Problem 2 Section 2.5 Computer Problem 1

```
Function:
function [x,i] = Jacobi_Method(a,b,k)
n = length(b);
d = diag(a);
r = a - diag(d);
x = zeros(n,1);
for i = 1:k
   x = (b - r*x)./d;
   if norm(ones(n,1) - x, 'inf') < 10^(-6)
      break;
   end
end
end

function [a,b] = sparsesetup(n)
e = ones(n,1);
a = spdiags([-e 3*e -e], -1:1,n,n);
b = ones(n,1);
b(1) = 2; b(n) = 2;
end

Main code:
% input n, k
format long
[a,b] = sparsesetup(n);
[x,i] = Jacobi_Method(a, b, k);
error = norm(a * x - b,'inf');
```

When n = 100, the number of steps i = 35, backward error = 6.867832081924874e-07
When n = 100000, the number of steps i = 35, backward error = 6.867832081924874e-07

Problem 3 Solve problem 2 again by Gauss-Seidel Method

```
Function:
function [x,i] = Gauss_Seidel(a,b,k)
n = length(b);
d = diag(a);
l = tril(a) - diag(d);
u = triu(a) - diag(d);
x = zeros(n,1);
for i = 1:k
   B = b - u * x;
   for j = 1:n
      x(j) = (B(j) - l(j,:) * x)./d(j);
```

```
    end
    if norm(ones(n,1) - x, 'inf') < 10^(-6)
        break;
    end
end
end
```

```
Main code:
% input n, k
format long
[a,b] = sparsesetup(n);
[x,i] = Gauss_Seidel(a, b, k);
error = norm(a * x - b,'inf');
```

When n = 100, the number of steps i = 20, backward error = 9.564705890641179e-07
When n = 100000, the number of steps i = 20, backward error = 9.564705892861625e-07

Problem 5 Section 2.6 Computer Problem 5

```
Function:
function [a,b] = sparsesetup2(n)
e = ones(n,1); n2 = n/2;
a = spdiags([-e 3*e -e],-1:1,n,n);
c = spdiags([e/2],0,n,n); c = fliplr(c); a = a + c;
a(n2+1,n2) = -1; a(n2, n2+1) = -1;
b = zeros(n,1);
b(1) = 2.5; b(n) = 2.5; b(2:n-1) = 1.5; b(n2:n2+1) = 1;
end

function [x,i,r] = conj_grad(a,b,k)
n = length(b);
x = zeros(n,1);
d = b - a * x;
r = d;
er = 1;
for i = 1:k
    if er == 0
        break;
    end
    alpha = r' * r ./ (d' * a * d);
  t = x;
    x = x + alpha * d;
    er = x - t;
    temp = r' * r;
    r = r - alpha * a * d;
    beta = r'* r ./ temp;
    d = r + beta * d;
end
end

% input n,k
format long
[a,b] = sparsesetup2(n);
[x,i,r] = conj_grad(a, b, k);
norm(r,'inf');
```

When n = 100, the size of the final residual is less than 3.071518984891662e-17, the number of steps is 35
When n = 1000, the size of the final residual is less than 2.257871305271420e-17, the number of steps is 36
When n = 10000, the size of the final residual is less than 2.299547838579736e-17, the number of steps is 36

Problem 7 Section 2.7 Computer Problem 1

```
Function:
function x = Newtons_meth(k)
syms u v
f1 = u^2 + v^2 - 1;
f2 = (u - 1)^2 + v^2 - 1;
df =jacobian([f1,f2],[u,v]);
n = 2; u = 1; v = 1; % change initial guess every time
x = [u;v];
for i = 1:k
    f = [eval(f1); eval(f2)];
    df1 = eval(df);
    [A,B] = gauss_em(df1,-f);
    s = back_sub(A,B);
    x = x + s';
    u = x(1); v = x(2);
end
end
Change f1 and f2 when compute different questions
```

Answer:
(a)
[u,v] =

0.500000000000000
$\pm$ 0.866025405007364

(b)
[u,v] =

$\pm$ 0.894427190999916
$\pm$ 0.894427190999916

(c)
[u,v] =

2.759591794226543
$\pm$ 0.950703275312869

Problem 8 Section 2.7 Computer Problem 7

```
Function:
function [x,i] = broyden(k)
syms u v
f1 = u^2 + v^2 - 1;
f2 = (u - 1)^2 + v^2 - 1;
% change f1, f2 when compute different questions
% f1 = u^2 + 4 * v^2 - 4;
% f2 = 4 * u^2 + v^2 - 4;
% f1 = u^2 - 4 * v^2 - 4;
% f2 = (u - 1)^2 + v^2 - 4;
n = 2; u = 1; v = 1;
x = [u;v];
a = eye(n);
for i = 1:k
    f = [eval(f1); eval(f2)];
    temp1 = x;
    x = x - inv(a) * f;
    u = x(1); v = x(2);
    temp2 = f;
    f = [eval(f1); eval(f2)];
    a = a +((f - temp2 - a * (x - temp1)) * (x - temp1)') / ((x - temp1)' * (x
- temp1));
    if x - temp1 == 0
        break;
    end
end
end
```

Answer:
When matlab will treat error as 0 when the error is less than 10^(-16), the precision is 10^(-16)
(a)
The number of steps i= 12

(b)

The number of steps i= 12

(c)

The number of steps i= 43