In this class:

- *A quick overview of the numerical methods perspective on applied mathematics*
- *Overview of Unit 3.*
- *Interpolation "grids"*
- *Polynomial interpolation via least squares.*
- *APPROXFUN*

October 24, 2016

Numerical Methods, Applied Mathematics, and Polynomial interpolation

Next class

Lagrange polynomials

Next next class

QUIZ
Piecewise interpolants & Splines

Applied mathematics

functions

not

numbers

Find a function with this property

Find a function that satisfies this equation

Find a property of this function

Applied mathematics

What is the value of the integral of a function?

What is the derivative function of a given function?

What function solves a given differential equation?

Problems in applied math

In this part of the course, it is much harder to find “simple real world” examples as the complexities of realistic problems multiply.

Many of the problems we’ll study are abstractions of more intricate real-world problems.

Numerical Methods for Applied Math

Error 1/Approx 1	1.	Take the continuous problem. e.g. integral
Error 2/Approx 2	2.	Compute a discrete representation.
Error 3/Approx 3	3.	Determine where to apply continuous & discrete properties to derive a *tractable* problem. e.g. linear system
Error 4/Approx 4	4.	Solve the tractable problem. e.g. LU factorization

Key assumption

There is something in the problem we can evaluate exactly.

where exactly means up to floating point error

e.g.

f(x) in an integral or derivative

the boundary condition of an ODE

Outline of Topics

1. Polynomial approximations & piecewise polynomial approximations (Chapter 8)
2. Numerical differentiation (Chapter 9)
3. Numerical integration (Chapter 10)
4. ODEs and PDEs (Chapter 11, 13, 14)
5. Optimization (Chapter 4)

Reading outline on website

www.cs.purdue.edu/homes/dgleich/cs314-2016/syllabus.html

Polynomial approximation

Polynomials are one of the most useful ways of representing 1d functions on a computer.

Alternatives

grid of points images

sins/cosines signals/MP3s

radial basis functions machine learning

Polynomial approximation.

Key points

Not all sets of point to “evaluate” are equal.

1d case is “solved”

Many different ways to write polynomials that “fit” a set of points exactly, *but they have different numerical properties*.

Piecewise polynomials are flexible models

Polynomial methods do not generalize to higher dimensions easily.

Numerical differentiation

Given a computer representation of a function, how can we determine or approximate it's derivative?

f is	1d	2d	X-d
A regular/uniform grid of points			
A polynomial			
A set of sines/cosines			
A scattered set of data			

Numerical differentiation

Key points

Numerical accuracy is tricky with regular grids

Polynomial representations make differentiation “easy”

There are some standard approaches to improve the accuracy of numerical derivatives on regular grids. (Richardson extrapolation)

… numerical integration …

… ODES and PDES …

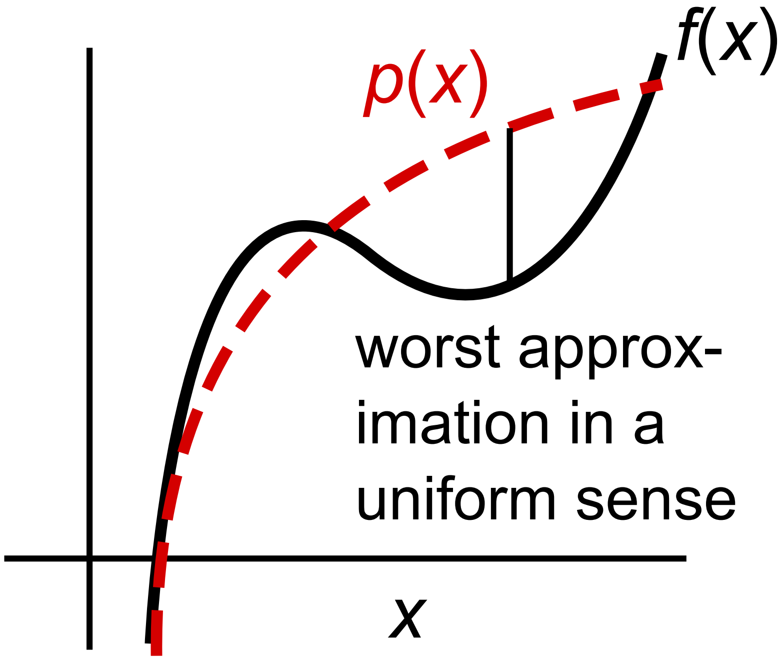
… optimization …

coming soon!

Why polynomial approximation?

Weierstrass approximation theorem

Every continuous function on an interval [a,b] can be *uniformly approximated* by as closely as desired by a polynomial



$$\text{uniform error} = \max_{x \in [a,b]} \left| f(x) - p(x) \right|$$

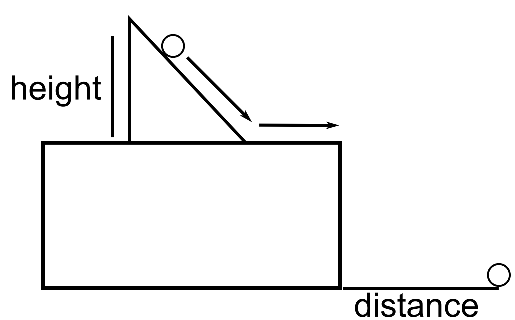
How to do polynomial interpolation?

We'll see a bunch of different ways to do this!

The easiest – via Least squares!

Given $(y_1, x_1), (y_2, x_2), \ldots, (y_m, x_m)$
Find $c_0, \ldots, c_n$
such that $y_i \approx c_0 + c_1 x_i + c_2 x_i^2 + \ldots + c_n x_i^n$.

Galileo wanted to find a mathematical relationship between ball height and horizontal distance in the following experiment.



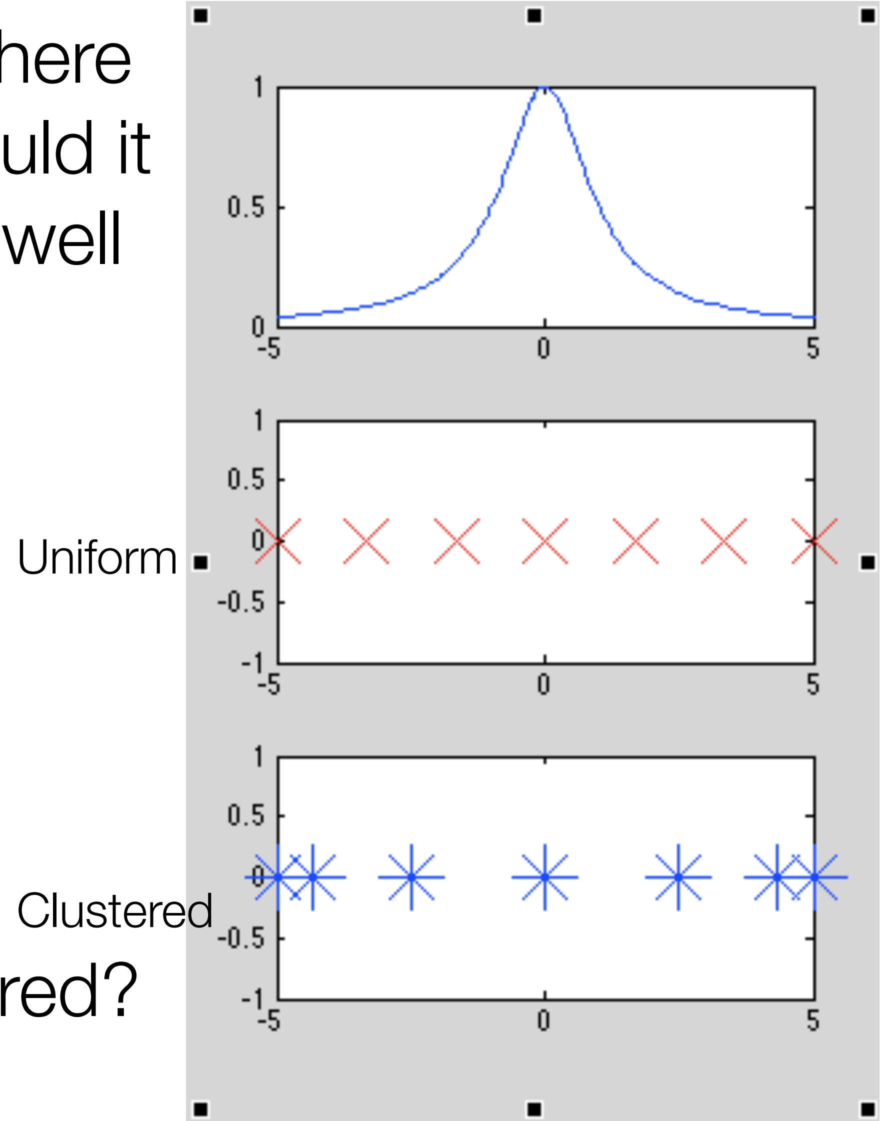
height	distance
0.282	0.752
0.564	1.102
0.752	1.248
0.940	1.410

QUIZ

Q1. If we could choose where to "see" the function *f*, would it make a difference to how well we can interpolate it with polynomials?

Q2. How can we fit a degree *n* polynomial to *n+1* points?

Q3. Which set of points is better to interpolate *f*, red/uniform or blue/clustered?



How to do polynomial interpolation?

We'll see a bunch of different ways to do this!

The easiest – via Least squares!

Given $(y_1, x_1), (y_2, x_2), \ldots, (y_m, x_m)$
Find $c_0, \ldots, c_n$
such that $y_i \approx c_0 + c_1 x_i + c_2 x_i^2 + \ldots + c_n x_i^n$.

$$\text{minimize} \left\| \underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^n \\ 1 & x_2 & x_2^2 & \ldots & x_2^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_m & x_m^2 & \ldots & x_m^n \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}}_{\mathbf{x}} - \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_{\mathbf{b}} \right\|_2^2$$

How to do polynomial interpolation?

```
"""c = polyfit(x,y,n): fit the coefficients of a poly
interp. fits a degree n polynomial to the data x,y"""
function polyfit(x,y,n)
  m = length(x)      # datapoints
  A = zeros(m,n+1)   # matrix
  for i=1:m
    xi = 1.
    for j=1:n+1
      A[i,j] = xi
      xi *= x[i]
     end
   end
  return A\y       # least-squares or linsys
end
```

Polynomial interpolation

… demo …

Lecture-26 on Juliabox!

Error in polynomial interpolation

THEOREM 8.4.1

Assume

that f is $n+1$ times cont. diff. in a region $[a, b]$, and that $x_0, \dots, x_n$ are distinct points in $[a, b]$.

Let

$p(x)$ be the unique polynomial of degree n that interpolates f at $x_0, \dots, x_n$.

Then

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi_x) \prod_{j=0}^{n} (x - x_i)$$

for some point ξ_x in $[a, b]$ that depends on x.

Analysis

Hard to control if
the n+1 derivative
isn't well behaved.

Goes down with n

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi_x) \prod_{j=0}^{n} (x - x_i)$$

A polynomial that is
zero at each x_i if this
is necessarily large,
we'll study this!

Interpolation at Chebyshev points

THEOREM 8.5.1

Let

f be a continuous function on $[-1, 1]$

p_n its degree n interpolant at Chebyshev points

p_n^* its best approximation among n degree polynomials in the uniform error

Then

uniform error in $p_n \leq (2 + \frac{2}{\pi} \log n)$uniform error in p_n^*

p_n converges exponentially fast to f if f is smooth

Analysis

If we interpolate f at Chebyshev points, we get something close to the *best possible result*

and if *f* is smooth, the polynomial approximation always converges "fast"

ApproxFun demo