Modélisation numérique en physique

Suites et relations de récurrence

Cours et calepins : https://phys-mod.readthedocs.io/en/latest/

Diata Traore (diata.traore@sorbonne-universite.fr)

Cours : Compréhensions de listes et récursion

Schémas de manipulation de listes

Schémas		Compréhensions de liste
Schéma de construction	Utiliser une méthode algorithmique	`[<elem> for <var> in <seq>]`
Schéma de transformation	Appliquer une fonction aux éléments d'une liste	`[<fonction> for <var> in <seq>]`

Schéma de filtrage	Filtrer les éléments d'une liste à partir d'une condition	`[<elem> for <var> in <seq> if <condition>]`
		
		`[<elem> if <condition1> else <condition2> for <var> in <seq>]`

Réduction	Extraire une information simple à partir d'une liste	exemple : sum([1,2,3,4]) len([1,2,3,4]) ...

Schéma de construction - exemples

Méthode 1 :

```
def naturelsFor(a, b):
    ''' retourne une liste d'entiers [a,b[ '''
    listRes = []
    for k in range(a, b):
        listRes.append(k)
    return listRes

naturelsFor (2,8)
>> [2,3,4,5,6,7]
```

Schéma de construction - exemples

Méthode 1 :

```
def naturelsFor(a, b):
    ''' retourne une liste d'entiers [a,b[ '''
    listRes = []
    for k in range(a, b):
        listRes.append(k)
    return listRes

naturelsFor (2,8)
>> [2,3,4,5,6,7]
```

Méthode 2 :

```
[<elem> for <var> in <seq>]
```

```
def naturelsFor2(a, b):
    ''' retourne une liste d'entiers [a,b[ '''
    return [k for k in range(a, b)]

naturelsFor2 (2,8)
>> [2,3,4,5,6,7]
```

Schéma de transformation - exemples

Méthode 1 :

```
[<fonction> for <var> in <seq>]
```

```
def carre(x):
    ''' retourne le carre de x '''
    return x*x

[carre(k) for k in range(1, 7)]
>> [1,4,9,16,25,36]

# Ou
[carre(k) for k in [1,2,3,4,5,6]]
>> [1,4,9,16,25,36]

# Ou
list = [1,2,3,4,5,6]
[carre(k) for k in list]
>> [1,4,9,16,25,36]
```

Schéma de transformation - exemples

Méthode 1 :

```
[<fonction> for <var> in <seq>]
```

```
[carre(k) for k in range(1, 7)]
>> [1,4,9,16,25,36]
```

Méthode 2 :

```
[<fonction> for <var> in <seq>]
```

```
[(lambda x : x**2)(x) for x in range(1,7)]
>> [1,4,9,16,25,36]
```

Schéma de transformation - exemples

Méthode 1 :

```
[carre(k) for k in range(1, 7)]
>> [1,4,9,16,25,36]
```

```
[<fonction> for <var> in <seq>]
```

Méthode 2 :

```
[(lambda x : x**2)(x) for x in range(1,7)]
>> [1,4,9,16,25,36]
```

```
[<fonction> for <var> in <seq>]
```

Méthode 3 :

```
list(map(lambda x : x**2, [1, 2, 3, 4, 5, 6]))
>> [1,4,9,16,25,36]
```

```
list(map(lambda : opération,
ma_liste))
```

Schéma de filtrage - exemples

Méthode 1 :

```
[k*k for k in range(1, 7) if k%2 == 0]
```

```
[<elem> for <var> in <seq> if <condition>]
```

Méthode 2 :

```
[k*k if k%2 == 0 else k*k*k for k in range(1, 7)]
```

```
[<elem> if <condition1> else <condition2> for <var> in <seq>]
```

Autres notions du cours

- **Tester une fonction** avec le mot clé `assert`
- Appliquer les schémas aux **chaînes de caractères**
- Appliquer les schémas à des listes de dimensions > 1 (**compréhension multiple**)
- Construire une **fonction de récursion**

Objectif de la séance

- Faire le calepin d'exercices

Avant la prochaine séance

- Faire le calepin "Représentation graphique d'une série mathématique"

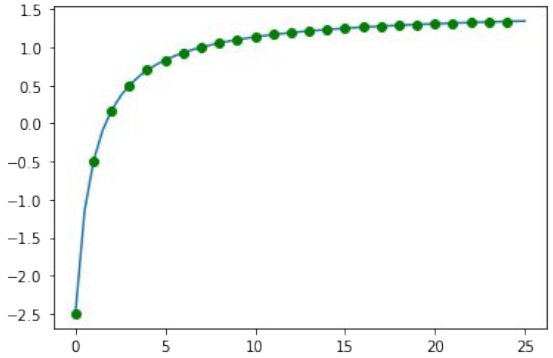
Cours : Représentation graphique d’une série mathématique

Représentation graphique d'un série mathématique

Représenter les éléments d'une suite :

Définir la fonction suite	$F(n) = \frac{3n-5}{2n+2}$	`def F(n):`
		`  ''' retourne l'element Sn de la suite '''`
		`  return (3*n - 5) / (2*n + 2)`

Calculer les éléments de ma suite et les stocker dans un tableau	$S_n = \frac{3n-5}{2n+2}$	`liste_elements = [F(k) for k in range(N)]`



Représentation graphique d'un série mathématique

Représenter une suite définie par récurrence :

$$u_{n+1} = u_n - \ln(u_n)$$

Initialiser la suite	$u_0 = 10$	
Définir la fonction associée à la suite	$G(x) = x - \ln(x)$	`def G(x):` `    return x - np.log(x)`

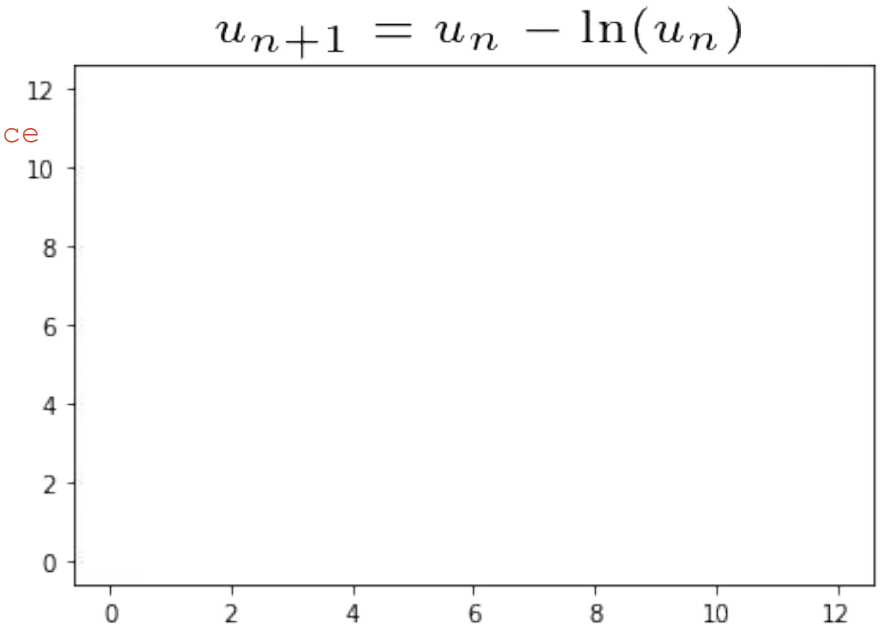
Calculer les termes suivants	(voir tableau)	`for k in range(N):`
		`    # etape 2: image du point U0 avec une ligne`
		`    #   verticale -> U_01`
		`    u1 = G(u0)`
		`    plt.plot([u0, u0], [u0, u1], 'k')`
		`    plt.plot([u0], [u1], 'og')`
		`    # etape 3: placer le point U1 sur la bissectrice`
		`    #   avec une ligne horizontale depuis U_01`
		`    plt.plot([u0, u1], [u1, u1], 'k')`
		`    plt.plot([u1], [u1], 'og')`
		`    # recurrence:`
		`    u0 = u1`

Représentation graphique d'un série mathématique

Représenter une suite définie par récurrence :

```
# Initialisation
plt.plot([u0], [u0], 'og')

# Récurrence
for k in range(N):
    # etape 2: image du point U0 avec une ligne
    #   verticale -> U_01
    u1 = G(u0)
    plt.plot([u0, u0], [u0, u1], 'k')
    plt.plot([u0], [u1], 'og')
    # etape 3: placer le point U1 sur la bissectrice
    #   avec une ligne horizontale depuis U_01
    plt.plot([u0, u1], [u1, u1], 'k')
    plt.plot([u1], [u1], 'og')
    # recurrence:
    u0 = u1
```

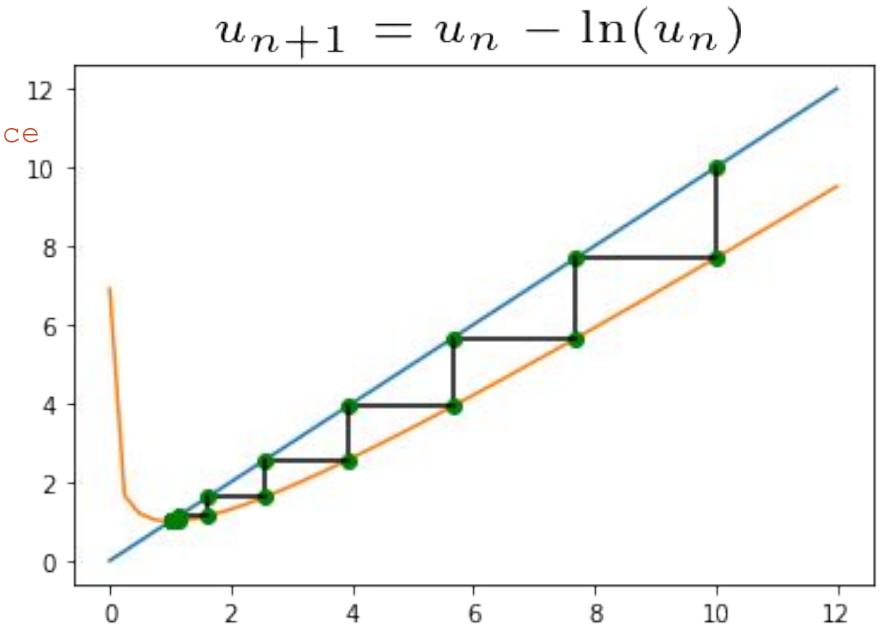


Représentation graphique d'un série mathématique

Représenter une suite définie par récurrence :

```
# Initialisation
plt.plot([u0], [u0], 'og')

# Récurrence
for k in range(N):
    # etape 2: image du point U0 avec une ligne
    #   verticale -> U_01
    u1 = G(u0)
    plt.plot([u0, u0], [u0, u1], 'k')
    plt.plot([u0], [u1], 'og')
    # etape 3: placer le point U1 sur la bissectrice
    #   avec une ligne horizontale depuis U_01
    plt.plot([u0, u1], [u1, u1], 'k')
    plt.plot([u1], [u1], 'og')
    # recurrence:
    u0 = u1
```



Mini-projet : Un exemple de systèmes chaotique

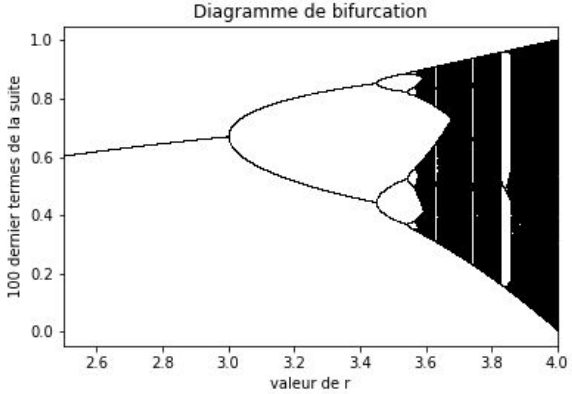
A déposer sur Moodle avant dimanche 07/03, 18h.

Objectif du mini-projet

Partie 1 : Diagramme de bifurcation

Objectif : Déterminer la dépendance de la **limite de** la suite x, **l'équation logistique,** à la valeur de r

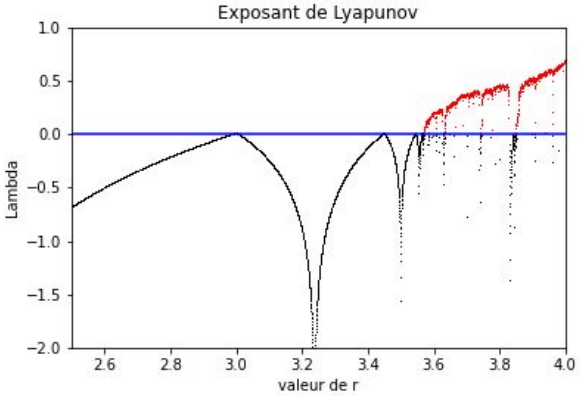
$$x_{i+1} = r(1 - x_i)x_i$$



Partie 2 : Exposant de Lyapunov

Objectif : Caractériser la **stabilité** de l'équation logistique

$$\lambda = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \log |f'(x_i)|$$



graphiques : https://phys-mod.readthedocs.io/en/latest/index.html

Préparation de la séquence “Ajuster un modèle”

- Faire le calepin de cours “Ajuster un modèle aux données”

- Avant la séance du jeudi : lire l’article du site de Carbon Brief