Contents

PHY 480 - Computational Phyics Class Notes

Thomas Bolden

1 Numerical Derivation

1.1 Taylor Expansions

$$f(x+h) = f(x) + hf' + \frac{h^2}{2!}f'' + \frac{h^3}{3!}f''' + O(h^4) \quad (i)$$
$$f(h-h) = f(x) - hf' + \frac{h^2}{2!}f'' - \frac{h^3}{3!}f''' + O(h^4)$$

with step length h

$$(i) \text{ becomes } f(x+h) - f(x) = hf' + \frac{h^2}{2!} + O(h^3)$$
$$\frac{f(x+h) - f(x)}{h} = f' + O(h)$$

This gives the first approximation of f'

There is a truncation error

$$f' \approx \frac{f(x+h) - f(x)}{h} \quad \text{Euler's forward equation}$$

Given

$$\frac{\mathrm{d}f}{\mathrm{d}t} = g(t) \Rightarrow \frac{f(t+h) - f(t)}{h} \approx g(t)$$

We can never do this in a machine: $\lim_{h\to 0} \frac{f(x+h) - f(x)}{h} = f'$

So we wonder how we can do this numerically. When we make h smaller and smaller, we run into problems with relative error which can cause things to blow up (after 10^{-5} you encounter rounding errors).

Let's discretize the error: [notes picture 1]

A function of domain $[a, b]$, with $a \leq x_i \leq b$, $x_i = x_0 + ih$
$\Rightarrow \quad f(c) = f(x_i) = f_i \quad , \quad f(x+h) = f_{i+1}$

$$\frac{f(x+h) - f(x)}{h} = \frac{f_{i+1} - f_i}{h} = f'$$
$$\frac{f(x) - f(x-h)}{h} = \frac{f_i - f_{i-1}}{h} = f'$$

(Backwards Euler truncation OCh)

Lets try (using the Taylor expansions)

$$f(x+h) - f(x-h) = 2hf' + \frac{2}{3!}h^3 f''' + O(h^5) \quad \longleftarrow \text{ notice we only get odd powers}$$

$$\Rightarrow \frac{f(x+h) - f(x-h)}{2h} = f' + O(h^2)$$

Which is a three point equation, which can represent quadratic functions well.
We get the result

$$\frac{f(x+h) + f(x-h) - 2f(x)}{h^2} = f'' + O(h^2)$$

which is a workhorse to solve differential equations!
4 formans formulae: (VERY IMPORTANT!!)

$$\frac{f_{i+1} - f_i}{h} = f' + O(h)$$
$$\frac{f_i - f_{i-1}}{h} = f' + O(h)$$
$$\frac{f_{i+1} - f_{i-1}}{2h} = f' + O(h^2)$$
$$\frac{f_{i+1} + f_{i-1} - 2f_i}{h^2} = f'' + O(h^2)$$

1.2 Intro to Project 1 - The Math

We are given a differential equation of the type: $\frac{\mathrm{d}^2}{\mathrm{d}x^2} = f(x)$. Assume $x \in [0,1]$, and there are boundary conditions (Drinichlet conditions) $U(0) = 0 \ , \ U(1) = 0$.
In order to discretize, set

$$x \in [0,1] \longrightarrow \{x_0, x_1, ..., x_m\}$$
$$0 \to x_0 \quad , \quad i \to x_i \quad , \quad x_i = x_0 + ih$$

The differential equatino gets replaced by:

$$\frac{U_{i+1} + U_{i-1} - 2U_i}{U^2} = f_i \text{ (not exactly precise but close enough)}$$

$$U_{i+1} + U_{i-1} - 2U_i = U^2 f_i = \tilde{f}_i$$

$$\hat{u} = \begin{bmatrix} u_0 \\ u_1 \\ . \\ . \\ . \\ u_{m-1} \\ u_m \end{bmatrix} \ , \ \hat{f} = \begin{bmatrix} f_0 \\ f_1 \\ . \\ . \\ . \\ f_{m-1} \\ f_m \end{bmatrix}$$

$$\hat{A} \cdot \hat{u} = \hat{\hat{f}}$$

$$\Rightarrow \hat{A} = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -2 \end{bmatrix}$$

1.3 Loss of Numerical Precision

Loss of numerical Precision on github

1.4 Using GitHub

Before doing anything, run brew update and brew upgrade to get the latest version of things.

Open the repository in the terminal from the GitHub desktop application.
To make a new folder in the repository:

```
MacBook-Pro: RepositoryName Thomas$ mkdir FolderName
```

Copy files from the computer to the folder in finder

MAKE SURE TO PUSH CHANGES TO GITHUB

1.5 Introducing C++

The Basics

Running Programs from the Command Line

The first step is changing the current directory to where the file is stored (use ls to list files):

```
MacBook-Pro:~ Thomas$  cd /path/to/file
```

Once we are in the correct directory, compile the code in C++ using -o:

```
MacBook-Pro: FileDirectory Thomas$  c++ -o filename.x filename.cpp
```

If the code has armadillo in it:

```
MacBook-Pro: FileDirectory Thomas$ c++ -o filename.x filename.cpp -larmadillo -llapack
-lblas -L/usr/local/lib -I/usr/local/include
```

Then run the code:

```
MacBook-Pro: FileDirectory Thomas$  ./filename.x
```

Here are some other useful terminal commands:
To view the current workind directory path

```
MacBook-Pro:~ Thomas$ pwd
```

To move up one level in a directory

```
MacBook-Pro:~ Thomas$ cd ..
```

Hello World

```
// comments look like this
#include<iostream> // input and output
include<cmath> // math functions
using namespace std;
int main(int argc, char* argv[])
{
    // convert the text argv[1] to double using atof:
    double r = atof(argv[1]); /* convert the text argv[1] to double */
    double s = sin(r);
    cout << "Hello, World! sin(" << r << ") =" << s << endl;
    return 0; /* success execution of the program */
}
```

In C++, you do not need to declare variables, but should anyways
Because there is an argv[1] in the program, you need to give the program a variable IN THE COMMAND LINE. Otherwise use cin »

Pointers

A pointer specifies where a value resides in computer memory (like an address)

Take advantage of "call by reference" !

1.6 Error Analysis

$$\epsilon = \log_{10}\left(\left|\frac{f''_{\text{computed}} - f''_{\text{exact}}}{f''_{\text{exact}}}\right|\right)$$

$$\epsilon_{\text{tot}} = \epsilon_{\text{approx}} + \epsilon_{\text{ro}}$$

$$f''_0 = \frac{f_h - 2f_0 + f_{-h}}{h^2} = \frac{(f_h - f_0) + (f_{-h} - f_0)}{h^2}$$

Our total error becomes [*see website slides]

2 Linear Algebra Methods

Useful Packages for computing linear algebra

- LINPACK: linear algebra and least square problems
- LANPACK
- BLAS (Basic Linear Algebra Subprograms)

C++ stores matrices in row major order! vs. column major order in Fortran

2.1 Gaussian Elimination

-Forward substitution
-Backward substitution
EX: $\hat{A} \in \mathbb{R}^{4\times4}$, $\hat{x} \in \mathbb{R}^{1\times4}$, $w \int \mathbb{R}^{1\times4}$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \longrightarrow \ldots \longrightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & \tilde{a}_{22} & \tilde{a}_{23} & \tilde{a}_{24} \\ 0 & 0 & \tilde{a}_{33} & \tilde{a}_{34} \\ 0 & 0 & 0 & \tilde{a}_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

The number of floating point operations on an $M \times N$ matrix goes as $\frac{2}{3}M^3$ (forward substitution) + M^2 (backward substitution).
Given $\hat{A} \in \mathbb{R}^{n\times n}$, and $\hat{A} = \hat{L} \cdot \hat{u}$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ \ell_{21} & 1 & 0 & 0 \\ \ell_{31} & \ell_{32} & 1 & 0 \\ \ell_{41} & \ell_{42} & \ell_{43} & 1 \end{bmatrix} \times \begin{bmatrix} u_{11} & 2_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix}$$

$$\det(A) = \det(Lu) = \det(L)\det(u) = \prod_{i=1}^{n} u_{ii} = \prod_{i=1}^{n} \frac{i+1}{i} = n+1$$

LU decomposition
We can avoid brute force and reduce the number of FLOPS to $4M$!!
FLOPS, Gaussian Elimination & tridiagonal solver (Thomas Algorithm).
By forward substitution,

$$\tilde{d}_i = d_i - \frac{e_{i-1}^2}{\tilde{d}_{i-1}}$$

2.2 Matrix Handling in C++

Row Major Order, Addition

```
int N = 100;
double A[100][100];
// initialize all elements to zero
for(i=0 ; i<N ; i++) {
    for( j=0 ; j<n ; j++) {
        come back from notes online
    }
}
```

There are a lot of lines here, why Armadillo is so helpful !!

```cpp
#include <armadillo>
#include <armadillo>

using namespace std;
using namespace arma;

int main(int argc, char** argv) {
    mat A = randu<mat>(5,5);
    mat N = randu<mat>(5,5);

    cout << A*B << endl;

    return 0;
}
```

3 Scientific Report Writing

4 Mat

5 Interpolation

5.1 Cubic Spline Interpolation

Situation:

$$\begin{array}{cc} x_0 & f(x_0) \\ x_1 & f(x_1) \\ x_2 & f(x_2) \\ . & . \\ . & . \\ . & . \\ x_n & f(x_n) \end{array}$$

$$\begin{aligned} f(x) \approx O_n(x) &= \sum_{i=0}^{n} a_i x^i \\ &= a_0 + a_1 x + a_2 x^2 + ... + a_n x^n \end{aligned}$$

(continue from written notes)

5.2 Lagrange Interpolation

6 Project 2

6.1 The physics of project 2

Quantum Dots

Schrödinger equation for 2 particles in a harmonic oscillaor well.

$$\left(-\frac{\hbar^2}{m}\frac{1}{r^2}\frac{d}{dr}r^2\frac{d}{dr}+\frac{1}{4}m\omega^2r^2+\frac{ke^2}{r}\right)\phi(r)=E_n\phi(r)$$

Equation to solve:

$$\left(\frac{d^2}{d\rho^2}+\rho^2+\frac{\beta}{\rho}\right)u(\rho)=\lambda u(\rho)$$

$$=\left(-\frac{d^2}{d\rho^2}+v(\rho)\right)u(\rho)=\lambda u(\rho)$$

$$\hat{A}\cdot\hat{u}$$