The Granularity of Parallel Homotopy Algorithms for Polynomial Systems of Equations

D.C.S. Allison, S. Harimoto, and L.T. Watson

TR 88-4

7-1-1988

The Granularity of Parallel Homotopy Algorithms for Polynomial Systems of Equations

D. C. S. Allison, S. Harimoto†, and L. T. Watson†
Department of Computer Science
Virginia Polytechnic Institute & State University
Blacksburg, VA 24061

Abstract. Polynomial systems consist of n polynomial functions in n variables, with real or complex coefficients. Finding zeros of such systems is challenging because there may be a large number of solutions, and Newton-type methods can rarely be guaranteed to find the complete set of solutions. There are homotopy algorithms for polynomial systems of equations that are globally convergent from an arbitrary starting point with probability one, are guaranteed to find all the solutions, and are robust, accurate, and reasonably efficient. There is inherent parallelism at several levels in these algorithms. Several parallel homotopy algorithms with different granularities are studied on several different parallel machines, using actual industrial problems from chemical engineering and solid modeling.

1. Introduction. Solving nonlinear systems of equations is a central problem in numerical analysis, with enormous significance for science and engineering. A very special case, namely small polynomial systems of equations, occurs frequently enough in solid modeling, robotics, computer vision, chemical equilibrium computations, chemical process design, mechanical engineering, and other areas to justify special algorithms. To put polynomial systems in perspective and for the purpose of this discussion, there are three classes of nonlinear systems of equations: (1) large systems with sparse Jacobian matrices, (2) small transcendental (nonpolynomial) systems with dense Jacobian matrices, and (3) small polynomial systems with dense Jacobian matrices. Sparsity for small problems is not significant, and large systems with dense Jacobian matrices are intractable, so these two classes are not counted.

Large sparse nonlinear systems of equations, such as equilibrium equations in structural mechanics, have two characterizing aspects: highly nonlinear and recursive scalar computations, and large matrix, vector operations. There is a great amount of parallelism in both aspects, but the nature of the parallelism is very different (or so it seems). Small dense transcendental systems of equations pose a major challenge, since they involve recursive, scalar intensive computation with a small amount of linear algebra. Finally, polynomial systems are unique in that they have many solutions, of which several may be physically meaningful, and there exist homotopy algorithms guaranteed to find all these meaningful solutions. The very special nature of polynomial systems and the power of homotopy algorithms are often not fully appreciated, perhaps because globally convergent probability-one homotopy methods are not widely known.

These globally convergent homotopy algorithms for polynomial systems have inherent parallelism at several levels. The purpose of the present paper is to study different granularities of parallel homotopy algorithms for polynomial systems, corresponding to different decomposition and communication strategies.

† The work of these authors was supported in part by AFOSR Grant 85-0250.

Much work has been done on solving linear systems of equations on parallel computers, mostly on vector machines [4], [5], [7], [8], [10]–[12], [14]–[16], [18], [23], [25]. Some work has been done on nonlinear equations and Newton's method [28], [31], [36], [37], and on finding the roots of a single polynomial equation [9], [27]. Parallel algorithms for polynomial systems were proposed by Morgan and Watson [22], but have not been studied much, nor have parallel homotopy algorithms for nonlinear systems of equations.

Section 2 summarizes the mathematics behind the homotopy algorithm, Section 3 discusses the special case of polynomial systems in some detail, and computational results on several parallel machines are presented and discussed in Section 4.

2. Homotopy algorithm.

Let E^n denote n-dimensional real Euclidean space. The fundamental mathematical result behind the homotopy algorithm for solving the nonlinear system of equations

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

where $F: E^n \to E^n$ is a C^2 (twice continuously differentiable) function, is as follows:

Proposition 1 ([6], [33]). Let $\rho: E^m \times [0,1) \times E^n \to E^n$ be a C^2 map and define $\rho_a(\lambda, x) = \rho(a, \lambda, x)$. Suppose that

1. the $n \times (m+n+1)$ Jacobian matrix $D\rho$ has full rank on $\rho^{-1}(0)$, the set of zeros of ρ;
2. $\rho_a(0,x) = 0$ has a unique solution W of E^n (depending on the homotopy parameter vector a of E^m);
3. $\rho_a(1,x) = F(x)$;
4. the set of zeros of $\rho_a(\lambda, x)$ is bounded.

Then for almost all a of E^m there is a zero curve γ of $\rho_a(\lambda, x)$ along which the Jacobian matrix $D\rho_a(\lambda, x)$ has full rank, emanating from $(0, W)$ and reaching a zero $\bar{x}$ of $F(x)$ at $\lambda = 1$. Furthermore, γ has finite arc length if $DF(\bar{x})$ is nonsingular.

The general idea of the algorithm is to follow the zero curve γ of ρ_a from $(0, W)$ until a zero $\bar{x}$ of $F(x)$ is reached at $\lambda = 1$. Although the homotopy algorithm for solving the nonlinear system of equations is conceptually simple, it is nontrivial to develop a viable numerical algorithm for tracking the curve. A typical form for the homotopy map is

$$\rho_W(\lambda, x) = \lambda F(x) + (1-\lambda)(x - W) = 0, \tag{2}$$

which has the same form as a standard continuation or embedding mapping. However, two crucial differences exist. In standard continuation methods, the embedding parameter λ increases monotonically from 0 to 1 as the trivial problem $x - W = 0$ is continuously deformed to the problem $F(x) = 0$. Homotopy algorithms permit λ to both increase and decrease along γ with no adverse effect. The second difference is that in homotopy algorithms there are no "singular points" which afflict standard continuation methods. This is guaranteed by the way in which the zero curve γ of ρ_a is followed and the fact that $D\rho_a$ has full rank along γ .

The zero curve γ of the homotopy map $\rho_a(\lambda, x)$ can be tracked by many different techniques. The present work used HOMPACK [34], a software package for solving systems of nonlinear equations based on the homotopy method. HOMPACK provides three approaches for tracking γ : 1) an ODE-based algorithm with some special refinements for the homotopy context; 2) a predictor-corrector algorithm whose corrector follows the flow normal to the Davidenko flow (a "normal flow"

algorithm); and 3) a version of Rheinboldt's linear predictor, quasi-Newton corrector algorithm (an "augmented Jacobian matrix" method). Since the "normal flow" algorithm is the technique used by HOMPACK to solve polynomial systems of equations, the other two algorithms will not be discussed in this paper.

The normal flow algorithm has three phases: prediction, correction, and step size estimation. In the prediction phase, the next point on the zero curve is predicted. Starting from the predicted point, the correction phase then iterates until a point on the zero curve is reached. An "optimal" step size is then estimated for the prediction of the next point on the curve.

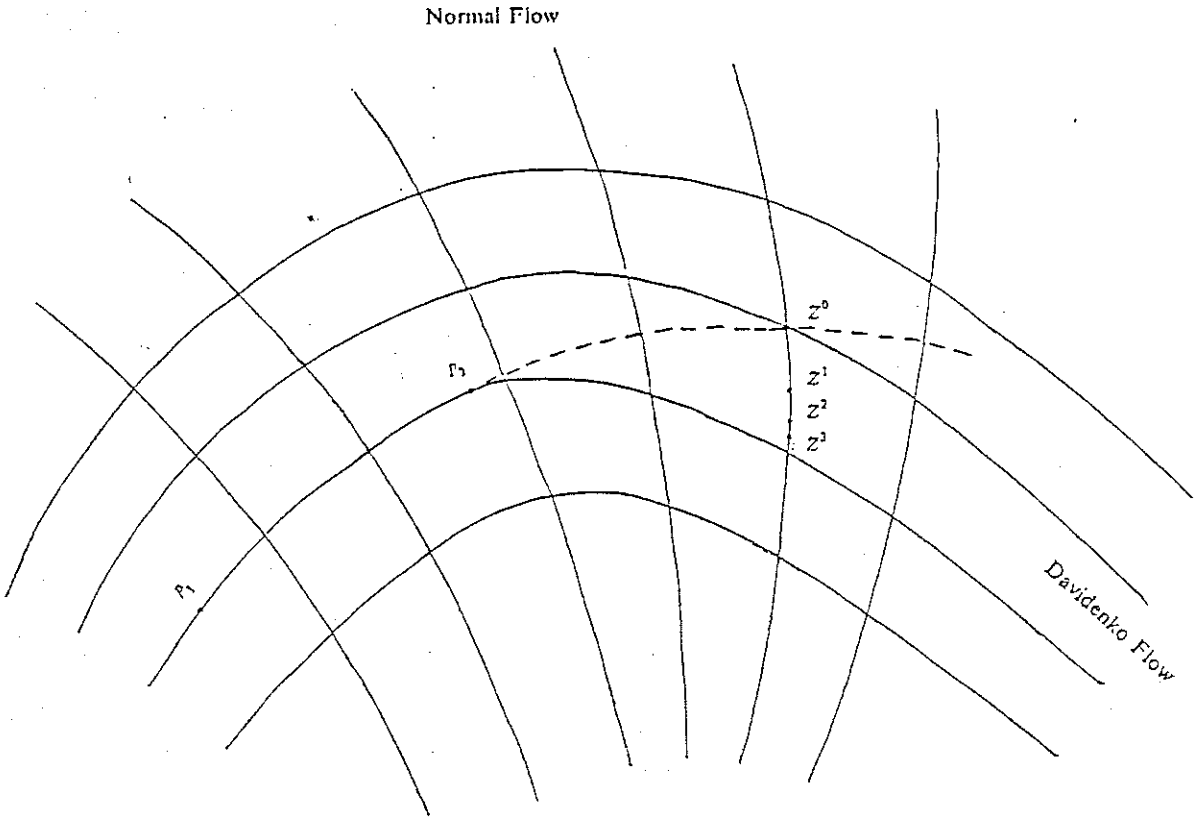

Figure 1. The prediction and the correction phases of the normal flow algorithm.

The normal flow algorithm is so called because the iterates of the correction phase converge from the predicted point back to the zero curve along the flow normal to the Davidenko flow (see Figure 1). The Davidenko flow is the family of zero curves formed from varying the parameter vector a of the homotopy map ρ_a.

The zero curve γ of $\rho_a(\lambda, x)$ is C^1 and can be parametrized by the arc length s. Thus $\lambda = \lambda(s)$ and $x = x(s)$ with initial conditions $\lambda(0) = 0$ and $x(0) = W$. When $\lambda(\bar{s}) = 1$, the corresponding $x(\bar{s}) = \bar{x}$ is a zero of $F(x)$.

Given that the following are available from previous calculations : two previous points on the curve, $P(s_1) = (\lambda(s_1), x(s_1))$ and $P(s_2) = (\lambda(s_2), x(s_2))$, their corresponding tangent vectors, $P'(s_1) = (d\lambda/ds(s_1), dx/ds(s_1))$ and $P'(s_2) = (d\lambda/ds(s_2), dx/ds(s_2))$, and h, an estimate of the next "optimal" step size (in arc length) to take along γ , the next point on the zero curve can be estimated by

$$Z^{(0)} = H(s_2 + h), \tag{3}$$

where $H(s)$ is the Hermite cubic polynomial which interpolates $P(s)$ at s_1 and s_2. Thus, $H(s_1) = P(s_1)$, $H'(s_1) = P'(s_1)$, $H(s_2) = P(s_2)$, and $H'(s_2) = P'(s_2)$.

Since

$$\rho_a(\lambda(s), x(s)) = 0 \tag{4}$$

on the zero curve γ ,

$$\frac{d}{ds}\left[\rho_a(\lambda(s), x(s))\right] = D\rho_a(\lambda(s), x(s))(d\lambda/ds, dx/ds)^T = 0, \tag{5}$$

where $(d\lambda/ds, dx/ds)$ is the tangent vector to the curve. Thus, the tangent vector can be calculated by finding the kernel of the Jacobian matrix $D\rho_a(\lambda(s), x(s))$, which has rank n by Proposition 1. Once the kernel is found, the derivative $(d\lambda/ds, dx/ds)$ at a given point on the zero curve can be uniquely determined by

$$||(d\lambda/ds, dx/ds)||_2 = 1 \tag{6}$$

and continuity of the tangent vector.

Starting at the predicted point $Z^{(0)}$, the iteration in the correction phase is

$$Z^{(k+1)} = Z^{(k)} - \left[D\rho_a\left(Z^{(k)}\right)\right]^{\dagger} \rho_a\left(Z^{(k)}\right), \qquad k = 0, 1, \ldots \tag{7}$$

where $\left[D\rho_a(Z^{(k)})\right]^{\dagger}$ is the Moore-Penrose pseudoinverse of $D\rho_a$. Reordering the equation, the corrector step $\Delta Z = Z^{(k+1)} - Z^{(k)}$ is the unique minimum norm solution of

$$[D\rho_a(Z^{(k)})]\Delta Z = -\rho_a(Z^{(k)}). \tag{8}$$

Fortunately ΔZ can be calculated at the same time as the kernel of $[D\rho_a]$ with just a little more effort. Normally for dense problems the kernel of $[D\rho_a]$ is found by computing the QR factorization of $[D\rho_a]$, followed by back substitution. By applying this QR factorization to $-\rho_a$ and using back substitution again, a particular solution v to (8) can be found. Let u be any non-zero vector in the kernel of $[D\rho_a]$. Then the minimum norm solution is

$$\Delta Z = v - \frac{v^t u}{u^t u} u. \tag{9}$$

Since the QR factorizations of $[D\rho_a]$ are computationally very expensive, the number of iterations required for convergence of (7) should be kept small (say <= 4) by adjusting the step size. An alternative is to use the QR factorization of $D\rho_a$ at the first predicted point $Z^{(0)}$ for several iterations. However, this results in linear convergence, which is not cost effective when compared to the asymptotically quadratic convergence of (7).

Note that the kernel of $[D\rho_a]$ is needed for the tangent vector used in the Hermite cubic interpolation at the beginning of the next step. When the iteration converges, the final iterate $Z^{(k+1)}$ is accepted as the next point on γ . Rather than calculating the tangent vector at the new point $Z^{(k+1)}$ on γ , a Jacobian matrix evaluation and a QR factorization can be saved by using the tangent vector calculated at $Z^{(k)}$. This substitution should not seriously affect the calculation of the next point on the curve since this tangent is used only in the prediction of the next point, and the tangent vector at $Z^{(k)}$ is a good estimate of the tangent vector at $Z^{(k+1)}$.

The estimation of an "optimal" step size h is an attempt to balance the number of iterations in the correction phase of the algorithm with the number of steps necessary to reach the "end" point on the zero curve where $\lambda(\bar{s}) = 1$. Increasing the step size decreases the number of steps necessary to reach the "end" of the curve. However, taking too large a step size would result in a substantial increase in the number of iterations necessary to correct the predicted point.

In order to estimate the next "optimal" step size $\bar{h}$, a few parameters for "measuring" the performance of the step size used in the last prediction must be defined. Let

$$L = \frac{\left\| Z^{(2)} - Z^{(1)} \right\|}{\left\| Z^{(1)} - Z^{(0)} \right\|}, \tag{10}$$

be the contraction factor,

$$R = \frac{\left\| \rho_a\left(Z^{(1)}\right) \right\|}{\left\| \rho_a\left(Z^{(0)}\right) \right\|}, \tag{11}$$

be the residual factor, and

$$D = \frac{\left\| Z^{(1)} - Z^{*} \right\|}{\left\| Z^{(0)} - Z^{*} \right\|}, \tag{12}$$

be the distance factor (where $Z^{*} = \lim_{k \to \infty} Z^{(k)}$ is the point on the zero curve γ to which the iteration (7) converges). Assume that the ideal values of L, R, and D are chosen to be $\bar{L}$, $\bar{R}$, and $\bar{D}$, respectively. When a step size produces a "bad" prediction, at least one of the factors would have a large value (close to 1.0) indicating slow convergence (refer to Figure 1). The objective then is to improve the rate of convergence by reducing the step size by a fraction related to the ratio of the "worst indicator" factor's ideal value to that indicator factor's actual observed value. Stated mathematically, the goal is to achieve

$$\frac{\bar{L}}{L} \approx \frac{\bar{R}}{R} \approx \frac{\bar{D}}{D} \approx \frac{\bar{h}^q}{h^q} \tag{13}$$

for a given q, where h is the distance from Z^{*} to the previous point $P(s_2)$ found on γ. Let

$$\hat{h} := \left(\min\{\bar{L}/L, \bar{R}/R, \bar{D}/D\}\right)^{1/q} h \tag{14}$$

be the worst case choice. To prevent chattering and unreasonable values, constants $h_{\min}$ (minimum allowed step size), $h_{\max}$ (maximum allowed step size), $B_{\min}$ (contraction factor), and $B_{\max}$ (expansion factor) are chosen to place bounds on $\bar{h}$ by choosing $\bar{h}$ in the following way:

$$\bar{h} := \min\left\{\max\{h_{\min}, B_{\min} h, \hat{h}\}, B_{\max} h, h_{\max}\right\}. \tag{15}$$

The choice of $\bar{h}$ can be refined further. If (7) converged in one iteration for the previous correction phase, then $\bar{h}$ should certainly not be any smaller than h. Otherwise, $\bar{h}$ would not be the "optimal" choice. Thus, set

$$\bar{h} := \max\{h, \bar{h}\} \tag{16}$$

if (7) converged in one iteration in the previous step.

If (7) has not converged after K iterations, the value of $\bar{h}$ is saved in h_{old} (the last step size which has been known to cause (7) to fail to converge after K iterations), $\bar{h}$ is halved and a

new prediction is computed. The value of $\bar{h}$ continues to be halved until the step size taken for predicting $Z^{(0)}$ is small enough such that (7) converges within K iterations.

If in the previous step, the convergence of (7) had failed, the new $\bar{h}$ should not be greater than the value h_{old} known to produce failure. Hence in this case

$$\bar{h} := \min\{h_{\text{old}}, \bar{h}\}. \tag{17}$$

Finally, if (7) required the maximum K iterations, the step size should not increase, so in this case set

$$\bar{h} := \min\{h, \bar{h}\}. \tag{18}$$

The logic in (16–18), when invoked, has a stabilizing effect on the algorithm.

3. Polynomial systems.

Section 2 described a homotopy algorithm for finding a single solution to a general nonlinear system of equations $F(x) = 0$. Proposition 1 provided the theoretical guarantee of convergence. The rich structure and multiple solutions of polynomial systems dictate that the general theory in Section 2 must be sharpened. This section develops a globally convergent (with probability one) homotopy algorithm that finds *all* solutions to a polynomial system, and provides the theoretical justification for that algorithm.

Suppose that the components of the nonlinear function $F(x)$ have the form

$$F_i(x) = \sum_{k=1}^{n_i} a_{ik} \prod_{j=1}^{n} x_j^{d_{ijk}}, \quad i = 1, \ldots, n. \tag{19}$$

The ith component $F_i(x)$ has n_i terms, the a_{ik} are the (real) coefficients, and the degrees d_{ijk} are nonnegative integers. The total degree of F_i is

$$d_i = \max_k \sum_{j=1}^{n} d_{ijk}. \tag{20}$$

For technical reasons it is necessary to consider $F(x)$ as a map $F : C^n \to C^n$, where C^n is n-dimensional complex Euclidean space. A system of n polynomial equations in n unknowns, $F(x) = 0$, may have many solutions. It is possible to define a homotopy so that all geometrically isolated roots of (19) have at least one associated homotopy path. Generally, (19) will have roots at infinity, which forces some of the homotopy paths to diverge to infinity as λ approaches 1. However, (19) can be transformed into a new system which, under reasonable hypotheses, can be proven to have no roots at infinity and thus bounded homotopy paths. Because scaling can be critical to the success of the method, a general scaling algorithm [34] is applied to scale the coefficients and variables in (19) before anything else is done.

Since the homotopy map defined below is complex analytic, the homotopy parameter λ is monotonically increasing as a function of arc length [21]. The existence of an infinite number of solutions or an infinite number of solutions at infinity does not destabilize the method. Some paths will converge to the higher dimensional solution components, and these paths will behave the way paths converging to any singular solution behave. Practical applications usually seek a subset of the solutions, rather than all solutions [19], [20]. However, the sort of generic homotopy algorithm

considered here must find all solutions and cannot be limited without, in essence, changing it into a heuristic.

Define $G : C^n \to C^n$ by

$$G_j(x) = b_j {x_j}^{d_j} - a_j, \qquad j = 1, \ldots, n, \tag{21}$$

where a_j and b_j are nonzero complex numbers and d_j is the (total) degree of $F_j(x)$, for $j = 1, \ldots, n$. Define the homotopy map

$$\rho_c(\lambda, x) = (1 - \lambda)\, G(x) + \lambda\, F(x), \tag{22}$$

where $c = (a, b)$, $a = (a_1, \ldots, a_n) \in C^n$ and $b = (b_1, \ldots, b_n) \in C^n$. Let $d = d_1 \cdots d_n$ be the *total degree* of the system. The fundamental homotopy result, proved and discussed at length in [19]–[21], is:

Theorem. For almost all choices of a and b in C^n, $\rho_c^{-1}(0)$ consists of d smooth paths emanating from $\{0\} \times C^n$, which either diverge to infinity as λ approaches 1 or converge to solutions to $F(x) = 0$ as λ approaches 1. Each geomètrically isolated solution of $F(x) = 0$ has a path converging to it.

A number of distinct homotopies have been proposed for solving polynomial systems. The homotopy map in (22) is from [20]. As with all such homotopies, there will be paths diverging to infinity if $F(x) = 0$ has solutions at infinity. These divergent paths are (at least) a nuisance, since they require arbitrary stopping criteria. Solutions at infinity can be avoided via the following projective transformation.

Define $F'(y)$ to be the homogenization of $F(x)$:

$$F'_j(y) = {y_{n+1}}^{d_j}\, F_j(y_1/y_{n+1}, \ldots, y_n/y_{n+1}), \qquad j = 1, \ldots, n. \tag{23}$$

Note that, if $F'(y^0) = 0$, then $F'(\alpha y^0) = 0$ for any complex scalar α. Therefore, "solutions" of $F'(y) = 0$ are (complex) lines through the origin in C^{n+1}. The set of all lines through the origin in C^{n+1} is called complex projective n-space, denoted CP^n, and is a smooth compact (complex) n-dimensional manifold. The solutions of $F'(y) = 0$ in CP^n are identified with the (finite) solutions and solutions at infinity of $F(x) = 0$ as follows. If $L \in CP^n$ is a solution to $F'(y) = 0$ with $y = (y_1, y_2, \ldots, y_{n+1}) \in L$ and $y_{n+1} \neq 0$, then $x = (y_1/y_{n+1}, y_2/y_{n+1}, \ldots, y_n/y_{n+1}) \in C^n$ is a solution to $F(x) = 0$. On the other hand, if $x \in C^n$ is a solution to $F(x) = 0$, then the line through $y = (x, 1)$ is a solution to $F'(y) = 0$ with $y_{n+1} = 1 \neq 0$. The most mathematically satisfying definition of *solutions to* $F(x) = 0$ *at infinity* is simply *solutions to* $F'(y) = 0$ *(in* CP^n*) generated by* y *with* $y_{n+1} = 0$.

A basic result on the structure of the solution set of a polynomial system is the following classical theorem of Bezout [21]:

Theorem. There are no more than d isolated solutions to $F'(y) = 0$ in CP^n. If $F'(y) = 0$ has only a finite number of solutions in CP^n, it has exactly d solutions, counting multiplicities.

Recall that a solution is *isolated* if there is a neighborhood containing that solution and no other solution. The multiplicity of an isolated solution is defined to be the number of solutions that appear in the isolating neighborhood under an arbitrarily small random perturbation of the system coefficients. If the solution is nonsingular (i.e., the system Jacobian matrix is nonsingular at the solution), then it has multiplicity one. Otherwise it has multiplicity greater than one.

Define a linear function

$$u(y_1, \ldots, y_{n+1}) = \xi_1 y_1 + \xi_2 y_2 + \cdots + \xi_{n+1} y_{n+1} \tag{24}$$

where $\xi_1, \ldots, \xi_{n+1}$ are nonzero complex numbers, and define $F'' : C^{n+1} \to C^{n+1}$ by

$$\begin{aligned} F''_j(y) &= F'_j(y), \qquad j = 1, \ldots, n, \\ F''_{n+1}(y) &= u(y) - 1. \end{aligned} \tag{25}$$

So $F''(y) = 0$ is a system of $n+1$ equations in $n+1$ unknowns, referred to as *the projective transformation of* $F(x) = 0$. Since $u(y)$ is linear, it is easy in practice to replace $F''(y) = 0$ by an equivalent system of n equations in n unknowns. The significance of $F''(y)$ is given by

Theorem[19]. If $F'(y) = 0$ has only a finite number of solutions in CP^n, then $F''(y) = 0$ has exactly d solutions (counting multiplicities) in C^{n+1} and no solutions at infinity, for almost all $\xi \in C^{n+1}$.

Under the hypothesis of the theorem, all the solutions of $F'(y) = 0$ can be obtained as lines through the solutions to $F''(y) = 0$. Thus all the solutions to $F(x) = 0$ can be obtained easily from the solutions to $F''(y) = 0$, which lie on bounded homotopy paths (since $F''(y) = 0$ has no solutions at infinity).

The projective transformation functions essentially as a scaling transformation. Its effect is to shorten arc lengths and bring solutions closer to the unit sphere. The coefficient and variable scaling is different, in that it directly addresses extreme values in the system coefficients. The two scaling schemes work well together; see [21] and [34].

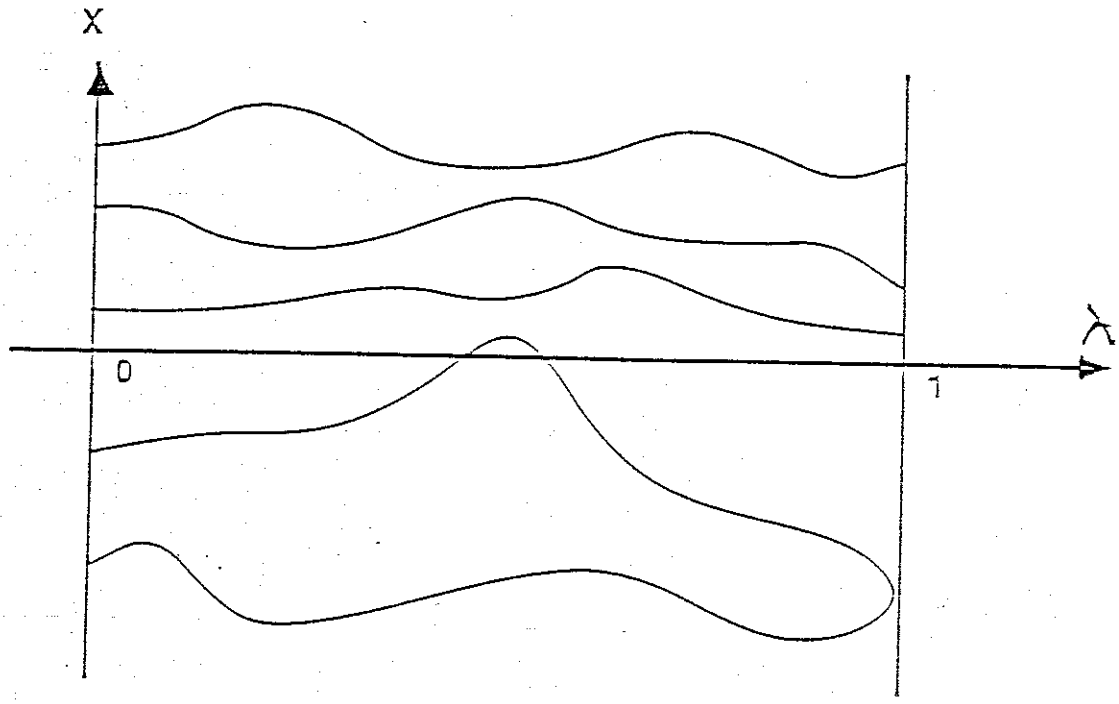

Figure 2. The set $\rho_a^{-1}(0)$ for a transformed polynomial system.

The import of the above theory is that the nature of the zero curves of the projective transformation $F''(y)$ of $F(x)$ is as shown in Figure 2. There are exactly d (the total degree of F) zero curves, which are monotone in λ and have finite arc length. The homotopy algorithm is to track these d curves, which contain all isolated (transformed) zeros of F.

4. Computational results.

There are two extreme approaches for parallelizing the homotopy algorithm. For the coarsest form of parallelism, each individual processor tracks as many solutions (paths) as possible until all the solutions for the system of polynomial equations are found. In the other extreme, where the granularity is the finest, the primary task of tracking the solutions is delegated to one of the processors and only during polynomial system evaluations, Jacobian matrix evaluations, and other numerical calculations is the work distributed among the processors.

In the first case, the division of work is at the highest level. Initially, the parameters defining the system of polynomial equations ($F(x) = 0$) are distributed to the processors. Each processor then works independently of the other processors. When a processor completes the tracking of a path, it takes the next path to be tracked. Since there is no knowledge about the paths, they are assigned on a first-come-first-serve basis. Thus, if a "bad" path exists (one which requires a large number of function evaluations with respect to the remaining paths to be tracked), the load would not be distributed evenly among the processors. The processor with the "bad" path would cause the other processors to remain idle after all the remaining paths have been tracked.

The second approach is an attempt to balance the load more evenly, hopefully, resulting in an overall speedup over the coarser grained algorithm. In this paper, the fine-grained parallel homotopy algorithm distributes the work of evaluating the system of polynomial equations and its partial derivatives to N processors, where N is the number of equations or the maximum number of processors, whichever is smaller.

Two parallel versions, a coarse-grained version and a fine-grained version, of the homotopy algorithm for polynomial systems were developed from the parallelization of HOMPACK. They were executed on several parallel machines: Balance 21000, Elxsi 6400, Alliant FX/8, and the Intel iPSC-32. The execution times are shown in Table 1 along with those for the execution of the serial algorithm. Superscript 1 denotes the coarse-grained version, superscript 2 the fine-grained version, and no superscript the serial version. The efficiencies are listed in Table 2. The problem number refers to an internal numbering scheme used at General Motors Research Laboratories. For example, problem 403 is

$$F_j(x) = a_{j1}x_1^2 + a_{j2}x_2^2 + a_{j3}x_1x_2 + a_{j4}x_1 + a_{j5}x_2 + a_{j6} = 0, \qquad \text{for } j = 1, 2,$$

where

$$\begin{array}{llll} a_{11} = -.00098 & a_{14} = -235 & a_{21} = -.01 & a_{24} = .00987 \\ a_{12} = 978000 & a_{15} = 88900 & a_{22} = -.984 & a_{25} = -.124 \\ a_{13} = -9.8 & a_{16} = -1.0 & a_{23} = -29.7 & a_{26} = -.25 \end{array}$$

The exact solutions (to four significant figures) are

$$\begin{aligned} (x_1, x_2) = &(.09089, -.09115), \\ &(2342, -.7883), \\ &(.01615 + 1.685i, .0002680 + .004428i), \\ &(.01615 - 1.685i, .0002680 - .004428i). \end{aligned}$$

These problems are all real engineering problems that have arisen at General Motors and elsewhere. The number in parenthesis beside the problem number is the number of equations in the polynomial system. It is also the maximum number of processors which can be utilized by the fine-grained

algorithm. The total degree of a problem refers to the number of solutions in the system of polynomial equations. Thus, if a machine has infinitely many processors, the maximum number of processors which can be utilized by the coarse-grained algorithm is determined by the total degree of the problem.

This paper is primarily concerned with the results obtained on bus-oriented, shared-memory parallel machines. The primary difference between implementations of parallel homotopy algorithms on shared-memory machines and implementations on distributed-memory machines is that, in distributed-memory machines, the master processor must communicate the tasks to the slave processors individually and wait for the results through some form of communication medium. After a set of results is received from one of the slave processors, the master processor assigns the next task to that free processor. Both the master processor and the slave processors must handle the processing of the communication protocols. Results for the Intel iPSC-32 are included for comparison only – granularity issues for solving polynomial systems on distributed memory machines are the subject of future work.

In shared-memory machines, the task of communicating the problem and obtaining the results is much simpler. It is handled through shared-memory, which is accessed by all the processors. The primary processor sets up the problem in the shared-memory, initiates the processors, and handles portions of the problem like the other spawned processors. Since the coordination of the processors is done through mutual exclusion of certain critical shared memory locations, no master processor is needed.

The calculations for the coarse-grained efficiencies are based on the maximum number of processors used on each of the parallel machines. The number of processors used on Sequent's Balance 21000, Elxsi's 6400, and Alliant's FX/8 are 8, 10, and 8, respectively, except for problems 402 and 403, where the total number of paths to be tracked is only 4.

The efficiencies for the coarse-grained algorithm with various number of processors on Alliant's FX/8 are shown in Table 3. As one would expect, the efficiency improves as the number of processors decreases.

5. Conclusions.

It is deceiving to compare the efficiencies of the two parallel homotopy algorithms from Table 2. One might conclude that for some problems, the fine-grained algorithm is just as efficient as the coarse-grained algorithm. As shown by Table 3, the efficiencies improve as the number of processors decreases. When the efficiencies of the two algorithms are compared using an equal number of processors, the coarse-grained algorithm is found to be more efficient than the fine-grained algorithm. The reason is that on the average over 90 percent of the serial time can be parallelized with the coarse-grained algorithm. On the other hand the fine-grained algorithm can parallelize only 50 to 80 percent of the serial time. The overhead cost of using shared memory can also be substantial, as shown by problems 803 and 5001 on the Elxsi (*cf.* Table 2).

In terms of speedup, the coarse-grained algorithm outperforms the fine-grained algorithm. This is due to the fact that the coarse-grained algorithm can use as many processors as are available on the parallel machine, unless the number of solutions to be tracked is smaller than the number of processors available. In this case, the coarse-grained algorithm is limited by the number of

solutions. The fine-grained algorithm is always limited by the number of equations in the system of polynomial equations.

The problem with the coarse-grained algorithm is that when some solutions have long paths with respect to other solutions, the efficiency can be very low. This depends on the order in which the solutions are found and the total number of solutions with respect to the number of processors. As shown in Figure 3 and Table 3 pertaining to the Alliant FX/8, the efficiency of a problem can vary drastically. Figure 3 demonstrates that by changing the order in which the solutions (of problem 602) are found, the distribution of the work load can either be very unbalanced or very well balanced.

The results in Figure 3 are based on the assumption that the execution time for each solution is proportional to the number of function evaluations performed in obtaining the solution. This is a reasonable assumption based on what has been observed on the actual assignment of processors to the solutions of problem 602 and other problems in the tables. In most cases, the processor assigned to track the solution which requires the least number of function evaluations is assigned to track the next solution. It is possible only when the difference in the number of function evaluations required between two solutions is not significant that a processor with the "longer" path (requiring more function evaluations) be assigned to track the next solution first.

In Figure 3, curve 1 shows that the work load can be distributed quite evenly among the processors when the "long" paths are assigned first. The distribution of the work load in the actual assignment of the paths (for problem 602) is shown by curve 2. The worst distribution (curve 3) occurs when the "longest" path is assigned last while the rest of the paths are assigned in decreasing order in terms of the number of function evaluations. From these distributions, curve 1 has the "best" efficiency (0.99) and curve 3 has the "worst" efficiency (0.56), whereas the actual efficiency was 0.67.

The distribution of work in the fine-grained algorithm may not necessarily be balanced. Since the decomposition is by the components of the polynomial system, some processors may have to handle more difficult components than others.

As stated previously, the percentage of the serial execution time that can be parallelized on the average ranges from 50 to 80 percent. This is the percentage of the serial time that is spent on function evaluations. Thus, as the complexity of the polynomial equations increases, the amount of time spent in the evaluation of functions increases.

In general, the coarse-grained parallel homotopy algorithm is more efficient and permits a higher degree of parallelism than the fine-grained algorithm. However, since the number of function evaluations required for each path can not be predicted, the efficiency of the coarse-grained algorithm can be very low. A possible solution to this problem might be a combination of the two algorithms. The coarse-grained algorithm could be used until there are no remaining paths to be tracked. Then the fine-grained algorithm could be used to finish the tracking of the uncompleted paths.

6. References.

[1] E. Allgower and K. Georg, *Simplicial and continuation methods for approximating fixed points*, SIAM Rev., 22 (1980), pp. 28–85.

[2] S. C. Billups, *An augmented Jacobian matrix algorithm for tracking homotopy zero curves*, M.S. Thesis, Dept. of Computer Sci., VPI & SU, Blacksburg, VA, Sept., 1985.

[3] P. Businger and G. H. Golub, *Linear least squares solutions by Householder transformations*, Numer. Math., 7 (1965), pp. 269–276.

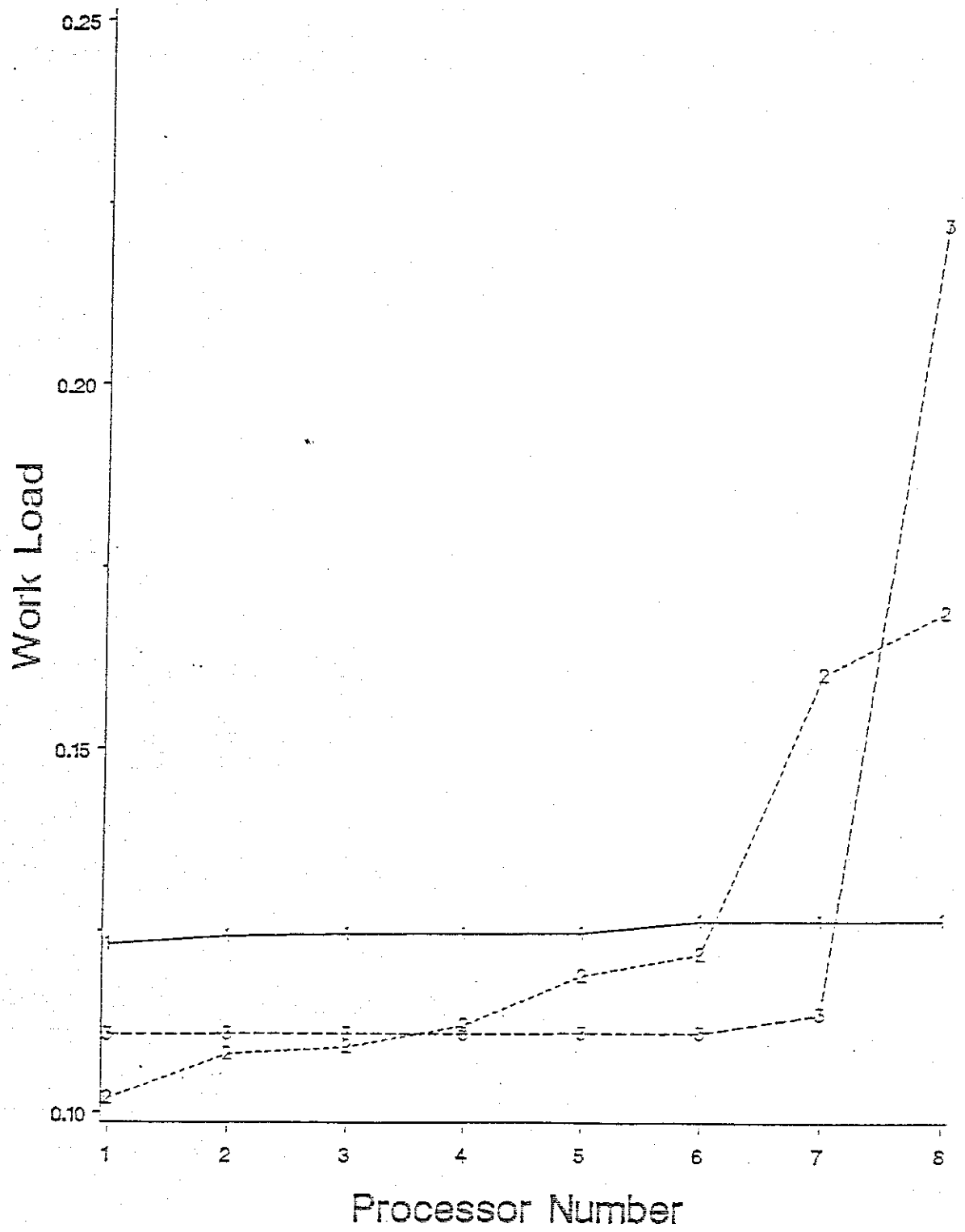

Figure 3. The distribution of work in coarse-grained parallelism for various orderings of paths for problem 602 on the Alliant FX/8.

[4] A. C. Chen and C. L. Wu, *Optimum solution to dense linear systems of equations*, Proc. 1984 Internat. Conf. on Parallel Processing, August 21–24, 1984, pp. 417–425.

[5] M. Y. Chern and T. Murata, *Fast algorithm for concurrent LU decomposition and matrix inversion*, Proc. Internat. Conf. on Parallel Processing, Computer Society Press, Los Alamitos, CA, 1983, pp. 79–86.

[6] S. N. Chow, J. Mallet-Paret, and J. A. Yorke, *Finding zeros of maps: Homotopy methods that are constructive with probability one*, Math. Comput., 32 (1978), pp. 887–899.

[7] E. Cloète and G. R. Joubert, *Direct methods for solving systems of linear equations on a parallel processor*, Proc. 8th South African Symp. on Numerical Mathematics, Durban, South Africa, July 19–21, 1982.

[8] M. Cosnard, Y. Robert, and D. Trystran, *Comparison of parallel diagonalization methods for solving dense linear systems*, Sessions of the French Acad. of Sci. on Math., Nov., 1985, p. 781ff.

[9] G. H. Ellis and L. T. Watson, *A parallel algorithm for simple roots of polynomials*, Comput. Math. Appl., 10 (1984), pp. 107–121.

[10] D. D. Gajski, A. H. Sameh, and J. A. Wisniewski, *Iterative algorithms for tridiagonal matrices on a WSI-multiprocessor*, Proc. Internat. Conf. Parallel Processing, Bellaire, MI, Aug. 24–27, pp. 82–89, 1982.

[11] W. Gentzsch and G. Schafer, *Solution of large linear systems on vector computers*, Parallel Computing 83, North Holland, Amsterdam, 1984, pp. 159–166.

[12] D. Heller, *A survey of parallel algorithms in numerical linear algebra*, SIAM Rev., 20 (1978), pp. 740–777.

[13] Intel Corporation, *iPSC Users' Manual*, 1985.

[14] Y. Kaneda and M. Kohata, *Highly parallel computing of linear equations on the matrix-broadcast-memory connected array processor system*, 10th IMACS World Congress, Vols. 1–5, pp. 320–322, 1982.

[15] J. S. Kowalik, *Parallel computation of linear recurrences and tridiagonal equations*, Proc. IEEE 1982 Internat. Conf. on Cybernetics and Society, 1982, pp. 580–584.

[16] J. S. Kowalik and S. P. Kumar, *An efficient parallel block conjugate gradient method for linear equations*, Proc. Internat. Conf. Parallel Processing, Bellaire, MI, Aug. 24–27, pp. 47–52, 1982.

[17] M. Kubicek, *Dependence of solutions of nonlinear systems on a parameter*, ACM Trans. Math. Software, 2 (1976), pp. 98–107.

[18] S. Lakshmivarahan and S. K. Dhall, *Parallel algorithms for solving certain classes of linear recurrences*, Foundations of Software Technology and Theoretical Computer Science, Lecture Notes in Computer Science, Vol. 206, Springer-Verlag, Berlin, 1985, pp. 457–477.

[19] A. P. Morgan, *A transformation to avoid solutions at infinity for polynomial systems*, Appl. Math. Comput., 18 (1986), pp. 77–86.

[20] ——, *A homotopy for solving polynomial systems*, Appl. Math. Comput., 18 (1986), pp. 87–92.

[21] ——, *Solving polynomial systems using continuation for engineering and scientific problems*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[22] A. P. MORGAN AND L. T. WATSON, *A globally convergent parallel algorithm for zeros of polynomial systems*, J. Parallel Distributed Comput., to appear.

[23] D. PARKINSON, *The solution of N linear equations using P processors*, Parallel Computing 83, North Holland, Amsterdam, 1984, pp. 81–87.

[24] W. PELZ AND L. T. WATSON, *Message length effects for solving polynomial systems on a hypercube*, Parallel Comput., to appear.

[25] D. A. REED AND M. L. PATRICK, *A model of asynchronous iterative algorithms for solving large sparse linear systems*, Proc. 1984 Internat. Conf. on Parallel Processing, August 21–24, 1984, pp. 402–410.

[26] W. C. RHEINBOLDT AND J. V. BURKARDT, *Algorithm 596: A program for a locally parameterized continuation process*, ACM Trans. Math. Software, 9 (1983), pp. 236-241.

[27] T. A. RICE AND L. J. SIEGEL, *A parallel algorithm for finding the roots of a polynomial*, Proc. Internat. Conf. Parallel Processing, Bellaire, MI, Aug. 24–27, pp. 57–61, 1982.

[28] H. SCHWANDT, *Newton-like interval methods for large nonlinear systems of equations on vector computers*, Computer Phys. Comm., 37 (1985), pp. 223–232.

[29] C. L. SEITZ, *The cosmic cube*, Commun. ACM, 28 (1985), pp. 22–33.

[30] L. F. SHAMPINE AND M. K. GORDON, *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*, W. H. Freeman, San Francisco, 1975.

[31] H. J. SIPS, *A parallel processor for nonlinear recurrence systems*, Proc. 1st Internat. Conf. on Supercomputing Systems, IEEE Computer Society Press, Los Alamitos, CA, 1984, pp. 660–671.

[32] L. T. WATSON AND D. FENNER, *Chow-Yorke algorithm for fixed points or zeros of C^2 maps*, ACM Trans. Math. Software, 6 (1980), pp. 252–260.

[33] L. T. WATSON, *A globally convergent algorithm for computing fixed points of C^2 maps*, Appl. Math. Comput., 5 (1979), pp. 297–311.

[34] L. T. WATSON, S. C. BILLUPS, AND A. P. MORGAN, *HOMPACK: A suite of codes for globally convergent homotopy algorithms*, Tech. Rep. 85–34, Dept. of Industrial and Operations Eng., Univ. of Michigan, Ann Arbor, MI, 1985, and ACM Trans. Math. Software, 13 (1987), pp. 281–310.

[35] L. T. WATSON, *Numerical linear algebra aspects of globally convergent homotopy methods*, Tech. Report TR–85–14, Dept. of Computer Sci., VPI&SU, Blacksburg, VA, 1985, and SIAM Rev., 28 (1986), pp. 529–545.

[36] R. WHITE, *Parallel Algorithms for Nonlinear Problems*, SIAM J. Algebraic Discrete Methods, 7 (1986), pp. 137–149.

[37] R. WHITE, *A Nonlinear Parallel Algorithm with Application to the Stefan Problem*, SIAM J. Numer. Anal., 23 (1986), pp. 639–652.

Table 1. Execution time (secs).

Problem number	total degree	80286/ 80287	iPSC–32[1]	VAX 11/750	VAX 11/780	IBM 3090/200	SUN-2 /50	SUN-3 /160	CRAY X-MP
102(4)	256	16257	645	2438	1248	77	10545	8041	251
103(4)	625	34692	1616	5260	2656	163	22634	17126	535
402(2)	4	255	54	41	18	2	158	111	5
403(2)	4	84	19	14	6	1	54	38	1
405(2)	64	3669	335	703	334	14	2958	2429	84
601(2)	60	9450	257	1707	796	78	7417	5903	249
602(2)	60	28783	2795	4332	2054	124	21897	16831	462
603(2)	12	1200	243	325	152	9	1339	1060	37
803(8)	256	—	11527	29779	16221	667	130311	77113	2523
1702(4)	16	1655	163	216	112	7	986	658	21
1703(4)	16	1657	162	216	112	7	984	658	21
1704(4)	16	1628	108	216	112	6	1005	667	21
1705(4)	81	14336	378	1884	999	55	8907	6313	204
5001(8)	576	—	11786	49736	27815	1997	—	237685	5148

Table 1 (continued). Execution time (secs).

Problem number	total degree	Balance 21000	Balance 21000[1]	Balance 21000[2]	Elxsi 6400	Elxsi 6400[1]	Elxsi 6400[2]
102(4)	256	4436	682	1120	508	63	442
103(4)	625	9316	1428	2379	1081	127	936
402(2)	4	72	18	37	10	7	11
403(2)	4	23	6	12	3	3	5
405(2)	64	1199	171	605	124	26	107
601(2)	60	3499	822	1321	392	105	289
602(2)	60	6775	983	2899	769	135	558
603(2)	12	577	133	334	63	20	64
803(8)	256	33746	6094	9961	6991	759	3045
1702(4)	16	399	81	140	50	15	37
1703(4)	16	399	81	142	50	15	37
1704(4)	16	399	68	129	50	11	34
1705(4)	81	3507	494	1053	426	53	267
5001(8)	576	97550	17829	32049	17449	1829	9051

Table 1 (continued). Execution time (secs).

Problem number	total degree	Alliant $FX/8$	Alliant $FX/8^1$	Alliant $FX/8^2$	Encore Multimax
102(4)	256	362	52	215	1022
103(4)	625	769	108	457	2157
402(2)	4	6	2	4	21
403(2)	4	2	1	1	5
405(2)	64	96	16	55	287
601(2)	60	245	35	126	836
602(2)	60	793	148	406	2317
603(2)	12	47	13	32	133
803(8)	256	4459	711	1642	10428
1702(4)	16	36	9	16	91
1703(4)	16	36	9	16	92
1704(4)	16	35	7	15	91
1705(4)	81	308	46	134	800
5001(8)	576	11579	1765	4736	28969

Table 2. Efficiency: [(serial time)/(parallel time)]/(number of processors used).

Problem number	total degree	iPSC-32^1	Balance 21000^1	Balance 21000^2	Elxsi 6400^1	Elxsi 6400^2	Alliant FX/8^1	Alliant FX/8^2
102(4)	256	.76	.81	.99	.81	.29	.87	.42
103(4)	625	.65	.82	.98	.85	.29	.89	.42
402(2)	4	.94	1.00	.97	.36	.40	.72	.83
403(2)	4	.88	.96	.96	.25	.28	.65	.80
405(2)	64	.33	.88	.99	.48	.58	.75	.87
601(2)	60	1.11	.53	1.32	.37	.68	.88	.97
602(2)	60	.31	.86	1.17	.57	.69	.67	.98
603(2)	12	.38	.54	.86	.32	.49	.45	.74
803(8)	256	—	.69	.68	.92	.29	.78	.34
1702(4)	16	.60	.62	.71	.33	.34	.48	.54
1703(4)	16	.60	.62	.70	.33	.34	.47	.54
1704(4)	16	.89	.73	.77	.45	.37	.67	.57
1705(4)	81	1.15	.89	.83	.80	.40	.84	.58
5001(8)	576	—	.68	.61	.95	.24	.82	.31

Table 3. Comparison of efficiencies on Alliant FX/8.

Problem number	total degree	Coarse Grain 8	6	4	2	Fine Grain
102(4)	256	.87	.92	.96	1.00	.42
103(4)	625	.89	.92	.95	1.00	.42
402(2)	4	—	—	.71	.89	.83
403(2)	4	—	—	.64	.79	.80
405(2)	64	.75	.89	.95	.98	.87
601(2)	60	.88	.89	.93	.99	.97
602(2)	60	.67	.75	.84	.93	.98
603(2)	12	.45	.58	.72	1.00	.74
803(8)	256	.78	.88	.93	.98	.34
1702(4)	16	.48	.62	.80	.97	.54
1703(4)	16	.47	.62	.79	.98	.54
1704(4)	16	.67	.79	.85	.97	.57
1705(4)	81	.84	.89	.93	.98	.58
5001(8)	576	.82	.91	.95	.99	.31