Draft Working Note

Design of a Problem Taxonomy for Mathematical Software

Ronald F. Boisvert
Shirley Browne
Eric Grosse
Bruce Miller
W. Van Snyder

U. S. Department of Commerce
Technology Administration
National Institute of Standards and Technology
Computing and Applied Mathematics Laboratory
Gaithersburg, MD 20899 USA

March 1995

Draft

Design of a Problem Taxonomy for Mathematical Software

Ronald F. Boisvert*
Shirley Browne†
Eric Grosse‡
Bruce Miller*
W. Van Snyder§

March 1995

Abstract

This report proposes the design for a taxonomy of mathematical and statistical problems. Its purpose is to provide a basis for the development of automated systems for mathematical software selection. The taxonomy shares features of both hierarchical and faceted classification systems.

*Computing and Applied Mathematics Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899 (boisvert@nist.gov, miller@cam.nist.gov)

†Department of Computer Science, University of Tennessee at Knoxville, Knoxville, TN (browne@cs.utk.edu)

‡AT&T Bell Laboratories, 2T-504, Murray Hill, NJ 07974-0636 (ehg@research.att.com)

§Jet Propulsion Laboratory, Mail Stop 601-108, 4800 Oak Grove Drive, Pasadena, CA 91109 (vsnyder@math.jpl.nasa.gov)

1 Introduction

Automated systems for the selection of reusable mathematical software components must use a variety of indicators to determine a given user's requirements. Examples are:

- The problem to be solved.
- The algorithm to be employed.
- The arithmetic precision required.
- The language used to interact with the program.
- The hardware/software environment in which the software must run.

For each such indicator a controlled vocabulary must be developed, along with a means to represent relationships between terms. In this report we use the term *taxonomy* to describe such a system.

This report presents a design for a taxonomy of mathematical and statistical problems. The taxonomy is meant to provide a basis for the development of automated systems for mathematical software selection. By selecting appropriate taxonomic terms a user can indicate what problem needs to be solved, and by assigning taxonomic terms to software objects repository managers can succinctly describe the purpose of that object. We believe that the use of such techniques can greatly improve the precision of automated tools for resource location.

A number of mathematical software taxonomies and related searching aids have been developed in recent years. Among these are some very deep taxonomies for particular subfields, such as approximation [9] and differential equations [1]. Probably the most widely used scheme for general mathematical and statistical problems is the Guide to Available Mathematical Software (GAMS) classification system [4, 5].

The GAMS taxonomy had its beginnings with a coarse classification scheme for software developed by the IBM users' group (SHARE) in the 1960s. In 1975 John Bolstad at Stanford extended and refined this system to create a detailed hierarchical classification scheme for mathematical software [7]. This scheme was used as a starting point for the development of a revised and updated hierarchical system for use by the GAMS project at NIST [3, 4, 5, 6], and this system has seen widespread use by commercial software vendors (e.g. Visual Numerics Inc., NAG Ltd., C. Abaci) and scientific computing facilities and repositories (e.g., SLATEC Common Math Library, netlib, e-Lib, Lawrence Livermore National Laboratory, Amoco Research, University of Texas System High Performance Computing Center).

The GAMS classification system accurately reflects the state of mathematical software in the early 1980s. However, since that time much new software has emerged, expanding the range of problems amenable to computer solution, and rendering the existing scheme incomplete in a number of areas. In addition, the emergence of landmark software packages has, in some cases, changed our views on how certain fields should be organized. Also, the original scheme was developed in an ad hoc manner, resulting in a number of inherent design flaws. As a result, we feel that there is a need for a substantial revision of the existing scheme that both accurately reflects the current state of the art and is founded on carefully conceived principles that will allow it to be useful far into the future. To do this we believe that a completely new broad taxonomy must be devised.

2 Taxonomic Structures

A wide variety of schemes have been proposed for the development of taxonomies [8, 10]. These can be classified as enumerative or synthetic. *Enumerative* schemes attempt to provide explicitly for all conjunctions and modifications of terms to be allowed. *Synthetic* schemes provide a set of fundamental terms, together with rules for combining them into the complex concepts required for the system.

A purely enumerative scheme tends to be bulky, rigid, and repetitive. It requires an item to be put in a pigeonhole closest to the subject of the item, and often lacks a facility for providing for unanticipated subjects. On the other hand, synthetic systems place a severe burden on users of having to break down a question into simpler components and learning the rules for combining them.

Examples of some basic taxonomic structures are the following.

Hierarchical classification
A tree-structured enumerative classification system in which traversing from parent to child nodes progresses from broad terms to narrower terms. An object is classified by specifying one (or more) nodes which best describe the object.

Faceted classification
A system in which a fixed number of broad classes (or *facets*) are specified. One can think of the facets as the axes of a multidimensional space. Each facet has an associated list of terms. An object is classified by picking one term from each facet. Faceted schemes share aspects of both enumerative and synthetic terminologies.

Thesauruses
A purely synthetic system composed of a list of terms, each linked to other terms in a set of predefined ways, such as *broader term*, *narrower term*, *related term*, *preferred term*. Standards have been developed for the construction and presentation of thesauruses [2].

Taxonomies used in practice are often hybrids of such basic structures. The use of synthetic features together with a hierarchical classification permits index terms to more precisely reflect the content of the item, while maintaining some of the ease of use of traditional schemes.

3 Design Principles

In this section we outline the basic design principles for the problem taxonomy we develop.

3.1 Scope

Eventually we expect to see a large problem taxonomy for computer software covering all disciplines. The present effort is an attempt to build a small portion of this system useful for indexing general-purpose mathematical and statistical software. As a result, the taxonomy will not encompass all of mathematics, and neither will it address specific application areas such as fluid mechanics or mathematical physics. However, we will include certain other areas representing utilities commonly distributed with scientific software packages and libraries, such as sorting, error handling, graphics, and parallel processing tools. We will endeavor to design an overall structure that will be naturally extensible to other areas.

The primary focus of the taxonomy will be mathematical problems rather than the methods used to solve them. In particular, it will not differentiate between those problems solved numerically and those solved by symbolic manipulation.

Finally, we expect that the taxonomy will be used to classify a wide variety of objects, including individual library subroutines, integrated software packages, and individual commands in an interactive system.

3.2 Structure

The problem taxonomy we develop will be a heterogeneous one, utilizing a variety of techniques combined in a way best suited to this task.

The system will be anchored by a hierarchical problem classification scheme. A hierarchy is easy to understand and use. Its structure can naturally follow from well-known organization of mathematical disciplines, and it can be used as a rudimentary decision tree. However, it is not our intention that the hierarchy itself be exhaustive in the sense that it completely partitions any particular software collection into unique classes. Not only would such a system be too large and unwieldy, but it would also be tied too closely to current software to be usable far into the future. In addition, once a hierarchy becomes too deep the natural mathematical organization is lost, forcing arbitrary decisions about what problem features will be represented at each level of the tree.

As a result, the hierarchy we build will be fairly coarse. We will supplement each leaf node by a synthetic facet-based classification, so that items within a given problem class may be indexed more precisely. The problem hierarchy will not have a fixed number of levels. The decision of when to switch from the tree structure to a faceted structure will be made separately for each problem area based on an assessment of when the tree is no longer natural or intuitive.

3.2.1 Problem Hierarchy

Each node in the problem hierarchy corresponds to a class of mathematical problems. The hierarchy is strictly tree-structured. Each node has a single parent representing the more general problem class, and zero or more children representing more specific instances. (For example, *Linear Algebra* $\rightarrow$ *Matrix Eigenvalue Problems* is a parent-child link.) Both verbose (i.e., complete) and terse (i.e., in context) headings, for nodes will be provided, as well as a short description of the problems represented by each class.

We assume that users of the taxonomy will admit multiple classifications of a single software item if necessary. We also expect that software items will be classified at any node in the tree, rather than simply at the leaves. This follows from the wide variety of software objects that could be classified. A library subprogram for a very specific problem might well be classified at a leaf node, while a comprehensive package solving many problems in a particular subfield might be classified at a higher-level node which represents the entire subfield.

Each node is assigned a tag so that it can be referred to succinctly. It is important that the notation be flexible enough to allow arbitrary insertions of new nodes in the future. For this purpose we will use a sequence of non-negative numbers separated by dashes. The numbers indicate the path from the root to the given node. Although the taxonomy we develop here will use only integers it is our intention to admit real numbers to facilitate arbitrary insertions. For completeness, each level has one additional class named *Other*, which is reserved for software belonging in the parent class, but not to any of the subclasses. This class is always numbered 0, although we list it last when displaying it with its siblings. An example of this notation is given in Table 1.

Although a hierarchical structure is quite natural and suffices in many cases, the field of mathematics is not strictly tree-structured. We believe that the taxonomy should reflect this situation. One way to accomplish this is to introduce additional sets of links between nodes. Such additional references are commonly employed, and can be classified as either permissive or prescriptive. A *prescriptive reference* prohibits use of one term, requiring use of another instead, whereas a *permissive reference* suggests that a related term may be of value for indexing or searching.

A common example of a permissive reference is a *See also* link, e.g.

```
1-7-3 Least Squares Approximation.  See also 2-5 (Regression)
```

For a problem-oriented taxonomy it is possible to associate a more specific meaning to such links. We define *reformulation links* as follows : $A \rightarrow B$ if problem A can be reformulated as problem B. Such a link implies that it is possible, though possibly non-intuitive, to use

Table 1: An example problem hierarchy.

1	Mathematics
1-1	Arithmetic
1-2	Algebra
1-3	Linear Algebra
1-3-1	Vector operations
1-3-2	Matrix/vector operations
1-3-3	Matrix operations
1-3-4	Tensor operations
1-3-5	Solution of linear systems
1-3-6	Other matrix equations
1-3-7	Eigenvalue problems
1-3-0	Other
1-4	Mathematical Functions
1-4-1	Elementary functions
1-4-1-1	integer-valued functions
1-4-1-2	powers and roots
1-4-1-0	other
1-4-2	Polynomials and related functions
1-4-2-1	polynomials
1-4-2-2	piecewise polynomials
1-4-2-3	rational polynomials
1-4-2-0	other
1-4-3	Trigonometric functions
1-4-4	Special functions
1-4-5	Integral transforms
1-4-5-1	Fourier
1-4-5-2	Laplace
1-4-5-3	Hilbert
1-4-5-0	other
1-4-6	Approximation of functions
1-4-0	Other
1-0	Other
0	Other

software for problem B to solve problems of type A. We will provide a set of reformulation links with the taxonomy.

Permissive references can be used to eliminate duplicate subtrees. Here one trims all but one of the duplicate, replacing them with special *referral nodes* which point the user to the prefered node for this problem. No items can be classified at a referral node. We will consider adding such links.

In summary, the following specific items will be associated with each node in the problem hierarchy.

TAG
A unique identifier of the form n_1-n_2-...-n_k, where each n_i is a real number. The successive initial subsequences of this tag are the tags of nodes traversed in the hierarchy from the root to this node. The citation order for sibling nodes is by increasing value of last component.

TERSE HEADING
A word or brief phrase describing the problem, meant for use when listed in context with its parent and sibling nodes, i.e. as in the example listing above.

VERBOSE HEADING
A comprehensive descriptive phrase, meant to be used when the context of the given class is not apparent.

DESCRIPTION
A short text that provides further elaboration to those attempting to understand the purpose of the given class. (Since these texts are meant to be read in the context of the hierarchy, information provided in all parental nodes can be assumed to be known to avoid repetition.)

PROBLEM SPECIALIZATION LINKS.
A list of tags of children of this class in the problem hierarchy. $A \rightarrow B$ if problem B is a specific instance of problem A.

REFORMULATION LINKS
A list of tags representing reformulation targets for this class. $A \rightarrow B$ if problem A can be reformulated as problem B.

3.2.2 Problem Facets

Associated with each leaf node in the problem hierarchy will be a set of broad concepts, called *facets*, which can be used to discriminate among the specialized problems in this class. Each facet will have a set of mutually exclusive *terms* representing possible values of the

Table 2: An example of facets related to matrices.

Facet		Terms	
A	Arithmetic Field	1	complex
		2	real
B	Matrix Block Structure	1	blocked
		2	unblocked
C	Matrix Symmetry	1	symmetric
		2	Hermitian
		3	nonsymmetric
D	Matrix Definiteness	1	positive or negative definite
		2	positive or negative semidefinite
		3	indefinite
E	Matrix Nonzero Structure	1	dense
		2	banded
		3	Hessenberg
		4	triangular
		5	tridiagonal
		6	diagonal
		7	sparse
F	Special Matrices	1	circulant
		2	Toeplitz
		3	Vandermonde
G	Matrix Storage Format	1	regular array
		2	packed symmetric
		3	diagonal (band)
		4	profile
		5	sparse – coordinate
		6	sparse – packed row
		7	sparse – packed column
		8	none – operator provided
		9	other

facet. To classify an item using such a scheme one picks a unique term from each of the facets.

For example, Table 2 illustrates a set of facets which might be associated with problems involving matrices, such as the solution of linear systems or simple eigenvalue problems. The facets of this example define a 7-dimensional classification space with a $2 \times 2 \times 3 \times 3 \times 7 \times 4 \times 9$ grid yielding 9,072 distinct classifications. Although not all of these will necessarily make sense, this provides a convenient way of consisely expressing a large number of concepts.

An arbitrary but fixed ordering for the facets of a given node and their terms will be established in order to facilitate development of a succinct notation for expressing classifications. In particular, the facets will be labeled with letters (i.e., A, B, ..., Z, AA, AB, ...) and their values with positive integers. This is illustrated in Table 2. To encode a set of selected facet terms into a single tag one concatenates each facet label with the label of its selected term and then strings them all out, separated by dashes, e.g. A2-B2-C3-D3-E1-F4-G1. We always write these tags in alphabetical order of the facet labels. Note that this implies that we could drop the facet labels in the tags without loss; however, we believe that they aid readability and hence should be retained.

In spite of our attempts to avoid it, cases where no terms or multiple terms are applicable are inevitable. In the former case, the concept represented by the facet itself may not be applicable; in the latter case the software being classified might have options representing several different terms. To express the fact that no term applies one selects 0 as the term label. If multiple terms are applicable, then multiple classifications are assigned; for example, if there are four facets in which both terms 1 and 2 apply for facet A, term 1 applies for facet B, no terms apply for facet C, and term 4 applies for facet D, the following facet tags are assigned:

```
A1-B1-C0-D4
A2-B1-C0-D4
```

To provide a complete classification tag, one encodes the problem hierarchy tag, followed by a slash, followed by the facet tag. For example, using the examples in Tables 1 and 2, one would encode the classification of the LAPACK routine CPBSV as

```
1-3-5/A1-B2-C2-D1-E2-F0-G3,
```

which translates to *Mathematics – Linear Algebra – Solution of Linear Systems / real – unblocked – Hermitian – positive or negative definite – banded – (not applicable) - diagonal (band) storage format.*

Note that since there is no logical ordering for the facets or their terms it is not necessary that the notation admit arbitrary insertions. Future insertions can simply be made at the end of the facet and term lists.

Table 3: An example of facets related to PDE boundary value problems.

Facet		Terms	
A	Arithmetic Field	1	real
		2	complex
B	Number of equations	1	one
		2	coupled system
C	Principal order	1	second
		2	fourth
		3	higher
D	Operator type	1	Laplace/Poisson/Hemlholtz/biharmonic
		2	linear constant coefficient
		3	generalized Helmholtz (i.e., $[\Delta + f]u = g$)
		4	separable (e.g., $[L_x + L_y]u = g$)
		5	self-adjoint (e.g., $(pu_x)_x + (qu_y)_y + ru = g$)
		6	linear variable coefficient
		7	quasilinear
		8	nonlinear
E	Domain	1	2D - rectangular
		2	2D - circular (including annular)
		3	2D - polygonal
		4	2D - surface spherical
		5	2D - irregular
		6	2D - other
		7	3D - box
		8	3D - cylinder (including annular)
		9	3D - sphere (including annular)
		10	3D - polygonal
		11	3D - irregular
		12	3D - other
		13	higher dimensions
F	Boundary Conditions	1	Dirichlet (solution specified)
		2	Neumann (flux specified)
		3	periodic
		4	Dirichlet+Neumann
		5	Dirichlet+Neumann+periodic
		6	linear constant coefficient
		7	linear variable coefficient
		8	quasilinear
		9	nonlinear

Several difficulties are still present in the faceted system proposed here. A particularly difficult one results from relationships among terms within a single facet. This is illustrated in Table 3 which suggests facets related to partial differential equation (PDE) boundary value problems. These facets do a fair job of mapping out the elementary concepts in this domain. However, consider facet D which describes the differential operator. It is certainly the case that software which handles nonlinear operators (D8) can also handle any of the other operator types. Thus, a classifier for such software might well be tempted to assign multiple classifications which enumerate all terms in this facet.

One way to circumvent such difficulties is to admit reformulation links among the terms of a given facet. For facet D in this example we would encode the following.

D1	(Laplace/Poisson/Hemlholtz/biharmonic)	$\rightarrow$	D2
D1		$\rightarrow$	D3
D1		$\rightarrow$	D4
D1		$\rightarrow$	D5
D2	(linear constant coefficient)	$\rightarrow$	D6
D3	(generalized Helmholtz)	$\rightarrow$	D5
D4	(separable)	$\rightarrow$	D5
D5	(self-adjoint)	$\rightarrow$	D6
D6	(linear variable coefficient)	$\rightarrow$	D7
D7	(quasilinear)	$\rightarrow$	D8

Note that this only includes the "closest" problem reformulations, since all others are available by transitivity. Similar networks of relations can be constructed for facets A, B, E and F of this example.

The taxonomy we develop will admit reformulation links among the terms of a given facet. This will allow classifiers to select a single most appropriate classification in most cases. Systems which use the taxonomy will be able to provide search facilities to follow the reformulation links if necessary.

Designing a good set of facets requires considerable care. In the facets of Table 2, for example, it might seem reasonable to add the terms *symmetric* and *Hermitian* to the Special Matrices facet and then delete Matrix Symmetry. However, the resulting set of terms would not be mutually exclusive (there are symmetric Toeplitz matrices). For the same reason the concept of matrix diagonal dominance would have to be a new facet rather than a term in any of the existing ones.

Another illustration of this difficulty is seen if one tries to add the new facet of Table 4 to the set of facets in Table 3. The terms of this new facet enumerate a variety of special features which occur in practical boundary value problems. Strictly speaking, they are not mutually exclusive and hence such a facet should be avoided. However, it is tempting to admit such facets because each of the terms is relatively specialized (in comparision with the other facets), and the alternative would be the introduction of seven new facets of limited applicability.

Table 4: An additional facet for PDE boundary value problems.

Facet		Terms	
G	Special feature	1	holes or lines in domain
		2	internal interfaces
		3	discontinuous coefficients
		4	exterior problem
		5	eigenvalue problem
		6	sharp gradients in solution
		7	free boundary

Table 5: Additional facets related to algorithms for PDE boundary value problems.

Facet		Terms	
H	Operator discretization	1	finite difference
		2	finite element
		3	boundary integral or element
		4	spectral
		5	wavelet
		6	other
I	Domain discretization	1	uniform rectangular
		2	irregular rectangular
		3	triangular or tetrahedral
		4	other
J	Grid type	1	fixed
		2	adaptive
K	Linear system solver	1	fast direct method (e.g., Fourier, FACR)
		2	direct solver
		3	iterative solver
		4	multigrid
		5	hybrid
		6	other

Another issue which must be addressed is whether facets which describe algorithms should be admitted. An example is illustrated in the facets of Table 5. These facets represent methods used to solve elliptic boundary value problems. Although 288 methods are encoded by this set of facets it still only scratches the surface of possible methods. Several difficulties are apparent.

- Such facets are inconsistent in nature with the rest of the taxonomy, which describe *problems*.
- Representing algorithms in the taxonomy will yield a large number of additional facets, increasing the burden on both the developers and classifiers.
- Algorithms have a much shorter life span that the problems which they solve, and hence this part of the taxonomy will be subject to greater maintenance.

It would seem that developing a systematic controlled vocabulary may not be the best way to allow search for algorithms. These might be better encoded as a set of keywords describing individual software objects. Thus, adding facets which related to algorithms will be avoided whenever possible.

4 Topmost Levels of the Hierarchy

In this section we propose an organization for the topmost levels of the hierarchy. This is presented to encourage discussion, and as a way to illustrate some of the associated materials.

```
1   Mathematics
1-1   Arithmetic
1-2   Algebra and Number Theory
1-3   Linear Algebra
1-4   Discrete Mathematics
1-5   Function Evaluation
1-6   Geometry
1-7   Data Analysis
1-8   Nonlinear Equations
1-9   Calculus
1-10  Functional Equations (e.g. Differential, Integral)
1-11  Optimization and Control
1-0   Other

2   Statistics
2-1   Elementary Data Handling and Analysis
2-2   Function Evaluation
2-3   Random Number Generation
```

```
2-4   Analysis of Variance
2-5   Regression
2-6   Categorical Data Analysis
2-7   Time Series Analysis
2-8   Correlation Analysis
2-9   Discriminant Analysis
2-10  Covariance Structure Models
2-11  Cluster Analysis
2-12  Life Testing, Survival Analysis
2-13  Multidimensional Scaling
2-0   Other

3   Utilities and Support (Scientific software}
3-1   Portability Support
3-2   Data Handling
3-3   Parallel and Vector Processing Tools
3-4   Software Development Tools
3-5   Graphics and Visualization
3-6   Benchmarks and Testing
3-7   Other
```

5 Discussion of the Proposed Hierarchy

In this section we briefly touch on the rationale behind each class in the coarse hierarchy of the previous section, including suggestions for possible subclasses.

1 Mathematics

1-1 Arithmetic

Software for implementing elementary arithmetic operations, especially on non-standard data types, such as extended precision, extended range, or interval arithmetic. Note that this class is *not* meant for programs that simply use special arithmetics for specific applications such as solving linear systems or evaluating special functions; these should be classified under software for those problems instead.

Example subcategories include

```
Special arithmetics
  complex
  rational
  multiple precision
  extended range
```

```
  interval
Change of representation
Arithmetic environment inquiry (See Portability Support)
```

1-2 Algebra and Number Theory

Contains software for elementary algebraic operations (e.g. simplifying algebraic expressions), as well as computations associated with abstract algebra and number theory (e.g. prime factorizations).

Example subcategories include

```
Elementary algebraic manipulation
Abstract algebra
Number theory
```

1-3 Linear Algebra

This section is reserved for vector and matrix operations, including the solution of systems of linear equations and matrix eigenvalue-eigenvector decompositions.

Example subcategories include

```
Vector operations
  elementary vector algebra (e.g., copy, scale)
  metrics (e.g., norms, distances)
  special transformations (e.g., rotations, reflections)
  outer product
Matrix-vector operations
  matrix-vector multiply
  bilinear mode (y-transpose)*A*x
Matrix operations
  elementary (e.g., copy, scale)
  reorderings
  low-rank update
  decompositions and normal forms
  orthogonalization
  determinants
  inverses
  generalized inverses
  other functions of matrices (e.g., exp(A))
Tensor operations
Solution of linear systems
  nonsingular
  singular
Eigenvalue problems
```

```
  ordinary eigenvalue problems
  generalized eigenvalue problems
  associated operations
Other matrix equations
```

1-4 Discrete Mathematics

Contains software for combinatorial algorithms, graph theory, string matching, etc. Includes discrete optimization problems.

Example subcategories include

```
Difference equations
Combinatorics
Graph theory
Combinatorial optimization
   Integer programming
   Network optimization
   Transportation or assignment problems
String matching
```

1-5 Function Evaluation

Contains software for evaluating or manipulating both elementary functions and the special functions of applied mathematics. This includes software for manipulating piecewise polynomials (splines). Also included is software for determining best approximations to functions. (Software for determining functions that interpolate or approximate discrete data is found under Data Analysis.) Finally, software for computing continuous integral transforms is found here.

Possible overlaps : Evaluation of statistical distributions should go under Statistics. Some functions, such as erf, might need to be classified in both places. Software for computing derivatives, integrals, and zeroes of specific functions (e.g. Bessel and Airy functions) should be classified here rather than under differentiation, integration, or zero finding.

Example subcategories include

```
Mathematical constants
Elementary functions
  integer-valued functions
  powers and roots
Polynomials, piecewise polynomials and related functions
  polynomials
  piecewise polynomials
  rational polynomials
Trigonometric functions
```

```
Special functions
Integral transforms
  Fourier transforms
  Laplace transforms
  Hilbert transforms
Approximation of functions
```

1-6 Geometry

Contains software for fundamental geometric calculations, such as intersections, projections, areas, and volumes. Also includes convex hulls, triangulations, and Voronoi diagrams.

Possible overlaps : Fitting of curves and surfaces, while important in computer-aided geometric design, also falls under Data Analysis.

Example subcategories include

```
Elementary geometry
   Intersections
   Nearest point, neighbors
   Coordinate transformations
   Rotations
   Projections
   Areas and volumes
Conformal mappings
Grid generation
Geometric modeling
```

1-7 Data Analysis

Contains software for processing of discrete data. Typical operations include interpolation, approximation, discrete integral transforms, smoothing, filtering and encoding.

Possible overlaps : This class has a significant overlap with Statistics. However, software classified here is not statistical in nature. In some cases software will be cross-classified.

Example subcategories include

```
Transformation
  integral transforms
  smoothing
  filtering
  encoding
Interpolation
  curve fitting
  surface fitting
```

```
Approximation
  curve fitting
  surface fitting
```

1-8 Nonlinear Equations

This category is meant for software that solves nonlinear equations (both single equations and systems of equations). This does *not* include functional equations such as differential and integral equations which have their own category.

Example subcategories include

```
Single equation
  polynomial equation
  nonpolynomial equation
System of equations
  systems depending on a parameter
    polynomial equations
    nonpolynomial equations
  nonparametric systems
    polynomial equations
    nonpolynomial equations
```

1-9 Calculus

Contains software for manipulating sequences and series (e.g. limits, summation, convergence acceleration). Also included is software for the fundamental operations of the calculus: differentiation and integration.

Example subcategories include

```
Sequences
   generation, recurrences
   limits
   convergence acceleration
Series
   expansions
   summation
Differentiation
Integration
   one dimensional
   multidimensional
   line integrals
   surface integrals
Vector calculus
```

1-10 **Functional Equations (e.g., Differential, Integral)**

Contains software for the solution of differential equations, integral equations, and other functional equations (e.g. difference equations, integro-differential equations). Note that we expect that eigenvalue problems will appear as subclasses under the basic equation type, e.g. boundary value problems (for Sturm-Liouville problems, for example) or under Partial differential equations—elliptic.

Example subcategories include

```
Ordinary differential equations
  initial value problems
  boundary value problems
Partial differential equations
  initial boundary value problems
  boundary value problems
Integral equations
Integro-differential equations
```

1-11 **Optimization and Control**

Contains software for optimization (e.g. minimizing a function with or without constraints). Included are linear programming, quadratic programming and nonlinear programming. Also included is software for optimal control.

Example subcategories include

```
Unconstrained optimization
  univariate
  multivariate
Constrained optimization
  linear programming
  integer and mixed-integer programming
  quadratic programming
  nonlinear programming
Optimal control
```

2 **Statistics**

The statistics chapter of the GAMS classification system was substantially revised in 1990, and is hence more up-to-date than other sections. As a result, we expect to be able to import major portions of this hierarchy directly. The changes we propose are only to allow this section to follow our new guidelines. In particular, we propose to

- reduce the number of classifications at the topmost level by collecting the first four under a new heading,

- delete the last subclass (Statistical Data Sets), since this belongs under Utilities,
- convert deep parts of the hierarchy to faceted form, and
- eliminate classes titled *Service routines* since the implicit *Other* classification can be used for this purpose.

The following lists the resulting top two levels of this subtree.

```
Elementary data handling and analysis
  Data summarization
  Data manipulation
  Elementary statistical graphics
  Elementary data analysis
Function evaluation
  Univariate
  Multivariate
Random number generation
  Univariate
  Multivariate
Analysis of variance (including analysis of covariance)
  One-way
  Two-way
  Three-way (e.g., Latin squares)
  Multi-way
  Multivariate
  Generate experimental designs
Regression
  Simple linear (i.e., $y = b_0 + b_1 x$)
  Polynomial (e.g., $y = b_0 + b_1 x + b_2 x^2$)
  Multiple linear (i.e., $y = b_0 + b_1 x_1 + ... + b_p x_p$)
  Polynomial in several variables
  Nonlinear (i.e., $y = F(X,b)$)
  Simultaneous (i.e., $Y = Xb$)
  Spline (i.e., piecewise polynomial)
  EDA (e.g., smoothing)
Categorical data analysis
  2-by-2 tables
  Two-way tables
  Log-linear model
  EDA (e.g., median polish)
Time series analysis
  Univariate
  Two time series
```

```
    Multivariate time series
    Two multi-channel time series
  Correlation analysis
  Discriminant analysis
  Covariance structure models
    Factor analysis
    Principal components analysis
    Canonical correlation
  Cluster analysis
    One-way
    Two-way
    Display
  Life testing, survival analysis
  Multidimensional scaling
```

3 Utilities and Support (Scientific Software)

This class contains software which is not mathematical or statistical software per se. However, the software here plays an important supporting function in many mathematical software packages, and is often included in mathematical software collections.

3-1 Portability Support

Included here are a variety of low-level utility functions whose use aid in the portability of programs.

Example subcategories include

```
Environmental inquiry
Mathematical and physical constants
Error processing
```

3-2 Data Handling

This category includes elementary data manipulation that is not commonly considered mathematical in nature, such as I/O, storage management, sorting and searching.

Example subcategories include

```
Input and output
Storage management
Sorting
Merging
Searching
Permuting
```

3-3 **Parallel and Vector Processing Tools**

Included here are tools and utilities that support the development of software for vector and parallel processors. Message-passing utilities for distributed-memory multiprocessors and gather/scatter routines for vector processors are examples.

```
Vector processing utilities
Message passing systems
```

3-4 **Software Development Tools**

This category includes software development tools of particular interest to the scientific computing community. Examples are Fortran syntax and standards checkers and macro processors.

Example subcategories include

```
Static analysis
Dynamic analysis
Transformation
```

3-5 **Graphics and Visualization**

Graphics and visualization are topics that deserve a taxonomy of their own. We provide this category here so that utilities made available as part of mathematical software packages can be classified.

3-6 **Benchmarks and Testing**

This category is reserved for benchmarks of particular interest to the mathematical software community, as well as for test data generators, and testing systems for mathematical software.

Example subcategories include

```
Data sets
Benchmark suites
Testing systems
```

6 Implementation Plan

Our plan for the development of the new taxonomy is as follows.

1. Develop a proposal for the overall structure of the taxonomy, as well as a proposal for the top two levels of the hierarchy. *(This document)*

2. Circulate the proposal among a working group to solicit comments.

3. Set the top two levels, and solicit volunteers to refine the various chapters.

4. Collect together the proposed refinements and circulate among the working group for comments and revisions.

5. Publish the working version in a variety of forums, including NA-NET, the SIGNUM newsletter, *netlib*, etc. to solicit comments.

6. Revise the taxonomy based upon public comment, circulate among the working group, and settle on a final version.

7. Publish the final version in a widely circulated journal such as the ACM Transactions on Mathematical Software.

8. NIST will act as a repository for classification of software under the new taxonomy.

9. Petitions for minor modifications of the scheme will be sent to NIST who will circulate them for consideration by the working group. If there is a consensus that the modification is both minor and worthwhile, then it will be incorporated into the next revision of the scheme. Revisions will be issued no more frequently than once per year.

7 Outstanding Issues

Some issue brought up in this report are still worthy of discussion. These include the following.

1. Should prescriptive references be allowed in the hierarchy?

2. What conventions should be used in generating facets? For example, we have suggested that terms of a given facet be mutually exclusive in a week sense, i.e. that a unique choice of terms will be appropriate in most cases. Terms which are specializations of other terms are allowed, and links showing these inclusions are being allowed. An alternative would be to eliminate these rules an introduce a notation to allow more arbitrary types of term compositions in a classification. The former is more difficult to enforce and yields a larger number of facets, but is more elegant and yields a simpler notation.

3. Should facets describing algorithms be allowed?

Acknowledgments

The following additional individuals contributed to the development of this design : Daniel W. Lozier, Marjorie McClain, and William Mitchell (NIST); and Ian Gladwell (Southern

Methodist University), David Handscomb (NAG, Ltd.), and Tom Rowan (Oak Ridge National Laboratory).

References

[1] C. A. Addison, W. H. Enright, P. W. Gaffney, I. Gladwell, and P. M. Hanson. Algorithm 687: A decision tree for the numerical solution of initial value ordinary differential equations. *ACM Trans. Math. Softw.*, 17(1):1–10, Mar. 1991.

[2] American National Standards Institute, New York. *Guidelines for Thesaurus Structure, Construction, and Use*, 1974. (ANS Z39.19-1974).

[3] R. F. Boisvert, S. E. Howe, and D. K. Kahaner. Guide to Available Mathematical Software. NBSIR 84-2824, National Bureau of Standards, 1984. Also available as PB84-171305 from the National Technical Information Service (NTIS), Springfield, VA 22161.

[4] R. F. Boisvert, S. E. Howe, and D. K. Kahaner. GAMS—a framework for the management of scientific software. *ACM Trans. Math. Softw.*, 11:313–355, 1985.

[5] R. F. Boisvert, S. E. Howe, and D. K. Kahaner. The Guide to Available Mathematical Software problem classification system. *Comm. Stat. – Simul. Comp.*, 20(4):811–842, 1991.

[6] R. F. Boisvert, S. E. Howe, D. K. Kahaner, and J. L. Springmann. Guide to Available Mathematical Software. NISTIR 90-4237, National Institute of Standards and Technology, 1990. Also available as PB90-216508/AS, National Technical Information Service, Springfield, VA 22161.

[7] J. Bolstad. A proposed classification scheme for computer program libraries. *SIGNUM Newsletter*, 10(2–3):32–39, 1975.

[8] E. D. Dym, editor. *Subject and Information Analysis.* Marcel Dekker, Inc., New York, 1985.

[9] E. Grosse. A catalogue of algorithms for approximation. In J. Mason and M. Cox, editors, *Algorithms for Approximation II*, pages 479–514 (of 514), London, 1990. Chapman and Hall.

[10] J. L. Milstead, editor. *Subject Access Systems.* Academic Press, Orlando, FL, 1984.