Microsoft Research

The right solution to a nearby problem?
Or a nearby solution to the right problem?

OPTIMIZATION

Microsoft

- Chase the high-hanging fruit.
- Try to make stuff really work.
- Look for things that confuse/annoy you.

Alex Kipman

"The greatest danger for most of us is not that our aim is too high and we miss it, but that it is too low and we reach it" -- Michelangelo

Yesterday at 06:38 · Like · Comment

- Write about it well.

How to get ideas

- Computing PCA, LDA, ….
- Clustering, Gaussian mixture fitting, ….

- Sometimes the optimization is easy…

…sometimes it's hard

- Sometimes the hard problem is the one you must solve

Given function

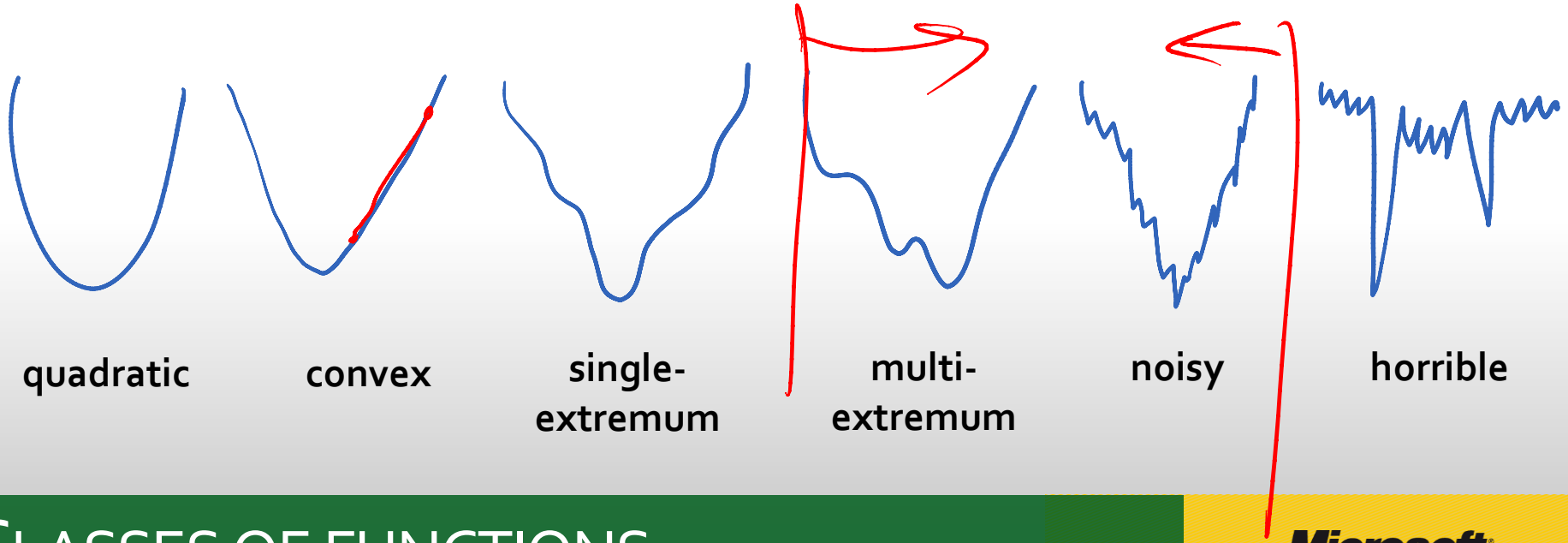
$$f(x): \mathbb{R}^d \mapsto \mathbb{R},$$

Devise strategies for finding x which minimizes f

Given function

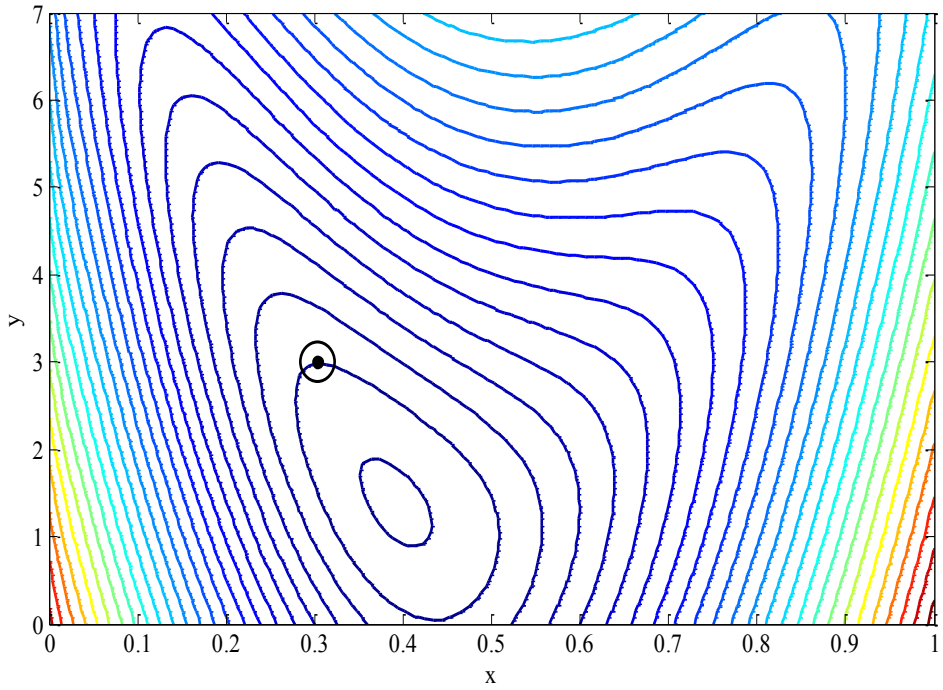
$$f(x): \mathbb{R}^d \mapsto \mathbb{R}$$

Devise strategies for finding x which minimizes f



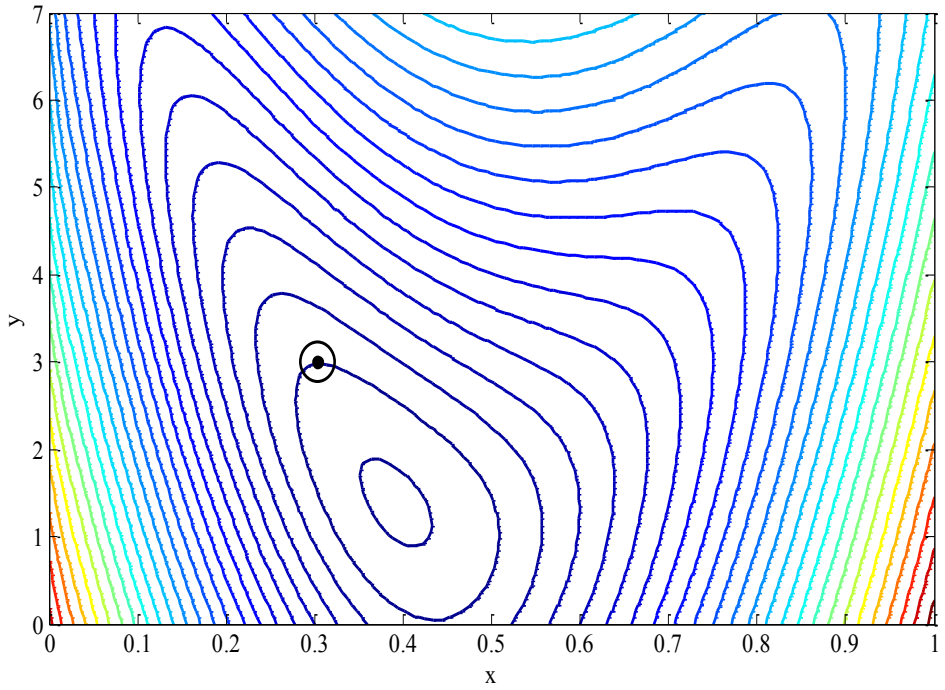
CLASSES OF FUNCTIONS

Microsoft



EXAMPLE

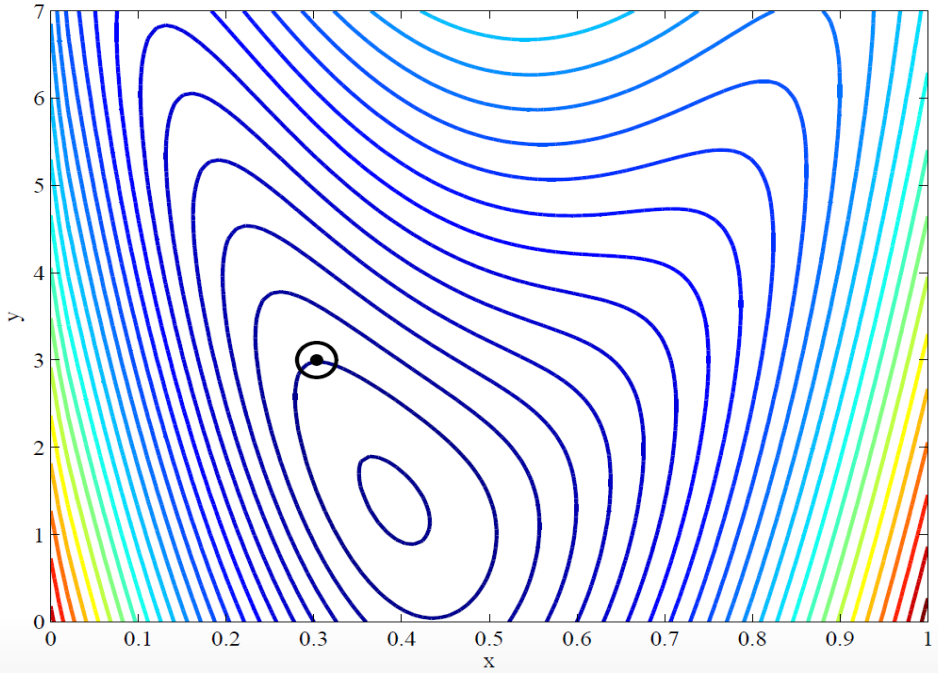
Microsoft



>> print -dmeta

EXAMPLE

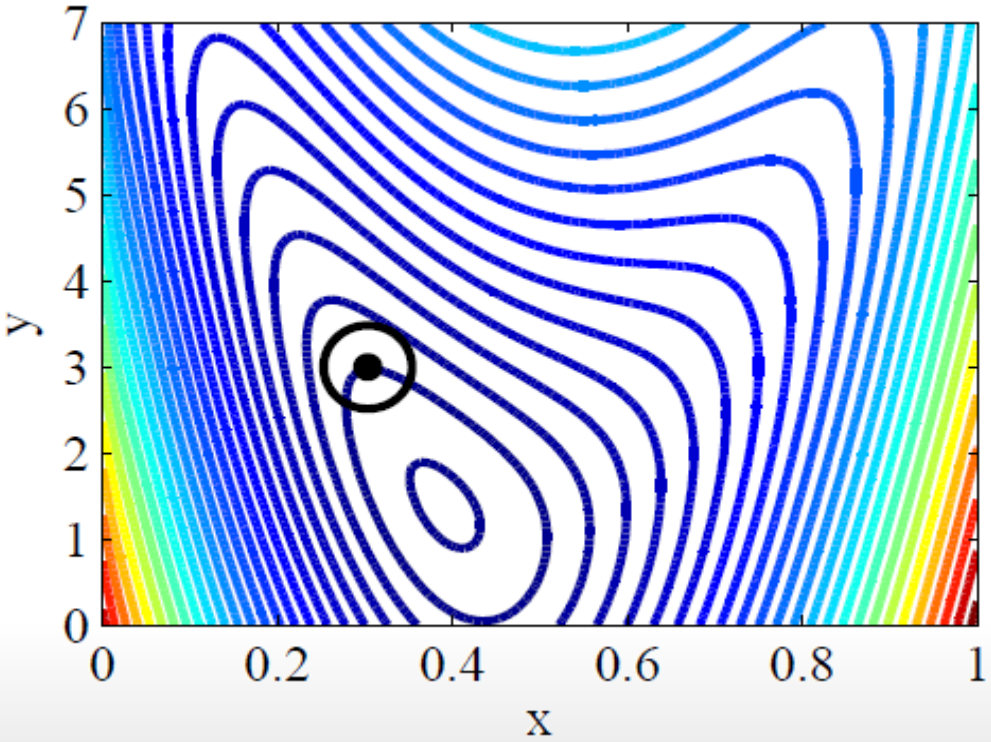
Microsoft



>> print –dpdf % then go to pdf and paste

EXAMPLE

Microsoft




```
>> set(gcf, 'paperUnits', 'centimeters', 'paperposition', [1 1 9 6.6])
>> print –dpdf % then go to pdf and paste
```

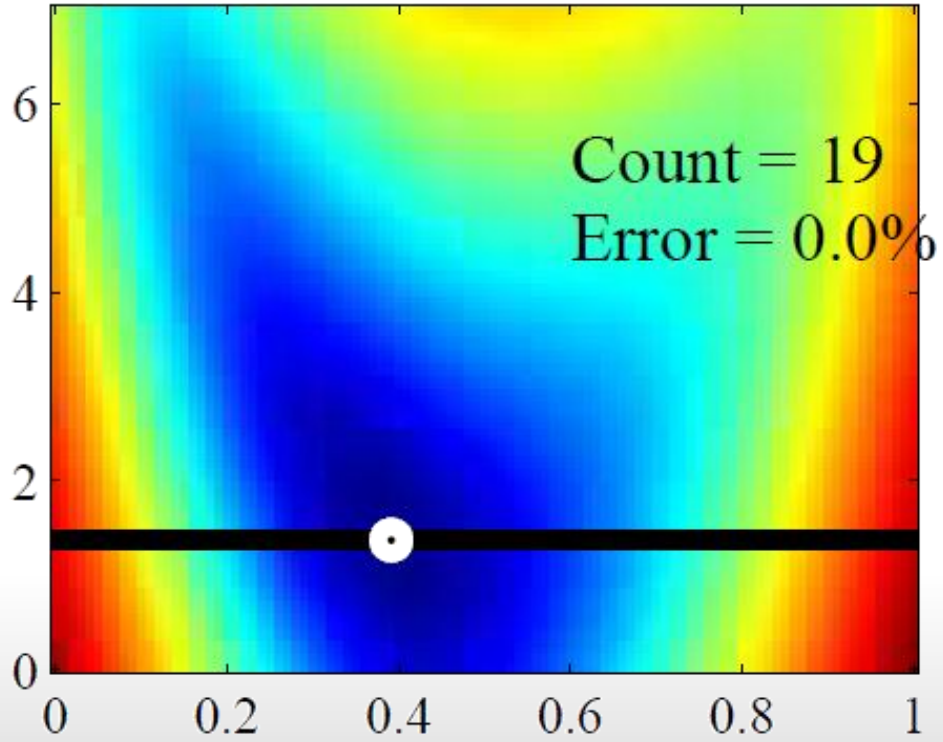
EXAMPLE

Microsoft

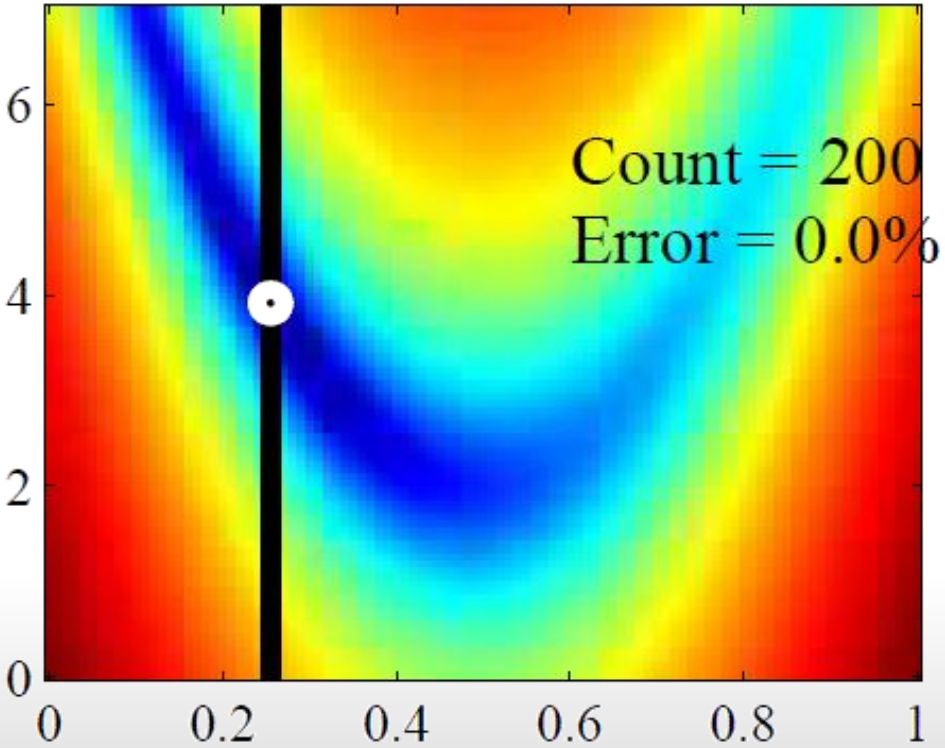
Microsoft Research

switch to matlab…

Microsoft


Count = 19
Error = 0.0%
6
4
2
0
0
0.2
0.4
0.6
0.8
1

Easy

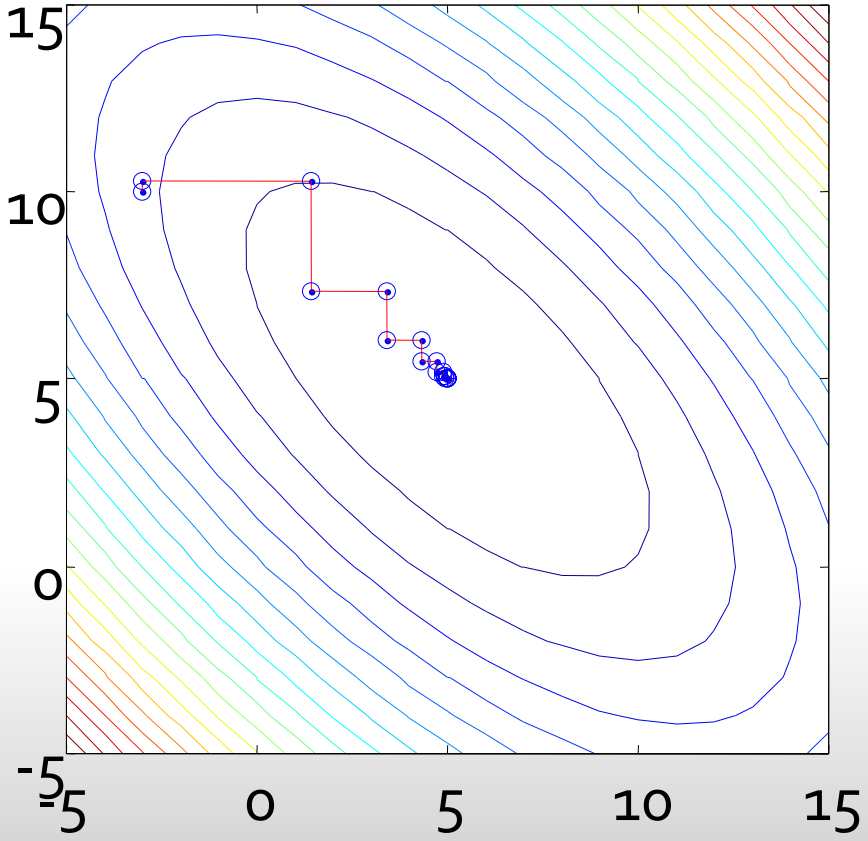

Count = 200
Error = 0.0%
6
4
2
0
0
0.2
0.4
0.6
0.8
1

Hard

- Ignore the hard ones
  - If you search, you'll find plenty of easy examples
  - But your customers won't…
- Or update more than one coordinate at once

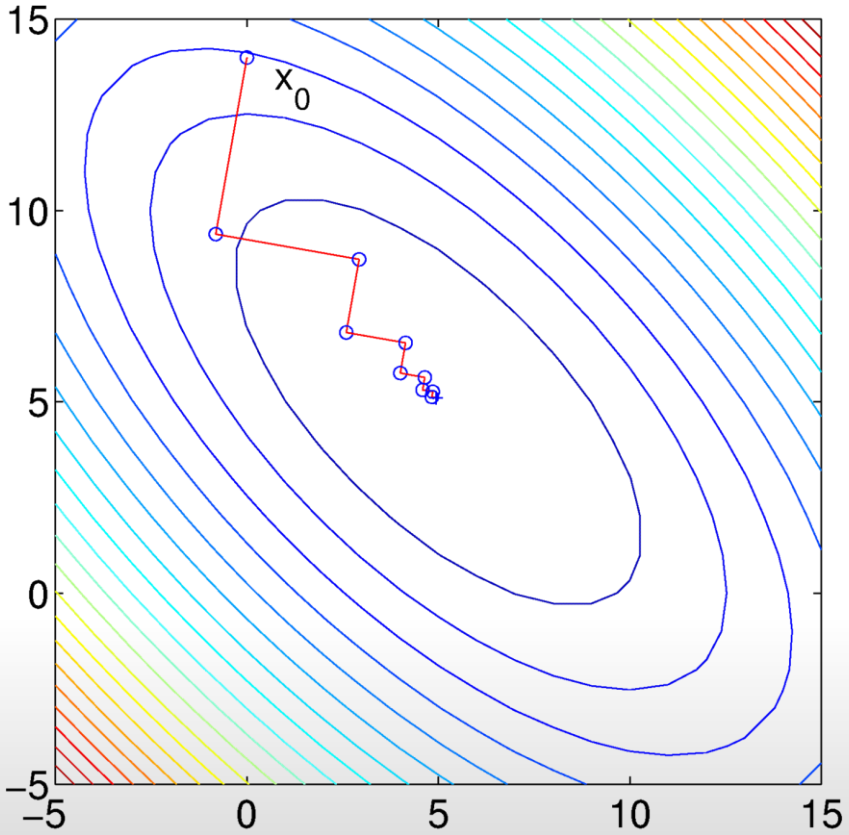
WHAT TO DO?

Microsoft



- Alternation is slow because valleys may not be axis aligned
- So try gradient descent?

Gradient Descent



Steepest descent ($x_0 = [0, 14]$)

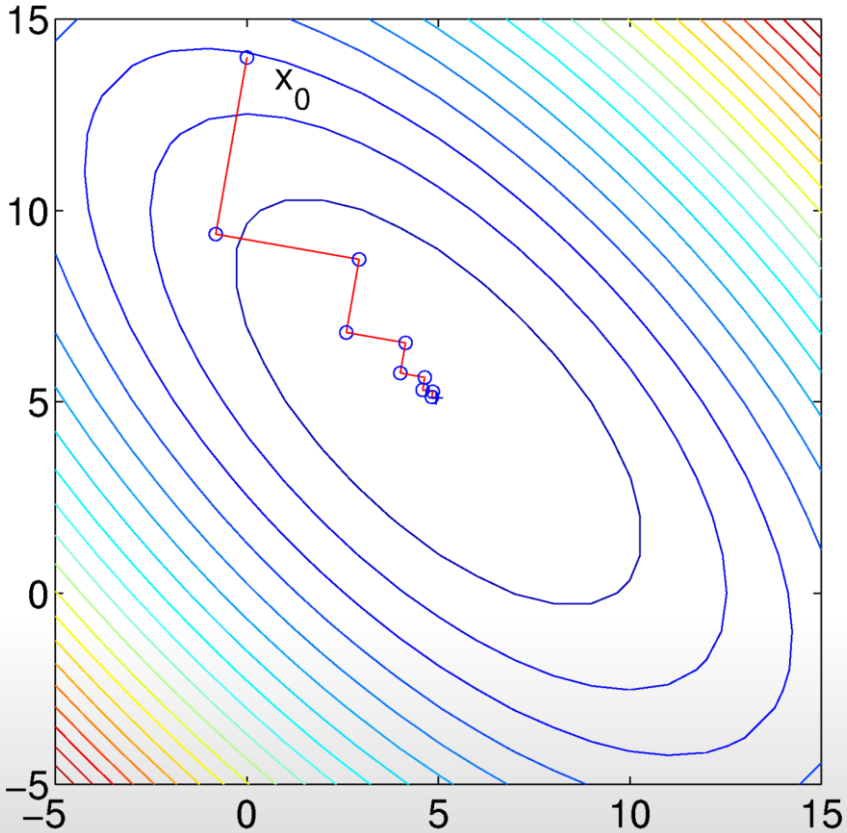
- Alternation is slow because valleys may not be axis aligned
- So try gradient descent?

[val, g] = f(x)

↑ ↑

ℝ ℝ^

Gradient Descent

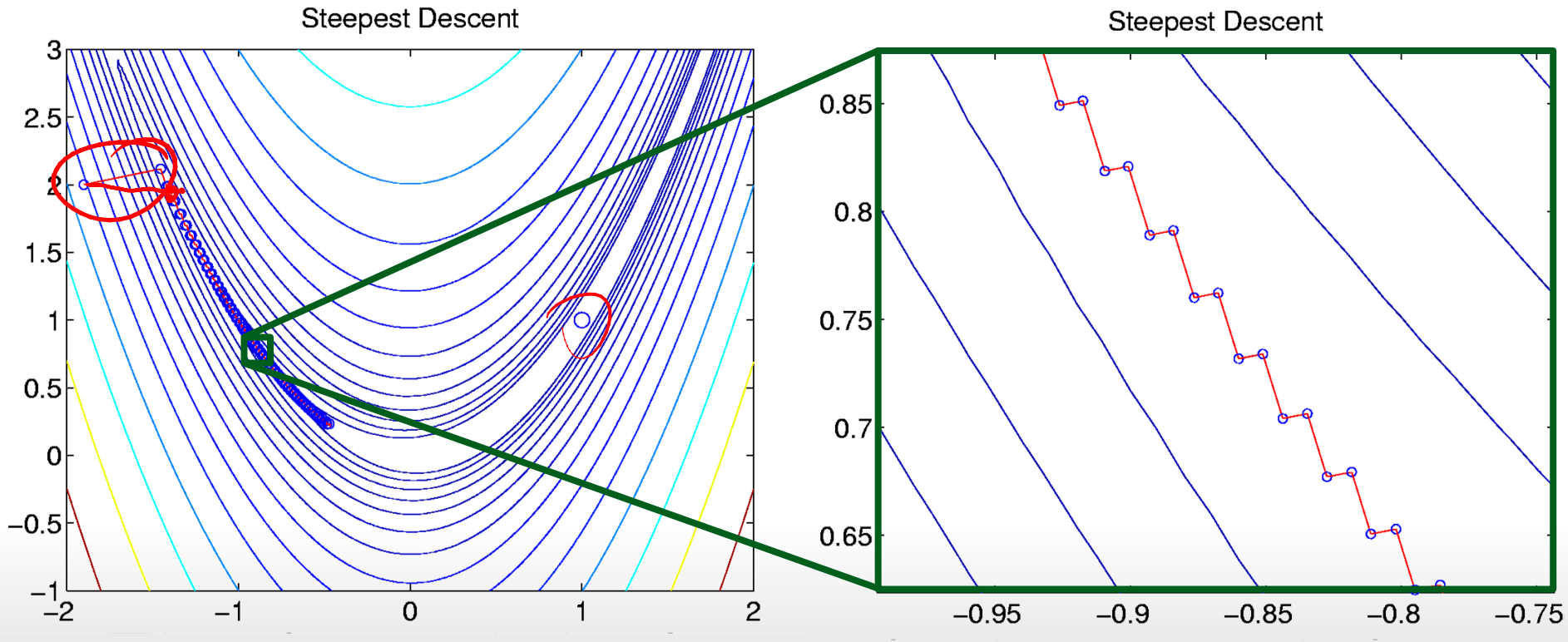


Steepest descent ($x_0 = [0, 14]$)

- Alternation is slow because valleys may not be axis aligned
- So try gradient descent?

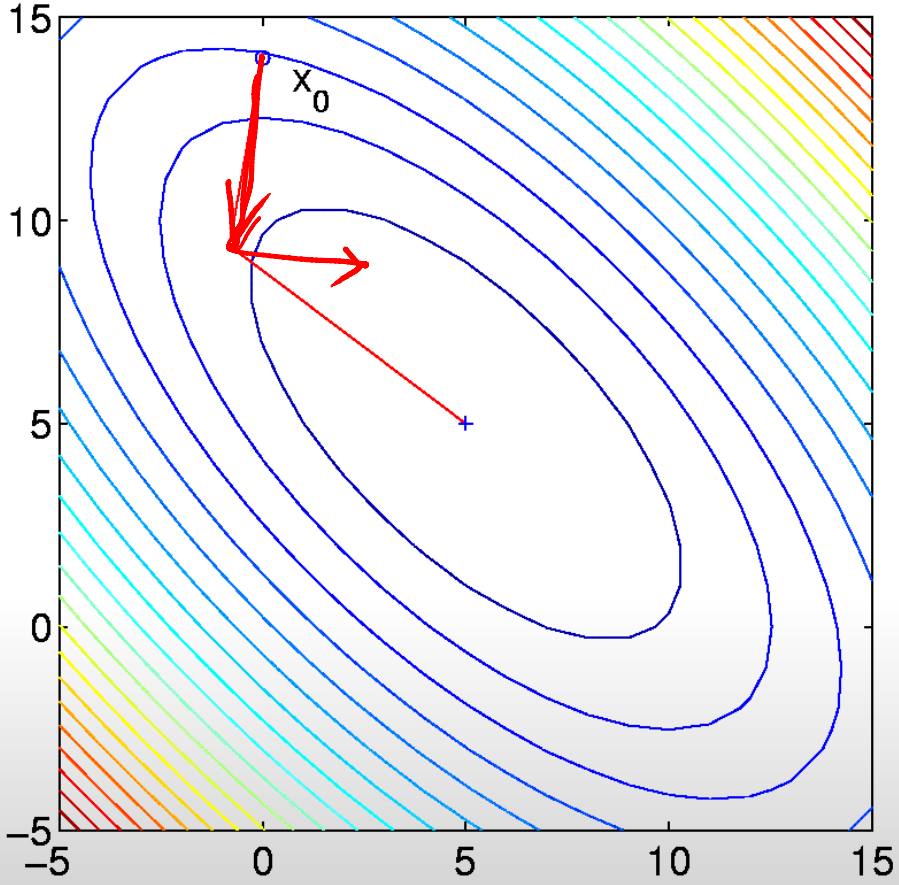
- Note that convergence proofs are available for both of the above
- But so what?

Gradient Descent



AND ON A HARD PROBLEM

Microsoft



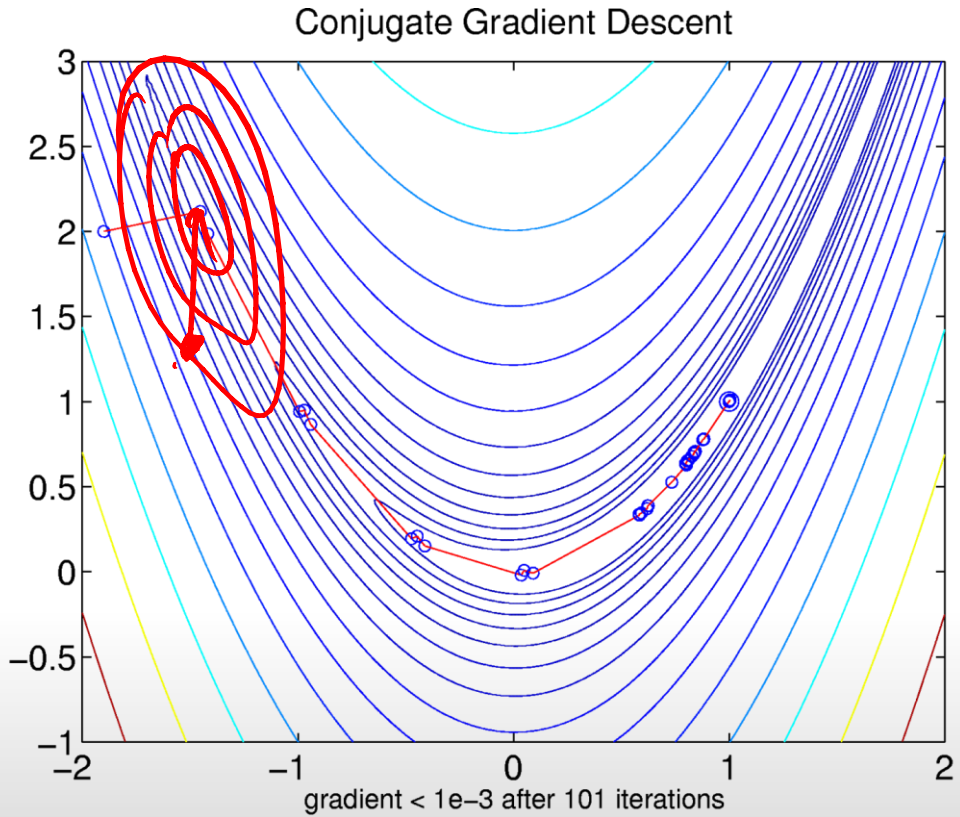
- (Nonlinear) conjugate gradients
- Uses 1st derivatives only
- Avoids "undoing" previous work

SCG

Scaled Conj Grad.

USE A BETTER ALGORITHM

Microsoft



- (Nonlinear) conjugate gradients
- Uses 1st derivatives only
- And avoids "undoing" previous work
- 101 iterations on this problem

USE A BETTER ALGORITHM

Microsoft

Microsoft Research

but we can do better…

Microsoft

- Starting with x how can I choose δ so that $f(x+\delta)$ is better than $f(x)$?
- So compute

$$\min_{\delta} \; f(x+\delta) \qquad g(\delta) = f(x+\delta)$$

- But hang on, that's the same problem we were trying to solve?

- Starting with x how can I choose δ so that $f(x+\delta)$ is better than $f(x)$?
- So compute

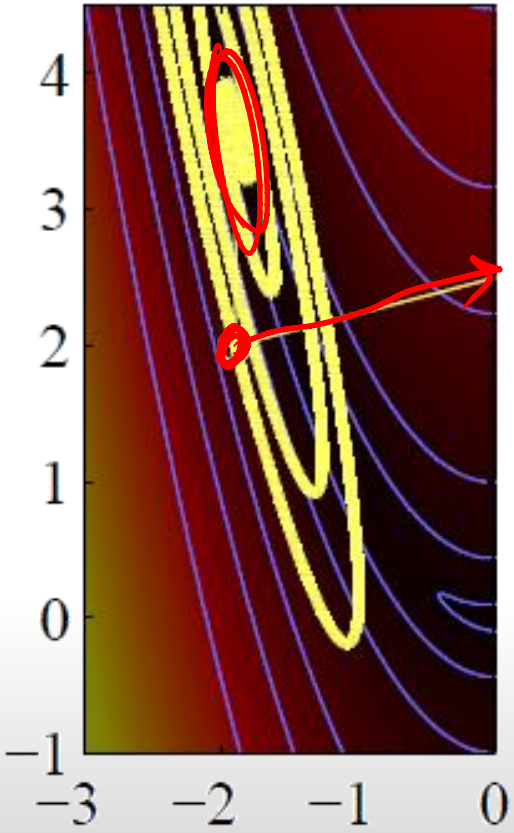
$$\min_{\delta} \; f(x+\delta) \qquad \delta \in \mathbb{R}^n$$

$$\approx \min_{\delta} \; f(x) + \delta^\top g(x) + \tfrac{1}{2}\delta^\top H(x)\delta$$

$$g(x) = \nabla f(x)$$

$$H(x) = \nabla \nabla^\top f(x)$$

USE SECOND DERIVATIVES…



- How does it look?

$$f(x) + \delta^\top g(x) + \frac{1}{2}\delta^\top H(x)\delta$$

$$g(x) = \nabla f(x) \in \mathbb{R}^n$$

$$H(x) = \nabla(\nabla^\top f(x)) \in \mathbb{R}^{n \times n}$$

$$\begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \\ \vdots & \ddots & \frac{\partial^2 f}{\partial x_i \partial x_j} & \vdots \end{bmatrix}$$

USE SECOND DERIVATIVES…

Microsoft

- Choose δ so that $f(x+\delta)$ is better than $f(x)$?
- Compute

$$\min_{\delta} f + \delta^\top g + \frac{1}{2}\delta^\top H\,\delta$$

$\nabla = g + H\delta$

[derive]

$a + bx + \frac{1}{2} x \cdot (c \cdot x)$ $\quad (cx^2 \to cx)$

$$\frac{df}{dx} = b + \frac{1}{2} x \frac{d}{dx}(cx) + \frac{1}{2}\left(\frac{dx}{dx}\right) cx$$

$$= b + \frac{1}{2} x c + \frac{1}{2} c x$$

$$= b + \frac{1}{2} c x + \frac{1}{2} c x$$

Tom Minka – Matrix Derivatives

USE SECOND DERIVATIVES…

Microsoft

- Choose δ so that $f(x+\delta)$ is better than $f(x)$?
- Compute

$$\min_{\boldsymbol{\delta}} f + \boldsymbol{\delta}^{\mathsf{T}} g + \tfrac{1}{2}\boldsymbol{\delta}^{\mathsf{T}} H\, \boldsymbol{\delta}$$

$g + H\delta = 0$

$H\delta = -g$

$\delta = -H^{-1}g$

$$\boldsymbol{\delta} = -H^{-1}g$$

$= -H \backslash g$ || solve(H, g)

USE SECOND DERIVATIVES…

Microsoft

```
demo_taylor_2d(0, 'newton', 'rosenbrock')
demo_taylor_2d(1, 'newton', 'rosenbrock')
demo_taylor_2d(1, 'newton', ‘sqrt_rosenbrock')
```

IS THAT A GOOD IDEA?

Microsoft

- Choose δ so that $f(x+\delta)$ is better than $f(x)$?
- Updates:

$$\boldsymbol{\delta}_{\text{Newton}} = -H^{-1}g$$

$$\boldsymbol{\delta}_{\text{GradientDescent}} = -\lambda g$$

USE SECOND DERIVATIVES…

- Updates:

$$\boldsymbol{\delta}_{\text{Newton}} = -H^{-1}g$$

$$\boldsymbol{\delta}_{\text{GradientDescent}} = -\lambda g$$

- So combine them:

$$\boldsymbol{\delta}_{\text{DampedNewton}} = -(H + \lambda^{-1}I_d)^{-1}g$$

$$= -\lambda(\lambda H + I_d)^{-1}g$$

- λ small ⇒conservative gradient step
- λ large ⇒Newton step

USE SECOND DERIVATIVES…

Microsoft

Levenberg-Marquardt

- Just damped Newton with approximate H
- For a special form of f

$$f(x) = \sum_i f_i(x)^2$$

- where $f_i(x)$ are
  - zero-mean
  - small at the optimum

Levenberg Marquardt

- Just damped Newton with approximate H
- For a special form of f

$$f(x) = \sum_i f_i(x)^2$$

$$\nabla f(x) = \sum_i 2 f_i(x) \nabla f_i(x)$$

$$\nabla \nabla^T f(x) = 2 \sum_i \nabla \left(f_i(x) \cdot \nabla f_i(x) \right) = \sum \nabla f_i(x) \nabla f_i(x) + f_i(x) \nabla^2 f_i(x)$$

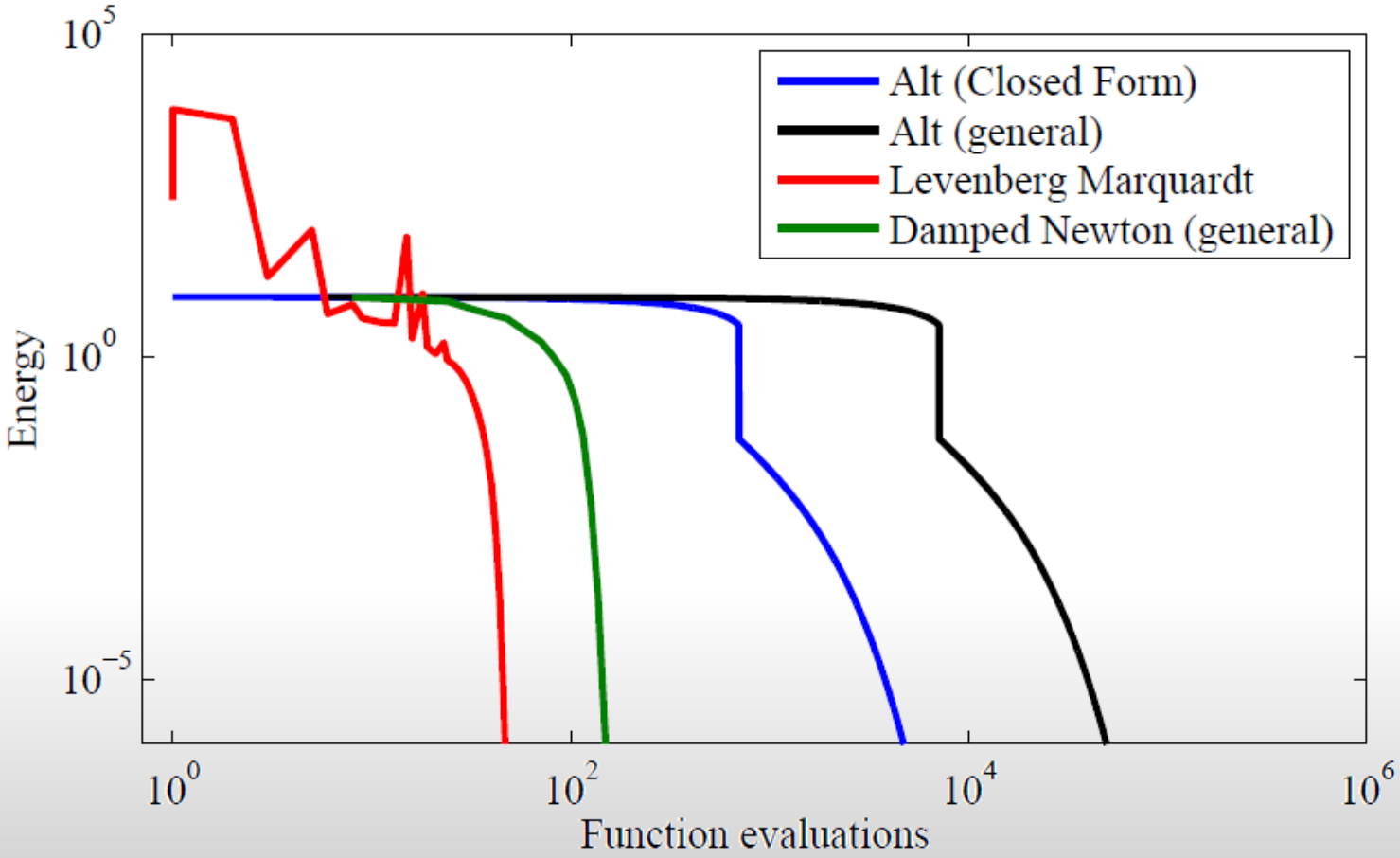
Levenberg Marquardt

- Just damped Newton with approximate H
- For a special form of f

$$f(x) = \sum_i f_i(x)^2$$

$$\nabla f(x) = \sum_i 2 f_i(x) \nabla f_i(x)$$

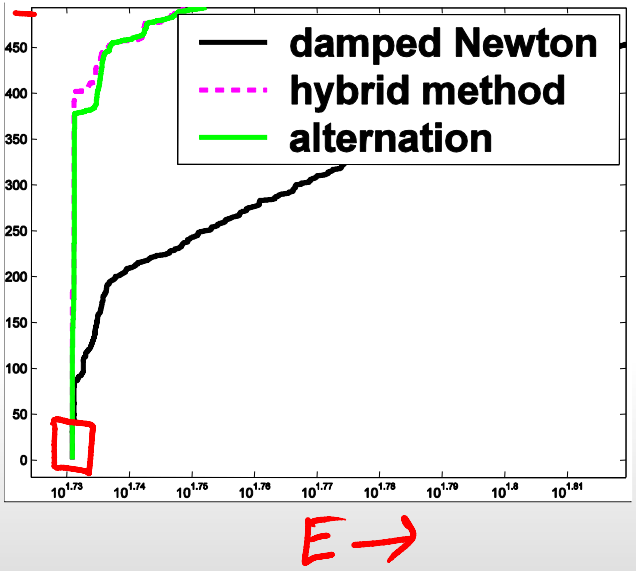
$$\nabla \nabla^\top f(x) = 2 \sum_i f_i(x) \nabla \nabla^\top f_i(x) + \nabla f_i(x) \nabla^\top f_i(x)$$



Conclusion: YMMV

Microsoft

GIRAFFE



500 runs

```
for k=1:500                                   f: ℝ^n → ℝ
    X0 = "randn"(n,1);
    X* = fmin(f, X0);        ← starting pt.
    E(k) = f(X*)
end
plot(sort(E), 1:500)
```

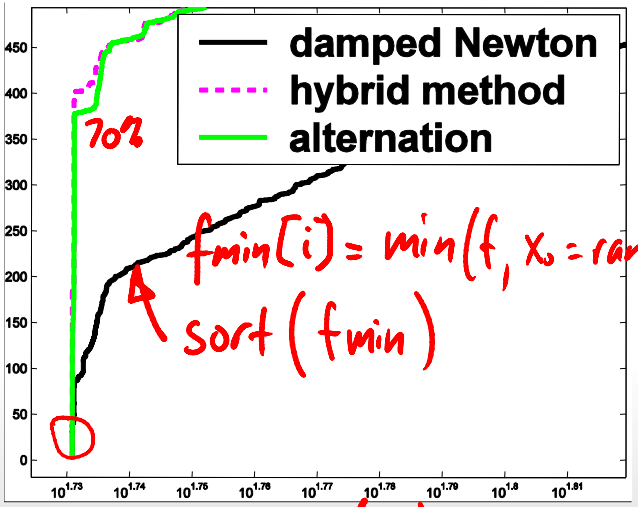
CONCLUSION: YMMV

Microsoft

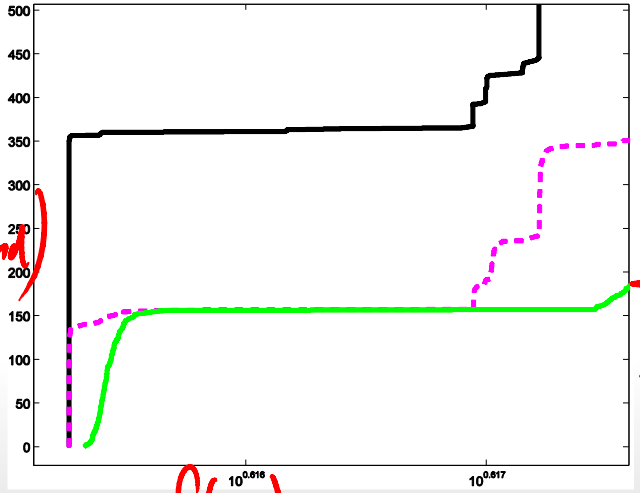
GIRAFFE

FACE

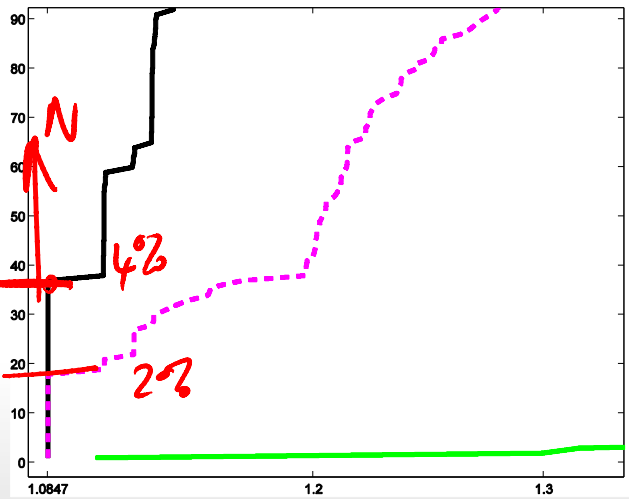
DINOSAUR



500 runs



1000 runs



1000 runs

Buchanan

CONCLUSION: YMMV

Microsoft

Truth

- On many problems, alternation is just fine
  - Indeed always start with a couple of alternation steps
- Computing 2nd derivatives is a pain
  - anyone with compiler/parser experience please contact me.

B*u*T

- Just alternation is fine
  - Unless you're willing to problem-select
- Alternation has a convergence guarantee
- Inverting the Hessian is always $O(n^3)$

There is a universal optimizer