Scientific Computing CS660 Fall '11
Notes after the midterm

Angjoo Kanazawa

December 2, 2011

1 Class 13 October 25th 2011

1.1 HW2 answers

Problem 1b: to compute many N, compute $S_N = \phi(x_1) + \cdots + \phi(x_n)$, take N samples, divide by N will give us I_N. THe error is $E_N = I - I_N$. TO generate I_{N+1}, take $S_{N+1} = S_N + \phi(x_{N+1})$ to compute I_{N+1}. But this way E_{N+1} is not independent from E_N. (Left top figure on the answers).

(Doing it entirely independently (how i did it) is the lower figure) Plotted $\log_N$ vs $\log(\frac{\sigma}{\sqrt{N}}$ The observation was that it somewhat follows the bound.

Our bound, $\frac{\sigma}{\sqrt{N}}$, comes from chebychev $P(|\frac{S_N}{N} - \mu| \geq c) \leq \frac{\sigma^2}{Nc^2}$. So if we want this P to be 95%, then $c \leq \frac{\sigma}{\sqrt{0.5N}}$. So loosely speaking, the error $E_N = I - I_N \approx \frac{\sigma}{\sqrt{0.5N}}$. The point of the question was that it's not exact.

On Page 3 of the solution, there's the definition of histogram for question 1c.

Problem 2: In P1, the error was $E_N = \frac{\sigma^2}{\sqrt{N}}$ If the variance is σ^2, std σ, then log of E_N is $\log\sigma - \frac{\log(0.5N)}{2}$.

In here, we expected the error to be $\frac{\sigma^2}{\sqrt{N}}$. If that were to happen, same derivation would give $\log(E_N) \approx 2\log(\sigma) - \frac{\log(0.5N)}{2}$. The blue line (boundary) would've been $2\log(\sigma) - \frac{\log(N)}{2}$ and E_N will hover around there. But it didn't work!! Because $sin(\pi X) = sin(\pi(1 - X))$, nothing changed. Function had to be monotone.

midterm: Equally distributed, 3 topics: floating pt/round off errors, mc, matrix factorization (big picture). (no newton's method, no machine representation of numbers)

1.2 Eigenvalue Hessemberg

Why do we want to reduce the matrix to Hessemberg form: Because doing QR on Hessenberg is $\mathcal{O}(n^2)$ instead of $\mathcal{O}(n^3)$.

How to make a hessemberg matrix: A is a big full matrix. Now QA will make everything below the second entry of first column to 0. You can do that with Householder transformation, or givens rotation.

However we do this, QAQ^T will not mess up the 0s.

$$QAQ^T = [Q(QA)^T]^T$$

Applying to Q to $(QA)^T$ will leave the first row alone, so $[Q(QA)^T]^T$ still keeps the first column's entry below 2 to 0.

Exercise, go back, do: can we perform QAQ^T where Q is a householder transformation s.t. QAQ^T is a first step hessemberg matrix. The reason why this works is because householder is in the $n-1$ subblock, so it doesn't do anything row 1. (It's like the 2nd household transformation in the QR reduction)

QR iteration for the eigenvalue problem:

0 Reduce A to hessenberg form, formally, $PAP^T = H$, where P orthogonal. (do $n-1$ householder transformation P_2).$A_0 = H$

1-k Iterate, $A_K = Q_K R_K$, $A_{K+1} = R_K Q_K$

Doing this $A = QR$ is $\mathcal{O}(n^3)$, but to do this until conversion (see ϵ in below diagonal, is n, so total $\mathcal{O}(n^4)$ (without making A into upper hessenberg). If A wer hessenberg, doing QR is $\mathcal{O}(n^2)$ instead of $\mathcal{O}(n^3)$, so the total cost is $\mathcal{O}(n^3) + \mathcal{O}(nn^2) = \mathcal{O}(n^3)$ with hessemberg.

2 Class 14 November 1st

Midterm: median 22, mean 20.8, max:29

2.1 Midterm

Problem 1: Given $Ax = b$, $A\hat{x} = b + \delta b$: $A(x - \hat{x}) = b - b\delta b$ so

$$x - \hat{x} = A^{-1}(\delta b)$$
$$||x - \hat{x}|| \leq ||A^{-1}||||\delta b||$$

And

$$b = Ax$$
$$||b|| \leq ||A||||x||$$
$$\frac{1}{||x||} \leq \frac{||A||}{||b||}$$

So

$$\frac{||x - \hat{x}||}{||x||} \leq ||A||||A^{-1}||\frac{||\delta b||}{||b||}$$

Problem 2:

a QR factorization by gram-schimid

b Making $r_{ik} =< \hat{q}_k, q_i >$ from $r_{ik} =< a_k, q_i >$, doesn't change anything. Because

$$r_{23} = \langle \hat{q}_3, q_2 \rangle$$
$$= \langle a_3 - r_{13}, q_2 \rangle$$
$$= \langle a_3, q_2 \rangle - r_{13}\langle q_1, q_2 \rangle$$

c Real question is: Will q_3 be orthogonal to q_2 and q_1? Look at $\hat{q}_3$ after ith loop is done. $\hat{q}_3 = \langle a_3 - r_{13}q_1 - r_{23}q_2, q2 \rangle = \langle a_3, q_2 \rangle - r_{13}\langle q_1, q_2 \rangle = -r_{23}\langle q_2, q_2 \rangle$ Problem is $\langle q_1, q_2 \rangle$ could be non-zero because of round-off error. But if we did the second way, $\langle \hat{q}_3, q_2 \rangle = \langle a_3 - r_{13}q_1, q_2 \rangle - r_{23}\langle q_3, q_2 \rangle$, is a little bit less-sensitive to round-off errors. Called the modified-Gram Schmid. (This can't be run parallel)

2.2 Singular Value Decomposition

Let $A \in \mathbf{R}^{m \times n}$. A can be factored as

$$A = U\Sigma V^T$$

, U is m by m, sigma is m by n, a diagonal matrix, where the entries are $sig_1, \ldots, sig_n$, everywhere else 0. sig_i's are called the singular values, and $\sigma_1 \leq sig_2 \leq \cdots \leq \sigma_n \leq 0$. V^T is n by n. Where $U^TU = I_m$ and $V^TV = I_n$.

How to use: **Least Squares**: minimize $||b - Ax||_2$, find x.

$$
\begin{aligned}
||b - Ax||_2^2 &= \langle b - Ax, b - Ax \rangle \\
&= \langle b - U\Sigma V^T x, b - U\Sigma V^T x \rangle \\
&\text{let } \hat{x} = V^T x \\
&= \langle U(U^T b - \Sigma\hat{x}), U(U^T b - \Sigma\hat{x}) \rangle \\
&= \langle U^T U(U^T b - \Sigma\hat{x}), U^T U(U^T b - \Sigma\hat{x}) \rangle \\
&= \langle U^T b - \Sigma\hat{x}, U^T b - \Sigma\hat{x} \rangle \\
&\text{let } \hat{b} = U^T b \\
&= \langle \hat{b} - \Sigma\hat{x}, \hat{b} - \Sigma\hat{x} \rangle \\
&= ||\hat{b} - \Sigma\hat{x}||_2^2
\end{aligned}
$$

This is

$$
\begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \vdots \\ \hat{b}_m \end{pmatrix} - \begin{pmatrix} \sigma_1\hat{x}_1 \\ \vdots \\ \sigma_n\hat{x}_n \\ 0 \\ \vdots 0 \end{pmatrix}
$$

So to minimize, set $\sigma_k\hat{x}_k = \hat{b}_k$, or

$$
\hat{x}_k = \frac{\hat{b}_k}{\sigma_k}
$$

The norm of minimum norm solution is $\sqrt{\hat{b}_{n+1}^2 + \cdots + \hat{b}_m^2}$, provided $\sigma_n > 0$. If all $\sigma_j > 0$, then $\hat{x}, x$ are unique. aka A is of full rank. If some of them are zero, then $\exists$ multiple solutions $\hat{x}$ and x. Rank of A is the index of smallest nonzero σ_js.

To solve this problem, compute U, V and Σ, fing $\hat{b}$, find $\hat{x}$, find $x = V\hat{x}$.

Remember: least squares solution can be found from $A^T Ax = A^T b$:

$$
\begin{aligned}
A^T Ax &= A^T b \\
V\Sigma^T U^T U\Sigma V^T x &= V\Sigma^T U^T b \\
V[\Sigma_1 0][\Sigma_1; 0]V^T x &= V\Sigma^T U^T b \\
V\Sigma_1^2 V^T x &= V[Sig_1 0][U^T b] \\
\Sigma_1^2 V^T x &= \Sigma_1[U^T b]
\end{aligned}
$$

This is just like the one before where $V^T x = \hat{x}$, $\hat{b} = U^T b$.

2.3 Pseudo-inverse

Of a rectangular matrix A, is written:

$$
A^\dagger = (A^T A)^{-1} A^T
$$

$(A^TA)^{-1}$ is square times wide rectangle, so a wide rectangle. Property: $A^\dagger A = (A^TA)^{-1}A^A = I$.

Given $A = U\Sigma V^T$,

$$\begin{aligned} A &= U\Sigma V^T \\ A^TA &= V\Sigma_1^2 V^T \\ (A^TA)^{-1} &= (V\Sigma_1^2V^T)^{-1} = (V^T)^{-1}(\Sigma_1^2)^{-1}V^{-1} \\ &= V\Sigma_1^{-2}V^T \end{aligned}$$

So

$$\begin{aligned} A^\dagger = (A^TA)^{-1}A^T &= V\Sigma_1^2V^TV([\Sigma_1 0])V^T \\ &= V([\Sigma_1^{-1}0])U^T \end{aligned}$$

3 Class 16 November 3rd 2011

3.1 Continue on SVD

Given $A \in \mathbf{R}^{m\times n}$, $m > n$, $A = U\Sigma V^T$, A full rank. $A = [U_1U_2][\Sigma_1; 0]V^T$ wher U_1 is m by n, U_2 is m by $m - n$. And

$$\begin{aligned} U^TU &= [U_1^T; U_2^T][U_1U_2] \\ &= \begin{pmatrix} U_1TU_1 & U_1^TU_2 \\ U_2^TU_1 & U_2^TU_2 \end{pmatrix} & = I = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} \end{aligned}$$

Because $U_1^TU_1 = I_n, U_2^TU_2 = I_{m-n}$, and $U_1^TU_2 = 0$

$A^T = V[\Sigma_1 0][U_1^T; U_2^T] = V\Sigma_1U_1^T$. Consider $w \in \mathbf{R}^m$, look at $A^Tw = V\Sigma_1U_1^Tw$. If $w \in range(U_2)$, $w = U_2z$ for some z, so

$$A^Tw = V\Sigma_1U_1^TU_2z = 0$$

i.e. $range(U_2) \subseteq null(A^T)$, $\supseteq$ is also true. So

$$null(A^T) = range(U_2).$$

Pf$[range(U_2) \supseteq null(A^T)]$ Suppose $w \in null(A^T)$, i.e. $A^Tw = 0$. $w = U_z$, for some z

$$\begin{aligned} w &= U_z \\ &= U_1z_1 + U_2z_2A^Tw & = V\Sigma_1U_1^T(U_1z_1 + U_2z_2) = V\Sigma_1z_1 + 0 \end{aligned}$$

If $z_1 \neq 0 \Rightarrow \Sigma_1z_1 \neq 0 \Rightarrow\Leftarrow$ because the assumption $range(U_2) \subseteq null(A^T)$, A^Tw should be 0. So it has to be that $z_1 = 0$. $\therefore w \in range(U_2)$

Another point: COlumns of U_1 span the range of A. Range of $A := \{Av | v \in \mathbf{R}^n\}$. Given $Av = U_1(\Sigma_1V^Tv) \subseteq range(U_1)$

We can see that $range(A)$ and $null(A^T)$ are orthogonal to each other. (This is a known fact in linear algebra, but SVD makes it intuitive)

end of matrix factorization!

4 New topic:Optimization

Given $f : \mathbf{R}^n \to \mathbf{R}$, we want to find $x \in \mathbf{R}^n s.t. f(x)$ is minimal (i.e. $-f(x)$ is maximal). We may want to find:

- Global minimum $\hat{x} s.t. f(\hat{x}) le f(x) \forall x$
- Local minimum $\hat{x} s.t. f(\hat{x}) le f(x) \forall x$ near $\hat{x}$ (rigorously: near means in some radius r.).

In this class we're looking for a local minimum.

If x is a local minimum value, consider $\phi(\alpha) = f(x + \alpha d)$, where $d \in \mathbf{R}^n$, some other vector, $\alpha \in \mathbf{R}$. Now look $\phi(\alpha)$'s taylor series: $\phi(0) + \phi'(0)\alpha + \mathcal{O}(\alpha^2)$

$$\phi'(\alpha) = [\nabla f(x + \alpha d)]^T d$$
$$\phi'(0) = [\nabla f(x + \alpha d)]^T d$$

(Where $\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} & \vdots & \frac{\partial f}{\partial x_n} \end{pmatrix}$)

Suppose $\nabla f(x) \neq 0$. *Claim*: $\exists d s.t. [\nabla f]^T d < 0$. Does such d exist? yes, take $d = -\nabla f(x)$. There's at least one.

For any such d, $\phi(x + \alpha d) = \phi(x) + [(\nabla f(x))^T d]\alpha + \mathcal{O}(\alpha^2)$ $\phi(x)$ is $f(x)$, and $(\nabla f(x))^T d < 0$

When $\alpha \ll$, positive, $\mathcal{O}(\alpha^2)$ is negligible incomparison to $[(\nabla f(x))^T d]\alpha$. So $\phi(x + \alpha d) < f(x) \forall$ small positive α. $\Rightarrow\Leftarrow$ because we started off with a local minimum! $\Rightarrow \nabla f(x) = 0$.

Summary: A necessary condition for x to be a minimizer is that $\nabla f(x) = 0$. (not sufficient): called the 1st order necessary condition. This can be solved by doing newton's method on $F = \nabla f(x)$.

Check the second derivative, if concave (second derivative positive), then we have a local minima.

Look at 3-term taylor series. $\phi''(\alpha) = d^T \nabla^2 f(x + \alpha d) d$, where $\nabla^2 f(x + \alpha d)$ is a matrix with (i, j) entry given by $H = \frac{\partial^2 f}{\partial x_i \partial x_j}(x + \alpha d)$. This is the **Hessian** matrix. So $\phi''(0) = d^T \nabla^2 f(x) d = d^T H d$ So:

$$\begin{aligned} \phi(\alpha) = f(x + \alpha d) &= \phi(0) + \phi'(0)\alpha + \frac{1}{2}\phi''(0)\alpha^2 + \mathcal{O}(\alpha^3) \\ &= f(x) + [\nabla f(x)]^T d]\alpha + \frac{1}{2} d^T H d \alpha^2 + \mathcal{O}(\alpha^3)(\text{ because } x \text{ is a local minimum, } \nabla f(x)]^T d == 0, \text{ and } d^T H \end{aligned}$$

(if $d^T H d < 0$, then $f(x + \alpha d) < f(x) \forall \alpha \ll$ ($\Rightarrow\Leftarrow$ just like the other reasoning)).

So another necessary conditions is $H = \nabla^2 f(x)$ has to be positive semidefinite, this is calloed the 2nd order necessary condition.

Note that these together are not sufficient.

Example: $n = 2$, $f(x) = \frac{1}{2}x_1^2 + x_1 + x_2^3$ then $\nabla f = \begin{matrix} x_1 + 1 & 3x_2^2 \end{matrix} = \begin{matrix} 0 & 0 \end{matrix}$ set $x_1 = -1, x_2 = 0$. So $\nabla^2 f = \begin{matrix} 1 & 0 \\ 0 & 6x_2 \end{matrix} = \begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix}$ at $x = (-1, 0)$

Condition for a matrix to be positive definite is that all its eigenvalues are strictly positive. Semi-definite is all its eigenvalues are $le 0$. So this is H that satisfies the condition and we have two necessary conditions satisfied at $x = (-1, 0)$. But it's not because we can find a point $\tilde{x} = (-1, x_2)$ $f(\tilde{x}) = \frac{1}{2} + x_2^3 < f(x) = -\frac{1}{2} \forall x_2 le 0$.

Sufficient conditions:

1. $\nabla f(x) = 0$
2. $\nabla^2 f(x) = H$ is positive-definite. (i.e. in 2-D function is concave at this point)

If we have some x not a minimizer, algorithm will chose a d s.t. $f(x + \alpha d) < f(x)$. One condition for $d = -\nabla f(x)$.

Another approach: try to find a root of the equation $F(x) = \nabla f(x) = 0$.

Recall *Newton's method*: In 1-D, $x_{n+1} = x_n + \frac{-f(x_n)}{f'(x_n)}$. In N-D, it's $J_F(x_n)d = -F(x_n)$ (not the same d)

5 Class 17,18 November 15th 2011

Missed for grace hopper

Optimization →

given $f: \mathbb{R}^n \to \mathbb{R}$. find x s.t. $f(x) \leq f(\hat{x})$ $\forall \hat{x}$ near x.

review from last class

necessary conditions for x to be a minimizer

→ first order condition

$$\nabla f(x) = 0$$

→ second order condition

$\nabla^2 f(x)$ is positive semi-definite

↑

hessian

a sufficient condition (guarantee x is a local min)

$$\nabla f(x) = 0$$

$\nabla^2 f(x)$ positive definite.

in fact, this $\Rightarrow$ $f(x) < f(\hat{x})$ $\forall \hat{x}$ near x

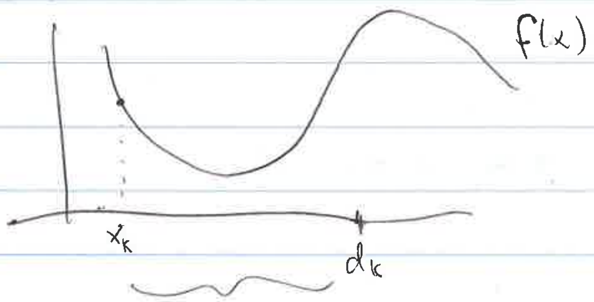
Algorithms for finding minimizers:

- generate $x_1, x_2, \ldots$ (starting w/ x_0)
- given x_k, find d_k (descent direction) s.t. $(\nabla f)(x_k)^T d_k < 0$
- update $x_{k+1} = x_k + \alpha_k d_k$ for some $\alpha_k > 0$

so, $\exists$ 2 issues we need to work on:

1. choice of direction d_k
2. choice of scalar α_k.

let's consider α_k (interpreted as steplength) for a moment: 1-d example.



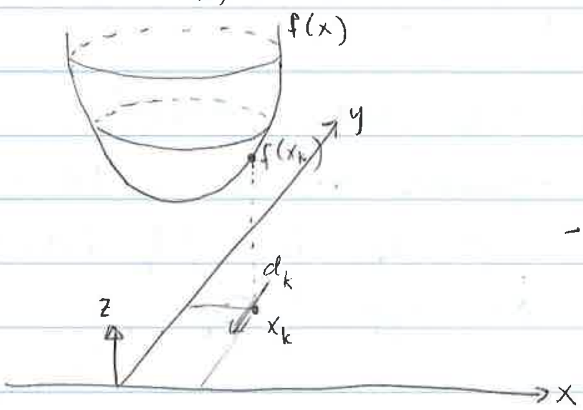
want to find direction to move that will give us a smaller f^n value.

suppose $|d_k| = 1$. also for x_k as shown,

$\nabla f(x_k) = f'(x_k) < 0$

for $d_k = 1$, $\underset{\uparrow \atop f'(x_k)}{f'(x_k)^T} \ \underset{\uparrow \atop 1}{d_k} < 0$

2D example

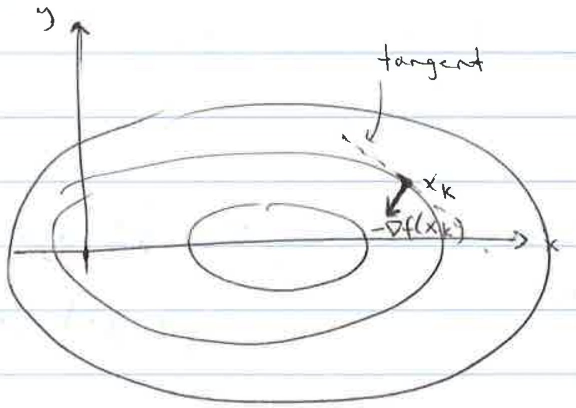


know

$\nabla f(x_k)^T \cdot (-\nabla f(x_k)) < 0$

$-\nabla f(x_k)$ is a descent direction.

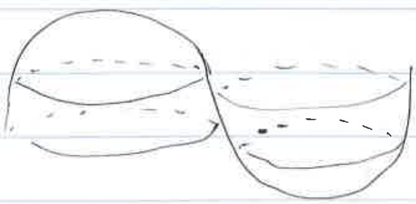
Contours



Contour Lines
$\Rightarrow f(\vec{x}) = \text{constant}.$

on the contour plot : $\nabla f(x_k)$ $\perp$ tangent Line to the contour through x_k.

but, suppose it later turned out the ~~bowl~~ bowl in the graph turned over its surface, eg.



then the steplength α_k could make a big difference.

So far, our only tool approach) ~~is~~ is to choose the negative gradient & go from there. but it's not the best way.

1e, Concerning item #1, one possible descent direction is $d_k = -\nabla f(x_k)$

For another approach, consider the taylor series for f.

$$f(x_k + d) = f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 f(x_k) d$$

$$+ O(\|d\|^3) \text{ for small } d.$$

call this region of interest $\Phi(d)$ s.t. $\Phi : \mathbb{R}^n \to \mathbb{R}$.

$$\nabla \Phi = (\nabla f)(x_k) + \underbrace{\left[\nabla^2 f(x_k)\right]}_{H_k} d \to \text{set} = 0$$

want to relate this back to the first order condition that $\nabla f(x) = 0$.

$$\Rightarrow d = -H_k^{-1} (\nabla f) x_k$$

(but we never multiply by inverse - instead we solve $H_k d = -\nabla f(x_k)$ for d)

$\nabla^2_d \Phi = H_k$ (b/c $\nabla \Phi$ ~~depend~~ on is linear f^n of d)

we know that d minimizes Φ if H_k is positive definite.

This strategy is known as Newton's method.

So - do we choose newton's or steepest descent method?

newton	steepest descent
expensive	cheap
could lead you astray if H_k is not positive definite	slow
much faster	safe (guaranteed to work)

Strategy: start w/ something safe & then switch in a specific region of interest.

Convergence note: x_k converges to x with rate r

if $$\lim_{k \to \infty} \frac{\| x - x_{k+1} \|}{\| x - x_k \|^r} = c$$ constant, independent of k.

ie, Concerning item #1, one possible descent direction is $d_k = -\nabla f(x_k)$

For another approach, consider the taylor series for f.

$$f(x_k + d) = f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 f(x_k) d + O(\|d\|^3) \text{ for small } d.$$

call this region of interest $\Phi(d)$ s.t. $\Phi : \mathbb{R}^n \to \mathbb{R}$.

$$\nabla \Phi = (\nabla f)(x_k) + \underbrace{\left[\nabla^2 f(x_k)\right]}_{H_k} d \to \text{set} = 0$$

want to relate this back to the first order condition that $\nabla f(x) = 0$.

$$\Rightarrow d = -H_k^{-1} (\nabla f) x_k$$

(but we never multiply by inverse - instead we solve $H_k d = -\nabla f(x_k)$ for d)

$\nabla^2_d \Phi = H_k$ (b/c $\nabla \Phi$ ~~depend~~ on is linear f^n of d)

We know that d minimizes Φ if H_k is positive definite.

This strategy is known as Newton's method.

So - do we choose Newton's or steepest descent method?

Newton	steepest descent
expensive	Cheap
Could lead you astray if H_k is not positive definite	slow
much faster	safe (guaranteed to work)

Strategy: start w/ something safe & then switch in a specific region of interest.

Convergence note: x_k converges to x with rate r

if $$\lim_{k \to \infty} \frac{\| x - x_{k+1} \|}{\| x - x_k \|^r} = C$$ constant, independent of k.

∃ 3 types of rates we are concerned about:

$r=1$ linear convergence, in this case we need $c<1$ (★ true for steepest descent).

$r=2$ quadratic rate of convergence. don't § need $c<1$, but we need denominator to not affect c. (★ this is true for newton's method)

Suppose $c = 1/2$ for steepest descent.
Suppose $c = 2$ for newton.

$c = 2$ / Newton	
0	$1/4$
1	$2 \cdot 1/16 = 1/8$
2	$2 \cdot 1/64 = 1/32$
3	$2 \cdot 1/32^2 = 2 \cdot 1/1024 = 1/512$
4	$2 \cdot 1/2^{..} = 1/2^{4}$

← compare w/ steepest descent much slower convergence

One more thing: it's possible to have $1 < r < 2$ - this is superlinear convergence. obtained by combining newton & other methods - "quasi" newton methods. saves some overhead.

6 Class 19 November 15th 2011

hw 2b) requires $(A \otimes B)(C \otimes D) = AC \otimes BD$

should prove this identity on the hw

back to optimization...

algorithms for finding minima of

$f(x)$ $f: \mathbb{R}^n \to \mathbb{R}$

structure of most algorithms goes like this:

start w/ some x_0

for $k = 0, 1, 2, \ldots$ until convergence

- compute $g_k = \nabla f(\vec{x}_k)$ ← deriv. of f wrt each component of f. ie, x is a vector

newton's method
$\left\{ \begin{array}{l} \text{maybe compute} \\ \text{the hesian (newton's method)} \\ H_k = \nabla^2 f(\vec{x}_k) \text{ \& solve} \\ H_k d_k = -g_k \text{ for } d_k \end{array} \right.$

OR

steepest descent $\{ \; d_k = -g_k$

$x_{k+1} = x_k + \alpha_k d_k$, for some α_k T.B.D.

how to determine "convergence"?

also, note that w/ hesian, (newton's method) need to check if $g_k^T d_k < 0$ $\Rightarrow$ we indeed have a descent direction. (it's guaranteed / by def$^{\underline{n}}$ for steepest descent).

If we know we're working w/ a descent direction, then for a reasonable choice of alpha we're going to go down.

Convergence test: $\|g_k\|$ is "small enough".

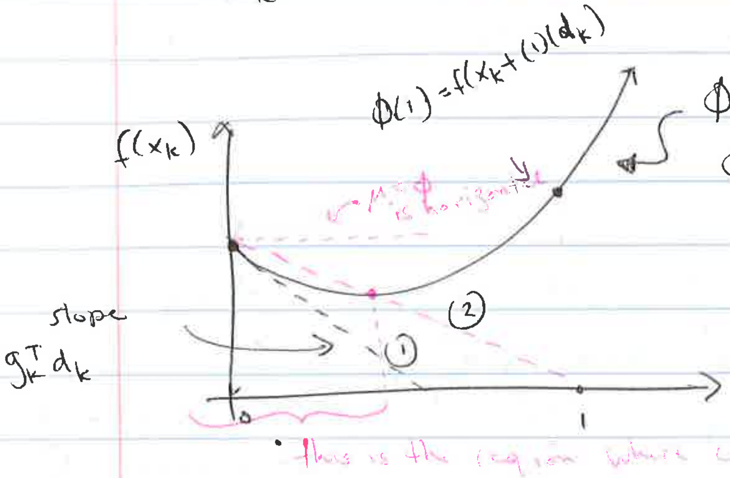
small enough might be abs. or relative error.

ie. $\frac{\|g_k\|}{\|f(x_k)\|} \leq \tau$, tolerance, which depends on the problem / context.

Cost - assume we have an analytic formula that includes expressions for ∇f & $\nabla^2 f$.

there are n iterations for steepest descent, n^2 for newton (? i think this is what he said...).

So, how to choose α_k? (assume we have a d_k already - by whatever method).



$\phi(\alpha) = f(x_k + \alpha d_k)$

$\phi(0) = f(x_k)$

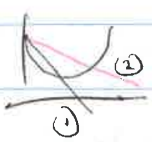
$\boxed{\text{Line search Method}}$

in this example $\phi(1) > f(x_k)$, so we want to pick another version of α. $\boxed{\text{Strategy?}}$

Taylor Series:

$$\phi(\alpha) = f(x_k + \alpha d_k) = \underbrace{f(x_k) + \alpha g_k^T d_k}_{\text{Line through } (0,\, f(x_k)) \text{ w/ slope } g_k^T d_k\ (<0)} + \underbrace{O(\alpha^2)}_{\text{ignore}}$$

since we know $d_k < 0$, since Line (1) is below function f & tangent to f @ 0, then if we create a new line w/ a larger (but still -ive) slope, then it is guaranteed to be above $\hat{f}$ f for at least some period of time. (even if you had something like



where f goes below line ①).

the pink dashed line is a line through $(0, f(x_k))$ w/ slope $\mu g_k^T d_k$, for $\mu \in (0,1)$. eg. $\mu = 1/2$.

$\mu = 0 \Rightarrow$ horizontal
$\mu = 1 \Rightarrow$ slope is $g_k^T d_k$, same as original line.

Req'mt on α_k:

$\mu = 1/2$ convention. people research this. (so, fun!)

eq ①
$$\underbrace{f(x_k + \alpha_k d_k)}_{\text{value of } \hat{f} \text{ line at } \alpha_k} \le \underbrace{f(x_k) + \mu \alpha_k g_k^T d_k}_{\text{value of pink line at } \alpha_k}$$

"simple-minded approach":
- try $\alpha_k = 1$. if req't is satisfied, then accept.
- otherwise, try $\alpha_k \leftarrow \alpha_k / 2$ (reduce α_k)
- iterate...

✱ note: ∃ still more book keeping to do, to ensure we have a global min.

For n-dimensions, think of the 2D graph as a cross-section in the direction of $\vec{d}_k$ (vector).

Typical convergence th'm

For a descent method applied to a function f that is bounded below, for which gradient g is Lipschitz continuous, then $\|g_k\| \to 0$

$\searrow$ Lipschitz continuous

$\equiv \|g(x) - g(y)\| \leq L\|x - y\|$

for some $L > 0$

$\forall\, x, y$ in the region of interest.

$\exists$ better ways to choose α? viz. simple minded approach discussed earlier - want a diff way that could be smarter ... ie, if we can use our knowledge of f to reduce stupid guesses, we are happier.

in particular, we'll have to compute eq^{n} (1) less often.

Cost of computing ① ? Well,

$f(x_k)$ - free

$\mu \alpha_k g_k^T d_k$ - cheap

but $f(x_k + \alpha_k d_k)$ (LHS) might be arbitrarily complex f^n f.

So, an alternative approach to meet the req't of eq^n ① :

start w/ $\alpha_{k_1} = 1$

α_{k_2} = something like $1/2 \alpha_{k_1}$

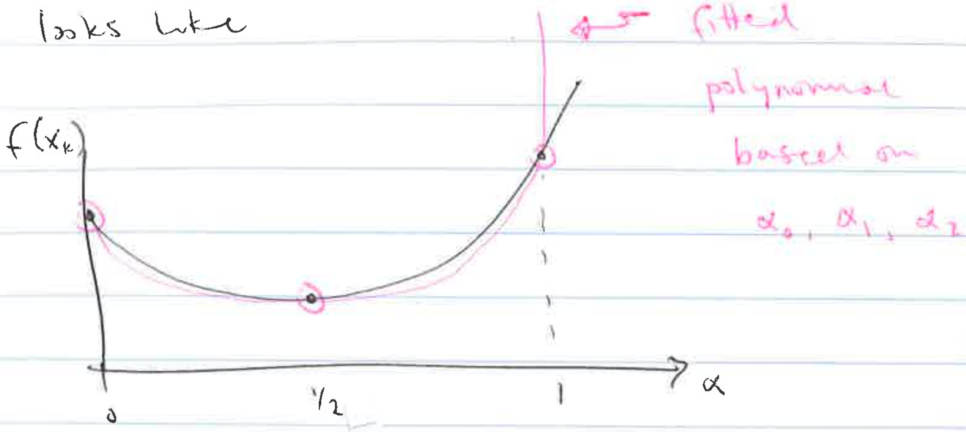
Now, we have candidate values:

$\alpha_k = 0$ $f(x_k)$ $\alpha_k = 1$ $f(x_k + d_k)$

$\alpha = 1/2$ $f(x_k + \frac{1}{2} d_k)$

let $p(\alpha)$ = quadratic polynomial that interpolates 3 points. Then, find α that minimizes $p(\alpha)$. Call this α_{k_3}.

looks like



the fitted polynomial makes use of values we've already computed.

6.1 Optimization

$f : \mathbf{R}^n \to \mathbf{R}$ Iteration $x_{k+1} \leftarrow x_k + \alpha_k d_k$, where d_k is the descent direction $d_k^T g_k < 0$, and α_k is the step length obtained from line search.

From last time: Choose α_k s.t.

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \mu \alpha_k d_k^T g_k$$

where μ is a fixed parameter. Called the *Armijo condition.* Refer to the graph on the notebook, but $\phi(\alpha) = f(x_k + \alpha d_k)$, $\phi(0) = f(x_k)$, $\phi'(0) = \nabla f(x_k)^T d_k = d_k^T g_k$, and $\phi_\mu(\alpha) = f(x_k) + \mu \alpha d_k^T g_k$.

Another way to do line search: try to find α near the value that minimizes $\phi(\alpha)$. Where $\phi'(\alpha) = [\nabla f(x_k + \alpha d_k)]^T d_k$, such minimizer of ϕ (if it exists) satisfies $\phi'(\alpha) = 0$.

Strategy: Choose α_ks.t. $\phi'(\alpha)$ is not too large. That is, require

$$|\phi'(\alpha)| \leq \eta |\phi'(0)|$$

for some fixed constant η (example $\eta = .9$).

Guess α by evaluating $|\nabla f(x_k + \alpha d_k)]^T d_k|$ compare it to $|g_k^T d_k|$. Called the *Wolfe Condition.* This way is more expenisve, because we need to compute the ∇f, which involves n computations, as opposed to computing f is just single real value. But this is prefered because: we are requiring the derivative to be smaller, it force us to chose a bigger step sizes and move closer to the minimum sooner. (Because say we have a candidate a very close to 0, the slope of $\phi'(a)$ will be very close to $\phi(0)$, so it will prevent us from chosing α that's too close to 0.

So far we've seen

- steepest descent $d_k = -g_k$ (slow but cheaper)
- Newton's method d_k solves $H_k d_k = -g_k$, called the Hessian. $\nabla^2 f(x_k)$ is the descent direction iff H_k is positive-definite. ($d_k^T g_k < 0 \to -g_k^T H_k g_k$ is guaranteed to be < 0 if H_K positive-definite, but may happen by luck so we want the condition example: $g_k = [1, 0], H_k = (\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix})$).

  Evaluating the hessian and solving system is very expensive.

6.2 Quasi-Newton Methosd

This method will construct B_K, an approximation to H_k somehow. Algorithm:

Start with an arbitrary x_0, B_0 ($= I$, for example), $g_0 = \nabla f(x_0)$
for $k = 0, 1, 2, \ldots$ until convergence **do**
  solve $B_k d_k = -g_k$
  update $x_{k+1} = x_k + \alpha_k d_k$
  compute $g_{k+1} = \nabla f(x_{k+1})$
  update $B_{k+1} = B_k -$ (some update)
end for

Q: How to choose B_{k+1}??

One approach: Consider $H(x_k)$ when f is a quadratic function.

$$f(x) = \frac{1}{2} x^T Q x + b^T x + c$$

, a typical quadratic function. $\nabla f(x) = Qx + b$ (exercise) $\nabla f(x+d) = Qx + Qd + b$, and $\nabla f(x+d) - \nabla f(x) = Qd$. ALso, $\nabla^2 f = Q$. This implies that

$$Hd = \nabla^2 f d = Qd = g(x+d) - g(x)$$

We will try to mimic this relationship for general f.

Try to eforce this conversation as follows: Choose B_{k+1} to satisfy $B_{k+1}d_k = g_{k+1} - g_k$. But this is not enough to get a good estimate so in addition, $B_{k+1}v = B_k v$ $\forall v$ orthogonal to d_k (because the first condition is just 1-D, this make sure that we don't do anything to whatever that's not in the same direction).

Consider

$$B_{k+1} = B_k - \frac{(B_k d_k - (g_{k+1} - g_k))d_k^T}{d_k^T d_k}$$

Check:

$$\begin{aligned} B_{k+1}d_k &= B_k d_k - \frac{(B_k d_k - (g_{k+1} - g_k))d_k^T d_k}{d_k^T d_k} \\ &= B_k d_k - (B_k d_k - (g_{k+1} - g_k)) \\ &= g_{k+1} - g_k \end{aligned}$$

Also

$$\begin{aligned} B_{k+1}v &= B_k v - \frac{(B_k d_k - (g_{k+1} - g_k))d_k^T v}{d_k^T v} \\ &= B_k v - 0 \text{ for } v \perp d_k \end{aligned}$$

This is called the *Broyden's method*, a rank-1 update. Way cheaper than evaluating the hessian.

If H is a hessian, it would be a symmetrix matrix (it's a second derivative $\frac{\partial^2 f}{\partial x_i \partial x_j}$). But B_k by Broyden's method is not symmetric.

So *Symmetric variant:* Let $y_k = g_{k+1} - g_k$,

$$B_{k+1} = B_K + \frac{(y_k - B_k d_k)(y_k - B_k d_k)^T}{(y_k - B_k d_k)^T d_k}$$

7 Class 20 November 17th 2011

Review of the HW3 #2: $\hat{A}_c = A_c + E$, we want A. THere is a linear relation between A and A_c. Unravel $A, A_c, \hat{A}_c$ column wise to get $a, a_c, \hat{a}_c$. The naive computation was to solve $Ka = \hat{a}_c$ but this runs out of memory, or takes way too long.

K has the form $B \otimes C$. Question was how do you take advantage of this and solve $Ka = \hat{a}_c$. Supposed we wanted to solve $(B \otimes C)x = y$ $B \otimes C = \begin{pmatrix} b_{11}C & b_{12}C & \cdots & b_{1n}C \\ \vdots & & \cdots & \vdots \\ b_{11}C & b_{12}C & \cdots & b_{1n}C \end{pmatrix}$ Change x s.t. it's m by n. THen, the first block is

$$C(b_{11}x_1 + b_{12}x_2 + \cdots + b_{1n}x_m) = C[x_1, x_2, \cdots, x_m] \begin{pmatrix} b_{11} \\ b_{12} \\ \vdots \\ b_{1n} \end{pmatrix} = \text{transpose of 1st row of } B \text{ is 1st col of } B^T$$

So

$$CXB^T = Y$$

where X and Y are reshaped versions of x and y in blocks. So the solution is $X = C^{-1}YB^{-T}$. Let $C = U_c\Sigma_cV_c^T$ and $B = U_b\Sigma_bV_b^T$, then X is $V_c\Sigma_c^{-1}U_c^TY(U_b\Sigma_b^{-1}V_b^T)$

7.1 Quasi-Newton Methods

From before:

Start with x_0, B_0 ($= I$, for example), $g_0 = \nabla f(x_0)$
for $k = 0, 1, 2, \ldots$ until convergence **do**
  solve $B_kd_k = -g_k$
  update $x_{k+1} = x_k + \alpha_kd_k$
  compute $g_{k+1} = \nabla f(x_{k+1})$
  update $B_{k+1} = B_k-$ (some update)
end for

We looked at *Broyden's method*

$$B_{k+1} = B_k - \frac{(B_kd_k - (g_{k+1} - g_k))d_k^T}{d_k^Td_k}$$

That satisfies both conditions, but B_k may not be symmetric (and should be because H_k is).

Symmetric variant: Let $y_k = g_{k+1} - g_k$,

$$B_{k+1} = B_K + \frac{(y_k - B_kd_k)(y_k - B_kd_k)^T}{(y_k - B_kd_k)^Td_k}$$

We can't impose the second condition that $B_kv = 0$ for $v \perp d_k$. Also B_k may not be positive definite.

Further refinment:

$$B_{k+1} = B_k - \frac{(B_kd_k)(B_kd_k)^T}{d_k^TB_kd_k} + \frac{y_ky_k^T}{y_k^Td_k}$$

It's easy to verify that $B_{k+1} d_k = g_{k+1} - g_k$

Theorem If $y^T d_k > 0$ then B_{k+1} is positive definite.

$y^T d_k > 0$ means $g_{k+1}^T d_k > g_k^T d_k$. d_k taht we're working with is a descent direction, so both sides of the inequality is negative, so it means the next descent direction is not as negative as the one before. i.e.

$$-g_k^T d_k > -g_{k+1}^T d_k$$

Returning to Wolfe condition for line search, it says $|g_{k+1}^T d_k| < \eta |g_k^T d_k|$ for some $\eta < 0$.

This is equivalent to

$$-g_{k+1}^T d_k \leq |g_{k+1}^T d_k| < \frac{1}{\eta} |g_{k+1}^T d_k| \leq -g_k^T d_k$$

(given $\eta < 1$). The end equality is the same condition about the theorem. i.e. if we impose the Wolfe condition, B_{k+1} is positive definite. This refined version where it guarantees positive definite ness is called the $BFGS$ **method**.

7.2 Constrained Optimization

Now we want to $\min f(x)$ $f \in \mathbf{R}^n$ subject to $g(x) \leq 0$, where $g : \mathbf{R}^n \mapsto \mathbf{R}^m$.

ex. $n = 2$. $x_2 = x_1 + 1$, $g(x) = x_2 - x_1 - 1 \geq 0$, then the possible region is the parts above the line.

Find x in constraint set that minimizes f.

8 Class 22 (skipped one) Nov 29th

HW3: first part, to show the error is proportional to $\kappa(Q)$, taylor series approximation is involved. Number of steps is approximately $\kappa(Q)$ based on the taylor approximation, assuming that $\kappa(Q)$ is big. Part b of problem 2, enforce certain equalities, gradually tweak so taht the equality is true. Do line search in several different ways and he talked about a way in class to do tweaking, do that for number 2. For number iii, use matlab minimization function. For i, you might have to do global search, but tweaking is what he has in mind..

8.1 Constrained Minimization

Minimize for $x \in \mathbf{R}$, $f(x)$ subject to $Ax = b$. A is a full rank m by n wide matrix ($m < n$).

- Method 1: start with a feasible $\bar{x}$ (satisfies $A\bar{x} = b$), let Z =matrix whose columns span the null space of A, $Z \in \mathbf{R}^{n\times(n-m)}$. (SVD's last $m - n$ columns of V). We can solve this problem by solving $\min v \in \mathbf{R}^{n-m} \hat{f}(v)$, $\hat{f}(v) = f(\bar{x} + Zv)$ (compute the gradient of $\nabla_v \hat{f} = Z^T(\nabla_x f(\bar{x} + Zv))$, $grad_v^2 \hat{f} = H = Z^T(\nabla_x^2 f(\bar{x} + Zv))Z$ then use steepest descent or newton using H)
- Method 2: Using the first order conditions for minimizing $\hat{f}$, we are led to the equation of the form $\nabla f(x) + A^T\lambda = 0$ and $Ax = b$. We want to find x, λ that satisfies these equations. (the first equation says that the gradient is in the range space of A) We don't need Z for this method: if n is large, computing Z is expensive.

  This is a nonlinear system of equations, to do it we'll discuss it in class.

For method 1, how do we get $\bar{x}$, a feasible point? Suppose we obtained Z by QR factorization. $A^T = QR = [Q_1Q_2][R_1; 0] = Q_1R_1$, where $Z = Q_2$ Q_1 spans the range space of A and Q_2 is the orthogonal complement of Q_1 so it spans the null space of A. And $A = R_1^T Q_1^T$. We want $A\bar{x} = b$:

$$A\bar{x} = b$$
$$R_1^T Q_1^T \bar{x} = b$$
$$Q_1^T \bar{x} = R_1^{-T} b \text{ let } R_1^{-T} b = \hat{b}$$
$$\text{Define } \hat{x} = Q_1 \hat{b}$$
$$Q_1^T \bar{x} = Q_1^T Q_1 \hat{b}$$
$$Q_1^T \bar{x} = Q_1^T Q_1 R_1^{-T} b$$
$$R_1^T Q_1^T \bar{x} = R_1^T R_1^{-T} b$$
$$A\bar{x} = b$$

Now we have $\bar{x}$.

8.2 Nonlinear constraint problem

minimize $f(x)$ for $x \in \mathbf{R}^n$, subject to $g(x) \geq 0$, where $g : \mathbf{R}^n \mapsto \mathbf{R}^m$. Example: $g(x) = Ax - b$, or $g_1(x), \ldots, g_m(x)$ all ≥ 0. Harder problem. Highlevel statemetn: Minimize the function inside, move closer to the boundary and see if that changes things.

Strategy:

- Define $\phi(x)$ s.t. $\phi(x) \to \infty$ as $g(x) \to 0$.
  - *Example*: $\phi(x) - \sum_1^m \log q_i(x)$ means log is well defined, if all g_is approach 0, $\sum_1^m \log q_i(x) \to -\infty$, so $\phi(x) \to \infty$.
  - *Example*: $\phi(x) = \sum_{i=1}^m \frac{1}{g_i(x)^2}$
- Introduce a scalar parameter $\mu > 0$ and consider the function

  $$\hat{f}_\mu(x) = f(x) + \mu\phi(x)$$

  (function of n+1 param, but think of μ as fixed here)
- Consider the unconstraint problem $\min_x \hat{f}_\mu(x)$. As x approaches the boundary of the constraint set i.e. $g(x) \equiv 0$, $\mu\phi(x) \to \infty$, so the solution/minimizer to this problem will be in the interior of our constraint, that is $g(x) > 0$. This minimizer x depends on μ. i.e. $x = \mu(x)$.
- next step, is to reduce μ and do it again.

Claim: as $\mu \to 0$, $x(\mu) \to x^*$, the solution to the constraint problem.

Algorithm - the Barrier Method: For a sequence $\mu_1 > \mu_2 > \ldots$, find $x(\mu_i)$, use $x(\mu_i)$ as the initial value for the problem with parameter $\mu_i + 1$ $(\hat{f}_\mu(x))$. As $\mu \to 0$, the conditioning of the hessian of $\hat{f}_\mu(x)$ grows. Meaning makes it harder for newton's method to solve.

8.3 Nonlinear equations

$F : \mathbf{R}^n \mapsto \mathbf{R}^n$, where $F = \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ f_2(x_1, \ldots, x_n) \\ \vdots f_n(x_1, \ldots, x_n) \end{pmatrix}$ We want to find x s.t. $F(x) = 0 \in \mathbf{R}^n$. Taylor series: $F(x + d) = F(x) + J_F(x)d + \mathcal{O}(||d||^2)$ for d small where $J_F = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$

Newton's method: Given an iterate $x^{(k)}$, compute $d^{(k)}$ update by ignoring the $\mathcal{O}(||d||^2)$ term and setting $F(x^{(k)})d_k = 0$, solve

$$J_F(x^{(k)})d^{(k)} = -F(x^{(k)})$$

then $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$ Need to solve systems of linear equations (LU but ppl don't really use LU anymore)

9 Class 23 December 1st 2011

$F : \mathbf{R}^n \to \mathbf{R}^n$, find x s.t. $F(x) = 0$.

Newton's method:

start with x_0
repeat
solve

$$J_F(x_k)d_k = -F(x_k)$$

set $x_{k+1} = x_k + d_k$
until $||F(x_k)||$ small enough

If $F(x) = G(x) - g$, often the tolerance looks like

$$||F(x_k)|| \leq \tau ||g||$$

Like a relative error, where g is like an input data.

Sometimes there is no g. Then we stop when $||F(x_k)|| \leq \tau$, some tolerance τ.

A few difference from optimization are that:

- J_F may not be symmetric
- Don't have a descent direction

9.1 Convergence

Loosely,

$$||x - x_{k+1}|| = c||x - x_k||^2$$

Whenever x_k is close enough to x and J_F is smooth enough.

Small enough means $J_F(x)$ is nonsingular and Lipschitz continuous

$$||J_F(z_1) - J_F(z_2)||le\gamma||z_1 - z_2||$$

$\forall z_1, z_2$ near x.

"near x" means: $\exists \delta > 0$ s.t. Lipschitz continuity holds $\forall z_{1,2}\,, s.t.||z_i - x|| < \delta$

These are called "standard assumptions". Usually try to get into this ball by doing steepest descent, then do newton because then convergence is faster.

9.2 Inexact Newton's method

Solving for $J_F(x_k)d_k = -F(x_k)$ is the most expensive part of the task. If we're not sure if we're in the ball, it doesn't pay enough to compute for an accurate d_k.

Consider a scenario where we're far from the solution. Suppose instead of d_k, we somehow get $\hat{d}_k$, s.t. the residual

$$|| - F(x_k) - J_F(x_k)\hat{d}_k|| \leq \eta_k ||F(x_k)||$$

i.e. not forcing d_k to be exact.

with this $x_{k+1} = x_k + \hat{d}_k$,

Theorem: under standard assumptions, the error satisfies

$$||x - x_{k+1}|| \leq c_1 ||x - x_k||^2 + c_2 \eta_k ||x - x_k||$$

So if we could choose $\eta_k = ||x - x_k||$, then we can recover quadratic convergence.

This is not possibel because we don't know x.

Note: *Lemma*: for x_k near x,

$$||x - x_k|| \approx c||F(x_K)||$$

Using the lemma, we could choose $\eta_k = \tau||F(x_k)||$ (in the theorem). Then if we can find $\hat{d}_k$ s.t.

$$||-F(x_k) - J_F(x_k)\hat{d}_k|| \leq \tau||F(x_K)||^2$$

we can *recover quadratic convergence.*

How to compute $\hat{d}_k$: Use an interative method, for each k, iterate over j: find $\hat{d}_{k,j} : \hat{d}_{k,1}, \hat{d}_{k,2}, \ldots, \hat{d}_{k,m}$ s.t. the residual $||-F(x_k) - J_F(x_k)\hat{d}_{k,j}||$ decreasing with j, so $\hat{d}_k = \hat{d}_{k,m}$, m is when you satisfy the condition: residual $\leq \tau||F(x_k)|^2$

9.3 Example of a nonlinear equation

Function $u(x, t)$ is the density of cars driving on a street at time t, where x-axis represents the 1 way street. It's modeled by:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = v\frac{\partial^2 u}{\partial x^2}$$

$v > 0$, where $u\frac{\partial u}{\partial x}$ is the transport term, moves things from left to right, and $v\frac{\partial^2 u}{\partial x^2}$ is the diffusion term.

There is no analytic solution because this is non-linear.

$u(x, 0)$ given at time $t = 0$, and $u(0, t), u(1, t)$ given.

We want to discretize in space and time. Then approximate:

$$\frac{\partial u}{\partial x}|_{x=x_i} \sim \frac{u(x_{i+1}, t) - u(x_{i-1}, t)}{2k}$$

$$\frac{\partial^2 u}{\partial x^2}|_{x=x_i} \sim \frac{u(x_{i+1}, t) - 2u(x_i, t) + u(x_{i-1}, t)}{k^2}$$

$$\frac{\partial u}{\partial t}|_{t=t_m} \approx \frac{u(x, t_{m+1}) - u(x, t_m)}{\Delta t}$$

Denote: $u(x_i, t_j) = u_{ij}$. Substitute approximations in PDE at x_i, t_{m+1}. Then we get:

$$\frac{u_{i,m+i} - u_{i,m}}{\Delta t} + u_{i,k+1}\left(\frac{u_{i+1,m+1} - u_{i-1,m+1}}{2k}\right) = v\left(\frac{u_{i+1,m+1} - 2u_{i,m+1} + u_{i-1,m+1}}{k^2}\right)$$

The unknowns are $u_{1,m+1}, u_{2,m+1}, \ldots, u_{n,m+1}$.