BRYN MAWR COLLEGE

Department of Physics

Computational Module 7 – Eigenequations

Prerequisite modules: Module 6 – Linear Equations
Estimated completion time: 5-8 hours
Learning objectives: Learn how eigenvectors and eigenvalues are found numerically, and explore some scientifically relevant applications
Note: This module requires some familiarity with linear algebra.

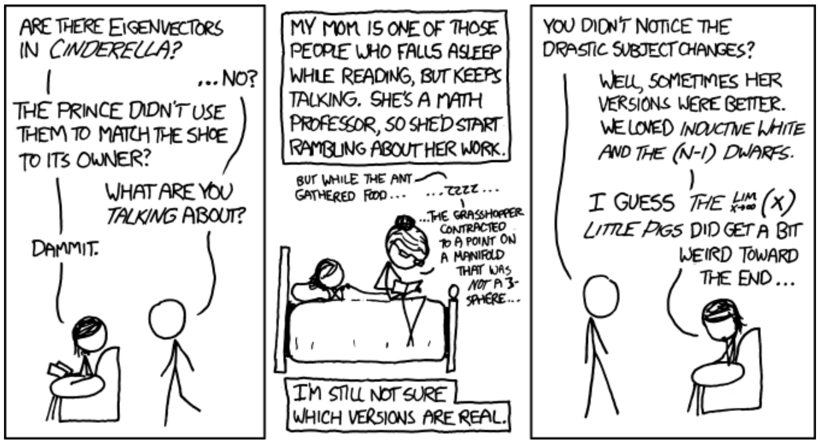

ARE THERE EIGENVECTORS IN CINDERELLA?
...NO?
THE PRINCE DIDN'T USE THEM TO MATCH THE SHOE TO ITS OWNER?
WHAT ARE YOU TALKING ABOUT?
DAMMIT.
MY MOM IS ONE OF THOSE PEOPLE WHO FALLS ASLEEP WHILE READING, BUT KEEPS TALKING. SHE'S A MATH PROFESSOR, SO SHE'D START RAMBLING ABOUT HER WORK.
BUT WHILE THE ANT GATHERED FOOD...
...ZZZZ...
...THE GRASSHOPPER CONTRACTED TO A POINT ON A MANIFOLD THAT WAS NOT A 3-SPHERE...
I'M STILL NOT SURE WHICH VERSIONS ARE REAL.
YOU DIDN'T NOTICE THE DRASTIC SUBJECT CHANGES?
WELL, SOMETIMES HER VERSIONS WERE BETTER. WE LOVED INDUCTIVE WHITE AND THE (N-1) DWARFS.
I GUESS THE LIM x→∞ (x) LITTLE PIGS DID GET A BIT WEIRD TOWARD THE END...

Image credit: xkcd.com
See http://www.explainxkcd.com/wiki/index.php/File:fairy_tales.png for an explanation.

This module[1] can be thought of as the second of a pair of modules exploring computational matrix methods, Module 6 being the first in the series. Here, we explore numerical methods for determining eigenvalues and eigenvectors of matrices. This ability is useful in classical mechanics for the determination of principal axes of rotation and their corresponding moments of inertia, and for understanding the motions of coupled oscillators; in quantum mechanics for the identification of the energy values and states of a system; in engineering, for predicting oscillatory motion of structures (e.g., in reaction to earthquakes) and for the design of MEMS (microelectromechanical systems) devices; and even in biology, for the study of protein conformation.

[1]Module adapted in part from *Computational Physics* by Mark Newman.

7.1 Eigenvalues and Eigenvectors

The general problem of interest in this module has the form

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}, \tag{1}$$

where $\mathbf{A}$ is a matrix, the vectors $\mathbf{v}$ that solve the equation are called ***eigenvectors***, and the corresponding scalar λ values are ***eigenvalues***. The eigenvectors represent vectors whose directions are not changed by the action of the matrix. (The eigenvalues indicate how much the magnitudes are changed.) An $N \times N$ matrix $\mathbf{A}$ will have N eigenvectors, which will be labeled $\mathbf{v}_i$, and N corresponding eigenvalues λ_i (some of which may be the same; i.e., “degenerate”).

It will turn out to be useful to define a new matrix $\mathbf{V}$ whose columns are the eigenvectors ($\mathbf{v}_1$ is the first column, $\mathbf{v}_2$ is the second column, etc.), and also turn the eigenvalues into a *diagonal* matrix $\mathbf{\Lambda}$ (so $\mathbf{\Lambda}_{11} = \lambda_1$, $\mathbf{\Lambda}_{22} = \lambda_2$, etc., and all non-diagonal elements of $\mathbf{\Lambda}$ are zero). By writing out the multiplications indicated in Eq. (1), it's not too hard to show that in terms of these new matrices the original problem can be expressed as

$$\mathbf{A}\mathbf{V} = \mathbf{V}\mathbf{\Lambda}. \tag{2}$$

The usual approach to solving Eq. (1) involves rewriting the equation as $(\mathbf{A} - \lambda\mathbf{I})\mathbf{v} = \mathbf{0}$, where $\mathbf{I}$ is the $N \times N$ unit matrix and $\mathbf{0}$ is an N-element vector of zeros. A nontrivial solution can be found only if $|\mathbf{A} - \lambda\mathbf{I}| = \mathbf{0}$. Imposing that requirement leads to an algebraic ***characteristic equation*** in terms of the eigenvalues. Once those are found, the eigenvectors then can be found by substitution of the eigenvalues back into Eq. (1).

As we've seen in other situations, the usual analytical approach in this case is not an efficient one for finding a numerical solution to the problem. One category of efficient numerical approaches is based on factorization of the matrix $\mathbf{A}$. A common factorization method for finding eigenvalues which we will explore here — specifically for the case of a *real*, *symmetric* $\mathbf{A}$ (whose eigenvalues are real) — is the ***QR algorithm*** mentioned in Module 6. This involves writing $\mathbf{A}$ as a product of an ***orthogonal*** matrix $\mathbf{Q}$ (obeying $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}^{-1}\mathbf{Q} = \mathbf{I}$, where $\mathbf{Q}^T$ is the transpose of $\mathbf{Q}$ and $\mathbf{I}$ is the identity matrix), and an upper-triangular matrix $\mathbf{R}$, in what is termed a ***QR decomposition***, which always is possible for a square matrix:

$$\mathbf{A} = \mathbf{Q}_1\mathbf{R}_1. \tag{3}$$

If we multiply on the left by $\mathbf{Q}_1^T$, then by orthogonality we get

$$\mathbf{Q}_1^T\mathbf{A} = \mathbf{Q}_1^T\mathbf{Q}_1\mathbf{R}_1 = \mathbf{I}\mathbf{R}_1 = \mathbf{R}_1. \tag{4}$$

We now define a new matrix $\mathbf{A}_1$ as the reverse product of the two matrices used to factorize $\mathbf{A}$:

$$\mathbf{A}_1 = \mathbf{R}_1\mathbf{Q}_1. \tag{5}$$

Using Eq. (4), we can rewrite this as

$$\mathbf{A}_1 = \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_1. \tag{6}$$

This process is now repeated, by constructing the QR decomposition of $\mathbf{A}_1$ in terms of a new orthogonal matrix $\mathbf{Q}_2$ and new upper-triangular matrix $\mathbf{R}_2$:

$$\mathbf{A}_1 = \mathbf{Q}_2\mathbf{R}_2. \tag{7}$$

Again define a new matrix as the reverse product of the new matrices,

$$\mathbf{A}_2 = \mathbf{R}_2\mathbf{Q}_2 = \mathbf{Q}_2^T\mathbf{A}_1\mathbf{Q}_2, \tag{8}$$

where in the last step we used Eq. (7) left-multiplied by $\mathbf{Q}_2^T$ $(= \mathbf{Q}_2^{-1})$. Expanding $\mathbf{A}_1$ using Eq. (6), obtain

$$\mathbf{A}_2 = \mathbf{Q}_2^T\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_1\mathbf{Q}_2. \tag{9}$$

Repeating the process further, we can construct a sequence of matrices:

$$\begin{aligned}
\mathbf{A}_1 &= \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_1 \\
\mathbf{A}_2 &= \mathbf{Q}_2^T\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_1\mathbf{Q}_2 \\
&\vdots \\
\mathbf{A}_n &= \left(\mathbf{Q}_n^T\ldots\mathbf{Q}_2^T\mathbf{Q}_1^T\right)\mathbf{A}\left(\mathbf{Q}_1\mathbf{Q}_2\ldots\mathbf{Q}_n\right).
\end{aligned} \tag{10}$$

What's the point of all of this? Well, it turns out that if this process is continued to large enough n, then $\mathbf{A}_n$ will be diagonal, or nearly so; i.e., the off-diagonal elements can be made as small as desired. Thus, we can write $\mathbf{A}_n \simeq \mathbf{D}$, with $\mathbf{D}$ diagonal. To make use of this fact, we define a new matrix

$$\mathbf{V} = \mathbf{Q}_1\mathbf{Q}_2\ldots\mathbf{Q}_n = \prod_{i=1}^{k}\mathbf{Q}_i, \tag{11}$$

which itself will be an orthogonal matrix since all of the $\mathbf{Q}_i$ are orthogonal. In terms of this matrix, Eq. (10) becomes

$$\mathbf{D} \simeq \mathbf{A}_n = \mathbf{V}^T\mathbf{A}\mathbf{V}. \tag{12}$$

Finally, multiplying on the left by $(\mathbf{V}^T)^{-1} = \mathbf{V}$ and exchanging the two sides of the equation gives

$$\mathbf{A}\mathbf{V} = \mathbf{V}\mathbf{D}. \tag{13}$$

This is identical to the matrix version of the eigenvalue equation seen in Eq. (2), if we identify $\mathbf{\Lambda} = \mathbf{D}$. Therefore, the eigenvectors of $\mathbf{A}$ are the columns of $\mathbf{V}$ as defined in Eq. (11), and the desired eigenvalues are the (diagonal) elements of $\mathbf{A}_n$. The game, then, is to find the $\mathbf{Q}_i$ matrices, construct $\mathbf{V}$ from Eq. (11), and then get $\mathbf{D}$ from Eq. (12).

A method for finding the $\mathbf{Q}_i$ matrices, which may be familiar from linear algebra, is based on the ***Gram-Schmidt orthogonalization*** (GSO) procedure used to construct a set of orthogonal vectors from a set of linearly independent ones. Recall that the idea is to start with two of the vectors, letting one — label it $\mathbf{v}_1$ — be the starting point. The corresponding unit vector would be $\mathbf{e}_1 \equiv \mathbf{v}_1/|\mathbf{v}_1|$. To make the second vector (call it $\mathbf{v}_2$) orthogonal to the first, we subtract from $\mathbf{v}_2$ its component along $\mathbf{v}_1$, giving us a new second vector: $\mathbf{v}_2' = \mathbf{v}_2 - (\mathbf{v}_2 \cdot \mathbf{e}_1)\,\mathbf{e}_1$. To construct a third orthogonal vector, $\mathbf{v}_3'$, we need to subtract off the components of the third original vector $\mathbf{v}_3$ that lie along $\mathbf{v}_1$ and $\mathbf{v}_2'$, and so on. If we wish these vectors to be normalized, we divide each of the resulting vectors by its magnitude.

To use GSO in the present application we start from the matrix $\mathbf{A}$ written as a set of column vectors, $\mathbf{a}_i$:

$$\mathbf{A} = \left[\begin{pmatrix} | \\ \mathbf{a}_1 \\ | \end{pmatrix} \begin{pmatrix} | \\ \mathbf{a}_2 \\ | \end{pmatrix} \cdots \begin{pmatrix} | \\ \mathbf{a}_N \\ | \end{pmatrix} \right]. \tag{14}$$

We next define a new set of mutually-orthogonal vectors, and corresponding unit vectors, as follows:

$$\begin{aligned} \mathbf{w}_1 &= \mathbf{a}_1, & \mathbf{e}_1 &= \mathbf{w}_1/|\mathbf{w}_1|; \\ \mathbf{w}_2 &= \mathbf{a}_2 - (\mathbf{a}_2 \cdot \mathbf{e}_1)\,\mathbf{e}_1, & \mathbf{e}_2 &= \mathbf{w}_2/|\mathbf{w}_2|; \\ \mathbf{w}_3 &= \mathbf{a}_3 - (\mathbf{a}_3 \cdot \mathbf{e}_1)\,\mathbf{e}_1 - (\mathbf{a}_3 \cdot \mathbf{e}_2)\,\mathbf{e}_2, & \mathbf{e}_3 &= \mathbf{w}_3/|\mathbf{w}_3|; \\ &\vdots & &\vdots \end{aligned} \tag{15}$$

Note that each $\mathbf{w}_i$ depends only on the $\mathbf{a}_j$ with $j = i$, and the $\mathbf{e}_j$'s with $j < i$; therefore, the $\mathbf{w}$'s can be built up sequentially, starting from $\mathbf{w}_1$.

Rearranging the preceding expressions, we can write

$$\begin{aligned} \mathbf{a}_1 &= |\mathbf{w}_1|\,\mathbf{e}_1, \\ \mathbf{a}_2 &= |\mathbf{w}_2|\,\mathbf{e}_2 + (\mathbf{a}_2 \cdot \mathbf{e}_1)\,\mathbf{e}_1, \\ \mathbf{a}_3 &= |\mathbf{w}_3|\,\mathbf{e}_3 + (\mathbf{a}_3 \cdot \mathbf{e}_1)\,\mathbf{e}_1 + (\mathbf{a}_3 \cdot \mathbf{e}_2)\,\mathbf{e}_2, \\ &\vdots \end{aligned}$$

These equations can be rewritten as the following matrix equation:

$$\mathbf{A} = \left[\begin{pmatrix} | \\ \mathbf{a}_1 \\ | \end{pmatrix} \begin{pmatrix} | \\ \mathbf{a}_2 \\ | \end{pmatrix} \cdots \begin{pmatrix} | \\ \mathbf{a}_N \\ | \end{pmatrix} \right] = \left[\begin{pmatrix} | \\ \mathbf{e}_1 \\ | \end{pmatrix} \begin{pmatrix} | \\ \mathbf{e}_2 \\ | \end{pmatrix} \cdots \begin{pmatrix} | \\ \mathbf{e}_N \\ | \end{pmatrix} \right] \begin{pmatrix} |\mathbf{w}_1| & \mathbf{a}_2 \cdot \mathbf{e}_1 & \mathbf{a}_3 \cdot \mathbf{e}_1 & \dots \\ 0 & |\mathbf{w}_2| & \mathbf{a}_3 \cdot \mathbf{e}_2 & \dots \\ 0 & 0 & |\mathbf{w}_3| & \dots \\ \vdots & \vdots & \vdots & \end{pmatrix}.$$

The first matrix on the right is orthogonal because the column vectors $\mathbf{e}_i$ are orthogonal, and the second matrix is of upper-triangular form. Therefore, we have succeeded in factoring $\mathbf{A}$ into the form $\mathbf{A} = \mathbf{QR}$, where

$$\mathbf{Q} = \left[\begin{pmatrix} | \\ \mathbf{e}_1 \\ | \end{pmatrix} \begin{pmatrix} | \\ \mathbf{e}_2 \\ | \end{pmatrix} \cdots \begin{pmatrix} | \\ \mathbf{e}_N \\ | \end{pmatrix} \right] \qquad \text{and} \qquad \mathbf{R} = \begin{pmatrix} |\mathbf{w}_1| & \mathbf{a}_2 \cdot \mathbf{e}_1 & \mathbf{a}_3 \cdot \mathbf{e}_1 & \dots \\ 0 & |\mathbf{w}_2| & \mathbf{a}_3 \cdot \mathbf{e}_2 & \dots \\ 0 & 0 & |\mathbf{w}_3| & \dots \\ \vdots & \vdots & \vdots & \end{pmatrix}. \tag{16}$$

Putting the pieces together, the complete QR algorithm is as follows:

1. Create an $N \times N$ matrix $\mathbf{V}$ to hold the eigenvectors and set it equal to the identity matrix $\mathbf{I}$. Additionally, choose an accuracy threshold ϵ for the off-diagonal elements of $\mathbf{V}$.

2. For a given matrix $\mathbf{A}$, compute the QR decomposition $\mathbf{A} = \mathbf{QR}$ by determining the $\mathbf{w}_i$ and corresponding $\mathbf{e}_i$ vectors as in Eqs. (15), and forming the matrices in Eqs. (16).

3. As in Eq. (5), update $\mathbf{A}$ to the new value $\mathbf{A} = \mathbf{RQ}$, using the $\mathbf{Q}$ and $\mathbf{R}$ found in step 3.

4. Update $\mathbf{V}$ by computing $\mathbf{VQ}$ using the latest $\mathbf{Q}$, as in Eq. (11).

5. Check if all the off-diagonal elements of $\mathbf{A}$ are less than ϵ; if so, end; if not, return to step 3.

Upon completion of this algorithm, the diagonal elements of $\mathbf{A}$ are the desired eigenvalues and the columns of $\mathbf{V}$ are the eigenvectors.

Implementing the full algorithm in Python is doable but time-consuming and, as might be expected, Python has its own functions to perform these computations. They are in the `numpy.linalg` package and go by the names `eigh` and `eigvalsh` (the "`h`"'s at the ends of the function names refer to "Hermitian"). `eigh` calculates both the eigenvalues and eigenvectors of Hermitian matrices (of which real symmetric matrices are a subset), and is called as `e,V = eigh(A)`, where `e` is a 1-D array of eigenvalues and `V` is a 2-D array whose columns are the corresponding eigenvectors. If one wants only the eigenvalues of a large matrix it is better to use `eigvalsh`, since it runs much faster than `eigh`. It outputs just a 1-D array of the eigenvalues. (Note: Python also has functions `eig` and `eigvals` to deal with non-Hermitian matrices, but problems involving such matrices are not common in physics.)

Breakpoint 1: Using GSO as presented in Eqs. (15), analytically create a set of normalized, orthogonal vectors from the set $\mathbf{u}_1 = (1, -1, 1)$, $\mathbf{u}_2 = (1, 0, 1)$, $\mathbf{u}_3 = (1, 1, 2)$.

7.2 Applications

In this section we consider a few situations of physical interest that can be analyzed using the eigenvalue/vector-finding techniques of the previous section. (There are, of course, many other examples.) The Exercises will give you some practice implementing the approaches presented below.

7.2.1 Principal Axes of Inertia

For a body rotating about a point O, its angular momentum $\mathbf{L}$ will *not* in general be parallel to the axis of rotation, as specified by a vector $\boldsymbol{\omega}$ (the magnitude $|\boldsymbol{\omega}| = \omega$ is the angular speed, and the direction of $\boldsymbol{\omega}$ is parallel to the axis of rotation). However, for every body there exist (at least) three (perpendicular) axes along which the angular momentum vector and the rotation axis can remain coincident; such an axis is termed a ***principal axis***. (For some bodies, e.g. a uniform sphere, there may be many such axes, possibly even an infinite number.) In this case, we can write $\mathbf{L} = \lambda\boldsymbol{\omega}$, for some real number λ. We also know that the angular momentum can be written in terms of the ***moment of inertia tensor***, $\mathbf{I}$, as $\mathbf{L} = \mathbf{I}\boldsymbol{\omega}$ (the rotational version of $\mathbf{p} = m\mathbf{v}$). Putting these two facts together, we have

$$\mathbf{I}\boldsymbol{\omega} = \lambda\boldsymbol{\omega}, \tag{17}$$

which is an eigenvalue equation. The eigenvectors $\boldsymbol{\omega}$ of this equation indicate the axes around which the body can rotate stably, and the corresponding eigenvalues λ are the moments of inertia for rotation about those axes.

7.2.2 Coupled Harmonic Oscillators

An important physical problem for which we might want to calculate eigenvalues and eigenvectors involves a collection of masses connected by springs (this is a reasonable model for a solid in which the masses are atoms and the springs represent the interatomic forces). Since this type of problem is treated in all advanced classical mechanics texts, the present discussion will omit the derivations of the relevant equations.

For this discussion, we'll consider a linear (1-D) arrangement of masses (see Figure 1 for one of the simplest cases of two coupled oscillators[2]), but the general procedure can be extended in a

[2] Figure from Wikipedia page on coupled oscillators

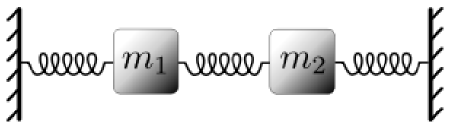


Figure 1: Two 1-D coupled harmonic oscillators

straightforward way to 2-D or 3-D arrangements. Each spring has an *equilibrium length* when no forces act to stretch or compress it; each of the N masses in the system, m_i, has a corresponding equilibrium position when the springs to which it is attached are at their equilibrium lengths. If a mass is displaced from its equilibrium position by distance x_i, the length of an attached spring will be changed away from its equilibrium length (unless the other attached mass is displaced identically). In the process, the spring will gain potential energy $V = \frac{1}{2}kd^2$, where k is its spring constant and d is the distance by which the spring length has been changed from its equilibrium length (d is negative for compression, positive for extension). A spring will change length as a result of movement of either or both of the masses that it connects; in the 1-D case, the potential energy associated with the spring connecting the i^{th} and j^{th} masses is $V_{ij} = \frac{1}{2}k_{ij}(x_i - x_j)^2$, where k_{ij} is the *spring constant* of the spring. In the case of a spring with one end attached to a fixed point, like the two outer springs in the figure, the potential energy depends only on the displacement of the single attached mass.

For the system shown in the figure, the potential energy resulting from displacements x_1 and x_2 of the two masses would be

$$U = \tfrac{1}{2}k_1x_1^2 + \tfrac{1}{2}k_{12}(x_1 - x_2)^2 + \tfrac{1}{2}k_2x_2^2, \tag{18}$$

where k_1, k_2, and k_{12} are the spring constants of the springs at the left, right, and center, respectively. In addition to potential energy, the system will have kinetic energy due to motion of the masses:

$$T = \tfrac{1}{2}m_1\dot{x}_1^2 + \tfrac{1}{2}m_2\dot{x}_2^2. \tag{19}$$

For a single object of mass m attached to one spring with spring constant k, the equation of motion is $m\ddot{x} = -kx$, or $m\ddot{x} + kx = 0$. For a set of two masses and springs as in the figure, the equations of motion (e.g., from Lagrange's equations) can be written in a similar form, as $\mathbf{M}\ddot{\mathbf{x}} + \mathbf{K}\mathbf{x} = 0$, where

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad \mathbf{M} = \begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix}, \quad \text{and} \quad \mathbf{K} = \begin{pmatrix} k_1 + k_{12} & -k_{12} \\ -k_{12} & k_2 + k_{12} \end{pmatrix}. \tag{20}$$

Here, $\mathbf{M}_{ij}$ is the coefficient in Eq. (19) of the term $\dot{x}_i\,\dot{x}_j$, ignoring the factor of $\frac{1}{2}$. (Since there are no terms of the form $\dot{x}_1\dot{x}_2$, the off-diagonal elements of $\mathbf{M}$ are zero.) Similarly, $\mathbf{K}_{ij}$ is the coefficient in Eq. (18) of the term $x_i\,x_j$ (again ignoring $\frac{1}{2}$ factors). Note that in Eq. (18), the term $\frac{1}{2}k_{12}\,(x_1 - x_2)^2$ expands to $\frac{1}{2}k_{12}\,x_1^2 + \frac{1}{2}k_{12}\,x_2^2 + \frac{1}{2}(-2k_{12})\,x_1x_2$. In constructing $\mathbf{K}$, this last term is thought of as $-\frac{1}{2}k_{12}(x_1x_2 + x_2x_1)$, and so the value $-k_{12}$ is assigned to both $\mathbf{K}_{12}$ and $\mathbf{K}_{21}$, making $\mathbf{K}$ symmetric.

In the usual way of solving such equations, we seek the ***normal modes*** of the system, and assume simple harmonic motion, i.e. $x_1 = \alpha_1 \, e^{i(\omega t - \delta_1)} = a_1 \, e^{i\omega t}$ and $x_2 = \alpha_2 \, e^{i(\omega t - \delta_2)} = a_2 \, e^{i\omega t}$, which in matrix notation looks like $\mathbf{x} = \mathbf{a} \, e^{i\omega t}$ (where the a_i's are complex, in general). Substitution into $\mathbf{M}\ddot{\mathbf{x}} + \mathbf{K}\mathbf{x} = 0$ then gives

$$\mathbf{K}\mathbf{a} = \omega^2 \, \mathbf{M}\mathbf{a}. \tag{21}$$

If the matrix $\mathbf{M}$ were the identity matrix $\mathbf{I}$ (with 1's along the diagonal and 0's everywhere else), this would be a straightforward eigenvalue equation, $\mathbf{K}\mathbf{a} = \omega^2 \, \mathbf{I}\mathbf{a} = \omega^2 \, \mathbf{a}$, with $\boldsymbol{\omega}^2$ being an eigenvalue. We would find the eigenvalues in the usual way, by solving the ***characteristic equation*** derived from the expression $|\mathbf{K} - \boldsymbol{\omega}^2 \mathbf{I}| = 0$ or, in a numerical approach, by computing the eigenvalues and eigenvectors of the matrix $\mathbf{K}$.

We can deal with the complicating $\mathbf{M}$ factor in Eq. (21) by multiplying both sides by $\mathbf{M}^{-1}$, assuming that $\mathbf{M}$ is invertible. (In all of our applications, $\mathbf{M}$ *will* be invertible since it will be diagonal, with the diagonal elements being the masses of the oscillating objects, none of which will be zero. In this case, the inverse matrix is just the diagonal matrix whose elements are the reciprocals of the masses in $\mathbf{M}$.) When we multiply Eq. (21) by $\mathbf{M}^{-1}$, we get the new equation

$$\left(\mathbf{M}^{-1}\mathbf{K}\right) \mathbf{a} = \omega^2 \, \mathbf{M}^{-1}\mathbf{M} \, \mathbf{a} = \omega^2 \, \mathbf{I}\mathbf{a} = \omega^2 \, \mathbf{a}. \tag{22}$$

This is now a standard eigenvalue equation, with $\mathbf{M}^{-1}\mathbf{K}$ as the matrix whose eigenvalues and eigenvectors we want to compute. Code to do this for the example shown in Figure 1 is here:

```
from numpy import dot
from numpy.linalg import inv

M = [[m1, 0.],[0., m2]]                   # mass matrix
Minv = inv(M)                             # invert M
K = [[k1 + k12, -k12], [-k12, k1 + k2]]   # spring-constant matrix
A = dot(Minv,K)                           # matrix multiply M^-1 K
e, V = eigh(A)
```

Breakpoint 2: Construct the matrix $\mathbf{M}^{-1}\mathbf{K}$ for the system of two coupled masses discussed above, and write out the corresponding characteristic equation for ω^2, $|\mathbf{M}^{-1}\mathbf{K} - \boldsymbol{\omega}^2\mathbf{I}| = 0$ (but don't try to solve it).

7.2.3 Energies of a Quantum System

If you've had a course in quantum mechanics, you may remember that very few problems can be solved analytically for the energies and (eigen-) states of the system. Generally, "real world" problems must be solved numerically, and in this section we'll see an example of how to do that.

The solution starts with the idea that the wavefunction $\Psi(x)$ for a particle or system can be expanded in a series of the form

$$\Psi(x) = \sum_n c_n\,\psi_n(x), \tag{23}$$

where the ψ_n are suitable ***basis functions*** for the particular problem and the c_n are expansion coefficients. These coefficients provide an alternate representation of the function Ψ since they (together with the fixed basis functions) fully capture that function. (They are general versions of the coefficients in a Fourier series.) The basis functions are designed to be orthonormal over some relevant range (a, b) so that

$$\int_a^b \psi_m(x)\,\psi_n(x)\,dx = \begin{cases} 1 & \text{if } m = n, \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

In ***bra-ket notation***,[3] $\Psi(x)$ is written as $|\Psi\rangle$, $\psi_n(x)$ becomes $|\psi_n\rangle$, and the completeness of the set of basis functions (their ability to represent any other relevant function) is encapsulated in the expansion $\hat{I} = \sum_n |\psi_n\rangle\langle\psi_n|$, where $\hat{I}$ is the identity operator which, acting on a function, leaves it unchanged. (In matrix form, $|\psi\rangle\langle\psi|$ represents the outer product of a column vector and a row vector, which creates a matrix; the expression for $\hat{I}$ then says that the identity matrix can be built from a sum of such outer products made of the basis functions in column and row vector forms.) In this notation, we have

$$|\Psi\rangle = \hat{I}\,|\Psi\rangle = \sum_n |\psi_n\rangle\langle\psi_n|\Psi\rangle\,. \tag{25}$$

(This expression is equivalent to this one for normal vectors in three dimensions, $\mathbf{v} = \sum_i (\mathbf{v}\cdot\mathbf{e}_i)\,\mathbf{e}_i$, where the $\mathbf{e}_i$ represent unit vectors in the x, y, and z directions.) By comparison with Eq. (23) we can see that

$$c_n = \langle\psi_n|\Psi\rangle\,. \tag{26}$$

(The c_n are like the components $v_i = (\mathbf{v}\cdot\mathbf{e}_i)$ found by projecting a vector $\mathbf{v}$ onto the unit basis vectors.) Now, an operator $\hat{F}$ acting on our wavefunction transforms it into a new wavefunction according to $|\Psi'\rangle = \hat{F}\,|\Psi\rangle$. If we construct the inner product $\langle\psi_m|\Psi'\rangle = \langle\psi_m|\ \hat{F}\ |\Psi\rangle$, and use the expansion of Eq. (25) between $\hat{F}$ and $|\Psi\rangle$, we find

$$\langle\psi_m|\Psi'\rangle = \sum_n \left\langle\psi_m\left|\hat{F}\right|\psi_n\right\rangle\langle\psi_n|\Psi\rangle\,. \tag{27}$$

The term containing $\hat{F}$ we can think of as a row vector times a matrix times a column vector, so

[3] In this notation, a ***bra***, $\langle\phi|$, essentially represents a row vector, and a ***ket***, $|\xi\rangle$, represents a column vector. Their inner product then is $\langle\phi|\xi\rangle$. See an advanced quantum text for more background.

it's just a number, F_{mn}, and then using Eq. (23) this expression reduces to

$$c'_m = \sum_n F_{mn}\, c_n. \tag{28}$$

This represents one row (the m^{th} one) of a matrix equation, $\mathbf{c}' = \mathbf{F}\mathbf{c}$, in which a matrix $\mathbf{F}$ (with elements F_{mn}) multiplies a column vector $\mathbf{c}$ (with elements c_n) to produce a new column vector $\mathbf{c}'$ (elements c'_m), and it shows how the operator $\hat{F}$ alters the expansion coefficients of Eq. (23) when it transforms a wavefunction. (It's important to emphasize that, in general, $c'_m \neq c_m$.)

The point of all of this is that if $\hat{F}$ is chosen to be the ***Hamiltonian*** operator of a quantum system,

$$\hat{H} = -\frac{\hbar^2}{2M}\frac{d^2}{dx^2} + V(x), \tag{29}$$

representing the energy of a particle of mass M in the potential $V(x)$, then the operator's eigenvalues are the allowed energies of the particle in that potential. To find those eigenvalues, the matrix elements $H_{mn} \equiv \langle\psi_m|\, \hat{H}\, |\psi_n\rangle$ must be computed. In traditional notation, this expression has the form

$$H_{mn} = \int_a^b \psi_m(x)\left[-\frac{\hbar^2}{2M}\frac{d^2}{dx^2} + V(x)\right]\psi_n(x)\, dx, \tag{30}$$

where a and b are the boundaries of the system's range.

Recap

- A leading method for finding eigenvalues and eigenvectors of a matrix is QR decomposition, which expresses the matrix as a product of an orthogonal matrix Q and an upper triangular matrix R.
- Gram-Schmidt orthogonalization can be used to find the matrices Q and R.

Exercises

Exercise #1

Write a program to implement QR decomposition (steps 1-3 in the algorithm above). It should input a single matrix and output the individual matrices Q & R, and their product, QR. Try it on the (symmetric) matrix below, and confirm that the product QR is the same matrix. Note: feel free to use `numpy`'s `dot` and `norm` functions. [Answer: the eigenvalues should be (21, 1, -3, -8).]

$$\begin{pmatrix} 1 & 4 & 8 & 4 \\ 4 & 2 & 3 & 7 \\ 8 & 3 & 6 & 9 \\ 4 & 7 & 9 & 2 \end{pmatrix}.$$

Exercise #2

Write a new program that makes use of your QR decomposition of Exercise #1 to implement the remaining steps of the QR algorithm listed above. Note: It might be useful to know that you can break out of a `for` loop using a combination of `if` and `break` commands; for instance:

```
for i in range(r):
   if [condition]:
      break
```

To break out of a *double* `for` loop, you can use the following construct:

```
for i in range(r):
   for j in range(s):
      if [condition]:
         break
   else:
      continue
   break
```

Try your program on the array of Exercise #1, and confirm its answers using `numpy`'s `eigh` function. Also test your function using this matrix, and compare it to `eigh`'s output:

$$\begin{pmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{pmatrix}.$$

You should find that this matrix has a doubly-*degenerate* eigenvalue, i.e. two equal ones. (A matrix can have more than two degenerate eigenvalues.) Remember that when eigenvalues are degenerate, their corresponding eigenvectors are not uniquely defined – they define a *space* of eigenvectors that all correspond to the same eigenvalue. In such a situation, one can choose any orthogonal set of vectors within that space as eigenvectors corresponding to the degenerate eigenvalue. Since the matrix above has a doubly-degenerate eigenvalue, there is a two-dimensional space of vectors corresponding to that eigenvalue. Therefore, while your program should give the *same* eigenvector as `eigh` does for the one non-degenerate eigenvalue, it probably will give eigenvectors for the degenerate eigenvalue that are *different* from those given by `eigh`. As a partial check of whether your program has produced legitimate results, confirm that all three eigenvectors found by your program are orthogonal to each other.

Exercise #3

The matrix below is the moment of inertia tensor $\mathbf{I}$ for a cube of mass M and side length a rotating about a corner. (The tensor is written in a coordinate system aligned with the edges of the cube.) (a) Using an appropriate function, find the principal axes for the cube, and the corresponding moments of inertia (in terms of the coefficient $Ma^2/12$). One of the eigenvectors can be nicely rewritten in terms of $1/\sqrt{3}$. What direction does this eigenvector correspond to in the cube? (b) Are the eigenvectors mutually orthogonal (all orthogonal to each other)? Check this! (c) Are all of the eigenvectors uniquely determined, or could you have gotten different ones? If so, which could have been different? Explain. (Hint: you might want to reread Exercise #2.)

$$\mathbf{I} = \frac{Ma^2}{12}\begin{pmatrix} 8 & -3 & -3 \\ -3 & 8 & -3 \\ -3 & -3 & 8 \end{pmatrix}.$$

Exercise #4

Two blocks, with masses m_1 and m_2, are connected in a linear arrangement by three springs, as in Figure 1. Each mass is connected to a wall by a spring with spring constants k_1 and k_2, and the masses are connected to each other by a third spring with spring constant k_{12}. Find the **normal modes** – represented by the eigenvectors – describing the motion of this system *in general*, and check it for the specific case that $m_1 = m_2 = 2.0$ and $k_1 = k_2 = k_{12} = 3.0$. (You might want to wrap your code in a Python function, so that you can pass the masses and spring constants as arguments.) What motions do the normal modes represent? What are the oscillation frequencies corresponding to those modes? Can you explain why the larger one is larger?

Exercise #5

Construct a Python function to find the normal-mode frequencies (ω) and eigenvectors for an arbitrary set of masses connected by springs in a linear arrangement, with the springs at the ends attached to fixed points. Apply your function to the particular case of three equal masses connected by four springs with equal spring constants, using the mass values and spring constants of Exercise #4.

Exercise #6

A **double pendulum** consists of one pendulum bob (with mass m_2) hanging from one end of a massless, rigid rod (length L_2) whose other end is attached to another bob (mass m_1) that hangs from another massless, rigid rod (length L_1) attached to a fixed pivot point. In terms of the angles of the two rods measured from the vertical, ϕ_1 and ϕ_2, the kinetic and potential energies of the double pendulum are

$$\begin{aligned} T &= \tfrac{1}{2}\,(m_1+m_2)\,L_1^2\,\dot{\phi}_1^2 \,+\, m_2\,L_1\,L_2\,\dot{\phi}_1\,\dot{\phi}_2 \,+\, \tfrac{1}{2}\,m_2\,L_2^2\,\dot{\phi}_2^2, \\ U &= \tfrac{1}{2}\,(m_1+m_2)\,g\,L_1\,\phi_1^2 \,+\, \tfrac{1}{2}\,m_2\,g\,L_2\,\phi_2^2. \end{aligned}$$

(a) Analytically construct the $\mathbf{M}$ and $\mathbf{K}$ matrices for this scenario. Check whether $\mathbf{M}^{-1}\mathbf{K}$ is symmetric when $m_1 = m_2$, $L_1 = L_2$. (b) Write a Python function that takes two masses and lengths as inputs and computes the eigenfrequencies (the square roots of the eigenvalues) and eigenvectors of the double pendulum, using the appropriate `numpy.linalg` function. (c) Try your function for the case that $m_1 = m_2 = 2.5$, $L_1 = L_2 = 4.0$. You should be able to relate the eigenvector components within one eigenvector to each other through a factor of $\sqrt{2}$. Describe the motions that the eigenvectors represent.

Exercise #7

In a course on quantum mechanics, you probably investigated Schrödinger's equation for the infinite one-dimensional potential well, with $V(x) = 0$ (for $0 < x < L$) and $V(x) = \infty$ (for $x \le 0$ and $L \le x$). That problem can be solved analytically, but the case of *general* $V(x)$ (for $0 < x < L$) must be solved numerically. Since the wavefunction in the well, $\Psi(x)$, must vanish at $x = 0$ and $x = L$, the typical normalized basis functions are $\psi_n(x) = \sqrt{\frac{2}{L}}\sin\left(n\frac{\pi x}{L}\right)$, with n a positive integer. Consider the specific case of a linear potential, $V(x) = ax/L$. The matrix elements to be computed are

$$H_{mn} = \frac{2}{L}\int_0^L \sin\left(m\frac{\pi x}{L}\right)\left[-\frac{\hbar^2}{2M}\frac{d^2}{dx^2} + \frac{ax}{L}\right]\sin\left(n\frac{\pi x}{L}\right)\,dx.$$

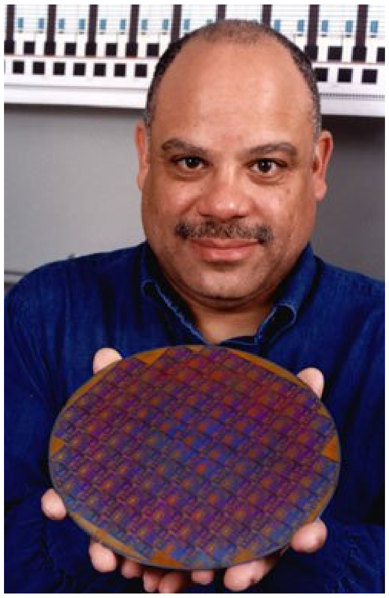
(a) Expand the integral above and use the orthonormality of the $\sin$ functions, Eq. (24), and the expression below to find a general expression for H_{mn}

$$\int_0^L x\sin\left(m\frac{\pi x}{L}\right)\sin\left(n\frac{\pi x}{L}\right)\,dx = \begin{cases} 0, & \text{if } m \ne n \text{ and both are even or odd;} \\ -\left(\dfrac{2L}{\pi}\right)^2\dfrac{mn}{(m^2-n^2)^2}, & \text{if } m \ne n \text{ and one is even, one odd;} \\ L^2/4, & \text{if } m = n. \end{cases}$$

(b) Write a Python function to evaluate the expression you found in (a) for an electron, arbitrary m and n, and for $L = 0.5$ nm, $a = 10$ eV. Be careful with units, and remember that the eigenvalue expansions involving $\sin(n\pi x/L)$ start at $n = 1$, but Python's array indices start with 0. [Python reminder: you can create an array of all zeros using the `zeros` function.]

(c) Using your function from (b), and your function for finding the eigenvalues of a matrix (or `numpy`'s `eigh` or `eigvalsh` functions), compute the first ten eigenvalues of the matrix H_{mn}. (You'll want to use just the first 10×10 portion of the infinite matrix, which will yield an approximation to the eigenvalues.) [Check:The ground state energy is $5.84\ \mathrm{eV}$.]

Scientist Profile



Mark Dean is an inventor and computer engineer, born on March 2, 1957 in Jefferson City, Tennessee. Dr. Dean earned a bachelors degree in Electrical Engineering from the University of Tennessee, a masters in Electrical Engineering from Florida Atlantic University, and a Ph.D. in Electrical Engineering from Stanford University. Upon graduating from college, Dean soon started working at a young IBM, where he rose to become the first African American IBM Fellow, the highest recognition of technical excellence at the company. In his time at IBM, where he spent the majority of his career, Dean became famous for three main contributions to the worlds of computer science and computer engineering: he helped develop the Industry Standard Architecture (ISA) systems bus, which allowed outside devices like the mouse and disk drives to be plugged directly into the computer; some of his work also helped lead to the development of the first color PC monitor; and, finally, he is known for leading the IBM team that created the first gigahertz processing chip. As evidence of his influence on the field, Dean holds three of IBMs original nine patents, and he has more than 20 patents in total to his name. His inspirational story as a pioneering African American in the sciences has been and continues to be a source of inspiration for aspiring computer scientists of all races.

Breakpoint Answers

Breakpoint 1

Start by choosing one of the vectors to be $\mathbf{v}_1$ and normalizing it:

$$\mathbf{v}_1 = \mathbf{u}_1/|\mathbf{u}_1| = \frac{1}{\sqrt{3}}(1,-1,1).$$

Next, compute the projection of $\mathbf{u}_2 = (1,0,1)$ on $\mathbf{v}_1$ and subtract the result from $\mathbf{u}_2$; then normalize the new result:

$$\mathbf{u}_2 - (\mathbf{u}_2 \bullet \mathbf{v}_1)\mathbf{v}_1 = (1,0,1) - \left[(1,0,1) \bullet \frac{1}{\sqrt{3}}(1,-1,1)\right] \cdot \frac{1}{\sqrt{3}}(1,-1,1) = \frac{1}{3}(1,2,1),$$

$$\mathbf{v}_2 = \frac{\frac{1}{3}(1,2,1)}{\sqrt{\frac{1}{9}\cdot 6}} = \frac{1}{\sqrt{6}}(1,2,1).$$

Follow the same procedure with $\mathbf{u}_3 = (1,1,2)$, subtracting its projections on both $\mathbf{v}_1$ and $\mathbf{v}_2$, find:

$$\begin{aligned}\mathbf{u}_3 - (\mathbf{u}_3 \cdot \mathbf{v}_1)\mathbf{v}_1 - (\mathbf{u}_3 \cdot \mathbf{v}_2)\mathbf{v}_2 &= (1,1,2) - \left[(1,1,2) \bullet \frac{1}{\sqrt{3}}(1,-1,1)\right] \cdot \frac{1}{\sqrt{3}}(1,-1,1) \\ &\quad - \left[(1,1,2) \bullet \frac{1}{\sqrt{6}}(1,2,1)\right] \cdot \frac{1}{\sqrt{6}}(1,2,1) = \frac{1}{2}(-1,0,1), \\ \mathbf{v}_3 &= \frac{\frac{1}{2}(-1,0,1)}{\sqrt{\frac{1}{4}\cdot 2}} = \frac{1}{\sqrt{2}}(-1,0,1).\end{aligned}$$

It's easy to confirm that these three vectors are orthonormal.

Breakpoint 2

$\mathbf{M}^{-1}$ is simply $\begin{pmatrix} m_1^{-1} & 0 \\ 0 & m_2^{-1} \end{pmatrix}$, so

$$\mathbf{M}^{-1}\mathbf{K} = \begin{pmatrix} m_1^{-1} & 0 \\ 0 & m_2^{-1} \end{pmatrix} \begin{pmatrix} k_1 + k_{12} & -k_{12} \\ -k_{12} & k_2 + k_{12} \end{pmatrix} = \begin{pmatrix} \dfrac{k_1 + k_{12}}{m_1} & -\dfrac{k_{12}}{m_1} \\ -\dfrac{k_{12}}{m_2} & \dfrac{k_2 + k_{12}}{m_2} \end{pmatrix}.$$

Therefore, the characteristic equation has the form

$$\begin{aligned}\left| \begin{pmatrix} \dfrac{k_1 + k_{12}}{m_1} & -\dfrac{k_{12}}{m_1} \\ -\dfrac{k_{12}}{m_2} & \dfrac{k_2 + k_{12}}{m_2} \end{pmatrix} - \begin{pmatrix} \omega^2 & 0 \\ 0 & \omega^2 \end{pmatrix} \right| &= \begin{vmatrix} \dfrac{k_1 + k_{12}}{m_1} - \omega^2 & -\dfrac{k_{12}}{m_1} \\ -\dfrac{k_{12}}{m_2} & \dfrac{k_2 + k_{12}}{m_2} - \omega^2 \end{vmatrix} \\ &= \left(\frac{k_1 + k_{12}}{m_1} - \omega^2\right)\left(\frac{k_2 + k_{12}}{m_1} - \omega^2\right) - \frac{k_{12}^2}{m_1 m_2} = 0.\end{aligned}$$