Requirements for iterative linear equation solvers in problems of solid- and structural mechanics

Reijo Kouhia
Tampere University
http://www.tut.fi/rakmek/personnel/kouhia/

MTU60: Let's make it sparse, Prague, Czech Republic, Jan 31 - Feb 2, 2019

Happy birthday Mirek!

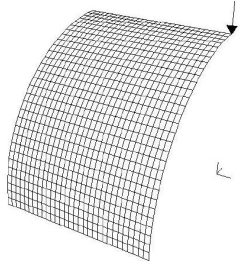
Veräjämäki May 27, 2000.

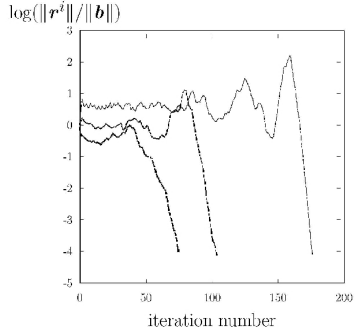
Some history

- I became interested in iterative linear solvers around 1995.
- After hitting my head to the wall with X preconditioners I found the paper:
  M. Benzi, C.D. Meyer, M. Tůma, A sparse approximate inverse preconditioner for the CG method, *SIAM J. Sci. Comp.*, **17**,1135-1149 (1996) and I felt fascinated about the idea.
- I sent a mail to Michele (autumn 1996?) and then everything started to evolve fast.
- Mirek, Michele and I wrote our first journal paper and it was published 1998 in *Commun. Numer. Methods Eng.*
- I met Mirek first time face to face in Prague May 1999 and Michele in Finland July 2004.
- Mirek visited Helsinki in May 2000.

It started with shells

Bending dominated loading, 30×30 uniform mesh, stabilized quadrilateral MITC4 shell elements 5489 unknowns. Convergence of the PCG depends highly on the "thinness" of the shell, here $t/R = 0.1, 0.01, 0.001$ and convergence with the IC(0)-preconditioner shown below.



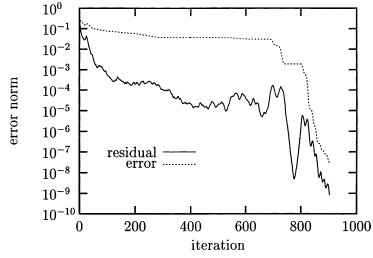


M. Benzi, RK, M. Tůma, An assessment of some preconditioning techniques in shell problems, *Communications in Numerical Methods in Engineering*, **14**, 1998, 897-906

In 1999 it continued with solids etc.

Veräjämäki May 27, 2000.

Block versions of the stabilized AINV
Discussion about artificial rhs-vectors



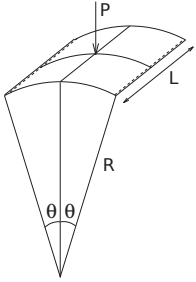
M. Benzi, RK, M. Tůma, Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics, *Computer Methods in Applied Mechanics and Engineering*, **190**, 2001, 6533-6554

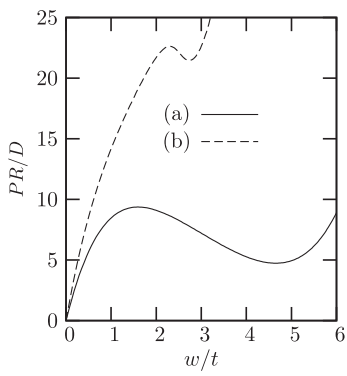
Why I am interested in shells? - Shells are prone to stability problems

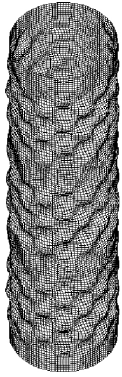
$R/L = 4$

(a) $R/t = 100$

(b) $R/t = 400$







Path-following methods

The non-linear mapping $\boldsymbol{f}$ defines the equilibrium path in the displacement $\boldsymbol{q}$ and load parameter λ space:

$$\boldsymbol{f}(\boldsymbol{q}, \lambda) \equiv \boldsymbol{r}(\boldsymbol{q}) - \lambda \boldsymbol{p}_r(\boldsymbol{q}) = \boldsymbol{0} \tag{1}$$

and constitutes the balance between internal- and external forces.

To overcome limit points an additional equation is usually augmented to (1), thus we have to solve

$$\begin{cases} \boldsymbol{f}(\boldsymbol{q}, \lambda) = \boldsymbol{r}(\boldsymbol{q}) - \lambda \boldsymbol{p}_{\mathrm{r}}(\boldsymbol{q}) = \boldsymbol{0} \\ c(\boldsymbol{q}, \lambda) = 0. \end{cases} \tag{2}$$

Newton's linearisation step results in

$$\boldsymbol{A} \delta \boldsymbol{x} = -\boldsymbol{g}, \tag{3}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{K} & \boldsymbol{p}_{\mathrm{r}} \\ \boldsymbol{c}^T & e \end{bmatrix}, \ \delta \boldsymbol{x} = \begin{Bmatrix} \delta \boldsymbol{q} \\ \delta \lambda \end{Bmatrix}, \ \boldsymbol{g} = \begin{Bmatrix} \boldsymbol{f} \\ \boldsymbol{c} \end{Bmatrix}, \ \boldsymbol{K} = \boldsymbol{f}' = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}, \quad \boldsymbol{c} = \frac{\partial c}{\partial \boldsymbol{q}}, \ e = \frac{\partial c}{\partial \lambda}.$$

Requirements for linear solvers in non-linear analysis

- Be able to handle indefinite matrices.
- Symmetry (if existing) of the Jacobian matrix of $\boldsymbol{f}$ should be exploitable.
- Information about definiteness (positive definite/indefinite) of the matrix should be extractable.

Stability of equilibrium state ?

Singularity test functions (STF)

- determinant
- smallest eigenvalue
- smallest pivot
- current stiffness parameter

Requirements for the STF

- mesh size independent
- "good predictor"
- indicates all kind of singular behaviour
- easy and cheap to evaluate

Check for the positivity

of a matrix $\boldsymbol{A}$ when applying preconditioned Krylov subspace methods to solve $\boldsymbol{Ax} = \boldsymbol{b}$ with accuracy $\|\boldsymbol{r}_i\| = \|\boldsymbol{b} - \boldsymbol{Ax}_i\| < \varepsilon_{\mathrm{r}}\|\boldsymbol{b} - \boldsymbol{Ax}_0\| + \varepsilon_{\mathrm{a}}$.

- λ_i is an eigenvalue of $\boldsymbol{A}$
- $\lambda_1 = \min \lambda_i$
- $|\lambda_1| < \delta$

Conjecture

If $\boldsymbol{A}$ is indefinite
then there exists with probability $1 - \eta(\delta, \varepsilon_{\mathrm{r}}, \varepsilon_{\mathrm{a}})$
at least one iterate k such that $\boldsymbol{d}_k^T \boldsymbol{A} \boldsymbol{d}_k < 0$.

To be usefull: $\eta \ll 1$ and $\boldsymbol{A}\boldsymbol{d}_k$ should be available

1. form $\boldsymbol{M}$ (or directly $\boldsymbol{M}^{-1}$),
2. initialize $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{Ax}_0$, apply preconditioner $\boldsymbol{d}_0 = \boldsymbol{M}^{-1}\boldsymbol{r}_0$, compute $\tau_0 = \boldsymbol{r}_0^T \boldsymbol{d}_0$,
3. iterate $i = 0, 1, 2, \ldots$ until convergence:
   - 3.1 compute: $\boldsymbol{s} = \boldsymbol{A}\boldsymbol{d}_i, \quad a_i = \tau_i / \boldsymbol{d}_i^T \boldsymbol{s}$,
   - 3.2 update: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + a_i \boldsymbol{d}_i$, $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - a_i \boldsymbol{s}$,
   - 3.3 apply preconditioner: $\boldsymbol{z} = \boldsymbol{M}^{-1}\boldsymbol{r}_{i+1}$,
   - 3.4 compute: $\tau_{i+1} = \boldsymbol{r}_{i+1}^T \boldsymbol{z}, \quad b_i = \tau_{i+1}/\tau_i$,
   - 3.5 update: $\boldsymbol{d}_{i+1} = \boldsymbol{z} + b_i \boldsymbol{d}_i$.

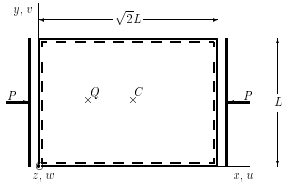
It works with PCG but not with preconditioned Bi-CGSTAB - Why?

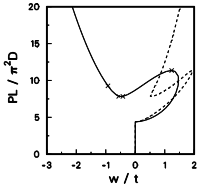
1. form $\boldsymbol{M}$ (or directly $\boldsymbol{M}^{-1}$),
2. compute $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, choose $\tilde{\boldsymbol{r}}$, compute $\rho_0 = \tilde{\boldsymbol{r}}^T \boldsymbol{r}_0$, set $\boldsymbol{d}_0 = \boldsymbol{r}_0$,
3. iterate $i = 0, 1, 2, \ldots$, until converged:
   - 3.1 apply preconditioner: $\boldsymbol{z} = \boldsymbol{M}^{-1}\boldsymbol{d}_i$,
   - 3.2 compute: $\boldsymbol{v}_i = \boldsymbol{A}\boldsymbol{z}, \quad a_i = \rho_i / \tilde{\boldsymbol{r}}^T \boldsymbol{v}_i, \quad \boldsymbol{s} = \boldsymbol{r}_i - a_i \boldsymbol{v}_i$,
   - 3.3 apply preconditioner: $\tilde{\boldsymbol{s}} = \boldsymbol{M}^{-1}\boldsymbol{s}$,
   - 3.4 compute: $\boldsymbol{w} = \boldsymbol{A}\tilde{\boldsymbol{s}}, \quad \omega_i = \boldsymbol{w}^T \boldsymbol{s} / \boldsymbol{w}^T \boldsymbol{w}$,
   - 3.5 update: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + a_i \boldsymbol{z} + \omega_i \tilde{\boldsymbol{s}}, \quad \boldsymbol{r}_{i+1} = \boldsymbol{s} - \omega_i \boldsymbol{w}$,
   - 3.6 compute: $\|\boldsymbol{r}_{i+1}\|_2$,
   - 3.7 compute: $\rho_{i+1} = \tilde{\boldsymbol{r}}^T \boldsymbol{r}_{i+1}$ and $b_{i+1} = (\rho_{i+1}/\rho_i)(a_i/\omega_i)$,
   - 3.8 update: $\boldsymbol{d}_{i+1} = \boldsymbol{r}_{i+1} + b_{i+1}(\boldsymbol{d}_i - \omega_i \boldsymbol{v}_i)$.

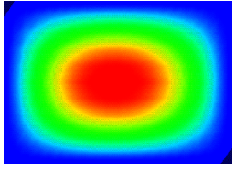
To the next topic - direct solution of singular points

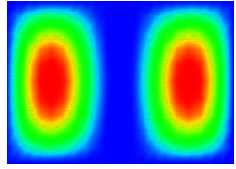
My old favourite - stability analysis

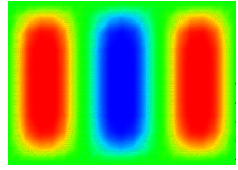
M.C. Escher, Metamorphosis II, 1940.











Stability eigenvalue problem

Definition for critical state: Find displacements $\boldsymbol{q}_{\mathrm{cr}}$, critical load λ_{cr} and the corresponding eigenmode $\boldsymbol{\phi}$ such, that

$$\begin{cases} \boldsymbol{f}'(\boldsymbol{q}_{\mathrm{cr}}, \lambda_{\mathrm{cr}})\boldsymbol{\phi} & = \mathbf{0} \\ \boldsymbol{f}(\boldsymbol{q}_{\mathrm{cr}}, \lambda_{\mathrm{cr}}) & = \mathbf{0} \end{cases} \tag{4}$$

where $\boldsymbol{f}' = \partial \boldsymbol{f} / \partial \boldsymbol{q}$.

System (4) is a non-linear eigenvalue problem, which is HARD TO SOLVE!

Solution algorithm for non-linear eigenproblem

Extended system:

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{q}, \boldsymbol{\phi}, \boldsymbol{\lambda}) = \begin{cases} \hat{\boldsymbol{f}}(\boldsymbol{q}, \boldsymbol{\lambda}) \equiv \boldsymbol{f}(\boldsymbol{q}, \lambda) + \boldsymbol{f}_0(\boldsymbol{q}, \boldsymbol{\lambda}) = \boldsymbol{0} \\ \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{\phi}, \boldsymbol{\lambda}) \equiv \boldsymbol{f}'(\boldsymbol{q}, \boldsymbol{\lambda})\boldsymbol{\phi} + \boldsymbol{h}_0(\boldsymbol{\phi}, \boldsymbol{\lambda}) = \boldsymbol{0} \\ \boldsymbol{c}(\boldsymbol{q}, \boldsymbol{\phi}, \boldsymbol{\lambda}) = \boldsymbol{0}. \end{cases}$$

Newton's linearisation step results in

$$\boldsymbol{A}\delta\boldsymbol{x} = -\boldsymbol{g},$$

where

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{K}_f & \boldsymbol{0} & \boldsymbol{P} \\ \boldsymbol{Z} & \boldsymbol{K}_h & \boldsymbol{N} \\ \boldsymbol{C}_q & \boldsymbol{C}_\phi & \boldsymbol{C}_\lambda \end{bmatrix}, \quad \delta\boldsymbol{x} = \begin{Bmatrix} \delta\boldsymbol{q} \\ \delta\boldsymbol{\phi} \\ \delta\boldsymbol{\lambda} \end{Bmatrix}, \quad \boldsymbol{g} = \begin{Bmatrix} \hat{\boldsymbol{f}} \\ \boldsymbol{h} \\ \boldsymbol{c} \end{Bmatrix}.$$

and

$$\begin{aligned} \boldsymbol{K}_f &= \hat{\boldsymbol{f}}'(\boldsymbol{q}, \boldsymbol{\lambda}) = \boldsymbol{K} + \boldsymbol{f}_0', & \boldsymbol{K}_h &= \partial\boldsymbol{h}/\partial\boldsymbol{\phi} = \boldsymbol{K} + \partial_0/\partial\boldsymbol{\phi}, \\ \boldsymbol{Z} &= (\boldsymbol{f}'\boldsymbol{\phi})', & \boldsymbol{P} &= \partial\hat{\boldsymbol{f}}/\partial\boldsymbol{\lambda}, \\ \boldsymbol{N} &= \partial\boldsymbol{h}/\partial\boldsymbol{\lambda}, & \boldsymbol{C}_\lambda &= \partial\boldsymbol{c}/\partial\boldsymbol{\lambda}, \\ \boldsymbol{C}_q &= \partial\boldsymbol{c}/\partial\boldsymbol{q}, & \boldsymbol{C}_\phi &= \partial\boldsymbol{c}/\partial\boldsymbol{\phi}. \end{aligned}$$

Block elimination

The solution vector is partitioned as

$$\delta \boldsymbol{q} = \boldsymbol{q}_f + \boldsymbol{Q}_p \delta \boldsymbol{\lambda},$$
$$\delta \boldsymbol{\phi} = \boldsymbol{\phi}_h + \boldsymbol{\Phi}_n \delta \boldsymbol{\lambda}.$$

Vectors $\boldsymbol{q}_f, \boldsymbol{\phi}_h$ and the $n \times p$ matrices $\boldsymbol{Q}_p, \boldsymbol{\Phi}_n$ solved from

$$\boldsymbol{K}_f \boldsymbol{q}_f = -\hat{\boldsymbol{f}}, \qquad \boldsymbol{K}_f \boldsymbol{Q}_p = -\boldsymbol{P},$$
$$\boldsymbol{K}_h \boldsymbol{\phi}_h = -\boldsymbol{h} - \boldsymbol{Z}\boldsymbol{q}_{\mathrm{f}}, \qquad \boldsymbol{K}_h \boldsymbol{\Phi}_n = -\boldsymbol{N} - \boldsymbol{Z}\boldsymbol{Q}_p,$$

and for the control parameters

$$\delta \boldsymbol{\lambda} = -(\boldsymbol{C}_\lambda + \boldsymbol{C}_q \boldsymbol{Q}_p + \boldsymbol{C}_\phi \boldsymbol{\Phi}_n)^{-1} (\boldsymbol{c} + \boldsymbol{C}_q \boldsymbol{q}_f + \boldsymbol{C}_\phi \boldsymbol{\phi}_h).$$

Suitable strategy if direct linear solver is used.

Iterative solution of the Newton step

Use precontioned Bi-CGSTAB method (or some other applicable accelerator) for the full non-symmetric equation systems

$$\begin{bmatrix} \boldsymbol{K}_f & \boldsymbol{0} & \boldsymbol{P} \\ \boldsymbol{Z} & \boldsymbol{K}_h & \boldsymbol{N} \\ \boldsymbol{C}_q & \boldsymbol{C}_\phi & \boldsymbol{C}_\lambda \end{bmatrix} \begin{Bmatrix} \delta \boldsymbol{q} \\ \delta \boldsymbol{\phi} \\ \delta \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \hat{\boldsymbol{f}} \\ \boldsymbol{h} \\ \boldsymbol{c} \end{Bmatrix}.$$

I prefer *monolithic* solution schemes!

RK, M. Tůma, J. Mäkinen, A. Fedoroff, H. Marjamäki, Implementation of a direct procedure for critical point computations using preconditioned iterative solvers, *Computers and Structures*, **108-109**, 2012, 110-117

The crucial thing is the **preconditioner**

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{K}_f & \boldsymbol{0} & \boldsymbol{P} \\ \boldsymbol{Z} & \boldsymbol{K}_h & \boldsymbol{N} \\ \boldsymbol{C}_q & \boldsymbol{C}_\phi & \boldsymbol{C}_\lambda \end{bmatrix}, \qquad \boldsymbol{M} = \begin{bmatrix} \boldsymbol{M}_f & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{Z} & \boldsymbol{M}_h & \boldsymbol{0} \\ \boldsymbol{C}_q & \boldsymbol{C}_\phi & \boldsymbol{D} \end{bmatrix}.$$

$$\boldsymbol{M}^{-1} = \begin{bmatrix} \boldsymbol{M}_f^{-1} & \boldsymbol{0} & \boldsymbol{0} \\ -\boldsymbol{M}_h^{-1}\boldsymbol{Z}\boldsymbol{M}_f^{-1} & \boldsymbol{M}_h^{-1} & \boldsymbol{0} \\ -\boldsymbol{D}^{-1}(\boldsymbol{C}_q\boldsymbol{M}_f^{-1} - \boldsymbol{C}_\phi\boldsymbol{M}_h^{-1}\boldsymbol{Z}\boldsymbol{M}_f^{-1}) & -\boldsymbol{D}^{-1}\boldsymbol{C}_\phi\boldsymbol{M}_h^{-1} & \boldsymbol{D}^{-1} \end{bmatrix}.$$

Preconditioning operation $\boldsymbol{y} = \boldsymbol{M}^{-1}\boldsymbol{s}$

$$\begin{Bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \boldsymbol{y}_3 \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{M}_f^{-1}\boldsymbol{s}_1 \\ \boldsymbol{M}_h^{-1}(\boldsymbol{s}_2 - \boldsymbol{Z}\boldsymbol{y}_1) \\ \boldsymbol{D}^{-1}\left(\boldsymbol{s}_3 - \boldsymbol{C}_\phi\boldsymbol{y}_2 - \boldsymbol{C}_q\boldsymbol{y}_1\right) \end{Bmatrix}.$$

Error analysis

Accuracy: $\mu_1(\boldsymbol{A}) = \|\boldsymbol{A} - \boldsymbol{M}\|_F$

Stability: $\mu_2(\boldsymbol{A}) = \|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_F$

If $\boldsymbol{K}_f = \boldsymbol{K}_h = \boldsymbol{K}$, thus $\boldsymbol{M}_f = \boldsymbol{M}_h = \boldsymbol{M}_1$, results in

$$\mu_1(\boldsymbol{A}) = \sqrt{2\mu_1(\boldsymbol{K})^2 + \|\boldsymbol{P}\|_F^2 + \|\boldsymbol{N}\|_F^2 + \|\boldsymbol{C}_\lambda - \boldsymbol{D}^{-1}\|_F^2},$$

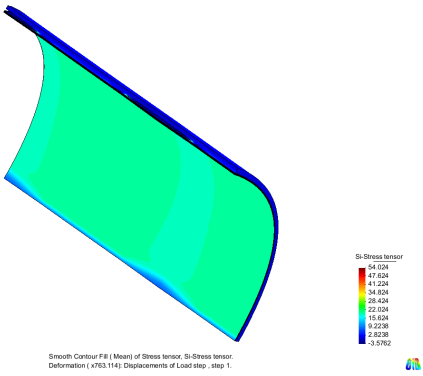
where $\mu_1(\boldsymbol{K}) = \|\boldsymbol{K} - \boldsymbol{M}_1\|$.

$$\mu_2(\boldsymbol{A}) \leq \eta_1\mu_2(\boldsymbol{K}) + \eta_2\|\boldsymbol{R}_{13}\|_F + \eta_3\|\boldsymbol{R}_{23}\|_F,$$

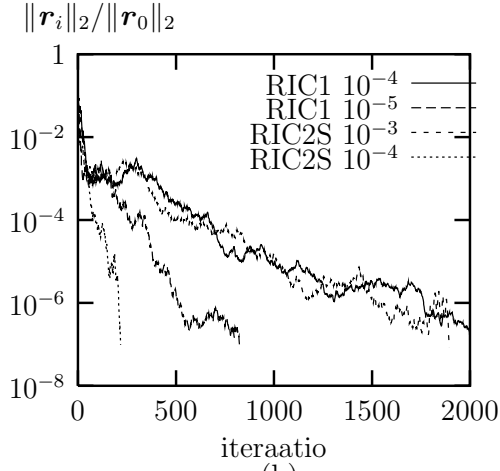
where

$$\boldsymbol{R}_{13} = \boldsymbol{M}_1^{-1}\boldsymbol{P}, \quad \boldsymbol{R}_{23} = \boldsymbol{M}_1^{-1}\boldsymbol{N} - \boldsymbol{M}_1^{-1}\boldsymbol{Z}\boldsymbol{M}_1^{-1}\boldsymbol{P}.$$

Some examples - Shell type structure analysed as 3D-solid



(a)



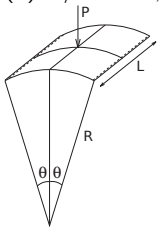
(b)

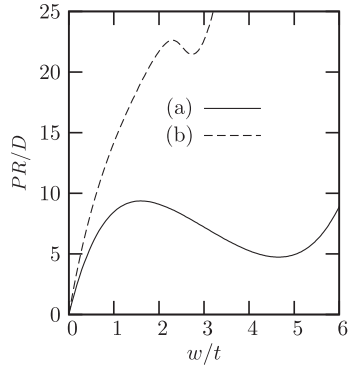
pohjustin	ψ	ψ_2	prec. dens.	it	p-time	i-time	p+i
RIC1	10^{-4}		3,3	2005	25,9	2360,4	2386,3
RIC1	10^{-5}		8,1	824	95,7	1851,1	1946,8
RIC1	10^{-6}		15,6	297	305,1	660,2	965,3
RIC2S	10^{-3}	10^{-6}	1,1	1864	59,4	779,6	839,0
RIC2S	$2 \cdot 10^{-4}$	10^{-6}	2,7	435	167,5	308,2	475,7
RIC2S	10^{-4}	10^{-6}	3,9	229	273,1	217,5	490,6

Stability analysis of a shallow shell

One layer with 27-node triquadratic 3D-solid elements.

(a) $R/t = 100, \theta = 0.279$ rad
(b) $R/t = 400, \theta = 0.2$ rad.





Cylindrical shell, 64×64-mesh, solution times in seconds. BE denotes the block elimination strategy. The preconditioned iterations two values are given: the first one is the preconditioner setup time and the second one is the acceleration iteration time.

		preconditioner			
method	R/t	type	ψ	av. it.	sol. time
BE-SKY	100	-	-	-	10910
BE-SKYB		-	-	-	1670
Bi-CGSTAB		RIC2S$_i$	10^{-3}	229	360+3861 = 4221
		RIC2S$_0$	10^{-3}	166	50+2558 = 2608
		RIC2S$_0$	10^{-4}	35	875+1014 = 1889
		SKYB$_0$	-	28	179+2754 =2933
BE-SKYB	400	-	-	-	1840
Bi-CGSTAB		RIC2S$_i$	10^{-3}	349	832+7389 = 8221
		RIC2S$_0$	10^{-4}	62	991+2219 = 3210
		SKYB$_0$	-	42	198+5571 = 5769

HP ProLiant DL785 G5 server with quad-core AMD Opteron 8360 SE (Barcelona) 2.5 GHz processor, 512 GB memory, at the CSC-IT Center for Science at Espoo, Finland.

Coupled problems - salt migration known as Elder's problem

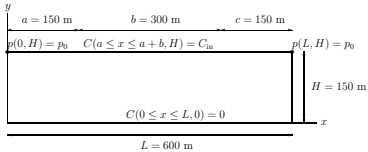
Unknowns are pressure p and salt concentration C:

$$\eta \left(\kappa_{\mathsf{f}} \frac{\partial p}{\partial t} + \zeta_{\mathsf{c}} \frac{\partial C}{\partial t} \right) + \eta \left(\kappa_{\mathsf{f}} \operatorname{grad} p + \zeta_{\mathsf{c}} \operatorname{grad} C \right) \cdot \boldsymbol{J}_{\mathsf{f}} + \operatorname{div} \boldsymbol{J}_{\mathsf{f}} = 0,$$

$$\eta \frac{\partial C}{\partial t} + \operatorname{grad} C \cdot \boldsymbol{J}_{\mathsf{f}} + \left(\kappa_{\mathsf{f}} \operatorname{grad} p + \zeta_{\mathsf{c}} \operatorname{grad} C \right) \cdot \boldsymbol{J}_{\mathsf{c}} + \operatorname{div} \boldsymbol{J}_{\mathsf{c}} = 0,$$

with fluxes

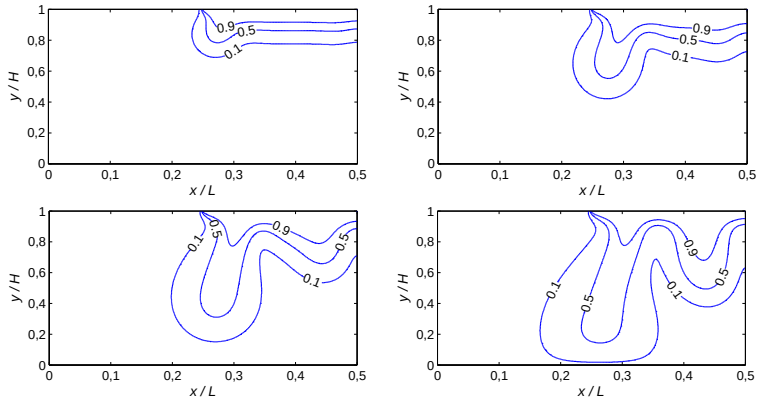
$$\boldsymbol{J}_{\mathsf{f}} = -\frac{k}{\mu} \left[\operatorname{grad} p - \left((1 - C) \bar{\rho}_{\mathsf{w}} + C \bar{\rho}_{\mathsf{c}} \right) \boldsymbol{g} \right], \quad \boldsymbol{J}_{\mathsf{c}} = -\eta D \left[a_1 \operatorname{grad} C - a_0 C (1 - C) (\bar{\rho}_{\mathsf{c}} - \bar{\rho}_{\mathsf{w}}) \boldsymbol{g} \right],$$



J. Hartikainen, RK, T. Wallroth. Permafrost simulations at Forsmark using a numerical 2D thermo-hydro-chemical model. Technical Report SKB TR-09-17, 2010.

Elder problem - evolution of salt concentration

After 1,2,3 and 4 years. 88-timesteps, maximum timestep 0.05 year.



Elder's problem -computational statistics

343643 unknowns, 88 time-steps, Newton-Raphson iteration for the non-linear system, ILUT-preconditioner with tolerance ψ

ψ	prec. dens.	ave it	max. it	total. it	ave. p-time	ave. i-time	total p+i
$1 \cdot 10^{-2}$	1,2	38	359	49608	0.7	18.6	28727
$5 \cdot 10^{-3}$	1,5	36	259	37209	0.9	22.0	23895
$1 \cdot 10^{-3}$	2,6	20	121	20367	2.0	19.2	21420
$5 \cdot 10^{-4}$	3,3	18	61	15529	2.7	14.9	15706
$1 \cdot 10^{-4}$	5,4	9	32	8617	6.5	12.9	18424
$8 \cdot 10^{-5}$	5,8	8	29	8060	7.4	11.7	18675
$5 \cdot 10^{-5}$	6,3	7	26	7247	9.0	10.1	20034
$1 \cdot 10^{-5}$	7,3	8	24	6427	12.3	12.9	20642
$1 \cdot 10^{-6}$	8,1	11	24	6171	16.7	18.8	20081

Coupled problems - magnetoelasticity

Many ways to formulate the magnetic problem.

- Using magnetic vector potential $\vec{A}$, such that $\vec{B} = \operatorname{curl} \vec{A}$. Mixed type weak form

$$\begin{aligned} (\operatorname{curl} \vec{\hat{A}}, \vec{H}) + (\vec{\hat{A}}, \operatorname{grad} p) &= (\vec{\hat{A}}, \vec{J}) - [\vec{\hat{A}}, \vec{\hat{H}}], \\ (\operatorname{grad} \hat{p}, \vec{A}) &= 0, \end{aligned} \tag{5}$$

  where the inner products are defined as

$$(\vec{B}, \vec{H}) = \int_\Omega \vec{B} \cdot \vec{H} \, \mathrm{d}V, \quad \text{and} \quad [\vec{A}, \vec{H}] = \int_{\partial\Omega} \vec{A} \cdot \vec{H} \, \mathrm{d}s.$$

  In 3D curl-conforming interpolation for $\boldsymbol{A}$ should be used and standard C_0-interpoation for p.

- Using the Lagrangian functional

$$\mathcal{L} = \frac{1}{2}(\operatorname{div} \vec{B}, \operatorname{div} \vec{B}) + (\vec{p}, \operatorname{curl} \vec{H} - \vec{J}).$$

  Divergence conforming interpolation for $\vec{B}$ and curl-conforming for $\vec{p}$.

How to solve the curl-curl problem in 3D?

Saddle point problem (5) in the matrix form

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{C} \\ \boldsymbol{C}^T & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{a} \\ \boldsymbol{p} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{h} \\ \boldsymbol{0} \end{Bmatrix}$$

The leading block $\boldsymbol{H}$ is singular!

Monolithic magnetoelasticity

In strong form

$$
\begin{aligned}
-\operatorname{div} \boldsymbol{\sigma}(\vec{u}, \vec{B}) &= \rho \vec{b}, \\
\operatorname{curl} \vec{H}(\vec{B}, \vec{u}) &= \vec{J}, \\
\operatorname{div} \vec{B} &= 0,
\end{aligned}
$$

and using the vector potential approach for the magnetic part gives the following linearized discrete equation system

$$
\begin{bmatrix} \boldsymbol{K} & \boldsymbol{L} & \boldsymbol{0} \\ \boldsymbol{F} & \boldsymbol{H} & \boldsymbol{C} \\ \boldsymbol{0} & \boldsymbol{C}^T & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \delta \boldsymbol{u} \\ \delta \boldsymbol{a} \\ \delta \boldsymbol{p} \end{Bmatrix} = \boldsymbol{g}
$$

What kind of preconditioner for such a system?

Open source JuliaFEM

See www.juliafem.org! Julia programming language try to combine programming flexibility and high-performance computing.
Eigenvalue analysis for natural frequences (10 lowest pairs), 12.6 millions unknowns. Solution time 3 h 4 min with 24 x Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60 GHz with 512 GB total memory.

Tero Frondelius, Jukka Aho. JuliaFEM – open source solver for both industrial and academia use. *Rakenteiden Mekaniikka (Journal of Structural Mechanics)*. **50** (3) 2017, 229-233.

Thank you!

Bohumil Kubišta, Taming snakes

I would like to express my sincere thanks to Mirek and Michele for fruitfull and enjoyable collaboration.

I would also like to thank my numerous friends/colleaques with whom I have worked, especially
Anouar Belahcen, Juha Hartikainen, Jari Mäkinen, Paavo Rasilo
and my teachers
Professors Martti Mikkola and Markku Tuomala.