FVD_FILTER Documentation

Vydání 1.0

Jan Ocenasek

12.04.2010

Obsah

FVD_FILTER je program v jazyce Python sloužící k numerické simulaci konsolidace saturovaných porézních materiálů s předpokadem velkých objemových změn.

KAPITOLA 1

Úvod

FVD_FILTER je programový nástroj sloužící k numerické simulaci konsolidace saturovaných porézních materiálů s předpokadem velkých objemových změn. Pro formulaci modelu je použit Eulerů popis a matematické vztahy vhodné pro popis velkých objemových deformací. Je předpokládána nestlačitelnost tekuté i pevné fáze, nelineární permeabilita. Difuze tekutiny je popsána Darcyho konstitutivním vztahem. Matematický popis modelu vede na soustavu jednodimenzionálních parciálních diferenciálních rovnic s nelineárními parametry na oblasti s pohyblivou hranicí. Soustava rovnic je proto transformována do konvektivních souřadnic a dále řešena pomocí Galerkinovy metody s explicitní integrací v čase. Software dále umožňuje volbu různých okrakových podmínek (Dirichletovy, Neumanovy).

Hlavní části programu jsou:

- `FVD_FILTER.py` ... řešič úlohy
- `SIMPLE_POST.py` ... program pro základní vizualizaci výsledků

Ostatní části programu, které nejsou samostatně spustitelné, jsou v těchto souborech:

assemble.py, *fortran_test.py*, *parametry.py*, *solvers.py*, *xls_export.py*

V adresáři (modulu) *fortran* je dále kód pro řešení soustavy rovnic napsaný ve fortranu a kompilovaný pomocí modulu F2py. Program *FVD_FILTER* se automaticky pokouší využít tento efektivní kompilovaný řešič. Pokud se to nepodaří, použije o něco pomalejší kód pythonu. Při přechodu na jinou platformu (operační systém) není zaručena kompatibilita kompilovaného kódu. V tom případě je třeba kód kompilovat znovu.

1.1 Teoretický úvod

Matematický model procesu konsolidace je popsán soustavou parciálních diferenciálních rovnic pro objemovou deformaci pevné fáze $\epsilon = \epsilon(t, \omega)$ a tlak intersticialni tekutiny $p_f = p_f(t, \omega)$

$$\begin{aligned} \eta \frac{\partial \epsilon}{\partial t} + \mathcal{F}(\epsilon) - \frac{\gamma}{\Phi(\epsilon)} p_f &= -P(t) + p_f \\ -\frac{\partial}{\partial \omega}\left(\frac{K(\Phi(\epsilon))}{\mu}(1-\Phi(\epsilon))\frac{\partial p_f}{\partial \omega}\right) + \frac{1+\gamma/\Phi(\epsilon)}{\eta(1-\Phi(\epsilon))} p_f &= \frac{1}{\eta}\frac{P(t)+\mathcal{F}(\epsilon)}{1-\Phi(\epsilon)}. \end{aligned} \tag{1.1}$$

Pro odvození a podrobný popis modelu viz výzkumná zpráva [i].

Při řešení je možné zvolit mezi dvěma typy okrajových podmínek.

1. Jednodužší podmínka (pro ideální filtr) uvažuje nulový tlak v místě síta $x = 0$, $p_f(t, 0) = 0$. Dále $\frac{\partial p_f}{\partial \omega}(t, (1 - \Phi_0)h_0) = 0$ a $\epsilon(0, \omega) = 0$.

2. Pro reálnou propustnost filtru Λ je podmínka $\frac{\partial p_f}{\partial \omega}(t,0) = \Lambda(p_f - p_{okoli})$. Dále stejně $\frac{\partial p_f}{\partial \omega}(t,(1-\Phi_0)h_0) = 0$ a $\epsilon(0,\omega) = 0$.

   Pozn.: Propustonost filtru Λ je v programu *FVD_FILTER* značena jako A, ve vstupním souboru jako *Alpha*.

Soustava je řešena Galerkinovo metodou s lineárním prvkem a explicitní integrací v čase. Pro časovou integraci je k dispozici Newtonova metoda 1. řádu a Runge-Kutta 4. řádu.

1.2 Instalace

Pro správnou funkci programu ja doporučena instalace Python verze 2.6.

Závislost na dalších modulech:

- numpy
- scipy
- re
- wx
- matplotlib
- pickle
- pyExcelerator (potřebný pouze pro export výsledků do .xls)

Instalaci pro Windows lze realizovat například instalací balíčku Enthought Python Distribution - EPD6, poté je ještě potřeba přidat modul *pyExcelerator*.

Modul *pyExcelerator* se přidává (po instalaci Pythonu) příkazem z instalačního adresáře modulu (viz také *readme.txt* tohoto modulu):

```
python setup.py install
```

Tutorial

2.1 FVD_FILTER

Program *FVD_FILTER.py* řeší popsaný problém, když jsou dány všechny materiálové a ostatní parametry úlohy vstupním souborem.

Spuštění programu se provede příkazem (z příkazové řádky):

```
python FVD_FILTER.py --input vstup.txt --output vystup.dat
```

nebo krátce:

```
python FVD_FILTER.py -ivstup.txt -ovystup.dat
```

bez parametrů (dafaultní vstupní a výstupní soubor):

```
python FVD_FILTER.py
```

je stejné jako:

```
python FVD_FILTER.py --input input.txt --output results.dat
```

Parametr `--input` zadává vstupní soubor, který obsahuje všechny parametry výpočtu.

Parametr `--output` určuje jméno standardního výstupního souboru.

2.1.1 Formát vstupního souboru

Vstupní soubor je textový soubor, který musí obsahovat všechny předepsané parametry. Seznam parametrů, jejich význam a formát souboru je zřejmý z následujícího příkladu:

```
@#
@#   Vstupni soubor pro program FVD_FILTER
@#
D=       4e-5     @# velikost elementu [m]
nel=     250      @# pocet prvku [1]
tsteps=  101      @# pocet casovych kroku [1]
```

```
tmax=    600.     @# max. cas [s]
Phi0=    0.5      @# pocatecni porosita [1]
mu=      55e-3    @# dynamicka viskozita tekute faze [Pa s]
gamma=   0.       @# parametr analogicky s Biotovym parametrem [1]
E=       117e6    @# objemovy modul pruznosti pevne faze [Pa]
eta=     1.8e9    @# viskozita pevne faze [Pa s]
alpha0=  3e10     @# parametr alpha0 konstitutivniho vztahu pro permeabilitu [m/kg]
C=       27.      @# exponent C konstitutivniho vztahu pro permeabilitu [1]
rho_s=   1052.    @# hustota skeletu [kg/m3]
Alpha=   1e-8     @# parametr sita [1/m]
pOkoli=  0.1e6    @# parametr sita [Pa]
P     =  30e6     @# zatizeni na pist [Pa]
method=  2        @# method=2 pro Runge-Kutta, jinak Newton
echo  =  1        @# echo=1 pro vypis prubehu vypoctu
xls   =  1        @# xls=1 pro vypis do MS-excel souboru
```

2.1.2 Popis parametrů

- **D** [m], **nel** [1]- Velikost elementu, počet elementů. Počáteční tlouška vzorku je tedy $h_0 = D * nel$.
- **tsteps** [1], **tmax** [s] - Počet časováých kroků, maximální čas. Určují dobu konsolidace a jemnost časových kroků.
- **Phi0** [1] - počáteční porosita
- **mu** [Pa s] - dynamická viskozita tekuté fáze
- **gamma** [1] - parametr analogický s Biotovým parametrem
- **E** [Pa] - objemový modul pružnosti pevné fáze (skletu)
- **eta** [Pa s] - viskozita pevné faze
- **alpha0** [m/kg], **C** [1] - Parametry konstitutivního vztahu pro permeabilitu. Parametr *alpha0* je počáteční odpor síta a *C* je exponent empirického vztahu Tillera a Yeha, viz [ii] rov. (7.7). Změnu konstitutivního vztahu lze provést v procedure *props* (assemble.py).
- **P** [Pa] - Zadává v čase konstatní zatížení na píst. Pokud bude třeba, v proceduře *p_load* (assembly.py) lze předepsat libovolnou časovou závislost tlaku na píst.
- **method** - Přepínač pro metodu integrace v čase. Hodnota `method = 2` pro Runge-Kutta čtvrtého řádu, jinak Newton prvního řádu. (Pro Newtonovu metodu je potřeba výrazně jemnější časový krok.)
- **echo** - Přepínač pro zobrazení průběhu výpočtu. Hodnota `echo = 1` pro výpis průběhu výpočtu, `echo = 0` bez výpisů.
- **xls** - přepínač pro export proměnných do MS-Excell tabulky. Hodnota `xls = 1` zapne export, `xls = 0` bez exportu. Pro tuto možnost je potřeba nainstalovat modul *pyExcelerator* viz. *Instalace*.

2.1.3 Výstup

Základním výstupem je binární soubor, který je možné prohlížet pomocí programu *SIMPLE_POST*. Obsahuje následující proměné

- **EPS** - objemová defromace [1]; `EPS(t,x)` - pole velikosti (tsteps+1, nel+1)
- **PHI** - porozita [1]; `PHI(t,x)` - pole velikosti (tsteps+1, nel+1)
- **PF** - efektivní tlak tekuté fáze [Pa]; `PF(t,x)` - pole velikosti (tsteps+1, nel+1)

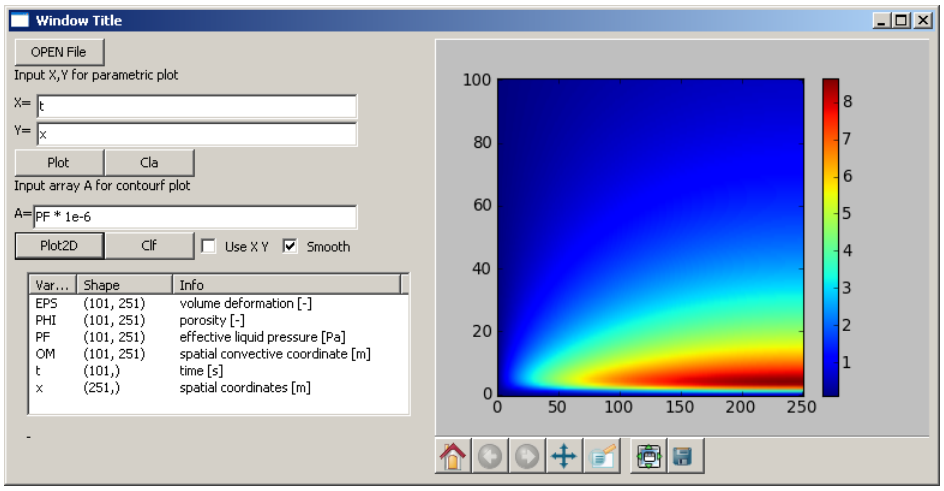
- **X** - souřadnice [m]; `X(t,x)` - pole velikosti (tsteps+1, nel+1)
- **t** - čas [s]; pole velikosti (tsteps+1, 1)
- **x** - souřadnice místa v nedeformované konfiguraci (pozice uzlů) [m]; pole velikosti (nel+1, 1)

2.2 SIMPLE_POST

Program *SIMPLE_POST* souží k jednoduchému prohlížení výstupního souboru s výsledky. Po spuštění příkazem:

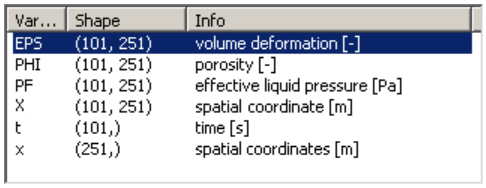
```
python SIMPLE_POST.py
```

se objeví toto okno s jednoduchým grafickým rozhraním



Obrázek 2.1: Grafické rozhraní *SIMPLE_POST.py*

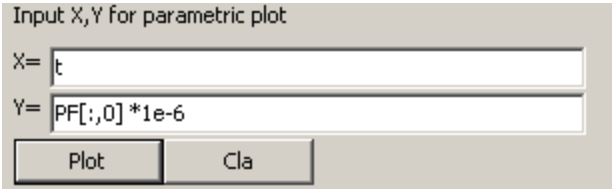
Dialog pro volbu datového souboru se objeví po kliknutí na `OPEN File`. Po úspěšném utevření souboru se objeví informace o načtených proměnných (jméno, rozměr pole, význam).



Obrázek 2.2: Informace o proměnných

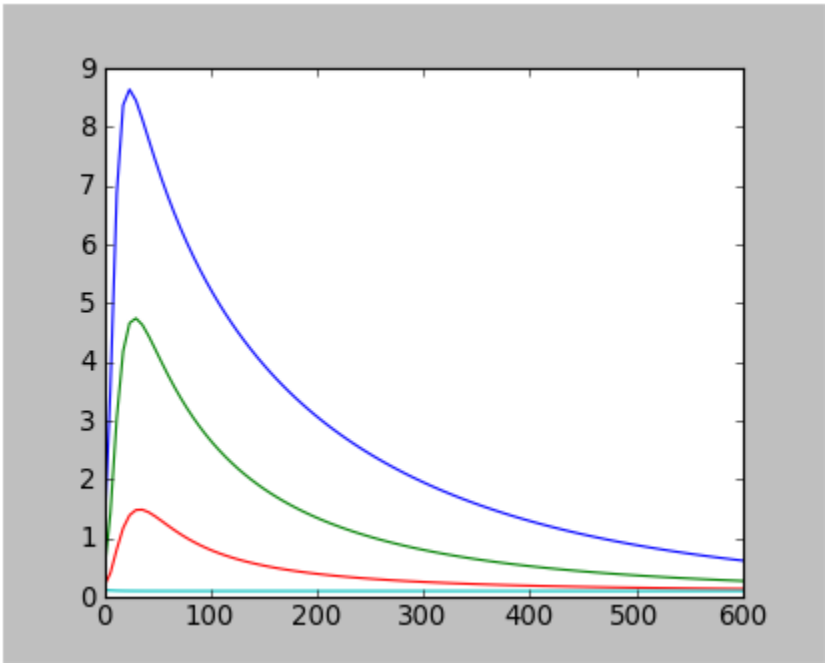
2.2.1 PLOT XY

Různé křivkové závislosti lze zobrazit pomocí tohoto formuláře:



Obrázek 2.3: Plot XY - Formulář pro zadání grafu

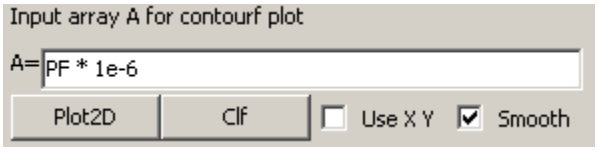
Syntaxe zadávání proměnných je shodná se syntaxí pythonu, viz Příklady. Je možné zobrazení více křivek. Tlačítka `Plot` a `cla` jsou pro zobrazení (přidání) křivky a smazání grafu.



Obrázek 2.4: Plot XY - Příklad grafu

2.2.2 PLOT 2D

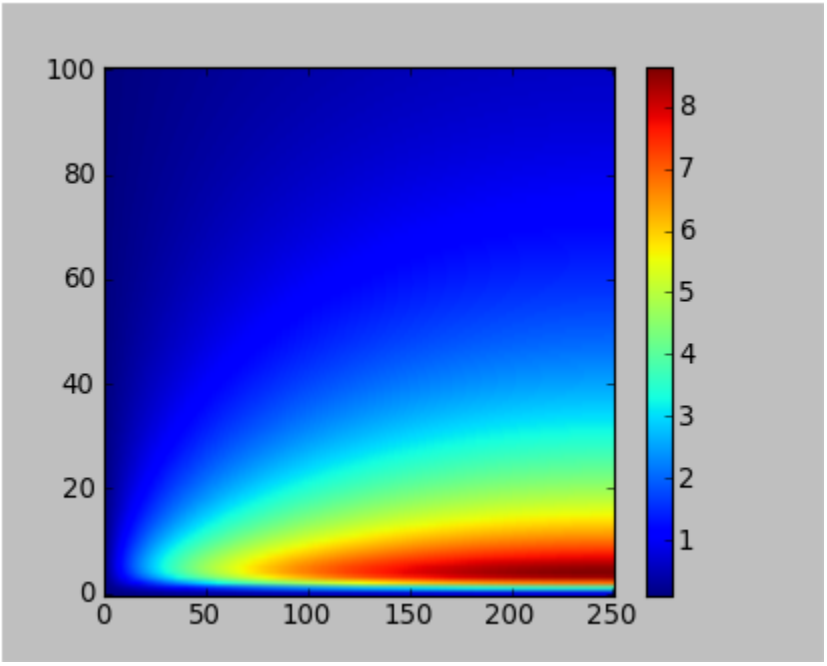
Pole, nebo část pole je možné zobrazit v barevných konturách pomocí tohoto formuláře:



Obrázek 2.5: Plot2D - Formulář pro zadání grafu

Tlačítka `Plot2D` a `cla` jsou pro zobrazení a smazání grafu. Možnost `Smooth` je pro plynulé přechody barev. Pokud není použitito volby `Use XY` je na osách zobrazen index pole. Pokud je použita volba `Use XY`, je potřeba zadat

do políček `X=` a `Y=` vektory odpovídající velikosti. Jejich hodnoty se zobrazí na osách. Pozor, zadání je poněkud neintuitivní: `X=` pro `první index` pole - tomu odpovídá svislá osa a `Y=` pro `druhý index` pole - tomu odpovídá vodorovná osa. V další verzi programu bude tento problém odstraněn.



Obrázek 2.6: Plot2D - Příklad grafu

2.2.3 Grafické nástroje

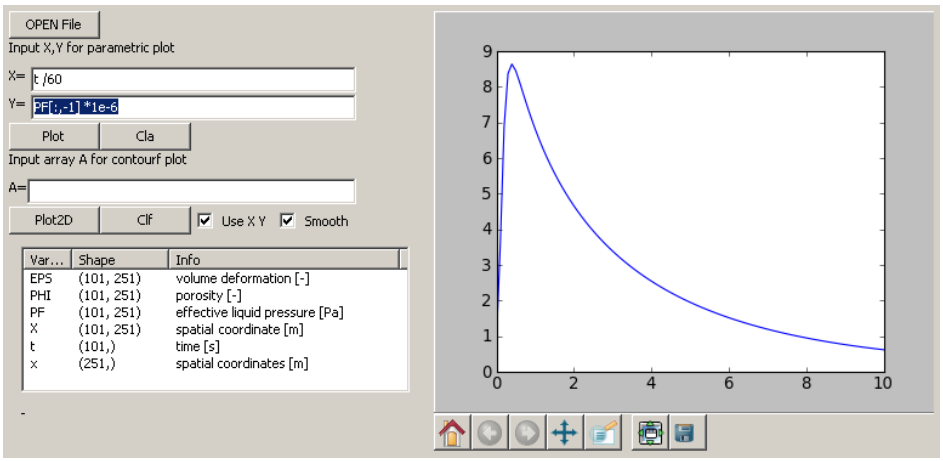
Pro zoom, a uložení grafů lze použít základní nástroje na liště pod grafem.

Obrázek 2.7: Grafické nástroje.

2.2.4 Příklady

1. Časový průběh (v minutách) tlaku tekutiny (v MPa) pod pístem.
   - `X = t /60.`
   - `Y = PF[:,-1] *1e-6`

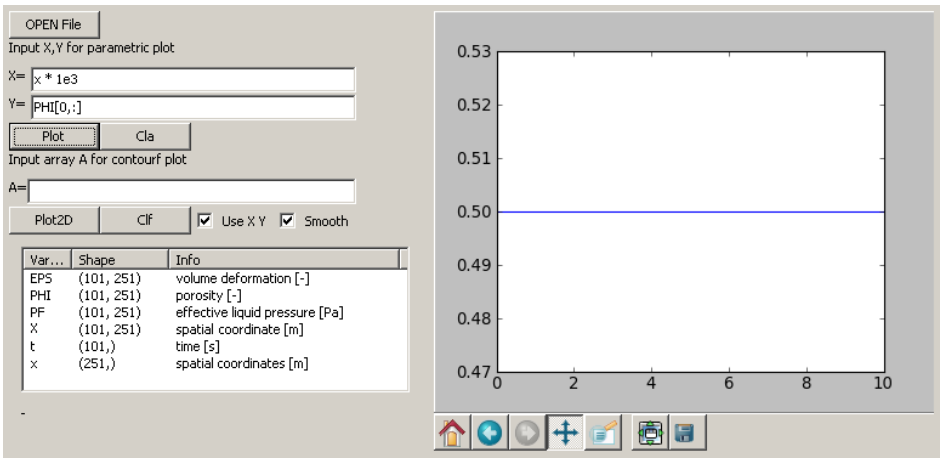
     Dvojtečka `:` značí všechny řádky pole. Index `-1` vybere poslední sloupec, ten odpovídá pozici pístu.
1. Průběh porozity (po délce koláče v mm) v t=0.
   - `X = x *1e3`



Obrázek 2.8: Příklad 1

- `Y = PHI[0,:]`

  Index `0` vybere první index (čas t=0). Dvojtečka `:` značí všechny řádky pole.



Obrázek 2.9: Příklad 2

1. Průběh porozity ve vzdálenosti od síta (mm) v t=tmax (na konci simulace). Je zobrazena vzdálenost v nedeformované konfiguraci.

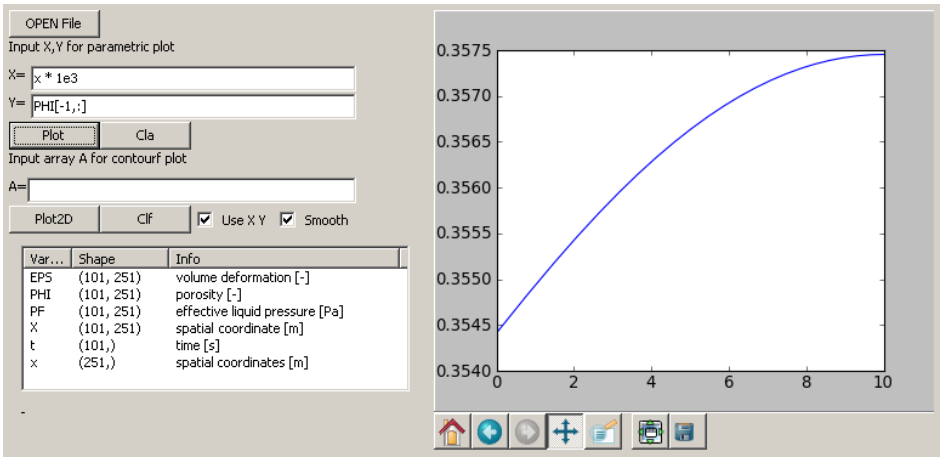
   - `X = x *1e3`
   - `Y = PHI[-1,:]`

     Index `-1` vybere poslední sloupec, ten odpovídá poslednímu indexu. Dvojtečka `:` značí všechny sloupce pole.

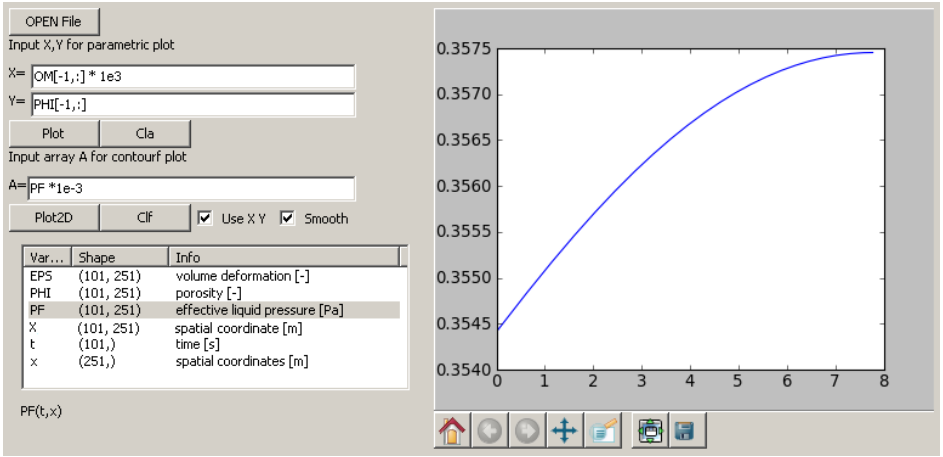
1. Průběh porozity ve vzdálenosti od síta (mm) v t=tmax (na konci simulace). Je zobrazena vzdálenost v aktuální (deformované) konfiguraci.

   - `X = x *1e3`
   - `Y = PHI[-1,:]`

     Index `-1` vybere poslední sloupec, ten odpovídá poslednímu indexu. Dvojtečka `:` značí všechny sloupce pole.

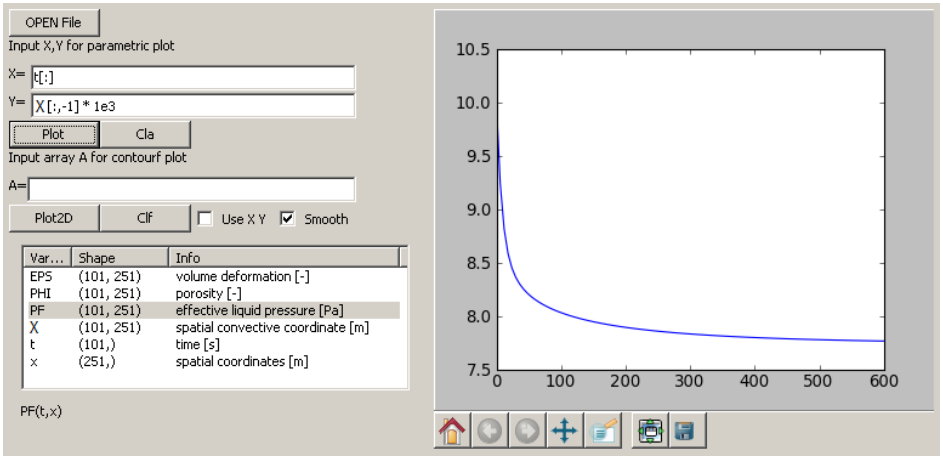


Obrázek 2.10: Příklad 3



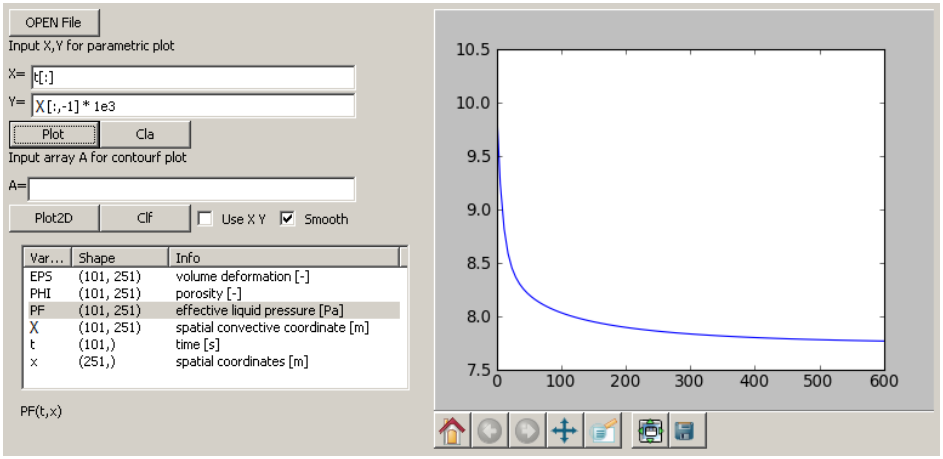
Obrázek 2.11: Příklad 4

1. Pozice pístu (mm) v čase (s).
   - `X = t`
   - `Y = X[:,-1] * 1e3`



Obrázek 2.12: Příklad 5

1. Průběh tlaku tekutiny. Na svislé ose je čas v [s], na vodorovné vzálenost od síta v nedeformované konfiguraci (mm).
   - `X = t`
   - `Y = x * 1e3`
   - `A = PF * 1e-3`



Obrázek 2.13: Příklad 6

Dokumentace

3.1 assemble.py

Rside(*C, D*)
Sestaveni prave strany

Parametry

- *C* – vektor hodnot C v uzlech
- *D* – delka mezi uzly

Vrací Q vektor prave strany

bands_k(*Ae, D*)
Sestaveni matice K

Parametry

- *Ae* – vektor hodnot A pro elementy o rozmeru (N,)
- *D* – delka mezi uzly

Vrací Kd, Kv

Proměnná Kd hodnoty na hlavni diagonale

Proměnná Kv vedlejsi diagonala

bands_m(*Be, D*)
Sestaveni matice M

Parametry

- *Be* – vektor hodnot A pro elementy o rozmeru (N,)
- *D* – delka mezi uzly

Vrací Kd, Kv

Proměnná Md hodnoty na hlavni diagonale

Proměnná Mv vedlejsi diagonala

epsdot(*parms, eps, pf, t*)
Casova derivace objemove deformace

Parametry

- *eps* – objemova deformace
- *pf* – tlak tekutiny
- *t* – cas

Vrací edot o stejnem rozmeru jako *eps* a *pf*

p_load(*parms, t*)

Definuje zatizeni na pist. (Zatizeni lze zadat zavisle na case t)

Parameter *parms* – parametry ulohy

Vrací P zatizeni na pist [Pa]

props(*parms, eps*)

Tato procedura vytvori vertory parametru odpovidajici deformovane kofiguraci dane vektorem *eps*. Tj. tyto veliciny:

•zakladni materialove parametry

•hodnoty permeability podle empirickeho modelu Tiller-Yeh, tj. podle [i] rovnice (7.2) a (7.7)

•vztah mezi zatizenim pevne faze a jeho deformaci (generalize Voigt) podle [i] rov. (5.5)

•**parametry sita *Alpha* a *pOkoli***

- pro hodnotu *Alpha*=0 je na situ nastavena Dirichletova podminka $p_f(t, x = 0) = 0$
- pro hodnotu Alpha*>0 je na situ nastaven tok $Flux = \Lambda * (p_f - p_{Okoli})$

Vstup Parametr *eps* (objemova deformace) jako radek delky N

Vystup Vraci nasledujici parametry (radky stejne delky N)

- Phi0 [1] ... pocatecni poresita
- mu [Pa s] ... dynamicka viskozita tekute faze
- gamma [1] ... parametr analogicky s Biotovym parametrem
- eta [Pa s] ... viskozita pevne faze
- E [Pa] ... objemovy modul pruznosti pevne faze
- Ee [Pa] ... viz clen F(e) rovnice (9) v clanku [1]
- eta [Pa s] ... viskozita pevne faze
- Alpha [?] ... parametr sita
- pOkoli [Pa] ... parametr sita

3.2 solvers.py

pde_solver(*parms*)

Hlavni resic soustavy pde

Parameter *parms* – obsahuje vsechny parametry ulohy (viz class parametry.Parametry)

Vraci

- **EPS** - objemova deformace

- **PHI** - porozita
- **PF** - tlak tekutiny
- **X** - materialova souradnice

Vsechny vystupni promenne jsou pole o rozmerech tsteps+1, nel+1. Tj. obsahuji hodnoty v mistech prvkovych uzlu a vsech casovych hladinach vcene t=0.

pf_solver(*parms, eps, t, D*)
Resic pro rozlozeni tlaku tekute faze

Parametry

- *eps* – objemova deformace
- *t* – cas

Param D velikost elementu

Vrací pf plak v [Pa]

Okrajove podminky jsou nastaveny v zavislosti na parametru Alpha.

t_diag_sym(*d, v, b, n*)
Resi soustavu A x = b s tridiagonalni symetrickou matici A a pravou stranou R.

Parametry

- *d* – diagonala A
- *v* – vedlejsi diagonala A
- *b* – prava strana

Vrací x reseni soustavy

3.3 fortran_test.py

f_test()
Tato procedura testuje funkcnost rychleho resice soustavy rovnic napsaneho ve fortranu. Vrati hodnotu 1, pokud je vse ok, jinak 0.

3.4 parametry.py

class Parametry(*parms_dic*)
This class is used to hold, read from file and save to file parameters given in dictionary parms_dic.

Example:

```
p_dic={'Time': 10.0, 'Method': 'lienar'}
p= Parametry(p_dic)
p.write('data.txt')
```

This would creat a file *data.txt* containing two lines

```
Time = 10.0 Method = 'linear'
```

To read it back define a dictionary defining parameter names and reload it:

```
p_dic2={'Time':None, 'Method':None}
p2= Parametry(p_dic2)
p2.read('data.txt')
```

To get a parameter value 'Time':

```
t = p.get('Time')
```

Comments in the file:

```
@# Any line starting with @# is ignored
Time = 10.0 @# This would also work
```

Flags

Variable *self.parms_flag* is a dictionary holding the same keys like *self.parms_dic*. Values are Flase or True. Line of the inputfile starting with asterix will set the flag to True:

```
*Time = 10.0 # Then  .get_flag('Time') returnss True
```

clear_values()
Set all values to None.

get(*key*)

get_empty()
Report all keys with None values.

get_flag(*key*)

get_marked()
List of parameters values "marked" with asterix

get_marked_names()
List of parameters names "marked" with asterix

read(*filename*)
Read values form file

set_marked(*x0*)
List of parameters values "marked" with asterix

write(*filename*)
Write values to file

3.5 xls_export.py

class XLSfile()
This is a simple class to export numpy.ndarrays in MS-Excel file. The array must be vector or an matrix. Each vector is placed in separate list. Some information given by *info_list* is placed at firts lines.

add_array(*sheet_name, array, info_list*)
This will add an *array* to a new list called *sheet_name* (Sheet names can not repeat within one file). You can add a list *info_list* giving information about the exported variable.

save(*filename*)
To wrtite the xls file.

3.6 fortran.t_diag_sym_f

t_diag_sym_f(*d, v, b, n*)

This module 't_diag_sym_f' was auto-generated with f2py (version:2) from the FORTRAN subroutine *t_diag_sym_f.for*. The code fuction is equal to *solvers.t_diag_sym(d, v, b, n)*, i.e. solving eq. A x = b.

Parametry

- *d* – main diagonal of A
- *v* – upper diagonal (equal to lower diag.)
- *b* – right hand side
- *n* – dimension of A

Vrací x solution vector x (length n)

Indices and tables

- *Index*
- *Rejstřík modulů*
- *Vyhledávací stránka*

Literatura

[i] Voldřich J., Výzkumná zpráva NTC ZČU v Plzni, č. NTC 01-05/09, 2009.

Index