

# Matrix-free methods in PDE-constrained Optimization

Paul D. Hovland

*Math. & Comp. Sci. Div., Argonne Nat. Lab.*

David E. Keyes

*Math. & Stat. Dept., Old Dominion University  
ISCR, Lawrence Livermore Nat. Lab.  
ICASE, NASA Langley Re. Ctr.*

Lois C. McInnes

*Math. & Comp. Sci. Div., Argonne Nat. Lab.  
Widodo Samyono*

*Math. & Stat. Dept., Old Dominion University*

## Acknowledgments

- Algorithmics
  - George Biros, Xiao-Chuan Cai, Omar Ghattas, David Young
- Applications and Software
  - Kyle Anderson, Satish Balay, Bill Gropp, Dinesh Kaushik, Barry Smith, Arun Verma
- Sponsors
  - DOE, NASA, NSF
- Computer facilities
  - DOE ASCI Alliance Program, Cray (Eagan, MN)

## 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.
- PDEs are *equality constraint* on the state variables in many optimization problems; hardly “auxiliary”, the PDE system may contain  $10^6$  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.

# Presentation Plan

- Survey of partitioned linear and nonlinear solvers
- Focus on Krylov-Schur, Krylov-Schwarz, Newton-Krylov-Schwarz
- A globalized parallel implicit rootfinder: Pseudo-transient NKS ( $\Psi$ NKS)
  - large-scale example: computational aerodynamics
- First-order optimality conditions of equality-constrained optimization using the Lagrangian
- A parallel optimization framework: Lagrange-Newton-Krylov-Schur/Schwarz (LNKS)
  - prototype parameter identification example: radiation transport
- Future Directions

# Extremely Informal Survey of Partitioned Solvers

- Linear
  - Schur (block Gaussian elimination)
  - Schwarz (block diagonal multisplitting)
- Nonlinear
  - Nonlinear Schur (nested elimination)
  - Nonlinear Schwarz (block diagonal nonlinear Picard)
  - Newton w/ partitioned nonlinear preconditioning
  - Newton w/ partitioned linear preconditioning of Jacobian

## Schur Methods for $Ax = f$

- For two nonoverlapping partitions:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

- form Schur complement and reduced RHS:

$$C \equiv A_{22} - A_{21}A_{11}^{-1}A_{12}, \quad g \equiv f_2 - A_{21}A_{11}^{-1}f_1$$

- solve for  $x_2$  in  $Cx_2 = g$  and backsolve for  $x_1$
- Domain decomposition application for PDEs:
  - $A_{11}$  is diagonal matrix of  $n$  subdomain interior blocks
  - $A_{22}$  is interface (subdomain separator) set
  - $n$ -fold concurrency in subdomain elimination
  - $\dim(A_{22}) \propto (\dim(A_{11}))^{d/d+1}$

## Krylov-Schur Methods for $Ax = f$

- Motivation:  $C$  is too expensive to form and store
  - costs  $\dim(A_{22})$  solves with  $A_{11}$Instead, apply Krylov acceleration to  $Cx_2 = g$ 
  - apply  $C$  to sequence of Krylov vectors
  - can solve in  $k$  steps,  $k \ll \dim(A_{22})$ , each requiring one solve with  $A_{11}$  to form action of  $C$  on Krylov vector
- For domain decomposition applications to elliptic problems:
  - $\text{cond}(C) \propto [\text{cond}(A)]^{1/2}$  and  $C$  inherits SPD property
  - given a good preconditioner  $M$  for  $C$  ( $M^{-1}C \approx I$ ), can build a good preconditioner  $B$  for  $A$  ( $B^{-1}A \approx I$ ) as

$$\begin{bmatrix} \tilde{A}_{11} & 0 \\ A_{21} & M \end{bmatrix} \begin{bmatrix} I & \tilde{A}_{11}^{-1} A_{12} \\ 0 & I \end{bmatrix}$$

# Notes on Krylov-Schur Methods

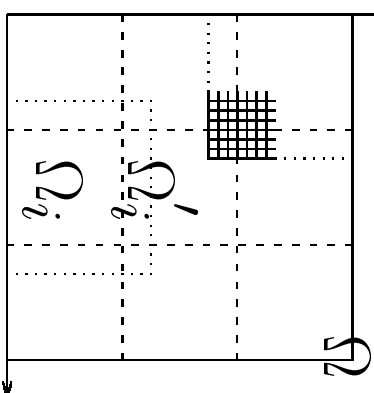
- Action of  $M^{-1}C$  requires exact solves with  $A_{11}$
- Action of  $B^{-1}A$  requires only inexact solves with  $A_{11}$ 
  - $x_1$  is retained to drive residual in full system
- When exact solves are used in  $B^{-1}$ , Krylov method on reduced and full systems yield identical iterates; see Gropp & K. (1986)
- When inexact solves with  $\tilde{A}_{11}$  are used in  $B^{-1}$ , vast savings relative to  $A_{11}$  usually overcome extra iterations
- Exact forward matrix-vector action of  $A_{11}$  still needed; in non-linear problems, where  $A$  is a Jacobian, this is obtainable without storing  $A_{11}$

## Schwarz Methods for $Ax = f$

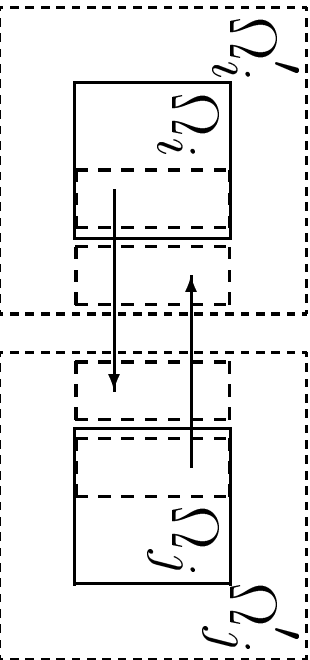
- Partition  $x$  into  $n$  subvectors, nonempty, possibly overlapping, whose union is all of the elements of  $x$
- Let Boolean rectangular matrix  $R_i$  extract the  $i^{\text{th}}$  subset of  $x$ :  
$$x_i = R_i x$$
- Let  $A_i = R_i A R_i^T$
- Let  $B^{-1} = z_i R_i^T A_i^{-1} R_i$
- Can iterate:  $x^{k+1} = (I - B^{-1} A) x^k + B^{-1} f$
- Since  $\rho(I - B^{-1} A)$  may be greater than unity in general, this additive splitting may not converge as a Richardson iteration
- Multiplicative Schwarz methods (Gauss-Seidel-like relative to the Jacobi-like additive) can be proved convergent when  $A$  derives from an elliptic PDE, under certain partitionings

# Additive Schwarz for $Ax = f$

- Form  $B^{-1} \approx A^{-1}$  out of (approximate) local solves on (possibly overlapping) subdomains



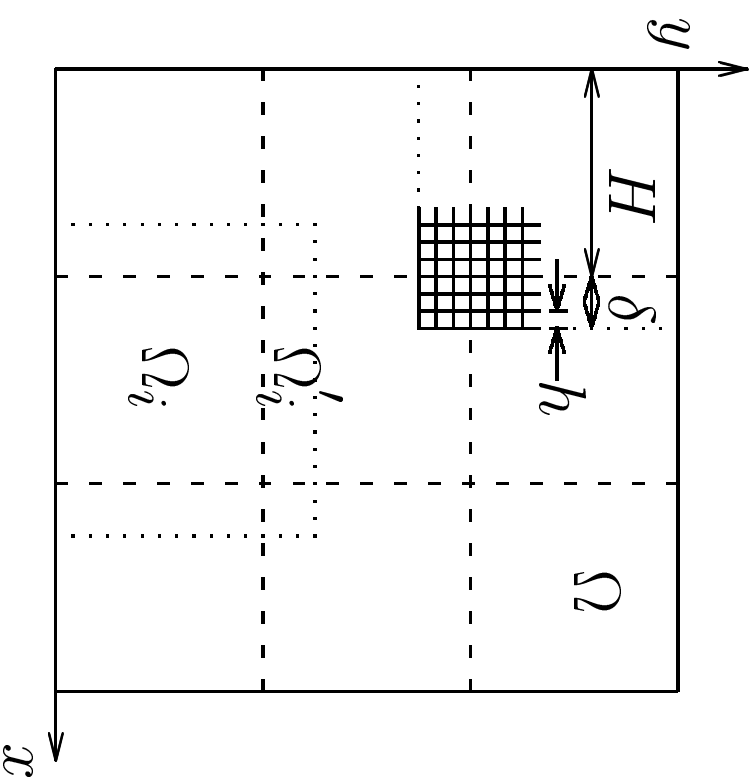
- $R_i$  and  $R_i^T$  are gather and scatter operations, mapping between a global vector and its  $i$ th subdomain support



$$A_i = R_i A R_i^T$$

$$B^{-1} = \sum_i R_i^T A_i^{-1} R_i$$

# Schematic of Overlapping Schwarz Parameters



Overlapping subdomains  $\Omega_i$ , and the scales of the mesh spacing ( $h$ ), the subdomain overlap ( $\delta$ ), and the subdomain diameter ( $H$ )

# Theoretical Condition Number Estimates

- $\kappa(B^{-1}A)$ , for self-adjoint positive-definite elliptic problems:

Point Jacobi  $\mathcal{O}(h^{-2})$

Domain-block Jacobi  $\mathcal{O}((hH)^{-1})$

1-level Additive Schwarz  $\mathcal{O}(H^{-2})$

2-level Additive Schwarz  $\mathcal{O}(1)$

- Schwarz estimates are for generous overlap  $\delta = \mathcal{O}(H)$ ; otherwise 2-level result is  $\mathcal{O}(1 + H/\delta)$
- Extensible to nonself-adjointness, indefiniteness, and inexact subdomain solvers
  - theory requires “sufficiently fine” coarse mesh,  $H$
  - practice is better than one has any right to expect
- Coarse space need *not* be nested in the fine space or in the decomposition into subdomains

## Krylov-Schwarz Methods for $Ax = f$

- Though  $\rho(I - B^{-1}A)$  may exceed unity,  $\sigma(B^{-1}A)$  is profoundly clustered, so Krylov methods should work well on

$$B^{-1}Ax = B^{-1}f$$

- When  $A$  derives from an elliptic operator and  $R_i$  is the characteristic function of unknowns in a subdomain, optimal convergence (independent of  $\dim(x)$  and the number of partitions) can be proved, with the addition of a coarse grid:

$$B^{-1} = R_0^T A_0^{-1} R_0 + \sum_{i>0} R_i^T A_i^{-1} R_i$$

## Convergence Estimates for Krylov-Schwarz

- Krylov-Schwarz methods typically converge in a number of iterations that scales as the square-root of the condition number of the Schwarz-preconditioned system
- For algorithmic scalability estimates, assume one subdomain per processor in a  $d$ -dimensional isotropic problem, where  $N = h^{-d}$  and  $P = H^{-d}$ , then iteration counts may be estimated as follows:

<i>Preconditioning</i>	$2D$	$3D$
Point Jacobi	$\mathcal{O}(N^{1/2})$	$\mathcal{O}(N^{1/3})$
Domain Jacobi	$\mathcal{O}((NP)^{1/4})$	$\mathcal{O}((NP)^{1/6})$
1-level Additive Schwarz	$\mathcal{O}(P^{1/2})$	$\mathcal{O}(P^{1/3})$
2-level Additive Schwarz	$\mathcal{O}(1)$	$\mathcal{O}(1)$

## Nonlinear Schur Methods for $F(x) = 0$

- For two nonoverlapping partitions,

$$\begin{cases} F_1(x_1, x_2) = 0 \\ F_2(x_1, x_2) = 0 \end{cases},$$

where  $\dim(F_i) = \dim(x_i)$  and where  $F_i$  is invertible for  $x_i$

- write  $x_1 = g(x_2)$  as formal solution of  $F_1(x_1, x_2) = 0$
- solve  $F_2(g(x_2), x_2) = 0$  for  $x_2$ 
  - \* for each  $x_2$  exactly solve inner system with  $F_1$
- Application to PDEs:
  - $F_1$  may be further partitionable into independent subsystems by domain decomposition, leaving  $F_2$  on the interface
  - $F_1$  may be less nonlinearly “stiff” than  $F_2$
  - See Young & K. (1996) for subsystem line search strategies

## Nonlinear Schwarz for $F(x) = 0$

- Partition  $x$  into  $n$  subvectors, nonempty, possibly overlapping, whose union is all of the elements of  $x$
- Let  $F_i(x_i, \bar{x}_i) = 0$  be a subset of  $F$  invertible for  $x_i$ , where  $\bar{x}_i$  is the complement of  $x_i$  in  $x$
- Iteratively solve systems  $F_i(x_i^{k+1}, \bar{x}_i^k) = 0$  for  $x_i^{k+1}$
- This is additive nonlinear Schwarz
- If the components of  $\bar{x}_i$  are updated as soon as available, an analogous multiplicative nonlinear Schwarz can be defined
- Convergence results are unimpressive (typically require strong conditions of monotonicity on  $F$  and its derivatives)

# Nonlinearly Schwarz-Preconditioned Newton

- Instead of  $F(x) = 0$ , solve  $\mathcal{F} \equiv G(F(x)) = 0$  with a matrix-free inexact Newton-Krylov method
- Practically, we need that  $G(F(u))$  be easily computable for any  $u$  and that  $G(F(u))'v$  be computable for any  $u, v$  pair
- We also require that if  $G(x) = 0$ , then  $x = 0$  to avoid introducing spurious roots, and that  $G \approx F^{-1}$  in some sense
- See Cai & K. (2000) for a  $G(\cdot)$  construction based on nonlinear Schwarz, applied to Navier-Stokes in a driven cavity
- Nonlinearly preconditioned Newton can be solved for Reynolds numbers far beyond the stagnation of Newton on  $F(x) = 0$
- Nonlinear Schwarz preconditioning can be combined with other robustification methods, such as continuation in time, parameters, or grid density; more comparative studies are needed

# Newton-Krylov-Schur/Schwarz Methods

- Let  $J \equiv \partial F / \partial x$
- Partitioning of  $x$  induces block structure on Jacobian
- Use global Newton method on  $F(x) = 0$ , using Krylov-Schur or Krylov-Schwarz linear iteration to solve Newton correction equations with  $J$
- As anticipated in the presentations of Schur and Schwarz above, we do not need any Jacobians explicitly
  - matrix-vector action of Jacobians is done with finite Frechét differencing or automatic differentiation
  - preconditioning of Jacobians is done with approximate operators, approximately solved

## Notes on Newton-Krylov-Schwarz

- Effective parallel implicit solver for large-scale nonlinear problems derived from PDEs (see, esp., Brown & collabs at LLNL and Knoll & collabs. at LANL)
- Applied to aerodynamics, radiation transport, porous media, semiconductors, geophysics, astrophysical MHD, etc.
- Commonly robustified with pseudo-transience or other continuation strategies
- 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, K. & Smith (1999)

## Split-Discretization Newton-Krylov-Schur/Schwarz

- In the Newton-Krylov formulation, base preconditioner  $B(u)$  on a more convenient discretization than is required for the nonlinear residual  $f$  itself; for example:
  - Jacobian with sacrificed coupling, for process concurrency (Schwarz), or segregation of physics (operator-splitting)
  - Jacobian of lower-order discretization, for fewer nonzeros and fewer colors in a minimal coloring
  - Jacobian of related discretization allowing “fast” solves
  - Jacobian with lagged values for any terms that are expensive or small or both
  - Jacobian stored in lower precision, for memory bandwidth

## Pseudo-transient Continuation

- Solves  $F(u) = 0$  through a series of modified problems

$$H_\ell(u) \equiv \frac{u - u_{\ell-1}}{\delta_\ell} + F(u) = 0, \quad \ell = 1, 2, \dots,$$

each of which is solved (approximately) for  $u_\ell$ .

- follows physical transient when  $\delta_\ell$  is small
- associated Jacobians well-conditioned when  $\delta_\ell$  is small
- $\delta_\ell$  advanced from  $\delta_0 \ll 1$  to  $\delta_\ell \rightarrow \infty$  as  $\ell \rightarrow \infty$ , so that  $u_\ell$  approaches the root of  $F(u) = 0$
- Does *not* require reduction in  $\|F(u)\|$  at each step; can climb hills!
- Three-phase convergence analysis in Kelley & K. (1998)

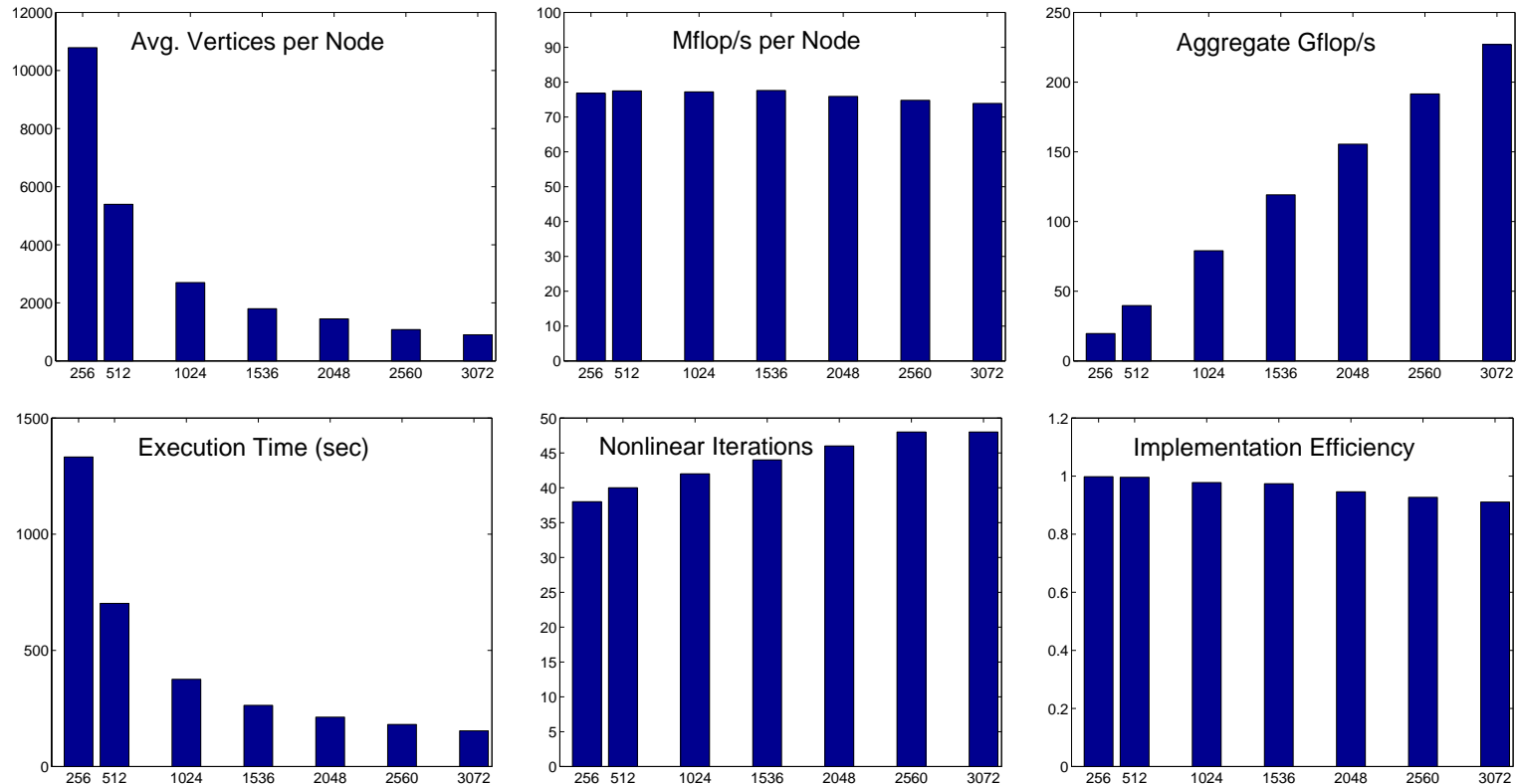
# Pseudo-transient Newton-Krylov-Schwarz ( $\Psi$ NKS)

NKS iteration is wrapped in pseudo-timestepping to globalize convergence to steady state:

```
do l = 1, n_time
  select time-step
  do k = 1, n_Newton
    compute nonlinear residual and Jacobian
    do j = 1, n_Krylov
      doall i = 1, n_Precon
        solve subdomain problems concurrently
      enddoall
    perform Jacobian-vector product
    enforce Krylov basis conditions
    update optimal coefficients
    check linear convergence
  enddo
  perform DAXPY update with robustness conditions
  check nonlinear convergence
enddo
```

# $\Psi$ NKS on Aerodynamics Problem

Incompressible Euler flow over an ONERA M6 Wing, 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 On-line References on $\Psi$ NKS Methods

- *Globalized Newton-Krylov-Schwarz Algorithms and Software for Parallel Implicit CFD* (with Gropp, McInnes & Tidriri), 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

## Recent On-line References on $\Psi$ NKS Methods, cont.

- *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
- 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 software
- Advantages of the latter:
  - reuse of quality parallel PDE software
  - the freedom to employ inexact solves, as finely resolved PDE discretizations in 3D prohibit exact elimination
  - 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.:

$$H \equiv W_{uu} - J_u^T J_x^{-T} W_{ux}^T + \left( J_u^T J_x^{-T} W_{xx} - W_{ux} \right) J_x^{-1} J_u$$

$$f \equiv -g_u + J_u^T J_x^{-T} g_x - \left( J_u^T J_x^{-T} W_{xx} - W_{ux} \right) J_x^{-1} h$$

**State Step** (last blockrow):

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

**Adjoint Step** (first blockrow):

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

## Complexity of Newton Reduced SQP

- 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

In order to avoid computing any Hessian blocks, the design step is approached in a quasi-Newton (e.g., BFGS) manner, and Hessian terms are dropped from the adjoint step RHS:

**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** (first blockrow):

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

# Complexity of Quasi-Newton Reduced SQP

- 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

# New Philosophy

- 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  $\lambda$  consistently (to eliminate them),
    - iterating
- Exact linearized analyses for updates to  $x$  and  $\lambda$  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)

## 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 temperature matching,  $c(u, x) = \frac{1}{2} \|T(x) - \bar{T}(x)\|^2$ , where  $\bar{T}$  is based on given  $\alpha$ ,  $\beta(x)$  profile
  - 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 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 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
- Exception: our trivial cost function differentiated “by hand”
- 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

# Convergence from Various Starting Points in Parameter Space

# Sample Convergence of State Variables

## 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 Arithmetic Complexity

We estimate the complexity of applying each Jacobian block, assuming only that  $h(x, u)$  is available in subroutine form and that all differentiated blocks are from AD tools, such as ADIC. 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  $J_u \equiv h, u$

Object	Cost: forward mode	Cost: fastest (hybrid) mode
$J_x, J_x^T$	$p_x C_h$	$p_x C_h$
$J_u v$	$2C_h$	$2C_h$
$J_u^T v$	$p_u C_h$	$r C_h^*$

\* reverse mode: consumes memory,  $r$  implementation dependent

## Hessian Arithmetic Complexity

Estimate the complexity of applying each forward 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. Define:

- $C_c$ , the cost of evaluating  $c$
- $q$ ,  $1 +$  number of nonzero rows in  $c''$
- $r$ , presently 5–50 for F77, 25–1000 for C; could be as low as 3–50 in future

Object	Cost: forward mode	Cost: fastest (hybrid) mode
$W_{xx}v, W_{ux}^T v$	$p_x C'_h + q C'_c$	$r(C'_h + C'_c)^*$
$W_{uu}v, W_{ux}v$	$p_u C'_h + q C'_c$	$r(C'_F + C'_c)^*$

\* reverse mode: consumes memory,  $r$  implementation dependent; reverse mode performance highly sensitive to memory bandwidth

## Arithmetic Complexity, cont.

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

## Promising Large-scale Results from **Biros & Ghattas**

- See August 2000 issue of SIAG-OPT newsletter (6-page feature article) on NSF-Sandia-funded TAOs project at CMU
- 3D Navier-Stokes flow boundary control problem for dissipation minimization on flow over a cylinder with suction/blowing
- Solved in domain-decomposed parallelism on 128 processors of a T3E, using optimization toolkit add-on to PETSc
- For  $6 \times 10^5$  state constraints and  $9 \times 10^3$  controls, full-space LNKS with approximate subdomain solves beats Quasi-Newton RSQP by an order of magnitude (4.1 hours versus 53.1 hours)
- QN-RSQP solves exactly with the NS Jacobian or its transpose 408 times; LNKS preconditioning solves with an approximate NS Jacobian or transpose an order of magnitude more times, but is far less expensive; see FRA226, Friday at 8am

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



**Lagrange**

optimizer



**Newton**

nonlinear solver



**Krylov**

accelerator



**Schwarz**

preconditioner

# Lagrange-Newton-Krylov-Schur: A Framework for Constrained Optimization



**Lagrange**

optimizer



**Newton**

nonlinear solver



**Krylov**

accelerator



**Schur**

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 system, 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

## Summary Remarks on LNKS

- Partitioning may improve robustness, conditioning, concurrency
- 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.
- The gamut of PDE solvers based on partitioning should be “mined” for application to the KKT necessary conditions of constrained optimization, and direct use on inverting the state Jacobian blocks inside the optimizer.

## Future Plans

- Migrate ADMAT/Matlab code into parallel ADIC/PETSc framework
- Increase physical dimensionality, parameter dimensionality
- Tune numerous preconditioning parameters for optimal parallel execution time
- Incorporate multilevel Schwarz linear preconditioning for spatial Jacobian
- Experiment with full nonlinear preconditioning of the KKT system

## Related URLs

- Personal
  - <http://www.math.odu.edu/~keye>*
  - papers, talks and workshop summaries
- PETSc project
  - <http://www.mcs.anl.gov/petsc>*
- ADIC project
  - <http://www.mcs.anl.gov/adic>*
- Int. Conferences on Domain Decomposition Methods
  - <http://www.ddm.org>*
- NSF Multidisciplinary Computing Challenges group
  - <http://www.math.odu.edu/~keye/nf>*
- DOE Accelerated Strategic Computing group
  - <http://www.math.odu.edu/~keye/asci>*