*Université Saad Dahleb – Blida*

*Département Energies Renouvelables*

**TP Analyse Numérique**

**TP 09 : Résolution Equations Différentielles Ordinaires**

**d’Ordre Supérieur**

*Cas d’une solution a des racines réelles et racines complexes*

*Supervisé par : Pr MERZOUK*

*Réalisé par : BELKHIR Amine*

*Le document et tous les scripts, fonctions et les codes sont disponibles sur Github*

*Pour toute suggestion de modification, veuillez envisager d'effectuer une demande d'extraction (Pull-request)*

**Résumé**

Soit à résoudre l’équation différentielle ordinaire du type :

Tel que

Avec les conditions initiales

**Solution analytique**

*a/ On commence par résoudre l’équation sans second membre, soit :*

Pour cela on calcule les racines de la caractéristique de l’équation :

* Si les racines sont réelles la solution dite complémentaire est de la forme :

Les coefficients A et B sont donnés par les conditions aux limites.

* Si la racine est double, la solution complémentaire est de la forme :
* Si les racines sont imaginaires, la solution complémentaire est de la forme :

Les racines de la caractéristique étant données par

*b/ Solution avec second membre :*

La solution avec membre ou solution particulière est donnée par :

La solution générale est donnée par :

**Solution Numérique :**

L’équation peut se mettre sous la forme

En posant on obtient le système :

On transforme ce système en fonction sur MATLAB  et résoudrons à l’aide de ode23 et ode45

**Travail Demandé :**

***Application 1***

On a l’équation différentielle de second ordre :

1. **Solution Analytique**

Depuis l’O.D.E. on a l’équation auxiliaire :

D’où

Alors

Avec la condition au limite :

La solution devient alors :

1. **Solution Numérique**

Appliquant le changement de variable sur notre équation on obtient :

Transformons ce système en fonction sur Matlab :

function dy = app1( t,y )

dy=zeros(2,1);

dy(1)=y(2);

dy(2)=-2.\*dy(1)-y(1);

end

Ou bien on peut aussi résoudre ce système utilisant la méthode de Runge-Kutta 2 à la fois pour le système

Remarque : Chaque étape de la méthode doit être effectué successivement pour les deux équations de système (commençons par l’équation de première ordre après passons à la deuxième)

clc

clearvars

yp=1; y=1; dx=0.1; x=0:dx:10;

for i=1:length(x)-1

z(i)=yp(i);

zp(i)=-2.\*z(i)-y(i);

y12(i)=y(i)+0.5\*dx.\*zp(i);

zp12(i)=-2.\*z(i)-y12(i);

yp12(i)=yp(i)+0.5.\*dx.\*zp12(i);

z12(i)=yp(i);

yp(i+1)=yp(i)+dx.\*zp12(i);

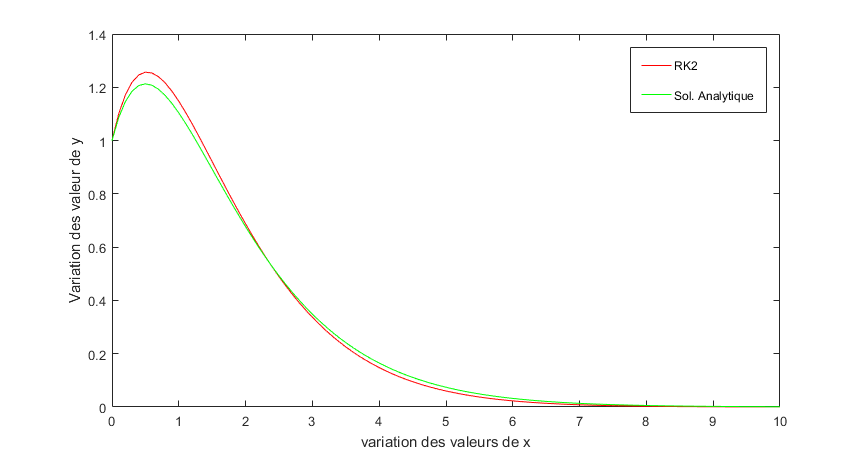
y(i+1)=y(i)+dx.\*z12(i);

end

yt=(1+2.\*x).\*exp(-x)

plot(x,y,'r',x,yt,'g')

Output :



Meme procedure pour Runge-Kutta 4 :

clc

clearvars

yp=1;y=1;dx=0.1;x=0:dx:10;

for i=1:length(x)-1

fp(i)=yp(i);

f(i)=-2.\*fp(i)-y(i);

ye12(i)=y(i)+0.5.\*dx.\*fp(i);

ype12(i)=yp(i)+0.5.\*dx.\*f(i);

fp12(i)=ype12(i);

f12(i)=-2.\*fp12(i)-ye12(i);

yee12(i)=y(i)+0.5.\*dx.\*fp12(i);

ypee12(i)=yp(i)+0.5.\*dx.\*f12(i);

ffp12(i)=ypee12(i);

ff12(i)=-2.\*ffp12(i)-yee12(i);

ye1(i)=y(i)+dx.\*ffp12(i);

ype1(i)=yp(i)+dx.\*ff12(i);

fp1(i)=ype1(i);

f1(i)=-2.\*fp1(i)-ye1(i);

fpbar(i)=(fp(i)+2.\*fp12(i)+2.\*ffp12(i)+fp1(i))./6;

fbar(i)=(f(i)+2.\*f12(i)+2.\*ff12(i)+f1(i))./6;

yp(i+1)=yp(i)+dx.\*fbar(i);

y(i+1)=y(i)+dx.\*fpbar(i);

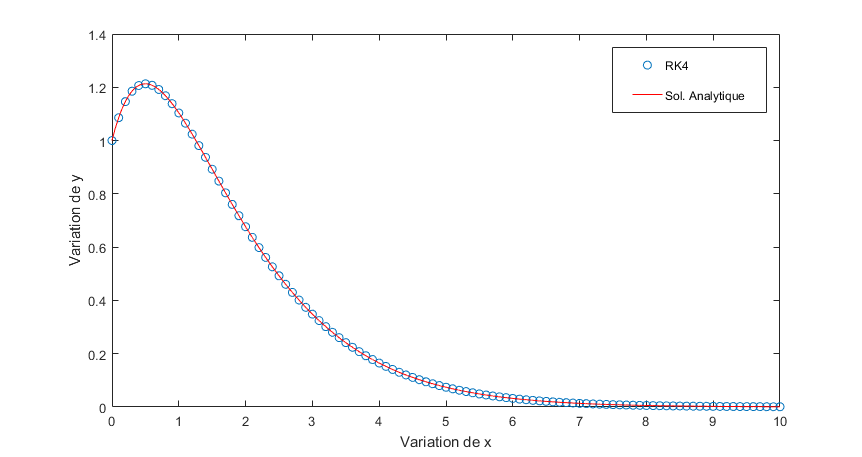
end

xx=0:dx./100:10;

yt=(1+2.\*xx).\*exp(-xx);

plot(x,y,'r',xx,yt,'g')

Output :



Maintenant utilisons built-in fonctions ***ode23*** et ***ode45***

Le script :

clc

clearvars

[X,Y]=ode23('app1',[0 10],[1 1]);

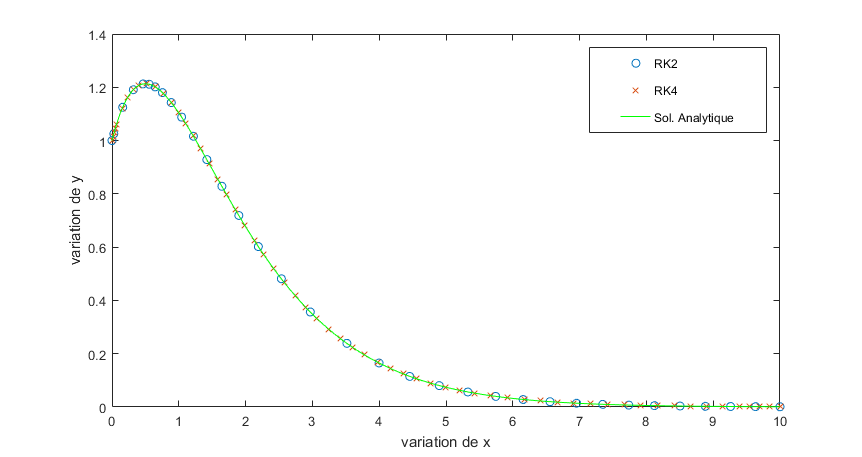
[x,y]=ode45('app1',[0 10] ,[1 1]);

xx=0:0.01:10;

yt=(1+2.\*xx).\*exp(-xx);

plot(X,Y(:,1),'o',x,y(:,1),'x',xx,yt,'g')

Output :



Commentaire :

On remarque depuis les 3 graphs qu’utilisons les méthodes RK2 et RK4 nous a donné des résultats presque confondu (le décalage dans le script de RK2 par rapport à la fonction RK2 est due au pas considéré, plus que le pas est petit plus que la précision augmente)

***Application 2***

On a l’équation différentielle de second ordre :

1. **Solution Analytique**

Depuis l’O.D.E. on a l’équation auxiliaire :

D’où

Alors

Avec la condition au limite :

La solution devient alors :

1. **Solution Numérique**

De même façon que le première exemple :

Le système :

La fonction :

function dy = app2( t,y )

dy=zeros(2,1);

dy(1)=y(2);

dy(2)=dy(1)-0.75.\*y(1);

end

Le script :

clc

clearvars

[X,Y]=ode23('app2',[0 10],[1 (1+sqrt(2))./2]);

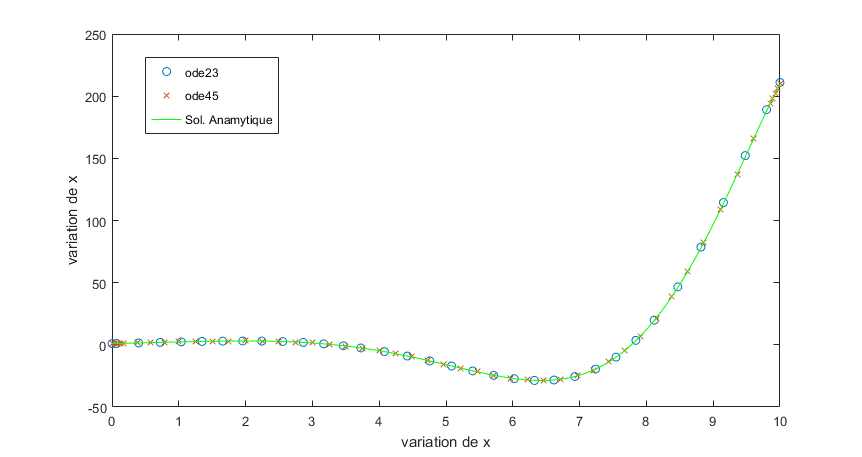
[x,y]=ode45('app2',[0 10],[1 (1+sqrt(2))./2]);

xx=0:0.01:10;

yt=(cos((sqrt(2)/2).\*xx)+sin((sqrt(2)/2).\*xx)).\*exp(xx./2);

plot(X,Y(:,1),'o',x,y(:,1),'x', xx,yt,'g')

Output :



Bonus :

On peut construire aussi notre script en fonction noté ***odeBEL2( )***  « 2 pour l’ordre »

La seule différence avec les Ode23,45 c’est que chaque équation de système doit être construise en fonction séparément

Syntaxe  : [x,y]= odeBEL2('Fn1','Fn2',[0 10],[1 1])

Les fonction de système  :

function dy=Fn1(t,y)

dy=y;

function dy=Fn2(t,y,y2)

dy=y2-0.75.\*y;

La fonction odeBEL2 :

function [ x,y ] = odeBEL2( fn1,fn2,t,y)

fn1=str2func(fn1);

fn2=str2func(fn2);

yp=y(1);

y=y(2);

dx=0.1;

x=0:dx:10;

for i=1:length(x)-1

fp(i)=fn1(x(i),yp(i));

f(i)=fn2(x(i),y(i),fp(i));

ye12(i)=y(i)+0.5.\*dx.\*fp(i);

ype12(i)=yp(i)+0.5.\*dx.\*f(i);

fp12(i)=fn1(x(i),ype12(i));

f12(i)=fn2(x(i),ye12(i),fp12(i));

yee12(i)=y(i)+0.5.\*dx.\*fp12(i);

ypee12(i)=yp(i)+0.5.\*dx.\*f12(i);

ffp12(i)=fn1(x(i),ypee12(i));

ff12(i)=fn2(x(i),yee12(i),ffp12(i));

ye1(i)=y(i)+dx.\*ffp12(i);

ype1(i)=yp(i)+dx.\*ff12(i);

fp1(i)=fn1(x(i),ype1(i));

f1(i)=fn2(x(i),ye1(i),fp1(i));

fpbar(i)=(fp(i)+2.\*fp12(i)+2.\*ffp12(i)+fp1(i))./6;

fbar(i)=(f(i)+2.\*f12(i)+2.\*ff12(i)+f1(i))./6;

yp(i+1)=yp(i)+dx.\*fbar(i);

y(i+1)=y(i)+dx.\*fpbar(i);

end

y(1,:)=y;

y(2,:)=yp;

end

le script :

clc

clearvars

[x,y]=odeBEL2('Fn1','Fn2',[0 10],[1 (1+sqrt(2))./2])

[X,Y]=ode45('app2',[0 10],[1 (1+sqrt(2))./2]);

xx=0:0.01:10;

yt=(cos((sqrt(2)/2).\*xx)+sin((sqrt(2)/2).\*xx)).\*exp(xx./2);

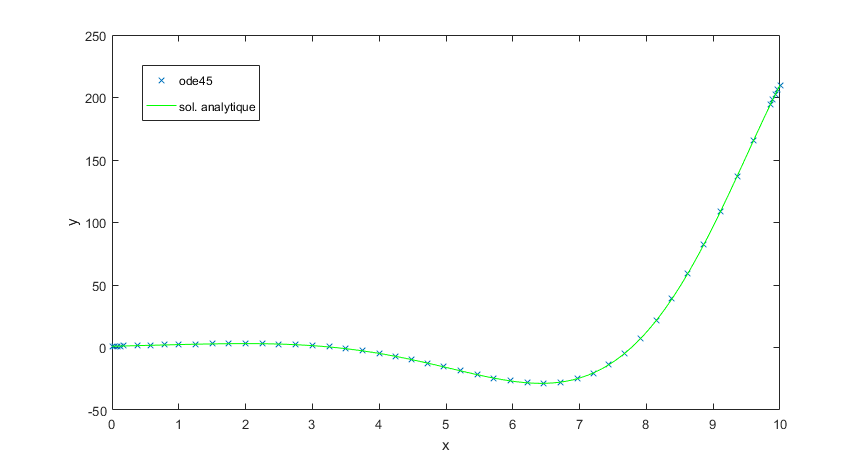
figure(1)

plot(X,Y(:,1),'x',xx,yt,'g')

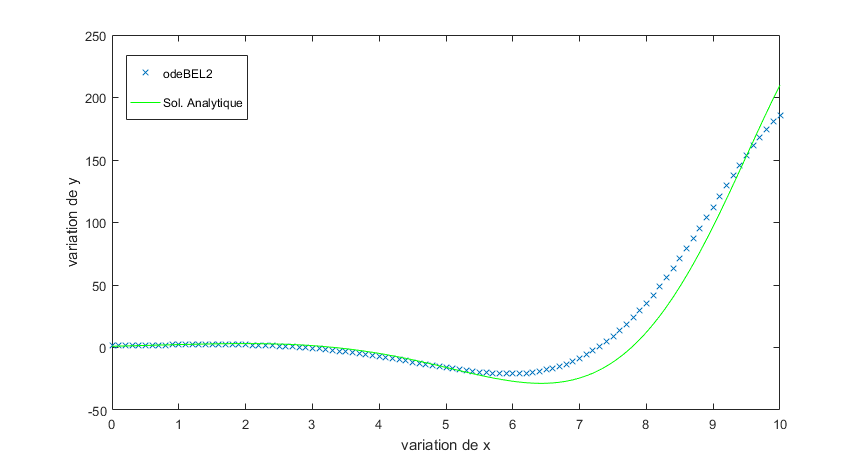
figure(2)

plot(x,y(1,:),'x',xx,yt,'g')

output :

fig (1) 

fig(2)



Commentaire : on remarque que pour cette ODE la fonction odeBEL2 montre une divergence.

**Résultats :**

Depuis les résultats obtenu on déduire que la résolution numérique des équations différentielle d’ordre supérieure (pas de seconde membre) est très précis et ça peut correspondre à celle de la solution analytique.