

# Toward a Memory-Efficient Linear Solver

Allison Baker

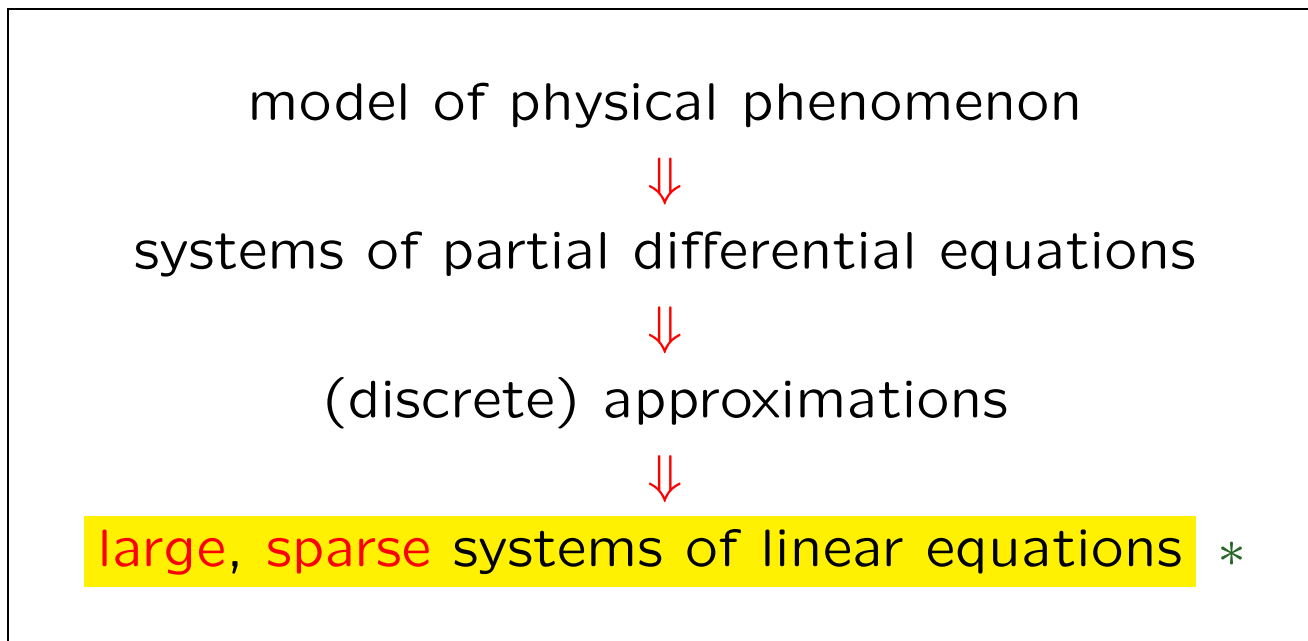
Department of Applied Mathematics

E.R. Jessup, J.M. Dennis  
Department of Computer Science

University of Colorado at Boulder



# Linear systems are ubiquitous

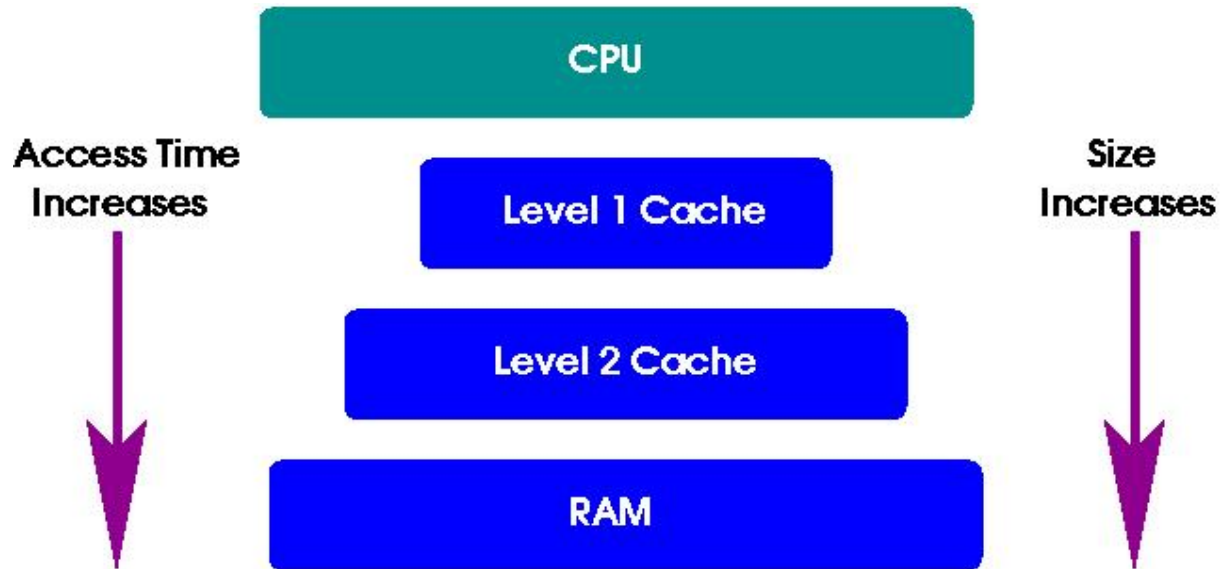


\* often time-consuming to solve!

# Can we solve large, sparse systems of linear equations more efficiently?

- Linear system:  $Ax = b$   
( $A$  is large, sparse, and nonsymmetric)
- Costs: (1) floating-point operations  
(2) movement of data through memory
- Our goal: reduce memory costs (2)

# Memory Hierarchy



component	Typical CPU latency value
L1 cache	1 - 3 cycles
L2 cache	6 - 20 cycles
RAM (main memory)	60 - 140 cycles
disk	10,000 cycles !!!

## Memory costs impact performance

- moving data is expensive
- number of floating-point operations required by algorithms has decreased

- getting worse:

microprocessor performance    vs.    memory performance:  
    ~ 60% per year                      < 10% per year

⇒ time the CPU spends waiting for data is increasing!

## Iterative linear solvers

- repeatedly revise estimate for solution  $\mathbf{x}$
- based on a sparse matrix-vector multiply
- access  $\mathbf{A}$  once per iteration loop for  $\mathbf{A} * \mathbf{x}$

⇒ number of floating point operations is not an accurate predictor of algorithm performance

⇒ need to minimize data movement

# Approach to improving an iterative linear solver

## Algorithmic developments:

- Solve  $\mathbf{Ax} = \mathbf{b}$  via  $\mathbf{AX} = \mathbf{B}$

$$\begin{bmatrix} \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \end{bmatrix} \Rightarrow \begin{bmatrix} \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{X} \end{bmatrix} = \begin{bmatrix} \mathbf{B} \end{bmatrix}$$

- choose  $\mathbf{X}$  and  $\mathbf{B}$  to promote convergence

## Performance programming techniques:

- recover costs of  $\mathbf{A} * \mathbf{X}$  vs.  $\mathbf{A} * \mathbf{x}$

## A common linear solver: GMRES( $m$ )

Why?: popular for nonsymmetric problems!

Solve:  $\mathbf{Ax} = \mathbf{b}$

initial guess  $\rightarrow \mathbf{x}_0$

residual  $\rightarrow \mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$

Iterate:

- $\mathbf{x}_{i+1} \in \mathbf{x}_i + \mathcal{K}_m(\mathbf{A}, \mathbf{r}_i)$

$$\mathcal{K}_m(\mathbf{A}, \mathbf{r}_i) \equiv \text{span}\{\mathbf{r}_i, \mathbf{Ar}_i, \dots, \mathbf{A}^{m-1}\mathbf{r}_i\}$$

- minimize the 2-norm of the residual:  $\|\mathbf{b} - \mathbf{Ax}_{i+1}\|_2$
- discard approximation space and restart ( $i = i + 1$ )

## What should we choose for $B$ for GMRES( $m$ )? ( $Ax = b \Rightarrow AX = B$ )

- goal:  
balance memory-usage and favorable numerical properties
- observation:  
*restarted* GMRES( $m$ ) throws away useful information with every restart, which slows convergence
- our approach:  
compensate for this loss  $\rightarrow$  put the lost information in  $B$ !

$$\mathbf{Ax} = \mathbf{b} \Rightarrow \mathbf{AX} = \mathbf{B}$$

### GMRES( $m$ ):

- restart cycle  $i$ :  $\mathbf{x}_i = \mathbf{x}_{i-1} + \mathbf{c}$ ,  $\mathbf{c} \in \mathcal{K}_m(\mathbf{A}, \mathbf{r}_{i-1})$
- ideally:  $\mathbf{c} = \mathbf{x}_{\text{exact}} - \mathbf{x}_{i-1} \longrightarrow \mathbf{x}_i = \mathbf{x}_{\text{exact}}$
- error approximation:  $\mathbf{z}_i \equiv \mathbf{x}_i - \mathbf{x}_{i-1}$

### idea:

- put the error approximation vector in  $\mathbf{B}$ :  $\mathbf{B} = [\mathbf{b}, \mathbf{z}_i]$

## A new algorithm: B-LGMRES

### B-LGMRES( $m, 1$ ):

- restart cycle  $i + 1$ :  $\mathbf{Ax} = \mathbf{b} \Rightarrow \mathbf{AX} = \mathbf{B}$

$$\mathbf{X} = [ \mathbf{x}_i, \mathbf{0} ]$$

$$\mathbf{B} = [ \mathbf{b}, \mathbf{z}_i ]$$

i.e., augment  $\mathbf{b}$  with an error approximation

- $\underline{\mathbf{x}_{i+1} \in \mathbf{x}_i + \mathcal{K}_m(\mathbf{A}, \mathbf{r}_i) + \mathcal{K}_m(\mathbf{A}, \mathbf{z}_i)}$   
||  
GMRES( $m$ )

(resembles a full conjugate gradient method with polynomial preconditioning)

## B-LGMRES: an efficient implementation

### Matrix-multivector multiply (Kaushik '99):

multiply  $\mathbf{A}^*4$  vectors in  $\approx 1.5$  the time for  $\mathbf{A}^*1$  vector

- interleave the vector components in memory:  
 $v_1(j), v_2(j), v_3(j), v_4(j)$  are adjacent
- unroll the inner matrix multiplication loop  
(across multivector elements)

### Other misc. multivector routines:

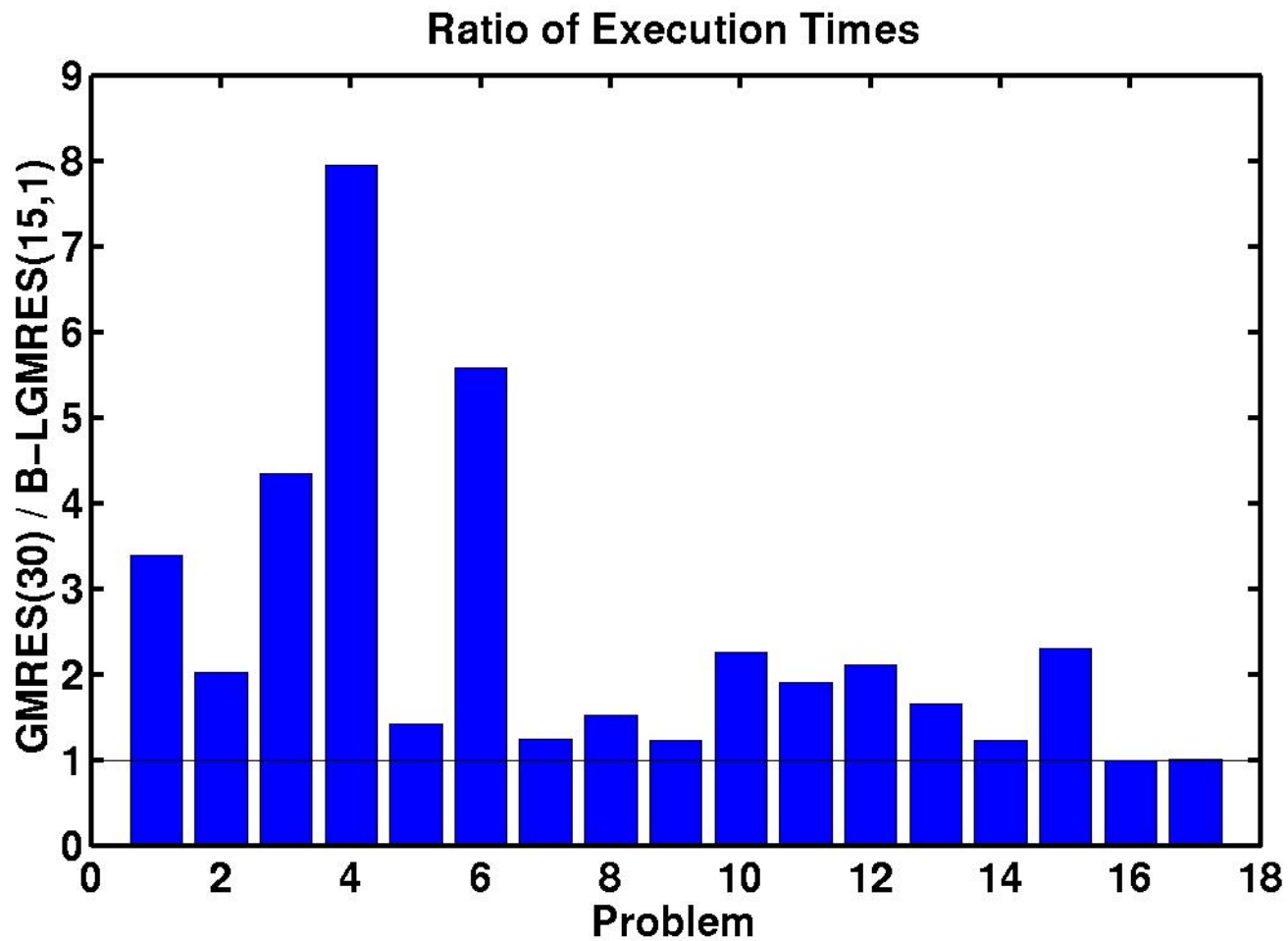
multivector dot, saxpy, maxpy, and forward and back solves

## Evaluating the new method:

### B-LGMRES(15, 1) vs. GMRES(30):

- equal approximation space size (30)
- variety of problems from several test collections
- problem size:  $n = 7,740$  to  $1,709,982$   
( $nnz = 79,566$  to  $1,717,792$ )
- either ILU preconditioner or no preconditioner
- Argonne's PETSc with modifications

## Some results...



⇒ the new method is generally faster

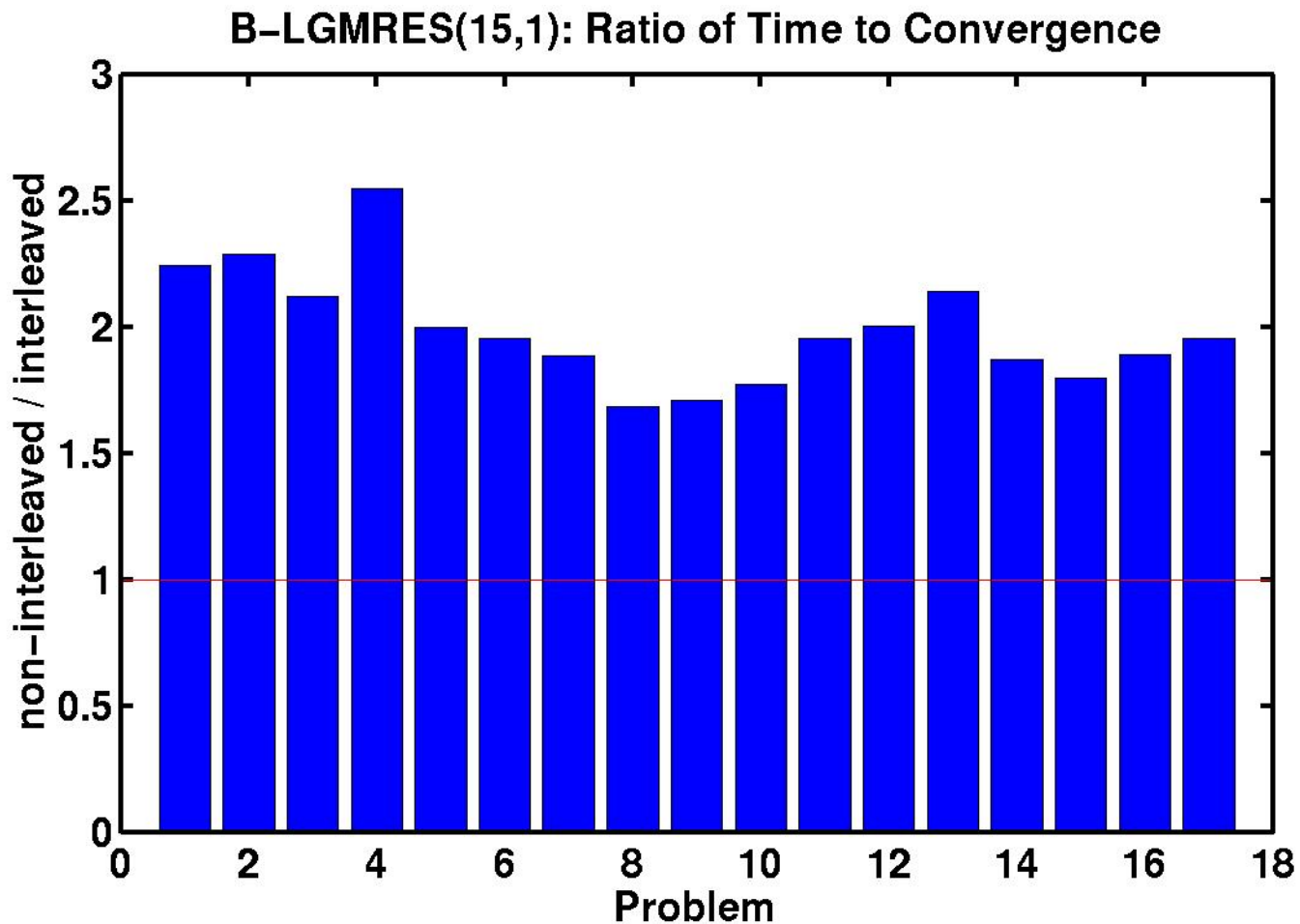
## Problem 12: 2-D fluid flow

size:  $n=13,535$ ,  $nnz=390,607$

method	execution time	accesses of $A$	matrix-vector multiplies
GMRES(30)	$165.9 \pm .6$ s	1960	1960
B-LGMRES(15,1)	$78.5 \pm .2$ s	1004	2008

⇒ execution time correlates to number of accesses of  $A$   
(related to memory usage, not floating-point operations)

# Importance of an efficient implementation



interleaved:  $v_1(j)$ ,  $v_2(j)$  are adjacent

non-interleaved:  $v_1(j)$ ,  $v_1(j+1)$  are adjacent

## In summary...

### Simple changes have large impacts on performance

- algorithmic changes: changing  $b$  to  $B$
- implementation: basic programming tricks

⇒ **modifying a linear solver to reduce data movement is a viable approach to gaining efficiency!**

## For more information...

- **A.H. Baker**, J.M. Dennis, and E.R. Jessup. **An efficient block variant of GMRES**. Technical Report #CU-CS-957-03, University of Colorado, Department of Computer Science, July, 2003. *Submitted for publication.*
- **A.H. Baker**, E.R. Jessup, and T. Manteuffel. **A technique for accelerating the convergence of restarted GMRES**. Technical Report #CU-CS-945-03, University of Colorado, Department of Computer Science, January, 2003. *Submitted for publication.*

Download from:

<http://amath.colorado.edu/student/allisonb/research.html>