Optimization Methods and Software
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/goms20

Solving nonlinear equations with the Newton-Krylov method based on automatic differentiation

Wei Xu [a] & Thomas F. Coleman [b]

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

[b] Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada, N2L 3G1

Accepted author version posted online: 24 Sep 2012.Version of record first published: 05 Nov 2012.

To cite this article: Wei Xu & Thomas F. Coleman (2012): Solving nonlinear equations with the Newton-Krylov method based on automatic differentiation, Optimization Methods and Software, DOI:10.1080/10556788.2012.733004

To link to this article: http://dx.doi.org/10.1080/10556788.2012.733004

Optimization Methods & Software
iFirst, 2012, 1–14

Taylor & Francis
Taylor & Francis Group

Solving nonlinear equations with the Newton–Krylov method based on automatic differentiation

Wei Xu[a]* and Thomas F. Coleman[b]

[a]*Department of Mathematics, Tongji University, Shanghai 200092, China;* [b]*Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada N2L 3G1*

(*Received 16 January 2012; final version received 19 September 2012*)

The Jacobian-free Newton–Krylov method is widely used in solving nonlinear equations arising in many applications. However, an effective preconditioner is required for each iteration and determining such may be hard or expensive. In this article, we propose an efficient two-sided bicolouring 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. The numerical experiments illustrate that the proposed bicolouring approach can be significantly more effective than the one-sided colouring method proposed in Cullum and Tůma [*Matrix-free preconditioning using partial matrix estimation*, BIT 46 (2006), pp. 711–729] and yields an effective preconditioning strategy.

Keywords: automatic differentiation; preconditioner iterative methods; nonlinear equation solvers; Newton method; intersection graphs; graph colouring

1. Introduction

The multiple dimensional zero-finding problem is

$$\text{Solve } F(x) = 0, \tag{1}$$

where $F(x) = (f_1(x),\ f_2(x), \ldots, f_n(x))^{\mathrm{T}}$ 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

$$\text{Find } d_x \in \mathbb{R}^n \quad \text{such that } J(x)d_x = -F(x), \tag{2}$$

where $J(x)$ is an n-by-n Jacobian matrix of $F(x)$, that is, $J(x) = (\partial f/\partial x_i)_{n\times n}$. 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 Newton iteration [3,28]. Although the computational cost of each

*Corresponding author. Email: wdxu@tongji.edu.cn

ISSN 1055-6788 print/ISSN 1029-4937 online

http://dx.doi.org/10.1080/10556788.2012.733004
http://www.tandfonline.com

iteration is thereby reduced, more iterations are usually required to converge to the root of (1). In fact, 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 [19] reviewed the Jacobian-free Newton–Krylov (JFNK) method for solving the nonlinear equations (1); this approach combines a matrix-free approach with nonsymmetric Krylov subspace methods. The Krylov subspace method was first proposed in 1950 [18]. This method is a projection, or generalized projection, method for solving a linear system $Ax = b$ using the Krylov subspace K_j,

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

where $r_0 = b - Ax_0$. There are a variety methods falling into the Krylov taxonomy [2,13,17,25, 26]. 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, which only requires the matrix–vector product, that is, Ax.

In order to construct a matrix–vector product $J(x) \cdot v$ for (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

$$J(x) \cdot v = \frac{F(x + \epsilon v) - F(x)}{\epsilon}, \tag{3}$$

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 [5,20,21].

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 iterative 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 [19], 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 (3) introduces some error into the Newton system (1). It is known that (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 and co-workers [8,11,12] proposed a matrix-free environment to construct updated preconditioners for a sequence of matrices based on partial matrix estimation. In those papers, only the triangular parts of the sequence of matrices are used to update the preconditioners. For solving the nonlinear equations (1), the Jacobian-vector product approximation (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 [1], Bellavia *et al.* proposed an another preconditioner construction technique for the Newton–Krylov method. Similar to [8], some part of the difference between the current Jacobian matrix and the initial matrix is also required for the preconditioner construction at each Newton iteration. However, they did not propose an efficient way to recover these nonzero entries.

In this article, we will not only answer the second question, evaluating the Jacobian-vector product $J(x)v$ exactly with less computational cost than the finite difference method, but also will evaluate the exact product $w^{\mathrm{T}}J(x)$ based on the automatic differentiation (AD) [16]. We then generalize Tůma's preconditioner construction method from one-sided colour updating to two-sided colour updating. It will turn out that two-sided colouring updating will be much more efficient than the one-sided colouring updating, especially when there are some dense rows in the lower triangular part of the Jacobian matrix. Moreover, the bicolouring technique can be generalized to recover any restricted partial Jacobian matrix in [15].

The rest of the article 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 propose our method to update the lower triangular part of the Jacobian matrices by the bicolouring graph technique with the AD. Then, some numerical results will show the efficiency of the bicolouring updating technique and the performance of solving nonlinear equations in Section 4. Finally, we present concluding remarks in Section 5.

2. Preconditioner construction

In [11,12], 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 $J = LDU$ is the incomplete triangular decomposition [14] 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

$$P \equiv (LD - \mathrm{tril}(B))U \quad \text{or} \quad P \equiv L(DU - \mathrm{triu}(B)), \tag{4}$$

where $\mathrm{tril}(\cdot)$ and $\mathrm{triu}(\cdot)$ are the lower triangular and upper triangular part of a matrix, respectively. Thus, we solve a preconditioned linear system $P^{-1}\bar{J}(x)d_x = -P^{-1}F(x)$, instead of (2). If we want to construct the preconditioner as in (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 article. The following theorem shows the quality of the preconditioner if $(\overline{LD - B})$, the approximation to $(LD - B)$, is favourably chosen.

THEOREM 2.1 *Assume that $LDU + E = J$ for some error matrix E and let $\|\overline{BD^{-1}L^{-1}}\|_2 \leq 1/c < 1$, where $\|\cdot\|_2$ denotes the Euclidean norm and $\overline{BD^{-1}L^{-1}} = I - (\overline{LD - B})D^{-1}L^{-1}$. Further assume that the singular values σ_i of*

$$B(I - U) + (\overline{LD - B} - (LD + EU^{-1} - B))U$$

satisfy

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

for some integer t, $t \ll n$ and some small $\delta > 0$, where $(\overline{LD - B})$ is an easily invertible approximation matrix of $(LD - B)$. Let $(\overline{LD - B})$ have nonzeros main diagonal, and $D = \mathrm{diag}(d_1, \ldots, d_n)$.

Then, there exist matrices F and Δ such that

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

with $rank(\Delta) < t$ *and*

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

The proof of Theorem 2.1 is similar to that of Theorem 2.2 in [8]. In that paper, a one-sided colouring method, that is, a solver for the diagonal partial colouring 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 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 preconditioner updates for the Newton system can be written as follows.

ALGORITHM 1 *Consider problem* (2), *the preconditioner update for the Newton system at each iteration can be constructed in 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) *Colouring. Get a colouring pattern of* $S(J(x_0))$ *via solving a partial colouring problem based on one-sided colouring.*
(5) *For* $i = 0 : n$

Estimate the lower triangular part of $J(x_i)$ *via the colouring pattern with the*

matrix–vector product approximation.

Compute the difference matrix tril(B) *and construct the preconditioner from* (4).

end

The key component of Algorithm 1 is the one-sided colouring pattern of $S(J(x_0))$, that is, recovering $J(x)$ through the matrix–vector multiplication $J \cdot v$, and estimation of the lower triangular part of $J(x_i)$ via the colouring pattern with the matrix–vector product approximation. When the matrix $J(x_i)$ has dense columns in its lower triangular part, the one-sided colouring 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 $\text{tril}(J^{(2)})$, $V^{(2)}$ has five columns, while $J^{(1)}$ only requires two columns in $V^{(1)}$. Thus, we have to find a way to handle the dense row in $J^{(2)}$. In [8], 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 [8]. In the next section, we propose a

partial bicolouring graph method with the matrix–vector products Jv and $w^{\mathrm{T}}J$ so as this technique can be effective when there are both dense columns and rows in the tril(J).

Example 1 Examples for matrix estimation through matrix–vector products.

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

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

3. Partial bicolouring matrix update

The bicolouring technique was proposed by Coleman and Verma in [7] to efficiently determine the Jacobian matrix J using AD techniques. In this article, we generalize their bicolouring 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 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 $W^{\mathrm{T}}J$ can be computed by the AD technique using the 'reverse mode' [16]. The computational cost of JV is $t_V \cdot \omega(F)$, while the cost for $W^{\mathrm{T}}J$ 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 $W^{\mathrm{T}}J$ 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 if $v_i \cdot w_i = 0,\ i = 1 : n$ 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 colouring 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 determine all nonzeros of J by $W^{\mathrm{T}}J$. 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 colouring technique can be a good solution for these neither approach preferable problems. In this article, 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 $W^{\mathrm{T}}J$. The product $W^{\mathrm{T}}J$ is computed by the reverse mode; the forward mode determines JV. As shown in [7], the determination of V and W is equivalent to construct p-colour and q-colour graphs. The number p represents the number of columns in V and q is the number of columns in W. Readers can refer to [15] for more details about the relationship between graph colouring and derivatives computation. Usually, there are two approaches for the bicolouring method for full or partial matrix determination, direct and substitution [7,15,27]. However, this article is focused on applying the bicolouring approach to accelerate the Newton–Krylov method, not partial matrix determination methods. Thus, we just adopt the direct approach in [27], which is also same as the technique for the partial Jacobian determination in [15]. Other partial matrix determination technique in [15,27] can replace the direct approach here for further acceleration.

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 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 [7] 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, that is, computed by JV; elements labelled $\triangle$ are computed from the row grouping, that is, computed by $W^{\mathrm{T}}J$; elements labelled X are ignored in the computation since we only need to determine the nonzero elements in the partial bipartition. The matrix $J^{(1)}$ in Example 2 indicates that $|G_C| = 3$ and $|G_R| = 2$ by the full bicolouring estimation in [7] and we can determine all the elements directly with $V = (e_1 + e_4, e_5, e_7)$ and $W = (e_4 + e_7, e_5)$, where e_i is the ith column of the identity matrix. The matrix $J^{(2)}$ indicates that $|G_C| = 2$ and $|G_R| = 1$ by our proposed partial bicolouring estimation; we can determine all 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 matrices 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 colouring of $\mathcal{G}_R^I$. With matrices V and W, the lower triangular part of J can be determined by $(JV,\ W^{\mathrm{T}}J)$ 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 [7] for the partition.

The MNCO algorithm [7] 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:

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

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 colours needed to colour $\mathcal{G}_C^I$ is bounded below by $\rho(J_C)$; the number of colours needed to colour $\mathcal{G}_R^I$ is bounded below by $\rho(J_R^{\mathrm{T}})$.

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 $(k \geq r \text{ or } k \geq s)$ 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 $(k \leq r \text{ or } k \leq s)$ 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 colour is in a group of G_C induced by colouring 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. In [15], the direct determination of a subset S of A through a partial star bicolouring technique was proved. As a specific case for subset S, the direct determination of the lower triangular part of A and a partial p-colouring graph problem are equivalent.

4. Numerical experiments

In this section, we present some numerical experiments on the partial bicolouring method for evaluating the lower triangular part of Jacobian matrices, and the performance of the JFNK subspace method for nonlinear equations based on our partial bicolouring method and AD. All experiments are carried on a machine with Intel Core i5-2500 3.30 GHz, 12 GB RAM, 1 TB hard driver running under Windows 7 Professional and Matlab R2009a. As we mentioned in previous section, the AD technique was employed to compute the matrix–vector products JV and $W^{\mathrm{T}}J$. Here, we use the Matlab AD package ADMAT-2.0 [4] for the implementation of the forward mode and reverse mode AD. The stop criterion for the Newton method and Krylov subspace method is 10^{-6}.

Table 1. Testing matrices collection.

Matrix	Type	Size	Nonzeros
circuit_1	Circuit simulation problem	2624	35,823
rajat01	Circuit simulation problem	6833	43,250
rajat12	Circuit simulation problem	1879	12,818
coupled	Circuit simulation problem	11,341	97,193
jan99jac020	Economic problem	6435	51,480
orani678	Economic problem	2529	90,158
Raefsky6	Structured problem	3402	130,371
can256	Structured problem	256	2916
G22	Undirected random graph	2000	39,980
nopoly	Undirected weighted graph	10,774	70,842
garon1	Computed fluid dynamic problem	3175	84,723
graham1	Computed fluid dynamic problem	9035	335,472
cavity16	Computed fluid dynamic problem	4562	137,887
bp_600	Subsequent option problem	822	4172
C-20	Optimization problem	2921	20,445
viscoplastic2	Material problem	32,769	381,326

Table 2. The number of matrix–vector products and running times in seconds for three estimations, full bicolouring estimation, partial bicolouring estimation and partial one-side colouring estimation to determine the seed matrices V and W.

	Full bicolouring		Partial bicolouring		Partial one-sided bicolouring	
Matrix	MVs	Time(s)	MVs	Time(s)	MVs	Time(s)
circuit_1	139	7.39	125	7.29	2570	123.49
rajat01	32	3.33	22	3.05	26	3.04
rajat12	39	0.56	27	0.48	1195	31.32
coupled	86	9.84	73	8.71	2152	303.32
jan99jac020	68	4.86	44	4.41	60	4.28
orani678	245	9.34	200	7.31	236	8.88
Raefsky6	125	5.71	111	4.77	126	5.01
can256	30	0.33	20	0.21	59	0.57
G22	82	3.27	60	3.11	62	2.21
nopoly	11	3.29	11	4.11	12	3.44
garon1	51	2.38	45	3.77	47	3.07
graham1	147	6.13	145	8.11	290	28.63
cavity16	69	2.07	48	1.97	48	1.88
bp_600	27	0.05	18	0.11	25	0.13
C-20	72	0.4	55	0.9	158	1.24
viscoplastic2	97	170.00	23	56.11	24	43.32

First, we compare the effectiveness of the partial bicolouring technique with the full bicolouring technique in [7] and the one-sided colouring technique in [11]. In contrast to [11], we do not set up a drop tolerance to remove nonzero elements in a row. In other words, we do not change the lower triangular sparsity of a Jacobian matrix in the whole computation procedure. As for the full bicolouring method in [7], we use it to recover the whole sparse Jacobian matrix, not just the lower triangular part. Test matrices reside from the Tim Davis collection [9] 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 bicolouring estimation, partial bicolouring estimation and partial one-sided colouring, to all these matrices. The number of matrix–vector products, MV, and the running times are listed in Table 2.

From Table 2, it can be observed that the partial bicolouring technique requires much fewer matrix–vector products than the full bicolouring method in most testing cases. Compared with the one-sided partial colouring technique in [8], the bicolouring technique can reduce the number of matrix–vector multiplications significantly. Figure 1 shows the sparsity patterns of matrices,

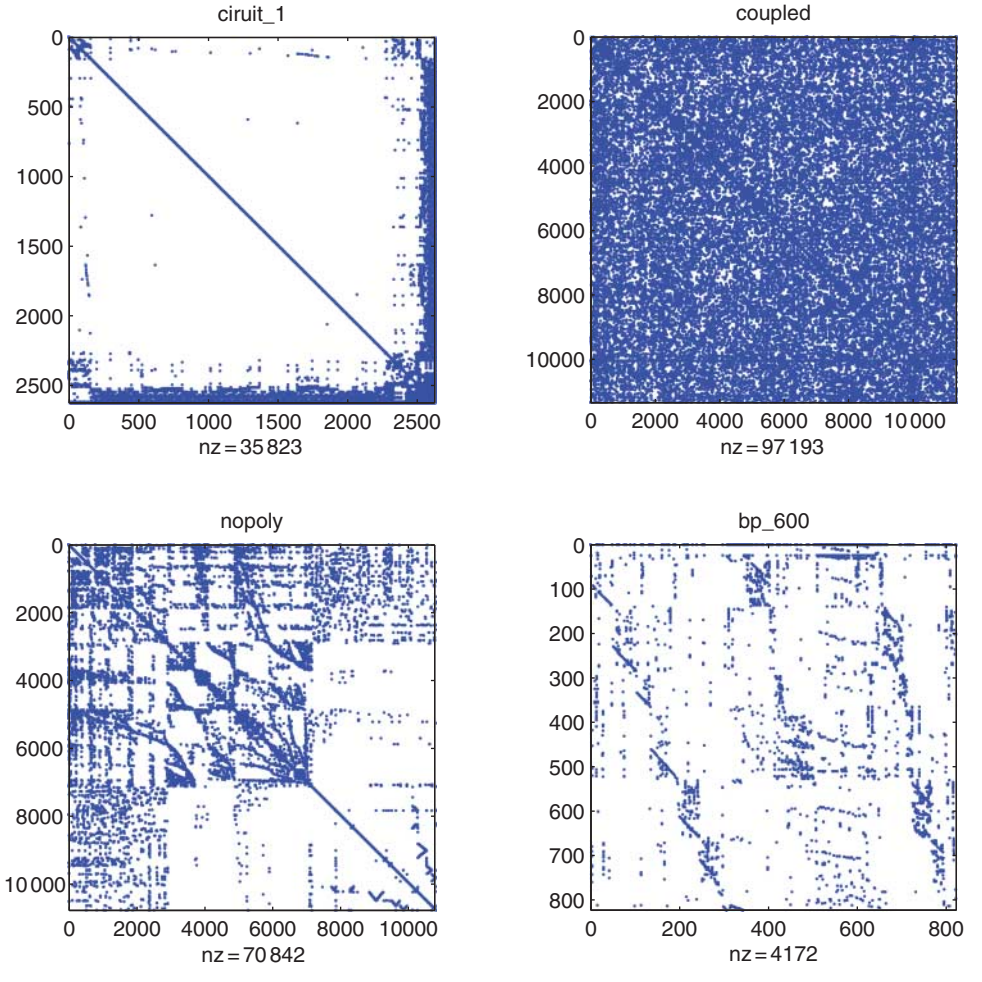

Figure 1. Sparsity patterns for matrices, circuit_1, coupled, nopoly and bp_600.

named circuit_1, coupled, nopoly and bp_600. It illustrates that when there is a dense row in the lower triangular part of the matrix, circuit_1 and coupled for example, our bicolouring partial estimation requires much fewer matrix–vector multiplications than the one-sides colouring method does. While there are no dense rows, say nopoly and bp_600, the bicolouring and one-sided colouring methods require the similar number of matrix–vector multiplications. On the other hand, the running times to determine the seed matrices V and W of a full matrix or partial matrix via bicolouring techniques are similar since both are required to use the MNCO to partition matrix into J_C and J_R. The partial estimation needs a little bit more computations based on Definitions 3.2 and 3.3. As for the partial one-sided colouring approach, it usually takes more time than the bicolouring approach when it requires much more matrix–vector multiplications to recover the lower triangular part. When the numbers of matrix–vector multiplication are similar between the one-sided and bicolouring methods, the bicolouring method usually requires more running times due to the matrix partition MNCO algorithm. However, the MNCO algorithm only runs once for the lower triangular part recovery in the JFNK method. Once the seed matrices V and W are determined, only V and W are required to recover the partial matrix for rest of the iterations.

Next, we test the Newton–Krylov method for solving some typical nonlinear equations [22] based on the AD. In the computational process, we first use ADMAT-2.0 [4] 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 (4) and solve the corresponding Jacobian system by the Krylov method for each iteration until we get the solution for the nonlinear equations. For the comparison, we choose the standard Newton trust region method [23,24]. The only difference between these two methods is the Newton trust region method forms the Jacobian matrix explicitly via AD and solves (2) by '\' in Matlab while the Newton–Krylov method forms the preconditioner through the partial bicolouring Jacobian estimation and solves (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 [22].

(1) *Extended Rosenbrock function:*

$$f_{2l-1}(x) = 10(x_{2l} - x_{2l-1}^2),\ f_{2l}(x) = 1 - x_{2l-1}, \quad \text{where the problem size } n = 2l.$$

(2) *Extended Powell singular function:*

$$f_{4l-3}(x) = x_{4l-3} + 10x_{4l-2}, \quad f_{4l-2}(x) = 5^{1/2}(x_{4l-1} - x_{4l})$$
$$f_{4l-1}(x) = (x_{4l-2} - 2x_{4l-1})^2, \quad f_{4l}(x) = 10^{1/2}(x_{4l-3} - x_{4l})^2,$$

where the problem size n is a multiple of 4.

(3) *Discrete boundary value (DBV) 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

Table 3. The performance comparison of the Newton–Krylov method with bicolouring and one-sided colouring partial Jacobian estimation and standard Powell's dogleg method for four typical nonlinear equations.

	Newton–Krylov				Newton	
Problem	Time	It	MVs	Krylov	Time	It
$n = 2000$						
Powell	0.57	9	3	6	2.79	9
Rosenbrock	0.32	5	2	2	0.58	4
DBV	0.87	5	3	3	2.36	5
Broyden	0.54	5	3	3	1.29	5
$n = 5000$						
Powell	1.76	9	3	6	31.06	9
Rosenbrock	1.28	4	2	2	3.61	4
DBV	3.52	4	3	3	18.30	4
Boyden	2.07	5	3	3	7.88	5
$n = 10,000$						
Powell	5.44	9	3	6	191.91	9
Rosenbrock	3.98	4	2	2	14.29	4
DBV	12.46	4	3	3	84.70	4
Boyden	6.96	5	3	3	31.29	5
$n = 12,000$						
Powell	7.72	9	3	6	287.76	9
Rosenbrock	5.59	4	2	2	21.10	4
DBV	18.35	4	3	3	137.16	4
Boyden	10.19	5	3	3	49.76	5

Notes: Column 'It' for the number of iterations; column 'MVs' for the number of matrix–vector products for each lower triangular matrix estimation; column 'Krylov' for the average number of Krylov iterations at each Newton iteration.

estimate the lower triangular part for the preconditioner construction (4) at each Newton iteration. The results indicate that the preconditioner leads to a fast convergence of the Krylov subspace method for solving (2). Thus, the Newton–Krylov method is superior to the standard Newton method when the problem size is large.

Finally, we analyse a flow problem of a fluid during injection into a long vertical channel [10]. Assume that the flow is modelled by the boundary value problem

$$\begin{aligned} u^{(4)} &= R(u'u'' - uu^{(3)}), \quad 0 \le t \le 1, \\ u(0) &= 0, \quad u(1) = 1, \quad u'(0) = u'(1) = 0, \end{aligned} \tag{5}$$

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

$$0 = t_1 < t_2 < \cdots < t_n < t_{n+1} = 1.$$

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

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

where $k \ge m$ ($m = 4$ 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 (5), and that the differential equation (5) holds on p_i at the collection points,

$$\xi_{ij} = t_i + h\rho_j, \quad \text{for } 1 \le i \le n,\ \ 1 \le j \le 4.$$

The polynomial of order 8 on each subinterval $[t_i,\ t_{i+1}]$ is of the form [11]

$$p_i = \sum_{i=1}^{4} \frac{(t-t_i)^{j-1}}{(j-1)!} v_{ij} + \sum_{j=1}^{4} \frac{(t-t_i)^{j+3}}{(j+3)!h^{j-1}} w_{ij}, \quad t \in [t_i,\ t_{i+1}],\ \ \text{for } 1 \le i \le n. \tag{6}$$

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

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

Furthermore, at the collection points, the differential equation (5) holds, that is

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

Thus, we have $8n$ equations in total to determine $8n$ parameters v_{ij} and w_{ij} in (6). In our experiment, we test four case with $n = 100$, $n = 500$, $n = 800$ and $n = 1000$ with $R = 0$, $R = 1$, $R = 10$ and $R = 50$, respectively. Table 4 provides 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 (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. On the other hand, the bicolouring method requires fewer matrix–vector multiplications than the one-sided method does. It leads to 10–20% time savings as the problem size and R increase. Although the speedup of the

Table 4. The performance comparison of the Newton–Krylov method and standard Powell's dogleg method for a flow channel problem.

	Newton–Krylov bicolouring			Newton–Krylov one-sided colouring		Newton	
Size (n)	Time	It	Krylov	Time	It	Time	It
$R = 0$							
100	3.23	–	3	3.67	–	27.86	–
500	13.41	–	3	15.38	–	134.24	–
800	31.37	–	3	36.81	–	345.34	–
1000	35.46	–	3	41.77	–	749.23	–
$R = 1$							
100	3.31	3	7	3.81	3	27.08	3
500	26.77	3	7	31.89	3	178.87	3
800	94.47	3	7	103.81	3	409.83	3
1000	115.49	3	7	133.62	3	805.34	3
$R = 10$							
100	7.73	7	8	8.43	7	32.34	7
500	33.11	7	8	42.94	7	187.22	7
800	109.94	7	8	124.45	7	318.34	7
1000	165.72	7	8	199.23	7	918.82	7
$R = 50$							
100	11.33	6	17	14.97	6	45.24	6
500	60.37	6	36	74.37	6	198.35	6
800	113.24	6	60	149.97	6	412.22	6
1000	357.78	7	43	443.28	7	1038.77	7

Notes: Column 'it' for the number of iterations; column 'Krylov' for the average number of Krylov iterations at each Newton iteration.

bicolouring approach is not so significant compared with one-sided colouring, the performance of bicolouring approach is still superior to the one-side colouring method. Therefore, from the above five nonlinear problems, it turns out that the bicolouring Newton–Krylov method is much better than the standard Newton method and one-side colouring Newton–Krylov method, especially when the Jacobian matrix is sparse and the problem size is large.

5. Conclusions

The JFNK method is widely used in solving nonlinear equations arising in many applications. However, constructing an economic and effective preconditioner is 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 colouring 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 colouring 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 AD technology developed in the context of sparse Jacobians [6]. In particular, we have proposed to adapt the two-sided bicolouring method developed in [15,27] 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 'flavour' the lower triangular half of the Jacobian (that defines the preconditioner), adapting the colouring procedure to account for both constant (nonzero) values and Jacobian entries that are 'copies', and taking advantage of other structural properties.

Acknowledgements

We are grateful to two anonymous reviewers for their valuable comments and suggestions, which have helped us make the article more precise and readable. This work was supported in part by the Ophelia Lazaridis University Research Chair (held by Thomas F. Coleman), the National Sciences and Engineering Research Council of Canada and the Natural Science Foundation of China (Project No: 11101310).

References

[1] S. Bellavia, D. Bertaccini, and B. Morini, *Nonsymmetric preconditioner updates in Newton-Krylov methods for nonlinear systems*, SIAM J. Sci. Comput. 33 (2011), pp. 2595–2619.
[2] P. Børstad, *Fast numerical solution of the biharmonic Dirichlet problem on rectangles*, SIAM J. Numer. Anal. 20 (1983), pp. 59–71.
[3] C.G. Broyden, *The convergence of an algorithm for solving sparse nonlinear systems*, Math. Comput. 25 (1971), pp. 285–294.
[4] Cayuga Research Associates, LLC, *ADMAT-2.0 User's Guide*, 2009. New York: Cayuga Research Associates.
[5] Y. Chen and C. Shen, *A Jacobian-free Newton-GMRES(m) with adaptive preconditioners and its application for power flow calculations*, IEEE Trans. Power Syst. 21 (2006), pp. 1096–1103.
[6] T.F. Coleman and J.Y. Cai, *The cyclic coloring problem and estimation of sparse Hessian matrices*, SIAM J. Alg. Discrete Methods 7 (1986), pp. 221–235.
[7] T.F. Coleman and A. Verma, *The efficient computation of sparse Jacobian matrices using automatic differentiation*, SIAM J. Sci. Comput. 19 (1998), pp. 1210–1233.
[8] J. Cullum and M. Tůma, *Matrix-free preconditioning using partial matrix estimation*, BIT 46 (2006), pp. 711–729.

[9] T. Davis and Y. Hu, *The university of Florida sparse matrix collection*, ACM Trans. Math. Software 38 (2011), pp. 1–25.
[10] E.D. Dolan, J.J. Morè, and T.S. Munson, *Benchmarking optimization software with COPS 3.0*, Tech. Rep. ANL/MCS-TM-273, Argonne National Laboratory, Argonne, 2004.
[11] J. Duintjer Tebbens and M. Tůma, *Efficient preconditioning of sequences of nonsymmetric linear systems*, SIAM J. Sci. Comput. 29 (2007), pp. 1918–1941.
[12] J. Duintjer Tebbens and M. Tůma, *Preconditioner updates for solving sequences of linear systems in matrix-free environment*, Numer. Linear Algebra Appl. 17 (2010), pp. 997–1019.
[13] G.H.G.R.W. Freund and N.M. Nachtigal, *Iterative solution of linear systems*, Acta Numerica 1 (1992), pp. 57–100.
[14] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[15] A.H. Grebremedhin, F. Manne, and A. Pothen, *What color is your Jacobian? Graph coloring for computing derivatives*, SIAM Rev. 47 (2005), pp. 629–705.
[16] A. Griewank and A. Walther, *Evaluating Derivatives: Principles, and Techniques of Algorithmic Differentiation*, 2nd ed., SIAM, Philadelphia, PA, 2005.
[17] M.H. Gutknecht, *Lanczos-type solvers for nonsymmetric linear systems of equations*, Acta Numerica 6 (1997), pp. 271–397.
[18] M.R. Hestenes and E.L. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bureau Standards, Section B 49 (1952), pp. 409–436.
[19] D.A. Knoll and D.E. Keyes, *Jacobian-free Newton-Krylov methods: A survey of approaches and applications*, J. Comput. Phys. 193 (2004), pp. 357–397.
[20] X. Li and F. Primeau, *A fast Newton-Krylov solver for seasonally varying global ocean biogeochemistry models*, Ocean Model. 23 (2008), pp. 13–20.
[21] E. Mizutani and J .Demmel, *Iterative scaled trust-region learning in Krylov subspaces via Pearl mutters implicit sparse Hessianvector multiply*, in *Advances in Neural Information Processing Systems (NIPS)*, S. Thrun, L.K. Saul, and B. Schölkopf, eds., Vol. 16, 2004, pp. 209–216.
[22] J.J. Moré, B.S. Grabow, and K.E. Hillstrom, *Test unconstrained optimization software*, ACM Trans. Math. Software. 7 (1981), pp. 17–24.
[23] H.B. Nielsen, *immoptibox – A MATLAB TOOLBOX FOR OPTIMIZATION*, IMM, Technical University of Denmark, 2006. Available at http://www2.imm.dtu.dk/hbn/immoptibox/
[24] M.J.D. Powell, *A hybrid method for nonlinear equations*, in *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz, ed., Gordon and Breach, London, 1970, pp. 87–114.
[25] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, 1996.
[26] Y. Saad and H.A. van der Vorst, *Iterative solution of linear systems in the 20th century*, J. Comput. Appl. Math. 123 (2000), pp. 1–33.
[27] W. Xu and T.F. Coleman, *Efficient(partial) determination of derivative matrices via automatic differentiation*, SIAM J. Sci. Comput., to appear in 2012.
[28] W. Xu, T.F. Coleman, and G. Liu, *A secant method for nonlinear least-squares minimization*, J. Comput. Optim. Appl. 51 (2012), pp. 159–173.