Solving Nonlinear Equations with Newton-Krylov Method Based on Automatic Differentiation [*]

Wei Xu[1] and Thomas F. Coleman[2]

[1] Department of Mathematics, Tongji University
Shanghai, China, 200092

[2] Department of Combinatorics and Optimization
University of Waterloo
Waterloo, On. Canada, N2L 3G1.

[1] wdxu@tongji.edu.cn
[2] tfcoleman@uwaterloo.ca

May 27, 2011

Abstract

The Jacobian-free Newton-Krylov method is widely used in solving nonlinear equations arising in many applications. However, an effective preconditioner is required each iteration and determining such may be hard or expensive. In this paper, we propose an efficient two-sided bicoloring method to determine the lower triangular half of the sparse Jacobian matrix via automatic differentiation. Then, with this lower triangular matrix, an effective preconditioner is constructed to accelerate the convergence of the Newton-Krylov method. We demonstrate that the proposed bicoloring approach is significantly more effective than the one-sided coloring method proposed in [13] and yields an effective preconditioning strategy.

Key words: Automatic differentiation, preconditioner iterative methods, nonlinear equation solvers, Newton method, intersection graphs, graph coloring.

1 Introduction

The multiple dimensional zero-finding problem is

$$\text{Solve } F(x) = 0, \tag{1.1}$$

[*]This work was supported in part by the Ophelia Lazaridis University Research Chair (held by Thomas F. Coleman) and the National Sciences and Engineering Research of Canada.

where $F(x) = (f_1(x), f_2(x), \cdots, f_n(x))^T$ and each $f_i(x)$ maps a real n-vector x into a scalar in a continuous and differentiable way. The Newton method is a typical method for solving this problem. The core of the Newton method is to successively solve linear systems, that is

$$\text{Solve} \quad J(x)d_x = -F(x), \tag{1.2}$$

where $J(x)$ is an n-by-n Jacobian matrix of $F(x)$, i.e., $J(x) = \left(\frac{\partial f}{\partial x_i}\right)_{n\times n}$. However, when the problem size n is large, evaluating the Jacobian matrix in each iteration of the Newton method is expensive. Thus, approximations of the Jacobian matrix are often used to replace the exact Jacobian matrix in the Newton iteration [3, 30]. Although the computational cost of each iteration is thereby reduced, more iterations are usually required to converge to the root of (1.1). However, the Jacobian matrix itself is structured in many applications, so it may not be necessary to form the Jacobian matrix $J(x)$ itself explicitly in each iteration. In 2004, Knoll and Keyes [23] reviewed the Jacobian-free Newton-Krylov method (JFNK) for solving the nonlinear equations (1.1); this approach combines a matrix-free approach with nonsymmetric Krylov subspace methods. The Krylov subspace method was first proposed in 1950 [22]. This method is a projection, or generalized projection, method for solving a linear system $Ax = b$ using the Krylov subspace K_j,

$$K_j = \text{span}(r_0, Ar_0, A^2 r_0, \cdots, A^{j-1} r_0),$$

where $r_0 = b - Ax_0$. There are a variety methods falling into the Krylov taxonomy [4, 21, 17, 28, 29]. One of the widely used method is the Generalized Minimal RESidual method (GMRES), which is an Arnoldi-based method. In the GMRES method, the Arnoldi basis vectors form a trial subspace. In each iteration of the GMRES method, a matrix-vector product is required to create a new trial vector. The method is terminated based on an estimation of the residual. Thus, in the GMRES method, it is not necessary to form the parameter matrix A explicitly. The method only requires the matrix-vector product, i.e., Ax.

In order to construct a matrix-vector product $J(x) \cdot v$ for (1.2), we can employ the finite difference method. For a nonlinear function $F(x) : \mathbb{R}^n \to \mathbb{R}^n$, its corresponding Jacobian-vector product can be approximated as

$$J(x) \cdot v = \frac{F(x + \epsilon v) - F(x)}{\epsilon}, \tag{1.3}$$

where ϵ is a small positive number. Thus, combining the GMRES method with finite differences, the JFNK method is obtained for solving nonlinear equations. In practice, this method works well in solving problems in fluid dynamics, power systems, plasma physics and so on [24, 6].

However, there are two problems with the JFNK method.

1. How should the preconditioner used in the GMRES procedure be chosen? It is well known that the preconditioner reduces the number of GMRES iterations by clustering

eigenvalues of the iteration matrix. In each Newton iteration of the JFNK method, the Jacobian matrix $J(x)$ is updated; how can we update the corresponding preconditioner effectively and efficiently? In [23], a few preconditioner techniques were proposed, but each is application dependent. Can we find an efficient way to update the preconditioner corresponding to the updated Jacobian matrix?

2. The Jacobian-vector product approximation (1.3) introduces some error into the Newton system (1.1). It is known that (1.3) is a first-order approximation of the Jacobian-vector product. We can increase its accuracy to second order by

$$J(x) \approx \frac{F(x+\epsilon v) - F(x-\epsilon v)}{\epsilon},$$

but this technique doubles the cost of forming $J(x)v$. Can we find an economic way to evaluate $J(x) \cdot v$ accurately?

Regarding the first question, Tůma et al proposed a matrix-free environment to construct updated preconditioners for a sequence of matrices based on partial matrix estimation [13, 15, 16]. In those papers, only the triangular parts of the sequence of matrices are used to update the preconditioners. For solving the nonlinear equations (1.1), the Jacobian-vector product approximation (1.3) is employed to update the lower triangular part of the Jacobian matrix with a pre-determined sparsity pattern. Their experiments illustrate the effectiveness of their new preconditioners.

In this paper, we will not only answer the second question, evaluating the Jacobian-vector product $J(x)v$ exactly with the less computational cost than the finite difference method, but also will evaluate the exact product $w^T J(x)$ based on the automatic differentiation (AD)[19]. We then generalize Tůma's preconditioner construction method from one-sided color updating to two-sided color updating. It will turn out that two-sided coloring updating will be much more efficient than the one-sided coloring updating, especially when there are some dense rows in the lower triangular part of the Jacobian matrix.

The rest of the paper is organized as follows. Section 2 introduces some basic ideas and implementations of Tůma's preconditioner with a pre-determined sparsity pattern. In Section 3, we will propose our method to update the lower triangular part of the Jacobian matrices by the bi-coloring graph technique with the automatic differentiation. Then, some numerical results will show the efficiency of the bi-coloring updating technique and the performance of solving nonlinear equations in Section 4. Finally, we present concluding remarks in Section 5.

2 Preconditioner Construction

In [15, 16], the triangular preconditioner updates for nonsymmetric linear systems are defined based on the difference between the matrix from the first linear system and the current system matrix of the sequence. Suppose J is the Jacobian matrix of the first Newton system and $\bar{J}$ is the current system Jacobian matrix. If LDU is the incomplete triangular decomposition [20] of J and $B = J - \bar{J}$ is the difference matrix, then the basic triangular updated preconditioner, P, of the current Jacobian system can be constructed as

$$P \equiv (LD - \mathrm{tril}(B))U, \qquad \text{or} \qquad P \equiv L(DU - \mathrm{triu}(B)), \tag{2.4}$$

where $\mathrm{tril}(\cdot)$ and $\mathrm{triu}(\cdot)$ are the lower triangular and upper triangular part of a matrix, respectively. Thus, we will solve a preconditioned linear systems $P^{-1}J(x)d_x = -P^{-1}F(x)$, instead of (1.2). If we want to construct the preconditioner as in (2.4), the lower triangular or upper triangular part of the current Jacobian matrix $\bar{J}$ has to be evaluated. For simplicity, we focus on $(LD - \mathrm{tril}(B))U$ in this paper. The following theorem shows the quality of the preconditioner if $(\overline{DU - B})$, the approximation to $(DU - B)$, is favorably chosen.

Theorem 2.1 *[13] Assume that $LDU+E = J$ for some error matrix E and let $\|\overline{U^{-1}D^{-1}B}\|_2 \leq 1/c < 1$ where $\|\cdot\|_2$ denotes the Euclidean norm. Further assume that the singular values σ_i of*

$$(I - L)B + L(\overline{DU - B} - (DU + L^{-1}E - B))$$

satisfy

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_t \geq \delta \leq \sigma_{t+1} \geq \cdots \geq \sigma_n,$$

for some integer t, $t \ll n$ and some small $\delta > 0$, where $(\overline{DU - B})$ is an easily invertible approximation matrix of $(DU - B)$. Let $(\overline{DU - B})$ have nonzeros main diagonal, and $D = diag(d_1, \cdots, d_n)$. Then there exist matrices F and Δ such that

$$(\overline{DU - B})^{-1}L^{-1}\bar{J} = I + \Delta + F,$$

with rank(Δ)$<t$ and

$$\|F\|_2 \leq \frac{c}{c-1}\max_i \frac{\delta}{|d_i|}\|L^{-1}\|_2\|U^{-1}\|_2.$$

In [13], a solver for the diagonal partial coloring problem is proposed to approximate the lower triangular part of B based on the Jacobian-vector product approximation Jv via the finite difference method. In the numerical experiments, a drop tolerance for each row of the testing sparse matrix is used where entries less than a tolerance are set to zero, so that the number of matrix-vector multiplications can be reduced significantly for some cases. The algorithm to form preconditioners update for the Newton system can be written as follows.

Algorithm 1 *Consider problem (1.2), the preconditioner update for the Newton system at each iteration can be constructed in the following steps. The start point is x_0.*

1. *Estimation. Estimate $J(x_0)$ with the finite difference method.*

2. *Initial factorization. Factorize $J(x_0)$ by the incomplete LU decomposition*

$$J(x_0) \approx LDU.$$

```
3. Sparsification. Filtrate $J(x_0)$ to get its sparsification pattern $S(J(x_0))$.

4. Coloring. Get a coloring pattern of $S(J(x_0))$ via solving a partial coloring problem
   based on one-sided coloring.

5. for $i = 0 : n$
      Estimate the lower triangular part of $J(x_i)$ via the coloring pattern with the
      matrix-vector product approximation.
      Compute the difference matrix tril($B$) and construct the preconditioner from (2.4).
   end
```

The key component of Algorithm 1 is the one-sided coloring pattern of $S(J(x_0))$ and estimation of the lower triangular part of $J(x_i)$ via the coloring pattern with the matrix-vector product approximation. When the matrix $J(x_i)$ has dense columns in its lower triangular part, the one-sided coloring pattern with the matrix-vector product, Jv, is an efficient way to estimate $\text{tril}(J(x_i))$. However, when there are some dense rows, the matrix-vector product Jv is not suitable.

Example 1 shows two extreme cases for these two situations. The left matrix, $J^{(1)}$, is an upper arrow matrix with a dense column in its lower triangular part. The right matrix, $J^{(2)}$, is a lower arrow matrix with a dense row in its lower triangular part. As we see, in order to estimate the lower triangular parts of $J^{(1)}$ and $J^{(2)}$, the corresponding $V^{(i)}$ for $J^{(i)}$ $(i = 1, 2)$ are totally different. To estimate $\text{tril}(J^{(2)})$, five vectors are required while $J^{(1)}$ only requires two. Thus, we have to find a way to deal with the dense row in $J^{(2)}$. In [13], Tůma uses filtration, which removes small nonzeros in each row. However, how to choose a drop tolerance, which guarantees a good preconditioner for the Newton system, is not discussed in [13]. In the next section, we will propose a partial bi-coloring graph method with the matrix-vector products Jv and $w^T J$ so as this technique can be effective when there are both dense columns and rows in the $\text{tril}(J)$.

Example 1 *Examples for matrix estimation through matrix-vector products.*

$$J^{(1)} = \begin{bmatrix} X & X & X & X & X \\ X & X & & & \\ X & & X & & \\ X & & & X & \\ X & & & & X \end{bmatrix} \qquad J^{(2)} = \begin{bmatrix} X & & & & X \\ & X & & & X \\ & & X & & X \\ & & & X & X \\ X & X & X & X & X \end{bmatrix}$$

$$V^{(1)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \qquad V^{(2)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

3 Partial Bicoloring Matrix Update

The bicoloring technique was proposed by Coleman and Verma in [9, 10] to efficiently determine the Jacobian matrix J using AD techniques. In this paper, we will generalize their bicoloring ideas to update the lower triangular part of Jacobian matrix J. For a nonlinear function $F(x) : \mathbb{R}^n \to \mathbb{R}^n$, the first derivative is the Jacobian matrix $J \in \mathbb{R}^{n\times n}$. In large scale problems, the matrix $J(x)$ is often sparse and it is important to exploit its sparsity and determine its lower triangular part at a given point x. In this section, we are concerned with the efficient computation of the lower triangular part of J via automatic differentiation techniques.

Given an arbitrary n-by-t_V matrix V, the exact product JV can be computed using the AD technique called the "forward mode"; given an arbitrary m-by-t_W matrix W, the product W^TJ can be computed by the AD technique using the "reverse mode" [19]. The computational cost of JV is $t_V \cdot \omega(F)$ while the cost of for W^TJ is $t_W \cdot \omega(F)$ where $\omega(F)$ is the cost to evaluate F. Compared with the finite difference technique, the AD technique can directly compute JV and W^TJ exactly without forming J explicitly. The cost of computing JV is equal to the finite difference technique.

When the Jacobian matrix J is sparse, the matrix J can be evaluated from the matrix-vector product, JV. First, the columns of J are partitioned into a set of groups G_C, where the number of groups in G_C is denoted by $|G_C|$. The columns in each group $G \in G_C$ are structurally orthogonal, which means $v_i \cdot w_i = 0,\ i = 1 : n$ for two n-vectors v and w. Each group $G \in G_C$ determines a column of V; $v_i = 1$ if and only if the column i is in group G_C; otherwise, $v_i = 0$. In particular, a coloring of the column-intersection graph of J yields a matrix V such that the nonzeros of J can be determined, directly, from the product JV.

Thus, the nonzeros of J can be immediately determined from the product JV. If we can partition the columns of J into a set of groups G_C as described above, with a small $t_V = |G_C|$, then this leads to a thin matrix $V \in \mathbb{R}^{n\times t_V}$, to obtain all the nonzeros of J. However, if there is a dense row in J, then $|G_C|$ cannot be small. Instead, we can choose the reverse mode to estimate the Jacobian matrix J. First, we partition the rows of J into groups G_R and determined all the nonzeros of J by W^TJ. However, if there is a dense column in J, then $|G_R|$ is not small, either. Obviously, there are some column or row preferable problems. Unfortunately, there are a lot of problems neither approach is preferable, for instance, two matrices in Example 1. Therefore, the two-sided coloring technique can be a good solution for these problems. In this paper, our task is to efficiently determine thin matrices V and W, so that the nonzeros of tril(J) can be readily extracted from the information of JV and W^TJ. The product W^TJ is computed by the reverse mode; the forward mode determines JV. As shown in [10], the determination of V and W is equivalent to construct p-color and q-color graphs. The number p represents the number of columns in V and q is the number of rows in W. Readers can refer to [18] for more details about the relationship between graph coloring and derivatives computation.

Definition 3.1 *A bipartition of a matrix J is a pair $(G_R,\ G_C)$ where G_R is a row partition of a subset of the rows of J and G_C is a column partition of a subset of the columns of J. Every nonzero of J is covered by at least one of G_R or G_C.*

The basic idea is to partition rows into G_R and columns into G_C, with $|G_C| + |G_R|$ as small as possible, so that the every nonzero element of $\mathrm{tril}(J)$ can be directly determined from either G_C or G_R. The following example shows the difference of full and partial bipartitions.

Example 2 *Examples for the full bipartition in [10] and the partial bipartition on the same matrix.*

$$J^{(1)} = \begin{bmatrix} \square_{11} & & & & \square_{15} & & & & \\ \square_{21} & & & & \square_{25} & & & & \\ \square_{31} & & & & \square_{35} & & & & \\ \square_{41} & & & & \square_{45} & \triangle_{46} & & & \\ & \triangle_{52} & \triangle_{53} & \triangle_{54} & \triangle_{55} & & \triangle_{57} & & \\ & & & \square_{64} & & & & & \\ & & & \square_{74} & & & \square_{77} & \triangle_{78} & \triangle_{79} \\ & & & & & & \square_{87} & & \\ & & & & & & \square_{97} & & \end{bmatrix}$$

$$J^{(2)} = \begin{bmatrix} \square_{11} & & & & X & & & & \\ \square_{21} & & & & X & & & & \\ \square_{31} & & & & X & & & & \\ \square_{41} & & & & X & X & & & \\ & \triangle_{52} & \triangle_{53} & \triangle_{54} & \triangle_{55} & & \triangle_{57} & & \\ & & & \square_{64} & & & & & \\ & & & \square_{74} & & & \square_{77} & X & X \\ & & & & & & \square_{87} & & \\ & & & & & & \square_{97} & & \end{bmatrix}$$

In Example 2, elements labelled $\square$ are computed from the column grouping, i.e., computed by JV; elements labelled $\triangle$ are computed from the row grouping, i.e., computed by $W^T J$; elements labelled X are ignored in the computation since we only need to determine the nonzeros elements in the partial bipartition. The matrix $J^{(1)}$ in Example 2 indicates that $|G_C| = 3$ and $|G_R| = 2$ by the full bicoloring estimation in [10] and we can determine all the elements directly with $V = (e_1 + e_4, e_5, e_7)$ and $W = (e_4 + e_7, e_5)$. The matrix $J^{(2)}$ indicates that $|G_C| = 2$ and $|G_R| = 1$ by our proposed partial bicoloring estimation; we can determine all the nonzero elements of the lower triangular part of $J^{(2)}$ directly with $V = (e_1 + e_4, e_7)$ and $W = (e_5)$. Our main goal is to obtain a bipartition (G_R, G_C) of J, from which we can determine its lower triangular part directly such that the total number of groups, $|G_R| + |G_C|$ is small.

There are two steps in our procedure. The first step is to partition the matrix J into two groups $\tilde{J} = P \cdot J \cdot Q = [J_C|J_R]$. This construction is crucial. The second step is to determine the matrix V and W. In this step, we define an approximation graph $\mathcal{G}_C^I$, $\mathcal{G}_R^I$ based on the partition $[J_C|J_R]$, then $\mathcal{G}_C^I$ yields a partition of a subset of G_C, which determines V. And matrix W is defined by the subset of G_R given the coloring of $\mathcal{G}_{\mathbb{R}}^I$. With matrices V and W, the lower triangular part of J can be determined by $(JV,\ W^TJ)$ directly. Thus, we first consider the problem of obtaining a useful partition $[J_C|J_R]$ and corresponding permutation matrices P and Q. Here, we adapt the minimum nonzero count ordering (MNCO) algorithm in [10] for the partition.

The MNCO algorithm [10] builds the partition J_C from the bottom up and the partition J_R from right to left. At the kth iteration either a new row is added to the partition J_C or a new column is added to J_R. This choice depends on a lower bound:

$$\rho(J_R^T) + \max(\rho(J_C), nnz(r)) < (\rho(J_C) + \max(\rho(J_R^T), nnz(c))),$$

where $\rho(A)$ is the maximum number of nonzeros in any row of matrix A, r is a row under consideration to be added to J_C and c is a column under consideration to be added to J_R. Therefore, the number of colors needed to color $\mathcal{G}_C^I$ is bounded below by $\rho(J_C)$; the number of colors needed to color $\mathcal{G}_R^I$ is bounded below by $\rho(J_R^T)$.

Upon completion of the MNCO algorithm, partitions J_C and J_R are defined. Then, we can define intersection graphs $\mathcal{G}_C^I$ and $\mathcal{G}_R^I$, based on the partition $[J_C|J_R]$. Following are the definitions of $\mathcal{G}_C^I$ and $\mathcal{G}_R^I$.

Definition 3.2 *The intersection partial graph $\mathcal{G}_C^I = (\mathcal{V}_C^I, \mathcal{E}_C^I)$ can be defined as follows.*

1. *vertex $j \in \mathcal{V}_C^I$ if $nnz(\text{column } j \cap J_C) \neq 0$;*
2. *$(r, s) \in \mathcal{E}_C^I$ if $r \in \mathcal{V}_C^I$, $s \in \mathcal{V}_C^I$, and there exists k ($k \geq r$ or $k \geq s$) such that $J_{kr} \neq 0$, $J_{ks} \neq 0$, and either $(k, r) \in J_C$ or $(k, s) \in J_C$.*

Definition 3.3 *The intersection partial graph $\mathcal{G}_R^I = (\mathcal{V}_R^I, \mathcal{E}_R^I)$ can be defined as follows.*

1. *vertex $j \in \mathcal{V}_R^I$ if $nnz(\text{row } j \cap J_R) \neq 0$;*
2. *$(r, s) \in \mathcal{E}_R^I$ if $r \in \mathcal{V}_R^I$, $s \in \mathcal{V}_R^I$, there exists k ($k \leq r$ or $k \leq s$) such that $J_{rk} \neq 0$, $J_{sk} \neq 0$ and either $(r, k) \in J_R$ or $(s, k) \in J_R$.*

The key point in the construction of partial graph $\mathcal{G}_C^I$ is that columns r and s are intersect if and only if their nonzero entries partially overlap in J_C, that is columns r and s intersect if $J_{kr} \cdot J_{ks} \neq 0$ and either $(k, r) \in J_C$ or $(k, s) \in J_C$ for some k larger than r or s. Similarly, for $\mathcal{G}_R^I$, if $J_{rk} \cdot J_{sk} \neq 0$ and either $(r, k) \in J_R$ or $(s, k) \in J_R$ for some k less than r or s, then rows r and s are intersect. To determine the lower triangular part of a matrix A, assume a nonzero entry $(i, j) \in J_C,\ i \geq j$, then column j assigned a color is in a group of G_C induced by coloring of $\mathcal{G}_C^I$. Thus, according to Definition 3.2, there is no other column in G_C with a nonzero in row i. It implies that entry (i, j) can be directly determined. For a nonzero entry $(r, s) \in J_R,\ r \geq s$, it can be directly determined according to Definition 3.3 as well. Therefore, the whole lower triangular part of A can be recovered. The theoretical proof of the equivalence between direct determination of the lower triangular part of A and a partial p-coloring graph problem can be found in Appendix.

4 Numerical Experiments

In this section, we present some numerical experiments on the partial bicoloring method for evaluating the lower triangular part of Jacobian matrices, and the performance of the Jacobian free Newton-Krylov subspace method for nonlinear equations based on our partial bicoloring method and automatic differentiation. All experiments are carried on a machine with Intel Dural Core 1.66 GHz, 1GB RAM, 60 GB hard driver running under Windows XP and Matlab 7.0. As we mentioned in previous section, the AD technique was employed to compute the matrix-vector products JV and $W^T J$. Here, we use the Matlab AD package ADMAT-2.0 [5] for the implementation of the forward mode and reverse mode AD.

First, we will compare the effectiveness of the partial bicoloring technique with the full bicoloring technique in [10] and the one-sided coloring technique in [15]. In contrast to [15], we do not set up a drop tolerance to remove nonzero elements in a row if they are less than a predetermined tolerance. In other words, we do not change the lower triangular sparsity of a Jacobian matrix in the whole computation procedure. As for the full bicoloring method in [10], we will use it to recover the whole sparse Jacobian matrix, not just the lower triangular part. Test matrices reside from the Tim Davis collection [14] and its subcollections. All selected matrices for testing are listed in Table 1. The first column of Table 1 is the names of selected matrices. The second column is the application background from which selected matrices were generated. The third and fourth columns are the matrix sizes and numbers of nonzeros of selected matrices, respectively. Then, we apply these three techniques, full bicoloring estimation, partial bicoloring estimation and partial one-sided coloring, to all these matrices. The number of matrix-vector products, MV is listed in Table 2.

From Table 2, it shows that the partial bicoloring technique requires much fewer matrix-vector products than the full bicoloring method in most testing cases. Compared with the one-sided partial coloring technique in [13], the bicoloring technique can reduce the number of matrix-vector multiplications significantly.

Next, we will test the Newton-Krylov method for solving some typical nonlinear equations [25] based on automatic differentiation. In the computational process, we first use ADMAT-2.0 [5] to evaluate an initial Jacobian matrix, J_0. Then, we get the sparsity of the Jacobian matrix and the incomplete LDU decomposition of J_0. Next, we solve the first Jacobian systems based on the incomplete LDU decomposition. After that iteration, we construct the preconditioner as in (2.4) and solve the corresponding Jacobian system by the Krylov

Matrix	Type	size	nonzeros
circuit_1	Circuit simulation problem	2624	35823
rajat01	Circuit simulation problem	6833	43250
rajat02	Circuit simulation problem	1879	12818
coupled	Circuit simulation problem	11341	97193
jan99jac020	Economic problem	6435	51480
orani678	Economic problem	2529	90158
Raefsky6	Structured problem	3402	130371
can256	Structured problem	256	2916
G22	Undirected random graph	2000	39980
nopoly	Undirected weighted graph	10774	70842
garon1	Computed fluid dynamic problem	3175	84723
graham1	Computed fluid dynamic problem	9035	335472
cavity16	Computed fluid dynamic problem	4562	137887
bp_600	Subsequent option problem	822	4172
C-20	Optimization problem	2921	20445
viscoplastic2	Material problem	32769	381326

Table 1: Testing matrices collection.

Matrix	Full bicoloring	Partial bicoloring	Partial one-sided coloring
circuit_1	139	125	2570
rajat01	32	22	26
rajat12	39	27	1195
coupled	86	73	2152
jan99jac020	68	44	60
orani678	245	200	236
Raefsky6	125	111	126
can256	30	20	59
G22	82	60	62
nopoly	11	11	12
garon1	51	45	47
graham1	147	145	290
cavity16	69	48	48
bp_600	27	18	25
C-20	72	55	158
viscoplastic2	97	23	24

Table 2: The number of matrix-vector products for three estimations, full bicoloring estimation, partial bicoloring estimation and partial one-side coloring estimation.

method for each iteration until we get the solution for the nonlinear equations. For the comparison, we choose the standard Newton trust region method [26, 27]. The only difference between these two methods is the Newton trust region method forms the Jacobian matrix explicitly via AD and solves (1.2) by '\' in Matlab while the Newton-Krylov method forms the preconditioner through the partial bicoloring Jacobian estimation and solves (1.2) by the GMRES method. In other words, the dogleg and trust region ideas are also implemented in the Newton-Krylov method. In this experiment, we choose a few classic nonlinear equations in Moré, Garbow and Hillstorm's collection [25].

1. Extended Rosenbrock function.
$$f_{2l-1}(x) = 10(x_{2l} - x_{2l-1}^2),\ f_{2l}(x) = 1 - x_{2l-1},\ \text{where the problem size } n = 2l.$$
2. Extended Powell singular function.
$$\begin{array}{ll} f_{4l-3}(x) = x_{4l-3} + 10x_{4l-2}, & f_{4l-2}(x) = 5^{1/2}(x_{4l-1} - x_{4l}) \\ f_{4l-1}(x) = (x_{4l-2} - 2x_{4l-1})^2, & f_{4l}(x) = 10^{1/2}(x_{4l-3} - x_{4l})^2, \end{array}$$
where the problem size n is a multiple of 4.
3. Discrete boundary value function.
$$f_i(x) = 2x_i - x_{i-1} - x_{i+1} + h^2(x_i + t_i + 1)^3/2,$$
where the problem size is n, $h = 1/(n+1)$, $t_i = ih$ and $x_0 = x_{n+1} = 0$.
4. Broyden tridiagonal function.
$$f_i(x) = (3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1.$$

Table 3 lists the comparison results of the Newton-Krylov method with the standard Newton method. Due to the sparsity of Jacobian matrices, only a few matrix-vector products are required to estimate the lower triangular part for the preconditioner construction (2.4) at each Newton iteration. The results indicate that the preconditioner leads to a fast convergence of Krylov subspace method for solving (1.2). Thus, the Newton-Krylov method is superior to the standard Newton method when the problem size is large.

Finally, we will analyze a flow problem of a fluid during injection into a long vertical channel [2]. Assume that the flow is modelled by the boundary value problem
$$\begin{array}{lll} u^{(4)} = R(u'u'' - uu^{(3)}), & & 0 \le t \le 1, \\ u(0) = 0, & u(1) = 1, & u'(0) = u'(1) = 0, \end{array} \tag{4.5}$$
where u is the potential function, u' is the tangential velocity of the fluid, and R is the Reynolds number. We solve this problem by the k-stage collection method. First, partition the interval [0, 1] into n subintervals uniformly with the length h, that is
$$0 = t_1 < t_2 < \cdots < t_n < t_{n+1} = 1.$$

$n = 100$						
	Newton-Krylov				Newton	
Problem	time	it	MVs	Krylov	time	it
Powell	1.14	9	3	6	0.06	9
Rosen	0.16	4	2	2	0.02	4
DBV	0.56	5	3	3	0.10	5
Broyden	0.40	5	3	3	0.05	5
$n = 500$						
	Newton-Krylov				Newton	
Problem	time	it	MVs	Krylov	time	it
Powell	1.26	9	3	6	1.06	9
Rosen	0.28	4	2	2	0.26	4
DBV	0.63	4	3	2	0.97	4
Boyden	0.50	5	3	3	0.52	5
$n = 1000$						
	Newton-Krylov				Newton	
Problem	time	it	MVs	Krylov	time	it
Powell	1.89	9	3	6	6.75	9
Rosen	0.50	4	2	2	0.94	4
DBV	1.24	4	3	2	3.85	4
Boyden	1.03	5	3	3	2.06	5
$n = 2000$						
	Newton-Krylov				Newton	
Problem	time	it	MVs	Krylov	time	it
Powell	3.26	9	3	6	44.03	9
Rosen	1.25	4	2	2	3.88	4
DBV	3.46	4	3	2	18.87	4
Boyden	3.30	5	3	3	7.99	5

Table 3: The performance comparison of the Newton-Krylov method and standard Powell's dogleg method for four typical nonlinear equations. (Column 'it' for the number of iterations; column 'MVs' for the number of matrix-vector products; column 'Krylov' for the average number of Krylov iterations at each Newton iteration).

Then, the k-stage collection method is defined in terms of k points,

$$0 < \rho_1 < \rho_2 < \cdots < \rho_k < 1,$$

where $k \geq m$ ($m = 4$ in this case) and ρ_i are the roots of the Legendre polynomial of order $2k$. This choice guarantees superconvergence at the mesh points t_i. The k-stage method approximates the solution of the boundary value problem by a piecewise polynomial p_i, where p_i is a polynomial of order $k+4$ in each subinterval $[t_i,\ t_{i+1}]$. In our experiment, we let k equal to 4. Thus, in the interval $[0,\ 1]$, p_i is defined in terms of $8n$ parameters. We have to specify these $8n$ parameters to satisfy $p_i \in C^3[0,\ 1]$, four boundary conditions in (4.5), and that the differential equation (4.5) holds on p_i at the collection points,

$$\xi_{ij} = t_i + hp_j, \qquad \text{for} \quad 1 \leq i \leq n, \quad 1 \leq j \leq 4.$$

The polynomial of order 8 on each subinterval $[t_i,\ t_{i+1}]$ is of the form [1]

$$p_i = \sum_{i=1}^{4} \frac{(t-t_i)^{j-1}}{(j-1)!} v_{ij} + \sum_{j=1}^{4} \frac{(t-t_i)^{j+3}}{(j+3)!h^{j-1}} w_{ij}, \qquad t \in [t_i,\ t_{i+1}], \quad \text{for} \quad 1 \leq i \leq n. \tag{4.6}$$

Then, based on the continuity condition $p \in C^3[0,\ 1]$ at the interior grid points, we have

$$p_i^{l-1}(t_i^+) = p_{i-1}^{(l-1)}(t_i^-), \qquad 1 \leq l \leq 4, \quad 1 \leq i \leq n.$$

Furthermore, at the collection points, the differential equation (4.5) holds, that is

$$p_i^{(4)}(\xi_{ij}) = R[p_i'(\xi_{ij})p_i''(\xi_{ij}) - p_i(\xi_{ij})p_i^{(3)}(\xi_{ij})], \qquad 1 \leq j \leq 4, \quad 1 \leq i \leq n.$$

Thus, we have $8n$ equations in total to determine $8n$ parameters v_{ij} and w_{ij} in (4.6). In our experiment, we test four case with $n = 100$, $n = 200$, $n = 300$ and $n = 400$ with $R = 0$, $R = 1$, and $R = 10$, respectively. Table 4 shows the performance results of the Newton-Krylov method and standard Newton method on solving the flow problem. As we know, when $R = 0$ the flow problem (4.5) is reduced to a linear problem by the k-stage collection method. As R goes up, the problem becomes harder to solve. From Table 4, we can find out that the speedup of the Newton-Krylov method increases significantly as the problem size and R increase. Also, the preconditioner is very effective to accelerate the convergence of the Krylov subspace method at each iteration. Therefore, from the above five nonlinear problems, it turns out that the Newton-Krylov method is much better than the standard Newton method, especially when the Jacobian matrix is sparse and the problem size is large.

5 Conclusions

The Jacobian-free Newton-Krylov method is widely used in solving nonlinear equations arising in many applications. However, constructing an economic and effective preconditioner is

$R = 0$					
	Newton-Krylov			Newton	
size(n)	time	it	Krylov	time	it
100	6.53	5	3	14.84	5
200	12.67	5	3	46.18	5
300	20.92	5	3	100.05	5
400	56.60	5	3	183.62	5
$R = 1$					
	Newton-Krylov			Newton	
size(n)	time	it	Krylov	time	it
100	6.56	4	4	11.87	4
200	15.65	4	4	36.67	4
300	30.84	4	4	78.36	4
400	47.08	4	4	142.02	4
$R = 10$					
	Newton-Krylov			Newton	
size(n)	time	it	Krylov	time	it
100	14.33	8	5	23.64	8
200	26.20	8	8	73.05	8
300	47.45	8	7	157.08	8
400	70.09	8	5	285.96	8

Table 4: The performance comparison of the Newton-Krylov method and standard Powell's dogleg method for a flow channel problem. (Column 'it' for the number of iterations; column 'Krylov' for the average number of Krylov iterations at each Newton iteration).

a challenge. In 2006, Tůma *et al.* proposed a method to construct the preconditioner based on the lower triangular part of the Jacobian matrix. The heart of this method is a one-sided coloring procedure to estimate the lower triangular half of the sparse Jacobian matrix via finite differences. However, when there are dense rows in the lower triangular part, this one-sided coloring procedure can be unacceptably expensive since it can then require many function evaluations (or the introduction of additional tolerances to force 'artificial' sparsity).

We have proposed a related preconditioning idea that remains effective when there are dense rows, based on automatic differentiation technology developed in the context of sparse Jacobians [eg. [7]]. In particular, we have proposed to adapt the two-sided bicoloring method developed in [10] to the case of determining the lower triangular half of the sparse Jacobian matrix. This new procedure not only handles both dense rows and dense columns, but also provides a more stable and accurate result than the previous finite differences approach. The numerical experiments in Section 4 illustrate the effectiveness of our new method and we conclude that this can be an effective approach.

Future directions of research and possible improvements, in this vein, include introducing an initial permutation matrix to appropriately 'flavor' the lower triangular half of the Jacobian (that defines the preconditioner), adapting the coloring procedure to account for both constant (non-zero) values and Jacobian entries that are 'copies', and taking advantage of other structural properties.

6 Appendix: Bipartite Graph

We have described the partial bicoloring idea in terms of intersection graphs. There is also an interesting bipartite graph interpolation. Based on the bipartition graph, we can construct the equivalence between the direct determination of lower triangular part of A and a p-coloring bipartite graph problem.

A p-coloring graph is defined as $\mathcal{G} = (V, E)$, where V is the set of vertices and E is the set of edges. There exists a function ϕ defined as

$$\phi : V \to \{1, 2, \cdots, p\},$$

such that $\phi(u) \neq \phi(v)$ if $(u, v) \in E$. Thus, the smallest p, for which $\mathcal{G}$ is p-coloring, defined as the chromatic number $\chi(\mathcal{G})$. A p-coloring ϕ of $\mathcal{G}$ partitions vertices $\mathcal{V}$ into p groups G_k, $k = 1, 2, \cdots, p$ such that

$$G_k = \{u \in V : \ \phi(u) = k\}.$$

Given a matrix $A \in \mathbb{R}^{n\times n}$, we can construct a bipartite graph $\mathcal{G}_b(A) = \{[V_1, V_2], \ E\}$, where $V_1 = \{r_1, r_2, \cdots, r_n\}$, $V_2 = \{c_1, c_2, \cdots, c_n\}$. c_j corresponds to jth column of A, and r_i corresponds to the ith row of A. If a_{ij} is nonzeros, then there is an edge between r_i and c_j, i.e., $(r_i, c_j) \in E$.

Definition 6.1 *A partial bipartition $(G_R,\ G_C)$ of J is consistent with direct determination of the lower triangular portion if for every nonzeros entry a_{ij} of J, either column j is in group of G_C which satisfies one of following conditions,*

1. *If $i \geq j$, there is no other columns having a nonzero in row i of J.*
2. *If $i < j$, there is no other column k $(k < i)$ having a nonzero in row i of J.*

or, row i is in a group of G_R, which satisfies one of following conditions,

1. *If $i \geq j$, there is no other rows having a nonzero in column j of J.*
2. *If $i < j$, there is no other row k $(k > j)$ having a nonzero in column j of J.*

Note that Definition 6.1 here is different from the one in [10], which defines a bipartition $(G_R,\ G_C)$ for all nonzeros of J, instead of the lower triangular portion, without the restriction on k.

A path p-coloring of a graph can be defined to be a vertex coloring using p colors with the property that every path with at least three edges uses at least three colors [7, 8]. Here, we give a new definition of a partial bipartite path p-coloring of a graph with the modification in [10], that is we assign a color "0" to some vertices, i.e., $\phi(x) = 0$, which means that vertex i is not assigned a color.

Definition 6.2 *Let $\mathcal{G}_b = ([V_1, V_2],\ E)$ be a bipartite graph. A mapping $\phi : [V_1, V_2] \rightarrow \{0, 1, 2, \cdots, p\}$ is a partial bipartite path p-coloring of $\mathcal{G}_b$ if the following conditions hold.*

1. *Adjacent vertices have different assignments, that is for $i \in V_1,\ j \in V_2,\ (i, j) \in E$, if $i \geq j$ then $\phi(i) \neq \phi(j)$, or if $j > i$ and there exists $k \in V_2,\ k \leq i$ and $(i, k) \in E$ then $\phi(i) \neq \phi(j)$, or if $j > i$ and there exists $k \in V_1,\ k \geq j$ and $(k, j) \in E$ then $\phi(i) \neq \phi(j)$.*
2. *The set of positive colors used by vertices in V_1 is disjoint from the set of positive colors used by vertices in V_2, i.e., $i \in V_1,\ j \in V_2\ \Rightarrow\ \{\phi(i) \neq \phi(j)$ or $\phi(i) = \phi(j) = 0\}$.*
3. *For vertices i and j are adjacent to vertex k with $\phi(k) = 0$, if $i, j \in V_2$ and $k \geq j$ or $k \geq i$ then $\phi(i) \neq \phi(j)$; if $i, j \in V_1$ and $k \leq j$ or $k \leq i$, then $\phi(i) \neq \phi(j)$.*
4. *A path of three edges, say $r_i - c_j - r_k - c_l$, if $k \geq j$, then it uses at least three colors.*

We denote $\chi_p(\mathcal{G}_b)$ as the smallest number for which $\mathcal{G}_b$ is partial bipartite. Following theorem builds up the connection of a bipartition consistent with direct determination with the lower triangular part of matrix A.

Theorem 6.1 *Let A be an n-by-n matrix with corresponding partial bipartite graph $\mathcal{G}_b(A) = ([V_1,\ V_2],\ E)$. The mapping $\phi : [V_1, V_2] \rightarrow \{0, 1, \cdots, p\}$ bipartition $[G_R, G_C]$ with $|G_R| + |G_C| = p$, consistent with direct determination of the lower triangular portion of A if and only if ϕ is a partial bipartite path p-coloring of $\mathcal{G}_b(A)$.*

Proof. The proof is similar to the one for Theorem 3.1 in [10].

($\Leftarrow$) Assume that ϕ is a partial bipartite path p-coloring of $\mathcal{G}_b(A)$, including a partial bipartite $(G_R,\ G_C)$ of rows and columns of A. If this partial bipartition is not consistent with direct determination, there exists a nonzero entry $a_{i,j}$ of A does not satisfy with conditions in Definition 6.1. This can happen only if following one of situations holds.

1. $\phi(i) = 0$, $\phi(j) \neq 0$, $i \geq j$ and there exists a column q with $a_{iq} \neq 0$, such that $\phi(j) = \phi(q)$, which leads to a contradiction of condition 3 of Definition 6.2.

2. $\phi(i) = 0$, $\phi(j) \neq 0$, $i < j$ and there exists a column q such that $i \geq q$ with $a_{iq} \neq 0$ such that $\phi(j) = \phi(q)$. It contradicts with the condition 3 of Definition 6.2.

3. $\phi(i) \neq 0$, $\phi(j) = 0$, $i \geq j$ and there exists a row q with $a_{qj} \neq 0$ such that $\phi(i) = \phi(q)$, which contradicts to condition 3.

4. $\phi(i) \neq 0$, $\phi(j) = 0$, $i < j$ and there exits a row q such that $q \geq j$ with $a_{qj} \neq 0$ such that $\phi(i) = \phi(q)$ which contradicts the condition 3.

5. $\phi(i) \neq 0$, $\phi(j) \neq 0$, $i \geq j$ and there exists a column q and a row p such that columns j and q are in the same group with $a_{iq} \neq 0$ and rows i and p are in the same group with $a_{pj} \neq 0$. This implies that ϕ is a 2-coloring of the path $(r_p - c_j - r_i - c_q)$, which contradicts of condition 4.

6. $\phi(i) \neq 0$, $\phi(j) \neq 0$, $i < j$ and there exists a column p such that and a row q such that rows i and p are in the same group with $a_{pj} \neq 0$ and $p \geq j$, and columns j and q are in the same group with $a_{pq} \neq 0$. Then, it implies that a 2-coloring path $(r_i - c_j - r_p - c_q)$, which contradicts condition 4.

($\Rightarrow$) Assume that ϕ induces a partial bipartite consistent with direct determination of A. It is obvious that condition 1-3 satisfied. We remain establish the condition 4. Assume there is a bicolored path $r_i - c_j - r_k - c_l$, where $k \geq j$, $\phi(i) = \phi(k)$ and $\phi(j) = \phi(l)$. It is clear from condition 3, that there are two colors are positive. If $k \geq j$ then the element a_{kj} cannot be determined directly since $\phi(i) = \phi(k)$ and $\phi(j) = \phi(l)$.

$\square$

References

[1] U.M. Ascher, R.M.M, Mattheij and R.D. Russell, *Numerical solution of boundary value problems for ordinary differential equations*, SIAM, 1995.

[2] B.M. Averick, R.G. Carter and J.J. Moré, *The MINIPACK-2 test problem collection*, Working paper, Technical Memorandum No. 150, Argonne National Laboragory, 1991.

[3] C.G. Broyden, *The convergence of an algorithm for solving sparse nonlinear systems*, Math. Comput., vol. 25, 1971, 285-294.

[4] P.Børstad, *Fast numerical solution of the biharmonic Dirichlet problem on rectangles*, SIAM J. Numer. Anal, Vol.20, 1983, 59-71.

[5] Cayuga Research Associates, LLC, *ADMAT-2.0 User's Guide*, 2009.

[6] Y.Chen and C. Shen, *A Jacobian-free Newton-GMRES(m) with adaptive preconditioners and its application for power flow calculations*, IEEE Transactions on Power Systems, Vol.21, 2006, 1096-1103.

[7] T.F. Coleman and J.Y. Cai, *The cyclic coloring problem and estimation of sparse Hessian matrices*, SIAM J. Alg. Discrete Methods, Vol. 7, 1986, 221-235.

[8] T.F. Coleman and J.J. Moré, *Estimation of sparse Hessian matrices and graph coloring problems*, Math. Programming, Vol. 28, 1984, 243-270.

[9] T.F. Coleman and G.F. Jonsson, *The efficient computation of structured gradients using automatic differentiation*, SIAM J. Sci. Comput. Vol. 20, 1999, 1430–1437.

[10] T.F.Coleman and A.Verma,*The efficient computation of sparse Jacobian matrices using automatic differentiation*, SIAM J. Sci. Comput. Vol. 19, 1998, 1210–1233.

[11] T.F. Coleman and A. Verma, *ADMIT-1: Automatic differentiation and MATLAB interface toolbox*, ACM Transaction on Mathematics Software, Vol.26, 2000, 150-175.

[12] T.F. Coleman and W. Xu, *ADMAT-2.0*, Available at http://www.math.uwaterloo.ca/CandO_Dept/securedDownloadsWhitelist/index.shtml, 2009.

[13] J. Cullum and M. Tůma. *Matrix-Free Preconditioning Using Partial Matrix Estimation*, BIT, Vol.46, 2006, 711-729.

[14] T. Davis and Y. Hu, *The university of Florida sparse matrix collection*, to appear on ACM Transaction on Mathematical Software, 2010.

[15] J. Duintjer Tebbens and M. Tůma, *Preconditioner updates for solving sequences of large and sparse nonsymmetric linear systems* , SIAM Journal on Scientific Computing, Vol.29, 2007, 1918-1941.

[16] J. Duintjer Tebbens and M. Tůma. Preconditioner updates for solving sequences of linear systems in matrix-free environment , Numerical Linear Algebra and Applications, electronically published 2010: 10.1002/nla.695

[17] G.H.G.R.W Freund and N.M. Nachtigal, *Iterative solution of linear systems*, Acta Numerica 1992, Cambridge University Press, 1992, 57-100.

[18] A.H. Grebremedhin, F. Manne and A. Pothen, *What Color is Your Jacobian? Graph Coloring for Computing Derivatives*, SIAM Review Vol.47, 2005, 629-705.

[19] A. Griewank and A. Walther, *Evaluating Derivatives : Principles, and Techniques of Algorithmic Differentiation* 2nd Ed., SIAM, Philadelphia, PA, 2005.

[20] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd Ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[21] M.H. Gutknecht, *Lanczos-type solvers for nonsymmetric linear systems of equations*, Acta Numerica 1997, Cambridge University Press, Cambridge, 1997, 271-398.

[22] M.R. Hestenes and E.L. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bureau of Standards, Section B 49, 1952, 409-436.

[23] D.A. Knoll and D.E. Keyes, *Jacobian-free Newton-Krylov methods: a survey of approaches and applications*, J. Computational Physics, Vol.193, 2004, 357-397.

[24] X.Li and F. Primeau, *A fast Newton-Krylov solver for seasonally varying global ocean biogeochemistry models*, Ocean Modell, Vol.23, 2008, 13-20.

[25] J.J. Moré, B.S. Grabow and K.E. Hillstrom, *Test unconstrained optimization software*, ACM Tran. Math. Soft. vol.7, 17-24, 1981.

[26] H.B. Nielsen, *immoptibox—A MATLAB TOOLBOX FOR OPTIMIZATION*, IMM, Technical University of Denmark, 2006. Available at http://www2.imm.dtu.dk/ hbn/immoptibox/

[27] M.J.D. Powell, *A hybrid method for nonlinear equations*, in Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon and Breach, London, 87-114, 1970.

[28] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, 1996.

[29] Y. Saad and H.A. van der Vorst, *Iterative solution of linear systems in the 20th century*, J. Comp. App. Math, Vol. 123, 2000, 1-33.

[30] W. Xu, T. F. Coleman and G. Liu,*A Secant Method for Nonlinear Least-Squares Minimization*, J. Computational Optimization and Applications, DOI: 10.1007/s10589-010-9336-4, 2010.