NOVÉ TECHNOLOGIE
VÝZKUMNÉ CENTRUM
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

MODELOVÁNÍ HETEROGENNÍCH MATERIÁLŮ A BIOMECHANICKÝCH SYSTÉMŮ

AUTORIZOVANÝ SOFTWARE

M2BPM-3D

SOFTWARE FOR MODELING MOVEMENT OF BIOT-TYPE POROUS MEDIUM IN 3D CHAMBER

Autor: *Robert Cimrman*
Eduard Rohan

Číslo projektu: *N*

Číslo výsledku: *NTC-SW-05-10*

Odpovědný pracovník: *Ing. Robert Cimrman, Ph.D.*

Vedoucí odboru: *Doc. Dr. Ing. Eduard Rohan*

Ředitel centra: *doc. Dr. RNDr. Miroslav Holeček*

PLZEŇ, PROSINEC 2010

Jazyk výsledku: *EN*

Hlavní obor: *JR*

Uplatněn: *ANO*

Název výsledku česky:

Software pro modelování pohybu porézního média Biotova typu ve 3D komoře

Název výsledku anglicky:

Software for modeling movement of Biot-type porous medium in 3D chamber

Abstrakt k výsledku česky:

Software implementuje matematický model pohybu porézního média Biotova typu prostorem 3D komory s uvažováním Coulombova tření na stěnách, který může být použit pro modelování procesu lisování olejniny ve šnekovém lisu. Pro popis procházejícího média je použita kombinace Lagrangeova a Eulerova popisu, kde velké deformace v čase jsou řešeny pomocí sekvence subproblémů pro malý časový přírůstek v Lagrangeově formulaci s aktualizovanou deformovanou konfigurací. Tyto subproblémy mají tvar Biotova modelu rozšířeného o předpětí vzniklé v předchozích krocích a vztažené z posunuté, zdefrormované oblasti zpět na původni, nezdeformovanou (Eulerův popis). Coulombovské tření na stěnách je uvažováno pomocí nehladké rovnice zahrnující podmínky komplementarity třecích sil a multiplikátoru.

Abstrakt k výsledku anglicky:

The software implements a mathematical model of movement of a Biot-type porous medium through the space of a 3D chamber allowing for Coulomb friction conditions on the walls, that can be used for modeling the oil extruding-expelling process of oilseeds in the screw press. A combination of Lagrangian and Eulerian description is used to describe the passing medium, where large deformations in time are solved by a sequence of subproblems for a small time increment in the Lagrangian formulation with updated deformed configuration. Those subproblems have form of the Biot model extended by a prestress originating in preceding time steps and transformed from the shifted, deformed domain back to the original, undeformed domain (Eulerian description). The Coulomb friction on the walls is taken into account using the non-smooth equation involving the complementarity conditions of friction forces and a multiplier.

Klíčová slova česky:

porézní materiály; Coulombovské tření; šnekový lis

Klíčová slova anglicky:

porous materials; Coulomb friction; screw press extruder-expellers

Vlastník výsledku:	*Západočeská univerzita v Plzni*
IČ vlastníka výsledku:	*49777513*
Stát:	*Česká republika*
Lokalizace:	*http://www.zcu.cz/ntc/vysledky/sw/NTC-SW-06-10.html*
Licence:	*ANO*
Licenční poplatek:	*NE*
Ekonomické parametry:	*Software umožňuje pomocí výpočetních simulací lépe porozumět procesu oddělujícího lisování porézního média, a následně zvýšit efektivitu tohoto procesu v řadě navržených technologických zařízení. Větší ekonomický přínos souvisí např. s lisováním olejnin.*
Technické parametry:	*Luděk Hynčík, Západočeská univerzita v Plzni, Nové technologie - Výzkumné centrum v západočeském regionu, Univerzitní 8, 306 14 Plzeň, 377634709, hyncik@ntc.zcu.cz*

M2BPM-3D Documentation

Release 0.1

Robert Cimrman, Eduard Rohan

February 24, 2011

CONTENTS

M2BPM-3D is an application project for movement with friction of a Biot-type porous medium through a 3D chamber walls built upon SfePy.

SfePy documentation: http://docs.sfepy.org/doc-devel

CHAPTER

ONE

ABSTRACT

The software implements a mathematical model of movement of a Biot-type porous medium through the space of a 3D chamber allowing for Coulomb friction conditions on the walls, that can be used for modeling the oil extruding-expelling process of oilseeds in the screw press. A combination of Lagrangian and Eulerian description is used to describe the passing medium, where large deformations in time are solved by a sequence of subproblems for a small time increment in the Lagrangian formulation with updated deformed configuration. Those subproblems have form of the Biot model extended by a prestress originating in preceding time steps and transformed from the shifted, deformed domain back to the original, undeformed domain (Eulerian description). The Coulomb friction on the walls is taken into account using the non-smooth equation involving the complementarity conditions of friction forces and a multiplier.

CHAPTER

TWO

USAGE

2.1 Setting up environment

1. You need SfePy and all its dependencies.
2. Make sure that *SfePy* can be found by Python. If you build *SfePy* in place, set the *PYTHONPATH* environment variable: for example on Linux, using bash shell (*$* is the prompt character, do not type it!):

   ```
   $ export PYTHONPATH=$PYTHONPATH:<path-to-SfePy>
   ```

3. Try running the solver by *friction_slip.py*:

   ```
   $ ./friction_slip.py
   ```

   Usage: friction_slip.py [options] base_filename

   Options:

--version	show program's version number and exit
-h, --help	show this help message and exit
-o filename	output file name [default: friction_out.vtk]
-l, --lagrange	use Lagrange multipliers to enforce non-penetration

2.2 Running a simulation

For example, the small-scale linear elasticity file called *elasticity_small.py* can be used as the input for the solver as follows:

```
$ ./friction_slip.py elasticity_small.py
```

The results of the simulation are then stored in a custom HDF5-based file or a VTK file in the *output* directory, as specified in the example file.

2.3 Postprocessing

The results stored in VTK files can be viewed by any VTK-capable viewer, for example Mayavi or ParaView. *SfePy* comes with a simple automatic post-processor based on Mayavi, that works both for the VTK files and the HDF5 files:

```
$ <path-to-SfePy>/postproc.py <file.h5> -b
$ <path-to-SfePy>/postproc.py <file.vtk> -b
```

Contents:

2.3.1 common module

common.**incwd**(*filename*)
Prepend the directory of this file to filename.

2.3.2 cps module

2.3.3 elasticity module

elasticity.**define**(*non_penetration='lagrange'*)

elasticity.**get_pars**(*ts*, *coor*, *mode=None*, *output_dir='.'*)

2.3.4 elasticity_small module

elasticity_small.**define**(*non_penetration='lagrange'*)

2.3.5 friction_slip module

friction_slip.**create_base_problem**(*base_filename*, *is_lagrange*)

friction_slip.**main**()

2.3.6 friction_example module

friction_example.**define**()

friction_example.**get_pars**(*ts*, *coor*, *mode=None*, *output_dir='.'*)

2.3.7 regions module

regions.**coors_in_cylinder**(*coors*, *centre*, *axis*, *radius*, *length*, *inside=True*)
Select coordinates in a cylinder given by centre, axis and length.

regions.**define_regions**(*filename*)

regions.**in_out_0**(*x*, *y*, *z*, *mode*)

regions.**in_out_cylinder**(*coors*, *mode*)

2.3.8 friction.evaluators module

class friction.evaluators.**BaseFrictionEvaluator**(*base_pb*, *friction_region*, *var_names*, *quad_order=1*, ***kwargs*)

create_base_matrix()
Assume linear base problem.

create_equations()
The equations consist of the base problem equations and the friction terms corresponding to matrices B, C.

Additionally, the friction multiplier unknown and test variables are appended to the equations' variables.

create_friction_matrices()
Create the friction terms and assemble corresponding matrices B, B^T, and C.

create_friction_variables()

create_lcbc_operators()

create_normal_matrices()

create_normal_variables()

fun_b(*vec*)

fun_b_grad(*vec*)

get_dual_norm(*vec_g*)

get_dual_norm_derivative(*vec_g*, *norm_g*)

get_fun_a(*vec*)

get_fun_a_grad(*vec*)

get_fun_smooth(*vec*)

get_fun_smooth_grad(*vec*)

get_stripped_state_vector(*state*)

make_full_vec(*vec*)

class friction.evaluators.**LCBCFrictionEvaluator**(*base_pb*, *friction_region*, *var_names*, *quad_order=1*, ***kwargs*)

create_equations()

create_lcbc_operators()
Create non-penetration substitution for base problem variables.

fun_a(*vec*)

fun_a_grad(*vec*)

fun_smooth(*vec*)

fun_smooth_grad(*vec*)

get_stripped_state_vector(*state*)
Apply non-penetration operator to base problem vector part.

class friction.evaluators.**LagrangeFrictionEvaluator**(*base_pb*, *friction_region*, *var_names*, *quad_order=1*, ***kwargs*)

assemble_linear_matrix_part()
Assemble the linear part of the tangent matrix of the smooth function.

fun_a(*vec*)

fun_a_grad(*vec*)

fun_smooth(*vec*)

fun_smooth_grad(*vec*)

2.3.9 friction.friction_field module

class friction.friction_field.**FrictionField**(*name*, *dtype*, *shape*, *region*)

extend_dofs(*dofs*, *fill_value=None*)
Extend DOFs to the whole primary domain using the *fill_value*, or the smallest value in *dofs* if *fill_value* is None.

get_coor(*nods=None*, *igs=None*)

get_dofs_in_region(*region*, *merge=False*, *clean=False*, *warn=False*, *igs=None*)

get_dual_surface(*ig*)

get_output_approx_order()

igs()

remove_extra_dofs(*dofs*)
Remove DOFs defined in higher order nodes (order > 1) - does nothing here apart interpolation to primary mesh.

setup_dof_conns(*dof_conns*, *dpn*, *dc_type*, *region*)

setup_dual_mesh()

setup_extra_data(*geometry*, *info*, *is_trace*)

setup_global_base()

2.3.10 friction.terms_friction module

class friction.terms_friction.**DualSurfaceMassTerm**(*name*, *arg_str*, *integral*, *region*, ***kwargs*)

Arguments virtual : q, state : p

class friction.terms_friction.**FrictionProjectionTerm**(*name*, *arg_str*, *integral*, *region*, ***kwargs*)

Arguments virtual : $\underline{b}$, state : $\underline{g}$

class friction.terms_friction.**SurfaceNormalProjectionTerm**(*name*, *arg_str*, *integral*, *region*, ***kwargs*)

Arguments virtual : $\hat{v}$, state : $\hat{g}$

class friction.terms_friction.**SurfaceProjectionTerm**(*name*, *arg_str*, *integral*, *region*, ***kwargs*)

Arguments virtual : $\underline{v}$, state : $\underline{g}$

friction.terms_friction.**create_multiplier_matrix**(*vec*)

friction.terms_friction.**prepare_primary_bases**(*ap*, *qp_coors*, *ds*, *n_comp=1*)

Primary face base evaluated in dual quadrature points for each sub-triangle.

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

INDEX