

ELMRES, An Oblique Projection Method for Solving Systems of Sparse Linear Equations

Gary Howell * Desmond Stephens †

January 7, 1999

1 Abstract

ELMRES constructs a basis for a Krylov subspace by the Hessenberg algorithm, requiring the same storage and half the operations as GMRES. The backward error associated with the Hessenberg matrix is in practice small and the basis for the Krylov subspace well-conditioned. ELMRES is amenable to parallel implementation.

2 Introduction

ELMRES (Elementary Residual Method) is inspired by the celebrated GMRES algorithm [5]. Both are Krylov subspace projection methods. GMRES uses an orthogonal basis. The ELMRES basis is nonorthogonal but in practice well-conditioned. GMRES can be accomplished by orthogonal reduction of a matrix to the first m columns of a Hessenberg matrix by Householder transformations. ELMRES corresponds to reduction to a similar Hessenberg matrix by elementary similarity transformations.

The cost to produce m columns for GMRES is $2m^2n$ flops (modified Gram-Schmidt without reorthogonalization). The cost for m columns for ELMRES is $m^2n - n^3/3$ flops. In preliminary investigations, the convergence behavior of GMRES and ELMRES is very similar. Both algorithms often demonstrate superlinear convergence for large enough bases. We hypothesize that Krylov space residual methods give similar convergence properties whenever the respective bases for the Krylov space are well-conditioned.

*Florida Institute of Technology, Melbourne, Florida 32901, e-mail howell@zach.fit.edu

†Department of Mathematics, Florida A&M, Tallahassee, FL e-mail dstephen@cis.famu.edu

In either case, large bases often require distributed data as a means of keeping bases in random access memory. Parallel GMRES is fine grain, requiring parallel inner products. The number of processors for parallel GMRES is therefore constrained by communication costs. Parallel ELMRES is medium grain, requiring a triangular solve. ELMRES can use more processors efficiently.

3 The ELMRES Algorithm

ELMRES can be thought of as an oblique variant of GMRES. Briefly, the m th basis vector for the GMRES algorithm is formed as follows.

```

 $w_m = A * v_m$ ; /* enlarge subspace
For  $i = 1 : m$ ,
     $h_{im} = w_m^T * v_i$ ; /* modified Gram-Schmidt
     $w_m = w_m - h_{im} * v_i$ ;
End;
 $h_{m+1,m} = \|w_m\|_2$ ;
 $v_{m+1} = w_m / h_{m+1,m}$  /* normalization step
At the  $m$ th step
 $y_m$  minimizes  $\|\beta e_1 - \tilde{H}_m y\|_2$ 
 $x_m = x_0 + V_m y_m$  ;

```

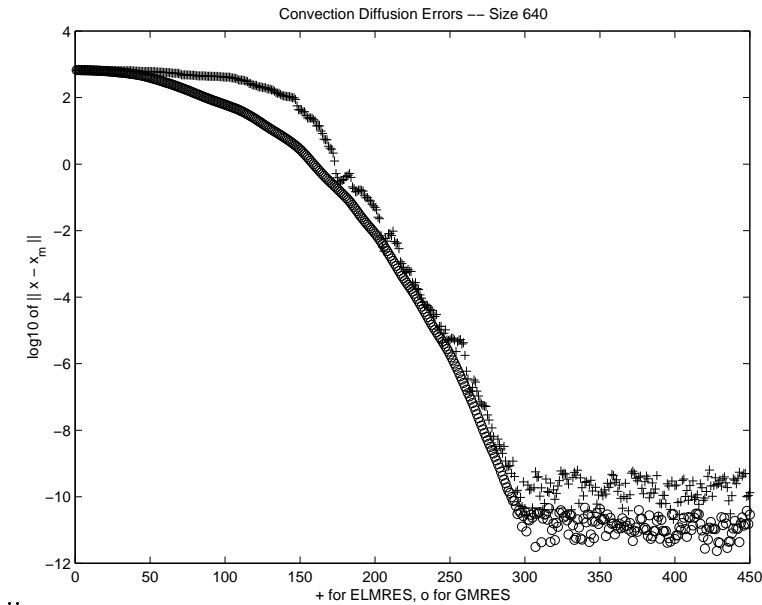


Figure 1: Errors of GMRES and ELMRES for a Convection Diffusion Problem

The m th step of ELMRES is as follows.

```

 $y_m = A * l_m; /*$  enlarge subspace
For  $i = 1 : m,$ 
     $h_{im} = y_m(i);$ 
     $y_m = y_m - h_{im} * l_i;$ 
    /* produce a zero in  $y_m(i)$ 
End;
 $h_{m+1,m} = \|y_m\|_\infty; /*$  partial pivot
 $l_{m+1} = y_m/h_{m+1,m} /*$  normalization step
At the  $m$ th step
 $z_m$  minimizes  $\|\beta e_1 - \tilde{H}_m y\|_2$ 
 $x_m = x_0 + L_m y_m ;$ 

```

As with GMRES, $\|\beta e_1 - \tilde{H}_m y\|_2$ decreases monotonically as m increases. For GMRES in exact arithmetic, $\|\beta e_1 - \tilde{H}_m y\|_2 = \|b - Ax_m\|_2$. For ELMRES, this is no longer the case. The data in Figure 1 was produced by a Matlab implementation.

4 Parallel Implementation of GMRES

Though we have not yet implemented ELMRES in parallel, we can use a simple model of parallel communication to predict its performance. For an example comparison of parallel performance of ELMRES and GMRES, we have made assumptions appropriate to a network of workstations running under PVM or MPI, connected by an ethernet bus. For a bus architecture broadcasts are performed in parallel, while gathers are sequential. We model communication between processors as

$$t_{communication} = t_{latency} + (\text{number of floats}) \times (\text{time/floating point}).$$

For our model implementations, we partition a matrix A and a matrix V of basis vectors by rows. The columns of V are basis vectors of a Krylov subspace. Store V and matrix A with the same partition. This is the natural distribution of data as it allows parallel computation of inner products in orthogonalization of a new basis vector.

We assume that GMRES uses modified Gram-Schmidt orthogonalization without orthogonalization. Block orthogonalizations parallelize more easily, but as in nonmodified Gram-Schmidt, linear independence of basis vectors tends to be lost, increasing the necessary basis size.

For the m th step

1. Broadcast a new vector v_m to each processor.
 - (a) Perform a matrix vector multiply on individual processors. Since each processor has all of some rows, computations are parallel.

2. For $i = 1 : m$,
 - (a) Each processor communicates its bit of $w_m^t v_i$ (parallel dot product) to a root processor.
 - (b) The root broadcasts $\alpha = w_m^t v_i$ to all, then perform in parallel $w_m - \alpha v_i$.
3. End Do

If time to start a message is significant, parallel dot products are efficient only with 10-20 thousand vector elements per processor. The number of messages is proportional to basis size, limiting basis size and requiring restarts. Frequent restarts may prevent and/or slow convergence.

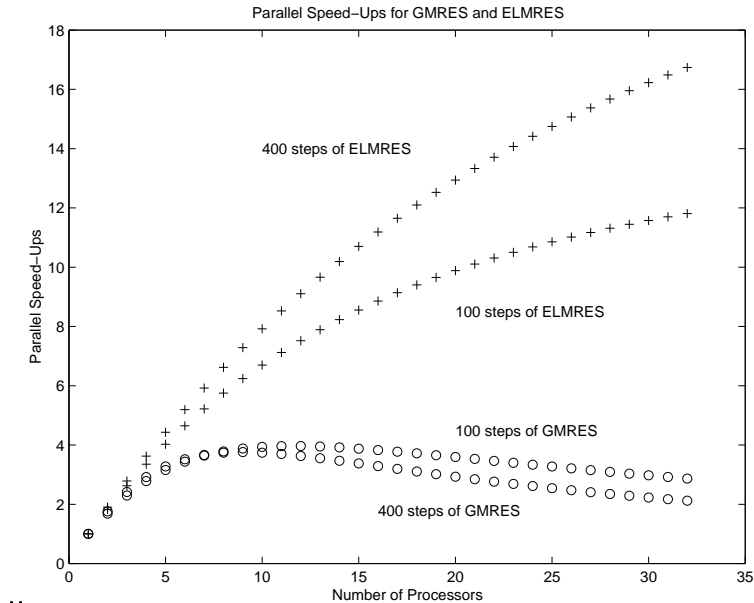


Figure 2: Parallel Speedups of GMRES and ELMRES

5 Parallel ELMRES

Suppose we partition A and the lower triangular matrix L of basis vectors by rows. For the m th step

1. Broadcast the new vector l_m to each processor
 - (a) Perform a matrix vector multiply on individual processors.

2. Broadcast $h_{1m} \dots h_{mm}$ to all processors
3. Each processor sends its candidate pivot to root.
4. Notify winning row
5. Send winning row of A to root.

Parallel ELMRES is medium grain. The number of messages is almost independent of the number of basis vectors, Parallel implementation is efficient even with high latency and relatively few rows per processor. Since many processors can be used, their available memory enlarges feasible problem and basis size. A larger basis typically means fewer restarts and more reliable convergence.

For the results of Figure 2 assume

- Latency = 3000 computations (time to start message)
- Time to transmit one floating point = 5 computations.
- 100 nonzero entries per row of a 100K matrix.

6 Hessenberg's Algorithm

ELMRES creates basis vectors by Hessenberg's algorithm. If it is carried out for n steps then in exact arithmetic the same Hessenberg matrix is obtained as by elementary similarity transformations. Wilkinson presents the Hessenberg algorithm as a Krylov subspace method and also shows how to incorporate partial pivoting [7].

Elementary similarity transformations are of the form $L = I - l_i e_i^t$, $L^{-1} = I + l_i e_i^t$. Multiplication on the left by L is a rank one update. Multiplication on the right of A by L^{-1} increments a column of A by Al_i . Hessenberg's algorithm performs the matrix vector multiply with the original (perhaps sparse) matrix. It does not seem to have been noted that Hessenberg's algorithm reduces a sparse matrix to similar Hessenberg form in $2/3n^3$ flops as opposed to the $5/3n^3$ flops required for the EISPACK [6] implementation of reduction to similar Hessenberg form by elementary similarity transformations. Using Householder transformations as in LAPACK dgehrd [1] requires $10/3n^3$ flops .

After many years of numeric experience, it is well known that conversion by elementary similarity transformations to Hessenberg form is in practice numerically stable. Wilkinson's error analysis neglects associativity and also applies to Hessenberg's algorithm. The error analysis is related to that for LU decomposition. For LU decomposition in floating point arithmetic,

$$LU = A + \Delta A, \quad |\Delta A| \leq \gamma_n |L| |U|$$

Here $\gamma_m = \frac{nu}{1-nu}$ where u is the largest number for which $fl(1+u) = 1$ and $|L|, |U|$ are the matrices formed by taking the absolute values of elements of L and U respectively [4]. For Hessenberg's algorithm, the following is a slight refinement of Wilkinson's result.

$$H + \Delta H = L(A + \Delta A)L^{-1},$$

$$|\Delta A| \leq \gamma_n |A|, \quad |\Delta H| \leq \gamma_n |H|$$

Large errors occur only in the rare event that element growth is large. The bound indicates that large errors can be detected by observing the size of elements of H . For further discussion, see [2], [3].

7 Conclusions

Further work is found in the dissertation of Desmond Stephens, expected to be available in May 1999.

References

- [1] E. ANDERSON, Z. BAI, C. BISCHOF, J. DEMMEL, J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, S. OSTROUCHOV, D. SORENSON, *LAPACK Users' Guide*, SIAM, Philadelphia (1992).
- [2] P. A. BUSINGER, *Reducing a Matrix to Hessenberg Form*, Math. Comp. v. 23, pp. 819-821, 1969.
- [3] , A. A. DUBRULLE, *Block Gauss Reduction to Hessenberg Form*, SIAM J. Sci Stat Comput. Vol 12, No. 5, pp. 1245-1253, September 1991.
- [4] N. J. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia (1996).
- [5] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston (1995).
- [6] B.T. SMITH ET AL., *Matrix Eigensystem Routines—EISPACK Guide*, 2nd ed., Springer-Verlag, New York, 1976.
- [7] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.