Mechalculus

William F. Barnes

December 11, 2020

Contents

Introduction

While machines have no intuition for abstract thinking or problem solving, computers compensate by having a superhuman attention span and dauntless rapidity for performing basic calculations. For a computer to be useful, the user must therefore cast a real-world (or otherwise interesting) problem into a model that can be solved in steps.

1 Systems of Equations

1.1 Equations and Unknowns

An *equation* is any mathematical statement with an equal sign (=). Equations can contain numbers, functions, operations, and most importantly, *unknown variables*. A simple 'system' of one equation containing one unknown variable can typically be solved, which entails producing a value for the unknown variable that doesn't lead to any false statements. Whenever it's possible for equation can be solved, it is said to be *determined*.

Two or more equations can be considered together as a *system of equations*. For a system of equations to be determined, the number of equations and the number of unknown variables must be equal. If there are more unknowns than equations, the system is called *under-determined*. On the other hand, when there are too many equations and not enough unknowns, the system is *over-determined*.

Transcendental Equations

An equation or system that is determined but cannot be solved analytically is a *transcendental* equation. For instance, the equation

$$\cos(x) - x^2 = \frac{1}{2}$$

cannot easily be solved for x. Some kind of approximation must take place to produce a value for x, which happens to be $\approx \pm 0.580037$.

1.2 Linear Systems

A *linear* system of equations requires all variables to occur only as single powers, i.e., no terms like x^2, $\cos x$, etc. For instance, the two equations containing two unknowns x, y is a linear system:

$$2x + 3y = 19$$
$$4x - y = 3$$

To solve the system, we might solve the second equation for y, giving $y = 4x - 3$, and then insert this into the first equation reduce the problem to one equation and one unknown

$$2x + 3(4x - 3) = 19\ ,$$

easily solved by $x = 2$. Inserting $x = 2$ into either equation in the system, we quickly find $y = 5$ to complete the solution.

1.3 Nonlinear Systems

A *nonlinear* system of equations allows unknown variables to occur at powers other than one. To modify the above system to contain a nonlinearity, we might write:

$$2x^2 + 3y = 19$$

$$4x - y = 3$$

Analytic Solution

To combine the two equations, multiply the second by a factor of 3 and add each to eliminate y altogether (after simplifying):

$$x^2 + 6x = 14$$

The problem is reduced to one equation containing one unknown, solved by the quadratic formula:

$$x = -3 \pm \sqrt{23}$$

Pursuing both channels, we find two values for x:

$$x_1 \approx 1.79583152331272 \qquad\qquad x_2 \approx -7.79583152331272$$

When combining the two equations to eliminate x instead, we get a nonlinear equation for y, namely

$$y^2 + 30y = 143\,,$$

solved simultaneously (using the quadratic formula again) by

$$y_1 \approx +4.183326093250878 \qquad\qquad y_2 \approx -34.18332609325088\,.$$

With multiple values for both x and y, the actual solution so the problem is any *two* of

$$(x_1, y_1) \qquad (x_1, y_2) \qquad (x_2, y_1) \qquad (x_2, y_2)\ .$$

Trying each solution in either of the original equations, we conclude the system is solved by:

$$(1.7953, 4.18333) \qquad\qquad (-7.79583, -34.1833)$$

Graphical Solution

A quick and dirty way to solve a system equations is to plot each equation and look for intersections. Using the example on hand, we solve each equation for y to get

$$y_1(x) = \frac{19 - 2x^2}{3} \qquad\qquad y_2(x) = 4x - 3\,,$$

which produce an upside-down parabola and a straight line, intersecting at two points when plotted.

The graphical solution also works for transcendental equations. Clearly though, analytical and graphical solutions entail plenty of human busywork, and things only get worse from here. If there is any hope of solving equations by an automated process, stronger notation must be developed.

2 Matrix Notation

When dealing with systems of equations, the algebra can get stiflingly messy, necessitating the need for stronger notation. To help motivate the our steps, we'll use an example linear system of three equations and three unknowns:

$$\begin{aligned} x + y + z &= 5 \\ 2x + 3y + 5z &= 8 \\ 4x + 5z &= 2 \end{aligned}$$

To solve for each unknown, one *could* proceed by solving the last equation for x and inserting the result into either of the first two equations, and then eliminate x altogether between them. This can be continued to eliminate y, and ultimately z can be isolated, reducing the unknowns to two. This may be repeated until all unknowns are solved for.

While the above process works, it takes little to imagine that a better notation should be used if we're to tackle larger and more complicated systems.

2.1 Vector

A *vector* is an ordered list of numbers or variables condensed into one mathematical object. For instance, the sequence of numbers 2, 5, and 7 can be written $\vec{V} = \langle 2, 5, 7 \rangle$, where V is the label for the vector, specially marked by an arrow ($\vec{}$). (Textbooks often denote vectors using bold letters instead of arrows. This is merely a notational convention.)

The individual members (or elements) of a vector are called *components*, and the total number of components is called the vector's *dimension*. The order in which the components of a vector are listed *does* matter. For example, the three-dimensional vector $\vec{V} = \langle 2, 5, 7 \rangle$ is not equivalent to the reversed version $\langle 7, 5, 2 \rangle$.

Any individual component is represented by the vector's symbol without the arrow, but including an index subscript. For instance, we could represent $\vec{V}$ as $\vec{V} = \langle V_a, V_b, V_c \rangle$ with $V_a = 2$, $V_b = 5$, $V_c = 7$, but the letters a, b, c could easily have been x, y, z, or perhaps i, j, k. Vector component labels are, after the dust settles, purely for bookkeeping.

2.2 Matrix

A *matrix* may be regarded as a 'vector of vectors'. That is, a matrix is a 'block' of numbers arranged in rows and columns, therefore requiring two subscripts. A matrix labeled A can generally have m rows and n columns

$$A_{ij} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1n} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{2n} \\ A_{31} & A_{32} & A_{33} & \cdots & A_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ A_{m1} & A_{m2} & A_{m3} & \cdots & A_{mn} \end{bmatrix},$$

where the indices i, j denote any one particular component in the matrix.

Matrix Multiplication

Two matrices A and B lend themselves to addition and subtraction if and only if they have the same number of rows and columns. Two matrices can be multiplied if and only if the number of columns in the first matrix equals the number of rows in the second matrix. Naturally, matrix multiplication is generally not a commutative process, that is $AB \neq BA$.

If we have a matrix A_{mn} with m rows and n columns, and similarly a matrix B_{np}, the resultant matrix $C_{mp} = A_{mn}B_{np}$ has m rows ad p columns, with components given by:

$$C_{ij} = \sum_{k=1}^{n} A_{ik}B_{jk} \qquad\qquad i = 1, \ldots, m \qquad\qquad j = 1, \ldots, p$$

Matrix as an Operator

A matrix can 'act' or 'operate' on a vector to produce a new vector. This is a special case of matrix multiplication in where the second matrix has only one column. Such an operation is denoted

$$A\vec{x} = \vec{b}\,,$$

where A is a matrix that operates on a vector $\vec{x}$, resulting in vector $\vec{b}$. Taking a common example where $\vec{x}$ is a three-dimensional vector, the $A\vec{x} = \vec{b}$ calculation looks like

$$A\vec{x} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + A_{13}x_3 \\ A_{21}x_1 + A_{22}x_2 + A_{33}x_3 \\ A_{31}x_1 + A_{32}x_2 + A_{33}x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \vec{b}\,.$$

Identity Matrix

One natural question implied by the $A\vec{x} = \vec{b}$ operation is, what form must A have in order to leave $\vec{x}$ unchanged? That is, what is the equivalent to 'multiplying by one' in matrix notation? The task is achieved using *identity* matrix, which must be square, containing only factors of one along the diagonal. The identity matrix exists in any number of dimensions, with common examples being

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad\qquad I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\,.$$

Inverse Matrix

While addition, subtraction, and multiplication have a role in matrix notation, the notion of division is undefined, and we instead deal with the *inverse* of a matrix. If we have a matrix A, the inverse $\bar{A}$ is defined such that

$$A\bar{A} = I\,.$$

That is, whatever matrix $\bar{A}$ that can be multiplied into A to result in the identity matrix must be the inverse. Using the above, we easily show that the operation $A\bar{A}$ commutes by

multiplying A across the equation:

$$\begin{aligned} A\bar{A} &= I \\ A\left(\bar{A}A\right) &= AI \\ \bar{A}A &= I \end{aligned}$$

Not every matrix has an attainable inverse - it may happen that division by zero occurs while attempting to find the inverse in certain cases. What's worse is that if A and B each have a well-defined inverse, the sum $A + B$ may not. While a general formula is harder to work out, the inverse of a two-dimensional matrix can always be calculated using

$$\bar{A} = \frac{1}{A_{11}A_{22} - A_{12}A_{21}} \begin{bmatrix} A_{22} & -A_{12} \\ -A_{21} & A_{11} \end{bmatrix} .$$

Solving Matrix Equations

The solution to $A\vec{x} = \vec{b}$ is trivial to to write by exploiting the inverse of A:

$$\begin{aligned} A\vec{x} &= \vec{b} \\ \left(\bar{A}A\right)\vec{x} &= \bar{A}\vec{b} \\ \vec{x} &= \bar{A}\vec{b} \end{aligned}$$

One subtlety is the inverse $\bar{A}$ need not be found explicitly, rather we simply need the result of it operating on $\vec{b}$.

2.3 Equations as a Matrix

Let us return to a typical three-dimensional linear system of equations:

$$\begin{aligned} x + y + z &= 5 \\ 2x + 3y + 5z &= 8 \\ 4x + 5z &= 2 \end{aligned}$$

In matrix notation, note that the right side of each equation may be arranged in a vector $\vec{b} = (5, 8, 2)$. On the left, we have a mixture of coefficients multiplied into x, y, z, which is indeed a coefficient matrix A operating on a vector $\vec{x} = (x, y, z)$ as follows:

$$A\vec{x} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & 5 \\ 4 & 0 & 5 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \\ 2 \end{bmatrix} = \vec{b} .$$

Augmented Matrix

As a notational convenience, the $A\vec{x} = \vec{b}$ construction can be condensed to a single matrix, which contains but doesn't explicitly write each equation, called the *augmented* matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 5 \\ 2 & 3 & 5 & 8 \\ 4 & 0 & 5 & 2 \end{bmatrix}$$

A linear system of N equations containing N unknowns is represented by an augmented matrix $B_{n,n+1}$.

Matrix Manipulation

A careful practitioner of algebra knows that equations may be manipulated by only two processes: multiply by one, or add zero. Anything else will modify the equations and lead to nonsense. Here we develop the tools to manipulate equations that are already embedded in matrix notation. Without violating the rules of algebra, the following operations can take place on the augmented matrix:

- Exchange two rows.
- Multiply a row by a (nonzero) constant.
- Replace a row by the sum of itself and another row.

If the proper operations are performed on the $A\vec{x} = \vec{b}$ equation, the system is solved when the form $I\vec{x} = \bar{A}\vec{b}$ is attained:

$$I\vec{x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \bar{A} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \vec{x}\,.$$

In augmented matrix notation, the above results emerges in *row-reduced echelon form*, also known as the *RREF*

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \end{bmatrix}\,,$$

where x, y, z solves the system of equations.

2.4 Gaussian Elimination

Knowing what a solution should look like, we finally address how to actually attain it using the allowed operations. The process that carries the unsolved system $A\vec{x} = \vec{b}$ into the solved system $\vec{x} = \bar{A}\vec{b}$ is called *Gaussian elimination*.

Denoting the rows of our example augmented matrix as R_j, the first three operations may go as follows: (i) Subtract $2R_1$ from R_2. (ii) Subtract $4R_1$ from R_3. (iii) Add $4R_2$ to R_3.

$$\begin{bmatrix} 1 & 1 & 1 & 5 \\ 2 & 3 & 5 & 8 \\ 4 & 0 & 5 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 4 & 0 & 5 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & -4 & 1 & -18 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 13 & -26 \end{bmatrix}$$

Note the new matrix has zeros down and left of the diagonal, called the *upper triangular form*. Continuing the elimination process: (i) Divide R_3 by 13. (ii) Subtract $3R_3$ from R_2. (iii) Subtract R_3 from R_1. (iv) Subtract R_2 from R_1.

$$\begin{bmatrix} 1 & 1 & 1 & 5 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 1 & -2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 & 5 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & -2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 0 & 7 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & -2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & -2 \end{bmatrix}$$

Reading off the right-hand column, we immediately see the solution to the system of equations is

$$x = 3 \qquad y = 4 \qquad z = -2\,.$$

3 Newton's Method

Much of what we ask computers to do is generate numerical approximations to functions $f(x)$. For continuous functions of a real variable, nearly all methods of numerical approximation stem from the definition of the derivative (rise-over-run) $f'(x)$, namely

$$\frac{d}{dx} f(x) = f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}\,.$$

The path of analytic calculus calls for h to arbitrarily approach zero, however a certain trick devised by Newton allows us to continue with h remaining finite.

3.1 One Dimensional Newton's Method

Starting with the derivative formula and making the change of notation $h = \Delta x$, where $\Delta x = x - x_0$, the first-order derivative equation looks like

$$f'(x_0) \approx \frac{f(x) - f(x_0)}{x - x_0}\,,$$

which can be rearranged into the equation of a line

$$f(x) \approx f'(x_0)(x - x_0) + f(x_0)\,.$$

Newton's method for solving equations $f(x)$ goes as follows: guess whatever x^* might be, and call that value x_0. Insert x_0 into the above and denote $x \to x_1$. Let $f(x_1) = 0$, and it *should* be true (for a good initial guess) that x_1 is closer to x^* than the original guess x_0. This can be repeated: feed x_1 into the above equation to generate x_2, which should be still closer to x^*, and so on. Solving for a the general x_{n+1} in terms of previous values, it follows that

$$x_{n+1} \approx x_n - \frac{f(x_n)}{f'(x_n)}\,,$$

where in the infinite-n limit, the x_n converge:

$$x_n \lim_{n \to \infty} = x^*$$

Examples

Square Roots

Think fast: what is $\sqrt{8}$? It's not a perfect square, so maybe 2.5? The middle school method for getting the answer is to check if 2.5^2 comes close to 8 on a calculator, and then perhaps increase the guess by smaller amounts until the answer is close enough.

We can do much better using Newton's method by letting

$$f(x) = x^2 - 8\,,$$

which is the equation of a shifted parabola, shown below, solved by $(x^*)^2 = 8$, making $f(x^*) = 0$.

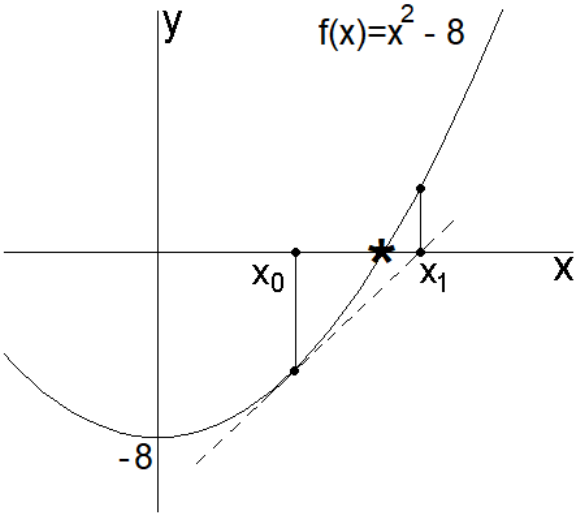

Figure 1: Parabola with initial guess and refined guess indicated.

To proceed we will need the analytical derivative of $f(x) = x^2$, which is simply

$$f'(x) = 2x\,.$$

The iterative formula for x_{n+1} for this problem is thus

$$x_{n+1} = x_n - \frac{x_n^2 - 8}{2x_n}\,.$$

Using $x_0 = 2.5$ as a guess, the result for x_1 reads

$$x_1 = 2.5 - \frac{2.5^2 - 8}{2 \cdot 2.5} = 2.85\,,$$

which is much closer to $\sqrt{8}$ than 2.5. Clearly this process can be repeated using 2.85 as a guess to produce x_2, and so on until the calculator settles on an answer. Continuing this using a standard calculator, the successive x_n come out to:

n	x_n
0	2.5
1	2.85
2	2.82850877193
3	2.82842712592
4	2.82842712475
5	2.82842712475

The answer stops changing (to TI-89 device precision) after the fourth iteration.

Digits of Pi

On a scientific calculator set to radians, the input

$$3.14 - \tan(3.14)$$

results in

$$3.14159265\cdots .$$

Somehow, the combination $x - \tan x$ took the crude guess $\pi \approx 3.14$ and refined it to six additional decimal places of π.

This is explained by Newton's method, for if we seek a function $f(x)$ where some value $x^* = \pi$ sets the function to zero, notice that $\sin(x)$ works perfectly. Using the derivative

$$\frac{d}{dx}\sin x = \cos x ,$$

the iterative formula for x_{n+1} becomes

$$x_{n+1} = x_n - \frac{\sin x_n}{\cos x_n} = x_n - \tan x_n ,$$

where inserting $x_0 = 3.14$ reproduces the first result.

Cosine and Sine Tables

The pair of derivative equations for the sine and cosine

$$\cos x = \frac{d}{dx}\sin x = \lim_{\Delta x \to 0} \frac{\sin(x + \Delta x) - \sin x}{\Delta x}$$

$$-\sin x = \frac{d}{dx}\cos x = \lim_{\Delta x \to 0} \frac{\cos(x + \Delta x) - \cos x}{\Delta x} ,$$

their values for any x can be approximated from based on cruder guesses, i.e. a table limited to integer values. Note of course x and Δx must occur in radians, not degrees.

Solving for $\cos(x + \Delta x)$ and $\sin(x + \Delta x)$ respectively, we find

$$\cos(x + \Delta x) = \cos x - \Delta x \sin x$$

$$\sin(x + \Delta x) = \sin x + \Delta x \cos x .$$

Using these, we can re-construct any cosine or sine value reported on a scientific calculator (to at least ten digits of accuracy) using only one-degree intervals between known x-values.

3.2 Multi-Dimensional Newton's Method

Newton's method works in one dimension because we supply one equation of one variable, namely $f(x)$, containing one unknown x^* that solves $f(x^*) = 0$. Here we expand Newton's method to solve cases with N equations of N variables solved by N unknowns. Denoting each

equation F_i in terms of each variable x^j, we take a first-order partial derivative expansion of $F_i\left(x^j\right)$ as a starting point:

$$F_i\left(x_j\right) \approx F_i\left(x^{j_0}\right) + \sum_{j=1}^{N} \left.\frac{\partial F_i}{\partial x^j}\right|_{x^{j_0}} \Delta x^j \qquad\qquad i = 1, \ldots, N$$

Introducing vector notation

$$\vec{F} = \left(F_1, F_2, \ldots F_N\right) \qquad\qquad \vec{x} = \left(x^1, x^2, \ldots x^N\right) \ ,$$

along with the Jacobian matrix

$$J = J_{ij} = \left.\frac{\partial F_i}{\partial x^j}\right|_{\vec{x}_0} \ ,$$

the above is written

$$\vec{F}\left(\vec{x}\right) \approx \vec{F}\left(\vec{x}_0\right) + J\Delta\vec{x} \ ,$$

where $\Delta\vec{x} = \vec{x} - \vec{x}_0$.

In analogy with the one-dimensional case, Newton's method in N dimension begins with an initial guess vector $\vec{x}_0$. Then, setting $\vec{F}\left(\vec{x}\right) = 0$ and denoting $\vec{x} \to \vec{x}_1$, we have

$$0 \approx \vec{F}\left(\vec{x}_0\right) + J\left(\vec{x}_1 - \vec{x}_0\right) \ ,$$

which must be solved for $\vec{x}_1$ in terms of $\vec{x}_0$ and applied iteratively.

Inverse Jacobian Approach

One way to proceed involves the inverse Jacobian matrix $\bar{J}$ such that $\bar{J}J = J\bar{J} = I$, were I is the diagonal identity matrix of dimension N. Distributing $\bar{J}$ into the above, and generalizing to any step n, we get

$$\vec{x}_{n+1} = \vec{x}_n - \bar{J}\vec{F}\left(\vec{x}_n\right) \ ,$$

which is enough to solve the system iteratively. Of course, this method relies on calculating the inverse of a matrix, which may not always be possible. For two-dimensional cases though, the inverse is trivial:

$$\bar{J} = \frac{1}{J_{11}J_{22} - J_{12}J_{21}} \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix}$$

RREF Approach

A varied approach to the solution avoids computing the inverse of the Jacobian matrix. Starting with $0 \approx \vec{F}\left(\vec{x}_0\right) + \left(\vec{x}_1 - \vec{x}_0\right)J$ put all $\vec{x}_1$-terms on the left, and all $\vec{x}_0$-terms on the right. Generalizing for any n, we have

$$J\vec{x}_{n+1} = J\vec{x}_n - \vec{F}\left(\vec{x}_n\right) \ ,$$

where the right side is completely determined, resolving to a vector we'll call $\vec{b}_n$.

The calculation

$$J\vec{x}_{n+1} = \vec{b}_n$$

can be expressed as an augmented matrix of width $N+1$, namely

$$\begin{bmatrix} J_{11} & J_{12} & J_{13} & \cdots & J_{1N} & b^1 \\ J_{21} & J_{22} & J_{23} & \cdots & J_{2N} & b^2 \\ J_{31} & J_{32} & J_{33} & \cdots & J_{3N} & b^3 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ J_{N1} & J_{N2} & J_{N3} & \cdots & J_{NN} & b^N \end{bmatrix} ,$$

where the solution to the system is found by attaining the row-reduced echelon form (RREF) matrix by Gaussian elimination, resulting in

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & x^1_{n+1} \\ 0 & 1 & 0 & \cdots & 0 & x^2_{n+1} \\ 0 & 0 & 1 & \cdots & 0 & x^3_{n+1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 & x^N_{n+1} \end{bmatrix} = I\vec{x}_{n+1} \, .$$

The refined guess vector $\vec{x}_{n+1}$ is picked out from the right-most column of the RREF matrix.

Examples

Let us solve the system of equations

$$2x^2 + 3y = 1$$

$$4x - 7y = 3$$

with $x_0 = -1$, $y_0 = -1$ as an initial guess. We pave a path to the solution by letting

$$F_1(x,y) = 2x^2 + 3y - 1 = 0 \qquad\qquad F_2(x,y) = 4x - 7y - 3 = 0 \, ,$$

so that the Jacobian matrix is

$$J_{11} = 4x \qquad J_{12} = 3 \qquad J_{21} = 4 \qquad J_{22} = -7 \, ,$$

or in matrix form,

$$J = \begin{bmatrix} 4x & 3 \\ 4 & -7 \end{bmatrix} .$$

The presence of an x-term in the Jacobian matrix tells us the system is non-linear, and must be solved by iteration. For special cases when J contains only numbers, corresponding to strictly linear systems, the solution emerges from just *one* iteration of the RREF calculation.

Solution via Inverse

Being a two-dimensional matrix, the inverse $\bar{J}$ is readily calculated using the formula above, giving

$$\bar{J} = \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix} = \frac{1}{-28x - 12} \begin{bmatrix} -7 & -3 \\ -4 & 4x \end{bmatrix} .$$

Next, we take the guess vector $\vec{x}_0 = (-1, -1)$ and crank it through the iterative formula

$$\vec{x}_{n+1} = \vec{x}_n - \bar{J}\vec{F}\left(\vec{x}_n\right)$$

to produce $\vec{x}_1$, namely

$$\vec{x}_1 = \vec{x}_0 - \frac{1}{-28x_0 - 12}\begin{bmatrix} -7 & -3 \\ -4 & 4x_0 \end{bmatrix}\begin{bmatrix} 2x_0^2 + 3y_0 - 1 \\ 4x_0 - 7y_0 - 3 \end{bmatrix} = \begin{bmatrix} -1.875 \\ -1.5 \end{bmatrix} .$$

Sending $\vec{x}_1 = (-1.875, -1.5)$ through the same same formula gives $\vec{x}_2 = (-1.610, -1.349)$. A third iteration should produce $\vec{x}_3 = (-1.580, -1.331)$, and so on until the results converge.

Solution via RREF

Proceeding instead with the formula $J\vec{x}_{n+1} = \vec{b}_n$, we supply the initial guess vector $\vec{x}_0 = (-1, -1)$ to set up an augmented matrix

$$\begin{bmatrix} -4 & 3 & b^1(-1,-1) \\ 4 & 7 & b^2(-1,-1) \end{bmatrix} = \begin{bmatrix} -4 & 3 & -3 \\ 4 & 7 & -3 \end{bmatrix} ,$$

which contains the solution to $\vec{x}_1$. Calculating the RREF matrix, we find

$$\begin{bmatrix} -4 & 3 & -3 \\ 4 & 7 & -3 \end{bmatrix} \qquad \rightarrow \qquad \begin{bmatrix} 1 & 0 & -1.875 \\ 0 & 1 & -1.5 \end{bmatrix} ,$$

indicating $\vec{x}_1 = (-1.875, -1.5)$, the same $\vec{x}_1$ calculated previously. Clearly the inverse approach and the RREF approach will converge to the same answer.

3.3 Second Order Newton's Method

It's possible to improve the convergence rate in Newton's method by including higher-order terms in the approximation. Writing the order-two Taylor series for a general function $f(x)$ based at x_0, we begin with

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2 ,$$

which is parabolic rather than linear.

Proceeding by analogy to the first-order case, we let $f(x) = 0$ and solve for $x \rightarrow x_1$ using the quadratic formula to get

$$x_1 \approx x_0 - \frac{f'(x_0)}{f''(x_0)} \pm \frac{f'(x_0)}{f''(x_o)}\sqrt{1 - \frac{2f(x_0)f''(x_0)}{(f'(x_0))^2}} .$$

The above formula will converge quickly for one-dimensional problems.

We can avoid the square root calculation by noting that $f(x_0)$ becomes vanishingly small near the solution, so the square-root term may be expanded according to

$$\sqrt{1 - z} \approx 1 - \frac{z}{2} - \frac{z^2}{8} - \cdots \qquad\qquad z \ll 1 .$$

Simplifying the above equation (choosing the positive root), arrive at

$$x_1 \approx x_0 - \frac{f(x_0)}{f'(x_0)} - \frac{1}{2}\frac{(f(x_0))^2 f''(x_0)}{(f'(x_0))^3},$$

which generalizes to any x_n:

$$x_{n+1} \approx x_n - \frac{f(x_n)}{f'(x_n)}\left(1 + \frac{f(x_n) f''(x_n)}{(f'(x_n))^2}\right)$$

Perhaps not surprisingly, the second-order approximation for x_{n+1} equals the first-order approximation plus a small correction based on the second derivative.

Examples

Square Roots

Returning to the previous square root example, we started with the equation of a shifted parabola

$$f(x) = x^2 - 8,$$

solved by $(x^*)^2 = 8$, making $f(x^*) = 0$. The first and second derivatives of $f(x)$ read

$$f'(x) = 2x \qquad\qquad f'(x) = 2,$$

telling us the recursive formula for x_{n+1} is

$$x_{n+1} \approx x_n - \frac{x_n^2 - 8}{2x_n}\left(1 + \frac{(x_n^2 - 8)}{2x_n^2}\right).$$

Starting with a guess of $x_0 = 2.5$ for consistency, the evolution of x_n goes as:

n	x_n
0	2.5
1	2.801
2	2.82829020666
3	2.82842712143
4	2.82842712475
5	2.82842712475

Repeating the calculation without the $\sqrt{1-z}$ approximation, the iterative formula simplifies to

$$x_{n+1} \approx x_n\sqrt{1 - \frac{(x_n^2 - 8)}{x_n^2}} = \sqrt{8}.$$

Evidently, a second-order approximation of the order-two equation $f(x) = x^2 - 8$ doesn't help - it just tells us to calculate $\sqrt{8}$ - the same question we're asking of the method!

Cube Roots

Modifying $f(x)$ to calculate cube roots, we begin with

$$f(x) = x^3 - a\,,$$

solved by $(x^*)^3 = a$, making $f(x^*) = 0$. The first and second derivatives of $f(x)$ read

$$f'(x) = 3x^2 \qquad\qquad f'(x) = 6x\,,$$

telling us the recursive formula for x_{n+1} is

$$x_{n+1} \approx x_n - \frac{x_n^3 - a}{3x_n^2}\left(1 + \frac{2\,(x_n^3 - a)}{3x_n^3}\right)\,,$$

which works for any guess x_0 reasonably near a.

4 Area Under a Curve

A main concern in calculus is evaluation of the definite integral

$$I = \int_{x_i}^{x_f} f(x)\; dx\,,$$

which returns the area trapped between the x-axis and the function $y = f(x)$ on the interval $x_i \leq x \leq x_f$. Considerations from symmetry can simplify the integral calculation, as a symmetric function obeying $f(-x) = f(x)$ gives

$$I = \int_{-a}^{a} f(x)\; dx = 2\int_{0}^{a} f(x)\; dx \qquad\qquad f(x)\;\text{ symmetric}$$

on the interval $-a \leq x \leq a$. Meanwhile, an antisymmetric function obeying $f(-x) = -f(x)$ on the same interval gives

$$I = \int_{-a}^{a} f(x)\; dx = 0 \qquad\qquad f(x)\;\text{ antisymmetric}\,.$$

Analytic solutions to integrals have been studied immensely, where most 'important' calculations (in physics, for instance) have readily-attainable solutions. Despite this, many integrals have no easily-attainable analytic solution, and must be instead approximated by some means in many small steps.

4.1 Integral as a Discrete Sum

Left Rectangle Rule

Reverse-engineering the definition of the definite integral, the area under a curve $f(x)$ can be approximated as the sum of many thin rectangular areas. The integration domain $x_f - x_i$ is divided into N intervals of constant width Δx. At the jth interval, the x-position is

$$x_j = x_i + (x_f - x_i)\,\frac{j}{N} = x_i + j \cdot \Delta x \qquad\qquad j = 0, \ldots, N\,,$$

and the height of the curve is $f(x_j)$. It follows that the area of the jth rectangle is

$$f(x_j) \cdot \Delta x = f(x_j) \cdot (x_{j+1} - x_j) \ ,$$

and the sum of all rectangles gives the approximate total area:

$$I_L(N) = \sum_{j=0}^{N-1} f(x_j) \, \Delta x$$

As pseudo-code, the above can be implemented by

```
FOR (j = xi TO (xf - dx) STEP dx) {
  sum += f(x) * dx
}
```

In the above, it is assumed x_i, x_f, dx, and $f(x)$ are defined. Since a machine can't take dx to a true infinitesimal limit, if turns out that the *left rectangle rule* just derived under-counts the true area when the slope of $f(x)$ is positive, but over-counts when the slope is negative.

Right Rectangle Rule

A modified rectangle rule entails evaluating $f(x)$ at the $j+1$th position, namely

$$I_R(N) = \sum_{j=0}^{N-1} f(x_{j+1}) \, \Delta x \ ,$$

which could be implemented in the following pseudo-code:

```
FOR (j = xi TO (xf - dx) STEP dx) {
  sum += f(x + dx) * dx
}
```

Of course, the so-called *right rectangle rule* is hardly an improvement of its left-sided counterpart. These are in fact the crudest of integration methods and should only be considered crudely approximate.

Midpoint Rule

A compromise between the left- and right rectangle rules is the midpoint rule, which evaluates $f(x_m)$ at the midpoint between x-values:

$$x_m = \frac{x_j + x_{j+1}}{2}$$

The sum becomes

$$I_R(N) = \sum_{j=0}^{N-1} f(x_m) \, \Delta x$$

4.2 Non-Rectangular Rules

Trapezoid Rule

An improvement over rectangle-based methods is the average the left- and right rules, which effectively turns rectangles into trapezoids, giving (you guessed it) the *trapezoid rule*:

$$\begin{aligned}
I_T(N) &= \frac{1}{2}\left(I_L(N) + I_R(N)\right) \\
I_T(N) &= \sum_{j=0}^{N-1} \frac{1}{2}\left(f(x_j) + f(x_{j+1})\right)\Delta x \\
I_T(N) &= \frac{1}{2}\left(f(x_0) + 2f(x_1) + f(x_2) + \cdots + f(x_N)\right)\Delta x \\
I_T(N) &= \frac{1}{2}\left(f(x_0) + f(x_N)\right)\Delta x + \sum_{j=1}^{N-1} f(x_j)\,\Delta x\,,
\end{aligned}$$

with corresponding pseudo-code:

```
FOR (j = xi TO (xf - dx) STEP dx) {
  sum += .5 * (f(x) + f(x + dx)) * dx
}
```

Simpson's Rule

An improvement over straight-line methods instead uses a quadratic function to approximate $f(x)$ at each evaluation step, known as *Simpson's rule*. In the most general case, a quadratic function appears as

$$g(x) = Ax^2 + Bx + C\,.$$

To proceed, consider the definite integral of $g(x)$ centered on x_m about an interval $2h$ that approximates the area under $f(x)$

$$\int_{x_0-h}^{x_0+h} f(x)\ dx \approx \int_{x_0-h}^{x_0+h} g(x)\ dx = I_{2h}\,,$$

where calculations are made simpler by shifting the x-axis such that x_0 lies at $x = 0$:

$$I_{2h} = \int_{-h}^{h} g(x)\ dx$$

Inserting the quadratic form of $g(x)$, the above readily evaluates to

$$I_{2h} = \frac{h}{3}\left(2Ah^2 + 2C\right)\ .$$

Meanwhile, let us examine the sum $g(0-h) + 4g(0) + g(0+h)$:

$$\begin{aligned}
g(-h) + 4g(0) + g(h) &= A(-h)^2 + B(-h) + C + 4C + A(h)^2 + B(h) + C \\
g(-h) + 4g(0) + g(h) &= 2Ah^2 + 6C \\
g(-h) + 4g(0) + g(h) &= \frac{h}{3} I_{2h}
\end{aligned}$$

Evidently, the area under the parabola defined on $-h \leq x \leq h$ is equal to the sum $g(-h) + 4g(0) + g(h)$. Un-shifting the x-axis to put x_0 back where it was, we have found

$$\int_{x_0-h}^{x_0+h} g(x)\ dx = \frac{h}{3}\left(g(x_0-h) + 4g(x_0) + g(x_0+h)\right)\ .$$

Repeating this over a wide integration domain, and noting that $h = \Delta x/2$, we write

$$\int_a^b f(x)\ dx \approx I_S(N) = \frac{\Delta x}{6}\sum_{j=0}^{N-1}\left(f(x_j) + 4f(x_m) + f(x_{j+1})\right)$$

$$I_S(N) = \frac{1}{6}\left(I_L(N) + 4I_M(N) + I_R(N)\right)\ .$$

Amazingly, the approximate area under $f(x)$ is a weighted sum of previously-established methods. Of course, the average of the left- and right rectangle rules is the trapezoid rule, so the above is equivalent to

$$I_S(N) = \frac{1}{3}\left(I_T(N) + 2I_M(N)\right)\ ,$$

the weighted sum of the trapezoid rule and the midpoint rule. As pseudo-code, Simpson's rule can be implemented as:

```
FOR (j = (xi + dx/2) TO (xf - dx/2) STEP dx) {
  x = j
  xa = x - dx/2
  xb = x + dx/2
  y = f(x)
  ya = f(xa)
  yb = f(xb)
  sum += (1/6) * (ya + 4*y + yb) * dx
}
```

5 Regression Analysis

Here we develop a general means for finding a best-fit function $y = f(x)$ to a set of data points $\{x_i, y_i\}$ based on a physical or mathematical model, often called *regression analysis*.

5.1 Linear Fit

Starting with a simple example, consider the following (six) data points occurring as ordered pairs:

x_i	0.6	1.8	2.8	3.6	4.2	5.6
y_i	1.6	1.6	2.6	2.0	4.0	3.6

A scatter plot of $\{x_i, y_i\}$ would show a roughly linear arrangement of points, perhaps approximated the equation

$$y = \frac{1}{2}x + 1\ ,$$

which could be discerned using a ruler. Of course, we want to do better than guessing, so begin with the most general form of a line in the xy-plane, namely

$$f(x) = mx + b\,.$$

Note that *some* choice of m and b correspond to a line that passes closer to the group of points $\{x_i, y_i\}$ than any other line.

The task now is to determine m and b based on the data given. To do this, suppose the line $f(x) = mx + b$ is sketched somewhere in the plane. Next, write the square of the vertical displacement from any given y_i (up or down) to $f(x_i)$ for a single point:

$$z_i^2 = \left(f(x_i) - y_i\right)^2\ ,$$

which can be done for all data points:

$$F = \sum_{i=1}^{N} z_i^2 = \sum_{i=1}^{N} \left(mx_i + b - y_i\right)^2$$

In this form, we can guarantee a best-fitting line by finding the proper m and b that minimizes F. Taking the partial derivative with respect to m and b respectively, we have

$$\frac{\partial F}{\partial m} = \sum_{i=1}^{N} x_i \cdot \left(mx_i + b - y_i\right) \qquad\qquad \frac{\partial F}{\partial b} = \sum_{i=1}^{N} \left(mx_i + b - y_i\right)\ ,$$

where setting $\partial F/\partial m = \partial F/\partial b = 0$ we get, after distributing the summation signs

$$0 = m\sum_{i=1}^{N} x_i^2 + b\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} x_i\, y_i \qquad\qquad 0 = m\sum_{i=1}^{N} x_i + b\sum_{i=1}^{N} (1) - \sum_{i=1}^{N} y_i\,.$$

Letting

$$X^2 = \sum_{i=1}^{N} x_i^2 \qquad X = \sum_{i=1}^{N} x_i \qquad Y = \sum_{i=1}^{N} y_i \qquad XY = \sum_{i=1}^{N} x_i\, y_i\,,$$

the above is easily recognized as a system of two equations and two unknowns m and b (the rest are all numbers):

$$0 = mX^2 + bX - XY \qquad\qquad 0 = mX + bN - Y$$

Note that the variables X^2, X, Y, XY don't follow ordinary algebra, that is $X \cdot X \neq X^2$, and so on. Solving for m and b, we find

$$m = \frac{N \cdot XY - X \cdot Y}{N \cdot X^2 - X \cdot X} \qquad\qquad b = \frac{Y \cdot X^2 - X \cdot XY}{N \cdot X^2 - X \cdot X}\,.$$

Evidently, m and b are calculated in one iteration with many sub-steps.

For the example on hand, we have $N = 6$, along with

$$X^2 = 73.4 \qquad X = 18.6 \qquad Y = 15.4 \qquad XY = 55.28\,,$$

finally giving

$$m = \frac{6 \cdot 55.28 - 18.6 \cdot 15.4}{6 \cdot 73.4 - 18.6^2} \approx 0.479 \qquad b = \frac{15.4 \cdot 73.4 - 18.6 \cdot 55.28}{6 \cdot 73.4 - 18.6^2} \approx 1.082\,,$$

corresponding to the best-fit line

$$y = 0.479\,x + 1.082\,.$$

5.2 Exponential Fit

Consider a new set of data points $\{x_i, y_i\}$ that seem to fit an exponential form, namely

$$f\,(x) = A\,e^{mx}\,,$$

containing two adjustable parameters A and m. Of course, it's always possible to proceed by letting $z_i^2 = \left(f\,(x_i) - y_i\right)^2$, but we exploit a trick to reduce the effort. Let us rewrite the parameter A as e^b such that

$$f\,(x) = e^{mx+b}\,,$$

which is just the exponential of a straight line. Taking the natural log of each side and letting $g\,(x) = \ln\left(f\,(x)\right)$, we have

$$g\,(x) = mx + b\,.$$

Evidently then, if we take all points y_i and transform them via

$$y_i \to \ln\left(y_i\right)\,,$$

the data fits to a straight line $g\,(x)$. Specifically, we let

$$X^2 = \sum_{i=1}^{N} x_i^2 \qquad X = \sum_{i=1}^{N} x_i \qquad Y = \sum_{i=1}^{N} \ln y_i \qquad XY = \sum_{i=1}^{N} x_i\,\ln y_i$$

such that the problem is reduced to a system of two equations and two unknowns:

$$\begin{bmatrix} N & X \\ X & X^2 \end{bmatrix} \begin{bmatrix} \ln A \\ m \end{bmatrix} = \begin{bmatrix} Y \\ XY \end{bmatrix}$$

5.3 Polynomial Fit

A highly general approach for best-fitting a curve to N data points $\{x_i, y_i\}$ is to guess a polynomial function of order no higher than N:

$$f\,(x) = A + Bx + Cx^2 + Dx^3 + Ex^4 + \cdots = \sum_{j=1}^{N} A_j\,x^{j-1}$$

Proceeding as previously, define a sum that will be minimized when the function best-approximates the data:

$$F = \sum_{i=1}^{N} z_i^2 = \sum_{i=1}^{N} \left(y_i - f\left(x_i\right)\right)^2$$

Since $f\left(x\right)$ contains N unknown coefficients, we must compute a separate partial derivative with respect to each variable. To minimize F, all partial derivate terms must equal zero. Carrying this out, find

$$\begin{aligned}
\frac{\partial F}{\partial A} &= \sum_{j=1}^{N} \left(y_j - A - Bx_j - Cx_j^2 - \cdots\right) = 0 \\
\frac{\partial F}{\partial B} &= \sum_{j=1}^{N} \left(y_j x_j - Ax_j - Bx_j^2 - Cx_j^3 - \cdots\right) = 0 \\
\frac{\partial F}{\partial C} &= \sum_{j=1}^{N} \left(y_j x_j^2 - Ax_j^2 - Bx_j^3 - Cx_j^4 - \cdots\right) = 0 \\
&\cdots \\
\frac{\partial F}{\partial A_N} &= \sum_{j=1}^{N} x_j^{N-1} \left(y_j - A - Bx_j - Cx_j^2 - \cdots\right) = 0 \,,
\end{aligned}$$

where if we denote

$$X^J = \sum_{i=1}^{N} x_i^J \qquad\qquad Y^J = \sum_{i=1}^{N} y_i^J \qquad\qquad Y^K X^J = \sum_{i=1}^{N} y_i^K x_i^J \,,$$

the above is readily expressed in matrix form

$$\begin{bmatrix}
N & X^1 & X^2 & \cdots & X^{N-1} \\
X^1 & X^2 & X^3 & \cdots & X^N \\
X^2 & X^3 & X^4 & \cdots & X^{N+1} \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
X^{N-1} & X^N & X^{N+1} & \cdots & X^{2(N-1)}
\end{bmatrix}
\begin{bmatrix}
A \\ B \\ C \\ \cdots \\ A_N
\end{bmatrix}
=
\begin{bmatrix}
Y^1 \\ Y^1 X^1 \\ Y^1 X^2 \\ \cdots \\ Y^1 X^{N-1}
\end{bmatrix} .$$

Amazingly, the system has exactly N equations and N unknowns, and can be solved in one pass by Gaussian elimination.

Example

Consider the following data points occurring as ordered pairs:

x_i	0	1	2	3	4	5	6	7	8	9	10
y_i	1	6	17	34	57	86	121	162	209	262	321

Plotting the data alone, it's unclear *which* curve should be optimized to best-fit all points. Supposing we had reason to take $f(x)$ up to order 3, the problem is contained in

$$\begin{bmatrix} N & X^1 & X^2 \\ X^1 & X^2 & X^3 \\ X^2 & X^3 & X^4 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} Y^1 \\ Y^1X^1 \\ Y^1X^2 \end{bmatrix} ,$$

which after evaluating each X- and Y-like term, reads

$$\begin{bmatrix} 11 & 55 & 385 \\ 55 & 385 & 3025 \\ 385 & 3025 & 25333 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 1276 \\ 9900 \\ 82434 \end{bmatrix} .$$

An exercise in Gaussian elimination leads to the solution $A = 1$, $B = 2$, $C = 3$.

6 Derivative Matching

Here we address the concept of fitting a curve $f(x)$ not simply to discrete data points, but to differentiable continua (points on curves). Then, constraints on derivatives of $f(x)$ have a general effect on the way $f(x)$ is optimized.

6.1 A Two-Roads Problem

One example of a simple boundary-value problem begins with two parabolic non-intersecting roads or train tracks given by

$$y_1(x) = -\frac{x^2}{4} - 1 \qquad\qquad y_2(x) = \frac{x^2}{4} + 1 .$$

Next, suppose we are required to find a function $f(x)$ that, in the smoothest way possible, connects the point $(-2, -2)$ on the y_1-curve to the point $(1, 5/4)$ on the y_2-curve.

Let us proceed by guessing that the solution, whatever it is, can be approximated by an N-order polynomial

$$f(x) = A + Bx + Cx^2 + Dx^3 + Ex^4 + \cdots = \sum_{j=1}^{N} A_j \, x^{j-1} ,$$

where some choice of coefficients $\{A_j\}$ contains the correct answer. The most obvious facts we can write about $f(x)$ is it must match $y_1(-2)$ and $y_2(1)$ as given in the problem statement. This means we can generate two equations:

$$\begin{aligned} y_1(-2) = f(-2) = -2 &= A - 2B + 4C - 8D + 16E - \cdots \\ y_2(1) = f(1) = 5/4 &= A + B + C + D + E + \cdots \end{aligned}$$

So far so good, but there are still more unknowns than equations - the system is under-determined without new information.

Looking again at the functions $y_{1,2}(x)$, we can differentiate each twice before reaching zero:

$$\frac{d}{dx}y_1(x) = -\frac{x}{2} \qquad \frac{d}{dx}y_2(x) = \frac{x}{2}$$
$$\frac{d^2}{dx^2}y_1(x) = -\frac{1}{2} \qquad \frac{d^2}{dx^2}y_2(x) = \frac{1}{2}$$

Evaluating each of these at the boundary points $(-2,-2)$ and $(1,5/4)$, we can calculate a number for the slope and the second derivative at each boundary point

$$\frac{d}{dx}y_1(-2) = 1 \qquad \frac{d}{dx}y_2(1) = \frac{1}{2}$$
$$\frac{d^2}{dx^2}y_1(-2) = -\frac{1}{2} \qquad \frac{d^2}{dx^2}y_2(1) = \frac{1}{2},$$

where higher-order derivatives are clearly zero. Meanwhile, we may also take the first and second derivatives of our guess function $f(x)$, giving:

$$\begin{aligned}\frac{d}{dx}f(x) &= B + 2Cx + 3Dx^2 + 4Ex^3 + 5Fx^4 + \cdots \\ &= \sum_{j=2}^{N}(j-1)A_j x^{j-2} \\ \frac{d^2}{dx^2}f(x) &= 2C + 3\cdot 2Dx + 4\cdot 3Ex^2 + 5\cdot 4Fx^3 + 6\cdot 5Gx^4\cdots \\ &= \sum_{j=3}^{N}(j-1)(j-2)A_j x^{j-3}\end{aligned}$$

We already know that the function $f(x)$ matches the curves $y_{1,2}$ at the boundary points. Now comes the crucial observation: the function *and its derivatives* match the curves *and their derivatives* at the boundary points. This means we gain four new equations

$$\begin{aligned}\frac{d}{dx}(y_1(-2)) &= \frac{d}{dx}(f(-2)) = 1 \\ \frac{d}{dx}(y_2(1)) &= \frac{d}{dx}(f(1)) = \frac{1}{2} \\ \frac{d^2}{dx^2}(y_1(-2)) &= \frac{d^2}{dx^2}(f(-2)) = -\frac{1}{2} \\ \frac{d^2}{dx^2}(y_2(1)) &= \frac{d^2}{dx^2}(f(1)) = \frac{1}{2},\end{aligned}$$

where in terms of the unknown coefficients, the above read:

$$\begin{aligned}1 &= B + 2C(-2) + 3D(-2)^2 + 4E(-2)^3 + 4F(-2)^4 + \cdots \\ 1/2 &= B + 2C + 3D + 4E + 4F + \cdots \\ -1/2 &= 2C + 3\cdot 2D(-2) + 4\cdot 3E(-2)^2 + 5\cdot 4F(-2)^3 + 6\cdot 5G(-2)^4 + \cdots \\ 1/2 &= 2C + 3\cdot 2D + 4\cdot 3E + 5\cdot 4F + 6\cdot 5G + \cdots\end{aligned}$$

Interestingly, we now have six total equations, up from two. This should mean if we choose $f(x)$ to contain six unknowns, which amounts to keeping the A- through F-terms, and discarding any G- or higher terms, then the system left is indeed determined, and can be solved exactly. Truncating the series accordingly, all information in the problem is contained in the augmented matrix

$$\begin{bmatrix} 1 & -2 & 4 & -8 & 16 & -32 & -2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 5/4 \\ 0 & 1 & -4 & 12 & -32 & 80 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 1/2 \\ 0 & 0 & 2 & -12 & 48 & -160 & -1/2 \\ 0 & 0 & 2 & 6 & 12 & 20 & 1/2 \end{bmatrix},$$

where Gaussian elimination hands us the solution as right-most column in the RREF matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 43/81 \\ 0 & 1 & 0 & 0 & 0 & 0 & 94/81 \\ 0 & 0 & 1 & 0 & 0 & 0 & -149/324 \\ 0 & 0 & 0 & 1 & 0 & 0 & -23/162 \\ 0 & 0 & 0 & 0 & 1 & 0 & 19/162 \\ 0 & 0 & 0 & 0 & 0 & 1 & 7/162 \end{bmatrix}.$$

Finally, the curve is given by

$$f(x) = \frac{43}{81} + \frac{94}{81}x - \frac{149}{324}x^2 - \frac{23}{162}x^3 + \frac{19}{162}x^4 + \frac{7}{162}x^5 \qquad -2 \le x \le 1$$

as shown.

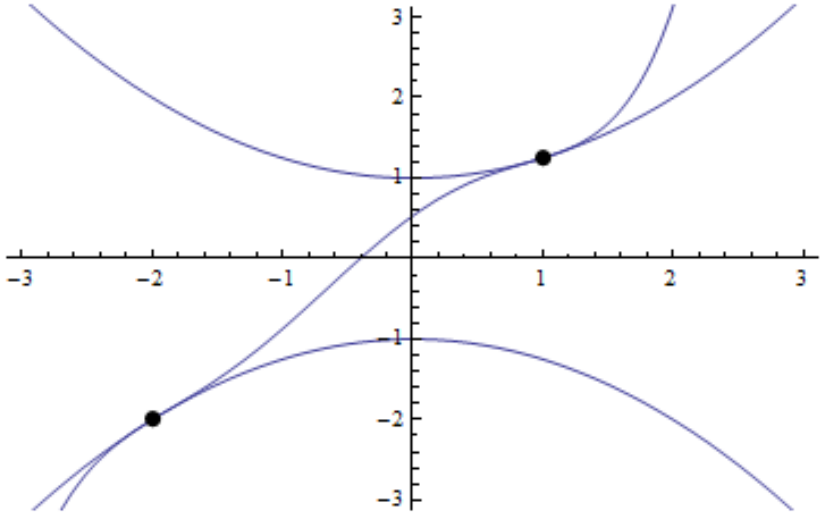

Figure 2: Parabolic roads with connecting curve.

6.2 Generalization

Reviewing the two-roads problem, note that the solution does not depend on the whole curves $y_{1,2}(x)$, but only on the value of the curve and its derivatives *at the boundary points*. It follows that we can abstract away the need for curves, and only require the value and slope information at two boundary points x_1 and x_2. In the general case, the analysis begins with

$$\begin{aligned} y_1(x_1) &= y_1 & y_2(x_2) &= y_2 \\ \frac{d}{dx} y_1(x_1) &= y_1' & \frac{d}{dx} y_2(x_2) &= y_2' \\ \frac{d^2}{dx^2} y_1(x_1) &= y_1'' & \frac{d^2}{dx^2} y_2(x_2) &= y_2'' , \end{aligned}$$

where each of the y-terms and their derivatives are specified by any means.

An N-order polynomial

$$f(x) = A + Bx + Cx^2 + Dx^3 + Ex^4 + Fx^5 + \cdots = \sum_{j=1}^{N} A_j \, x^{j-1}$$

with the proper choice of coefficients $\{A_j\}$ will satisfy the boundary conditions. Since we're sticking with second derivative analysis, it suffices to truncate the series after the x^5-term. (If more derivatives of $y_{1,2}$ are specified, higher-order terms in the series must be maintained.) The first- and second derivative of $f(x)$ read

$$\begin{aligned} f'(x) &= \frac{d}{dx} f(x) = \sum_{j=2}^{N} (j-1) A_j x^{j-2} \\ f''(x) &= \frac{d^2}{dx^2} f(x) = \sum_{j=3}^{N} (j-1)(j-2) A_j x^{j-3} . \end{aligned}$$

Setting f, f', f'' to match the boundary value data, we end up with a six-dimensional augmented matrix equation

$$\left[\begin{array}{ccccccc} 1 & x_1 & x_1^2 & x_1^3 & x_1^4 & x_1^5 & y_1 \\ 1 & x_2 & x_2^2 & x_2^3 & x_2^4 & x_2^5 & y_2 \\ 0 & 1 & 2x_1 & 3x_1^2 & 4x_1^3 & 5x_1^4 & y_1' \\ 0 & 1 & 2x_2 & 3x_2^2 & 4x_2^3 & 5x_2^4 & y_2' \\ 0 & 0 & 2 & 3 \cdot 2x_1 & 4 \cdot 3x_1^2 & 5 \cdot 4x_1^3 & (1/2)\, y_1'' \\ 0 & 0 & 2 & 3 \cdot 2x_2 & 4 \cdot 3x_2^2 & 5 \cdot 4x_2^3 & (1/2)\, y_2'' \end{array}\right] ,$$

which is readily reduced by Gaussian elimination to deliver the unknown coefficients $\{A_j\}$.

7 Interpolation

Here we combine the idea of regression analysis, which concerns fitting one curve to many data points, along with derivative matching, which adds slope considerations to curve fitting.

These form the basis for *interpolation* techniques, which can fit smooth curves to many classes of data. Taking a concrete example, consider the following data points occurring as ordered pairs:

x_i	0	1	2	3	4	5	6	7	8	9	10
y_i	0	0.84	0.91	0.14	-0.76	-0.96	-0.28	0.66	0.99	0.41	-0.54

In this analysis, we visit several ways to to connect each data point to its neighbors.

7.1 Taylor Expansion

Any well-behaved function can be approximated near a base value in terms of its derivatives, known as the Taylor expansion

$$f(x) = f(x_0) + (x - x_0)\frac{d}{dx}f(x_0) + \frac{(x - x_0)^2}{2!}\frac{d^2}{dx^2}f(x_0) + \cdots + \frac{(x - x_0)^n}{n!}\frac{d^n}{dx^n}f(x_0) \ .$$

For small-enough intervals $x - x_0$, most the of the terms on the right side vanish, and the expansion reduces to Newton's first-order derivative equation.

7.2 Rectangle Approximation

The crudest way to map a function $f(x)$ to a set of data points is to let f be equal to $y_i(x_i)$ in the neighborhood enclosing x_i. In such a case, $f(x)$ isn't really a function, as its slope is vertical across dissimilar y_i:

$$f(x_i \leq x < x_{i+1}) = y_i$$

7.3 Connect-the-Dots Approximation

The fancy name for connecting the dots in a data plot is *linear interpolation*. In this case, best-fit curve $f(x)$ is composed of straight line segments. Defining the interval as the space enclosed by x_i and x_{i+1}, it's easy to reason that a straight line connecting points y_i and y_{i+1} is given by

$$f(x) = y_i(x_i) + (x - x_i)\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right) \ ,$$

which is suspiciously like the order-one Taylor expansion term.

7.4 Quadratic Approximation

Supposing we involve three points x_{i-1}, x_i, x_{i+1} at once in each analysis step, the approximating curve $f(x)$ can have a parabolic component, which entails discretizing the second-order derivate term

$$\frac{(x - x_0)^2}{2!}\frac{d^2}{dx^2}f(x_0) \ .$$

The second derivative is simply the slope of the slope, which means

$$\begin{aligned}\frac{d^2}{dx^2}f &= \frac{d}{dx}f' = \frac{f'(x+\Delta x)-f'(x)}{\Delta x} \\ &= \frac{1}{\Delta x}\left(\frac{f(x+2\Delta x)-f(x+\Delta x)-f(x+\Delta x)+f(x-\Delta x)}{\Delta x}\right) \\ &= \frac{1}{\Delta x^2}\left(f(x+2\Delta x)-2f(x+\Delta x)+f(x-\Delta x)\right)\ ,\end{aligned}$$

a weighted sum of nearby f-values. It quickly follows that the second-order correction to the approximation $f(x)$ reads

$$\frac{(x-x_i)^2}{2}\left(\frac{y_{i+1}-2y_i+y_{i-1}}{(x_{i+1}-x_i)^2}\right)$$

such that

$$f(x) = y_i(x_i) + (x-x_i)\left(\frac{y_{i+1}-y_i}{x_{i+1}-x_i}\right) + \frac{(x-x_i)^2}{2}\left(\frac{y_{i+1}-2y_i+y_{i-1}}{(x_{i+1}-x_i)^2}\right)\ .$$

The above equation will surely join each data point $\{x_i, y_i\}$ with a curve, but it is still awkwardly jumpy at each point.

7.5 Spline Calculation

We are able to 'smooth out' the entire curve $f(x)$ through the set of N data points $\{x_i, y_i\}$ by constraining the derivative and the second derivative of f to be continuous across each x_i-boundary. For this, we need at least a cubic approximation to f, which is a special case of our 'generalized two-roads problem' studied previously.

Specifically, we define a function $f_i(x)$ valid on the interval $x_i \le x \le x_{i+1}$ such that

$$f_i(x) = A_i + B_i(x-x_i) + C_i(x-x_i)^2 + D_i(x-x_i)^3\ ,$$

where A_i, B_i, C_i, D_i can be calculated subject to the following continuity conditions

$$\begin{aligned} f_i(x_i) &= y_i & f_{i+1}(x_{i+1}) &= y_{i+1} \\ f_i(x_i) &= f_{i-1}(x_i) & f_i(x_{i+1}) &= f_{i+1}(x_{i+1}) \\ f_i'(x_i) &= f_{i-1}'(x_i) & f_i'(x_{i+1}) &= f_{i+1}'(x_{i+1}) \\ f_i''(x_i) &= f_{i-1}''(x_i) & f_i''(x_{i+1}) &= f_{i+1}''(x_{i+1})\ ,\end{aligned}$$

which are valid for $2 \le i \le N-1$. Most of the left column is in fact redundant to the right column, so let us toss the whole thing.

In total, we have $N-1$ polynomials $f_i(x)$ with $4(N-1)$ unknowns A_i, B_i, C_i, D_i. The constraint information only hands us $4(N-2)$ unique equations though, where remembering

$$f_1(x_1) = y_1 \qquad\qquad f_N(x_N) = y_N$$

takes us down two 'missing' equations. Note that on the boundaries the function and its first derivative are readily available from the data points, however the second derivative is ambiguous. For this reason, we agree to take each f'' term as exactly zero at the boundaries, known as the *natural spline*:

$$f_1''(x_1) = 0 \qquad\qquad f_N''(x_N) = 0$$

Denoting points i and $i+1$ as 1 and 2, respectively, the coefficients A_i, B_i, C_i, D_i at each step are calculated from the augmented matrix

$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & y_1 \\ 1 & x_2 & x_2^2 & x_2^3 & y_2 \\ 0 & 1 & 2x_1 & 3x_1^2 & y_1' \\ 0 & 1 & 2x_2 & 3x_2^2 & y_2' \end{bmatrix} ,$$

straightforwardly solved by Gaussian elimination. Iterating this between all pairs of x-points, the final curve is guaranteed to be smooth. (The spline calculation is essentially the two-roads problem iterated many times.)

8 Euler's Method

Euler's method is essentially Newton's method (first-order Taylor expansion) applied to differential equations. Differential equations arise in a myriad of systems from population models, trajectory calculations, financing, etc.

8.1 Exponential Growth and Decay

Consider a population of bacteria in an environment of unlimited resources (the inside of a rotten sandwich, for instance). If a single cell reproduces by splitting in two after some characteristic time, it follows that the total population is modeled by the differential equation

$$\frac{d}{dt}N(t) = kN(t) \ ,$$

where $N(t)$ is the population at a given time, and k is a constant proportional to the reproduction rate of any (and each) cell. To fully specify a solution, the population N must be known at some time t, typically at $t = 0$.

A similar equation applies to the decay of a solid object by radioactivity. If a given atom has tends to decay over a characteristic time τ, then the number of remaining atoms is governed by

$$\frac{d}{dt}N(t) = -t/\tau \ .$$

Each of the equations above can be solved analytically. In more generic variables, we separate the differential equation and integrate:

$$\frac{dy}{dx} = \alpha y \qquad \rightarrow \qquad \frac{dy}{y} = \alpha\, dx \qquad \rightarrow \qquad \int \frac{dy}{y} = \alpha \int dx$$

$$\ln y = \alpha x + \tilde{A} \qquad \rightarrow \qquad y = A\, e^{\alpha x}$$

For most problems, an analytic solution may be cumbersome or impossible to attain, in which case we turn to approximation methods. Taking a similar starting point

$$\frac{d}{dt} f\left(x_0\right) \approx \frac{f\left(x\right) - f\left(x_0\right)}{x - x_0} \,,$$

solve for $f\left(x\right)$ to get

$$f\left(x\right) \approx f\left(x_0\right) + \left(x - x_0\right) \frac{d}{dx} f\left(x_0\right) \,.$$

Dividing the integration domain into N copies with interval Δx between steps, the evolution of f is calculated by iterative equation

$$f\left(x_{i+1}\right) = f\left(x_i\right) + \Delta x \frac{d}{dx} f\left(x_i\right) \,,$$

where $f\left(x_0\right)$ is given. This is indeed a general construction, which applies to exponential growth and decay as a special case:

$$N\left(t_{i+1}\right) = N\left(t_i\right) + k\, \Delta t\, N\left(t_i\right) \qquad \text{Bacerial growth}$$

$$N\left(t_{i+1}\right) = N\left(t_i\right) - \frac{1}{\tau}\, \Delta t\, N\left(t_i\right) \qquad \text{Radioactive decay}$$

8.2 Forced Motion

The classical trajectory of a particle of mass m is governed by Newton's second law, a second-order differential equation

$$\frac{d^2}{dt^2} \vec{x} = \frac{\vec{F}}{m} \,,$$

which is in fact a system of two first-order equations:

$$\frac{d}{dt} \vec{v} = \frac{\vec{F}}{m} \qquad\qquad \frac{d}{dt} \vec{x} = \vec{v} \,,$$

where $\vec{x}$, $\vec{v}$, and $\vec{F}/m$ are functions of time. In the above, $\vec{F}$ can have more-or-less any form, with $\vec{F} = 0$ corresponding to straight-line motion.

Discretizing the two first-order system, Euler's method hands us two equations to advance the calculation from the ith step:

$$\vec{x}_{i+1} = \vec{x}_i + \vec{v}_i\, \Delta t \qquad\qquad \vec{v}_{i+1} = \vec{v}_i + \frac{\vec{F}_i}{m} \Delta t$$

Solving the above requires both $\vec{x}$ and $\vec{v}$ to be known at a definite time, typically $t_0 = 0$. Of course, the vector sign over each symbol tells us these are multi-component objects, meaning we have a separate equation for each component (x, y, z).

8.3 Applications

Kinematics

For projectile motion near the surface of the Earth, particles are subject to (at least) a uniform gravitational field, pointed strictly downward. The field influences a particle via

$$\frac{\vec{F}_{grav}}{m} = -g\,\hat{z}\,,$$

where g is the local gravitation constant, taken as $9.8\,m/s^2$. To make use of this equation, simply plug $\vec{F}/m$ into the iterative equations for $\vec{v}$ and $\vec{x}$.

Air Resistance and Wind

The effects of wind and air resistance can be 'worked in' as an alteration to the velocity at each step i. Taking a simple model where the resistance on a moving object is proportional to its velocity, we have

$$\vec{F}_{resist} = -b\,\vec{v}\,,$$

where b is a proportionality constant. Like 'real' air resistance, the force is always against the velocity vector, and vanishes for $v = 0$.

To simulate wind in the xy-plane, remove the $\vec{v}$-dependence from the above and write a more general statement, namely

$$\vec{F}_{wind} = w_x\,\hat{x} + w_y\,\hat{y}\,.$$

Uniform wind simply has w_x and w_y being constant. For more exotic wind patterns, let w_x and w_y be functions of space and time, and include a z-component.

Air Drag

A more careful air resistance calculation tells us that drag is proportional to the square of the velocity, as in

$$\vec{F}_{drag} = -\frac{C_d}{2}\rho A v^2\,\hat{v}\,,$$

where ρ is the air density, A is the forward-facing area of the object in motion, and C_d is a constant classifying the surface friction of the object.

Lift Force

An object with sufficient spin velocity is subject to interesting forces while moving in air. It can be shown that the force on a near-spherical object that is spinning about an axis $\hat{\omega}$ perpendicular to both (i) its direction of motion and (ii) the gravitational force vector is

$$\vec{F}_{lift} = \frac{C_l}{2}\rho A v^2 \frac{\vec{\omega}\times\vec{v}}{|\vec{\omega}\times\vec{v}|}\,.$$

In the above, ω is the spinning angular velocity. It actually turns out hat C_l is not always constant, as experimentally the relationship

$$C_l \propto \frac{r\omega}{v}$$

emerges.

Magnus Force

A generalization of the lift force is the so-called *Magnus* force, which permits $\hat{\omega}$ to point anywhere:

$$\vec{F}_{magus} = S_0 \left(\vec{\omega} \times \vec{v}\right)$$

Of course, the trouble is in accurately modeling S_0, which has to be tackled separately for each problem.

Orbital Motion

Several systems in classical physics, with the most famous example being planetary orbits in the gravitational field of the sun, are governed by an inverse-square force law:

$$\frac{\vec{F}_{grav}}{m} = -\frac{GM}{r^2}\,\hat{r}$$

In the above, G is Newton's universal gravitation constant, and M is the mass of the sun, centered at the origin. At a point $\vec{r}$ in space, the force is inversely proportional to r^2 and is purely attractive. Perpendicular to $\hat{r}$ is another unit vector $\hat{\theta}$, which form the basis for polar coordinates:

$$\vec{r} = r\cos\theta\,\hat{x} + r\sin\theta\,\hat{y}$$

As a check on orbital motion simulations, it's useful to verify Kepler's laws for a two-body problem:

- Planetary orbits are closed ellipses with the sun at a focus.
- A line joining the sun and a planet sweeps out equal areas in equal times.
- The square of a planet's orbital period is proportional to the cubs of its semi-major axis.

Sinusoidal Motion

Perhaps the simplest (and most important) force that leads to harmonic oscillatory motion is caused by a Hookean spring, in which case the force is proportional to the displacement from an equilibrium value:

$$\vec{F}_{spring} = -k\left(\vec{x} - \vec{x}_0\right)$$

Oscillatory Motion

A more general oscillatory system is a pendulum suspended in a gravitational field. For displacements in the pendulum angle θ away from its equilibrium value $\theta = 0$ (straight down), standard Newtonian analysis tells us the system is governed by

$$\omega = \frac{d}{dt}\theta\,(t) \qquad\qquad \frac{d}{dt}\omega = -\frac{g}{L}\sin\theta\,,$$

where L is the length of the pendulum, and g is the local gravitation constant. Note the mass m has already canceled. Such a system can be solved analytically, with one interesting result being the period T of potion. In the general case, an initial displacement θ_0 corresponds to a period of

$$T = 2\pi\sqrt{\frac{L}{g}}\sqrt{1 + \left(\frac{1}{2}\right)^2 k^2 + \left(\frac{1\cdot 3}{2\cdot 4}\right)^2 k^4 + \left(\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\right)^2 k^6 + \cdots}\,,$$

where $k = \sin(\theta_0/2)$. Note that small values of θ_0 allow us to ignore the square root term.

8.4 Breakdown of Euler's Method

While it may seem we have attained a means for solving any possible problem, one quickly finds that Euler's method is perhaps the *worst* of the calculus-based approximation methods, as it starts getting things wrong early. Going back to the pair of equations

$$\vec{x}_{i+1} = \vec{x}_i + \vec{v}_i\,\Delta t \qquad\qquad \vec{v}_{i+1} = \vec{v}_i + \frac{\vec{F}_i}{m}\Delta t\,,$$

note that the velocity is always calculated at the point $\vec{x}_i$. This leads to trouble when $\vec{x}$ reaches a critical point where the slope suddenly changes sign. The corresponding $\vec{x}_{i+t}$ will be significantly off-true.

Taking a simple example, consider an object of mass m attached to a spring with Hookean constant k set into oscillatory motion. The iterative equations suggested by Euler's method are

$$q_{i+1} = q_i + \frac{p_i}{m}\Delta t \qquad\qquad p_{i+1} = p_i - q_i k\,\Delta t\,.$$

At the $i+1$th step, the energy of the system must be

$$\begin{aligned}
E_{i+1} &= \frac{k}{2}\left(q_i + \frac{p_i}{m}\Delta t\right)^2 + \frac{1}{2m}\left(p_i - q_i k\,\Delta t\right)^2 \\
E_{i+1} &= \frac{p_i^2}{2m}\left(1 + \frac{k}{m}\Delta t^2\right) + \frac{k}{2}q_i^2\left(1 + \frac{k}{m}\Delta t^2\right) \\
E_{i+1} &= E_i\left(1 + \frac{k}{m}\Delta t^2\right)\,.
\end{aligned}$$

Evidently, the energy fails to remain constant. Even worse, the energy increases exponentially across many iterations.

8.5 Euler's Backward Method

An alternative approach uses Euler's *backward* method, which uses information at the $i+1$th point to calculate the corresponding step:

$$\vec{x}_{i+1} = \vec{x}_i + \vec{v}_{i+1}\,\Delta t \qquad\qquad \vec{v}_{i+1} = \vec{v}_i + \frac{\vec{F}_{i+1}}{m}\Delta t\,.$$

Applying this again to the mass-and-spring oscillator, we write

$$q_{i+1} = q_i + \frac{p_{i+1}}{m}\Delta t \qquad\qquad p_{i+1} = p_i - q_{i+1}k\,\Delta t\,.$$

Of course, one may wonder how $i+1$ terms can exist on the right side of the equation, however the above is a system of two equations and two unknowns, which we can solve exactly:

$$q_{i+1} = \frac{q_i + (p_i/m)\,\Delta t}{(1 + (k/m)\,\Delta t^2)} \qquad\qquad p_{i+1} = \frac{p_i - q_i k\,\Delta t}{(1 + (k/m)\,\Delta t^2)}$$

The resulting equations are equal to those derived in the (ordinary) forward Euler's method, scaled by a factor of $(1 + (k/m)\,\Delta t^2)$. Running these through the same energy calculation above, we quickly discover the energy decreases at each step, which means exponential decay across many iterations:

$$E_{i+1} = \frac{E_i}{(1 + (k/m)\,\Delta t^2)}$$

8.6 Mixed Euler's Method

Almost surely by accident, it turns out that Euler's method *with a typo* correctly advances the motion of the simple pendulum. Recalling the equations of motions for a simple pendulum in a uniform gravitational field, we again have

$$\omega\,(t) = \frac{d}{dt}\theta\,(t) \qquad\qquad \frac{d}{dt}\omega\,(t) = -\frac{g}{L}\sin\theta\,,$$

where θ is the angle of vertical deflection, L is pendulum length, and g is the local gravity constant.

Indeed, the iterative equations

$$q_{i+1} = q_i + \frac{p_i}{m}\Delta t \qquad\qquad p_{i+1} = p_i - \frac{g}{L}\sin\left(q_{i+1}\right)\Delta t$$

predict pendulum motion without accumulating error. This is a so-called 'typo' because it uses Euler's forward method to advance q, and the backward method to advance p. It just happens that their errors cancel out, but it's not obvious (yet) why, or if this will work in the general case.

8.7 Iterative Matrix Equations

We've seen iterative equations often occurring in sets, namely as separate equations for position q and momentum p. In most cases, a system can be reduced to a set of linear equations, which lends itself to matrix notation. Jumping right in, recall three separate integration schemes named after Euler that apply to the harmonic oscillator:

$$\text{Forward:} \qquad q_{i+1} = q_i + \frac{p_i}{m}\Delta t \qquad\qquad p_{i+1} = p_i - q_i k\,\Delta t$$

$$\text{Backward:} \qquad q_{i+1} = \frac{q_i + (p_i/m)\,\Delta t}{(1 + (k/m)\,\Delta t^2)} \qquad\qquad p_{i+1} = \frac{p_i - q_i k\,\Delta t}{(1 + (k/m)\,\Delta t^2)}$$

$$\text{Mixed:} \qquad q_{i+1} = q_i + \frac{p_i}{m}\Delta t \qquad\qquad p_{i+1} = \frac{p_i - q_i k\,\Delta t}{(1 + (k/m)\,\Delta t^2)}$$

As matrix equations, the above are equivalent to:

$$\text{Forward :} \qquad \begin{bmatrix} 1 & \Delta t/m \\ -k\,\Delta t & 1 \end{bmatrix} \begin{bmatrix} q_i \\ p_i \end{bmatrix} = \begin{bmatrix} q_{i+1} \\ p_{i+1} \end{bmatrix}$$

$$\text{Backward :} \qquad \frac{1}{(1 + (k/m)\,\Delta t^2)} \begin{bmatrix} 1 & \Delta t/m \\ -k\,\Delta t & 1 \end{bmatrix} \begin{bmatrix} q_i \\ p_i \end{bmatrix} = \begin{bmatrix} q_{i+1} \\ p_{i+1} \end{bmatrix}$$

$$\text{Mixed :} \qquad \begin{bmatrix} 1 & \Delta t/m \\ -k\,\Delta t & (1 - (k/m)\,\Delta t^2) \end{bmatrix} \begin{bmatrix} q_i \\ p_i \end{bmatrix} = \begin{bmatrix} q_{i+1} \\ p_{i+1} \end{bmatrix} .$$

To bundle up the notation, we can write each of the above as a matrix equation having general form

$$M \begin{bmatrix} q_i \\ p_i \end{bmatrix} = \begin{bmatrix} q_{i+1} \\ p_{i+1} \end{bmatrix} ,$$

where N is a 2×2 matrix when there are two unknown variables.

9 Runge-Kutta Techniques

The The *Runge-Kutta* method (a.k.a. RK method) corrects Euler's method in solving second-order differential equations. In general, RK methods apply to equations of the form

$$\frac{d}{dt} x\,(t) = f\,(x, t)$$

subject to the known initial condition $x_0 = x\,(t = 0)$.

9.1 Midpoint Method

The second-order Runge-Kutta method, also knows as the RK2 method or the *midpoint method*, delivers exact results for certain trajectories, especially projectile motion. In accordance with the Figure shown, Let us take the midpoint $\vec{x}_m$ between $\vec{x}_i$ and $\vec{x}_{i+1}$ such that

$$\vec{x}_m = \vec{x}_i + \frac{1}{2}\vec{v}_i\,\Delta t \qquad\qquad \vec{v}_m = \vec{v}_i + \frac{1}{2}\frac{\vec{F}\left(\vec{x}_i, \vec{v}_i\right)}{m}\,\Delta t\,.$$

Then, to move forward from the ith step, we have

$$\vec{x}_{i+1} = \vec{x}_i + \vec{v}_m\,\Delta t \qquad\qquad \vec{v}_{i+1} = \vec{v}_i + \frac{\vec{F}\left(\vec{x}_m, \vec{v}_m\right)}{m}\Delta t\,.$$

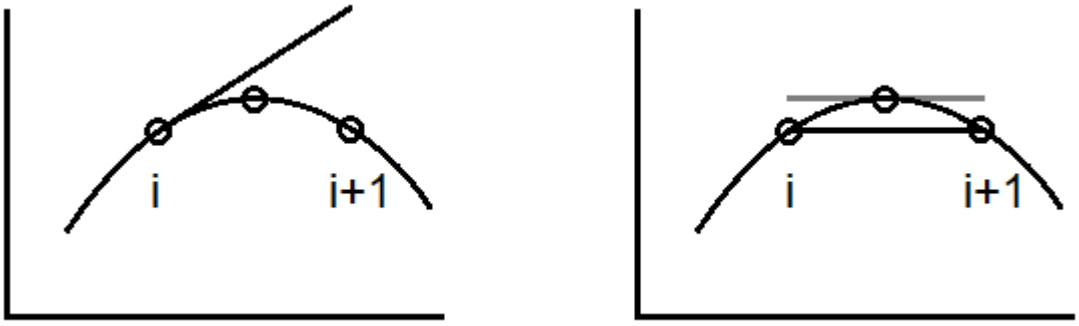

Figure 3: Breakdown of Euler's method (left). Correction using slope at midpoint (right).

Derivation

To derive the above, Taylor-expand the value for $x\left(t\right)$ up to a half-step increment

$$x\left(t + \frac{\Delta t}{2}\right) \approx x\left(t\right) + \frac{1}{2}\,f\left(x\left(t\right), t\right)\,\Delta t\,,$$

where the slope term is replaced by f. Next, note that the 'ordinary' first-order expansion

$$x\left(t + \Delta t\right) \approx x\left(t\right) + f\left(x\left(t\right), t\right)\Delta t$$

can be replaced by the more accurate statement

$$x\left(t + \Delta t\right) \approx x\left(t\right) + f\left(x\left(t + \frac{\Delta t}{2}\right), t + \frac{\Delta t}{2}\right)\Delta t\,.$$

Inserting $x\left(t + \Delta t/2\right)$ into the above, we have

$$x\left(t + \Delta t\right) \approx x\left(t\right) + f\left(x\left(t\right) + \frac{1}{2}\,f\left(x\left(t\right), t\right)\,\Delta t, t + \frac{\Delta t}{2}\right)\Delta t\,,$$

which is the basis for the two-step version.

Energy Conservation

Taking the case of a falling object in one dimension near Earth's surface free of air drag, we have the iterative equations

$$x_{i+1} = x_i + v_m\,\Delta t \qquad\qquad v_m = v_i + \frac{1}{2}\frac{F\left(x_i, v_i\right)}{m}\,\Delta t = v_i - \frac{g}{2}\Delta t\,.$$

Then inserting the form for v_m into x_{i+1} gives

$$x_{i+1} = x_i + v_i\,\Delta t - \frac{g}{2}\Delta t^2 \qquad\qquad v_{i+1} = v_i - g\,\Delta t\,,$$

which is surely exact.

Supposing we were tracking the energy of the object, it's straightforward to show that the value remains fixed across iterations for this example:

$$\begin{aligned}
E_{i+1} &= mgx_{i+1} + \frac{1}{2}mv_{i+1}^2 \\
E_{i+1} &= mg\left(x_i + v_i\,\Delta t - \frac{g}{2}\Delta t^2\right) + \frac{1}{2}m\left(v_i - g\,\Delta t\right)^2 \\
E_{i+1} &= mgx_i + \frac{1}{2}mv_i^2 + \cancel{\Delta t\left(mgv_i - mgv_i\right)} + \cancel{\Delta t^2\left(\frac{mg}{2} - \frac{mg}{2}\right)} \\
E_{i+1} &= E_i
\end{aligned}$$

9.2 Fourth-Order Method

Perhaps the most-commonly used approximation is the fourth-order Runge-Kutta method, abbreviated RK4. Jumping right to the result, the way we advance to the next step is

$$x_{i+1} = x_i + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)\Delta t\,,$$

where:

$$\begin{aligned}
k_1 &= f\left(x, t\right) \\
k_2 &= f\left(x + \frac{\Delta t}{2}k_1, t + \frac{\Delta t}{2}\right) \\
k_3 &= f\left(x + \frac{\Delta t}{2}k_2, t + \frac{\Delta t}{2}\right) \\
k_4 &= f\left(x + \Delta t\,k_3, t + \Delta t\right)
\end{aligned}$$

The RK4 method stays 'true enough' for many problems, and is exact for many others.

Example

Recalling (again) the setup for a simple pendulum in a uniform gravitational field, we have

$$\omega\left(t\right) = \frac{d}{dt}\theta\left(t\right) \qquad\qquad \frac{d}{dt}\omega\left(t\right) = -\frac{g}{L}\sin\theta\,,$$

where θ is the angle of vertical deflection, L is pendulum length, and g is the local gravity constant. For given initial conditions θ_0, ω_0, the advance to the $i+1$th step is contained in the following pseudo-code:

```
k1w = -(g / l) * SIN(q1temp)
k1t = p1temp
w2 = p1temp + k1w * dt / 2
t2 = q1temp + k1t * dt / 2
k2w = -(g / l) * SIN(t2)
k2t = w2
w3 = p1temp + k2w * dt / 2
t3 = q1temp + k2t * dt / 2
k3w = -(g / l) * SIN(t3)
k3t = w3
w4 = p1temp + k3w * dt
t4 = q1temp + k3t * dt
k4w = -(g / l) * SIN(t4)
dwdt = (k1w + 2 * k2w + 2 * k3w + k4w) / 6
dtdt = (k1t + 2 * k2t + 2 * k3t + k4t) / 6
p1 = p1temp + dwdt * dt
q1 = q1temp + dtdt * dt
```

10 Symplectic Integrator

10.1 Hamiltonian Systems

Physical systems not subject to large effects from quantum mechanics or general relativity are well-predicted by Hamilton's equations of motion, a generalization of framing Newtonian mechanics. We begin with an object H, which a function of generalized coordinated q and p, each a function of time. For most cases, it's possible to show that the three equations

$$H(q,p) = T(p) + U(q) \qquad \frac{\partial q}{\partial t} = \frac{\partial H}{\partial p} \qquad \frac{\partial p}{\partial t} = -\frac{\partial H}{\partial q}$$

determine the evolution of the system from initial q_0 and p_0. In the above, the function $T(p)$ is the kinetic energy, almost always equal to $p^2/2m$, whereas $U(q)$ is a spatially-dependent potential energy function. If the system conserves energy, then the Hamiltonian is simply the total energy E.

Discretization

Expanding out the q-derivative formulas in terms of small differential terms, we get

$$\frac{\partial q}{\partial t} = \frac{\partial H}{\partial p} \qquad \rightarrow \qquad \frac{q_{i+1} - q_i}{\Delta t} = \frac{H(q_i, p_{i+i}) - H(q_i, p_i)}{p_{i+1} - p_i}\,.$$

By now, we've seen there exists some freedom in where we choose to base the calculations for the $i+1$th step. Taking a hint from the mixed Euler's method, let us write the derivative with respect to q evaluated at p_{i+1}:

$$\frac{\partial p}{\partial t} = -\frac{\partial H}{\partial q} \qquad \rightarrow \qquad \frac{p_{i+1} - p_i}{\Delta t} = -\frac{H\left(q_{i+1}, p_{i+i}\right) - H\left(q_i, p_{i+1}\right)}{q_{i+1} - q_i} \,.$$

Energy Conservation

To assure that the above equations don't cause the Hamiltonian to change, we combine them as follows

$$\cancel{H\left(q_i, p_{i+i}\right)} - H\left(q_i, p_i\right) + H\left(q_{i+1}, p_{i+i}\right) - \cancel{H\left(q_i, p_{i+i}\right)} = \frac{\left(q_{i+1} - q_i\right)\left(p_{i+1} - p_i\right)}{\Delta t}\left(1 - 1\right)$$

$$H\left(q_{i+1}, p_{i+i}\right) = H\left(q_i, p_i\right) \,,$$

which indeed remains the same. Note that this construction lends itself to a very wide variety of problems in physics, and easily generalizes to more degrees of freedom (extra dimensions or particles).

10.2 Symplectic Notation

A system with N degrees of freedom requires N pairs, or $2N$ generalized coordinates q^j, p^j, where $j = 1 \ldots N$. Let us construct a single column vector containing all generalized coordinates:

$$\vec{\eta} = \begin{bmatrix} \vec{q} \\ \vec{p} \end{bmatrix} = \begin{bmatrix} q^1 \\ q^2 \\ \cdots \\ p^1 \\ p^2 \\ \cdots \end{bmatrix} \qquad \eta^i = q^i \qquad \eta^{i+N} = p^i$$

Then, Hamilton's equations of motion take the form

$$\frac{\partial \eta^{i+N}}{\partial t} = -\frac{\partial H}{\partial \eta^i} \qquad \frac{\partial \eta^i}{\partial t} = \frac{\partial H}{\partial \eta^{i+N}} \,.$$

The 'J' Matrix

Proceeding with a simple $N = 2$ example, we can easily write the above as a matrix

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -\partial_t p^1 \\ -\partial_t p^2 \\ \partial_t q^1 \\ \partial_t q^2 \end{bmatrix} = \begin{bmatrix} \partial_t q^1 \\ \partial_t q^2 \\ \partial_t p^1 \\ \partial_t p^2 \end{bmatrix} \,,$$

or in tighter notation, we write the $2N \times 2N$ matrix as J such that

$$J \frac{\partial}{\partial \vec{\eta}} H = \frac{\partial}{\partial t} \vec{\eta} \, .$$

Of course, each of the above results generalizes to more than $N = 2$, in which case J gains more rows and columns but retains the same structure. In the general case, we readily find

$$JJ = -I \qquad\qquad J^T = J^{-1} = -J \qquad\qquad |J| = 1 \, .$$

The 'M' Matrix

Recalling our study of Euler's method, recall that coordinates are advanced to the $i+1$th step using using a matrix

$$M \begin{bmatrix} \vec{q}_i \\ \vec{p}_i \end{bmatrix} = \begin{bmatrix} \vec{q}_{i+1} \\ \vec{p}_{i+1} \end{bmatrix} \, ,$$

which in symplectic notation reads

$$M \vec{\eta}_i = \vec{\eta}_{i+1} \, .$$

10.3 Symplectic Condition

So far, we have worked out the notation to support iterative equations for determining the evolution of a physical system. At the ith step, we have

$$\frac{\partial}{\partial t} \vec{\eta}_i = J \frac{\partial}{\partial \vec{\eta}_i} H \left(\vec{q}_i, \vec{p}_i \right) \, .$$

To assure the Hamiltonian (and thus the energy) is not altered by our choice of variables, let us demand that the form

$$\frac{\partial}{\partial t} \vec{\eta}_{i+1} = J \frac{\partial}{\partial \vec{\eta}_{i+1}} H \left(\vec{q}_{i+1}, \vec{p}_{i+1} \right)$$

remain true. This passes all freedom to the matrix M, as J is constant and H is assumed given.

Substituting $M \vec{\eta}_i = \vec{\eta}_{i+1}$ into the second equation, we get

$$\frac{\partial}{\partial t} \left(M \vec{\eta}_i \right) = J \frac{\partial}{\partial \left(M \vec{\eta}_i \right)} H \left(\vec{q}_{i+1}, \vec{p}_{i+1} \right) \, ,$$

which simplifies to tell us

$$\frac{\partial}{\partial t} \vec{\eta}_i = M J M^T \frac{\partial}{\partial \vec{\eta}_i} H \left(\vec{q}_i, \vec{p}_i \right) \, .$$

Comparing this to the first equation, we find

$$J = M J M^T \, ,$$

known as the *symplectic condition*. This is a very important result, as it can test any proposed M matrix without needing a simulation.

Examples

Since we already have some M-matrices laying around from our study of Euler's method, we may test each against the symplectic condition:

$$\text{Forward}: \quad \begin{bmatrix} 1 & \Delta t/m \\ -k\,\Delta t & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -k\,\Delta t \\ \Delta t/m & 1 \end{bmatrix}$$

$$\text{Backward}: \quad \left(\frac{1}{(1+(k/m)\,\Delta t^2)}\right)^2 \begin{bmatrix} 1 & \Delta t/m \\ -k\,\Delta t & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -k\,\Delta t \\ \Delta t/m & 1 \end{bmatrix}$$

$$\text{Mixed}: \quad \begin{bmatrix} 1 & \Delta t/m \\ -k\,\Delta t & (1-(k/m)\,\Delta t^2) \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -k\,\Delta t \\ \Delta t/m & (1-(k/m)\,\Delta t^2) \end{bmatrix}$$

Denoting the forward, backward, and mixed matrices as M_F, M_B, M_S, respectively, the above simplifies to

$$\text{Forward}: \quad \left(MJM^T\right)_F = \left(1+\frac{k}{m}\Delta t^2\right) J$$

$$\text{Backward}: \quad \left(MJM^T\right)_B = \left(1+\frac{k}{m}\Delta t^2\right)^{-1} J$$

$$\text{Mixed}: \quad \left(MJM^T\right)_S = J\,.$$

Not surprisingly, the forward and backward methods fail the symplectic test. The mixed case however slips through flawlessly! That is, the mixed Euler's method, as applied to harmonic motion, satisfies the symplectic test.

10.4 Equations of Motion

Returning to the general case where the Hamiltonian and its derivatives determine everything, namely

$$H\left(\vec{q},\vec{p}\right) = T\left(\vec{p}\right) + U\left(\vec{q}\right) \qquad \frac{\partial \vec{q}}{\partial t} = \frac{\partial H}{\partial \vec{p}} \qquad \frac{\partial \vec{p}}{\partial t} = -\frac{\partial H}{\partial \vec{q}}\,,$$

we now know to discretize the derivative terms via

$$\vec{q}_{i+1} = \vec{q}_i + \frac{\partial T}{\partial \vec{p}_i}\Delta t \qquad \vec{p}_{i+1} = \vec{p}_i - \frac{\partial V}{\partial \vec{q}_{i+1}}\Delta t\,,$$

which preserves the value of H across iteration steps.