FRANC3D

Concepts & Users Guide

Version 2.6

2003

1. Introduction

This document is a general concepts and user's guide for FRANC3D, a FRacture ANalysis Code for 3 Dimensional problems. It explains some of the concepts behind the program and general remarks on the use of the program for analyzing cracked structures. More detailed information on specific menu commands is available in the Menu & Dialogue Reference manual. The FRANC3D Tutorial manual gives some specific examples that illustrate both the use and capabilities of the program for analyzing cracked structures.

A list of references is added in the next section in case a deeper understanding of the ideas behind the program is needed. The remainder of this document is then divided into several parts. The first will present a high-level description of the basic ideas that form the core of the program, or at least point the reader to references where these ideas are explained. The next section includes a description of the geometric information, specifically the format FRANC3D requires for surface patch information, as well as descriptions of the meshing algorithms, the simulation attributes, and the methodology of 3D crack propagation in FRANC3D. The last part is a description of the program environment, and explains how to interact with and control the program and its various components.

2. References

The following references were either used in the development of FRANC3D, or provide additional in-depth background reading on relevant topics.

2.1 Solid Modeling:

Baer, A., Eastman, C., and Henrion, M., Geometric Modelling: A Survey, Comp.-Aided Design, Vol. 11, No. 5, Sept. 1979, pp. 253-272.

Casale, M., Stanton, E., An Overview of Analytic Solid Modeling, IEEE Comp. Graphics and App., Feb. 1985, pp. 45-56.

Mantyla, M., **An Introduction to Solid Modeling**, Computer Science Press, Rockville, Maryland, 1988.

Mortenson, M.E. **Geometric Modeling**. John Wiley & Sons, New York, 1985.

2.2 Splines, etc.:

Bartels, R., Beatty, J., Barsky, B, An Introduction to the Use of Splines in Computer Graphics, SIGGRAPH '85, 1985.

Bohm, W., Farin, G., Kahmann, J., A Survey of Curve and Surface Methods in CAGD, Comp.-Aided Geometric Design, 1, 1984, pp. 1-60.

2.3 Radial Edge Data Structure:

Mantyla, M., Sulonen, R., GWB: A Solid Modeler with Euler Operators, IEEE Comp. Graphics and App., Sept., 1982, pp. 17-31.

Weiler, K., Topological Structures for Geometrical Modeling, Ph.D. Dissertation, Rensselaer Polytechnic Institute, Troy, NY, 1986.

Weiler, K., Non-Manifold Geometric Boundary Modeling and Two Taxonomies for Geometric Modeling Representations, SIGGRAPH '87 Advanced Solid Modeling Tutorial, 1987.

Weiler, K., The Radial-Edge Structure: A Topological Representation for Non-Manifold Geometric Boundary Representations, **Geometric Modelling for CAD Applications**, North Holland, pp. 3-36, 1988.

Wilson, P.R., Euler Formulas and Geometric Modeling, IEEE Comp. Graphics and App., August, 1985, pp. 24-36.

2.4 Computational Mechanics System:

Bittencourt, T.N. Computer simulation of linear and non-linear crack propagation in cementitious materials. PhD Thesis, Cornell University, Ithaca, NY, 1993.

Lutz, E. D., Numerical Methods for Hypersingular and Near Singular Boundary Integrals in Fracture Mechanics, PhD. Thesis, Cornell University, Ithaca, NY, 1991.

Martha, L.F., Topological and Geometrical Modelling Approach to Numerical Discretization and Arbitrary Fracture Simulation in Three-Dimensions, Ph.D. Thesis, Cornell University, Ithaca, NY, 1989.

D.O. Potyondy, J.F. Abel, and A.R. Ingraffea, An Interactive Environment for the Simulation of 3D Concrete Subassemblages, Sci-C 1990, Second Int. Conf. on Computer-Aided Analysis and Design of Concrete Structures, Zell Am See, Austria, April 1990.

Potyondy, D.O., Toward the Simulation of Three-Dimensional Reinforced Concrete Subassemblages, M.S. Thesis, Cornell University, Ithaca, NY, 1990.

Potyondy, D.O., A Software Framework for Simulating Curvilinear Crack growth in Pressurized Shells, Ph.D. Thesis, Cornell University, Ithaca, NY, 1993.

Shepard, M.S., Finite Element Modelling Within an Integrated Geometric Modelling Environment: Parts I and II, Engng. with Comp., Vol. 1, No. 2, 1985, pp. 61-85.

Sousa,J.L.,Three-Dimensional Simulation of Near-Wellbore Phenomena Related to Hydraulic Fracturing From A Perforated Wellbore. Ph.D. Thesis, Cornell University, 1992.

Sousa,J.L. *et al.*, Simulation of Non-planar Crack Propagation in Three-dimensional Structures in Concrete and Rock**, Fracture of Concrete and Rock: Recent Developments** (Ed. S.P.Shah), Elsevier Applied Science, 1989, pp.254-261.

Srivastav, S.S., 3D Modelling of Building for Nonlinear Seismic Analysis, to be presented at Eurodyn '90, European Conference on Structural Dynamics, Bochum, Germany, 1990.

Wawrzynek, P.A., Interactive Finite Element Analysis of Fracture Processes: An Integrated Approach, M.S. Thesis, Cornell University, Ithaca, NY, 1987.

Wawrzynek, P.A., Martha, L.F., and Ingraffea, A.R., A Computational Environment for the Simulation of Fracture Processes in Three Dimensions, Analytical, Numerical, and Experimental Aspects of Three Dimensional Fracture Processes, ASME AMD-Vol. 91., Ed. A.J. Rosakis *et. al*, pp. 321-327, 1988.

2.5 Programming Environment

Cox, B.J. **Object Oriented Programming: An Evolutionary Approach** , Addison-Wesley, 1986.

Nye, A. and O'Reilley,T., X Toolkit Intrinsics Programming Manual, Volume 4, O'Reilley & Associates Inc., 1990, esp. Chap. 1 and 2.

2.6 FRANC3D

Carter, B.J., Ingraffea, A.R. and Bittencourt, T.N. Topology-controlled modeling of linear and non-linear 3D-crack propagation in geomaterials. **Symposium on Fracture of Brittle Disordered Materials: Concrete, Rock and Ceramics**, Conference of the International Union of Theoretical and Applied Mechanics, Brisbane, Australia, E&FN Spon Publishers, 1993.

Martha, L.F., Wawrzynek, P.A. and Ingraffea, A.R. (1993) Arbitrary crack representation using solid modeling. Engineering with Computers, 9, 63-82.

Wawrzynek, P. A., Discrete Modeling of Crack Propagation: Theoretical Aspects and Implementation Issues in Two and Three Dimensions, Ph.D. Thesis, Cornell University, Ithaca, NY, 1991.

Wawrzynek, P.A., Carter, B.J., Ingraffea, A.R. and Potyondy, D.O. A topological approach to modeling arbitrary crack propagation in 3D in "DIANA Computational Mechanics 94". Proc. 1st Int. DIANA Conf., Delft, The Netherlands, Edited by: G. Kusters and M. Hendriks. Kluwer Academic Publishers, p.69-84, 1994.

Wawrzynek, P., Martha, L. and Ingraffea, A.R. FRANSYS: a software system for the simulation of crack propagation in three dimensions, in Discretization Methods in Structural Mechanics, IUTAM/IACM Symposium, G. Kuhn and H. Mang, eds., Springer-Verlag, New York, 273-282, 1990.

2.7 Fracture

Erdogan, F. and Sih, G.C. On the crack extension of plates under plane loading and transverse shear. J. Basic Engng., 85, 519-527 (1963).

Hodgdon, J. and Sethna, J.P. Beyond the principle of local symmetry: derivation of a general crack propagation law. Phys. Rev. B, 47, 4831 (1992).

Leblond, J.B. Crack kinking and curving in three-dimensional elastic solids–application to the study of crack path stability in hydraulic fracturing, in **Mixed-mode Fatigue and Fracture**. H.P. Rossmanith and K.J. Miller, eds., European Structural Integrity Society, Mech. Engng. Publ., 219-243 (1993).

Lim, I.L., Johnston, I.W. and Choi, S.K. Comparison between various displacement-based stress intensity factor computation techniques. Int. J. Fract., (1992).

Sih, G.C. Strain-energy-density factor applied to mixed-mode crack problems. Int. J. Fract. Mech., 10, 305-321 (1974).

Sih, G.C. A three-dimensional strain energy density factor theory of crack propagation, in **Three Dimensional Crack Problems, Mechanics of Fracture, 2**, XV-LIII, Noordhoff, Leyden (1975).

2.8 Fatigue

Broek, D., **Elementary Engineering Fracture Mechanics**, *4th Edition*, pp. 279-282, Martinus Nijhoff, 1986.

Forman, R.G., Shivakumar, V., Newman, J.C., Fatigue Crack Growth Computer Program "NASA/FLAGRO" Version 2.0, Johnson Space Center, Houston, Texas, Rpt. #JSC-22267A, 1994.

Forman, R.G., Mettu, S.R., "Behavior of surface and corner cracks subjected to tensile and bending loads in Ti-6Al-4V Alloy," Fracture Mechanics: Twenty-second Symposium, Vol. 1, ASTM STP 1131, Ernst, Saxena, & McDowell, eds. American Society for Testing and Materials, Philadelphia, 1992, pp. 519-546.

Newman, Jr. J.C., "A crack opening stress equation for fatigue crack growth," International Journal of Fracture, Vol. 24, No. 3, March 1984, pp. R131-R135.

Schijve, J., "Observations on the prediction of fatigue crack growth propagation under variable-amplitude loading," Fatigue and Crack Growth Under Spectrum Loads, ASTM *STP 595*, American Society for Testing and Materials, Philadelphia, 1976, pp. 3-23.

Tanaka, K., Nakai, Y., and Yamashita, M., "Fatigue growth threshold of small cracks," International Journal of Fracture, Vol. 17, No. 5, October 1981, pp. 519-533.

Wheeler, O.E., “Spectrum loading and crack growth,” J Basic Engng, Vol. 94, 1972, pp. 181-186.

Willenborg, J., Engle, R.M., and Wood, H.A., A crack growth retardation model using an effective stress concept, AFFDL-TM-71-1-FBR, Wright-Patterson AFB, 1971.

3 Basic Ideas of the System

Performing an engineering simulation of structural response of a *realistic* 3D structure is difficult. It requires more than simply an analysis program. The analysis itself may be only one part of the total simulation process, and presents no real difficulty if standard engineering simplifications are made, e.g., linear material and geometric response. The basic procedures have been well established in both the boundary element and finite element literature, and numerous commercial codes exist.

However, simulation encompasses all aspects of the modeling process, from initial data preparation to final results visualization (and iterative re-analysis if necessary). A simulation system should provide a framework to help the engineer conceptualize the stages of the modeling process. The framework should be intuitive and simple, yet enable enough complexity to be useful. The system should hide as much of the modeling complexity as possible from the user, thereby, allowing one to focus on the behavior of the structure without needing to be overly concerned with the details of model representation, attribute specification, or mesh generation. First and foremost, it is the insight (not just the numerical values) that the simulation can provide which is of value to the research engineer. A well-designed simulation system provides the tools, and packages them in a way that promotes such insight.

Many concepts and ideas have been brought together in FRANC3D in an effort to address the above objectives. Taken together, they form a coherent approach to performing engineering simulations. The following major concepts are embodied in FRANC3D:

- solid modeling tools;
- a topological data structure that allows topology to be separated from geometry;
- the association of model attributes with topological entities;
- a hierarchy of topological models to organize and guide the discretization process;
- use of interactive computer graphics on high-performance engineering workstations and; and
- a comprehensive user-interface to combine the above into one cohesive package with a consistent look and feel.

3.1 A Conceptual Model of Crack Growth Simulation

In this section, a conceptual model of the discrete crack growth simulation process is presented. This model is used as a framework for a detailed discussion of the independent components that follow.

The crack simulation process is an incremental one, where a series of steps is repeated for a progression of models. Each iteration in the process relies on previously computed results, and represents one crack configuration. There are four primary collections of data, or databases, required for each iteration. The first is the representational database, denoted $\mathbf{R}_i$, (where the subscript identifies the iteration or increment number). The

representational database contains all the information necessary for an unambiguous description of a cracked body. This includes a description of the solid model geometry (including any cracks), the applied tractions, displacements, and body forces (and any associated rates), and material constants, along with any associated material state or history information. It is devoid of information that is specific to a particular numerical technique for performing stress analysis.

The representational database is transformed by a discretization (meshing) process to a stress analysis database, $\mathbf{A_i}$. This contains a complete, but approximate description of the body, suitable for input to a specific stress analysis program. Typically, this includes a series of nodal points where primary field variables will be computed or prescribed, a mesh connecting these points (surface for boundary element analysis, volume for finite element analysis), a specification of known conditions at the nodal points, material state and constants, and information necessary to compute body forces, and residual stresses and strains.

A solution procedure is used to transform the analysis database to an equilibrium database, $\mathbf{E_i}$ that consists of primary (loads and displacements) and secondary (stresses and strains) field variables that define an equilibrium solution for the analysis model $\mathbf{A_i}$. The solution procedure is usually a standard finite or boundary element technique. The equilibrium model should contain field variables and material state information for all locations in the body. These values may be stored explicitly at nodal or integration points, or evaluated using interpolation or extrapolation. In the context of crack growth simulation, the equilibrium model will also contain values for stress-intensity factors, or other fracture parameters, at all points along all crack fronts.

The equilibrium database is used in conjunction with the current representational database to create a new representation, $\mathbf{R_{i+1}}$. The new model represents an incremental step of growth of the crack front based on previously computed results. This process is repeated until a condition for terminating the simulation is satisfied (e.g. the onset of unstable cracking, a crack larger than a maximum allowable size, or a crack propagating completely through a body).

The crack growth simulation process can be described symbolically as follows. A meshing function, **M**, transforms a representational description of a cracked body to a discrete model suitable for stress analysis,

$$\mathbf{M}(\mathbf{R}_i) \rightarrow \mathbf{A}_i .$$

A stress analysis procedure, **S**, computes unknown field variables and fracture parameters, here denoted F_i for all points along all crack fronts,

$$\mathbf{S}(\mathbf{A}_i) \rightarrow \mathbf{E}_i, \mathbf{F}_i .$$

A function which updates the representational model, **U**, takes the equilibrium state field variables, the existing representation, and a function which predicts crack shape evolution, **C**, and creates a new representational database,

$\mathbf{U}(\mathbf{E}_i, \mathbf{R}_i, \mathbf{C}(\mathbf{F}_i)) \rightarrow \mathbf{R}_{i+1}$.

This process is performed incrementally, and is repeated until a suitable termination condition is reached (Figure 1). Useful results of such a simulation will be one or more of the following: a final crack geometry, a loading versus crack size history, a crack opening profile, or a history of the crack-front fracture parameters.

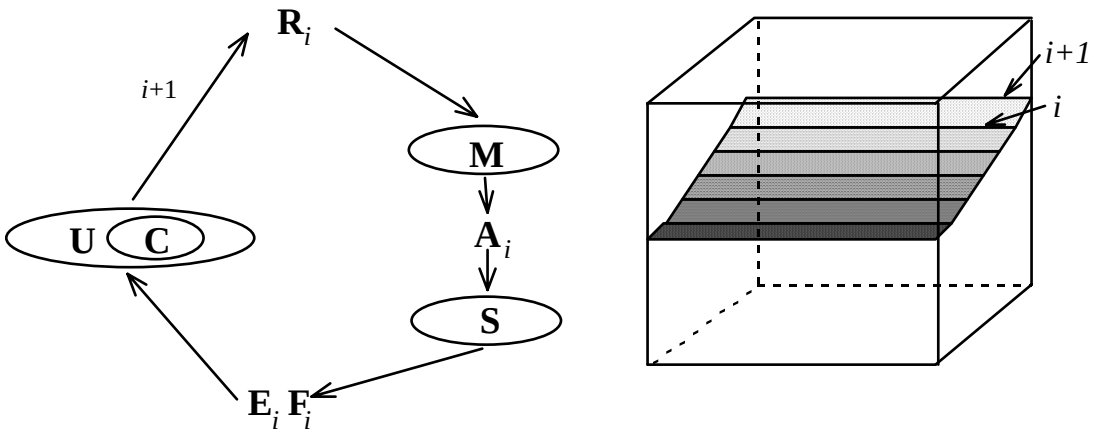


Figure 1. Incremental crack growth simulations.

Before useful engineering simulations can be performed, the abstract databases, $\mathbf{R}_i$, $\mathbf{A}_i$, $\mathbf{E}_i$, and $\mathbf{F}_i$, and the abstract functions, **M**, **S**, **C**, and **U**, must be redefined in terms of data structures and algorithms. The focus of the following sections is on mechanisms and methodologies used for FRANC3D and how they differ in the context of fracture mechanics from other applications. These include aspects of geometrical modeling, the use of computational topology, hierarchies and constraint, and special aspects of attributes.

3.2 Model Representation

3.2.1 Solid Modelling

Solid modeling refers to the representation and processing of geometric information on three-dimensional solid objects. Put simply, solid modeling encompasses the various methods for representing geometric information on a computer. See Mantyla (1988) for a good introduction to solid modeling.

Engineering simulation in FRANC3D utilizes a hierarchy of models. These models range from a solid model of a mechanical idealization (in our case, a general 3D continuum) of a real structure, to an analytical model of the structure. The solid model

captures those geometric aspects of the real structure deemed necessary to represent the behavior of interest, and the analytical model (finite or boundary elements) approximates the mechanical behavior (displacements, stresses, etc.).

In the past, most engineering simulations were performed on a single analytical model consisting of a mesh and its associated attributes; thus, in such a system, *the mesh was the model.* All geometric information had to be inferred from the mesh. Furthermore, attributes, such as boundary conditions and material properties, had to be directly attached to elements and/or nodes instead of being more naturally attached to the geometric volumes, surfaces, edges, and points.

Solid modeling techniques now make it possible to represent the desired geometrical aspects of the structure explicitly. It is to this solid model that attributes are attached. The eventual discretization can be thought of as being mapped onto (or into) this original solid model. When an analysis is to be performed, the attributes can be inherited automatically by the elements comprising the discretization. Also, if the simulation requires any remeshing, the solid model remains valid and the new mesh can be remapped into the model and will again automatically inherit the necessary attributes (Shepard, 1985).

The simulation of crack propagation introduces additional complexities not seen in other applications of solid modeling. The two mated faces of a closed crack represent distinct surfaces that are geometrically coincident, a situation that cannot be represented by some existing approaches to solid modeling. This is illustrated by the process of point classification, a fundamental capability of any solid modeling system. Conventionally, this means given any point, determine if the point is in a body, outside a body, or on the surface of a body. A point on a crack surface cannot be classified as any *one* of these three. It is simultaneously on the surface of the body at two distinct independent locations, with no adjacent points that lie outside the body. A solid modeling capability will be suitable for crack growth simulations only if proper classification of points on crack faces can be performed.

One possible approach for avoiding this difficulty is to assume that cracks have an arbitrarily selected finite opening. This is undesirable because the introduction of a small fictitious crack opening may lead to ambiguous computations due to tolerance and round-off errors. To this, one may argue that real crack faces are not mathematically coincident, but rather some finite opening and interpenetration of crack faces exists. This is true, but the scales over which these effects take place are usually small relative to other dimensions of a body, and depend on specific material properties and the rate of fracture. In the context of material science, such accurate modeling of crack face details may be important, but such detail is not important for most engineering applications.

Crack faces are surfaces, or more properly, two surfaces that share a common geometrical description. This can be modeled quite readily with a boundary representation (B-rep) modeler, which stores surfaces and surface geometries explicitly. In this case, either crack surface must be flagged to indicate that it represents two surfaces, or if explicit topological adjacency information is available (see the next

section), two topologically distinct surfaces may share a common geometrical description. Point classification for crack surfaces presents no particular difficulty if such an explicit representation of crack surfaces is available. Thus, the only significant enhancement required of a B-rep modeler for simulating crack growth is the ability to flag surfaces as mated crack faces. However, as is outlined in the next section, having topological adjacency information available explicitly will ease the implementation of other tasks such as meshing.

3.2.2 Topological Data Structure

The internal data representation of FRANC3D has been developed to facilitate the storage and manipulation of three-dimensional objects whose topology and geometry may vary during the course of a simulation. It was decided to deviate from the conventional means of storing data for finite and boundary element analysis (connectivity, coordinate lists, etc.) for the following reasons:

- the desire to store surface descriptions (including surfaces of material boundaries), boundary conditions, and attributes explicitly and separately from the mesh;
- the desire for rapid topological queries to provide for real-time user interaction; and
- the desire to utilize mapped meshing algorithms wherever possible to generate meshes.

The above considerations have led to a representation which is comprised of a hierarchy of models, each of which is a radial-edge topological representation of the structure being analyzed. The following subsections describe both topology and the radial-edge representation. The hierarchy of models is described later in its own section.

3.2.3 Topology

Topology is a unified high-level abstraction of geometric information. There are many forms of topology, but it is adjacency topology that is meant herein. Adjacency topology describes the adjacency of topological primitives. The primitives, for example, might consist of vertices, edges, and faces. The idea of topology can be understood by considering its relation to geometry, as geometry is a more intuitively familiar concept. Any physical object can be described fully by its geometry which gives the exact spatial locations of all portions of the object. Topology is a subset of the geometrical information. The topology can be derived from the geometry, but the geometry cannot be reconstructed solely from the topology.

For example, imagine a hexahedral solid object. It is topologically a cube, i.e., it has six faces, twelve edges, and eight vertices. Furthermore, each of these topological primitives is adjacent to one another in some specific fashion, e.g., each edge is adjacent to two faces, each vertex is adjacent to three edges, etc. One could be given a data structure enumerating these adjacency relationships, but unless one also knew the geometric attributes of the topological primitives (e.g., Cartesian coordinates of each vertex or planar equation for each face), one could not reconstruct the hexahedron in Cartesian space.

Having explicit topological information available is not essential for crack growth simulation. However, there are at least four compelling reasons why the topological representation of an object is a fruitful framework for organizing the necessary information.

1. Topological information, unlike geometrical information, can be stored exactly, with no approximations or ambiguity. In addition, if an explicit topological representation is available, the intended properties of an object can be represented, even with the presence of certain geometric inaccuracies.
2. There is a rich theoretical background supporting the concepts of the topology of boundary graphs. This theory can be used to develop formal and rigorous procedures for storing and manipulating these types of data (see Mantyla, 1988 or Weiler, 1986).
3. Any topological configuration can represent an infinite number of geometrical configurations. Many representational problems can be solved in a generic sense in a topological space, and then mapped into a specific geometrical instance.
4. With crack propagation simulations, the geometry of an object changes with each crack increment. The topology of the object, however, changes much less frequently. Local modifications can be made without the need to perform global reorganizations of the rest of the data. Such frequent local modifications are common in the simulation of crack growth. The relative persistence of the topological description of an object makes it an ideal candidate for the representation to which geometric and attribute information is related.

3.2.4 Separate Geometry from Topology

As demonstrated by the hexahedron example given above, topological information alone is not sufficient to describe a three-dimensional solid object. Geometrical information must also be associated with each topological primitive. In solid modeling systems utilizing adjacency topological representations, the topology can be thought of as a framework or "glue" which holds all the geometrical information together. It functions as the organizing schema for the data structures of the system. In such a system, Cartesian coordinates are associated with vertices; 3D curve equations are associated with edges; and 3D surface equations are associated with faces (Figure 2).

It should be noted that the edges and faces need not be straight and/or planar. Any geometrical representation, for example, splines, could be used to represent the geometry of the edge and face primitives; therefore, an infinite number of distorted geometrical shapes could be described by the same topological data. The current implementation of FRANC3D supports three separate surface representations: planar polygons; quadrilateral, bi-cubic, non-rational B-splines; and tri-cubic Bezier-splines. Edge representations supported include straight edges and cubic B-splines.

3.2.5 Radial-Edge Topological Data Structure

There are many data structures which could be utilized to describe the adjacency relationships between topological primitives. The most common of these structures can model two-manifold surfaces in three-dimensions. Two-manifold surfaces can be thought of as the external surfaces of a polyhedron. But since the system described herein is intended to model complex three-dimensional objects with internal features (to allow for multi-material types and internal surfaces, e.g., the internal faces of a mesh), a more sophisticated data structure which can accommodate these non-manifold conditions is needed. The radial-edge topological data structure (Weiler, 1986) satisfies the above requirements.

Figure 3 illustrates the mechanism by which the radial-edge structure represents non-manifold conditions. Suppose that one wishes to represent a solid hexahedral structure composed of two separate materials. Such a model could be constructed by starting with a single solid and then adding an interior surface (topologically equivalent to a face) to split this box into two regions, each of which could then be assigned appropriate material property attributes. The interface addition results in the splitting of one region, four faces, and four edges, the addition of four edges and four vertices, as well as the addition of the interface (face) itself. There is a non-manifold condition occurring along each of these added edges, for example, edge A-B in Figure 3. The radial-edge structure stores the adjacency information of faces about this edge. One can loop radially about the edge A-B and extract the list of adjacent faces: [f8,f7,f1].

Maintaining the consistency of such a complex data structure during interactive modeling operations is a difficult task. It is handled via a set of operators which guarantee the topological validity of the model during all stages of model construction. These operators provide a clean and consistent interface to the data structure, which has resulted

in an efficient and modular system. Also, since adjacency relationships between topological entities are maintained explicitly, queries on the database are always performed locally and with algorithms whose asymptotic performance is, in the worst case, linearly proportional to the number of topological elements involved. Thus, the system provides rapid interactive response even with large models.

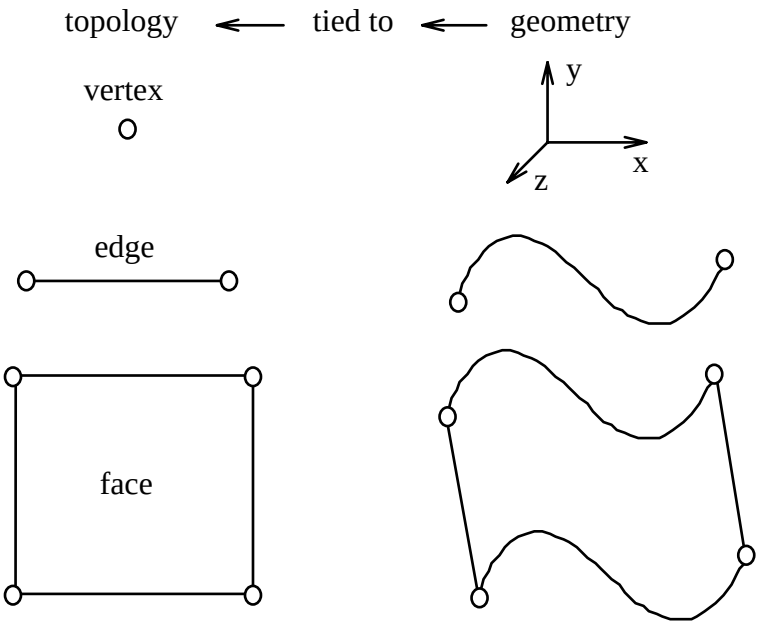


Figure 2. Relationship between topology and geometry; a topological entity can have any number of geometric descriptions.

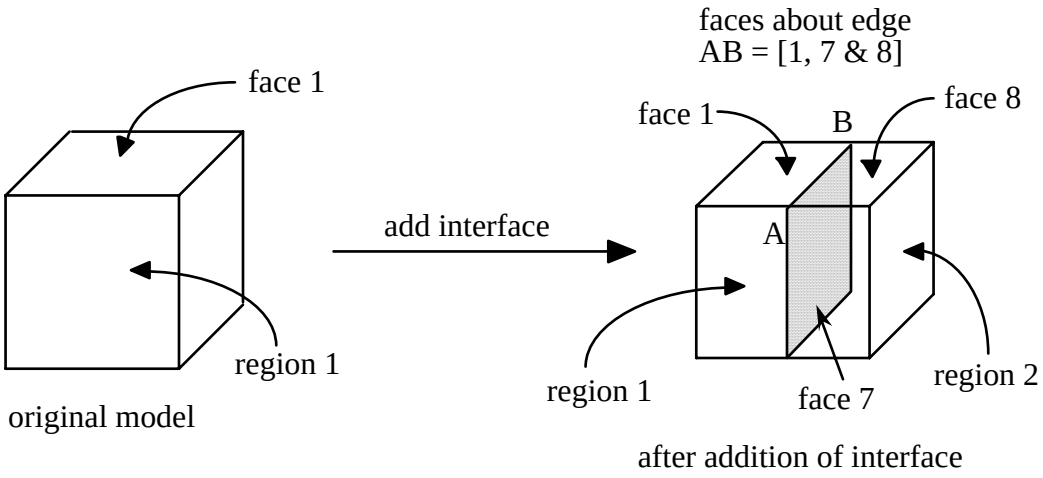


Figure 3. Radial-edge representation of a non-manifold condition.

3.2.6 Association of Model Attributes with Topological Entities

There are many attributes to be associated with the numerical model. Such attributes include: boundary conditions, material properties, intermediate modeling data, etc. In a conventional finite or boundary element program, the attributes are associated with the mesh. In FRANC3D, all attributes are associated with topological entities. For example, surface tractions are associated with faces, material properties with regions, and displacements with vertices and edges. Edges hold midside node information. Just as in the case of geometric attributes, the topology also acts as an organizing schema for the storage of analysis attributes. Simulation attributes are discussed further in a separate section below.

3.2.7 Hierarchy of Topological Models

Throughout a simulation, FRANC3D maintains a hierarchy of models, each of which is a radial-edge topological representation of the structure (Figure 4). Although the hierarchy is relatively transparent to the user, it does provide an underlying framework for the modeling process. Each level of the hierarchy corresponds to various stages of the discretization process as one moves from a solid model of the original structure to an eventual mesh with its associated attributes. The hierarchy also allows portions of the structure to be tied together into larger units upon which modeling operations can be performed and to which attributes can be assigned.

The hierarchy is comprised of the following five models:

- Geometry Model (GEO)
- Volume Decomposition Model (SDM)
- Face Decomposition Model (SRG)
- Edge Decomposition Model (SDV)
- Mesh Discretization Model (MSH)

The models are associated with one another through attribute information. For example, faces of the *mesh model* inherit appropriate boundary conditions from their parent faces in the *geometry model*. A hierarchy is enforced among the models such that each model inherits all the information from the models above itself, but may also contain additional information that has been specified by the user during the discretization process.

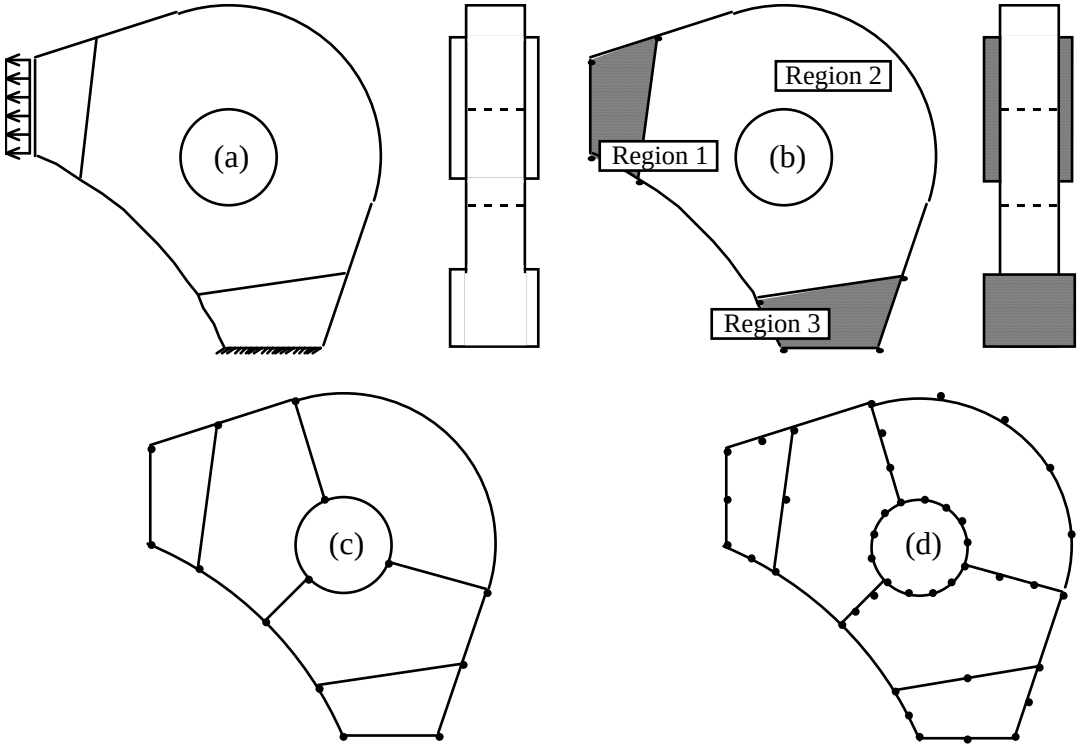
The geometry and mesh description represent, respectively, the conceptually highest and lowest levels in the five-level hierarchy. The second highest level is a volume decomposition level. Here, regions of the geometric model can be divided into subregions. One may desire to do this for a number of reasons, such as to define regions of different materials or to define a regularly shaped region to which a volume meshing algorithm can be applied.

In the next hierarchical level, entities of one lower level of dimensionality are divided. This is the surface subdivision level. In this level, surfaces can be divided into a number of subsurfaces. These new subsurfaces are constrained by the original surfaces. This

step may be necessary for a number of reasons, including the specification of surface boundary conditions over a portion of an existing surface, or to decompose an irregularly shaped surface into subsurfaces to which meshing algorithms can more easily be applied.

Following the same approach, the topological features of the next lower level of dimensionality are the edges. Likewise, the next level in the representation hierarchy is the edge subdivision level. The purpose of this level is twofold. First, at all the higher levels, the geometries of edges are represented as spline space-curves. At the edge subdivision level, these curves are approximated by piecewise linear line segments. The second purpose of this level is that the amount of edge subdivision is later used as a metric for local volume or surface mesh density.

The final level in the hierarchy is the level of the mesh (whether it's a surface or volume mesh, or a combination). This level, as are all levels, is constrained by the levels above it, so subdivided edge segments become element edges. Additionally, because this level is constrained by all levels above it, an element will not span different subsurfaces or subvolumes, and ultimately will be constrained by the geometry of the object.



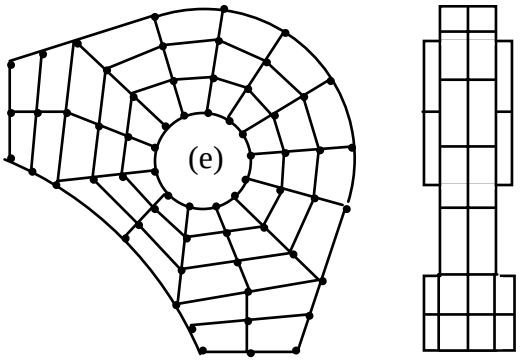


Figure 4. Constrained model hierarchy, (a) geometry, (b) volume decomposition, (c) surface decomposition, (d) edge subdivision, and (e) mesh.

There are two primary purposes for such a hierarchy. One is to provide constraints and inheritance for interactive modeling, and the other is to aid in localizing damage to the model during incremental analysis.

It is clearly important that the mesh model be constrained to the geometry of the structure. The hierarchy allows the mesh level model to inherit not only the geometry, but the simulation attributes as well. Simulation attributes, which consist of such things as boundary conditions and material properties are part of the original geometry model. However, the stress analysis procedure requires this data as well, meaning that the mesh model must inherit the data from the geometry. Element interpolation functions are used to extract data from the geometry model so that it can be attached to nodes or elements of the mesh model.

"Localized damage" is a concept useful in incremental simulations. When a portion of a model is modified, a certain amount of related information becomes obsolete. The concept of local damage says that the amount of information lost should be minimized. For example, in the case of crack propagation, small modifications to the geometry are made in the region near the crack front (Figure 5). This invalidates the mesh in this region. However, portions of the model remote from the crack need not be affected (need not be remeshed). The five levels of representation defined above provide a hierarchical framework within which local (or minimal) "damage" to the database can be enforced.

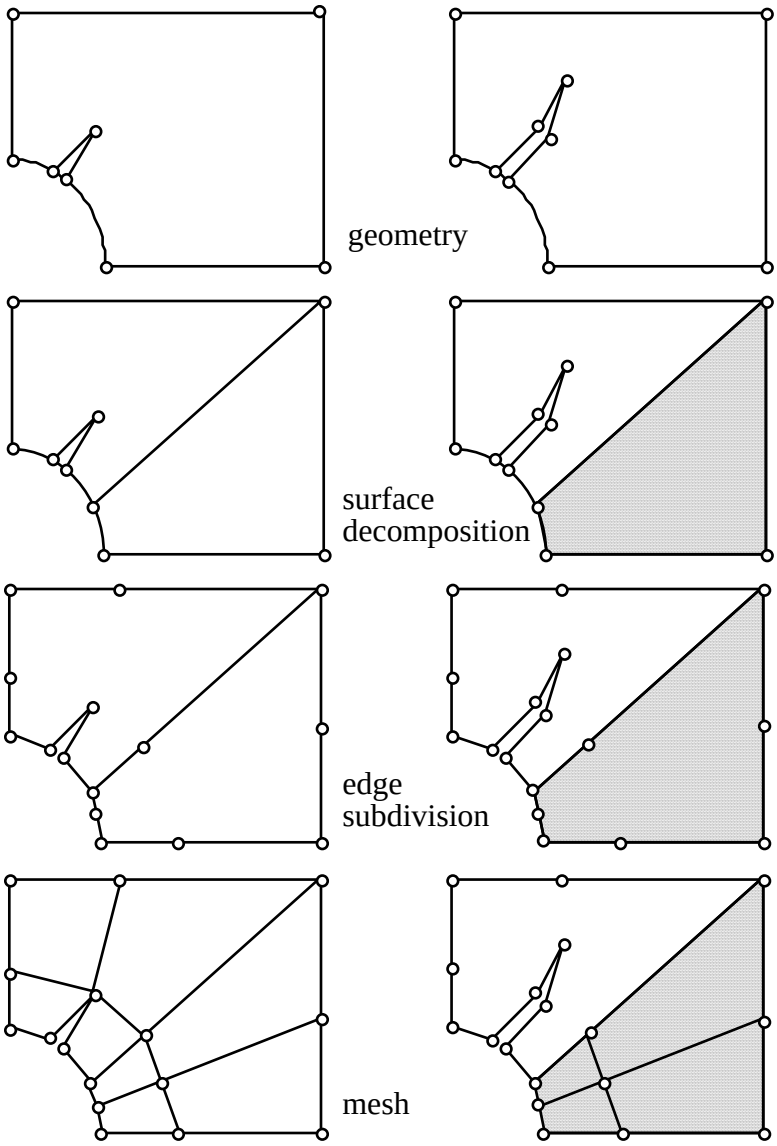


Figure 5. Concept of localized "damage" to the model during crack propagation.

3.2.8 A Succinct Description of the Radial-Edge Topological Representation

In Weiler (1986), there is a comprehensive description of topology and topological adjacency relationships in the framework of topological data structures. The essential topological components that constitute the radial-edge data structure, as well as their adjacency relationships, are summarized here. This section draws heavily on the work of Weiler and follows very closely along the lines of some of the developments in Weiler (1986).

One additional note is that here, as well as in Weiler's work, geometry is completely separated from topology. One can view geometry as attributes attached to topological elements in a data structure. For example, vertex coordinates are kept in the database as attributes of vertices. One can change the geometry of a model by manipulating the vertex coordinates without changing the topology of the model. The elegance and beauty of Weiler's work lies in the fact that topology is represented by the *use* of topological elements in the adjacency relationship information, rather than by the topological elements themselves. The introduction of the various *use* structures greatly simplifies most of the algorithms that modify and query the topological representation. However, physical topological elements – *region*, *face*, *edge*, and *vertex* (Figure 6) – are also included in the radial-edge representation. Although internally the *uses* of these elements are the actual links of this representation, in general the external interface with the topology is defined through basic, physical topological entities. This provides a clean and modular interface.

3.2.9 Radial-Edge Topological Elements

The description of the radial-edge topological entities is made through quotations from Weiler (1986):

> "A *model* is a single three-dimensional topological modeling space, consisting of one or more distinct (though perhaps adjacent) regions of space. A model is not strictly a topological element as such, but acts as a repository for all topological elements contained in a geometric model...
> A *region* is a volume of space (Figures 7 and 8). There is always at least one in a model. Only one region in a model may have infinite extent (e.g. region R_O of Figures 7 and 8); all others have a finite extent, and when more than one region exists in a model, all regions have a boundary...
>
> A *shell* is an oriented boundary surface of a region (Figures 7 and 8). A single region may have more than one shell, as in the case (shown in Figure 8) of a solid object with a void contained within it... A shell may consist of a connected set of faces which form a closed volume or may be an open set of adjacent faces, a wireframe, or a combination of these, or even a single point.
>
> A *face* is a bounded portion of a shell. It is orientable, though not oriented, as two region boundaries (shells) may use different sides of the same face. Thus only the

use of a face by a shell is oriented[1] . Strictly speaking, a face consists of the piece of surface it covers, but does not include its boundaries.

A *loop* is a connected boundary of a single face. A face may have one or more loops, for example (as shown in Figure 9) a polygon would require one loop and a face with a hole in it would require two loops. Loops normally consist of an alternating sequence of edges and vertices in an open circuit, but may consist of only a single vertex. Loops are also orientable but not oriented, as they bound a face which may be used by up to two different shells. Thus, it is the use of a loop that is oriented[2].

An *edge* is a portion of a loop boundary between two vertices. Topologically, an edge is a boundary curve segment which may serve as part of a loop boundary for one or more faces which meet at that edge. Every edge is bounded by a vertex at each end (possible the same one). An edge is orientable, though not oriented; it is the use of an edge which is oriented[3].

A *vertex* is a topologically unique point in space, that is, no two vertices may exist at the same geometric location (although the topology alone does not specify any exact geometric location beyond these constraints). Single vertices may also serve as boundaries of faces and as complete shell boundaries.

A *faceuse* is one of the two uses (sides) of a face. Faceuses, the use of a face by a shell, are oriented with respect to the face geometry.

A *loopuse* is one of the uses of a loop associated with one of the two uses of a face. It is oriented with respect to the associated faceuse.

An *edgeuse* is an oriented boundary curve segment on a loop-use of a faceuse and represents the use of an edge by that loopuse (see Figure 10), or if a wireframe edge, by endpoint vertices. Orientation is specified with respect to edge geometry. There may be many uses of a single edge in a model, but there will always be an even number of edge-uses (since each use by a face produces two edgeuses, one for each face side – see Figure 11). A wireframe edge produces two edgeuses, one for each end of the edge.

A *vertexuse* is a structure representing the adjacency use of a vertex by an edge as an edge point (as shown in Figure 10), by a loop in the case of a single vertex loop, or by a shell in the case of a single vertex shell."

[1] In FRANC3D the convention is adopted that a face receives the orientation of its first (face)use.
[2] As for a face, a loop in FRANC3D receives the orientation of its first (loop)use.
[3] An edge also receives the orientation of its first (edge)use in FRANC3D.

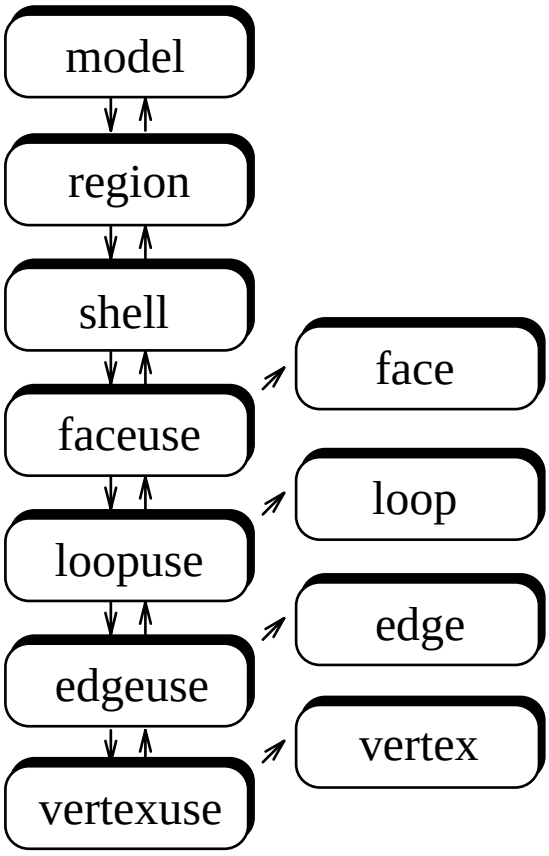


Figure 6. Radial edge topological elements.

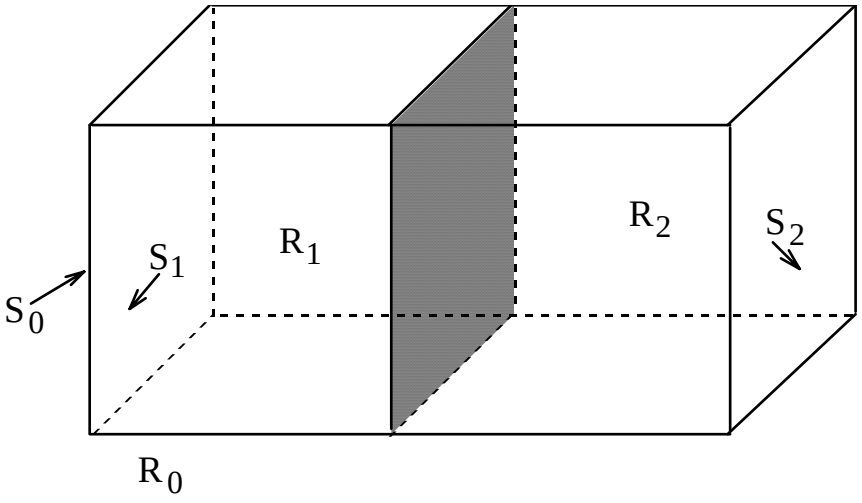


Figure 7. Multiple regions and their shells.

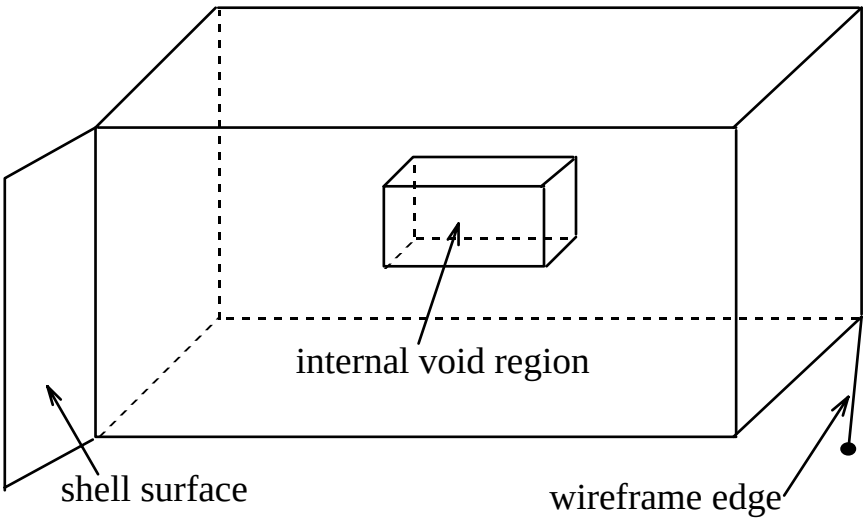


Figure 8. Non-manifold features such as wireframe edges, shell surfaces, and internal voids inside a region.

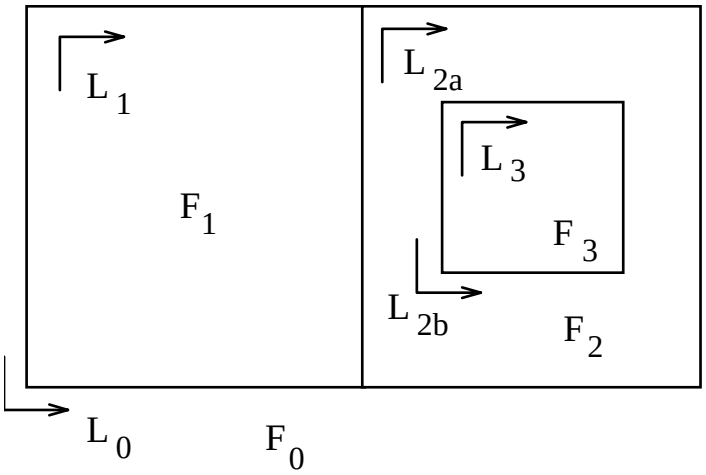


Figure 9. Faces and loops.

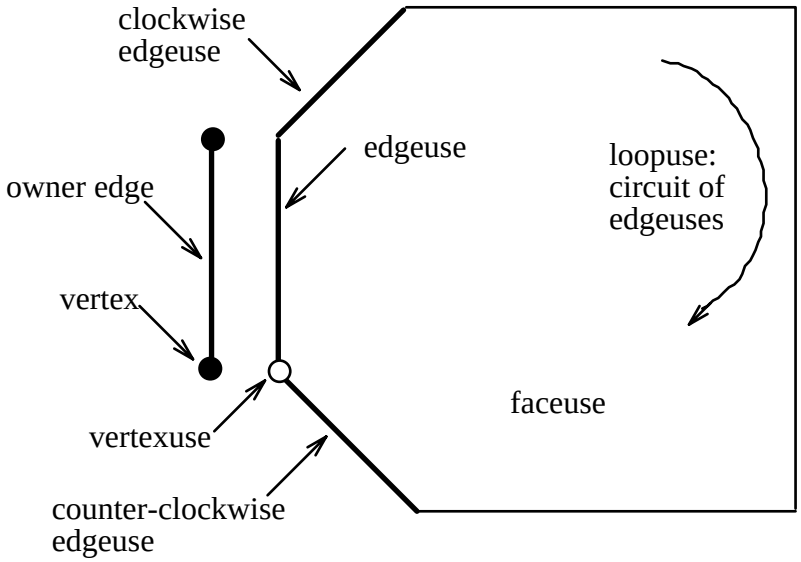


Figure 10. Faceuse, loopuse, edgeuse, vertexuse.

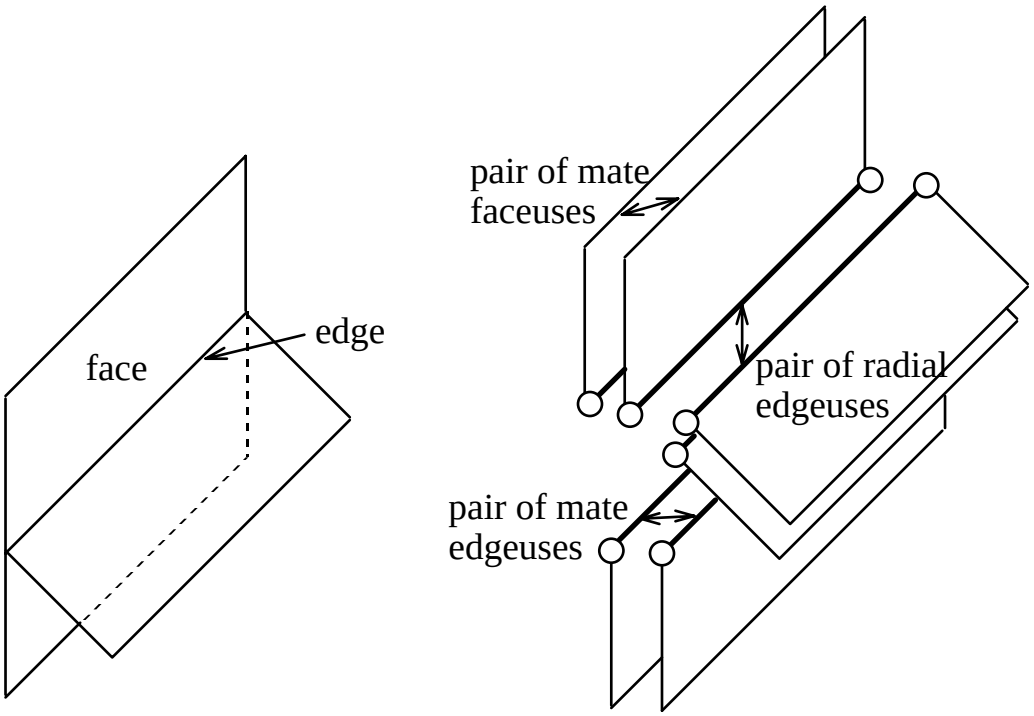


Figure 11. The radial-edge database uses the list of edge-uses, ordered radially about an edge to manage the manifold and non-manifold features.

3.3 Topological Representation of Cracks and Crack Propagation

Cracks are maintained at the geometry level, but all geometric entities have counterparts at the lower levels. Crack propagation affects all hierarchical levels by changing the original model geometry. However, only the entities directly adjacent to the growing crack front are affected.

Cracks are formed from a collection of faces, edges and vertices (Figure 12). A crack will generally consist of a main and mate side (a symmetry crack is modeled using only the main side). All faces will have mate faces, all edges except the crack front edges will have mate edges, and all vertices except the vertices on the crack front will have mate vertices. The crack tip vertices are those vertices which define the boundary between the crack front edges and the crack mouth or open boundary edges. For a simple 3D crack, there are either one or zero mate entities for the faces, edges and vertices which comprise the crack topology. The entities which form the crack are given crack codes based on the position of the entity on the crack. The main or mate side forms one half of the crack code. The second half is defined by the crack surface, crack front, and crack tip definitions.

A single crack in a 3D body can be classified as either internal or surface. An internal crack does not intersect the surface of the body whereas a surface crack may intersect one or more boundary surfaces. An internal crack could become a surface crack as it propagates and eventually could create a complete discontinuity as all crack front edges reach the bounding surface (Figure 13). A 3D crack or discontinuity, thus, can have zero, one, or multiple crack fronts.

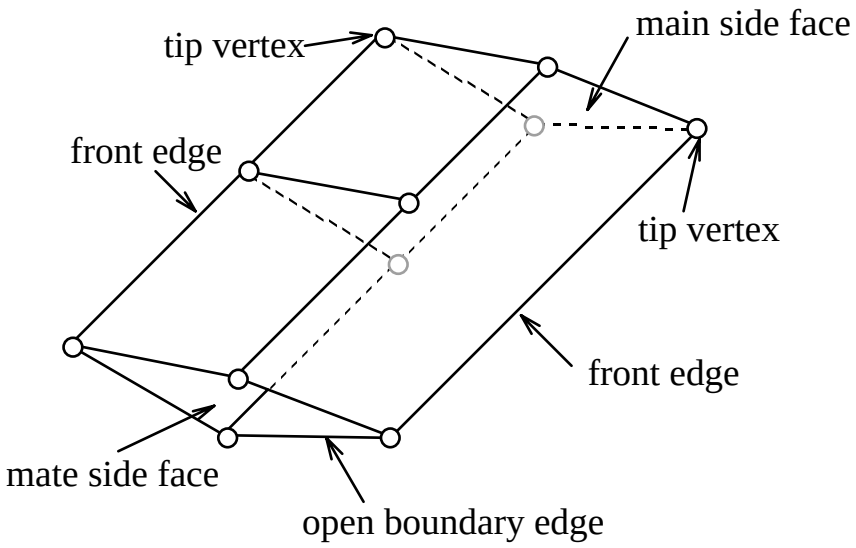


Figure 12. The topology of a typical crack.

An internal crack front consists of a closed-loop of edges which lie entirely within the body while a surface crack consists of a series of open boundaries on the surface and crack-front edges inside the body. However, a surface-crack front may form a closed-loop also, as in the case of circumferentially-cracked bar or cylinder (Figure 14). A distinction between the crack types is necessary because the topological operations for propagating the cracks are different. The propagation of a closed-loop surface crack is similar to the propagation of an internal crack, except that the front moves into the body rather than towards the boundary surfaces. Topological operations are similar because of the closed-loop of edges on the crack front; there is no clearly defined crack tip.

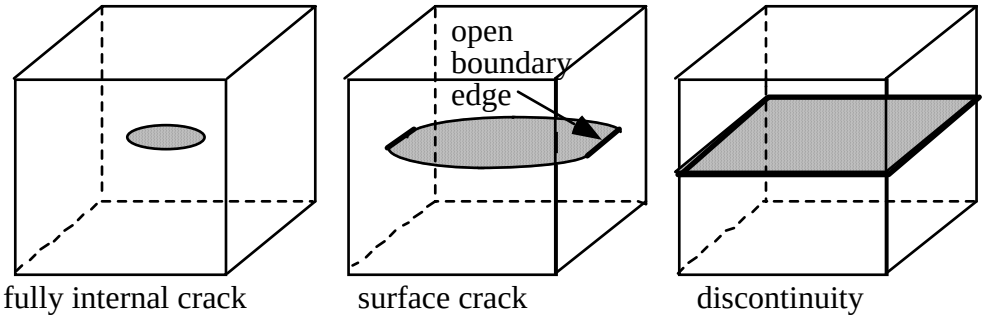


Figure 13. An internal crack could become a surface crack and eventually a complete discontinuity as it propagates. The shaded surface is the crack and the thick lines represent open boundaries on the model surface.

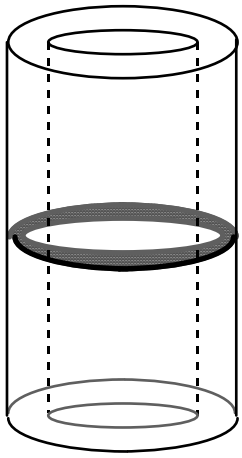


Figure 14. A circumferentially cracked cylinder representing a closed-loop surface crack.

3.3.1 Topological Representation of Crack Propagation

At least one of the crack fronts must move if the crack is to propagate. If there are multiple fronts in a single crack, each front must be examined separately. As the crack fronts move, they may change shape, size and orientation. A single front may intersect the model surface and form a series of smaller crack front segments separated by open-boundary edges. Crack fronts from adjacent cracks may merge, creating a single crack. Finally, all the crack fronts may disappear if the crack becomes a complete discontinuity. Therefore, routines for simulating crack propagation must be able to handle all the cases discussed here.

Crack initiation and propagation is done in one of two ways: using the manifold tear operator (temfl–tear edge, make face and loop) described by Martha *et al.* (1993), or through general purpose tear operators described in Carter *et al.* (1994). The first method is restricted to crack topologies that can be created from a manifold set of edges on manifold surfaces. A manifold edge implies that only two faces frame into the edge. An edge that lies on a face and has no faces framing into it from other directions is also manifold.

The creation of a part-through or penny-surface crack uses the temfl operator. For a part-through crack, a series of edges on the surface are torn, creating a new face and loop between the two sets of the edges (Figure 15). By adding an additional edge to this face, which extends from one end of the series of edges (future crack tip) to the other end, a crack front edge is created. The addition of this edge splits the one original face into two faces. These two faces represent the two crack surfaces–the main and mate side. The geometry of the crack is completed by setting the geometry of the crack front edge which also defines the geometry of the crack surfaces. Penny-shape cracks are created in a similar manner, but the number of edges and faces is increased.

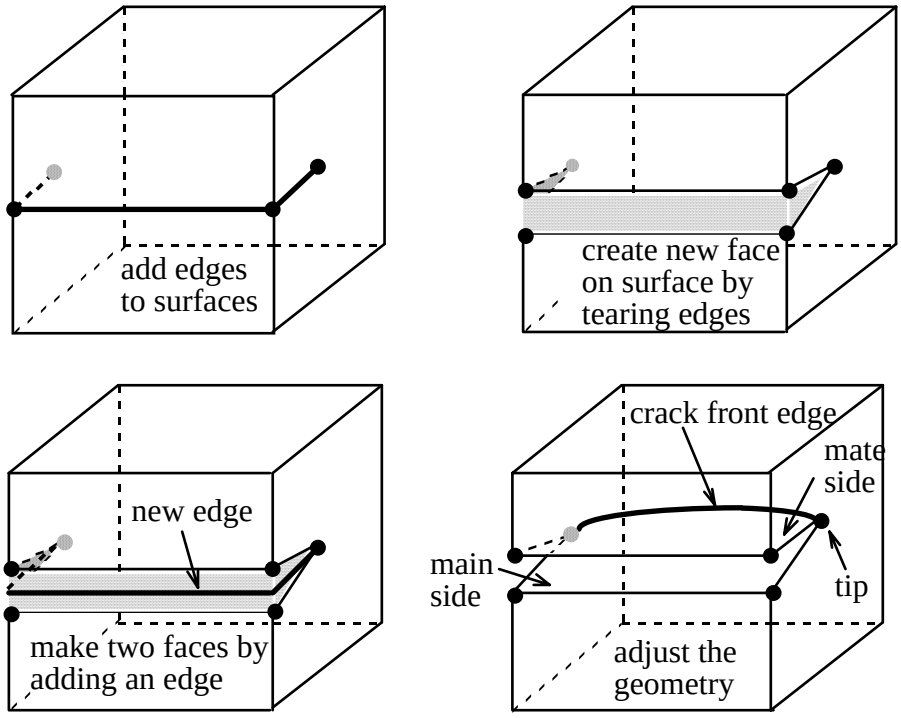


Figure 15. The manifold operations for creating a part-through surface crack using the temfl operator (after Martha, 1989)

3.3.2 Crack Propagation with General Purpose Tear Operators

An internal crack creates a non-manifold structure. The temfl operator cannot be used for the interactive addition of internal cracks. However, the crack geometry can be created by external methods and read into the database with the initial model geometry. An internal crack can be propagated once the geometry is created by expanding or changing the geometric description of the topological surfaces.

The temfl operator is also rather restrictive for nucleating and propagating complex, non-planar cracks. It is not possible to propagate an internal crack to the model surface, thereby creating a crack with multiple crack fronts. Consequently, it is also not possible to create a complete discontinuity with the temfl operator. A crack within or across a material interface is not possible either. Branching and intersecting cracks are impossible to create with the manifold temfl operator. These types of cracks cannot be created with the temfl operator because either the final crack is not manifold, or the intermediate steps produce a non-manifold topology. The creation of a crack across a material interface is a good example. The internal face which separates the two regions in the model is a non-manifold feature, and the creation of a crack across this interface requires a non-manifold operator.

Despite these restrictions on the temfl operator, the radial-edge data structure which is implemented in FRANC3D is capable of representing the topology of very complex crack geometries. With the addition of routines for interactively creating vertices, edges and faces in the model, including non-manifold entities, and the addition of low-level Euler operators for tearing the topological entities, the more complex cracks can be created interactively. Weiler (1986) describes several glue operators for gluing two vertices, two edges, and two faces together to create a single entity. The complementary tear operators have been implemented to create cracks.

The tear-operators allow a face (or set of faces and adjacent edges and vertices) to be torn apart, creating a new crack or an addition to an existing crack. This is accomplished through a sequence of tear-operations on the individual entities that comprise the crack topology (Figure 16).

The *tear* operators allow a face, or a set of faces along with their adjacent edges and vertices, to be torn apart, thereby creating a new crack or propagating an existing crack. This is accomplished through a sequence of *tear* operations on the individual topological entities that comprise the crack:

1) Tear the faces. A *tear face* operation creates a new mate face (F_2 in Figure 16) which uses the same topological edges and vertices as well as the same geometric description as the original main face (F_1 in Figure 16). A null volume region is created between the main and mate faces. For example, consider a single isolated face in the interior of a body. A new internal crack is formed by tearing this face; the bounding edges of the faces form the crack front.
2) Tear the edges. A *tear edge* operation is required for any topological edge that is adjacent to either: two torn faces, or a torn face and the free surface (model boundary). For example, consider two adjacent faces in the interior of a body (F_1 and F_2 in Figure 17a). Tearing both faces creates two new null volume regions. In order to make these two regions contiguous, such that a single crack is formed, the edges (E_1 and E_2 in Figure 17b) between the two original main faces must be torn. This produces main (E_1 and E_2) and mate edges (E_3 and E_4) on the respective sides of the crack.

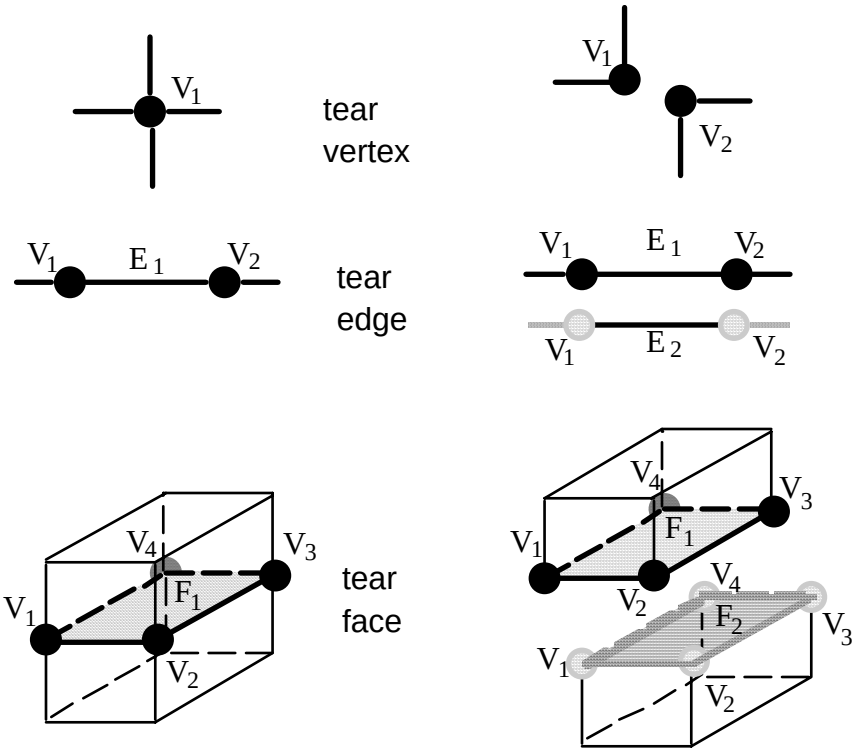


Figure 16. The individual tear operators.

3) Tear the vertices. A *tear vertex* operation is required for any topological vertex that is common to two torn main edges, excluding vertices that lie on the crack front edge. For example, crack tip vertices (see Figure 6) are not torn. The vertex (V_3) that is adjacent to the torn edges (E_1 and E_2 in Figure 17c) must be torn so that the crack surfaces are completely separated. Vertices that are not adjacent to two torn edges (V_1 and V_2) are shared by the main and mate faces.

These tear operators can be used to create any type of crack or discontinuity. If a geometric face can be created in the geometry model, then a crack can be formed. The complementary glue operators allow the crack to be deleted in order to simulate crack healing or in case the crack was created incorrectly. Although these tear operators can be used to create branching and intersecting cracks, additional capabilities for handling the intersections, specifically the topological entities at the crack intersections, such as the common edge of a branch crack, are necessary to model the more complex crack topologies.

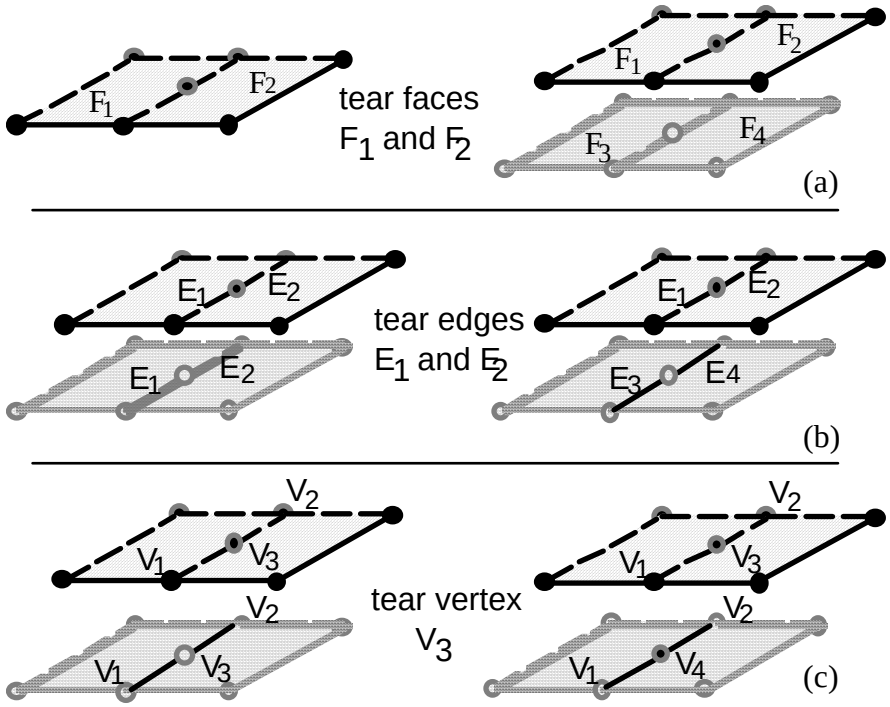


Figure 17. Figure 9. The *tear* operators are used in sequence to create a crack.

3.3.3 Some Crack Examples

Some examples of branching cracks and interface cracks are shown below. Not all of these can be created in FRANC3D as of yet, but will be in the near future. The examples include simple branching cracks (Figure 18), intersecting cracks (Figure 19 and 20), combination intersecting and branching cracks (Figure 21), and interface cracks (Figures 22 and 23), both in and across interfaces. For cracks that cut across material interfaces, the crack front is divided into two (or more) fronts, with the vertex at the interface defined as a crack tip. Edges must start or end at the interface, and defining the vertex at the interface as a crack tip allows the crack front to be traversed in sections. Branch cracks also have multiple crack fronts that must be treated separately during propagation.

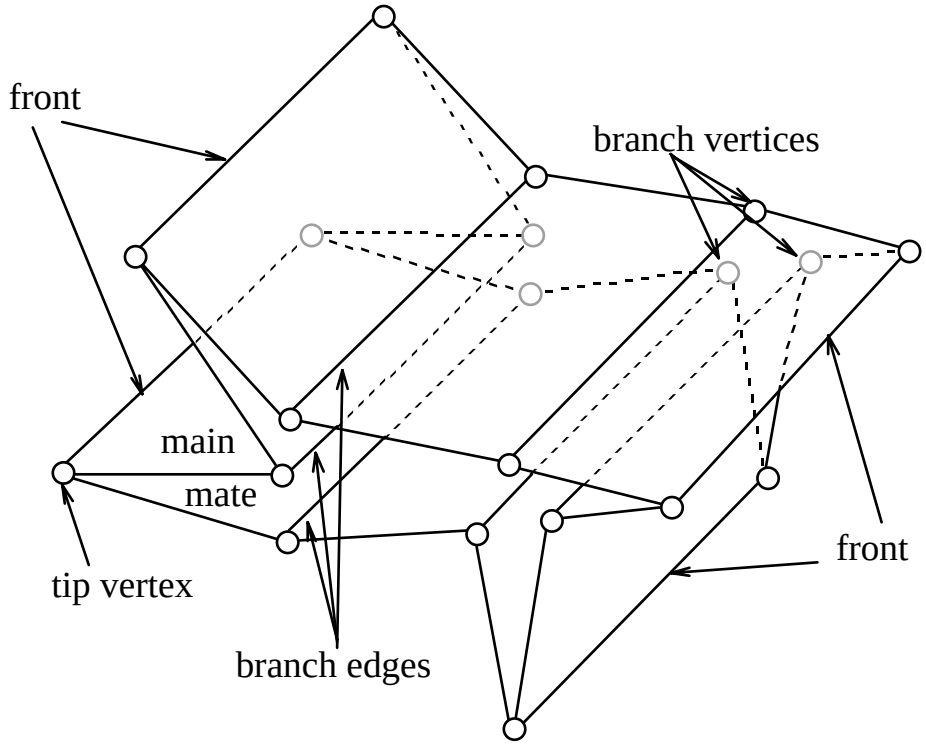


Figure 18. Topological representation of a typical 3D branch crack.

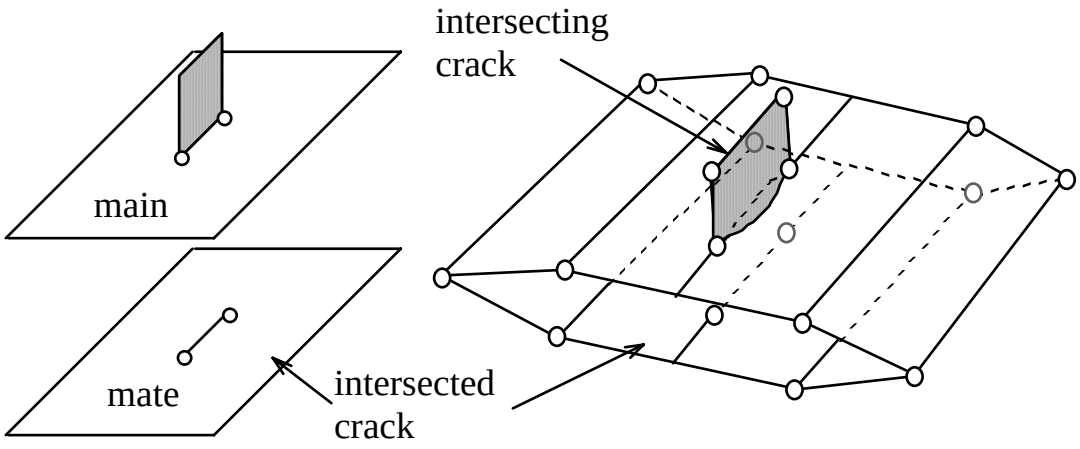


Figure 19. Topological representation of intersecting cracks: (a) before tearing the intersecting crack, (b) after tearing.

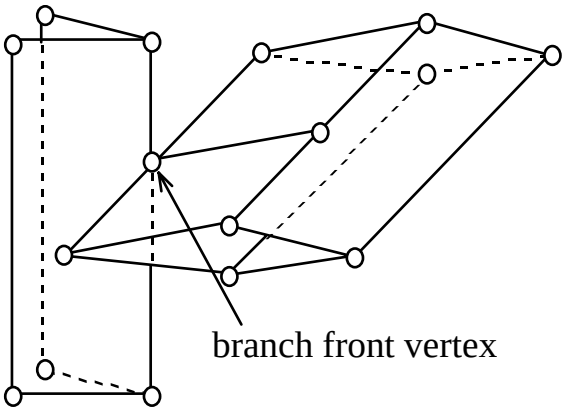


Figure 20. Topological representation of intersecting crack fronts beginning to cross.

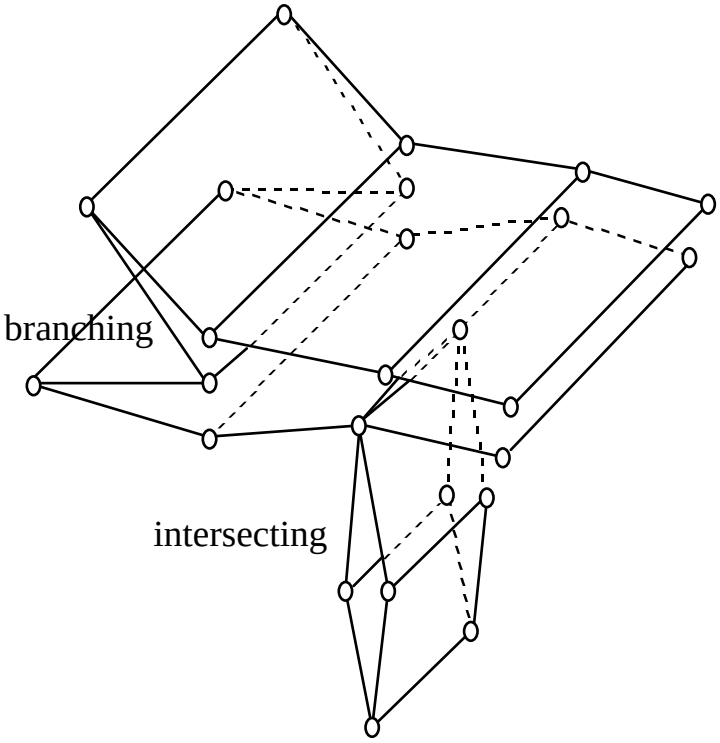


Figure 21. Topological representation of a combination of branching and intersecting crack.

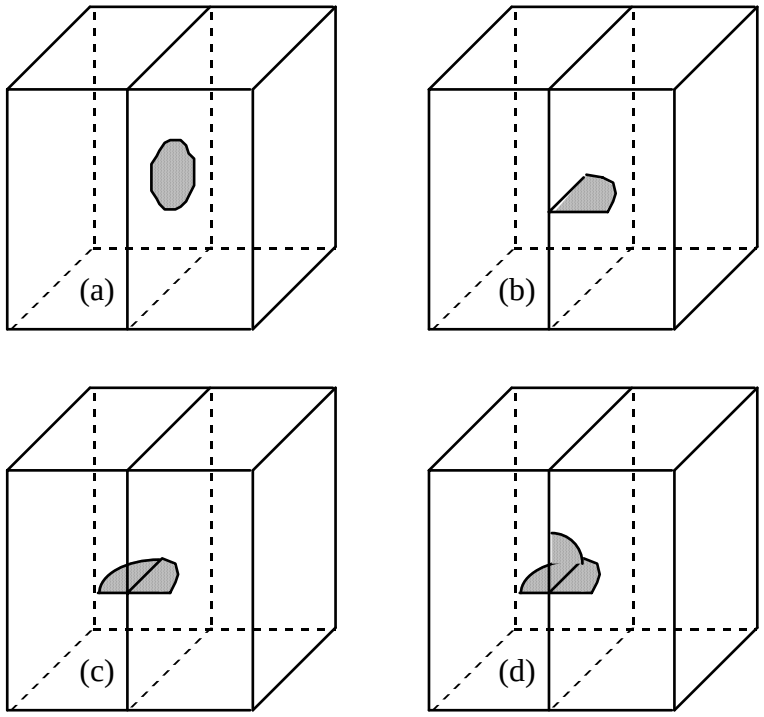


Figure 22. Topological representation of interface cracks, (a) within, (b) ending at, (c) across, and (d) combined in and across the interface.

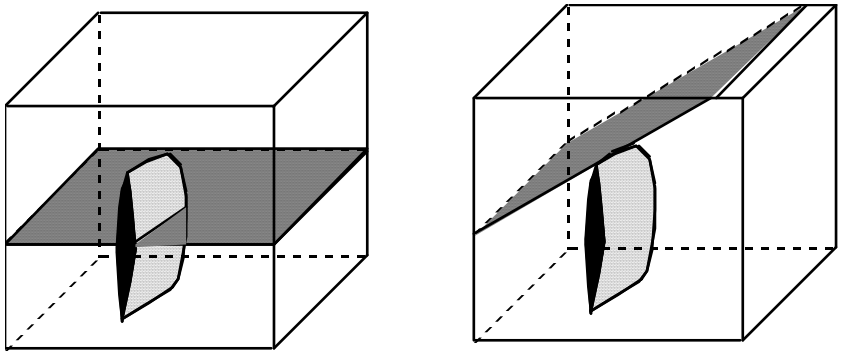


Figure 23. Topological representation of cracks crossing material interfaces.

4 Concepts Specific to the FRANC3D System

4.1 Surface Patch Geometry

FRANC3D reads model information from three file types: .dat, .fys ,and .afys. The usual FRANC3D file (extension .fys) is the machine-readable (binary) format that can be both written and read by FRANC3D. This file contains a complete description of both geometry and attributes defined during an interactive session of FRANC3D. An Ascii FRANC3D file (extension .afys) is an Ascii version of the .fys file. These two file types can be both read and written by FRANC3D, (Figure 24). The binary .fys files (a) use less disk space and (b) are read and written faster than the Ascii .afys files for the same model. However, they are not portable between different workstation platforms.

A FRANC3D geometry file (extension .dat) is a human readable (Ascii) file that can be read by FRANC3D at the time a new solid model is initiated. This file contains only geometric data for faces and cracks of the solid. The geometry file format can describe planar polygons, rectangular cubic patches, triangular cubic patches, and elliptical interior cracks. FRANC3D cannot write this format.

The geometry format is not a fully general way to describe complex models. It supports (a) both surface and interior cracks, and (b) interior faces separating multiple material regions. However, it does not support (a) edge or surface subdivision, (b) meshing, (c) material properties, or (d) attributes (boundary conditions). These items must be entered during subsequent interaction.

4.1.1 Reading Geometry Files into FRANC3D

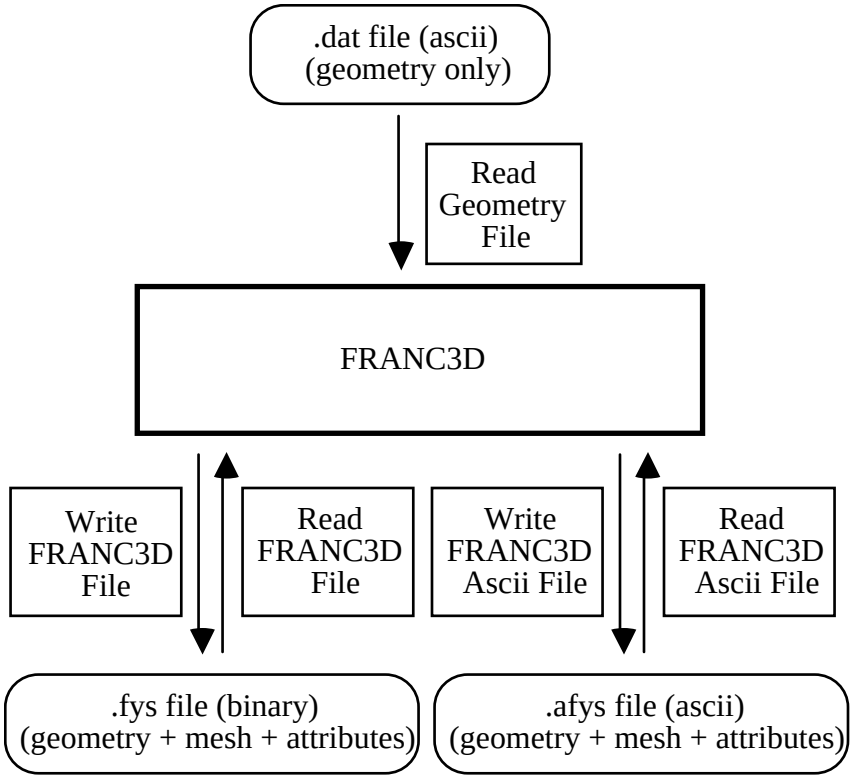
The user requests reading of a geometry file by selecting the **Read Geometry File** entry on the main menu cluster in FRANC3D. FRANC3D then presents a list of all files found on the current directory with extension .dat. After the user selects a file, FRANC3D deletes the current model and constructs a new model using face data from that file. Reading a .dat file always causes the prior model to be deleted and a new one created. It is not possible to read 'additional' geometry into an existing model.

4.1.2 Notes About Faces Entered via a .dat file

1) Faces should be arranged in the following order in the **.**dat file:
    a) the exterior shell of the overall solid;
    b) cracks;
    c) interior faces separating material regions.
2) Cracks may be constructed from any number of faces sharing edges. The crack geometry is entered only once, and FRANC3D replicates it for the opposite side. The user may choose either side of the crack as its 'main' face, but must use that orientation consistently when assembling multiple faces.

3) As the data file is read, FRANC3D makes geometric tests on the face itself and between the new face and prior geometry. The following error conditions are considered fatal and will cause the entire file to be rejected: a) non-coplanar points on a planar face, and b) two faces of the exterior shell oriented so that a common edge is traversed in the same direction on both faces. This is usually an indication that one of the faces is being viewed from the wrong side.

4) When the file reading is complete, a warning is issued if the faces fail to close and bound a solid. This error is sometimes caused by failure to identify intermediate vertices on long edges. For example, if two planar faces share the edges formed by the collinear vertex sequence b-c-d (see Figure 25), entry of point loops a-b-c-d-e and b-g-f-d (note the omission of c from the second face) will result in a 'hole' between the faces (Figure 25).



Note: .fys and .afys files contain equivalent data.
.afys file is portable between different computers,
.fys file is not portable

Figure 24. Geometry File I/O for FRANC3D.

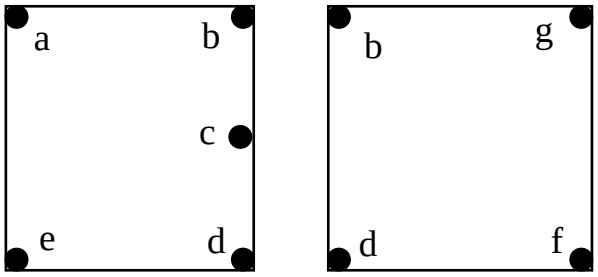


Figure 25. Failure to identify internal node.

4.1.3 Overall structure of Geometry File

The overall structure of a geometry file is:

<tolerance>
<name> <face type> <face data>
<name> <face type> <face data>
....
<name> -1

All data is entered in free format, with 1 or more blanks, tabs, and new lines separating each item.

<tolerance>
is the geometric distance to be used in comparing point coordinates to decide if two points are identical.

<name>
is a user-defined name. FRANC3D echoes all names to the terminal window as faces are read, but makes no further use of them.

<face type>
is a number indicating the type of face that follows. Its values are:

value	face type
1	planar polygon of exterior surface
2	quadrilateral cubic patch of exterior surface
3	triangular cubic patch of exterior surface
4	planar polygon of internal surface
5	quadrilateral cubic patch of internal surface
6	triangular cubic patch of internal surface
7	planar polygon of internal crack surface
8	quadrilateral cubic patch of internal crack surface
9	triangular cubic patch of internal crack surface

10 elliptical interior crack

End of file is indicated by a -1 in the face type position.

<face data>

depends on the face type, as described in following sections. For all faces, it is important to understand which side of the face is being viewed as the 'primary' side. For exterior faces, the 'primary' side of the face is the one seen from the **outside** of the solid. For cracks, the 'primary' side is the one seen when viewing the main crack face.

4.1.4 Planar Polygons

The face data for a planar polygon is (Figure 26):

<N> <xyz 1> ..... <xyz N>

where

N = number of points around face

<xyz 1> .... <xyz N> = xyz coordinates of the points.

For exterior polygons, the points must be entered in **CLOCKWISE** order when viewed from the primary side.

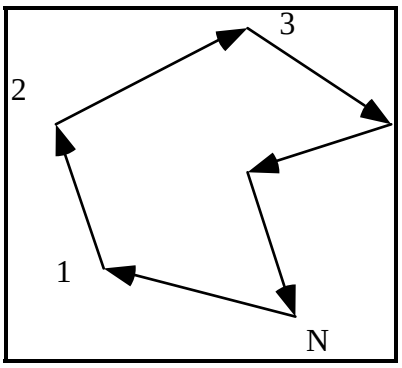


Figure 26. Point numbering for planar polygon viewed from outside the solid.

4.1.5 Quadrilateral Patches

A quadrilateral patch is defined by (m+1)*(n+1) points in a net. Points are organized numbering the net points with *integer* pairs (u,v), with net point (0,0) at the lower left and (m,n) a the upper right, the u-axis considered horizontal, and the v-axis considered vertical. Note that point numbering is from 0 to the upper limit in each direction (Figure 27).

FRANC3D constructs a cubic B-spline patch to interpolate the given coordinate data. The four corner points of the net are always entered into the model as vertices. The input may specify points along the edge of the net that are to be treated as vertices.

The face data for a rectangular patch is

<m> <n> <Nv>
<xyz 0 0> < xyz 1 0> ....< xyz m 0>
< xyz 0 1> ... .
... ...
< xyz 0 n> .... < xyz m n>
<u_1 v_1> .... <u_{Nv} v_{Nv}>

where

<m> = number of intervals in 'u' direction of patch
<n> = number of intervals in the 'v' direction of patch
<Nv> = number of (u,v) coordinates to appear (after the coordinate data) identifying points on the edge of the net that are to be treated as vertices in the FRANC3D model.
<xyz u v> = xyz coordinates of net point u v
<u_i v_j> = integer net point coordinate to be treated as a vertex.

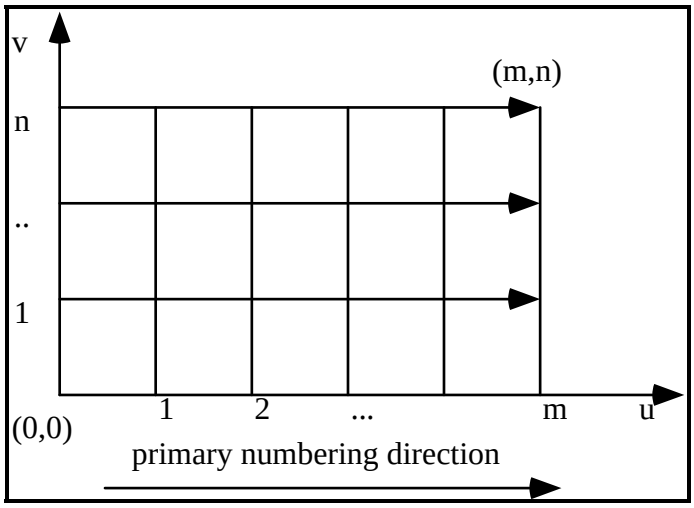


Figure 27. Point numbering for B-spline patch viewed from outside the solid.

4.1.6 Triangular Patches

A triangular patch is defined by ten points in a triangular net. Points are numbered as below, viewing the primary side of the face, Figure 28. FRANC3D constructs a tri-cubic Bezier patch to interpolate the given coordinate data. The three corner points of the net are always entered into the model as vertices. The input may specify additional points along the edge of the net that are to be treated as vertices.

The face data for a triangular patch is

<Nv>
<xyz 0 > < xyz 1> ....< xyz 9>
<I_1> < I_{Nv}>

where

<Nv> = number of net points ids to appear (after the coordinate data) identifying net points that are to be treated as vertices in the model.
<xyz> = xyz coordinates of net point
<I> = integer net point coordinate to be treated as a vertex.

4.1.7 Elliptical Internal Crack

An elliptical internal crack is defined by its center point and four major/ minor axis endpoints (Figure 29).

The face data for crack is

<xyz center>
<xyz 1>
<xyz 2>
<xyz 3>
<xyz 4>

where the points are located as shown in Figure 29.

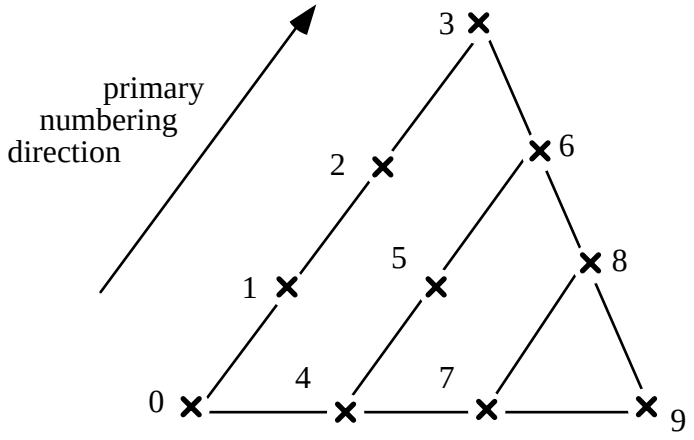


Figure 28. Point numbering for triangular cubic patch viewed from outside the solid.

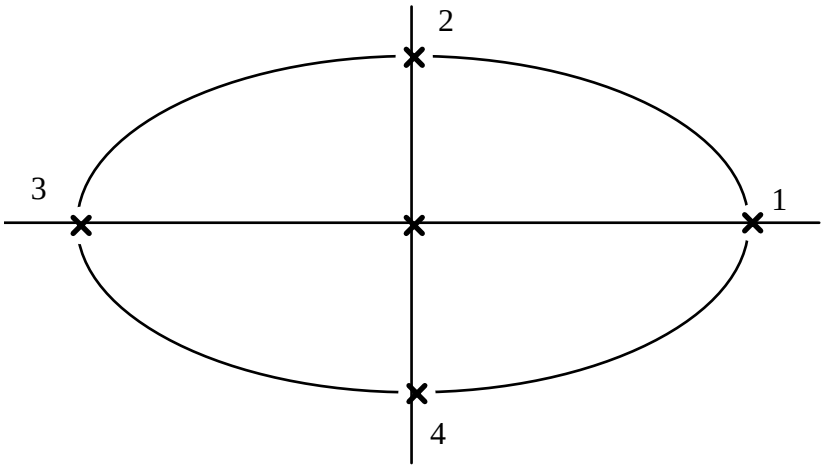


Figure 29. Point numbering for internal elliptical crack.

4.2 Discretization and Meshing

The concept of model hierarchy was explained in Section 3.2.1. Also, the concept of local damage during crack growth and remeshing was discussed in Section 3.2.7. The ideas that have not been discussed yet are the control of the mesh by edge subdivisions and the meshing algorithms themselves. These ideas are discussed below with particular emphasis on meshing cracked structures.

Meshes can be constructed of either triangular or quadrilateral elements, or a combination of both. If quadrilateral elements are chosen, but the surface cannot be meshed entirely with quadrilateral shaped elements, triangles are used. If triangular elements are chosen, all elements will be triangular. The pattern that is to be used when forming the triangles can also be specified.

4.2.1 Edge Subdivision

Before surfaces can be meshed, edges must be subdivided into line segments. The nodes of the edge subdivision subsequently become nodes of the mesh. The density of edge subdivision points therefore controls the mesh density. The number of subdivisions can be increased and the ratio changed on the edges to increase the number of elements for a specific location on the face. Subdividing the faces prior to subdividng the edges can also help to produce a reasonable mesh. Midside nodes of the elements are not explicitly defined. Rather, they can be generated when writing the analysis files.

In general, a larger number of smaller elements are required at points of singularity. For instance, at crack tips and along the crack front, an increased number of smaller elements will produce more accurate results. It is a good idea to create a row of uniformly sized quadrilateral shaped elements along the crack front. For shell cracks, a template of crack tip elements can be generated as well.

4.2.2 Mapped Meshing

A mapped mesh is one in which a specific regular mesh - e.g. an *m* by *n* grid over a four sided region - is constructed on a surface. Mapping can only be applied if (a) the region can be treated as if it has exactly three or four sides (some of which may contain several actual edges chained together), and (b) the number of nodes along the various sides agrees with a mapping method. When these conditions apply, mapping produces smooth meshes of well-shaped elements. Mapping may not be applicable in transition regions.

Mapped meshing requires that key nodes on a face be identified as corner points of the mapping. Each mapped meshing command begins by prompting for a choice between (a) user-specified corner points or (b) automatic with no corner points specified. If the user-specified corner points option is chosen, FRANC3D issues one prompt for a single face, followed immediately by (depending on the mapping) three or four prompts to identify key corner points. If the "no corner points" option is chosen, FRANC3D prompts only for a face touch. FRANC3D examines the face and selects the key points.

There are three schemes for generating mapped meshes. These include bi-linear, bi-linear collapsed and tri-linear. When meshing a face using the bi-linear transfinite mapping algorithm, the face to be meshed must be decomposable into four super-edges such that opposing super-edges have the same number of subdivisions. A regular grid of quadrilateral shaped elements is generated unless the user has chosen to use all triangles; then the quadrilaterals are split into triangles. When meshing a face using the collapsed bi-linear transfinite mapping algorithm, the face to be meshed must be decomposable into three super-edges such that two of the super-edges have the same number of subdivisions. If two super-edges have the same number of vertices and the other super-edge has a different number of vertices, the triangular elements will emanate from the vertex at which the two super-edges with the same number of vertices connect. When meshing a face using the tri-linear transfinite mapping algorithm, the face to be meshed must be decomposable into three super-edges such that all super-edges have the same number of vertices. In both bi-linear collapsed and tri-linear meshing, the face is split into both quadrilateral and triangular shaped elements (unless the user has chosen to use all triangles).

4.2.3 Arbitrary Region Surface Meshing

The arbitrary surface region meshing algorithm allows a face of arbitrary shape to be meshed using either all triangular or all quadrilateral shape elements. A complete description of the meshing algorithm is available in Potyondy (1993). Additional internal points can be defined or automatically generated to produce the mesh. A number of parameters can be set in a dialogue box (Figure 30) to control the mesh density and appearance. The options are explained as the meshing algorithm is discussed. The steps in meshing a region with triangular elements are discussed below. A quadrilateral element mesh involves similar steps.

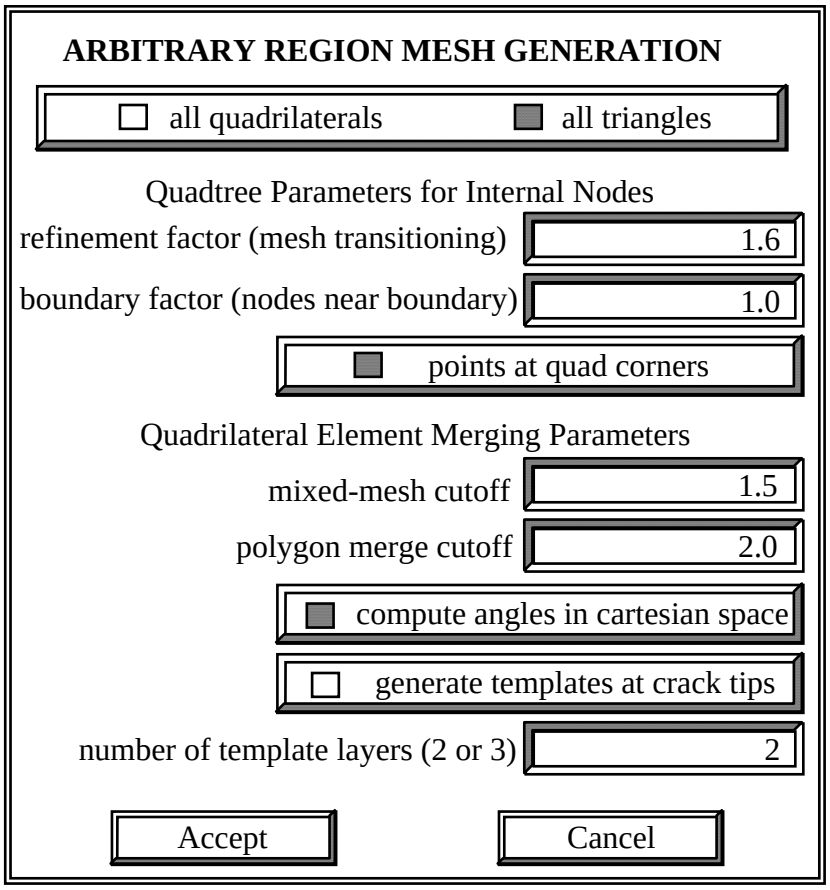


Figure 30. Dialogue box for the arbitrary surface region meshing algorithm.

Internal nodes are generated for the region to be meshed. This is done by decomposing the region using a quadtree recursive spatial decomposition (RSD) procedure. RSD algorithms act upon a region of space and subdivide the region into a number of smaller regions of a similar shape. The process is repeated an arbitrary number of times for each of the smaller regions. Figure 31 shows two examples of RSD procedures. In the first, a triangular region is subdivided into four similar triangular regions recursively. In the second, a rectangular region is subdivided into four similar rectangular regions recursively. RSD procedures only work for a small number of region shapes, in two-dimensions these include triangles, rectangles, and hexagons. For a rectangular region, every subregion is either undivided or divided into four similar regions. This information can be stored conveniently in a tree data structure, where each node in the tree has exactly zero or four children. Such a data structure is called a *quadtree*. Figure 32 shows a simple example of a divided region and its corresponding quadtree. The undivided regions, which correspond to leaf nodes in the tree, are called terminal quadrants. The size of a terminal quadrant can be determined from its level in the tree.

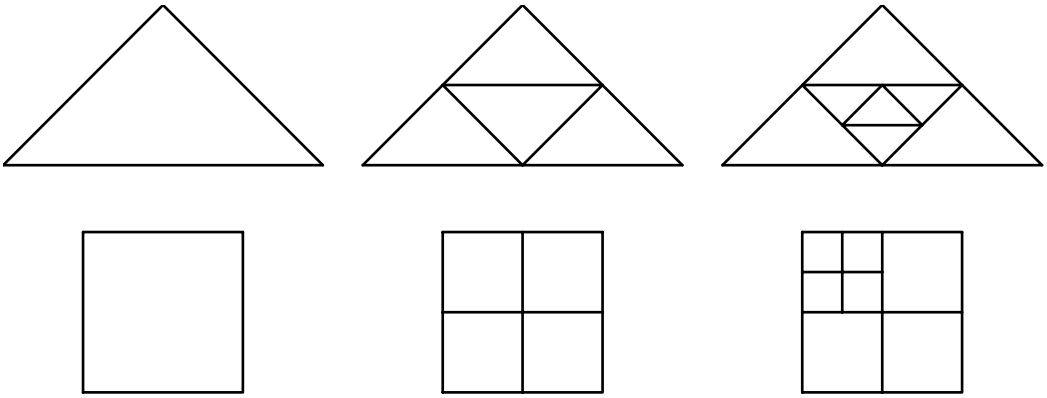


Figure 31. A triangular and a rectangular recursive subdivision procedure.

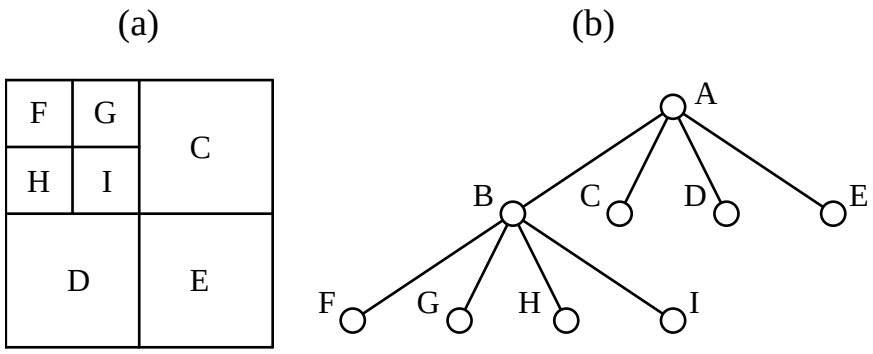


Figure 32. A decomposed region and the corresponding quadtree.
Terminal quadrants are labeled in (a).

In the meshing algorithm, the level of decomposition is a function of the nodal spacing on the nearby boundaries. The refinement factor then controls how quickly the level of decomposition transitions towards the interior of the region. A refinement factor of one indicates that the decomposition can only change by one level for adjacent quadtree cells. The boundary factor controls the placement of internal points near the boundaries. A small value allows points to be placed close to the boundary. Internal nodes can be generated either at the center or the corners of the terminal quadtree cells if the location is sufficiently far from the boundary (Figure 33).

A mesh comprised of triangles is generated using a boundary contraction scheme (Figure 34). A list of all the boundary edges is created. Starting with the first edge in the list, all the internal and boundary nodes are tested to determine which node produces the largest included angle when edges are extended from the two ends of the edge to the node. An additional parameter allows the user to decide whether to calculate the included angle in parametric or Cartesian space, usually Cartesian space gives the best elements. The best vertex is then used to create a triangular element. The list of boundary edges is updated, removing the edge that was started with and adding the new edges if they are not part of

an existing element. The region, therefore, contracts by extracting elements one at a time.

The mesh is smoothed by repositioning each internal node to lie at the centroid of its surrounding polygon (Figure 35). The smoothing uses an iterative approach in which each internal node is repositioned based on the current nodal positions of its surrounding polygon. This process is repeated until there is no change of nodal positions between iterations. In practice, no significant changes in nodal positions are observed after a few iterations.

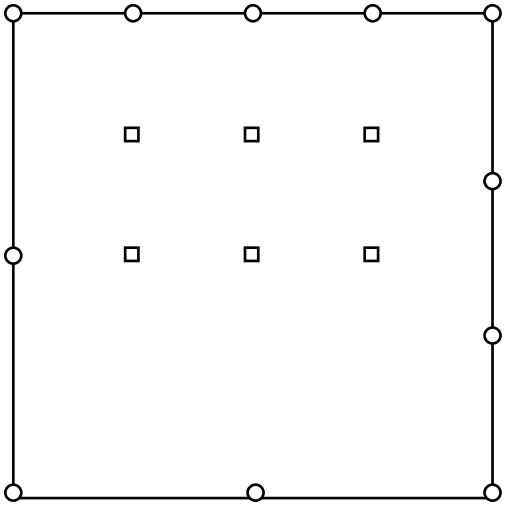


Figure 33. Generated interior points using a quadtree procedure.

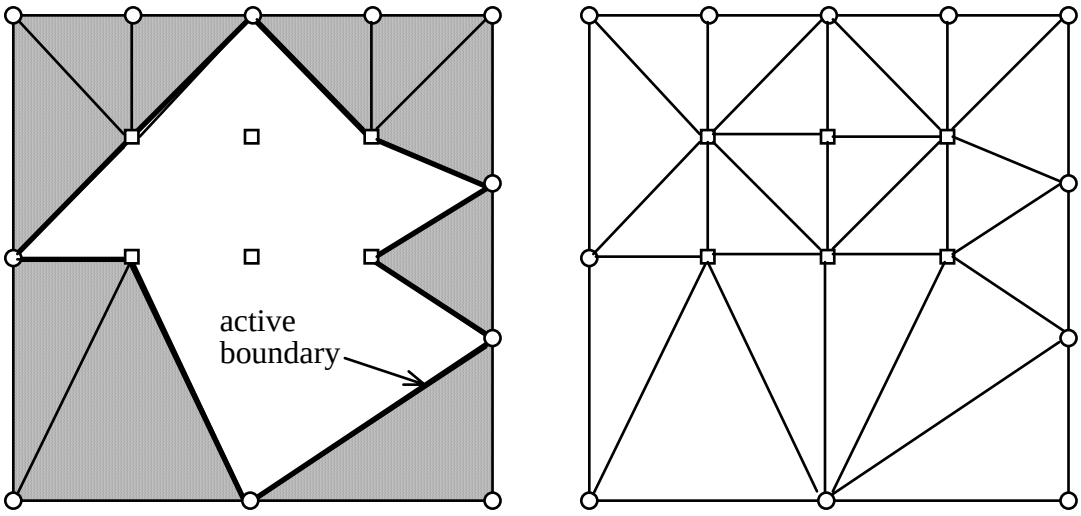
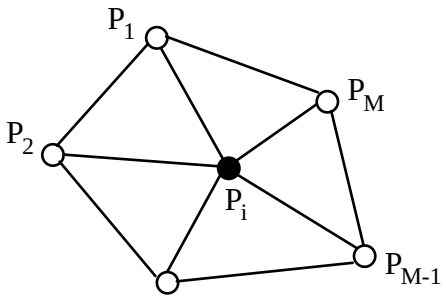


Figure 34. Generated triangular element mesh using a boundary contraction procedure.

For a quadrilateral element mesh, there are two extra parameters, mixed mesh cutoff and polygon merge cutoff that control the shape of the elements that are formed. Consult

Potyondy (1994) for more details. In addition, templates of quadrilateral crack tip elements can be generated at shell crack tips, either two or three layers. These elements help in obtaining more accurate stress intensity factors for shell cracks. Again, consult Potyondy (1994) for more details.



$$P_i = \frac{1}{M}\sum_{m=1}^{M} P_m$$

M = number of nodes element-adjacent to node i

Figure 35. The mesh is smoothed by adjusting the position of each internal node P_i, so that it lies at the centroid of the polygon formed by its adjacent elements.

4.3 Simulation Attributes

Simulation attributes can be organized into the following five categories: material properties, geometric properties, analysis control information, discretization control information, and boundary conditions. The *material properties* are inherent physical properties of the material being analyzed. Elastic constants are examples of material properties. The *geometric properties* consist of geometric information about the idealization which cannot be derived from the geometric model itself. For example, in a shell idealization, the shell thickness is a geometric property, since in the geometric model the shell is stored as a face with an associated surface geometry -- by definition, the surface geometry lacks a thickness component. In a beam idealization, the beam cross section is a geometric property. The *analysis control information* controls the generation of the analysis model. The type of analysis, either finite or boundary element, linear or geometrically nonlinear, as well as mesh element type, are examples of analysis control information. The *discretization control information* controls the mesh generation algorithms. Desired mesh density, mesh transitioning, and mesh topology (triangular or quadrilateral, brick or wedge, etc.) are examples of discretization control information. The *boundary conditions* are constraints on motion and applied loads. Specified fixities and forces are examples of boundary conditions.

All simulation attributes are attached to unique topological entities of the FRANC3D Geometry Model. Each node or element of the computational mesh can find its parent Geometry Model entity and can thus inherit this information.

4.3.1 Material Properties

Material properties allowed in Version 1 consist of the linear elastic constants, elastic modulus and Poisson's ratio, along with the fracture toughness and the density. These properties are attached to regions and faces of the Geometry Model. The regions are used for a three-dimensional continuum and the faces for a shell idealization.

4.3.2 Geometric Properties

Geometric properties are necessary only for shell structures and consist of shell thickness and possible layering information. These properties are attached to faces of the Geometry Model. The information consists of a list of layers. Each layer has a thickness and an offset from a reference surface location. It is assumed that the thickness of each layer is constant.

4.3.3 Discretization Control Information

Discretization control information is used by the mesh generation algorithms to determine element density and mesh topology. This information is stored via the hierarchy of models which provide a framework to aid in its interactive specification.

4.3.4 Boundary Conditions

All boundary conditions are attached to topological faces, edges, and vertices of the Geometry Model. They are specified by providing either a kinematic or a force/traction value for each of three orthogonal directions (in terms of either a local or a global coordinate system), along with a description of the distribution of the value over the topological entity.

For three-dimensional solid analysis, the specified kinematic value corresponds to a displacement and the force value corresponds to a traction or pressure on a surface or a force at a point or along an edge. For shell analysis, an additional kinematic value of rotation and force value of moment is provided.

The distribution of a specified value is either uniform or arbitrarily varying. The mesh representation (MRP) objects, described in the next section, are used to associate arbitrarily varying vector field information with Geometry Model faces and edges.

The global coordinate system remains fixed and corresponds to the Cartesian axes of the modeling space in which the geometric model is defined. Each Geometry Model edge and face has its own local coordinate system consisting of an orthogonal set of axes whose orientation with respect to the global axes may vary over the edge or face. The analyst can both view and alter the orientations of each local coordinate system. The local coordinate systems are a necessity for shell analysis in which loadings are applied to shell edges that are not aligned with the global system and for response visualization of surface-related quantities, e.g., stress resultants.

The local coordinate system Φ_e, for an edge that is parametrized by *s*, is given by

$$\Phi_e(s) = \begin{Bmatrix} \mathbf{n}(s) \\ \mathbf{t}(s) \\ \mathbf{p}(s) \end{Bmatrix} \quad ; \quad \mathbf{n} \times \mathbf{t} = \mathbf{p}$$

where **t** is tangent to the edge, and **n** is normal to a face adjacent to the edge. **t** is obtained from the geometric description of the edge along with a specification of edge direction. **n** is obtained from the geometric description of the adjacent face.

The local coordinate system Φ_f, for a face that is parametrized by (u, v), is given by

$$\Phi_f(u,v) = \begin{Bmatrix} \mathbf{u}(u,v) \\ \mathbf{v}(u,v) \\ \mathbf{n}(u,v) \end{Bmatrix} \quad ; \quad \mathbf{n} \times \mathbf{u} = \mathbf{v}$$

where **u** and **v** lie in the face tangent plane, and **n** is normal to the face. **n** is obtained from the geometric description of the face along with a specification of normal direction. **u** is obtained from the geometric description of the face along with a specification of the angle between **u** and one of the surface parametric coordinate directions.

4.3.5 Mesh Representation of Boundary Conditions

A procedure is described next for associating arbitrarily varying vector field information with geometric surface patches in FRANC3D. For each surface patch, there exists a two-parameter coordinate system, u, v. Each point on the surface is associated with a specific value of the vector field. Note that each point can be described either in terms of its Cartesian x, y, z coordinates or in terms of its surface u, v coordinates. A mapping function, whose input consists of the surface point coordinates and whose output consists of the value of the vector field at that point, is required. The mapping function may be either mathematically explicit or computationally implicit. In FRANC3D, a computationally implicit function provides an algorithm for determining the field value.

The vector field is given in terms of a finite or boundary element mesh and field values are provided at discrete points. The mesh represents both the field variation and the geometry of the underlying surface. This mesh might not have been generated from the geometric model; thus, its geometric description of the surface may not correspond exactly with that of the geometric model. For example, the user can define a mesh consisting of one planar element to define the boundary conditions for a set of non-planar geometric surfaces. If both the geometric model and the mesh approximate the same underlying surface, then for each point on the surface of the geometric model there will be a unique corresponding point within the surface of the mesh. The corresponding point will be defined to be the point on the mesh which is the closest (in Cartesian space) to the point on the geometric model. Note that these points may not be coincident because of the differing approximations of the underlying surface by each model.

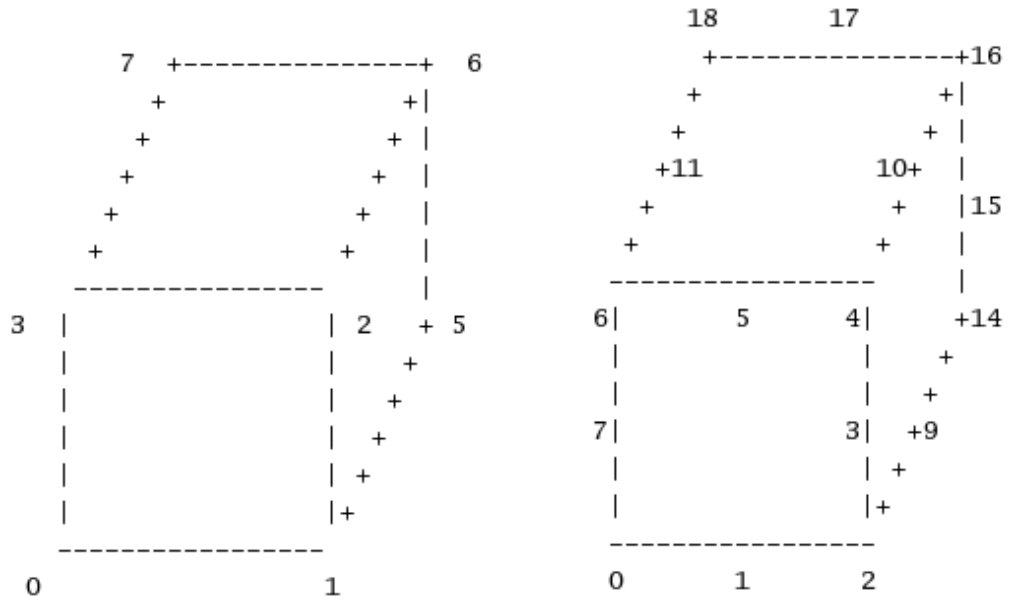
Response information is associated with individual faces of the Geometry Model in the form of a surface mesh. All of the elements of the mesh that are associated with a particular Geometry Model face are collected together into a mesh representation (MRP). The MRP returns the field value in response to a query that specifies the point at which the field value is desired. Both non-linear search algorithms for mapping between spaces, and interpolation algorithms for determining the response value at any point of the surface mesh form part of the process.

The data and file format of the MRP are described below. Some assumptions and restrictions are noted first:

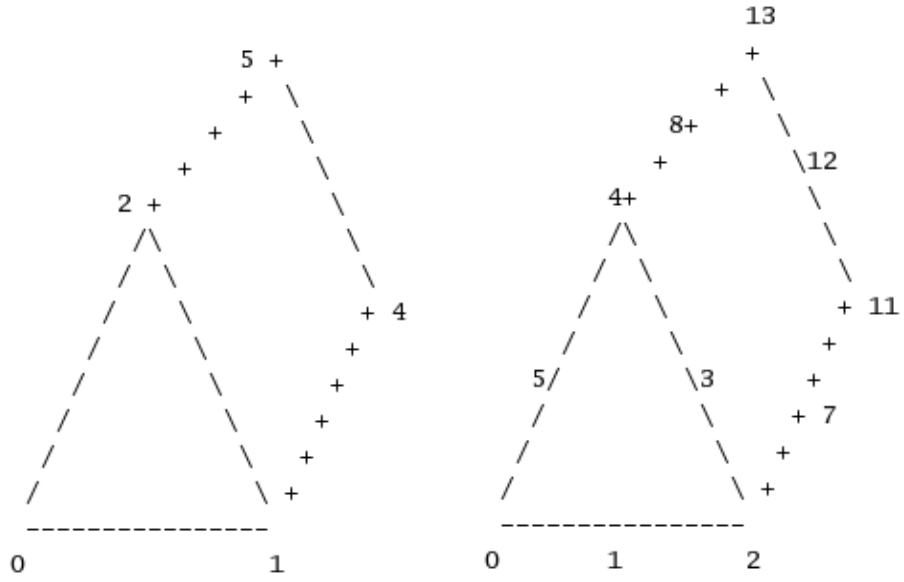
1. Edge (1D), surface (2D), and volume (3D) elements are possible, but volume elements are not implemented.
2. Both the geometry and the field variation are interpolated using the same functional form given by the element type.
3. Intra-element connectivity is not monitored; it is assumed that the user has insured validity The MRP is essentially a collection of untied elements.
4. A unique integer numbering of nodes such that all node numbers are greater than or equal to zero is required. The nodes need not be numbered consecutively.
5. A unique integer numbering of elements such that all element numbers are greater than or equal to zero is required. The elements need not be numbered consecutively.
6. Nodal field values are stored uniquely.
7. Element field values are stored uniquely for each element. The current implementation stores one element field value vector for num_spots on the element; each spot is uniquely associated with a node on that element.

The element types are defined as:

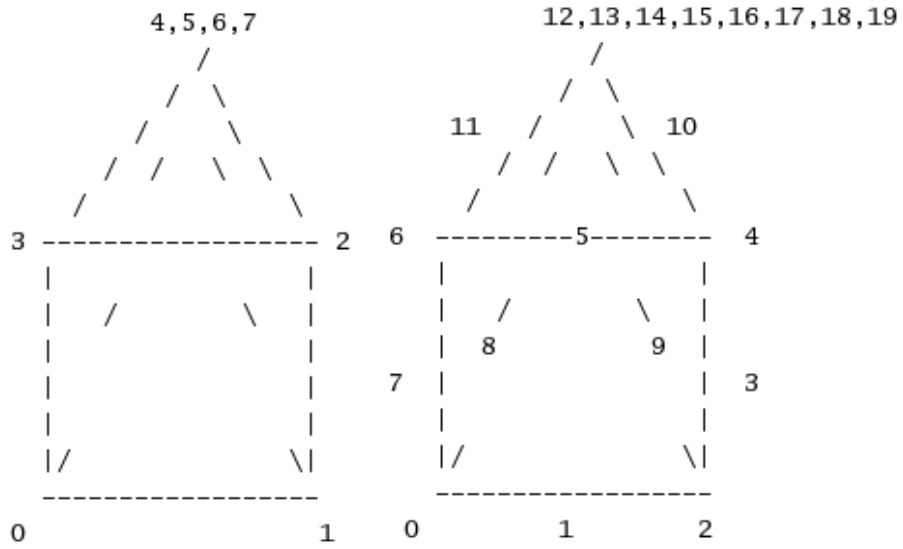
1 => 2-noded, linear edge
2 => 3-noded, linear triangle
3 => 4-noded, linear quadrilateral
4 => 6-noded, quadratic triangle
5 => 8-noded, quadratic quadrilateral
9 => 8-noded, linear brick
10 => 20-noded, quadratic brick
12 => 6-noded, linear wedge
13 => 15-noded, quadratic wedge
15 => 5-noded, linear pyramid
16 => 13-noded, quadratic pyramid
18 => 4-noded, linear tetrahedron
19 => 10-noded, quadratic tetrahedron



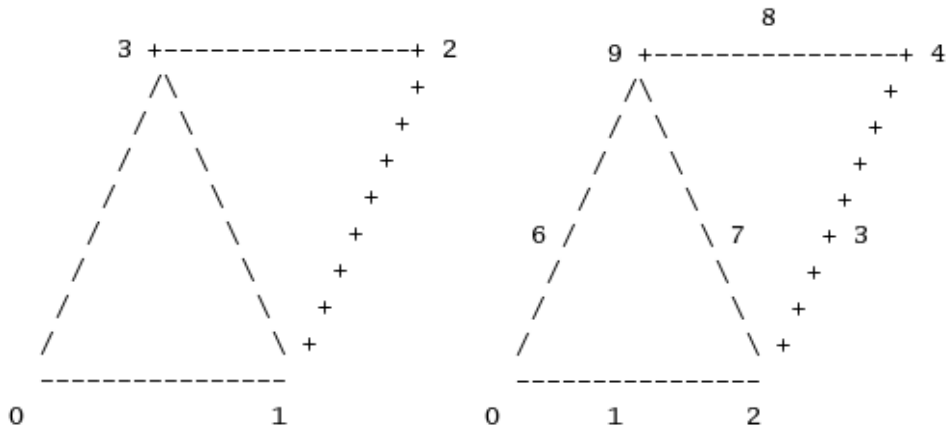
brick elements: 8 and 20-noded



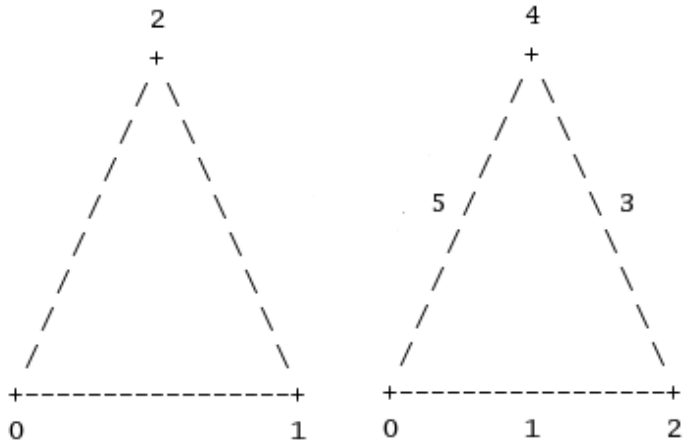
wedge elements: 6 and 15-noded



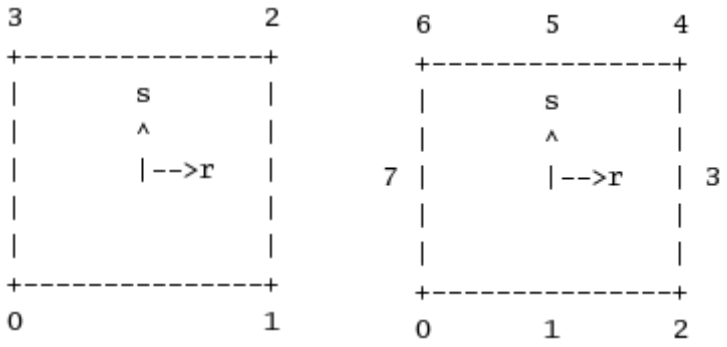
pyramid elements: 5 and 13-noded



tetrahedral elements: 4 and 10-noded



triangular elements: 3 and 6-noded



quadrilateral elements: 4 and 8-noded

Figure 36. Element node numberings.

The node numbers and element numbers must all be greater than or equal to zero, but need not start at zero. The data must exist in an Ascii text file with the following record structure:

Record Type 1: Control information

There is always only one record of this type.

```
%+  title  [tol=tolerance]
ne   nn   nfvn   efv_flag   nfve   ncrkelem   flag
```

title	(On a separate line, it must start with "%+".)
ne	(Number of elements in mesh.)
nn	(Number of nodes in mesh.)
nfvn	(Number of field values per node.)
nfve	(Number of field values per node per element.)
ncrkelem	(Number of elements near the crack.)
flag	(Interpolate=0 or extrapolate=1.)

Note: tolerance value can be specified in title string.
Note: interpolate versus extrapolate allows the user to specify surface boundary conditions based on volume elements.

Record Type 2: Element information

There is one record of this type for each element. For surface elements, nodal ordering must be ccw. For line elements, nodal ordering must be tail-to-tip.

enum	(Element number.)
matnum	(Material number.)
etype	(Element type code.)

nnodes (Number of nodes in element)

node_1 (element node 1 number)
... " " "
node_n (element node n number, where n = nnodes for this element)

Record Type 3: Nodal coordinates

There is one record of this type for each node

nnum (Node number.)
coord_x,y,z (x, y, and z coordinates.)

Record Type 4: Nodal field value codes

There is one record of this type.

code_i (Nodal field value code i, where i =1 to nfvn.)

Record Type 5: Nodal field values

There is one record of this type for each node.

nnum (Node number.)
real_resp (Flag indicating valid response data. Set to 0 or 1 if valid.)
nfv_i (Nodal field value for node i, where i =1 to nfvn.)

Record Type 6: Element field value codes

There is one record of this type.

code_i (Element field value code i, where i =1 to nfve.)

Record Type 7 : Element field values

There is one record of this type for each element.

enum (Element number.)
nne (Number of nodes in element.)
real_resp (Flag indicating valid response data. Set to 0 or 1 if valid.)
efv_i_j (Nodal field value i,j where i = 1 to nne and j=1 to nfve.)
(The node field value order must match the element definition.)

Record Type 8 : Crack elements

There is one record of this type.

list_enum (List of element numbers.)

list_crk_flag (List of adjacency flags; main=3 or other side=4.)

An example input file follows for the configuration of surface (a 4-noded quadrilateral and a 3-noded triangular) elements shown in Figure 36b. The nodal field values are FORCE_X, FORCE_Y, FORCE_Z and are set to 99.0. The element field values are FORCE_X, FORCE_Z and are set to 99.0 and -99.0.

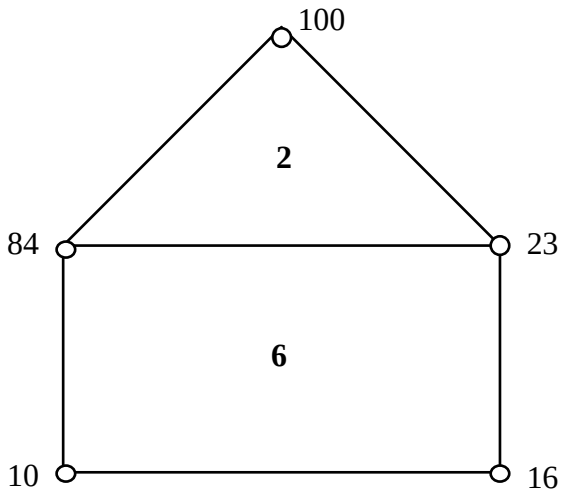


Figure 36b. Example mesh representation.

```
%+ Example mrp input file [tol=0.001]
2 5 3 2 0 1

2  1  2  3  100 84 23
6  1  3  4  84 10 16 23

16    2.0    0.0    0.0
84    0.0    2.0    0.0
100   1.0    4.0    0.0
10    0.0    0.0    0.0
23    2.0    2.0    0.0

10  11  12

100   1  99.0    99.0    99.0
23    1  99.0    99.0    99.0
84    1  99.0    99.0    99.0
10    1  99.0    99.0    99.0
16    1  99.0    99.0    99.0

10  12

2  3  1  99.0  -99.0  99.0  -99.0  99.0  -99.0
6  4  1  99.0  -99.0  99.0  -99.0  99.0  -99.0  99.0  -99.0
```

MRP data can be entered in FRANC3D using the interactive dialog boxes. The MRP files are created by OSM when translating 3D finite element files also.

The full set of response value codes are:

DISPL_X 1
DISPL_Y 2
DISPL_Z 3

ROT_X 4
ROT_Y 5
ROT_Z 6

TRACT_X 7
TRACT_Y 8
TRACT_Z 9

FORCE_X 10
FORCE_Y 11
FORCE_Z 12

MOMENT_X 13
MOMENT_Y 14
MOMENT_Z 15

SIGMA_XX 20
SIGMA_YY 21
SIGMA_ZZ 22
SIGMA_XY 23
SIGMA_XZ 24
SIGMA_YZ 25

SIGMA_1 30
SIGMA_2 31
SIGMA_3 32

STR_ENGY_DNSTY 40
TAU_MAX 41
TRACT_NORM 42
TRACT_SHEAR_PARL 43
TRACT_SHEAR_PERP 44

RES_NUU 50
RES_NVV 51
RES_NUV 52
RES_MUU 53
RES_MVV 54
RES_MUV 55

RES_QU 56
RES_QV 57

EFF_SR 60
SIGMA_UU_TOP 61
SIGMA_UU_BTM 62
SIGMA_VV_TOP 63
SIGMA_VV_BTM 64

EPSILON_XX 70
EPSILON_YY 71
EPSILON_ZZ 72
EPSILON_XY 73
EPSILON_XZ 74
EPSILON_YZ 75

EPSILON_1 80
EPSILON_2 81
EPSILON_3 82

COD 90
CSD 91
CTD 92

X_COORD 100
Y_COORD 101
Z_COORD 102

TEMPERATURE 200

4.3.6 Model Boundary Conditions (Interpolating FEM Results)

Model boundary conditions include accelerations and rotations as well as results mapped from other finite element results. In version 1.15, crack face tractions were applied using model boundary conditions; the user entered a .conn and .fstr file name. The .conn and .fstr files have been replaced by the .mrp file and crack face tractions must be applied as face boundary conditions. Model boundary conditions are reserved for such things as nodal temperatures. Note that nodal stresses can be applied as model boundary conditions, but stresses will be applied to every node in the model, not just the crack face.

4.4 Methodology of 3D Crack Propagation

Three dimensional crack propagation requires the coupling of the physics and mathematics that govern the propagation of multiple, non-planar, interacting cracks with the topological and geometric representation in FRANC3D. FRANC3D is capable of creating multiple, non-planar cracks. However, the physics and mathematics of arbitrary,

non-planar 3D crack propagation is not understood very well. The current technique that is implemented in FRANC3D for propagating 3D cracks is described below.

Currently in FRANC3D, the stress intensity factors are calculated using a displacement correlation technique. Propagation direction is evaluated at discrete points along the crack front using 2D, plane strain equations. The crack growth increment is calculated using simple expressions such as Paris' model with a user-supplied maximum extension. The effects of free surfaces, interfaces, and neighboring cracks are taken into account only through the relative displacements. The user should determine the effects of these interactions on the propagation. The main problems associated with the numerical simulation of the propagation of non-planar 3D cracks can be summarized then as:

1) accurate extraction of stress intensity factors along an arbitrary 3D crack front,
2) determination of the direction of extension and the crack growth increment based on the above stress intensity factors,
3) the effect of neighboring features such as free boundaries, other cracks, and material interfaces on the stress intensity factors and the propagation.

The three modes of fracture propagation include opening, sliding and tearing modes, called mode I, II and III respectively (Irwin, 1960). The stress intensity factors, given the symbols K_I, K_{II} and K_{III} for mode I, II and III respectively, define the stress field at the crack front for a given structural geometry and boundary conditions. Pure mode I crack growth causes self-similar crack growth (Figure 37a), essentially producing a planar crack that is perpendicular to the maximum tension (tension is positive). Mode II, in-plane sliding, produces tilting of the crack front (Figure 37b), and mode III, out-of-plane tearing, produces twisting of the crack front (Figure 37c).

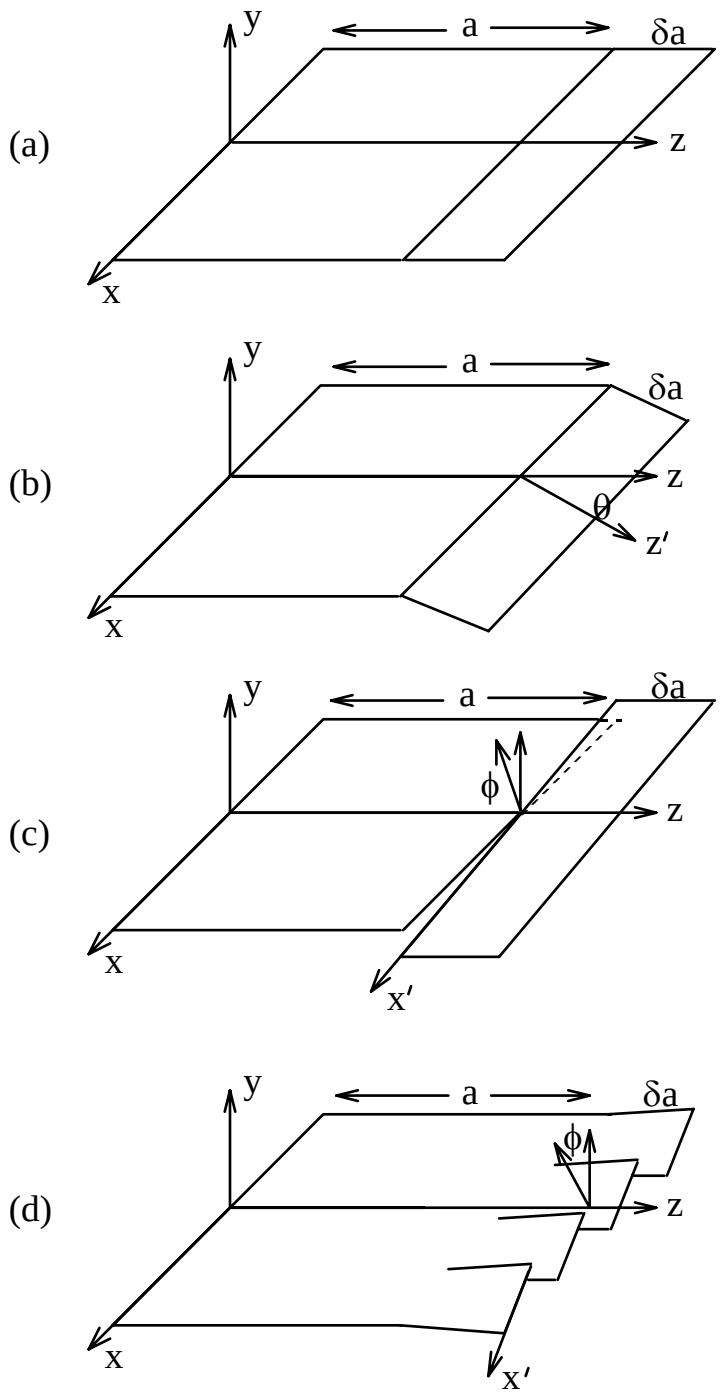


Figure 37. The three modes of fracture. Mode I (a), Mode II (b), and Mode III (c and d). The crack front cannot twist over the entire length (c), thus, it splits into segments (d).

There are a number of methods for extracting the stress intensity factors from a numerical simulation of crack growth; FRANC3D uses displacement correlation. Displacement correlation is a simple technique and is easy to implement for non-planar crack geometries. The displacement extrapolation technique is available, but it has been shown to give erratic results (Lim *et al.*, 1992). The displacements on the crack surface – opening, sliding and tearing – are directly related to the three stress intensity factors. The displacements are based on interpolated values calculated at points which are a measured distance from the geometric crack front (Figure 38).

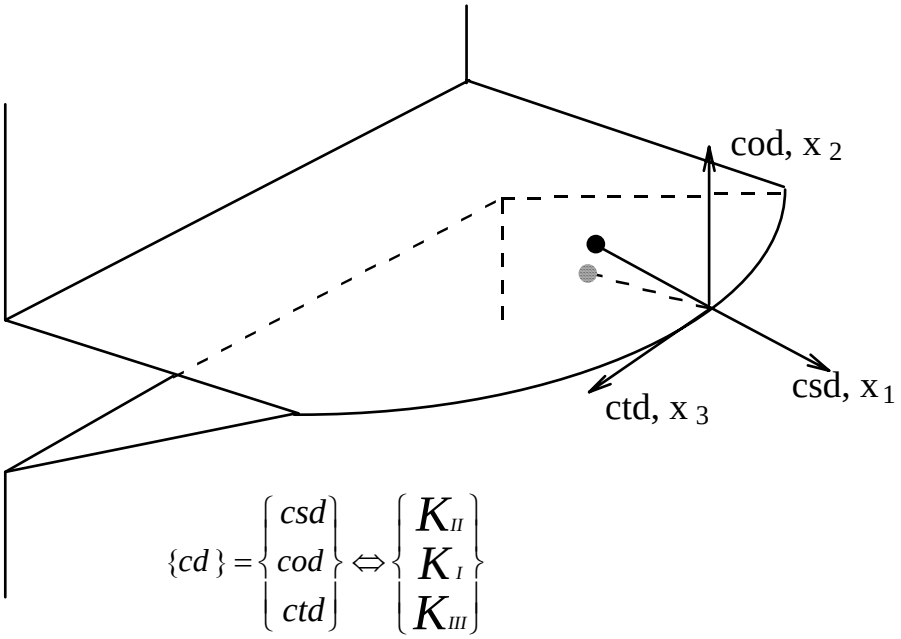


Figure 38. Crack displacements calculated at a measured distance from the crack front are directly related to the stress intensity factors at the crack front.

Mixed-mode I and II crack growth has been studied in 2D for many years, leading to several models describing the direction and amount of crack extension. These include the maximum tangential stress theory (Erdogan and Sih, 1963), the maximum strain energy release rate (Hussain *et al.*, 1974 and many others), the minimum strain energy density or S-criterion (Sih, 1974; Sih, 1975), and others. Although some work has been done (e.g. Hodgdon and Sethna, 1992; Leblond, 1993), a general rule for 3D crack propagation in a mixed-mode I, II and III stress field is not currently available.

In 2D, fracture propagation is controlled by the relative magnitudes of the stress intensity factors at the crack front. The direction is usually determined using one of the above theories. The amount of extension is often based on simple power-law relationships between stress intensity factor and crack length, like Paris' model. In FRANC3D, the direction is computed using one of the above theories and the maximum amount of extension is controlled by the user. The 2D, plane strain equations are applied in the plane normal to the crack front tangent to determine the direction of propagation. The

extension along the entire crack front is determined based on the relative values of stress intensity factor along the crack front and the user supplied maximum extension. The new crack front is determined by combining the direction and the amount of propagation at points along the existing crack front Figure 39).

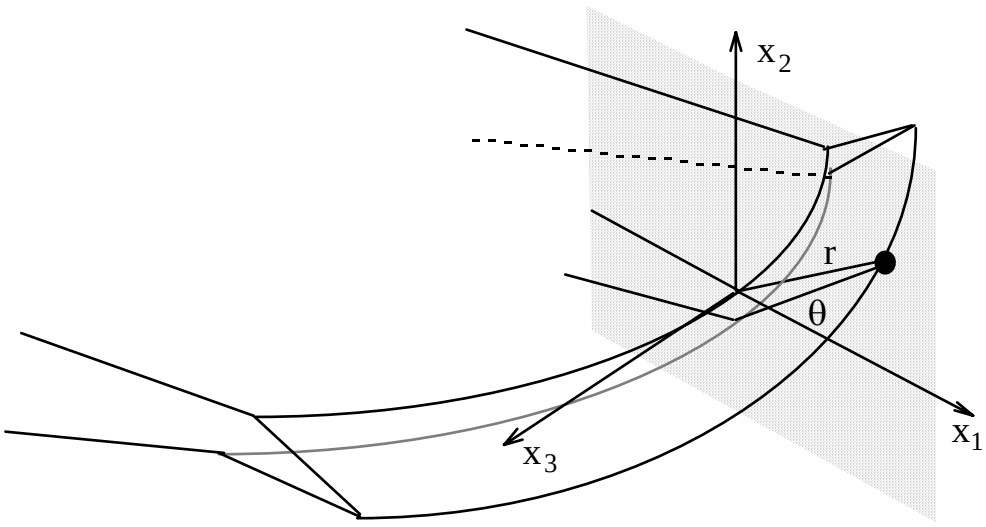


Figure 39. A 3D crack front grows by propagating a series of points along the crack front in the plane normal to the crack front in the plane normal to the front using 2D propagation theories.

4.5 Volume Meshing

Volume meshing is becoming more important as FRANC3D is coupled to finite element programs, such as ANSYS. The advancing front meshing algorithm that is employed in FRANC3D to produce tetrahedral meshes has been described in:

Cavalcante Neto, J.B., Wawrzynek, P.A., Carvalho, M.T.M., Martha, L.F., and Ingraffea, A.R., "An Algorithm for Three-Dimensional Mesh Generation for Arbitrary Regions with Cracks," *Engineering with Computers,* 17: 75-91, 2001.

5 FRANC3D Program Environment

FRANC3D is a menu-driven, interactive type of program with graphical displays for all commands and entries. The commands available in FRANC3D are organized in different menus. The menu system utilized is highly interactive and designed in such a way that menu hierarchy is always preserved. All the options are active above the current menu level. Entries are organized by dialog boxes which allow text, integer and floating point numbers to be input into the program Dialog boxes are blocking, i.e. you can only work

inside the dialog box when it is displayed on the screen. These items along with instructions for interacting with them are discussed in the following sections.

5.1 Mouse and Window Interaction

This section describes how to use the mouse to manipulate windows, menus and dialog boxes in FRANC3D. Sections below describe: naming the mouse buttons, menus (menu cluster), modeling and contouring windows, dialog boxes, and xy-plot windows.

5.1.1 Naming the Mouse Buttons

FRANC3D is designed for use with a 3 button mouse. The 3 buttons are numbered **MB1, MB2,** and **MB3** from left to right (Figure 40):

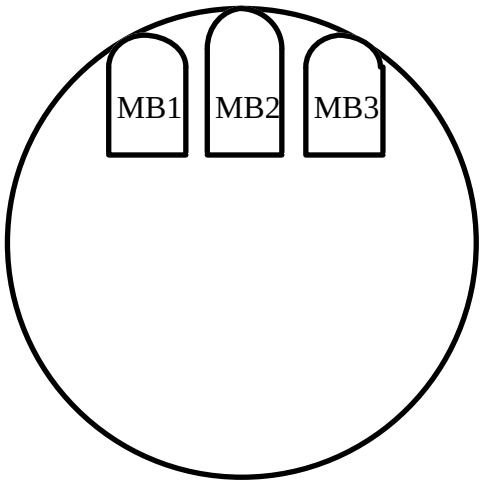


Figure 40. The mouse and its three buttons.

5.2 View Specification

One important feature of an interactive, three-dimensional graphics system is that the user is given the ability to rotate and translate objects on the screen. When designing such a system, one must decide what set of controls should be supplied to manipulate the view of an object. This section describes the controls that are implemented in FRANC3D. The view of a model or the camera position in a 3D window is controlled by the buttons on the view control panel shown below (Figure 41). Each of the buttons is explained below.

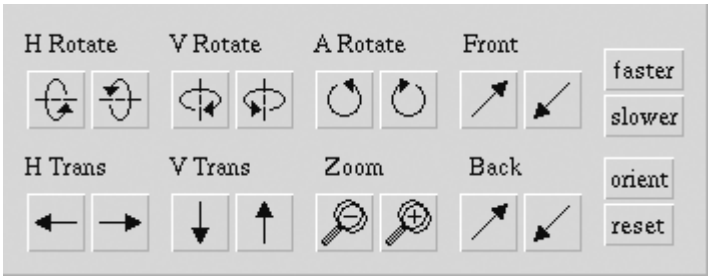


Figure 41. View control panel.

The object is considered to be at a 'reference position' which is viewed from a 'camera position' (Figure 42). The view is controlled by changing the relative position of the camera with respect to the reference position. In addition, front and back 'clipping planes' can be moved toward, or away from the camera and reference positions to cut away portions of the object that can be seen on the screen.

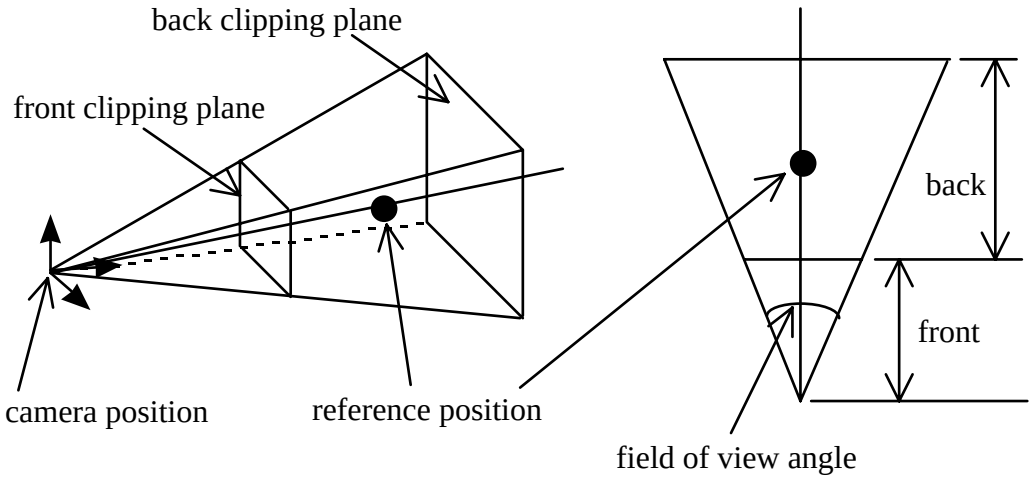


Figure 42. Camera and reference positions with front and back clipping planes.

H Rotate

Rotates the model about a horizontal axis. The left button rotates the model upwards (clockwise when viewed from the right) and the right button rotates the model downwards (counterclockwise when viewed from the right).

V Rotate

Rotates the model about a vertical axis. The left button rotates the model to the left (clockwise when viewed from the top), and the right button rotates the model to the right (counterclockwise when viewed from the top).

A Rotate

Rotates the model in a clockwise/counterclockwise direction (about an axis which extends out of the screen). The left button rotates the model counterclockwise and the right button rotates the model clockwise.

Front

Moves the front clipping (cutting) plane. The left button ↗ pushes the front plane in towards the model, the right button ↙ pulls the front plane out away from the model towards the camera position. Pushing the front plane towards the model allows you to cut away the front portion of the object in order to see the interior of the model.

H Trans

Translates the model in the horizontal direction. The left button moves the model to the left and the right button moves the model to the right.

V Trans

Translates the model in the vertical direction. The left button moves the model downward and the right button moves the model upward.

Zoom

The left button zooms out (pushes the model away making it appear smaller) and the right button zooms in.

Back

Moves the back clipping (cutting) plane. The left button ↗ pushes the back plane away from the model and the right button ↙ pulls the plane forward towards the model. Pulling the back plane towards the model and camera position allows you to cut away the back of the object so that you can see the front portion more clearly.

faster/slower

These two buttons control the speed of the above buttons. The speed is incremented each time one of the buttons is pressed.

option

This button presents a dialog box (Figure 43) that allows the user to snap the model into a view along one of the axes.

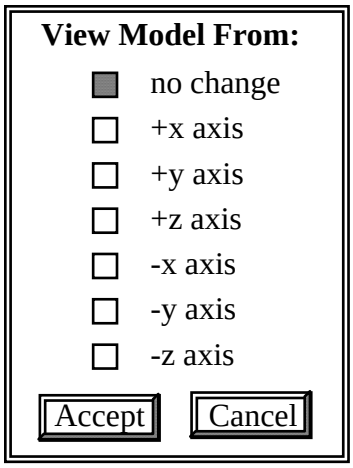


Figure 43. Optional view dialog box.

reset
This button resets the view to the initial default view.

The view can also be controlled with the mouse along with the Shift and Ctrl buttons. Holding the right mouse button down and dragging it allows the user to translate the model in the drag direction. Holding the Shift button down while holding the right mouse button down and dragging the mouse allows the user to rotate the model. Holding the Ctrl button down while holding down the right mouse button and dragging the mouse allows the user to zoom in and out. Holding the Shift button down and holding the middle mouse button down moves the front cutting plane forward and backward. Holding the Ctrl button down and holding the middle mouse button down moves the back cutting plane forward and backward.

5.3 Menu System

Specific menus are referenced by their name which is the top entry in each of the menu clusters. The name is capitalized and performs no apparent action when selected, other than to bring the menu cluster to the foreground. See the Menu & Dialogue Reference Manual for a description of commands on the various menus.

5.3.1 Menus (Menu Clusters)

Menus (sometimes called menu clusters) are manipulated as follows:

Selecting a Command --
- Click **MB1** on the menu entry

Dragging -- To drag a menu cluster to a different screen location:
- Press **MB3** down anywhere on the menu.
- While holding **MB3** down, move (drag) the menu to its new position.

- Release **MB3**.

Bring to Foreground -- To bring a menu (or any other item) in front of others that are obscuring it:

- Click **MB2** on the top menu entry or menu name.

Dismiss -- A **return** command at the bottom of a menu dismisses the menu. Menus that are 'children' of the menu -- i.e. were displayed as the result of selecting an entry on the menu -- are also dismissed.

5.3.2 Menu System Commands and Prompts

Each menu button is unique and the actions that follow once a menu button is pushed vary. The types of responses are:

- direct action (e.g. Write FRANC3D File)
- create a child menu (e.g. Develop Model)
- pop up a sub-list or sub-menu (e.g. Linear/Quadratic/Cancel from Write BES File)
- invoke a collector to gather data interactively from the user (e.g. Part-Thru Crack)
- invoke a dialogue box directly (e.g. Active Subdivision Data)

The result of each menu command should be apparent from the information displayed on the screen. Informative or prompt messages are often written to the information text box that is (initially) located in the lower left corner of the desktop. The most common result of pressing a menu button is the invocation of a collector. Collectors are used to gather information by getting the user to interactively select entities in the main visualization window. A collector is active whenever the **RUBOUT/QUIT/HELP** buttons appear on the lower right quadrant of the screen. Other buttons can also be present and they will be located close to these three buttons.

The process of interactively collecting entities in the main modeling window is quite simple. Vertices and edges can be collected as long as they are visible, and are selected by picking a point that is on or very close to them. Faces are chosen by selecting a point on the face. This means that you must be able to reach the face without passing through other faces. Regions are selected by picking a point within the region. This can be done by entering a set of coordinates of a point in the region, or by picking a point interactively. To select interactively, pick a point that is within the region; this generates a vector. Rotate the model and pick another point along this vector and within the region.

Some menu buttons invoke child menus. All entries above the child menu are active. If a menu button is inactive, it is displayed in a different color and will not produce the indicated action when pressed. All child menus are removed when the parent is removed (selecting return). Some menu buttons have sub-lists or sub-menus attached to them rather than separate child menus. Each entry in a sub-menu performs a unique action. Sub-lists are used to display a list of available items that can be selected to control other

commands. All of the sub-lists and sub-menus have a cancel option which simply exits the sub-menu or sub-list without causing any changes or actions.

5.4 Dialog Boxes

A dialogue box is a rectangular display that requests the user to enter numeric data or text strings, and to set toggle buttons (Figure 44). When a dialog box is displayed, the user must enter data and complete the dialogue by touching **Accept** or **Cancel** on the dialogue box before proceeding to other operations. All touches outside the dialogue box–e.g. zoom/rotate/pan or window moving–are ignored. Dialogue boxes can be dragged, however, by selecting the dialogue with **MB2** and dragging.

A typical dialog box might appear as follows:

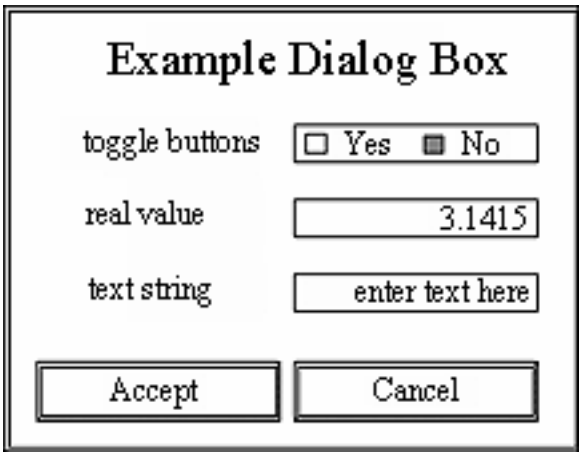


Figure 44. Example dialog box.

The dialog is carried out as follows:

Cancel -- Dismiss the dialog box and cancel the command that started the dialog.

Accept -- Accept the current settings and continue with the command that started the dialog.

Text and Numeric Fields --

- Single or double click in the entry field.
- The field is highlighted (reverse normal colors) to indicate the selection.
- Type new entry.

Exclusive Radio Buttons --

- Click on the button.
- The button is toggled to the other setting.

Non-exclusive Radio Buttons --

- Click on the button.
- The button is toggled on or off.

5.5 Modeling Windows

Windows that display a 3D perspective are called *3D windows*. Examples of 3D windows are (a) the main view of the solid model, (b) color contour and deformed shape windows, and (c) the line plot window.

5.5.1 Main Modeling Window

The main modeling and visualization window is where the model is displayed and where interactive collection of the entities of the model is performed. The window is shown in Figure 45 with one of the pull-down menus exposed. A 3D window appears with a *label* and a *menu bar* across the top. The *label* bar has two grip boxes that are accessed by selecting them with **MB1**:

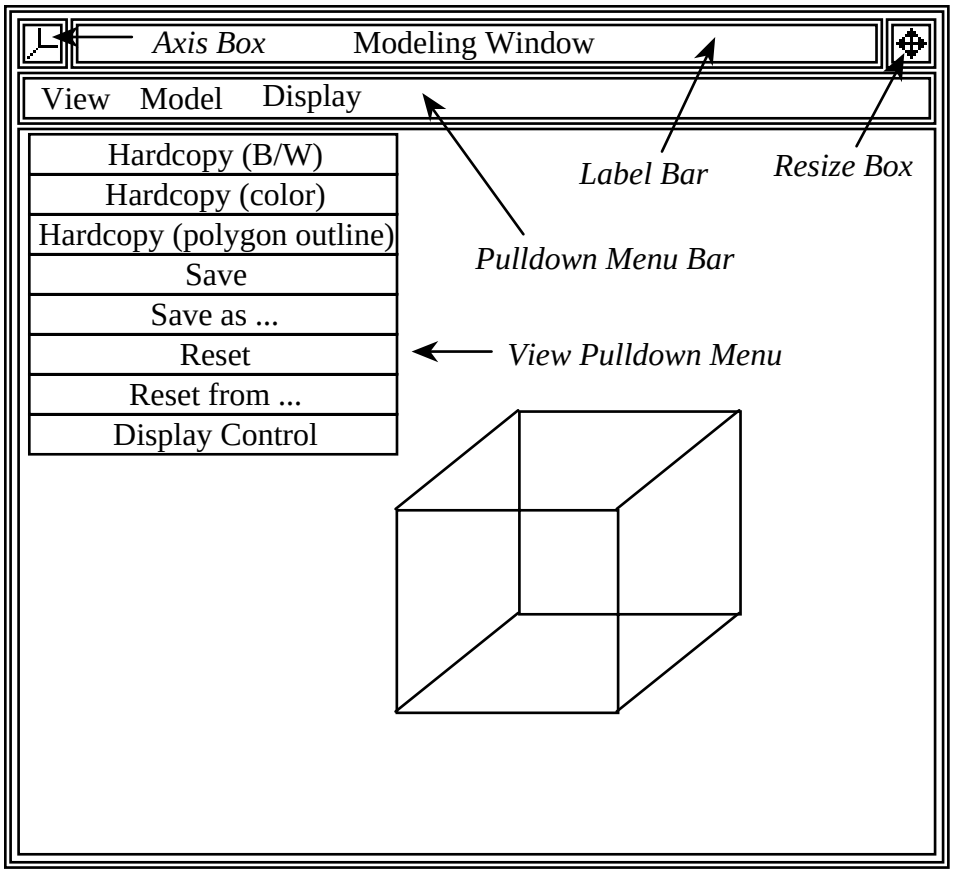


Figure 45. The main modeling window as an example of a 3D window.

(On the left: Orientation box) -- When **MB1** is held down on the axis orientation box, a small window is displayed showing the global x, y, and z axis orientations in the window. The display is removed when **MB1** is released.

(On the right: Resize box) -- To change the size of the window:

- Press **MB1** down in the resize box.
- While holding **MB1** down, move the cursor to a new position for the upper right corner of the window.
- Release **MB1**.

The following operations are done by touching the menu bar:

Dragging -- To drag a 3D window to a different screen location:

- Press **MB3** down anywhere on the menu bar.
- While holding **MB3** down, move (drag) the window to its new position.
- Release **MB3.**

Bring to Foreground -- To bring a 3D window in front of others that are obscuring it:

- Click **MB2** on the menu bar.

5.5.2 Pull Down Menus

The main modeling window has a pull-down menu bar with three options. These menus provide control of the display of the model in the modeling window, the ability to dump the current display to a file, and the ability to save the current view specification.

5.5.2.1 View Pull-down Menu

This menu presents choices for dumping the current display to a Postscript file and saving the current camera position (Figure 46).

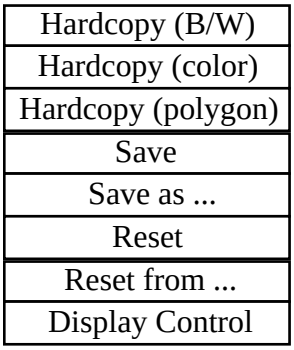


Figure 46. View pull-down menu.

Hardcopy (B/W)
Generates a black and white Postscript file of the current view. A file selector box is presented with a list of all the .ps files. You can overwrite an existing file or create a new file by entering the name in the Filename text field. The .ps extension will be added for you if you forget to type it. These files may be printed directly on a Postscript printer.

Hardcopy (color)
Generates a color Postscript file of the current view. See Hardcopy (B/W) for details.

Hardcopy (polygon)
Generates a Postscript file of the current view with surfaces shaded and hidden lines behind the shaded surfaces not shown. See Hardcopy (B/W) for details.

Save
Save the current camera position as the default.

Save as...
Save the current camera position to a file. A file selector box is presented with a list of all the .cam files. You can overwrite an existing file or create a new file by entering the name in the Filename text field. The .cam extension will be added for you if you forget to type it.

Reset

Reset the view based on the default camera position stored by Save.

Reset from...

Reset the view based on the camera position from a file. A file selector box is presented with a list of all the .cam files. You can select a file by double clicking on the file name or by highlighting the file name and then selecting Accept.

Display Control

Allows the user to turn on and off the display of vectors and polygons.

5.5.2.2 Model Pull-down Menu

This menu presents a dialog box with choices for showing different hierarchical models in the current display (Figure 47). At any time, only one of the levels of the model hierarchy is displayed. The dialog box allows the user to select one of the model levels to be displayed. Hierarchy levels consist of geometry, volume decomposition, face decomposition, edge decomposition, and mesh.

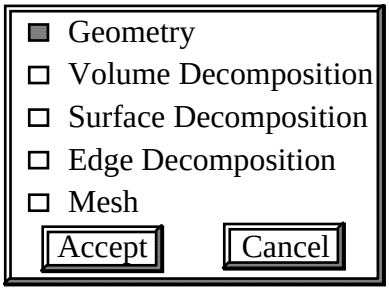


Figure 47. Model pull-down dialog box.

5.5.2.3 Display Pull-down Menu

This menu presents a menu with choices for showing different topological features in the current display (Figure 48).

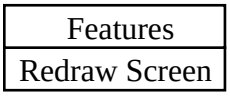


Figure 48. Display pull-down menu.

Features

A dialog box is presented (Figure 49) which allows the user to turn on and off the display of surfaces, edges, and vertices, including hilighting the crack surfaces, edges, and vertices.

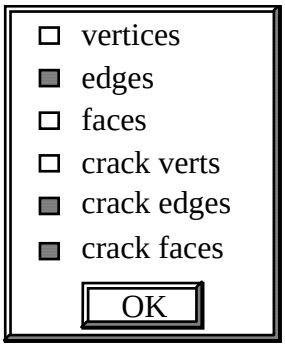


Figure 49. Features pull-down dialog box.

Redraw Screen

Redraws the current display and scales the drawing based on the current camera position.

5.5.3 Deformation and Contouring Window

The deformation/contour 3D window is capable of displaying deformed shapes and color contours. There is an additional grip box along the top menu bar, the Dismiss box.

(Dismiss) -- Pressing **MB1** on the dismiss button deletes the entire window.

There are three pull-down menus.

View

This is the same as that of the main modeling window.

Deformation

This pull-down menu controls the display of the deformed shape. Under Deformation is Alter magnification and Display control (Figure 50). Alter magnification pops up a dialogue box and allows the magnification factor to be changed. The magnification factor affects the display of the deformed shape and depends on the displacement magnitude, as well as the view. The current view is redrawn upon selecting Accept. Display control presents a dialog box that allows you to select different views of the deformed and undeformed shape. Under Display control, any or all of the options can be selected, such as undeformed edges with deformed boundary edges, depending on what the user wishes to see.

Contour

This pull-down menu controls the display of color contours. Under Contour is Response Value, Alter Range, Clear Range, Display Control, and Clear Controls. Selecting Response Value gives a list of responses that can be contoured (Figure 51). Only one response value can be checked at any one time. In order to view the contours, click on the Display Control button and check the desired option. Again, any or all options may be checked, but it is usually best to check just one

option in this case. The contours can be displayed on the undeformed or deformed model in unsmoothed or smoothed formats. The Alter Range menu entry allows the color contours to be restricted to a certain range of values. Selecting Alter Range causes a dialogue box to be displayed. The minimum and maximum values for contouring can then be entered. The current color contours are redrawn using the new range. The color contour scale is displayed on the left hand side of the window.

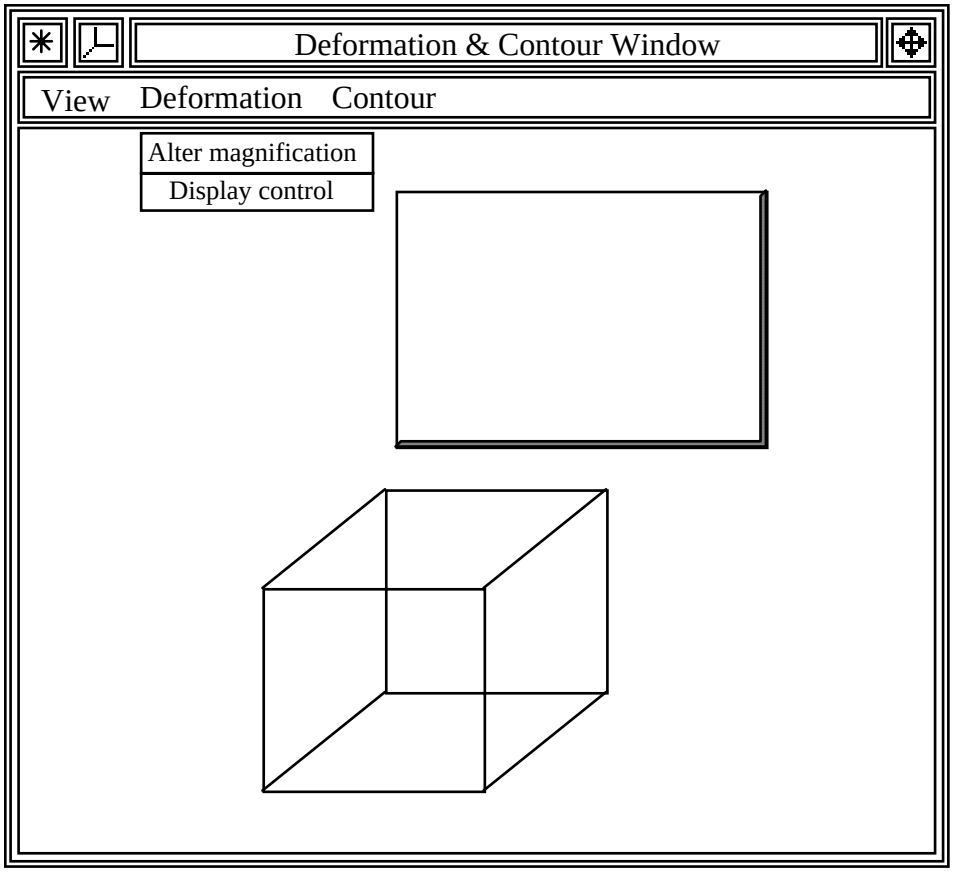


Figure 50. Deformation and contour window showing the Display control pull-down menu.

5.5.4 Line Plot Window

The line plot window (Figure 52) is similar to the deformation/contour window, except it has a line plot window adjacent to the 3D window. It has three pull-down menus, one of them being the View pull-down. The second is the File pull-down menu which allows the user to save the x-y data to a file in either Ascii or Postscript format. The third pull-down menu is Data which allows the user to select the response value to be plotted and then plot the response in an x-y plot window.

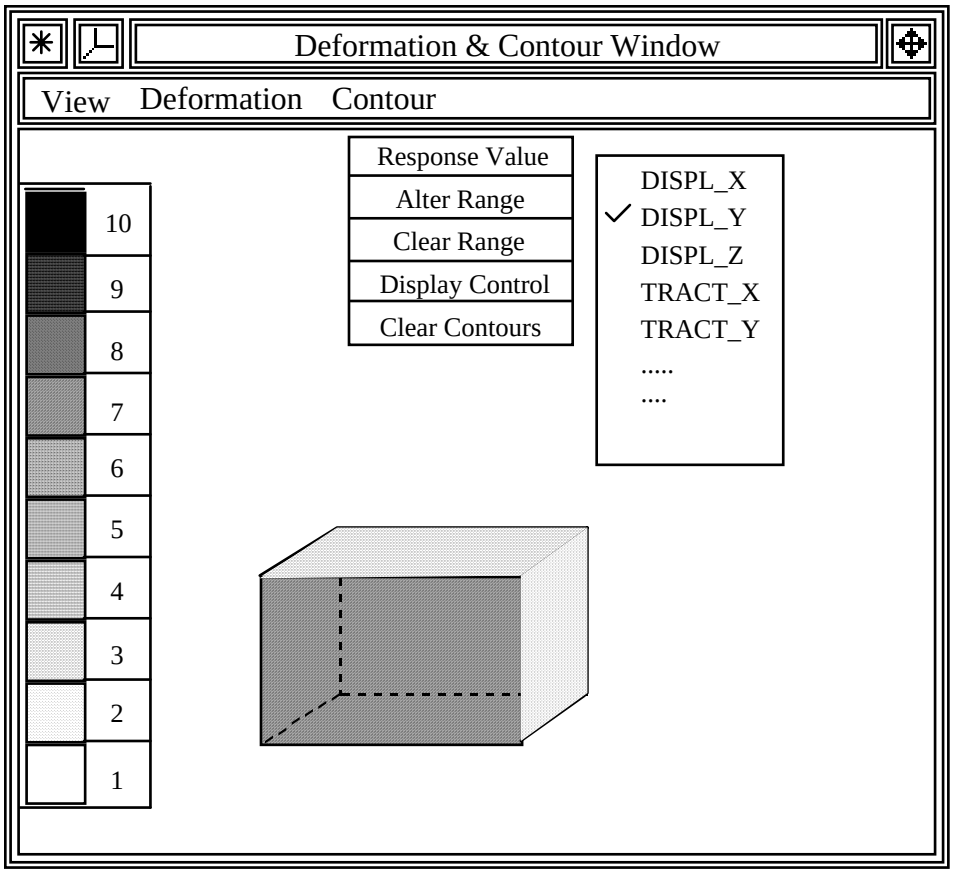


Figure 51. Deformation and contour window showing the Response value pull-down menu.

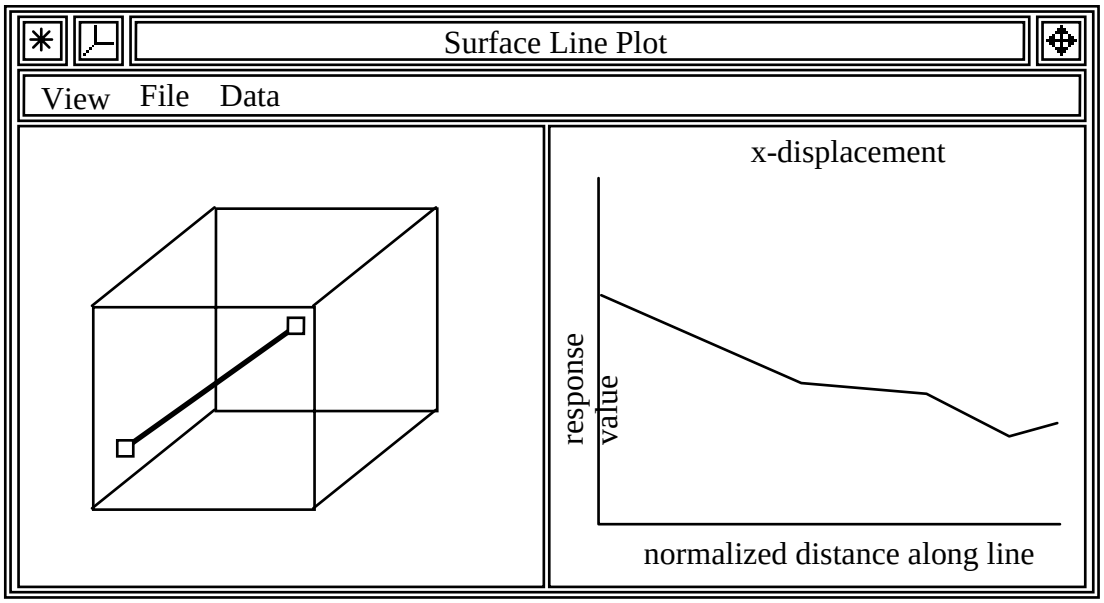


Figure 52. Example of a line plot window.

5.5.5 XY-Plot Windows

Windows that display xy data plots appear as shown in Figure 53. There are two grip boxes on the top menu bar, dismiss on the left side and resize on the right side. There is one pull-down menu for saving the data in a file.

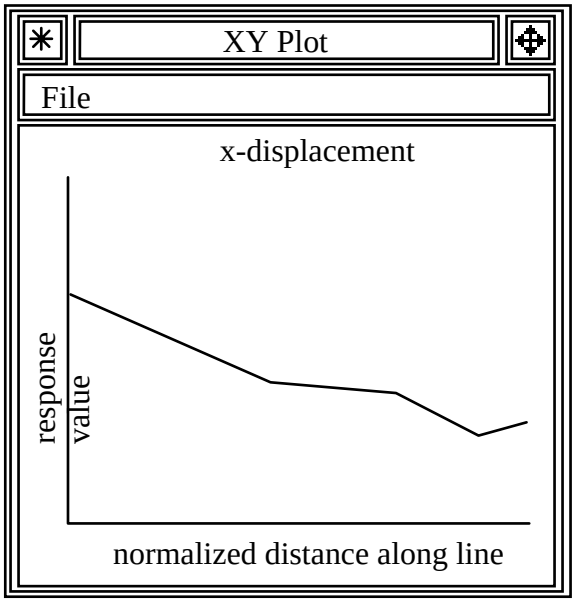


Figure 53. Example of an x-y plot window.

Mouse touches in the window are interpreted as follows:

Dismiss

Pressing **MB1** on the dismiss button deletes the entire window.

Resize button

To change the size of the window:

- Press **MB1** down in the resize button.
- While holding **MB1** down, move the cursor to a new position for the upper right corner of the window.
- Release **MB1**.

The following operations are done by touching the window:

Dragging

To drag an XY plot window to a different screen location:

- Press **MB3** down anywhere in the window.
- While holding **MB3** down, move (drag) the window to its new position.

Bring to Foreground

To bring an XY plot in front of the others that are obscuring it:

- Click **MB2** on the menu bar.

Hardcopy

- Select the **File** pull-down menu.
- An Ascii file of the data can be written for the current plot.
- A Postscript file can be written for the current plot. The image is dumped to a HardCopy window which allows the image to be saved as a Postscript file.

6 Automated Crack Growth Simulations

FRANC3D is capable of automatically simulating 3D crack growth. Traditionally, the program has been used interactively to analyze a single step of crack growth at a time. This includes manually propagating the crack, manually remeshing the model after growing the crack, manually writing the files for the new model, and manually starting the stress analysis procedure. These steps can be completed automatically once the first crack model is created and the crack growth parameters are set.

6.1 Automated Crack Growth

Automatic crack growth can be performed from within the FRANC3D menu environment by selecting the **Run Automated** command from the **AUTOMATIC PROPAGATION** menu. An initial cracked model must exist and the crack growth model must be defined prior to selecting this command. The **AUTOMATIC CRACK PROPAGATION CONTROL PARAMETERS** dialog box (Figure 54) is used to define the number of steps of propagation, the current step in the analysis, and a number

of other parameters in sub-level dialog boxes to control the rate of crack advance per analysis step and the remesh refinement near the crack.

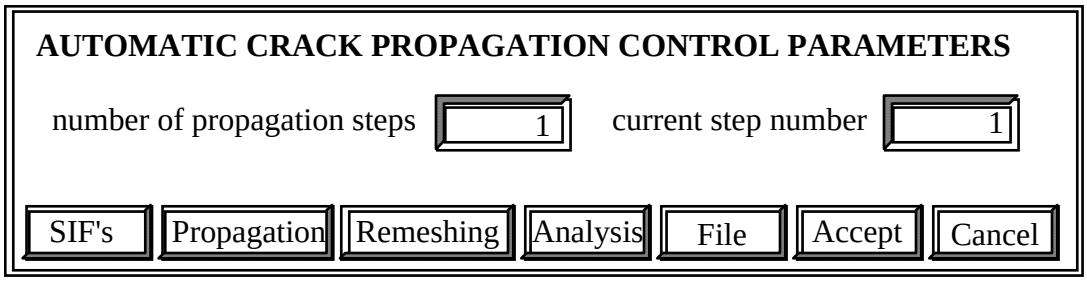


Figure 54. The **AUTOMATIC CRACK PROPAGATION MODEL** dialog box.

All files for each step of propagation/analyis are saved. The model name is used as the first part of the file name. Then a "_STEP_#" is added after the model name prior to the appropriate file extension. The # corresponds to the propagation step number as set in the above dialog box.

After setting the parameters to control the crack growth simulation process, the user can select **Run Automated** to start the process. The .fys file with the _STEP_# is first saved. Then the analysis input file is saved and the stress analysis procedure is automatically invoked from within FRANC3D. FRANC3D "sleeps" while the stress analysis procedure runs. When the stress analysis is completed, FRANC3D "wakes up" and reads in the output file from the stress analysis. The stress intensity factors are computed and the crack is propagated based on the parameters set earlier using the dialog box in Figure 54. If the crack propagates successfully, re-meshing is performed where needed. The .fys and stress input files are then saved with an incremented _STEP_# and the process continues until the specified number of steps is completed or until FRANC3D has difficulty in automatically propagating the crack. In the latter case, the simulation can be continued by manually finishing the crack propagation and re-meshing and then selecting the **Run Automated** command again.

6.2 Interactive versus Batch Modes

The previous section described how to do automated crack growth simulations with FRANC3D in an interactive mode. Running in this mode requires the FRANC3D window to be open during the entire simulation process and the user therefore cannot log out of the computer. Therefore, FRANC3D also can be run in a batch mode in the background allowing the user to log out. To run in batch mode and in the background, simply add a flag "-batch" after the franc3d executable name, followed by the flag "-file", the .fys file name, the flag "-crack", and the file containing the crack growth parameters, followed finally by the "&" character. For example:

```
csh> franc3d -batch -file bracket.fys -crack
crack_growth_model &
```

Note that a batch mode exists for reading and writing files as well. For instance, if you transported a number of .afys files from one workstation to another and then wanted to resave them as .fys files, you can do this with a simple script. For example:

```
csh> franc3d -in bracket.afys -out bracket.fys
```

NOTE: The batch mode is designed to work only with BES, the boundary element code, for this version (2.2). There are four executables for BES Version 5; these are cge_bes, cocqr_bes, cit_bes, and cocit_bes. These should be in the user's PATH or in the current working directory. The default Gauss elimination solver is contained in cge_bes, which is an in-core solver. For those users where computer memory is short, the out-of-core QR solver in cocqr_bes can be used; the user must have sufficient disk space to store the temporary coef.dat file however.

7 Fatigue Life Prediction

This section describes the fatigue life prediction module of the FRANC3D program. This module allows an analyst to use a computed stress intensity factor history, along with appropriate properties, to compute a relationship between crack size and the number of applied load cycles.

An idealized plot of a typical relationship between stress intensity factor range and crack growth rate for a metal, for a given mean cyclic stress, is shown in Figure 55. The curve is characterized by three regions. In Region I, crack growth is very slow, and may retard completely below a "threshold" stress intensity factor range. In Region II, the curve is nearly linear in the log-log space. In Region III, the maximum applied stress intensity factor approaches the toughness of the material, and the crack growth rate accelerates.

As indicated in the figure, the curve is a function of R , the ratio of the minimum applied stress intensity factor to the maximum (or equivalently, the ratio of the minimum applied cyclic load to the maximum). As R increases, the crack growth rate tends to increase.

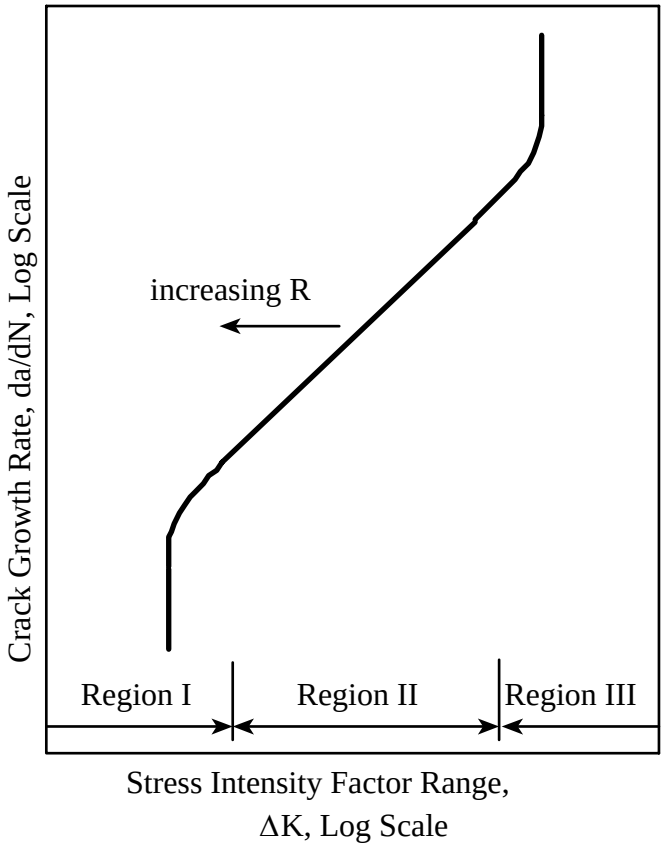


Figure 55 An idealized plot of a typical relationship between stress intensity factor range and crack growth rate for a metal for a given mean cyclic stress.

7.1 Crack Growth Rate Models

7.1.1 Paris Model

The Paris model is the simplest widely used crack growth rate model. It assumes a power law relationship between the stress intensity factor range and the crack growth rate. That is, it models Region II, and ignores Region I and Region III. The expression for the Paris model is

$$\frac{da}{dN} = C(\Delta K)^n. \tag{7.1}$$

In this equation, C and n are constants that are found by a curve fit to experimental data. Notice that the crack growth rate in this model is not an explicit function of R. This means that C is only valid for R (mean cyclic stress) used in the test from which it is determined.

While very simple, the Paris model is widely used in the aerospace community because for many components Region II represents the majority of the fatigue life. Also, with

only two empirical constants, it can be used when a large amount of test data are not available.

7.1.2 Forman-Newman-de Koning Model

The Forman-Newman-de Koning equation starts with the basic Paris relationship, and adds modifications to account for retardation near threshold, acceleration near fast fracture, and the effect of *R*. The expression for this model is

$$\frac{da}{dN} = \frac{C'(1-f)^n \Delta K^n \left(1 - \frac{\Delta K_{th}}{\Delta K}\right)^p}{(1-R)^n \left(1 - \frac{\Delta K}{(1-R)K_c}\right)^q} \tag{7.2}$$

where *C*, *n*, *p*, and *q* are empirical constants (note that *C'* is not the same as the *C* used in the Paris model; the *n*'s, however, are the same) [Forman 92 & 94].

In this equation *f* is a function that incorporates the effect of *R* by analyses for plasticity-induced crack closure for constant amplitude loading. This is defined as [Newman 84]:

$$f = \begin{cases} \max(R, A_0 + A_1 R + A_2 R^2 + A_3 R^3) & R \geq 0 \\ A_0 + A_1 R & -2 \leq R < 0 \end{cases}. \tag{7.3}$$

The coefficients are defined as:

$$\begin{aligned} A_0 &= (0.825 - 0.34\alpha + 0.05\alpha^2)\left[\cos\left(\frac{\pi}{2}\frac{S_{\max}}{\sigma_0}\right)\right]^{1/\alpha} \\ A_1 &= (0.415 - 0.071\alpha)\frac{S_{\max}}{\sigma_0} \\ A_2 &= 1 - A_0 - A_1 - A_3 \\ A_3 &= 2A_0 + A_1 - 1 \end{aligned} \tag{7.4}$$

In these equations, *a* is a plane stress/strain constraint factor, and S_{max}/s_0 is the ratio of the maximum applied stress to the material flow stress. Despite their physical significance, both are treated as fitting parameters for this model.

The threshold stress intensity factor, ΔK_{th}, is a function of *R*, the current crack length, *a*, and an "intrinsic" crack length, a_0, which has a constant value of 0.004 in (0.102 mm). The expression for this is:

$$\Delta K_{th} = \Delta K_0 \left[\frac{4}{\pi}\tan^{-1}(1-R)\right]\left(\frac{a}{a+a_0}\right)^{1/2}. \tag{7.5}$$

The intrinsic crack length term accounts for small crack effects [Tanaka 81]. ΔK_0 is assumed to be an empirical constant.

The critical stress intensity factor is a function of the thickness. The expression for this is:

$$K_c = K_{Ic}\left(1 + B_k e^{-(A_k / t_0)^2}\right) \quad \text{with} \quad t_0 = 2.5\left(K_{Ic}/\sigma_{ys}\right)^2 \tag{7.6}$$

While the Forman-Newman-de Koning equation is much more general than the Paris equation, there are a large number of empirically determined constants. Finding experimental results and performing the required curve fitting for a given material system represents a substantial amount of work. Fortunately, this has been done by NASA for over 300 different material and environment combinations and has been incorporated into the NASGRO material data base. The NASGRO material database, in turn, has been incorporated into FRANC3D and is readily available to an analyst. A plot of the Forman-Newman-de Koning equation for a Ti-6Al-4V forging is shown in Figure 56.

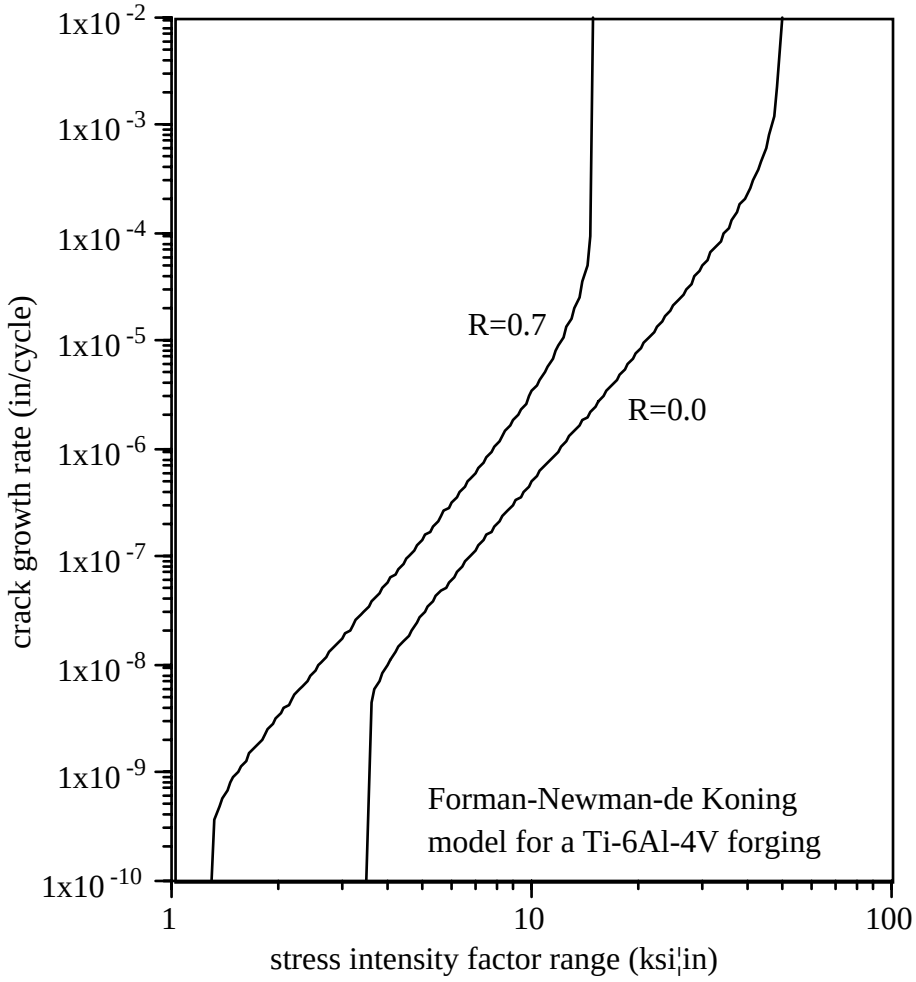


Figure 56. A plot of the Forman-Newman-de Koning equation for a Ti-6Al-4V forging.

7.1.3 SINH Model

A third model based on a hyperbolic sine function has been included in the program. It has the form:

$$\log\left(\frac{da}{dN}\right) = A\sinh(B(\log(\Delta K) + C)) + D. \qquad (7.7)$$

This equation has four empirical constants (*A, B, C,* and *D*). Since *R* does not appear explicitly in the equation, the constants must be determined for a given mean stress. The equation is plotted in Figure 57.

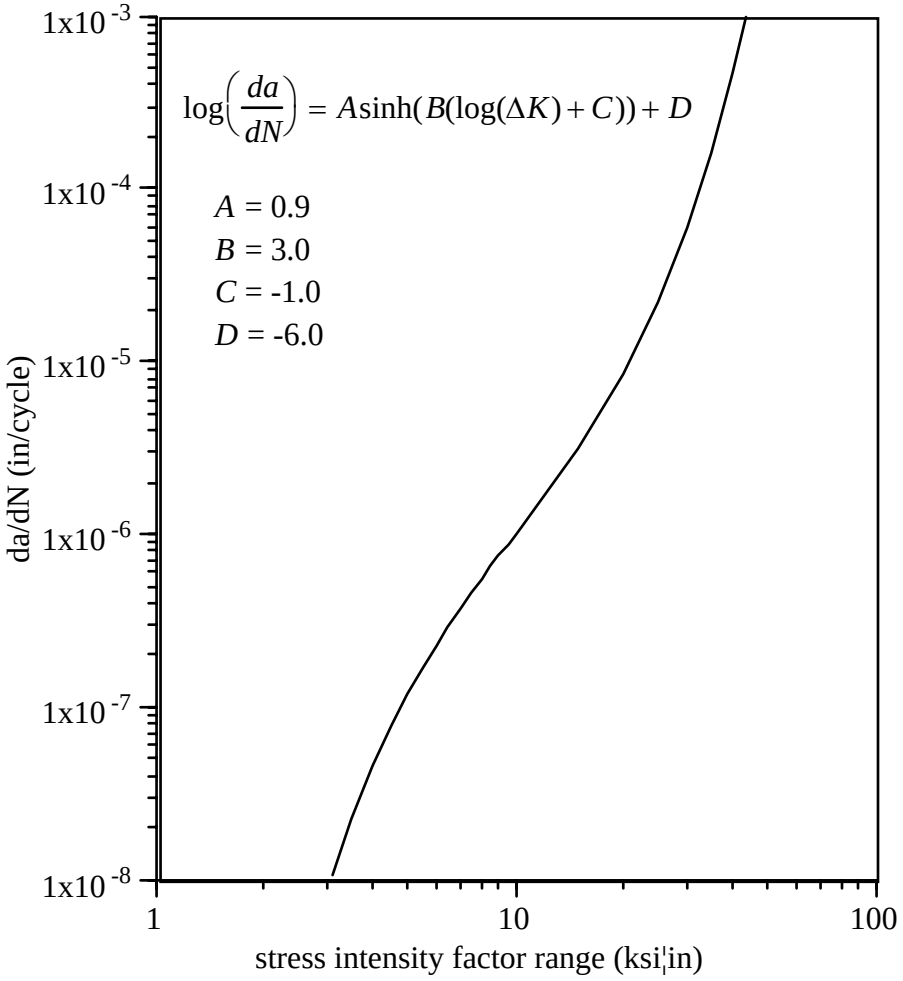


Figure 57. The sinh-based crack growth rate model.

7.1.4 Willemborg Retardation

Variable amplitude loading can result in crack growth rates that may differ from those predicted by constant amplitude growth rate data. Variable amplitude cycles have been observed to retard or accelerate crack growth depending on the sequence of load amplitudes [Schijve 1976]. A number of simple analytical models have been proposed to model this effect. Most notable are those due to Wheeler [1972] and Willemborg [1971]. The Willemborg model has been implemented in FRANC3D. A nice feature of the model is that, unlike Wheeler's model, no additional material properties are required.

The Willemborg model was originally proposed in terms of cyclic stresses. Broek [1978] generalized this in terms of cyclic stress intensity factors. The basic idea of the model is that the stress intensity factors for cycles following an overload will be reduced due to the plastic zone created at the crack tip during the overload. This is shown schematically in Figure 58.

In the model, following an overload, the residual stresses in the plastic zone reduce the K_{max} and K_{min} by an amount K_{red}. This is defined by the following function,

$$K_{red} = K_{max,req} - K_{max,i} \tag{7.8}$$

where $K_{max,req}$ is the K_{max} required to overcome the plastic zone created by the overload, and $K_{max,i}$ is the K_{max} due to the *i*th cycle following an overload.

In the first cycle following the overload, $K_{max,req}$ is equal to the K_{max} of the overload cycle. As the crack grows through the plastic zone created by the overload, $K_{max,req}$ decreases until it equals $K_{max,i}$. At this point there is no more retardation. $K_{max,req}$ is calculated from the equation:

$$6\pi \frac{K_{max,req}^2}{\sigma_{ys}^2} = a_0 + r_{p0} - a_i \ . \tag{7.9}$$

In this equation, a_0 is the crack length at the overload, and a_i is the current crack length (see Figure 58). The effective cyclic stress intensity factors are then computed as:

$$\begin{aligned} K_{max,eff,i} &= K_{max,i} - K_{req} = 2K_{max,i} - K_{max,req} \\ K_{min,eff,i} &= K_{min,i} - K_{req} = K_{min,i} - (K_{max,req} - K_{max,i}) \end{aligned} \tag{7.10}$$

From which it follows that

$$\Delta K_{eff,i} = K_{max,eff,i} - K_{min,eff,i} \tag{7.11}$$

and

$$R_{eff,i} = \frac{K_{min,eff,i}}{K_{max,eff,i}} \ . \tag{7.12}$$

7.2 Fatigue Life Prediction Parameters

The fatigue crack Life Prediction dialog is accessed through the "Predict Fatigue Life" command on the "Fracture Analysis" menu. Before this command can be accessed, however, a stress intensity factor history must be specified. This is done with the "Stress Intensity Factor History" command, also on the "Fracture Analysis" menu. The menus and dialog boxes are described fully in the **FRANC3D Dialog & Menu Reference**.

Nine life prediction parameters must be defined to perform a fatigue life calculation. Figure 59 shows the top-level fatigue life dialog box. Each of the nine parameters can be configured from this dialog box as described in the **FRANC3D Dialog & Menu Reference**.

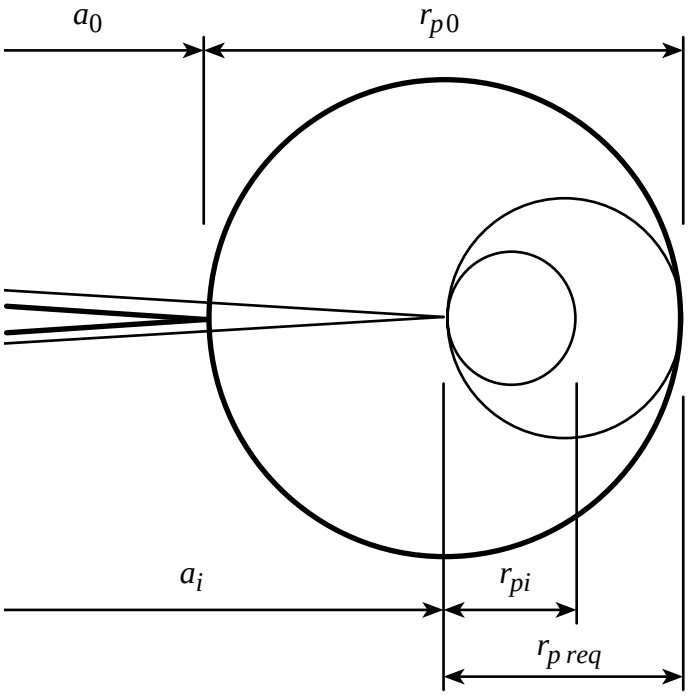


Figure 58. Schematic of the crack tip region following an overload.

7.2.1 SIF History

If parameters are specified in a SIF History file, the first line of the file must contain "%%" in columns one and two. After this, parameters are specified one per line. The format for each line is <name>: <value1> [<value 2> ...].
Another line with "%%" in columns one and two indicates that all parameters have been specified. The remainder of the file contains pairs (one per line) of crack lengths and the corresponding K values.

The parameter names and values are defined as follows (square brackets "[]" indicate optional values, the vertical bar "|" means one value *or* the other):

units: psi_in | ksi_in | mpa_mm | mpa_m
sets the type of units to be used for the calculations

hist_trans: <multiplication factor> <offset>
sets the multiplication factor and offset for the SIF history transformation

paris: <C> <n> [<yield stess> [<material name>]]
specifies that a Paris model is to be used, and sets the model parameters

fnk: <material code>
specifies that the Forman-Newman-de Koning model is to be uses, and sets the fnk material code

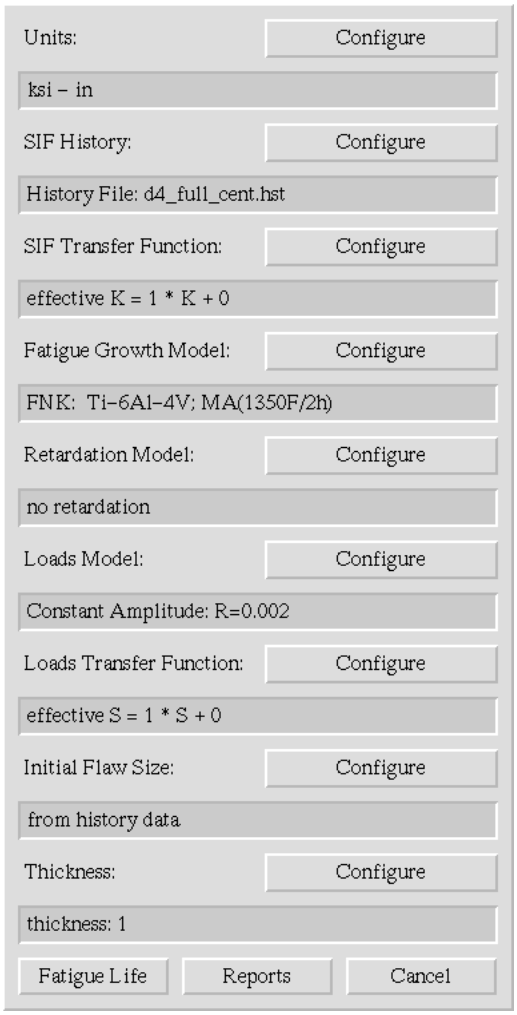


Figure 59. The Life Prediction Dialog.

retard_model: Willenborg
specifies that a retardation model is to be uses. Currently the only model available is that of Willenborg

constant_amp: <R>
specifies that constant amplitude loading is to be used, and sets the R ratio value

spectrum: <spectrum file name>
specifies that spectrum loading is to be used, and sets the name of the file that contains the spectrum description

load_trans: <multiplication factor> <offset>
sets the multiplication factor and offset for the load transformation

initial_size: <initial flaw size>
sets the initial flaw size

7.2.2 The Material Database

Fatigue life prediction depends on material parameters. For the Paris model, the C and n parameters need to be entered in a separate dialog box. For the FNK model, a material database of crack growth model data from the NASGRO database has been encoded into FRANC3D. The user can select from any of the available materials using the dialog box shown in Figure 60.

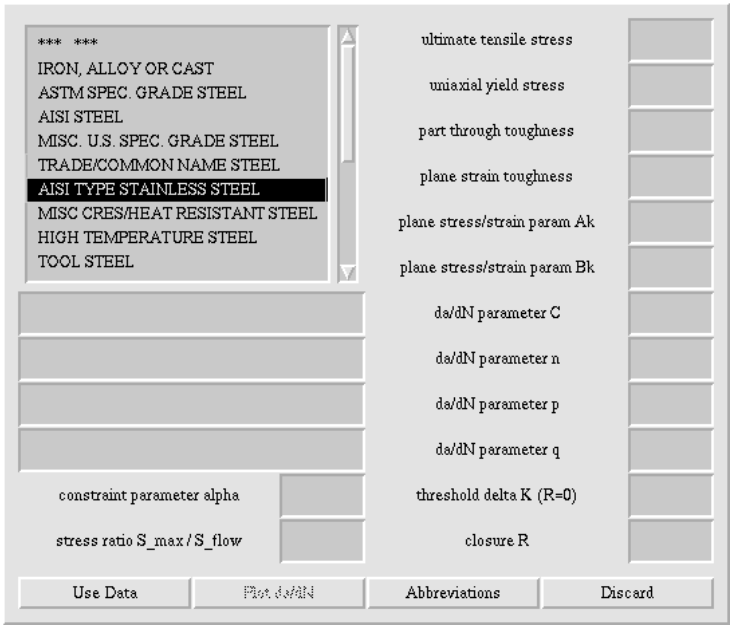


Figure 60. FNK material database.

7.3 Spectrum Loading

The Load Spectrum dialog allows an analyst to define, read, and view a load spectrum, Figure 61. The dialog box is accessed through the main life prediction dialog box. During a fatigue life prediction, once the program steps all the way through the spectrum, it automatically starts over again at the beginning of the spectrum.

In this dialog, a load spectrum is defined by one or more load blocks. Each load block is defined by one or more load events. A load event is comprised of one or more sets of minimum and maximum values (sometimes called excursions). The program does not impose any specific actual interpretation of the block and events. The user assigns the meaning. For example, one possible interpretation is that each block represents one flight. The events represent identifiable portions of the flight such as taxi, climb out, cruise, maneuver, landing, and taxi. Within each event would be series of minimum and maximum load values characteristic of those expected for the appropriate flight segment.

A spectrum definition is generated by typing in values in the appropriate places in three scrolling "spread sheets". The tab or arrow keys can be used to move within a spreadsheet. One must use the mouse to move between spread sheets (point and click on a cell). Moving past the "bottom" of the spread sheet will automatically create additional rows.

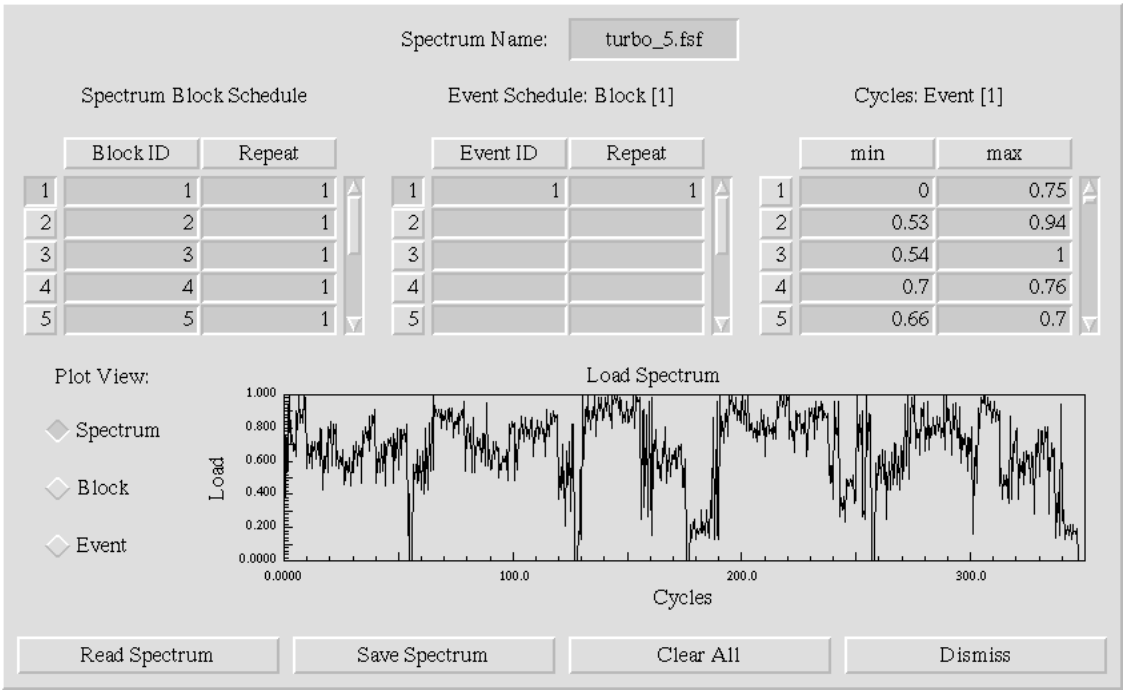


Figure 61. An image of the load spectrum definition dialog box.

As a simple example, consider a load sequence of 10 constant amplitude load cycles that vary between zero and one. This is followed by one overload to a value of two, followed by a load reversal to minus one. The spectrum will consist of five of these load sequences.

To define the spectrum, starting with the block schedule, we define block number 1, which will be repeated 5 times. In the event schedule, we will define three events. Event number 1 is the constant amplitude cycles, these are repeated 10 times. Event number 2 is the overload, this is "repeated" only once. Event number 3 is the underload. This is also "repeated" only once.

Now, to define the cycles, you must insure that the cycles are being defined within the appropriate event. The current event is shown in the label above the cycles spread sheet. Notice that the corresponding event row number is highlighted in the event schedule spread sheet. As you move between rows in the event schedule spread sheet, the cycle definitions for the corresponding events are shown in the cycles spread sheet.

Select one of the cells in the first row of the event schedule. This will display the cycles for event number one. In the cycles spreadsheet, enter zero for the minimum value and one for the maximum value.

Now select one of the rows in the second row of the event schedule. This will display the cycle definition for event number 2 (currently nothing is defined). Move to the cycles spread sheet, and enter a minimum value of zero, and a maximum value of two.

Repeat the process for the third row, this time enter a minimum value of -1 and a maximum value of zero. If you now change the input focus cell, the spectrum will be displayed in the graph.

This same spectrum could have been defined a number of different ways for example, we could have used only two events. The first event would have been the same, but the second event would contain two cycles, the first from zero to two, and the second from minus 1 to zero.

8 FRANC3D Resource File

The 'Franc3D' file contains resources which can be modified to change the colors, fonts, default sizes, and title of the FRANC3D modeling environment. A description of the resources that can be set is given in Figure 62. The Franc3D file should be placed in the user's home or login directory. If the file does not exist, reasonable default values are used instead.

Name	Type	Default	Description
Font			

*facFont	Font	*times-medium-r-*-120*	Font used for menus and dialongs
*msgFont	Font	*courier-medium-r-*-100*	Font used for the message box
*facTitleFont	Font	*helvetica-bold-r-*-120*	Font used for title box
Color			
*numColorLevel	Number	10	Number of shades of colors
*facInsensitiveColor	Color	FireBrick	Color of inactive menu options
*facRemBackground	Color	DarkSlateGray	Background color of the main screen
*facWritingColor	Color	MidnightBlue	Color used for menu text, dialogs, and simple graphics
*msgBorderColor	Color	Black	Border color for the message box
*messageBox.backgroud	Color	Tan	Background color for the message box
*TdGraphics.background	Color	Black	Background color for 3D windows
*conColor1	Color	rgbi:0.369/0.427/0.596	Colors for contours of response
*conColor2	Color	rgbi:0.376/0.329/0.525	
*conColor3	Color	rgbi:0.369/0.227/0.396	
*conColor4	Color	rgbi:0.388/0.200/0.200	
*conColor5	Color	rgbi:0.439/0.259/0.110	
*conColor6	Color	rgbi:0.498/0.388/0.086	
*conColor7	Color	rgbi:0.529/0.498/0.000	
*conColor8	Color	rgbi:0.549/0.620/0.200	
*conColor9	Color	rgbi:0.498/0.620/0.396	
*conColor10	Color	rgbi:0.349/0.427/0.188	
*hilightPointColor	Color	rgbi:0.941/0.000/0.000	Highlighing color for points
*hilightVectorColor	Color	rgbi:0.000/0.941/0.000	Highlighting color for vectors
*hilightFaceColor	Color	rgbi:0.000/0.000/0.941	Highlighting color for surfaces
*hilightRegionColor	Color	rgbi:0.941/0.941/0.941	Highlighting color for regions
*defMeshColor	Color	rgbi:0.941/0.000/0.000	Deformed mesh color
*crackEdgeColor	Color	rgbi:0.941/0.941/0.941	Crack edge color
*crackVertexColor	Color	rgbi:0.941/0.941/0.941	Crack vertex color
*tdgVectColor	Color	Yellow	Vector and edge color
Title			
*FacTitle.facTitle	Text	FRANC 3D	Main title in the title box

Figure 62. Contents of the Franc3D resource file.