

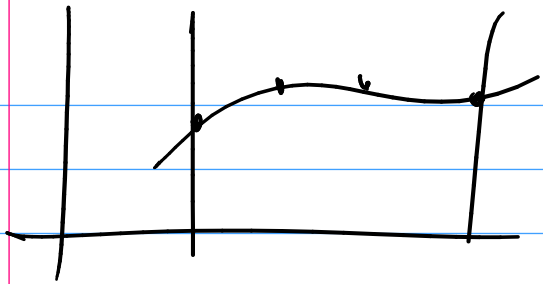
Overview

nd solve

opt

1D

2D



$$\forall \alpha_i \in \mathbb{R}$$

$$\tilde{f}(x) = \alpha_0 p_0(x) + \dots + \alpha_{n-1} p_{n-1}(x)$$

$$\int \tilde{f}(x) dx = \alpha_0 \left(\int p_0 \right) + \dots + \alpha_{n-1} \left(\int p_{n-1} \right)$$

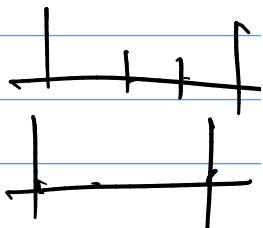
p_{n-1}

$$\vec{I}^T \begin{pmatrix} S \varphi_0 \\ \vdots \\ S \varphi_{n-1} \end{pmatrix}$$

$$\vec{I}^T (V^{-1} \vec{f}) \leftarrow \frac{h^T}{h^3}$$

$$= \underbrace{(\vec{I}^T V^{-1})}_{\substack{\text{row} \\ \vec{w}}} \vec{f} \leftarrow h$$

precompute!



Solving Nonlinear Equations

What is the goal here?

Demo: [Three quadratic functions](#) (click to visit)

Newton's method

What does Newton's method look like in n dimensions?

$$f \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$f \begin{pmatrix} a \\ ? \end{pmatrix} = \begin{pmatrix} < 0 \\ 0 \end{pmatrix}^n$$

$$f \begin{pmatrix} b \\ ? \end{pmatrix} \approx \begin{pmatrix} > 0 \\ 0 \end{pmatrix}$$

$$\vec{x}_{k+1} = \vec{x}_k + h \vec{p}$$



$$f(\vec{x}_k + h \vec{p}) = f(\vec{x}_k)$$

$$f \rightarrow \frac{\partial f}{\partial x_1} \quad \left| \frac{\partial f}{\partial x_n} \right| \quad h \vec{p}$$

says "how much does f change if h_1 changes"

$$\vec{0} = \vec{f}(\vec{x}_k + \vec{h}) = \vec{f}(\vec{x}_k) + \mathcal{J}_f(\vec{x}_k) \cdot \vec{h}$$

$$\vec{0} = \vec{f}(\vec{x}_k) + \mathcal{J}_f(\vec{x}_k) \cdot \vec{h}$$

$$\mathcal{J}_f^{-1}(\vec{x}_k) \cdot \left(-\vec{f}(\vec{x}_k) \right) = \vec{h}$$

$$\vec{x}_{k+1} = \vec{x}_k - \mathcal{J}_f^{-1}(\vec{x}_k) \cdot \vec{f}(\vec{x}_k)$$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Newton: Example

Set up Newton's method to find a root of

$$\mathbf{f}(x, y) = \begin{pmatrix} x + 2y - 2 \\ x^2 + 4y^2 - 4 \end{pmatrix} = \mathbf{0}$$

$$\frac{\partial f_1}{\partial x} \quad \frac{\partial f_1}{\partial y}$$

Demo: [Newton's method in n dimensions](#) (click to visit)

$$\mathcal{J}\mathbf{f}\left(\begin{matrix} x \\ y \end{matrix}\right) = \begin{pmatrix} 1 & 2 \\ 2x & 8y \end{pmatrix}$$

Secant in n dimensions?

What would the secant method look like in n dimensions?

It doesn't,

$$\begin{aligned} \leftarrow f(x_n) \quad p(x_{n+1}) &\rightarrow O(h) \\ \underbrace{\quad}_{\mathcal{J}p(x_n)} &\rightarrow O(h^n) \end{aligned}$$

$$f(x) = (x-15) / (x-10)(x-5)$$

$$g(x) = \frac{f(x)}{(x-15)}$$

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

- Making Interpolation Work
Better

- Calculus on Interpolants

Iteration and Convergence

Solving One Equation

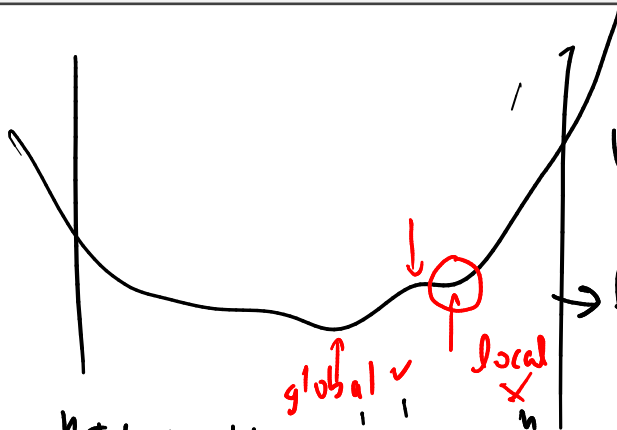
Solving Many Equations

**Finding the Best: Optimization in
1D**

Optimization in n Dimensions

Optimization

State the problem.



"Find the minimum"

Sufficient cond: $f''(x) > 0$

Simplified idea:
Use Newton
to solve:

$$f'(x) = 0$$

Necessary
condition

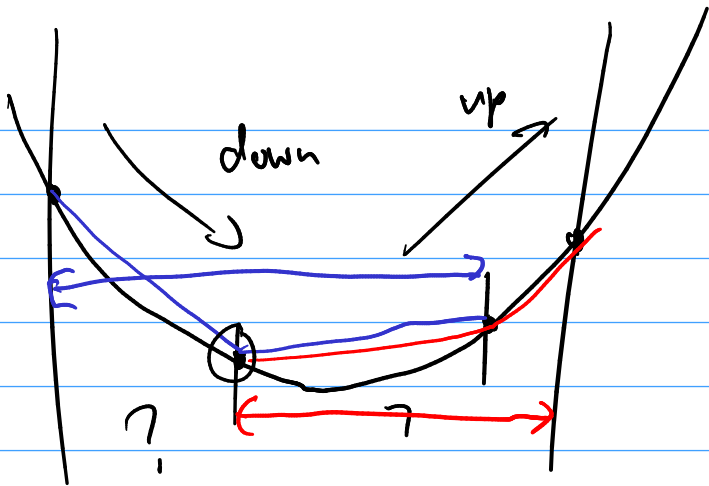
Newton for 1D opt?

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

still quadratically convergent.



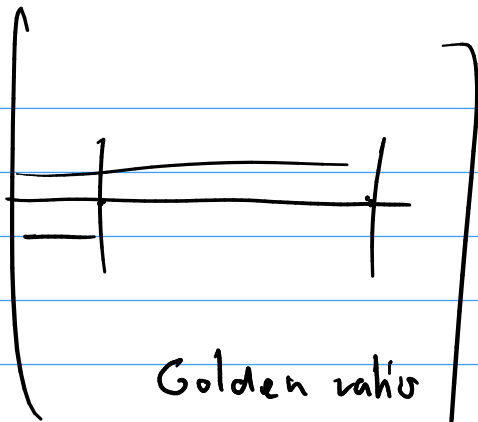
Um, no.



" unimodal "

$\frac{1}{3}$

$\frac{2}{3}$



Golden ratio

Golden section

Golden section search

Optimization: What could go wrong?

What are some potential problems in optimization?



- lokasi minimum

Optimization: What is a solution?

How can we tell that we have a (at least local) minimum? (Remember calculus!)

Newton's Method

Let's steal the idea from Newton's method for equation solving:
Build a simple version of f and minimize that.

Demo: [Newton's Method in 1D](#) (click to visit)

In-class activity: Optimization Methods

Golden Section Search

Would like a method like bisection, but for optimization.

In general: No invariant that can be preserved.

Need *extra assumption*.

Demo: [Golden Section Search Proportions](#) (click to visit)

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

- Making Interpolation Work
Better

- Calculus on Interpolants

Iteration and Convergence

Solving One Equation

Solving Many Equations

Finding the Best: Optimization in
1D

Optimization in n Dimensions

Optimization in n dimensions: What is a solution?

How can we tell that we have a (at least local) minimum? (Remember calculus!)

Find $\min f(x, y)$

Steepest Descent

Given a scalar function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ at a point x , which way is down?