, $\hat{\beta} \in \mathbb{R}^p$ will consist of the means over the y_i of the respective category.

Consider as in the example above the 1-hot predictor encoding

$$\underline{x}_i = \begin{pmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{pmatrix}, \quad x_{ij} \in \{0, 1\}, \quad \sum_{j=1}^p x_{ij} = 1 \quad (1001)$$

So how does a category being active influence the outcome?

The number of occurrences of each category are

$$N_j = \sum_{i=1}^N x_{ij} \quad (1002)$$

(in our example 1 cow, 2 manatees, 1 elephant).

We can calculate $\hat{\beta}$ as

$$\hat{\beta} = (\underline{X}^T \underline{X})^{-1} \underline{X}^T Y = \begin{pmatrix} N_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & N_p \end{pmatrix}^{-1} \begin{pmatrix} \sum_i y_i x_{i1} \\ \vdots \\ \sum_i y_i x_{ip} \end{pmatrix} = \begin{pmatrix} \frac{1}{N_1} \sum_i y_i x_{i1} \\ \vdots \\ \frac{1}{N_p} \sum_i y_i x_{ip} \end{pmatrix} = \begin{pmatrix} \bar{y}_{\text{cat.}} \\ \vdots \\ \bar{y}_{\text{cat.p}} \end{pmatrix} \quad (1003)$$

so the components of $\hat{\beta}_j$ is the mean of the outcome for the j -th category.

19.2.2 Testing if categories have the same mean

Let us make the null hypothesis of equal means

$$H_0 : \mu_1 = \mu_2 = \mu_3, \quad \bar{\beta} := \hat{\beta}_1 = \hat{\beta}_2 = \hat{\beta}_3 \quad (1004)$$

If the means are equal and as we have found that for each category we predict the mean, we can fit the full model

$$\text{full model (I): } y_i = \sum_{j=1}^p \beta_j x_{ij} + \epsilon_i \quad (1005)$$

compare the reduced model under H_0

$$\text{reduced model (II): } y_i = \bar{\beta} + \epsilon_i, \quad \bar{\beta} = \frac{1}{N} \sum_{i=1}^N y_i \quad (1006)$$

We can do an F-test based on the sums of squares of the residuals of the two models

$$\mathcal{F} = \frac{\left(\text{SSQ}(\underline{\beta}^{(II)}) - \text{SSQ}(\underline{\beta}^{(I)}) \right) / (p - q)}{\text{SSQ}(\underline{\beta}^{(I)}) / (N - p - 1)} \sim F_{p-q, N-p-1} \quad (1007)$$

p # parameters in the full model, q # parameters in the reduced model

19.3 Multivariate linear model | multiple outcomes

19.3.1 Multivariate model

Multiple linear model refers to multiple predictors $\underline{y}(\underline{x})$, in a multivariate linear model, we also have multiple outcomes $\underline{y}(\underline{x})$.

For instance based on many environmental variables, we want to predict temperature and humidity at the same time.

Let us write the multivariate data of N measurements of q outcomes given p predictors as

$$\underline{\underline{Y}} \in \mathbb{R}^{N \times q} \text{ with } \underline{y}_i \in \mathbb{R}^q, \quad \underline{\underline{X}} \in \mathbb{R}^{N \times p} \text{ with } \underline{x}_i \in \mathbb{R}^p \quad (1008)$$

and the **multivariate linear model** as

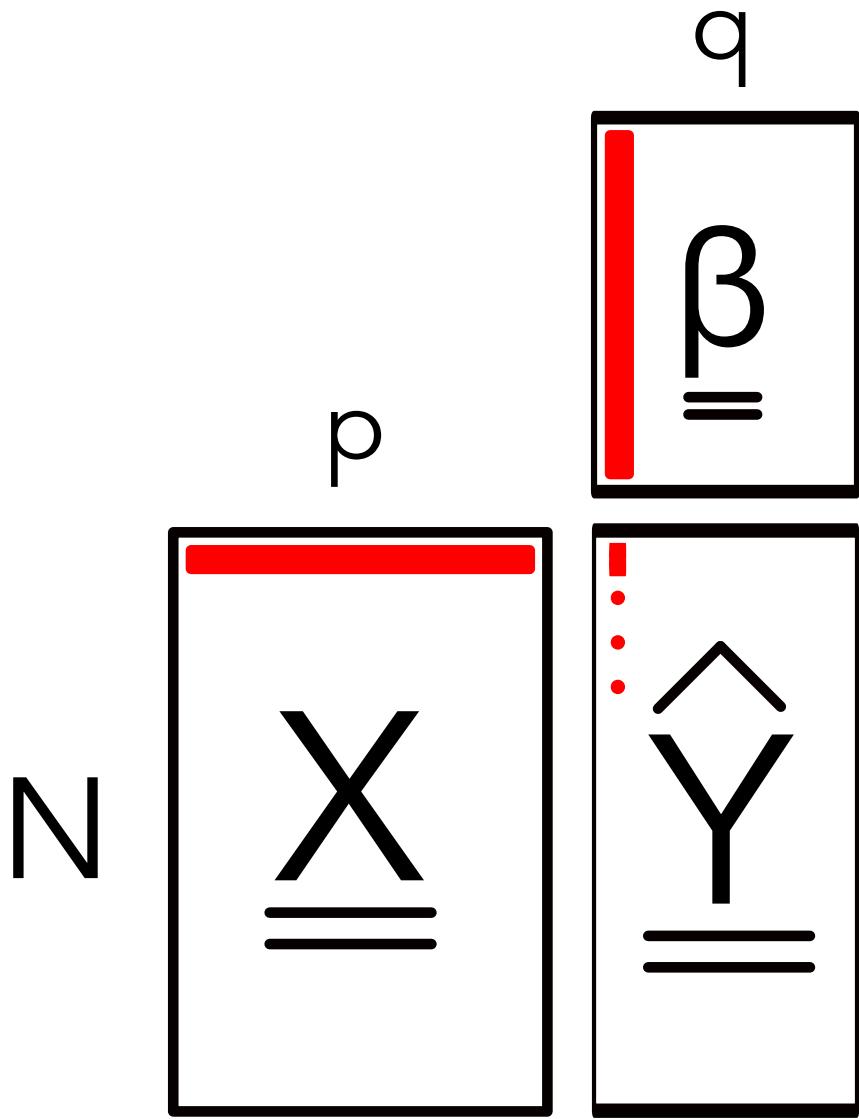
$$\begin{aligned} \underline{\underline{Y}} &= \underline{\underline{X}} \underline{\underline{B}} + \underline{\underline{E}}, & \text{parameter-matrix } \underline{\underline{B}} &\in \mathbb{R}^{p \times q} \\ && \text{error-matrix } \underline{\underline{E}} &\in \mathbb{R}^{N \times q} \end{aligned} \quad (1009)$$

19.3.2 Except for possible correlations in the error terms, it's just a concatenated model

Each column of $\underline{\underline{B}}$ represents the relationship between $\underline{\underline{X}}$ and one output variable (e.g. the first the temperature, the second the humidity), see figure 166.

$\hat{\underline{\underline{B}}}$ is given by

$$\hat{\underline{\underline{B}}} = \left(\underline{\underline{X}}^T \underline{\underline{X}} \right)^{-1} \underline{\underline{X}}^T \underline{\underline{Y}} \quad (1010)$$



every column of $\underline{\beta}$ fits one of the response variables, e. g. the temperature at one location

Figure 166: The multivariate linear model

19.3.3 The error matrix $\underline{\underline{E}}$ can contain correlations

Possible correlations between the outputs (e.g. correlation between temperature and humidity, or if y would be multiple temperature measurements at different locations their correlation.) can be encoded in $\underline{\underline{E}}$ as correlated error terms.

Each column of $\underline{\underline{E}}$ are the errors in one output variable over the N measurements in the sample

$$\underline{\underline{E}} = \begin{pmatrix} | & \cdots & | \\ \underline{\epsilon}_1 & \cdots & \underline{\epsilon}_q \\ | & \cdots & | \end{pmatrix} \quad (1011)$$

- measurements of a single output variable over the whole sample have uncorrelated errors

$$k\text{-th column of } \underline{\underline{E}}, \underline{e}_k \sim \mathcal{N}\left(\underline{0}_N, \sigma_{kk}^2 \underline{\underline{1}}_{N \times N}\right) \quad (1012)$$

- error terms in over different outputs in one measurement can be correlated

$$i\text{-th row of } \underline{\underline{E}}, \underline{e}_i \sim \mathcal{N}\left(\underline{0}_{-q}, \underline{\Sigma}\right), \quad \underline{\Sigma} \text{ covariance matrix } \in \mathbb{R}^{q \times q} \quad (1013)$$

19.3.4 Distribution of $\hat{\underline{\underline{B}}}$

Let vec be the vectorization (column under column) of a matrix. Then

$$\text{vec } \hat{\underline{\underline{B}}} \sim \mathcal{N}\left(\text{vec } \underline{\underline{B}}^*, \left(\underline{\underline{X}}^T \underline{\underline{X}}\right)^{-1} \otimes \underline{\Sigma}\right) \quad (1014)$$

with \otimes the Kronecker product⁴².

So the difference between doing separate linear models for each output variable is that we have an error matrix with possible correlations and thus a more intricate sampling distribution of $\hat{\underline{\underline{B}}}$. Possible correlation of outputs shows in the sampling distribution of $\hat{\underline{\underline{B}}}$.

19.3.5 Hypothesis testing in the multivariate case - considering parameters jointly

Consider we have a two-dimensional output $\underline{y}_i \in \mathbb{R}^2$ and a categorical predictor $\underline{x}_i \in \mathbb{R}^3$ with three categories A, B, C . If we consider the parameters jointly, we can formulate more complicated combined hypotheses, e.g. $H_0 : \hat{\beta}_{y_1}^2 + \hat{\beta}_{y_2}^2 < c$. This is illustrated in figure 167.

⁴²Example Kronecker product:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \otimes \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} = \begin{pmatrix} 1 \cdot \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} & 2 \cdot \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} \\ 3 \cdot \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} & 4 \cdot \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} \\ 5 \cdot \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} & 6 \cdot \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 7 & 8 & 14 & 16 \\ 9 & 0 & 18 & 0 \\ 21 & 24 & 28 & 32 \\ 27 & 0 & 36 & 0 \\ 35 & 40 & 42 & 48 \\ 45 & 0 & 54 & 0 \end{pmatrix} \quad (1015)$$

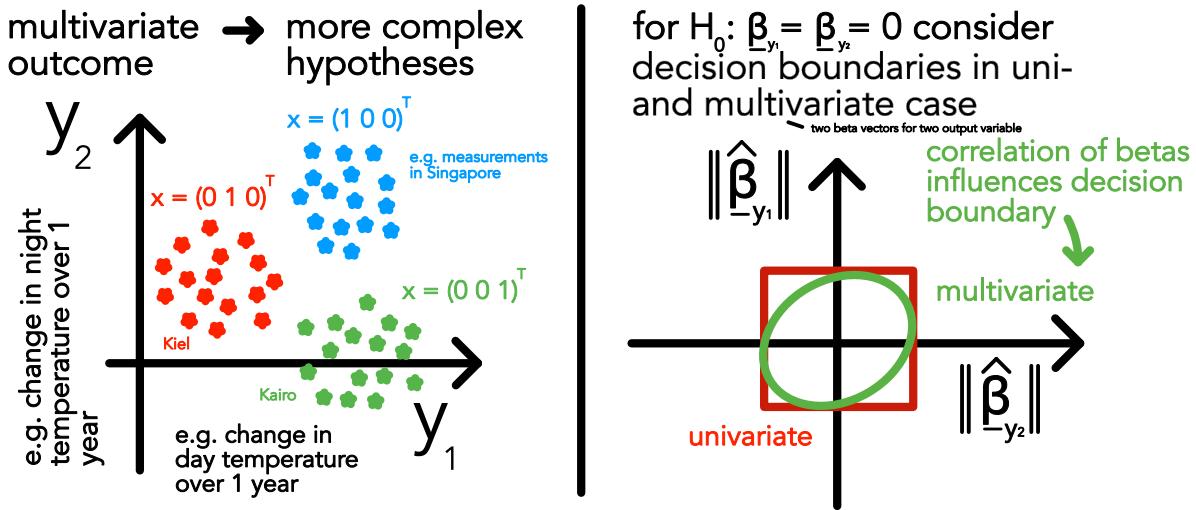


Figure 167: Hypothesis testing on linear regression parameters in the multivariate (multiple outputs) case

19.4 Regression Models Suitable for Non-Linear Relationships

Let us go back to the univariate case. We have a sample $\{\underline{x}_i, y_i\}_{i=1}^N$ and we want to model the relationship between \underline{x} and y , so given a new \underline{x} (e.g. new climate parameters) we can predict y (e.g. a new temperature).

Now consider the relationship between \underline{x} and y is not linear, as illustrated in figure 168.

Our principal options of going about this are **adaptations of simple linear regression**, still linear in the parameters

- **feature engineering:** linear regression is linear in the parameters, not necessarily in the original predictors - we can just add e.g. polynomial features as columns to \underline{X} , or evaluations of any other base function. This is **very robust** if we know appropriate base functions.
- **spline regression:** in the polynomial feature engineering approach, we get flexible smooth curves, but one problem is that they are trained globally, therefore we can use piecewise polynomial regression (splines), fitted over different regions
- **local linear regression:** make the regression parameters $\underline{\beta}$ dependent on \underline{x} , so we can model non-linear relationships.

or using the **framework** of

- **generalized linear models (GLM):** A mean of a distribution is modeled via a link function applied to $\underline{\beta}^T \underline{x}$, so as of the link function the modeled mean can be non-linear in the predictors and parameters.

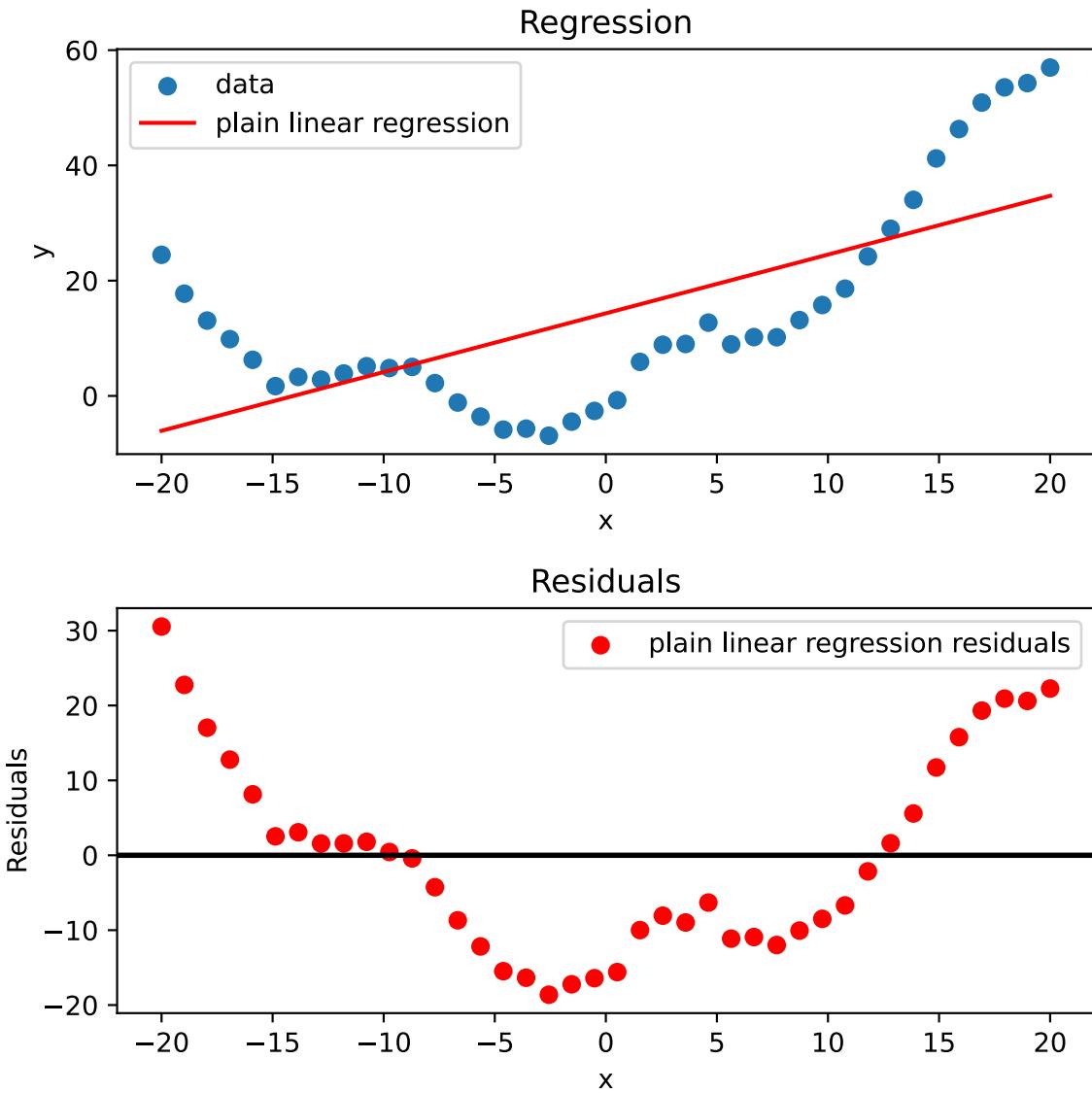


Figure 168: For this sample, a model linear in the predictors is not suitable.

A more sophisticated model, which can also be obtained from generalizing linear regression to a possibly infinitely large number of basis function is

- **Gaussian process regression:** set in Bayesian perspective; based on introducing a kernel, an infinite basis function expansion can be done

Or alternatively approaches from the *algorithmic modeling culture* (more complex or non parametric models)

- **k -nearest neighbors regression:** approximate $y(\underline{x})$ by the average of y_i of the k nearest neighbors of \underline{x} in the sample
- **kernel regression:** based on k -nearest neighbor regression, weighting the neighbors

by a kernel function

- **decision tree regression:** recursive partitioning of the feature space, fitting a constant in each region
- **regression by neural networks:** possibly very flexible model
- ...

which can have the advantages of **good predictive performance**, but the disadvantage of being **more difficult to interpret**.

19.4.1 Regression with Feature Engineering

In the example of figure 168, the true relationship is

$$y_i = x_i + 0.1x_i^2 + 5 \sin(0.5x_i) + \epsilon_i \quad (1016)$$

Using a feature matrix

$$\underline{\underline{X}} = \begin{pmatrix} 1 & x_1 & x_1^2 & \sin(0.5x_1) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & \sin(0.5x_N) \end{pmatrix} \quad (1017)$$

we can again just use our linear regression model, see figure 169.

Generally, by feature engineering

$$\underline{x} \in \mathbb{R}^p \mapsto \underline{\phi}(\underline{x}) \in \mathbb{R}^{p'}, \quad p' \geq p \quad (1018)$$

and the data matrix is transformed to

$$\underline{\underline{X}} \in \mathbb{R}^{N \times p} \mapsto \underline{\underline{\Phi}} \in \mathbb{R}^{N \times p'} \quad (1019)$$

so the linear regression model is

$$\underline{y} = \underline{\underline{\Phi}}\beta + \underline{\epsilon} \quad (1020)$$

with the least squares estimate

$$\hat{\beta} = (\underline{\underline{\Phi}}^T \underline{\underline{\Phi}})^{-1} \underline{\underline{\Phi}}^T \underline{y} \quad (1021)$$

Note: This is still linear in the parameters. Using $\sin(\beta_s x)$ with a parameter in the function is not possible. So we have to choose the basis functions wisely.

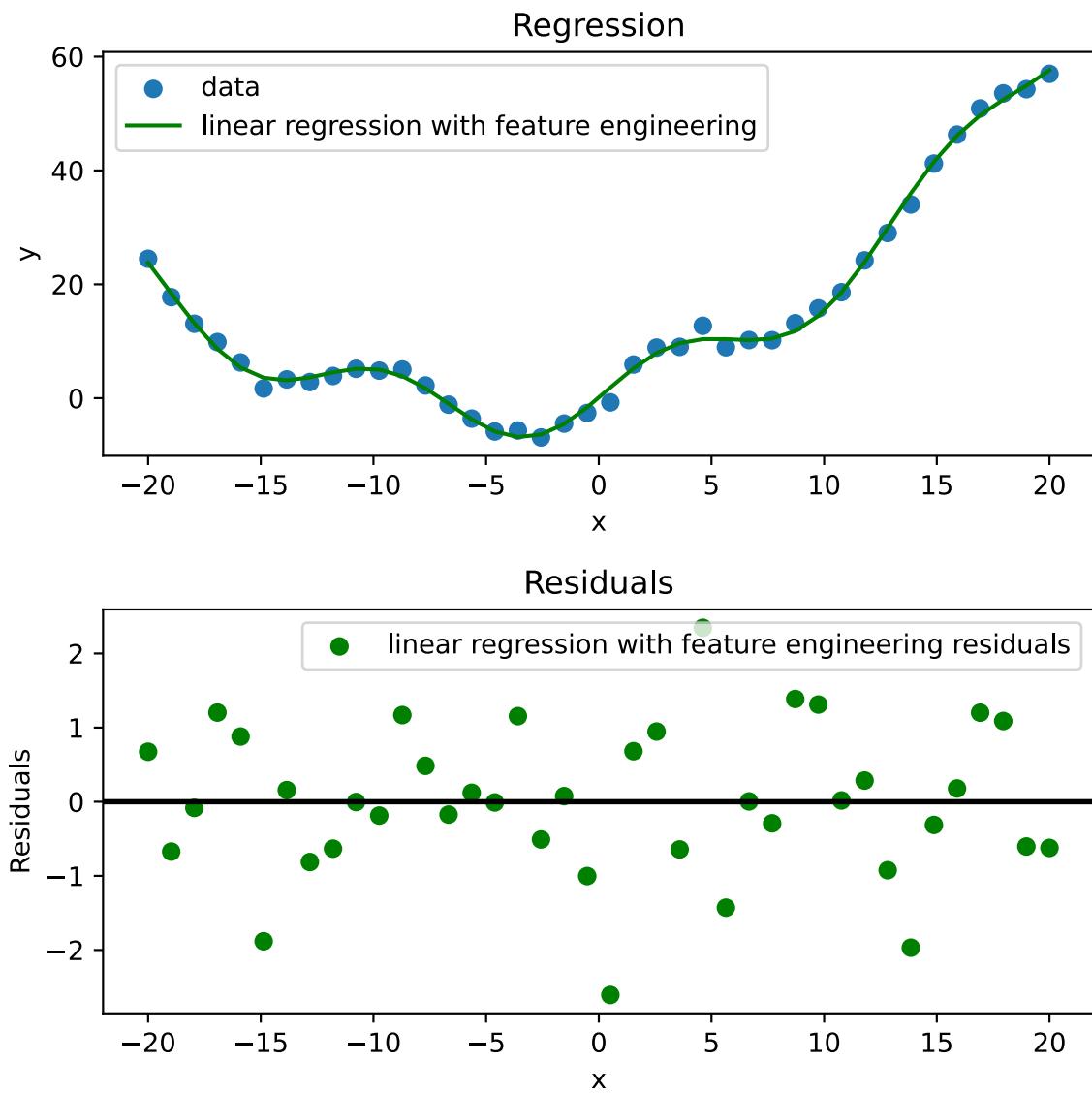


Figure 169: Feature engineering for a non-linear relationship.

19.4.2 Spline Regression

In spline regression, we do a piecewise polynomial regression.

The **local model** is just linear regression with feature engineering

$$\phi(\underline{x}) = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_p \\ x_1^2 \\ x_1 x_2 \\ \vdots \\ x_p^2 \\ \vdots \end{pmatrix} \quad (1022)$$

For the **global model**, we stitch polynomials pieces together at so called-knots, so in a model with one knot, we have (for originally 1D x)

$$\begin{aligned} y_i &= y_A = \underline{\beta}_A^T \phi(x), & x_i < c \\ y_i &= y_B = \underline{\beta}_B^T \phi(x), & x_i \geq c \end{aligned} \quad (1023)$$

where the parameters $\underline{\beta}_A, \underline{\beta}_B$ are further restricted by

$$y_A(c) = y_B(c), \quad \partial_x y_A(c) = \partial_x y_B(c), \quad \partial_x^2 y_A(c) = \partial_x^2 y_B(c) \quad (1024)$$

so demanding the function to be continuously differentiable. Each equations constrains one parameter in $\underline{\beta}_A, \underline{\beta}_B$.

A cubic piecewise polynomial with these three constraints is called a cubic spline.

A global model is illustrated in figure 170.

knit together polynomials

- knots, number of which is hyperparameter
↑spline regression must go through

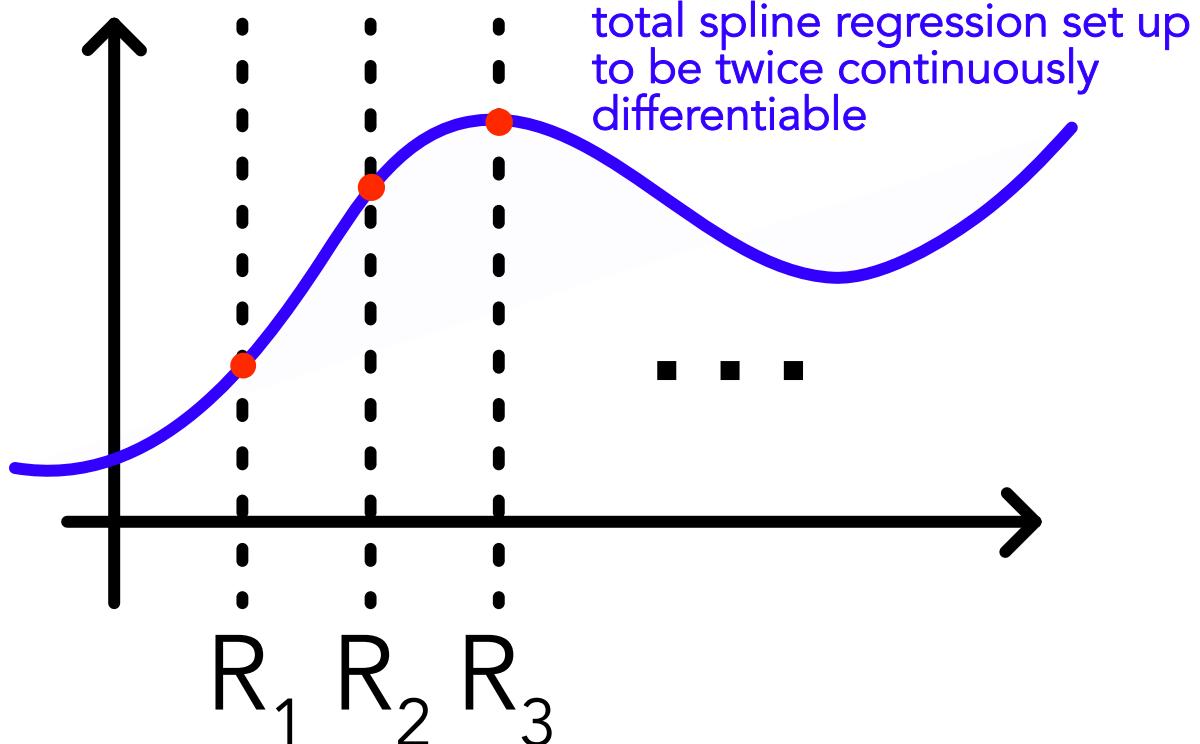


Figure 170: Spline regression.

19.4.3 Local Linear Regression

19.4.3.1 Model and parameter estimates in local linear regression

Idea: In local linear regression, we at each point of interest fit a linear model locally, which is done by introducing a weighting kernel into the sum of squared residuals centered at the point of interest. The coefficients $\underline{\beta}$ are then dependent on \underline{x} .

Model At a given (test) point \underline{x}_t , our local linear model evaluates to

$$\hat{y}_t = \underline{\beta}(\underline{x}_t)^T \underline{x}_t \quad (1025)$$

(\underline{x} might also be a transformed version of \underline{x} , e.g. $\phi(\underline{x})$).

Loss and Kernel $\hat{\underline{\beta}}$ follows from the loss (on $D = \{(\underline{x}_i, y_i)\}_{i=1}^N$) being

$$L(\underline{\beta}(\underline{x}_t)) = \sum_{i=1}^N K_\lambda(\underline{x}_t, \underline{x}_i)(y_i - \underline{\beta}(\underline{x}_t)^T \underline{x}_i)^2 \quad (1026)$$

with $K_\lambda(\underline{x}_t, \underline{x}_i)$ being a kernel fulfilling

$$\sum_{i=1}^N K_\lambda(\underline{x}_t, \underline{x}_i) = 1, \quad K_\lambda(\underline{x}_t, \underline{x}_i) \geq 0 \quad (1027)$$

λ in K_λ is a bandwidth parameter, controlling the width of the kernel and effectively the smoothness of our model (more later).

Using the weighting matrix

$$\underline{\underline{W}}_\lambda(\underline{x}_t) = \text{diag}(K_\lambda(\underline{x}_t, \underline{x}_1), \dots, K_\lambda(\underline{x}_t, \underline{x}_N)) \quad (1028)$$

we can write the loss (sum of squares) as

$$L(\underline{\beta}_{\underline{x}_t}) = \|\underline{\underline{W}}_{\lambda, \underline{x}_t}^{1/2} (\underline{Y} - \underline{\underline{X}} \underline{\beta}(\underline{x}_t))\|_2^2 \quad (1029)$$

Estimate for $\underline{\beta}(\underline{x}_t)$ From the loss we can obtain the estimate (same situation as in the usual linear regression)

$$\hat{\underline{\beta}}_{\underline{x}_t} = \left(\underline{\underline{X}}^T \underline{\underline{W}}_{\lambda, \underline{x}_t}^{-1} \underline{\underline{X}} \right)^{-1} \underline{\underline{X}}^T \underline{\underline{W}}_{\lambda, \underline{x}_t}^{-1} \underline{Y} \quad (1030)$$

Example Kernel

$$K_\lambda(\underline{x}_t, \underline{x}_j) = \frac{\exp\left(-\frac{\|\underline{x}_t - \underline{x}_j\|^2}{\lambda^2}\right)}{\sum_{i=1}^N \exp\left(-\frac{\|\underline{x}_t - \underline{x}_i\|^2}{\lambda^2}\right)} \quad (1031)$$

19.4.3.2 Choosing the bandwidth λ - the hyperparameter of local linear regression

The larger λ the more points are considered in the regression and for $\lambda \rightarrow \infty$ we just have linear regression (most likely underfitting), while for λ too small overfitting to the data will occur and no meaningful information is extracted.

This is illustrated in figure 171.

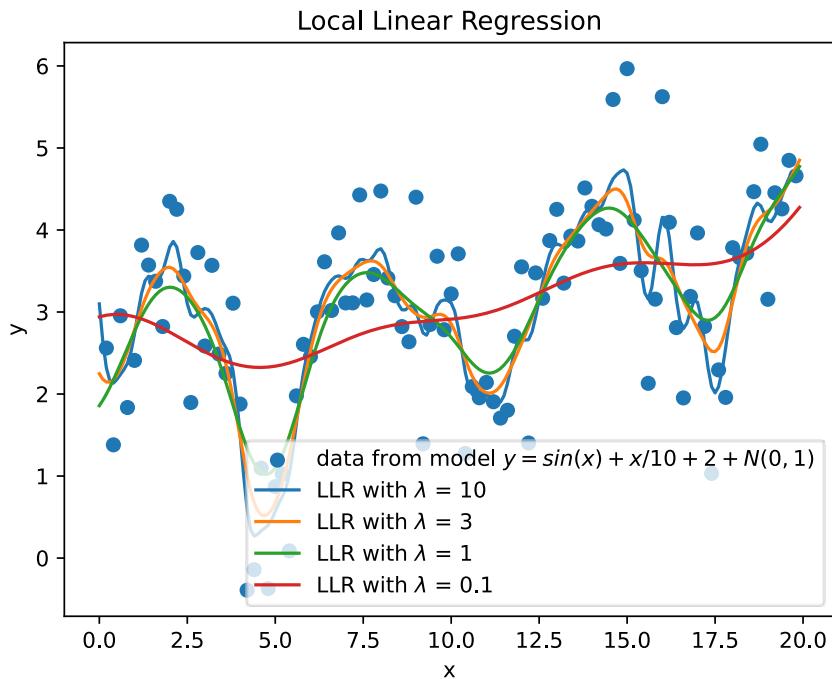


Figure 171: Local linear regression for different bandwidths λ .

19.4.4 Generalized Linear Models (GLM)*

Idea: Our prediction is the mean of a distribution from the exponential family modeled via a link function where one finds the parameters by maximizing the likelihood of the data.

So for any given \underline{x} we have the same distribution over y but with a different mean depending on \underline{x} .

19.4.4.1 Repetition on Linear Regression

Remember that in a simple linear regression Model, we assume the underlying model to be

$$y(\underline{x}) = \underline{\beta}^T \underline{x} + \epsilon(\underline{x}), \quad \epsilon(\underline{x}) \sim \mathcal{N}(0, \sigma^2) \quad (1032)$$

so linear in the parameters $\underline{\beta}$ and we assume a normal distribution of the noise $\epsilon(\underline{x})$. We can also write this as

$$p(y|\underline{x}, \underline{\beta}, \sigma^2) = \mathcal{N}(y|\underline{\beta}^T \underline{x}, \sigma^2), \quad E[y|\underline{x}, \underline{\beta}, \sigma^2] = \underline{\beta}^T \underline{x} \quad (1033)$$

Naturally, such a model is not well suited for binary data or only positive data, ..., so we need to generalize this.

19.4.4.2 Generalized Linear Models

In the Generalized Linear Model, we assume a distribution which mean is modeled using

$$E[y|\underline{x}, \underline{\beta}] = \mu = g^{-1}(\eta), \quad \eta = \underline{\beta}^T \underline{x} \quad (1034)$$

where g is called the link function and g^{-1} the mean function. The distribution of which we model the mean is from the exponential family

$$p(y|\theta) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)\right), \quad (1035)$$

with real functions a, b, c , and parameters θ, ϕ

for which holds

$$E[y] = \mu = \partial_\theta b(\theta) \quad (1036)$$

connecting the parameter θ to the mean we model as

$$\mu = g^{-1}(\eta) = \partial_\theta b(\theta) \quad (1037)$$

where for the canonical link function $\eta = \theta$.

Given we choose a distribution from the exponential family with parameters θ to model y and some function modeling the mean of this distribution $\mu = g^{-1}(\eta), \eta = \underline{\beta}^T \underline{x}$, we can

1. relate the model parameters θ to the predictors $\eta = \underline{\beta}^T \underline{x}$ via $\mu = g^{-1}(\eta) = \partial_\theta b(\theta)$
2. from given \underline{x} and $\underline{\beta}$ we can therefore calculate θ and with this the likelihood of some y
3. by maximizing the likelihood over a sample, we can find the parameters $\hat{\underline{\beta}}$
4. using the mean function $g^{-1}(\eta)$ (with the estimated $\hat{\underline{\beta}}$) we can predict the mean of the distribution of y given an unknown \underline{x}

Also, one can calculate the variance as

$$\text{Var}[Y] = a(\phi) \partial_\theta^2 b(\theta) \quad (1038)$$

19.4.4.3 Finding the parameters $\underline{\beta}$ for a GLM

The parameters $\underline{\beta}$ are found by maximizing the likelihood of the data, or minimizing the negative log-likelihood, so

$$\hat{\underline{\beta}} = \underset{\underline{\beta}}{\operatorname{argmin}} \left(- \sum_{i=1}^N \log p(y_i | \underline{x}_i, \underline{\beta}) \right) \quad (1039)$$

19.4.4.4 Canonical Link Functions for a Bernoulli GLM - Logistic Regression

Here our response variable is $y \in \{0, 1\}$.

Consider the binomial distribution given by

$$\begin{aligned} f(y|p) &= p^y (1-p)^{1-y} \\ &= \exp \left(y \underbrace{\log \frac{p}{1-p}}_{\text{define } \theta} - [-\log(1-p)] \right) \\ &= \exp \left(y\theta - \log \left(1 + \frac{1}{1-p} \right) \right) \\ &= \exp(y\theta - \log(1 + \exp \theta)) \\ &= \exp(y\theta - b(\theta)) \end{aligned} \quad (1040)$$

with $a(\phi) = 1$, $c(y, \phi) = 0$, $b(\theta) = \log(1 + \exp \theta)$ and $\theta = \log \frac{p}{1-p}$. From this we can find the well-known mean as

$$\mu = \partial_\theta b(\theta) = \frac{\exp \theta}{1 + \exp \theta} = p \quad (1041)$$

so for the canonical link ($\eta = \theta$) we have the mean function (a sigmoid)

$$g^{-1}(\eta) = \mu = \frac{\exp \eta}{1 + \exp \eta} \quad (1042)$$

and thus the link function (log-odds)

$$g(\mu) = \eta = \log \frac{\mu}{1 - \mu} \quad (1043)$$

19.4.4.5 Complementary Log-Log Link Function

Let us discuss the general route. Our model is $f(y|p) = p^y (1-p)^{1-y}$ and we choose the link function

$$g(\mu) = \eta = \log(-\log(1 - \mu)) \quad (1044)$$

and the mean function is

$$g^{-1}(\eta) = \mu = 1 - \exp(-\exp \eta) \quad (1045)$$

1. In the Bernoulli model, the model-parameter p is the mean, $p = \mu = g^{-1}(\eta)$
2. Given \underline{x} and $\underline{\beta}$, the likelihood therefore is

$$p(y|\underline{x}, \underline{\beta}) = (g^{-1}(\underline{\beta}^T \underline{x}))^y (1 - g^{-1}(\underline{\beta}^T \underline{x}))^{1-y} \quad (1046)$$

3. We therefore find the parameters $\hat{\underline{\beta}}$ by maximizing the likelihood on $D = \{(\underline{x}_i, y_i)\}_{i=1}^N$ with $\eta_i = \underline{\beta}^T \underline{x}_i$ by

$$\begin{aligned} \hat{\underline{\beta}} &= \operatorname{argmin}_{\underline{\beta}} \left(- \sum_{i=1}^N \log p(y_i|\underline{x}_i, \underline{\beta}) \right) \\ &= \operatorname{argmin}_{\underline{\beta}} \left(- \sum_{i=1}^N y_i \log(g^{-1}(\eta_i)) + (1 - y_i) \log(1 - g^{-1}(\eta_i)) \right) \\ &= \dots \\ &= \operatorname{argmin}_{\underline{\beta}} \left(- \sum_{i=1}^N y_i \log(\exp(\exp \underline{\beta}^T \underline{x}_i) - 1) - \exp \underline{\beta}^T \underline{x}_i \right) \end{aligned} \quad (1047)$$

4. Based on those parameters, we can predict the mean of the distribution of y given an unknown \underline{x}

A cloglog-link is useful, when we observe and count events and observe either 0 (denoted by $y = 0$) or more events (denoted by $y = 1$).

An example cloglog-fit is shown in figure 172.

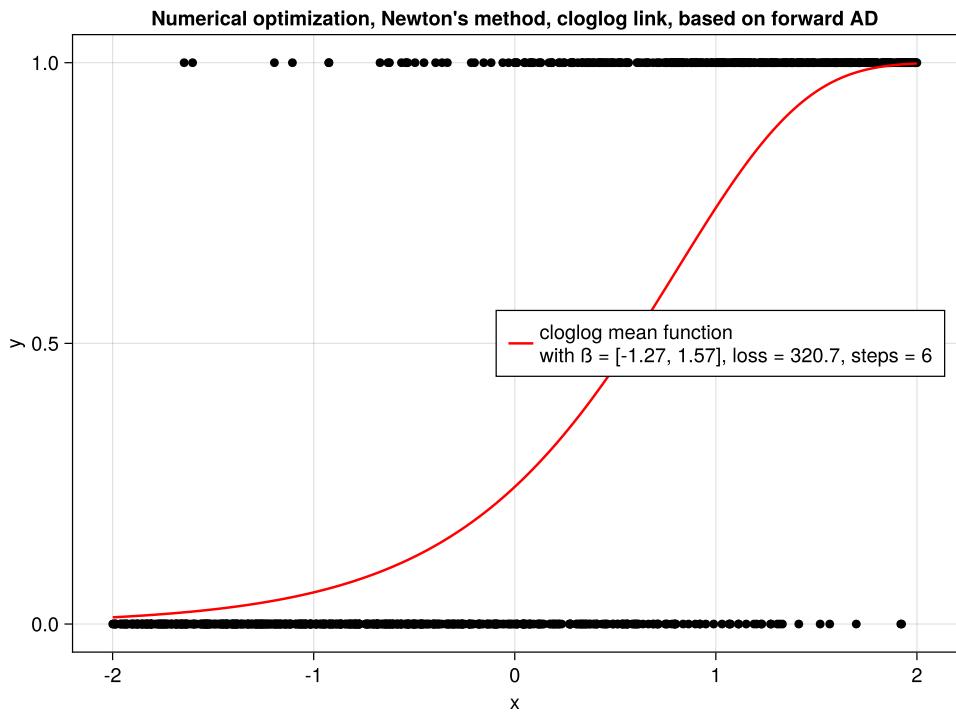


Figure 172: Complementary log-log fit.

19.4.5 Gaussian Process Regression*

19.4.5.1 Weight-space view - generalization of linear regression

Given parameters $\underline{\beta} \in \mathbb{R}^p$ and data $\underline{\underline{X}} \in \mathbb{R}^{N \times p}$, $\underline{Y} \in \mathbb{R}^N$ (N measurements in the sample) is distributed as

$$\underline{Y} \sim \mathcal{N}(\underline{\underline{X}}\underline{\beta}, \sigma^2 \underline{\underline{I}}) \quad (1048)$$

Now assume that the parameters $\underline{\beta}$ have a prior distribution

$$\underline{\beta} \sim \mathcal{N}(\underline{0}, \sigma^2 \underline{\underline{\Lambda}}_0^{-1}) \quad (1049)$$

Note: In a non-Bayesian setting this is equivalent to a regularization term in the loss function, e.g. the L_2 -norm of $\underline{\beta}$, see Williams and Rasmussen, 1995 for more details.

Based on the data $\underline{\underline{X}}, \underline{Y}$ the posterior of the parameters is

$$\underline{\beta} | \underline{\underline{X}}, \underline{Y} \sim \mathcal{N}\left(\underline{\beta}_{\text{post}}, \sigma^2 \underline{\underline{\Lambda}}_{\text{post}}^{-1}\right) \quad (1050)$$

with

$$\underline{\underline{\Lambda}}_{\text{post}} = \underline{\underline{X}}^T \underline{\underline{X}} + \underline{\underline{\Lambda}}_0, \quad \underline{\beta}_{\text{post}} = \underline{\underline{\Lambda}}_{\text{post}}^{-1} \underline{\underline{X}}^T \underline{Y} \quad (1051)$$

With this we can calculate the distribution of an averaged output y_* over all possible parameters $\underline{\beta}$ evaluated at a new \underline{x}_* as

$$\begin{aligned} p(y_* | \underline{x}_*, \underline{\underline{X}}, \underline{\underline{Y}}) &= \int p(y_* | \underline{x}_*, \underline{\beta}) p(\underline{\beta} | \underline{\underline{X}}, \underline{\underline{Y}}) d\underline{\beta} \\ &= \mathcal{N}\left(\underline{x}_*^T \underline{\beta}_{\text{post}}, \underline{x}_*^T \sigma^2 \underline{\Lambda}_{\text{post}}^{-1} \underline{x}_*\right) \end{aligned} \quad (1052)$$

We then do feature expansion

$$\underline{x} \in \mathbb{R}^p \mapsto \underline{\phi}(\underline{x}) \in \mathbb{R}^{p'}, \quad p' \geq p \quad (1053)$$

and rewrite the above expression only in terms of the kernel function

$$k(\underline{x}, \underline{x}') = \underline{\phi}(\underline{x})^T \sigma^2 \underline{\Lambda}_{\text{post}}^{-1} \underline{\phi}(\underline{x}') = \underline{\psi}(\underline{x})^T \underline{\psi}(\underline{x}') \quad (1054)$$

which we replace by a function directly operating on the low-dimensional input space \underline{x} , which allows us to model an expression equivalent to one would obtain from an infinite basis function expansion, e.g. by

$$k(\underline{x}, \underline{x}') = \exp\left(-\frac{1}{2} \|\underline{x} - \underline{x}'\|^2\right) \quad (1055)$$

See Williams and Rasmussen, 1995 for the continuation.

At some point continue...

20 Bias-Variance Tradeoff and dealing with model complexity

We are again starting with some sample data $\mathcal{D} = \{(\underline{x}_i, y_i)\}_{i=1}^n$ which we assume to have been generated by a model

$$y = f_{\theta^*}(\underline{x}) + \epsilon \quad (1056)$$

Aim: For unseen data, we want to predict y from \underline{x} as accurately as possible.

We find the model by minimizing a loss, e.g. the mean squared error between the data in the sample and the model's prediction.

We ask ourselves:

- What are the limiting factors to generalizability?
- How do we measure how good our model generalizes to unseen data?

20.1 Bias-Variance-Tradeoff

20.1.1 Model Bias and Variance

There are two principal aspects to generalizability

- **bias error:** error from erroneous assumptions in our model compared to the true model
 - the stronger we constrain our parameters (by introducing a bias), the higher the likelihood of our model missing relations between features and target output
- **variance error:** our sample - our *training set* - contains random noise, when our model picks up on that noise (i.e. is very sensitive to that noise), different samples will lead to very different models (the model (/ model parameters) have a high sampling variance), which also limits generalizability

Model complexity is how freely we allow our model to fit the data, how large the space of possible models is. Naturally for a complex model, we assume a high variance and low bias, and for a simple model, we assume a low variance and high bias. This is the bias-variance tradeoff.

20.1.2 Bias-Variance Tradeoff

Classically, the fundamental limitation to a perfect model is, that increasing the bias decreases the variance and vice versa. By making stronger assumptions on our model parameters we can reduce their sensitivity to the noise in our sample, but also loose explanation-power. While we might find a sweet spot, we can never eliminate both.

Let us illustrate the Bias-Variance tradeoff at the hand of Local Linear Regression, see table 35.

- small support λ : high variance, low bias
- large support λ : low variance, high bias

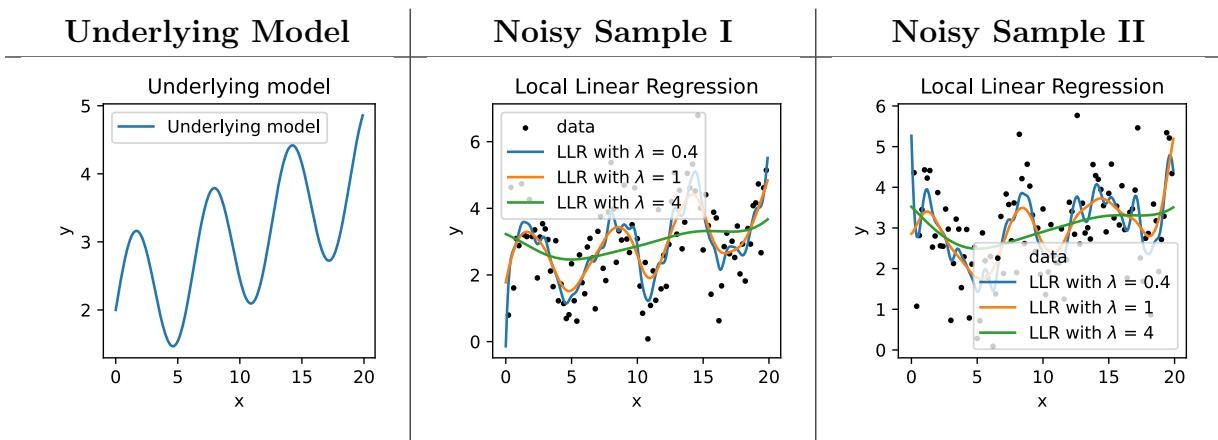


Table 35: Local Linear Regression: Underlying model and two noisy samples. The more complex model with smaller support has higher variance. The simpler model with larger support has higher bias.

Further examples of situations where we must choose model complexity, and thus face the bias-variance tradeoff, are

- how many features to include in a linear regression
- how many layers and nodes to include in a neural network
- ...

Hyperparameters: Parameters of the model not learned in the direct optimization process; based on them, like the support λ in Local Linear Regression, we can control the model's complexity.

20.2 Measuring generalization: Train and Test error and their relation to Bias and Variance

Idea: We can measure generalization by only training our model on a subset of the available data and then testing it on the rest (e.g. calculating the sum of squared residuals).

- Train error: error on the training set - error as of bias (no variance on a single sample)
- Test error: error on the test set - total error as of bias and variance

More accurate ways of obtaining a good model error (that will itself generalize well) is obtained via Cross-Validation, as discussed later.

20.2.1 Bias-Variance-Decomposition of the (Mean-Squared) prediction error

Consider the training data $\mathcal{D} = \{(\underline{x}_i, y_i)\}_{i=1}^n$ with

$$y_i = f_{\theta^*}(\underline{x}_i) + \epsilon_i, \quad \text{true model } f_{\theta^*} \text{ noise } \epsilon_i \text{ with } E[\epsilon_i] = 0, \text{Var}[\epsilon_i] = \sigma^2 \quad (1057)$$

from which we train a model $\hat{f}_{\underline{\theta}}(\underline{x})$.

20.2.1.1 Definition of the prediction error

We want to evaluate our model on $\underline{x}_0 \notin \mathcal{D}$, i.e. on unseen data, with our estimate being $\hat{y}_0 = \hat{f}_{\underline{\theta}}(\underline{x}_0)$ and the true value being y_0 . We define the prediction error as

$$\text{PE} := E[(y_0 - \hat{y}_0)^2] \quad (1058)$$

Note: As $\hat{f}_{\underline{\theta}}$ was trained on \mathcal{D} , PE depends on \mathcal{D} .

20.2.2 Bias-Variance-Decomposition of the prediction error

We can decompose (proof follows)

$$\begin{aligned} \text{PE} &= E[(y_0 - \hat{y}_0)^2] \\ &= E[(y_0 - \hat{f}_{\underline{\theta}}(\underline{x}_0))^2] \\ &= \text{bias}^2 + \text{variance} + \sigma^2 \\ &= \left(E[\hat{f}_{\underline{\theta}}(\underline{x}_0)] - f_{\underline{\theta}}(\underline{x}_0) \right)^2 + E \left[\left(\hat{f}_{\underline{\theta}}(\underline{x}_0) - E[\hat{f}_{\underline{\theta}}(\underline{x}_0)] \right)^2 \right] + E[(y_0 - f(\underline{x}_0))^2] \end{aligned} \quad (1059)$$

$E[\hat{f}_\theta(\underline{x}_0)]$ results from calculating $\hat{f}_\theta(\underline{x}_0)$ on different training sets and averaging the results.

The terms in the decomposition are

- **bias**: error due to simplifying assumptions in the model (e.g. bias for the model to be linear)
- **variance**: the stray of $\hat{f}_\theta(\underline{x}_0)$ from its mean, which is the larger, the more sensitive the model is to the difference between samples due to noise
- σ^2 : the noise in the true model, irreducible error

20.2.3 Derivation of the Bias-Variance-Decomposition

Let us abbreviate

$$\hat{f} := \hat{f}_\theta(\underline{x}_0), \quad f := f_\theta(\underline{x}_0), \quad y := y_0 \quad (1060)$$

we can then write

$$\text{PE} = E[(y - \hat{f})^2] = E[y^2] - 2E[y\hat{f}] + E[\hat{f}^2] \quad (1061)$$

- the third term $E[\hat{f}^2]$ is rewritten using the general $\text{Var}[X] = E[X^2] - E[X]^2$

$$E[\hat{f}^2] = \text{Var}[\hat{f}] + E[\hat{f}]^2 \quad (1062)$$

- the first term $E[y^2]$ is rewritten using $y = f + \epsilon$ and as f deterministic (independent of the sample) $E[f] = f$

$$E[y^2] \underset{E \text{ linear}}{=} E[f^2] + 2E[f\epsilon] + E[\epsilon^2] = f^2 + 2f \underbrace{E[\epsilon]}_{=0} + E[\epsilon^2] = f^2 + \sigma^2 \quad (1063)$$

in the last step using $E[\epsilon^2] = \text{Var}[\epsilon] + \underbrace{E[\epsilon]}_{=0}^2 = \sigma^2$.

- the second term $E[y\hat{f}]$ is rewritten to

$$E[y\hat{f}] = E[(f + \epsilon)\hat{f}] = E[f\hat{f}] + E[\epsilon\hat{f}] \underset{\epsilon, \hat{f} \text{ indep.}}{=} fE[\hat{f}] + E[\epsilon]E[\hat{f}] = fE[\hat{f}] \quad (1064)$$

where plugging everything back in gives the decomposition.

20.2.4 Striking a balance between Bias and Variance

Consider the total error PE as previously derived.

- Bias has a negative first-order derivative in response to model complexity
- Variance has a positive first-order derivative in response to model complexity

At the optimum complexity, the total error is minimal and

$$\partial_{\text{model complexity}} \text{bias} = -\partial_{\text{model complexity}} \text{variance} \quad (1065)$$

- exceeding this spot yields *overfitting* - the model is too complex and fits the noise in the sample
- falling short of this spot yields *underfitting* - the model is too simple and misses relations in the data

The bias variance tradeoff is illustrated in figure 173.

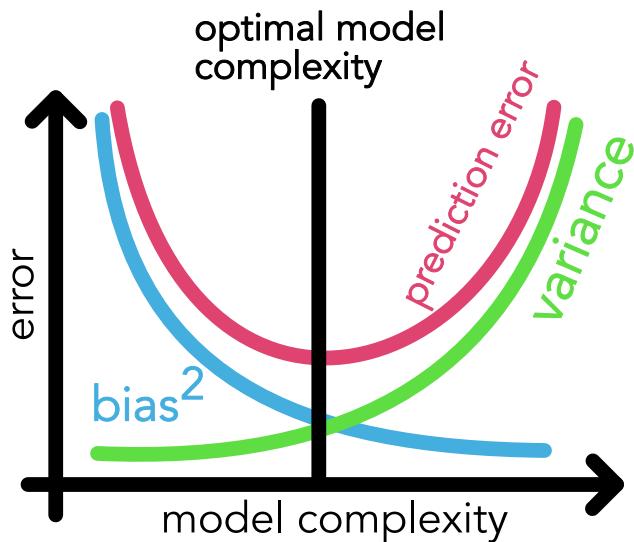


Figure 173: Bias-Variance Tradeoff

20.2.4.1 Effect of more training data on the balance

- The bias $E[\hat{f}_\theta(\underline{x}_0)] - f_\theta(\underline{x}_0)$ is independent of the training set size, as it is the expected value of the model's prediction.
- The variance $E[(\hat{f}_\theta(\underline{x}_0) - E[\hat{f}_\theta(\underline{x}_0)])^2]$ decreases with the training set size, as $\hat{f}_\theta(\underline{x}_0)$ moves closer to its expected value.

Therefore, more training data flattens the variance curve and moving the optimal complexity to higher complexity. With more training data we can afford using a more complex model.

This is illustrated in figure 174.

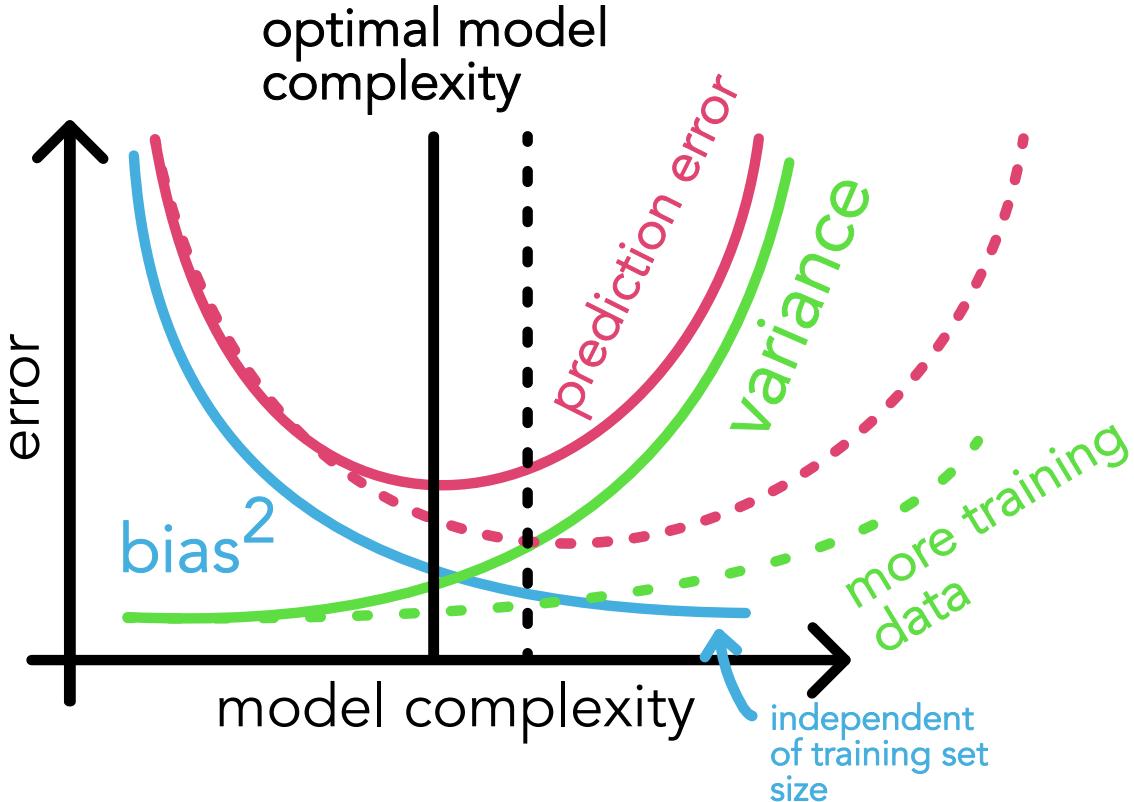


Figure 174: Bias-Variance Tradeoff with more training data

20.3 Finding good hyperparameters | partitioning, cross-validation and analytical model selectors

Aim: Consider our model has hyperparameters, like the support λ in Local Linear Regression. How can we find hyperparameters yielding the model that generalizes best?

20.3.1 Choosing hyperparameters if we are rich in training data - partitioning (train-test-validation-split)

We partition the data randomly into

- training set: used to train the model, e.g. 80% of the data
- validation set: used to choose the hyperparameters, e.g. 10% of the data
- test set: used to evaluate the model, e.g. 10% of the data

and perform the following steps

1. For all (many) combinations of hyperparameters (grid-search) train a model on the training data set
2. Evaluate all these models on the validation set (to get the prediction error) and pick the hyperparameters that worked best
3. Evaluate the performance on the test set (only once)
4. With the best hyperparameters, train on the whole data set (union of train, validation and test) and deploy

Note: As we directly optimize the model on the training set, the training error is not a good measure of generalization. And as we optimize the hyperparameters on the validation set, we need the test set - that a certain combination of hyperparameters performs well on the validation set might just be due to chance.

Common problems

- train set is contaminated by validation or test samples, e.g. by (close) duplicates in the data
- partitioning into train, test and validation is in some sense not random
- cheating by testing on the test set multiple times
- in classification problems: class imbalance in the train, test and validation set - for instance if the test set only contains samples of one class (e.g.) dogs, a *stupid* model only predicting dogs will perform perfectly

Improvement for classification - stratification: Here we try to make the splits such that the proportions of the classes are kept as in the total data set to avoid the problem mentioned above.

20.3.2 Choosing hyperparameters if we are poorer in training data - K-folds Cross-Validation

20.3.2.1 K-Fold for estimating the prediction error

Very generally, if we want to estimate how a model generalizes, so what the expected error on unseen data is, we can

- partition the data randomly into K folds (e.g. $K = 10$)

- train K models $\hat{f}^{-k}, k = 1, \dots, K$ on all data except on the k -th fold each
- evaluate each model on the k -th fold it was not trained on
- obtain the average and spread of the prediction error

Therefore the cross-validation error for a model with given parameter λ is

$$\text{CV}(\lambda) = \frac{1}{K} \sum_{k=1}^K \sum_{i \in S_{-k}} \text{Err}[y_i, \hat{f}^{-k}(\underline{x}_i)] \quad (1066)$$

where S_{-k} is the set of all samples except the k -th fold.

20.3.2.2 K-Fold for choosing hyperparameters

We can calculate $\text{CV}(\lambda)$ for different λ and pick the one yielding the smallest cross-validation error. With this λ , we can then train the model on the whole data set and deploy.

With cross-validation we can also estimate how good our estimate of the prediction error is via the spread of the cross-validation errors.

For local linear regression the cross-validation error is plotted in figure 175.

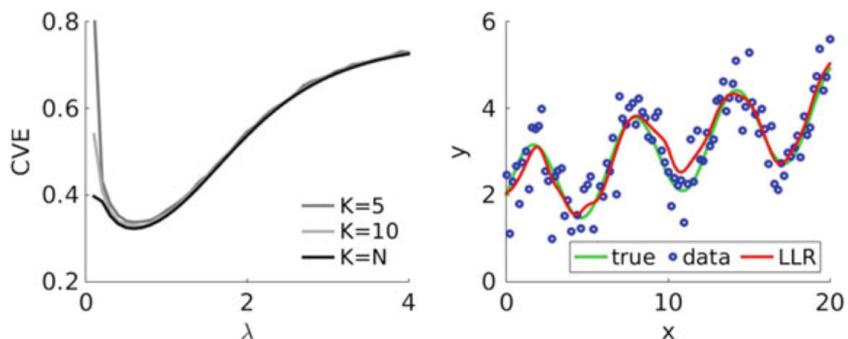


Figure 175: Cross-Validation Error for Local Linear Regression, estimating the prediction error for different λ . Model complexity is lower for larger λ .

Cross-validation is itself subject to bias-variance tradeoff:

- for $K = N$ (leave-one-out) the error (only calculated on one data point) will vary highly between different partitions and the trained models will - as the training sets only differ by two data-points each - be very similar
- for small K we will train on significantly fewer data and the set of hyperparameters found in this case might not be the best for the model trained on all data

K-Fold cross validation is good for small data sets and gives the spread of the estimated prediction error but the accuracy obtained can be too optimistic.

20.3.3 Analytical Model Selectors

Information criteria try to balance goodness of fit and model complexity.

For instance, the Akaike Information Criterion (AIC) is defined as

$$\text{AIC} = -2 \log \mathcal{L}(\hat{\beta}_{\text{MLE}}) + 2p$$

with $\mathcal{L}(\hat{\beta}_{\text{MLE}})$ the likelihood of the model,
 p effective number of parameters

(1067)

so in case of a linear model

$$\text{AIC}(\lambda) = N \log \left(\sum_{i=1}^N (y_i - \hat{y}_i)^2 \right) + 2p(\lambda) + \text{const.}, \quad p = \text{trace}(\underline{\underline{S}}), \quad \hat{\underline{Y}} = \underline{\underline{X}} \hat{\underline{\beta}} = \underline{\underline{S}} \underline{Y}$$

in Ridge Regression $\underline{\underline{S}} = \underline{\underline{X}}^T (\underline{\underline{X}} \underline{\underline{X}}^T + \lambda \underline{\underline{I}})^{-1} \underline{\underline{X}}$

(1068)

The Bayesian Information Criterion (BIC) is defined as

$$\text{BIC} = -2 \log \mathcal{L}(\hat{\beta}_{\text{MLE}}) + p \log N, \quad \text{sample size } N$$
(1069)

An example is given in figure 176.

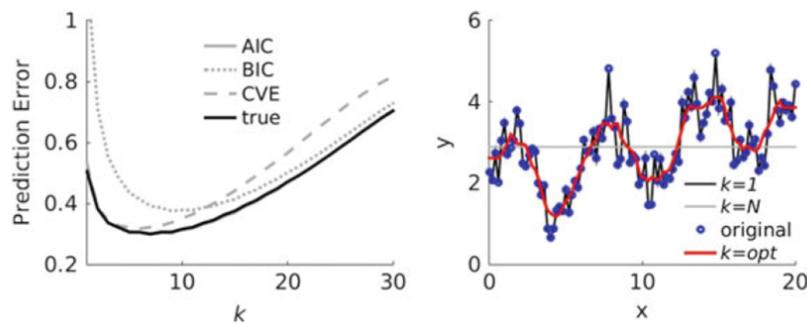


Figure 176: Information Criteria and Cross Validation estimates of the prediction error for KNN Regression. The larger K , the lower the number of effective parameters (lower model complexity). BIC is more conservative, choosing a less complex model (less risky regarding generalizability).

20.4 Breakpoint of Bias-Variance Tradeoff - double descent

Usually, one would say »a model with zero training error is overfit to the training data and will typically generalize poorly«. In double descent for (implicitly regularized models) a second descent in test error over complexity is observed in the overparameterized (data undersampled) regime (depending on the dataset-model pair, more descents are possible).

»By considering larger function classes, which contain more candidate predictors compatible with the data, we are able to find interpolating functions that have smaller norm and are thus “simpler”. Thus increasing function class capacity improves performance of classifiers.« (Belkin et al., 2019)

Double descent is illustrated in figure 177.

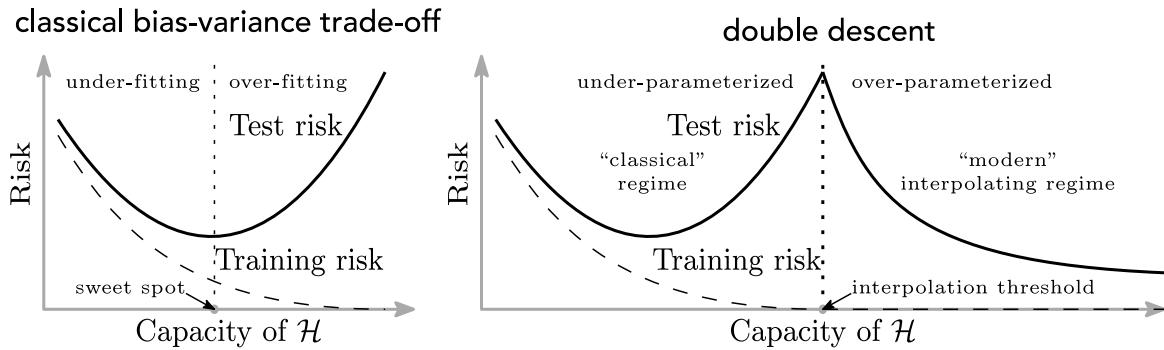


Figure 177: Double Descent in Test Error over Model Complexity

Note: Double descent is not limited to complex neural networks but also observed in ordinary overparameterized linear regression, see figure 178.

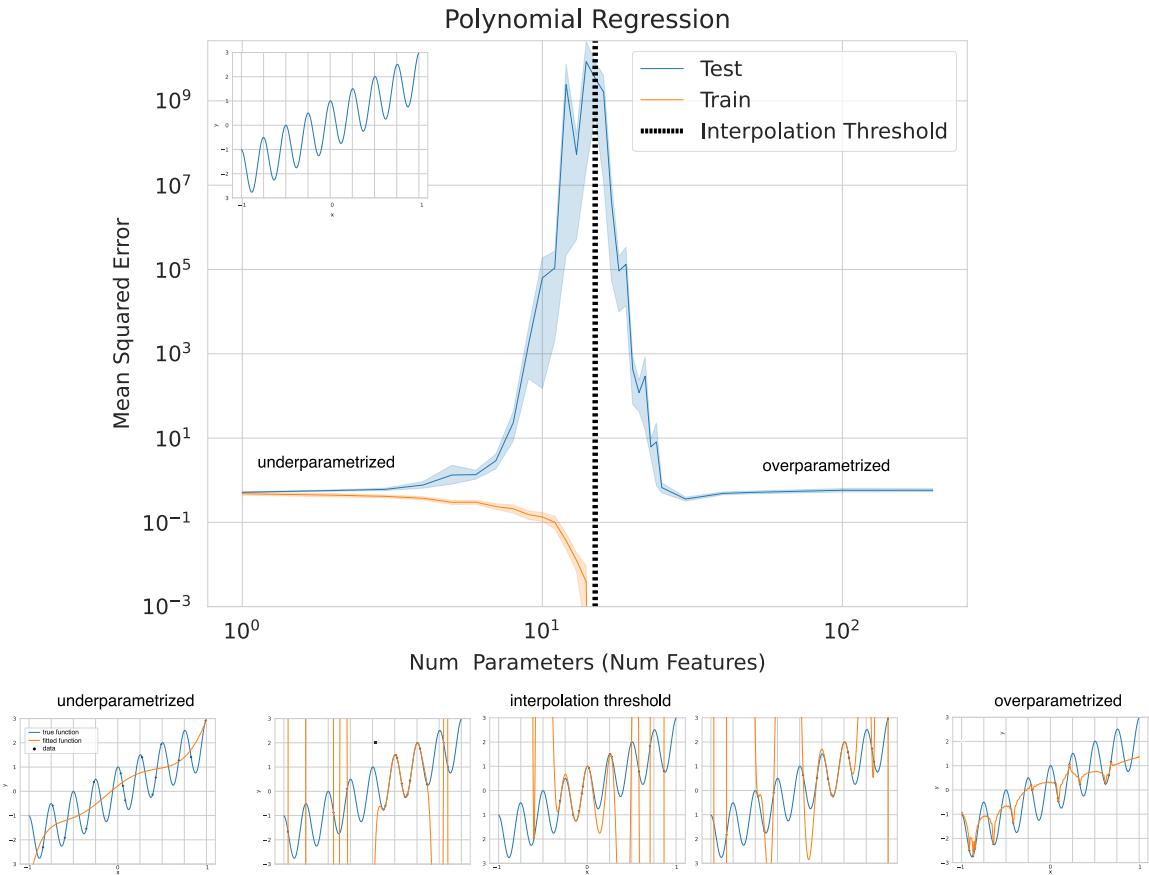


Figure 178: In the overparameterized regime, the model can exactly fit the training data (small bias) but the model is also regularized towards a small-norm solution, making variance small.

20.5 Curse of Dimensionality in Modeling*

Curse of dimensionality is a term used to describe many phenomena.

20.5.1 Curse of Dimensionality in Modeling / Sampling

- **Number of data points perspective:** Consider we find that to correctly model something in one dimension we need N_1 data points. Then in d dimensions we would need N_1^d data points which quickly becomes infeasible with today's high-dimensional training data sets.
- **Resolvable scale perspective:** Consider we have a fixed number of data points N sampling some distribution. Consider our scale is the volume in which we expect to find one data point. For a box with length L , in one dimension, this scale is

$$\Delta x_{1D} = \frac{L}{N}, \quad \Delta x_{2D}^2 = \frac{L^2}{N} \quad \rightarrow \quad x_{2D} = \frac{L}{\sqrt{N}} \quad (1070)$$

so generally, in d dimensions

$$x_{dD} = \frac{L}{N^{1/d}} \quad (1071)$$

So the higher the dimension, the worse our resolution for fixed N . This is illustrated in figure 179.

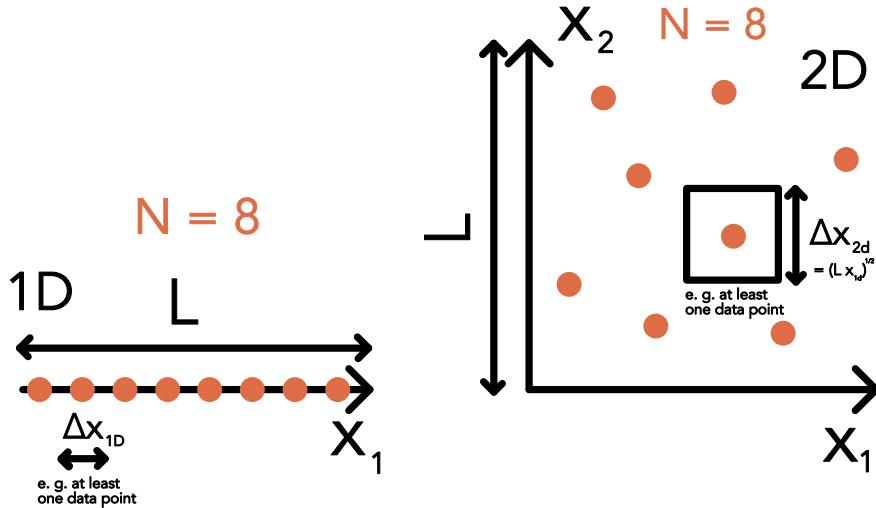


Figure 179: Curse of dimensionality in data density

So we need more data to model things in higher dimensions. So classically, as number of features or dimensions grows, the amount of data we need to generalize accurately grows exponentially

20.5.2 Example for Curse of Dimensionality in number of possible models*

Consider we are given

$$\underline{\underline{X}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad \underline{Y} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad (1072)$$

so data from a boolean function $f : \{0, 1\}^3 \rightarrow \{0, 1\}$. There are 2^3 possible inputs of such a function where a function can give 2 outputs so there are $2^{2^3} = 256$ possible boolean functions of three variables. By fixing 5 of the possible 8 inputs, there are still $2^3 = 8$ possible functions - which gets worse with more dimensions.

For a model with more parameters than data points, we need regularization to find the *best no-bias model*.

20.6 Overview on dealing with complexity

Three approaches for dealing with model complexity are

1. Feature selection: Only use features (components of the measurements) relevant to the outcome, e.g. simply remove features with least variance
2. Regularization: Penalize model complexity / drive unnecessary parameters to zero
3. Dimensionality reduction: Project the data into lower dimensions which still capture the relevant information

20.7 Feature selection - best subset selection

Consider a data set $\mathcal{D} = \{(\underline{x}_i, y_i)\}_{i=1}^N$ with $\underline{x}_i \in \mathbb{R}^p$.

1. Start with an empty set $S = \emptyset$
2. Add a feature $\{x_{ij}\}_{i=1}^N$ to S which reduces the prediction error of the model the most
3. Continue adding features until the prediction error rises again (overfitting)
4. Backward elimination: As feature combinations added later might have made previous ones obsolete, one prunes $S \leftarrow S \setminus \{x_{ij}\}_{i=1}^N$ as long as we can reduce the prediction error by doing so

20.8 Regularization I | General idea and pre-step

Aim: Penalize model complexity to improve generalization.

We penalize model complexity by modifying the loss to

$$L_{\text{reg}} = L + \lambda \cdot \text{penalty}, \quad \text{regularization parameter } \lambda \quad (1073)$$

where the penalty term is usually a L_1 or L_2 norm of the model parameters.

20.8.1 Pre-step to regularization - centering, standardizing

Consider the linear model

$$y_i = \beta_0 + \sum_{j=1}^p \beta_j x_{ij} + \epsilon_i, \quad \underline{\underline{X}} \in \mathbb{R}^{N \times p}, \underline{Y} \in \mathbb{R}^N \quad (1074)$$

Now consider we would like to regularize the model and driving down parameters of irrelevant features to zero.

Problem:

- the penalty term is not scale-invariant: consider two equally important features, one with a large numerical scale one with a small scale, in usual linear regression, the large-scale feature will have a smaller parameter but as of model linearity, the same effect on the prediction. But consider we would drive small parameters to zero - then large scale features would be penalized more. So we scale and center the features in the feature matrix $\underline{\underline{X}}$.
- we do not want to penalize the intercept term $\beta_0 = \bar{y} - \sum_{j=1}^p \bar{x}_j \beta_j$, so \underline{Y} should be centered

20.8.1.1 Centering and standardizing (z-scoring)

$$\begin{aligned}\tilde{x}_i &= \frac{x_i - \bar{x}}{s_x}, & \bar{x} &= \frac{1}{N} \sum_{i=1}^N x_i \\ \tilde{y}_i &= \frac{y_i - \bar{y}}{s_y}, & \bar{y} &= \frac{1}{N} \sum_{i=1}^N y_i\end{aligned}\tag{1075}$$

where s_x collects the estimated standard deviations of the features (element-wise division).

20.9 Bayesian perspective on Regularization

Bayesian approaches guard from overfitting with the prior acting as a bias, **the MAP in the Bayesian perspective is equivalent to a penalized maximum likelihood, penalized with the logarithm of the prior.**

$$\begin{aligned}\hat{\underline{\beta}}_{\text{MLE, reg}} &= \underset{\underline{\beta}}{\operatorname{argmin}} \left(-\log \mathcal{L}(\underline{\beta} | \underline{\underline{X}}) + \lambda P(\underline{\beta}) \right) \\ \hat{\underline{\beta}}_{\text{MAP}} &= \underset{\underline{\beta}}{\operatorname{argmin}} \left(-\log \left(\mathcal{L}(\underline{\beta} | \underline{\underline{X}}) g(\underline{\beta}) \right) \right) \\ &= \underset{\underline{\beta}}{\operatorname{argmin}} \left(-\log \left(\mathcal{L}(\underline{\beta} | \underline{\underline{X}}) \right) - \log g(\underline{\beta}) \right)\end{aligned}\tag{1076}$$

where for $\hat{\underline{\beta}}_{\text{MAP}}$ the evidence $h(\underline{\underline{X}})$ is constant and can be dropped.

Note, however, that in the Bayesian setting, the MAP plays no special role, and if a point estimate is necessary, the posterior mean is usually preferred.

20.10 Regularization II | Regularized regression

Problem: Note that

- for $p > N$, standard underparameterized linear regression breaks down, as $\underline{\underline{X}}^T \underline{\underline{X}}$ is not invertible
- for collinear features, there is also no unique solution to the parameters, consider e.g. bad training data

$$\underline{x}_i = \begin{pmatrix} 1 \\ x \\ -x \end{pmatrix} \rightarrow y = \beta_0 + \beta_1 x - \beta_2 x = \beta_0 + (\beta_1 - \beta_2) x \quad (1077)$$

then there is a unique solution to $\beta_0, \beta_1 - \beta_2$ but not to β_1, β_2 separately. When the collinearity is not perfect, we might be able to invert $\underline{\underline{X}}^T \underline{\underline{X}}$ its likely unstable, e.g. β_1, β_2 might be driven to very large values of opposite sign.

Idea: By regularization constrain the magnitude of the coefficients, bias towards smaller coefficients, so limiting the problem of collinearity.

$$\begin{aligned} \hat{\underline{\beta}}_{\text{regu}} &= \underset{\underline{\beta}}{\operatorname{argmin}} L = \underset{\underline{\beta}}{\operatorname{argmin}} (\text{SSQ}(\underline{\beta}) + \lambda P(\underline{\beta})) \\ \text{SSQ}(\underline{\beta}) &= \|\underline{Y} - \underline{\underline{X}}\underline{\beta}\|^2, \quad \text{some penalty } P(\underline{\beta}) \\ &\quad \text{regularization strength (hyperparameter)} \lambda \end{aligned} \quad (1078)$$

Different regularizations use different penalties.

20.10.1 Ridge Regression

We penalize on the squared norm of $\underline{\beta}$

$$P(\underline{\beta}) = \|\underline{\beta}\|_2^2 = \sum_{j=1}^p \beta_j^2 \quad (1079)$$

So $\hat{\underline{\beta}}_{\text{ridge}}$ follows from

$$\frac{\partial(\text{SSQ} + \lambda P)}{\partial \underline{\beta}} = 0 \rightarrow \hat{\underline{\beta}}_{\text{ridge}} = (\underline{\underline{X}}^T \underline{\underline{X}} + \lambda \underline{\underline{1}})^{-1} \underline{\underline{X}}^T \underline{Y} = " \frac{\underline{\underline{X}}^T \underline{Y}}{\underline{\underline{X}}^T \underline{\underline{X}} + \lambda \underline{\underline{1}}} " \quad (1080)$$

Intuition: The regularization inflates the scatter matrix $\underline{\underline{X}}^T \underline{\underline{X}}$ (along all dimensions) by λ , keeping the matrix invertible.

20.10.1.1 Influence of λ and Bias-Variance Tradeoff

With stronger regularization strength λ , the parameters are driven towards zero, as illustrated in figure ???. The smaller the coefficients, the less sensitive will the model output be to changes in the input, so the variance is reduced but the bias is increased.

For $\lambda \rightarrow 0$ we have ordinary least squares linear regression. A sweet spot between bias and variance, can be found by cross-validation.

This is illustrated in figure ??.

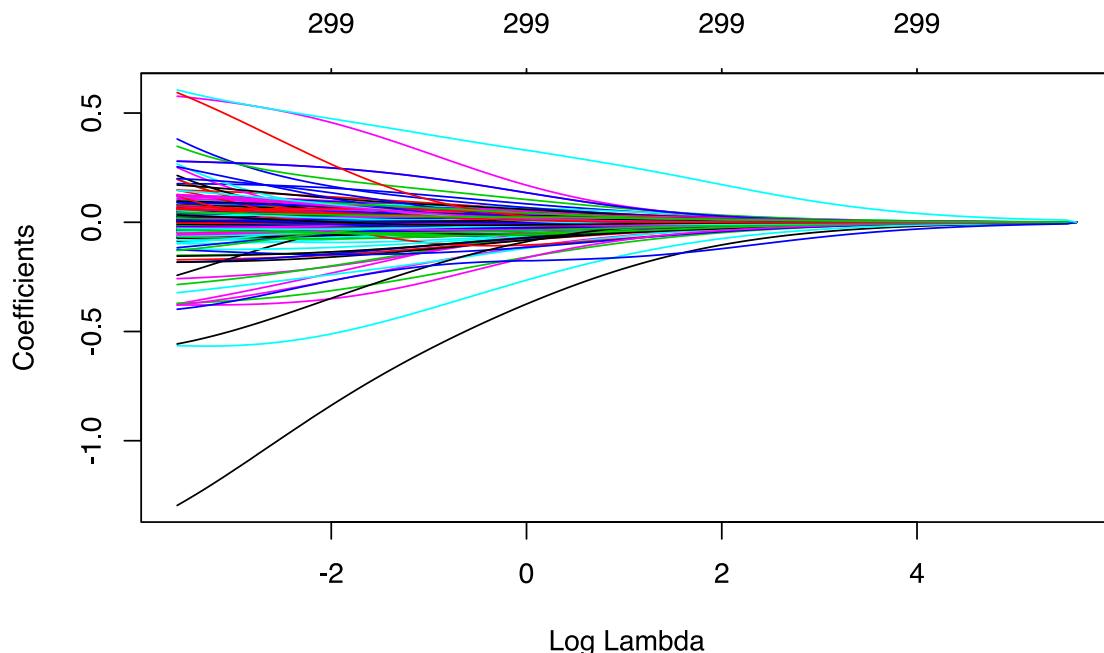


Figure 180: Ridge Regression - influence of λ on the coefficients

20.10.1.2 Bayesian perspective on Ridge regression

Ridge regression is equivalent to using the prior

$$\underline{\beta}_{\text{prior}} \sim \mathcal{N}(0, \frac{\sigma^2}{\lambda} \underline{\underline{I}}) \quad (1081)$$

as the corresponding negative log-likelihood of the prior is

$$-\log p(\underline{\beta}) = \frac{1}{2} \underline{\beta}^T \frac{\lambda}{\sigma^2} I \underline{\beta} = \frac{\lambda}{2\sigma^2} \|\underline{\beta}\|_2^2 \quad (1082)$$

(where the $\frac{1}{\sigma^2}$ was used so it can be multiplied out from the minimization together with the $\frac{1}{\sigma^2}$ in the term negative log-likelihood term).

20.10.1.3 Discussion of the ridge regression

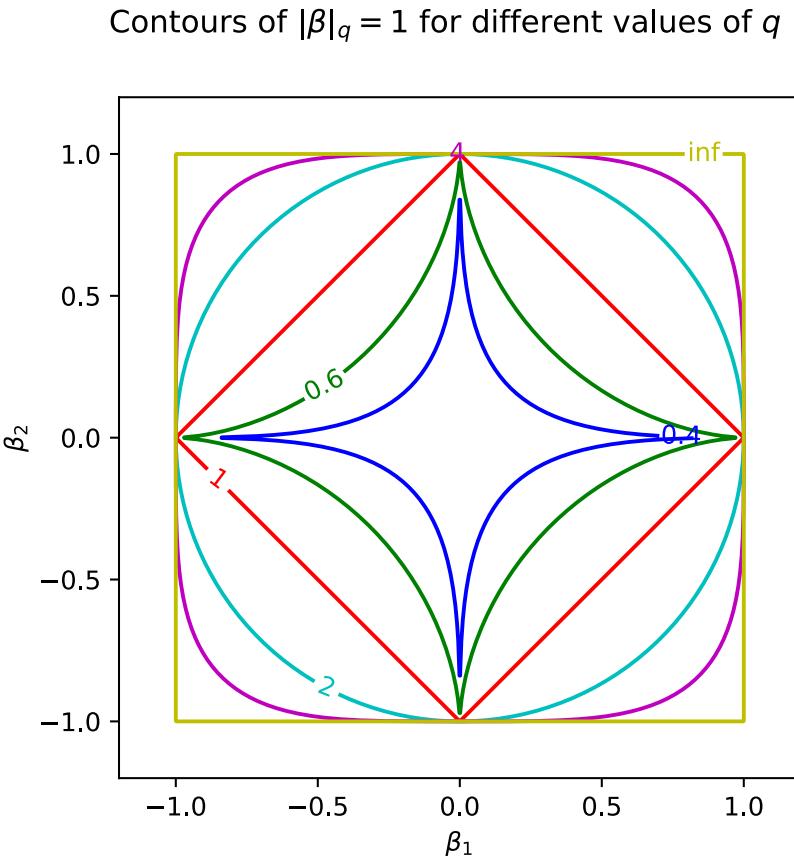
- bias is introduced
- **stabilization:** coefficients for correlated features are pushed towards each other rather than one being wildly positive and the other wildly negative
- **non important variables are pushed closer to zero**
- all variables are retained - no feature selection

20.10.2 On choosing the penalty

In ridge regression, we used the L_2 norm in $P(\underline{\beta}) = \|\underline{\beta}\|_2^2$. More generally, we can use the L_q norm

$$P(\underline{\beta}) = \|\underline{\beta}\|_q^q = \sum_{j=1}^p |\beta_j|^q \quad (1083)$$

Different norms are illustrated in figure 181.

Figure 181: Isocontours of $\|\underline{\beta}\|_q = 1$

20.10.3 Using the L_0 norm (under $0^0 = 0$)

The L_0 norm is the number of non-zero elements in a vector, so the number of explanatory variables used in the model - its not really a norm. Depending on the regularization strength, the L_0 norm drives us to using less and less variables, as illustrated in figure 182.

Problem: The L_0 norm is not convex (also not really a norm^a) (and also not differentiable), so the optimization problem is not, so its hard to optimize.

^aBecause it is not homogeneous, scaling \underline{x} does not scale the norm.

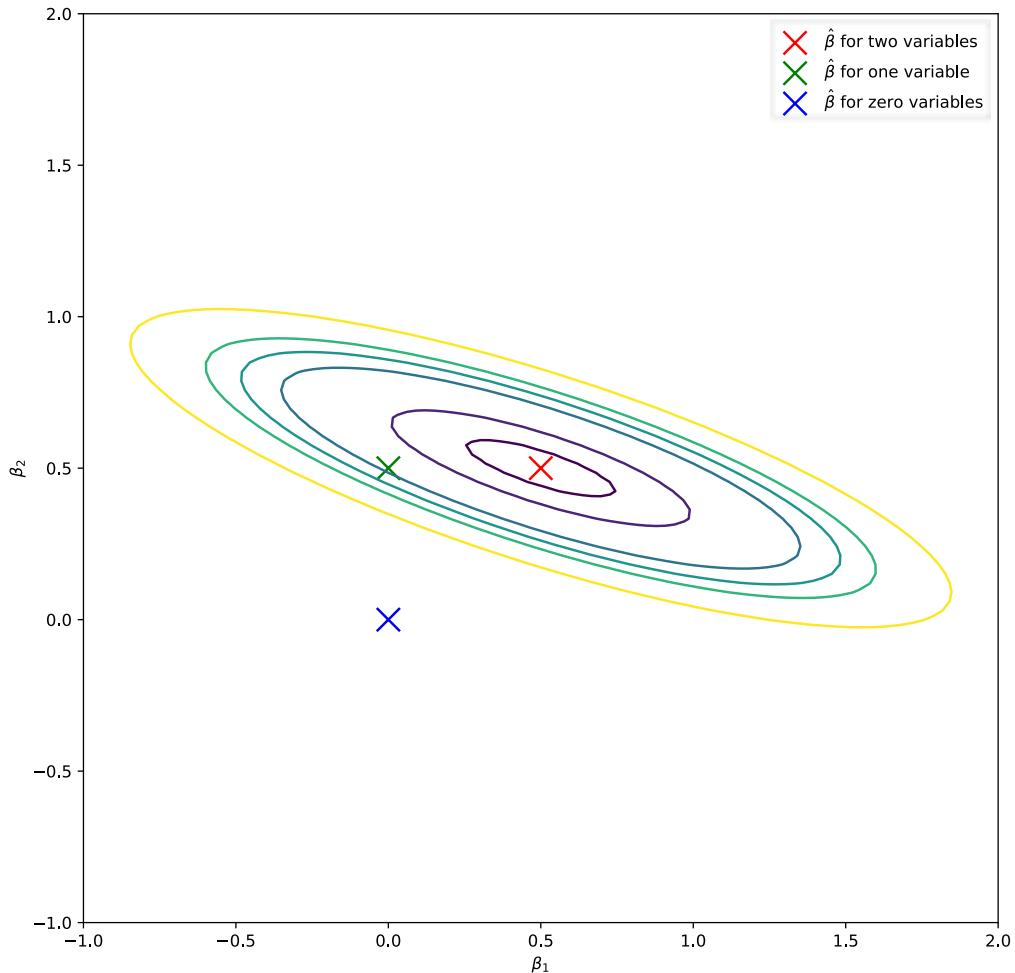


Figure 182: As λ increases, we are first driven to the green, then to the blue estimate $\hat{\beta}$ (drawn are the sum of squared residual contours).

20.10.4 Lasso Regression

A convex norm still selecting features is the L_1 norm, the sum of the absolute coefficient values.

20.10.4.1 Penalty in the Lasso - L_1 norm

$$P(\underline{\beta}) = \|\underline{\beta}\|_1 = \sum_{j=1}^p |\beta_j| = \text{sign}(\underline{\beta})^T \underline{\beta}, \quad [\text{sign}(\underline{\beta})]_i = \begin{cases} -1, & \beta_i < 0 \\ 0, & \beta_i = 0 \\ 1, & \beta_i > 0 \end{cases} \quad (1084)$$

20.10.4.2 Why does Lasso drive coefficients to exactly zero? | Soft-thresholding of the coefficients in Lasso regression

Lasso regression sets coefficients to exactly zero. This can be written as a soft-thresholding one the ordinary least squares solution, setting sufficiently small coefficients to zero.

For $P(\underline{\beta}) = \|\underline{\beta}\|_1$, the loss becomes

$$L(\underline{\beta}) = \text{SSQ}(\underline{\beta}) + \lambda \|\underline{\beta}\|_1 = \|\underline{Y} - \underline{\underline{X}}\underline{\beta}\|^2 + \lambda \|\underline{\beta}\|_1 \quad (1085)$$

Note: $\|\underline{\beta}\|_1$ is not differentiable at zero, so in the following assume $\beta_j \neq 0$. By consistency argumentation we can then say, when we should have $\beta_j = 0$.

$$\partial_{\underline{\beta}} L = -2\underline{\underline{X}}^T \underline{Y} + 2\underline{\underline{X}}^T \underline{\underline{X}} \underline{\beta} + \lambda \underline{\underline{D}} \underline{1} = 0, \quad D_{jj} = \begin{cases} 1, & \beta_j > 0 \\ -1, & \beta_j < 0 \end{cases} \quad (1086)$$

which also places the constraint

$$D_j j = \text{sign}(\beta_j) \quad (1087)$$

Derivation of the thresholding for scalar $x_i \in \mathbb{R}$, so $\beta \in \mathbb{R}$ ($\beta_0 = 0$ as we centered the data) In 1D we have

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \underline{\beta}} &= -2 \sum_{i=1}^N x_i y_i + 2\beta \sum_{i=1}^N x_i^2 + \lambda d = 0, \quad d = \begin{cases} -1, & \beta < 0 \\ 1, & \beta > 0 \end{cases} \\ \rightarrow \hat{\beta}_{\text{lasso}} &= \frac{\sum_{i=1}^N x_i y_i}{\sum_{i=1}^N x_i^2} - \frac{\lambda d}{2 \sum_{i=1}^N x_i^2} = \hat{\beta}_{MLE} - \frac{\lambda d}{2 \sum_{i=1}^N x_i^2} \end{aligned} \quad (1088)$$

Consistency of $\hat{\beta}_{\text{lasso}}$ with the sign of d requires

$$d = 1 \rightarrow \hat{\beta}_{MLE} > \frac{\lambda}{2 \sum_{i=1}^N x_i^2}, \quad d = -1 \rightarrow \hat{\beta}_{MLE} < -\frac{\lambda}{2 \sum_{i=1}^N x_i^2} \quad (1089)$$

So if $\hat{\beta}_{\text{lasso}}$ is to be non-zero, $\hat{\beta}_{MLE}$ must be larger (in magnitude) than a certain value. So otherwise, $\hat{\beta}_{\text{lasso}}$ is set to zero.

Note: The equation derived above for $d = 0$ so $\hat{\beta}_{\text{lasso}} = 0$ says $\hat{\beta}_{\text{MLE}} = 0$, a contradiction so it seems. But note that the equation is derived for $\beta \neq 0$, as of differentiability, so inserting $d = 0$ is not even valid.

This is illustrated in figure 183.

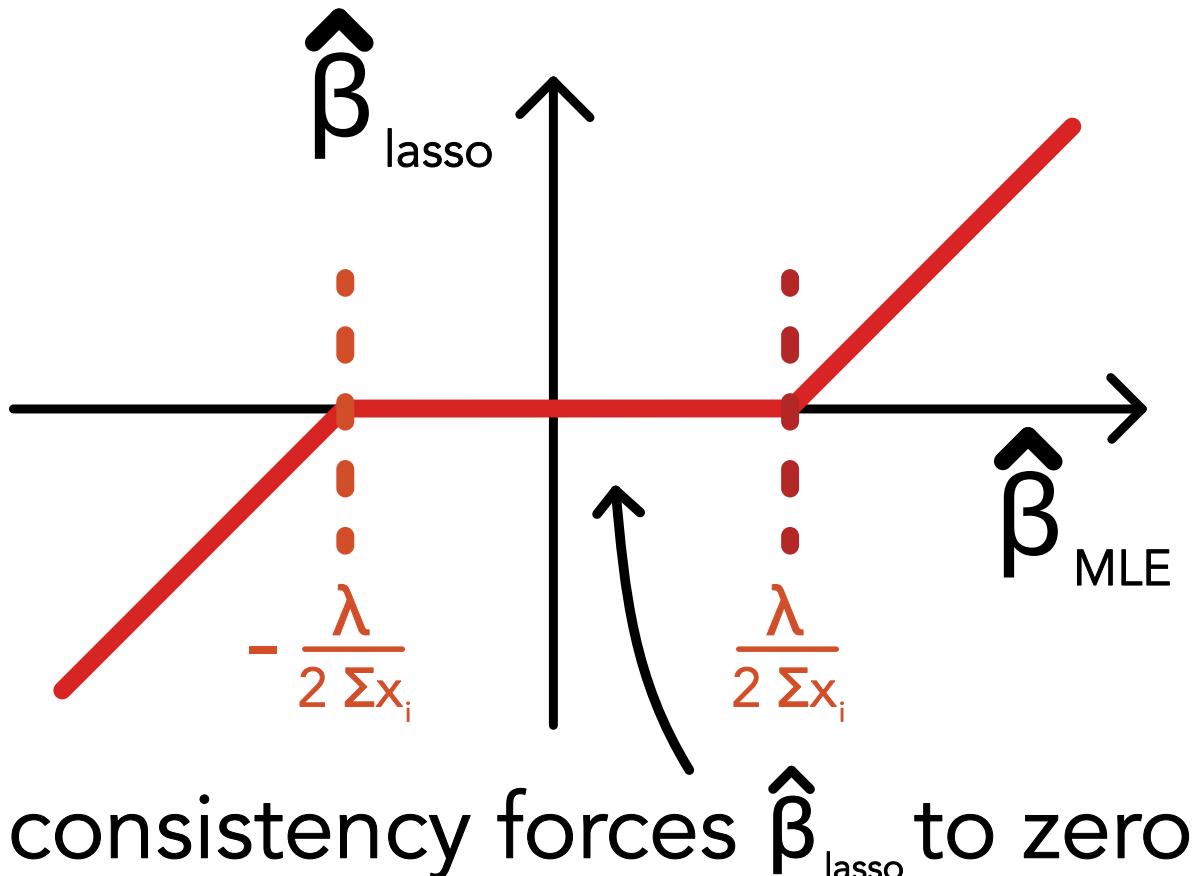


Figure 183: Soft thresholding in Lasso regression

Loss contours of lasso regressio and ridge regression are illustrated in figure 184.

20.10.4.3 Influence of λ on the coefficients in Lasso regression

The feature selection performed by Lasso is illustrated in figure 185.

For strongly correlated features, one is often pushed fully to zero.

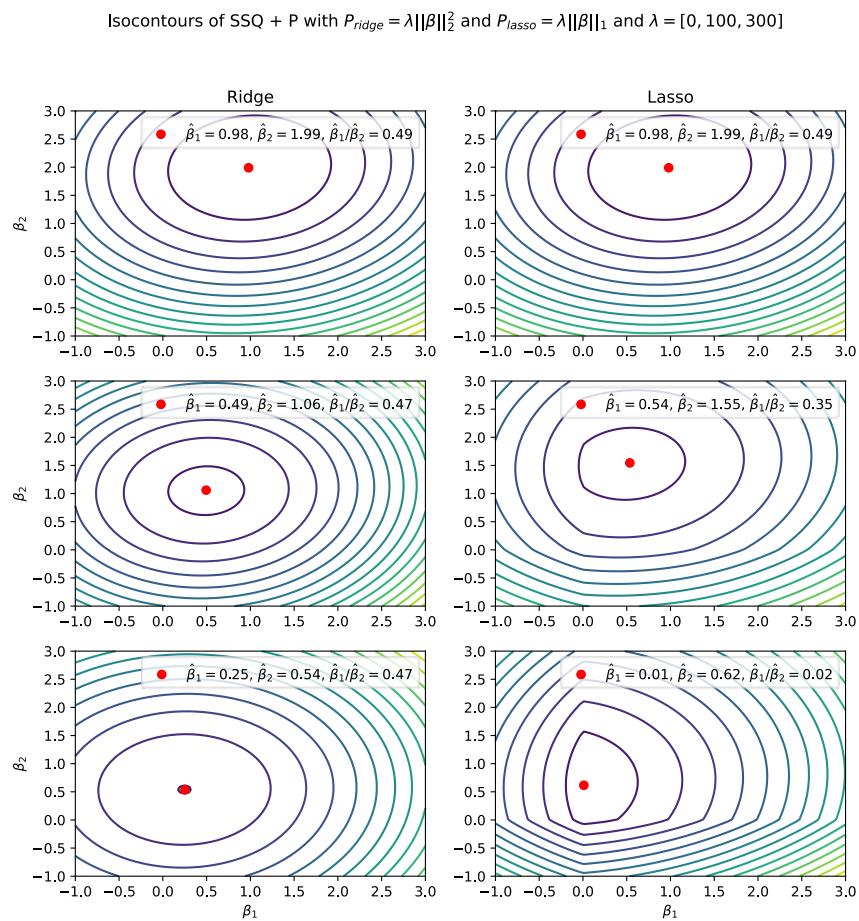


Figure 184: Loss contours of lasso and ridge regression

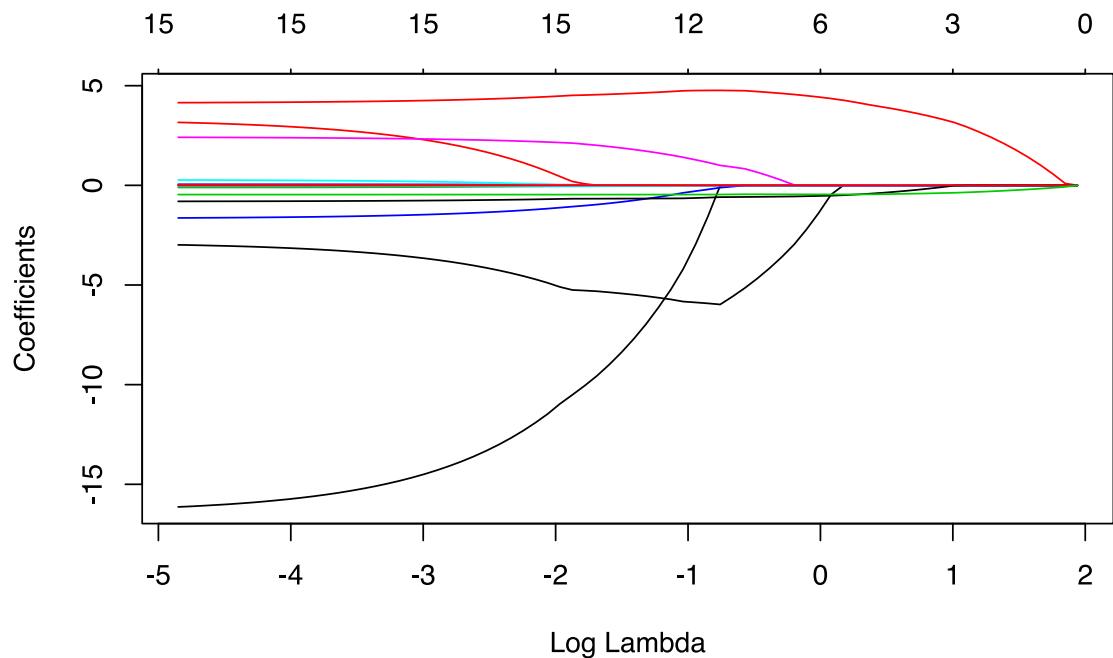


Figure 185: Influence of λ on the coefficients in Lasso regression

20.10.5 Why does Lasso regression lead to feature selection (*induces sparsity*) and Ridge regression does not?

We have already gained some intuition, why by consistency in Lasso regression, coefficients are driven to zero. Another angle comes from the constraint perspective.

20.10.5.1 Constraint perspective and regularized regression

The Ridge minimization

$$\hat{\underline{\beta}}_{\text{ridge}, \lambda} = \operatorname{argmin}_{\underline{\beta}} (\text{SSQ}(\underline{\beta}) + \lambda \|\underline{\beta}\|_2^2) \quad (1090)$$

can be written as as the constraint optimization problem

$$\hat{\underline{\beta}}_{\text{ridge}, \gamma} = \operatorname{argmin}_{\underline{\beta}} \text{SSQ}(\underline{\beta}) \quad \text{s.t.} \quad \|\underline{\beta}\|_2^2 \leq \gamma \quad (1091)$$

where

$$\forall \lambda > 0 \quad \exists \gamma : \hat{\underline{\beta}}_{\text{ridge}, \lambda} = \hat{\underline{\beta}}_{\text{ridge}, \gamma} \quad (1092)$$

The Lasso minimization

$$\hat{\underline{\beta}}_{\text{lasso}, \lambda} = \operatorname{argmin}_{\underline{\beta}} (\text{SSQ}(\underline{\beta}) + \lambda \|\underline{\beta}\|_1) \quad (1093)$$

can analogously be written as as the constraint optimization problem

$$\hat{\underline{\beta}}_{\text{lasso}, \gamma} = \operatorname{argmin}_{\underline{\beta}} \text{SSQ}(\underline{\beta}) \quad \text{s.t.} \quad \|\underline{\beta}\|_1 \leq \gamma \quad (1094)$$

where

$$\forall \lambda > 0 \quad \exists \gamma : \hat{\underline{\beta}}_{\text{lasso}, \lambda} = \hat{\underline{\beta}}_{\text{lasso}, \gamma} \quad (1095)$$

In the constraint perspective, $\hat{\underline{\beta}}_{\gamma}$ is the value of $\underline{\beta}$ which minimizes the sum of squared residuals, and lies within the contour $\|\underline{\beta}\|_q^q = \gamma$, so $\|\underline{\beta}\|_2^2 \leq \gamma$ for Ridge and $\|\underline{\beta}\|_1 \geq \gamma$ for Lasso.

20.10.5.2 Intuition for the difference in feature selection

The constraint solution found is the point of intersection of the first isocontour of the sum of squared residuals And the constraining contour. For the Lasso regression, the constraining contour is a diamond, so the intersection is more likely to be on the axis, so at least one

coefficient is set to zero. This is illustrated in figure 186.

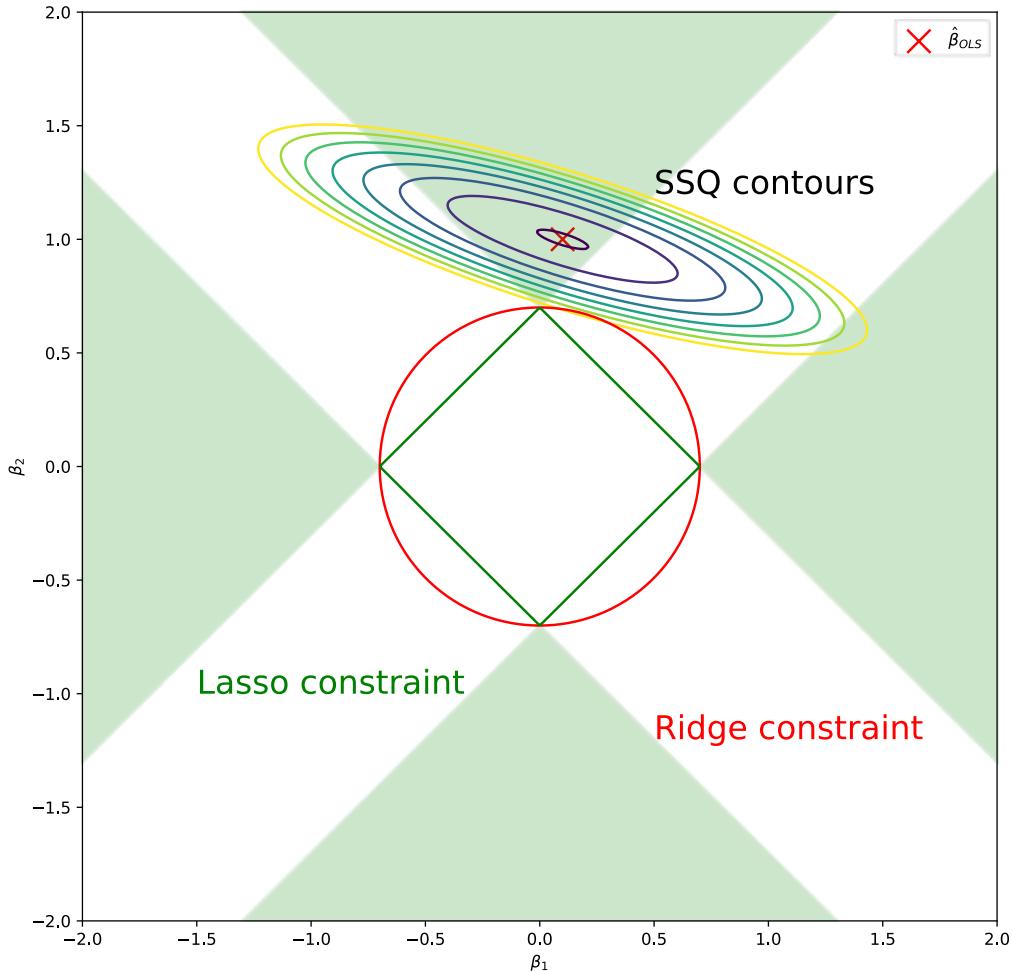


Figure 186: Sum of squares contours and constraining contours for Ridge and Lasso regression. If the OLS estimate falls within one of the green-shaded areas, one parameter is set to zero. This is not the case for Ridge regression.

20.11 Preview: Fighting overfitting and improving performance in Neural Networks: Dropout

By randomly leaving out connections / neurons in a neural network a more distributed representation and robust solutions can be obtained.

21 Classification

As in regression, we are in the land of supervised learning. However, here our labeling is a category, $\mathcal{D} = \{\underline{x}_i, c_i\}$, where $c_i \in \{1, 2, \dots, C\}$, and C is the number of classes.

Aim: Given a new \underline{x} , we want to predict its class.

Regions where certain classes are predicted are separated by decision boundaries. An example classification is shown in figure 187.

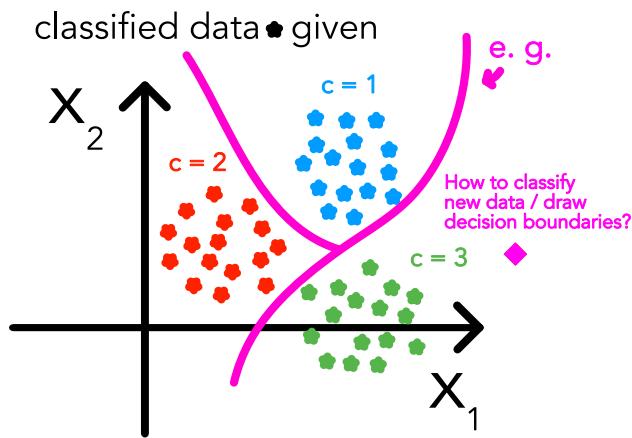


Figure 187: Example classification.

21.1 Overview on approaches to classification

In the Bayesian perspective

- we model the likelihood of data given the class and from Bayes theorem we can make predictions on class label probabilities → **Bayesian classifiers**, e.g. Naive Bayes, etc.

where

- modeling the likelihoods of data given class with a simple Gaussian distribution gets us to **discriminant analysis** (*data modeling culture*)

More belonging to the *algorithmic culture*, we can

- assign a label to a new point based on a majority vote of its k nearest neighbors → **k-NN**
- partition feature space by recursive binary splits in a tree-structure, devising a partitioning where each region is assigned a class → **decision trees**

- use multiple of such trees e.g. trained on bootstrap samples of the data and e.g. majority vote for a decision → **random forests**

In the special case of two classes, we can

- find the linear border between the classes (if possible) minimizing the closest distance to any point of the classes → **support vector machines**
- use a GLM for a Bernoulli distribution, so directly model the likelihood of class labels given a probability model which mean is modeled depending on the features → **logistic regression**

Multiple classes can e.g. be modeled by K 1-vs-all classifications for K classes, where each classifier is trained to distinguish one class from all other $K - 1$ classes.

Logistic regression can also easily be extended to multiple classes by using a softmax function, to model the class probabilities, which are the parameters of a multinomial distribution, describing the likelihood of the observed class labels given the data.

21.2 Discriminant analysis

Discriminant analysis is a Bayesian classifier where the class of a new data point is given by

$$k^* = \operatorname{argmax}_k p(c = k | \underline{x}) \quad (1096)$$

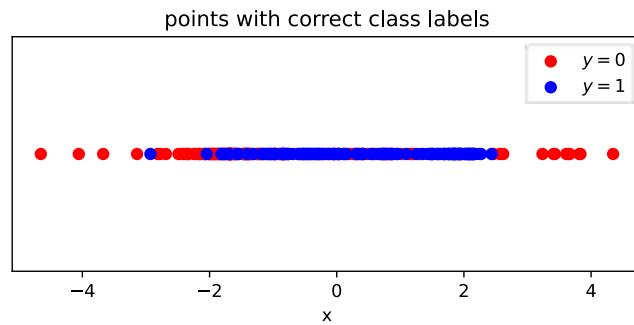
In discriminant analysis, the likelihood of the data given a class is modeled by a Gaussian whose mean and covariance are estimated from the training data which is of that class.

$$\begin{aligned} p(\underline{x}_i | c_i = k) &= \mathcal{N}(\underline{\mu}_k, \underline{\Sigma}_k) \\ &= (2\pi)^{-\frac{p}{2}} (\det \underline{\Sigma}_k)^{-\frac{1}{2}} \exp \left(-\frac{1}{2} (\underline{x}_i - \underline{\mu}_k)^T \underline{\Sigma}_k^{-1} (\underline{x}_i - \underline{\mu}_k) \right) \end{aligned} \quad (1097)$$

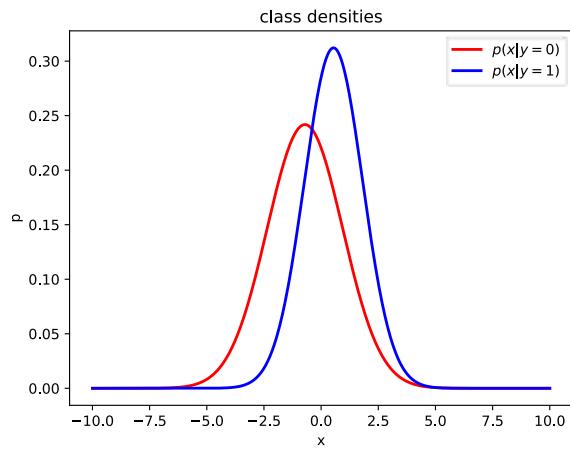
Applying Bayes' theorem yields

$$p(k | \underline{x}_i) = \frac{p(\underline{x}_i | k) p(k)}{\sum_{l=1}^K p(\underline{x}_i | l) p(l)}, \quad p(l) = \frac{\# \text{ data points of class } l}{\# \text{ all data points in training set}} \quad (1098)$$

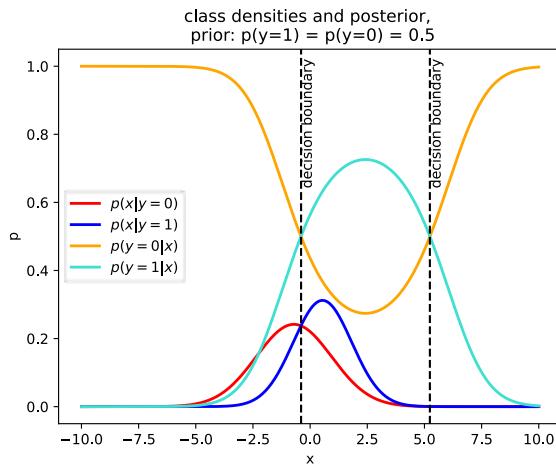
In figure 188, we see an example of discriminant analysis in 1D.



(a) Given data. For instance the weights of exemplars of African Forest Elephants (red) and Asian Elephants (blue), for which a Gaussian distribution seems reasonable.



(b) Fits of the class probability densities. So really just a Gaussian density fit to the African and Asian elephant measurements respectively.



(c) Posteriors, as the red class has higher spread, outer points are assigned to it. Given the weight, what kind of elephant do we assume? As the forest elephant comes in a wider range of weights, at the extremes we assume forest elephant.

Figure 188: Example of discriminant analysis in 1D.

21.2.1 Linear discriminant analysis (LDA) and Mahalanobis distance

Assumption of LDA: We assume all classes to share a common covariance matrix $\underline{\Sigma}_k = \underline{\Sigma}$ (also leading to a more robust estimation of $\underline{\Sigma}$).

Discriminant function: For assigning a class, we only need to know which posterior is larger, not the absolute value, so we can equivalently consider the discriminant function (as always the evidence term $p(\underline{x})$ is the same for all classes, and we apply the strictly monotonic log function):

$$\begin{aligned}\delta_k(\underline{x}_i) &:= \ln \left(\exp \left(-\frac{1}{2} (\underline{x}_i - \underline{\mu}_k)^T \underline{\Sigma}^{-1} (\underline{x}_i - \underline{\mu}_k) \right) p(k) \right) \\ &= -\frac{1}{2} (\underline{x}_i - \underline{\mu}_k)^T \underline{\Sigma}^{-1} (\underline{x}_i - \underline{\mu}_k) + \ln(p(k)) \\ &= -D_{\text{mah}}^2 + \ln(p(k))\end{aligned}\tag{1099}$$

with the decision criterion

$$\hat{k} = \operatorname{argmax}_k p(c = k | \underline{x}_i) = \operatorname{argmax}_k \delta_k(\underline{x}_i)\tag{1100}$$

which is the [Bayes optimal classifier](#) if our assumption $\sim \mathcal{N}(\underline{\mu}_k, \underline{\Sigma})$ is valid.

Mahalanobis distance: D_{mah} measures the distance of a measurement \underline{x}_i to the class mean $\underline{\mu}_k$ normalized by the covariance matrix $\underline{\Sigma}$. The Mahalanobis distance to a class with high spread might be smaller than to a class with low spread, even if the Euclidean distance to the mean of the class with high spread is larger, see figure 189. This is not the case in LDA though, as $\underline{\Sigma}_k = \underline{\Sigma}$ is assumed.

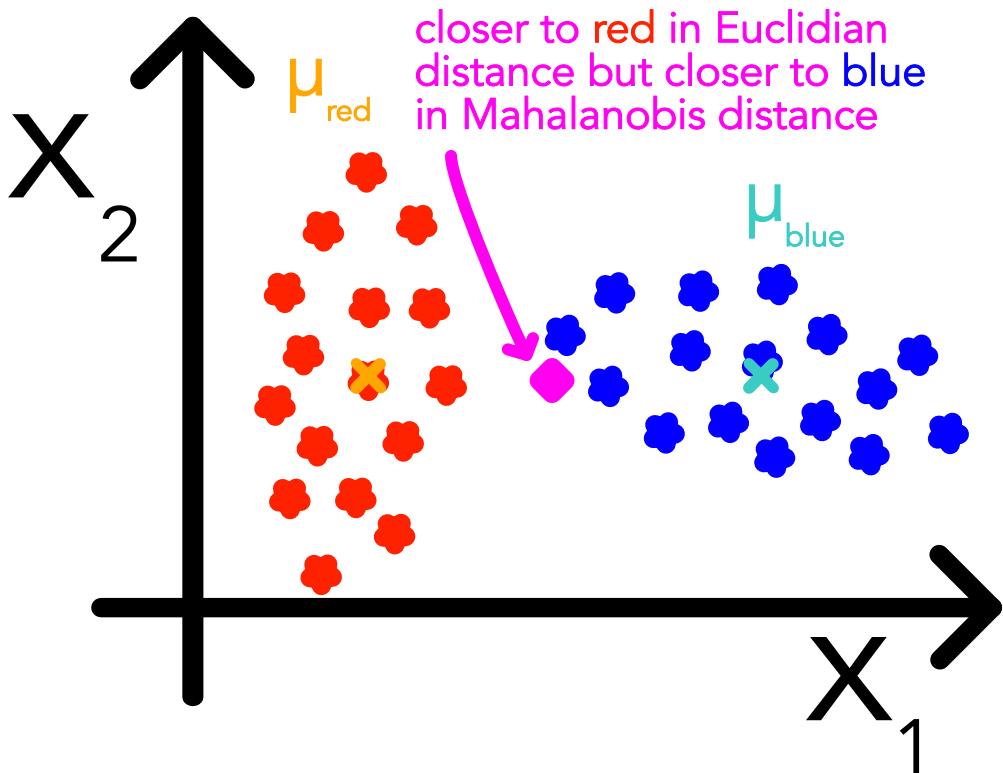


Figure 189: Mahalanobis distance.

Sidenote on multivariate hypothesis testing: By the t-test we can test if given data supports the hypothesis that a mean is of a certain value or means of two samples (or classes) are equal **in the univariate case**. The multivariate analogue of the one-sample t-test is the **Hotelling's T-squared test** and the test statistic $t^2 = D_{\text{mah}}^2 \sim T_{p,N-1}^2 = \frac{p(N-1)}{N-p} F_{p,N-p}$, where p is the number of dimensions, N the number of data points, and $F_{p,N-p}$ the F-distribution. For $\underline{\Sigma}$ the sample covariance

$$\hat{\Sigma} = \frac{1}{N-1} \sum_{i=1}^N (\underline{x}_i - \underline{\mu})(\underline{x}_i - \underline{\mu})^T \quad (1101)$$

is used. An analogue two-sample test, to compare if two classes have the same mean in the multivariate case, is done with a pooled covariance matrix.

21.2.1.1 The LDA decision surface is linear

The decision surface between class k and l is given by

$$\delta_k(\underline{x}) = \delta_l(\underline{x}) \quad (1102)$$

so given by

$$\log \frac{p(k)}{p(l)} - \frac{1}{2} \left(\underline{x} - \underline{\mu}_k \right)^T \underline{\Sigma}_k^{-1} \left(\underline{x} - \underline{\mu}_k \right) + \frac{1}{2} \left(\underline{x} - \underline{\mu}_l \right)^T \underline{\Sigma}_l^{-1} \left(\underline{x} - \underline{\mu}_l \right) = 0 \quad (1103)$$

where for LDA, so $\underline{\Sigma}_k = \underline{\Sigma}_l = \underline{\Sigma}$, the quadratic terms cancel, and we are left with a linear decision boundary.

Quadratic discriminant analysis (QDA): Not assuming the same covariance matrices, we get quadratic decision boundaries in \underline{x} , so ellipses, hyperboloids, parabolas, parallel lines. An example for QDA is shown in figure 190.

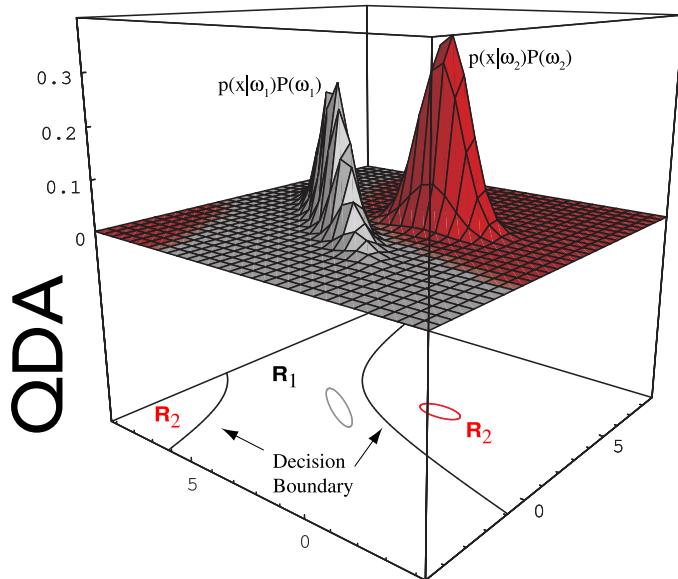


Figure 190: Example of QDA. Class likelihoods in red and grey, decision boundaries drawn below.

21.2.2 Estimation of the prior, mean and covariance in discriminant analysis

The prior is estimated by class proportions in the training data as

$$\hat{p}(c_i = k) = \frac{\# \text{ data points of class } k}{\# \text{ all data points in training set}} = \frac{N_k}{N}, \quad N = \sum_{l=1}^K N_l \quad (1104)$$

the class means intuitively by

$$\hat{\underline{\mu}}_k = \frac{1}{N_k} \sum_{\underline{x}_i \text{ for } c_i=k} \underline{x}_i, \quad \underline{x}_i \in \mathbb{R}^p \quad (1105)$$

and the class covariances by

$$\hat{\Sigma}_k = \frac{1}{N_k - 1} \left(\underline{\underline{X}}_{(k)} - \frac{1}{N_k \times 1} \hat{\mu}^T \right)^T \left(\underline{\underline{X}}_{(k)} - \frac{1}{N_k \times 1} \hat{\mu}^T \right) \quad (1106)$$

where in LDA, we use the pooled covariance matrix

$$\hat{\Sigma} = \frac{1}{N - K} \sum_{k=1}^K (N_k - 1) \hat{\Sigma}_k \quad (1107)$$

21.2.2.1 Advantages of Discriminant Analysis

While Gaussian class likelihoods are a strong assumption

- Discriminant analysis is very robust, no hyperparameters to tune
- and Gaussian class probabilities are common by the central limit theorem

21.3 Linear Classifiers

A linear decision boundary (or hyperplane) can generally be given by its normal vector \underline{w} and an offset b as

$$g(\underline{x}) = \underline{w}^T \underline{x} - b \quad (1108)$$

(we will sometimes use $\underline{\beta} = \underline{w}$ and $\beta_0 = -b$). The **decision** on what class to assign to a point \underline{x} is given by the sign of $g(\underline{x})$. For instance

- in Support Vector Machines, we have class labels $y_i \in \{-1, 1\}$, so $\hat{c}_i = \text{sign}(g(\underline{x}_i))$, a sign-classifier is illustrated in figure 191
- in logistic regression, we model the probability of class 1, $p(c = 1|\underline{x}) = \sigma(g(\underline{x}))$, so $p(c = 0|\underline{x}) = 1 - \sigma(g(\underline{x}))$

The decision boundary is given by

$$\underline{x} \text{ on boundary} : g(\underline{x}) = 0 \quad (1109)$$

In each case, our aim is to find the in some sense best \underline{w} and b , specifying the separating hyperplane and decision function.

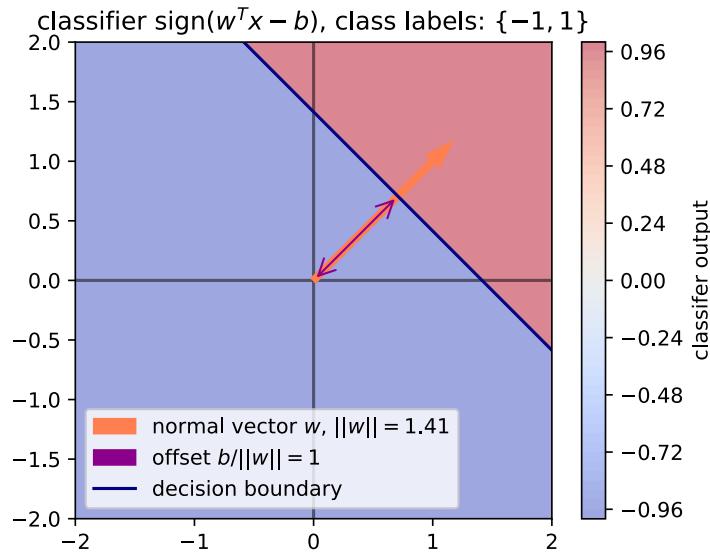


Figure 191: Sign classifier.

21.4 Support Vector Machines (SVM)

An SVM is a non-probabilistic binary classifier with linear decision boundary. Given a training set $\mathcal{D} = \{\underline{x}_i, c_i\}$, $c_i \in \{-1, 1\}$, we want to assign a class to a new point \underline{x} .

21.4.1 Basic Idea of SVMs - maximum margin classifier

We are using a liner classifier

$$g(\underline{x}) = \underline{\beta}^T \underline{x} + \beta_0 \quad (1110)$$

with sign decision function

$$\hat{c} = \text{sign}(g(\underline{x})) \quad (1111)$$

What seperating hyperplane is best?

Consider the example of two linearly separable classes in figure 192. In the figure H_1 does not event separate the classes - H_2 and H_3 both do is perfectly. So why is H_3 the better choice?

Choose the hyperplane (so $\underline{\beta}$ and β_0), so that the distance to the closest point of any class is maximized, minimizing the risk of misclassification.

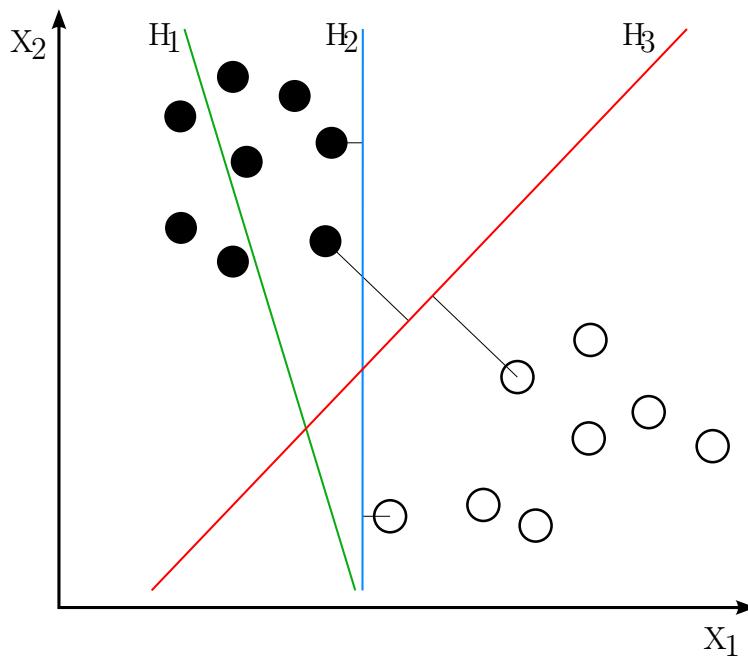


Figure 192: Example of two linearly separable classes.

21.4.2 The margins - intuition for linearly separable classes

We can phrase finding the decision boundary maximizing the minimum distance to any point, as finding two parallel hyperplanes that separate the classes and have the maximum distance between them, **the margins**, with the decision boundary in the middle.

As in the linear classifier, we are free to choose the scale of β and β_0 , we set the margins to be defined by

$$\underline{x} \text{ on margin : } g(\underline{x}) = \pm 1 \quad (1112)$$

so that the distance between the two margins (called margin) is $2/\|\beta\|$.

The best decision boundary is the maximum-margin hyperplane.

Vectors on the margin are called **support vectors**. Regardless of the number of dimensions or size of data set, the number of support vectors could be as little as 2.

Margins and decision boundary are illustrated in figure 193.

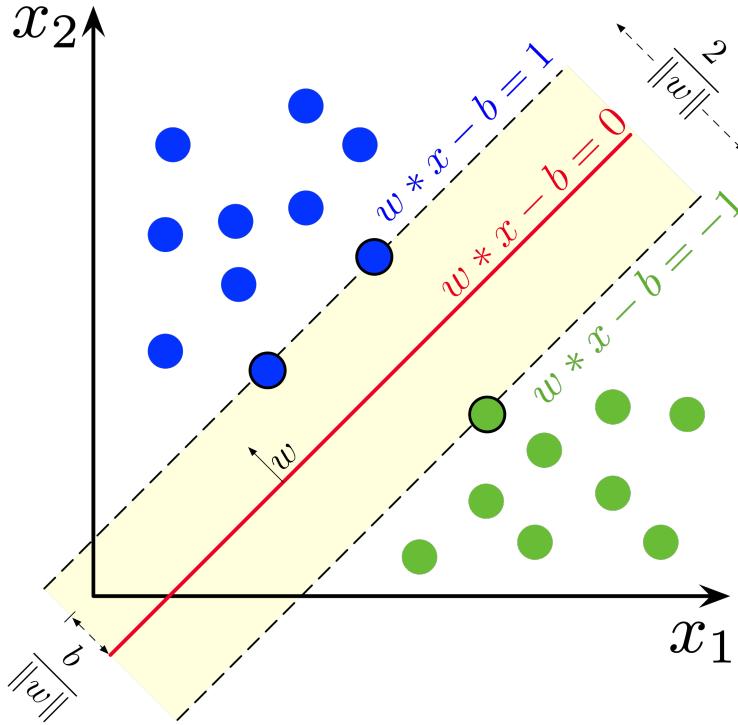


Figure 193: Margins and decision boundary.

21.4.3 Finding $\underline{\beta}$ and β_0 specifying the maximum-margin hyperplane

21.4.3.1 Formulating the optimization problem

From figure 194 we see that the distance between a point \underline{x}^* and a hyperplane $g(\underline{x}) = \underline{w}^T \underline{x} - b = 0$ is given by

$$\text{distance} = \left| \frac{\underline{w}^T \underline{x}^*}{\|\underline{w}\|} - \frac{b}{\|\underline{w}\|} \right| = \frac{|g(\underline{x}^*)|}{\|\underline{w}\|} \quad (1113)$$

($\underline{w} = \underline{\beta}$, $b = -\beta_0$).

Therefore, maximizing the minimum distance to any point in $\mathcal{D} = \{\underline{x}_i, c_i\}$ is equivalent to

$$\underline{\beta}, \beta_0 = \underset{\underline{\beta}, \beta_0}{\text{argmax}} \left\{ \min_i \frac{c_i \cdot (\underline{x}_i^T \underline{\beta} + \beta_0)}{\|\underline{\beta}\|} \right\} \quad (1114)$$

where $c_i g(\underline{x}_i) = |g(\underline{x}_i)|$ for correct classification by the decision function, which we require.

21.4.3.2 Reformulating into constraint minimization

Any change in magnitude of $\underline{\beta}$ can be compensated by a change in β_0 with regards to the decision function. We therefore set

$$c_0 \cdot (\underline{x}_0^T \underline{\beta} + \beta_0) = 1 \text{ for the closest point } \underline{x}_0 \text{ to the hyperplane} \quad (1115)$$

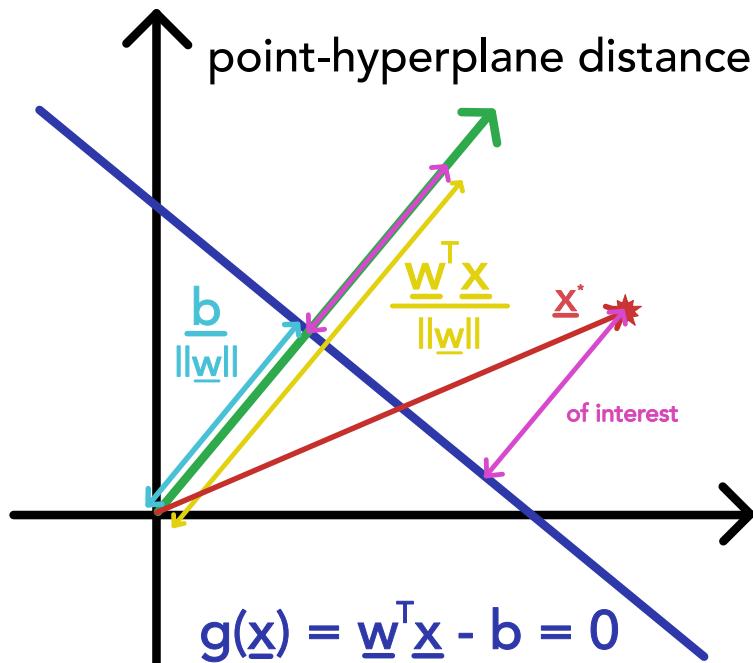


Figure 194: Distance between a point and a hyperplane.

without loss of generality, also defining the margins as $g(\underline{x}) = \pm 1$. From this we follow the constraint

$$\forall i : c_i \cdot (\underline{x}_i^T \underline{\beta} + \beta_0) \geq 1 \quad (1116)$$

so equation 1114 can be turned into minimizing $\|\underline{\beta}\|$, under the above constraint

$$\underline{\beta}, \beta_0 = \underset{\underline{\beta}, \beta_0}{\operatorname{argmin}} \frac{1}{2} \|\underline{\beta}\|^2 \text{ s.t. } c_i \cdot (\underline{x}_i^T \underline{\beta} + \beta_0) \geq 1, \forall i \in \{1, \dots, N\} \quad (1117)$$

which is called the **primal problem**.

This problem can be solved with a standard quadratic programming solver.

Problem: This primal problem scales mostly with the dimensionality of the feature matrix, which will be very large, when we later go to kernelized SVMs, which we do as in high-dimensions classes tend to become linearly separable.

By going from the *primal* to the *dual problem*, we can go from a problem scaling mostly with the dimensionality of the feature matrix to one scaling mostly with the number of data points. This makes problems with huge numbers of data points difficult, but shines for moderate amounts of data but huge feature vectors.

21.4.3.3 Intermezzo: Primal and Dual Problems in Quadratic Programming*

Consider the general problem (*primal problem*)

$$\underline{\beta}^* = \operatorname{argmin}_{\underline{\beta}} f(\underline{\beta}) \text{ s.t. } g_j(\underline{\beta}) \leq 0, h_k(\underline{\beta}) = 0 \quad (1118)$$

where $f(\underline{\beta})$ is quadratic in $\underline{\beta}$, and $g_j(\underline{\beta})$ and $h_k(\underline{\beta})$ are affine in $\underline{\beta}$ (linear plus constant). g_j and h_k are the *inequality* and *equality constraints*.

Every *primal problem* has a *dual problem*.

We get from the primal to the dual problem, by writing the Lagrangian

$$\mathcal{L}(\underline{\beta}, \underline{\alpha}, \underline{\lambda}) = f(\underline{\beta}) + \sum_j \alpha_j g_j(\underline{\beta}) + \sum_k \lambda_k h_k(\underline{\beta}) \quad (1119)$$

and by using the conditions for the extremum

$$\partial_{\beta_i} \mathcal{L} = 0, \quad \partial_{\lambda_k} \mathcal{L} = 0 \quad (1120)$$

to eliminate $\underline{\beta}$ from the Lagrangian, yielding the reduced Lagrangian $\mathcal{L}(\underline{\alpha}, \underline{\lambda})$.

By *strong duality* and the *Karush-Kuhn-Tucker conditions*, the solution of the dual problem

$$\underline{\alpha}^* = \operatorname{argmax}_{\underline{\alpha}} \mathcal{L}(\underline{\alpha}, \underline{\lambda}), \quad \text{s.t. } \alpha_j \geq 0 \quad (1121)$$

is the same as the solution of the primal problem.

21.4.3.4 Reformulating into a dual problem

We can write the minimization in terms of a Lagrangian

$$\mathcal{L}(\underline{\beta}, \beta_0, \underline{\alpha}) = \frac{1}{2} \underline{\beta}^T \underline{\beta} - \sum_{i=1}^N \alpha_i [c_i (\underline{x}_i^T \underline{\beta} + \beta_0) - 1] \quad (1122)$$

with *Lagrange multipliers* α_i .

By using

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \underline{\beta}} &= 0 \rightarrow \hat{\beta} = \sum_{i=1}^N \alpha_i c_i \underline{x}_i \\ \frac{\partial \mathcal{L}}{\partial \beta_0} &= 0 \rightarrow \sum_{i=1}^N \alpha_i c_i = 0 \end{aligned} \quad (1123)$$

with this we can simplify the Lagrangian

$$\begin{aligned}
 \mathcal{L}(\underline{\beta}, \beta_0, \underline{\alpha}) &= \frac{1}{2} \underline{\beta}^T \underline{\beta} - \sum_{i=1}^N \alpha_i [c_i (\underline{x}_i^T \underline{\beta} + \beta_0) - 1] \\
 &= \frac{1}{2} \underline{\beta}^T \underline{\beta} - \underbrace{\left(\sum_{i=1}^N \alpha_i c_i \underline{x}_i^T \right)}_{=\underline{\beta}^T} \underline{\beta} - \beta_0 \underbrace{\sum_{i=1}^N \alpha_i c_i}_{=0} + \sum_{i=1}^N \alpha_i \\
 &= -\frac{1}{2} \underline{\beta}^T \underline{\beta} + \sum_{i=1}^N \alpha_i
 \end{aligned} \tag{1124}$$

and finally eliminate $\underline{\beta}$, yielding the *dual problem*.

$$\underline{\alpha}^* = \underset{\underline{\alpha}}{\operatorname{argmax}} \left\{ -\frac{1}{2} \left\| \sum_{i=1}^N \alpha_i c_i \underline{x}_i \right\|^2 + \sum_{i=1}^N \alpha_i \right\} \text{ s.t. } \alpha_i \geq 0, \sum_{i=1}^N \alpha_i c_i = 0 \tag{1125}$$

We can rewrite this to

$$\underline{\alpha}^* = \underset{\underline{\alpha}}{\operatorname{argmin}} \frac{1}{2} \underline{\alpha}^T \operatorname{diag}(\underline{c}) \underbrace{\underline{X} \underline{X}^T}_{=:G \in \mathbb{R}^{N \times N}} \operatorname{diag}(\underline{c}) \underline{\alpha} - \underline{1}^T \underline{\alpha} \text{ s.t. } \underline{\alpha} \geq 0, \underline{c}^T \underline{\alpha} = 0 \tag{1126}$$

So we have turned a problem of finding $\underline{\beta} \in \mathbb{R}^p$ into a problem of finding $\underline{\alpha} \in \mathbb{R}^N$.

Here the data occurs only in form of the Gram matrix $G = \underline{X} \underline{X}^T$, which is a $N \times N$ matrix, so the scaling of the problem is benevolent for large feature vectors (large p)

Having obtained the optimal $\underline{\alpha}^*$ the decision function is

$$g(\underline{x}) = \underline{\beta}^T \underline{x} + \beta_0 = \sum_{i=1}^N \alpha_i^* c_i \underline{x}_i^T \underline{x} + \beta_0 \tag{1127}$$

So also in the decision function, we only need scalar products $\underline{x}_i^T \underline{x}$, which are later replaced by kernel functions (the same goes for the entries of the Gram matrix) - allowing for non-linear decision boundaries (separation in theoretically infinitely expanded feature space). And for the decision we only need $\underline{\alpha}^* \in \mathbb{R}^N$ and β_0 not $\underline{\beta} \in \mathbb{R}^p$, which is problematic for large p .

21.4.4 Relaxing the constraint of linear separability - soft-margin SVM

Idea: Add a regularization to the loss penalizing points crossing the margin (not the decision boundary), depending on how far they have strayed off from their correct side.

We use the penalty (*slack variable*)

$$\xi_i = \begin{cases} \text{point on the correct side of the margin} & \xi_i = 0 \\ \text{point on the wrong side of the margin} & \xi_i = |c_i - g(\underline{x}_i)| \end{cases} \geq 0 \forall i \quad (1128)$$

What are the support vectors in the relaxed case?: Now the points on the wrong side of the margin are also support vectors. Generally support vectors are those that have an influence on the location of the decision boundary. Note that SVM is not a probability model, we just draw a boundary. The support vectors are those where when we would leave away all other vectors in the data set we would still retain the same optimal boundary (which can of course be described by less parameters than by the support vectors).

A soft margin is illustrated in figure 195.

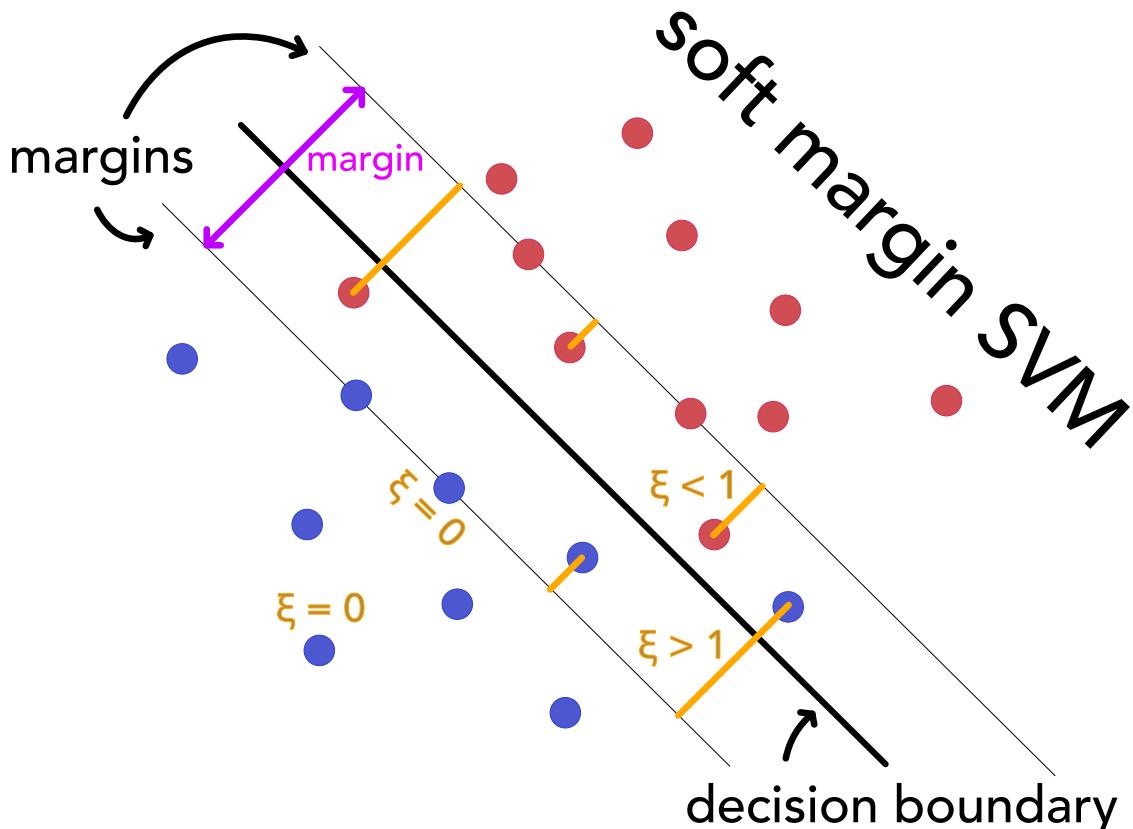


Figure 195: Soft margin.

The new optimization problem is

$$\begin{aligned} \underline{\beta}, \beta_0 &= \underset{\underline{\beta}, \beta_0}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\underline{\beta}\|^2 - \sum_{i=1}^N \alpha_i [c_i \cdot (\underline{x}_i^T \underline{\beta} + \beta_0) - 1] + \gamma \sum_{i=1}^N \xi_i - \sum_{i=1}^N \lambda_i \xi_i \right\} \\ \text{s.t. } c_i \cdot (\underline{x}_i^T \underline{\beta} + \beta_0) &\geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \lambda_i \geq 0 \end{aligned} \quad (1129)$$

where γ regulates the strength of the penalty.

Note on γ : Dividing by γ , we can interpret this as a regularization on $\|\underline{\beta}\|^2$.

- for large γ , we have less misclassification, but a smaller margin, as the margin is $\frac{1}{\|\underline{\beta}\|}$, and larger γ can be interpreted as a smaller regularization on $\|\underline{\beta}\|^2$
- for small γ , we have a larger margin, but more misclassification errors

This is illustrated in table 36. **The stronger the regularization γ , the smaller the number of support vectors.** As expected the more we penalize points on the wrong side of the margin, the closer the margins (less soft) and the less support vectors, so points influencing the position of the margin, we have.

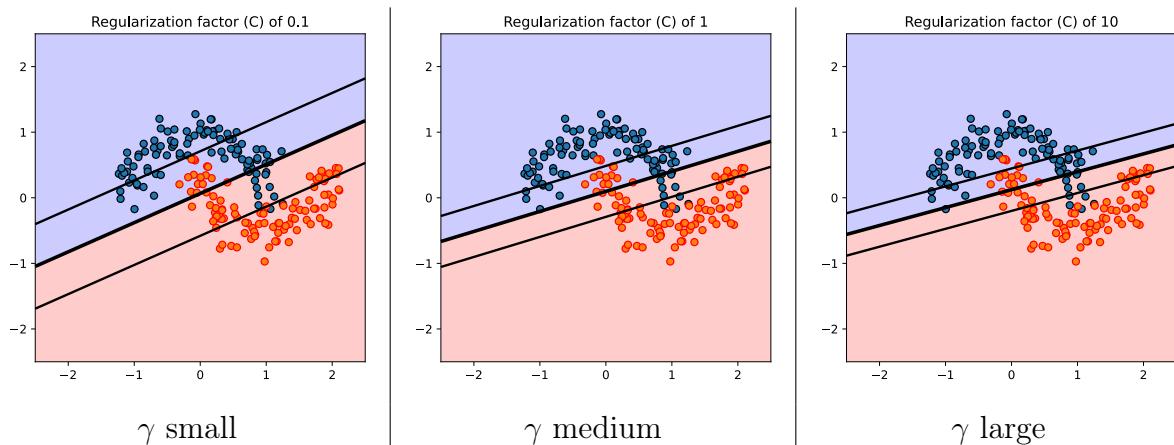


Table 36: Effect of γ (C in the figures) on the soft margin.

21.4.5 Kernelized SVM

Idea: By basis expansion - projection into a higher dimensional feature space - we can make a problem that is not linearly separable linearly separable, as illustrated in figure 196.

In a space with sufficiently high dimensionality, arbitrary complex patterns are very likely to be linearly separable (Cover's Function Counting theorem).

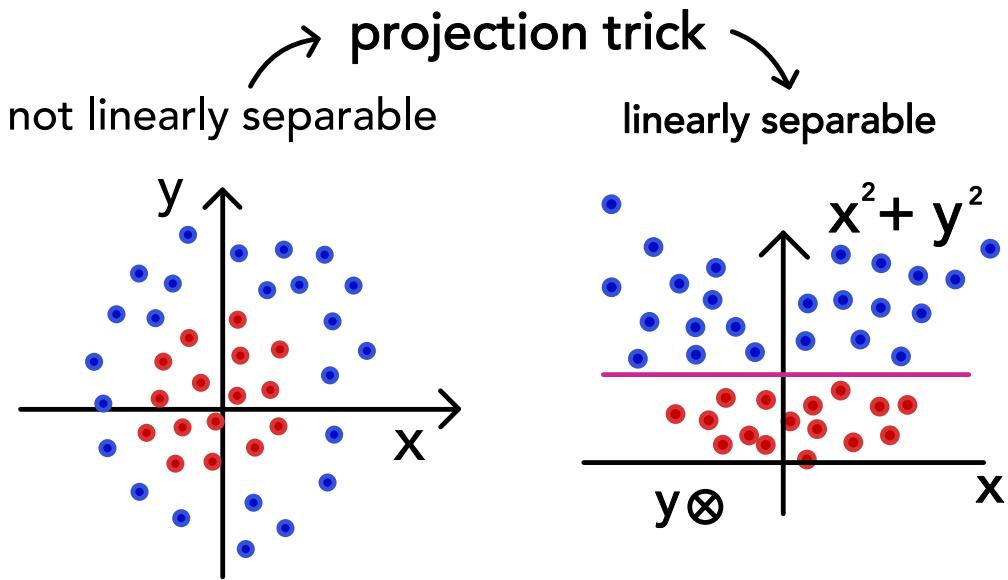


Figure 196: Projection trick.

We expand into a higher dimensional feature space by

$$\underline{\phi} : \mathbb{R}^p \rightarrow \mathbb{R}^q, \quad q > p \quad (1130)$$

for instance

$$\underline{\phi}(\underline{x}) = \begin{pmatrix} x_{i1} \\ \vdots \\ x_{ip} \\ x_{i1}^2 \\ x_{i1}x_{i2} \\ \vdots \\ x_{i(p-1)}x_{ip} \\ x_{ip}^2 \end{pmatrix} \quad (1131)$$

21.4.5.1 Kernel trick for high-D computations

Let us replace the scalar product in the high dimensional feature space by a kernel function

$$k(\underline{x}_i, \underline{x}_j) = \underline{\phi}(\underline{x}_i)^T \underline{\phi}(\underline{x}_j) \quad (1132)$$

which operates on the vectors in the original features space.

Example: Kernel for polynomial basis expansion For instance

$$k(\underline{x}_i, \underline{x}_j) = (1 + \underline{x}_i^T \underline{x}_j)^d \quad (1133)$$

based on the low-dimensional scalar product $\underline{x}_i^T \underline{x}_j$ is equivalent to a polynomial basis expansion of degree d . For $d = 2$ this corresponds to the feature expansion

$$\underline{\phi}(\underline{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \quad (1134)$$

Further Kernels: Sigmoid and Gaussian RBF

- **Sigmoid kernel:** $k(\underline{x}_i, \underline{x}_j) = \tanh(\kappa \underline{x}_i^T \underline{x}_j + \theta)$, parameters κ, θ
- **Gaussian RBF kernel:** $k(\underline{x}_i, \underline{x}_j) = \exp\left(-\frac{\|\underline{x}_i - \underline{x}_j\|^2}{2\sigma^2}\right)$, parameter σ

The Gaussian Kernel is a similarity measure projecting into an infinite-dimensional space of all polynomial kernels of degrees $p \geq 0$ (which we can see from Taylor expanding the Gaussian).

In figure 197 we see the effect of different kernels in SVMs.

21.4.5.2 Kernel SVM is where the dual formulation shines

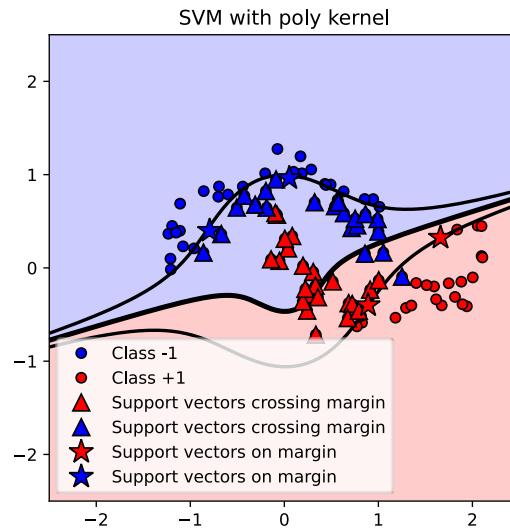
For Kernelized SVMs, in the dual formulation

$$\underline{\alpha}^* = \underset{\underline{\alpha}}{\operatorname{argmin}} \frac{1}{2} \underline{\alpha}^T \operatorname{diag}(\underline{c}) \underbrace{\underline{\underline{X}} \underline{X}^T}_{=: G \in \mathbb{R}^{N \times N}} \operatorname{diag}(\underline{c}) \underline{\alpha} - \underline{1}^T \underline{\alpha} \text{ s.t. } \underline{\alpha} \geq 0, \underline{c}^T \underline{\alpha} = 0 \quad (1135)$$

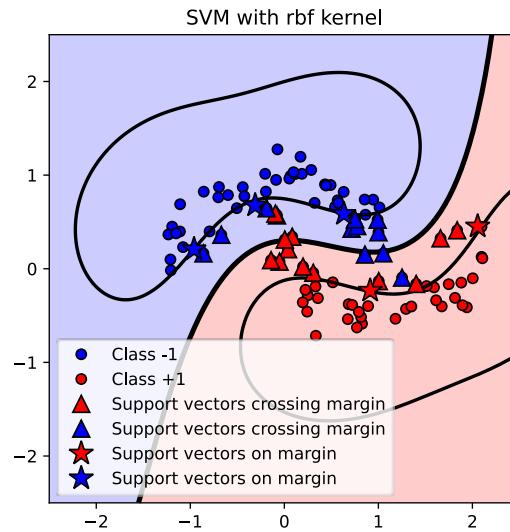
the Gram matrix changes to

$$\underline{\underline{G}} = \underline{\underline{\Phi}} \underline{\Phi}^T, \quad G_{ij} = k(\underline{x}_i, \underline{x}_j) \quad (1136)$$

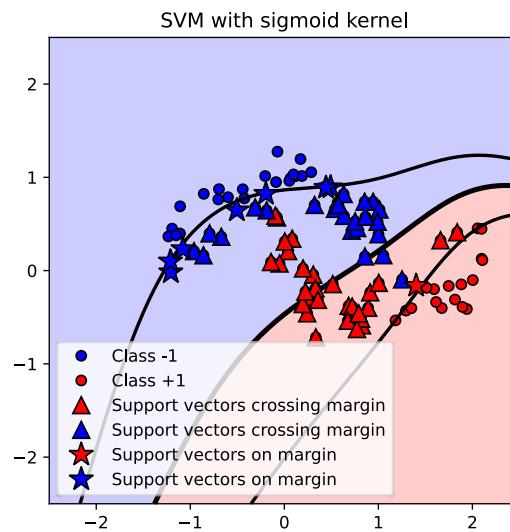
where we need $N \times N$ applications of the kernel function k .



(a) SVM with polynomial kernel.



(b) SVM with RBF kernel.



(c) SVM with sigmoid kernel.

Figure 197: Effect of different kernels in SVMs.

Classification of a new point \underline{x} is done by

$$g(\underline{x}) = \sum_{i=1}^N \alpha_i^* c_i k(\underline{x}_i, \underline{x}) + \beta_0, \quad N \text{ samples in training set} \quad (1137)$$

21.4.6 Multiclass SVM

A simple option is to train a 1-vs-all classifier for all classes and then use

$$\hat{c} = \operatorname{argmax}_k g_k(\underline{x}), \quad g_k(\underline{x}) = \underline{\beta}_k^T \underline{x} + \beta_{0k} \quad (1138)$$

Note: »but it cannot capture correlations between the different classes since it breaks a multiclass problem into multiple independent binary problems.« There are also approaches using multi-class objective functions (Crammer and Singer, 2001).

So everything just depends on the kernel and we can go to effectively infinite dimensions by a kernel function. **The dual formulation enables kernelization.**

21.5 K-Nearest Neighbor classifier

kNN is a non-parametric, algorithmic modeling approach, where a new point

- kNN-regression: is assigned the average of the k nearest neighbors
- kNN-classification: is assigned a class by majority vote of the k nearest neighbors

There is **no training needed** but the **model is the training data itself (so the model is possibly very large)**.

In classification, the probability of class l at a point \underline{x}_0 is given by

$$\hat{p}(c_0 = l | \underline{x}_0) = \frac{\#\text{nearest neighbors of class } l}{k} = \frac{|\{\underline{x}_i \in l \cap H_k(\underline{x}_0)\}|}{|\{\underline{x}_i \in H_k(\underline{x}_0)\}|} \quad (1139)$$

A weighted approach with closer neighbors having more influence is also possible.

21.5.1 Dependence of the classification on k - balance between over- and underfitting

We note

- **Small k :** more flexible, but more sensitive to noise (for $k = 1$ Delaunay triangulation, so Voronoi cells)
- **Large k :** less flexible, but more robust to noise

which is illustrated in table 37.

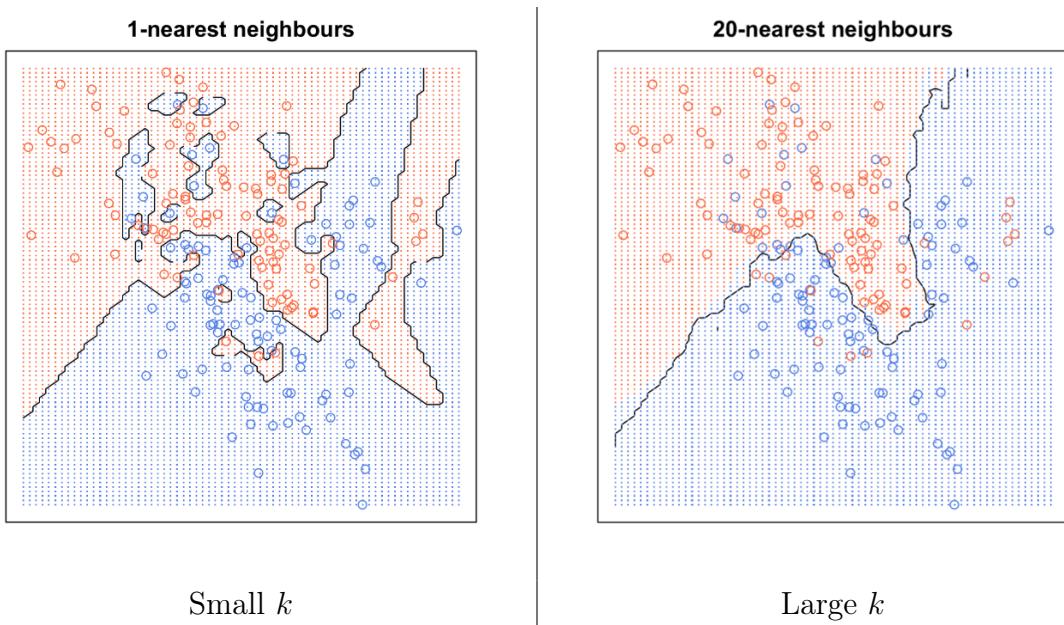


Table 37: Effect of k on kNN.

Number of effective parameters: While kNN is non-parametric, so there is no fixed number of parameters learned independent of the size of the training set, there are effectively $\sim \frac{k}{N}$ neighborhoods where one parameter - the class - is learned.

21.5.2 Advantages and Disadvantages of kNN

- **Advantages:** can approximate any distribution, fast training (only store the data)
- **Disadvantages:**
 - entire data-set is stored (large model); use e.g. Hart condensing to reduce the data set while keeping the decision boundary

21.6 Logistic classification

We have a data set $\mathcal{D} = \{\underline{x}_i, c_i\}$, $c_i \in \{0, 1\}$, and we want to model the probability of class 1 at a new point \underline{x} .

21.6.1 GLM perspective on logistic regression

At every point \underline{x} , we model the class probability by a Bernoulli distribution

$$p(c|\underline{x}) = \begin{cases} 1 - \mu(\underline{x}) & c = 0 \\ \mu(\underline{x}) & c = 1 \end{cases} \quad (1140)$$

where the model of $\mu(\underline{x})$ is given by the sigmoid

$$g^{-1}(\eta) = \mu = \sigma(\eta) = \frac{1}{1 + \exp(-\eta)}, \quad \eta = \underline{\beta}^T \underline{x} \quad (1141)$$

which in the GLM framework can be followed from the finding the canonical link function. For 1D data, this is illustrated in figure 198.

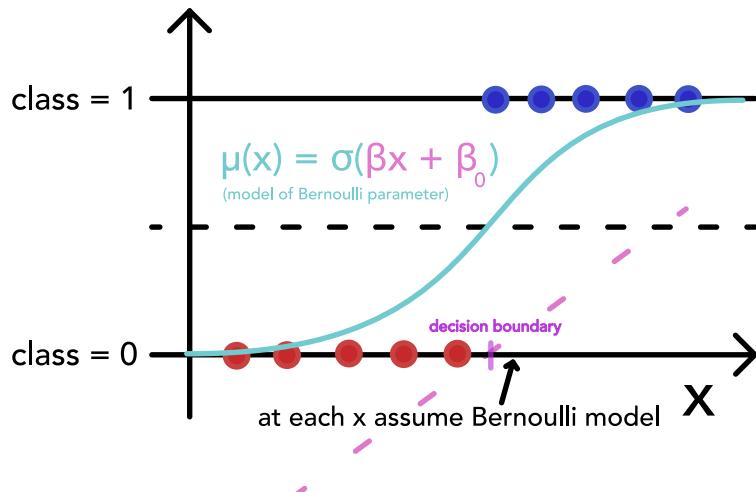


Figure 198: Logistic regression.

Note: Here we assume \underline{x} to have a constant 1 as first entry, allowing for an offset β_0 .

The link function is given by

$$g(\mu) = \eta = \underline{\beta}^T \underline{x} = \log \frac{\mu}{1 - \mu} \quad (1142)$$

So the logarithmic odds

$$\text{odds} = \frac{\text{probability that outcome occurs}}{\text{probability that outcome does not occur}} = \frac{\mu}{1 - \mu} \quad (1143)$$

are a linear function of the independent variables.

For $\underline{x} \in \mathbb{R}^p$, we have $\underline{\beta} \in \mathbb{R}^p$, so as many parameters to learn as features. Fitting a multidimensional normal to each class would generally be quadratic in p as of the covariance matrix.

Our decision rule is

$$\begin{aligned}\hat{c} &= \operatorname{argmax}_c p(c|\underline{x}; \underline{\beta}) \\ &= \begin{cases} 1 & \mu(\underline{x}) \geq 0.5 \\ 0 & \mu(\underline{x}) < 0.5 \end{cases} \\ &= \begin{cases} 1 & \log \frac{\mu(\underline{x})}{1-\mu(\underline{x})} \geq 0 \\ 0 & \log \frac{\mu(\underline{x})}{1-\mu(\underline{x})} < 0 \end{cases} \\ &= \begin{cases} 1 & \underline{\beta}^T \underline{x} \geq 0 \\ 0 & \underline{\beta}^T \underline{x} < 0 \end{cases}\end{aligned}\tag{1144}$$

21.6.2 Linear classifier perspective on logistic regression

Let us start by modeling the likelihood with a heaviside function $p(c = 1|\underline{x}) = \theta(g(\underline{x})) \in \{0, 1\}$, $g(\underline{x}) = \underline{w}^T \underline{x} - b$, where

$$\theta(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}\tag{1145}$$

This is illustrated in figure 199.

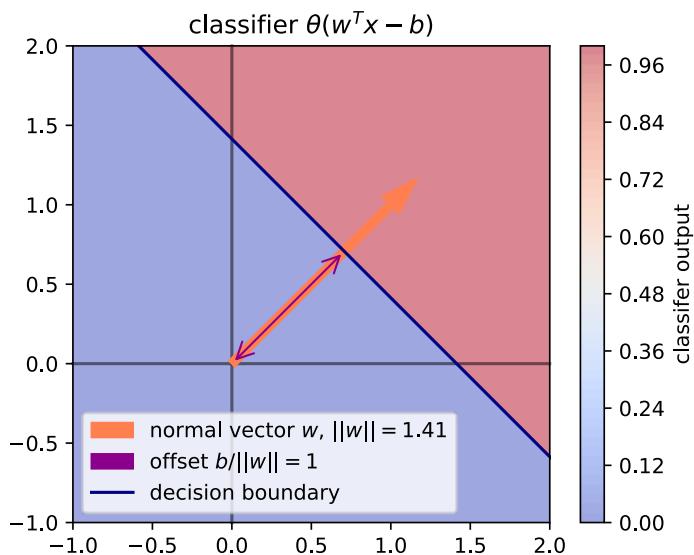


Figure 199: Heaviside classifier.

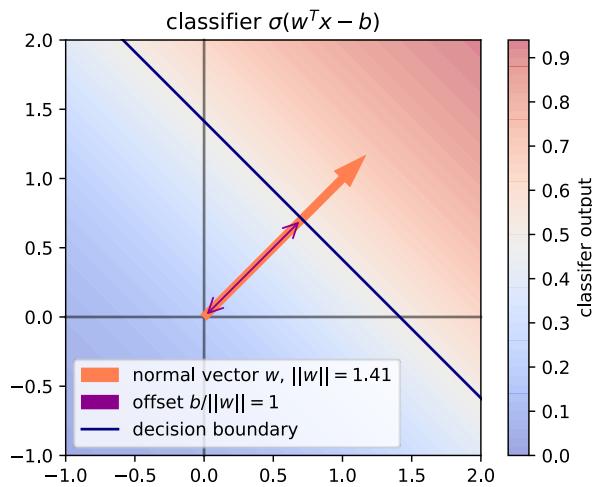
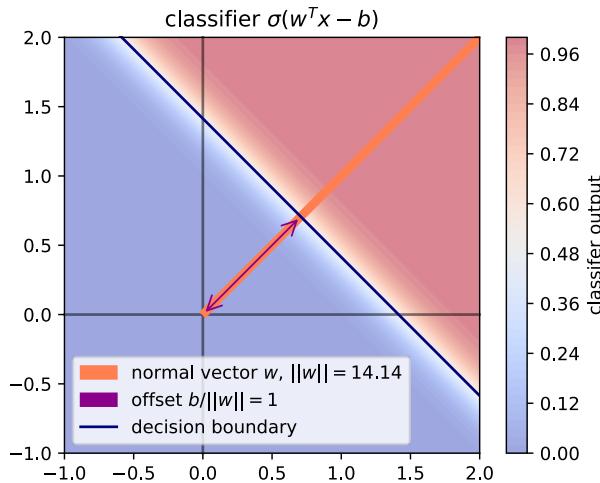
(a) Sigmoid classifier with small $||w||$.(b) Sigmoid classifier with large $||w||$.

Figure 200: Sigmoid classifier.

Problem: We cannot use the heaviside as a likelihood model, because there it does not differentiate between different boundaries separating the data.

Idea: Use the sigmoid $p(c = 1|\underline{x}) = \sigma(g(\underline{x}))$ as a likelihood model. Where $||\underline{w}||$ gives how fast the decision transitions from 0 to 1 and $\frac{b}{||\underline{w}||}$ gives the offset of the decision plane along \underline{w} . For $||\underline{w}|| \rightarrow \infty$ we have a step function again.

A sigmoid model for the likelihood is illustrated in figure 200.

21.6.3 Finding the parameters of $\mu(\underline{x})$, $\underline{\beta}$ by MLE

The likelihood of a single data point is given by

$$p(c_i | \underline{x}_i) = (1 - \sigma(\underline{\beta}^T \underline{x}_i))^{1-c_i} \sigma(\underline{\beta}^T \underline{x}_i)^{c_i} \quad (1146)$$

As usual, we minimize the negative log-likelihood

$$\begin{aligned} \hat{\underline{\beta}}, \hat{\beta}_0 &= \underset{\underline{\beta}, \beta_0}{\operatorname{argmin}} \sum_{i=1}^N -\log(p(c_i | \underline{x}_i)) \\ &= \sum_{i=1}^N -c_i \log \sigma(\underline{\beta}^T \underline{x}_i) - (1 - c_i) \log(1 - \sigma(\underline{\beta}^T \underline{x}_i)) \\ &= -\sum_{i=1}^N (c_i \underline{\beta}^T \underline{x}_i - \log(1 + \exp(\underline{\beta}^T \underline{x}_i))) \end{aligned} \quad (1147)$$

which is called logistic loss or cross-entropy loss or log-loss which is concave, so we use Newton-Raphson to find the minimum (there is no closed-form solution).

21.6.3.1 Understanding the terms in the negative log-loss as penalties

For a single data-point, the negative log-loss given μ and the true class c is illustrated in figure 201. Predicting $\mu > 0.5$ when the true class is 0 comes with an increasingly high penalty.

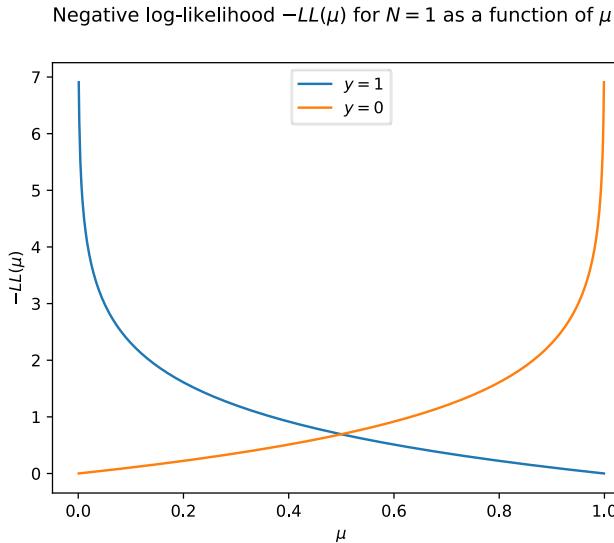


Figure 201: Negative log-loss for a single data point.

21.6.3.2 For linearly separable data, $\|\underline{\beta}\| \rightarrow \infty$ minimizes the loss

For linearly separable data

$$c_i = 1 \rightarrow \underline{\beta}^T \underline{x} \geq 0, \quad y_i = 0 \rightarrow \underline{\beta}^T \underline{x} < 0 \quad (1148)$$

is possible. Then $\|\underline{\beta}\| \rightarrow \infty$ minimizes the loss for any \underline{x} (we can also see this from the original loss as of $\sigma(\underline{\beta}^T \underline{x}) \rightarrow \theta(\underline{\beta}^T \underline{x})$ for $\|\underline{w}\| \rightarrow \infty$).

Intuition: In the large $\|\underline{\beta}\|$ limit, different classifiers are not distinguishable anymore by the likelihood, as illustrated in figure 202.

Note: For data that is not linearly separable, $\|\underline{\beta}\| \rightarrow \infty$ leads to divergent error terms so is not optimal, so $\|\underline{\beta}\| \rightarrow \infty$ is not optimal, see figure 203.

21.6.3.3 The decision boundary in logistic regression is linear

The decision boundary is at $\mu = \frac{1}{2}$, so

$$\underline{\beta}^T \underline{x} = \log \frac{\mu}{1 - \mu} = 0 \quad (1149)$$

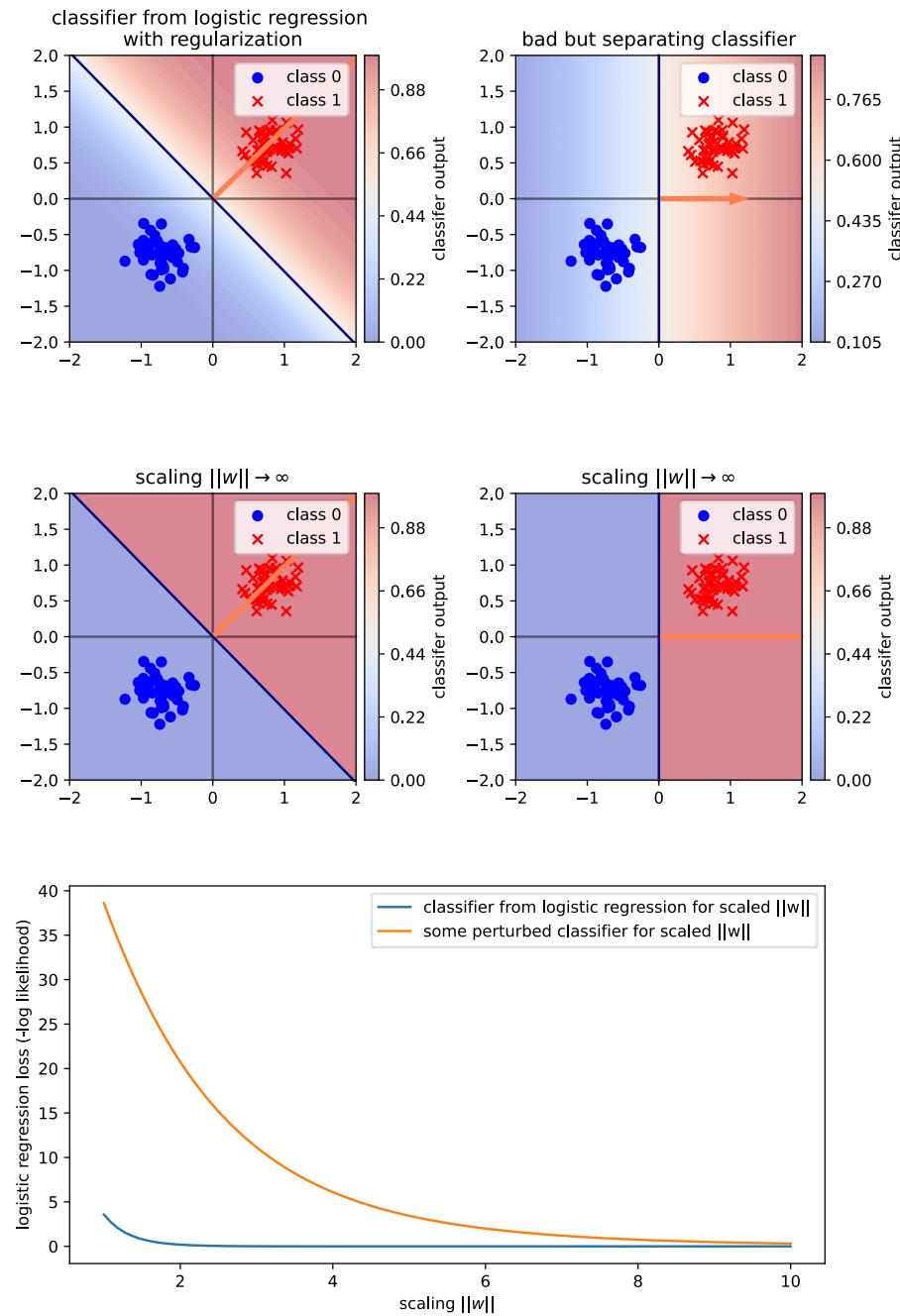


Figure 202: Large $\|\underline{\beta}\|$ limit for linearly separable data.

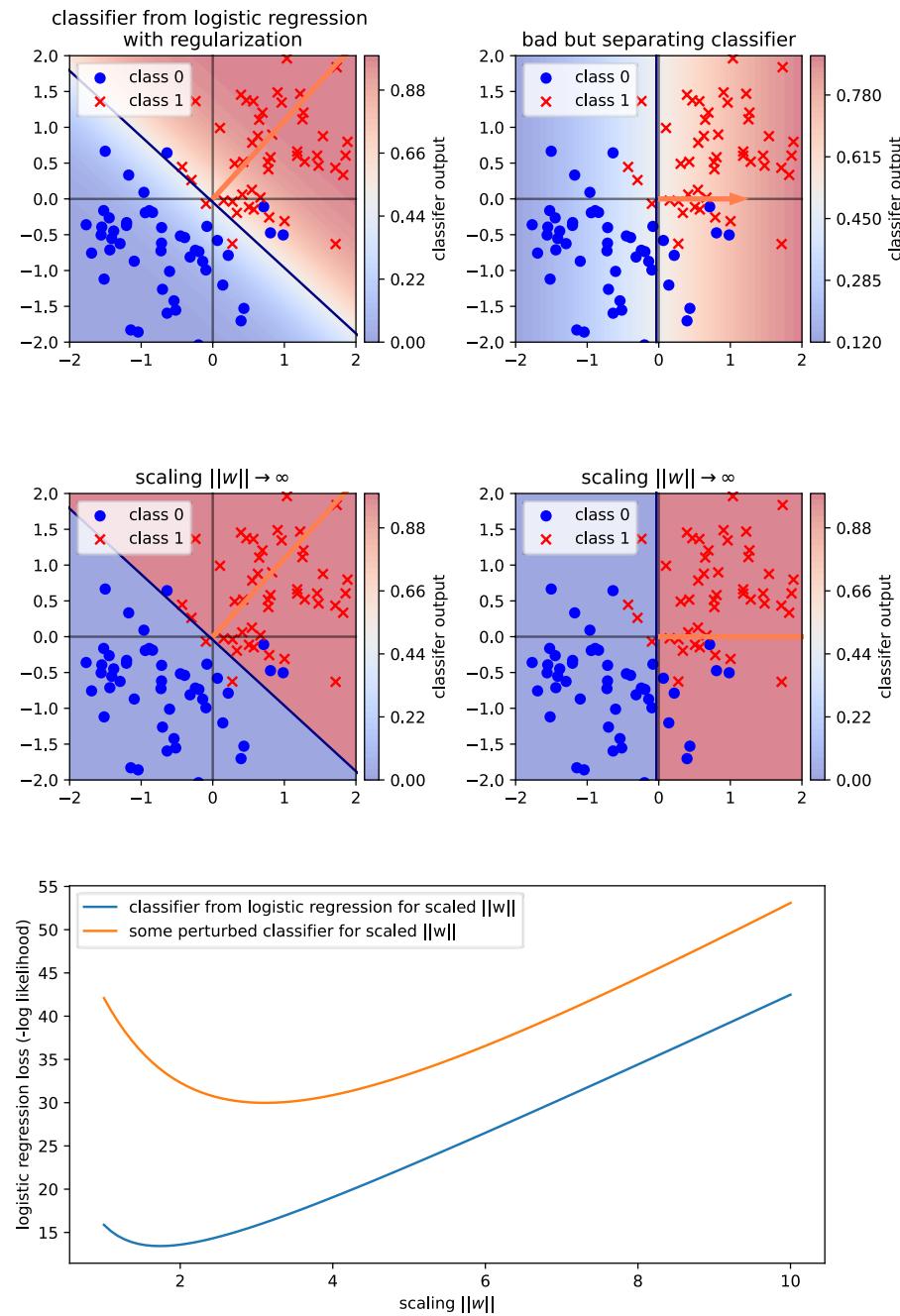


Figure 203: Large $\|\underline{\beta}\|$ limit for non-linearly separable data.

21.6.4 Multinomial Logistic Regression

Consider multiclass data $\mathcal{D} = \{\underline{x}_i, c_i\}$, $c_i \in \{1, \dots, K\}$, and we want to model the probability of class k at a new point \underline{x} .

We directly model the class probabilities, the parameters of a categorical model, by a soft(arg)max function. We do not make assumptions on the distribution of the features (like a Gaussian, as done in discriminant analysis).

21.6.4.1 Perspective of a multicategorical probability model

Our probability model for obtaining class $c_i = k$ at a point \underline{x}_i is given by the multicategorical

$$p(c_i = k | \underline{x}_i; \{\mu_\ell\}_{\ell=1}^K) = \mu_k(\underline{x}_i), \quad \sum_{k=1}^K \mu_k(\underline{x}_i) = 1 \quad (1150)$$

which if \underline{k} is the 1-hot encoding of k is given by

$$p(c_i = \underline{k} | \underline{x}_i; \{\mu_\ell\}_{\ell=1}^K) = \prod_{j=1}^K \mu_j(\underline{x}_i)^{k_j} \quad (1151)$$

where again, we model the parameters of our probability model $p_k = \mu_k$ as a function of the features \underline{x} .

Note: This is a different perspective than discriminant analysis. There we assume that given the class, the features are brought forth by a given distribution and from Bayes' theorem we can get class probabilities at a given location. Here we assume the \underline{x}_i to be fixed and directly model the likelihood of the given class probabilities under a probabilistic model which mean (/parameters) are a function of the features.

But what is our model for the class probabilities?

21.6.4.2 Derivation of the class probabilities | soft(arg)max

Let our starting point be, that the odds are linear in the parameters, as in the two-class case, but now with a reference class K .

$$\text{logistic odds for } \ell < K : \log \frac{p(c_i = \ell | \underline{x}_i)}{p(c_i = K | \underline{x}_i)} = \underline{\beta}_{\ell}^T \underline{x}_i, \quad \underline{x}_i = \begin{pmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{ip} \end{pmatrix} \quad (1152)$$

$$\rightarrow p_{\ell} = \exp(\underline{\beta}_{\ell}^T \underline{x}_i) \underbrace{p(c_i = K | \underline{x}_i)}_{p_K}$$

For the probability of the reference class, we can use

$$\sum_{\ell=1}^K p_{\ell} = 1 \quad \rightarrow \quad p_K = 1 - \sum_{\ell=1}^{K-1} p_{\ell} = 1 - \sum_{\ell=1}^{K-1} \exp(\underline{\beta}_{\ell}^T \underline{x}_i) p_K$$

$$\rightarrow p_K = \left[1 + \sum_{\ell=1}^{K-1} \exp(\underline{\beta}_{\ell}^T \underline{x}_i) \right]^{-1} \quad (1153)$$

giving us the class probabilities

$$p_{\ell} = \frac{\exp(\underline{\beta}_{\ell}^T \underline{x}_i)}{1 + \sum_{\xi=1}^{K-1} \exp(\underline{\beta}_{\xi}^T \underline{x}_i)} \quad (1154)$$

Assume we would apply the same log-odds formula to the reference class,

$$0 = \log \frac{p(c_i = K | \underline{x}_i)}{p(c_i = K | \underline{x}_i)} = \underline{\beta}_K^T \underline{x}_i \quad (1155)$$

which always holds only if $\underline{\beta}_K = 0$. With this we get the soft(arg)max function

$$p_{\ell} = \text{softargmax} \left(\begin{pmatrix} \underline{\beta}_1^T \underline{x}_i \\ \vdots \\ \underline{\beta}_K^T \underline{x}_i \end{pmatrix} \right)_{\ell} = \frac{\exp(\underline{\beta}_{\ell}^T \underline{x}_i)}{\sum_{\xi=1}^K \exp(\underline{\beta}_{\xi}^T \underline{x}_i)} \quad (1156)$$

21.6.4.3 An engineering style approach to multinomial logistic classification

Let us start somewhat fresh.

Given data $\mathcal{D} = \{\underline{x}_i, c_i\}$, $c_i \in \{1, \dots, K\}$, for a new point \underline{x} we want to estimate the class.

Idea: Use a linear decision function $g_k(\underline{x}) = \underline{\beta}_k^T \underline{x}, k = 1, \dots, K$ for each class and predict

$$\hat{c} = \operatorname{argmax}_k g_k(\underline{x}) = \operatorname{argmax}_k \underline{\beta}_k^T \underline{x} \quad (1157)$$

so if c is given in 1-hot encoding

$$\hat{c}_k = \begin{cases} 1 & k = \operatorname{argmax}_k \underline{\beta}_k^T \underline{x} \\ 0 & \text{else} \end{cases} \rightarrow \hat{c} \in \{0, 1\}^K \quad (1158)$$

Much more than a 1-hot result \hat{c} , we want a vector of class probabilities $\underline{\mu}(\underline{x})$, which we get by using the soft(arg)max function instead of the argmax function.

Let us collect the $\underline{\beta}_k$ in a matrix

$$\underline{\underline{B}} = \begin{pmatrix} - & \underline{\beta}_1^T & - \\ - & \vdots & - \\ - & \underline{\beta}_K^T & - \end{pmatrix} \quad (1159)$$

then at a point \underline{x} , the class probabilities are given by the model

$$\underline{\mu}(\underline{x}) = \operatorname{softargmax}(\underline{\underline{B}}\underline{x}) = \frac{1}{\sum_{\xi=1}^K \exp(\underline{\beta}_{\xi}^T \underline{x})} \begin{pmatrix} \exp(\underline{\beta}_1^T \underline{x}) \\ \vdots \\ \exp(\underline{\beta}_K^T \underline{x}) \end{pmatrix} \quad (1160)$$

which are the parameters of our categorical distribution, from which we follow the likelihood of the class labels given the \underline{x}_i and $\underline{\underline{B}}$, where $\underline{\underline{B}}$ can be found by MLE.

The formula 1160 can be represented by an engineering style diagram, see figure 204.

This diagram (fig. 204) represents the output layer of a neural network for a classification into K classes.

21.6.4.4 Perceptron for binary classification

Let us go back to the binary case, where we model $p(c = 1|\underline{x}) = \mu(\underline{x}) = \sigma(\underline{\beta}^T \underline{x})$ by a sigmoid function ($p(c = 0|\underline{x}) = 1 - \mu(\underline{x})$). This model is illustrated in figure 205 (with an offset here not by engineering on \underline{x} but a bias term b).

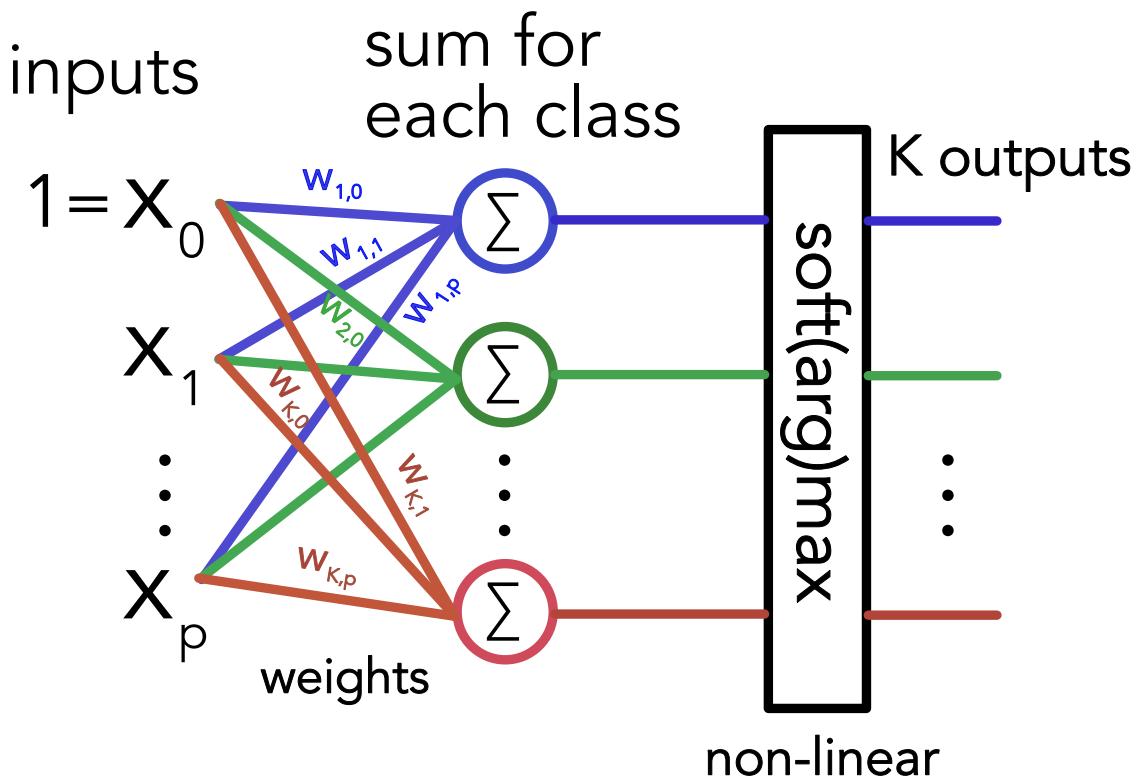


Figure 204: Soft(arg)max function in an engineering style diagram.

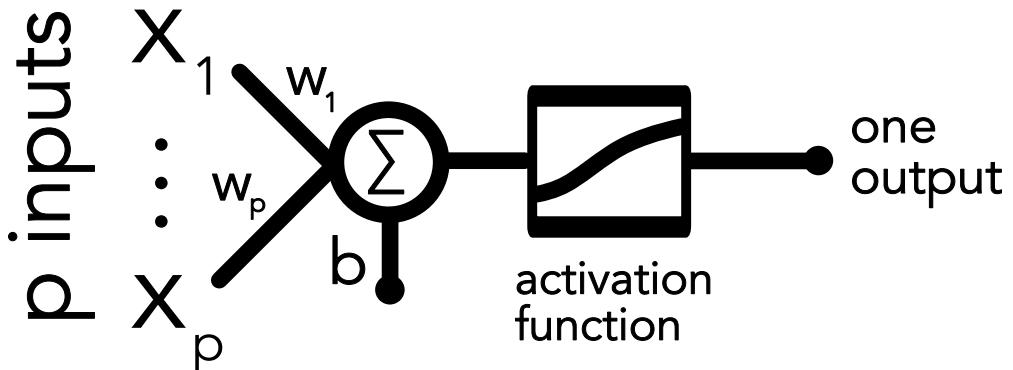


Figure 205: Perceptron.

The perceptron is visualizes the logistic regression model for binary classification. It can linearly separate classes with decision boundary perpendicular to β and offset $-\frac{b}{\|\beta\|}$ (minted here $+b$ not $-b$ as previously in the formula).

21.7 A primer on neural networks in the context of classification

Problem: The perceptron can only separate linearly separable data.

Idea: Use multiple layers to project the data into a higher dimensional space, where it is linearly separable.

A more complex separation of two classes in 1D is illustrated in figure 206, a 2D separation in figure 207.

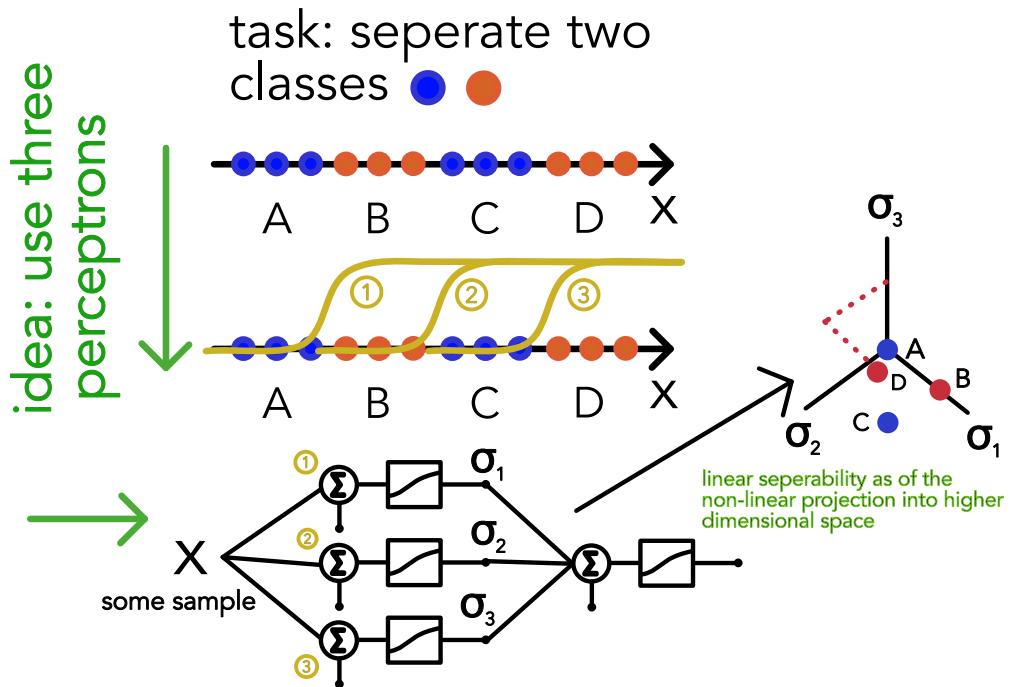


Figure 206: Complex separation of two classes in 1D.

Example: Multi-Layer perceptron for an XOR Gate: Consider an XOR with

$$(x_1, x_2) \in \{-1, 1\}^2, \quad \text{unusual convention } -1 : \text{true}, 1 : \text{false} \quad (1161)$$

A visualization and perceptron can be found in table 38.

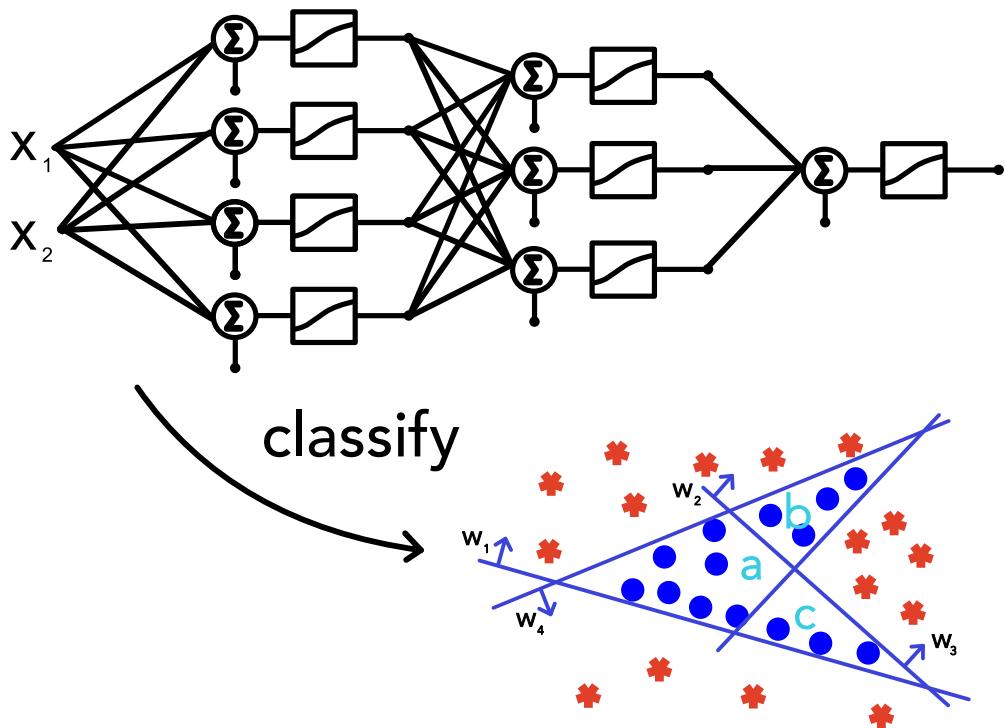


Figure 207: Complex separation of two classes in 2D.

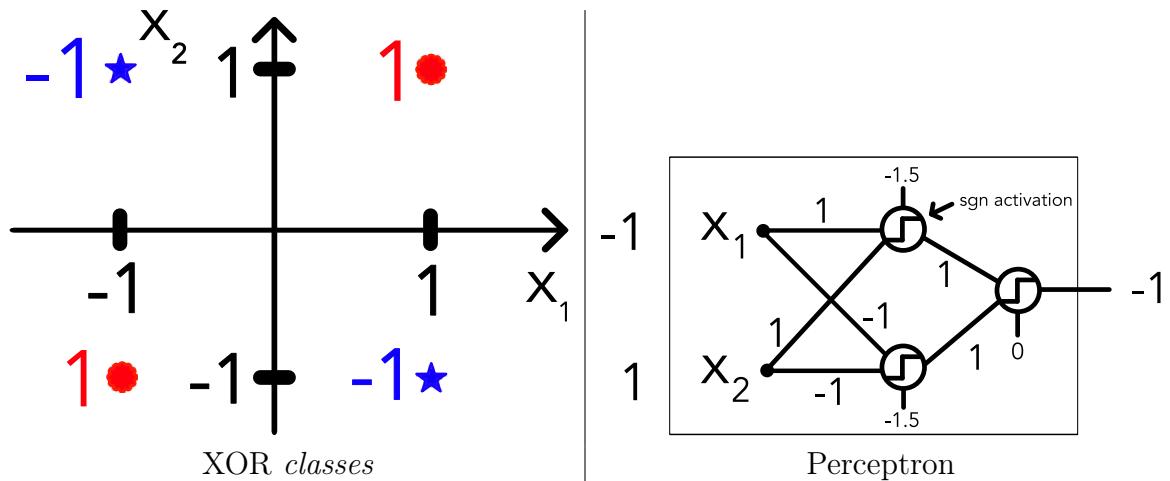


Table 38: XOR and Perceptron.

21.7.1 Neural Network Loss for Classification

How do we find the weights (and biases) \underline{W} (grouped into one vector) of a neural network for the classification of K classes?

After multiple layers, the output of our neural network are class probabilities

neural network for classification: $f_{\underline{W}}(\underline{x}_i) \in [0, 1]^K$, $\sum_{k=1}^K f_{W,k}(\underline{x}_i) = 1$ (1162)

How can we for each data pair $\underline{x}_i, \underline{c}_i$ (\underline{c}_i in 1-hot encoding) find the error of our prediction?

We use the cross-entropy loss

$$\epsilon = H(\{\underline{c}_i\}, \{\underline{f}_{\underline{W}}(\underline{x}_i)\}) = - \sum_{i=1}^N \underline{c}_i^T \log(\underline{f}_{\underline{W}}(\underline{x}_i)) \quad (1163)$$

Neural Network loss in a regression setting: In the regression setting

21.7.1.1 Intermezzo: Information measures for probability distributions

The shannon information content $h(x) = \log_2 \frac{1}{p(x)}$ of an event x is a measure of surprise - rare events (low $p(x)$) have high information content, as they are very surprising. The expected information content is the entropy

$$H(X) = - \sum_{x \in X} p(x) \log_2 p(x) \quad (1164)$$

which is high, if we expect a lot of surprise (rare events).

Perspective from information theory: We would like to send a message consisting of the letters $\{A, B, C, D\} = X$. Assuming all letters are equally likely, we could let the message consist of a sequence of two-bit letters, $A = 00, B = 01, C = 10, D = 11$. However, we could also encode unambiguously using $A = 0, B = 10, C = 110, D = 111$ if we assume $P(A) \gg P(B) \gg P(C) \gg P(D)$. $H(X)$ then is the number of bits we need on average given the probabilities $P(A), P(B), P(C), P(D)$ to encode a character.

21.7.1.2 Introduction of Cross-Entropy

Consider for the same feature vectors \underline{x}_i different probabilities \underline{p}_i and \underline{q}_i are assigned. Cross-entropy is a measure of dissimilarity between the two distributions

$$H(\underline{p}, \underline{q}) = -E_p[\log q] = - \sum_{k=1}^K p_k \log(q_k) \quad (1165)$$

For $p = q$, we get back to the entropy

$$H(\underline{p}, \underline{p}) = H(\underline{p}) = -E_p[\log p] = - \sum_{k=1}^K p_k \log(p_k) = H(\underline{p}) \geq 0 \text{ as } 0 \leq p_k \leq 1 \quad (1166)$$

which by Gibb's inequality

$$H(\underline{p}, \underline{q}) \geq H(\underline{p}), \quad \text{with equality if and only if } \underline{p} = \underline{q} \quad (1167)$$

is the minimum of the cross entropy.

The more similar two distributions are, the lower the cross-entropy.

21.7.1.3 Relation between cross-entropy and the negative log-likelihood - they're the same

Consider for a single data point from the training set \underline{x}_i , our model predicts the class probabilities

$$\underline{\mu} = f_{\underline{W}}(\underline{x}_i) \in [0, 1]^K, \quad \sum_{k=1}^K \mu_k = 1 \quad (1168)$$

Problem: We do not have the true distribution of the classes, but only the true class \underline{c}_i in 1-hot encoding.

Idea: Take \underline{c}_i as a proxy for the true distribution of the classes.

With this for the class distribution at a single \underline{x}_i , the cross-entropy loss is given by

$$\epsilon_i = H(\underline{c}_i, \underline{\mu}) = - \sum_{k=1}^K c_{ik} \log(\mu_k) = - \log \prod_{k=1}^K \mu_k^{c_{ik}} \quad (1169)$$

which is just the negative log likelihood at this point. Summing over all data points we obtain the total loss.

22 A Primer on Neural Networks*

We have already seen that we can write logistic and multiclass classification in a *electrical engineering style* diagram. This can analogously be done for linear regression.

22.1 The final layer of a neural network for classification and regression

Assume a linear model

$$\hat{y}_i = \beta_0 + \sum_{j=1}^p \beta_j x_{ij} = 1(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}) \quad (1170)$$

where $1(\cdot)$ is the identity function. We can illustrate this in the same style, we used for logistic regression, as illustrated in figure 208.

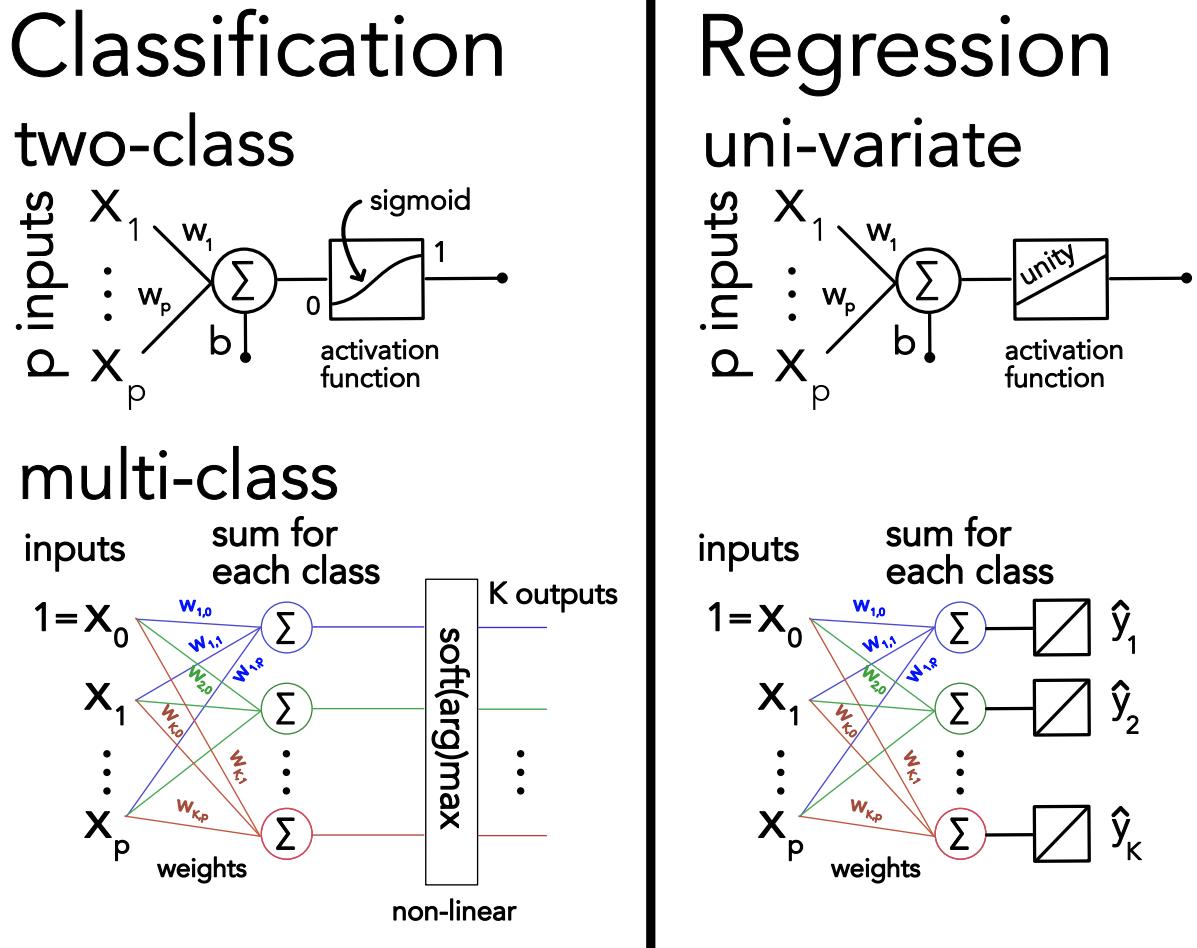


Figure 208: Electrical engineering style diagrams.

A comparison between classification and regression is given in table 39.

	Regression	Classification
Data	(Multiple) Continuous output(s) \underline{y} $\{\underline{x}_i, \underline{y}_i\}_{i=1}^N$ $\underline{x}_i \in \mathbb{R}^p, \underline{y}_i \in \mathbb{R}^K$	1-hot class label $\underline{c} \in \{0, 1\}^K$ $\{\underline{x}_i, \underline{c}_i\}_{i=1}^N$ $\underline{x}_i \in \mathbb{R}^p, \underline{c}_i \in \{0, 1\}^K, \sum_{k=1}^K c_{ik} = 1$
Model	$\hat{\underline{y}} = \underline{W}\underline{x}$	$\hat{\underline{c}} = \text{softargmax}(\underline{W}\underline{x})$
Loss	Squared residuals loss $\mathcal{L}(\underline{y}, \hat{\underline{y}}) = \frac{1}{N} \sum_{i=1}^N (\underline{y}_i - \hat{\underline{y}}(\underline{x}_i))^2$	Cross-entropy loss $\mathcal{L}(\underline{c}, \hat{\underline{c}}) = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K \underline{c}_i^T \log(\hat{\underline{c}}(\underline{x}_i))$

Table 39: Aim in both cases: Find \underline{W} that minimizes the loss. An offset is achieved by a 1 entry in the \underline{x} vector. We are in a supervised setting with supervisions \underline{y} and \underline{c} respectively.

22.2 Adding hidden layers - path to a neural network

Why should we add more layers?: We have previously seen, that by the function counting theorem, projection to higher dimensions can make a problem linearly separable. In the context of linear regression a basis expansion helped us to fit more complex non-linear functions.

We can also remember the generalized linear model, where we use a mean function g applied to a linear model, to have a mean prediction

$$\mu_i = g(\underline{\beta}^T \underline{x}_i) \quad (1171)$$

Idea: Construct a whole network of such nodes, where the outputs of one layer are fed forward into the next layer, first an input layer, then *hidden layers* and finally the output layer, different for classification and regression, as discussed. This is illustrated in figure 209.

Later our central question will be, what weights to choose to minimize the loss.

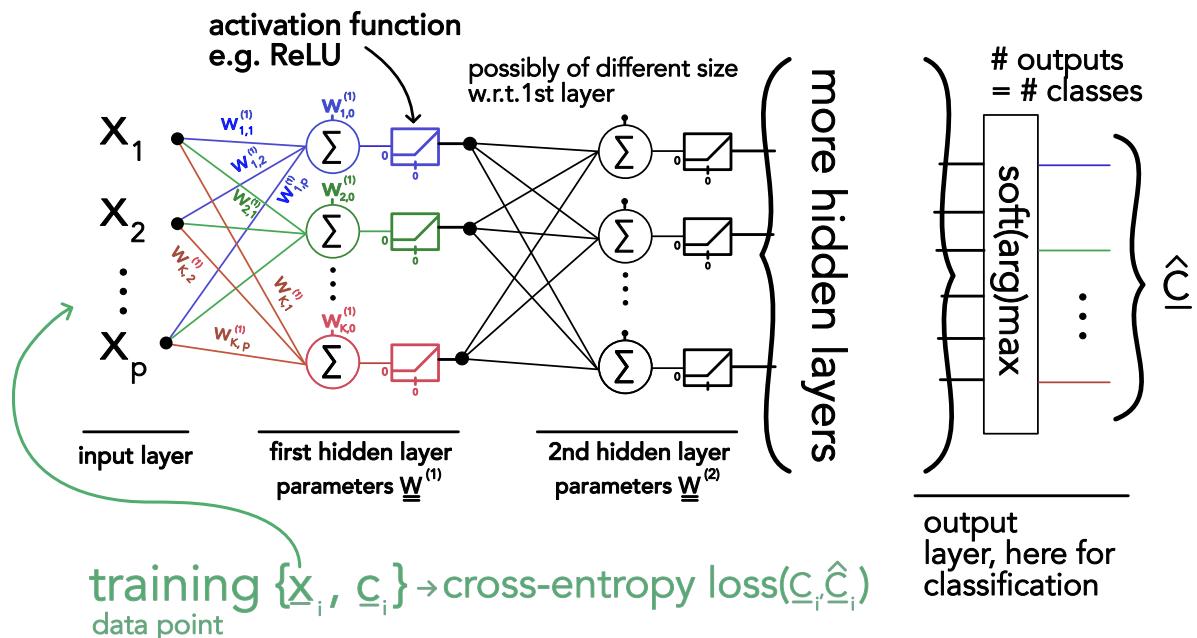


Figure 209: A neural network.

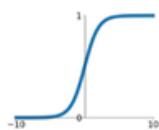
22.2.1 On the choice of activation function

Different activation functions are illustrated in figure 210. Based on their different shapes, they give different outputs, e.g. different decision boundaries in classification or different fits in regression. Their properties also effect the training, so finding appropriate weights and biases (the offset terms).

Activation Functions

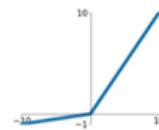
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



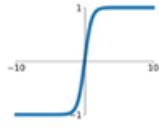
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$



Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ReLU

$$\max(0, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

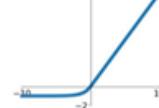


Figure 210: Different activation functions.

22.2.2 Neural Network are universal function approximators

By a single hidden layer, a neural network with finite number of nodes can approximate any continuous function on a compact set to arbitrary precision.

Consider a continuous, non-constant bounded activation function

$$g(z) \rightarrow \begin{cases} 1 & z \rightarrow \infty \\ 0 & z \rightarrow -\infty \end{cases} \quad (1172)$$

then any continuous f , here on $I_p = [0, 1]^p$, can be approximated by a finite size neural network

$$\forall \epsilon > 0, \forall f \in C(I_p), \exists \text{finite sum } F_{\underline{\theta}}(\underline{x}) = \sum_{i=1}^N \theta_i g(\underline{w}_i^T \underline{x} + w_{0i}) : \forall \underline{x} \in I_p, |f(\underline{x}) - F_{\underline{\theta}}(\underline{x})| < \epsilon \quad (1173)$$

22.2.3 Neural Networks and Algorithmic modeling

A neural network $F_{\underline{\theta}}$ (parameters $\underline{\theta}$) is deterministic universal approximator. When we use a neural network to predict the relationship between input \underline{x} and output \underline{y} , we directly model the relation between input and output, it is an algorithmic modeling approach (as random forests), different from a data modeling approach, where we would assume a certain model that has generated the data and some noise term (Breiman, 2001) (and usually predict $E[y|x]$ under the model).

While in algorithmic modeling, we are not limited by a certain, possibly wrong model, interpretability is usually limited.

The cultures of modelling are illustrated in figure 211.

Cultures of modelling

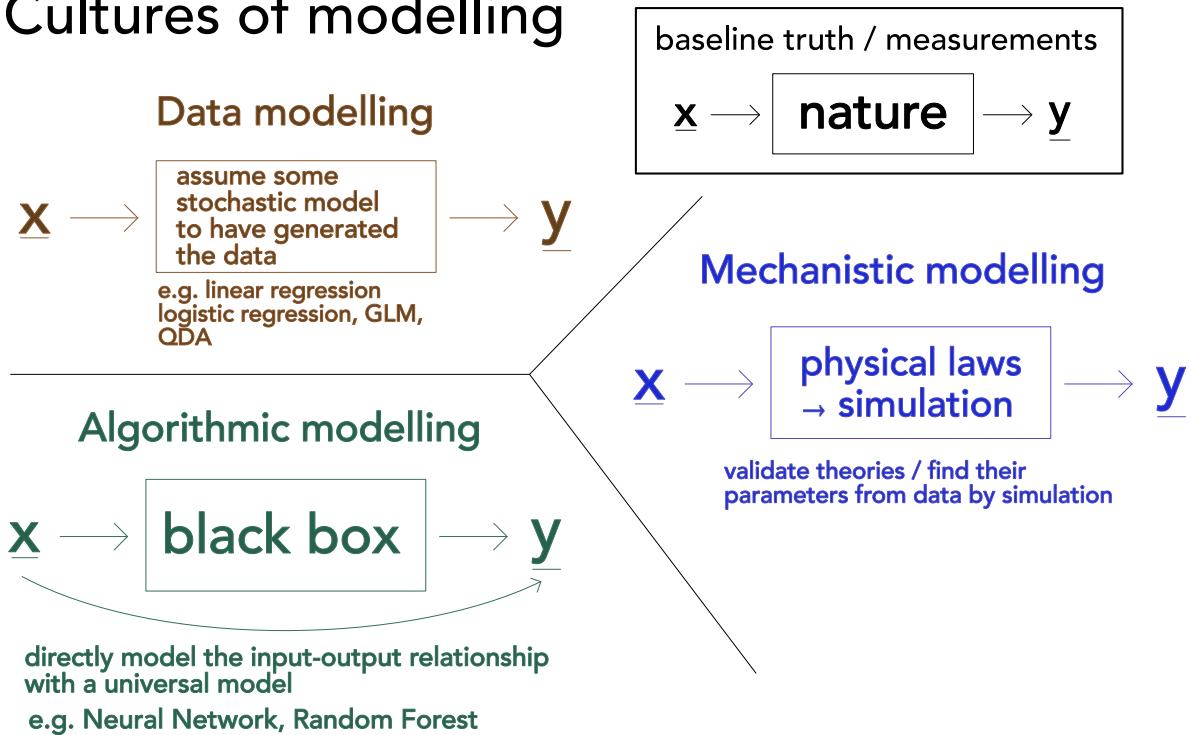


Figure 211: Different cultures of modelling.

22.3 Training of a neural network

22.3.1 Problem statement of training NNs

Let us consider a network with parameters collected into a vector \underline{W} . For an observation \underline{x}_i the network output is $f(\underline{x}_i, \underline{W})$. The supervision in the training set is \underline{y}_i , $\mathcal{D} = \{(\underline{x}_i, \underline{y}_i)\}_{i=1}^N$.

We can formulate the training problem as

$$\hat{\underline{W}} = \underset{\underline{W}}{\operatorname{argmin}} \mathcal{Q}(\underline{W}) = \underset{\underline{W}}{\operatorname{argmin}} \sum_{i \in \mathcal{D}} \text{loss}(f(\underline{x}_i, \underline{W}), \underline{y}_i) + \Omega_{\text{architecture}}(\underline{W}) + \Omega_{\text{output}}(f(\underline{x}_i, \underline{W})) \quad (1174)$$

with regularizations $\Omega_{\text{architecture}}$ and Ω_{output} .

In the regression setting the loss would be

$$\text{loss}(f(\underline{x}_i, \underline{W}), \underline{y}_i) = (f(\underline{x}_i, \underline{W}) - \underline{y}_i)^2 \quad (1175)$$

22.3.2 Stochastic Gradient Descent for minimizing the loss

We use gradient descent for the parameter estimates.

22.3.2.1 Normal gradient descent

Starting from an initial guess $\underline{W}^{(0)}$, we update the parameters by

$$\underline{W}^{(t)} = \underline{W}^{(t-1)} - \alpha \underline{g}^{(t)}, \quad \underline{g}^{(t)} = \frac{\partial Q(\underline{W})}{\partial \underline{W}} \Big|_{\underline{W}^{(t-1)}}, \quad \text{learning rate } \alpha \quad (1176)$$

where the loss is calculated over the whole training dataset

$$Q(\underline{W}) = \sum_{i \in \mathcal{D}} \text{loss} \left(f(\underline{x}_i, \underline{W}), \underline{y}_i \right) \quad (1177)$$

22.3.2.2 Stochastic gradient descent

We introduce stochasticity (against local minima) and reduce the length of the summation by drawing an index subset $k^* \subset 1, \dots, N$ so a corresponding mini-batch dataset $\mathcal{D}^* \subset \mathcal{D}$

$$Q(\underline{W}) = \sum_{i \in \mathcal{D}^*} \text{loss} \left(f(\underline{x}_i, \underline{W}), \underline{y}_i \right) \quad (1178)$$

22.3.3 How to compute the gradients? | backpropagation

22.3.3.1 Gradient we want to compute

At each update step in gradient descent, we want to compute

$$\frac{\partial Q(\underline{W})}{\partial \underline{W}} = \sum_{i \in \mathcal{D}^*} \frac{\partial \text{loss} \left(f(\underline{x}_i, \underline{W}), \underline{y}_i \right)}{\partial \underline{W}} \quad (1179)$$

So to update our weights from $\underline{W}^{(t-1)}$ to $\underline{W}^{(t)}$ we need to compute

$$\frac{\partial \text{loss} \left(\underline{y}_i, f(\underline{x}_i, \underline{W}) \right)}{\partial \underline{W}} \Big|_{\underline{W}^{(t-1)}} = \begin{pmatrix} \frac{\partial \mathcal{Q}(\underline{W})}{\partial \underline{W}_1} \Big|_{\underline{W}^{(t-1)}} \\ \frac{\partial \mathcal{Q}(\underline{W})}{\partial \underline{W}_2} \Big|_{\underline{W}^{(t-1)}} \\ \vdots \\ \frac{\partial \mathcal{Q}(\underline{W})}{\partial \underline{W}_j} \Big|_{\underline{W}^{(t-1)}} \\ \frac{\partial \mathcal{Q}(\underline{W})}{\partial \underline{W}_{\mathfrak{P}}} \Big|_{\underline{W}^{(t-1)}} \end{pmatrix} \quad (1180)$$

on each training sample $\{\underline{x}_i, \underline{y}_i\}$ in the minibatch, to later sum over the minibatch

22.3.3.2 Why can't we use finite differencing?

The derivative with respect to one weight can be approximated by

$$\frac{\partial Q(\underline{W})}{\partial W_k} = \frac{\text{loss}(\underline{y}_i, f(\underline{x}_{i'}(W_1, \dots, W_k + \Delta W, \dots, W_M))) - \text{loss}(\underline{y}_{i'}, f(\underline{x}_i, \underline{W}))}{\Delta W} \quad (1181)$$

Problem:

- Per training data sample we would need as many computations of the loss, so forward passes through the network, as there are weights - infeasible.
- The finite difference approximation is not exact.

22.3.3.3 Backpropagation to the help

Neural Network as a composition of functions Let

- L be the number of layers in our network
- \underline{x} be our sample feature with target output \underline{y}

Let $\underline{a}^{(l-1)}$ be the input to the l -th layer, which consists of

- the matrix containing the weights of the l -th layer $\underline{\underline{W}}^{(l)}$ with elements $w_{j,k}^{(l)}$ sometimes denoted $w_k^{l,j}$. $w_{j,k}^{(l)}$ is the weight from the k -th node in the $l-1$ -th layer to the j -th node in the l -th layer. After weighting, we get $\underline{z}^{(l)} = \underline{\underline{W}}^{(l)} \underline{a}^{(l-1)}$.
- the activation functions $\underline{h}^{(l)}$ of the l -th layer

Then the output of the l -th layer is

$$\underline{a}^{(l)} = \underline{h}^{(l)}(\underline{z}^{(l)}) = \underline{h}^{(l)}(\underline{\underline{W}}^{(l)} \underline{a}^{(l-1)}) \quad (1182)$$

The output of our whole network can then be written as

$$f(\underline{x}, \underline{W}) = \underline{a}^{(L)} = \underline{h}^{(L)}(\underline{\underline{W}}^{(L)} \underline{a}^{(L-1)}) = \underline{h}^{(L)}(\underline{\underline{W}}^{(L)} \underline{h}^{(L-1)}(\underline{\underline{W}}^{(L-1)} \dots \underline{h}^{(1)}(\underline{\underline{W}}^{(1)} \underline{x}))) \quad (1183)$$

Chain rule to the help for efficiently calculating gradients Consider the exemplary network in figure 212.

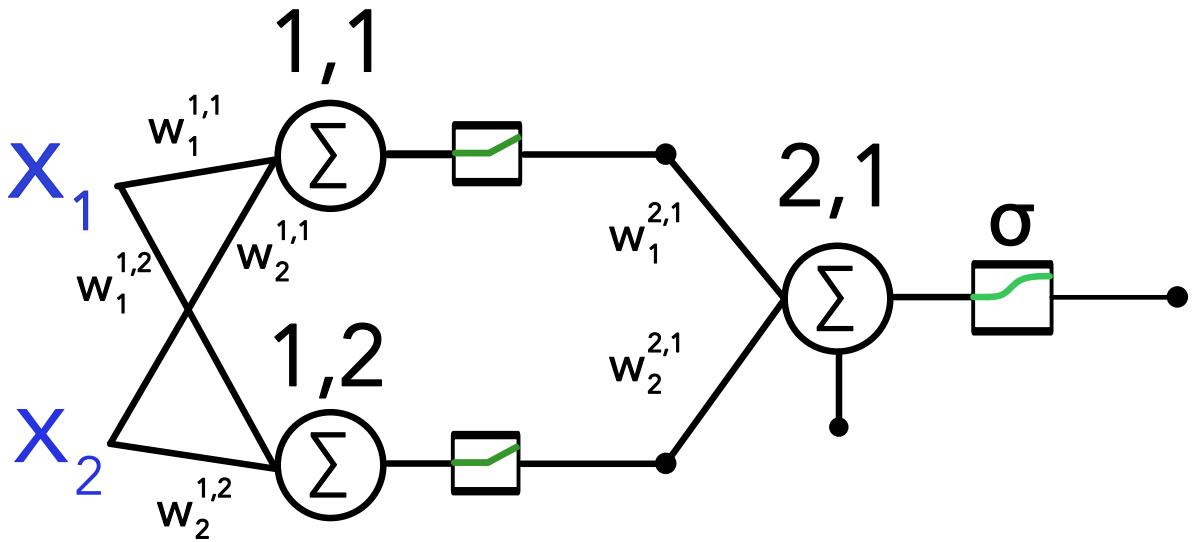


Figure 212: A simple neural network.

Aim: We want to calculate the partial derivatives of $\text{loss}(\underline{f}(\underline{x}, \underline{W}), \underline{y})$ with respect to for instance $w_{1,1}^{(1)} = w_1^{1,1}$ and $w_{1,2}^{(1)} = w_2^{1,1}$ of the first layer.

Chain rule gives us

$$\begin{aligned} \frac{\partial \text{loss} \left(\underline{y}, \sigma \left(\underline{\underline{W}}^{(2)} \underline{h}^{(1)} \left(\underline{\underline{W}}^{(1)} \underline{x} \right) \right) \right)}{w_{1,1}^{(1)}} &= \frac{\partial \text{loss}(\dots)}{\partial \sigma(\dots)} \frac{\partial \sigma(\dots)}{\partial \underline{h}^{(1)}(\dots)} \frac{\partial \underline{h}^{(1)}(\dots)}{\partial \underline{\underline{W}}^{(1)} \underline{x}} \frac{\partial \underline{\underline{W}}^{(1)} \underline{x}}{\partial w_{1,1}^{(1)}} \\ \frac{\partial \text{loss} \left(\underline{y}, \sigma \left(\underline{\underline{W}}^{(2)} \underline{h}^{(1)} \left(\underline{\underline{W}}^{(1)} \underline{x} \right) \right) \right)}{w_{1,2}^{(1)}} &= \frac{\partial \text{loss}(\dots)}{\partial \sigma(\dots)} \frac{\partial \sigma(\dots)}{\partial \underline{h}^{(1)}(\dots)} \frac{\partial \underline{h}^{(1)}(\dots)}{\partial \underline{\underline{W}}^{(1)} \underline{x}} \frac{\partial \underline{\underline{W}}^{(1)} \underline{x}}{\partial w_{1,2}^{(1)}} \end{aligned} \quad (1184)$$

- calculating this from left to right is called *reverse mode differentiation* or *backpropagation*
- calculating this from right to left is called *forward mode differentiation*

Note that when going from left to right, the partial derivatives of the example only differ in the last term.

For a system with few inputs and many outputs, we prefer the forward mode, for a system with many inputs (the weights are also kind of inputs to the system), but few outputs, we prefer the reverse mode.

22.3.3.4 Intermezzo: Automatic differentiation

Let us better understand the concept of derivatives propagating forwards and backwards.

On the derivatives of composite functions Consider we have a function $\underline{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and we want to calculate

$$\frac{\partial f_i}{\partial x_i} \quad (1185)$$

Now consider f is a **composite function**

$$\underline{f}(\underline{x}) = \underline{h} \circ \underline{g}(\underline{x}) = \underline{h}(g(\underline{x})), \quad \underline{x} \in \mathbb{R}^n, g : \mathbb{R}^n \rightarrow \mathbb{R}^p, h : \mathbb{R}^p \rightarrow \mathbb{R}^m \quad (1186)$$

then the chain rule is given by

$$\frac{\partial f_i}{\partial x_j} = \sum_{k=1}^p \frac{\partial h_i}{\partial g_k} \frac{\partial g_k}{\partial x_j} \quad (1187)$$

so intuitively to see how wiggling at x_j changes f_i , we look how wiggling at x_j affects the intermediate g_k and how wiggling at g_k affects f_i and chain the result. The point of evaluation is the x_j where we wiggle around.

Writing a function as a computational graph Consider the computational graph in figure 213.

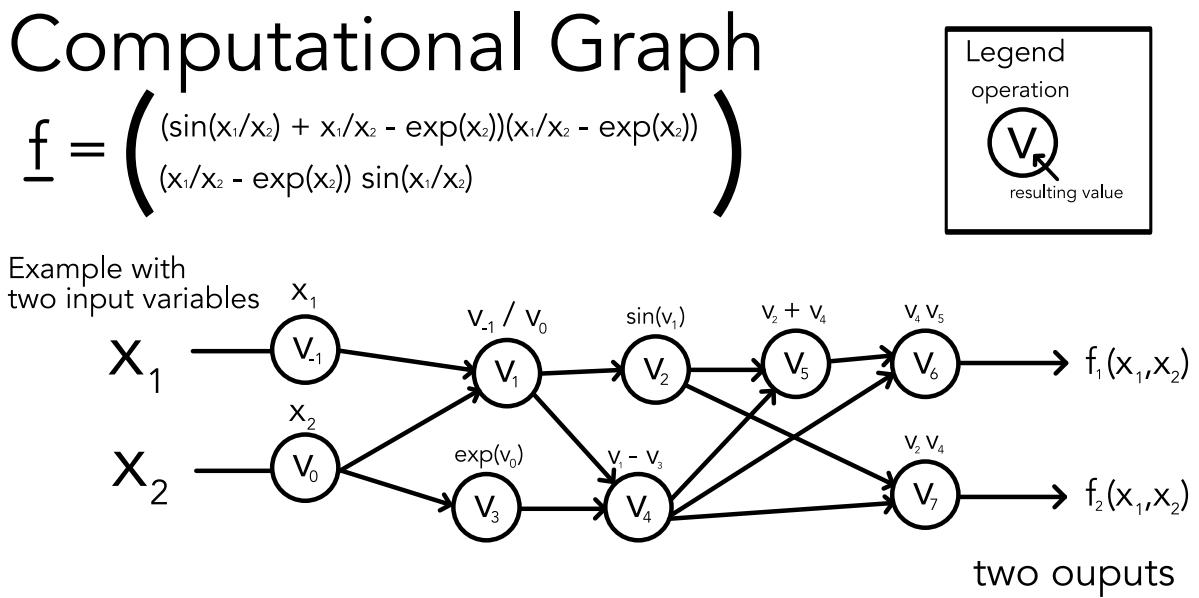


Figure 213: A computational graph.

Aim: At a given \underline{x} , we want to calculate

$$\begin{aligned} \partial_{x_1} f_1|_{\underline{x}=\underline{x}}, & \quad \partial_{x_2} f_1|_{\underline{x}=\underline{x}} \\ \partial_{x_1} f_2|_{\underline{x}=\underline{x}}, & \quad \partial_{x_2} f_2|_{\underline{x}=\underline{x}} \end{aligned} \quad (1188)$$

At given \underline{x} and *wiggling* $\xi = \begin{pmatrix} x'_1 & x'_2 \end{pmatrix}$ we can calculate the directional derivative

$$\frac{\partial f_1}{\partial x_1} \xi_1 + \frac{\partial f_1}{\partial x_2} \xi_2 \quad (1189)$$

by a forward pass through the graph, as illustrated in figure 214.

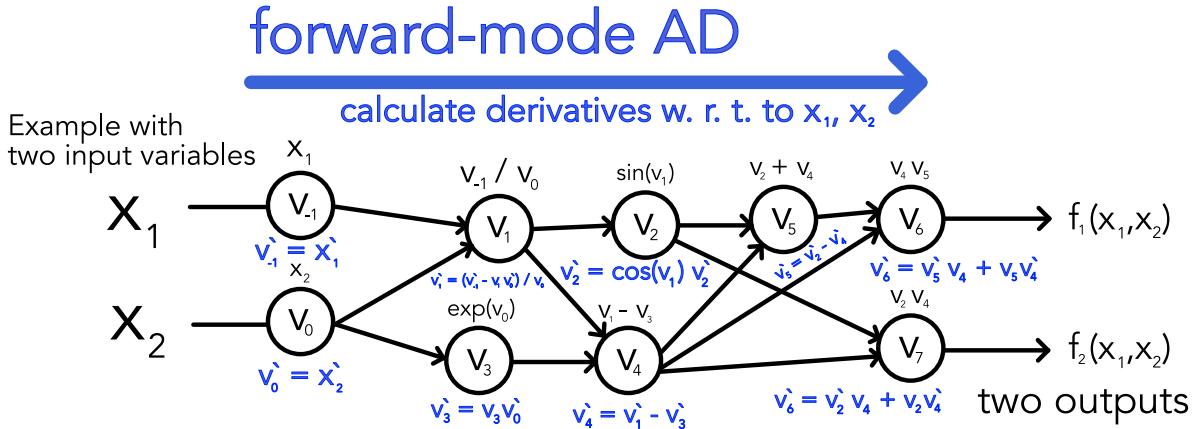


Figure 214: Forward pass through the computational graph.

For $\xi = \begin{pmatrix} 1 & 0 \end{pmatrix}$ we calculate $\partial_{x_1} f$ derivatives of both outputs with respect to a change in one input variable in one forward AD (automatic differentiation) pass.

Computationally this can just be done by passing a dual number through f

Backpropagation Here we go the other way around, as illustrated in figure 215.

22.3.3.5 Error backpropagation in a neural network

We have seen that based on the chain rule, we can efficiently calculate gradients of functions with lots of inputs and few outputs - e.g. lots of weights, which are also inputs, and one loss function.

What errors are propagating? Remember that the neurons output is given by $h_j^{(l)}(z_j^{(l)})$.

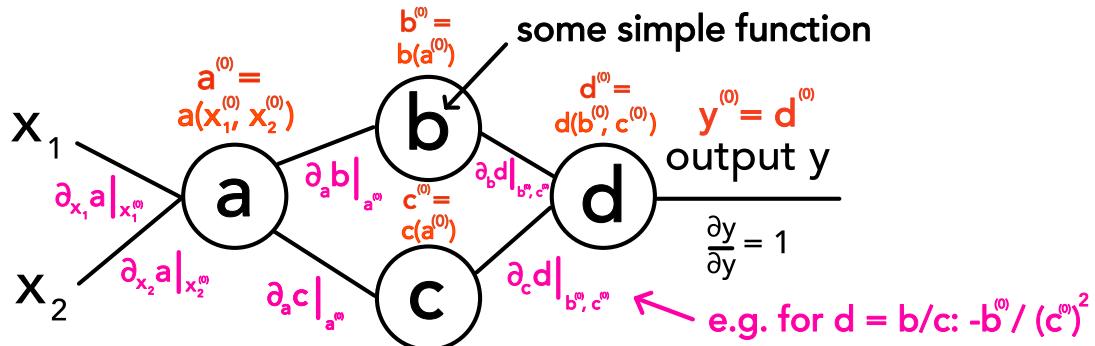
Imagine there is a little benign demon that can slightly tweak $z_j^{(l)}$.

Consider the quantity

$$\delta_j^{(l)} = \frac{\partial \mathcal{Q}(W)}{\partial z_j^{(l)}} \quad (1190)$$

Let us call $\underline{\delta}^{(l)}$ the error in the l -th layer, with $\delta_j^{(l)}$ being the error of the j -th neuron in the l -th layer.

Aim: calculate $\partial_{x_1} f$ and $\partial_{x_2} f$ at some $x^{(0)}$



1. at the given $x^{(0)}$, calculate f and the values of a, b, c, d ← forward pass
2. with the calculated values, calculate the derivatives of the nodes w.r.t. their inputs
3. From right to left you can just see the chain rule, so ← backprop.

$$\begin{aligned}\partial_{x_1} d \Big|_{x_1^{(0)}} &= \partial_{x_1} a \Big|_{x_1^{(0)}} \left(\partial_a c \Big|_{a^{(0)}} \partial_c d \Big|_{b^{(0)}, c^{(0)}} + \partial_a b \Big|_{a^{(0)}} \partial_b d \Big|_{b^{(0)}, c^{(0)}} \right) \\ \partial_{x_2} d \Big|_{x_2^{(0)}} &= \partial_{x_2} a \Big|_{x_2^{(0)}} \quad || =\end{aligned}$$

Figure 215: Backpropagation through the computational graph.

Why is $\delta_j^{(l)}$ called an error?: If $\delta_j^{(l)}$ is small our daemon can't improve the loss much by tweaking $z_j^{(l)}$ but if $\delta_j^{(l)}$ is large, there is lots of room for improvement.

The error in the output layer L is given by

$$\delta_j^{(L)} = \frac{\partial \text{loss}}{\partial z_j^{(L)}} = \frac{\partial \text{loss}}{\partial a_j^{(L)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \quad (1191)$$

Consider the quadratic loss $\text{loss} = \frac{1}{2} \sum_{j=1}^m (a_j^{(L)} - y_j)^2$ (for supervision $y \in \mathbb{R}^m$) and as typical for regression, the output activation function, different from the hidden layers, is the identity function $a_j^{(L)} = z_j^{(L)}$.

Then we have

$$\delta_j^{(L)} = a_j^{(L)} - y_j \quad (1192)$$

So in the regression setting $\delta_j^{(L)}$ is quite literally the error between the output of the neuron and the wanted output.

By the chain rule, this error is propagated backwards by (**error backpropagation**)

$$\delta_j^{(l)} = \frac{\partial \text{loss}}{\partial z_j^{(l)}} = \sum_{k \in \text{children}(v_i)} \frac{\partial \text{loss}}{\partial z_k^{(l+1)}} \frac{\partial z_k^{(l+1)}}{\partial z_j^{(l)}} = \left. \frac{\partial h_j^{(l)}}{\partial z} \right|_{z_j^{(l)}} \sum_{k \in \text{children}(v_i)} \delta_k^{(l+1)} w_{kj}^{(l+1)} \quad (1193)$$

And the **partial derivatives of interest** are given by

$$\frac{\partial \text{loss}(\underline{y}, \underline{f}(\underline{x}, \underline{W}))}{\partial w_{j,k}^{(l)}} = a_k^{(l-1)} \delta_j^{(l)} \quad (1194)$$

When does a weight learn slowly?: A weight will change only little in the gradient descent steps, if

- the input to the neuron is small, $a_k^{(l-1)}$ is small
- the output has saturated, $\delta_j^{(l)}$ is small by (see backpropagation formula) $\left. \frac{\partial h_j^{(l)}}{\partial z} \right|_{z_j^{(l)}}$ being small

Backpropagation algorithm:

1. For each training sample

- Feedforward: Compute $\underline{a}^{(l)}$ and $\underline{z}^{(l)}$ for all layers
- Output errors: calculate $\underline{\delta}^{(L)}$
- Backpropagate errors: calculate $\underline{\delta}^{(l)}$ for all layers, starting from the output layer, using the previous results on those from the feedforward pass along the way
- Compute the gradients:

$$\frac{\partial \text{loss}(\underline{y}, \underline{f}(\underline{x}, \underline{W}))}{\partial w_{j,k}^{(l)}} = a_k^{(l-1)} \delta_j^{(l)}$$

2. Update the weights based on gradient descent using the average of the gradients over the minibatch

22.3.3.6 Problem in training deep networks: Exploding and vanishing gradients

Reconsider the error-backpropagation formula

$$\delta_j^{(l)} = \frac{\partial \text{loss}}{\partial z_j^{(l)}} = \sum_{k \in \text{children}(v_i)} \frac{\partial \text{loss}}{\partial z_k^{(l+1)}} \frac{\partial z_j^{(l+1)}}{\partial z_j^{(l)}} = \left. \frac{\partial h_j^{(l)}}{\partial z} \right|_{z_j^{(l)}} \sum_{k \in \text{children}(v_i)} \delta_k^{(l+1)} w_{kj}^{(l+1)} \quad (1195)$$

Which is a multiplicative chain (with summations) through the network starting from the output layer. So, for deep networks

$$\left| \frac{\partial z_j^{(l+1)}}{\partial z_j^{(l)}} \right| \rightarrow \begin{cases} > 1 & \text{exploding gradients} \\ < 1 & \text{vanishing gradients} \end{cases} \quad (1196)$$

But we sometimes want to use deep networks as they might be more expressive (e.g. because sublayers learn simpler patterns).

ReLU against exploding and vanishing gradients: For a ReLU activation function

$$\text{ReLU for } z_j^{(l)} > 0 : \frac{\partial h_j^{(l)}}{\partial z} \Big|_{z_j^{(l)}} = 1 \quad (1197)$$

But we have a problem with vanishing gradients for $z_j^{(l)} < 0$ - so maybe use a leaky ReLU. Another alleviation might be layerwise pretraining.

22.4 Outlook

- use NN architectures reflecting the problem symmetry
- bake in physical knowledge into the loss, regularizing on a differential equation
- ...

23 Dimensionality Reduction

We are now in the land of unsupervised learning.

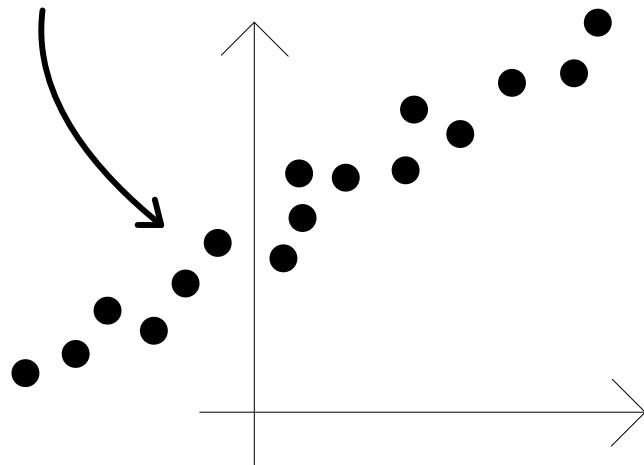
Unsupervised Learning Overview*: Given only features \underline{x} so a data matrix $\underline{\underline{X}} \in \mathbb{R}^{N \times p}$, we want to find and use structure in the data. Typical unsupervised tasks are

- clustering: find groups of similar data points, e.g. similar measurements of solar wind, similar pictures, ...
 - **mean-shift**: group measurements in feature space to modes of their density by shifting points to the mean of their neighborhood
 - **k-means**: a vector quantizer maps vectors in \mathbb{R}^p to a set of k centroids, centroids are placed, all vectors are assigned to the nearest centroid, centroids are updated to the center of mass of their assigned vectors
 - **hierarchical clustering**: start with each point as its own cluster, merge clusters (what clusters to merge e.g. based on average distance between their points), possibly restricted by connectivity constraints (e.g. by kNN graph)
- dimensionality reduction: find a lower-dimensional representation of the data
 - **PCA**: find the directions of maximal variance in the data, the eigenvectors of the covariance matrix
 - approaches retaining local distance information, e.g. **Multidimensional Scaling (MDS)**
 - **Autoencoders**: neural networks that learn to encode and decode data
 - **Independent Component Analysis (ICA)**: decomposition of a signal into independent non-Gaussian sources
- density estimation: estimate the probability density function of the data (e.g. histograms, kernel density estimation)

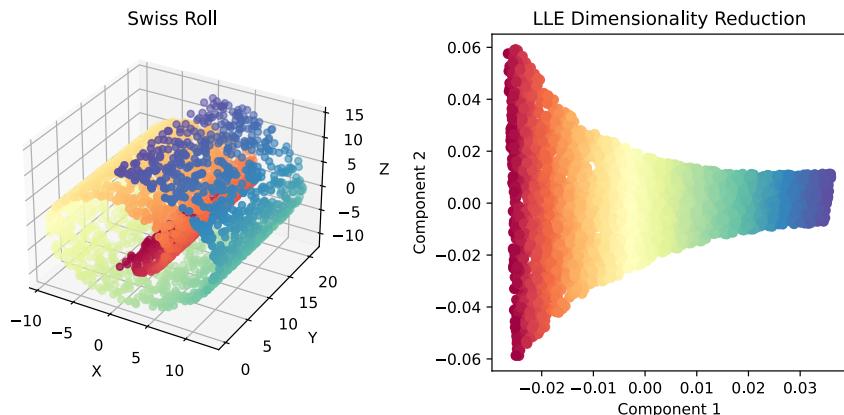
Consider we have measured data $\underline{x}_1, \dots, \underline{x}_N \in \mathbb{R}^p$ which we collect into a data matrix $\underline{\underline{X}} \in \mathbb{R}^{N \times p}$. Our measurements might not reflect the true dimensionality of our data / some dimensions might not be as informative as others. Consider for instance we capture movement by multiple cameras - each camera captures millions of pixels (each e.g. a greyscale dimension), but the intrinsic dimensionality of the movement is much lower (e.g. maximally 3D). Simple examples in 1D and 2D can be found in figure 216.

2d measurement only 1 intrinsic dimension

e.g. 1d-movement filmed by two sensors



(a) 2D data with 1 relevant dimension.



(b) 3D data with 2 relevant dimensions.

Figure 216: Simple examples of data with lower intrinsic dimensionality.

Aim: Given $\underline{\underline{X}} \in \mathbb{R}^{N \times p}$, we want to find a lower-dimensional representation $\underline{\underline{Z}} \in \mathbb{R}^{N \times q}$ with $q < p$ still capturing the (main) structure of the data. We project onto lower-dimensional **latent-manifolds**. Latent variables can only be inferred indirectly from our observed variables and might correspond to aspects of physical reality (e.g. our 1d movement captured by the cameras). In the latent space items resembling each other are closer together.^a

^aNote that e.g. the latent space of a machine learning model might be completely unintuitive.

Applications of dimensionality reduction include

- visualization of high-dimensional data

- lossy data compression
- speed-up of learning algorithms
- noise reduction

23.1 Principal Component Analysis (PCA)

In PCA we linearly transform the dataset into a new coordinate system such that most of the variation in the data is captured by the lower dimensions.

The principal components are an orthonormal basis, with the first principal component being along the direction of the most variance and so on.

An example of PCA applied to a 2D dataset is shown in figure 217.

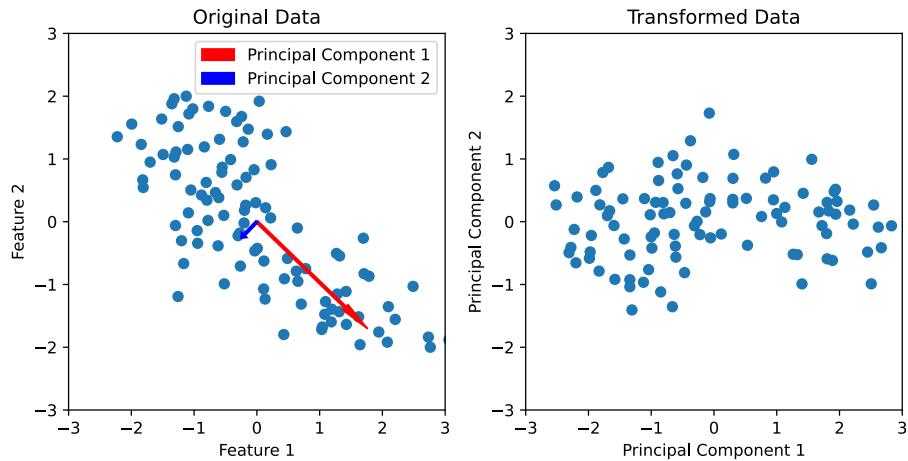


Figure 217: PCA applied to a 2D dataset.

23.1.1 Deriving the principal components from finding the direction with maximum variance

Note: One should mean-center the features before applying PCA - it is a linear transformation. We set $\underline{\underline{X}}$ to

$$\underline{\underline{X}} = \underline{\underline{X}} - \frac{1}{N} \underline{\underline{\mu}}^T, \quad \text{means of features } \underline{\underline{\mu}} \in \mathbb{R}^p \quad (1198)$$

so the covariance matrix

$$\underline{\underline{S}} = \frac{1}{N} \left(\underline{\underline{X}} - \frac{1}{N} \underline{\underline{\mu}}^T \right)^T \left(\underline{\underline{X}} - \frac{1}{N} \underline{\underline{\mu}}^T \right) = \frac{1}{N} \underline{\underline{X}}^T \underline{\underline{X}} \quad (1199)$$

The first principal component is given by

$$\underline{v}^* = \underset{\underline{v}, \|\underline{v}\|=1}{\operatorname{argmax}} \operatorname{Var}(\underline{\underline{X}}\underline{v}) \quad (1200)$$

We can formulate this as the optimization problem

$$\underline{v}^* = \underset{\underline{v}}{\operatorname{argmax}} \left\{ \underbrace{\underline{v}^T \underline{\underline{S}} \underline{v} - \lambda (\underline{v}^T \underline{v} - 1)}_{:= \mathcal{L}(\underline{v}, \lambda)} \right\} \quad (1201)$$

where λ is a Lagrange multiplier to include the condition $\|\underline{v}\| = 1$ ($\partial_\lambda \mathcal{L} = 0 \rightarrow \underline{v}^T \underline{v} = 1$).

Then

$$\partial_{\underline{v}} \mathcal{L} = 2 \underline{\underline{S}} \underline{v} - 2 \lambda \underline{v} = 0 \rightarrow \underline{\underline{S}} \underline{v} = \lambda \underline{v} \quad (1202)$$

where multiplying by \underline{v}^T from the left and using $\underline{v}^T \underline{v} = 1$ gives

$$\underline{v}^T \underline{\underline{S}} \underline{v} = \operatorname{Var}(\underline{\underline{X}}\underline{v}) = \lambda \quad (1203)$$

So the direction of maximum variance is the eigenvector with the largest eigenvalues λ_{map} .

Note: An alternative perspective is finding the directions to which all points in the dataset have the lowest squared distance (different from Linear Regression as here an orthogonal distance is minimized).

23.1.1.1 The principal components and dimensionality reduction

The principal components are the eigenvectors of the covariance Matrix (here assuming mean-centered data)

$$\underline{\underline{S}} = \frac{1}{N} \underline{\underline{X}}^T \underline{\underline{X}} \text{ symmetric, real, positive semi-definite} (\underline{u}^T \underline{\underline{S}} \underline{u} \geq 0 \forall \underline{u}) \quad (1204)$$

so S has only real non-negative eigenvalues and the eigenvectors form an orthonormal basis (symmetric matrices are diagonalizable, EVs of real symmetric matrices are orthogonal) - our principal components.

PCA rotates the data so that it becomes variance-aligned.

We project to lower dimensions by projecting our data onto the principal components.

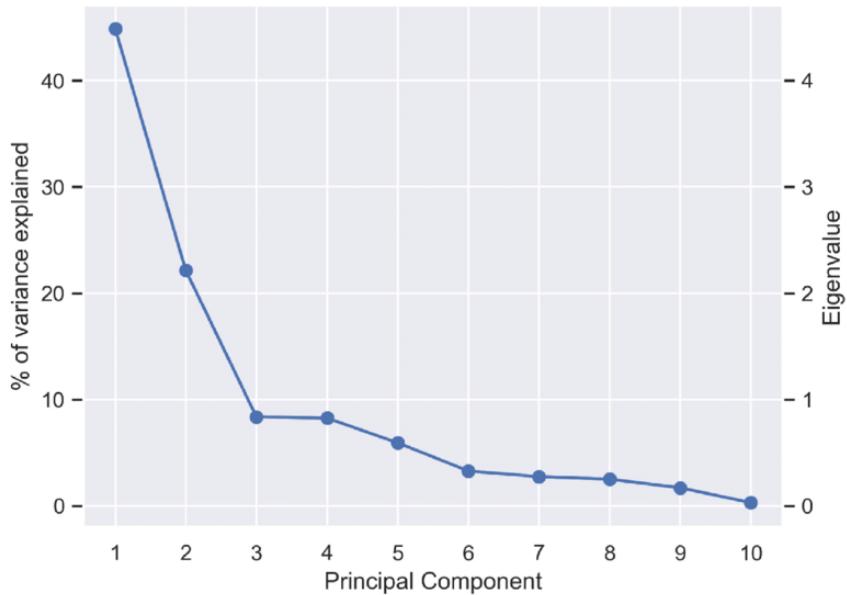


Figure 218: Explained variance of the principal components (scree plot).

How many principal components to keep? One can look at the explained variance via the eigenvalues and plot them, as illustrated in figure 218.

23.2 Nonlinear Dimensionality Reduction I: Nonlinear PCA

Problem: PCA is a linear transformation, it will not be able to reduce non-linear structures to lower dimensions, as illustrated in figure 219.

Idea: Map the data to a higher dimensional space, where the structure might become reducible to lower dimensions by a linear transformation, see figure 220.

As before we do a transformation to a higher dimensional space by

$$\underline{\phi} : \mathbb{R}^p \rightarrow \mathbb{R}^q, \quad q > p \quad (1205)$$

where our aim is not to directly use such a transformation, but to replace the scalar product in this high-dimensional space by a kernel function

$$k(\underline{x}_i, \underline{x}_j) = \underline{\phi}(\underline{x}_i)^T \underline{\phi}(\underline{x}_j) \quad (1206)$$

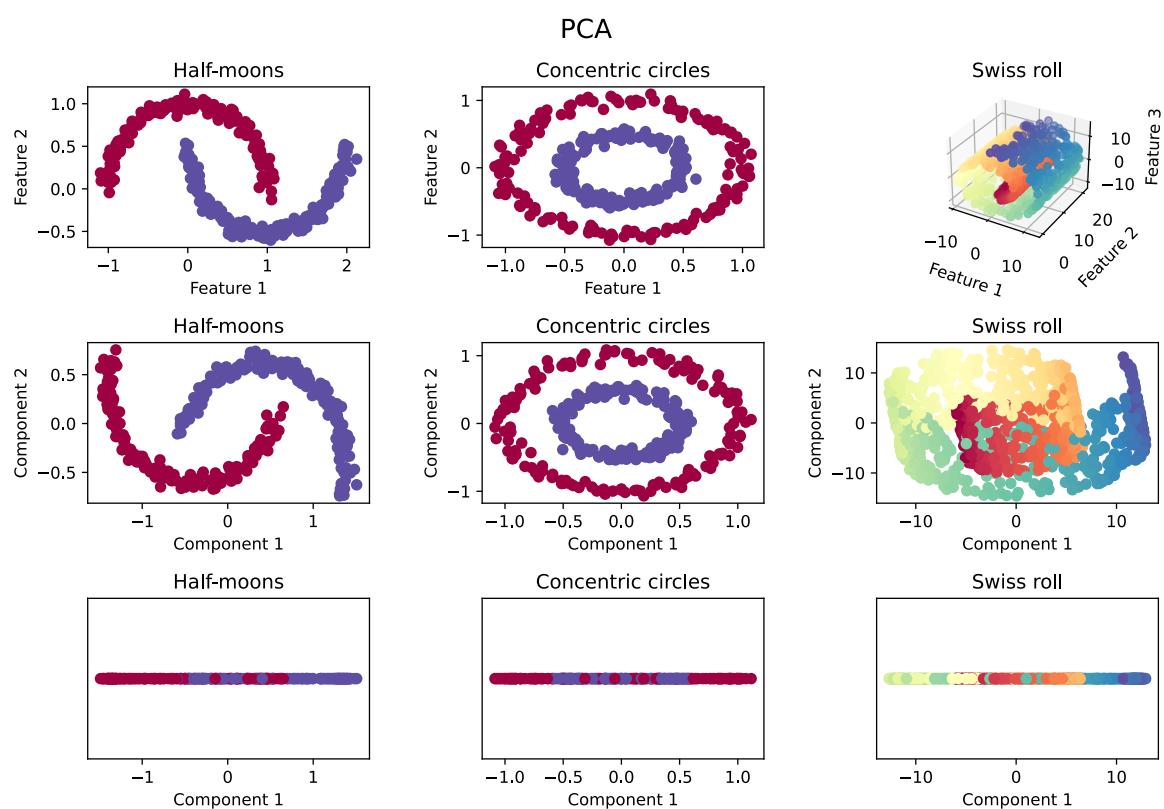


Figure 219: PCA applied to data with non-linear structure.

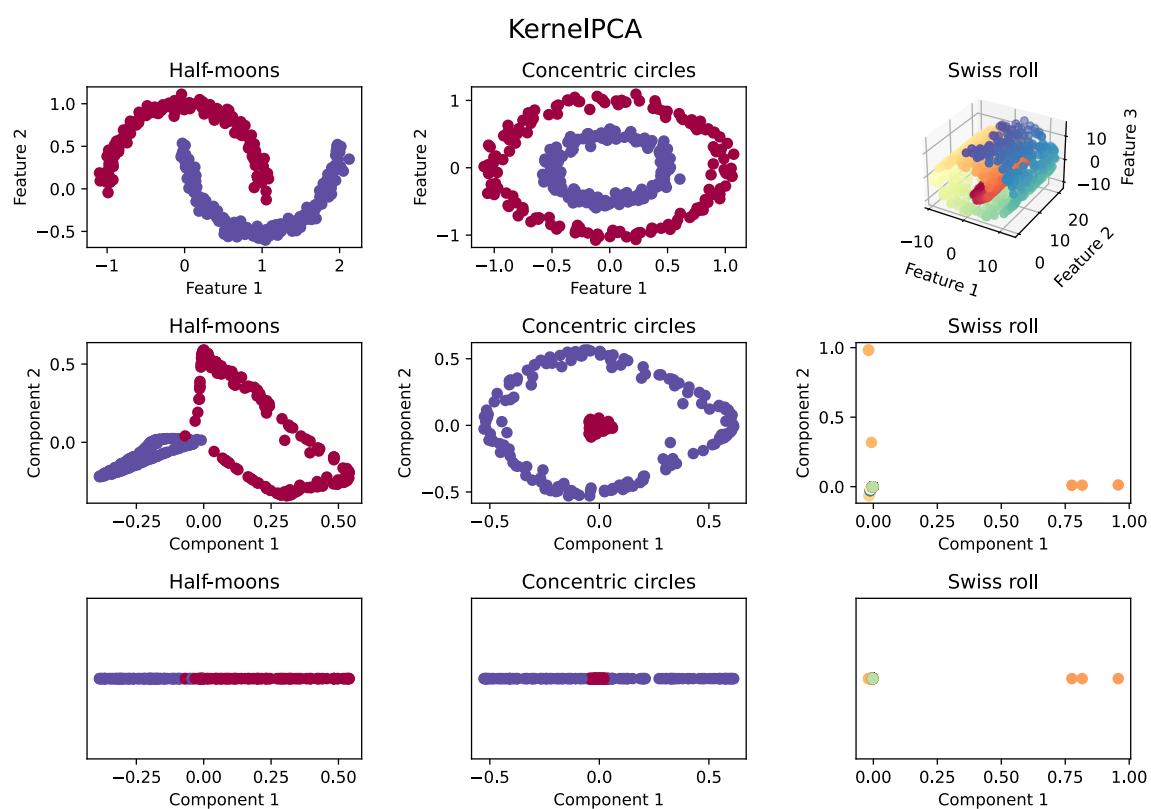


Figure 220: Nonlinear PCA (Kernel PCA) applied to data with non-linear structure.

23.2.1 Kernel PCA

Our principal components in the high-dimensional space are the eigenvectors of the covariance matrix

$$\underline{\underline{S}} = \frac{1}{N} \left(\underline{\underline{\Phi}} - \frac{1}{N \times 1} \underline{\underline{\bar{\Phi}}}^T \right)^T \left(\underline{\underline{\Phi}} - \frac{1}{N \times 1} \underline{\underline{\bar{\Phi}}}^T \right) \in \mathbb{R}^{q \times q} \quad (1207)$$

Note: While $\underline{\underline{X}}$ might have centered features, we still need to center the features in $\underline{\underline{\Phi}}$. We will later do this in the kernel, for now assume $\underline{\underline{\tilde{\Phi}}}$ to have centered features.

23.2.1.1 Finding a formulation where the projection to the principal components is only expressed in terms of a kernel function

Problem: The covariance matrix is $q \times q$ but we want to go very high dimensional, and this only implicitly by the kernel function. The eigenvectors will also be $\in \mathbb{R}^q$ - we will never explicitly be able to compute them. We never explicitly operate in the high-dimensional space.

Idea: What if we could instead somehow use the eigenvalues and -vectors of the Gram matrix $\underline{\underline{\tilde{\Phi}}}^T \underline{\underline{\tilde{\Phi}}} \in \mathbb{R}^{N \times N}$ to formulate the projection onto the principal components?

Consider the eigenvectors of the Gram matrix

$$\frac{1}{N} \underline{\underline{\tilde{\Phi}}}^T \underline{\underline{\tilde{\Phi}}} \underline{\underline{u}} = \lambda \underline{\underline{u}}, \quad \underline{\underline{u}} \in \mathbb{R}^N \quad (1208)$$

and multiply from the left with $\underline{\underline{\tilde{\Phi}}}^T$

$$\frac{1}{N} \underline{\underline{\tilde{\Phi}}}^T \underline{\underline{\tilde{\Phi}}} \left(\underline{\underline{\tilde{\Phi}}}^T \underline{\underline{u}} \right) = \lambda \left(\underline{\underline{\tilde{\Phi}}}^T \underline{\underline{u}} \right) \quad (1209)$$

Then we see that $\underline{\underline{v}} = \underline{\underline{\tilde{\Phi}}}^T \underline{\underline{u}}$ is an eigenvector of the covariance matrix to the same eigenvalue (not of unit length though). We can find the normalization by

$$\|\underline{\underline{v}}\|^2 = \underline{\underline{v}}^T \underline{\underline{v}} = \underline{\underline{u}}^T \underline{\underline{\tilde{\Phi}}}^T \underline{\underline{\tilde{\Phi}}} \underline{\underline{u}} = N \lambda \|\underline{\underline{u}}\|^2 \quad (1210)$$

so if $\underline{\underline{u}}$ is a normalized eigenvector of the Gram matrix $\underline{\underline{G}} = \frac{1}{N} \underline{\underline{\tilde{\Phi}}}^T \underline{\underline{\tilde{\Phi}}}$

$$\underline{\underline{v}} = \frac{1}{\sqrt{N \lambda}} \underline{\underline{\tilde{\Phi}}}^T \underline{\underline{u}} = \frac{1}{\sqrt{N \lambda}} \sum_{i=1}^N u_i \phi_{\underline{x}_i} \quad (1211)$$

is a normalized eigenvector of the covariance matrix $\underline{\underline{S}} = \frac{1}{N} \underline{\underline{\tilde{\Phi}}}^T \underline{\underline{\tilde{\Phi}}}$.

So given a vector $\underline{x}_j \in \mathbb{R}^p$ and its high dimensional representation $\underline{\phi}_{\underline{x}_j} \in \mathbb{R}^q$ (which will not be necessary explicitly), we get the projection on a principal component

$$\underline{\phi}_{\underline{x}_j}^T v = \frac{1}{\sqrt{N\lambda}} \sum_{i=1}^N u_i \underline{\phi}_{\underline{x}_j}^T \underline{\phi}_{\underline{x}_i} = \frac{1}{\sqrt{N\lambda}} \sum_{i=1}^N u_i \tilde{k}(\underline{x}_i, \underline{x}_j) \quad (1212)$$

where $u \in \mathbb{R}^N$ is the eigenvector of the Gram matrix $\underline{\underline{G}} = \frac{1}{N} \underline{\underline{\tilde{\Phi}}} \underline{\underline{\tilde{\Phi}}}^T \in \mathbb{R}^{N \times N}$ with

$$G_{ij} = \frac{1}{N} \tilde{k}(\underline{x}_i, \underline{x}_j) \quad (1213)$$

to the eigenvalue λ (largest eigenvalue for first principal component).

We therefore have a formulation of the projection to the principal components of a kernelized PCA only in terms of kernel expressions.

23.2.1.2 Kernel expression for mean centered projected features

What's now left to do is to find an expression for

$$\tilde{k}(\underline{x}_i, \underline{x}_j) = \underline{\phi}_{\underline{x}_i}^T \underline{\phi}_{\underline{x}_j} \quad (1214)$$

for the mean centered features

$$\underline{\phi}_{\underline{x}_i} = \underline{\phi}_{\underline{x}_i} - \frac{1}{N} \sum_{k=1}^N \underline{\phi}_{\underline{x}_k} \quad (1215)$$

so

$$\begin{aligned}
\tilde{k}(\underline{x}_i, \underline{x}_j) &= \underbrace{\tilde{\phi}_{\underline{x}_i}^T \tilde{\phi}_{\underline{x}_j}}_{:=k(\underline{x}_i, \underline{x}_j)} \\
&= \underbrace{\phi_{\underline{x}_i}^T \phi_{\underline{x}_j}}_{:=k(\underline{x}_i, \underline{x}_j)} \\
&\quad - \frac{1}{N} \sum_{k=1}^N \underbrace{\phi_{\underline{x}_i}^T \phi_{\underline{x}_k}}_{:=k(\underline{x}_i, \underline{x}_k)} \\
&\quad - \frac{1}{N} \sum_{k=1}^N \underbrace{\phi_{\underline{x}_k}^T \phi_{\underline{x}_j}}_{:=k(\underline{x}_k, \underline{x}_j)} \\
&\quad + \frac{1}{N^2} \sum_{k=1}^N \sum_{l=1}^N \underbrace{\phi_{\underline{x}_k}^T \phi_{\underline{x}_l}}_{:=k(\underline{x}_k, \underline{x}_l)}
\end{aligned} \tag{1216}$$

so we have found an expression just in terms of the kernel function, e.g. with the RBF (radial basis function) kernel

$$k(\underline{x}_i, \underline{x}_j) = \exp\left(-\frac{\|\underline{x}_i - \underline{x}_j\|^2}{2\sigma^2}\right), \quad \text{free parameter } \sigma \tag{1217}$$

23.3 Nonlinear Dimensionality Reduction II: Further techniques

23.3.1 Linear discriminant analysis for dimensionality reduction of classified data

Here we find directions better separating groups. One approach is to calculate the in-between-class and withing-class scatter matrices for all classes and use the eigenvectors of them, sorted by the eigenvalues as *principal components*, akin to PCA.

23.3.2 Retaining (local) distance information or neighbor structure

Consider high dimensional samples $\underline{x}_i \in \mathbb{R}^{p^*}$ with distances

$$d_{ij}^* = \|\underline{x}_i - \underline{x}_j\|^2 \tag{1218}$$

Aim: Find a low dimensional embedding $\underline{z}_i \in \mathbb{R}^p$ with $p < p^*$ such that with distances $d_{ij} = \|\underline{z}_i - \underline{z}_j\|^2$ that distance information is preserved.

23.3.2.1 Global preservation of distance information

In this approach, we choose the set $\{z_i\}$ such that

$$\{z_i\} = \operatorname{argmin}_{\{z_i\}} \sum_i \sum_j M(d_{ij}, d_{ij}^*) \quad (1219)$$

with M being a metric comparing distances, for instance

$$M_{\text{MDS,Kruskal}} = (d_{ij} - d_{ij}^*)^2, \quad M_{\text{MDS,Sammon}} = \frac{(d_{ij} - d_{ij}^*)^2}{d_{ij}^*} \quad (1220)$$

where **MDS** stands for **Multidimensional Scaling** - a specific dimensionality reduction technique.

Note: Kernel PCA with Gaussian kernel and MDS with Kruskal metric are equivalent.

$$\sum_i \sum_j (d_{ij} - d_{ij}^*)^2 \quad (1221)$$

is minimized when the data matrix is projected onto the kernelized principal components.

23.3.2.2 Local preservation of distance information

We follow the steps

1. Construct a symmetric k-nearest-neighbor-graph (KNNG)

Intermezzo: K-nearest-neighbor-graph: In the KNNG \underline{x}_i is connected to \underline{x}_j if \underline{x}_j is among the k nearest neighbors of \underline{x}_i among all points in the dataset. Note that while \underline{x}_j might be in the k-nearest neighbor list of \underline{x}_i , \underline{x}_i might not be in the k-nearest neighbor list of \underline{x}_j . In a symmetric KNNG, we ignore this directionality, two points are connected if one is in the nearest neighbor list of the other.

2. We phrase the preservation of local structure as a potential problem
 - (a) if \underline{x}_i and \underline{x}_j are symmetric neighbors, an attractive potential is assigned to \underline{z}_i and \underline{z}_j
 - (b) a weak repulsion is assigned to all pairs \underline{z}_i and \underline{z}_j ($\mathcal{O}(N^2)$ repulsion terms)
3. Minimize the potential starting from e.g. random \underline{z}_i

Based on different potentials, different techniques are derived, as illustrated in figure 221.

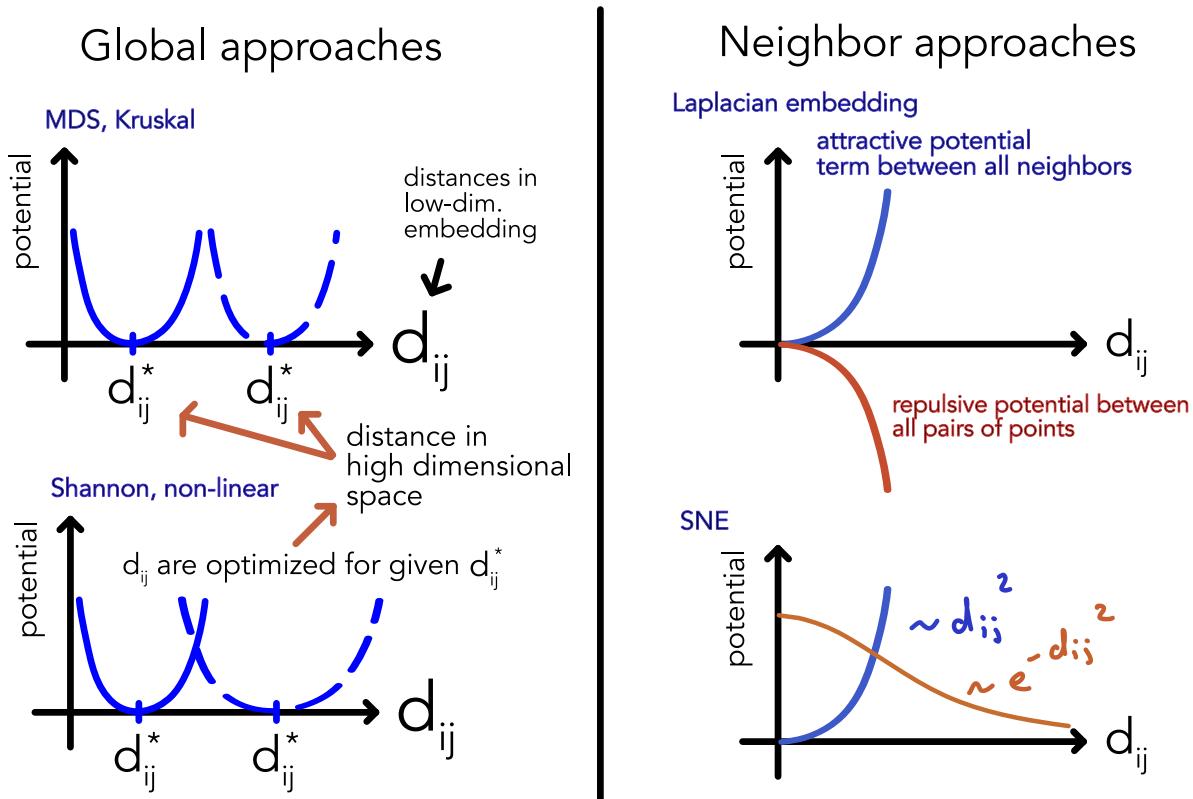


Figure 221: Different approaches to retain distance information.

23.3.2.3 Isomap - distance preservation along geodesics

Problem of MDS: Euclidian distances in the original space might not best represent the true distances in the data.

Idea: Preserve distance information along the geodesics of the data, approximated by shortest paths between points on the K-nearest-neighbor-graph (\rightarrow Djikstra algorithm). Then apply MDS to the geodesic distances.

This is illustrated in figure 222.

23.3.2.4 Locally Linear Embedding (LLE)

1. A set of nearest neighbors is found as each point
2. A set of weights is computed that describes the point as a linear combination of its neighbors
3. a low dimensional embedding is found such that a point is still described with the same weighted linear combination of its neighbors

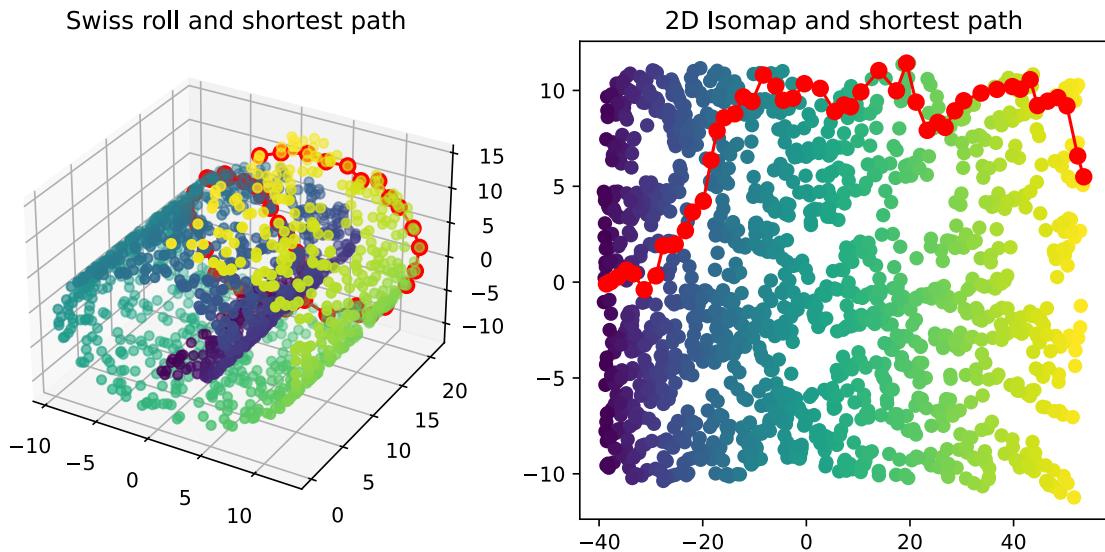


Figure 222: Isomap applied to data with non-linear structure.

23.3.2.5 t-SNE (t-distributed stochastic neighbor embedding)

- construct a probability distribution over pairs of the high-dimensional data points, such that similar objects have higher probabilities
- a similar low dimensional probability distribution is constructed and the Kullback-Leibler divergence⁴³ (relative entropy) between the two distributions is minimized

23.3.3 Autoencoders

Idea: Train a neural network to compress and decompress the data - extraction at the compressed layer is the low-dimensional representation.

An autoencoder is illustrated in figure 223.

The loss used in training is

$$\mathcal{L} = \sum_i^N \|\underline{x}_i - \tilde{\underline{x}}_i\|^2 \quad (1223)$$

⁴³

$$D_{\text{KL}}(P\|Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \quad (1222)$$

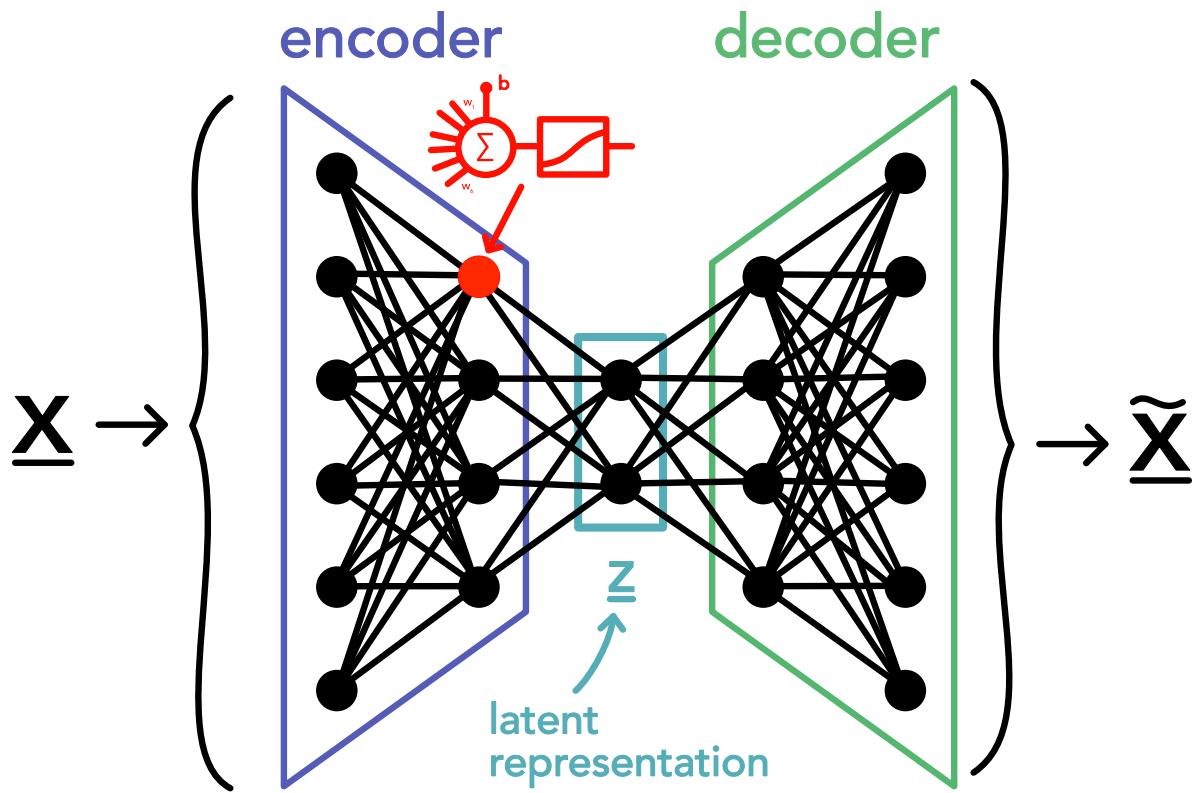


Figure 223: Autoencoder.

24 Latent Variable Models

From measurements, we want to find the underlying, latent signals, so rather than finding a model relating multiple observed variables, we want to find underlying unobserved variables linked to our observation.

For instance

- we have measured neural activity by 100 electrodes on the scalp and want to find the underlying neural signals
- we have multiple stock prices and many points in time (N measurements) and want to find the underlying factors that drive the stock prices

24.1 The Cocktail Party Problem - Blind Source Separation

Consider three instruments playing at the center of the room, one plays a sine signal

$$\{S_{i1}\}_{i=1}^N = \{\sin 2t_i\}_{i=1}^N, \quad t = 0.000 : 0.004 : 8.000 \quad (1224)$$

another a square signal

$$\{S_{i2}\}_{i=1}^N = \{\text{square}(2t_1)\}_{i=1}^N, \quad t = 0.000 : 0.004 : 8.000 \quad (1225)$$

and the third a sawtooth signal

$$\{S_{i2}\}_{i=1}^N = \{\text{sawtooth}(2t_1)\}_{i=1}^N, \quad t = 0.000 : 0.004 : 8.000 \quad (1226)$$

We measure this signal by three microphones at different positions in the room. They measure different linearly mixed signals

$$\begin{aligned} X_{i1} &= S_{i1} + S_{i1} + S_{i1} + \epsilon_{i1} \\ X_{i2} &= 0.5S_{i2} + 2S_{i2} + S_{i2} + \epsilon_{i2} \\ X_{i3} &= 1.5S_{i3} + S_{i3} + 2c_3S_{i3} + \epsilon_{i3} \end{aligned} \quad (1227)$$

where the ϵ_{i1} are the noise terms and we have inserted an arbitrary mixture.

Note: We assume the same mixture at all times, time information is not used. We just consider N samples of the signals. So our transformation is really a matrix transformation in $p \times p$ space (here $p = 3$).

In matrix form, we have

$$\underline{\underline{X}} = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \underline{\underline{\Gamma}} \underline{\underline{S}} + \underline{\underline{\epsilon}} \quad (1228)$$

holding for all N samples. For all samples we can write

$$\underline{\underline{X}} = \underline{\underline{S}} \underline{\underline{\Gamma}} + \underline{\underline{E}}, \quad \underline{\underline{X}}, \underline{\underline{S}} \in \mathbb{R}^{N \times p}, \quad \underline{\underline{\Gamma}} \in \mathbb{R}^{p \times p} \quad (1229)$$

here with $p = 3$. The original signals of the instruments and the microphone recordings are shown in figure 224.

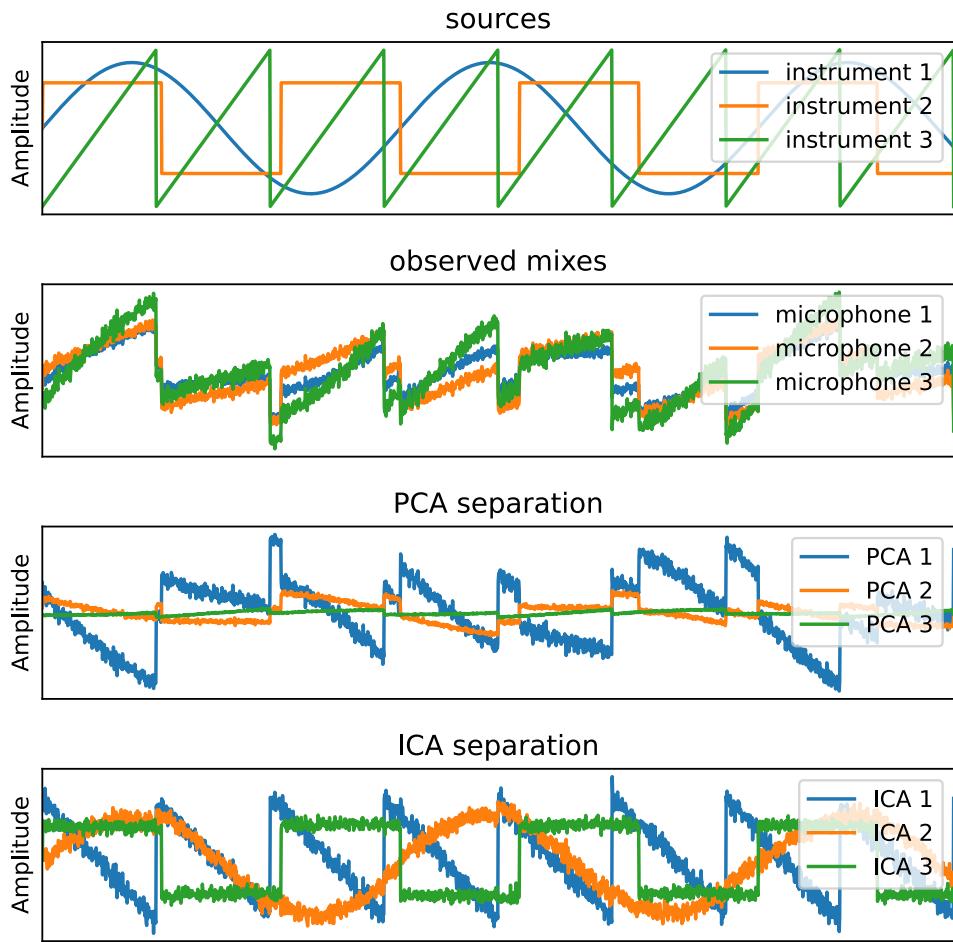


Figure 224: Mixture and separation of signal. Note the space where we operate is really p -dimensional space (here $p = 3$), so the number of original signals and number of separations. **The order / time data is not used.** We might try to separate the signals by doing PCA in this p -dimensional space (leading to p separated signals).

Aim: Given p perspectives (p microphones) on a mixed signal, we want to find p (generally $q \leq p$) original **latent** (not directly observed) source signals.

24.2 Independent Component Analysis (ICA)

24.2.1 Intuition

From mixes we want to find original not directly observable (latent) signals.

Consider two uniformly distributed source signals S_1, S_2 and the mixes

$$\begin{aligned} X_1 &= S_1 + \frac{1}{2}S_2 \\ X_2 &= S_1 - \frac{1}{2}3S_2 \end{aligned} \tag{1230}$$

This is illustrated in figure 225.

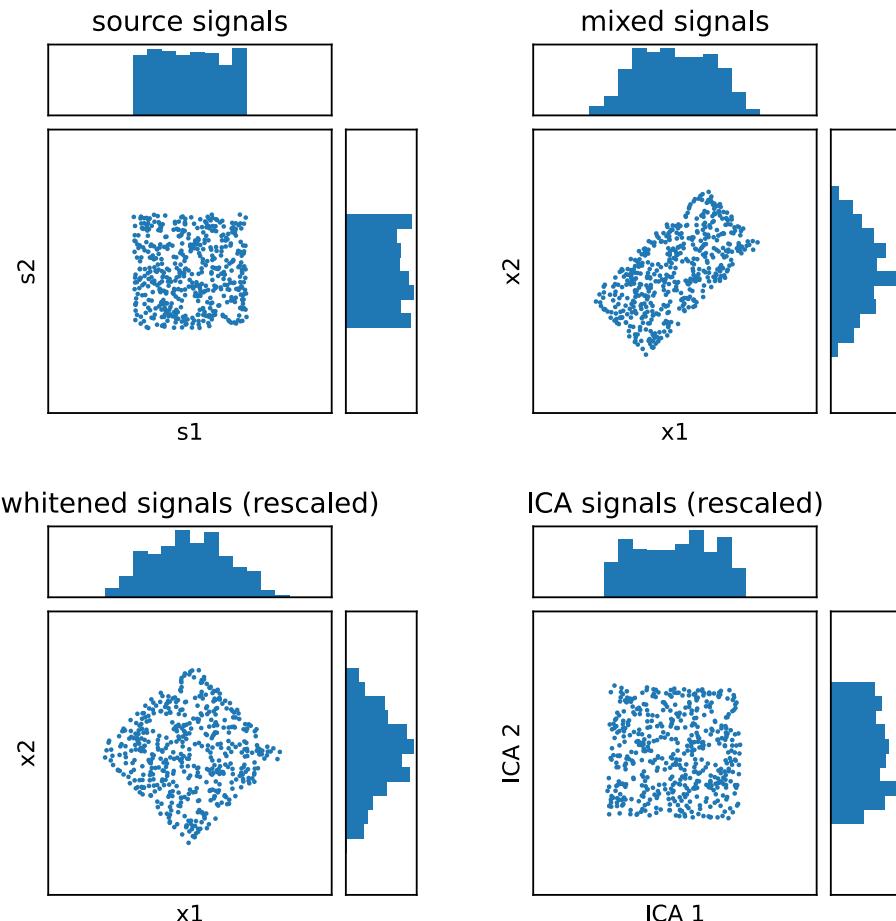


Figure 225: Illustration of ICA.

Rough idea: Loosely speaking when we add two signals, the mixture becomes more Gaussian, so we might try to find sources by finding a transformation that makes the result as non-Gaussian as possible, or relatedly maximize their independence.

Note: We first whiten the data, i.e. remove any correlations, the different channels in \underline{X} are forced to be uncorrelated. Here the intuition is that after whitening, ICA only needs to rotate the data-matrix for a high independence of the sources.

24.2.2 Formalization and aim

Different from before let $\underline{\underline{X}} \in \mathbb{R}^{p \times N}$ be the matrix with p features and N samples, as it makes the notation here a bit nicer. We assume $\underline{\underline{X}}$ to have centered and normalized features.

We assume the model

$$\begin{aligned} \frac{\underline{\underline{X}}}{p \times N} &= \frac{\underline{\Gamma}}{p \times q} \frac{\underline{\underline{S}}}{q \times N}, \quad q \leq p \\ \text{latent sources } \underline{\underline{S}}, \text{ linear mixing matrix } \underline{\underline{\Gamma}}, \quad \text{observed signals } \underline{\underline{X}} \end{aligned} \tag{1231}$$

so without a noise term⁴⁴.

In independent component analysis (ICA), the S_i are assumed to be statistically independent (not just uncorrelated) and non-gaussian.⁴⁵

Aim: Find the mixing matrix $\underline{\underline{\Gamma}}$ and the latent sources $\underline{\underline{S}}$ from the observed signals $\underline{\underline{X}}$, such that the sources are as independent as possible, $\gg p(\underline{\underline{S}}) = p(S_1) \cdot p(S_2) \cdots \cdots p(S_q) \ll$.

24.2.3 Pre-step: Whitening of the observed data

Consider the singular-value decomposition $\underline{\underline{X}} = \underline{\underline{U}} \underline{\underline{D}} \underline{\underline{V}}^T$ ($\underline{\underline{U}} \in \mathbb{R}^{p \times p}$). Since the rows of $\underline{\underline{U}}$ and $\underline{\underline{V}}$ are orthonormal vectors, $\underline{\underline{U}} \underline{\underline{V}}^T$ will be white ($(\underline{\underline{U}} \underline{\underline{V}}^T)(\underline{\underline{U}} \underline{\underline{V}}^T)^T \underline{\underline{U}} (\underline{\underline{V}}^T \underline{\underline{V}}) \underline{\underline{U}}^T = \underline{\underline{I}}$). Which we use as our $\underline{\underline{X}}$ henceforth.

Since $\underline{\underline{X}}$ is whitened to $\text{Cov}(\underline{\underline{X}}) = \underline{\underline{I}}$ (so for centered features $\text{Cov}(\underline{\underline{X}}) = \frac{1}{N} \underline{\underline{X}} \underline{\underline{X}}^T = \underline{\underline{I}}$, so $\underline{\underline{X}}$ orthogonal) and we also request covariance of the sources to be $\text{Cov}(\underline{\underline{S}}) = \underline{\underline{I}}$ (a weaker condition than independence), $\underline{\underline{\Gamma}}$ must be orthogonal, so $\underline{\underline{\Gamma}}^{-1} = \underline{\underline{\Gamma}}^T$.

Reformulated aim: Find an orthogonal $\underline{\underline{\Gamma}}$ such that the components of the vector random variable $\underline{\underline{S}} = \underline{\underline{\Gamma}}^T \underline{\underline{X}}$ are independent (and non-Gaussian).

⁴⁴There is also noisy ICA.

⁴⁵Independent Gaussian components can only be determined up to a rotation.

Why does requesting no correlation does not give unique sources?: Assume we would only request that the sources are uncorrelated, so $\text{Cov}(\underline{S}) = \underline{\underline{1}}$. As discussed, no correlation can be obtained by whitening. However, consider

$$\underline{X} = \underline{\underline{\Gamma}} \underline{S} = \underline{\underline{\Gamma}} \underline{R} \underline{R}^T \underline{S} = (\underline{\underline{\Gamma}} \underline{R})(\underline{\underline{R}}^T \underline{S}) = \underline{\underline{\Gamma}}^* \underline{S}^* \quad (1232)$$

then $\text{Cov}(\underline{S}^*) = \underline{\underline{R}} \text{Cov}(\underline{S}) \underline{\underline{R}}^T = \underline{\underline{R}} \underline{\underline{1}} \underline{\underline{R}}^T = \underline{\underline{1}}$, so there are many such compositions and it is impossible to recover unique underlying sources.

Why do we exclude Gaussian sources? Any Gaussian independent components can only be determined up to a rotation, so we would have the same non-uniqueness problem as with uncorrelated sources.

24.2.4 Minimizing dependence of the sources

Dependence of random variables can be measured by the mutual information (with $p(S_i)$ the marginal densities of the source-components)

$$\begin{aligned} \text{MI}(\underline{S}) &= \underbrace{D_{KL}\left(p(\underline{S}) \parallel \prod_{i=1}^q p(S_i)\right)}_{\text{Kullback-Leibler divergence}} \\ &\quad \text{aka Relative Entropy} \\ &= \int p(\underline{S}) \log \frac{p(\underline{S})}{\prod_{i=1}^q p(S_i)} d\underline{S} \\ &= \sum_{i=1}^q H(S_i) - H(\underline{S}) \end{aligned} \quad (1233)$$

where we used the expression for the entropy

$$H(\underline{S}) = - \int p(\underline{S}) \log p(\underline{S}) d\underline{S} \quad (1234)$$

Assuming $\text{Cov}(\underline{S}) = \underline{\underline{1}}$ without loss of generality (can be enforced by choice of $\underline{\underline{\Gamma}}$), and with $\text{Cov}(\underline{X}) = \underline{\underline{1}}$, we have $\underline{S} = \underline{\underline{\Gamma}}^T \underline{X} \in \mathbb{R}^q$, from which one can show $H(\underline{S}) = H(\underline{X}) + \log |\det \underline{\underline{\Gamma}}| = H(\underline{X})$, so

$$\text{MI}(\underline{S}) = \sum_{i=1}^q H(S_i) - H(\underline{X}) \quad (1235)$$

so

$$\hat{\underline{\Gamma}} = \underset{\underline{\Gamma} = \text{const.}}{\operatorname{argmin}} \sum_{i=1}^q H((\hat{\underline{\Gamma}}^T \underline{X})_i) - \underbrace{H(\underline{X})}_{=} = \underset{\underline{\Gamma} = \text{const.}}{\operatorname{argmin}} \sum_{i=1}^q H((\hat{\underline{\Gamma}}^T \underline{X})_i) \quad (1236)$$

Note: This is equivalent to minimizing the sum of the entropies of the separate components of \underline{S} , which is equivalent to maximizing their non-Gaussianity (e.g. Wasserschein distance to a Gaussian).

Problem: We want to calculate

$$H(S_i) = H((\hat{\underline{\Gamma}}^T \underline{X})_i) \quad (1237)$$

but given the $\underline{\Gamma}$ we only have N samples $S_i^{(1)}, S_i^{(2)}, \dots, S_i^{(N)}$ of the i -th source, not a probability distribution $p(S_i)$.

Idea: By estimating the probability distribution of the sources, we can estimate the entropy, e.g. by

$$\hat{H}_{\text{RADICAL}}(S_i) = \frac{1}{N-m} \sum_{k=1}^{N-m} \log \left(\frac{N+1}{m} \left(S_i^{(k+m)} - S_i^{(k)} \right) \right) \quad (1238)$$

with a small integer m (a parameter) (e.g. $m = 1$). This is called ICA with spacings estimates of entropy, see Learned-Miller and III, 2003.

Note: ICA cannot recover

- the actual number of source signals
- a uniquely correct ordering of the source signals
- the proper scaling (including sign) of the source signals
- ICA can only extract sources that were mixed linearly

Note: While it might look like it, ICA is not an inverse problem in the classical sense because there is no forward model to invert.

24.3 Factor Analysis

24.3.1 Going deeper than relating observed data - in search of the unobserved

Consider we have observed temperatures y_i and climate factors x_{ij} , which we relate by a linear model

$$y_i = \sum_{j=1}^p \beta_j x_{ij} + \epsilon_i \quad (1239)$$

in which the β_j can be determined by linear regression as usual. A step deeper we can ask ourselves what well separated unknown sources \underline{S} can have given rise to \underline{x} and y , e.g. industrial policies, etc.

24.3.2 Setting of Factor Analysis

Consider we have observed p features $\underline{x} \in \mathbb{R}^p$.

Idea: What if there are common latent sources S_1, \dots, S_q ($\underline{S} \in \mathbb{R}^q$) of the variation amongst the features?

For instance a set stock prices (e.g. all from the DAX) might be driven by a few common factors like central bank policies etc. which we want to find. Or from a set of tests in school, we might want to find common factors like drive, diligence, etc.

$$\underline{x}_i = \underline{\mu}_i + \underline{\Gamma} \underline{S}_i^T + \underline{\epsilon}_i, \quad i = 1, \dots, N \quad (1240)$$

observations \underline{x}_i , latent sources \underline{S}_i , loading matrix $\underline{\Gamma}$, mean $\underline{\mu}_i$, noise $\underline{\epsilon}_i$

The latent variables S_1, \dots, S_q account for the common sources of variance and their correlation in the x_1, \dots, x_p , while the ϵ_j account for the remaining variance.

Typically S_j and ϵ_j are assumed to be Gaussian and uncorrelated.

We assume

$$\underline{S}_i \sim \mathcal{N}(0, \underline{\underline{1}}), \quad \underline{\epsilon}_i \sim \mathcal{N}(0, \underline{\underline{\Psi}}) \quad (1241)$$

Note that for the S_j to be jointly normal and uncorrelated ($\text{Cov}(X, Y) = E[XY] - E[X]E[Y] = 0 \Leftrightarrow E[XY] = E[X]E[Y]$), means they are independent ($p(x, y) = p(x)p(y)$)).

Link to generative latent models: Given the assumption $\underline{S}_i \sim \mathcal{N}(0, \underline{\underline{1}})$, in this space we can easily generate new samples of \underline{S}_i by drawing from a normal distribution and given we have found the loading matrix $\underline{\Gamma}$, we can generate new samples of \underline{x}_i as $\underline{x} = \underline{\mu} + \underline{\Gamma}\underline{S} + \underline{\epsilon}_i$.

Problem: Remember the discussed identifiability issue for the case of independent Gaussian sources - the user can search for rotated versions of the factors that are more easily interpretable.

Difference to PCA: Different from PCA, we also have the noise term $\underline{\epsilon}_i$ in the model.

24.3.3 Deriving the parameters in Factor Analysis by Maximum Likelihood

We want to find the parameters $\underline{\mu}_i, \underline{\Gamma}, \underline{\Psi}$ by maximum likelihood. Let us collect the parameters in a vector $\underline{\theta} = (\underline{\mu}_i, \underline{\Gamma}, \underline{\Psi})$ for shorthand notation.

We want to maximize the likelihood of the observed data given the parameters. Assuming the measurements to be i.i.d, we get

$$\hat{\underline{\theta}}_{\text{MLE}} = \underset{\underline{\theta}}{\operatorname{argmax}} \prod_{i=1}^N p_{\underline{\theta}}(\underline{x}_i) \quad (1242)$$

As we do not know the latent variables \underline{S}_i , we have to integrate them out, so

$$p_{\underline{\theta}}(\underline{x}) = \int_S p_{\underline{\theta}}(\underline{x}, \underline{S}) d\underline{S} \stackrel{\text{Bayes}}{=} \int_S p_{\underline{\theta}}(\underline{x} | \underline{S}) p(\underline{S}) d\underline{S} \quad (1243)$$

where in case of our factor analysis

$$\underline{S}_i \sim \mathcal{N}(0, \underline{\underline{1}}) \quad \rightarrow \quad p(\underline{S}_i) = (2\pi)^{-\frac{g}{2}} \exp\left(-\frac{1}{2} \underline{S}_i^T \underline{S}_i\right) \quad (1244)$$

and by our model

$$\underline{x}_i | \underline{S}_i \sim \mathcal{N}(\underline{\mu}_i + \underline{\Gamma}\underline{S}_i, \underline{\underline{\Psi}}) \quad (1245)$$

so

$$p_{\underline{\theta}}(\underline{x}_i | \underline{S}_i) = (2\pi)^{-\frac{p}{2}} \exp\left(-\frac{1}{2} (\underline{x}_i - \underline{\mu}_i - \underline{\Gamma}\underline{S}_i)^T \underline{\underline{\Psi}}^{-1} (\underline{x}_i - \underline{\mu}_i - \underline{\Gamma}\underline{S}_i)\right) \quad (1246)$$

with *factor loadings* $\underline{\Gamma}$, and factor scores \underline{S}_i .

Note: In a more general setting than factor analysis, $p_{\theta}(\underline{x}_i | \underline{S}_i)$ might be a different model, e.g. for p neurons we want to find underlying latent variables s_i modelling if they are firing $x_{ij} \in \{0, 1\}, i = 1, \dots, p, j = 1, \dots, N$ modelled by the Bernoulli distribution.

$$p(x_{ij} | s_i) = \Pi_i(s_i)^{x_{ij}} (1 - \Pi_i(s_i))^{1-x_{ij}} \quad (1247)$$

with e.g. $\Pi_i(s_i) = (\text{soft}(\arg)\max(s_1, \dots, s_p))_i$.

Problem: How do we go about the (intractable) integral over \underline{S}_i ?

24.4 Generative Latent Models

24.4.1 Introduction and problem of the intractable marginal likelihood of the data

Assume we have observations $\underline{x} \in \mathbb{R}^p$ in the data space \mathcal{X} . Also assume there is unobserved latent (or missing) data $\underline{z} \in \mathbb{R}^q$ in the latent space \mathcal{Z} .

- Assume the distribution over the latent variables \underline{z} is given by the prior distribution $p_{\theta_{\text{lat}}}(\underline{z})$ with unknown parameters θ_{lat} .
- Assume a mapping from latent space to data space $f : \mathcal{Z} \rightarrow \mathcal{X}$, and a noise model in the data space $p_{\theta_{\text{obs}}}(\underline{x} | \underline{z})$ with unknown parameters θ_{obs} .

Let us write the parameters in one vector $\underline{\theta} = (\theta_{\text{lat}}, \theta_{\text{obs}})$.

Idea: If we could find the parameters $\underline{\theta}$ and so the distribution $p_{\theta}(\underline{z})$ we could sample data from the simpler latent space and map it to the data space - generative modeling.

Problem: At hand, we only have N samples \underline{x}_i in the data space, and we want to find the parameters $\underline{\theta}$

By Bayes theorem, we can write

$$p_{\theta}(\underline{x}, \underline{z}) = p_{\theta_{\text{lat}}}(\underline{z}) p_{\theta_{\text{obs}}}(\underline{x} | \underline{z}) \quad (1248)$$

so the likelihood of the observed data given the parameters is

$$p_{\underline{\theta}}(\underline{x}) = \int p_{\underline{\theta}}(\underline{x}, \underline{z}) d\underline{z} = \int p_{\underline{\theta}}(\underline{z}) p_{\underline{\theta}}(\underline{x}|\underline{z}) d\underline{z} \quad (1249)$$

Problem: This integral is intractable in general as it is performed over the whole latent space. More profoundly, while we might use importance sampling to calculate this integral for given $\underline{\theta}$, how we integrate itself (so what samples we take) depends in $\underline{\theta}$.

But if we could calculate this marginal likelihood of the observed data, the parameters would follow by

$$\hat{\underline{\theta}}_{\text{MLE}} = \underset{\underline{\theta}}{\operatorname{argmax}} \prod_{i=1}^N p_{\underline{\theta}}(\underline{x}_i) \quad (1250)$$

Latent posterior: Approximating the posterior $p_{\underline{\theta}}(\underline{z}|\underline{x})$ (the discriminative model) would also be nice to do statistical inference in the latent space. Or assuming the latent variables are e.g. unobserved class labels (so clustering), by the discriminative model (the latent posterior), we can assign clusters of new data.

24.4.2 Evidence Lower Bound (ELBO)

We need the marginal likelihood of the observed data, $p_{\underline{\theta}}(\underline{x})$ for calculating the likelihood (and from this the parameters $\underline{\theta}$) but it is intractable.

Idea: Do not maximize the log-likelihood directly but a lower bound on it, which is generally more tractable.

Consider the logarithm of the marginal likelihood of the observed data

$$\log p_{\underline{\theta}}(\underline{x}) = \log \int p_{\underline{\theta}}(\underline{x}, \underline{z}) d\underline{z} \quad (1251)$$

For some fixed $\underline{\theta}$ this is called the **evidence** of the model (if this is high, we would assume that we have chosen a good model and good parameters).

Idea: Introduce a simple, easy to use variational density, one we can easily sample from

$$q_{\underline{\phi}}(\underline{z}) \quad (1252)$$

with parameters $\underline{\phi}$ which aims to approximate the more complicated true posterior $p_{\underline{\theta}}(\underline{z}|\underline{x})$.

Then one can find (proof follows) for any distribution $q_{\underline{\phi}}(\underline{z})$ that

$$\log p_{\underline{\theta}}(\underline{x}) \geq \text{ELBO}(\underline{\theta}) := E_{\underline{z} \sim q_{\underline{\phi}}} \left[\log \frac{p_{\underline{\theta}}(\underline{x}, \underline{z})}{q_{\underline{\phi}}(\underline{z})} \right] \quad (1253)$$

This can be estimated using Monte Carlo importance sampling. For samples $\underline{z}_1, \dots, \underline{z}_M \sim q_{\underline{\phi}}(\underline{z})$ we can estimate

$$\text{ELBO}(\underline{\theta}) \approx \frac{1}{M} \sum_{m=1}^M \log \frac{p_{\underline{\theta}}(\underline{x}, \underline{z}_m)}{q_{\underline{\phi}}(\underline{z}_m)} \quad (1254)$$

(which by Jensen's inequality is biased downwards).

So what have we won in comparison to the original problem?

- the samples from $q_{\underline{\phi}}(\underline{z})$ do not depend on $\underline{\theta}$, so we can optimize the ELBO, different from the original $\int p_{\underline{\theta}}(\underline{z}) p_{\underline{\theta}}(\underline{x}|\underline{z}) d\underline{z}$ where an update to θ would have made new samples necessary.
- if we also optimize the variational density $q_{\underline{\phi}}(\underline{z})$, we can make it more similar to the true posterior, so we get a posterior in the latent space from which we can easily sample and generate new data by the mapping f . Then we can also approximate

$$p_{\underline{\theta}}(\underline{x}) = \frac{p_{\underline{\theta}}(\underline{x}|\underline{z}) p(\underline{z})}{q_{\underline{\phi}}(\underline{z})} \quad (1255)$$

so not by the integral as before.

24.4.2.1 Proof of the ELBO

We have

$$\log p_{\underline{\theta}}(\underline{x}) = \log \int_{\underline{z}} q_{\underline{\phi}}(\underline{z}) \frac{p_{\underline{\theta}}(\underline{x}, \underline{z})}{q_{\underline{\phi}}(\underline{z})} d\underline{z} \stackrel{\text{Jenssen's inequality}}{\geq} \int_{\underline{z}} q_{\underline{\phi}}(\underline{z}) \log \frac{p_{\underline{\theta}}(\underline{x}, \underline{z})}{q_{\underline{\phi}}(\underline{z})} d\underline{z} \quad (1256)$$

where Jenssens inequality holds because \log is concave.

We see that if $q_{\phi}(\underline{z}) = p_{\theta}(\underline{z}|\underline{x})$, then the ELBO is equal to the log-likelihood, as by the *path-rule*

$$\frac{p_{\theta}(\underline{x}, \underline{z})}{p_{\theta}(\underline{z}|\underline{x})} = p_{\theta}(\underline{x}) \quad (1257)$$

On Jenssen's inequality: Generally, Jenssens inequality is given by

$$\begin{aligned} \text{discrete case: } f(\sum \alpha_i x_i) &\geq \sum \alpha_i f(x_i) \\ &\text{for } f \text{ concave, } \alpha_i \geq 0, \sum \alpha_i = 1 \\ \text{continuous case: } f\left(\int \alpha(x) g(x) dx\right) &\geq \int \alpha(x) f(g(x)) dx \\ &\text{for } f \text{ concave, } \alpha(x) \geq 0, \int \alpha(x) dx = 1 \end{aligned} \quad (1258)$$

Which in 1D states, that the secant line between two points on a concave function is always below the function, so

$$f(\alpha x_1 + (1 - \alpha)x_2) \geq \alpha f(x_1) + (1 - \alpha)f(x_2), \quad \alpha \in [0, 1] \quad (1259)$$

This is illustrated in figure 226.

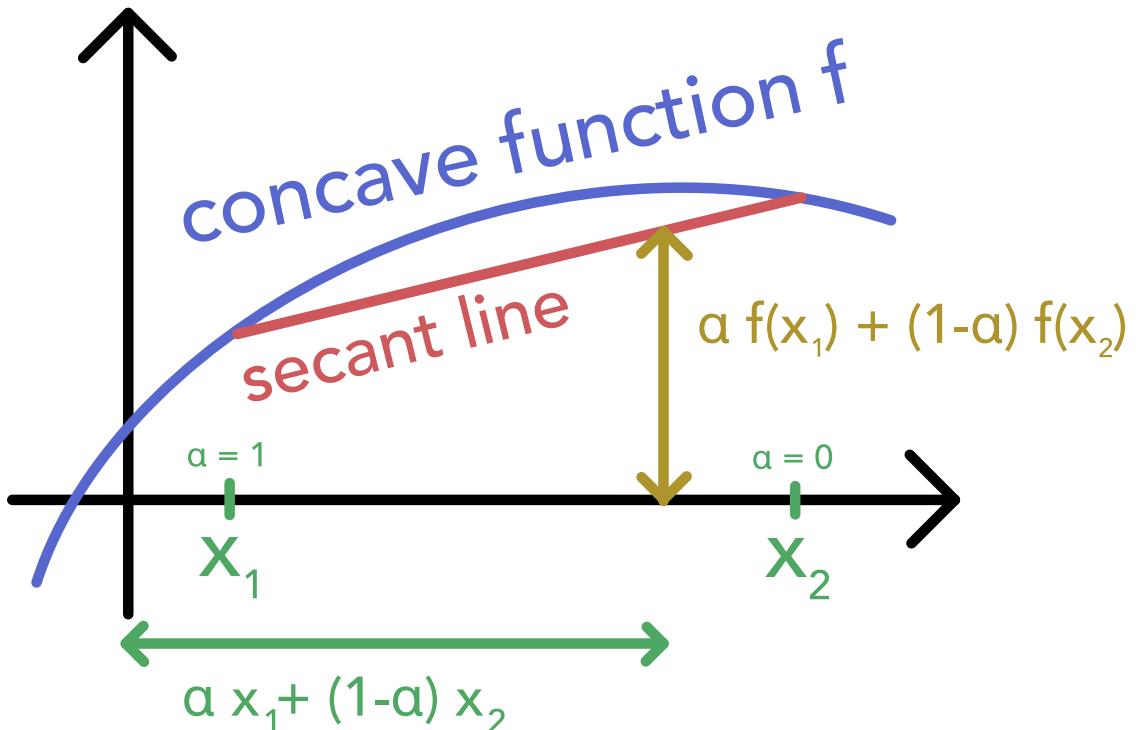


Figure 226: Illustration of Jenssen's inequality.

24.4.2.2 Gap between evidence and ELBO

The gap between the evidence

$$\text{evidence} := \log p_{\underline{\theta}}(\underline{x}) \quad (1260)$$

and the evidence lower bound (ELBO)

$$\text{ELBO}(\underline{\theta}, q_{\underline{\phi}}) = E_{\underline{z} \sim q_{\underline{\phi}}} \left[\log \frac{p_{\underline{\theta}}(\underline{x}, \underline{z})}{q_{\underline{\phi}}(\underline{z})} \right] \quad (1261)$$

turns out to be the Kullback-Leibler divergence between the true posterior and the variational density

$$\begin{aligned} \text{KL}(q_{\underline{\phi}}(\underline{z}) || p_{\underline{\theta}}(\underline{z} | \underline{x})) &= E_{\underline{z} \sim q_{\underline{\phi}}} \left[\log \frac{q_{\underline{\phi}}(\underline{z})}{p_{\underline{\theta}}(\underline{z} | \underline{x})} \right] \\ &= E_{\underline{z} \sim q_{\underline{\phi}}} \left[\log q_{\underline{\phi}}(\underline{z}) \right] - E_{\underline{z} \sim q_{\underline{\phi}}} \left[\underbrace{\log p_{\underline{\theta}}(\underline{z} | \underline{x})}_{= \frac{p_{\underline{\theta}}(\underline{x}, \underline{z})}{p_{\underline{\theta}}(\underline{x})}} \right] \quad (1262) \\ &= E_{\underline{z} \sim q_{\underline{\phi}}} \left[\log q_{\underline{\phi}}(\underline{z}) \right] - E_{\underline{z} \sim q_{\underline{\phi}}} \left[\log p_{\underline{\theta}}(\underline{x}, \underline{z}) \right] + \log p_{\underline{\theta}}(\underline{x}) \\ &= \log p_{\underline{\theta}}(\underline{x}) - \text{ELBO}(\underline{\theta}, q_{\underline{\phi}}) \\ &= \text{evidence} - \text{ELBO} \end{aligned}$$

So while the direct calculation of the posterior $p_{\underline{\theta}}(\underline{z} | \underline{x})$ is intractable, we can approximate it by the variational density $q_{\underline{\phi}}(\underline{z})$ when we maximize the ELBO, under variation of $\underline{\phi}$.

This is illustrated in figure 227.

So maximizing ELBO, gives us both the generative model $p_{\underline{\theta}}(\underline{x}, \underline{z})$ and the discriminative model $p_{\underline{\theta}}(\underline{z} | \underline{x}) \approx q_{\underline{\phi}}(\underline{z})$.

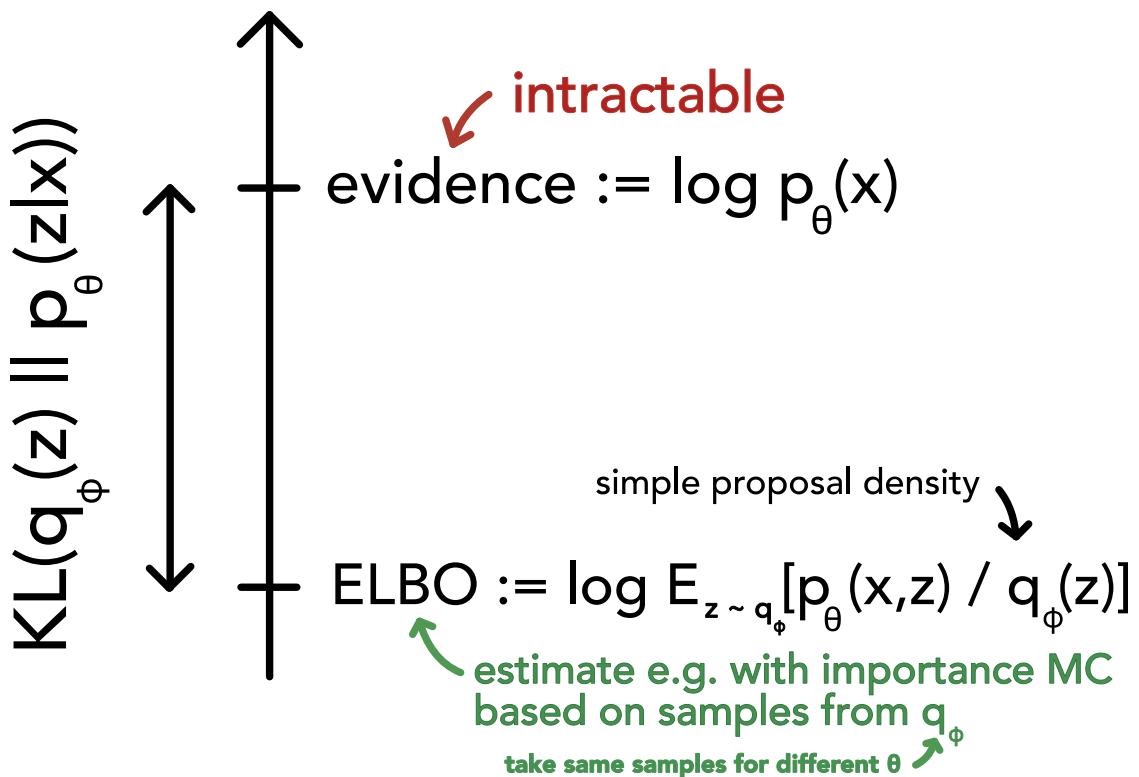


Figure 227: Illustration of the gap between the evidence and the ELBO.

24.4.3 Expectation-Maximization (EM) algorithm

Based on the introduction of ELBO, the following aims are natural

Aims:

- find $q_{\underline{\phi}}(\underline{z})$ so that it is close to the true posterior $p_{\underline{\theta}}(\underline{z}|\underline{x})$
- find parameters $\underline{\theta}$ (our model parameters) that maximize the ELBO

The EM algorithm consists of (here given as a maximization-maximization procedure)

- **Expectation-step:** Our baseline problem is that we do not know the distribution of the latent variables given our observations. So given a current estimate of the model parameters $\underline{\theta}^*$, find a possible distribution $q_{\underline{\phi}}(\underline{z})$ that is close to the true posterior $p_{\underline{\theta}}(\underline{z}|\underline{x})$ so optimally maximizing the ELBO

$$\underline{\phi}^* = \operatorname{argmax}_{\underline{\phi}} \text{ELBO}(\underline{\theta}^*, q_{\underline{\phi}}) \quad (1263)$$

We might not do this maximization explicitly, but ask ourselves what latent distribu-

tion would be reasonable given our current parameters. With this we can calculate

$$\text{ELBO}(\underline{\theta}) = E_{\underline{z} \sim q_{\phi^*}} \left[\log \frac{p_{\underline{\theta}}(\underline{x}, \underline{z})}{q_{\phi^*}(\underline{z})} \right] \quad (1264)$$

which is then **only a function of $\underline{\theta}$** .

- **Maximization-step:** Assume the above expectation was calculated under some $\underline{\phi}^*$ reasonably good, find the model parameters $\underline{\theta}$ that maximize the ELBO

$$\underline{\theta}^* = \underset{\underline{\theta}}{\operatorname{argmax}} \text{ELBO}(\underline{\theta}) \quad (1265)$$

Starting with initial estimates $\underline{\theta}^{(0)}, \underline{\phi}^{(0)}$, the EM algorithm iterates until $\Delta \text{ELBO} < \epsilon$ for some small ϵ .

Note: This is not guaranteed to converge but converges for some problems.

As of our estimated parameters, $p_{\underline{\theta}}(\underline{z})$ and $p_{\underline{\theta}}(\underline{x}|\underline{z})$ are known (tractable distributions in the first place) (mind that $\underline{\theta}$ is the collection of parameters of both) and the intractable latent posterior

$$p_{\underline{\theta}}(\underline{z}|\underline{x}) \approx q_{\underline{\phi}}(\underline{z}) \quad (1266)$$

and the intractable marginal likelihood of the data

$$p_{\underline{\theta}}(\underline{x}) = \int p_{\underline{\theta}}(\underline{x}, \underline{z}) d\underline{z} \approx \frac{p_{\underline{\theta}}(\underline{x}|\underline{z}p_{\underline{\theta}}(\underline{z}))}{q_{\underline{\phi}}(\underline{z})} \quad (1267)$$

can be approximated.

24.4.4 Application of the EM algorithm: Gaussian Mixture Models (GMM)

Consider we want to estimate the probability distribution of data $\underline{\underline{X}}$ by a mixture of Gaussians. This is illustrated in figure 228.

In this case our latent variables are assignments of data points to specific Gaussians, so to specific clusters.

We approximate the density of the data $\mathcal{D} = \{\underline{x}_1, \dots, \underline{x}_N\}$ (i.i.d.) by

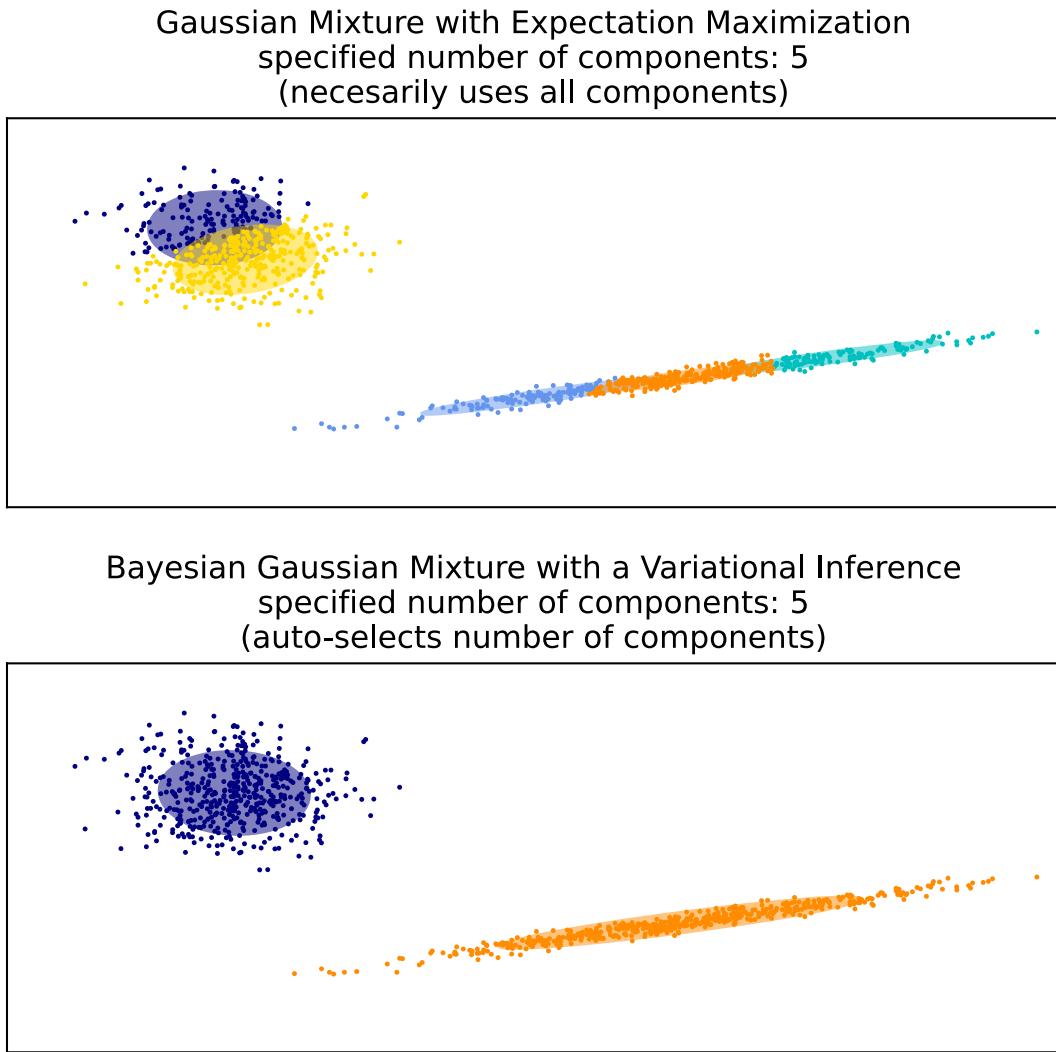


Figure 228: Illustration of a Gaussian Mixture Model.

$$f(\underline{x}) = \sum_{k=1}^K \Pi_k \mathcal{N}(\underline{x} | \underline{\mu}_k, \underline{\Sigma}_k), \quad \sum_{k=1}^K \Pi_k = 1 \quad (1268)$$

(other basic models, e.g. Bernoulli would also be possible).

The model parameters are

$$\underline{\theta} = \{\Pi_1, \underline{\mu}_1, \underline{\Sigma}_1, \dots, \Pi_K, \underline{\mu}_K, \underline{\Sigma}_K\} \quad (1269)$$

so the likelihood is

$$\log \mathcal{L}_{\underline{\theta}}(\mathcal{D}) = \log p_{\underline{\theta}}(\mathcal{D}) = \log \prod_{i=1}^N \sum_{k=1}^K \Pi_k p(\underline{x}_i | \underline{\theta}_k), \quad p(\underline{x}_i | \underline{\theta}_k) = \mathcal{N}(\underline{x}_i | \underline{\mu}_k, \underline{\Sigma}_k) \quad (1270)$$

Now introduce the latent variables $\underline{z}_i \in \{0, 1\}^K$ with $z_{ik} = 1$ if \underline{x}_i is assigned to the k -th Gaussian (1-hot encoded).

E-step - updating the distribution over \underline{z}_i , $q_{\underline{\phi}}(\underline{z}_i)$ Naturally, the probability that \underline{x}_i is assigned to the k -th Gaussian is given by the probability of that Gaussian at \underline{x}_i (with its weighting as in the density approximation) divided by the sum of the probabilities of all Gaussians at \underline{x}_i (with their respective general weightings),

$$\begin{aligned} \forall i, k : \quad p(z_{ik} = 1 | \underline{x}_i, \underline{\theta}^{(n)}) &= \frac{p(\underline{x}_i | \underline{z}; \underline{\theta}^{(n)}) p(z_{ik} = 1 | \underline{\theta}^{(n)})}{p(\underline{x}_i | \underline{\theta}^{(n)})} \\ &= \frac{\Pi_k \mathcal{N}(\underline{x}_i | \underline{\mu}_k, \underline{\Sigma}_k)}{\sum_{l=1}^K \Pi_l \mathcal{N}(\underline{x}_i | \underline{\mu}_l, \underline{\Sigma}_l)} \end{aligned} \quad (1271)$$

where here at each point we have the posterior categorical distribution

$$q_{\underline{\phi}}(\underline{z}_i) = \prod_{k=1}^K (\phi_{ik})^{z_{ik}}, \quad \phi_{ik} = p(z_{ik} = 1 | \underline{x}_i, \underline{\theta}^{(n)}) \quad (1272)$$

M-step - updating the model parameters We want to find the model parameters that maximize the ELBO. As an approximation to the true posterior $p_{\underline{\theta}}(\underline{z} | \underline{x})$, we use the variational density $q_{\underline{\phi}}(\underline{z})$.

$$\begin{aligned} \underline{\theta}^{(n+1)} &= \operatorname{argmax}_{\underline{\theta}} E_{\underline{z} \sim q_{\underline{\phi}}} [\log p_{\underline{\theta}}(\{\underline{x}, \underline{z}\})] \\ &= \operatorname{argmax}_{\underline{\theta}} \sum_{\underline{z}} q_{\underline{\phi}}(\underline{z}) \log \underbrace{p_{\underline{\theta}}(\{\underline{x}, \underline{z}\})}_{=\prod_{i=1}^N p_{\underline{\theta}}(\underline{x}_i, z_i)} \\ &= \operatorname{argmax}_{\underline{\theta}} \sum_{i=1}^N \sum_{k=1}^K \underbrace{\phi_{ik}}_{\text{from E-step}} \log p_{\underline{\theta}}(\underline{x}_i, z_i) \\ &= \operatorname{argmax}_{\underline{\theta}} \sum_{i=1}^N \sum_{k=1}^K \phi_{ik} [\log p_{\underline{\theta}}(\underline{x}_i | z_i) + \log p_{\underline{\theta}}(z_i)] \\ &= \operatorname{argmax}_{\underline{\theta}} \sum_{i=1}^N \sum_{k=1}^K \phi_{ik} \left[\log \mathcal{N}(\underline{x}_i | \underline{\mu}_k, \underline{\Sigma}_k) + \log \prod_{r=1}^K \Pi_r^{z_{ir}} \right] \end{aligned} \quad (1273)$$

We maximize this in $\underline{\theta} = \{\Pi_1, \underline{\mu}_1, \underline{\Sigma}_1, \dots, \Pi_K, \underline{\mu}_K, \underline{\Sigma}_K\}$ to find the next iteration of the model parameters $\underline{\theta}^{(n+1)}$.

Gaussian mixture as a generative model:

1. From the categorical distribution of the $\Pi_l, l = 1, \dots, K$, we can sample the class of a new data point
2. Given the class, we can sample a new data point from the Gaussian distribution with the parameters $\underline{\mu}_l, \underline{\Sigma}_l$

In general the generative process is

$$\underline{z} \sim p_{\underline{\theta}}(\underline{z}) \quad \rightarrow \quad \underline{x} \sim p_{\underline{\theta}}(\underline{x}|\underline{z}) \quad (1274)$$

Note: The Gaussian mixture model with the EM algorithm can not itself find the number of clusters K in the data, use e.g. cross-validation for this.

24.4.5 Variational Inference

24.4.5.1 Recapitulation on parameter estimation | parameters vs. latent variables

Let us go back a step. Previously, we had a model for the likelihood given parameters $p(\underline{x}|\underline{\theta})$ - for instance a Gaussian with given parameters μ and σ so over our whole dataset $\mathcal{D} = \{\underline{x}_1, \dots, \underline{x}_N\}$ under i.i.d. assumptions the likelihood

$$p(\mathcal{D}|\underline{\theta}) = \mathcal{L}_{\mathcal{D}}(\underline{\theta}) = \prod_{i=1}^N p(\underline{x}_i|\underline{\theta}) \quad (1275)$$

where

- in the *frequentist* MLE approach we would find a point-estimate of the parameters maximizing the likelihood
- and in the Bayesian setting, we would assume a prior distribution $p(\underline{\theta})$ (and as before the likelihood) and by Bayes theorem calculate a full distribution over the parameters

$$p(\underline{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\underline{\theta})p(\underline{\theta})}{\int p(\underline{\theta})p(\mathcal{D}|\underline{\theta})d\underline{\theta}} \quad (1276)$$

Where as of the *evidence integral* this posterior would usually be intractable, but we can sample from it using Marcov Chain Monte Carlo (MCMC) (where the evidence

drops out), and thus calculate integrals over the posterior, e.g. the first moment, the posterior mean (just the mean over the MCMC samples). Note that also here, the prior and posterior are distributions which themselves have parameters, not explicit here which would be updated in a Bayesian update step.

Latent variables vs. parameters: Parameters are often connoted to the idea of having fixed values, not being random variables with a distribution. In the above Bayesian setting it therefore makes more sense to think of the *parameters* as **latent variables**.

24.4.5.2 The general problem of inference: Sampling vs. Variational approach

We have

- data $\mathcal{D} = \{\underline{x}_i, \dots, \underline{x}_N\}$
- a prior on unobserved / hidden latent variables \underline{z} , e.g. parameters $\underline{\theta}$, e.g. flat if we have no prior knowledge
- a model of the likelihood $p(\underline{x}|\underline{z})$, which in the general latent setting might need more parameters to be specified (finding which one can e.g. use expectation maximization), or if we are in the above inference setting is fully given by our parameters of interest $\underline{z} = \underline{\theta}$, those we are looking for

Our joint model is

$$p(\underline{z}, \underline{x}) = p(\underline{z})p(\underline{x}|\underline{z}) \quad (1277)$$

where the latent variables help govern the distribution of the data.

Aim: Given a measurement \underline{x} we are interested in the posterior (discriminative model) $p(\underline{z}|\underline{x})$ so in the parameter inference setting $p(\underline{\theta}|\underline{x})$ (or rather this posterior given the full dataset).

The main two approaches are

- **Sampling Techniques:** We can sample from $p(\underline{z}|\underline{x})$ by MCMC (which only requires knowing the likelihood and prior, the evidence drops out) and construct an empirical estimate of the posterior.
 - this is unbiased in the sense that we make no distributional assumption on the posterior
 - this might be too costly - trillions of MCMC samples might be necessary
- **Variational Inference:** Use optimization rather than sampling. Use an approximate

density from some family $q(\underline{z}) \in \mathcal{Q}$ meant to approximate the posterior (ideally expressive and tractable). Find the family-member best approximating the true posterior by minimizing the Kullback-Leibler divergence

$$q^*(\underline{z}) = \underset{q(\underline{z}) \in \mathcal{Q}}{\operatorname{argmin}} D_{KL}(q(\underline{z}) || p(\underline{z} | \underline{x})) \quad (1278)$$

which we do by maximizing the evidence lower bound as before.

When to use MCMC, when variational methods?: MCMC methods tend to be computationally intensive but guarantee producing asymptotically exact samples from the target density. Variational inference (VI) does not have these guarantees but is usually faster.

- use MCMC when: smaller dataset, higher computational cost for more precise samples is fine
- use VI when: large data sets, quickly want to explore many models, so when speed is requested, posterior is sufficiently simple to be modeled by the given family \mathcal{Q}

The comparison of the two methods is illustrated in figure 229.

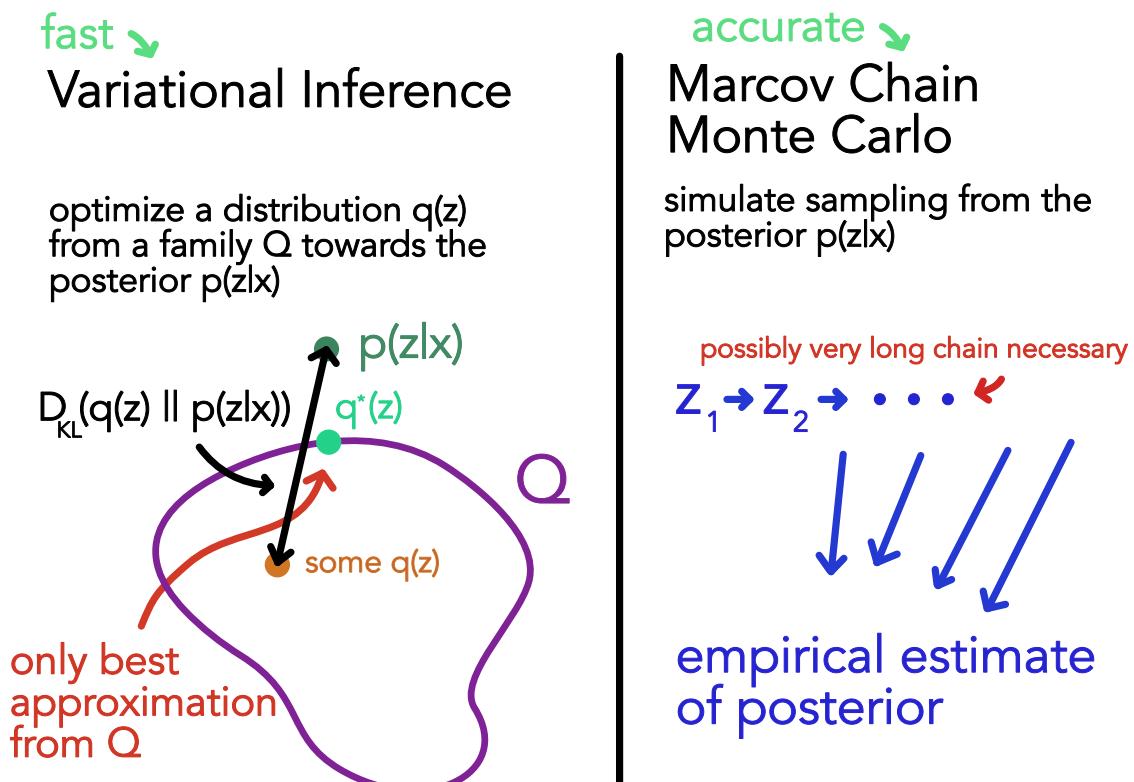


Figure 229: Comparison of variational inference and MCMC.

24.4.5.3 Variational inference

Idea: Approximate the posterior $p(\underline{z}|\underline{x})$ by $q(\underline{z}) \in \mathcal{Q}$

$$\begin{aligned}
q^*(\underline{z}) &= \underset{q(\underline{z}) \in \mathcal{Q}}{\operatorname{argmin}} D_{KL}(q(\underline{z})||p(\underline{z}|\underline{x})) \\
&= \underset{q(\underline{z}) \in \mathcal{Q}}{\operatorname{argmin}} \left(\int q(\underline{z}) \log \frac{q(\underline{z})}{p(\underline{z}|\underline{x})} d\underline{z} \right) \\
&= \underset{q(\underline{z}) \in \mathcal{Q}}{\operatorname{argmin}} \left(\int q(\underline{z}) \log \frac{q(\underline{z})p(\underline{x})}{p(\underline{z}, \underline{x})} d\underline{z} \right) \\
&= \underset{q(\underline{z}) \in \mathcal{Q}}{\operatorname{argmin}} \left(\underbrace{\int q(\underline{z}) \log p(\underline{x}) d\underline{z}}_{=\text{evidence } \log p(\underline{x})} - \underbrace{\int q(\underline{z}) \log \frac{p(\underline{z}, \underline{x})}{q(\underline{z})} d\underline{z}}_{=\text{ELBO}(q)} \right) \\
&= \underset{q(\underline{z}) \in \mathcal{Q}}{\operatorname{argmax}} \text{ELBO}(q)
\end{aligned} \tag{1279}$$

24.4.5.4 Mean-Field Algorithm

Here we assume the variational density (the approximate posterior) to factorize over the latent variables

$$q(\underline{z}) = \prod_{i=1}^N q_i(z_i) \tag{1280}$$

(or at least to be partitionable), where for the q_i one can find the best approximation to the true posterior by calculus of variations, by distributional assumptions, an explicit form for the best q_i can often be found.

24.4.6 Variational Autoencoder (VAE)

24.4.6.1 Back to the problem of likelihood optimization under latent variables

Let us go back to the problem of maximizing the likelihood $p_{\underline{\theta}}(\underline{x}) = p(\underline{x}|\underline{\theta})$. If this likelihood is sufficiently simple, we can find parameters $\underline{\theta}$ by usual MLE.

Problem: What if we have further latent, unobserved variables, e.g. unobserved class labels?

We marginalize them out by

$$p_{\underline{\theta}}(\underline{x}) = \int p_{\underline{\theta}}(\underline{x}, \underline{z}) d\underline{z} \underset{\text{chain rule}}{=} \int p_{\underline{\theta}}(\underline{x}|\underline{z}) p_{\underline{\theta}}(\underline{z}) d\underline{z} \tag{1281}$$

e.g. $p_{\underline{\theta}}(\underline{x}|\underline{z})$ is a Gaussian and thus $p_{\underline{\theta}}(\underline{x})$ is a mixture of Gaussians.

Problem:

- This integral is generally problematic, more so if we want to optimize θ through it.
- The posterior $p_{\underline{\theta}}(\underline{z}|\underline{x})$ is expensive / intractable to calculate.

Idea: As previously, we introduce the approximation

$$q_{\underline{\phi}}(\underline{z}|\underline{x}) \underset{\text{goal}}{\approx} p_{\underline{\theta}}(\underline{z}|\underline{x}) \quad (1282)$$

Finding a good approximation $q_{\underline{\phi}}(\underline{z}|\underline{x})$ later amortizes, because we can then infer a \underline{z} from a \underline{x} without doing any integrals (*amortized inference*).

Also as previously, we find good parameters $\underline{\theta}^*$ and $\underline{\phi}^*$ by maximizing the ELBO

$$\underline{\theta}^*, \underline{\phi}^* = \underset{\underline{\theta}, \underline{\phi}}{\operatorname{argmax}} \text{ELBO}_{\underline{\theta}, \underline{\phi}}(\underline{x}) \quad (1283)$$

where a high evidence lower bound is wished for as

$$\text{ELBO}_{\underline{\theta}, \underline{\phi}}(\underline{x}) = E_{\underline{z} \sim q_{\underline{\phi}}(\underline{z})} \left[\log \frac{p_{\underline{\theta}}(\underline{x}, \underline{z})}{q_{\underline{\phi}}(\underline{z})} \right] = \underbrace{\log p_{\underline{\theta}}(\underline{x})}_{\substack{\text{quality of} \\ \text{the fit of} \\ \text{the model}}} - \underbrace{D_{KL}(q_{\underline{\phi}}(\underline{z}) || p_{\underline{\theta}}(\underline{z}|\underline{x}))}_{\substack{\text{correctness of latent} \\ \text{model} \\ \text{to the} \\ \text{data}}} \quad (1284)$$

Note: The notation $\text{ELBO}_{\underline{\theta}, \underline{\phi}}(\underline{x})$ makes the dependence on \underline{x} more explicit than previously. The distributions $q_{\underline{\phi}}(\underline{z})$ are normally parameterized for each individual data point in a separate optimization process. More explicitly, one would always write $q_{\underline{\phi}}(\underline{z}|\underline{x})$.

24.4.6.2 The Variational Autoencoder

We now take a probabilistic approach to the autoencoder

- **Probabilistic Encoder:** The encoder $h_{\Omega_{\text{Dec}}}(\underline{x})$ takes input points from the normal data space \underline{x} and outputs parameters $\underline{\phi}$ for the variational density $q_{\underline{\phi}}(\underline{z})$. One can then for this location \underline{x} sample a \underline{z} from $q_{\underline{\phi}}(\underline{z})$ with the obtained parameters.
- **Probabilistic Decoder:** The decoder $f_{\Omega_{\text{Dec}}}(\underline{z})$ maps from the latent space \underline{z} to parameters $\underline{\theta}$ for the likelihood $p_{\underline{\theta}}(\underline{x}|\underline{z})$, from which we can sample a \underline{x} . (Only mapping

to the mean of the likelihood is also possible.)

While the distributions $q_\phi(\underline{z})$ and $p_\theta(\underline{x}|\underline{z})$ might generally be very simple, as of the possibly complex mappings $h_{\Omega_{\text{Dec}}}$ and $f_{\Omega_{\text{Dec}}}$, the variational autoencoder can model very complex distributions.

The general variational autoencoder idea is illustrated in figure 230.

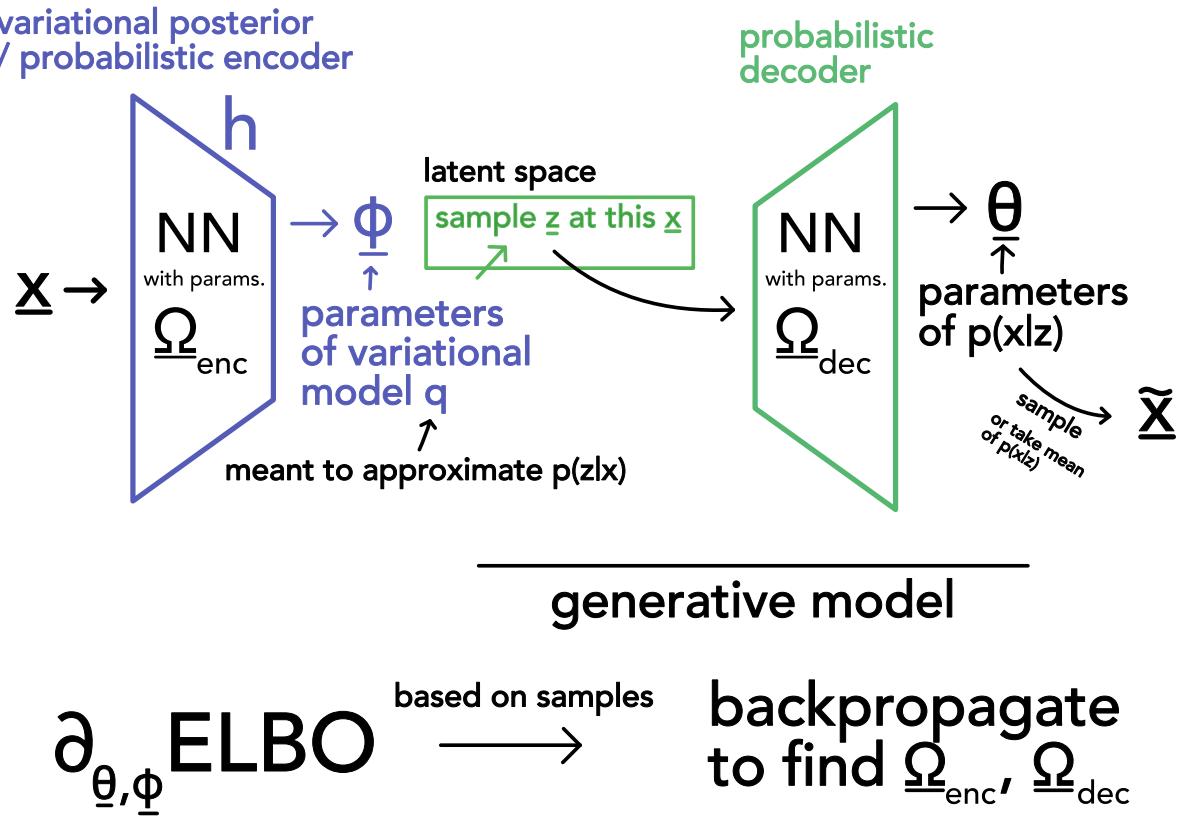


Figure 230: Illustration of the variational autoencoder idea.

24.4.6.3 Gradient of the ELBO

The ELBO is given by

$$\text{ELBO}_{\underline{\theta}, \underline{\phi}}(\underline{x}) = E_{\underline{z} \sim q_\phi(\underline{z})} \left[\log \frac{p_\theta(\underline{x}, \underline{z})}{q_\phi(\underline{z})} \right] \quad (1285)$$

For a given \underline{x} , we calculate the parameters $\underline{\phi}$ for the variational density $q_\phi(\underline{z})$ by the encoder $\underline{\phi} = h_{\Omega_{\text{Dec}}}(\underline{x})$ and draw

$$\underline{z}^{(l)} \sim q_\phi(\underline{z}), \quad l = 1, \dots, L \quad (1286)$$

With the decoder we calculate $\underline{\theta} = f_{\Omega_{\text{Dec}}}(\underline{z}^{(l)})$. With this an estimate of the ELBO is given by

$$\text{ELBO}_{\underline{\theta}, \underline{\phi}}(\underline{x}) \approx \frac{1}{L} \sum_{l=1}^L \left(\log p_{\underline{\theta}}(\underline{x} | \underline{z}^{(l)}) - \log q_{\underline{\phi}}(\underline{z}^{(l)}) \right) \quad (1287)$$

For being able to backpropagate to $\underline{\Omega}_{\text{Dec}}$ and $\underline{\Omega}_{\text{Dec}}$, we need to calculate the gradient of the ELBO with respect to the parameters

$$\partial_{\underline{\theta}, \underline{\phi}} \text{ELBO}_{\underline{\theta}, \underline{\phi}}(\underline{x}) \quad (1288)$$

Note: In other articles, θ and ϕ are often used to denote the parameters of the encoder and decoder, respectively (as the parameters of the variational density and the likelihood follow from them).

We can readily calculate

$$\partial_{\underline{\theta}} \text{ELBO}_{\underline{\theta}, \underline{\phi}}(\underline{x}) = \partial_{\underline{\theta}} E_{\underline{z} \sim q_{\underline{\phi}}(\underline{z})} \left[\log \frac{p_{\underline{\theta}}(\underline{x}, \underline{z})}{q_{\underline{\phi}}(\underline{z})} \right] = E_{\underline{z} \sim q_{\underline{\phi}}(\underline{z})} [\partial_{\underline{\theta}} \log p_{\underline{\theta}}(\underline{x} | \underline{z})] \quad (1289)$$

Problem: In the calculation

$$\partial_{\underline{\phi}} \text{ELBO}_{\underline{\theta}, \underline{\phi}}(\underline{x}) = \partial_{\underline{\phi}} E_{\underline{z} \sim q_{\underline{\phi}}(\underline{z})} \left[\log \frac{p_{\underline{\theta}}(\underline{x}, \underline{z})}{q_{\underline{\phi}}(\underline{z})} \right] \quad (1290)$$

we cannot put the derivative inside the expectation, because this expectation (how we sample the \underline{z}) depends on the parameters $\underline{\phi}$ itself.

Reparametrization trick: Write $z^{(l)}$ as a deterministic function depending on $\underline{\phi}$ and a parameter-free random variable $\epsilon^{(l)}$ which is independent of $\underline{\phi}$.

$$z^{(l)} = g_{\underline{\phi}}(\epsilon^{(l)}, \underline{x}) \quad (1291)$$

for instance assume

$$z^{(l)} \sim \mathcal{N}(\mu_{\underline{\phi}}(\underline{x}), \sigma_{\underline{\phi}}(\underline{x})) \quad \rightarrow \quad z^{(l)} = \mu_{\underline{\phi}}(\underline{x}) + \sigma_{\underline{\phi}}(\underline{x})\epsilon^{(l)}, \quad \epsilon^{(l)} \sim \mathcal{N}(0, 1) \quad (1292)$$

(where μ and σ are the outputs of the encoder) then we can rewrite

$$E_z[h(z)] = E_{\epsilon} \left[h(g_{\underline{\phi}}(\epsilon, \underline{x})) \right] \quad (1293)$$

So we do not directly sample from $q_{\underline{\phi}}(\underline{z})$, but from a surrogate model, where the distribution does not depend on the parameters $\underline{\phi}$ but we still ensure that the sampled \underline{z} follows our variational density. This allows us to put the derivative inside the expectation.

References

- Ardourel, Vincent (2017). »Irreversibility in the derivation of the Boltzmann equation«. In: *Foundations of physics* 47.4, pages 471–489.
- Bagla, J. S. (Dec. 2002). »TreePM: A code for cosmological N-body simulations«. In: *Journal of Astrophysics and Astronomy* 23.3–4, pages 185–196. DOI: 10.1007/bf02702282. URL: <http://dx.doi.org/10.1007/BF02702282>.
- Belkin, Mikhail et al. (July 2019). »Reconciling modern machine-learning practice and the classical bias–variance trade-off«. In: *Proceedings of the National Academy of Sciences* 116.32, pages 15849–15854. DOI: 10.1073/pnas.1903070116. URL: <http://dx.doi.org/10.1073/pnas.1903070116>.
- Biamonte, Jacob et al. (2017). »Quantum machine learning«. In: *Nature* 549.7671, pages 195–202.
- Box, G. E. P. and Mervin E. Muller (1958). »A Note on the Generation of Random Normal Deviates«. In: *The Annals of Mathematical Statistics* 29.2, pages 610–611. DOI: 10.1214/aoms/1177706645. URL: <https://doi.org/10.1214/aoms/1177706645>.
- Breiman, Leo (2001). »Statistical modeling: The two cultures (with comments and a rejoinder by the author)«. In: *Statistical science* 16.3, pages 199–231.
- Bryant, Randal E and David Richard O'Hallaron (2011). *Computer systems: a programmer's perspective*. Prentice Hall.
- Chou, C et al. (2007). »Numerical methods for stiff reaction-diffusion systems«. In: *DISCRETE AND CONTINUOUS DYNAMICAL SYSTEMS SERIES B* 7.3, page 515.
- Courant, R., K. Friedrichs, and H. Lewy (Dec. 1928). »Über die partiellen Differenzengleichungen der mathematischen Physik«. In: *Mathematische Annalen* 100.1, pages 32–74. DOI: 10.1007/BF01448839. URL: <https://doi.org/10.1007/BF01448839>.
- Crammer, Koby and Yoram Singer (2001). »On the algorithmic implementation of multiclass kernel-based vector machines«. In: *Journal of machine learning research* 2.Dec, pages 265–292.
- Efron, Bradley and R.J. Tibshirani (May 1994). *An Introduction to the Bootstrap*. Chapman and Hall/CRC. DOI: 10.1201/9780429246593. URL: <http://dx.doi.org/10.1201/9780429246593>.
- Gould, Harvey et al. (1996). »An introduction to computer simulation methods: applications to physical systems«. In: *Computers in Physics* 10.4, pages 349–349.
- Graziani, F. et al. (2022). »Shock physics in warm dense matter: A quantum hydrodynamics perspective«. In: *Contributions to Plasma Physics* 62.2, e202100170. DOI: <https://doi.org/10.1002/ctpp.202100170>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/ctpp.202100170>.

- Hairer, Ernst (June 2006). »Long-time energy conservation of numerical integrators«. In: *Lecture Notes Ser. FoCM Santander 2005* 331. DOI: 10.1017/CBO9780511721571.005.
- Hairer, Ernst, Christian Lubich, and Gerhard Wanner (2003). »Geometric numerical integration illustrated by the Störmer–Verlet method«. In: *Acta Numerica* 12, pages 399–450. DOI: 10.1017/S0962492902000144.
- Hairer, Ernst and Gerhard Wanner (1996). *Solving Ordinary Differential Equations II*. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-05221-7. URL: <https://doi.org/10.1007/978-3-642-05221-7>.
- Hairer, Ernst, Gerhard Wanner, and Christian Lubich (2006). *Geometric Numerical Integration*. Springer-Verlag. DOI: 10.1007/3-540-30666-8. URL: <https://doi.org/10.1007/3-540-30666-8>.
- Hairer, Ernst, Gerhard Wanner, and Syvert P. Nørsett (1993). *Solving Ordinary Differential Equations I*. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-78862-1. URL: <https://doi.org/10.1007/978-3-540-78862-1>.
- Heiter, Pascal Frederik (2012). »On Numerical Methods for Stiff Ordinary Differential Equation Systems«. Master's thesis. Ulm University. URL: https://www.uni-ulm.de/fileadmin/website_uni_ulm/mawi.inst.070/abschlussarbeiten/masterthesis_pfh.pdf.
- Hesterberg, Tim C. (2015). »What Teachers Should Know About the Bootstrap: Resampling in the Undergraduate Statistics Curriculum«. In: *The American Statistician* 69.4. PMID: 27019512, pages 371–386. DOI: 10.1080/00031305.2015.1089789. URL: <https://doi.org/10.1080/00031305.2015.1089789>.
- Higham, Nicholas J. (2002). *Accuracy and Stability of Numerical Algorithms*. Second. Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9780898718027. URL: <https://epubs.siam.org/doi/abs/10.1137/1.9780898718027>.
- Hirashima, Keiya et al. (2023). *Surrogate Modeling for Computationally Expensive Simulations of Supernovae in High-Resolution Galaxy Simulations*.
- Jagode, Heike (2006). »Fourier Transforms for the BlueGene / L Communication Network«. In: URL: <https://api.semanticscholar.org/CorpusID:18464112>.
- Kincl, Ondřej and Michal Pavelka (Mar. 2023). »Globally time-reversible fluid simulations with smoothed particle hydrodynamics«. In: *Computer Physics Communications* 284, page 108593. DOI: 10.1016/j.cpc.2022.108593. URL: <http://dx.doi.org/10.1016/j.cpc.2022.108593>.
- Lambert, J.D. (1991). *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. Wiley. URL: <https://books.google.de/books?id=P0vPnQEACAAJ>.
- Learned-Miller, Erik G and John W Fisher III (2003). »ICA using spacings estimates of entropy«. In: *The Journal of Machine Learning Research* 4, pages 1271–1295.

- Lefever, R. and G. Nicolis (1971). »Chemical instabilities and sustained oscillations«. In: *Journal of Theoretical Biology* 30.2, pages 267–284. DOI: [https://doi.org/10.1016/0022-5193\(71\)90054-3](https://doi.org/10.1016/0022-5193(71)90054-3). URL: <https://www.sciencedirect.com/science/article/pii/0022519371900543>.
- Lewis, Roland W, Perumal Nithiarasu, and Kankanhalli N Seetharamu (2004). *Fundamentals of the finite element method for heat and fluid flow*. John Wiley & Sons.
- Margolin, L. G. and N. M. Lloyd-Ronning (June 2023). »Artificial viscosity—then and now«. In: *Meccanica* 58.6, pages 1039–1052. DOI: 10.1007/s11012-022-01541-5. URL: <https://doi.org/10.1007/s11012-022-01541-5>.
- Matsumoto, Makoto and Takuji Nishimura (1998). »Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator«. In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8.1, pages 3–30.
- Mitchell, Nigel L., Eduard I. Vorobyov, and Gerhard Hensler (Nov. 2012). »Collisionless stellar hydrodynamics as an efficient alternative to N-body methods«. In: *Monthly Notices of the Royal Astronomical Society* 428.3, pages 2674–2687. DOI: 10.1093/mnras/sts228. URL: <https://doi.org/10.1093/mnras/sts228>.
- Moser, Jürgen (1978). »Is the solar system stable?« In: *The Mathematical Intelligencer* 1.2, pages 65–71. DOI: 10.1007/BF03023062. URL: <https://doi.org/10.1007/BF03023062>.
- Particle-Data-Group et al. (Aug. 2020). »Review of Particle Physics«. In: *Progress of Theoretical and Experimental Physics* 2020.8, page 083C01. DOI: 10.1093/ptep/ptaa104. URL: <https://doi.org/10.1093/ptep/ptaa104>.
- Press, William H. et al. (2007). *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd edition. USA: Cambridge University Press.
- Rackauckas, Christopher and Qing Nie (2017). »DifferentialEquations.jl—a performant and feature-rich ecosystem for solving differential equations in julia«. In: *Journal of Open Research Software* 5.1, page 15.
- Rackauckas, Christopher, Sciemon, et al. (Nov. 2022). *SciML/SciMLBook: v1.1*. Version v1.1. DOI: 10.5281/zenodo.7347643. URL: <https://doi.org/10.5281/zenodo.7347643>.
- Rein, Hanno and David S. Spiegel (Nov. 2014). »ias15: a fast, adaptive, high-order integrator for gravitational dynamics, accurate to machine precision over a billion orbits«. In: *Monthly Notices of the Royal Astronomical Society* 446.2, pages 1424–1437. DOI: 10.1093/mnras/stu2164. URL: <http://dx.doi.org/10.1093/mnras/stu2164>.
- Schlottke-Lakemper, Michael et al. (2021). »A purely hyperbolic discontinuous Galerkin approach for self-gravitating gas dynamics«. In: *Journal of Computational Physics* 442, page 110467.
- Springel, Volker et al. (2023). *Lecture notes in Fundamentals of Simulation Methods*.
- Ulmann, Bernd (May 2020). *Analog and Hybrid Computer Programming*. De Gruyter. DOI: 10.1515/9783110662207. URL: <http://dx.doi.org/10.1515/9783110662207>.

- van Leer, Bram (1979). »Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method«. In: *Journal of Computational Physics* 32.1, pages 101–136. DOI: [https://doi.org/10.1016/0021-9991\(79\)90145-1](https://doi.org/10.1016/0021-9991(79)90145-1). URL: <https://www.sciencedirect.com/science/article/pii/0021999179901451>.
- Williams, Christopher and Carl Rasmussen (1995). »Gaussian processes for regression«. In: *Advances in neural information processing systems* 8.