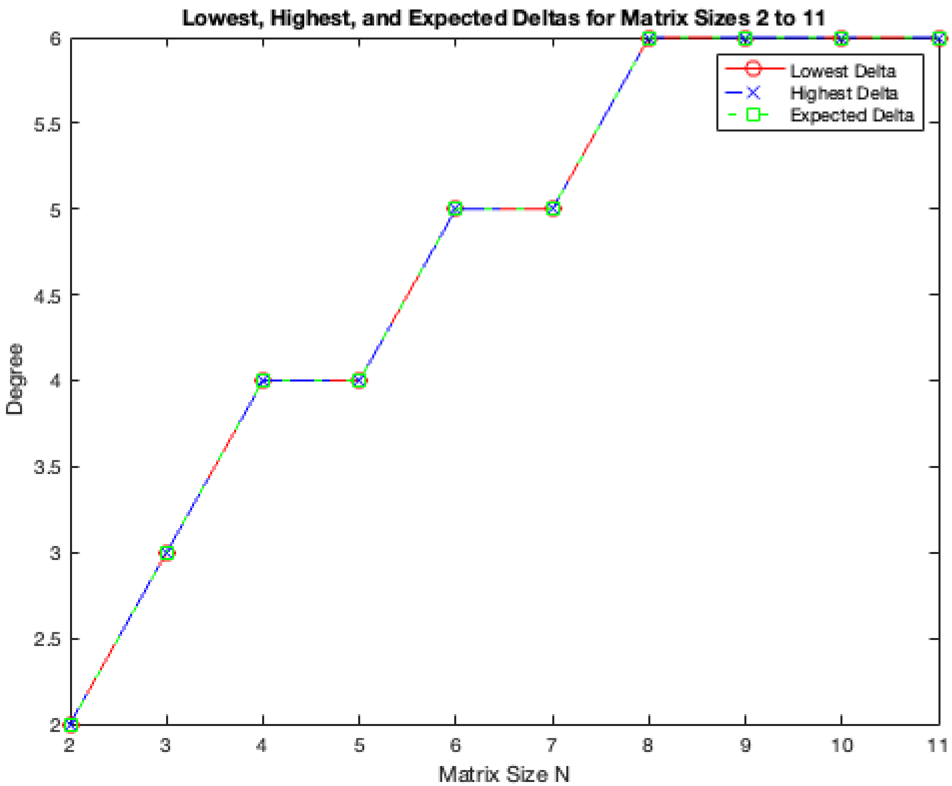
Table of Contents

Multi-Case & Plots

```
function main
    % Run for matrix sizes 2 to 11
    sizes = 2:11;
    lowestDeltas = zeros(size(sizes));
    highestDeltas = zeros(size(sizes));
    expectedDeltas = zeros(size(sizes));

    for i = 1:length(sizes)
        n = sizes(i); % Current matrix size
        X = randn(n); % Random X matrix
        Y = cyclicMatrix(n); % Cyclic Y matrix

        % Find bounds for the degrees
        [delta, Delta] = findMatrixBounds(X, Y, n);
        base = ceil(log2(n^2 + 1) - 1);

        % Store the results
        lowestDeltas(i) = delta;
        highestDeltas(i) = Delta;
        expectedDeltas(i) = base;
    end

    % Plotting the results with markers and different line styles
    plot(sizes, lowestDeltas, 'r-o', 'LineStyle', '-', 'MarkerSize', 8); %
Solid line with circle markers
    hold on;
    plot(sizes, highestDeltas, 'b-x', 'LineStyle', '--', 'MarkerSize', 8); %
Dashed line with cross markers
    plot(sizes, expectedDeltas, 'g-s', 'LineStyle', ':', 'MarkerSize', 8); %
Dotted line with square markers
    hold off;

    xlabel('Matrix Size N');
    ylabel('Degree');
    title('Lowest, Highest, and Expected Deltas for Matrix Sizes 2 to 11');
    legend('Lowest Delta', 'Highest Delta', 'Expected Delta');
end

function Y = cyclicMatrix(n)
    % Generate a cyclic matrix for a given size n
    cyclic_matrix = zeros(n); % Initialize an n x n matrix of zeros
    for i = 1:n
```

```
        cyclic_matrix(i, :) = circshift(1:n, [0, i-1]);
    end
    Y = cyclic_matrix;
end

%    % Display the results
%     clc; close all;
%     fprintf('The smallest degree delta is: %d\n', delta);
%     fprintf('The largest degree Delta is: %d\n', Delta);
%     fprintf('The expected delta is: %d\n', base);
%     clear;
% end

function [delta, Delta] = findMatrixBounds(X, Y, n)
    % Initialize
    delta = 0;
    Delta = 0;
    spanM = []; % Matrix to hold vectorized monomials
    monomialList = {eye(n)}; % Start with the identity matrix

    % Add monomials of degree 1
    monomialList{end+1} = X;
    monomialList{end+1} = Y;

    % Vectorize and add to spanM
    for i = 1:length(monomialList)
        spanM(:, end+1) = monomialList{i}(:);
    end

    % Begin generating monomials of higher degrees
    currentDegree = 2;
    while delta == 0 || Delta == 0
        newMonomials = {};
        for monomial = monomialList
            mX = X * monomial{1};
            mY = Y * monomial{1};
            newMonomials{end+1} = mX;
            newMonomials{end+1} = mY;
            spanM(:, end+1) = mX(:);
            spanM(:, end+1) = mY(:);
        end

        % Check if the current set of monomials spans the space of n x n
matrices
        if rank(spanM) == n^2
            if delta == 0
                delta = currentDegree;
            end
            Delta = currentDegree; % Keep updating Delta until no new rank
increase
        end

        % Break if we've reached the maximum possible degree without a full
rank
```

```
        if currentDegree == n^2 && rank(spanM) < n^2
            delta = 0;
            Delta = 0;
            break;
        end

        % Prepare for the next iteration
        monomialList = newMonomials;
        currentDegree = currentDegree + 1;
    end
end
```



Single Case Base Code

function main n = 5; % Example for 3x3 matrices. Change n for different sizes X = randn(n); % Replace with your specific X matrix Y = randn(n); % Replace with your specific Y matrix

```
% Cyclic case
cyclic_matrix = zeros(n); % Initialize an n x n matrix of zeros

    % Fill the matrix
    for i = 1:n
        cyclic_matrix(i, :) = circshift(1:n, [0, i-1]);
    end

Y = cyclic_matrix;
```

```
    % Find bounds for the degrees
    [delta, Delta] = findMatrixBounds(X, Y, n);
    base = ceil(log2(n^2 + 1) - 1);

    % Display the results
    clc; close all;
    fprintf('The smallest degree delta is: %d\n', delta);
    fprintf('The largest degree Delta is: %d\n', Delta);
    fprintf('The expected delta is: %d\n', base);
    clear;
end
```

function [delta, Delta] = findMatrixBounds(X, Y, n) % Initialize delta = 0; Delta = 0; spanM = []; % Matrix to hold vectorized monomials monomialList = {eye(n)}; % Start with the identity matrix

```
    % Add monomials of degree 1
    monomialList{end+1} = X;
    monomialList{end+1} = Y;

    % Vectorize and add to spanM
    for i = 1:length(monomialList)
        spanM(:, end+1) = monomialList{i}(:);
    end

    % Begin generating monomials of higher degrees
    currentDegree = 2;
    while delta == 0 || Delta == 0
        newMonomials = {};
        for monomial = monomialList
            mX = X * monomial{1};
            mY = Y * monomial{1};
            newMonomials{end+1} = mX;
            newMonomials{end+1} = mY;
            spanM(:, end+1) = mX(:);
            spanM(:, end+1) = mY(:);
        end

        % Check if the current set of monomials spans the space of n x n matrices
        if rank(spanM) == n^2
            if delta == 0
                delta = currentDegree;
            end
            Delta = currentDegree; % Keep updating Delta until no new rank increase
        end

        % Break if we've reached the maximum possible degree without a full rank
        if currentDegree == n^2 && rank(spanM) < n^2
            delta = 0;
            Delta = 0;
            break;
        end

        % Prepare for the next iteration
        monomialList = newMonomials;
        currentDegree = currentDegree + 1;
```

```
    end
end
```

Pseudocode

Algorithm: Find Degree Bounds for Spanning Monomials Input: An integer n representing the size of the matrices, and matrices X and Y

Function main: Set n to the desired size of the matrices Generate a random n x n matrix X Generate a cyclic n x n matrix Y Find the bounds for the degrees delta and Delta that span the space of n x n matrices Calculate the base which is an expected lower bound for delta Display delta, Delta, and the expected delta

Function findMatrixBounds(X, Y, n): Initialize delta and Delta to 0 Initialize an empty list to hold vectorized monomials spanM Add the identity matrix to the list of monomials Add the matrices X and Y to the list of monomials Vectorize and add the initial monomials to spanM

```
Starting at degree 1, generate new monomials by multiplying each monomial by X and Y
For each new monomial:
    Vectorize and add it to spanM
    Check if spanM now has full rank (equal to n^2)
    If full rank is achieved:
        Set delta to the current degree if delta has not been set
        Update Delta to the current degree
        If Delta has been updated and is greater than delta, break out of the loop

If we reach monomials of degree n^2 without achieving full rank:
    Set delta and Delta to 0

Return delta and Delta
```

Function main end

Function findMatrixBounds end

Published with MATLAB® R2023b