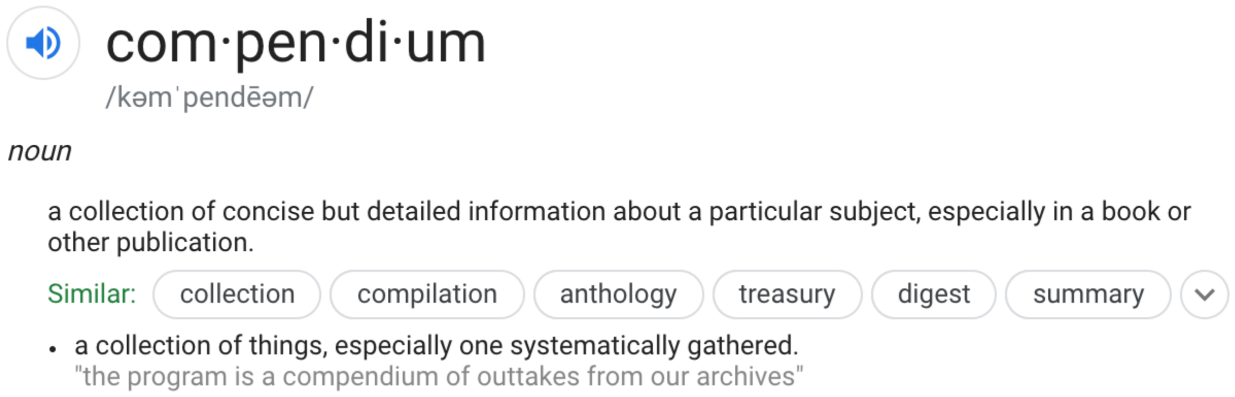
Notes on numerical analysis

Sean Laverty

December 5, 2019



com·pen·di·um

/kəmˈpendēəm/

noun

a collection of concise but detailed information about a particular subject, especially in a book or other publication.

Similar: collection compilation anthology treasury digest summary

- a collection of things, especially one systematically gathered.
  "the program is a compendium of outtakes from our archives"

Contents

1 Introduction

1.1 Polynomial form and evaluation speed

To explore the relationship between computational time and polynomial form, consider three forms of the same polynomial

$$\begin{aligned} f_1(x) &= x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120 && \text{(normal)} \\ f_2(x) &= (x-1)(x-2)(x-3)(x-4)(x-5) && \text{(factored)} \\ f_3(x) &= -120 + x(274 + x(-225 + x(85 + x(-15 + x)))) && \text{(nested)} \end{aligned}$$

We can evaluate each function at 1000 randomly generated points (to be used for all functions and all replicates) and, using an R package called microbenchmark, repeat this experiment 1000 times for each form. Results of this experiment are given in Figure 1.

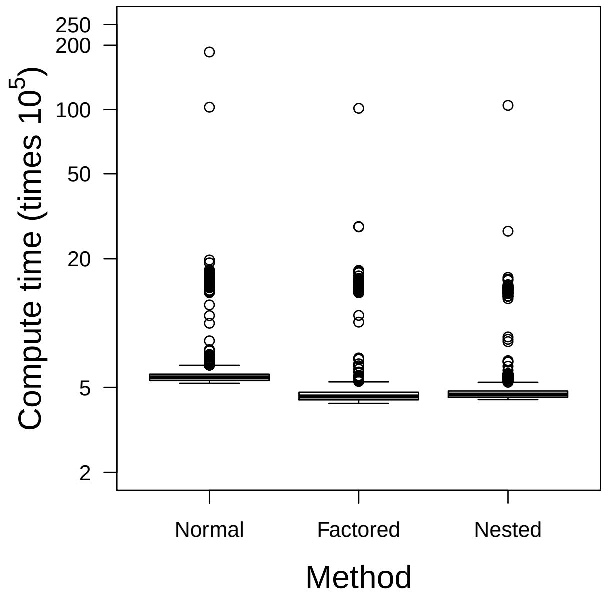


Figure 1: Median computing times (measured execution time in microseconds) are 607.1064 for f_1, 511.8887 for f_2, and 516.9337 for f_3.

With respect to class, you might ask yourself why we have emphasized nesting, but not factoring, when it comes to working with polynomials.

2 Root-finding and fixed-point problems

2.1 Fixed-point iteration

Consider the function

$$g(x) = 2^{-x} = e^{-\ln(2)x}$$

and the associated fixed-point problem $g(x) = x$. The result obtained by cobwebbing (graphically) is shown in the figure below, with the iterates marked in ticks on the bottom axis. Beyond $n = 4$ things get pretty crowded near the fixed point. As tabular output we have the first few steps as well. Note

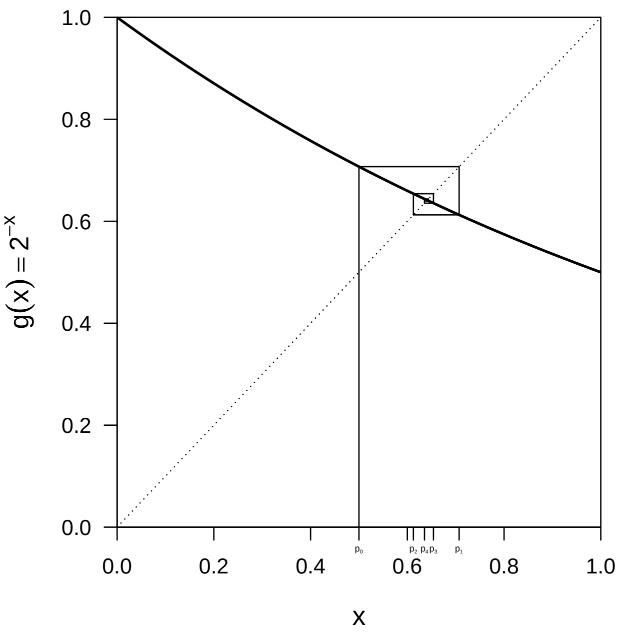


Figure 2: A fixed-point problem.

that p_n are approximates to the root p.

n	p_n
0	0.5000000
1	0.7071068
2	0.6125473
3	0.6540409
4	0.6354978

Now, for a given degree of accuracy, how many iterations do we actually need? Consider

$$2^{-x} = x \text{ on } \left[\frac{1}{3}, 1\right]$$

The bounds are given by

$$|p_n - p| \le k^n \max(p_0 - a, b - p_0)$$

using the initial guess, and by

$$|p_n - p| \le \frac{k^n}{1-k}|p_1 - p_0|$$

using the initial guess and first iteration. We will look at a few applications of the bounds to the problem above. First notice that $g'(x) = -\ln(2)2^{-x}$ and $|g'(x)| \le \ln(2)2^{-1/3} < k = 0.551$, where $k = 0.551$ is a bound on the magnitude of $g'(x)$.

The worst possible initial guess would be at one of the endpoints, so we will start there (this maximizes the term $\max(p_0 - a, b - p_0)$, which in this case we actually want to do in order to generate a conservative bound). Taking D as the desired accuracy (i.e., an accuracy within 10^{-D}), this gives,

$$\begin{aligned} k^n \max(p_0 - a, b - p_0) &< 10^{-D} \\ (0.551)^n \left(\frac{2}{3}\right) &< 10^{-D} \\ (0.551)^n &< \left(\frac{3}{2}\right) 10^{-D} \\ n\log(0.551) &< \log\left(\frac{3}{2}\right) - D \\ n &> \frac{\log\left(\frac{3}{2}\right) - D}{\log(0.551)} \end{aligned}$$

In the last line, the inequality has been reversed since we are dividing by a negative. With $D = 4$ this gives $n > 14.77277$ which requires $N = 15$ steps.

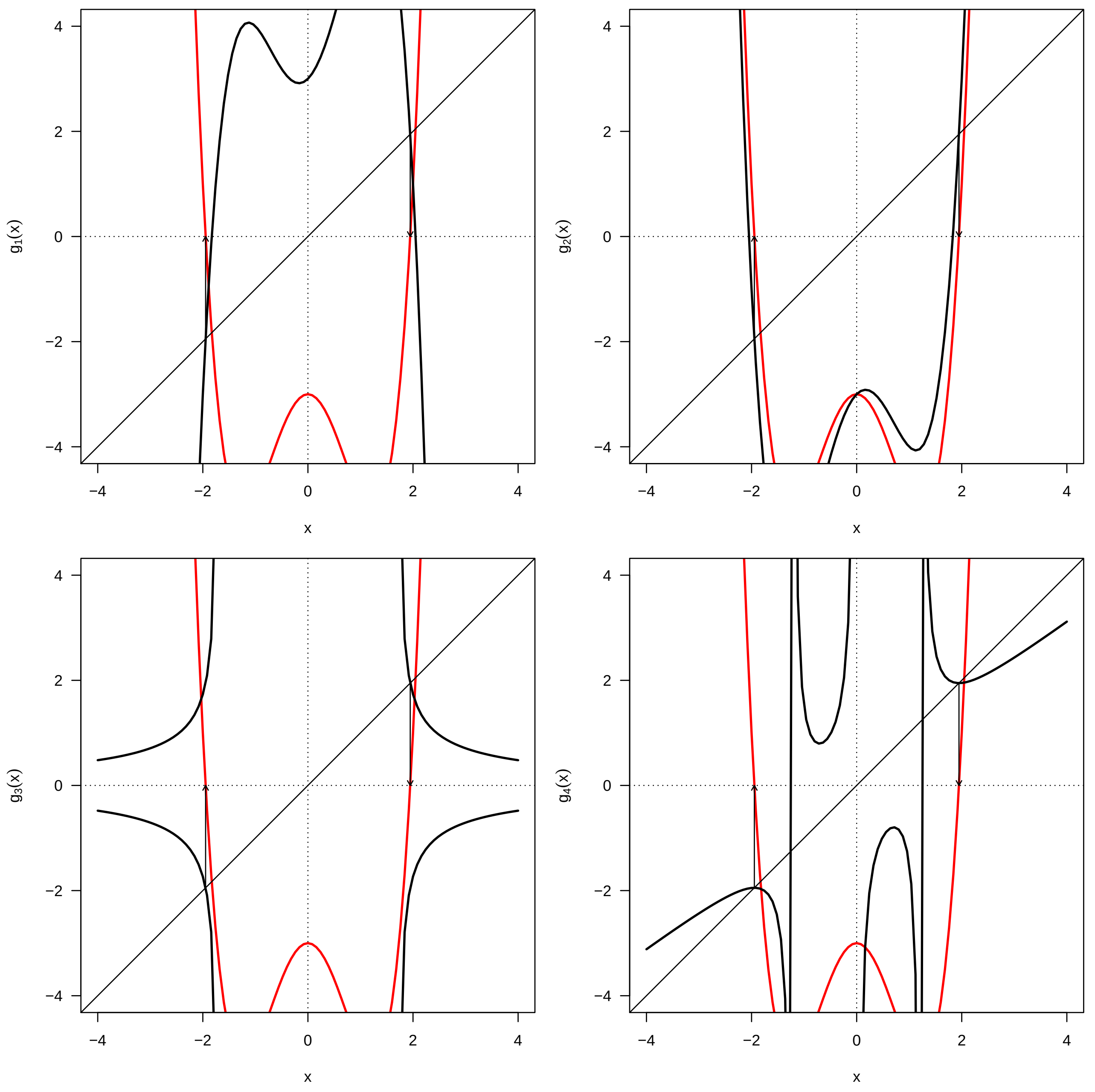
For the second bound, we actually need p_1 in addition to p_0. From $p_0 = \frac{2}{3}$, we have $p_1 = 2^{-1/3}$ (so $|p_1 - p_0| = |2^{-1/3} - \frac{1}{3}| \approx 0.4604$). Similarly, from $p_0 = 1$, we have $p_1 = \frac{1}{2}$ (so $|p_1 - p_0| = |\frac{1}{2} - 1| = 0.5$). We will use the second of these which is larger in value.

$$\begin{aligned} \frac{k^n}{1-k}|p_1 - p_0| &< 10^{-D} \\ \frac{(0.551)^n}{1 - 0.551}(0.5) &< 10^{-D} \\ (0.551)^n &< \left(\frac{1 - 0.551}{0.5}\right) 10^{-D} \\ n\log(0.551) &< \log\left(\frac{1 - 0.551}{0.5}\right) - D \\ n &> \frac{\log\left(\frac{1-0.551}{0.5}\right) - D}{\log(0.551)} \end{aligned}$$

For consistency, with $D = 4$ this gives $n > 15.63357$ which requires $N = 16$ steps. We have to do at least 16 steps to ensure we are within the bound, though we may satisfy this much more quickly. Notice that this is quite a bit more work than our bound for the Bisection method required.

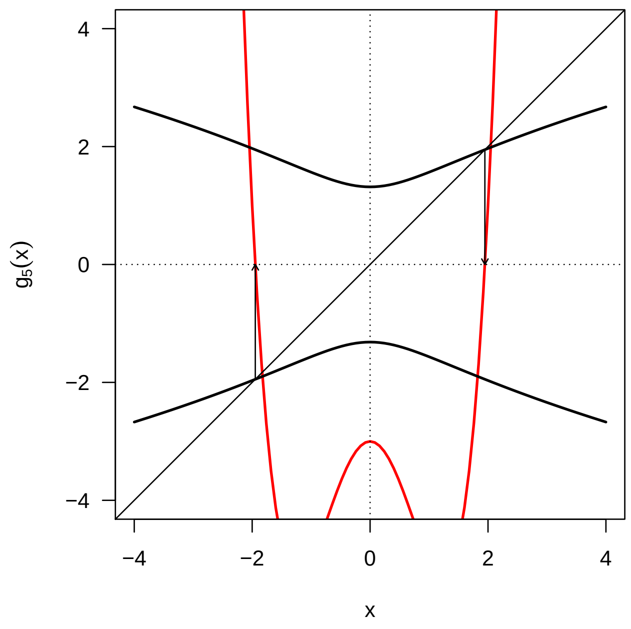
Consider the root-finding problem $f(x) = x^4 - 3x^2 - 3$. It has many different, but equivalent, fixed-point problems.

$$\begin{aligned} g_1(x) &= x - f(x) = x \\ g_2(x) &= x + f(x) = x \\ g_3(x) &= \sqrt{\frac{3}{x^2 - 3}} = x \\ g_4(x) &= x - \frac{x^4 - 3x^2 - 3}{4x^3 - 6x} \end{aligned}$$


$g_1(x)$
$g_2(x)$
$g_3(x)$
$g_4(x)$
x
−4
−2
0
2
4

A better formulation is given by

$$g_5(x) = \sqrt[4]{3x^2+3}$$



Consider the root-finding problem $3x^2 - e^x = 0$. It has a few different, but equivalent, fixed-point problems.

$$g_1(x) = \pm\sqrt{\frac{e^x}{3}} = x \tag{1}$$

$$g_2(x) = \ln(3x^2) = x \tag{2}$$

$$g_3(x) = 3x^2 - e^x + x = x \tag{3}$$

Unfortunately these differ with regard to theory of fixed-point iterations. In particular, keep in mind

for existence: the trapping region $g \in [a,b]$ for $x \in [a,b]$

for uniqueness: the slope criterion $0 < |g'(x)| < 1$

The first function, $g_1(x)$ is able to capture two of the three points of interest, with nicely trapped solutions and small derivatives on those intervals. It does fail to capture the third (rightmost) root, where the slope exceeds one (see Fig. 3). Though our intervals cannot include $x = 0$, $g_2(x)$ works well for the largest root, where the derivative is small and the appropriate trapping region can be constructed (see Fig. 4). The simplest formulation algebraically is $g_3(x)$, yet this has by far the worst characteristics with respect to the fixed-point theory for existence and uniqueness (see Fig. 5). It is difficult to constrict suitable trapping regions and there are only narrow ranges for which the derivative is appropriately bounded in magnitude.

Consider the fixed-point problems $\sin(x) = x$ (below, left) and $\cos(x) = x$ (below, right).

A total of 20 iterations of each problem are shown in the color figure above. Time is indicated in two ways: on the graph early steps are in dark blue and fade to yellow. Along the axis earlier approximations are marked by tall (also dark blue) tick marks and later iterations are marked by shorter (also yellow) tick marks. The scaling of the above-axis tick marks is nothing quantitative or necessarily related to the values of the approximation itself, it is just meant to show the progression of the iterations.

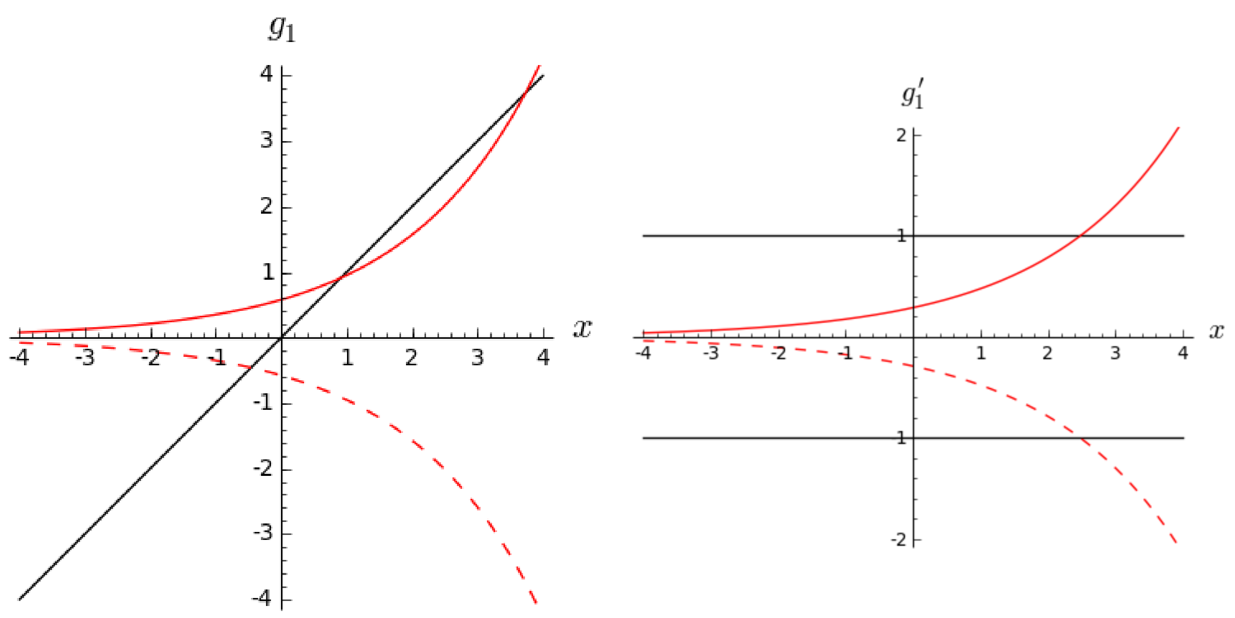


Figure 3: Two functions that give rise to fixed-point problems. Dotted corresponds to the negative value of the square root and the solid line corresponds to the positive value. Left: intersections of each with the black 1 : 1 line indicate fixed points. Right: values between −1 and 1 indicate that a fixed point, if it exists, is unique.

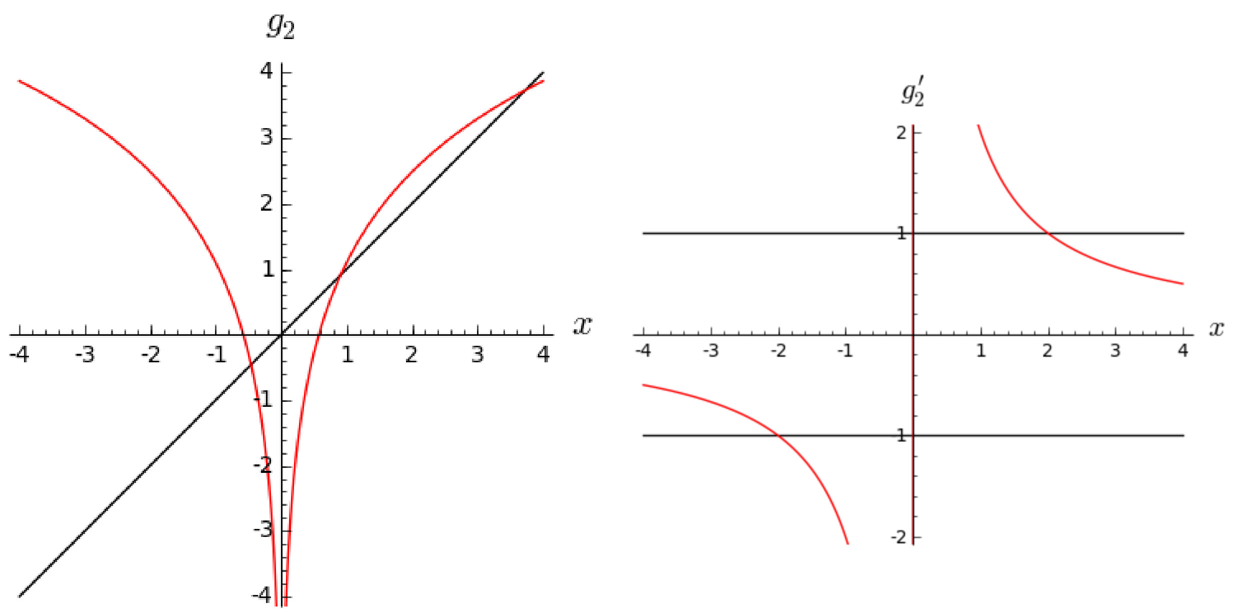


Figure 4: Left: intersections with the black 1 : 1 line indicate fixed points. Right: values between −1 and 1 indicate that a fixed point, if it exists, is unique. Notice that the smaller the value of the fixed point, the steeper the function $g_2(x)$.

For two problems starting from $p_0 = 1$, we see the iterations approaching the fixed point visible on each panel of the graph. Solutions to the first fixed-point problem (left) converge very slowly, while those to the second (right) converge very quickly. Details of the approaches to the fixed points are interesting as well. Approximations converging to the true solution of $p = 0$ for the first problem decrease monotonically, while those converging to the true solution of the second problem oscillate.

Challenge For each of the above, state an associated rootfinding problem, an interval on which you could apply the bisection method, and the number of iterations it would take to approximate the root to an accuracy of 10^{-6} with bisection.

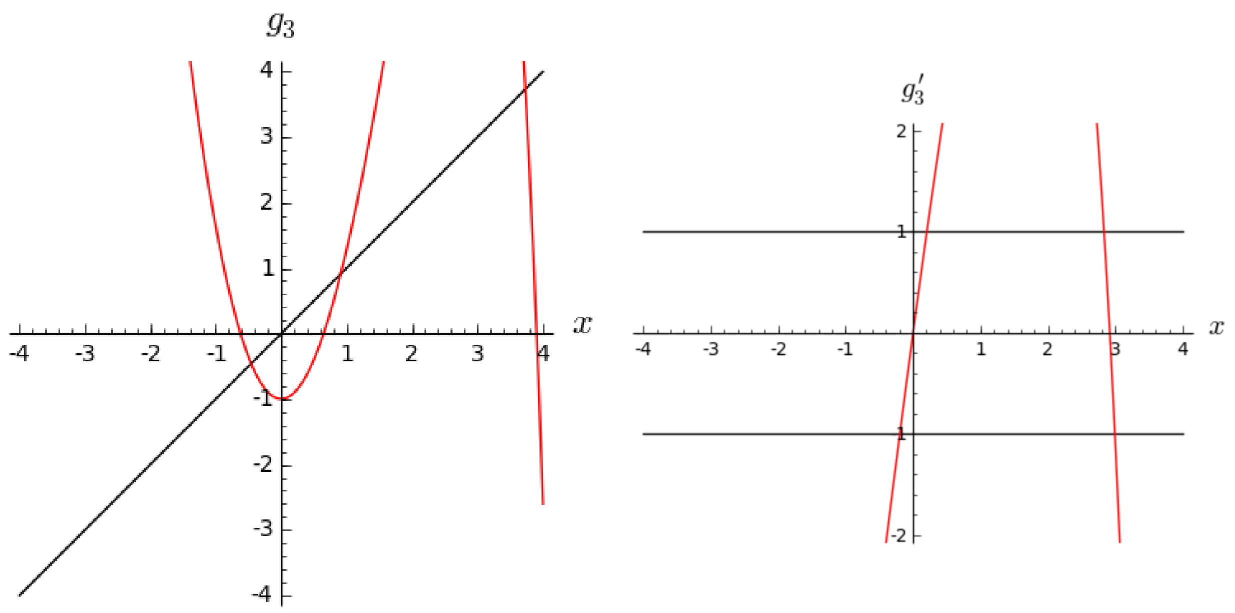


Figure 5: Left: intersections with the black 1 : 1 line indicate fixed points. Right: values between -1 and 1 indicate that a fixed point, if it exists, is unique.

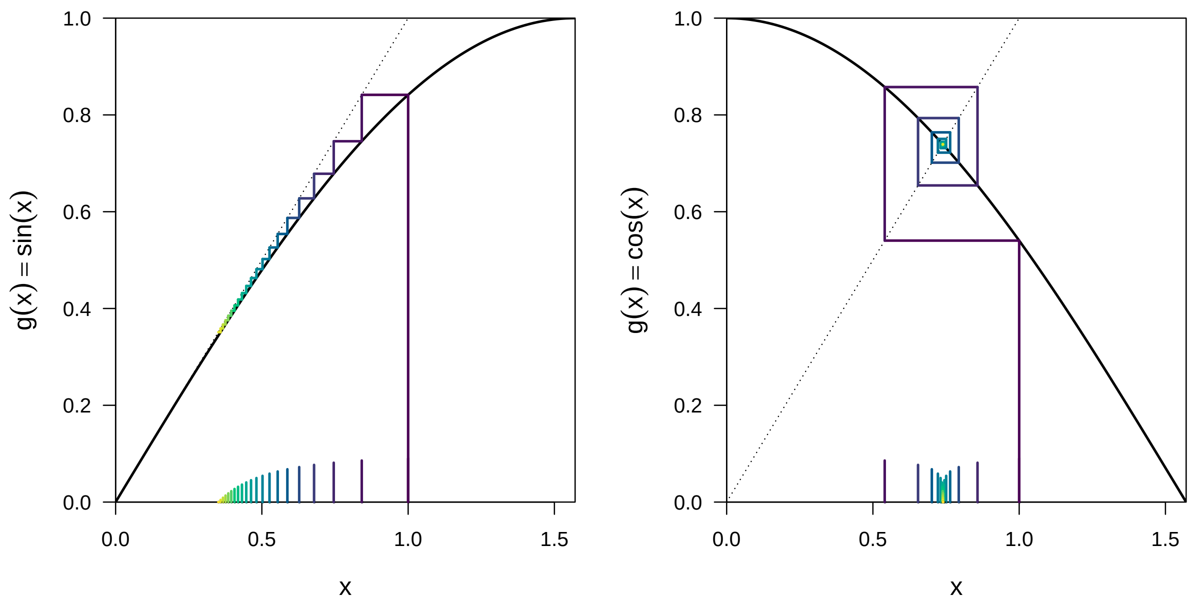


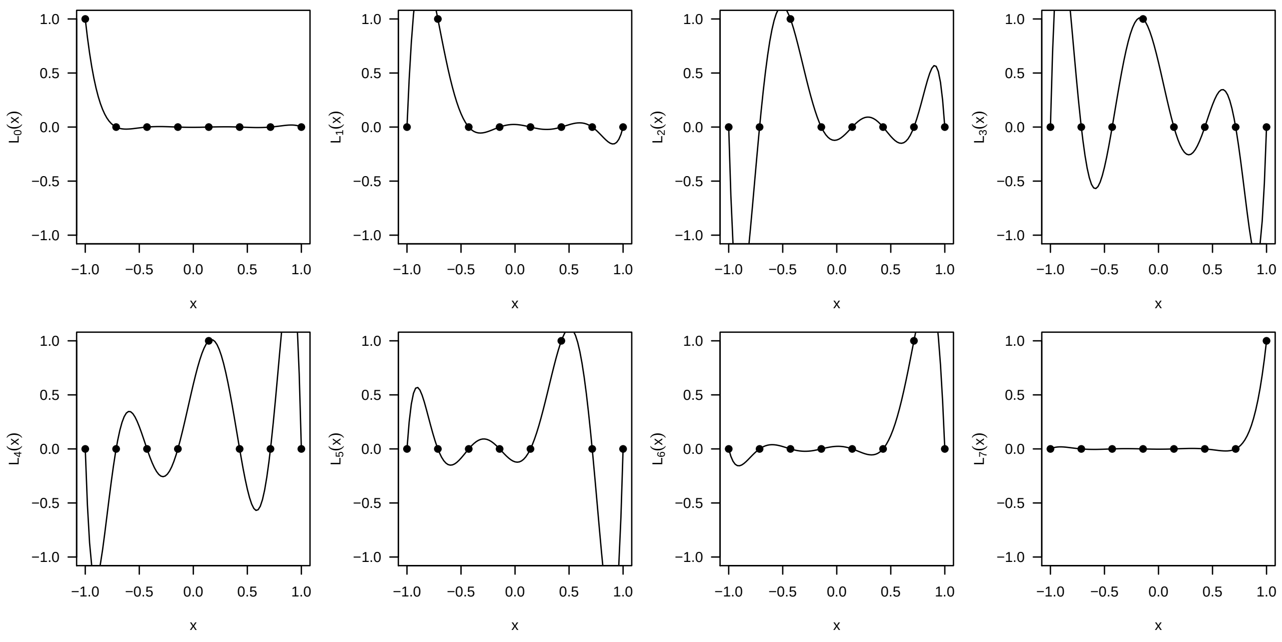
Figure 6: A trigonometric fixed-point problem.

3 Interpolation and approximation

3.1 Polynomial interpolation

3.1.1 Lagrange interpolation

Below are $L_k(x) = L_{7,k}(x)$ for 8 equally spaced points $x_0 = -1, \ldots, x_7 = 1$ on $[-1, 1]$.



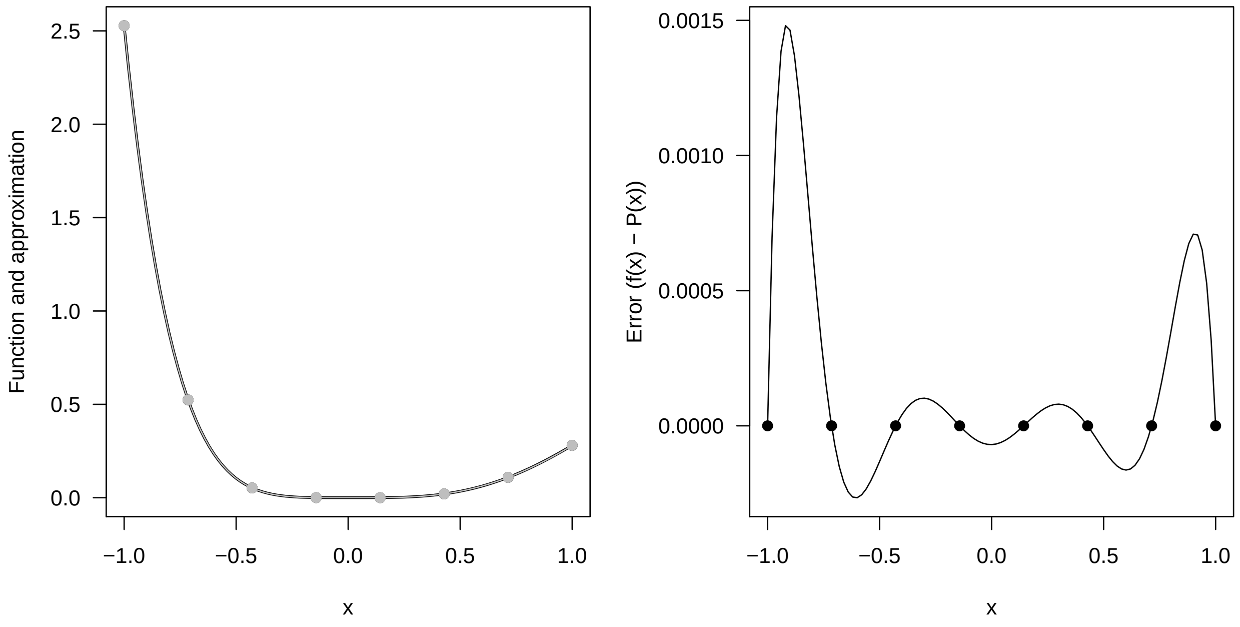
Considering the function

$$f(x) = x^3 e^{-1.1x} \sin(x)$$

we show the approximation

$$P(x) = P_7(x) = \sum_{k=0}^{7} f(x_k) L_k(x)$$

(along with the function) on the left and the error on the right.



3.1.2 Cubic spline interpolation

Note: Before you begin, please note that there is a typo in the clamped spline example in the book (Example 2, page 148, eighth edition), the correct solution will be given below.

Intro: When computing a cubic spline with $n+1$ grid points, keep in mind that we have n intervals, n spline segments, and an $(n+1) \times (n+1)$ matrix. For example, if the data is known at the grid points x_0, x_1, x_2, x_3, x_4 we have 5 points, 4 intervals, 4 spline segments, and a 5×5 matrix. The spacings between grid points, defined $h_j = x_{j+1} - x_j$, are used below.

General approach: We can verify the conditions of the spline by definition, but ultimately the way to implement this, even for small datasets, is to use matrix algebra. Let's use the vector $\mathbf{r}$ (as in $\mathbf{Ax} = \mathbf{r}$, rather than $\mathbf{Ax} = \mathbf{b}$) for the right-hand side to avoid likely confusion between entries of the right-hand side vector and the coefficients b_j of the splines. The first and last rows of $\mathbf{A}$ and $\mathbf{r}$ depend on the choice of boundary conditions, but the interior rows are illustrated below. On the interior rows of the main diagonal, $\mathbf{A}$ has entries

$$2(h_0 + h_1), 2(h_1 + h_2), \cdots, 2(h_{n-3} + h_{n-2}), 2(h_{n-2} + h_{n-1})$$

On the interior rows of the sub-diagonal (below main), $\mathbf{A}$ has entries

$$h_0, h_1, \cdots, h_{n-3}, h_{n-2}$$

On the interior rows of the super-diagonal (above main), $\mathbf{A}$ has entries

$$h_1, h_2, \cdots, h_{n-2}, h_{n-1}$$

$$\mathbf{A} = \begin{bmatrix}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
h_0 & 2(h_0+h_1) & h_1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\
\cdot & \ddots & \ddots & \ddots & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
0 & \cdot & \cdot & 0 & h_{j-1} & 2(h_{j-1}+h_j) & h_j & 0 & \cdot & \cdot & 0 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix}$$

$$\mathbf{r} = \begin{bmatrix} \cdots \\ 3\left(\dfrac{a_2 - a_1}{h_1} - \dfrac{a_1 - a_0}{h_0}\right) \\ \vdots \\ 3\left(\dfrac{a_n - a_{n-1}}{h_{n-1}} - \dfrac{a_{n-1} - a_{n-2}}{h_{n-2}}\right) \\ \cdots \end{bmatrix}$$

Natural BCs: The interior rows of $\mathbf{A}$ are as described above, but the first row begins with $1, 0, \ldots$ and the last row ends with $\ldots, 0, 1$. To satisfy the boundary condition, the first and last rows of of $\mathbf{r}$ are exactly 0.

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 1 \end{bmatrix}$$

$$\mathbf{r} = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{bmatrix}$$

Clamped BCs: The interior rows of $\mathbf{A}$ are as described above, but the first row begins with $2h_0, h_0, 0, \ldots$ and the last row ends with $\ldots, 0, h_{n-1}, 2h_{n-1}$. To satisfy the boundary condition, the first and last rows of of $\mathbf{r}$ are exactly $3\left(\frac{a_1 - a_0}{h_0} - f'(a)\right)$ and $3\left(f'(b) - \frac{a_n - a_{n-1}}{h_{n-1}}\right)$, respectively.

$$\mathbf{A} = \begin{bmatrix} 2h_0 & h_0 & 0 & \cdot & \cdot & \cdot & \cdot & 0 \\ \cdot & \ddots & \ddots & \ddots & \cdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \ddots & \ddots & \ddots & \cdot \\ \cdot & \cdots & \cdot & \cdot & \cdot & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix}$$

$$\mathbf{r} = \begin{bmatrix} 3\left(\frac{a_1 - a_0}{h_0} - f'(a)\right) \\ \vdots \\ \vdots \\ \vdots \\ 3\left(f'(b) - \frac{a_n - a_{n-1}}{h_{n-1}}\right) \end{bmatrix}$$

Parameterizing the spline: Once the c_j's have been solved, the b_j's and d_j's can be specified *in reverse order* from $j = n-1, n-2, \ldots, 0$ (see Alg. 3.4 (Step 6) or Alg. 3.5 (Step 7)).

$$\begin{aligned} b_j &= \frac{a_{j+1} - a_j}{h_j} - \frac{h_j(c_{j+1} + 2c_j)}{3} \\ d_j &= \frac{c_{j+1} - c_j}{3h_j} \end{aligned}$$

Notice that c_n (from $\mathbf{x}$) is used in the calculation of d_{n-1}, but is never actually used in a spline. Regardless of the boundary condition, the calculated coefficients will appear in the spline of the form

$$S(t) = \begin{cases} \cdots & , \quad \cdots \\ S_j(t) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, & x_j \le x < x_{j+1} \\ \cdots & , \quad \cdots \end{cases}$$

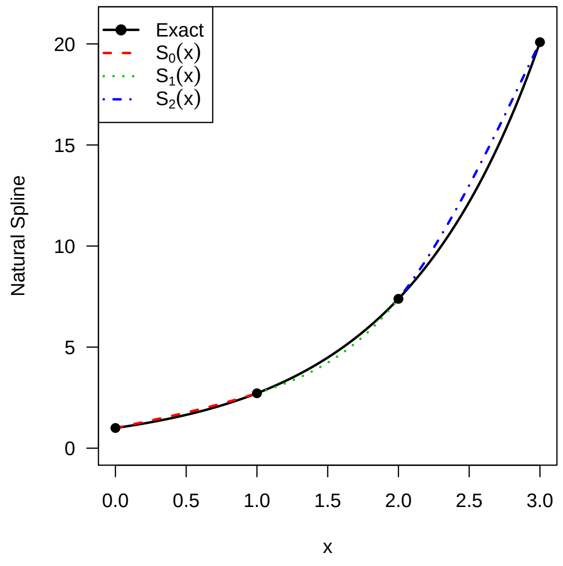
Note that $x_0 = a$ and $x_n = b$.

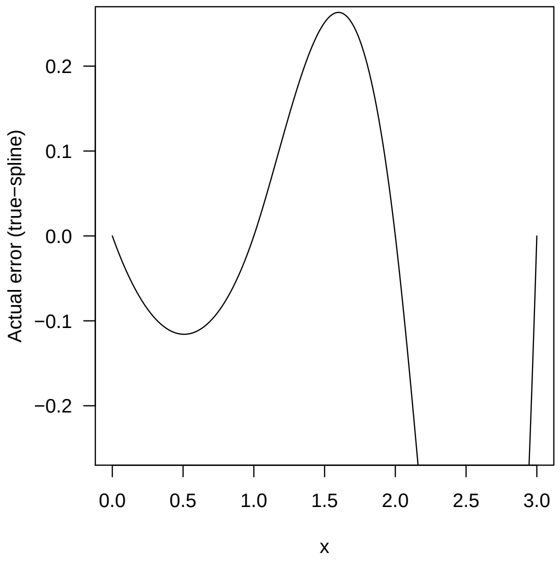
Keep reading: (next page, please)

Example Consider the data $x_0 = 0, x_1 = 1, x_2, = 2, x_3 = 3$ and $f(x) = e^x$.

A natural spline: See Example 1 on page 143.

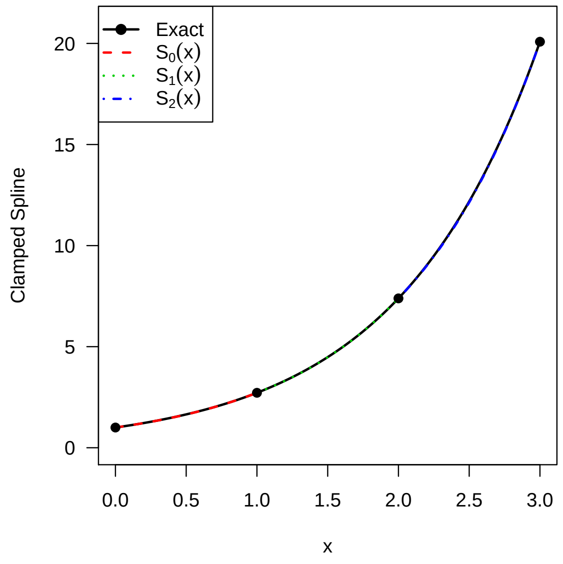
j	a_j	b_j	c_j	d_j
0	1.000000	1.465998	0.0000000	0.2522842
1	2.718282	2.222850	0.7568526	1.6910714
2	7.389056	8.809770	5.8300668	-1.9433556

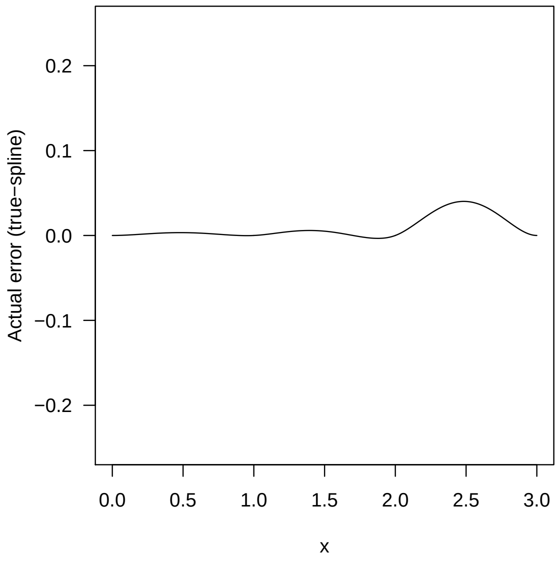




A clamped spline: See Example 2 on page 148.

j	a_j	b_j	c_j	d_j
0	1.000000	1.000000	0.4446825	0.2735993
1	2.718282	2.710163	1.2654805	0.6951308
2	7.389056	7.326516	3.3508729	2.0190916





3.2 Discrete least squares approximation

Given the data in the vehicle speed problems (see sections 3.3, 3.4 homework), we can plot a few functions representing the position data (see Figure 7). The Hermite (black) and spline (red) interpolations exactly pass through the position data points. On the other hand, the linear approximation (blue) does not exactly pass through a single of these points.

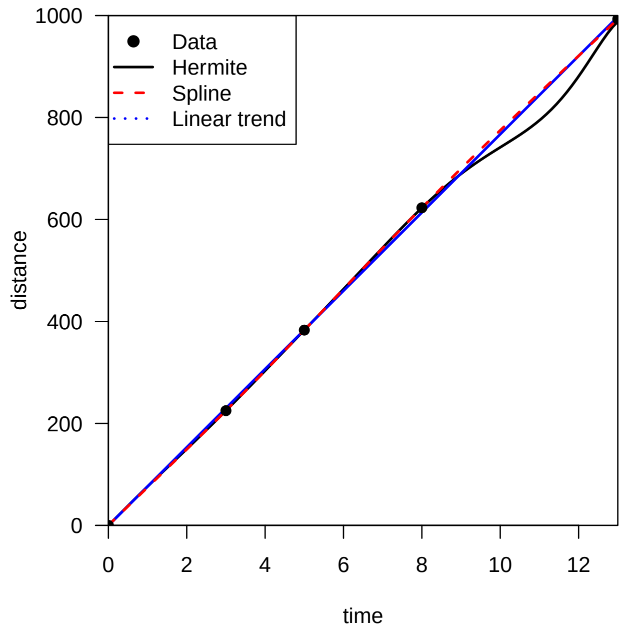


Figure 7: Data and interpolating and approximating polynomials.

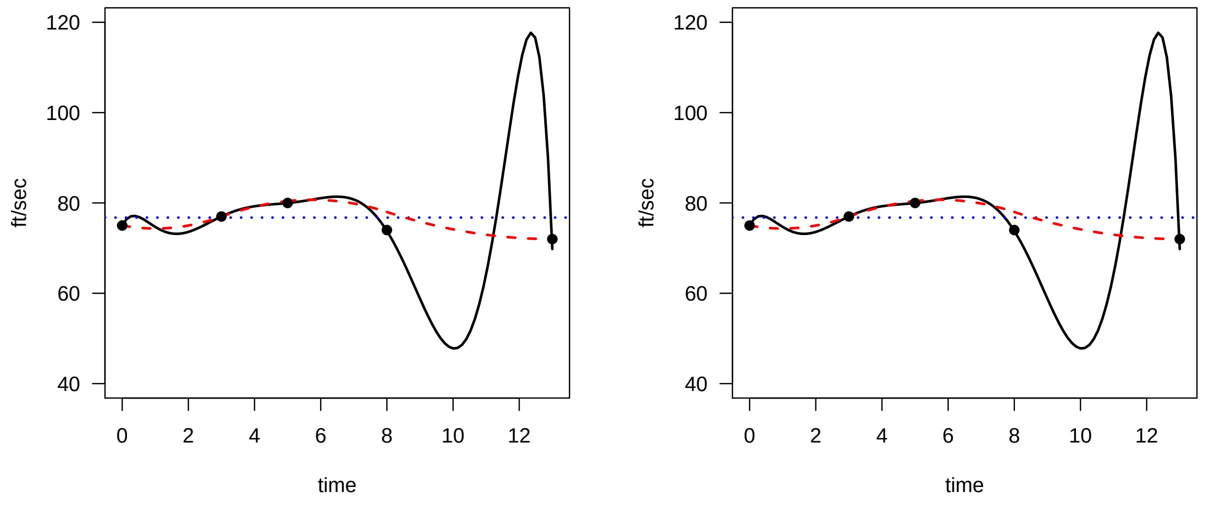


Figure 8: Speed data and predicted speed by interpolating and approximating polynomials (**left:** feet per second and **right:** miles per hour).

There are a variety of factors to weigh when choosing a method for representing data by a function.

Discrete case (i.e., 8.1, pg. 486)

Consider the last term

$$\sum_{i=1}^{m}\Big(P_n(x_i)\Big)^2=\sum_{i=1}^{m}\left(\sum_{j=0}^{n}a_jx_i^j\right)^2$$

in the definition of the error $E=E_2(a_0,a_1,\dots,a_n)$ for discrete least squares approximation. We have the following (though I do not find this particularly illuminating and instead prefer to differentiate before addressing the sums), where it might initially be helpful to remember that $x_i^0=1$ and $x_i^1=x_i$ (e.g., $a_0x_i^0=a_0\cdot 1=a_0$ and $a_1x_i^1=a_1x_i$),

$$\begin{aligned}
\sum_{i=1}^{m}\left(\sum_{j=0}^{n}a_jx_i^j\right)^2 &= \sum_{i=1}^{m}(a_0+a_1x_i+\cdots+a_nx_i^n)(a_0+a_1x_i+\cdots+a_nx_i^n)\\
&= \sum_{i=1}^{m}\Big[a_0(a_0+\cdots+a_nx_i^n)+\cdots+a_jx_i^j(a_0+\cdots+a_nx_i^n)+\cdots+a_nx_i^n(a_0+\cdots+a_nx_i^n)\Big]\\
&= \sum_{i=1}^{m}\sum_{j=0}^{n}a_jx_i^j(a_0+a_1x_i+\cdots+a_nx_i^n)\\
&= \sum_{j=0}^{n}a_j\sum_{i=1}^{m}x_i^j(a_0+a_1x_i+\cdots+a_nx_i^n)\\
&= \sum_{j=0}^{n}a_j\sum_{i=1}^{m}x_i^j\sum_{k=0}^{n}a_kx_i^k\\
&= \sum_{j=0}^{n}a_j\sum_{i=1}^{m}\sum_{k=0}^{n}a_kx_i^{j+k}\\
&= \sum_{j=0}^{n}\sum_{k=0}^{n}a_ja_k\Big(\sum_{i=1}^{m}x_i^{j+k}\Big)
\end{aligned}$$

Our goal is the partial derivative of this term with respect to a_j. Notice the following, where key steps are moving differentiation under the sum, applying the product rule, and rearranging the sum,

$$\begin{aligned}
\frac{\partial}{\partial a_j}\left(\sum_{i=1}^{m}\left(\sum_{j=0}^{n} a_j x_i^j\right)^2\right) &= \frac{\partial}{\partial a_j}\Big(\sum_{i=1}^{m}(a_0 + a_1 x_i + \cdots + a_n x_i^n)(a_0 + a_1 x_i + \cdots + a_n x_i^n)\Big) \\
&= \sum_{i=1}^{m}\frac{\partial}{\partial a_j}\Big((a_0 + a_1 x_i + \cdots + a_n x_i^n)(a_0 + a_1 x_i + \cdots + a_n x_i^n)\Big) \\
&= \sum_{i=1}^{m}\Big(x_i^j(a_0 + a_1 x_i + \cdots + a_n x_i^n) + (a_0 + a_1 x_i + \cdots + a_n x_i^n)x_i^j\Big) \\
&= \sum_{i=1}^{m} 2\Big(x_i^j(a_0 + a_1 x_i + \cdots + a_n x_i^n)\Big) \\
&= \sum_{i=1}^{m} 2\left(x_i^j \sum_{k=0}^{n}(a_k x_i^k)\right) \\
&= 2\sum_{k=0}^{n} a_k \sum_{i=1}^{m} x_i^{j+k}
\end{aligned}$$

Related to this, to find the least squares polynomial approximation to $\{(x_i, y_i)\}_{i=1}^m$ by $P_n(x) = \sum\limits_{k=0}^{n} a_k x^k$, we solve

$$D^T D\vec{a} = D^T\vec{y}$$

for the vector of unknown coefficients $\vec{a}$ where

$$D = \begin{bmatrix} (x_1)^0 & (x_1)^1 & \cdots & (x_1)^n \\ (x_2)^0 & (x_2)^1 & \cdots & (x_2)^n \\ \vdots & \vdots & \vdots & \vdots \\ (x_m)^0 & (x_m)^1 & \cdots & (x_m)^n \end{bmatrix}$$

Continuous case (i.e., 8.2, pg. 495)

Consider the last term

$$\int_a^b \left(\sum_{k=0}^{n} a_k x^k\right)^2$$

in the definition of the error $E = E_2(a_0, a_1, \dots, a_n)$ for least squares polynomial approximation. We have,

$$\int_a^b \left(\sum_{k=0}^{n} a_k x^k\right)^2 dx = \int_a^b (a_0 + a_1 x + \cdots + a_n x^n)(a_0 + a_1 x + \cdots + a_n x^n)\, dx$$

Our goal is the partial derivative of this term with respect to a_j. Notice the following, where key steps are moving differentiation under the integral, applying the product rule, and rearranging the sum,

$$\begin{aligned}
\frac{\partial}{\partial a_j}\int_a^b \left(\sum_{k=0}^{n} a_k x^k\right)^2 dx &= \frac{\partial}{\partial a_j}\int_a^b (a_0 + a_1 x + \cdots + a_n x^n)(a_0 + a_1 x + \cdots + a_n x^n)\, dx \\
&= \int_a^b \frac{\partial}{\partial a_j}\Big((a_0 + a_1 x + \cdots + a_n x^n)(a_0 + a_1 x + \cdots + a_n x^n)\Big)\, dx \\
&= \int_a^b \left(x^j(a_0 + a_1 x + \cdots + a_n x^n) + (a_0 + a_1 x + \cdots + a_n x^n)x^j\right)\, dx \\
&= \int_a^b 2x^j(a_0 + a_1 x + \cdots + a_n x^n)\, dx \\
&= 2\int_a^b x^j \sum_{k=0}^{n} a_k x^k\, dx \\
&= 2\int_a^b \sum_{k=0}^{n} a_k x^{j+k}\, dx \\
&= 2\sum_{k=0}^{n} a_k \int_a^b x^{j+k}\, dx
\end{aligned}$$

3.3 Least squares function approximation

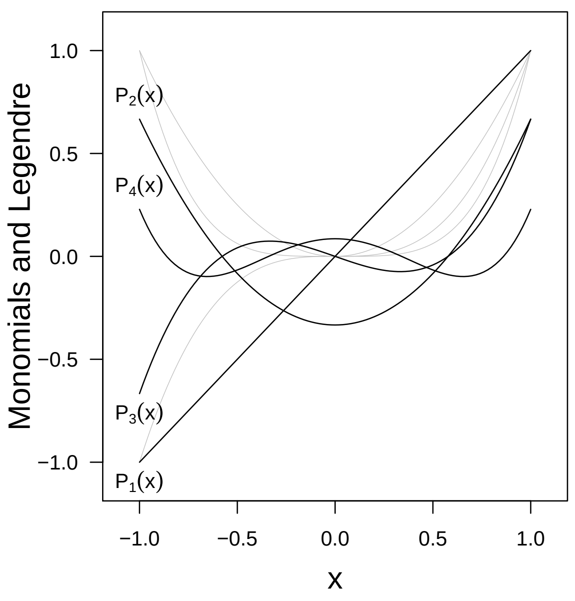
3.3.1 Legendre polynomials

The standard monomials for function approximation, shown as the familiar unlabeled gray curves in Figure 9, are $\phi_i(x) = x^i$ for $i = 0, 1, \dots, n$. These lack the property of orthogonality that is achieved by the Legendre polynomials, also shown Figure 9, but in black and labeled. The first few Legendre polynomials are

3.3.2 Chebyshev polynomials

The standard monomials for function approximation, shown as the familiar unlabeled gray curves in Figure 10, are $\phi_i(x) = x^i$ for $i = 0, 1, \dots, n$. These lack the property of orthogonality that is achieved by the Chebyshev polynomials, also shown Figure 10, but in black and labeled. The first few Chebyshev polynomials are

Since the alternate definition of the Chebyshev polynomials is $T_n(x) = \cos(n \arccos(x))$ for $n \geq 0$, it should seem reasonable that these functions are bounded in magnitude by one, as illustrated in



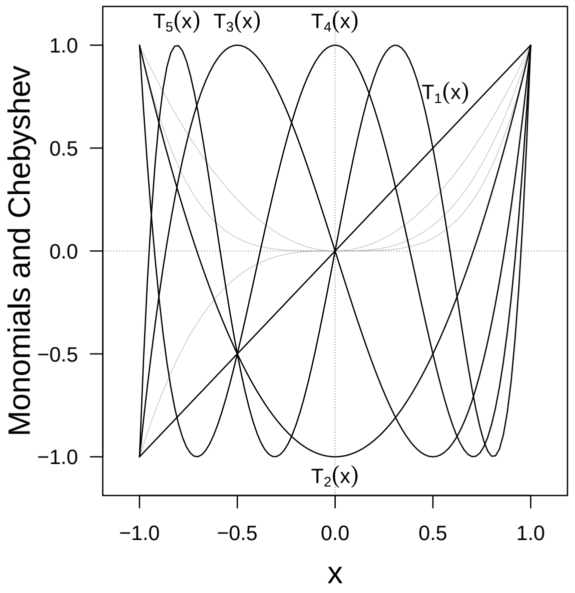
$$
\begin{aligned}
P_0(x) &= 1\\
P_1(x) &= x\\
P_2(x) &= x^2 - \frac{1}{3}\\
P_3(x) &= x^3 - \frac{3}{5}x\\
&\dots\\
P_n(x) &= (x - B_k)P_{n-1}(x) - C_k P_{x-2}(x)
\end{aligned}
$$

where

$$
B_k = \frac{\int_{-1}^{1} x \cdot 1 \cdot [P_{k-1}(x)]^2\, dx}{\int_{-1}^{1} 1 \cdot [P_{k-1}(x)]^2\, dx}
$$

$$
C_k = \frac{\int_{-1}^{1} x \cdot 1 \cdot [P_{k-1}(x)P_{k-2}(x)]\, dx}{\int_{-1}^{1} 1 \cdot [P_{k-2}(x)]^2\, dx}
$$

Figure 9: In gray the intuitive monomials $\phi_i(x) = x^i$ with the Legendre polynomials $P_i(x)$ in black.



$$
\begin{aligned}
T_0(x) &= 1\\
T_1(x) &= x\\
T_2(x) &= 2x^2 - 1\\
T_3(x) &= 4x^3 - 3x\\
&\dots\\
T_n(x) &= 2xT_{n-1}(x) - T_{n-2}(x)
\end{aligned}
$$

Figure 10: In gray the intuitive monomials $\phi_i(x) = x^i$ with the Chebyshev polynomials $T_i(x)$ in black.

Figure 10. The monic Chebyshev polynomials (not illustrated, but handy for Chebyshev economization of power series), with leading coefficient 1, are defined by

$$
\tilde{T}_n(x) = \frac{1}{2^{n-1}} T_n(x)
$$

For these polynomials the extrema, which occur *at* the same points, are those of $T_n(x)$ but reduced in value by a factor of $\frac{1}{2^{n-1}}$, that is, the extrema are, for $k = 0, 1, \dots, n$

$$
\tilde{T}_n'(\bar{x}_k) = \frac{(-1)^k}{2^{n-1}}
$$

3.3.3 Chebyshev points and Lagrange nodes

By a theorem, we have that the optimal choice of nodes for polynomial approximation by a degree n polynomial is given by the zeros of of the $(n+1)^{\text{st}}$ Chebyshev polynomial $T_{n+1}(x)$. For example, as given in Figure 11, we take the approximation of $f(x) = e^x$ on $[0, 1]$ with three points. We expect to to take, $\tilde{x}_0 = 0$, $\tilde{x}_1 = 0.5$, and $\tilde{x}_2 = 1.0$, but achieve better performance with Chebyshev nodes $\bar{x}_i$. The nodes for interpolation are given by the zeros of the Chebyshev polynomial $T_3(x)$, which in turn are given by $\bar{x}_k = \cos\left(\frac{2k-1}{2n}\pi\right)$ for $k = 1, 2, 3$.

$$\begin{aligned}
\bar{x}_1 &= \cos\left(\frac{2(1)-1}{2(3)}\cdot\pi\right) = \cos\left(\frac{1\cdot\pi}{6}\right) = \frac{\sqrt{3}}{2} \approx 0.866 \\
\bar{x}_2 &= \cos\left(\frac{2(2)-1}{2(3)}\cdot\pi\right) = \cos\left(\frac{3\cdot\pi}{6}\right) = 0 \\
\bar{x}_3 &= \cos\left(\frac{2(3)-1}{2(3)}\cdot\pi\right) = \cos\left(\frac{5\cdot\pi}{6}\right) = -\frac{\sqrt{3}}{2} \approx -0.866
\end{aligned}$$

From $\tilde{x}_i \in [a, b]$ we can compute $\bar{x}_i \in [-1, 1]$ by

$$x_i = \frac{2\tilde{x}_i - a - b}{b-a}$$

and in the reverse we can use

$$\tilde{x}_i = \frac{1}{2}\left((b-a)\bar{x}_i + a + b\right)$$

As an example,

i	$\bar{x}_i$	$\tilde{x}_i$
1	0.866	0.933
2	0.0	0.500
3	−0.866	0.067

The näively chosen nodes that include the endpoints (red) are illustrated in the left-hand panel of Figure 11, and as shown give a reasonable approximate. Yet, the approximation is improved by use of the Chebyshev points (blue). Notice, from Figure 12, that the zeros accumulate or cluster near the boundary of the interval. This helps to 'clamp down' the polynomial interpolation near the boundary.

3.3.4 Chebyshev economization

As we will soon see, the 'best' reduced order polynomial approximation P_{n-1} to a polynomial $P_n(x)$ is given by

$$P_{n-1}(x) = P_n(x) - a_n\tilde{T}_n(x)$$

where $\tilde{T}_n(x)$ is the n^{th} order monic Chebyshev polynomial and a_n is the coefficient of the highest order term in $P_n(x)$. To economize the polynomial approximation to $f(x) = e^x$ given by a 4^{th} order Maclaurin polynomial, we can subtract $\frac{1}{24}\tilde{T}_4(x)$ from the original $P_4(x)$ (note that $a_4 = \frac{1}{24}$ in the original polynomial representation). This alters coefficients of all powers of x of lower orders that are present in $T_4(x)$ and eliminates the 4^{th} order term entirely. Notice in Figure 13 that the maximum error for the third order function is only slightly worse for $x = 1.0$ than for the fourth order function.

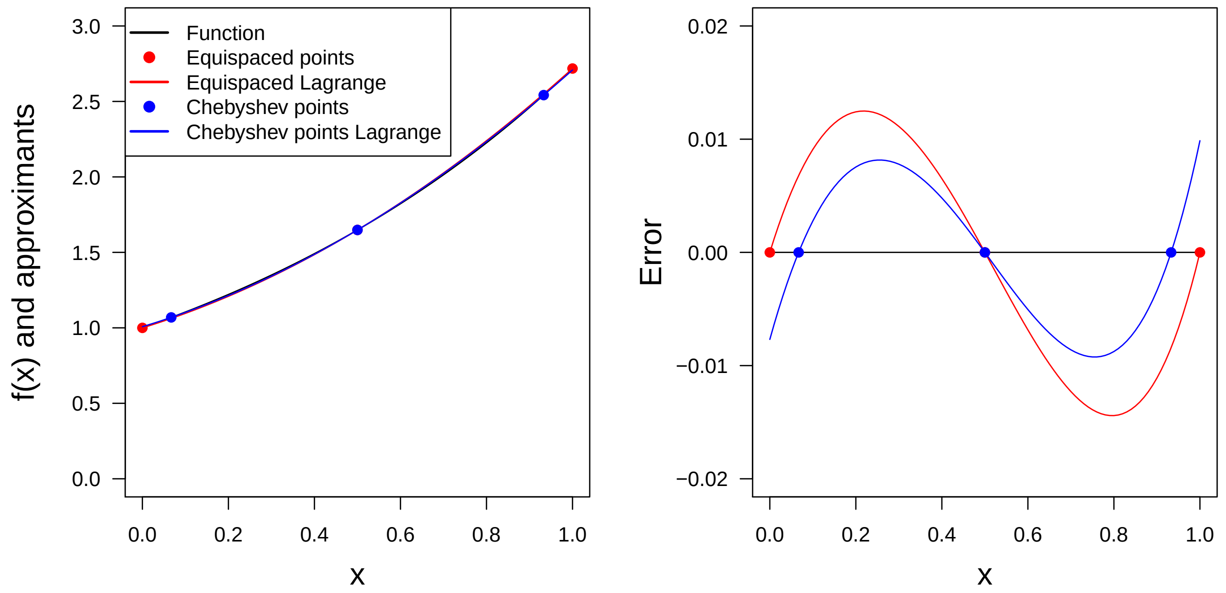


Figure 11: Lagrange interpolation of $f(x) = e^x$ on $[0, 1]$, but with Chebyshev points instead of equispaced points.

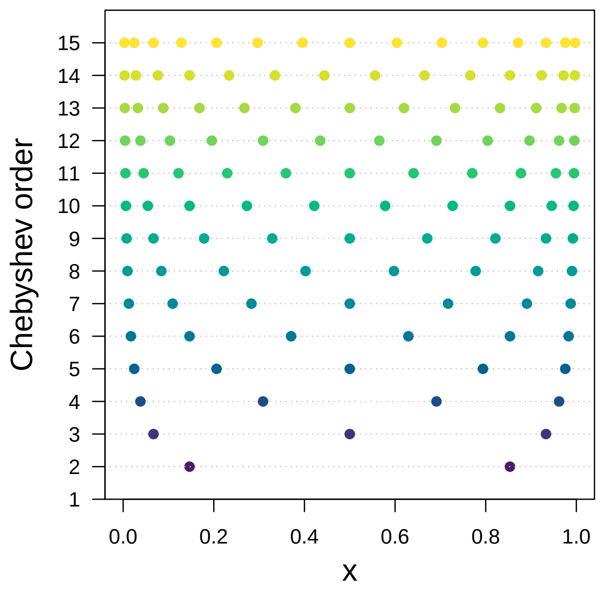


Figure 12: Location of zeros to $T_n(x)$ in the interval $[0, 1]$ for Chebyshev functions of orders from $n = 2, 3, \ldots, 15$.

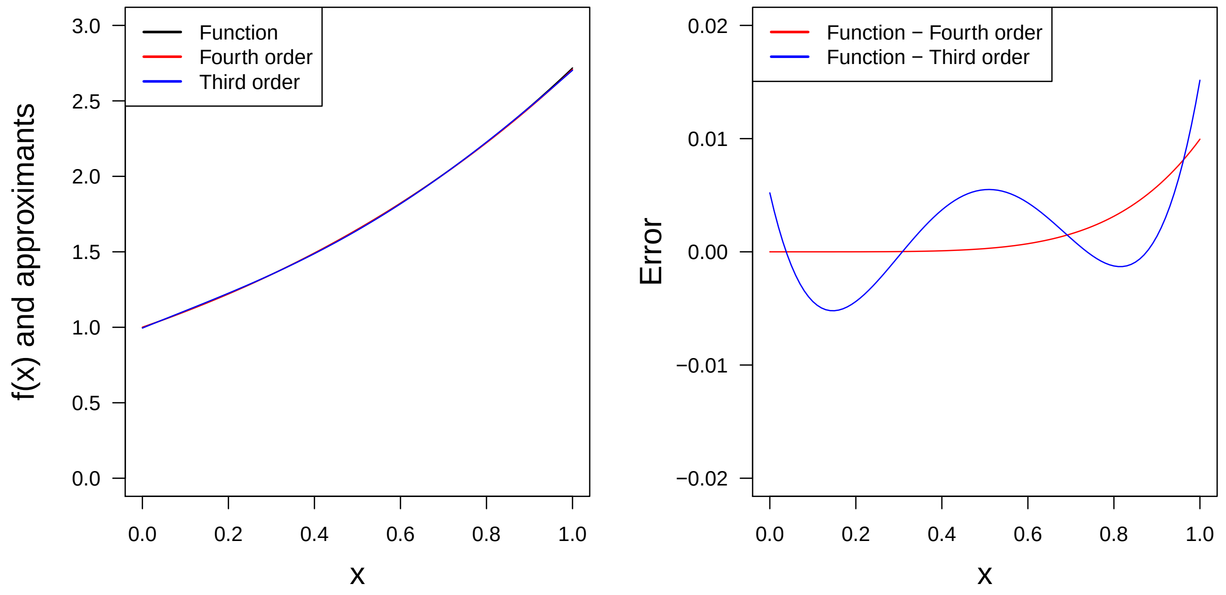


Figure 13: Left: Full- and reduced-order approximations of $f(x) = e^x$ on $[0, 1]$ by $P_3(x) = P_4(x) - a_4\tilde{T}_4(x)$. Right: Error in approximations of $f(x) = e^x$ by full- and reduced-order polynomials.

4 Numerical calculus

4.1 Numerical differentiation

Consider the function $f(x) = e^x$ at a point $x_0 = 1$ and approximations to its derivative $f'(x_0) \approx 2.71281$. Approximations can be made by to somewhat straightforward approximations: the first-order forward difference and the second-order forward and centered differences. The error in these approximations is illustrated below, with rounding to simulate calculations by single-precision and double-precision numbers.

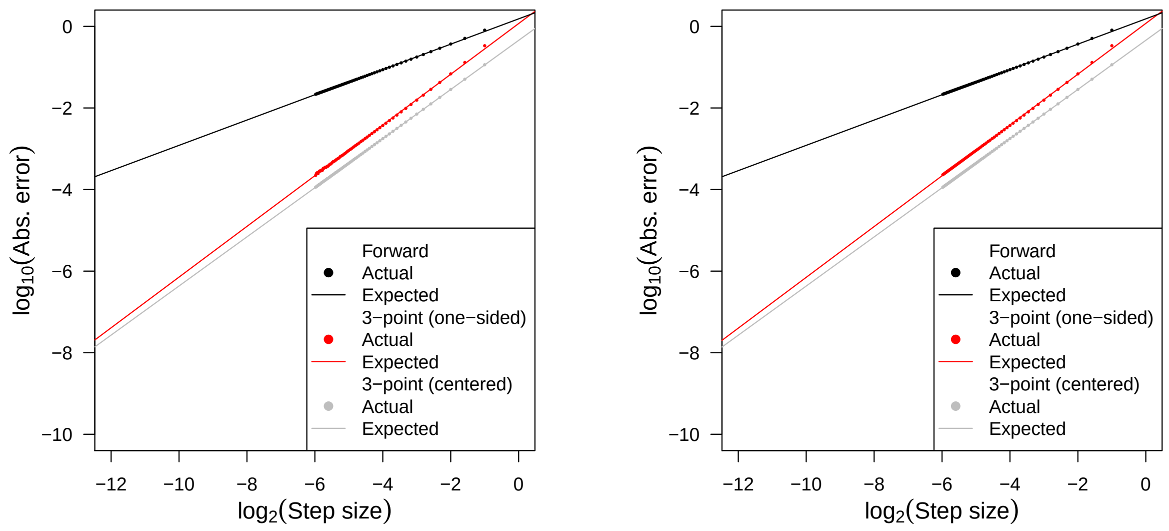


Figure 14: Points indicate calculated $\log_{10}$(abs. error) for first-order forward (black), second-order forward (red), and centered (gray) difference approximations. Line illustrates predicted reduction in error as step size shrinks. **Left:** 8 digit calculation (simulated single-precision), **Right:** 16 digit calculation (simulated double-precision)

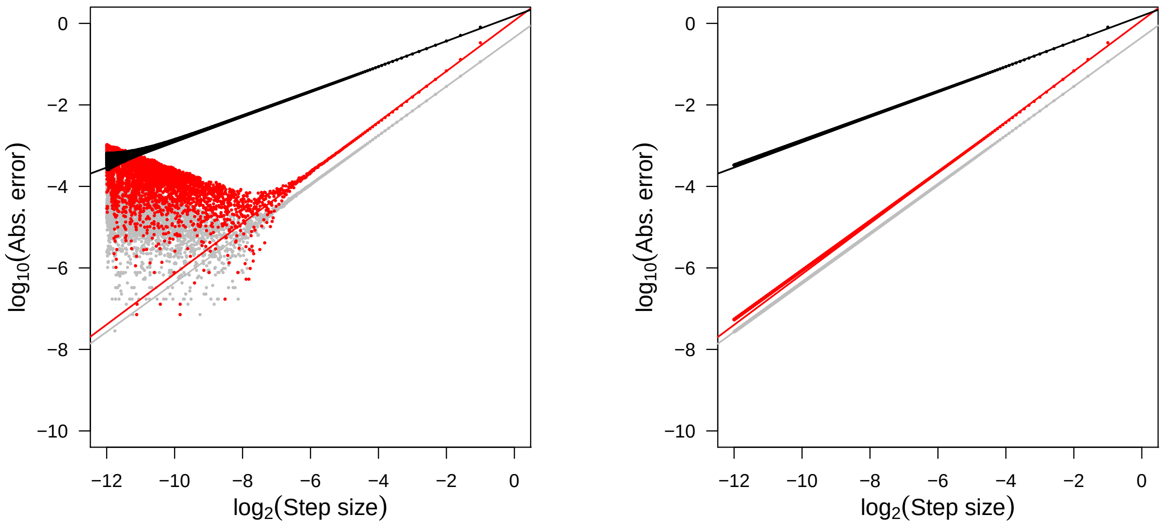


Figure 15: As in Figure 14, but with smaller step sizes included. Note that error *increases* due to roundoff for both methods, but especially for the centered method as the step size is reduced below $h = 0.01$.

4.2 Numerical integration (Monte Carlo)

This section may be imperfect, but reflects the main idea.

Though not emphasized here, the techniques that follow are most useful in higher dimensional integrals. We can integrate $f(x) = x^2$ using the Fundamental Theorem of Calculus,

$$\int_0^2 x^2\,dx = \left(\frac{x^3}{3}\right)\Bigg|_0^2 = \frac{2^3}{3} - \frac{0^3}{3} = \frac{8}{3}$$

From the Mean Value Theorem for Integrals we have that

$$\hat{f} = \frac{1}{2-0}\int_0^2 x^2\,dx = \frac{1}{2}\cdot\frac{8}{3} = \frac{4}{3}$$

The shaded region on the left panel (Fig 16) shows the area under the curve. The area on the right is the same size, but bounded by a constant function whose value is $\hat{f}$.

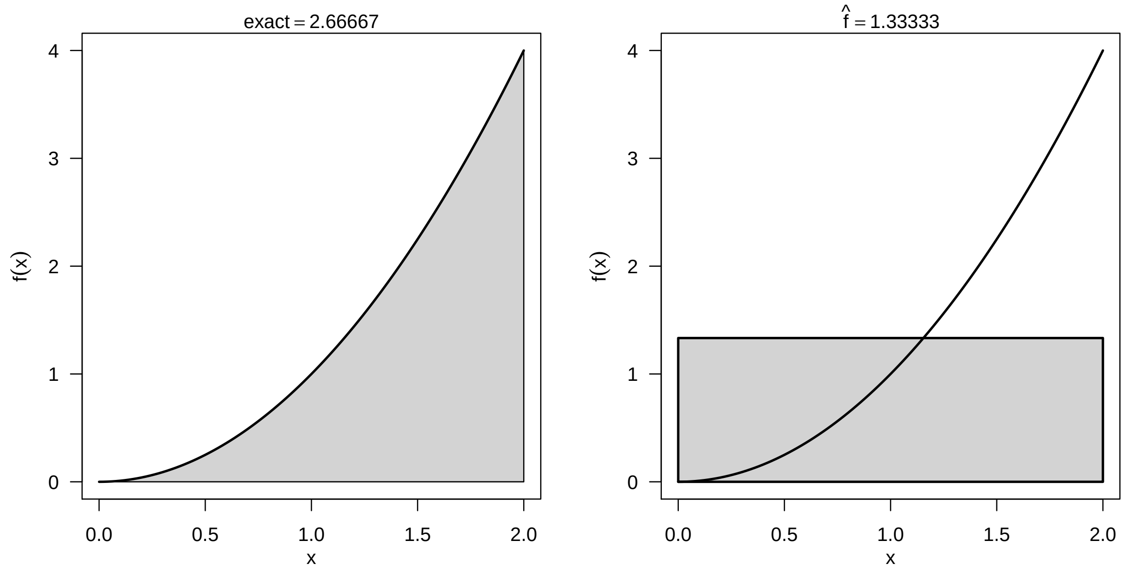


Figure 16: Left: Traditional calculus interpretation of the area under the curve. Right: Average value of a function defined by equal area.

One scheme of Monte Carlo integration uses function values at randomly chosen points to calculate areas which are then averaged. Here

$$\int_a^b f(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N}\underbrace{(b-a)f(x_i)}_{i\text{th area}} = (b-a)\underbrace{\frac{1}{N}\sum_{i=1}^{N}f(x_i)}_{\text{average height}}$$

Fig 17 (left panel) shows 5 of the 100 random rectangle regions. The shaded area on the right panel corresponds to the region whose height is the average of the random function values. Notice that this height very closely compares with the average value of the function, thus the areas of the regions are similar.

An alternate scheme (Fig 18, left) plots points in the plane and considers the fraction of points that fall below the curve (i.e., ‘the acceptance region‘). This fraction is multiplied by the total area of

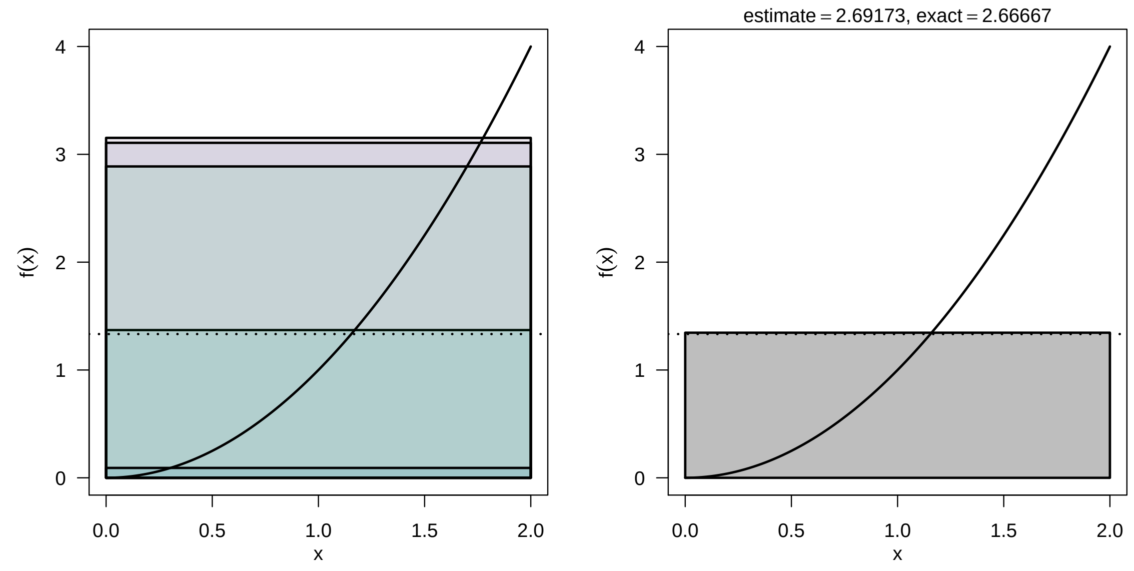


Figure 17: Dashed line indicates $\hat{f}$ from traditional calculus. Left: 5 sample rectangles (shaded). Right: Average area and function height by random sampling.

the region (easier to calculate since the shape is likely square). Though this is easier to program than the notation might suggest, consider the indicator function

$$I(i) = \begin{cases} 1, & y_i < f(x_i) \\ 0, & \text{otherwise} \end{cases}$$

and take

$$\int_a^b f(x)\,dx \approx (f_{max} - f_{min})(b-a)\frac{1}{N}\sum_{i=1}^{N} I(i)$$

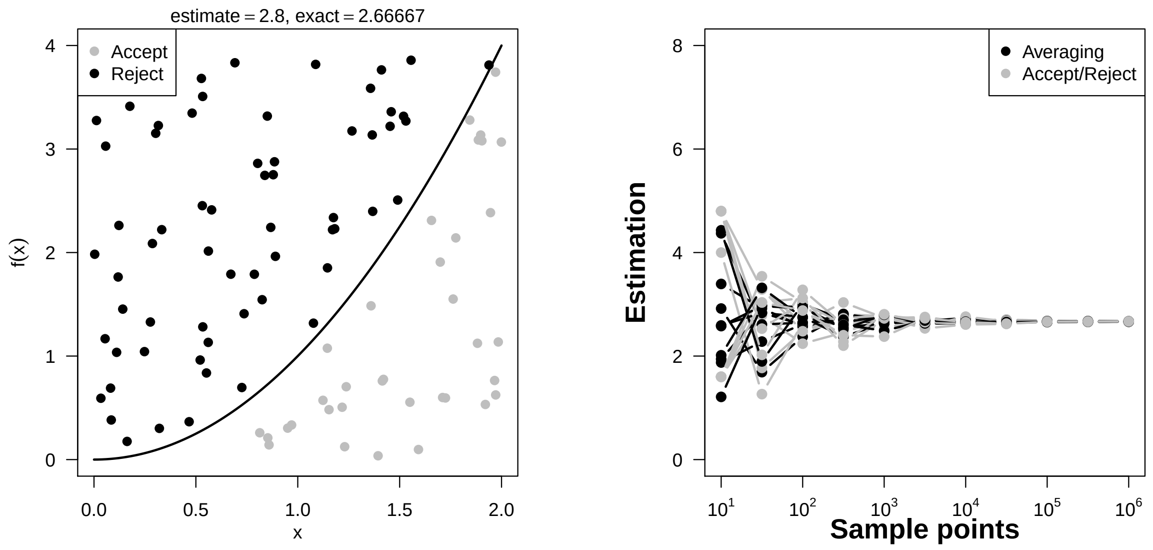


Figure 18: Left: Accepted points fall below the graph of the function. Right: Comparison of rectangle method and rejection method for ten different trials of each.