A GEOMETRY-BASED ANALYSIS FRAMEWORK

Mark W. Beall
Mark S. Shephard
Scientific Computation Research Center
Rensselaer Polytechnic Institute
Troy, NY 12180-3590

SUMMARY

This paper provides an overview of an analysis framework which operates directly from a general geometry-based specification. The framework is designed using object-oriented methodologies to allow for easy extension to analyze new problem classes and introduce additional adaptive control techniques.

INTRODUCTION

The numerical analysis of a physical problem can be seen as a series of idealization steps, each of which may introduce errors into the solution. The manner in which these errors can be understood and controlled is through error estimation and solution adaptivity. Since adaptive control should be applied to each idealization step, the numerical analysis procedures must operate from the original problem definition which is best described with respect to a geometric model. This paper provides a brief overview of an analysis framework which operates directly off such geometry-based problem specifications.

To increase the usefulness of the framework, it is designed for easy extension to include new analysis capabilities and adaptive idealization control techniques. This extensibility is aided by the application of object-oriented programming techniques.

We can identify three levels of description that arise in the analysis of a physical problem (Figure 1). The highest level description is that of the physical problem. The physical problem description is posed in terms of physical objects interacting with their environment. Since we often want to estimate the response through modeling we idealize the behavior in terms of a mathematical problem description. The mathematical problem description consists of a domain definition (geometry), a description of the external forces acting on the object and the properties of the object (attributes), and, in the classes of physical problems considered here, a set of appropriate partial differential equations which describe the behavior of interest. Construction of a numerical problem from a mathematical problem involves another set of idealizations. From a single mathematical problem it is possible to construct any number of levels of numerical problems, which are idealizations of one another.

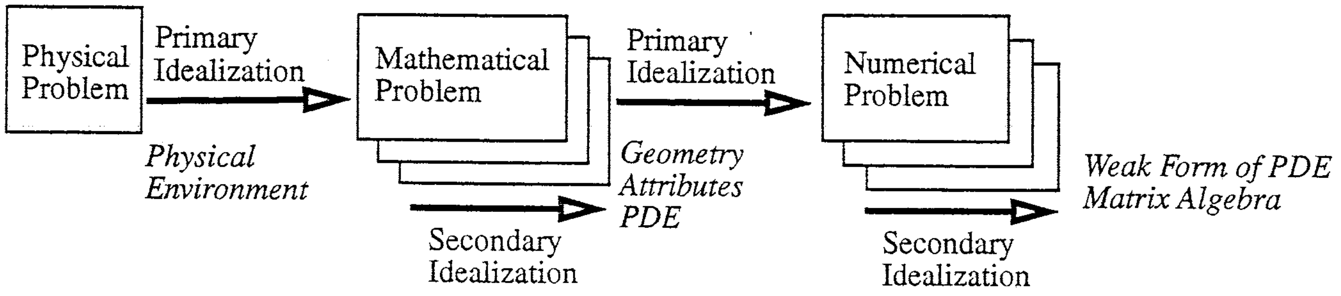

Figure 1. Idealizations of a physical problem to be solved.

The framework described in this paper starts at the level of a mathematical problem description, allowing multiple numerical problems to be formulated, solved, and the solution related back to the original problem description. The analysis framework is designed to be extended. It is possible to add new problem types that can be solved as well as adding new solution techniques. Current implementation efforts are focused on finite element discretizations. However, the framework is designed to be general to utilize other types of numerical solution procedures.

Since the analysis framework must take a problem description consisting of a geometric model and attributes and construct a solution to the problem specified, it is important to understand abstractions for the various types of data that the framework uses. As outlined in the next section, geometry-based descriptions are best suited to meet these needs. The following section briefly introduces the process of performing geometry-based analyses.

DATA COMPONENTS OF A GEOMETRY-BASED ANALYSIS FRAMEWORK

The structures used to support the problem definition, the discretizations of the model and their interactions are central to the analysis framework. The two structures of the geometric model and attributes are used to house the problem definition. The general nature of the attribute structures allow them to also be used for defining numerical analysis attributes. The analysis discretizations are housed in the mesh structure which is linked to the geometric model. The final structure is the field structure which houses the distributions of numerical solution results over the domain of the problem.

Geometric Model

The geometric model representation used by the analysis framework is a boundary representation based on the Radial Edge Data Structure (Weiler 1988). In this representation the model is a hierarchy of topological entities called regions, shells, faces, loops, edges and vertices (Figure 2). This representation is completely general and is capable of representing non-manifold models that are common in engineering analyses. The use of a boundary representation is very convenient for attribute association and mesh generation processes since the boundaries of the model are explicitly represented.

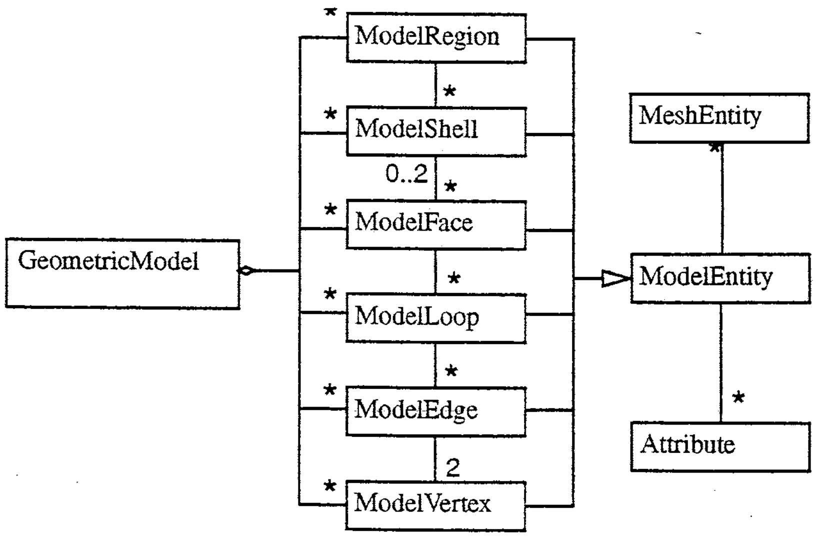

Figure 2. Boundary representation.

The geometric model classes support operations to find the various model entities that make up a model and to find which model entities are adjacent to a given entity. Other operations relating to performing geometric queries are also supported. The details of these operations are not important in the current context. Much more important is the fact that there are associations between the ModelEntity class and both the Attribute and MeshEntity classes. These associations are central to being able to support generalized adaptive analysis procedures that operate from a general problem definition.

Attributes

In addition to geometry, information that describes such things as material properties, loads and boundary conditions (Shephard 1988) is needed. This other information is described in terms of tensor valued attributes that may vary in both space and time. Attribute information is organized into a directed acyclic graph (DAG). There are three basic types of nodes in the graph. The leaf nodes of the graph are information nodes. These nodes hold the actual attribute information (e.g. an information node might define a vector with a certain variation in space and time). Above the information nodes are two types of grouping nodes which allow for the flexible combination of attributes to form analysis cases which drive the numerical analysis process.

Tensor valued attributes only make sense when applied to and associated with a geometric model entity. During this process the graph is traversed, and when the information nodes are encountered at the leaves of the graph, attribute objects are created. These attributes are a particular instance of the information represented in the attribute graph. One reason for the distinction between the information nodes and attributes is that the interpretation of the information node can depend on the path in the graph traversed to get to that node. Thus one information node may give rise to multiple attributes with different values.

A simple example of a problem definition is shown in Figure 3. The problem being modeled here is a dam subjected to loads due to gravity and due to the water behind the dam. There are a set of attribute information nodes that are all under the attribute case for the problem definition. When this case is associated with the model, attributes (indicated by triangles with A's inside of them) are created and attached to the individual model entities on which they act.

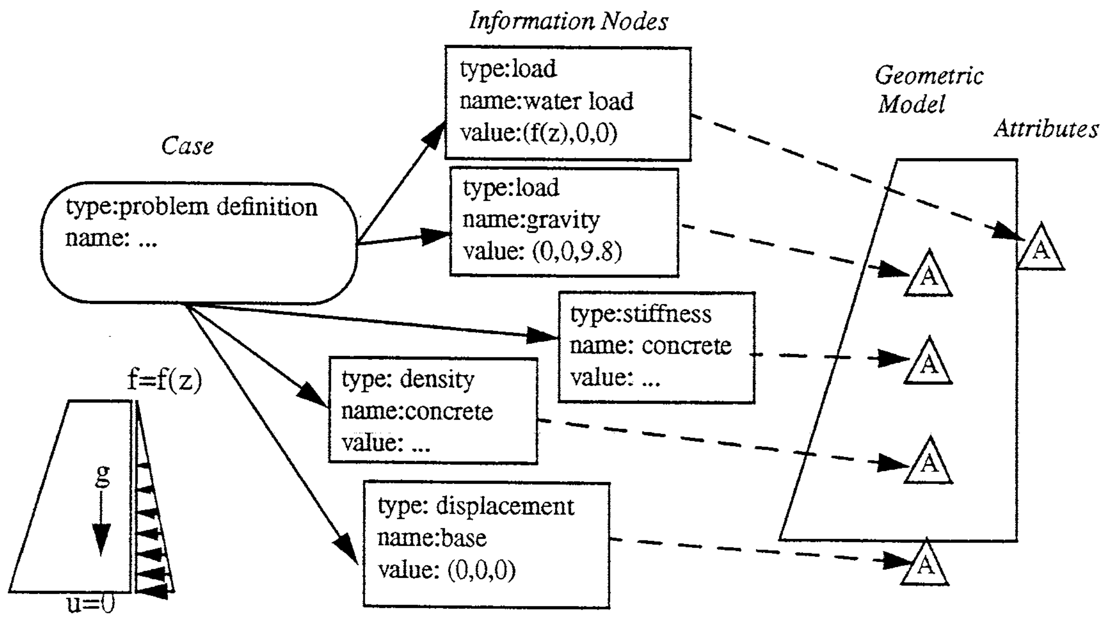

Figure 3. Attribute example.

Nodes in the attribute graph have another important property. They can represent an object that is to be created when the attribute graph is traversed. This object is called the image of the attribute and represents the run time interpretation of the information of the attribute node and its children.

Mesh

The representation used for a mesh is similar to that used for a geometric model (Beall & Shephard 1997). A hierarchy of regions, faces, edges and vertices makes up the mesh. In addition, each mesh entity maintains a relation, called the classification of the mesh entity, to the model entity that it was created to partially represent as indicated in Figures 2 and 4. This representation of the mesh is very useful for mesh adaptivity. Also understanding how the mesh relates to the geometric model allows an understanding of how the solution relates back to the original problem description. The topological representation can be used to great advantage in performing adaptive p-version analyses as polynomial orders can be directly assigned to the various entities (Shephard, Dey & Flaherty 1996).

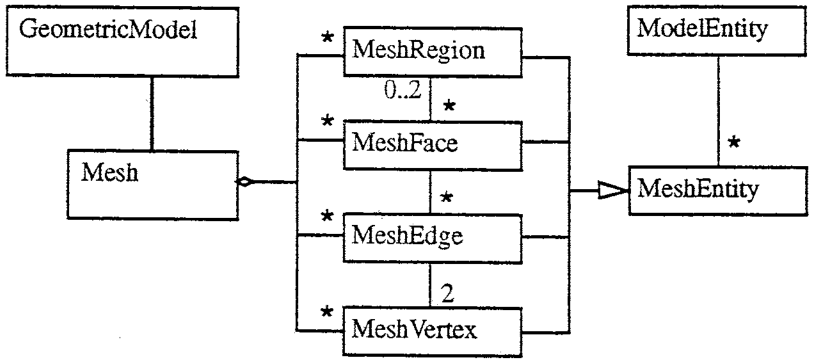

Figure 4. Mesh representation.

Field

A problem with many "classic" finite element codes is that the solution of an analysis is given in terms of the values at a certain set of discrete points (e.g. nodal locations or integration points). However the finite element discretization actually has more information than just the values at these points, there is also information about the interpolations that were used in the analysis. Therefore, when the standard process of storing just the discrete pointwise values is maintained, information is lost after the analysis is run. Without knowing the specifics of the analysis code it is impossible to reconstruct the interpolations used and one can not define the values at general locations. This makes it much more difficult to use the solution in a subsequent step in the analysis (e.g. error estimation, or as an attribute for another analysis). The analysis framework eliminates this problem by introducing a construct known as a field.

A field describes the variation of some tensor field over one or more entities in a geometric model. The spatial variation of the field is defined in terms of interpolations defined over a discrete representation of the geometric model entities, which is currently the finite element mesh. A field is a collection of individual interpolations, all of which are interpolating the same quantity (Figure 5). Each interpolation is associated with one or more entities in the discrete representation of the model.

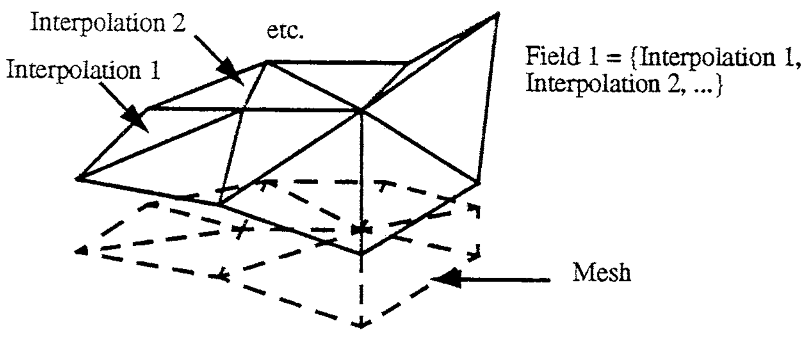

Figure 5. Example of a field.

GEOMETRY-BASED ANALYSIS PROCESSES

The framework represents the analysis process as a series of transformations of the problem from the original mathematical problem description through to the sets of algebraic equations approximately representing the problem (Figure 6). This transformation starts at the mathematical problem description level

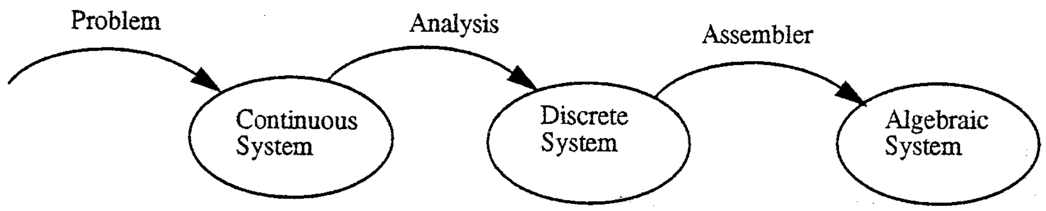

Figure 6. Analysis transformation process.

which contains the geometric model and the attributes which apply to that model. The attributes for a particular problem are specified by a particular case node in the attribute graph. All of the attributes under this case node are used for the given problem. An instance of a ContinuousSystem is then transformed to an instance of the class DiscreteSystem which represents the discretized version of the model and attributes and the weak form of the partial differential equation (PDE). This transformation is done by an object that is an instance of a class that is part of a hierarchy of analysis classes. The particular analysis class that is used depends on the selected weak form of the PDE to be solved.

For each problem definition it is possible to define any number of analyses. An analysis is defined by combining a problem definition with one or more cases that contain the rest of the information needed to perform the analysis. Here an analysis is defined by combining a problem definition case with a numeric case (which contains information relating to the specific numerical techniques used to solve the problem) and a meshing case (which contains information describing the parameters needed to generate a mesh for the model being used). The responsibilities of an Analysis class are to:

1. Create a DiscreteSystem of a type appropriate for the problem.
2. Interpret attributes associated with the geometric model and appropriately create StiffnessContributors, ForceContributors and EssentialBCs and add them to the DiscreteSystem.
3. Create an AlgebraicSystem with an appropriate solver.

4. Invoke the solve method of the AlgebraicSystem.

The DiscreteSystem class represents the problem in terms of contributions from a set of objects that live on the discrete representation of the model. These objects are called SystemContributors. There are three types of SystemContributors: StiffnessContributors contribute coupling terms between degrees of freedom of the system, ForceContributors contribute terms to the right hand side vector, Constraints set specific values to given degrees of freedom (e.g. setting the value of a certain degree of freedom to zero). The SystemContributors are created by the Analysis object and correspond to an interpretation of attributes consistent with the weak form that the Analysis implements. For example, in a heat transfer analysis, material property attributes will give rise to StiffnessContributors, applied heat fluxes will give rise to ForceContributors and prescribed temperatures will give rise to Constraints. Typically a SystemContributor corresponds to a mesh entity classified on the model entity where the attribute is applied.

The Analysis class creates all of the SystemContributors and adds them to an instance of a DiscreteSystem. There is a hierarchy of DiscreteSystem classes that represent different time orders of PDEs. This transformation of the problem from the ContinuousSystem to the DiscreteSystem allows the various solution routines to work on a representation that is independent of the type of problem being solved.

The next step in the solution process is to set up and solve the linear algebra. The setting up of the linear algebra consists of transforming a DiscreteSystem into an AlgebraicSystem. This transformation is handled by an Assembler object. Essentially an Assembler maps the contributions of each StiffnessContributor and ForceContributor in a DiscreteSystem into the correct entries in the global stiffness matrix and global force vector in an AlgebraicSystem.

Each type of operation that needs to form a global matrix or vector must use an assembler (either defining a new one or using an existing one). The base class Assembler provides the operations needed to do the perfrom the process of assembling the global system through it's assemble method (this method is only accessible to subclasses of Assembler). Each derived class must implement the operations that need to be carried out on the matrices returned by the ForceContributors and StiffnessContributors and then call the base classes assemble method.

REFERENCES

Beall, M. W.; Shephard, M. S. (1997): A general topology-based mesh data structure. *International Journal for Numerical Methods in Engineering*, to appear.

Shephard, M. S. (1988): The specification of physical attribute information for engineering analysis. *Engineering with Computers*, Vol. 4, pp. 145-155.

Shephard, M. S.; Dey, S.; Flaherty, J. E. (1996): A straightforward structure to construct shape functions for variable p-order meshes. *Computer Methods In Applied Mechanics and Engineering*, to appear.

Weiler, K. J. (1988): The radial-edge structure: A topological representation for non-manifold geometric boundary representations. M.J. Wozny, H.W. McLaughlin, J.L. Encarnacao, editors. *Geometric modeling for CAD applications*, North Holland, pp. 3-36.