MATH 446: Project 05

Zachary Ferguson

Contents

Code

Gaussian Elimination

```
% Solve a system of linear equations, Ax = b, using naive Guassian Elimination
% Written by Zachary Ferguson

function x = gaussian_elimination(A, b, eps)
    % Solve the equation Ax = b
    % Input:
    %   A - matrix of coefficients to the linear equations
    %   b - Right hand side of the linear equations
    %   eps - tolerance of a zero pivot
    % Output:
    %   x - solved value
    if nargin < 3
            eps = 1e-9;
    end

    n = size(A, 1);

    % Elimination step (O(2/3 * n^3))
    for j = 1 : n-1
        if abs(A(j, j)) < eps
            error('Zero Pivot encountered.');
        end
        for i = j+1 : n
            mult = A(i, j)/A(j, j);
            for k = j+1 : n
                A(i, k) = A(i, k) - mult * A(j, k); % Row operation
            end
            b(i) = b(i) - mult * b(j);
        end
    end

    x = zeros(size(b));

    % Perform Back Substitution
```

```
    for i = n : -1 : 1
        for j = i+1 : n
            b(i) = b(i) - A(i, j) * x(j);
        end
        x(i) = b(i) / A(i, i);
    end
end
```

Hilbert Matrix

```
% Generates the n x n Hilbert matrix where H(i, j) = 1 / (i+j-1)
% Written by Zachary Ferguson

function H = hilbert_matrix(n)
    % Generates the n x n Hilbert matrix where H(i, j) = 1 / (i+j-1)
    % Input:
    %   n - size of matrix
    % Output:
    %   H - nxn Hilbert matrix
    H = zeros(n, n);

    for i = 1 : n
        for j = 1 : n
            H(i, j) = 1 / (i + j - 1);
        end
    end
end
```

LU Decomposition

```
% Decompose the matrix A in to an L and U matrix such that A = LU
% Written by Zachary Ferguson

function [L, U] = lu_decomposition(A, eps)
    % Decompose the matrix A in to an L and U matrix such that A = LU
    % Input:
    %   A - matrix to decompose
    %   eps - tolerance of a zero pivot
    % Output:
    %   [L, U] - deomposed version of A
    if nargin < 2
            eps = 1e-9;
    end

    n = size(A, 1);

    % L is the multipliers of A to get U
    L = eye(n);

    % Elimination step (O(2/3 * n^3))
    for j = 1 : n-1
        if abs(A(j, j)) < eps
            error('Zero Pivot encountered.');
```

```
        end
        for i = j+1 : n
            mult = A(i, j) / A(j, j);
            for k = j : n
                A(i, k) = A(i, k) - mult * A(j, k); % Row operation
            end
            L(i, j) = mult;
        end
    end

    % U is the row echelon form of A
    U = A;
end
```

Solving using LU

```
% Solve a system of linear equations, Ax = b, using LU decomposition
% Written by Zachary Ferguson

function x = lu_solve(L, U, b)
    % Solve the equation Ax = LUx = b acording to LU decomposition
    % Input:
    %   L - Lower triangular matrix of guassian multiplication values
    %   U - Row echelon form of A
    %   b - Right hand side of the linear equations
    % Output:
    %   x - solved value

    n = size(L, 1);

    % Perform Forward Substitution
    c = zeros(size(b));
    for i = 1 : n
        for j = 1 : i-1
            b(i) = b(i) - L(i, j) * c(j);
        end
        c(i) = b(i) / L(i, i);
    end


    % Perform Back Substitution
    x = zeros(size(b));
    for i = n : -1 : 1
        for j = i+1 : n
            c(i) = c(i) - U(i, j) * x(j);
        end
        x(i) = c(i) / U(i, i);
    end
end
```

Main

```
% MATH 446: Project 05
% Written by Zachary Ferguson

function main()
    fprintf('MATH 446: Project 05\nWritten by Zachary Ferguson\n\n');

    fprintf('Gaussian Elimination:\n\n');

    % Q1a
    A = [2 -2 -1 ; 4 1 -2 ; -2 1 -1];
    b = [-2 ; 1 ; -3];
    x = gaussian_elimination(A, b);
    fprintf('Q1a:\n');
    print_Axb(A, x, b);

    % Q1b
    A = [1 2 -1 ; 0 3 1 ; 2 -1 1];
    b = [2 ; 4 ; 2];
    x = gaussian_elimination(A, b);
    fprintf('Q1b:\n');
    print_Axb(A, x, b);

    % Q1c
    A = [2 1 -4 ; 1 -1 1 ; -1 3 -2];
    b = [-7 ; -2 ; 6];
    x = gaussian_elimination(A, b);
    fprintf('Q1c:\n');
    print_Axb(A, x, b);

    % Q2a
    n = 2;
    H = hilbert_matrix(n);
    b = ones(n, 1);
    x = gaussian_elimination(H, b);
    fprintf('Q2a:\n\tn = %d\n', n);
    print_Axb(H, x, b, 'H');

    % Q2b
    n = 5;
    H = hilbert_matrix(n);
    b = ones(n, 1);
    x = gaussian_elimination(H, b);
    fprintf('Q2b:\n\tn = %d\n', n);
    print_Axb(H, x, b, 'H');

    % Q2a
    n = 10;
    H = hilbert_matrix(n);
    b = ones(n, 1);
    x = gaussian_elimination(H, b, 1e-10);
    fprintf('Q2c:\n\tn = %d\n', n);
    print_Axb(H, x, b, 'H');
```

```
    fprintf('\nLU Decomposition:\n\n');

    % Q1a
    A = [3 1 2 ; 6 3 4 ; 3 1 5];
    [L, U] = lu_decomposition(A);
    fprintf('Q1a:\n');
    print_ALU(A, L, U);

    % Q1b
    A = [4 2 0 ; 4 4 2 ; 2 2 3];
    [L, U] = lu_decomposition(A);
    fprintf('Q1b:\n');
    print_ALU(A, L, U);

    % Q1c
    A = [1 -1 1 2 ; 0 2 1 0 ; 1 3 4 4 ; 0 2 1 -1];
    [L, U] = lu_decomposition(A);
    fprintf('Q1c:\n');
    print_ALU(A, L, U);

    % Q2a
    A = [3 1 2 ; 6 3 4 ; 3 1 5];
    b = [0 ; 1 ; 3];
    [L, U] = lu_decomposition(A);
    x = lu_solve(L, U, b);
    fprintf('Q2a:\n');
    print_AbLUx(A, b, L, U, x);

    % Q2b
    A = [4 2 0 ; 4 4 2 ; 2 2 3];
    b = [2 ; 4 ; 6];
    [L, U] = lu_decomposition(A);
    x = lu_solve(L, U, b);
    fprintf('Q2b:\n');
    print_AbLUx(A, b, L, U, x);
end


function print_Axb(A, x, b, nameA)
    if nargin < 4
        nameA = 'A';
    end

    fprintf('\t%s = \n', nameA);
    disp(A);
    fprintf('\tb = \n');
    disp(b);
    fprintf('\tx = \n');
    disp(x);
    fprintf('\tBackwards Error = ||%sx - b|| = %.10f\n\n', nameA, ...
        max(abs(A*x - b)));
end
```

```
function print_ALU(A, L, U, nameA)
    if nargin < 4
        nameA = 'A';
    end

    fprintf('\t%s =\n', nameA);
    disp(A);
    fprintf('\tL =\n');
    disp(L);
    fprintf('\tU =\n');
    disp(U);
    diff = abs(A - L*U);
    fprintf('\tBackwards Error = ||%s - LU|| = %.10f\n\n', nameA, ...
        max(diff(:)));
end

function print_AbLUx(A, b, L, U, x, nameA)
    if nargin < 6
        nameA = 'A';
    end

    fprintf('\t%s =\n', nameA);
    disp(A);
    fprintf('\tb =\n');
    disp(b);
    fprintf('\tL =\n');
    disp(L);
    fprintf('\tU =\n');
    disp(U);
    fprintf('\tx =\n');
    disp(x);
    fprintf('\tBackwards Error = ||%sx - b|| = %.10f\n\n', nameA, ...
        max(abs(A*x - b)));
end
```

Output

```
MATH 446: Project 05
Written by Zachary Ferguson

Gaussian Elimination:

Q1a:
    A =
     2    -2    -1
     4     1    -2
    -2     1    -1

    b =
    -2
     1
    -3
```

```
    x =
     1
     1
     2

    Backwards Error = ||Ax - b|| = 0.0000000000

Q1b:
    A =
     1     2    -1
     0     3     1
     2    -1     1

    b =
     2
     4
     2

    x =
     1
     1
     1

    Backwards Error = ||Ax - b|| = 0.0000000000

Q1c:
    A =
     2     1    -4
     1    -1     1
    -1     3    -2

    b =
    -7
    -2
     6

    x =
    -1
     3
     2

    Backwards Error = ||Ax - b|| = 0.0000000000

Q2a:
    n = 2
    H =
    1.0000    0.5000
    0.5000    0.3333

    b =
     1
     1

    x =
```

```
  -2.0000
   6.0000

   Backwards Error = ||Hx - b|| = 0.0000000000

Q2b:
   n = 5
   H =
   1.0000    0.5000    0.3333    0.2500    0.2000
   0.5000    0.3333    0.2500    0.2000    0.1667
   0.3333    0.2500    0.2000    0.1667    0.1429
   0.2500    0.2000    0.1667    0.1429    0.1250
   0.2000    0.1667    0.1429    0.1250    0.1111

   b =
    1
    1
    1
    1
    1

   x =
  1.0e+03 *

   0.0050
  -0.1200
   0.6300
  -1.1200
   0.6300

   Backwards Error = ||Hx - b|| = 0.0000000000

Q2c:
   n = 10
   H =
   1.0000    0.5000    0.3333    0.2500    0.2000    0.1667    0.1429    0.1250    0.1111    0.1000
   0.5000    0.3333    0.2500    0.2000    0.1667    0.1429    0.1250    0.1111    0.1000    0.0909
   0.3333    0.2500    0.2000    0.1667    0.1429    0.1250    0.1111    0.1000    0.0909    0.0833
   0.2500    0.2000    0.1667    0.1429    0.1250    0.1111    0.1000    0.0909    0.0833    0.0769
   0.2000    0.1667    0.1429    0.1250    0.1111    0.1000    0.0909    0.0833    0.0769    0.0714
   0.1667    0.1429    0.1250    0.1111    0.1000    0.0909    0.0833    0.0769    0.0714    0.0667
   0.1429    0.1250    0.1111    0.1000    0.0909    0.0833    0.0769    0.0714    0.0667    0.0625
   0.1250    0.1111    0.1000    0.0909    0.0833    0.0769    0.0714    0.0667    0.0625    0.0588
   0.1111    0.1000    0.0909    0.0833    0.0769    0.0714    0.0667    0.0625    0.0588    0.0556
   0.1000    0.0909    0.0833    0.0769    0.0714    0.0667    0.0625    0.0588    0.0556    0.0526

   b =
    1
    1
    1
    1
    1
    1
    1
```

```
  1
  1
  1

 x =
1.0e+06 *

-0.0000
 0.0010
-0.0238
 0.2402
-1.2610
 3.7832
-6.7258
 7.0004
-3.9377
 0.9237

 Backwards Error = ||Hx - b|| = 0.0000000002
```

LU Decomposition:

Q1a:

```
    A =
     3     1     2
     6     3     4
     3     1     5

    L =
     1     0     0
     2     1     0
     1     0     1

    U =
     3     1     2
     0     1     0
     0     0     3

    Backwards Error = ||A - LU|| = 0.0000000000
```

Q1b:

```
    A =
     4     2     0
     4     4     2
     2     2     3

    L =
    1.0000         0         0
    1.0000    1.0000         0
    0.5000    0.5000    1.0000

    U =
     4     2     0
```

```
     0     2     2
     0     0     2

    Backwards Error = ||A - LU|| = 0.0000000000

Q1c:
    A =
     1    -1     1     2
     0     2     1     0
     1     3     4     4
     0     2     1    -1

    L =
     1     0     0     0
     0     1     0     0
     1     2     1     0
     0     1     0     1

    U =
     1    -1     1     2
     0     2     1     0
     0     0     1     2
     0     0     0    -1

    Backwards Error = ||A - LU|| = 0.0000000000

Q2a:
    A =
     3     1     2
     6     3     4
     3     1     5

    b =
     0
     1
     3

    L =
     1     0     0
     2     1     0
     1     0     1

    U =
     3     1     2
     0     1     0
     0     0     3

    x =
    -1
     1
     1

    Backwards Error = ||Ax - b|| = 0.0000000000
```

```
Q2b:
    A =
     4     2     0
     4     4     2
     2     2     3

    b =
     2
     4
     6

    L =
    1.0000         0         0
    1.0000    1.0000         0
    0.5000    0.5000    1.0000

    U =
     4     2     0
     0     2     2
     0     0     2

    x =
     1
    -1
     2

    Backwards Error = ||Ax - b|| = 0.0000000000
```