Problem 1. Simpson's rule, 15pts

Part (a)

```
% Function
f = @(x) 1./log(x);

% Given
a = 100;
b = 1000;
n = 2000;

% calculate each slice
h = (b-a)/n;
x = a:h:b;
% use function
fx = f(x)
```

```
fx = 1×2001
    0.2171    0.2169    0.2167    0.2165    0.2163    0.2161    0.2159    0.2157 ···
```

```
%approximation
approximation = (x/3)*(fx(n) + fx(n+1) + 4*h/3*sum(fx(2:2:n)) + 2*h/
3*sum(fx(3:2:n-1)))
```

```
approximation = 1×2001
10^4 ×
    0.4924    0.4946    0.4968    0.4990    0.5013    0.5035    0.5057    0.5079 ···
```

```


% Gauss estimate
prime = sum(isprime(a:b))
```

```
prime = 143
```

Part (b)

```
% Integral function
prime2 = integral(@(x) 1./log(x), a, b)
```

```
prime2 = 147.4835
```

Problem 2. Degree of precision, 10 pts

Trapezoidal rule

```
%integration interval
a = 0; b = 1
```

```
b = 1
```

```
interval = [a, b];
%subintervals
n = 2;
```

```
for k = 0:1000
    f = @(x) x.^k;
    m = mytrap(f,interval,n)
    exact_m = 1/(k+1);
    eror = abs(m - exact_m);
    if eror > 1e-12
        final_value = k-1
        break % break when error exceeds 1e-12
    end
end
```

```
m = 1
m = 0.5000
m = 0.3750
final_value = 1
```

```
display(m)
```

```
m = 0.3750
```

```
display(exact_m)
```

```
exact_m = 0.3333
```

```
display(eror)
```

```
eror = 0.0417
```

Simpson's rule

```
%integration interval
a = 0; b = 1;
interval = [a, b];
%subintervals
n = 2;

for k = 0:1000
    f = @(x) x.^k;
    m = mysimp(f,interval,n)
    exact_m = 1/(k+1);
    eror = abs(m - exact_m);
    if eror > 1e-12
        final_value = k-1
        break % break when error exceeds 1e-12
    end
end
```

```
m = 1
m = 0.5000
m = 0.3333
m = 0.2500
m = 0.2083
final_value = 3
```

```
display(m)
```

```
m = 0.2083
```

```
display(exact_m)
```

```
exact_m = 0.2000
```

```
display(eror)
```

```
eror = 0.0083
```

Problem 3. Gaussian Quadrature, 20 pts

```
clear
xi = [-0.96816023950763 -0.83603110732664 -0.61337143270059
-0.32425342340381 0 0.32425342340381 0.61337143270059 0.83603110732664
0.96816023950763]
```

```
xi = 1×9
   -0.9682   -0.8360   -0.6134   -0.3243         0    0.3243    0.6134    0.8360 ···
```

```
ci = [0.081274388361574 0.18064816069486 0.26061069640294 0.31234707704
0.33023935500126 0.31234707704 0.26061069640294 0.18064816069486
0.081274388361574]
```

```
ci = 1×9
    0.0813    0.1806    0.2606    0.3123    0.3302    0.3123    0.2606    0.1806 ···
```

```
f1 = @(x) (x-1).^2.*exp(-x.^2)
```

```
f1 = function_handle with value:
    @(x)(x-1).^2.*exp(-x.^2)
```

```
f2 = @(x) (1./(1+x.^2))
```

```
f2 = function_handle with value:
    @(x)(1./(1+x.^2))
```

```
exact_f1 = integral(f1,0,1);
exact_f2 = integral(f2,0,1);

approx_f1=0;
approx_f2=0;

n = 9;

for i = 1:n
    approx_f1 = approx_f1 + ci(i) * f1(xi(i));
end
```

```
for i = 1:n
    approx_f2 = approx_f2 + ci(i) * f2(xi(i));
end

% error
error_f1 = abs(approx_f1 - exact_f1)
```

```
error_f1 = 1.5684
```

```
error_f2 = abs(approx_f2 - exact_f2)
```

```
error_f2 = 0.7854
```

Simpson's rule

```
clear;
%integration interval
a = 0; b = 1;
interval = [a, b];

%subintervals
ns = 8;
hs = (b-a)/ns;

for k = 0:1000
    f1 = @(x) (x-1).^2.*exp(-x.^2)
    ms = mysimp(f1,interval,ns);
    exact_ms = 1/(k+1);
    errors = abs(ms - exact_ms);
    if errors > 1e-12
        final_value = k-1
        break % break when error exceeds 1e-12
    end
end
```

```
f1 = function_handle with value:
    @(x)(x-1).^2.*exp(-x.^2)
final_value = -1
```

```
display(ms)
```

```
ms = 0.3042
```

```
display(exact_ms)
```

```
exact_ms = 1
```

```
display(errors)
```

```
errors = 0.6958
```

Trapezoidal rule

```
clear;
%integration interval
a = 0; b = 1;
interval = [a, b];
%subintervals
n = 2;

for k = 0:1000
    f1 = @(x) (x-1).^2.*exp(-x.^2)
    m = mytrap(f1,interval,n)
    exact_m = 1/(k+1);
    eror = abs(m - exact_m);
    if eror > 1e-12
        final_value = k-1
        break % break when error exceeds 1e-12
    end
end
```

```
f1 = function_handle with value:
    @(x)(x-1).^2.*exp(-x.^2)
m = 0.3474
final_value = -1
```

```


display(m)
```

```
m = 0.3474
```

```
display(exact_m)
```

```
exact_m = 1
```

```
display(eror)
```

```
eror = 0.6526
```

Problem 4. Numerical Differentiation, 15 pts

Part (a)

```
clear;
load('free_fall.mat');
time = t;
height = y;

h = time(2) - time(1);
velocity = zeros(size(height));

% calculate equation 1
velocity(1) = (1/h)*(-0.5*height(3) + 2*height(2) - 1.5*height(1));
```

```
% calculate equation 2
for j = 2:(length(height)-1)
    velocity(j) = (1/(2*h))*(height(j+1) - height(j-1));
end

% calculate equation 3
velocity(end) = (1/h)*(1.5*height(end) - 2*height(end-1) +
0.5*height(end-2));

% plot
plot(time,velocity,'c.-');
xlabel('time (s)');
ylabel('velocity (m/s)');
title('Velocity of delivery package');
grid on;
```

Part (b)

```
display('The approximate time that each parachute deploys is demonstrated on
the velocity curve. The first change happens at 5 seconds where we can see
a sharp change in the curve, and the second parachute deploys at around 10
seconds.')
```

```
The approximate time that each parachute deploys is demonstrated on the velocity curve. The first change h
```

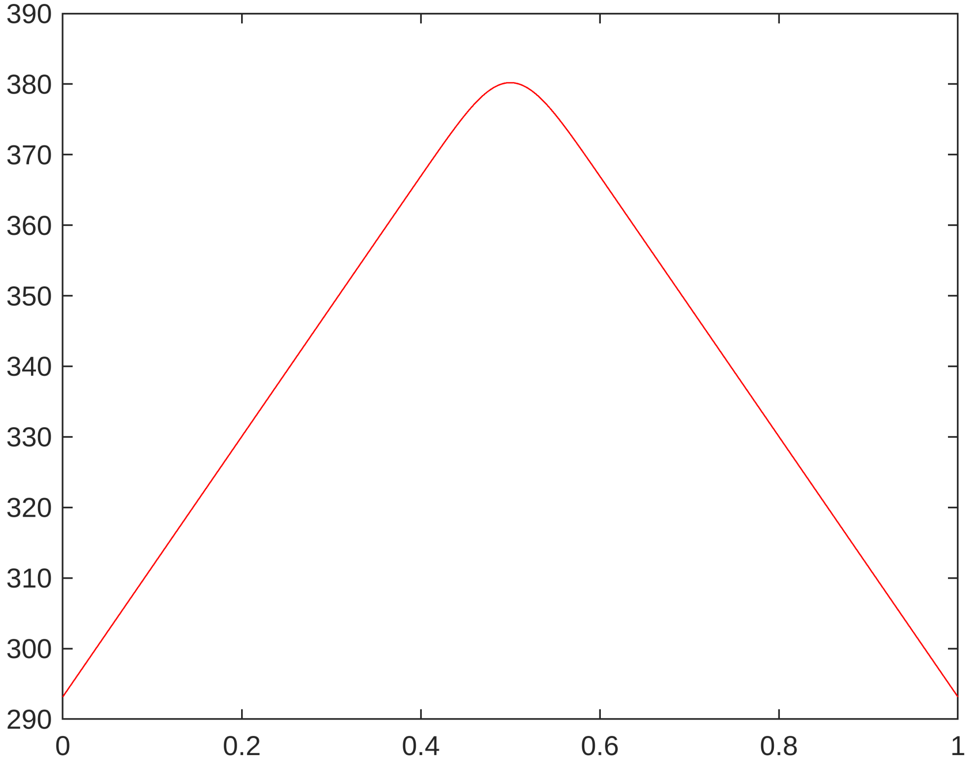
Part (c)

```
% impact in (m/s)
impact_velocity  = velocity(end)
```

```
impact_velocity = -27.1337
```

Problem 5. Extra Credit: Integral equations, 20 pts

```
%% Boundry value problems
S = 2.5e4;
epsilon = 5e-2;
beta = 2.3e-3;
cp = 210.5;
rho = 12.4;
a = 293.15;
b = a;

f = @(t) -S/(rho*cp*beta)*exp(-((t-0.5)/epsilon).^2);

integrand1 = @(t) t.*f(t);
integrand2 = @(t) (t-1).*f(t);

N = 1000;
j = 0:N;
h = 1/N;
x = j*h;
u = zeros(1, N+1);

for j = 0:N
    u(j+1) = (a-integral(integrand1,0,x(j+1)))*(1-x(j+1))+...
        (b+integral(integrand2,x(j+1),1))*x(j+1);
end

plot(x,u,'r-')
```




```
%% PART B

%define function g
%define u exact

uexact = @(x) (a-integral(integrand1,0,x)).*(1-x) +...
    (b + integral(integrand2,x,1)).*x;
g = @(x) uexact(x) - (273.15+70);
xbunrn = fzero(g,[0 0.5])
```

```
xbunrn = 0.2710
```

Functions

```
function intf = mytrap(f,intrvl,n)

h = (intrvl(2)-intrvl(1))/n;
j = 0:n;
x = intrvl(1) + j*h;
fx = f(x);
```

```
intf = h/2*(fx(1)+fx(n+1)) + h*sum(fx(2:n));
end



function intf = mysimp(f,intrvl,n)

h = (intrvl(2) - intrvl(1))/n;
j = 0:n;
x = intrvl(1) + j*h;
fx = f(x);

intf = h/3*(fx(1)+fx(n+1)) + 4*h/3*sum(fx(2:2:n)) + 2*h/3*sum(fx(3:2:n-1));
end
```