

# *Iterative Methods and Preconditioning Techniques*

Esmond G. Ng  
(EGNg@lbl.gov)

Lawrence Berkeley National Laboratory



# *Direct Solution of Linear Systems of Equations*

- ❑ Need a triangular factorization of the coefficient matrix.
- ❑ Require a finite number of operations to compute the solution.
- ❑ Can pivot to maintain numerical stability.
  
- ❑ Suffer from fill (zero entries turn into nonzero) when coefficient matrix is sparse.
  - Manage fill.
  - Cost of Gaussian elimination.
- ❑ Need extra processing for sparse matrices.
  - Ordering
  - Symbolic factorization

# *Iterative Solution of Linear Systems of Equations*

- ❑ Start with an initial guess of the solution.
- ❑ Generate a sequence of approximations.
- ❑ Often require only matrix-vector multiplications and inner products.
  - Great for sparse linear systems since no matrix factorizations are needed.
  
- ❑ Generating the sequence???
- ❑ Convergence of the sequence???
- ❑ Often unpredictable number of operations.

# *Iterative Solution of Linear Systems of Equations*

- ❑ Many iterative methods and lot of theory.
  
- ❑ Excellent references:
  - Richard Barrett, Michael Berry, Tony Chan, James Demmel, June Donato, Jack Dongara, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk van der Vorst, "Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods", SIAM, 1994.
  
  - Yousef Saad, "Iterative Methods for Sparse Linear Systems", available from <http://www-users.cs.umn.edu/~saad/books.html>.
  
  - Henk van der Vorst, "Iterative Krylov Methods for Large Linear Systems", to be published in 2003.

# Iterative Solution of Linear Systems of Equations

- ❑ Some are simple:
  - Jacobi
  - Gauss-Seidel
  - Successive overrelaxation and symmetric successive overrelaxation
  
- ❑ Others are more complicated:
  - Conjugate gradient and generalized conjugate gradient
  - Minimum residual and generalized minimal residual
  - Biconjugate gradient and biconjugate gradient stabilized
  - Conjugate gradient squared
  - Quasi-minimal residual
  - Chebyshev iterations

# The Conjugate Gradient Method

- ❑ For symmetric positive definite linear systems, the conjugate gradient method is the most popular method.
- ❑ Based on minimizing a convex quadratic function.

$$\phi(x) = \frac{1}{2} x^T A x - x^T b$$

- ❑ The minimizer is given by the solution of  $Ax = b$  .

# The Conjugate Gradient Method

- Basic idea in minimizing  $\phi(x) = \frac{1}{2}x^T A x - x^T b$  :
  - Given an approximation to the minimizer,  $x^{(i-1)}$ .
  - Select a search direction  $p^{(i)}$ .
  - Minimize  $\phi(x)$  along the direction  $x^{(i-1)} + \alpha p^{(i)}$ .
    - ◆ This becomes a problem of a single (scalar) variable:  $\alpha$ .
  - Let  $\alpha_i$  be the minimizer.
  - Then set  $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$ .
  - Repeat ...
  
- Choice of search directions  $p^{(i)}$  ?
  - Chosen so that they are A-orthogonal (or A-conjugate).  
 $[p^{(i)}]^T A [p^{(j)}] = 0$  , for  $i \neq j$  .

# The Conjugate Gradient Method

Given an initial guess  $x^{(0)}$ .

Compute the initial residual  $r^{(0)} = b - Ax^{(0)}$ .

Set  $p^{(0)} = 0$ ;  $\rho_{-1} = 1$ .

for  $i = 1, 2, \dots$

$$\rho_{i-1} = [r^{(i-1)}]^T r^{(i-1)}$$

$$\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$$

$$p^{(i)} = r^{(i-1)} + \beta_{i-1} p^{(i-1)}$$

$$q^{(i)} = Ap^{(i)}$$

$$\alpha_i = \rho_{i-1} / [p^{(i)}]^T q^{(i)}$$

$$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$$

$$r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$$

2 inner products

1 matrix-vector multiplication

3 vector updates

Check convergence; continue if necessary

end

# The Conjugate Gradient Method

- ❑ Need 2 inner products and 1 matrix-vector multiplication.
  - Great for sparse linear systems.
  - No fill to worry about.
  
- ❑ Parallel implementation of the conjugate gradient method.
  - Assume that each vector is partitioned among the processors.
  - Partial inner products can be computed in parallel and then globally summed.
  
  - Need parallel sparse matrix-vector multiplication!

# Notion of Preconditioning

- Lot of theory on the conjugate gradient method ...
  - The conjugate gradient method converges in at most  $n$  iterations.
    - ◆ Krylov subspaces.
    - ◆ In practice, difficult to say because of roundoff errors.
  - If  $A$  has  $m$  distinct eigenvalues, then the conjugate gradient method requires  $m$  iterations to converge to a solution.
    - ◆ Desirable to have a small number of clustered eigenvalues.
- To improve the convergence, change the eigenvalues of the coefficient matrix.
  - The preconditioning step.
  - Many choices.

# Preconditioning

□ Consider  $Ax = b$  ;  $A$  symmetric positive definite.

□ Let  $K$  be a nonsingular matrix.

□ Then  $KAK^T$  is symmetric positive definite.

□ Instead of solving  $Ax = b$ , solve

$$(KAK^T)(K^{-T}x) = \boxed{By = c} = Kb .$$

with  $B = KAK^T$  as the coefficient matrix,  $c = Kb$  as the right-hand side, and  $y = K^{-T}x$  as the unknown vector.

□ The eigenvalue distribution for  $B = KAK^T$  may be quite different from that of  $A$  .

▪  $K$  is referred to as the preconditioning matrix or preconditioner.

□ Choice of  $K$  ?

# Preconditioning ... An Example

- ❑  $A$  is symmetric and positive definite.
- ❑ Let  $L$  be the Cholesky factor of  $A$  :  $A = LL^T$  .
- ❑ Suppose we were able to set  $K = L^{-1}$  .
- ❑ Then  $KAK^T = L^{-1}AL^{-T} = L^{-1}(LL^T)L^{-T} = (L^{-1}L)(L^TL^{-T}) = I$  .
- ❑ The conjugate gradient method, when applied to  
 $(KAK^T) (K^{-T} x) = (K b)$  ,  
will converge in 1 iteration.
- ❑ Need more realistic choices.

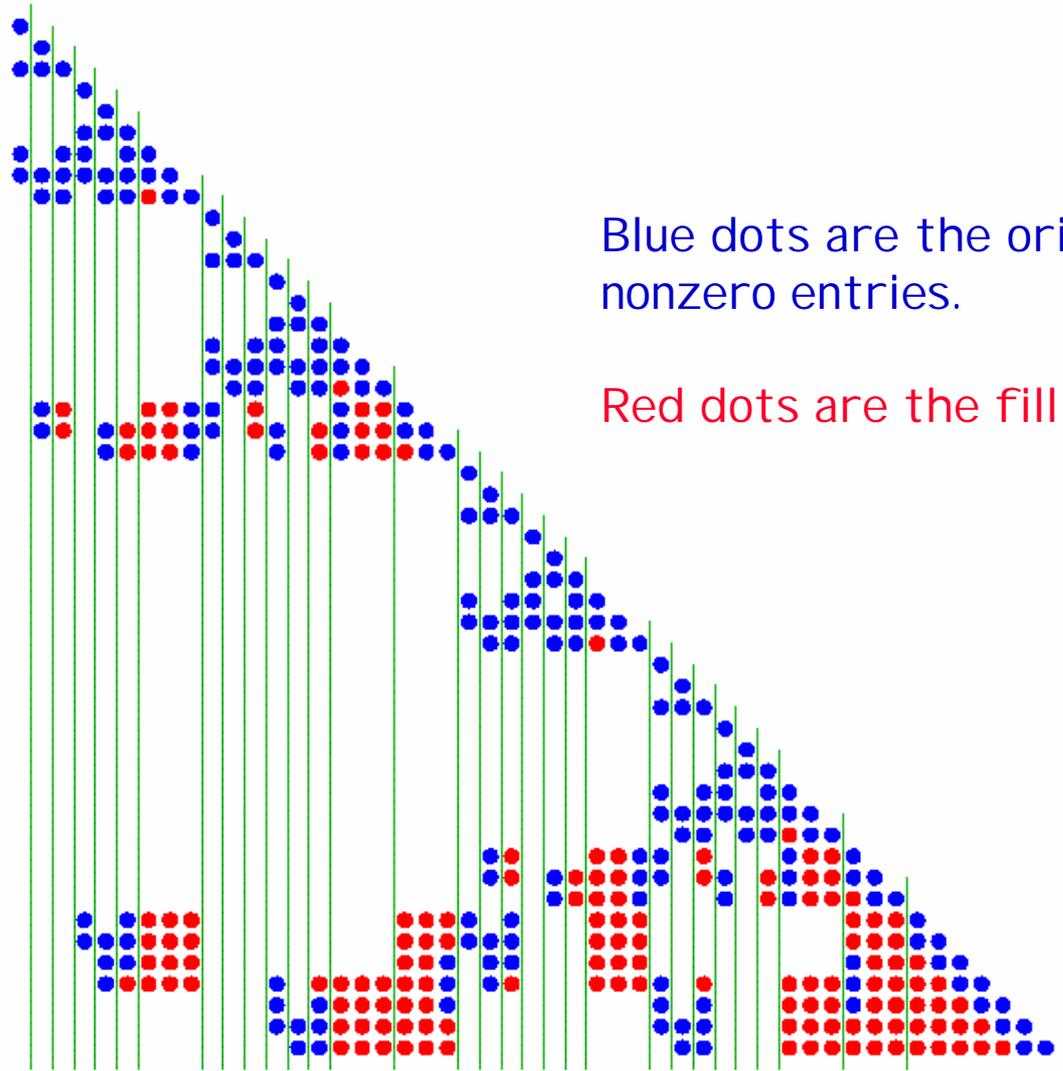
# Approximate Triangular Factorization

- ❑ Talked about sparse Gaussian elimination last time.
- ❑ If  $A$  is symmetric positive definite, then  $A = LL^T$ , with  $L$  being lower triangular (Cholesky factorization).
- ❑ Instead of computing  $L$  exactly, we can compute an approximation to  $L$ :  $L_a$ .
- ❑ Then set  $K = L_a^{-1}$ .
- ❑ If  $L_a$  is a good approximation to  $L$ , then  $KAK^T$  will be close to  $I$ , and  $L_a$  will be a good preconditioner for  $A$ .
- ❑ How to construct the approximation?

# Incomplete Triangular Factorization

- ❑ Recall that the triangular factorization of a sparse matrix  $A$  will change some of the zero entries into nonzero (fill).
- ❑ Idea for constructing the approximation:
  - Discard some of the fill entries in the factorization.
  - The discarded fill entries may or may not participate in updating future columns (depending on the algorithms and location of fill).
  - Resulting triangular factor is inexact, and is called an incomplete factor.
  - Simplest dropping criterion: Discard all fill entries.
  - Many other ways to drop fill to produce an incomplete factorization.
- ❑ Matrix may lose positive definiteness.

# Example of Sparse Cholesky Factor



Blue dots are the original nonzero entries.

Red dots are the fill entries.

# Computing Incomplete Factorization in Parallel

- Once a dropping criterion is decided, the incomplete factorization can be computed in parallel, much like sparse complete factorization although there are fewer operations and less fill.
  - Column modifications  $\text{cmod}(j_1, k)$  and  $\text{cmod}(j_2, k)$  can be performed in parallel as long as columns  $j_1$  and  $j_2$  are assigned to different processors and as long as column  $k$  of the factor is made available to  $j_1$  and  $j_2$ .
  - Some of the  $\text{cdiv}$  operations can also be performed in parallel.
- Related issues: ordering and symbolic factorization.
  - Different from sparse direct methods and can be much more difficult.

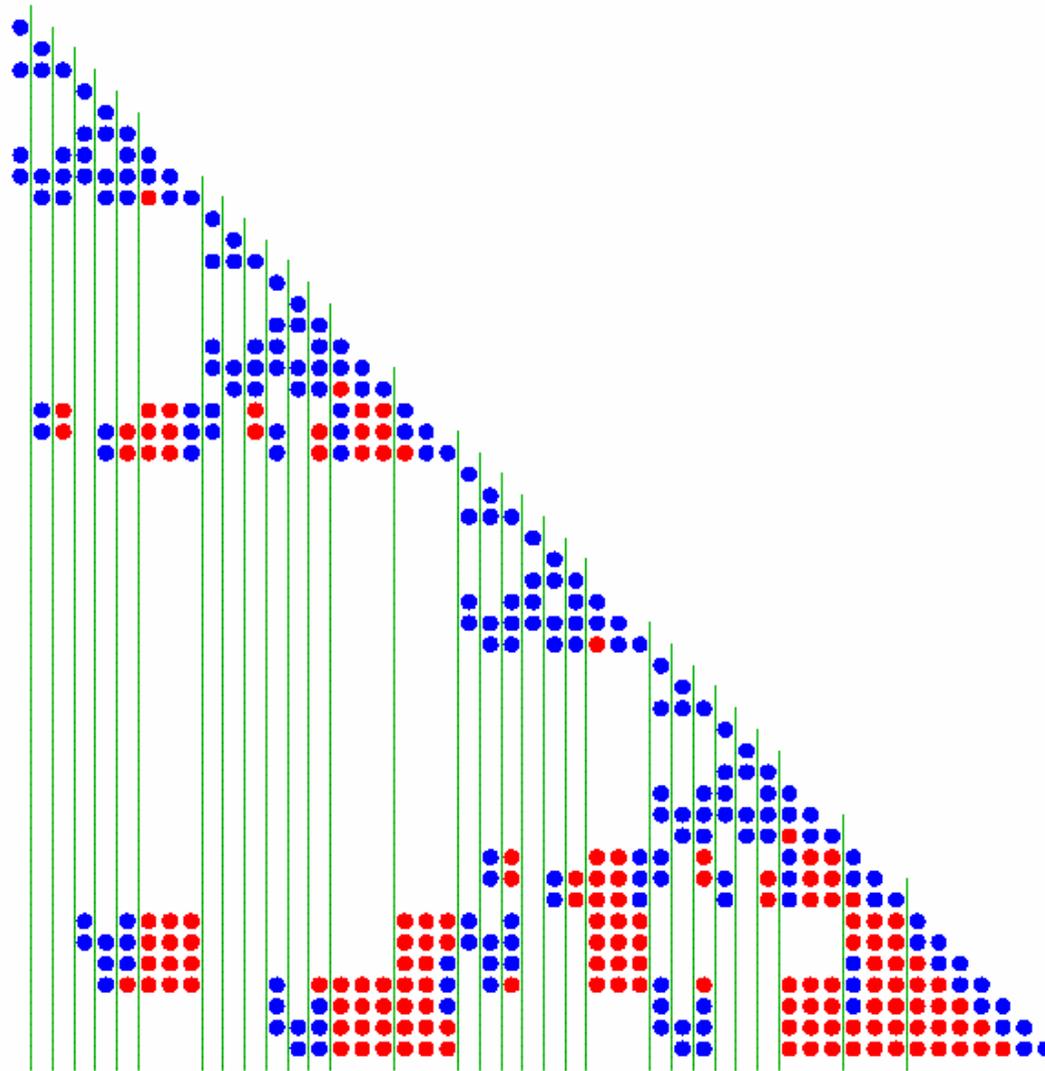
# Computing Incomplete Factorization in Parallel

- ❑ It is much harder to get good efficiency in parallel incomplete factorization than in parallel complete factorization.
- ❑ Why?
  - Usually much fewer operations to perform.
  - Unlike sparse complete factorization, usually no dense submatrices to allow memory hierarchy to be exploited.

# A Left-looking Block Incomplete Factorization

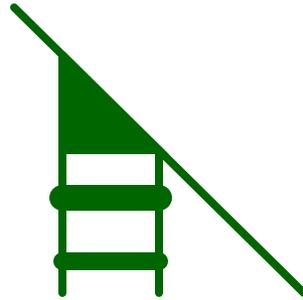
- ❑ Based on the theory and left-looking algorithms for sparse complete Cholesky factorization [Ng/Raghavan '99-'01].
  
- ❑ Choice of dropping criterion:
  - Nonzero entries that do not satisfy a threshold condition are dropped from the Cholesky factor, and they do not participate in updating later columns of  $A$ .
  
- ❑ Take advantage of the supernodal structure to exploit memory hierarchy.
  - Drop rows of nonzero entries instead of individual nonzero entries.
    - ◆ Within a supernode, a row of nonzero entries is dropped if all entries satisfy the threshold condition.
    - ◆ Force dense blocks to be retained.

# Left-looking Block Incomplete Factorization



# Left-looking Block Incomplete Factorization

- ❑ Requiring a row of nonzero entries within a supernode to be dropped is a very stringent condition ... may lead to too few nonzero entries being dropped in large supernodes.



- ❑ Remedy: Divide each large supernode into small blocks (e.g., block size 2, 4, or 8).
- ❑ This forces a block structure in the incomplete factorization.
- ❑ Can be implemented easily in the framework of (serial/parallel) sparse complete factorization.
  - Still exploit memory hierarchy.

# Preconditioned Conjugate Gradient Method

Given an initial guess  $x^{(0)}$ . Compute incomplete factor  $K$ .

Compute the initial residual  $r^{(0)} = (Kb) - (KAK^T) x^{(0)}$ .

Set  $p^{(0)} = 0$ ;  $\rho_{-1} = 1$ .

for  $i = 1, 2, \dots$

$$\rho_{i-1} = [r^{(i-1)}]^T r^{(i-1)}$$

$$\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$$

$$p^{(i)} = r^{(i-1)} + \beta_{i-1} p^{(i-1)}$$

$$q^{(i)} = (KAK^T) p^{(i)}$$

$$\alpha_i = \rho_{i-1} / [p^{(i)}]^T q^{(i)}$$

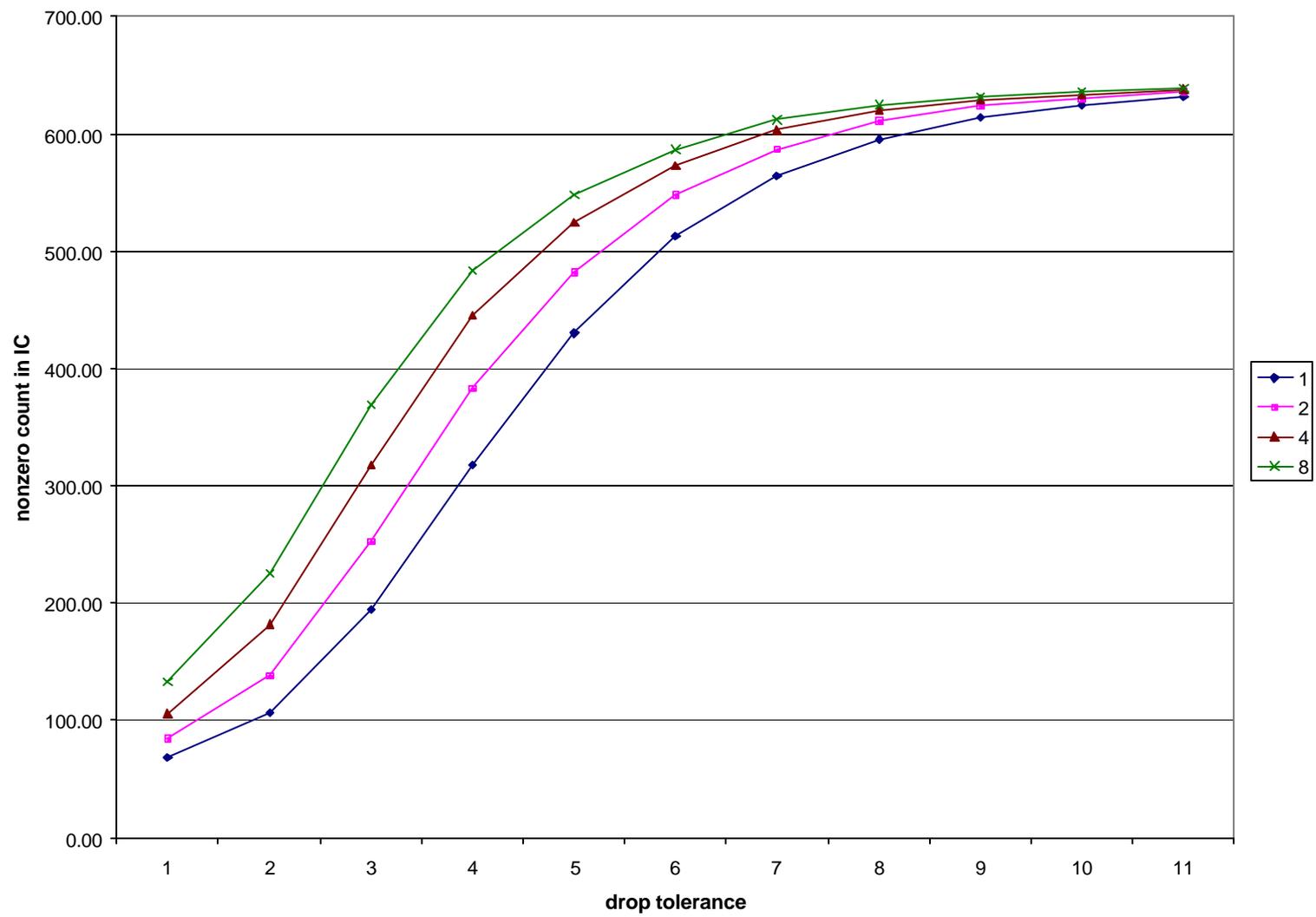
$$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$$

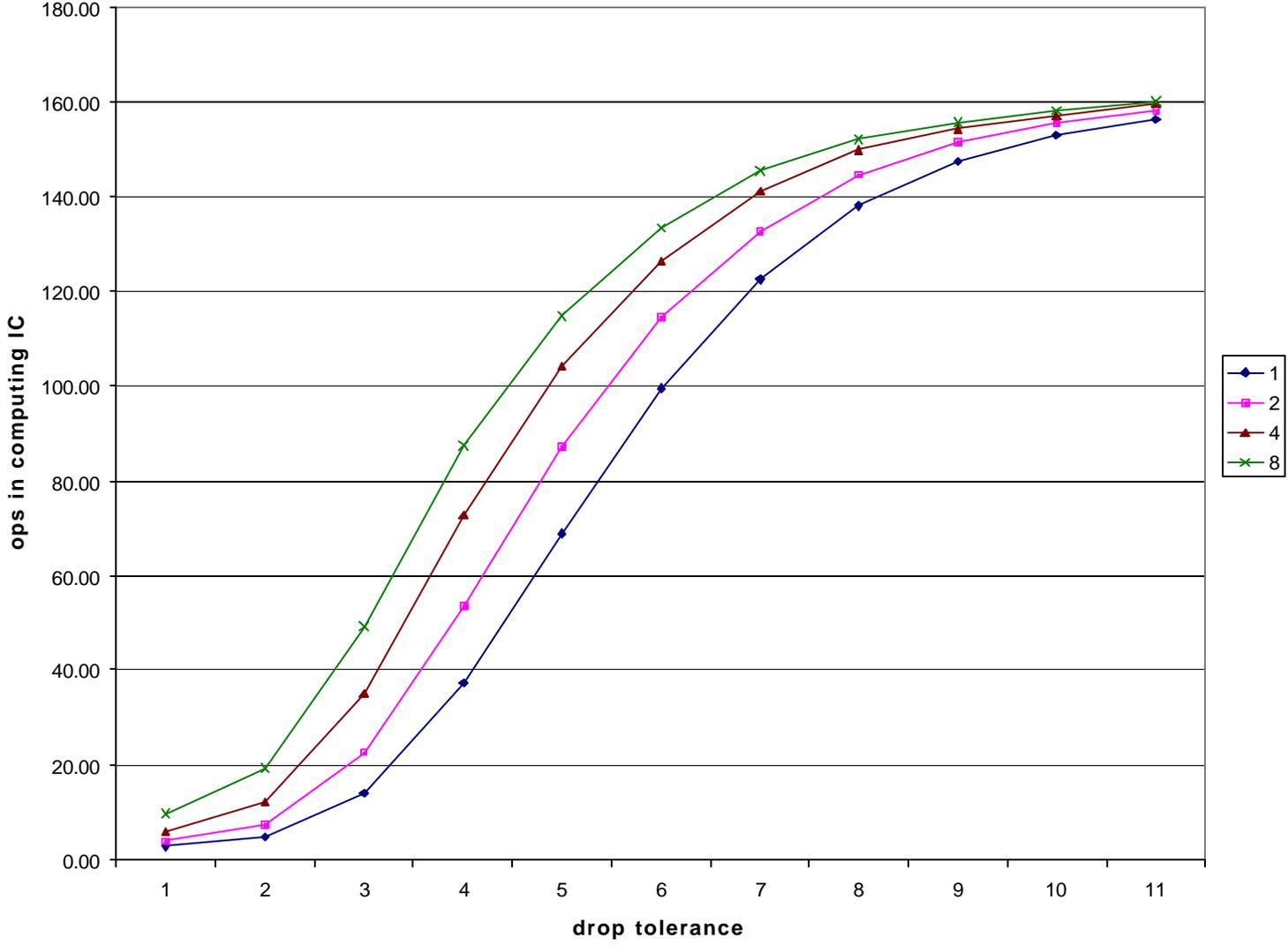
$$r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$$

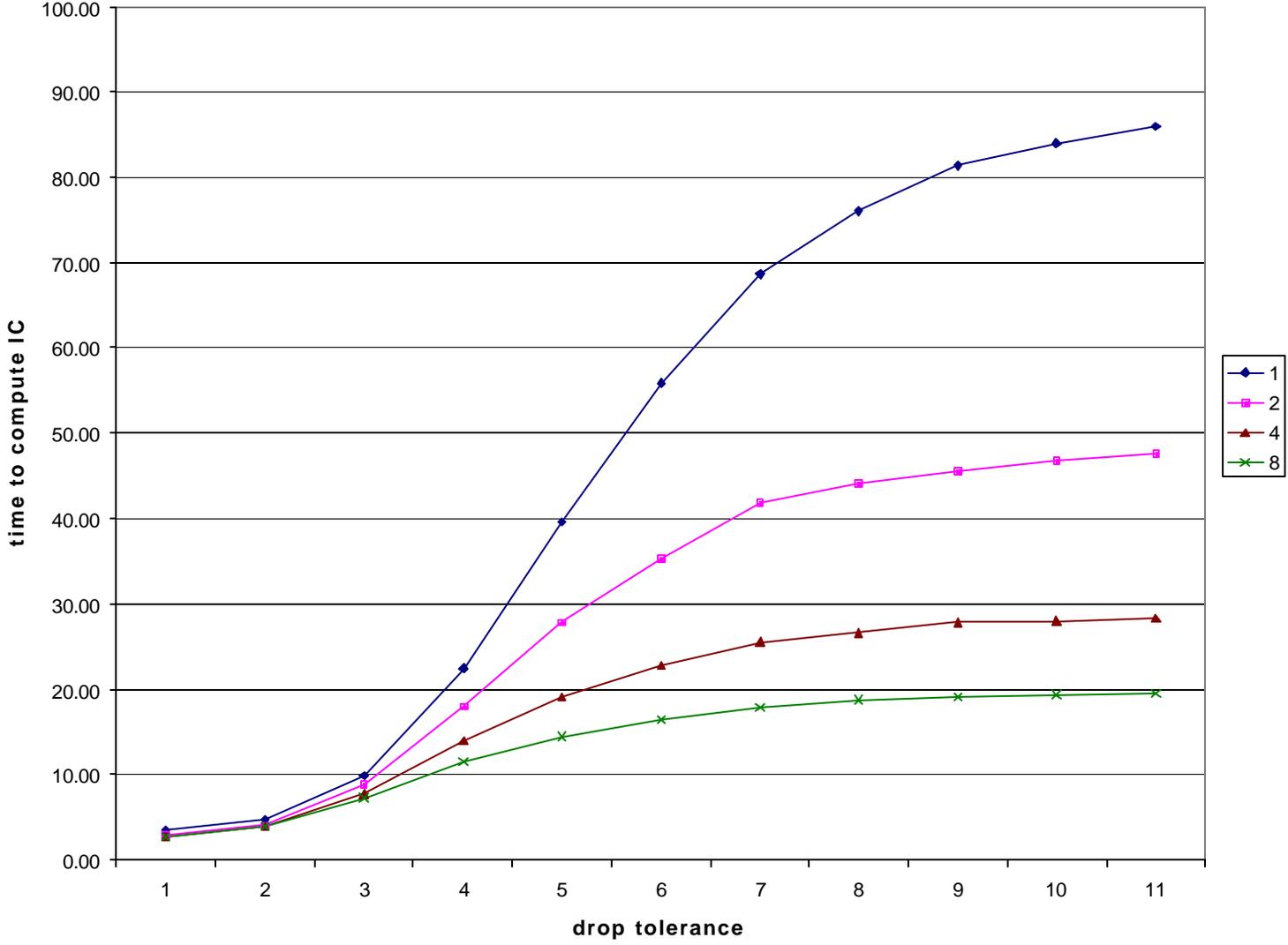
Check convergence; continue if necessary

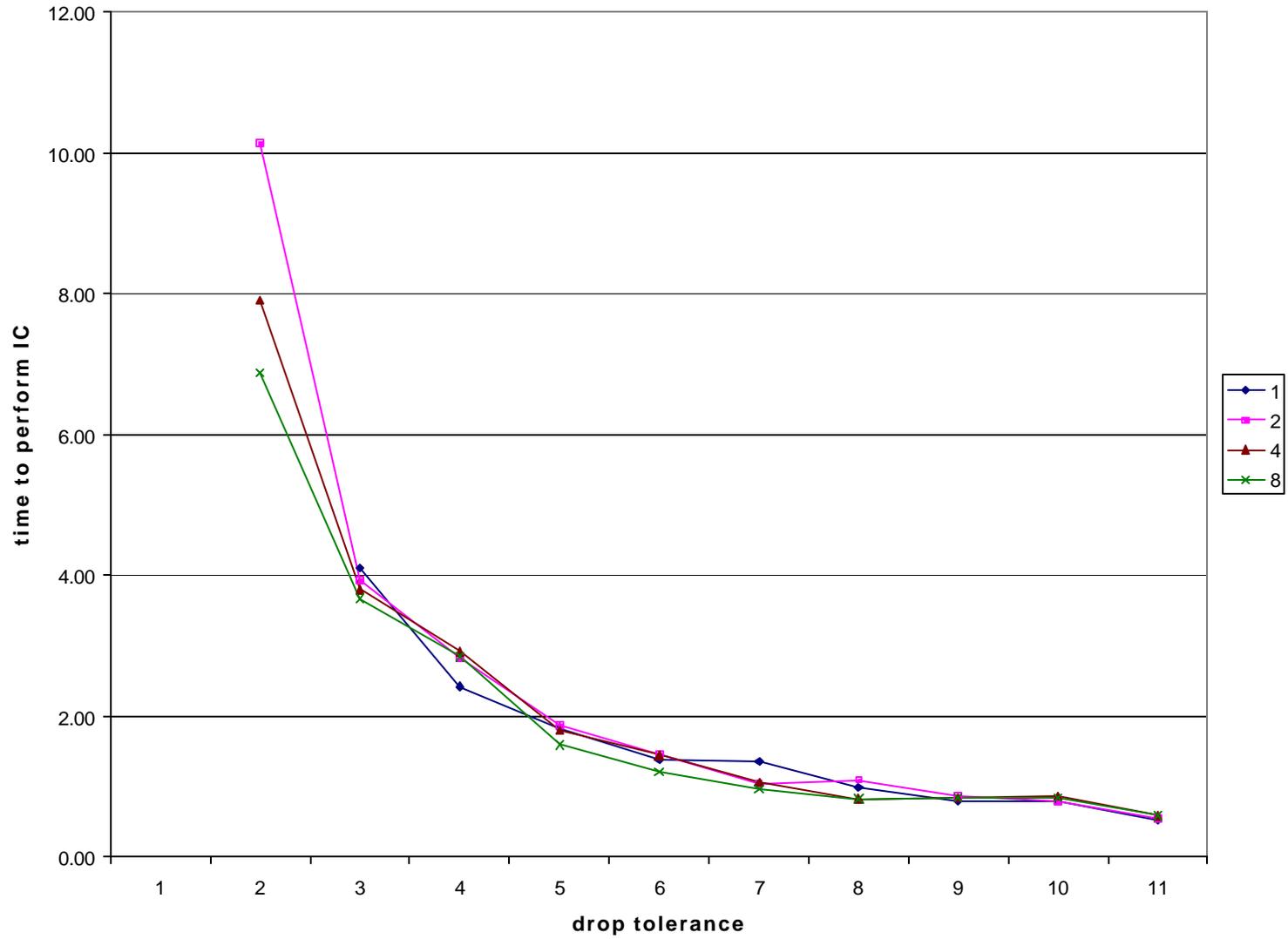
end

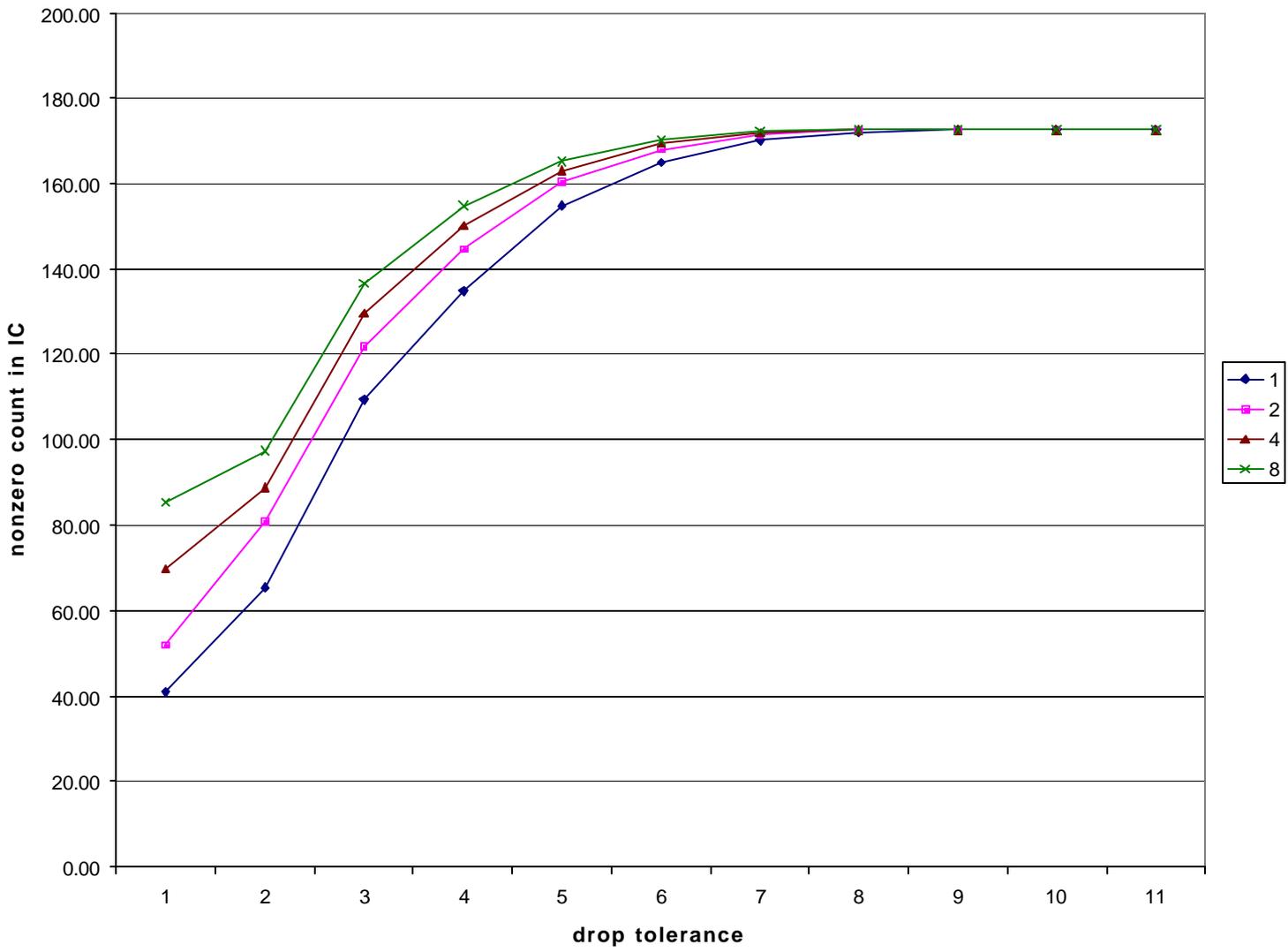
If converged, set  $x \leftarrow K^T x^{(i)}$ .

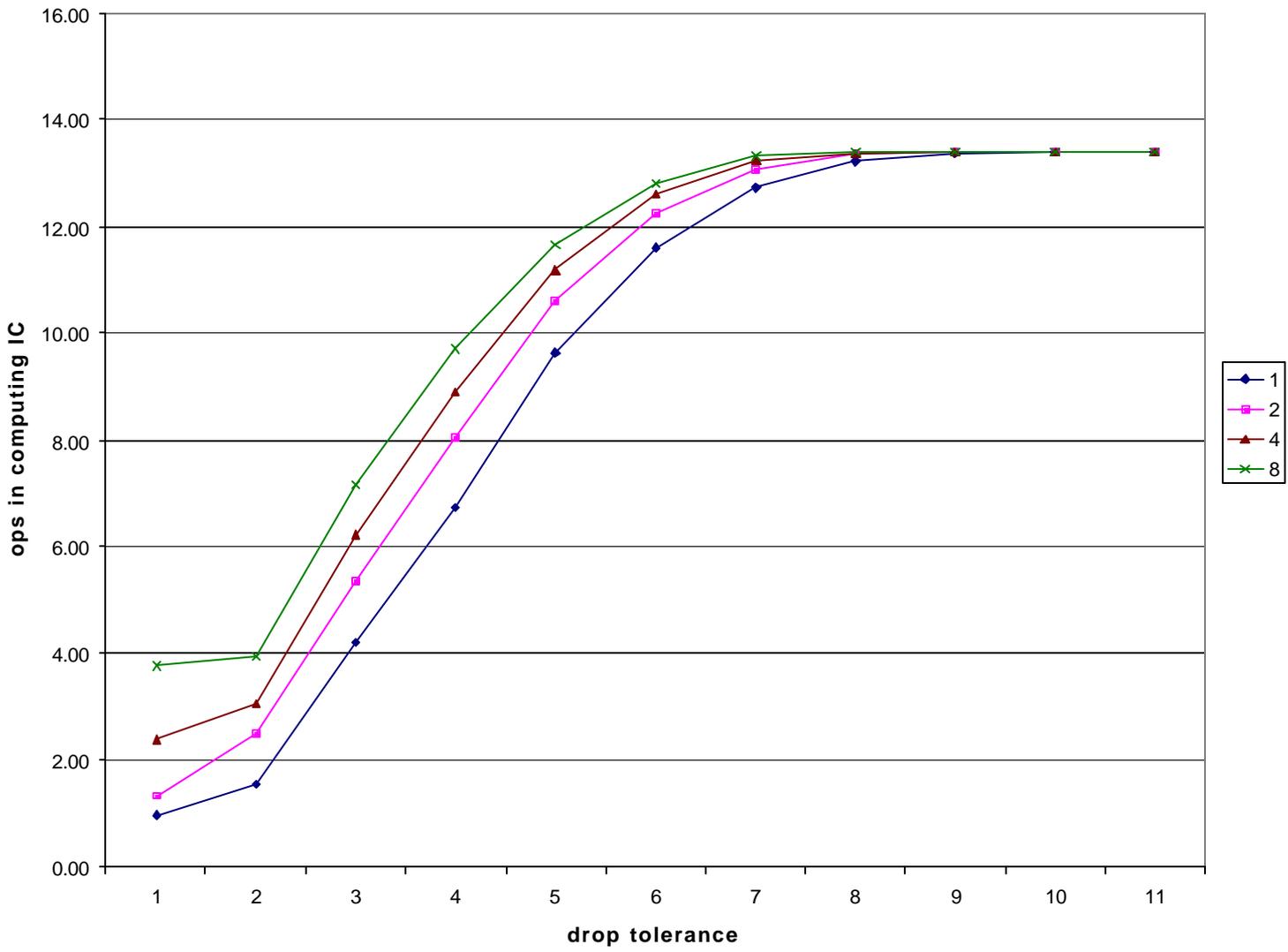


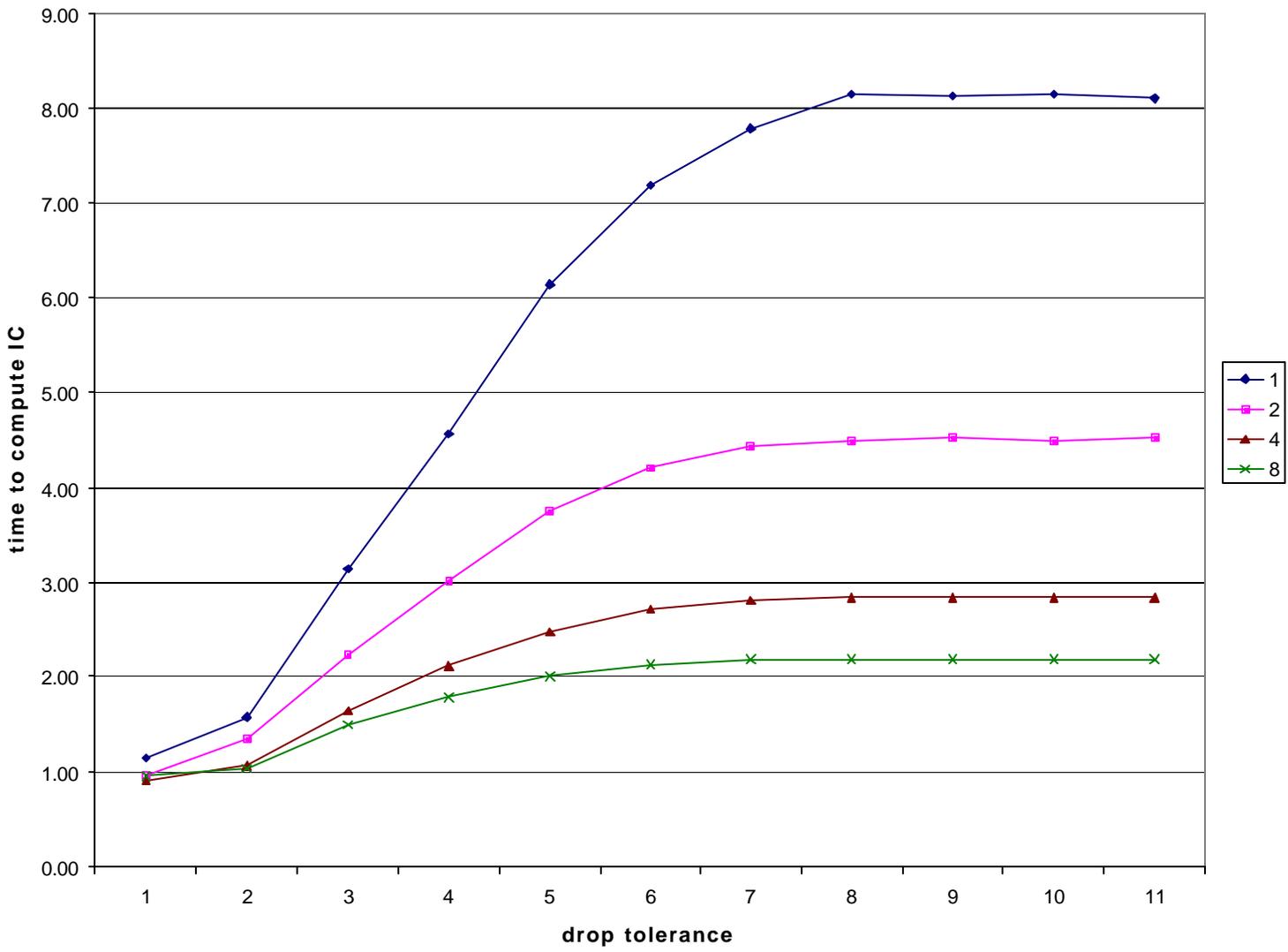


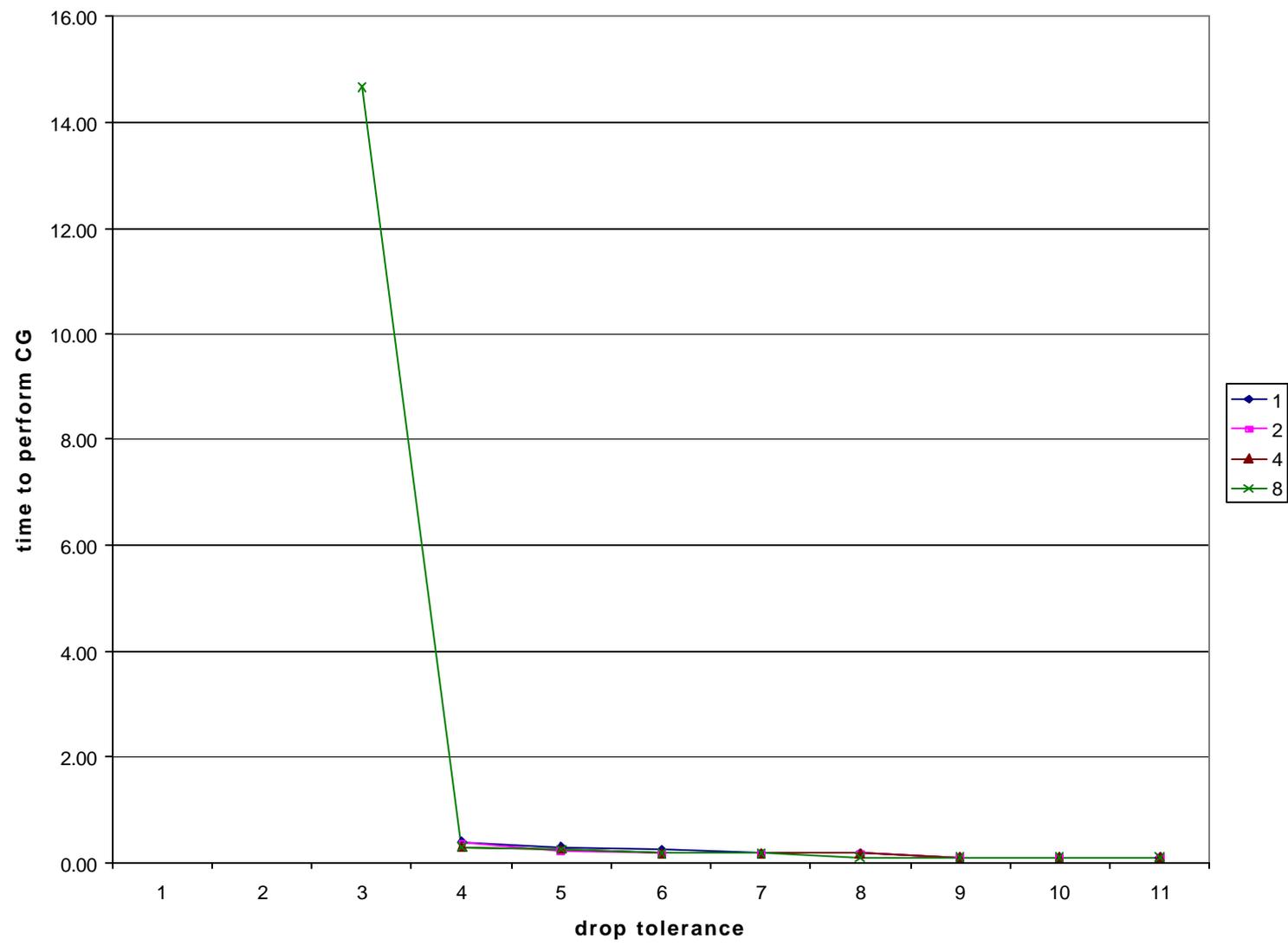












# Observations

---

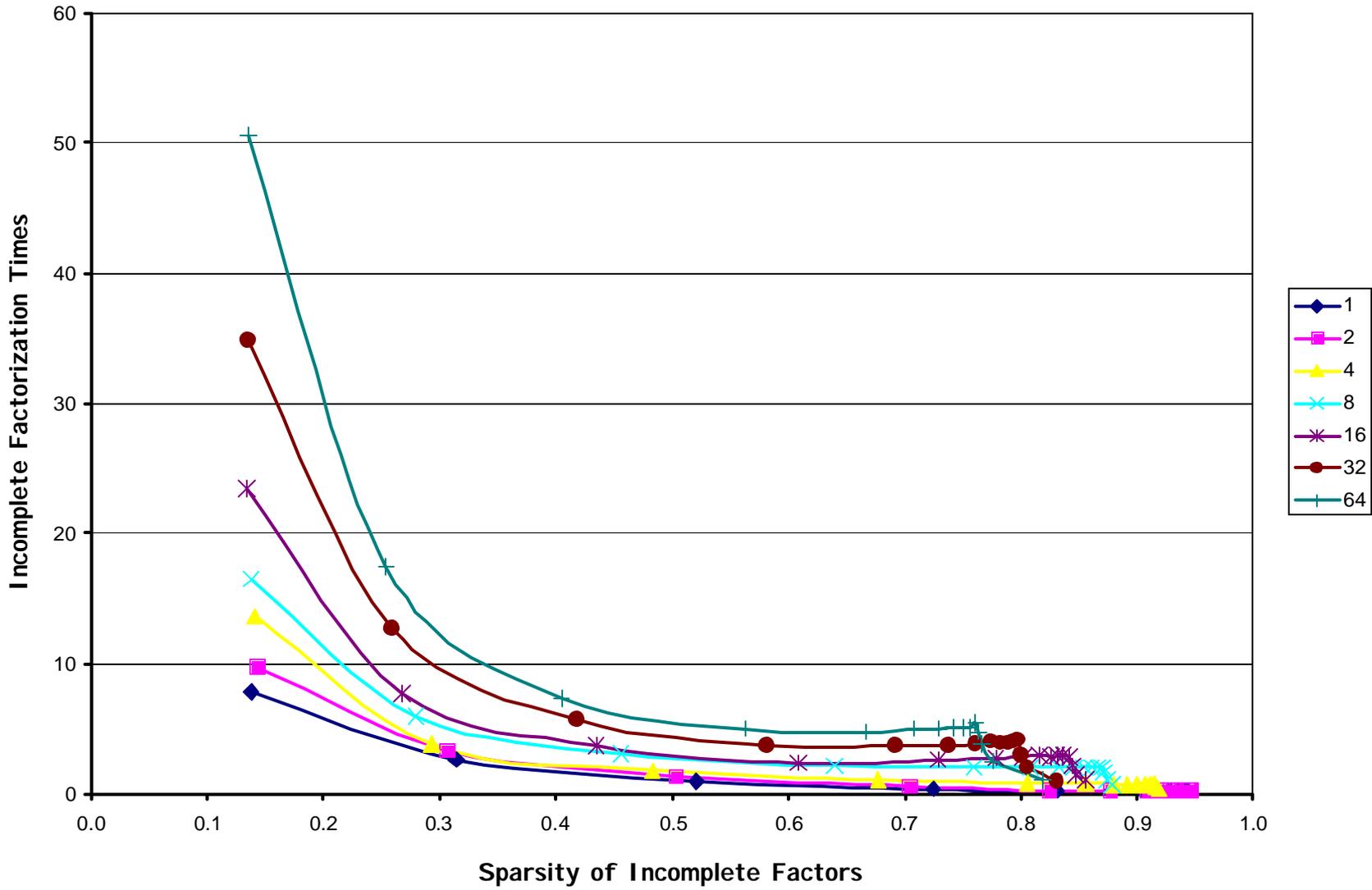
- ❑ Effect of increasing block size:
  - Increase fill and operation count, but improve quality of incomplete factor as a preconditioner.
  - Decrease time to compute incomplete factor.
  - Modest change in triangular solution time.

# Experimental Results

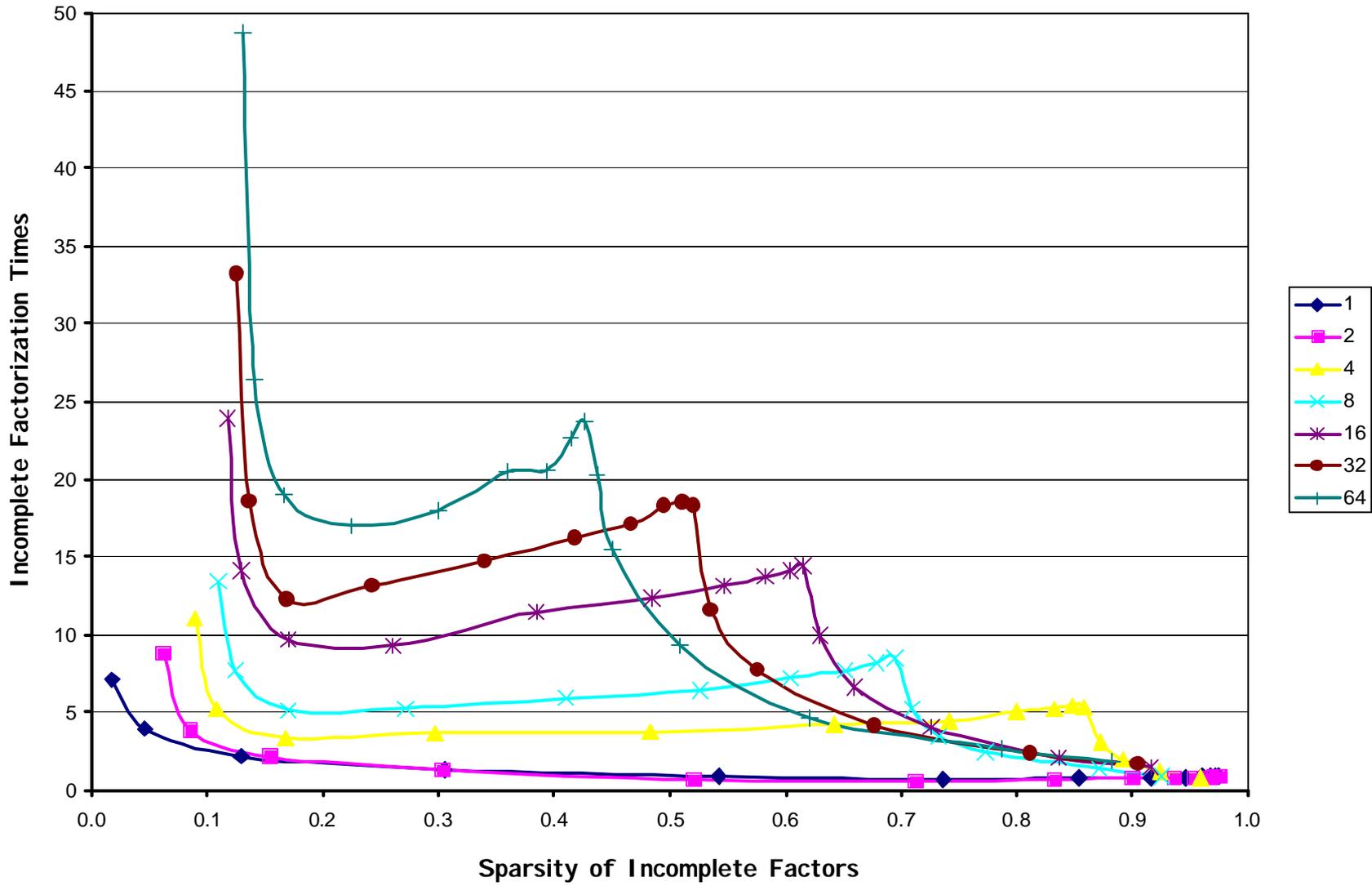
- ❑ Heat equations on 2-D and 3-D grids.
- ❑ Problems chosen so that number of nonzero entries per processor is approximately constant as number of processors increases.

$p$	N/2-D	N/3-D
1	40,000	27,000
2	73,441	42,875
4	138,384	74,088
8	252,004	125,000
16	467,856	195,112
32	891,136	314,432
64	1,679,616	531,441

# Incomplete Factorization Times for 2D Grids



# Incomplete Factorization Times for 3D Grids



# Preconditioned Conjugate Gradient Method

Given an initial guess  $x^{(0)}$ .

Compute the initial residual  $r^{(0)} = (Kb) - (KAK^T) x^{(0)}$ .

Set  $p^{(0)} = 0$ ;  $\rho_{-1} = 1$ .

for  $i = 1, 2, \dots$

$$\rho_{i-1} = [r^{(i-1)}]^T r^{(i-1)}$$

$$\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$$

$$p^{(i)} = r^{(i-1)} + \beta_{i-1} p^{(i-1)}$$

$$q^{(i)} = (KAK^T) p^{(i)}$$

$$\alpha_i = \rho_{i-1} / [p^{(i)}]^T q^{(i)}$$

$$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$$

$$r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$$

Check convergence; continue if necessary

end

If converged, set  $x \leftarrow K^T x^{(i)}$ .

need sparse triangular solutions

# Parallel Conjugate Gradient Method

- ❑ Need to solve 2 sparse triangular systems at every iteration of conjugate gradient.
- ❑ Can become the bottleneck in a parallel environment ...
  - Triangular solutions are sequential in nature because of the substitution process.
  - There is parallelism in sparse triangular systems, but too much synchronization and too few operations, resulting in poor scalability.

# Parallel Sparse Triangular Solutions

- ❑ Replace the substitution process in a triangular solution by a sequence of matrix-vector multiplications, which has a higher degree of parallelism.
  
- ❑ The extreme:
  - Explicitly invert a triangular matrix?
    - ◆ Explicit inverse of a sparse triangular matrix is typically dense.
    - ◆ Expensive to compute and expensive to store.
  
- ❑ Alternatives?

# Consider The Dense Case

□ Let  $T$  be a lower triangular matrix.

□ Solve  $Tx = b$ .

□ Suppose  $T$  is partitioned into a block  $2 \times 2$  matrix:  $T = \begin{bmatrix} T_{11} & 0 \\ T_{21} & T_{22} \end{bmatrix}$

□ Partition  $b$  and  $x$  accordingly:  $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ ;  $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

□ Then the solution  $x$  is given by

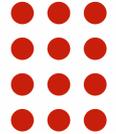
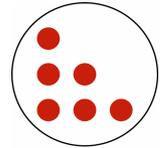
$$T_{11}x_1 = b_1 \Rightarrow x_1 = T_{11}^{-1}b_1$$

$$T_{21}x_1 + T_{22}x_2 = b_2 \Rightarrow x_2 = T_{22}^{-1}(b_2 - T_{21}x_1)$$

□ Can avoid triangular solution if we have  $T_{11}^{-1}$  and  $T_{22}^{-1}$  explicitly.

# Sparse Triangular Solutions

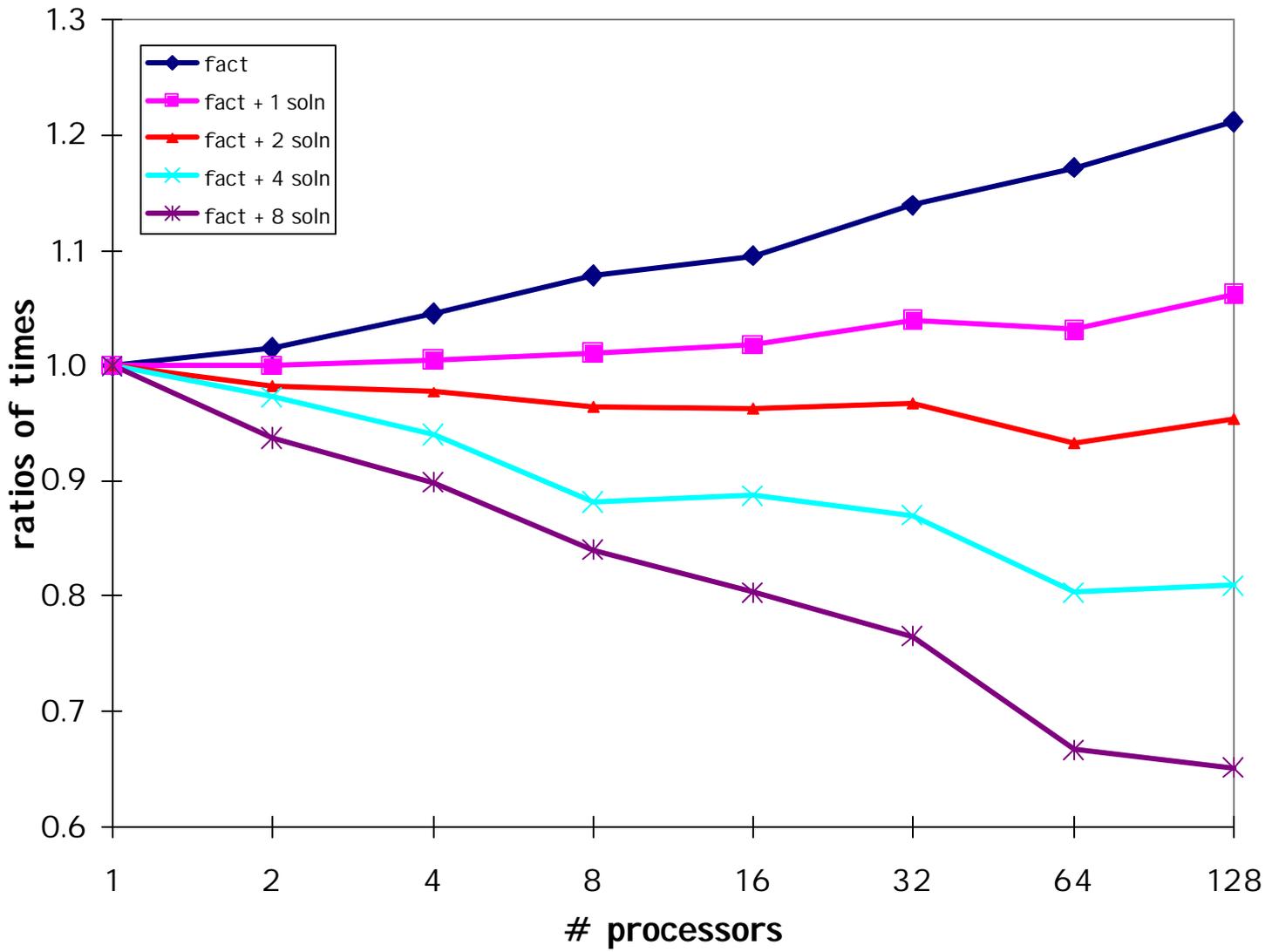
- Apply the idea to sparse triangular matrices that exhibit dense blocks on the diagonal, such as those from block (complete/incomplete) factorizations.
  - **Selective inversion** – Compute the inverse of each dense diagonal block [Raghavan '97].
  - Related work: Anderson/Saad '89; Alvarado/Schreiber '93; Heath/Raghavan '98; Raghavan '98; Teranishi/Raghavan/Ng '02.



# Performance of Selective Inversion

- ❑ Consider sparse symmetric positive definite matrices.
- ❑ Performance results based on complete Cholesky factorization.
- ❑ Consider cost of factorization + cost of multiple triangular solutions on a distributed-memory multiprocessor machine.
- ❑ Compare traditional substitution and selective inversion.
- ❑ Finite element matrices from a square grid.
  - Grid size varies with number of processors so that triangular solution work per processor is about constant.

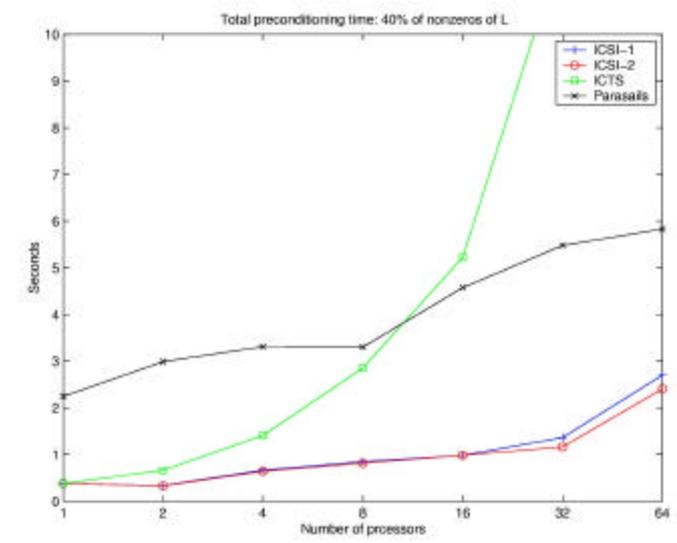
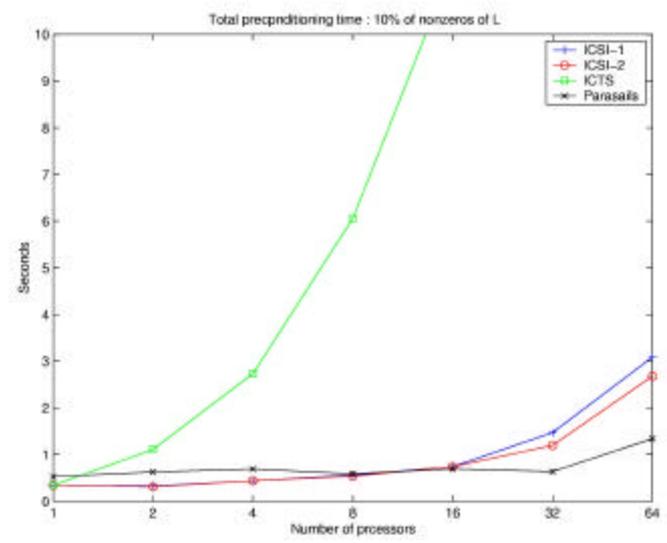
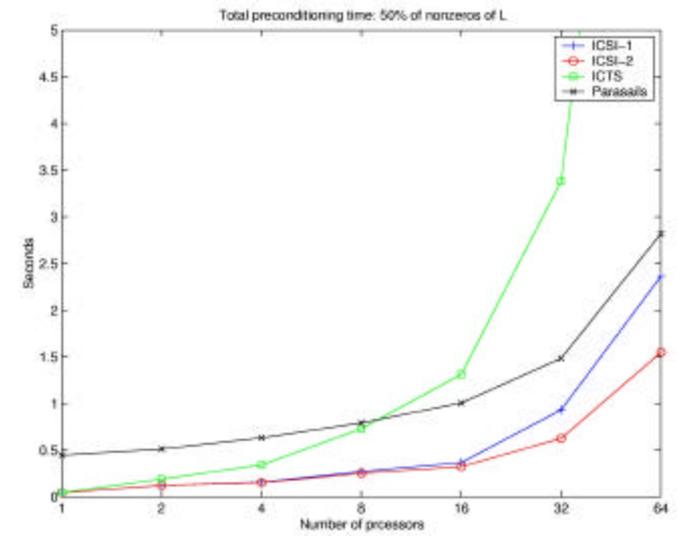
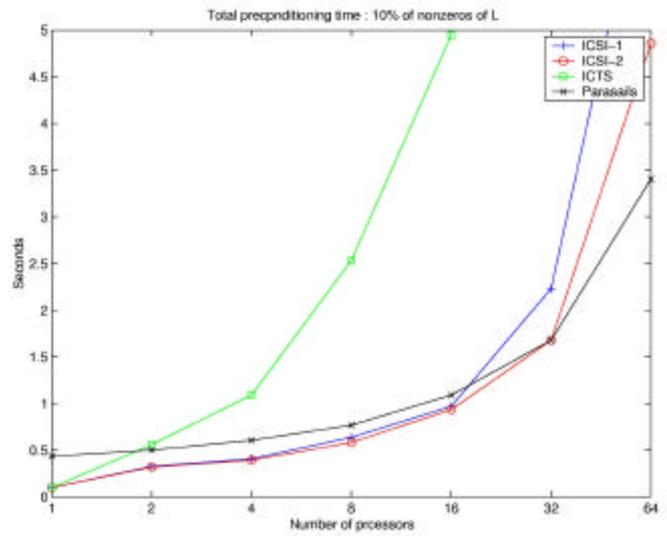
# Performance of Selective Inversion



# Performance of Selective Inversion

- ❑ Small penalty in computing inverses of diagonal blocks.
- ❑ Big improvements when solving multiple right-hand sides.
- ❑ The results show that selective inversion has the potential of improving the performance of multiple triangular solutions in iterative methods with incomplete factorizations as preconditioners.

# Performance of Selective Inversion



If one is willing to compute some sort of factorization of a sparse matrix as a preconditioner, then there is essentially no difference between sparse direct methods and preconditioned iterative methods.

# Project Idea

- ❑ Talked about SuperLU/SuperLU\_MT/SuperLU\_DIST last time.
  - Exploit dense blocks in supernodes.
- ❑ Incorporate selective inversion into sparse triangular solution.
  - Need to understand the data structure in SuperLU.
  - Need to identify the diagonal blocks.
  - Invert the diagonal blocks.
  - Rewrite the triangular solution in terms of inverses of the diagonal blocks.
- ❑ Both Sherry Li (XSLi@lbl.gov) and I (EGNg@lbl.gov) can help.