ON DECOMPOSING LARGE SPARSE SYSTEMS OF NONLINEAR EQUATIONS

MIRCEA DRAGAN

Abstract. In this paper an algorithm is presented for decomposing large sparse nonlinear systems into smaller subsystems in order to solve them in parallel. The algorithm is independent on the method used to solve the system.

1. Introduction

Solving large nonlinear systems of equations is required by many real-life problems. When the system is large, treating it as a whole is a quite hard task and is time consuming. Most large systems are sparse and we can take advantage of the sparsity by solving the system in parallel.

In this paper an algorithm is presented that can be used to solve in parallel a large sparse nonlinear system. The algorithm uses the mathematical form of the system (as used in mathematical packages like Mathematica or Maple) and can be easily integrated into a mathematical package.

The algorithm is a generalization of the method presented in [8] and is based on Tarjan's algorithm [11]. The algorithm is used for a $n \times n$ system and can be easily extended to a $m \times n$ system.

Section 2 presents motivation and background of the algorithm. In Section 3 the algorithm is presented and Section 4 deals with local convergence and stability issues. Applications to the algorithm are presented in Section 5.

2. Motivation and background

In solving a nonlinear system three situations can occur. The simplest situation is when the system is composed of two or more independent subsystems, as is illustrated in the

---

Date: February 4, 2001.

1991 *Mathematics Subject Classification.* 65Y05, 65H10.

Key words and phrases. Parallel algorithms, asynchronous algorithms, iterative methods, sparse nonlinear system.

following example:

$$\begin{cases} 2x_1^2 - 2\cos(x_1 - 1) & = 0 \\ \frac{x_2}{2}\sin(x_3) + \sin(x_2) & = 0 \\ \tan(x_1) + x_1x_4 & = 0 \\ 2(x_5 - x_6) - 2\cos(x_7) - \sin(x_8 - 1) & = 0 \\ \frac{x_6}{2}\sin(x_5 - x_7) - \sin(x_7) + \cos(x_8) & = 0 \\ x_5 - \tan(x_6) + x_7 - x_8 & = 0 \\ x_5^2 + x_6 - 4x_8 + 2 & = 0 \\ x_1x_2 - 4x_3 + 2 & = 0 \end{cases} \tag{2.1}$$

It is easy to see that the subsystem of equations 1, 2, 3 and 8 is independent from the subsystem of equations 4,5,6 and 7, because they have no variables in common. The situation above is very particular.

A more general situation is reflected by the system

$$\begin{cases} x_1^3 + x_2^3 + 4 & = 0 \\ x_1 \exp\left(x_2 - 1\right) + 2x_1x_2 & = 0 \\ \sin\left(x_3 - x_4\right) + 2x_3 + x_3x_4 - 3 & = 0 \\ x_3x_4 + x_4^2 - 2 & = 0 \\ x_1 \cos\left(x_6 - 2x_5\right) + 1 & = 0 \\ x_5x_6 - 3 & = 0 \end{cases} \tag{2.2}$$

It is easy to see that in this situation we have three subsystem, consisting of equations 1 and 2, 3 and 4, respectively 5 and 6. The subsystem consisting of equations 3 and 4 is independent from the others, while the two others have the unknown x_1 in common.

Another situation can be seen in the system

$$\begin{cases} 2x_1x_2 - 2\cos\left(x_1 - 1\right) - 10 & = 0 \\ \sin\left(x_1 + x_3\right) + \frac{x_2}{2} + \sin\left(x_1 + x_2\right) & = 0 \\ x_4 \tan\left(x_3 + x_4\right) + 12 & = 0 \\ 2x_4 - 5x_3 + 2 & = 0 \\ 2\left(x_5 - x_6\right) - 2\cos\left(x_7\right) - \sin\left(x_8 - 1\right) & = 0 \\ \frac{x_6}{2}\sin\left(x_5 - x_6\right) + \sin\left(x_8 - \sin\left(x_7\right)\right) - 2 & = 0 \\ x_7 - \tan\left(x_6\right) + x_5 - x_8 + 5 & = 0 \\ \tan\left(x_1x_5\right) + x_6 \sin\left(x_5 + x_8\right) + 2 & = 0 \end{cases} . \tag{2.3}$$

In this case we don't have any independent subsystem.

Because the last two situations are similar, we can consider only the first two situations.

To solve a sparse nonlinear system several techniques have been proposed. In [13] is described a Jacobi-type method for solving a sparse nonlinear system, which uses a block diagonal Broyden matrix for solving a nonlinear system. This Broyden matrix can be decomposed using some block partitioning strategies.

In [7] is presented an asynchronous parallel algorithm for solving the nonlinear simultaneous equations $Ax + \phi(x) = 0$, where $A = (a_{ij}) \in L\left(R^n\right)$ and $\phi : R^n \to R^n$ is a diagonal isotone and continuous mapping. This algorithm is based by expressing A, ϕ and x in the block form.

In [9] are described and compared eight algorithms of Newton and quasi-Newton type for solving large sparse systems of nonlinear equations. The sparsity of each of these algorithms are used for solving linear systems which results from applying these algorithms.

In [8] is presented a method for solving a linear system based on Tarjan's algorithm [11], which uses a transversal algorithm based on work by Duff [5]. In the algorithm presented we use this method, but there are some several restrictions resulting from nonlinearity of the system. These restrictions will be described shortly.

Consider the nonlinear operator $F : D \subset R^n \to R^n$, $F = (f_1, f_2, \ldots, f_n)^T$, of class $\mathcal{C}^1$ such that the system $F(x) = 0$ has a solution $x^\star = (x_1^\star, x_2^\star, \ldots, x_n^\star) \in D$ which can be computed with an approximation method. The system above can be written as

$$\begin{cases} f_1\left(x_1, x_2, \ldots, x_n\right) = 0 \\ f_2\left(x_1, x_2, \ldots, x_n\right) = 0 \\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots \\ f_n\left(x_1, x_2, \ldots, x_n\right) = 0 \end{cases} \tag{2.4}$$

We can associate the $(n \times n)$ matrix A to this system such that $a_{ij} = 1$ if unknown x_j appears in function f_i and $a_{ij} = 0$ otherwise. For example, for the system (2.3) the matrix A is

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Also we introduce the following definitions.

Definition 2.1. [8] Given a nonsymmetric $(n \times n)$ matrix A, the digraph associated with A is defined to be the graph $G = (V, E)$ with $|V| = n$ such that $(i, j) \in E$ if and only if $a_{ij} \neq 0$.

Definition 2.2. [11] Let $G = (V, E)$ be a graph. For each vertex $v \in V$, the *adjacency list* for vertex v is the list containing all vertices $w \in V$ such that $(v, w) \in E$. A set of such lists, one for each vertex in G, is called *adjacency structure* of G.

Using this matrix we will use the ideas from $H^\star$ ordering ([8]) to solve a large sparse nonlinear system. In the linear case the technique presented in [8] is based on the linearity of the system and cannot be used in the nonlinear case, so we have to modify the algorithm in order to reflect this case.

3. DECOMPOSING ALGORITHM

In this section we present an algorithm to achieve the decomposition of the system (2.4) into subsystems. The algorithm uses the mathematical representation of the system and gives the subsystems in the same representations. The advantage of using external representation is that the algorithm can be easily integrated into a symbolic system solver. Also the decomposition of a system is independent of any algorithm used for solving the system. The external representation can be achieved by manual coding the equations or by using for example the user interface presented in [4].

To analyze each equation from the system we can use a simplified form of the algorithm presented in [3]:

```
repeat
   read next token
   if token is not function and token is variable
      search variable and remember the line
   endif
until input string is empty
```

The equations are numbered from 1, each equation is considered to be on a single line. The unknowns are numbered also from 1 as they are encountered in each equation. When an unknown is found, it is searched if it was found before. If it was not found, it is considered a new unknown (the next one) and the line number is stored. If the unknown was found in the same line, it is ignored, otherwise the line (equation) number is stored. The process continues until the whole system was examined.

Once we have the line numbers in which each variable appears, we can split the system into the subsystems. First we will construct the digraph G based on matrix A associated with the system (2.4) and the adjacency structure for G. The digraph is constructed using Definition 2.1. Once we have the adjacency structure for G we can split the system into subsystems, which means to determine the biconnected components of G. The algorithm is based on Tarjan's algorithm [11] and is modified to reflect our needs.

The procedure *decompose* has as input the graph G and gives the list s of the subsystems.

The algorithm below is explained shortly. The edges are placed on a stack as they are traversed. When an articulation point is found, the corresponding edges are all on top of the stack. An articulation point is an equation which belongs to two subsystems.

A connected component is in our case a subsystem. In the Tarjan's algorithm, when an edge (u_1, u_2) is removed from the stack, it is added to the current biconnected component. In our case it means that we add equations u_1 and u_2 to the current subsystem, if they are not yet added.

When procedure **biconnect** terminates, the edge stack must be empty. If the edge stack is not empty, we have another subsystem.

```
proc decompose($G, s$)
   empty the edge stack
   construct the adjacency structure of $G$
   for $w$ a vertex do
      if $w$ is not yet numbered then
         biconnect($w, 0$)
      endif
      if edge stack is not empty then
      empty the edge stack and get a new subsystem
    endif
  endfor
  identify equations for each subsystem
  determine articulation points which belongs to two subsystems
end
```

```
proc biconnect($v, u$)
  $number[v] = ++i$
  $lowpt[v] = number[v]$
  for $w$ in the adjacency list of $v$ do
    if $w$ is not yet numbered then
      add $(v, w)$ to edge stack
      biconnect($w, v$)
      $lowpt[v] = \min(lowpt[v], lowpt[w])$
      if $lowpt[w] \geqslant number[v]$ then
        start new subsystem
        while new top edge $e = (u_1, u_2)$ on edge stack
                    has $number[u_1] \geqslant number[w]$ do
          delete $(u_1, u_2)$ from edge stack
          if equation $u_1$ is not yet in current subsystem
            add equation $u_1$ to current subsystem
          endif
          if equation $u_2$ is not yet in current subsystem
            add equation $u_2$ to current subsystem
          endif
        enddo
        delete $(v, w)$ from edge stack
        if equation $v$ is not yet in current subsystem
          add equation $v$ to current subsystem
        endif
        if equation $w$ is not yet in current subsystem
          add equation $w$ to current subsystem
        endif
      endif
    else if $(lowpt[w] \geqslant number[v])$ and $(w \neq u)$ then
        add $(v, w)$ to edge stack
        $lowpt[v] = \min(lowpt[v], number[w])$
    endif
  endfor
end
```

It is obvious that, in the worst case, when the system (2.1) is dense, we get the whole system, which means that the system cannot be decomposed into smaller ones using this technique.

In the section 5 some applications of the algorithm are presented.

4. LOCAL CONVERGENCE AND STABILITY ISSUES

Before studying the stability we need the definition of an *asynchronous iterative method* and the definition given below is similar to the definition in [2] and [6]. Such a method is used to solve the equation

$$x = F\left(x\right). \tag{4.1}$$

It is obvious that this system and the system $F\left(x\right) = 0$ can be transformed into each other.

Definition 4.1. Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be an operator. An *asynchronous iteration* corresponding to the operator F and starting with a given vector $x\left(0\right)$ is a sequence $x\left(j\right)$, $j = 0, 1, \ldots,$ of vectors of $\mathbb{R}^n$ defined recursively by

$$x_i\left(j\right) = \left\{ \begin{array}{ll} x_i\left(j-1\right), & i \notin J_j \\ f_i\left(x_1\left(s_1\left(j\right)\right), \ldots, x_n\left(s_n\left(j\right)\right)\right), & i \in J_j \end{array} \right. ,$$

where $\mathcal{J} = \left\{J_j : j = 1, 2, \ldots\right\}$ is a sequence of nonempty subsets of $\left\{1, \ldots, n\right\}$ and $\mathcal{S} = \left\{s_1\left(j\right), \ldots, s_n\left(j\right) : j = 1, 2, \ldots\right\}$ is a sequence of elements in $\mathbb{N}^n$ such that for every $i = 1, \ldots, n$,

(a) $s_i\left(j\right) \leqslant j - 1$, $j = 1, 2, \ldots$;
(b) $\lim\limits_{j\to\infty} s_i\left(j\right) = \infty$;
(c) i occurs infinitely many often in the sets J_j, $j = 1, 2, \ldots$.

This definition can be applied in the case when the system can be decomposed into independent subsystems.

In the parallel evaluation of $F\left(x\right)$, the idea is to allow different processes to compute different subsets of the components. This idea corresponds to the decomposition given by the algorithm.

A generalization of Definition 4.1 is described similar as in [2] and [6]. Given $F : \left(\mathbb{R}^n\right)^m \to \mathbb{R}^n$ an operator, the problem is to find a vector $\xi \in \mathbb{R}^n$, called such that

$$\xi = \lim_{\left(x^1\to\xi,\ldots,x^m\to\xi\right)} F\left(x^1, \ldots, x^m\right) \tag{4.2}$$

The vector ξ is called *fixed point* for the operator F. Similar to asynchronous iterative methods to solve equation (4.1) we introduce the class of *asynchronous iterative method with memory* to solve equation (4.2).

Definition 4.2. Let $F : \left(\mathbb{R}^n\right)^m \to \mathbb{R}^n$ be an operator. An *asynchronous iteration with memory* corresponding to the operator F and starting with a given set of vectors $x\left(0\right), \ldots, x\left(m-1\right)$ is a sequence $x\left(j\right)$, $j = 1, 2, \ldots,$ of vectors of $\mathbb{R}^n$ defined for $j = m, m+1, \ldots$ by

$$x_i\left(j\right) = \left\{ \begin{array}{ll} x_i\left(j-1\right), & i \notin J_j \\ f_i\left(z^1, \ldots, z^m\right), & i \in J_j \end{array} \right. ,$$

where z^r, $1 \leqslant r \leqslant m$, is the vector with components $z_i^r = x_i\left(s_i^r(j)\right)$, $1 \leqslant i \leqslant n$, $\mathcal{J} = \{J_j : j = m, m+1, \ldots\}$ is a sequence of nonempty subsets of $\{1, \ldots, n\}$ which correspond to the subsets of components evaluated at each step and

$$\mathcal{S} = \left\{\left(s_1^1(j), \ldots s_n^1(j), s_1^2(j), \ldots, s_n^m(j)\right) : j = m, m+1, \ldots\right\}$$

is a sequence of elements in $(\mathbb{N}^n)^m$ which satisfies condition (c) from Definition 4.1 and

(a) $\max\{s_i^r(j) : 1 \leqslant r \leqslant m\} \leqslant j-1$ for $j = m, m+1, \ldots$,

(b) $\lim\limits_{j\to\infty} \min\{s_i^r(j) : 1 \leqslant r \leqslant m\} = \infty$.

This definition can be applied when the subsystems obtained by applying the algorithm described in the previous section are not independent.

Suppose that the system $F(x) = 0$ can be decomposed by the above described algorithm into $p \geqslant 1$ smaller subsystems, $F = (F_1, F_2, \ldots F_p)^T$. Then the system is equivalent with

$$F_1\left(x^1\right) = 0, \ldots, F_p\left(x^p\right) = 0, \tag{4.3}$$

where $x = (x^1, \ldots, x^p)$ and x^i, $1 \leqslant i \leqslant p$, contains the unknowns of the subsystem i, $1 \leqslant i \leqslant p$, such that if a variable appears in two subsystems it is taken only once.

Using the definitions and notations above we can give the following theorem

Theorem 4.1. *If the system $F(x) = 0$ has a solution $x^\star = (x_1^\star, x_2^\star, \ldots, x_n^\star) \in D$ which can be computed with an approximation method, then the subsystems (4.3) has the solutions*

$$F_1\left(x^{1\star}\right) = 0, \ldots, F_p\left(x^{p\star}\right) = 0, \tag{4.4}$$

where $x^\star = \left(x^{1\star}, \ldots, x^{k\star}\right)$.

Proof. If $p = 1$ the theorem is trivial. If $p > 1$ we can have one of the situations as in systems (2.1), (2.2) or (2.3). The situations described in systems (2.2) and (2.3) are similar and we will consider only the system (2.2).

From [2] and [6] it results that the system $F(x) = 0$ can be solved using asynchronous iterations and asynchronous iterations with memory. We have to show that the subsystems given by the algorithm are particular cases of asynchronous iterations or asynchronous iterations with memory.

Without loosing generality we can suppose $p = 2$ (i.e. the system can be decompose into two subsystems).

If we have the situation (2.1) then each subsystem is independent on the others and can be solved separately. In this case the system can be solved using asynchronous iterations and each subsystem can be solved on a different processor.

If we have the situation (2.2) then there is only one unknown which appears in both subsystems as it results from the algorithm, and all other unknowns appears in only one subsystem. In this case the system can be solved using asynchronous iterations with memory. Each subsystem can be solved on a different processor and uses for the common unknown whatever value is currently available when needed. □

Conjecture 4.2. *Let $F(x) = 0$ be a nonlinear subsystem which can be decomposed with the algorithm from the Section 3 into subsystems (4.3). If x^0 is a starting value of an approximation method for the solution $x^\star$ of the equation $F(x) = 0$, then, starting from the $x^0 = (x^{1,0}, x^{2,0}, \ldots, x^{p,0})$ with the same approximation method, subsystems (4.4) have the solution $\left(x^{1\star}, \ldots, x^{k\star}\right) = x^\star$.*

5. Applications

To illustrate the decomposition above, we will consider some examples. First, consider the system (2.1). This system can be decomposed in the subsystems

$$\begin{cases} 2x_1^2 - 2\cos(x_1 - 1) & = 0 \\ \frac{x_2}{2}\sin(x_3) + \sin(x_2) & = 0 \\ \tan(x_1) + x_1x_4 & = 0 \\ x_1x_2 - 4x_3 + 2 & = 0 \end{cases}$$

$$\begin{cases} 2(x_5 - x_6) - 2\cos(x_7) - \sin(x_8 - 1) & = 0 \\ \frac{x_6}{2}\sin(x_5 - x_7) - \sin(x_7) + \cos(x_8) & = 0 \\ x_5 - \tan(x_6) + x_7 - x_8 & = 0 \\ x_5^2 + x_6 - 4x_8 + 2 & = 0 \end{cases}$$

Each of those subsystems can be solved on a different processor. The system (2.2) can be decomposed by the algorithm in three subsystems,

$$\begin{cases} x_1^3 + x_2^3 + 4 & = 0 \\ x_1 \exp\left(x_2 - 1\right) + 2x_1x_2 & = 0 \\ x_1 \cos\left(x_6 - 2x_5\right) + 1 & = 0 \end{cases}$$

$$\begin{cases} \sin\left(x_3 - x_4\right) + 2x_3 + x_3x_4 - 3 & = 0 \\ x_3x_4 + x_4^2 - 2 & = 0 \end{cases}$$

$$\begin{cases} x_1 \cos\left(x_6 - 2x_5\right) + 1 & = 0 \\ x_5x_6 - 3 & = 0 \end{cases}$$

The system (2.3) is decomposed in three subsystems,

$$\begin{cases} 2x_1x_2 - 2\cos\left(x_1 - 1\right) - 10 & = 0 \\ \sin\left(x_1 + x_3\right) + \frac{x_2}{2} + \sin\left(x_1 + x_2\right) & = 0 \\ \tan\left(x_1x_5\right) + x_6 \sin\left(x_5 + x_8\right) + 2 & = 0 \end{cases}$$

$$\begin{cases} \sin\left(x_1 + x_3\right) + \frac{x_2}{2} + \sin\left(x_1 + x_2\right) & = 0 \\ x_4 \tan\left(x_3 + x_4\right) + 12 & = 0 \\ 2x_4 - 5x_3 + 2 & = 0 \end{cases}$$

$$\begin{cases} 2\left(x_5 - x_6\right) - 2\cos\left(x_7\right) - \sin\left(x_8 - 1\right) & = 0 \\ \frac{x_6}{2}\sin\left(x_5 - x_6\right) + \sin\left(x_8 - \sin\left(x_7\right)\right) - 2 & = 0 \\ x_7 - \tan\left(x_6\right) + x_5 - x_8 + 5 & = 0 \\ \tan\left(x_1x_5\right) + x_6 \sin\left(x_5 + x_8\right) + 2 & = 0 \end{cases}$$

Is easy to see that neither of these systems is independent on the others and the equations 2 and 5 from the system (2.3) links these subsystems.

6. Conclusions and future work

For a nonlinear system most algorithms deals with a particular form of the system. The algorithm presented can be used for any form of a system (in particular, it can be used even for a linear system).

If one of the subsystems has a large cycle, the algorithm cannot deal with it. Also in practice it is possible that the system could be decomposed into subsystems which have in common more than one unknown and in such a situation the system cannot be decomposed into subsystems using the algorithm. In such situations the system could be decomposed into smaller ones following the idea in [8], by introduction the notion of *separator set*. The method presented in [8] cannot be applied as it is and has to be adapted to the nonlinear case.

References

[1] A. Basermann, B. Reichel, C. Schelthoff, *Preconditioned CG methods for sparse matrices on massively parallel machines*, Parallel Computing 23 (1997), 381-398.

[2] G.M. Baudet, *Asynchronous iterative methods for multiprocessors*, Journal of ACM 25 (2) (1978), 226-244.

[3] M. Dragan, *Using the polish reverse notation in solving nonlinear systems equations*, Anal. Univ. Timisoara, (to appear).

[4] M. Dragan, *A user interface for solving nonlinear systems*, Anal. Univ. Timisoara, (to appear).

[5] I.S. Duff, *On algorithms for obtaining a maximum transversal*, ACM Trans. Math. Software 7 (3) (1981), 315-330.

[6] M.N. El Tarazi, *Some convergence results for asynchronous algorithms*, Numer. Math. 39 (1992), 325-340.

[7] D.J. Evans, W. Deren, *An asynchronous parallel algorithm for solving a class of nonlinear simultaneous equations*, Parallel Computing 17 (1991), 165-180.

[8] K.A. Gallivan, B.A. Marsolf, H.A.G. Wijshoff, *Solving large nonsymmetric sparse linear systems using MCSPARSE*, Parallel Computing 22 (1996), 1291-1333.

[9] M.A. Gomes-Ruggiero, J.M. Martinez, A.C. Moretti, *Comparing algorithms for solving sparse nonlinear systems of equations*, SIAM J. Sci. Stat. Comput. 13 (2) (1992), 459-483.

[10] S.A. Savari, D.P. Bertsekas, *Finite termination of asynchronous iterative algorithms*, Parallel Computing 22 (1996), 39-56.

[11] R. Tarjan, *Depth-first search and linear graph algorithms*, SIAM J. Comput. 1 (2) (1972), 146-160.

[12] J.N. Tsitsiklis, *On the stability of asynchronous iterative processes*, Math. Systems Theory 20 (1987), 137-153.

[13] G. Yang, L.C. Dutto, M. Fortin, *Inexact block Jacobi-Broyden methods for solving nonlinear systems of equations*, SIAM J. Sci. Comput., 18 (5) (1997), 1367-1392.

SOFTWARE COMPETENCE CENTER, SOFTWAREPARK HAGENBERG, HAUPTSTRASSE 99, A-4232 HAGENGERG, AUSTRIA

E-mail address: `mircea.dragan@lscch.at`