Numerical Analysis / Scientific Computing
CS450

Andreas Kloeckner

Spring 2019

Today:

- Set the stage
- About the class
- Numerics

Outline

What's the point of this class?

'*Scientific Computing*' describes a family of approaches to obtain approximate solutions to problems...

...once they've been stated mathematically.

Name some applications

- Large-scale engineering simulation
- Machine learning
  - ↪ Optimization
- Image and audio processing
  - ↪ Interpolation

What do we study, and how?

Problems with real numbers (i.e. *continuous* problems)

i.e. not discrete

Problem 1: $\mathbb{R} \to$ computer?

What's the general approach?

- build model in terms of repr. — degrees of freedom: numbers used to repr. solution
- solving that model: existence? uniqueness?
- are we answering the question that we care about

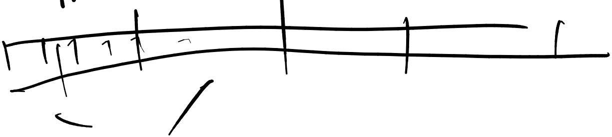
What makes for *good* numerics?

How good of an answer can we expect to our problem?

- Answers are always approximate
- how far off?

How fast can we expect the computation to complete?

- what algorithm will we use?
- what's the cost?
- is the approach efficient



Implementation concerns

How do numerical methods *get implemented*?

- layer cake of abstractions ("lies")
  - tools / languages
  - methods: ease of / customization
  - robustness

Class web page

https://bit.ly/cs450-s19

- Assignments
  - HW0!
  - Pre-lecture quizzes
  - In-lecture interactive content (bring computer or phone if possible)
- Textbook
- Exams
- Class outline (with links to notes/demos/activities/quizzes)
- Virtual Machine Image
- Piazza
- Policies
- Video
- Inclusivity Statement

Programming Language: Python/numpy

- Reasonably readable
- Reasonably beginner-friendly
- Mainstream (top 5 in 'TIOBE Index')
- Free, open-source
- Great tools and libraries (not just) for scientific computing
- Python 2/3? 3!
- `numpy`: Provides an array datatype
  Will use this and `matplotlib` all the time.
- See class web page for learning materials

Demo: Sum the squares of the integers from 0 to 100. First without numpy, then with numpy.

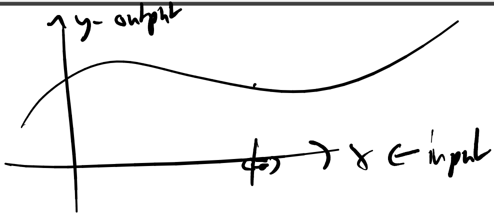
Supplementary Material

- Numpy (from the SciPy Lectures)
- 100 Numpy Exercises
- Dive into Python3

What problems *can* we study in the first place?

To be able to compute a solution (through a process that introduces errors), the problem...

- they have a solution
- the solution is unique
- the solution depends continuously on the input



Dependency on Inputs

We excluded discontinuous problems–because we don't stand much chance for those.
...what if the problem's input dependency is just *close to discontinuous*?

- Those problems are called sensitive
- opposite: insensitive

Approximation

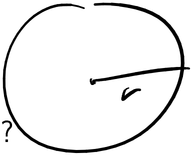
When does approximation happen?

Input
- model
- measurement / repr. of input data
- computation of input data

Comp.
- truncation
- rounding

Demo: Truncation vs. Rounding

Example: Surface Area of the Earth



Compute the surface area of the earth.
What parts of your computation are approximate?

$$A = 4\pi r^2$$

– all of them (not a sphere, rounding)

Measuring Error

How do we measure error?

Idea: Consider all error as being *added onto* the result.

Absolute error $=$ |True value $-$ approx value|

relative error $= \dfrac{|\text{absolute error}|}{|\text{True value}|}$ (not if true $= 0$)

Upper bounds on error

Recap: Norms

What's a norm?

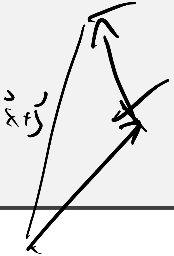
$f : \mathbb{R}^n \to \mathbb{R}_0^+$ "magnitude"

$\vec{x} \qquad \|\vec{x}\|$

Define *norm*.

$\|\cdot\|$ is called a norm if

- $\|\vec{x}\| > 0 \iff \vec{x} \neq 0$
- $\|\alpha \vec{x}\| = |\alpha| \, \|\vec{x}\|$
- $\|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\|$



Norms: Examples

Examples of norms?

$$\left\|\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}\right\|_p = \sqrt[p]{\sum_{i=1}^{n} |x_i|^p}$$

$$p = 1, 2, \infty$$

Demo: Vector norms

Norms: Which one?

Does the choice of norm really matter much?

Given any two norms in $\mathbb{R}^n$, $n < \infty$

$\|\cdot\|$, $\|\cdot\|_*$

there exist numbers α, β:

$$\alpha \|x\| \le \|x\|^* \le \beta \|x\|$$

Norms and Errors

If we're computing a vector result, the error is a vector.
That's not a very useful answer to 'how big is the error'.
What can we do?

abs error : $\|\vec{true} - \vec{approx}\|$

rel error : $\dfrac{\|\vec{true} - \vec{approx}\|}{\|\vec{true}\|}$

Forward/Backward Error

Suppose we're *intending* to compute $y = f(x)$,
but *actually obtain* $\hat{y} = \hat{f}(x)$.

What are the forward error and the backward error?

Forward/Backward Error: Example

Suppose you wanted $y = \sqrt{2}$ and got $\hat{y} = 1.4$.
What's the (magnitude of) the forward error?