MA1845_Laboration_fece23

MA1485 Laboration

Course: Linear Algebra (MA1485)

Student: Felix Cenusa (fece23)

Personal Number: 010711-3953

Date: 03-10-2024

Uppgift 1

Personal Info:

- Personal Number: 010711-3953
- Name: Felix Cenusa
- Acronym: fece23

Problem Formulation:

Solve the following system of linear equations using MATLAB:

[
x_1 + x_2 + 2x_3 + 2x_4 + 3x_5 + 3x_6 = 102
x_1 + 2x_2 + 3x_3 + x_4 + 2x_5 + 3x_6 = 122
2x_1 + 3x_2 + x_3 + 3x_4 + x_5 + 2x_6 = 140
2x_1 + 3x_2 + x_3 + x_4 + 3x_5 + 2x_6 = 158
2x_1 + 3x_2 + x_3 + 3x_4 + 2x_5 + x_6 = 150
3x_1 + x_2 + 2x_3 + 2x_4 + x_5 + 3x_6 = 120
]

Comments:

```
The problem can be solved manually and we have done that in class with for example
x y and z unknown parameters, but this exercise is made for us to learn to use
matlab a little so we use matlabs power to calculate everything efficiently.
```

Solution:

1. Represent the system in matrix form: (AX = B).
2. Use MATLAB to solve the system using the backslash operator (`\`).

```
% Felix Cenusa fece23

% 010711-3953

% Exercise 1 here:

A = [1 1 2 2 3 3;

1 2 3 1 2 3;

2 3 1 3 1 2;

2 3 1 1 3 2;

2 3 1 3 2 1;

3 1 2 2 1 3];

B = [102; 122; 140; 158; 150; 120];

X = A \ B

% check solution is right:

sameBHopefully = A * X
```

Result by running the code:

```
```Running the code gave me this:
X =
```
```

```
   20.0000
   24.0000
    9.0000
    2.0000
   11.0000
    1.0000


sameBHopefully =


  102.0000
  122.0000
  140.0000
  158.0000
  150.0000
  120.0000
```

Uppgift 2

Personal Info:

- Personal Number: 010711-3953
- Name: Felix Cenusa
- Acronym: fece23

Problem Formulation:

In this task, we are asked to perform a reflection transformation using a matrix (S), where the vector (V) is defined based on my personal parameters:

- (alpha = p4 + 1)
- (beta = p5 + 1)
- (gamma = p6 + 1)

We need to:

1. Create a 3x1 vector (V) where:
   V = \alpha , beta , gamma
2. Define the matrix (S) using the formula:
   S = I - (2 / V_T_V) *(V* V_T)
   (I) is a 3x3 identity matrix.
3. Choose 3 arbitrary vectors, plot them in blue, apply the transformation matrix (S), and plot the transformed vectors in green.
4. Compute the eig of matrix (S), and plot the eig vectors in red.
5. Finally, we'll determine the reflection plane based on the vector (V) and draw it on the plot.

Comments:

The matrix (S) represents a **reflection**, which reflects vectors across a plane perpendicular to (V). This exercise is about understanding how matrix transformations work and learning how to visualize these transformations in MATLAB using eig quiver3 and surf for the plane.

Solution:

1. First, I calculated my personal parameters and used them to define the vector (V).
2. Using (V), I constructed the matrix (S) that represents the reflection transformation.
3. I applied this transformation to three arbitrary vectors and visualized both the original vectors and their transformed versions.
4. Then, I computed the eig of the matrix (S) and plotted the eig vectors to show how the matrix (S) acts on these directions.
5. Lastly, I visualized the reflection plane that's perpendicular to (V).

```
% Felix Cenusa fece23

% 010711-3953

% Exercise 2 here:

% α = p4 + 1 β = p5 + 1 γ = p6 + 1

% meaning that

p4 = 7

p5 = 1
```

```
p6 = 1

% so now we set alpha beta and theta to the corresponding numbers

alpha = p4 + 1

beta = p5 + 1

theta = p6 + 1

% apparently αβ or γ are not valid names for variables.

V = [alpha;beta;theta;]

V_T = V'

V_T_V = V_T * V % dot product here

% now we define the identity matrix size:

I = eye(3); %3x3 size using eye.

%now we define S in terms of what was defined in the assignment:

S = I - (2 / V_T_V) * (V * V_T) %S matrix here.

% Paranthetis not needed but make it clearer.

% now we define 3-5(3) collumn vectors, i will put them in "v_x"

v_1 = [1; 0; 0];

v_2 = [0; 1; 0];

v_3 = [0; 0; 1];

%transform the vectros

t_v_1 = S * v_1;

t_v_2 = S * v_2;

t_v_3 = S * v_3;

% show the origianl vectors in blue:
```

```
figure;

quiver3(0, 0, 0, v_1(1), v_1(2), v_1(3), "b");

hold on;% so it stays when something new is rendered (green incomming)

quiver3(0, 0, 0, v_2(1), v_2(2), v_2(3), "b");

quiver3(0, 0, 0, v_3(1), v_3(2), v_3(3), "b"); % the original 3 vectors

% show the transformed vectors in green

quiver3(0, 0, 0, t_v_1(1), t_v_1(2), t_v_1(3), "g");

quiver3(0, 0, 0, t_v_2(1), t_v_2(2), t_v_2(3), "g");

quiver3(0, 0, 0, t_v_3(1), t_v_3(2), t_v_3(3), "g"); % transformed vectors

[V_eig, D_eig] = eig(S); % v_eig is eig of s, d_eig is diagonal matrix eig.

quiver3(0, 0, 0, V_eig(1,1), V_eig(2,1), V_eig(3,1), "r"); % Plot each eig in red

quiver3(0, 0, 0, V_eig(1,2), V_eig(2,2), V_eig(3,2), "r"); % Plot each eigenvector
in red

quiver3(0, 0, 0, V_eig(1,3), V_eig(2,3), V_eig(3,3), "r"); % Plot each eigenvector
in red

% The matrix S represents a reflection, reflecting vectors

% across the plane orthogonal to the vector V

% Define a grid to represent the plane orthogonal to V

[x_grid, y_grid] = meshgrid(-1:0.1:1, -1:0.1:1);

z_grid = -(V(1) * x_grid + V(2) * y_grid) / V(3); % Equation of the plane

% Plot the plane as a surface

surf(x_grid, y_grid, z_grid, "FaceAlpha", 0.1);
hold off;
```

Result by running the code:

```
>> MA1485_Uppgift2_fece23

p4 =

     7


p5 =

     1


p6 =

     1


alpha =

     8


beta =

     2


theta =

     2


V =

     8
     2
     2


V_T =

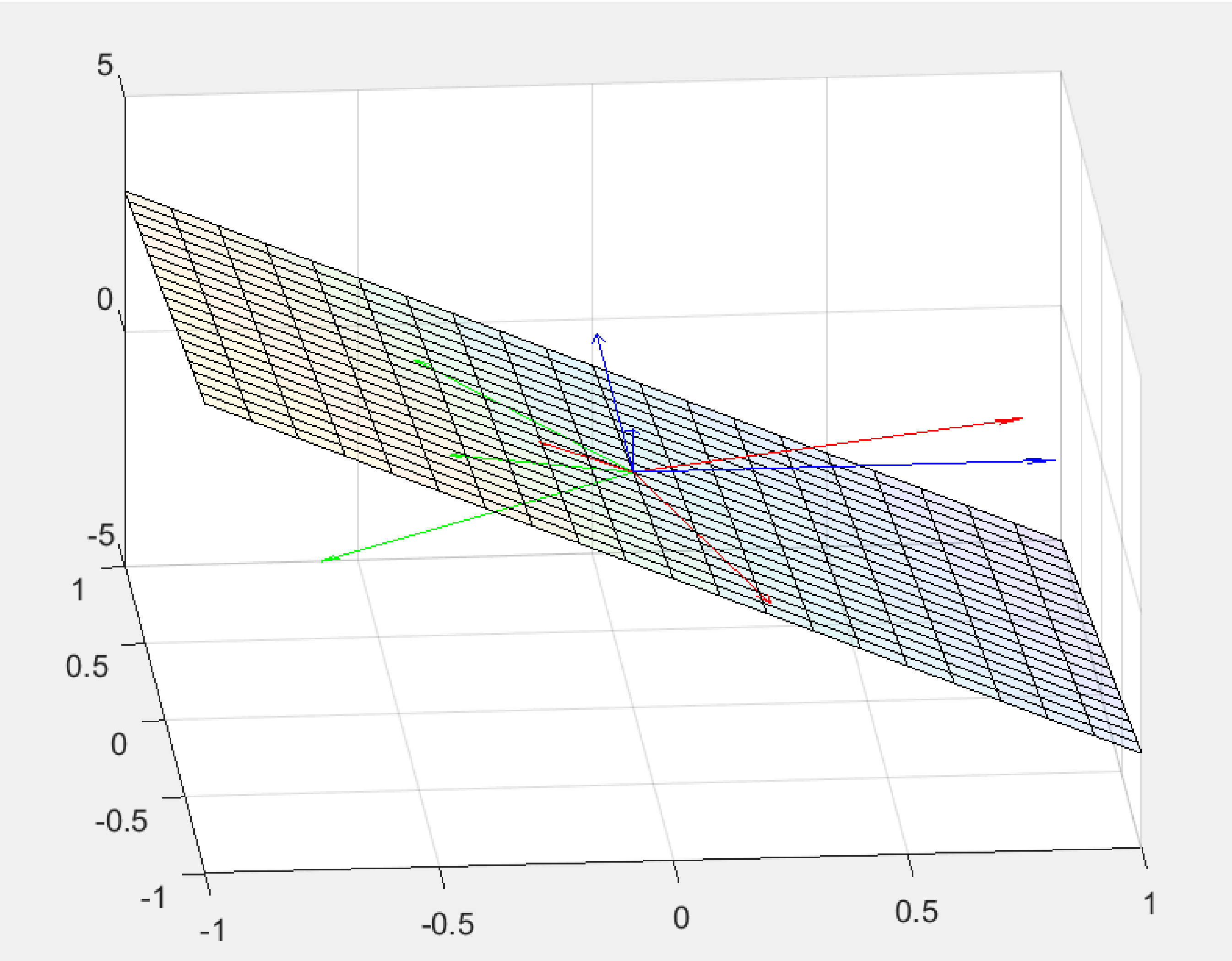
     8     2     2
```

```
V_T_V =

    72


S =

   -0.7778   -0.4444   -0.4444
   -0.4444    0.8889   -0.1111
   -0.4444   -0.1111    0.8889
```



Uppgift 3

Personal Info:

- Personal Number: 010711-3953
- Name: Felix Cenusa
- Acronym: fece23

Problem Formulation:

In this task, we need to project a 4-dimensional hypercube onto a 3-dimensional space (a hyperplane where the fourth dimension is zero). The hypercube has 16 vertices, which are all the possible combinations of (±1, ±1, ±1, ±1).
The projection will be done along a vector U, which is based on my personal number:

- U = (p4 + 1, p5 + 1, p6 + 1, p7 + 1)
- In my case, U=(8,2,2,4)

The steps are as follows:

1. Generate the 16 vertices of the 4D hypercube.
2. Project these vertices onto 3D space along the vector U.
3. Plot the 3D coordinates of the projected vertices.
4. Connect nearby vertices with edges to form the 3D projection of the hypercube.
5. Ensure that the resulting figure correctly represents the projected hypercube.
6. Force matlab to display this in 3D.

Comments:

Cool personal 3d cube. The main goal is to understand how objects from higher dimensions can be projected into lower dimensions and visualized in MATLAB.

Solution:

1. I generated the 16 vertices of the 4D hypercube.
2. I defined my personal projection vector U = (8, 2, 2, 4).
3. I projected the 4D vertices onto 3D by moving each point along the direction of U until the fourth coordinate became zero.
4. I plotted the 3D coordinates and connected adjacent vertices with edges to visualize the hypercube in 3D.

```
% Felix Cenusa fece23

% 010711-3953

% Exercise 3 here:

p4 = 7;

p5 = 1;

p6 = 1;

p7 = 3;

U = [p4+1,p5+1,p6+1,p7+1]

% we use ngrid to generate the combinations plusminus 1 of the 4

% dimentional cube

[x1, x2, x3, x4] = ndgrid([-1, 1], [-1, 1], [-1, 1], [-1, 1]);

vertices = [x1(:), x2(:), x3(:), x4(:)] % Converting the grid to a list of 16x4
vertices,

% each row is a vertex

% each collumn is the xyzw coordinates of the vertex i hope

% now we project the verticies from 4d to 3d

projected_to_3d_vertices = zeros(16, 3); % To store the projected 3d points

for i = 1:16

v = vertices(i, :);

% Calculate how far along vector U to project the point so that the 4th coordinate
becomes 0

% if this was 3d to 2d, the vector U(2d) is the direction / path that

% the points need to move in to reach 2d.

t = v(4) / U(4);
```

```
% Project onto the 3D space by subtracting t * U

projected_to_3d_vertices(i, :) = v(1:3) - t * U(1:3);

end

% Plot the projected 3D hypercube

figure;

hold on;

view(3) % need this to force matlab to show result in 3d

axis equal; % annoying when it snaps to a different size so turned it off.

grid on; % easier to see size and distance

scatter3(projected_to_3d_vertices(:,1), projected_to_3d_vertices(:,2),
projected_to_3d_vertices(:,3), 'filled');

% Connect the vertices with edges

for i = 1:16

for j = i+1:16

% Check if vertices differ by exactly one coordinate (one edge)

if sum(abs(vertices(i, :) - vertices(j, :))) == 2

% Draw an edge between these two vertices

plot3([projected_to_3d_vertices(i,1), projected_to_3d_vertices(j,1)], ...

[projected_to_3d_vertices(i,2), projected_to_3d_vertices(j,2)], ...

[projected_to_3d_vertices(i,3), projected_to_3d_vertices(j,3)], 'k-');

end

end

end

hold off;
```

Result by running the code:

```
>> MA1485_Uppgift3_fece23

U =

     8     2     2     4


vertices =

    -1    -1    -1    -1
     1    -1    -1    -1
    -1     1    -1    -1
     1     1    -1    -1
    -1    -1     1    -1
     1    -1     1    -1
    -1     1     1    -1
     1     1     1    -1
    -1    -1    -1     1
     1    -1    -1     1
    -1     1    -1     1
     1     1    -1     1
    -1    -1     1     1
     1    -1     1     1
    -1     1     1     1
     1     1     1     1
```

Pretty cube: