Rochester Institute of Technology

Numerical Analysis Qualifier Review

Mario L. Gutierrez Abed

January 19, 2021

Contents

0 Preliminaries

Theorem 1 (**Intermediate Value Theorem**). *Let f be C^0 on the interval $[a,b]$. Then f realizes every value between $f(a)$ and $f(b)$. More precisely, if y is a number between $f(a)$ and $f(b)$, then there exists a number $\xi \in [a,b]$ such that $f(\xi) = y$.*

Theorem 2 (**Generalized Intermediate Value Theorem**). *Let f be C^0 on the interval $[a,b]$. Let $x_0, \dots, x_n$ be points in $[a,b]$, and $a_0, \dots, a_n > 0$. Then there exists a number $\xi \in [a,b]$ such that*

$$(a_0 + \cdots + a_n) f(\xi) = a_0 f(x_0) + \cdots + a_n f(x_n). \tag{0.1}$$

Theorem 3 (**Mean Value Theorem**). *Let f be C^1 on an interval (a,b). Then there exists a number $\xi \in (a,b)$ such that*

$$f'(\xi) = \frac{f(b) - f(a)}{b - a}. \tag{0.2}$$

A corollary of the *Mean Value Theorem* is the following:

Corollary 1 (**Rolle's Theorem**). *Let f be C^1 on (a,b) and assume that $f(a) = f(b)$. Then there exists a number $\xi \in (a,b)$ such that $f'(\xi) = 0$.*

Theorem 4 (**Generalized Rolle's Theorem**). *Let f be C^n on the interval (a,b). If f is zero at $n+1$ distinct points $x_0, \dots, x_n$ in $[a,b]$, then there exists a number $\xi \in (a,b)$ such that $f^{(n)}(\xi) = 0$.*

Theorem 5 (**Weighted Mean Value Theorem for Integrals**). *Suppose that f is continuous on $[a,b]$. Let g be another function such that its Riemann integral exists, and g does not change sign on $[a,b]$. Then, there exists a number $\xi \in (a,b)$ that satisfies*

$$\int_a^b f(x) g(x) \, \mathrm{d}x = f(\xi) \int_a^b g(x) \, \mathrm{d}x. \tag{0.3}$$

Definition 1. *A solution is said to be **correct within p decimal places** if the error is less than 0.5×10^{-p}.*

1 Root Finding

In this section we cover the four main root-finding techniques we covered in class, namely the *Bisection Method*, *Fixed Point Iteration*, *Newton's Method*, and the *Secant Method*.

1.1 Bisection Method

Bisection Method

```
Given initial interval $[a, b]$ such that $f(a)f(b) < 0$
while $(b - a)/2 >$ TOL
    $c = (a + b)/2$
    if $f(c) = 0$, stop, end
    if $f(a)f(c) < 0$
        $b = c$
    else
        $a = c$
    end
end
The final interval $[a, b]$ contains a root.
The approximate root is $(a + b)/2$.
```

Example 1. *Apply two steps of the bisection method on*

$$x^5 + x = 1. \tag{1.1}$$

to find an approximate root within $1/8$ of the true root.

Solution: *Since $f(x) = x^5 + x - 1$ is continuous on, say, the interval $[0, 1]$, it realizes every value between $f(0)$ and $f(1)$; in other words, there must be some $c \in [0, 1]$ for which $f(c) \in [f(0), f(1)]$. In fact, since $f(0) \cdot f(1) = -1 \cdot 1 = -1$, the function changes sign in this interval and therefore there must be a root $r \in [0, 1]$. Thus $[0, 1]$ is a suitable interval of length one which contains a root.*

Now let us iterate the bisection method for two steps, so that we may find an approximate root within $1/2^{2+1} = 1/8$ of the true root r:

- *Define the midpoint $c = (a + b)/2 = (0 + 1)/2 = 1/2$;*
- *Test whether c is a root: $f(c) = f(1/2) = -15/32 \neq 0$; $\times$*
- *Test whether there is a change in signs between the function evaluated at the midpoint and at one of the original endpoints, say, b: $f(b) \cdot f(c) = f(1) \cdot f(1/2) = 1 \cdot (-15/32) = -15/32$;*
- *Since f changes signs in the interval $[c, b]$, the root must be here. Thus we relabel $a = c$;*
  [End of Iteration 1]
- *Define the new midpoint $c = (a + b)/2 = (1/2 + 1)/2 = 3/4$;*

- *Test whether c is a root:* $f(c) = f(3/4) = -13/1024 \neq 0$; $\times$
- *Test whether there is a change in signs between the function evaluated at the newly defined midpoint and at one of the endpoints, say, b:* $f(b) \cdot f(c) = f(1) \cdot f(3/4) = 1 \cdot (-13/1024) = -13/1024$;
- *Since f changes signs in the interval $[c, b]$, the root must be here. Thus we relabel $a = c$;*
  [End of Iteration 2]

Hence, after two iterations we obtained the approximate root $r = 3/4 = 0.75$. □

1.2 Fixed Point Iteration

Fixed-Point Iteration

$$x_0 = \text{initial guess}$$
$$x_{i+1} = g(x_i) \textbf{ for } i = 0, 1, 2, \ldots$$

My C++ function for implementing FPI is given here:

```cpp
void FPI (double (*func)(double), double x0, double epsilon, int itmax){
    double x {x0};    //initialization of variable x to our initial guess x0
    int it {0};
    for (int i {0}; i <= itmax; i++){
        x = func(x);
        it += 1;
        cout << "Iteration # " << it << "." << endl;
        if (abs(func(x) - x) <= epsilon) {
            break;
        }
    }
    cout << "The solution (that is, the fixed point) is x = " << x << "." <<
    endl;
}
```

Definition 2. *The real number r is a **fixed point** of the function g if $g(r) = r$.*

Definition 3. *Let r be the root that we are searching for, and let $e_i = |x_i - r|$ denote the error at the i^{th} step of an iterative method. If*

$$\lim_{i \to \infty} \frac{e_{i+1}}{e_i} = S < 1, \tag{1.2}$$

*then the method is said to obey **linear convergence** with rate S.*

Theorem 6. *Assume that g is continuously differentiable, r is a fixed point of g (i.e., $g(r) = r$), and that $S = |g'(r)| < 1$. Then the FPI method converges linearly with rate S to the fixed point r for initial guesses sufficiently close to r.*

Definition 4. *An iterative method is said to be **locally convergent** to r if the method converges to r for initial guesses sufficiently close to r.*

Example 2. *Derive three different $g(x)$ for finding roots to six correct decimal places of $f(x) = 2x^3 - 6x - 1$ by FPI (that is, convert it to the form $x = g(x)$). Run FPI for each $g(x)$ and report results, convergence or divergence. The equation $f(x) = 0$ has three roots. Derive more $g(x)$ if necessary until all roots are found by FPI. For each convergent run, determine the value of S from the errors e_{n+1}/e_n, and compare with S from calculus (that is, compare it with $S = |g'(r)|$).*

Solution: *We set $f(x) = 0$ and rewrite it in the following three ways*

$$x = \frac{1}{3}x^3 - \frac{1}{6} \equiv g_1(x); \tag{1.3a}$$

$$x = \sqrt[3]{3x + \frac{1}{2}} \equiv g_2(x); \tag{1.3b}$$

$$x = \frac{1}{2(x^2 - 3)} \equiv g_3(x). \tag{1.3c}$$

We now report results for all three functions, using the C++ FPI code written above, except that now we also ask the code to give us the values x_i at each step, since we want to determine the value of S from the errors at the end.

- *For g_1, starting with initial guess $x_0 = 0.3$,*

```
FPI(g, 0.3, 5.0 * 10e-8, 100);
```

we get excellent results:

```
Iteration # 0. x = 0.3.
Iteration # 1. x = -0.157667.
Iteration # 2. x = -0.167973.
Iteration # 3. x = -0.168246.
The solution (that is, the fixed point) is x = -0.168254.
Program ended with exit code: 0
```

It only took three iterations to find our fixed point solution to six correct decimal places! That means that S must be very small; let us see now both variants of this value:

$$S = \lim_{i\to\infty} \frac{e_{i+1}}{e_i} \approx \frac{e_3}{e_2} = \frac{|x_3 - r|}{|x_2 - r|} = \frac{|-0.168246 - (-0.168254)|}{|-0.167973 - (-0.168254)|} = 0.0284698; \tag{1.4a}$$

$$S = |g_1'(r)| = |g_1'(-0.168254)| = 0.0283094. \tag{1.4b}$$

- *We now attempt g_2, starting with initial guess $x_0 = 1.3$,*

```
FPI(g, 1.3, 5.0 * 10e-8, 100);
```

Here we also get convergence, although not as fast as with g_1 (but still, quite good!):

```
Iteration # 0. x = 1.3.
Iteration # 1. x = 1.638643.
Iteration # 2. x = 1.756134.
Iteration # 3. x = 1.793433.
Iteration # 4. x = 1.804955.
Iteration # 5. x = 1.808485.
Iteration # 6. x = 1.809564.
Iteration # 7. x = 1.809893.
Iteration # 8. x = 1.809994.
Iteration # 9. x = 1.810024.
Iteration # 10. x = 1.810034.
Iteration # 11. x = 1.810037.
The solution (that is, the fixed point) is x = 1.810038.
Program ended with exit code: 0
```

This time it took eleven iterations to find our fixed point solution to six correct decimal places. Thus we expect S to have a larger value than that of g_1:

$$S \approx \frac{|x_{11} - r|}{|x_{10} - r|} = \frac{|1.810037 - 1.810038|}{|1.810034 - 1.810038|} = 0.25; \tag{1.5a}$$

$$S = |g_2'(1.810038)| = 0.305228. \tag{1.5b}$$

We notice some larger disparity now between the two variants of S, although this can commonly happen due to the truncation of $\lim_{i\to\infty}$ that must happen in a finite grid ... The important thing here is that both variants show $S < 1$, which ensures convergence.

- *On to g_3, starting with initial guess $x_0 = -1.3$,*

```
FPI(g, -1.3, 5.0 * 10e-8, 100);
```

This one also yield fantastic results, but we have a problem... the root we find with g_3 is the same one we found earlier with g_1:

```
Iteration # 0. x = -1.3.
Iteration # 1. x = -0.381679.
Iteration # 2. x = -0.175173.
Iteration # 3. x = -0.168389.
Iteration # 4. x = -0.168257.
The solution (that is, the fixed point) is x = -0.168254.
Program ended with exit code: 0
```

It merely took four iterations to find our fixed point solution to six correct decimal places. Thus S value must be very small; indeed,

$$S \approx \frac{|x_4 - r|}{|x_3 - r|} = \frac{|-0.168257 - (-0.168254)|}{|-0.168389 - (-0.168254)|} = 0.0222222; \tag{1.6a}$$

$$S = |g_3'(-0.168254)| = 0.0190528. \tag{1.6b}$$

We are still missing one root; let us attempt another fixed-point function, g_4, and see if it yields the missing root. We rewrite $f(x) = 0$ as $2x^3 - 6x = 1$ and add $3x^3$ to both sides to get

$$x = \frac{3x^3 + 1}{5x^2 - 6} \equiv g_4(x). \tag{1.7}$$

Then, with initial guess $x_0 = -2$,

```
FPI(g, -2.0, 5.0 * 10e-8, 100);
```

we get an excellent result that also happens to find our missing root!:

```
Iteration # 0. x = -2.
Iteration # 1. x = -1.642857.
Iteration # 2. x = -1.641398.
Iteration # 3. x = -1.641923.
Iteration # 4. x = -1.641733.
Iteration # 5. x = -1.641802.
Iteration # 6. x = -1.641777.
Iteration # 7. x = -1.641786.
Iteration # 8. x = -1.641783.
The solution (that is, the fixed point) is x = -1.641784.
Program ended with exit code: 0
```

After eight iterations we found our missing fixed point solution to six correct decimal places. Let us check on the S values:

$$S \approx \frac{|x_8 - r|}{|x_7 - r|} = \frac{|-1.641783 - (-1.641784)|}{|-1.641786 - (-1.641784)|} = 0.5; \tag{1.8a}$$

$$S = |g_2'(-1.641784)| = 0.360485. \tag{1.8b}$$

One peculiar aspect of g_4 is that, unlike our previous choices, it actually yields all three roots(!) depending on where you start the algorithm. For instance, had we started with initial guess $x_0 = 0$,

```
FPI(g, 0.0, 5.0 * 10e-8, 100);
```

we get convergence to the root $r = -0.168254$ after just two iterations!:

```
Iteration # 0. x = 0.
Iteration # 1. x = -0.166667.
Iteration # 2. x = -0.168246.
The solution (that is, the fixed point) is x = -0.168254.
Program ended with exit code: 0
```

Had we instead started at $x_0 = 2$,

```
FPI(g, 2.0, 5.0 * 10e-8, 100);
```

we get convergence to the root $r = 1.810038$ after ten iterations:

```
Iteration # 0. x = 2.
Iteration # 1. x = 1.785714.
Iteration # 2. x = 1.818478.
Iteration # 3. x = 1.807462.
Iteration # 4. x = 1.810859.
Iteration # 5. x = 1.80978.
Iteration # 6. x = 1.81012.
Iteration # 7. x = 1.810012.
Iteration # 8. x = 1.810046.
Iteration # 9. x = 1.810035.
Iteration # 10. x = 1.810039.
The solution (that is, the fixed point) is x = 1.810038.
Program ended with exit code: 0
```

Anyhow, we have found all three roots of $f(x) = 2x^3 - 6x - 1$ by using our very own FPI code. We conclude by showing a graph of f, showcasing all three roots we just found.

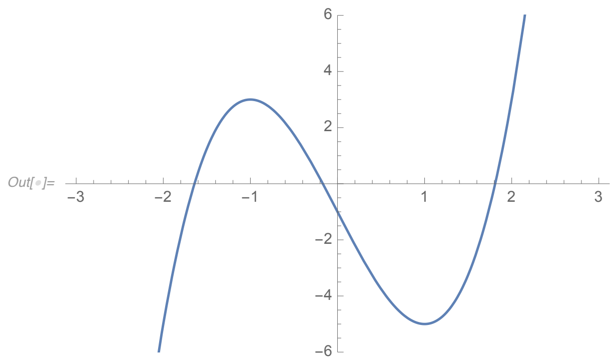


Figure 1.1: The function $f = 2x^3 - 6x - 1$ and its three roots, $\{-1.641784, -0.168254, 1.810038\}$.

$\square$

1.3 Newton's Method

Newton's Method

$$x_0 = \text{initial guess}$$
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad \textbf{for} \quad i = 0, 1, 2, \ldots.$$

Newton's method is among the most popular methods to solve

$$f(x) = 0.$$

The idea is to choose an initial guess x_0 and draw a tangent line to the function f at $(x_0, f(x_0))$. The tangent line will approximately follow the function down to the x-axis towards the root. The intersection point of the tangent line with the x-axis is an approximate root. This geometric idea defines an iterative scheme.
The equation of the tangent line is:

$$y - f(x_0) = f'(x_0)(x - x_0).$$

By substituting $y = 0$, we obtain

$$f'(x_0)(x - x_0) = 0 - f(x_0) \quad \Rightarrow \quad x = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Here is my C++ function that implements Newton's Method:

```cpp
void Newton (double (*func)(double), double (*funcder)(double), double x0,
             double epsilon = 0.5 * 1e-8, int itmax = 100){
    double x {x0};    //initialization of variable x to our initial guess x0
    double newx {};  //initialization of variable newx

    for (int i {0}; i <= itmax; i++){
        cout << "Iteration # " << i << "." << " x = " << x << "." << endl;
        newx = x - ( func(x) )/( funcder(x) );
        if (abs(newx - x) <= epsilon) {
            cout << "Newton Method has converged within given tolerance.
                     The root found is x = " << newx << "." << endl;
            break;
        }
        if (i == itmax) {
            cout << "Newton Method has failed to converge after the maximum
                    allowed number of iterations. " << endl;
        }
        x = newx;
    }
}
```

Definition 5. *Assume that r is a root of the differentiable function f. Then, if the m^{th} derivative is the first non-vanishing one, i.e., if*

$$0 = f(r) = f'(r) = f''(r) = \cdots = f^{(m-1)}(r), \qquad \text{but } f^{(m)}(r) \neq 0,$$

then we say that f has a ***root of multiplicity*** m *at r. We say that f has a* ***multiple root*** *at r if the multiplicity is greater than one. The root is instead called a* ***simple root*** *if its multiplicity is one; i.e., if $f(r) = 0$ but $f'(r) \neq 0$.*

Definition 6. *Let r be the root that we are searching for, and let $e_i = |x_i - r|$ denote the error after the i^{th} step of an iteration method. Then the iteration is said to be* ***quadratically convergent*** *if*

$$M = \lim_{i\to\infty} \frac{e_{i+1}}{e_i^2} < \infty. \tag{1.9}$$

Theorem 7. *Let f be twice continuously differentiable and $f(r) = 0$. If $f'(r) \neq 0$ (i.e., if r is a simple root of f), then Newton's Method is locally and quadratically convergent to r. The error e_i at step i satisfies*

$$\lim_{i\to\infty} \frac{e_{i+1}}{e_i^2} = M < \infty, \tag{1.10}$$

where

$$M = \left| \frac{f''(r)}{2f'(r)} \right|. \tag{1.11}$$

Proof. We start by showing the local (linear) convergence of Newton's Method, writing Newton's method in FPI form and following Theorem 6: Let

$$x_{n+1} = g(x_n),$$

where

$$g(x) = x - \frac{f(x)}{f'(x)}.$$

Then, differentiating,

$$g'(x) = 1 - \frac{(f'(x))^2 - f(x)f''(x)}{f'(x))^2} = \frac{f(x)f''(x)}{f'(x))^2}.$$

But then, since r is a root,

$$g'(r) = \frac{\overbrace{f(r)}^{=0} f''(r)}{f'(r))^2} = 0.$$

Thus, we have $S = |g'(r)| = 0 < 1$, which means (from Theorem 6) that Newton's Method converges linearly.

In fact, we will now show that the convergence of Newton's Method is more than just linear; it is in fact quadratic (i.e., it satisfies Eq. (1.9)). We start by Taylor-expanding f to first order in a neighborhood of the root r,

$$f(r) = f(x_n) + (r - x_n)f'(x_n) + \frac{(r - x_n)^2}{2} f''(\xi_n),$$

where ξ_n is somewhere between x_n and r. But then, since r is a (simple) root,

$$\begin{aligned}
0 &= f(x_n) + (r - x_n)f'(x_n) + \frac{(r - x_n)^2}{2} f''(\xi_n) \\
\implies -\frac{f(x_n)}{f'(x_n)} &= r - x_n + \frac{(r - x_n)^2}{2}\frac{f''(\xi_n)}{f'(x_n)} \\
\implies \overbrace{x_n - \frac{f(x_n)}{f'(x_n)}}^{=x_{n+1}} - r &= \frac{(r - x_n)^2}{2}\frac{f''(\xi_n)}{f'(x_n)} \\
\implies \overbrace{x_{n+1} - r}^{\text{the abs} = e_{n+1}} &= \frac{\overbrace{(r - x_n)^2}^{=e_n^2}}{2}\frac{f''(\xi_n)}{f'(x_n)} \\
\implies \frac{e_{n+1}}{e_n^2} &= \left|\frac{f''(\xi_n)}{2f'(x_n)}\right|.
\end{aligned}$$

Thus, since $\lim_{n\to\infty} x_n = \lim_{n\to\infty} \xi_n = r$, we have

$$\lim_{n\to\infty} \frac{e_{n+1}}{e_n^2} = \left|\frac{f''(r)}{2f'(r)}\right|,$$

as we wanted to prove. □

The following example shows the issues that arise when we are dealing with a multiple root: in such cases Theorem 7 does not apply, and we end up with linear convergence instead. The example also shows a way to remedy this issue by slightly modifying Newton's Method.

Example 3. *Consider the function* $f(x) = e^{\sin^3 x} + x^6 - 2x^4 - x^3 - 1$ *on the interval* $[-2,2]$. *Plot the function on the interval, and find all three roots to six correct decimal places. Determine which roots converge quadratically, and find the multiplicity of the roots that converge linearly.*

Solution: *Here is the graph of the function:*

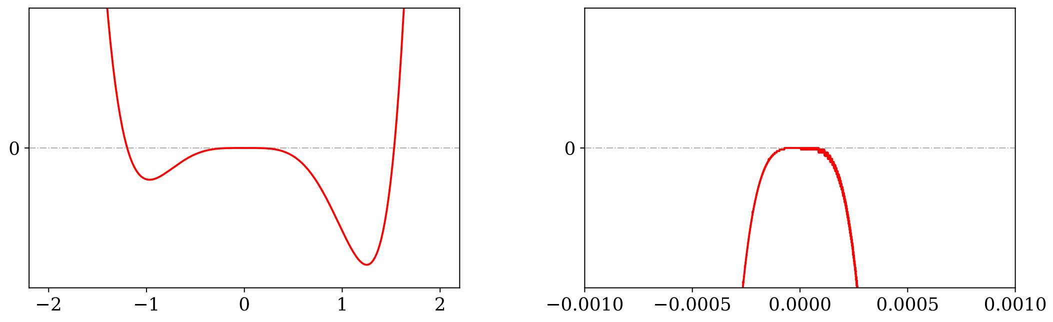


Figure 1.2: Graph of $f(x) = e^{\sin^3 x} + x^6 - 2x^4 - x^3 - 1$ on the interval $[-2,2]$ (left). We can see that it has three roots, one of which has nontrivial multiplicity. In fact, from the figure on the left one may be mislead to think that the function vanishes in a whole interval surrounding $x = 0$, but as the zoom-in on the right figure shows, that is not the case.

We can tell from just looking at the plot that Newton's Method will fail badly if we start with initial guess $x_0 = 0$, *since the tangent (and hence the derivative) there vanishes and, consequently, Newton's Method attempts to divide by zero. The algorithm is divergent in this case, albeit we can apply it in the vicinity and get convergence. Before tackling the more complicated root* $x = 0$, *let us look at the other two, for which we easily get convergence:*

```
double f (double x){
    return exp( pow(sin(x), 3) ) + pow(x,6) - ( 2.0 * pow(x,4) ) - pow(x,3) -
     1.0;
}

double fder (double x){
    return ( cos(x) * pow(sin(x), 2) * 3.0 * exp( pow(sin(x), 3) ) ) + ( 6.0
    * pow(x,5) ) - ( 8.0 * pow(x,3) ) - ( 3.0 * pow(x,2) );
}

int main(int argc, const char * argv[]) {

    cout << setprecision(7);
    Newton(f, fder, -1.0, 0.5 * 1e-6);

    return 0;
}
```

which has output

```
Iteration # 0. x = -1.
Iteration # 1. x = -2.221536.
Iteration # 2. x = -1.896474.
Iteration # 3. x = -1.637268.
Iteration # 4. x = -1.438844.
Iteration # 5. x = -1.300688.
Iteration # 6. x = -1.224206.
Iteration # 7. x = -1.199918.
Iteration # 8. x = -1.197643.
Iteration # 9. x = -1.197624.
Newton Method has converged within given tolerance. The root found is x =
    -1.197624.
Program ended with exit code: 0
```

Thus, with starting guess $x_0 = -1$, Newton's Method has found the root $r = -1.197624$ after just nine iterations. If we instead choose initial guess $x_0 = 2$, we get another root after just six iterations:

```
Iteration # 0. x = 2.
Iteration # 1. x = 1.779275.
Iteration # 2. x = 1.629974.
Iteration # 3. x = 1.552496.
Iteration # 4. x = 1.531542.
Iteration # 5. x = 1.53014.
Iteration # 6. x = 1.530134.
Newton Method has converged within given tolerance. The root found is x =
    1.530134.
Program ended with exit code: 0
```

These two are cases of simple roots, *since $f'(r) \neq 0$ in either case. The function is also twice continuously differentiable, with $f''(r)$ having values*

```
In[1]:=
f[x_] := E^(Sin[x])^3 + x^6 - 2 x^4 - x^3 - 1;
fprime[x_] := D[f[x], x];
D[fprime[x], x] /. x -> -1.197624
D[fprime[x], x] /. x -> 1.530134

Out[3]= 35.6295
Out[4]= 91.0323
```

In such situations, we get very fast (that is, quadratic) *convergence, as Theorem 7 suggests. Let us now evaluate this M value (both the exact value and the truncated error ratio) for both single roots:*

- *For $r = -1.197624$,*

$$M \approx \frac{e_8}{e_7^2} = \frac{|-1.197643 - (-1.197624)|}{(|-1.199918 - (-1.197624)|)^2} \approx 3.6 \tag{1.12a}$$

$$M = \left|\frac{f''(-1.197624)}{2f'(-1.197624)}\right| = \left|\frac{35.6295}{2 \cdot (-4.92059)}\right| \approx 3.6. \tag{1.12b}$$

- *For* $r = 1.530134$,

$$M \approx \frac{e_5}{e_4^2} = \frac{|1.530140 - 1.530134|}{(|1.531542 - 1.530134|)^2} \approx 3.03 \tag{1.13a}$$

$$M = \left| \frac{f''(1.530134)}{2f'(1.530134)} \right| = \left| \frac{91.0323}{2 \cdot (14.9728)} \right| \approx 3.04. \tag{1.13b}$$

Now, for the remaining root we apply our code with initial guess $x_0 = 0.5$,

```
Newton(f, fder, 0.5, 0.5 * 1e-6);
```

The output now (showing only the last couple of iterations) is

```
Iteration # 28. x = 0.000155763.
Iteration # 29. x = 0.0001227148.
Iteration # 30. x = 8.516709e-05.
Newton Method has converged within given tolerance. The root found is x =
    8.516709e-05.
Program ended with exit code: 0
```

We see that the method is converging to the root $x = 0$; *due to our set precision, the value* $8.516709 \times 10^{-5} \approx 0.000085$ *is as close as we can get.*[1] *However, note that this time it took 30 iterations to converge to a solution within the given tolerance, while for the previous two roots the method was much faster. This is an example of a* multiple root; *i.e., a root for which* $f'(r) = 0$. *In fact, we can see that* $x = 0$ *is a root of* f *of multiplicity* 4, *since* $0 = f(0) = f'(0) = f''(0) = f^{(3)}(0)$, *but* $f^{(4)}(0) = -48 \neq 0$; *we show this in the following Mathematica snippet:*

```
In[1]:=
f[x_] := E^(Sin[x])^3 + x^6 - 2 x^4 - x^3 - 1;
fprime[x_] := D[f[x], x];
fpprime[x_] := D[fprime[x], x];
fppprime[x_] := D[fpprime[x], x];
fpppprime[x_] := D[fppprime[x], x];

fprime[x] /. x -> 0
fpprime[x] /. x -> 0
fppprime[x] /. x -> 0
fpppprime[x] /. x -> 0

Out[6]= 0
Out[7]= 0
Out[8]= 0
Out[9]= -48
```

[1]Reaching ***exactly*** zero is impossible for a computer with finite precision!

In this case, the following theorem from Sauer's applies:

Theorem 1.12 (Sauer)

Assume that f is C^{m+1} on an interval (a, b) and has a root r of multiplicity m in such interval. Then Newton's Method is locally (linear) convergent to r, and the errors satisfy

$$\lim_{i\to\infty} \frac{e_{i+1}}{e_i} = S, \tag{1.14}$$

where

$$S = \frac{m-1}{m}. \tag{1.15}$$

Let us evaluate the S value (both the exact value and the truncated error ratio) for our root $x = 0$:

$$S \approx \frac{e_{29}}{e_{28}} = \frac{|0.0001227148 - 0|}{|0.000155763 - 0|} \approx 0.79 \tag{1.16a}$$

$$S = \frac{4-1}{4} = \frac{3}{4} = 0.75. \tag{1.16b}$$

How could we have improved our results when dealing with the complications brought about by the $x = 0$ root? Very easily actually, thanks to the Modified Newton Method*:*

Theorem 1.13 (Sauer)

If f is C^{m+1} on an interval (a, b) and has a root r of multiplicity $m > 1$ in such interval. Then the Modified Newton Method

$$x_{i+1} = x_i - \frac{mf(x_i)}{f'(x_i)} \tag{1.17}$$

converges locally and quadratically(!) to r.

Putting eq. (1.17) to good use in our code, with $m = 4$, we get an incredible boost in performance! Check out the insane difference!:

```
Iteration # 0. x = 0.5.
Iteration # 1. x = -0.03166195.
Iteration # 2. x = 7.533888e-05.
Newton Method has converged within given tolerance. The root found is x =
    7.533888e-05.
Program ended with exit code: 0
```

We see that this modified version of Newton's Method converges to the root $x = 0$ after just two iterations! ☐

1.4 Secant Method

If we do not have a derivative of f at hand, we cannot use Newton's Method. The Secant Method comes to rescue in such situations, replacing the derivative $f'(x_i)$ with the difference quotient

$$\frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}.$$

Secant Method

$$\begin{aligned} x_0, x_1 &= \text{initial guesses} \\ x_{i+1} &= x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \textbf{ for } i = 1, 2, 3, \ldots. \end{aligned}$$

The error term for the Secant Method, assuming that r is a simple root, is given by

$$e_{i+1} \approx \left| \frac{f''(r)}{2f'(r)} \right| e_i e_{i-1},$$

which implies

$$e_{i+1} \approx \left| \frac{f''(r)}{2f'(r)} \right|^{\alpha - 1} e_i^{\alpha},$$

where $\alpha = (1 + \sqrt{5})/2 \approx 1.62$. As you can see, the error is not quite linear nor quadratic, but rather something in-between...The convergence is said to be ***superlinear***.

2 Interpolation

This section covers the main interpolation techniques covered in class, namely, *Lagrange Interpolation*, *Divided Differences*,*Cubic Splines*, and the *Chebyshev Polynomials*.

Theorem 8 (**Main Theorem of Polynomial Interpolation**). *Let $(x_0, y_0), \dots, (x_n, y_n)$ be $n+1$ points in the plane with distinct x_i. Then there exists $\underline{\text{one and only one}}$ polynomial p of degree n or less that satisfies $p(x_i) = y_i$ for $i = 0, \dots, n$.*

Theorem 9 (**Interpolation Error Theorem**). *Assume that $p(x)$ is the (degree n or less) interpolating polynomial fitting the $n+1$ points $(x_0, y_0), \dots, (x_n, y_n)$. The interpolation error for some function f is given by*

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i), \tag{2.1}$$

where ξ lies between the smallest and largest of the numbers $x, x_0, \dots, x_n$.

2.1 Lagrange Interpolation & Newton's Divided Differences

The following examples fully demonstrate how to to apply both of these interpolating techniques:

Example 4. *Use Lagrangian interpolation and Newton's divided differences to find interpolating polynomials of the points $(-1,0)$, $(2,1)$, $(3,1)$, $(5,2)$.*

Solution. We now present interpolating polynomials for these points using both methods:

- **Lagrange Interpolation:** Let

$$\ell_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \qquad \text{for } i = 0, \dots, n. \tag{2.2}$$

(Note the peculiar property $\ell_i(x_j) = \delta_{ij}$.) Then the ***Lagrange polynomial*** interpolating points (x_i, y_i), for $i = 0, \dots, n$, is given by

$$^{\mathcal{L}}p_n(x) = \sum_{i=0}^{n} y_i \ell_i(x). \tag{2.3}$$

Thus, in the case at hand,

$$^{\mathcal{L}}p_3(x) = \sum_{i=0}^{3} y_i \ell_i(x)$$

$$
\begin{aligned}
&= 0 \cdot \frac{(x-2)(x-3)(x-5)}{(-1-2)(-1-3)(-1-5)} + 1 \cdot \frac{(x-(-1))(x-3)(x-5)}{(2-(-1))(2-3)(2-5)} \\
&\quad + 1 \cdot \frac{(x-(-1))(x-2)(x-5)}{(3-(-1))(3-2)(3-5)} + 2 \cdot \frac{(x-(-1))(x-2)(x-3)}{(5-(-1))(5-2)(5-3)} \\
&= \frac{x^3-7x^2+7x+15}{9} - \frac{x^3-6x^2+3x+10}{8} + \frac{x^3-4x^2+x+6}{18} \\
&= \frac{1}{24}\left(x^3-6x^2+11x+18\right). \qquad (2.4)
\end{aligned}
$$

- **Divided Differences:** In what follows we use $f[x_0 \dots x_n]$ to denote the coefficient of the x^n term in the (unique) polynomial that interpolates the $n+1$ points $(x_0, f(x_0)), \dots, (x_n, f(x_n))$.[2] These coefficients are given by the recursive relation

$$
\begin{aligned}
f[x_k] &= f(x_k) \\
f[x_k\, x_{k+1}] &= \frac{f[x_{k+1}] - f[x_k]}{x_{k+1} - x_k} \\
f[x_k\, x_{k+1}\, x_{k+2}] &= \frac{f[x_{k+1}\, x_{k+2}] - f[x_k\, x_{k+1}]}{x_{k+2} - x_k} \\
f[x_k\, x_{k+1}\, x_{k+2}\, x_{k+3}] &= \frac{f[x_{k+1}\, x_{k+2}\, x_{k+3}] - f[x_k\, x_{k+1}\, x_{k+2}]}{x_{k+3} - x_k}, \qquad (2.5)
\end{aligned}
$$

and so on ... The ***Newton's divided difference formula*** for an n-degree polynomial interpolating $n+1$ points (x_i, y_i), for $i = 0, \dots, n$ is then given by

$$
\begin{aligned}
{}^{\mathcal{N}}p_n(x) &= f[x_0] + \sum_{i=1}^{n} \left\{ f[x_0 \cdots x_i] \prod_{j=0}^{i-1} (x - x_j) \right\} \\
&= f[x_0] + f[x_0\, x_1](x-x_0) \\
&\qquad + f[x_0\, x_1\, x_2](x-x_0)(x-x_1) \\
&\qquad + f[x_0\, x_1\, x_2\, x_3](x-x_0)(x-x_1)(x-x_2) \\
&\qquad + \dots \\
&\qquad + f[x_0 \cdots x_n](x-x_0)\cdots(x-x_{n-1}). \qquad (2.6)
\end{aligned}
$$

The recursive nature of Eq. (2.5) allows arrangement into a convenient table form. For four points we have

$$
\begin{array}{c|cccc}
x_0 & f[x_0] & & & \\
 & & f[x_0\, x_1] & & \\
x_1 & f[x_1] & & f[x_0\, x_1\, x_2] & \\
 & & f[x_1\, x_2] & & f[x_0\, x_1\, x_2\, x_3] \\
x_2 & f[x_2] & & f[x_1\, x_2\, x_3] & \\
 & & f[x_2\, x_3] & & \\
x_3 & f[x_3] & & &
\end{array} \qquad (2.7)
$$

[2] Please note that my notation deviates from the one we are following in class, because my main programming languages are C++ and PYTHON, where the index count conventionally starts from 0.

The coefficients of the Newton polynomial Eq. (2.6) can then be read from the top edge of the triangle of this table.

Let us put all this machinery to good use for the case at hand. We can compute the coefficients using Eq. (2.5):

$$\begin{aligned}
f[x_0\, x_1] &= \frac{f[x_1]-f[x_0]}{x_1-x_0} = \frac{1-0}{2-(-1)} = \frac{1}{3}\\
f[x_1\, x_2] &= \frac{f[x_2]-f[x_1]}{x_2-x_1} = \frac{1-1}{3-2} = 0\\
f[x_2\, x_3] &= \frac{f[x_3]-f[x_2]}{x_3-x_2} = \frac{2-1}{5-3} = \frac{1}{2}\\
f[x_0\, x_1\, x_2] &= \frac{f[x_1\, x_2]-f[x_0\, x_1]}{x_2-x_0} = \frac{0-\frac{1}{3}}{3-(-1)} = -\frac{1}{12}\\
f[x_1\, x_2\, x_3] &= \frac{f[x_2\, x_3]-f[x_1\, x_2]}{x_3-x_1} = \frac{\frac{1}{2}-0}{5-2} = \frac{1}{6}\\
f[x_0\, x_1\, x_2\, x_3] &= \frac{f[x_1\, x_2\, x_3]-f[x_0\, x_1\, x_2]}{x_3-x_0} = \frac{\frac{1}{6}-\left(-\frac{1}{12}\right)}{5-(-1)} = \frac{1}{24}.
\end{aligned}$$

Or we could have easily used the table form, in which calculations are much faster:

$$\begin{array}{r|ccccc}
-1 & 0 & & & \\
 & & \frac{1}{3} & & \\
2 & 1 & & -\frac{1}{12} & \\
 & & 0 & & \frac{1}{24}\\
3 & 1 & & \frac{1}{6} & \\
 & & \frac{1}{2} & & \\
5 & 2 & & &
\end{array} \tag{2.8}$$

Reading the coefficients off the top of the triangle and plugging them back into Eq. (2.6), we get

$$\begin{aligned}
{}^{\mathcal{N}}p_3(x) &= f[x_0] + f[x_0\, x_1](x-x_0)\\
&\qquad + f[x_0\, x_1\, x_2](x-x_0)(x-x_1)\\
&\qquad + f[x_0\, x_1\, x_2\, x_3](x-x_0)(x-x_1)(x-x_2)\\
&= 0 + \frac{1}{3}(x-(-1))\\
&\quad + \left(-\frac{1}{12}\right)(x-(-1))(x-2)\\
&\quad + \frac{1}{24}(x-(-1))(x-2)(x-3)\\
&= \frac{1}{3}(x+1) - \frac{1}{12}(x+1)(x-2) + \frac{1}{24}(x+1)(x-2)(x-3)
\end{aligned}$$

$$= \frac{1}{24}\left(x^3 - 6x^2 + 11x + 18\right). \tag{2.9}$$

We see that the polynomials ${}^{\mathcal{L}}p_3$ and ${}^{\mathcal{N}}p_3$ are identical, which must be true according to the ***Main Theorem of Polynomial Interpolation***, stated above. □

Example 5. *Let $P(x)$ be the degree 9 polynomial that takes the value 112 at $x = 1$, takes the value 2 at $x = 10$ and equals 0 for $x = 2, \ldots, 9$. Calculate $P(0)$.*

Solution. Let us find the polynomial using Lagrange interpolation; using Eq. (2.3), we have

$${}^{\mathcal{L}}p_9(x) = \sum_{i=0}^{9} y_i \ell_i(x) = y_0\ell_0(x) + y_9\ell_9(x). \tag{2.10}$$

Note that we are only left with two terms in the summation, since $y_i = 0$ for $i = 1, \ldots, 8$, by construction. Thus we are left with $y_0 = 112$ and $y_9 = 2$, and

$$\ell_0(x) = \prod_{j\neq 0} \frac{x - x_j}{x_0 - x_j} = \frac{x-2}{1-2}\cdot\frac{x-3}{1-3}\cdots\frac{x-10}{1-10} = -\frac{1}{9!}\prod_{k=2}^{10}(x-k)$$

$$\ell_9(x) = \prod_{j\neq 9} \frac{x - x_j}{x_9 - x_j} = \frac{x-1}{10-1}\cdot\frac{x-2}{10-2}\cdots\frac{x-9}{10-9} = \frac{1}{9!}\prod_{k=1}^{9}(x-k).$$

Plugging this back into Eq. (2.10), we get

$${}^{\mathcal{L}}p_9(x) = -112\frac{1}{9!}\prod_{k=2}^{10}(x-k) + 2\frac{1}{9!}\prod_{k=1}^{9}(x-k). \tag{2.11}$$

Evaluating this polynomial at $x = 0$ yields

$$\begin{aligned}
{}^{\mathcal{L}}p_9(0) &= -112\frac{1}{9!}\prod_{k=2}^{10}(-k) + 2\frac{1}{9!}\prod_{k=1}^{9}(-k) \\
&= -112\left(-\frac{10!}{9!}\right) + 2\left(-\frac{9!}{9!}\right) \\
&= 1120 - 2 \\
&= 1118.
\end{aligned}$$

□

Example 6. *Given the data points $(1, 0)$, $(2, \log 2)$, $(4, \log 4)$, find the degree-2 interpolating polynomial and use it to approximate $\log 3$. Give an error bound and compare the actual error to your error bound.*

Solution. We use divided differences in its simple and elegant form:

$$\begin{array}{c|ccc}
1 & 0 & & \\
 & & \log 2 & \\
2 & \log 2 & & -\frac{\log 2}{6} \\
 & & \frac{\log 2}{2} & \\
4 & \log 4 & &
\end{array} \tag{2.12}$$

Reading the coefficients off the top of the triangle and plugging them back into Eq. (2.6), we get

$$\begin{aligned}
{}^{\mathcal{N}}p_2(x) &= f[x_0] + f[x_0\, x_1](x - x_0) + f[x_0\, x_1\, x_2](x - x_0)(x - x_1) \\
&= 0 + \log 2(x-1) - \frac{\log 2}{6}(x-1)(x-2) \\
&= -\frac{\log 2}{6}\left(x^2 - 9x + 8\right). \qquad (2.13)
\end{aligned}$$

In order to approximate $\log 3$, we now evaluate this polynomial at $x = 3$:

$${}^{\mathcal{N}}p_2(3) = -\frac{\log 2}{6}\left(3^2 - 9(3) + 8\right) = \frac{5\log 2}{3}. \tag{2.14}$$

To find the error bound we use Theorem 9; applying Eq. (2.1) to $f(x) = \log x$, we have

$$\begin{aligned}
\log x - {}^{\mathcal{N}}p_2(x) &= \frac{\log^{(3)}(\xi)}{3!}\prod_{i=0}^{2}(x - x_i) \\
&= \frac{2}{3\cdot 2\xi^3}(x-1)(x-2)(x-4) \\
&= (x-1)(x-2)(x-4)\frac{1}{3\xi^3},
\end{aligned}$$

where ξ lies between the smallest and largest of the numbers $\{x, 1, 4\}$. Since we are interested in the error at $x = 3$, we have $\xi \in (1, 4)$. Thus,

$$|\log 3 - {}^{\mathcal{N}}p_2(3)| = |-2\frac{1}{3\xi^3}|.$$

The largest possible error we can get is when ξ is smallest, i.e., when $\xi \to 1$. Hence,

$$|\log 3 - {}^{\mathcal{N}}p_2(3)| \leq \frac{2}{3} \tag{2.15}$$

is an upper bound for the error. Now, the actual error (to 10 decimals precision) is

$$|\log 3 - {}^{\mathcal{N}}p_2(3)| = |\log 3 - \frac{5\log 2}{3}| \approx |-0.05663301227| = 0.05663301227 \ll \frac{2}{3}. \tag{2.16}$$

This shows that our upper bound (2.15) is way too generous. Sauer does warn us that in order to have smaller errors we're better off evaluating far from one of the endpoints. For instance, if we take $\xi = 2$, then the error upper bound we get would be

$$\frac{2}{3\cdot 2^3} \approx 0.083,$$

which is much closer to the actual error given by (2.16). □

2.2 Cubic Splines

The following two examples fully showcase how to use cubic splines:

Example 7. *Take four data points (x_i, y_i), with $i = 0, \dots, 3$. Give detailed procedure for computing cubic splines with natural, curvature-adjusted, clamped, parabolically-terminated and not-a-knot endpoint conditions. Check your results for the data points $(0,3)$, $(1,5)$, $(2,4)$, and $(3,1)$.*

Solution. A ***cubic spline*** $S(x)$ through $n+1$ data points $(x_0, y_0), \dots, (x_n, y_n)$ is a set of cubic polynomials S_i, with $0 \le i \le n-1$, given by [3]

$$\begin{aligned} S_0(x) &= y_0 + b_0(x-x_0) + c_0(x-x_0)^2 + d_0(x-x_0)^3 \qquad \text{for } x \in [x_0, x_1] \\ &\vdots \\ S_{n-1}(x) &= y_{n-1} + b_{n-1}(x-x_{n-1}) + c_{n-1}(x-x_{n-1})^2 + d_{n-1}(x-x_{n-1})^3 \qquad \text{for } x \in [x_{n-1}, x_n], \end{aligned} \tag{2.17}$$

where $\{b_i, c_i, d_i\}$ are coefficients that need to be determined. These polynomials need to satisfy the following four properties:

- **Property I:** $S_i(x_i) = y_i$ for $i = 0, \dots, n-1$.
- **Property II:** $S_i(x_{i+1}) = y_{i+1}$ for $i = 0, \dots, n-1$.
- **Property III:** $S'_{i-1}(x_i) = S'_i(x_i)$ for $i = 1, \dots, n-1$.
- **Property IV:** $S''_{i-1}(x_i) = S''_i(x_i)$ for $i = 1, \dots, n-1$.

Hence, constructing a spline from a set of data points means finding the coefficients $\{b_i, c_i, d_i\}$ that satisfy these four properties stated above.

Property I follows easily from just plugging $x = x_i$ into Eq. (2.17). Property II, on the other hand, yields n independent equations, one for each i, that need to be satisfied by the coefficients:

$$\begin{aligned} y_1 &= S_0(x_1) = y_0 + b_0(x_1-x_0) + c_0(x_1-x_0)^2 + d_0(x_1-x_0)^3 \\ &\vdots \\ y_n &= S_{n-1}(x_n) = y_{n-1} + b_{n-1}(x_n-x_{n-1}) + c_{n-1}(x_n-x_{n-1})^2 + d_{n-1}(x_n-x_{n-1})^3. \end{aligned} \tag{2.18}$$

Similarly, Property III yields $n-1$ additional equations:

$$0 = S'_0(x_1) - S'_1(x_1) = b_0 - b_1 + 2c_0(x_1-x_0) + 3d_0(x_1-x_0)^2$$

[3] I decided to explain the procedure for the more general case of $n+1$ data points; we get the desired result for this particular exercise by setting $n = 3$.

$$\begin{aligned}&\vdots \\ 0 &= S'_{n-2}(x_{n-1}) - S'_{n-1}(x_{n-1}) = b_{n-2} - b_{n-1} + 2c_{n-2}(x_{n-1} - x_{n-2}) + 3d_{n-2}(x_{n-1} - x_{n-2})^2.\end{aligned} \tag{2.19}$$

And a further $n-1$ equations from Property IV:

$$\begin{aligned}0 &= S''_0(x_1) - S''_1(x_1) = 2(c_0 - c_1) + 6d_0(x_1 - x_0) \\ &\vdots \\ 0 &= S''_{n-2}(x_{n-1}) - S''_{n-1}(x_{n-1}) = 2(c_{n-2} - c_{n-1}) + 6d_{n-2}(x_{n-1} - x_{n-2}).\end{aligned} \tag{2.20}$$

Hence, in total, we have

$$\overbrace{n}^{\text{From Prop. II}} + \overbrace{n-1}^{\text{From Prop. III}} + \overbrace{n-1}^{\text{From Prop. IV}} = 3n - 2.$$

Moreover, there are 3 coefficients $\{b_i, c_i, d_i\}$ on each polynomial S_i, and there are n of the latter; thus we have a total of $3n$ coefficients. In other words, we have an underdetermined system of equations, since we have

$$3n \text{ unknowns} \quad \& \quad 3n-2 \text{ equations.}$$

Hence we have infinitely many solutions; i.e., infinitely many cubic splines passing through the arbitrary set of $n+1$ data points $(x_0, y_0), \ldots, (x_n, y_n)$. If, however, we were to impose two further constraints (i.e., equations) we would get a fully determined system (same number of unknowns and equations). This is precisely how we classify splines, depending on which two constraints we add to the system (these extra couple of constraints are usually applied to the left and right ends of the spline, so they are called ***endpoint conditions***). The classification goes as follows: A spline is said to be

- ***natural*** if $S''_0(x_0) = 0$ and $S''_{n-1}(x_n) = 0$;
- ***curvature-adjusted*** if $S''_0(x_0) = \kappa_0$ and $S''_{n-1}(x_n) = \kappa_n$, where κ_0 and κ_n are user-defined, nonzero values;
- ***clamped*** if $S'_0(x_0) = \nu_0$ and $S'_{n-1}(x_n) = \nu_n$, where ν_0 and ν_n are user-defined, nonzero values;
- ***parabolically-terminated*** if $\deg S_0 \leq 2$ and $\deg S_{n-1} \leq 2$. That is, the first and last polynomials of the spline –S_0 and S_{n-1}, respectively– are forced to have degree at most 2. This can be enforced by setting $d_0 = d_{n-1} = 0$.
- ***not-a-knot*** if $S'''_0(x_1) = S'''_1(x_1)$ and $S'''_{n-2}(x_{n-1}) = S'''_{n-1}(x_{n-1})$. Equivalently, we may set $d_0 = d_1$ and $d_{n-2} = d_{n-1}$, Since S_0 and S_1 are polynomials of degree ≤ 3, requiring their third derivatives to agree at x_1, while their zeroth, first, and second derivatives already agree there, causes S_0 and S_1 to be identical cubic polynomials. Thus, x_1 is not needed as a base point: the spline is given by the same formula $S_0 = S_1$ on the entire interval $[x_0, x_2]$. The same reasoning shows that $S_{n-2} = S_{n-1}$, so both x_1 and x_{n-1} are "no longer knots."

Let us now put together all of this machinery and write down the general solution for each type of spline:

★ **`Natural Spline:`** Now that we have $3n$ equations to solve $3n$ unkowns, we could use some linear algebra solver in C++ (or whatever language of preference). However, it turns out that we can drastically simplify the system by decoupling the equations first. Let us introduce the notation

$$\begin{aligned} {}^x\Delta_i &= x_{i+1} - x_i \\ {}^y\Delta_i &= y_{i+1} - y_i. \end{aligned}$$

Now consider (2.20), for any $i = 1, \dots, n-1$:

$$0 = 2(c_i - c_{i+1}) + 6d_i {}^x\Delta_i.$$

Isolating d_i, we get

$$d_i = \frac{c_{i+1} - c_i}{3\,{}^x\Delta_i}. \tag{2.21}$$

Substituting this expression into (2.18) and solving for b_i, we get

$$b_i = \frac{{}^y\Delta_i}{{}^x\Delta_i} - \frac{{}^x\Delta_i}{3}\left(c_{i+1} + 2c_i\right). \tag{2.22}$$

Plugging both of these expressions, (2.21)–(2.22), into (2.19), we get $n-1$ equations in $c_0, \dots, c_n$:

$$\begin{aligned} {}^x\Delta_0 c_0 + 2\left({}^x\Delta_0 + {}^x\Delta_1\right)c_1 + {}^x\Delta_1 c_2 &= 3\left(\frac{{}^y\Delta_1}{{}^x\Delta_1} - \frac{{}^y\Delta_0}{{}^x\Delta_0}\right) \\ &\vdots \\ {}^x\Delta_{n-2} c_{n-2} + 2\left({}^x\Delta_{n-2} + {}^x\Delta_{n-1}\right)c_{n-1} + {}^x\Delta_{n-1} c_n &= 3\left(\frac{{}^y\Delta_{n-1}}{{}^x\Delta_{n-1}} - \frac{{}^y\Delta_{n-2}}{{}^x\Delta_{n-2}}\right). \end{aligned} \tag{2.23}$$

Adding the two additional constraints that pertain to natural splines, namely

$$S_0''(x_0) = 0 \quad \Longrightarrow \quad 2c_0 = 0; \tag{2.24a}$$
$$S_{n-1}''(x_n) = 0 \quad \Longrightarrow \quad 2c_n = 0, \tag{2.24b}$$

we end up with $n+1$ equations for the $n+1$ unknowns $c_0, \dots, c_n$. In matrix form, this looks like

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & & \\ {}^x\Delta_0 & 2({}^x\Delta_0 + {}^x\Delta_1) & {}^x\Delta_1 & \ddots & & \\ 0 & {}^x\Delta_1 & 2({}^x\Delta_1 + {}^x\Delta_2) & {}^x\Delta_2 & & \\ \vdots & \ddots & \ddots & \ddots & & \\ & & & {}^x\Delta_{n-2} & 2({}^x\Delta_{n-2} + {}^x\Delta_{n-1}) & {}^x\Delta_{n-1} \\ & & & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ \\ \vdots \\ \\ c_n \end{pmatrix} = \begin{pmatrix} 0 \\ 3\left(\frac{{}^y\Delta_1}{{}^x\Delta_1} - \frac{{}^y\Delta_0}{{}^x\Delta_0}\right) \\ \vdots \\ 3\left(\frac{{}^y\Delta_{n-1}}{{}^x\Delta_{n-1}} - \frac{{}^y\Delta_{n-2}}{{}^x\Delta_{n-2}}\right) \\ 0 \end{pmatrix} \tag{2.25}$$

Once we obtain the c_i from (2.25), the d_i and b_i follow from (2.21) and (2.22), respectively.

Now, FINALLY, we construct a natural spline for the provided data points $(0,3)$, $(1,5)$, $(2,4)$, and $(3,1)$. For four data points ($n = 3$), the matrix equation (2.25) reduces to

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ {}^x\Delta_0 & 2({}^x\Delta_0 + {}^x\Delta_1) & {}^x\Delta_1 & 0 \\ 0 & {}^x\Delta_1 & 2({}^x\Delta_1 + {}^x\Delta_2) & {}^x\Delta_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 3\left(\frac{{}^y\Delta_1}{{}^x\Delta_1} - \frac{{}^y\Delta_0}{{}^x\Delta_0}\right) \\ 3\left(\frac{{}^y\Delta_2}{{}^x\Delta_2} - \frac{{}^y\Delta_1}{{}^x\Delta_1}\right) \\ 0 \end{pmatrix}. \tag{2.26}$$

In the case at hand, we have

$$\begin{aligned} {}^x\Delta_0 &= x_1 - x_0 = 1 - 0 = 1 \\ {}^x\Delta_1 &= x_2 - x_1 = 2 - 1 = 1 \\ {}^x\Delta_2 &= x_3 - x_2 = 3 - 2 = 1 \\ {}^y\Delta_0 &= y_1 - y_0 = 5 - 3 = 2 \\ {}^y\Delta_1 &= y_2 - y_1 = 4 - 5 = -1 \\ {}^y\Delta_2 &= y_3 - y_2 = 1 - 4 = -3. \end{aligned}$$

Plugging this into (2.26), we have

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ -9 \\ -6 \\ 0 \end{pmatrix}, \tag{2.27}$$

which has solution

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ -2 \\ -1 \\ 0 \end{pmatrix}. \tag{2.28}$$

As for the remaining coefficients, we use Eqs. (2.21)–(2.22) and plug in the values just acquired for the c_i:

$$\begin{aligned} d_0 &= \frac{c_1 - c_0}{3\,{}^x\Delta_0} = \frac{-2-0}{3\cdot 1} = -\frac{2}{3} \\ d_1 &= \frac{c_2 - c_1}{3\,{}^x\Delta_1} = \frac{-1-(-2)}{3\cdot 1} = \frac{1}{3} \\ d_2 &= \frac{c_3 - c_2}{3\,{}^x\Delta_2} = \frac{0-(-1)}{3\cdot 1} = \frac{1}{3} \\ b_0 &= \frac{{}^y\Delta_0}{{}^x\Delta_0} - \frac{{}^x\Delta_0}{3}(c_1 + 2c_0) = \frac{2}{1} - \frac{1}{3}(-2 + 2\cdot 0) = \frac{8}{3} \\ b_1 &= \frac{{}^y\Delta_1}{{}^x\Delta_1} - \frac{{}^x\Delta_1}{3}(c_2 + 2c_1) = \frac{-1}{1} - \frac{1}{3}(-1 + 2\cdot(-2)) = \frac{2}{3} \end{aligned}$$

$$b_2 = \frac{{}^y\Delta_2}{{}^x\Delta_2} - \frac{{}^x\Delta_2}{3}(c_3 + 2c_2) = \frac{-3}{1} - \frac{1}{3}(0 + 2\cdot(-1)) = -\frac{7}{3}.$$

Hence, our natural spline is given by

$$\boxed{S(x) = \begin{cases} 3 + \frac{8}{3}x - \frac{2}{3}x^3 & \text{if } x \in [0,1]; \\ 5 + \frac{2}{3}(x-1) - 2(x-1)^2 + \frac{1}{3}(x-1)^3 & \text{if } x \in [1,2]; \\ 4 - \frac{7}{3}(x-2) - (x-2)^2 + \frac{1}{3}(x-2)^3 & \text{if } x \in [2,3]. \end{cases}} \tag{2.29}$$

(This is the only spline I will fully derive; for the remaining cases I will simply mention the difference in procedure and will skip all the messy algebra. It's obvious that if you know how to derive one, you know how to derive the rest…it's just tedious algebra.)

★ **`Curvature-Adjusted Spline`:** The only change here is that now we have nonzero values at c_0 and c_n, namely

$$S_0''(x_0) = \kappa_0 \quad\Longrightarrow\quad 2c_0 = \kappa_0; \tag{2.30a}$$
$$S_{n-1}''(x_n) = \kappa_n \quad\Longrightarrow\quad 2c_n = \kappa_n. \tag{2.30b}$$

In turn, the matrix equation (2.26) changes to

$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ {}^x\Delta_0 & 2({}^x\Delta_0 + {}^x\Delta_1) & {}^x\Delta_1 & 0 \\ 0 & {}^x\Delta_1 & 2({}^x\Delta_1 + {}^x\Delta_2) & {}^x\Delta_2 \\ 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} \kappa_0 \\ 3\left(\frac{{}^y\Delta_1}{{}^x\Delta_1} - \frac{{}^y\Delta_0}{{}^x\Delta_0}\right) \\ 3\left(\frac{{}^y\Delta_2}{{}^x\Delta_2} - \frac{{}^y\Delta_1}{{}^x\Delta_1}\right) \\ \kappa_3 \end{pmatrix}. \tag{2.31}$$

Hence, our system is

$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} \kappa_0 \\ -9 \\ -6 \\ \kappa_3 \end{pmatrix}, \tag{2.32}$$

which has solution

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\kappa_0 \\ \frac{1}{30}\kappa_3 - \frac{2}{15}\kappa_0 - 2 \\ -\frac{2}{15}\kappa_3 + \frac{1}{30}\kappa_0 - 1 \\ \frac{1}{2}\kappa_3 \end{pmatrix}. \tag{2.33}$$

Determining the b_i and d_i then follow as we did above.

★ **`Clamped Spline`:** This time it is the first derivatives that have user-defined values. Using (2.21)–(2.22), we can write the two extra constraints as

$$S_0'(x_0) = \nu_0 \quad\Longrightarrow\quad 2\,{}^x\Delta_0\, c_0 + {}^x\Delta_0\, c_1 = 3\left(\frac{{}^y\Delta_0}{{}^x\Delta_0} - \nu_0\right); \tag{2.34a}$$

$$S'_{n-1}(x_n) = v_n \quad \Longrightarrow \quad {}^x\Delta_{n-1}\, c_{n-1} + 2\,{}^x\Delta_{n-1}\, c_n = 3\left(v_n - \frac{{}^y\Delta_{n-1}}{{}^x\Delta_{n-1}}\right). \tag{2.34b}$$

In turn, the matrix equation (2.26) changes to

$$\begin{pmatrix} 2\,{}^x\Delta_0 & {}^x\Delta_0 & 0 & 0 \\ {}^x\Delta_0 & 2({}^x\Delta_0 + {}^x\Delta_1) & {}^x\Delta_1 & 0 \\ 0 & {}^x\Delta_1 & 2({}^x\Delta_1 + {}^x\Delta_2) & {}^x\Delta_2 \\ 0 & 0 & {}^x\Delta_2 & 2\,{}^x\Delta_2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 3\left(\frac{{}^y\Delta_0}{{}^x\Delta_0} - v_0\right) \\ 3\left(\frac{{}^y\Delta_1}{{}^x\Delta_1} - \frac{{}^y\Delta_0}{{}^x\Delta_0}\right) \\ 3\left(\frac{{}^y\Delta_2}{{}^x\Delta_2} - \frac{{}^y\Delta_1}{{}^x\Delta_1}\right) \\ 3\left(v_3 - \frac{{}^y\Delta_2}{{}^x\Delta_2}\right) \end{pmatrix}. \tag{2.35}$$

Solving for $\{c_i, b_i, d_i\}$ follows as above. (We may pick some arbitrary values for the user-defined variables to simplify the algebra).

★ **`Parabolically-Terminated Spline:`** Using (2.21), we can write the two extra constraints as

$$d_0 = 0 \quad \Longrightarrow \quad c_0 = c_1 \tag{2.36a}$$

$$d_{n-1} = 0 \quad \Longrightarrow \quad c_{n-1} = c_n. \tag{2.36b}$$

This turns the matrix equation (2.25) into

$$\begin{pmatrix} 1 & -1 & 0 & 0 \\ {}^x\Delta_0 & 2({}^x\Delta_0 + {}^x\Delta_1) & {}^x\Delta_1 & 0 \\ 0 & {}^x\Delta_1 & 2({}^x\Delta_1 + {}^x\Delta_2) & {}^x\Delta_2 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 3\left(\frac{{}^y\Delta_1}{{}^x\Delta_1} - \frac{{}^y\Delta_0}{{}^x\Delta_0}\right) \\ 3\left(\frac{{}^y\Delta_2}{{}^x\Delta_2} - \frac{{}^y\Delta_1}{{}^x\Delta_1}\right) \\ 0 \end{pmatrix}. \tag{2.37}$$

Solving for $\{c_i, b_i, d_i\}$ follows as above.

★ **`Not-a-Knot Spline:`** The two extra constraints are now

$$S_0'''(x_1) = S_1'''(x_1) \implies d_0 = d_1 \implies {}^x\Delta_1\, c_0 - ({}^x\Delta_0 + {}^x\Delta_1)\, c_1 + {}^x\Delta_0\, c_2 = 0 \tag{2.38a}$$

$$S_{n-2}'''(x_{n-1}) = S_{n-1}'''(x_{n-1}) \implies d_{n-2} = d_{n-1} \implies {}^x\Delta_{n-1}\, c_{n-2} - ({}^x\Delta_{n-2} + {}^x\Delta_{n-1})\, c_{n-1} + {}^x\Delta_{n-2}\, c_n = 0. \tag{2.38b}$$

(The right-most implications come from using Eq. (2.21).) This turns the matrix equation (2.25) into

$$\begin{pmatrix} {}^x\Delta_1 & -({}^x\Delta_0 + {}^x\Delta_1) & {}^x\Delta_0 & 0 \\ {}^x\Delta_0 & 2({}^x\Delta_0 + {}^x\Delta_1) & {}^x\Delta_1 & 0 \\ 0 & {}^x\Delta_1 & 2({}^x\Delta_1 + {}^x\Delta_2) & {}^x\Delta_2 \\ 0 & {}^x\Delta_2 & -({}^x\Delta_1 + {}^x\Delta_2) & {}^x\Delta_1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 3\left(\frac{{}^y\Delta_1}{{}^x\Delta_1} - \frac{{}^y\Delta_0}{{}^x\Delta_0}\right) \\ 3\left(\frac{{}^y\Delta_2}{{}^x\Delta_2} - \frac{{}^y\Delta_1}{{}^x\Delta_1}\right) \\ 0 \end{pmatrix}. \tag{2.39}$$

Solving for $\{c_i, b_i, d_i\}$ follows as above. □

Example 8. *Find c in the following cubic splines. Which of the three end conditions (natural, parabolically-terminated, or not-a-knot), if any, are satisfied?*

$$S(x) = \begin{cases} 4 - \frac{11}{4}x + \frac{3}{4}x^3 & \text{if } x \in [0,1]; \\ 2 - \frac{1}{2}(x-1) + c(x-1)^2 - \frac{3}{4}(x-1)^3 & \text{if } x \in [1,2]. \end{cases} \tag{2.40a}$$

$$\widehat{S}(x) = \begin{cases} 3 - 9x + 4x^2 & \text{if } x \in [0,1]; \\ -2 - (x-1) + c(x-1)^2 & \text{if } x \in [1,2]. \end{cases} \tag{2.40b}$$

$$\widetilde{S}(x) = \begin{cases} -2 - \frac{3}{2}x + \frac{7}{2}x^2 - x^3 & \text{if } x \in [0,1]; \\ -1 + c(x-1) + \frac{1}{2}(x-1)^2 - (x-1)^3 & \text{if } x \in [1,2]; \\ 1 + \frac{1}{2}(x-2) - \frac{5}{2}(x-2)^2 - (x-2)^3 & \text{if } x \in [2,3]. \end{cases} \tag{2.40c}$$

Solution. We start with Eq. (2.40a). Using Eq. (2.21) and notation from the previous example, we have

$$\begin{aligned} d_0 &= \frac{c_1 - c_0}{3\,{}^x\Delta_0} \\ \frac{3}{4} &= \frac{c-0}{3 \cdot 1} \\ c &= \frac{9}{4}. \end{aligned}$$

Now let's test for the endpoint conditions:

$$\begin{aligned} S_0(0)'' &= \frac{9}{2} \cdot 0 = 0; \\ S_1(2)'' &= 2 \cdot \frac{9}{4} - \frac{9}{2} \cdot (2-1) = 0. \end{aligned}$$

Hence, we conclude that the natural spline condition is satisfied. $\surd$

Moving on to Eq. (2.40b),

$$\begin{aligned} d_0 &= \frac{c_1 - c_0}{3\,{}^x\Delta_0} \\ 0 &= \frac{c-4}{3 \cdot 1} \\ c &= 4. \end{aligned}$$

Since the degree of both $\widehat{S}_0$ and $\widehat{S}_1$ is at most 2, we conclude that the spline is parabolically-terminated. However, it is curious to note that, if we evaluate the second derivatives at the endpoints, we get the curvature-adjusted condition:

$$\begin{aligned} \widehat{S}_0(0)'' &= 8; \\ \widehat{S}_1(2)'' &= 2 \cdot 4 = 8. \end{aligned}$$

(Question for Dr. Khan: What's going on? Is it possible for a spline to satisfy more than one endpoint condition?) $\surd$

Lastly, we tackle Eq. (2.40c), this time using Eq. (2.22)

$$\begin{aligned} b_1 &= \frac{{}^y\Delta_1}{{}^x\Delta_1} - \frac{{}^x\Delta_1}{3}(c_2 + 2c_1) \\ c &= \frac{3-2}{2-1} - \frac{2-1}{3}((-1) + 2(-1)) \\ c &= 1. \end{aligned}$$

From just eyeballing this spline, it is not hard to guess that it'll be a not-a-knot spline; let us test that:

$$\begin{aligned} \widetilde{S}_0(1)''' &= -6 = \widetilde{S}_1(1)'''; \\ \widetilde{S}_1(2)''' &= -6 = \widetilde{S}_2(2)'''. \end{aligned}$$

Hence, the not-a-knot condition is indeed satisfied. $\surd$ ☐

2.3 Chebyshev Polynomials

1. The choice of base point spacing can have significant effect on the interpolation error.
2. Chebyshev interpolation refers to a particular optimal way of spacing the points.
3. The primary motivation for Chebyshev interpolation is to improve control of the maximum value of the interpolation error

$$\frac{(x-x_1)(x-x_2)\cdots(x-x_n)}{n!} f^{(n)}(c)$$

on the interpolation interval.

Definition 7. *The n^{th} **Chebyshev polynomial of the first kind** T_n is given by*

$$T_n(x) = \cos(n \arccos x). \tag{2.41}$$

*The n^{th} **Chebyshev polynomial of the second kind** U_n is given by*

$$U_n(x) = \frac{1}{n+1} T'_{n+1}(x). \tag{2.42}$$

Let $x = \cos\varphi$, so that $T_n(x) = \cos(n\varphi)$. Then, for $n \geq 1$,

$$\begin{aligned} T_{n+1}(x) &= \cos[(n+1)\varphi] = \cos(n\varphi + \varphi) = \cos(n\varphi)\cos\varphi - \sin(n\varphi)\sin\varphi \\ T_{n-1}(x) &= \cos[(n-1)\varphi] = \cos(n\varphi - \varphi) = \cos(n\varphi)\cos\varphi - \sin(n\varphi)\underbrace{\sin(-\varphi)}_{=-\sin(\varphi)}. \end{aligned}$$

Adding these two expressions,

$$T_{n+1}(x) + T_{n-1}(x) = 2\cos(n\varphi)\cos\varphi = 2x\,T_n(x).$$

So we end up with the recursive relation

$$\boxed{T_{n+1}(x) = 2x\,T_n(x) - T_{n-1}(x)} \tag{2.43}$$

for $n \geq 1$. Using this relation, along with the definition (2.41), we find

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x, \end{aligned}$$

and so on ... Thus, the Chebyshev polynomials –as odd as they may seem at first– are indeed polynomials in the traditional sense.

There is also a similar recursive relation for the Chebyshev polynomials of the second kind, namely

$$\boxed{U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x)} \tag{2.44}$$

for $n \geq 1$. We will show this on an example below.

A Chebyshev polynomial of either kind with degree n has n different simple roots, called ***Chebyshev roots***, in the interval $[-1, 1]$:

Definition 8. *Let $n \geq 1$. Then the roots of the T_n are given by*

$$x_k = \cos\left(\frac{[2k+1]\pi}{2n}\right) \qquad \text{for } k = 0, \ldots, n. \tag{2.45}$$

*These are called the **Chebyshev points of the first kind** (also known as the **Chebyshev nodes**,* [4] *or the **Gauss points**).*

Meanwhile, the roots of T_n' are given by

$$x_k = \cos\left(\frac{k\pi}{n}\right) \qquad \text{for } k = 1, \ldots, n-1. \tag{2.46}$$

[4]The more apt name would be, indeed, *Chebyshev nodes*, because they are used as nodes in polynomial interpolation.

*These are called the **Gauss-Lobato points**.*

Lastly, the roots of the U_n are given by

$$x_k = \cos\left(\frac{k\pi}{n+1}\right) \qquad \text{for } k = 1, \dots, n. \tag{2.47}$$

*These are called the **Chebyshev points of the second kind**.*

Remark 1. *Note, however, that since the degree-n polynomial $T_n(x)$ will only have n roots, and we would like to have $n+1$ Chevyshev nodes (roots) to construct an n-degree polynomial interpolate, instead of using Eq. (2.45), we use the roots of T_{n+1} instead:*

$$\boxed{x_k = \cos\left(\frac{[2k+1]\pi}{2(n+1)}\right) \qquad \text{for } k = 0, \dots, n} \tag{2.48}$$

For instance, let's say we want a degree-4 Chebyshev interpolation; then we use the five roots of $T_5(x)$:

$$\begin{aligned}
x_0 &= \cos\left(\frac{(2\times 0+1)\pi}{2(4+1)}\right) = \cos\frac{\pi}{10}\\
x_1 &= \cos\left(\frac{(2\times 1+1)\pi}{2(4+1)}\right) = \cos\frac{3\pi}{10}\\
x_2 &= \cos\left(\frac{(2\times 2+1)\pi}{2(4+1)}\right) = \cos\frac{5\pi}{10}\\
x_3 &= \cos\left(\frac{(2\times 3+1)\pi}{2(4+1)}\right) = \cos\frac{7\pi}{10}\\
x_4 &= \cos\left(\frac{(2\times 4+1)\pi}{2(4+1)}\right) = \cos\frac{9\pi}{10}.
\end{aligned}$$

Theorem 10 (Chebyshev Theorem). *The choice of real numbers $-1 \le x_0, \dots, x_n \le 1$ that makes the value of*

$$\max_{-1\le x\le 1} |(x-x_0)\cdots(x-x_n)| \tag{2.49}$$

as small as possible is the Chebyshev nodes, given by (2.48). The minimum value is $1/2^n$, and it is achieved by

$$(x-x_0)\cdots(x-x_n) = \frac{1}{2^n}T_{n+1}(x). \tag{2.50}$$

We conclude from the theorem that interpolation error (c.f., 2.1)

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i) \tag{2.51}$$

can be minimized if the $n+1$ interpolation base points in $[-1,1]$ are chosen to be the roots of the degree $n+1$ Chebyshev interpolating polynomial $T_{n+1}(x)$.

Example 9. *Find a worst-case error bound for the difference on $[-1,1]$ between $f(x) = e^x$ and the degree-4 Chebyshev interpolating polynomial.*

Solution. From the error formula,

$$f(x) - p_4(x) = \frac{(x - x_0)\cdots(x - x_4)}{5!} f^{(5)}(\xi),$$

where $\xi \in [-1,1]$ and the x_i are the roots of T_5, which were shown above. Then, according to Eq. (2.50) on Theorem 10, for $x \in [-1,1]$ we have

$$(x - x_0)\cdots(x - x_4) = \frac{1}{2^4} T_5(x).$$

But since $T_5(x) \leq 1$ (recall the definition of the Chebyshev polynomials as cosines of some angle), we end up with the error bound

$$(x - x_0)\cdots(x - x_4) \leq \frac{1}{2^4}.$$

Moreover, $|f^{(5)}| \leq e$ on $[-1,1]$, so the interpolation error is

$$|e^x - p_4(x)| \leq \frac{e}{2^4 5!} \approx 0.00142 \qquad \forall x[-1,1].$$

□

Example 10. *Recall Eq. (2.42) for the Chebyshev polynomial of the second kind, namely*

$$U_n(x) = \frac{1}{n+1} T'_{n+1}(x) \qquad \textit{for } n \geq 0,$$

where $T_{n+1}(x)$ is the Chebyshev polynomial of the first kind.

a) *Using the form $T_n(x) = \cos(n\theta)$, $x = \cos\theta$, $x \in [-1,1]$, derive a similar expression for $U_n(x)$.*

b) *Prove the relation (2.44) for Chebyshev polynomials of the second kind; that is, show*

$$\begin{aligned} U_0(x) &= 1, \\ U_1(x) &= 2x, \\ U_{n+1}(x) &= 2xU_n(x) - U_{n-1}(x). \end{aligned}$$

c) *Show that the Chebyshev polynomials of the second kind are orthogonal with respect to the inner product*

$$\langle f, g \rangle = \int_{-1}^{1} f(x)g(x)\sqrt{1 - x^2}\,\mathrm{d}x. \qquad (2.52)$$

Solution to a). Using $T_n(x) = \cos(n\theta)$ and $x = \cos\theta$, we have

$$\begin{aligned} T_{n+1}(x) &= \cos[(n+1)\theta] = \cos(n\theta + \theta) \\ &= \cos(n\theta)\cos\theta - \sin(n\theta)\overbrace{\sin\theta}^{=\sqrt{1-\cos^2\theta}} \end{aligned}$$

$$= \cos(n \arccos x) x - \sin(n \arccos x)\sqrt{1-x^2}.$$

Now, taking the derivative with respect to x of this expression, we end up with

$$T'_{n+1}(x) = \frac{(1+n)\left[\sqrt{1-x^2}\cos(n\arccos x) + x\sin(n\arccos x)\right]}{\sqrt{1-x^2}}.$$

Hence,

$$\begin{aligned} U_n(x) &= \frac{1}{n+1} T'_{n+1}(x) \\ &= \cos(n\arccos x) + \frac{x\sin(n\arccos x)}{\sqrt{1-x^2}} \qquad (2.53) \\ &= \cos(n\theta) + \frac{\cos\theta\,\sin(n\theta)}{\sin\theta}. \end{aligned}$$

□

Solution to b). Using (2.53),

$$\begin{aligned} U_0(x) &= \cos(0\arccos x) + \frac{x\sin(0\arccos x)}{\sqrt{1-x^2}} \\ &= \cos 0 + \frac{x\sin 0}{\sqrt{1-x^2}} \\ &= 1. \qquad \surd \end{aligned}$$

$$\begin{aligned} U_1(x) &= \cos(\arccos x) + \frac{x\sin(\arccos x)}{\sqrt{1-x^2}} \\ &= x + \frac{x \overbrace{\sin\theta}^{=\sqrt{1-\cos^2\theta}}}{\sqrt{1-x^2}} \\ &= x + \frac{x\sqrt{1-x^2}}{\sqrt{1-x^2}} \\ &= x + x \\ &= 2x. \qquad \surd \end{aligned}$$

More generally, for any $n \geq 1$,

$$\begin{aligned} U_{n+1}(x) &= \cos[(n+1)\theta] + \frac{x\sin[(n+1)\theta]}{\sqrt{1-x^2}} \\ &= \cos(n\theta)\cos\theta - \sin(n\theta)\sin(\theta) + \frac{x}{\sqrt{1-x^2}}\left[\sin(n\theta)\cos\theta + \sin\theta\cos(n\theta)\right]; \end{aligned}$$

$$U_{n-1}(x) = \cos[(n-1)\theta] + \frac{x\sin[(n-1)\theta]}{\sqrt{1-x^2}}$$

$$= \cos(n\theta)\cos\theta + \sin(n\theta)\sin(\theta) + \frac{x}{\sqrt{1-x^2}}\left[\sin(n\theta)\cos\theta - \sin\theta\cos(n\theta)\right];$$

$$\begin{aligned}
U_{n+1}(x) + U_{n-1}(x) &= \cos(n\theta)\cos\theta - \sin(n\theta)\sin(\theta) + \frac{x}{\sqrt{1-x^2}}\left[\sin(n\theta)\cos\theta + \sin\theta\cos(n\theta)\right] \\
&\quad + \cos(n\theta)\cos\theta + \sin(n\theta)\sin(\theta) + \frac{x}{\sqrt{1-x^2}}\left[\sin(n\theta)\cos\theta - \sin\theta\cos(n\theta)\right] \\
&= 2\cos(n\theta)\cos\theta + 2\frac{x}{\sqrt{1-x^2}}\sin(n\theta)\cos\theta \\
&= 2\cos\theta\left[\cos(n\theta) + \frac{x}{\sqrt{1-x^2}}\sin(n\theta)\right] \\
&= 2xU_n(x).
\end{aligned}$$

Hence, we have shown that

$$U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x),$$

as desired. This result is very similar to the recursive relation for Chebyshev polynomials of the first kind, T_n, which also satisfy

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \qquad \text{for } n \geq 1,$$

the only difference being that $T_1(x) = x$, whereas $U_1(x) = 2x$. □

Solution to c). From Eq. (2.53), using $x = \cos\theta$, we have

$$\begin{aligned}
U_n(x) &= \cos(n\arccos x) + \frac{x\sin(n\arccos x)}{\sqrt{1-x^2}} \\
&= \frac{\cos(n\theta)\sin\theta + \cos\theta\sin(n\theta)}{\sin\theta} \\
&= \frac{\sin[\theta(n+1)]}{\sin\theta}.
\end{aligned}$$

Hence, applying Eq. (2.52) to two Chebyshev polynomials of the second kind, U_n and U_m, we get

$$\begin{aligned}
\langle U_n, U_m\rangle &= \int_{-1}^{1} U_n(x)U_m(x)\sqrt{1-x^2}\,\mathrm{d}x \\
&= \int_{\pi}^{0} \frac{\sin[\theta(n+1)]}{\sin\theta}\frac{\sin[\theta(m+1)]}{\sin\theta}\sin\theta\,\mathrm{d}[\cos\theta] \\
&= \int_{0}^{\pi} \frac{\sin[\theta(n+1)]}{\sin\theta}\frac{\sin[\theta(m+1)]}{\sin\theta}\sin^2\theta\,\mathrm{d}\theta \\
&= \int_{0}^{\pi} \sin[\theta(n+1)]\sin[\theta(m+1)]\,\mathrm{d}\theta \\
&= 0 \quad \text{if } n \neq m.
\end{aligned}$$

□

3 Numerical Integration

3.1 Newton-Cotes

Our starting point in the derivation of Newton-Cotes formulæ is the Lagrange interpolation of the function $f(x)$, given by

$$f(x) \approx \sum_{i=0}^{n} f_i \ell_i(x), \tag{3.1}$$

where $f_i \equiv f(x_i)$ and

$$\ell_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \qquad \text{for } i = 0, \dots, n. \tag{3.2}$$

Then, letting the *weights* w_i be given by

$$w_i := \int_a^b \ell_i(x)\, \mathrm{d}x, \tag{3.3}$$

we have the ***quadrature formula***

$$\boxed{\int_a^b f(x)\, \mathrm{d}x = \sum_{i=0}^{n} w_i f_i + \frac{1}{(n+1)!} \int_a^b \prod_{i=0}^{n} (x - x_i) f^{(n+1)}(\xi(x))\, \mathrm{d}x} \tag{3.4}$$

where $\xi \in [a, b]$. This formula is of the form

$$\int_a^b f(x)\, \mathrm{d}x = P + E,$$

where P is the *quadrature approximation*

$$P = \sum_{i=0}^{n} w_i f_i, \tag{3.5}$$

and E is the *error term*

$$E = \frac{1}{(n+1)!} \int_a^b \prod_{i=0}^{n} (x - x_i) f^{(n+1)}(\xi(x))\, \mathrm{d}x. \tag{3.6}$$

If we let $n = 1$ in (3.4), the approximation is linear and we get the *Trapezoidal Rule*, while for $n = 2$ the approximation is quadratic and yields *Simpson's* $1/3$ *rule*. Setting $n = 3$, on the other hand, yields a variation of Simpson's rule known as *Simpson's* $3/8$ *rule*.

$n = 1$: Trapezoidal rule

$$\int_{x_0}^{x_1} f(x)\,dx = \frac{h}{2}[f(x_0) + f(x_1)] - \frac{h^3}{12} f''(\xi), \quad \text{where} \quad x_0 < \xi < x_1. \tag{4.25}$$

$n = 2$: Simpson's rule

$$\int_{x_0}^{x_2} f(x)\,dx = \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] - \frac{h^5}{90} f^{(4)}(\xi), \quad \text{where} \quad x_0 < \xi < x_2. \tag{4.26}$$

$n = 3$: Simpson's Three-Eighths rule

$$\int_{x_0}^{x_3} f(x)\,dx = \frac{3h}{8}[f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)] - \frac{3h^5}{80} f^{(4)}(\xi), \tag{4.27}$$

where $x_0 < \xi < x_3$.

The following example illustrates the construction of Simpson's $3/8$ rule:

Example 11. *Derive a Newton-Cotes formula for* $\int_0^1 f(x)\,\mathrm{d}x$ *based on the nodes* $\{0, 1/3, 2/3, 1\}$.

Solution. Given the four nodes $\{x_0 = 0,\ x_1 = 1/3,\ x_2 = 2/3,\ x_3 = 1\}$, we put $n = 3$ in (3.4) and get a cubic polynimial to integrate in place of $f(x)$:

$$\int_0^1 f(x)\,\mathrm{d}x = \sum_{i=0}^{3} w_i f_i + \frac{1}{4!}\int_0^1 \prod_{i=0}^{3}(x - x_i) f^{(4)}(\xi(x))\,\mathrm{d}x.$$

Let's look at the approximation and error terms, one at a time. Starting with the approximation,

$$\begin{aligned} P &= w_0 f_0 + w_1 f_1 + w_2 f_2 + w_3 f_3 \\ &= \int_0^1 \frac{x - x_1}{x_0 - x_1}\frac{x - x_2}{x_0 - x_2}\frac{x - x_3}{x_0 - x_3} f_0\,\mathrm{d}x + \int_0^1 \frac{x - x_0}{x_1 - x_0}\frac{x - x_2}{x_1 - x_2}\frac{x - x_3}{x_1 - x_3} f_1\,\mathrm{d}x \\ &+ \int_0^1 \frac{x - x_0}{x_2 - x_0}\frac{x - x_1}{x_2 - x_1}\frac{x - x_3}{x_2 - x_3} f_2\,\mathrm{d}x + \int_0^1 \frac{x - x_0}{x_3 - x_0}\frac{x - x_1}{x_3 - x_1}\frac{x - x_2}{x_3 - x_2} f_3\,\mathrm{d}x. \end{aligned} \tag{3.7}$$

Instead of looking for a tricky change of variables here or just doing a straightforward, but LONG antiderivation, we make our lives easier by summoning Mathematica to the rescue:

```
Integrate[ (x - x1)/(x0 - x1) * (x - x2)/(x0 - x2) * (x - x3)/(x0 - x3),
        {x, a, b}] /. {x0 -> 0, x1 -> 1/3, x2 -> 2/3, x3 -> 1}
Integrate[ (x - x0)/(x1 - x0) * (x - x2)/(x1 - x2) * (x - x3)/( x1 - x3),
        {x, a, b}] /. {x0 -> 0, x1 -> 1/3, x2 -> 2/3, x3 -> 1}
Integrate[ (x - x0)/(x2 - x0) * (x - x1)/(x2 - x1) * (x - x3)/(x2 - x3),
        {x, a, b}] /. {x0 -> 0, x1 -> 1/3, x2 -> 2/3, x3 -> 1}
Integrate[ (x - x0)/(x3 - x0) * (x - x1)/(x3 - x1)*(x - x2)/( x3 - x2),
        {x, a, b}] /. {x0 -> 0, x1 -> 1/3, x2 -> 2/3, x3 -> 1}
```

The above yields

```
-(3/8) ((8a)/3 - (22 a^2)/3 + 8a^3 - 3a^4 +  b(-(8/3) + (22b)/3 - 8b^2 + 3b
    ^3))

9/8 (-4a^2 + (20a^3)/3 - 3a^4 + b(4b - (20b^2)/3 + 3b^3))

-(9/8) (-2a^2 + (16a^3)/3 - 3a^4 + b(2b - (16b^2)/3 + 3b^3))

3/8 (-((4a^2)/3) + 4a^3 - 3a^4 + b((4 b)/3 - 4b^2 + 3b^3))
```

That shows the result for a general $[a, b]$, but it simplifies greatly once we substitute actual values...In our case, with $a = 0$, $b = 1$, the results simplify to

$$\{w_0, w_1, w_2, w_3\} = \left\{\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}\right\},$$

so that all the weights add up to 1. Hence, the quadrature approximation is

$$P = \sum_{i=0}^{3} w_i f_i = \frac{1}{8}\left(f_0 + 3f_1 + 3f_2 + f_3\right). \tag{3.8}$$

In fact, for a general $[a, b]$, this result takes the form

$$P = \sum_{i=0}^{3} w_i f_i = \frac{b-a}{8}\left(f_0 + 3f_1 + 3f_2 + f_3\right). \tag{3.9}$$

Moreover, can also rewrite this in terms of $\mathrm{h} = x_{i+1} - x_i$. Since there are four interpolating nodes in the interval $[a, b]$ in this approach, $[a, b]$ splits into three subintervals $\{[a = x_0, x_1], [x_1, x_2], [x_2, x_3 = b]\}$. Thus, $\mathrm{h} = (b - a)/3$, and we have

$$P = \sum_{i=0}^{3} w_i f_i = \frac{3\mathrm{h}}{8}\left(f_0 + 3f_1 + 3f_2 + f_3\right). \tag{3.10}$$

(Whence the name for this approach: *Simpson's* 3/8 *rule*.)

Similarly, for the error term,

```
Integrate[
  1/4! * Product[x - Subscript[x, i], {i, 0, 3}], {x, 0, 1}]
        /. {Subscript[x, 0] -> 0, Subscript[x, 1] -> 1/3,
            Subscript[x, 2] -> 2/3, Subscript[x, 3] -> 1}
```

which yields $-1/6480$. Hence,

$$\begin{aligned} E &= \frac{1}{4!}\int_0^1 \prod_{i=0}^{3}(x - x_i)\, f^{(4)}(\xi(x))\,\mathrm{d}x \\ &= -\frac{1}{6480} f^{(4)}(\xi(x)). \end{aligned} \tag{3.11}$$

Note that here we were able to pull the factor $f^{(4)}(\xi(x))$ out of the integral because $\prod_{i=0}^{3}(x - x_i)$ is a fourth-order polynomial which does not change sign in the entire interval $[0,1]$; thus we were able to apply the *Weighted Mean Value Theorem for Integrals* (c.f., Theorem 5). The error term (3.11) generalizes to any interval $[a,b]$, by using $\mathrm{h} = (b-a)/3$ as before:

$$E = -\frac{3}{80}\mathrm{h}^5 f^{(4)}(\xi(x)). \tag{3.12}$$

□

The construction of the compositive versions of these formulæ is pretty straightforward. For instance, for Simpson's $1/3$ rule, an integral

$$I = \int_a^b f(x)\,\mathrm{d}x$$

is approximated by partitioning the interval $[a,b]$ into N evenly spaced segments $a = x_0 < x_1 < \cdots < x_N = b$ with spacing $\mathrm{h} \equiv (b-a)/N$, and then putting

$$I \approx \frac{\mathrm{h}}{3}\left[f_0 + 4\sum_{\substack{i=1\\ i \text{ is odd}}}^{N-1} f_i + 2\sum_{\substack{i=2\\ i \text{ is even}}}^{N-2} f_i + f_N \right]. \tag{3.13}$$

(Note that N must be even in order for (3.13) to work.)

Example 12. *Develop a composite version of the following formula and give the error term*

$$\int_{x_0}^{x_4} f(x)\,\mathrm{d}x = \frac{4\mathrm{h}}{3}\left[2f(x_1) - f(x_2) + 2f(x_3)\right] + \frac{14\mathrm{h}^2}{45} f^{(4)}(c), \tag{3.14}$$

where

$$\mathrm{h} = \frac{x_4 - x_0}{4}, \qquad x_1 = x_0 + \mathrm{h}, \qquad x_2 = x_0 + 2\mathrm{h}, \qquad x_3 = x_0 + 3\mathrm{h}, \qquad \textit{and} \qquad c \in (x_0, x_4).$$

Solution. Let's extend the integral in (3.14) to a larger, general interval $[a,b] = [x_0, x_n]$, and let us use, say, m panels. In order to mimic Eq. (3.14), in each of these panels $[x_i, x_{i+4}]$ we want three interior nodes, besides the two panel endpoints x_i and x_{i+4}. So, we have

$$\int_{x_0}^{x_n} f(x)\,\mathrm{d}x = \underbrace{\int_{x_0}^{x_4} f(x)\,\mathrm{d}x}_{1^{\text{st}}\text{ panel}} + \cdots + \underbrace{\int_{x_{n-4}}^{x_n} f(x)\,\mathrm{d}x}_{m^{\text{th}}\text{ panel}}.$$

Since we are using evenly spaced nodes, h remains unchanged for all i:

$$\mathrm{h} := \frac{x_{i+4} - x_i}{4} \qquad \text{for } i = 0, \ldots, n-4.$$

Putting all this together, we get

$$\int_{x_0}^{x_n} f(x)\,\mathrm{d}x = \frac{4\mathrm{h}}{3}\left[2\sum_{\substack{i=1\\ i \text{ is odd}}}^{n-1} f_i - \sum_{i\in I} f_i\right] + \frac{14\mathrm{h}^2}{45}\sum_{j=1}^{m} f^{(4)}(c_j), \tag{3.15}$$

where $I = \{2, 6, 10, \ldots, n-2\}$ and c_j is located in the j^{th} panel. $\square$

3.2 Gaussian Quadrature

Definition 9. *The set of nonzero functions $\{p_0, \ldots, p_n\}$ on the interval $[a,b]$ is* ***orthonormal*** *on $[a,b]$ if*

$$\int_a^b p_i(x)p_j(x)\,\mathrm{d}x = \delta_{ij}, \tag{3.16}$$

and, similarly, the set is simply ***orthogonal*** *on $[a,b]$ if*

$$\int_a^b p_i(x)p_j(x)\,\mathrm{d}x = \begin{cases} 0 & \text{if } i \neq j; \\ \neq 0 & \text{if } i = j. \end{cases} \tag{3.17}$$

Theorem 11. *If $\{p_0, \ldots, p_n\}$ is an orthogonal set of polynomials on the interval $[a,b]$, where $\deg(p_k) = k$, then $\{p_0, \ldots, p_n\}$ is a basis for $\Pi_n([a,b])$, the vector space of degree at most n polynomials on $[a,b]$.*

Theorem 12. *If $\{p_0, \ldots, p_n\}$ is an orthogonal set of polynomials on the interval $[a,b]$, and if $\deg(p_k) = k$, then p_k has k distinct roots in the interval (a,b).*

Definition 10. *The set of* ***Legendre polynomials*** *is given by*

$$p_k(x) = \frac{1}{2^k k!}\frac{\mathrm{d}^k}{\mathrm{d}x^k}\left[(x^2-1)^k\right] \qquad \text{for } 0 \leq k \leq n. \tag{3.18}$$

This set is orthogonal on the interval $[-1,1]$.

Hence, since the Legendre polynomials are orthogonal on the interval $[-1,1]$, Theorem 12 guarantees that the $(n+1)^{\text{st}}$ Legendre polynomial has $n+1$ roots $x_0, \ldots, x_n$ in $[-1,1]$. The ***Gaussian Quadrature*** (or, more specifically, the ***Gauss-Legendre Quadrature***) of a function is simply a linear combination of function evaluations at the Legendre roots. We achieve this by approximating the integral of the desired function by the integral of the interpolating polynomial, whose nodes are the Legendre roots.

The *Gauss-Legendre Quadrature*, for some fixed n, proceeds as follows: In the interval $[-1,1]$,

Step 1: Find the $n+1$ roots $x_0,\dots,x_n$ of the $(n+1)^{\text{st}}$ Legendre polynomial

$$p_{n+1}(x) = \frac{1}{2^{n+1}(n+1)!}\frac{\mathrm{d}^{n+1}}{\mathrm{d}x^{n+1}}\left[(x^2-1)^{n+1}\right].$$

Step 2: Lagrange-interpolate the function $f(x)$ that you want to integrate, as usual,

$$f(x) \approx \sum_{i=0}^{n} f_i \ell_i(x),$$

with

$$\ell_i(x) = \prod_{j\neq i}\frac{x-x_j}{x_i-x_j} \qquad \text{for } i=0,\dots,n,$$

except now the nodes $x_0,\dots,x_n$ inserted to compute the $\ell_i(x)$'s are the Legendre roots you found in Step 1.

Step 3: Letting

$$c_i := \int_{-1}^{1}\ell_i(x)\,\mathrm{d}x,$$

we have the quadrature

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n} c_i f_i. \tag{3.19}$$

Remark 2. *If the Gauss quadrature is to take place on a more general interval $[a,b]$, the problem needs to be translated back to $[-1,1]$. Using the substitution*

$$t = \frac{2x-a-b}{b-a}, \tag{3.20}$$

we get

$$\int_a^b f(x)\,\mathrm{d}x = \int_{-1}^{1} f\left(\frac{(b-a)t+b+a}{2}\right)\frac{b-a}{2}\,\mathrm{d}t. \tag{3.21}$$

Example 13. *Approximate the integral $\int_{-1}^{1}\cos(\pi x)\,\mathrm{d}x$ using $n=3$ Gaussian Quadrature.* [5]

Proof. For $n=3$ Gaussian Quadrature we need to use as nodes the three roots of the degree-3 Legendre polynomial

$$p_3(x) = \frac{1}{2^3 3!}\frac{\mathrm{d}^3}{\mathrm{d}x^3}\left[(x^2-1)^3\right]$$

[5] For this problem I switch to indexing starting at 1 in order to avoid confusion, since in my convention this problem is actually asking for $n=2$ Gaussian Quadrature.

$$\begin{aligned} &= \frac{1}{48}\left(120x^3 - 72x\right) \\ &= x\left(\frac{5}{2}x^2 - \frac{3}{2}\right). \end{aligned} \tag{3.22}$$

So one root is 0, and as for the other two,

$$\frac{5}{2}x^2 - \frac{3}{2} = 0 \implies x = \pm\sqrt{\frac{3}{5}}.$$

So the three roots are

$$x_1 = -\sqrt{\frac{3}{5}}, \quad x_2 = 0, \quad x_3 = \sqrt{\frac{3}{5}}.$$

Now we Lagrange-interpolate the function $f(x) = \cos(\pi x)$ as usual,

$$\cos(\pi x) \approx \sum_{i=1}^{3} \cos(\pi x_i)\ell_i(x), \tag{3.23}$$

except now the Lagrange polynomials ℓ_i are applied on the Legendre roots we have just found. Start with ℓ_1:

$$\begin{aligned} \ell_1(x) &= \frac{x - x_2}{x_1 - x_2}\,\frac{x - x_3}{x_1 - x_3} \\ &= \frac{x - 0}{-\sqrt{\frac{3}{5}} - 0}\,\frac{x - \sqrt{\frac{3}{5}}}{-\sqrt{\frac{3}{5}} - \sqrt{\frac{3}{5}}} \\ &= \frac{5}{6}x\left(x - \sqrt{\frac{3}{5}}\right). \end{aligned} \tag{3.24}$$

Integrating, we get the first coefficient

$$c_1 = \int_{-1}^{1} \ell_1(x)\,\mathrm{d}x = \int_{-1}^{1} \left[\frac{5}{6}x\left(x - \sqrt{\frac{3}{5}}\right)\right] \mathrm{d}x = \frac{5}{9}.$$

An identical calculation shows that the remaining two coefficients are $c_2 = 8/9$ and $c_3 = c_1 = 5/9$. Hence, putting it all together and integrating Eq. (3.23), we get we have the quadrature

$$\begin{aligned} \int_{-1}^{1} \cos(\pi x)\,\mathrm{d}x &= \sum_{i=1}^{3} c_i \cos(\pi x_i) \\ &= \frac{5}{9}\cdot\cos\left(-\sqrt{\frac{3}{5}}\pi\right) + \frac{8}{9}\cdot 1 + \frac{5}{9}\cdot\cos\left(\sqrt{\frac{3}{5}}\pi\right) \\ &\approx 0.045. \end{aligned}$$

□

Example 14. *Show how the Gaussian quadrature rule*

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \frac{5}{9} f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9} f(0) + \frac{5}{9} f\left(\sqrt{\frac{3}{5}}\right) \tag{3.25}$$

can be used for $\int_a^b f(x)\,\mathrm{d}x$*. Apply this result to evaluate* $\int_0^{\pi/2} x\,\mathrm{d}x$*.*

Solution. This is the same $n = 3$ Gaussian quadrature that we derived on the previous example. If the quadrature is to take place on a more general interval $[a, b]$, the problem needs to be translated back to $[-1, 1]$. Using the substitution from Eq. (3.20), i.e.,

$$t = \frac{2x - a - b}{b - a},$$

a Gaussian quadrature rule of the form

$$\int_{-1}^{1} f(t)\,\mathrm{d}t \approx \sum_{i=0}^{n} A_i f(t_i) \tag{3.26}$$

can be used over the interval $[a, b]$ as in Eq. (3.21); that is,

$$\int_a^b f(x)\,\mathrm{d}x = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{(b-a)t + b + a}{2}\right) \mathrm{d}t.$$

For this problem we have $[a, b] = [0, \pi/2]$. Hence, plugging into (3.21),

$$\begin{aligned} \int_0^{\frac{\pi}{2}} x\,\mathrm{d}x &= \frac{\frac{\pi}{2} - 0}{2} \int_{-1}^{1} f\left(\frac{(\frac{\pi}{2} - 0)\,t + \frac{\pi}{2} + 0}{2}\right) \mathrm{d}t \\ &= \frac{\pi}{4} \int_{-1}^{1} f\left(\frac{\pi}{4} t + \frac{\pi}{4}\right) \mathrm{d}t \\ &= \frac{\pi}{4} \left[\frac{5}{9} f\left(\frac{\pi}{4} - \frac{\pi}{4}\sqrt{\frac{3}{5}}\right) + \frac{8}{9} f\left(\frac{\pi}{4}\right) + \frac{5}{9} f\left(\frac{\pi}{4}\sqrt{\frac{3}{5}} + \frac{\pi}{4}\right)\right]. \qquad \text{(Using (3.26))} \end{aligned} \tag{3.27}$$

Then, since $f(x) = x$ in this case,

$$\begin{aligned} \int_0^{\frac{\pi}{2}} x\,\mathrm{d}x &= \frac{\pi}{4} \left[\frac{5}{9} \left(\frac{\pi}{4} - \frac{\pi}{4}\sqrt{\frac{3}{5}}\right) + \frac{8}{9} \left(\frac{\pi}{4}\right) + \frac{5}{9} \left(\frac{\pi}{4}\sqrt{\frac{3}{5}} + \frac{\pi}{4}\right)\right] \\ &= \frac{\pi^2}{8} \approx 1.2337. \end{aligned}$$

□

4 Norms

- ***Vector norms:***

$$\begin{aligned}
\|\mathbf{x}\|_\infty &= \max_{1\le i\le m} |x_i| \\
\|\mathbf{x}\|_1 &= \sum_{i=1}^{m} |x_i| \\
\|\mathbf{x}\|_2 &= \left(\sum_{i=1}^{m} |x_i|^2\right)^{1/2} \\
\|\mathbf{x}\|_p &= \left(\sum_{i=1}^{m} |x_i|^p\right)^{1/p}.
\end{aligned}$$

- ***Matrix norms:***

$$\|A\|_F = \sqrt{\sum_{j=1}^{n}\sum_{i=1}^{m} |a_{ij}|^2} = \sqrt{\operatorname{Tr}(A^*A)} = \sqrt{\operatorname{Tr}(AA^*)} \qquad (\textbf{\textit{Frobenius norm}})$$

$$\|A\|_p = \max_{\mathbf{x}\neq 0} \frac{\|A\mathbf{x}\|_p}{\|\mathbf{x}\|_p} = \max_{\|\mathbf{x}\|_p=1} \|A\mathbf{x}\|_p$$

$$\begin{aligned}
\|A\|_\infty = \max_{\|\mathbf{x}\|_\infty=1} \|A\mathbf{x}\|_\infty &= \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}| \\
&= \max_{1\le i\le m} \|\mathbf{r}_i\|_1 \\
&= \text{the largest absolute row sum.}
\end{aligned}$$

$$\begin{aligned}
\|A\|_1 = \max_{\|\mathbf{x}\|_1=1} \|A\mathbf{x}\|_1 &= \max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}| \\
&= \max_{1\le j\le n} \|\mathbf{c}_j\|_1 \\
&= \text{the largest absolute column sum.}
\end{aligned}$$

$$\|A\|_2 = \max_{\|\mathbf{x}\|_2=1} \|A\mathbf{x}\|_2 = \sqrt{|\lambda_{\max}|} \qquad (A \in \mathbb{R}^{m\times n};\ \lambda_{\max} \text{ is the largest eigenvalue of } A^\top A).$$

Example 15. *Show that $\|\mathbf{x}\|_\infty \le \|\mathbf{x}\|_2 \le \|\mathbf{x}\|_1$ for all $\mathbf{x} \in \mathbb{R}^m$. Moreover, show that for any $\mathbf{x} \in \mathbb{R}^m$ and any $A \in \mathbb{R}^{m\times n}$, the following inequalities hold:*

$$\|\mathbf{x}\|_1 \le m\|\mathbf{x}\|_\infty, \tag{4.1a}$$

$$\|\mathbf{x}\|_2 \le \sqrt{m}\|\mathbf{x}\|_\infty, \tag{4.1b}$$

$$\|A\|_\infty \le \sqrt{n}\|A\|_2, \tag{4.1c}$$

$$\|A\|_2 \le \sqrt{m}\|A\|_\infty. \tag{4.1d}$$

Solution. Let $\hat{i} \in [1, m]$ be the index that maximizes $|x_i|$; that is,

$$\|\mathbf{x}\|_\infty = \max_{1\le i\le m} |x_i| = |x_{\hat{i}}|.$$

Then,

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{m}|x_i|^2\right)^{1/2} = \left(|x_{\hat{i}}|^2 + \sum_{i\ne\hat{i}}|x_i|^2\right)^{1/2} \ge |x_{\hat{i}}| = \|\mathbf{x}\|_\infty. \qquad \surd$$

As for the second inequality, note that

$$\|\mathbf{x}\|_2^2 = \sum_{i=1}^{m}|x_i|^2 \le \sum_{i=1}^{m}|x_i|^2 + 2\sum_{i\ne j}|x_i||x_j| = \|\mathbf{x}\|_1^2 \quad \Longrightarrow \quad \|\mathbf{x}\|_2 \le \|\mathbf{x}\|_1. \qquad \surd$$

For Eq. (4.1a),it is clear that

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{m}|x_i| \le \sum_{i=1}^{m}|x_{\hat{i}}| = m|x_{\hat{i}}| = m\|\mathbf{x}\|_\infty. \qquad \surd$$

Similarly, for Eq. (4.1b) we have

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{m}|x_i|^2\right)^{1/2} \le \left(\sum_{i=1}^{m}|x_{\hat{i}}|^2\right)^{1/2} = \left(m|x_{\hat{i}}|^2\right)^{1/2} = \sqrt{m}|x_{\hat{i}}| = \sqrt{m}\|\mathbf{x}\|_\infty. \qquad \surd$$

As for the last two inequalities, I'm fairly convinced that there was a typo on the problem and m and n should be switched... Here's my argument: For Eq. (4.1c),

$$\begin{aligned}
\|A\|_\infty &= \max_{\mathbf{x}\ne0}\frac{\|A\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} \\
&\le \max_{\mathbf{x}\ne0}\frac{\|A\mathbf{x}\|_2}{\|\mathbf{x}\|_\infty} && \text{(Since } \|\cdot\|_\infty \le \|\cdot\|_2\text{)} \\
&\le \max_{\mathbf{x}\ne0}\frac{\|A\mathbf{x}\|_2}{\frac{\|\mathbf{x}\|_2}{\sqrt{m}}} && \text{(By Inequality (4.1b))} \\
&= \sqrt{m}\max_{\mathbf{x}\ne0}\frac{\|A\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \sqrt{m}\|A\|_2. \qquad \surd
\end{aligned}$$

Lastly, for Eq. (4.1d),

$$\begin{aligned}
\|A\|_2 &= \max_{\mathbf{x}\neq 0} \frac{\|A\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \\
&\leq \sqrt{n}\max_{\mathbf{x}\neq 0} \frac{\|A\mathbf{x}\|_\infty}{\|\mathbf{x}\|_2} && \text{(By Inequality (4.1b))} \\
&\leq \sqrt{n}\max_{\mathbf{x}\neq 0} \frac{\|A\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} && \text{(Since } \|\cdot\|_\infty \leq \|\cdot\|_2\text{)} \\
&= \sqrt{n}\|A\|_\infty.
\end{aligned}$$

□

Example 16. *For a nonsingular $A \in \mathbb{R}^{m\times m}$ show that the weighted norm $\|\mathbf{x}\|_A = \|A\mathbf{x}\|_p$ is a vector norm.*

Solution. In order for $\|\mathbf{x}\|_A$ to be a vector norm, it would have to satisfy the following properties:

(i) $\|\mathbf{x}\|_A \geq 0$ for all $\mathbf{x} \in \mathbb{C}^n$, and $\|\mathbf{x}\|_A = 0$ iff $\mathbf{x} = 0$.

Proof. We have, for any $\mathbf{x} \in \mathbb{C}^n$,

$$\|\mathbf{x}\|_A = \|A\mathbf{x}\|_p \geq 0$$

since $\|\cdot\|_p \geq 0$. In fact, since A is nonsingular $A\mathbf{x} = 0 \implies \mathbf{x} = 0$; thus $\|\mathbf{x}\|_A = \|A\mathbf{x}\|_p = 0 \iff \mathbf{x} = 0$. $\surd$

(ii) $\|\mathbf{x}+\mathbf{y}\|_A \leq \|\mathbf{x}\|_A + \|\mathbf{y}\|_A$, for all $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$.

Proof. For any $\mathbf{x}, \mathbf{y} \in \mathbb{C}^n$,

$$\begin{aligned}
\|\mathbf{x}+\mathbf{y}\|_A &= \|A(\mathbf{x}+\mathbf{y})\|_p \\
&= \|A\mathbf{x} + A\mathbf{y}\|_p && \text{(By linearity of } A\text{)} \\
&\leq \|A\mathbf{x}\|_p + \|A\mathbf{y}\|_p && \text{(By Triangle Inequality of } \|\cdot\|_p\text{)} \\
&= \|\mathbf{x}\|_A + \|\mathbf{y}\|_A. \quad \surd
\end{aligned}$$

(iii) $\|\alpha\mathbf{x}\|_A = |\alpha|\,\|\mathbf{x}\|_A$ for all $\alpha \in \mathbb{C}$ and $\mathbf{x} \in \mathbb{C}^n$.

Proof. For any $\mathbf{x} \in \mathbb{C}^n$ and $\alpha \in \mathbb{C}$, we have

$$\|\alpha\mathbf{x}\|_A = \|A(\alpha\mathbf{x})\|_p = \|\alpha A\mathbf{x}\|_p = |\alpha|\,\|A\mathbf{x}\|_p = |\alpha|\,\|\mathbf{x}\|_A,$$

where on the second equality we used linearity of A, and on the second-to-last equality we used the property $\|\alpha\mathbf{y}\|_p = |\alpha|\,\|\mathbf{y}\|_p$. $\surd$

Thus we have shown that $\|\cdot\|_A$ satisfies all properties of a vector norm. □

Example 17. *Evaluate the Frobenius matrix norm and the induced* 1-, 2-, *and* $\infty-$*norms for the following matrix:*

$$A = \begin{bmatrix} 4 & -2 & 4 \\ -2 & 1 & -2 \\ 4 & -2 & 4 \end{bmatrix}$$

Solution. For the Frobenius matrix norm, we can either just sum (the square of the absolute value of) all the entries, or take the trace of AA^* (or, equivalently, of A^*A). Let's show both approaches, since this is a relatively small matrix... Starting with the first approach,

$$\|A\|_F = \sqrt{4^2 + (-2)^2 + 4^2 + (-2)^2 + 1^2 + (-2)^2 + 4^2 + (-2)^2 + 4^2} = \sqrt{81} = 9. \qquad \surd$$

Now, for the second approach, since we are in the reals we swap $A^* \leftrightarrow A^\top$; moreover, note that A is symmetric, so $A = A^\top \implies AA^\top = A^2$. So we have

$$AA^\top = A^2 = \begin{bmatrix} 4 & -2 & 4 \\ -2 & 1 & -2 \\ 4 & -2 & 4 \end{bmatrix} \begin{bmatrix} 4 & -2 & 4 \\ -2 & 1 & -2 \\ 4 & -2 & 4 \end{bmatrix} = \begin{bmatrix} 36 & -18 & 36 \\ -18 & 9 & -18 \\ 36 & -18 & 36 \end{bmatrix}.$$

Then,

$$\|A\|_F = \sqrt{\mathrm{Tr}(AA^\top)} = \sqrt{36 + 9 + 36} = \sqrt{81} = 9. \qquad \surd$$

Now, for the induced p-norms, note that since A is symmetric, we have $\|A\|_\infty = \|A\|_1$. Moreover, the first and third columns of A are identical, i.e., $\mathbf{c}_1 = \mathbf{c}_3$. Thus,

$$\|A\|_\infty = \|A\|_1 = \max_{1\le j\le n} \|\mathbf{c}_j\|_1 = \|\mathbf{c}_1\|_1 = \|\mathbf{c}_3\|_1 = \sum_{i=1}^{3} |a_{i3}| = |4| + |-2| + |4| = 10. \qquad \surd$$

Lastly, we calculate $\|A\|_2$. Note that

$$AA^\top - \lambda I = \begin{bmatrix} 36 & -18 & 36 \\ -18 & 9 & -18 \\ 36 & -18 & 36 \end{bmatrix} - \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} = \begin{bmatrix} 36-\lambda & -18 & 36 \\ -18 & 9-\lambda & -18 \\ 36 & -18 & 36-\lambda \end{bmatrix}.$$

Hence, solving the characteristic polynomial,

$$0 = \det(AA^\top - \lambda I) = -\lambda^3 + 81\lambda^2 \implies \lambda = \{0, 81\}.$$

Thus $\lambda_{\max} = 81$, and therefore

$$\|A\|_2 = \sqrt{|81|} = 9. \qquad \surd$$

□

5 SVD Decomposition

Example 18. *Consider the matrix*

$$A = \begin{bmatrix} -2 & 11 \\ -10 & 5 \end{bmatrix}.$$

a) *Determine, on paper, the SVD of A in the form $A = U\Sigma V^\top$. List singular values, left singular vectors, and right singular vectors of A.*

b) *What are the 1-, 2-, ∞- and Frobenius norms of A?*

c) *Find A^{-1} not directly, but via the SVD.*

d) *Find the eigenvalues λ_1, λ_2 of A.*

e) *Verify that $\det(A) = \lambda_1\lambda_2$ and $|\det(A)| = \sigma_1\sigma_2$.*

Solution to a). For the calculation of the nonzero singular values $\sigma_i \in \Sigma$, we start by focusing on the direction of the largest action of A; thus setting $\sigma_1 = \|A\|_2 = \sqrt{|\lambda_{\max}|}$, where $\lambda_{\max}$ is the largest eigenvalue of $AA^\top$ (or, equivalently, of $A^\top A$). Then σ_2 would be the second largest such value and so on ... Hence, we start with the calculation

$$\begin{aligned} AA^\top - \lambda I &= \begin{bmatrix} -2 & 11 \\ -10 & 5 \end{bmatrix} \begin{bmatrix} -2 & -10 \\ 11 & 5 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \\ &= \begin{bmatrix} 125-\lambda & 75 \\ 75 & 125-\lambda \end{bmatrix}, \end{aligned}$$

which leads to

$$0 = \det(AA^\top - \lambda I) = \lambda^2 - 250\lambda + 10000 \implies \lambda_1 = 200, \lambda_2 = 50.$$

Thus we have the *singular values* σ_1 and σ_2 given by

$$\sigma_1 = \sqrt{|\lambda_{\max}|} = \sqrt{|\lambda_1|} = \sqrt{200} = 10\sqrt{2} \qquad \text{and} \qquad \sigma_2 = \sqrt{|\lambda_2|} = \sqrt{50} = 5\sqrt{2},$$

so that

$$\Sigma = \begin{bmatrix} 10\sqrt{2} & 0 \\ 0 & 5\sqrt{2} \end{bmatrix}.$$

Now, from the calculations

$$AA^\top = \left(U\Sigma V^\top\right)\left(U\Sigma V^\top\right)^\top = U\Sigma V^\top V \Sigma^\top U^\top = U\left(\Sigma\Sigma^\top\right)U^\top \tag{5.1a}$$

$$A^\top A = \left(U\Sigma V^\top\right)^\top\left(U\Sigma V^\top\right) = V\Sigma^\top U^\top U\Sigma V^\top = V\left(\Sigma^\top\Sigma\right)V^\top, \tag{5.1b}$$

and the fact that both U and V are (by construction) orthogonal, we see that $AA^\top$ and $A^\top A$ are *similar* to $\Sigma\Sigma^\top$ and $\Sigma^\top\Sigma$, respectively, so they share the same eigenvalues. Moreover, the column vectors $\mathbf{u}_{(i)} \in U$ and $\mathbf{v}_{(i)} \in V$ are the eigenvectors of $AA^\top$ and $A^\top A$, respectively. We proceed first with the calculation of the $\mathbf{v}_{(i)}$:

- For $\lambda_1 = 200$,

$$\begin{aligned}
\left(A^\top A - \lambda_1 I\right)\mathbf{v}_{(1)} &= 0 \\
\left(\begin{bmatrix} -2 & -10 \\ 11 & 5 \end{bmatrix}\begin{bmatrix} -2 & 11 \\ -10 & 5 \end{bmatrix} - \begin{bmatrix} 200 & 0 \\ 0 & 200 \end{bmatrix}\right)\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
\left(\begin{bmatrix} 104 & -72 \\ -72 & 146 \end{bmatrix} - \begin{bmatrix} 200 & 0 \\ 0 & 200 \end{bmatrix}\right)\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
\begin{bmatrix} -96 & -72 \\ -72 & -54 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}.
\end{aligned}$$

A straightforward Gaussian elimination yields

$$\left[\begin{array}{cc|c} 1 & \frac{3}{4} & 0 \\ 0 & 0 & 0 \end{array}\right] \implies v_2 = \alpha \in \mathbb{R},\, v_1 = -\frac{3}{4}\alpha.$$

Letting $\alpha = 4$ and normalizing, we have our first *right singular vector*

$$\mathbf{v}_{(1)} = \frac{1}{\sqrt{(-3)^2 + 4^2}}\begin{bmatrix} -3 \\ 4 \end{bmatrix} = \begin{bmatrix} -3/5 \\ 4/5 \end{bmatrix}.$$

- For $\lambda_2 = 50$,

$$\begin{aligned}
\left(A^\top A - \lambda_2 I\right)\mathbf{v}_{(2)} &= 0 \\
\left(\begin{bmatrix} 104 & -72 \\ -72 & 146 \end{bmatrix} - \begin{bmatrix} 50 & 0 \\ 0 & 50 \end{bmatrix}\right)\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
\begin{bmatrix} 54 & -72 \\ -72 & 96 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}.
\end{aligned}$$

A straightforward Gaussian elimination yields

$$\left[\begin{array}{cc|c} 1 & -\frac{4}{3} & 0 \\ 0 & 0 & 0 \end{array}\right] \implies v_2 = \alpha \in \mathbb{R},\, v_1 = \frac{4}{3}\alpha.$$

Letting $\alpha = 3$ and normalizing, we have our second right singular vector

$$\mathbf{v}_{(2)} = \begin{bmatrix} 4/5 \\ 3/5 \end{bmatrix}.$$

Thus we end up with

$$V = \begin{bmatrix} -3/5 & 4/5 \\ 4/5 & 3/5 \end{bmatrix}.$$

Now, for the computation of the *left singular vectors* $\mathbf{u}_{(i)} \in U$, we can proceed by calculating the eigenvectors of $AA^\top$, as we stated above; however, note that from the SVD decomposition of A (i.e., $A = U\Sigma V^\top$), we have $AV\Sigma^{-1} = U$, and thus

$$\mathbf{u}_{(i)} = \frac{1}{\sigma_i}A\mathbf{v}_{(i)}. \tag{5.2}$$

This yields

$$\begin{aligned}
\mathbf{u}_{(1)} &= \frac{1}{\sigma_1} A\mathbf{v}_{(1)} \\
&= \frac{1}{10\sqrt{2}} \begin{bmatrix} -2 & 11 \\ -10 & 5 \end{bmatrix} \begin{bmatrix} -3/5 \\ 4/5 \end{bmatrix} \\
&= \begin{bmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix};
\end{aligned}$$

$$\begin{aligned}
\mathbf{u}_{(2)} &= \frac{1}{\sigma_2} A\mathbf{v}_{(2)} \\
&= \frac{1}{5\sqrt{2}} \begin{bmatrix} -2 & 11 \\ -10 & 5 \end{bmatrix} \begin{bmatrix} 4/5 \\ 3/5 \end{bmatrix} \\
&= \begin{bmatrix} \sqrt{2}/2 \\ -\sqrt{2}/2 \end{bmatrix}.
\end{aligned}$$

Hence, in conclusion, we have

$$\underbrace{\begin{bmatrix} -2 & 11 \\ -10 & 5 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & -\sqrt{2}/2 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 10\sqrt{2} & 0 \\ 0 & 5\sqrt{2} \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} -3/5 & 4/5 \\ 4/5 & 3/5 \end{bmatrix}}_{V^\top}. \qquad \square$$

Solution to b).

$$\begin{aligned}
\|A\|_1 &= \max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}| = |11| + |5| = 16; \qquad \surd \\
\|A\|_2 &= \sqrt{|\lambda_{\max}|} = \sigma_1 = 10\sqrt{2}; \qquad \surd \\
\|A\|_\infty &= \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}| = |-10| + |5| = 15; \qquad \surd \\
\|A\|_F &= \sqrt{\mathrm{Tr}(A^\top A)} = \sqrt{104 + 146} = 5\sqrt{10}. \qquad \surd
\end{aligned} \qquad \square$$

Solution to c). Using the fact that both U and V are orthogonal matrices, we have

$$\begin{aligned}
A^{-1} &= \left(U\Sigma V^\top\right)^{-1} \\
&= V\Sigma^{-1}U^\top \\
&= \begin{bmatrix} -3/5 & 4/5 \\ 4/5 & 3/5 \end{bmatrix} \begin{bmatrix} \sqrt{2}/20 & 0 \\ 0 & \sqrt{2}/10 \end{bmatrix} \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & -\sqrt{2}/2 \end{bmatrix} \\
&= \begin{bmatrix} 1/20 & -11/100 \\ 1/10 & -1/50 \end{bmatrix}.
\end{aligned} \qquad \square$$

Solution to d). The characteristic polynomial yields

$$A - \lambda I = \begin{bmatrix} -2-\lambda & 11 \\ -10 & 5-\lambda \end{bmatrix}; \quad 0 = \det(A - \lambda I) = \lambda^2 - 3\lambda + 100 \implies \lambda_{\substack{1\\2}} = \frac{3 \pm i\sqrt{391}}{2}.$$

□

Solution to e). Note that on the one hand,

$$\det A = -2 \cdot 5 - (-10) \cdot 11 = 100,$$

while also

$$\lambda_1 \lambda_2 = \frac{3 + i\sqrt{391}}{2} \frac{3 - i\sqrt{391}}{2} = 100. \qquad \surd$$

Thus the equality $\det A = \lambda_1\lambda_2$ does hold. Moreover,

$$\sigma_1\sigma_2 = 10\sqrt{2} \cdot 5\sqrt{2} = 100 = |\det A|. \qquad \surd$$

□

6 QR Factorization

Example 19. *Apply Householder reflectors and Gram-Schmidt orthogonalization to find QR-factorization of the following matrices:*

$$A = \begin{bmatrix} 4 & 8 & 1 \\ 0 & 2 & -2 \\ 3 & 6 & 7 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 1 \\ 1 & -1 \\ 2 & 1 \end{bmatrix}.$$

Solution. We first tackle both matrices using Householder reflectors, and then Gram-Schmidt:

- *Householder:* A ***Householder reflector*** is a symmetric, orthogonal matrix H of the form

$$\boxed{H = I - 2\frac{\mathbf{u}\mathbf{u}^\top}{\mathbf{u}^\top\mathbf{u}}} \tag{6.1}$$

Applying such a matrix to an m-vector $\mathbf{x}$ reflects this vector accross an $(m-1)$-dimensional plane, while preserving its norm; thus $H\mathbf{x} = \mathbf{y}$, with $\|\mathbf{y}\| = \|\mathbf{x}\|$. Our goal is to find a decomposition of a matrix A such that

$$A = \underbrace{H_1 \cdots H_k}_{Q} R,$$

where Q is an orthogonal matrix formed by a product of Householder matrices $H_1, \ldots, H_k$, and R is a matrix whose square upper section is in upper-triangular form (if R is square, then of course it is an upper-triangular matrix). As for the ***Householder vector*** $\mathbf{u}$ we set [6]

$$\mathbf{u} = \mathbf{x} - \operatorname{sgn}(x_1)\|\mathbf{x}\|_2\mathbf{e}_{(1)}.$$

Applying this procedure to the matrix A, the vector columns $\mathbf{a}_{(i)}$ play the role of $\mathbf{x}$ from the discussion above; thus we start with

$$\mathbf{a}_{(1)} = \begin{bmatrix} 4 \\ 0 \\ 3 \end{bmatrix}.$$

Then,

$$\begin{aligned} \mathbf{u} &= \mathbf{a}_{(1)} - \operatorname{sgn}((a_{(1)})_1)\|\mathbf{a}_{(1)}\|_2\mathbf{e}_{(1)} \\ &= \begin{bmatrix} 4 \\ 0 \\ 3 \end{bmatrix} - \operatorname{sgn}(4) \cdot \left(\sqrt{|4|^2+|3|^2}\right) \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 4 \\ 0 \\ 3 \end{bmatrix} - \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

[6] In the case where $x_1 = 0$, we simply choose $\operatorname{sgn}(x_1) = 1$. Also note that this is a matter of convention; occasionally we may find the opposite sign in the literature, i.e., $\mathbf{u} = \mathbf{x} + \operatorname{sgn}(x_1)\|\mathbf{x}\|_2\mathbf{e}_{(1)}$.

$$= \begin{bmatrix} -1 \\ 0 \\ 3 \end{bmatrix}.$$

This Householder vector $\mathbf{u}$ yields the Householder matrix

$$\begin{aligned} H_1 &= I - 2\frac{\mathbf{u}\mathbf{u}^\top}{\mathbf{u}^\top\mathbf{u}} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - 2\left(\begin{bmatrix} -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 3 \end{bmatrix}\right)^{-1} \begin{bmatrix} -1 \\ 0 \\ 3 \end{bmatrix} \begin{bmatrix} -1 & 0 & 3 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{5}\begin{bmatrix} 1 & 0 & -3 \\ 0 & 0 & 0 \\ -3 & 0 & 9 \end{bmatrix} \\ &= \begin{bmatrix} 4/5 & 0 & 3/5 \\ 0 & 1 & 0 \\ 3/5 & 0 & -4/5 \end{bmatrix}. \end{aligned}$$

This yields

$$H_1 A = \begin{bmatrix} 4/5 & 0 & 3/5 \\ 0 & 1 & 0 \\ 3/5 & 0 & -4/5 \end{bmatrix} \begin{bmatrix} 4 & 8 & 1 \\ 0 & 2 & -2 \\ 3 & 6 & 7 \end{bmatrix} = \begin{bmatrix} 5 & 10 & 5 \\ 0 & 2 & -2 \\ 0 & 0 & -5 \end{bmatrix} = R.$$

Note how the resulting matrix R is already in upper-triangular form, so we only had to construct one Householder matrix. Since Householder matrices are both symmetric and orthogonal, we have $H = H^{-1}$; thus

$$H_1 A = R \implies A = H_1^{-1} R = H_1 R.$$

Thus, setting $Q \equiv H_1$, we have our QR decomposition of A:

$$\underbrace{\begin{bmatrix} 4 & 8 & 1 \\ 0 & 2 & -2 \\ 3 & 6 & 7 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 4/5 & 0 & 3/5 \\ 0 & 1 & 0 \\ 3/5 & 0 & -4/5 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} 5 & 10 & 5 \\ 0 & 2 & -2 \\ 0 & 0 & -5 \end{bmatrix}}_{R}.$$

We now follow the same steps for the matrix B, where

$$\mathbf{b}_{(1)} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}.$$

Now,

$$\mathbf{u} = \mathbf{b}_{(1)} - \operatorname{sgn}((b_{(1)})_1)\|\mathbf{b}_{(1)}\|_2 \mathbf{e}_{(1)}$$

$$
\begin{aligned}
&= \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}.
\end{aligned}
$$

This yields the Householder matrix

$$
\begin{aligned}
H_1 &= I - 2\frac{\mathbf{u}\mathbf{u}^\top}{\mathbf{u}^\top\mathbf{u}} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - 2\left(\begin{bmatrix} -1 & 1 & 2 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}\right)^{-1}\begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}\begin{bmatrix} -1 & 1 & 2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{3}\begin{bmatrix} 1 & -1 & -2 \\ -1 & 1 & 2 \\ -2 & 2 & 4 \end{bmatrix} \\
&= \begin{bmatrix} 2/3 & 1/3 & 2/3 \\ 1/3 & 2/3 & -2/3 \\ 2/3 & -2/3 & -1/3 \end{bmatrix}.
\end{aligned}
$$

Thus, we have

$$
H_1 B = \begin{bmatrix} 2/3 & 1/3 & 2/3 \\ 1/3 & 2/3 & -2/3 \\ 2/3 & -2/3 & -1/3 \end{bmatrix}\begin{bmatrix} 2 & 1 \\ 1 & -1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 0 & -1 \\ 0 & 1 \end{bmatrix}.
$$

That takes care of the first column; we must now work on the (truncated) second column

$$
\widehat{\mathbf{b}}_{(2)} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.
$$

The corresponding Householder vector is given by

$$
\begin{aligned}
\mathbf{u} &= \widehat{\mathbf{b}}_{(2)} - \operatorname{sgn}((\hat{b}_{(2)})_1)\|\widehat{\mathbf{b}}_{(2)}\|_2\widehat{\mathbf{e}}_{(1)} \\
&= \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \begin{bmatrix} \sqrt{2} \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} -1 + \sqrt{2} \\ 1 \end{bmatrix}.
\end{aligned}
$$

This yields the Householder matrix

$$
\widehat{H}_2 = \widehat{I} - 2\frac{\mathbf{u}\mathbf{u}^\top}{\mathbf{u}^\top\mathbf{u}}
$$

$$
\begin{aligned}
&= \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - 2\left(\begin{bmatrix}-1+\sqrt{2} & 1\end{bmatrix}\begin{bmatrix}-1+\sqrt{2}\\ 1\end{bmatrix}\right)^{-1}\begin{bmatrix}-1+\sqrt{2}\\ 1\end{bmatrix}\begin{bmatrix}-1+\sqrt{2} & 1\end{bmatrix}\\
&= \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - \frac{2}{4-2\sqrt{2}}\begin{bmatrix}3-2\sqrt{2} & -1+\sqrt{2}\\ -1+\sqrt{2} & 1\end{bmatrix}\\
&= \begin{bmatrix}1/\sqrt{2} & -1/\sqrt{2}\\ -1/\sqrt{2} & -1/\sqrt{2}\end{bmatrix}.
\end{aligned}
$$

Then, setting

$$H_2 = \begin{bmatrix}1 & 0 & 0\\ 0 & \widehat{H}_2 & \\ 0 & & \end{bmatrix},$$

we get

$$H_2H_1B = \begin{bmatrix}1 & 0 & 0\\ 0 & 1/\sqrt{2} & -1/\sqrt{2}\\ 0 & -1/\sqrt{2} & -1/\sqrt{2}\end{bmatrix}\begin{bmatrix}3 & 1\\ 0 & -1\\ 0 & 1\end{bmatrix} = \begin{bmatrix}3 & 1\\ 0 & -\sqrt{2}\\ 0 & 0\end{bmatrix} = R.$$

Thus, since $H = H^{-1}$ for Householder matrices, we get

$$H_2H_1B = R \implies B = (H_2H_1)^{-1}R = H_1^{-1}H_2^{-1}R = H_1H_2R.$$

Thus, setting

$$Q \equiv H_1H_2 = \begin{bmatrix}2/3 & 1/3 & 2/3\\ 1/3 & 2/3 & -2/3\\ 2/3 & -2/3 & -1/3\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & 1/\sqrt{2} & -1/\sqrt{2}\\ 0 & -1/\sqrt{2} & -1/\sqrt{2}\end{bmatrix} = \begin{bmatrix}2/3 & -1/(3\sqrt{2}) & -1/\sqrt{2}\\ 1/3 & 2\sqrt{2}/3 & 0\\ 2/3 & -1/(3\sqrt{2}) & 1/\sqrt{2}\end{bmatrix},$$

which is indeed orthogonal (i.e., $Q^{-1} = Q^\top$), we have our QR decomposition of B:

$$\underbrace{\begin{bmatrix}2 & 1\\ 1 & -1\\ 2 & 1\end{bmatrix}}_{B} = \underbrace{\begin{bmatrix}2/3 & -1/(3\sqrt{2}) & -1/\sqrt{2}\\ 1/3 & 2\sqrt{2}/3 & 0\\ 2/3 & -1/(3\sqrt{2}) & 1/\sqrt{2}\end{bmatrix}}_{Q}\underbrace{\begin{bmatrix}3 & 1\\ 0 & -\sqrt{2}\\ 0 & 0\end{bmatrix}}_{R}.$$

- *Gram-Schmidt:* In typical Gram-Schmidt fashion, if we start with an arbitrary basis $\{\mathbf{a}_{(1)}, \dots, \mathbf{a}_{(n)}\}$ we may form an orthonormal basis $\{\mathbf{q}_{(1)}, \dots, \mathbf{q}_{(n)}\}$ such that at the j^{th}-step we have

$$
\begin{aligned}
\mathbf{q}_{(j)} &= \frac{\mathbf{a}_{(j)} - \left(\mathbf{q}_{(1)}^\top\mathbf{a}_{(j)}\right)\mathbf{q}_{(1)} - \left(\mathbf{q}_{(2)}^\top\mathbf{a}_{(j)}\right)\mathbf{q}_{(2)} - \cdots - \left(\mathbf{q}_{(j-1)}^\top\mathbf{a}_{(j)}\right)\mathbf{q}_{(j-1)}}{\|\mathbf{a}_{(j)} - \left(\mathbf{q}_{(1)}^\top\mathbf{a}_{(j)}\right)\mathbf{q}_{(1)} - \left(\mathbf{q}_{(2)}^\top\mathbf{a}_{(j)}\right)\mathbf{q}_{(2)} - \cdots - \left(\mathbf{q}_{(j-1)}^\top\mathbf{a}_{(j)}\right)\mathbf{q}_{(j-1)}\|_2}\\
&= \frac{\mathbf{a}_{(j)} - \sum_{i=1}^{j-1}\left(\mathbf{q}_{(i)}^\top\mathbf{a}_{(j)}\right)\mathbf{q}_{(i)}}{\|\mathbf{a}_{(j)} - \sum_{i=1}^{j-1}\left(\mathbf{q}_{(i)}^\top\mathbf{a}_{(j)}\right)\mathbf{q}_{(i)}\|_2}. \qquad (6.2)
\end{aligned}
$$

This way, by construction, $\mathbf{q}_{(j)}$ is orthogonal to $\mathbf{q}_{(i)}$ for $i < j$ (i.e., $\mathbf{q}_{(i)}\mathbf{q}_{(j)}^\top = 0$ for all $i < j$), and also the vectors $\mathbf{q}_{(j)}$ have norm 1 by the denominator of Eq. (6.2). Thus, $\{\mathbf{q}_{(1)}, \dots, \mathbf{q}_{(n)}\}$ is indeed an orthonormal basis.

Now, if we use the notation

$$r_{ij} \equiv \mathbf{q}_{(i)}^\top \mathbf{a}_{(j)} \ \forall i < j \qquad \text{and} \qquad r_{jj} \equiv \|\mathbf{a}_{(j)} - \sum_{i=1}^{j-1} \left(\mathbf{q}_{(i)}^\top \mathbf{a}_{(j)}\right) \mathbf{q}_{(i)}\|_2,$$

then Eq. (6.2) can be rewritten as

$$\mathbf{a}_{(j)} = \sum_{i=1}^{j} \mathbf{q}_{(i)} r_{ij}$$

or, putting together all the column vectors in matrix form,

$$\underbrace{\begin{bmatrix} \mathbf{a}_{(1)} & \cdots & \mathbf{a}_{(n)} \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \mathbf{q}_{(1)} & \cdots & \mathbf{q}_{(n)} \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & & \ddots & \\ 0 & 0 & \cdots & r_{nn} \end{bmatrix}}_{R}. \tag{6.3}$$

This is how we get QR factorization of a matrix via ***Classical Gram-Schmidt*** (CGS). We will now apply this procedure to the two matrices given in this problem. Starting with A, we have

$$\mathbf{a}_{(1)} = \begin{bmatrix} 4 \\ 0 \\ 3 \end{bmatrix}; \qquad \mathbf{a}_{(2)} = \begin{bmatrix} 8 \\ 2 \\ 6 \end{bmatrix}; \qquad \mathbf{a}_{(3)} = \begin{bmatrix} 1 \\ -2 \\ 7 \end{bmatrix}.$$

Successive applications of Eq. (6.2) yield

$$\mathbf{q}_{(1)} = \frac{\mathbf{a}_{(1)}}{r_{11}} = \frac{\mathbf{a}_{(1)}}{\|\mathbf{a}_{(1)}\|_2} = \frac{1}{5}\begin{bmatrix} 4 \\ 0 \\ 3 \end{bmatrix} = \begin{bmatrix} 4/5 \\ 0 \\ 3/5 \end{bmatrix};$$

$$\mathbf{q}_{(2)} = \frac{\mathbf{a}_{(2)} - r_{12}\mathbf{q}_{(1)}}{r_{22}} = \frac{\mathbf{a}_{(2)} - r_{12}\mathbf{q}_{(1)}}{\|\mathbf{a}_{(2)} - r_{12}\mathbf{q}_{(1)}\|_2} = \frac{1}{2}\left(\begin{bmatrix} 8 \\ 2 \\ 6 \end{bmatrix} - 10\begin{bmatrix} 4/5 \\ 0 \\ 3/5 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix};$$

$$\mathbf{q}_{(3)} = \frac{\mathbf{a}_{(3)} - r_{13}\mathbf{q}_{(1)} - r_{23}\mathbf{q}_{(2)}}{r_{33}} = \frac{\mathbf{a}_{(3)} - r_{13}\mathbf{q}_{(1)} - r_{23}\mathbf{q}_{(2)}}{\|\mathbf{a}_{(3)} - r_{13}\mathbf{q}_{(1)} - r_{23}\mathbf{q}_{(2)}\|_2} = \frac{1}{5}\left(\begin{bmatrix} 1 \\ -2 \\ 7 \end{bmatrix} - 5\begin{bmatrix} 4/5 \\ 0 \\ 3/5 \end{bmatrix} + 2\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} -3/5 \\ 0 \\ 4/5 \end{bmatrix}.$$

On these calculations we used

$$r_{12} = \mathbf{q}_{(1)}^\top \mathbf{a}_{(2)} = \begin{bmatrix} 4/5 & 0 & 3/5 \end{bmatrix} \begin{bmatrix} 8 \\ 2 \\ 6 \end{bmatrix} = 10,$$

$$\implies r_{22} = \|\mathbf{a}_{(2)} - r_{12}\mathbf{q}_{(1)}\|_2 = \left\| \begin{bmatrix} 8 \\ 2 \\ 6 \end{bmatrix} - 10 \begin{bmatrix} 4/5 \\ 0 \\ 3/5 \end{bmatrix} \right\|_2 = \left\| \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} \right\|_2 = 2;$$

$$r_{13} = \mathbf{q}_{(1)}^\top \mathbf{a}_{(3)} = \begin{bmatrix} 4/5 & 0 & 3/5 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 7 \end{bmatrix} = 5,$$

$$r_{23} = \mathbf{q}_{(2)}^\top \mathbf{a}_{(3)} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 7 \end{bmatrix} = -2,$$

$$\implies r_{33} = \|\mathbf{a}_{(3)} - r_{13}\mathbf{q}_{(1)} - r_{23}\mathbf{q}_{(2)}\|_2 = \left\| \begin{bmatrix} 1 \\ -2 \\ 7 \end{bmatrix} - 5 \begin{bmatrix} 4/5 \\ 0 \\ 3/5 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right\|_2 = \left\| \begin{bmatrix} -3 \\ 0 \\ 4 \end{bmatrix} \right\|_2 = 5.$$

Hence we end up with the QR decomposition of A:

$$\underbrace{\begin{bmatrix} 4 & 8 & 1 \\ 0 & 2 & -2 \\ 3 & 6 & 7 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 4/5 & 0 & -3/5 \\ 0 & 1 & 0 \\ 3/5 & 0 & 4/5 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} 5 & 10 & 5 \\ 0 & 2 & -2 \\ 0 & 0 & 5 \end{bmatrix}}_{R}.$$

We now follow the same steps for B, where

$$\mathbf{b}_{(1)} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}; \qquad \mathbf{b}_{(2)} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}.$$

Successive applications of Eq. (6.2) yield

$$\mathbf{q}_{(1)} = \frac{\mathbf{b}_{(1)}}{r_{11}} = \frac{\mathbf{b}_{(1)}}{\|\mathbf{b}_{(1)}\|_2} = \frac{1}{3} \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2/3 \\ 1/3 \\ 2/3 \end{bmatrix};$$

$$\mathbf{q}_{(2)} = \frac{\mathbf{b}_{(2)} - r_{12}\mathbf{q}_{(1)}}{r_{22}} = \frac{\mathbf{b}_{(2)} - r_{12}\mathbf{q}_{(1)}}{\|\mathbf{b}_{(2)} - r_{12}\mathbf{q}_{(1)}\|_2} = \frac{1}{\sqrt{2}} \left(\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} - \begin{bmatrix} 2/3 \\ 1/3 \\ 2/3 \end{bmatrix} \right) = \begin{bmatrix} 1/(3\sqrt{2}) \\ -2\sqrt{2}/3 \\ 1/(3\sqrt{2}) \end{bmatrix}.$$

On these calculations we used

$$r_{12} = \mathbf{q}_{(1)}^\top \mathbf{b}_{(2)} = \begin{bmatrix} 2/3 & 1/3 & 2/3 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = 1,$$

$$\implies r_{22} = \|\mathbf{b}_{(2)} - r_{12}\mathbf{q}_{(1)}\|_2 = \left\| \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} - \begin{bmatrix} 2/3 \\ 1/3 \\ 2/3 \end{bmatrix} \right\|_2 = \left\| \begin{bmatrix} 1/3 \\ -4/3 \\ 1/3 \end{bmatrix} \right\|_2 = \sqrt{2}.$$

Hence we end up with the (reduced) QR decomposition of B:

$$\underbrace{\begin{bmatrix} 2 & 1 \\ 1 & -1 \\ 2 & 1 \end{bmatrix}}_{B} = \underbrace{\begin{bmatrix} 2/3 & 1/(3\sqrt{2}) \\ 1/3 & -2\sqrt{2}/3 \\ 2/3 & 1/(3\sqrt{2}) \end{bmatrix}}_{\widehat{Q}} \underbrace{\begin{bmatrix} 3 & 1 \\ 0 & \sqrt{2} \end{bmatrix}}_{\widehat{R}}.$$

To get the full QR decomposition of B, we introduce an extra column vector

$$\widehat{\mathbf{b}}_{(3)} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Then,

$$\begin{aligned}
\mathbf{q}_{(3)} &= \frac{\widehat{\mathbf{b}}_{(3)} - \left(\mathbf{q}_{(1)}^\top \widehat{\mathbf{b}}_{(3)}\right)\mathbf{q}_{(1)} - \left(\mathbf{q}_{(2)}^\top \widehat{\mathbf{b}}_{(3)}\right)\mathbf{q}_{(2)}}{\left\| \widehat{\mathbf{b}}_{(3)} - \left(\mathbf{q}_{(1)}^\top \widehat{\mathbf{b}}_{(3)}\right)\mathbf{q}_{(1)} - \left(\mathbf{q}_{(2)}^\top \widehat{\mathbf{b}}_{(3)}\right)\mathbf{q}_{(2)} \right\|_2} \\
&= \frac{2}{\sqrt{2}} \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \frac{2}{3} \begin{bmatrix} 2/3 \\ 1/3 \\ 2/3 \end{bmatrix} - \frac{1}{3\sqrt{2}} \begin{bmatrix} 1/(3\sqrt{2}) \\ -2\sqrt{2}/3 \\ 1/(3\sqrt{2}) \end{bmatrix} \right) \\
&= \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ -1/\sqrt{2} \end{bmatrix},
\end{aligned}$$

where we used

$$\mathbf{q}_{(1)}^\top \widehat{\mathbf{b}}_{(3)} = \begin{bmatrix} 2/3 & 1/3 & 2/3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \frac{2}{3},$$

$$\mathbf{q}_{(2)}^\top \widehat{\mathbf{b}}_{(3)} = \begin{bmatrix} 1/(3\sqrt{2}) & -2\sqrt{2}/3 & 1/(3\sqrt{2}) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{3\sqrt{2}},$$

$$\implies \left\| \widehat{\mathbf{b}}_{(3)} - \left(\mathbf{q}_{(1)}^\top \widehat{\mathbf{b}}_{(3)}\right)\mathbf{q}_{(1)} - \left(\mathbf{q}_{(2)}^\top \widehat{\mathbf{b}}_{(3)}\right)\mathbf{q}_{(2)} \right\|_2 = \left\| \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \frac{2}{3} \begin{bmatrix} 2/3 \\ 1/3 \\ 2/3 \end{bmatrix} - \frac{1}{3\sqrt{2}} \begin{bmatrix} 1/(3\sqrt{2}) \\ -2\sqrt{2}/3 \\ 1/(3\sqrt{2}) \end{bmatrix} \right\|_2$$

$$= \left\| \begin{bmatrix} 1/2 \\ 0 \\ -1/2 \end{bmatrix} \right\|_2 = \frac{\sqrt{2}}{2}.$$

Thus we have the full QR decomposition of B,

$$\underbrace{\begin{bmatrix} 2 & 1 \\ 1 & -1 \\ 2 & 1 \end{bmatrix}}_{B} = \underbrace{\begin{bmatrix} 2/3 & 1/(3\sqrt{2}) & 1/\sqrt{2} \\ 1/3 & -2\sqrt{2}/3 & 0 \\ 2/3 & 1/(3\sqrt{2}) & -1/\sqrt{2} \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} 3 & 1 \\ 0 & \sqrt{2} \\ 0 & 0 \end{bmatrix}}_{R}. \qquad \square$$

7 Iterative Methods

If we have a ***large and sparse*** linear system $A\mathbf{x} = \mathbf{b}$, we rewrite it in an equivalent form

$$\mathbf{x} = B\mathbf{x} + \mathbf{d}. \tag{7.1}$$

(How the matrix B and the vector $\mathbf{d}$ are defined depends on which iterative method we use; see respective subsections below.) Then, starting with an initial approximation $\mathbf{x}^{(0)}$ of the solution vector $\mathbf{x}$, we generate a sequence $\{\mathbf{x}^{(k)}\}$ by the iterative scheme

$$\mathbf{x}^{(k+1)} = B\mathbf{x}^{(k)} + \mathbf{d} \qquad k = 0, 1, \dots \tag{7.2}$$

We stop this algorithm when the ***relative residual norm*** satisfies

$$\frac{\|\mathbf{b} - A\mathbf{x}^{(k)}\|}{\|\mathbf{b}\|} \leq \epsilon \tag{7.3}$$

for some user-defined tolerance $\epsilon > 0$.

The first order of business for all such iterative methods is to rewrite the matrix A in the form $A = L + D + U$, where

L = lower triangular with zeroes on the diagonal;
D = diagonal;
U = upper triangular with zeroes on the diagonal.

That is,

$$\underbrace{\begin{pmatrix} \ddots & & & & U \\ & \ddots & & & \\ & & D & & \\ & & & \ddots & \\ L & & & & \ddots \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} 0 & & & & 0 \\ & \ddots & & & \\ & & 0 & & \\ & & & \ddots & \\ * & & & & 0 \end{pmatrix}}_{L} + \underbrace{\begin{pmatrix} a_{11} & & & & 0 \\ & \ddots & & & \\ & & a_{ii} & & \\ & & & \ddots & \\ 0 & & & & a_{nn} \end{pmatrix}}_{D} + \underbrace{\begin{pmatrix} 0 & & & & * \\ & \ddots & & & \\ & & 0 & & \\ & & & \ddots & \\ 0 & & & & 0 \end{pmatrix}}_{U}.$$

7.1 Jacobi Iteration

For the Jacobi method we rewrite $(L + D + U)\mathbf{x} = \mathbf{b}$ as

$$D\mathbf{x} = \mathbf{b} - (L + U)\mathbf{x},$$

which implies

$$\begin{aligned}\mathbf{x} &= D^{-1}\left(\mathbf{b} - (L+U)\mathbf{x}\right)\\ &= \underbrace{-D^{-1}(L+U)}_{B}\mathbf{x} + \underbrace{D^{-1}\mathbf{b}}_{\mathbf{d}}.\end{aligned}$$

Thus, from $\mathbf{x} = D^{-1}\left(\mathbf{b} - (L+U)\mathbf{x}\right)$, we see that the Jacobi algorithm is given by

$$\boxed{x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1\\ i\neq j}}^{n} a_{ij}x_j^{(k)}\right)} \tag{7.4}$$

7.2 Gauss-Seidel Iteration

For the Gauss-Seidel method we rewrite $(L+D+U)\mathbf{x} = \mathbf{b}$ as

$$(L+D)\mathbf{x} = \mathbf{b} - U\mathbf{x},$$

which implies

$$\begin{aligned}\mathbf{x} &= (L+D)^{-1}\left[\mathbf{b} - U\mathbf{x}\right]\\ &= \underbrace{-(L+D)^{-1}U}_{B}\mathbf{x} + \underbrace{(L+D)^{-1}\mathbf{b}}_{\mathbf{d}}.\end{aligned}$$

If we write $\mathbf{x} = (L+D)^{-1}\left[\mathbf{b} - U\mathbf{x}\right]$ in the iterative form

$$\mathbf{x}^{(k+1)} = (L+D)^{-1}\left[\mathbf{b} - U\mathbf{x}^{(k)}\right],$$

we see that this expression may also be written as

$$\mathbf{x}^{(k+1)} = D^{-1}\left[\mathbf{b} - L\mathbf{x}^{(k+1)} - U\mathbf{x}^{(k)}\right]. \tag{7.5}$$

This shows the main difference between the Jacobi and Gauss-Seidel algorithms: Gauss-Seidel updates the new values as soon as they become available, whereas Jacobi always updates the new values in terms of the values from the previous iteration.

Hence, from Eq. (7.5) we see that the Gauss-Seidel algorithm is given by

$$\boxed{x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right)} \tag{7.6}$$

7.3 Successive Overrelaxation (SOR)

The Gauss-Seidel method can be slow to converge in some applications. SOR is a fairly minimal adjustment that can help performance tremendously. We introduce a ***relaxation factor*** ω which is typically in the range $1 < \omega < 2$,[7] so that $\omega A\mathbf{x} = \omega\mathbf{b}$ takes the form

$$(\omega L + \omega D + \omega U)\mathbf{x} = \omega\mathbf{b}.$$

This in turn implies

$$\begin{aligned}(D + \omega L)\mathbf{x} &= \omega\mathbf{b} - \omega U\mathbf{x} + (1-\omega)D\mathbf{x} \\ \implies \mathbf{x} &= \underbrace{(D+\omega L)^{-1}\left[(1-\omega)D - \omega U\right]}_{B}\mathbf{x} + \underbrace{\omega(D+\omega L)^{-1}\mathbf{b}}_{\mathbf{d}}.\end{aligned}$$

This yields the SOR algorithm

$$\boxed{x_i^{(k+1)} = \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right) + (1-\omega)x_i^{(k)}} \tag{7.7}$$

7.4 Convergence of Iterative Methods)

Definition 11. *A matrix $A = [a_{ij}]_{n\times n}$ is said to be* **strictly (row)-diagonally-dominant** *if, for each $1 \le i \le n$,*

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|.$$

Theorem 13. *If A is strictly (row)-diagonally-dominant, then both the Jacobi and Gauss-Seidel algorithms converge to the solution of $A\mathbf{x} = b$ for any arbitrary initial guess $\mathbf{x}^{(0)}$.*

Definition 12. *A matrix A is said to be* **positive-definite** *if, for any nonzero $\mathbf{x}$,*

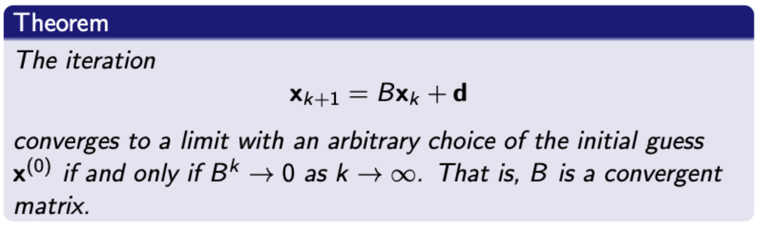
$$\mathbf{x}^* A\mathbf{x} > 0.$$

It is **positive-semidefinite** *if, for every $\mathbf{x}$,*

$$\mathbf{x}^* A\mathbf{x} \ge 0.$$

[7] If $\omega > 1$, then when calculating the $(k+1)^{\text{st}}$ iteration there is more weight put on the current values than when $\omega < 1$. The case $\omega = 1$ reduces to the Gauss-Seidel method.

Theorem 14. *If A is symmetric positive-definite, then Gauss-Seidel converges to the solution of $A\mathbf{x} = b$ for any arbitrary initial guess $\mathbf{x}^{(0)}$. SOR also converges under this condition if, in addition, we also impose that $0 < \omega < 2$.*

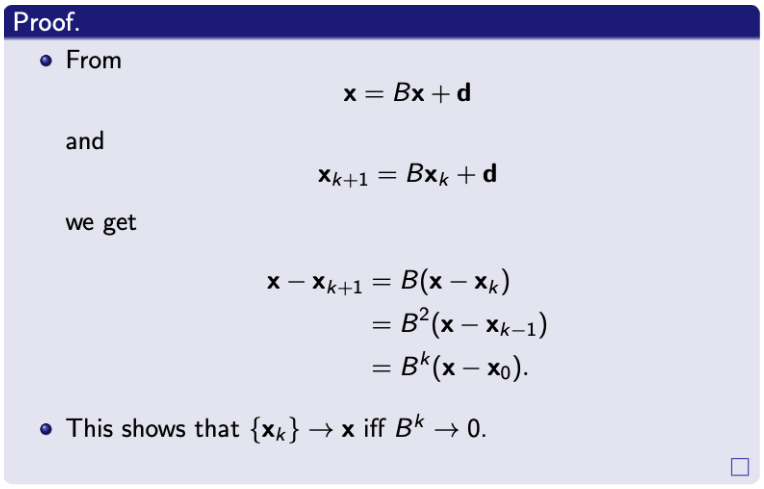


Theorem

The iteration

$$\mathbf{x}_{k+1} = B\mathbf{x}_k + \mathbf{d}$$

converges to a limit with an arbitrary choice of the initial guess $\mathbf{x}^{(0)}$ if and only if $B^k \to 0$ as $k \to \infty$. That is, B is a convergent matrix.



Proof.

- From
$$\mathbf{x} = B\mathbf{x} + \mathbf{d}$$
and
$$\mathbf{x}_{k+1} = B\mathbf{x}_k + \mathbf{d}$$
we get
$$\begin{aligned}\mathbf{x} - \mathbf{x}_{k+1} &= B(\mathbf{x} - \mathbf{x}_k)\\ &= B^2(\mathbf{x} - \mathbf{x}_{k-1})\\ &= B^k(\mathbf{x} - \mathbf{x}_0).\end{aligned}$$
- This shows that $\{\mathbf{x}_k\} \to \mathbf{x}$ iff $B^k \to 0$.

□

Theorem 15. *Let λ be an eigenvalue of A. Then, for any consistent pair of matrix-vector norms, we have*

$$|\lambda| \leq \|A\|. \tag{7.8}$$

In particular, $\rho(A)$, the **spectral radius** *of A (largest eigenvalue in magnitude) is bounded by $\|A\|$: $\rho(A) \leq \|A\|$.*

Proof. From $A\mathbf{x} = \lambda\mathbf{x}$, we have

$$\|\lambda\mathbf{x}\| = \|A\mathbf{x}\| \leq \|A\|\,\|\mathbf{x}\|.$$

But $\|\lambda\mathbf{x}\| = |\lambda|\,\|\mathbf{x}\|$, so

$$|\lambda|\,\|\mathbf{x}\| \leq \|A\|\,\|\mathbf{x}\| \quad \Longrightarrow \quad |\lambda| \leq \|A\|.$$

Since this inequality holds for *any* eignevalue λ, it also applies to the one with largest magnitude, i.e., the spectral radius. □

Remark 3. *Using the* Jordan Canonical Theorem, *it can be shown that the matrix B is convergent iff $\rho(B) < 1$. But, by the theorem above, $\rho(B) < \|B\|$, so if we can show that $\|B\| < 1$, then we have another way of showing convergence. (This is convenient, since computing a norm is usually far less involved than computing eigenvalues.)*