Université Toulouse 3 Paul Sabatier 2018-2019
L2 Parcours Spécial

TP 2 - Calcul Scientifique

In [1]:
```
%pylab inline

# %pylab inline est un équivalent de l'importation* de numpy et de matplotli
#%matplotlib inline
#from matplotlib.pyplot import *
#from numpy import *
#from scipy import *
```

```
Populating the interactive namespace from numpy and matplotlib
```

Exercice 1 - Résolution d'équations

On souhaite comparer, pour la résolution de l'équation non linéaire suivante, une méthode de point fixe et la méthode de Newton:

$$f(x) = 0, \qquad \text{avec} \qquad f(x) = x - \frac{1}{5}\sin x - \frac{1}{2}.$$

1) Vérifier que cette équation admet une unique solution notée $\bar{x}$ dans $[0, 2\pi]$.

2) Soit $(x_n)_{n\geq 0}$ la suite de réels définis par :

$$\begin{cases} x_{n+1} = \phi(x_n), \\ x_0 = \alpha \in \mathbb{R}, \end{cases} \qquad \text{avec} \qquad \phi(x) = \frac{1}{5}\sin x + \frac{1}{2}.$$

Implémenter une fonction `pointfixe` qui

- prend en argument la donnée initiale α et une tolérance tol
- renvoie la première valeur de x_k telle que $|f(x_k)| < tol$ ainsi que le premier entier k pour lequel cette valeur est atteinte.

Tester cette méthode avec $\alpha = -5$. Combien faut-il d'itérations de la méthode pour les tolérances 10^{-3}, 10^{-5} et 10^{-9} ?

3) Soit $(y_n)_{n\geq 0}$ la suite définie par:

$$\begin{cases} y_{n+1} = y_n - \frac{f(y_n)}{f'(y_n)}, \\ y_0 = \alpha \in \mathbb{R}, \end{cases}$$

Implémenter une fonction `Newton` qui

- prend en argument la valeur initiale α et une tolérance tol
- renvoie la première valeur de y_k telle que $|y_{k+1} - y_k| < tol$ ainsi que le premier entier k pour lequel cette valeur est atteinte.

Tester cette méthode avec $\alpha = -5$. Combien faut-il d'itérations pour les tolérances 10^{-3}, 10^{-5} et 10^{-9} ?

Correction question 1 : Etudions la fonction f sur $[0, 2\pi]$: f est dérivable et

$$\forall x \in [0, 2\pi],\ f'(x) = 1 - \frac{1}{5}\cos(x) \geq \frac{4}{5} > 0.$$

Donc la fonction f est croissante sur $[0, 2\pi]$. De plus, $f(0) = -\frac{1}{2} < 0$ et $f(2\pi) = 2\pi - \frac{1}{2} > 0$. Le théorème des valeurs intermédiaire et la monotonie de f assurent qu'il existe un unique $x^* \in [0, 2\pi]$ tel que $f(x^*) = 0$.

In [3]:
```
#Correction question 2
def f(x):
    return(x-0.2*sin(x)-0.5)

def phi(x):
    return(0.2*sin(x)+0.5)

# pour la fonction pointfixe, on va faire une boucle pour définir la suite p
# dès que |f(x_k)|< tol, donc c'est une boucle while avec le test |f(x_k)|<
# calculer le nombre d'itérations nécessaires. C'est à cela que sert k:
# la fonction ressort la valeur approchée et le nombre d'itéartions nécéssai
def pointfixe(a,tol):
        x=a
        k=1
        while abs(f(x))>tol:
            x=phi(x)
            k=k+1
        return(x,k)

print(pointfixe(-5,10**(-3)))
print(pointfixe(-5,10**(-5)))
print(pointfixe(-5,10**(-9)))
print(pointfixe(-5,10**(-12)))
```

```
(0.61578960906424507, 5)
(0.61547674021849985, 7)
(0.61546817048526936, 12)
(0.61546816949067318, 16)
```

In [4]:
```
#Correction question 3
#Pour implémenter la méthode de Newton on a besoin de f et de f', il faut do
# et on définit alors la fonction f et la fonction df qu'on appellera dans l
# On arrete la méthode de Newton quand |x_(k+1)-x_k| est plus petit que tol,
# (ici c'est ce que l'on a noté a, c'est pour cela que le test porte sur a)

def f(x):
    return(x-0.2*sin(x)-0.5)

def df(x):
    return(1-0.2*cos(x))

def newton(beta,tol,g,dg):
    x=beta
    k=1
    err=1
    while err>tol:
        a=g(x)/dg(x)
        x=x-a
        err=abs(a)
        k=k+1
    return(x,k)

print(newton(-5,10**(-3),f,df))
print(newton(-5,10**(-5),f,df))
print(newton(-5,10**(-9),f,df))
print(newton(-5,10**(-12),f,df))
```

```
(0.61546816950771188, 5)
(0.61546816948996541, 6)
(0.61546816948996541, 6)
(0.61546816948996541, 7)
```

Exercice 2 - Résolution approchée d'Equations différentielles linéaires

L'évolution au cours du temps de la concentration de certaines réactions chimiques peut être décrite par l'équation suivante:

$$\begin{cases} y'(t) = \frac{1}{1+t^2} y(t), \ t \geq 0 \\ y_0 = y_0 \in \mathbb{R}, \end{cases}$$

1) Calculer la solution exacte du problème

2) Ecrire les méthodes d'Euler explicite et implicite pour la résolution numérique de ce problème.

3) Implémenter deux fonctions `EulerE` et `EulerI` qui

- prennent en argument y0, N (le nombre de pas de temps) et T (le temps final de calcul)
- renvoient le vecteur des y_n^e et y_n^i pour $n = 0$ à N, obetnus par les méthodes d'Euler explicite et implicite respectivement. En prenant $y_0 = 5, T = 50$ et $N = 100$ calculer les vecteurs obtenus.

4) Tracer sur un même graphique des valeurs y_n^e, y_n^i et $y(t_n)$ calculés par les méthodes d'Euler explicite, implicite et la solution exacte. Recommencer pour $N = 1000$ puis $N = 10000$.

In [ ]:

Correction question 1 : On montre que la fonction $y(t) = y_0\, e^{\arctan(t)}$ est solution de l'équation différentielle (voir vidéo)

Correction question 2 : En utilisant les formules du cours sur les méthodes d'Euler explicite et implicte on obtient, pour le problème considéré :

euler explicite :

$$\begin{cases} y_0\, donné \\ \forall n = 0..N-1,\ y_{n+1} = y_n + h * \frac{y_n}{1+t_n^2}, \end{cases}$$

où $t_n = n * h,\ h = T/N.$

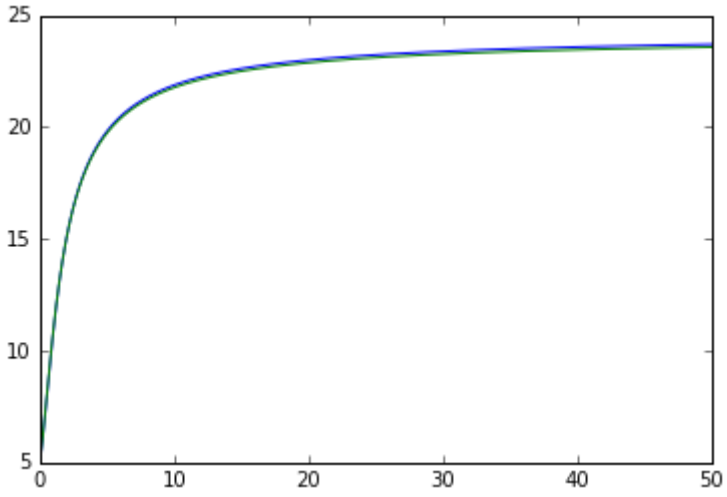
euler implicite :

$$\begin{cases} y_0\, donné \\ \forall n = 0..N-1,\ y_{n+1} = \frac{y_n}{1-\frac{h}{1+t_{n+1}^2}}, \end{cases}$$

où $t_n = n * h,\ h = T/N.$

In [5]:
```
# La fonction EulerE va renvoyer un vecteur qui contient y0 y1 ... yN. Pour
# N+1 qui ne contient que des zéros et on va ranger les yi dedans avec une l
# range dans le vecteur de sortie (il s'appelle res dans le code)

def EulerE(y0,N,T):
    h=T/N # le pas de la méthode : h=T/N
    res=zeros(N+1)  # ca va etre le vecteur de sortie dans lequel on range y
    res[0]=y0  # on range y0 à la place 0
    for n in range(N):  # on fait une boucle pour calculer chaque y_(n+1) et
        res[n+1]=res[n]+h*res[n]/(1+n**2*h**2)
    return(res)
y0=5.0
T=50.0
N=1000
Ye=EulerE(y0,N,T)

t=linspace(0,T,N+1)
Yexact=y0*exp(arctan(t))
plot(t,Ye,t,Yexact)
#On trace les solutions approchées données par la méthode d'Euler explicite
# et la solution exacte sur le même garphe.
#On remarque que plus N est grand (et donc h petit), plus les courbes sont p
# c'est la convergence de la méthode qui l'explique (voir cours).
```

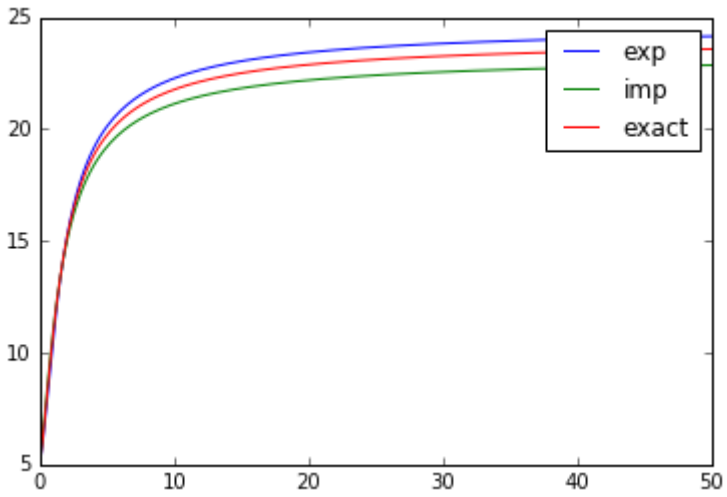
Out[5]:
```
[<matplotlib.lines.Line2D at 0x10dabb5d0>,
 <matplotlib.lines.Line2D at 0x10dabb7d0>]
```



In [7]:

```
#On fait exactement pareil mais en changeant la formule pour la méthode d'Eu

def EulerI(y0,N,T):
    h=T/N
    res=zeros(N+1)
    res[0]=y0
    for n in range(N):
        res[n+1]=res[n]/(1-h/(1+(n+1)**2*h**2))
    return(res)
y0=5.0
T=50.0
N=200
Ye=EulerE(y0,N,T)
Yi=EulerI(y0,N,T)

# on trace les 3 sur un même graphique pour voir les différents résultats...
t=linspace(0,T,N+1)
Yexact=y0*exp(arctan(t))
plot(t,Ye,t,Yi,t,Yexact)
legend(['exp','imp','exact'])
```

Out[7]: <matplotlib.legend.Legend at 0x10bf7a110>



Exercice 3 - Equation logistique

On considère l'équation logistique utilisée pour modéliser la croissance de populations:

$$\begin{cases} y'(t) = \alpha(1 - \frac{y(t)}{K})y(t),\ t \geq 0 \\ y_0 = y_0 \in \mathbb{R}, \end{cases}$$

où α et K sont des paramètres strictement positifs décrivant respectivement la vitesse de croissance de la population pour des petites populations et la population maximale que le milieu peut accueillir.

1) Vérifier que $y(t) = \frac{Ky_0}{y_0+(K-y_0)e^{-\alpha t}}$ est solution de ce problème.

2) En déduire que, dès que $y_0 \in]0, K[$, alors, pour tout $t > 0$, on a $y(t) \in]0; K[$ et $\lim_{t\to\infty} y(t) = K$.

3) On considère un intervalle de temps $[0, T]$ sur lequel on cherche à approcher la solution. On définit une discrétisation à $N + 1$ points équidistants $(t_n)_{0\leq n\leq N}$ de pas h.

On définit enfin la méthode de Heun pour la résolution approchée de $y'(t) = f(y(t))$ comme suit :

$$\begin{cases} y_0 \\ y_{n+1} = y_n + \frac{h}{2}[f(y_n) + f(y_n + hf(y_n))]\ ,\ \forall n = 0..N-1 \end{cases}$$

Ecrire les itérations des méthodes d'Euler explicite et de Heun pour l'équation logistique et implémenter une fonction `EulerE` et une fonction `Heun` qui

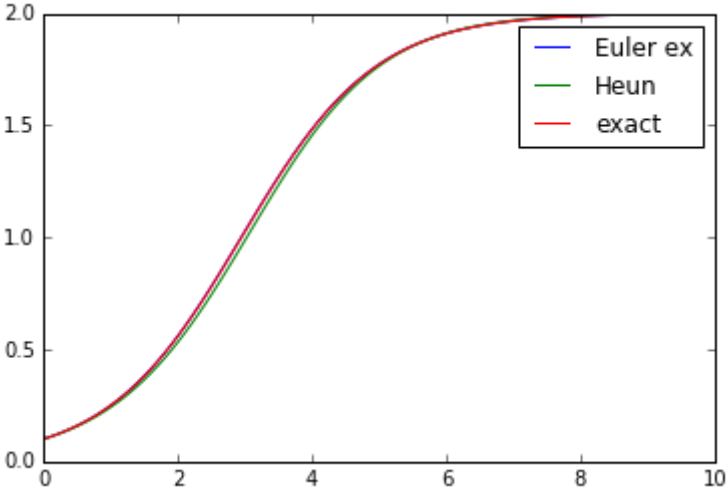
- prennent en argument $y_0,\ N,\ T$
- renvoient les vecteurs $(y_n^e)_{n=0..N}$ et $(y_n^H)_{n=0..N}$ obtenus par les méthodes d'Euler explicite et Heun respectivement.

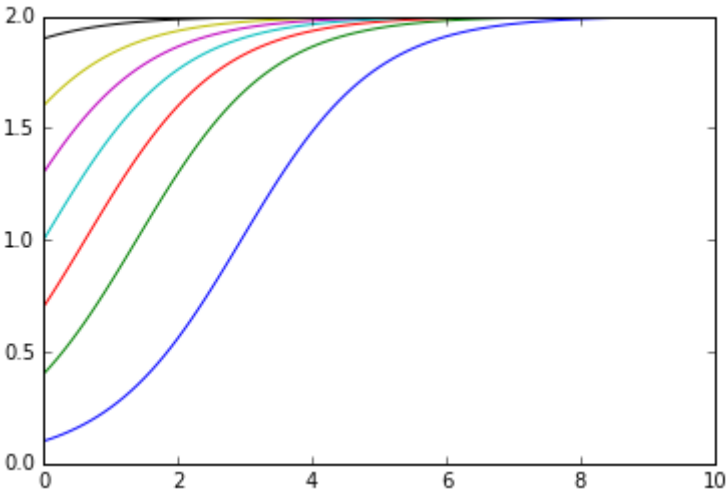
4) Tracer sur un même graphe les solutions approchées par ces deux méthodes et la solution exacte pour $T = 5$ et $N = 500$. Tester le comportement obtenu en fonction du choix de y_0.

Correction question 1 : il suffit de le vérifer en calculant $y'(t)$ et $\alpha(1 - \frac{y(t)}{K})y(t)$ (voir vidéo)

Correction question 2 : on le voit grâce à la formule.

In [8]:
```
#Correction question 3

alpha=1
K=2

def exact(y0,t):
    return(K*y0/(y0+(K-y0)*exp(-alpha*t)))

def F(x):
    return(alpha*x*(1-x/K))

def EulerE(y0,N,T):
    h=T/N
    res=zeros(N+1)
    res[0]=y0
    for n in range(N):
        res[n+1]=res[n]+h*F(res[n])
    return(res)

def Heun(y0,N,T):
    h=T/N
    res=zeros(N+1)
    res[0]=y0
    for n in range(N):
        res[n+1]=res[n]+h/2*(F(res[n])+F(res[n]+h*F(res[n])))
    return(res)

N=100
T=10.0
y0=0.1
t=linspace(0,T,N+1)
Ye=EulerE(y0,N,T)
Yh=Heun(y0,N,T)
Yex=exact(y0,t)
plot(t,Yex,t,Ye,t,Yh)
legend(['Euler ex','Heun','exact'])

#Comportement pour plusieurs données initiales
figure()
for y0 in arange(0.1,2,0.3):
    Yex=exact(y0,t)
    plot(t,Yex)
```


Euler ex
Heun
exact
2.0
1.5
1.0
0.5
0.0
0
2
4
6
8
10


2.0
1.5
1.0
0.5
0.0
0
2
4
6
8
10

In [ ]: