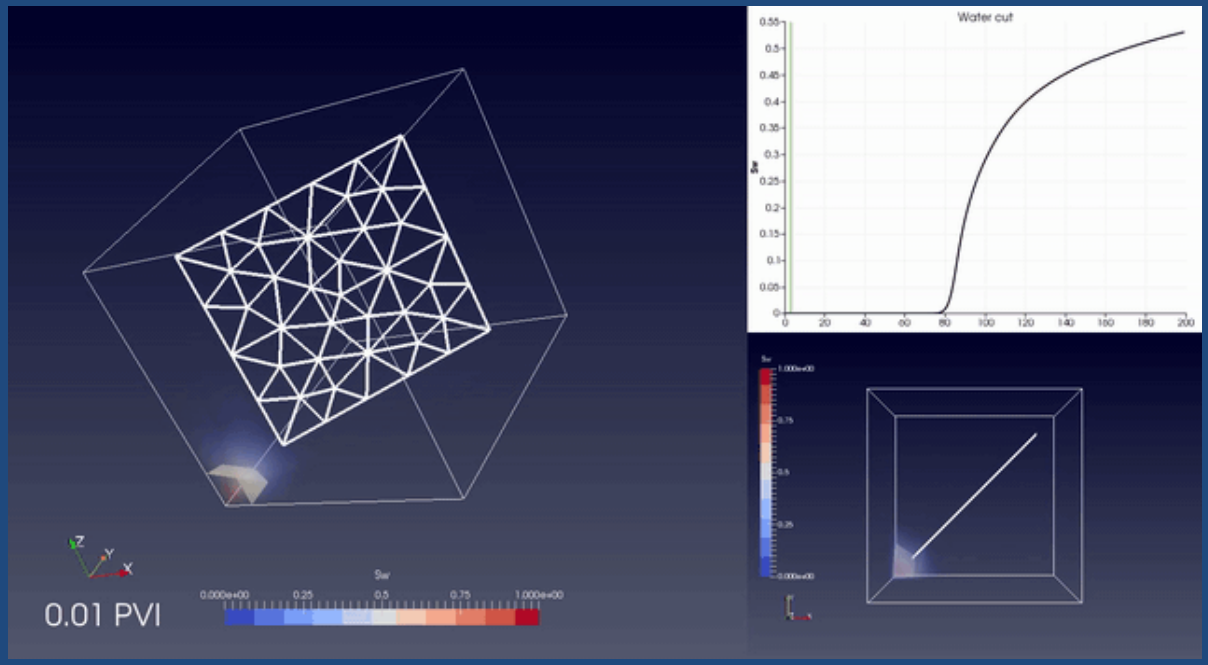
Séminaire du réseau Données et Numérique

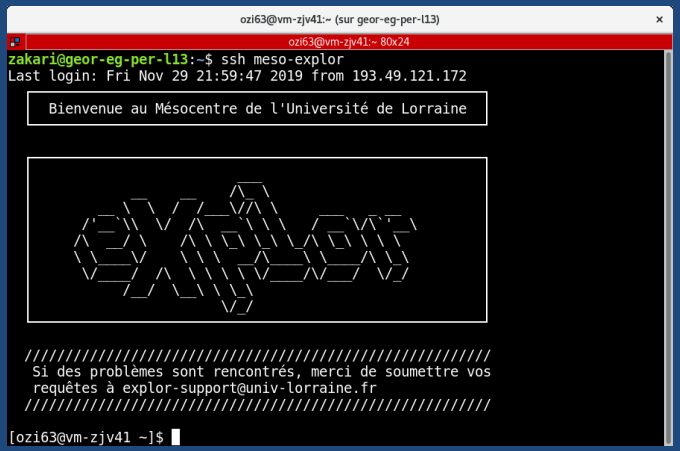


Mustapha Zakari

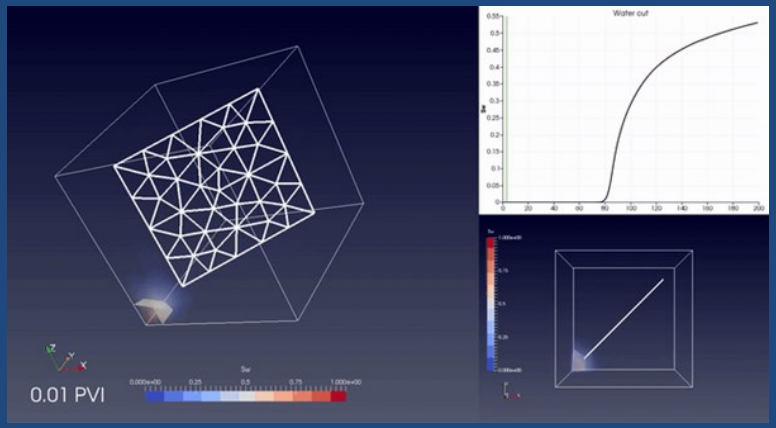
(3/12/2019)

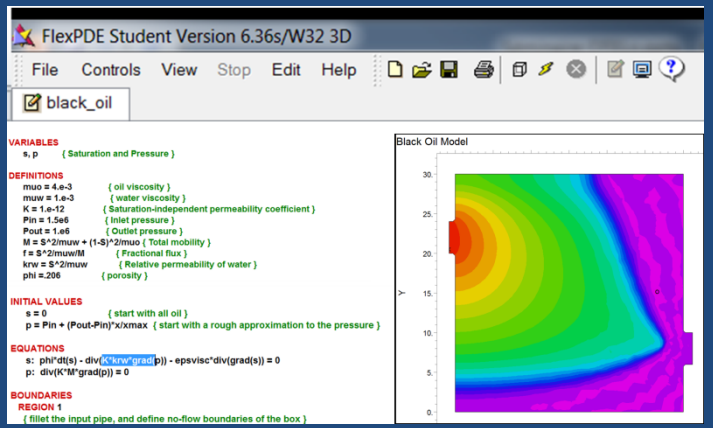
Plan

- EXPLOR – Accélération de codes de calcul



- Modélisation d'écoulements





EXPLOR
Accélération de codes de calcul

- EXPLOR : 6356 coeurs
  - Système d'exploitation : CentOS 7.4
  - Ordonnanceur : SLURM 17.11.3
  - Gestionnaire Environnement : LMOD 7.4
- Parallélisation du calcul approché de π (Fortran 90 +MPI)

EXPLOR – Calcul parallèle de π
Code Fortran + MPI

ozi63@vm-zjv41:~/tests/mpif90 (sur geor-eg-per-l13)

ozi63@vm-zjv41:~/tests/mpif90 97x28

```
program pi
  use mpi
  implicit none
  integer :: code,rang, nbprocs
  integer, parameter :: dp = kind(1.d0)
  integer, parameter :: li = selected_int_kind(15)
  integer(kind=li):: nbbloc,i,debut,fin
  real(kind=dp) :: largeur,somme,global,x

  call MPI_INIT(code)
  call MPI_COMM_RANK(MPI_COMM_WORLD,rang,code)
  call MPI_COMM_SIZE(MPI_COMM_WORLD,nbprocs,code)
  somme = 0._dp
  ! Nombre d'intervalles
  nbbloc = 3*1000*1000_li*100
  ! largeur des intervalles
  largeur = 1._dp / real(nbbloc,dp)
  debut = (rang*nbbloc)/nbprocs+1;  fin = ((rang+1)*nbbloc)/nbprocs
  print "(i2,a,i11,a,i11,a,i11)", rang, " debut: ", debut, " fin: ", fin, " delta: ", fin-debut+1
  do i=debut, fin
    x = largeur*(i-0.5_dp)
    somme = somme + largeur*(4._dp / (1._dp + x*x))
  end do
  call MPI_REDUCE(somme, global, 1, MPI_DOUBLE_PRECISION, MPI_SUM, 0, MPI_COMM_WORLD,code)
  if (rang ==0) print *, "Pi =", global
  call MPI_FINALIZE(code)
end program
```

Méthode des rectangles

$$\pi = \int_0^1 \frac{4}{1+x^2} dx \approx \sum_{i=1}^{n} \frac{4}{1+\left(\frac{i-0.5}{n}\right)^2}$$

EXPLOR – Calcul parallèle de π Soumission + résultats

Fortran90 + MPI

ozi63@vm-zjv41:~/tests/mpif90 (sur geor-eg-per-l13)

ozi63@vm-zjv41:~/tests/mpif90 88x24

```
[ozi63@vm-zjv41 mpif90]$ sbatch -N 1 -n 2 script2submit.sh
Submitted batch job 161986
[ozi63@vm-zjv41 mpif90]$ tail submitted/test_161986.out
 0 debut:           1 fin:   150000000 delta:   150000000
 1 debut:   150000001 fin:   300000000 delta:   150000000
 Pi =   3.1415926535893495

real    0m2.845s
user    0m5.110s
sys     0m0.042s
[ozi63@vm-zjv41 mpif90]$
```

1 nœud, 2 processeurs

```
[ozi63@vm-zjv41 mpif90]$ sbatch -N 1 -n 4 script2submit.sh
Submitted batch job 161987
[ozi63@vm-zjv41 mpif90]$ tail submitted/test_161987.out
 0 debut:           1 fin:    75000000 delta:    75000000
 1 debut:    75000001 fin:   150000000 delta:    75000000
 3 debut:   225000001 fin:   300000000 delta:    75000000
 2 debut:   150000001 fin:   225000000 delta:    75000000
 Pi =   3.1415926535895338

real    0m1.454s
user    0m5.137s
sys     0m0.067s
[ozi63@vm-zjv41 mpif90]$
```

1 nœud, 4 processeurs

EXPLOR – Exemples d'utilisation

- Le mésocentre : 6356 coeurs
  - Système d'exploitation : CentOS 7.4
  - Ordonnanceur : SLURM 17.11.3
  - Gestionnaire Environnement : LMOD 7.4

✓ Parallélisation du calcul approché de π (Fortran 90 +MPI)

- Code fortran
- Code MPI
- Link + compilation : [2procs:2,854s , 4procs:1,454s]

- Peut-on paralléliser des codes Python, Matlab ?
  - Parallélisation (Multithreading) du calcul approché de π :
    - Python + Multiprocessing
    - Matlab + PARallel FOR loops (parfor)

EXPLOR – Calcul parallèle de π
Code Python 3.6 + Multiprocessing

ozi63@vm-zjv41:~/tests/python (sur geor-eg-per-l13)

ozi63@vm-zjv41:~/tests/python 86x26

```python
import multiprocessing as mp
nprocs = mp.cpu_count()
print(f"Number of CPU cores: {nprocs}")

nsteps = 10000000
dx = 1.0 / nsteps
pi = 0.0
```

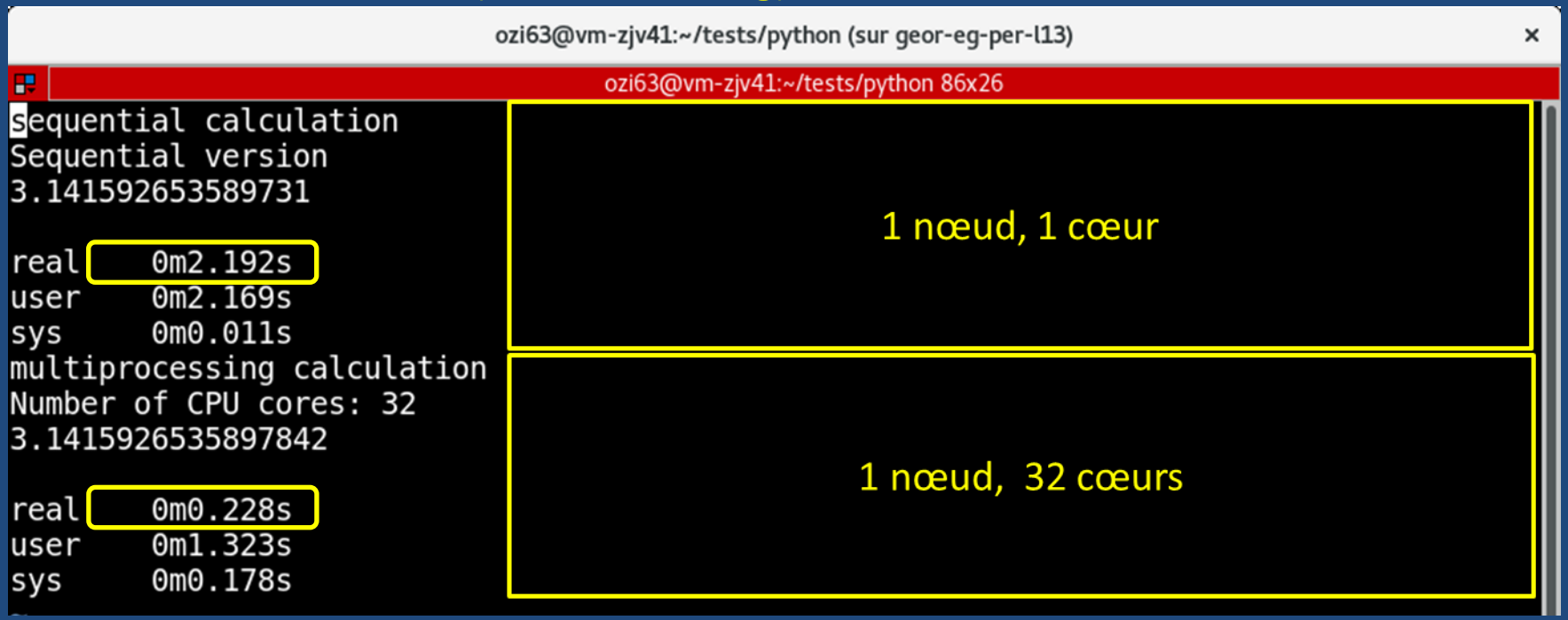
Méthode des rectangles

$$\pi = \int_0^1 \frac{4}{1+x^2} dx \approx \sum_{i=0}^{n} \frac{4}{1 + \left(\frac{i+0.5}{n}\right)^2}$$

```python
def calc_partial_pi(rank, nprocs, nsteps, dx):
    partial_pi = 0.0
    for i in range(rank, nsteps, nprocs):
        x = (i + 0.5) * dx
        partial_pi += 4.0 / (1.0 + x * x)
    partial_pi *= dx
    return partial_pi

inputs = [(rank, nprocs, nsteps, dx) for rank in range(nprocs)]

#Create pool object !STARTS A PARALLEL ZONE!
pool = mp.Pool(processes=nprocs)
result = pool.starmap(calc_partial_pi, inputs)
pi = sum(result)
```

EXPLOR – Calcul parallèle de π Résultats

Python + Multiprocessing (Multithreading)

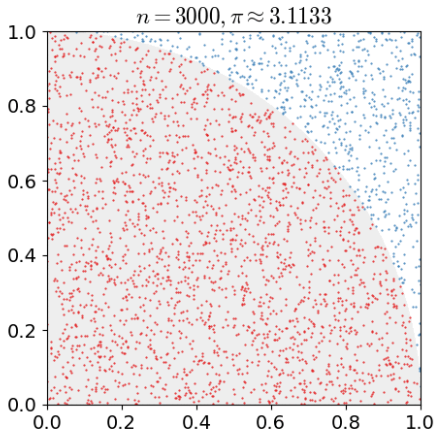


EXPLOR – Calcul parallèle de π
Code Matlab + Parfor

Matlab + Parfor — ozi63@vm-zjv41:~/tests/matlab (sur geor-eg-per-l13)

ozi63@vm-zjv41:~/tests/matlab 86x28

```
% Parallel Monte Carlo calculation of PI
%=========================================================================
parpool('local', str2num(getenv('SLURM_CPUS_ON_NODE')))
R = 1;
darts = 1e7;
count = 0;
tic
parfor i = 1:darts
    % Compute the X and Y coordinates of where the dart hit the
    % square using Uniform distribution.....................
    x = R*rand(1);
    y = R*rand(1);
    if x^2 + y^2 <= R^2
        % Increment the count of darts that fell inside of the.....
        % circle.....................................................
        count = count + 1; % Count is a reduction variable.
    end
end
% Compute pi.....................................................
myPI = 4*count/darts;
T = toc;
fprintf('The computed value of pi is %8.7f.n',myPI);
fprintf('The parallel Monte-Carlo method is executed in %8.2f seconds.n', T);
delete(gcp);
```



Méthode Monte-Carlo

$$\pi = 4\frac{counts}{darts}$$

Wikipedia (30 000) darts)

EXPLOR – Calcul parallèle de π Résultats

Matlab + Parfor

ozi63@vm-zjv41:~/tests/matlab (sur geor-eg-per-l13)

ozi63@vm-zjv41:~/tests/matlab 86x28

```
                    < M A T L A B (R) >
          Copyright 1984-2017 The MathWorks, Inc.
           R2017b (9.3.0.713579) 64-bit (glnxa64)
                      September 14, 2017

connected to 1 workers.                                  1 nœud, 1 cœur

The computed value of pi is 3.1417856.nThe parallel Monte-Carlo method is executed in
   14.88 seconds.nParallel pool using the 'local' profile is shutting down.
  message with properties:


                    < M A T L A B (R) >
          Copyright 1984-2017 The MathWorks, Inc.
           R2017b (9.3.0.713579) 64-bit (glnxa64)
                      September 14, 2017

connected to 32 workers.                                 1 nœud,  32 cœurs

The computed value of pi is 3.1414208.nThe parallel Monte-Carlo method is executed in
   0.92 seconds.nParallel pool using the 'local' profile is shutting down.
  message with properties:
```

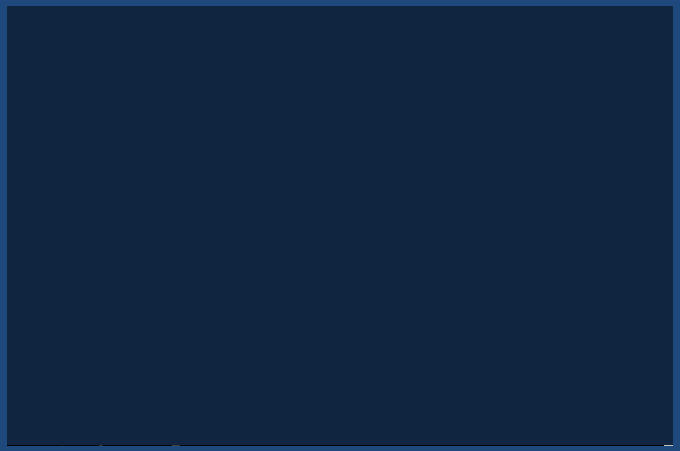
EXPLOR – Exemples d'utilisation Bilan

- ✓ Parallélisation du calcul approché de π (Fortran 90 +MPI)
  - Code fortran
  - Code MPI
  - Link + compilation : [2procs:2,854s ; facteur 1.96 ; 4procs:1,454s]
- ✓ Peut-on paralléliser des codes Python, Matlab ? OUI
  - Parallélisation (Multithreading) du calcul approché de π :
    - ✓ Python + Multiprocessing : [1c:2,192s ; facteur 9.61 ; 32c:0,228s]
    - ✓ Matlab + PARallel FOR loops (parfor) : [1c:14,88s ; facteur 16.2 ; 32c:0,92s]

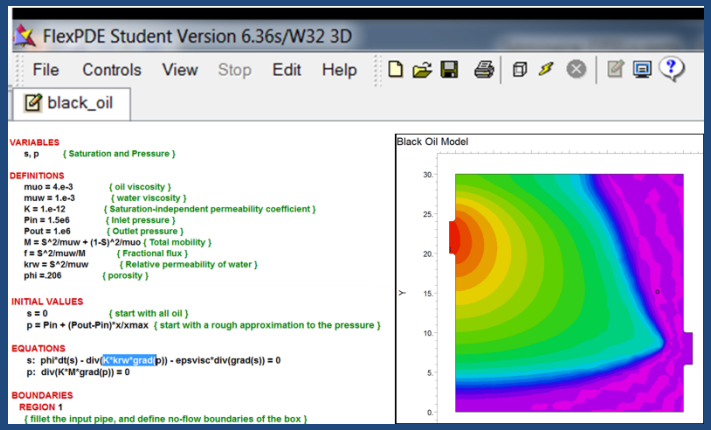
EXPLOR permet d'accélérer des codes Python et/ou Matlab

Plan

- EXPLOR – Exemples d'utilisation

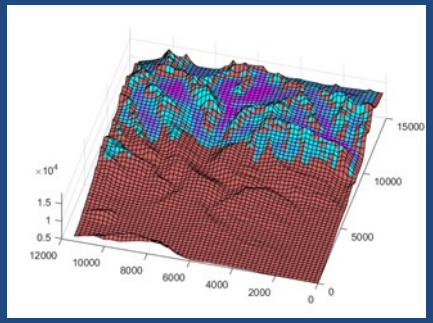


- Modélisation d'écoulements et extrapolations

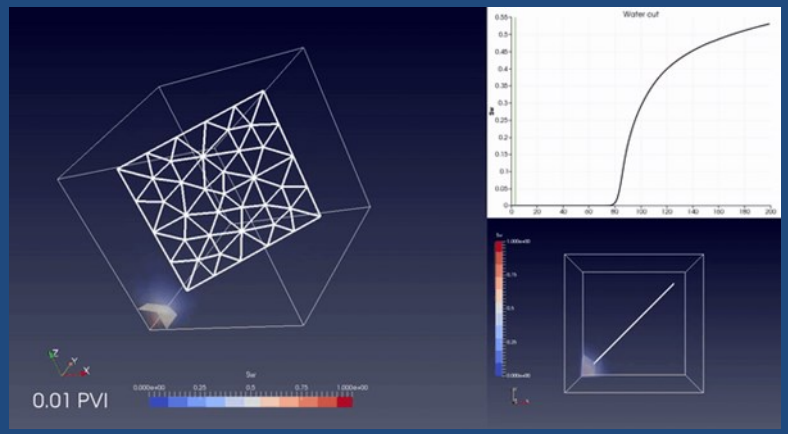


Modélisation d'écoulements
Extrapolations

- Ecoulements glaciaires
  - Equation du modèle
  - Discrétisation
  - Premiers Résultats



- Ecoulements diphasiques en milieux poreux fracturés
  - Equations du modèle
  - Discrétisation
  - Premiers Résultats



Ecoulements glaciaires

- **Equation du modèle** : Shallow Ice Approximation [Hutter 1983]
  - Loi de conservation + loi visqueuse de Glen

$$\begin{cases} \frac{\partial H}{\partial t} - \nabla \cdot \mathbf{q} = a \\ \mathbf{q} = D \nabla s \\ D = C H^{n+2} \left| \nabla s \right|^{n-1} \\ s = b + H \end{cases}$$

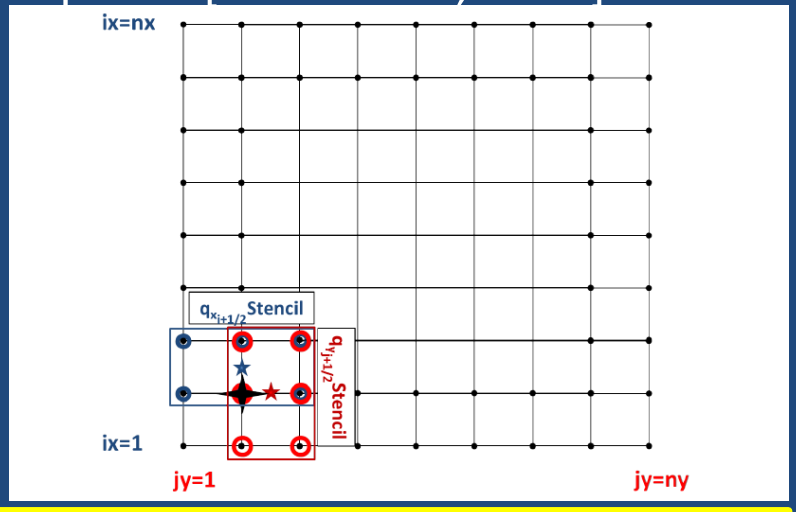
- Avec :
  - H l'épaisseur de glace
  - b et s le bedrock et la hauteur totale
  - a le terme d'accumulation/ablation
  - q le flux de glace et D le coefficient de diffusion
  - C une constante fixant la loi d'écoulement

Ecoulements glaciaires

– **Discrétisation** : **Differences Finies** + Explicite [Hindmarsh, 1996]

$$\begin{cases} \dfrac{\partial H}{\partial t} - \nabla \cdot \mathbf{q} &= a \\ \mathbf{q} &= D \nabla s \\ D &= C H^{n+2} \left|\nabla s\right|^{n-1} \\ s &= b + H \end{cases}$$

$$H_{i,j}^{k+1} = H_{i,j}^{k} + \frac{\Delta t}{\Delta x}\left(q_{x_{i+\frac{1}{2}}}^{k} - q_{x_{i-\frac{1}{2}}}^{k} + q_{y_{j+\frac{1}{2}}}^{k} - q_{y_{j-\frac{1}{2}}}^{k}\right) + \Delta t \times a_{i,j}^{k}$$

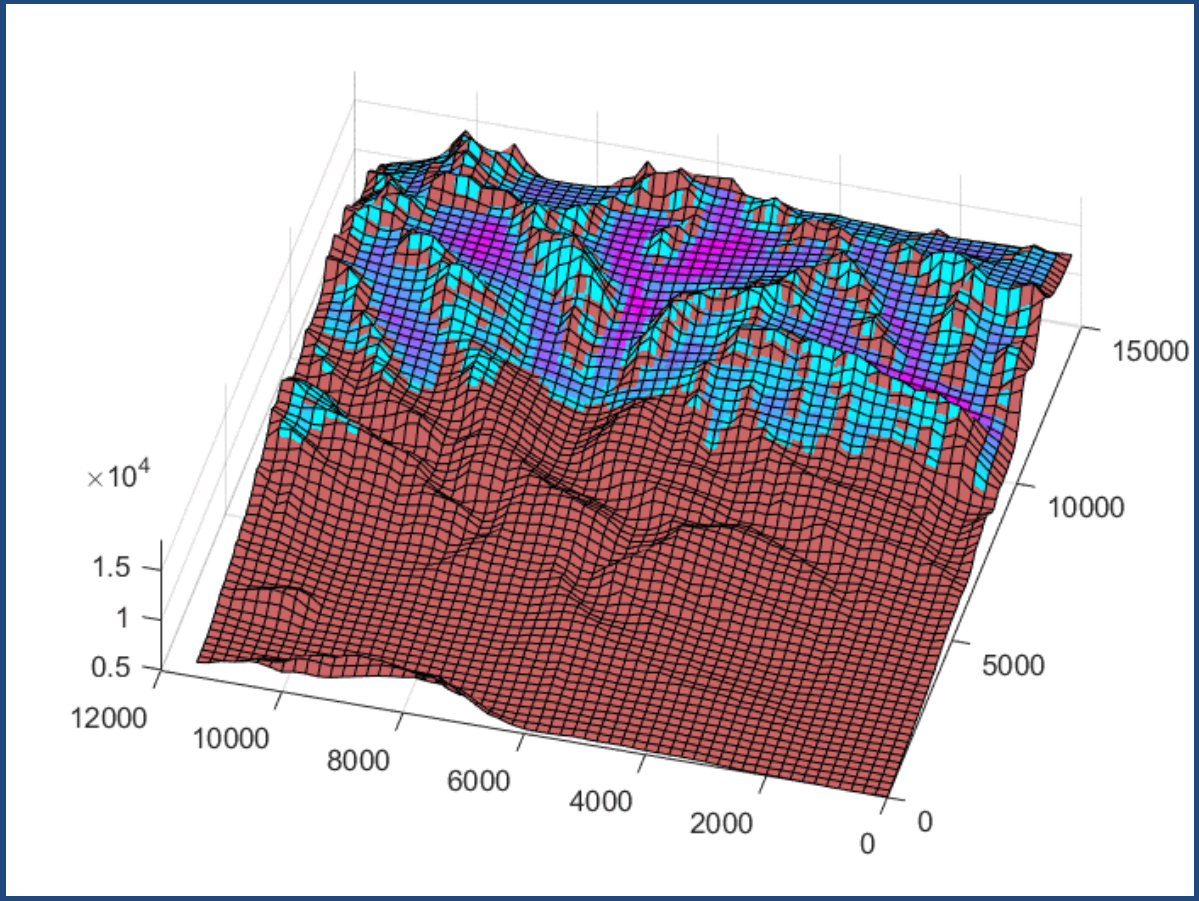


$$\begin{cases} q_{x_{i+\frac{1}{2}}} &= C_{i,j}\left[\frac{H_i+H_{i+1}}{2}\right]^5 \left(\left(\frac{H_{i+1}-H_i}{\Delta x}\right)^2 + \left(\frac{H_{i,j+1}-H_{i,j-1}+H_{i+1,j+1}-H_{i+1,j-1}}{4\Delta y}\right)^2\right)\left(\frac{H_{i+1}-H_i}{\Delta x}\right) \\ q_{x_{i-\frac{1}{2}}} &= C_{i,j}\left[\frac{H_{i-1}+H_i}{2}\right]^5 \left(\left(\frac{H_i-H_{i-1}}{\Delta x}\right)^2 + \left(\frac{H_{i-1,j+1}-H_{i-1,j-1}+H_{i,j+1}-H_{i,j-1}}{4\Delta y}\right)^2\right)\left(\frac{H_i-H_{i-1}}{\Delta x}\right) \end{cases}$$

$$\begin{cases} q_{y_{j+\frac{1}{2}}} &= C_{i,j}\left[\frac{H_j+H_{j+1}}{2}\right]^5 \left(\left(\frac{H_{j+1}-H_j}{\Delta y}\right)^2 + \left(\frac{H_{i+1,j}-H_{i-1,j}+H_{i+1,j+1}-H_{i-1,j+1}}{4\Delta x}\right)^2\right)\left(\frac{H_{j+1}-H_j}{\Delta y}\right) \\ q_{y_{j-\frac{1}{2}}} &= C_{i,j}\left[\frac{H_{j-1}+H_j}{2}\right]^5 \left(\left(\frac{H_j-H_{j-1}}{\Delta y}\right)^2 + \left(\frac{H_{i+1,j-1}-H_{i-1,j-1}+H_{i+1,j}-H_{i-1,j}}{4\Delta x}\right)^2\right)\left(\frac{H_j-H_{j-1}}{\Delta y}\right) \end{cases}$$

Modélisation d'écoulements glaciaires

– **Simulation** Matlab



Hauteur de glace sur 1000 ans, : [Lavé,Zakari CRPG]

Ecoulements diphasiques en milieux poreux fracturés

- **Equations du modèle** : Darcy + lois de conservations
  - Pression:

$$\nabla.(-\lambda_T \nabla P) = \frac{q_w}{\rho_w} + \frac{q_o}{\rho_o} \qquad \lambda_T = \lambda_w + \lambda_o$$

  - Saturation en eau

$$\Phi \frac{\partial S_w}{\partial t} + \nabla.\left(\frac{\lambda_W}{\lambda_T} \nabla P\right) = \frac{q_w}{\rho_w}$$

- Avec :
  - w l'eau et o l'huile
  - S_w la saturation en eau
  - Φ La porosité
  - P la pression
  - λ_T la mobilité totale
  - ρ la masse volumique
  - q les termes d'injection/aspiration

Ecoulements diphasiques en milieux poreux fracturés

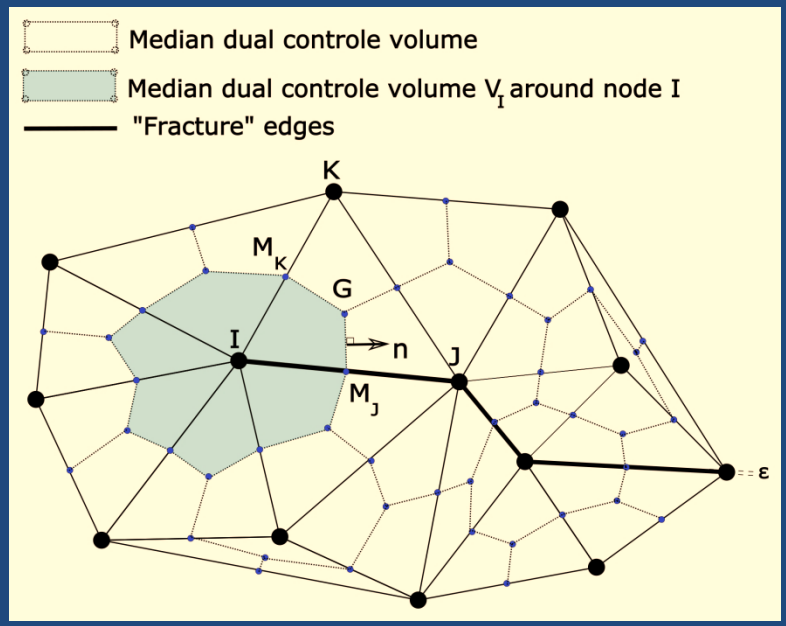
– **Discrétisation** : **Volumes-Finis** centrés aux nœuds, Explicite en

$$\sum_{IJK\epsilon\Omega_m}\left[-|M_JG|\,\lambda_T(S_{w,M_JG})\sum_{j=1}^{3}P_j\nabla\Phi_j.\,\boldsymbol{n}_{M_JG}-|M_KG|\,\lambda_T(S_{w,M_KG})\sum_{j=1}^{3}P_j\nabla\Phi_j.\,\boldsymbol{n}_{M_KG}\right]$$
$$+\sum_{IJ\epsilon\Omega_f}\left[-0.5\varepsilon\left(\lambda_T(S_{w,J})-\lambda_T(S_{w,I})\right)\frac{P_J-P_I}{IJ}\right]=q_IA_I$$

$$\boldsymbol{\nabla.\left(-\lambda_T\nabla P\right)=\frac{q_w}{\rho_w}+\frac{q_o}{\rho_o}\qquad\lambda_T=\lambda_w+\lambda_o}$$

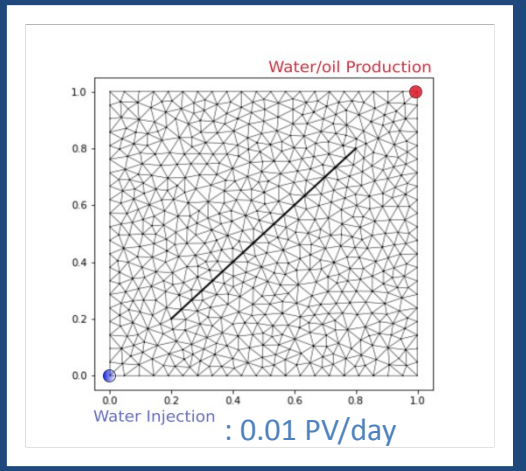
$$S_{w,I}^{t+\Delta t}=S_{w,I}^{t}$$
$$+\frac{\Delta t}{Apor_I}\sum_{IJK\epsilon\Omega_m}\left[f_w(S_{w,M_JG}^{up})\,|M_JG|\,\lambda_T(S_{w,M_JG})\sum_{j=1}^{3}P_j\nabla\Phi_j.\,\boldsymbol{n}_{M_JG}\right]$$
$$+\frac{\Delta t}{Apor_I}\sum_{IJK\epsilon\Omega_m}\left[f_w(S_{w,M_KG}^{up})\,|M_KG|\,\lambda_T(S_{w,M_KG})\sum_{j=1}^{3}P_j\nabla\Phi_j.\,\boldsymbol{n}_{M_KG}\right]$$
$$+\frac{\Delta t}{Apor_I}\sum_{IJ\epsilon\Omega_f}\left[0.5\varepsilon f_w(S_{w,IJ}^{up})\left(\lambda_T(S_{w,J})-\lambda_T(S_{w,I})\right)\frac{P_J-P_I}{IJ}\right]$$
$$+\frac{\Delta t}{Apor_I}(max(q_I,0)+f_w min(q_I,0))A_I$$

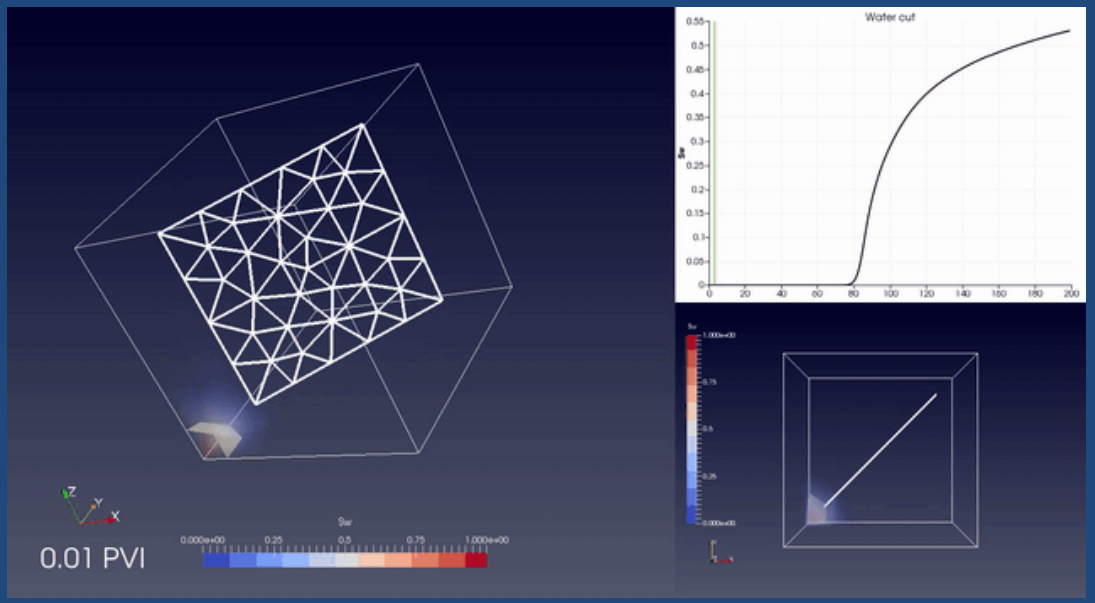
$$\boldsymbol{\Phi\frac{\partial S_w}{\partial t}+\nabla.\left(\frac{\lambda_W}{\lambda_T}\nabla P\right)=\frac{q_w}{\rho_w}}$$

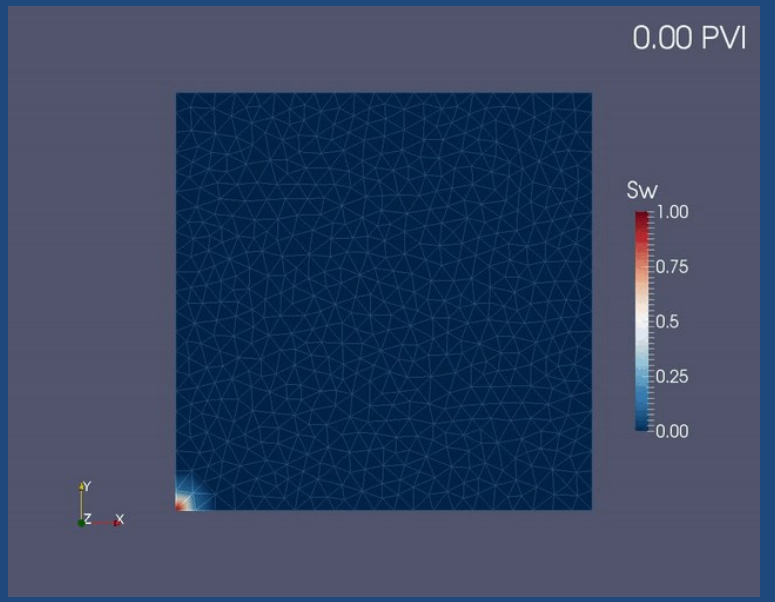


Ecoulements diphasiques en milieux poreux fracturés

– **Résultats** 2D et 3D : Saturation en eau

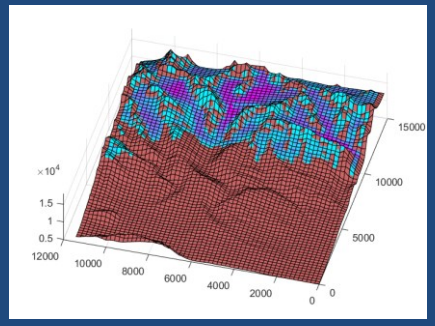




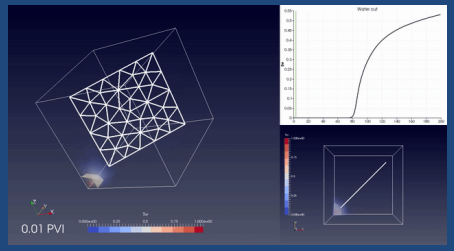


Modélisation d'écoulements
Extrapolations

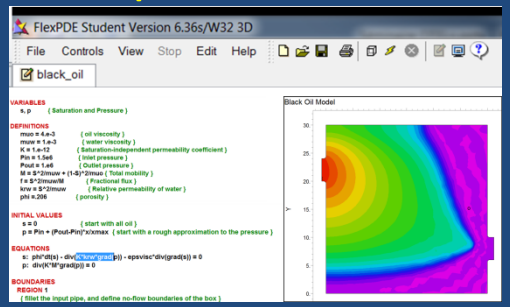
- Ecoulements glaciaires
  - Equation du modèle
  - Discrétisation
  - Premiers Résultats



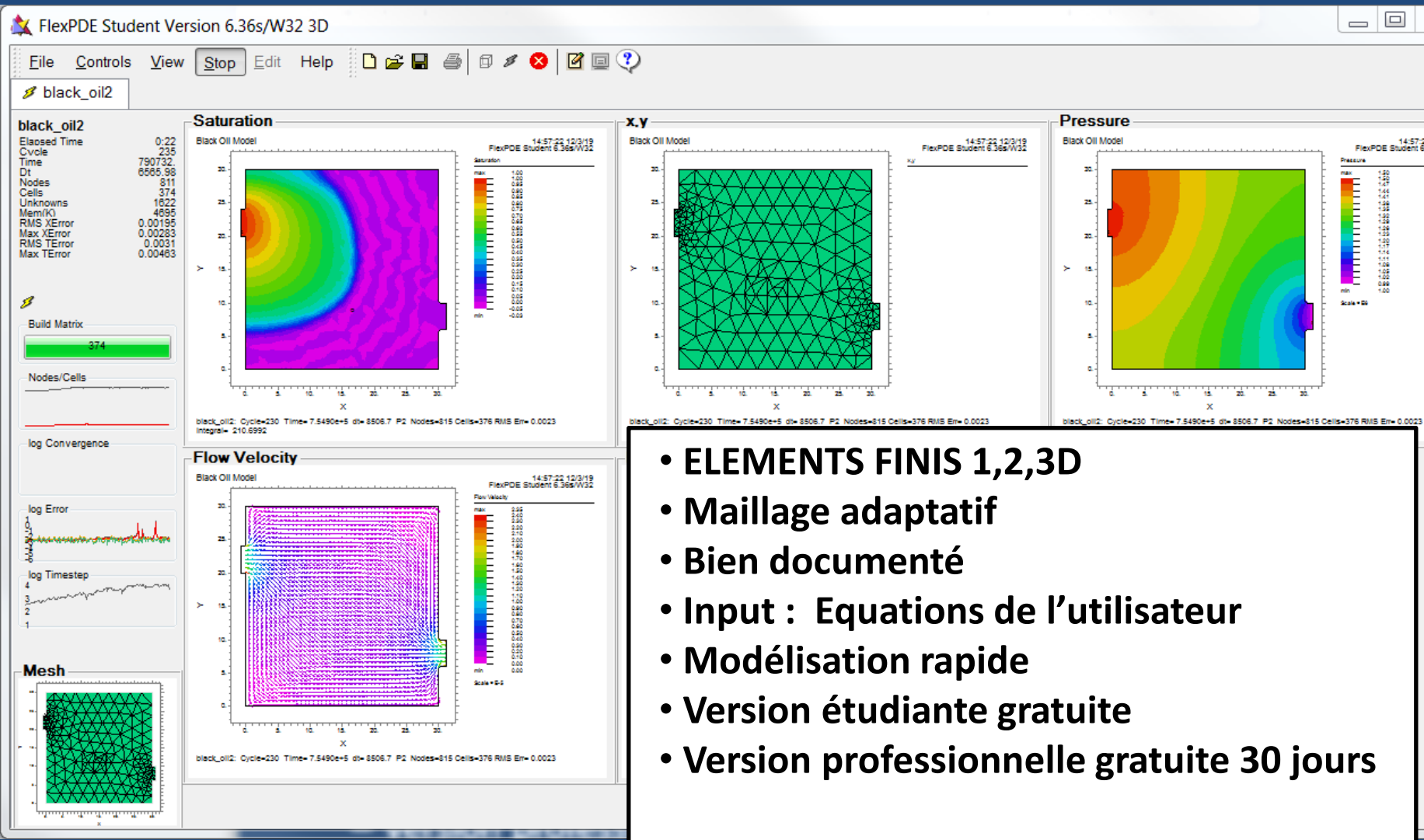
- Ecoulements diphasiques en milieux poreux fracturés
  - Equations du modèle
  - Discrétisation
  - Premiers Résultats



- Connaissant les équations d'un modèle peut-on ?
  - **Eviter de discrétiser ?**
  - **Eviter de tout coder ?**
    - **Ecoulements** avec FlexPDE



FlexPDE

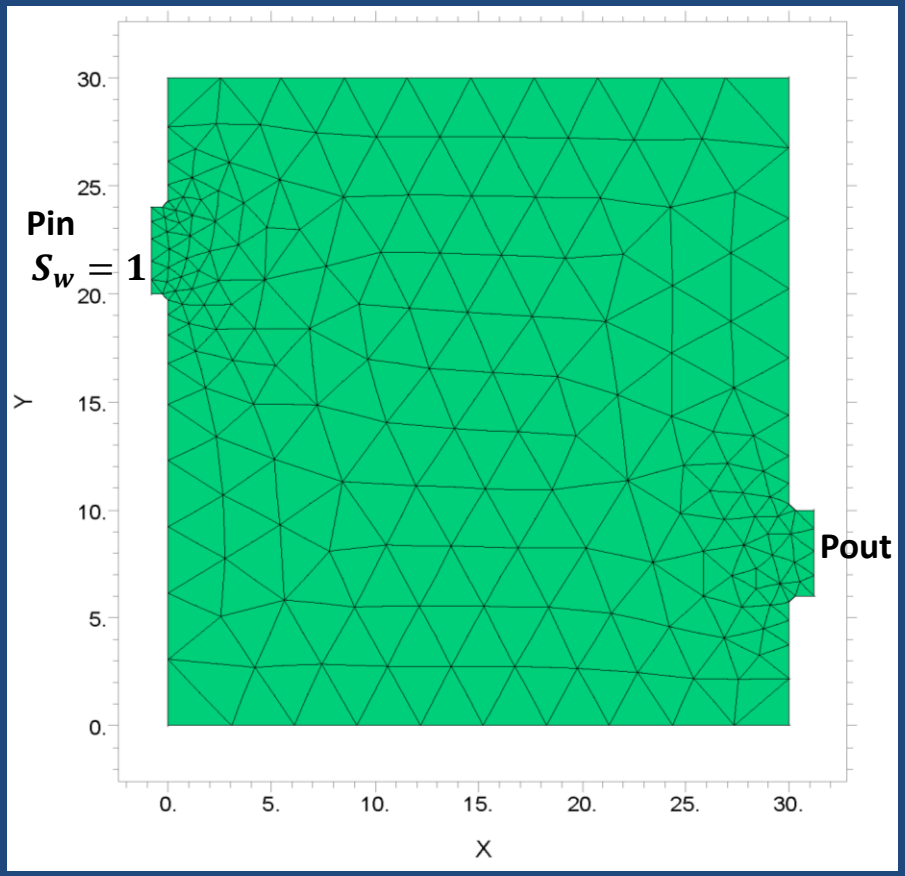


- **ELEMENTS FINIS 1,2,3D**
- **Maillage adaptatif**
- **Bien documenté**
- **Input : Equations de l'utilisateur**
- **Modélisation rapide**
- **Version étudiante gratuite**
- **Version professionnelle gratuite 30 jours**

Ecoulements avec FlexPDE
Exemple : Black Oil - Equations

$$\Phi \frac{\partial S_w}{\partial t} + \nabla . \left(\frac{\lambda_W}{\lambda_T} \nabla P \right) = 0$$

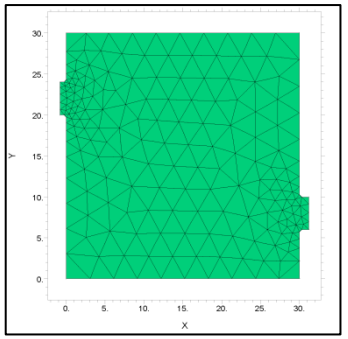
$$\nabla . (-\lambda_T \nabla P) = 0 \qquad \lambda_T = K k r_w$$



Ecoulements avec FlexPDE
Exemple : Black Oil – Script FlexPDE1

$$\Phi \frac{\partial S_w}{\partial t} + \nabla . \left(\frac{\lambda_W}{\lambda_T} \nabla P \right) = 0$$

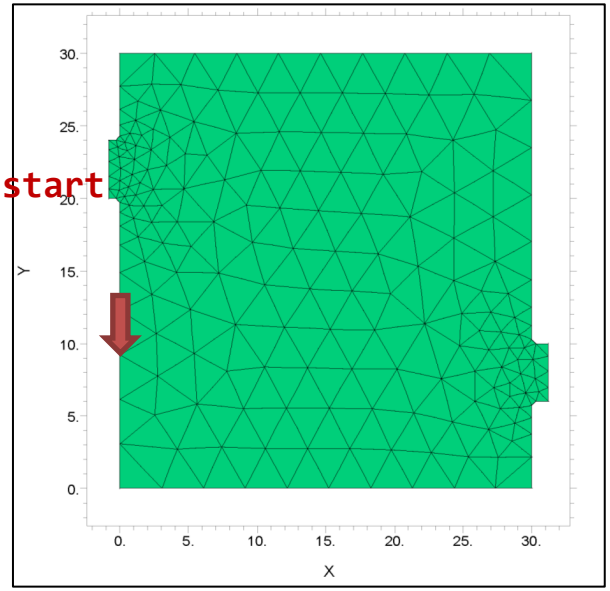
$$\nabla . (-\lambda_T \nabla P) = 0 \qquad \lambda_T = K k r_w$$



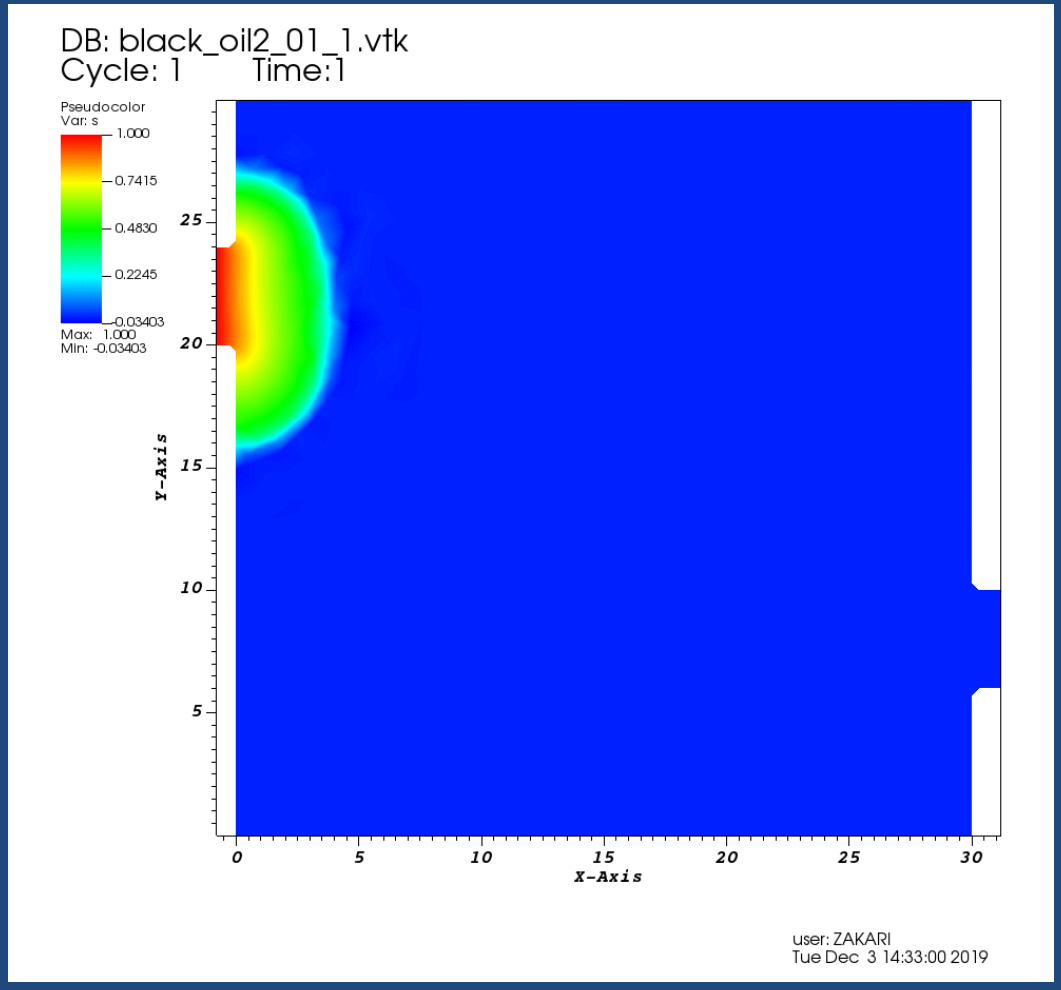
```
TITLE 'Black Oil Model'

VARIABLES
      s, p {Saturation and Pressure}
DEFINITIONS
      Pin  = 1.5e6          { Inlet pressure }
      Pout = 1.e6           { Outlet pressure }
      phi =.206             { porosity }
      K = 1.e-12            { Saturation-independent permeability coefficient }
      krw = S^2/muw         { Relative permeability of water }
      epsvisc = 1.e-6       { A little artificial diffusion 2 smooth the solution }
      sint = integral(s)    { the total extraction integral }
      …
INITIAL VALUES
       s = 0                          { start with all oil }
       p = Pin + (Pout-Pin)*x/xmax    { Initial Pressure approximation}

 EQUATIONS
       s:  phi*dt(s) - div(K*krw*grad(p)) - epsvisc*div(grad(s)) = 0
       p:  div(K*M*grad(p)) = 0
```

Ecoulements avec FlexPDE
Exemple : Black Oil – Script FlexPDE2

```
BOUNDARIES
   REGION 1
     { fillet the input pipe, and define no-flow boundaries of the box }
     start(-2*rad,in_ctr-diam)
       natural(p)=0 natural(s) = 0
       line to (0,in_ctr-diam) bevel(rad)
       line to (0,0) to (xmax,0) to (xmax,out_ctr-diam) bevel(rad)
       line to (xmax+3*rad,out_ctr-diam)

       { set constant outlet pressure, and "tautological" saturation flux }
       value(p) = Pout
       natural(s) =  -K*krw*dx(p)
       line to (xmax+3*rad,out_ctr+diam)

       { reset no-flow box boundaries }
       natural(p)=0  natural(s)=0
       line to (xmax,out_ctr+diam) bevel(rad)
       line to (xmax,ymax) to (0,ymax)
            to (0,in_ctr+diam) bevel(rad)
       line to (-2*rad,in_ctr+diam)

       { set constant inlet pressure and saturation }
       value(p) = Pin   value(s) = 1
       line to close
```

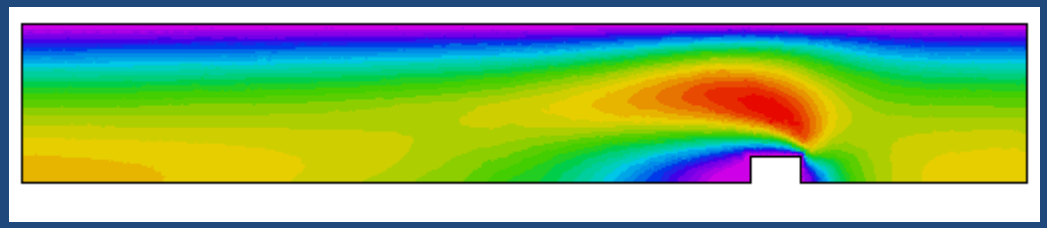


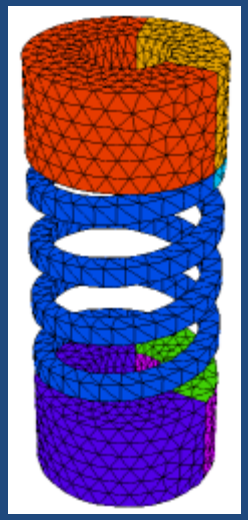
Ecoulements avec FlexPDE
Exemple : Black Oil - Résultats

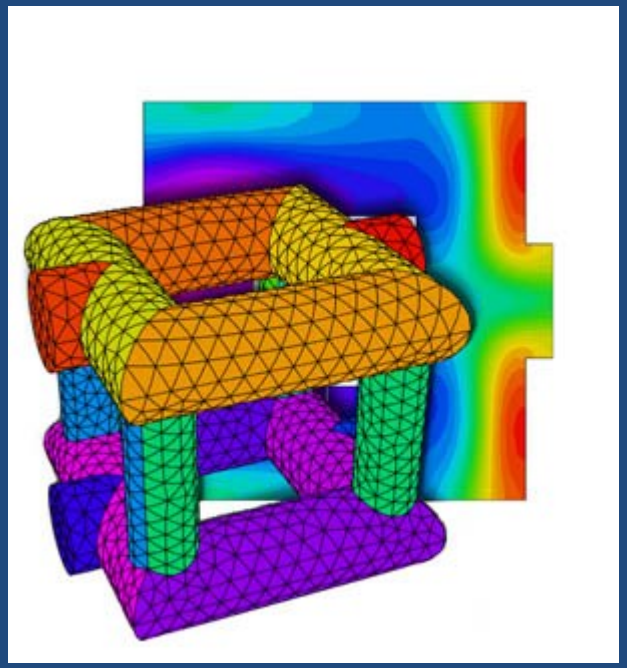


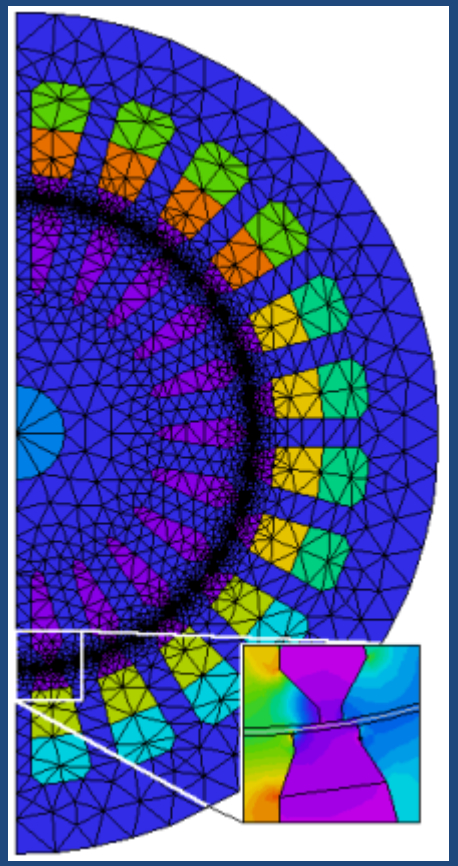
Saturation en eau

Autres modélisations FlexPDE



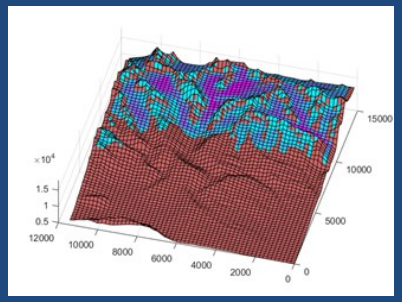




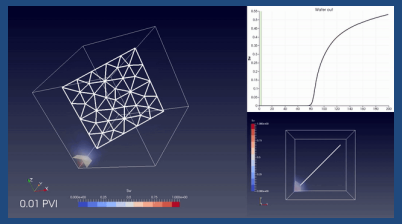


Modélisation d'écoulements
Extrapolations

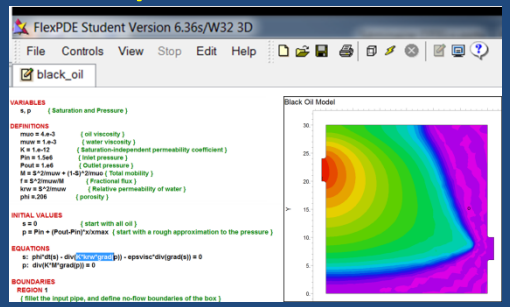
- Ecoulements glaciaires
  - Equation du modèle
  - Discrétisation
  - Premiers Résultats



- Ecoulements diphasiques en milieux poreux fracturés
  - Equations du modèle
  - Discrétisation
  - Premiers Résultats



- Connaissant les équations d'un modèle peut-on ? OUI
  - ✓ **Eviter de discrétiser ?**
  - ✓ **Eviter de tout coder ?**
    - ✓ **Ecoulements** avec FlexPDE



Conclusion

- Calcul parallèle possible en python (Multiprocessing)
- Calcul parallèle possible en Matlab (Parfor)

- Modélisations Eléments-Finis réalisables avec FlexPDE
  - Sans discrétiser
  - Sans vraiment coder

Merci pour votre attention