The Merits of Keeping it Smooth: Implementing a Smooth Exact Penalty Function for Nonlinear Optimization

Ron Estrin
ICME, Stanford University

UBC SCAIM Seminar
Sept. 18, 2018

Joint Work with Michael Friedlander, Dominique Orban, and Michael Saunders

Constrained Optimization

Equality-constrained nonlinear program:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad c(x) = 0,$$

with n variables, $m \leq n$ constraints, and $f, c \in C_3$.

Focus on equality constraints only for now, discuss inequality/bound constraints later.

Such problems are ubiquitous in the computational sciences for applications including:

- Structural Design
- Systems biology
- Machine learning
- Optimal control
- Robotics planning
- ...

Example: PDE-Constrained Optimal Control

Inverse Poisson Problem:

$$\begin{aligned}
\min_{u,z} \quad & \frac{1}{2}\int_\Omega (u(x)-\bar{u}(x))^2 dx + \frac{\alpha}{2}\int_\Omega z(x)^2 dx \\
\text{s.t} \quad & -\nabla\cdot(z\nabla u) = f, \qquad \text{in } \Omega \\
& u = 0, \qquad \text{on } \partial\Omega,
\end{aligned}$$

where $\Omega = [0,1]^2$, $\bar{u}$ is observed data, and f is given source.

- u is the *state* (e.g., temperature)
- z is the *control* (e.g., heat conduction coefficient)

Example: PDE-Constrained Optimal Control

Inverse Poisson Problem:

$$\begin{aligned}
\min_{u,z} \quad & \frac{1}{2}\int_\Omega (u(x)-\bar{u}(x))^2 dx + \frac{\alpha}{2}\int_\Omega z(x)^2 dx \\
\text{s.t} \quad & -\nabla\cdot(z\nabla u) = f, \qquad \text{in } \Omega \\
& u = 0, \qquad \text{on } \partial\Omega,
\end{aligned}$$

where $\Omega = [0,1]^2$, $\bar{u}$ is observed data, and f is given source.

- u is the *state* (e.g., temperature)
- z is the *control* (e.g., heat conduction coefficient)

Problem structure:

- Want computed state to match observations
- Want control variable to be "reasonable" (regularization)
- State must be physically meaningful given control

Solving Constrained Optimization Problems

Necessary conditions for optimal primal-dual solution (x^*, y^*):

$$\begin{aligned} c(x^*) &= 0, \\ g(x^*) &= A(x^*)y^*. \end{aligned}$$

Equivalently, $L(x, y) = f(x) - y^T c(x)$ and

$$\nabla L(x^*, y^*) = 0.$$

Constrained solvers built around root-finding of above equations.

Notation:

$$g := \nabla f(x), \qquad A := \begin{bmatrix} \nabla c(x) \end{bmatrix}.$$

Solving Constrained Optimization Problems

Plethora of methods available, but still an active area of research!

Common difficulties with constrained methods:

- Finding a feasible point is just as difficult as solving problem
- Complicated heuristics to trade off optimality vs. feasibility
- Require feasibility restoration phases
- Most solvers built using explicit matrix-factorizations

Solving Constrained Optimization Problems

Plethora of methods available, but still an active area of research!

Common difficulties with constrained methods:

- Finding a feasible point is just as difficult as solving problem
- Complicated heuristics to trade off optimality vs. feasibility
- Require feasibility restoration phases
- Most solvers built using explicit matrix-factorizations

Alternative approach:

- Move constraint violation into the objective
- Solve unconstrained problem
- Additionally: want to avoid matrix-factorizations

Penalty methods for Nonlinear Programming

Add some measure of constraint violation in objective

- **Quadratic penalty**

$$\min_x f(x) + \frac{\sigma_k}{2}\|c(x)\|_2^2$$

  - Perturbs the solution.
  - Need to solve sequence of problems with $\sigma_k \to \infty$.

Penalty methods for Nonlinear Programming

Add some measure of constraint violation in objective

- **Quadratic penalty**

$$\min_x f(x) + \frac{\sigma_k}{2}\|c(x)\|_2^2$$

  - Perturbs the solution.
  - Need to solve sequence of problems with $\sigma_k \to \infty$.

- ℓ_1**-penalty**

$$\min_x f(x) + \sigma\|c(x)\|_1$$

  - Non-smooth.

Penalty methods

Add some measure of constraint violation in objective

- **Augmented Lagrangian method**

$$\min_{x} f(x) - y_k^T c(x) + \frac{\sigma_k}{2}\|c(x)\|_2^2$$

  - Need to solve sequence of problems.

Penalty methods

Add some measure of constraint violation in objective

- **Augmented Lagrangian method**

$$\min_x f(x) - y_k^T c(x) + \frac{\sigma_k}{2} \|c(x)\|_2^2$$

  - Need to solve sequence of problems.

- Would like *exact* penalty function which is *smooth*...

Fletcher's Penalty Function

Primal only Lagrangian:

$$L(x, y) := f(x) - y^T c(x)$$

Fletcher's penalty function:

$$\begin{aligned} \phi_\sigma(x) &:= f(x) - c(x)^T y_\sigma(x) \\ y_\sigma(x) &:= \arg\min_y \tfrac{1}{2}\|g - Ay\|_2^2 + \sigma c(x)^T y \end{aligned}$$

Notation:

$$g := \nabla f(x), \qquad A := \begin{bmatrix} \nabla c(x) \end{bmatrix}, \qquad Y_\sigma := \begin{bmatrix} \nabla y_\sigma(x) \end{bmatrix}.$$

Variations

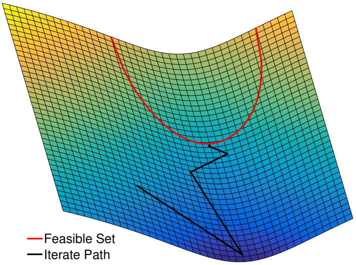
Smooth Primal (Fletcher, 1970)

$$\phi_{0,\rho}(x) := f(x) - y(x)^T c(x) + \tfrac{1}{2}\rho\|c(x)\|_2^2$$

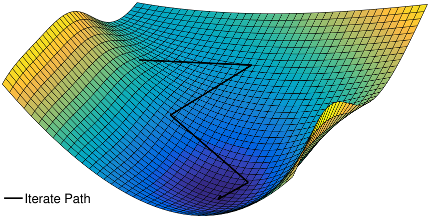
Smooth Primal-Dual (Di Pillo and Grippo, 1989; Zavala and Anitescu, 2014)

$$\psi_{\alpha,\beta}(x,y) := L(x,y) + \tfrac{1}{2}\alpha\|c(x)\|_2^2 + \tfrac{1}{2}\beta\|\nabla L(x,y)\|_2^2$$

An Illustration



(a) 'hs007' with feasible set.



(b) ϕ_σ for problem 'hs007'.

Derivatives

More notation (sorry!):

$$g_\sigma(x) = \nabla_x L(x, y_\sigma), \qquad H_\sigma(x) = \nabla_{xx} L(x, y_\sigma)$$

Derivatives

More notation (sorry!):

$$g_\sigma(x) = \nabla_x L(x, y_\sigma), \qquad H_\sigma(x) = \nabla_{xx} L(x, y_\sigma)$$

Gradient of penalty function:

$$\nabla \phi_\sigma(x) = g_\sigma - Y_\sigma c$$

Hessian of penalty function:

$$\begin{aligned}\nabla^2 \phi_\sigma(x) &= H - \sum_{i=1}^{m} (y_\sigma)_i H_i - A Y_\sigma^T - Y_\sigma A^T - \nabla[Y_\sigma(\cdot)c] \\ &= H_\sigma - A Y_\sigma^T - Y_\sigma A^T - \nabla[Y_\sigma(\cdot)c]\end{aligned}$$

First-Order Optimality Conditions

(x_*, y_*) is first-order KKT point:

$$c(x_*) = 0, \qquad g(x_*) - A(x_*)y_* = 0.$$

First-Order Optimality Conditions

(x_*, y_*) is first-order KKT point:

$$c(x_*) = 0, \qquad g(x_*) - A(x_*)y_* = 0.$$

Gradient of penalty function:

$$\nabla \phi_\sigma(x) = g - Ay_\sigma - Y_\sigma c.$$

Then $y_* = y_\sigma(x_*)$ and $\nabla \phi_\sigma(x_*) = 0$

$\implies$ **1st order KKT points** are **stationary points** of ϕ_σ.

Second-Order Optimality Conditions

(x_*, y_*) is second-order KKT point:

$$d^T \nabla^2_{xx} L(x_*, y_*) d \geq 0, \qquad \text{for } d \text{ such that } A(x_*)^T d = 0$$

Second-Order Optimality Conditions

(x_*, y_*) is second-order KKT point:

$$d^T \nabla^2_{xx} L(x_*, y_*) d \geq 0, \qquad \text{for } d \text{ such that } A(x_*)^T d = 0$$

Hessian of Penalty function:

$$\begin{aligned} \nabla^2 \phi_\sigma(x_*) &= H_\sigma - AY_\sigma^T - Y_\sigma A^T - \nabla[Y_\sigma(\cdot)c] \\ &= H_\sigma - H_\sigma P_A - P_A H_\sigma + 2\sigma P_A \\ &= \bar{P}_A H_\sigma \bar{P}_A - P_A H_\sigma P_A + 2\sigma P_A \end{aligned}$$

Projectors: $\qquad P_A = A(A^T A)^{-1} A^T, \qquad \bar{P}_A = I - P_A.$

Second-Order Optimality Conditions

Hessian of Penalty function:

$$\nabla^2\phi_\sigma(x_*) = \bar{P}_A H_\sigma \bar{P}_A - P_A H_\sigma P_A + 2\sigma P_A$$

Then $\nabla^2\phi_\sigma(x_*) \succeq 0 \iff \sigma \geq \sigma_* = \frac{1}{2}\lambda_{\max}(P_A H_\sigma P_A)$

$\implies$ **2nd order KKT points** are **minimizers** of ϕ_σ for $\sigma \geq \sigma_*$.

Solving NLP with ϕ_σ

$\phi_\sigma(x)$ is a smooth, exact penalty function.

If σ is chosen large enough, it is enough to minimize $\phi_\sigma(x)$ once to obtain KKT point to original NLP.

- (Weakly-exact: spurious minima rare but still possible)
- Can use $\phi_{\sigma,\rho} = \phi_\sigma(x) + \frac{1}{2}\rho\|c(x)\|_2^2$ to resolve this

Benefits of this approach

- Conceptually simple minimization (no optimality/feasibility trade-off heuristics)
- No feasibility restoration*
- No Maratos effect (slow convergence for nonsmooth penalties)
- Solve single unconstrained problem if σ known in advance

Benefits of this approach

- Conceptually simple minimization (no optimality/feasibility trade-off heuristics)
- No feasibility restoration*
- No Maratos effect (slow convergence for nonsmooth penalties)
- Solve single unconstrained problem if σ known in advance
- Naturally accommodates *matrix-free* optimization (only matrix-vector products; no factorizations)
- Naturally accommodates inexact optimization

The technical part

In order to minimize ϕ_σ, need procedures for:

- function evaluation, $\phi_\sigma(x)$,
- gradient evaluation, $\nabla\phi_\sigma(x)$, and
- approximate Hessian-vector products: compute for any $v \in \mathbb{R}^n$

$$u = B(x)v, \qquad B(x) \approx \nabla^2\phi_\sigma(x)$$

The technical part

In order to minimize ϕ_σ, need procedures for:

- function evaluation, $\phi_\sigma(x)$,
- gradient evaluation, $\nabla\phi_\sigma(x)$, and
- approximate Hessian-vector products: compute for any $v \in \mathbb{R}^n$

$$u = B(x)v, \qquad B(x) \approx \nabla^2\phi_\sigma(x)$$

Therefore need procedures for computing:

- y_σ
- products with Y_σ and Y_σ^T

Function Evaluation

Multiplier estimate:

$$y_\sigma(x) = \arg\min_y \frac{1}{2}\|g - Ay\|_2^2 + \sigma c(x)^T y$$

which is solved by

$$A^T A y_\sigma = A^T g - \sigma c$$

or equivalently,

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} g_\sigma \\ y_\sigma \end{bmatrix} = \begin{bmatrix} g \\ \sigma c \end{bmatrix}$$

Products with Y_σ

First, Jacobian of y_σ:

$$Y_\sigma = (H_\sigma - \sigma I)A(A^TA)^{-1} + S(x, g_\sigma)^T(A^TA)^{-1}$$

where

$$S(x,v)u = \begin{bmatrix} v^TH_1u \\ \vdots \\ v^TH_mu \end{bmatrix}, \qquad S(x,v)^Tu = \sum_{i=1}^m u_iH_iv$$

(Notice that $S(x_*, g_\sigma) = 0$ since $g_\sigma = 0$ at solution).

Products with Y_σ

$$\begin{aligned} Y_\sigma u &= (H_\sigma - \sigma I)A(A^T A)^{-1} u + S(x, g_\sigma)^T (A^T A)^{-1} u \\ &= (H_\sigma - \sigma I)(-w) + \sum_{i=1}^{m} v_i H_i g_\sigma \end{aligned}$$

where

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ -u \end{bmatrix}.$$

Products with Y_σ^T

$$
\begin{aligned}
Y_\sigma^T u &= (A^TA)^{-1}A^T(H_\sigma - \sigma I)u + (A^TA)^{-1}S(x, g_\sigma)u \\
&= v
\end{aligned}
$$

where

$$
\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} (H_\sigma - \sigma I)u \\ -S(x, g_\sigma)u \end{bmatrix}.
$$

Gradient Computation

Gradient:

$$\nabla \phi_\sigma = g_\sigma - Y_\sigma c$$

Need to solve one additional augmented system.

Hessian products

Hessian product:

$$
\begin{aligned}
\nabla^2 \phi_\sigma(x) v &= H_\sigma v - A Y_\sigma^T v - Y_\sigma A^T v - \nabla[Y_\sigma(\cdot) c] v \\
&\approx H_\sigma v - A Y_\sigma^T v - Y_\sigma A^T v \qquad \text{(remove third derivatives)} \\
&\approx H_\sigma v - P_A H_\sigma v - H_\sigma P_A v + 2\sigma P_A v
\end{aligned}
$$

Can further approximate by removing terms which are zero at solution.

Two augmented system solves per product.

Hessian products

Hessian product:

$$
\begin{aligned}
\nabla^2\phi_\sigma(x)v &= H_\sigma v - AY_\sigma^T v - Y_\sigma A^T v - \nabla[Y_\sigma(\cdot)c]v \\
&\approx H_\sigma v - AY_\sigma^T v - Y_\sigma A^T v \qquad \text{(remove third derivatives)} \\
&\approx H_\sigma v - P_A H_\sigma v - H_\sigma P_A v + 2\sigma P_A v
\end{aligned}
$$

Can further approximate by removing terms which are zero at solution.

Two augmented system solves per product.

Fletcher showed that even with these approximations, **superlinear/quadratic convergence** can still be retained.

“Algorithm”

$\phi_\sigma(x)$ is smooth and exact.

"Algorithm"

$\phi_\sigma(x)$ is smooth and exact.

Can evaluate function, gradient and approximate Hessian products.

“Algorithm”

$\phi_\sigma(x)$ is smooth and exact.

Can evaluate function, gradient and approximate Hessian products.

Pick your favourite unconstrained minimizer, and done!

“Algorithm”

$\phi_\sigma(x)$ is smooth and exact.

Can evaluate function, gradient and approximate Hessian products.

Pick your favourite unconstrained minimizer, and done!

The end! Questions?

“Algorithm”

$\phi_\sigma(x)$ is smooth and exact.

Can evaluate function, gradient and approximate Hessian products.

Pick your favourite unconstrained minimizer, and done!

The end! Questions?
...Not quite

Practicalities

A few notes and observations:

Practicalities

A few notes and observations:

- Trust-region $>$ linesearch (indefinite Hessians)

Practicalities

A few notes and observations:

- Trust-region $>$ linesearch (indefinite Hessians)
- Solving augmented system (direct vs. iterative)

Practicalities

A few notes and observations:

- Trust-region $>$ linesearch (indefinite Hessians)
- Solving augmented system (direct vs. iterative)
- Handling linear constraints

Practicalities

A few notes and observations:

- Trust-region $>$ linesearch (indefinite Hessians)
- Solving augmented system (direct vs. iterative)
- Handling linear constraints
- Singular Jacobians (regularization)

Practicalities

A few notes and observations:

- Trust-region $>$ linesearch (indefinite Hessians)
- Solving augmented system (direct vs. iterative)
- Handling linear constraints
- Singular Jacobians (regularization)
- Adjusting penalty parameter, initial points, unboundedness...

Solving the Augmented System

- Direct Solvers
  - Factorize augmented system once per iteration
  - Sparse LDL
  - Semi-normal equations (Q-less QR) + iterative refinement
- Iterative methods
  - Results in *matrix-free* implementation
  - Inexact function/gradient evaluation controlled via iterative solver tolerance
  - Efficient with good preconditioners

Solving the Augmented System: LSLQ/LNLQ

Iterative methods for least-squares (LSLQ) or least-norm problems (LNLQ).

- Equivalent to SYMMLQ on normal equations
- Given $\sigma_{est} \leq \sigma_{\min}(A)$, can monitor $\|w_* - w_k\|$ and $\|v_* - v\|$
- For near-optimal preconditioners, where $A^T = \begin{bmatrix} A_u^T & A_z^T \end{bmatrix}$ and $\mathcal{P} = A_u^T A_u$, then $\sigma_{\min}\left(\mathcal{P}^{-1} A^T A\right) \geq 1$

Inexact Evaluation

Certain trust-region methods (Conn, Gould and Toint 2000; Heinkenschloss and Ridzal, 2014) converge provided that

$$\begin{aligned} \|\phi_\sigma - \widetilde{\phi_\sigma}\| &\le M\eta_1, \\ \|\nabla\phi_\sigma - \widetilde{\nabla\phi_\sigma}\| &\le M\eta_2, \end{aligned}$$

where η_i is a prescribed accuracy, and $M > 0$ is a fixed constant (need not be known).

Inexact Evaluation

Certain trust-region methods (Conn, Gould and Toint 2000; Heinkenschloss and Ridzal, 2014) converge provided that

$$\|\phi_\sigma - \widetilde{\phi_\sigma}\| \le M\eta_1,$$
$$\|\nabla\phi_\sigma - \widetilde{\nabla\phi_\sigma}\| \le M\eta_2,$$

where η_i is a prescribed accuracy, and $M > 0$ is a fixed constant (need not be known).

Can bound terms according to residual or error of augmented system.

Error expressions tedious—in practice, use ad-hoc fixed error bound.

Handling Linear Constraints

Suppose some constraints are linear:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad c(x) = 0, \qquad B^T x = d,$$

Handling Linear Constraints

Suppose some constraints are linear:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad c(x) = 0, \qquad B^T x = d,$$

Change penalty function minimization to:

$$\begin{aligned} \min_{x} \quad & \phi_\sigma(x) := f(x) - c(x)^T y_\sigma(x) \\ \text{s.t.} \quad & B^T x = d \end{aligned}$$

Handling Linear Constraints

Suppose some constraints are linear:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad c(x) = 0, \qquad B^T x = d,$$

Change penalty function minimization to:

$$\begin{aligned} \min_x \quad & \phi_\sigma(x) := f(x) - c(x)^T y_\sigma(x) \\ \text{s.t.} \quad & B^T x = d \end{aligned}$$

Benefits:

- Threshold value σ_* decreases
- Penalty function better conditioned

Regularization

If $A = \nabla c$ is rank-deficient, then ϕ_σ can be undefined.

Regularization

If $A = \nabla c$ is rank-deficient, then ϕ_σ can be undefined.

Regularize least-squares problem:

$$\begin{aligned}
\phi_\sigma(x;\delta) &= f(x) - c(x)^T y_\sigma(x;\delta) \\
y_\sigma(x;\delta) &= \arg\min_y \frac{1}{2} \left\| \begin{bmatrix} g \\ 0 \end{bmatrix} - \begin{bmatrix} A \\ \delta I \end{bmatrix} y \right\|_2^2 + \sigma c(x)^T y
\end{aligned}$$

Regularization

If $A = \nabla c$ is rank-deficient, then ϕ_σ can be undefined.

Regularize least-squares problem:

$$\begin{aligned} \phi_\sigma(x;\delta) &= f(x) - c(x)^T y_\sigma(x;\delta) \\ y_\sigma(x;\delta) &= \arg\min_y \frac{1}{2} \left\| \begin{bmatrix} g \\ 0 \end{bmatrix} - \begin{bmatrix} A \\ \delta I \end{bmatrix} y \right\|_2^2 + \sigma c(x)^T y \end{aligned}$$

Only change: augmented system becomes

$$\begin{bmatrix} I & A \\ A^T & -\delta^2 I \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

Regularization

Assume that $A(x^*)$ has full-rank at x_* (LICQ).

- $\delta > 0$ perturbs the solution.

Regularization

Assume that $A(x^*)$ has full-rank at x_* (LICQ).

- $\delta > 0$ perturbs the solution.
- Want $\delta \to 0$ to recover exact solution.

Regularization

Assume that $A(x^*)$ has full-rank at x_* (LICQ).

- $\delta > 0$ perturbs the solution.
- Want $\delta \to 0$ to recover exact solution.
- Want to retain superlinear/quadratic convergence.

Regularization

Assume that $A(x^*)$ has full-rank at x_* (LICQ).

- $\delta > 0$ perturbs the solution.
- Want $\delta \to 0$ to recover exact solution.
- Want to retain superlinear/quadratic convergence.

---

for k=1,2,... **do**
  $\delta_k \leftarrow \max\left\{\min\{\delta_{k-1}, \|\nabla\phi_\sigma(x_k;\delta_{k-1})\|\}, \delta_{k-1}^2\right\}$
  Get x_{k+1} from one step on $\phi_\sigma(x_k;\delta_k)$
end for

If quadratically convergent method used, entire method above remains quadratically convergent.

Inequality Constraints

Consider problem

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0, \\ & x \geq 0. \end{aligned}$$

Inequality Constraints

Consider problem

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0, \\ & x \geq 0. \end{aligned}$$

Modify Fletcher's penalty function to

$$\begin{aligned} \min_x \quad & \phi_\sigma(x) := f(x) - c(x)^T y_\sigma(x) \\ \text{s.t.} \quad & x \geq 0 \\ & y_\sigma(x) := \arg\min_y \frac{1}{2}\|g - Ay\|_{X}^2 + \sigma c(x)^T y \end{aligned}$$

Inequality Constraints

- Minimize smooth function over bound constraints
- Smoothness holds for $x > 0$ (use interior method)
- Exactness still holds
- Need to now solve augmented system:

$$\begin{bmatrix} I & X^{\frac{1}{2}}A \\ A^T X^{\frac{1}{2}} & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Numerical Experiments: Inverse Poisson Problem

Solve inverse-poisson problem from earlier.

Discretize using finite elements, $n = 2050$, $m = 961$, $\alpha = 10^{-4}$.

Use Matlab implementation of Newton-CG trust-region solver (TRON). Solve linear system with error accuracy η:

η	Iter.	$\#Hv$	$\#Av$	$\#A^Tv$
10^{-2}	29	874	1794	2608
10^{-4}	27	830	1950	2728
10^{-6}	27	866	2317	3129
10^{-8}	27	866	2673	3485
10^{-10}	27	866	3145	3957

Thanks to Drew Kouri (Sandia) for implementing the model!

Numerical Experiments: Linear Constraints

Problem	n	m_{lin}	m_{nln}	σ^*_{impl}	σ^*_{expl}	σ	Impl.	Expl.
Chain100	202	102	1	0.0047	0	10^{-3} 0.005	$*$ 8	13 10
Chain200	402	202	1	0.0024	0	10^{-3} 0.003	$*$ 7	11 10
Chain400	802	402	1	0.0012	0	10^{-3} 0.002	$*$ 7	10 10
Channel100	800	400	400	0	0	10^{-3} 1	– –	5 5
Channel200	1600	800	800	0	0	10^{-3} 1	– –	5 5
Channel400	1600	800	800	0	0	10^{-3} 1	– –	5 5

Numerical Experiments: Regularization (mss1)

- $n = 90$, $m = 73$, $\sigma = 10^3$
- Fails at step 1 when $\delta_0 = 0$
- Start with $\delta_0 = 10^{-2}$

iter	merit	objective	opt Error	du Feas	nln Feas	||y||	penalty	delta
0	3.525544e+05	-4.050000e+03	5.964e+03	4.979e+02	8.900e+01	1.818e+06	1.0e+03	1.0e-02
1	3.388800e+05	-1.806637e+03	8.528e+03	3.277e+02	3.915e+01	1.815e+06		
2	3.369551e+05	-1.802047e+03	8.540e+03	3.263e+02	3.905e+01	1.810e+06		
3	3.350170e+05	-1.797264e+03	8.533e+03	3.271e+02	3.896e+01	1.804e+06		
4	3.330739e+05	-1.792274e+03	8.507e+03	3.279e+02	3.886e+01	1.799e+06		
5	3.311332e+05	-1.787067e+03	8.465e+03	3.287e+02	3.877e+01	1.794e+06		
...								
45	-1.600001e+01	-1.600155e+01	8.443e-04	4.761e-02	1.656e-06	1.741e+02		
46	-1.600001e+01	-1.600154e+01	6.603e-05	4.748e-02	1.656e-06	1.741e+02		1.0e-04
47	-1.600000e+01	-1.599998e+01	6.924e-03	3.513e-03	3.921e-07	1.753e+02		
48	-1.600000e+01	-1.600000e+01	1.898e-04	1.905e-04	9.392e-10	1.753e+02		
49	-1.600000e+01	-1.600000e+01	1.762e-05	1.762e-05	2.893e-10	1.753e+02		
50	-1.600000e+01	-1.600000e+01	1.813e-06	5.356e-06	1.674e-10	1.753e+02		
51	-1.600000e+01	-1.600000e+01	5.934e-08	5.361e-06	1.660e-10	1.753e+02		
52	-1.600000e+01	-1.600000e+01	4.840e-09	5.554e-06	1.663e-10	1.753e+02		1.0e-07
53	-1.600000e+01	-1.600000e+01	7.565e-08	3.905e-08	4.923e-12	1.753e+02		

EXIT: Optimal solution found

Conclusions and Future Work

Simpler* approach to nonlinear programming!

- Shows promise when augmented systems efficiently solvable (e.g. PDE-constrained optimization)
- Current implementation Matlab and C++ (part of ROL package of Sandia's Trilinos library)

Conclusions and Future Work

Simpler* approach to nonlinear programming!

- Shows promise when augmented systems efficiently solvable (e.g. PDE-constrained optimization)
- Current implementation Matlab and C++ (part of ROL package of Sandia's Trilinos library)

Many practical matters to be resolved:

- Robust tolerance rules for inexact solves
- Good heuristics for updating penalty parameter
- Stability of inequality constrained problems near boundary
- Test on more PDE-constrained optimization problems!