第五章作业报告

问题描述：火箭发射过程的速度可由如下公式计算：

$$v = u\,ln\left(\frac{m_0}{m_0 - qt}\right) - qt$$

其中，v 是向上的速度，u 是燃料相对于火箭喷出的速度，m_0是火箭在 t=0 时的初始质量，q 是燃料消耗速度，g 是重力加速度。假设 u=1800m/s，m_0=160000kg，q-2500kg/s，g=9.8$m\,/\,s^2$，请：

（1）采用不同的数值积分方法计算火箭载 30s 时能上升多高，并分析误差。

（2）利用数值微分方法画出火箭加速度与时间的关系图。

问题分析：

（1）数值积分方法有 Newton-Cotes 方法，其中有梯形公式法，辛普森公式法，复合梯形公式法，复合辛普森公式法。还有龙贝格积分法和高斯积分法。

（2）数值微分方法是通过有限差商来逼近导数，有着一阶微分两点公式和一阶微分三点公式。除此以外，还可通过 Richardson 外推法来计算导数。

Matlab 程序组织结构：

在主程序 main.m 里面可以运行各函数程序，main.m 里面也有着题干表达式的参数值

函数：

数值积分函数：

Trapezium.m	梯形公式
Simpson_3.m	辛普森$\frac{1}{3}$公式
Simpson_8.m	辛普森$\frac{3}{8}$公式
Compound_T.m	复合梯形公式
Compound_S.m	复合辛普森$\frac{1}{3}$公式
Richardson.m	积分形式的 Richardson 外推法公式
Rombreg.m	龙贝格积分
Guass——Legndre.m	高斯积分

数值微分函数：

df.m	两点向前差商公式
db.m	两点向后差商公式
dm.m	两点中心差商公式和三点中心差商公式
df3.m	三点向前差商公式
db3.m	三点向后差商公式
d_Richardson.m	微分形式的 Richardson 外推法公式

数值积分 Matlab 程序：

通过 main.m 里的函数表达式和 matlab 计算积分的公式可以计算得到从 0 到 30 的真实积分值为 10879.619。

代码如下：

```matlab
%%
%参数
u = 1800;
m0 = 160000;
q = 2500;
g = 9.8;

syms x;

%所求函数
v = @(t)(u*log(m0./(m0-q*t)) - g*t);
output_T = vpa(integral(v,0,30),8)
y = v(x);
```

```
output_T =

10879.619
```

一、梯形公式

```matlab
function [output] = Trapezium(a,b,f)%传入起始点，终点，函数表达式
    %梯形公式
    output = (b-a)*(f(b)+f(a))/2;
end
```

对应 main.m 代码如下为利用梯形公式进行积分

```matlab
%%
%梯形公式
output = vpa(Trapezium(0,30,v),8)
et = vpa(abs(output_T - output),4)
g = matlabFunction(diff(y,2));
ea = vpa(abs(integral(g,0,30)/30*(30-0)^3 *(-1/12)),4)
```

```
output =

12668.109


et =

1788.0


ea =

1861.0
```

结果如右图所示，可以看到，梯形公式计算值为 12668.109，真实误差 Et = 1788.0，

估计误差 Ea = $-\frac{(b-a)^3}{12}f''(\xi)$，其中$f''(\xi)=\frac{\int_a^b v^{(2)}dt}{b-a}$，计算的 Ea = 1861.0

二、辛普森公式

1.Simpson$\frac{1}{3}$法则需要起点，中点和终点的函数值：

$$I(f)=\frac{h}{3}\left[f(a)+4f\left(\frac{b+a}{2}\right)+f(b)\right]$$

函数代码如下，Simpson_3.m：

```matlab
function [output] = Simpson_3(a,b,f)%传入参数为起点，终点，函数表达式
    %Simpson1/3公式
    output = (b-a)/6*(f(a)+4*f((a+b)/2)+f(b));
end
```

对应 main.m 代码的代码节如下:

```
%%
%辛普森三分之一公式
output = vpa(Simpson_3(0,30,v),8)
et = vpa(abs(output_T - output),4)
g = matlabFunction(diff(y,4));
ea = vpa(abs(integral(g,0,30)/30*(30-0)^5 *(-1/90)),4)
```

```
output =

10896.963


et =

17.34


ea =

21.9
```

结果如右图所示，可以看到，辛普森三分之一公式计算值为 10896.963，真实误差 Et = 17.34，估计误差 Ea = $-\frac{h^5}{90}f^{(4)}(\xi)$ ，其中$f^{(4)}(\xi) = \frac{\int_a^b v^{(4)}\,dt}{b-a}$，计算的 Ea = 21.9

2.辛普森$\frac{3}{8}$法则需要将（a,b）的距离分为三段，即需要 4 个等距点

$$I(f) = (b-a)\frac{[f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)]}{8}$$

Simpson_8.m 函数代码如下:

```
function [output] = Simpson_8(x0,x1,x2,x3,f)
    %Simpson3/8公式
    output = (x3-x0)/8 * (f(x0)+3*f(x1)+3*f(x2)+f(x3));
end
```

main.m 对应代码节如下:

```
%%
%辛普森八分之三公式
output = vpa(Simpson_8(0,10,20,30,v),8)
et = vpa(abs(output_T - output),4)
g = matlabFunction(diff(y,4));
ea = vpa(abs(integral(g,0,30)/30*(30-0)^5/6480),4)
```

```
output =

10887.525


et =

7.906


ea =

9.733
```

运行结果如右图所示，可以看到，辛普森八分之三公式计算值为 10887.525，真实误差 Et = 7.906，估计误差 Ea = $-\frac{(b-a)^5}{6480}f^{(4)}(\xi)$ ，其中$f^{(4)}(\xi) = \frac{\int_a^b v^{(4)}\,dt}{b-a}$，计算的 Ea = 9.733

从上述结果可以看出，辛普森公式比起梯形公式，有着更好的精确度。
而 Simpson 八分之三原则比 Simpson 三分之一原则更加精确，且两式都有着相同的代数精度——4。但是 Simpson 八分之三原则比 Simpson 三分之一计算量更大，一般情况下，优先使用 Simpson 三分之一原则计算。

梯形公式和辛普森公式都属于 Newton-Cotes 公式，n=1 为梯形公式，n=2 为 Simpson 三分之一公式，n=3 为 Simpson 八分之三公式，n=4 时为 Cotes 公式，初始数据的误差在求

和过程中会扩大，将导致计算的不稳定。因此，高阶 Newton-Cotes 公式不能保证等距数值积分的收敛性。因此，一般不采用高阶（n>7）的 Newton-Cotes 求积公式。

三、复合 Newton-Cotes 公式

先将积分区间分成几个小区间，并在每个小区间上用低阶 Newton-Cotes 公式计算积分近似值，然后对这些近似值求和，从而得到所求积分的近似值。

1.复合梯形公式

$$I = (b-a)\frac{[f(a)+2\sum_{i=1}^{n-1} f(x_i)+f(b)]}{2n}$$

函数代码 Compound_T.m 如下：

```
function [output] = Compound_T(a,b,f,n)
    %复合梯形公式，传入参数为区间始终点a和b，函数f，被分成的区间数n
    output = f(a) + f(b);
    h = (b-a)/n;
    for i=1:n-1
        output = output + 2*f(h*i+a);
    end
    output = output*(b-a)/2/n;
end
```

将 n=1，2，3，4，5，6，7，8，9，10 代入进行对比，代码和结果如下：

```
%%
%复合梯形公式
output = [];
ea =[];
et =[];
for i = 1:10
    output(end+1) = vpa(Compound_T(0,30,v,i),8);
    et(end+1) = vpa(abs(output_T - output(end)),4);
    g = matlabFunction(diff(y,2));
    ea(end+1) = vpa(abs(integral(g,0,30)/30*(30-0)^3/12/i/i),4);
end
patients = table((1:10)',output',et',ea','VariableNames',{'n','result','Ea','Et'})
```

```
patients =

  10×4 table
```

n	result	Ea	Et
1	12668	1788.5	1861.2
2	11340	460.13	465.3
3	11085	205.75	206.8
4	10996	115.99	116.33
5	10954	74.31	74.449
6	10931	51.633	51.7
7	10918	37.948	37.984
8	10909	29.06	29.081
9	10903	22.965	22.978
10	10898	18.603	18.612

从结果中可以看出，子区间数目增加时，误差随之减小，误差与 n^2成反比。

且子区间数目的增加，也让估计误差和真是误差越来越接近，所得结果越来越接近真值。

2.复合Simpson公式,n为偶数个子区间

$$S(f)=\frac{h}{3}\left(f(a)+4\sum_{i=0}^{m-1}f(x_{2i+1})+2\sum_{i=1}^{m-1}f(x_{2i})+f(b)\right)$$

代码Compound_S.m如下：

```
function [output] = Compound_S(a,b,f,n)
    %复合Simpson公式,n为偶数个子区间
    h = (b-a)/n;
    output = f(a) + f(b);
    m = n/2;
    for i=0:m-1
        if i==0
            output = output + f(a+(2*i+1)*h)*4;
        else
            output = output + f(a+(2*i+1)*h)*4 + f(a+(2*i)*h)*2;
        end
    end
    output = output*h/3;
end
```

在 main.m 里面的使用的代码：

```
%%
%复合Simpson公式，选择n=4与复合梯形公式进行对比
output = vpa(Compound_S(0,30,v,4),8)
et = vpa(abs(output_T - output),4)
g = matlabFunction(diff(y,4));
ea = vpa(abs(integral(g,0,30)/30*(30-0)^5/180/(4^4)),4)
```

```
output =

10880.894

et =

1.275

ea =

1.369
```

结果如右图，可知结果为 10880.894，真实误差为 1.275，估计误差通过公式

Ea = $-\frac{(b-a)^5}{180n^4}f^{(4)}(\xi)$可得为 1.369。

与梯形公式 n=8 的情况进行对比，两种方法都用到 9 个点的函数值，计算量基本相同，但复合 Simpson 公式得到的近似值比复合梯形公式得到的近似值精确。

四、龙贝格积分

基于外推算法：用若干个积分近似值来推算更精确的新的近似值的方法

$$I(f)=T_{2n}(f)+\frac{1}{3}\left(T_{2n}(f)-T_n(f)\right)$$

比 T（2n）更好的接近积分真值。

验证可得：

$$s_n(f)=\frac{4}{3}T_{2n}(f)-\frac{1}{3}T_n(f)$$

$$C_n(f)=\frac{16}{15}S_{2n}(f)-\frac{1}{15}S_n(f)$$

$$R_n(f)=\frac{64}{63}C_{2n}(f)-\frac{1}{63}C_n(f)$$

可得 Richardson 外推法公式：

$I - G_{2n} = \frac{1}{4^m-1}(G_{2n} - G_n)$， m =1， 2， 3

当 m>4 时， 上式已趋近于 0， 再进行外推已无必要。故在实际中只做到 R 为止。

Romberg.m 函数代码如下：

```
function [output] = Romberg(a,b,f,e)%传入参数分别为起始点，终点，函数表达式，误差上界
    T = [];
    S = [];
    C = [];
    R = [];
    e0 = 1;
    T(end+1) = (b-a)*(f(a)+f(b))/2;
    T1 = [];%T1用来储存用于进行比较计算误差的值
    T1(end+1) = T(1);
    i=1;
    while e0 > e           %迭代，直到达到误差界未知
        T(end+1) = Compound_T(a,b,f,2^i);%复合梯形公式
        S(end+1) = Richardson(T(end-1),T(end),1);%Richarson外推公式计算S
        if i == 1
            T1(end+1) = S(1);
        elseif i == 2
            C(end+1) = Richardson(S(end-1),S(end),2);%Richarson外推公式计算C
            T1(end+1) = C(1);
        elseif i >= 3
            C(end+1) = Richardson(S(end-1),S(end),2);%Richarson外推公式计算R
            R(end+1) = Richardson(C(end-1),C(end),3);
            T1(end+1) = R(end);
        end
        i = i + 1;
        e0 = abs(T1(end)-T1(end-1));
    end
    output = T1(end);
    T
    S
    C
    R
end
```

其中 Richardson 函数代码如下：

```
function [output] = Richardson(T1,T2,m)
%Richardson外推法
    output = ((4^m)*T2 - T1)/(4^m-1);
end
```

在 main.m 中调用此函数

```
%%
%龙贝格积分
output = vpa(Romberg(0,30,v,2e-5),8)
et = vpa(abs(output_T - output),4)
```

结果如下：

```
output =

10879.619
```

```
et =

4.519e-8
```

可以看到，龙贝格积分得到的积分结果精确度十分高，误差也很小。它是在梯形公式、 辛普森公式 和柯特斯公式之间关系的基础上，构造出一种加速计算积分的方法。 作为一种外推算法，在不增加计算量的前提下提高了误差的精度。

五、Guass_Legendre 积分公式

高斯积分公式：对节点不加限制，可以适当选取 x0，x1，使得积分公式具有更高的代数精度。$\int_a^b f(x)\,dx = \sum_{k=0}^{n} A_k f(x_k)$，若代数精度达到 2n+1，则称高斯求积公式，响应的求积节点成为高斯点。

构造方法：1.待定系数法，xk，Ak，令 f(x)=1,x²，…，x^{2n+1}使得求积公式精确成立

2.利用[a,b]的 n+1 次正交多项式确立高斯点，再利用高斯点确定系数

将[a,b]变为[-1,1]的方法：

将 x 变为 xd，可以得到$x = \frac{(a+b)+(b-a)x_d}{2}$ dx = $\frac{(b-a)dx_d}{2}$

常见的 Guass-Legendre 求积公式节点与系数表如下：

n	x_k	A_k	n	x_k	A_k
1	0	2	6	±0.9324695142 ±0.6612093865 ±0.2386191861	0.1713244924 0.3607615730 0.4679139346
2	±0.5773502692	1			
3	±0.7745966692 0	0.5555555556 0.8888888889	7	±0.9491079123 ±0.7415311856 ±0.4058451514 0	0.1294849662 0.2797053915 0.3818300505 0.4179591837
4	±0.8611363116 ±0.3399810436	0.3478548451 0.6521451549			
5	±0.9061798459 ±0.5384693101 0	0.2369268851 0.4786286705 0.5688888889	8	±0.9602898565 ±0.7966664774 ±0.5255324099 ±0.1834346425	0.1012285363 0.2223810345 0.3137066459 0.3626837834

则将这些系数和节点放入程序中来计算，程序如下：

```matlab
function [output] = Guass_Legendre(a,b,f,n)%传入参数分别为起始点，终点，函数表达式，结点个数
    %Guass_Legendre方法求解数值积分
    g = @(x)(f((a+b)/2 + (b-a)/2*x)*(b-a)/2)
    switch n
        case 1
            output = 2*g(0);
        case  2
            output = 1*g(0.5773502692)+1*g(-0.5773502692);
        case  3
            output = 0.5555555556*g(0.7745966692) + 0.5555555556*g(-0.7745966692) + 0.8888888889*g(0);
        case  4
            output = 0.3478548451*g(0.8611363116) + 0.3478548451*g(-0.8611363116) + 0.6521451549*g(0.3399810436) + 0.6521451549*g(-0.3399810436);
        case  5
            output = 0.2369268851*g(0.9061798459) + 0.2369268851*g(-0.9061798459) + 0.4786286705*g(0.5384693101) + 0.4786286705*g(-0.5384693101) + 0.5688888889*g(0);
        case  6
            output = 0.1713244924*g(0.9324695142) + 0.1713244924*g(-0.9324695142) + 0.3607615730*g(0.6612093865) + 0.3607615730*g(-0.6612093865) + 0.4679139346*g(0.2386191861) + 0.4679139346*g(-0.238619
    end
end
```

将 n=1，2，3，4，5，6 代入计算

main.m 程序对应部分和结果如下：

```matlab
%%
%高斯积分法
output = [];
et = [];
for i = 1:6
    output(end+1) = vpa(Guass_Legendre(0,30,v,i),8);
    et(end+1) = vpa(abs(output_T - output(end)),4);
end
patients = table((1:6)',output',et','VariableNames',{'n','result','Ea'})
```

```
patients =

  6×3 table

    n    result        Ea
    _    ______    __________

    1    10011         868.23
    2    10868         11.377
    3    10879        0.19168
    4    10880      0.0035895
    5    10880     7.0763e-05
    6    10880     1.4246e-06
```

可以看到结果精确，误差很小。

高斯积分法的计算量小，精度高。但是 n 改变大小时，节点和系数几乎都需要变化，利用余项控制精度较为困难，需要计算节点处的函数值，不适用于函数表达式未知的形式。

数值微分 Matlab 程序：

一、差商近似：

向前差商：　$f'(x_0) \approx \frac{f(x_0+h)-f(x_0)}{h}$

向后差商：　$f'(x_0) \approx \frac{f(x_0)-f(x_0-h)}{h}$

中间差商：　$f'(x_0) \approx \frac{f(x_0+h)-f(x_{0-h})}{2h}$

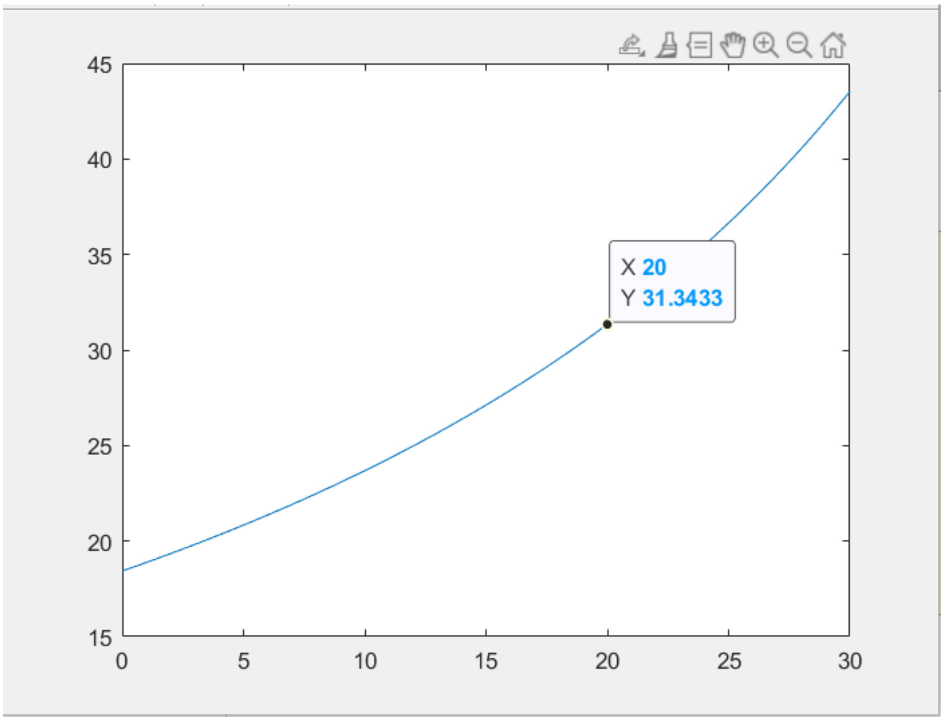
h 越小，误差越小，但同时舍入误差增大。

```
function [output] = df(x,h,f)%参数为被导的x，步长h，函数f
    output = (f(x+h)-f(x))/h;
end
```

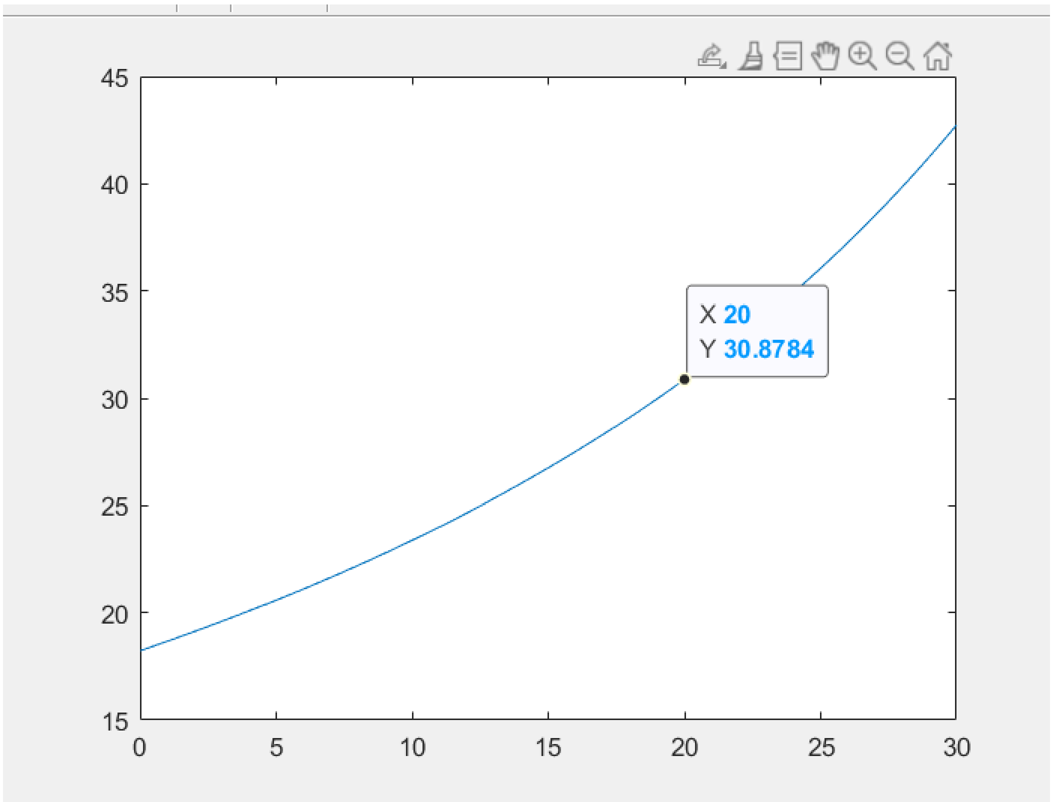
```
function [output] = db(x,h,f)%参数为被导的x，步长h，函数f
    output = (f(x)-f(x-h))/h;
end
```

```
function [output] = dm(x,h,f)%参数为被导的x，步长h，函数f
    output = (f(x+h)-f(x-h))/h/2;
end
```

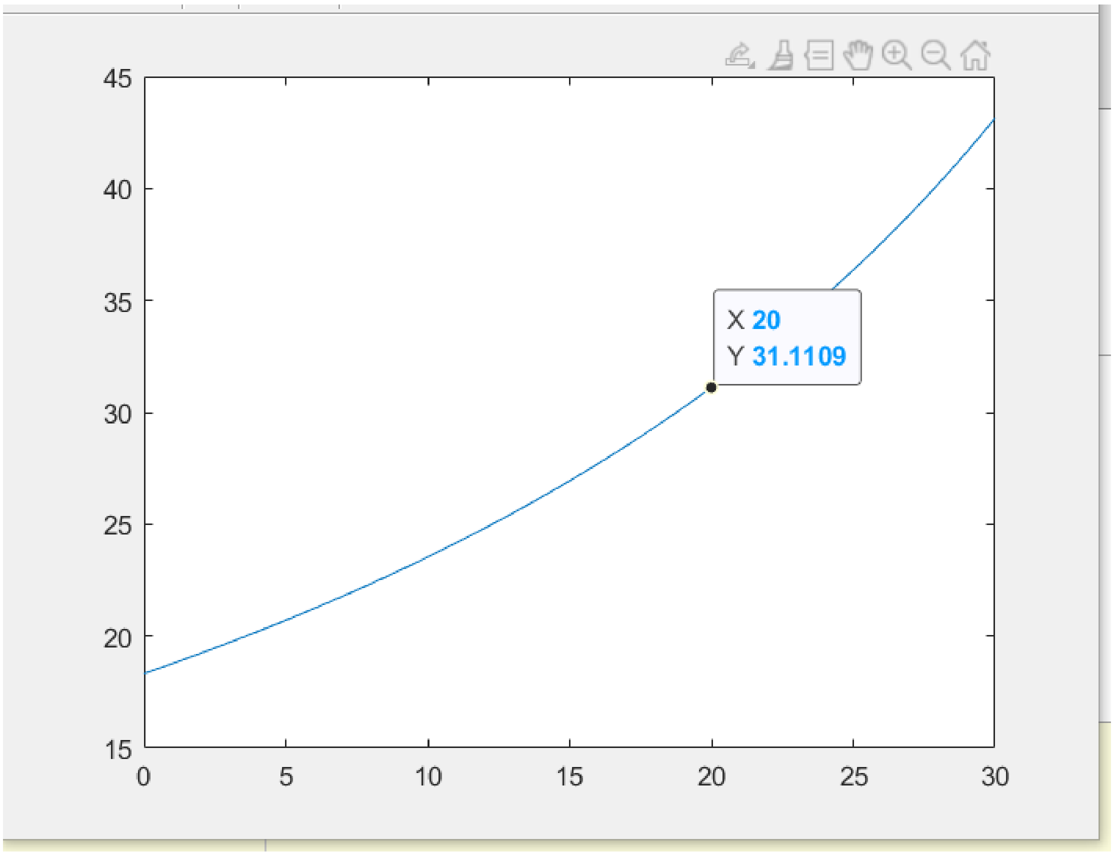
利用三种差商公式在（0，30）上对 v 进行一阶微分运算，选取 h 为 0.2，可以得到加速度与时间的关系图像分别如下三张所示：



前向差商


45
40
35
30
25
20
15
0
5
10
15
20
25
30
X 20
Y 30.8784

向后差商


45
40
35
30
25
20
15
0
5
10
15
20
25
30
X 20
Y 31.1109

中间差商

二、一阶微分三点公式

中间差商公式和上面方法相同。

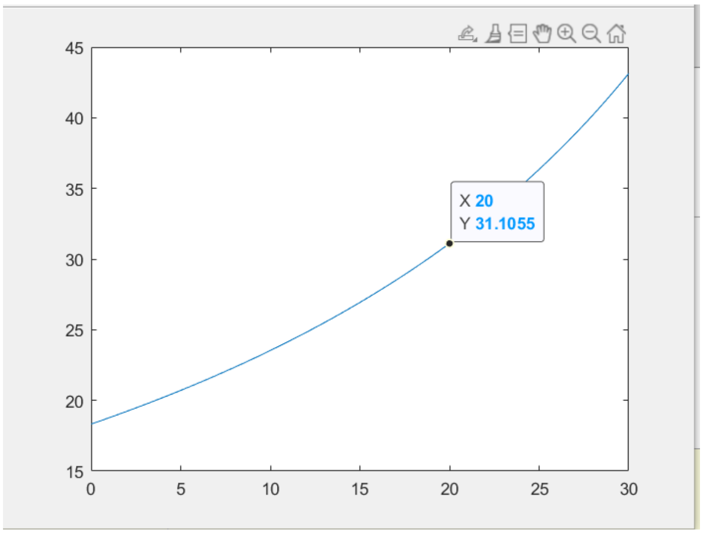
三点向前差商公式：$f'_{(x_0)} = \frac{1}{2h}[-3f(x_0) + 4f(x_1) - f(x_2)]$

三点向后差商公式：$f'_{(x_2)} = \frac{1}{2h}[f(x_0) - 4f(x_1) + 3f(x_2)]$

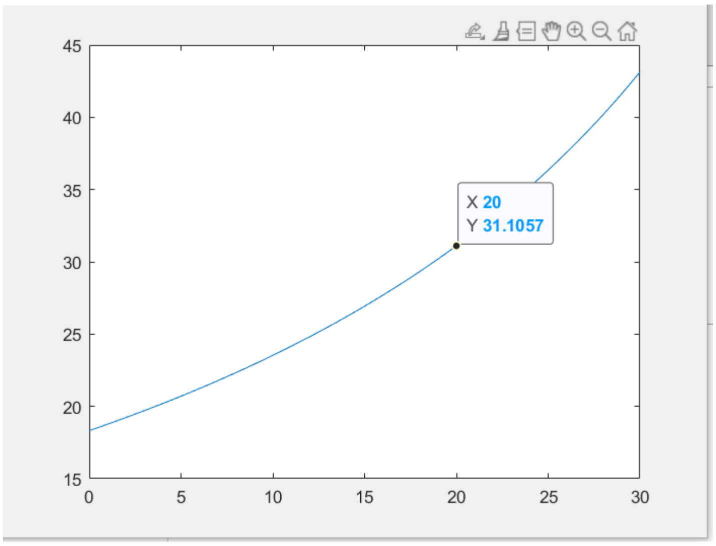
```
function [output] = df3(x, h, f)
    %三点向前差商公式
    output = (-3*f(x) + 4*f(x+h) - f(x+2*h))/2/h;
end
```

```
function [output] = db3(x, h, f)
    %三点向后差商公式
    output = (f(x-2*h) - 4*f(x-h) + 3*f(x))/2/h;
end
```

利用三点差商公式在（0，30）上对 v 进行一阶微分运算，选取 h 为 0.2，可以得到加速度与时间的关系图像分别如下所示：



三点向前差商公式



三点向后差商公式

三、Richardson 外推法

类似数值积分的 Richardson 外推法，可以写出数值积分的外推公式：

$$T(h) = \frac{1}{2h}[f(x+h) - f(x-h)]$$

$$S(h) = \frac{4}{3}T(h) - \frac{1}{3}T(2h)$$

$$C(h) = \frac{16}{15}S(h) - \frac{1}{15}S(2h)$$

计算 f'(x)的近似值，直到$|C_{k+1} - C_k| < \varepsilon$，此时 f'(x)≈$C_{k+1}$

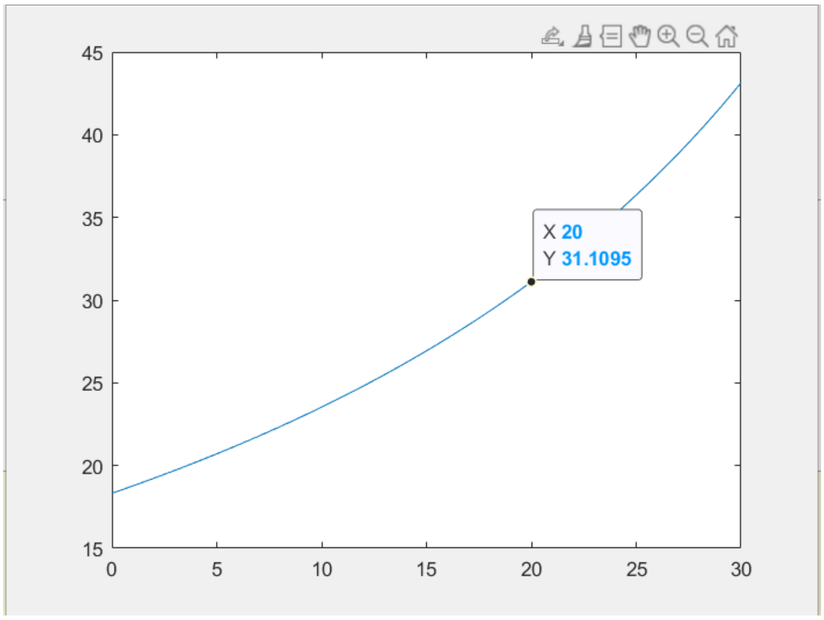
Matlab 程序：d_Richardson.m

```
function [output] = d_Richardson(x,h,f,e)
    %D_RICHARDSON 外推法求解微分
    T = [];
    S = [];
    C = [];
    es = 1;

    R = [];%记录值
    T(1) = dm(x,h,f);
    R(end+1) = T(1);
    i=1;
    while es > e
        T(end+1) = dm(x,h/2,f);
        S(end+1) = 4/3*T(end) - 1/3*T(end-1);
        if i==1
            R(end+1) = S(1);
        else
            C(end+1) = 16/15*S(end) - 1/15*S(end-1);
            R(end+1) = C(end);
        end
        i = i + 1;
        es = abs(R(end) - R(end-1));
    end

    output = R(end);
end
```

运行此方法绘制加速度与时间图像如下:



Richardson 外推法

将以上方法在 t=20 所得的微分值进行汇总并与真实值比较如下:

方法	前向差商	后向差商	中间差商	三点前向差商	三点后向差商	Richardson 外推法	真实值
结果	31.3433	30.8784	31.1109	31.1055	31.1057	31.1095	31.1091
误差	0.2342	0.2307	0.0018	0.0036	0.0034	0.0004	

比较可以发现，中间差商公式的精度较高，三点公式结果比两点公式结果更准确。Richardson 在一定程度上可以获得更为接近的真实值。