3-1 已知初值问题$\begin{cases} y' = \dfrac{2x}{3y^2} \\ y(0) = 1 \end{cases}$ 的精确解为 $y = \sqrt[3]{1+x^2}$，用差分方法研究这个问题在$[0,1]$上，$h = 0.1$时的数值解。可以首先验证其精确解如图：

```
>> for i=1:10
X(i)=f0(i/10);
end
>> X'

ans =

   1.003322283542089
   1.013159403820177
   1.029142466571507
   1.050717574498580
   1.077217345015942
   1.107931651350893
   1.142164759185383
   1.179273707994073
   1.218688907741976
   1.259921049894873
```

1. 欧拉方法

欧拉方法用差分形式$\dfrac{f(x+h)-f(x)}{h}$来近似代替 $f'(x)$，推导出递推方程 $y_{n+1} = y_n + hf(x_n, y_n)$进行计算，计算量小，但欧拉方法精度较低，仅有 1 阶代数精度，从结果的比较中可见精确度只有10^{-1}

```
>> f=@(x,y)2*x/(3*y^2);
>> Euler(f,0,1,10,1)

ans =

                   0   1.000000000000000
   0.100000000000000   1.000000000000000
   0.200000000000000   1.006666666666667
   0.300000000000000   1.019823984328173
   0.400000000000000   1.039053996210607
   0.500000000000000   1.063753742840065
   0.600000000000000   1.093211287375390
   0.700000000000000   1.126680984094971
   0.800000000000000   1.163443468397965
   0.900000000000000   1.202844551131971
   1.000000000000000   1.244314379696825
```

```
function E = Euler( f,a,b,N,ya )

h=(b-a)/N;
y=zeros(1,N+1);
x=zeros(1,N+1);
y0=zeros(1,N+1);
y(1)=ya;
x=a:h:b;
for i=1:N
    y(i+1)=y(i)+h*feval(f,x(i),y(i));
end
E=[x',y'];
```

2. 改进欧拉方法

改进欧拉方法运用了预报校正系统来修正误差，与欧拉方法相比精度显著提高：

$$\begin{cases} \bar{y}_{n+1} = y_n + hf(x_n, y_n) \\ y_{n+1} = y_n + \dfrac{h}{2}[f(x_n, y_n) + f(x_{n+1}, \bar{y}_{n+1})] \end{cases}$$

```
>> MendEuler(f,0,1,10,1)

ans =

                   0   1.000000000000000
   0.100000000000000   1.003333333333333
   0.200000000000000   1.013180434398852
   0.300000000000000   1.029171244550309
   0.400000000000000   1.050751079998023
   0.500000000000000   1.077252310612832
   0.600000000000000   1.107965053358377
   0.700000000000000   1.142194135689444
   0.800000000000000   1.179297284217600
   0.900000000000000   1.218705575564524
   1.000000000000000   1.259930265862033
```

```
function E = MendEuler( f,a,b,N,ya )

h=(b-a)/N;
y=zeros(1,N+1);
x=zeros(1,N+1);
y(1)=ya;
x=a:h:b;
for i=1:N
    y1=y(i)+h*feval(f,x(i),y(i));
    y2=y(i)+h*feval(f,x(i+1),y1);
    y(i+1)=(y1+y2)/2;
end
E=[x',y'];

end
```

3. 四阶龙格-库塔方法

四阶龙格-库塔方法将步长区间$[x_n, x_{n+1}]$四等分并计算四个子区间上函数斜率的加权平均，其中为计算子区间斜率设计了预报校正系统，从比较中可以看到精确度达到了10^{-6}，在精确度与计算量间取得可以满意的平衡：

```
>> Rungkuta4(f,0,1,10,1)

ans =

                   0   1.000000000000000
   0.100000000000000   1.003322292719565
   0.200000000000000   1.013159438200695
   0.300000000000000   1.029142535439115
   0.400000000000000   1.050717679021905
   0.500000000000000   1.077217479999272
   0.600000000000000   1.107931808368870
   0.700000000000000   1.142164929384162
   0.800000000000000   1.179273883780467
   0.900000000000000   1.218689083410464
   1.000000000000000   1.259921221582087

fx >>
```

```
function R = Rungkuta4( f,a,b,N,ya )

h=(b-a)/N;
y=zeros(1,N+1);
x=zeros(1,N+1);
y(1)=ya;
x=a:h:b;
for i=1:N
    k1=feval(f,x(i),y(i));
    k2=feval(f,x(i)+h/2,y(i)+k1*h/2);
    k3=feval(f,x(i)+h/2,y(i)+k2*h/2);
    k4=feval(f,x(i)+h,y(i)+k3*h);
    y(i+1)=y(i)+(k1+2*k2+2*k3+k4)*h/6;
end
R=[x',y'];

end
```

3-2 四阶亚当姆斯方法的基本思路依然是求子区间斜率的加权和，但通过利用之前已经完成计算的点进行迭代，能有效降低计算量，同时从结果来看精确度较高，能达到10^{-5}；改进的四阶亚当姆斯方法设计了预估校正，计算结果精确度更高，达到10^{-7}

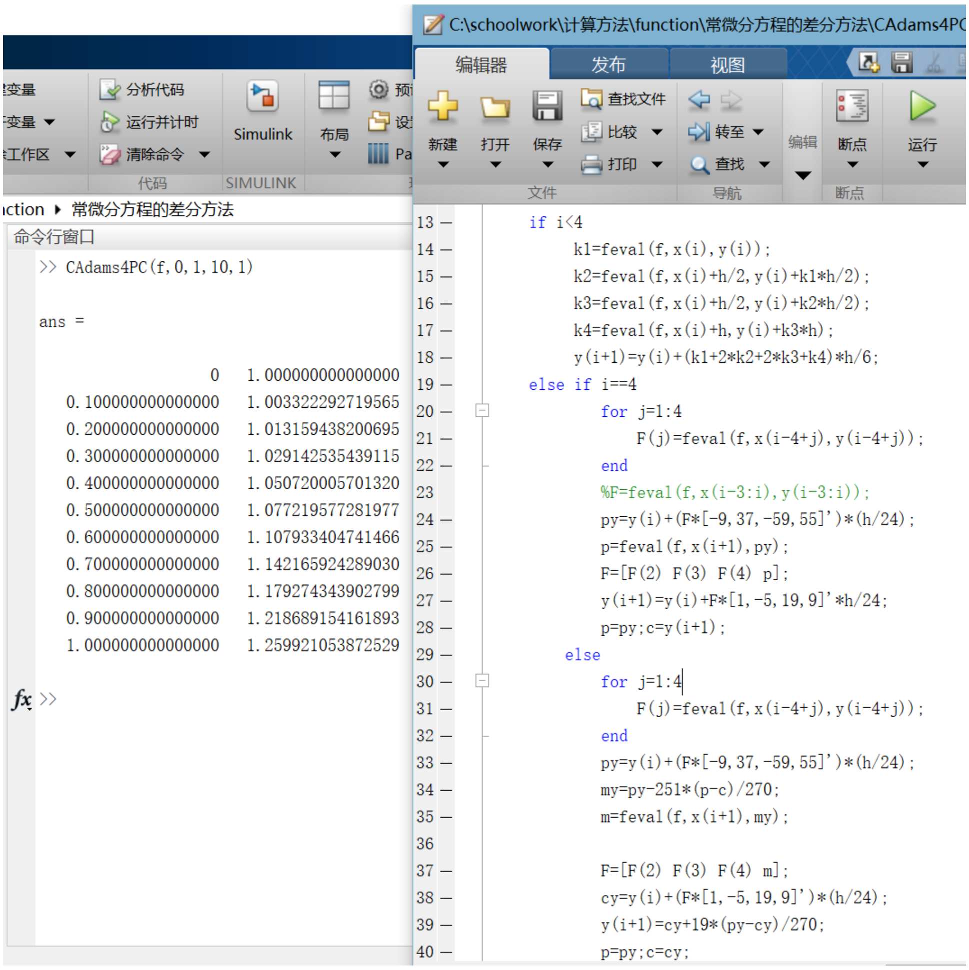
```
命令行窗口
>> Adams4PC(f,0,1,10,1)

ans =

                   0   1.000000000000000
   0.100000000000000   1.003322292719565
   0.200000000000000   1.013159438200695
   0.300000000000000   1.029142535439115
   0.400000000000000   1.050720005701320
   0.500000000000000   1.077222006226289
   0.600000000000000   1.107937969725546
   0.700000000000000   1.142171994240437
   0.800000000000000   1.179281183806477
   0.900000000000000   1.218696130520936
   1.000000000000000   1.259927725124502

fx >>
```

```
if N<4
    return;
end
h=(b-a)/N;
x=zeros(1,N+1);
y=zeros(1,N+1);
x=a:h:b;
y(1)=ya;
F=zeros(1,4);
for i=1:N
    if i<4
        k1=feval(f,x(i),y(i));
        k2=feval(f,x(i)+h/2,y(i)+k1*h/2);
        k3=feval(f,x(i)+h/2,y(i)+k2*h/2);
        k4=feval(f,x(i)+h,y(i)+k3*h);
        y(i+1)=y(i)+(k1+2*k2+2*k3+k4)*h/6;
    else
        for j=1:4
            F(j)=feval(f,x(i-4+j),y(i-4+j))
        end
        py=y(i)+F*[-9,37,-59,55]'*h/24;
        p=feval(f,x(i+1),py);
        F=[F(2) F(3) F(4) p];
        y(i+1)=y(i)+F*[1,-5,19,9]'*h/24;
    end
end
```



实验体会：差分方法的最基本手段是用 $\dfrac{f(x+h)-f(x)}{h}$（或其极限意义上的等价形式）近似代替 $f'(x)$，欧拉方法基于这条思路设计，但光凭此精度较低不能满意；龙格-库塔方法通过四等分子区间，分别求斜率并加权求和得到精度较高的解；亚当姆斯方法同样求加权求和解，但由于其复用之前的计算结果，能有效减少计算量。值得注意的是预报校正系统的运用，这几乎适用于任何差分方法，而同时大大提高精度，在设计其他算法时值得考虑加入预报校正系统。