Partial differential equations

ETH

= 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

ETH

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:

$$\Delta\Phi = 0, \qquad \nabla_n \Phi(\Gamma) = \Psi_0$$

von Neuman boundary condition

Examples for PDEs

ETH

example: vectorial boundary value problem

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

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

Lamé equation of elasticity

(elliptic eq.)

Examples for PDEs

ETH

wave equation

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

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

diffusion equation

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

initial boundary problem

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

Examples for PDEs

$\vec{v}(\vec{x},t)$ **vector field in space and time**

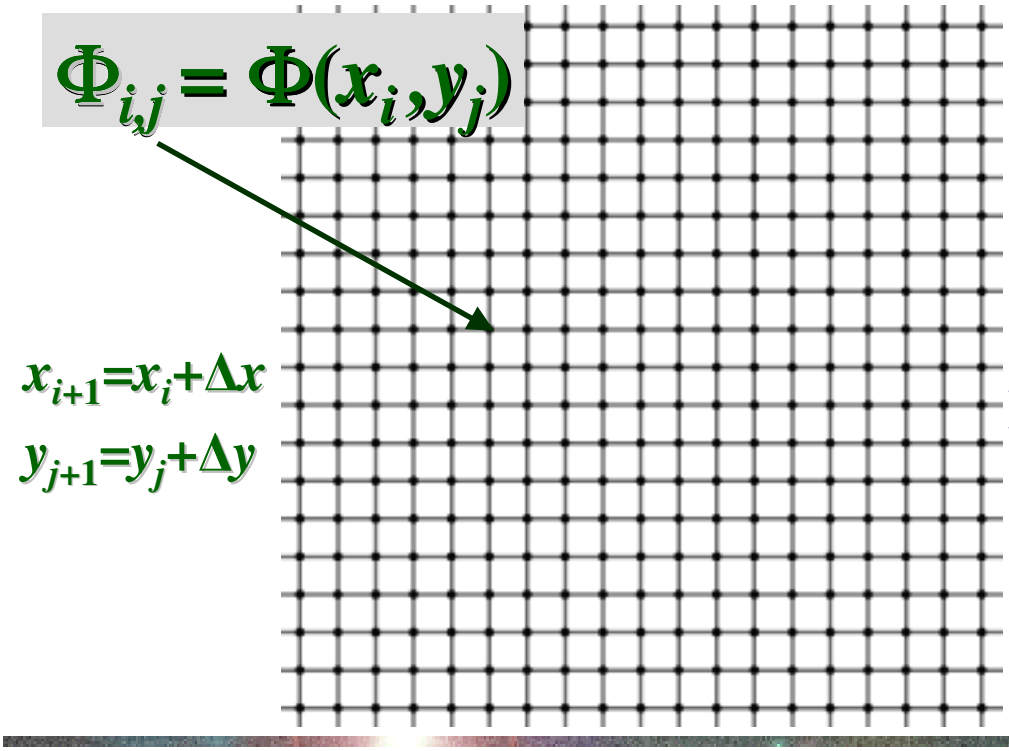
$$\frac{\partial \vec{v}}{\partial t} + (\vec{v}\vec{\nabla})\vec{v} = -\frac{1}{\rho}\vec{\nabla}p + \mu\,\Delta\vec{v}\ ,\quad \vec{\nabla}\vec{v} = 0$$

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

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

Navier – Stokes eq. for fluid motion

Discretization of space



Finite Difference Method

Discretization of derivatives

Δx small , $x_n = n \cdot \Delta x$

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

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

or better

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

Discretization of derivatives

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

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

Derivatives in higher dimension

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

2 d

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

3 d

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

Poisson equation

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

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

be $\Phi_n \equiv \Phi(x_n)$

discretization of the Poisson equation:

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

Dirichlet boundary conditions: $\Phi_0 = c_0$ **and** $\Phi_N = c_1$

$\Rightarrow$ **System of N-1 coupled linear equations**

Poisson equation in 1d

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

and Dirichlet boundary conditions

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

Poisson equation in 2d

ETH

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

($\Delta x = \Delta y$)

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

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

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

$\Rightarrow$ System of $N = L^2$ coupled linear equations:

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

„helical boundary conditions“: $1 \equiv N$

Poisson equation in 2d

ETH

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.

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

Exact solution

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

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

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

$$q_{ik} = -\frac{a_{ik}}{a_{kk}} \qquad \text{for } k = 1,\ldots, N$$

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

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

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

$$\Phi^*_N = \frac{b_N}{a_{NN}} \qquad \text{once matrix is triangular}$$

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

Poisson equation in 2d

Independently of the size of the system

each row or column

has only five non-zero matrix elements

$\Rightarrow$ 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

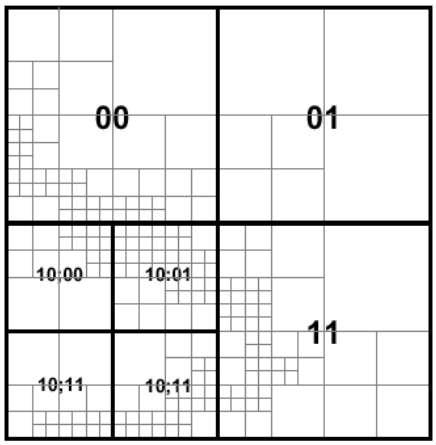
Table 2: Package authors, references, and availability

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

ETH

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

ETH

Computational effort for Gauss elimination ~ N^3.

For a lattice $100 \times 100 = 10^4$ one needs 2 days.

$\Rightarrow$ Abandon exact solution and use approximation.

But for that $\overleftrightarrow{A}$ 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:

$$\Delta\Phi(x,y) = b(x,y)$$

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

general:

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

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

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

Error of Jacobi relaxation

Exact solution is only reached for $t \to \infty$.

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:

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

($\vec{b}$: AA^{-1})

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

Error of Jacobi relaxation

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

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

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

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

Error of Jacobi relaxation

real error:

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

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

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

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

Gauss-Seidel relaxation

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

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

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

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

fixed point is the exact solution

Error in Gauss-Seidel

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

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

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

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

λ largest EV of $\vec{\vec{\Lambda}}$

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

Jacobi relaxation

Gauss-Seidel relaxation

Overrelaxation

Successive overrelaxation = SOR

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

Fixed point is the exact solution.

ω is the overrelaxation parameter.

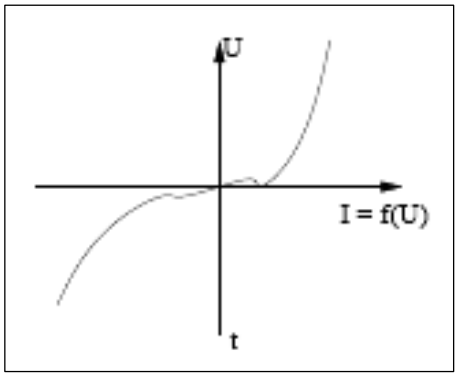
$$1 \le \omega < 2$$

ω = 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:



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

Solve with relaxation:

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

Gradient methods

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

The residuum (error)

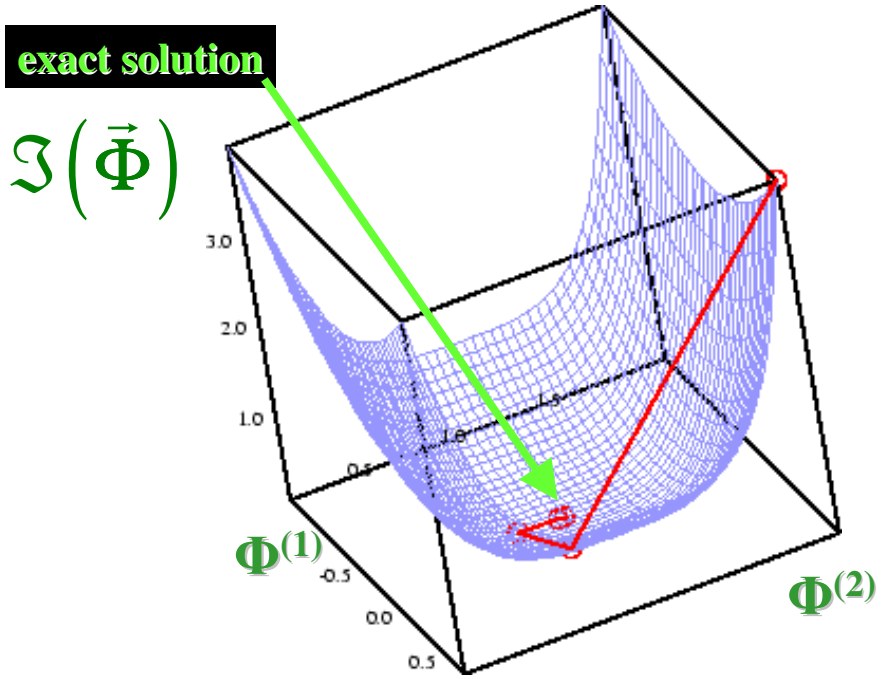
$$\vec{r}=\overleftrightarrow{A}\,\vec{\delta}=\overleftrightarrow{A}\left(\overleftrightarrow{A}^{-1}\vec{b}-\vec{\Phi}\right)=\vec{b}-\overleftrightarrow{A}\,\vec{\Phi}$$

is a measure for the error.

Minimize the functional:

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

Gradient methods



Gradient methods

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

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

Minimize along lines:

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

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

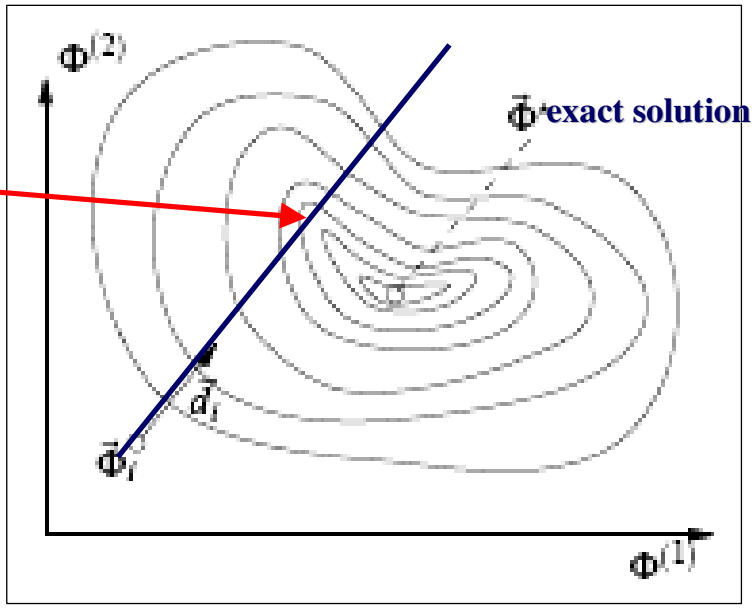
minimization condition with respect to α_i:

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

Gradient methods

minimum given by

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



Method of steepest descent

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

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

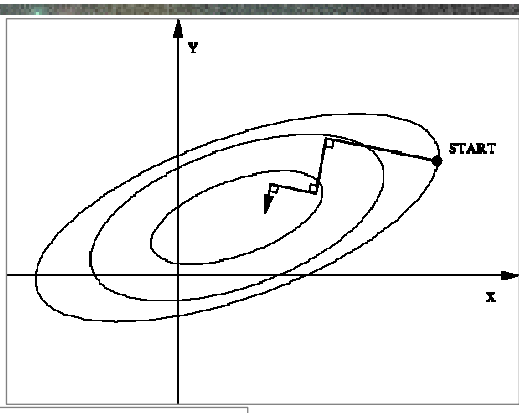
iterate: $\vec{u}_i = \overleftrightarrow{A}\vec{r}_i$, $\alpha_i = \dfrac{\vec{r}_i^{\,2}}{\vec{r}_i^{\,t}\vec{u}_i}$

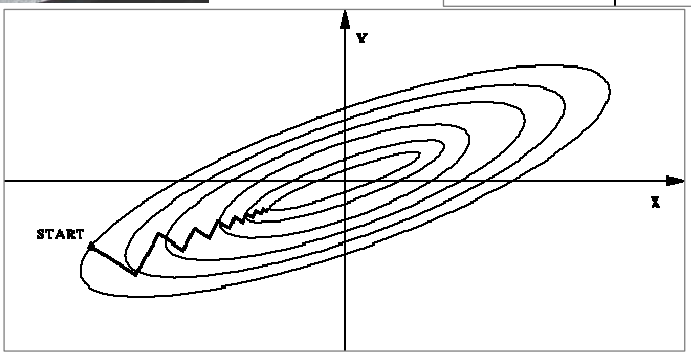
$$\vec{\Phi}_{i+1} = \vec{\Phi}_i + \alpha_i \vec{r}_i$$

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

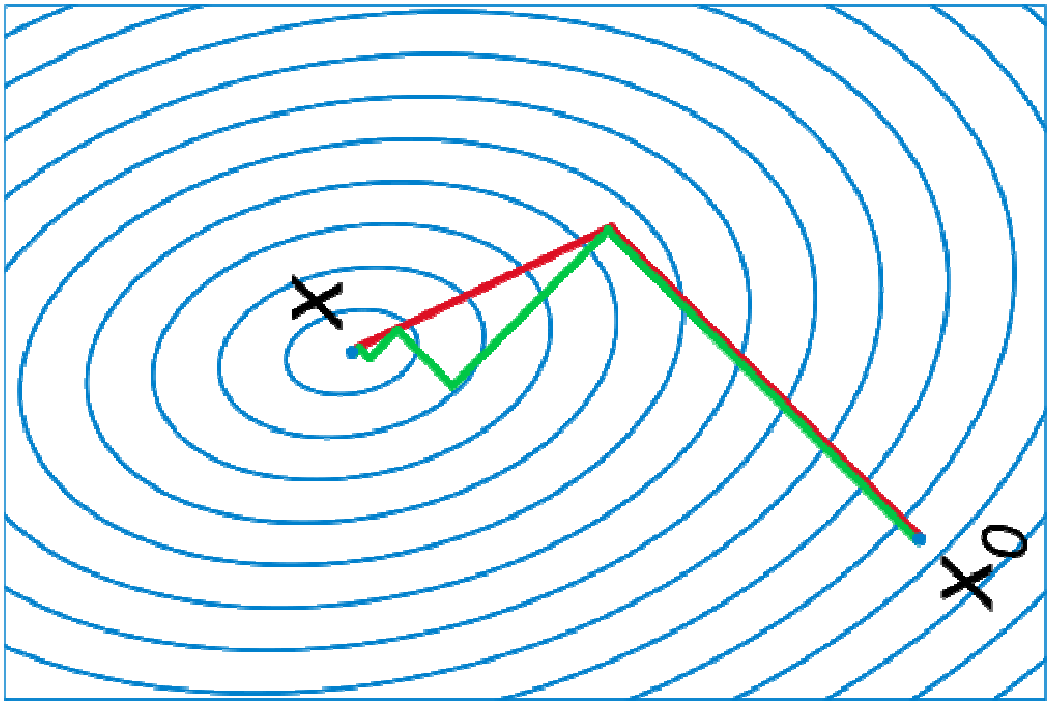
each step ~ N^2, but when matrix $\overleftrightarrow{A}$ sparse ~ N

Gradient methods





More efficient gradient method



Conjugate gradient

ETH

Hestenes and Stiefel (1957)

Choose $\vec{d}_i$ conjugate to each other:

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

as before:

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

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

Conjugate gradient

ETH

Construct conjugate basis using an orthogonalization procedure:

(Gram – Schmidt)

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

one can also show:

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

Conjugate gradient

ETH

1. initialize: $\vec{r}_1 = \vec{b} - \vec{\vec{A}}\vec{\Phi}_1 \quad , \quad \vec{d}_1 = \vec{r}_1$

2. iterate:

$$c = \left(\vec{d}_i^{\,t}\,\vec{\vec{A}}\,\vec{d}_i\right)^{-1} \quad , \quad \alpha_i = c\vec{d}_i\vec{r}_i \quad , \quad \vec{\Phi}_{i+1} = \vec{\Phi}_i + \alpha_i\vec{d}_i$$

$$\vec{r}_{i+1} = \vec{b} - \vec{\vec{A}}\vec{\Phi}_{i+1} \quad , \quad \vec{d}_{i+1} = \vec{r}_{i+1} - \left(c\vec{r}_{i+1}\vec{\vec{A}}\vec{d}_i\right)\vec{d}_i$$

3. stop when: $\vec{r}_i^{\,t}\vec{r}_i < \varepsilon$

Applet

Conjugate gradient

ETH

If matrix not symmetric than use biconjugate gradient method.

Consider two residuals:

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

This method does not always converge and can be unstable.

Biconjugate gradient

ETH

1. initialize:

$$\vec{r}_1 = \vec{b} - \overleftrightarrow{A}\,\vec{\Phi}_1 \quad , \quad \vec{d}_1 = \vec{r}_1$$
$$\tilde{\vec{r}}_1 = \vec{b} - \overleftrightarrow{A}^t \vec{\Phi}_1 \quad , \quad \tilde{\vec{d}}_1 = \tilde{\vec{r}}_1$$

2. iterate:

$$\vec{r}_{i+1} = \vec{r}_i - \alpha_i \overleftrightarrow{A}\,\vec{d}_i \quad , \quad \tilde{\vec{r}}_{i+1} = \tilde{\vec{r}}_i - \alpha_i \overleftrightarrow{A}^t \tilde{\vec{d}}_i \quad , \quad \alpha_i = c\,\vec{r}_i^t \vec{r}$$
$$\vec{d}_{i+1} = \vec{r}_i + \tilde{\alpha}_i \vec{d}_i \quad , \quad \tilde{\vec{d}}_{i+1} = \tilde{\vec{r}}_i + \tilde{\alpha}_i \tilde{\vec{d}}_i \quad , \quad \tilde{\alpha}_i = \tilde{c}\,\vec{r}_i^t \vec{r}_i$$
$$\text{with} \quad c = \left(\vec{d}_i^t \overleftrightarrow{A} \vec{d}_i\right)^{-1} \quad \text{and} \quad \tilde{c} = \left(\tilde{\vec{d}}_i^t \vec{d}_i\right)^{-1}$$

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

Preconditioning

ETH

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

ETH

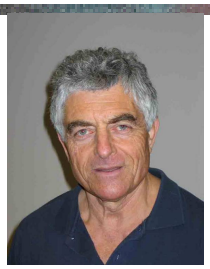
example: SOR preconditioner:

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

$\Rightarrow$ **Preconditioned Conjugate Gradient**

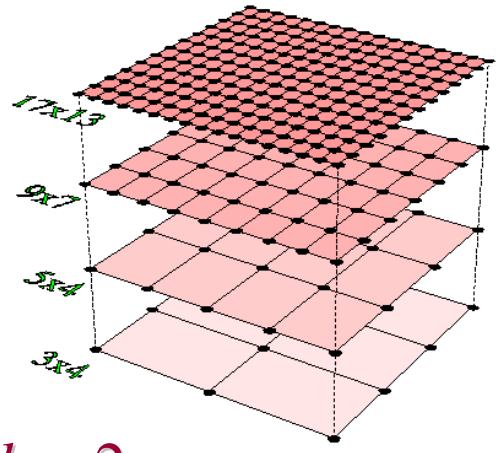
Multigrid procedure

ETH



Achi Brandt (1970)

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



$h = 2$

Multigrid procedure

ETH

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.

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

Multigrid procedure

ETH

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

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

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

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

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

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

Multigrid procedure

ETH

5. Get new approximate solution through:

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

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

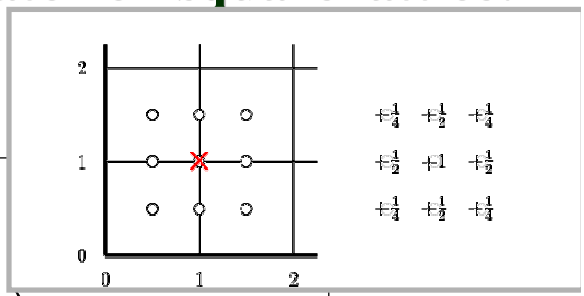
On each level one can also smoothen the error using several Gauss-Seidel relaxation steps.

Multigrid procedure

ETH

Example for extension operator on square lattice:

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

ETH

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.

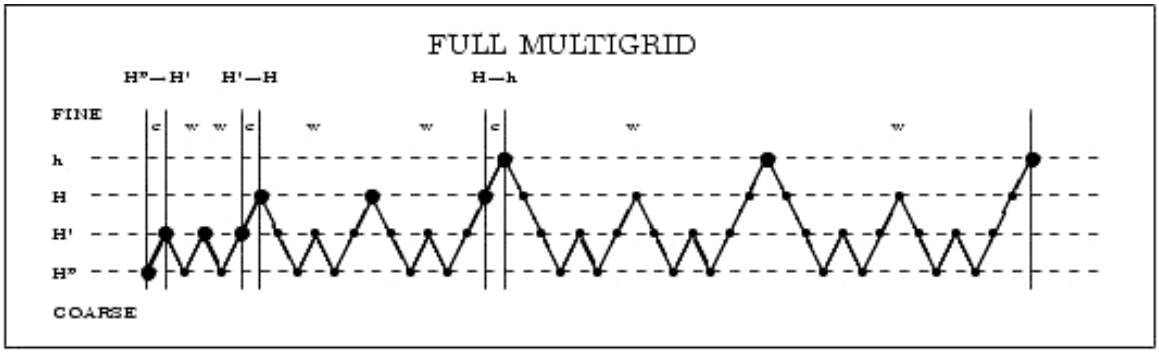
$$\sum_{x,y} \mathcal{P}\ \hat{v}(\hat{x},\hat{y}) \cdot u(x,y) = h^2 \sum_{\hat{x},\hat{y}} \hat{v}(\hat{x},\hat{y}) \cdot \mathcal{R}\ u(x,y)$$

Multigrid procedure

ETH

One can also vary the protocol

$\Rightarrow$ V-cycles, W-cycles, …



Solving PDEs

ETH

discretize ⇒ system of coupled linear equations

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

- **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

ETH

Finite Elements at ETH

- Gerald Kress: Strukturanalyse mit FEM
- Christoph Schwab: Numerik der Dgln.
- Pavel Hora: Grundlagen der nichtlinearen FEM
- Andrei Gusev: FEM in Solids and Structures
- Falk Wittel: Eine kurze Einführung in FEM
- Michael Faber: Methods of Finite Elements
- NachfolgerIn von Sanjay Govindjee:….

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

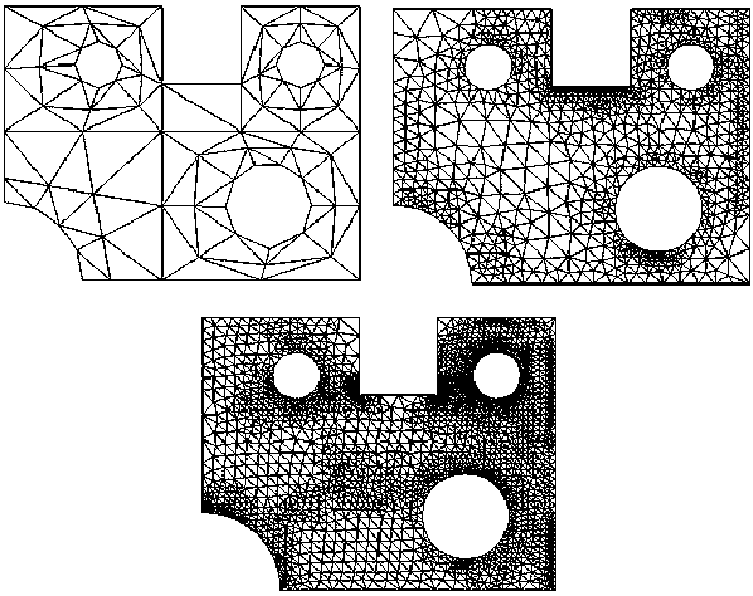
Clough (1960)

Advantage of finite elements over finite differences

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

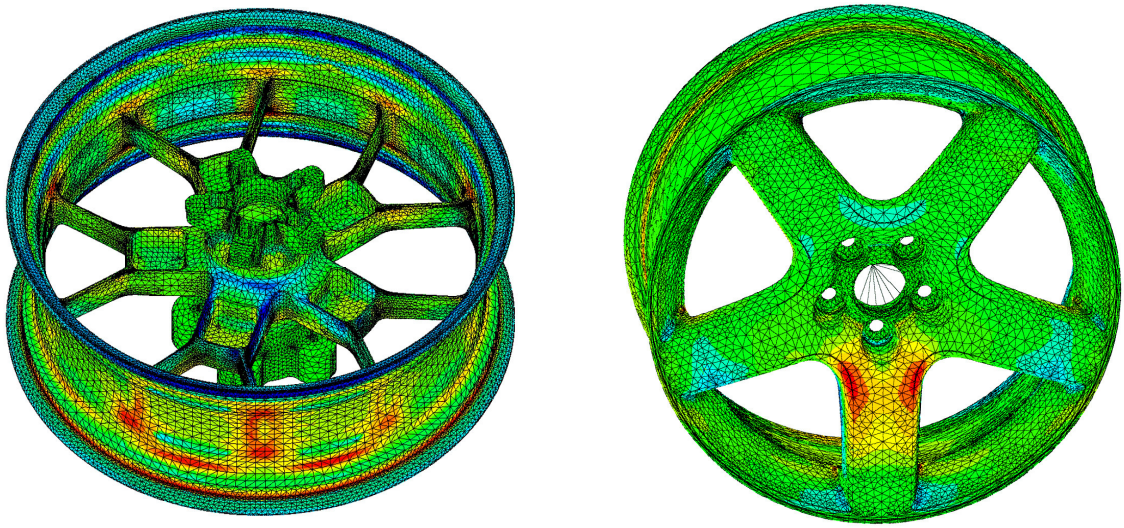
adaptive meshing, e.g. triangulation

Adaptive meshing in 2d



triangulations with different resolution

Adaptive meshing in 3d



triangulation of a wheel-rim

One dimensional example

ETH

Poisson equation:

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

Expand in terms of localized basis functions u_i:

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

One dimensional example

ETH

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.

$\Rightarrow$ system of linear equations $\quad \overleftrightarrow{A} \cdot \vec{a} = \vec{b}$

One dimensional example

$$\overleftrightarrow{A} \cdot \vec{a} = \vec{b}$$ with

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

and

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

One dimensional example

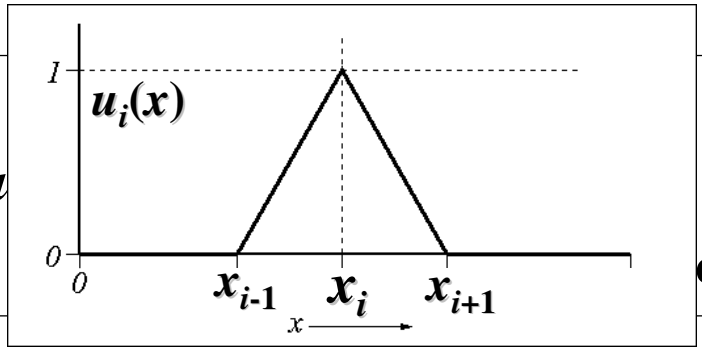
Example for basis functions $u_i(x)$ are hat functions centered around i:

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

$\Delta x \equiv x_i - x_{i\text{-}1}$

="element"

$\Rightarrow$ $A_{ij} = \int_0^L u$ $\quad i = j$ $\quad i = j \pm 1$ erwise



One dimensional example

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

If $\Phi(0)=\Phi_0 \;,\; \Phi(L)=\Phi_1$

then use following decomposition:

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

Non-linear PDEs

1d example : $\Phi(x)\frac{d^2\Phi}{dx^2}(x)=-4\pi\rho(x)$

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

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

$\sum_{i,j} A_{ijk}a_i a_j = b_k$ **with** $A_{ijk}=-\int_0^L u_i(x)\,u_j''(x)\,w_k(x)\,dx$

Picard iteration

ETH

Start with a guess Φ_0.

Solve linear equation for Φ_1:

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

Then iterate:

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

Function on Element

ETH

Higher dimensions

In 2d define function over

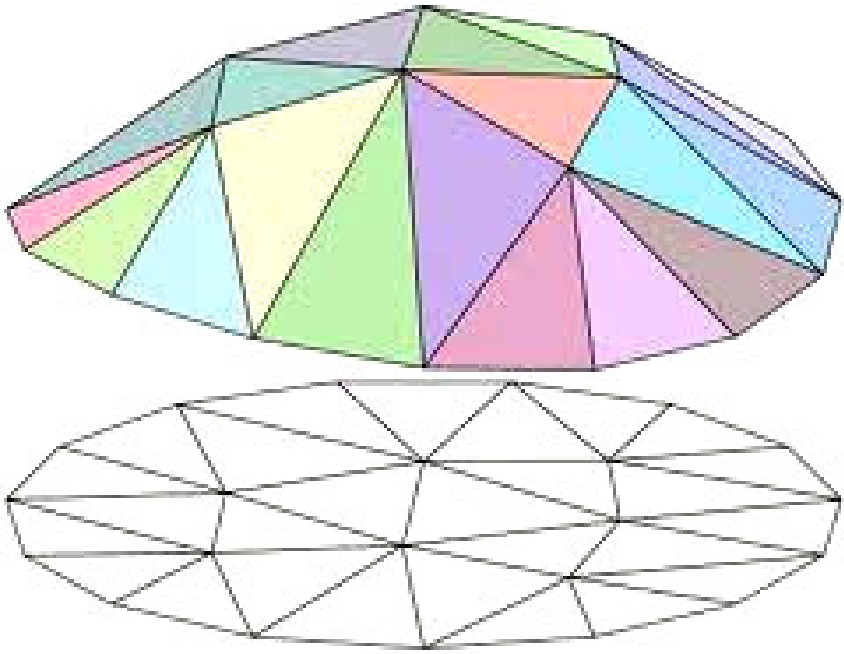
one element = triangle of the triangulation

e.g. linearly: $\Phi(\vec{r}) \approx c_1 + c_2 x + c_3 y$

or by a paraboloid:

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

Linear case



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:

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

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

Variational Approach

ETH

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

$a = 0$ Poisson equation

$\mathbf{b} = 0$ Helmholtz equation

First term of total energy

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

can be brought into the form:

$$E = \vec{\Phi}\ \vec{A}\ \vec{\Phi} + \vec{b}\ \vec{\Phi}$$

Minimizing then gives:

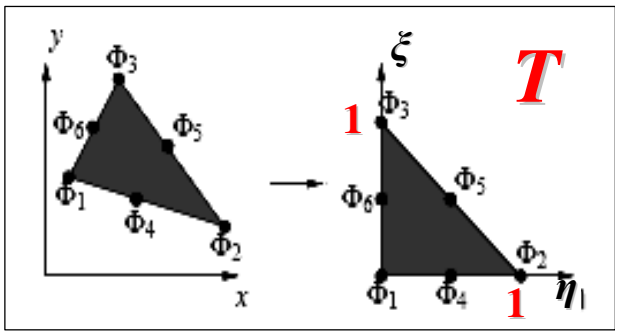
$$\frac{\partial E}{\partial \Phi} = 0 \quad \Rightarrow \quad \vec{A}\ \vec{\Phi} + \vec{b} = 0$$

Standard Form

ETH

Transform any element j into the standard form.

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



$$\Leftrightarrow$$

$$\eta = \left((y - y_1)(x_2 - x_1) - (x - x_1)(y_2 - y_1) \right) / D$$
$$\xi = \left((x - x_1)(y_3 - y_1) - (y - y_1)(x_3 - x_1) \right) / D$$
$$D = (y_3 - y_1)(x_2 - x_1) - (x_3 - x_1)(y_2 - y_1)$$

Coordinate transformation

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

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

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

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

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

Coordinate transformation

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

Jacobi matrix

$$\vec{\vec{J}} = \begin{pmatrix} \frac{\partial x}{\partial\xi} & \frac{\partial x}{\partial\eta} \\ \frac{\partial y}{\partial\xi} & \frac{\partial y}{\partial\eta} \end{pmatrix}$$

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

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

Coordinate transformation

ETH

Inserting gives for each element

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

where the coefficients are only calculated once for each element.

$$c_1 = \frac{(y_3 - y_1)^2}{D} + \frac{(x_3 - x_1)^2}{D}$$

$$c_2 = 2\frac{(y_3 - y_1)(y_2 - y_1)}{D} + 2\frac{(x_3 - x_1)(x_2 - x_1)}{D}$$

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

Basis functions

ETH

Decompose on standard element in basis functions N_i

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

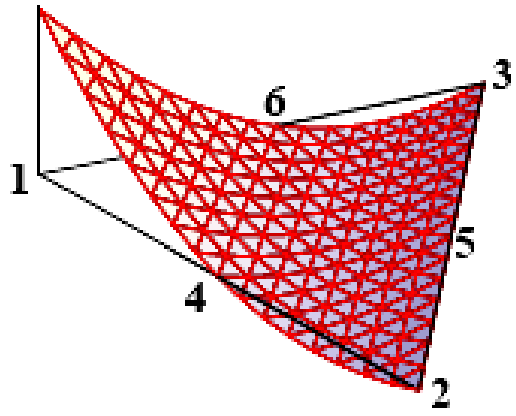
$$N_1 = (1-\xi-\eta)(1-2\xi-2\eta) \quad , \quad N_2 = \xi(2\xi-1)$$

$$N_3 = \eta(2\eta-1) \quad , \quad N_4 = 4\xi(1-\xi-\eta)$$

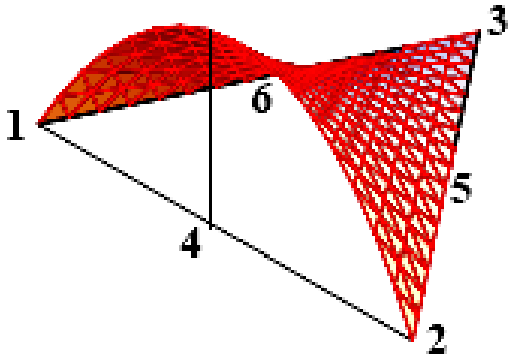
$$N_5 = 4\xi\eta \quad , \quad N_6 = 4\eta(1-\xi-\eta)$$

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

Shape of basis functions

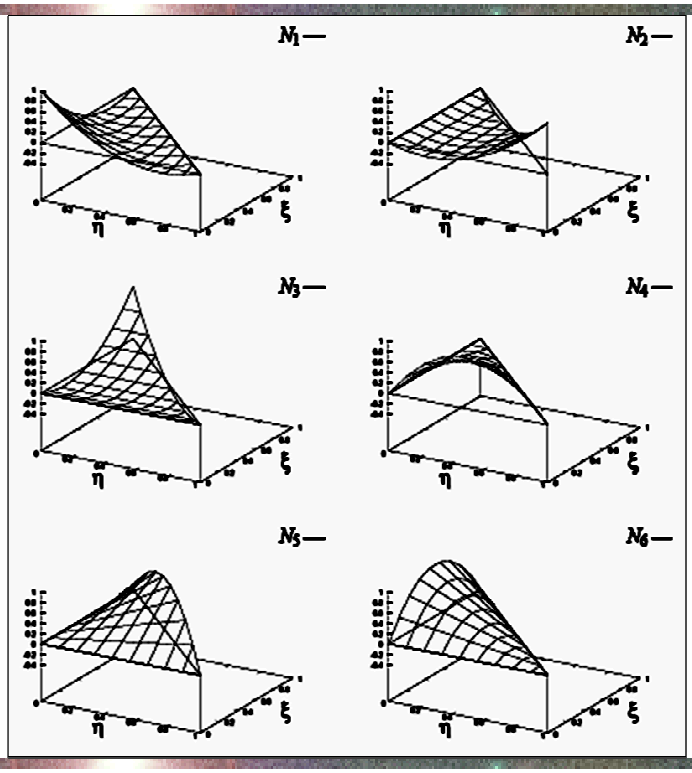


$$N_1^e = \zeta_1(2\zeta_1 - 1)$$

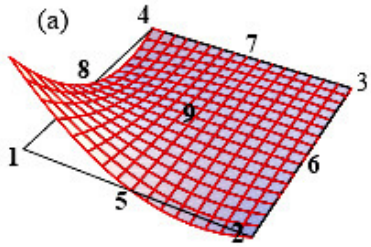


$$N_4^e = 4\zeta_1\zeta_2$$

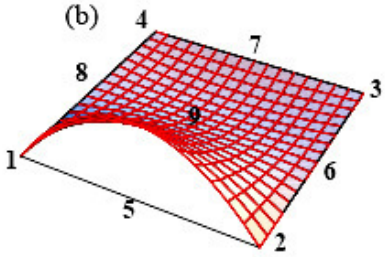
Shape of basis functions



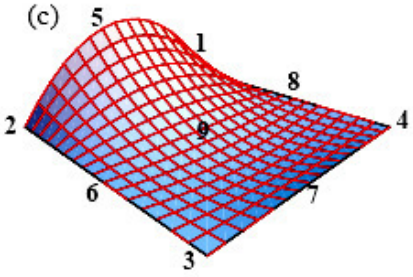
Shape functions on square lattice



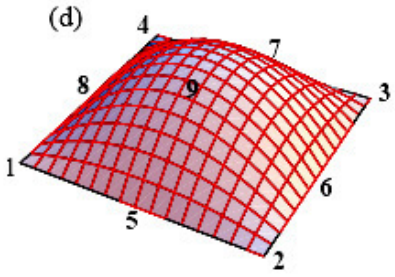
$N_1^e = \frac{1}{4}(\xi - 1)(\eta - 1)\xi\eta$



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



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



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

Energy Integrals

Calculate the energy integrals on standard element

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

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

and analogously

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

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

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

Rigidity Matrix

$$\vec{\varphi}^t \overleftrightarrow{S} \vec{\varphi} = \iint_{G_j} (\nabla \Phi)^2 \, dxdy$$

$$= \iint_T \left(c_1 \Phi_\xi^2 + 2c_2 \Phi_\xi \Phi_\eta + c_3 \Phi_\eta^2 \right) d\xi d\eta$$

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

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

Mass Matrix

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

$$\iint_{G_j} a \, \Phi^2 dxdy = \iint_T a \left(\vec{\varphi}_j \, \vec{N}(\xi,\eta) \right)^2 D_j \, d\xi d\eta$$

$$= \vec{\varphi}_j^t \underbrace{a \iint_T \vec{N} \, \vec{N}^t D_j \, d\xi d\eta}_{\overleftrightarrow{M}} \vec{\varphi}_j \equiv \vec{\varphi}_j^t \overleftrightarrow{M} \vec{\varphi}_j$$

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

$$\Rightarrow \quad E = \vec{\Phi}^t \, \overleftrightarrow{A} \, \vec{\Phi} \quad \text{with} \quad \vec{\Phi} = \left(\vec{\varphi}_j \right) \quad \text{and} \quad \overleftrightarrow{A} = \bigotimes_j \left(\overleftrightarrow{S}_j + \overleftrightarrow{M}_j \right)$$

Field term

$$\iint_{G_j} \mathrm{b}\Phi dxdy = \iint_{T} \mathrm{b}\ \vec{\varphi}_j\ \vec{N}(\xi,\eta)\ D_j d\xi d\eta$$

$$= \vec{\varphi}_j\ \underbrace{\mathrm{b}\iint_{T} \vec{N}(\xi,\eta)\ D_j d\xi d\eta}_{\vec{b}_j} = \vec{b}_j \vec{\varphi}_j$$

$$\Rightarrow$$

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

Variational approach

The solution of $\Delta\Phi = a\,\Phi + \mathrm{b}$ is minimum of

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

which can be brought into the form:

$$E = \vec{\Phi}\ \vec{\vec{A}}\ \vec{\Phi} + \vec{b}\ \vec{\Phi}$$

so that minimizing gives:

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

Numerical task of FEM

ETH

$\Rightarrow$ **Solve system** $\boxed{\overleftrightarrow{A}\,\vec{\Phi}+\vec{b}=0}$

of $3 \cdot N$ linear equations where N is the number of elements.
Matrix $\overleftrightarrow{A}$ 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

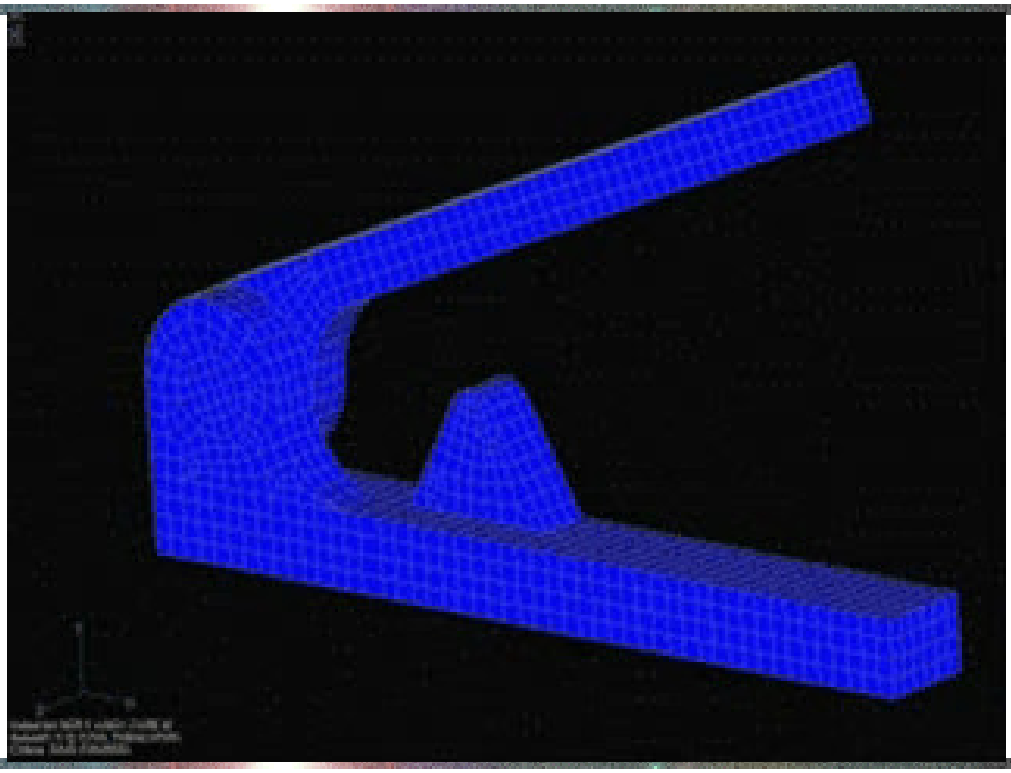
ETH

The connection between the elements gives off-diagonal terms in the matrix $\overleftrightarrow{A}$.
Finally one must also include the boundary terms.

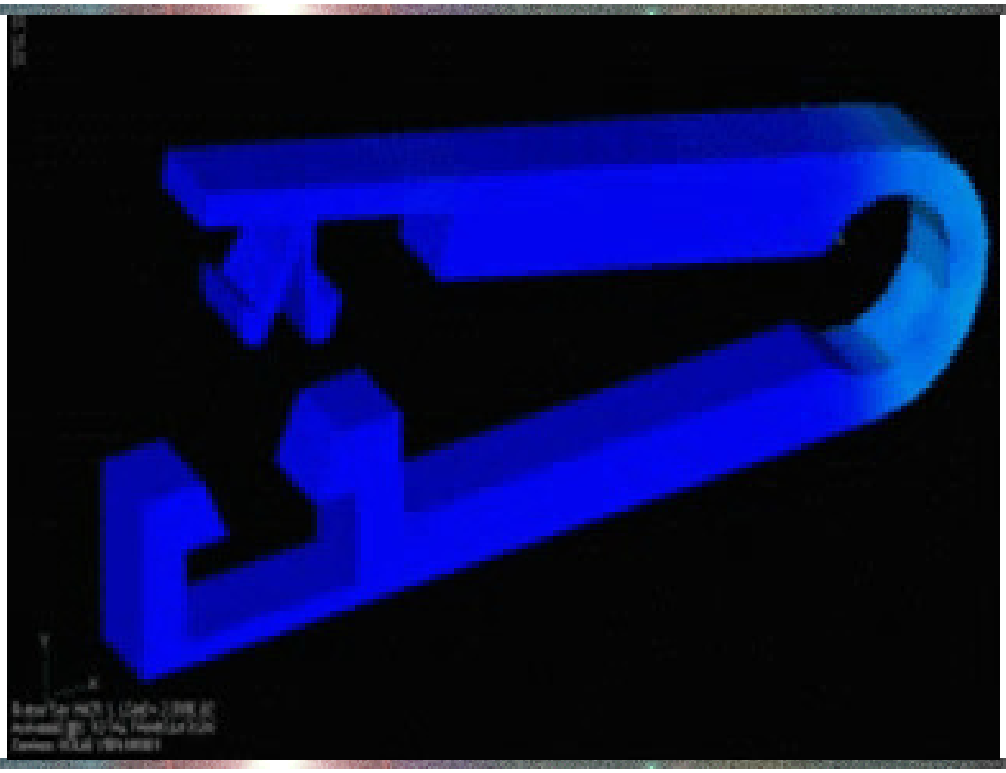
Applet

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

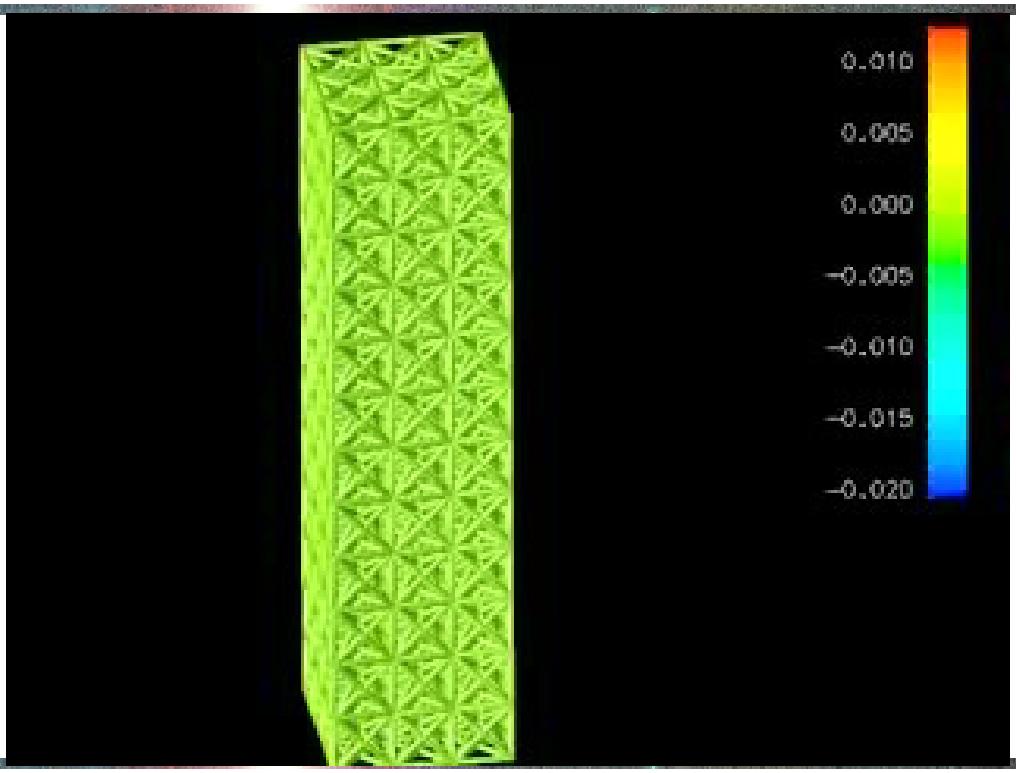
Stresses in a hinge



Stresses in a clip



Network of trusses



Time dependent PDE‘s

Simple example is heat equation:

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

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 $\dfrac{\kappa\,\Delta t}{C\rho\,\Delta x^2}\geq\dfrac{1}{4}$

Crank - Nicolson method

(1947)

implicit algorithm

John Crank

Phyllis Nicolson

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

define

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

Crank - Nicolson method

ETH

Define operator O

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

Then Crank – Nicolson becomes:

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

Crank - Nicolson method

ETH

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

Then Crank – Nicolson becomes:

$$\big(2\cdot\mathbf{1}-\mathbf{O}\big)\,\vec{T}(t+\Delta t)=\big(2\cdot\mathbf{1}+\mathbf{O}\big)\,\vec{T}(t)+\frac{\Delta t}{C\rho}\Big(\vec{W}(t)+\vec{W}(t+\Delta t)\Big)$$

where 1 is the unity operator.

Crank - Nicolson method

Calculate the inverted operator B before:

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

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

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 length smaller than λ.

Navier – Stokes equation

Computational Fluid Dynamics = CFD

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

viscosity

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

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

$$P_0(\vec{x})$$

$$\vec{v}(\Gamma,t) = \vec{\mathbf{v}}_0($$

$$(t)$$

equation of mo … ssible fluid

Navier – Stokes equation

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

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

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

Insert incompressibility condition:

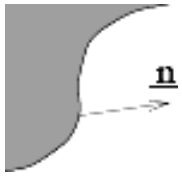
$$\vec{\nabla}\vec{v}_{k+1} = \vec{\nabla}\vec{v}_k = 0$$

Navier – Stokes equation

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

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

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

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

ETH

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

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

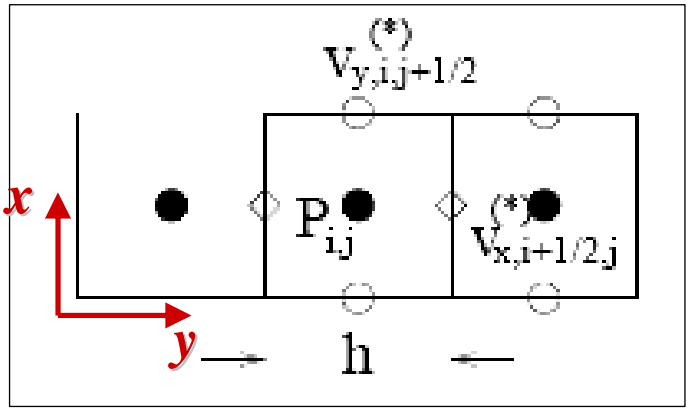
Projecting on the normal $\vec{n}$ to the boundary

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

Spatial discretization

ETH

MAC = Marker and Cell is a staggered lattice:

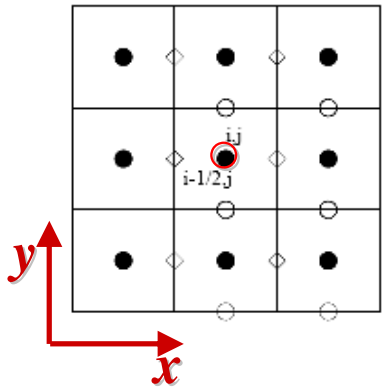


h is the lattice spacing

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

Spatial discretization

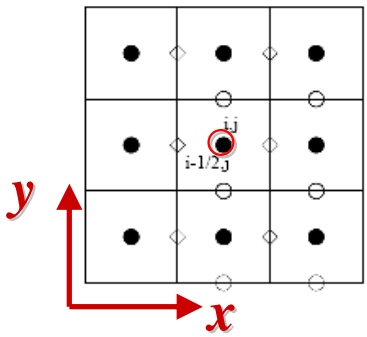
$$\left(\vec{\nabla} p\right)_{x,i+1/2,j} = \frac{1}{\mathbf{h}}\left(p_{i+1,j} - p_{i,j}\right)$$



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

Spatial discretization

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

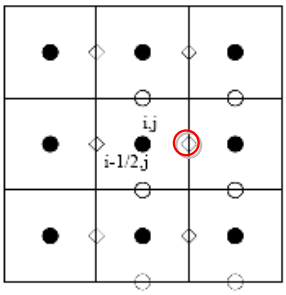


$$\nabla^2 p_{k+1} = \frac{\vec{\nabla}\vec{v}^*}{\Delta t}$$

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

Spatial discretization

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

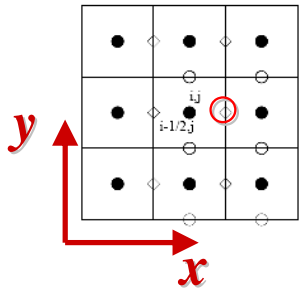


The equations for the velocity components are solved on the edges.

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

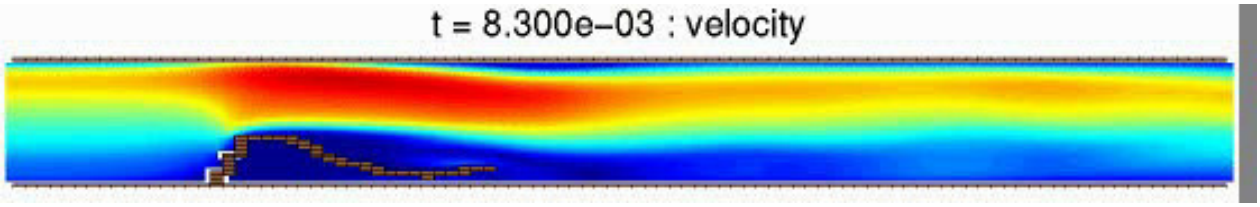
Spatial discretization

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



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

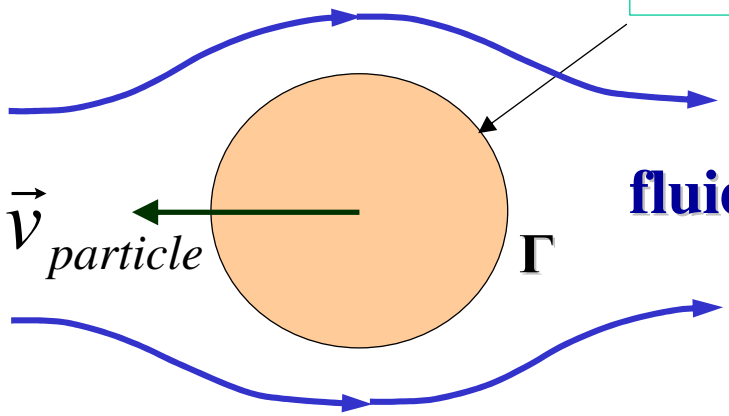
Flow around a vocal chord



One particle in fluid

e.g. pull sphere through fluid

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



moving boundary condition

create shear in fluid : exchange momentum

Drag force

drag force

(Bernoulli's principle)

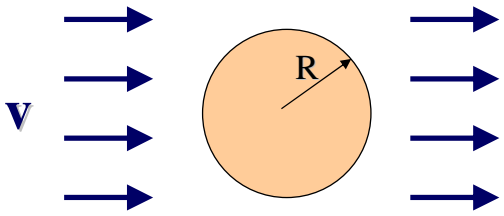
$$\vec{F}_D = \int_\Gamma \vec{\Theta} d\vec{A}$$

stress tensor

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

$\eta = \rho\,\mu$ 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

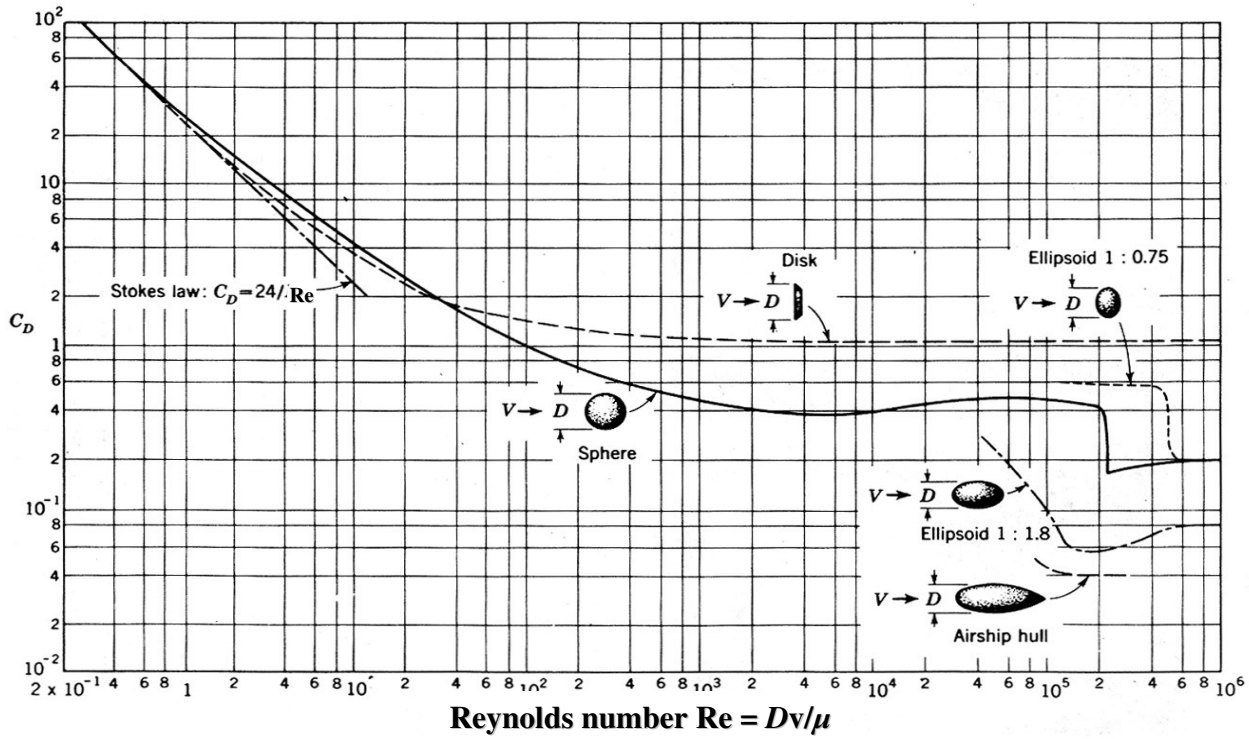
Re >> 1 Newton's law: $F_D = 0.22\pi\,\rho\,R^2v^2$

general drag law:

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

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: $$L = C_L \times \rho \times \frac{V^2}{2} \times A$$

C_L is „lift coefficient“

---

when particle rotates: Magnus effect

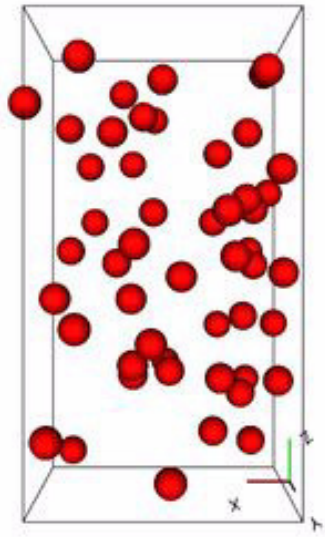
important for soccer

Many particles in fluids

ETH

- 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:

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



simulating particles moving in a sheared fluid

Stokes limit

ETH

hydrodynamic interaction between the particles

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

for Re = 0 mobility matrix exact

Stokesian Dynamics (Brady and Bossis)

invert a full matrix ⇒ only a few thousand particles

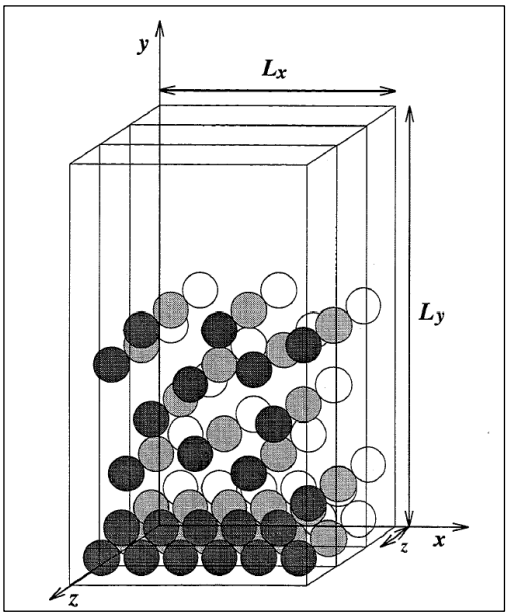
Numerical techniques

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

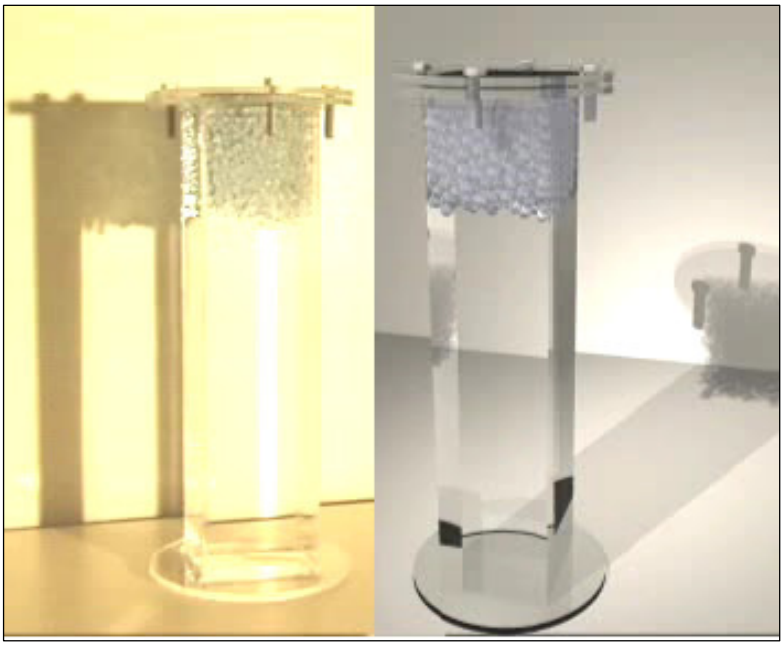
2 **Method of Fogelson and Peskin: Advect markers that were placed in the particle and then put springs between their new an their old position. These springs then pull the particle.**

Sedimentation

- Sedimentation ist the descent of particles in a fluid due to the action of gravity.
- The interaction between the particles and the fluid is given by the condition that the velocity of the fluid on the entire surface of each particle is equal to the velocity of this particle.
- Measure settling velocity, i.e. velocity of the upper front.
- If particles are of different species then one has several fronts.
- Open question: size dependence of the density fluctuations.



Sedimentation



Glass beads descending in silicon oil

comparing experiment and simulation

Discrete fluid solvers

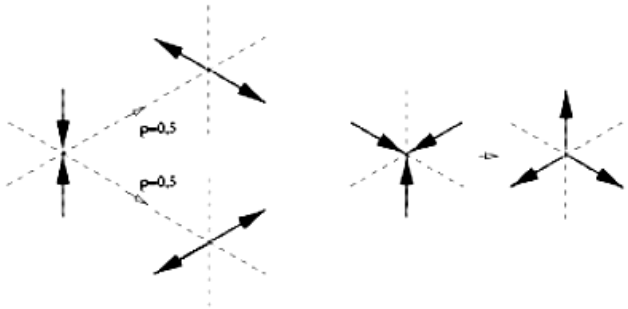
- Lattice Gas Automata (LGA)
- Lattice Boltzmann Method (LBM)
- Dissipative Particle Dynamics (DPD)
- Smooth Particle Hydrodynamics (SPH)
- Stochastic Rotation Dynamics (SRD)
- Direct Simulation Monte Carlo (DSMC)

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

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

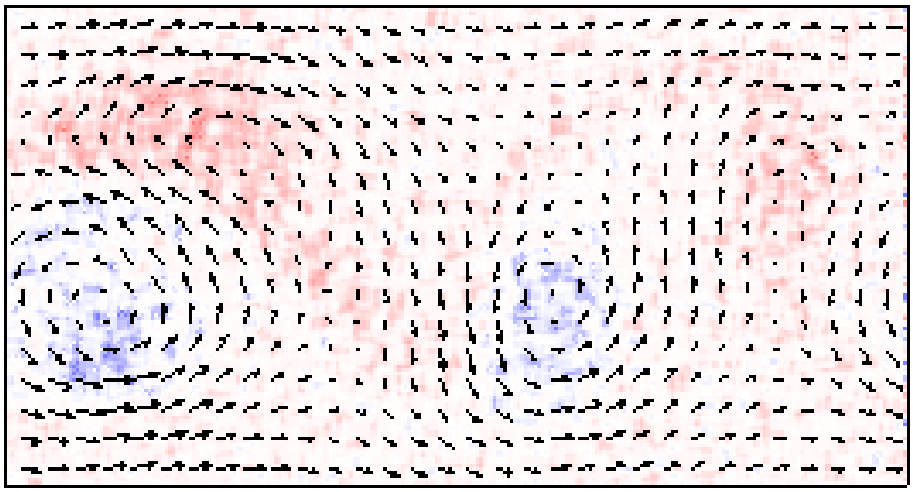


Momentum is conserved at each collision.
It can be proven (Chapman-Enskog) that its continuum limit is the Navier Stokes eq.

von Karman street

ETH

velocity field of a fluid behind an obstacle



Each vector is an average over time and space.

Lattice Boltzmann

ETH

From LGCA to Lattice Boltzmann Models (LBM)

- **(Boolean) molecules to (discrete) distributions**

$$n_i \longrightarrow f_i = \langle n_i \rangle$$

- **(Lattice) Boltzmann equations (LBE)**

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

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

Lattice Boltzmann

ETH

Lattice BGK (Bhatnagar-Gross-Krook)

Since Re depends only on ν , single time relaxation only

$$f_i\left(\vec{x}+\vec{c}_i,t+1\right)-f_i\left(\vec{x},t\right)=-\frac{1}{\tau}\left(f_i-f_i^{eq}\right)$$

Viscosity $\nu = c_s^2\left(\tau - \frac{1}{2}\right)$

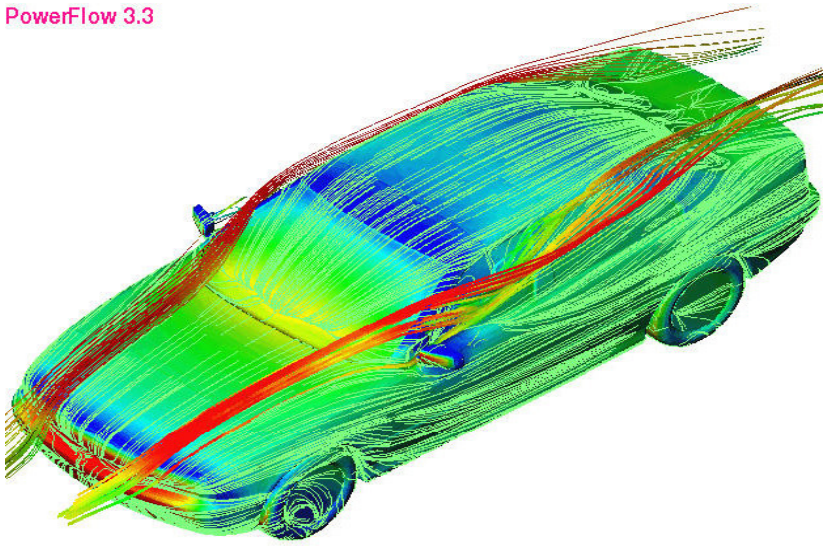
$c_s^2 = \frac{1}{3}$ **(lattice sound speed)**

Lattice Boltzmann

ETH

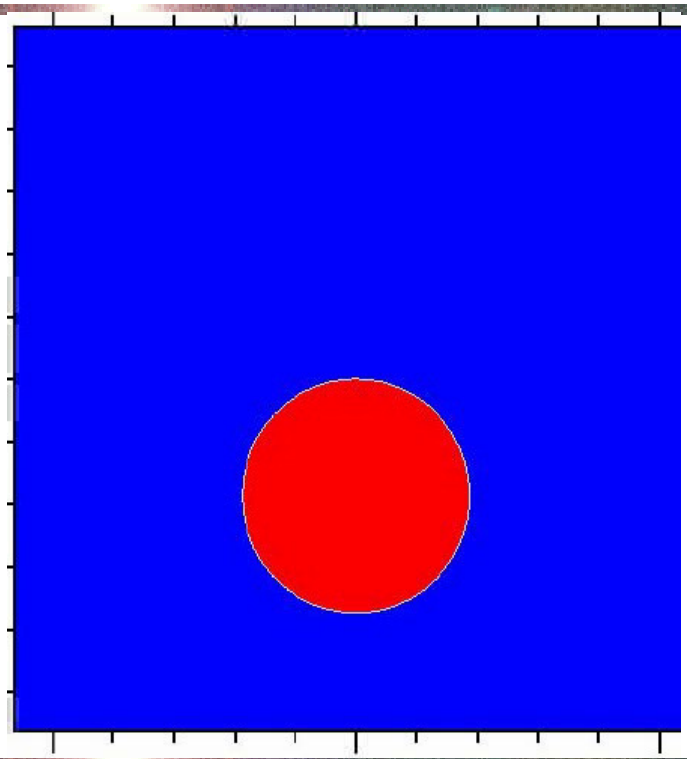
Car design

PowerFlow 3.3



Powerflow, EXA

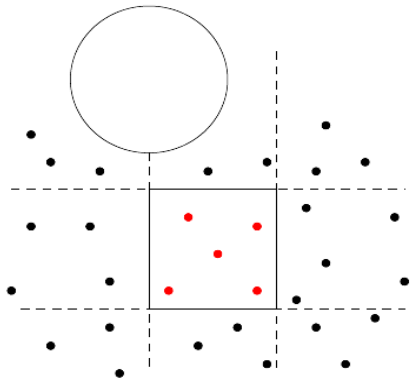
Raising of a bubble



Stochastic Rotation Dynamics

Stochastic Rotation Dynamics (SRD)

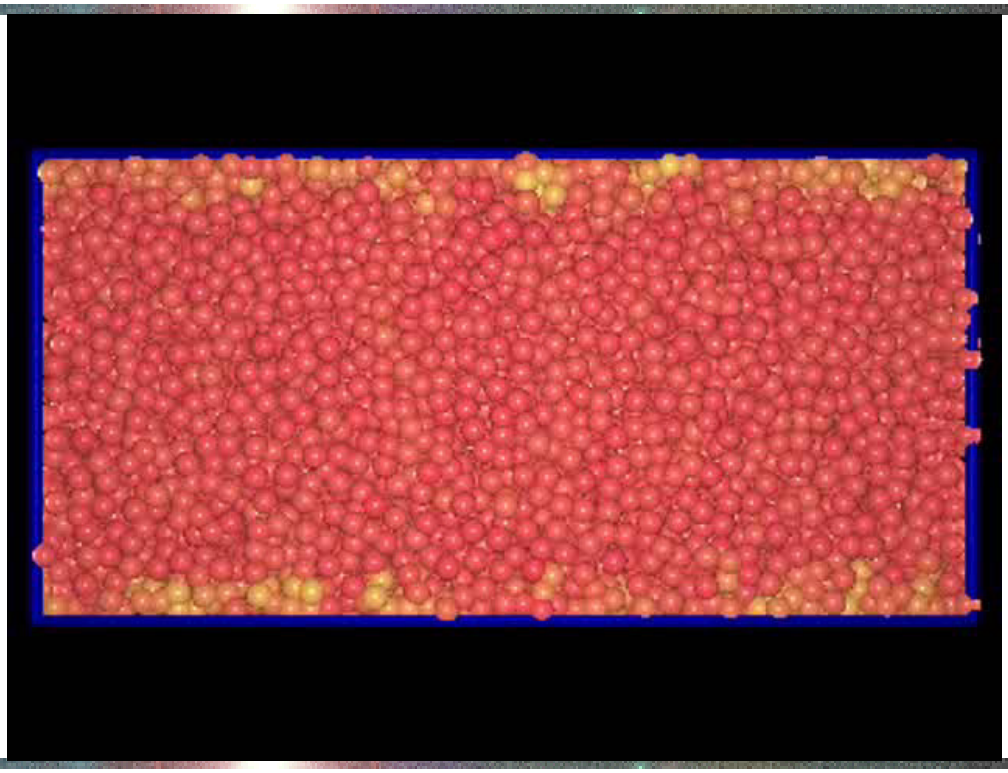
- introduction of representative fluid particles
- collective interaction by rotation of local particle velocities
- very simple dynamics, but recoveres hydrodynamics correctly
- Brownian motion is intrinsic



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

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

Shear flow



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)

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

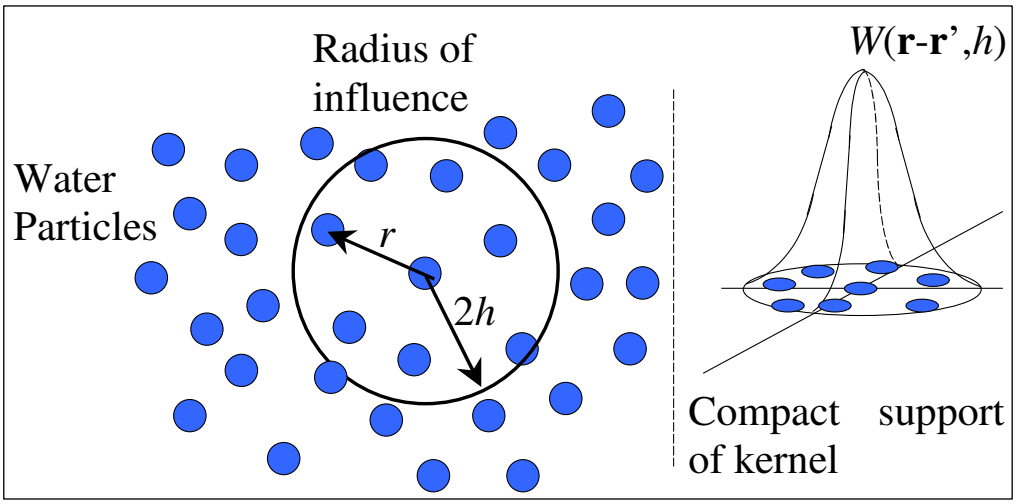
(W is the smoothing kernel)

- 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)



- **Example: quadratic kernel**

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

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

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

Equations of Motion

- **Navier-Stokes equations:**

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

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

- **Recast in particle form as:**

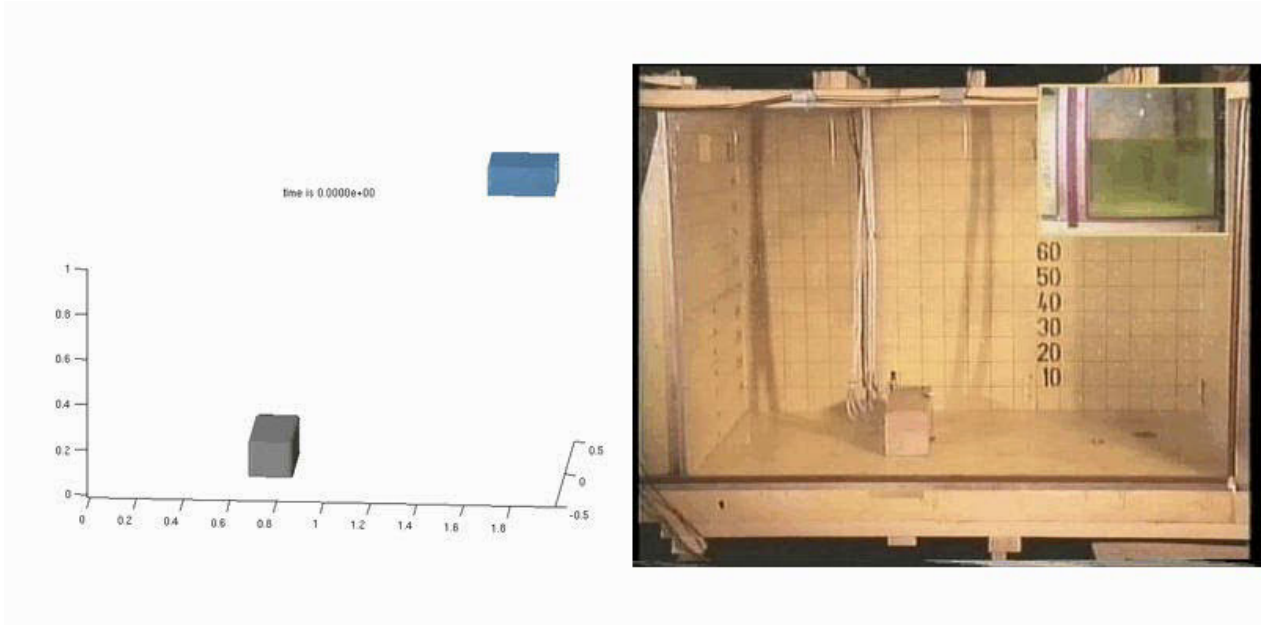
$$\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t} = \mathbf{v}_i + \varepsilon\sum_j m_j\left(\frac{\mathbf{v}_{ji}}{\bar{\rho}_{ij}}\right)W_{ij}$$

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

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

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

Simulation of free surface



15 relevant questions

ETH

- 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

ETH

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

Next semester

402-0810 Computational Quantum Physics

M. Troyer and P. de Forcrand

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

402-0812 Computational Statistical Physics

H. Herrmann

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

327-5102 Computational Polymer Physics

E. Del Gado

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

Computational Quantum Physics

Matthias Troyer 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

ETH

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

ETH

Lanczos method

Path integral Monte Carlo

Bosonic world lines

QCD, lattice gauge theory

Fermions, QFT

Computational Polymer Physics

ETH

Emanuela Del Gado

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

Monte Carlo for self-avoiding walk (SAW):
GSAW, trails, KSAW, worm-like chains
dynamics of single chains, diluted and semi-diluted polymers and melts
configuration flip, reptation, etc

Computational Polymer Physics

ETH

Non-equilibrium Molecular Dynamics:
thermostats, FENE (entropic potentials)
Rouse chain (springs)
linear response, viscosity, shear modulus
Fokker-Planck and Langevin equations:
Brownian dynamics and closure technique
orientation distribution (liquid crystals)

Computational Polymer Physics

Polymer networks, sol-gel transition

Membranes

Lattice gas, Lattice Boltzmann, SPH

Computational Statistical Physics

Hans Herrmann

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