In [1]:

```
from IPython.display import display, Latex
from IPython.core.display import HTML
css_file = './custom.css'
HTML(open(css_file, "r").read())
```

Out[1]:

M62 TP 4 - Implémentation de schémas

Vous pouvez commencer à exécuter un serveur de notebook en écrivant dans un terminal :

`jupyter notebook &`

Cela imprimera des informations dans votre console et ouvrira un navigateur Web.
La page d'arrivée est le tableau de bord qui affiche les notebook actuellement disponibles (par défaut, le répertoire à partir duquel le serveur du bloc-notes a été démarré).

Vous pouvez

- soit créer un nouveau notebook à partir du tableau de bord avec le bouton (en haut à droite) `Nouveau`
- soit ouvrir un notebook existant en cliquant sur leur nom (par exemple, vous pouvez télécharger le notebook de présentation ici (http://nbviewer.jupyter.org/url/faccanoni.univ-tln.fr/user/enseignements/20182019/M62-CM1.ipynb).)

Pour ce TP, on importera tous les modules necessaires comme suit:

In [2]:

```
%reset -f
%matplotlib inline
%autosave 300

from math import *
from matplotlib.pylab import *
from scipy.optimize import fsolve
```

```
Autosaving every 300 seconds
```

Table of Contents

1 Implémentation des schémas

On écrit les schémas numériques :

- les nœuds d'intégration $[t_0, t_1, \dots, t_N]$ sont contenus dans le vecteur `tt`
- les valeurs $[u_0, u_1, \dots, u_N]$ pour chaque méthode sont contenues dans le vecteur `uu` .

1.1 Schémas explicites

Schéma d'Euler progressif = de Adam-Bashforth à 1 pas:

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

In [3]:

```
def euler_progressif(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    for i in range(len(tt)-1):
        uu.append(uu[i]+h*phi(tt[i],uu[i]))
    return uu
```

Schéma de Adam-Bashforth à 2 pas:

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0), \\ u_{n+1} = u_n + \frac{h}{2}\left(3\varphi(t_n, u_n) - \varphi(t_{n-1}, u_{n-1})\right) \quad n = 1, 2, 3, 4, 5, \dots N \end{cases}$$

In [4]:

```
def AB2(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    uu.append(uu[0]+h*phi(tt[0],uu[0]))
    for i in range(1,len(tt)-1):
        k1 = h * phi( tt[i], uu[i] )
        k2 = h * phi( tt[i-1], uu[i-1] )
        uu.append( uu[i] + (3.*k1-k2) /2.0 )
    return uu
```

Schéma de Adam-Bashforth à 3 pas:

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0), \\ u_2 = u_1 + \frac{h}{2}\Big(3\varphi(t_1, u_1) - \varphi(t_0, u_0)\Big), \\ u_{n+1} = u_n + \frac{h}{12}\Big(23\varphi(t_n, u_n) - 16\varphi(t_{n-1}, u_{n-1}) + 5\varphi(t_{n-2}, u_{n-2})\Big) \end{cases}$$

In [5]:

```
def AB3(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    uu.append(uu[0]+h*phi(tt[0],uu[0]))
    uu.append(uu[1]+h*(3.*phi(tt[1],uu[1])-phi(tt[0],uu[0]))/2
    for i in range(2,len(tt)-1):
        k1 = h * phi( tt[i], uu[i] )
        k2 = h * phi( tt[i-1], uu[i-1] )
        k3 = h * phi( tt[i-2], uu[i-2] )
        uu.append( uu[i] + (23.*k1-16.*k2+5.*k3) /12.0 )
    return uu
```

Schéma de Adam-Bashforth à 4 pas:

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0), \\ u_2 = u_1 + \frac{h}{2}\Big(3\varphi(t_1, u_1) - \varphi(t_0, u_0)\Big), \\ u_3 = u_2 + \frac{h}{12}\Big(23\varphi(t_2, u_2) - 16\varphi(t_1, u_1) + 5\varphi(t_0, u_0)\Big), \\ u_{n+1} = u_n + \frac{h}{24}\Big(55\varphi(t_n, u_n) - 59\varphi(t_{n-1}, u_{n-1}) + 37\varphi(t_{n-2}, u_{n-2}) - \end{cases}$$

In [6]:

```
def AB4(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    uu.append(uu[0]+h*phi(tt[0],uu[0]))
    uu.append(uu[1]+h*(3.*phi(tt[1],uu[1])-phi(tt[0],uu[0]))/2
    uu.append(uu[2]+h*(23*phi(tt[2],uu[2])-16*phi(tt[1],uu[1])
    for i in range(3,len(tt)-1):
        k1 = h * phi( tt[i], uu[i] )
        k2 = h * phi( tt[i-1], uu[i-1] )
        k3 = h * phi( tt[i-2], uu[i-2] )
        k4 = h * phi( tt[i-3], uu[i-3] )
        uu.append( uu[i] + (55.*k1-59.*k2+37.*k3-9.*k4) /24.0
    return uu
```

Schéma de Adam-Bashforth à 5 pas:

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0), \\ u_2 = u_1 + \frac{h}{2}\Big(3\varphi(t_1, u_1) - \varphi(t_0, u_0)\Big), \\ u_3 = u_2 + \frac{h}{12}\Big(23\varphi(t_2, u_2) - 16\varphi(t_1, u_1) + 5\varphi(t_0, u_0)\Big), \\ u_4 = u_3 + \frac{h}{24}\Big(55\varphi(t_3, u_3) - 59\varphi(t_2, u_2) + 37\varphi(t_1, u_1) - 9\varphi(t_0, u_0)\Big) \\ u_{n+1} = u_n + \frac{h}{720}\Big(1901\varphi(t_n, u_n) - 2774\varphi(t_{n-1}, u_{n-1}) + 2616\varphi(t_{n-2}, \end{cases}$$

In [7]:

```
def AB5(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    uu.append(uu[0]+h*phi(tt[0],uu[0]))
    uu.append(uu[1]+h*(3.*phi(tt[1],uu[1])-phi(tt[0],uu[0]))/2
    uu.append(uu[2]+h*(23*phi(tt[2],uu[2])-16*phi(tt[1],uu[1])
    uu.append(uu[3]+h*(55*phi(tt[3],uu[3])-59*phi(tt[2],uu[2])
    for i in range(4,len(tt)-1):
        k1 = h * phi( tt[i], uu[i] )
        k2 = h * phi( tt[i-1], uu[i-1] )
        k3 = h * phi( tt[i-2], uu[i-2] )
        k4 = h * phi( tt[i-3], uu[i-3] )
        k5 = h * phi( tt[i-4], uu[i-4] )
        uu.append( uu[i] + (1901.*k1-2774.*k2+2616.*k3-1274.*k
    return uu
```

Schéma de Nylström à 2 pas:

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0), \\ u_{n+1} = u_{n-1} + 2h\varphi(t_n, u_n) \quad n = 1, 2, 3, 4, 5, \dots N-1 \end{cases}$$

In [8]:

```
def N2(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    uu.append(uu[0]+h*phi(tt[0],uu[0]))
    for i in range(1,len(tt)-1):
        k1 = h * phi( tt[i], uu[i] )
        uu.append( uu[i-1] + 2.0*k1 )
    return uu
```

Schéma de Nylström à 3 pas:

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0), \\ u_2 = u_0 + 2h\varphi(t_1, u_1), \\ u_{n+1} = u_{n-1} + \frac{h}{3}\Big(7\varphi(t_n, u_n) - 2\varphi(t_{n-1}, u_{n-1}) + \varphi(t_{n-2}, u_{n-2})\Big) \quad n \end{cases}$$

In [9]:

```
def N3(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    uu.append(uu[0]+h*phi(tt[0],uu[0]))
    uu.append(uu[0]+2*h*phi(tt[1],uu[1]))
    for i in range(2,len(tt)-1):
        k1 = h * phi( tt[i], uu[i] )
        k2 = h * phi( tt[i-1], uu[i-1] )
        k3 = h * phi( tt[i-2], uu[i-2] )
        uu.append( uu[i-1] + (7.*k1-2.*k2+k3)/3. )
    return uu
```

Schéma de Nylström à 4 pas:

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0), \\ u_2 = u_0 + 2h\varphi(t_1, u_1), \\ u_3 = u_1 + \frac{h}{3}\Big(7\varphi(t_2, u_2) - 2\varphi(t_1, u_1) + \varphi(t_0, u_0)\Big), \\ u_{n+1} = u_{n-1} + \frac{h}{3}\Big(8\varphi(t_n, u_n) - 5\varphi(t_{n-1}, u_{n-1}) + 4\varphi(t_{n-2}, u_{n-2}) - \varphi($$

In [10]:

```
def N4(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    uu.append(uu[0]+h*phi(tt[0],uu[0]))
    uu.append(uu[0]+2*h*phi(tt[1],uu[1]))
    uu.append(uu[1]+(7*h*phi(tt[2],uu[2])-2.*h*phi(tt[1],uu[1]
    for i in range(3,len(tt)-1):
        k1 = phi( tt[i], uu[i] )
        k2 = phi( tt[i-1], uu[i-1] )
        k3 = phi( tt[i-2], uu[i-2] )
        k4 = phi( tt[i-3], uu[i-3] )
        uu.append( uu[i-1] + (8.*k1-5.*k2+4.*k3-k4)*h/3. )
    return uu
```

Schéma de Runke-Kutta RK4:

$$\begin{cases} u_0 = y(t_0) = y_0, \\ \tilde{u}_{n+1/2} = u_n + \frac{h}{2}\varphi(t_n, u_n), \\ \check{u}_{n+1/2} = u_n + \frac{h}{2}\varphi(t_n + \frac{h}{2}, \tilde{u}_{n+1/2}), \\ \hat{u}_{n+1} = u_n + h\varphi(t_{n+1}, \check{u}_{n+1/2}), \\ u_{n+1} = u_n + \frac{h}{6}\left(\varphi(t_n, u_n) + 2\varphi(t_n + \frac{h}{2}, \tilde{u}_{n+1/2}) + 2\varphi(t_n + \frac{h}{2}, \check{u}_{n+1/2})\right. \end{cases}$$

In [11]:

```
def RK4(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    for i in range(len(tt)-1):
        k1 = phi( tt[i], uu[i] )
        k2 = phi( tt[i]+h/2. , uu[i]+k1/2. )
        k3 = phi( tt[i]+h/2. , uu[i]+k2/2. )
        k4 = phi( tt[i+1]    , uu[i]+k3 )
        uu.append( uu[i] + (k1+2.0*k2+2.0*k3+k4)*h/6 )
    return uu
```

1.2 Schémas implicites

Schéma d'Euler régressif :

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

Attention : u_{n+1} est solution de l'équation $x = u_n + h\varphi(t_{n+1}, x)$, c'est-à-dire un zéro de la fonction (en générale non linéaire)
$\lambda : x \mapsto -x + u_n + h\varphi(t_{n+1}, x)$.

In [12]:

```
def euler_regressif(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    for i in range(len(tt)-1):
        temp = fsolve(lambda x: -x+uu[i]+h*phi(tt[i+1],x), uu[
        uu.append(temp)
    return uu
```

Schéma de Crank-Nicolson :

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

Attention : u_{n+1} est solution de l'équation $x = u_n + \frac{h}{2}(\varphi(t_n, u_n) + \varphi(t_{n+1}, x))$, c'est-à-dire un zéro de la fonction (en générale non linéaire) $\lambda : x \mapsto -x + u_n + \frac{h}{2}(\varphi(t_n, u_n) + \varphi(t_{n+1}, x))$

In [13]:

```
def CN(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    for i in range(len(tt)-1):
        temp = fsolve(lambda x: -x+uu[i]+0.5*h*( phi(tt[i+1],x
        uu.append(temp)
    return uu
```

Schéma de AM-2 :

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + \frac{h}{2}\Big(\varphi(t_1, u_1) + \varphi(t_0, u_0)\Big), \\ u_{n+1} = u_n + h\Big(5\varphi(t_{n+1}, u_{n+1}) + 8\varphi(t_n, u_n) - \varphi(t_{n-1}, u_{n-1})\Big) \end{cases} \quad n =$$

In [14]:

```
def AM2(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    temp = fsolve(lambda x: -x+uu[0]+0.5*h*( phi(tt[1],x)+phi(
    uu.append(temp)
    for i in range(1,len(tt)-1):
        temp = fsolve(lambda x: -x+uu[i]+h*( 5.*phi(tt[i+1],x)
        uu.append(temp)
    return uu
```

Schéma de AM-3 :

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + \frac{h}{2}\Big(\varphi(t_1,u_1) + \varphi(t_0,u_0)\Big), \\ u_2 = u_n + h\Big(5\varphi(t_2,u_2) + 8\varphi(t_1,u_1) - \varphi(t_0,u_0)\Big), \\ u_{n+1} = u_n + h\Big(9\varphi(t_{n+1},u_{n+1}) + 19\varphi(t_n,u_n) - 5\varphi(t_{n-1},u_{n-1}) + \varphi(t_{} \end{cases}$$

In [15]:

```
def AM3(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    temp = fsolve(lambda x: -x+uu[0]+0.5*h*( phi(tt[1],x)+phi(
    uu.append(temp)
    temp = fsolve(lambda x: -x+uu[1]+h*( 5.*phi(tt[2],x)+8.*ph
    uu.append(temp)
    for i in range(2,len(tt)-1):
        temp = fsolve(lambda x: -x+uu[i]+h*( 9.*phi(tt[i+1],x)
        uu.append(temp)
    return uu
```

Schéma de AM-4 :

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + \frac{h}{2}\Big(\varphi(t_1,u_1) + \varphi(t_0,u_0)\Big), \\ u_2 = u_1 + h\Big(5\varphi(t_2,u_2) + 8\varphi(t_1,u_1) - \varphi(t_0,u_0)\Big), \\ u_3 = u_2 + h\Big(9\varphi(t_3,u_3) + 19\varphi(t_2,u_2) - 5\varphi(t_1,u_1) + \varphi(t_0,u_0)\Big), \\ u_{n+1} = u_n + h\Big(251\varphi(t_{n+1},u_{n+1}) + 646\varphi(t_n,u_n) - 264\varphi(t_{n-1},u_{n-1}) \end{cases}$$

In [16]:

```
def AM4(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    temp = fsolve(lambda x: -x+uu[0]+0.5*h*( phi(tt[1],x)+phi(
    uu.append(temp)
    temp = fsolve(lambda x: -x+uu[1]+h*( 5.*phi(tt[2],x)+8.*ph
    uu.append(temp)
    temp = fsolve(lambda x: -x+uu[2]+h*( 9.*phi(tt[3],x)+19.*pl
    uu.append(temp)
    for i in range(3,len(tt)-1):
        temp = fsolve(lambda x: -x+uu[i]+h*( 251.*phi(tt[i+1],
        uu.append(temp)
    return uu
```

Schéma BDF2 :

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + h\varphi(t_1, u_1), \\ u_{n+1} = \frac{4}{3}u_n - \frac{1}{3}u_{n-1} + \frac{2}{3}\varphi(t_{n+1}, u_{n+1}) \quad n = 1, 2, \dots N-1 \end{cases}$$

In [17]:

```
def BDF2(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    temp = fsolve(lambda x: -x+uu[0]+h*phi(tt[1],x), uu[0])
    uu.append(temp)
    for i in range(1,len(tt)-1):
        temp = fsolve(lambda x: -x+4./3.*uu[i]-1./3.*uu[i-1] +
        uu.append(temp)
    return uu
```

Schéma BDF3 :

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + h\varphi(t_1, u_1), \\ u_2 = \frac{4}{3}u_1 - \frac{1}{3}u_0 + \frac{2}{3}\varphi(t_2, u_2), \\ u_{n+1} = \frac{4}{3}u_n - \frac{1}{3}u_{n-1} + \frac{2}{3}\varphi(t_{n+1}, u_{n+1}) \quad n = 2, 3, \dots N-1 \end{cases}$$

In [18]:

```
def BDF3(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    temp = fsolve(lambda x: -x+uu[0]+h*phi(tt[1],x), uu[0])
    uu.append(temp)
    temp = fsolve(lambda x: -x+4./3.*uu[1]-1./3.*uu[0] + 2./3.
    uu.append(temp)
    for i in range(2,len(tt)-1):
        temp = fsolve(lambda x: -x+18./11.*uu[i]-9./11.*uu[i-1
        uu.append(temp)
    return uu
```

1.3 Schémas predicteur-correcteur

Schéma d'Euler modifié:

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

In [19]:

```
def euler_modifie(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    for i in range(len(tt)-1):
        k1 = h * phi( tt[i], uu[i] )
        uu.append( uu[i]+h*phi(tt[i]+h/2.,uu[i]+k1/2.) )
    return uu
```

Schéma de Heun:

$$\begin{cases} u_0 = y_0, \\ \tilde{u} = u_n + h\varphi(t_n, u_n) \\ u_{n+1} = u_n + \frac{h}{2}\Big(\varphi(t_n, u_n) + \varphi(t_{n+1}, \tilde{u})\Big) \quad n = 0, 1, 2, \dots N-1 \end{cases}$$

In [20]:

```
def heun(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    for i in range(len(tt)-1):
        k1 = h * phi( tt[i], uu[i] )
        k2 = h * phi( tt[i+1], uu[i] + k1  )
        uu.append( uu[i] + (k1+k2) /2.0 )
    return uu
```

Schéma AM-2 AB-1 :

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0), \\ \tilde{u} = u_n + h\varphi(t_n, u_n), \\ u_{n+1} = u_n + \frac{h}{12}\left(5\varphi(t_{n+1}, \tilde{u}) + 8\varphi(t_n, u_n) - \varphi(t_{n-1}, u_{n-1})\right) \quad n = 1, \end{cases}$$

In [21]:

```
def AM2AB1(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    uu.append(uu[0]+h*phi(tt[0],uu[0]))
    for i in range(1,len(tt)-1):
        pred = uu[i] + h*phi(tt[i],uu[i])
        uu.append(uu[i]+h*(5.*phi(tt[i+1],pred)+8.*phi(tt[i],u
    return uu
```

Schéma AM-3 AB-2 :

$$\begin{cases} u_0 = y_0, \\ u_1 = u_0 + h\varphi(t_0, u_0), \\ u_2 = u_0 + \frac{h}{2}(3\varphi(t_1, u_1) - \varphi(t_0, u_0)), \\ \tilde{u} = u_n + \frac{h}{2}(3\varphi(t_n, u_n) - \varphi(t_{n-1}, u_{n-1})), \\ u_{n+1} = u_n + \frac{h}{24}\left(9\varphi(t_{n+1}, \tilde{u}) + 19\varphi(t_n, u_n) - 5\varphi(t_{n-1}, u_{n-1}) + \varphi(t_{n-

In [22]:

```
def AM3AB2(phi,tt):
    h = tt[1]-tt[0]
    uu = [y0]
    uu.append(uu[0]+h*phi(tt[0],uu[0]))
    uu.append(uu[0]+0.5*h*(3.*phi(tt[1],uu[1])-phi(tt[0],uu[0]
    for i in range(2,len(tt)-1):
        k1 = h * phi( tt[i], uu[i] )
        k2 = h * phi( tt[i-1], uu[i-1] )
        pred = uu[i] + (3.*k1-k2) /2.0
        uu.append(uu[i]+h*(5.*phi(tt[i+1],pred)+8.*phi(tt[i],u
    return uu
```

2 Comparaison sur un exemple

Considérons le problème de Cauchy

trouver la fonction $y\colon I \subset \mathbb{R} \to \mathbb{R}$ définie sur l'intervalle $I = [0, 1]$ telle que

$$\begin{cases} y'(t) = 2ty(t), & \forall t \in I = [0, 1], \\ y(0) = 1 \end{cases}$$

dont la solution est $y(t) = e^{t^2}$.

On se propose de

1. calculer la solution approchée obtenue avec la méthode d'Euler explicite avec $h = 1/N$ et $N = 8$ (pour bien visualiser les erreurs);
2. même exercice pour les méthodes AB_2, AB_3, AB_4, AB_5, N_2, N_3, N_4, RK_4;
3. même exercice pour les méthodes d'Euler implicite, de Crank-Nicolson, AM_1, AM_2, AM_3, AM_4, BDF_2, BDF_3
4. même exercice pour les méthodes predictor-corrector Euler modifié, Heun, AM_2-AB_1, AM_3-AB_2.

On définit la solution exacte:

In [23]:

```
sol_exacte = lambda t,y0 : y0*exp(t**2)
```

On définit l'équation différentielle : `phi` est une fonction python qui contient la fonction mathématique $\varphi(t, y) = 2ty$ dépendant des variables t et y.

In [24]:

```
phi = lambda t,y : 2.*y*t
```

On initialise le problème de Cauchy

In [25]:

```
t0 = 0.
tfinal = 1.
y0 = 1.
```

On introduit la discrétisation: les nœuds d'intégration $[t_0, t_1, \ldots, t_N]$ sont contenus dans le vecteur `tt`

In [26]:

```
N = 8
h = (tfinal-t0)/N
tt = linspace(t0,tfinal,N+1)
```

On calcule les solutions exacte et approchées:

In [27]:

```
yy = [sol_exacte(t,y0) for t in tt]

uu_ep   = euler_progressif(phi,tt)
uu_AB2  = AB2(phi,tt)
uu_AB3  = AB3(phi,tt)
uu_AB4  = AB4(phi,tt)
uu_AB5  = AB5(phi,tt)
uu_N2   = N2(phi,tt)
uu_N3   = N3(phi,tt)
uu_N4   = N4(phi,tt)
uu_RK4  = RK4(phi,tt)

uu_er   = euler_regressif(phi,tt)
uu_CN   = CN(phi,tt)
uu_AM2  = AM2(phi,tt)
uu_AM3  = AM3(phi,tt)
uu_AM4  = AM4(phi,tt)
uu_BDF2 = BDF2(phi,tt)
uu_BDF3 = BDF3(phi,tt)

uu_em   = euler_modifie(phi,tt)
uu_heun = heun(phi,tt)
uu_AM2AB1 = AM2AB1(phi,tt)
uu_AM3AB2 = AM3AB2(phi,tt)
```

On compare les graphes des solutions exacte et approchées et on affiche le maximum de l'erreur:

In [29]:

```
figure(1,figsize=(20,16))

subplot(4,5,1)
plot(tt,yy,'b-',tt,uu_ep,'r-D')
title('AB1=EE - max(|err|)=%1.3f'%(max([abs(uu_ep[i]-yy[i]) fo

subplot(4,5,2)
plot(tt,yy,'b-',tt,uu_AB2,'r-D')
title('AB2 - max(|err|)=%1.3f'%(max([abs(uu_AB2[i]-yy[i]) for

subplot(4,5,3)
plot(tt,yy,'b-',tt,uu_AB3,'r-D')
title('AB3 - max(|err|)=%1.3f'%(max([abs(uu_AB3[i]-yy[i]) for

subplot(4,5,4)
plot(tt,yy,'b-',tt,uu_AB4,'r-D')
title('AB4 - max(|err|)=%1.3f'%(max([abs(uu_AB4[i]-yy[i]) for

subplot(4,5,5)
plot(tt,yy,'b-',tt,uu_AB5,'r-D')
title('AB5 - max(|err|)=%1.3f'%(max([abs(uu_AB5[i]-yy[i]) for

subplot(4,5,6)
plot(tt,yy,'b-',tt,uu_N2,'r-D')
title('N2 - max(|err|)=%1.3f'%(max([abs(uu_N2[i]-yy[i]) for i

subplot(4,5,7)
plot(tt,yy,'b-',tt,uu_N3,'r-D')
title('N3 - max(|err|)=%1.3f'%(max([abs(uu_N3[i]-yy[i]) for i

subplot(4,5,8)
plot(tt,yy,'b-',tt,uu_N4,'r-D')
title('N4 - max(|err|)=%1.3f'%(max([abs(uu_N4[i]-yy[i]) for i

subplot(4,5,9)
plot(tt,yy,'b-',tt,uu_RK4,'r-D')
title('RK4 - max(|err|)=%1.3f'%(max([abs(uu_RK4[i]-yy[i]) for

subplot(4,5,10)
plot(tt,yy,'b-',tt,uu_er,'r-D')
title('AM0=EI - max(|err|)=%1.3f'%(max([abs(uu_er[i]-yy[i]) fo

subplot(4,5,11)
plot(tt,yy,'b-',tt,uu_CN,'r-D')
title('AM1=CN - max(|err|)=%1.3f'%(max([abs(uu_CN[i]-yy[i]) fo

subplot(4,5,12)
plot(tt,yy,'b-',tt,uu_AM2,'r-D')
title('AM2 - max(|err|)=%1.3f'%(max([abs(uu_AM2[i]-yy[i]) for

subplot(4,5,13)
plot(tt,yy,'b-',tt,uu_AM3,'r-D')
title('AM3 - max(|err|)=%1.3f'%(max([abs(uu_AM3[i]-yy[i]) for

subplot(4,5,14)
plot(tt,yy,'b-',tt,uu_AM4,'r-D')
title('AM4 - max(|err|)=%1.3f'%(max([abs(uu_AM4[i]-yy[i]) for
```

```
subplot(4,5,15)
plot(tt,yy,'b-',tt,uu_BDF2,'r-D')
title('BDF2 - max(|err|)=%1.3f'%(max([abs(uu_BDF2[i]-yy[i]) fo

subplot(4,5,16)
plot(tt,yy,'b-',tt,uu_BDF3,'r-D')
title('BDF3 - max(|err|)=%1.3f'%(max([abs(uu_BDF3[i]-yy[i]) fo

subplot(4,5,17)
plot(tt,yy,'b-',tt,uu_em,'r-D')
title('Euler modifie - max(|err|)=%1.3f'%(max([abs(uu_em[i]-yy

subplot(4,5,18)
plot(tt,yy,'b-',tt,uu_heun,'r-D')
title('Heun - max(|err|)=%1.3f'%(max([abs(uu_heun[i]-yy[i]) fo

subplot(4,5,19)
plot(tt,yy,'b-',tt,uu_AM2AB1,'r-D')
title('AM2AB1 - max(|err|)=%1.3f'%(max([abs(uu_AM2AB1[i]-yy[i]

subplot(4,5,20)
plot(tt,yy,'b-',tt,uu_AM3AB2,'r-D')
title('AM3AB2 - max(|err|)=%1.3f'%(max([abs(uu_AM3AB2[i]-yy[i]
```