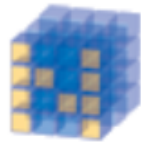


Calcul scientifique avec Python



NumPy
Base N-dimensional array
package



SciPy library
Fundamental library for
scientific computing



Matplotlib
Comprehensive 2D Plotting

≈MATLAB



FiPy: A Finite Volume PDE Solver Using Python

≈ COMSOL

Pourquoi Python ?

- Python : langage de programmation gratuit (libre de droit d'usage) facile à apprendre et à relire (point - : moins rapide que Fortran ou C)
- De nombreuses bibliothèques d'outils spécialisés
-> équivalent de Matlab ou de Comsol
- Une communauté d'utilisateurs (forum)

- Intérêt croissant pour ce langage (des lycées à Google !)
- Un plus important sur le CV (on peut faire bien plus que du calcul)
-> un élément différenciant pour un recrutement

Comment l'installer ?

- Le compilateur / interpréteur
 - Python : <https://www.python.org>
- Le langage + les bibliothèques
 - Sous windows : plateforme Canopy pour Python sous windows <https://www.enthought.com/products/canopy/> (gratuit pour les étudiants et académiques)
 - Sous linux en installant Anaconda
- En ligne
 - <https://www.pythonanywhere.com/>
 - <http://www.sagemath.org/fr/>

- Notebook **IP[y]:** IPython
Interactive Computing
- IPython notebook server in the cloud as [Wakari](#), [Authorea](#) or [CoCalc](#)

Les bibliothèques indispensables

- Pour le calcul

- Numpy
- Scipy
- Matplotlib

Equivalent
de Scilab ou Matlab

- Pour le calcul différentiel

- Fipy

Equivalent
de Comsol



Numpy

- Création de tableau
- Caractéristiques d'un tableau a
 - a.shape -> dimensions
 - a.dtype -> type des éléments
 - a.size -> nombre total d'éléments

```
In [6]: import numpy as np  
a= np.zeros(5)  
a
```

```
Out[6]: array([ 0.,  0.,  0.,  0.,  0.])
```

```
In [7]: b=np.linspace(-2,3,6)  
c=np.arange(-2,3,1)  
print b  
print c
```

```
[-2. -1.  0.  1.  2.  3.]  
[-2 -1  0  1  2]
```

```
In [12]: d=np.array(range(10))  
d
```

```
Out[12]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [11]: x,y=[1,2,3],[4,5,6]  
e=np.array([x,y])  
e
```

```
Out[11]: array([[1, 2, 3],  
               [4, 5, 6]])
```

```
In [20]: print e.shape  
print e.dtype  
print e.size
```

```
(2L, 3L)  
int32  
6
```



Numpy : allocation dans un tableau

- Tableaux alloués en mémoire selon alignement C ou fortran



Indices commencent à 0

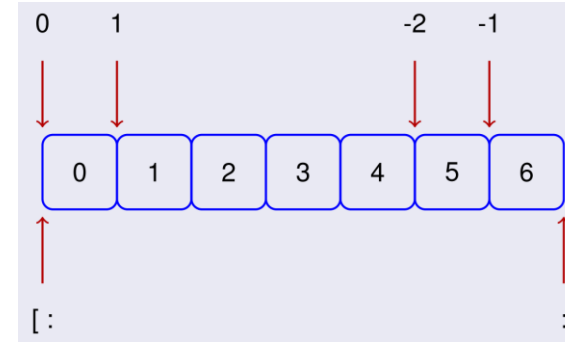
- Accès aux éléments

```
x,y=[1,2,3],[4,5,6]
e=np.array([x,y])
print e
print e[0,0] # La 1er ligne et 1ere colonne
print e[1,2] # La 2eme ligne et 3eme colonne
print e[-1,-1] # La dernière ligne et dernière colonne
print e[:,1] # toutes les lignes et 2eme colonne
print e[e>2]
```

```
[[1 2 3]
 [4 5 6]]
```

```
1
6
6
```

```
[2 5]
[3 4 5 6]
```



- Modification des éléments

```
e[1,1]=10
print e
[[ 1  2  3]
 [ 4 10  6]]
```

```
e[:,1]=10
print e
[[ 1 10  3]
 [ 4 10  6]]
```



Numpy : redimensionnement d'un tableau

- Tableaux alloués en mémoire

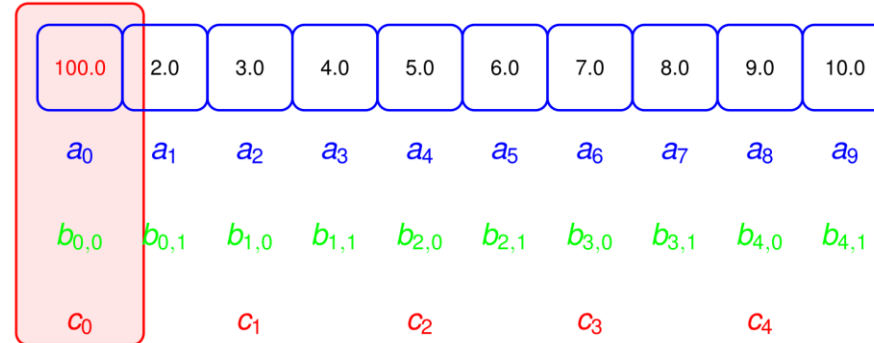
- Redimensionnement .reshape



Attention, la modification d'un élément s'applique à tous les tableaux liés

- Pour créer, un nouveau tableau (nouvelle allocation en mémoire)

utiliser `copy()`



```

a= np.linspace(1,10,10)
b= a.reshape((5,2))
c=b[:,0]
a[0]=100
print a
print b
print c

```

```

[ 100.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
[[ 100.  2.]
 [  3.  4.]
 [  5.  6.]
 [  7.  8.]
 [  9. 10.]]
[ 100.  3.  5.  7.  9.]

```

```

c=a.copy()
a[1]=200.
print a
print c

```

```

[ 100. 200.  3.  4.  5.  6.  7.  8.  9. 10.]
[ 100.  2.  3.  4.  5.  6.  7.  8.  9. 10.]

```





Numpy : opérations sur les tableaux

- Somme d'éléments : `a.sum()` ou `np.sum(a)`
- Somme de tableau : `a+a`
- Multiplication : `a*a`
- Produit vectoriel ou matriciel : `np.dot(a,a)`

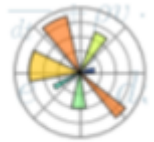


Numpy

- Fonctions de calcul basiques

```
In [11]: from numpy import *
x= linspace (0.,1., 100)
f= 10.*exp(-((x-0.5)/0.1)**2)
print f

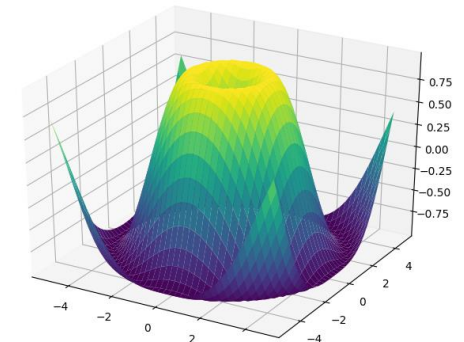
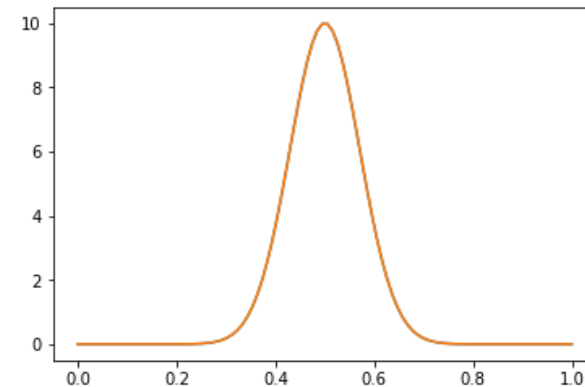
[ 1.38879439e-10  3.77474939e-10  1.00525455e-09  2.62302071e-09
 6.70602464e-09  1.67983382e-08  4.12292325e-08  9.91475372e-08
 2.33612654e-07  5.39322510e-07  1.21994003e-06  2.70374774e-06
 5.87126398e-06  1.24920812e-05  2.60420842e-05  5.31929936e-05
 1.06456180e-04  2.08749322e-04  4.01067085e-04  7.54999601e-04
 1.39256082e-03  2.51663004e-03  4.45617596e-03  7.73113109e-03
 1.31420010e-02  2.18885859e-02  3.57200177e-02  5.71141054e-02
 8.94772943e-02  1.37347279e-01  2.06568943e-01  3.04402171e-01
 4.39509449e-01  6.21765240e-01  8.61831599e-01  1.17045882e+00
 1.55749843e+00  2.03065828e+00  2.59408275e+00  3.24689734e+00
 3.98190635e+00  4.78466215e+00  5.63312339e+00  6.49807924e+00
 7.34443672e+00  8.13335512e+00  8.82508158e+00  9.38221701e+00
 9.77304662e+00  9.97452490e+00  9.97452490e+00  9.77304662e+00
 9.38221701e+00  8.82508158e+00  8.13335512e+00  7.34443672e+00
 6.49807924e+00  5.63312339e+00  4.78466215e+00  3.98190635e+00
 3.24689734e+00  2.59408275e+00  2.03065828e+00  1.55749843e+00
 1.17045882e+00  8.61831599e-01  6.21765240e-01  4.39509449e-01
 3.04402171e-01  2.06568943e-01  1.37347279e-01  8.94772943e-02
 5.71141054e-02  3.57200177e-02  2.18885859e-02  1.31420010e-02
 7.73113109e-03  4.45617596e-03  2.51663004e-03  1.39256082e-03
 7.54999601e-04  4.01067085e-04  2.08749322e-04  1.06456180e-04
 5.31929936e-05  2.60420842e-05  1.24920812e-05  5.87126398e-06
 2.70374774e-06  1.21994003e-06  5.39322510e-07  2.33612654e-07
 9.91475372e-08  4.12292325e-08  1.67983382e-08  6.70602464e-09
 2.62302071e-09  1.00525455e-09  3.77474939e-10  1.38879439e-10]
```



Matplotlib

- Tracé

```
In [13]: import matplotlib.pyplot as plt
plt.plot(x, f)
plt.show()
```



(voir [matplotlib](#) pour toutes les options de graphes
prévoir plusieurs soirées !)

Capacity Building

Coopération et innovation pédagogique: Eau-Energie-Habitat



Erasmus+



Coopération et innovation pédagogique: Eau-Energie-Habitat

Scipy : pour le calcul scientifique

- Modules de calculs scientifiques
 - Solveur [optimize](#)
 - **Bisect**
 - [Newton](#)
 - [fsolve](#)
 - Interpolation [interpolate](#)
 - Integration [integrate](#)
 - Equation différentielle ordinaire [odeint](#)

➔ Application à des cas fréquents retrouvés sur les problèmes
Chemical Engineering



Recherche de racine

Racine de la fonction

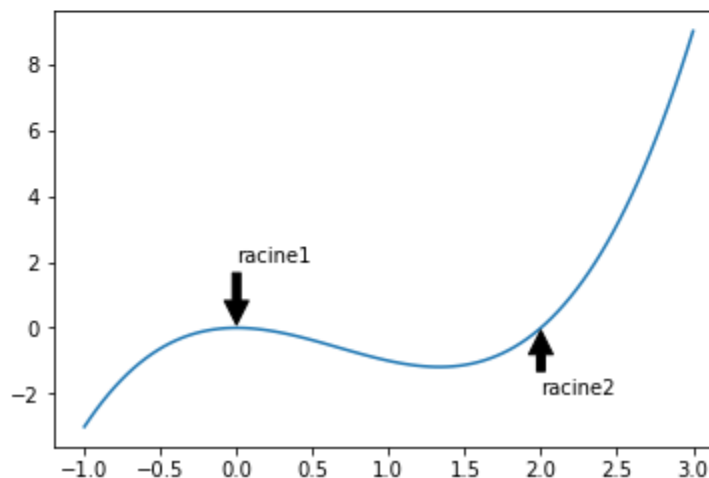
$$x^3 - 2x^2 = x^2(x-2)$$

```
import matplotlib.pyplot as plt
import numpy as np
def f(x):
    return (x**3)-2*(x**2)
x=np.linspace(-1,3,100)
plt.plot(x,f(x))

from scipy.optimize import newton
x1=newton(f,x0=1.)
x2=newton(f,x0=3.)
print x1
print x2
plt.annotate('racine1', xy=(x1, 0), xytext=(x1, 2),arrowprops=dict(facecolor='black', shrink=0.05),)
plt.annotate('racine2', xy=(x2, 0), xytext=(x2, -2),arrowprops=dict(facecolor='black', shrink=0.05),)
plt.show()
```

-1.82851760603e-08

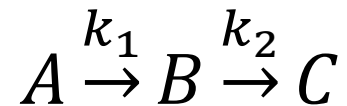
2.0





Intégration d'EDO

Succession de deux réactions chimiques de premier ordre



$$\frac{dA(t)}{dt} = -k_1 A(t)$$

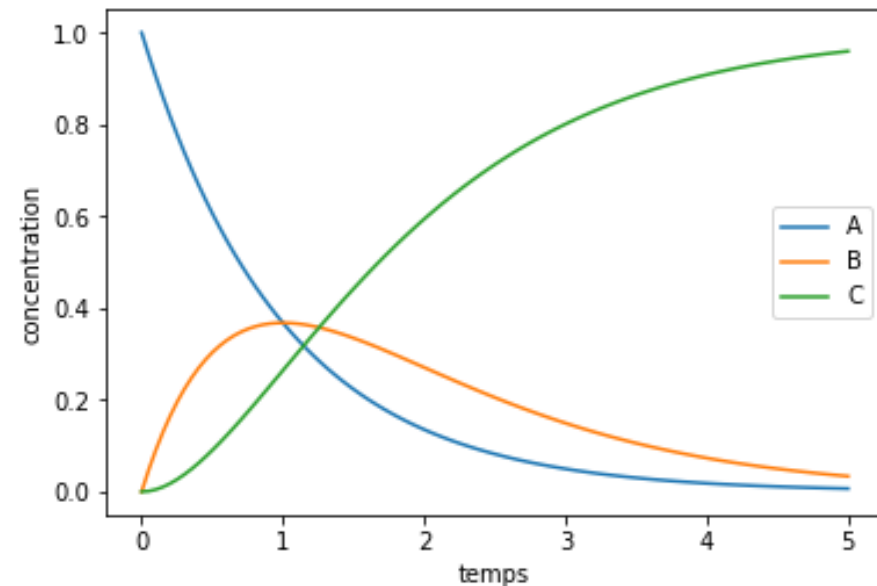
$$\frac{dB(t)}{dt} = k_1 A(t) - k_2 B(t)$$

$$\frac{dC(t)}{dt} = k_2 B(t)$$

```
In [27]: from numpy import *
         from scipy import integrate
         k1, k2=1.,1.
         def dX_dt(X, t=0):
             return array([-k1*X[0],k1*X[0]-k2*X[1],k2*X[1]])
         t = linspace(0, 5, 100)           # time
         X0 = array([1, 0, 0])           # conditions initiales
         X, infodict = integrate.odeint(dX_dt, X0, t, full_output=True)
         infodict['message']
```

Out[27]: 'Integration successful.'

```
In [28]: import matplotlib.pyplot as plt
         [A,B,C]=plt.plot (t, X )
         plt.legend([A,B,C],[ "A", "B", "C"], loc='best')
         plt.xlabel ('temps')
         plt.ylabel ('concentration')
         plt.show()
```



Erasmus+



Capacity Building

Coopération et innovation pédagogique: Eau-Energie-Habitat



Application 1 : réacteur d'ozonation

Cinétique de production d'ozone : $3O_2 \xrightarrow{k} 2O_3$

Voir cours K. Serrano

$$-\frac{\varepsilon}{V} \frac{dn_{O_2}}{dt} = k [O_2]^{\frac{3}{2}}, \quad n_{O_2} = n_0(1 - x), \quad \varepsilon = -1/3$$

$$\frac{dx}{dt} = -\frac{k}{\varepsilon} \left(\frac{n_0}{V_0} \right)^{1/2} \frac{(1 - x)^{3/2}}{(1 + \varepsilon x)^{1/2}}$$

- 1- Tracer le taux de conversion en fonction du temps
- 2 -Trouver le temps pour atteindre 99% de conversion
- 3- un CT égal à $2 \text{ mg} \cdot \text{min} \cdot \text{L}^{-1}$ est suffisant pour détruire à 99 % les bactéries, les virus et l'ensemble des kystes de *Giardia*.



Erasmus+

<https://www.suezwaterhandbook.fr/procedes-et-technologies/oxydation-desinfection/oxydation-et-desinfection-par-l-ozone/applications>

MADEEH
Energie - Eau - Habitat - Informatique - Anthropologie

Capacity Building

Coopération et innovation pédagogique: Eau-Energie-Habitat à Madagascar

P. Bacchin Univ. Toulouse

13



Université
de Toulouse

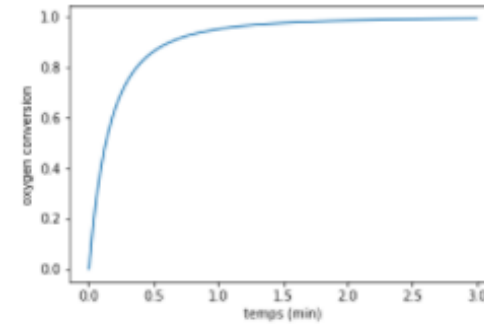
- Solution :

[http://nbviewer.jupyter.org/
url/www.patricebacchin.fr/p
ython/Ozonation.ipynb](http://nbviewer.jupyter.org/url/www.patricebacchin.fr/python/Ozonation.ipynb)

```
from numpy import *
from scipy import integrate
k1, c0, eps = 0.316, 40.09, -1./3. # mol-1/2.m-3/2, mol.m-3, -
kc = -k1*(c0**0.5)/eps
def dX_dt(X, t):
    return kc*((1-X)**1.5)/((1+eps*X)**0.5)
t = linspace(0, 3, 100) # time
X0 = 0. # conditions initiales
X, infodict = integrate.odeint(dX_dt, X0, t, full_output=True)
infodict['message']
```

'Integration successful.'

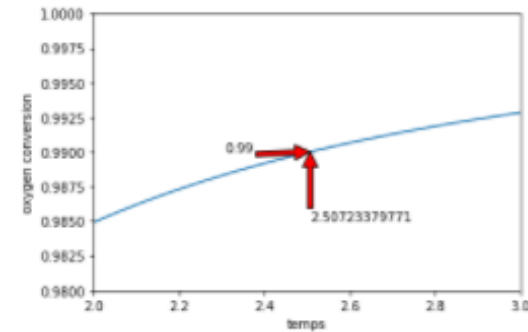
```
import matplotlib.pyplot as plt
plt.plot(t, X)
plt.xlabel('temps (min)')
plt.ylabel('oxygen conversion')
plt.show()
```



```
cible=0.99
def dt_dX(X):
    return 1./(kc*((1-X)**1.5)/((1+eps*X)**0.5))
tb=integrate.quad(dt_dX, 0, cible)[0]
print tb, 'min'
```

2.50723379771 min

```
plt.plot(t, X)
plt.xlim(2,3)
plt.ylim(0.98,1)
plt.xlabel('temps')
plt.ylabel('oxygen conversion')
plt.annotate(cible, xy=(tb, cible), xytext=(tb-0.2, cible),arrowprops=dict(facecolor='red'),)
plt.annotate(tb, xy=(tb, cible), xytext=(tb, cible-0.005),arrowprops=dict(facecolor='red'),)
plt.show()
```





Convection-diffusion-réaction

Un problème classique en génie chimique : réacteur piston, transport de polluant dans un sol ou une rivière mais aussi une membrane catalytique, une dialyse avec adsorbant ...

$$\frac{dc}{dt} = -\frac{dj}{dz} - kc$$

$$j = uc - D\frac{dc}{dz}$$

$$\left\{ \begin{array}{l} c = c_0 \quad z = 0 \\ \frac{dc}{dz} = 0 \quad z = \delta \end{array} \right.$$

$$D \frac{d^2c}{dz^2} - u \frac{dc}{dz} - kc = 0$$

Système de 2 EDO

$$\left\{ \begin{array}{l} \frac{dc}{dz} = v \\ \frac{dv}{dz} = \frac{u}{D} v + \frac{k}{D} c \end{array} \right.$$

$$\left\{ \begin{array}{l} \frac{d\hat{c}}{d\hat{z}} = \hat{v} \\ \frac{d\hat{v}}{d\hat{z}} = Pe\hat{v} + Ha^2\hat{c} \end{array} \right.$$

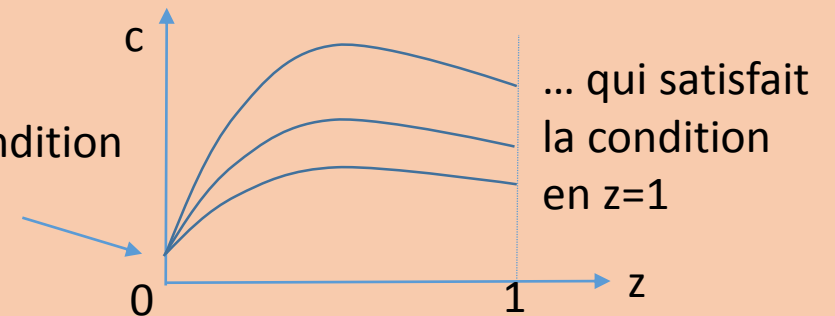
Méthode du tir ou Shooting method

Souvent, les 2 conditions limites s'appliquent aux extrémités du domaine.

Les intégrations doivent s'effectuer avec 1 condition limite fixée de façon aléatoire en $z=0$ puis déterminée par itérations pour respecter la condition limite en $z=1$

Intégration **-odeint-** (tir) avec différents $\frac{dc}{dz}|_{z=0}$ puis trouver par optimization **-newton-** la condition en $z=0$ qui permet d'avoir la condition souhaitée en $z=L$

Trouver la condition en $z=0$...





Application 2 : convection-diffusion-reaction

1- Ecrire le code pour la résolution numérique en régime permanent pour les conditions suivantes

$$\begin{aligned} c &= c_0 & z &= 0 \\ \frac{dc}{dz} &= 0 & z &= \delta \end{aligned}$$

2 -Trouver la concentration en sortie pour un Péclet de 1 et un nombre de Hatta de 2



Convection-diffusion -réaction

Valeur $\frac{dc}{dz}|_{z=1}$ souhaitée

Valeur $\frac{dc}{dz}|_{z=0}$ arbitraire

Fonction cible retournant après intégration

$\frac{dc}{dz}|_{z=1}$ calculée - souhaitée

Optimisation pour trouver $\frac{dc}{dz}|_{z=0}$
afin que la fonction cible soit nulle

```
# Convection-diffusion-reaction (first order) equation at steady state
# Shooting method to solve the problem for boundary conditions at both end
from numpy import *
from scipy import integrate
from scipy import optimize
Pe, Ha=1.,2.
C0=1.
dC1=0.
n=100
z = linspace(0, 1, n)

#Set of ordinary differential equations
def dX_dt(X, z):
    return array([X[1],Pe*X[1]+Ha*Ha*X[0]])

dC0_try=-Pe
# function giving the gap between dc/dz|z=1 and the targeted boundary conditions
def cible(dC0_try):
    X0 = array([C0, dC0_try]) # boundary conditions at t=0
    X, infodict = integrate.odeint(dX_dt, X0, z, full_output=True)
    return X[n-1,1]-dC1

#find the dc/dz|z=0 to have a cible=0
dC0_found=optimize.newton(cible, dC0_try)
print "Solution found for dc/dz|z=0 at", dC0_found

X0 = array([C0, dC0_found])
X, infodict = integrate.odeint(dX_dt, X0, z, full_output=True)
print "c|z=0 = ", X[0,0], " c|z=1 = ", X[n-1,0]
print "dc/dz|z=0 =", X[0,1], " dc/dz|z=1 =", X[n-1,1]
```

```
Solution found for dc/dz|z=0 at [-1.52124686]
c|z=0 = 1.0 c|z=1 = 0.334411336713
dc/dz|z=0 = -1.52124686145 dc/dz|z=1 = 1.88003432267e-12
```



Convection-diffusion -réaction

Tracé de la solution numérique

Comparaison aux solutions analytiques publiées pour un ordre un :

$$c = c_0 \quad z = 0$$

$$\frac{dc}{dz} = 0 \quad z = \delta$$

Applications aux membranes catalytiques

Gu, Y., Bacchin, P., Lahitte, J. F., Remigy, J. C., Favier, I., Gómez, M., ... & Noble, R. D. (2017). Catalytic membrane reactor for Suzuki-Miyaura C-C cross-coupling: Explanation for its high efficiency via modeling. *AIChE Journal*, 63(2), 698-704.

Applications à la dialyse avec absorbant

$$c = c_0 \quad z = 0$$

$$c = c_1 \quad z = \delta$$

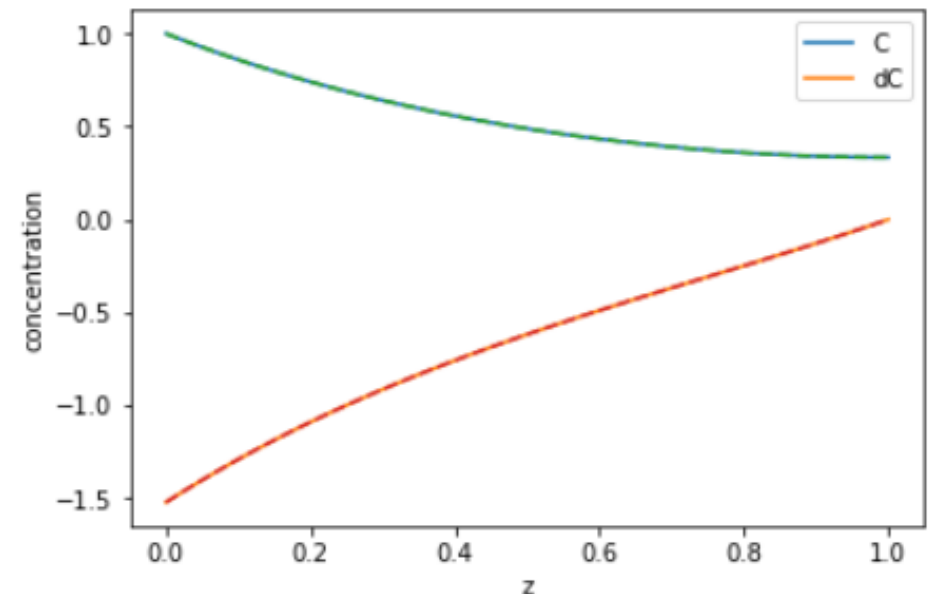
Snisarenko, D., Pavlenko, D., Stamatialis, D., Aimar, P., Causserand, C., & Bacchin, P. (2017). Insight into the transport mechanism of solute removed in dialysis by a membrane with double functionality. *Chemical Engineering Research and Design*, 126, 97-108.

```
import matplotlib.pyplot as plt
[C, dC]=plt.plot (z, X )

#analytical solution
delta=(Pe*Pe+4*Ha*Ha)**0.5
c=exp((Pe-delta)*z/2.)/(1-(Pe-delta)*exp(-delta)/(Pe+delta)
dc=((Pe-delta)/2.)*exp((Pe-delta)*z/2.)/(1-(Pe-delta)*exp(

plt.plot(z,c,'--')
plt.plot(z,dc,'--')
plt.legend([C, dC],["C","dC"], loc='best')
plt.xlabel ('z')
plt.ylabel ('concentration')
plt.show()
```

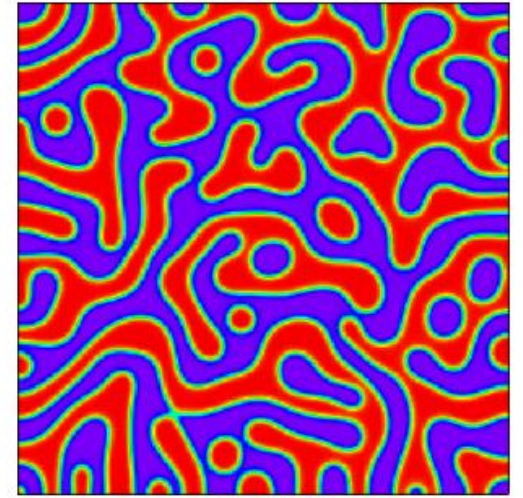
-1.52124683975



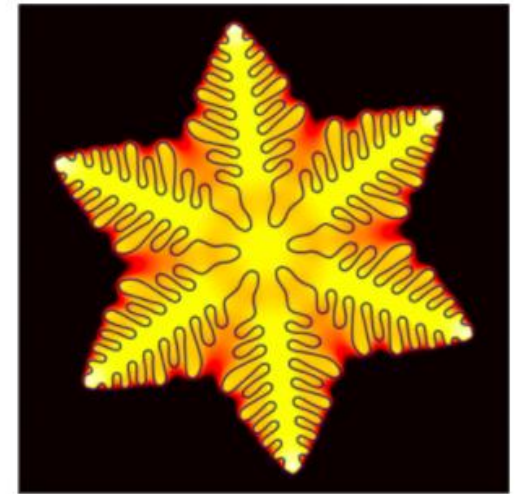


FIPY

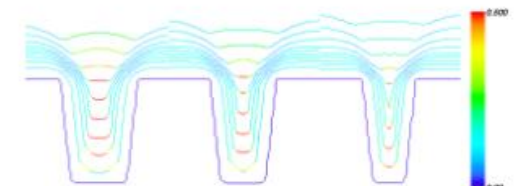
- Résolution d'équations différentielles partielles EDP
 - Transport de matière en 2D-3D en instationnaire
 - Navier Stokes



Cahn-Hilliard



Phase Field



Level Set

Pour une future formation ?



Capacity Building
Coopération et innovation pédagogique: Eau-Energie-Habitat à Madagascar

Références

- Documentation en ligne : [python](#) , [numpy](#), [matplotlib](#), [scipy](#), [fipy](#)
- <http://code.activestate.com/recipes/langs/python/>
- <https://docs.scipy.org/doc/scipy/reference/>
- Tutoriel en Français : <https://docs.python.org/fr/3/tutorial/index.html>

- Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3).

- Guyer, J. E., Wheeler, D., & Warren, J. A. (2009). FiPy: Partial differential equations with Python. *Computing in Science & Engineering*, 11(3).

Annexe

- [Lien vers le formulaire d'évaluation de la formation](#)

Données


- Liste [,] : serie d'objets non nécessairement identiques
- Tuple (,) : liste non modifiable
- Dictionnaire { , } : clé/valeur non ordonnées

Commandes de tests

indentation → If <condition>:
 <commandes>
Elif <condition>:
 <commandes>
Else :
 <commandes>

Try:
 <commandes>
Except <condition>:
 <commandes>
Else :
 <commandes>
Finally:
 <commandes>

Commandes de boucles

indentation  For <variables> in <serie>:
 <commandes>
Else :
 <commandes>

While <conditions> :
 <commandes>
Else :
 <commandes>

Fonction

```
Def <nom> ( <arg>, ....., <arg1>=<valeur>, ...):  
    <commandes>  
    return <valeur>
```

Classes et objets

Permet de réaliser une structuration spéciale de données (association de données et de fonction)

Outils intéressant pour structurer des concepts physiques et la logique de traitement

Class <nom>: