Table of Contents

E7 - Lab 11 Solutions

```
format compact
format short
clear all
clc
close all
```

Question 1.1

```
type speedFD
t = [0 13 20 24.5 28 31 33.5 35.5 37];
x = [0 200 400 600 800 1000 1200 1400 1600];


function [speed, acceleration] = speedFD(x, t, output_units)

if strcmpi(output_units, 'mph')
```

```
    x = x/5280; %Convert to miles
    t = t/3600; %Convert to hours
end

last_index = numel(x);

%Speed Calculations
speed(1) = (x(2) - x(1))/(t(2) - t(1));
speed(2:(last_index - 1)) = (x(3:end) - x(1:(end - 2)))./(t(3:end)
 -...
    t(1:(end - 2)));
speed(last_index) = (x(last_index) - x(last_index - 1))/
(t(last_index)...
    - t(last_index - 1));

%Acceleration Calculations
acceleration(1) = (speed(2) - speed(1))/(t(2) - t(1));
acceleration(2:(last_index - 1)) = (speed(3:end) - ...
    speed(1:(end - 2)))./(t(3:end) - t(1:(end - 2)));
acceleration(last_index) = (speed(last_index) - ...
    speed(last_index - 1))/(t(last_index) - t(last_index - 1));

end
```

Test case 1

```
[v1, a1] = speedFD(x, t, 'fps')

v1 =
  Columns 1 through 7
   15.3846   20.0000   34.7826   50.0000   61.5385   72.7273   88.8889
  Columns 8 through 9
  114.2857  133.3333
a1 =
  Columns 1 through 7
    0.3550    0.9699    2.6087    3.3445    3.4965    4.9728    9.2352
  Columns 8 through 9
   12.6984   12.6984
```

Test case 2

```
[v2, a2] = speedFD(x, t, 'mph')

v2 =
  Columns 1 through 7
   10.4895   13.6364   23.7154   34.0909   41.9580   49.5868   60.6061
  Columns 8 through 9
   77.9221   90.9091
a2 =
   1.0e+04 *
  Columns 1 through 7
    0.0871    0.2381    0.6403    0.8209    0.8582    1.2206    2.2668
  Columns 8 through 9
```

```
    3.1169    3.1169
```

Test case 3

```
t = [0 1 2.1 3.6 5 6.5 8 9.2 10];
x = [1 4 8   5   6 7.8 9 8   4.5];
[v3, a3] = speedFD(x, t, 'mph')

v3 =
  Columns 1 through 7
    2.0455    2.2727    0.2622   -0.4702    0.6583    0.6818    0.0505
  Columns 8 through 9
   -1.5341   -2.9830
a3 =
   1.0e+03 *
  Columns 1 through 7
    0.8182   -3.0569   -3.7979    0.4917    1.4301   -0.7294   -2.9545
  Columns 8 through 9
   -5.4602   -6.5199
```

Question 1.2

```
type myGradient


function [grad] = myGradient(f, bbox, n)

%Set Up Vectors
x_data = linspace(bbox(1), bbox(2), n);
y_data = linspace(bbox(3), bbox(4), n);

%Set Up Partial Derivative Arrays
x_partial = zeros(n, n - 2);
y_partial = zeros(n - 2, n);

%Calculate Partial X Derivatives
for i = 1:numel(x_data)
    for j = 2:(numel(x_data) - 1)
        x_partial(i, (j - 1)) = (f(x_data(j + 1), y_data(end + 1 -
 i))...
            - f(x_data(j - 1), y_data(end + 1 - i)))/(x_data(j + 1)
 -...
            x_data(j - 1));
    end
end

%Calculate Partial Y Derivatives
for i = 2:(numel(y_data) - 1)
    for j = 1:numel(y_data)
        y_partial((i - 1), j) = (f(x_data(j), y_data(end - i + 2))...
            - f(x_data(j), y_data(end - i)))/(y_data(end - i + 2)
 -...
            y_data(end - i));
```

```
    end
end

%Define Gradient
grad = zeros(n - 2, n - 2, 2);
grad(:, :, 1) = x_partial(2:(end - 1), :);
grad(:, :, 2) = y_partial(:, 2:(end - 1));

end
```

Test Case 1

```
f=@(x,y) x.^2-y.^2+1;
grad=myGradient(f,[-1 1 -1 1], 5);
dfdx=grad(:,:,1)
dfdy=grad(:,:,2)
```

```
dfdx =
    -1     0     1
    -1     0     1
    -1     0     1
dfdy =
    -1    -1    -1
     0     0     0
     1     1     1
```

Test case 2

```
g=@(x,y)exp(x).*sin(y);
grad=myGradient(g,[-1 1 -1 1], 6);
dgdx=grad(:,:,1)
dgdy=grad(:,:,2)
```

```
dgdx =
    0.3182    0.4747    0.7082    1.0565
    0.1120    0.1670    0.2492    0.3717
   -0.1120   -0.1670   -0.2492   -0.3717
   -0.3182   -0.4747   -0.7082   -1.0565
dgdy =
    0.4410    0.6579    0.9814    1.4641
    0.5236    0.7812    1.1654    1.7386
    0.5236    0.7812    1.1654    1.7386
    0.4410    0.6579    0.9814    1.4641
```

Test case 3

```
h=@(x,y)((x.^2+x.^3).*(1+y)-y.^2);
grad=myGradient(h,[-10 10 -10 10], 5);
dhdx=grad(:,:,1)
dhdy=grad(:,:,2)
```

```
dhdx =
   540   150   660
```

```
    90    25   110
  -360  -100  -440
dhdy =
  -110   -10   140
  -100     0   150
   -90    10   160
```

Question 2.1

```
type GaussIntegral
```

```
function [I] = GaussIntegral(A, n)

%Define Interval
interval = linspace(-A, A, (n + 1));

%Define the Step Size Between Each Value in Interval
step_size = interval(2) - interval(1);

%Define Function to be Integrated
f = @(x) (1/sqrt(2*pi))*exp(-(x.^2)/2);

%Compute the Integral
I = sum(f(interval(1:(end - 1))).*step_size);

end
```

Test case 1

```
I1 = GaussIntegral(1,25)
```

```
I1 =
    0.6824
```

Test case 2

```
I2 = GaussIntegral(2,40)
```

```
I2 =
    0.9543
```

Test case 3

```
I3 = GaussIntegral(3,50)
```

```
I3 =
    0.9973
```

Test case 4

```
I4 = GaussIntegral(1,5)
```

```
I4 =
    0.6762
```

Question 2.2

```
type roofSheetLength
```

```
function [L_Trap, L_Simp, L_Riem] = roofSheetLength(L_C, H, P, N)

%Define Independent Variable for Integration
x = linspace(0, L_C, N+1);

%Define the Step Size Between Each Value in Interval
step_size = x(2) - x(1);

%Define the Function for Integration
f = @(x) sqrt(1 + (((2*pi*H)/P)*cos((2*pi*x)/P)).^2);

%Calculate Integral Using Trapezoidal Rule
L_Trap = sum(step_size*((1/2)*(f(x(1:(end - 1))) + f(x(2:end)))));

%Calculate Integral Using Simpson's Rule
L_Simp = sum((step_size/3).*(f(x(1:2:(end - 2))) + 4*f(x(2:2:(end -
 1)))...
    + f(x(3:2:end))));

%Calculate Integral Using Riemann Integral
L_Riem = sum(f(x(1:(end - 1))).*step_size);

end
```

Test case 1

```
[L_T1, L_S1, L_R1] = roofSheetLength(72, 1.5, 2*pi, 50)
```

```
L_T1 =
  102.8949
L_S1 =
  102.5351
L_R1 =
  102.9242
```

Test case 2

```
[L_T2, L_S2, L_R2] = roofSheetLength(108, 2, 5, 20)
```

```
L_T2 =
  214.5902
L_S2 =
  214.7156
L_R2 =
```

```
  215.7755
```

Test case 3

```
[L_T3, L_S3, L_R3] = roofSheetLength(90, 2, 8, 30)
```

```
L_T3 =
  131.6522
L_S3 =
  132.6049
L_R3 =
  132.9454
```

Question 3.1

```
type LagrangePolynomial
```

```
function [P] = LagrangePolynomial(x, y)

%Calculate the Number of Data Points
n = numel(x);

%Define the A Matrix
A = zeros(n);
exponent = 0:(n - 1);

for i = 1:n
    A(i, :) = x(i).^exponent;
end

%Define the b Matrix
b = y;

%Solve for the Lagrange Coefficients
P = A\b;

end
```

Test Case 1

```
x1=[1;2]; y1=[2;0];
P1 = LagrangePolynomial(x1,y1)
```

```
P1 =
     4
    -2
```

Test Case 2

```
x2 = [1;2;5;7;8]; y2=[2;0;3;-4;5];
```

```
P2 = LagrangePolynomial(x2,y2)
```

```
P2 =
   19.4444
  -29.5548
   14.6242
   -2.6738
    0.1599
```

Test Case 3

```
x3=(1:10)'; y3=sin(x3);
P3 = LagrangePolynomial(x3,y3)
```

```
P3 =
   -0.6299
    2.8371
   -2.1862
    1.2526
   -0.5601
    0.1483
   -0.0220
    0.0018
   -0.0001
    0.0000
```

Question 3.2

```
type TrinomialIntegral
```

```
function [I] = TrinomialIntegral(a, b, P)

%Calculate Integral
I = (P(1)*(b - a)) + ((P(2)/2)*((b^2) - (a^2))) + ((P(3)/3)*((b^3) -
 (a^3)));

end
```

Test case 1

```
J1 = TrinomialIntegral(0,3,[1;2;1])
```

```
J1 =
    21
```

Test case 2

```
J2 = TrinomialIntegral(1,5,[3;0;3])
```

```
J2 =
   136
```

Test case 3

```
J3 = TrinomialIntegral(-5,8,[1;2;-3])
```

```
J3 =
  -585
```

Question 3.3

```
type SimpsonIntegral
```

```
function [I, e] = SimpsonIntegral(f, a, b, n)

%Define Independent Variable for Integration
x = linspace(a, b, (2*n) + 1);

%Use the Previous Functions to Integrate Via Simpson's Integral
I = 0;

for i = 1:2:(numel(x) - 2)
    current_polynomials = LagrangePolynomial(x(i:(i + 2))', f(x(i:(i +
 2)))');
    I = I + TrinomialIntegral(x(i), x(i + 2), current_polynomials);
end

%Calculate Integral Using the Integral Function
true = integral(f, a, b);

%Calculate the Error Between MATLAB's Integral Function and Simpson's
 Rule
e = abs(true - I);

end
```

Test case 1

```
f = @(x) sin(x);
[I1,e1] = SimpsonIntegral(f,0,pi/2,10)
```

```
I1 =
    1.0000
e1 =
   2.1155e-07
```

Test case 2

```
g = @(x) x.^2 .* cos(x);
[I2,e2] = SimpsonIntegral(g,0,5,15)
```

```
I2 =
  -19.2188
```

```
e2 =
   1.1485e-04
```

Test case 3

```
h = @(x) sin(sin(x));
[I3,e3] = SimpsonIntegral(h, 0, 0.5, 30)
```

```
I3 =
    0.1200
e3 =
   3.2257e-11
```

Published with MATLAB® R2015b