

Lagrange-Newton-Krylov-Schur-Schwarz: Matrix-free Methods for PDE-constrained Optimization

Paul D. Hovland

Mathematics & Computer Science Division, Argonne National Laboratory

David E. Keyes

Mathematics & Statistics Department, Old Dominion University

Institute for Scientific Computing Research, Lawrence Livermore National Laboratory

Institute for Computer Applications in Sci. & Eng., NASA Langley Research Center

Lois C. McInnes

Mathematics & Computer Science Division, Argonne National Laboratory

Widodo Samyono

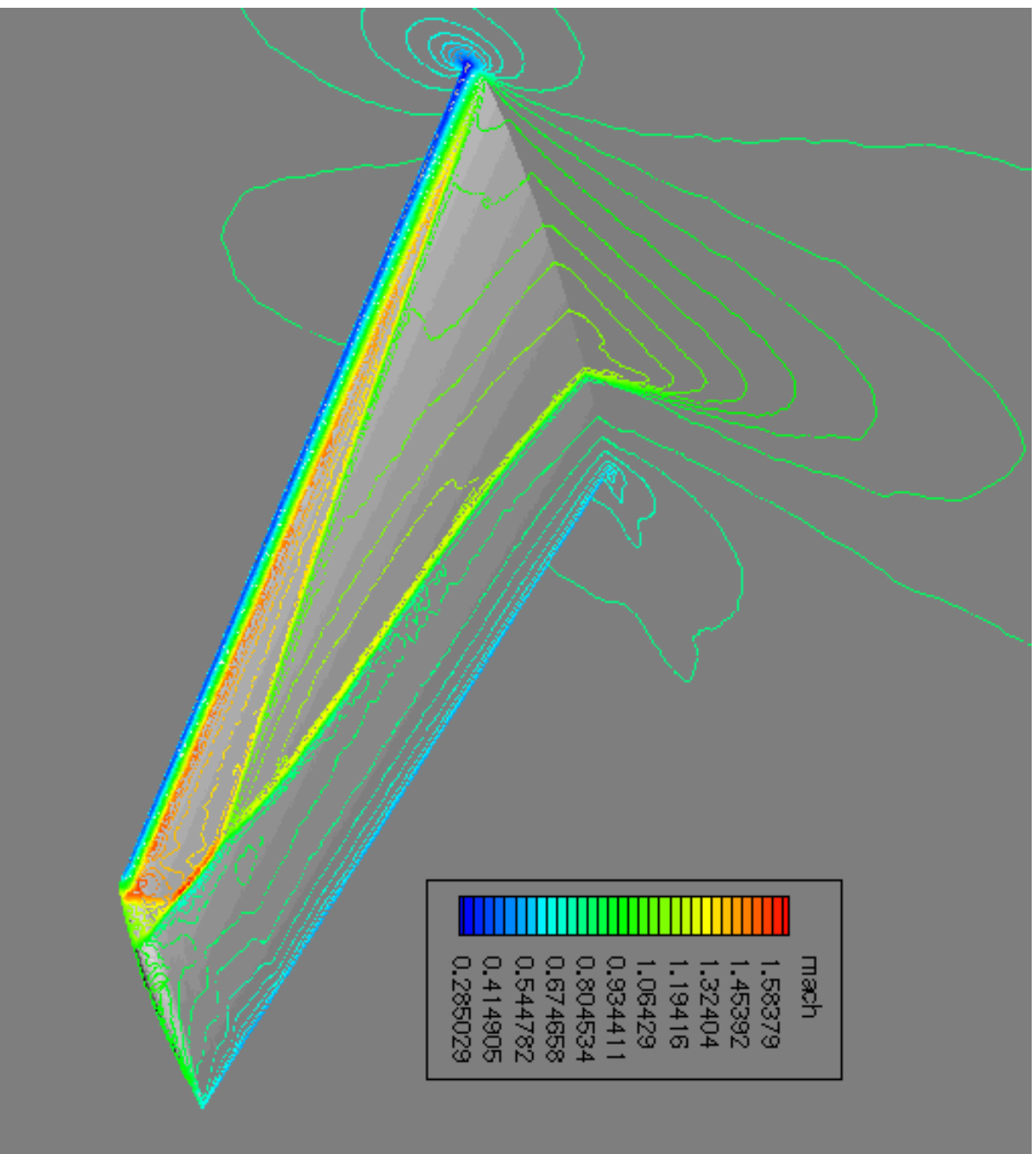
Mathematics & Statistics Department, Old Dominion University

Perspective

“Have parallel PDE solver, will optimize”

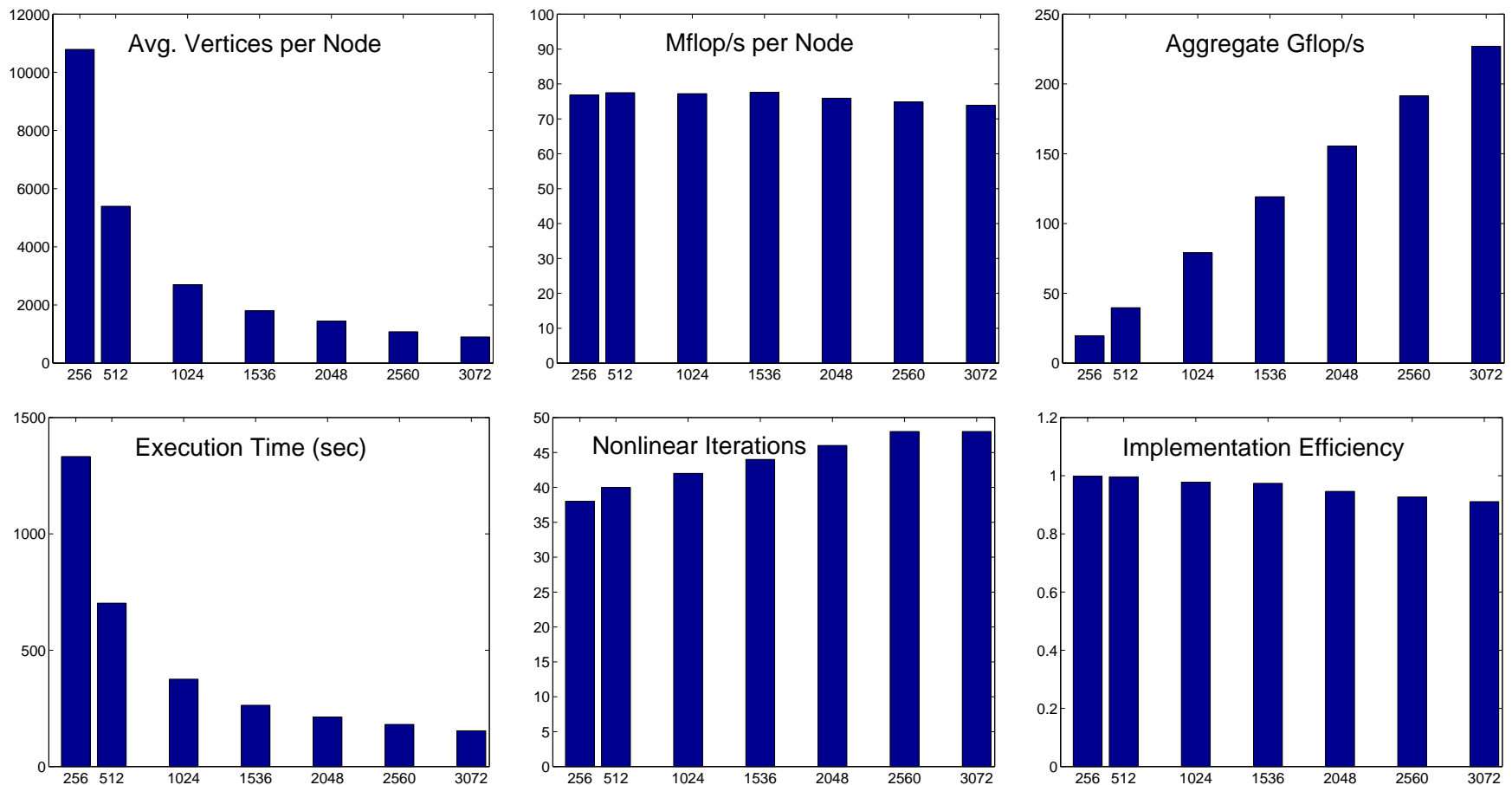
- Years of two-sided (from architecture up, from applications down) algorithms research has put us in a position to solve implicit PDE problems reasonably scalably, with “Newton-Krylov-Schwarz” iterative methods.
- Effective parallel implicit solver for large-scale nonlinear problems derived from PDEs: aerodynamics, radiation transport, porous media, semiconductors, geophysics, astrophysics
- Implemented in parallel matrix-free object-oriented framework, including both FD and AD distributed matvecs, in PETSc (from Argonne)
- Applied to unstructured computational aerodynamics problem on 6144 processors of ASCI Red for a 1999 Gordon Bell Prize; see Anderson, Gropp, Kaushik, Keyes & Smith (1999)
- PDEs are *equality constraints* on the state variables in many optimization problems; hardly “auxiliary”, the PDE system may contain a million or more degrees of freedom.
- Optimization is easily incorporated into a Newton-like parallel PDE framework that accommodates substructuring.
- *Editorial: A PDE solver not part of an optimization framework is probably far short of what the client really wants.*

Example of NKS on Aerodynamics Problem



Example of NKS on Aerodynamics Problem

Euler flow on a tetrahedral grid of 2,761,744 vertices, based on KMeTiS-PETSc implementation of NASA code FUN3D run on up to 3072 nodes of ASCI Red at 0.2–0.3 Tflop/s



Recent References on ΨNKS Methods

(on-line with others at <http://www.math.odu.edu/~keyes>)

- *Globalized Newton-Krylov-Schwarz Algorithms and Software for Parallel Implicit CFD* (with Gropp, Melnes & Tidiri), 2000, Int. J. High Performance Computer Applications **14**:102–136.
 - software focus, overview
 - 3D structured-grid transonic Euler example
- *On the Interaction of Architecture and Algorithm in the Domain-Based Parallelization of an Unstructured Grid Incompressible Flow Code* (with Kaushik & Smith), 1998, in “Proc. of the 10th Intl. Conf. on Domain Decomposition Methods”, AMS, pp. 311–319.
 - HPC focus
 - 3D unstructured-grid incompressible Euler example
- *Convergence Analysis of Pseudo-Transient Continuation* (with Kelley), 1998, SIAM J. Num. Anal. **35**:508–523.
 - theory focus
 - 2D unstructured-grid subsonic Euler example
- *Parallel Newton-Krylov-Schwarz Algorithms for the Transonic Full Potential Equation* (with Cai, Gropp, Melvin & Young), 1998, SIAM J. Sci. Comp. **19**:246–265.
 - application focus
 - 2D structured-grid transonic full potential example
- *Newton-Krylov-Schwarz Methods for Aerodynamics Problems: Compressible and Incompressible Flows on Unstructured Grids* (with Kaushik & Smith), 1998, “Proc. of the 11th Intl. Conf. on Domain Decomposition Methods”, pp. 513–520.
 - multi-platform comparisons
 - 3D unstructured-grid Euler example (Mach 0 to Mach 1.2)
- *How Scalable is Domain Decomposition in Practice?* 1998, “Proc. of the 11th Intl. Conf. on Domain Decomposition Methods”, pp. 286–297.
 - parallel complexity focus
 - 3D unstructured-grid incompressible Euler example

Implications of NKS for Optimization

- Equality constrained optimization leads, through the Lagrangian formulation, to a multivariate nonlinear rootfinding problem for the gradient (first-order necessary conditions)
- Canonical framework: choose m design variables u to minimize objective function, $c(u, x)$, subject to n state constraints, $h(u, x) = 0$, where x is the vector of state variables
- Lagrange framework: find stationary point of the Lagrangian

$$\mathcal{L}(x, u, \lambda) \equiv c(x, u) + \lambda^T h(x, u)$$

- Natural “outer” partitioning: controls are often of lower dimension than states and multipliers, suggesting *Schur*-complement preconditioning
- Natural “inner” partitioning: states and their multipliers are of high dimension and corresponding matrix blocks are sparse, suggesting *Schwarz*-like domain decomposition

Reduced or Full Systems?

- As in domain decomposition, choice to be made between:
 - exact elimination of the states and multipliers by satisfying constraint feasibility at every step (reduced system), or
 - making progress in all variables, possibly violating constraints until convergence (full system)
- Advantage of the former:
 - existence of quality, robust “black box” optimization software
- Advantages of the latter:
 - reuse of quality, efficient parallel PDE software
 - employment of inexact solves while retaining “exact” Jacobian in outer iteration
 - ease of application of automatic differentiation software, without having to differentiate through subiterations

Examples of PDE-constrained Optimization

- **Design optimization** (esp. shape optimization): u parameterizes the domain of the PDE (e.g., lifting surface) and c is a cost-to-benefit ratio of forces, energy expenditures, etc. Typically, m is small compared to n , and does not scale directly with it. But m may still be several hundred.
- **Optimal control**: u parameterizes a continuous control function acting on the surface of the domain, and c is the norm of the difference between desired and actual responses of the system. Typically, $m \propto n^{2/3}$.
- **Parameter identification / data assimilation**: u parameterizes an unknown continuous constitutive or forcing function defined throughout the domain, and c is the norm of the difference between measurements and simulation results. Typically, $m \propto n$.

Optimality Conditions

We look for saddle points of the Lagrangian:

$$\frac{\partial \mathcal{L}}{\partial x} \equiv \frac{\partial c}{\partial x} + \lambda^T \frac{\partial h}{\partial x} = 0$$

$$\frac{\partial \mathcal{L}}{\partial u} \equiv \frac{\partial c}{\partial u} + \lambda^T \frac{\partial h}{\partial u} = 0$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} \equiv h = 0$$

by finding a correction,

$$\begin{pmatrix} \delta x \\ \delta u \\ \delta \lambda \end{pmatrix} \quad \text{to the iterate} \quad \begin{pmatrix} x \\ u \\ \lambda \end{pmatrix}$$

using Newton's method.

Optimality Conditions

With subscript notation for partial derivatives, the Newton (Karush-Kuhn-Tucker) equations are:

$$\begin{bmatrix} (c_{,xx} + \lambda^T h_{,xx}) & (c_{,xu} + \lambda^T h_{,xu}) & h_{,x}^T \\ (c_{,ux} + \lambda^T h_{,ux}) & (c_{,uu} + \lambda^T h_{,uu}) & h_{,u}^T \\ h_{,x} & h_{,u} & 0 \end{bmatrix} \begin{pmatrix} \delta x \\ \delta u \\ \delta \lambda \end{pmatrix} = - \begin{pmatrix} c_{,x} + \lambda^T h_{,x} \\ c_{,u} + \lambda^T h_{,u} \\ h \end{pmatrix}$$

or

$$\begin{bmatrix} W_{xx} & W_{ux}^T & J_x^T \\ W_{ux} & W_{uu} & J_u^T \\ J_x & J_u & 0 \end{bmatrix} \begin{pmatrix} \delta x \\ \delta u \\ \lambda_+ \end{pmatrix} = - \begin{pmatrix} g_x \\ g_u \\ h \end{pmatrix},$$

where $W_{ab} \equiv \frac{\partial^2 c}{\partial a \partial b} + \lambda^T \frac{\partial^2 h}{\partial a \partial b}$, $J_{ua} \equiv \frac{\partial h}{\partial a}$, and $g_a = \frac{\partial c}{\partial a}$, for $a, b \in \{x, u\}$, and where $\lambda_+ = \lambda + \delta \lambda$.

Newton Reduced SQP

Design Step (Schur complement for middle blockrow):

$$H \delta u = f ,$$

where H and f are the reduced Hessian and gradient, resp.:

$$\begin{aligned} H &\equiv W_{uu} - J_u^T J_x^{-T} W_{ux}^T + (J_u^T J_x^{-T} W_{xx} - W_{ux}) J_x^{-1} J_u \\ f &\equiv -g_u + J_u^T J_x^{-T} g_x - (J_u^T J_x^{-T} W_{xx} - W_{ux}) J_x^{-1} h \end{aligned}$$

State Step (last blockrow):

$$J_x \delta x = -h - J_u \delta u$$

Adjoint Step (first blockrow):

$$J_x^T \lambda_+ = -g_x - W_{xx} \delta x - W_{ux}^T \delta u$$

In each overall iteration:

- must form and solve with H
- must solve with J_x and J_x^T (negligible compared with the cost of forming H)

Number of overall iterations is few (asymptotically independent of m)

Cost of forming H at each design iteration is m solutions with J_x — potentially concurrent, but prohibitive

Quasi-Newton Reduced SQP

Design Step (severe approximation for middle blockrow):

$$Q \delta u = -g_u + J_u^T J_x^{-T} g_x ,$$

where Q is a quasi-Newton approximation to the reduced Hessian

State Step (last blockrow):

$$J_x \delta x = -h - J_u \delta u$$

Adjoint Step (approximation to first blockrow):

$$J_x^T \lambda_+ = -g_x$$

In each overall iteration:

- must perform low rank update on Q or its inverse
- must solve with J_x and J_x^T

Number of overall iterations is many. Since BFGS is equivalent to unpreconditioned CG for quadratic objective functions, $\mathcal{O}(m^p)$ sequential cycles ($p > 0$, $p \approx \frac{1}{2}$) may be anticipated

Hence, RSQP is not scalable in the number of design variables, and no ready form of parallelism can address this

Proposed Full System Approach

- Conventional RSQP methods apply a (quasi-)Newton method to the optimality conditions:
 - solving an approximate $m \times m$ system to update u ,
 - updating x and λ consistently (to eliminate them),
 - iterating
- Exact linearized analyses for updates to x and λ appear in the inner loop
- Consider replacing the exact elimination steps of RSQP with preconditioning steps in an outer loop
- Earlier proposed in this or closely related context by Batterman & Heinkenschloss (1996), Biros & Ghattas (1998)
- Algebraically identical to handling of interface constraints in Schur-complement-based domain decomposition by Keyes & Gropp (1987)

Full Space Lagrange-NKS Method

Apply KS directly to the $(2n + m) \times (2n + m)$ KKT system:

$$\begin{bmatrix} W_{xx} & W_{ux}^T & J_x^T \\ W_{ux} & W_{uu} & J_u^T \\ J_x & J_u & 0 \end{bmatrix} \begin{pmatrix} \delta x \\ \delta u \\ \lambda_+ \end{pmatrix} = - \begin{pmatrix} g_x \\ g_u \\ h \end{pmatrix}$$

- Need action of the full matrix on the full space vector
- Need a good full system preconditioner, for algorithmic scalability
- One Newton SQP iteration is a perfect preconditioner — a block factored solver, based on forming the reduced Hessian of the Lagrangian H , but, of course, far too expensive
- Backing off wherever things get impractical in the preconditioner generates a family of methods

Example of Parameter Identification in Radiation Transport

- General governing equation

$$\frac{\partial T}{\partial t} = \nabla \cdot (\beta(x) T^\alpha \nabla T)$$

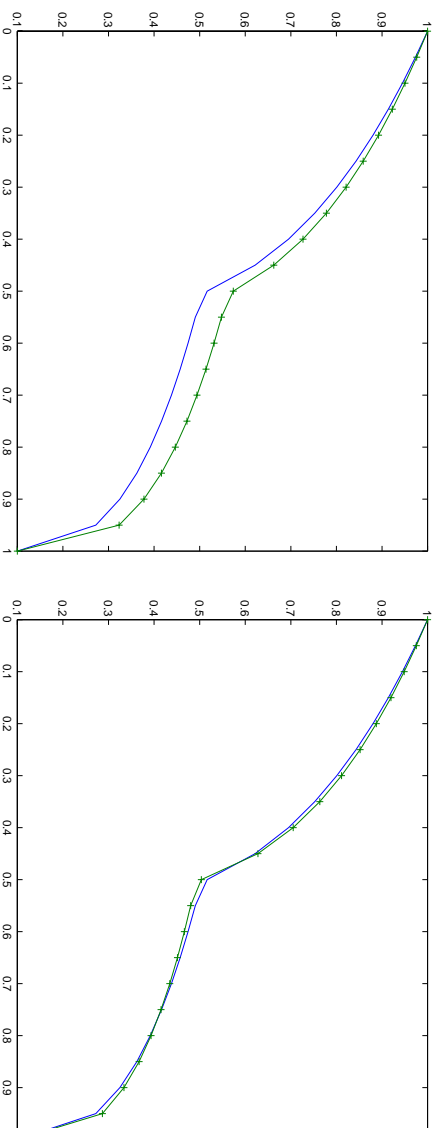
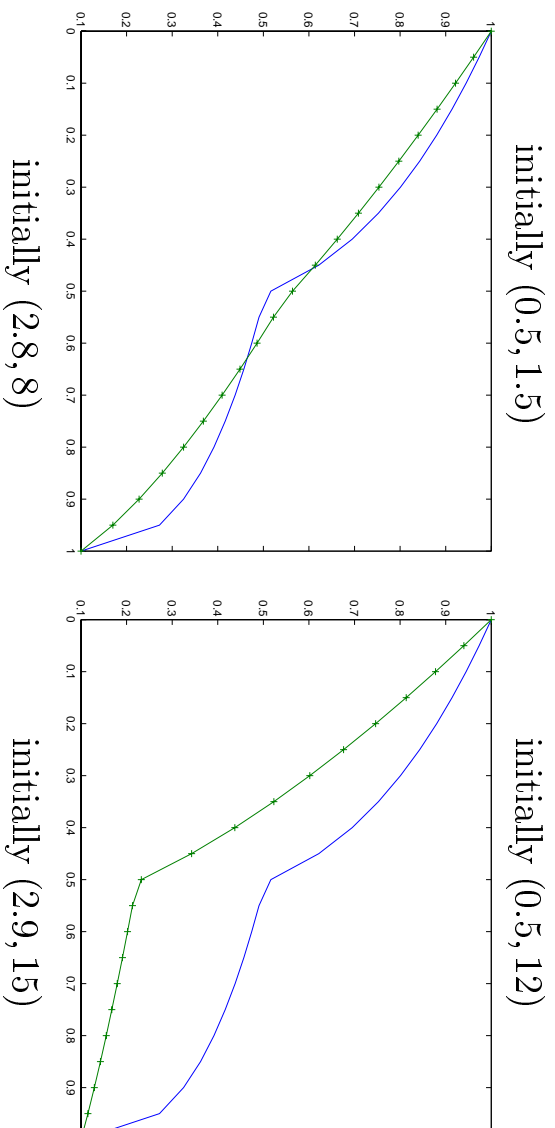
- Initial conditions: impulsive boundary heating for T (Marshak wave)
- Use 1D steady example with jump in material properties
- Cost function is simple temperature matching based on a given α , $\beta(x)$ profile, \bar{T} : $c(u, x) = \frac{1}{2} \|T(x) - \bar{T}(x)\|^2$, where
 - “Brisk-Spitzer” $\alpha = 2.5$;
 - $\beta(x) = 1, 0 \leq x \leq 0.5$; $\beta(x) = 10, 0.5 < x \leq 1.0$
 - more generally, $\bar{T}(x)$ is a desired or experimental profile

Implementation in Matlab and ADMMAT

- ADMMAT (Verma & Coleman) is an automatic differentiation framework for Matlab, based on operator overloading
- After supplying an m-file for the cost function and constraint functions, all gradients and Jacobians and Hessians (as well as their transposes and their contracted action on vectors) are computed without further user effort
- Preconditioner is RSQP block factorization, except that reduced Hessian is replaced with cost function Hessian alone, sparing sensitivity computations in a preconditioner
- Reduced Hessian should be replaced with quasi-Newton reduced Hessian in the future
- Simple Newton method without robustification of any kind

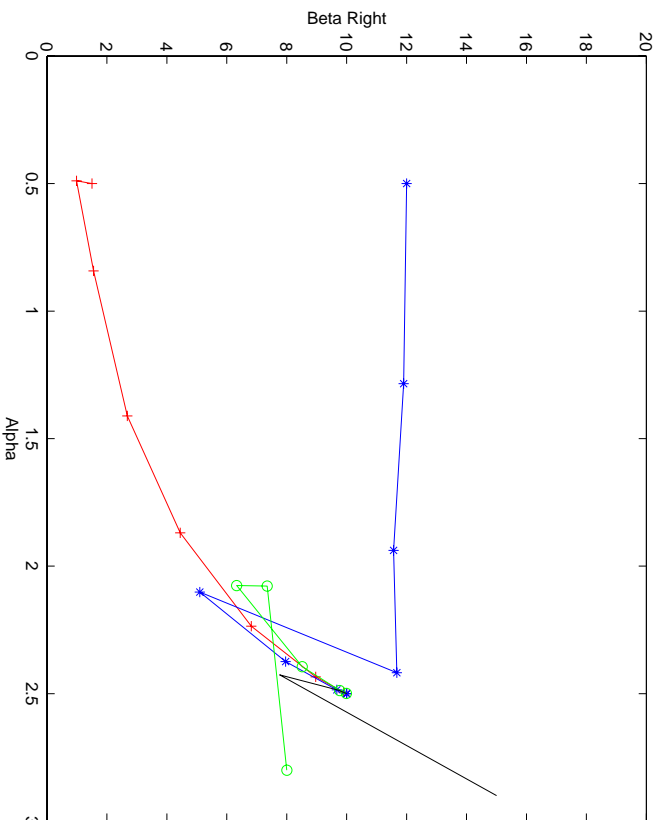
LNKS for Radiation Diffusion Example

$T(x)$ at initial (green) and final (blue) (α, β_R)

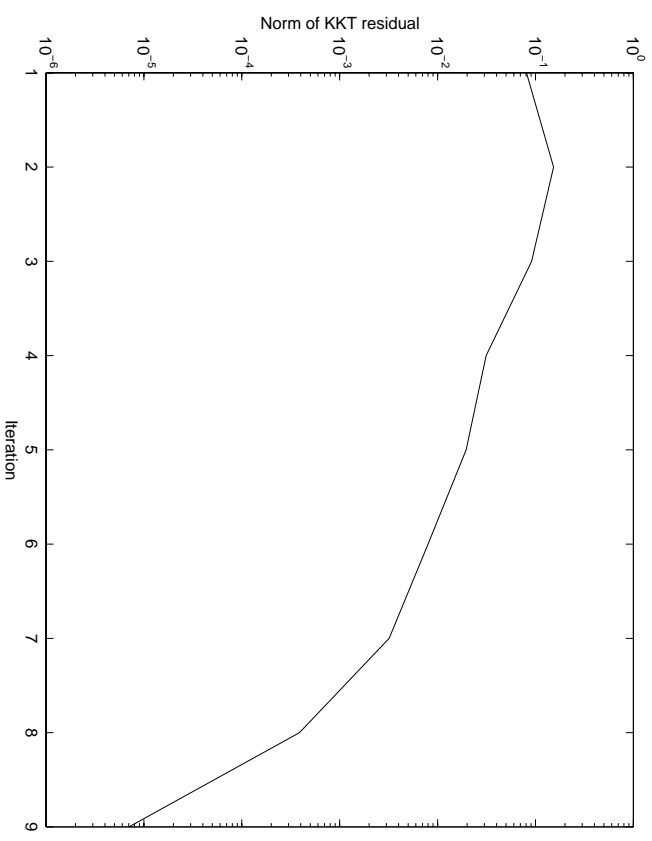


LNKS Convergence History

(α, β_R) trajectories from four cases (all converging to $(2.5, 10)$)



KKT Norm Convergence History for the case $(0.5, 1.5)$



Preconditioning the Full System

The shopping list of matrix actions in preconditioning is:

- W_{xx} , W_{uu} , W_{ux} , and W_{ux}^T
- J_u and J_u^T
- J_x^{-1} and J_x^{-T}
- H^{-1}

The first six (together with J_x and J_x^T) are also needed in the full system matrix-vector multiply. We require “working accuracy” comparable to the state of the art in numerical differentiation.

The accurate action of the last three is required in RSQP, but *not* in the full system preconditioner. We use approximate factorizations of lower quality approximations, including possibly just W_{uu} for H , or a traditional quasi-Newton rank-updated approximation to the inverse.

Jacobian and Hessian Arithmetic Complexity

We estimate the complexity of applying each Jacobian block to a vector, assuming only that $h(x, u)$ is available in subroutine form and that all differentiated blocks are from AD tools, such as ADIC. We estimate the complexity of applying each Hessian block to a vector. Assume that $h(x, u)$ and $c(x, u)$ are available and that all differentiated blocks are results of AD tools. We assume that J_x is needed, element-by-element, in order to factor it; hence, J_x^T is also available. Define:

- C_h , the cost of evaluating h
- p_x , 1 + the chromatic number of $J_x \equiv h_{,x}$
- p_u , 1 + the chromatic number of $J_u \equiv h_{,u}$
- C_c , the cost of evaluating c
- q , 1 + number of nonzero rows in c''
- r , implementation-dependent constant, approx. 3–50

Object	Cost: forward mode	Cost: fastest (hybrid) mode
J_x, J_x^T	$p_x C_h$	$p_x C_h$
$J_u v$	$2C_h$	$2C_h$
$J_u^T v$	$p_u C_h$	$r C_h$
$W_{xx} v, W_{ux}^T v$	$p_x C_h + q C_c$	$r(C_h + C_c)$
$W_{uu} v, W_{ux} v$	$p_u C_h + q C_c$	$r(C_h + C_c)$

Overall Arithmetic Complexity

For the inverse blocks, we need only low quality approximations, and low-fill (or low-rank correction inverse updates) of the square systems:

- J_x^{-1} and J_x^{-T}
- H^{-1}

Complexity is only linear in subsystem dimensions n and m . Summarizing, all operations required to apply the full system matrix-vector product and its preconditioner are at worst linear in n or m , with coefficients that depend upon:

- chromatic numbers (affected by stencil connectivity and inter-component coupling of the PDE, and by separability structure of the objective function)
- implementation efficiency of AD tools

Lagrange-Newton-Krylov-Schur-Schwarz: A Parallel Optimizer for BVP-constrained Problems



Lagrange

optimization
formulation



Newton

nonlinear
accelerator



Krylov

linear
accelerator



Schur

subspace
preconditioner



Schwarz

subdomain
preconditioner

Remarks on Full Space Lagrange-Newton Method

- As with any Newton method, globalization strategies are important:
 - parameter continuation (physical and algorithmic)
 - mesh sequencing and multilevel iteration (for the PDE subsystem, at least; probably controls, too)
 - discretization order progression
 - model fidelity progression
- KKT system is a preconditioning challenge, but an exact factored preconditioner is known, and departures of preconditioned eigenvalues from unity can be quantified with comparisons of original blocks with blockwise substitutions in inexact models and solves (see, e.g., E. Sachs et al.)
- Orders of magnitude of savings may be available by converging the state variables and the design variables within the same outer iterative process, rather than a conventional SQP process that exactly satisfies the “auxiliary” state constraints
- With the extra work of forming Jacobian transposes and Hessian blocks, but no extra work in Jacobian preconditioning, any parallel analysis code may be converted into a (limited) parallel optimization code — and automatic differentiation tools will shortly make this relatively painless

Future Prospects

- Increased penetration of object-oriented coding practices will make it easier to reuse subspace solvers as preconditioners in a tightly-coupled Newton process.
- Automatic differentiation tools will remove the burden of coding the KKT blocks that are “missing” from a parallel solver, as well as some that now require hand coding for the solver.
- Mathematical challenges and practical importance will attract more numerical analysts to the preconditioning and globalization problems of large-scale PDE-constrained optimization.

See <http://www.math.odu.edu/~keyes> for papers, talks, and workshop summaries