Partial differential equations

= PDE

- *Numerical Solution of Partial Differential Equations*, K.W. Morton and D.F. Mayers (Cambridge Univ. Press, 1995)
- *Numerical Solution of Partial Differential Equations in Science and Engineering*, L. Lapidus and G.F. Pinder (Wiley, 1999)
- *Finite Difference Schemes and Partial Differential Equations*, J.C. Strikwerda (Wadsworth, Belmont, 1989)

Examples for PDEs

examples for scalar boundary value problems (elliptic eqs.)

field Φ depends on $\vec{x}$

Poisson equation:

$$\Delta\Phi = \rho(\vec{x}), \qquad \Phi(\Gamma) = \Phi_0$$

Dirichlet boundary condition

Laplace equation:

$$\Delta\Phi = 0, \qquad \nabla_{\mathbf{n}}\Phi(\Gamma) = \Psi_0$$

von Neuman boundary condition

Examples for PDEs

example: vectorial boundary value problem

$\vec{u}(\vec{x})$ is a vector field defined on space

$$\boxed{\vec{\nabla}(\vec{\nabla}\vec{u}(\vec{x})) + (1-\nu)\Delta\vec{u}(\vec{x}) = 0}$$

Lamé equation of elasticity

(elliptic eq.)

Examples for PDEs

wave equation

$$\boxed{\Phi(\vec{x},t)}$$

$$\frac{\partial^2\Phi}{\partial t^2} = c^2\Delta\Phi \ , \qquad \Phi(\vec{x},t_0) = \widetilde{\Phi}_0(\vec{x})$$

diffusion equation $\Phi(\Gamma,t) = \Phi_0(t)$

$$\frac{\partial\Phi}{\partial t} = \kappa\,\Delta\Phi$$

initial boundary problem

Examples

$\vec{v}(\vec{x},t)$ vect … nd time

$$\begin{pmatrix} v_x \frac{\partial v_x}{\partial x} + v_y \frac{\partial v_x}{\partial y} + v_z \frac{\partial v_x}{\partial z} \\ v_x \frac{\partial v_y}{\partial x} + v_y \frac{\partial v_y}{\partial y} + v_z \frac{\partial v_y}{\partial z} \\ v_x \frac{\partial v_z}{\partial x} + v_y \frac{\partial v_z}{\partial y} + v_z \frac{\partial v_z}{\partial z} \end{pmatrix}$$

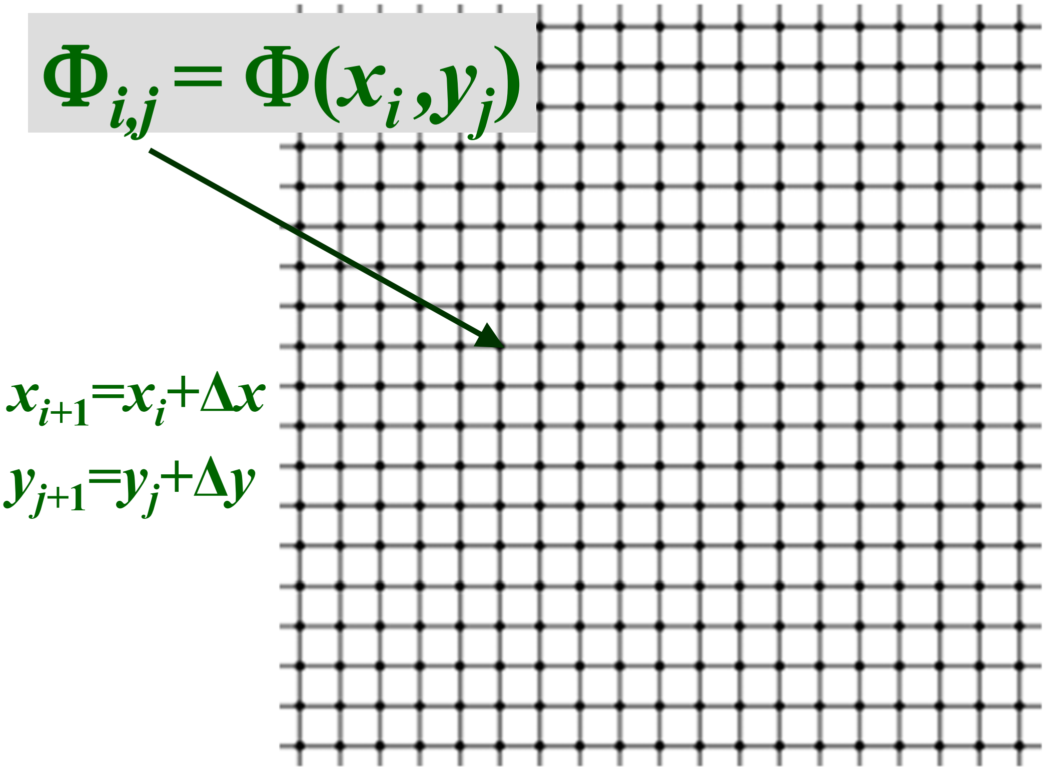
$$\frac{\partial \vec{v}}{\partial t} + \overbrace{(\vec{v}\vec{\nabla})}\vec{v} = -\frac{1}{\rho}\nabla p + \mu\,\Delta\vec{v}\ , \quad \vec{\nabla}\vec{v} = 0$$

$$\vec{v}(\vec{x},t_0) = \vec{V}_0(\vec{x})\ , \quad p(\vec{x},t_0) = P_0(\vec{x})$$

$$\vec{v}(\Gamma,t) = \vec{\mathrm{v}}_0(t)\ , \quad p(\Gamma,t) = \mathrm{p}_0(t)$$

Navier – Stokes eq. for fluid motion

Discretization of space



$\Phi_{i,j} = \Phi(x_i, y_j)$

$x_{i+1} = x_i + \Delta x$

$y_{j+1} = y_j + \Delta y$

Finite Difference Method

Discretization of derivatives

$$\Delta x \ \text{ small} \quad , \qquad x_n = n \cdot \Delta x$$

first derivative in 1d

$$\frac{\partial \Phi}{\partial x} = \frac{\Phi(x_{n+1}) - \Phi(x_n)}{\Delta x} + O(\Delta x)$$

$$= \frac{\Phi(x_n) - \Phi(x_{n-1})}{\Delta x} + O(\Delta x)$$

$$= \frac{\Phi(x_{n+1}) - \Phi(x_{n-1})}{2\Delta x} + O(\Delta x^2)$$

Discretization of derivatives

second derivative in one dimension

$$\boxed{\frac{\partial^2 \Phi}{\partial x^2} = \frac{\Phi(x_{n+1}) + \Phi(x_{n-1}) - 2\Phi(x_n)}{\Delta x^2} + O(\Delta x^2)}$$

or better

$$\boxed{\frac{\partial^2 \Phi}{\partial x^2} = \frac{-\Phi(x_{n-2}) + 16\,\Phi(x_{n-1}) - 30\,\Phi(x_n) + 16\,\Phi(x_{n+1}) - \Phi(x_{n+2})}{12\,\Delta x^2} + O(\Delta x^4)}$$

Discretization of derivatives

insert in $\boxed{\frac{\partial^i \Phi}{\partial x^i} = \frac{1}{\Delta x^i} \sum_{k=-l}^{l} a_k \Phi(x_{n+k})}$ **Taylor expansion:**

$$\Phi(x_{n+k}) = \Phi(x_n) + k\Delta x \frac{\partial \Phi}{\partial x}(x_n) + \frac{k^2}{2} \Delta x^2 \frac{\partial^2 \Phi}{\partial x^2}(x_n) + \frac{k^3}{6} \Delta x^3 \frac{\partial^3 \Phi}{\partial x^3}(x_n) + O(\Delta x^4)$$

$i = 2 \Rightarrow$

$$\frac{\partial^2 \Phi}{\partial x^2} = \frac{-\Phi(x_{n-2}) + 16\Phi(x_{n-1}) - 30\Phi(x_n) + 16\Phi(x_{n+1}) - \Phi(x_{n+2})}{12\Delta x^2} + O(\Delta x^4)$$

third derivative

$$\boxed{\frac{\partial^3 \Phi}{\partial x^3} = \frac{-\Phi(x_{n-2}) + 2\,\Phi(x_{n-1}) - 2\,\Phi(x_{n+1}) + \Phi(x_{x+2})}{\Delta x^3} + O(\Delta x^2)}$$

Derivatives in higher dimension

Be $\Delta x = \Delta y = \Delta z$.

2 d

$$\boxed{\begin{aligned} \Delta\Phi\ \Delta x^2 \quad &= \quad \Phi(x_{n+1}, y_n) + \Phi(x_{n-1}, y_n) \\ &+ \Phi(x_n, y_{n+1}) + \Phi(x_n, y_{n-1}) - 4\,\Phi(x_n, y_n) \end{aligned}}$$

3 d

$$\boxed{\begin{aligned} \Delta\Phi\ \Delta x^2 \quad = \quad &\Phi(x_{n+1}, y_n, z_n) \\ &+ \Phi(x_{n-1}, y_n, z_n) + \Phi(x_n, y_{n+1}, z_n) + \Phi(x_n, y_{n-1}, z_n) \\ &+ \Phi(x_n, y_n, z_{n+1}) + \Phi(x_n, y_n, z_{n-1}) - 6\,\Phi(x_n, y_n, z_n) \end{aligned}}$$

Poisson equation

$$\boxed{\Delta\Phi(\vec{x}) = \rho(\vec{x})}$$

discretize one-dimensional space by x_n , $n = 1,\ldots,N$

be $\boxed{\Phi_n \equiv \Phi(x_n)}$

discretization of the Poisson equation:

$$\boxed{\Phi_{n+1} + \Phi_{n-1} - 2\,\Phi_n = \Delta x^2 \cdot \rho(x_n)}$$

Dirichlet boundary conditions: $\boxed{\Phi_0 = c_0 \text{ and } \Phi_N = c_1}$

⇒ System of *N*-1 coupled linear equations

Poisson equation in 1d

example: chain of $N = 5$ with $\rho = 0$

and Dirichlet boundary conditions

$$\boxed{\begin{pmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{pmatrix} \cdot \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{pmatrix} = -\begin{pmatrix} c_0 \\ 0 \\ 0 \\ c_1 \end{pmatrix}}$$

Poisson equation in 2d

two-dimensional discretized equation on grid $L \times L$:

$(\Delta x = \Delta y)$

$$\boxed{\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\,\Phi_{i,j} = \Delta x^2 \rho_{i,j}}$$

replace indices i and j by $k = i + (j\,\text{-}1)\,(L\text{-}2)$

$$\boxed{\Phi_{k+1} + \Phi_{k-1} + \Phi_{k+L-2} + \Phi_{k-L+2} - 4\,\Phi_k = \Delta x^2 \rho_k}$$

⇒ System of $N = (L\text{-}2)^{\,2}$ coupled linear equations:

$$\boxed{\overset{\leftrightarrow}{A} \cdot \vec{\Phi} = \vec{b}}$$

Laplace equation in 2d

Example 5 × 5 lattice with $\rho = 0$ and $\Phi_m = \Phi_0$ for all $m \in \Gamma$, i.e. Dirichlet boundary condition with fixed Φ_0 on Γ.

$$\boxed{\begin{pmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & & & (L\text{-}2)^2 \times (L\text{-}2)^2\ \ \text{matrix} & & & & & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{pmatrix} \cdot \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ \Phi_7 \\ \Phi_8 \\ \Phi_9 \end{pmatrix} = - \begin{pmatrix} 2 \\ 1 \\ 2 \\ 1 \\ 0 \\ 1 \\ 2 \\ 1 \\ 2 \end{pmatrix} \Phi_0}$$

Exact solution

$$\begin{pmatrix} a_{11}\Phi_1 & . & . & a_{1N}\Phi_N \\ . & . & . & . \\ . & . & . & . \\ a_{N1}\Phi_1 & . & . & a_{NN}\Phi_N \end{pmatrix} = \begin{pmatrix} b_1 \\ . \\ . \\ b_N \end{pmatrix}$$

$$\boxed{\overleftrightarrow{A}\cdot\vec{\Phi} = \vec{b}}$$

solution $\boxed{\vec{\Phi}^{*} = \overleftrightarrow{A}^{-1}\vec{b}}$

Gauss elimination procedure ⇒ matrix $\overleftrightarrow{A}$ triangular

$$q_{ik} = -\frac{a_{ik}}{a_{kk}}$$

for $k = 1,\ldots, N$

$$a'_{jl} = a_{jl} + q_{jk}a_{kl}\ , \quad \forall j,\ l > k$$

$$b'_i = b_i + q_{ik}b_k$$

$$\boxed{\Rightarrow O\,(N^3) \sim O\,(L^{3d})}$$

$$\Phi^*_N = \frac{b_N}{a_{NN}}$$

once matrix is triangular

$$\Phi^{*} \quad 1\left(b_i - \sum_{j=i}^{N-1} a_{jj+1}\Phi^*_{j+1}\right)$$

Poisson equation in 2d

Independently of the size of the system each row or column has only maximally five non-zero matrix elements

⇒ sparse matrix

Invert with LU decomposition

Use sparse matrix solvers !

Sparse matrices

Store non-zero elements in a vector and also their coordinates *i* and *j* in vectors.

⇒ Yale Sparse Matrix Format

example:

Hanwell Subroutine Library

Iain Duff

For more details see:

www.cise.ufl.edu/research/sparse/codes

Sparse matrix solvers

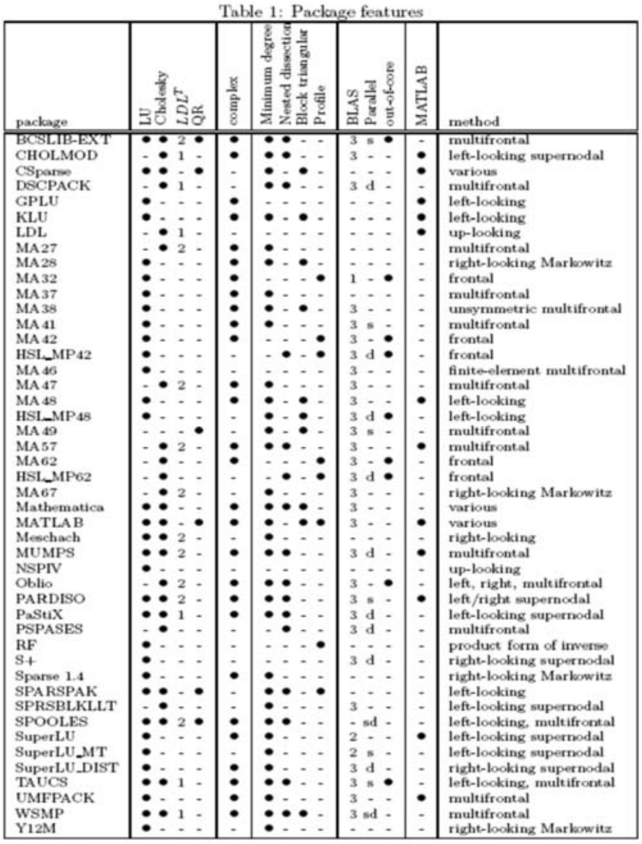


Table 1: Package features

package	LU	Cholesky	LDL^T	QR	complex	Minimum degree	Nested dissection	Block triangular	Profile	BLAS	Parallel	out-of-core	MATLAB	method
BCSLIB-EXT	●	●	2	●	●	●	●	-	-	3	s	●	-	multifrontal
CHOLMOD	-	●	1	-	●	●	●	-	-	3	-	-	●	left-looking supernodal
CSparse	●	●	-	●	-	●	-	●	-	-	-	-	●	various
DSCPACK	-	●	1	-	-	●	●	-	-	3	d	-	-	multifrontal
GPLU	●	-	-	-	●	-	-	-	-	-	-	-	●	left-looking
KLU	●	-	-	-	●	●	-	●	-	-	-	-	●	left-looking
LDL	-	●	1	-	-	-	-	-	-	-	-	-	●	up-looking
MA27	-	●	2	-	●	●	-	-	-	-	-	-	-	multifrontal
MA28	●	-	-	-	●	●	-	●	-	-	-	-	-	right-looking Markowitz
MA32	●	-	-	-	●	-	-	-	●	1	-	●	-	frontal
MA37	●	-	-	-	●	●	-	-	-	-	-	-	-	multifrontal
MA38	●	-	-	-	●	●	-	●	-	3	-	-	-	unsymmetric multifrontal
MA41	●	-	-	-	●	●	-	-	-	3	s	-	-	multifrontal
MA42	●	-	-	-	●	-	-	-	●	3	-	●	-	frontal
HSL_MP42	●	-	-	-	-	-	●	-	●	3	d	●	-	frontal
MA46	●	-	-	-	-	-	-	-	-	3	-	-	-	finite-element multifrontal
MA47	-	●	2	-	●	●	-	-	-	3	-	-	-	multifrontal
MA48	●	-	-	-	●	●	-	●	-	3	-	-	●	left-looking
HSL_MP48	●	-	-	-	-	●	-	●	-	3	d	●	-	left-looking
MA49	-	-	-	●	-	●	-	●	-	3	s	-	-	multifrontal
MA57	-	●	2	-	●	●	●	-	-	3	-	-	●	multifrontal
MA62	-	●	-	-	●	-	-	-	●	3	-	●	-	frontal
HSL_MP62	-	●	-	-	-	-	●	-	●	3	d	●	-	frontal
MA67	-	●	2	-	-	●	-	-	-	3	-	-	-	right-looking Markowitz
Mathematica	●	●	-	-	●	●	●	●	-	3	-	-	-	various
MATLAB	●	●	-	●	●	●	-	●	●	3	-	-	●	various
Meschach	●	●	2	-	-	●	-	-	-	-	-	-	-	right-looking
MUMPS	●	●	2	-	●	●	●	-	-	3	d	-	●	multifrontal
NSPIV	●	-	-	-	-	-	-	-	-	-	-	-	-	up-looking
Oblio	-	●	2	-	●	●	●	-	-	3	-	●	-	left, right, multifrontal
PARDISO	●	●	2	-	●	●	●	-	-	3	s	-	●	left/right supernodal
PaStiX	●	●	1	-	●	●	●	-	-	3	d	-	-	left-looking supernodal
PSPASES	-	●	-	-	-	-	●	-	-	3	d	-	-	multifrontal
RF	●	-	-	-	-	-	-	-	●	-	-	-	-	product form of inverse
S+	●	-	-	-	-	-	-	-	-	3	d	-	-	right-looking supernodal
Sparse 1.4	●	-	-	-	●	●	-	-	-	-	-	-	-	right-looking Markowitz
SPARSPAK	●	●	-	●	-	●	●	-	●	-	-	-	-	left-looking
SPRSBLKLLT	-	●	-	-	-	●	-	-	-	3	-	-	-	left-looking supernodal
SPOOLES	●	●	2	●	●	●	●	-	-	-	sd	-	-	left-looking, multifrontal
SuperLU	●	-	-	-	●	●	-	-	-	2	-	-	●	left-looking supernodal
SuperLU_MT	●	-	-	-	-	●	-	-	-	2	s	-	-	left-looking supernodal
SuperLU_DIST	●	-	-	-	●	●	-	-	-	3	d	-	-	right-looking supernodal
TAUCS	●	●	1	-	●	●	●	-	-	3	s	●	-	left-looking, multifrontal
UMFPACK	●	-	-	-	●	●	-	-	-	3	-	-	●	multifrontal
WSMP	●	●	1	-	●	●	●	●	-	3	sd	-	-	multifrontal
Y12M	●	-	-	-	-	●	-	-	-	-	-	-	-	right-looking Markowitz

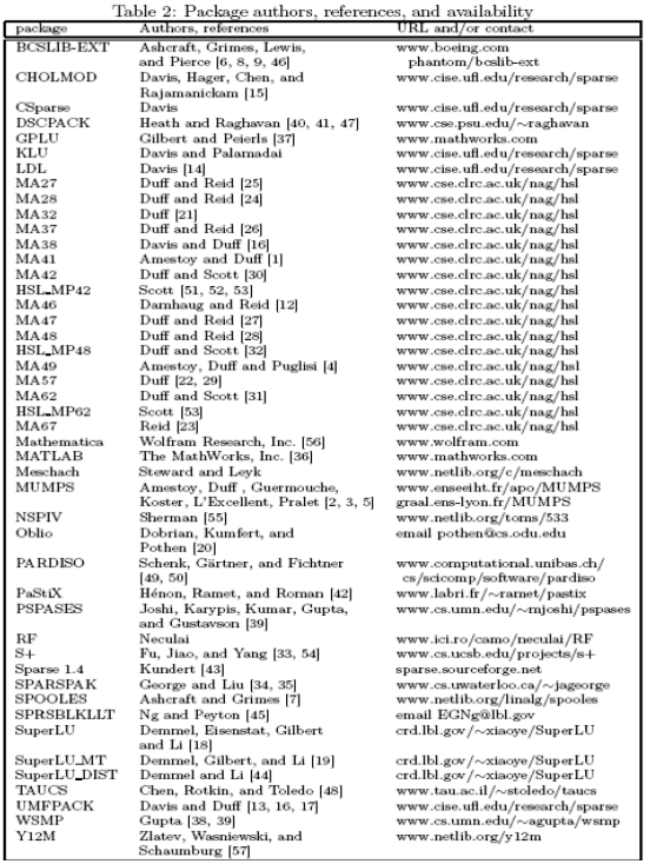
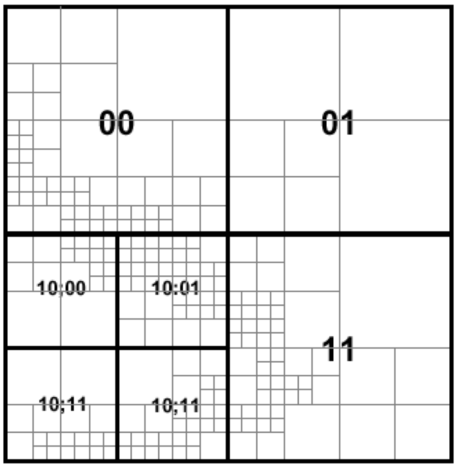


Table 2: Package authors, references, and availability

package	Authors, references	URL and/or contact
BCSLIB-EXT	Ashcraft, Grimes, Lewis, and Pierce [6, 8, 9, 46]	www.boeing.com phantom/bcslib-ext
CHOLMOD	Davis, Hager, Chen, and Rajamanickam [15]	www.cise.ufl.edu/research/sparse
CSparse	Davis	www.cise.ufl.edu/research/sparse
DSCPACK	Heath and Raghavan [40, 41, 47]	www.cse.psu.edu/~raghavan
GPLU	Gilbert and Peierls [37]	www.mathworks.com
KLU	Davis and Palamadai	www.cise.ufl.edu/research/sparse
LDL	Davis [14]	www.cise.ufl.edu/research/sparse
MA27	Duff and Reid [25]	www.cse.clrc.ac.uk/nag/hsl
MA28	Duff and Reid [24]	www.cse.clrc.ac.uk/nag/hsl
MA32	Duff [21]	www.cse.clrc.ac.uk/nag/hsl
MA37	Duff and Reid [26]	www.cse.clrc.ac.uk/nag/hsl
MA38	Davis and Duff [16]	www.cse.clrc.ac.uk/nag/hsl
MA41	Amestoy and Duff [1]	www.cse.clrc.ac.uk/nag/hsl
MA42	Duff and Scott [30]	www.cse.clrc.ac.uk/nag/hsl
HSL_MP42	Scott [51, 52, 53]	www.cse.clrc.ac.uk/nag/hsl
MA46	Damhaug and Reid [12]	www.cse.clrc.ac.uk/nag/hsl
MA47	Duff and Reid [27]	www.cse.clrc.ac.uk/nag/hsl
MA48	Duff and Reid [28]	www.cse.clrc.ac.uk/nag/hsl
HSL_MP48	Duff and Scott [32]	www.cse.clrc.ac.uk/nag/hsl
MA49	Amestoy, Duff and Puglisi [4]	www.cse.clrc.ac.uk/nag/hsl
MA57	Duff [22, 29]	www.cse.clrc.ac.uk/nag/hsl
MA62	Duff and Scott [31]	www.cse.clrc.ac.uk/nag/hsl
HSL_MP62	Scott [53]	www.cse.clrc.ac.uk/nag/hsl
MA67	Reid [23]	www.cse.clrc.ac.uk/nag/hsl
Mathematica	Wolfram Research, Inc. [56]	www.wolfram.com
MATLAB	The MathWorks, Inc. [36]	www.mathworks.com
Meschach	Steward and Leyk	www.netlib.org/c/meschach
MUMPS	Amestoy, Duff , Guermouche, Koster, L'Excellent, Pralet [2, 3, 5]	www.enseeiht.fr/apo/MUMPS graal.ens-lyon.fr/MUMPS
NSPIV	Sherman [55]	www.netlib.org/toms/533
Oblio	Dobrian, Kumfert, and Pothen [20]	email pothen@cs.odu.edu
PARDISO	Schenk, Gärtner, and Fichtner [49, 50]	www.computational.unibas.ch/ cs/scicomp/software/pardiso
PaStiX	Hénon, Ramet, and Roman [42]	www.labri.fr/~ramet/pastix
PSPASES	Joshi, Karypis, Kumar, Gupta, and Gustavson [39]	www.cs.umn.edu/~mjoshi/pspases
RF	Neculai	www.ici.ro/camo/neculai/RF
S+	Fu, Jiao, and Yang [33, 54]	www.cs.ucsb.edu/projects/s+
Sparse 1.4	Kundert [43]	sparse.sourceforge.net
SPARSPAK	George and Liu [34, 35]	www.cs.uwaterloo.ca/~jageorge
SPOOLES	Ashcraft and Grimes [7]	www.netlib.org/linalg/spooles
SPRSBLKLLT	Ng and Peyton [45]	email EGNg@lbl.gov
SuperLU	Demmel, Eisenstat, Gilbert and Li [18]	crd.lbl.gov/~xiaoye/SuperLU
SuperLU_MT	Demmel, Gilbert, and Li [19]	crd.lbl.gov/~xiaoye/SuperLU
SuperLU_DIST	Demmel and Li [44]	crd.lbl.gov/~xiaoye/SuperLU
TAUCS	Chen, Rotkin, and Toledo [48]	www.tau.ac.il/~stoledo/taucs
UMFPACK	Davis and Duff [13, 16, 17]	www.cise.ufl.edu/research/sparse
WSMP	Gupta [38, 39]	www.cs.umn.edu/~agupta/wsmp
Y12M	Zlatev, Wasniewski, and Schaumburg [57]	www.netlib.org/y12m

Quadtrees



Tree data structure where each node has up to four children corresponding to the four quadrants. That means that each node can contain several pointers indexed by two binary variables representing coordinates *i* and *j*.

Computational considerations

Computational effort for Gauss elimination ~ N^3.

For a lattice 100 × 100 = 10^4 one needs 2 days.

⇒ Abandon exact solution and use approximation.

But for that $\overset{\leftrightarrow}{A}$ must be well-conditioned:

example for ill-conditioned situation:

$$\begin{pmatrix} 2.0 & 6.0 \\ 2.0 & 6.00001 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 8.0 \\ 8.00001 \end{pmatrix} \Rightarrow \begin{cases} x = 1.0 \\ y = 1.0 \end{cases}$$

$$\begin{pmatrix} 2.0 & 6.0 \\ 2.0 & 5.99999 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 8.0 \\ 8.00002 \end{pmatrix} \Rightarrow \begin{cases} x = 10.0 \\ y = -2.0 \end{cases}$$

Jacobi relaxation method

example 2d Poisson equation: start with any $\Phi_{ij}(0)$

$$\Phi_{ij}(t+1) = \frac{1}{4}\left(\Phi_{i+1j}(t) + \Phi_{i-1j}(t) + \Phi_{ij+1}(t) + \Phi_{ij-1}(t) - b_{ij}\right)$$

fixed point is the exact solution: $\boxed{\Delta\Phi(x,y) = b(x,y)}$

$$\Phi^*_{ij} = \frac{1}{4}\left(\Phi^*_{i+1j} + \Phi^*_{i-1j} + \Phi^*_{ij+1} + \Phi^*_{ij-1} - b_{ij}\right)$$

general:

$\overleftrightarrow{A} \cdot \vec{\Phi} = \vec{b}$ decompose: $\overleftrightarrow{A} = \overleftrightarrow{D} + \overleftrightarrow{O} + \overleftrightarrow{U}$

$$\Rightarrow \qquad \overleftrightarrow{D}\,\vec{\Phi} = \vec{b} - (\overleftrightarrow{O} + \overleftrightarrow{U})\,\vec{\Phi}$$

$$\boxed{\vec{\Phi}(t+1) = \overleftrightarrow{D}^{-1}\left(\vec{b} - (\overleftrightarrow{O} + \overleftrightarrow{U})\,\vec{\Phi}(t)\right)}$$

Error of Jacobi relaxation

Exact solution is only reached for *t*→∞.

Define required precision *ε*

and stop when :

$$\delta'(t+1) \equiv \frac{\left\|\vec{\Phi}(t+1) - \vec{\Phi}(t)\right\|}{\left\|\vec{\Phi}(t)\right\|} \le \varepsilon$$

real error:

$$\vec{\delta}(t+1) \equiv \underbrace{\overleftrightarrow{A}^{-1}\vec{b}}_{exact\ solution} - \underbrace{\vec{\Phi}(t+1)}_{approximate\ solution} = \overleftrightarrow{A}^{-1}\vec{b} - \overleftrightarrow{D}^{-1}\left(\underset{\overset{\uparrow}{AA^{-1}}}{\vec{b}} - (\overleftrightarrow{O} + \overleftrightarrow{U})\vec{\Phi}(t)\right)$$

$$= -\overleftrightarrow{D}^{-1}(\overleftrightarrow{O} + \overleftrightarrow{U})\underbrace{\left(\overleftrightarrow{A}^{-1}\vec{b} - \vec{\Phi}(t)\right)}_{\vec{\delta}(t)} = -\overleftrightarrow{D}^{-1}(\overleftrightarrow{O} + \overleftrightarrow{U})\vec{\delta}(t)$$

Error of Jacobi relaxation

$$\boxed{\vec{\delta}(t+1) = -\overleftrightarrow{\Lambda}\cdot\vec{\delta}(t) \quad \textbf{with} \quad \overleftrightarrow{\Lambda} = \overleftrightarrow{D}^{-1}\left(\overleftrightarrow{O}+\overleftrightarrow{U}\right)}$$

be λ the largest eigenvalue of $\overleftrightarrow{\Lambda}$ **$0 < |\lambda| < 1$**

for large t :
$$\boxed{\vec{\Phi}(t) \approx \vec{\Phi}^* + \vec{c}\lambda^t}$$

$$\boxed{\frac{\left\|\vec{\Phi}(t+1)-\vec{\Phi}(t)\right\|}{\left\|\vec{\Phi}(t)-\vec{\Phi}(t-1)\right\|} \approx \frac{\lambda^{t+1}-\lambda^t}{\lambda^t-\lambda^{t-1}} = \lambda}$$

Error of Jacobi relaxation

real error:
$$\boxed{\delta(t) \equiv \frac{\left\|\vec{\Phi}^*-\vec{\Phi}(t)\right\|}{\left\|\vec{\Phi}(t)\right\|} \approx \frac{\left\|\vec{c}\right\|}{\left\|\vec{\Phi}(t)\right\|}\lambda^t}$$

$$\boxed{\delta'(t+1) = \frac{\left\|\vec{\Phi}(t+1)-\vec{\Phi}(t)\right\|}{\left\|\vec{\Phi}(t)\right\|} \approx \frac{\left\|\vec{c}(\lambda^{t+1}-\lambda^t)\right\|}{\left\|\vec{\Phi}(t)\right\|} = \frac{\left\|\vec{c}\right\|}{\left\|\vec{\Phi}(t)\right\|}\lambda^t\left|\lambda-1\right|}$$

$$\boxed{\Rightarrow \quad \delta'(t+1) = (1-\lambda)\ \delta(t)}$$

$$\boxed{\delta(t) = \frac{\delta'(t+1)}{1-\lambda} \approx \frac{\left\|\vec{\Phi}(t)-\vec{\Phi}(t-1)\right\|^2}{\left\|\vec{\Phi}(t)\right\|\left(\left\|\vec{\Phi}(t)-\vec{\Phi}(t-1)\right\|-\left\|\vec{\Phi}(t+1)-\vec{\Phi}(t)\right\|\right)}}$$

Gauss-Seidel relaxation

$$\boxed{\Phi_i(t+1) = -\frac{1}{a_{ii}}\left(\sum_{j=i+1}^{N} a_{ij}\Phi_j(t) + \sum_{j=1}^{i-1} a_{ij}\Phi_j(t+1) - b_j\right)}$$

$$\overleftrightarrow{A}\cdot\vec{\Phi} = \vec{b} \qquad \textbf{and} \qquad \overleftrightarrow{A} = \overleftrightarrow{D} + \overleftrightarrow{O} + \overleftrightarrow{U}$$

$$\Rightarrow \qquad (\overleftrightarrow{D} + \overleftrightarrow{O})\ \vec{\Phi} = \vec{b} - \overleftrightarrow{U}\ \vec{\Phi}$$

$$\vec{\Phi}(t+1) = (\overleftrightarrow{D} + \overleftrightarrow{O})^{-1}\left(\ \vec{b} - \overleftrightarrow{U}\ \vec{\Phi}(t)\right)$$

fixed point is the exact solution

Error in Gauss-Seidel

$$\vec{\delta}(t+1) = \underbrace{\overleftrightarrow{A}^{-1}\vec{b}}_{exact\ solution} - \underbrace{(\overleftrightarrow{D} + \overleftrightarrow{O})^{-1}\left(\ \vec{b} - \overleftrightarrow{U}\ \vec{\Phi}(t)\right)}_{approximate\ solution}$$

$$= -(\overleftrightarrow{D} + \overleftrightarrow{O})^{-1}\overleftrightarrow{U}\underbrace{\left(\ \overleftrightarrow{A}^{-1}\vec{b} - \vec{\Phi}(t)\right)}_{\vec{\delta}(t)} = -(\overleftrightarrow{D} + \overleftrightarrow{O})^{-1}\overleftrightarrow{U}\ \vec{\delta}(t)$$

$$\vec{\delta}(t+1) = -\overleftrightarrow{\Lambda}\cdot\vec{\delta}(t) \quad \textbf{with} \qquad \overleftrightarrow{\Lambda} = (\overleftrightarrow{D} + \overleftrightarrow{O})^{-1}\overleftrightarrow{U}$$

$$\delta(t) = \frac{\left\|\vec{\Phi}(t+1) - \vec{\Phi}(t)\right\|}{(1-\lambda)\ \left\|\vec{\Phi}(t)\right\|} \le \varepsilon$$

λ largest EV of $\overleftrightarrow{\Lambda}$

Partial differential equations (PDF)

$$\boxed{\overleftrightarrow{A}\cdot\vec{\Phi}=\vec{b}}$$

Jacobi relaxation

Gauss-Seidel relaxation

Gauss-Seidel relaxation

$$\Phi_i(t+1) = -\frac{1}{a_{ii}}\left(\sum_{j=i+1}^{N} a_{ij}\Phi_j(t) + \sum_{j=1}^{i-1} a_{ij}\Phi_j(t+1) - b_j\right)$$

$$\overleftrightarrow{A}\cdot\vec{\Phi}=\vec{b} \qquad \text{and} \qquad \overleftrightarrow{A}=\overleftrightarrow{D}+\overleftrightarrow{O}+\overleftrightarrow{U}$$

$$\Rightarrow \qquad (\overleftrightarrow{D}+\overleftrightarrow{O})\,\vec{\Phi}=\vec{b}-\overleftrightarrow{U}\,\vec{\Phi}$$

$$\vec{\Phi}(t+1)=(\overleftrightarrow{D}+\overleftrightarrow{O})^{-1}\left(\,\vec{b}-\overleftrightarrow{U}\,\vec{\Phi}(t)\right)$$

fixed point is the exact solution

Overrelaxation

Successive overrelaxation = SOR

$$\boxed{\vec{\Phi}(t+1) = (\boldsymbol{D}+\omega\,\boldsymbol{O})^{-1}\left(\omega\,\vec{\boldsymbol{b}} + \left[(1-\omega)\boldsymbol{D} - \omega\,\boldsymbol{U}\right]\,\vec{\Phi}(t)\right)}$$

Fixed point is the exact solution.

ω is the overrelaxation parameter.

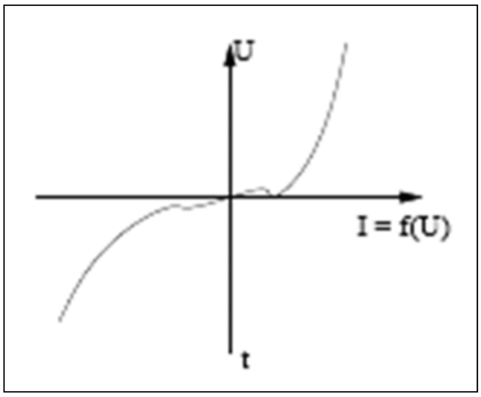
$$1 \leq \omega < 2$$

$\omega = 1$ Gauss-Seidel relaxation

Applet

Non-linear problem

Consider a network of resistors with a non-linear I-*U* relation *f*. Then Kirchhoff's law takes the form:



$$\boxed{f(U_{i+1j} - U_{ij}) + f(U_{ij} - U_{i-1j}) + f(U_{ij+1} - U_{ij}) + f(U_{ij} - U_{ij-1}) = 0}$$

Solve with relaxation:

$$\boxed{\begin{aligned} &f(U_{i+1j}(t) - U_{ij}(t+1)) + f(U_{ij}(t+1) - U_{i-1j}(t)) \\ &+ f(U_{ij+1}(t) - U_{ij}(t+1)) + f(U_{ij}(t+1) - U_{ij-1}(t)) = 0 \end{aligned}}$$

Gradient methods

Be matrix $\overleftrightarrow{A}$ positive and symmetric.

The residuum **error**

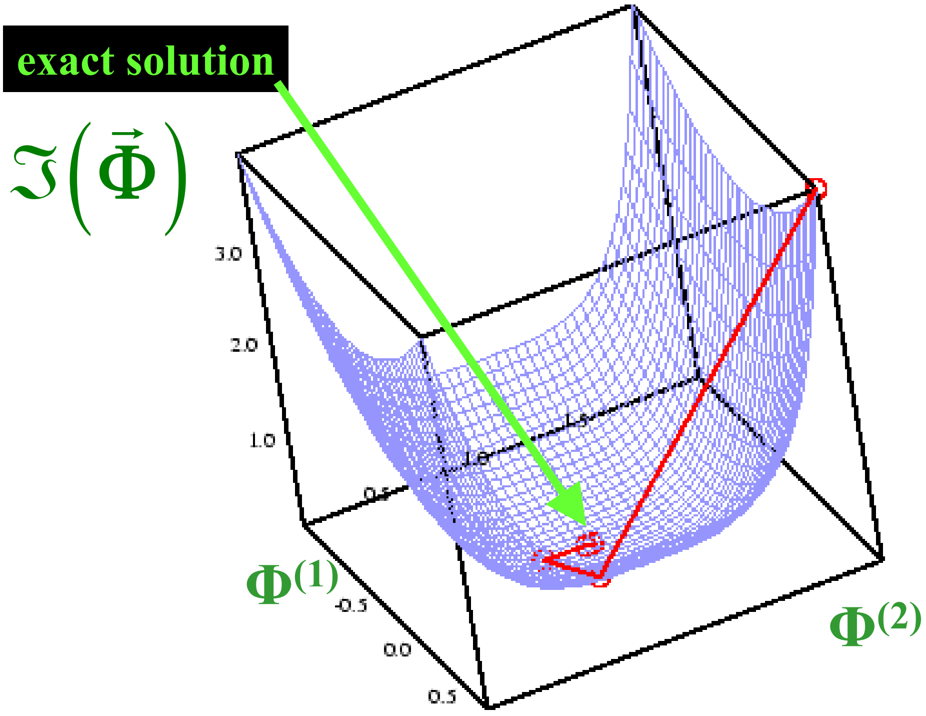
$$\boxed{\vec{r} = \overleftrightarrow{A}\,\vec{\delta} = \overleftrightarrow{A}\left(\overleftrightarrow{A}^{-1}\vec{b} - \vec{\Phi}\right) = \vec{b} - \overleftrightarrow{A}\,\vec{\Phi}}$$

is a measure for the error.

Minimize the functional:

$$\boxed{\Im = \vec{r}^{\,t}\,\overleftrightarrow{A}^{-1}\vec{r} = \begin{cases} 0 \quad \text{if} \quad \vec{\Phi} = \vec{\Phi}^{*} \\ > 0 \quad \text{otherwise} \end{cases}}$$

Gradient methods



Gradient methods

$$\Im = (\vec{b} - \overleftrightarrow{A}\,\vec{\Phi})^t\,\overleftrightarrow{A}^{-1}(\vec{b} - \overleftrightarrow{A}\,\vec{\Phi}) = \vec{b}^t\,\overleftrightarrow{A}^{-1}\vec{b} + \vec{\Phi}^t\,\overleftrightarrow{A}\,\vec{\Phi} - 2\,\vec{b}\,\vec{\Phi}$$

Be $\vec{\Phi}_i$ the *i* th approximation.

Minimize along lines:

$$\boxed{\vec{\Phi} = \vec{\Phi}_i + \alpha_i \vec{d}_i}$$

$$\Im = \vec{b}^t\,\overleftrightarrow{A}^{-1}\vec{b} + \vec{\Phi}_i^t\,\overleftrightarrow{A}\,\vec{\Phi}_i + 2\alpha_i \vec{d}_i^t\,\overleftrightarrow{A}\,\vec{\Phi}_i + \alpha_i^2 \vec{d}_i^t\,\overleftrightarrow{A}\,\vec{d}_i - 2\vec{b}^t\vec{\Phi}_i - 2\alpha_i \vec{b}^t \vec{d}_i$$

minimization condition with respect to α_i:

$$\frac{\partial \Im}{\partial \alpha_i} = 2\vec{d}_i^t\left(\bar{\alpha}_i \overleftrightarrow{A}\,\vec{d}_i - \vec{r}_i\right) = 0 \quad \Rightarrow \quad \bar{\alpha}_i = \frac{\vec{d}_i^t \vec{r}_i}{\vec{d}_i^t\,\overleftrightarrow{A}\,\vec{d}_i}$$

Method of steepest descent

Start with $\vec{\Phi}_1$ and choose $\vec{d}_i = \vec{r}_i$

$$\vec{r}_1 = \vec{b} - \overleftrightarrow{A}\vec{\Phi}_1$$

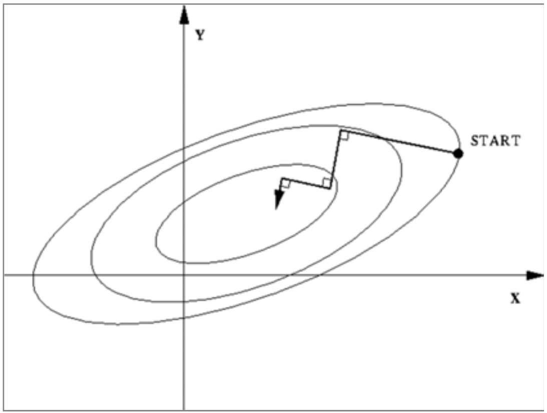
iterate: $\quad \vec{u}_i = \overleftrightarrow{A}\vec{r}_i \quad , \qquad \alpha_i = \dfrac{\vec{r}_i^{\,2}}{\vec{r}_i^{\,t}\vec{u}_i}$

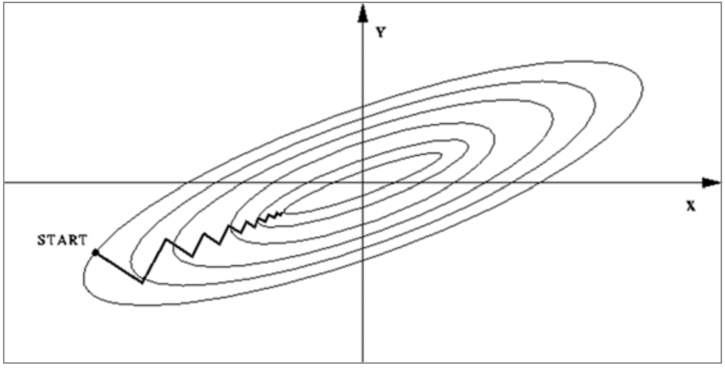
$$\vec{\Phi}_{i+1} = \vec{\Phi}_i + \alpha_i \vec{r}_i$$

$$\vec{r}_{i+1} = \vec{r}_i + \alpha_i \vec{u}_i$$

each step ~ N^2, but when matrix $\overleftrightarrow{A}$ sparse ~ N

Gradient methods





Conjugate gradient

Hestenes and Stiefel (1957)

Choose $\vec{d}_i$ conjugate

to each other:

$$\boxed{\vec{d}_i^{\,t}\overleftrightarrow{A}\vec{d}_j = 0 \quad \textbf{if} \quad i \neq j}$$

as before:

$$\boxed{\vec{r}_i = \vec{b} - \overleftrightarrow{A}\vec{\Phi}_i \;\;,\quad \alpha_i = \frac{\vec{r}_i\vec{d}_i}{\vec{d}_i^{\,t}\overleftrightarrow{A}\vec{d}_i} \;\;,\quad \vec{\Phi}_{i+1} = \vec{\Phi}_i + \alpha_i\vec{d}_i}$$

$$\boxed{\Rightarrow \quad \vec{r}_i = \vec{b} - \overleftrightarrow{A}\left(\vec{\Phi}_1 + \sum_{j=1}^{i-1}\alpha_j\vec{d}_j\right)}$$

Conjugate gradient

Construct conjugate basis using an orthogonalization procedure:

(Gram – Schmidt)

$$\boxed{\vec{d}_1 = \vec{r}_1 \quad , \qquad \vec{d}_i = \vec{r}_i - \sum_{j=1}^{i-1} \frac{\vec{d}_j^{\,t}\, \overleftrightarrow{A}\, \vec{r}_i}{\vec{d}_j^{\,t}\, \overleftrightarrow{A}\, \vec{d}_j}\, \vec{d}_j}$$

one can also show:

$$\boxed{\vec{r}_i^{\,t}\, \overleftrightarrow{A}\, \vec{d}_j = 0 \quad \textbf{if} \quad i \neq j}$$

Conjugate gradient

1. initialize:

$$\boxed{\vec{r}_1 = \vec{b} - \overleftrightarrow{A}\vec{\Phi}_1 \quad , \qquad \vec{d}_1 = \vec{r}_1}$$

2. iterate:

$$\boxed{\begin{array}{l} c = \left(\vec{d}_i^{\,t}\, \overleftrightarrow{A}\, \vec{d}_i\right)^{-1} \quad , \qquad \alpha_i = c\vec{d}_i\vec{r}_i \quad , \qquad \vec{\Phi}_{i+1} = \vec{\Phi}_i + \alpha_i \vec{d}_i \\ \vec{r}_{i+1} = \vec{b} - \overleftrightarrow{A}\vec{\Phi}_{i+1} \quad , \qquad \vec{d}_{i+1} = \vec{r}_{i+1} - \left(c\vec{r}_{i+1}\overleftrightarrow{A}\vec{d}_i\right)\, \vec{d}_i \end{array}}$$

3. stop when:

$$\boxed{\vec{r}_i^{\,t}\vec{r}_i < \varepsilon}$$

Applet

Conjugate gradient

If matrix not symmetric then use biconjugate gradient method.

Consider two residuals:

$$\boxed{\vec{r} = \vec{b} - \overleftrightarrow{A}\vec{\Phi} \qquad \textbf{and} \qquad \tilde{\vec{r}} = \vec{b} - \overleftrightarrow{A}^{t}\vec{\Phi}}$$

This method does not always converge and can be unstable.

Biconjugate gradient

1. initialize:

$$\begin{array}{ll} \vec{r}_1 = \vec{b} - \overleftrightarrow{A}\,\vec{\Phi}_1 \;, & \vec{d}_1 = \vec{r}_1 \\ \tilde{\vec{r}}_1 = \vec{b} - \overleftrightarrow{A}^{t}\vec{\Phi}_1 \;, & \tilde{\vec{d}}_1 = \tilde{\vec{r}}_1 \end{array}$$

2. iterate:

$$\begin{array}{lll} \vec{r}_{i+1} = \vec{r}_i - \alpha_i \overleftrightarrow{A}\,\vec{d}_i \;, & \tilde{\vec{r}}_{i+1} = \tilde{\vec{r}}_i - \alpha_i \overleftrightarrow{A}^{t}\tilde{\vec{d}}_i \;, & \alpha_i = c\;\vec{r}_i{}^{t}\vec{r} \\ \vec{d}_{i+1} = \vec{r}_i + \tilde{\alpha}_i \vec{d}_i \;, & \tilde{\vec{d}}_{i+1} = \tilde{\vec{r}}_i + \tilde{\alpha}_i \tilde{\vec{d}}_i \;, & \tilde{\alpha}_i = \tilde{c}\;\vec{r}_i{}^{t}\vec{r}_i \end{array}$$

$$\textbf{with} \quad c = \left(\vec{d}_i^{\,t}\overleftrightarrow{A}\vec{d}_i\right)^{-1} \quad \textbf{and} \quad \tilde{c} = \left(\tilde{\vec{d}}_i^{\,t}\vec{d}_i\right)^{-1}$$

3. stop when: $\boxed{\vec{r}_i{}^{t}\vec{r}_i < \varepsilon}$ $\Rightarrow$ $\boxed{\vec{\Phi}_n = \vec{\Phi}_1 + \sum_i^n \alpha_i \vec{d}_i}$

Preconditioning

Choose a **preconditioning matrix**

$\overleftrightarrow{P}$ such that $\overleftrightarrow{P}^{-1}\overleftrightarrow{A} \approx \overleftrightarrow{1}$

and solve equation: $\left(\overleftrightarrow{P}^{-1}\overleftrightarrow{A}\right)\vec{\Phi} = \overleftrightarrow{P}^{-1}\vec{b}$

example: Jacobi preconditioner:

$$P_{ij} = A_{ii}\delta_{ij} = \begin{cases} A_{ii} & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases} \Rightarrow P_{ij}^{-1} = \frac{1}{A_{ii}\delta_{ij}}$$

Preconditioning

example: SOR preconditioner:

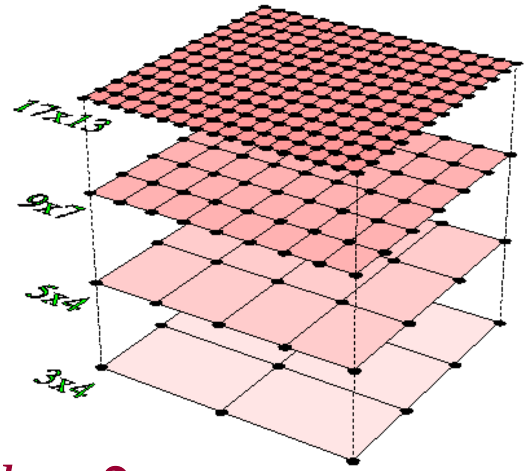
$$\overleftrightarrow{P} = \left(\frac{\overleftrightarrow{D}}{\omega} + \overleftrightarrow{U}\right)^{-1} \frac{\omega}{2-\omega}\overleftrightarrow{D}^{-1}\left(\frac{\overleftrightarrow{D}}{\omega} + \overleftrightarrow{O}\right)$$

$\Rightarrow$ **Preconditioned Conjugate Gradient**

Multigrid procedure

Achi Brandt (1970)

Consider coarser lattices on which the long-wave errors are damped out.



$h = 2$

Multigrid procedure

W.L. Briggs, A Multigrid Tutorial
(Soc. For Ind. & Appl. Math, 1991)

Strategy: solve the equation for the error on the coarser lattice.

Two-level procedure:

1. Determine residuum $\vec{r}$ on the original lattice.

$$\boxed{\vec{r}_n = \overleftrightarrow{b} - \overleftrightarrow{A}\overleftrightarrow{\Phi}_n \quad , \quad \vec{\delta}_n = \overleftrightarrow{A}^{-1}\vec{r}_n}$$

Multigrid procedure

2. Define the residuum on the coarser lattice through a restriction operator $\mathcal{R}$:

$$\boxed{\hat{\vec{r}}_n = \mathcal{R}\ \vec{r}_n}$$

3. Then obtain the error on the coarser lattice solving equation:

$$\boxed{\hat{\overleftrightarrow{A}}\hat{\vec{\delta}}_{n+1} = \hat{\vec{r}}_n}$$

4. Then get the error on the original lattice through an extension operator $\mathcal{P}$:

$$\boxed{\vec{\delta}_{n+1} = \mathcal{P}\ \hat{\vec{\delta}}_{n+1}}$$

Multigrid procedure

5. Get new approximate solution through:

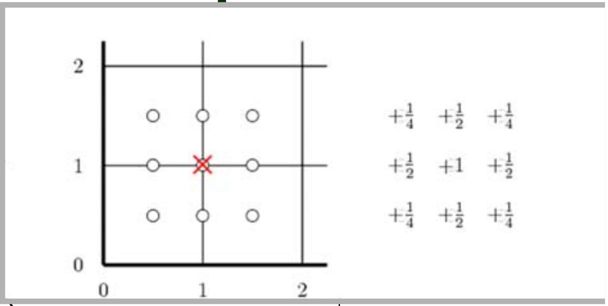
$$\boxed{\vec{\Phi}_{n+1} = \vec{\Phi}_n + \vec{\delta}_{n+1}}$$

In an *m*-level procedure one solves the equation only on the last (coarsest) level.

On each level one can also smoothen the error using several Gauss-Seidel relaxation steps.

Multigrid procedure

Example for extension operator on square lattice:



2
1
0
0
1
2
$+\frac{1}{4}$ $+\frac{1}{2}$ $+\frac{1}{4}$
$+\frac{1}{2}$ $+1$ $+\frac{1}{2}$
$+\frac{1}{4}$ $+\frac{1}{2}$ $+\frac{1}{4}$

bilinear interpolation

$$\mathcal{P}\,\hat{\vec{r}} \mapsto \begin{cases} r_{2i,2j} = \hat{r}_{i,j} \\ r_{2i+1,2j} = \frac{1}{2}\left(\hat{r}_{i,j} + \hat{r}_{i+1,j}\right) \\ r_{2i,2j+1} = \frac{1}{2}\left(\hat{r}_{i,j} + \hat{r}_{i,j+1}\right) \\ r_{2i+1,2j+1} = \frac{1}{4}\left(\hat{r}_{i,j} + \hat{r}_{i+1,j} + \hat{r}_{i,j+1} + \hat{r}_{i+1,j+1}\right) \end{cases}$$

Multigrid procedure

Corresponding restriction operator:

$$\mathcal{R}\,\vec{r} \mapsto \begin{cases} \hat{r}_{i,j} = \frac{1}{4} r_{i,j} + \frac{1}{8}\left(r_{i+1,j} + r_{i-1,j} + r_{i,j+1} + r_{i,j-1}\right) \\ \qquad + \frac{1}{16}\left(r_{i+1,j+1} + r_{i-1,j+1} + r_{i-1,j+1} + r_{i-1,j-1}\right) \end{cases}$$

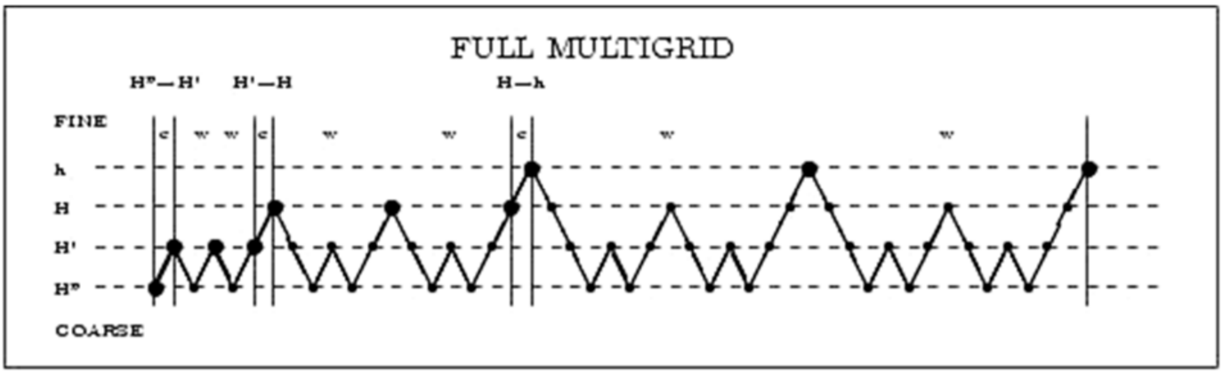
They are adjunct to each other, i.e.

$$\sum_{x,y} \mathcal{P}\,\hat{v}(\hat{x},\hat{y}) \cdot u(x,y) = h^2 \sum_{\hat{x},\hat{y}} \hat{v}(\hat{x},\hat{y}) \cdot \mathcal{R}\,u(x,y)$$

Multigrid procedure

One can also vary the protocol

⇒ V-cycles, W-cycles, …



Solving PDEs

discretize ⇒ system of coupled linear equations

$$\boxed{\overleftrightarrow{A}\cdot\vec{\Phi}=\vec{b}}$$

- **Finite difference methods:**

  Field Φ is discretized on sites: Φ_i.

- **Finite element methods = FEM:**

  Field Φ is patched together from a discrete set of continuous functions.

The Fathers of FEM

J. Argyris R.W. Clough O.C. Zienkiewicz

Finite Elements at ETH

- Gerald Kress: Strukturanalyse mit FEM
- Christoph Schwab: Numerik der Dgln.
- Peter Arbenz: Introduction to FEM
- Pavel Hora: Grundlagen der nichtlinearen FEM
- Andrei Gusev: FEM in Solids and Structures
- Falk Wittel: Eine kurze Einführung in FEM
- Eleni Chatzi: Method of Finite Elements

Literature for FEM

- O.C. Zienkiewicz: „The Finite Element Method“ (3 Volumes), 6th edition (Butterworth-Heinemann, 2005)
- K.J. Bathe: „Finite Element Procedures“ (Prentice Hall, 1996)
- H.R. Schwarz: „Finite Element Methods“ (Academic Press, 1988)

Finite Elements

Strukturmechanik/Anwendung:

[6] J. Altenbach und U. Fischer: Finite-Elemente Praxis, Fachbuchverlag Leipzig (1991)

[7] P. Fröhlich: FEM-Anwendungspraxis. Einstieg in die Finite Elemente Analyse, Vieweg Verlag (2005)

[8] B. Klein: FEM, Vieweg-Verlag 6. Aufl. (2005)

[9] K. Knothe and H. Wells: Finite Elemente, Springer-Verlag (1991)

[10] F.U. Mathiak: Die Methode der finiten Elemente (FEM) – Einführung und Grundlagen (2002).

[11] G. Müller und I. Rehfeld: FEM für Praktiker, Expert-Verlag (1992)

[12] M. Link: Finite Elemente in der Statik und Dynamik, Teubner-Verlag 3. Aufl. (2002)

[13] H. Tottenham und C. Brebbia: Finite Element Techniques in Structural Mechanics, Southhamptom.

[14] R. Steinbuch: Finite Elemente - Ein Einstieg, Springer-Verlag (1998)

Properties of FEM

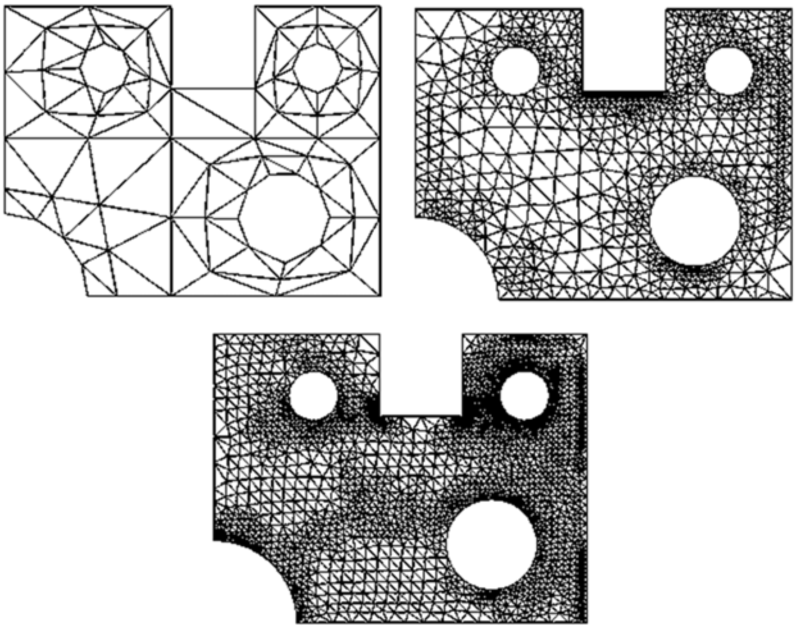
Clough (1960)

Advantage of finite elements over finite differences

- Irregular geometries
- Strongly inhomogeneous fields
- Moving boundaries
- Non-linear equations

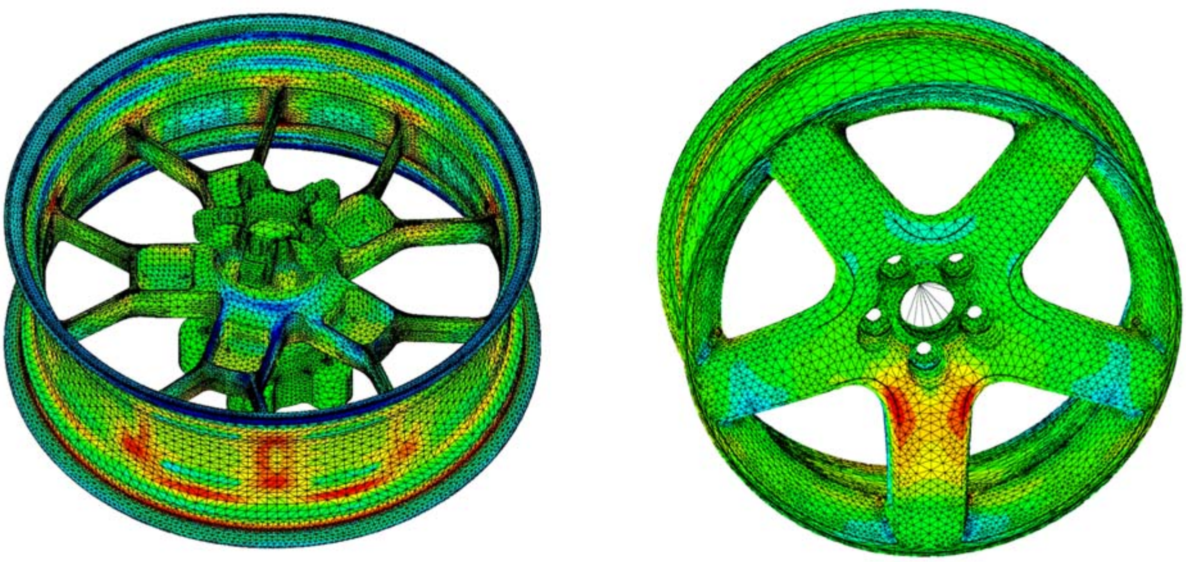
adaptive meshing, e.g. triangulation

Adaptive meshing in 2d



triangulations with different resolution

Adaptive meshing in 3d



triangulation of a wheel-rim

One dimensional example

Poisson equation:

$$\boxed{\frac{d^2\Phi}{dx^2}(x) = -4\pi\rho(x) \quad \text{with} \quad \Phi(0) = \Phi(L) = 0}$$

Expand in terms of localized basis functions u_i:

$$\boxed{\Phi(x) = \sum_{i=1}^{\infty} a_i u_i(x) \approx \Phi_N(x) = \sum_{i=1}^{N} a_i u_i(x)}$$

One dimensional example

Define weight functions $w_j(x)$ and get a_i from:

$$-\sum_{i=1}^{N} a_i \int_0^L \frac{\partial^2 u_i}{\partial x^2}(x)\, w_j(x)\, dx = 4\pi \int_0^L \rho(x)\, w_j(x)\, dx\,, \quad j = 1,\ldots,N$$

$w_j(x) = u_j(x)$ is called the Galerkin method.

⇒ system of linear equations $\boxed{\overset{\leftrightarrow}{A} \cdot \vec{a} = \vec{b}}$

One dimensional example

$\boxed{\overset{\leftrightarrow}{A} \cdot \vec{a} = \vec{b}}$ **with**

$$\boxed{A_{ij} = -\int_0^L u_i''(x)\, w_j(x)\, dx = \int_0^L u_i'(x)\, w_j'(x)\, dx}$$

and $\boxed{b_j = 4\pi \int_0^L \rho(x)\, w_j(x)\, dx}$

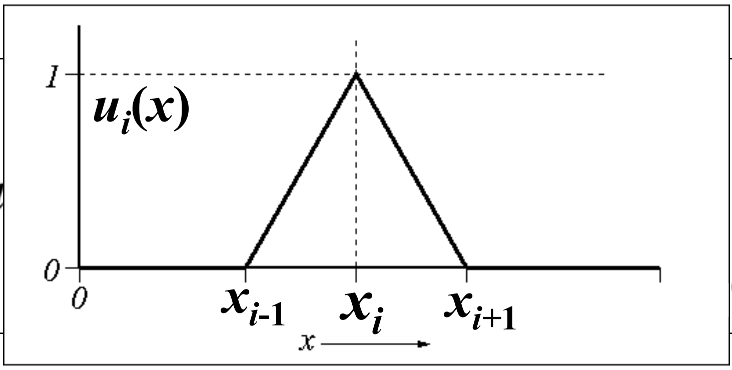
One dimensional example

Example for basis functions $u_i(x)$ are hat functions centered around x_i:

$$\boxed{\begin{aligned}\Delta x &\equiv x_i - x_{i\text{-}1}\\ &= \text{„element“}\end{aligned}}$$

$$u_i(x) = \begin{cases} (x - x_{i-1})/\Delta x & \text{for } x \in [x_{i-1}, x_i] \\ (x_{i+1} - x)/\Delta x & \text{for } x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise} \end{cases}$$

$$\Rightarrow \quad A_{ij} = \int_0^L u$$



$i = j$

$i = j \pm 1$

erwise

One dimensional example

Boundary conditions are automatically fulfilled because basis functions were zero at both ends.

If $\boxed{\Phi(0) = \Phi_0 \quad , \quad \Phi(L) = \Phi_1}$

then use following decomposition:

$$\boxed{\Phi_N(x) = \frac{1}{L}\left(\Phi_0 (L - x) + \Phi_1 x + \sum_{i=1}^{N} a_i u_i(x)\right)}$$

Non-linear PDEs

1d example :

$$\boxed{\Phi\left(x\right)\frac{d^2\Phi}{dx^2}\left(x\right)=-4\pi\rho\left(x\right)}$$

Then solve:

$$\boxed{\int_0^L\left[\Phi\left(x\right)\frac{d^2\Phi}{dx^2}\left(x\right)+4\pi\rho\left(x\right)\right]w_k\left(x\right)\;dx=0}$$

i.e. the coupled non-linear system of equations:

$$\boxed{\sum_{i,j}A_{ijk}a_ia_j=b_k}\quad \text{with}\quad \boxed{A_{ijk}=-\int_0^L u_i\left(x\right)\;u_j''\left(x\right)\;w_k\left(x\right)\;dx}$$

Picard iteration

Start with a guess Φ_0 .

Solve linear equation for Φ_1 :

$$\boxed{\Phi_0(x)\frac{d^2\Phi_1}{dx^2}(x)=-4\pi\rho(x)}$$

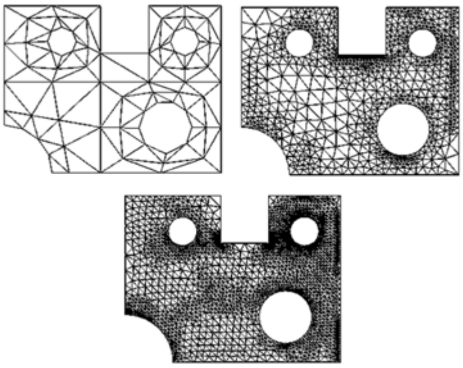
Émile Picard

Then iterate:

$$\boxed{\Phi_n(x)\frac{d^2\Phi_{n+1}}{dx^2}(x)=-4\pi\rho(x)}$$

Finite Elements

$$\boxed{\Delta\Phi(x,y) + a\Phi + b = 0}$$



Decompose in basis functions N_i

$$\boxed{\Phi(x,y) = \sum_{i=1}^{n} \Phi_i N_i(x,y)}$$

Variational Approach

Minimize the functional: **Argyris (1954)**

$$E = \iint_G \left(\frac{1}{2}(\nabla\Phi)^2 + \frac{1}{2} a\,\Phi^2 + \mathrm{b}\,\Phi \right) dxdy + \int_\Gamma \left(\frac{\alpha}{2}\Phi^2 + \beta\,\Phi \right) ds$$

$$\delta E = \iint_G \left(\nabla\Phi\,\delta\nabla\Phi + a\Phi\,\delta\Phi + \mathrm{b}\,\delta\Phi \right) dxdy + \int_\Gamma \left(\alpha\Phi\,\delta\Phi + \beta\delta\Phi \right) ds$$

first Green's theorem:

$$\iint_G \nabla\Phi\nabla\Psi dxdy = -\iint_G \Psi\Delta\Phi\, dxdy + \int_\Gamma \frac{\partial\Phi}{\partial n}\Psi ds \quad \Rightarrow$$

$$\delta E = \iint_G \boxed{\left(-\Delta\Phi + a\Phi + \mathrm{b} \right)} = \mathbf{0} :dy + \int_\Gamma \left(\alpha\Phi + \beta + \frac{\partial\Phi}{\partial n} \right) \delta\Phi\; ds = 0$$

Variational Approach

$$\boxed{\Delta\Phi = \boldsymbol{a}\ \Phi + \mathbf{b}}$$

$a = 0$ Poisson equation

$\mathrm{b} = 0$ Helmholtz equation

First term of total energy

$$E = \sum_{elements\ j} \iint_{G_j} \left(\frac{1}{2}(\nabla\Phi)^2 + \frac{1}{2}\boldsymbol{a}\ \Phi^2 + \mathbf{b}\ \Phi \right) dxdy$$

can be brought into the form:

$$E = \frac{1}{2}\vec{\Phi}\ \overleftrightarrow{A}\ \vec{\Phi} + \vec{b}\ \vec{\Phi}$$

Minimizing then gives:

$$\frac{\partial E}{\partial \Phi} = 0 \quad \Rightarrow \quad \overleftrightarrow{A}\ \vec{\Phi} + \vec{b} = 0$$

Function on Element

Higher dimensions

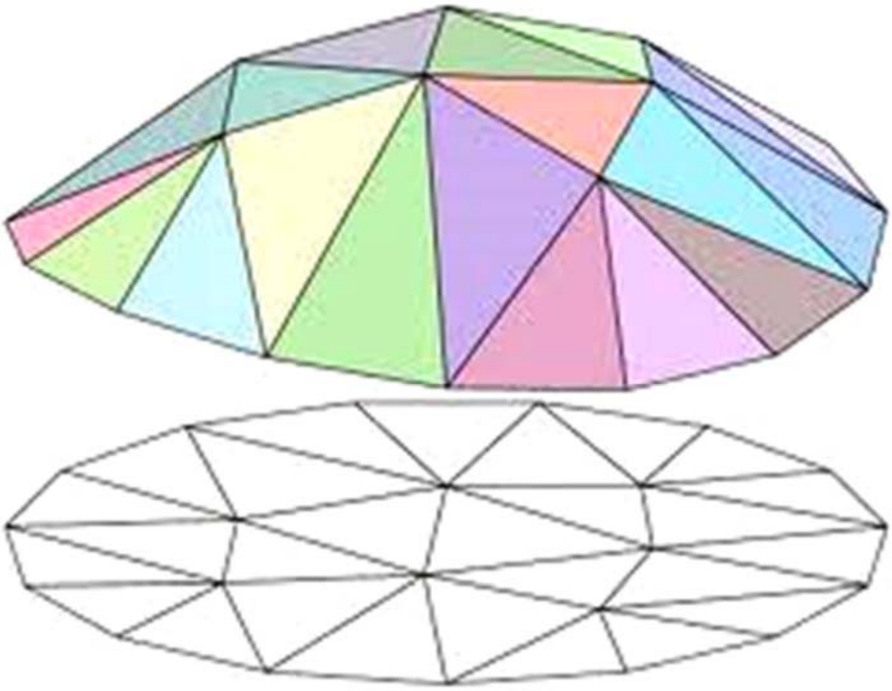
In 2d define function over one element = triangle of the triangulation

e.g. linearly: $\Phi(\vec{r}) \approx a_1 + a_2 x + a_3 y$

or by a paraboloid:

$$\Phi(\vec{r}) \approx a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy + a_6 y^2$$

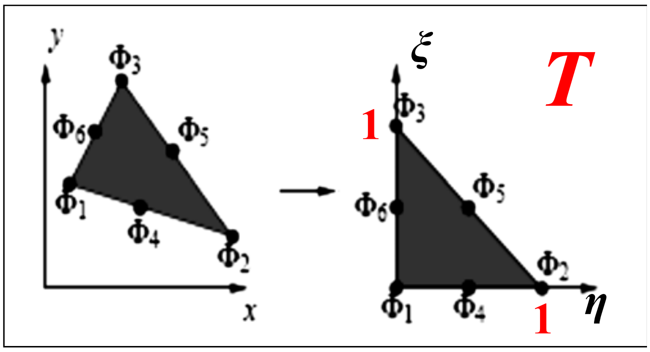
Linear case



Standard Form

Transform any element *j* into the standard form.

$$x = x_1 + (x_2 - x_1)\xi + (x_3 - x_1)\eta$$
$$y = y_1 + (y_2 - y_1)\xi + (y_3 - y_1)\eta$$



$$\Leftrightarrow \quad \begin{aligned} \eta &= \left((y - y_1)(x_2 - x_1) - (x - x_1)(y_2 - y_1)\right)/D \\ \xi &= \left((x - x_1)(y_3 - y_1) - (y - y_1)(x_3 - x_1)\right)/D \\ D &= (y_3 - y_1)(x_2 - x_1) - (x_3 - x_1)(y_2 - y_1) \end{aligned}$$

Coordinate transformation

$$\nabla\Phi=\left(\frac{\partial\Phi}{\partial x},\frac{\partial\Phi}{\partial y}\right)\rightarrow\nabla\Phi=\left(\frac{\partial\Phi}{\partial\xi}\frac{\partial\xi}{\partial x}+\frac{\partial\Phi}{\partial\eta}\frac{\partial\eta}{\partial x},\frac{\partial\Phi}{\partial\xi}\frac{\partial\xi}{\partial y}+\frac{\partial\Phi}{\partial\eta}\frac{\partial\eta}{\partial y}\right)$$

$$\frac{\partial\xi}{\partial x}=\frac{y_3-y_1}{D}\qquad\frac{\partial\xi}{\partial y}=-\frac{x_3-x_1}{D}\qquad\frac{\partial\eta}{\partial x}=-\frac{y_2-y_1}{D}\qquad\frac{\partial\eta}{\partial y}=\frac{x_2-x_1}{D}$$

$$\left(\frac{\partial\Phi}{\partial x}\right)^2=\left(\underset{\equiv\Phi_\xi}{\boxed{\frac{\partial\Phi}{\partial\xi}}}\frac{\partial\xi}{\partial x}+\underset{\equiv\Phi_\eta}{\boxed{\frac{\partial\Phi}{\partial\eta}}}\frac{\partial\eta}{\partial x}\right)^2$$

$$=\frac{(y_3-y_1)^2}{D^2}\Phi_\xi^2-2\frac{(y_3-y_1)(y_2-y_1)}{D^2}\Phi_\xi\Phi_\eta+\frac{(y_2-y_1)^2}{D^2}\Phi_\eta^2$$

$$\left(\frac{\partial\Phi}{\partial y}\right)^2=\frac{(x_3-x_1)^2}{D^2}\Phi_\xi^2-2\frac{(x_3-x_1)(x_2-x_1)}{D^2}\Phi_\xi\Phi_\eta+\frac{(x_2-x_1)^2}{D^2}\Phi_\eta^2$$

Coordinate transformation

$$\iint_{G_j}\ldots dxdy=\iint_{T}\ldots\det\left(\overleftrightarrow{J}\right)\,d\xi d\eta$$

Jacobi matrix

$$\overleftrightarrow{J}=\begin{pmatrix}\dfrac{\partial x}{\partial\xi} & \dfrac{\partial x}{\partial\eta}\\ \dfrac{\partial y}{\partial\xi} & \dfrac{\partial y}{\partial\eta}\end{pmatrix}$$

$$\det\left(\overleftrightarrow{J}\right)=\frac{\partial x}{\partial\xi}\frac{\partial y}{\partial\eta}-\frac{\partial x}{\partial\eta}\frac{\partial y}{\partial\xi}$$

$$=\left(x_2-x_1\right)\left(y_3-y_1\right)-\left(x_3-x_1\right)\left(y_2-y_1\right)=D$$

Coordinate transformation

Inserting gives for each element

$$\iint_{G_j}\left(\Phi_x^2+\Phi_y^2\right)\,dxdy=\iint_T\left(c_1\Phi_\xi^2+2c_2\Phi_\xi\Phi_\eta+c_3\Phi_\eta^2\right)\,d\xi d\eta$$

$$\iint_{G_j}\left(\Phi_x^2+\Phi_y^2\right)\,dxdy=c_1\iint_T\Phi_\xi^2\,d\xi d\eta+2c_2\iint_T\,\Phi_\xi\Phi_\eta d\xi d\eta+c_3\iint_T\,\Phi_\eta^2 d\xi d\eta$$

coefficients are only calculated once for each element.

$$c_1=\frac{\quad}{D}+\frac{\quad}{D}$$

$$c_2=-\frac{\left(y_3-y_1\right)\left(y_2-y_1\right)}{D}-\frac{\left(x_3-x_1\right)\left(x_2-x_1\right)}{D}$$

$$c_3=\frac{\left(y_2-y_1\right)^2}{D}+\frac{\left(x_2-x_1\right)^2}{D}$$

Basis functions

In 2d define function over

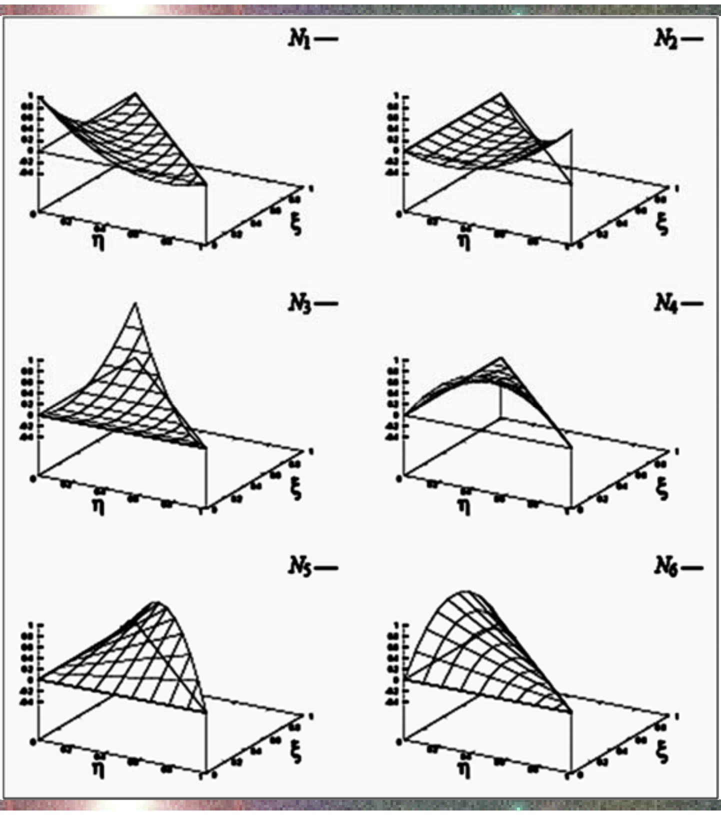
one element = triangle of the triangulation

e.g. linearly: $\Phi(\vec{r})\approx a_1+a_2x+a_3y$

or by a paraboloid:

$$\Phi(\vec{r})\approx a_1+a_2x+a_3y+a_4x^2+a_5xy+a_6y^2$$

Shape of basis functions



Basis functions

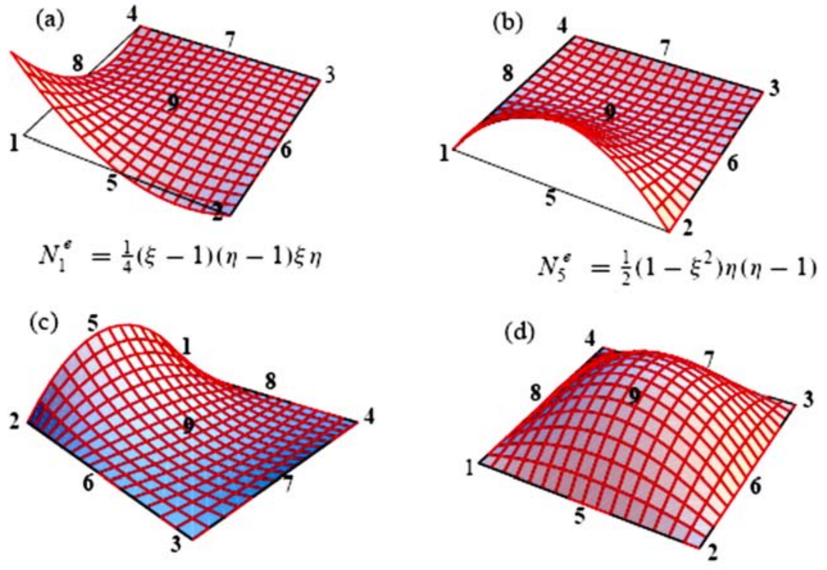
Decompose on standard element in basis functions *N_i*

$$\boxed{\Phi(\xi,\eta)=\sum_{i=1}^{6}\Phi_i N_i(\xi,\eta)=\vec{\varphi}\,\vec{N}(\xi,\eta)}$$

$$\begin{array}{|ll|}\hline N_1=(1-\xi-\eta)(1-2\xi-2\eta)\ , & N_2=\xi(2\xi-1)\\ N_3=\eta(2\eta-1)\ , & N_4=4\xi(1-\xi-\eta)\\ N_5=4\xi\eta\ , & N_6=4\eta(1-\xi-\eta)\\ \hline\end{array}$$

$$\boxed{\vec{\varphi}=(\Phi_1,\dots,\Phi_6)\ ,\quad \vec{N}=(N_1,\dots,N_6)}$$

Shape functions on square lattice



$N_1^e = \frac{1}{4}(\xi-1)(\eta-1)\xi\eta$

$N_5^e = \frac{1}{2}(1-\xi^2)\eta(\eta-1)$

$N_5^e = \frac{1}{2}(1-\xi^2)\eta(\eta-1)$ (back view)

$N_9^e = (1-\xi^2)(1-\eta^2)$

$$\boxed{\Phi(\vec{r}) \approx c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2 + c_7 xy^2 + c_8 x^2 y + c_9 x^2 y^2}$$

Energy Integrals

Calculate the energy integrals on standard element

$$I_1 = \iint_T \Phi_\xi^2 d\xi d\eta = \iint_T \left(\vec{\varphi}\, \vec{N}_\xi(\xi,\eta)\right)^2 d\xi d\eta$$

$$= \iint_T \vec{\varphi}^t \vec{N}_\xi \vec{N}_\xi^t \vec{\varphi}\, d\xi d\eta = \vec{\varphi}^t \underbrace{\iint_T \vec{N}_\xi \vec{N}_\xi^t d\xi d\eta}_{\overset{\leftrightarrow}{S}_1} \vec{\varphi}$$

and analogously

$$I_2 = \iint_T \Phi_\xi \Phi_\eta d\xi d\eta = \vec{\varphi}^t \overset{\leftrightarrow}{S}_2 \vec{\varphi}$$

$$I_3 = \iint_T \Phi_\eta^2 d\xi d\eta = \vec{\varphi}^t \overset{\leftrightarrow}{S}_3 \vec{\varphi}$$

6 x 6

defining matrices $\overset{\leftrightarrow}{S}_1$, $\overset{\leftrightarrow}{S}_2$ and $\overset{\leftrightarrow}{S}_3$ on standard triangle.

Rigidity Matrix

$$\boxed{\begin{gathered}\iint_{G_j} (\nabla\Phi)^2 \, dxdy = \\ \iint_T \left(c_1\Phi_\xi^2 + 2c_2\Phi_\xi\Phi_\eta + c_3\Phi_\eta^2\right) d\xi d\eta = \vec{\varphi}^t \overleftrightarrow{S}\vec{\varphi}\end{gathered}}$$

defines the rigidity matrix $\overleftrightarrow{S}$ for any element:

$$\boxed{\overleftrightarrow{S} = c_1\overleftrightarrow{S}_1 + 2c_2\overleftrightarrow{S}_2 + c_3\overleftrightarrow{S}_3}$$

Mass Matrix

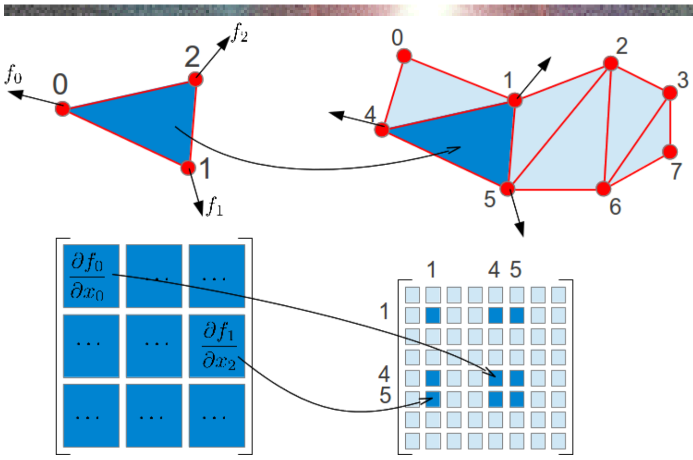
Analogously one defines the mass matrix $\overleftrightarrow{M}$:

$$\boxed{\begin{gathered}\iint_{G_j} a\;\Phi^2 dxdy = \iint_T a\;\left(\vec{\varphi}_j\,\vec{N}(\xi,\eta)\right)^2 D_j\;d\xi d\eta \\ = \vec{\varphi}_j^t\;\underbrace{a\iint_T \vec{N}\;\vec{N}^t D_j\;d\xi d\eta}_{\overleftrightarrow{M}_j}\;\vec{\varphi}_j \equiv \vec{\varphi}_j^t\overleftrightarrow{M}_j\vec{\varphi}_j\end{gathered}}$$

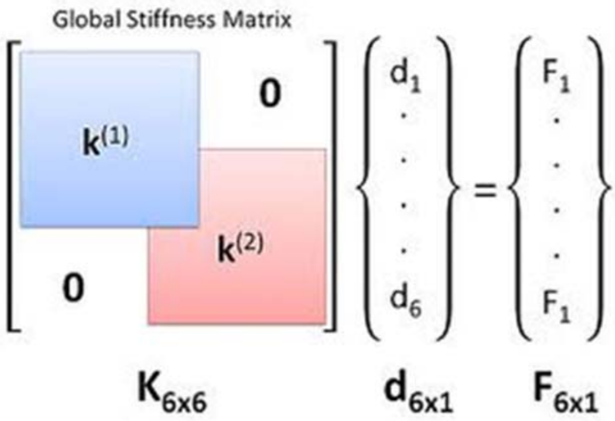
$$\boxed{E = \sum_{elements\ j}\iint_{G_j}\left((\nabla\Phi)^2 + a\;\Phi^2\right) dxdy = \sum_{elements\ \ j}\vec{\varphi}_j^t\left(\overleftrightarrow{S}_j + \overleftrightarrow{M}_j\right)\vec{\varphi}_j}$$

$$\boxed{\Rightarrow\; E = \vec{\Phi}^t\;\overleftrightarrow{A}\;\vec{\Phi} \quad \textbf{with}\;\; \vec{\Phi} = \left(\vec{\varphi}_j\right) \quad \textbf{and}\;\; \overleftrightarrow{A} = \therefore\overset{\text{assembly}}{\underset{j}{\otimes}}\therefore\left(\overleftrightarrow{S}_j + \overleftrightarrow{M}_j\right)}$$

Assembly of the Matrix


f_0
0
2
f_2
1
f_1
0
1
2
3
4
5
6
7
$\frac{\partial f_0}{\partial x_0}$
…
…
…
…
$\frac{\partial f_1}{\partial x_2}$
…
…
…
1
4 5
1
4
5

The elements must be joined such that the field is continuous.



This is done by identifying the values of the coefficients at each vertex for all elements that share this vertex.

Field term

$$\boxed{\begin{aligned}&\iint_{G_j} \mathbf{b}\Phi\, dxdy = \iint_T \mathbf{b}\ \vec{\varphi}_j\ \vec{N}(\xi,\eta)\ D_j d\xi d\eta \\ &= \vec{\varphi}_j\ \underbrace{\mathbf{b}\iint_T \vec{N}(\xi,\eta)\ D_j d\xi d\eta}_{\vec{b}_j} = \vec{b}_j\vec{\varphi}_j\end{aligned}}$$

$$\Rightarrow$$

$$\boxed{E = \vec{\Phi}\ \overleftrightarrow{A}\ \vec{\Phi} + \vec{b}\ \vec{\Phi} \quad \textbf{with}\ \ \vec{b} = \left(\vec{b}_j\right)}$$

Numerical task of FEM

⇒ Solve system $\boxed{\overset{\leftrightarrow}{A}\,\vec{\Phi}+\vec{b}=0}$

of N linear equations where N is the number of vertices.

Matrix $\overset{\leftrightarrow}{A}$ and vector $\vec{b}$ only depend on the triangulation and on the basis functions and the unknowns are the coefficients $\vec{\Phi}=(\vec{\varphi}_i)$.

FEM

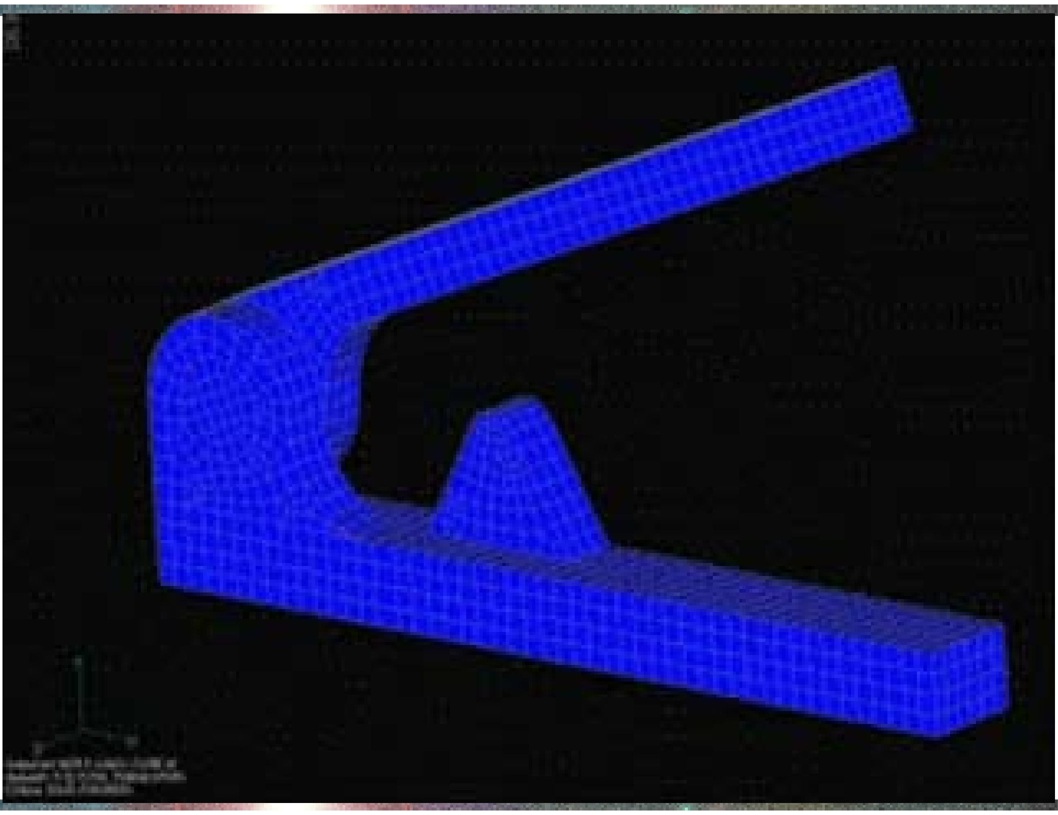
The connection between the elements gives off-diagonal terms in the matrix $\overset{\leftrightarrow}{A}$.

Finally one must also include the boundary terms, which appear as before on the right side of the equation.

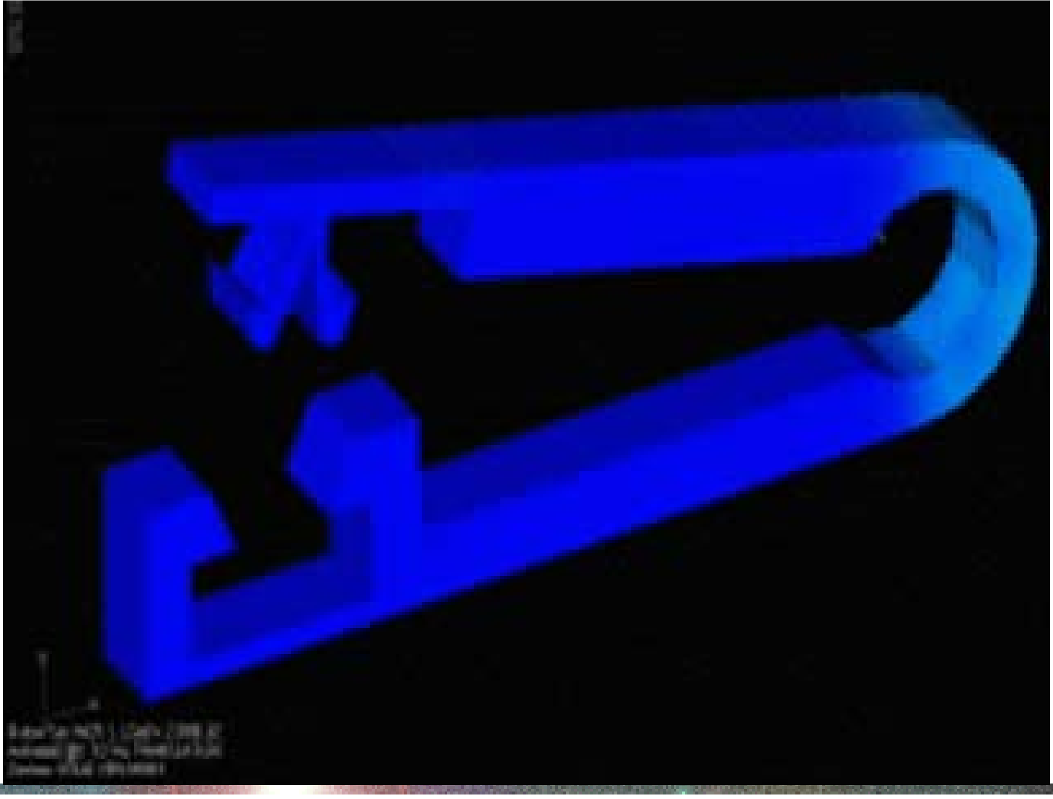
Applet

http://www.lnm.mw.tum.de/teaching/tmapplets/

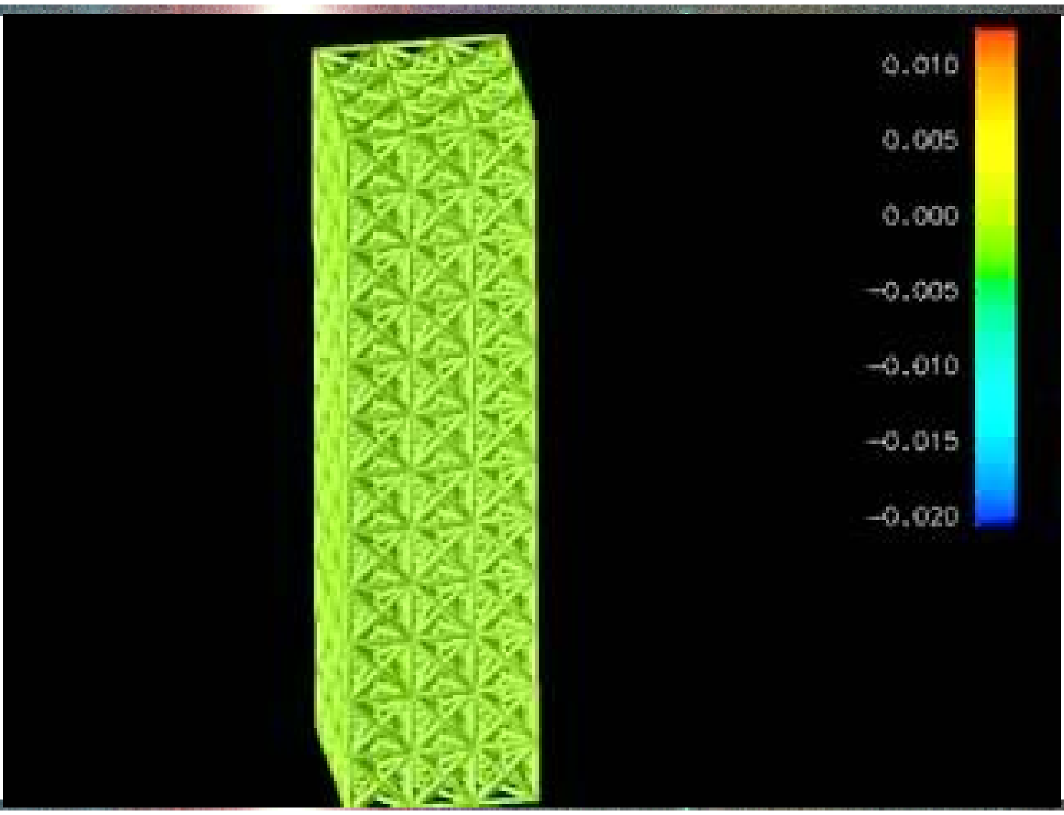
Stresses in a hinge



Stresses in a clip



Network of trusses



Time dependent PDE‘s

Simple example is heat equation:

$$\boxed{\frac{\partial T}{\partial t}(\vec{x},t)=\frac{\kappa}{C\rho}\nabla^2 T(\vec{x},t)+\frac{1}{C\rho}W(\vec{x},t)}$$

T **is temperature,** ***C*** **is specific heat**
ρ **is density,** ***κ*** **is thermal conductivity**
and ***W*** **are external sources or sinks.**

Time dependent PDE‘s

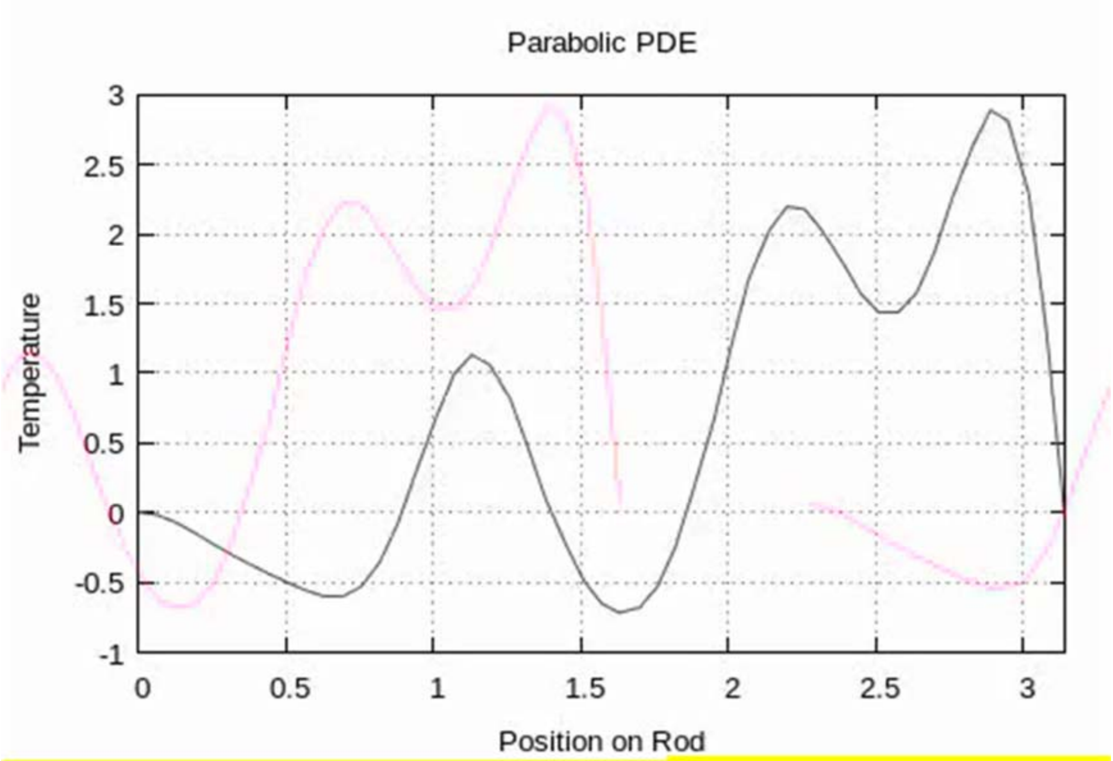
„line method“ in two dimensions:

$$T(x_{ij},t+\Delta t)=T(x_{ij},t)+\frac{\kappa\ \Delta t}{C\rho\ \Delta x^2}\Big(T(x_{i+1j},t)+T(x_{i-1j},t)$$
$$+T(x_{ij+1},t)+T(x_{ij-1},t)-4T(x_{ij},t)\Big)+\frac{\Delta t}{C\rho}W(x_{ij},t)$$

clearly unstable if $\boxed{\frac{\kappa\ \Delta t}{C\rho\ \Delta x^2}\geq\frac{1}{4}}$

Courant-Friedrichs-Lewy (CFL) condition (1928)

Unstable 1d parabolic PDE



Crank - Nicolson method

(1947)

implicit algorithm

John Crank

Phyllis Nicolson

$$T(\vec{x},t+\Delta t) = T(\vec{x},t) + \frac{\kappa\,\Delta t}{2C\rho}\left(\nabla^2 T(\vec{x},t) + \nabla^2 T(\vec{x},t+\Delta t)\right) + \frac{\Delta t}{2C\rho}\left(W(\vec{x},t) + W(\vec{x},t+\Delta t)\right)$$

define

$$\vec{T}(t) = \left(T(x_n,t)\right)\quad,\quad \vec{W}(t) = \left(W(x_n,t)\right)\quad,\quad n = 1,\ldots,L^2$$

Crank - Nicolson method

Define operator O

$$\mathbf{O}T(x_n,t) = \frac{\kappa\,\Delta t}{C\rho\,\Delta x^2}\left(T(x_{n+1},t) + T(x_{n-1},t) + T(x_{n+L},t) + T(x_{n-L},t) - 4T(x_n,t)\right)$$

Then Crank – Nicolson becomes:

$$T(\vec{x},t+\Delta t) = T(\vec{x},t) + \frac{1}{2}\left(\mathbf{O}T(\vec{x},t) + \mathbf{O}T(\vec{x},t+\Delta t)\right) + \frac{\Delta t}{2C\rho}\left(W(\vec{x},t) + W(\vec{x},t+\Delta t)\right)$$

Crank - Nicolson method

$$2\;T(\vec{x},t+\Delta t) = 2\;T(\vec{x},t) + \left(\mathbf{O}T(\vec{x},t) + \mathbf{O}T(\vec{x},t+\Delta t)\right) + \frac{\Delta t}{C\rho}\left(W(\vec{x},t) + W(\vec{x},t+\Delta t)\right)$$

Then Crank – Nicolson becomes:

$$\left(2\cdot\mathbf{1}-\mathbf{O}\right)\;\vec{T}(t+\Delta t) = \left(2\cdot\mathbf{1}+\mathbf{O}\right)\;\vec{T}(t) + \frac{\Delta t}{C\rho}\left(\;\vec{W}(t) + \vec{W}(t+\Delta t)\right)$$

where $\mathbf{1}$ is the unity operator.

Crank - Nicolson method

Calculate the inverted operator $\mathbf{B}$ before:

$$\mathbf{B} = \left(2\cdot\mathbf{1}-\mathbf{O}\right)^{-1}$$

$$\vec{T}(t+\Delta t) = \mathbf{B}\left[\left(2\cdot\mathbf{1}+\mathbf{O}\right)\;\vec{T}(t) + \frac{\Delta t}{C\rho}\left(\;\vec{W}(t) + \vec{W}(t+\Delta t)\right)\right]$$

Crank - Nicolson method

Example: 1d diffusion equation:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$$

Courant-Friedrichs-Lewy (CFL) number

$$\frac{u_i(t+\Delta t)-u_i(t)}{\Delta t} = \frac{D}{2(\Delta x)^2}\Big[\big(u_{i+1}(t+\Delta t)-2u_i(t+\Delta t)+u_{i-1}(t+\Delta t)\big) + \big(u_{i+1}(t)-2u_i(t)+u_{i-1}(t)\big)\Big]$$

$$\mu = \frac{D\Delta t}{2(\Delta x)^2}$$

tridiagonal problem

$$-\mu u_{i+1}(t+\Delta t)+(1+2\mu)u_i(t+\Delta t)-\mu u_{i-1}(t+\Delta t) = \mu u_{i+1}(t)+(1-2\mu)u_i(t)+\mu u_{i-1}(t)$$

Tridiagonal matrix problem

equation:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$$

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}$$

modify coefficients:

$$c'_i = \begin{cases} \dfrac{c_i}{b_i} & ; \quad i = 1 \\ \dfrac{c_i}{b_i - c'_{i-1}a_i} & ; \quad i = 2,3,\dots,n-1 \end{cases}$$

$$d'_i = \begin{cases} \dfrac{d_i}{b_i} & ; \quad i = 1 \\ \dfrac{d_i - d'_{i-1}a_i}{b_i - c'_{i-1}a_i} & ; \quad i = 2,3,\dots,n. \end{cases}$$

solution:

$$x_n = d'_n$$

$$x_i = d'_i - c'_i x_{i+1} \qquad ; \; i = n-1, n-2, \dots, 1.$$

Algorithm goes like O(N) (instead of O(N^3) in Gauss elimination).

Wave equation

$$\frac{\partial^2 y}{\partial t^2} = c^2 \; \nabla^2 y \quad \text{with} \quad c = \sqrt{k/\rho}$$

$$\Rightarrow \quad \frac{y(x_n, t_{k+1}) + y(x_n, t_{k-1}) - 2y(x_n, t_k)}{\Delta t^2} \approx c^2 \nabla^2 y(x_n, t_k)$$

$$\Rightarrow \quad y(x_n, t_{k+1}) = 2(1-2\lambda^2)\, y(x_n, t_k) - y(x_n, t_{k-1})$$
$$+\lambda^2 \left(y(x_{n+1}, t_k) + y(x_{n-1}, t_k) + y(x_{n+L}, t_k) + y(x_{n-L}, t_k) \right)$$

with $\lambda = c\Delta t / \Delta x \; < 1/\sqrt{2}$

which corresponds to cut off modes for wave lengths smaller than λ .

Initialization

To start the iterations one needs to know the field at two times *t* and *t*-Δ*t*.

That means, one needs to know $y(x_n,0)$ and $\frac{\partial y}{\partial t}(x_n, 0)$

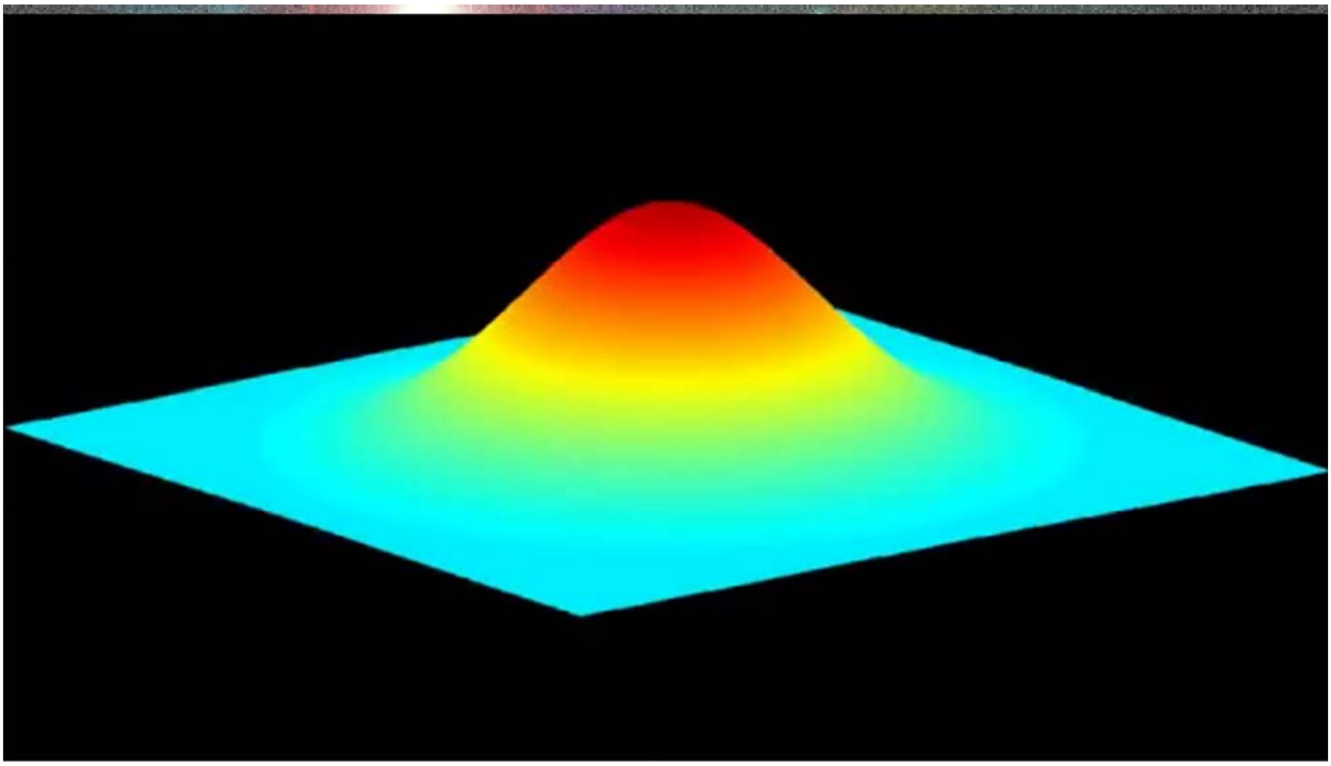
Set $$y(x_n, \Delta t) = y(x_n, 0) + \Delta t \frac{\partial y}{\partial t}(x_n, 0)$$ error O(Δ*t*)

better

$$y(x_n, \Delta t) \quad = \quad (1-\lambda^2)\, y(x_n, 0) + \Delta t \frac{\partial y}{\partial t}(x_n, 0)$$
$$+\frac{\lambda^2}{4}\left(y(x_{n+1}, 0) + y(x_{n-1}, 0) + y(x_{n+L}, 0) + y(x_{n-L}, 0) \right)$$

error O(Δ*t*²)

Solution of the wave equation



Navier – Stokes equation

$\vec{v}(\vec{x},t)$ velocity field, $p(\vec{x},t)$ pressure field

$$\frac{\partial\vec{v}}{\partial t}+(\vec{v}\vec{\nabla})\vec{v}=-\frac{1}{\rho}\vec{\nabla}p+\mu\ \nabla^2\vec{v}\ ,\ \ \vec{\nabla}\vec{v}=0$$

↖ viscosity

Euler: $$\frac{\partial\vec{v}}{\partial t}+(\vec{v}\vec{\nabla})\vec{v}=-\frac{1}{\rho}\vec{\nabla}p$$

Stokes: $$\frac{\partial\vec{v}}{\partial t}=-\frac{1}{\rho}\vec{\nabla}p+\mu\ \nabla^2\vec{v}$$

equation of motion for incompressible fluid

Solvers for NS equation

- Penalty method with MAC
- Finite Volume Method (FLUENT, OpenFOAM)
- Spectral method
- Lattice Boltzmann (Ladd)
- Discrete methods: DPD, SPH, SRD, LGA,…
- k-ε model for turbulence

CFD = Computational Fluid Dynamics

Navier – Stokes equation

$$\boxed{\frac{\vec{v}_{k+1}-\vec{v}_k}{\Delta t} = -\vec{\nabla} p_{k+1} - \mu\,\nabla^2 \vec{v}_k - (\vec{v}_k \cdot \vec{\nabla})\vec{v}_k}$$

Apply on both sides $\vec{\nabla}$:

$$\boxed{\Rightarrow \qquad \cancel{\frac{\vec{\nabla}\vec{v}_{k+1}-\vec{\nabla}\vec{v}_k}{\Delta t}} = -\nabla^2 p_{k+1} - \mu\,\cancel{\nabla^2\left(\vec{\nabla}\vec{v}_k\right)} - \vec{\nabla}\left(\vec{v}_k\vec{\nabla}\right)\vec{v}_k}$$

Insert incompressibility condition:

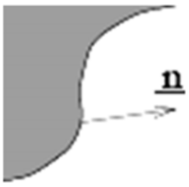
$$\boxed{\vec{\nabla}\vec{v}_{k+1} = \vec{\nabla}\vec{v}_k = 0}$$

Navier – Stokes equation

$$\boxed{\nabla^2 p_{k+1} = -\vec{\nabla}\left((\vec{v}_k \cdot \vec{\nabla})\vec{v}_k\right)}$$

Poisson equation ⇒ determine pressure p_{k+1}

To solve it, one needs boundary conditions for the pressure which one obtains projecting the NS equation on the boundary. This must be done numerically.



Operator splitting

Introduce auxiliary variable field $\vec{v}^*$

$$\boxed{\frac{\vec{v}_{k+1} - \vec{v}^* + \vec{v}^* - \vec{v}_k}{\Delta t} = -\vec{\nabla} p_{k+1} - \mu\ \nabla^2 \vec{v}_k - (\vec{v}_k \cdot \vec{\nabla})\ \vec{v}_k}$$

and split in two equations:

$$\frac{\vec{v}^* - \vec{v}_k}{\Delta t} = -\mu\ \nabla^2 \vec{v}_k - (\vec{v}_k \cdot \vec{\nabla})\ \vec{v}_k$$

$$\Rightarrow \vec{v}^*$$

$$\frac{\vec{v}_{k+1} - \vec{v}^*}{\Delta t} = -\vec{\nabla} p_{k+1}$$

Operator splitting

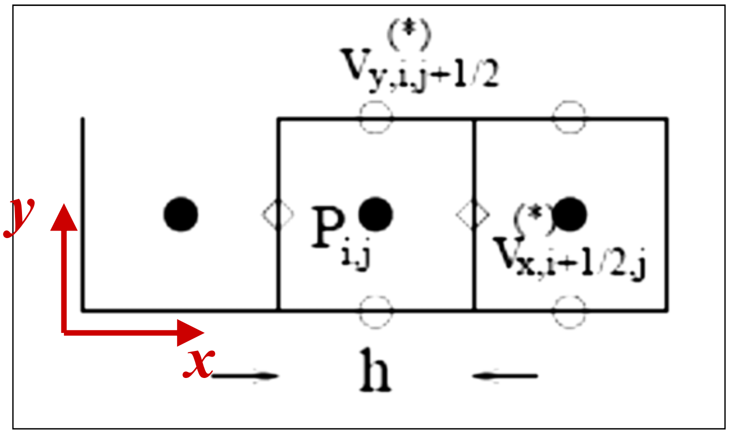
Applying $\vec{\nabla}$ on $\boxed{\dfrac{\vec{v}_{k+1}-\vec{v}^{*}}{\Delta t}=-\vec{\nabla}p_{k+1}}$

one obtains $\boxed{\nabla^{2}p_{k+1}=\dfrac{\vec{\nabla}\vec{v}^{*}}{\Delta t}}$

Projecting on the normal $\vec{n}$ to the boundary one obtains: $\boxed{\dfrac{\partial p_{k+1}}{\partial n}\equiv\left(\vec{n}\cdot\vec{\nabla}\right)p_{k+1}=\dfrac{1}{\Delta t}\vec{n}\left(\vec{v}^{*}-\vec{v}_{k+1}\right)}$

Spatial discretization

MAC = Marker and Cell is a staggered lattice:

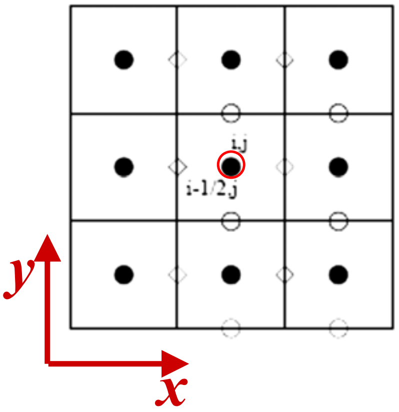


h is the lattice spacing

Place components of velocity on middle of edges and pressures in the centers of the cells.

Spatial discretization

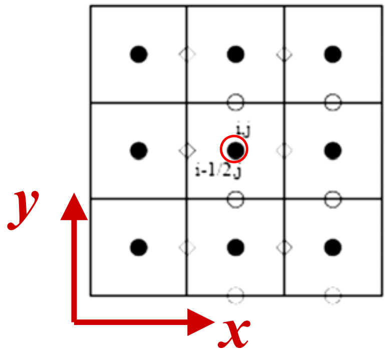
$$\left(\vec{\nabla} p\right)_{x,i+1/2,j} = \frac{1}{\mathbf{h}}\left(p_{i+1,j} - p_{i,j}\right)$$



$$\nabla^2 p_{i,j} = \frac{1}{\mathbf{h}^2}\left(p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - 4p_{i,j}\right)$$

Spatial discretization

$$\vec{\nabla}\vec{v}^{*}_{i,j} = \frac{1}{\mathbf{h}}\left(v^{*}_{x,i+\frac{1}{2},j} - v^{*}_{x,i-\frac{1}{2},j} + v^{*}_{y,i,j+\frac{1}{2}} - v^{*}_{y,i,j-\frac{1}{2}}\right)$$

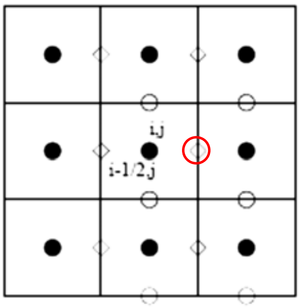


$$\nabla^2 p_{k+1} = \frac{\vec{\nabla}\vec{v}^{*}}{\Delta t}$$

Poisson equation for the pressure p_{k+1} is solved on the centers of the cells (•) .

Spatial discretization

$$\vec{v}_{k+1} = \vec{v}_k + \Delta t\left(-\vec{\nabla} p_{k+1} - \mu\, \nabla^2 \vec{v}_k - (\vec{v}_k \cdot \vec{\nabla})\vec{v}_k\right)$$

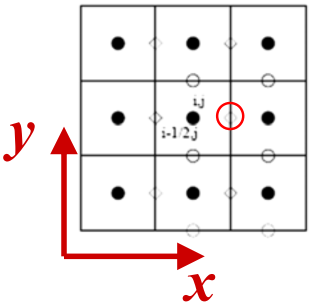


The equations for the velocity components are solved on the edges.

$$\left(\vec{v}\cdot\vec{\nabla}\right)\, v^x = v^x \left(\partial/\partial x\right)\, v^x + v^y \left(\partial/\partial y\right)\, v^x$$

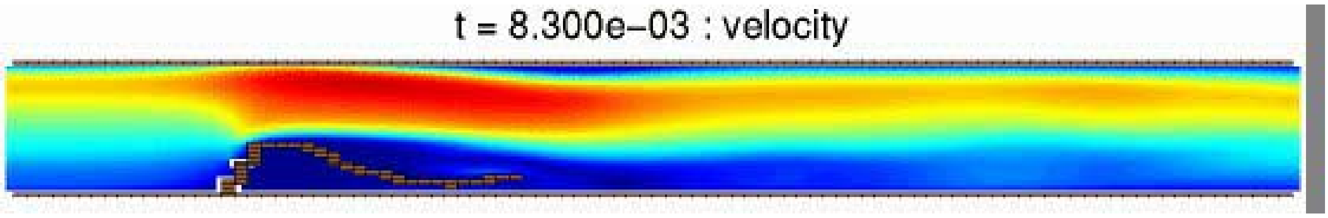
Spatial discretization

$$\left(v^x \frac{\partial v^x}{\partial x}\right)_{i+\frac{1}{2},j} = v^x_{i+\frac{1}{2},j} \cdot \frac{1}{2\mathrm{h}}\left(v^x_{i+\frac{3}{2},j} - v^x_{i-\frac{1}{2},j}\right)$$

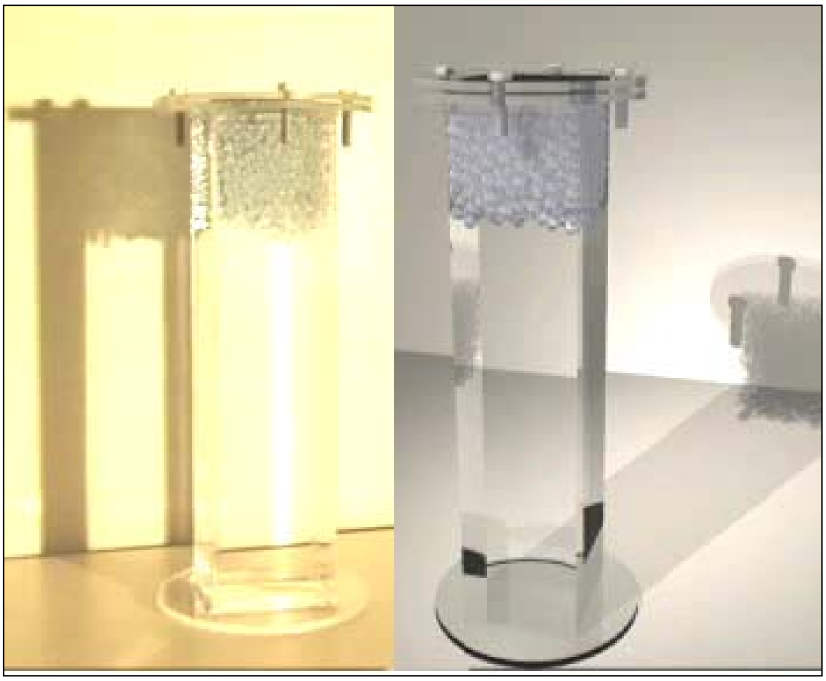


$$\left(v^y \frac{\partial v^x}{\partial y}\right)_{i+\frac{1}{2},j} = \frac{1}{4}\left(v^y_{i,j+\frac{1}{2}} + v^y_{i,j-\frac{1}{2}} + v^y_{i+1,j+\frac{1}{2}} + v^y_{i+1,j-\frac{1}{2}}\right) \times \frac{1}{2\mathrm{h}}\left(v^x_{i+\frac{1}{2},j+1} - v^x_{i+\frac{1}{2},j-1}\right)$$

Flow around a vocal chord



Sedimentation



Glass beads descending in silicon oil

comparing experiment and simulation

Finite Volume Method

R.J. LeVeque, «Finite Volume Methods for Hyperbolic Problems» (Cambridge Univ. Press, 2002)

Solve conservation law $\frac{\partial}{\partial t} v(x,t) + \nabla f(v(x,t)) = g(v(x,t))$

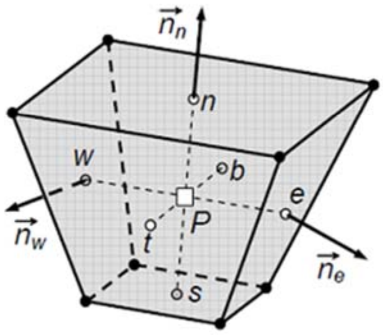
in integral form $\int_{G_i} \left(\frac{\partial v}{\partial t} + \nabla f(v) \right) dV = \int_{G_i} g(v)\, dV$

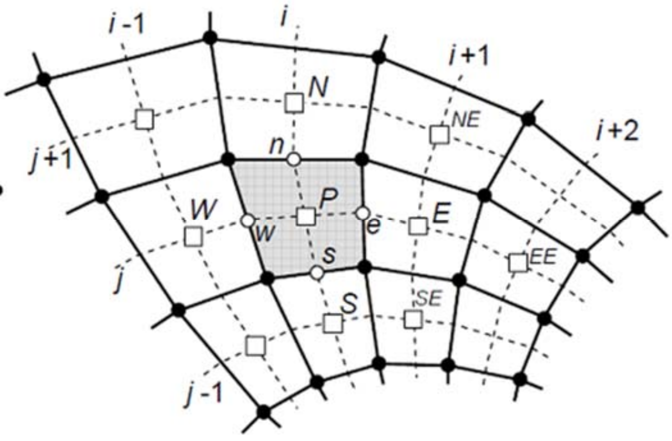
using Green's theorem:

$$\int_{G_i} \frac{\partial v}{\partial t} dV + \int_{\partial G_i} f(v)\; n\; dS = \int_{G_i} g(v)\, dV$$

Finite Volume Method

$$\int_{G_i} \frac{\partial v}{\partial t} dV + \int_{\partial G_i} f(v)\; n\; dS = 0$$





change of value in volume *i*

$$\frac{\partial v_i}{\partial t} = -\frac{1}{|G_i|} \int_{\partial G_i} f(v)\; n\; dS$$

Forward-Time Central-Space (FTCS)

$$\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{t}} = -\nabla \boldsymbol{f}\left(\boldsymbol{v}, \boldsymbol{x}, \boldsymbol{t}, \frac{\partial^2 \boldsymbol{v}}{\partial \boldsymbol{x}^2}\right)$$

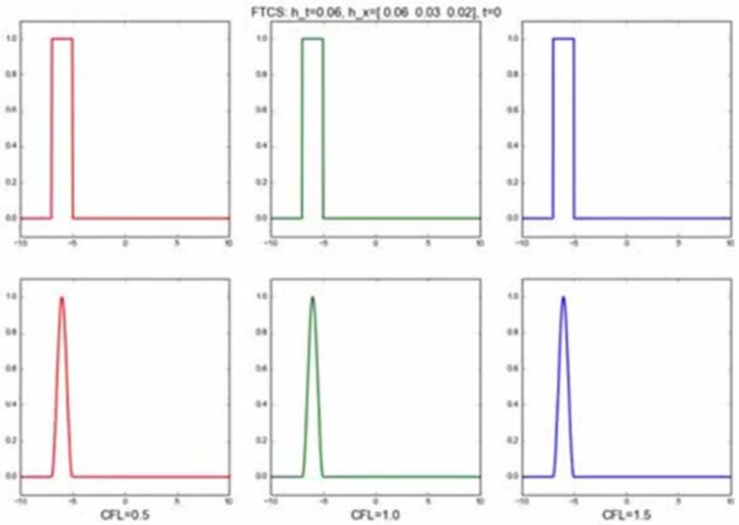
$$\frac{v_i(t+\Delta t) - v_i(t)}{\Delta t} = -\nabla f\left(v, i, t, \frac{\partial^2 v}{\partial x^2}\right)$$

f **is spatially discretized in a central difference scheme**

$$v_i(t+\Delta t) = v_i(t) - \frac{\Delta t}{2\Delta x}\left(f(v_{i+1}(t)) - f(v_{i-1}(t))\right)$$

FTCS

Time evolution of the inviscid Euler equation using a forward time central space scheme



Lax-Friedrichs Scheme

Peter Lax

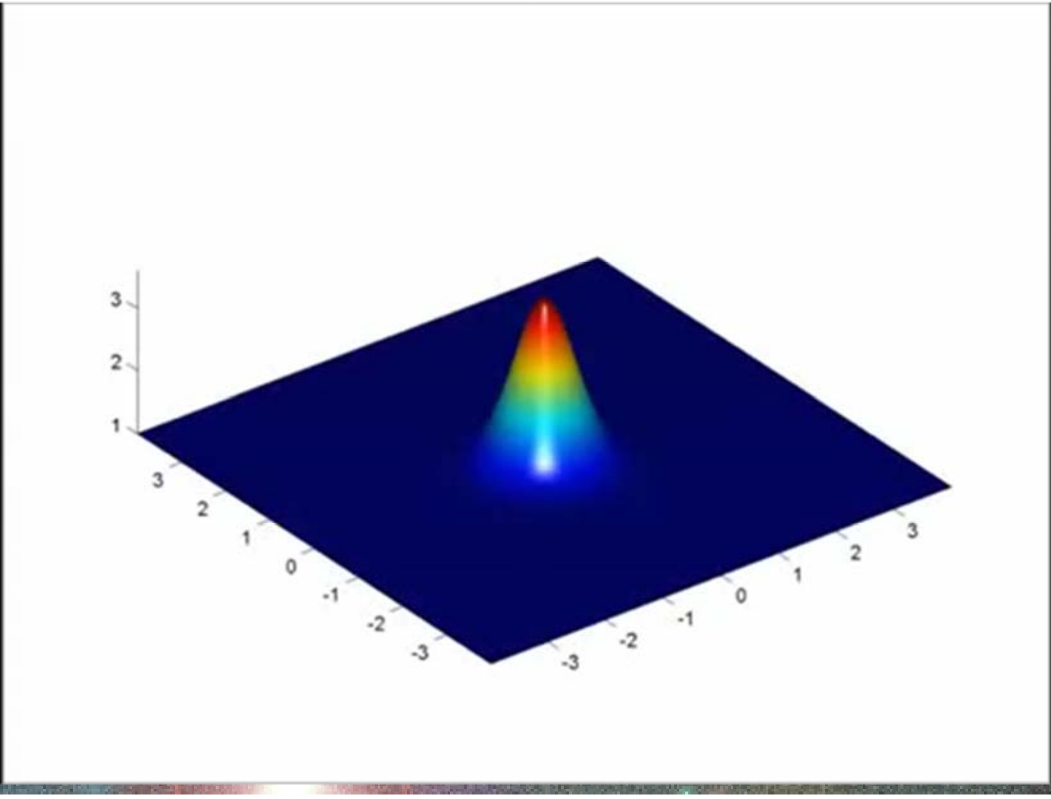
Kurt Friedrichs

$$v_i(t+\Delta t)$$

$$= \frac{1}{2}\left(v_{i+1}(t)+v_{i-1}(t)\right) - \frac{\Delta t}{2\Delta x}\left(f(v_{i+1}(t)) - f(v_{i-1}(t))\right)$$

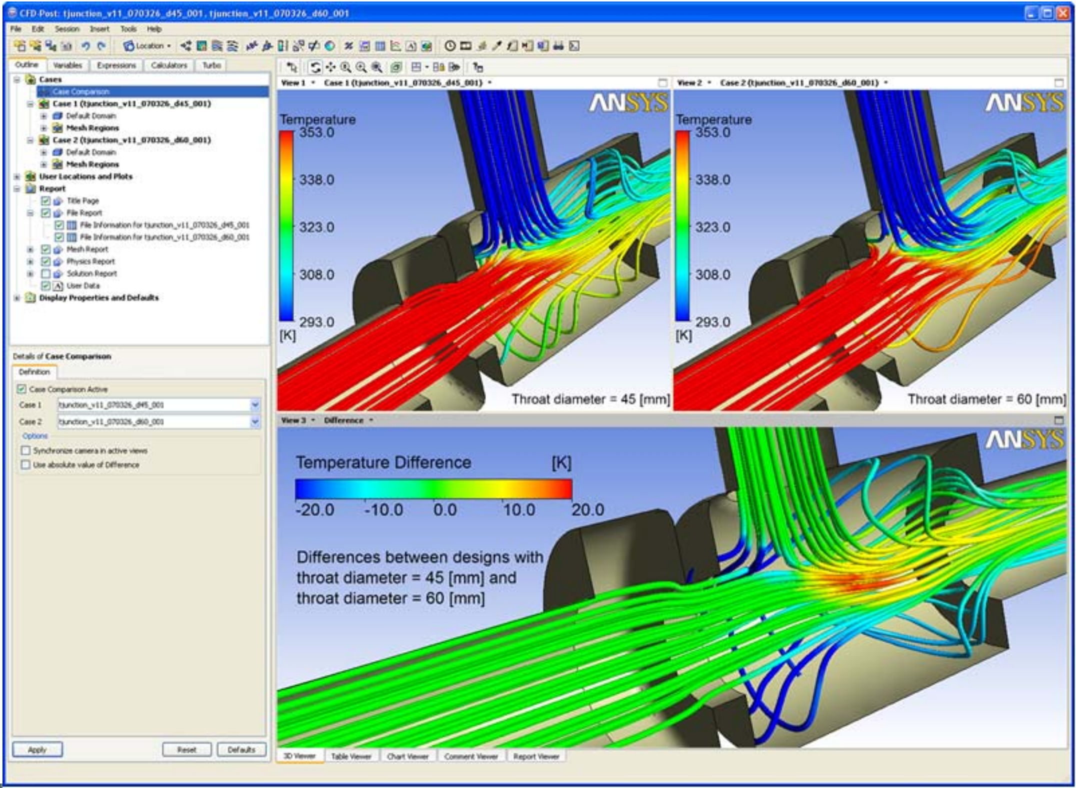
Lax-Friedrichs Scheme

2d Euler equation with reflecting boundaries



FLUENT

ETH



CFD-Post: tjunction_v11_070326_d45_001, tjunction_v11_070326_d60_001
File Edit Session Insert Tools Help
Location
Outline Variables Expressions Calculators Turbo
Cases
Case Comparison
Case 1 (tjunction_v11_070326_d45_001)
Default Domain
Mesh Regions
Case 2 (tjunction_v11_070326_d60_001)
Default Domain
Mesh Regions
User Locations and Plots
Report
Title Page
File Report
File Information for tjunction_v11_070326_d45_001
File Information for tjunction_v11_070326_d60_001
Mesh Report
Physics Report
Solution Report
User Data
Display Properties and Defaults
View 1 Case 1 (tjunction_v11_070326_d45_001)
View 2 Case 2 (tjunction_v11_070326_d60_001)
View 3 Difference
ANSYS
Temperature
353.0
338.0
323.0
308.0
293.0
[K]
Throat diameter = 45 [mm]
Throat diameter = 60 [mm]
Temperature Difference [K]
-20.0 -10.0 0.0 10.0 20.0
Apply Reset Defaults
3D Viewer Table Viewer Chart Viewer Comment Viewer Report Viewer

Details of Case Comparison

Definition

☑ Case Comparison Active

Case 1 tjunction_v11_070326_d45_001

Case 2 tjunction_v11_070326_d60_001

Options

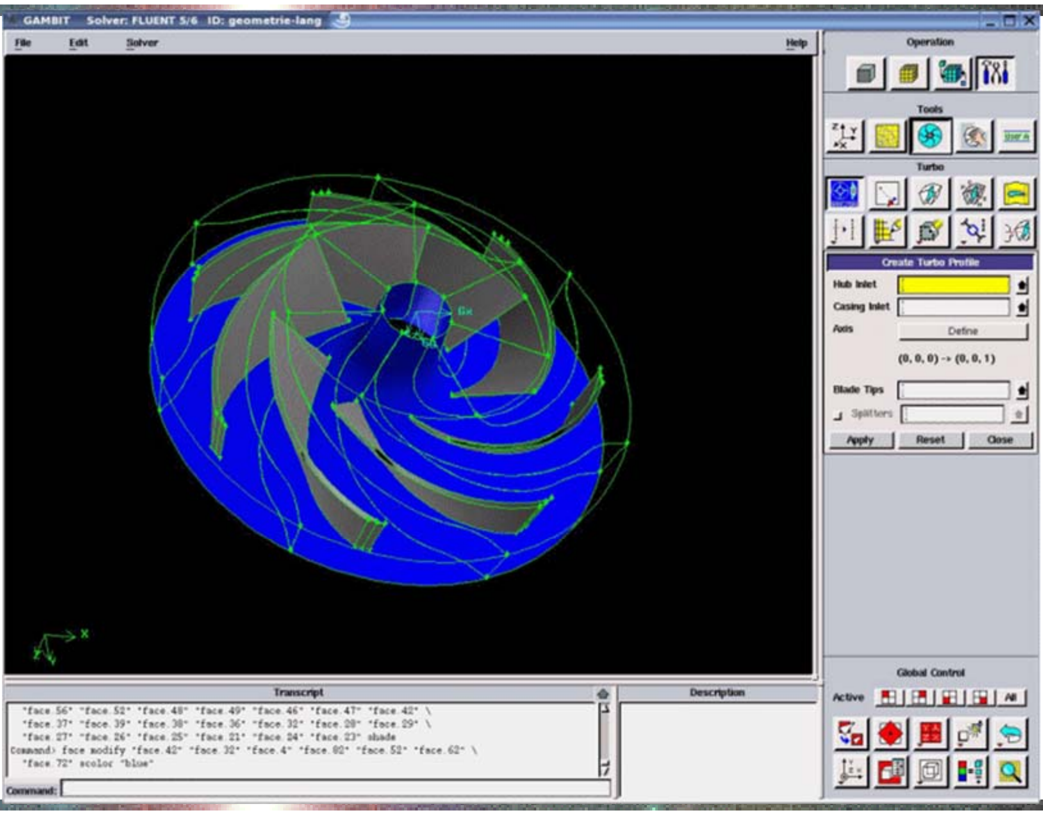
☐ Synchronize camera in active views

☐ Use absolute value of Difference

Differences between designs with throat diameter = 45 [mm] and throat diameter = 60 [mm]

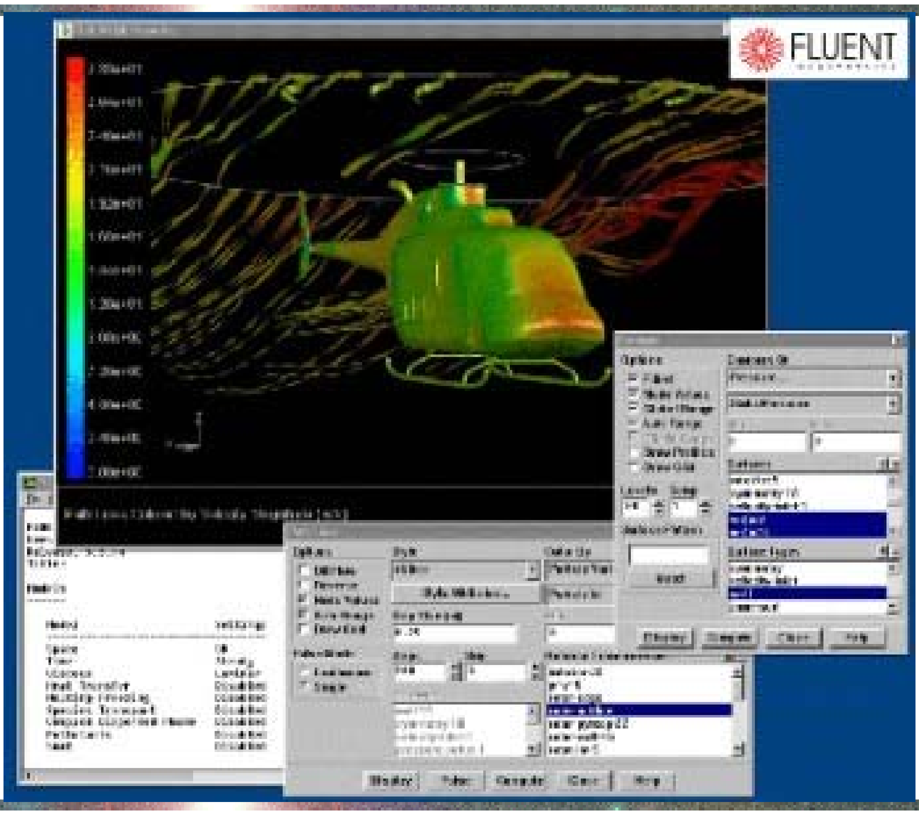
FLUENT

ETH

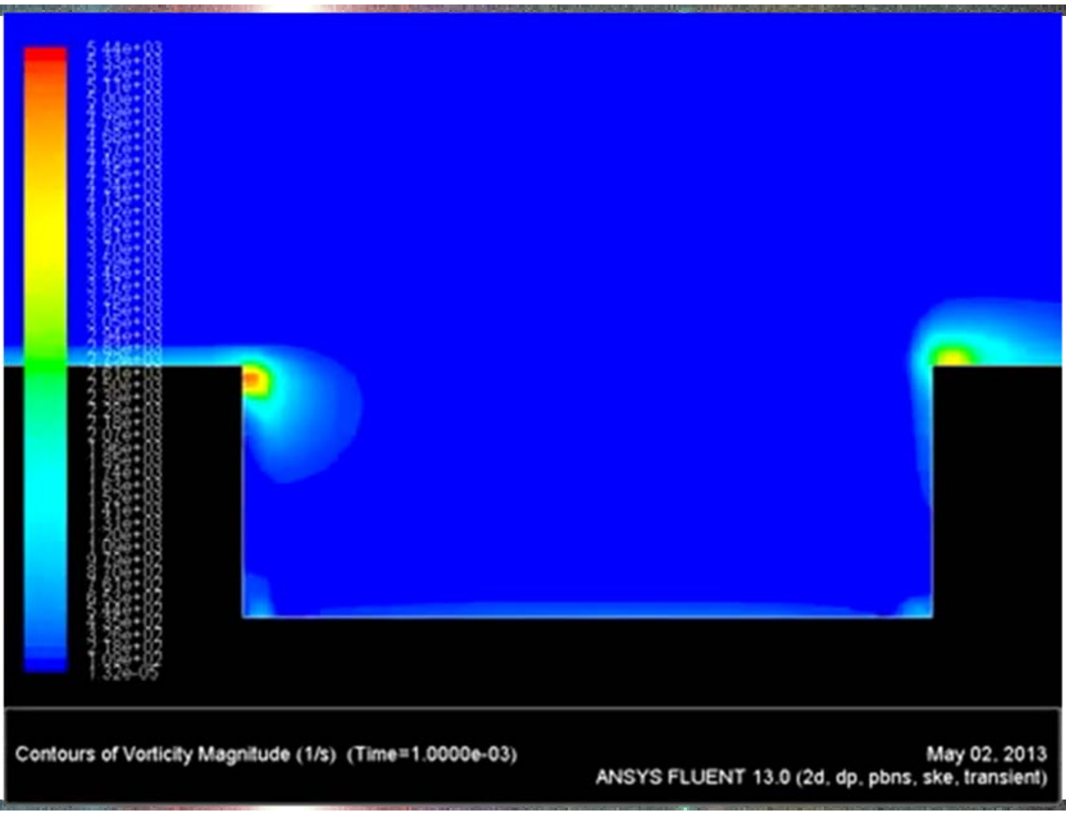


```
  "face.56" "face.52" "face.48" "face.49" "face.46" "face.47" "face.42" \
  "face.37" "face.39" "face.38" "face.36" "face.32" "face.28" "face.29" \
  "face.27" "face.26" "face.25" "face.21" "face.24" "face.23" shade
Command> face modify "face.42" "face.32" "face.4" "face.82" "face.52" "face.62" \
  "face.72" scolor "blue"
```

FLUENT



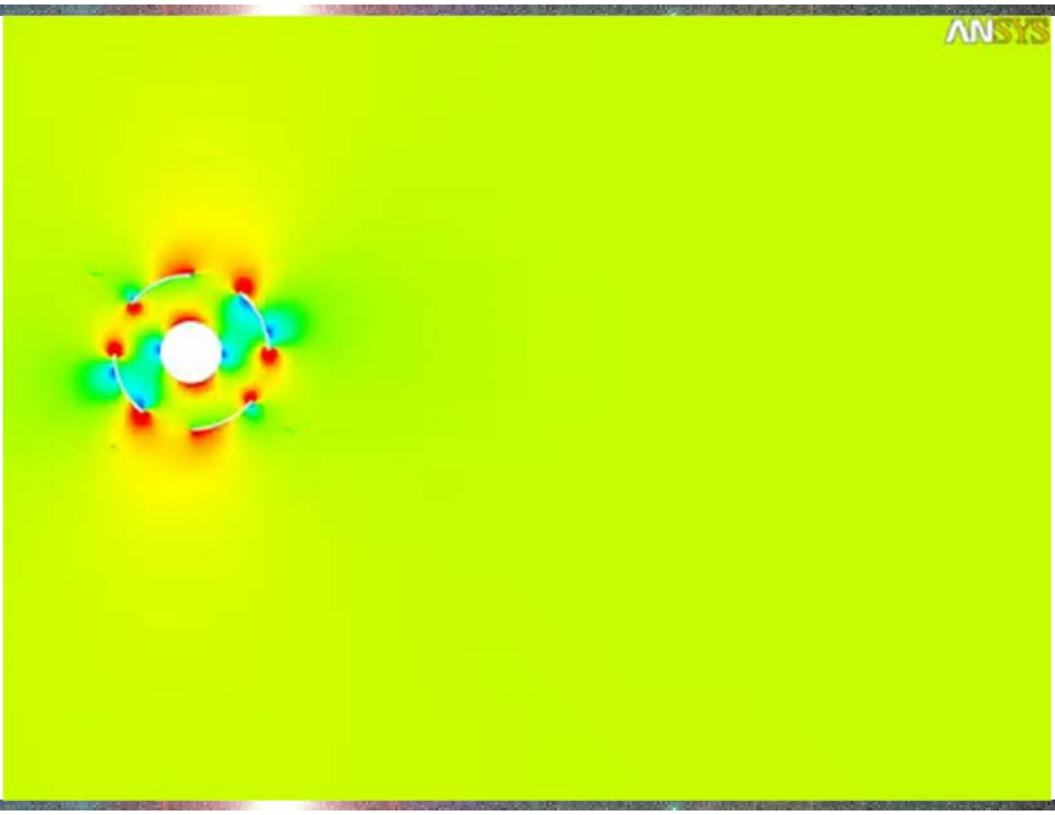
Cavity with FLUENT



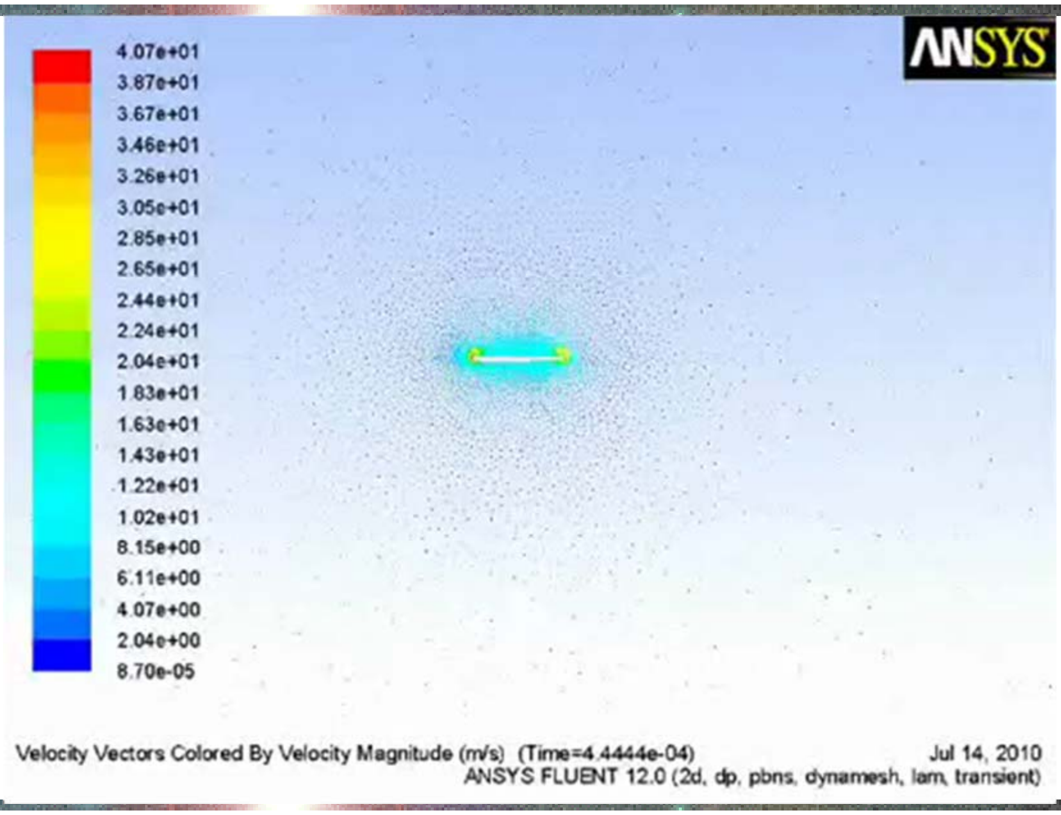
Contours of Vorticity Magnitude (1/s) (Time=1.0000e-03)

May 02, 2013
ANSYS FLUENT 13.0 (2d, dp, pbns, ske, transient)

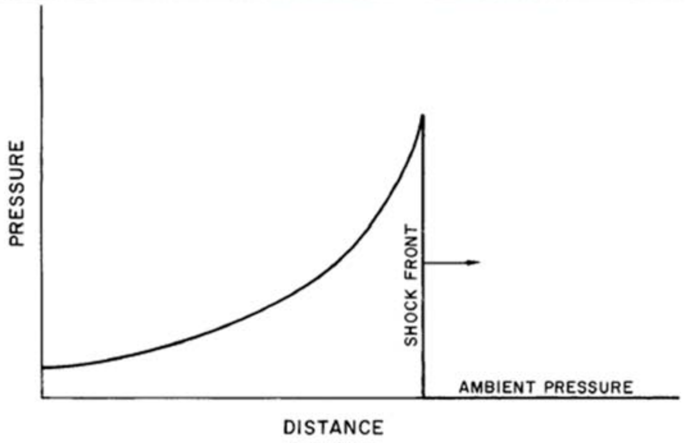
Turbine with FLUENT



Airfoil with FLUENT



Shock waves

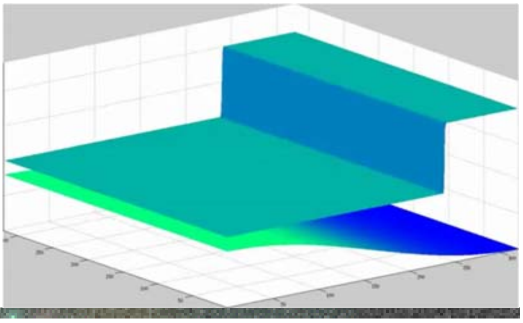


Solutions of parabolic equations which move with constant velocity and develop a sharp front.

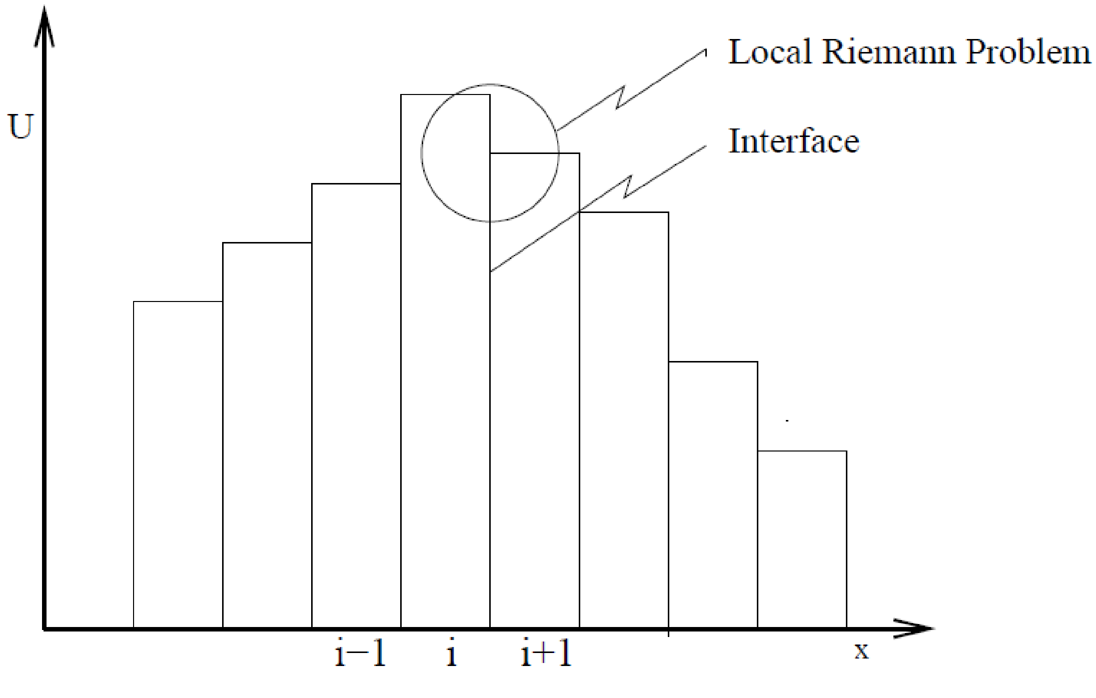
typical initial condition:

Riemann problem

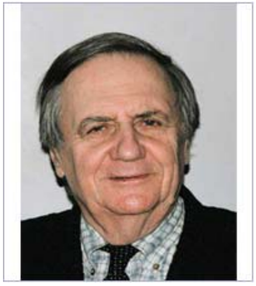
example: tsunami



Shock waves



Godunov Scheme



Sergei K. Godunov (1959)

Example

1d inviscid Burgers equation: $\boxed{\frac{\partial \rho}{\partial t} + \rho \frac{\partial \rho}{\partial x} = 0}$

$$\rho_i(t+\Delta t) = \rho_i(t) + \frac{\Delta t}{\Delta x}\Big[\overbrace{F\big(\rho_i(t-\Delta t), \rho_i(t)\big)}^{\text{in-flow}} - \overbrace{F\big(\rho_i(t), \rho_i(t+\Delta t)\big)}^{\text{out-flow}}\Big]$$

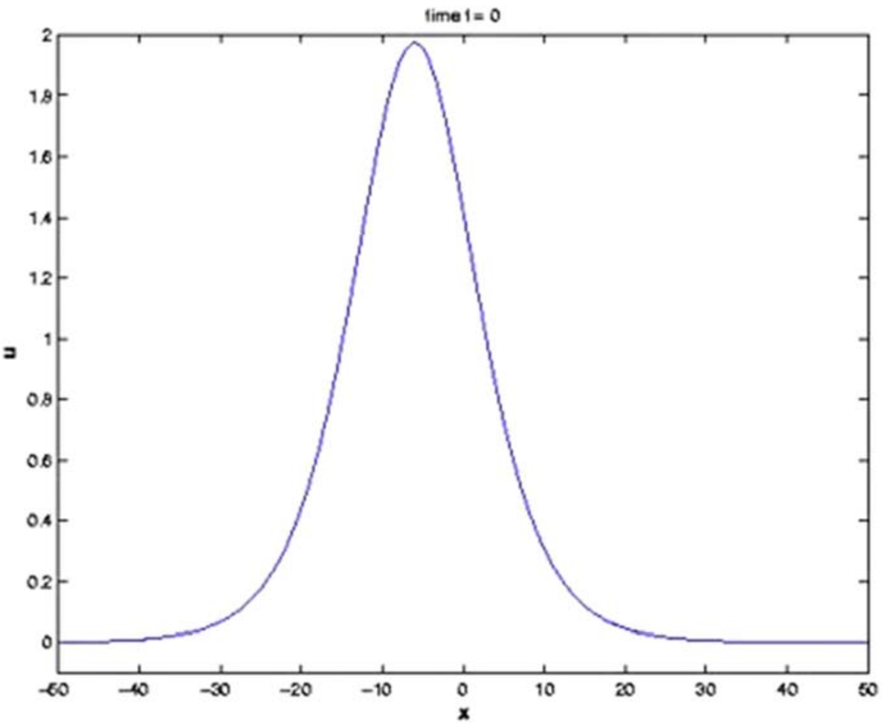
$\boxed{\bar{\rho} = (\rho_L + \rho_R)/2}$

with

$$F(\rho_L, \rho_R) = \frac{g^2}{2} \quad , \; g = \begin{cases} \rho_L & \text{if} \quad \rho_L > 0 \\ \rho_R & \text{if} \quad \rho_R < 0 \\ 0 \text{ if } \rho_L \le 0 < \rho_R \end{cases} \quad \boxed{\rho_L \ge \rho_R} \qquad , \; g = \begin{cases} \rho_L \text{ if } \bar{\rho} > 0 \\ \rho_R \text{ if } \bar{\rho} \le 0 \end{cases} \quad \boxed{\rho_L < \rho_R}$$

1d Burgers equation

formation of shock wave



Spectral Methods

Steve Orszag (1968)

PDE solver for smooth solutions without adaptive meshing. Has excellent convergence properties.

Finite elements:

basis functions: **local** smooth functions

Spectral methods:

basis functions: **global** smooth functions

Spectral Methods

PDE:

$$Lu(x,t) = f(u(x,t))$$

with $u(0,t) = u_B$ and $u(x,0) = u_I(x)$

L differential operator

e.g. $Lu(x,t) = \left(\frac{\partial}{\partial t} - \frac{\partial}{\partial x}\right) u(x,t)$

Expand in terms of basis functions ϕ_i:

$$u(x,t) = \sum_{i=1}^{\infty} a_i(t)\phi_i(x) \approx u_N(x,t) = \sum_{i=1}^{N} a_i(t)\phi_i(x)$$

Spectral Methods

Define *N* (orthogonal) test functions $w_j(x)$:

$$\int_0^L \left[Lu(x,t) + f(u(x,t)) \right] w_j\left(x\right) \, dxdt = 0 \quad , \quad j = 1, ..., N$$

$w_j(x) = \phi_j(x)$ is called the Galerkin method

and $w_j(x) = \delta(x\text{-}x_j)$ is called a collocation.

Spectral Methods

Example 1: 1d advection equation

$$\frac{\partial u}{\partial t} - \frac{\partial u}{\partial x} = 0 \quad \text{on} \quad (0, 2\pi)$$

truncated expansion:

$$u^{(N)}(x,t) = \sum_{l=-N/2}^{N/2} a_l(t)\phi_l(x)$$

Spectral Methods

trigonometric basis and test functions:

$$\boxed{\phi_l(x) = e^{ilx} \quad \text{and} \quad w_k(x) = \frac{1}{2\pi} e^{-ikx}}$$

$$\boxed{\int_0^{2\pi} e^{i(l-k)x} dx = 2\pi\delta_{lk}}$$

$\Rightarrow$

$$\boxed{\frac{1}{2\pi}\int_0^{2\pi}\left[\left(\frac{\partial}{\partial t} - \frac{\partial}{\partial x}\right)\sum_{l=-N/2}^{N/2} a_l(t) e^{ilx}\right] e^{-ikx} dx = 0}$$

$$\Rightarrow \quad \boxed{\frac{da_k}{dt} - ika_k = 0 \quad , \quad \forall k = -\frac{N}{2}, \ldots, \frac{N}{2}}$$

Spectral Methods

solve **initial condition**

$$\boxed{\frac{da_k}{dt} - ika_k = 0} \quad \text{with} \quad \boxed{a_k(0) = \int_0^{2\pi} u_I(x)\, e^{-kx} dx}$$

choose for instance $\boxed{u_I(x) = \sin\left(\pi\cos(x)\right)}$

Bessel function

$$\Rightarrow \quad \boxed{a_k(t) = \sin\left(\frac{k\pi}{2}\right) J_k(\pi)\, e^{ikt}}$$

Spectral Methods

$$\boxed{a_k(t) = \sin\left(\frac{k\pi}{2}\right) J_k(\pi)\, e^{ikt}}$$

From asymptotic behaviour of Bessel functions:

$$\boxed{\forall p: \quad k^p a_k(t) \rightarrow 0 \quad \text{for} \quad k \rightarrow \infty}$$

$$\Rightarrow \quad \boxed{u^{(N)}(x,t) = \sum_{k=-N/2}^{N/2} a_k(t)\, e^{ikx}}$$

converges faster than any power of 1/*N*.

Spectral Methods

Example 2: 1d (full) Burgers equation

$$\boxed{\partial_t u + u \partial_x u = \mu\, \partial_{xx} u}$$

integral or «weak» form, $\forall w, \forall t$:

$$\boxed{\langle \partial_t u, w \rangle + \langle u \partial_x u, w \rangle = \mu\, \langle \partial_{xx} u, w \rangle} \quad \text{with} \quad \boxed{\langle f, w \rangle = \int_0^{2\pi} f(x) \overline{w}(x) dx}$$

Fourier-Galerkian expansion

$$\boxed{u^{(N)}(x,t) = \sum_{k=-N/2}^{N/2} a_k(t)\, e^{ikx}} \qquad \boxed{w(x) = e^{ikx} \quad , \; k = -\frac{N}{2}, \ldots, \frac{N}{2}}$$

Spectral Methods

$$\boxed{\langle \partial_t u, w\rangle + \langle u\partial_x u, w\rangle = \mu\ \langle \partial_{xx} u, w\rangle}$$

$$\boxed{\langle \partial_t u, e^{ikx}\rangle = \left\langle \partial_x\left(-\frac{1}{2}u^2 + \mu\,\partial_x u\right), e^{ikx}\right\rangle}$$

integrating by parts:

$$\boxed{\langle \partial_t u, e^{ikx}\rangle = \left\langle \frac{1}{2}u^2 - \mu\,\partial_x u, \partial_x e^{ikx}\right\rangle = \left\langle \frac{1}{2}u^2 - \mu\,\partial_x u, ike^{ikx}\right\rangle}$$

Spectral Methods

to solve $$\boxed{\langle \partial_t u, e^{ikx}\rangle = \left\langle \frac{1}{2}u^2 - \mu\,\partial_x u, ike^{ikx}\right\rangle}$$

use orthogonality relation $$\boxed{\langle e^{ilx}, e^{ikx}\rangle = \int_0^{2\pi} e^{i(l-k)x}dx = 2\pi\delta_{lk}}$$

$$\boxed{\langle \partial_t u, e^{ikx}\rangle = \left\langle \sum_{l=-N/2}^{N/2} \partial_t a_l(t)\, e^{ilx}, e^{ikx}\right\rangle = 2\pi\,\partial_t a_k}$$

$$\left\langle \frac{1}{2}u^2 - \mu\,\partial_x u, ike^{ikx}\right\rangle = \left\langle \frac{1}{2}\sum_{l,m=-N/2}^{N/2} a_k a_l\ e^{i(l+m)x} - i\mu\sum_{l=-N/2}^{N/2} l\ a_l\ e^{ilx}, ike^{ikx}\right\rangle$$

$$= -\frac{ik}{2}\left\langle \sum_{l,m} a_k a_l\ e^{i(l+m)x}, e^{ikx}\right\rangle - \mu k\left\langle \sum_{l} l\ a_l\ e^{ilx}, e^{ikx}\right\rangle = -i\pi k\sum_{l+m=k} a_m a_l - 2\pi\ \mu k^2 a_k$$

Spectral Methods

$$\boxed{2\pi\,\partial_t a_k = -i\pi k \sum_{l+m=k} a_k a_l - 2\pi\mu k^2 a_k}$$

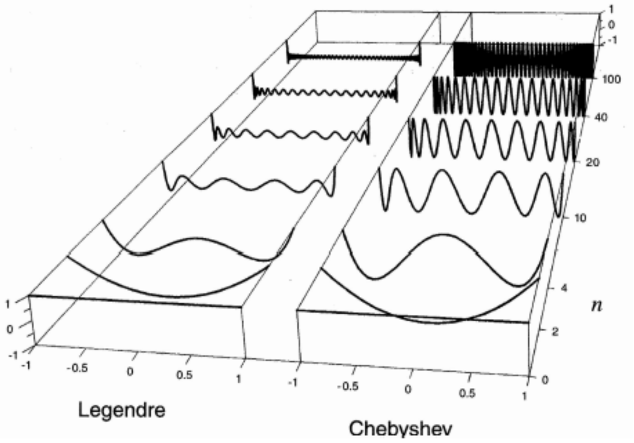
$$\boxed{\frac{\partial a_k}{\partial t}(t) = -\frac{ik}{2} \sum_{l+m=k} a_k(t) a_l(t) - \mu k^2 a_k(t)}$$

This system of coupled ODE can be solved e.g. with Runge Kutta using the Fourier transformed initial condition:

$$\boxed{a_k(0) = \frac{1}{2\pi}\left\langle u(x,0), e^{ikx}\right\rangle = \frac{1}{2\pi}\int_0^{2\pi} u(x,0)e^{-ikx}\,dx}$$

Spectral Methods with other basis functions

Fourier decomposition is good when functions are periodic. Families of orthogonal polynomials on [-1,1] are Legendre and Chebychev polynomials.



Legendre polynomials:

$$\int_{-1}^{1} P_m(x)P_n(x)\,dx = \frac{2}{2n+1}\delta_{mn}$$

$$P_0(x) = 1,\ P_1(x) = x,\ P_2(x) = \frac{3}{2}x^2 - \frac{1}{2}$$

Chebyshev polynomials:

$$\int_{-1}^{1} T_m(x)T_n(x)\frac{dx}{\sqrt{1-x^2}} = \frac{\pi}{2}(1+\delta_{0n})\,\delta_{mn}$$

$$T_0(x) = 1,\ T_1(x) = x,\ T_2(x) = 2x^2 - 1$$

Laguerre polynomials on [0,∞)

Hermite polynomials on (-∞, ∞)

Discrete fluid solvers

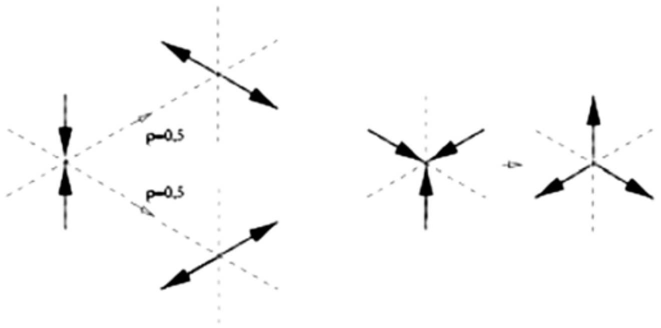
- Lattice Gas Automata (LGA)
- Lattice Boltzmann Method (LBM)
- Dissipative Particle Dynamics (DPD)
- Smooth Particle Hydrodynamics (SPH)
- Stochastic Rotation Dynamics (SRD)
- Direct Simulation Monte Carlo (DSMC)

Lattice gas Automata

- D.H. Rothman and S. Zaleski, „Lattice-Gas Cellular Automata“ (Cambridge Univ. Press, 1997)
- J.-P. Rivet and J.P. Boon, „Lattice Gas Hydrodynamics“ (Cambridge Univ. Press, 2001)
- D.A. Wolf-Gladrow, „Lattice-Gas Cellular Automata and Lattice Boltzmann Models“ (Lecture Notes, Springer, 2000)

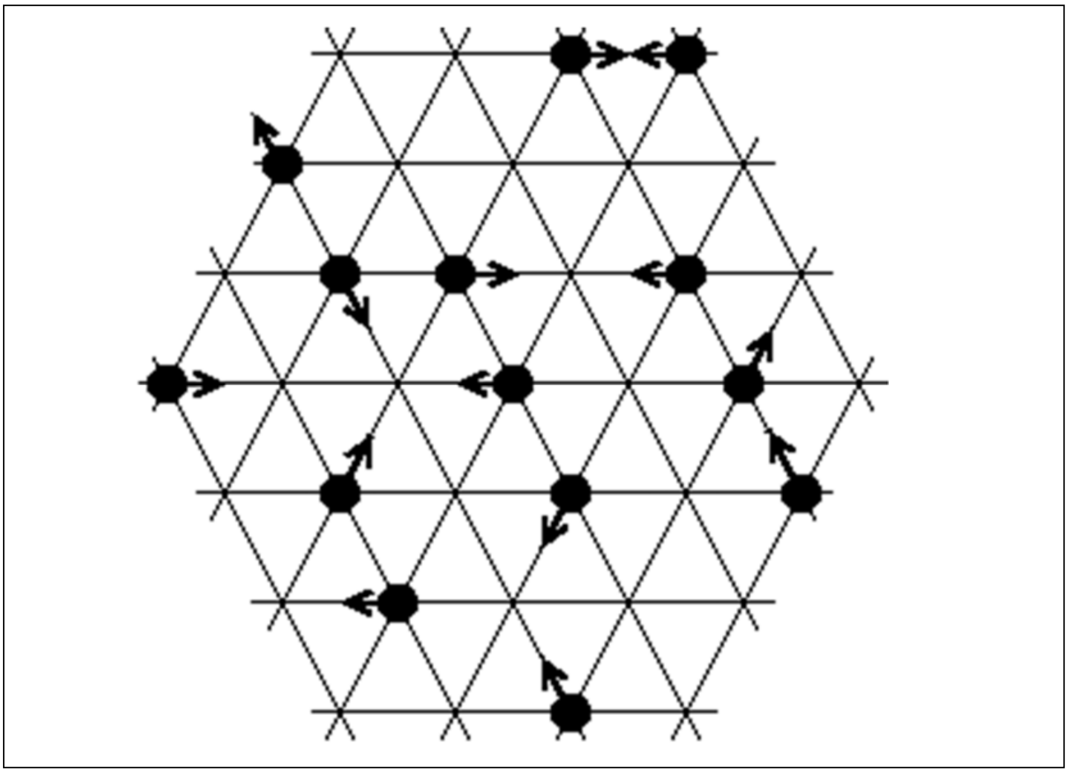
Lattice gas Automata

Particles move on a triangular lattice and follow the following collision rules:



Momentum is conserved at each collision.
It can be proven (Chapman-Enskog) that its continuum limit is the Navier Stokes eq.

Lattice gas Automata



von Karman street

velocity field of a fluid behind an obstacle

Each vector is an average over time of the velocities inside a square cell of 25 triangles.

Lattice gas Automata

Problem in three dimensions, because there exists no translationally invariant lattice which is locally isotropic. One must study the model in 4d and then project down to 3d. Start with 4d face centered hypercube that has 24 directions giving 2^{24} = 1677216 possible states. Projecting onto a 3d hyperplane that already contains 12 directions adds another six new directions giving 18 in 3d.

Discrete fluid solvers

- Lattice Gas Automata (LGA)
- Lattice Boltzmann Method (LBM)
- Dissipative Particle Dynamics (DPD)
- Smooth Particle Hydrodynamics (SPH)
- Stochastic Rotation Dynamics (SRD)
- Direct Simulation Monte Carlo (DSMC)

Lattice Boltzmann

From LGCA to Lattice Boltzmann Models (LBM)

- (Boolean) molecules to (discrete) distributions

$$n_i \longrightarrow f_i = < n_i >$$

n_i is the number of particles in a cell going in direction i

- (Lattice) Boltzmann equations (LBE)

$$\boxed{f_i(\vec{x}+\vec{c}_i,t+1)-f_i(\vec{x},t)=C_i(f)}$$

S.Succi, The Lattice Boltzmann equation for fluid dynamics and beyond, Oxford Univ. Press, 2001

Boltzmann equation

distribution function

Ludwig Boltzmann

$\boxed{f\left(\vec{x},\vec{v},t\right)\Delta\vec{x}\Delta\vec{v}}$ **is the number of particles having at time *t* velocities between $\vec{v}$ and $\vec{v}+\Delta\vec{v}$ in the elementary volume between $\vec{x}$ and $\vec{x}+\Delta\vec{x}$.**

Taylor expansion:

$$f\left(\vec{x}+\Delta\vec{x},\vec{v}+\Delta\vec{v},t+\Delta t\right)=f\left(\vec{x},\vec{v},t\right)+\Delta t\ \partial_t f+\Delta\vec{x}\ \partial_x f+\Delta\vec{v}\ \partial_v f$$

$$\boxed{\lim_{\Delta t\to 0}\frac{f\left(\vec{x}+\Delta\vec{x},\vec{v}+\Delta\vec{v},t+\Delta t\right)-f\left(\vec{x},\vec{v},t\right)}{\Delta t}=\partial_t f+\vec{v}\ \partial_x f+\vec{a}\ \partial_v f}$$

$$\vec{a}=\lim_{\Delta t\to 0}\frac{\Delta\vec{v}}{\Delta t}$$

Boltzmann equation

Due to collisions between particles in the volume $\Delta\vec{x}$ during the time interval Δt some additional $\Delta f_{coll}^{+}\left(\vec{x},\vec{v},t\right)$ particles acquire velocities between $\vec{v}$ and $\vec{v}+\Delta\vec{v}$ and some $\Delta f_{coll}^{-}\left(\vec{x},\vec{v},t\right)$ particles do not anymore have velocities between $\vec{v}$ and $\vec{v}+\Delta\vec{v}$, giving the collision term:

$$\boxed{\Omega_{coll}=\Delta f_{coll}^{+}\left(\vec{x},\vec{v},t\right)-\Delta f_{coll}^{-}\left(\vec{x},\vec{v},t\right)}$$

Boltzmann equation

This gives the Boltzmann equation:

$$\boxed{\partial_t f + \vec{v}\ \partial_x f + \vec{a}\ \partial_v f = \Omega_{coll}}$$

In thermal equilibrium one expects the Maxwell-Boltzmann distribution:

$$\boxed{f^{eq} = \frac{\rho_n}{\sqrt{2\pi kT}}\ e^{-\frac{(\vec{v}-\vec{u})^2}{2kT/m}}}$$

$$\boxed{\vec{u}(\vec{x},t)}$$

BGK collision term

P.L. Bhatnagar, E.P. Gross and M. Krook (1954)

P.L. Bhatnagar

BGK model: $$\boxed{\Omega_{coll} = \frac{f - f^{eq}}{\tau}}$$

where τ is a relaxation time

$$\boxed{\tau = \frac{\mu m}{kT} = \frac{\mu}{c_s^2}}$$

c_s is «sound speed»

μ is viscosity

$$\boxed{c_s^2 \equiv \frac{kT}{m}}$$

Averaged quantities

Moments of the velocity distribution:

mass density:

$$\boxed{\rho(\vec{x},t)=\int m f(\vec{x},\vec{v},t)\,d\vec{v}}$$

momentum density:

$$\boxed{\rho(\vec{x},t)\vec{u}(\vec{x},t)=\int m\;\vec{v}\;f(\vec{x},\vec{v},t)\,d\vec{v}}$$

energy density:

$$\boxed{\rho(\vec{x},t)e(\vec{x},t)=\int m\;\frac{(\vec{v}-\vec{u})^2}{2}\;f(\vec{x},\vec{v},t)\,d\vec{v}}$$

Knudsen number

Validity of the continuum description:
characteristic length of system *L* must be much larger than the mean free path *l* of the molecules (distance between two subsequent collisions).

$$\boxed{K=l/L}$$

Navier-Stokes equation: 0.01 > *K*

Boltzmann equation: 0.005 > *K*

Chapman-Enskog

Chapman-Enskog expansion

$$\boxed{f=\sum_{n=0}^{\infty}K^n f^{(n)}}$$

where the small parameter *K* is the Knudsen number

$$\boxed{f^{(0)}=f^{eq}}$$

$$\boxed{\nabla_x=\sum_{n=1}^{\infty}K^n\nabla_x^{(n)}\quad,\quad\frac{\partial}{\partial t}=\sum_{n=1}^{\infty}K^n\frac{\partial}{\partial t^{(n)}}}$$

Chapman-Enskog

momentum conservation

$$\boxed{\frac{\partial}{\partial t^{(1)}}\left(\rho\vec{u}\right)+\nabla_x^{(1)}\left(\rho\vec{u}\otimes\vec{u}\right)=-\nabla_x^{(1)}\left(\rho e\right)+\rho\vec{a}}$$

$$\boxed{\frac{\partial f^{(0)}}{\partial t^{(2)}}+\frac{\partial f^{(1)}}{\partial t^{(1)}}}\boxed{\frac{\partial f^{(0)}}{\partial t^{(2)}}+\frac{\partial f^{(1)}}{\partial t^{(1)}}+\vec{v}\nabla_x^{(1)}f^{(1)}+\vec{a}\nabla_v^{(1)}f^{(1)}=-\frac{1}{\tau}f^{(2)}}\boxed{=-\frac{1}{\tau}f^{(2)}}$$

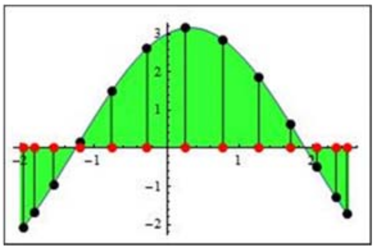
Navier Stokes equation:

$$\boxed{\frac{\partial\rho\vec{u}}{\partial t}+\nabla\Pi=0\quad,\quad\Pi_{xy}=\int\vec{v}\otimes\vec{v}\left(f^{eq}+\left(1-\frac{1}{2\tau}\right)f^{(1)}\right)dv}$$

Gaussian quadrature theorem

Be $g(x)$ a polynomial of at most degree $2n+1$

$$\boxed{\int_a^b g(x)\, w(x)\, dx = \sum_{i=0}^{n} w_i\, g(x_i)}$$



with $$\boxed{w_i = \int_a^b w(x) \prod_{k\neq i} \frac{x - x_k}{x_i - x_k}\, dx \quad , \quad i = 0, \dots, n}$$

if for the positive weight function $w(x)$ there exists a polynomial $p(x)$ of degree $n+1$ such that

$$\boxed{\int_a^b x^k\, p(x)\, w(x)\, dx = 0 \quad , \quad \forall\ k = 0, \dots, n}$$

and x_i , $i = 0,\dots,n$ are the zeros of $p(x)$.

Lattice Boltzmann

$$\boxed{f^{eq} = \frac{\rho}{m\left(2\pi kT/m\right)^{d/2}}\, e^{-\frac{\left(\vec{v}-\vec{u}\right)^2}{2kT/m}}}$$

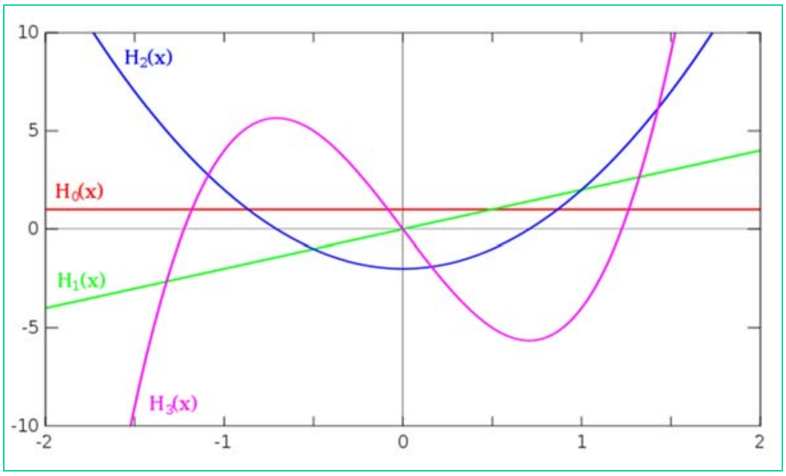
small parameter:

$$\boxed{\frac{\|\vec{u}\|}{c_s} \qquad \text{with} \qquad c_s^2 \equiv \frac{kT}{m}}$$

$$f^{eq} \approx \underbrace{\frac{\rho}{m\left(2\pi c_s^2\right)^{d/2}}\, e^{-\frac{\vec{v}^2}{2c_s^2}}}_{\boxed{w(x)}} \underbrace{\left[1 + \frac{\vec{v}\vec{u}}{c_s^2} + \frac{\left(\vec{v}\vec{u}\right)^2}{2c_s^4} - \frac{\vec{u}^2}{2c_s^2}\right]}_{\boxed{p(x)}\ \boxed{= \sum_{i=0}^{2} a_i H_i(\vec{v})}}$$

Hermite Polynomials

$$H_0(x)=1\ ,\ H_1(x)=x\ ,\ H_2(x)=x^2-1\ ,\ H_3(x)=x^3-3x$$



$$\int_{-\infty}^{\infty} H_i(x)\, H_j(x)\, e^{-x^2}\, dx = \sqrt{2\pi}\ i!\ \delta_{ij}$$

Lattice Boltzmann

one dimensional case:

$$w(v)=\frac{1}{\sqrt{2\pi c_s}}e^{-\frac{v^2}{2c_s^2}}$$

$$n+1=3 \qquad v_i=-\frac{1}{\sqrt{3}},0,\frac{1}{\sqrt{3}}$$

$$w_i=\frac{(n+1)!}{(n+1)^2\left[H_n(v_i)\right]^2}=\left\{\frac{1}{6},\frac{2}{3},\frac{1}{6}\right\}_{i=0,1,2}$$

Lattice Boltzmann

three dimensional case:

$$\boxed{e^{-\frac{\vec{v}^2}{2c_s^2}} = e^{-\frac{\vec{v}_x^{\,2}}{2c_s^2}} e^{-\frac{\vec{v}_y^{\,2}}{2c_s^2}} e^{-\frac{\vec{v}_z^{\,2}}{2c_s^2}}}$$

27 discrete velocity vectors

$$w_{(0,0,0)} = w_0 w_0 w_0 = 8/27$$

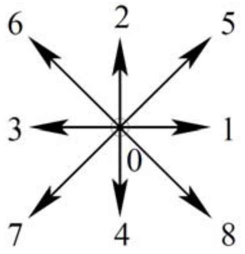
$$w_{(\pm 1/\sqrt{3},0,0)} = w_{(0,\pm 1/\sqrt{3},0)} = w_{(0,0,\pm 1/\sqrt{3})} = w_{1/\sqrt{3}} w_0 w_0 = 2/27$$

$$w_{(\pm 1/\sqrt{3},\pm 1/\sqrt{3},0)} = w_{(0,\pm 1/\sqrt{3},\pm 1/\sqrt{3})} = w_{(\pm 1/\sqrt{3},0,\pm 1/\sqrt{3})} = w_{1/\sqrt{3}} w_{1/\sqrt{3}} w_0 = 1/54$$

$$w_{(\pm 1/\sqrt{3},\pm 1/\sqrt{3},\pm 1/\sqrt{3})} = w_{1/\sqrt{3}} w_{1/\sqrt{3}} w_{1/\sqrt{3}} = 1/216$$

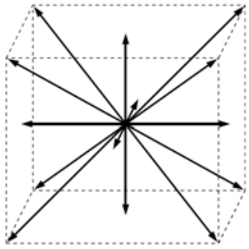
Lattice Boltzmann

D2Q9



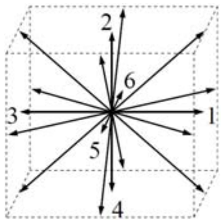
$$w_i = \begin{cases} 4/9 & i = 0 \\ 1/9 & i = 1,2,3,4 \\ 1/36 & i = 5,6,7,8 \end{cases}$$

D3Q15



$$w_i = \begin{cases} 2/9 & i = 0 \\ 1/9 & i = 1-6 \\ 1/72 & i = 7-14 \end{cases}$$

D3Q19



$$w_i = \begin{cases} 1/3 & i = 0 \\ 1/18 & i = 1-6 \\ 1/36 & i = 7-18 \end{cases}$$

Lattice Boltzmann

Define on each site *x* of a lattice on each outgoing bond *i* a velocity distribution function *f*(*x*,*v*$_i$,*t*) which is updated as:

$$\boxed{f_i(x+v_i,v_i,t+1)-f_i(x,v_i,t)+F_i(v_i)=\frac{1}{\tau}\left[f_i^{\,0}(\rho_n,u,T)-f_i(x,v_i,t)\right]}$$

where the equilibrium distribution is defined as:

$$\boxed{f_i^0=\rho_n w_i\left[1+\frac{3\vec{v}\vec{u}}{c_s^2}+\frac{9\left(\vec{v}\vec{u}\right)^2}{2c_s^4}-\frac{3\vec{u}^2}{2c_s^2}\right]}$$

Lattice Boltzmann

discretization

CFL number $\boxed{\dfrac{\left|\vec{v}\right|\Delta t}{\left|\Delta\vec{x}\right|}=1}$

$$\boxed{\tau=\frac{\mu}{c_s^2}+\frac{\Delta t}{2}}$$

$$\boxed{f\left(\vec{x}+\Delta\vec{x},\vec{v}+\Delta\vec{v},t+\Delta t\right)-f\left(\vec{x},\vec{v},t\right)=\Delta t\left(\partial_t+\vec{v}\vec{\nabla}\right)f+\frac{\Delta t^2}{2}\left(\partial_t+\vec{v}\vec{\nabla}\right)^2 f}$$

Multi-Relaxation-Time (MRT) LBM

$$\left| f(\vec{x}+\vec{c}_\alpha \delta t, t+\delta t)\right\rangle_\alpha - \left| f(\vec{x},t)\right\rangle_\alpha = -\sum_{j=0}^{N} \frac{s_j}{\left\langle \phi_j \middle| \phi_j \right\rangle}(m_j - m_j^{eq})\left|\phi_j\right\rangle_\alpha$$

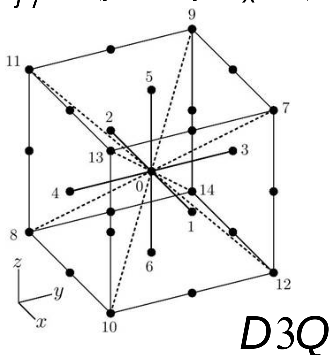
$\left|\phi_i\right\rangle$ Orthogonal polynomials

s_j is the inverse of a relaxation time.

$\left|m_j\right\rangle = \left\langle \phi_j \middle| f \right\rangle$ Projections of the distribution

P. Lallemand and L.S. Luo
Phys.Rev.E 61, 6546 (2000)

$\left|m_j\right\rangle = (\rho, \ldots, \rho u_x, \ldots)$



$D3Q15$

Chapman-Enskog expansion:

$$\mu = c_s^{\,2}\left(\frac{1}{s_{9,\ldots,13}} - \frac{1}{2}\right)$$

Shear viscosity

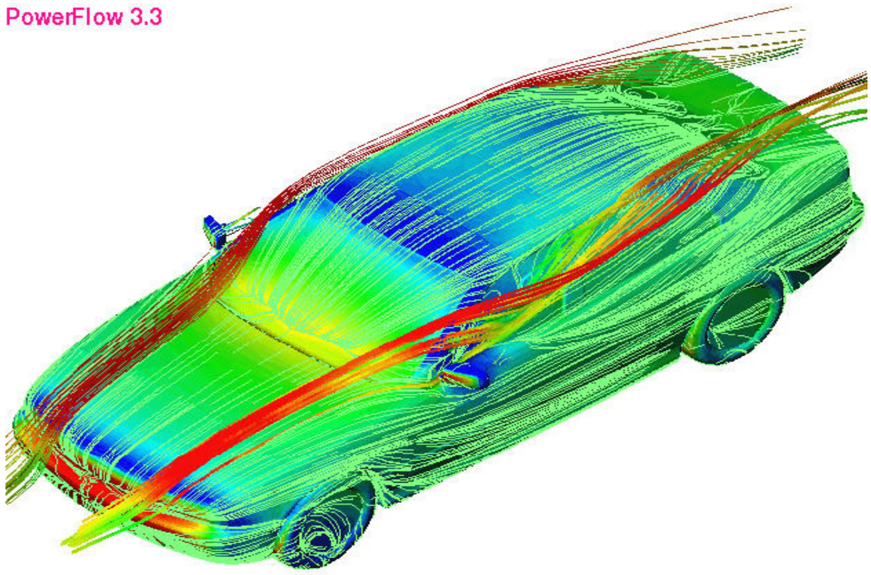
$$\xi = \frac{5-9c_s^{\,2}}{9}\left(\frac{1}{s_2} - \frac{1}{2}\right)$$

Bulk viscosity

Lattice Boltzmann

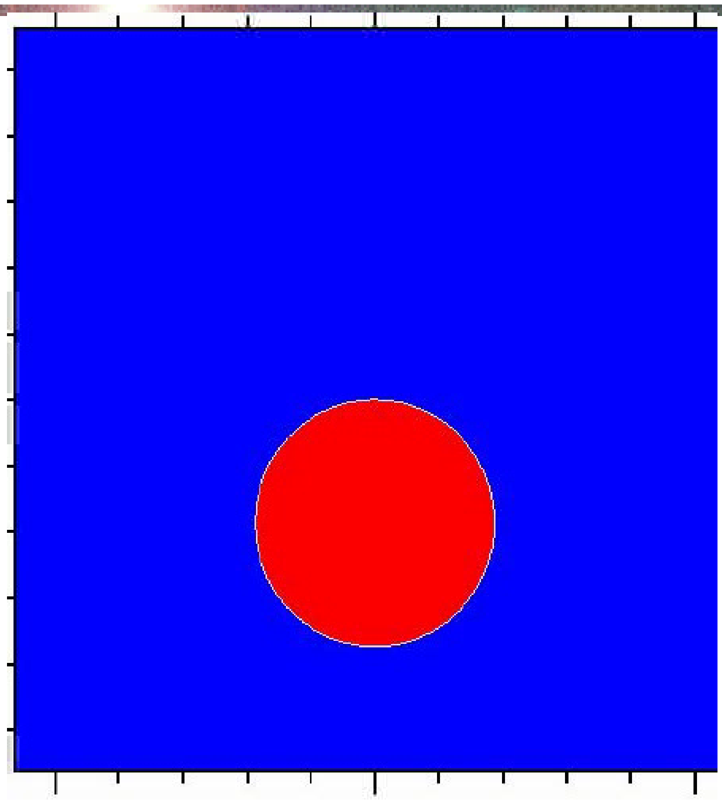
PowerFlow 3.3

Car design

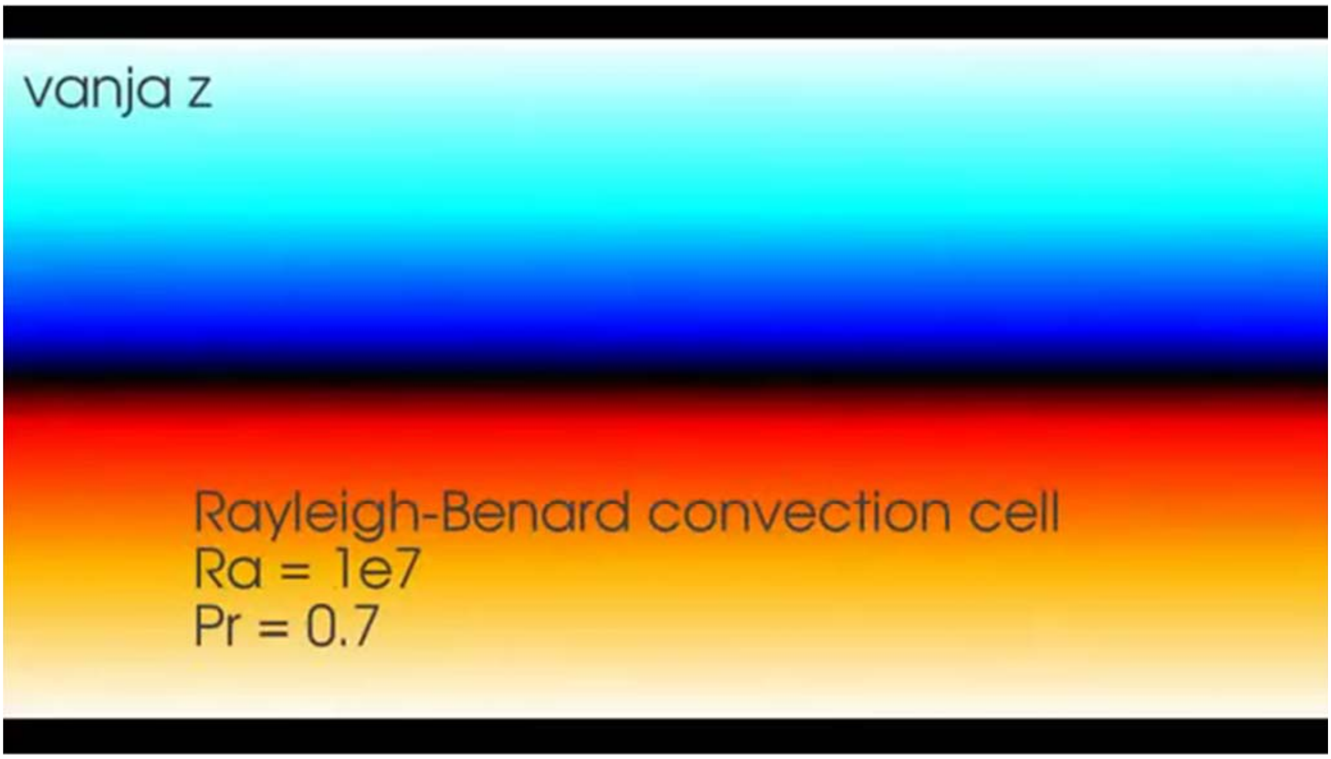


Powerflow, EXA

Raising of a bubble



3d Rayleigh Benard

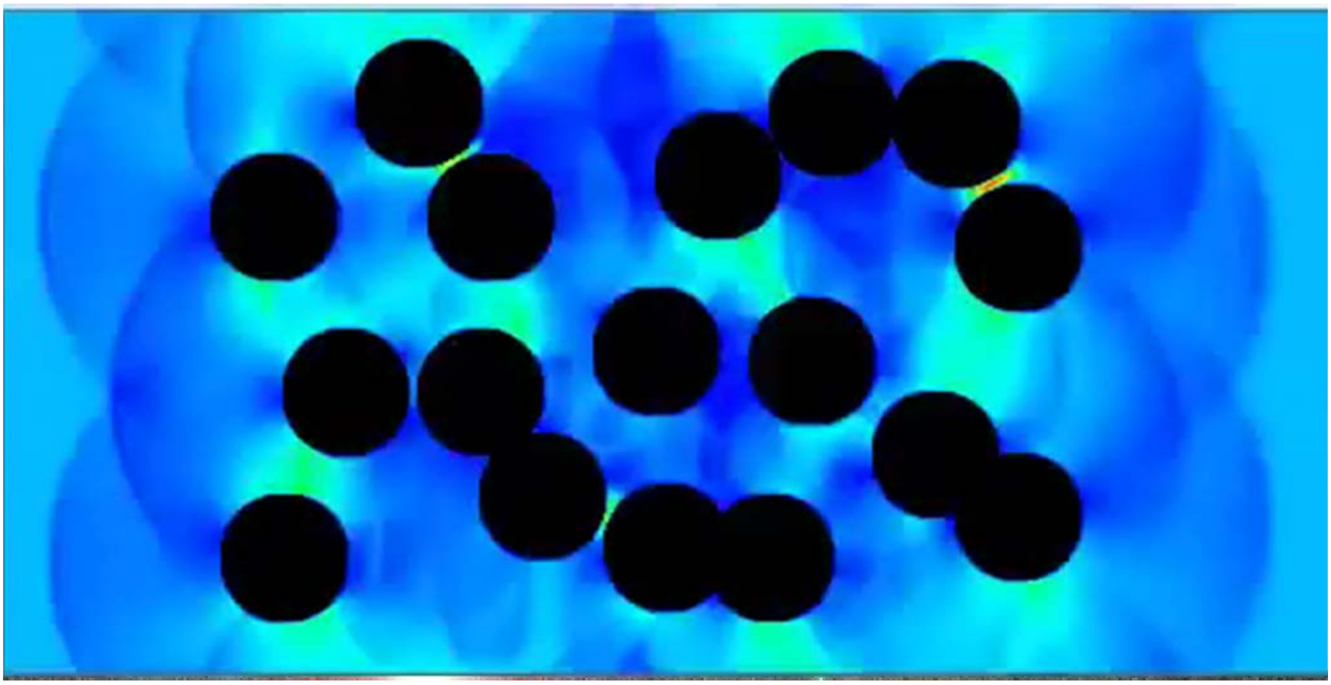


vanja z

Rayleigh-Benard convection cell
Ra = 1e7
Pr = 0.7

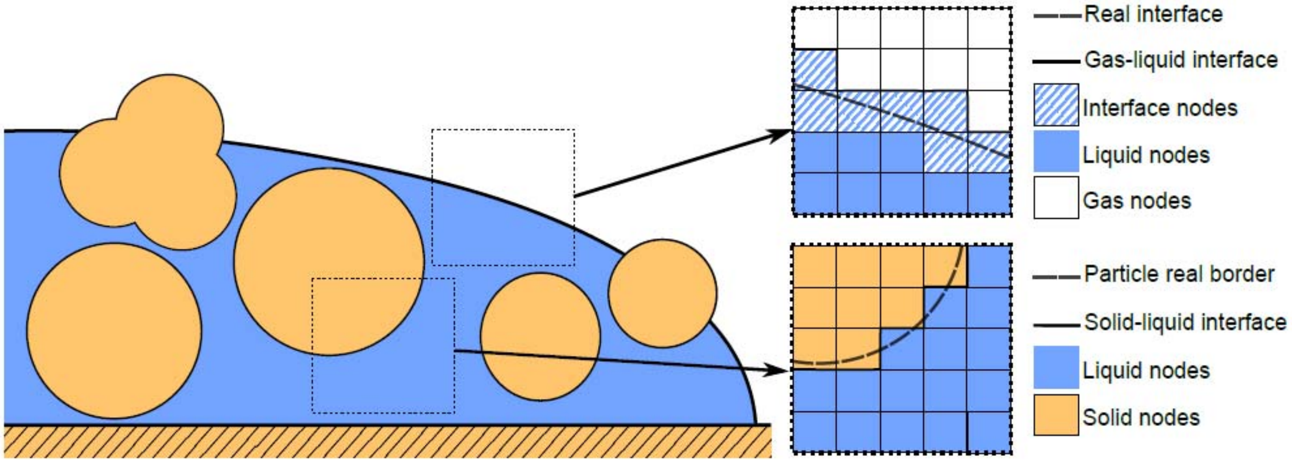
Flow through porous medium in 2d

using a NVidia GTX680



Surface Flow with Moving and Deforming Objects

Interfaces and free surfaces



Discrete fluid solvers

- Lattice Gas Automata (LGA)
- Lattice Boltzmann Method (LBM)
- Dissipative Particle Dynamics (DPD)
- Smooth Particle Hydrodynamics (SPH)
- Stochastic Rotation Dynamics (SRD)
- Direct Simulation Monte Carlo (DSMC)

Smooth Particle Hydrodynamics

- **SPH** describes a fluid by replacing its continuum properties with locally (smoothed) quantities at discrete Lagrangian locations ⇒ meshless
- **SPH** is based on integral interpolants (Lucy 1977, Gingold & Monaghan 1977, Liu 2003)

$$A(\mathbf{r}) = \int_\Omega A(\mathbf{r}')W(\mathbf{r}-\mathbf{r}',h)\mathrm{d}\mathbf{r}'$$

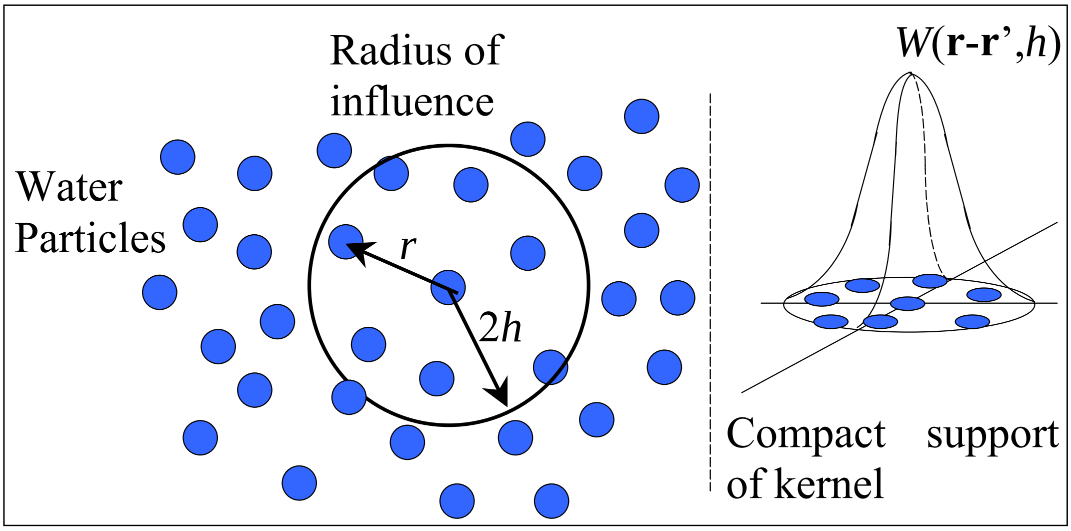
(W is the smoothing kernel)

- These can be approximated discretely by a summation interpolant

$$A(\mathbf{r}) \approx \sum_{j=1}^{N} A(\mathbf{r}_j)W(\mathbf{r}-\mathbf{r}_j, h)\frac{m_j}{\rho_j}$$

Smooth Particle Hydrodynamics

The kernel (or weighting Function)



Radius of influence
Water Particles
r
$2h$
$W(\mathbf{r}\text{-}\mathbf{r}',h)$
Compact support of kernel

- **Example: quadratic kernel**

$$\boxed{W(r,h) = \frac{3}{2\pi h^2}\left(\frac{1}{4}q^2 - q + 1\right)}$$

$$\boxed{q = \frac{r}{h},\ r = |\mathbf{r}_a - \mathbf{r}_b|}$$

Smooth Particle Hydrodynamics

- Spatial gradients are approximated using a summation containing the gradient of the chosen kernel function

$$\nabla A_i = \sum_j \frac{m_j}{\rho_j} A_j \nabla_i W_{ij}$$

$$\rho_i (\nabla . \mathbf{u})_i = \sum_j m_j \left(\mathbf{u}_i - \mathbf{u}_j \right) . \nabla_i W_{ij}$$

- Advantages are:
  - spatial gradients of the data are calculated analytically
  - the characteristics of the method can be changed by using a different kernel

Smooth Particle Hydrodynamics

Equations of Motion

- **Navier-Stokes equations:**

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = -\rho \nabla . \mathbf{v}$$

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -\frac{1}{\rho} \nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{F}_i$$

- **Recast in particle form as:**

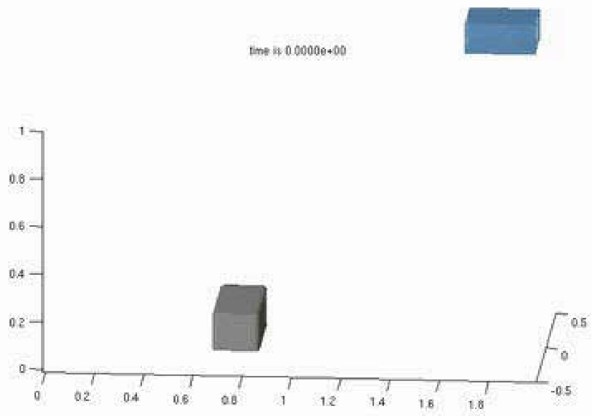
$$\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t} = \mathbf{v}_i + \varepsilon \sum_j m_j \left(\frac{\mathbf{v}_{ji}}{\bar{\rho}_{ij}} \right) W_{ij}$$

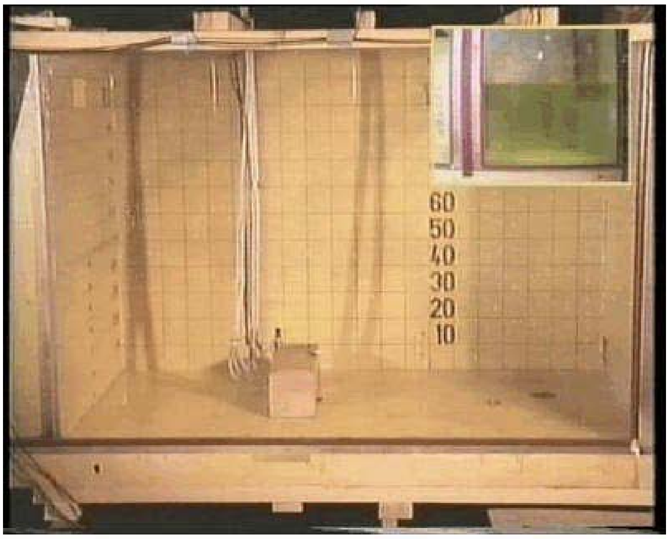
$$\left(\frac{\mathrm{d}m_i}{\mathrm{d}t} = 0 \right)$$

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}t} = \sum_j m_j \left(\mathbf{v}_i - \mathbf{v}_j \right) \cdot \nabla_i W_{ij}$$

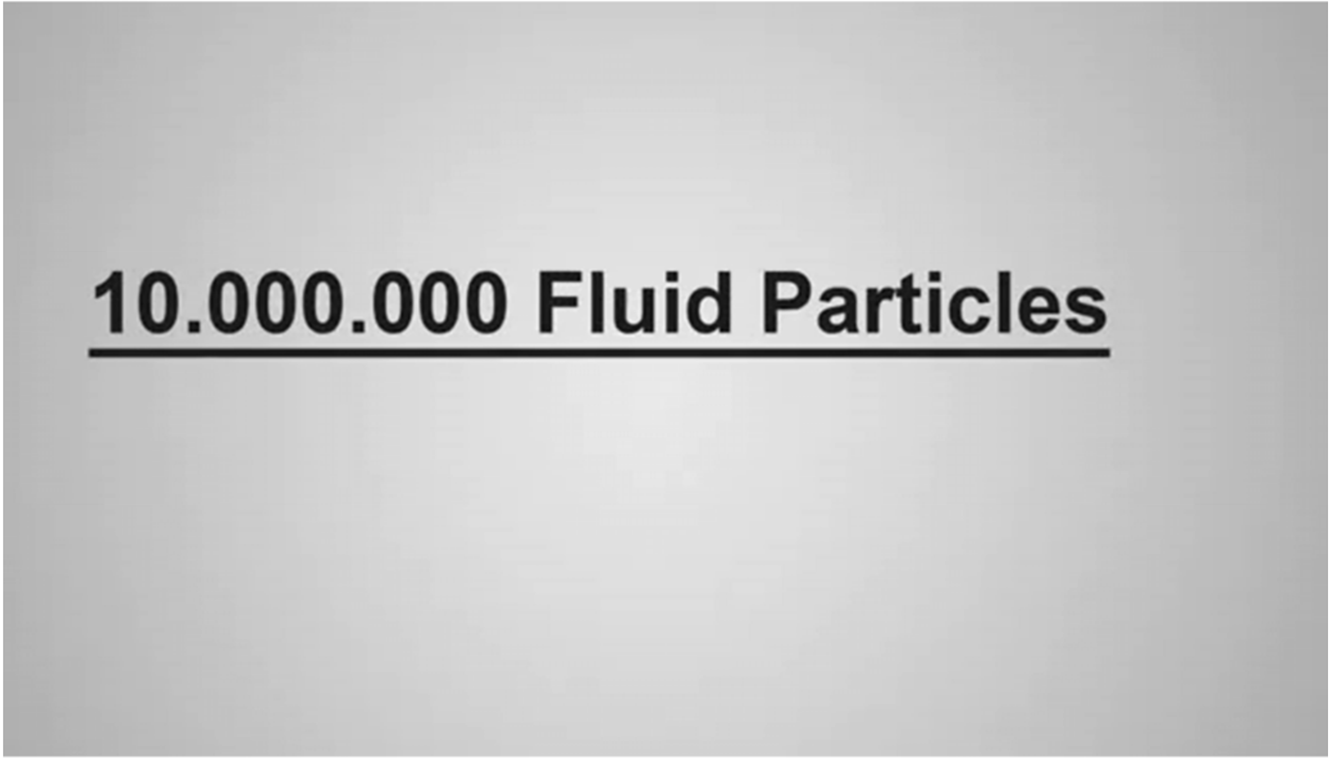
$$\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij} \right) \nabla_i W_{ij} + \mathbf{F}_i$$

Simulation of free surface





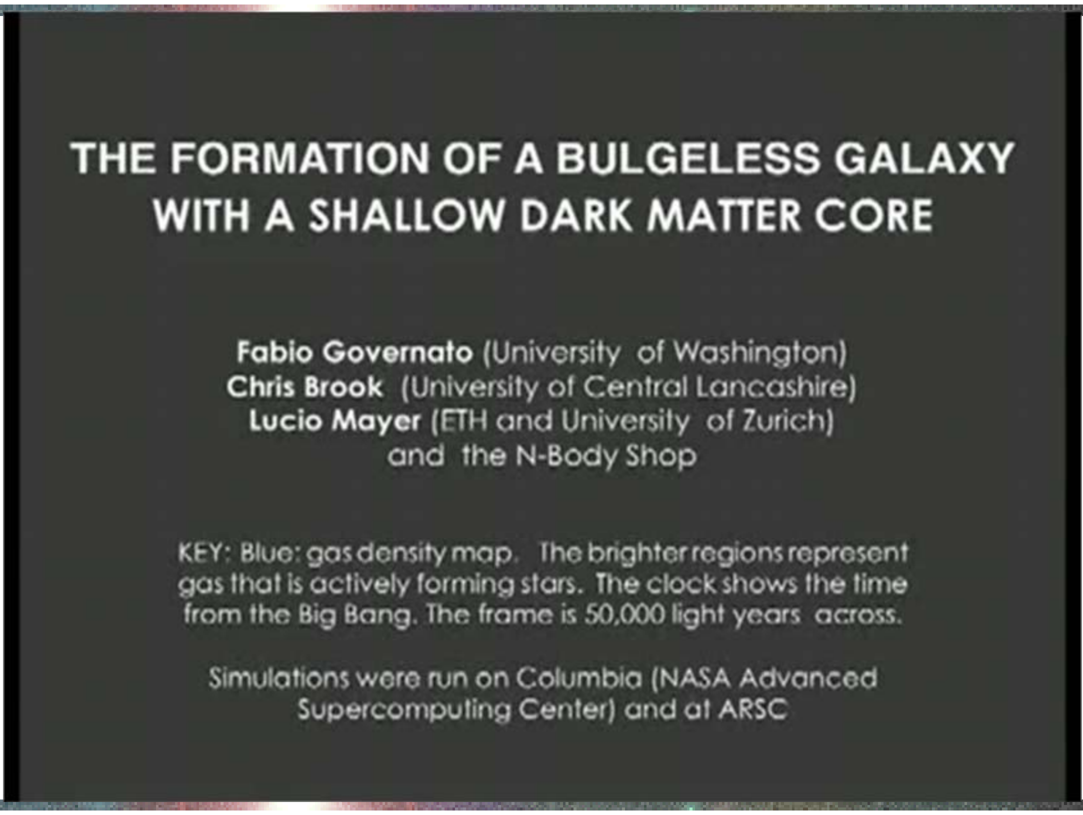
Simulation of free surface



10.000.000 Fluid Particles

Simulation of free surface

Dwarf Galaxy Formation



THE FORMATION OF A BULGELESS GALAXY WITH A SHALLOW DARK MATTER CORE

Fabio Governato (University of Washington)
Chris Brook (University of Central Lancashire)
Lucio Mayer (ETH and University of Zurich)
and the N-Body Shop

KEY: Blue: gas density map. The brighter regions represent gas that is actively forming stars. The clock shows the time from the Big Bang. The frame is 50,000 light years across.

Simulations were run on Columbia (NASA Advanced Supercomputing Center) and at ARSC

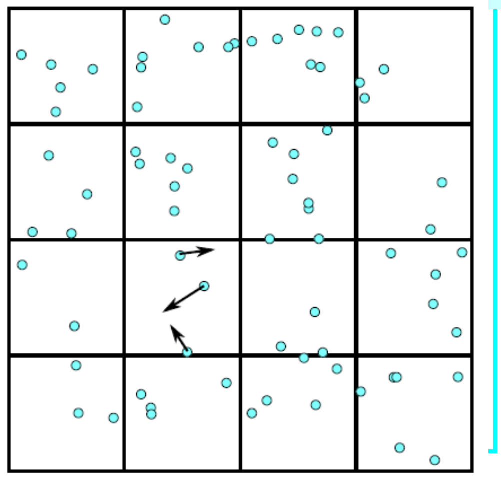
Discrete fluid solvers

- Lattice Gas Automata (LGA)
- Lattice Boltzmann Method (LBM)
- Dissipative Particle Dynamics (DPD)
- Smooth Particle Hydrodynamics (SPH)
- Stochastic Rotation Dynamics (SRD)
- Direct Simulation Monte Carlo (DSMC)

Stochastic Rotation Dynamics

Stochastic Rotation Dynamics (SRD)

- introduction of representative fluid particles
- collective interaction by rotation of local particle velocities
- very simple dynamics, but recovers hydrodynamics correctly
- Brownian motion is intrinsic

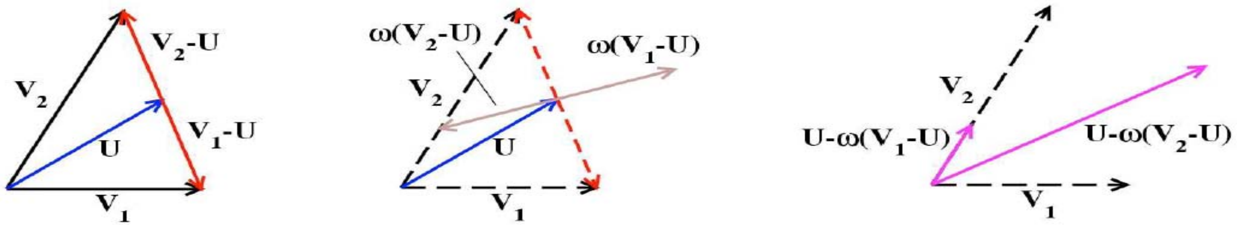


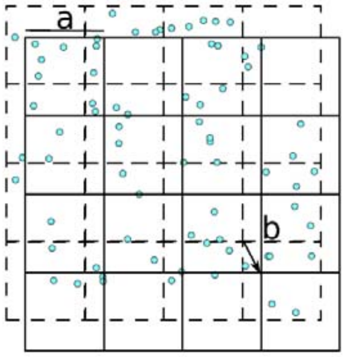
$$
\begin{aligned}
\vec{x}'_n &= \vec{x}_n + \vec{v}_n \,\Delta t \\
\vec{v}'_n &= \vec{u} + \Omega(\vec{v}_n - \vec{u}) + \vec{g} \\
\Omega_z^{\pm} &= \begin{pmatrix} \cos\alpha & \pm\sin\alpha & 0 \\ \mp\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
\vec{u} &= <\vec{v}_n>
\end{aligned}
$$

A. Malevanets, J. Chem. Phys. 110 (1999)
J.T. Padding, A.A. Louis, Phys. Rev. Lett. 93 (2004)

Stochastic Rotation Dynamics

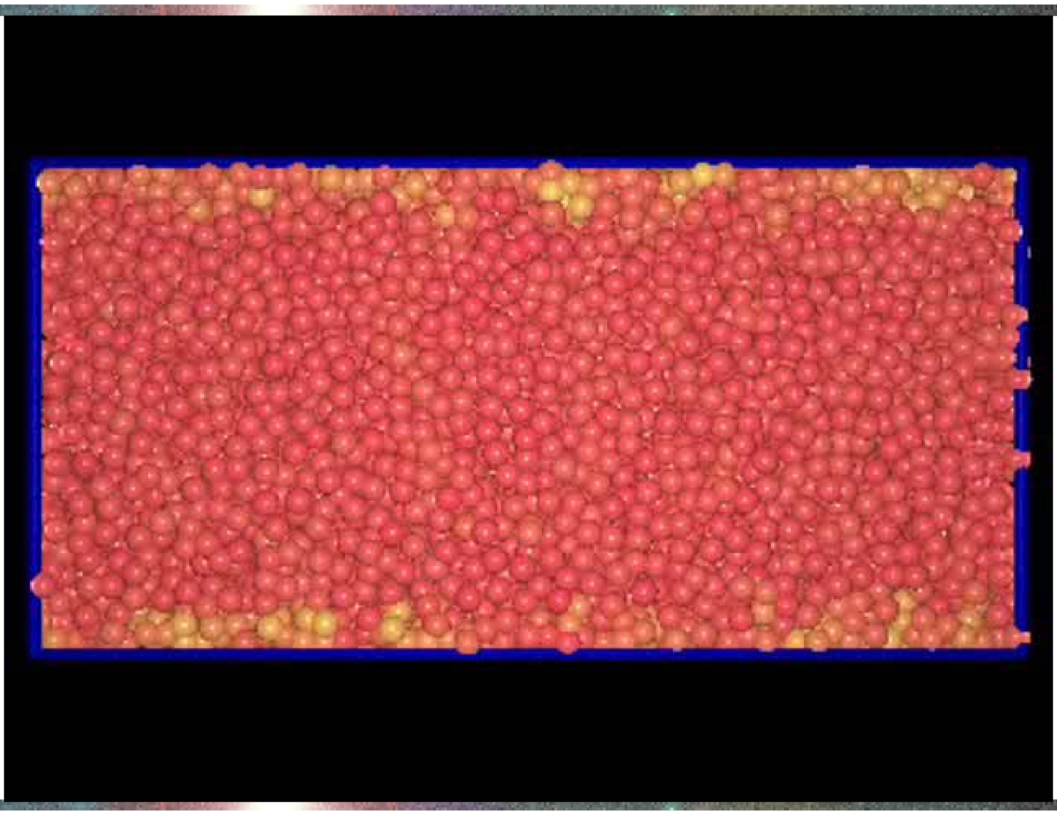
Example of two particles in cell:





Shift grid to impose Galilean invariance.

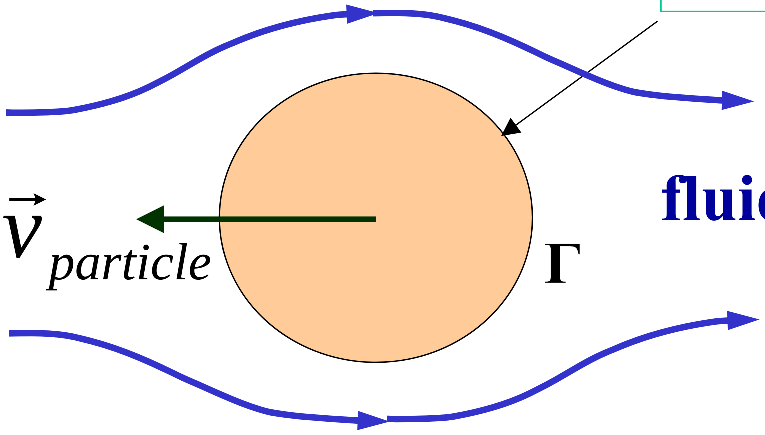
Shear flow



One particle in fluid

e.g. pull sphere through fluid

no-slip condition: $\boxed{\vec{v}_\Gamma = \vec{v}_{particle}}$



$\vec{v}_{particle}$

Γ

fluid

moving boundary condition

create shear in fluid : exchange momentum

Drag force

drag force

(Bernoulli‘s principle)

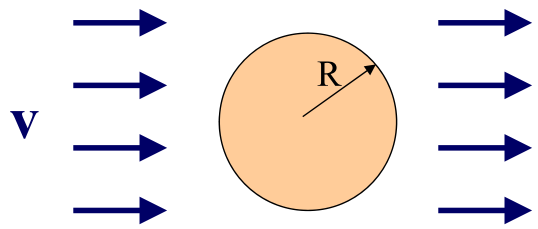
$$\boxed{\vec{F}_D = \int_\Gamma \overleftrightarrow{\Theta} d\vec{A}}$$

stress tensor

$$\boxed{\Theta_{ij} = -p\delta_{ij} + \eta\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)}$$

$\eta = \rho\mu$ is static viscosity

Homogeneous flow



Re << 1   Stokes law:

$$F_D = 6\pi\,\eta\,R\,v$$

(exact for Re = 0)

R is particle radius, v is relative velocity

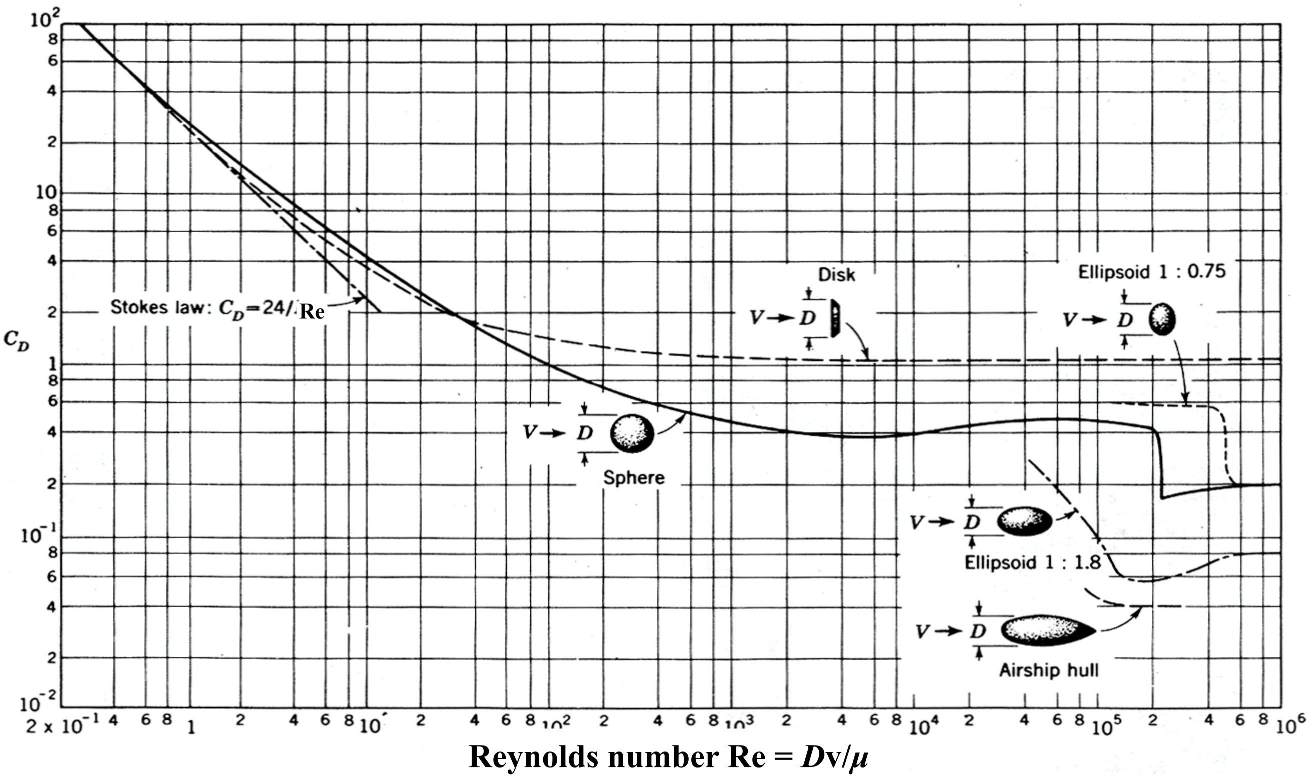
Re >> 1   Newton's law: $F_D = 0.22\pi\,\rho\,R^2 v^2$

general drag law:

$$\boxed{F_D = \frac{\pi}{8}\frac{\eta^2}{\rho} C_D \,\mathrm{Re}^2}$$

C_D is the drag coefficient

Drag coefficient C_D



Inhomogeneous flow

In velocity or pressure gradients: Lift forces

are perpendicular to the direction of the external flow, important for wings of airplanes.

lift force:

$$\boxed{L = C_L \times \rho \times \frac{v^2}{2} \times A}$$

C_L is „lift coefficient“

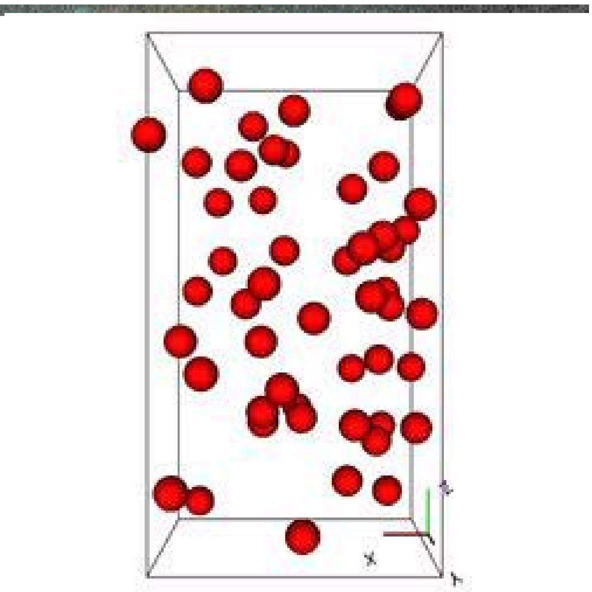
when particle rotates: Magnus effect

important for soccer

Many particles in fluids

- The fluid velocity field follows the incompressible Navier Stokes equations.
- Many industrial processes involve the transport of solid particles suspended in a fluid. The particles can be sand, colloids, polymers, etc.
- The particles are dragged by the fluid with a force:

$$\boxed{F_D = \frac{\pi}{8}\frac{\eta^2}{\rho} C_D \,\mathrm{Re}^2}$$



simulating particles moving in a sheared fluid

Stokes limit

hydrodynamic interaction between the particles

$$\boxed{\vec{v}_i = \sum_{j\neq i} \underbrace{M_{ij}(\vec{r}_i - \vec{r}_j)}_{\text{mobility}-\text{matrix}} \vec{v}_j}$$

for Re = 0 mobility matrix exact

Stokesian Dynamics (Brady and Bossis)

invert a full matrix ⇒ only a few thousand particles

Numerical techniques

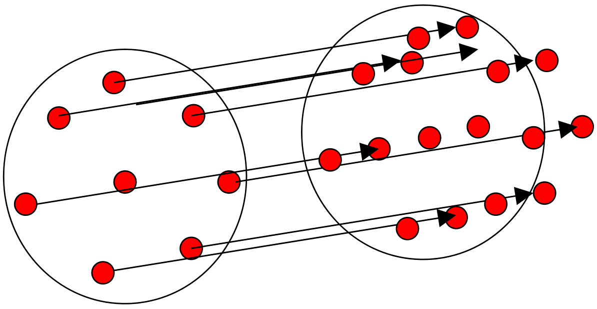
1 Calculate stress tensor directly by evaluating the gradients of the velocity field through interpolation on the numerical grid, e.g. using Chebychev polynomials .

2 Method of Fogelson and Peskin:
Advect markers that were placed in the particle and then put springs between their new an their old position.
These springs then pull the particle.

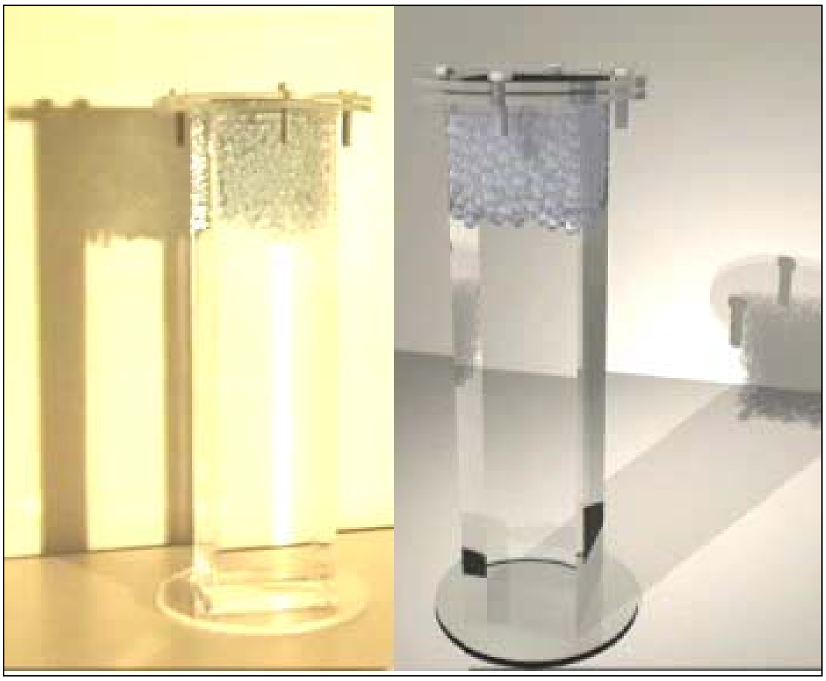
Numerical techniques

2

Method of A.L. Fogelson and C.S. Peskin: Advect markers that were placed in the particle and then put springs between their new an their old position. These springs then pull the particle.



Sedimentation



Glass beads descending in silicon oil

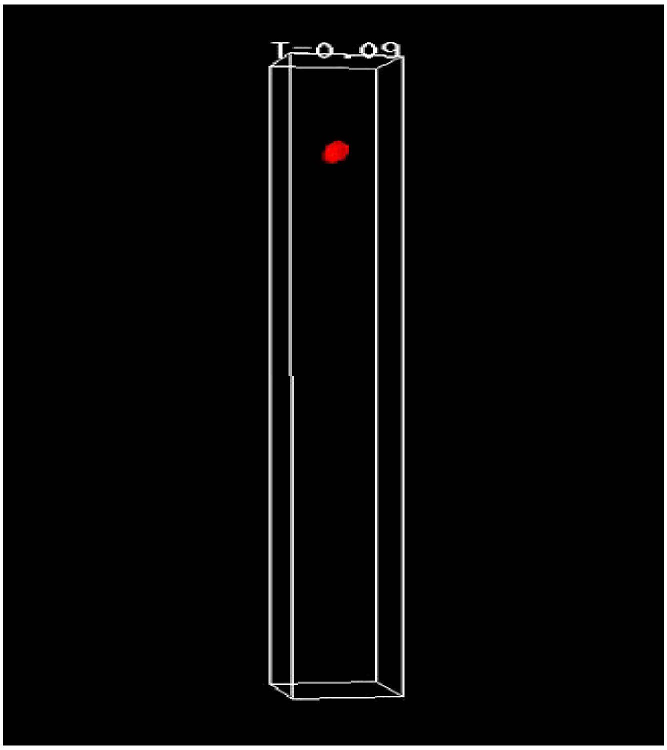
comparing experiment and simulation

Sedimentation of platelets

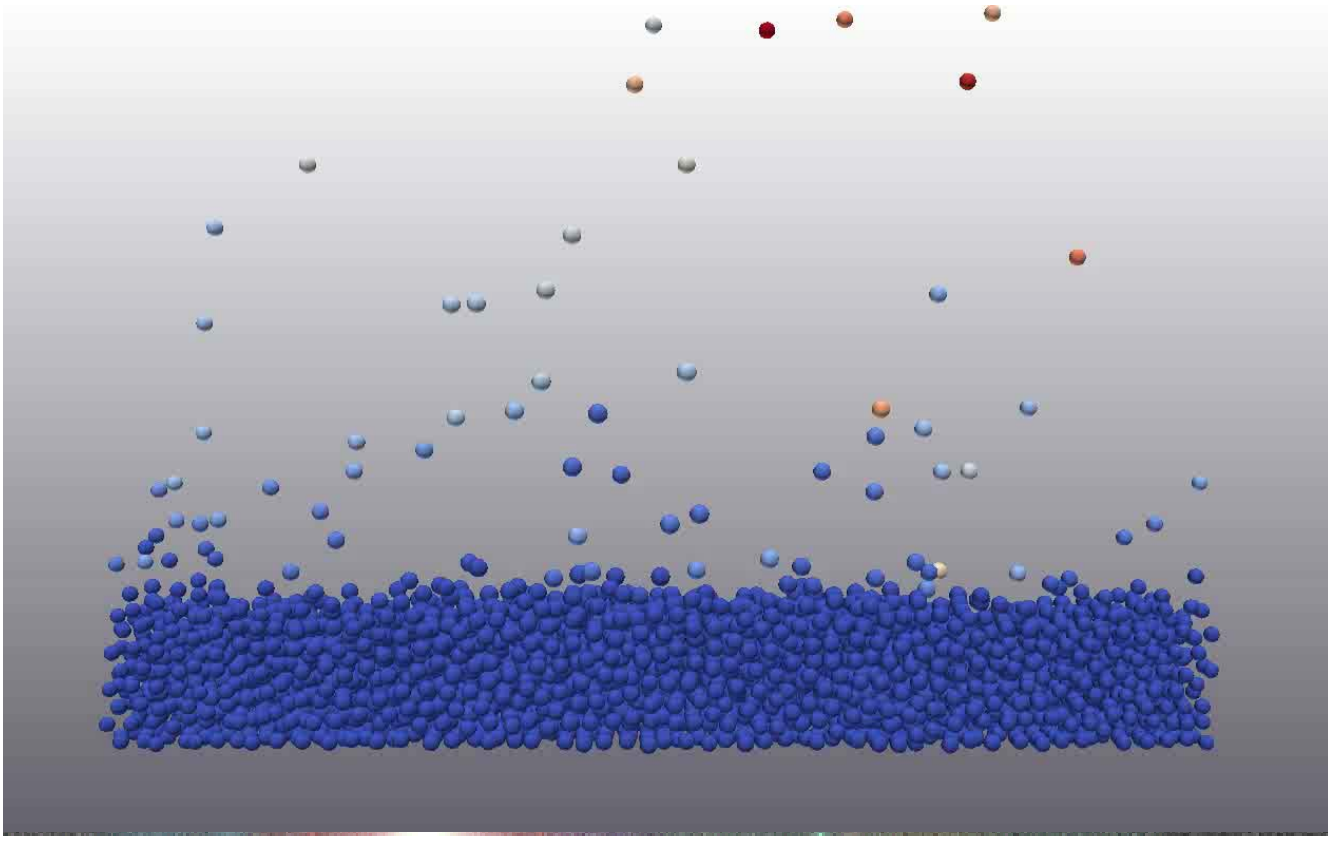
Oblate ellipsoids descend in a fluid under the action of gravity.

This has applications in biology (blood), industry (paint) and geology (clay).

Thesis of Frank Fonseca



$\theta = 0.15$ in 3d



Oral exams

Jan.22-Feb.02
2017

15 relevant questions

- Congruential and lagged-Fibbonacci RN
- Definition of percolation
- Fractal dimension and sand-box method
- Hoshen-Kopelman algorithm
- Finite size scaling
- Integration with Monte Carlo
- Detailed balance and MR^2T^2
- Ising model

15 relevant questions

- Simulate random walk
- Euler method
- 2nd order Runge-Kutta
- 2nd order predictor-corrector
- Jacobi and Gauss-Seidel relaxation
- Gradient methods
- Strategy of finite elements, finite volumes and spectral methods

Next semester

402-0810 Computational Quantum Physics

Giuseppe Carleo and Philippe de Forcrand

Tuesday afternoon: V Di 14-16, U Di 16-18

402-0812 Computational Statistical Physics

Mirko Lukovic and Miller Mendoza

Friday morning: V Fr 11-13, U Fr 9-11

327-5102 Molecular Materials Modelling

Daniele Passerone

Friday afternoon: V Fr 14-16, U Fr 16-18

Computational Quantum Physics

Giuseppe Carleo and Philippe de Forcrand

Tuesday afternoon: V Di 14-16, U Di 16-18

One particle quantum mechanics:

- scattering problem, time evolution
- shooting technique
- Numerov algorithm

Computational Quantum Physics

Many particle systems:

- Fock space, etc (≈ 2 weeks theory)
- Hartree-Fock approximation
- density functional theory and
- electron structure (He & H_2)
- strongly correlated electrons
- Hubbard and T-J models

Computational Quantum Physics

Lanczos method

Path integral Monte Carlo

Bosonic world lines

QCD, lattice gauge theory

Fermions, QFT

Molecular Materials Modelling

Daniele Passerone

Friday afternoon; V Fr 14-16, U Fr 16-18

Empirical potentials and transition rates

Bio-force fields, charges, peptides

Embedded atom models, Wilff‘s theorem

Pair-correlation function with MD

for neutron scattering

Molecular Materials Modelling

Melting temperature from phase coexistence

MO-theory, basic SCF, chemical reactions

Density functional theory, pseudopotentials

DFT on realistic systems, hybrids

Linear scaling, GPW

Electronic spectroscopies, STM

Bandstructure, graphene, free energies

Computational Statistical Physics

Mirko Lukovic and Miller Mendoza

Friday morning: V Fr 11-13, U Fr 9-11

Advanced Monte Carlo techniques:

- continuous variables (XY, Heisenberg)
- multi-spin coding, bit-manipulation
- vectorization, parallelization
- histogram methods, multi canonical

Computational Statistical Physics

Kawasaki dynamics, heat bath

microcanonical, Creutz algorithm, Q2R

critical slowing down, dynamical scaling

cluster algorithms (Swendsen-Wang, Wolff)

Monte Carlo Renormalization Group

Molecular Dynamics Simulations:

Verlet and leap frog methods

linked cell method, Verlet tables

Computational Statistical Physics

parallelization, realistic potentials

Ewald sums, reaction field method

Nose-Hoover thermostat, rescaling

constant pressure MD, melting

Discrete Elements, friction, inelasticity

rotation and quaternions

ab- initio calculations, Car Parinello