# Scientific applications of SfePy

Robert Cimrman & Vladimír Lukeš

Department of Mechanics & New Technologies Research Centre
University of West Bohemia in Plzeň, Czech Republic

EuroSciPy 2009, July 25, Leipzig, Germany

ZÁPADOČESKÁ UNIVERZITA V PLZNI

NEW TECHNOLOGIES RESEARCH CENTRE

SfePy

# Outline


1. Introduction
2. Choice of Language
   - Mixing Languages — Best of Both Worlds
   - Software Dependencies
3. Example
   - Setting
   - Problem Description File
   - Running SfePy
   - Time-dependent Problem
   - Top-level Scripts
4. Applications (Homogenization)
   - Prediction of Acoustic Band Gaps
   - Acoustic Waves in Thin Perforated Layer
   - Porous Media
5. Conclusion

# Introduction

- SfePy = simple finite elements in Python
  - general finite element analysis software
  - solving systems of PDEs
- BSD open-source license
- now multi-platform! (Linux, Mac OS X, Windows)
- available at
  - `http://sfepy.org` (developers)
    - mailing lists, issue (bug) tracking
    - we encourage and support everyone who joins!
  - `http://sfepy.kme.zcu.cz` (project information)
- selected applications:
  - homogenization of porous media (parallel flows in a deformable porous medium)
  - acoustic band gaps (homogenization of a strongly heterogenous elastic structure: phononic materials)
  - acoustic waves in thin perforated layers
  - shape optimization in incompressible flow problems
  - finite element formulation of Schrödinger equation

# Introduction

- SfePy = simple finite elements in Python
  - general finite element analysis software
  - solving systems of PDEs
- BSD open-source license
- now multi-platform! (Linux, Mac OS X, Windows)
- available at
  - `http://sfepy.org` (developers)
    - mailing lists, issue (bug) tracking
    - we encourage and support everyone who joins!
  - `http://sfepy.kme.zcu.cz` (project information)
- selected applications:
  - homogenization of porous media (parallel flows in a deformable porous medium)
  - acoustic band gaps (homogenization of a strongly heterogenous elastic structure: phononic materials)
  - acoustic waves in thin perforated layers
  - shape optimization in incompressible flow problems
  - finite element formulation of Schrödinger equation

# Mixing Languages — Best of Both Worlds

- low level code (C): element matrix evaluations, ...
- high level code (Python): logic of the code, particular applications, configuration files, problem description files

www.python.org

SfePy = Python + C (+ fortran)

- notable features:
  - small size (complete sources + examples with meshes are just about 4.6 MB, May 2009)
  - problem description files in pure Python
  - problem description form similar to mathematical description "on paper"
  - `isfepy` ... interactive shell based in IPython

# Mixing Languages — Best of Both Worlds

- low level code (C): element matrix evaluations, ...
- high level code (Python): logic of the code, particular applications, configuration files, problem description files

www.python.org

python™

$$\text{SfePy} = \text{Python} + \text{C} \ (+ \text{ fortran})$$

- notable features:
  - small size (complete sources + examples with meshes are just about 4.6 MB, May 2009)
  - problem description files in pure Python
  - problem description form similar to mathematical description "on paper"
  - `isfepy` ... interactive shell based in IPython

# Software Dependencies

- to install and use SfePy, several other packages or libraries are needed:
  - NumPy and SciPy: free (BSD license) collection of numerical computing libraries for Python
    - enables Matlab-like array/matrix manipulations and indexing
  - other: UMFPACK, Pyparsing, Matplotlib, Pytables (+ HDF5), swig
  - visualization of results: ParaView, MayaVi2, or any other VTK-capable viewer
  - isfepy requires IPython
- missing:
  - free (BSD license) 3D mesh generation and refinement tool
  - ...can use netgen, tetgen
- but otherwise there are completely free tools for everything!

# Software Dependencies

- to install and use SfePy, several other packages or libraries are needed:
  - NumPy and SciPy: free (BSD license) collection of numerical computing libraries for Python
    - enables Matlab-like array/matrix manipulations and indexing
  - other: UMFPACK, Pyparsing, Matplotlib, Pytables (+ HDF5), swig
  - visualization of results: ParaView, MayaVi2, or any other VTK-capable viewer
  - isfepy requires IPython
- missing:
  - free (BSD license) 3D mesh generation and refinement tool
  - ...can use netgen, tetgen
- but otherwise there are completely free tools for everything!

# Setting

- problem description file is a regular Python module, i.e. all Python syntax and power is accessible
- consists of entities defining:
  - fields of various FE approximations, variables
  - equations in the weak form, quadratures
  - boundary conditions (Dirichlet, periodic, "rigid body")
  - FE mesh file name, options, solvers, ...
- simple example: the Laplace equation:

$$c\Delta u = 0 \text{ in } \Omega, \quad u = \bar{u} \text{ on } \Gamma, \text{ weak form: } \int_\Omega c\,\nabla u \cdot \nabla v = 0, \quad \forall v \in V_0$$

# Problem Description File

## Solving Laplace Equation — FE Approximations

- mesh → define FE approximation to $\Omega$:
  `filename_mesh = 'simple.mesh'`
- regions → define domain $\Omega$, regions $\Gamma_{\text{left}}$, $\Gamma_{\text{right}}$, $\Gamma = \Gamma_{\text{left}} \cup \Gamma_{\text{right}}$:
  - $_h$ omitted from now on ...

```
regions   = {
            'Omega'         :  ('all', {}),
            'Gamma_Left'    :  ('nodes in (x < 0.0001)', {}),
            'Gamma_Right'   :  ('nodes in (x > 0.0999)', {}),
}
```

- fields → define space $V_h$:

```
fields   = {
           'temperature'   :  ((1,1), 'real', 'Omega',
                              {'Omega' :  '3_4_P1'})
}
```

'3_4_P1' means P1 approximation, in 3D, on 4-node FEs (tetrahedra)

- variables → define $u_h$, $v_h$:

```
variables   = {
              'u'   :  ('unknown field', 'temperature', 0),
              'v'   :  ('test field', 'temperature', 'u'),
}
```

# Problem Description File

## Solving Laplace Equation — FE Approximations

- mesh → define FE approximation to $\Omega$:
  `filename_mesh = 'simple.mesh'`
- regions → define domain $\Omega$, regions $\Gamma_{\text{left}}$, $\Gamma_{\text{right}}$, $\Gamma = \Gamma_{\text{left}} \cup \Gamma_{\text{right}}$:
  - $_h$ omitted from now on ...

```
regions   = {
            'Omega'         :  ('all', {}),
            'Gamma_Left'    :  ('nodes in (x < 0.0001)', {}),
            'Gamma_Right'   :  ('nodes in (x > 0.0999)', {}),
}
```

- fields → define space $V_h$:

```
fields   = {
           'temperature'   :  ((1,1), 'real', 'Omega',
                               {'Omega' :  '3_4_P1'})
}
```

'3_4_P1' means P1 approximation, in 3D, on 4-node FEs (tetrahedra)

- variables → define $u_h$, $v_h$:

```
variables   = {
              'u'   :  ('unknown field', 'temperature', 0),
              'v'   :  ('test field', 'temperature', 'u'),
}
```

# Problem Description File

## Solving Laplace Equation — FE Approximations

- mesh → define FE approximation to $\Omega$:

  `filename_mesh = 'simple.mesh'`

- regions → define domain $\Omega$, regions $\Gamma_{\text{left}}$, $\Gamma_{\text{right}}$, $\Gamma = \Gamma_{\text{left}} \cup \Gamma_{\text{right}}$:
  - $_h$ omitted from now on ...

```
regions   = {
            'Omega'         :  ('all', {}),
            'Gamma_Left'    :  ('nodes in (x < 0.0001)', {}),
            'Gamma_Right'   :  ('nodes in (x > 0.0999)', {}),
}
```

- fields → define space $V_h$:

```
fields   = {
           'temperature'   :  ((1,1), 'real', 'Omega',
                               {'Omega' :  '3_4_P1'})
}
```

'3_4_P1' means P1 approximation, in 3D, on 4-node FEs (tetrahedra)

- variables → define $u_h$, $v_h$:

```
variables   = {
              'u'   :  ('unknown field', 'temperature', 0),
              'v'   :  ('test field', 'temperature', 'u'),
}
```

# Problem Description File

## Solving Laplace Equation — FE Approximations

- mesh → define FE approximation to $\Omega$:

  `filename_mesh = 'simple.mesh'`

- regions → define domain $\Omega$, regions $\Gamma_{\text{left}}$, $\Gamma_{\text{right}}$, $\Gamma = \Gamma_{\text{left}} \cup \Gamma_{\text{right}}$:
  - $_h$ omitted from now on ...

```
regions   = {
            'Omega'         :  ('all', {}),
            'Gamma_Left'    :  ('nodes in (x < 0.0001)', {}),
            'Gamma_Right'   :  ('nodes in (x > 0.0999)', {}),
}
```

- fields → define space $V_h$:

```
fields   = {
           'temperature'   :  ((1,1), 'real', 'Omega',
                               {'Omega' :  '3_4_P1'})
}
```

  '3_4_P1' means P1 approximation, in 3D, on 4-node FEs (tetrahedra)

- variables → define $u_h$, $v_h$:

```
variables   = {
              'u'   :  ('unknown field', 'temperature', 0),
              'v'   :  ('test field', 'temperature', 'u'),
}
```

# Problem Description File

## Solving Laplace Equation — Equations

- materials → define $c$:

```
materials   = {
            'm'   :  ('here', 'Omega', {'c' :  1.0}),
}
```

- integrals → define numerical quadrature:

```
integrals   = {
            'i1'   :  ('v', 'gauss_o1_d3')
}
```

- equations → define what and where should be solved:

```
equations   = {
            'eq'   :  'dw_laplace.i1.Omega( m.c, v, u ) = 0'
}
```

- Dirichlet BCs → define $\bar{u}$ on $\Gamma_{\text{left}}$, $\Gamma_{\text{right}}$:

```
ebcs   = {
       't_left'    :  ('Gamma_Left', 'u.0' :  2.0),
       't_right'   :  ('Gamma_Right', 'u.all' :  -2.0),
}
```

# Problem Description File

## Solving Laplace Equation — Equations

- materials → define $c$:

```
materials   = {
            'm'   :  ('here', 'Omega', {'c' :  1.0}),
}
```

- integrals → define numerical quadrature:

```
integrals   = {
            'i1'   :  ('v', 'gauss_o1_d3')
}
```

- equations → define what and where should be solved:

```
equations   = {
            'eq'   :  'dw_laplace.i1.Omega( m.c, v, u ) = 0'
}
```

- Dirichlet BCs → define $\bar{u}$ on $\Gamma_{\text{left}}$, $\Gamma_{\text{right}}$:

```
ebcs   = {
       't_left'    :  ('Gamma_Left', 'u.0' :  2.0),
       't_right'   :  ('Gamma_Right', 'u.all' :  -2.0),
}
```

# Problem Description File

## Solving Laplace Equation — Equations

- materials → define $c$:

```
materials   = {
            'm'   :  ('here', 'Omega', {'c' :  1.0}),
}
```

- integrals → define numerical quadrature:

```
integrals   = {
            'i1'   :  ('v', 'gauss_o1_d3')
}
```

- equations → define what and where should be solved:

```
equations   = {
            'eq'   :  'dw_laplace.i1.Omega( m.c, v, u ) = 0'
}
```

- Dirichlet BCs → define $\bar{u}$ on $\Gamma_{\text{left}}$, $\Gamma_{\text{right}}$:

```
ebcs   = {
       't_left'    :  ('Gamma_Left', 'u.0' :  2.0),
       't_right'   :  ('Gamma_Right', 'u.all' :  -2.0),
}
```

# Problem Description File

## Solving Laplace Equation — Equations

- materials → define $c$:

```
materials   = {
            'm'   :  ('here', 'Omega', {'c' :  1.0}),
}
```

- integrals → define numerical quadrature:

```
integrals   = {
            'i1'   :  ('v', 'gauss_o1_d3')
}
```

- equations → define what and where should be solved:

```
equations   = {
            'eq'   :  'dw_laplace.i1.Omega( m.c, v, u ) = 0'
}
```

- Dirichlet BCs → define $\bar{u}$ on $\Gamma_{\text{left}}$, $\Gamma_{\text{right}}$:

```
ebcs   = {
       't_left'    :  ('Gamma_Left', 'u.0' :  2.0),
       't_right'   :  ('Gamma_Right', 'u.all' :  -2.0),
}
```

# Running SfePy

```
$ ./simple.py input/poisson.py
sfepy: reading mesh (database/simple.mesh)...
sfepy: ...done in 0.00 s
sfepy: setting up domain edges...
sfepy: ...done in 0.02 s
sfepy: setting up domain faces...
sfepy: ...done in 0.03 s
sfepy: creating regions...
sfepy:     leaf Gamma_Right region_Gamma_Right
sfepy:     leaf Omega region_Omega
sfepy:     leaf Gamma_Left region_Gamma_Left
sfepy: ...done in 0.04 s
sfepy: equation "Temperature":
sfepy: dw_laplace.i1.Omega( coef.val, s, t ) = 0
sfepy: setting up dof connectivities...
sfepy: ...done in 0.00 s
sfepy: describing geometries...
sfepy: ...done in 0.00 s
sfepy: using solvers:
                nls: newton
                 ls: ls
sfepy: matrix shape: (300, 300)
sfepy: assembling matrix graph...
sfepy: ...done in 0.01 s
sfepy: matrix structural nonzeros: 3538 (3.93e-02% fill)
sfepy: updating materials...
sfepy:     coef
sfepy: ...done in 0.00 s
sfepy: nls: iter: 0, residual: 1.176265e-01 (rel: 1.000000e+00)
sfepy:   rezidual:    0.00 [s]
sfepy:      solve:    0.00 [s]
sfepy:     matrix:    0.00 [s]
sfepy: nls: iter: 1, residual: 1.173819e-16 (rel: 9.979198e-16)
```

- top level of SfePy code is a collection of executable scripts tailored for various applications
- `simple.py` is dumb script of brute force, attempting to solve any equations it finds by the Newton method
- ...exactly what we need here (solver options were omitted in previous slides)

↓

→

# Time-dependent Problem

- adding time to the Laplace equation:

$$\frac{\partial u}{\partial t} - c\Delta u = 0 \text{ in } \Omega, \quad u(\cdot, x) = \bar{u} \text{ on } \Gamma, \quad u(0, \cdot) = 0,$$

$$\text{weak form: } \int_\Omega v \cdot \frac{\partial u}{\partial t} + \int_\Omega c\, \nabla u \cdot \nabla v = 0, \quad \forall v \in V_0$$

- variables

```
variables   = {
            'u'   :  ('unknown field', 'temperature', 0, 'previous'),
            'v'   :  ('test field', 'temperature', 'u'),
}
```

- equations

```
equations   = {
            'eq'   :  """dw_mass_scalar.i1.Omega( v, du/dt )
                       + dw_laplace.i1.Omega( m.c, v, u ) = 0"""
}
```

- + define time-stepping solver options

# Time-dependent Problem

- adding time to the Laplace equation:

$$\frac{\partial u}{\partial t} - c\Delta u = 0 \text{ in } \Omega, \quad u(\cdot, x) = \bar{u} \text{ on } \Gamma, \quad u(0, \cdot) = 0,$$

$$\text{weak form: } \int_\Omega v \cdot \frac{\partial u}{\partial t} + \int_\Omega c\, \nabla u \cdot \nabla v = 0, \quad \forall v \in V_0$$

- variables

```
variables   = {
            'u'   :  ('unknown field', 'temperature', 0, 'previous'),
            'v'   :  ('test field', 'temperature', 'u'),
}
```

- equations

```
equations   = {
            'eq'   :  """dw_mass_scalar.i1.Omega( v, du/dt )
                       + dw_laplace.i1.Omega( m.c, v, u ) = 0"""
}
```

- + define time-stepping solver options

# Time-dependent Problem

- adding time to the Laplace equation:

$$\frac{\partial u}{\partial t} - c\Delta u = 0 \text{ in } \Omega, \quad u(\cdot, x) = \bar{u} \text{ on } \Gamma, \quad u(0, \cdot) = 0,$$

$$\text{weak form: } \int_\Omega v \cdot \frac{\partial u}{\partial t} + \int_\Omega c\, \nabla u \cdot \nabla v = 0, \quad \forall v \in V_0$$

- variables

```
variables   = {
            'u'   :  ('unknown field', 'temperature', 0, 'previous'),
            'v'   :  ('test field', 'temperature', 'u'),
}
```

- equations

```
equations   = {
            'eq'   :  """dw_mass_scalar.i1.Omega( v, du/dt )
                       + dw_laplace.i1.Omega( m.c, v, u ) = 0"""
}
```

- + define time-stepping solver options

# Time-dependent Problem

- adding time to the Laplace equation:

$$\frac{\partial u}{\partial t} - c\Delta u = 0 \text{ in } \Omega, \quad u(\cdot, x) = \bar{u} \text{ on } \Gamma, \quad u(0, \cdot) = 0,$$

$$\text{weak form: } \int_\Omega v \cdot \frac{\partial u}{\partial t} + \int_\Omega c\, \nabla u \cdot \nabla v = 0, \quad \forall v \in V_0$$

- variables

```
variables    = {
             'u'    :  ('unknown field', 'temperature', 0, 'previous'),
             'v'    :  ('test field', 'temperature', 'u'),
}
```

- equations

```
equations    = {
             'eq'    :  """dw_mass_scalar.i1.Omega( v, du/dt )
                         + dw_laplace.i1.Omega( m.c, v, u ) = 0"""
}
```

- + define time-stepping solver options

↓

# Top-level Scripts

Main scripts / applications (version 2009.2):

- `runTests.py` ... run all/selected unit tests
- `simple.py` ... generic problem solver (stationary, time-dependent)
- `isfepy` ... interactive SfePy interpreter
- `postproc.py` ... quick mayavi2-based visualization of results
- `eigen.py` ... application: acoustic band gaps in strongly heterogenous media
- `schroedinger.py` ... Schrödinger equation solver
- `probe.py` ... data probes plotter

Auxiliary:

- `extractor.py` ... HDF5 → VTK data extractor
- `findSurf.py` ... mesh surface extractor
- `gen` ... LaTeX documentation generator, requires pexpect, lxml
- `genPerMesh.py` ... scale and periodically repeat a reference volume mesh

# Top-level Scripts

Main scripts / applications (version 2009.2):

- `runTests.py` ... run all/selected unit tests
- `simple.py` ... generic problem solver (stationary, time-dependent)
- `isfepy` ... interactive SfePy interpreter
- `postproc.py` ... quick mayavi2-based visualization of results
- `eigen.py` ... application: acoustic band gaps in strongly heterogenous media
- `schroedinger.py` ... Schrödinger equation solver
- `probe.py` ... data probes plotter

Auxiliary:

- `extractor.py` ... HDF5 → VTK data extractor
- `findSurf.py` ... mesh surface extractor
- `gen` ... LaTeX documentation generator, requires pexpect, lxml
- `genPerMesh.py` ... scale and periodically repeat a reference volume mesh

# Prediction of Acoustic Band Gaps

- phononic materials (similar to photonic crystals but for sound)
- band gaps: frequency ranges of disabled wave propagation
  - in any periodic heteorogeneous structure
  - no band gaps in linear homogeneous material
- example applications: smart materials, wave guides, silencers
- strongly heterogeneous materials
  - constitutive parameters (elasticity, density, piezoelectricity) depend on scale $\varepsilon$

- modelling tool: homogenization
  - much easier band gap determination in low frequency range than classical methods
- homogenized material
  - metamaterial – "negative" mass
  - $\Rightarrow$ hyperbolic equation changes to elliptic one

# Prediction of Acoustic Band Gaps

piezo-phononic crystals

- microstructure decomposition
  - unknowns: $\mathbf{u}^\varepsilon$ ... displacement field, $\varphi^\varepsilon$ ... electric potential
  - $\boldsymbol{\sigma}^\varepsilon = (\sigma^\varepsilon_{ij})$ ... stress tensor, $\mathbf{D}^\varepsilon$ electric displacement

**Constitutive equations**

$$\sigma^\varepsilon_{ij} = c^\varepsilon_{ijkl} e_{kl}(\mathbf{u}^\varepsilon) - g^\varepsilon_{kij}\partial_k \varphi^\varepsilon,$$
$$D^\varepsilon_k = g^\varepsilon_{kij} e_{kl}(\mathbf{u}^\varepsilon) + d^\varepsilon_{kl}\partial_l \varphi^\varepsilon.$$

- material coefficients – scaling in $\Omega^\varepsilon_2$
  - density

$$\rho^\varepsilon(x) = \begin{cases} \rho^1 & \text{in } \Omega^\varepsilon_1, \\ \rho^2 & \text{in } \Omega^\varepsilon_2, \end{cases}$$

elasticity

$$c^\varepsilon_{ijkl}(x) = \begin{cases} c^1_{ijkl} & \text{in } \Omega^\varepsilon_1, \\ \varepsilon^2 c^2_{ijkl} & \text{in } \Omega^\varepsilon_2, \end{cases}$$

  - piezo-coupling

$$g^\varepsilon_{kij}(x) = \begin{cases} g^1_{kij} & \text{in } \Omega^\varepsilon_1, \\ \varepsilon^2 g^2_{kij} & \text{in } \Omega^\varepsilon_2, \end{cases}$$

dielectricity

$$d^\varepsilon_{ij}(x) = \begin{cases} d^1_{ij} & \text{in } \Omega^\varepsilon_1, \\ \varepsilon^2 d^2_{ij} & \text{in } \Omega^\varepsilon_2. \end{cases}$$

# Prediction of Acoustic Band Gaps

coefficient scaling – interactions & fluctuations

*local fluctuations*

$$|\text{grad}| \leqslant C$$

*macro-micro interaction*

$$|\text{grad}| \leqslant \frac{C}{\varepsilon}$$

The $\varepsilon^2$ scaling of elasticity (piezoelectricity) preserves band gap frequency intervals for $\varepsilon \to 0$.

# Finding the Band Gaps

- mass tensor: $\omega$-dependent, defined for $\omega \neq \sqrt{\lambda^r}$, $r = 1, 2, \ldots$
  - effect of anti-resonators:

$$M^*_{ij}(\omega) = -\omega^2 \sum_{r \in \mathcal{I}} \frac{1}{\omega^2 - \lambda^r} m^r_i m^r_j + \delta_{ij}\bar{\rho}\,, \qquad m^r_i = \int_{Y_2} \rho_2(y)\, \varphi^r_i(y)\, dy$$

    where $\bar{\rho}$ is the *average density* and $(\lambda^r, \varphi^r)$ are *eigenpaires* of elastic clamped inclusion,
  - when negative definite, the model has only an evanescent solution!
  - eigenvalues: $\mathbf{M}^*(\omega)\boldsymbol{\xi}^k = \mu^k \boldsymbol{\xi}^k$
- band gaps: $\omega \longrightarrow \mu^k$

| | |
|---|---|
| Strong b.g. | yellow |
| Weak b.g. | white |
| Propag. zone | green |

# Elasticity: Soft Inclusions with Rigid Heavy Particles

- aluminium matrix & epoxy inclusions with steel bodies
- shifting rigid inclusion: breaking symmetry
- weak band gaps appear

# Elasticity: Soft Inclusions with Rigid Heavy Particles

eigenproblem solutions for varying rigid inclusion shift

shift value: 0.0

# Elasticity: Soft Inclusions with Rigid Heavy Particles

eigenproblem solutions for varying rigid inclusion shift

shift value: 0.04

# Elasticity: Soft Inclusions with Rigid Heavy Particles

eigenproblem solutions for varying rigid inclusion shift

shift value: 0.08

# Elasticity: Soft Inclusions with Rigid Heavy Particles

eigenproblem solutions for varying rigid inclusion shift

shift value: 0.12

# Elasticity: Soft Inclusions with Rigid Heavy Particles

eigenproblem solutions for varying rigid inclusion shift

shift value: 0.16

# Elasticity: Soft Inclusions with Rigid Heavy Particles

eigenproblem solutions for varying rigid inclusion shift

shift value: 0.20

# Piezo-phononic Material

## eigenproblem solution

- parameters: inclusion: BaTiOx3, matrix: PZT5A
- example of eigenproblem solutions
  - top sequence - potential
  - bottom sequence - displacements

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 0°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 5°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 10°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 15°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 20°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 25°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 30°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 35°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 40°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 45°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 50°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 55°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 60°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 65°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 70°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 75°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 80°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 85°

polarization w.r.t. incident wave

wave number

# Piezo-phononic Material: Dispersion Diagrams

results for varying incident wave angles

incident wave angle in the $xy$ plane: 90°

polarization w.r.t. incident wave

wave number

# Elastic Matrix + Liquid Inclusion (no shear viscosity)

- parameters: $\lambda = 5.898$, $\mu = 2.681$, $\gamma = 0.002$, $\eta \in [10^{-10}, 10]$
- example of eigenproblem solutions, $\boldsymbol{\varphi}^k \cdot \mathbf{n} = 0$ on $\Gamma$

# Elastic Matrix + Liquid Inclusion (no shear viscosity)

band gaps for varying fluid viscosity: $\eta = 10^{-10}$

# Elastic Matrix + Liquid Inclusion (no shear viscosity)

band gaps for varying fluid viscosity: $\eta = 10^{-9}$

# Elastic Matrix + Liquid Inclusion (no shear viscosity)

band gaps for varying fluid viscosity: $\eta = 10^{-8}$

# Elastic Matrix + Liquid Inclusion (no shear viscosity)

band gaps for varying fluid viscosity: $\eta = 10^{-7}$

# Elastic Matrix + Liquid Inclusion (no shear viscosity)

band gaps for varying fluid viscosity: $\eta = 10^{-6}$

# Elastic Matrix + Liquid Inclusion (no shear viscosity)

band gaps for varying fluid viscosity: $\eta = 10^{-5}$

# Elastic Matrix + Liquid Inclusion (no shear viscosity)

band gaps for varying fluid viscosity: $\eta = 10^{-4}$

# Elastic Matrix + Liquid Inclusion (no shear viscosity)

band gaps for varying fluid viscosity: $\eta = 10^{-3}$

# Elastic Matrix + Liquid Inclusion (no shear viscosity)

band gaps for varying fluid viscosity: $\eta = 10^{-2}$

# Elastic Matrix + Liquid Inclusion (no shear viscosity)

band gaps for varying fluid viscosity: $\eta = 10^{-1}$

# Elastic Matrix + Liquid Inclusion (no shear viscosity)

band gaps for varying fluid viscosity: $\eta = 10^0$

# Elastic Matrix + Liquid Inclusion (no shear viscosity)

band gaps for varying fluid viscosity: $\eta = 10^1$

# Acoustic Waves in Thin Perforated Layer

- acoustic medium (inviscid, compressible fluid – air) + stationary incident waves – no convection, no turbulence
- problem: compute acoustic pressure $p$ in $\Omega^+$, $\Omega^-$

$$c^2\nabla^2 p + \omega^2 p = 0 \quad \text{in } \Omega^+ \cup \Omega^- \tag{1}$$

  + transmission condition on perforated plane $\Gamma_0$
- goal: to replace the perforated layer by a "homogenized surface"
- acoustic pressure $p$ – complex variable!
- allows for optimal design of perforations

# Acoustic Waves in Thin Perforated Layer

- two scale homogenization $\longrightarrow$ microscopic and macroscopic problems
- Microscopic scale (corrector functions):

$$K\xi^{\pm} = \frac{|Y|}{c^2\varkappa} f,$$
$$K\pi^{\beta} = Ky^{\beta},$$

- equations

```
equations_ksi    = {
                 'eq_ksi'    :  """dw_acoustic.i1.Y( mat_k2.param, q, p ) =
                                 + %f * dw_surface_integrate.i2.Yp( q )
                                 - %f * dw_surface_integrate.i2.Ym( q )"""
                                 % (Y_domain/(sound_vel * sound_vel * kappa),
                                 Y_domain/(sound_vel * sound_vel * kappa)),
}

equations_pi     = {
                 'eq_pi'     :  """dw_acoustic.i1.Y( mat_k2.param, q, p ) =
                                 - dw_acoustic.i1.Y( mat_k2.param, q, Pi )"""
}
```

- homogenized coefficients – expressions, requirements

```
expr_coefs    = {
                'A'    :  '%f * dw_acoustic.i1.Y( mat.par, corr1, corr2 )' % (c1,),
                'B'    :  '%f * d_surface_integrate.i2.Yp( corr1 )
                           - %f * d_surface_integrate.i2.Ym( corr1 )' % (c2, c2),
                ...
}
coefs    = {
         'A'    :  {'requires' :  ['pis', 'corrs_pi'],
                    'variables' :  ['corr1', 'corr2'],
                    'expression' :  expr_coefs['A'],
                    'class' :  ca.CoefDimM1PDimM1P},
         'B'    :  {'requires' :  ['corrs_pi'],
                    'variables' :  ['corr1'],
                    'expression' :  expr_coefs['B'],
                    'class' :  ca.CoefDimM1},
         ...
}
requirements    = {
                 'pis'         :  {'variables' :  ['p'],
                                   'class' :  ca.ShapeDimM1, },
                 'corrs_pi'    :  {'requires' :  ['pis'],
                                   'variables' :  ['Pi', 'p', 'q'],
                                   'ebcs' :  ['fixed'],
                                   'epbcs' :  all_periodic,
                                   'equations' :  equations_pi,
                                   'class' :  ca.CorrDimM1,
                                   'save_name' :  'corrs_pi',
                                   'dump_variables' :  ['p'], },
}
```

# Acoustic – examples, 2D perforation geometry

| | Mic. 2D/#1 | Mic. 2D/#2 | Mic. 2D/#3 |
|---|---|---|---|
| $\xi^{\pm}$ | | | |
| $\pi^1$ | | | |
| $A$ [(m/s)$^2$] | 99271.58 | 77973.96 | 51535.28 |
| $B$ [m] | 0 | $-0.112957$ | $-0.452692$ |
| $F$ [s$^2$] | $1.405082 \times 10^{-5}$ | $1.530726 \times 10^{-5}$ | $3.344039 \times 10^{-5}$ |

# Acoustic – examples, 3D perforation geometry

| | Mic. 3D/#1 | Mic. 3D/#2 | Mic. 3D/#3 |
|---|---|---|---|
| | | | |
| $A$ [(m/s)$^2$] | $\begin{bmatrix} 98415 & 0 \\ 0 & 98415 \end{bmatrix}$ | $\begin{bmatrix} 98654 & 207 \\ 207 & 98155 \end{bmatrix}$ | $\begin{bmatrix} 75295 & 0 \\ 0 & 81814 \end{bmatrix}$ |
| $B$ [m] | $\begin{bmatrix} 0.0 & 0.0 \end{bmatrix}$ | $\begin{bmatrix} 0.0 & 0.0 \end{bmatrix}$ | $\begin{bmatrix} 0.14233 & 0.14233 \end{bmatrix}$ |
| $F$ [s$^2$] | $1.754429 \times 10^{-5}$ | $1.647584 \times 10^{-5}$ | $2.838839 \times 10^{-5}$ |

# Acoustic Waves in Thin Perforated Layer

- Macroscopic scale (acoustic waveguide):

- acoustic pressure $p$ – complex variable

```
fields    = {
            'pressure'    :  ((1,1), 'complex', 'Omega', {'Omega' :  '3_8_Q1' }),
}
```

# Formulation of the optimal perforation design problem

- Aim of the optimization:
  to minimize/maximize the transmission losses
  $\Phi(p) = 20\,log\left(\frac{|\bar{p}_{|\Gamma_{\rm in}}|}{|p_{|\Gamma_{\rm out}}|}\right)$
  by designing the microstructure of $\Gamma_0$.

- complex sensitivity analysis of hierarchical structure

# Porous Media

- modeling blood perfusion in muscles, brain
- compact bone poromechanics
- fading memory effects
- stress recovery at micro-level

microstructure: two channels + porous matrix

microscopic corrector functions

# Modelling of dual porous fluid saturated layers

- cerebral blood flow – challenge for microstructure-related modelling
- perfusion – hierarchical levels, we focus on the lower level hierarchies

# Modelling of dual porous fluid saturated layers

- geometry – layred structure

- channels $\Omega_A^{\varepsilon\delta}$, $\Omega_B^{\varepsilon\delta}$ & matrix $\Omega_M^{\varepsilon\delta}$
- double-porous medium: strongly heterogeneous permeability
- two scale homogenization $\longrightarrow$ microscopic and macroscopic subproblems

# Modelling of dual porous fluid saturated layers

- microscopic scale – local subproblems for correctors
  - in channels A,B

$\pi_A$

$\pi_B$

$\gamma_A^1$

  - in matrix M

$\eta_A$

$\gamma^+$

$\gamma^-$

# Modelling of dual porous fluid saturated layers

- correctors $\longrightarrow$ homogenized coefficients
  - homogenized in-plane permeability of channels A, B – $\mathcal{K}^A_{\alpha\beta}$, $\mathcal{K}^B_{\alpha\beta}$
  - homogenized interface permeability – $\mathcal{G}$
  - matrix drainage coefficient – $\mathcal{F}^{A+/-}$, $\mathcal{F}^{B+/-}$
  - branch saturation coefficients – $\mathcal{S}^{A,k}_\alpha$, $\mathcal{S}^{B,k}_\alpha$

```
coefs   = {
        'G'     :  {'requires' :  ['corrs_etaA'],
                    'variables' :  ['corr_M1'],
                    'expression' :  expr_coefs['G'],
                    'class' :  ca.CoefScalar},
        'SA'    :  {'requires' :  ['corrs_gammaA'],
                    'variables' :  ['corr_A1'],
                    'expression' :  expr_coefs['SA'],
                    'regions' :  { 'bYAX' : ['bYA1', 'bYA2', 'bYA3'] },
                    'class' :  cp.CoefRegion},
        ...
}
```

# Modelling of dual porous fluid saturated layers

## Further work

- multiple layers — hierarchical structure
- non-stationary problem
- 3D implementation
- convected diffusion of the contrast fluid

# Compact Bone Segment

## geometry

macro

micro M1: disconnected channels

micro M2: connected channels

# Compact Bone Segment

loads, boundary conditions, macro: overview

M1 (magnified displacements)

M2 (magnified displacements)

- bottom plane: $u_3 = 0$, A: $\mathbf{u} = 0$, B: $u_1 = 0$
- top plane: traction force $\mathbf{g} = [0, 0, g_3]$,
  $g_3 = a(t - (t - t_0)H(t - t_0)) + bx_1 \sin(\omega t)$, $t \in [0, 20]$ s, $t_0 = 2$ s
- insulated surface

# Micro-solutions and coefficients

M1

- examples of correctors

- fading memory coefficients

- permeability $\mathcal{C}_{ij}$

$$\begin{bmatrix} 0.00182 & -0.00001 & . \\ -0.00001 & 0.00410 & . \\ . & . & . \end{bmatrix}$$

M2

$$\begin{bmatrix} 0.00182 & -0.00001 & 0 \\ -0.00001 & 0.00412 & -0.00048 \\ 0 & -0.00048 & 0.01272 \end{bmatrix}$$

# Macro-solutions
## in selected points

- average values in two selected elements
- pressure

- diffusion velocity

# Micro-recovery

## in a macroscopic point

- displacements $\mathbf{u}^{\mathrm{mic}}$, pressure $p^{\mathrm{mic}}$
- micro-level $\ell^{\mu_1}$, sub-micro-level $\ell^{\mu_2}$ diffusion velocities

# Conclusion

- What is done:
  - basic FE element engine:
    - finite-dimensional approximations of continuous fields
    - variables, boundary conditions, FE assembling
    - equations, terms, regions
    - materials, material caches
  - various solvers accessed via abstract interface
  - unit tests, automatic documentation generation
  - mostly linear problems, but multiphysical
- What is not done:
  - general FE engine, possibly with symbolic evaluation (SymPy)
  - good documentation
  - fast problem-specific solvers (!)
  - adaptive mesh refinement (!)
  - parallelization (petsc4py)
  - GUI
  - real symbolic parsing/evaluation of equations

http://sfepy.org

# Conclusion

- What is done:
  - basic FE element engine:
    - finite-dimensional approximations of continuous fields
    - variables, boundary conditions, FE assembling
    - equations, terms, regions
    - materials, material caches
  - various solvers accessed via abstract interface
  - unit tests, automatic documentation generation
  - mostly linear problems, but multiphysical
- What is not done:
  - general FE engine, possibly with symbolic evaluation (SymPy)
  - good documentation
  - fast problem-specific solvers (!)
  - adaptive mesh refinement (!)
  - parallelization (petsc4py)
  - GUI
  - real symbolic parsing/evaluation of equations

http://sfepy.org

# Conclusion

- What is done:
  - basic FE element engine:
    - finite-dimensional approximations of continuous fields
    - variables, boundary conditions, FE assembling
    - equations, terms, regions
    - materials, material caches
  - various solvers accessed via abstract interface
  - unit tests, automatic documentation generation
  - mostly linear problems, but multiphysical
- What is not done:
  - general FE engine, possibly with symbolic evaluation (SymPy)
  - good documentation
  - fast problem-specific solvers (!)
  - adaptive mesh refinement (!)
  - parallelization (petsc4py)
  - GUI
  - real symbolic parsing/evaluation of equations

http://sfepy.org

# Conclusion

- What is done:
  - basic FE element engine:
    - finite-dimensional approximations of continuous fields
    - variables, boundary conditions, FE assembling
    - equations, terms, regions
    - materials, material caches
  - various solvers accessed via abstract interface
  - unit tests, automatic documentation generation
  - mostly linear problems, but multiphysical
- What is not done:
  - general FE engine, possibly with symbolic evaluation (SymPy)
  - good documentation
  - fast problem-specific solvers (!)
  - adaptive mesh refinement (!)
  - parallelization (petsc4py)
  - GUI
  - real symbolic parsing/evaluation of equations

`http://sfepy.org`

# References I

The research was financially supported by the Grant Agency of the Czech Republic, project GAČR 106/09/0740, and the Czech Government project MSM 497751303.

A. Ávila, G. Griso, B. Miara, and E. Rohan.
Multiscale modeling of elastic waves: Theoretical justification and numerical simulation of band gaps.
*Multiscale Modeling & Simulation*, 7(1):1–21, 2008.

R. Cimrman et al.
SfePy: simple finite elements in python, 2009.
http://sfepy.kme.zcu.cz, http://sfepy.org.

G. Griso and E. Rohan.
On the homogenization of a diffusion-deformation problem in strongly heterogeneous media.
*Ricerche di Matematica*, 56(2):161–188, 2007.

E. Rohan.
Modelling large deformation induced microflow in soft biological tissues.
*Theor. Comput. Fluid Dynamics*, (20):251–276, 2006.

E. Rohan, R. Cimrman, S. Naili, and T. Lemaire.
Homogenized model of bone poroelasticity and deformation induced microflow.
In *Proceedings of the 8th World Congress on Computational Mechanics*, 2008.

E. Rohan, R. Cimrman, S. Naili, and T. Lemaire.
Multiscale modelling of a fluid saturated medium with dual porosity: application to the compact bone.
*Int. Jour. Solids and Structures*, 2009.
submitted.

# References II

E. Rohan, B. Miara, and F. Seifrt.
Numerical simulation of acoustic band gaps in homogenized elastic composites.
*accepted to Int. J. Eng. Sci.*, 2009.
doi:10.1016/j.ijengsci.2008.12.003.

# This is not a slide!