UNIT 4

ODEs, PDEs, SDEs, Neural ODEs

Markus Holzleitner, Johannes Brandstetter
Institute for Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ

Institute for
Machine Learning

Copyright statement:

ODEs

What is an ODE?

- An ordinary differential equation is a relation of the form

$$\frac{d^n x(t)}{dt^n} = F(t, x(t), \frac{dx(t)}{dt}, \cdots, \frac{d^{n-1}x(t)}{dt^{n-1}}),$$

  where $x(t)$ is sought function and t independent variable.
- Characteristic properties:
  - n is called **order** of the ODE
  - If $x(t)$ and derivatives appear only linearly in F, the ODE is called **linear**:

$$F(t, x(t), \frac{dx(t)}{dt}, \cdots, \frac{d^{n-1}x(t)}{dt^{n-1}}) = g(t) + \sum_{k=0}^{n-1} h_k(t) \frac{d^k x(t)}{dt^k}$$

  - The **coefficients** h_k may be constant.
  - If F does not depend on t, it is called **homogeneous**.

ODE example: separable

- Let's start with a simple equation:

$$\frac{dx(t)}{dt} = atx(t)$$

  - Is **first order**, **linear** with variable coefficients, and **homogeneous**.
  - It is called **separable**: $\frac{dx(t)}{dt} = f(x(t)) \cdot g(t)$.
  - Put all variables on one side and integrate:

$$\frac{dx}{x} = atdt \Rightarrow \int \frac{dx}{x} = \int atdt \Rightarrow \log(x) = \frac{at^2}{2} + c.$$

$$x(t) = \underbrace{e^c}_{=c'} \cdot e^{\frac{at^2}{2}}$$

  - There are **infinitely** many solutions parametrized by c'.
  - If we specify **initial condition** $x(0) = 2 \Rightarrow$ uniquely solvable: $x(t) = 2e^{\frac{at^2}{2}}$

ODE example: linear+homogenous (1)

- Let's consider:

$$\frac{d^2x(t)}{dt^2} - 3\frac{dx(t)}{dt} - 4x(t) = 0$$

  - which is of **second order**, **linear**, and **homogeneous**
- We use the ansatz $Ce^{\lambda t}$. The characteristic equation is:

$$\lambda^2 - 3\lambda - 4 = 0$$

- We obtain: $x(t) = C_1e^{4t} + C_2e^{-t}$
- Specifying $x(0) = 1$ and $\frac{dx(0)}{dt} = 0$ we obtain

$$C_1 + C_2 = 1$$

$$4C_1 - C_2 = 0$$

yielding $x(t) = \frac{1}{5}e^{4t} + \frac{4}{5}e^{-t}$.

ODE example: linear+homogenous (2)

$$\frac{d^2x(t)}{dt^2} - 8\frac{dx(t)}{dt} + 20x(t) = 0$$

- Same ansatz. Characteristic equation: $\lambda^2 - 8\lambda + 20 = 0$
- Solving it we obtain: $x(t) = C_1 e^{(4+2i)t} + C_2 e^{(4-2i)t}$.
- Can be simplified (Euler formula, only real solutions): $x(t) = C_1 e^{4t}\cos(2t) + C_2 e^{4t}\sin(2t)$.

$$\frac{d^2x(t)}{dt^2} - 8\frac{dx(t)}{dt} + 16x(t) = 0$$

- The characteristic equation is:

$$\lambda^2 - 8\lambda + 16 = (\lambda - 4)^2 = 0$$

- In this case: 4 is root of order two, and solution is:

$$x(t) = C_1 e^{4t} + C_2 t e^{4t}$$

Systems of linear ODEs (1)

- Consider again

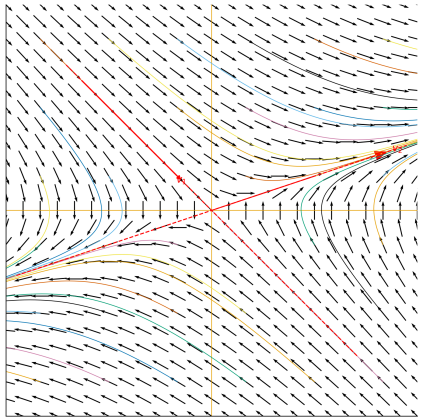
$$\frac{d^2x(t)}{dt^2} - 3\frac{dx(t)}{dt} - 4x(t) = 0 \tag{1}$$

- Now we introduce the vector $\boldsymbol{u}(t) = (x(t), x'(t))$
- Then solving (1) is equivalent to solving the following **first order system**:

$$\boldsymbol{u}'(t) = \boldsymbol{A}\boldsymbol{u}(t) = \begin{pmatrix} 0 & 1 \\ 4 & 3 \end{pmatrix} \boldsymbol{u}(t)$$

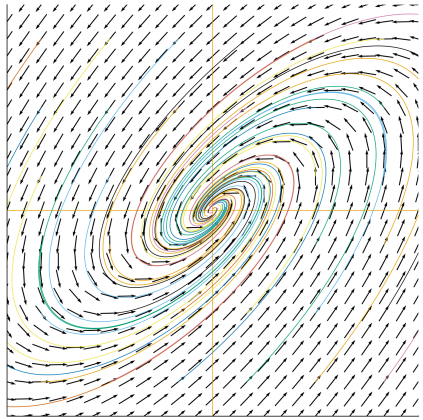
- **Eigenvalues** $\lambda_{1,2}$ of $\boldsymbol{A}$ are zeros of characteristic equation.
- $\boldsymbol{u}(t) = C_1 e^{\lambda_1 t}\boldsymbol{v}_1 + C_1 e^{\lambda_2 t}\boldsymbol{v}_2$, where $\boldsymbol{v}_1, \boldsymbol{v}_2$: **eigenvectors.**
- Eigenvalues and -vectors of $\boldsymbol{A}$ fully determine qualitative behavior.

Systems of linear ODEs (2)



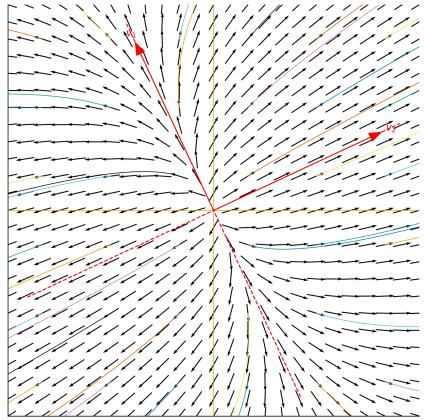
Saddle $\lambda_1 > 0$,

$\lambda_2 < 0$

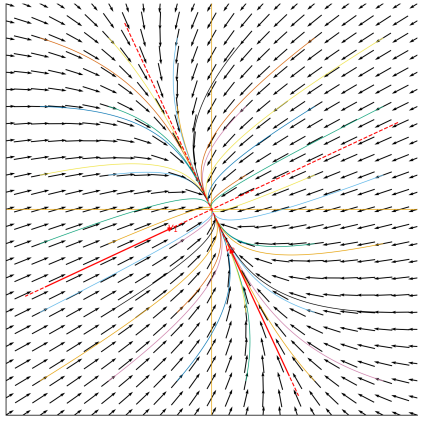


Stable spiral

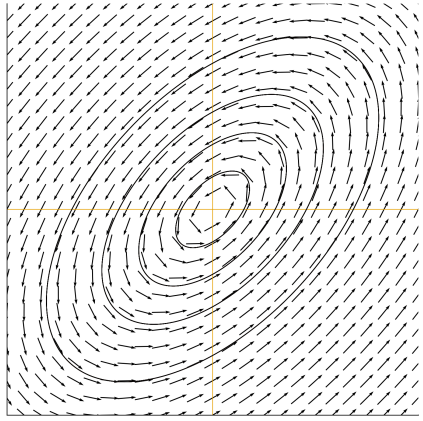
$\Re(\lambda_{1,2}) < 0$



Source $\lambda_1, \lambda_2 > 0$

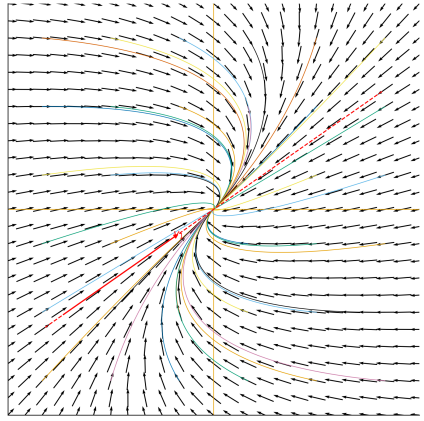


Sink $\lambda_1, \lambda_2 < 0$



Center

$\Re(\lambda_{1,2}) = 0$



Degenerate node

$\lambda_1 = \lambda_2 < 0$

Inhomogenous linear ODE

$$\frac{d^2x(t)}{dt^2} - 3\frac{dx(t)}{dt} - 4x(t) = 3e^{2t}$$

- First solve homogeneous equation: $x_h(t) = C_1e^{4t} + C_2e^{-t}$
- Second, find particular solution. Form is chosen such that the exponential will cancel out: $x_p(t) = Ae^{2t}$
- This leads to: $4A - 6A - 4A = 3$, thus: $A = -\frac{1}{2}$.
- General solution: sum of homogeneous and particular: $x(t) = x_h(t) + x_p(t) = C_1e^{4t} + C_2e^{-t} - \frac{1}{2}e^{2t}$

Existence and Uniqueness

- Let's consider:

$$\frac{d\boldsymbol{x}(t)}{dt} = \mathbf{F}(\boldsymbol{x}(t), t) \ , \qquad \boldsymbol{x}(t_0) = \boldsymbol{x}_0$$

- Integration yields:

$$\boldsymbol{x}(t) = \boldsymbol{x}_0 + \int_{t_0}^{t} \mathbf{F}(\boldsymbol{x}(\tau), \tau) d\tau$$

- Picard's iteration: Starting from $\psi_0(t) = \boldsymbol{x}_0$, we obtain:

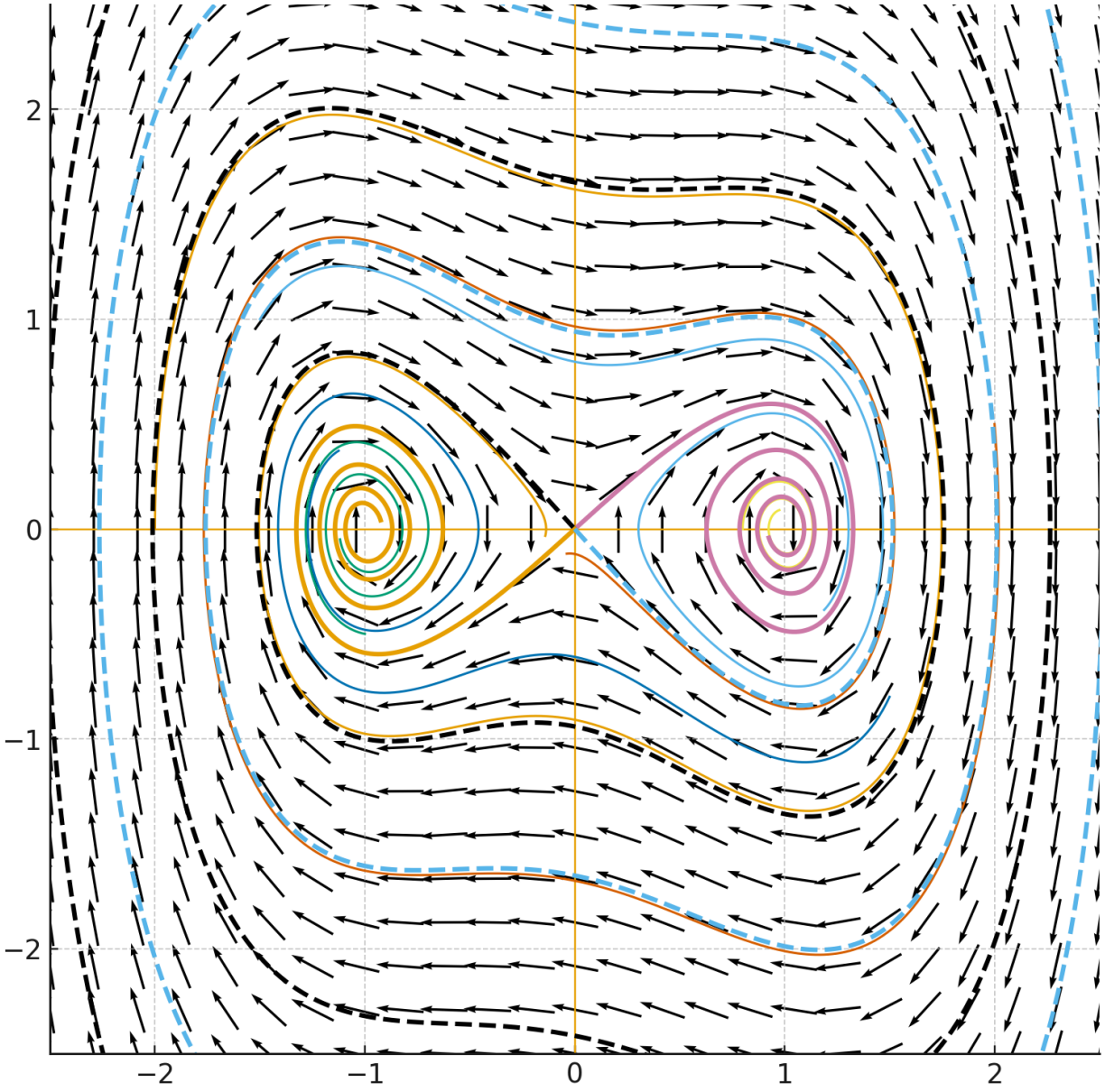
$$\psi_{n+1}(t) = \boldsymbol{x}_0 + \int_{t_0}^{t} \mathbf{F}(\psi_n(\tau), \tau) d\tau \ ,$$

- Converges to unique solution $\lim_{n\to\infty} \psi_n(t) = \boldsymbol{x}(t)$ if $\mathbf{F}(\boldsymbol{x}, t)$ is continuous in both arguments and locally Lipschitz continuous in the first argument.
- Implication: ODE has unique solution at interval around t_0.

Stability and asymptotics (1)

- Further basic questions:
  - Do small changes in initial condition lead to small changes in solution?
  - If solution exists for all times: what about behavior at $t \to \infty$ for a given initial condition $\boldsymbol{x}_0$?
- Often interesting/insightful, since explicit solutions only available in special cases.
- For autonomous systems: care about stability of fixed points $\mathbf{F}(\boldsymbol{x}_0) = 0$ (no dynamics there).
- A lot of information can be read of from Jacobian $\nabla \mathbf{F}(\boldsymbol{x}_0)$: eigenvalues determine behavior near x_0: similar to **linear system**.

Stability and asymptotics (2)



Numerical solutions

- Recall:

$$\boldsymbol{x}(t+\Delta t) = \boldsymbol{x}(t) + \int_t^{t+\Delta t} \mathbf{F}(\boldsymbol{x}(\tau), \tau) d\tau$$

- If we knew how to compute the integral on the right hand side, we could generate the solution at time steps t_0, $t_1 = t_0 + \Delta t$, $t_2 = t_0 + 2\Delta t$:

$$\boldsymbol{x}(t_0+\Delta t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t_0+\Delta t} \mathbf{F}(\boldsymbol{x}(\tau), \tau) d\tau$$

$$\boldsymbol{x}(t_0+2\Delta t) = \boldsymbol{x}(t_0+\Delta t) + \int_{t_0+\Delta t}^{t_0+2\Delta t} \mathbf{F}(\boldsymbol{x}(\tau), \tau) d\tau$$

$$\boldsymbol{x}(t_0+3\Delta t) = \boldsymbol{x}(t_0+2\Delta t) + \int_{t_0+2\Delta t}^{t_0+3\Delta t} \mathbf{F}(\boldsymbol{x}(\tau), \tau) d\tau$$

Euler method

- For Euler integration: pick a step-size $h > 0$, a number of steps n, and define:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\mathbf{F}(\boldsymbol{x}_n, t_n)$$

- Most basic numerical integrator with the integration scheme: $\int_t^{t+\Delta t} \mathbf{F}(\boldsymbol{x}(\tau), \tau) d\tau \approx \mathbf{F}(\boldsymbol{x}(t), t)\Delta t$
- Intuition: evolve the trajectories by iteratively taking small steps in the direction of slope

Euler method derivation

- Mathematical argument: forward finite difference formula for the derivative:

$$\frac{d\boldsymbol{x}(t_n)}{dt} = \lim_{h\to 0} \frac{\boldsymbol{x}(t_n + h) - \boldsymbol{x}(t_n)}{h} \approx \frac{\boldsymbol{x}(t_n + h) - \boldsymbol{x}(t_n)}{h}$$

- Rearranging terms:

$$\boldsymbol{x}(t_n + h) = \boldsymbol{x}(t_n) + h\frac{d\boldsymbol{x}(t_n)}{dt}$$

- Error estimate: Taylor expansion of the solution $\boldsymbol{x}(t_n)$ around t_{n+1}:

$$\boldsymbol{x}(t_{n+1}) = \boldsymbol{x}(t_n) + h\frac{d\mathbf{F}}{dt}\bigg|_{t_n} + O(h^2)$$

Runge Kutta method

- Euler method is the simplest numerical integrator in a family of methods known as Runge-Kutta methods
- For example, RK4 method (scalar-valued update formula):

$$x_{n+1} = x_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$$

With:

$$k_1 = \mathbf{F}(x_n, t_n)$$

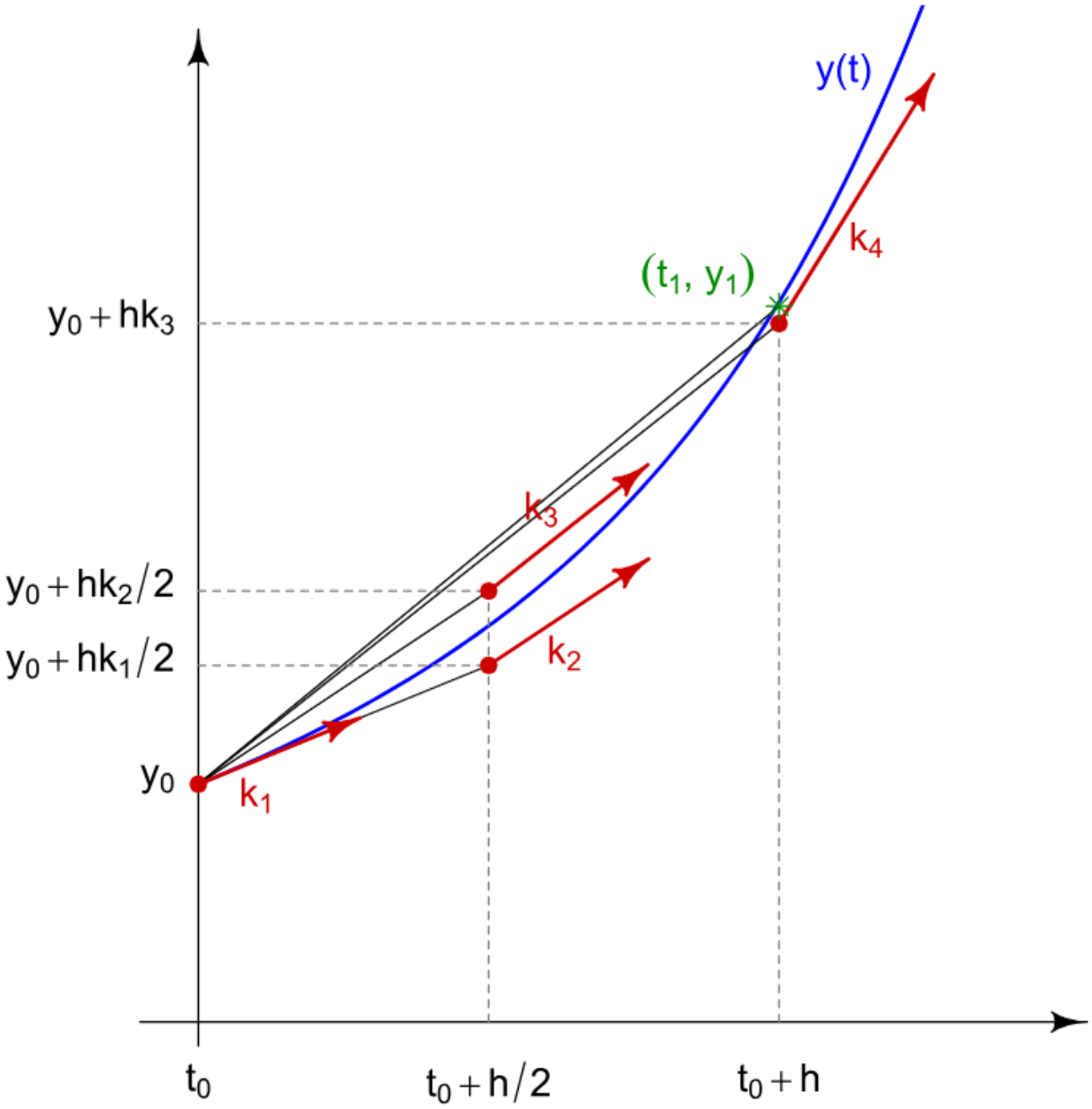
$$k_2 = \mathbf{F}(x_n + h\frac{k_1}{2}, t_n + \frac{h}{2})$$

$$k_3 = \mathbf{F}(x_n + h\frac{k_2}{2}, t_n + \frac{h}{2})$$

$$k_4 = \mathbf{F}(x_n + hk_3, t_n + h)$$

- Euler method is just RK4 but considering only k_1

Runge Kutta method



Implicit methods / adaptive step sizes

- Explicit vs implicit solvers:
  - An explicit method solves $\mathbf{F}(t + \Delta t) = \mathbf{G}(\mathbf{F}(t))$
  - An implicit method solves $\mathbf{G}(\mathbf{F}(t), \mathbf{F}(t + \Delta t)) = 0$
- Adaptive stepsize is often used to control errors when there is a large variation in the size of the derivative

PDEs

Partial Differential Equation (PDE)

- Here the unknown quantity is a **multivariate function** and the equation involves derivatives of the unknown function w.r.t. more variables, i.e., we need **partial derivatives**
- General expression for time-dependent PDEs:

$$\frac{\partial \boldsymbol{u}}{\partial t} = \boldsymbol{F}(t, \boldsymbol{x}, \boldsymbol{u}, \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}, \frac{\partial^2 \boldsymbol{u}}{\partial \boldsymbol{x}^2}, \ldots)$$

- Can also be classified, similar to ODEs.
- E.g., the Korteweg-de Vries equation:

$$\frac{\partial u(x,t)}{\partial t} - 6u(x,t)\frac{\partial u(x,t)}{\partial x} + \frac{\partial^3 u(x,t)}{\partial x^3} = 0$$

  - is a **time-dependent** PDE
  - is **non-linear** du to the term $6u(x,t)\frac{\partial u(x,t)}{\partial x}$
  - is **third-order**, because it contains the third derivative

Other PDEs

- The Burgers' equation:

$$\frac{\partial u(x,t)}{\partial t} = -u(x,t)\frac{\partial u(x,t)}{\partial x} + \nu\frac{\partial^2 u(x,t)}{\partial x^2}$$

  is again non-linear due to the term $u(x,t)\frac{\partial u(x,t)}{\partial x}$

- The Laplace equation (elliptic PDE):

$$\Delta u(\boldsymbol{x}) = \frac{\partial^2 u(\boldsymbol{x})}{\partial x^2} + \frac{\partial^2 u(\boldsymbol{x})}{\partial y^2} + \ldots = 0$$

- The heat equation (parabolic PDE):

$$\frac{\partial u(\boldsymbol{x},t)}{\partial t} = \Delta u(\boldsymbol{x},t)$$

- The wave equation (hyperbolic PDE):

$$\frac{\partial^2 u(\boldsymbol{x},t)}{\partial t^2} = c^2\Delta u(\boldsymbol{x},t)$$

Fundamental PDE terms

- Divergence: $\nabla \cdot \boldsymbol{u}(\boldsymbol{x}) = \begin{pmatrix} \partial_x \\ \partial_y \\ \partial_z \end{pmatrix} \cdot \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} = \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}$
- Gradient: $\nabla \boldsymbol{u}(\boldsymbol{x}) = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} = \left(\frac{\partial \boldsymbol{u}}{\partial x}, \frac{\partial \boldsymbol{u}}{\partial y}, \frac{\partial \boldsymbol{u}}{\partial z} \right)^\top$
- Convection of $\boldsymbol{v}$ along $\nabla \boldsymbol{u}$: $(\boldsymbol{v} \cdot \nabla)\boldsymbol{u}$
- Advection of $\boldsymbol{v}$ along ∇u: $(\boldsymbol{v} \cdot \nabla)u$
- Diffusion: $\Delta \boldsymbol{u} = \left(\frac{\partial^2}{\partial x^2}, \frac{\partial^2}{\partial y^2}, \frac{\partial^2}{\partial z^2} \right)^\top \boldsymbol{u}$
- Example: Navier-Stokes equations:

$$\frac{\partial \boldsymbol{u}}{\partial t} = \nu \Delta \boldsymbol{u} - (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \boldsymbol{f} - \nabla p$$
$$\nabla \cdot \boldsymbol{u} = 0$$

Boundary and initial conditions

- Important for time-dependent PDEs on domain $\mathbb{X}$ and time-interval $[0, T]$:
  - Specify **boundary conditions:**

$$B[\mathbf{u}](\mathbf{x}, t) = h(\mathbf{x}, t) \qquad (t, \mathbf{x}) \in [0, T] \times \partial\mathbb{X}$$

  - Specify **initial conditions:**

$$\mathbf{u}(\mathbf{x}, 0) = g(\mathbf{x})$$

- Solution properties, mathematical and numerical techniques largely depend on their structure.
- As example: we solve 1-dimensional **heat equation**:

$$u_t(x, t) = u_{xx}(x, t)$$

  - on the **interval** $\mathbb{X} = (0, 1)$ using **Fourier series**.
  - on the **whole domain** $\mathbb{X} = \mathbb{R}$ using **Fourier transform**.

Heat equation on $(0,1)$ (1): physical meaning

$$u_t(x,t) = u_{xx}(x,t) \quad \mathbb{X} = (0,1)$$

- $u(t,x)$: temperature of substance at point x, time t.
- Consider **flux function** f. Heat is conserved:

$$\frac{\partial}{\partial t}\int_0^1 u(x,t)dx = f(u(0,t)) - f(u(1,t))$$

$$\int_0^1 \frac{\partial u(x,t)}{\partial t} + f(u(x,t))_x dx = 0$$

- **Fick's law:** f proportional to temp. gradient: $f(u) = cu_x$.
- Specify **boundary conditions:** $u(0,t) = 0 = u(1,t)$ and initial conditions: $u(x,0) = g(x)$ for any $x \in (0,1)$.

Heat equation on $(0,1)$ (2): separation of variables

- **Separation of variables ansatz:** $u(x,t) = w(t)y(x)$.
  Plugging into heat equation yields:
  $$\frac{w'(t)}{w(t)} = \frac{y''(x)}{y(x)}.$$
- Holds for any t, x: must be equal to constant $-\kappa < 0$.
- Following relations (see ODE part):
  - $-w'(t) = \kappa w(t) \Rightarrow w(t) = c_1 e^{-\kappa t}$
  - $-y''(x) = \kappa y(x) \Rightarrow y(x) = c_2 \cos(\sqrt{\kappa}x) + c_3 \sin(\sqrt{\kappa}x)$
  - $y(0) = y(1) = 0$ implies $y(x) = \sin(x\sqrt{\kappa})$ and $\kappa = (n\pi)^2, n \in \mathbb{N}$.
- Gives family of solutions:
  $$u_n(x,t) = c_n e^{-(n\pi)^2 t} \sin(\pi n x) \quad n \in \mathbb{N}$$

Heat equation on $(0,1)$ (3): initial condition

- **Superposition principle:** finite linear combination of solutions again solutions.
- If $\sum_n |c_n| < \infty$: $u(x,t) = \sum_n c_n e^{-(n\pi)^2 t} \sin(\pi n x)$ is solution.
- Provided that $g(x) = \sum_n c_n \sin(\pi n x)$ and $\sum_n |c_n| < \infty$: u satisfies initial condition.
- Since $\sin(\pi n x)$ is orthogonal in L^2: $c_n = \int_0^1 g(y) \sin(\pi n x) dx$: **Fourier (sine) series**.
- Large class of functions admitting such a representation (Wiener algebra)
- Exercise: compute solution to heat equation for $g_1(x) = \sin(\pi x), g_2(x) = x(x-1)$

Towards full line: Fourier transforms

- Continuous generalization of Fourier series:

$$\hat{f}(k) = \mathcal{F}\{f\}(k) = \int_{-\infty}^{\infty} f(x)e^{-ikx}dx$$

- Corresponding inverse:

$$f(x) = \mathcal{F}^{-1}\{\hat{f}\}(x) = \mathcal{F}\{\hat{f}\}(-x) = \int_{-\infty}^{\infty} \hat{f}(k)e^{ikx}dk$$

- Important properties
  - Derivatives into multiplication by factor ik:

$$\mathcal{F}\{\frac{d^n f}{dx^n}\}(k) = (ik)^n \mathcal{F}\{f\}(k) \quad \mathcal{F}\{x^n f(x)\}(k) = i^n \frac{d^n}{dk^n}\mathcal{F}(f)(k)$$

  - Multiplication into convolution:

$$\mathcal{F}\{f \cdot g\}(k) = \frac{1}{2\pi}\mathcal{F}\{f\}(k) \star \mathcal{F}\{g\}(k)$$

  - Gaussian into rescaled Gaussian: Fix t:

$$\mathcal{F}\{e^{-x^2 t}\}(k) = \sqrt{\frac{\pi}{t}}e^{-\frac{k^2}{4t}}$$

Heat equation on $\mathbb{R}$ via Fourier transform

$$u_t(x,t) = u_{xx}(x,t) \quad \mathbb{X} = \mathbb{R}, \quad u(x,0) = g(x)$$

- Computing the FT wrt. x on both sides yields:

$$\partial_t \widehat{u}(k,t) = -k^2\, \widehat{u}(k,t), \qquad \widehat{u}(k,0) = \widehat{g}(k).$$

- Solving the ODE yields: $\widehat{u}(k,t) = e^{-k^2 t}\, \widehat{g}(k)$
- Invert the transform:

$$u(x,t) = \mathcal{F}^{-1}(e^{-k^2 t} \cdot \hat{g}(k))(x) = \frac{1}{2\pi}(G(k,t) \star g(k))(x),$$

- $G(k,x) = \mathcal{F}^{-1}(e^{-k^2 t})(x) = \sqrt{\frac{\pi}{t}} e^{-\frac{x^2}{4t}}$
- $u(x,t) = \frac{1}{\sqrt{4\pi t}} \int_{-\infty}^{\infty} e^{-\frac{k^2}{4t}}\, g(x-k)dk$

Exemplary numerical scheme

- $$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + f(x,t)$$
- Discretization:
  - Discretize space using points $x_i = i\Delta x$.
  - Discretize time using steps $t_n = n\Delta t$.
  - Let $u_i^n \approx u(x_i, t_n)$.
- Finite Difference Approximations:
  - $\frac{\partial u}{\partial t} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t}$ (Forward difference)
  - $\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}$ (Central difference)
- The Explicit Scheme:
  - Substitute the approximations into the PDE:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + f(x_i, t_n)$$

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{\Delta x^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) + \Delta t \cdot f(x_i, t_n)$$

Brownian motion, stochastic calculus and SDEs

Motivation: continuos random walks

- Consider Bernoulli RV X taking values ± 1, so that $P(X = 1) = P(X = -1) = \frac{1}{2}$.
- N-th position of random walk on the line, starting at 0: $S_N = \sum_{n=1}^{N} X_n$, where X_n: iid. copies of X.
- $\mathrm{E}(S_N) = 0$ and $\mathrm{Var}(S_N) = N$
- CLT implies: $S_N \sim \mathcal{N}(0, N)$ for sufficiently large N: **continuous approximation**.
- Can this be done for continuous timesteps $N = t$ as well? See animations.

Brownian motion (BM): intuition

- Imagine small dust particle floating on body of water
- Sensitivity to fluctuations: real movement from external forces would completely overtake the one of particle at any time-step.
- Some properties:
  - **Starting point** anchor process in space.
  - **Positional symmetry** For each step: expected displacement is zero.
  - **Independence** Steps at different times are independent. Displacement between two different intervals of time is independent.
  - **Continuity** No jumps or gaps
  - **Normality** As in discrete case: distribution of position at given time should be normal

BM: mathematical

- A BM is a collection of RVs $W(t)$, $t \geq 0$, so that:
  - $W(0) = 0$
  - $W(t) \sim \mathcal{N}(0, t)$
  - Consider $\Delta W(t_1, t_2) = W(t_2) - W(t_1)$. Then for $t_1 < t_2 < t_3$: $\Delta W(t_1, t_2)$ is independent from $\Delta W(t_2, t_3)$
- Continuity of $t \mapsto W(t)$ is implied by previous properties: more involved, technical result.
- Overall: $W(t)$ is continuous, random, zero-mean process with variance proportional to time.
- **Globally somewhat, locally not at all predictable!**

Intuition for Ito Calculus

- How to define integral wrt. $W(t)$?
- W is **nowhere differentiable:** consider small increment dt and $dW = \Delta W(t, t+dt) \sim \sqrt{dt}\mathcal{N}(0,1)$. Then:

$$\frac{dW}{dt} = \frac{\mathcal{N}(0,1)}{\sqrt{dt}}$$

- As $dt \to 0$: diverging, dominated by random fluctuations
- **Rules out standard calculus $\Rightarrow$ Ito Calculus**
- $\mathrm{E}(dW) = 0$, $\mathrm{Var}(dW) = \mathrm{E}((dW)^2) = dt$,
- $\mathrm{Var}((dW)^2) = 2(dt)^2$: follows from $\mathrm{Var}(\mathcal{N}(0,1)^2) = 2$, can be neglected as $dt \to 0$.
- In standard calculus: $(dt)^2$ is too small to matter, however, in Ito-calculus: $(dW)^2 = dt$, at least formally.

Heuristic for Ito Integral

- Consider partition $0 = s_0 < \cdots < s_n = t$ of $[0, t]$.
- Regular calculus:

$$\int_0^t f(s)ds \approx \sum_{i=0}^{n-1} f(s_i)(s_{i+1} - s_i)$$

- Ito calculus:

$$\int_0^t f(s)dW(s) \approx \sum_{i=0}^{n-1} f(s_i)\Delta W(s_i, s_{i+1})$$

  - Result is a random variable!
  - Evaluation of f always at left endpoint.
  - Limits can be made precise.

Heuristic for Ito's Lemma

- Consider $f(t, W(t))$. Regular calculus yields:

$$df = \frac{\partial f}{\partial t} dt + \frac{\partial f}{\partial W} dW$$

- Roughness of Brownian motion requires second order term. Taylor expand f:

$$\begin{aligned} df = & \frac{\partial f}{\partial t} dt + \frac{\partial f}{\partial W} dW \\ & + \frac{1}{2} \left(\frac{\partial^2 f}{\partial W^2} (dW)^2 + 2 \frac{\partial^2 f}{\partial W \partial t} dW\, dt + \frac{\partial^2 f}{\partial t^2} (dt)^2 \right) \end{aligned}$$

- $dW\, dt$ and $(dt)^2$ vanish as $dt \to 0$, however $(dW)^2 = dt$ stays significant, yielding:

$$df = \frac{\partial f}{\partial t} dt + \frac{\partial f}{\partial W} dW + \frac{1}{2} \frac{\partial^2 f}{\partial W^2} dt$$

A Stochastic Differential Equation (SDE)

- ... is an equation, describing how a system evolves when subjected to a combination of deterministic and **fluctuating** ("random") forces
- General structure:

$$dX(t) = \mu(t, X(t))dt + \sigma(t, X(t))dW(t) \tag{2}$$

  - has an evolving term X
  - has a deterministic term μdt (**drift**)
  - has a stochastic term $\sigma dW(t)$ (**diffusion**)

Ito's Lemma revisited

- Ito's Lemma can also be applied to $f(t, X(t))$ with more general $X(t)$ (instead of only $W(t)$).
- Observing from (2) that $dX = \mathcal{O}(dW)$: only consider terms up to $(dX)^2 = \mathcal{O}((dW)^2)$. Then:

$$\begin{aligned} df &= \frac{\partial f}{\partial t}dt + \frac{\partial f}{\partial X}dX + \frac{1}{2}\frac{\partial^2 f}{\partial X^2}(dX)^2 \\ &= \frac{\partial f}{\partial t}dt + \frac{\partial f}{\partial X}(\mu dt + \sigma dW) + \frac{1}{2}\frac{\partial^2 f}{\partial X^2}(\mu dt + \sigma dW)^2 \\ &= (\frac{\partial f}{\partial t} + \frac{\partial f}{\partial X}\mu + \frac{1}{2}\frac{\partial^2 f}{\partial X^2}\sigma^2)dt + \frac{\partial f}{\partial X}\sigma dW \end{aligned}$$

  - First equation: Ito's Lemma as above.
  - Second equation: Insert (2).
  - Last equation: $dW\,dt$ and $(dt)^2$ are neglected, $(dW)^2 = dt$

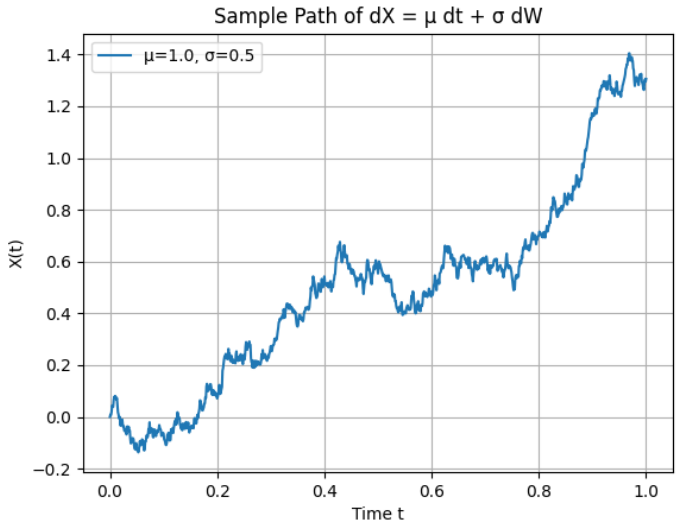
SDE example 1: constant drift/diffusion

$$dX(t) = \mu dt + \sigma dW(t)$$

- Starting at $X(0) = 0$ and integrating yields:

$$X(t) = \int_0^t \mu dt + \int_0^t \sigma dW(t) = t\mu + \sigma W(t)$$

- Since $W(t) \sim \mathcal{N}(0, t)$: $X(t) \sim \mathcal{N}(\mu t, \sigma^2 t)$
- Drifts linearly with noise spreading over time
- Basic model e.g. for stock with steady growth and volatility



SDE example 2: geometric Brownian motion

$$dX(t) = \mu X(t)dt + \sigma X(t)dW(t)$$

- $X(t)$: **state**, $\mu X(t)$: **prop. drift**, $\sigma X(t)$: **prop. noise**
- Apply Ito with $f = \log(X)$, $\frac{\partial f}{\partial t} = 0$, $\frac{\partial f}{\partial X} = \frac{1}{X}$, $\frac{\partial^2 f}{\partial X^2} = \frac{-1}{X^2}$:

$$d(\log(X)) = (\frac{\mu X}{X} - \frac{(X\sigma)^2}{2X^2})dt + \frac{\sigma X}{X}dW = (\mu - \frac{\sigma^2}{2})dt + \sigma dW$$

- Integrate:

$$\log(X(t)) = \log(X(0)) + (\mu - \frac{\sigma^2}{2})t + \sigma W(t)$$

$$X(t) = X(0)\exp((\mu - \frac{\sigma^2}{2})t + \sigma W(t))$$

- Drift is adjusted by $-\frac{\sigma^2}{2}$ due to second-order effect of noise, $\sigma W(t)$ adds fluctuations. See plot/simulation.
- Underlying principle of Black-Scholes model in finance.

Basic questions (see ODEs, PDEs)

- Similar existence result as for ODEs. Ofc. more delicate, as precise introduction of Ito integral is technical.
- However: as for ODEs/PDEs: not many SDEs explicitly solvable.
- Rely again on numerical tools, e.g. stepping:

$$X(t+\Delta t) = X(t) + \mu \Delta t + \sigma \sqrt{\Delta t} \mathcal{N}(0,1)$$

- And or: statistical analysis of evolution of **particle density** $p(x,t)$ by relating with PDEs, mainly **Focker-Planck equation** (here in 1-d):

$$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[a(x,t)\, p(x,t)\right] + \frac{1}{2}\frac{\partial^2}{\partial x^2}\left[b^2(x,t)\, p(x,t)\right].$$

Differential equations for building DL architectures

Residual networks

- Remember Euler integration with step size $h > 0$, and number of steps n:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + h\boldsymbol{f}(\boldsymbol{x}_n, t_n) \quad \Leftrightarrow \quad \boldsymbol{x}_{l+1} = \boldsymbol{x}_l + f(\boldsymbol{x}_l, \boldsymbol{\theta}_l)$$

- Compare to residual mappings from layer l to layer $l+1$:
  - Let f be a bijective mapping that transforms a layer's input $\boldsymbol{x}_l$ into an output $\boldsymbol{x}_{l+1}$
  - Consider X_l and X_{l+1} as random variables connected via $X_{l+1} = f(X_l) \Rightarrow$ entropy $\mathcal{H}$ between X_l and X_{l+1}:

$$\mathcal{H}(X_{l+1}) \leq \mathcal{H}(X_l) + \mathbb{E}_{p(X_l)}\left(\log\left|\frac{\partial X_{l+1}}{\partial X_l}\right|\right)$$

  - Whenever the determinant of the Jacobian is smaller than 1, information will be lost in the transformation, and we will observe vanishing gradients
    - The key (deep) learning is to ensure: $\left|\frac{\partial X_{l+1}}{\partial X_l}\right| = 1$

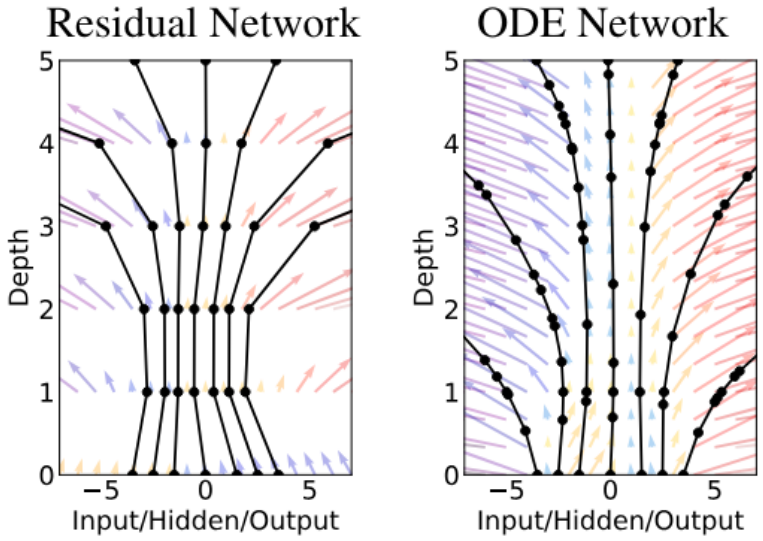
Neural ODE

- In Neural ODEs, the update $\boldsymbol{x}_{l+1} = \boldsymbol{x}_l + f(\boldsymbol{x}_l, \boldsymbol{\theta}_l)$ is treated as (Euler) discretization of:

$$\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x}, t, \boldsymbol{\theta})$$

- We end up, solving an ODE in the forward pass:

$$\begin{aligned}\boldsymbol{x}(t + \Delta t) &= \boldsymbol{x}(t) + \int_t^{t+\Delta t} \boldsymbol{f}(\boldsymbol{x}(\tau), \tau, \boldsymbol{\theta}) d\tau \\ &= \text{ODESolve}(\boldsymbol{x}, \boldsymbol{f}, t, t + \Delta t, \boldsymbol{\theta})\end{aligned}$$



Neural ODE

- To use the same notation as the original paper, we rewrite the ODE forward pass w.r.t to the scalar loss function L to:

$$L(\boldsymbol{z}(t_1)) = L\left(\boldsymbol{z}(t_0) + \int_{t_0}^{t_1} f(\boldsymbol{z}(t), t, \boldsymbol{\theta})dt\right)$$
$$= L\left(\text{ODESolve}(\boldsymbol{z}(t_0), f, t_0, t_1, \boldsymbol{\theta})\right)$$

```
# Set the time sample grid.
T = 100.
t = np.linspace(0, T, 200)

from scipy.integrate import solve_ivp
sol_ps = solve_ivp(fun=f,
                   t_span=[t[0], t[-1]],
                   y0=u0,
                   method='Radau',
                   t_eval=t,
                   args=(L,),
                   atol=tol,
                   rtol=tol)
```

Remember: delta errors

- The gradient descent update is given by
  $\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{\text{old}} - \eta \nabla_{\boldsymbol{\theta}} R_{\text{emp}}$
- We select a weight θ_{ij} in an arbitrary layer that connects from the unit j to the unit i by

$$\frac{\partial L}{\partial \theta_{ij}} = \frac{\partial L}{\partial h_i} \frac{\partial h_i}{\theta_{ij}} ,$$

  where the δ-errors are defined as $\delta_i = \dfrac{\partial L}{\partial h_i}$, i.e., the derivative of the loss w.r.t the pre-activations h_i
- The gradient of the weights is obtained via:

$$\frac{\partial L}{\partial \theta_{ij}} = \delta_i \frac{\partial h_i}{\theta_{ij}} = \underbrace{\delta_i s_j}_{\delta \text{ times activations of previous layer}}$$

- The δ-error at units not in the output layer, i.e. hidden layers, are defined as $\delta_j = \sum_i \frac{\partial L}{\partial h_i} \frac{h_i}{h_j}$

Adjoint method

- To optimize $L\,(\text{ODESolve}(\boldsymbol{z}(t_0), f, t_0, t_1, \theta)$, we require gradients with respect to θ
- As with conventional NNs the analogue to computing these gradients are the deltas, i.e., the derivative of the loss with respect to the activations $\boldsymbol{z}(t)$, i.e., $\dfrac{\partial L}{\partial \boldsymbol{z}(t)}$
- For Neural ODEs, this results in another ODE:

$$\frac{d}{dt}\frac{\partial L}{\partial \boldsymbol{z}(t)} = -\left(\frac{\partial L}{\partial \boldsymbol{z}(t)}\right)^{\top} \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}$$

- Substituting $\boldsymbol{a} = \dfrac{\partial L}{\partial \boldsymbol{z}(t)}$, we obtain:

$$\frac{d\boldsymbol{a}^{\top}(t)}{dt} = -\boldsymbol{a}^{\top}(t)\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}$$

Gradients w.r.t θ

- Computing the gradients with respect to the parameters θ requires evaluating a third integral, which depends on both $\boldsymbol{z}(t)$ and $\boldsymbol{a}(t)$
- The gradients with respect to the parameters θ follow another ODE:

$$\frac{d}{dt}\frac{\partial L}{\partial \boldsymbol{\theta}} = -\boldsymbol{a}^\top(t)\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

- which is again solved by integration:

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -\int_{t_1}^{t_0} \boldsymbol{a}^\top(t)\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

Augmented dynamics

- A training step contains:
  - Forward pass and loss calculation via $L\left(\text{ODESolve}(\boldsymbol{z}(t_0), f, t_0, t_1, \boldsymbol{\theta}\right)$
  - Backward pass with respect to the activations, following the ODE: $\frac{d}{dt}\frac{\partial L}{\partial \boldsymbol{z}(t)} = -\frac{\partial L}{\partial \boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} \Rightarrow$ starting from $\frac{\partial L}{\partial \boldsymbol{z}(t_1)}$ and working the way backwards
  - Computing the gradients w.r.t to $\boldsymbol{\theta}$, following $\frac{\partial L}{\partial \boldsymbol{\theta}} = -\int_{t_1}^{t_0} \boldsymbol{a}^\top(t)\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}$ starting again at t_1
- Problem: If we use e.g., implicit solvers, we (quite likely) have different time steps in the forward then in the backword ODE
  - Solution: We calculate $\boldsymbol{z}(t_1)$ and $\frac{\partial L}{\partial \boldsymbol{z}(t_1)}$ and work our way backwards $\Rightarrow$ each of the ODEs is solved backwards

Long Short-Term Memory

- RNNs such as the LSTM result in very deep neural networks when unrolled in time
- LSTMs are networks where each cell state stores information:
  - Read, write and delete access to this information is controlled via gates
  - LSTM is an ODE integrator over time!

$$\boldsymbol{c}_t = \boldsymbol{c}_{t-1} + \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

Long Short-Term Memory

- LSTM is an ODE integrator over time!

$$\boldsymbol{c}_t = \boldsymbol{c}_{t-1} + \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

$$\boldsymbol{c}_t = \mathbf{f}_t \odot \boldsymbol{c}_{t-1} + \mathbf{i}_t \odot \boldsymbol{z}_t \qquad \text{cell state}$$

$$\boldsymbol{h}_t = \mathbf{o}_t \odot \tilde{\boldsymbol{h}}_t\,, \qquad \tilde{\boldsymbol{h}}_t = \psi\left(\boldsymbol{c}_t\right) \qquad \text{hidden state}$$

$$\boldsymbol{z}_t = \varphi\left(\tilde{\boldsymbol{z}}_t\right)\,, \qquad \tilde{\boldsymbol{z}}_t = \boldsymbol{W}_{\boldsymbol{z}}\,\boldsymbol{x}_t + \boldsymbol{R}_{\boldsymbol{z}}\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\boldsymbol{z}} \qquad \text{cell input}$$

$$\mathbf{i}_t = \sigma\left(\tilde{\mathbf{i}}_t\right)\,, \qquad \tilde{\mathbf{i}}_t = \boldsymbol{W}_{\mathbf{i}}\,\boldsymbol{x}_t + \boldsymbol{R}_{\mathbf{i}}\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\mathbf{i}} \qquad \text{input gate}$$

$$\mathbf{f}_t = \sigma\left(\tilde{\mathbf{f}}_t\right)\,, \qquad \tilde{\mathbf{f}}_t = \boldsymbol{W}_{\mathbf{f}}\,\boldsymbol{x}_t + \boldsymbol{R}_{\mathbf{f}}\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\mathbf{f}} \qquad \text{forget gate}$$

$$\mathbf{o}_t = \sigma\left(\tilde{\mathbf{o}}_t\right)\,, \qquad \tilde{\mathbf{o}}_t = \boldsymbol{W}_{\mathbf{o}}\,\boldsymbol{x}_t + \boldsymbol{R}_{\mathbf{o}}\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\mathbf{o}} \qquad \text{output gate}$$

Diffusion models

- Remember SDEs, e.g., the Langevin equation:
  $$m\frac{d\boldsymbol{v}(t)}{dt} = -\lambda\boldsymbol{v}(t) + \boldsymbol{\eta}(t)$$
- Without deterministic part, we obtain Brownian motion as used in the forward diffusion process:
  $$\boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \beta_t\boldsymbol{\eta}_t \ ,$$
  - $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_T$ are i.i.d. samples from $\mathcal{N}(0, I)$
- ResNets/Transformers are **ODE integrators over blocks**, forward diffusion is an **SDE integrator over the input**