

PY410 / 505
Computational Physics 1

Salvatore Rappoccio

Next up : Linear algebra

- Covered in Chapter 4 of Garcia and Chapter 2 of Numerical Recipes
- Huge number of applications!
 - Complex circuit diagrams
 - Coupled oscillators
 - General solution of fitting arbitrary curves
- We'll learn how to :
 - Solve linear equations
 - Compute eigenvalues
 - Apply these to various applications

Linear Algebra

- You should all be familiar with the basics of linear algebra

<http://xkcd.com/184/>

- Vectors $\mathbf{Ax} = \mathbf{b}$,
- Matrices
- Solving matrix equations
- Gaussian (or Gauss-Jordan) elimination

- http://en.wikipedia.org/wiki/Gaussian_elimination

$$\begin{bmatrix} \cos 90^\circ & \sin 90^\circ \\ -\sin 90^\circ & \cos 90^\circ \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ a_1 \end{bmatrix}$$

- We'll go over the computational issues here

Linear Algebra

- Recall Cramer's rule :
 - http://en.wikipedia.org/wiki/Cramer's_rule

- If we have the determinant of a matrix A :

$$|\mathbf{A}| = \sum_{i=1}^n (-1)^{i+j} A_{ij} |\mathbf{R}_{ij}| ,$$

- where R_{ij} is the “residual matrix” or “minor” of A , removing row i and column j
- Then the inverse of the matrix is :

$$A_{ij}^{-1} = (-1)^{i+j} \frac{|\mathbf{R}_{ji}|}{|\mathbf{A}|} .$$

- This is a recursive rule

Linear Algebra

- Computing this “brute force” way is okay for small n , but problematic for large n (>10)
- Scales as n factorial ($n!$)

- To see this, consider an expansion of the determinant :

$$|\mathbf{A}| = \sum_P (-1)^P A_{1p_1} A_{2p_2} \cdots A_{np_n} ,$$

- Here, P runs over the $n!$ permutations of the indices
- $20! = 2.43 \times 10^{18}$ yipes! (which is “yipes, factorial”)
- OK, well, scratch that idea.
 - What else ya got, Sal?

Linear Algebra

- Medium sized matrices ($n \sim 10^2 - 10^3$)
 - Gaussian elimination, LU-decomposition, and Householder method
- Larger matrices ($n > 10^3$)
 - Storage becomes a problem
 - Cannot practically do this for arbitrary matrices
 - However, most matrices are “sparse” with lots of zeroes in practical applications
 - Can, however, store and solve these fairly well

Linear Algebra

- Linear algebra is the raison d'être for numpy.
 - Let's just use it.
- Most of the scipy algorithms for linear algebra are from LAPACK. (Linear Algebra Package)
 - <http://www.netlib.org/lapack/>
- Other options:
 - BLAS (Basic Linear Algebra Subprograms)
 - http://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms
 - LAPACK
 - BOOST Basic Linear Algebra Library
 - http://www.boost.org/doc/libs/1_54_0/libs/numeric/ublas/doc/index.htm
 - matlab (the “mat” in “matlab” stands for “matrix”)
 - <http://www.mathworks.com/products/matlab/>

Linear Algebra

- Game plan:
 - Use numpy software.
 - Go over algorithms that are used internally
 - Check into use cases

Linear Algebra

- So if you recall the Gaussian Elimination, we have a matrix equation :

$$\mathbf{Ax} = \mathbf{b} ,$$

- Or, written out for the case of $n=3$:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} ,$$

$$\begin{pmatrix} a_{00}x_0 + a_{01}x_1 + a_{02}x_2 \\ a_{10}x_0 + a_{11}x_1 + a_{12}x_2 \\ a_{20}x_0 + a_{21}x_1 + a_{22}x_2 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} .$$

Linear Algebra

- We take linear combinations of the rows to convert the matrix into “reduced row echelon form” :
 - Multiply first equation by a_{10}/a_{00} and subtract from second:

$$a_{10}x_0 + a_{11}x_1 + a_{12}x_2 - (a_{10}/a_{00})(a_{00}x_0 + a_{01}x_1 + a_{02}x_2) = b_1 - (a_{10}/a_{00})b_0 .$$

- Then x_0 is eliminated from the second equation:

$$(a_{11} - a_{10}a_{01}/a_{00})x_1 + (a_{12} - a_{10}a_{02}/a_{00})x_2 = b_1 - (a_{10}/a_{00})b_0 ,$$

- which can be written as:

$$a'_{11}x_1 + a'_{12}x_2 = b'_1 .$$

- Repeat until you run out of rows

- Example:

$$\left[\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 1 & 1 & -1 & 1 \\ 3 & 11 & 5 & 35 \end{array} \right] \rightarrow \left[\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 0 & -2 & -2 & -8 \\ 0 & 2 & 2 & 8 \end{array} \right] \rightarrow \left[\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 0 & -2 & -2 & -8 \\ 0 & 0 & 0 & 0 \end{array} \right] \rightarrow \left[\begin{array}{ccc|c} 1 & 0 & -2 & -3 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 0 & 0 \end{array} \right]$$

Linear Algebra

- Can also think about this in terms of an “augmented” matrix where you add the vector b as another column:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & b_0 \\ a_{10} & a_{11} & a_{12} & b_1 \\ a_{20} & a_{21} & a_{22} & b_2 \end{pmatrix}$$

- You eliminate each row iteratively, reducing the dimension by one each time:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & b_0 \\ \frac{a_{10}-a_{00}a_{10}}{a_{00}} & a_{11} - \frac{a_{01}a_{10}}{a_{00}} & a_{12} - \frac{a_{02}a_{10}}{a_{00}} & b_1 - b_0a_{10}/a_{00} \\ a_{20} - \frac{a_{00}a_{20}}{a_{00}} & a_{21} - \frac{a_{01}a_{20}}{a_{00}} & a_{22} - \frac{a_{02}a_{20}}{a_{00}} & b_2 - b_0a_{20}/a_{00} \end{pmatrix}$$

$$\equiv \begin{pmatrix} a'_{00} & a'_{01} & a'_{02} & b'_0 \\ 0 & a'_{11} & a'_{12} & b'_1 \\ 0 & a'_{21} & a'_{22} & b'_2 \end{pmatrix}$$

$$\begin{pmatrix} a'_{00} & a'_{01} & a'_{02} & b'_0 \\ 0 & a'_{11} & a'_{12} & b'_1 \\ 0 & a'_{21} - a'_{11}a'_{21}/a'_{11} & a'_{22} - a'_{12}a'_{21}/a'_{11} & b'_2 - b'_1a'_{21}/a'_{11} \end{pmatrix}$$

$$\equiv \begin{pmatrix} a''_{00} & a''_{01} & a''_{02} & b''_0 \\ 0 & a''_{11} & a''_{12} & b''_1 \\ 0 & 0 & a''_{22} & b''_2 \end{pmatrix}$$

Linear Algebra

- To solve for the actual equation, we then use “back substitution”, starting at the last row

–For instance

$$x_2 = \frac{b_2''}{a_{22}''} .$$

–Then we use the second equation

$$x_1 = \frac{1}{a_{11}''} [b_1'' - a_{12}''x_2] .$$

–And finally

$$x_0 = \frac{1}{a_{00}''} [b_0'' - a_{01}''x_1 - a_{02}''x_2] .$$

Linear Algebra

- Gaussian elimination is $O(n^3)$ operations
 - n^2 matrix elements, and $O(n)$ row operations on each
- Back-substitution is $O(n^2)$ operations
- So the total is $O(n^3) + O(n^2) \sim O(n^3)$ for large n
- Much better than $O(n!) \sim O(n^3)$
- Easy to extend to arbitrarily large n 's
 - But, as we mentioned earlier, storage becomes a problem

Linear Algebra

- You can see one problem already
- If any of the rows have a zero as the first element, then you divide by zero and get an exception
- So, need to make sure this doesn't happen by “partial pivoting” :
 - Before performing the i th row operation, search for the element a_{ki} ($k=i, \dots, n-1$) with the largest magnitude
 - If $k \neq i$, interchange rows i and k of the augmented matrix, and interchange $x_i \longleftrightarrow x_k$
 - Perform as usual
- Will not fail for small a_i 's, also stable to roundoff errors
- Note : in “full” pivoting you swap rows AND columns



Linear Algebra

- Another variation is Gauss-Jordan elimination
 - Does not require the backsubstitution step
 - Replaces A by the inverse “in place”, avoiding memory copy
- So, we have another augmented equation (again in $n=3$):

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_0 & a_{00}^{-1} & a_{01}^{-1} & a_{02}^{-1} \\ x_1 & a_{10}^{-1} & a_{11}^{-1} & a_{12}^{-1} \\ x_2 & a_{20}^{-1} & a_{21}^{-1} & a_{22}^{-1} \end{pmatrix} = \begin{pmatrix} b_0 & 1 & 0 & 0 \\ b_1 & 0 & 1 & 0 \\ b_2 & 0 & 0 & 1 \end{pmatrix}$$

- This is a way of stating :

$$\mathbf{Ax} = \mathbf{b} , \quad \text{and} \quad \mathbf{AA}^{-1} = \mathbf{1} .$$

Linear Algebra

- Algorithm :
 - Start with zeroth row, divide by $a(00)$ and subtract
 - Subtract all zeroth column entries (as in Gaussian elimination)
 - For row $i=1, \dots, n-1$, divide by diagonal element $a(ii)$ and eliminate elements in column i other than the diagonal by subtracting $a(ij)$ times the i th row elements from the j th row
 - Gaussian elimination : subtracts only those below diagonal
 - Gauss-Jordan : subtracts ALL elements where $i \neq j$
 - Simultaneously perform on the augmented $(b \ 1)$ matrix on the RHS

Linear Algebra

- In the end we get :

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_0 & a_{00}^{-1} & a_{01}^{-1} & a_{02}^{-1} \\ b_1 & a_{10}^{-1} & a_{11}^{-1} & a_{12}^{-1} \\ b_2 & a_{20}^{-1} & a_{21}^{-1} & a_{22}^{-1} \end{pmatrix} = \begin{pmatrix} x_0 & a_{00}^{-1} & a_{01}^{-1} & a_{02}^{-1} \\ x_1 & a_{10}^{-1} & a_{11}^{-1} & a_{12}^{-1} \\ x_2 & a_{20}^{-1} & a_{21}^{-1} & a_{22}^{-1} \end{pmatrix}$$

- A is replaced by a unit matrix
- b is reduced to input vector x
- Unit matrix on RHS is replaced by inverse A^{-1}
- Can also implement pivoting in this algorithm to make sure we have numeric stability

Linear Algebra

- Another variation is “LU decomposition” (“lower-upper”)
- Reduce A to an L^*U product: $\mathbf{Ax} = \mathbf{b}$,

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} \alpha_{00} & 0 & 0 \\ \alpha_{10} & \alpha_{11} & 0 \\ \alpha_{20} & \alpha_{21} & \alpha_{22} \end{pmatrix} \begin{pmatrix} \beta_{00} & \beta_{01} & \beta_{02} \\ 0 & \beta_{11} & \beta_{12} \\ 0 & 0 & \beta_{22} \end{pmatrix}$$

- Then solve the problem in two steps :
 - Solve $\mathbf{Ly} = \mathbf{b}$ with forward substitution:

$$y_0 = \frac{b_0}{\alpha_{00}}, \quad y_i = \frac{1}{\alpha_{ii}} \left[b_i - \sum_{j=0}^{i-1} \alpha_{ij} y_j \right], \quad i = 1, 2, \dots, n-1.$$

- Solve $\mathbf{Ux} = \mathbf{y}$, since $\mathbf{Ax} = \mathbf{L}(\mathbf{Ux}) = \mathbf{Ly} = \mathbf{b}$,
with backward substitution :

$$x_{n-1} = \frac{y_{n-1}}{\beta_{n-1,n-1}}, \quad x_i = \frac{1}{\beta_{ii}} \left[y_i - \sum_{j=i+1}^{n-1} \beta_{ij} x_j \right], \quad i = n-2, \dots, 0.$$

Linear Algebra

- Factoring the matrix $A = LU$ can be done with “Crout’s Algorithm” :

- Set $\alpha_{ii} = 1$ for $i=0, \dots, n-1$

- for each $j = 0, \dots, n-1$:

- for $i = 0, \dots, j$

- compute

$$\beta_{ij} = a_{ij} - \sum_{k=0}^{i-1} \alpha_{ik} \beta_{kj} .$$

- for $j = j+1, \dots, n-1$

- compute

$$\alpha_{ij} = \frac{1}{\beta_{jj}} \left[a_{ij} - \sum_{k=0}^{j-1} \alpha_{ik} \beta_{jk} \right] .$$

- Replaces A by LU “in place”:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \longrightarrow \begin{pmatrix} \beta_{00} & \beta_{01} & \beta_{02} \\ \alpha_{10} & \beta_{11} & \beta_{12} \\ \alpha_{20} & \alpha_{21} & \beta_{22} \end{pmatrix}$$

The RH matrix
is NOT a matrix!
It is a storage unit
in the computer!

Linear Algebra

- Simple special case : Tridiagonal matrices
- If you have a problem such as :

$$\mathbf{M} = \begin{pmatrix} 2-c & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2-c & -1 & \cdots & 0 & 0 \\ 0 & -1 & 2-c & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2-c & -1 \\ 0 & 0 & 0 & \cdots & -1 & 2-c \end{pmatrix},$$

- This matrix is sparse!
 - Can solve in $O(n)$ operations
- Equations are:

$$M_i^- u_{i-1} + M_i^0 u_i + M_i^+ u_{i+1} = b_i, \quad i = 1, \dots, n-1.$$

- Can show that (recursively):

$$\alpha_{i-1} = -\frac{M_i^-}{M_i^0 + \alpha_i M_i^+}, \quad \beta_{i-1} = \frac{b_i - \beta_i M_i^+}{M_i^0 + \alpha_i M_i^+}.$$

Linear Algebra

- Can start from the right boundary value for $i=n-2, \dots, 0$

$$u_n = \alpha_{n-1}u_{n-1} + \beta_{n-1} \quad \text{if} \quad \alpha_{n-1} = 0, \quad \beta_{n-1} = u_n,$$

- and then solve from the left boundary value for $i=1, \dots, n-1$

$$u_{i+1} = \alpha_i u_i + \beta_i,$$

- So we “sweep twice” and obtain $O(n)$ operations

Linear Algebra

- Examples:
 - Polynomial fits (again)
 - Circuit diagrams
 - Boundary value problems

Recall : General fitting of curves

- This is a matrix equation, so we define the “design matrix” :

$$A_{ij} = \frac{Y_j(x_i)}{\sigma_i}$$

$$\mathbf{A} = \begin{bmatrix} Y_1(x_1)/\sigma_1 & Y_2(x_1)/\sigma_1 & \dots \\ Y_1(x_2)/\sigma_2 & Y_2(x_2)/\sigma_2 & \dots \\ \dots & \dots & \dots \end{bmatrix}$$

- Then our chi2 minimization becomes :

$$(\mathbf{A}^T \mathbf{A}) \vec{a} = \mathbf{A}^T \vec{b}$$

- so :

$$\vec{a} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \vec{b}$$

Recall : General fitting of curves

- If we define the “correlation matrix” :

$$\mathbf{C} = (\mathbf{A}^T \mathbf{A})^{-1}$$

- Then the uncertainty on a_j is :

$$\sigma_{a_j} = \sqrt{C_{jj}}$$

Recall : General fitting of curves

- As a first example, let's look at polynomial fits

$$y = \sum_{k=0}^{m-1} a_k x^k .$$

- Slight generalization of the linear fit we did previously
- General solution is to minimize the chi2 :

$$\chi^2(\vec{a}) \equiv \sum_{i=0}^{n-1} \left(\frac{y_i - y(x; \vec{a})}{\sigma_i} \right)^2$$

- In this case :

$$\chi^2(\vec{a}) = \sum_{i=0}^{n-1} \left(\frac{y_i - \sum_{j=0}^M a_j x^j}{\sigma_i} \right)^2$$

Recall : General fitting of curves

- Our design matrix is therefore :

$$A_{ij} = x_i^j / \sigma_i$$

- Caveat : This oftentimes is ill-formed, so don't go too crazy here. Typically we do quadratic, cubic, quartic, but above that it strains credibility.

Polynomial fits

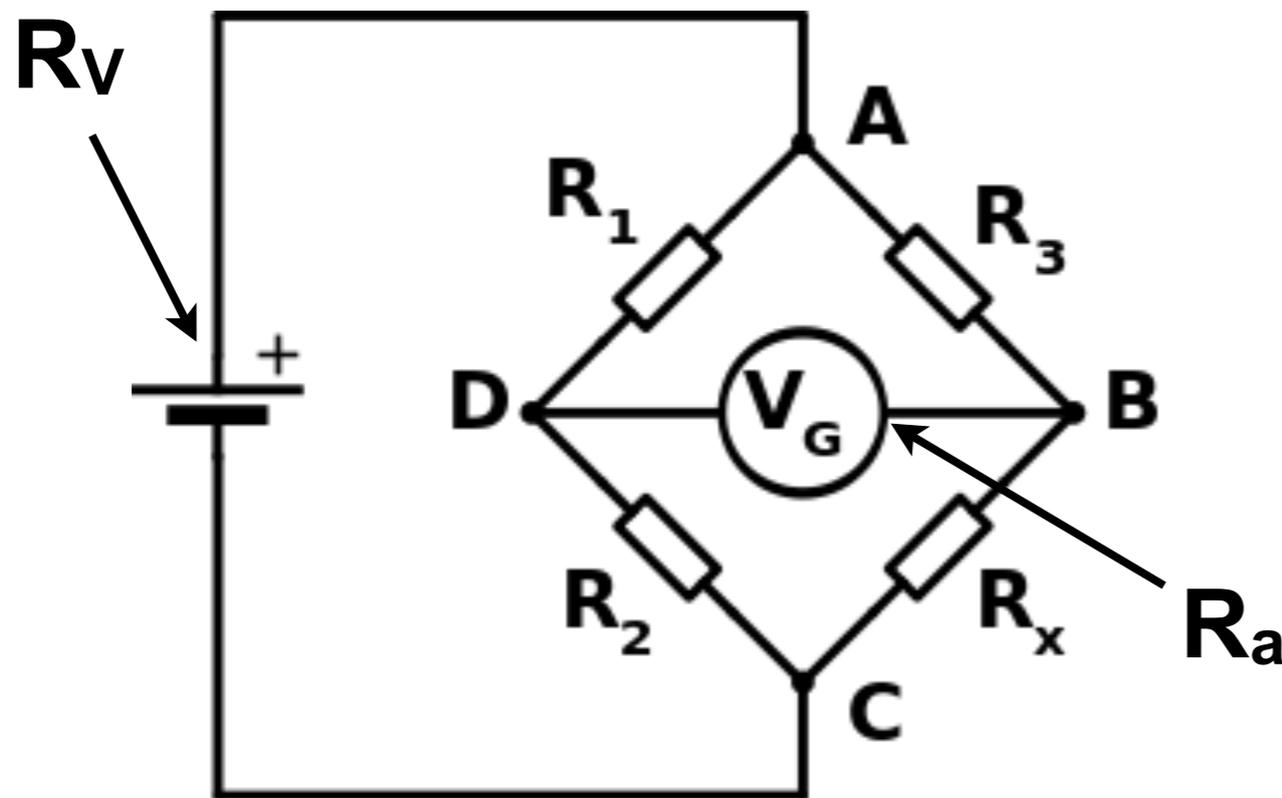
- Can finally generalize our formalism to arbitrary functional fits
- Example : quadratic fit for our CO₂ data!

Linear Algebra

- Example :

- Kirchoff's Law for a Wheatstone bridge :

- http://en.wikipedia.org/wiki/Wheatstone_bridge



Solve for R_x given R_1, R_2, R_2, i and V

- This is a matrix equation: $\mathbf{Ri} = \mathbf{v}$,

$$\begin{pmatrix} R_1 + R_v & R_2 & R_v \\ R_1 + R_a & -R_a & -R_3 \\ R_x + R_a & -R_2 - R_x - R_a & R_x \end{pmatrix} \begin{pmatrix} i_1 \\ i_2 \\ i_3 \end{pmatrix} = \begin{pmatrix} v \\ 0 \\ 0 \end{pmatrix}$$

Linear Algebra

- Try to “zero” the potential at V_G :

$$V_G = \left(\frac{R_x}{R_3 + R_x} - \frac{R_2}{R_1 + R_2} \right) V_s$$

Linear Algebra

- Example : boundary value problems

–Consider :

$$\frac{d^2u}{dx^2} = -\frac{\pi^2}{4}(u + 1) ,$$

–with Dirichlet boundary conditions $u(0) = 0$ and $u(1)=1$

–We can discretize this :

$$\frac{2u_i - u_{i+1} - u_{i-1}}{h^2} = \frac{\pi^2}{4}(u_i + 1) , \quad i = 1, \dots, N - 1$$

–This is therefore a sparse matrix equation with $c = h^2 \pi^2/4$:

$$\mathbf{M} = \begin{pmatrix} 2-c & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2-c & -1 & \cdots & 0 & 0 \\ 0 & -1 & 2-c & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2-c & -1 \\ 0 & 0 & 0 & \cdots & -1 & 2-c \end{pmatrix} , \quad \mathbf{x} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix} , \quad \mathbf{b} = \begin{pmatrix} u_0 + c \\ c \\ c \\ \vdots \\ c \\ u_N + c \end{pmatrix} ,$$

Hands on!

Linear Algebra

- Today : Eigenvalues and eigenvectors, more hands on
- See Chapter 11 of Numerical Recipes
- In this case, even they recommend using packaged software for eigenvalues and eigenvectors, but let's get the general gist here

Linear Algebra

- Example : normal modes of a harmonic oscillator between n objects

$$\mathcal{L} = \frac{1}{2} \sum_{j,k}^n M_{jk} \dot{q}_j \dot{q}_k - \frac{1}{2} \sum_{j,k}^n A_{jk} q_j q_k ,$$

- “Normal mode” is the mode of the system where all of the coordinates oscillate with some frequency ω :

$$q_j(t) = x_j e^{i\omega t} , \quad \sum_{j=1}^n [A_{jk} - M_{jk}\omega^2] x_j = 0 , \quad (\mathbf{A} - \mathbf{M}\omega^2)\mathbf{x} = 0 .$$

- Homogeneous matrix equation has solutions for ω for which the determinant is zero:

$$\det |\mathbf{A} - \mathbf{M}\omega^2| = 0 .$$

Eigenvalues and Eigenvectors

- This is an instance of a general class of eigenvalue and eigenvector problems
- So, if A is an $n \times n$ matrix, x is a column vector (the “right” eigenvector), λ is a number (the “eigenvalue”), and :

$$\mathbf{Ax} = \lambda \mathbf{x} ,$$

- Then the equation is satisfied when :

$$|\mathbf{A} - \lambda \mathbf{1}| = \sum_{k=0}^n a_k \lambda^k = 0 ,$$

- This is an n -th degree polynomial in λ that depends on the matrix elements
- N -degree polynomial \implies n different eigenvalues, so we need to solve for them

Eigenvalues and Eigenvectors

- Example :

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$$

- The eigenvalue equation and eigenvalues are :

$$\mathbf{A} = \begin{pmatrix} 1 - \lambda & 1 \\ 1 & 2 - \lambda \end{pmatrix}$$

$$(1 - \lambda)(2 - \lambda) - 1 = \lambda^2 - 3\lambda + 1 = 0 \quad \Rightarrow \quad \lambda_0 = \frac{3 + \sqrt{5}}{2}, \quad \lambda_1 = \frac{3 - \sqrt{5}}{2}.$$

- We solve for the eigenvectors :

$$\begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = (1 \pm \sqrt{2}) \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \quad \mathbf{x}^{(0)} = \begin{pmatrix} \sqrt{\frac{1}{3}} \\ \sqrt{\frac{2}{3}} \end{pmatrix}, \quad \mathbf{x}^{(1)} = \begin{pmatrix} \sqrt{\frac{1}{3}} \\ -\sqrt{\frac{2}{3}} \end{pmatrix}.$$

Eigenvalues and Eigenvectors

- Also consider :

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} .$$

- So we have:

$$(1 - \lambda)^2 + 1 = \lambda^2 - 2\lambda + 2 = 0$$

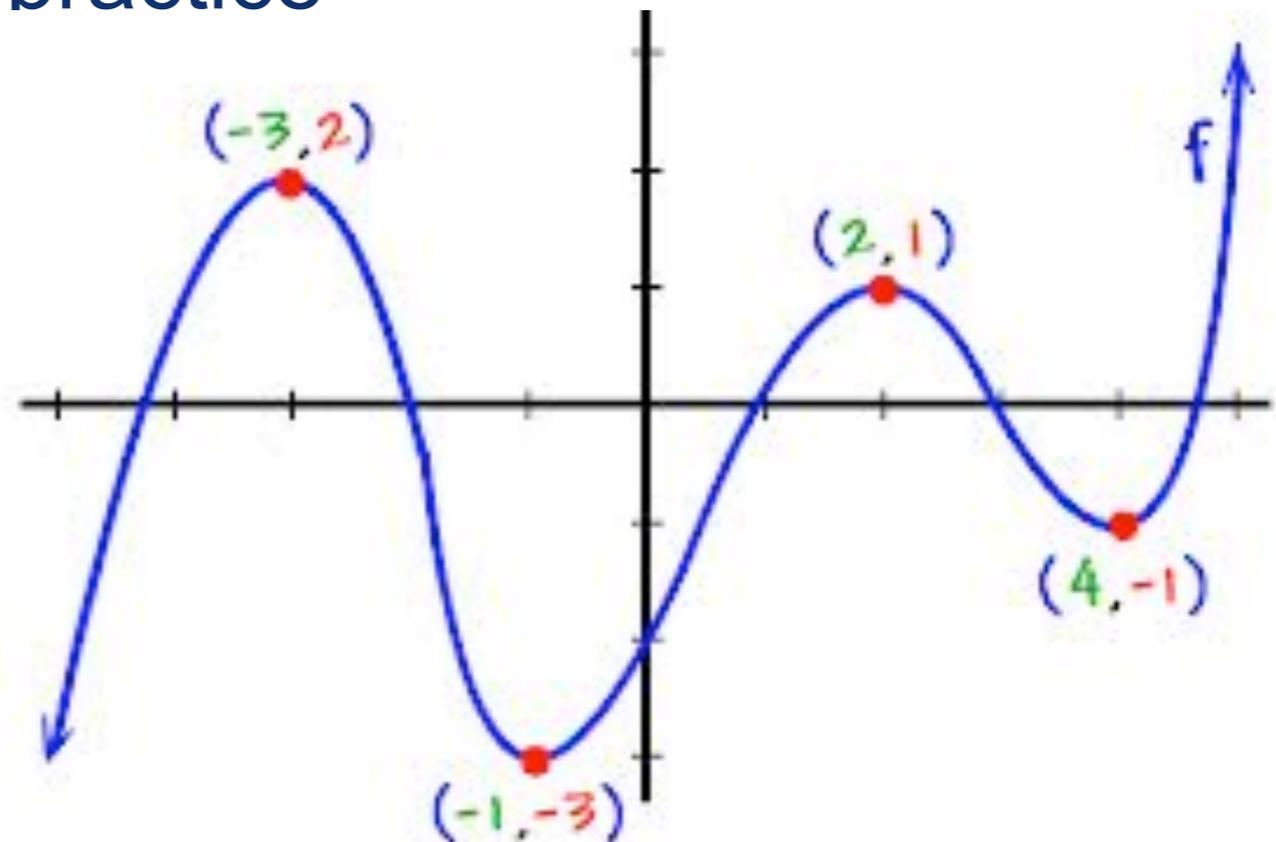
- And thus the eigenvalues are complex :

$$\lambda_0 = 1 + i , \quad \lambda_1 = 1 - i , \quad \text{where} \quad i = \sqrt{-1} .$$

Eigenvalues and Eigenvectors

- “Hey Sal! I know what we can do!” you say. “We can just find the roots of the characteristic polynomial with our root-finding methods!”
- “Excellent idea in principle, my clever students,” I say. “But unfortunately, it’s hard in practice”

- Need to know roughly where the roots are ahead of time
- You don’t get the eigenvectors this way either



- So, we’ll look at ways to do both!

Eigenvalues and Eigenvectors

- First, define “left” eigenvectors:

$$\mathbf{y}\mathbf{A} = \lambda\mathbf{y} ,$$

- In this case, \mathbf{y} is a row vector.

- This implies : $\mathbf{A}^T\mathbf{y} = \lambda\mathbf{y} ,$

- Since : $|\mathbf{A}^T - \lambda\mathbf{1}| = |\mathbf{A} - \lambda\mathbf{1}| ,$

then the left and right eigenvalues are the same

- However the left and right eigenvectors are not necessarily the same
- Left eigenvector $\mathbf{0}$, though, is orthogonal to right eigenvector $\mathbf{1}$, etc

Eigenvalues and Eigenvectors

- If we consider a complex matrix A , we can define the Hermitian conjugate :

$$(\mathbf{A}^\dagger)_{ij} = a_{ji}^* ,$$

- This is also referred to as the “conjugate transpose”
- If the a_{ij} are real, this is just the transpose
- Define a “normal” matrix if the conjugate transpose commutes with the matrix:

$$\mathbf{N} \cdot \mathbf{N}^\dagger = \mathbf{N}^\dagger \cdot \mathbf{N} .$$

- Note that normal matrices have the same left and right eigenvector sets

Eigenvalues and Eigenvectors

- A Hermitian matrix (or self-adjoint matrix) is defined when

$$\mathbf{H}^\dagger = \mathbf{H} ,$$

- When the elements are real, this is called a “symmetric” matrix:

$$\mathbf{S}^T = \mathbf{S} ,$$

- These special matrices have :
 - n real eigenvalues
 - eigenvectors are orthogonal
 - the set of eigenvectors is “complete” (spans the n-dim space)

Eigenvalues and Eigenvectors

- Computation of eigenvalues/eigenvectors for symmetric or Hermitian matrices depends on a nice property
- If we consider the matrices formed by the right and left eigenvectors \mathbf{X} and \mathbf{Y} :

$$\mathbf{X} = \begin{pmatrix} \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} \\ \sqrt{\frac{2}{3}} & -\sqrt{\frac{2}{3}} \end{pmatrix}, \quad \mathbf{Y} = \begin{pmatrix} \sqrt{\frac{2}{3}} & \sqrt{\frac{1}{3}} \\ \sqrt{\frac{2}{3}} & -\sqrt{\frac{1}{3}} \end{pmatrix} = \frac{2\sqrt{2}}{3} \mathbf{X}^{-1}.$$

- In general : $\mathbf{Y} \propto \mathbf{X}^{-1}$
- Can choose the normalization appropriately so that

$$\mathbf{Y} = \mathbf{X}^{-1}$$

Eigenvalues and Eigenvectors

- How does this help us?

$$\begin{aligned}\mathbf{Y} \cdot \mathbf{A} \cdot \mathbf{X} &= \mathbf{X}^{-1} \cdot \mathbf{A} \cdot \mathbf{X} \\ &= \mathbf{X}^{-1} \cdot \lambda \cdot \mathbf{X} \\ &= \lambda\end{aligned}$$

- Can exploit this fact to compute the eigenvalues!

Eigenvalues and Eigenvectors

- This is a similarity transformation!

$$\mathbf{A} \rightarrow \mathbf{Z}^{-1} \cdot \mathbf{A} \cdot \mathbf{Z}$$

- The interesting bit is that symmetry transformations leave the eigenvalues unchanged:

$$\begin{aligned} \det |\mathbf{Z}^{-1} \cdot \mathbf{A} \cdot \mathbf{Z} - \lambda \mathbf{1}| &= \det |\mathbf{Z}^{-1} \cdot (\mathbf{A} - \lambda \mathbf{1}) \cdot \mathbf{Z}| \\ &= \det |\mathbf{Z}| \det |\mathbf{A} - \lambda \mathbf{1}| \det |\mathbf{Z}^{-1}| \\ &= \det |\mathbf{A} - \lambda \mathbf{1}| \end{aligned}$$

- So, we can solve this easier problem instead

Eigenvalues and Eigenvectors

- Let X be the matrix of eigenvectors
 - If the matrix A is real and symmetric, then the eigenvectors are real and orthonormal, and :

$$\mathbf{X}^{-1} = \mathbf{X}^T$$

- If the matrix A is Hermitian, the matrix of eigenvectors is unitary :

$$\mathbf{X}^{-1} = \mathbf{X}^\dagger$$

Eigenvalues and Eigenvectors

- Generalized eigenvalue problem : solve

$$\mathbf{A} \cdot \mathbf{x} = \lambda \mathbf{B} \cdot \mathbf{x}$$

- Strategy : Successively apply similarity transformations until the matrix is “almost” diagonal
 - Either diagonal, block diagonal, or tridiagonal and also easy to solve
- Nonsymmetric matrices will not, in general, have similarity matrices with real components
 - So, cannot use real similarity matrix
 - But! “Almost” can : will reduce to block-diagonal with two-by-two blocks replacing the complex eigenvalues

– Think $\begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}$

Eigenvalues and Eigenvectors

- Can then repeatedly apply similarity transformations until we can solve the problem “easily” :

$$\begin{aligned} \mathbf{A} &\rightarrow \mathbf{P}_1^{-1} \cdot \mathbf{A} \cdot \mathbf{P}_1 \rightarrow \mathbf{P}_2^{-1} \cdot \mathbf{P}_1^{-1} \cdot \mathbf{A} \cdot \mathbf{P}_1 \cdot \mathbf{P}_2 \\ &\rightarrow \mathbf{P}_3^{-1} \cdot \mathbf{P}_2^{-1} \cdot \mathbf{P}_1^{-1} \cdot \mathbf{A} \cdot \mathbf{P}_1 \cdot \mathbf{P}_2 \cdot \mathbf{P}_3 \rightarrow \text{etc.} \end{aligned}$$

- If we end up in completely diagonal form, then the eigenvectors are just the columns of the total transformation:

$$\mathbf{X}_R = \mathbf{P}_1 \cdot \mathbf{P}_2 \cdot \mathbf{P}_3 \cdot \dots$$

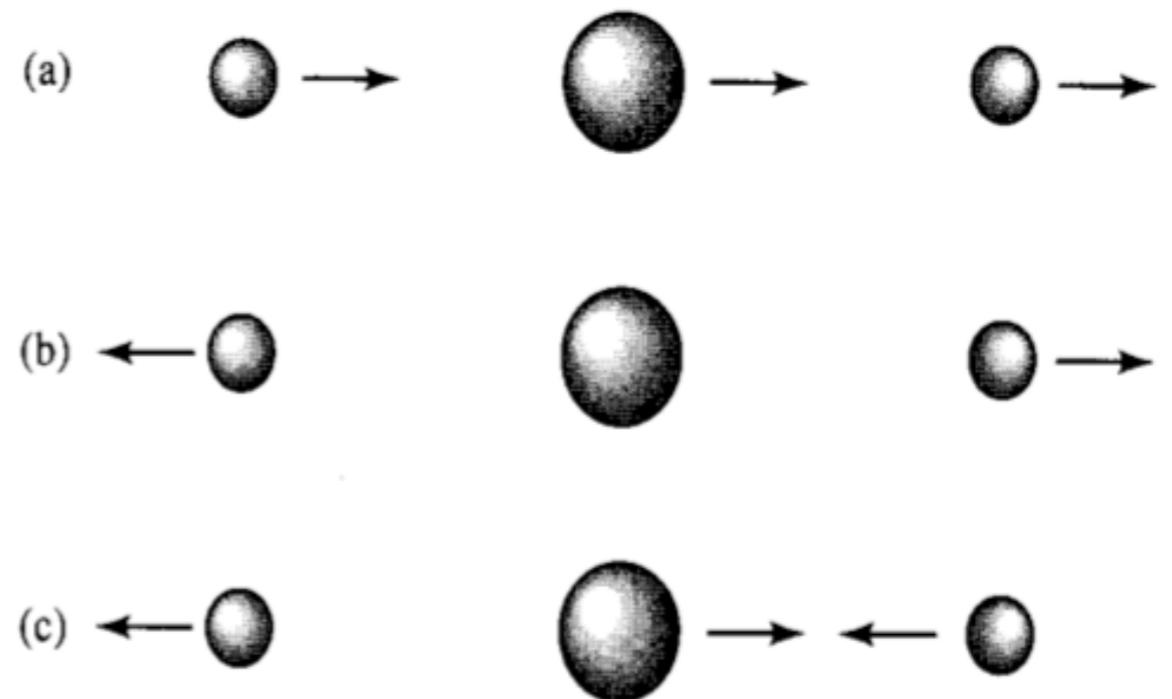
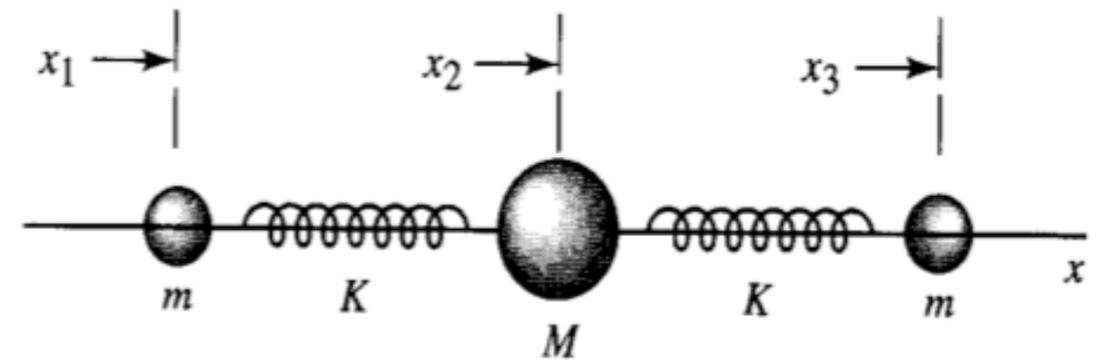
- Sometimes just want eigenvalues, in which case can stop when we reduce to triangular form
 - Then eigenvalues are the diagonal elements!

Eigenvalues and Eigenvectors

- One strategy we will look at :
 - Reduce to tridiagonal form by the Householder algorithm (Chapter 11 Section 2 of Numerical Recipes)
 - Solve the tridiagonal matrix problem using a QL algorithm with implicit shifts (Chapter 11 Section 3 of Numerical Recipes)
 - Q = orthogonal
 - L = lower triangular matrix
- These are basically “uninteresting” to go through the gory details, because they’re mostly just math tricks
- Instead, let’s just focus on an application

Eigenvalues and Eigenvectors

- Consider a linear triatomic molecule
 - Fowles and Cassiday, Chapter 11, Section 4
 - Taylor, Chapter 11, Section 6
- Approximate by masses on springs



Eigenvalues and Eigenvectors

- Lagrangian for the general case is :

$$L = \frac{1}{2}m_1\dot{x}_1^2 + \frac{1}{2}m_2\dot{x}_2^2 + \frac{1}{2}m_3\dot{x}_3^2 - \frac{1}{2} [k_{12}(x_1 - x_2)^2 + k_{23}(x_2 - x_3)^2]$$

- The equations of motion are

$$m_1\ddot{x}_1 = \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{x}_1} \right) = \frac{\partial L}{\partial x_1} = -k_{12}x_1 + k_{12}x_2$$

$$m_2\ddot{x}_2 = \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{x}_2} \right) = \frac{\partial L}{\partial x_2} = k_{12}x_1 - (k_{12} + k_{23})x_2 + k_{23}x_3$$

$$m_3\ddot{x}_3 = \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{x}_3} \right) = \frac{\partial L}{\partial x_3} = -k_{23}x_3 + k_{23}x_2$$

- For our specific problem, we have

$$m_1 = m_3, m_2 = 2m_1, k_{12} = k_{23} = K$$

Eigenvalues and Eigenvectors

- Can be written as a matrix equation:

$$\mathbf{M}\ddot{\mathbf{q}} = -\mathbf{K}\mathbf{q}$$

- with normal modes of the form :

$$\mathbf{q}(t) = \mathbf{a} \cos(\omega t - \delta)$$

- The normal mode frequencies are eigenvalues of the generalized eigenvalue equation:

$$\mathbf{K}\mathbf{a} = \omega^2 \mathbf{M}\mathbf{a}$$

Eigenvalues and Eigenvectors

- The eigenvalues are therefore:

$$\omega_1^2 = 0, \quad \omega_2^2 = \frac{K}{m}, \quad \omega_3^2 = \frac{K}{m} \left(1 + \frac{2m}{M} \right)$$

- And the eigenvectors are :

$$\mathbf{a}_1 = a_{11} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad \mathbf{a}_2 = a_{12} \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \quad \mathbf{a}_3 = a_{13} \begin{pmatrix} 1 \\ -\frac{2m}{M} \\ 1 \end{pmatrix},$$

Eigenvalues and Eigenvectors

- “triatomic” python notebook