Parallel Newton methods for sparse systems of nonlinear equations

GIOVANNI ZILLI and LUCA BERGAMASCHI

Abstract

In this paper we give the results found in solving consistent sparse systems of nonlinear equations by an inexact Newton and Quasi-Newton method both combined with a block iterative row-projection linear solver of Cimmino-type. A simple partitioning of the Jacobian matrix was used for solving two nonlinear test problems, that is a tridiagonal problem of size $n = 131072$ and a nonlinear Poisson problem with $n = l \times l$ grid with l up to 64. The results are obtained on the CRAY T3E installed at CINECA (Bologna, Italy) with 32 nodes. The Fortran code runs under MPI implementation.

1 Introduction

We are interested in the numerical solution of a system of nonlinear equations

$$F(x) = 0 \qquad F = (f_1, ..., f_n)^T \tag{1}$$

where $F : R^n \to R^n$ is a nonlinear C^1 function, and its Jacobian matrix J(x) is sparse and n is large, by an inexact Newton and Quasi-Newton method combined with a block iterative row-projection linear solver of Cimmino-type (truncated or inexact versions, in the sense of [3], [4]).

This paper is the continuation of [12] where some preliminary results were found. We will shortly outline the block iterative method of Cimmino-type. For more information, convergence results and references see [1], [2] and [10].

Let us consider the linear system $Ax = b$ where A is $n \times n$ sparse, generally unstructured, matrix. *Partition* A into p row blocks : $A^T = [A_1, A_2, \ldots, A_p]$, where $1 \le p \le n$, and partition the vector b conformally. The row projection algorithm can be derived noting that the i^{th} block of equations $A_i x = b_i$, where A_i is $m \times n$ matrix, with $rank(A_i) = m$, and $m \le n$, defines a linear manifold and the vector solution x is the intersection point of the p manifolds. The system $A_i x = b_i$ is underdetermined and has infinitely many solutions. We are interested in the minimum norm solution, that is, the (unique) vector $x^* \in \mathcal{R}(A_i^T)$. For $z \in R^m$, we have $x^* = A_i^T z$, therefore $A_i A_i^T z = b_i$ and $x^* = A_i^+ b_i$, where $A_i^+ = A_i^T (A_i A_i^T)^{-1}$ is the Moore-Penrose pseudo-inverse of A_i. From the usual decomposition:

$$x = P_{\mathcal{R}(A_i^T)} x + P_{\mathcal{N}(A_i)} x = A_i^+ b_i + (I - A_i^+ A_i) x = x + (A_i^+ b_i - A_i^+ A_i x)$$

applying the Richardson method, $x_{k+1} = x_k + r_k$ where r_k is the residual vector, to the (equivalent) preconditioned system of linear equations $HAx = Hb$, where

$$H = [A_1^+, \dots, A_i^+, \dots, A_p^+],$$

i.e. $\sum_{i=1}^p A_i^+ A_i x = \sum_{i=1}^p A_i^+ b$ we find the iterative scheme:

$$x_{k+1} = x_k + \omega \sum_{i=1}^{p} A_i^+ \left(b_i - A_i x_k\right) = x_k + \omega \sum_{i=1}^{p} \delta_{i,k} \tag{2}$$

$\omega \in R^+$ being an acceleration parameter. The p pseudo-residual vectors $\delta_{i,k} = A_i^+ \left(b_i - A_i x_k\right)$ (and the orthogonal projections onto the manifolds) can be calculated and then added, so the algorithm works well with *parallel computers.*
Let D be the diagonal matrix extracted from $A_i A_i^T$, i.e. :

$$D = \mathrm{diag}(\|a_1\|_2^2, \|a_2\|_2^2, \dots, \|a_m\|_2^2) \tag{3}$$

where a_r, $1 \le r \le m$, is the r^{th} row of A_i. If we take the *relaxation matrix* $\Phi = \omega\,(A_i A_i^T)^{-1}$ equal to a diagonal matrix defined by $\Phi = \omega\, D^{-1}$, where D is the diagonal matrix defined in equation (3), the iterative row-projection method described in equation (2) includes Kaczmarz's method in the extreme case $p = n$ (n blocks of a single row), and Cimmino's method in the case $p = 1$ (a single block of n rows). For $p = 1$, with A an $m \times m$ matrix, we get

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \omega A^+ \left(b - Ax^{(k)}\right) x^{(k)} + A^T(\omega\,(AA^T)^{-1}) \left(b - Ax^{(k)}\right) \\ &= x^{(k)} + A^T \Phi \left(b - Ax^{(k)}\right) = x^{(k)} + A^T \omega\, D^{-1} \left(b - Ax^{(k)}\right). \end{aligned}$$

For $\omega = \dfrac{2m_h}{\sum_{h=1}^m m_h}$ we retrieve the original Cimmino method

$$x^{(k+1)} = x^{(k)} + \frac{2}{\displaystyle\sum_{h=1}^{m} m_h} \sum_{h=1}^{m} m_h \frac{b_h - \langle a_h, x^{(k)} \rangle}{\|a_h\|_2^2} a_h.$$

A particular case is with $m_h = \dfrac{1}{m} \Longrightarrow \displaystyle\sum_{h=1}^{m} m_h = 1, \quad \omega = \frac{2}{m}.$

Convergence to the minimum norm solution is obtained under the assumption that $b \in \mathcal{R}(A)$, $x^{(0)} \in \mathcal{R}(A^T)$, and $0 < \omega < 2/\rho(E)$, $(0 < \omega < 2)$, where $\rho(E)$ is the spectral radius of the matrix $E = \sum_{i=1}^p A_i^+ A_i$. The convergence rate can be slow even with the optimal parameter ω. Since the sum of the orthogonal projectors $\sum_{i=1}^p A_i^+ A_i$ is a symmetric positive definite operator, *conjugate gradient acceleration* (C.G.) can be used to solve the preconditioned system $HAx = Hb$, $(\omega = 1)$. The p linear least squares subproblems (underdetermined) in the pseudoresidual unknown $\delta_{i,k}$

$$A_i \delta_{i,k} = \left(b_i - A_i x_k\right), \quad 1 \le i \le p$$

must be solved at *each* conjugate gradient iteration. To facilitate the construction of the matrices $(A_i A_1^T)^{-1}$, suitable partitionings of the matrix A may be adopted. This would allow to simplify the solution of the least squares subproblems at each step of the inner iteration. In this paper, the natural partitioning was used, according to which each matrix is partitioned into p $(1 \leq p \leq n)$ almost equal-sized blocks A_i of size $m_i \times n$, with $\sum_{i=1}^{p} m_i = n$. Then we solve *concurrently* the p linear least squares subproblems with the iterative Lanczos algorithm LSQR [8]. Some limitations in terms of speedup efficiency were detected with this solver (see in the section 3), in that it is an essentially sequential procedure, except the BLAS2 kernels (sparse mat-vet). In order to increase the speedup efficiency of the procedure, experiments with other partitionings (see [10]) and solvers (the sparse QR solver [7]) are presently under investigation.

2 The inexact Newton methods

Now let us turn to the equations (1). Combining the *inexact Newton* method [4] and the block Cimmino method of section 1 we obtain the block inexact Newton-Cimmino algorithm (see also [11], [12]), in which at a major (Newton) iteration k, the linear system

$$J(x_k)\ z_k = -F(x_k) \qquad x_{k+1} = x_k + z_k, \tag{4}$$

where $J(x^*)$ is the Jacobian matrix, is solved in *parallel* by the block Cimmino method with the p partitioning $J^T = [J_1, J_2, \ldots, J_p]$, $F^T = [F_1, F_2, \ldots, F_p]$.
The outer exit test is based on the relative residual:

$$\|F(x_k)\| \leq \varepsilon_1\ \|F(x_0)\|. \tag{5}$$

The *inner* iteration (C.G.-Cimmino), runs up to a value m_k such that for the relative residual the following inequality is retained: $\|r_{k,m}\| \leq \varepsilon_2 \|F(x_k)\|$. The LSQR solver also stops at iteration $i_{m,k} = i_k$ when a relative residual test with a tolerance ε_3 is verified. These choices may be problem-dependent, [4], [9]. Relative to the inexact Newton method, in [4] it is proved that the method has a local, linear convergence in an appropriate norm, and it may converge superlinearly or even quadratically under convenient assumptions.
Following [12], we may also use the block Cimmino method as an inner solver in an inexact *Quasi-Newton* method for solving (1). This allows us to solve the inner linear systems with the same coefficient matrix A ($= J(x_k) = J(x_0)$, for every k) at every outer iteration k. For this purpose we consider the following inexact Quasi-Newton Cimmino method.
Quasi-Newton methods obey the formulae

$$B_k z_k = -F(x_k) \qquad x_{k+1} = x_k + z_k,$$

where the matrix $B(x_k)$ is an approximation to the Jacobian matrix $J(x_k)$ chosen in such a way to satisfy the *secant equation*

$$B_{k+1} z_k = y_k \qquad y_k = F(x_{k+1}) - F(x_k).$$

According to the Broyden secant update

$$B_{k+1} = B_k + \frac{(y_k - B_k z_k) z_k^T}{z_k^T z_k} = B_k + \frac{F(x_{k+1}) z_k^T}{z_k^T z_k}, \quad B_0 = J(x_0),$$

by induction, it becomes

$$B_{k+1} = B_0 + \sum_{j=0}^{k} \frac{F(x_{j+1}) z_j^T}{z_j^T z_j}.$$

At iteration k we have

$$\left(B_0 + \sum_{j=0}^{k-1} \frac{F(x_{j+1}) z_j^T}{z_j^T z_j} \right) z_k = -F(x_k).$$

We obtain the *step* z_k as

$$z_k = \left[\sum_{j=0}^{k-1} \frac{-B_0^{-1} F(x_{j+1})}{z_j^T z_j} \; z_j^T z_k \right] + \left[-B_0^{-1} F(x_k) \right]. \tag{6}$$

Finally, we solve the linear equations

$$z_i^T z_k + \sum_{j=0}^{k-1} \frac{z_i^T B_0^{-1} F(x_{j+1})}{z_j^T z_j} z_j^T z_k = z_i^T \left(-B_0^{-1} F(x_k) \right).$$

in the unknowns $c_i = z_i^T z_k$, for $0 \leq i \leq k-1$. Hence the step z_k is obtained according to the equation (6). We note that at each outer iteration k the coefficient matrix of this system is obtained simply by bordering the previous one. Moreover the vectors z_k and $B_0^{-1} F(x_k)$, i.e. the vectors $\{(z_0, z_1, \ldots, z_{k-1})\}$ and $\{(B_0^{-1} F(x_1), \ldots, B_0^{-1} F(x_k))\}$ are stored. These last vectors are calculated in *parallel*, by solving a linear system of size n with the row-projection Cimmino method.

As in the inexact Newton case [4], *it can be proved* [13] that the method has a local, linear convergence in the norm $||x||_{HA} = \sqrt{x^T H A x}$, and it may converge superlinearly too.

3 Computational aspects and numerical tests

The two procedures have been tested with the following nonlinear sparse problems $F(x) = (f_1, ..., f_n)^T = 0$:

1. Broyden tridiagonal problem [5], with h a real parameter:

$$\begin{aligned} f_1(x) &= (3 - h x_1) x_1 - 2 x_2 + 1 = 0, \\ f_i(x) &= -x_{i-1} + (3 - h x_i) x_i - 2 x_{i+1} + 1 = 0, \quad i = 2, \ldots, n-1, \\ f_n(x) &= -x_{n-1} + (3 - h x_n) x_n + 1 = 0. \end{aligned}$$

2. Poisson problem, obtained by application of the finite difference method to the two-dimensional boundary problem:

$$\Delta u = \frac{u^3}{1+x^2+y^2}, \qquad 0 \le x \le 1, \qquad 0 \le y \le 1,$$

$$u(0,y) = 1, \qquad u(1,y) = 2 - e^y, \qquad 0 \le y \le 1,$$

$$u(x,0) = 1, \qquad u(x,1) = 2 - e^x, \qquad 0 \le x \le 1,$$

by discretizing the equation by the 5-point formula on a $l \times l$ grid with a total number of unknowns equals to $n = l \times l$. The resulting linear system is, as we know, a tridiagonal l-block system with every l-blocks of size l.
To ensure that each processor does the same amount of work, the Jacobian matrix is partitioned into p almost equal-sized blocks, where $p = 1, 2, 4, 8, 16$ and 32, according to the number of processors used.
The p blocks partitioning is described in Table 1, relative to the tridiagonal problem with $n = 131072$, and in Table 2, relative to Poisson's problem with $l = 64$ (and $n = 4096$). The results are obtained on the CRAY T3E installed at CINECA

MATRIX	$N = 131072$ ($= 2^{17}$)
p= 1	131072 (n_1 = 393214)
p= 2	65536 (n_1 =196607, n_1 =196607)
p= 4	32768 (n_1 =98303, n_2 =98304, n_1 =98303)
p= 8	16384 (n_1 =49151, n_2 =49152, n_1 = 49151)
p= 16	8192 (n_1 =24575, n_2 = 24576, n_1 = 24575)
p= 32	4096 (n_1 = 12287, n_2 = 12288, n_1 = 12287)

Table 1: Number of rows of every block of the partinioning on each of the p nodes and number n_1 of non-zero elements (in parentheses) on the first and last nodes, and number n_2 on the remaining $(p-2)$ (with $p > 2$) intermediate nodes respectively, relative to the *tridiagonal* problem with $N = 131072$.

(Bologna, Italy). The Fortran code runs under **MPI** implementation. We show the results obtained on a maximum of 32 processors.

1. Results for the tridiagonal problem
The results found with the inexact Newton and Quasi-Newton methods are summarized in Tables 3 and 4, respectively. The test parameters used are: $h = 2$, $x_0 = (-1, \ldots, -1)^T$. The time refers to the iterative part of the code: Newton outer iterations with updating of the Jacobian matrix (for inexact Newton case only) and the right hand side vector. At each nonlinear iteration k, m_k Cimmino iterations are performed, each of them essentially consisting of a conjugate gradient procedure in which the pseudoresiduals $\delta_{i,k}$ are *concurrently* computed by i_k LSQR

MATRIX	$l = 64$	$N = l \times l = 4096$
p= 1	1 l-blocks	4096 (n_1 =20224)
p= 2	1/2 l-blocks	2048 (n_1 =10112, n_1 =10112)
p= 4	1/4 l-blocks	1024 (n_1 =5024, n_2 =5088, n_1 =5024)
p= 8	1/8 l-blocks	512 (n_1 =2480, n_2 =2544, n_1 =2480)
p= 16	1/16 l-blocks	256 (n_1 =1208, n_2 =1272, n_1 =1208)
p= 32	1/32 l-blocks	128 (n_1 =572, n_2 =636, n_1 =572)

Table 2: Number of rows of every block of the partinioning on each of the p nodes and number n_1 of non-zero elements (in parentheses) on the first and last nodes, and number n_2 on the remaining $(p-2)$, (with $p > 2$) intermediate nodes respectively, relative to the two grids of the *Poisson problem* of size $N = l \times l$.

iterations. The sparse Jacobian matrix is stored by row, according to the *compressed sparse row* (CSR) format. From Tables 3 and 4, where the elapsed time for BLAS routines called by LSQR are also given, we can see that the consumed time of the algorithm is essentially due to the LSQR solver. This will be repeated in all experiments.
The key to our procedure to be successful relies mainly on the sparse solver of the underdetermined system of linear equations. From Tables 3 and 4 we see that only the sparse *matvet* routine (BLAS2 kernel) gives the expected speedup, while BLAS1 kernels of length n are to be parallelized further on.

We note now that the parallelism of the whole algorithm depends essentially on the performance of the linearized system solver, at every nonlinear iteration. Therefore the speedup obtained with only one nonlinear iteration does not change whatever linear iterations the Newton method requires to achieve convergence. For this reason we show here the parallel performance of the block Cimmino method when applied to a *linear* test problem $Ax = b$, where A is the nonsymmetric sparse SAMEH matrix. It is obtained by a 5-point finite difference discretization of the following PDE:

$$-u_{xx} - u_{yy} + 1000\, e^{xy}\, (u_x - u_y) = g, \qquad u = x + y,$$

on a 64×64 grid. The symmetric part of A is not positive definite. It is known (see [6], [10]) that the restarted Krylov subspace methods stagnate or fail to converge with this problem. Instead, convergence is attained by the CG accelerated block Cimmino method. The right hand side is chosen so that the solution is $x = (1, 2, \ldots, n)^T$. Starting from a zero initial guess, we stop the iterations as soon as the relative residual norm $||r_m||/||r_0||$ is smaller than a prescribed tolerance ε_2. Taking advantage of experiments on the convergence of the two nonlinear problems, we could set the tolerance ε_2 to a not too small accuracy ($\varepsilon_2 = 10^{-3}$ in our test). On the contrary, the LSQR routine has to accurately solve the underdetermined subproblems and so we set $\varepsilon_3 = 10^{-12}$. The results are shown in

N=131072	$p=1$	$p=2$	$p=4$	$p=8$	$p=16$	$p=32$
Time	126.2 (1)	79.4 (1.6)	58.6 (2.2)	47.5 (2.7)	42.2 (3.0)	40.6 (3.2)
k it.	4	4	4	4	4	4
m_k it.	2	2	2	2	2	2
i_k it.	30	30	30	30	30	30
$\frac{\|F(x_k)\|}{\|F(x_0)\|}$	$O(10^{-6})$	$O(10^{-6})$	$O(10^{-6})$	$O(10^{-6})$	$O(10^{-6})$	$O(10^{-6})$
LSQR	122.7	76.9	56.2	45.0	39.8	37.9
	97%	97%	96%	95%	94%	93%
Blas 1	57.3	42.1	36.0	31.9	30.3	29.5
	47%	55%	64%	71%	76%	78%
Blas 2	59.6	29.0	14.5	7.3	3.7	1.9
	49%	38%	26%	16%	9%	5%
Speedup (Blas1)	1	1.4	1.6	1.8	1.9	1.9
Speedup (Blas2)	1	2.0	4.1	8.0	15.7	30.0

Table 3: Time (in seconds) obtained with $p = 1, 2, 4, 8$, 16 and 32 nodes, for solving the *tridiagonal* problem with $N = 131072$ ($= 2^{17}$) on Cray T3E with the *Inexact Newton-Cimmino method*, with $k_{\max} = 20$, $m_k = 2$, $i_k = 30$ and $\varepsilon_1 = 10^{-6}, \varepsilon_2 = 10^{-5}, \varepsilon_3 = 10^{-12}$. Speedup values are given in parentheses. Time (and percentages) spent due to LSQR routine and the Blas routines are also given (their percentages are referred to LSQR time).

Table 5. From this Table it is worth noting that with $p = 1$ the Cimmino method acts as an *exact* preconditioner applied to $n \times n$ problem and hence the solution is attained just at the first iteration (with $i = 2258$ LSQR iterations). With $p \geq 2$, the number of Cimmino iterations may increase largely, causing a corresponding worsening of the CPU time. In particular this happens with problems involving the Laplace equation for which it is known that the conditioning requires many C.G. iterations (as we will see, this also occurs with the Poisson test problem). It is therefore reasonable to evaluate the performance of the parallel Cimmino method with respect to the T_2 CPU time ($p = 2$). The speedups reported in Table 5 are consistently computed as $S_p = T_p/T_2$, $p \geq 2$. The promising speedup values shown in the table account for the fact that both the numbers of LSQR iterations and the CPU time which they require, decrease with growing p, opposed to an increasing number of Cimmino iterations necessary to solve the underdetermined subproblems. This behaviour is summarized in Table 5 by the number of LSQR iterations which are shown to decrease, starting from the $p = 2$ case on. We finally note that with 8 processors, we obtain a total CPU time which is also less than the $p = 1$ CPU time.

N=131072	$p=1$	$p=2$	$p=4$	$p=8$	$p=16$	$p=32$
Time	147.2 (1)	94.4 (1.6)	69.2 (2.1)	57.2 (2.6)	51.2 (2.9)	48.8 (3.3)
k it.	5	5	5	5	5	5
m_k it.	2	2	2	2	2	2
i_k it.	30	30	30	30	30	30
$\frac{\|F(x_k)\|}{\|F(x_0)\|}$	$O(10^{-6})$	$O(10^{-6})$	$O(10^{-6})$	$O(10^{-6})$	$O(10^{-6})$	$O(10^{-6})$
LSQR	142.6	90.3	65.2	52.7	47.3	44.4
	97%	96%	94%	92%	92%	91%
Blas 1	66.6	49.2	41.4	37.3	36.0	35.2
	47%	54%	63%	71%	76%	79%
Blas 2	68.9	34.4	17.0	8.6	4.4	2.3
	48%	38%	26%	16%	9%	5%
Speedup (Blas1)	1	1.4	1.6	1.8	1.9	1.9
Speedup (Blas2)	1	2.1	4.1	8.2	16.2	31.4

Table 4: Time (in seconds) obtained with $p = 1, 2, 4, 8$, 16 and 32 nodes, for solving the *tridiagonal* problem with $N = 131072$ ($= 2^{17}$) on Cray T3E with the *Inexact Quasi Newton-Cimmino method,* with $k_{\max} = 20$, $m_k = 2$, $i_k = 30$ and $\varepsilon_1 = 10^{-6}, \varepsilon_2 = 10^{-5}, \varepsilon_3 = 10^{-12}$. Speedup values are given in parentheses. Time (and percentages) spent due to LSQR routine and the Blas routines are also given (their percentages are referred to LSQR time).

If we turn now to the inexact Newton tridiagonal problem we find that if we fix $\varepsilon_1 = O(10^{-12})$, the number k of outer iterations occurring to fulfill the exit test (5) increased (of one iteration) on passing from $p = 1$ node to $p > 1$ nodes. Therefore, for a correct speedup comparison and to estimate the degree of parallelism of the method, we fixed $\varepsilon_1 = O(10^{-6})$ obtaining an equal number of outer iterations k for every p.
From Tables 3 and 4 we can see that $k = 4$ and $k = 5$ outer iterations occurred to achieve the exit test. In both the tables $m_k = 2$ inner iterations occurred to achieve the inner exit test with $\varepsilon_2 = O(10^{-5})$ and $i_k = 30$ LSQR cycles for every p (with $\varepsilon_3 = O(10^{-9}) \div O(10^{-12})$).
From the two Tables we can also see that the speedups obtained with the Inexact Quasi-Newton method are roughly the same as in the Inexact Newton method.

2. Results for the Poisson problem
For comparison with the linear test problem, we restricted ourselves to test a problem of size $N = 4096$, relative to a grid of $n = l \times l$ nodes, $l = 64$. The test parameters were $\varepsilon_1 = 10^{-3}$, $\varepsilon_2 = 10^{-4}$, $\varepsilon_3 = 10^{-12}$, starting from an initial guess $x_0 = (-1, \ldots, -1)^T$. We see that only $k = 2$ outer iterations occurred with both

N=4096	$p=1$	$p=2$	$p=4$	$p=8$	$p=16$	$p=32$
Time	27.4	74.8 (1)	42.6 (1.8)	23.7 (3.2)	17.5 (4.3)	12.6 (5.9)
m it.	1	28	45	64	90	127
i it.	2258	384	211	118	73	38
$m \times i$	2258	10740	9481	7542	6569	4843
$\frac{\lVert r_m \rVert}{\lVert r_0 \rVert}$	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}
LSQR	27.3	74.6	42.3	23.3	16.1	11.6
	99.9%	99.8%	99.7%	99.2%	96.3%	96.1%
Blas 1	9.8	32.4	23.4	15.3	11.5	8.9
	36.3%	44.0%	55.2%	64.5%	72.5%	77.1%
Blas 2	16.3	37.6	14.8	4.9	2.0	0.7
	60.1%	49.9%	35.3%	22.5%	11.6%	6.0%
Speedup (Blas1)		1	1.4	2.1	2.8	3.6
Speedup (Blas2)		1	2.5	7.6	19.2	54.4

Table 5: Time (in seconds) obtained with $p = 1, 2, 4, 8$, 16 and 32 nodes, for solving the *linear nonsymmetric* problem of size $N = 4096$ with partitioning into p blocks of rows of the same dimension N/p, with the *row block Cimmino method.* Speedups values (with respect to T_2-time) are reported in parentheses. Time (in percentages) spent due to LSQR routine and the Blas routines are also given (their percentages are referred to LSQR time). Moreover, the number of outer m, the inner i iterations performed at each step and the overall number of LSQR iterations are shown, with $\varepsilon_2 = 10^{-3}, \varepsilon_3 = 10^{-12}$.

the methods to achieve the exit test. Therefore the two methods took the same time and an equal number of iterations. In fact, the inexact Quasi-Newton scheme should be tested with most difficult problems with respect to the evaluation of the Jacobian matrix, in order to show computational advantages with respect to the inexact Newton approach.
The results are summarized in Table 6. From the Table we shortly see that the remarks on the parallel performance already made for the linear problem are still valid.

References

[1] Arioli, M., Duff, I., Noailles, J., and Ruiz,D., *A block projection method for sparse matrices,* Siam J. Sci. Stat. Comput. **13 (1)** (1992) 47-70.

[2] Bramley R., and Sameh, A., *Row projections methods for large nonsymmetric linear systems,* Siam J. Sci. Stat. Comput. **13 (1)** (1992), 168-193

N=4096	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$
Time	100.8	2120.7 (1)	1521.8 (1.4)	702.1 (3.0)	311.8 (6.8)	159.8 (13.3)
k it.	2	2	2	2	2	2
m_k it.	1	89	212	303	391	525
i_k it.	4276	1715	813	372	161	68
$m_k \times i_k$	4276	165400	176960	113920	62841	35428
$\frac{\|F(x_k)\|}{\|F(x_0)\|}$	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}
LSQR	100.7	2116.5	1517.2	690.2	301.2	150.4
	99.9%	99.8%	99.7%	98.3%	96.6%	94.1%
Blas 1	35.2	908.0	831.4	450.0	217.2	114.3
1	35.0%	42.9%	54.8%	65.2%	72.1%	76.0%
Blas 2	61.8	1081.5	541.7	149.1	36.5	9.6
	61.4%	51.1%	35.7%	21.6%	12.1%	6.4%
Speedup (Blas1)		1	1.1	2.0	4.1	8.1
Speedup (Blas2)		1	2.0	7.2	29.6	112.7

Table 6: Time (in seconds) obtained with $p = 1$, 2, 4, 8 , 16 and 32 nodes, for solving the *Poisson* problem with $N = 4096$ with partitioning into p blocks of rows of the same size N/p, with $k = 2$, for each p, with the *Inexact Newton and Quasi Newton-Cimmino methods.* Speedup values (with respect to T_2-time) are given in parentheses. Time (in percentages) spent due to LSQR routine and the Blas routines are also given (their percentages are referred to LSQR time). Moreover, the number of outer m_k, the inner i_k iterations performed at each step and the overall number of LSQR iterations are shown, with $\varepsilon_1 = 10^{-3}, \varepsilon_2 = 10^{-4}, \varepsilon_3 = 10^{-12}$.

[3] Dembo, R.S. and Steihaug, T., *Truncated Newton algorithms for large-scale unconstrained optimization,* Series # 48, Yale University, New Haven, CT, September 1980.

[4] Dembo, R.S., Eisenstat, S.C., and Steihaug, T., *Inexact Newton methods,* SIAM. J. on Numeric. Anal. **19** (1992) 400-408.

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

[6] Kamath, C. and Sameh, A., *A projection method for solving nonsymmetric linear systems on multiprocessors,* Parallel Computing **9** (1988/89), 291-312.

[7] Matstoms, P, *Parallel sparse QR factorization on shared memory architectures,* Computing, (1994).

[8] Paige, G.G. and Saunders, M.A., *LSQR: An algorithm for sparse linear equations and sparse least squares,* ACM Trans. Math. Software **8** (1982), 43-71.

[9] Eisenstat, S.C., and Walker, H. F., *Choosing the forcing terms in an inexact Newton method,* SIAM. J. Sci Comput. **17-1** (1996), 16-32.

[10] Zilli, G., *Parallel implementation of a row-projection method for solving sparse linear systems,* Supercomputer **53, X-1** (1993), 33-43.

[11] Zilli, G., *Parallel method for sparse non-symmetric linear and non-linear systems of equations on a transputer network,* Supercomputer **6, XII-4** (1996), 4-15.

[12] Zilli, G. and Bergamaschi, L., *Truncated block Newton and Quasi Newton methods for sparse systems of nonlinear equations. Experiments on parallel platforms*, Recent advances in parallel virtual machine and parallel message passing interface, (M. Bubak, J. Dongarra, J. Wasniewski eds.), Lectures Notes in Computer Sciences, 1332, Springer, (1997), 390-397.

[13] Bergamaschi L., Moret, I. and Zilli G., *Inexact Block Quasi-Newton methods for sparse nonlinear systems of equations,* submitted

GIOVANNI ZILLI
Dipartimento di Metodi e Modelli Matematici per le Scienze Applicate, University of Padova
Via Belzoni 7 - 35131 PADOVA (Italy)

LUCA BERGAMASCHI
Dipartimento di Metodi e Modelli Matematici per le Scienze Applicate, University of Padova
Via Belzoni 7 - 35131 PADOVA (Italy)