In [1]:
```
%reset -f
from IPython.display import display, Latex
from IPython.core.display import HTML
css_file = './custom.css'
HTML(open(css_file, "r").read())
```

Out[1]:

Table of Contents

M62-Examen

Soit le schéma de Runge-Kutta dont la *matrice de Butcher* est

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \alpha & \alpha & 0 \\ \hline & \frac{2\alpha-1}{2\alpha} & \frac{1}{2\alpha} \end{array}$$

Chaque sujet correspond à une valeur de α choisie parmi les suivantes:

$$\left\{\frac{1}{6},\frac{1}{4},\frac{1}{3},\frac{5}{12},\frac{7}{12},\frac{2}{3},\frac{3}{4},\frac{5}{6},\frac{11}{12}\right\}$$

In [2]:
```
from fractions import Fraction
ALPHA=[Fraction(2,12),Fraction(3,12),Fraction(4,12),Fraction(5,12),
       Fraction(7,12),Fraction(8,12),Fraction(9,12),Fraction(10,12),Fraction(11,12)]
for alpha in ALPHA:
    print("c_2=a_{21}=",alpha,"\tb_1=",1-Fraction(1,2*alpha),"\tb_2=",Fraction(1,2*alpha))
```

```
c_2=a_{21}= 1/6         b_1= -2         b_2= 3
c_2=a_{21}= 1/4         b_1= -1         b_2= 2
c_2=a_{21}= 1/3         b_1= -1/2       b_2= 3/2
c_2=a_{21}= 5/12        b_1= -1/5       b_2= 6/5
c_2=a_{21}= 7/12        b_1= 1/7        b_2= 6/7
c_2=a_{21}= 2/3         b_1= 1/4        b_2= 3/4
c_2=a_{21}= 3/4         b_1= 1/3        b_2= 2/3
c_2=a_{21}= 5/6         b_1= 2/5        b_2= 3/5
c_2=a_{21}= 11/12       b_1= 5/11       b_2= 6/11
```

Exercice: écriture schéma

Écrire le schéma sous la forme d'une suite définie par récurrence.

Correction : écriture schéma

Considérons le problème de Cauchy

> trouver une fonction y: $I \subset \mathbb{R} \to \mathbb{R}$ définie sur un intervalle $I = [t_0, T]$ telle que
> $$\begin{cases} y'(t) = \varphi(t, y(t)), \quad \forall t \in I = [t_0, T], \\ y(t_0) = y_0, \end{cases}$$
> avec y_0 une valeur donnée et supposons que l'on ait montré l'existence et l'unicité d'une solution y pour $t \in I$.

Pour $h > 0$ soit $t_n \overset{\text{déf.}}{=} t_0 + nh$ avec $n = 0, 1, 2, \ldots, N$ une suite de $N + 1$ nœuds de I induisant une discrétisation de I en N sous-intervalles $I_n = [t_n; t_{n+1}]$ chacun de longueur $h > 0$ (appelé le *pas de discrétisation*).

Pour chaque nœud t_n, on cherche la valeur inconnue u_n qui approche la valeur exacte $y_n \equiv y(t_n)$.

- L'ensemble de $N + 1$ valeurs $\{t_0, t_1 = t_0 + h, \ldots, t_N = T\}$ représente les points de la discrétisation.
- L'ensemble de $N + 1$ valeurs $\{y_0, y_1, \ldots, y_N\}$ représente la solution exacte.
- L'ensemble de $N + 1$ valeurs $\{u_0 = y_0, u_1, \ldots, u_N\}$ représente la solution numérique. Cette solution approchée sera obtenue en construisant une suite récurrente.

Le schéma qu'on va construire permet de calculer explicitement u_{n+1} à partir de u_n et il est donc possible de calculer successivement u_1, u_2,..., en partant de u_0 par la formule de récurrence

$$\begin{cases} u_0 = y_0 \\ K_1 = \varphi\left(t_n, u_n\right) \\ K_2 = \varphi\left(t_n + \alpha h, u_n + \alpha h K_1\right) \\ u_{n+1} = u_n + \frac{h}{2\alpha}((2\alpha - 1)K_1 + K_2) \quad n = 0, 1, \ldots N - 1 \end{cases}$$

Exercice : étude de la A-stabilité

Étudier théoriquement la A-stabilité.

Correction : étude de la A-stabilité

Définition de A-stabilité

Soit $\beta > 0$ un nombre réel positif et considérons le problème de Cauchy

$$\begin{cases} y'(t) = -\beta y(t), \quad \text{pour } t > 0, \\ y(0) = y_0 \end{cases}$$

où $y_0 \neq 0$ est une valeur donnée. Sa solution est $y(t) = y_0 e^{-\beta t}$ donc

$$\lim_{t \to +\infty} y(t) = 0.$$

Si, sous d'éventuelles conditions sur h, on a

$$\lim_{n \to +\infty} u_n = 0,$$

alors on dit que **le schéma est A-stable.**

Le schéma donné appliqué à cette équation devient

In [3]:
```
import sympy as sym
sym.init_printing()
sym.var('h,Alpha,beta,u_n,u_np1')

Phi=lambda y : -beta*y

K_1=Phi(u_n)
display(K_1)
K_2=Phi(u_n+Alpha*h*K_1)
display(K_2.expand().factor(u_n))
display(sym.Eq(u_np1,(u_n+h/(2*Alpha)*((2*Alpha-1)*K_1+K_2  )).factor()))
```

$-\beta u_n$

$\beta u_n \left(A\beta h - 1\right)$

$u_{np1} = \frac{u_n}{2}\left(\beta^2 h^2 - 2\beta h + 2\right)$

c'est-à-dire

$$\begin{cases} u_0 = y_0 \\ u_{n+1} = \frac{2-2(\beta h)+(\beta h)^2}{2} u_n \quad n = 0, 1, \dots N-1 \end{cases}$$

Par induction on obtient

$$u_n = \left(1 - (\beta h) + \frac{1}{2}(\beta h)^2\right)^n u_0.$$

Notons que u_n ne dépend pas du choix de α.

Étant une suite géométrique, $\lim_{n \to +\infty} u_n = 0$ si et seulement si

$$\left|1 - (\beta h) + \frac{1}{2}(\beta h)^2\right| < 1.$$

Notons x le produit βh et q le polynôme $q(x) = \frac{1}{2}x^2 - x + 1$.

In [4]:
```
%matplotlib inline
import sympy as sym
from sympy.plotting import plot
sym.init_printing()
sym.var('x',positive=True)

q=1-1*x+x**2/2
print("|q(x)|<1 ssi", end=' ')
display(sym.solve(abs(q)<1))
print("q'(x)=")
dq=sym.diff(q,x)
display(dq)
print("q'(x)=0 ssi x=")
x_sommet=sym.solve(dq)[0]
display(x_sommet)
print("Comme q(x_sommet)=")
y_sommet=q.subs(x,x_sommet)
display(y_sommet)
print("donc q(x)>0 pour tout x et q(x)<1 ssi x<",x_sommet*2)
sym.plot(q,1,-1,(x,0,x_sommet*2));
```

|q(x)|<1 ssi

$x < 2$

q'(x)=

$x - 1$

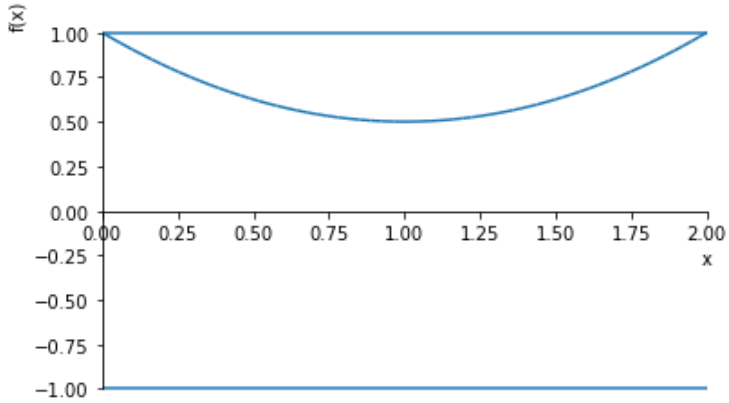
q'(x)=0 ssi x=

1

Comme q(x_sommet)=

$\frac{1}{2}$

donc q(x)>0 pour tout x et q(x)<1 ssi x< 2



Il s'agit de la parabole convexe de sommet $\left(1, \frac{1}{2}\right)$.
Nous avons $|q(x)| < 1$ si et seulement si $x < 2$. La relation $\lim_{n \to +\infty} u_n = 0$ est donc satisfaite si et seulement si

$$h < \frac{2}{\beta}.$$

Cette condition de stabilité limite le pas h d'avance en t lorsqu'on utilise le schéma.

Nous avons $q(x) > 0$ pour tout x donc cette convergence est monotone.

Exercice : implémentation et test

Implémenter le schéma et le tester avec le problème de Cauchy utilisé pour la A-stabilité sur l'intervalle $[0;1]$.
On choisira le nombre minimal de points pour avoir A-stabilité.

Chaque sujet correspond à une valeur de β choisie parmi les suivantes:

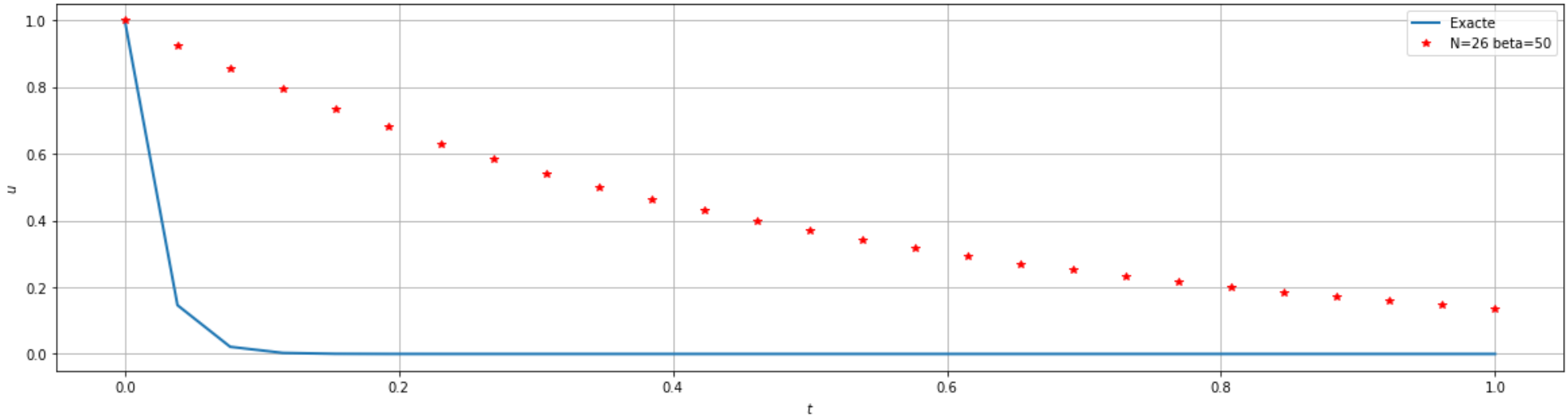
$$\{50, 60, 70, 80, 90, 100\}$$

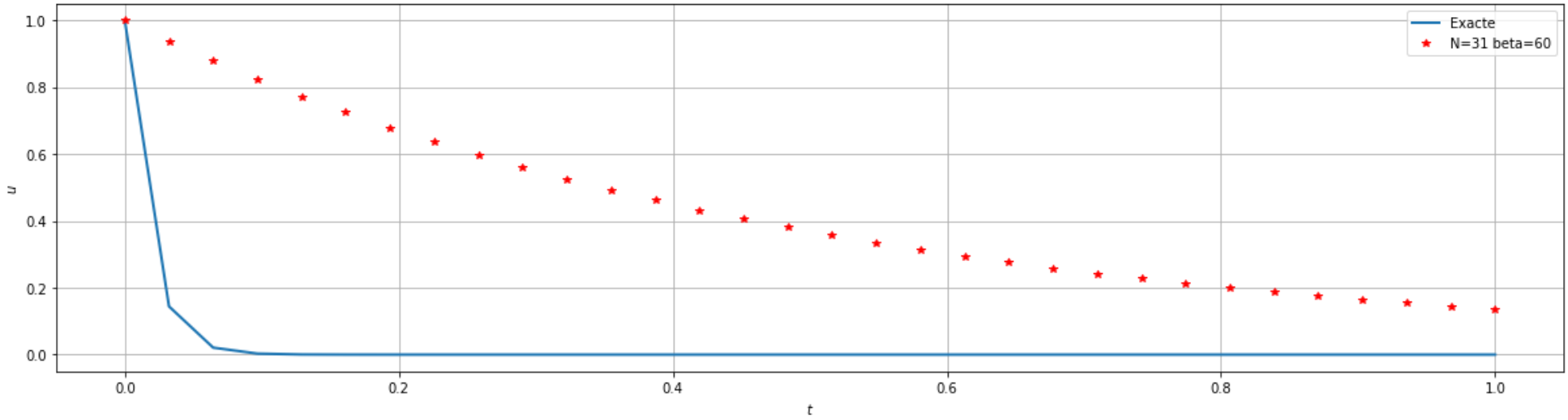
Correction : implémentation et test

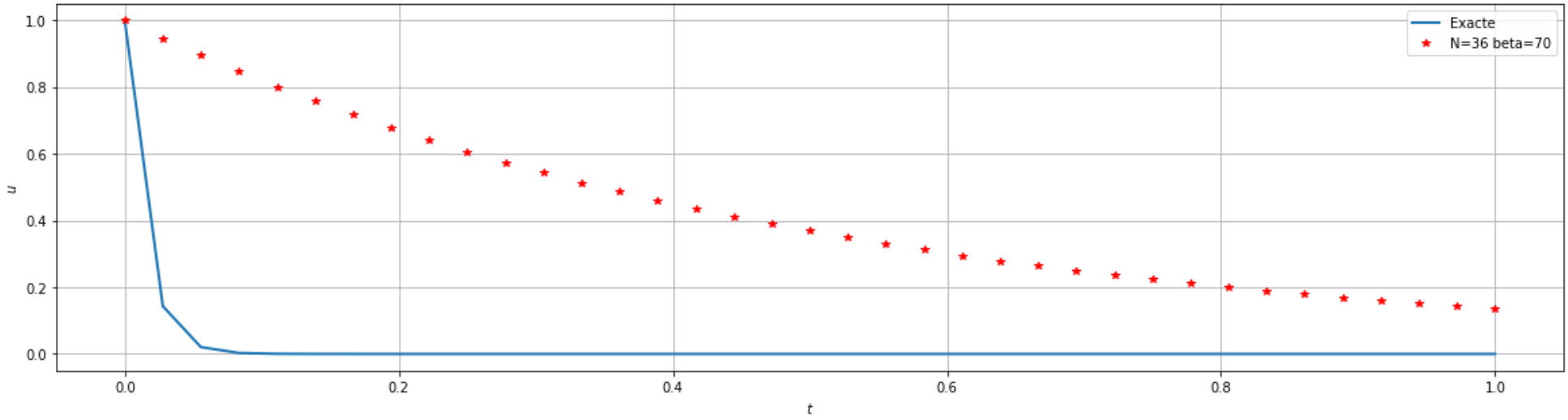
$h < \frac{2}{\beta}$ sur l'intervalle $[0;1]$ impose $N > \frac{\beta}{2}$.

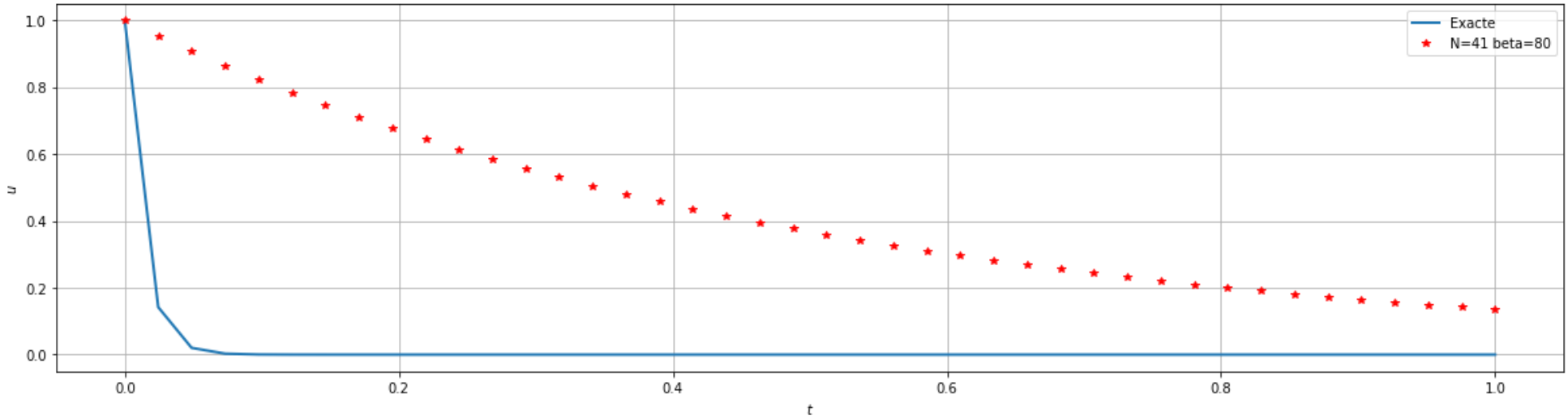
In [5]:
```
def RKalpha(phi,tt):
    uu = [y0]
    for i in range(len(tt)-1):
        k1 = phi( tt[i], uu[i] )
        k2 = phi( tt[i]+h*alpha , uu[i]+h*alpha*k1 )
        uu.append( uu[i] + h*((2*alpha-1)*k1+k2 )/(2*alpha))
    return uu
```

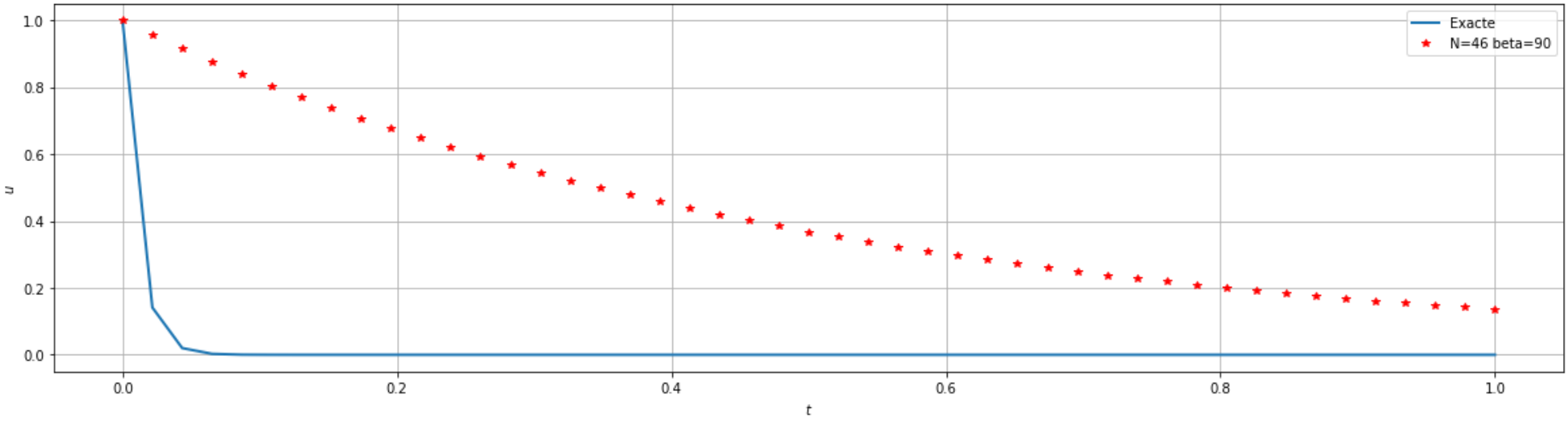
In [6]:
```
from matplotlib.pylab import *

BETA=arange(50,101,10)

t0, y0, tfinal = 0, 1, 1

sol_exacte = lambda t : y0*exp(-beta*t)
phi = lambda t,y : -beta*y

i=0


for beta in BETA:
    N=(int(beta/2)+1)
    tt = linspace(t0,tfinal,N+1)
    h = tt[1]-tt[0]
    yy = [sol_exacte(t) for t in tt]
    alpha=ALPHA[0]
    uu = RKalpha(phi,tt)
    i+=1
    figure(i,figsize=(20,5))
    plot(tt,yy,lw=2,label=("Exacte"))
    plot(tt,uu,'r*',label=("N="+str(N)+" beta="+str(beta)))
    xlabel('$t$')
    ylabel('$u$')
    legend()
    grid(True)
```

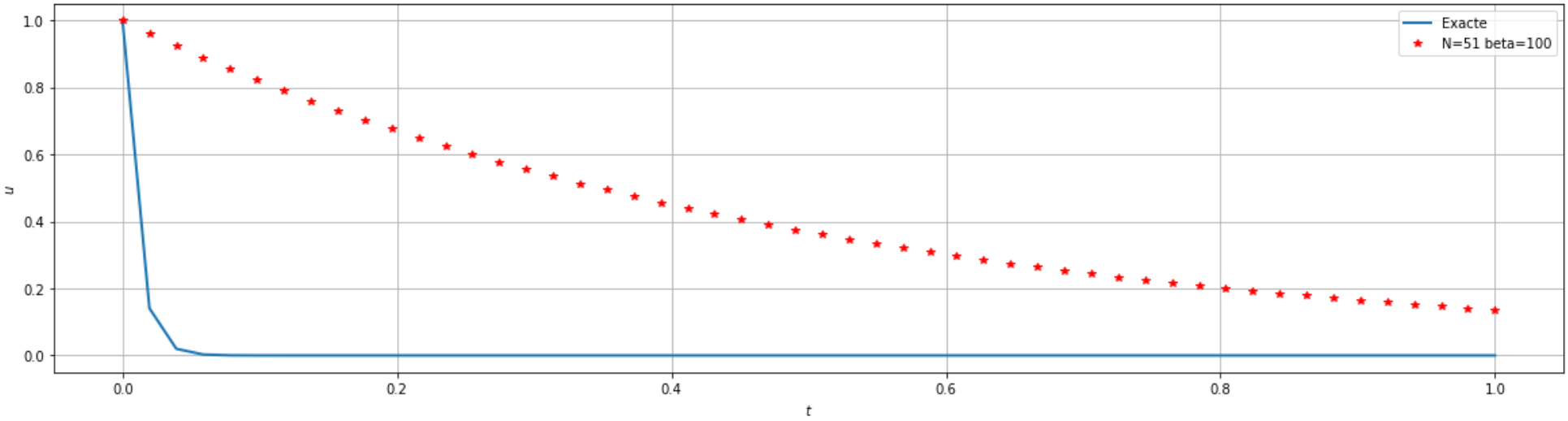

Exacte
N=26 beta=50
u
t
0.0
0.2
0.4
0.6
0.8
1.0


Exacte
N=31 beta=60
u
t
0.0
0.2
0.4
0.6
0.8
1.0


Exacte
N=36 beta=70
u
t
0.0
0.2
0.4
0.6
0.8
1.0


Exacte
N=41 beta=80
u
t
0.0
0.2
0.4
0.6
0.8
1.0





Exercice : étude de l'ordre

Étudier théoriquement l'ordre du schéma.

Correction : étude de l'ordre

- C'est un schéma explicite à deux étage: l'ordre est au plus 2.
- $c_1 = a_{11} + a_{12}$, $c_2 = a_{21} + a_{22}$ et $1 = b_1 + b_2 \implies$ il est consistante
- $c_1 b_1 + c_2 b_2 = \frac{1}{2} \implies$ il est d'ordre 2

In [7]:
```
for alpha in ALPHA:
    print("alpha=",alpha)
    c=[0,alpha]
    b=[1-1/(2*alpha),1/(2*alpha)]
    A=[[0,0],[alpha,0]]
    s=len(c)
    ordre=[]
    ordre.append(sum(b)==1)
    for i in range(s):
        ordre.append(sum(A[i])==c[i])
    ordre.append(sum([b[i]*c[i] for i in range(s)])==Fraction(1,2))
    print(set(ordre))
    print("---------------------")
```

```
alpha= 1/6
{True}
---------------------
alpha= 1/4
{True}
---------------------
alpha= 1/3
{True}
---------------------
alpha= 5/12
{True}
---------------------
alpha= 7/12
{True}
---------------------
alpha= 2/3
{True}
---------------------
alpha= 3/4
{True}
---------------------
alpha= 5/6
{True}
---------------------
alpha= 11/12
{True}
---------------------
```

Exercice : solution exacte

Calculer la solution exacte du problème de Cauchy

$$\begin{cases} y'(t) = -2t(y(t))^2, \quad t \in [0;3] \\ y(0) = 2 \end{cases}$$

Correction : solution exacte

On peut soit calculer la solution "à la main" soit utiliser `sympy`.

Il s'agit d'une EDO à variables séparables. La fonction $y(t) = 0$ pour tout t est solution de l'EDO mais elle ne vérifie pas la CI. Toute autre solution de l'EDO sera non nulle et se trouve formellement comme suit:

$$y'(t) = -2ty^2(t) \quad \Longrightarrow \quad \frac{y'(t)}{y^2(t)} = -2t \quad \Longrightarrow \quad \int y^{-2}\mathrm{d}y = -2\int t\mathrm{d}t \quad \Longrightarrow \quad y(t) = \frac{1}{t^2+c},\ c \in \mathbb{R}.$$

En imposant la CI on obtient $2 = 1/C$ d'où l'unique solution du problème de Cauchy: $y(t) = \frac{2}{2t^2+1}$.

In [8]:
```
import sympy as sym
sym.init_printing()
```

In [9]:
```
sym.var('t,C1')
u=sym.Function('u')
f=-2*t*(u(t))**2
edo=sym.Eq(sym.diff(u(t),t),f)
display(edo)
```

$$\frac{d}{dt}u(t) = -2tu^2(t)$$

In [10]:
```
solgen=sym.dsolve(edo,u(t))
display(solgen)
```

$$u(t) = \frac{1}{C_1 + t^2}$$

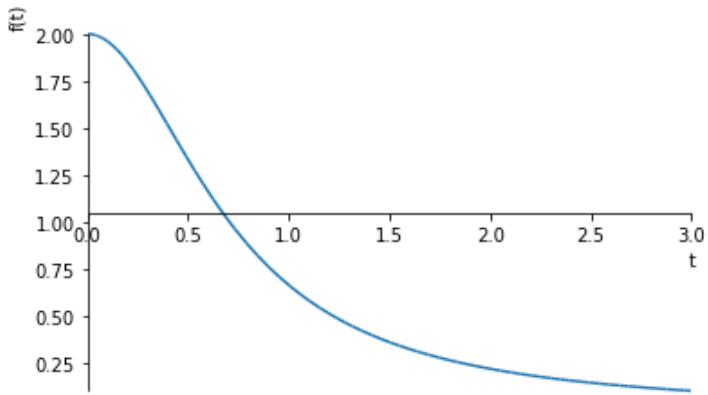
In [11]:
```
t0=0
u0=2
consts = sym.solve( [solgen.rhs.subs(t,t0)-u0 ], dict=True)[0]
display(consts)
```

$$\left\{C_1 : \frac{1}{2}\right\}$$

In [12]:
```
solpar=solgen.subs(consts)
display(solpar)
```

$$u(t) = \frac{1}{t^2 + \frac{1}{2}}$$

In [13]:
```
%matplotlib inline
sym.plot(solpar.rhs,(t,0,3));
```



Exercice : convergence

Vérifier empiriquement l'ordre de convergence en affichant la courbe de convergence et en estimant la pente sur ce problème de Cauchy.

Correction : étude empirique de l'ordre de convergence

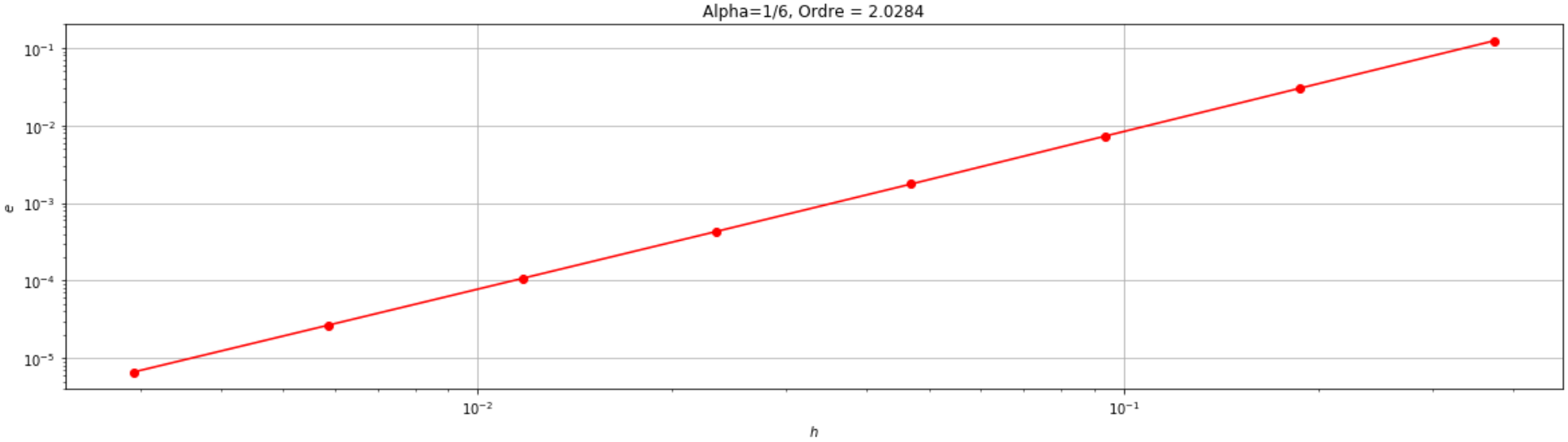
On calcule la solution approchée avec différentes valeurs de $h_k = 1/N_k$, à savoir $N_k = 2^3$, ... 2^{10}. On sauvegarde les valeurs de h_k dans le vecteur `H`.

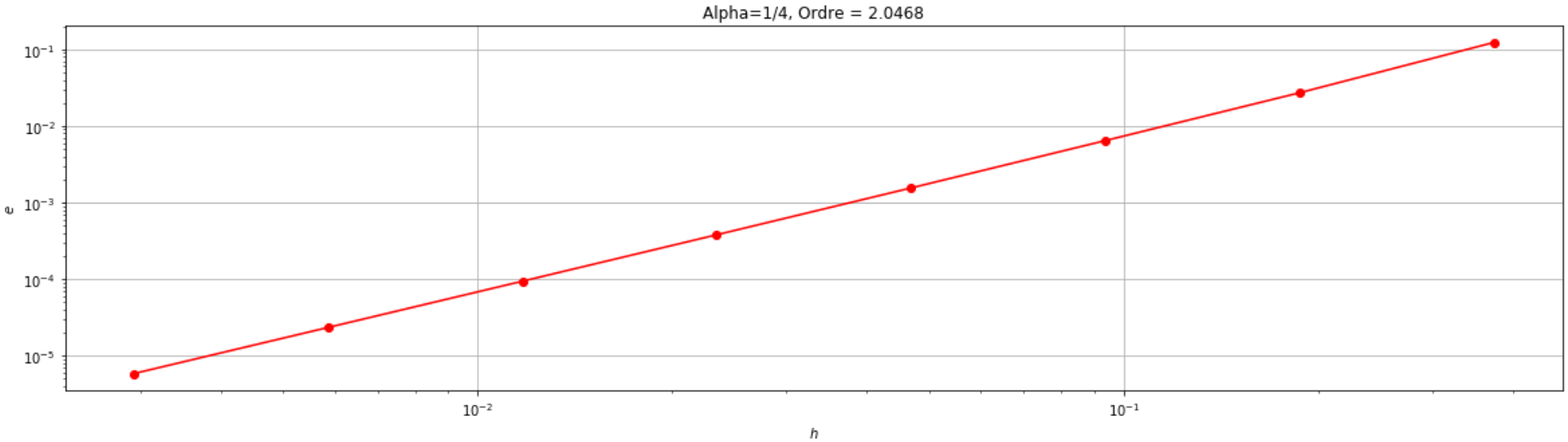
Pour chaque valeur de h_k, on calcule le maximum de la valeur absolue de l'erreur et on sauvegarde toutes ces erreurs dans le vecteur `err` de sort que `err[k]` contient $e_k = \max_{i=0,\dots,N_k} |y(t_i) - u_i|$ avec $N_k = 2^{k+1}$.

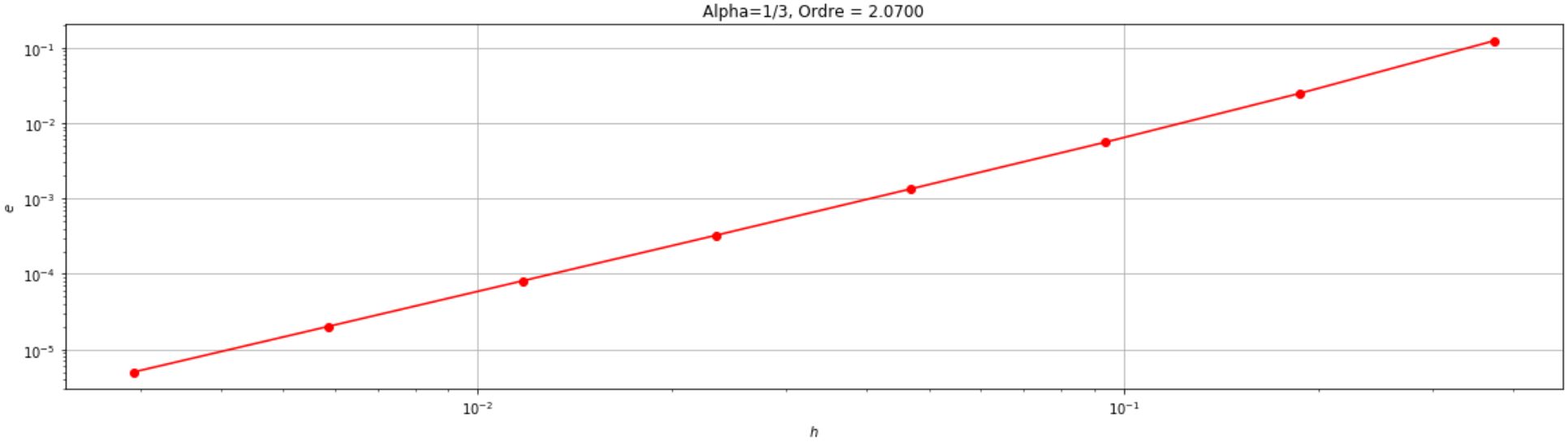
Pour afficher l'ordre de convergence on utilise une échelle logarithmique : on représente $\ln(h)$ sur l'axe des abscisses et $\ln(\mathrm{err})$ sur l'axe des ordonnées ainsi, si $\mathrm{err} = Ch^p$, alors $\ln(\mathrm{err}) = \ln(C) + p\ln(h)$. En échelle logarithmique, p représente donc la pente de la ligne droite $\ln(\mathrm{err})$.

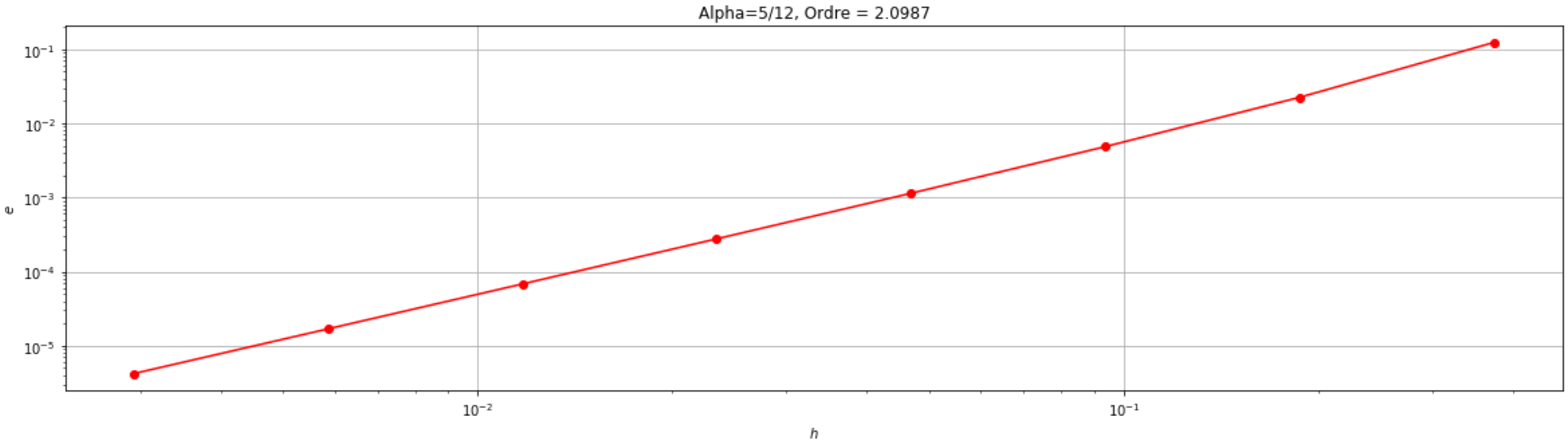
Pour estimer l'ordre de convergence on estime la pente de la droite qui relie l'erreur au pas k à l'erreur au pas $k+1$ en echelle logarithmique en utilisant la fonction `polyfit` basée sur la régression linéaire.

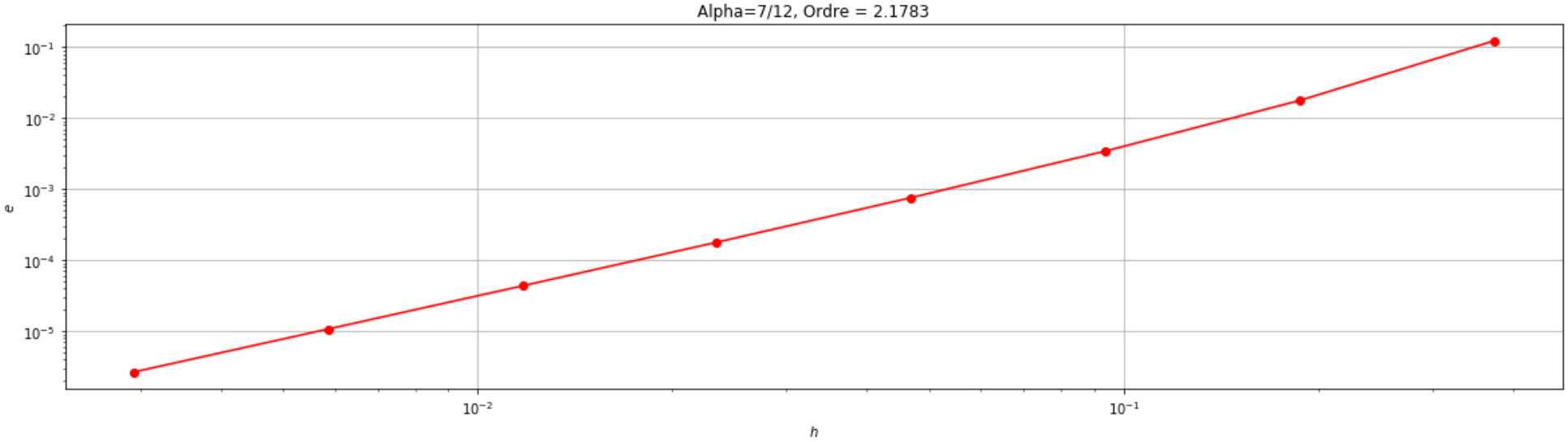
In [14]:
```
t0, y0, tfinal = 0, 2, 3
sol_exacte = lambda t : 1/(t**2+1/y0)
phi = lambda t,y : -2*t*y**2

i=0
for alpha in ALPHA:
    H = []
    err = []
    for k in range(8):
        N = 2**(k+3)
        tt = linspace(t0,tfinal,N+1)
        h = tt[1]-tt[0]
        H.append(h)
        yy = [sol_exacte(t) for t in tt]
        uu = RKalpha(phi,tt)
        err.append(max([abs(uu[i]-yy[i]) for i in range(len(uu))]))
    i+=1
    figure(i,figsize=(20,5))
    loglog(H,err, 'r-o')
    title('Alpha={}, Ordre = {:1.4f}'.format(alpha,polyfit(log(H),log(err), 1)[0]))
    xlabel('$h$')
    ylabel('$e$')
    legend()
    grid(True)
```

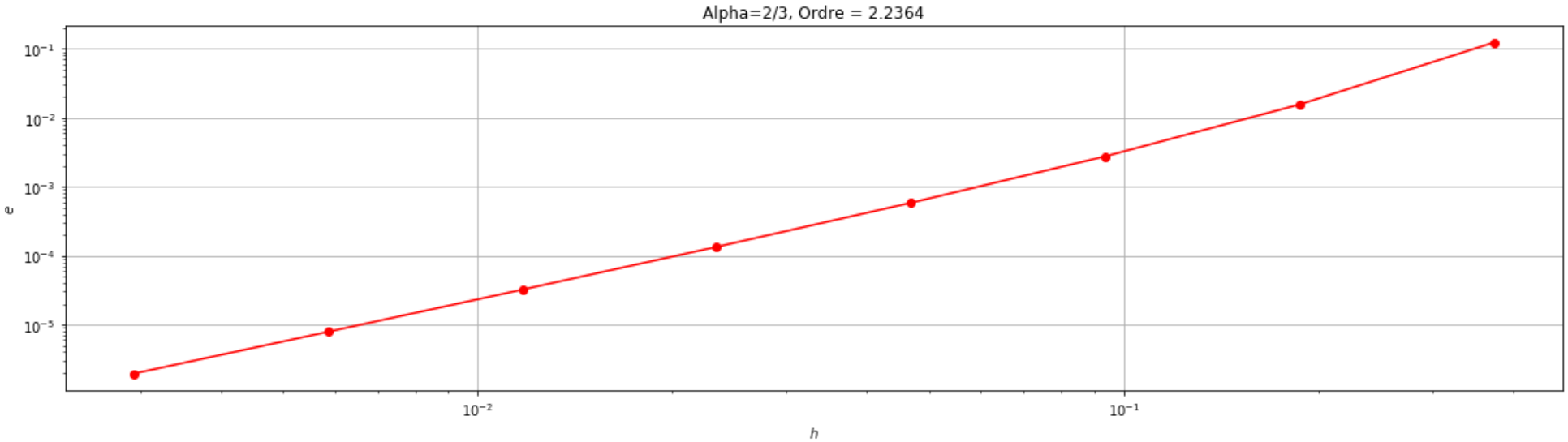

Alpha=1/6, Ordre = 2.0284
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
e
10^{-2}
10^{-1}
h


Alpha=1/4, Ordre = 2.0468
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
e
10^{-2}
10^{-1}
h

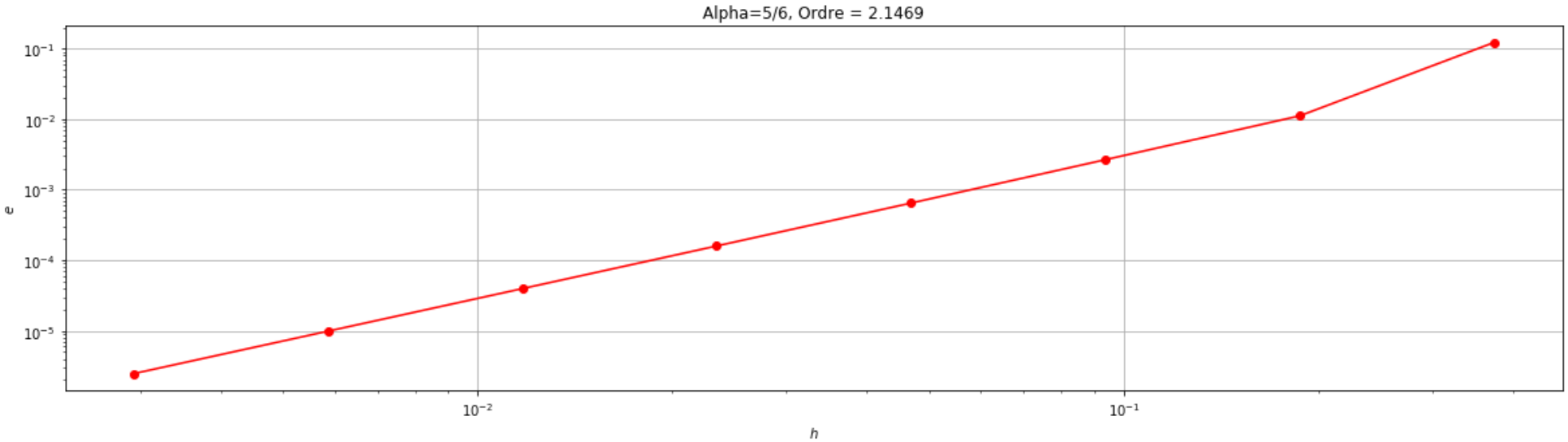

Alpha=1/3, Ordre = 2.0700
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
e
10^{-2}
10^{-1}
h

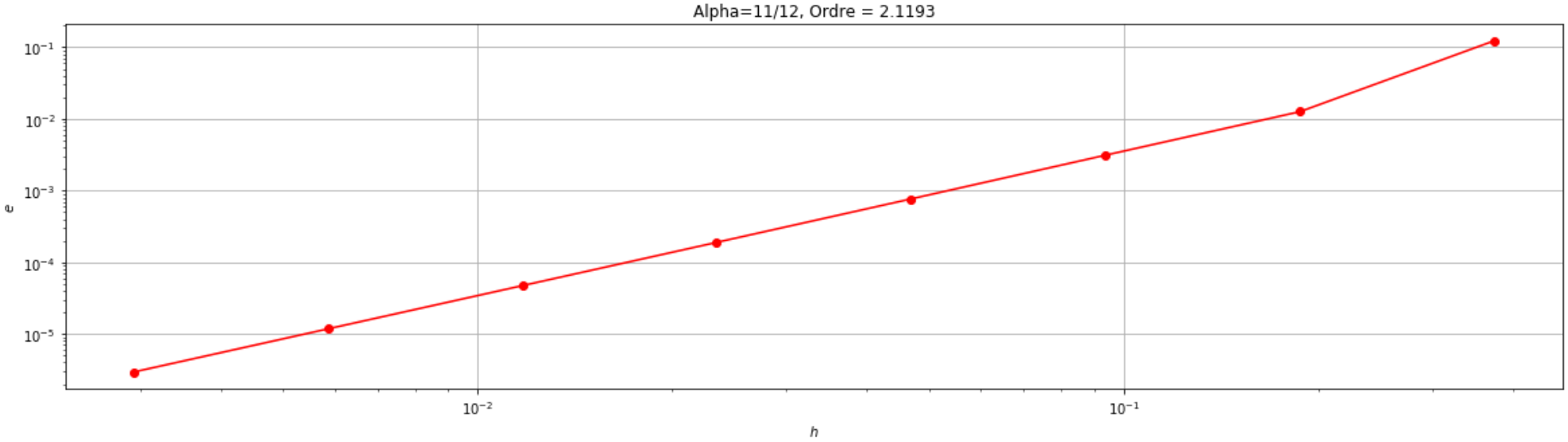

Alpha=5/12, Ordre = 2.0987
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
e
10^{-2}
10^{-1}
h


Alpha=7/12, Ordre = 2.1783
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
e
10^{-2}
10^{-1}
h


Alpha=2/3, Ordre = 2.2364
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
e
10^{-2}
10^{-1}
h


Alpha=3/4, Ordre = 2.2009
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
e
10^{-2}
10^{-1}
h


Alpha=5/6, Ordre = 2.1469
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
e
10^{-2}
10^{-1}
h


Alpha=11/12, Ordre = 2.1193
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
e
10^{-2}
10^{-1}
h