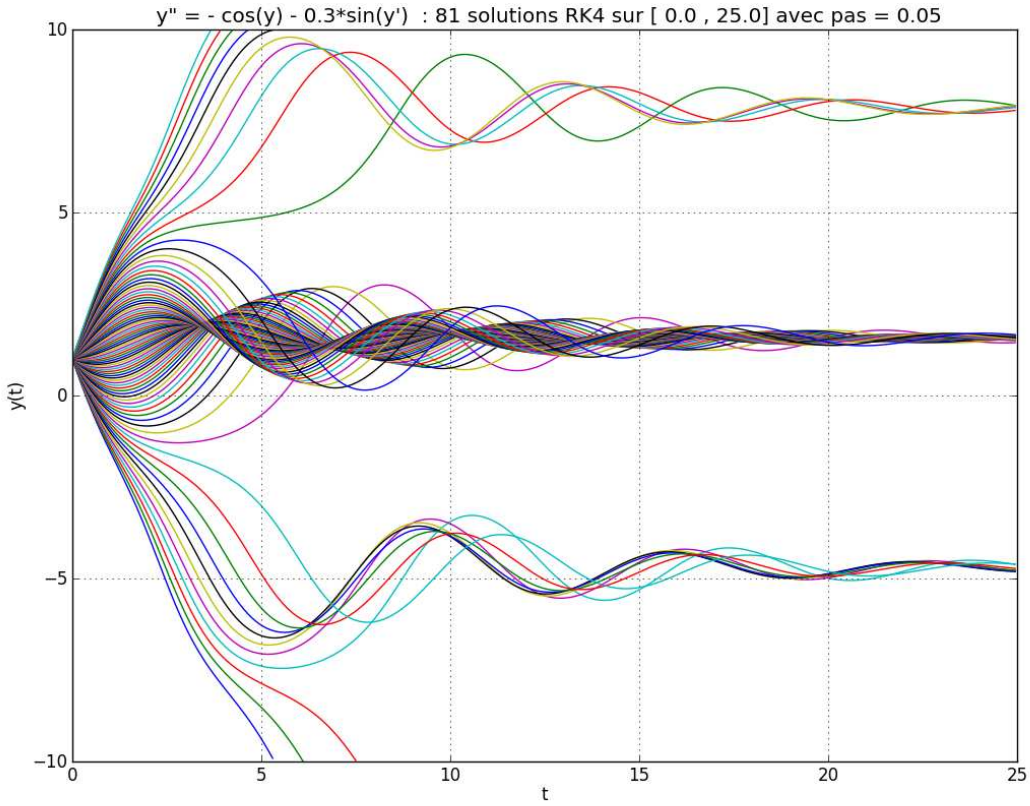
TP-cours : Résolution numérique des EDO - Méthodes et exemples en tous genres



Sommaire

1 Principe général des algorithmes numériques de résolution des EDO

1.1 Position du problème - Schéma numérique explicite

NB : dans ce document, t désignera systématiquement la variable de toute fonction. Cela ne restreint pas pour autant ce cours à l'étude de la résolution de problèmes dynamiques du temps, la transposition à des problèmes spatiaux étant immédiate.

On souhaite résoudre par voie numérique une équation différentielle du premier ordre dans un premier temps. On considère le problème de Cauchy sous sa forme la plus générale :

$$\text{soit} \quad \left[\begin{array}{l} \dfrac{dv(t)}{dt} = \dot{v}(t) = f[t, v(t)] \\ \\ v(t_0) = v_0 \end{array} \right.$$

On cherche à calculer les valeurs approchées de $v(t)$ pour t donné. Le principe est le suivant :

- On fait évoluer le temps par incrément discret avec une récurrence simple $t_{n+1} = t_n + h$, h étant le pas fixe d'incrément que l'on fixe dans la subdivision uniforme de l'intervalle $[t_0; t_f]$.
- Dans chaque intervalle de $[t_n, t_{n+1}]$, on calcule une valeur approchée $\tilde{v}_{n+1}$ de $v(t_{n+1})$ en t_{n+1} en posant le schéma numérique suivant :

$$\Rightarrow \quad \boxed{\left[\begin{array}{ll} \tilde{v}_{n+1} = \tilde{v}_n + h \cdot \phi(t_n, \tilde{v}_n, h) & \text{avec } n \in [0, N-1] \\ \tilde{v}_0 = v_0 \end{array} \right.}$$

avec la fonction $\phi(t_n, \tilde{v}_n, h)$ appelée **fonction d'incrément** que l'on déterminera dans les deux méthodes explorées plus bas : méthodes d'Euler (MPSI) et de Runge-Kutta (hors programme).

1.2 Définition des conditions de consistance et de stabilité

Définition - (1.2) - 1:

Le schéma numérique est dit **consistant** *avec l'équation différentielle (e) si pour toute fonction v, solution de l'équation, on vérifie la limite suivante :*

$$lim_{\substack{h \to 0 \\ 0 \leq n \leq N}} \left| \frac{v(t_{n+1}) - v(t_n)}{h} - \phi(t_n, v(t_n), h) \right| = lim_{\substack{h \to 0 \\ 0 \leq n \leq N}} [\eta_n(h)] = 0$$

en posant $\eta_n(h) = \dfrac{v(t_{n+1}) - v(t_n)}{h} - \phi(t_n, v(t_n), h)$ *l'erreur locale (car en n) de troncature*

$$\boxed{\eta_n(h) \text{ doit être en } \mathcal{O}(h) \text{ pour assurer la consistance.}}$$

De même on définit le résidu (erreur locale sur $v(t)$) par $\epsilon_n = h\eta_n(h) = v(t_{n+1}) - v(t_n) - h\phi(t_n, v(t_n), h))$; il doit être en $\mathcal{O}(h)$ pour assurer la consistance

Définition - (1.2) - 2:

Le schéma numérique est dit **stable** *si pour h fixé on a :*

$$\lim_{n \to \infty} \tilde{v}_n \neq \pm\infty$$

2 Méthodes d'Euler

2.1 Méthode d'Euler explicite

a - Principe et schéma

Dans cette méthode, on élabore la fonction d'incrément $\phi(t_n, \tilde{v}(t_n), h)$ en développant au premier ordre $v(t_n+h) = v(t_{n+1})$ autour de t_n :

$$v(t_n + h) = v(t_{n+1}) = v(t_n) + h\frac{dv}{dt}(t_n) + \mathcal{O}(h^2)$$

soit :

$$\frac{dv}{dt}(t_n) = \frac{v(t_{n+1}) - v(t_n)}{h} + \mathcal{O}(h)$$

L'approximation de la méthode d'Euler consiste :
- à négliger le terme en $\mathcal{O}(h)$ (approximation de premier ordre) :
- à assimiler $v(t_n)$ à son approchant $\tilde{v}_n$ (à la même date)

$$\frac{dv}{dt}(t_n) \simeq \frac{v(t_{n+1}) - v(t_n)}{h} \simeq \frac{\tilde{v}_{n+1} - \tilde{v}_n}{h}$$

soit :

$$\tilde{v}_{n+1} \simeq \tilde{v}_n + h \cdot \frac{dv}{dt}(t_n)$$

ainsi on dégage la fonction d'incrément dans le modèle d'Euler avec : $\frac{dv}{dt}(t_n) \simeq f(t_n, \tilde{v}_n) = \phi(t_n, \tilde{v}_n, h)$
Il en découle **le schéma numérique d'Euler** :

$$\Rightarrow \quad \boxed{\begin{cases} \tilde{v}_{n+1} = \tilde{v}_n + h \cdot f(t_n, \tilde{v}_n) & \text{avec } n \in [0, N-1] \\ \tilde{v}_0 = v_0 \end{cases}}$$

b - Consistance

Dans le cas du schéma d'Euler, l'erreur locale $\eta_n(h)$ est :

$$\eta_n(h) = \frac{v(t_{n+1}) - v(t_n)}{h} - f(t_n, v(t_n))$$

soit :

$$\eta_n(h) = \frac{dv}{dt}(t_n) + \mathcal{O}(h) - f(t_n, v(t_n)) \qquad \text{or } \frac{dv}{dt}(t_n) = f[t_n, v(t_n)]$$

donc :

$$\boxed{\eta_n(h) = \mathcal{O}(h) \quad \Rightarrow \quad \text{le schéma d'Euler est } \textbf{consistant}}$$

c - Exemples de résolution dans le cas d'un schéma d'Euler stable

■ PREMIER EXEMPLE SIMPLE STABLE : CHUTE FREINÉE

On souhaite intégrer l'équation différentielle régissant la vitesse v d'un mobile en chute dans le champ de pesanteur terrestre et freiné par un frottement visqueux $f_f = -\alpha \cdot v$:

$$m\frac{dv}{dt} = -\alpha v + mg$$

avec g l'intensité de la pesanteur et α est une grandeur dimensionnée quantifiant l'intensité de la force de frottement.
Cette équation devient, en posant les grandeurs dimensionnées $\tau = \dfrac{m}{\alpha}$ (temps caractéristique du système) et $v_l = \dfrac{mg}{\alpha}$ (vitesse limite) : $\dfrac{dv}{dt} + \dfrac{v}{\tau} = \dfrac{v_l}{\tau}$

On peut adimensionner cette équation en posant les changements de variable et de fonction suivants : $\begin{cases} t' = \dfrac{t}{\tau} \\ u = \dfrac{v_l}{v} \end{cases}$

Ainsi l'équation devient en renommant t' en t : $\dfrac{du}{dt} + u = 1$ soit : $\boxed{\dfrac{du}{dt} = 1 - u}$

La fonction d'incrément est donc $\phi[t, u(t)] = 1 - u(t)$

La solution de cette équation étant connue[1], on pourra constater graphiquement la précision de la solution approchée par méthode numérique.

EXERCICE N°1: **Résolution par méthode d'Euler explicite**

Compléter le code `Chute_freinee_Euler_explicite.py` *pour réaliser l'intégration numérique et le tracé de la vitesse d'évolution du mobile lors de sa chute freinée ; on ajoutera sur le même graphique le tracé de la solution analytique afin d'évaluer la qualité de l'intégration.*

■ SECOND EXEMPLE STABLE : PROBLÈME PROIES-PRÉDATEURS PAR MODÈLE DE LOTKA-VOLTERRA - SYSTÈME DIFFÉRENTIEL

▶ **Résolution numérique par représentation scalaire**

On propose ici la résolution numérique du problème PROIES-PRÉDATEURS dont le modèle fut introduit par Lotka et Volterra entre 1925 et 1931. Il s'agit d'un système de deux équations différentielles non linéaires couplées, dont on donne brièvement le principe de construction :
On appelle $x(t)$ la population des proies et $y(t)$ celle des prédateurs. En l'absence de prédateur, la population de proies croît de manière exponentielle ; son évolution est donc régie par une équation linéaire du premier ordre :

$$\frac{\dot{x(t)}}{x(t)} = a > 0$$

De même, en l'absence de nourriture, la population de prédateurs, notée $y(t)$ décroît également selon une loi de premier ordre :

$$\frac{\dot{y(t)}}{y(t)} = -c < 0$$

1. $u(t) = 1 - e^{-t}$ avec la CI u(0)=0

Si maintenant proies et prédateurs interagissent, le taux de variation de la population des proies doit diminuer proportionnellement à la population des prédateurs ($-b < 0$), et inversement, celui de la population des prédateurs doit croître proportionnellement à la population des proies (taux de conversion ; on traduit cela de la manière suivante à l'aide des coefficients constants $(a, b, c, d) > 0$:

$$\begin{cases} \dfrac{\dot{x(t)}}{x(t)} = a - b \cdot y(t) \\ \dfrac{\dot{y(t)}}{y(t)} = -c + d \cdot x(t) \end{cases}$$

Ce qui donne le système différentiel de Lotka-Voltera :

$$\boxed{\begin{cases} \dot{x}(t) = a \cdot x(t) - b \cdot y(t) \cdot x(t) \\ \dot{y}(t) = -c \cdot y(t) + d \cdot x(t) \cdot y(t) \end{cases}}$$

NB : ce système est non linéaire et sa résolution numérique se fait à l'aide des conditions initiales : $x(t = 0)$ et $y(t = 0)$

EXERCICE N°2: **Résolution par la méthode d'Euler**

❶ *Compléter dans le code* `Lotka_Volterra_Euler.py` *les fonctions Python* `Volterra(T,h,x0,y0,a,b,c,d,fx,fy)`, `fx((a,b,c,d,x,y))` *et* `fy((a,b,c,d,x,y))` *avec :*

$$\begin{cases} T \text{ : durée de l'expérience} \\ h \text{ : pas d'intégration} \\ a, b, c, d \text{ : les coefficients du modèle de Lotka-Volterra} \\ fx \text{ et } fy \text{ : deux fonctions de } x(t) \text{ et } y(t) \text{ renvoyant respectivement les expressions des taux} \\ \qquad \text{de variations } \dot{x}(t) \text{ et } \dot{y}(t) \text{ en fonction de } x(t) \text{ et } y(t) \text{ (c'est le système différentiel !).} \end{cases}$$

❷ *Tester cette méthode pour* $T = 50, h = 0.01, x0 = 5, y0 = 1, a = 0.6, b = 0.8, c = 0.6, d = 0.3$. *Faire tracer sur un même graphique les évolutions des populations de proies et de prédateurs. Tracer également le portrait de phase de la solution.*

Si tout s'est bien passé dans la résolution de cet exercice, vous devriez obtenir les courbes suivantes :

▶ Résolution numérique par représentation vectorielle

Il est également possible de résoudre des systèmes différentiels par méthode d'Euler en exploitant des vecteurs, et non des fonctions scalaires. Cela assure une présentation claire et compacte du problème ; par exemple ici, pour la résolution du système de Lotka-Volterra, on ne définit plus qu'une seule fonction vectorielle `F` contenant donc les deux taux de variation des espèces proies et prédateurs.

Ainsi, le schéma différentiel prend alors la forme suivante : soit $\begin{cases} \dot{XY}(t) = F[t, XY(t)] \\ XY(0) = XY_0 \end{cases}$

avec : $XY(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$, $F[t, XY(t)] = \begin{bmatrix} a \times x(t) - b \times x(t) \times y(t) \\ -c \times y(t) + d \times x(t) \times y(t) \end{bmatrix}$ et $XY_0 = \begin{bmatrix} x(0) \\ y(0) \end{bmatrix}$

Ce qui permet de poser le schéma numérique d'Euler suivant :

$$\Rightarrow \boxed{\begin{cases} \tilde{XY}_{n+1} = \tilde{XY}_n + h \cdot F(t_n, \tilde{XY}_n) \qquad \text{avec } n \in [0, N-1] \\ \tilde{XY}_0 = XY_0 \end{cases}}$$

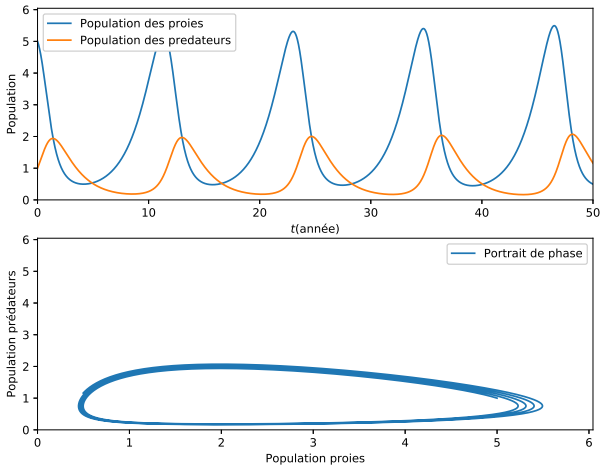


FIGURE IX.1 – Evolution des populations de proies et prédateurs dans le modèle de Lotka-Volterra.

On pourra par exemple structurer le vecteur solution $\tilde{XY}$ de la manière suivante :

$$\left[\begin{array}{l}\text{la première colonne contient la population des proies } \tilde{x}_n \\ \text{la seconde colonne contient la population des prédateurs } \tilde{y}_n\end{array}\right.$$

soit :

$$XY = \begin{bmatrix} XY[0,0] = x0 & XY[0,1] = y0 \\ XY[1,0] & XY[1,1] \\ XY[2,0] & XY[2,1] \\ \vdots & \vdots \end{bmatrix}$$

EXERCICE N°3: **Implémentation vectorielle de la méthode d'Euler**

Compléter le code `Lotka_Volterra_Euler_vectoriel.py` *afin de réaliser l'intégration du problème proies-prédateurs par méthode d'Euler vectorielle.*

d - Exemples de résolution dans le cas d'un schéma d'Euler instable : l'oscillateur harmonique

On considère un problème physique à un degré de liberté x dont l'évolution temporelle est régie par une équation harmonique (système-masse ressort, pendule plan en approximation des petits angles, etc....) :

$$\frac{d^2x(t)}{dt^2} + \omega^2 \cdot x(t) = 0 \qquad (e)$$

On peut facilement ramener cette équation du second ordre à un système de deux équations du premier ordre en introduisant la nouvelle variable v :

$$(e) \quad \Leftrightarrow \quad \left[\begin{array}{l} v(t) = \dfrac{dx(t)}{dt} \\ \dfrac{dv(t)}{dt} = -\omega^2 \cdot x(t) \end{array}\right.$$

Le schéma numérique correspondant est immédiatement déduit :

$$\left[\begin{array}{l}\tilde{x}_{n+1}=\tilde{x}_n+h\cdot\tilde{v}_n\\ \tilde{v}_{n+1}=\tilde{v}_n-h\cdot\omega^2\tilde{x}_n\end{array}\right.$$

ETUDE DE LA STABILITÉ DU SCHÉMA NUMÉRIQUE :

On peut réécrire le schéma numérique sous forme matricielle avec :

$$\underbrace{\begin{pmatrix}\tilde{x}_{n+1}\\ \tilde{v}_{n+1}\end{pmatrix}}_{=X_{n+1}}=\underbrace{\begin{pmatrix}1 & h\\ -h\cdot\omega^2 & 1\end{pmatrix}}_{=A}\cdot\underbrace{\begin{pmatrix}\tilde{x}_n\\ \tilde{v}_n\end{pmatrix}}_{=X_n}$$ avec A appelé **matrice d'amplification** du schéma

soit :

$$X_{n+1}=A\cdot X_n$$

Ainsi l'erreur sur la $n^{\text{ième}}$ récurrence $e_{n+1}=X_{n+1}-X_n$ peut s'écrire :

$$\boxed{e_{n+1}=A\cdot X_n-A\cdot X_{n-1}=A\cdot e_n}$$

En appelant P la matrice de passage dans la base propre du système différentiel, et $D=\begin{pmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{pmatrix}$ la matrice diagonale des valeurs propres λ_1 et λ_2 de A, on a $A=PDP^{-1}$, et l'équation précédente devient :

$$\begin{pmatrix}e_{\tilde{x}_{n+1}}\\ e_{\tilde{v}_{n+1}}\end{pmatrix}=PDP^{-1}\cdot\begin{pmatrix}e_{\tilde{x}_n}\\ e_{\tilde{v}_n}\end{pmatrix}$$

soit en multipliant l'équation par la matrice P^{-1} :

$$\underbrace{P^{-1}\begin{pmatrix}e_{\tilde{x}_{n+1}}\\ e_{\tilde{v}_{n+1}}\end{pmatrix}}_{=E'_{X_{n+1}}\text{(vecteur erreur en base propre)}}=D\underbrace{P^{-1}\cdot\begin{pmatrix}e_{\tilde{x}_n}\\ e_{\tilde{v}_n}\end{pmatrix}}_{=E'_{X_n}\text{(vecteur erreur en base propre)}}$$

Soit explicitement : $\boxed{\begin{pmatrix}e'_{\tilde{x}_{n+1}}\\ e'_{\tilde{v}_{n+1}}\end{pmatrix}=\begin{pmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{pmatrix}\cdot\begin{pmatrix}e'_{\tilde{x}_n}\\ e'_{\tilde{v}_n}\end{pmatrix}}$

PROPRIÉTÉ - (2.1) - 1:

La stabilité du schéma numérique est assurée lorsque la récurrence **ne provoque pas d'amplification de l'erreur**. *Donc, dans la base propre :*
Si $V=Max[|\lambda_1|,|\lambda_2|]$, *alors le schéma est :*

- *incondtionnellement instable si* $\forall h$ *on a* $V>1$
- *incondtionnellement stable si* $\forall h$ *on a* $V<1$
- *conditionnellement stable si pour* $h<h_{max}$ *on a* $V<1$

EXERCICE N°4: **Instabilité du schéma numérique d'Euler**

❶ *Montrer que le schéma numérique d'Euler est instable dans la résolution numérique de l'oscillateur harmonique.*

❷ *Compléter le code* `Oscillateur_harmonique_Euler_explicite.py`, *et constater graphiquement l'instabilité du schéma numérique. Si tout se passe bien, vous devriez obtenir le tracé ci-dessous :*

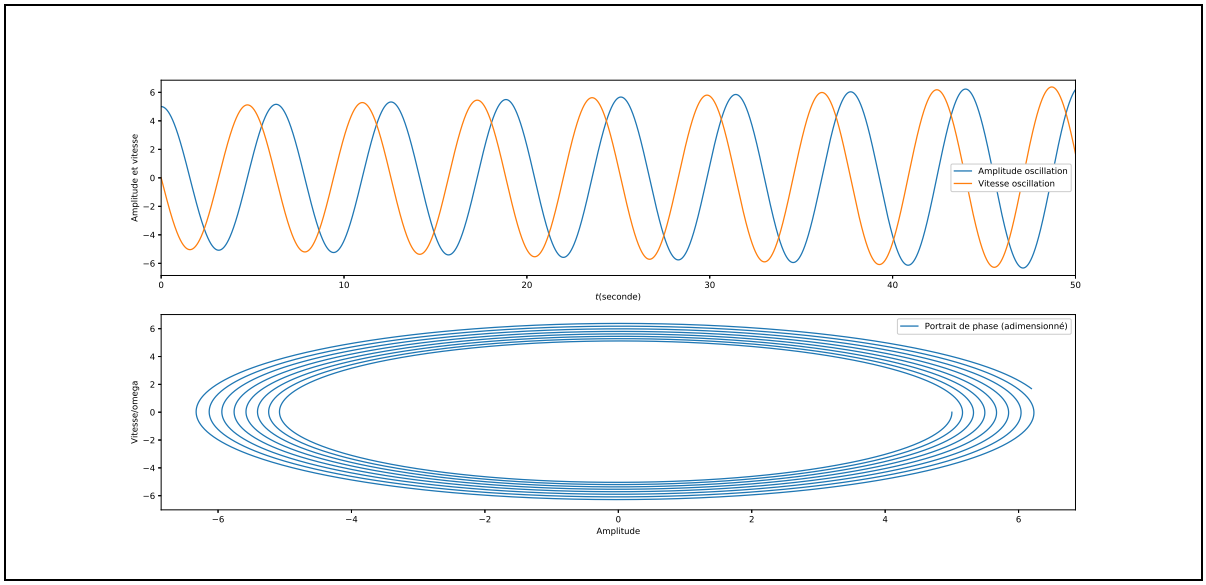


FIGURE IX.2 – Résolution de l'oscillateur harmonique par méthode d'Euler explicite : instable!!!

2.2 Quelques mots sur la méthode d'Euler implicite (ou rétrograde)

a - Principe

Supposons toujours à résoudre le problème de Cauchy, soit :

$$\left[\begin{array}{l} \dfrac{dv(t)}{dt} = \dot{v}(t) = f[t, v(t)] \\ \\ v(t_0) = v_0 \end{array}\right.$$

La méthode d'Euler dite *implicite*, ou *rétrograde* consiste à poser le schéma numérique de la façon suivante :

$$\tilde{v}_{n+1} = \tilde{v}_n + h \cdot f(t_{n+1}, \tilde{v}_{n+1})$$

NB : par rapport au schéma d'Euler explicite, on exprime ici la fonction d'incrément par un développement d'ordre 1 à la date t_{n+1}.

Cette méthode est appelée implicite puisqu'elle nécessite le calcul de v_{n+1} On peut facilement réaliser ceci à l'aide de la méthode de Newton.Posons la fonction F suivante :

$$\begin{array}{rcccl} F & : & \mathbb{R} & \to & \mathbb{R} \\ & & x & \mapsto & F(x) = \tilde{v}_n + hf(t_{n+1}, x) - x \end{array}$$

D'après le schéma d'Euler rétrograde, $\tilde{v}_{n+1}$ vérifie l'équation :

$$\boxed{F(\tilde{v}_{n+1}) = \tilde{v}_n + hf(t_{n+1}, \tilde{v}_{n+1}) - \tilde{v}_{n+1} = 0}$$

Ainsi, on dégage la solution par la recherche du *zéro* d'une fonction. On peut par exemple réaliser cela à l'aide de la méthode de Newton revue en chapitre de révisions n°4.

On rappelle que le principe est de calculer les candidats pour une solution approchée de $F(x) = 0$ par la relation de récurrence suivante :

$$x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)}$$

jusqu'à ce que $F(x_n) \leq err$ avec err tolérance (ou erreur) donnée.

NB : la méthode implicite implique donc de lancer la résolution d'une équation algébrique à itération du schéma d'Euler, ce qui en fait une méthode particulièrement coûteuse en temps machine. En revanche, elle présente souvent un caractère plus stable que la méthode explicite (cf exercice de l'oscillateur harmonique plus bas).

b - Exemples de mise en oeuvre

▶ Chute freinée :

On reprend ici la résolution de l'équation adimensionnée de la chute freinée vue en 2.1c) :

$$\frac{du}{dt} + u = 1 \quad \text{soit :} \quad \boxed{\frac{du}{dt} = 1 - u}$$

Code :

Listing IX.1 –

```
###### Méthode d'Euler implicite ######
def Euler_implicite(T,h,t0,v0,f):
    N=int(T/h)
    TPS=[t0]
    V=[v0]
    for i in range(N):
        TPS.append(TPS[i]+h) #incrément du temps
        vn1=Newton(h,V[i],f) #calcule de la valeur v(n+1)
        V.append(V[i]+h*f(vn1))
    return TPS,V

###### Fonctions associées à la résolution #####
def F(h,v,vn,f):
    return vn+h*f(v)-v
def Newton(h,vn,f):
    err=1.0
    eps=1e-9
    v=vn+h*f(vn) #initialisation de v(n+1) à une valeur "type" méthode explicite
    while err>eps:
        dF=(F(h,v+h,vn,f)-F(h,v,vn,f))/h #calcul de la dérivée au point v0 (nouveau candidat)
        v1=v-F(h,v,vn,f)/dF
        err=abs(v1-v)
        v=v1
    return v1
###### Lancement calcul et tracé #####
T,h,t0,v0=3.0,0.1,0.0,0.0
def f(u):
    return 1-u
TPS,V=Euler_implicite(T,h,t0,v0,f)
plt.plot(TPS,V,'--',label=u"v(t) par méthode Euler implicite", linewidth=2)
plt.xlabel('temps')
plt.ylabel('vitesse')
plt.legend(loc=2)
plt.show()
```

▶ Oscillateur harmonique :

Les schémas numériques implicites, bien que plus lourds à mettre en oeuvre, sont souvent plus stables que leurs homologues explicites.

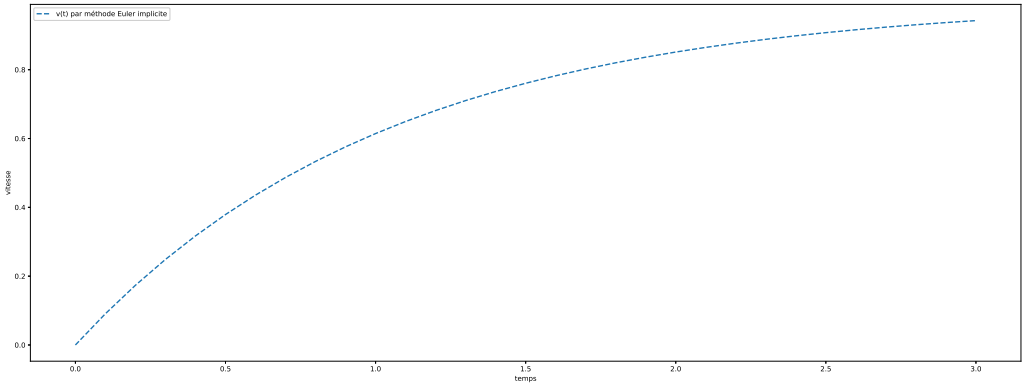


FIGURE IX.3 – Résolution de la chute freinée par méthode d'Euler implicite

L'exercice qui suit propose d'illustrer cela en étudiant l'application de la méthode d'Euler implicite au cas de l'oscillateur harmonique, dont on a vu que la résolution par schéma explicite conduisait à une instabilité.

EXERCICE N°5: **Schéma d'Euler implicite pour l'oscillateur harmonique**

❶ *Donner le schéma numérique d'Euler implicite de l'oscillateur harmonique donnant* x_{n+1} *et* v_{n+1}.

❷ *Montrer que le calcul de* x_{n+1} *et* v_{n+1} *nécessite de rechercher (par méthode de Newton donc) les racines de la fontion :* $F(x) = x - h^2 f(x) - (x_n + h v_n)$

❸ *Déterminer la matrice d'amplification du schéma numérique, et montrer que celui-ci est* **inconditionnellement stable**.

❹ *Compléter* **sur machine**, *le code suivant. L'exécuter et commenter.*

Listing IX.2 –

```
##### Méthode d'Euler implicite #####
def Euler_implicite(T,h,x0,v0,f): #def Oscill_Euler_expl(T,h,x0,v0,omega):
    N=int(T/h)
    TPS=[0]
    X=[x0]
    V=[v0]
    for i in range(N):
        TPS.append(TPS[i]+h) #incrément du temps
        xn1,vn1=Newton(.......) #calcule des valeurs x(n+1) et v(n+1)
        X.append(xn1)
        V.append(vn1)
    return TPS,X,V

## Définition de la fonction F pour le calcul implicite ##
def F(h,x,xn,vn,f):
    return ................
#### Définition de la fonction f ####
def f(u):
    return ................

##### Méthode de Newton #####
def Newton(h,xn,vn,f):
    err=1.0
    eps=1e-9
```

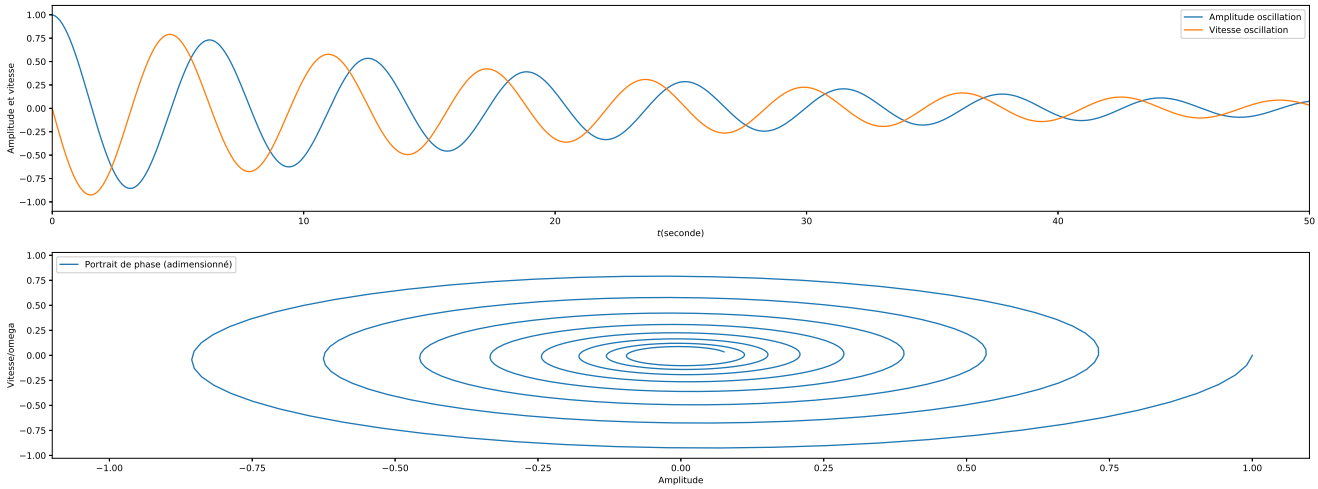


FIGURE IX.4 – Résolution de l'oscillateur harmonique par méthode d'Euler implicite.

```
    x=xn #initialisation de x(n+1) à une valeur "type" méthode explicite (il faut bien choisir une valeur!)
    while err>eps:
        dF=(F(h,x+h,xn,vn,f)-F(h,x,xn,vn,f))/h #calcul de la dérivée au point en cours x
        x1=x-F(h,x,xn,vn,f)/dF #calcul du nouveau candidat x1
        err=abs(x1-x)
        x=x1 #mise à jour de la valeur de x
    v1 = ..... #calcul de la nouvelle valeur v(n+1) pour la vitesse
    return x1,v1

###### Lancement######
T,h,t0,x0,v0,omega=50,0.1,0.0,1.0,0.0,1
TPS,X,V=Euler_implicite(T,h,x0,v0,f)
```

Une fois le code complété (et avec les commandes de tracé fournies dans le programme), vous devriez obtenir le tracé suivant :

3 Exemple d'amélioration du schéma numérique : la méthode de Runge-Kutta explicite d'ordre 4

3.1 Améliorer Euler !

Le schéma numérique un peu simpliste de la méthode d'Euler, qui est une approximation de premier ordre, entraîne imprécision et parfois comme nous l'avons vu, **une instabilité**.

$\Longrightarrow$ UNE IDÉE SIMPLE CONSISTE À AMÉLIORER LA FONCTION D'INCRÉMENT $\phi(t_n, v_n, h)$ AFIN QU'ELLE "ÉPOUSE" AU MIEUX LA SOLUTION VRAIE.

RAPPEL : pour le problème de Cauchy :

$$\text{soit} \quad \left[\begin{array}{l} \dfrac{dv(t)}{dt} = \dot{v}(t) = f[t, v(t)] \\ \\ v(t_0) = v_0 \end{array} \right.$$

la solution exacte est :

$$v(t+dt) = v(t) + \int_t^{t+dt} f[t, v(t)] \cdot dt = v(t) + dt \cdot D(t, v(t)$$

en posant **la fonction d'incrément "exacte"** $D(t, v(t))$ qui s'écrit :

$$D(t, v(t)) = \frac{\int_t^{t+dt} f[t, v(t)] \cdot dt}{dt}$$

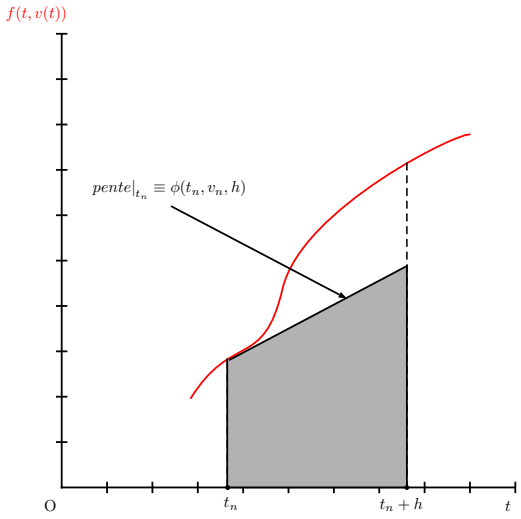
et qui correspond à l'aire sous la courbe $f(t, v(t))$ dans l'intervalle $[t, t+dt]$ divisée par dt.

Nous avons vu en 2.1.a) que la fonction d'incrément approchée $\phi(t_n, v_n, h)$ employée dans la méthode d'Euler était simplement la dérivée à la date t_n avec :

$$\tilde{v}_{n+1} = \tilde{v}_n + h \cdot \phi(t_n, \tilde{v}_n, h) \qquad \text{avec} \qquad \phi(t_n, \tilde{v}_n, h) = \frac{dv}{dt}(t_n)$$

Pour résumer, la résolution numérique du problème de Cauchy se voit réduite à la meilleure approche du calcul numérique d'une aire, et donc la recherche de **la meilleure fonction d'incrément $\phi(t_n, \tilde{v}_n, h)$ possible**.

Diverses méthodes permettant de se rapprocher au mieux de la fonction idéale d'incrément $D(t, v(t))$ ont été imaginées :



- Méthode d'Euler améliorée : on évalue l'aire en prenant la pente au "point milieu" $\left(t_n + \frac{h}{2}, f(t_n + \frac{h}{2})\right)$ et non en $(t_n, f(t_n))$
- **Méthode de Runge-Kutta d'ordre 4 (RK4)** objet du paragraphe suivant.

3.2 Principe des méthodes de Runge-Kutta explicites - Schéma numérique et représentation en tableau de Butcher

La fonction d'incrément ϕ des méthodes de Runge-Kutta s'appuie sur les différentes méthodes numérique pour le calcul des aires sous les courbes. Elle consiste simplement à augmenter le nombre d'évaluations de la fonction $f(t, v(t))$ entre t_n et $t_n + h$.

Le schéma numérique de la méthode de Runge-Kutta à s étages est le suivant :

$$\left[\begin{array}{l} k_1 = f(t_n, y_n) \\ k_2 = f(t_n + c_2 h, y_n + h a_{21} k_1) \\ \vdots \\ k_s = f(t_n + c_s h, y_n + h \sum_{i=1}^{s-1} a_{si} k_i) \\ \tilde{v}_{n+1} = \tilde{v}_n + h \sum_{i=1}^{s} b_i k_i \end{array} \right.$$

où les coefficients c_i, a_{ij}, et b_i sont des constantes qui définissent précisément le schéma. On supposera toujours dans la suite que $c_1 = 0$, et $c_i = \sum_{j=1}^{i-1} a_{ij}$ pour $i = 2, \dots, s$.

On représente en général ce schéma par un tableau synthétique appélé tableau de Butcher :

c_1					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
$\vdots$	$\vdots$	$\vdots$			
c_s	a_{s1}	a_{s2}	$\dots$	s_{ss-1}	
	b_1	b_2	$\dots$	b_{s-1}	b_s

FIGURE IX.5 – Tableau de Butcher de la méthode de Runge-Kutta explicite à s étages

EXERCICE N°6: *Représenter le tableau de Butcher de la méthode d'Euler.*

La suite du cours propose d'exploiter la méthode de Runge-Kutta d'ordre 4 dont le tableau de Butcher est le suivant :

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{1}{6}$

NB : on montre que sous de bonnes hypothèses, ce schéma numérique est d'ordre 4, soit une erreur en $\mathcal{O}(h^4)$

3.3 Exemples d'applications de la méthode RK4

a - Encore la chute freinée

On reprend ici (une toute dernière fois !) la résolution numérique du problème de la chute freinée.

EXERCICE N°7: *Compléter le code* `Chute_freinee_RK4_explicite.py` *pour qu'il réalise la résolution et le tracé numériques de la chute freinée.*

b - L'oscillateur de Van Der Pol

SOURCE WIKIPEDIA :

L'oscillateur de van der Pol a été imaginé par le physicien néerlandais Balthasar van der Pol alors qu'il était employé par les laboratoires Philips. Van der Pol découvrit que ce circuit contenant un tube à vide développait des oscillations stables, qu'il appela « oscillation de relaxation » et que l'on désigne aujourd'hui plutôt comme des cycles limites des circuits électriques. Lorsque ces circuits sont excités à une fréquence proche de celle du cycle limite il se crée un couplage, c'est-à-dire que le signal de commande impose sa fréquence au courant. Van der Pol et son collègue van der Mark publièrent en 1927 qu'à certaines fréquences de commande, il apparaissait un bruit irrégulier. Ce bruit se déclenchait toujours au voisinage des fréquences naturelles de couplage. Ce fut l'une des premières mises en évidence de l'existence d'un chaos déterministe.

L'équation différentielle décrivant l'oscillateur de Van der Pol est non linéaire et dépend d'un paramètre $\varepsilon \neq 0$:

$$\boxed{\ddot{y_1}(t) - \epsilon \cdot \omega_0 \cdot (1 - y_1^2(t)) \cdot \dot{y_1}(t) + \omega_0^2 \cdot y_1(t) = 0}$$

NB : le cas $\epsilon = 0$ correspond à un oscillateur harmonique.

Le problème de Cauchy correspondant s'écrit donc sous forme d'un système de deux équations différentielles (idem oscillateur harmonique) :

$$\begin{cases} \dot{y_1}(t) = y_2(t) \\ \dot{y_2}(t) = +\varepsilon \cdot \omega_0 \left(1 - y_1^2(t)\right) \cdot y_2(t) - \omega_0^2 \cdot y_1(t) \end{cases}$$

La solution est de dimension 2 et nécessitera de calculer des 4 coefficients k_i ($i = \{1, 2, 3, 4\}$) pour chacune des deux fonctions y_1 et y_2. On présentera donc la solution sous forme d'un tableau Y dont la première colonne contiendra l'évaluation numérique de la fonction y_1 et la seconde, celle de la fonction $y_2 = \dot{y_1}$:

$$Y = \begin{bmatrix} \tilde{y}_{1_0} & \tilde{y}_{2_0} \\ \tilde{y}_{1_1} & \tilde{y}_{2_1} \\ \vdots & \vdots \\ \tilde{y}_{1_i} & \tilde{y}_{2_i} \\ \vdots & \vdots \\ \tilde{y}_{1_N} & \tilde{y}_{2_N} \end{bmatrix}$$

EXERCICE N°8: **Résolution numérique de l'oscillateur de Van der Pol**

❶ *Compléter le code* `Oscillateur_Van_der_Pol_RK4_explicite.py` *pour qu'il assure la résolution de l'oscillateur de Van der Pol par méthode RK4 (et le tracé).*

❷ *Modifier les conditions initiales ($Y[0, :]$), exécuter le code, et commenter le résultat.*

❸ *Modifier la valeur du paramètres ϵ, exécuter le code, et commenter le résultat.*

Si votre code est correct, vous devriez obtenir le tracé ci-dessous :

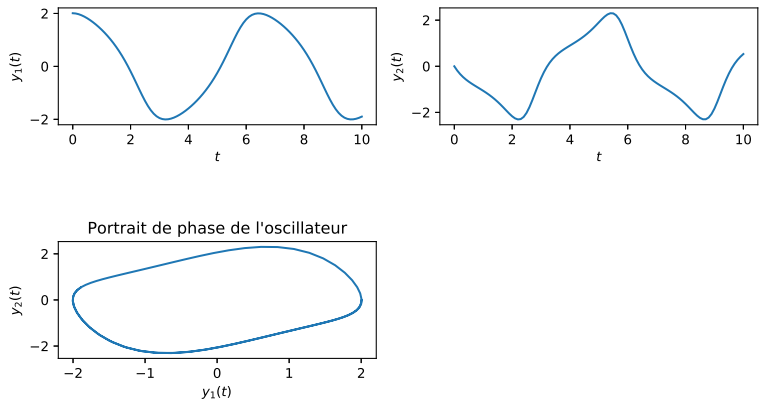


FIGURE IX.6 – Oscillations de Van der Pol et portrait de phase