Preconditioning for large scale micro finite element analyses of 3D poroelasticity using Trilinos

Erhan Turan & Peter Arbenz

June 5, 2012

Department of Computer Science - ETH Zürich

Motivation

- Osteoporosis (second largest)
- Simulation of bone formation - Remodelling
- ParFE [1]
- ParFE-nl
- ParOSol - Cyril Flaig
- PorFE: ParFE with Poroelasticity

ETH

inf Informatik Computer Science

Geometry

Figure: ∼75M dof

Poroelasticity

$$(\lambda + \mu)\,\nabla\,(\nabla \cdot \mathbf{u}) + \mu\nabla^2\mathbf{u} - \alpha\nabla p + \mathbf{F} = 0, \tag{1}$$

$$\mathbf{f} = -\frac{k}{\eta}\nabla p. \tag{2}$$

$$\alpha\frac{\partial\epsilon_{kk}}{\partial t} + S_\epsilon\frac{\partial p}{\partial t} + \nabla\cdot\mathbf{f} = 0 \tag{3}$$

- Elasticity + Fluid Mechanics
- Linear Elasticity + Darcy Flow (Filter)
- Biot's Consolidation [4]

Bone Poroelasticity

- Structural differences
- Bone strength and stability
- Reconstructive surgery
- Physical parameters
- Complicated geometry
- Loads on the body
- Simulation!

Details on ParFE

- Trilinear elements on equal sized voxels
- Programmed in C++
- Uses Trilinos Framework [3]
- ParMETIS: Partitioning
- HDF5: Mesh, I/O
- matrix-free — matrix-ready

- Epetra, AztecOO, ML, Isorhoppia
- 4000 Cores & 1 Bn dof. (matrix-free)

Trilinos

Trilinos [3] is utilized as the parallel framework

- Belos
- IFPACK
- Amesos
- AztecOO (inner solver)
- Epetra, ML

Finite Element Formulation: (u/p) vs. **(u/f/p)**

- more dof but fewer nonzeros.
- Stable, does not need numerical experiments.
- Primary variables are kept in the system: Stokes flow
- Constant flux across boundaries: continuity

Finite Element Formulation: (u/p) vs. **(u/f/p)**

- more dof but fewer nonzeros.
- Stable, does not need numerical experiments.
- Primary variables are kept in the system: Stokes flow
- Constant flux across boundaries: continuity

- Different FE Spaces: Q_1 for u, RT_0 for f, P_0 for p
- nodal, facial and elemental unknowns.
- The resulting system is symmetric indefinite.

Displacement/flux/pressure formulation

$$\begin{aligned}
\mathcal{A}(\mathbf{u}^h,\mathbf{v}^h) \qquad\qquad\qquad\qquad & -\mathcal{B}(p^h,\mathbf{v}^h) = 0 \\
-\mathcal{M}(\mathbf{f}^h,\mathbf{g}^h) \; & -\mathcal{B}(p^h,\mathbf{g}^h) = 0 \\
-\mathcal{B}(q^h,\mathbf{u}^h) - \mathcal{B}(q^h,\mathbf{f}^h) \; & -\mathcal{D}(p^h,q^h) = -(S,q^h)
\end{aligned} \tag{4}$$

$$\begin{bmatrix} \mathbf{A}_{uu} & 0 & \mathbf{A}_{pu}^T \\ 0 & \mathbf{A}_{ff} & \mathbf{A}_{pf}^T \\ \mathbf{A}_{pu} & \mathbf{A}_{pf} & \mathbf{A}_{pp} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{f} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{b} \end{bmatrix} \tag{5}$$

Voxel Geometry

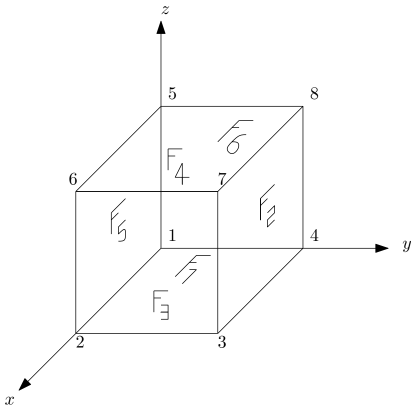
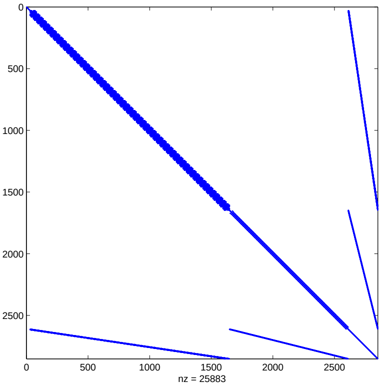


Figure: Reference element

Displacement/flux/pressure - sparsity



Difficulties

- How to deal with faces?
- How to store the 3x3 block matrix?
- How to deal with off-diagonal blocks?
- How to solve an indefinite problem?
- How to represent matrix-vector product?
- How to represent the preconditioner?
- How to deal with time dependency?

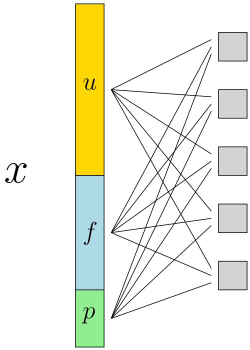
Remedies

- Faces: Element-to-Face connectivity
- Store each matrix separately: Epetra_VbrMatrix, Epetra_CrsMatrix ...
- Two maps for rows and colums. careful with bc's
- Minres: Belos! (since Trilinos 10.8)
- Define an abstract class for `Apply` method
- Next Slides
- Hard coded first order implicit Euler (FD)

How to Apply Apply I

```
PoroMatrix::PoroMatrix(Epetra_RowMatrix&   A_uu_,
        Epetra_CrsMatrix &  A_ff_,
        Epetra_Vector &     A_p_,
        Epetra_CrsMatrix &  A_pu_,
        Epetra_CrsMatrix &  A_pf_,
        const Epetra_MpiComm      comm_)
```

How to Apply `Apply` II




```
composed_map = new Epetra_Map(glob_size, loc_size, indices, 0, comm);
```

How to Apply Apply III

```
int Apply(const Epetra_MultiVector& X, Epetra_MultiVector& Y) const;

// Returns -1.
int ApplyInverse(const Epetra_MultiVector& X, Epetra_MultiVector& Y) co

bool UseTranspose()                  const { return false; };

...

const Epetra_Comm& Comm()            const { return comm; };

const Epetra_Map& OperatorDomainMap() const { return (*composed_map); }
```

How to Apply Apply IV

```
int PoroMatrix::Apply(const Epetra_MultiVector& X, Epetra_MultiVector& 

    A_uu.Apply(X_uform   ,  tmp_uform );
    for (int i=0; i<size_loc_uform; i++)
      (*Y_vec)[ i ]  = tmp_uform[i];

    A_pu.Multiply( true , X_pform   ,  tmp_uform );
    for (int i=0; i<size_loc_uform; i++)
      (*Y_vec)[ i ]  += tmp_uform[i];

    A_ff.Multiply(false, X_fform   ,  tmp_fform );
    A_pf.Multiply(true , X_pform   ,  tmp_fform );
    A_pu.Multiply( false, X_uform   ,  tmp_pform );
    A_pf.Multiply( false, X_fform   ,  tmp_pform );
    . . .
    for (int i=0; i<size_loc_pform; i++){
      (*Y_vec)[i+size_loc_uform+size_loc_fform ] += X_pform[i]*A_p[i];
```

How to Apply Apply V

```
PoroMatrix *myPoro = new PoroMatrix(*AuuP, *Aff, *Ap, *Apu, *Apf, comm)
RCP<Epetra_Operator> pA = rcp(my_poro);

Belos::LinearProblem<double, MV, OP>* my_problem =
      new Belos::LinearProblem<double, MV, OP>(pA, pX, pB);


RCP<Belos::LinearProblem<double, MV, OP> > problem  = rcp(my_problem);


RCP< Belos::SolverManager<double, MV, OP> > solver;
solver =
      rcp( new Belos::BlockGmresSolMgr<double, MV, OP>(problem, belosLi
```

Preconditioning

- Try to represent the inverse directly
- Block Diagonal preconditioner by Lipnikov [2]

$$\begin{bmatrix} \mathbf{ML} : \mathbf{A}_{uu} & 0 & 0 \\ 0 & \mathbf{A}_{ff} & 0 \\ 0 & 0 & \mathbf{ML} : \mathbf{S} \end{bmatrix} \tag{6}$$

$$\mathbf{S} = \mathbf{A}_{pp} - \mathbf{A}_{pu}\mathbf{A}_{uu}^{-1}\mathbf{A}_{pu}^{T} - \mathbf{A}_{pf}\mathbf{A}_{ff}^{-1}\mathbf{A}_{pf}^{T} \tag{7}$$

- Matrix-Matrix products!

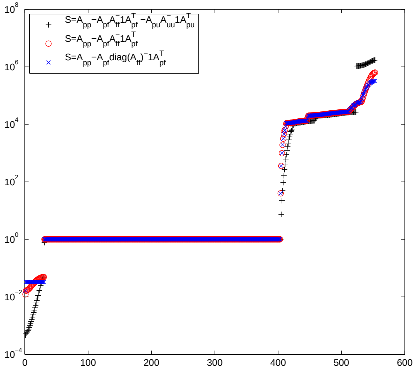
Schur Complement ($\hat{\mathbf{S}}$)

$$\mathbf{S} = \mathbf{A}_{pp} - \mathbf{A}_{pu}\mathbf{A}_{uu}^{-1}\mathbf{A}_{pu}^{T} - \mathbf{A}_{pf}\mathbf{A}_{ff}^{-1}\mathbf{A}_{pf}^{T} \tag{8}$$

$$\mathbf{S} \approx \hat{\mathbf{S}} = \mathbf{A}_{pp} - \mathbf{A}_{pf}\,diag\left(\mathbf{A}_{ff}\right)^{-1}\mathbf{A}_{pf}^{T} \tag{9}$$

- $\mathbf{S}$ and $\hat{\mathbf{S}}$ are spectrally equivalent.
- $\mathbf{A}_{pp}$ is a Epetra_Vector but the $\hat{\mathbf{S}}$ is a Epetra_CrsMatrix.
- $\hat{\mathbf{S}}$ can be generated directly considering the neighborhood of an element.

Eigenvalue spectrum ($\hat{\mathbf{S}}$)



Preconditioner A

- $\mathbf{ML} : \mathbf{A}_{uu}$ implemented in ParFE
- Multilevel and Schur Complement

$$\begin{bmatrix} \mathbf{ML} : \mathbf{A}_{uu} & 0 & 0 \\ 0 & diag\left(\mathbf{A}_{ff}\right) & 0 \\ 0 & 0 & \mathbf{ML} : \hat{\mathbf{S}} \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} \mathbf{ML} : \mathbf{A}_{uu} & 0 & 0 \\ 0 & \mathbf{A}_{ff} & 0 \\ 0 & 0 & \mathbf{ML} : \hat{\mathbf{S}} \end{bmatrix} \tag{11}$$

Preconditioner B

$$\begin{bmatrix} \mathbf{PCG} + \mathbf{ML} : \mathbf{A}_{uu} & 0 & \mathbf{A}_{pu}^T \\ 0 & IC : \mathbf{A}_{ff} & \mathbf{A}_{pf}^T \\ 0 & 0 & \mathbf{PCG} + \mathbf{ML} : \hat{\mathbf{S}} \end{bmatrix} \tag{12}$$

$$\begin{bmatrix} \mathbf{PCG} + \mathbf{ML} : \mathbf{A}_{uu} & 0 & \mathbf{A}_{pu}^T \\ 0 & \mathbf{PCG} + \mathbf{IC} : \mathbf{A}_{ff} & \mathbf{A}_{pf}^T \\ 0 & 0 & \mathbf{PCG} + \mathbf{ML} : \hat{\mathbf{S}} \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} \mathbf{PCG} + \mathbf{ML} : \mathbf{A}_{uu} & 0 & \mathbf{A}_{pu}^T \\ 0 & \mathbf{AMESOS} : \mathbf{A}_{ff} & \mathbf{A}_{pf}^T \\ 0 & 0 & \mathbf{PCG} + \mathbf{ML} : \hat{\mathbf{S}} \end{bmatrix} \tag{14}$$

How to Apply ApplyInverse - A

```
int PoroMatrixPreconditioner::
    ApplyInverse(const Epetra_MultiVector& X, Epetra_MultiVector& Y) co

    MA_uu->ApplyInverse(X_uform  ,  tmp_uform );
    for (int i=0; i<size_loc_uform; i++)
      (*Y_vec)[ i ] = tmp_uform[i];

    // Xf/Af
    for (int i=0; i<size_loc_fform; i++)
      (*Y_vec)[ i + size_loc_uform ]  = X_fform[ i ]/A_f[i];

    MS_pp->ApplyInverse(X_pform  ,  tmp_pform );

    for (int i=0; i<size_loc_pform; i++)
      (*Y_vec)[ i + size_loc_uform + size_loc_fform ]   = tmp_pform[i];
}
```

How to Apply ApplyInverse - B I

Constructor:

```
mySppProblem = new Epetra_LinearProblem;
mySppProblem->SetOperator(&S_pp);

myAuuProblem = new Epetra_LinearProblem;
myAuuProblem->SetOperator(&A_uu);

myAffProblem = new Epetra_LinearProblem;
myAffProblem->SetOperator(&A_ff);

myAffSolver->SymbolicFactorization(); //!
myAffSolver->NumericFactorization(); //!

Prec = iFactory.Create(PrecType, &A_ff, OverlapLevel);
Prec->Compute();
```

How to Apply ApplyInverse - B II

In ApplyInverse:

```
mySppProblem->SetLHS(&tmp_pform);
mySppProblem->SetRHS(&X_pform);

AztecOO mySppSolver(*mySppProblem);
mySppSolver.SetAztecOption(AZ_solver, AZ_cg);
mySppSolver.SetAztecOption(AZ_output, AZ_none);
mySppSolver.SetPrecOperator(MS_pp);

mySppSolver.Iterate (50, SppTol);

tmp_pform=mySppProblem->GetLHS()->operator()(0);

for (int i=0; i<size_loc_pform; i++)
  (*Y_vec)[ i + size_loc_uform + size_loc_fform ]   = tmp_pform[i];
```

How to Apply ApplyInverse - B III

Off-diagonal blocks' update:

```
A_pu.Multiply(true , tmp_pform  ,  tmp_uform );

for (int i=0; i<size_loc_uform; i++)
  tmp_uform[i]                       = X_uform[i] - tmp_uform[i];

// A_ff.Multiply(false , tmp_fform  ,  tmp_fform ); /// yf is zero

A_pf.Multiply(true , tmp_pform  ,  tmp_fform );

for (int i=0; i<size_loc_fform; i++)
  tmp_fform[i]                       = X_fform[i] - tmp_fform[i];
```

Variables

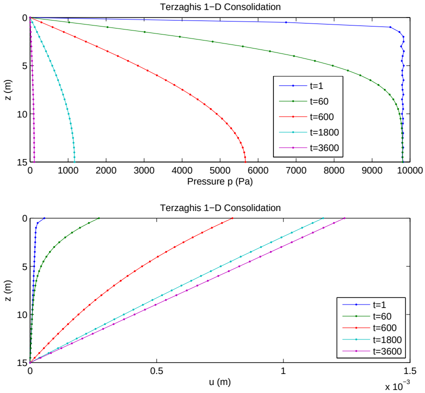
- Linear Maps for new blocks or ParMetis on $\mathbf{A}_{uu}$, adapted by $\mathbf{A}_{pu}$, $\hat{\mathbf{S}}$
- Matrix entries are evenly distributed (row-wise)
- Epetra_VbrMatrix - $\mathbf{A}_{uu}$
- Epetra_CrsMatrix - $\mathbf{A}_{ff}$, $\mathbf{A}_{pf}$, $\mathbf{A}_{pu}$, $\hat{\mathbf{S}}$
- Epetra_Vector - $\mathbf{A}_{pp}$
- Each primitive variable is distributed along the processors.

Model Definition

parameter	value
λ	40.0 MPa
μ	40.0 MPa
α	1
k	1.02×10^{-6} mm^2
η	1.0×10^{-9} MPa s
S_ϵ	1.65×10^{-4} MPa^{-1}

- Terzaghi's 1D Consolidation (Transient analytical)
- Modeled with 3D geometries using proper bc's
- upto 36000 time steps
- FGMRES

Validation

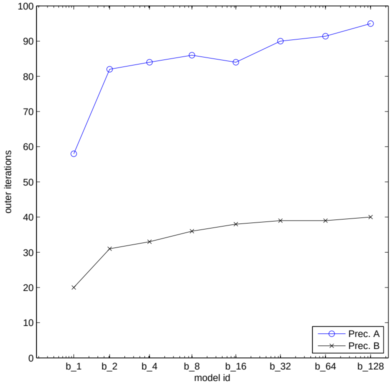

Terzaghis 1−D Consolidation
z (m)
Pressure p (Pa)
0
5
10
15
0 1000 2000 3000 4000 5000 6000 7000 8000 9000 10000
t=1
t=60
t=600
t=1800
t=3600
Terzaghis 1−D Consolidation
z (m)
u (m)
0
5
10
15
0 0.5 1 1.5
x 10^−3
t=1
t=60
t=600
t=1800
t=3600

Test Models

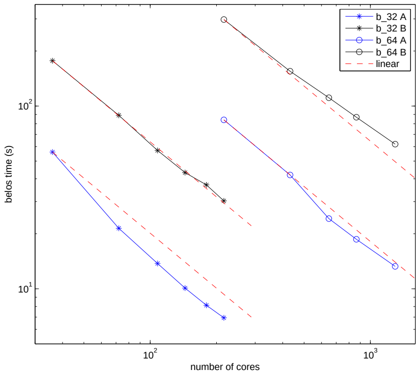
mesh_id	elements	nodes	faces	total dof
b1_1	30	124	151	553
b1_2	240	529	964	2 851
b1_4	1 920	3 025	6 736	17 731
b1_8	15 360	19 521	49 984	123 907
b1_16	122 880	139 009	384 256	924 163
b1_32	983 040	1 046 529	3 011 584	7 134 211
b1_64	7 864 320	8 116 225	23 842 816	56 055 811
b1_128	62 914 560	63 918 081	189 743 104	444 411 904

Table: Test meshes for the first benchmark problem

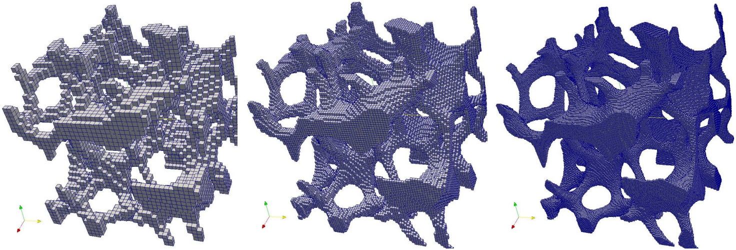
Number of iterations



Strong scalability (belos)



Samples

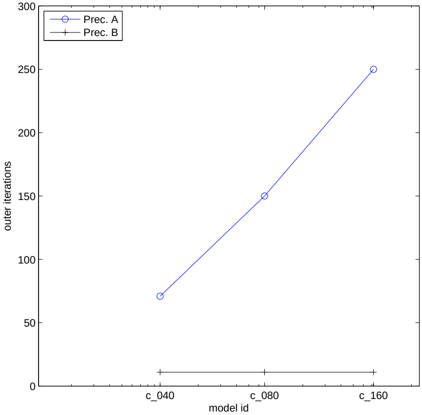


Test Models

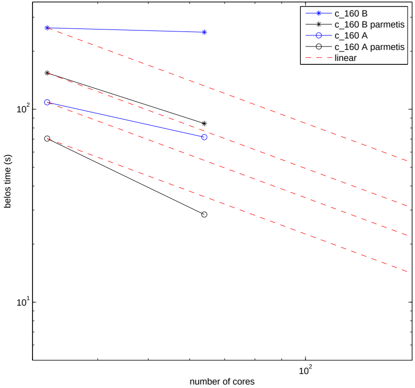
mesh_id	elements	nodes	faces	total dof
c_40	8 737	16 356	32 866	90 671
c_80	69 837	99121	236772	603 972
c_160	557 691	671 995	1 783 221	4 356 897
w	9 013 446	12 178 452	30 063 142	75 611 944

Table: Test meshes on a bone sample

Number of iterations



Strong scalability (belos)



Summary

- ParFE extendend to include poroelastic effects
- Belos is the main solver. AztecOO for subblocks.
- Mixed finite element formulation is implemented
- Improvements on preconditioners.

Acknowledgments

- Prof. Dr. Peter Arbenz & Group Arbenz

References

- P. Arbenz, G. H. van Lenthe, U. Mennel, R. Müller, and M. Sala.
  A scalable multi-level preconditioner for matrix-free μ-finite element analysis of human bone structures.
  Internat. J. Numer. Methods Engrg., 73(7):927–947, 2008.
- K Lipnikov.
  Numerical Methods for the Biot Model in Poroelasticity.
  PhD thesis, University of Houston, 2002.
- The Trilinos Project Home Page.
  `http://trilinos.sandia.gov/`.
- H F Wang.
  Theory of Linear Poroelasticity with Applications to Geomechanics and Hydrogeology.
  Princeton University Press, New Jersey, 2000.

Numbering strategy

- Faces are numbered following the elements one by one.
- Free faces are also counted.
- u+f+p

Typical Values of Physical Parameters

parameter	definition	typical values	dimension
λ	Lamé parameter	5.72×10^9	Pa
$\mu\,(\equiv G)$	Shear modulus	5.94×10^9	Pa
α	Biot-Willis coefficient	0.151	-
k	permeability	1.1×10^{-21}	m^2
η	dynamic viscosity	1×10^{-3}	Pa s
S_ϵ	constrained specific storage	0.0275×10^{-9}	Pa^{-1}

Secondary Parameters

parameter	definition	typical values	dimension
E	Young's modulus	15.7×10^9	Pa $\left(= \frac{kg}{ms^2}\right)$
ν	Poisson's ratio	0.325	-
K	Bulk modulus	14.99×10^9	Pa $\left(= \frac{kg}{ms^2}\right)$
B	Skempton's Coefficient	0.344	-
M	Biot Modulus	33.60×10^9	Pa $\left(= \frac{kg}{ms^2}\right)$
ϕ	Porosity	0.05	-

Table: List of Secondary Parameters

Comparison

n^3 elements	$\mathbf{u}/p$	$\mathbf{u}/\mathbf{f}/p$
local DOF	32 $(3\times8 + 1\times8)$	31 $(3\times8 + 1\times6 + 1)$
total DOF	$4(n+1)^3$	$3(n+1)^3 + 3n^2(n+1) + n^3$
local nonzeros	808	457
total nonzeros	$\approx 297n^3$	$\approx 229n^3$

Table: Comparison of both formulations