

Adaptivity features of iterative methods represented as components

Example of SPARSKIT-CCA components

Masha Sosonkina

Ames Laboratory/DOE

July 2007 TASCs meeting

What is SPARSKIT?

- Well-known library of *serial* sparse matrix kernels by Yousef Saad (University of Minnesota).
- Written in Fortran77.
- Provides a range of functions for sparse matrix computations with a focus on iterative solution techniques
- **Accelerator** iterative procedure to obtain approximate solution in a particular subspace.
- **Preconditioner** helps the accelerator to converge by transforming the original problem into one that is easier to solve and has the same solution.
- BLASSM is a suite of BLAS-like operations on sparse matrices.

Design choices

Low-level: Expresses all sparse matrix operations by components.

Medium-level: Only major sparse matrix operations are components.

High-level: Entire iterative solution is encapsulated into a component.

SPARSKIT is a suite of medium-level CCA components

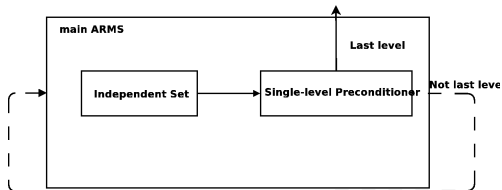
- A component implements particular preconditioner creation process.
- A component implements particular accelerator.
- Component interfaces have standard argument lists (e.g., for different preconditioners of a certain type).

Adaptivity features of preconditioners

Algebraic Recursive Multilevel Solver (ARMS) is an **adaptive preconditioner framework**.

Preconditioner interfaces

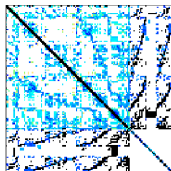
- `BasePreconditioner`, for the preconditioners with Fortran-style sparse matrix representations as arguments.
- `GenericPreconditioner`, for the preconditioners with more general sparse matrix representations.



Last-level preconditioner change in ARMS

An example

- Two types of incomplete LU factorization preconditioners are at the last level.
 - ILUC or ILUT
- Consider two difficult to solve linear systems, *scircuit* and *igbt3*.



matrix	<i>nnz</i>	Precon	its	time
scircuit	958,936	ARMS+ILUC	7	124.9
		ARMS+ILUT	7	125.1
		ILUC	55	11.2
		ILUT	*	*
igbt3	234,006	ARMS+ILUC	8	20.23
		ARMS+ILUT	6	19.5
		ILUC	*	*
		ILUT	19	0.55

Tasks requiring CQoS infrastructure

- 1 Instantiate more than one “last-level” preconditioner.
- 2 At preconditioner creation stage,
 - 1 Estimate the *quality* of ARMS with different last level preconditioners.
 - 2 Choose **statically** the better one within ARMS
- 3 Capture the convergence progress information (current residual norm reduction).
- 4 Time individual components (via proxies).
- 5 Choose preconditioner **dynamically** based on items 3 and 4 within ARMS.

May switch preconditioners at each iteration if flexible FGMRES is used.

Performance analysis using TAU

<i>nnz</i>	Func	Calls	SKIT-CCA	SKIT	Norm, %
38,880	lusol	46	20	16	25.0
	create	1	18	16	12.5
	amux	47	12	11	9.1
	apply	93	12	23	0.0
45,847	create	1	25	20	25.0
	lusol	46	20	20	0.0
	amux	47	13	13	0.0
	apply	93	13	12	8.3
76,760	lusol	36	34	34	2.4
	create	1	34	34	0.0
	apply	73	28	27	2.9
	amux	38	22	22	0.0
179,800	create	1	96	95	1.3
	apply	73	96	89	7.3
	lusol	36	91	80	13.1
	amux	38	65	65	0.0