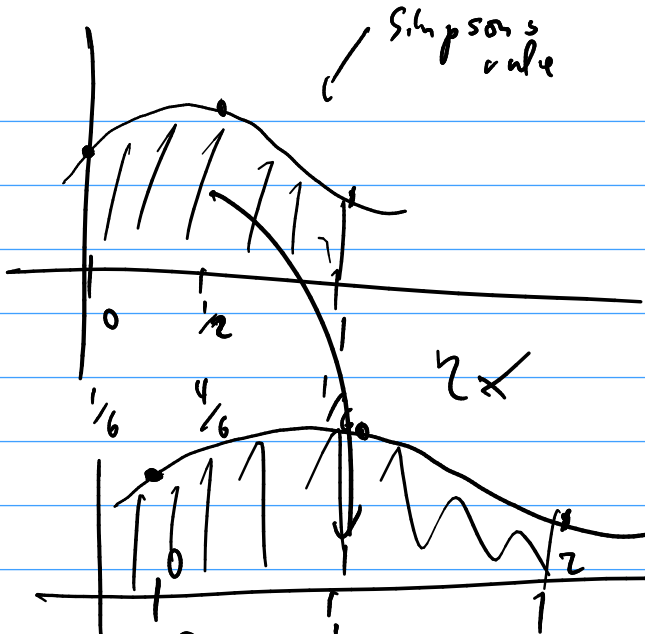
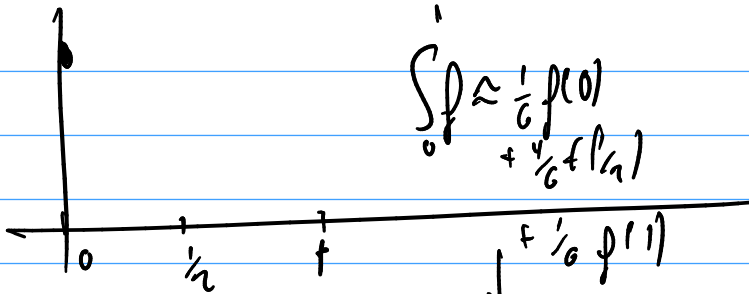


Overview

- Shifting / scaling
- Acc quad
- Convergence
- Solving eq.





$$\int_0^2 f \approx \frac{2}{6} f(0) + \frac{4}{6} f(1) + \frac{2}{6} f(2)$$

To find a quadrature rule
on any interval $[a, b]$:

1. Cook up $\varphi(x) = mx + n$
so that $\varphi(0) = a$ $\varphi(1) = b$

2. new nodes: $\tilde{x}_i = \varphi(x_i)$

3. weights: $\tilde{w}_i = \varphi'(x_i) w_i$

$$= m \cdot w_i$$

Example: Building a Quadrature Rule

Demo: Computing the Weights in Simpson's Rule

Suppose we know

$$f(x_0) = 2 \quad f(x_1) = 0 \quad f(x_2) = 3$$

$$x_0 = \cancel{1}0 \quad x_1 = \frac{1}{2} \quad x_2 = 1$$

How can we find an approximate integral?

Facts about Quadrature

nodes: $0, \frac{1}{2}, 1$ $w_i: \frac{1}{6}, \frac{4}{6}, \frac{1}{6}$

What does Simpson's rule look like on $[0, 1/2]$?

$$\frac{1}{6} f(0) + \frac{4}{6} f\left(\frac{1}{4}\right) + \frac{1}{6} f\left(\frac{1}{2}\right)$$

What does Simpson's rule look like on $[5, 6]$?

$$\frac{1}{6} f(5) + \frac{4}{6} f(5.5) + \frac{1}{6} f(6)$$

How accurate is Simpson's rule with polynomials of degree n ?

$$\int_0^h = h \cdot \int_0^1 \text{interpolant}$$

Error for interpolation

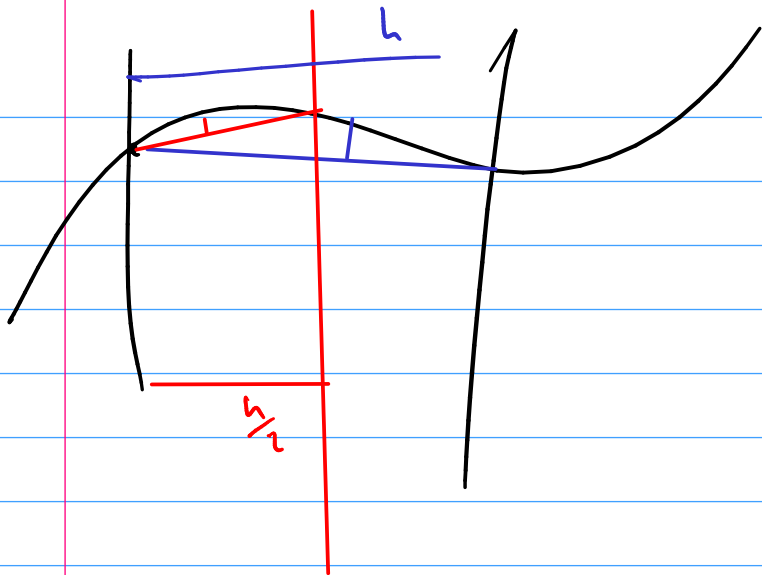
$$|p(x) - \tilde{p}(x)| \leq \underline{C \cdot h^{n+1}}$$

n poly degree

$$\begin{aligned} |S p - S \tilde{p}| &\leq \int_0^h |p - \tilde{p}(x)| \\ &\leq C \cdot \int_0^h C \cdot h^{n+1} \end{aligned}$$

$$= C \cdot h \int_0^1 h^{k+1}$$

$$\leq C \cdot h^{k+2} \int_0^1 \dots$$

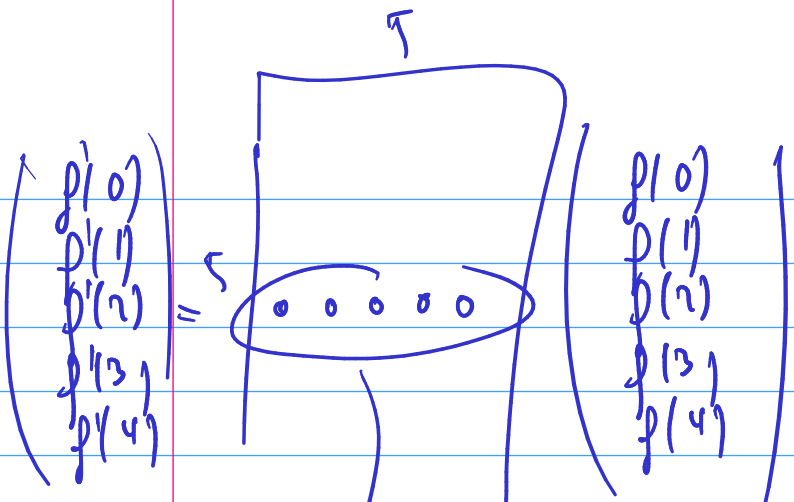


Why do the weights in Simpson's rule add up to 1?

$$\begin{aligned} & \underbrace{\frac{1}{6}} \cdot f(0) + \underbrace{\frac{4}{6}} f\left(\frac{1}{2}\right) + \underbrace{\frac{1}{6}} \cdot f(1) \\ &= \frac{1}{6} + \frac{4}{6} + \frac{1}{6} \end{aligned}$$

$$v' \left(v^{-1}(\vec{p}) \right)$$

eval_deriv.(compute_coeff(\vec{p}))



$$V' V^{-1}$$

$$f'(2) = \underbrace{\quad} f(0) + \dots + \underbrace{\quad} f(4)$$

Outline

Python, Numpy, and Matplotlib
Making Models with Polynomials
Making Models with Monte Carlo
Error, Accuracy and Convergence
Floating Point
Modeling the World with Arrays
 The World in a Vector
 What can Matrices Do?
 Graphs
 Sparsity
Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations:
Eigenvalues and Steady States
Eigenvalues: Applications

Approximate Undo: SVD and Least Squares

SVD: Applications

Solving Funny-Shaped Linear Systems

Data Fitting

Norms and Condition Numbers

Low-Rank Approximation

Interpolation

Iteration and Convergence

Solving One Equation

Solving Many Equations

Finding the Best: Optimization in 1D

Optimization in n Dimensions

What is linear convergence? quadratic convergence?

$$e_k = \|x_k - \hat{x}\|$$

$$e_{k+1} = \underbrace{\frac{\lambda_k}{\lambda_1}}_{e_k} \cdot e_k$$

"linear convergence"

$$\frac{e_{k+1}}{e_k} = c \quad e_{k+1} = C \cdot e_k$$

actually works if $C < 1$

"quadratically convergent"

$$\frac{e_{k+1}}{e_k^2} = C \quad \Leftrightarrow \quad e_{k+1} = \underline{C} \cdot e_k^2$$

$$e_1 = 0.1 \quad C = 0.9$$

About Convergence Rates

Demo: Rates of Convergence

Characterize linear, quadratic convergence in terms of the 'number of accurate digits'.

linear: gains a const. # of digits

quad: doubles # of digits

Outline

Python, Numpy, and Matplotlib
Making Models with Polynomials
Making Models with Monte Carlo
Error, Accuracy and Convergence
Floating Point
Modeling the World with Arrays
 The World in a Vector
 What can Matrices Do?
 Graphs
 Sparsity
Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations:
Eigenvalues and Steady States
Eigenvalues: Applications

Approximate Undo: SVD and Least Squares

SVD: Applications

Solving Funny-Shaped Linear Systems

Data Fitting

Norms and Condition Numbers

Low-Rank Approximation

Interpolation

Iteration and Convergence

Solving One Equation

Solving Many Equations

Finding the Best: Optimization in 1D

Optimization in n Dimensions

Solving Nonlinear Equations

What is the goal here?

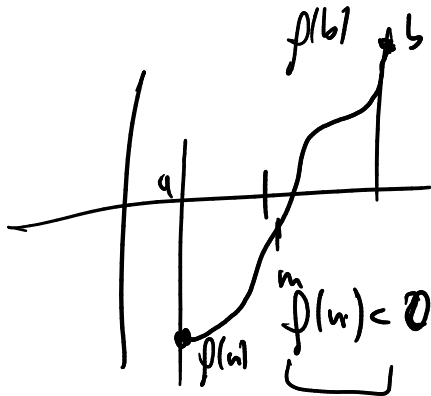
$$x^{17} + 3x^3 + 5x^2 + 3x + 7 = 0$$

$f(x)$

$$f(a) < 0$$

$$f(b) > 0$$

Bisection



$$\begin{array}{cc} \tilde{a} + & \tilde{b} \cdot 2^{-57} \\ \uparrow & \uparrow \\ 64 & 64 \end{array}$$

$$(a + b \cdot 2^{-57}) \cdot (c + d \cdot 2^{-57})$$

Bisection Method

Assume continuous function f has a zero on the interval $[a, b]$ and

$$\text{sign}(f(a)) = -\text{sign}(f(b)).$$

Perform binary search: check sign of $f((a + b)/2)$ and define new search interval so that ends have opposite sign.

Demo: [Bisection Method](#)

What's the rate of convergence? What's the constant?

Newton's Method

Derive Newton's method.