Table of Contents


```
% Craig Medlin
% AERO 220
% HW 3

% Use a 'switch' statement to allow selection of different problems

function [answer] = CMedlin_HW3(h)

clc;
problem = h;
answer = 0;

for problem = 1:5

switch(problem)

    case 1
```

Problem 1 - Jacobi Iterative Method

```
        clear all;

        % Initialize values
        A = [5 6 7 5 -1; 8 -4 -1 0 -3; 2 1 -1 3 6; ...
            -9 10 1 -4 6; 9 5 -5 -8 4];

        b = [-3; 5; 2; 9; -2];

        eps = 10^-6;

        n = size(A);

        % Enforce Diagonally Dominant
        for i=1:n
            r_sum = 0;
            for j=1:n
                r_sum = r_sum + abs(A(i,j));
            end
            A(i,i) = r_sum;
        end
```

```
% Solve
[x_k, iterations, fnorm] = jacobiSolve(A,b,eps);

% Print results
array2table(x_k)
fprintf('Final 2-norm: %d\n', fnorm);
fprintf('Number of iterations: %i\n', iterations);
```

```
ans =

      x_k
    _________

    -0.29996
      0.4697
     0.15874
     0.05104
   -0.014415

Final 2-norm: 5.986616e-07
Number of iterations: 19
```

```
    case 2
```

Problem 2 - Gauss-Seidel

```
clear all;

A = [24 6 7 5 -1; 8 16 -1 0 -3; 2 1 13 3 6; ...
    -9 10 1 30 6; 9 5 -5 -8 31];

b = [-3; 5; 2; 9; -2];

eps = 10^-6;

[x_k, iterations, fnorm] = gSeidelSolve(A,b,eps);

% Print results
array2table(x_k)
fprintf('Final 2-norm: %d\n', fnorm);
fprintf('Number of iterations: %i\n', iterations);
```

```
ans =

      x_k
    _________

    -0.29996
      0.4697
     0.15874
    0.051039
```

```
    -0.014415

Final 2-norm: 2.650819e-07
Number of iterations: 9

    case 3
```

Problem 3 - Newton's Method

```
        format long

        % Provided functions
        f1 =  @(x,y) (y + x^(1/2));
        f2 =  @(x,y) ((x-3)^2 + y^2 - 5);

        % Error Tolerance
        eps = 10^-4;

        % Jacobian components (evaluated using gradient(f#) and
 entered
        % manually)
        df1_x = @(x) 1/(2*x^(1/2));
        df1_y = @(y) 1;
        df2_x = @(x) 2*x - 6;
        df2_y = @(y) 2*y;


        % Initial guesses
        a = [1;0];
        b = [5;-2];

        % **** Part A ****

        % Initialize estimates
        j_eval = [df1_x(a(1)) df1_y(a(1));df2_x(a(1)) df2_y(a(1))];
        f_eval = [f1(a(1),a(2));f2(a(1),a(2))];
        est = a - j_eval\f_eval;

        % Initialize incremental vector
        incremental = [1;1];

        % Iterate until tolerance is met
        while norm(incremental) > eps

            % Evaluate Jacobian using previous estimate
            j_eval = [df1_x(est(1)) df1_y(est(1));df2_x(est(1))
 df2_y(est(1))];

            % Evaluate function using previous estimate
            f_eval = [f1(est(1),est(2));f2(est(1),est(2))];

            % Evaluate new estimate
            est2 = est - j_eval\f_eval;
```

```
            % Evaluate incremental vector
            incremental = est2 - est;

            % Store new estimate as current estimate
            est = est2;

        end % While

        % Store solution for part a
        est_a = est;

        % Compute solution accuracy
        f_est_a = [f1(est(1),est(2)); f2(est(1), est(2))];

        %***** Part B  *****

        % Initialize estimates
        j_eval = [df1_x(b(1)) df1_y(b(1));df2_x(b(1)) df2_y(b(1))];
        f_eval = [f1(b(1),b(2));f2(b(1),b(2))];
        est = b - j_eval\f_eval;

        % Initialize incremental vector
        incremental = [1;1];

        % Iterate until tolerance is met
        while norm(incremental) > eps

            % Evaluate Jacobian using previous estimate
            j_eval = [df1_x(est(1)) df1_y(est(1));df2_x(est(1))
df2_y(est(1))];

            % Evaluate function using previous estimate
            f_eval = [f1(est(1),est(2));f2(est(1),est(2))];

            % Evaluate new estimate
            est2 = est - j_eval\f_eval;

            % Evaluate incremental vector
            incremental = est2 - est;

            % Store new estimate as current estimate
            est = est2;

        end % While

        % Store solution for part a
        est_b = est;

        % Compute solution accuracy
        f_est_b = [f1(est(1),est(2)); f2(est(1), est(2))];
```

```
        % Print results
        fprintf('***** Problem 3: Part A *****\n');
        fprintf('x0 = %i, y0 = %i\n', a(1), a(2));
        array2table(est_a,'RowNames', {'x_est','y_est'})
        array2table(f_est_a, 'RowNames',
 {'f1(x_est,y_est)','f2(x_est,y_est)'})

        fprintf('\n***** Problem 3: Part B *****\n');
        fprintf('x0 = %i, y0 = %i\n', b(1), b(2));
        array2table(est_b,'RowNames', {'x_est','y_est'})
        array2table(f_est_b, 'RowNames',
 {'f1(x_est,y_est)','f2(x_est,y_est)'})
```

```
***** Problem 3: Part A *****
x0 = 1, y0 = 0

ans =

                    est_a
             ____________________

    x_est     0.999975067486462
    y_est    -0.999987533840395


ans =

                             f_est_a
                       ______________________

    f1(x_est,y_est)    -1.74868786118054e-10
    f2(x_est,y_est)     7.47985119780026e-05


***** Problem 3: Part B *****
x0 = 5, y0 = -2

ans =

                    est_b
             ___________________

    x_est     1.00005249369253
    y_est    -1.00002624727651


ans =

                             f_est_b
                       ______________________

    f1(x_est,y_est)    -7.74679653758881e-10
    f2(x_est,y_est)    -0.000157476772612242
```

```
        case 4
```

Problem 4 - Interpolating monomial

```
% Given data points
p = [4.1168 4.19236 4.20967 4.46908; ...
    0.213631 0.214232 0.21441 0.218788];

% Initialize variables
sum = 0;
n = size(p,2);
x_vals = p(1,:);
b = p(2,:)';

% Solve for coefficients for the interpolating monomial
% c0 + c1*x + x2*x^2 + c3*x^3
v = fliplr(vander(x_vals));
c = v\b;

% Print results
array2table(c, 'RowNames',{'c0','c1','c2','c3'})
```

```
ans =

                 c
         ____________________

c0         0.87183881430253
c1       -0.386007738945542
c2       0.0695519306309334
c3     -0.00355244621780782
```

```
        case 5
```

Problem 5 - Neville Table / Divided Difference

```
clear all;
format short;
x0 = [4.1168 4.19236 4.20967 4.45908];
f = @(x) pi*x^3-7*x^2+sin(x)-1;

n = size(x0,2);

P = zeros(n);

% Value to interpolate at
x_est = 4.3;

x = x0;

% Rearrange data in order of closeness to x_est
```

```
        for i = 1:n
            for j = 1:n
                    % If selected element is closer to x_est, swap
                    if abs(x(i)-x_est) > abs(x(j)-x_est)
                        temp = x(i);
                        x(i) = x(j);
                        x(j) = temp;
                    end % if
            end % for j
        end % for i

        % Correct order
        x = fliplr(x)';

        % Calculate f(x) values
        for i = 1:n
            P(i,1) = f(x(i));
        end % for i

        table(:,1) = linspace(1,n,n);

        % For each column
        for j = 2:n

            % For each row
            for i = 1:n-j+1
                P(i,j) = ((x_est - x(i))*P(i+1,j-1) + (x(i+j-1)-
x_est)*P(i,j-1))/(x(i+j-1)-x(i));
            end % i loop

        end % j loop

        table = horzcat(table,x,P);

        % Output Results for Part A
        array2table(table,'VariableNames',
{'i','xi','Pi0','Pi1','Pi2',...
            'Pi3'})

        % Calculate f(x) values
        Y = zeros(n,1)
        for i = 1:n
            Y(i,1) = f(x0(i));
        end % for i


        % Calculate Divided Differences
        DivDiff = divDiff(x0',Y(:,1))


ans =

    i      xi         Pi0        Pi1        Pi2        Pi3
    _    _______    _______    _______    _______    _______
```

```
    1     4.2097     108.44     118.11     118.44     118.43
    2     4.1924     106.59     119.02     118.44          0
    3     4.4591     137.39     119.42          0          0
    4     4.1168      98.73          0          0          0


Y =

     0
     0
     0
     0


DivDiff =

   98.7301  103.9879   32.7580    3.2166
  106.5875  107.0301   33.8590         0
  108.4402  116.0610         0         0
  137.3869         0         0         0


    otherwise
        error('Invalid Problem Number');
end

end

end
```

Published with MATLAB® R2015b