Is Mathematical Software a Legitimate Research Area?

John R. Rice

Purdue University, jrr@cs.purdue.edu

Report Number:
80-362
IS MATHEMATICAL SOFTWARE A LEGITIMATE RESEARCH AREA?

John R. Rice
Mathematical Sciences
Purdue University

CSD-TR 362

ABSTRACT

This essay presents criteria for legitimacy of a research area and applies them to mathematical software. It is concluded that mathematical software is emerging as a legitimate research area. It is also concluded that any area (and mathematical software in particular) does not have to be legitimately established for one to do significant research in the area.

IS MATHEMATICAL SOFTWARE A LEGITIMATE RESEARCH AREA?

John R. Rice
Purdue University

Prologue: Recently the chairman of a Computer Science department wrote to solicit my views regarding the legitimacy of mathematical software development as a research area that can stand by itself. This question arose in that department's deliberations on promotions and tenure and they sensed "a clear split in the numerical analysis community on this question. The opposing view takes the position that mathematical software development is a sub-branch of the area of numerical analysis for which the software is being developed, and is not itself sufficient to establish research credentials for a numerical analyst." The following is my response to this solicitation.

The question posed is: Is mathematical software development a legitimate research area that can stand by itself. I approach this question from three viewpoints and consider several more specific, related questions:

1. How does any research area become legitimate?
2. Is software development a legitimate research area?
3. What is the relationship between numerical analysis and mathematical software?

What is the nature of software research? How does mathematical software research differ from other software research? How does mathematical software research relate to the general requirements for legitimacy?

I divide software research into two categories: internal and external. Internal research does not involve what the software is supposed to do. It is the software itself, the user's problems. This category includes programming languages, operating systems, compilers, etc. Research in these subcategories involves an interplay between internal software topics, the theory and facts from the application area, and theory and facts specific to the subcategory. For an illustration in the area of mathematical software, consider software for adaptive numerical integration. Good software must be portable, well documented, certified as correct, well structured, etc.; all these aspects are from internal software research. The underlying algorithms use certain numerical quadrature rules and error estimates of traditional numerical analysis; these are from the application area.

Criterion 3: There are accepted successful methodologies for many tasks.

I propose three criteria to test an area for legitimacy in scientific research:

Criterion 1: There are important and difficult scientific problems to be solved.

Criterion 2: There is a large and accepted body of facts and theories to apply.

After developing these questions, I answer the original question with a qualified yes; mathematical software development is emerging as a legitimate research area. Its patterns are still only partially formed and there are many false starts yet to be made. Even so, I cite first class scientific achievements in mathematical software that could never be accomplished within the framework of traditional numerical analysis. However, I also conclude that the legitimacy of mathematical software as a research area is irrelevant to the evaluation of particular research results.

I propose three criteria to test an area for legitimacy in scientific research:

Criterion 1: There are important and difficult scientific problems to be solved.

Criterion 2: There is a large and accepted body of facts and theories to apply.

Criterion 3: There are accepted successful methodologies for many tasks.

With these criteria we see that (a) Arithmetic does not qualify because it is not difficult (but number theory does qualify); (b) Politics does not qualify because there is no body of accepted facts and theories; (c) Betting on the horses does not qualify because there are no successful methodologies for winning; (d) Numerical analysis, computational complexity and programming language theory do qualify.

There are areas in Computer Science whose qualifications are widely doubted, e.g., artificial intelligence, business systems and software. I consider the last of these in more detail.

I divide software research into two categories: internal and external. Internal research does not involve what the software is supposed to do. It is the smallest of the two categories and includes specialties like program verification, software portability, software science, etc. The external research is "applications" oriented; the principal subcategories include operating systems, language processing, business systems and mathematical software. Research in these subcategories involves an interplay between internal software topics, the theory and facts from the application area, and theory and facts specific to the subcategory. For an illustration in the area of mathematical software, consider software for adaptive numerical integration. Good software must be portable, well documented, certified as correct, well structured, etc.; all these aspects are from internal software research. The underlying algorithms use certain numerical quadrature rules and error estimates of traditional numerical analysis; these are from the application area. The research also involves (a) comparative performance evaluations of adaptive numerical integration software including the definition and justification of performance criteria, (b) analysis of the trade-off between time, space and reliability among the data structures that can be selected for the algorithm (c) identification of the problem class for which the algorithm is effective; these items are specific to mathematical software.

I propose three criteria to test an area for legitimacy in scientific research:

Criterion 1: There are important and difficult scientific problems to be solved.

Criterion 2: There is a large and accepted body of facts and theories to apply.

Criterion 3: There are accepted successful methodologies for many tasks.

With these criteria we see that (a) Arithmetic does not qualify because it is not difficult (but number theory does qualify); (b) Politics does not qualify because there is no body of accepted facts and theories; (c) Betting on the horses does not qualify because there are no successful methodologies for winning; (d) Numerical analysis, computational complexity and programming language theory do qualify.

There are areas in Computer Science whose qualifications are widely doubted, e.g., artificial intelligence, business systems and software. I consider the last of these in more detail.

I divide software research into two categories: internal and external. Internal research does not involve what the software is supposed to do. It is the smallest of the two categories and includes specialties like program verification, software portability, software science, etc. The external research is "applications" oriented; the principal subcategories include operating systems, language processing, business systems and mathematical software. Research in these subcategories involves an interplay between internal software topics, the theory and facts from the application area, and theory and facts specific to the subcategory. For an illustration in the area of mathematical software, consider software for adaptive numerical integration. Good software must be portable, well documented, certified as correct, well structured, etc.; all these aspects are from internal software research. The underlying algorithms use certain numerical quadrature rules and error estimates of traditional numerical analysis; these are from the application area. The research also involves (a) comparative performance evaluations of adaptive numerical integration software including the definition and justification of performance criteria, (b) analysis of the trade-off between time, space and reliability among the data structures that can be selected for the algorithm (c) identification of the problem class for which the algorithm is effective; these items are specific to mathematical software.
All of the external software research areas involve a mixture of theory and practice. The research does not exclusively follow the mathematical pattern; there are engineering aspects (getting things to work well without a complete theory) and experimental science aspects (making observations, classifying phenomena and testing hypotheses). The latter aspects are analogous to much of the research in the biological sciences, astronomy and geology.

In principle, mathematical software research does not differ substantially from other external subcategories of software research. In practice it differs in two ways. First, there was a large body of theory and scientists established in the application area before software became important. Thus, it is easy to distinguish between the software and the application area here while in the other subcategories of software the software and applications theory are thoroughly mingled in one's mind and education. Second, mathematical software is older than the others, it is more mature, and some structure is emerging from the original chaos. In spite of this additional maturity (which is denied by some in other software areas), mathematical software is relatively less mature when compared to its associated application areas.

I next consider how well mathematical software development meets the qualifications to be a legitimate research area. There is no doubt that there are important scientific problems to be solved. Some question whether they are difficult problems is that they believe once the numerical analysis is done, it is straightforward to transform this into high quality mathematical software and, more to the point, there are no significant difficulties beyond those of the numerical analysis. This viewpoint is especially prevalent among those who have never produced any high quality software. The COGERS panel on numerical computation research identified the top twelve accomplishments in this field since 1945. One of these is the software for ordinary differential equations developed by Gear, Shampine and others. In contrast, the fundamental work of Dahlquist on the stability of numerical methods for ordinary differential equations did not make the top twelve list. This panel of experts believed that mathematical software research can be both important and difficult and can be research of the first rank. There is not a large and widely accepted body of facts and theory in mathematical software, but this is developing. A text book with a substantial discussion of mathematical software will appear in 1981 and I expect most texts on numerical computation in the late 1980's to be a balanced blend of numerical analysis and mathematical software. I note that there are several successful methodologies for mathematical software tasks, for example, library and systematized package technology, portability of numerical software in Fortran, polyalgorithms, performance evaluation of mathematical software, preprocessors for problem areas (statistics, PDE's, linear programming, etc.) and reliability in error estimation through redundancy.

Not all mathematical software research is great or even good; its average quality is probably lower than normal for several reasons: (1) the ground rules are not yet well developed and understood, (2) the pressure of real problems forces many projects to be attempted with inadequate understanding, ability and resources and (3) many experts from other fields believe they can dash off significant software without giving it any thought or learning how it is done.

The preceding arguments are enough for me to answer with a qualified yes to the original question posed. Yet the question posed is not the real question to be answered. The real question is: Can one do significant research in mathematical software which is not in fact numerical analysis research? To answer this question, I explore the relationship between these two specialties. First, it is clear that there is no precise statement of the content of either specialty nor is there a person or group who can say "what we do is all of numerical analysis and nothing more". Second, most workers in the general area recognize the flavor of numerical analysis. It has the taste of mathematics; definitions, theorems and proofs. Its theory is a natural descendent of what existed 35 years ago when it was operated pretty much in a vacuum, very rarely was any real computation undertaken then. This flavor is still the principal one found in text books on numerical computation. On the other hand, mathematical software has the flavor of real computing; writing programs and analyzing efficiency in terms of computer or human resources used.

I see these two specialties as the theory and practice of numerical computation. There are many theoreticians who have no experience or understanding of practice and whose work has no impact on the practice. One can make important discoveries with little knowledge of the theory. Just as the American continent was discovered, so were ADI methods, adaptive quadrature, FFT (many times since the late 1800's) and the QR algorithm. All these discoveries have had important long term consequences.

However, one can do little more in mathematical software research than make chance discoveries if one does not have an understanding of and facility with the theory. The relationship here between theory and practice is similar to that in many other fields.

Theoreticians traditionally claim everything done by practitioners is just the routine elaboration of their theories. One can observe that much of the practice is the straightforward application of technical skill and not research. Theoreticians tend to forget that many things are done in practice because they are useful and not because they are novel or interesting or difficult, or research projects. However, there is also research in the practice and there is a simple test for the significance of a result: Have clever, knowledgeable and dedicated people tried to accomplish it and failed? Or, for more original work, is it a provocative result that many such people could not achieve? Have others
recognized the work as interesting and significant. Hundreds of programs were written for solving ordinary differential equations before the current software appeared. The great superiority of the current, good software plus the multitude of previous inadequate attempts is why developing this software is regarded as one of the major research accomplishments in numerical computation.

The above test to evaluate research results completely avoids the issue raised originally. I believe there are many instances of significant research in areas that are not only not legitimate (as defined above), but areas that do not even exist (the result obtained could be the only one in the area). Thus, while I believe mathematical software is emerging as a legitimate area of research, this condition is not relevant for the evaluation of specific research results.
ON THE EFFECTIVENESS OF ITERATION FOR THE

GALERKIN METHOD EQUATIONS

John R. Rice
Mathematical Sciences
Purdue University

CSD-TR 362
March 25, 1981

This paper is to be presented at the IMACS conference:
Computer Methods for Partial Differential Equations, June
30 - July 2, 1981. It will appear in the proceedings:
Advances in Computer Methods for Partial Differential
Equations, IV, (.R. Vishnevetsky, ed.), IMACS, New Brunswick,
N.J., 1981.

Post Script: It has been discovered that the Fortran compiler
on the VAX was changed a few days before the data in Table 5
were measured. Considerable experimentation suggests that the
execution times in Table 5 are from 75-80% of the times using
the previous compiler (the one used for the other tables). To
compensate one should multiply the times in Table 5 by a factor
of 1.25 before comparisons are made with the data in Table 4.
ON THE EFFECTIVENESS OF ITERATION FOR THE
GALEKIN METHOD EQUATIONS

John R. Rice
Division of Mathematical Sciences
Purdue University

SUMMARY

This note reports on an experimental study of the effectiveness of matrix iteration methods when applied to systems of linear equations obtained from the Galerkin method using bicubic Hermite polynomials for two-dimensional elliptic partial differential equations. Two iteration methods are used: one is SOR and the other is Jacobi. These systems are the Galerkin systems for the three problems considered. They are compared to two direct methods: the recent LINPACK Gauss elimination routine for symmetric positive definite band matrices and a Yale Sparse Matrix Package routine. The entire study was done within the ELLPACK system for the performance evaluation of software for partial differential equations. The data shows conclusively that iteration methods are eventually (as the accuracy desired increases) more efficient than direct methods and the expected value for the cross-over point between iteration and Gauss elimination is for a 7x7 to 9x9 mesh which corresponds to an accuracy of about 0.1%. The data also shows that the LINPACK Gauss elimination routine is more efficient than the sparse matrix routine for meshes larger than 7x7. The implications for pipeline, parallel and microprocessor array computers are discussed.

I. THE EXPERIMENT

Large linear systems of equations are generated by finite element methods, such as the Galerkin method, which are then solved by the Galerkin method. These systems have been solved traditionally by direct methods e.g. variants of Gauss elimination. The data shown conclusively that iteration methods are eventually (as the accuracy desired increases) more efficient than direct methods and the expected value for the cross-over point between iteration and Gauss elimination is for a 7x7 to 9x9 mesh which corresponds to an accuracy of about 0.1%. The data also shows that the LINPACK Gauss elimination routine is more efficient than the sparse matrix routine for meshes larger than 7x7. The implications for pipeline, parallel and microprocessor array computers are discussed.

The study is experimental of the following nature:

A set of 13 partial differential equations (PDE's) are chosen from the population of Rice et al's their numbers are:

| 1-1 | 1-1 | 2-1 | 3-1 | 4-1 | 5-1 | 6-1 | 7-1 | 10-2 | 10-3 | 21-3 | 31-1 | 44-1 | 44-2 | 14-3 |

For the convenience of the reader these are given in the appendix of this paper. The results are shown in Table 5.1 of the Appendix and are quite compatible.

II. THE PERFORMANCE DATA

The basic criterion of performance is the computer time required to solve the linear system. All problems were solved on a uniform, square mesh of size NX by NX and the linear system is of order $\frac{1}{2}NX^2$. As hoped, log(time) increases linearly with log(NX) for all the problems considered so the slope of time versus NX (on a log-log scale) is taken as the primary measure of performance. In all cases, LINPACK is faster for NX=3, so there is the question of where the cross-over points lie. These are the points where iteration and Gauss elimination are equal.

Less precise but perhaps more interesting is the relationship between accuracy achieved and computer time required. We present some data for the performance measure of time needed to achieve a certain accuracy for three levels of accuracy - 5%, 0.5% and 0.05%.

The experimental data is considered in two parts. First, there is data for LINPACK SPD BAND, ITPACK SOR, and ITPACK JACOBI GG from two computers: the CDC 6600 using the XNF compiler and the UNIX (Berkeley version) compiler. The results are given in Tables 5 and 4 of the Appendix and are quite compatible.

These programs are part of the ELLPACK system. A small study suggested that SOR and JACOBI GG are the most efficient of the five ITPACK programs in ELLPACK for these equations.

The ELLPACK system is itself part of a larger system for the performance evaluation of software for partial differential equations. The methodology used for this study has been presented earlier.

SPD BAND: A LINPACK program for symmetric positive definite band matrices

SOR: An ITPACK program for successive over-relaxation

JACOBI GG: An ITPACK program for the Jacobi method accelerated by a conjugate gradient technique

YALE SPARSE: A Yale Sparse Matrix Package program which makes an LU factorization from the envelope form

These programs are part of the ELLPACK system. A small study suggested that SOR and JACOBI GG are the most efficient of the five ITPACK programs in ELLPACK for these equations.

The ELLPACK system is itself part of a larger system for the performance evaluation of software for partial differential equations. The methodology used for this study has been presented earlier.

These programs are part of the ELLPACK system. A small study suggested that SOR and JACOBI GG are the most efficient of the five ITPACK programs in ELLPACK for these equations.

The ELLPACK system is itself part of a larger system for the performance evaluation of software for partial differential equations. The methodology used for this study has been presented earlier.

These programs are part of the ELLPACK system. A small study suggested that SOR and JACOBI GG are the most efficient of the five ITPACK programs in ELLPACK for these equations.

The ELLPACK system is itself part of a larger system for the performance evaluation of software for partial differential equations. The methodology used for this study has been presented earlier.

These programs are part of the ELLPACK system. A small study suggested that SOR and JACOBI GG are the most efficient of the five ITPACK programs in ELLPACK for these equations.

The ELLPACK system is itself part of a larger system for the performance evaluation of software for partial differential equations. The methodology used for this study has been presented earlier.
The application of parallel computers to solving linear systems has been studied in detail and it is well known that it is desirable to develop programs which achieve most of the potential speed-up of parallel computers. Without going into a detailed analysis, we note that iteration methods have an inherent speed-up advantage over direct methods for microprocessor array computers. Consider the linear system to be the simple band matrix with order $N$ and bandwidth $K$. For two-dimensional elliptic PDE $K$ is about $\sqrt{N}$, for three-dimensional ones $K$ is about $3/\sqrt{N}$.

Consider first the use of vector oriented computers (e.g., CDC STAR-100, TI-ASC or Cray-I) to solve this system directly. It is difficult, perhaps infeasible in practice, to effectively use vectors of length more than $K$ in a direct method of solution for a band matrix. Thus, it will be difficult to effectively use a vector computer with a thousand processors on most applications. Many applications do not even require $K$ to be 100, while electronic fabrication technology is making it quite feasible to build computers with 1000 processors.

On the other hand, iteration methods allow one to effectively use vectors of length $N$ rather than length $K$. This means that iteration methods potentially allow one to make full use of highly parallel machines except for relatively small problems. In the latter case the problem is quickly solved anyway.

Several groups are exploring the use of arrays of microcomputers for solving PDEs by finite element methods. The natural idea is to have one processor per element which generates the linear system; thus the microprocessor array in some sense models the physical context of the PDE. Once the linear system is generated, it is not so easy to convert the microprocessor array to an efficient machine for solving the linear system directly. However, iteration methods are naturally adaptable to such arrays, one has one processor per equation or, more likely, one processor per group of equations.

Our performance evaluation has been for sequential computations and the reasoning outlined above suggests that the relative performance of direct and iteration methods would remain unchanged for parallel computers with a low level of parallelism. However, for computers which are highly parallel (involving 100s or perhaps 1000s of processors), the iteration methods have an inherent advantage which appears to make them the method of choice. This conjecture must, of course, be tested by analysis and implementation of real problems on actual machines.

IV. CONCLUSIONS

We conclude that iteration methods are more powerful than direct methods on finite element problems. The data show that iteration methods gain in efficiency over the direct methods as the requested accuracy increases (mesh refinement becomes smaller). Thus there is a "cross-over point" where the two methods can meet efficiency. This conjecture must, of course, be tested by analysis and implementation of real problems on actual machines.
methods (as well as other finite element methods) offers an inherent advantage over direct methods when one is using highly parallel computers such as arrays of microprocessors.

REFERENCES


APPENDIX: THE PDEs AND THE DATA

The partial differential equation problems used for this study are listed below. The domain for each problem is the unit square \( 0 \leq x, y \leq 1 \) and the boundary conditions are all homogeneous i.e. \( \cdot(x,y) = 0 \) on the boundary. In each case the forcing term \( f(x,y) \) is determined to produce a particular true situation.

\[
\begin{align*}
1. & \quad -1 (e^{x^2})_{xx} + (e^{-x^2})_{yy} = u/(1-x+y) = f \\
2. & \quad \nabla^2 u = f \\
5. & \quad -4 u_{xx} + u_{yy} = f \\
5. & \quad -4 u_{xx} + u_{yy} = 10u = f \\
6. & \quad -1 u_{xx} + u_{yy} = -(100 \cdot \cos(5x) + \sin(2y))u = f \\
7. & \quad -1 u_{xx} + u_{yy} = 10u = f \\
10-2 & \quad u_{xx} + u_{yy} = f \\
10-3 & \quad u_{xx} + u_{yy} = f \\
20-3 & \quad (u_{xx})_{xx} + (u_{yy})_{xx} = 1 \text{ where } w = 100 \text{ for } 0 \leq x, y < 1 \\
& \quad = 1 \text{ otherwise} \\
41-1 & \quad u_{xx} + u_{yy} = 10u = f \\
44-1 & \quad u_{xx} + u_{yy} = w, \quad w = -2.030625 e^{(r/(1+r/2))} \\
& \quad r(x,y) \text{ tabulated} \\
44-2 & \quad u_{xx} + u_{yy} = w, \quad w = -0.030625 e^{(r/(1+r/2))} \\
& \quad r(x,y) \text{ tabulated} \\
44-3 & \quad u_{xx} + u_{yy} = w, \quad w = -2.030625 e^{(r/(1+r/25))} \\
& \quad r(x,y) \text{ tabulated}
\end{align*}
\]
Table 3. The performance data for the CDC 6500. The SPD BAND solution times are independent of the PDE and are 0.06, 0.65, 3.82 seconds, respectively for 3x3, 5x5, 7x7 and 8x8 meshes.

<table>
<thead>
<tr>
<th>MESH</th>
<th>PDE 1-1</th>
<th>PDE 2-1</th>
<th>PDE 3-1</th>
<th>PDE 4-1</th>
<th>PDE 5-1</th>
<th>PDE 6-1</th>
<th>PDE 7-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACCURACY</td>
<td>SOR JACOBI</td>
<td>ACCURACY</td>
<td>SOR JACOBI</td>
<td>ACCURACY</td>
<td>SOR JACOBI</td>
<td>ACCURACY</td>
</tr>
<tr>
<td>3x3</td>
<td>9.5E-3</td>
<td>0.16</td>
<td>0.39</td>
<td>3.0E-3</td>
<td>0.15</td>
<td>0.19</td>
<td>2.5E-2</td>
</tr>
<tr>
<td>5x5</td>
<td>7.7E-3</td>
<td>1.00</td>
<td>1.45</td>
<td>2.5E-4</td>
<td>1.09</td>
<td>1.10</td>
<td>2.8E-4</td>
</tr>
<tr>
<td>7x7</td>
<td>1.0E-4</td>
<td>2.79</td>
<td>3.12</td>
<td>5.4E-5</td>
<td>2.56</td>
<td>2.74</td>
<td>4.8E-4</td>
</tr>
<tr>
<td>8x8</td>
<td>1.0E-4</td>
<td>3.18</td>
<td>3.62</td>
<td>3.3E-5</td>
<td>2.28</td>
<td>2.27</td>
<td>3.8E-4</td>
</tr>
</tbody>
</table>

Table 4. The performance data for the DEC VAX. The SPD BAND solution times (in seconds) are independent of the PDE and are listed first in a short table.

<table>
<thead>
<tr>
<th>NX</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>11</th>
<th>13</th>
<th>15</th>
<th>17</th>
<th>19</th>
<th>21</th>
<th>23</th>
<th>25</th>
<th>27</th>
<th>29</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>0.1</td>
<td>0.9</td>
<td>5.7</td>
<td>5.7</td>
<td>3.8</td>
<td>17.5</td>
<td>33.2</td>
<td>57.7</td>
<td>93.2</td>
<td>143.0</td>
<td>213.5</td>
<td>306.5</td>
<td>414.8</td>
<td>561.1</td>
<td>743.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PDE 1-1</th>
<th>PDE 2-1</th>
<th>PDE 3-1</th>
<th>PDE 4-1</th>
<th>PDE 5-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESH</td>
<td>ACCURACY</td>
<td>SOR JACOBI</td>
<td>ACCURACY</td>
<td>SOR JACOBI</td>
</tr>
<tr>
<td>3x3</td>
<td>9.5E-3</td>
<td>0.16</td>
<td>0.39</td>
<td>3.0E-3</td>
</tr>
<tr>
<td>5x5</td>
<td>7.7E-3</td>
<td>1.00</td>
<td>1.45</td>
<td>2.5E-4</td>
</tr>
<tr>
<td>7x7</td>
<td>1.0E-4</td>
<td>2.79</td>
<td>3.12</td>
<td>5.4E-5</td>
</tr>
<tr>
<td>8x8</td>
<td>1.0E-4</td>
<td>3.18</td>
<td>3.62</td>
<td>3.3E-5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PDE 5-4</th>
<th>PDE 6-1</th>
<th>PDE 7-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESH</td>
<td>ACCURACY</td>
<td>SOR JACOBI</td>
</tr>
<tr>
<td>3x3</td>
<td>2.5E-2</td>
<td>0.18</td>
</tr>
<tr>
<td>5x5</td>
<td>3.8E-4</td>
<td>0.17</td>
</tr>
<tr>
<td>7x7</td>
<td>1.4E-3</td>
<td>0.35</td>
</tr>
<tr>
<td>8x8</td>
<td>1.4E-3</td>
<td>0.52</td>
</tr>
</tbody>
</table>
The performance data of Yale Sparse for the DEC VAX. The execution time (in seconds) for indexing the linear equations is Time 2; the time to solve the linear system is Time 3.

<table>
<thead>
<tr>
<th>MESH</th>
<th>ACCURACY</th>
<th>SOR</th>
<th>JACOBI</th>
<th>MESH</th>
<th>ACCURACY</th>
<th>SOR</th>
<th>JACOBI</th>
<th>MESH</th>
<th>ACCURACY</th>
<th>SOR</th>
<th>JACOBI</th>
</tr>
</thead>
<tbody>
<tr>
<td>POE 10-2</td>
<td>3x3</td>
<td>5.7E-2</td>
<td>0.3</td>
<td>0.3</td>
<td>POE 10-3</td>
<td>0.3E-2</td>
<td>0.3</td>
<td>0.2</td>
<td>POE 28-3</td>
<td>4.1E-2</td>
<td>0.0</td>
</tr>
<tr>
<td>POE 10-2</td>
<td>5x5</td>
<td>6.9E-2</td>
<td>1.2</td>
<td>1.0</td>
<td>POE 10-3</td>
<td>8.7E-2</td>
<td>1.3</td>
<td>1.0</td>
<td>POE 28-3</td>
<td>1.0E-2</td>
<td>2.0</td>
</tr>
<tr>
<td>POE 10-2</td>
<td>7x7</td>
<td>9.1E-2</td>
<td>5.2</td>
<td>3.2</td>
<td>POE 10-3</td>
<td>5.9E-3</td>
<td>3.1</td>
<td>3.2</td>
<td>POE 28-3</td>
<td>9.8E-2</td>
<td>4.9</td>
</tr>
<tr>
<td>POE 10-2</td>
<td>9x9</td>
<td>1.7E-2</td>
<td>7.9</td>
<td>6.9</td>
<td>POE 10-3</td>
<td>1.2E-3</td>
<td>7.3</td>
<td>6.6</td>
<td>POE 28-3</td>
<td>2.0E-2</td>
<td>11.0</td>
</tr>
<tr>
<td>POE 10-2</td>
<td>13x13</td>
<td>7.2E-2</td>
<td>27.2</td>
<td>20.7</td>
<td>POE 10-3</td>
<td>2.0E-4</td>
<td>26.5</td>
<td>20.5</td>
<td>POE 28-3</td>
<td>3.0E-2</td>
<td>50.9</td>
</tr>
<tr>
<td>POE 10-2</td>
<td>17x17</td>
<td>1.1E-2</td>
<td>60.8</td>
<td>41.2</td>
<td>POE 10-3</td>
<td>1.0E-4</td>
<td>62.0</td>
<td>42.3</td>
<td>POE 28-3</td>
<td>2.9E-2</td>
<td>53.3</td>
</tr>
<tr>
<td>POE 10-2</td>
<td>21x21</td>
<td>1.3E-2</td>
<td>110.3</td>
<td>70.8</td>
<td>POE 10-3</td>
<td>4.7E-5</td>
<td>120.9</td>
<td>76.3</td>
<td>POE 28-3</td>
<td>4.1E-3</td>
<td>119.3</td>
</tr>
<tr>
<td>POE 10-2</td>
<td>25x25</td>
<td>3.6E-2</td>
<td>218.9</td>
<td>116.9</td>
<td>POE 10-3</td>
<td>1.4E-5</td>
<td>197.3</td>
<td>127.5</td>
<td>POE 28-3</td>
<td>3.5E-2</td>
<td>214.3</td>
</tr>
<tr>
<td>POE 10-2</td>
<td>29x29</td>
<td>1.2E-6</td>
<td>271.7</td>
<td>170.7</td>
<td>POE 10-3</td>
<td>2.8E-6</td>
<td>302.6</td>
<td>184.8</td>
<td>POE 28-3</td>
<td>3.7E-2</td>
<td>294.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MESH</th>
<th>ACCURACY</th>
<th>SOR</th>
<th>JACOBI</th>
<th>MESH</th>
<th>ACCURACY</th>
<th>SOR</th>
<th>JACOBI</th>
<th>MESH</th>
<th>ACCURACY</th>
<th>SOR</th>
<th>JACOBI</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDE 41-1</td>
<td>3x3</td>
<td>6.4E-2</td>
<td>0.4</td>
<td>0.3</td>
<td>PDE 44-1</td>
<td>1.6E-2</td>
<td>0.3</td>
<td>0.3</td>
<td>PDE 44-2</td>
<td>4.5E-2</td>
<td>0.8</td>
</tr>
<tr>
<td>PDE 41-1</td>
<td>5x5</td>
<td>1.4E-3</td>
<td>1.4</td>
<td>1.0</td>
<td>PDE 44-1</td>
<td>2.4E-4</td>
<td>1.3</td>
<td>1.0</td>
<td>PDE 44-2</td>
<td>1.5E-2</td>
<td>1.6</td>
</tr>
<tr>
<td>PDE 41-1</td>
<td>7x7</td>
<td>1.4E-3</td>
<td>3.8</td>
<td>3.0</td>
<td>PDE 44-1</td>
<td>6.9E-5</td>
<td>3.6</td>
<td>3.4</td>
<td>PDE 44-2</td>
<td>4.5E-3</td>
<td>4.1</td>
</tr>
<tr>
<td>PDE 41-1</td>
<td>9x9</td>
<td>2.1E-3</td>
<td>6.6</td>
<td>6.2</td>
<td>PDE 44-1</td>
<td>5.1E-5</td>
<td>7.6</td>
<td>7.4</td>
<td>PDE 44-2</td>
<td>2.7E-3</td>
<td>7.1</td>
</tr>
<tr>
<td>PDE 41-1</td>
<td>13x13</td>
<td>1.4E-3</td>
<td>16.2</td>
<td>16.2</td>
<td>PDE 44-1</td>
<td>3.8E-6</td>
<td>28.7</td>
<td>20.1</td>
<td>PDE 44-2</td>
<td>5.7E-4</td>
<td>16.4</td>
</tr>
<tr>
<td>PDE 41-1</td>
<td>17x17</td>
<td>1.6E-3</td>
<td>33.5</td>
<td>32.0</td>
<td>PDE 44-1</td>
<td>1.0E-5</td>
<td>77.9</td>
<td>41.0</td>
<td>PDE 44-2</td>
<td>6.4E-4</td>
<td>35.4</td>
</tr>
<tr>
<td>PDE 41-1</td>
<td>21x21</td>
<td>2.1E-3</td>
<td>61.2</td>
<td>51.5</td>
<td>PDE 44-1</td>
<td>1.1E-5</td>
<td>111.9</td>
<td>72.5</td>
<td>PDE 44-2</td>
<td>7.0E-4</td>
<td>56.2</td>
</tr>
<tr>
<td>PDE 41-1</td>
<td>25x25</td>
<td>1.4E-3</td>
<td>102.1</td>
<td>87.6</td>
<td>PDE 44-1</td>
<td>8.1E-6</td>
<td>294.6</td>
<td>123.4</td>
<td>PDE 44-2</td>
<td>7.2E-4</td>
<td>94.5</td>
</tr>
<tr>
<td>PDE 41-1</td>
<td>29x29</td>
<td>1.4E-5</td>
<td>165.0</td>
<td>144.5</td>
<td>PDE 44-1</td>
<td>7.6E-6</td>
<td>324.0</td>
<td>218.0</td>
<td>PDE 44-2</td>
<td>4.7E-4</td>
<td>135.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MESH</th>
<th>ACCURACY</th>
<th>SOR</th>
<th>JACOBI</th>
<th>MESH</th>
<th>ACCURACY</th>
<th>SOR</th>
<th>JACOBI</th>
<th>MESH</th>
<th>ACCURACY</th>
<th>SOR</th>
<th>JACOBI</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDE 44-3</td>
<td>3x3</td>
<td>1.8E-3</td>
<td>0.3</td>
<td>0.3</td>
<td>PDE 44-3</td>
<td>2.4E-3</td>
<td>0.3</td>
<td>0.3</td>
<td>PDE 44-3</td>
<td>4.5E-2</td>
<td>0.8</td>
</tr>
<tr>
<td>PDE 44-3</td>
<td>5x5</td>
<td>1.4E-3</td>
<td>1.3</td>
<td>1.0</td>
<td>PDE 44-3</td>
<td>2.4E-4</td>
<td>1.3</td>
<td>1.0</td>
<td>PDE 44-3</td>
<td>1.5E-2</td>
<td>1.6</td>
</tr>
<tr>
<td>PDE 44-3</td>
<td>7x7</td>
<td>6.9E-5</td>
<td>3.4</td>
<td>3.4</td>
<td>PDE 44-3</td>
<td>5.1E-5</td>
<td>3.6</td>
<td>3.4</td>
<td>PDE 44-3</td>
<td>4.5E-3</td>
<td>4.1</td>
</tr>
<tr>
<td>PDE 44-3</td>
<td>9x9</td>
<td>5.2E-5</td>
<td>8.0</td>
<td>5.7</td>
<td>PDE 44-3</td>
<td>5.2E-5</td>
<td>7.6</td>
<td>7.4</td>
<td>PDE 44-3</td>
<td>2.7E-3</td>
<td>7.1</td>
</tr>
<tr>
<td>PDE 44-3</td>
<td>13x13</td>
<td>5.8E-6</td>
<td>28.7</td>
<td>19.5</td>
<td>PDE 44-3</td>
<td>3.8E-6</td>
<td>28.7</td>
<td>20.1</td>
<td>PDE 44-3</td>
<td>5.7E-4</td>
<td>16.4</td>
</tr>
<tr>
<td>PDE 44-3</td>
<td>17x17</td>
<td>1.1E-5</td>
<td>109.6</td>
<td>71.1</td>
<td>PDE 44-3</td>
<td>1.1E-5</td>
<td>111.9</td>
<td>72.5</td>
<td>PDE 44-3</td>
<td>6.4E-4</td>
<td>35.4</td>
</tr>
<tr>
<td>PDE 44-3</td>
<td>21x21</td>
<td>1.4E-5</td>
<td>251.9</td>
<td>137.5</td>
<td>PDE 44-3</td>
<td>1.1E-5</td>
<td>294.6</td>
<td>123.4</td>
<td>PDE 44-3</td>
<td>7.2E-4</td>
<td>94.5</td>
</tr>
<tr>
<td>PDE 44-3</td>
<td>25x25</td>
<td>8.4E-6</td>
<td>202.1</td>
<td>137.5</td>
<td>PDE 44-3</td>
<td>7.6E-6</td>
<td>324.0</td>
<td>218.0</td>
<td>PDE 44-3</td>
<td>4.7E-4</td>
<td>135.6</td>
</tr>
</tbody>
</table>

Table 5. The performance data of Yale Sparse for the DEC VAX. The execution time (in seconds) for indexing the linear equations is Time 2; the time to solve the linear system is Time 3.
<table>
<thead>
<tr>
<th>MESH</th>
<th>Time 2</th>
<th>Time 3</th>
<th>Time 2</th>
<th>Time 3</th>
<th>Time 2</th>
<th>Time 3</th>
<th>Time 2</th>
<th>Time 3</th>
<th>Time 2</th>
<th>Time 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3x3</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>5x5</td>
<td>0.6</td>
<td>0.8</td>
<td>0.5</td>
<td>0.8</td>
<td>0.6</td>
<td>0.8</td>
<td>0.6</td>
<td>0.8</td>
<td>0.6</td>
<td>0.8</td>
</tr>
<tr>
<td>7x7</td>
<td>1.9</td>
<td>3.4</td>
<td>1.7</td>
<td>3.1</td>
<td>1.8</td>
<td>3.1</td>
<td>1.9</td>
<td>3.1</td>
<td>1.9</td>
<td>3.1</td>
</tr>
<tr>
<td>9x9</td>
<td>4.9</td>
<td>8.8</td>
<td>5.0</td>
<td>9.8</td>
<td>5.0</td>
<td>9.8</td>
<td>5.0</td>
<td>9.8</td>
<td>5.0</td>
<td>9.8</td>
</tr>
<tr>
<td>13x13</td>
<td>12.9</td>
<td>35.9</td>
<td>15.0</td>
<td>49.0</td>
<td>14.0</td>
<td>49.0</td>
<td>13.2</td>
<td>40.9</td>
<td>13.2</td>
<td>40.9</td>
</tr>
<tr>
<td>17x17</td>
<td>32.1</td>
<td>115.2</td>
<td>35.5</td>
<td>139.1</td>
<td>29.4</td>
<td>106.0</td>
<td>29.4</td>
<td>105.6</td>
<td>29.4</td>
<td>105.6</td>
</tr>
<tr>
<td>21x21</td>
<td>55.6</td>
<td>239.1</td>
<td>58.1</td>
<td>252.4</td>
<td>48.7</td>
<td>221.1</td>
<td>50.8</td>
<td>274.2</td>
<td>50.8</td>
<td>274.2</td>
</tr>
<tr>
<td>25x25</td>
<td>104.6</td>
<td>523.3</td>
<td>102.5</td>
<td>485.5</td>
<td>84.0</td>
<td>482.1</td>
<td>79.5</td>
<td>400.3</td>
<td>79.5</td>
<td>400.3</td>
</tr>
<tr>
<td>29x29</td>
<td>155.9</td>
<td>878.0</td>
<td>116.5</td>
<td>688.6</td>
<td>115.5</td>
<td>693.5</td>
<td>113.8</td>
<td>736.4</td>
<td>113.8</td>
<td>736.4</td>
</tr>
</tbody>
</table>

Figure 1. A typical set of data comparing the linear equations solution time for iterative and direct methods. The plot is log of error on a 20x20 grid versus log of time in seconds.