Optimization Algorithms and Simulation Based Optimization

Jorge Nocedal

Northwestern University

Thanks :

Richard Byrd, U. Colorado

X.C. Cai, U. Colorado

Outline

- View from general-purpose optimization research
- Why not use standard methods?
- Restrictions/challenges that must be addressed
- Simulation-based optimization too varied…
- Focus on smooth (or almost smooth) problems
- Focus on derivative-based methods
- Cope with existing simulation packages
- Guide future software development

Challenges for optimization

- Problems so large that must be solved using parallel computers. Information that is needed by optimization method can be scattered in hundreds of processors

- Cannot solve linearized constraints by direct method. Approximate decomposition preconditioners

- Constrained by algorithms/software for simulation
- Approximations, varying resolutions, shocks
- Costly simulations, ill-conditioning, lack of second derivatives

Simulation First, then Optimization

- System of PDEs, DAEs,other…

$$c(v) = 0$$

  Solved with special algorithms/software

- Introduce free parameters u and optimize

$$\begin{aligned} \min \quad & f(u,v) \\ s.t. \quad & c(u,v) = 0 \qquad \text{e.g. system of PDEs} \\ & l \le (u,v) \le a \end{aligned}$$

Will need multiple simulations (not jus a few!)

Focus on equality constraints

Fluid Control by MEMS Devices (Tsai, Byrd,N)

Determine position of flaps to optimize quality, f(u,v), of exhaust flow
subject to incompressible Navier-Stokes equations
c(u,v)=0 on domain with moving boundary

Phase II: boundary control

- Current: convectional actuators
- New: Micro ElectroMecanical Systems
- Flow control, drag reduction, flow mixing in combustion chambers
- Lack of numerical simulators (Navier-Stokes)
- Goal: suppress certain flow structures using MEMS actuators. Minimize turbulence, match a flow pattern
- Inequalities

$$0 \le p(t,x) \le p_{\max}$$

- Handle through interior point method

Two Cases

Solve system $c(v) = 0$

by Newton's method

$$J(v)d = -c(v)$$

Case I: J can be formed and factored;

can use some optimization methods

Case II: J cannot be formed or factored;

largely unexplored in optimization (LANCELOT)

State-of-the-art algorithms

$$F(x) = \begin{pmatrix} \nabla f(x) + \lambda^T c(x) \\ c(x) \end{pmatrix} \quad \begin{pmatrix} W & A^T \\ A & 0 \end{pmatrix} \begin{bmatrix} p \\ l \end{bmatrix} = - \begin{bmatrix} \nabla f \\ c \end{bmatrix}$$

- Active set: Snopt, FilterSQP factor subset of A, reduced Hessian
- Interior: LOQO, etc (full space), KNITRO (reduced space) factor

$$\begin{pmatrix} W & A \\ A^T & 0 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix}$$

- Exception LANCELOT: apply CG

$$\nabla^2 L(x,\lambda) + \mu A A^T$$

- Why factor? $$\begin{pmatrix} W & A^T \\ A & 0 \end{pmatrix} \begin{bmatrix} p \\ l \end{bmatrix} = - \begin{bmatrix} \nabla f \\ c \end{bmatrix}$$

- Satisfy constraints accurately (ill-cond) (LOQO)
- Work on reduced space: SNOPT, FilterSQP, KNITRO

Satisfying the Constraints (Newton Krylov)

System

$$c(v) = 0$$

Solved by Newton's method

$$J(v)d = -c(v)$$

- Linear system using Krylov method; requires only matrix vector products Jw. GMRES, QMR
- Jw approximated by finite differences or computed by automatic differentiation
- Properties of approximate solution?

First Difficulty: Failure of Newton-Krylov

Failure (slow converg) solving

GMRES

$$J(v)d = -c(v)$$

too accurately (oversolving). Walker et al, Cai et al

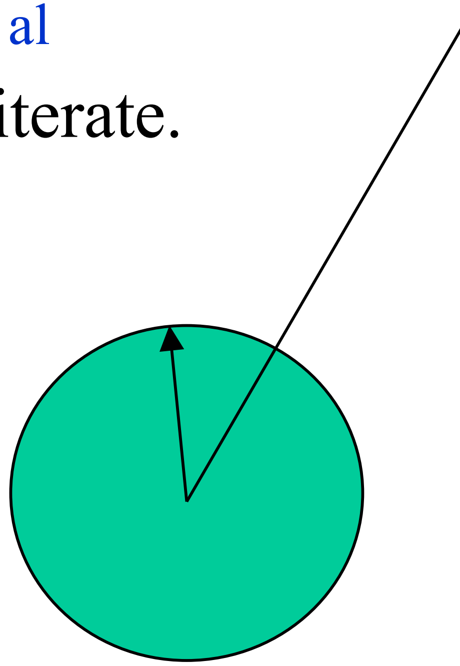
Reduce accuracy (heuristic), select intermediate iterate.
Properties of iteration?

Trust region methods:

- Dogleg method (effective?)

$$(J'J + \lambda I)d = -c(v)$$

- Exact solution (use CG?)



Optimization

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & c(x) = 0 \end{aligned}$$

$$L(x,\lambda) = f(x) + \lambda^T c(x)$$

$$\begin{aligned} \nabla f - A^T \lambda &= 0 \\ c(x) &= 0 \end{aligned}$$

Newton's method, SQP

$$\begin{pmatrix} W & A^T \\ A & 0 \end{pmatrix} \begin{bmatrix} p \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} \nabla f \\ c \end{bmatrix}$$

Adjoint eqn, control eqn
State eqns

$$x^+ = x + \alpha p, \qquad \lambda^+$$

Textbook: Full space, null space, range space

Part I: Full Space Approach

Full Space Approach: ideal case

$$F(x) = \begin{pmatrix} \nabla f(x) + \lambda^T c(x) \\ c(x) \end{pmatrix} \qquad \begin{pmatrix} W & A^T \\ A & 0 \end{pmatrix} \begin{bmatrix} p \\ l \end{bmatrix} = -\begin{bmatrix} \nabla f \\ c \end{bmatrix}$$

Solve by direct method

Limits size

Merit function $\quad f(x) + \mu \| c(x) \|$

Lack of positive definiteness. W+eI (Vanderbei et al)

Jacobian singularities

Full Space Approach: iterative method

$$F(x) = \begin{pmatrix} \nabla f(x) + \lambda^T c(x) \\ c(x) \end{pmatrix} \qquad \begin{pmatrix} W & A^T \\ A & 0 \end{pmatrix} \begin{bmatrix} p \\ l \end{bmatrix} = -\begin{bmatrix} \nabla f \\ c \end{bmatrix}$$

GMRES, QMR

Merit function ||F(x)||:

can trap iteration, max not min, converge to singularity

Merit function $f(x) + \mu \| c(x) \|$

No descent for inexact Newton step

Projected CG: good way to deal with indefiniteness

Factor Jacobian – possible?

Pros/Cons

- Newton- KKT may lead to a max
- Why not used before?
- Subsystems better conditioned than KKT
- Must solve KKT to high accuracy to make progress towards a stationary point

- Good match with PDE solvers
- Matrix-free, Schwarz preconditioners (divide and conquer; subdomains to different processors)

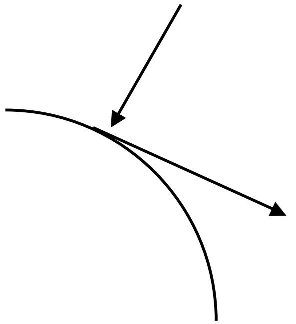
Part II: Null Space Approach

Null Space Approach

Iterative method on null space of constraints

Decouple Newton equations

$$\begin{pmatrix} W & A^T \\ A & 0 \end{pmatrix} \begin{bmatrix} p \\ l \end{bmatrix} = -\begin{bmatrix} \nabla f \\ c \end{bmatrix}$$



Nulls space basis $A^T Z = 0$

$$Ap_c = -c \qquad p = Zp_Z + p_c$$

$$Z^T W p = -Z^T \nabla f \quad \Rightarrow \quad Z^T W Z p_Z = -Z^T \nabla f - Z^T W p_c$$

$$p = Zp_Z + p_c$$

Advantages of null space approach:

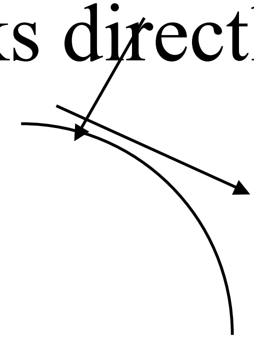
1. Symmetric system, can use CG, handle indefiniteness (trust region)

$$Z^T W Z p_Z = -Z^T \nabla f - Z^T W p_C$$

CG: low storage, computation, reliable convergence properties

$$p = Z p_Z + p_C \qquad \|p\| \leq \Delta$$

2. Step p is a descent direction. Works directly on optimality, not simply solution of KKT system. Good quality



Challenges of null space approach

- Generate iterates in null space of A: $Ap + c = 0$
- Incorporate preconditioner for A and Hessian:

$$\overline{W} \approx W \equiv \nabla^2 L(x,\lambda) \qquad \overline{\mathrm{A}} \approx \mathrm{A}$$

- Representation in null space: partition into state and decision variables

$$A = [\,A_S \quad A_D\,] \qquad\qquad Z = \begin{bmatrix} -A_S^{-1} A_D \\ I \end{bmatrix}$$

$$Z,\ Z^T \quad \Rightarrow \quad (A_s^T)^{-1},\ \mathrm{A}_s^{-1}$$

- Every CG iteration requires one soln of linearized PDE. Factor A_s. Not possible! Use super LU to factor A_s

Preconditioning

- Assume preconditioners

$$\overline{W} \approx W \equiv \nabla^2 L(x,\lambda) \qquad \overline{\mathrm{A}} \approx \mathrm{A}$$

- $\overline{\mathrm{A}}$ used as before
- First idea: to precondition

$$Z^T W Z p_Z = -Z^T \nabla f$$

form

$$\overline{Z}^T \overline{W}\,\overline{Z} p_Z$$

not practical

One possible approach Coleman-Verma

$$\overline{W} \approx W \equiv \nabla^2 L(x, \lambda) \qquad \overline{\mathrm{A}} \approx \mathrm{A}$$

- Computing $(Z^T W Z)^{-1} r$
- Is equivalent to solving

$$\begin{pmatrix} \overline{W} & \overline{A}^T \\ \overline{A} & 0 \end{pmatrix} \begin{bmatrix} z \\ u \end{bmatrix} = -\begin{bmatrix} 0 \\ Z^T \nabla f \\ 0 \end{bmatrix}$$

Similar to Biros-Ghattas reduced preconditioner (CG, descent)

Null-Space Approach: Pros

- Reduced Hessian symmetric, pd.
- Feasible up to first order
- Reduced space preconditioner
- Indefiniteness treated by trust regions
- Can handle multilevels in A

Null-Space Approach: Cons

- Must estimate/compute Z, Z^T
- Expensive to compute and store A
- Factor of A difficult to parallelize
- Matrix-vector products with Z, Z^T costly
- Too many PDE solves? 2 for every CG iteration!
- Null space Krylov (GMRES): one matrix vector product per iteration

Ideal solution: a compromise?

Part III: Unconstrained Formulation