MA378 – Numerical Analysis II (NA2) 2016/2017

Niall Madden

March 29, 2017

Original materials contained here are licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.

Chapter 0

MA378: Introduction

0.1 Welcome!

This is a one-semester upper-level module on numerical analysis, taken by various groups of students, including those in Mathematics & Education, Mathematical Science, Mathematics, and Applied Mathematics.

The basic information for the course is as follows:

Lecturer: Dr Niall Madden, School Mathematics, Statistics and Applied Mathematics.
Office: AdB-1013, Arás de Brún.
Email: Niall.Madden@NUIGalway.ie
Phone (091 49) 3803.

Lectures: Monday at 17.00 in AC215, and Wednesday at 13.00 in AC213.

Labs/tutorials: To be arranged.

Assessment:

- Three MATLAB labs, each worth 2%.
- Two written assignments, each worth 8%.
- An in-class test, worth 8%.
- A 2-hour exam at the end of the semester, worth 70%.

The labs provide an opertunity for you to implement the algorithms we study, as well as their extensions and limitations. The written assignments promote in-depth engagement with specific topics, while the class test encourages one to take a broad view of the module.

0.1.1 Books

Süli and Mayers [SM03] *An Introduction to Numerical Analysis*, Cambridge University Press. Library call number is 519.4 MAY.

This is the main text for the course. All topics in the semester are covered in the book, except for one very short section on numerical differentiation.

GW Stewart [S96] *Afternotes on Numerical Analysis*, (519.4 STE) serves as a good companion text. It covers much of the earlier parts of the course. The 2nd volume in the series [S98] *Afternotes goes to Graduate School* includes a section on Cubic Splines. (Differential equations are not covered).

These books are freely available online through the NUI Galway library, because we have a subscription to SIAM ebooks.

Cleve Moler [M04] *Numerical Computing with MATLAB*. The emphasis is on the implementation of algorithms in MATLAB, but the techniques are well explained and there are some nice exercises. Also, it is freely available online.

0.1.2 Website

The course web-site is hosted on BlackBoard. There you'll find various pieces of information, including these notes. Much of the material can also be accessed directly at `http://www.maths.nuigalway.ie/~niall/MA378`. All grades will be communicated through Backboard. For lab assignments, you will have to submit you work through Blackboard.

The **lecture notes** are organised section-by-section. The content relevant to a particular class should be available 24 hours before the class. The notes contain most of the main remarks, statements of theorems, results and exercises. However, they will not contain proofs of theorems, examples, solutions to exercises, etc. Please let me know of the typos and mistakes that you spot. Each section of the notes has a set of exercises. *The homework assignments, class test, and final exam will be primarily based on these exercises*.

If possible, *please print out the day's notes and bring them with you to class* (or be able to access them during the class). Slides used during class are based on the notes. They are will be on the website too.

0.1.3 Topics

Numerical analysis is the design, analysis and implementation of numerical algorithms that yield *exact* or *approx-*

imate solutions to mathematical problems. The specific problems we will study are

1. Interpolation I: Polynomial interpolation.
2. Interpolation II: Piecewise polynomial "splines".
3. Numerical Integration I: Newton-Cotes quadrature.
4. Numerical Integration II: Gaussian quadrature.
5. Numerical solution of Boundary Value Problems by the Finite Element Method.

Although these might seem like diverse topics, in each case we will attempt to find the most suitable polynomial that solves the problems.

0.1.4 What is NA2 really about?

The big idea is:

> *Suppose we have a problem to solve, for which we know there is a solution, but that the solution is very hard to find. Or maybe impossible to find. We replace the problem with one that is easier, but has a similar solution, and solve that instead.*

While there are many variations, there is a single core idea we will return to again and again. If the difficult problem is expressed in terms of some complicated function, then we approximate that function with a simple polynomial, usually of degree 3 or less. Choosing that polynomial is the "design part", and is usually quite interesting.

We then have to devise a set of steps for computing that polynomial, and finding the solution to our particular problem. This is the "implementation" stage. When this is done by hand, it can easily become very boring. But through the use of computers, it take on an important, creative role in the process.

Finally we have the "analysis" part: this is the most interesting and mathematically challenging aspect: *can we say how close our approximate solution is to true solution?* That we can answer this question in a precise manner is a bit surprising. For how can I give an accurate estimate for how close my approximation is to the true solution, when I don't know what that true solution is?

0.1.5 Mathematical Preliminaries

Anyone who can remember their first and second years of analysis and algebra should be able to handle this course. Students who know a little about boundary-value differential equations will find a certain sections a little easier than those who haven't.

If it is a while since you covered basic calculus, you will find it very helpful to revise the following: the Intermediate Value Theorem; **Rolle's Theorem**; The Mean Value Theorem; Taylor's Theorem, and the triangle inequality: $|a+b| \leqslant |a|+|b|$. You'll find them in any good text book, e.g., Appendix 1 of Süli and Mayers.

0.1.6 Licensing

Except where indicated otherwise, the notes are licences use the Creative Commons Attribution-ShareAlike 4.0 International License. You are free to copy and redistribute the material in any medium or format, and to modify them any way you like, providing you give appropriate credit, and use the same licence on derived materials.

Most of the images and illustrations have been created by me. The rest —mainly photos—are, to the best of my knowledge, in the public domain, and sources have been provided.

Most of the exercises, and some examples, are taken from the text-books listed below. Where this has been done, references are provided.

Please email Niall.Madden@NUIGalway if you think there is an attribution error.

Bibliography

[SM03] Endre Süli and David Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, 2003. 519.4 MAY.

[S96] G.W. Stewart, *Afternotes on Numerical Analysis*, SIAM, 1996. 519.4 STE. `http://epubs.siam.org/doi/book/10.1137/1.9781611971491`

[S98] G.W. Stewart, *Afternotes goes to Graduate School*, SIAM, 1998. 519.4 STE. `http://epubs.siam.org/doi/book/10.1137/1.9781611971422`

[SB92] Stoer and Bulirsch, *An Introduction to Numerical Analysis*, Springer.

[QSS00] Quarteroni, Sacco and Saleri, *Numerical Mathematics*, Springer.

[M04] Cleve Moler, *Numerical Computing with MATLAB*, Cambridge University Press. Also available free from http://www.mathworks.com/moler

[E02] James F Epperson, *An introduction to numerical methods and analysis*. `519.4EPP`

Chapter 1

Polynomial Interpolation

1.1 Introduction

Suppose that we have a two sets of $n+1$ real numbers $\{x_i\}_{i=0}^{n+1}$ and $\{y_i\}_{i=0}^{n+1}$, and that the x_i are *strictly* increasing: $x_0 < x_1 < x_2 < \cdots < x_n$. *Interpolation* problems are of the form: *Find a function* p*, that is continuous and defined on* $[x_0, x_n]$*, such that*

$$p(x_k) = y_k, \quad \textit{for} \quad k = 0, 1, \ldots, n.$$

We say that p *interpolates* the points (x_0, y_0), (x_1, y_1), ..., (x_n, y_n).

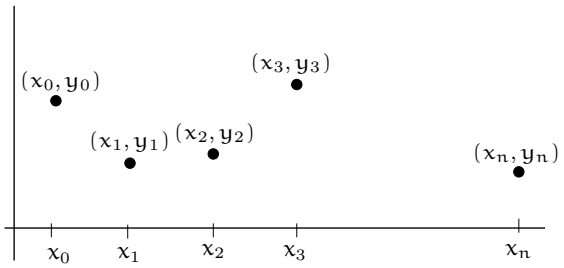


Fig. 1.1: Some points to interpolate

Why Bother? There are several possibilities, including

- The points belong to an underlying, but unknown function. We wish to establish likely values of f at points other than x_0, x_1, ..., x_n. The values of f may have been obtained from physical experiments, or numerical procedures (e.g., Newton's method for initial value problems). Or it may be that some values of the function are easily available. For example $2! = 2$, and $3! = 6$, but what about $2\frac{1}{2}!$ or $\pi!$?
- We may know the function, but prefer to work with an interpolant to it. For example, in order to estimate derivatives or integrals of a function.

Applications? In mathematics, from number theory to information theory, and nearly every aspect of numerical analysis. Elsewhere, the methods are used in fields ranging from aircraft design to computer animation.

The main reference for this section is Chapter 6 of [SM03]. See also, Lectures 18–20 of [S96].

1.1.1 Polynomial Interpolation

Definition 1.1.1. $\mathcal{P}_n$ is the set of polynomials of degree less than or equal to n and real-valued coefficients, i.e., $p_n \in \mathcal{P}_n$ if

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n,$$

where $a_i \in \mathbb{R}$.

Examples:
Take notes:

It is particularly important to note that if p_n and q_n both belong to $\mathcal{P}_n$, then so too does their sum.

The Polynomial Interpolation Problem comes in two forms.

Polynomial Interpolation Problem 1 (PIP1): *Given is set of points* $x_0 < x_1 < \cdots < x_n$*, and a set of real numbers* $y_0, y_1, \ldots, y_n$*, find* $p_n \in \mathcal{P}_n$ *such that*

$$p_n(x_k) = y_k, \quad \textit{for } k = 0, 1, \ldots, n. \tag{1.1}$$

Polynomial Interpolation Problem 2 (PIP2): *Given is set of points* $x_0 < x_1 < \cdots < x_n$*, and a function* $f : [x_0, x_n] \to \mathbb{R}$*, find* $p_n \in \mathcal{P}_n$ *such that*

$$p_n(x_k) = f(x_k), \quad \textit{for } k = 0, 1, \ldots, n. \tag{1.2}$$

Clearly PIP2 is just PIP1 with $y_k = f(x_k)$.

The questions that we must ask (and answer) are

(i) Is there a solution to the polynomial interpolation problem.

(ii) Is it unique?

(iii) How do we find it?

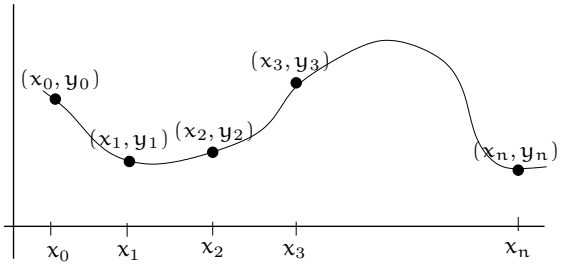


Fig. 1.2: A possible interpolation of the points in Figure 1.1

(iv) How accurate is it? That is, if f is the underlying function (i.e., $f(x_k) = y_k$), can we find an upper bound for

$$\max_{x_0 \leqslant x \leqslant x_n} \{|f(x) - p_n(x)|\}?$$

1.1.2 Exercises

Exercise 1.1. Suppose that $p \in \mathcal{P}_m$ and $q \in \mathcal{P}_n$.

(a) What is the maximum possible degree of $p + q$?

(b) What is the minimum possible degree of $p - q$?

(c) What is the maximum possible degree of pq?

Exercise 1.2. Find out what a *vector space* is. Convince yourself that $\mathcal{P}_n$ is a vector space.

Exercise 1.3. (a) Is it always possible to find a polynomial of degree 1 that interpolates the single point (x_0, y_0)? If so, how many such polynomials are there? Explain your answer.

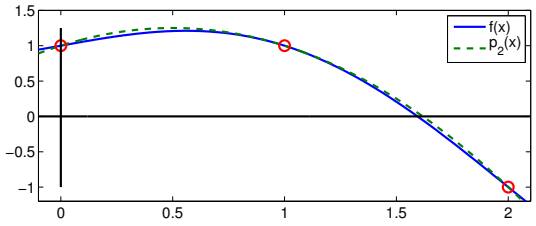
(b) Is it always possible to find a polynomial of degree 1 that interpolates the two points (x_0, y_0) and (x_1, y_1)? If so, how many such polynomials are there? Explain your answer.

(c) Is it ever possible to find a polynomial of degree 1 that interpolates the three points (x_0, y_0), (x_1, y_1), and (x_2, y_2)? If so, give an example.

1.2 Finding the polynomial

1.2.1 An example

Example 1.2.1. Show that the polynomial of degree 2 that interpolates $f(x) = 1 - x + \sin(\pi x/2)$ at the points $x_0 = 0$, $x_1 = 1$ and $x_2 = 2$ is $p_2 = -x^2 + x + 1$.



Take notes:

1.2.2 Uniqueness

It is not hard to convince ourselves that $-x^2 + x + 1$ is the solution to the PIP in Example 1.2.1. But how to we know we have found the *only* solution? More generally, *under what conditions is there exactly one polynomial that solves the PIP*?

Lemma 1.2.2. *If $p_n \in \mathcal{P}_n$ has $n+1$ zeros, then $p_n \equiv 0$ (i.e., $p_n(x) = 0$ for all x).*

Take notes:

Theorem 1.2.3 (There is a unique solution to the PIP)**.** *There is at most one polynomial of degree at most n that interpolates the $n+1$ points (x_0, y_0), (x_1, y_1), ..., (x_n, y_n) where x_0, x_1, ..., x_n are distinct.*

Take notes:

1.2.3 The Vandermonde matrix method

In Example 1.2.1 we gave a polynomial the solves a particular PIP, but this didn't tell us *how* to find the solution. It turns out that the most obvious approach may not be the best.

Suppose we are trying to solve the problem as follows: *find p_2 such that*

$$p_2(x_0) = y_0, \quad p_2(x_1) = y_1, \quad \textit{and} \quad p_2(x_2) = y_2.$$

Since $p_2(x)$ is of the form $a_0 + a_1 x + a_2 x^2$, this just amounts the finding the values of the coefficients a_0, a_1, and a_2. One might be tempted to solve for them using the system of equations

$$\begin{aligned} a_0 + a_1 x_0 + a_2 x_0^2 &= y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 &= y_1 \\ a_0 + a_1 x_2 + a_2 x_2^2 &= y_2. \end{aligned}$$

This is known as the *Vandermonde System*. [1]. Writing it in matrix-vector format we get:

$$\begin{pmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix}, \quad \text{or} \quad Va = y. \tag{1.3}$$

But this may not be a good idea. (*You can skip the next bit if you did not take MA385*).

In MA385 we learned about the relationship between the *condition number* of a matrix, V, and the relative error in the (numerical) solution to a matrix-vector equation with V as the coefficient matrix. The condition number is

$$\kappa(V) = \|V\|\|V^{-1}\|,$$

for some subordinate matrix norm $\|\cdot\|$.

Example 1.2.4. ([S96, Lecture 18]) Suppose $x_0 = 100$, $x_1 = 101$ and $x_2 = 102$. Then it is not hard to check that

$$\|X\|_\infty = \max_i \sum_j |X_{ij}| = 10,507.$$

Also,

$$V^{-1} = \frac{1}{2} \begin{pmatrix} 10302 & -20400 & 10100 \\ -203 & 404 & -201 \\ 1 & -2 & 1 \end{pmatrix},$$

so $\|V^{-1}\|_\infty = 20401$. So $\kappa(V) = 214,353,307$.

[1] *Alexandre-Théophile Vandermonde* (France, 1735–1796) only began studying mathematics at the age of 35. Prior to that, he was a professional musician. Is credited for inventing the Vandermonde Matrix (and determinant), which he didn't do, but not for publishing the first paper in modern algebra, which he may have done.

1.2.4 Lagrange Interpolation

We'll now look at a much easier method for solving the Polynomial Interpolation Problem. As a by-product, we get a constructive proof of the existence of a solution to the PIP. (Here "constructive" means that we'll prove it exists by actually computing it).

Example 1.2.5. Consider the problem: *find* $p_3 \in \mathcal{P}_3$ *such that*

$$p_3(0) = 2, \quad p_3(1) = -1/2, \quad p_3(2) = 1, \quad p_3(3) = -1.$$

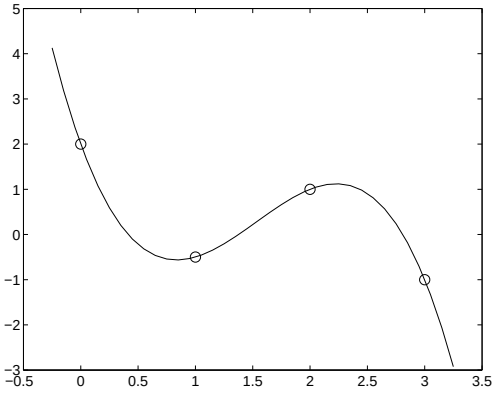


Fig. 1.1: A possible solution to Example 1.2.5

This is not so easy to solve; the following problem is easier: *find* $L_0 \in \mathcal{P}_3$ *such that*

$$L_0(0) = 1, \quad L_0(1) = 0, \quad L_0(2) = 0, \quad L_0(3) = 0.$$

Obviously, because L_0 is a cubic and has zeros at $x = 1, 2, 3$ it is of the form $L_0(x) = C(x-1)(x-2)(x-3)$. Then, choosing C so that $L_0(0) = 1$, we get

$$L_0(x) =$$

Similarly, let $L_1 \in \mathcal{P}_3$ be the cubic polynomial such that

$$L_1(0) = 0, \quad L_1(1) = 1, \quad L_1(2) = 0, \quad L_1(3) = 0,$$

then

$$L_1(x) =$$

In the same style, $L_2(x_i) = \begin{cases} 1 & i = 2 \\ 0 & i = 0, 1, 3 \end{cases}$ gives

$$L_2(x) =$$

Also, if we define

$$L_3(x_i) = \begin{cases} 1 & i = 3 \\ 0 & i = 0, 1, 2 \end{cases},$$

then clearly,

$$L_3(x) = \frac{(x-0)(x-1)(x-3)}{(3-0)(3-1)(3-2)} = \prod_{j=0, j\neq 3}^{n} \frac{(x-x_j)}{(x_3-x_j)}.$$

Plots of these polynomials are shown in Figure 1.2.

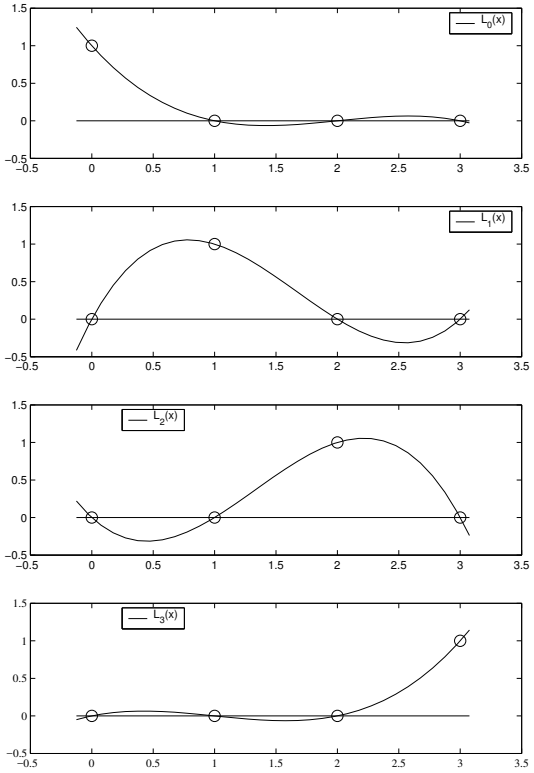


Fig. 1.2: The polynomials $L_0(x)$, $L_1(x)$, $L_2(x)$, and $L_3(x)$

Because each of L_0, L_1, L_2, and L_3 is a cubic, so too is any linear combination of them. So

$$p_3(x) = 2L_0(x) - (1/2)L_1(x) + (1)L_2(x) + (-1)L_3(x),$$

is a cubic. Furthermore

$$\begin{array}{rcccc}
p_3(0) &=& 2L_0(0) & -(1/2)L_1(0) & +(1)L_2(0) & +(-1)L_3(0) \\
&=& 2(1) & -(1/2)(0) & +(1)(0) & +(-1)(0) \\
&=& 2, & & & \\
p_3(1) &=& 2L_0(1) & -(1/2)L_1(1) & +(1)L_2(1) & +(-1)L_3(1) \\
&=& 2(0) & -(1/2)(1) & +(1)(0) & +(-1)(0) \\
&=& -1/2, & & & \\
p_3(2) &=& 2L_0(2) & -(1/2)L_1(2) & +(1)L_2(2) & +(-1)L_3(2) \\
&=& 2(0) & -(1/2)(0) & +(1)(1) & +(-1)(0) \\
&=& 1, & & & \\
p_3(3) &=& 2L_0(3) & -(1/2)L_1(3) & +(1)L_2(3) & +(-1)L_3(3) \\
&=& 2(0) & -(1/2)(0) & +(1)(0) & +(-1)(1) \\
&=& -1. & & &
\end{array}$$

Thus p_3 solves the problem in Example 1.2.5

1.2.5 The Lagrange Form

We can generalise this idea to solve any PIP using what is called *Lagrange*[2] interpolation.[3] We'll now look how to solve the general problem.

Definition 1.2.6. The **Lagrange Polynomials** associated with $x_0 < x_1 < \cdots < x_n$ is the set $\{L_i\}_{i=0}^n$ of polynomials in $\mathcal{P}_n$ such that

$$L_i(x_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}. \tag{1.4a}$$

and are given by the formula

$$L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}. \tag{1.4b}$$

Definition 1.2.7. The **Lagrange form of the Interpolating Polynomial**[a]

$$p_n(x) = \sum_{i=0}^{n} y_i L_i(x), \tag{1.5a}$$

or

$$p_n(x) = \sum_{i=0}^{n} f(x_i) L_i(x). \tag{1.5b}$$

[a] Take care not to confuse the *Lagrange Polynomials*, which are the L_i with the *Lagrange Interpolating Polynomial*, which is the p_n defined in (1.5).

Theorem 1.2.8 (Lagrange). *There exists a solution to the Polynomial Interpolation Problem and it is given by*

$$p_n(x) = \sum_{i=0}^{n} y_i L_i(x).$$

Take notes:

Example 1.2.9. [SM03, E.g. 6.1] Write down the Lagrange form of the polynomial interpolant to the function $f(x) = e^x$ at interpolation points $\{-1, 0, 1\}$.

Take notes:

Figure 1.3 shows the solution to Example 1.2.9 (top) and the difference between the function e^x and its interpolant (bottom). It would be interesting to see how this error depends on

(i) the function (and its derivatives)

(ii) the number of points used.

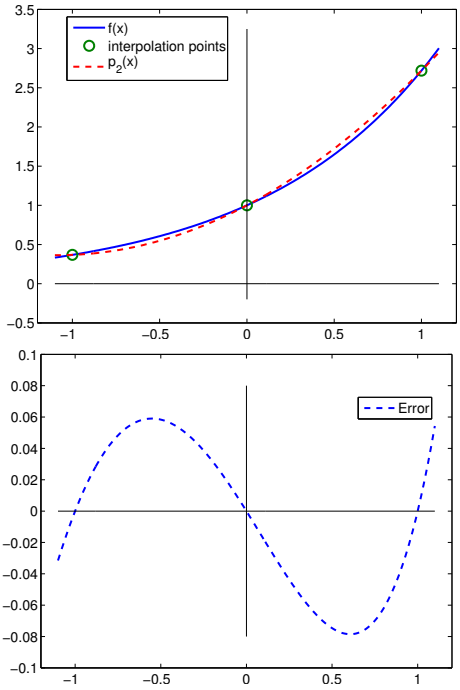


Fig. 1.3: The solution to Eg. 1.2.9 and the error

1.2.6 Some terminology

- The process of finding a polynomial, p_n, that solves the PIP is called *interpolation*.
- The solution, p_n, is called an *interpolant*.
- The difference between $f(x)$ and $p_n(x)$ is called the *interpolation error*.

[2] 

Joseph-Louis Lagrange, born 1736 in Turin, died 1813 in Paris. He made great contributions to many areas of Mathematics, including *Calculus of Variations*, which we'll see a little of at the end of the semester.
Image source: `http://jeff560.tripod.com/stamps.html`

[3] This formula was actually discovered by E. Waring in 1776; rediscovered by Euler in 1783, and published by Lagrange in 1795

1.2.7 Exercises

Exercise 1.4. Give a proof of Lemma 1.2.2 that is different from the one we did in class.

Exercise 1.5. The general form of the *Vandermonde* Matrix is

$$V_n = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}.$$

Its determinant is

$$\det(V_n) = \prod_{1 \leqslant i < j \leqslant n} (x_j - x_i). \tag{1.6}$$

Verify this for the 2×2 and 3×3 cases.

(Note that from formula (1.6) we can deduce directly that the PIP has a unique solution *if and only if* the points $x_0, x_1, \ldots, x_n$ are all distinct.)

Aside: Here is a hint for proving that in general

$$\det(V_n) = \prod_{1 \leqslant i < j \leqslant n} (x_j - x_i).$$

First note that $\det(V_n) = \det(V_n^T)$ and now consider the determinant of

$$V_n^T = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_0 & x_1 & x_2 & \cdots & x_n \\ x_0^2 & x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & & \vdots \\ x_0^n & x_1^n & x_2^n & \cdots & x_n^n \end{pmatrix}.$$

Using elementary row operations (which preserve the determinant), add to each row, the row above it multiplied by $-x_0$, to show that we need to compute the determinant of

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & x_1 - x_0 & x_2 - x_0 & \cdots & x_n - x_0 \\ 0 & x_1^2 - x_1 x_0 & x_2^2 - x_2 x_0 & \cdots & x_n^2 - x_n x_0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & x_1^n - x_1^{n-1} x_0 & x_2^n - x_2^{n-1} x_0 & \cdots & x_n^n - x_n^{n-1} x_0 \end{pmatrix}.$$

Now try to continue by induction...

Exercise 1.6. $\star$ (MA378 Semester 2 exam, 2015/2016) Let p_3 be the polynomial that interpolates $f(x) = \sin(x/2)$ at the points $x_0 = 0$, $x_1 = 1/3$, $x_2 = 2/3$ and $\mathbf{x_3 = 1}$. Use Cauchy's Theorem to give a bound for the error at $x = 1/2$. (You don't have to give a formula for p_3).[4].

Exercise 1.7. Find the polynomial p_1 that interpolates the function $f(x) = x^3$ at the points $x_0 = 0$ and $x_1 = a$. Find the point $\sigma \in [0, a]$ that maximises $|f(x) - p_1(x)|$, and hence compute $\max_{0 \leqslant x \leqslant a} |f(x) - p_1(x)|$.
Source: Chapter 6 of Süli and Mayers.

[4] An earlier version of this stated, incorrectly, that $x_4 = 1$

Exercise 1.8. $\star$ Show that

$$\sum_{i=0}^{n} L_i(x) = 1 \quad \text{for all } x.$$

Exercise 1.9. Write down the Lagrange Form of p_2, the polynomial of degree 2 that interpolates the points $(0, 3)$, $(1, 2)$ and $(2, 4)$.
Source: Chapter 2 of Stoer and Bulirsch.

Exercise 1.10. Show that all the following represent the same polynomial (usually called the "Chebyshev Polynomial of Degree 3"), $T_3(x) = 4x^3 - 3x$.

(a) Horner form: $\big((4x + 0)x - 3\big)x + 0$.

(b) Lagrange form: $\displaystyle\sum_{k=0}^{3} \left(\prod_{j=0, j \neq k}^{3} \frac{x - x_j}{x_k - x_j} \right) (-1)^{k+1}$, where $x_0 = -1, x_1 = -1/2, x_2 = 1/2, x_3 = 1$.

(c) Recurrence relation: $T_0 = 1$, $T_1 = x$, and $T_n = 2xT_{n-1} - T_{n-2}$ for $n = 2, 3, \ldots$

(d) Trigonometric form: $T_3(x) = \cos\big(3\cos^{-1}(x)\big)$.

1.3 Polynomial Interpolation Errors

1.3.1 Introduction

In Example 1.2.9, we wrote down the polynomial of degree $n = 2$ interpolating $f(x) = e^x$ at $x_0 = -1$, $x_1 = 0$ and $x_2 = 1$.

We now want to investigate how, in general, error in polynomial interpolation depends on

(i) the function (and its derivatives)

(ii) the number of points used (or, equivalently, degree of the polynomial used).

The main ingredient we need is the following theorem.

Theorem 1.3.1 (Rolle's[5] Theorem)**.** *Let g be a function that is continuous and differentiable on the interval $[a, b]$. If $g(a) = g(b)$, then there is at least one point c in (a, b) where $g'(c) = 0$.*

Our "proof" is by picture:[6]
Take notes:

1.3.2 Error estimate for n=0

The simplest case is when $n = 0$, so the interpolant is a constant, i.e., it is p_0 interpolating a function f at a point x_0. Here is one way we can deduce the *interpolation error.*[7]
Take notes:

It is important to understand what this formula is telling us:
Take notes:

1.3.3 Error estimates for $n \geqslant 1$

We will use this Rolle's Theorem to prove the most important theorem of NA2; it is used repeatedly through-out the course. It's often called the *Polynomial Interpolation Error Theorem*, but we'll call it *Cauchy's Theorem* for short.

First, we need to define an important polynomial.

Definition 1.3.2. The **Nodal Polynomial** π_{n+1} associated with the interpolation points that $a = x_0 < x_1 < \cdots < x_n = b$ is

$$\pi_{n+1}(x) = (x - x_0)(x - x_1)\dots(x - x_n) = \prod_{i=0}^{n}(x - x_i).$$

Theorem 1.3.3 (Cauchy, 1840)**.** *Suppose that $n \geqslant 0$ and f is a real-valued function that is continuous and defined on $[a, b]$, such that the derivative of f of order $n + 1$ exists and is continuous on $[a, b]$. The p_n be the polynomial of degree n that interpolates f at the $n + 1$ points $a = x_0 < x_1 < \cdots < x_n = b$. Then, for any $x \in [a, b]$ there is a $\tau \in (a, b)$ such that*

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\tau)}{(n+1)!}\pi_{n+1}(x). \qquad (1.7)$$

Proof: Take notes:

[5] Michel Rolle, Born 1652 in Ambert, Basse-Auvergne (France), died 1719 in Paris. This is his most famous result.

[6] One can easily deduce Rolle's Theorem from the Mean Value Theorem (MVT). But since the standard proof of the MVT uses Rolle's Theorem, that would be cheating

[7] There is an easier way, involving the MVT, but the approach given here is easier to generalise

Example 1.3.4. Recall Theorem 1.2.9, where we wrote down the Lagrange form of the polynomial, p_2, that interpolates $f(x) = e^x$ at the points $\{-1, 0, 1\}$. Give a formula for $e^x - p_2(x)$.

Take notes:

Usually (and as in the above example), we can't calculate $f(x) - p_n(x)$ exactly from Formula (1.7), because we have no way of finding τ. However, we are typically not so interested in what the error is at some given point, but what is the maximum error over the whole interval $[x_0, x_n]$. That is given by:

Corollary 1.3.5. *Define*

$$M_{n+1} = \max_{x_0 \leqslant \sigma \leqslant x_n} |f^{(n+1)}(\sigma)|.$$

Then

$$|f(x) - p_n(x)| \leqslant \frac{M_{n+1}}{(n+1)!}|\pi_{n+1}(x)|. \qquad (1.8)$$

Example 1.3.6. Let p_1 be the polynomial of degree 1 that interpolates a function f at distinct points x_0 and x_1. Letting $h = x_1 - x_0$, show that

$$\max_{x_0 \leqslant x \leqslant x_1} |f(x) - p_1(x)| \leqslant \frac{1}{8}h^2 M_2.$$

Take notes:

1.3.4 Exercise

Exercise 1.11. Let p_2 be the polynomial of degree 2 that interpolates a function f at the points x_0, x_1 and x_2. If $x_1 - x_0 = x_2 - x_1 = h$, show that

$$\max_{x_0 \leqslant x \leqslant x_2} |f(x) - p_2(x)| \leqslant \frac{1}{6}\frac{2}{3\sqrt{3}}h^3 M_3 = \frac{1}{9\sqrt{3}}h^3 M_3.$$

Hint: simplify the calculations by taking $t = x - x_1$, *writing* $(x - x_0)(x - x_1)(x - x_2)$ *in terms of* h *and* t.

1.4 Computing the polynomial interpolant

1.4.1 Synthetic Division

The Lagrange form of the polynomial interpolant has great theoretical importance. It is also useful for $n = 0, 1, 2$, say. It practice, for larger n, it is easy to write down the Lagrange form of the polynomial interpolant, but it is tedious (and computationally expensive) to work with. Evaluating each of the

$$L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

at a given value of x requires $2n - 1$ multiplications. So therefore computing

$$p_n(x) = \sum_{i=0}^{n} y_i L_i(x) = \sum_{i=0}^{n} \left(y_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \right),$$

for a given value of x requires $2n^2 + 2n$ multiplications. On the other hand, suppose we have p_n in the standard form

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n.$$

At first glance it seems that this takes $(n^2 - n)/2$ multiplications, which is only a little better. However there is a faster way of doing this called *synthetic division*. As an example we'll consider how this works for the case $n = 3$. For the general case, please have a look at [S96, Lecture 19].

Take notes:

In summary,

- The Lagrange form is easy to find and difficult to evaluate.
- The standard form (i.e., $p_n = a_0 + a_1 x + \cdots + a_n x^N$), is hard to find, but easy to evaluate.
- So we want to find a form for the polynomial is easy to construct *and* efficient to evaluate.

There are plenty of possibilities – too many to mention – and we'll consider one of them: the *Newton Form*.

1.4.2 The Newton Form of the Interpolant

This section is just for your information: it will not be covered in class or be part of any assessment.

These notes are based on §2.1.3 of [SB92] and on [S96, Lecture 19].

Suppose we were to write the interpolating polynomial p_n as

$$\begin{aligned} p_n(x) = a_0 &+ a_1(x - x_0) \\ &+ a_2(x - x_0)(x - x_1) \\ &+ a_3(x - x_0)(x - x_1)(x - x_2) \\ &+ \cdots + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}). \end{aligned}$$

Then it could actually be evaluated as

$$\begin{aligned} p_n(x) = a_0 &+ \\ &(x - x_0)\Big(a_1 + (x - x_1)\big(a_2 + (x - x_2)(a_3 + \\ &\qquad (\cdots + a_n(x - x_{n-1}))\cdots\big)\Big). \end{aligned} \quad (1.9)$$

This is called the *Newton Form of the Interpolating Polynomial*. [8]

Let π_k be the nodal polynomial $\pi_k = \prod_{i=0}^{k}(x - x_i)$. Then the Newton form is

$$p_n(x) = a_0 + a_1\pi_0 + a_2\pi_1 + a_3\pi_2 + \cdots + a_n\pi_{n-1}.$$

So how do we find the coefficients? Let

$$F[x_i] = f(x_i),$$

$$F[x_i, x_{i+1}] = \frac{F[x_{i+1}] - F[x_i]}{x_{i+1} - x_i}$$

$$\vdots$$

$$F[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{F[x_{i+1}, x_{i+2}, \ldots x_{i+k}] - F[x_i, x_{i+1}, \ldots x_{i+k-1}]}{x_{i+k} - x_i}$$

Then

$$a_0 = f[x_0], \qquad a_1 = F[x_0, x_1], \quad \ldots$$

$$a_n = F[x_0, x_1, \ldots x_n].$$

Or, if you prefer:

$$p_n(x) = F[x_0] + \sum_{k=1}^{n} F[x_0, x_1, \ldots, x_k]\pi_k(x). \quad (1.10)$$

[8] Sir Issac Newton Isaac Newton, 1643 - 1727, England. Easily one of the greatest scientist of all time. This image is of a Polish stamp commemorating the scientific achievements. Source: `http://jeff560.tripod.com/stamps.html`

It is not hard to show that the formula given in (1.10) yields the interpolating polynomial. The argument is inductive. Before we do that, we will demonstrate that it works for a particular example.

Example 1.4.1. Write down the Newton form of the polynomial p_2 that interpolates e^x at $x_0 = 0$, $x_1 = 1$ and $x_2 = 2$.

Solution: (For the Newton form)

	$F[x_i]$	$F[x_i x_{i+1}]$	$F[x_i x_{i+1} x_{i+2}]$
x_0	1		
		$e-1$	
x_1	e		$\frac{1}{2}(e^2-2e+1)$
		e^2-e	
x_2	e^2		

This gives

$$\begin{aligned} p_2(x) &= F[x_0] + (x-x_0)\big(F[x_0x_1] + (x-x_1)F[x_0x_1x_2]\big), \\ &= 1 + x\big((e-1) + (x-2)\frac{1}{2}(e^2-2e+1)\big). \end{aligned}$$

Written in the form of (1.10) above, this is

$$p_2(x) = 1 + (e-1)(x) + \frac{1}{2}(e^2-2e+1)(x)(x-1).$$

One can quickly check that this

1. is a polynomial of degree n,
2. interpolates e^x at $x = 0$, $x = 1$ and $x = 2$.

It is not hard to show that the formula given in (1.10) yields the interpolating polynomial. The argument is inductive. Clearly its true for $n = 0$, because in that case the interpolant is just the constant polynomial $p_0 = F[x_o] = f(x_0)$.

Next, suppose it is true for n points. Then let

$$\begin{aligned} p_n &\text{ interpolate } f(x) \text{ at } x = x_0, x_1, \dots x_n, \\ q_{n-1}(x) &\text{ interpolate } f(x) \text{ at } x = x_0, x_1, \dots x_{n-1}, \\ r_{n-1}(x) &\text{ interpolate } f(x) \text{ at } x = x_1, x_2 \dots x_n. \end{aligned}$$

We must convince ourselves that

$$p_n(x) = q_{n-1}(x) + \frac{x-x_0}{x_n-x_0}\big(r_{n-1}(x) - q_{n-1}(x)\big),$$

First, since both q_{n-1} and r_{n-1} are polynomials of degree $n-1$, it follows that the expression on the right is indeed a polynomial of degree n.

Next, check that it is the interpolant of f at x_0, x_1, $\dots$, x_n. For the first point, x_0:

$$\begin{aligned} p_n(x_0) = q_{n-1}(x_0) + \frac{x_0-x_0}{x_n-x_0}\big(r_{n-1}(x_0) - q_{n-1}(x_0)\big) \\ = q_{n-1}(x_0) = f(x_0). \end{aligned}$$

For the last point, x_n:

$$\begin{aligned} p_n(x_n) = q_{n-1}(x_n) + \frac{x_n-x_0}{x_n-x_0}\big(r_{n-1}(x_n) - q_{n-1}(x_n)\big) \\ = q_{n-1}(x_n) + \big(r_{n-1}(x_n) - q_{n-1}(x_n)\big) \\ = r_{n-1}(x_n) = f(x_n). \end{aligned}$$

And for every other point, x_i, $i = 1, \dots, n-1$,

$$\begin{aligned} p_n(x_i) = q_{n-1}(x_i) + \frac{x_i-x_0}{x_n-x_0}\big(r_{n-1}(x_i) - q_{n-1}(x_i)\big) \\ = f(x_i) + \frac{x_i-x_0}{x_n-x_0}\big(f(x_i) - f(x_i)\big) = f(x_i). \end{aligned}$$

Finally, we need to compare the coefficient of x^n on the left- and right-hand sides of this equations.

The coefficient of x^{n-1} in q_{n-1} is $F[x_0, x_1, \dots, x_{n-1}]$. The coefficient of x^{n-1} in r_{n-1} is $F[x_1, x_2, \dots, x_n]$. So the coefficient of x^n in p_n is

$$\frac{1}{x_n-x_0}\big(F[x_1, x_2, \dots, x_n] - F[x_0, x_1, \dots, x_{n-1}]\big),$$

as required.

1.4.3 Exercise

Exercise 1.12. Do Exercise 6.3 from Süli and Mayers, *An Introduction to Numerical Analysis*.

1.5 Hermite Interpolation

1.5.1 The idea

Hermite[9] interpolation is a variant on the standard Polynomial Interpolation Problem: we seek a polynomial that not only agrees with a given function f at the interpolation points, but its first derivative also matches f' at those points. We are not that interested in this problem for its own sake, but the idea recurs again in the sections in piecewise polynomial interpolation and Gaussian quadrature.

Formally, the problem is

The Hermite Polynomial Interpolation Problem (HPIP)

Given a set of interpolation points $x_0 < x_1 < \cdots < x_n$ *and a continuous, differentiable function* f*, find* $p_{2n+1} \in \mathcal{P}_{2n+1}$ *such that*

$$p_{2n+1}(x_i) = f(x_i) \quad \textit{and} \quad p'_{2n+1}(x_i) = f'(x_i).$$

It is not hard to see that if there is a solution to this problem, then it is unique:

Take notes:

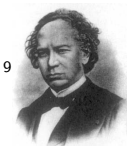
1.5.2 Constructing Hermite Interpolants

It is possible to solve this problem using an extension of the Lagrange Polynomial approach of §1.2.4. Given the Lagrange Polynomials L_i as defined in (1.4) let

$$H_i(x) = [L_i(x)]^2(1 - 2L_i'(x_i)(x - x_i)),$$

$$K_i(x) = [L_i(x)]^2(x - x_i).$$

Examples of two such functions are shown in Figure 1.1.



[9] Charles Hermite, France, 1822–1901. A part from this form of interpolation, his contributions to Mathematics included the first proof that e is transcendental. His methods were later used to show that π is transcendental.

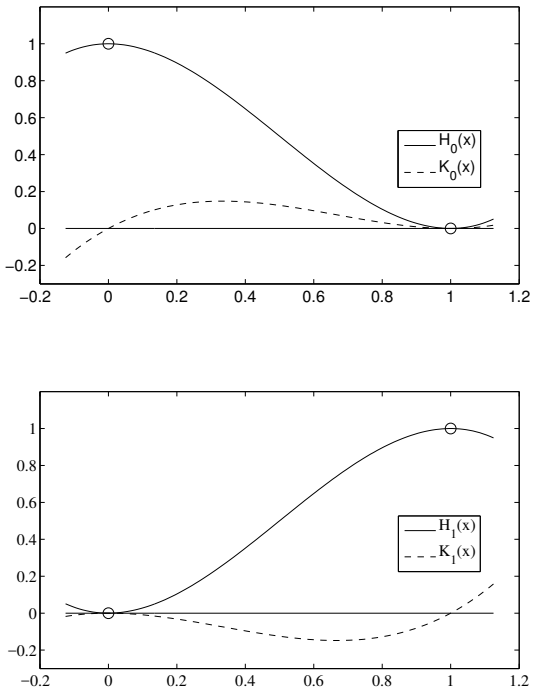


Fig. 1.1: Hermite bases functions H_0, K_0 (top) and H_1, K_1 (bottom) for $n = 1$, $x_0 = 0$ and $x_1 = 1$

We can show that, for $i, k = 0, 1, \ldots n$,

$$H_i(x_k) = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases} \qquad H_i'(x_k) = 0 \ \forall k,$$

$$K_i(x_k) = 0, \qquad K_i'(x_k) = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases}$$

We'll show this for H and H', and leave the corresponding results for K and K' to Exercise 1.16.

Take notes:

Armed with these identities, we can now show that the solution to the HPIP exists and is given by

$$p_{2n+1}(x) = \sum_{i=0}^{n} \big(f(x_i)H_i(x) + f'(x_i)K_i(x)\big).$$

(Again, see Exercise 1.16 for details).

Example 1.5.1. Find the polynomial of degree 3 that interpolates $\exp(x^2)$, and its first derivative, at $x_0 = 0$ and $x_1 = 1$. (See Figure 1.2).

Take notes:

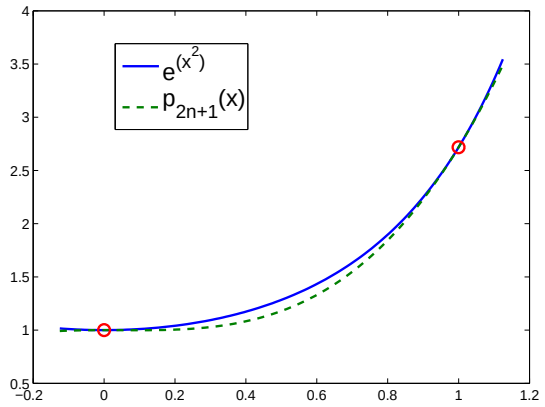


Fig. 1.2: The Hermite interpolant to $e^{(x^2)}$ at $x = 0$ and $x = 1$

Theorem 1.5.2. *Let be a real-valued function that is continuous and defined on* $[a, b]$*, such that the derivatives of* f *of order* $2n + 2$ *exist and are continuous on* $[a, b]$*. Let* p_{2n+1} *be the Hermite interpolant to* f*. Then, for any* $x \in [a, b]$ *there is an* $\tau \in (a, b)$ *such that*

$$f(x) - p_{2n+1}(x) = \frac{f^{(2n+2)}(\tau)}{(2n+2)!}[\pi_{n+1}(x)]^2.$$

We won't do a proof of this in class. However, later in this course we'll be interested in the particular example of finding p_3 the cubic Hermite Polynomial Interpolant to a function f at the points x_0 and x_1. Also, see Exercise 1.13.

1.5.3 Exercises

Exercise 1.13. $\star$ For *just* the case $n = 1$, state and prove an appropriate version of Theorem 1.5.2 (i.e., error in the Hermite interpolant). Use this to find a bound for $\|f - p_3\|_{[x_0,x_1]}$ in terms of f and $h = x_1 - x_0$. (Here $\|g\|_{[x_0,x_1]}$ is short-hand for $\max_{x_0 \leqslant x \leqslant x_1} |g(x)|$.)

Exercise 1.14. Let $n = 2$ and $x_0 = -1$, $x_0 = 1$ and $x_1 = 1$. Write out the formulae for H_i and K_i for $i = 0, 1, 2$ and give a rough sketch of each of these six functions that shows the value of the function and its derivative at the three interpolation points.

Exercise 1.15. Do Exercise 6.6 from from Süli and Mayers, *An Introduction to Numerical Analysis*.

Exercise 1.16. $\star$ Let $L_0, L_1, \ldots, L_n$ be the usual Lagrange polynomials for the set of interpolation points $\{x_0, x_1, \ldots, x_n\}$. Now define

$$H_i(x) = [L_i(x)]^2(1 - 2L_i'(x_i)(x - x_i)),$$

and

$$K_i(x) = [L_i(x)]^2(x - x_i).$$

We saw in class that, for $i, k = 0, 1, \ldots n$,

$$H_i(x_k) = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases} \qquad H_i'(x_k) = 0.$$

Show that, for $i, k = 0, 1, \ldots n$,

$$K_i(x_k) = 0, \qquad K_i'(x_k) = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases}$$

Conclude that the solution to the Hermite Polynomial Interpolation Problem is

$$p_{2n+1}(x) = \sum_{i=0}^{n} \big(f(x_i)H_i(x) + f'(x_i)K_i(x)\big).$$

Exercise 1.17. Write down that formula for q_3, the *Hermite* polynomial that interpolates $f(x) = \sin(x/2)$, and its derivative, at the points $x_0 = 0$ and $x_1 = 1$. Give an upper bound for $|f(1/2) - q_3(1/2)|$. How does this compare with the bound in Exercise 1.6?

Exercise 1.18. (This exercise is based on Exer 6.5 from Süli and Mayers' *Introduction to Numerical Analysis*). Consider the following problem.

> Take $n+1$ distinct interpolation points $x_0 < x_1 < \cdots < x_n$. Let p_{2n+1} be the polynomial of degree $2n + 1$ with the property that
>
> $$p_{2n+1}(x_i) = f(x_i),$$
>
> and
>
> $$p_{2n+1}''(x_i) = f''(x_i).$$

In general this problem does *not* have a unique problem.

(i) Explain briefly but carefully why the arguments, based on Rolle's Theorem, used to prove **uniqueness** of solutions to the HPIP, will not work here.

(ii) Show that there is no $p_5(x)$ that solves this problem when

- $x_0 = -1$, $x_1 = 0$, $x_2 = 1$.
- $f(-1) = 1$, $f(0) = 0$, $f(1) = 1$.
- $f''(-1) = 0$, $f''(0) = 0$, $f''(1) = 0$.

1.6 Wrap-up

1.6.1 Convergence & Runge's Example

The celebrated Weierstrass approximation theorem states that, given f and a positive number ε, there is a polynomial p such that

$$\max_{x\in[a,b]} |f(x) - p(x)| := \|f - p\|_\infty \leqslant \varepsilon.$$

Now suppose that f is a continuous function on $[a, b]$ and that $\{p_n\}_{n=0}^\infty$ is a sequence of polynomials that interpolate f at $n+1$ **equally spaced** points. One might be inclined to believe that

$$\lim_{n\to\infty} \|f - p_n\|_\infty = 0.$$

Equivalently, recalling (1.8):

$$|f(x) - p_n(x)| \leqslant \frac{M_{n+1}}{(n+1)!}|\pi_{n+1}(x)|,$$

one might expect that

$$\lim_{n\to\infty} \max_{x\in[a,b]} \frac{M_{n+1}}{(n+1)!}|\pi_{n+1}(x)| = 0.$$

In order words, we might think that, in order to find an interpolating polynomial that is as accurate as we would like, we just need to choose large enough n.

And some times it might work. For example, suppose that $a = -5$, $b = 5$, and $f(x) = e^{\sin(x/2)}$. In Table 1.1 the errors for successive interpolants are shown. A few of these are shown in Figure 1.3.

Table 1.1: Errors in polynomial interpolants to $e^{\sin(x/2)}$ on $[-5, 5]$

n	$\|f - p_n\|_\infty$
2	1.27e-00
4	2.94e-01
6	8.39e-02
8	5.75e-02
16	1.07e-03

However, there is a famous example of a simple function that cannot be successfully interpolated in this manner. This is *Runge's Example*:

$$f(x) = \frac{1}{1+x^2} \qquad \text{on } [-5, 5].$$

The errors are shown in Table 1.2. Notice that they *increase* with n. We don't have the scope to go into *why* this is the case (but see the notes from class on a heuristic explanation).

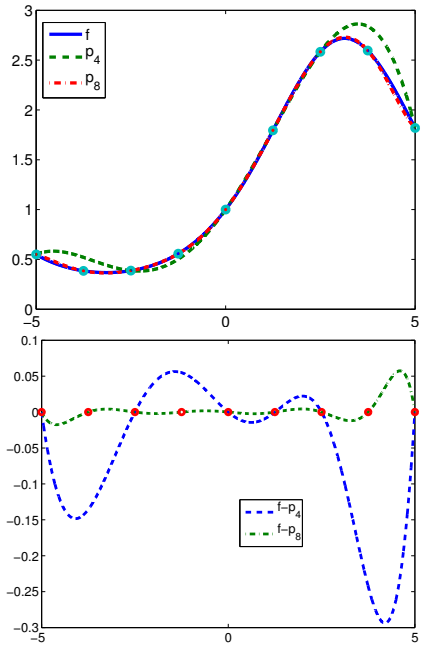


Fig. 1.3: Polynomial interpolants to $e^{\sin(x/2)}$ on $[-5, 5]$, and their errors (bottom)

Table 1.2: Errors in interpolants to $1/(1+x^2)$ on $[-5, 5]$

n	$\|f - p_n\|$
2	0.65
4	0.44
6	0.62
8	1.05
16	14.39
20	59.66
22	122.91
24	257.21

1.7 Where to from here?

So now it looks like polynomial interpolation is bad, at least on equidistant points. However, our experience in Lab 1 might lead us to be more optimistic: we were able to find a set of points that made the approximation as good we wanted (until round-off error dominated).

Unfortunately, just because we have a good set of points for interpolating one particular function, it does not follow that that set is good for every continuous function: this is *Faber's Theorem*[10]. This has often led numerical analysts to abandon the idea of interpolation by high-order polynomials completely.

However, there is a set of points that are useful, if f is smooth enough: the Chebyshev points of Lab 1. If you are interested, there please look at the essay *Inverse Yogiisms* by Lloyd N. (Nick) Trefethen, Notices AMS, Dec. 2016.[11] To investigate this numerically in MATLAB, try

[10] Georg Faber (18771966)

[11] `http://people.maths.ox.ac.uk/trefethen/essays.html` or `http://www.ams.org/publications/journals/notices/`

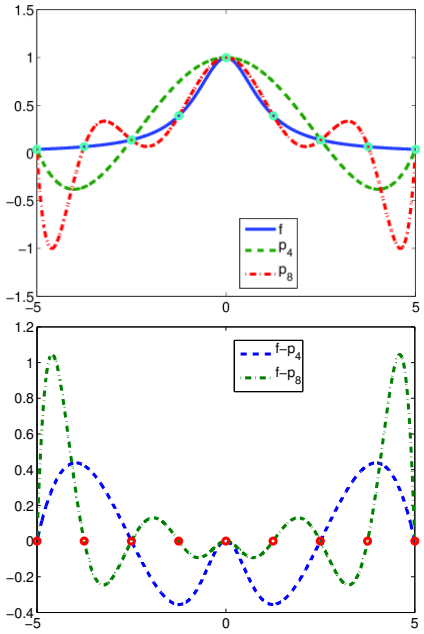


Fig. 1.4: Polynomial interpolants to $1/(1+x^2)$ on $[-5,5]$

exploring the Chebfun toolbox.

The approach we will take is different. In (1.3.6) we saw that

$$\max_{x_0 \leqslant x \leqslant x_1} |f(x) - p_1(x)| \leqslant \frac{1}{8} h^2 M_2.$$

So, assuming M_2 is bounded (which is reasonable), we can make p_1 as close to f as we would like by taking a small enough interval $[x_0, x_1]$. The next section of this module is devoted to seeing how this can be used in theory and practice.

Chapter 2

Piecewise polynomial interpolation

In Section 1.6.1, and in Lab 1, we learned that it is not a good idea to interpolate functions by a high-order polynomials at equally spaced points. However, it transpires that it is possible to obtain a very good approximations using a very simple method. The trick is to use a *spline*: a *piecewise polynomial interpolating function*.

We'll consider three important example of splines:

1. *linear splines*
2. *(natural) cubic splines*.
3. *Hermite* piecewise cubics.

In this section, we always have N equally spaced points: let $h = (b-a)/N$, then

$$a = x_0, \quad b = x_N \quad \text{and} \quad x_i = x_0 + ih.$$

Often these are referred to as *knots points* (or simply as *knots*), and denote the set of knot points by $\omega^N := \{x_i\}_{i=0}^N$.

For more details about splines, have a look at [SM03, Chap. 11], and [S98, Lectures 10 and 11].

2.1 Linear Interpolating Splines

We first study the *piecewise linear interpolant*, also called a *linear spline*. We will see that they have important properties, including

(a) they are easy to construct and analyse;

(b) the bound on the error decreases as the number of interpolation points increases;

(c) the error we get using a linear spline is no more than twice the error using the best possible (piecewise linear) approximation; amd

(d) of all the interpolants to f at a given set of points, the linear spline is the one with the smallest 1st derivative.

2.1.1 Construction

Definition 2.1.1. Let f be a function that is continuous on $[a, b]$. The *linear spline interpolant* to f is the continuous function l such that

(i) $l(x_i) = f(x_i)$ for each $i = 0, 1, \ldots, N$,

(ii) l is a linear function l_i on each interval $[x_{i-1}, x_i]$.

It is easy to write down a formula for the l_i, based on Lagrange polynomials. Set $h = (b-a)/N$. Then, for $x \in [x_{i-1}, x_i]$

$$l_i(x) = f(x_{i-1})\frac{x_i - x}{h} + f(x_i)\frac{x - x_{i-1}}{h}. \qquad (2.1)$$

Example 2.1.2. Write down the linear spline interpolant to $f(x) = e^x$ at the knot points $\{-1, 0, 1\}$.

Take notes:

2.1.2 Analysis

We know that if p_N is the polynomial of degree N that interpolates f at N equally spaced points, it does **not** follow that $p_N \to f$ as $n \to \infty$. But as we will see, the piecewise linear interpolant to f converges to f, albeit slowly.

This is verified in the following theorem, which is a direct consequence of Theorem 1.3.3.

Theorem 2.1.3. *Suppose that* f, f' *and* f'' *are all continuous and defined on the interval* $[a, b]$. *Let* l *be the linear spline interpolant to* f *on the* $N+1$ *equally spaced points* $a = x_0 < x_1 \cdots < x_N = b$ *with* $h = x_i - x_{i-1} = (b-a)/N$. *Then*

$$\|f - l\|_\infty \leqslant \frac{h^2}{8}\|f''\|_\infty,$$

(Here, as usual, $\|g\|_\infty$ is defined as $\max_{a \leqslant x \leqslant b} |g(x)|$.)

Take notes:

It now follows directly from Theorem 2.1.3 that

$$\lim_{n\to\infty} \|f - l\|_\infty = 0.$$

Example 2.1.4. Figure 2.1 shows linear spline interpolations of Runge's example:

$$f(x) = \frac{1}{1+x^2} \text{ on } [-5, 5].$$

These diagrams appear to support our assertion that the error tends to zero as $n \to \infty$.

Example 2.1.5. Suppose you are interpolating $f(x) = e^x$ on N equally spaced intervals between $x_0 = -1$ and $x_N = 1$. What value of N would you have to take to ensure that the maximum error is less than 10^{-2}?

Take notes:

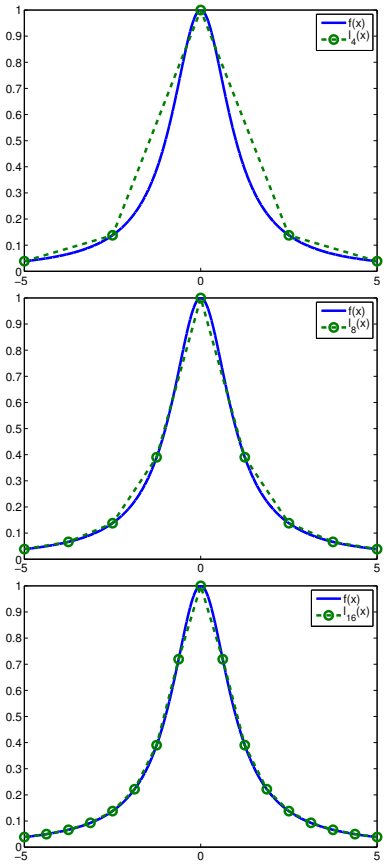


Fig. 2.1: Linear spline interpolants to $1/(1+x^2)$ on $[-5, 5]$. Contrast with Figure 1.4

2.1.3 Best approximation

For the next part of the analysis it will help to think of piecewise linear interpolation as an *operator*. Then we can compare the linear spline to all the other piecewise linear approximations.

First, observe that one can define an infinite number of piecewise linear functions on a given set of knot points ω^N. We'll call the set of these functions $\mathcal{L}$.

Definition 2.1.6. For a fixed set of knot points ω^N, let L be the operator that maps the continuous function f to its linear spline interpolant $l \in \mathcal{L}$.

Now suppose that $g \in \mathcal{L}$. Then $L(g) = g$. That is L is a *projection*: $L(L(f)) = L(f)$.

It is not hard to see that one could find a different function $\hat{l} \in \mathcal{L}$ that is a better approximation of f in sense that

$$\max_{x_0 \leqslant x \leqslant x_n} |f(x) - \hat{l}(x)| < \max_{x_0 \leqslant x \leqslant x_n} |f(x) - l(x)|.$$

However, l is very easy to find, and the associated error is no worse than twice $\|f(x) - \hat{l}(x)\|_\infty$.

Theorem 2.1.7 (Stewart's "Afternotes goes to grad school",

Lecture 10). *Let* $l = L(f)$. *For all* $\hat{l} \in \mathcal{L}$,

$$\|f - l\|_\infty \leqslant 2\|f - \hat{l}\|_\infty.$$

(That L is a projection is key to the proof.)

Take notes:

2.1.4 Minimum Energy

The final interesting property of l that we will study is called the *minimum energy property*.

Definition 2.1.8. Let u be a function that is continuous and defined on the interval $[a, b]$ except, maybe, at the (countable set) ω^N of knot points[1] Then the 2-norm of u is

$$\|u\|_{2,[a,b]} := \Big(\int_a^b u^2(x)dx\Big)^{1/2}.$$

Usually we just write this as $\|u\|_2$.

Let H^1 be the set of all functions u that are continuous on $[a, b]$ and have $\|u'\|_2 < \infty$. Note that $l' \in H^1$, even though we have not properly defined l' at the mesh points ω^N.

Theorem 2.1.9 (Süli and Mayers, Thm. 11.2)**.** *Let* w *be* any *function in* H^1 *that interpolates the function* f *at the points in* ω^N. *Let* l *we the linear spline interpolant of* f. *Then*

$$\|l'\|_2 \leqslant \|w'\|_2.$$

Take notes:

[1] More precisely, we should say "everywhere, except on a set of measure zero". What that means is very important to the areas of measure theory and functional analysis. However, since some of you are not taking those courses until next year, we'll air-brush over the exact definition.

2.1.5 Exercises

Exercise 2.1. Page 28 of the Department of Education's old Mathematics Tables ("The *Log Tables*") reports that $\ln(1) = 0$, $\ln(1.5) = 0.4055$ and $\ln(2) = 0.6931$.

(i) Write down the linear spline l that interpolates $f(x) = \ln(x)$ at the points $x_0 = 1$, $x_1 = 1.5$ and $x_2 = 2$.

(ii) Use this to estimate $\ln(x)$ at $x = 1.2$. How does this compare to the value in the tables? (0.1823)

(iii) Give an estimate for the maximum error:

$$\max_{1 \leqslant x \leqslant 2} |f(x) - l(x)|.$$

(iv) What value of n would you choose to ensure that $|f(x) - l(x)| \leqslant 0.001$ for all $x \in [1, 2]$.

Exercise 2.2. As an alternative to (2.1), one can define the linear spline interpolant to a function is as a linear combination of a set of piecewise linear basis functions $\{\psi_i\}_{i=0}^N$:

$$\psi_i(x_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

(i) Write down a formula for the $\psi_i(x)$;

(ii) derive a formula for $l(x)$ in terms of the ψ_i.

This exercise is useful: we'll be using these basis functions (called "hat" functions) in the final section of the course.

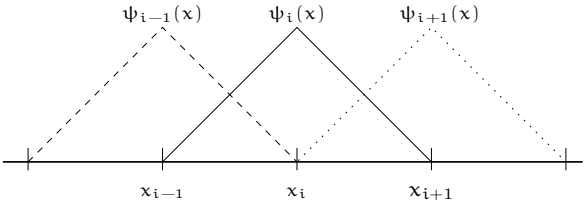


Fig. 2.2: Some hat functions

2.2 Cubic Splines

The Cubic Spline[2] is perhaps the most useful and popular interpolating function used in numerical analysis, automotive and aeronautical engineering, computer and film animation, digital photography, financial modelling, and as many other areas as there are human endeavours where continuous processes are modelled by discrete ones.

One could argue that the most fundamental shortcoming of the linear spline interpolant to f (see, for example, Figure 2.1) is that they are not "smooth": that is, although $l_i(x_i) = l_{i+1}(x_i)$, it is **not** generally the case that $l_i'(x_i) = l_{i+1}'(x_i)$. Also, the interpolating function cannot capture the "curvature" of f. (If you think about this last statement, you'll see that it can be expressed as $l''(x) = 0$ for all $x \in [a, b]$.)

Cubic splines try to balance the simplicity of the linear spline approach — by using a low-order polynomial on each interval — with the desire for curvature and more smoothness – by using cubics, and by forcing adjacent ones, and their first and second derivatives, to agree at knot points.

Definition 2.2.1. Let f be a function that is continuous on $[a, b]$. The *cubic spline interpolant* to f is the continuous function S such that

(i) for $i = 1, \ldots, N$, on each interval $[x_{i-1}, x_i]$ let $S(x) = s_i(x)$, where each of the s_i is a cubic polynomial.

(ii) $s_i(x_{i-1}) = f(x_{i-1})$ for $i = 1, \ldots, N$,

(iii) $s_i(x_i) = f(x_i)$ for $i = 1, \ldots, N$,

(iv) $s_i'(x_i) = s_{i+1}'(x_i)$ for $i = 1, \ldots, N-1$,

(v) $s_i''(x_i) = s_{i+1}''(x_i)$ for $i = 1, \ldots, N-1$.

So, we have defined the cubic spline S as a function that interpolates f at $N+1$ points, has continuous first and second derivatives on $[x_0, x_N]$ and is a cubic polynomial on each of the n intervals $[x_{i-1}, x_i]$. That is, it is *piecewise cubic*. We know that one can write a cubic as

$$p_3(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3.$$

Thus it takes 4 terms to uniquely define a single cubic. To uniquely determine N cubics requires $4N$ terms. They can be found by solving $4N$ (linearly independent) equations. But an inspection of (ii)–(iv) above reveals only $4N - 2$ equations.

[2] It is generally accepted that (mathematical) splines were first described by Isaac Schoenberg (1903-1990) during WW2. However their physical realisation had been used in the ship-building and aircraft industries prior to that.

The "missing" equations can be chosen in a number of ways:

(i) by setting $S''(x_0) = 0$ and $S''(x_N) = 0$. This is called a *natural* spline, and is the approach we'll take.

(ii) by setting $S'(x_0) = 0$ and $S'(x_N) = 0$. This is called a *clamped* spline.

(iii) set $S'(x_0) = S'(x_N)$ and $S''(x_0) = S''(x_N)$. This is the *periodic* spline and is used for interpolating, say, trigonometric functions.

(iv) only use $N - 2$ components of the spline: $s_2, \ldots, s_{N-1}$. But extend to two end ones so that $s_2(x_0) = f(x_0)$ and $s_{N-1} = f(x_N)$. This is called the *not-a-knot* condition.

In Figure 2.3 we show the natural cubic spline interpolants to $f(x) = 1/(1+x^2)$ (with $a = -5$ and $b = 5$ as usual) on $N = 4$, 8 and 16 intervals. Compared with Figure 2.1, we seem to get significantly better approximation. Moreover, the rate at which the error decreases with respect to h is much faster.

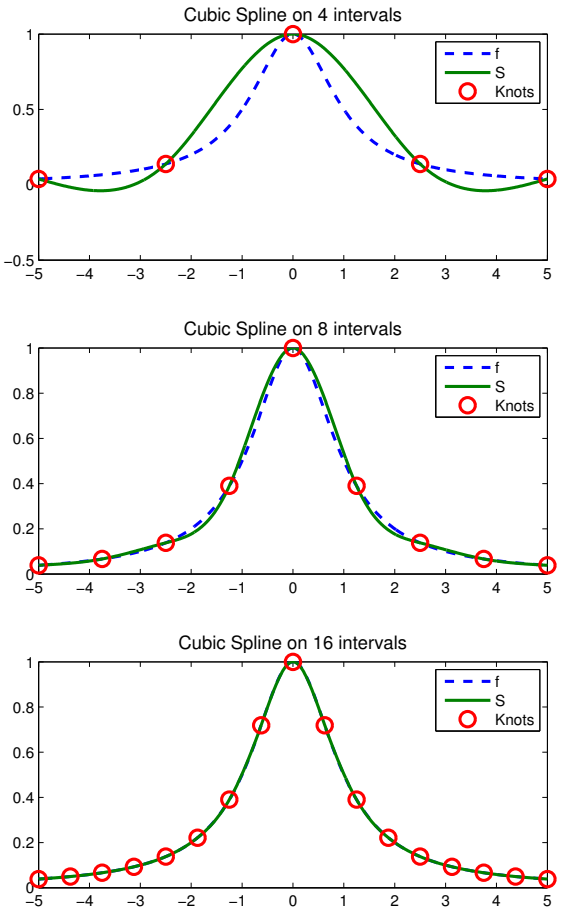


Fig. 2.3: Cubic spline interpolants to $1/(1+x^2)$. Compare with Figure 2.1

2.2.1 Constructing Cubic Splines

Some notation:

- $h = x_i - x_{i-1} = (x_N - x_0)/N$ for all i.
- $f_i := f(x_i)$.

To construct the spline, first observe that if S is piecewise cubic, then S' is piecewise quadratic, and S'' is piecewise linear.

(i) Let $\sigma_i = S''(x_i)$ for $i = 0, \ldots, N$.

Take notes:

(ii) Integrate twice:

Take notes:

We get

$$s_i(x) = \alpha_i(x - x_{i-1}) + \beta_i(x_i - x) + \frac{\sigma_{i-1}}{6h}(x_i - x)^3 + \frac{\sigma_i}{6h}(x - x_{i-1})^3, \quad (2.2)$$

for $x \in [x_{i-1}, x_i]$.

(iii) The α_i and β_i arose as the constants of integration. To find them:

Take notes:

and so, for $i = 1, 2, \ldots, N$

$$\alpha_i = \frac{f_i}{h} - \frac{h}{6}\sigma_i, \quad \beta_i = \frac{f_{i-1}}{h} - \frac{h}{6}\sigma_{i-1}. \quad (2.3)$$

Notice that this gives $\beta_i = \alpha_{i-1}$.

(iv) So, now we "just" need the equations for σ_i. Two of these come from the fact that this is a "natural" cubic spline, with $S''(x_0) = 0$ and $S''(x_N) = 0$, therefore $\sigma_0 = 0$ and $\sigma_N = 0$.

The remaining $N-1$ equations come from the fact that S' is continuous at $x_1, x_2, \ldots, x_{N-1}$. So we set that

$$s_i(x_i) = s_{i+1}(x_i).$$

After some work (see Exercise 2.3), we can show this means that the system is:

$$\sigma_0 = 0, \quad (2.4a)$$

$$\frac{1}{6}(\sigma_{i-1} + 4\sigma_i + \sigma_{i+1}) = \frac{1}{h^2}(f_{i-1} - 2f_i + f_{i+1}) \quad (2.4b)$$

for $i = 1, \ldots, N-1$,

$$\sigma_N = 0. \quad (2.4c)$$

2.2.2 A cubic spline example

Example 2.2.2. Find the natural cubic spline interpolant to f at the points $x_0 = 0$, $x_1 = 1$, $x_2 = 2$ and $x_3 = 3$ where $f_0 = 0, f_1 = 2, f_2 = 1$ and $f_3 = 0$.

[We won't do all the details in class, so here they are]

Solution:

From (2.2) we see we are looking for $S(x)=$

$$\alpha_1 x + \beta_1(1-x) + \frac{\sigma_0}{6h}(1-x)^3 + \frac{\sigma_1}{6h}x^3, x \in [0,1],$$
$$\alpha_2(x-1) + \beta_2(2-x) + \frac{\sigma_1}{6h}(2-x)^3 + \frac{\sigma_2}{6h}(x-1)^3, x \in [1,2],$$
$$\alpha_3(x-x_2) + \beta_3(3-x) + \frac{\sigma_2}{6h}(3-x)^3 + \frac{\sigma_3}{6h}(x-2)^3, x \in [2,3].$$

We first solve for the σ_i using (2.4):

$$\frac{1}{6}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \sigma_0 \\ \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{pmatrix} = \begin{pmatrix} 0 \\ -18 \\ 0 \\ 0 \end{pmatrix}$$

This gives

$$\sigma_0 = 0, \quad \sigma_1 = -\frac{24}{5}, \quad \sigma_2 = \frac{6}{5}, \quad \sigma_3 = 0.$$

Now use (2.3) to get

$$\alpha_1 = \frac{14}{5}, \quad \alpha_2 = \frac{4}{5}, \quad \alpha_3 = 0.$$

and

$$\beta_1 = 0 \quad \beta_2 = \frac{14}{5}, \quad \beta_3 = \frac{4}{5}.$$

The answer is

$$S(x) = \begin{cases} \frac{14}{5}x - \frac{4}{5}x^3, & x \in [0,1], \\ \frac{4}{5}(x-1) + \frac{14}{5}(2-x) - \\ \qquad \frac{4}{5}(2-x)^3 + \frac{1}{5}(x-1)^3, & x \in [1,2], \\ \frac{4}{5}(3-x) + \frac{1}{5}(3-x)^3. & x \in [2,3]. \end{cases}$$

This is shown in Figure 2.4.

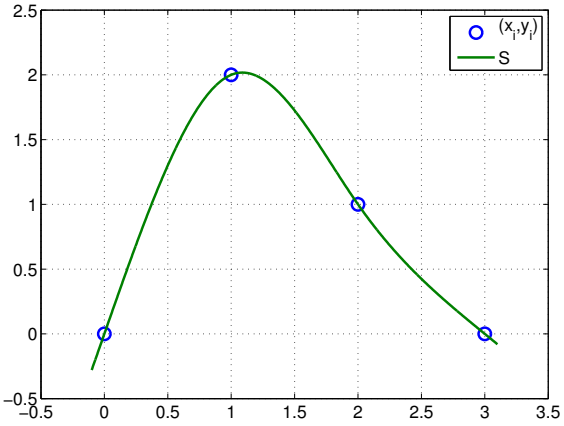


Fig. 2.4: Spline interpolant from Example 2.2.2

2.2.3 Error Estimates

We state the following error estimates without proof. You don't need to know these, or be able to prove them. But you should be able to use them if required.

Theorem 2.2.3. *If* $f \in C^4[a,b]$ *and* S *is its cubic spline interpolant on* $N+1$ *equally spaced points, then*

$$\|f - S\|_\infty \leqslant \frac{5}{384} M_4 h^4,$$
$$\|f' - S'\|_\infty \leqslant \frac{1}{24} M_4 h^3,$$
$$\|f'' - S''\|_\infty \leqslant \frac{3}{8} M_4 h^2,$$

where $M_4 = \|f^{(iv)}\|_\infty$.

Example 2.2.4. Give an upper bound on the error for the cubic spline interpolant to $f = e^x$ on the interval $[-1,1]$ with $N = 10$ mesh points.

Take notes:

Example 2.2.5. What is the smallest value of N that you must take to ensure that, if interpolating $f(x) = e^x$ on N equally sized intervals, the error is less that 10^{-8}? How does this compare with a linear spline interpolation (see Example 2.1.5)?

Take notes:

Recall Theorem 2.1.9 for linear splines. There is an analogous result for natural cubic splines.

Theorem 2.2.6. *Let* u *be any function that interpolates* f *at* ω^N*, and is such that* $u \in H^2(x_0, x_N)$*. Then*

$$\|S''\|_2 \leqslant \|u''\|_2.$$

The proof is not hard - it is analogous to the proof of Theorem 2.1.9.

2.2.4 Exercises

Exercise 2.3. When deducing the system of equations for the natural cubic spline, we showed how to construct the formulation given in (2.2) and the relationship between σ_i, α_i and β_i in (2.3). Now carefully show to deduce the system (2.4).

Exercise 2.4. (For students who did MA385). Write the equations in (2.4) as a linear system $A\boldsymbol{\sigma} = \mathbf{b}$, where A is an $n \times n$ matrix. Show that A is nonsingular, and hence that the system as a unique solution.

Exercise 2.5. Find the natural cubic spline interpolant to $f(x) = \sin(\pi x/2)$ at the nodes $\{x_i\}_{i=0}^3 = \{0,1,2,3\}$.

Calculate value of the interpolant at $x = 2.5$. What is the error at this point?

Exercise 2.6. Take $f(x) = \ln(x)$, $x_0 = 1$, $x_N = 2$. What value of N would you have to take to ensure that $|\ln(x) - S(x)| \leqslant 10^{-4}$ for all $x \in [1,2]$?

Exercise 2.7. (Loosely based on Burden and Faires, Q3.4.7) Suppose that S is a natural cubic spline on $[0,2]$ with

$$S(x) = \begin{cases} -3x + 2(1-x) + a(1-x)^3 + \frac{2}{3}x^3, & x \in [0,1), \\ b(2-x) + c(2-x)^3 + d(x-1)^3, & x \in [1,2]. \end{cases}$$

Find a, b, c, and d.

2.3 Piecewise Hermite Interpolation

In this section we introduce another type of cubic spline. It is not as smooth as the natural spline of the precious section—only it and its first derivative are continuous on $[x_0, x_N]$—and it requires that we know $f'(x_i)$. But its easier to construct (don't have to solve a linear system) and analyse than that natural spline.

As before, for short-hand, we write $f(x_i)$ as f_i and $f'(x_i)$ as f'_i.

And, as ever, we'll simplify the analysis by taking the points to be equally spaced: $x_i - x_{i-1} = h$ for each i).

2.3.1 PCHIP

Definition 2.3.1. Given a set of interpolation points $x_0 < x_1 < \cdots < x_N$, the *Piecewise Cubic Hermite Spline Interpolant* (*PCHIP*), S, to the function f, satisfies

(i) $S \in C^1[x_0, x_N]$,

(ii) $S(x_i) = f(x_i)$ and $S'(x_i) = f'(x_i)$ for $i = 0, 1, \ldots, N$.

(iii) On each interval $[x_{i-1}, x_i]$, S is a cubic polynomial.

We can construct these splines as follows. On each interval $[x_{i-1}, x_i]$, let S be the cubic polynomial S_i given by

$$S_i(x) = c_0 + c_1(x - x_{i-1}) + c_2(x - x_{i-1})^2 + c_3(x - x_{i-1})^3, \quad (2.5)$$

Then we can show... Take notes:

This gives that

$$\begin{aligned} c_0 &= f_{i-1}, \\ c_1 &= f'_{i-1}, \\ c_2 &= \frac{3}{h^2}(f_i - f_{i-1}) - \frac{1}{h}(f'_i + 2f'_{i-1}), \\ c_3 &= \frac{1}{h^2}(f'_i + f'_{i-1}) - \frac{2}{h^3}(f_i - f_{i-1}). \end{aligned}$$

Figure 2.5 shows some PCHIP interpolants to $f(x) = 1/(1+x^2)$ on the interval $[-5, 5]$. Compare with Figures 2.1 and 2.3.

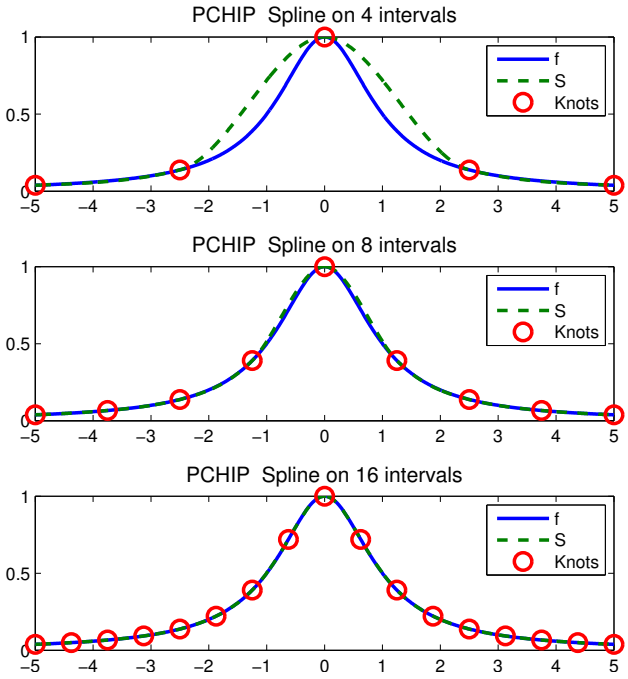


Fig. 2.5: PCHIP interpolants to $1/(1+x^2)$. Compare with Figure 2.3

We now want to prove an error estimate. The norm we use is

$$\|f - S\|_\infty := \max_{x_0 \leqslant x \leqslant x_N} |f(x) - S(x)|$$

Theorem 2.3.2. *Let $f \in C^4[x_0, x_N]$ and let S be the Hermite Cubic Spline interpolant to it at the N equally spaced points $a = x_0 < x_1 < \cdots < x_N = b$. Then*

$$\|f - S\|_\infty \leqslant \frac{h^4}{384}\|f^{(iv)}\|_\infty.$$

Take notes:

2.3.2 Estimating derivatives

You can ignore this section. It is included for completeness. It should make sense to anyone who took MA385, given that it relies only on Taylor series.

As we defined the PCHIP interpolant one needs to know f' in order to be able to construct it. However, you might notice that the MATLAB function `>> pchip` only requires f, not f'. How does it do this, one might wonder?

It turns out that there are important variants on the PHCIP scheme that don't involve knowing f'_0, f'_1, ..., f'_N, but instead uses *approximations* for f'. There are several approaches for deriving these estimates, including

(i) Geometrically. Recall, all we are really looking for is the slope of the tangent to f at $x = x_i$.

(ii) By finding a polynomial that interpolates f and differentiating that.

(iii) Taylor's Theorem.

For more details see [S96, Lecture 24]. Or look back at your notes for MA385, Lecture 14.

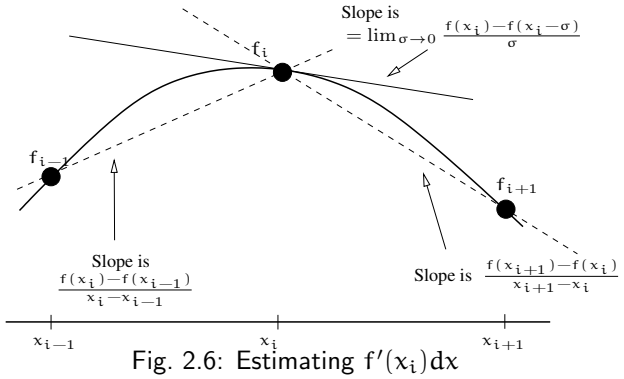


Fig. 2.6: Estimating $f'(x_i)dx$

As suggested by Figure 2.6 one could take

$$f'(x_i) \approx \frac{f_i - f_{i-1}}{x_i - x_{i-1}} = \frac{1}{h}(f_i - f_{i-1})$$

Or

$$f'(x_i) \approx \frac{f_{i+1} - f_i}{x_{i+1} - x_i} = \frac{1}{h}(f_{i+1} - f_i).$$

Since one seems to over estimate $f'(x_i)$, and the other seems to under-estimate, we could take the average:

$$f'(x_i) \approx \frac{-f_{i-1} + f_{i+1}}{x_{i+1} - x_{i-1}} = \frac{1}{2h}(-f_{i-1} + f_i)$$

These approximations can also be easily derived from Taylor's Theorem:

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2}f''(a) + \frac{(x-a)^3}{3!}f'''(a) + \cdots + \frac{(x-a)^n}{n!}f^{(n)}(a) + \frac{(x-a)^{n+1}}{(n+1)!}f^{(n+1)}(\eta),$$

for some $\eta \in (a, b)$. Using this we can get

$$f_{i+1} = f_i + hf'(x_i) + \frac{h^2}{2}f''(x_i) + \frac{h^3}{6}f'''(\tau_1),$$
$$f_{i-1} = f_i - hf'(x_i) + \frac{h^2}{2}f''(x_i) - \frac{h^3}{6}f'''(\tau_2),$$

for $\tau_1 \in (x_i, x_{i+1})$ and $\tau_2 \in (x_{i-1}, x_i)$. Now subtract the 1st equation from the second to get the formula and error estimate.

2.3.3 Exercises

Exercise 2.8. Recall Exercise 2.5. Calculate the value to the PCHIP interpolant to $f(x) = \sin(\pi x/2)$ at the nodes $\{x_i\}_{i=0}^3 = \{0, 1, 2, 3\}$ at the point $x = 2.5$. What is the error at this point?

Exercise 2.9. Let $f(x) = \ln(x)$. Let l and S be the piecewise linear and Hermite cubic spline interpolants (respectively) to f on $n+1$ equally spaced points $1 = x_0 < x_1 < \cdots < x_N = 2$. What value of n would you have to take to ensure that

(i) $\max_{1\leqslant x\leqslant 2} |f(x) - l(x)| \leqslant 10^{-4}$?

(ii) $\max_{1\leqslant x\leqslant 2} |f(x) - S(x)| \leqslant 10^{-4}$?

Also, using the code you developed in Lab 2, compare these theoretical results with the value needed on practice.

Exercise 2.10. There are ways of constructing the PCHIP, other than (2.5). For example, let $s = x - x_{k-1}$, then

$$S(x) = \frac{h^3 - 3hs^2 + 2s^3}{h^3}f_{k-1} + \frac{3hs^2 - 2s^3}{h^3}f_k + \frac{s(s-h)^2}{h^2}f'_{k-1} + \frac{s^2(s-h)}{h^2}f'_k$$

Show that this is the same as the PCHIP.

Chapter 3

Numerical Integration

3.1 Introduction

Problem: *Given a real-valued function* f *that is continuous on* $[a, b]$*, can we find an estimate for*

$$I(f) := \int_a^b f(x)dx?$$

And if we can, can we say how accurate it is?

Why bother?

- many problems in applicable mathematics require definite integrals to be evaluated.[1]
- evaluating them by finding the anti-derivative can be hard, and very hard to automate.
- some times, although the function is integrable, its anti-derivative doesn't exist in a closed form.

The process of numerically estimating a definite integral is called *Numerical Integration* or *Quadrature*.

The formulae we'll derive all look like

$$Q_n(f) := q_0 f(x_0) + q_1 f(x_1) + q_2 f(x_2) + \dots q_n f(x_n).$$

Here the points x_i are called *quadrature points* and the q_i are *quadrature weights*. So we need a way of choosing these. The simplest approach is to take the points to be equally spaced, i.e., $x_i = a + hi$ where $h = (b - a)/n$.

How to choose the weights? We've spent quite a while talking and thinking about approximating functions with polynomials. So why not find a polynomial interpolant to f and take the integral of that to be the answer? The appeal of this approach is due to the fact that

- Finding polynomial interpolants is easy.
- Integrating polynomials is easy.
- We can estimate the error easily (yet again, we'll make use of Theorem 1.3.3)

[1] These methods were originally motivated by problems in astronomy. They're credited to the English mathematician/astronomer Roger Cotes (1682–1716), working with Isaac Newton.

This leads to the Newton-Cotes methods, which are the subject of this section, and Section 3.2 Later again, we'll look at more sophisticated methods, called *Gaussian Methods* which use non-uniformly spaced points.

3.1.1 Newton-Cotes methods

Definition 3.1.1. The *Newton-Cotes* quadrature rule for $\int_a^b f(x)dx$ with $n + 1$ points is derived by integrating exactly the polynomial of degree n that interpolates f at the n equally spaced points $a = x_0 < x_1 < \cdots < x_n = b$. The method is written as

$$Q_n(f) := q_0 f_0 + q_1 f_1 + q_2 f_2 + \dots q_n f_n.$$

That is, the quadrature weights are chosen so that

$$Q_n(f) = \int_a^b p_n(x)dx,$$

where p_n is the polynomial of degree n that interpolates f at the $n + 1$ quadrature points. However, it turns out that we can compute the weights $q_0, \dots, q_n$, *without* knowing p_n. We'll do this for $n = 1$ in the next section, and $n = 2$ (the most interesting case) in Section 3.2.

3.1.2 The Trapezium rule

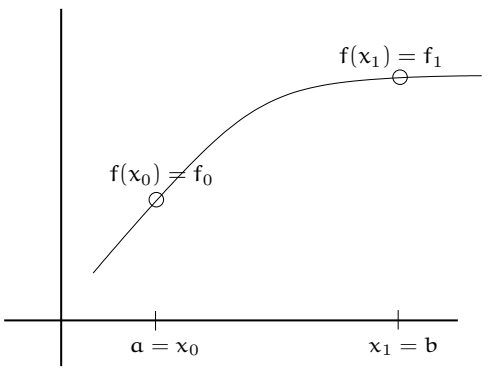


Fig. 3.1: Want to estimate $\int_a^b f(x)dx$

We want to estimate the integral of a function, f, on $[a, b]$, shown in Figure 3.1. Here are three approaches we could take.

Method 1: We could try to estimate the area of the trapezium the fits under the graph, as show in Figure 3.2. Clearly this comes to

$$Q_1(f) = (b-a)f_0 + (b-a)\frac{f_1 - f_2}{2} = (b-a)\frac{f_1+f_0}{2}. \quad (3.1)$$

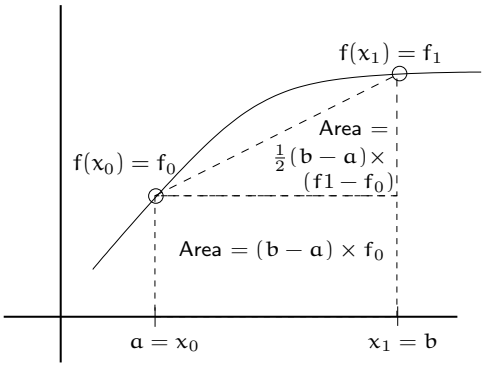


Fig. 3.2: Whats the area under the graph?

Method 2: As shown in Figure 3.3 we could find p_1, the polynomial of degree 1 that interpolates f at $x = a$ and $x = b$.

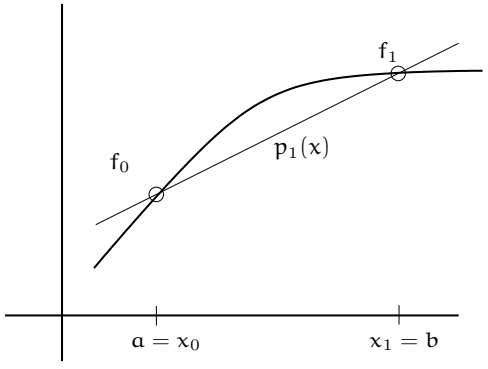


Fig. 3.3: Integrate $p_1(x)$ from x_0 to x_1

Take notes:

Note that this shows that $q_i = \int_a^b L_i(x)dx$, where, as usual, the L_i are the Lagrange Polynomials. (See §1.2.4).

Method 3: The third approach for generating the Trapezium Rule is called the *Method of Undetermined Coefficients*. Because the method is based on integrating a linear function we expect it to yield an exact solution for any constant or linear function (i.e., there should be no error). To keep the algebra simple, we'll take $a = 0$ and $b = 1$. So,

$$Q_1(f) = q_0 f(0) + q_1 f(1),$$

and, setting $f(x) \equiv 1$, and then $f(x) = x$ we get

Take notes:

Now we need to extend this to estimating $\int_a^b g(x)dx$ as follows:

Take notes:

Example 3.1.2. Use the Trapezium Rule to estimate

$$\int_0^{\pi/4} \cos(x)dx.$$

Calculate the (exact) error $|\int_a^b f(x)dx - Q_1(f)|$.

Take notes:

3.1.3 Exercises

Exercise 3.1. Let $q_0, q_1, \ldots, q_n$ be the quadrature weights for the Newton-Cotes rule $Q_n(f)$. Show that $q_i = q_{n-i}$ for $i = 0, \ldots n$.

Exercise 3.2. $\star$ Show that $\sum_{i=0}^{n} q_i = b - a$.

3.2 Simpson's Rule

Next we'll consider the *3-point Newton-Cotes scheme* which is based on integrating the quadratic interpolant to $f(x)$.

Simpson's Rule

$$Q_2(f) = \frac{b-a}{6}\big(f(a) + 4f\big(\frac{a+b}{2}\big) + f(b)\big). \tag{3.2}$$

This is called *Simpson's Rule*[2]. To show how to derive it, we'll use the Method of Undetermined Coefficients again. First restrict our attention to approximating $\int_0^1 g(x)dx$. This method should be exact for all constant, linear and quadratic polynomials. Taking $g(x) \equiv 1$, $g(x) = x$ and $g(x) = x^2$ we get the set of equations

Take notes:

This is easily solved giving

$$\int_0^1 g(x)dx \approx \frac{1}{6}g(0) + \frac{2}{3}g(1/2) + \frac{1}{6}g(1). \tag{3.3}$$

To extend this to the interval $[a, b]$, we again use a change of variables to get the general Simpson's Rule (3.2) .

Example 3.2.1. Use Simpson's rule to estimate $\int_0^{\pi/4} \cos(x)dx$, and calculate the (exact) error $|\int_a^b f(x)dx - Q_2(f)|$.

Take notes:

[2]Thomas Simpson, 1710–1761. One of the most distinguished of a group of itinerant lecturers who taught in the London coffee-houses, Hutton (famous text-book writer) said of him

> *It has been said that Mr Simpson frequented low company, with whom he used to guzzle porter and gin: but it must be observed that the misconduct of his family put it out of his power to keep the company of gentlemen, as well as to procure better liquor.*

The method was known well before Simpson's time: it had been used by Cavalieri (a student of Galileo) in 1639, James Gregory, Johannes Kepler, and others.

3.2.1 Newton-Cotes error estimates

We'll now derive error estimates for general Newton-Cotes methods, and look at the specific cases of the Trapezium and Simpson's rules.

Theorem 3.2.2. *Let*

$$M_{n+1} := \max_{a \leqslant x \leqslant b} |f^{(n+1)}(x)|,$$

and $\pi_{n+1}(x)$ be the usual nodal polynomial. Define

$$\mathcal{E}_n := \Big|\int_a^b f(x)dx - Q_n(f)\Big|.$$

Then

$$\mathcal{E}_n \leqslant \frac{M_{n+1}}{(n+1)!}\int_a^b |\pi_{n+1}(x)|dx,$$

The proof just comes directly Cauchy's Theorem (Theorem 1.3.3).

Take notes:

Error estimates for the Trapezium Rule

Theorem 3.2.3. *For the Trapezium Rule (3.1)*

$$\mathcal{E}_1 \leqslant \frac{(b-a)^3}{12}M_2. \tag{3.4}$$

The proof is an exercise.

Example 3.2.4. Use (3.4) to get an upper bound on the error for the estimate of $\int_0^{\pi/4} \cos(x)dx$ using the Trapezium rule. How does this compare with the actual error that we found in Example 3.1.2?

Take notes:

Example 3.2.5. If use the Trapezium Rule to estimate the integral of x^2 on the interval $[0, 1]$ we get

$$\int_0^1 x^2dx = \frac{1}{3} \qquad \text{and} \qquad Q_1(x^2) = \frac{1}{2}(0+1) = \frac{1}{2}.$$

So the error is $1/6$, exactly as the theory predicts.

Error estimates for Simpson's rule

One could also use Theorem 3.2.2 to show that, for Simpson's Rule,

$$\mathcal{E}_2 \leqslant \frac{(b-a)^4}{196}M_3, \tag{3.5}$$

but don't bother because, although correct, it is not *sharp* (that is, it is pessimistic).

Example 3.2.6. Use (3.5) to get an upper bound on the error for the estimate of $\int_0^{\pi/4} \cos(x)dx$ using **Simpson's** rule. How does this compare with the actual error that we found in Theorem 3.2.1?

Take notes:

Example 3.2.7. We expect Simpson's Rule to give *exactly* the right answer for integrals of constant, linear and quadratic functions. If we take $f(x) = x^3$, $a = 0$ and $b = 1$, then formula above suggests that (approx) $\mathcal{E}_2 \leqslant 0.03$. But

Take notes:

3.2.2 Exercises

Exercise 3.3. Deduce the 4-point Newton-Cotes Rule for estimating the integral $\int_0^1 f(x)dx$:

$$Q_3(f) = q_0 f(x_0) + q_1 f(x_1) + q_2 f(x_2) + q_3 f(x_3).$$

Extend the rule to estimate the integral of functions over $[a, b]$.

Exercise 3.4. Prove the error bound given for the Trapezium rule. That is, show that

$$|\int_a^b f(x)dx - Q_1(f)| := \mathcal{E}_1 \leqslant \frac{(b-a)^3}{12} M_2.$$

3.3 Precision and Composition

3.3.1 Precision

We know from Example 3.2.7 that Simpson's Rule applied to approximating $\int_0^1 x^3 dx$ yields exactly the right answer (i.e., the error is zero).

We now claim that Simpson's Rule is exact for *any* polynomial of degree 3 or less.

In class to simplified by taking $a = -1$ and $b = 1$. Here are the **details** for the general case. Denote by $Q_2(f)$ the approximation of $\int_a^b f(x)dx$ with Simpson's Rule:

$$Q_2(f) = \frac{b-a}{6}\Big(f(a) + 4f\big(\frac{a+b}{2}\big) + f(b)\Big).$$

Since the method can be derived by integrating the quadratic that interpolates $f(x)$ at the three points a, $(a+b)/2$, and b, it is clearly exact for all quadratics. (We also know this from Theorem 3.2.3).

Let $f(x) = c_0 + c_1x + c_2x^2 + c_3x^3$. Then

$$\int_a^b f(x)dx = \int_a^b \big(c_0 + c_1x + c_2x^2\big)dx + c_3\int_a^b x^3dx = \int_a^b \big(c_0 + c_1x + c_2x^2\big)dx + c_3\frac{b^4 - a^4}{4}.$$

Also,

$$Q_2(f) = Q_2(c_0 + c_1x + c_2x^2) + Q_2(c_3x^3) = \int_a^b \big(c_0 + c_1x + c_2x^2\big)dx + c_3\big(\frac{b-a}{6}\big)\Big(a^3 + 4\big(\frac{a+b}{2}\big)^3 + b^3\Big). \quad (3.6)$$

With a bit of symbolic manipulation we get that

$$\frac{b^4 - a^4}{4} = \big(\frac{b-a}{6}\big)\Big(a^3 + 4\big(\frac{a+b}{2}\big)^3 + b^3\Big),$$

as required.

In (3.5) we gave the result of a nïeve attempt to derive an upper bound for the error in Simpson's rule:

$$\mathcal{E}_2 \leqslant \frac{(b-a)^4}{196}M_3.$$

This is not wrong – just not sharp. For example it does not give that Simpson's Rule is exact for all cubics. The sharp result is

Theorem 3.3.1.

$$|E_2(x)| = |\int_a^b f(x)dx - Q_2\big(f(x)\big)| \leqslant \frac{(b-a)^5}{2880}M_4.$$

For the proof see the text book (Theorem 7.2 of [SM03]). Instead of working through it in class we'll prove a more general version of a consequence of Theorem 3.3.1.

Definition 3.3.2 (Precision of a Quadrature Rule)**.** A quadrature rule has *precision* n if it is exact for all polynomials of degree n or less. That is, the rule $Q(f)$ has precision n if

$$Q(p_n) = \int_a^b p_n(x)dx \quad \text{for all } p_n \in \mathcal{P}_n.$$

Example 3.3.3. By construction, the $(n+1)$-point Newton-Cotes rule has precision n.

Theorem 3.3.4. *If $Q_{2k}(\cdot)$ is a Newton-Cotes quadrature rule on $2k+1$ points, then $Q_{2k}(\cdot)$ has in fact precision $2k+1$.*

Proof: Let p_{n+1} be a polynomial of degree $n+1$. We wish to show that $Q_n(p_{n+1}) = \int_a^b p_{n+1}(x)dx$. We can take $a = -1$, $b = 1$ because a simple linear transformation can be used to map to an arbitrary interval.

Also, since the quadrature points are equally spaced on $[-1, 1]$ we have that $x_i = -x_{n-i}$.

Furthermore (see Exercise 3.1), the quadrature weights are symmetric: $q_i = q_{n-i}$.

Take notes:

3.3.2 Composite Rules

Suppose that we want to estimate $\int_a^b f(x)dx$ and the Trapezium rule is not sufficiently accurate. We could try Simpson's Rule, which should be better. Failing that, we could try a Newton-Cotes rule that uses more points. However, quite apart from the fact that it might be tedious to derive these rules, such schemes are based on poynomial interpolation, and we know (Runge's example again!) that high-order polynomial interpolation can be very inaccurate. See Exercise 3.5

It is better to use a *Composite Rule*. This is analogous to the piecewise polynomial interpolation introduced in Section 2.1. For the Composite Trapezium Rule, as shown in Figure 3.4, we divide $[a, b]$ into N intervals of size $h = (b-a)/N$. Applying the Trapezium Rule on each interval $[x_{i-1}, x_i]$ we get

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx h\frac{f_{i-1}+f_i}{2}.$$

Summing for the n intervals we get

$$\int_a^b f(x)dx \approx \frac{b-a}{N}\left(\frac{f_0}{2} + f_1 + f_2 + \cdots + f_{N-1} + \frac{f_N}{2}\right). \quad (3.7)$$

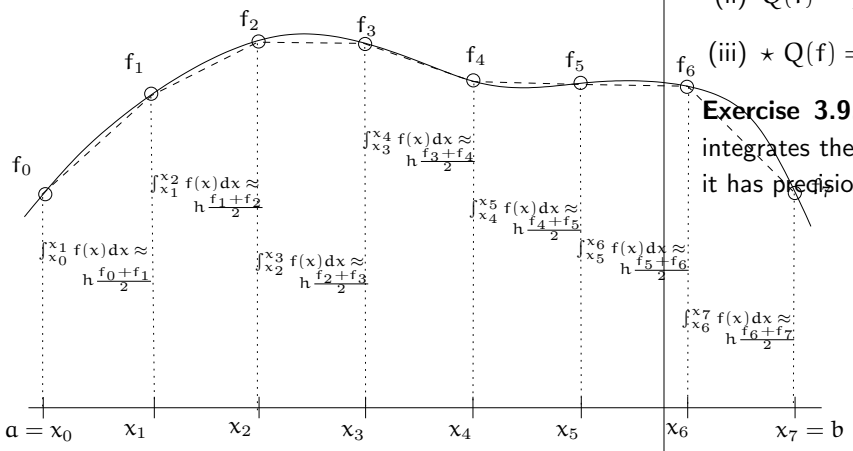


Fig. 3.4: Estimating $\int_a^b f(x)dx$ using the Composite Trapezium Method

3.3.3 Exercises

Exercise 3.5. Explain clearly, with an example, why in general it is not true that $Q_n(f) \to \int_a^b f(x)dx$ as $n \to \infty$.

Exercise 3.6.

(i) Use Theorem 3.2.2 to deduce an error estimate for the Composite Trapezium Rule (3.7).

(ii) Taking $N = 10$, give an upper bound for the error in the Composite Trapezium Rule when approximating $\int_1^2 \ln(x)dx$.

(iii) What value of n would you have to take to ensure that the error was less that 10^{-5}?

Exercise 3.7.

(i) Deduce the formula for the *composite Simpson's Rule*, and use Theorem 3.3.1 to derive an error estimate.

(ii) What value of N would you have to take to ensure that the error in the estimate of $\int_1^2 \ln(x)dx$ is less that 10^{-6}?

(iii) Denote the $(N+1)$-point Composite Simpson's Rule by $S_N(f) \approx \int_a^b f(x)dx$. Show that, for sufficiently smooth $f(x)$,

$$\lim_{n\to\infty} S_N(f) = \int_a^b f(x)dx.$$

Exercise 3.8. Determine the precision of the following schemes for estimating $\int_0^1 f(x)dx$.

(i) $Q(f) = f(\frac{1}{2})$.

(ii) $Q(f) = \frac{1}{4}f(0) + \frac{3}{4}f(\frac{2}{3})$.

(iii) $\star$ $Q(f) = \frac{3}{2}f(\frac{1}{3}) - 2f(\frac{1}{2}) + \frac{3}{2}f(\frac{2}{3})$.

Exercise 3.9. Suppose that a quadrature rule exactly integrates the polynomials 1, x, x^2, ..., x^n. Show that it has precision n.

3.4 Gaussian Quadrature

3.4.1 Introduction

To date we have found some numerical schemes that approximate $\int_a^b f(x)dx$ as the weighted average of values of f at $n+1$ equally spaced points. These methods have precision n (or $n+1$ in special cases).

With Gaussian Quadrature [3] we choose both the quadrature weights and points in such as way as to maximize the precision of the method. There are three equivalent approaches to doing this.

(i) *Undetermined Coefficients*: The obvious way for, say, $n = 2$. But, unlike Newton-Cotes, we have to solve a system of *nonlinear* equations. Even for $n = 3$ this can become difficult.

(ii) *Base the method on integrating the Hermite Interpolant of the integrand, f (see §1.5)*, and choose the points so that the coefficients of $f'(x_i)$ are zero. This approach is the easiest to analyse, but less useful for construction.

(iii) *Finding the zeros of the members of a sequence of orthogonal monic polynomials*. This is the approach we will emphasise most, as it gives us an easy way of proving the precision of the methods.

This section is perhaps the most mathematically rich in the course. I'd encourage you to read further: Chapter 10 of Süli and Mayers [SM03] is devoted to this. However, much of the basic theory is developed in Section 9.4 on Orthogonal Polynomials. See also [S96, Lectures 22 and 23] and [SB92, §3.6].

3.4.2 Undetermined Coefficients

Example 3.4.1. *Find a two point rule*

$$\int_{-1}^{1} f(x)dx \approx G_1(f) := w_0 f(x_0) + w_1 f(x_1),$$

that is exact for all polynomials of degree 3 or less.

This leads a the non-linear system which we must solve

Take notes:

That is

$$\int_{-1}^{1} f(x)dx \approx G_1(f) := f(\frac{-1}{\sqrt{3}}) + f(\frac{1}{\sqrt{3}}). \qquad (3.8)$$

Example 3.4.2. Let $f(x) = \exp(-x)$. If we estimate $\int_{-1}^{1} f(x)dx$ using each of the Trapezium Rule, Simpson's Rule and the Gaussian Rule above we find the errors are, respectively, 0.735, 0.01165 and 0.00771. Using the composite trapezium rule, we find we would have to take $N = 11$ to obtain an estimate that is more accurate than the two-point Gaussian Rule.

Example 3.4.3. If you use the $G_1(\cdot)$ rule to estimate $\int_0^{\pi/4} \cos(x)dx$ you'll find that

$$\Big|\int_0^{\pi/4} \cos(x)dx - G_1(\cos)\Big| = \Big|\frac{1}{\sqrt{2}} - 0.07070432596\Big| \approx 6.35 \times 10^{-5}.$$

Compare with results for the same problem when

- The trapezium rule is used (Theorem 3.1.2).
- Simpson's rule is used (Theorem 3.2.1).

Example 3.4.4. A variation of Gaussian quadrature is *constrained Gaussian quadrature*, also known is the *Gauss-Lobatto method*. Rather than allowing all of the quadrature points to vary in order to maximize the precision of the method, we fix some of them — usually the end points. Use undetermined coefficients to derive the 3-point *Gauss-Lobatto rule*:

$$\int_{-1}^{1} f(x)dx \approx w_0 f(-1) + w_1 f(x_1) + w_2 f(1),$$

where we fixed $x_0 = 0$ and $x_2 = 1$.

Take notes:

[3] 

Johann Carl Friedrich Gauß, born 1777 in Braunschweig, died 1855 in Göttingen.
Image source: http://jeff560.tripod.com/stamps.html

3.4.3 Exercises

Exercise 3.10. Use a change of variables, as we did with the Trapezium rule, to show that the rule for approximating $\int_0^1 f(x)dx$ is

$$G_1(f) = \frac{1}{2}\left(f(\frac{1}{2} - \frac{1}{2\sqrt{3}}) + f(\frac{1}{+}\frac{1}{2\sqrt{3}})\right).$$

More generally, extend the $G_1(f)$ rule in (3.8) to an arbitrary interval $[a, b]$.

Exercise 3.11. Use $G_1(x)$ to estimate $\int_1^2 \ln(x)dx$. How does this compare with the Trapezium and Simpson's Rule?

Exercise 3.12. Derive a 3-point Gaussian Quadrature Rule to estimate $\int_{-1}^1 f(x)dx$. *Hint:* $x_1 = 0$.

3.5 Orthogonal Polynomials

High order Newton-Cotes methods are of little use because of the problems associated with interpolation be high degree polynomials at equally spaced points. However, high-order Gaussian methods are very useful.

Driving such methods by undetermined coefficients is not practical, however. There is a simpler way, but some mathematical preliminaries are required.

3.5.1 Inner products

Definition 3.5.1 (Vector Space)**.** V is a *vector space* (a.k.a., a *linear space*) over a field F (e.g, the real or complex numbers) if for all $\mathbf{u}, \boldsymbol{v}, \boldsymbol{w} \in V$ and $a, b \in F$:

(i) $\mathbf{u} + \boldsymbol{v} \in V$ (closed under addition)

(ii) $\mathbf{u} + \boldsymbol{v} = \boldsymbol{v} + \mathbf{u}$ (Commutativity)

(iii) $(\mathbf{u} + \boldsymbol{v}) + \boldsymbol{w} = \mathbf{u} + (\boldsymbol{v} + \boldsymbol{w})$ (Associativity)

(iv) V has a zero vector $\mathbf{0}$ such that $\mathbf{u} + \mathbf{0} = \mathbf{u}$.

(v) $-\mathbf{u} \in V$

(vi) $a\mathbf{u} \in V$

(vii) $a(b\mathbf{u}) = (ab)\mathbf{u}$

(viii) F contains 0 and 1 such that $1\mathbf{u} = \mathbf{u}$, $0\mathbf{u} = \mathbf{0}$.

(ix) $a(\mathbf{u} + \boldsymbol{v}) = a\mathbf{u} + a\boldsymbol{v}$, and $(a + b)\mathbf{u} = a\mathbf{u} + b\mathbf{u}$.

Examples:

Definition 3.5.2 (Inner Product)**.** Let V is a real vector space. An **Inner Product** (IP) is a real-valued function $(\cdot, \cdot)$ on $V \times V$ such that, for all $f, g, h \in V$,

(i) $(f + g, h) = (f, h) + (g, h)$,

(ii) $(\lambda f, g) = \lambda(f, g)$, for $\lambda \in \mathbb{R}$.

(iii) $(f, g) = (g, f)$,

(iv) $(f, f) \geqslant 0$. $\quad (f, f) = 0 \iff f \equiv 0$.

Example 3.5.3. Let $\mathbb{R}^n$ be our vector space, with $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$. Then

$$(\mathbf{x}, \mathbf{y}) := \sum_{i=1}^{n} x_i y_i,$$

is an inner product.

Example 3.5.4. The set of real-valued functions that are continuous and defined on the ionterval $[a, b]$, denoted $C[a, b]$, is a vector space. And

$$(f, g) := \int_a^b f(x)g(x)dx, \tag{3.9}$$

is an inner product on this space.

We could consider the more general family "weighted" inner products:

$$(f, g) := \int_a^b w(x)f(x)g(x)dx,$$

where $w(x)$ is some positive "weight function." However the IP we will use for the remainder of this chapter is always the one defined in (3.9). That is, we are just concerned with the case $w(x) \equiv 1$.

3.5.2 A Sequence of Orthogonal Monic Polynomials

(Please see [S96, Lecture 23] for more details).

Definition 3.5.5 (Monic Polynomial)**.** A polynomial is *monic* if the coefficient of its leading term is 1.

Example 3.5.6.

Take notes:

Definition 3.5.7. Two elements a, b, of a vector space are *orthogonal* with respect to a given inner product $(\cdot, \cdot)$ if $(a, b) = 0$.

Example 3.5.8. Take notes:

Example 3.5.9. Take the space of polynomials of degree 2 or less and the IP

$$(f, g) = \int_{-1}^{1} f(x)g(x)dx.$$

Let $p(x) \equiv 1$, $q(x) \equiv x$, $r(x) \equiv x^2 - 1/3$, and $f(x) = 3x - 4$, then:
Take notes:

3.5.3 A sequence of orthogonal polynomials

As given in Definition 3.5.5, a polynomial is *monic* if the coefficient of the leading term is 1:

$$p_n = x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-1} + \cdots + c_1x + c_0.$$

We'll now look at a sequence of such polynomials $\{\widetilde{p}_0, \widetilde{p}_1, \widetilde{p}_2, \ldots, \widetilde{p}_n, \ldots\}$ that have the property they are orthogonal to each other:

$$(\widetilde{p}_i, \widetilde{p}_j) := \int_a^b \widetilde{p}_i(x)\widetilde{p}_j(x)dx = 0 \quad \text{if } i \neq j.$$

We want to establish some important facts about monic polynomials:

- A set of monic polynomials of degrees 1, 2, ..., n, forms a basis for $\mathcal{P}_n$.
- If the members of that set are orthogonal to each other, then they are orthogonal to *all* polynomials of lower degree.
- We can construct such as set.

Lemma 3.5.10. *Let $\{p_i\}_{i=0}^n$ be a sequence of polynomials where each p_i is monic an* exactly *of degree i. This sequence forms as basis for $\mathcal{P}_n$.*

Proof:
Take notes:

This lemma means that if q is a polynomial of degree n then it can be written uniquely as a linear combination of the p_i:

$$q(x) = \sum_{i=0}^{n} a_i p_i(x),$$

for some unique choice of the real coefficients a_i.

Definition 3.5.11. The sequence $\{\widetilde{p}_i\}_{i=0}^n$ is a sequence of *monic*, *orthogonal* polynomials if each $\widetilde{p}_i$ is monic and *exactly* of degree i and

$$(\widetilde{p}_i, \widetilde{p}_j) = 0 \quad \text{if } i \neq j.$$

Lemma 3.5.12. *If $\widetilde{p}_j \in \{\tilde{p}_i\}_{i=0}^\infty$ then $\widetilde{p}_j$ is orthogonal to all polynomials of degree less than j.*

Proof:
Take notes:

3.5.4 Constructing the Sequence

Theorem 3.5.13. *The sequence $\{\tilde{p}_i\}_{i=0}^\infty$ exists and can be constructed as follows: Let α and β be defined as*

$$\alpha_{n+1} = \frac{(x\widetilde{p}_n, \widetilde{p}_n)}{(\widetilde{p}_n, \widetilde{p}_n)}, \quad \textit{and} \quad \beta_{n+1} = \frac{(x\widetilde{p}_n, \widetilde{p}_{n-1})}{(\widetilde{p}_{n-1}, \widetilde{p}_{n-1})},$$

then the sequence is given by

$$\widetilde{p}_0(x) \equiv 1, \qquad \widetilde{p}_1(x) = x - \alpha_1$$

and

$$\widetilde{p}_{n+1}(x) = (x - \alpha_{n+1})\widetilde{p}_n(x) - \beta_{n+1}\widetilde{p}_{n-1}(x),$$

for $n \geqslant 1$.

The proof uses *Gram-Schmidt Orthogonalization*.
Take notes:

Example 3.5.14. If we use the inner product $(f, g) := \int_0^1 f(x)g(x)$ then the first 4 polynomials in the sequence are:
Take notes:

Example 3.5.15. The zeros of $\widetilde{p}_2$ are ...

Take notes:

3.5.5 Properties of the sequence

One of the ways of constructing Gaussian Quadrature rule $G_n(\cdot)$ on $n+1$ is to take the quadrature points as the roots of $\widetilde{p}_{n+1}$. We know (from the fundamental theorem of algebra) a polynomial of degree $n+1$ has exactly $n+1$ roots in $\mathbb{C}$ up to multiplicity.

However, the polynomials $\widetilde{p}$ have the special properties, established by the following result.

Lemma 3.5.16. *Let* $\widetilde{p}_i \in \{\tilde{p}_i\}_{i=0}^{\infty} = \{\widetilde{p}_0, \widetilde{p}_1, \dots, \}$ *be the set of monic polynomials that are orthogonal with respect to the (usual) inner product.*

(i) The zeros of each $\widetilde{p}_i \in \{\tilde{p}_i\}_{i=0}^{\infty}$ *are simple (not repeated).*

(ii) All the zeros of $\widetilde{p}_i$ *are real numbers in the interval* $[a, b]$.

(*A slightly different proof of these facts are given in [SM03, Thm 9.4].*

Take notes:

3.5.6 Exercises

Exercise 3.13. Suppose that $(\cdot, \cdot)$ is an inner product. Show that $\|u\| := \sqrt{(u, u)}$ is a norm.

Exercise 3.14. $\mathcal{P}_n$, the space of polynomials of degree (at most) n forms a vector space. Is it true that the space of *monic* polynomials of degree n forms a vector space?

Exercise 3.15. (i) Using the Inner Product

$$(f, g) := \int_{-1}^{1} f(x)g(x)dx,$$

find $\widetilde{p}_0(x)$, $\widetilde{p}_1(x)$, $\widetilde{p}_2(x)$ and $\widetilde{p}_3(x)$.

(ii) Find the zeros of $\widetilde{p}_2(x)$ and call them x_0 and x_1. Construct a quadrature rule for $\int_{-1}^{1} f(x)dx$ taking these as the quadrature points, and the weights as the integrals to the corresponding Lagrange polynomials. Verify that this is the same rule as given in (3.8).

(iii) Repeat this exercise using the zeros of $\widetilde{p}_3(x)$ as the quadrature points. Verify that the rule you get is the same as Exercise 3.12.

3.6 Gaussian Quadrature via Orthogonal Polynomials

At the start of this section, we introduced the using Gaussian Quadrature technique for estimating integrals

$$\int_a^b f(x)dx \approx G_n(f) := \sum_{k=0}^{n} w_k f(x_k),$$

where the points x_k and weights w_k are chosen to maximise the precision of the method.

We now have an equivalent way of defining the method, $G_n(\cdot)$, and deriving the coefficients:

(a) Write down the set of monic polynomials $\{\widetilde{p}_0, \widetilde{p}_1, \ldots, \widetilde{p}_{n+1}\}$ that is orthogonal with respect to the inner product

$$(u, v) := \int_a^b u(x)v(x)dx.$$

Note that a and b, the limits of integration for the IP, are the same as for the integral we are trying to estimate.

(b) From Theorem 3.5.16, we know that $\widetilde{p}_{n+1}$ has $n+1$ zeros in the interval $[a, b]$. Let these be the quadrature points of the method.[4]

For example, the polynomial $\widetilde{p}_5 = x^5 - (10/9)x^3 + (5/21)x$ is shown in Figure 3.5, with its zeros highlighted.

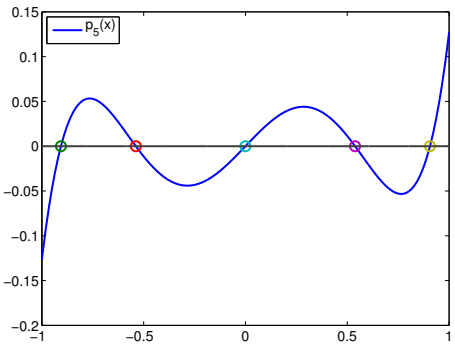


Fig. 3.5: The polynomial $\widetilde{p}_5 = x^5 - (10/9)x^3 + (5/21)x$

(c) Take the quadrature weights to be $w_k = \int_a^b L_k(x)dx$, where the L_k are the usual Lagrange polynomials for this set of points.

The key property of this method in stated in the following theorem.[5]

[4] In practice one would take a simpler IP such as

$$(u, v) = \int_{-1}^{1} u(x)v(x)\,dx,$$

and then transform the zeros from $[-1, 1]$ to $[a, b]$ using a linear transformation.

[5] In an earlier version of these notes, this was number Theorem 3.6.16

Theorem 3.6.1. *Let $x_0, \ldots, x_n$ to be the zeros of $\widetilde{p}_{n+1}$, the $(n+1)$th polynomial in the sequence of orthogonal monic polynomials $\{\tilde{p}_i\}_{i=0}^{\infty}$. Set*

$$G_n(f) = w_0 f(x_0) + \cdots + w_n f(x_n)$$
$$\text{where } w_i = \int_a^b L_i(x)dx. \quad (3.10)$$

Then $G_n(f)$ has precision $2n+1$.

The proof of this is an exercise.

3.6.1 Error estimates

Our final task associated with numerical integration is to prove that, as $n \to \infty$, so $G_n(f) \to \int_a^b f(x)dx$.

We are not going to do this in full detail, but the key ideas will be presented.

We would like to prove the following error estimate, which is closely related to Theorem 1.5.2 (error in Hermitian interpolation):

Theorem 3.6.2.

$$\int_a^b f(x)dx - G_n(f) = \frac{f^{(2n+2)}(\tau)}{(2n+2)!}\int_a^b \pi_{n+1}(x)^2 dx.$$

The trick here is that we are claiming that Gaussian Quadrature is the same as integrating the Hermite interpolant to f, even though the latter depends on $f'(x_k)$ although the former does not.

Moreover, we would like to show that

$$G_n(f) \to \int_a^b f(x)dx \text{ as } n \to \infty.$$

To do this we need to establish several facts. In each case we make use of Theorem 3.6.1, a consequence of which is that, if f is a polynomial of degree at most $2n-1$, then

$$\int_a^b f(x)dx = \sum_{k=0}^{n} w_k f(x_k).$$

First, recall the basis functions that we use for Hermite Interpolation Section 1.5:

$$H_i(x) = [L_i(x)]^2(1 - 2L_i'(x_i)(x - x_i)),$$

$$K_i(x) = [L_i(x)]^2(x - x_i).$$

Take notes:

Now we can deduce Theorem 3.6.2.

Take notes:

Next we want to show that each of the w_k are positive. From (3.10) we have that the Gaussian Quadrature weights are $w_k = \int_a^b L_k(x)dx$ where the L_k are the usual Lagrange Polynomials:

$$L_k(x_j) = \begin{cases} 1 & k = j \\ 0 & k \neq j, \end{cases}$$

associated with the Gaussian interpolation points. Then in fact
Take notes:

and so that the w_k are all positive. It follows directly that $0 < w_k < (b-a)$, for $k = 0, 1, 2, \ldots, n$:
Take notes:

3.6.2 Convergence

Section 10.4 of [SM03] also covers this, though from a different angle. One of the most interesting aspects of this theory is given in Theorem 10.2:

Theorem 3.6.3.

$$\lim_{n\to\infty} G_n(f) = \int_a^b f(x)dx.$$

(*Here we'll just given an out line of the proof. You are not required to know this for the exam*).

The Weierstrass approximation theorem, which we mentioned back in Section 1.6.1, tells us that, for any $\epsilon > 0$, there exists a polynomial p such that $|f(x)-p(x)| \leqslant \epsilon$. Let n be the degree of this polynomial. Let $G_n(\cdot)$ be the $n+1$ point Gaussian Quadrature rule. Then

$$\int_a^b f(x)dx - G_n(f) = \int_a^b f(x) - p(x)dx + \int_a^b p(x)dx - G_n(p) + G_n(p) - G_n(f).$$

But because $G_n(\cdot)$ is exact for polynomials of degree n, $\int_a^b p(x)dx - G_n(p) = 0$. Using this, and the triangle inequality,

$$\left|\int_a^b f(x)dx - G_n(p)\right| \leqslant \left|\int_a^b f(x) - p(x)dx\right| + \left|G_n(p) - G_n(f)\right|.$$

But

$$\left|\int_a^b f(x) - p(x)dx\right| \leqslant \int_a^b \varepsilon dx = \varepsilon(b-a).$$

Also,

$$|G_n(f) - G_n(p)| = |G_n(p-f)| = \left|\sum_{k=0}^n w_k\big(f(x_k) - p(x_k)\big)\right| = \sum_{k=0}^n w_k\big|f(x_k) - p(x_k)\big|,$$

because all the w_i are positive. Then, using that $\sum_k w_k = b-a$ and $|f(x)-p(x)| \leqslant \varepsilon$, we get

$$\big|G_n(f) - G_n(p)\big| \leqslant \varepsilon(b-a).$$

Combining these we have shown that for any ε, one can always find n such that

$$\left|\int_a^b f(x)dx - G_n(p)\right| \leqslant 2\varepsilon(b-a).$$

. . . . . .

One of the key ingredients to this proof was that the w_i are all positive. That is not the case for Newton-Cotes methods—for large enough n their quadrature weights can be negative—so no similar result is possible.

3.6.3 Exercise

Exercise 3.16. Prove Theorem 3.6.1.

Exercise 3.17. $\star$ Show that it is impossible to choose $n+1$ quadrature points and weights so that the $n+1$-point quadrature rule

$$\int_a^b f(x)dx \approx \sum_{k=0}^n w_k f(x_k)$$

has precision $2n+2$.

Hint: To show the method does not have precision $2n+2$, you just need to give a an example of a single polynomial p of degree exactly $2n+2$ for which $\int_a^b p(x)dx \neq \sum_{k=0}^n w_k f(x_k)$.

Chapter 4

Finite Element Methods for BVPs

4.1 Boundary value problems

In this final section of MA378 we consider numerical schemes for particular ordinary differential equations called *Boundary Value Problems (BVPs)*. The main differences between them and the initial value problems (IVPs) of MA385/530 are:

- The equation gives the solution to the differential equation *two* (boundary) points, not just one (initial) point.
- The second derivative of the solution *always* appears in the equation. The IVPs we studied usually only had first derivatives.
- We usually think of the independent variable, x, as representing *space* rather than time.

For some BVPs it is possible to write down the exact solution; an example of such a case is given below in Example 4.1.1. Usually, however, this is not easy, or, indeed, possible. So numerical methods are needed to given *approximate* solutions. Two of the most popular numerical methods are

- *Finite Difference Methods* (FDMs), where one approximates the differential equation, based on Taylor series expansions.
- *Finite Element Methods* (FEMs), where the solution space is approximated, using ideas related to interpolation.

Since FEMs are in keeping with the philosophy of this course, we will study them.

More details are in Chap. 14 of the textbook [SM03].

4.1.1 Boundary value problems

To formulate a model BVP we first define a differential operator

$$L(u) := -u''(x) + r(x)u(x). \tag{4.1}$$

Then the general form of a BVP is: *find a function* u*, defined on the interval* $[a, b]$*, such that*

$$\begin{aligned} L(u) = f(x) \quad \text{for} \quad a < x < b, \\ u(a) = 0, u(b) = 0. \end{aligned} \tag{4.2}$$

We make the assumptions that r and f are continuous and have as many continuous derivatives as we would like. Furthermore we always have that $r(x) > 0$ for all $x \in [a, b]$.

Some BVPs are reasonably easy to solve by hand. That is, we can write down a solution in terms of elementary functions. That is certainly the case if the functions r and f are constant:

Example 4.1.1. Verify that, for any constants c_0 and c_1, $u(x) = c_0 e^{2x} + c_1 e^{-2x} + x/4$ is a solution to

$$-u''(x) + 4u(x) = x \quad \text{for} \quad 0 \leqslant x \leqslant 3, \tag{4.3}$$

Take notes:

If we have two boundary conditions, we can uniquely determine c_0 and c_1. See Exercise 4.2.

Unlike Example 4.1.1, most boundary value problems are difficult or impossible to solve analytically and so, in general, we need approximations. However, it can often be possible to obtain *qualitative* information about the solutions.

Lemma 4.1.2 (Maximum Principle)**.** *Suppose that* L *is as defined in* (4.1)*, with* $r(x) > 0$ *for all* x*, and* u *is a function such that* $Lu \geqslant 0$ *on* (a, b)*. If* $u(a) \geqslant 0$*,* $u(b) \geqslant 0$*. Then* $u \geqslant 0$ *for all* $x \in [a, b]$*.*

Proof:
Take notes:

This lemma is as useful as it is simple. See the following lemma, and also Exercise 4.4

Lemma 4.1.3. *There is at most one solution to (4.2).*

Take notes:

4.1.2 Exercises

Exercise 4.1. Suppose, instead of the differential operator defined in (4.1), we had the more general one:

$$L_q(u) := -u''(x) + q(x)u'(x) + r(x)u(x).$$

Does this L_q also satisfy a maximum principle? If so, provide a proof. If not, give a counter example.

Exercise 4.2. Verify that

$$u(x) = \frac{x}{4} + \frac{3e^6(e^{-2x} - e^{2x})}{4(e^{12} - 1)}$$

is the exact solution to (4.1.1) with the boundary conditions $u(0) = 0$, $u(3) = 0$,

Exercise 4.3. In this section of the course, we'll always assume homogeneous boundary conditions. That is, that $u(x) = 0$ at the boundaries. Suppose the problem we wish to solve is

$$-u''(x) + r(x)u(x) = f(x) \quad u(0) = \alpha, u(1) = \beta.$$

Show how to find a problem which has the same left-hand side as this one, homogeneous boundary conditions, and with a solution that differs from this one only by a known linear function.

Exercise 4.4. Suppose that u solves

$$-u''(x) + r(x)u(x) = f(x) \quad \text{on } (0, 1),$$

and $u(0) = u(1) = 0$. Let ρ be such $r(x) \geqslant \rho > 0$, and define

$$C = \max_{0 \leqslant x \leqslant 1} |f(x)|/\rho.$$

Prove that $u(x) \leqslant C$.

4.2 The variational formulation

4.2.1 The idea

The key sequence of ideas for FEMs is

(i) First replace the differential equation with an integral equation, using integration by parts to reduce the order of the derivatives. This is called the "*variational*" or "*weak*" form of the equation. (Details are below).

(ii) If we had a candidate for the true solution to the differential equation, we could trial it by substituting it back into the BVP.

(iii) But the set of possible solutions is infinitely large, so we can't check them all.

(iv) So we choose a much smaller subset, and look for the solution there. The space we will use is the space of *piecewise linear splines*, from Section 2.1.

(v) For every value of the spline that we have to determine, we write down a version of the integral equation that must be satisfied.

(vi) This will give us a linear system of equations to solve.

(vii) A simple but clever idea shows that the approximation we find is the best possible one.

4.2.2 Where the solution lives

The boundary value problem (4.2) above equates the terms in u and u'' with a function f that is continuous on (a, b), so we must have that u, u' and u'' are all continuous on the interval (a, b). That is, u belongs to the space of functions $C^2(a, b)$.

So we can state the problem (4.2) more precisely: *find* $u \in C^2(a, b)$ *such that*

$$-u''(x) + r(x)u(x) = f(x) \quad \textit{for all } x \textit{ in } (a, b),$$

$$\textit{and } u(a) = u(b) = 0.$$

There are (uncountably) many functions in the space $C^2(a, b)$ (we say it is *infinite dimensional*) – far too many to check one-by-one. So we choose small subset of $C^2(a, b)$ that has only finitely many members. The FEM will let us find the element of that subset that is "closest" to the true solution.

4.2.3 The variational/weak form

Before we give a numerical method for computing an approximate solution to a boundary value problem, we will rewrite it as an integral equation.

Definition 4.2.1. We define

- $C_0^2(a, b)$ to be the space of all functions in $C^2(a, b)$ that are zero at $x = a$ and $x = b$.
- the inner product: $(u, v) := \int_a^b u(x)v(x)dx$; and the induced norm: $\|u\|_2 := (u, u)^{1/2}$,
- and $H_0^1(a, b)$ to be the space of (absolutely) continuous functions on $[a, b]$ such that if $w \in H_0^1(a, b)$ then $w(a) = w(b) = 0$ and $\|w'\|_2 < \infty$. (We met similar space before in the section on cubic splines.)

Consider the Boundary Value Problem: *find* $u \in C^2(a, b)$ *such that*

$$\begin{aligned} -u''(x) + r(x)u(x) = f(x) \quad & \textit{on } (a, b), \\ u(a) = u(b) = 0. & \end{aligned} \tag{4.4}$$

Suppose that we have a solution u to this. Then,... Take notes:

Restrict the set of possible v so that $v(a) = v(b) = 0$ to get another way of stating (4.4): *Find* u *so that*

$$(u', v') + (ru, v) = (f, v)$$

$$\text{for all } v \text{ such that } v(a) = v(b) = 0.$$

However, in (4.4) we required that u be twice differentiable; that seems unnecessary here where we have the much *weaker* requirements that u' exist and be integrable.[1]

[1] Actually, a more correct interpretation of the expression *weak*" formulation is that it does not necessarily involve classical "strong" derivatives, but a more general concept of the weak derivative based on Lebesgue integration. But this is beyond the scope of this course, and so we will ignore this technicality.

Definition 4.2.2 (Variational formulation)**.** The weak/-variational formulation of (4.4) is: *Find* $u \in H_0^1(a,b)$ *such that*

$$A(u,v) = L(v) \quad \textit{for all} \quad v \in H_0^1(a,b). \tag{4.5}$$

where $\mathcal{A}(\cdot,\cdot)$ is the (symmetric) bilinear functional

$$\mathcal{A}(u,v) := (u',v') + (ru,v),$$

and $L(v)$ is the linear functional

$$L(v) = (f,v).$$

This $\mathcal{A}$ has several important properties. One of these is that, for any function w,

$$A(w,w) \geqslant 0,$$

and

$$A(w,w) = 0 \iff w = 0.$$

In immediate consequence of this is the following lemma.

Lemma 4.2.3. *The variational problem* (4.5) *has at most one solution.*

Take notes:

4.2.4 Exercise

Exercise 4.5. $\star$ Consider the differential equation:

$$-u''(x) = \exp(x+1), \text{ on } (0,2), \text{ and } u(0) = u(2) = 0.$$

(i) State the variational formulation of this differential equation.

(ii) Show that the solution to the variational problem is unique.

4.3 The FEM

4.3.1 On Infinite- and Finite-Dimensional Spaces

A *space* is a collection of functions. The *dimension* of that space is smallest number of pieces of information that is required to uniquely describe any of its members.

Example 4.3.1. The set of points in $\mathbb{R}^2$. Each member can be written as (x_1, x_2) so we need two pieces of information ("coordinates") to describe the point. So $\mathbb{R}^2$ has dimension 2.

Example 4.3.2. The space of quadratic polynomials (over the reals). Each member can be written as $a + bx + cx^2$. So we need to know a, b and c for each member of the set; the space has dimension 3.

Another way of thinking about this is considering that these space have certain members that can be combined is various ways to give us every other member; they form a *basis* for the space.
Examples of bases for $\mathbb{R}^2$ and $\mathcal{P}^2$:
Take notes:

Example 4.3.3. All the solutions to the differential equation $u''(x) + u(x) = 0$ (note: no boundary conditions) can be written in the form $u(x) = \alpha \sin(x) + \beta \cos(x)$. So $\{\sin(x), \cos(x)\}$ is a basis for this space which is of dimension 2.

All of the spaces mentioned have *finite-dimension*. But consider the space of *all continuous functions on* $[a, b]$. This is an *infinite dimensional space*: we would have to write down an infinite number of values to describe a member uniquely.

Example 4.3.4. The space of functions $C^2(a, b)$ is infinite-dimensional, as is $C_0^2(a, b)$. So too is $H_0^1(a, b)$.

Example 4.3.5. The space of functions that are of the form $g(x) = \gamma(x - a)(x - b)$ for any $\gamma \in \mathbb{R}$ is *finite-dimensional* (because it has dimension 1). But every member of this space also belongs to the infinite dimensional space $H_0^1(a, b)$; it is a *finite-dimensional subspace of* $H_0^1(a, b)$.

4.3.2 The Galerkin Basis Functions

Example 4.3.6. To get another **very important** example of a finite dimensional subspace of $H_0^1(a, b)$, first fix a "*mesh*" on $[a, b]$. This is just a set of points $\{a = x_0 < x_1 < x_2 < \cdots < x_n = b\}$. Then consider the space of all functions that are *piecewise linear* on this mesh and that vanish at $x = a$ and $x = b$.

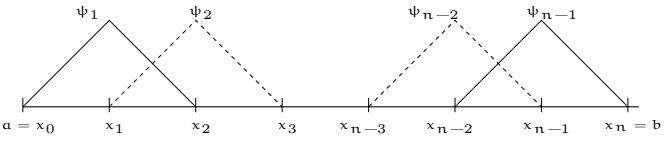
This is a a finite-dimensional sub-space of $H_0^1(a, b)$. A reasonable basis for this space would be the hat functions $\{\psi_1, \psi_2, \ldots, \psi_{n-1}\}$ given by

$$\psi_i(x) = \begin{cases} (x - x_{i-1})/h & x_{i-1} \leqslant x < x_i \\ (x_{i+1} - x)/h & x_i \leqslant x \leqslant x_{i+1} \\ 0 & \text{otherwise.} \end{cases}$$

where $h = (b - a)/n$ is the distance between adjacent points. Then we can write any function u_h as

$$u_h(x) = \lambda_1 \psi_1(x) + \lambda_2 \psi_2(x) + \cdots + \lambda_{n-1} \psi_{n-1}(x).$$

This basis set, shown below, are often called *hat functions* or *Galerkin*[2] *Basis functions*. We met them before in Section 2.1 on piecewise linear interpolation.



4.3.3 The Discrete Variational Form

Recall again that we are trying to solve the problem: *find* $u \in C^2(a, b)$

$$-u''(x) + r(x)u(x) = f(x) \quad \textit{on } (a, b),$$
$$u(a) = u(b) = 0,$$

where $r(x) > 0$ *for all* $x \in (a, b)$. We defined $\mathcal{A}(u, v) := (u', v') + (ru, v)$ and $L(v) := (f, v)$, and wrote the weak form of the DE as: *Find* $u \in H_0^1(a, b)$ *such that*

$$\mathcal{A}(u, v) = L(v) \quad \text{for all} \quad v \in H_0^1(a, b).$$

Now imagine we were trying to solve it by taking a function from $H_0^1(a, b)$ and checking if this integral equation is true for all functions v in $H_0^1(a, b)$. This would take forever because there are an infinite number of candidates. So instead we restrict our attention to a *finite-dimensional* subspace S of $H_0^1(a, b)$. Now we can select a function u_h from S and "put it on trial" by testing it against (all) the functions v_h in S. This leads to the terminology of calling u_h a *trial* function and v_h a *test* function.

Moreover, it leads to the following method.

[2] Born, 4 March 1871 in Polotsk, Belarus. Died 12 June 1945 in Moscow.

Definition 4.3.7 (The Finite Element Method)**.** Let S be the finite dimensional subspace of $H_0^1(a,b)$ made up of the piecewise linear functions on a fixed mesh $a = x_0 < x_1 < \cdots < x_n = b$. Then the *Galerkin Finite Element method* is: *find* $u_h \in S$ *such that*

$$\mathcal{A}(u_h, v_h) = (f, v_h) \qquad \textit{for all} \qquad v_h \in S. \quad (4.6)$$

4.3.4 FE implementation

We now want to look at how to turn Definition 4.3.7 into an algorithm (ideally, one that we can implement on a computer).

Let S be the space of piecewise linear functions on the mesh $x_i = a + ih$, where $h = (b-a)/n$. As above, u_h is can be written as

$$u_h(x) = \lambda_1\psi_1(x) + \lambda_2\psi_2(x) + \cdots + \lambda_{n-1}\psi_{n-1}(x).$$

So u_h has $n-1$ unknowns: $\lambda_1, \lambda_2, \ldots, \lambda_{n-1}$. To solve for these, we need $n-1$ equations. To get these, we just choose $n-1$ different (i.e., linearly independent) possible v_h, and substitute into (4.6).

The most obvious, and (it turns out) sensible, choice for these $n-1$ equations are the $n-1$ hat functions ψ_1, $\psi_2, \ldots, \psi_{n-1}$.

This gives us $n-1$ equations to solve:

$$\mathcal{A}(u_h, \psi_i) = (f, \psi_i) \qquad \text{for } i = 1, \ldots n-1. \quad (4.7)$$

It is now not too difficult to see that, if we write these equations as a matrix-vector equation, $Ax = F$, then

$$a_{i,j} = \mathcal{A}(\psi_i, \psi_j)$$

Take notes:

Some important observations:

- Any given "hat" function ψ_i is only non-zero on the region $[x_{i-1}, x_{i+1}]$.
- We have to compute

$$a_{i,j} = \mathcal{A}(\psi_i, \psi_j) = \int_a^b \psi_i'(x)\psi_j'(x) + r(x)\psi_i(x)\psi_j(x)dx. \quad (4.8)$$

But this will be non-zero only if there is overlap between $[x_{i-1}, x_{i+1}]$ and $[x_{j-1}, x_j+1]$.

- This gives that $A(\psi_i, \psi_j) = 0$ if $|i-j| > 1$, and otherwise

$$\mathcal{A}(\psi_i, \psi_j) = \int_{x_{i-1}}^{x_{i+1}} \psi_i'(x)\psi_j'(x) + r(x)\psi_i(x)\psi_j(x)dx.$$

Such a system is called *tridiagonal*. The left-hand side looks like this:

$$\begin{pmatrix} \mathcal{A}(\psi_1,\psi_1) & \mathcal{A}(\psi_2,\psi_1) & 0 & 0 & \cdots & 0 \\ \mathcal{A}(\psi_1,\psi_2) & \mathcal{A}(\psi_2,\psi_2) & \mathcal{A}(\psi_3,\psi_2) & 0 & \cdots & 0 \\ 0 & \mathcal{A}(\psi_3,\psi_2) & \mathcal{A}(\psi_3,\psi_3) & \mathcal{A}(\psi_3,\psi_4) & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \cdots & \vdots \end{pmatrix}$$

There are two consequences to this:

(i) It takes less effort to set up the linear systems of equations than one might have thought.

(ii) It is relatively easy to solve.

We should also observe the the matrix, A, is **symmetric**.

Note: in general a quadrature rule is used to compute in integrals

$$\int_{x_{i-1}}^{x_{i+1}} r(x)\psi_i(x)\psi_j(x).$$

The Gaussian Quadrature methods are the most popular for this.

An example

Use the FEM on the mesh $\{0,1,2,3\}$ *to find an approximate solution to*

$$-u'' + 3u = x \quad \textit{on} \quad (0,3), \quad u(0) = u(3) = 0. \quad (4.9)$$

Solution: The FEM is: *Find* $u_h(x) \in S$ *such that*

$$\mathcal{A}(u_h, v_h) := (u_h', v_h') + 3(u_h, v_h) = (x, u_h) \qquad \textit{for all } v_h(x) \in S.$$

We have that $h = 1$ so let

$$\psi_1(x) = \begin{cases} x & 0 \leqslant x < 1 \\ 2-x & 1 \leqslant x \leqslant 2 \\ 0 & \text{otherwise,} \end{cases}$$

$$\psi_2(x) = \begin{cases} x-1 & 1 \leqslant x < 2 \\ 3-x & 2 \leqslant x \leqslant 3 \\ 0 & \text{otherwise,} \end{cases}$$

and

$$u_h(x) = \lambda_1\psi_1(x) + \lambda_2\psi_2(x).$$

Our two equations are:

$$(\lambda_1\psi_1' + \lambda_2\psi_2', \psi_1') + 3(\lambda_1\psi_1 + \lambda_2\psi_2, \psi_1) = (x, \psi_1),$$
$$(\lambda_1\psi_1' + \lambda_2\psi_2', \psi_2') + 3(\lambda_1\psi_1 + \lambda_2\psi_2, \psi_2) = (x, \psi_2).$$

giving

$$\lambda_1\left(\int_0^3 \psi_1'\psi_1'\,dx + 3\int_0^3 \psi_1\psi_1\,dx\right) + \lambda_2\left(\int_0^3 \psi_2'\psi_1'\,dx + 3\int_0^3 \psi_2\psi_1\,dx\right) = \int_0^3 x\psi_1\,dx$$

$$\lambda_1\left(\int_0^3 \psi_1'\psi_2' dx + 3\int_0^3 \psi_1\psi_2 dx\right) +$$
$$\lambda_2\left(\int_0^3 \psi_2'\psi_2' dx + 3\int_0^3 \psi_2\psi_2 dx\right) = \int_0^3 x\psi_1 dx.$$

We now need to evaluate these integrals. For example, from the 1st equation:

$$\int_0^3 \psi_1'\psi_1' dx = \int_0^1 (1)^2 dx + \int_1^2 (-1)^2 dx = 2,$$
$$\int_0^3 \psi_1\psi_1 dx = \int_0^1 x^2 dx + \int_1^2 (2-x)^2 dx = 2/3,$$

so the coefficient of λ_1 is $2 + 3(2/3) = 4$. Also,

$$\int_0^3 \psi_2'\psi_1' dx = \int_1^2 \psi_2'\psi_1' dx = \int_1^2 (1)(-1) dx = -1,$$
$$\int_0^3 \psi_2\psi_1 dx = \int_1^2 \psi_2\psi_1 dx = \int_1^2 (x-1)(2-x) dx = 1/6.$$

So the coefficient of λ_2 is $-1 + 3(1/6) = -1/2$.

For the right-hand side:

$$\int_0^3 x\psi_1 dx = \int_0^1 (x)(x) dx + \int_1^2 (x)(2-x) dx = 1/3 + 2/3 = 1.$$

Similarly,

$$\int_0^3 x\psi_2 dx = 2.$$

The final system is

$$4\lambda_1 - \frac{1}{2}\lambda_2 = 1, \qquad -\frac{1}{2}\lambda_1 + 4\lambda_2 = 2.$$

This can also be written as a matrix-vector equation:

$$\begin{pmatrix} 4 & -\frac{1}{2} \\ -\frac{1}{2} & 4 \end{pmatrix}\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Solving, we get $\lambda_1 = 20/63$ and $\lambda_2 = 34/63$. The solution is $u^h(x) = (20/63)\psi_1(x) + (34/63)\psi_2(x)$. This is shown in Figure 4.1

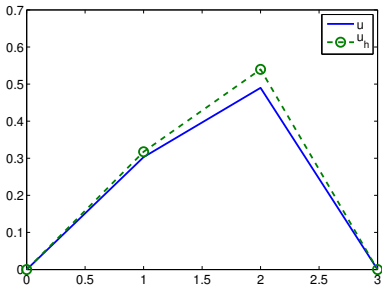


Fig. 4.1: The FE solution to (4.9) with $n = 3$

4.3.5 Exercises

Exercise 4.6. Show that u_h solves (4.7) *if and only if* it solves (4.6).

Exercise 4.7. Consider the problem:

$$-u''(x) = 9x \qquad u(0) = 0, u(1) = 0.$$

Use the FEM to find an approximate solution on the mesh $\{0, 1/3, 2/3, 1\}$.

Also write down the true solution to this problem.

Exercise 4.8. Suppose we want to use a finite element method to solve

$$-u''(x) + u(x) = 1 \text{ on } (0,1),$$

with $u(0) = u(1) = 0$, using the usual piecewise linear basis functions on on uniform mesh $\{x_0, x_1, \ldots, x_n\}$. Let the resulting linear system is written as the matrix-vector equation $Au_h = F$.

(i) Show that the matrix A is symmetric (i.e. $a_{ij} = a_{ji}$).

(ii) Show that A is tridiagonal (i.e., if $|i-j| > 1$ then $a_{ij} = 0$).

(iii) Derive the formula for the entries of A in terms of h. That is, give an expression for $a_{i,i-1}$, $a_{i,i}$ and $a_{i,i+1}$.

4.4 Error analysis

4.4.1 Cea's Lemma

We now show that the member of S found by the FEM is the "closest" to the true solution.

Lemma 4.4.1 (Cea's Lemma)**.** *Let u be the solution to (4.5) and let u_h be the solution to (4.6).*

(i) The difference between the true and approximate solutions is orthogonal to S, i.e.,

$$\mathcal{A}(u - u_h, v_h) = 0 \text{ for all } v_h \in S,$$

and

(ii) There is no element of S that is closer to u than u_h:

$$\mathcal{A}(u - u_h, u - u_h) = \min_{v_h \in S} \mathcal{A}(u - v_h, u - v_h),$$

Proof. (See also Theorem 14.6 of [SM03])
Take notes:

□

Since $\mathcal{A}(\cdot,\cdot)$ is an inner product (see Definition 3.5.2) it induces a *norm:*

$$|||u||| := \sqrt{\mathcal{A}(u,u)}.$$

So we can write (ii) of Cea's Lemma as

$$|||u - u_h||| \leqslant |||u - v_h||| \qquad \text{for all } v_h \in S.$$

4.4.2 An example

This is as far as we will take the analysis. With a bit more work (and a little Fourier analysis) we could show that

$$\|u - u_h\|_2 \leqslant Ch^2 \|u''\|_2.$$

That is, the error is proportional to h^2. We can then further deduce that the method converges:

$$\lim_{h \to 0} \|u - u_h\|_2 = 0.$$

In place of a rigorous analysis, let us reason as follows. Let l be the piecewise linear interpolant to u as described in Section 2.1). Note that l belongs to S. So, u_h as at least as good an approximation to u as l. That is

$$\|u - u_h\|_2 \leqslant \|u - l\|_2$$

And Theorem 2.1.3 told us that

$$\|u - l\|_\infty \leqslant \frac{h^2}{8} \|u''\|_\infty.$$

So, if you believe that

$$\|u - l\|_2 \approx \|u - l\|_\infty,$$

Then it will follow that

$$\|u - u_h\|_2 \lessapprox Ch^2,$$

for some constant C. One can also show that

$$|||u - u_h||| \lessapprox Ch.$$

However that we have used three different norms here. Therefore some more work would be required to prove a rigorous result.

In Table 4.1 we shown the maximum error, over all mesh points, in the finite element solution to (4.9). One can see that the error is proportional to n^{-2} (and thus to h^2). See also Figure 4.2

These results were generated by a MATLAB program, which you can download from www.maths.nuigalway.ie/ niall/MA378/fe.m

Table 4.1: $\|u - u_h\|_\infty$ when solving (4.9)

n	$\|u - u_h\|_\infty$
4	2.908e-02
8	6.446e-03
16	1.629e-03
32	4.043e-04
64	1.009e-04
128	2.522e-05
256	6.304e-06
512	1.576e-06
1024	3.940e-07

4.4.3 Exercises

Exercise 4.9. Suppose that we want to solve

$$-u''(x) + u'(x) = 1 \text{ on } (a,b),$$

(a) Write down the system of linear equations that we would have to solve in terms of h.

(b) Does the analysis of Theorem 4.4.1 still hold? That is, can we use a similar argument to show that

$$|||u - u_h||| \leqslant |||u - v_h||| \quad \text{for all } v_h \in S?$$

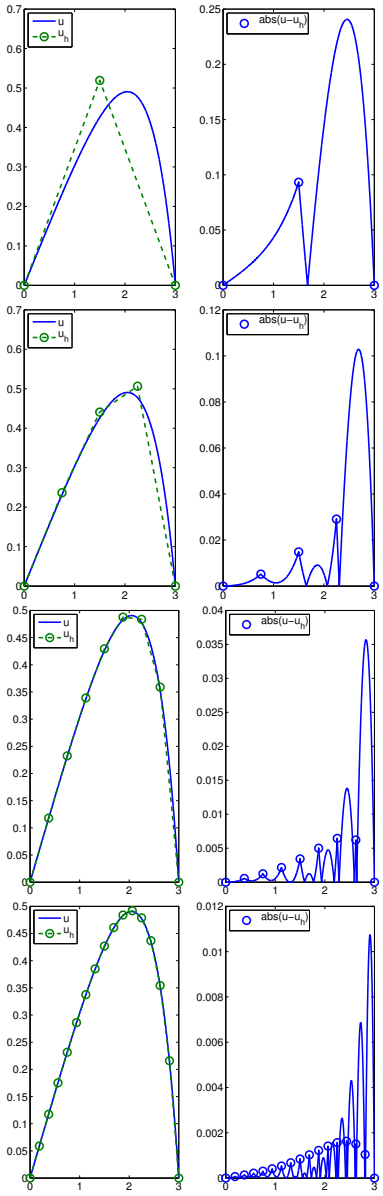


Fig. 4.2: Computed solutions and errors solving (4.9) with (from top) $n = 2, 4, 8, 16$

Exercise 4.10. Show that, for any function $f \in C^2[a, b]$,

$$\|f\|_2 \leqslant \sqrt{b-a}\|f\|_\infty,$$

where

$$\|f\|_2 := \left(\int_a^b (f(x))^2 dx\right)^{1/2} = \sqrt{(f, f)},$$

and

$$\|f\|_\infty := \max_{a \leqslant x \leqslant b} |f(x)|.$$

Exercise 4.10 shows that if we have a bound for $\|f\|_\infty$, we can get one for $\|f\|_2$. However, as the next exercise shows, the converse is not true.

Exercise 4.11. Show that, given any $\epsilon > 0$, no matter how small, it is possible to construct a function $f \in C^2[a, b]$, for which

$$\|f\|_2 \leqslant \varepsilon$$

but

$$\|f\|_\infty = 1.$$

4.5 FE Wrap-Up

There are many aspects of finite element methods that we did not cover, including

1. Finite element methods are *the most commonly used methods* for solving problems formulated as differential equations.
2. There are many other choices of basis functions given here. One could used cubic splines, or, indeed, higher-order polynomials.
3. When we try to improve the accuracy of the method by reducing h, this is called a h-FEM (and is the most common type).
4. We can also try to improve the accuracy of the method by increasing the order of the polynomials. This is called a p-FEM.
5. The ideas presented here extend to far more general problems. In particular, they work very well for problems in higher dimensions, and on weird-shaped domains.

The end!

I hope you enjoyed the course and now feel confident in your abilities as a Numerical Mathematician. Remember, if you even need to approximate a function, estimate an integral, derive a discrete derivative, or find a numerical solution to a boundary value differential equation, just reach for the nearest polynomial.

NM. March 2017