Section 5.4 Runge-Kutta Methods

Motivation

- How to design a high-order accurate method without knowledge of derivatives of $f(t,y)$, *which are cumbersome to compute*.
- Recall Taylor method of order 2:

$$w_{i+1} = w_i + hT^{(2)}(t_i, w_i), \text{ where}$$
$$T^{(2)}(t_i, w_i) = f(t_i, w_i) + \frac{h}{2}\left(\frac{\partial f}{\partial t}(t_i, w_i) + \frac{\partial f}{\partial y}(t_i, w_i) f(t_i, w_i)\right)$$

- Main idea: design a method of the form

$w_{i+1} = w_i + h\, a_1 f(t_i + \alpha_1, w_i + \beta_1)$, then determine the coefficients a_1, α_1, β_1 such that

$$|a_1 f(t + \alpha_1, y + \beta_1) - T^{(2)}(t, y)| = O(h^2)$$

Derivation of Runge-Kutta Method of Order Two

a) $T^{(2)}(t,y) = f(t,y) + \frac{h}{2}\frac{\partial f}{\partial t}(t, y(t)) + \frac{h}{2}\frac{\partial f}{\partial y}(t, y(t)) \cdot f(t, y(t)).$

b) Using Talyor theorem of two variables (**Thm 5.13 in textbook**),

$$a_1 f(t+\alpha_1, y+\beta_1) = a_1 f(t,y) + a_1\alpha_1 \frac{\partial f}{\partial t}(t,y) + a_1\beta_1 \frac{\partial f}{\partial y}(t,y)$$

$$+a_1 R_1(t+\alpha_1, y+\beta_1)$$ <-- *remainder*

- Matching coefficients in a) and b) $--\rightarrow$

$$a_1 = 1, \qquad \alpha_1 = \frac{h}{2}, \qquad \beta_1 = \frac{h}{2} f(t,y).$$

- The remainder term $a_1 R_1(t+\alpha_1, y+\beta_1) = O(h^2)$ ---- >

$$|a_1 f(t+\alpha_1, y+\beta_1) - T^{(2)}(t,y)| = O(h^2)$$

➔➔➔The new method is of order 2!!! (same order as Taylor 2)

Midpoint Method

- The midpoint method:

$$w_{i+1} = w_i + hf\left(t_i + \frac{h}{2}, w_i + \frac{h}{2} f(t_i, w_i)\right),$$

$$\text{for each } i = 0, 1, 2, \cdots, N-1.$$

Remark: Local truncation error of Midpoint method is $O(h^2)$.
The method is second-order accurate.

Two stage formula of Midpoint Method

$$\begin{cases} k_1 = f(t_i, w_i) \\ k_2 = f\left(t_i + \frac{h}{2}, w_i + \frac{h}{2} k_1\right) \end{cases}$$

$$w_{i+1} = w_i + hk_2$$

$$\text{for each } \ i = 0, 1, 2, \cdots, N-1.$$

- **Example 2**. Use the Midpoint method with $N = 2$ to solve the IVP

$$y' = y - t^2 + 1, \qquad 0 \ \le t \le 2, \qquad y(0) = 0.5.$$

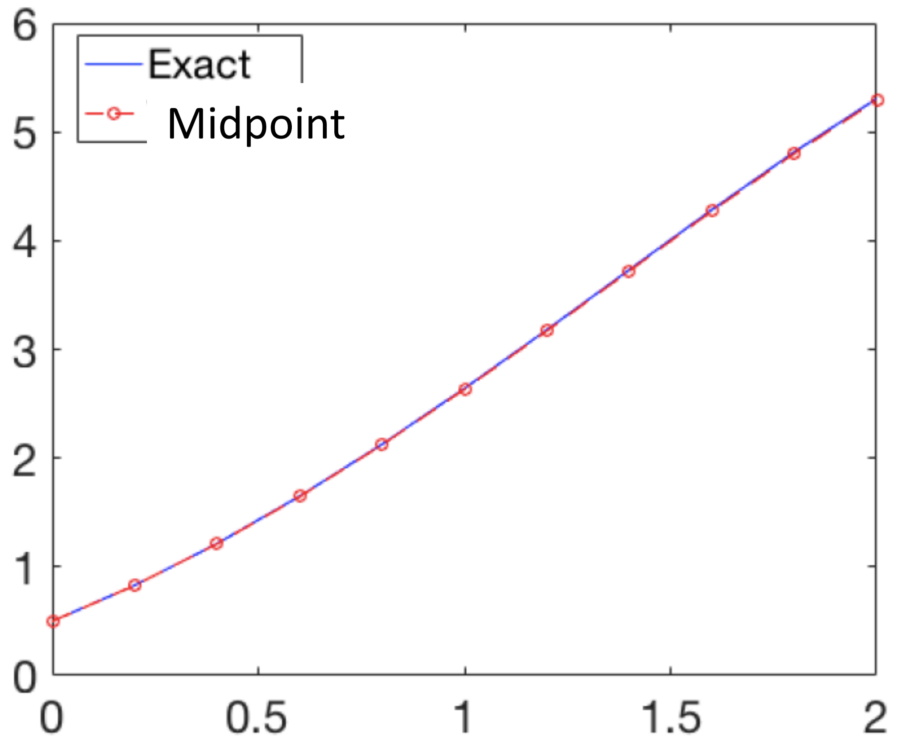
(MATLAB) Implement the method using $h = 0.2$. Record the maximum error.

```matlab
% inputs
f = @(t,y) y - t.^2 +1;
tend = 2; y0 = 0.5; h = 0.2;
yex = @(t) (t+1).^2-0.5*exp(t); % exact solution

% Midpoint method
tGrid = [0:h:tend]; N = length(tGrid)-1;
wGrid = zeros(1,N+1); wGrid(1) = y0; % initial data
for i = 1:N
    ti = tGrid(i); wi = wGrid(i);
    k1 = f(ti,wi);
    k2 = f(ti+h/2, wi+h/2*k1);
    wGrid(i+1) = wi + h*k2; % update
end
% print the max err
err = max(abs(yex(tGrid)-wGrid));
fprintf('\n max err: %.3e\n', err)

plot(tGrid, yex(tGrid),'b', tGrid, wGrid, 'ro--')
legend('Exact', 'Taylor 2','Location','Best')
set(gca, 'FontSize',24)
shg
```

max err: 1.510e-02



Modified Euler Method

- This is another Runge-Kutta method of order two:

The modified Euler method:

$$w_{i+1} = w_i + \frac{h}{2}\left[f(t_i, w_i) + f\big(t_{i+1}, w_i + hf(t_i, w_i)\big)\right]$$

$$\text{for each } \; i = 0, 1, 2, \cdots, N-1.$$

Remark: Local truncation error of Modified Euler method is $O(h^2)$.

Two stage formula of Modified Euler Method

$$\begin{cases} k_1 = f(t_i, w_i) \\ k_2 = f(t_i + h, w_i + hk_1) \end{cases}$$

$$w_{i+1} = w_i + \frac{h}{2}[k_1 + k_2],$$

$$\text{for each } i = 0, 1, 2, \cdots, N-1.$$

Example 2. Use the modified Euler method with $N = 2$ to solve the IVP

$$y' = y - t^2 + 1, \qquad 0 \le t \le 2, \qquad y(0) = 0.5.$$

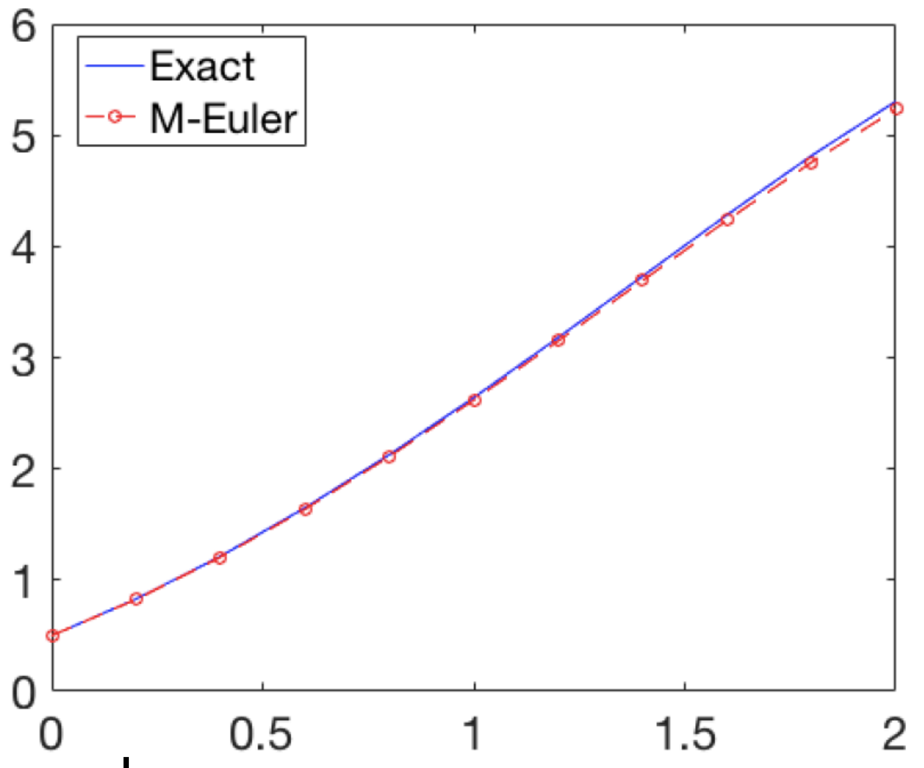
(MATLAB) Implement the method using $h = 0.2$. Record the maximum error.

```matlab
% inputs
f = @(t,y) y - t.^2 +1;
tend = 2; y0 = 0.5; h = 0.2;
yex = @(t) (t+1).^2-0.5*exp(t); % exact solution

% Modified Euler method
tGrid = [0:h:tend]; N = length(tGrid)-1;
wGrid = zeros(1,N+1); wGrid(1) = y0; % initial data
for i = 1:N
    ti = tGrid(i); wi = wGrid(i);
    k1 = f(ti,wi);
    k2 = f(ti+h, wi+h*k1);
    wGrid(i+1) = wi + 0.5*h*(k1+k2); % update
end
% print the max err
err = max(abs(yex(tGrid)-wGrid));
fprintf('\n max err: %.3e\n', err)

plot(tGrid, yex(tGrid),'b', tGrid, wGrid, 'ro--')
legend('Exact', 'M-Euler','Location','Best')
set(gca, 'FontSize',24)
shg
```

```
max err: 7.242e-02
```



Runge-Kutta Order Four (RK4)

$$\begin{cases} k_1 = f(t_i, w_i) \\ k_2 = f\left(t_i + \frac{h}{2}, w_i + \frac{h}{2}k_1\right) \\ k_3 = f(t_i + \frac{h}{2}, w_i + \frac{h}{2}k_2) \\ k_4 = f(t_i + h, w_i + h\, k_3) \end{cases},$$

$$w_{i+1} = w_i + \frac{h}{6}[k_1 + 2k_2 + 2k_3 + k_4],$$

for each $i = 0, 1, 2, \cdots, N-1.$

Remark: RK4 has four stages, and is **the** most widely used RK method. Its local truncation error is $O(h^4)$., $i.e.$, fourth order accurate.

Example 3. Use RK4 with $N = 2$ to solve the IVP

$$y' = y - t^2 + 1, \qquad 0 \le t \le 2, \qquad y(0) = 0.5.$$

(MATLAB) Implement the method using $h = 0.2$. Record the maximum error.

```
% inputs
f = @(t,y) y - t.^2 +1;
tend = 2; y0 = 0.5; h = 0.2;
yex = @(t) (t+1).^2-0.5*exp(t); % exact solution

% Modified Euler method
tGrid = [0:h:tend]; N = length(tGrid)-1;
wGrid = zeros(1,N+1); wGrid(1) = y0; % initial data
for i = 1:N
    ti = tGrid(i); wi = wGrid(i);
    k1 = f(ti,wi);
    k2 = f(ti+h/2, wi+h/2*k1);
    k3 = f(ti+h/2, wi+h/2*k2);
    k4 = f(ti+h, wi+h*k3);
    wGrid(i+1) = wi + h/6*(k1+2*k2+2*k3+k4); % update
end
% print the max err
err = max(abs(yex(tGrid)-wGrid));
fprintf('\n max err: %.3e\n', err)
```

```
max err: 1.089e-04
```