Week 4

程雨歌 12307110079

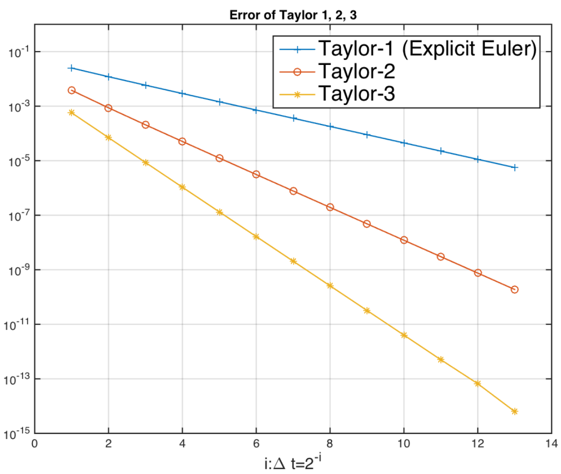
1 用 Taylor 级数法 (q=2,3) 计算 $\frac{\mathrm{d}u}{\mathrm{d}t} = u - u^2$，并与 Euler 显格式比较（P74/1）

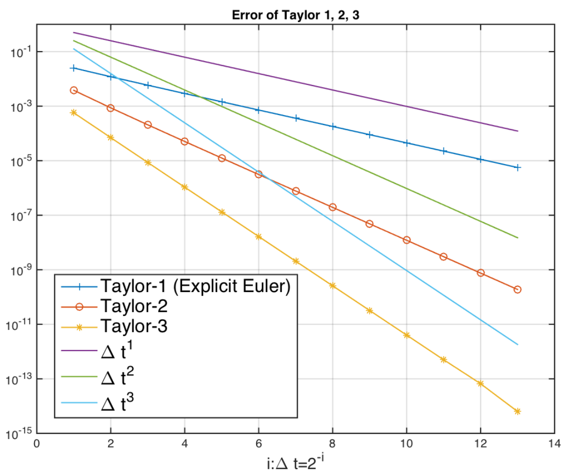
代码如下：

```
%% definitions
f = @(u) u - u .^ 2 ;
u = @(t, u0) 1 ./ ( ( 1 / u0  - 1 ) * exp( -t ) + 1 );
F = @(u) ( 1 - 2 .* u ) .* f(u) ;
G = @(u) f(u) .^ 2 .* (-2) ;
phi2 = @(u, dt) f(u) + dt ./ 2 .* F(u) ;
phi3 = @(u, dt) f(u) + dt ./ 2 .* F(u) ...
    + dt .^ 2 ./ 6 .* ( G(u) + (1 - 2 .* u) .* F(u) );
t0 = 0 ; T = 8 ;
u0 = 0.5 ; % u0 = 1.5

%% compute error of Explicit Euler, phi2, phi3
r = 13;
dts = 2 .^ -(1: r);
e1 = zeros(1, r);
e2 = zeros(1, r);
e3 = zeros(1, r);
for i = 1 : r
    dt = dts(i);
    tdt = t0 : dt : T ;
    n = length(tdt);
    uu = u( tdt , u0 );
    u1 = zeros ( 1 , n );
    u2 = zeros ( 1 , n );
    u3 = zeros ( 1 , n );
    u1(1) = u0 ; u2(1) = u0 ; u3(1) = u0 ;
    for j = 1 : n - 1
        u1(j + 1) = u1(j) + dt * f(u1(j));
        u2(j + 1) = u2(j) + dt * phi2(u2(j), dt);
        u3(j + 1) = u3(j) + dt * phi3(u3(j), dt);
    end
    e1(i) = max(abs(u1 - uu));
    e2(i) = max(abs(u2 - uu));
    e3(i) = max(abs(u3 - uu));
end

%% plot
semilogy((1: r), e1, '-+'); hold on; grid on;
semilogy((1: r), e2, '-o');
semilogy((1: r), e3, '-*');
semilogy((1: r), dts);
```

```
semilogy((1: r), dts .^ 2);
semilogy((1: r), dts .^ 3);
set(gca,'ytick', 10 .^ (-15 : 2 : -1));
title('Error of Taylor 1, 2, 3');
h = legend('Taylor-1 (Explicit Euler)', 'Taylor-2', 'Taylor-3', ...
    '\Delta t^1', '\Delta t^2', '\Delta t^3');
xlabel('i:\Delta t=2^{-i}', 'Fontsize', 14);
set(h, 'Fontsize', 16);

%% verify degree of convergence
figure(2);
for i = 1 : 4
    d1 = e1 ./ (dts .^ i);
    d2 = e2 ./ (dts .^ i);
    d3 = e3 ./ (dts .^ i);
    subplot(2,2,i);
    semilogy((1: r), d1); hold on;
    semilogy((1: r), d2);
    semilogy((1: r), d3);
    legend('Explicit Euler', 'Taylor-2', 'Taylor-3');
    title(sprintf('p=%d', i));
end
```

Euler 显式格式即当 q=1 时的 Taylor 级数法。得到的误差比较如下：

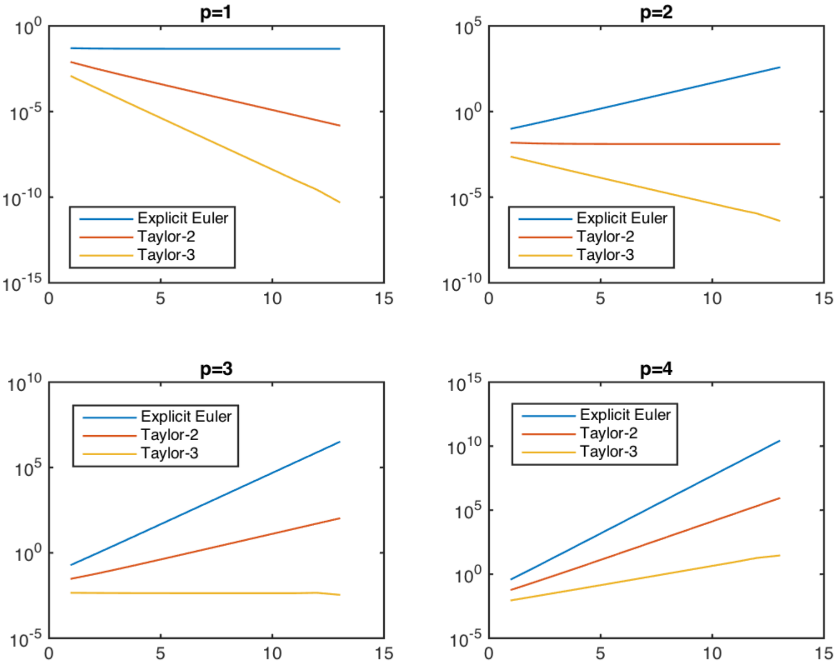




将 $\Delta t^1, \Delta t^2, \Delta t^3$ 画出可以由平行关系看出各方法的收敛阶数。或者通过分别枚举 p=1,2,3,4, 并将每种方法得到的结果代入

$$\frac{|u(t) - u_n|}{\Delta t^p},$$

可以得到最高的具有常数上界结果的 p 即其收敛阶数：



可以看出，显式欧拉格式、2 阶、3 阶 Taylor 级数法的收敛阶数分别为 1、2、3。

2 用一到四阶格式的 Runge-Kutta 方法计算 $\frac{\mathrm{d}u}{\mathrm{d}t} = u - u^2$，观察收敛阶（P79/3）

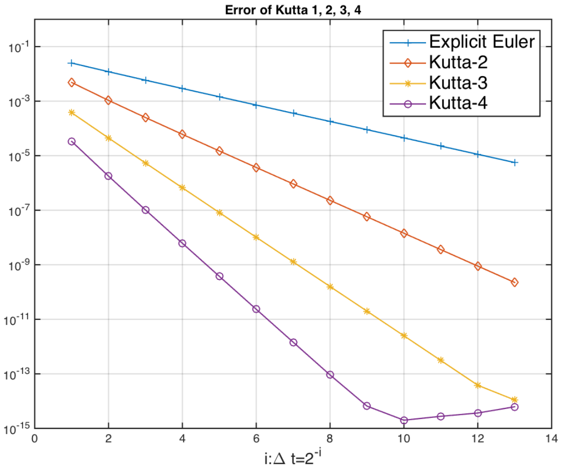
代码如下：

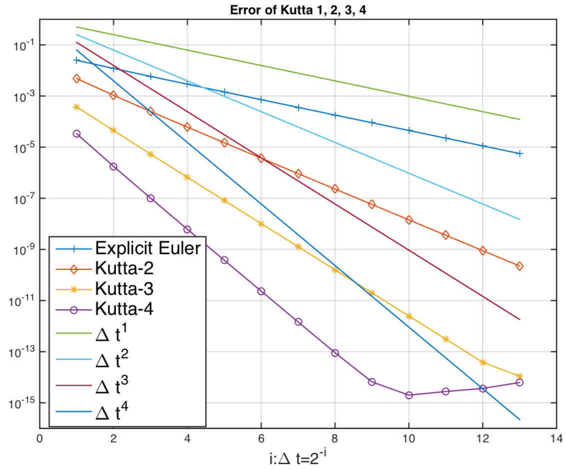
```
%% definitions
f = @(u) u - u .^ 2 ;
u = @(t, u0) 1 ./ ( ( 1 / u0  - 1 ) * exp( -t ) + 1 );
phi2 = @(u, dt) 1/2 *( f(u) + f(u + dt * f(u)) );
phi3 = @(u, dt) 1/6 *( f(u) + 4 * f(u + dt/2 * f(u)) ...
    + f(u + dt *(-f(u) + 2* f(u + dt/2 * f(u)))) );
phi4 = @(u, dt) 1/8 *( f(u) + 3 * f(u + dt/3 * f(u)) ...
    + 3 * f(u + dt *(-1/3 * f(u) + f(u + dt/3 * f(u)))) ...
    + f(u + dt *(f(u) - f(u + dt/3 * f(u)) ...
     + f(u + dt *(-1/3 * f(u) + f(u + dt/3 * f(u)))))) );
t0 = 0 ; T = 8 ;
u0 = 0.5 ; % u0 = 1.5

%% compute error of Explicit Euler, phi2, phi3, phi4
r = 13;
dts = 2 .^ -(1: r);
e1 = zeros(1, r);
e2 = zeros(1, r);
e3 = zeros(1, r);
e4 = zeros(1, r);
for i = 1 : r
```

```matlab
    dt = dts(i);
    tdt = t0 : dt : T ;
    n = length(tdt);
    uu = u( tdt , u0 );
    u1 = zeros ( 1 , n );
    u2 = zeros ( 1 , n );
    u3 = zeros ( 1 , n );
    u4 = zeros ( 1 , n );
    u1(1) = u0 ; u2(1) = u0 ; u3(1) = u0 ; u4(1) = u0 ;
    for j = 1 : n - 1
        u1(j + 1) = u1(j) + dt * f(u1(j));
        u2(j + 1) = u2(j) + dt * phi2(u2(j), dt);
        u3(j + 1) = u3(j) + dt * phi3(u3(j), dt);
        u4(j + 1) = u4(j) + dt * phi4(u4(j), dt);
    end
    e1(i) = max(abs(u1 - uu));
    e2(i) = max(abs(u2 - uu));
    e3(i) = max(abs(u3 - uu));
    e4(i) = max(abs(u4 - uu));
end

%% plot
semilogy((1: r), e1, '-+'); hold on; grid on;
semilogy((1: r), e2, '-d');
semilogy((1: r), e3, '-*');
semilogy((1: r), e4, '-o');
semilogy((1: r), dts);
semilogy((1: r), dts .^ 2);
semilogy((1: r), dts .^ 3);
semilogy((1: r), dts .^ 4);
set(gca,'ytick', 10 .^ (-15 : 2 : -1));
title('Error of Kutta 1, 2, 3, 4');
h = legend('Explicit Euler', 'Kutta-2', 'Kutta-3', 'Kutta-4', ...
    '\Delta t^1', '\Delta t^2', '\Delta t^3', '\Delta t^4');
xlabel('i:\Delta t=2^{-i}', 'Fontsize', 14);
set(h, 'Fontsize', 16);

%% verify degree of convergence
figure(2);
for i = 1 : 4
    d1 = e1 ./ (dts .^ i);
    d2 = e2 ./ (dts .^ i);
    d3 = e3 ./ (dts .^ i);
    d4 = e4 ./ (dts .^ i);
    subplot(2,2,i);
    semilogy((1: r), d1); hold on;
    semilogy((1: r), d2);
    semilogy((1: r), d3);
    semilogy((1: r), d4);
    legend('Explicit Euler', 'Kutta-2', 'Kutta-3', 'Kutta-4');
    title(sprintf('p=%d', i));
end
```

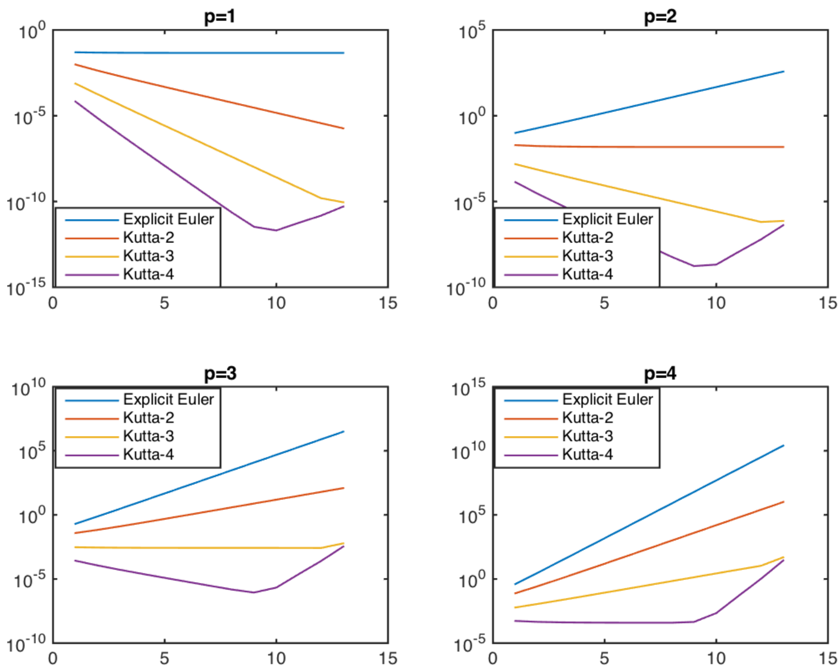
Euler 显式格式即当 q=1 时的 Runge-Kutta 方法。4 阶采用的是 3/8 的 Kutta 方法。

得到的误差比较如下图，可以看出 4 阶 Kutta 方法在步长为 2^{-10} 时误差就达到了 10^{-15} 的数量级，已经接近机器精度了，再减小步长已经不能提高精度了。





同样可通过两种方法考察其收敛阶数各为多少：



可以看出，显式欧拉格式、2 阶、3 阶、4 阶 Kutta 方法的收敛阶数分别为 1、2、3、4。