Matrix-free Methods for PDE-constrained Optimization

Lagrange-Newton-Krylov-Schur-Schwarz
Perspective

Editorial: A PDE solver not part of an optimization framework is probably far short of what the client really wants.

- A PDE solver incorporated into a Newton-like parallel PDE framework is probably what clients want.

- Work that accommodates substructuring optimization is easily incorporated into a Newton-like parallel PDE framework.

- Implicit problems, hardly "auxiliary," the PDE system may contain a million or more degrees of freedom.

- PDEs are equality constraints on the state variables in many optimization problems; hardly "auxiliary.

- Applied to unstructured computational aeroacoustics problem on 64 processors of ASCI Red for a 1999 Gordon Bell Prize; see Anderson, Gropp, Keyes, Keyes & Smith (1999).

- Effective parallel implicit solver for large-scale nonlinear aerodynamics problems, including both PD and AD

- Effective parallel implicit solver, with "Newton-Krylov-Schwarz" iterative methods.

- 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.

- "Have parallel PDE solver, will optimize."
Example of NKS on Aeronautics Problem
Example of NKS on Aerodynamics Problem

Euler flow on a tetrahedral grid of 2,761,744 vertices, based on KMTeTiS-PETSc implementation of NASA code FUN3D run on up to 3072 nodes of ASCI Red at 0.2–0.3 Tflop/s
Globalized Newton-Krylov-Schwarz Methods at
http://www.math.odu.edu/~keyes

Recent References on NKS Methods

- Globalized Newton-Krylov-Schwarz Methods and Software
- Provided in Applications 14:102-136.


AMS, pp. 311-319.


- 3D unstructured-grid incompressible Euler example
- 3D unstructured-grid incompressible Euler example
- Parallel complexity focus
- 3D unstructured-grid incompressible Euler example


- Parallel complexity focus
- 3D unstructured-grid incompressible Euler example

Implications of NKS for Optimization

Equality constrained optimization leads, through the Lagrangian formulation, to a multiobjective nonlinear optimization problem.

- Canonical framework: choose an objective function, constraints, and state variables.

\[(n', x) \gamma + (n', x) c = (n', x) J\]

- Lagrange framework: find stationary point of the Lagrangian

\[\gamma = 0, \text{ where } x \text{ is the vector of state variables,}\]

\[\text{subject to } n \text{ state constraints,}\]

- Natural inner partitioning: states and multipliers are of high dimension and corresponding matrix blocks are sparse, suggesting Schur-complement preconditioning.

- Natural outer partitioning: controls are often of lower dimension than states and multipliers, suggesting Schwarz-like domain decomposition-like domain decomposition.

- Implications of NKS for Optimization

Equality constrained optimization leads, through the Lagrangian formulation, to a multiobjective nonlinear optimization problem.
Reduced or Full Systems?

As in domain decomposition, choice to be made between:

- exact elimination of the states and multipliers by satisfying constraints feasibly at every step (reduced system), or
- exact elimination of the states and multipliers by satisfying constraints feasibly in all variables, possibly violating constraints constraining feasibility at every step (full system),

Advantage of the former:
- existence of quality robust "black box" optimization software

Advantage of the latter:
- reuse of quality efficient parallel PDE software
- employment of inexact solves while retaining "exact" Jacobian in outer iteration
- reuse of quality, efficient parallel PDE software

As in outer iteration, 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 constraining feasibility

- existence of quality, robust "black box" optimization software

- reuse of quality efficient parallel PDE software
- employment of inexact solves while retaining "exact" Jacobian in outer iteration

- reuse of quality efficient parallel PDE software

- existence of quality robust "black box" optimization software
Examples of PDE-constrained Optimization

Design optimization:

design parameterizes a continuous constitutive or forcing function defined throughout the domain, and \( c \) is the norm of the difference between desired and actual responses of the system.

Parameter identification/data assimilation:

\[ n \times n, \] which may still be several hundred. But in many cases, \( m \) is small compared to \( n \), and does not scale directly with \( n \). Typically, \( m \) is still several hundred.

Optimal control:

Typically, \( m \approx n^{2/3} \).

Optimal parameter identification/n parameterization/data assimilation:

Typically, \( m \approx n^{2/3} \).
Optimality Conditions

Using Newton’s method

\[
\left( \begin{array}{c}
\chi \\
n \\
x
\end{array} \right)
\]

to the iterate

\[
0 = \eta \equiv \frac{\chi}{J} \epsilon
\]

\[
0 = \frac{n}{\eta} J \chi + \frac{n}{\epsilon} \equiv \frac{n}{J} \epsilon
\]

\[
0 = \frac{x}{\eta} J \chi + \frac{x}{\epsilon} \equiv \frac{x}{J} \epsilon
\]

By finding a correction,

We look for saddle points of the Lagrangian:
Optimality Conditions

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

\[
\begin{align*}
\forall q + \gamma &= +\gamma \quad \text{and where} \quad \{n', x\} \not\in q, \\
&
\end{align*}
\]

\[
\begin{pmatrix}
\eta \\
nq \\
xq
\end{pmatrix}
- \begin{pmatrix}
+\gamma \\
nq \\
xq
\end{pmatrix}
= \begin{pmatrix}
0 & n_{f} & x_{f} \\
n_{f} & n_{M} & x_{M} \\
x_{f} & x_{M} & x_{M}
\end{pmatrix}
\]

or

\[
\begin{pmatrix}
\eta \\
n'q \\
xq
\end{pmatrix}
- \begin{pmatrix}
\gamma q \\
n'q \\
xq
\end{pmatrix}
= \begin{pmatrix}
0 & n'_{q} & x'_{q} \\
n'_{q} & (nn'_{q} \gamma + nn'_{q}) & (xn'_{q}\gamma + xn'_{q}) \\
x'_{q} & (nx'_{q}\gamma + nx'_{q}) & (xx'_{q}\gamma + xx'_{q})
\end{pmatrix}
\]

Kuhn-Tucker equations are:

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

\[
\begin{align*}
\forall q + \gamma &= +\gamma \\
&
\end{align*}
\]

or

\[
\begin{pmatrix}
\eta \\
n'q \\
xq
\end{pmatrix}
- \begin{pmatrix}
\gamma q \\
n'q \\
xq
\end{pmatrix}
= \begin{pmatrix}
0 & n'_{q} & x'_{q} \\
n'_{q} & (nn'_{q} \gamma + nn'_{q}) & (xn'_{q}\gamma + xn'_{q}) \\
x'_{q} & (nx'_{q}\gamma + nx'_{q}) & (xx'_{q}\gamma + xx'_{q})
\end{pmatrix}
\]
Cost of forming at each design iteration is $n \phi$ solutions with $H$, potentially concurrent, but prohibitive.

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

- $H$ must form and solve with $x_f$.
- $x_f$ must form and solve with $H$.

In each overall iteration:

\[
\phi \begin{bmatrix} x_n \end{bmatrix} ^T M - x \begin{bmatrix} x_n \end{bmatrix} x M - x \begin{bmatrix} b \end{bmatrix} = + \chi \begin{bmatrix} x \end{bmatrix} f
\]

\[ n \phi \begin{bmatrix} n \end{bmatrix} f - \eta = x \begin{bmatrix} \phi \end{bmatrix} ^T \]

Adjacent Step (first blockrow):

\[ n \phi \begin{bmatrix} n \end{bmatrix} f - \eta = x \begin{bmatrix} \phi \end{bmatrix} f \]

State Step (last blockrow):

\[ n \phi \begin{bmatrix} n \end{bmatrix} f - \eta = x \begin{bmatrix} \phi \end{bmatrix} f \]

\[ n \phi \begin{bmatrix} n \end{bmatrix} f - \eta = x \begin{bmatrix} \phi \end{bmatrix} f \]

Design Step (Schur complement for middle blockrow):

\[ f = n \phi \begin{bmatrix} H \end{bmatrix} \]
Design Step (severe approximation for middle blockrow):

Quasi-Newton Reduced SQP

State Step (last blockrow):

adjoint Step (approximation to first blockrow):

where is a quasi-Newton approximation to the reduced Hessian

Subject Step (last blockrow):

Adjoint Step (approximation to first blockrow):

In each overall iteration:

must perform low rank update on or its inverse

must solve with and

where (may be antiquated objective functions) and sequential cycles (equivalent cycles) of BFGS is equivalent to unpreconditioned CG for quadratic

Hence, RSQP is not scalable in the number of design variables, and no ready form of parallelism can be anticipated.

number of overall iterations is many. Since BFGS is equivalent to unpreconditioned CG for quadratic

First SIAM Conference on Computational Science \\& Engineering 2/2 September 2000
Proposed Full System Approach

Conventional RSQP methods apply a quasi-Newton method to the optimality conditions:

\[ \text{solve an approximate } m \times m \text{ system to update } n, \]

- updating \( x \) and \( \chi \) consistently (to eliminate them)
- solving \( \chi \) consistently
- applying \( \text{a quasi-Newton method} \)


Consider replacing the exact elimination steps with preconditioning steps in an outer loop.

\[ \text{inner loop} \]

\[ \text{inner loop} \]

Interacting

\[ \text{exact linearized analyses for updates to } x \text{ and } \chi \text{ appear in the} \]

\[ \text{proposed full system approach} \]
Full Space Lagrange-NKS Method

Backings off wherever things get impractical in the preconditioned Lagrangian, but, of course, far too expensive. One Newton SQP iteration is a perfect preconditioner — a block

Need a good full system preconditioner, for algorithmic scalability.

Need action of the full matrix on the full space vector.

\[
\begin{pmatrix}
  \eta \\
  n\partial \\
  x\partial
\end{pmatrix}
- =
\begin{pmatrix}
  +\chi \\
  n\partial \\
  x\partial
\end{pmatrix}
\begin{bmatrix}
  0 & nI & xI \\
  nI & nnM & xnM \\
  xI & xnM & xxM
\end{bmatrix}
\]

Apply KS directly to the (KKT system): 

\[(m + u) (2m + u) \times (m + u)\]
Example of Parameter Identification in Radiation Transport

General governing equation

Initial conditions: Impulsive boundary heating for (Marshak)

Cost function is simple temperature matching based on a given

More generally, is a desired or experimental profile

where

\[ \| (x) - (x) \|_2 = (x, n) \cdot (x) \] the profile, (x)

Use 1D steady example with jump in material properties

First SIAM Conference on Computational Science & Engineering 2000

\[ \Delta = \frac{\mathcal{L}}{\mathcal{L}} \] General governing equation

in Radiation Transport
Implementation in Matlab and ADMAT

Simple Newton method without robustification of any kind

Hessian in the future

Reduced Hessian should be replaced with quasi-Newton reduced

Sensitivity computations in a preconditioner

Hessian is replaced with cost function Hessian alone, sparing

Preconditioner is RSQP block factorization, except that reduced

Preconditioned without further user effort

computed without further robustification of any kind

as their transposes and their contracted action on vectors are

After supplying an m-file for the cost function and constraint

framework for Matlab, based on operator overloading

(Verma & Coleman) is an automatic differentiation

Implementatin in Matlab and ADIAT
LNKS Convergence History

For the case \((0, 0.5, 1.5)\)

KKT Norm Convergence History

LNKS Convergence History

cases (all converging to \((2.5, 1.0)\))

\((0.5, 1.5)\) trajectories from four
Preconditioning the Full System

The shopping list of matrix actions in preconditioning is:

Preconditioning the Full System

\[ H^{-1} \]
\[ x \]
\[ x \]
\[ x \]
\[ x \]
\[ x \]
\[ x \]

For a traditional quasi-Newton rank-updated approximation \( H \), or a traditional quasi-Newton rank-updated approximation \( H \), or a traditional quasi-Newton rank-updated approximation \( H \), or a traditional quasi-Newton rank-updated approximation \( H \), we require the last three is required in RSQP, but not.

The accurate action of the first six (together with \( x \) and \( x \)) 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 first six (together with \( x \) and \( x \)) are also needed in the full system matrix-vector multiply. We require working accuracy comparable to the state of the art in numerical differentiation.
We estimate the complexity of applying each Jacobian block to a vector, assuming only that $h(x)\in\mathbb{R}$. We estimate the complexity of applying each Hessian block to a vector, assuming that $J(x)$ and $c(x)$ are available and that all differentiable blocks are results of AD tools. We estimate the complexity of applying each Hessian block to a vector, assuming that $J(x)$ and $c(x)$ are available and that all differentiable blocks are results of AD tools. We estimate the complexity of applying each Jacobian block to a vector, assuming only that $h(x)\in\mathbb{R}$.
Overall Arithmetic Complexity

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

\[ J^{-1} H J \]

\[ L \cdot I^{-1} \]

\[ J^{-1} \]

Systems:

\[ x \]

Complexity is only linear in subsystem dimensions \( n \) and \( m \). Sum:

Component coupling of the PDE, and by separability structure

chromatic numbers (affected by stencil connectivity and inter-

\[ J \]

and \[ J \]

Coefficient coupling of the PDE, and by separability structure

\[ J \]

with coefficients that depend upon:

\[ J \]

Operations required to apply the full system matrix-

\[ J \]

Summarizing, all operations required to apply the full system matrix-

\[ J \]

Component coupling of the PDE, and by separability structure

\[ J \]

implementation efficiency of AD tools

( of the objective function

\[ J \]

Implementation efficiency of AD tools

\[ J \]
A Parallel Optimizer for BVP-constrained Problems

Lagrange–Newton–Krylov–Schur–Schwarz

Lagrange

Newton

Krylov

Schur

Schwarz

Schwarz
RemarksonFullSpaceLagrange/-NewtonMethod

As with any Newton method, globalization strategies are important:

- With the extra work of forming Jacobian transposes and Hessian blocks, but
- With globalization into a (limitted) parallel optimization code — and automatic
  parameter continuation, and parallel analyses code may
  no extra work in Jacobian preconditioning, any parallel analysis code may

  mesh sequencing and multilevel iteration (for the PDE subsystem, at least;)
  mesh sequencing and multilevel iteration (physical and algorithmic)
  physical and algorithmic
  probably controls, too

KKTsystem is a preconditioning challenge, but an exact factored preconditioner

Orders of magnitude of savings may be available by converging the state
  variables and the design variables within the same outer iterative process,

state constraints

rather than a conventional SQP process that exactly satisfies the „auxiliary“

Orders of magnitude of savings (see, e.g., E. Sachs et al.)

tilations in exact models and solves (see, e.g., E. Sachs et al.) can be quantified with comparisons of original blocks with blockwise subsp.

with departures of preconditioned eigenvalues from unity

diagonally is known, and departures of preconditioned eigenvalues from unity

model identity preconditioning

- discretization order preconditioning

- probably controls, too

As with any Newton method, globalization strategies are important:

Remarks on Full Space Taylor-Newton Method
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. Mathematical challenges and practical importance will attract more numerical analysts to the preconditioning and globalization of large-scale PDE-constrained optimization problems. As some that now require hand coding for the solver, the KKT blocks that are “missing” from a parallel solver, as well as automatic differentiation tools will remove the burden of coding.

Automatic differentiation tools will remove the burden of coding.

Increased penetration of object-oriented coding practices will

Mathematical challenges and practical importance will attract more numerical analysts to the preconditioning and globalization of large-scale PDE-constrained optimization problems.

See http://www.math.odu.edu/~keyes for papers, talks, and work.

First SIAM Conference on Computational Science & Engineering 2000