Electronic Transactions on Numerical Analysis.
Volume 6, pp. 246-254, December 1997.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

FAST SOLUTION OF MSC/NASTRAN SPARSE MATRIX PROBLEMS USING A MULTILEVEL APPROACH*

C.-A. THOLE†, S. MAYER‡, AND A. SUPALOV§

Abstract. As part of the European Esprit project EUROPORT, 38 commercial and industrial simulation codes were parallelized for distributed memory architectures. During the project, sparse matrix solvers turned out to be a major obstacle for high scalability of the parallel version of several codes. The European Commission therefore launched the PARASOL project to develop fast parallel direct solvers and to test parallel iterative solvers on their applicability and robustness in an industrial framework.

This paper presents initial results using a special multilevel method as preconditioner for matrices resulting from MSC/NASTRAN linear static analysis of solid structures. P-elements in MSC/NASTRAN allow the polynomial degree of the base functions to be specified either globally or for each element. Solution dependent adaptive "refinement" of the p-level can be selected. Discretisations with lower p-level can therefore be used as coarser grids for a multilevel method.

Tests have been performed using such a method as preconditioner for a regular cube and a complicated industrial part, which were modelled by tetrahedrons and hexagonal elements. Preliminary performance comparisons on small test cases (about 10000 degrees of freedom) indicate that the multilevel approach is at least as fast as the currently available fastest iterative MSC/NASTRAN solver. Substantial performance improvements are expected for full-size industrial problems.

Key words. finite elements, multigrid methods, parallel computation.

AMS subject classifications. 65N30, 65N55, 65Y05.

1. Introduction.

1.1. Parallel computing needs iterative solvers.

The significant amount of work required to move large commercial application programs to parallel architectures in a portable way has hindered the acceptance of parallel technology in industry, thereby preventing this technology from being fully utilised to increase competitiveness. Therefore, the European Commission supported industry in Europe through the ESPRIT initiative EUROPORT by partially funding the porting of 38 industrially relevant codes (including PAM-CRASH, MSC/NASTRAN, POLYFLOW, STAR-CD, and others) to parallel computers [11].

EUROPORT adopted the message passing paradigm for parallelization that, clearly, requires some programming effort. However, it provides the only way to obtain portability across virtually all HPC platforms including those with shared memory and clusters of workstations.

Figure 1.1 shows the performance of the parallel CFD-code POLYFLOW using two different solvers [10]. POLYFLOW is a fully implicit special purpose CFD code for strongly non-linear problems involving rheologically complex liquid flows (plastic) in industrial processing applications. The coat hangar die test case simulates the production of thin foils, a fluid flow problem involving in each time step the solution of a sparse matrix problem with 48000 degrees of freedom (dofs). Figure 1.1 shows that the direct solver actually slows down, if more than 16 processors are used. An iterative solver based on domain decomposition is even faster and more than 16 nodes can be used efficiently.

* Received May 17, 1997. Accepted for publication October 24, 1997. Communicated by D. Melson. This work was supported by the European Commission via the Esprit Project 20160 PARASOL.
† GMD-SCAI.WR, Schloß Birlinghoven, D-53754 Sankt Augustin, Germany (thole@gmd.de).
‡ MacNeal Schwendler GmbH, Innsbrucker Ring 15, D-81612, Munich, Germany (stefan.mayer@macsch.com).
§ GMD-SCAI.WR, Schloß Birlinghoven, D-53754 Sankt Augustin, Germany (supalov@gmd.de).

ETNA
Kent State University
etna@mcs.kent.edu

As representative for performanc results from structural analysis simulation, figure 1.2 shows the elapsed times for the parallel sparse matrix solver used in MSC/NASTRAN. MSC/NASTRAN is a widely used structural analysis code and its parallel sparse matrix solver is highly optimized. The test case is a sparse matrix resulting from static analysis of a BMW-3 series car body, which consists out of 249911 dofs mostly resulting from shell elements. For the solution of this matrix problem a speed-up of 4.9 is achieved on 8 processors. This is an excellent result for a sparse direct matrix solver, however, in general a higher scalability is desired in order to (cost-)efficiently solve much larger problems as they are used in industry today.

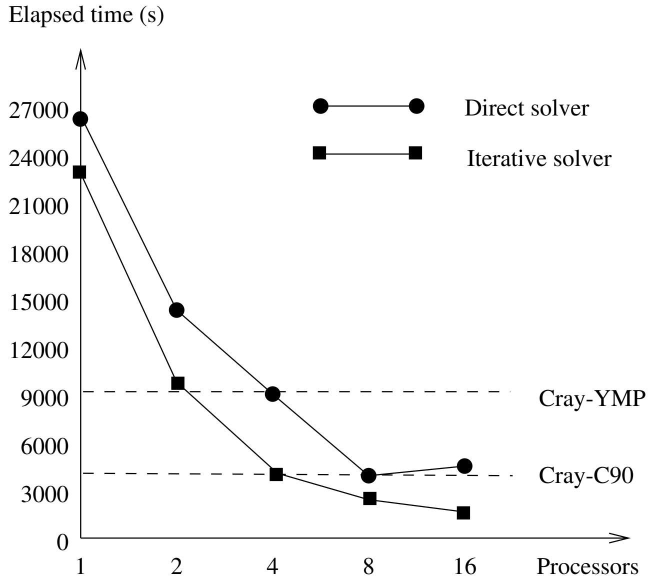

FIG. 1.1. *Performance of POLYFLOW on an IBM-SP2 for the coat-hanger-die test case.*

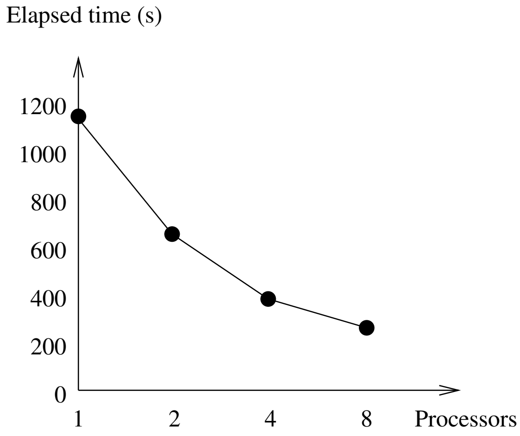

FIG. 1.2. *Performance of MSC/NASTRAN on an IBM-SP2 for the BMW-3 series test case.*

In order to overcome this situation the Esprit PARASOL project was launched.

1.2. PARASOL - An integrated programming environment for parallel sparse matrix solvers. Considerable progress has been achieved in the development of fast and robust iterative solvers for the solution of sparse matrix problems resulting from the discretisation of partial differential equations. This holds in particular for domain decomposition algorithms and hierarchical solvers like multigrid methods. However, most of the commercial codes are still structured such that the sparse matrix problem is set up and afterwards solved without any additional information about the underlying partial differential equations. This limits the use of iterative solvers to CG methods using incomplete factorisations as preconditioners (and to Algebraic Multigrid).

The aim of PARASOL[9] is to bring together developers of industrial finite element codes and experts in the area of parallel sparse matrix solvers in order to test innovative solvers in an industrial environment for performance and robustness. Direct and iterative parallel sparse matrix solvers will be developed and integrated into the PARASOL library. Based on the Rutherford-Boeing sparse matrix file format specification [5], an interface specification was developed suited for parallel machines. This parallel interface is also able to transport geometry and other information which might be needed by iterative solvers. Based on this interface, parallel direct solvers as well as iterative sparse matrix solvers are being developed by CERFACS, GMD, ONERA, RAL and University Bergen. The iterative solvers use domain decomposition methods and hierarchical approaches. Industrial simulation codes will be modified in such a way that they support the parallel interface and will evaluate the different solvers on industrial test cases. In addition to the CFD-code POLYFLOW, PARASOL involves the structural analysis codes MSC/NASTRAN and DNV SESAM as well as the metal forming code INDEED from INPRO and ARC3D from APEX (simulation of composite rubber metal part deformations).

Typical problem sizes for MSC/NASTRAN and DNV SESAM are in the order of 10^6–10^7 degrees of freedom. Metal forming and CFD codes require the solution of a sparse matrix in each time step. This restricts the problem size to about 10^5–10^6 in these cases.

2. MSC/NASTRAN p-element discretisations. The hierarchical solver developed by GMD has been evaluated first on test cases generated by MSC/NASTRAN. MSC/NASTRAN is a commercial structural analysis code used for the analysis and optimisation of the static and dynamic behavior of structures. This includes a variety of products such as cars, airplanes, satellites, ships and buildings. An overview of the technical basis of MSC/NASTRAN is given in [7].

In order to support the user in achieving the desired accuracy with minimal effort, MSC has extended the MSC/NASTRAN element basis by p-elements. For these elements, an error estimator is provided and in several steps the order of each base function is adapted until the desired accuracy is reached. Neighbouring elements might arrive at different order (even in different directions). Solid elements, which may be used as p-elements, are the standard solid elements (see also [7]):

1. HEXA: six sided solid element,
2. PENTA: five sisded solid element,
3. TETRA: four sided solid element.

MSC/NASTRAN supports adaptive p-elements for linear static as well as normal modes analysis and non-adaptive p-elements also for frequency response and transient analysis. For each of the p-elements or for a group, it is possible to specify an initial and maximal order of the base functions or to set a uniform p-order.

3. A hierarchical solver exploiting p-elements. A multigrid solver requires the definition of a coarse grid, interpolation, restriction, coarse-grid problem and the smoothing procedures. In the context of unstructured grids for finite element applications, four different approaches have been proposed:

1. Hierarchy of nested grids. A number of theoretical papers on hierarchical methods assume that a sequence of grids is already given. Each finer grid is a strict refinement of a coarser grid and therefore the vector space of the coarse grid base functions are a subset of the vector space of the base functions related to the fine grid [4]. In practice, this approach can only be applied, if the finite element solver uses some kind of adaptive refinement.

2. Merging finite elements. PLTMG [2, 3], is an example in which elements are merged according to certain rules. In this way a non-nested sequence of grids may be generated. Edge swapping is used to further improve the coarsening.

3. Fictitious space method. The fine grid is projected onto an auxiliary regular grid [8]. The solution on the regular grid by multigrid methods is used as a preconditioner in an iterative gradient algorithm for the unstructured grid problem.

4. Algebraic Multigrid. A subset of the degrees of freedom and related matrix, interpolation and restriction are defined solely on the basis of the matrix coefficients [6]. Algebraic Multigrid has been successfully applied problems resulting from a single partial differential equation and promising results are also reported for matrices resulting from systems of equations [12].

The approach used for MSC-NASTRAN p-elements is based on the first strategy, using the following definitions:

1. G_h the discretisation of the computational domain;

2. $V_p(G_h)$ vector space spanned by the base functions generated by the p-element approach;

3. $V_1(G_h)$ vector space spanned by the linear base functions;

4. $V_p^1(G_h)$ subspace of all $f \in V_p(G_h)$, for which f evaluates to 0 at the nodal points of G_h.

We have

$$V_1(G_h) \subset V_p(G_h), \quad V_1(G_h) \subset V_p(G_h) \quad \text{and} \quad \text{span}\left(V_1(G_h) \cup V_p^1(G_h)\right) = V_p(G_h).$$

In MSC/NASTRAN, the stiffness matrix K and vectors u for p-elements are organised as follows:

$$K = \left[\begin{array}{cc} K_{11} & K_{1d} \\ K_{d1} & K_{dd} \end{array}\right], \quad u = \left[\begin{array}{c} u_1 \\ u_d \end{array}\right].$$

K_{11} and u_1 correspond to the degrees of freedom related to $V_1(G_h)$, K_{dd} and u_d to those of $V_p^1(G_h)$. For the solution of

$$Ku = f,$$

two algorithms similar to those discussed by Axelsson [1] were implemented to improve the approximation of the solution u.

Method D:

$$\begin{array}{rcl} u^{\nu,0} & = & u^{\nu} \\ u_d^{\nu,1} & = & {K_{dd}}^{-1}\left(f_d - K_{d1}u_1^{\nu,0}\right) \\ u_1^{\nu,2} & = & {K_{11}}^{-1}\left(f_1 - K_{1d}u_d^{\nu,1}\right) \\ u_d^{\nu,2} & = & {K_{dd}}^{-1}\left(f_d - K_{d1}u_1^{\nu,2}\right) \\ u^{\nu+1} & = & u^{\nu,2} \end{array}$$

ETNA
Kent State University
etna@mcs.kent.edu

Method R(n):

$$
\begin{array}{rcl}
u^{\nu,0} & = & u^{\nu} \\
\text{for } i & = & 1 \text{ to } n \text{ do} \\
 & & u_1^{\nu,i} = \mathrm{RELAX}\left(K_{11}, \left(f_1 - K_{1d} u_1^{\nu,i-1}\right), u_1^{\nu,i-1}\right) \\
 & & u_d^{\nu,i} = \mathrm{RELAX}\left(K_{dd}, \left(f_d - K_{d1} u_1^{\nu,i}\right), u_d^{\nu,i-1}\right) \\
u_1^{\nu,n+1} & = & {K_{11}}^{-1}\left(f_1 - K_{1d} u_d^{\nu,n}\right) \\
u_d^{\nu,n+1} & = & u_d^{\nu,n} \\
\text{for } i & = & n+2 \text{ to } 2n+1 \text{ do} \\
 & & u_d^{\nu,i} = \mathrm{RELAX}\left(K_{dd}, \left(f_d - K_{d1} u_1^{\nu,i}\right), u_d^{\nu,i-1}\right) \\
 & & u_1^{\nu,i} = \mathrm{RELAX}\left(K_{11}, \left(f_1 - K_{1d} u_1^{\nu,i-1}\right), u_1^{\nu,i-1}\right) \\
u^{\nu+1} & = & u^{\nu,2n+1}
\end{array}
$$

Here, $\mathrm{RELAX}\,(K, f, \bar{u})$ performs a smoothing step on the problem $Ku = f$ with $\bar{u}$ as initial approximation to the solution. Pointwise Gauß-Seidel iterations have been used for the numerical experiments.

Method D-CG and method R(n)-CG denote algorithms, in which the respective procedures have been used as the preconditioner for a conjugate gradient method.

Method D corresponds to a two-space method. In this case, three large systems of equations have to be solved in each iteration step. Because this is very expensive, this method has been used only for comparisons with method R(n). Methods R(n) and R(n)-CG are called p-solvers in the rest of this paper.

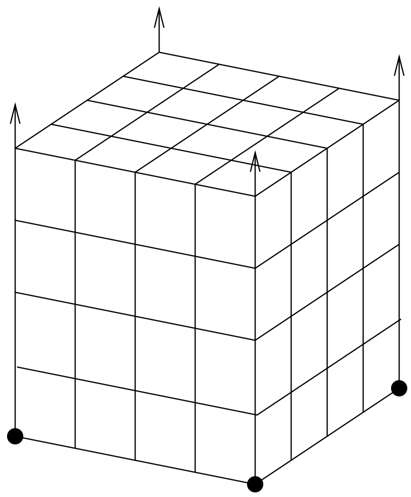

FIG. 4.1. *BCELL test case.*

4. Numerical results for a model problem. Figure 4.1 shows the grid for the 3D BCELL test case, in which HEXA p-elements are used to approximate a cube. The four corners at the bottom are fixed, and parallel forces are applied to the top corners of the cube.

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 4.1
BCELL test cases.

test case	p-level	size	linear dofs	total dofs	non-zeros
BCELL4.2	2 fixed	$4 \times 4 \times 4$	363	1263	153153
BCELL8.2	2 fixed	$8 \times 8 \times 8$	2175	8007	1153991
BCELL8.2a	2 adaptive	$8 \times 8 \times 8$	2175	7995	675393

TABLE 4.2
Reduction factors for the BCELL test cases.

test case	method					
	D	R(1)	R(2)	R(3)	R(4)	R(5)
BCELL4.2	.876	.860	.745	.702		.666
BCELL8.2	.865	.850	.740	.703	.684	.673
BCELL8.2a	.877					

Table 4.1 shows the basic properties of the different variants of the test case. The small difference between the number of degrees of freedom for the cases with fixed and adaptive p-level 2 is due to bubble functions, which are used in addition to the linear base functions for those elements, which remain linear. A factor 4 between the linear degrees of freedom and the total degrees of freedom for the BCELL8.2 case results in a reduction of about 10 of the effort for an LU decomposition.

Tables 4.2 and 4.3 show performance results for the BCELL test cases. For methods D and R(n), in Table 4.2 (as well as in the other tables in this paper), the reduction factor for one iteration of the l_2-norm of the residual after 100 iteration is given. In the case of CG methods (cf. Table 4.3), the number of iterations to reduce the l_2-norm of the residual by 6 orders of magnitude is given. The results for BCELL4.2 and BCELL8.2 indicate that the reduction factor and the number of CG iterations is independent of the grid size. The results show also that R(n)-CG is much more efficient than R(n) for this test case. (R(n) needs 73 iterations for 6 digits residual reduction.) A comparison with D-CG and D shows that some coarse-grid frequencies benefit from CG-acceleration. The solution of test case BCELL8.2a by R(n) requires about twice as many iteration steps as BCELL8.2. A comparison with D-CG shows that in this case the smoothing is not as effective as before. The reason for this deterioration might be those elements which interface second order elements with first order elements.

In order to improve the smoothing, experiments have been performed using block relaxation. 3x3 blocks and 15x15 blocks ensure that all dofs related to one node are treated together. However, the results in Table 4.4 show no improvements. Block relaxation involving all dofs related to the same element might yield some improvement.

The behavior of the p-solver for tetrahedral elements was evaluated with the TET5000-4-P test case. For this test case, each cube of the BCELL geometry is divided into 5 TETRA elements and a fixed order of 2 was requested for all elements. Table 4.5 shows that in this case the relation between the total number of dofs and the linear part is even more favourable. R(1)-CG requires 19 iterations like in the BCELL8.2 case.

5. Industrial test case. In addition to the model problems, one practical case has been considered. The KNUCK-MOD-P uses 454 HEXA and 121 PENTA elements. Figure 6.1 shows the geometry of this test case. The number of R(1)-CG iterations is larger compared to the BCELL8.2a problem (c.f. Table 5.1). This most likely results from the presence of distorted finite elements that are typically found in discretisations of the industrial designs. The methods intended for alleviation of this difficulty are currently under investigation.

TABLE 4.3
Number of iterations for the BCELL test cases.

test case	method					
	D-CG	R(1)-CG	R(2)-CG	R(3)-CG	R(4)-CG	R(5)-CG
BCELL4.2	20	19	14	12		11
BCELL8.2	20	19	14	12	11	11
BCELL8.2a	20	53	30	23	20	17

TABLE 4.4
Number of iterations for the BCELL test cases using block Gauß- Seidel relaxation as smoothener with various block sizes.

test case	block size	R(1)-CG	R(3)-CG
BCELL8.2	1×1	19	12
BCELL8.2	3×3	19	11
BCELL8.2a	1×1	53	23
BCELL8.2a	15×15	54	23

6. Performance comparison. The p-solver is implemented in Fortran 90 and uses the DGSF routine from the IBM ESSL library for sparse matrix decomposition. This solver is compared to MSC/NASTRAN (version 69) solvers on the basis of elapsed time on a dedicated IBM SP-2 thin2 node with 512 Mbytes of main memory. For all solvers the data fits into main memory and therefore swapping does not influence the computing times. Therefore, it should be mentioned that the reduced memory requirements of iterative solvers are not reflected. Also, highly optimised MSC/NASTRAN solvers are being compared to a prototype implementation of the p-solver. The measurements of Table 6.1 show that already for these small test cases the prototype implementation of the p- solver outperforms all tested iterative MSC solvers as of version 69. Only in the case of the KNUCK-MOD-P test case is the MSC direct solver faster.

For the p-solver, the time for the preparation and all iterations is given. The major part of the preparation time is spent in the DGSF sparse matrix decomposition routine; for example 8 seconds out of 13 for the KNUCK-MOD-P benchmark. The fact that the MSC direct solver needs only 13 seconds for a problem with 4 times as many degrees of freedom shows the quality of the MSC direct solver implementation and the improvement potential of the IBM ESSL library routine.

7. Summary. The results show that a hierarchical solver can efficiently treat second order sparse matrix problems. There is a difference of a factor of 4 in the effort which is needed for model problems and realistic test cases. In any case, the p-element solvers were able to outperform actual MSC/NASTRAN iterative solvers (as of version 69) on the models used for these tests.

With respect to parallelism, the p-solver is already a substantial improvement. The relaxation methods are fully parallelizable. The part of the direct solver will take only a fraction of the computing time, if efficient implementations are used or the direct solver is replaced by an iterative solver. Further improvements of the solver will therefore include advanced relaxation technology and the use of iterative (multigrid) solvers also for the linear problem.

REFERENCES

[1] O. AXELSSON AND I. GUSTAFSSON, *Preconditioning and two-level multigrid methods for arbitrary degree of approximation*, Math. Comp., 40 (1983), pp. 219–242.

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 4.5
Performance results for the TET5000-4-P test case.

test case	p-level	size	linear dofs	total dofs	non-zeros	R(1)-CG
TET5000-4-P	2 fixed	$10 \times 10 \times 10 \times 5$	3981	32871	2573533	19

TABLE 5.1
Performance results for the KNUCK-MOD-P test case.

test case	p-level	linear dofs	total dofs	non-zeros	R(1)-CG
KNUCK-MOD-P	2 fixed	2760	10827	1233457	85

[2] R. E. BANK, PLTMG: *A software package for solving elliptic partial differential equations*: *Users' guide 7.0*, Frontiers in Applied Mathematics, Vol. 15, SIAM, Philadelphia, 1994.
[3] R. E.BANK AND J. XU, *An algorithm for coarsening unstructured meshes*, Numer. Math., 73 (1996), pp. 1–36.
[4] J. H. BRAMBLE, J. E. PASZIAK, AND J. XU, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–22.
[5] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *The Rutherford-Boeing Sparse Matrix Collection*, Technical Report RAL-TR-97-031, Rutherford Appleton Laboratory, Oxon OX11 0QX, England, 1997.
[6] A. KRECHEL AND K. STÜBEN, *Operator dependent interpolation in algebraic multigrid*, in Proceedings of the 5th European Multigrid Conference, Stuttgart 01-04 Oct. 1996, Springer Lecture Notes in Comp. Science and Engineering, to appear.
[7] R. H. MACNEAL, *Finite Elements*: *Their Design and Performance*, Mechanical Engineering Series, L. Faulkner, ed., Vol. 89, Marcel Dekker Inc., 1994.
[8] S. NEPOMNYASCHIKH, *Fictitious space method on unstructured meshes*, East-West J. Numer. Math., 3 (1995), pp. 71–79.
[9] J. REID, A. SUPALOV, AND C.-A. THOLE, *PARASOL interface to new parallel solvers for industrial applications*, in Proceedings of the ParCo 97, Bonn, Germany, 1997, Elsevier Science Publishers, Amsterdam, to appear (see also `http://www.genias.de/parasol`).
[10] K. STÜBEN, H. MIERENDORFF, C.-A. THOLE, AND O. THOMAS, *EUROPORT: Parallel CFD for industrial applications*, in Parallel Computational Fluid Dynamics, P. Schiano, A. Ecer, J. Periaux, and N. Satofuka, eds., Elsevier Science Publishers, Amsterdam, 1997, pp. 39–48.
[11] ———, *Industrial parallel computing with real codes*, Parallel Comp., 22 (1996), pp. 725–737.
[12] P. VANEK, J. MANDEL, AND M. BREZINA, *Algebraic multigrid on unstructured meshes*, UCD/CCM Report 34, University of Colorado at Denver, 1994.

TABLE 6.1
Elapsed times on a IBM SP-2 thin2 node (512 Mbytes of memory).

test case	solver	elapsed times (sec)		
		total	preparation	iteration
TETRA-5000-4-P	p-solver	68	49	18
	MSC direct	216		
	MSC PBDJ (iterative)	75		
KNUCK-MOD-P	p-solver	46	13	34
	MSC direct	13		
	MSC PBCJ (iterative)	75		
	MSC PBDJ (iterative)	93		
	MSC PBDC (iterative)	69		

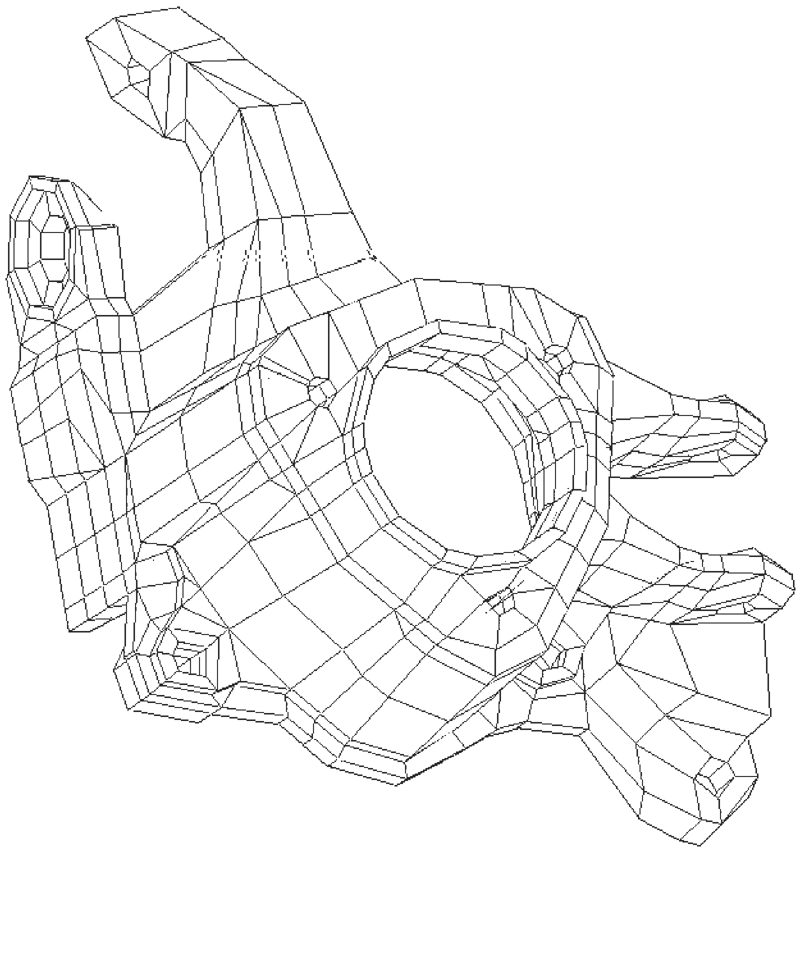

FIG. 6.1. *Geometry of the KNUCK-MOD-P test case.*