# Contents

1 Introduction ........................................ 2
  1.1 Instructor ........................................... 2
  1.2 Time and place ..................................... 2
  1.3 Textbook ............................................. 2
  1.4 Software ............................................. 3
    1.4.1 Mathematical Software ......................... 3
    1.4.2 Word Processing Software ..................... 4
  1.5 Class procedures ................................... 4
    1.5.1 Attendance ...................................... 4
    1.5.2 Lectures ........................................ 4
    1.5.3 Homework ....................................... 4
    1.5.4 Projects ....................................... 5
    1.5.5 Examinations ................................... 5
    1.5.6 Grading ......................................... 5
    1.5.7 Other Rules and Regulations ................... 5
  1.6 Schedule ............................................. 6

2 Assignments: .......................................... 7
  2.1 Due February 14 ..................................... 7
  2.2 Due February 21 ..................................... 7
  2.3 Due February 28 ..................................... 7
  2.4 Due March 7 ......................................... 7
  2.5 Due March 14 ........................................ 8
  2.6 Due March 28 ........................................ 8

3 Projects .................................................. 9
  3.1 Gauss-Kronrod Integration .......................... 9
  3.2 Genetic Algorithms ................................ 9
  3.3 The Wave Equation .................................. 9
  3.4 Matrix Exponentials ................................ 9
  3.5 Solving Equations .................................. 10
  3.6 Calculator Algorithms ............................... 10
  3.7 Boundary Value Problems ......................... 10

4 Introduction to Numerical Analysis ..................... 11

5 Polynomial Interpolation ............................... 14
  5.1 The General Theory using Lagrange Interpolation .. 14
  5.2 Newton Interpolation ................................ 17
    5.2.1 Divided Differences ............................ 18
Chapter 1  Introduction

Numerical analysis is the science of finding approximate solutions to mathematics problems and bounding the error in the approximation. Both the approximation and the error bound are always required; otherwise you couldn’t know if the answer was any good. Bonus points are awarded for algorithms that go faster or are guaranteed to eventually converge to the right answer.

Numerical analysis is important because most mathematics problems arising in science, engineering and business cannot be solved exactly. Even when exact solutions are available, they are sometimes too expensive when a satisfactory approximate solution will suffice.

The invention of useful numerical approximations precedes Newton, but it was the invention of the digital computer that made numerical analysis a key economic engine. Combining computers with clever algorithms has made possible the solution of a huge number of engineering problems, and the solutions are often embedded in electronic devices you hold in your hands. For example algorithms for rapid signal analysis are used every time you make a cell phone call or view an image on your computer.

Numerical analysis has suffered some spectacular failures. See

http://www.ima.umn.edu/~arnold/disasters/disasters.html

and

http://www.cs.clemson.edu/~steve/Spiro/stories.htm

. Let these be a lesson to you. Program carefully, put in lots of consistency checks, and don’t always assume that a standard algorithm will work on your problem.

This course will teach you a few of the thousands of useful algorithms that have been developed. Hopefully you will gain confidence in your ability to find and use numerical algorithms for whatever problems you encounter in the future.

1.1  Instructor

David Meredith
Office: TH 937
Phone: 415.338.2251
E-mail: meredith@sfsu.edu
URL: http://online.sfsu.edu/~meredith

Also: this course uses SFSU Blackboard for distribution of course documents and assignments, discussion boards and gradebook.

1.2  Time and place

MWF 14:10-15:00, TH 211

1.3  Textbook

Our official text is Scheid, Numerical Analysis (Second Edition) from Schaum’s Outlines. If you want even more explanation, all the topics we cover are included in all the standard $100 numerical analysis textbooks. The most popular is by Burden and
Faires. For a long-term investment, you might want to check out Press, et.al, Numerical Recipes from Cambridge University Press. This popular handbook is very light on theory, but it has hundreds of useful algorithms already coded for you. For a good overview of numerical analysis theory, try Higham, Accuracy and Stability of Numerical Algorithms (SIAM) or Süli and Mayers, An Introduction to Numerical Analysis (Cambridge U. Press).

1.4 Software

1.4.1 Mathematical Software

In this course you will write computer programs to calculate mathematical results. There are several programs you can use.

1.4.1.1 Mathematica

Mathematica computes numerically and symbolically. You can calculate \( \sum_{i=1}^{50} \frac{1}{i^2} \) or \( \frac{d}{dx} e^{3x} \cos 4x \). With Mathematica, you can work interactively or you can write subroutines. The Math Dept. at SFSU has decided to use this program in many of its courses. Mathematica is freely available M-F, 10-3, in the Math/Stat Computing Lab, TH404. Buying your own copy is not necessary, but it might be a good investment. If you learn to use Mathematica effectively, you will find it very valuable in all your science, engineering and math courses. The Mathematics Department will sell you a copy for $60 that must be renewed at the end of 2000; the bookstore will sell you a copy for $150 that lasts forever.

1.4.1.2 Maple

Maple is a symbolic and numerical powerhouse similar to Mathematica but not as widely used outside of academia.

1.4.1.3 Matlab

Matlab is the classic linear algebra program with the best numerical algorithms, and it is the software preferred by professional numerical analysts. A student version is available at the bookstore. If you are going into engineering or applied mathematics, it might pay to learn this program. Matlab now includes Maple, so some symbolic calculations can be performed with Matlab. If your focus is symbolic work, Mathematica or Maple is a better choice.

1.4.1.4 X(PLORE)

Truth in advertising: I wrote X(PLORE). It is freely available from my web site and in the Math/Stat Computing Lab. You can do most of the calculations for this course in X(PLORE). The advantage is the cost; the disadvantage is the lack of symbolic computation.

1.4.1.5 C++

You can write all the algorithms for this course in C++. The advantage is that your programs may run faster and you will be reinforcing an economically valuable skill. The disadvantage is that you will not have access to the advanced mathematical operators available in a more mathematical programming language like Mathematica or Matlab, and you will have difficulty illustrating your results graphically.
1.4.2 Word Processing Software

You can write mathematical reports with *Mathematica*, but not easily. Moreover, the result is ugly. Much better tools are any standard word processor with equation-writing capability (*Word* or *WordPerfect* will do nicely, but you may have to install the equation module for *Word*) or a special scientific work processor like *Scientific WorkPlace* ($600) or *Scientific Notebook*($110). I highly recommend *Scientific WorkPlace* and *Scientific Notebook*. They include MuPad, symbolic and numerical mathematical software, so you can do simple computations inside the word processor. The output looks good, much better than the output from either *Mathematica* or *Word*. Sometimes the bookstore stocks *Scientific Notebook*. More information about these products can be found at http://www.mackichan.com.

This document was produced with *Scientific WorkPlace*.

1.5 Class procedures

1.5.1 Attendance

I take attendance every day.

1.5.2 Lectures

I will lecture, more or less, three times a week. I will always stop for questions, but I will also assume that if you don’t ask any questions then you understand perfectly everything I’m saying. Since that’s unlikely to be true, you should be asking lots of questions in class.

1.5.3 Homework

Homework is the most important component of this class. You could read the book instead of listening to lectures, but there is no substitute for doing problems yourself. Problems will be of two types: theory and practice. Theory problems are paper-and-pencil problems where you analyze some aspect of an algorithm; practice problems are calculations to be done on a computer.

Each homework problem is to be a mini-essay written in correct mathematical English and correctly typeset and printed. Homework which does not meet this standard will not be graded. Begin with an explanation of the problem you will solve or the calculation you will make. Then give your results. Theory problems must include an explanation or proof; calculations must include an error bound and an explanation of how you arrived at your answer and your error bound. If you need to include computer code as part of your answer, it should usually be attached after your essay as an appendix, and it must be carefully formatted and copiously commented so as to be readable.

Do not just copy the question at the beginning of your paper; begin by restating your goal in positive terms as something you will do. If the problem asks: "Find the smallest positive root for the polynomial ..."; you start with something like: "Consider the polynomial.... We will find the smallest positive root of this polynomial and find an error bound for our calculation."

The general criterion for an acceptable homework assignment is this: anyone with a knowledge of numerical analysis should be able to read your paper and know what problem you were solving, how you solved it, and why your answer is valid. You can refer to theorems in our text or class notes, but beyond that your paper should be self-contained and readable by someone who has never seen the assignment you are doing.

There is a reason for the homework requirements. More important even than learning the methods of numerical analysis is learning how to communicate your tech-
nical ideas to another person. I’m fairly tolerant of small grammatical errors, but
I do insist on electronically printed papers written in complete sentences with clear
explanations.

1.5.4 Projects

Each student will do a term project using numerical analysis. Projects may be com-
pleted by teams of up to three students (but no more). More information about the
projects and some suggested topics are printed below.

1.5.5 Examinations

We will have one midterm and a final exam. Test questions will consists of theory
questions and questions asking you to compare the advantages and disadvantages of
different algorithms for the same problem. You may use one page of notes for the
midterm and two pages for the final.

1.5.6 Grading

<table>
<thead>
<tr>
<th>Grading System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attendance</td>
</tr>
<tr>
<td>Homework</td>
</tr>
<tr>
<td>Projects</td>
</tr>
<tr>
<td>Midterm</td>
</tr>
<tr>
<td>Final Exam</td>
</tr>
</tbody>
</table>

Final grades will be assigned according to a scale no harsher than: \( A \geq 85\% \),
\( B \geq 70\% \), \( C \geq 55\% \).

1.5.7 Other Rules and Regulations

Here is a short version of the University calendar for Fall, 2004. Note that the Math-
ematics Department strictly enforces the deadlines for CR/NCR grading and with-
drawals.

February 11    Last day to add classes
February 25    Last day to drop classes online
March 21-25    Spring Break
March 31       Cesar Chavez Holiday
April 1        Last day to select CR/NCR grading
April 29       Last day to withdraw from a course
May 5          Advising Day. Classes cancelled
May 20         Last day of instruction
May 23-27      Final exams
June 3         Grades due from instructors

1.5.7.1 CR/NCR Grading

Most Mathematics classes allow CR/NCR grading, but many majors—including Mathematics—
do not count CR/NCR grades towards the major. Mathematics majors should not
take their Mathematics classes CR/NCR. All other majors should check with their
academic advisors before deciding to take a Mathematics class CR/NCR.

If—after checking with your advisor—you want to apply for CR/NCR grading, you
must log onto the web site www.sfsu.edu/student and sign up for CR/NCR grading
before the April 1 deadline. I will not pass out a CR/NCR sheet in class.
1.5.7.2 Incompletes

The Incomplete grade (I) is assigned only to students doing satisfactory work until the last few weeks of a course, when events beyond the students’ control prevented them from completing the course. If this happens to you, discuss with your instructor the possibility of taking an Incomplete rather than withdrawing from a class that you cannot finish.

Incompletes must be made up within twelve months of the date they are assigned. Your instructor will tell you how to make up your incomplete. Do not enroll in the same course again. You can only take a course once.

1.5.7.3 Late and Retroactive Withdrawals

Petitions for withdrawal from a class after the November 15 deadline, either before the end of the semester (late withdrawal) or after the semester ends (retroactive withdrawal) must be justified by events that occurred after the deadline. In general, only petitions for withdrawal from all courses will be approved. Late withdrawal from your math course alone is usually not approved.

1.5.7.4 Students with Disabilities

Students with disabilities needing reasonable accommodations must bring an official written request to their instructor from the Disability Programs and Resource Center (Student Services Building, Room 110, (415) 338-1041, drc@sfsu.edu). The DPRC is available to facilitate the reasonable accommodations process.

1.5.7.5 Religious Holidays

Reasonable accommodations will be made for you to observe religious holidays when such observances require you to be absent from class activities. It is your responsibility to inform the instructor during the first two weeks of class, in writing, about such holidays.

1.6 Schedule

January 31: Introduction to Numerical Analysis
February 7: Interpolating Polynomials
February 14: Newton’s Algorithm for Interpolating Polynomials
February 21: Cubic Splines
February 28: Numerical Integration
March 7: Gaussian Integration
March 14: Singular Integrals
March 21: Spring Break
March 28: Midterm Wednesday, March 30.
April 4: Equation Solving, Local Methods
April 11: Equation Solving, Global Methods
April 18: Minimizing Functions of Several Variables
April 25: Ordinary Differential Equations—Runge-Kutta methods
May 2: Ordinary Differential Equations—Other methods
May 9: Floating Point Numbers
May 16: Project Reports
May 23: Final Exam Monday, May 23, 1:30-4:00, TH 211
Chapter 2  Assignments:

2.1 Due February 14

Using only arithmetic operations (+, −, ∗, /), evaluate $e^{-100}$. (Hint: $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$, truncating this series leads to an approximate value of $e^x$. Also remember the algebraic properties of $e^x$: $e^{-x} = \frac{1}{e^x}$, $e^{x+y} = e^x e^y$, $e^{xy} = (e^x)^y$; etc.) With sufficient cleverness you can do this problem on a calculator, but feel free to write computer code or use a calculating program if you want. Don’t forget to say how accurate you think your answer is and why.

2.2 Due February 21

1. How many arithmetic operations does Newton’s algorithm use to calculate the coefficients of an interpolating polynomial through $n+1$ points?

2. We’ve seen in class that polynomial approximations with evenly spaced points converge to the function $\frac{1}{1+x^2}$ on the domain $[-1,1]$ but do not converge on $[-3,3]$. Determine experimentally if polynomial approximations converge to $e^{-x^2}$ on $[-2,2]$.

2.3 Due February 28

Fit a natural cubic spline to the function $f(x) = \frac{1}{1+x^2}$ using an even number of segments (at least four) on the interval $[-3,3]$. Let $g(x)$ be the cubic spline. Compare the functions $f(x)$ and $g(x)$ on the interval $[-3,3]$ three ways:

1. Find the maximum difference between $f(x)$ and $g(x)$.

2. Evaluate $\int_{-3}^{3} (f(x) - g(x))^2 \, dx$

3. Invent your own "goodness of fit" test, explain it and apply it.

2.4 Due March 7

1. Write a program for Simpson’s method and use it to evaluate $\int_{0}^{10} \sqrt{1+\sin^4 x} \, dx$ to an accuracy of $\pm 10^{-3}$. Explain why you believe that you have achieved the desired accuracy.

2. Write a program for Romberg’s method based on the midpoint approximation and use it to evaluate $\int_{0}^{10} \sqrt{1+\sin^4 x} \, dx$ to an accuracy of $\pm 10^{-3}$. Explain why you believe that you have achieved the desired accuracy.
3. Analyze Simpson’s Three-Eighths Rule. In this method the number of weights is 4

\begin{align*}
  r &= 3 \\
  z_0 &= x_i \\
  z_1 &= \frac{2}{3} x_i + \frac{1}{3} x_{i+1} \\
  z_2 &= \frac{1}{3} x_i + \frac{2}{3} x_{i+1} \\
  z_3 &= x_{i+1} \\
  w_0 &= \frac{1}{8} = w_3 \\
  w_1 &= \frac{3}{8} = w_2
\end{align*}

Find which polynomials this method integrates exactly, and conclude that \( \int_a^b f(x) \, dx = ST(f,a,b,n) + O(\Delta x)^r \) where \( ST \) stands for Simpson’s Three-Eighths Method.

2.5 Due March 14

Use Gaussian quadrature to evaluate \( \int_0^20 \frac{\sin x}{x} \, dx \) to within \( 10^{-3} \). Use the Gauss error estimate to determine the accuracy of your answer. Be sure to tell how you got the Gaussian points and weights that you used.

2.6 Due March 28

Estimate \( \int_0^\infty \frac{\sin x}{x} \, dx \). Your answer will be of the form \( a \pm b \); points are awarded for a correct answer, for a small value of \( b \); and for a clear explanation of how you arrived at your estimate.
Chapter 3 Projects

For all projects, you should do a web search on the topic and a search of Math Reviews or some other bibliography of applied mathematics to find out what is known about your problem. Math Reviews is available through the SFSU Library. Remember, your report has to include a bibliography of at least two items besides the class textbook.

Projects may be completed by teams of up to three students.

Each project has to be on a different subject, so tell me when you are ready to start. You are welcome to suggest your own project in addition to the ones below, but check it with me before doing a lot of work.

3.1 Gauss-Kronrod Integration

Explain what Gauss-Kronrod integration is and how the coefficients are found. Test your methods on the integrals

\[ \int_0^1 e^x \sin (e^x) \, dx \]
\[ \int_1^2 e^x \sin (e^x) \, dx \]
\[ \int_2^3 e^x \sin (e^x) \, dx \]
\[ \int_3^4 e^x \sin (e^x) \, dx \]

and compare your results to the exact value of the integral. How can you use Gauss-Kronrod integration to accurately estimate the last integral?

3.2 Genetic Algorithms

Describe one or more genetic algorithms and use them to minimize functions. Pick some test functions to stress your algorithm. Report on how you tuned the parameters of the algorithm. Include a minimization of \[ \frac{x^2 + y^2}{100} + \sin^2 x + \cos^2 x, \] on the domain \([-5, 15] \times [-5, 15]\).

3.3 The Wave Equation

Use grid methods to solve \( f_{xx} + f_{yy} = 0 \) on a square with different boundary conditions. Graph your solutions. Discuss the accuracy of your methods.

3.4 Matrix Exponentials

3.5 Solving Equations

Develop a really good equation solver that uses a variety of methods to handle all sorts of equations. Decide on a user-interface, including what information the user must provide in addition to the equation. Discuss different kinds of difficult equations and explain how your solver handles them.

3.6 Calculator Algorithms

Report on the algorithms used to compute transcendental functions in calculators, and implement them in software. Discuss the underlying mathematics.

3.7 Boundary Value Problems

Report on the algorithms used to solve boundary value problems for second order ordinary differential equations.
Chapter 4  Introduction to Numerical Analysis

1. Go over syllabus

2. The goal of numerical analysis is always to calculate something useful and bound
the error

(a) Some things that are hard to compute but worthwhile
   i. Solution of large linear system
   ii. Eigenvalues of large matrices
   iii. Roots of polynomial
   iv. Multiple integral
   v. Solution to system of differential or partial differential equations

(b) Some calculations that you might think are settled, but are still important
   i. Even famous programs like Mathematica don’t always get them right
   ii. arithmetic, especially differences and comparisons
   iii. transcendental functions

   i. Matlab and Borland C++ also get it wrong; My HP calculator and
   Mupad get it right
   A. \( \ln(1.0000001) = 9.9999995000003 \times 10^{-8} \)
   B. Try your calculator: \( \ln(1.0000001) - 0.0000001 \) should be \(-4.999997 \times 10^{-15}\)

3. Here are some more examples for innocuous looking formulae that are hard to
   compute for some values of the variable

   (a) \( \frac{1 - \cos x}{x} \)

      i. Try a trig identity \( \frac{1 - \cos x}{x} = \frac{2(\sin \frac{x}{2})^2}{x} \)

   (b) \( \frac{e^x - 1}{x} \)

      i. Use power series expansion: \( \frac{e^x - 1}{x} = 1 + \frac{1}{2}x + \frac{1}{3!}x^2 + \cdots \)

4. What is the source of the error in these two examples: destructive subtraction
or subtraction of two nearly equal quantities

   (a) Another source of error are accumulated round-off error

      i. Correct rounding tends to average errors, but for example consistent
         truncation leads to a growing error

      ii. See stock market errors

   (b) Other sources of error are bad algorithms, bad hardware (rare now), mis-
matched code, unit errors, even bad data

5. So our first goal is to compute accurately, or at least to know how accurately
we are computing

\[
\text{In[144]}=: \text{1.00000000000001} = 1
\]

\[
\text{Out}[144]= \text{True}
\]
(a) Discuss first homework problem

6. Our second goal is speed

(a) From visiting professor Peter Ho in CS:

The project goal is to estimate the level of waters in the Everglades Lake
in the South Florida, which is the second largest lake in US. ... It takes 45
days for one-year simulation. They need at least 30-year simulation.

(b) Some say it takes two days to predict tomorrow’s weather accurately

(c) Small speed gains come from better hardware; big speed gains from better
software

(d) Simple example: evaluate a polynomial, How many operations (+, −, ∗, /)

i. \(3x^2 - 2x + 5\)

ii. \(2x^5 - 3x^4 - 2x^3 + 4x^2 + 5x - 3\)

iii. \(a_nx^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0\)

(e) To be more efficient, we use (a special case of) Horner’s Method

i. \(3x^2 - 2x + 5 = (3x - x)x + 5\)

ii. \(2x^5 - 3x^4 - 2x^3 + 4x^2 + 5x - 3 = (((2x - 3) x - 2) x + 4) x + 5) x - 3\)

iii. \((\cdots ((a_nx + a_{n-1})x + a_{n-2}) \cdots + a_1)x + a_0\)

A. \(p = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0\)

B. Define \(p_n = a_n; p_i = p_{i+1}x + a_i\) for \(i = n - 1, \ldots, 0\). Then \(p_0 = p\).

C. This is a computer program

7. Let’s return to error issues. A problem is said to be ill conditioned if a small
change in the input makes a large change in the output.

\[\frac{d(output)}{d(input)}\]

(a) measure by the relative rate of change:

(b) Example: input is a small non-zero number \(\varepsilon\); the output is two numbers \(x\) and \(y\). The rule connecting the input to the output is:

\[x + y = 1\]

\[(1 + \varepsilon) x + y = 2\]

(c) The solution is:

\[x = \frac{1}{\varepsilon}\]

\[y = 1 - \frac{1}{\varepsilon}\]
(d) The derivatives are:

\[
\frac{dx/d\varepsilon}{x} = -\frac{1}{\varepsilon} \\
\frac{dy/d\varepsilon}{y} = \frac{1}{\varepsilon (\varepsilon - 1)} \approx \frac{-1}{\varepsilon}
\]

i. if \( \varepsilon \) is small then the relative rates of change are large.
ii. As the first coefficient goes from 1.001 to 1.002, a change of 0.1%, \( x \) goes from 1000 to 500, a change of 50%.

(e) Ill-conditioned problems are inherently hard to solve because a small approximation error at one point can introduce a huge change in the answer.

i. Analyze the relation between input and output like we did here.
ii. At least test for ill-conditioning by changing the input slightly and seeing if the answer changes by a lot.
Chapter 5  Polynomial Interpolation

5.1 The General Theory using Lagrange Interpolation

1. Two related problems

(a) Given points \((x_1, y_1), (x_2, y_2), \ldots, (x_{n+1}, y_{n+1})\), find a polynomial \(p(X)\) such that \(p(x_i) = y_i\) for \(i = 1, \ldots, n+1\).

i. Does such a polynomial always exist?
ii. How many such polynomials are there?

A. Is there a unique one of smallest degree?

iii. Remarks

A. Does such a polynomial always exist?

No. You can’t find a polynomial through \((1, 1)\) and \((1, 2)\) some additional conditions are needed on the points \((x_i, y_i)\) before the problem can be solved.

B. How many such polynomials are there?

If \(p(X)\) is a solution and \(\pi(X) = (X - x_1) \cdots (X - x_{n+1})\) and \(q(x)\) is any other polynomial, then \(p(X) + \pi(X)q(X)\) is also a solution, so there are lots of solutions if there is one.

Statement without proof: if \(p(X)\) is the solution of least degree, then all other solutions have the form given here. We prove this in modern algebra, Math 325.

C. Is there a unique one of smallest degree?

Yes provided all the \(x_i\) are distinct. We will prove this.

(b) Given a function \(f(X)\) defined on a closed interval \([a, b]\) and points \(a \leq x_1 < \cdots < x_{n+1} \leq b\), find a polynomial \(p(X)\) such that \(p(x_i) = f(x_i)\) for \(i = 1, \ldots, n+1\).

i. If the problem in part (a) can be solved, then this problem can also be solved.

ii. The uniqueness of the minimal-degree solution follows from the uniqueness of the minimal degree solution to (a).

iii. There remains the question: what is the maximum “error” \(|f(x) - p(x)|\) for \(x \in [a, b]\).

A. “Error” is too moral a word. “Difference” would be better.

B. There is no reason to think that just because \(f(X)\) and \(p(X)\) agree at some points they should be close at other points.

C. But the reason for finding \(p(X)\) is usually to get a polynomial approximation to \(f(X)\), and we want to know how good the approximation is. The difference between \(p(X)\) and \(f(X)\) is the error in the approximation.

D. We will derive a formula for the difference.

Theorem 1 Let \((x_i, y_i), i = 1, \ldots, n+1\) be points in \(\mathbb{R}^2\) with the \(x_i\) distinct. Then there exists a unique polynomial \(p(X) \in \mathbb{R}[X]\) such that \(p(x_i) = y_i, i = 1, \ldots, n+1\) and \(\deg(p(X)) \leq n\). We call \(p(X)\) the interpolating polynomial for the set of points \(\{(x_i, y_i)\}\).

Class restate theorem for case \(n = 0\) and check that it is true.

Class restate theorem for case \(n = 1\) and check that it is true. Why do we need \(\deg(p(X)) \leq n\) instead of \(\deg(p(X)) = n\)?

Proof. First we show that the solution \(p(X)\) exists. Then we will show that the solution is unique.
Define the Lagrange polynomial

\[ L_i (X) = \frac{(X-x_1) \cdots (X-x_i) \cdots (X-x_{n+1})}{(x_i-x_1) \cdots (x_i-x_i) \cdots (x_i-x_{n+1})} \]

Obviously:

\[ L_i (x_i) = 1 \]
\[ L_i (x_j) = 0 \text{ if } j \neq i \]
\[ \deg (L_i (X)) = n \]

Defining:

\[ p (X) = y_1 L_1 (X) + \cdots + y_{n+1} L_{n+1} (X) \]

we have:

\[ p (x_i) = y_1 (0) + \cdots + y_i (1) + \cdots + y_{n+1} (0) \]
\[ = y_i \]
\[ \deg (p (X)) \leq \max_{1 \leq i \leq n+1} (\deg L_i) \]
\[ = n \]

So \( p (X) \) solves our problem.

To see that \( p (X) \) is the only solution with degree no more than \( n \), suppose \( q (X) \) satisfies \( q (x_i) = y_i \) and \( \deg (q (X)) \leq n \). We must show \( p (X) = q (X) \). Define \( r (X) = p (X) - q (X) \). Then

\[ r (x_i) = 0, \ i = 1, \ldots, n+1 \]
\[ \deg (r (X)) \leq n \]

By the Fundamental Theorem of Algebra a non-zero polynomial cannot have more distinct roots than its degree, so \( r (X) = 0 \) and \( p (X) = q (X) \).

**Example 2** See Mathematica example PolynomialInterpolation

**Theorem 3** Let \( n \) be a non-negative integer, and let \( f (X) \) be a function on the domain \([a, b]\) with continuous derivatives of order up to \( n + 1 \). Suppose have chosen points \( a = x_1 < x_2 < \cdots < x_{n+1} = b \) and define \( y_i = f (x_i) \). Let \( p (X) \) be the interpolating polynomial through \((x_i, y_i)\). Define \( \pi (X) = (X-x_1) \cdots (X-x_{n+1}) \).

Then for each \( x \in [a, b] \) there exists \( \xi \in [a, b] \) such that

\[ f (x) - p (x) = \frac{f^{(n+1)} (\xi)}{(n+1)!} \pi (x) \]

**Corollary 4** There exists a constant \( K \) independent of \( n \) and the choice of points \( x_i \) such that for all \( x \in [a, b] \):

\[ |f (x) - p (x)| \leq \frac{K}{(n+1)!} \pi (x) \]

**Proof.** of Corollary. Take

\[ K = \max_{\xi \in [a, b]} \left| f^{(n+1)} (\xi) \right| \]

**Proof.** of Theorem. Remember \( x \) is a fixed value throughout this proof. The theorem is obviously true if \( x = x_i \) for some \( i \), since both sides are 0 no matter what value of \( \xi \) we choose. For the remainder of this proof, assume \( x \neq x_i \) for all \( i \).

Since \( x \neq x_i \), \( \pi (x) \neq 0 \) and there exists a unique constant \( C \) such that:

\[ f (x) - p (x) = C \pi (x) \]
Polynomial Interpolation with Mathematica

Approximate the function \( f(X) = \frac{1}{1 + X^2} \) on the interval \([a, b]\).

\[
f[X_] = \frac{1}{1 + X^2};
\]
\[
a = -1.0;
\]
\[
b = 1.0;
\]

The approximation points are equally spaced from \(a\) to \(b\).

\[
n = 4; (* number of approximation points *)
\]
\[
x = Table[t, \{t, a, b, (b-a)/n\}]; (* list of x-values *)
\]
\[
y = f[x]; (* List of y-values *)
\]

Define Lagrange Polynomials and interpolating polynomial

\[
L[i_, X_] := Product[(X - x[[j]]) / (x[[i]] - x[[j]]), \{j, 1, i-1\}] *

Product[(X - x[[j]]) / (x[[i]] - x[[j]]), \{j, i+1, n+1\}]
\]
\[
p[X_] := Sum[y[[i]] * L[i, X], \{i, 1, n+1\}]
\]

Plot function and interpolating polynomial

\[
\text{graph1} = \text{Plot}[f[X], \{X, a, b\}, \text{PlotStyle} \to \text{Hue}[0.1],
\]
\[
\quad \text{DisplayFunction} \to \text{Identity}];
\]
\[
\text{graph2} = \text{Plot}[p[X], \{X, a, b\}, \text{PlotStyle} \to \text{Hue}[0.5],
\]
\[
\quad \text{DisplayFunction} \to \text{Identity}];
\]
\[
\text{graph3} = \text{ListPlot}[\text{Transpose}[\{x, y\}], \text{Prolog} \to \text{PointSize}[0.03],
\]
\[
\quad \text{DisplayFunction} \to \text{Identity}];
\]
\[
\text{Show}[\text{graph1}, \text{graph2}, \text{graph3}, \text{Prolog} \to \text{PointSize}[0.03],
\]
\[
\quad \text{DisplayFunction} \to \$\text{DisplayFunction}]
\]

\[
\text{Maximize}[\text{Abs}[f[X] - p[X]], a \leq X \&\& X \leq b, \{X\}]
\]
\[
\quad \{0.0222819, (X \to 0.827317)\}
\]

Figure 1
Define \( F(X) = f(X) - p(X) - C\pi(X) \). Then \( F(x_i) = 0 \) for \( i = 1, \ldots, n + 1 \) and \( F(x) = 0 \). Thus \( F \) has \( n + 2 \) zeros in the domain \([a, b]\). By Rolle’s theorem we can find \( n + 1 \) points in \([a, b]\) where \( F’(X) \) vanishes. Similarly there exist \( n \) points in \([a, b]\) where \( F''(X) \) vanishes. Continuing, there is one point \( \xi \in [a, b] \) such that \( F^{(n+1)}(\xi) = 0 \). Thus
\[
F^{(n+1)}(\xi) = f^{(n+1)}(\xi) - p^{(n+1)}(\xi) - C\pi^{(n+1)}(\xi) = 0
\]
However \( p^{(n+1)}(X) = 0 \) because \( p \) is a polynomial of degree no more than \( n \), and \( \pi^{(n+1)}(X) = (n + 1)! \) because \( \pi(X) \) is a monic polynomial of degree \( n \). Therefore:
\[
f^{(n+1)}(\xi) = C(n + 1)!
\]
Replacing the value of \( C \) in the first equation, we have:
\[
f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n + 1)!} \pi(x)
\]

\[
[\text{Lemma 5}]
\]

\textbf{5.2 Newton Interpolation}

The Lagrange polynomials are great for proving the existence of interpolating polynomials, but they are not the most efficient way to create and evaluate interpolating polynomials. If we want to actually calculate with interpolating polynomials, we need a more efficient scheme. We begin with an intermediate result that is interesting in its own right and needed for a proof later.

\textbf{Lemma 5} \textit{Let} \((x_i, y_i), i = 1, \ldots, n + 1\) \textit{be points with distinct} \(x_i\). \textit{Suppose} \(p(X)\) \textit{is the interpolating polynomial for} \((x_i, y_i), i = 1, \ldots, n\), \textit{and suppose} \(q(X)\) \textit{is the interpolating polynomial for} \((x_i, y_i), i = 2, \ldots, n + 1\). \textit{Then}
\[
r(X) = \frac{(X - x_1)q(X) - (X - x_{n+1})p(X)}{x_{n+1} - x_1}
\]
\textit{is the interpolating polynomial for} \((x_i, y_i), i = 1, \ldots, n + 1\).

\textbf{Proof.} We don’t have to show that \(r(X)\) is equal to anything. Since the interpolating polynomial is unique, all we have to prove is:
\[
r(x_i) = y_i, i = 1, \ldots, n + 1
\]
\[
\text{deg } (r(X)) \leq n
\]
For the first part, just calculate:
\[
r(x_1) = \frac{(x_1 - x_1)q(x_1) - (x_1 - x_{n+1})p(x_1)}{x_{n+1} - x_1}
\]
\[
= \frac{x_1 - x_1}{x_{n+1} - x_1}
\]
\[
= y_1
\]
For \(2 \leq i \leq n:\)
\[
r(x_i) = \frac{(x_i - x_1)q(x_i) - (x_i - x_{n+1})p(x_i)}{x_{n+1} - x_1}
\]
\[
= \frac{(x_i - x_1 - (x_i - x_{n+1}))y_i}{x_{n+1} - x_1}
\]
\[
= y_i
\]
Finally:
\[
r(x_{n+1}) = \frac{(x_{n+1} - x_1)q(x_{n+1}) - (x_{n+1} - x_{n+1})p(x_{n+1})}{x_{n+1} - x_1}
\]
\[
= \frac{x_{n+1} - x_1}{x_{n+1} - x_1}
\]
\[
= y_{n+1}
\]
For the last statement about degree, since \(\text{deg } (r(X)) \leq n - 1\) and \(\text{deg } (q(X)) \leq n - 1\), we have \(\text{deg } (r(X)) \leq n\).
5.2.1 Divided Differences

1. Suppose we have points \((x_i, y_i), i = 1, \ldots, n + 1\) with \(x_i\) distinct. We want to construct the divided-difference table for these points.

(a) The table will be a triangular array \(z_{i,j}, j = 1, \ldots, n+1, i = 1, \ldots, n+2-j\). (I think of \(i\) as the row number and \(j\) the column number.)

(b) \(z_{i,1} = y_i\)

(c) For \(2 \leq j \leq n+1\) define \(z_{i,j} = \frac{z_{i+1,j-1} - z_{i,j-1}}{x_{i+j-1} - x_i}\)

(d) Example: \(x = (1, 3, 4, 6), y = (2, 1, 3, 4)\)

\[
\begin{array}{c|c|c|c}
2 & -0.5 & 0.833 & -0.267 \\
1 & 2 & -0.5 & \\
3 & 0.5 & \\
4 & & & \\
\end{array}
\]

(e) Note that this divided-difference table contains subtables for consecutive subsets of points \((x_i, y_i), \ldots, (x_{i+j}, y_{i+j})\).

(f) You can add a point \((x_{n+2}, y_{n+2})\) to the table without changing any of the existing entries.

**Theorem 6** Given points \((x_i, y_i), i = 1, \ldots, n+1\) and the divided difference table \(z_{i,j}\), the interpolating polynomial is:

\[
r(X) = z_{1,1} + z_{1,2}(X-x_1) + \cdots + z_{1,n+1}(X-x_1) \cdots (X-x_n)
\]

**Proof.** We proceed by induction on \(n\). What does the theorem say when \(n = 0\)? Is it true?

Let \(p(X)\) be the interpolating polynomial for \((x_1, y_1), \ldots, (x_n, y_n)\) and let \(q(X)\) be the interpolating polynomial for \((x_2, y_2), \ldots, (x_{n+1}, y_{n+1})\). By induction:

\[
p(X) = \sum_{j=1}^{n} z_{1,j} \prod_{i=1}^{j-1} (X-x_i)
\]

\[
q(X) = \sum_{j=1}^{n} z_{2,j} \prod_{i=2}^{j} (X-x_i)
\]

By the Lemma, the interpolating polynomial for all the points is:

\[
r(X) = \frac{(X-x_1)q(X) - (X-x_{n+1})p(X)}{x_{n+1} - x_1}
\]

\[
= \frac{1}{x_{n+1} - x_1} \left[ \sum_{j=1}^{n} z_{2,j} (X-x_1) \left( \prod_{i=2}^{j} (X-x_i) \right) - \sum_{j=1}^{n} z_{1,j} \left( \prod_{i=1}^{j-1} (X-x_i) \right) (X-x_{n+1}) \right]
\]

\[
= \frac{1}{x_{n+1} - x_1} \left[ \sum_{j=1}^{n} z_{2,j} (X-x_1) \left( \prod_{i=2}^{j} (X-x_i) \right) - \sum_{j=1}^{n} z_{1,j} \left( \prod_{i=1}^{j-1} (X-x_i) \right) (X-x_j + x_j - x_{n+1}) \right]
\]

\[
= \frac{1}{x_{n+1} - x_1} \left[ \sum_{j=1}^{n} z_{2,j} \left( \prod_{i=1}^{j} (X-x_i) \right) - \sum_{j=1}^{n} z_{1,j} \left( \prod_{i=1}^{j} (X-x_i) \right) (x_j - x_{n+1}) \right]
\]
= \frac{1}{x_{n+1} - x_1} \left[ \sum_{j=1}^{n} (z_{2j} - z_{1j}) \left( \prod_{i=1}^{j} (X - x_i) \right) - \sum_{j=1}^{n} z_{1j} \left( \prod_{i=1}^{j-1} (X - x_i) \right) (x_j - x_{n+1}) \right]

= \frac{1}{x_{n+1} - x_1} \left[ \sum_{j=1}^{n} (x_{j+1} - x_1) z_{1,j+1} \left( \prod_{i=1}^{j} (X - x_i) \right) - \sum_{j=1}^{n} z_{1j} \left( \prod_{i=1}^{j-1} (X - x_i) \right) (x_j - x_{n+1}) \right]

= \frac{1}{x_{n+1} - x_1} \left[ \sum_{j=2}^{n+1} x_j z_{1,j} \left( \prod_{i=1}^{j-1} (X - x_i) \right) + \sum_{j=1}^{n} (x_{n+1} - x_1) z_{1,j} \left( \prod_{i=1}^{j-1} (X - x_i) \right) - x_1 z_{11} + x_{n+1} z_{11} - x_1 z_{1,n+1} \left( \prod_{i=1}^{n} (X - x_i) \right) \right]

= z_{11} + \sum_{j=2}^{n} z_{1,j} \left( \prod_{i=1}^{j-1} (X - x_i) \right) + z_{1,n+1} \left( \prod_{i=1}^{n} (X - x_i) \right)

= \sum_{j=1}^{n+1} z_{1,j} \left( \prod_{i=1}^{j-1} (X - x_i) \right)

1. You can use the Newton form of the interpolating polynomial to evaluate the polynomial rapidly.

   (a) Once the coefficients $z_{ij}$ are defined, you can evaluate the interpolating polynomial $p(X)$ at $X = x$ with the routine:

   $p = z_{11} \quad$ for $i = 1$ to $n$ do
   $\quad p = p \ast (x - x_i) + z_{1,i+1}$

1. The Newton form of the interpolating polynomial is also useful when you want to add a point to the set of points to be interpolated.

   (a) Adding a new point means creating only a new diagonal in the divided-difference table and adding one more term to the polynomial.

2. You can also use an existing divided-difference table to create the interpolating polynomial through an adjacent subset of the points to be interpolated.