031155-3-T

User's and Test Case Manual for FEMATS

Arindam Chatterjee, John Volakis,
Mike Nurnberger and John Natzke

National Aeronautics
& Space Administration
Moffett Field, CA 94035

Naval Weapons Center
China Lake, CA 93555-6001

April 26, 1995
(2nd Revision)

31155-3-T = RL-2436

Report #031155-3-T

NASA Ames Grant NAG 2-866

Grant Title:	Development and parallelization of the Finite Element Method with Mixed Termination Schemes
Report Title*:	User's and Test Case Manual for FEMATS
Report Authors:	Arindam Chatterjee, John Volakis, Mike Nurnberger and John Natzke
Primary University Collaborator:	John L. Volakis volakis@um.cc.umich.edu Telephone: (313) 764-0500
Primary NASA-Ames Collaborator:	Alex Woo woo@ra-next.arc.nasa.gov
University Address:	Radiation Laboratory Department of Electrical Engineering and Computer Science The University of Michigan Ann Arbor, MI 48109-2122
Date:	April 1995

Funds for the support of this study have been allocated in part by the NASA-Ames Research Center, Moffett Field, California, under Grant No. NAG 2-866

User's and Test Case Manual for FEMATS

Arindam Chatterjee, John Volakis,
Mike Nurnberger and John Natzke

Radiation Laboratory
Dept. of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109-2122

April 26, 1995
(2nd Revision)

For FEMATS-related questions and bug reports, please call either John Natzke at (313) 747-1794 or Mike Nurnberger at (313) 764-0502.

This work was supported in part by NASA-Ames Research Center, Moffett Field, CA, under Grant #NAG2-866 (grant monitor Alex Woo; e-mail: woo@ra-next.arc.nasa.gov).

Contents

1 Introduction

The FEMATS program incorporates first order edge-based finite elements and vector absorbing boundary conditions into the scattered field formulation for computation of the scattering from three-dimensional geometries. The code has been validated extensively for a large class of geometries containing inhomogeneities and satisfying transition conditions (see [1] for formulation). For geometries that are too large for the workstation environment, the FEMATS code has been optimized to run on various supercomputers. Currently, FEMATS has been configured to run on the HP 9000 workstation, vectorized for the Cray Y-MP, and parallelized to run on the Kendall Square Research (KSR) architecture and the Intel Paragon.

2 Geometric Data Generation

The computation of scattering from a specific geometry with FEMATS is a multi-stage process, as shown in Figure 1. Once the geometric parameters of the target are known, a solid model is constructed in the Solid Modeling family of SDRC I-DEAS, a commercial CAD/CAE/CAM software package. The solid model is then exported to the Finite Element Modeling and Analysis family of I-DEAS, and the nodes and elements necessary for the scattering analysis are generated. This data is written to an output file, called a Universal file, and operated on by several preprocessors, generating the necessary input files for FEMATS.

The process of object modeling and mesh generation is an art, *not* a science. Hence, it cannot be taught, or demonstrated—it must be learned through experience. Hence, this manual is not by any means an I-DEAS FE mesh generation manual. In fact, it assumes (and requires) a working knowledge of, and familiarity with, the I-DEAS Solid Modeling and Finite Element Analysis families of tasks.

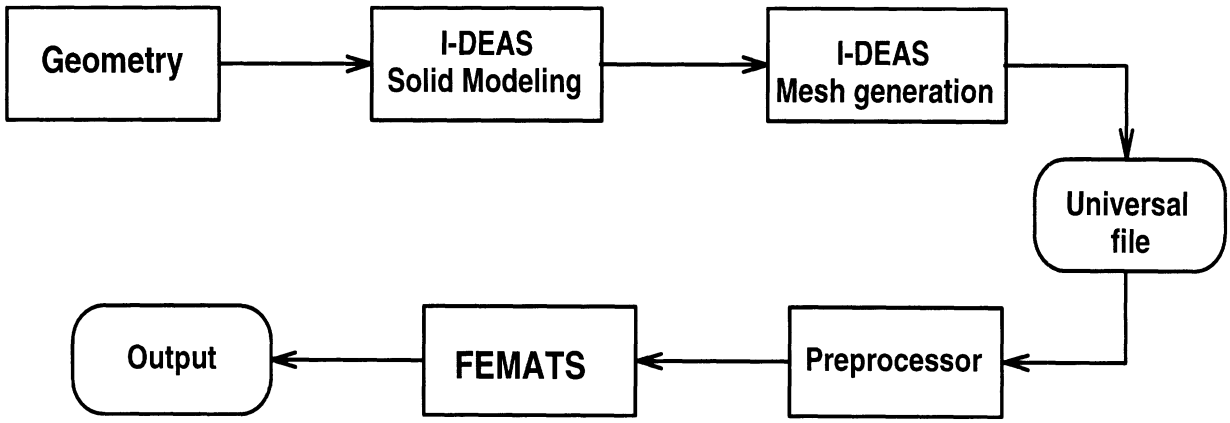


Figure 1: Stages involved in scattering computation from arbitrary 3D geometries

2.1 Solid modeling

Once the geometry of the target is specified, it is constructed using the I-DEAS Solid Modeling Family of tasks. There is a tendency to downplay the importance of the solid model, and treat it only as a stepping stone towards a final product. However, the solid model is the framework for the finite element mesh, and as such, has a direct bearing on the quality of the mesh. Because of this, it is wise to keep the mesh generation problem as simple as possible. This helps to ensure a better mesh, and a more accurate answer.

In general, the object or body being meshed will contain various planes of symmetry. It is nearly always advisable to take advantage of whatever symmetry is available, as doing so will greatly reduce the amount of time necessary to generate the mesh and the amount of storage space it requires. In fact, the geometry may require such subdivision to make meshing possible. For more details about the creation of the solid model for FEMATS, please see the I-DEAS Solid Modeling User's Guide.

Note: When creating the Solid Model, FEMATS requires the dimensions to be in wavelengths.

2.2 Mesh generation

I-DEAS generates the finite element mesh by creating mesh areas on surfaces, and then combining these mesh areas into mesh volumes ($2\frac{1}{2}$-D mesh generation). Each mesh volume is then filled with the chosen element type. When the solid model is imported into the Finite Element Modeling and Analysis Family of I-DEAS, these mesh areas and mesh volumes are automatically created. Generally, however, I-DEAS does not choose the correct element order (linear vs. parabolic), and the mesh volumes must be modified to reflect the correct element order. Even if the guidelines mentioned in Section 2.1 are followed, the mesh areas that are auto-created by I-DEAS can become quite complex. It is then prudent to break the mesh volume into smaller, more manageable mesh volumes. For more details on mesh creation, please see the I-DEAS Finite Element Modeling User's Guide.

2.3 Modifying material property labels

After all the mesh volumes have been created, the material property labels of each need to be modified according to the type of material each mesh volume contains. The elements in the volume between the target and the outer boundary usually have a material property label of 1. If the target contains a dielectric-filled volume, the material property labels of the elements in that volume should be 2. In this way, the code can accommodate up to 9 different material fillings. If the geometry requires more than 9 different materials,

the dimensions of the vectors ϵ and μ should be modified (wherever they appear) to reflect the necessary number of materials.

Please see Appendix A for more details.

2.4 Type of meshing

The global element length also needs to be specified (usually 0.075–0.085 units); finer meshing can be done in regions with rapidly changing fields or large curvatures by specifying the local element length or curvature-based size parameters. The geometry is then *free*-meshed using the I-DEAS Mesh Creation Task. It is essential to use *free* meshing and not *mapped* meshing, since the latter maps the mesh volume into a rectangular box and back, thus distorting the elements. No such distortion occurs in space when an electromagnetic wave travels through it; a *mapped* mesh, therefore, alters the physics of the problem and leads to inaccuracies in the final result.

Please see Appendix A for more details.

2.5 Grouping nodes

The finite element method essentially solves a boundary value problem; thus, it is crucial to identify surfaces or surface edges on which the boundary conditions are imposed. In the current version of FEMATS, this is carried out by grouping the nodes which lie on the surfaces where the boundary conditions are imposed. If the surface nodes coincide with a perfect electric conductor, the group is labeled C, if the nodes lie on a resistive sheet, the group is labeled R, and so on. Detailed information about node grouping is given in Appendix A.

Care must be taken when grouping surfaces that intersect, since edges connecting two such nodes may not lie on the surface at all. For example, suppose three surfaces intersect at the corner of a cube. If the nodes on each of these surfaces are not grouped separately, the processing program will generate 'surface' edges which actually do not lie on the surface. Another anomaly may arise when the surfaces are separated by a single element. This is due to the fact that the processing program considers an edge to lie on the surface if two nodes in the group connect to form an edge. As a rule of thumb, it is best to group each surface separately. They may be grouped together only when the user is certain that spurious surface edges will not be created by the processing program.

2.6 Universal file

The mesh information obtained from I-DEAS is then written to an ASCII file called the Universal file. The Universal file has a specific format for identifying the nodes, elements, and groups which can be obtained from the I-DEAS User's guide.

It should be noted that only the FE entities and Groups need to be included in the Universal file, and the specific format required by FEMATS is described in great detail in Appendix D. Also, while this discussion has dealt primarily with I-DEAS, any mesh generator that writes a Universal file will work just fine.

3 Running FEMATS

As shown in Figure 1, two processing stages are necessary to obtain the final output once the universal file has been generated for the specified geometry. The preprocessing stage is performed by a number of smaller programs that operate on the Universal file to construct the necessary geometry data files in the specific format required by FEMATS. For a given geometry, this operation only needs to be done once. FEMATS also requires the input parameters, such as the material properties, desired tolerance, maximum number of iterations, and aspect parameters (type of pattern, etc.), to be specified for each subsequent run. The main processor then reads in all of the geometry data and run-time parameters and returns the backscatter or bistatic pattern in a separate file. See Appendix B for a description of the code theory of operation.

The preprocessing and main processing programs are all initiated by a single script. The only exception is if the final processing is to be done on a different machine. In most cases this would be a supercomputer, and the preprocessing would have been done on the workstation. This requires that a second script be started on the supercomputer once the geometry and run-time data files have been transferred. It is recommended that all of the preprocessing be done on the workstation, since it is much more efficient.

To run the FEMATS script, type

`femats` *file*`.unv`

where *file*`.unv` is the name of the universal file containing the mesh information. All of the subsequently created files containing the geometry data and input parameters are given the same prefix *file* as the universal file. If the preprocessing of the universal file has not yet been done, these programs are started and the user must respond to a number of prompts, as explained below in Section 3.1. Otherwise, if this preprocessing has already been accomplished, the user is asked:

`Do you wish to preprocess` *file*`.unv again (y/n)?`

which may be necessary if a correction must be made in the preprocessing or if a change has been made to *file*`.unv`. In either case, the user is next prompted for the input parameters for that particular run, which are described in detail in Section 3.2. Upon completing the input parameters, the user is asked:

`Will the final processing be done on a different machine (y/n)?`

If no, then the final processing will be done on the same machine, and the FEMATS script will continue by initiating the main processor program. A complete example run is shown in Appendix C for this case using an HP workstation. Otherwise, if the final processing is to be done on a different machine, then the script will `tar` and `compress` all of the required data files, producing the single file *file*`.tar.Z`. This file should then be transferred to the other machine, and the appropriate script used to complete the processing as described in Section 3.3.

Note: For efficiency, the dimensioning of the arrays in the source files may be changed to reflect the size of the problem. For the preprocessor files, the relevant dimension statements are contained in the file

`(path)/femats/src/preproc/parmvl`

For the re-dimensioning to take effect, recompile the preprocessors by typing `make all` in the `(path)/femats/src/preproc` directory. Similarly, for the main processor source files, the relevant dimension statements are in the file

`(path)/femats/src/{arch}/fem_data.h`

For these changes to take effect, recompile the necessary files by typing `make all` in the `(path)/femats/src/{arch}` directory, where `{arch}` is the architecture (either `cray`, `hp`, `ksr`, or `paragon`). If any of the array dimensions are too small for a particular geometry, an overflow error will be given.

3.1 Preprocessing

The preprocessing stage requires a knowledge of the construction of the FE mesh contained within the universal file. The prompts given by the preprocessor programs are as follows:

`Does geometry mesh need to be reflected? 1) yes 2) no`

As discussed in Sections 2.1 and 2.2, it is judicious to take advantage of any inherent symmetry of the geometry when creating the FE mesh. If this has been done, then the first part of the preprocessor reflects the elements and nodes given the following information.

`Enter separation dist. for nodes to be coincident:`

A typical value is 0.001. This information is required so that the nodes lying on the reflection boundaries are not reflected.

`Reflect along 1) x-y plane 2) y-z plane or 3) z-x plane?`

The plane of reflection must be specified. Multiple reflections are done one at a time.

```
Point of reflection on w-axis:
```

where w = x, y, or z for the y-z, z-x, or x-y plane of reflection, respectively. Enter the point about which the reflection is to be made.

```
Reflect all permanent groups ? 1) yes 2) no
```

By making this distinction, the preprocessor can allow for some asymmetries in the geometry. If all of the groups are not to be reflected, then a list of the defined groups is given—

```
 Group #  Material Type
 -------  -------------
    1           C
    2           C
    3           A
    4           A
    5           O
    6           O
```

In this example, there are six permanent groups which have been defined in the universal file. The material types correspond to the definitions given in Appendix A, and the numbering corresponds to their order of inclusion in the universal file.

```
Total no. of groups not to be reflected:

Group(s) not to be reflected:
```

Enter the number of groups and then the specific groups identified by their group numbers which should not be reflected.

```
One more reflection? 1) yes 2) no
```

Repeat the above process until all of the reflections are completed.

Once the reflections (if necessary) have been completed, the user is asked to input the specifications of the outer boundary, i.e. the mesh termination. The code works for spherical, cylindrical, or flat mesh termination boundaries or any combination of these. All dimension values should be entered in units of wavelengths.

```
Choose type of outer boundary used to terminate
the mesh (i.e., its shape AFTER reflecting):
  1) spherical, 2) cylindrical, 3) rectangular box,
  4) mixed boundary
```

For the first three cases, the user is prompted for the center coordinates in (x, y, z) and the pertinent dimensions of the boundary shape. For an outer boundary having any other shape, the user should enter 4 to choose the mixed boundary type.

```
MIXED ---
Enter no. of separate surfaces in outer boundary:
```

The total number of distinct surfaces in the outer boundary must be entered for the mixed boundary type. For example, a cylinder with rounded ends has three separate surfaces—the cylindrical surface and the two spherical surfaces at the top and at the bottom.

```
Enter X,Y,Z coordinates of the center and the
radius of each surface:
 For a SPHERICAL section --
   Center coordinates xc yc zc and radius
 For a CYLINDRICAL section --
   X-directed axis - 200000. yc      zc      radius
   Y-directed axis - xc      200000. zc      radius
   Z-directed axis - xc      yc      200000. radius
 For a PLANAR section --
   Parallel to YZ plane - xc      200000. 200000. 0.
   Parallel to ZX plane - 200000. yc      200000. 0.
   Parallel to XY plane - 200000. 200000. zc      0.
```

The program needs the center coordinates and the radius of each distinct surface on the mesh termination for the mixed boundary type. For a spherical section, this poses no problem. In order to have complete information about a cylindrical section, we require its axis direction as well as the center coordinates and the corresponding radius of the curved surface. For example, a cylinder whose axis is oriented along the x-axis and is centered along $(x, 1, -1)$ with a radius of 2 is described by the following input line:

```
200000. 1. -1. 2.
```

Similarly, a flat planar section parallel to one of the principal planes is determined uniquely by specifying the magnitude of the constant coordinate. For example, the $z = 2$ plane is described by the following input line:

```
200000. 200000. 2. 0.
```

In this way, a combination of simple surfaces making up the outer termination boundary can be input to the program. FEMATS handles only these three surfaces types at present since these lead to symmetric systems of equations when incorporated into the finite element matrix. The theory can handle any doubly curved surface; they would, however, lead to an unsymmetric system of linear equations.

There remains one last set of specifications for the preprocessor—

```
Does input geometry contain any 2-D objects, or are they
   strictly 3-D (2/3)?
```

By definition, a 2-D object is one which is infinitesimally thin and bounded by FE mesh elements on both sides, e.g. an open PEC surface or an open or closed resistive surface. Included in this definition is the far-field integration surface when it is taken over an open surface. If there are any 2-D objects, then more information is required as follows:

```
Any 2-D objects contained within the geometry are
associated with one or more of the following surface groups:
  1) PEC, 2) R-card, 3) Impedance sheet, or
  4) Far-field integration surface

How many of these 4 surface group types describe 2-D objects?

Enter surface group type(s) as numbered above:
```

Enter the total number of surface group types which describe 2-D objects and then the specific surface group types as identified in the list.

3.2 Input parameters

The input parameters are those which the user may want to alter each time FEMATS is run. For the user's convenience, the default or previous parameter values appear in the brackets '[ ]' with each prompt; one simply hits return to use these values.

If there are two or more material property labels defined in the universal file, then the following prompts will occur:

```
The geometry contains n different material property labels --

For material property label 1, eps=(1.,0.), mu=(1.,0.).

For material property label i, enter

  eps: [ (1.,0.) ]

  mu: [ (1.,0.) ]
```

Input the complex constitutive relative parameters, ϵ and μ, for each material property label $i = 2, \ldots, n$. Note that material property label 1 is free-space by default. If the mesh designates material property label 1 to anything other than free-space, the program will not run.

If there is a resistive card inside the geometry (this version can handle only one), then the user must

```
Input: a) Material property label on top of card
       b) Material property label on bottom of card
       c) Normalized resistivity
```

The mesh must be constructed in such a way that the material property labels on the top and bottom surfaces of the R-card are different. Likewise, if there is an impedance sheet within the geometry, the user will be prompted to

```
Input: a) Material property label on top of impedance sheet
       b) Normalized impedance
```

The data entered until now has been related to the material properties of the geometry. The data entered from this point will be related to the iteration count, number of look angles, etc.:

```
Enter required tolerance: [ 0.0005 ]
```

The tolerance of the residual is usually kept between 0.001 and 0.0005. This is 0.1%–0.05% of the solution norm.

```
Maximum iterations: [ 15000 ]
```

The maximum number of iterations is determined by trial and error. A typical value for PEC targets is $N/100$ for $N > 25000$ and $N/120$ for $N > 75000$. The code uses a diagonally preconditioned biconjugate gradient method to solve the system, so the residual error will jump to abnormal values quite frequently.

```
1) Bistatic or 2) Backscatter: [ 2 ]
```

Enter 1 for a bistatic pattern, 2 for backscatter.

```
All angle values should be integers ---
Bistatic
--------
Angle of incidence --
Theta: [ 0 ]

Phi: [ 0 ]

Angle of observation --
Fix angle 1) phi or 2) theta? [ 1 ]

Specify value of fixed angle: [ 0 ]

For sweep angle theta, enter
 start: [ 0 ]

 end: [ 90 ]

 increment: [ 5 ]
```

Either ϕ or θ is set to a fixed angle. Then the other angle becomes the sweep angle.

```
Backscatter
-----------
Angle of incidence --
Fix angle 1) phi or 2) theta? [ 1 ]

Specify value of fixed angle: [ 0 ]

For sweep angle theta, enter
 start: [ 0 ]

 end: [ 90 ]

 increment: [ 5 ]
```

For each increment in the sweep angle for the backscatter case, FEMATS will generate 4 additional RCS data points between $\pm 2°$ of the increment angle using a bi- to monostatic conversion. Thus a 5° increment yields a computed pattern in 1° increments.

```
Polarization angle (alpha=0 => H_z=0, alpha=90 => E_z=0): [ 90 ]
```

Enter the desired polarization angle.

```
Do you wish to re-enter any of the input parameters (y/n)? [n]
```

The user is given the option of going through the run-time specifications over again to make any changes to the input parameters. Any parameters that had been entered become the default values for the next pass through.

3.3 Completing the FEMATS Run on a Supercomputer

The following instructions assume that the preprocessing has been done on a different machine, in most cases a workstation. The final processing of the job on the supercomputer is then completely non-interactive.

After the compressed tar file *file*`.tar.Z` has been transferred to the supercomputer, a script is run there to initiate the main FEMATS program. To run this script, type

`femats.{arch}` *file*

where {`arch`} is either `cray`, `ksr`, or `paragon`, and *file* is the prefix name of the original universal file. After uncompressing and untarring the given file, this script will initiate another preprocessor to convert the mesh data files

from ASCII to binary format. On the supercomputers, it is more efficient to read the mesh files in binary format, since the ASCII format is quite slow for most machines and prohibitively slow on the KSR. The script will then continue by initiating the main program. Specific run-time information is given below for each of the supported architectures.

Cray-specific run-time information

The code was only vectorized on the Cray series of machines. Thus it always runs on a single processor with enhanced vectorization. To execute FEMATS, type

`./femats.cray` *file*

as shown above.

KSR-specific run-time information

Before executing FEMATS, the user must inform the operating system of the required number of processors. To do this, type

```
allocate_cells -A ##
```

where **##** is the requested number of cells. This command starts a new shell, and gives that shell control of **##** cells. Because of this, it is important to exit the shell when FEMATS finishes, so that others may use the processor cells. After the new shell is running, the user must let the operating system know how many total threads it may run on the allocated cells by typing

```
setenv PL_NUM_THREADS ##
```

where **##** is the same as in the previous command. FEMATS may now be run safely.

If any problems arise, consult the KSR manuals, and the system administrator. The above steps assume that everything works the way it is supposed to.

Intel Paragon-specific run-time information

To execute FEMATS on the Paragon, type

`femats.paragon` *file* `-sz ##`

where **##** is the number of processors to be used.

Appendices

A Stipulations for Mesh Generation

- The region surrounding the scatterer should have a material property label of 1, i.e., the least possible value.
- For a surface draped by a resistive card, it is essential to differentiate the top surface from the bottom surface of the resistive card. The only way the program can discern this from the available data is by checking the material property labels of the elements on the top and bottom surfaces. The material property label of the top surface must therefore be different from that of the bottom surface.
- When meshing a mesh-volume filled with a dielectric having a certain permeability and permittivity, the material property label of the dielectric should be different from that of the surrounding mesh.
- When grouping surface nodes, the group labels should start with a
  - **C** for nodes that lie on a perfect electrical conductor
  - **R** for nodes that lie on a resistive card
  - **I** for nodes that lie on an impedance sheet
  - **D** for nodes that lie on a dielectric or on the interfaces of materials having different constitutive parameters
  - **A** for nodes that lie on the outer boundary (i.e. on the mesh termination)
  - **O** for nodes that lie on the far-field integration surface, usually the outer surface of the scatterer

  The above order (C, R, I, D, A, O) *must* be maintained when grouping nodes.

B Code Theory of Operation

proc.f

proc.f consists of the programs **u2c.f** and **c2p.f**. It converts the mesh information stored in the Universal file into a more usable form for analysis by FEMATS. It first reads in the nodal co-ordinates, nodal connectivity and the grouped nodes from the Universal file. Since FEMATS uses edge-based shape functions, the edges and the nodes connecting them need to be identified. Because each edge is shared by more than one element, care must be taken so that the same edge is not counted more than once. A comparison of the connecting nodes must therefore be made to identify the old edges, and create the new ones. This can be a computationally intensive task if a brute force approach is taken, especially if the problem size is very large. It is necessary to use an algorithm that would scale at most linearly with the number of nodes or edges, i.e. the number of comparisons required for identifying old or new edges should be an $O(N)$ process.

In order to realize this requirement, the ITPACK scheme [2] is utilized to store the node connectivity information. The ITPACK scheme is attractive because the number of comparisons required while augmenting the connectivity matrix depends only on the locality of the corresponding node and not on the total number of nodes or edges. In the ITPACK storage scheme, the number of rows of the connectivity matrix is equal to the number of nodes and the number of columns equals the maximum number of nodes connected to a particular node. However, this approach wastes space when the number of connecting nodes varies widely, so a modified ITPACK format is used—the number of columns in the connectivity matrix now equals the average number of nodes connected to a particular node, and the number of rows is slightly more than the total number of nodes. The storage requirement for such a matrix is usually $1.1N_n \times 16$ integers, where N_n equals the number of nodes.

After generating the edges, FEMATS uses the same storage scheme for finding the surface edges and elements from the grouped nodes. These surface edges are then sorted in ascending order by element number for the various materials and boundaries on which they lie. All components of the code are extremely fast, with the slowest being the sorting routine.

The output data files from **proc.f** are given the following extensions:

- **enode**
  Contains coordinates of all the nodes in the geometry.

- **eglob**
  Contains the edges making up each element.

- **edge**
  Contains the nodes making up each edge.

- **esize**
  Contains the number of nodes, elements, and edges in the geometry, as well as the number of elements with surface edges on the PEC, R-card, impedance, dielectric, outer boundary, and far-field integration surfaces. This information is also contained in **otpt** with explanatory text.

- **esurfed**
  Contains the element number, node numbers and corresponding edge numbers of the on-surface edges.

- **geom**
  Contains the number of different material property labels and the specifications of the surface(s) making up the outer boundary.

Required storage is about $18N$ real Words, where N is the number of unknowns and is equal to the number of edges making up the mesh.

count.f

count.f reads from the files **eglob** and **esize** and calculates the number of non-zero entries per row for the finite element system. The number usually varies from 9 to 31 for a typical system. The output is written to **cntr**, and the required storage is about $13N$ real Words, where N again denotes the number of unknowns.

prompt.f

This program prompts the user for the necessary run-time parameters required by FEMATS. These parameters include any material properties that need to be specified, the desired tolerance, maximum number of iterations, and the aspect parameters (type of pattern, etc.). The run-time data file that is created has the extension **in**.

fr.f

When the main processing of FEMATS is done on a supercomputer, **fr.f** is used to convert the mesh files from ASCII to binary format.

fem.f

This is the main program (FEMATS) which computes backscatter or bistatic patterns after reading in the geometry data files created by **proc.f** and **count.f** and the run-time parameter file created by **prompt.f**. The backscatter or bistatic pattern is returned in a separate file. If the code fails to run for some reason, a list of errors is returned in the error file. The flow of control

of FEMATS is given in Figure 2. The formulation for the methodology is given in [2].

Input files: The input files containing the geometry and mesh information and parameters for running the problem are read in. When run on the supercomputer, the mesh files are read in binary format, since the ASCII format is quite slow for most machines and prohibitively slow on the KSR.

Processing data: Some preliminary processing is done to verify the radii of the outer boundary surfaces which were declared in **proc.f**.

FE matrix generation/assembly: The finite element matrix generation is done on an element-by-element basis. The elemental matrix is first computed, and then assembled into the global sparse matrix. The assembly is simplified since the number of non-zero entries per row of the matrix is known *apriori* and the order of the entries is not important. The non-zero entries of the final sparse matrix are stored in a long complex vector, the corresponding column numbers are stored in an integer vector, and the location of the first non-zero entry for each row is stored in another integer vector. This is the well-known Compressed Sparse Row (CSR) format used in public domain software packages like SLAP and SPARSPAK. The coefficient matrix is not a function of the angle of incidence.

The code also uses a simple diagonal preconditioner for speeding up the iterative process. Other complicated preconditioning strategies are also available. However, except for the block ILU preconditioner, none of them compare favourably with the point diagonal preconditioner in terms of solution time.

Excitation vector generation: The excitation vector generation is not very CPU-intensive, since the vectors are always quite sparse. It is a function of the angle of incidence.

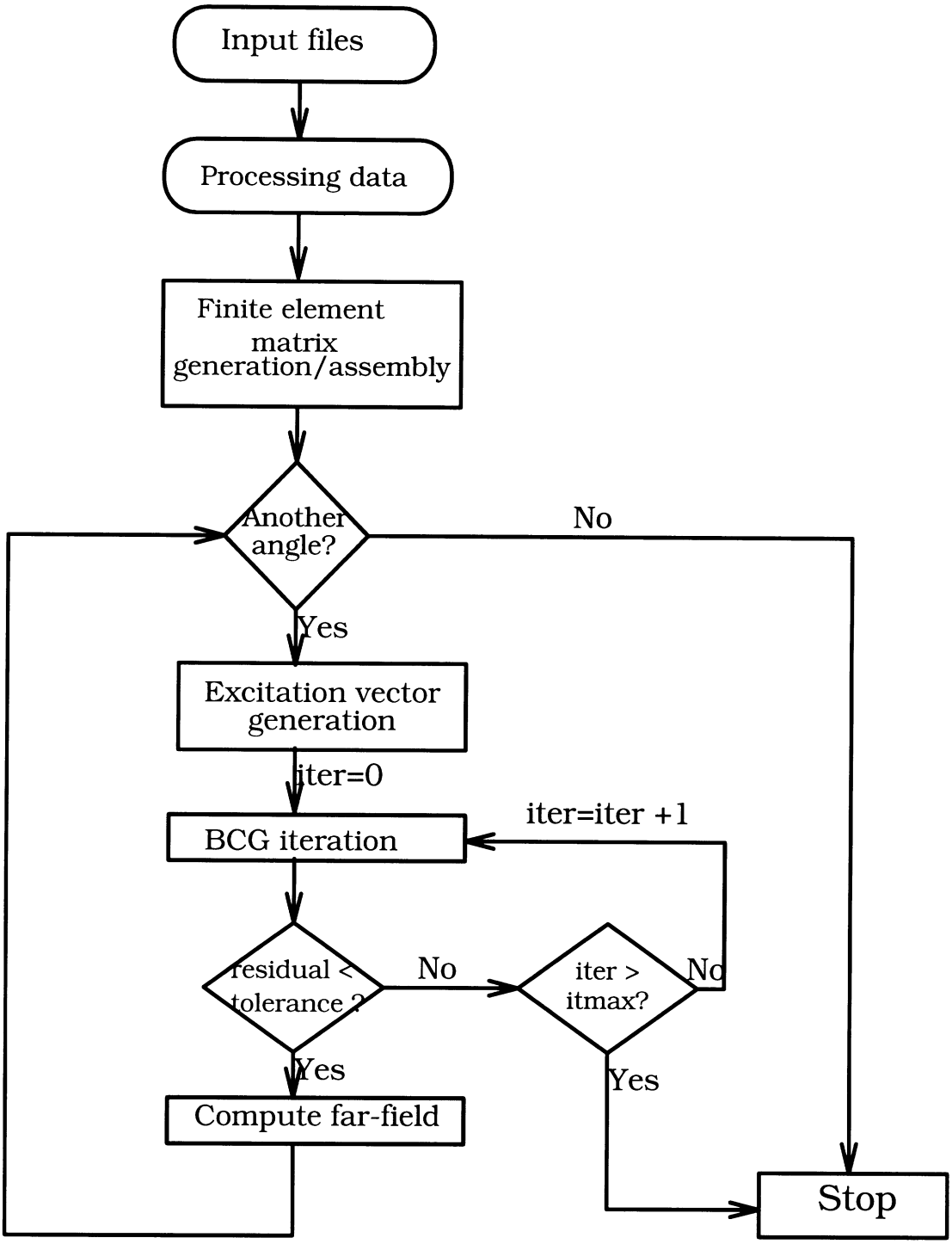
BCG iteration: The biconjugate gradient (BCG) algorithm is used with preconditioning to solve the sparse, symmetric system of linear equations. Each iteration of the algorithm involves 1 sparse-matrix vector product, 3 vector updates and 3 inner products. The norm of the residual vector is computed after every iteration to check for convergence. Reliable results have been obtained by setting the convergence criterion to be

$$\|r^k\| < 0.001 * \|b\|$$

where r^k is the residual vector after the kth iteration and b is the excitation vector.

Far-field evaluation: The far-field is evaluated by integrating the near-zone fields over a closed surface using the Stratton-Chu integral equation. The surface is usually taken to be very close to or on the body itself to achieve maximum accuracy.

Storage required for the code is at present $36N$ complex Words, where N is the number of unknowns. The storage can be cut by 40% if only the



Flowchart of finite element code

Figure 2: Flowchart for FEMATS

symmetric upper triangular part of the object matrix is stored; the code, however, slows down significantly.

Subroutine functions in fem.f

basis.f
Calculates the two constant vectors of the bases for the finite element discretization as well as the element volume.

bd1.f
Assigns each surface element on the mesh termination boundary its surface particulars.

bi2mono.f
Computes the far-field pattern for a particular angle of incidence. If a monostatic pattern is desired, then monostatic values are approximated for the computed angle of incidence as well as for the four adjacent angles between $\pm 2°$.

bicg.f
The diagonally preconditioned biconjugate gradient solver for symmetric, linear systems of equations.

calc.f
Computes the volume integral for the finite element discretization analytically.

comput.f
Calculates the basis functions at the mid-point of each edge.

crux.f
Computes the element matrix from the volume integral.

cruxd.f
Imposes the boundary condition for dielectric volumes and generates the corresponding excitation vector.

diag.f
Generates the diagonal preconditioner for solving the system of equations iteratively.

f1.f
Carries out the volume integration of $\mathbf{W}_i \cdot \mathbf{W}_j$ analytically.

fcmb.f
Carries out the surface integration for the absorbing boundary condition

employed on the mesh termination boundary.

finc.f
Computes the volume integral for a dielectric volume to be used in the excitation vector.

hpsrt.f
A standard heapsort routine.

incc.f
Imposes the boundary condition for a perfect electric conductor. If *iter* is 0, the excitation vector is computed, otherwise changes are made to the element matrix.

incd.f
Imposes the boundary condition for a dielectric surface.

inci.f
Imposes the boundary condition for an impedance sheet.

incr.f
Imposes the boundary condition for a resistive card.

mat.f
Performs finite element matrix generation and assembly.

mult.f
Carries out the sparse matrix-vector multiplication for the iterative solver.

norm2d.f
Computes the element normal for a 2D geometry.

norma.f
Computes the element normal for a surface element.

ops.f
Performs elementary vector and matrix operations.

ord.f
Identifies the global nodes and edges in the local context.

sort.f
Sorts the edges in a element according to a specific numbering scheme.

surfint.f
Imposes the absorbing boundary condition on the mesh termination boundary.

value.f
Computes the far-field using the Stratton-Chu integral equation.

volume.f
Calculates the element volume.

xc.f
Generates the excitation vector for each angle of incidence.

C Example FEMATS Run

A $0.3 \times 0.3\lambda$ perfectly conducting plate was run on an HP 9000/715 workstation. The geometry was enclosed by a spherical outer boundary and the backscatter pattern was sought for $\theta = 0$–$10°$ with $\alpha = 90°$. The problem had 9712 unknowns and a diagonally preconditioned BCG solver was used. (This transcript is contained in the subdirectory `femats/test/hp`*, along with the original universal file and output files.)*

```
tull% femats small_plate.unv

femats.pre: starting femats preprocessor...

u2c: processing I-DEAS Version 6 format Universal file small_plate.unv...
u2c: perform whatever manipulations are necessary.

 Name of universal file ?
  small_plate.unv
 There are 478 nodes in the geometry.
 There are 1846 elements.
 There are 3 groups containing 213 nodes.
 Max. volume = 8.45531E-04, Min. volume = 2.52417E-06
 Element with min. volume = 1278

 REFLECTION SPECIFICATIONS
 Does geometry mesh need to be reflected? 1) yes 2) no
1
 Enter separation dist. for nodes to be coincident:
.001
 Reflect along 1) x-y plane 2) y-z plane or 3) z-x plane?
2
 Point of reflection on x-axis:
0.
 There are now 881 nodes and 3692 elements
 Number of old groups =  3
 Reflect all permanent groups ? 1) yes 2) no
1
 Number of new groups =  6

 One more reflection? 1) yes 2) no
1
 Reflect along 1) x-y plane 2) y-z plane or 3) z-x plane?
3
 Point of reflection on y-axis:
0.
 There are now 1625 nodes and 7384 elements
 Number of old groups =  6
 Reflect all permanent groups ? 1) yes 2) no
```

```
1
 Number of new groups =  12

 One more reflection? 1) yes 2) no
2
 After reordering -
 There are 12 surface groups containing 852 nodes.

 OUTER BOUNDARY SPECIFICATIONS

 Choose type of outer boundary used to terminate
 the mesh (i.e., its shape AFTER reflecting):
   1) spherical, 2) cylindrical, 3) rectangular box,
   4) mixed boundary
1
 Enter center coordinates (x,y,z) and radius:
0. 0. 0. .45

u2c: done.

femats.pre:  Does input geometry contain any 2-D objects,
             or are they strictly 3-D (2/3)? 2

c2p_2d: Enter required data as necessary.
c2p_2d: please be patient...this could take a while...

 Reading in data . . .
 Be patient . . .
 Starting surface node generation...

 Any 2-D objects contained within the geometry are
 associated with one or more of the following surface groups:
   1) PEC, 2) R-card, 3) Impedance sheet, or
   4) Far-field integration surface

 How many of these 4 surface group types describe 2-D objects?
2
 Enter surface group type(s) as numbered above:
1 4
 Finding/sorting surface edges and faces . . .
 Problem size ---
  There are 1625 nodes and 7384 elements
  Edge count =  9712
   180 element faces/edges on the pec
   0 surface elements on resistive sheet
   0 surface elements on impedance sheet
   0 surface elements on dielectric
```

```
   1408 surface elements on outer boundary
   64 surface elements on integration surface

c2p_2d: done.

count: processing data files.
count: please be patient...this could take a while...

  Average bandwidth  15
  Peak bandwidth  31

count: done.

femats.pre:  the following geometry data files have been created:

    small_plate.cntr, small_plate.edgy, small_plate.eglob,
    small_plate.enoddy, small_plate.esize, small_plate.esurfed,
    small_plate.geom


prompt:  starting prompts for run-time input data...

 RUN-TIME SPECIFICATIONS

 NOTE:  to use default values appearing in the [],
        simply hit return.

 Enter required tolerance: [ 5.00000E-04 ]

 Maximum iterations: [ 15000 ]

 1) Bistatic or 2) Backscatter: [ 2 ]

 All angle values should be integers ---
 Angle of incidence --
 Fix angle 1) phi or 2) theta? [ 1 ]

 Specify value of fixed angle: [ 0 ]


 Note:  For each increment in the sweep angle, femats will
        generate 4 extra RCS data points between +/- 2 deg
        using a bi- to monostatic conversion.  Thus a 5 deg
        increment yields a computed pattern in 1 deg increments.

 For sweep angle theta, enter
  start: [ 0 ]
```

```
  end: [ 90 ]
10
  increment: [ 5 ]

 Polarisation angle (alpha=0 => H_z=0, alpha=90 => E_z=0): [ 90 ]

 Do you wish to re-enter any of the input parameters (y/n)? [n]

prompt:  the run-time data file small_plate.in has been created

femats:  Will the final processing be done on a different
         machine (y/n)? n

femats:  continuing with processing
fr_hp:  doing overflow check of dimensioned arrays

 Prefix name of input file:
  small_plate

fr_hp:  done.

femats:  starting femats main processor...

 Prefix name of input data files:
  small_plate
 ***********************************************
                Problem size
 Number of nodes = 1625
 Number of elements = 7384
 Number of edges/unknowns = 9712
 ***********************************************
 No. of surface faces on
  1) PEC                            :  180
  2) R-card                         :  0
  3) Impedance sheet                :  0
  4) Dielectric interface           :  0
  5) Outer boundary (mesh term.)    :  1408
  6) Far-field integration surface  :  64
 ***********************************************
 Outer boundary shape is
   Spherical
 ***********************************************
 Backscatter pattern will be computed
 Incident angle from 0 to 10
   in steps of 5
 Sweeping through theta; phi = 0
```

```
Polarisation angle = 90
**********************************************
Reading in element/edge/node data . . .
Time spent for unformatted I/O =  2.1484375 secs
Time spent for I/O =  2.2109375 seconds
**********************************************

Generating finite element matrix . . .
Outerloop --
     .0000           .0000         90.0000
Gen_vector_soln . . .
Solver . . .
  iter=  100, tol=  .05493, time=    18.4297s
  iter=  200, tol=  .00914, time=    36.3750s
  iter=  300, tol=  .00536, time=    54.2734s
  iter=  400, tol=  .00267, time=    72.0000s
  Convergence achieved in 435 iterations
  Time spent in 435 iterations = 78.234375 seconds
Comput_results . . .
Monostatic angle = .0 for incidence of 0
Backscatter RCS = -5.16235124511881 dB

Monostatic angle = 1.0 for incidence of 0
Backscatter RCS = -5.16404174641413 dB

Monostatic angle = 2.0 for incidence of 0
Backscatter RCS = -5.1691052748699 dB

    5.0000           .0000         90.0000
Gen_vector_soln . . .
Solver . . .
  iter=  100, tol=  .02634, time=    17.7031s
  iter=  200, tol=  .01121, time=    35.4063s
  iter=  300, tol=  .00413, time=    53.1172s
  iter=  400, tol=  .00180, time=    70.8984s
  iter=  500, tol=  .00205, time=    88.5938s
  Convergence achieved in 517 iterations
  Time spent in 517 iterations = 91.640625 seconds
Comput_results . . .
Monostatic angle = 3.0 for incidence of 5
Backscatter RCS = -5.18361322126989 dB

Monostatic angle = 4.0 for incidence of 5
Backscatter RCS = -5.18762443806741 dB

Monostatic angle = 5.0 for incidence of 5
Backscatter RCS = -5.1949990584817 dB
```

```
Monostatic angle = 6.0 for incidence of 5
Backscatter RCS = -5.20570715150933 dB

Monostatic angle = 7.0 for incidence of 5
Backscatter RCS = -5.21970412507492 dB

   10.0000           .0000         90.0000
Gen_vector_soln . . .
Solver . . .
  iter=  100, tol=  .03225, time=    17.8516s
  iter=  200, tol=  .00699, time=    35.5859s
  iter=  300, tol=  .00945, time=    53.2891s
  iter=  400, tol=  .01602, time=    71.0000s
  iter=  500, tol=  .00215, time=    88.7031s
  Convergence achieved in 524 iterations
  Time spent in 524 iterations = 92.984375 seconds
Comput_results . . .
Monostatic angle = 8.0 for incidence of 10
Backscatter RCS = -5.2621385331921 dB

Monostatic angle = 9.0 for incidence of 10
Backscatter RCS = -5.27505870524297 dB

Monostatic angle = 10.0 for incidence of 10
Backscatter RCS = -5.29124158611137 dB

Total computational time =  266.734375 seconds

Output files --

  Table of RCS data:         tab.mon.p0.90
  Total RCS vs. sweep angle: rcs.mon.p0.90

for a fixed angle phi = 0 with alpha = 90.
```

D I-DEAS Universal File Information

To facilitate development of data file translators for mesh generation packages other than I-DEAS, a brief (but hopefully complete) description of the universal file format and the pertinent datasets follows. Each universal file is a sequentially-formatted text file containing records a maximum of 80 characters long, and is divided into sections called datasets. Between datasets, the universal file may contain lines (e.g., comments) which are not part of any dataset and will be ignored.

Datasets

The first and last record of each dataset is marked by the dataset delimiter, which consists of a minus sign in column 5 and a numeric one in column 6. The remainder of the line is blank. The second record in the dataset is the dataset type. This is an integer between 1 and 32767, right-justified in columns 1 through 6. The remainder of the line is blank. The rest of the records in the dataset follow the format specified for that dataset in the I-DEAS manual. The pertinent ones for FEMATS are discussed below.

Universal file processing

Processing of the universal file begins by searching for the first dataset delimiter. Next, the dataset type line is read to determine whether the program needs to process this dataset. If the dataset is to be processed, the program reads the data per the specifications for that dataset. Otherwise, the program skips to the next dataset delimiter, which marks the end of the dataset, and begins the search process for another dataset delimiter again. This cycle continues until the end of file condition is reached.

It should be noted that the end of file condition should only occur between datasets. An end of file occuring during dataset processing indicates an incomplete dataset. Finally, datasets may occur in any order in the universal file, but with the restriction that a dataset may reference data in other datasets only if the other dataset precedes it in the file.

FEMATS-specific datasets

As mentioned above, FEMATS only requires two sets of data—the FE Entities and the Groups. This is equivalent to three datasets in the universal file—datasets 752, 780, and 781. These are, respectively, the Permanent Groups, the Elements, and the Double Precision Nodes datasets. A description of each follows, along with an example. The format description for each line is given in terms of the FORTRAN equivalent.

Dataset 752: Permanent groups

Record 1:	FORMAT(6I10)	
	Field 1	- group number
	Field 2	- active constraint set no. for group
	Field 3	- active restraint set no. for group
	Field 4	- active load set no. for group
	Field 5	- active dof set no. for group
	Field 6	- number of entities in group
Record 2:	FORMAT(20A2)	
	Field 1	- group name
Record 3–N:	FORMAT(8I10)	
	Field 1	- entity type code
	Field 2	- entity tag
	Field 3	- entity type code
	Field 4	- entity tag
	Field 5	- entity type code
	Field 6	- entity tag
	Field 7	- entity type code
	Field 8	- entity tag

Repeat Record 3 for all entities as defined by Record 1, Field 6. Records 1 thru N are repeated for each group in the model.

The group number (Record 1, Field 1) is strictly a numeric counter, starting at one. In FEMATS, Fields 2–5 of Record 1 are not used, and should be set to zero. Record 1, Field 6 is self-explanatory.

Record 2, Field 1 is a 40 character name, describing the group. In I-DEAS this name may have any number of spaces, etc. However, FEMATS expects certain naming conventions. As mentioned in Section 2.5 and Appendix A, the group names must begin with a certain letter, and occur in a certain order, as follows: C, R, I, D, A, and O (for conductor, resistive card, impedance sheet, dielectric, outer boundary, and far-field integration surface, respectively).

Records 3–N describe the entity type and identifying tag number. For FEMATS, the entity type code will always be 7—a finite element node. Type codes for other entities may be found in the I-DEAS Core Utilities manual.

Example:

```
    -1
   752
         1         0         0         0         0        14
C1
```

```
         7         1         7         2         7         3         7         4
         7         8         7         9         7        10         7        11
         7        12         7        13         7        14         7        15
         7        64         7        65
         2         0         0         0         0       146
A1
         7         5         7         6         7         7         7        27
         7        28         7        29         7        30         7        31
         7        32         7        33         7        34         7        35

etc....

         7       264         7       265         7       266         7       267
         7       268         7       269         7       270         7       271
         7       272         7       273
         3         0         0         0         0        14
01
         7         1         7         2         7         3         7         4
         7         8         7         9         7        10         7        11
         7        12         7        13         7        14         7        15
         7        64         7        65
    -1
```

In this example, there are three groups, numbered 1 through 3, and named C1, A1, and O1, respectively. Groups C1 and O1 have 14 entities associated with them, while group A1 has 146 entities. All the entities in all the groups are entity type 7—nodes.

Dataset 780: Elements

Record 1:	FORMAT(8I10)	
	Field 1	- element label
	Field 2	- fe descriptor id
	Field 3	- physical property table bin number
	Field 4	- physical property table number
	Field 5	- material property table bin number
	Field 6	- material property table number
	Field 7	- color
	Field 8	- number of nodes on element
Record 2:	FORMAT(8I10)	
	Fields 1–n	- node labels defining element

Records 1 and 2 are repeated for each element in the model.

The element label (Record 1, Field 1) is strictly a numeric counter, starting at one. Record 1, Field 2 defines the element type. Although many

element types exist, FEMATS only uses one—a solid linear tetrahedron—element type 111. Record 1, Fields 3 and 4 are not used by FEMATS, and should be set to 1. The material properties (Record 1, Fields 5 and 6), however, are used by FEMATS to distinguish between the different materials in each mesh volume. Generally, Record 1, Field 5 is set to 1, and Record 1, Field 6 indicates the material property label discussed in Section 2.3 and Appendix A of the FEMATS User's Manual. Record 1, Field 7 indicates the element color, and is not used in FEMATS. Record 1, Field 8 is self-explanatory.

Record 2 contains enough entries, and thus, enough lines, to list all the nodes on a particular element. For FEMATS, since only solid linear tetrahedra are used, there will only be 4 nodes, and thus only one line per element.

Example:

```
   -1
  780
        1       111         1         2         1         1         7         4
        9        22        11         2
        2       111         1         2         1         1         7         4
       22       293        92         9
        3       111         1         2         1         1         7         4
       22        11       293         9

etc....

    18222       111         1         2         1         2         7         4
     3294      3249      1363      3250
    18223       111         1         2         1         2         7         4
     3291      1341      3249      1340
    18224       111         1         2         1         2         7         4
     3291      3249      1341      3294
   -1
```

In this example, there are 18,224 elements, all solid linear tetrahedra. They all have the same material properties, and are thus all in the same medium. As would be expected, they all have 4 nodes, and their color is 7 (green).

Dataset 781: Nodes - Double precision

Record 1:	FORMAT(4I10)	
	Field 1	- node label
	Field 2	- definition coordinate system number
	Field 3	- displacement coordinate system number
	Field 4	- color
Record 2:	FORMAT(1P3D25.16)	
	Fields 1–3	- 3-dimensional coordinates of node in the definition system

Records 1 and 2 are repeated for each node in the model.

The node label (Record 1, Field 1) is strictly a numeric counter, starting at one. Record 1, Field 2 indicates the coordinate system used to define the node locations. While the three standard coordinate systems are available—Cartesian, cylindrical, and spherical (numbers 0, 1, and 2, respectively), FEMATS only uses Cartesian. Hence, both Record 1, Field 2 and Record 1, Field 3 should be zero. Record 1, Field 4 indicates the node color, and is not used in FEMATS.

Fields 1–3 of Record 2 are the 3-dimensional coordinates of the node locations, in double precision format. In general, these three values are expressed in terms of the appropriate coordinate variables, i.e. for spherical coordinates, the coordinate variables are (r, θ, ϕ), while for cylindrical coordinates, the appropriate variables are (r, θ, z). Since FEMATS only uses the Cartesian system, the appropriate variables are (x, y, z).

Example:

```
   -1
  781
         1         0         0        11
 -1.1250000447034835E+00   0.0000000000000000E+00   1.4375000260770320E+00
         2         0         0        11
  0.0000000000000000E+00  -1.1250000447034835E+00   1.4375000260770320E+00
         3         0         0        11
  0.0000000000000000E+00   0.0000000000000000E+00   1.4375000260770320E+00

etc....

      3718         0         0        11
 -1.1962018850819641E-01  -7.3728312687791492E-02   9.1661646783763676E-02
      3719         0         0        11
 -5.5161825599575895E-02  -6.6190053659418503E-02   9.6638066869683866E-02
      3720         0         0        11
 -8.5961703548581673E-02  -1.1321300839107060E-01   5.2343589137973839E-02
```

```
    -1
```

In this example, there are 3720 nodes, all defined in the Cartesian coordinate system, and their color is 11 (red).

NOTE: Although these datasets have been presented in numerical order by their number, it should be realized that dataset 781 (Nodes) must precede dataset 780 (Elements) in the universal file, as dataset 780 references information from dataset 781. Also, since the groups depend on both the nodes and elements, dataset 752 (Groups) should appear after both dataset 780 and 781.

Following is an example universal file generated in I-DEAS for use with FEMATS. Because I-DEAS requires other information in excess of that necessary for FEMATS, there are several datasets included in the following universal file which are not described above. As these datasets are not necessary for FEMATS, and are only necessary if the user is attempting to write a full conversion routine from some other mesh generation package to I-DEAS, or trying to import their mesh into I-DEAS, they will not be covered in this manual, and the user is referred to both the I-DEAS Core Utilities Manual and the I-DEAS PEARL Manual for more information.

Sample I-DEAS Universal File:

```
    -1
   151
temp1
temp1
SDRC  I-DEAS VI.i:  Monitor
04-AUG-93   10:55:54         6         0
04-AUG-93   14:00:32
SDRC  I-DEAS VI.i:  FE_Modeling_&_Analysis
04-AUG-93   14:01:59
    -1
    -1
   164
         5Modified SI (mm)             2
  1.00000000000000000E+03  1.00000000000000000E+03  1.00000000000000000E+00
  2.73149999999999960E+02
    -1
    -1
   800
         1
WORKING_SET1
    -1
    -1
   770
         1         0
```

```
MAIN
    -1
    -1
   771
         1         1         1
FE MODEL1
    -1
    -1
   781
         1         0         0        11
   1.5000000712461767E-01  -2.7105054312137611E-17   0.0000000000000000E+00
         2         0         0        11
   0.0000000000000000E+00   0.0000000000000000E+00   0.0000000000000000E+00
         3         0         0        11
   2.1175823681357506E-17   1.5000000712461767E-01   0.0000000000000000E+00

etc....

   3.7500877506659887E-01   1.4697603711765345E-01   8.7404428380300838E-02
       451         0         0        11
   3.7094838851549648E-01   6.9079955860908915E-02   8.3694518214300438E-02
       452         0         0        11
   3.9089404906600184E-01   9.8862596006164716E-02   4.3313292570483312E-02
    -1
    -1
   780
         1       111         1         2         1         1         7         4
         9        22        11         2
         2       111         1         2         1         1         7         4
        22       293        92         9
         3       111         1         2         1         1         7         4
        22        11       293         9

etc...

      1762       111         1         2         1         1         7         4
        65         8        15       296
      1763       111         1         2         1         1         7         4
        92         8        65       296
      1764       111         1         2         1         1         7         4
        91         8        92       296
    -1
    -1
   734
         1        17         1 1.000000E+00         0 1.000000E+00
         2 0.000000E+00 0.000000E+00
 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 1.000000E+00 0.000000E+00
```

```
 0.000000E+00 0.000000E+00 0.000000E+00-1.000000E+00-1.000000E+00 1.000000E+00
 1.000000E+00 0.000000E+00 0.000000E+00 1.000000E+00 1.000000E+00 0.000000E+00
 1.000000E+00 0.000000E+00 1.000000E+00 1.000000E+00 1.000000E+00 1.000000E+00
-1.000000E+00-1.000000E+00-1.000000E+00-1.000000E+00 1.250000E+00 1.250000E+00
 0.000000E+00 1.500000E+00
    -1
    -1
   752
         1         0         0         0         0        14
C1
         7         1         7         2         7         3         7         4
         7         8         7         9         7        10         7        11
         7        12         7        13         7        14         7        15
         7        64         7        65
         2         0         0         0         0       146
A1
         7         5         7         6         7         7         7        27
         7        28         7        29         7        30         7        31
         7        32         7        33         7        34         7        35

etc....

         7       264         7       265         7       266         7       267
         7       268         7       269         7       270         7       271
         7       272         7       273
         3         0         0         0         0        14
O1
         7         1         7         2         7         3         7         4
         7         8         7         9         7        10         7        11
         7        12         7        13         7        14         7        15
         7        64         7        65
    -1
```

E Benchmark Test Case Manual

Most of the benchmark geometries specified by the Electromagnetics Code Consortium (EMCC) have been run with FEMATS. These benchmark cases demonstrate the code's validity and allow the user to verify the correct operation of the code on his/her machine. The following test cases are included here:

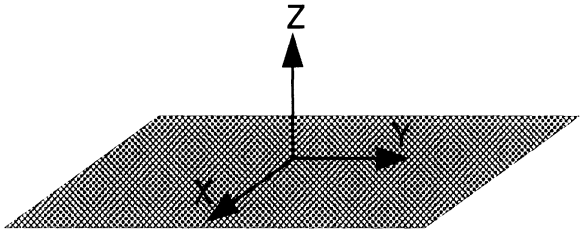
1. EMCC Target #4, Rectangular PEC Plate; $\phi = 0°$, vv-pol
2. EMCC Target #4, Rectangular PEC Plate; $\phi = 90°$, vv-pol
3. EMCC Target #4, Rectangular PEC Plate; $\phi = 90°$, hh-pol
4. EMCC Target #6, Foam Cylinder, Co-Linear Wires
5. EMCC Target #6, Foam Cylinder, Wires in Echelon
6. EMCC Target #8, Glass Plate
7. $1.0\lambda \times 1.0\lambda \times 1.5\lambda$ Rectangular Inlet, Rectangular Mesh Termination, vv-pol
8. $1.0\lambda \times 1.0\lambda \times 1.5\lambda$ Rectangular Inlet, Rectangular Mesh Termination, hh-pol
9. $1.0\lambda \times 1.0\lambda \times 1.5\lambda$ Spherical Inlet, Rectangular Mesh Termination, vv-pol
10. $1.0\lambda \times 1.0\lambda \times 1.5\lambda$ Spherical Inlet, Rectangular Mesh Termination, hh-pol
11. 1.875λ height $\times 1.25\lambda$ dia Circular Inlet, Cylindrical Mesh Termination, vv-pol
12. 1.875λ height $\times 1.25\lambda$ dia Circular Inlet, Cylindrical Mesh Termination, hh-pol

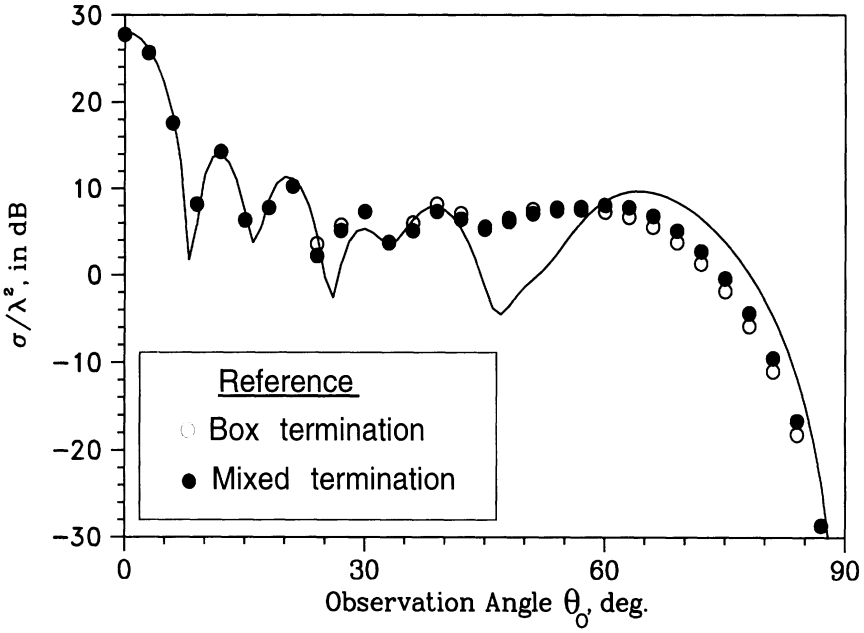
Plots of each case are included in order following this introduction.

The original universal file and the preprocessor input files, along with the file containing the plot data, are included in the appropriate subdirectory in the benchmark subdirectory (`~femats/bench`) of the FEMATS installation tree, given at the bottom of each plot.

Rectangular PEC Plate Benchmark Target

3.5 λ x 2.0 λ Plate, ϕ = 0°
VV-Polarization



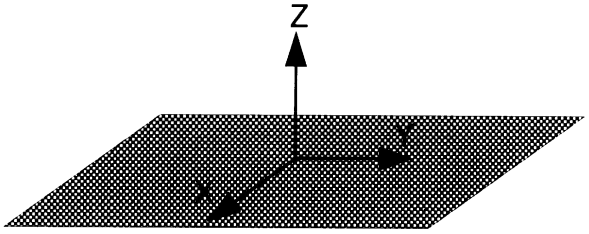


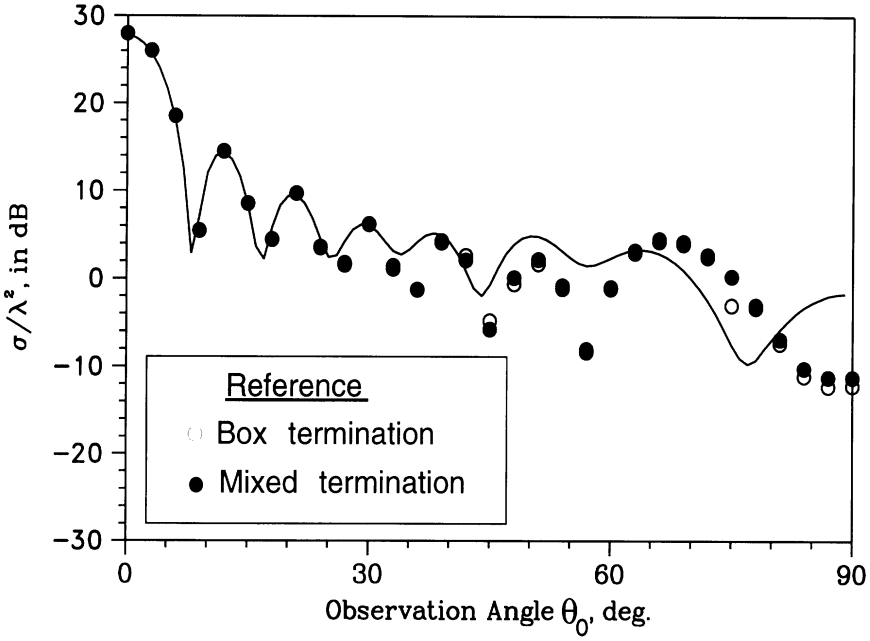
Backscatter Pattern of Metal Plate

This file (rect_plate_p0_vv.ps) and its associated input, output, and runtime data in ~femats/bench/rect_plate.

Rectangular PEC Plate Benchmark Target

3.5 λ x 2.0 λ Plate, ϕ = 0°
HH-Polarization



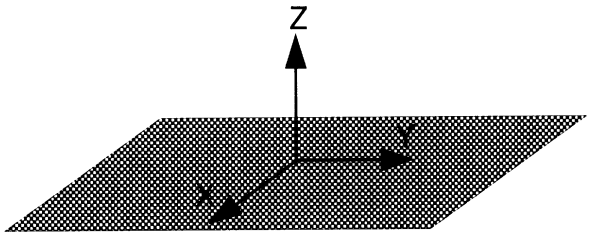


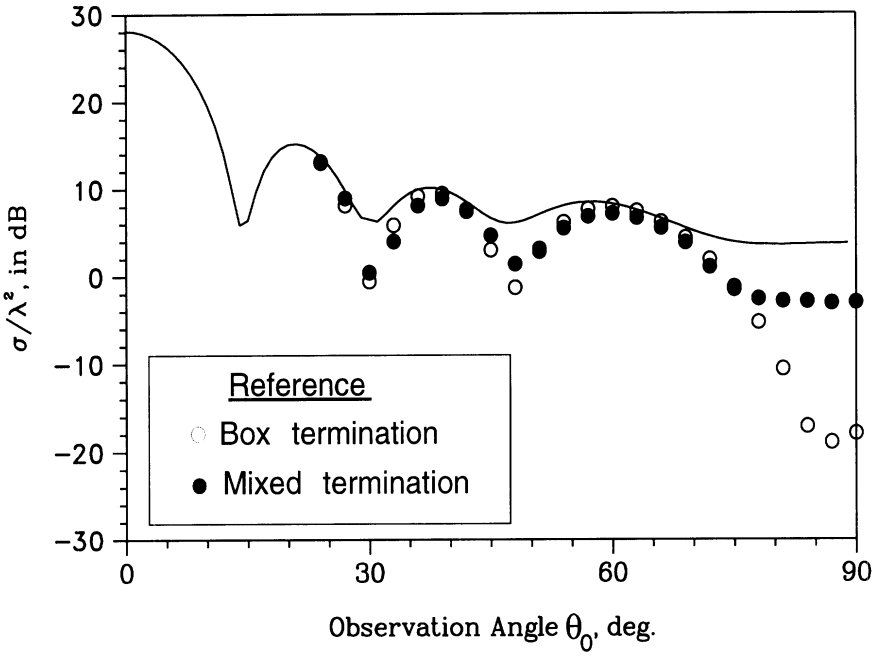
Backscatter Pattern of Metal Plate

This file (rect_plate_p0_hh.ps) and its associated input, output, and runtime data in ~femats/bench/rect_plate.

Rectangular PEC Plate Benchmark Target

3.5 λ x 2.0 λ Plate, ϕ = 90°
HH-Polarization

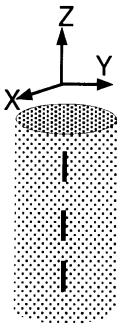


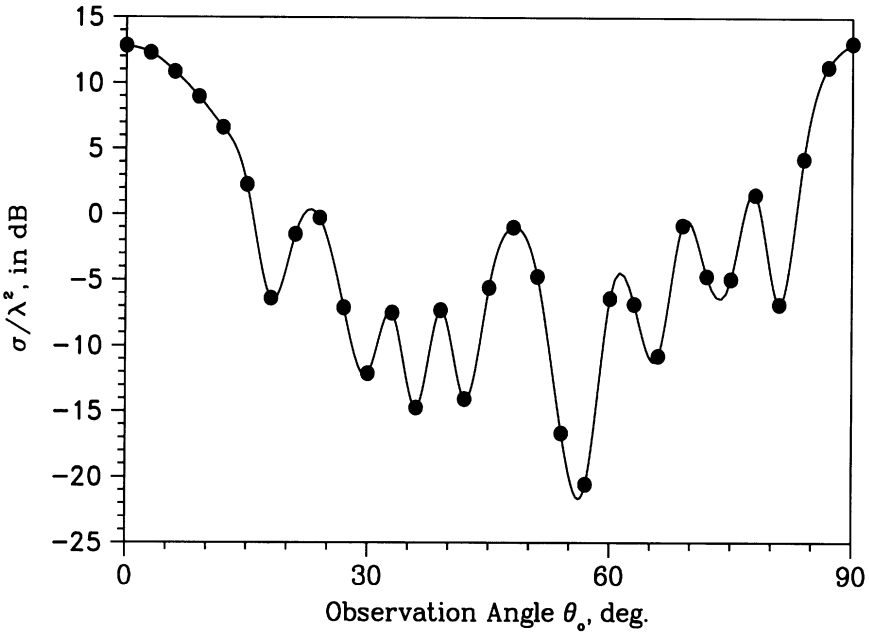


Backscatter Pattern of Metal Plate

This file (rect_plate_p90_hh.ps) and its associated input, output, and runtime data in ~femats/bench/rect_plate.

Foam Cylinder Benchmark Target, Co-Linear Wires





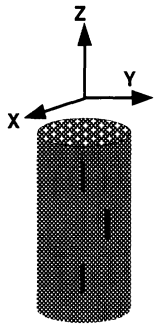
Backscatter Pattern of Foam Cylinder

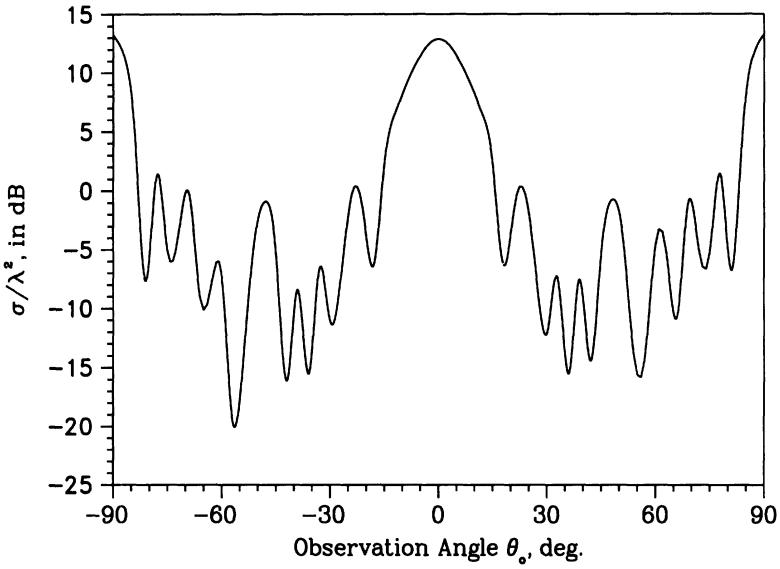
Cylinder Radius = 1.0 λ, Cylinder Height = 3.5 λ.
Three Wires of Length 0.5 λ, Spaced 0.5 λ Apart.

Cylindrical Mesh Termination Placed 0.45 λ Away.

This file (foam_cyl_colinear.ps) and its associated input, output, and runtime data in ~femats/bench/foam_cyl.

Foam Cylinder Benchmark Target, Wires in Echelon





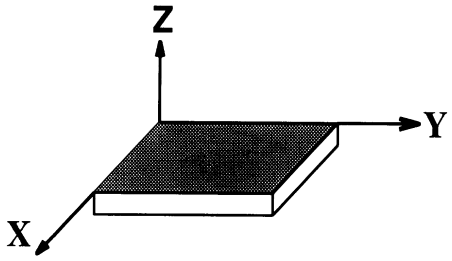
Backscatter Pattern of Foam Cylinder

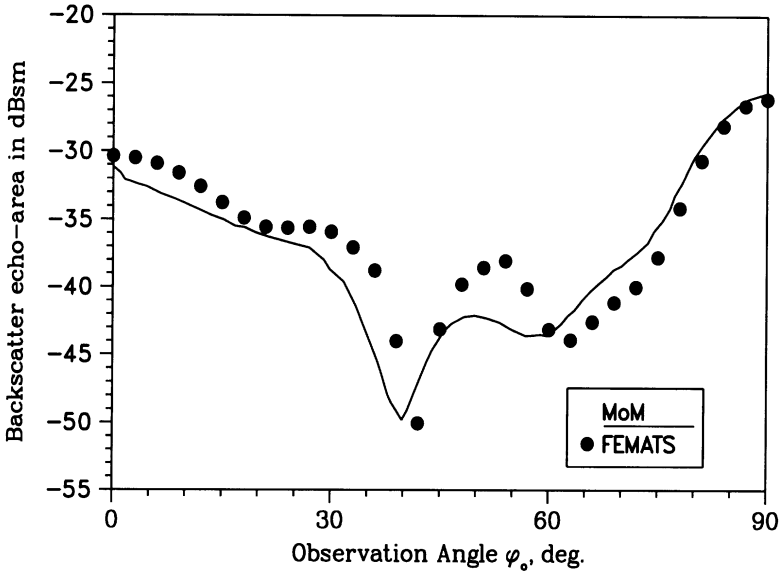
Cylinder Radius = 1.0 λ, Cylinder Height = 3.5 λ.
Three Wires of Length 0.5 λ, Spaced 0.5 λ Apart.

Cylindrical Mesh Termination Placed 0.45 λ Away.

This file (foam_cyl_echelon.ps) and its associated input, output, and runtime data in ~femats/bench/foam_cyl.

Glass Plate Benchmark Target





Backscatter pattern of glass plate

$\theta = 80^{o}$ conical cut ; $\varepsilon_r = 3 - j0.09$

Plate Dim.: 1.75 λ x 1.0 λ x 0.125 λ

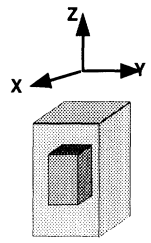
Mixed Mesh Termination Boundary

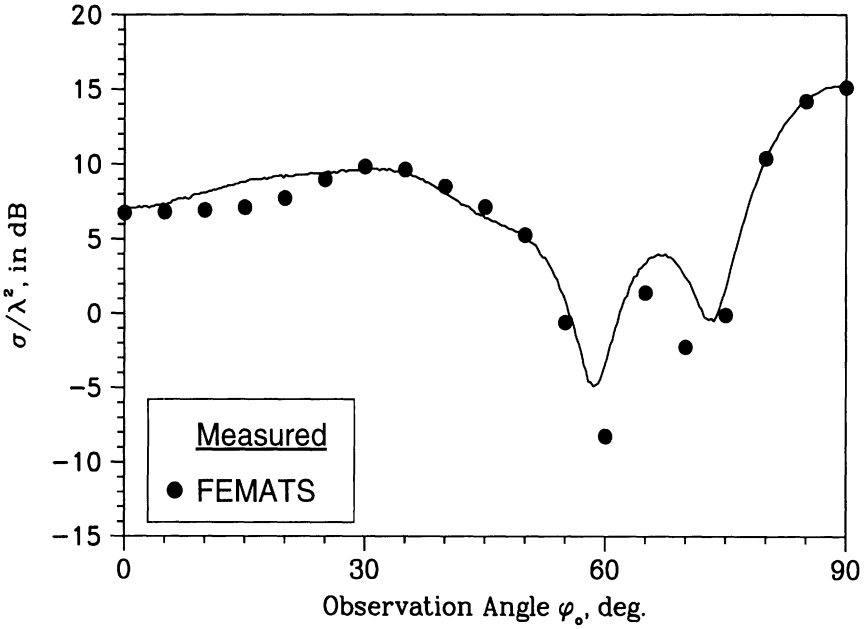
This file (glass_plate.ps) and its associated input, output, and runtime data in ~femats/bench/glass_plate.

Rectangular Inlet Benchmark Target

VV-Polarization

Rectangular Mesh Termination Boundary





Backscatter Pattern of Rectangular Inlet

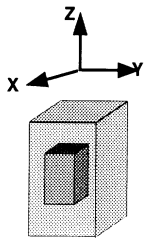
Inlet Dim.: 1.0 λ x 1.0 λ x 1.5 λ

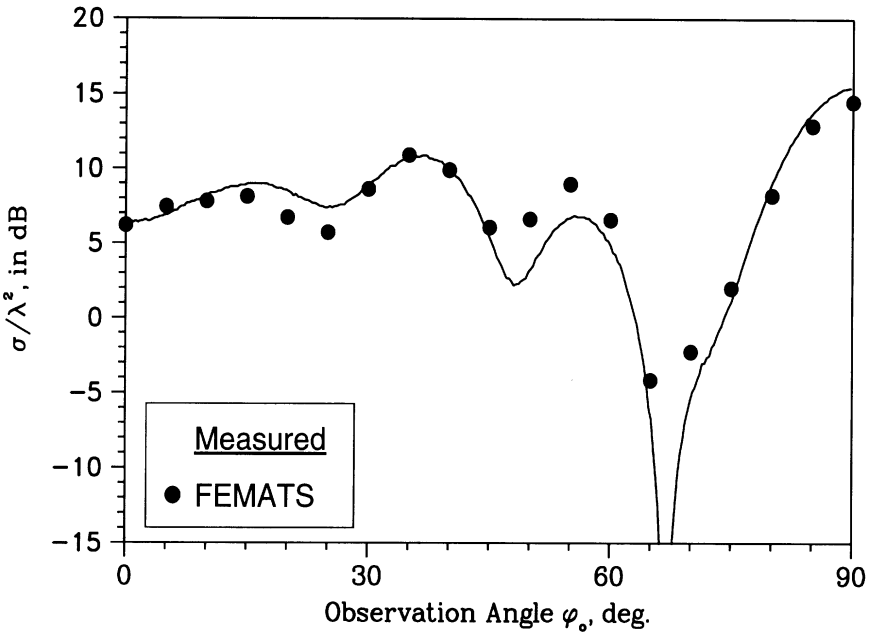
This file (rect_inlet_vv_r.ps) and its associated input, output, and runtime data in ~femats/bench/rect_inlet.

Rectangular Inlet Benchmark Target

HH-Polarization

Rectangular Mesh Termination Boundary





Backscatter Pattern of Rectangular Inlet

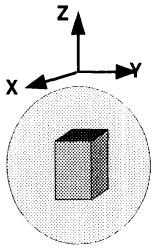
Inlet Dim.: 1.0 λ x 1.0 λ x 1.5 λ

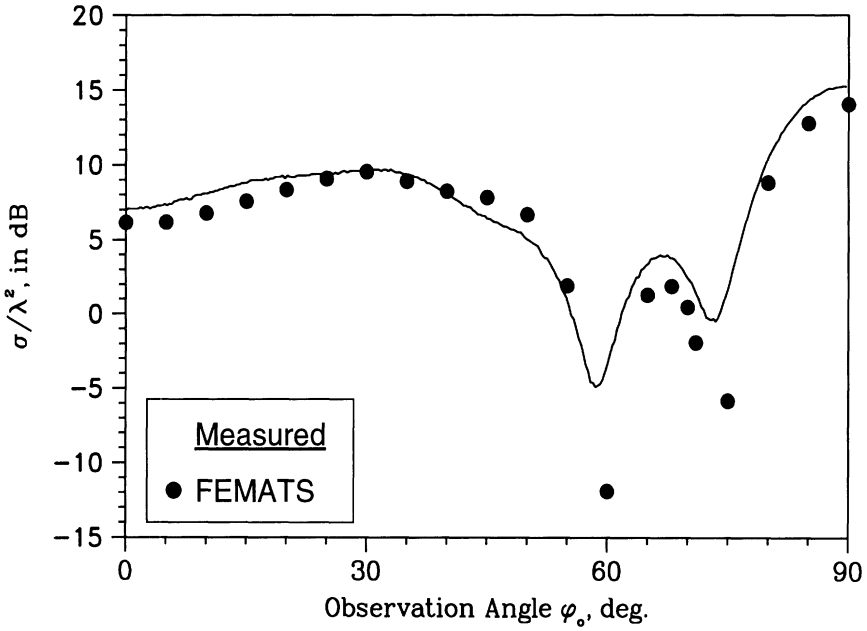
This file (rect_inlet_hh_r.ps) and its associated input, output, and runtime data in ~femats/bench/rect_inlet.

Rectangular Inlet Benchmark Target

VV-Polarization

Spherical Mesh Termination Boundary





Backscatter Pattern of Rectangular Inlet

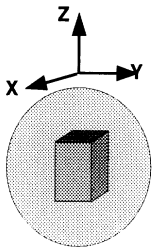
Inlet Dim.: 1.0 λ x 1.0 λ x 1.5 λ

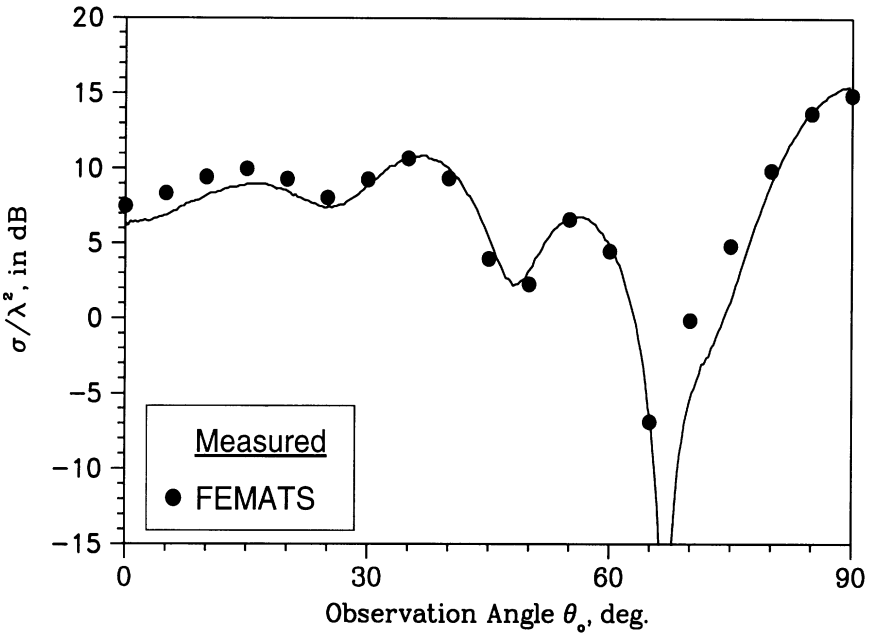
This file (rect_inlet_vv_s.ps) and its associated input, output, and runtime data in ~femats/bench/rect_inlet.

Rectangular Inlet Benchmark Target

HH-Polarization

Spherical Mesh Termination Boundary





Backscatter Pattern of Rectangular Inlet

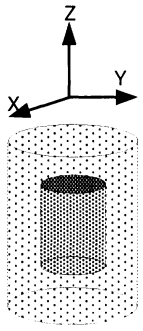
Inlet Dim.: 1.0 λ x 1.0 λ x 1.5 λ

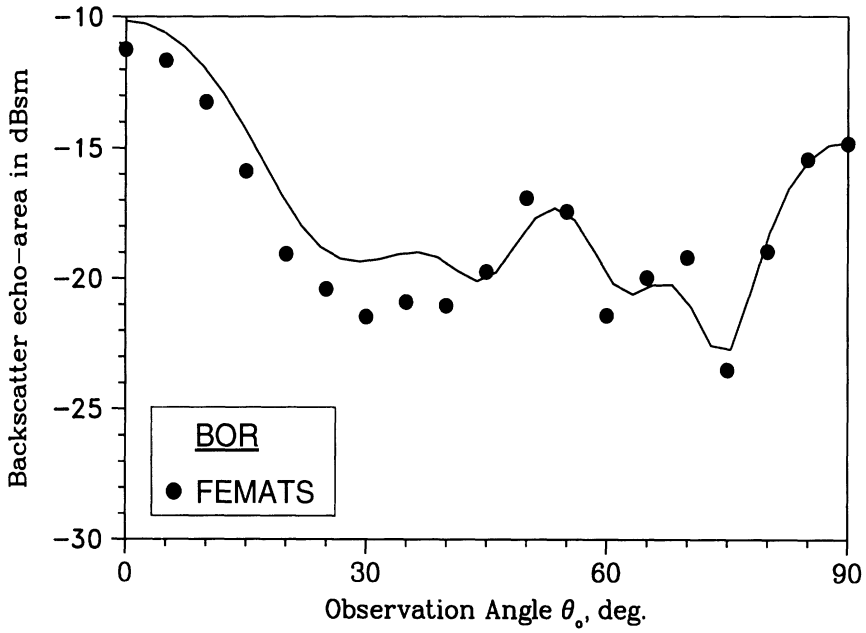
This file (rect_inlet_hh_s.ps) and its associated input, output, and runtime data in ~femats/bench/rect_inlet.

Circular Inlet Benchmark Target

VV-Polarization

Cylindrical Mesh Termination Boundary





Backscatter Pattern of Circular Inlet

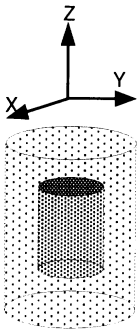
Inlet Dim.: $h = 1.875\ \lambda$, $dia = 1.25\ \lambda$

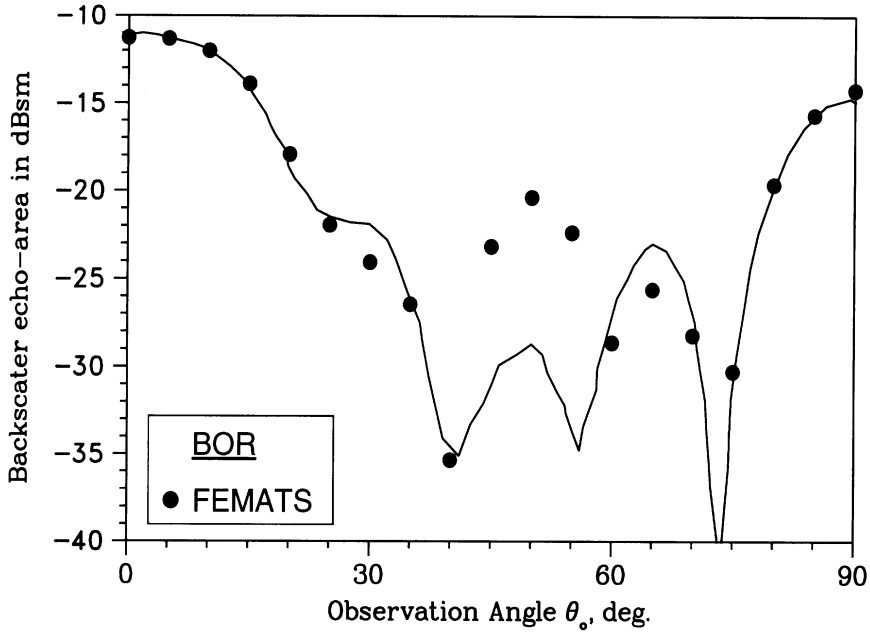
This file (circ_inlet_vv_c.ps) and its associated input, output, and runtime data in ~femats/bench/circ_inlet.

Circular Inlet Benchmark Target

HH-Polarization

Cylindrical Mesh Termination Boundary





Backscatter Pattern of Circular Inlet

Inlet Dim.: $h = 1.875\ \lambda$, $dia = 1.25\ \lambda$

This file (circ_inlet_hh_c.ps) and its associated input, output, and runtime data in ~femats/bench/circ_inlet.

F Installation

FEMATS is designed to run on multiple computing platforms to best utilize various machine capabilities. The preprocessing programs can be readily run on a UNIX workstation, while the main processing might be done on a different machine, e.g. a CRAY supercomputer, if the memory and speed requirements demand so for a given geometry. Hence, there are two sets of source code included on the tape, one for the preprocessor and one for the main processor, along with two installation procedures. Also included is the small test case used to generate the example run in Appendix C, starting with the I-DEAS universal file `small_plate.unv` and ending with the FEMATS output files.

Installation instructions

1. Place the distribution tape in the tape drive. If the drive is not the default system drive, you will need to find out what device it is.

2. Retrieve the appropriate files from the tape. For example, if you wanted to install FEMATS in your home directory, you would say (assuming the tape drive being used is the system default tape drive):

   ```
   cd
   tar xv README femats.bench.tar.Z femats.doc.tar.Z
     femats.{arch}.tar femats.preproc.tar
   ```

   where `{arch}` is one of the supported architectures: `cray`, `hp`, `ksr`, or `paragon`.

   This places the named files in your home directory.

   There are eight files on the tape:

   ```
   README
   femats.bench.tar.Z
   femats.cray.tar
   femats.doc.tar.Z
   femats.hp.tar
   femats.ksr.tar
   femats.paragon.tar
   femats.preproc.tar
   ```

   `README` is a short description of the files on the tape.

   `femats.bench.tar.Z` is a compressed tar file containing the benchmark runs mentioned in Appendix E of this manual.

   `femats.cray.tar` is a tar file containing the source code and test files for the CRAY version of FEMATS.

`femats.doc.tar.Z` is a compressed tar file containing the documentation for all of FEMATS (LaTeXand .ps formats).

`femats.hp.tar` is a tar file containing the source code and test files for the HP workstation version of FEMATS.

`femats.ksr.tar` is a tar file containing the source code and test files for the KSR version of FEMATS.

`femats.paragon.tar` is a tar file containing the source code and test files for the Intel Paragon version of FEMATS.

`femats.preproc.tar` is a tar file containing the source code and test files for the preprocessing portion of FEMATS.

3. If necessary, `ftp` the supercomputer portion of FEMATS to the supercomputer, putting it in the appropriate directory.

4. If `femats.preproc.tar` is not in the directory where you want to install FEMATS, then put it there. **Note**: When the files are 'untar'd, a directory named `femats` will be created, and the appropriate files and subdirectories placed in it. (See Appendix G for a full file listing.)

5. Uncompress `femats.preproc.tar.Z`: type

   ```
   uncompress femats.preproc.tar.Z
   ```

6. Untar `femats.preproc.tar`: type

   ```
   tar xvf femats.preproc.tar
   ```

7. Change directories to `femats/src/preproc`, and type `make install`. This will compile the preprocessors and automatically place them in the `femats/bin` directory. If you wish, you may then type `make clean` to remove the `.o` files.

8. Follow the same steps for the chosen workstation or supercomputer source code files, starting with step 4, and changing `femats.preproc` to `femats.{arch}` in all cases.

9. For both the workstation and the supercomputer, FEMATS assumes that it will be run from the user's current working directory. To accomodate this, the `femats/bin` directory must be included in the search path. To do this, modify the appropriate startup file (the following assumes the C Shell, i.e. the `.cshrc` file):

   `set path=($path` *femats_dir*`/bin)`

   where *femats_dir* is the full path of the `femats` directory. For example,

   ```
   set path=($path /l/usr/femats/bin)
   ```

Then type `source .cshrc` to ensure that the new path takes effect. If any difficulties are encountered, ask your system/site administrator.

10. If any problems occur, don't hesitate to look in the scripts—they are quite simple, and there may be some machine or OS version dependencies that were missed...

G Directory Listing of Full Distribution

```
femats:
total 9
drwxrwxrwx  7 mwnurnbe      512 Oct  5 10:36 .
drwx------  4 mwnurnbe      512 Oct  5 15:24 ..
-rw-r--r--  1 mwnurnbe     1308 Oct  3 16:10 README
drwxrwxrwx  7 mwnurnbe      512 Oct  5 11:56 bench
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:07 bin
drwxrwxrwx  2 mwnurnbe     1024 Oct  5 10:46 doc
drwxrwxrwx  6 mwnurnbe      512 Oct  5 10:41 src
drwxrwxrwx  4 mwnurnbe      512 Oct  5 09:10 test

femats/bench:
total 7
drwxrwxrwx  7 mwnurnbe      512 Oct  5 11:56 .
drwxrwxrwx  7 mwnurnbe      512 Oct  5 10:36 ..
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:46 circ_inlet
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:47 foam_cyl
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:51 glass_plate
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:52 rect_inlet
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:57 rect_plate

femats/bench/circ_inlet:
total 388
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:46 .
drwxrwxrwx  7 mwnurnbe      512 Oct  5 11:56 ..
-rw-r--r--  1 mwnurnbe   191099 May  5 12:37 circ_inlet_hh_c.ps
-rw-r--r--  1 mwnurnbe   180165 May  5 12:32 circ_inlet_vv_c.ps
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:46 cyl_term
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:46 rect_term

femats/bench/circ_inlet/cyl_term:
total 12285
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:46 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:46 ..
-rw-r--r--  1 mwnurnbe   660392 May  8 21:29 cntr
-rw-r--r--  1 mwnurnbe  2518518 May  8 21:29 cyl_inlt_c.unv
-rw-r--r--  1 mwnurnbe  2019142 May  8 21:29 edgy
-rw-r--r--  1 mwnurnbe       15 May  8 21:29 eg
-rw-r--r--  1 mwnurnbe  5603435 May  8 21:29 eglob
-rw-r--r--  1 mwnurnbe   697562 May  8 21:29 enoddy
-rw-r--r--  1 mwnurnbe   993240 May  8 21:29 esurfed
-rw-r--r--  1 mwnurnbe      166 May  8 21:29 ip_femats
```

```
-rw-r--r--  1 mwnurnbe       64 May  8 21:29 ip_mesh

femats/bench/circ_inlet/rect_term:
total 15949
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:46 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:46 ..
-rw-r--r--  1 mwnurnbe   849216 May  8 21:29 cntr
-rw-r--r--  1 mwnurnbe  3231558 May  8 21:29 cyl_inlet_b.unv
-rw-r--r--  1 mwnurnbe  2637600 May  8 21:29 edgy
-rw-r--r--  1 mwnurnbe       15 May  8 21:29 eg
-rw-r--r--  1 mwnurnbe  7372490 May  8 21:29 eglob
-rw-r--r--  1 mwnurnbe   880955 May  8 21:29 enoddy
-rw-r--r--  1 mwnurnbe  1294056 May  8 21:29 esurfed
-rw-r--r--  1 mwnurnbe      242 May  8 21:29 ip_femats
-rw-r--r--  1 mwnurnbe       65 May  8 21:29 ip_mesh

femats/bench/foam_cyl:
total 412
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:47 .
drwxrwxrwx  7 mwnurnbe      512 Oct  5 11:56 ..
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:47 colinear
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:51 echelon
-rw-r--r--  1 mwnurnbe   204989 May  4 19:05 foam_cyl_colinear.ps
-rw-r--r--  1 mwnurnbe   185065 May  4 19:00 foam_cyl_echelon.ps

femats/bench/foam_cyl/colinear:
total 33069
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:47 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:47 ..
-rw-r--r--  1 mwnurnbe  1738460 May  8 21:23 cntr
-rw-r--r--  1 mwnurnbe  5537161 May  8 21:23 edgy
-rw-r--r--  1 mwnurnbe       15 May  8 21:23 eg
-rw-r--r--  1 mwnurnbe 15792859 May  8 21:23 eglob
-rw-r--r--  1 mwnurnbe  1767211 May  8 21:23 enoddy
-rw-r--r--  1 mwnurnbe  2329948 May  8 21:23 esurfed
-rw-r--r--  1 mwnurnbe  6625155 May  8 21:23 f_cyl_.45.unv
-rw-r--r--  1 mwnurnbe      160 May  8 21:23 ip_femats
-rw-r--r--  1 mwnurnbe       66 May  8 21:23 ip_mesh

femats/bench/foam_cyl/echelon:
total 33197
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:51 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:47 ..
-rw-r--r--  1 mwnurnbe  1744840 May  8 21:24 cntr
-rw-r--r--  1 mwnurnbe  5556858 May  8 21:24 edgy
```

```
-rw-r--r--  1 mwnurnbe       15 May  8 21:24 eg
-rw-r--r--  1 mwnurnbe 15855216 May  8 21:24 eglob
-rw-r--r--  1 mwnurnbe  1769881 May  8 21:24 enoddy
-rw-r--r--  1 mwnurnbe  2338557 May  8 21:24 esurfed
-rw-r--r--  1 mwnurnbe  6647785 May  8 21:24 fom_off.unv
-rw-r--r--  1 mwnurnbe      163 May  8 21:24 ip_femats
-rw-r--r--  1 mwnurnbe       68 May  8 21:24 ip_mesh

femats/bench/glass_plate:
total 11741
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:51 .
drwxrwxrwx  7 mwnurnbe      512 Oct  5 11:56 ..
-rw-r--r--  1 mwnurnbe   600856 May  8 21:35 cntr
-rw-r--r--  1 mwnurnbe  1815397 May  8 21:35 edgy
-rw-r--r--  1 mwnurnbe       15 May  8 21:35 eg
-rw-r--r--  1 mwnurnbe  5105896 May  8 21:35 eglob
-rw-r--r--  1 mwnurnbe   641234 May  8 21:35 enoddy
-rw-r--r--  1 mwnurnbe  1226841 May  8 21:35 esurfed
-rw-r--r--  1 mwnurnbe   217865 May  5 10:39 glass_plate.ps
-rw-r--r--  1 mwnurnbe  2326058 May  8 21:35 gplt_rcs.unv
-rw-r--r--  1 mwnurnbe      332 May  8 21:35 ip_femats
-rw-r--r--  1 mwnurnbe       59 May  8 21:35 ip_mesh

femats/bench/rect_inlet:
total 676
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:52 .
drwxrwxrwx  7 mwnurnbe      512 Oct  5 11:56 ..
-rw-r--r--  1 mwnurnbe   157459 May  5 11:21 rect_inlet_hh_r.ps
-rw-r--r--  1 mwnurnbe   157728 May  5 11:18 rect_inlet_hh_s.ps
-rw-r--r--  1 mwnurnbe   157892 May  5 11:20 rect_inlet_vv_r.ps
-rw-r--r--  1 mwnurnbe   161102 May  5 11:15 rect_inlet_vv_s.ps
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:52 rect_term
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:52 sphere_term

femats/bench/rect_inlet/rect_term:
total 11661
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:52 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:52 ..
-rw-r--r--  1 mwnurnbe   625116 May  8 21:33 cntr
-rw-r--r--  1 mwnurnbe  1907209 May  8 21:33 edgy
-rw-r--r--  1 mwnurnbe       15 May  8 21:33 eg
-rw-r--r--  1 mwnurnbe  5269840 May  8 21:33 eglob
-rw-r--r--  1 mwnurnbe   642125 May  8 21:33 enoddy
-rw-r--r--  1 mwnurnbe  1038069 May  8 21:33 esurfed
-rw-r--r--  1 mwnurnbe      247 May  8 21:33 ip_femats
```

```
-rw-r--r--  1 mwnurnbe       62 May  8 21:33 ip_mesh
-rw-r--r--  1 mwnurnbe  2383439 May  8 21:33 rect.075.unv

femats/bench/rect_inlet/sphere_term:
total 17308
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:52 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:52 ..
-rw-r--r--  1 mwnurnbe  1341128 May  8 21:33 cntr
-rw-r--r--  1 mwnurnbe  3444638 May  8 21:33 edgy
-rw-r--r--  1 mwnurnbe       13 May  8 21:33 eg
-rw-r--r--  1 mwnurnbe  9319037 May  8 21:33 eglob
-rw-r--r--  1 mwnurnbe  2256561 May  8 21:33 enoddy
-rw-r--r--  1 mwnurnbe  1302251 May  8 21:33 esurfed
-rw-r--r--  1 mwnurnbe       98 May  8 21:33 ip_femats

femats/bench/rect_plate:
total 668
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:57 .
drwxrwxrwx  7 mwnurnbe      512 Oct  5 11:56 ..
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:57 box_term
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:58 mixed_term
-rw-r--r--  1 mwnurnbe   222419 May  5 09:45 rect_plate_p0_hh.ps
-rw-r--r--  1 mwnurnbe   218655 May  5 09:28 rect_plate_p0_vv.ps
-rw-r--r--  1 mwnurnbe   197611 May  5 09:55 rect_plate_p90_hh.ps

femats/bench/rect_plate/box_term:
total 10213
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:57 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:57 ..
-rwxr-xr-x  1 mwnurnbe  2105651 Oct  5 11:52 c_plt_box.45.unv
-rwxr-xr-x  1 mwnurnbe   551348 Oct  5 11:52 cntr
-rwxr-xr-x  1 mwnurnbe  1670445 Oct  5 11:52 edgy
-rwxr-xr-x  1 mwnurnbe       15 Oct  5 11:52 eg
-rwxr-xr-x  1 mwnurnbe  4513976 Oct  5 11:52 eglob
-rwxr-xr-x  1 mwnurnbe   578889 Oct  5 11:52 enoddy
-rwxr-xr-x  1 mwnurnbe   955160 Oct  5 11:52 esurfed
-rwxr-xr-x  1 mwnurnbe      215 Oct  5 11:52 ip_femats
-rwxr-xr-x  1 mwnurnbe       67 Oct  5 11:52 ip_mesh

femats/bench/rect_plate/mixed_term:
total 11021
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:58 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 11:57 ..
-rwxr-xr-x  1 mwnurnbe   594252 Oct  5 11:53 cntr
-rwxr-xr-x  1 mwnurnbe  2272118 Oct  5 11:53 cplt_mxd.unv
```

```
-rwxr-xr-x  1 mwnurnbe  1813873 Oct  5 11:53 edgy
-rwxr-xr-x  1 mwnurnbe       15 Oct  5 11:53 eg
-rwxr-xr-x  1 mwnurnbe  4921745 Oct  5 11:53 eglob
-rwxr-xr-x  1 mwnurnbe   628853 Oct  5 11:53 enoddy
-rwxr-xr-x  1 mwnurnbe   976573 Oct  5 11:53 esurfed
-rwxr-xr-x  1 mwnurnbe      324 Oct  5 11:53 ip_femats
-rwxr-xr-x  1 mwnurnbe       63 Oct  5 11:53 ip_mesh

femats/bin:
total 27
drwxrwxrwx  2 mwnurnbe      512 Oct  5 11:07 .
drwxrwxrwx  7 mwnurnbe      512 Oct  5 10:36 ..
-rwxr-xr-x  1 mwnurnbe     4547 Oct  5 09:50 femats.cray
-rwxr-xr-x  1 mwnurnbe     5319 Oct  5 11:07 femats.ksr
-rwxr-xr-x  1 mwnurnbe     5354 Oct  5 10:30 femats.paragon
-rwxr-xr-x  1 mwnurnbe     7614 Oct  5 08:47 femats.preproc

femats/doc:
total 5010
drwxrwxrwx  2 mwnurnbe     1024 Oct  5 10:46 .
drwxrwxrwx  7 mwnurnbe      512 Oct  5 10:36 ..
-rw-r--r--  1 mwnurnbe   191099 Sep 26 14:51 circ_inlet_hh_c.ps
-rw-r--r--  1 mwnurnbe   180165 Sep 26 14:51 circ_inlet_vv_c.ps
-rw-r--r--  1 mwnurnbe     3695 Sep 26 14:51 f.ps
-rw-r--r--  1 mwnurnbe     7584 Sep 26 14:51 flow.ps
-rw-r--r--  1 mwnurnbe   204989 Sep 26 14:51 foam_cyl_colinear.ps
-rw-r--r--  1 mwnurnbe   185065 Sep 26 14:51 foam_cyl_echelon.ps
-rw-r--r--  1 mwnurnbe   217865 Sep 26 14:51 glass_plate.ps
-rw-r--r--  1 mwnurnbe     1129 Sep 26 14:51 macros.tex
-rw-r--r--  1 mwnurnbe     2921 Sep 26 14:51 manual.aux
-rw-r--r--  1 mwnurnbe    80888 Sep 26 14:51 manual.dvi
-rw-r--r--  1 mwnurnbe    29460 Sep 26 14:51 manual.log
-rw-r--r--  1 mwnurnbe  2508214 Sep 26 14:51 manual.ps
-rw-r--r--  1 mwnurnbe    32231 Sep 26 14:51 manual.tex
-rw-r--r--  1 mwnurnbe     1604 Sep 26 14:51 manual.toc
-rw-r--r--  1 mwnurnbe    32562 Sep 26 14:51 manual1.tex
-rw-r--r--  1 mwnurnbe   157459 Sep 26 14:51 rect_inlet_hh_r.ps
-rw-r--r--  1 mwnurnbe   157728 Sep 26 14:51 rect_inlet_hh_s.ps
-rw-r--r--  1 mwnurnbe   157892 Sep 26 14:51 rect_inlet_vv_r.ps
-rw-r--r--  1 mwnurnbe   161102 Sep 26 14:51 rect_inlet_vv_s.ps
-rw-r--r--  1 mwnurnbe   222419 Sep 26 14:51 rect_plate_p0_hh.ps
-rw-r--r--  1 mwnurnbe   218655 Sep 26 14:51 rect_plate_p0_vv.ps
-rw-r--r--  1 mwnurnbe   197611 Sep 26 14:51 rect_plate_p90_hh.ps
-rw-r--r--  1 mwnurnbe       24 Sep 26 14:51 sngl_side.pro
```

```
femats/src:
total 6
drwxrwxrwx  6 mwnurnbe      512 Oct  5 10:41 .
drwxrwxrwx  7 mwnurnbe      512 Oct  5 10:36 ..
drwxrwxrwx  4 mwnurnbe      512 Oct  5 10:38 cray
drwxrwxrwx  4 mwnurnbe      512 Oct  5 10:39 ksr
drwxrwxrwx  5 mwnurnbe      512 Oct  5 10:40 paragon
drwxrwxrwx  2 mwnurnbe      512 Oct  4 20:49 preproc

femats/src/cray:
total 42
drwxrwxrwx  4 mwnurnbe      512 Oct  5 10:38 .
drwxrwxrwx  6 mwnurnbe      512 Oct  5 10:41 ..
-rw-r--r--  1 mwnurnbe     1789 Oct  4 16:33 Makefile
drwxrwxrwx  2 mwnurnbe      512 Oct  4 14:56 asc2bin
-rw-r--r--  1 mwnurnbe     5177 Sep 16 19:20 bicg.F
-rw-r--r--  1 mwnurnbe     1046 Sep 19 14:44 diag.F
-rw-r--r--  1 mwnurnbe    13065 Sep 26 11:04 fem.F
-rw-r--r--  1 mwnurnbe       94 Sep 26 11:03 fem_data.h
-rw-r--r--  1 mwnurnbe     1082 Sep 19 14:32 jaggd.F
-rw-r--r--  1 mwnurnbe     6332 Sep 16 19:20 k2.F
drwxrwxrwx  2 mwnurnbe      512 Oct  4 14:56 sub
-rw-r--r--  1 mwnurnbe     4160 Sep 19 14:33 xc.F

femats/src/cray/asc2bin:
total 4
drwxrwxrwx  2 mwnurnbe      512 Oct  4 14:56 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 10:38 ..
-rw-r--r--  1 mwnurnbe     1563 Sep 26 11:03 fr.f

femats/src/cray/sub:
total 74
drwxrwxrwx  2 mwnurnbe      512 Oct  4 14:56 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 10:38 ..
-rw-r--r--  1 mwnurnbe     1980 Sep 25 23:18 basis1.f
-rw-r--r--  1 mwnurnbe    10618 Sep 16 19:20 bd1.f
-rw-r--r--  1 mwnurnbe     4143 Sep 19 14:31 bi2mono.f
-rw-r--r--  1 mwnurnbe      820 Sep 16 19:20 calc1.f
-rw-r--r--  1 mwnurnbe     1507 Sep 16 19:20 comput.f
-rw-r--r--  1 mwnurnbe      928 Sep 16 19:20 crux.f
-rw-r--r--  1 mwnurnbe     1383 Sep 25 23:18 cruxd1.f
-rw-r--r--  1 mwnurnbe      908 Sep 19 14:31 f11.f
-rw-r--r--  1 mwnurnbe    13576 Sep 19 14:31 fcmb.f
-rw-r--r--  1 mwnurnbe     2600 Sep 19 14:31 finc.f
-rw-r--r--  1 mwnurnbe      801 Sep 19 14:31 heapsort.f
```

```
-rw-r--r--  1 mwnurnbe     2182 Sep 19 14:31 incc1.f
-rw-r--r--  1 mwnurnbe     2108 Sep 19 14:32 incd1.f
-rw-r--r--  1 mwnurnbe     2247 Sep 19 14:32 inci.f
-rw-r--r--  1 mwnurnbe     2311 Sep 19 14:32 incr.f
-rw-r--r--  1 mwnurnbe     1547 Sep 19 14:32 norm2d.f
-rw-r--r--  1 mwnurnbe     2458 Sep 19 14:32 norma.f
-rw-r--r--  1 mwnurnbe     1359 Sep 19 14:32 ops.f
-rw-r--r--  1 mwnurnbe      560 Sep 19 14:32 ord.f
-rw-r--r--  1 mwnurnbe      625 Sep 19 14:32 sort.f
-rw-r--r--  1 mwnurnbe      354 Sep 19 14:32 string.f
-rw-r--r--  1 mwnurnbe     1534 Sep 19 14:32 surfint1.f
-rw-r--r--  1 mwnurnbe     3371 Sep 19 14:32 value.f
-rw-r--r--  1 mwnurnbe      439 Sep 19 14:32 volume.f

femats/src/ksr:
total 58
drwxrwxrwx  4 mwnurnbe      512 Oct  5 10:39 .
drwxrwxrwx  6 mwnurnbe      512 Oct  5 10:41 ..
-rw-r--r--  1 mwnurnbe     1748 Oct  4 15:02 Makefile
drwxrwxrwx  2 mwnurnbe      512 Oct  4 13:53 asc2bin
-rw-r--r--  1 mwnurnbe    11026 Oct  3 14:37 bd1.F
-rw-r--r--  1 mwnurnbe     8441 Oct  3 14:37 bicg.F
-rw-r--r--  1 mwnurnbe     1076 Oct  3 14:38 diag.F
-rw-r--r--  1 mwnurnbe    14590 Oct  3 14:35 fem.F
-rw-r--r--  1 mwnurnbe       98 Oct  3 15:01 fem_data.h
-rw-r--r--  1 mwnurnbe     6665 Oct  3 14:37 k2.F
-rw-r--r--  1 mwnurnbe     1658 Oct  3 14:38 mult.F
drwxrwxrwx  2 mwnurnbe      512 Oct  4 13:53 sub
-rw-r--r--  1 mwnurnbe     4163 Oct  3 14:38 xc.F

femats/src/ksr/asc2bin:
total 4
drwxrwxrwx  2 mwnurnbe      512 Oct  4 13:53 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 10:39 ..
-rw-r--r--  1 mwnurnbe     1563 Oct  3 15:10 fr.f

femats/src/ksr/sub:
total 65
drwxrwxrwx  2 mwnurnbe      512 Oct  4 13:53 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 10:39 ..
-rw-r--r--  1 mwnurnbe     2298 Oct  3 14:39 basis1.F
-rw-r--r--  1 mwnurnbe     4572 Oct  3 14:41 bi2mono.F
-rw-r--r--  1 mwnurnbe      820 Oct  3 14:39 calc1.F
-rw-r--r--  1 mwnurnbe     1507 Oct  3 14:39 comput.F
-rw-r--r--  1 mwnurnbe      968 Oct  3 14:39 crux.F
```

```
-rw-r--r--  1 mwnurnbe     1351 Oct  3 14:39 cruxd1.F
-rw-r--r--  1 mwnurnbe      908 Oct  3 14:40 f11.F
-rw-r--r--  1 mwnurnbe    13589 Oct  3 14:41 fcmb.F
-rw-r--r--  1 mwnurnbe     2419 Oct  3 14:41 finc.F
-rw-r--r--  1 mwnurnbe     1210 Oct  3 14:41 hpsrt.F
-rw-r--r--  1 mwnurnbe     2182 Oct  3 14:41 incc1.F
-rw-r--r--  1 mwnurnbe     1914 Oct  3 14:41 incd1.F
-rw-r--r--  1 mwnurnbe     2247 Oct  3 14:41 inci.F
-rw-r--r--  1 mwnurnbe     2311 Oct  3 14:41 incr.F
-rw-r--r--  1 mwnurnbe     1547 Oct  3 14:42 norm2d.F
-rw-r--r--  1 mwnurnbe     2458 Oct  3 14:42 norma.F
-rw-r--r--  1 mwnurnbe     1359 Oct  3 14:42 ops.F
-rw-r--r--  1 mwnurnbe      560 Oct  3 14:42 ord.F
-rw-r--r--  1 mwnurnbe      625 Oct  3 14:43 sort.F
-rw-r--r--  1 mwnurnbe      354 Oct  3 14:43 string.F
-rw-r--r--  1 mwnurnbe     1534 Oct  3 14:43 surfint1.F
-rw-r--r--  1 mwnurnbe     4838 Oct  3 14:43 value.F
-rw-r--r--  1 mwnurnbe      410 Oct  3 14:43 volume.F

femats/src/paragon:
total 9
drwxrwxrwx  5 mwnurnbe      512 Oct  5 10:40 .
drwxrwxrwx  6 mwnurnbe      512 Oct  5 10:41 ..
-rw-r--r--  1 mwnurnbe     2106 Oct  5 10:29 Makefile
drwxrwxrwx  2 mwnurnbe      512 Oct  4 20:45 asc2bin
-rw-r--r--  1 mwnurnbe      462 Oct  4 16:02 fem_data.h
drwxrwxrwx  2 mwnurnbe      512 Oct  4 16:02 intel
drwxrwxrwx  2 mwnurnbe      512 Oct  4 16:02 intel-src

femats/src/paragon/asc2bin:
total 7
drwxrwxrwx  2 mwnurnbe      512 Oct  4 20:45 .
drwxrwxrwx  5 mwnurnbe      512 Oct  5 10:40 ..
-rw-r-----  1 mwnurnbe      462 Sep 26 11:10 fem_data.h
-rw-r-----  1 mwnurnbe     3709 Sep 26 11:10 fr.f

femats/src/paragon/intel:
total 70
drwxrwxrwx  2 mwnurnbe      512 Oct  4 16:02 .
drwxrwxrwx  5 mwnurnbe      512 Oct  5 10:40 ..
-rw-r--r--  1 mwnurnbe    10796 Sep 26 11:09 bd1.f
-rw-r--r--  1 mwnurnbe     4202 Sep 26 11:09 bicg.f
-rw-r--r--  1 mwnurnbe      592 Sep 26 11:09 broadcast_scalar.f
-rw-r--r--  1 mwnurnbe      735 Sep 26 11:09 collect_vector.f
-rw-r--r--  1 mwnurnbe      939 Sep 26 11:09 diag.f
```

```
-rw-r--r--  1 mwnurnbe     4770 Sep 26 11:09 distribute_matrix.f
-rw-r--r--  1 mwnurnbe     1126 Sep 26 11:09 distribute_vector.f
-rw-r--r--  1 mwnurnbe    15595 Sep 26 11:09 fem.f
-rw-r--r--  1 mwnurnbe     1765 Sep 26 11:09 gather_data.f
-rw-r--r--  1 mwnurnbe     6611 Sep 26 11:09 k2.f
-rw-r--r--  1 mwnurnbe     3614 Sep 26 11:09 make_fetch_send_list.f
-rw-r--r--  1 mwnurnbe     1090 Sep 26 11:10 mult.f
-rw-r--r--  1 mwnurnbe     4944 Sep 26 11:10 support.f
-rw-r--r--  1 mwnurnbe      107 Sep 26 11:10 user_seconds.f
-rw-r--r--  1 mwnurnbe     4272 Sep 26 11:10 xc.f

femats/src/paragon/intel-src:
total 62
drwxrwxrwx  2 mwnurnbe      512 Oct  4 16:02 .
drwxrwxrwx  5 mwnurnbe      512 Oct  5 10:40 ..
-rw-r--r--  1 mwnurnbe     2298 Sep 26 11:13 basis1.F
-rw-r--r--  1 mwnurnbe     4395 Sep 26 11:13 bi2mono.F
-rw-r--r--  1 mwnurnbe      820 Sep 26 11:13 calc1.F
-rw-r--r--  1 mwnurnbe     1507 Sep 26 11:13 comput.F
-rw-r--r--  1 mwnurnbe      968 Sep 26 11:13 crux.F
-rw-r--r--  1 mwnurnbe     1386 Sep 26 11:19 cruxd1.F
-rw-r--r--  1 mwnurnbe      908 Sep 26 11:13 f11.F
-rw-r--r--  1 mwnurnbe    13589 Sep 26 11:13 fcmb.F
-rw-r--r--  1 mwnurnbe     2600 Sep 26 11:13 finc.F
-rw-r--r--  1 mwnurnbe     2130 Sep 26 11:13 incc1.F
-rw-r--r--  1 mwnurnbe     2108 Sep 26 11:13 incd1.F
-rw-r--r--  1 mwnurnbe     2247 Sep 26 11:13 inci.F
-rw-r--r--  1 mwnurnbe     2311 Sep 26 11:13 incr.F
-rw-r--r--  1 mwnurnbe     1547 Sep 26 11:13 norm2d.F
-rw-r--r--  1 mwnurnbe     2458 Sep 26 11:13 norma.F
-rw-r--r--  1 mwnurnbe     1359 Sep 26 11:13 ops.F
-rw-r--r--  1 mwnurnbe      560 Sep 26 11:13 ord.F
-rw-r--r--  1 mwnurnbe      625 Sep 26 11:13 sort.F
-rw-r--r--  1 mwnurnbe      354 Sep 26 11:13 string.F
-rw-r--r--  1 mwnurnbe     1534 Sep 26 11:13 surfint1.F
-rw-r--r--  1 mwnurnbe     2351 Sep 26 11:13 value.F
-rw-r--r--  1 mwnurnbe      439 Sep 26 11:13 volume.F

femats/src/preproc:
total 60
drwxrwxrwx  2 mwnurnbe      512 Oct  4 20:49 .
drwxrwxrwx  6 mwnurnbe      512 Oct  5 10:41 ..
-rw-r--r--  1 mwnurnbe      682 Oct  4 14:48 Makefile
-rw-r--r--  1 mwnurnbe    19856 Oct  3 13:46 c2p.f
-rw-r--r--  1 mwnurnbe    22608 Oct  3 13:39 c2p_2d.f
```

```
-rw-r--r--  1 mwnurnbe     3702 Oct  3 13:39 count.f
-rw-r--r--  1 mwnurnbe      164 May  8 21:45 parmvl
-rw-r--r--  1 mwnurnbe     8956 Oct  4 14:43 u2c.f

femats/test:
total 4
drwxrwxrwx  4 mwnurnbe      512 Oct  5 09:10 .
drwxrwxrwx  7 mwnurnbe      512 Oct  5 10:36 ..
drwxrwxrwx  2 mwnurnbe      512 Oct  5 15:21 preproc
drwxrwxrwx  2 mwnurnbe     1024 Oct  5 15:21 supercomp

femats/test/preproc:
total 983
drwxrwxrwx  2 mwnurnbe      512 Oct  5 15:21 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 09:10 ..
-rw-r--r--  1 mwnurnbe    37756 Oct  4 19:57 small_plate.cntr
-rw-r--r--  1 mwnurnbe   195800 Oct  4 19:57 small_plate.cnv
-rw-r--r--  1 mwnurnbe      308 Oct  4 19:57 small_plate.data
-rw-r--r--  1 mwnurnbe    93477 Oct  4 19:57 small_plate.edgy
-rw-r--r--  1 mwnurnbe       12 Oct  4 19:57 small_plate.eg
-rw-r--r--  1 mwnurnbe   238485 Oct  4 19:57 small_plate.eglob
-rw-r--r--  1 mwnurnbe    46232 Oct  4 19:57 small_plate.enoddy
-rw-r--r--  1 mwnurnbe    50541 Oct  4 19:57 small_plate.esurfed
-rw-r--r--  1 mwnurnbe     4481 Oct  4 19:57 small_plate.grp
-rw-r--r--  1 mwnurnbe      251 Oct  4 19:57 small_plate.otpt
-rw-r--r--  1 mwnurnbe   286783 Oct  4 19:57 small_plate.unv
-rw-r--r--  1 mwnurnbe     3903 Oct  4 19:57 transcript.preproc

femats/test/supercomp:
total 19649
drwxrwxrwx  2 mwnurnbe     1024 Oct  5 15:21 .
drwxrwxrwx  4 mwnurnbe      512 Oct  5 09:10 ..
-rw-r--r--  1 mwnurnbe     1950 Oct  5 09:40 bis.p0.0
-rw-r--r--  1 mwnurnbe     1950 Oct  5 09:40 bis.p0.90
-rw-r-----  1 mwnurnbe       12 Oct  5 09:10 eg
-rw-r--r--  1 mwnurnbe       60 Oct  5 09:40 input.p0.0
-rw-r--r--  1 mwnurnbe       61 Oct  5 09:40 input.p0.90
-rw-r-----  1 mwnurnbe    37756 Oct  5 09:10 small_plate.cntr
-rw-r-----  1 mwnurnbe      308 Oct  5 09:10 small_plate.data
-rw-r-----  1 mwnurnbe    93477 Oct  5 09:10 small_plate.edgy
-rw-r-----  1 mwnurnbe       12 Oct  5 09:10 small_plate.eg
-rw-r-----  1 mwnurnbe   238485 Oct  5 09:10 small_plate.eglob
-rw-r-----  1 mwnurnbe    46232 Oct  5 09:10 small_plate.enoddy
-rw-r-----  1 mwnurnbe    51793 Oct  5 09:10 small_plate.esurfed
-rw-r-----  1 mwnurnbe       44 Oct  5 09:10 small_plate.in
```

```
-rw-r-----  1 mwnurnbe      251 Oct  5 09:10 small_plate.otpt
-rw-r-----  1 mwnurnbe       22 Oct  5 09:10 small_plate.otpt.new
-rw-r-----  1 mwnurnbe 11200008 Oct  5 09:10 test1.dat
-rw-r-----  1 mwnurnbe  1960008 Oct  5 09:10 test2.dat
-rw-r-----  1 mwnurnbe  3920008 Oct  5 09:10 test3.dat
-rw-r-----  1 mwnurnbe  1400008 Oct  5 09:10 test4.dat
-rw-r-----  1 mwnurnbe  1080008 Oct  5 09:10 test5.dat
-rw-r-----  1 mwnurnbe      913 Oct  5 09:10 transcript.ksr
```

H References

1. A. Chatterjee, J.M. Jin and J.L. Volakis, "Application of edge-based finite elements and ABCs to 3-D scattering," *IEEE Trans. Antennas Propagat.*, vol. 41, pp. 221–26, February 1993.

2. D.R. Kincaid and T.C. Oppe, "ITPACK on supercomputers," *Numerical Methods, Lecture Notes in Mathematics*, vol. 1005, pp. 151–61, Springer, Berlin, 1982.