

# 1 Numerical Methods for ODEs

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

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

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

## 1.1 Euler Method

Euler method scheme:

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

### Example 1.1

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

```
1 % Euler1.m
2 % Euler method for the ODE model
3 % u'(t)=t^2+t-u, t \in [0,1]
4 % Initial condition : u(0)=0 ;
5 % Exact solution : u(t)=-exp(-t)+t^2-t+1.
6 clear all
7 h=0.1;
8 x=0:h:1; % function interval
9 n=length(x)-1;
10 u(1)=0; % initial value
11 fun=@(t,u) t.^2+t-u; % RHS
12 for i=1:n
13     u(i+1)=u(i)+h.*fun(x(i),u(i));
14 end
15 ue=-exp(-x)+x.^2-x+1; % exact solution
16 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1.5)
17 xlabel('x','fontsize',16), ylabel('y','fontsize',16,'Rotation',0)
18 legend('Exact ', 'Numerical', 'location', 'North')
19 title('Euler Method','fontsize',14)
20 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

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

```
1 % Euler2.m
2 % Modified Euler method for the ODE model
3 % u'=t^2+t-u, t \in [0,1]
4 % Initial condition : u(0)=0
5 % Exact solution : u(t)=-exp(-t)+t^2-t+1.
6 clear all
7 h=0.1;
8 x=0:h:1; % function interval
9 n=length(x)-1;
10 u(1)=0; % initial value
11 fun=@(t,u) t.^2+t-u; % RHS
12 for i=1:n
13     k1=fun(x(i),u(i));
14     k2=fun(x(i+1),u(i)+h*k1);
15     u(i+1)=u(i)+(h/2)*(k1+k2);
16 end
17 ue=-exp(-x)+x.^2-x+1; % exact solution
18 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1.5)
19 xlabel('x','fontsize',16), ylabel('y','fontsize',16,'Rotation',0)
20 legend('Exact','Numerical','location','North')
21 title('Modified Euler Method','fontsize',14)
22 set(gca,'fontsize',14)
```

## 1.3 Runge-Kutta Method

Runge-Kutta method scheme:

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

### Example 1.3

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

```
1 % RungeKutta.m
2 % Runge-Kutta method for the ODE model
3 % u'=t^2+t-u, t \in [0,1]
4 % Initial condition : u(0)=0
5 % Exact : u(t)=-exp(-t)+t^2-t+1.
6 clear all
7 h=0.1;
8 x=0:h:1; % function interval
9 n=length(x)-1;
10 u(1)=0; % initial value
11 fun=@(t,u) t.^2+t-u; % RHS
12 for i=1:n
13     k1=fun(x(i),u(i));
14     k2=fun(x(i)+h./2,u(i)+h.*k1/2);
15     k3=fun(x(i)+h./2,u(i)+h.*k2/2);
16     k4=fun(x(i)+h,u(i)+h.*k3);
17     u(i+1)=u(i)+h.*(k1+2.*k2+2.*k3+k4)./6;
18 end
19 ue=-exp(-x)+x.^2-x+1; % exact solution
20 plot(x,ue,'b-',x,u,'r+', 'LineWidth',1.5)
21 xlabel('x','fontsize',16), ylabel('y','fontsize',16,'Rotation',0)
22 legend('Exact','Numerical','location','North')
23 title('Runge-Kutta Method','fontsize',14)
24 set(gca,'fontsize',14)
```

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

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

is defined by

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

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

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

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

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

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

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

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

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

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

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

Butcher array

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

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

**Example 1.4**

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

```

1 % IRK2s_error.m
2 % Implicit Runge-Kutta (Gauss method) 2 stage and order 4
3 % u'=u in [0,1] with initial condition u(0)=1
4 % exact solution: ue=exp(x)
5 clear all
6 Nvec=[10 50 100 200 500 1000];
7 Err=[];
8 for n=1:length(Nvec)
9     N=Nvec(n); h=1/N;
10    x=[0:h:1];
11    u(1)=1;
12    X0=[1;1];
13    % Newton iteration
14    for i=1:N
15        k=u(i);
16        r=X0; tol=1;
17        while tol>1.0e-6
18            X=r;
19            D=[1-0.25*h, -h*(0.25-(sqrt(3))/6);...
20              -h*(0.25+(sqrt(3))/6), 1-h*0.25]; % Jacobian matrix
21            F=[X(1)-k-h*(0.25*X(1)+(0.25-(sqrt(3))/6)*X(2));...
22              X(2)-k-h*((0.25+(sqrt(3))/6)*X(1)+0.25*X(2))]; % RHS
23            r=X-D\F;
24            tol=norm(r-X);
25        end
26        k1=r(1); k2=r(2);
27        u(i+1)=k+(h/2)*(k1+k2);
28    end
29    ue=exp(x); % exact solution
30    err=max(abs(u-ue)); % maximum error
31    Err=[Err,err];
32 end
33 plot(log10(Nvec), log10(Err), 'ro-', 'MarkerFaceColor', 'w', 'LineWidth', 1.5)
34 hold on,
35 plot(log10(Nvec), log10(Nvec.^(-4)), '--')
36 grid on,
37 xlabel('log_{10}N', 'fontsize', 16), ylabel('log_{10}Error', 'fontsize', 16)
38 title('Convergence order of Gauss method ', 'fontsize', 14)
39 set(gca, 'fontsize', 14)
40 for i=1:length(Nvec)-1 % computing convergence order
41     order(i)=-log(Err(i)/Err(i+1))/(log(Nvec(i)/Nvec(i+1)));
42 end
43 Err
44 order

```