Meros: Specialized Preconditioners for Problems with Coupled Simultaneous Solution Variables

Victoria E. Howle
Department of Mathematics & Statistics
Texas Tech University

Copper Mountain 2008, Trilinos Workshop
April 7, 2008

In collaboration with:

- Ray Tuminaro (Sandia National Labs)
- Robert Shuttleworth (Exxon)
- Howard Elman (University of Maryland)
- John Shadid (Sandia National Labs)
- David Silvester (University of Manchester)

Incompressible Navier-Stokes

$$\begin{aligned} \alpha \mathbf{u}_t - \nu \nabla^2 \mathbf{u} + (\mathbf{u} \cdot \mathrm{grad})\, \mathbf{u} + \mathrm{grad}\, p &= \mathbf{f} \\ -\mathrm{div}\, \mathbf{u} &= 0 \end{aligned}$$

- $\mathbf{u}$ = velocity; p = pressure; ν = viscosity
- $\alpha = 0$ steady-state; $\alpha = 1$ unsteady flow

Linearization and discretization (possibly stabilized) leads to:

$$\begin{bmatrix} F & B^T \\ B & -\frac{1}{\nu} C \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ g \end{bmatrix}$$

- B and B^T are discrete divergence and gradient operators
- F operates on the discrete velocity space
- Generally $C = 0$ for *div-stable* discretizations; otherwise C is a nonzero stabilization parameter

Schur Complement Preconditioners

- Consider preconditioners of the form

$$P = \begin{bmatrix} F & B^T \\ & X \end{bmatrix}$$

This is an optimal (right) preconditioner when X is the Schur complement $S = BF^{-1}B^T + \frac{1}{\nu}C$

- The Schur complement is computationally expensive; so need to approximate
- We want the scalability of multigrid (h-independence)
  - Can be difficult to apply multigrid to whole system
  - X spectrally equivalent to S $\rightarrow$ h-independence for P for Stokes problem (Silvester & Wathen, 1994)

$$P^{-1} = \begin{bmatrix} F^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & B^T \\ & I \end{bmatrix} \begin{bmatrix} I & \\ & X^{-1} \end{bmatrix}$$

- Suppose $B^T F_p = F B^T$ and $X^{-1} = F_p (B B^T)^{-1}$
  Then $S X^{-1} = (B F^{-1} B^T) F_p (B B^T)^{-1} = I \quad (C = 0)$
- ***Pressure Convection-Diffusion*** preconditioner of Kay, Loghin, and Wathen (2002) and Silvester, Elman, Kay, and Wathen (2001)

$$S \approx X = A_p F_p^{-1} M_p$$

  - $M_p =$ pressure mass matrix associated with the pressure discretization
  - $A_p =$ discrete Laplace operator defined on pressure space.
  - $F_p =$ discrete convection-diffusion operator defined on pressure space.
- This approach has a practical issue: user software must supply F_p and A_p.
- Other methods developed to minimize need for nonstandard operators.

- ***Least Squares Commutator***
  Elman, VH, Shadid, Shuttleworth, and Tuminaro (2006)
  $$S \approx X = (BM_*^{-1}B^T)\,(BM_*^{-1}FM_*^{-1}B^T)^{-1}(BM_*^{-1}B^T).$$
  - $M_* =$ (diagonal part of) velocity mass matrix
- **Stabilized LSC**
  Elman, VH, Shadid, Silvester, Tuminaro (2007)
  - Fully algebraic method:
    $$X^{-1} = A_p^{-1}(BM_*^{-1}FM_*^{-1}B^T)A_p^{-1} + \alpha D$$
    $A_p = (BM_*^{-1}B^T + \gamma C)$; simple formulas for α and γ;
    $D = diag(B(diag\ F)^{-1}B^T + C)$
  - "Element-based" method:
    $$X^{-1} = A_P^{-1}(BM_*^{-1}FM_*^{-1}B^T + \frac{\nu}{h^4}C)A_p^{-1}$$
    $A_p = B(M_*^{-1})B^T + \frac{1}{h^2}C$

Implementation in Meros

- Block algorithms implemented in Meros package
  - Scalable block preconditioning package within Trilinos
  - Currently implements several block methods
    - pressure convection-diffusion
    - least squares commutator
    - SIMPLE
- Publicly released (LGPL) within Trilinos
- Based on Thyra abstract interface
- Uses Thyra, Teuchos, AztecOO
- Accepts Thyra linear operators and Epetra matrices
- Tested in internal version of MPSalsa (incompressible flow code) with good results
- Tested in Sundance: good preliminary results on microfluidics problems

- Trilinos provides parallel linear algebra kernels (Epetra), an abstract interface that allows block and composed operations (Thyra), solvers (AztecOO, Belos, ML), etc.
- With Thyra, we can easily write block systems that reflect the mathematical algorithms. E.g., in PCD preconditioner:

```
Finv = inverse(*FSolveStrategy_, F, ...);
Apinv = inverse(*ApSolveStrategy_, Ap, ...);
Mpinv = inverse(*MpSolveStrategy_, Mp, ...);
Xinv = Mpinv * Fp * Apinv;
Ivel = identity<double>(Bt.range());
Ipress = identity<double>(Bt.domain());
ConstLinearOperator<double> zero;
P1 = block2x2( Finv, zero, zero, Ipress );
P2 = block2x2( Ivel, (-1.0)*Bt, zero, Ipress );
P3 = block2x2( Ivel, zero, zero, (-1.0)*Xinv );

PCDprec = P1 * P2 * P3;
```

- (Glossing over templates, typing, and some other arguments.)

$$P^{-1} = \begin{bmatrix} F^{-1} & \\ & I \end{bmatrix} \begin{bmatrix} I & B^T \\ & I \end{bmatrix} \begin{bmatrix} I & \\ & X^{-1} \end{bmatrix}$$

$$X^{-1} = M_p^{-1} F_p A_p^{-1}$$

- At the user level, we need to specify which preconditioner, and provide parameters for subsolves (or accept defaults).
- For example, for the PCD preconditioner:

```
merosPrecFac = new PCDPreconditionerFactory(
  SolveStrategies or ParameterLists for F, Ap, Mp );
Prcp = merosPrecFac->createPrec();
PCDOpSrc = rcp(new PCDOperatorSource(blockOp, Fp, Ap, Mp));
merosPrecFac->initializePrec(PCDOpSrc, &*Prcp);
```

Then we specify an outer solver strategy (param list) and do the solve:

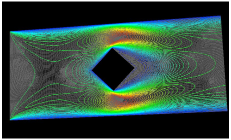
```
Pinv = Prcp->getRigthPrecOp();
saddleInv = new InverseOperator(blockOp * Pinv, azSaddleStrategy);
solnblockvec = saddleInv * rhs;
```

Steady 3D lid driven cavity in MPSalsa

Re	Mesh size	DD	PCD (Meros)	Nprocs
10	$32 \times 32 \times 32$	67.0	28.0	1
	$64 \times 64 \times 64$	159.8	28.4	8
50	$32 \times 32 \times 32$	62.2	40.2	1
	$64 \times 64 \times 64$	162.6	47.8	1
100	$32 \times 32 \times 32$	61.7	56.0	1
	$64 \times 64 \times 64$	168.5	62.1	1

- DD is default domain decomposition
- PCD is pressure convection-diffusion preconditioner
- Results show average number of outer linear iterations per Newton step
- DD was faster on 1 proc.; PCD was faster on 8 procs.

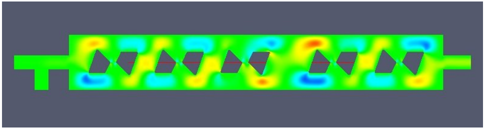
Steady 2D flow over a diamond obstruction in MPSalsa



Re	Unknowns	DD	PCD (Meros)	Nprocs
10	64K	110.8	20.5	1
	256K	284.6	22.5	4
	1M	1329.0	22.9	16
	4M	NC	29.4	64
25	64K	101.7	32.9	1
	256K	273.8	35.9	4
	1M	1104.8	38.3	16
	4M	NC	48.0	64
40	64K	70.4	54.6	1
	256K	203.9	70.1	4
	1M	997.1	65.4	16
	4M	NC	79.8	64

Microfluidics in Sundance

- Meros used in Sundance in modeling the design of a microfluidic mixing device
  - induced charge electroosmosis, by which flow through device is driven by a set of charged obstacles
  - optimizing (APPSPACK) shape and orientation of the obstacles to maximize fluid mixing within device
  - constrained optimization problem; function evaluations require numerical solutions of PDEs
    - electrostatic potential equation
    - incompressible Navier-Stokes (most expensive)
    - mass transport
- Shuttleworth, Elman, Long, Templeton
- See Bob Shuttleworth's talk on Thursday



- Conclusions
  - Using problem structure to develop preconditioning methods
  - Methods implemented in Meros
  - Meros released LGPL within Trilinos
  - Extended to stabilized discretizations
  - Tested and used in problems in MPSalsa and Sundance with good results
- Ongoing efforts
  - Implement other methods in Meros
    - stabilized LSC methods
    - other preconditioners
  - Documentation
  - Connections to Belos, Stratimikos, PyTrilinos, etc.
  - Other work on the methods themselves, e.g.
    - NS coupled with other physics (e.g., chemically reactive flow)
    - boundary conditions for PCD