Function minimization in N dimensions

(1) Downhill simplex, a.k.a. amoeba (*NR* §10.4) OCTAVE & MATLAB: fminsearch( )
- O(N^2) storage ⇒ UNSUITABLE FOR VERY LARGE PROBLEMS!
- robust, simple (NO DERIVATIVES USED; CAN TOLERATE DISCONTINUITIES)
- INTUITIVE. MODEST CONVERGENCE RATE.

(2) Conjugate gradient (*NR* §10.6)
- Uses Brent's method (*NR* §10.3) in 1-D
- O(N) storage ==> large problems
- fast/efficient
- requires derivatives (==> smoothness assumed)

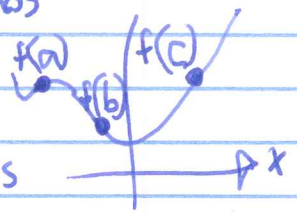
(3) Simulated annealing (*NR* §10.9)
- most robust and simplest of algorithms!
- often finds *global* minima, even of rough functions
- O(N) storage ==> large problems
- slow as molasses in January
- DOESN'T REQUIRE THAT $(\vec{x})$ BE A CONTINUOUS VARIABLE!

1D MINIMIZATION OF SMOOTH FUNCTIONS: $\min_x f(x)$

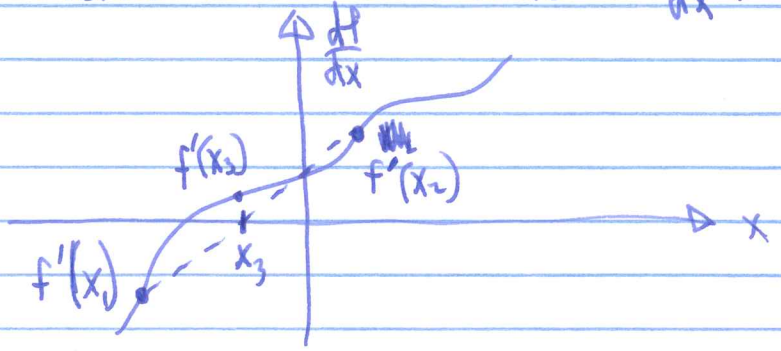
BUILDING BLOCKS:

I. LOCATE A BRACKET: $[a,b,c]$, $f(b) < \{f(a), f(c)\}$

THE GENERAL IDEA IS TO BEGIN WITH A TRIAL GUESS AND SEARCH, STARTING FROM A SMALL LOCAL NEIGHBORHOOD AND EXPANDING DOWNHILL WITH EXPONENTIALLY WIDENING STEPS UNTIL A BRACKET IS LOCATED.



II. SECANT METHOD FOR FINDING ROOTS OF $\frac{df}{dx}$:



SIMPLY INTERPOLATE (OR EXTRAPOLATE) USING 2 POINTS. THEN CHOOSE 2 OF THE 3 POINTS TO PERFORM THE NEXT EXTRAPOLATION. NOTE THAT ONCE THE ROOT HAS BEEN BRACKETED, CONVERGENCE CAN BE GUARANTEED.

III. ROLL THE SECANT METHOD INTO BRENT'S METHOD (NR §10.3)

GIVEN BRACKET $[a,b,c]$ FROM I,

1. $f'(b)$ DECIDES WHICH INTERVAL TO TEST, $[a,b]$ OR $[b,c]$
2. USE 2 LOWEST POINTS OF $[a,b,c]$ TO EXTRAPOLATE USING SECANT METHOD TO POINT d.
3. IS d IN THE INTERVAL SPECIFIED BY 1? IF NOT, REPLACE BY A BISECTION OF THAT INTERVAL
4. EVALUATE $f(d)$ AND UPDATE BRACKET $[a,b,c]$ ACCORDINGLY.
5. RINSE & REPEAT. CONVERGES FAST AS $f \to$ QUADRATIC ON $[a,c]$.

FAST MINIMIZATION OF $f(\vec{x})$ (NR §10.6)

CAN WE ACHIEVE FASTER CONVERGENCE THAN AMOEBA IF WE ASSUME

1. $f(\vec{x})$ RELATIVELY SMOOTH
2. WE HAVE A GOOD INITIAL GUESS

WE BEGIN BY ASSUMING THAT WE HAVE A FAST, ROBUST SOLUTION TO THE 1D PROBLEM:

MINIMIZE $g(x)$ WITH RESPECT TO x.

- NUMERICAL RECIPES: dlinmin()
- MATLAB: fminbnd()
- OCTAVE: fminbnd()

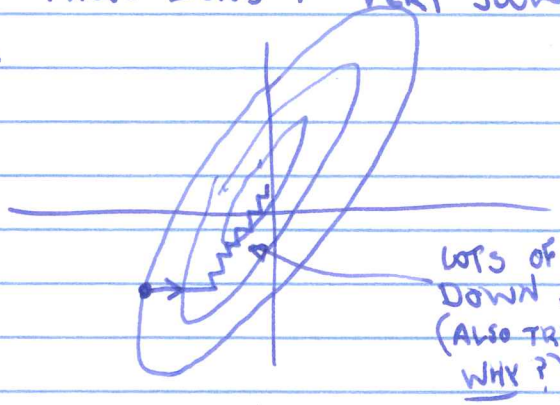
Q: HOW DO WE PARLAY A 1D MINIMIZER INTO AN APPROACH TO OUR GENERAL PROBLEM,

$$\min_{\vec{x}} f(\vec{x}), \quad \vec{x} \in \mathbb{R}^N \; ?$$

POSSIBLE ANSWERS:

- SUCCESSIVE MINIMIZATION ON EACH OF THE COORDINATES OF $\vec{x}$
- MINIMIZE ALONG DIRECTION OF STEEPEST DESCENT.

UNFORTUNATELY, THESE LEAD TO VERY SLOW CONVERGENCE IN MANY CASES.



LOTS OF LITTLE STAIRSTEPS DOWN A NARROW VALLEY (ALSO TRUE OF STEEPEST DESCENT; WHY?)

CONJUGATE GRADIENT

APPLICATIONS:

① SOLVE $A\vec{x} = \vec{b}$ FOR SPARSE A; $N \times N$, SYMMETRIC.
("BICONJUGATE GRADIENT" FOR NON-SYMMETRIC A).

② "NONLINEAR" CONJUGATE GRADIENT TO MINIMIZE $f(\vec{x})$, $\vec{x} \in \mathbb{R}^N$ (OUR APPLICATION).

SEE: "AN INTRODUCTION TO THE CONJUGATE GRADIENT METHOD WITHOUT THE AGONIZING PAIN" BY JONATHAN RICHARD SHEWCHUK (COMPUTER SCIENCE, CMU)
http://solar.physics.montana.edu/kankel/ph567/resources/conjugate_gradient/
(USED WITH PERMISSION)

+ NUMERICAL RECIPES (2^{nd} ED.) § 10.6

KEY STRATAGEM: CHOOSE OPTIMAL DIRECTIONS FOR 1D MINIMIZATION.

TAYLOR EXPANSION:

$$f(\vec{x}) = f(\vec{x}_0) + \left(\nabla f(\vec{x}_0)\right)\cdot(\vec{x}-\vec{x}_0) + \tfrac{1}{2}(\vec{x}-\vec{x}_0)\cdot A(\vec{x}-\vec{x}_0) + \cdots$$

WHERE $A_{ij} = \left.\dfrac{\partial^2 f}{\partial x_i \partial x_j}\right|_{\vec{x}_0}$ IS THE HESSIAN.

NOTE THAT A IS SYMMETRIC. IF $f(\vec{x})$ IS MERELY A QUADRATIC, THEN A IS CONSTANT AND $\nexists$ HIGHER ORDER TERMS.

TO MINIMIZE $f(\vec{x})$, SET $\nabla f = 0$:

$$\nabla f(\vec{x}) = \nabla f(\vec{x}_0) + A(\vec{x}-\vec{x}_0) = 0$$

$$\Rightarrow \boxed{A(\vec{x}-\vec{x}_0) = -\nabla f(\vec{x}_0),} \qquad (0)$$

WHICH IS A PROBLEM OF THE FORM $A\vec{x} = \vec{b}$.

NOW, STARTING WITH AN INITIAL GUESS $\vec{x}_0$:

① PICK N LINEARLY INDEPENDENT DIRECTIONS $\vec{h}_i$.

② MOVE ALONG EACH DIRECTION JUST ONCE, REACHING $\vec{x}_N = \vec{x}$, THE SOLUTION, IN N MOVES:

$$\boxed{\vec{x}_{i+1} = \vec{x}_i + \alpha_i \vec{h}_i} \qquad (i),$$

NEED TO SOLVE FOR α_i.

AT EACH STEP, THE REMAINING ERROR IS $\vec{e}_i = \vec{x}_i - \vec{x}$, SO

$$\boxed{\vec{x}_i = \vec{x} + \vec{e}_i.} \qquad (ii)$$

SO — HOW DO WE DO THIS N-MOVE MIRACLE?!

FIRST IDEA — METHOD OF "NO-NAME" (SHENKNUK, §7).

AT EACH STEP, CHOOSE α_i SO THAT

$$\vec{h}_i \cdot \vec{e}_{i+1} = 0. \ *$$

SINCE $\vec{h}_0 \perp \vec{e}_1$, WE NEED NOT MOVE IN DIRECTION $\vec{h}_0$ AGAIN. SIMILARLY $\vec{h}_1 \perp \vec{e}_2$ AND SO ON.

THIS MEANS WE HAVE CHOSEN $\vec{h}_i$ MUTUALLY ORHOGONAL. *

NOW, ALL WE NEED IS A WAY TO CHOOSE α_i:

USING (i), (ii), AND OUR ORTHOGONALITY REQUIREMENT:

$$\vec{h}_i \cdot (\vec{e}_i + \alpha_i \vec{h}_i) = 0$$

$$\Rightarrow \quad \alpha_i = -\frac{\vec{h}_i \cdot \vec{e}_i}{|\vec{h}_i|^2}.$$

THIS IS USELESS BECAUSE WE DON'T KNOW $\vec{e}_i$ (IF WE DID, WE'D BE DONE!).

---

* NOTE THAT $\vec{h}_i \cdot \vec{e}_{i+1} = 0 \Rightarrow \vec{h}_i \cdot \vec{h}_j = 0$, $i \neq j$;
BUT THE IMPLICATION ALSO GOES THE OTHER WAY, UNDER THE ASSUMPTION THAT $\vec{x}_N = \vec{x}$ (SOLUTION IN N STEPS).
COMPARE THE ARGUMENT ON PAGE 5.

BETTER IDEA: METHOD OF CONJUGATE DIRECTIONS

INSTEAD OF ORTHOGONALITY ($\vec{h}_i \cdot \vec{h}_j = 0$, $i \neq j$)
USE A-ORTHOGONALITY (A-CONJUGACY):

$$\vec{h}_i \cdot A \vec{h}_j = 0, \quad i \neq j. \quad (\star)$$

BY ANALOGY TO PAGE (4), WE'LL ACCOMPLISH THIS BY REQUIRING:

$$\boxed{\vec{h}_i \cdot A \vec{e}_{i+1} = 0} \quad (iii)$$

SHEWCHUK'S FIG. 22 PROVIDES A HELPFUL VISUAL INTERPRETATION OF A-CONJUGACY (NEXT PAGE)

IN FACT, IF WE ASSUME A-CONJUGACY OF THE $\vec{h}$ VECTORS, ($\star$), AND IF WE REQUIRE A SOLUTION IN N STEPS (I.E. $\vec{x}_N = \vec{x}$), THEN (iii) IS NECESSARY. QUICK PROOF:

$$\vec{h}_i \cdot A\left(-\sum_{j=i+1}^{N} \alpha_j \vec{h}_j\right) = 0 \quad \text{BY } \star.$$

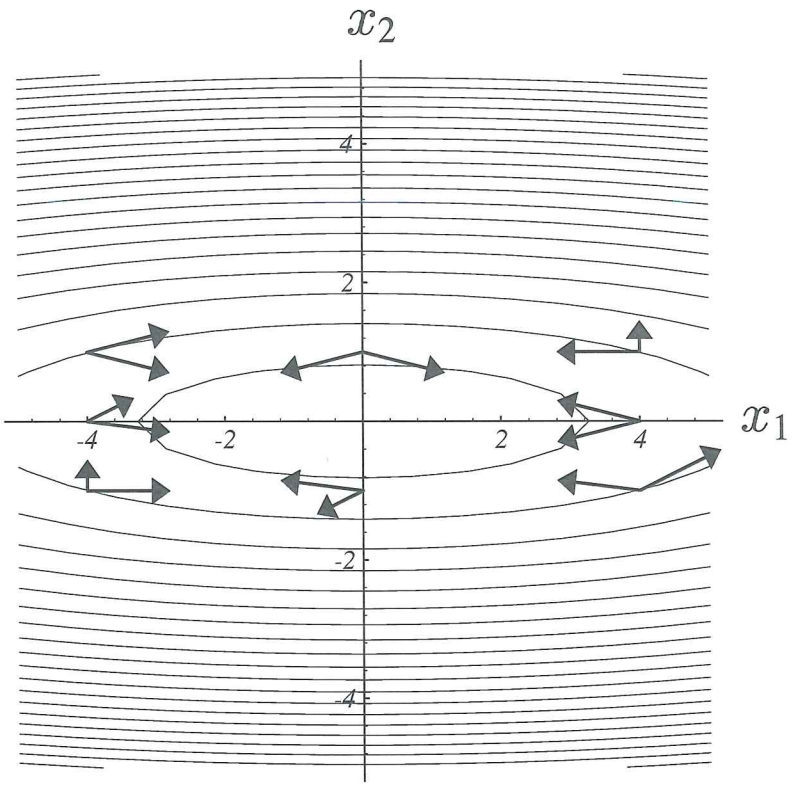
BUT THE SUM IS BY DEFINITION

$$\vec{e}_{i+1} \equiv \vec{x}_{i+1} - \vec{x} = \vec{x}_{i+1} - \vec{x}_N$$

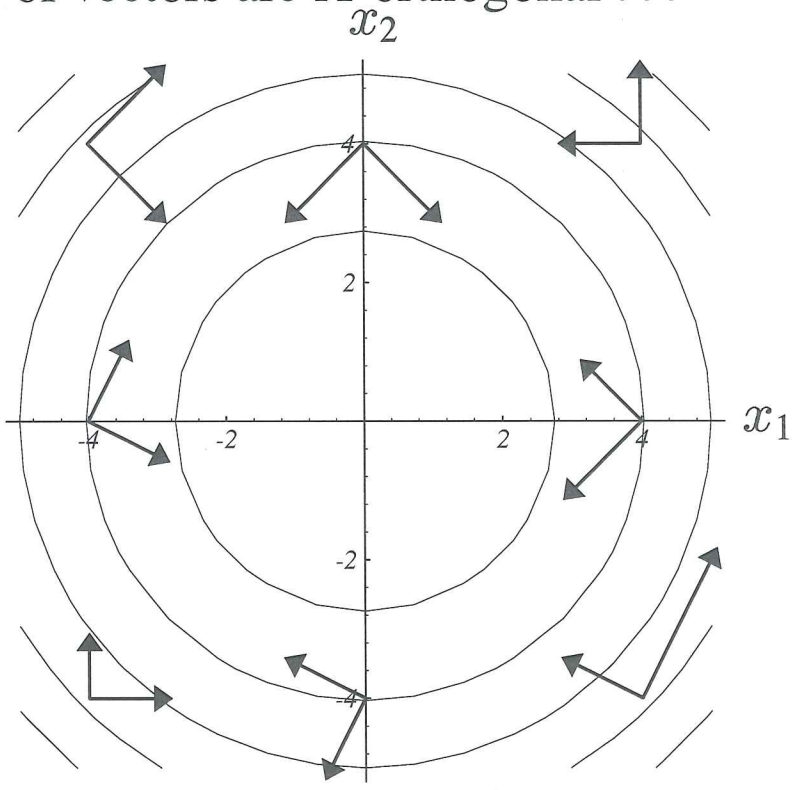
($\vec{x}$ REACHED IN N STEPS!)

$$= -\sum_{j=i+1}^{N} \alpha_j \vec{h}_j \quad \text{(BY EQUATION } i\text{)}.$$

$$\therefore \vec{h}_i \cdot A(\vec{e}_{i+1}) = 0 \quad \text{QED}.$$



These pairs of vectors are A-orthogonal . . .



. . . because these pairs of vectors are orthogonal.

WE NOW KNOW HOW TO CHOOSE $\vec{h}_i$, BUT THE REAL QUESTION IS HOW TO CHOOSE α_i.

BEGIN WITH (iii):

$$\vec{h}_i \cdot (A\vec{e}_{i+1}) = 0$$

USING (ii),

$$A(\vec{x}_{i+1} - \vec{x}) \cdot \vec{h}_i = 0$$

COMBINING WITH (0), AND (i):

$$\nabla f(\vec{x}_{i+1}) \cdot \frac{d}{d\alpha_i} \vec{x}_{i+1} = 0$$

USING (i) AGAIN,

$$\nabla f(\vec{x}_i + \alpha_i \vec{h}_i) \cdot \frac{d}{d\alpha_i}(\vec{x}_i + \alpha_i \vec{h}_i) = 0$$

$$\Rightarrow \boxed{\frac{d}{d\alpha_i} f(\vec{x}_i + \alpha_i \vec{h}_i) = 0}$$ CHAIN RULE

INTERPRETATION: CHOOSE α_i BY 1D MINIMIZATION OF f ALONG THE $\vec{h}$ DIRECTION.

A SET OF MUTUALLY A-CONJUGATE $\vec{h}_i$ MAY BE CHOSEN A PRIORI USING "GRAM SCHMIDT CONJUGATION" (SHEWCHUK §7.2).

— THIS COMPLETES THE METHOD OF CONJUGATE DIRECTIONS —

CONJUGATE GRADIENT METHOD (FLETCHER-REEVES)

CHOOSE STARTING POINT $\vec{x}_0$.

LET $\vec{h}_0 = \vec{g}_0 \equiv -\nabla f(\vec{x}_0)$ ... DOWNHILL.

FOR $i = 0$ TO n {

MINIMIZE $f(\vec{x}_i + \alpha_i \vec{h}_i)$ W.R.T. α_i.

$$\vec{x}_{i+1} = \vec{x}_i + \alpha_i \vec{h}_i$$

$$\vec{g}_{i+1} \equiv -\nabla f(\vec{x}_{i+1})$$

$$\vec{h}_{i+1} = \vec{g}_{i+1} + \underbrace{\frac{|\vec{g}_{i+1}|^2}{|\vec{g}_i|^2}}\, \vec{h}_i \qquad *$$

WITHOUT THIS TERM, WE WOULD HAVE THE METHOD OF STEEPEST DESCENT.

}

*IT CAN BE SHOWN THAT $\vec{h}_i \cdot A\vec{h}_j = 0$, $i \neq j$.

POLAK-RIBIERE: REPLACE * WITH

$$\vec{h}_{i+1} = \vec{g}_{i+1} + \frac{(\vec{g}_{i+1} - \vec{g}_i) \cdot \vec{g}_{i+1}}{|\vec{g}_i|^2}\, \vec{h}_i$$

THIS OFFERS IMPROVEMENT AT EACH n ITERATIONS FOR NONLINEAR PROBLEMS.

NR § 10.6 frprmn() USES POLAK-RIBIERE.

QUICK EXAMPLE BY HAND...

$f(x,y) = x^2 + 10y^2$

LONG, NARROW VALLEY WITH MINIMUM AT (0,0)

$$\vec{x}_0 = [10, 1]$$

$$\nabla f = \begin{pmatrix} 2x \\ 20y \end{pmatrix}$$

$$\vec{h}_0 = \vec{g}_0 = -\nabla f(\vec{x}_0) = -\begin{pmatrix} 20 \\ 20 \end{pmatrix}$$

MINIMIZE: $f\left(\begin{bmatrix} 10 \\ 1 \end{bmatrix} + \alpha \begin{bmatrix} -20 \\ -20 \end{bmatrix}\right) = (10-20\alpha)^2 + 10(1-20\alpha)^2$

$$\frac{\partial f}{\partial \alpha} = 0 = -40(10-20\alpha) - 400(1-20\alpha)$$

$$= -800 + 8800\alpha$$

$$\alpha = \frac{1}{11}$$

NOW, $\vec{x}_1 = \vec{x}_0 + \alpha \vec{h}_0 = \begin{pmatrix} 10 - 20/11 \\ 1 - 20/11 \end{pmatrix} = \begin{pmatrix} 90/11 \\ -9/11 \end{pmatrix}$

$$= \frac{9}{11}\begin{pmatrix} 10 \\ -1 \end{pmatrix}$$

THEN,

$$\vec{g}_1 = -\nabla f(\vec{x}_1) = \frac{9}{11}\begin{pmatrix} -20 \\ +20 \end{pmatrix}$$

$$\vec{h}_1 = \vec{g}_1 + \frac{g_1^2}{g_0^2}\vec{h}_0 = \frac{9}{11}\begin{pmatrix} -20 \\ 20 \end{pmatrix} + \frac{\frac{81}{121}(800)}{800}\begin{pmatrix} -20 \\ -20 \end{pmatrix}$$

$$= \frac{9}{121}\begin{pmatrix} -220 - 180 \\ 220 - 180 \end{pmatrix} = \frac{9}{121}\begin{pmatrix} -400 \\ 40 \end{pmatrix}$$

(NOTE $\vec{h}_1 = (\text{SCALAR})\,\vec{x}_1$!)

$$\Rightarrow \vec{x}_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

C-G IS PRETTY GOOD, AND IT'S NOT INTIMIDATING TO CODE UP YOURSELF IF YOU NEED IT.

THERE ARE OTHER TYPES OF ALGORITHMS THAT TAKE ADVANTAGE OF THE CONCEPT OF OPTIMAL DIRECTIONS MUCH LIKE C-G:

- QUASI-NEWTON METHODS ("UNCONSTRAINED" OPTIMIZATION, LIKE C-G)
  SOMEWHAT FASTER THAN CG, AND REQUIRE A LITTLE MORE STORAGE. BFGS IS A POPULAR MEMBER OF THIS CLASS.
- SEQUENTIAL QUADRATIC PROGRAMMING (SQP)
  FORMULATED USING A LAGRANGIAN, THIS TECHNIQUE ALLOWS THE INCORPORATION OF EQUATIONS OF CONSTRAINT. HANDY!

OCTAVE: `sqp()` — CONSTRAINED (SQP)
`fminunc()` — UNCONSTRAINED (BFGS)

MATLAB: `fmincon()` — CONSTRAINED MINIMIZATION
(INCORPORATES SQP AND SEVERAL OTHER ALGORITHMS AS SELECTABLE OPTIONS)
`fminunc()` — UNCONSTRAINED MINIMIZATION
(QUASI-NEWTON OR TRUST REGION ALGORITHM)

EXAMPLE: N CHARGES ON A THIN, FINITE WIRE: FIND THE ELECTROSTATIC EQUILIBRIUM CONFIGURATION.