TACC Technical Report TR-09-01

An on-demand Test Problem Server

Victor Eijkhout, Bill Barth, James Kneeland, Steve Mock, John Peterson

This work is supported by NSF grant 123456789

Abstract

Matrix algorithms, especially sparse matrix methods, play an important role in computational science. The researchers that develop such algorithms (such as linear system solvers, eigensolvers, mathematical optimization, and graph algorithsm) require test matrices that are, ideally, drawn from a large, varied, and realistic test collection. Unfortunately, existing collections (e.g., the Harwell-Boeing collection, Matrix Market, and the University of Florida collection) are often old, contain matrices that are unrealistically small for current computers, or are otherwise limited.

We describe the design of a dynamic test collection based on solicited research and production codes, hosted at TACC (the Texas Advanced Computing Center of The University of Texas at Austin). These codes will be made available in the form of a test problem server, run on demand with properties (both size and problem parameters) specified by the user. Also, they will be used to generate a static test collection of large, precomputed test problems that can be searched through an extensive repertoire of properties.

In both cases, the matrices will be made available via the web for download, as a TeraGrid Data Collection accessible at high speed from other TeraGrid resources, and directly from within codes running on TACC's TeraGrid resources. Stored matrices will be accessible through a database that allows searching on a wide range of numerical features.

Part I

Overview

1 Introduction

Numerical linear algebra is generally recognized to be of great value to computational science. Many codes in physics, engineering, structural mechanics, and other disciplines, have linear system solvers, eigenvalue calculations, optimization problems, or graph algorithms at their core. Correspondingly, many computer scientists and numerical analysts are involved in refining existing algorithms and finding new algorithms that have favourable time or space complexity. Papers reporting this kind of research typically use problems that are either synthetically generated or taken from publicly available collections of test matrices.

The difficulty with this is two-fold. First, test problems found in public collections, while often from realistic applications, offer only a single instance of the problems that application produces: no problems are available that are only a small variation on the given one. On the other hand, synthetic test problems, generated especially for demonstrating the efficacy of algorithm research, are typically fairly simple. In particular, they tend to use simple geometries. This opens up some researchers to criticism for the use of unrealistically simple test cases.

> We aim to be a resource for algorithm researchers by providing a *test problem server*, based on research and production codes, which generates test problems on demand, from user-supplied parameters.

In this report, we first justify in detail the need for a test server as envisioned here, then we describe the implementation of our solution.

Part II

Justification

In this section we take stock of some of the types of research that would be serviced by a properly set-up test collection as we are proposing. In Section 2 we briefly discuss various research areas, and in Section 3 we summarize the properties of an ideal test collection that would support that research.

Although many different application areas give rise to similar mathematical problems, for instance of the types described below, the design of our collection is aware of the needs of different areas. While we have certain areas that we target initially (CFD, acoustics, electro-magnetics), we are searching actively for different problem types from different parts of science (life sciences, materials and nano-physics, bio-informatics) and we seek to acquire problem generators for these areas.

2 Research enabled by test collections

Test matrices are an indispensable resource for various branches of algorithm research, such as sparse factorizations, preconditioners, iterative methods for linear systems, graph algorithms, eigenvalue solvers, and distributed computing. This is apparent from a bibliography of 170 papers, mostly relating to these topics, that used the Matrix Market test collection [22]. As this research has to prove its relevance in a highly parallel environment, the availability of large matrices, and of families of related matrices of increasing size, becomes crucial.

In the next few subsections we will address the demands on collections of test matrices as they arise from various areas of research. It is not an exaggeration to state that existing test collections are quite inadequate in meeting these demands, while our collection is set up to address them. Furthermore, we actively seek input from the research community on research direction and application areas that need to be addressed by our server.

Application areas In the following sections we will describe research by type of linear algebra problem. However, in the setup of the test server we also bear in mind the differences that come from various application areas, as well as modeling schemes. To maintain the relevance of our test server we invite and seek out interaction with the community.

2.1 Sparse Linear System Solvers

Solvers for sparse linear systems of equations fall into two categories.

Direct solvers depend largely on the sparsity structure of the matrix. An analysis of the nonzero structure is used to minimize fill-in, thereby reducing runtime and storage demands of the solver [13]. This behaviour of the solver is to an extent analyzable with graph-theoretical methods (see, for instance, [19, 20]) in terms of the underlying problem parameters, most notably the mesh size h of the discretization for problems arising from PDE simulation problems. Therefore, for testing of solvers it is valuable to have several collections of matrices that come from increasing refinements of the same discretization. Note that only in the symmetric positive definite case can the solver be fully analyzed in terms of the matrix graph; in the general case, where pivoting is needed, graph analysis is still an important tool.

Iterative solvers do not have the storage problems that direct solvers have. On the other hand, they are very sensitive to numerical properties of the matrix. Consequently, for testing of both iterative methods and preconditioners it is crucial to have realistic test matrices. Since a theoretical dependency of the method's behaviour on matrix properties can often be identified [37], the availability of sets of matrices with gradually varying characteristics is highly desirable.

Since iterative methods also depend on the relationship between the right-hand side and the coefficient matrix, it is important to have realistic right-hand sides as well, that is, generated by the application rather than obtained by multiplying a synthetic solution vector.

2.2 Eigenvalue Calculations

Eigenvalue methods, like the iterative methods to which they are often closely related, depend strongly on the spectrum of a matrix. For a test of practical relevance, again realistic test matrices are needed. A finite element discretization gives rise to both a stiffness matrix and a mass matrix. While linear system solvers typically concern themselves with only one matrix, which is often a composite of the stiffness and mass matrices, for eigensolvers the presence of these two matrices leads to generalized eigenvalue problems which are harder to solve than the single-matrix eigenvalue problem. Therefore, a test collection which includes mass/stiffness matrix pairs is highly desirable.

2.3 Nonlinear systems

Nonlinear systems are also a source of solver research [24, 36, 25]. For research into the linear solvers that support many nonlinear solution algorithms, it would be advantageous to have a sequence of matrices and right-hand sides from the sequence of in the nonlinear solver. No existing test collection offers this, but it is easy for us to create.

Offering test problems for nonlinear solvers themselves is harder since the sequence of problems is a function of the results of each underlying linear solver step and the nonlinear solver algorithm itself. Static test collections are unable to handle this scenario, but our approach of using generator codes can accommodate it.

Rather than have a generator code be a subroutine, callable from the user's test code (section 9.2), we now need for the user's algorithm to be callable from the generator or for the generator and the user's test code to be mutually callable. We provide an facility for this usage scenario in our FEM generator (Section 6.3).

2.4 Performance of sparse operations

Optimization of sparse operations such as the matrix-vector product is another area where test matrices are needed [5, 39, 38]. Since the on-processor and inter-processor parts of the algorithm can be optimized separately, with the inter-processor part being a graph problem (see the next section), research has focused on matrices that fit a single processor. However, even in this case it is desirable to generate related matrices of different sizes to show how processor characteristics interact with matrix properties in the algorithm. Research in the papers cited above was clearly limited by insufficient richness of the available test collections.

2.5 Graph algorithms

As already indicated in Section 2.1 above, graph algorithms play an important role in sparse matrix research, and this motivates the need for test problems with non-trivial graphs. Another area where graph algorithms play an important role is in partitioning for distributed-memory parallelism. The goal here is to take a problem that can be described by a connectivity graph, and partition it over parallel processors in such a way that the work load is evenly balanced while traffic between processors, measured in number or weight of edge cuts, is minimized [14, 15, 26]. Of particular interest is dynamic load balancing, where an initial load distribution has to be updated to reflect changes in the problem that is being partitioned.

Graph algorithm research in this sense is particularly poorly served by the existing test collections of sparse matrices. Since any real test would address a large number of processors, large problems are needed—certainly larger than are currently available. Secondly, no test collection contains sequences of problems that come from adaptive mesh refinement or other processes that would necessitate dynamic load balancing.

2.6 Distributed and grid computing

Grid research and research in other loosely couple distributed system (see for instance [12]) requires large test problems that span a considerable number of processors. Researchers here are faced with a dilemma: most available test matrices are too small for large-scale parallel tests, while synthetically generated matrices are typically an oversimplification of actual problems.

2.7 Other

The need for varied test matrices arises in several other research areas. We mention two that are concerned with the modeling of numerical behaviour through statistical or machine learning techniques:

- The Online Condition Number Query System and the Intelligent Preconditioner Recommendation System of Zhang and Xu [40, 17]
- The Self-Adapting Large-scale Solver Architecture of the PI of this project [4, 10]

In both cases, the modeling techniques require test data that has both variety and that covers the parameter space locally. Current data collections fall short of this ideal.

3 The ideal test collection

We briefly summarize the various capabilities that an ideal test collection would have.

Realism and variety The test problems should come from real applications—preferably from a variety of different fields.

Graded problem features Isolated data points in a numerical test are rarely useful: one is always interested in seeing tendencies noting both asymptotic and 'small case' behaviour. Therefore, a test collection should provide:

- Families of test problems, corresponding to different physical problems, where
- Each family consists of problems that are related by the gradual change of relevant properties.
- Changes in problem parameters should be orthogonal.
- Preferably, numerical features of the problems should be given, so that solver behaviour can be plotted against them.

More than just a matrix Instead of only test matrices, for many problems it is interesting to have:

- Mass matrix and stiffness matrix for generalized eigenvalue problems.
- Matrix and realistic right-hand side for linear solvers. The right-hand side should come from the finite element discretization.
- Unassembled matrices for certain types of preconditioners.
- Test problems together with domain information such as coordinates or a finite element connectivity graph.

Discretization information For test problems from finite element and other PDE simulation codes, it is useful to have

- Coordinates of the nodes (such information is used in packages such as Dscpack [27]).
- The connectivity graph of the nodes. For single-variable physical problems this is the same as the matrix graph, but for multiple physical variables per node this offers essentially more information which can be exploited by solvers.

In the next section we show how existing test collections satisfy few of these demands, and in Section 5 we show how our setup addresses them.

4 Existing test collections

We will now briefly discuss existing collections of test matrices.

4.1 Harwell-Boeing

The Harwell-Boeing sparse matrices [9] (and the later Rutherford-Boeing matrices) form probably the oldest and best known collection of test problems. The collection has assembled and unassembled matrices as well as problems which include right-hand sides and known solutions. However, the collection is close to 20 years old, and the matrices are tiny by contemporary notions.

We note that the Harwell-Boeing file format used to store this collection [8] has some idiosyncrasies that make it hard to interpret in languages other than Fortran. Notably, its file header contains a Fortran format string. Furthermore, it is clearly designed to be space-economical in a way that has long ceased to be of importance. For this reason, the Matrix Market format (section 4.2) seems to have supplanted it as the *de facto* standard for a matrix exchange format.

4.2 Matrix Market

The Matrix Market collection of matrices includes the Harwell-Boeing set (in the Matrix Market format) as well as some more up-to-date problems. However, the matrices are generally of a quite modest size—only two approach a hundred thousand unknowns. These matrices have a size of approximately 20MB, which is of the order of the size of the largest level 2 caches in some current processors (9MB on an iTanium2, and 30MB in DRAM on a Power5).

Matrix Market format The Matrix Market file format is easily parsed, containing a few keywords in the file header, plus counts of the number of matrix rows, columns, and nonzeros. The file header also has space for accommodating arbitrary amounts of comment data about the matrix object.

Matrix Market generator codes Matrix Market has a set of codes that can generate arbitrarily sized matrices. However, these codes are quite simple in nature. Some generate the sort of ‘counter-example’ matrices that are found in the mathematical literature (Hilbert matrix, counterexamples to the Lapack condition number estimator, et cetera), while others generate somewhat physically meaningful problems. However, the latter codes are distinct simplifications of the sort of problems that real-world application codes generate.

Matrix Market Search tool The Matrix Market web site has a search tool for matrices. This can mostly be used to search for matrices based on nonzero structure (matrix size and number of nonzeros) or symmetry type (Hermitian, symmetric, etc.). While the individual matrix descriptions list some of matrix properties like the Frobenius norm and an estimated condition number, the search tool can not search based on these.

4.3 University of Florida Collection of Sparse Matrices

The University of Florida sparse matrix collection [6] overlaps with Matrix Market to an extent, but it seems to be more up to date and contains some larger matrices. However, by standards of parallel computing these are still of a very modest size.

The web site for this collection offers the possibility to browse matrices sorted by various properties, but as in the Matrix Market case, it is not possible to search on numerical quantities.

Part III

Infrastructure

5 Setup of a dynamic test collection

Our test collection is based on donated and found research and production codes; henceforth to be referred to as 'generator codes'. We use these codes in two ways: to build a static collection of searchable matrices (section 5.1), and to be run on demand to produce matrices from user-specified parameters (see Section 5.2). The details of the delivery mechanism are discussed in section 9.

5.1 Statically stored collection

In Section 4 we described how existing matrix test collections are overwhelmingly static sets of stored matrices for anyone to download. While this approach suffers from limiting the amount of variety available, it does have the important advantage that one can compute characteristics of the matrices, and offer to the user a search facility that allows specification of values or ranges for these characteristics.

We offer permanent storage of a collection of small to medium test problems, appropriate for 1–10 processors. (Depending on available storage space we may store a few large problems, suitable for hundreds of processors.) This static collection, far larger than existing collections, is searchable through a web interface by its numerical properties. The searchable properties are the matrix characteristics that are computed by the AnaMod package [28, 10] (see Section 5.3).

For the stored matrices, we use the Matrix Market file format as plain storage, and the PETSc binary format as a full-precision, compact, storage.

5.2 Matrices on demand

While we store a considerable number of matrices, corresponding to a wide range of input parameters of the generator codes, the user may still desire other parameter choices, or matrices that are larger than any that are stored. For this we have a mechanism for generating matrices on demand.

Generated matrices from user specifications can be requested, as for the stored matrices, through a web interface or through an interactive mechanism on the TACC and other TeraGrid computational resources (see Section9.2). While the database interface for searching through the matrix properties is the same for all matrices, each generator code has its own set of input parameters which need to be specified by the user.

We provide the option of having AnaMod properties (Section 5.3) computed for the matrices we generate. Since this can be fairly expensive, it increases the amount of time a user request takes. This optional extra time demand does not exist for the stored test problems since we pre-compute the properties.

5.2.1 TeraGrid

As a Science Gateway project within the TeraGrid, this project is able to leverage the investment TACC has made as an RP and GIG participant. We use the fact that TACC is a TeraGrid RP in two ways. First, TACC's high-bandwidth connections are used for copying large datasets to other TeraGrid sites and other sites connected through the National Lambda Rail. Consequently, we can let a user generate test data at TACC, but instead of processing it at TACC (section 9.2) have the data copied at high speed to the remote site. For this mode we may extend the web interface to use, for instance, Firefox plug-in for GridFTP.

Secondly, where licensing allows it, we install our generator codes and utilities in the TeraGrid Community Software Area (CSA). This space is available on every TeraGrid resource to provide a backed-up, consistent location for executables, tools, and libraries that are used by a community of users. All TeraGrid researchers can apply for CSA space by request. This allows as many users as possible to generate large-scale parallel matrices for testing across the different TG sites, using a consistent, easy-to-find location that is encoded as part of the TG user login environment.

5.3 Matrix features

Test matrices are valuable for algorithm research only if they bring out salient features of the algorithms which depends on them possessing certain numerical or structural properties. Consequently, being able to measure the properties of test matrices or to generate matrices with certain properties is crucial. We discuss this issue briefly.

In solver research, often a qualitative relation between certain mathematical properties and practical solver behaviour is known or expected, but quantitative aspects can only be determined by experiment. Thus, it is valuable for a researcher to be able to find problems with pre-specified properties, or to know such properties for matrices that are retrieved from a test collection.

We use the AnaMod package [10] for computing matrix features. This package, developed by the PI of this current project, can compute a large number of characteristics:

structural Features describing the nonzero structure of a matrix, such as bandwidth, sparsity, number of nonzeros per row.

norm-like Various matrix norms and norms of the symmetric and anti-symmetric part of a matrix.

spectral Largest and smallest eigenvalues, by magnitude, real and imaginary part, both by magnitude and absolute size.

normality The departure from normality.

Features computed through AnaMod are stored in a database. The user si then able to formulate queries in a web interface to search for matrices based on these properties. For matrices generated on demand, generation of AnaMod properties can be requested.

The current version of AnaMod is geared towards use in production codes, so it will compute certain features only approximately. Prime examples here are the spectral and normality categories of features, which are only estimated through a Lanczos process and other bounds from the literature. For the current project we will extend AnaMod to include exact computation of such features, using LAPACK and other dense linear algebra software.

Part IV

Generator codes

Above, we have described how generator codes are at the heart of this project. In this section we describe the ways we will pursue acquiring or constructing them. As described in Section 3, we try to go beyond generating only matrices, delivering, where appropriate, linear systems including right-hand sides, mass/stiffness matrix pairs, unassembled matrices, and nonlinear problems.

In general, our mission is to seek out those areas where test problem generation is not trivial, and where problems can be characterized by some set of input parameters. This rules out many research areas for instance in life sciences, where test problems are either randomly generated, or based on real life datasets. These are better served by a test problem database, rather than the server we are proposing.

We now describe how we are building our collection of codes (section 6) and the work involved in porting and adapting codes (section 7).

6 Acquiring codes

We use some existing generators, adapt donated codes, and construct fairly general purpose problem generators ourselves.

6.1 Donated codes

We have the following codes already promised:

- MGF is a 3D, parallel, unstructured code for solving finite element problems on user-specified partial differential equation systems [3]. It uses a LaTeX-like input language to allow the user to describe problems to be solved by giving the linearized weak form of the equations. Using this mechanism, steady and unsteady solutions of both linear and non-linear systems may be computed. It has been applied to non-linear Darcy Flow with solute transport for groundwater contaminant transport, incompressible Navier-Stokes with heat transfer for the study of natural convection and pipe-flow problems of generalized-Newtonian fluids, and a wide range of canonical problems including the Poisson problem and time-dependent convection-diffusion.
- The code 3Dhp90 [7] is a fully-automatic hp-adaptive finite element code for linear elliptic PDE systems and time-harmonic Maxwell's equations in three dimensions. Starting from an initial, regular hexahedral mesh, the code automatically generates a sequence of coarse and fine hp-meshes such that the discretization error converges exponentially with respect to the problem size. It has been applied to solve industrial problems related to heat conduction, elastic deformation, the vibrations of a visco-elastic solid, acoustic and electromagnetic scattering and the simulation of so-called "logging while drilling" tools.
  This is code is actively being developed at The University of Texas at Austin.

The local refinement used in this code is of particular interest to research in dynamic load balancing; section 2.5.

- We have a promise of the Department of Mechanical and Aerospace Engineering of the University at Buffalo, Buffalo, NY to make available two codes that use two complex approximation schemes for solving partial differential equations. The first is a code that uses adaptive *hp* finite elements and the second is a mesh-free method using a least square principle as opposed to a classical Galerkin. Both of these generate systems of equations that are irregularly sparse and poorly conditioned and hence good candidates for stress testing solution methodologies.
- Subject to agreement from DOE and the Commerce Dept, we will get access to the source of the PFLOTRAN groundwater code from Argonne National Lab. PFLOTRAN consists of two separate modules PFLOW and PTRAN that can be run either in stand-alone or coupled modes. PFLOW solves multiphase flow equations and PTRAN solves multicomponent reactive transport equations. In coupled mode, flow velocities, saturation, pressure and temperature fields computed from PFLOW are fed into PTRAN. For transient problems, sequential coupling of PFLOW and PTRAN enables changes in porosity and permeability due to chemical reactions to alter the flow field. PFLOW uses an efficient and modular mechanism to handle variable switching during phase transitions, allowing great flexibility in choosing the set of primary variables, addition of new equations of state (EoS), mixing rules, etc. The Jacobian matrix is evaluated numerically using a finite difference method. This approach gives PFLOTRAN great flexibility allowing easy addition of new phases. Chemical reactions are read from an extensive thermodynamic database in terms of a user-defined set of primary species read from the input.

We will also pursue contacts with the TOPS Scidac project [33] with which we are affiliated. In particular, we will try to acquire codes for nonlinear problems.

The above codes reflect our interests and background in partial differential equation. This is only the initial orientation of the generator codes. We will later explore other sources of linear algebra problems.

6.2 Existing generators

There are a number of matrix generators in existence that we will install and adapt for our purposes.

- The test problem generator of Howard Elman's Incompressible Flow and Iterative Solver Software [11]
- Pete Steward's Eigentest [29]
- A simulator code being developed by Matt Knepley of the Petsc team at Argonne National Lab.

6.3 Our own FEM generator

We developing our own finite element problem generator based on:

- Some common, and easily parameterizable, physical domains such as the L-shaped domain, circles with wedges removed, regions external to cylinders and spheres. We will also look into acquiring real-life geometries, such as the NACA profiles collection [21].
- Public domain mesh generators such as TetGen [31] and Triangle [34]
- Common partial differential equations such as Poisson, heat conduction, incompressible Navier-Stokes, elasticity, etc.
- Public domain finite element software such as LibMesh [18].

The user parameters here would be

- Domain parameters, such as the angle of the wedge removed from the circular domain, cylinder diameter, aspect ratios, etc.
- PDE parameters, for instance describing convection or material coefficients.
- Discretization parameters such as the mesh size, and the type and degree of the finite elements used.

This code will independently produce the mass matrices and stiffness matrices of the test problems for use with eigensolver testing, and the fully-combined form for the solution of the unsteady problems. Since we parameterize domain geometries, matrices with clustered eigenvalues, which are problematic in both iterative methods and eigenvalue solvers, can easily be produced.

In addition to producing mass and stiffness matrices, our code may have the facility to produce the unassembled matrix, as well as coordinate information of the nodes, and the mesh graph. These latter possibilities are pertinent in the case of multi-physics codes.

6.4 Open Channel

We may acquire codes from the Open Channel Foundation [23] (and possible other sources) to run them as test generators. Open Channel Software is a non-profit organization devoted to publishing university codes, while allowing for the commercialization of the most promising programs. Current codes cover such diverse fields as antenna design, bioinformatics, digital signal processing, and optics. Most of the Open Channel codes are available under a source license.

7 Adapting codes

Each generator code requires a modest amount of work to adapt it to our purposes. The following aspects are tackled:

running modes All codes need to be able to run in two modes. For the generation of the static collection and for use through the browser interface the codes need to run standalone. For codes that are delivered to us as a library this means writing a small main program.
For the in-situ delivery of test problems (Section 9.2) it is necessary that the codes be able to as a subroutine linked to the user's solver tester. This means turning the main program of each generator code into a subroutine, removing MPI initialization, et cetera.

input It is necessary to inventory the inputs of a code including both parameters and larger structures such as meshes or geometry descriptions. We use scripts so that all codes can be activated with the appropriate inputs from the browser interface. Providing the inputs when the code is used for in-situ delivery is a bit harder, and unfortunately can not be completely standardized. Each code has its own interface, for which documentation is provided.

output Typically, codes have a fairly defined location where a solver is applied to a generated problem. At that place we insert code for extracting the matrices. In some cases this may require assembling the problem from the code data structure. Additionally, since we have decided to standardize on the Matrix Market file format for extracted matrices, we have made an extension to this format to handle large distributed matrices. For in-situ delivery of test problems we convert the generated data to Petsc format.

Part V

User portal and delivery mechanisms

8 Delivery formats

We deliver test problems in a small number of formats. Here we give our justification for the choices we made.

Harwell-Boeing format We have decided to abandon this format [9]. While it has a veritable history, and is somewhat flexible, we feel that it it more geared towards economy of space, a consideration that is not as pressing as it was in the past. This makes it hard to parse. The extension to parallel problems is not trivial, and probably not worth the effort.

Matrix-market format (Section 4.2) We feel that this is a suitable *lingua franca* of matrix formats, a *least common denominator* in a way. Extensions to this format, for parallelism or the multiple matrix case, are easy to make and easy to parse. The only problem with this format is that it is wasteful in space, which we address with our other preferred format, discussed next.

PETSc binary format Since a fair number of our generators will be based on PETSc, and since we expect a good portion of our users to be familiar with this package, we have decided to use the PETSc binary file format as a standard for space-efficient matrix storage.

9 Delivery mechanisms

As a Resource Provider (RP) in the NSF TeraGrid project, and a member of its Grid Integration Group (GIG), TACC is well-positioned to provide this new resource. The static part of the collection will be provided to the international community as an open TeraGrid Data Collection [30], and the whole project will be run as a TeraGrid Science Gateway [32].

The TeraGrid Data Collection projects provide a organized, searchable method of accessing permanent collections of scientific data to the general public. TeraGrid Science Gateways are designed to bring the power of national resources to communities with common scientific goals.

In order to bring this test matrix collection to the widest possible community of researchers, we will leverage the TG Science Gateway and Data Collection projects to implement the following interfaces:

Web Portal Through the use of a web browser, users will be able to search our database of matrices or generate new ones with with specified parameters. These matrices will be of size appropriate to download through the browser.

Interactive and programmatic For people with allocations on TeraGrid resources, we will offer the possibility to generate matrices on demand for use in a batch job.

9.1 Web Portal

We will design a web portal that allows for the searching and downloading of stored matrices and for the on-demand generation and downloading of matrices from the generator codes through the web browser.

Our web browser interface will first of all give direct access to the collection of stored matrices. This interface will offer the possibility of selecting test problems by name or by generating application. In the latter case, the problem will be able to be chosen by the available values of the problem parameters. For each stored matrix, there will be the possibility of delivering an XML or text file of computed matrix features (see Section 5.3).

The browser interface will also offer a search possibility. The pre-computed characteristics of the problems in the stored collection will be stored in a database, and the browser will be able be used to form queries into the database. Problems satisfying the query will then be available for download individually or in bulk.

Finally, the browser interface will also give access to the generator codes directly. The user will specify a generator and parameter settings for it and have the test problem generated for subsequent download. There will also be the possibility of generating the problem characteristics, although because of the expense of this we may put certain limits on this capability.

We will design a database schema that will accommodate the storage of the metadata for the both static collection. This database will be used to drive the web site and to support search and display of available matrices. Additionally, the database infrastructure will be used to provide personalized account tracking, allowing individual users to keep track of their dynamically generated matrices and the parameters they used to generate them. We will use a standard open source relational database management system (RDBMS) such as MySQL or PostgreSQL.

9.2 Interactive and Programmatic

It is quite conceivable that users may want to generate test problems on a scale that precludes download through a browser interface. For users with access to the TeraGrid resources, we offer the possibility of retrieving or generating test problems from inside a batch job. We will offer two interfaces.

The most general way of delivering test problems to a user code, is to generate them in a standard file format and place the file in a place accessible to the user on the parallel file system available at each supported TG site. We will use the Matrix Market file format (Section 4.2), which is easily parsed by user codes. Generating or retrieving matrices this way will be done through a command line interface that can be used as part of a batch script. The Matrix Market file format is only for sequential problems, but we will define a parallel extension for matrices partitioned into blocks of rows.

We will also offer a way to have test problems directly available inside the user's code. This has two advantages. First, the relatively slow file system operations can be avoided. Second, the user will not need to parse the file storing the problem. The seemingly obvious choice of yielding them as compressed row storage (CRS) arrays puts a large burden on the user, since, in the absence of standard for a parallel CRS storage format, it requires the user to do a considerable amount of setup for parallel operations. For this reason we will limit ourselves initially to making matrices available inside user codes as stored in Petsc [1, 2] format. In the future we may also support the Trilinos format as well [16, 35].

Note that, for this mode of making problems available, the generator codes have to be adapted so that they can be called as a subroutine in the users' test codes.

We may put limits on the availability of certain problem parameters, in particular the size of the generated problem. In practice, such limits will already be imposed by the number of processors allocated to a user's job.

10 Authentication and history

We distinguish a number of service levels, which need different kinds of authentication.

Download Our stored collection of matrices will be accessible to anyone without any sort of authentication or registration. This will make the stored part of the data accessible to researchers anywhere in the world.

Dynamically generated Data that is generated on demand may take appreciable time to generate, especially if computed features are requested, and the download may also take long. For this reason we will offer emailed download links. This requires users to register with a working email address. For these users we also offer a history facility and possible some concept of 'state'.

Web services Our more sophisticated services require authentication through the Teragrid Portal or the TACC User Portal.

11 Reproducible research through global histories

In order to promote 'reproducible research', we generate for each run of a generator code a unique 'ticket number'. This number can be included in research report, making it possible for other researcher to reproduce and check the stated results.

12 Other portal functions

The web portal to the server will have user forums. (User accessibility will be fairly liberal, only controlled through registration with a working email address.) We see at least two benefits for having forums:

- They offer an easy way to offer user support. In particular, users can support each other.

- We encourage users to report the performance of their methods on test data. Since test data is identifiable through its ticket number (see section 11), we hope that this will spur more comparative research between methods than is currently done.

Part VI

Project status

This project has been funded by NSF per spring 2008. A server has been acquired and installed and the first number of generator codes are being ported. We hope to hae a first version of the server available to the public by the second quarter of 2009.

As we are still developing this project, feedback on any part of it is very much welcome.

References

[1] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhauser Press, 1997.

[2] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. PETSc home page. http://www.mcs.anl.gov/petsc, 1999.

[3] William L. Barth. *Simulation of Non-Newtonian Fluids on Workstation Clusters*. PhD thesis, The University of Texas at Austin, May 2004.

[4] S. Bhowmick, V. Eijkhout, Y. Freund, E. Fuentes, and D. Keyes. Application of machine learning to the selection of sparse linear solvers. *Int. J. High Perf. Comput. Appl.*, 2006. submitted.

[5] Alfredo Buttari, Victor Eijkhout, Julien Langou, and Salvatore Filippone. Performance optimization and modeling of blocked sparse kernels. Technical Report ICL-UT-04-05, ICL, Department of Computer Science, University of Tennessee, 2004. to appear in IJHPCA.

[6] T. Davis. University of florida sparse matrix collection. `http://www.cise.ufl.edu/research/sparse/matrices`, `ftp://ftp.cise.ufl.edu/pub/faculty/davis/matrices`, described in NA Digest, vol. 92, no. 42, October 16, 1994, NA Digest, vol. 96, no. 28, July 23, 1996, and NA Digest, vol. 97, no. 23, June 7, 1997.

[7] L. Demkowicz, D. Pardo, and W. Rachowicz. 3D *hp*-adaptive finite element package (3Dhp90) version 2.0. the ultimate (?) data structure for three-dimensional,

anisotropic *hp* refinements. Technical Report 02-24, TICAM, The University of Texas at Austin, 2002.

[8] I. S. Duff, R. G. Grimes, and J. G. Lewis. Users' guide for the Harwell-Boeing sparse matrix collection (release I). Technical Report RAL 92-086, Rutherford Appleton Laboratory, 1992.

[9] Iain Duff, Roger Grimes, and John Lewis. Sparse matrix test problems. *ACM Trans. Math. Software*, 15:1–14, March 1989.

[10] Victor Eijkhout and Erika Fuentes. A proposed standard for numerical metadata. *ACM Trans. Math. Software*, 2006. submitted.

[11] Howard Elman. Incompressible flow and iterative solver software. `http://www.cs.umd.edu/~elman/ifiss.html`.

[12] Nahid Emada, S.-A. Shahzadeh-Fazelia, and Jack Dongarra. An asynchronous algorithm on the netsolve global computing system. *Future Gen. Comp. Sys.*, 22:279–290, February 2006.

[13] N. I. M. Gould, Y. Hu, and J. A. Scott. Complete results from a numerical evaluation of sparse direct solvers for the solution of large, sparse, symmetric linear systems of equations. Technical Report 2005-1, CCLRC, Numerical Algorithms Group.

[14] Bruce Hendrickson and Robert Leland. A multilevel algorithm for partitioning graphs. In *Proc. Supercomputing '95*. ACM.

[15] Bruce Hendrickson and Robert Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM J. Sci. Comput.*, 16:452–469, 1995.

[16] Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P. Pawlowski, Eric T. Phipps, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. An overview of the trilinos project. *ACM Trans. Math. Softw.*, 31(3):397–423, 2005.

[17] Laboratory for High Performance Scientific Computing & Computer Simulation. Online condition number query system. `http://www.cs.uky.edu/~hipscns/HiPSCCS_Projects/OCNQS/done.htm`.

[18] `http://libmesh.sourceforge.net/`.

[19] Richard J. Lipton, Donald J. Rose, and Robert Endre Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.*, 16:346–358, 1979.

[20] Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36:177–189, 1979.

[21] `http://www.pdas.com/avd.htm`.

[22] National Institute of Standards and Technology. Matrix market bibliography.

[23] `http://www.openchannelfoundation.org/`.

[24] R. P. Pawlowski, J. N. Shadid, J. P. Simonis, and H.F Walker. Globalization techniques for Newton-Krylov methods and applications to the fully-coupled solution of

the Navier-Stokes equations. *SIAM Review*. to appear; also report SAND2004-1777, Sandia National Laboratories, May, 2004.

[25] M. Pernice and H.F. Walker. NITSOL: a Newton iterative solver for nonlinear systems. *SIAM J. Sci. Comput.*, 19:302–318, 1998.

[26] Alex Pothen, Horst D. Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis and Applications*, 11(3):430–452, July 1990.

[27] P. Raghavan. DSCPack: A domain-separator cholesky package. Technical report, University of Tennessee, Knoxville, TN, 1999.

[28] Salsa Project. SourceForge: Self-Adapting Large-scale Solver Architecture. `http://sourceforge.net/projects/salsa/`.

[29] G.W. Stewart. `ftp://ftp.cs.umd.edu/pub/stewart/reports/eigentest.tar`.

[30] `http://www.teragrid.org/userinfo/data/collections.php`.

[31] `http://tetgen.berlios.de/`.

[32] The TeraGrid Project. TeraGrid Science Gateways. `http://www.teragrid.org/programs/sci_gateways/`.

[33] `http://www.tops-scidac.org/`.

[34] `http://www.cs.cmu.edu/~quake/triangle.html`.

[35] `http://software.sandia.gov/trilinos/index.html`.

[36] R.S. Tuminaro, H.F. Walker, and J.N. Shadid. On backtracking failure in Newton-GMRES methods with a demonstration for the Navier-Stokes equations. *J. Comp. Physics*, 180:549–558, 2002.

[37] A. van der Sluis and H.A. van der Vorst. The rate of convergence of conjugate gradients. *Numer. Math.*, 48:543–560, 1986.

[38] R. Vuduc, J. Demmel, and K. Yelikk. Oski: A library of automatically tuned sparse matrix kernels. In *(Proceedings of SciDAC 2005, Journal of Physics: Conference Series, to appear.*, 2005.

[39] Richard W. Vuduc. *Automatic Performance Tuning of Sparse Matrix Kernels*. PhD thesis, University of California Berkeley, 2003.

[40] ShuTing Xu, Eun-Joo Lee, and Jun Zhang. An interim analysis report on preconditioners and matrices. Technical Report 388-03, University of Kentucky, Lexington; Department of Computer Science, 2003.