Question 1.

a.

$$
\begin{aligned}
\int_0^4 \cos(\sqrt{x})dx = \int_0^2 2u\cos(u)du \\
&= 2u\sin(u)|_0^2 - 2\int_0^2 \sin(u)du \\
&= 4\sin(2) + 2\cos(u)|_0^2 \\
&= 4\sin(2) + 2\cos(2) - 2 \\
&\approx 0.8049
\end{aligned}
$$

b.

```
>> abs(MySimpson(a,b,f,n) - 4*sin(2) - 2*cos(2) + 2)

ans =

      3.03179211513793e-07

>> abs(TrapezoidRule(a,b,f,n) - 4*sin(2) - 2*cos(2) + 2)

ans =

       0.00363559944609371

>>
```

Figure 1:

Thus Simpson gives a more accurate estimation.

c.

```
>> abs(MyRomberg(a,b,f,5) -4*sin(2) - 2*cos(2) + 2)

ans =

  Columns 1 through 3

         0.362810292697273         0.804896034208442         0.804896034208442
        0.0908445187751643      0.000189260801128022         0.804896034208442
        0.0227200095768105      1.18398440256406e-05      1.17802190224836e-08
       0.00568055751637919      7.40162901635699e-07      1.84160242611142e-10
       0.00142017407625472      4.62628797492926e-08      2.87814216903826e-12

  Columns 4 through 5

         0.804896034208442         0.804896034208442
         0.804896034208442         0.804896034208442
         0.804896034208442         0.804896034208442
      9.54791801177635e-14         0.804896034208442
      4.44089209850063e-16      4.44089209850063e-16
```

Figure 2:

Minimum m is thus 3.

2.

```
A = [4 1 -1 1; 1 4 -1 -1; -1 -1 5 1; 1 -1 1 3];
b = [-2 -1 0 1]';
tic;
part_a = A^-1 * b
toc;
x0 = [0 0 0 0];
tic;
jacobi = Jacobi(x0, A, b, 20, 10^-4)'
toc;
tic;
gs = GS(x0, A, b, 20, 10^-4)'
toc;
jacobi_diff_sum = sum(abs(jacobi - part_a))
gs_diff_sum = sum(abs(gs - part_a))
```

```
part_a =

        -0.753424657534247
        0.0410958904109589
        -0.280821917808219
         0.691780821917808

Elapsed time is 0.000702 seconds.

jacobi =

        -0.753358462485527
        0.0410526408513485
        -0.280792010968246
         0.691699564164386

Elapsed time is 0.000505 seconds.

gs =

        -0.753393307476671
        0.0410851125291763
        -0.280807121120624
         0.691761847042157

Elapsed time is 0.000402 seconds.

jacobi_diff_sum =

      0.000220609201725036


gs_diff_sum =

      7.58995026053921e-05
```

Figure 3:

d.

Based on the sum of absolute value of the difference, the more accurate method is Gauss-Seidel and the fastest one is Gauss-Seidel.

```
function Sol = Jacobi(x0,A,b , N0 , Tol )
  Sol = NaN;
  i=1;
  x1 = zeros(1, length(x0)) ;
  while i ≤ N0
    for j =1: length ( x1 )
      x1(j) = (b(j) + A(j,j) * x0(j) - dot(A(j,:),x0))/A(j,j);
    end
    if norm(x0 - x1) < Tol
      Sol = x1;
      break;
    end
    i= i +1;
    x0 = x1 ;
  end
  if i > N0
    fprintf ( ' More iteration than % d is needed .\ n ' , N0 )
  end
  Iter = i;
end
```

Figure 4:

```
function Sol = GS( x0 , A , b , N0 , Tol )
  i = 1;
  x1 = x0 ;
  while i ≤ N0
    y = x0 ;
    for j=1:length( x1 )
      x1(j) = ( b( j ) + A(j , j ) * x0( j ) - dot( A(j ,:) , x0 ) ) / A(j ,j ) ;
      x0 = x1 ;
    end
    if norm ( x1 - y ) < Tol
      Sol = x1 ;
      break ;
    end
    i = i +1;
  end
  if i > N0
    fprintf ( ' More iteration than % d is needed . \ n ' , N0 )
  end
end
```

Figure 5:

3.

```
Xd = [5 4 6 5 5 5 6 6 2 7 7]';
Yd = [85 103 70 82 89 98 66 95 169 70 48]';
m = 1;
coef = LSQ(Xd,Yd,m)
f = @(x) coef(2)*x + coef(1);
hold on ;
fplot(f,[2,8])
plot(Xd, Yd, "o")
f(3)
m = 3;
coef = LSQ(Xd,Yd,m)
g = @(x) coef(4) * x^3 + coef(3) * x^2 + coef(2)*x + coef(1);
fplot(g,[2, 8])
plot(Xd, Yd, "o")
g(3)
```

```
coef =

  195.4685
  -20.2613


ans =

  134.6847


coef =

  340.1549
 -122.4378
   21.1127
   -1.3369

Warning: Function behaves unexpectedly on array inputs. To improve performance,
properly vectorize your function to return an output with the same size and
shape as the input arguments.

ans =

  126.7606
```

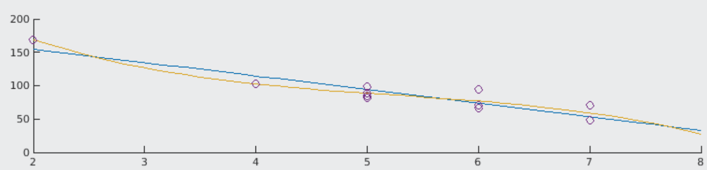

Figure 6:

4.

a.

$$\int_{-\pi/2}^{\pi/2} 1^2 dx = \pi$$

$$\int_{-\pi/2}^{\pi/2} x^2 dx = \frac{\pi^3}{12}$$

$$\int_{-\pi/2}^{\pi/2} (x^2 - \pi^2/12)^2 dx = \frac{\pi^5}{180}$$

$$\int_{-\pi/2}^{\pi/2} x dx = 0$$

$$\int_{-\pi/2}^{\pi/2} (x^2 - \pi^2/12) dx = 0$$

$$\int_{-\pi/2}^{\pi/2} x(x^2 - \pi^2/12) dx = 0$$

b.

```
f = @(x) x*cos(x);
phi1 = @(x) 1;
phi2 = @(x) x;
phi3 = @(x) x^2 - pi^2/12;
phi1S = @(x) phi1(x) * phi1(x);
phi2S = @(x) phi2(x) * phi2(x);
phi3S = @(x) phi3(x) * phi3(x);
a = -pi/2;
b = pi/2;
n = 20;
alpha1 = MySimpson(a,b,phi1S,n);
alpha2 = MySimpson(a,b,phi2S,n);
alpha3 = MySimpson(a,b,phi3S,n);
a0 = MySimpson(a,b,@(x) f(x)* phi1(x), n)/alpha1
a1 = MySimpson(a,b,@(x) f(x)* phi2(x), n)/alpha2
a2 = MySimpson(a,b,@(x) f(x)* phi3(x), n)/alpha3
g = @(x) a0 * phi1(x) + a1 * phi2(x) + a2 * phi3(x);
hold on ;
fplot(f)
fplot(g)
hold off;
```

```
a0 =

  -1.4803e-17


a1 =

    0.3618


a2 =

   6.8375e-18

Warning: Function behaves unexpectedly on array inputs. To improve performance,
properly vectorize your function to return an output with the same size and
shape as the input arguments.
Warning: Function behaves unexpectedly on array inputs. To improve performance,
properly vectorize your function to return an output with the same size and
shape as the input arguments.
```

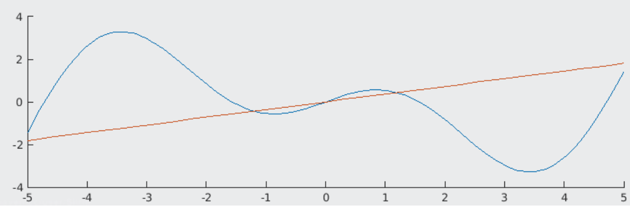

Figure 7:

5.

0.1 a.

We can first rewrite it

$$F(x, y, z) = (x^2 + y - 37, x - y^2 - 5, x + y + z - 3)$$

and the jacobian is

$$\begin{pmatrix} 2x & 1 & 0 \\ 1 & -2y & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

b.

```
N0 =20; Tol =10^(-3) ;
X0 =[0 0 0]';
F = @(x ,y , z ) [x^2+y-37;x-y^2-5;x+y+z-3];
J = @(x ,y , z ) [2*x 1 0; 1 -2*y 0; 1 1 1];
i =1;
while i <= N0
  X1 = X0 - J ( X0 (1) , X0 (2) , X0 (3) ) ^( -1) * F ( X0 (1) , X0 (2) ,X0 (3) ) ;
  if norm ( X1 - X0 ) < Tol
    X1
    break ;
  end
  i = i +1;
  X0 = X1 ;
end
```

```
X1 =

    6.0000
    1.0000
   -4.0000
```

Figure 8: