MITMatlab is a small software research project to bring the benefits of interactivity to parallel supercomputing particularly for large problems. The user types the traditional friendly MATLAB commands on a parallel machine and obtains the speed and memory benefits of parallel computation. Since users have their own MATLAB codes and MATLAB itself is built from scripts, the software reuse benefits of this approach can not be overstated.

We believe that MITMatlab is the only one of the many competing "parallel MATLAB" projects that obtains its capability through the combination of 1) object oriented MATLAB scripts and 2) a parallel server to distributed processors. For hardware, any parallel set up with MPI is a viable platform for MITMatlab. MITMatlab requires one copy of traditional MATLAB (MultiMatlab [6] requires several, FALCOLN [2] requires none). Matrices are constructed in parallel, not on the front end (the front end strategy is taken by NetSolve [3] and RCS [1]). Therefore we are not limited by the memory size of one machine nor the cost of transferring a big matrix.

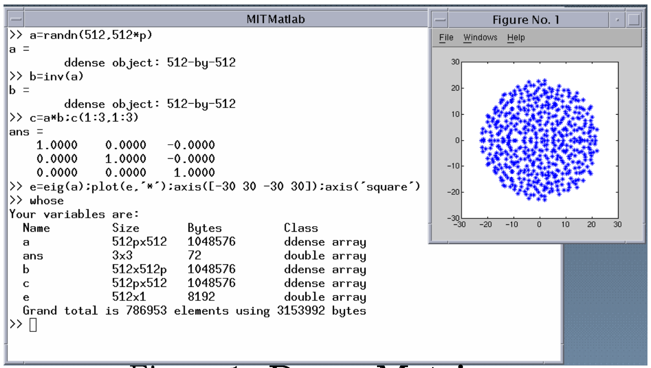

Figure 1: **Dense Matrices**

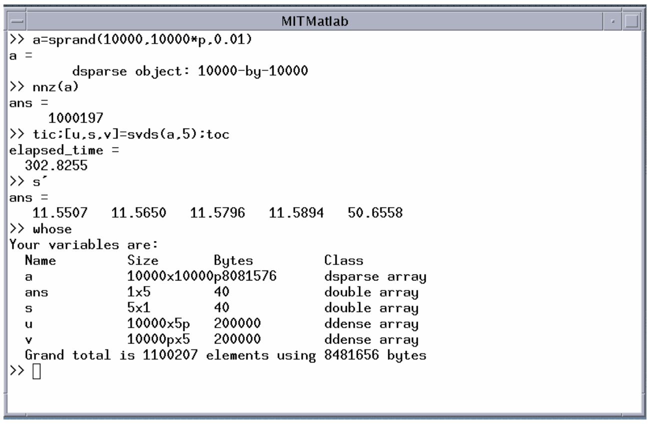

Figure 2: **Sparse Matrices**

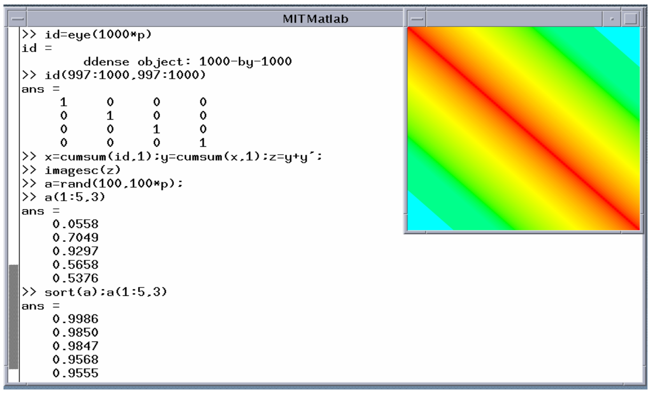

Figure 3: **Miscellaneous Functions**

The key to be able to reuse traditional MATLAB software is our innovative "p" notation which uses Matlab 5's object oriented features to call a home grown server which sends commands to the parallel processors. A user need only type `a=ones(1000*p,1000)` or `b=rand(1000,1000*p)` to obtain two 1000 $\times$ 1000 matrices distributed in parallel. The reader can guess the use of `p`: it indicates that `a` is distributed by rows and `b` by columns. For MATLAB, the `p` creates `a` and `b` as pointers to special distributed dense and sparse types known as `ddense` and `dsparse`. Therefore any call to such commands as `eig`, `svd`, `inv`, and `*` `(mtimes)` with `a` and/or `b` as arguments will be recognized as a call to the server and not as a traditional MATLAB command.

What is `p`? Its purpose, as explained before, is to indicate a parallel dimension propagating the parallelism through the dimension variables rather than the arrays themselves. We found further that additional clever choices in `p`'s definition and influence were crucial towards our goal of software reuse.

The reader might gain some insight by considering a few experimental inputs and outputs:

Input:	`p`	`p*10`	`p+10`	`[p 4 5*p]`
Output:	`1p`	`10p`	`11p`	`1p 4 5p`

Observe that the `p` behaves just like the integer "1"in many ways. However when called with such constructors as `rand, randn, zeros, ones, eye, sprand,` and `sprandn` a parallel array is constructed. A little thought and the reader may be convinced that many MATLAB functions may now work in parallel simply by calling the functions with a `p` argument. To test this concept, we found that most of Higham's [4] matrix toolbox worked in parallel without any modification. Of course this did not happen by accident. Some very careful choices of overloading indexing were needed.

Underneath the hood, whenever a function is applied to an array defined by or derived from a `p` object, we call a parallel problems server [5] designed to run on any Unix-like platform supporting MPI. The server communicates with clients at run-time using a simple request-response protocol and is directly extensible via compiled libraries called packages.

A number of research and class projects have been implemented with MITMatlab and the applications that we can support grow every day.

References

[1] P. Arbenz, W. Gander, and M. Oettli. The Remote Computation System. Technical Report 245, ETH Zurich, 1996.

[2] Falcon Group at the University of Illinois at Urbana-Champaign. The falcon project. http://www.csrd.uiuc.edu/falcon.

[3] Henri Casanova and Jack Dongarra. NetSolve: A Network Server for Solving Computational Science Problems. In *Proceedings of SuperComputing 1996*, 1996.

[4] Nicholas Higham. The Test Matrix Toolbox, 3.0. http://www.ma.man.ac.uk/~higham/testmat.html.

[5] Parry Husbands and Charles Isbell. The parallel problems server: A client-server model for interactive large scale scientific computing. 1998.

[6] Anne E. Trefethen, Vijay S. Menon, Chi-Chao Chang, Gregorz J. Czajkowski, Chris Myers, and Lloyd N. Trefethen. MultiMATLAB: MATLAB on Multiple Processors. http://www.cs.cornell.edu/Info/People/lnt/multimatlab.html, 1996.