Homework 6

1. Problem 19

a.) Approximate Jacobian with finite differences

J(1,1)	J(2,2)
-3.827471045703409e-01	1.077119999999716e+00

b.) Modified Newton Jacobian approximation

X1(2)	x2(2)
-3.732058376687573e-01	5.626232820955775e-02

2. Problem 20

a.)

J(1,2)	J(2,1)
-5.699987519708682e+00	-4.429176102630938e+00

b.) Implement modified newton method

```
function [x,res,niter] = newtonsys_approxJ_fdh(Ffun, ...
    x0, h, tol, nmax, varargin)

    niter = 1; err = tol + 1; x = x0;
    while err >= tol && niter < nmax
        J = approxJ_fdh(Ffun, x0, h, varargin{:});
        F = Ffun(x, varargin{:});
        delta = - J\F;
        x0 = x;
        x = x + delta;
        err = norm(delta);
        niter = niter + 1;
    end
    res = norm(Ffun(x, varargin{:}));
    if (niter==nmax && err> tol)
        fprintf([' Fails to converge within maximum ',...
            'number of iterations.\n',...
            'The iterate returned has relative ',...
            'residual %e\n'], res);
        err
    else
        fprintf(['The method converged at iteration ',...
            '%i with  residual %e\n'],niter, res);
    end
    return
```

c.) Solve system of nonlinear equations from 20a

X1	-3.732058376695438e-01
X2	5.626232820945344e-02
Res	3.744555404993960e-12
Niter	6

3. Problem 21

a.) Implement Jacobian approximation for complex variable method

```
function J = approxJ_compl(fun, x, EPS, varargin)
    n = length(x);
    J = zeros(n, n);
    %EPS = 2*eps;
    for j = 1:n
        % form the approximate Jacobian
        e = zeros(n,1); e(j) = 1;
        J(:, j) = (1/EPS)*imag(fun(x + EPS*sqrt(-1)*e, varargin{:}));
    end
```

b.)

Jf(1,2)	J(2,1)
-5.791364443706561e+00	-4.432905227626845e+00

c.) Implement newtonsys_approxJ_compl

```
function [x, res, niter] = newtonsys_approxJ_compl(Ffun, ...
x0, EPS, tol, nmax, varargin)

niter = 1; err = tol + 1; x = x0;
    while err >= tol && niter < nmax
        J = approxJ_compl(Ffun, x, EPS, varargin{:});
        F = Ffun(x, varargin{:});
        delta = - J\F;
        x0 = x;
        x = x + delta;
        err = norm(delta);
        niter = niter + 1;
    end
    res = norm(Ffun(x, varargin{:}));
    if (niter==nmax && err> tol)
        fprintf([' Fails to converge within maximum ',...
            'number of iterations.\n',...
            'The iterate returned has relative ',...
            'residual %e\n'], res);
        err
    else
        fprintf(['The method converged at iteration ',...
            '%i with  residual %e\n'],niter, res);
```

d.)

	1
	2
	7

4. Problem 22

a.) Implement Broyden's method

```
function [x, res, niter, err, B]=broyden(Ffun, B0, ...
x0, tol, nmax, varargin)

B = B0;
x = x0;
err = tol + 1;
niter = 0;
F = Ffun(x, varargin{:});

while err >= tol && niter < nmax
    delta = -B\F;
    x = x + delta;
    F = Ffun(x, varargin{:});
    B = B + F*delta'/(delta'*delta);
    err = norm(delta);
    niter = niter + 1;
end
res = norm(Ffun(x, varargin{:}));

if (niter==nmax && err> tol)
    fprintf([' Fails to converge within maximum ',...
        'number of iterations.\n',...
        'The iterate returned has relative ',...
        'residual %e\n'], res);
    err;
else
    fprintf(['The method converged at iteration ',...
        '%i with  residual %e\n'],niter, res);
end
```

b.) Approximating x

K	x1(k)	x2(k)
1	-3.135967032377301e-01	3.555231731285171e-01
2	-2.393615167309388e-02	1.264580761861792e-01

c.) Use Bryoden to solve system

X1	-3.667010034769886e-01
X2	5.612745145275295e-02
Res	1.600627259928237e-12
Err	2.745149447436020e-10
niter	6