IV.6. Exercices d'application sous Matlab

Exercice IV.6.1: (Méthode d'Euler)

Résoudre l'équation différentielle suivante en utilisant la méthode d'Euler:

$y' = -2x + 5e^{(-4t)}$, avec $y(0) = 1$ sur l'intervalle $[0, 2]$.

Solution :

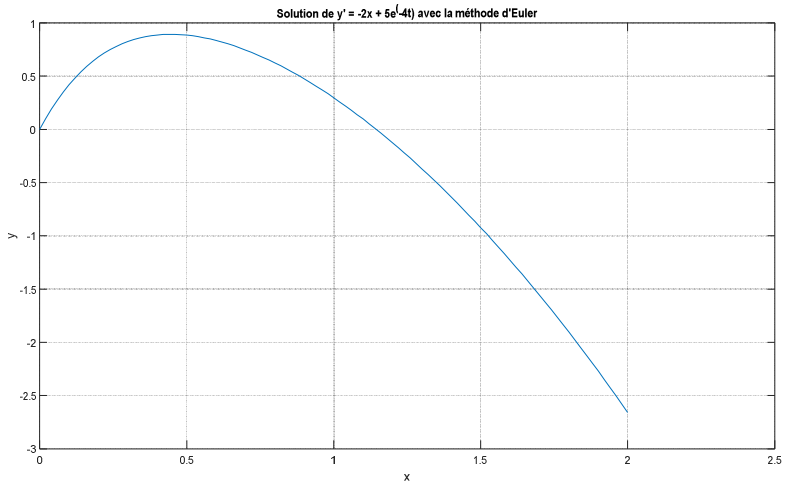
```
% Définir les paramètres
a = 0; % borne inférieure de l'intervalle
b = 2; % borne supérieure de l'intervalle
N = 100; % nombre de points de discrétisation
% Calculer la taille du pas de discrétisation
h = (b - a) / N;
% Initialiser les variables
x = zeros(N+1, 1); % vecteur des points x
y = zeros(N+1, 1); % vecteur des points y

% Définir les conditions initiales
x(1) = a;
y(1) = 0;

% Appliquer la méthode d'Euler pour résoudre l'équation
différentielle
for i = 1:N
    x(i+1) = x(i) + h;
    y(i+1) = y(i) + h*(-2*x(i) + 5*exp(-4*x(i)));
end
% Tracer la solution
plot(x, y);
grid on ;
xlabel('x');
ylabel('y');
title('Solution de y'' = -2x + 5e^(-4t) avec la méthode
d''Euler');
```

Dans ce code, nous avons défini les paramètres de l'intervalle, le nombre de points de discrétisation, ainsi que la condition initiale. Nous avons ensuite calculé la taille du pas de discrétisation, initialisé les variables, et appliqué la méthode d'Euler en utilisant une boucle for.

Enfin, nous avons tracé la solution en utilisant la fonction plot( ) de Matlab.



Exercice IV.6.2: (Méthode de Runge-Kutta**)**

Résoudre l'équation différentielle suivante en utilisant la méthode de Runge-Kutta d'ordre 4 :$y' = -y + t^2$.

Solution :

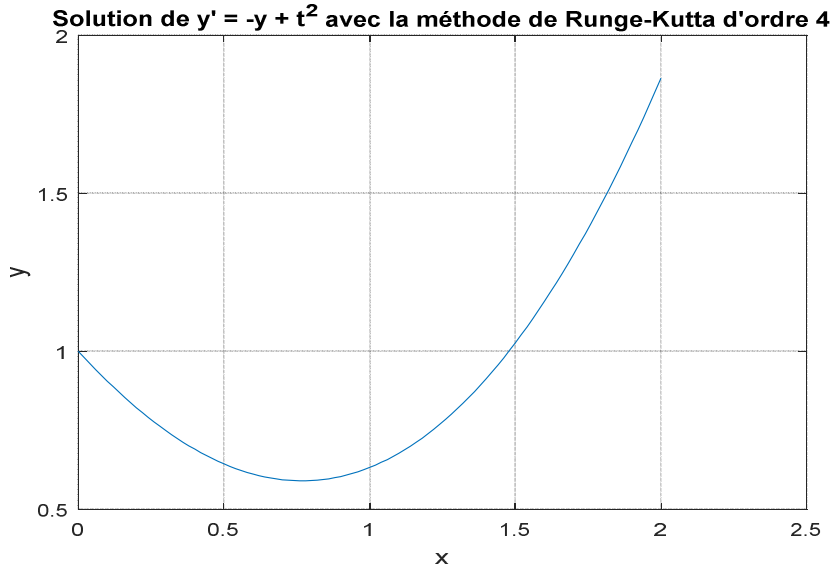
```
% Définir les paramètres
a = 0; % borne inférieure de l'intervalle
b = 2; % borne supérieure de l'intervalle
N = 100; % nombre de points de discrétisation
% Calculer la taille du pas de discrétisation
h = (b - a) / N;
% Initialiser les variables
x = zeros(N+1, 1); % vecteur des points x
y = zeros(N+1, 1); % vecteur des points y

% Définir les conditions initiales
x(1) = a;
y(1) = 0;

% Appliquer la méthode d'Euler pour résoudre l'équation
différentielle
for i = 1:N
    x(i+1) = x(i) + h;
    y(i+1) = y(i) + h*(-2*x(i) + 5*exp(-4*x(i)));
end
% Tracer la solution
plot(x, y);
grid on ;
xlabel('x');
ylabel('y');
```

```
% Tracer la solution
plot(x, y);
grid on;
xlabel('x');
ylabel('y');
title('Solution de y'' = -y + t^2 avec la méthode de Runge-Kutta d''ordre 4');
```

Dans ce code, nous avons défini les paramètres de l'intervalle, le nombre de points de discrétisation, ainsi que la condition initiale. Nous avons ensuite calculé la taille du pas de discrétisation, initialisé les variables, et appliqué la méthode de Runge-Kutta d'ordre 4 en utilisant une boucle for.

Notez que la méthode de Runge-Kutta d'ordre 4 nécessite le calcul de quatre pentes (k1, k2, k3 et k4) à chaque itération, qui sont ensuite combinées pour mettre à jour la valeur de y. La méthode de Runge-Kutta d'ordre 4 est plus précise que la méthode d'Euler, mais elle est également plus coûteuse en termes de calcul.

Enfin, nous avons tracé la solution en utilisant la fonction plot( ) de Matlab.