4. TRICKS OF THE TRADE

4.1 Structure of an optimization program

* Elements of an optimization problem and "templates".

* Why would I want to do this? It is easier to "debug" and verify and reuse. The number of errors per functional block may be larger; but they are easier to find. Also easier to grade :-)

* Elements: 1) The problem 2) The iteration 3) Stopping criterion 4) Update of parameters

* Good software practices: functional forms and comments, consistency tests

Mihai Anitescu's Notes

4.12 Elements:

1) Problem (e.g. feval_wrap.m)

function [f, g, h] = myproblem (x, param_l

% param_list: wderivs
% problem_version

2) One iteration (e.g. newtonlikeIt)

function [x_out, info] = myiteration
(functionHandle, x_in, param_list

% param_list = iteration_type
% iteration_index

.....

[f, g, H] = functionHandle (x, ...)

$\boxed{\text{info} = \text{grad value?}}$

3) Stopping criterion: $\boxed{\|\nabla f\| \le \varepsilon}$

$\|d^k\| \le \varepsilon$

Tricky. $\longleftarrow$ $\|x^{k+1} - x^k\| \le \varepsilon$

Mihai Anitescu's Notes

4.1.3 4) Update of parameters (e.g. trust region). Some people would write a separate function for it.

5) main

```
while not_stopping (4)
[xout, ...] = myiteration(@myproblem, x_i
:
update_pars (4)
:
```

Statistics

Mihai Anitescu's Notes

4. Special topics.
4.2. Using Sparsity

(*) Most Hessian matrices of interest in applications are not dense.

E.g. Building Control Problem.

Variational discretization of PDEs

(Laplace in 1d : $V_{x+1} - 2V_x + V_{x-1} = 2f$

(*) Therefore, can store them only by

[i, j, s] = find(M) :-)

(*) nnz; i(*) ; j(*) ; s(*)

column index

row index

value

"It can be reduced even further; some of the row and column information is redundant ("compressed row storage"

Mihai Anitescu's Notes

(*) Advantage: If $nnz(A) \sim O(n)$

4.2.2 then $A * x$ takes only $O(n)$ and

A needs only $O(n)$ Storage

(*) Disadvantage: Now all operations

must be aware of sparsity but most

math library support this format

(*) Problem: A may be sparse, but

$A = LL^T \Rightarrow L$ may be dense; so

storage will be a problem

(*) Idea: Use iteration methods such as

CG, which needs only Matrix-Vector

multiplication and terminates early...

Mihai Anitescu's Notes

4. Special topics
4.3. Conjugate gradient

* Idea: for A p.d.; solving $Ax=b$ is equivalent to $\min \frac{1}{2}x^TAx - b^Tx = f(x)$

* Steepest descent is slow; "zig-zag's". I tend to REPEAT search directions. How do I prevent this?

* Idea: Use ORTHOGONAL search directions and minimize error along them. $\langle d^i, d^j \rangle = 0; i$

Then $e^0 = Ax^0 - b = -\sum_{i=1}^{n} \alpha^i d^i$

$\Rightarrow \langle e^{0T}, d^i \rangle = -\alpha^i \langle d^{iT}, d^i \rangle \Rightarrow \alpha^i = -\frac{\langle e^{0T}, d^i \rangle}{\langle d_i^T, d_i \rangle}$

Define $e^{(j)} \equiv e^0 + \sum_{i=1}^{j} \alpha^{(i)} d^i$

$\Rightarrow \boxed{\alpha^{(i)} = -\frac{\langle e^{(j)T}, d^j \rangle}{\langle (d^j)^T, d^j \rangle}} \Leftrightarrow e^{j+1} = e^j + \alpha^j d^j$ and minimize

Mihai Anitescu's Notes

4.3.2 ⊗ Problem I don't know e^0 !! if I use Euclidean $\langle , \rangle$

But I do know $Ae^0 = A(x^0 - x^*) = Ax^0 - \ldots = -r^0$!!

$$\langle e^j, d^i \rangle_A = e^{j^T} A d^j = -r^{0^T} d^j \text{ !!}$$

⊗ Idea: Require A-ORTOGONAL search directions: $\boxed{(d^i)^T A d_j = 0 \quad i \neq j \text{ !!}}$

$$\Rightarrow e^{j+1} = e^j + \alpha^j d^j$$

$$\Leftrightarrow x^{j+1} = x^j + \alpha^j d^j$$

$$\Leftrightarrow \boxed{-r^{j+1} = -r^j + \alpha^j d^j}$$

Redo calculation $\Rightarrow$

$$\boxed{\alpha^j = \frac{+(r^j)^T d^j}{(d^j)^T A (d^j)}}$$

Note: $\min_\alpha \frac{1}{2}(x^j + \alpha d^j) A (x^j + \alpha d^j) - b^T(x^j + \alpha \ldots$

Mihai Anitescu's Notes

4.3.3 Optimality conditions:

$$d^{j\,T} A(x^j + \alpha^j d) - d^{j\,T} b = 0 \iff$$

$$\alpha^j d^{j\,T} A d^j - (d^j)^T r^j = 0$$

$$\Rightarrow \boxed{\alpha^j = \frac{(d^j)^T r^j}{d^{j\,T} A d^j}}$$

* Observation: Each CG step reduces the objective function. Terminates in n iterations.

* Nice, but; how do I choose conjugate directions?

Idea: start with $u^1, u^2, \ldots u^n$ linearly independent. Choose d^i by Gram-Schmidt with $\langle\ \rangle_A$ inner product.

Mihai Anitescu's Notes

4.3.4. $d^i = u^i + \sum_{k=0}^{i-1} \beta_{ik} d^k$

$\Leftarrow \cdot d^{j^T} A ; j<i$

$$\Rightarrow d^{i^T} A d^j = d^{j^T} A u^i + \beta_{ij} d^{j^T} A d^j = 0$$

$$\Rightarrow \beta_{ij} = -\frac{d^{j^T} A u^i}{d^{j^T} A d^j} \qquad j = 0, \dots i-1$$

$\boxed{\text{But ... need to store ALL } d \dots \Rightarrow O(n^3) \text{ Effort increases at each iteration.}}$

Observations:

$$e^0 + \sum_{i=0}^{n-1} \alpha^i d^i = 0 \Rightarrow e^j + \sum_{i=j}^{n-1} \alpha^i d^i = 0$$

$$\Rightarrow \underbrace{d_\ell^T A e^j}_{= -r^j} + \sum_{i=j+1}^{n} \underbrace{d_i^{T^T} A d^\ell}_{=0} \alpha^i = 0 \qquad (\ell < j)$$

$$(\alpha) \Rightarrow \boxed{d^{\ell^T} r^j = 0} ; \ \ell < j ; \longrightarrow (\beta)$$

$$(\beta) \ \cancel{\boxed{d^{j^T} r^j = u^{j^T} A^j}}$$

Mihai Anitescu's Notes

4.3.5

(*) Return now to Gram-Schmidt.

$$r^{j^T} d^i = r^{j^T} u^i + \sum_{k=0}^{i-1} \beta_{ik}\, r^{j^T} d^k$$

If $i < j \Rightarrow$ by (*) we have

$$\boxed{r^{j^T} u^i = 0} \quad (\beta) \quad \text{for } i < j$$

$$\boxed{r^{i^T} u^i = r^{i^T} d^i}$$

(*) Recursion

$$x^{k+1} = x^k + \alpha^k d^k \quad \Rightarrow (r = b - Ax)$$

$$-r^{k+1} = -r^k + \alpha^k A d^k$$

$$\Rightarrow \boxed{r^{k+1} = r^k - \alpha^k A d^k} \quad (\alpha).$$

Mihai Anitescu's Notes

4.3.6 Recap

If solving $Ax=b$ (or, $\min \frac{1}{2} x^T A x - b^T x$)

by $x^{k+1} = x^k + \alpha^k d^k$ where $\boxed{d^i A d^j = 0}$; $i \neq j$

AND obtained by GS from u^i:

$$\boxed{d^i = u^i + \sum_{k=0}^{i-1} \beta_{ik} d^k} \; ; \; \boxed{\beta_{ij} = - \frac{d^{j^T} A u}{d^{j^T} A}}$$

by $\boxed{\text{exact minimization}}$: $\boxed{\alpha^j = \frac{(d^j)^T r^j}{d^{j^T} A}}$

$$\boxed{r^j = b - Ax^j}$$

Then:

$$\boxed{\begin{aligned} & d^{l^T} r^j = 0 \; ; \; l < j \\ & d^{j^T} r^j = d^{j^T} r^j \\ & r^{j+1} = r^j - \alpha^j A d^j \; ; \; j = 0, \ldots \end{aligned}}$$

Mihai Anitescu's Notes

4.3.7 What if I choose $u^j = r^j$?

Then $\boxed{r^{i\,T} r^j = 0}$ $i \neq j$.

$$(d^{i\,T} A d^j)\beta_{ij} = \cdots - d_j^T A u^i \qquad j = 0, 1, \ldots i-1$$

$$= -\frac{1}{\alpha^j}\left(r^j - r^{j+1}\right)^T r^i \quad \begin{cases} = \frac{1}{\alpha^j}(r^i)^T(r^i) & j = i- \\ 0 & \text{otherwise} \end{cases}$$

$$\cancel{= \left\{ -\frac{1}{\alpha^j}(r^j - \right.}$$

$\Rightarrow$ $\boxed{\text{Only 1 } \beta \text{ is active!!}}$

$$\Rightarrow \boxed{d^i = r^i + \underbrace{\frac{1}{\alpha^{i-1}} \cdot \frac{(r^i)^T(r^i)}{(d^{i-1})^T A d^{i-1}}}_{\text{"}\beta^i\text{"}} \cdot d^{(i-1)}}$$

$$= \boxed{\frac{1}{(r^{i-1})^T(r^{i-1})}(r^i)^T(r^i)\, d^{i-1} + r^i}$$

Mihai Anitescu's Notes

43.8 $\boxed{G.G}$ $r^0 = b^0 - Ax^0 = d^0$

"Line search" $\alpha^i = \dfrac{r^{iT} r^i}{d^{iT} A d^i}$; $x^{i+1} = x^i + \alpha^i d^i$

$$r^{i+1} = r^i - \alpha^i A d^i$$

"Search direction update"

$$\beta^{i+1} = \frac{(r^{i+1})^T r^{i+1}}{r^{iT} r^i}$$

$\Rightarrow$ Only 1 Matrix-vec multiplicat

$$d^{i+1} = r^{i+1} + \beta^{i+1} d^i$$

Variant (Nocedal and Wright)

$$r^0 = Ax^0 - b = -d^0$$

"Line Search": $\alpha^i = \dfrac{r^{iT} r^i}{d_i^T A d_i}$; $x^{i+1} = x^i + \alpha^i d^i$

$$r^{i+1} = r^i + \alpha^i A d^i$$

"Search direction Update"

$$\beta^{i+1} = \frac{(r^{i+1})^T r^{i+1}}{r^{iT} r^i} ;$$

$$d^{i+1} = -r^{i+1} + \beta^{i+1} d^i$$

Mihai Anitescu's Notes

4. Special topics

4.4 Early termination of CG

<u>Claim.</u> Solve $\min \frac{1}{2} x^T A x - b^T x$ by CG

s.t $d^{i\,T} A d^j = 0 \; ; \; i \neq j \leq k$

with $d^0 = r^0 = b$. Then $\|b\|^2$

$x^0 = 0$ $\qquad$ $x^{k\,T} b > 0$

<u>Proof</u>

$$\frac{1}{2} x^T A x = \frac{1}{2} \sum (\alpha^i)^2 d^{i\,T} A d^i =$$

$$\frac{1}{2} \sum (\alpha^i) \cdot (r^{i\,T} r^i) = \frac{1}{2} \sum \frac{(r^{i\,T} r^i)^2}{d^{i\,T} A d^i}$$

$$> 0$$

$$\frac{1}{2} x^T A x - b^T x \leq \phi(0) = 0$$

$$\Rightarrow b^T x > 0$$

QED

Mihai Anitescu's Notes

4.4.2 When applied to optimization problem; produces descent direction.

$$b = -\nabla f \Rightarrow -\nabla f^T p^k > 0.$$

with Hessian (or approx).

Mihai Anitescu's Notes

4.~~8~~ Special topics
4.5 Inexact Newton Methods

(*) Idea: Framework for solving the subproblem inexactly; i.e.; by iterative methods.

Therefore

$$\boxed{r_k = \nabla^2 f(x^k) p_k + \nabla f(x^k)} \rightarrow \boxed{\text{Inexact NM}}$$

Terminate CG iterations, when

$$\|r_k\| \leq \eta_k \|\nabla f(x^k)\|$$

r^k is called the forcing sequence

~~Theorem 7.1. Local convergence. Suppose $\nabla^2 f(x^k)$ exists and is~~

4.5.2 Assume

1) $\nabla^2 f(x) \in C^2(\mathcal{N}(x^*))$

2) $x^{k+1} = x^k + p^k$

3) $0 < \eta^k \leq \eta < 1 \quad \forall k$

Then $\exists\ \eta \leq \boxed{\hat{\eta}}$ s.t

$$\boxed{\|\nabla^2 f(x^*)(x^{k+1} - x^*)\| \leq \hat{\eta} \cdot \|\nabla^2 f(x^*)(x^k - x^*)\|}$$

(*) Observation

You will at least converge !!

<u>Theorem 7.2</u>: Assume $x^k \to x^*$,

Then $\eta^k \to 0 \Rightarrow$ Convergence is superlinear

$\eta^k = O(\|\nabla f(x^k)\|) \Rightarrow$ convergence is quadratic

Superlinear: e.g. $\eta^k = \min(0.5, \sqrt{\|\nabla f(x^k)\|})$

Mihai Anitescu's Notes

4. Special topics

4.6 Line Search Newton-CG

(A) Idea: Solve

$$B^k p + \nabla f(x^k) = 0 \quad \text{approximately} \quad B^k = \nabla^2 f(x^k)$$

by CG

Define $r^k = B^k p + \nabla f(x^k)$; $\|r^k\| \le \varepsilon^k =$

$$= \min\left(0.5, \sqrt{\|\nabla f(x^k)\|}\right) \|\nabla f(x^k)\|$$

$$\boxed{\begin{aligned} \alpha^j &= r_j^T r_j / d_j^T B^k d_j \\ x^{j+1} &= x^j \end{aligned}}$$

Mihai Anitescu's Notes

4.6.2 Given initial x^0

for $k = 0, 1, 2$

Set $z^0 = 0$; $r^0 = \nabla f(x^k)$; $d^0 = -\nabla f(x^0)$

for $j = 0, 1, 2, \ldots$

if $d^{j\,T} B^k d^j \le 0$

if $j = 0$

return $p^k = -\nabla f(x^k)$

else

return $p^k = x^j$

end(if)

$$\alpha^j = (r^j)^T r^j \Big/ d^{j\,T} B^k d^j$$

$$x^{j+1} = x^j + \alpha^j d^j$$

$$r^{j+1} = r^j + \alpha^j B^k d^j$$

if $\|r^{j+1}\| \le \varepsilon^k$, return $p^k = x^{j+1}$

$$\beta^{j+1} = (r^{j+1})^T (r^{j+1}) \Big/ (r^j)^T (r^j)$$

$$d^{j+1} = -r^{j+1} + \beta^{j+1} d^j$$

end(for)

Set $x^{k+1} = x^k + \alpha^k p^k$; uday Armijo ( )

end(for)

Mihir Anitesh's Notes

4.6.3 ⊛ By my Lemma: it ALWAYS descends

If near SOSC => superlinear convergence

⊛ Weakness: in $\nabla^2 f(x^k)$ is nearly singular but PD; direction is poor; and need ALOT to compute it. Trust region prevents this...

Mihai Anitescu's Notes

4. Special topics

4.7 $\boxed{\text{CG-Steihaug}}$

(*)

$$\min_{p \in \mathbb{R}^n} m_k(p) = f^k + (\nabla f(x^k))^T p + \frac{1}{2} p^T B^k p$$

s.t $\|p\| \leq \Delta^k$

$B^k = \nabla f^2(x^k)$

(*) I will show only how to solve the subproblem; the rest of machinery is the same.

Mihai Anitescu's Notes

4.7.2 Algorithm 7.2 (CG-Steihaug)

Set $z^0 = 0$; $r^0 = \nabla f(x^k)$; $d^0 = -\nabla f(x^k)$

if $\|r^0\| < \varepsilon^k$; return $\|z^0\| = 0$

for $j = 0, 1, 2 \ldots$

if $d_j^T B_k d^j \geq 0$

find τ such that $p_k^k = z^j + \alpha^j d^j$ minimizes $\|m^k(p^k)\|$ and satisfies $\|p^k\| = \Delta^k$; return p^k

$\alpha^j = r^{jT} r^j / d^{jT} B^k d^j$

$z^{j+1} = z^j + \alpha^j d^j$

if $\|z^{j+1}\| \geq \Delta^k$

find $\tau \geq 0$ such that $z^j + \tau d^j = p^k$ has $\|p^k\| = \Delta^k$

return p^k

$r^{j+1} = r^j + \alpha^j B^k d^j$

if $\|r^{j+1}\| \leq \varepsilon^k$

return $p_k = z^{j+1}$

$\beta^{j+1} = r^{j+1T} r^{j+1} / r^{jT} r^j$

$d^{j+1} = -r^{j+1} + \beta^{j+1} d^j$

end (for)

Minai Antessa's Notes

4.7.3. Observation.

① If $\nabla f(x^k) \geq \varepsilon^k$; algorithm terminates at $m_k(p^k) \leq m^k(p_k^C)$

Proof: If first test fails $\Rightarrow$ $\nabla f(x^k) B^k \nabla f(x^k) \leq 0$; so search of model along d^0 goes all the way to boundary which is exactly CP!! $p^k = -\Delta^k \cdot \frac{\nabla f}{\|\nabla f\|}$

If not; $z_1 = \alpha^0 d^0 = \frac{r_0^T r_0}{d_0^T B^k d^0} \cdot d_0 =$

$$= -\frac{\nabla f(x^k)^T \nabla f(x^k)}{\nabla f(x^k)^T B^k \nabla f(x^k)} \cdot \nabla f(x^k)$$

If $z_1 \leq \Delta^k \Rightarrow$ cauchy point

If not; its truncation is Cauchy point

Mihai Anitescu's Notes

4.7.4 Either way $m(z^1) \le m(p^{k,c})$;

So algorithm is globally convergent!!

(same not true about CG)-LS)

① If the first if triggers:

$$m(p^k) \le m(z^k)$$

Proof

$$m(z^j + \tau d^j) = m(z^k) +$$

$$+ \underbrace{\nabla f(x^k)^T d^j \cdot \tau} + \underbrace{z^k B d^j \cdot \tau} +$$

$$+ \frac{1}{2} \tau^2 d^{jT} B d^j$$

If $\tau = -\operatorname{sgn}(\nabla f(x^k)^T d^j + z^{kT} B d^j)$

at least one of them satisfies:

$$\boxed{m(z^j + \tau d^j) < m(z^j)}$$

Mihai Anitescu's Notes

4. Special topics

4.8 Preconditioning

Idea: If we apply CG, cannot afford to let it run for n steps if n is large

Worst-case convergence: Same as S.D

$$\|x^k - x^*\|_A \le 2\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^k \|x_0 - x^*\|$$

$\kappa(A) = \lambda_n / \lambda_1$: condition number

Idea: cluster eigenvalues

$$Ax = b \qquad Cx = x^1$$

$$C^{-1}AC^{-1}x = C^{-1}b$$

But $\lambda_n / \lambda_1 \ll \lambda_n / \lambda_1$ so convergence is faster

Milan Anitescu's Notes

4.8.2

Idea: Rescale TR...

$$\min_{p \in \mathbb{R}^n} f(x^k) + \nabla f(x^k)^T p + \frac{1}{2} p^T B^k p$$

$$\|Dp\| \leq \Delta$$

$$\begin{cases} \hat{p} = Dp \; ; \; \hat{g}_k = D^{-1} \nabla f(x^k) \\ \hat{B}_k = D^{-T} (\nabla f(x^k)) D^{-1} \end{cases}$$

$(=)$

$$\begin{cases} \min_{\hat{p} \in \mathbb{R}^n} f(x^k) + \hat{g}_k \hat{p} + \frac{1}{2} \hat{p}^T \hat{B}^k \hat{p} \\ \|\hat{p}\| \leq \Delta \end{cases}$$

And now use $\boxed{\text{Steihaug CG}}$

Mihai Anitescu's Notes

4.8.3 How to choose D?

eigs of $\hat{B}^k$ as clustered as possible

$\Rightarrow$ fast convergence

1) ~~Incomp~~ Modified Incomplete Cholesky

2) Diagonal preconditioning

Cholesky

$$\boxed{B^k = LL^T - W}$$

L is obtained by forcing sparsity, pattern, drop tolerance.

Mihai Anitescu's Notes

4.

4.9. Obtaining differentiation information

Finite difference. Approx. the gradient

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x+\varepsilon e_i) - f(x)}{\varepsilon} +$$

Requires $n+1$ fcn evaluations

From Taylor's formula

$$\| f(x+p) - f(x) - \nabla f(x)^T p \| \leq L_2 \cdot \|p\|^2$$

"upper bound on $\nabla f(x)$"

$$\Downarrow$$

$$\frac{\partial f}{\partial x_i}(x) = \frac{f(x+\varepsilon e_i) - f(x)}{\varepsilon} + \delta_\varepsilon$$

$$|\delta_\varepsilon| \leq L_2 \cdot \varepsilon$$

Assume relative error is unit round

$$|\text{fcomp}(f(x)) - f(x)| \leq u \cdot L_f$$

$$|\text{comp}(f(x+\varepsilon_i)) - f(x+\varepsilon_i)| \leq u L_f$$

$$\Rightarrow \text{Error} \leq \frac{2uL_f}{\varepsilon} + L_2 \varepsilon$$

Mihai Anitescu's Notes

4.9.2 How do I choose ε? by minimization

$$\Rightarrow -\frac{2uL_f}{\varepsilon^2} + L/2 = 0 \Rightarrow$$

$$\cancel{\varepsilon^2 = \frac{L}{4uL}}$$

$$\varepsilon^2 = \frac{4uL_f}{L} \Rightarrow \boxed{\varepsilon = O(\sqrt{u})}$$

$$\boxed{u = 10^{-16}} \Rightarrow \boxed{\varepsilon \approx 10^{-8}}$$

~~Should~~ No sense $\boxed{\varepsilon < 10^{-8}}$

Approximating the Hessian

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{f(x+\varepsilon e_i+\varepsilon e_j) - f(x+\varepsilon e_i) - f(x+\varepsilon e_j) + f}{\varepsilon^2} + O(\varepsilon$$

$\Rightarrow$ Need $f(x+\varepsilon e_i+\varepsilon e_j)$ for $\forall i,j,0$

$f(x+\varepsilon_i)$ $\frac{n(n+1)}{2}$

$$\boxed{\frac{n(n-1)}{2} + n = \frac{n(n+1)}{2}} + n \Rightarrow \frac{n(n+3)}{2} \approx n^2/2 + O($$

Mihai Anitescu's Notes

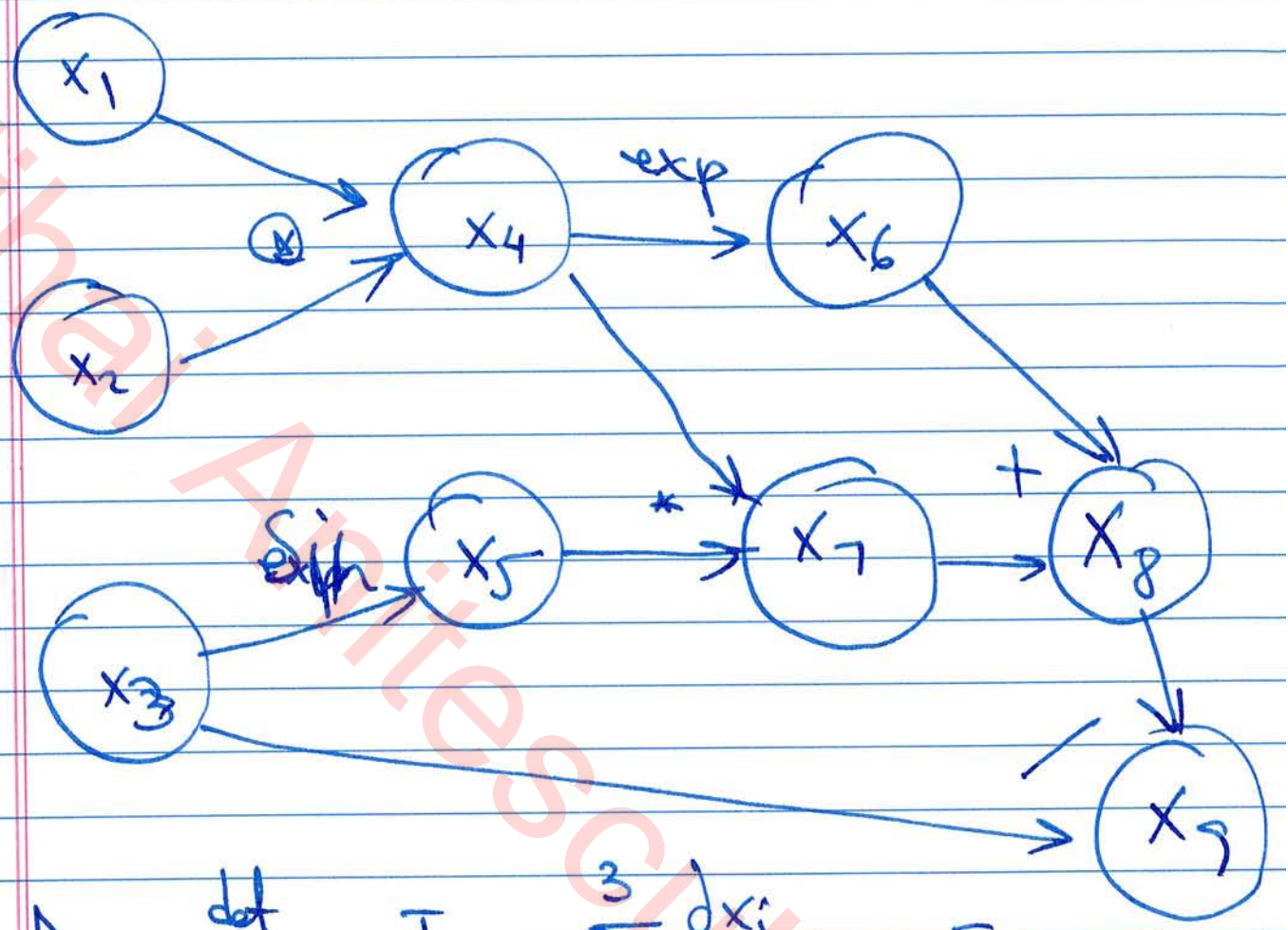
4.9.3

$$\text{Error}(\varepsilon) = \frac{c}{\varepsilon^2} \cdot u + L \cdot \varepsilon$$

$$\Rightarrow \varepsilon = O(\sqrt[3]{u}) \simeq 10^{-6}$$

Mihai Anitescu's Notes

4.10.2 AD: Forward Mod.



$$D_p x_i \overset{\text{def}}{=} \nabla x_i^T p = \sum_{j=1}^{3} \frac{\partial x_i}{\partial x_j} p_j \quad [\text{e.g } 1,0,0]$$

$$D_p x_i = \sum_{j \in \text{parents}(i)} \frac{\partial x_i}{\partial x_j} \cdot D_p x_j$$

Push parents kids. Have all ~~kids~~ parents written

Compute / push

Mihai Anitescu's Notes

4.10.3 The fact that I have to do it per component suggests effort = ~~O(n)~~ order n * time comput funct

On the other hand; once I write child contribution, I can erase. Can compute as I evaluate function.

AD: Reverse Mode

$$\boxed{\frac{\partial f}{\partial x_i} = \sum_{j \in \text{children}(i)} \frac{\partial f}{\partial x_j} \cdot \frac{\partial x_j}{\partial x_i}}$$

$$\frac{\partial f}{\partial x_{fin}} = \frac{\partial x_{fin}}{\partial x_{fin}} = 1$$

Problem: ~~$\frac{\partial f}{}$~~ : I have to store all variable.

Traverse forward → obtain x value

backward → $\frac{\partial f}{\partial x_j}$: "adjoint variables"

Mihai Anitescu's Notes

4.10.4. But, in one sweep; I have All of them

It seems effort is comparable to few value...

Theorem (Griewank). There is a reverse mode strategy which will compute the ∇f in a ~~time~~ no larger than 5 times number of operations the one needed for f

Mihai Anitescu's Notes

4. Tricks of the trade
4.10

Automatic differentiation

The intval package, free for download.

$$f(x) = (x_1 x_2 \sin x_3 + e^{x_1 x_2}) / x_3$$

Reduce to binary operations and few maps:

$$x_4 = x_1 \cdot x_2$$

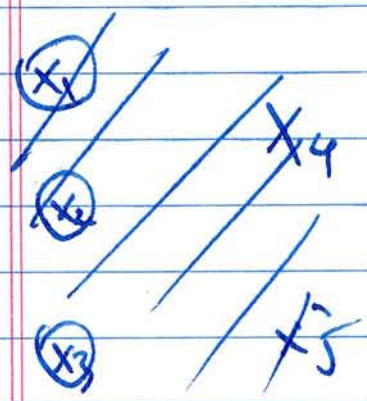
$$x_5 = \sin x_3$$

$$x_6 = e^{x_4}$$

$$x_7 = x_4 \cdot x_5$$

$$x_8 = x_7 + x_6$$

$$x_9 = x_8 / x_3$$



Mihai Anitescu's Notes

4. TRICKS OF THE TRADE

4.11 Quasi-Newton methods

Q: What if I am willing to compute the gradients but not the Hessian?

- Idea: use secant method

$$\nabla^2 f(x_2)\cdot(x_2-x_1) \simeq \nabla f(x_2) - \nabla f(x_1) + O(\|x_2-x_1\|)$$

$$\Rightarrow \nabla^2 f(x_2) = \cancel{\frac{1}{\|x_2\|}}\left[\nabla f(x_2) - \nabla f(x_1)\right]/(x_2-x_1) + O(\|x_2-x_1\|)$$

- In multiple dimensions, the secant condition becomes:

$$\nabla^2 f(x^k)\cdot \overset{\alpha^j p^j}{\overset{\|}{d^j}} \simeq \nabla f(x^j+d^j) - \nabla f(x^j) = g^j + O([\|d^j\|^2, \|x^j - \ldots$$

$j = k-1, \ldots k-n$

$$\Rightarrow \nabla^2 f(x^k) \simeq B^k = [g^{k-n} \ldots g^{k-1}]\cdot[d^{k-n} \ldots d^{k-1}]^{-1}$$

$$B^k \cdot d^j = g^j; \quad j = k-1 \ldots k-n$$

$\Rightarrow$ No need to compute B^k!!

Mihai Anitescu's Notes

4.11.2 ⊛ Problems: 1) What if d^j is short. (Why is it not a good idea to use larger one?)

2) What if not symmetric? PSD.

⊛ Idea; define it iteratively; be sure each iterate is of high enough quality

Digression: Line Search: $B^k p^k = -\nabla f^k$

~~Two ways~~

So, I would like $p^k = -H^k \nabla f^k$; so it is the inverse I may like to approximate.

$$H^k g^j = d^j \quad \text{... and decent}$$

⊛ So: two approaches.

① $\min_B \|B - B^k\|_W$

$B = B^T \quad B \cdot d^k = g^k$

② $\min_H \|H - H^k\|_W$

$H = H^T;$

$H g^k = d^k$

$$\boxed{\| \quad \|_I = \| \quad \|_F = \sum_{i,j=1}^{n} b_{ij}^2}$$

Mihai Anitescu's Notes

4.11.3

$$H \cdot \underline{s}^k = d^k$$

$$H^{k+1} = \left(1 - \rho^k d^k \cdot \underline{s}^{k^T}\right) H^k \left(1 - \rho^k \cdot \underline{s}^k \cdot d^{k^T}\right) + \rho^k d^k d^{k^T}$$

$$\rho^k = d^{k^T} \cdot \underline{s}^k$$

$$\boxed{\text{SMW}: \left(A + ab^T\right)^{-1} = A^{-1} - \frac{A^{-1} a b^T A^{-1}}{1 + b^T A^{-1} a}}$$

<u>Notes</u>

1) Rank 1 update : "the minimum modification

2) Symmetric obvious ; so is consistency

3) $\rho^k > 0 \iff H^{k+1}$ is p.d $\Rightarrow H^k$ is p.d

(if not, perturb d^k; $\underline{s}^k$) it is true near convergence

$\boxed{\cancel{\text{SMW}: (A + bb^T)^{-1} = A^{-1} + \frac{bb^T}{bA^{-1}b}}}$

4) B^k obtained by SMW ; after working out H^{k+1}

5) Update can be done in $O(n^2)$ steps!

(homework)

6) Memory is $O(n^3)$

Minai Antiseptic Notes

4.11.4 BFGS methods (w. line search) $\Rightarrow$ all compute rank 1 updates of B or H

<u>Properties</u> (a lot of them: SR1; BFGS; DFP; Broyden

1) Need $O(n^2)$ per iteration: $O(n^3)$ storage

2) Need only 1 gradient evaluation per step. (no Hessian evaluation)

3) Nevertheless; if n is large and $\nabla^2 f(x)$ is sparse: both computation AND storage may kill you.

4) However, under some conditions

$$\lim_{k \to \infty} \frac{\| B^k p^k - \nabla^2 f(x^*) p^k \|}{\| p^k \|} = 0 \Rightarrow \text{Superlinear convergence.}$$

Mihai Anitescu's Notes