1 Numerical Methods for ODEs

Consider the initial value problem of ordinary differential equation, $f(t,u)$ is continuous on area G : $0 \leqslant t \leqslant T$, $|u| < \infty$, $u = u(t)$ satisfy the equation

$$\begin{cases} \dfrac{du}{dt} = f(t,u), & 0 < t \leqslant T\,, \\ u(0) = 0\,. \end{cases}$$

generally, f satisfy Lipschitz condition: $|f(t,u_1) - f(t,u_2)| \leqslant L|u_1 - u_2|$.

1.1 Euler Method

Euler method scheme:

$$u_{n+1} = u_n + hf(t_n\,, u_n)$$

Example 1.1

$$\begin{cases} \dfrac{du}{dt} = t^2 + t - u\,, & t \in [0\,,1] \\ u(0) = 0\,. \end{cases}$$

```
% Euler1.m
% Euler method for a first-order ODE
% u'(t)=t^2+t-u, t \in [0,1]
% Initial value : u(0)=0 ;
% Exact solution : u(t)=-exp(-t)+t^2-t+1.
clear all
h=0.1;
x=0:h:1;                              % function interval
n=length(x)-1;
u(1)=0;                               % initial value
fun=@(t,u) t.^2+t-u;                  % RHS
for i=1:n
    u(i+1)=u(i)+h.*fun(x(i),u(i));
end
ue=-exp(-x)+x.^2-x+1;                 % exact solution
plot(x,ue,'b-',x,u,'r+','LineWidth',1.5)
xlabel('x','fontsize', 16), ylabel('y','fontsize',16,'Rotation',0)
legend('Exact ','Numerical','location','North')
title('Euler Method','fontsize',14)
set(gca,'fontsize',14)
```

1.2 Modified Euler Method

Modified Euler method scheme:

$$u_{n+1} = u_n + \frac{h}{2}[f(t_n, u_n) + f(t_{n+1}, u_{n+1})]$$

Example 1.2

$$\begin{cases} \dfrac{du}{dt} = t^2 + t - u, & t \in [0,1] \\ u(0) = 0. \end{cases}$$

```
% Euler2.m
% Modified Euler method for a first-order ODE
% u'=t^2+t-u,  t \in [0,1]
% Initial value : u(0)=0
% Exact solution : u(t)=-exp(-t)+t^2-t+1.
clear all
h=0.1;
x=0:h:1;                    % function interval
n=length(x)-1;
u(1)=0;                     % initial value
fun=@(t,u) t.^2+t-u;        % RHS
for i=1:n
    k1=fun(x(i),u(i));
    k2=fun(x(i+1),u(i)+h*k1);
    u(i+1)=u(i)+(h/2)*(k1+k2);
end
ue=-exp(-x)+x.^2-x+1;       % exact solution
plot(x,ue,'b-',x,u,'r+','LineWidth',1.5)
xlabel('x','fontsize', 16), ylabel('y','fontsize',16,'Rotation',0)
legend('Exact','Numerical','location','North')
title('Modified Euler Method','fontsize',14)
set(gca,'fontsize',14)
```

1.3 Runge-Kutta Method

Runge-Kutta method scheme:

$$\begin{cases} u_{n+1} = u_n + \dfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = f(t_n, u_n) \\ k_2 = f\left(t_n + \dfrac{h}{2}, u_n + \dfrac{1}{2}k_1\right) \\ k_3 = f\left(t_n + \dfrac{h}{2}, u_n + \dfrac{1}{2}k_2\right) \\ k_4 = f(t_n + h, u_n + k_3) \end{cases}$$

Example 1.3

$$\begin{cases} \dfrac{du}{dt} = t^2 + t - u, \quad t \in [0, 1] \\ u(0) = 0\,. \end{cases}$$

```
% RungeKutta.m
% Runge-Kutta method for a first-order ODE
% u'=t^2+t-u,   t \in [0,1]
% Initial value : u(0)=0
% Exact : u(t)=-exp(-t)+t^2-t+1.
clear all
h=0.1;
x=0:h:1;                        % function interval
n=length(x)-1;
u(1)=0;                         % initial value
fun=@(t,u) t.^2+t-u;            % RHS
for i=1:n
    k1=fun(x(i),u(i));
    k2=fun(x(i)+h./2,u(i)+h.*k1/2);
    k3=fun(x(i)+h./2,u(i)+h.*k2/2);
    k4=fun(x(i)+h,u(i)+h.*k3);
    u(i+1)=u(i)+h.*(k1+2.*k2+2.*k3+k4)./6;
end
ue=-exp(-x)+x.^2-x+1;           % exact solution
plot(x,ue,'b-',x,u,'r+','LineWidth',1.5)
xlabel('x','fontsize', 16), ylabel('y','fontsize',16,'Rotation',0)
legend('Exact','Numerical','location','North')
title('Runge-Kutta Method','fontsize',14)
set(gca,'fontsize',14)
```

The general s-stage Runge-Kutta method for the problem

$$y' = f(x,y), \quad y(a) = \eta, \quad f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$$

is defined by

$$\text{(I)} \begin{cases} y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i \\ k_i = f(x_n + c_i h, y_n + h \sum_{j=1}^{s} a_{ij} k_j), \quad i = 1, 2, \ldots, s \end{cases}$$

Assume that the following (the row-sum condition) holds

$$c_i = \sum_{j=1}^{s} a_{ij}, \quad i = 1, 2, \ldots, s$$

It is convenient to display the coefficients as a Butcher array:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & & & \vdots \\ c_{s1} & a_{s2} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

$$c = [c_1, c_2, \ldots, c_s]^T, \quad b = [b_1, b_2, \ldots, b_s]^T, \quad A = \left(a_{ij}\right)_s$$

$$\text{(II)} \begin{cases} y_{n+1} = y_n + h \sum_{i=1}^{s} b_i f\,(x_n + c_i h, Y_i) \\ Y_i = y_n + h \sum_{j=1}^{s} a_{ij} f\,\left(x_n + c_j h, Y_j\right), \quad i = 1, 2, \ldots, s \end{cases}$$

The forms (I) and (II) are seen to be equivalent if we make the interpretation

$$k_i = f\,(x_n + c_i h, Y_i), \quad i = 1, 2, \ldots, s$$

Implicit Runge-Kutta Method (Gauss Method) 2 stage order 4:

$$
\text{(I)}\begin{cases}
y_{n+1} = y_n + \dfrac{1}{2}\left(K_1 + K_2\right), \quad n = 0, 1, \cdots, N-1, \\
K_1 = hf\left(t_n + (\dfrac{1}{2} - \dfrac{\sqrt{3}}{6})h, y_n + \dfrac{1}{4}K_1 + (\dfrac{1}{4} - \dfrac{\sqrt{3}}{6})K_2\right), \\
K_2 = hf\left(t_n + (\dfrac{1}{2} + \dfrac{\sqrt{3}}{6})h, y_n + (\dfrac{1}{4} + \dfrac{\sqrt{3}}{6})K_1 + \dfrac{1}{4}K_2\right).
\end{cases}
$$

Butcher array

$$
\begin{array}{c|cc}
\frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\
\frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\
\hline
 & \frac{1}{2} & \frac{1}{2}
\end{array}
$$

$$
\text{(II)}\begin{cases}
y_{n+1} = y_n + h\displaystyle\sum_{i=1}^{2} b_i f\left(x_n + c_i h, Y_i\right), \\
Y_1 = y_n + h\displaystyle\sum_{j=1}^{2} a_{1j} f\left(x_n + c_j h, Y_j\right), \\
Y_2 = y_n + h\displaystyle\sum_{j=1}^{2} a_{2j} f\left(x_n + c_j h, Y_j\right),
\end{cases}
$$

Example 1.4

$$
\begin{cases}
\dfrac{du}{dt} = u, \quad t \in [0, 1] \\
u(0) = 1.
\end{cases}
$$

```
% IRK2s_order.m
% Implicit Runge-Kutta(Gauss method) 2 stage and order 4
% u'=u in [0,1] with initial condition u(0)=1
% exact solution: ue=exp(x)
clear all
Nvec=[10 50 100 200 500 1000];
MErr=[];
for n=1:length(Nvec)
    N=Nvec(n);  h=1/N;
    x=[0:h:1];
    u(1)=1;
    X0=[1;1];
    % Newton iteration
    for i=1:N
        k=u(i);
        r=X0; tol=1;
        while tol>1.0e-6
            X=r;
            D=[1-0.25*h,-h*(0.25-(sqrt(3))/6);...
            -h*( 0.25+(sqrt(3))/6),1-h*0.25];    % Jacobian matrix
            F=[X(1)-k-h*(0.25*X(1)+(0.25-(sqrt(3))/6)*X(2));...
            X(2)-k-h*((0.25+(sqrt(3))/6)*X(1)+0.25*X(2))];   % RHS
            r=X-D\F;
            tol=norm(r-X);
        end
        k1=r(1);  k2=r(2);
        u(i+1)=k+(h/2)*(k1+k2);
    end
    ue=exp(x);                % exact solution
    Merr=max(abs(u-ue));    % maximum error
    MErr=[MErr,Merr];
end
plot(log10(Nvec),log10(MErr),'r*-','LineWidth',1.5)
hold on,
plot(log10(Nvec), log10(Nvec.^(-4)), '--')
grid on,
xlabel('N','fontsize', 16), ylabel('Error','fontsize',16)
title('Convergence order of Gauss method ','fontsize',14)
set(gca,'fontsize',14)
for i=1:length(Nvec)-1      % computating convergence order
    order(i)=-log(MErr(i)/MErr(i+1))/(log(Nvec(i)/Nvec(i+1)));
end
MErr
order
```