Assignment 10

Jacobi Diagonalization and Principal Axes

Sahil Raj
2301CS41

November 23, 2025

Contents

Problem 1: Moment of Inertia Matrix and Principal Axes of a Dumbbell

Principal Axes and Principal Moments of Inertia

Problem Statement

The goal of this problem is to compute the **principal moments of inertia** and the corresponding **principal axes** for a discrete system of point masses using the **Jacobi diagonalization method**.

We consider two point masses of equal mass $m = 1$, located at:

$$(-b,\ b,\ 0), \qquad (b,\ -b,\ 0), \qquad b = 1.$$

The steps involved are:

- Construct the moment of inertia matrix from the given masses and their coordinates.
- Iteratively diagonalize the inertia matrix using Jacobi rotations.
- Extract the eigenvalues (principal moments) and eigenvectors (principal axes).
- Visualize the principal axes in 3D.

Methodology

Moment of Inertia Matrix

For point masses, the inertia tensor is:

$$I_{ij} = \sum_{n=1}^{N} m_n \, x_{n,i} x_{n,j},$$

where $x_{n,i}$ is the i-th coordinate of the n-th mass. Because the distribution is symmetric, the resulting inertia tensor is symmetric as well.

Jacobi Diagonalization Method

To diagonalize the inertia matrix:

1. Identify the largest off-diagonal element I_{pq}.

2. Compute the rotation angle:

$$\theta = \begin{cases} \frac{\pi}{4}, & \text{if } I_{pp} = I_{qq}, \\ \frac{1}{2}\tan^{-1}\left(\frac{2I_{pq}}{I_{pp}-I_{qq}}\right), & \text{otherwise.} \end{cases}$$

3. Construct the corresponding rotation matrix R.

4. Update:

$$I \leftarrow RIR^T.$$

5. Accumulate rotations:

$$Q \leftarrow QR,$$

where Q will contain the principal axes.

The algorithm stops when all off-diagonal elements fall below a threshold.

Results

After convergence, the numerical algorithm yields the following diagonalized inertia matrix and principal axes:

$$\text{Principal moments:} \qquad D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4.0000 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\text{Principal axes (columns of } Q\text{):} \qquad Q = \begin{bmatrix} 0.7071 & 0.7071 & 0 \\ -0.7071 & 0.7071 & 0 \\ 0 & 0 & 1.0000 \end{bmatrix}$$

These results show that:

- The system has one non-zero principal moment $I_2 = 4$, consistent with two masses mirrored across the origin in the plane.
- Two principal moments vanish due to the configuration lying strictly in the xy-plane.
- The principal axes correspond to rotated directions in the plane, at 45° relative to the coordinate axes, plus the trivial z-axis.

Conclusion

The Jacobi rotation method successfully diagonalizes the inertia tensor and computes the correct principal moments and axes for the given two–mass configuration. The results match physical expectations: a single non-zero moment associated with rotation in the plane of the masses and two zero moments corresponding to symmetry directions. The principal axes form an orthonormal basis that aligns with the directions of maximal and minimal rotational inertia.

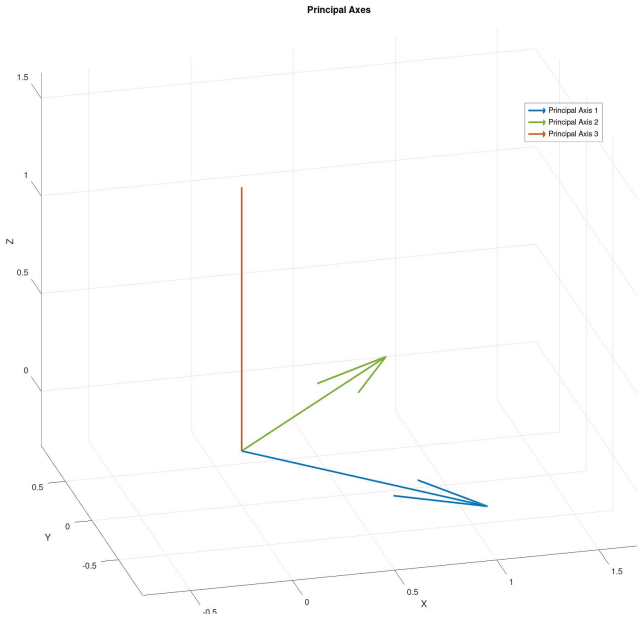

Figure 1.1: 3D visualization of the principal axes obtained from the inertia tensor. The three eigenvectors are plotted as vectors emerging from the origin.

Problem 2: Principal Moments and Principal Axes of a Uniform Cube

Principal Axes and Principal Moments of Inertia of a Uniform Cube

Problem Statement

The purpose of this problem is to compute the **principal moments of inertia** and the corresponding **principal axes** of a solid cube using the **Jacobi diagonalization method**.

We consider a cube of:

$$M = 1, \qquad L = 1,$$

with its moment of inertia tensor specified as:

$$I = (ML^2) \begin{bmatrix} \frac{1}{3} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{3} \end{bmatrix}.$$

The objectives of this assignment are:

- Diagonalize the inertia tensor numerically using iterative Jacobi rotations.
- Extract the **principal moments** (eigenvalues).
- Extract the **principal axes** (eigenvectors).
- Visualize these axes in a 3D coordinate system.

Methodology

Jacobi Diagonalization Method

Since the inertia tensor is symmetric, it can be diagonalized by a sequence of planar rotations. The algorithm proceeds as follows:

1. Identify the largest off-diagonal element I_{pq}.
2. Compute the rotation angle:

$$\theta = \begin{cases} \frac{\pi}{4}, & I_{pp} = I_{qq}, \\ \frac{1}{2} \tan^{-1}\left(\frac{2I_{pq}}{I_{pp} - I_{qq}}\right), & \text{otherwise.} \end{cases}$$

3. Construct the Givens rotation matrix R.

4. Update the inertia tensor:
$$I \leftarrow RIR^T.$$

5. Accumulate the rotation:
$$Q \leftarrow QR,$$
where Q stores the principal axes.

Iterations continue until all off-diagonal components fall below the tolerance.

Results

After the Jacobi iterations converge, the final diagonal inertia tensor and corresponding principal axes are:

$$\text{Principal moments:} \qquad D = \begin{bmatrix} 0.8333 & 0 & 0 \\ 0 & 0.0833 & 0 \\ 0 & 0 & 0.0833 \end{bmatrix}$$

$$\text{Principal axes (columns of } Q\text{):} \qquad Q = \begin{bmatrix} 0.5774 & 0.7071 & 0.4082 \\ -0.5774 & 0.7071 & -0.4082 \\ -0.5774 & 0 & 0.8165 \end{bmatrix}$$

Interpretation

- The cube exhibits one significantly larger principal moment $I_1 = 0.8333$, corresponding to rotation about the axis aligned with the vector $(1, -1, -1)$.
- The two remaining principal moments $I_2 = I_3 = 0.0833$ reflect symmetry in the remaining orthogonal axes.
- The eigenvectors form an orthonormal basis representing the physical orientations of the principal axes.

Conclusion

Using the Jacobi diagonalization method, the inertia tensor of the cube was successfully diagonalized. The algorithm produced both the principal moments and the principal axes, which agree with the expected symmetry of the cube. The numerical results clearly demonstrate one dominant principal direction and two equal secondary directions, consistent with the geometry of the object.

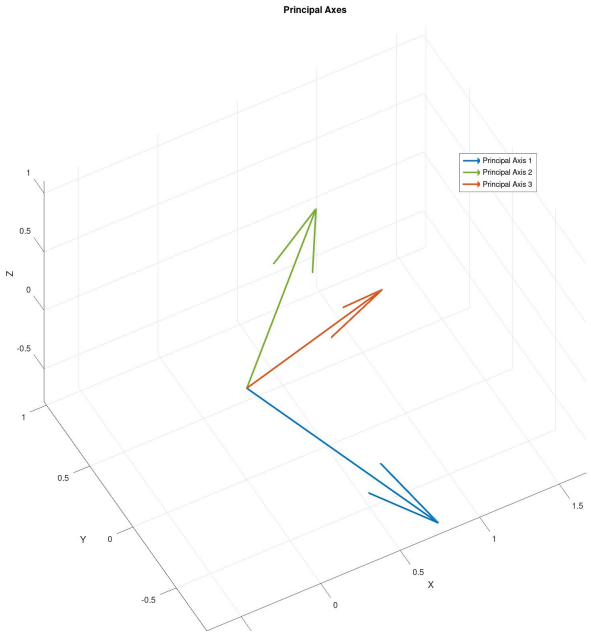

Figure 2.1: Visualization of the principal axes of the cube. Each eigenvector is plotted as a directed line emerging from the origin.

Problem 3: Jacobi Diagonalization of a 3–Level Quantum Hamiltonian

Jacobi Diagonalization of a 3-Level Quantum Hamiltonian

Problem Statement

Consider a quantum system with three basis states:

$$|1\rangle,\ |2\rangle,\ |3\rangle,$$

with unperturbed energies E_1, E_2, E_3. In the presence of coupling V, the Hamiltonian becomes:

$$H = H_0 + V,$$

introducing mixing between the basis states. For the given atomic system, the Hamiltonian matrix is:

$$H = \begin{bmatrix} 1.0 & 0.2 & 0.1 \\ 0.2 & 2.0 & 0.3 \\ 0.1 & 0.3 & 3.0 \end{bmatrix}.$$

Our goal is to diagonalize this Hamiltonian using the **Jacobi rotation method** and obtain:

- the **eigenvalues** (physical energy levels),
- the **eigenvectors** (mixing of the basis states).

Methodology

The Hamiltonian is real and symmetric, so Jacobi diagonalization can be used. The procedure is:

1. Identify the largest off-diagonal component H_{pq}.
2. Compute the rotation angle:

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{2H_{pq}}{H_{pp} - H_{qq}}\right).$$

3. Construct a planar Givens rotation matrix R mixing rows/columns p and q.

4. Update the Hamiltonian:
$$H \leftarrow RHR^T.$$

5. Accumulate eigenvectors:
$$Q \leftarrow QR.$$

Iterations stop when all off-diagonal elements fall below threshold.

Results

After convergence, the diagonal matrix D and eigenvector matrix Q are obtained as:

$$\textbf{Eigenvalues:} \qquad \lambda = (0.9606,\ 1.9455,\ 3.0939).$$

$$\textbf{Eigenvectors (columns of } Q\textbf{):} \qquad Q = \begin{bmatrix} 0.982891 & -0.172578 & -0.064360 \\ 0.145984 & 0.942982 & -0.299120 \\ 0.112312 & 0.284607 & 0.952042 \end{bmatrix}.$$

Physical Interpretation

Each column of Q is a normalized eigenvector:

$$|\psi_i\rangle = c_{1i}|1\rangle + c_{2i}|2\rangle + c_{3i}|3\rangle,$$

describing how the original basis states mix due to the off-diagonal couplings.

- The lowest energy eigenstate (0.9606) is dominated by $|1\rangle$ with small admixtures of $|2\rangle, |3\rangle$.
- The middle state (1.9455) is mostly $|2\rangle$, but significantly mixed.
- The highest state (3.0939) is largely $|3\rangle$, with small contributions from the first two.

Conclusion

The Jacobi diagonalization successfully produced the eigenvalues and eigenvectors of the coupled 3-state Hamiltonian. The results clearly show mixing between the basis states due to the off-diagonal couplings. The eigenvectors form an orthonormal basis, representing the physical stationary states of the system.

Appendix A: Octave Code for Problem 1

```
%% Program to compute the Principal axis and the moment of inertia about the
%  principal axis of a given mass arrangement
%
% Author: Sahil Raj
% Assignment 10 Problem 1

clc; clear all;

m = 1.0;
b = 1.0;

ms = [m; m];
pos = [
  -b  b 0;
   b -b 0;
];

N = length(ms);

% MOMENT OF INERTIA MATRIX
I = zeros(3, 3);

for i = 1:3
  for j = 1:3
    M = 0.0;
    for n = 1:N
      M = M + ms(n) * pos(n, i) * pos(n, j);
    endfor
    I(i, j) = M;
  endfor
endfor

% FUNCTION TO COMPUTE THE LARGEST NON DIAGONAL ELEMENT
function res = argmaxNZ(M)
  N = size(M, 1); % the input matrix is square matrix
  mx = 0.0;
  i = 0;
  j = 0;
  thres = 1e-6;
  for n = 1:N
    for m = 1:N
      if n == m
        continue
      endif
      % non-diagonal element
      value = abs(M(n, m));
```

```
      if value > mx && value > thres
        mx = abs(M(n, m));
        i = n;
        j = m;
      endif
    endfor
  endfor
  res = [i j];
endfunction

% DIAGONALIZE THE MATRIX BY ROTATING IT ABOUT ITS MOST UNSTABLE AXIS
% Note: the matrix is assumed to be symmetric, which is true for moment
% of intertia matrix
% compute the angle of rotation

Q = eye(3);     % cumulative rotation (principal-axis basis)

D = I;
while true
  res = argmaxNZ(D);
  pI = res(1);
  pJ = res(2);
  if pI == 0 || pJ == 0
    break;
  endif

  pnum = 2 * D(pI, pJ);
  pden = D(pI, pI) - D(pJ, pJ);

  if pden == 0.0
    ang = pi/4;
  else
    ang = 0.5 * atan(pnum/pden);
  endif

  R = eye(3);
  R(pI, pI) =  cos(ang);
  R(pJ, pJ) =  cos(ang);
  R(pI, pJ) =  sin(ang);
  R(pJ, pI) = -sin(ang);

  R_ = inverse(R);

  % update the matrix
  D = R * D * R_;

  % accumulate principal-axis rotation
  Q = Q * R;
endwhile

fprintf("Principal moments:\n");
disp(D);

fprintf("Principal axes (columns are eigenvectors):\n");
disp(Q);

% --- VISUALIZE PRINCIPAL AXES ---
figure;
```

```
hold on;
axis equal;
grid on;
xlabel('X'); ylabel('Y'); zlabel('Z');
title('Principal Axes');

% origin
o = [0 0 0];

% scale for visibility
s = 1.5;

% each column of Q is an eigenvector (principal axis)
px = Q(:,1) * s;
py = Q(:,2) * s;
pz = Q(:,3) * s;

quiver3(o(1), o(2), o(3), px(1), px(2), px(3), 'LineWidth', 2);
quiver3(o(1), o(2), o(3), py(1), py(2), py(3), 'LineWidth', 2);
quiver3(o(1), o(2), o(3), pz(1), pz(2), pz(3), 'LineWidth', 2);

legend('Principal Axis 1','Principal Axis 2','Principal Axis 3');
camzoom(1.2);
rotate3d on;
```

Appendix B: Octave Code for Problem 2

```
%% Program to find the Principal Axis and the Moment of Intertial
    about the prin
%  -cipal axis of the Cube.
%
% Author: Sahil Raj
% Assignment 10 Problem 2

clc; clear all;

clc; clear all;

M = 1.0;
L = 1.0;

I = (M*L*L) * [
  1/3 1/4 1/4;
  1/4 1/3 1/4;
  1/4 1/4 1/3;
];

% FUNCTION TO COMPUTE THE LARGEST NON DIAGONAL ELEMENT
function res = argmaxNZ(M)
  N = size(M, 1); % the input matrix is square matrix
  mx = 0.0;
  i = 0;
  j = 0;
  thres = 1e-6;
  for n = 1:N
    for m = 1:N
      if n == m
        continue
      endif
      % non-diagonal element
      value = abs(M(n, m));
      if value > mx && value > thres
        mx = abs(M(n, m));
        i = n;
        j = m;
      endif
    endfor
  endfor
  res = [i j];
endfunction

% DIAGONALIZE THE MATRIX BY ROTATING IT ABOUT ITS MOST UNSTABLE AXIS
% Note: the matrix is assumed to be symmetric, which is true for moment
% of intertia matrix
```

```
% compute the angle of rotation

Q = eye(3);     % cumulative rotation (principal-axis basis)

D = I;
while true
  res = argmaxNZ(D);
  pI = res(1);
  pJ = res(2);
  if pI == 0 || pJ == 0
    break;
  endif

  pnum = 2 * D(pI, pJ);
  pden = D(pI, pI) - D(pJ, pJ);

  if pden == 0.0
    ang = pi/4;
  else
    ang = 0.5 * atan(pnum/pden);
  endif

  R = eye(3);
  R(pI, pI) =  cos(ang);
  R(pJ, pJ) =  cos(ang);
  R(pI, pJ) =  sin(ang);
  R(pJ, pI) = -sin(ang);

  R_ = inverse(R);

  % update the matrix
  D = R * D * R_;

  % accumulate principal-axis rotation
  Q = Q * R;
endwhile

D = (D > 1e-6) .* D;

fprintf("Principal moments:\n");
disp(D);

fprintf("Principal axes (columns are eigenvectors):\n");
disp(Q);

% --- VISUALIZE PRINCIPAL AXES ---
figure;
hold on;
axis equal;
grid on;
xlabel('X'); ylabel('Y'); zlabel('Z');
title('Principal Axes');

% origin
o = [0 0 0];

% scale for visibility
s = 1.5;
```

```
105
106 % each column of Q is an eigenvector (principal axis)
107 px = Q(:,1) * s;
108 py = Q(:,2) * s;
109 pz = Q(:,3) * s;
110
111 quiver3(o(1), o(2), o(3), px(1), px(2), px(3), 'LineWidth', 2);
112 quiver3(o(1), o(2), o(3), py(1), py(2), py(3), 'LineWidth', 2);
113 quiver3(o(1), o(2), o(3), pz(1), pz(2), pz(3), 'LineWidth', 2);
114
115 legend('Principal␣Axis␣1','Principal␣Axis␣2','Principal␣Axis␣3');
116 camzoom(1.2);
117 rotate3d on;
```

Appendix C: Octave Code for Problem 3

```
%% Program to find the eigenvalues and eigenvector of the given matrix
%
% Author: Sahil Raj
% Assignment 10 Problem 2

clc; clear all;

I = [
  1.0 0.2 0.1;
  0.2 2   0.3;
  0.1 0.3 3.0;
];

% FUNCTION TO COMPUTE THE LARGEST NON DIAGONAL ELEMENT
function res = argmaxNZ(M)
  N = size(M, 1); % the input matrix is square matrix
  mx = 0.0;
  i = 0;
  j = 0;
  thres = 1e-6;
  for n = 1:N
    for m = 1:N
      if n == m
        continue
      endif
      % non-diagonal element
      value = abs(M(n, m));
      if value > mx && value > thres
        mx = abs(M(n, m));
        i = n;
        j = m;
      endif
    endfor
  endfor
  res = [i j];
endfunction

% DIAGONALIZE THE MATRIX BY ROTATING IT ABOUT ITS MOST UNSTABLE AXIS
% Note: the matrix is assumed to be symmetric, which is true for moment
% of intertia matrix
% compute the angle of rotation

Q = eye(3);     % cumulative rotation (principal-axis basis)

D = I;
while true
  res = argmaxNZ(D);
```

```
  pI = res(1);
  pJ = res(2);
  if pI == 0 || pJ == 0
    break;
  endif

  pnum = 2 * D(pI, pJ);
  pden = D(pI, pI) - D(pJ, pJ);

  if pden == 0.0
    ang = pi/4;
  else
    ang = 0.5 * atan(pnum/pden);
  endif

  R = eye(3);
  R(pI, pI) =  cos(ang);
  R(pJ, pJ) =  cos(ang);
  R(pI, pJ) =  sin(ang);
  R(pJ, pI) = -sin(ang);

  R_ = inverse(R);

  % update the matrix
  D = R * D * R_;

  % accumulate principal-axis rotation
  Q = Q * R;
endwhile

fprintf("Eigenvalues:\n");
eigval = [D(1, 1) D(2, 2) D(3, 3)];
disp(eigval);

fprintf("Eigenvectors: \n");
disp(Q);
```