Contents

- Name: Gurpreet Singh
- Problem# 1
- a. Approximate f(0.25) using linear interpolation with x0 = 0, and x1 = 0.5.
- b. Approximate f(0.75) using linear interpolation with x0 = 0.5, and x1 = 1.
- c. Approximate f(0.25) and f(0.75) by using the second degree interpolating polynomial with x0 = 0, x1 = 1, and x2 = 2.
- d. Which approximations are better and why?
- Problem# 2
- a. Show that the polynomials P(x) = 3 - 2(x + 1) + 0(x + 1)(x) + (x + 1)(x)(x - 1) and Q(x) = -1 + 4(x + 1) - 3(x + 2)(x + 1) + (x + 2)(x + 1)(x) both interpolate the data f(-2) = -1, f(-1) = 3, f(0) = 1, f(1) = -1, f(2) = 3.
- b. Why does part (a) not violate the uniqueness property of interpolating polynomials?
- 3. Problem #3
- Problem # 4

Name: Gurpreet Singh

```
% HW# 2
```

Problem# 1

```
% Let f(x) = e^x, for 0 <= x <= 2
```

a. Approximate f(0.25) using linear interpolation with x0 = 0, and x1 = 0.5.

```
n = linspace(0, 2, 5);   %      0    0.5000    1.0000    1.5000    2.0000
y = exp(1).^n;           % 1.0000    1.6487    2.7183    4.4817    7.3891

% y(x)   =          a0 + a1(x)
% y(0)   = 1.0000 = a0 + a1(0)
% y(0.5) = 1.6487 = a0 + a1(0.5)

% [1   0]  [a0]     [1.0000]
% [1 0.5]  [a1]  =  [1.6487]

% a0 = 1.0000
% a1 = 1.2974

% ans = 1.0000 + 1.2974*(0.25);
% ans = 1.3244

y1 = [y(1); y(2)];
x1 = [0; 0.5];
a1 = polyfit(x1, y1, 1);
q1 = polyval(a1, 0.25)
```

```
q1 =

    1.3244
```

b. Approximate f(0.75) using linear interpolation with x0 = 0.5, and x1 = 1.

```
% y(x)     =          a0 + a1(x)
% y(0.5)   = 1.6487 = a0 + a1(0.5)
% y(1.0)   = 2.7183 = a0 + a1(0.75)

% [1 0.50]  [a0]     [1.6487]
% [1 1.00]  [a1]  =  [2.7183]

% a0 = 0.5792
% a1 = 2.1391

% ans = 0.5792 + 2.1391*(0.75);
% ans = 2.1835

y2 = [y(2); y(3)];
x2 = [0.5; 1.0];
a2 = polyfit(x2, y2, 1);
q2 = polyval(a2, 0.75)
```

```
q2 =

    2.1835
```

c. Approximate f(0.25) and f(0.75) by using the second degree interpolating polynomial with x0 = 0, x1 = 1, and x2 = 2.

```
% y(x)          = a0 + a1(x) + a2(x)^2
% y(0) = 1.0000 = a0 + a1(0) + a2(0)^2
% y(1) = 2.7183 = a0 + a1(1) + a2(1)^2
% y(2) = 7.3891 = a0 + a1(2) + a2(2)^2

% [1   0   0]  [a0]  =  [1.0000]
% [1   1   1]  [a1]  =  [2.7183]
% [1   2   4]  [a2]  =  [7.3891]

% x = [1 0 0; 1 1 1; 1 2 4;];
% y = [1; 2.7183; 7.3891];
% x\y

% a0 = 1.0000
% a1 = 0.2421
% a2 = 1.4762

% f(0.25)
% -------
```

```
% ans = 1.0000 + 0.2421*(0.25) + 1.4762*(0.25)^2;
% ans = 1.1528

y3 = [y(1); y(3); y(5)];
x3 = [0.0; 1.0; 2.0];
a3 = polyfit(x3, y3, 2);
q3 = polyval(a3, 0.25)

% f(0.75)
% -------
% ans = 1.0000 + 0.2421*(0.75) + 1.4762*(0.75)^2;
% ans = 2.0119
q4 = polyval(a3, 0.75)
```

```
q3 =

    1.1528


q4 =

    2.0119
```

d. Which approximations are better and why?

```
nn = linspace(0, 2, 100);
yy = exp(1).^nn;

% disp('The linear interpolation is a better approximation. It can be seen in the graph below.
 Linear interpolation worked better because the data was small and it was smooth.')

% figure
% plot(nn, yy)
% hold on
% plot(n, y)
% hold on
% plot(0.25, q1, 'b*')
% hold on
% plot(0.75, q2, 'b*')
% hold on
% plot(0.25, q3, 'r*')
% hold on
% plot(0.75, q4, 'r*')
% % , 0.75, q2, 0.25, q3, 0.75, q4)
%
% legend('e^x 100 points 0 <= x <= 2 ', 'e^x 5 points 0 <= x <= 2', 'linear interpolation 1st
degree', 'second degree interpolating polynomial');
```

Problem# 2

a. Show that the polynomials P(x) = 3 - 2(x + 1) + 0(x + 1)(x) + (x + 1)(x)(x - 1) and Q(x) = -1 + 4(x + 1) - 3(x + 2)(x + 1) + (x + 2)(x + 1)(x) both interpolate the data f(-2) = -1, f(-1) = 3, f(0) = 1, f(1) = -1, f(2) = 3.

```
% a.
% ----
name = {'n';'Q(x)';'P(x)'};
n = [1; 2; 3];
px = zeros(1, 100000);
qx = zeros(1, 100000);

for x=1:100000
   px(x) = 3 - 2*(x + 1) + 0*(x + 1)*(x) + (x+1)*(x)*(x - 1);
   qx(x) = -1 + 4*(x+2) - 3*(x+2)*(x+1) + (x+2)*(x+1)*(x);
end

% plotted the graph of px, qx from 1, 100000, and the lines are both the
% same because they output the same value for the f(x). They are the same
% polynomial.
% figure
% plot(px)
% hold on
% plot(qx)
% hold on
% plot(-qx)
```

b. Why does part (a) not violate the uniqueness property of interpolating polynomials?

```
% ----
% P(x) = 3 - 2(x + 1) + 0(x + 1)(x) + (x + 1)(x)(x - 1)
% P(x) = 3 - 2(x + 1) + (x + 1)(x)(x - 1)
% P(x) = 3 - 2x - 2 + (x + 1)(x)(x - 1)
% P(x) = 3 - 2x - 2 + x^3 - x
% P(x) = x^3 - 3x + 1

% Q(x) = -1 + 4(x + 2) - 3(x + 2)(x + 1) + (x + 2)(x + 1)(x)
% Q(x) = -1 + 4x + 8 - 3x^2 - 9x - 6 + x^3 + 3x^2 + 2x
% Q(x) = -1 + 4x + 8 - 9x - 6 + x^3 + 2x
% Q(x) = -1 - 3x + 8 - 6 + x^3
% Q(x) = x^3 - 3x + 1

% P(x) = Q(x) both interpolate the same polynomials.
```

3. Problem #3

```
% Analogous to LinInterp2D, write a function CubicInterp2D(xc,yc,a,b,n,c,d,m,fA) that does cub
ic interpolation from a matrix of
% f(x, y) evaluations. Start by figuring out ?where? (xc, yc) is in the grid with respect to x
 = linspace(a,b,n) and
% y = linspace(c,d,m). Suppose this is the situation: as in LinearInterp2D.

a = 0; b = 5; n = 10;
c = 0; d = 5; m = 10;

fA = SetUp('Humps2D', a, b, n, c, d, m);
xc = a+3;
yc = c+4;
```

```
% cubically interpolated value for f(xc, yc).
z = CubicInterp2D(xc, yc, a, b, n, c, d, m, fA);
E = abs(z - Humps2D(xc, yc))
```

```
E =

    0.1448
```

Problem # 4

```
% Suppose f(u,v) is a function of two variables defined everywhere in the plane.  Assume that x, y and fvals
% are column 3-vectors and p is a column 2-vector.  Assume that (p(1), p(2)) is inside the triangle
% defined by (x(1),y(1)), (x(2),y(2)),  and (x(3),y(3)) and that  fvals(i) = f(x(i),y(i)) for i=1:3.
% fp is an estimate of f(p(1),p(2)) obtained by linear interpolation.

x = [0; 5; 1];
y = [0; 10; 5];
fvals = [0; 50; 5];
p = [5; 1];
InterpTri(x, y, fvals, p);
```

```
fp =

         0   -0.6250         0
         0    5.0000         0
         0    0.5000         0
```