

Benchmarks—LINPACK and MATLAB

Fame and fortune from megaflops

by Cleve Moler



© Jeff Hummer/The Image Bank

The LINPACK benchmark was once our industry's most widely used benchmark for measuring performance in scientific computation.

The closest I've ever come to being *really* famous involves a commercial during a Monday Night Football broadcast on CBS radio. A guy goes into his neighborhood computer store and is completely turned off by the nerdy salesman making claims about "Fortran LINPACK Megaflops." So he goes to Sears, is won over by the user-friendly salesperson, and buys an IBM PS/1.

I'm at least partly responsible for the LINPACK Megaflops pitch that was disparaged in the commercial. In the late 1970's, Jack Dongarra, Pete Stewart, Jim Bunch, and I developed the LINPACK Fortran subroutine library. This is not only the basis for some of MATLAB's matrix algorithms, but is also the source of what was once our industry's most widely used benchmark for measuring performance in scientific computation. The LINPACK benchmark involves the Fortran subroutines DGEFA and DGESL, but the MATLAB equivalent would be

```
n = 100;
A = randn(n,n);
b = randn(n,1);
tic
x = A\b;
t = toc
mflops = (2/3*n^3 + 2*n^2)/t/1.e6
```

This measures the time in seconds required to solve a 100-by-100 linear system of equations. It then computes the machine's arithmetic speed in millions of floating-point operations per second, or megaflops. The n^3 term in the numerator of the last expression is roughly the number of floating point additions and multiplications required to compute $A \setminus b$. The $1.e6$ term in the denominator converts flops to megaflops.

I'm writing this column with our new Notebook for Windows (see the lead article on page 4). My input statements are in green and the resulting output is in blue. When I highlight the input statements, and press **Ctrl+Enter**, the output is inserted directly into the document.

```
t =
    0.2800
mflops =
    2.4524
```

(My computer is a PC, with an Intel 486DX2 chip, running at 66 Mhz. And, although you can't see this, each time I press **Ctrl+Enter**, I generate a new matrix and get new, maybe slightly different, times and megaflop rates.)

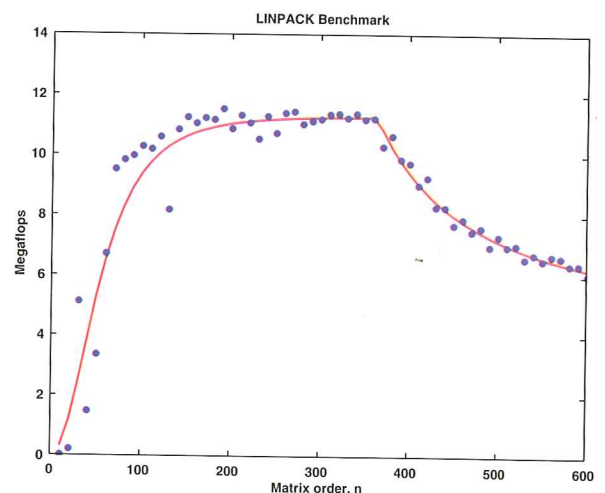
The *LINPACK User's Guide* contains a table showing the megaflop rates obtained on the mainframe computers at 22 test sites in 1977. The test sites were primarily central computer centers at various universities and government labs. Since 1977, Jack Dongarra has continued to collect these and other benchmark results, and periodically issues a report which now covers several hundred machines. (The report is available via email from netlib@ornl.gov or netlib@research.att.com. Your request should say "send performance.ps from benchmark".)

The original matrix size, $n = 100$, was chosen to be about as large as we could expect the test sites to handle in 1977. But today, it's a tiny problem. The clock that is readily accessible on most PCs, Macs, and workstations has a resolution of only 0.01 seconds, and can solve a 100-by-100 system in a handful of clock ticks.

It is more interesting to solve systems of increasing size, say, with a loop like

```
for n = 10:10:600
```

and then plot the megaflop rate as a function of n . The plot shows the results from a SPARC-10.



The circles are the actual measured megaflops rates for one run. The solid line is obtained by modeling the measured times with two different cubics in n —one for $n \leq 360$ and one for $n > 360$ —and then using the model to compute megaflops. The break point in the graph is determined by the size of the cache memory on the SPARC-10. This machine has a one-megabyte cache, which corresponds to a square matrix of order

```
n = floor(sqrt(2^20/8))
n =
    362
```

The peak megaflop rate is obtained for a matrix that just fits in the cache. Smaller matrices have more indexing overhead; larger matrices require more access to the slower main memory. These cache effects are most pronounced for large matrices on fast workstations and supercomputers. They are one of the primary motivations for the LAPACK project, which is a successor to the LINPACK project, but that's a story for another time.

Even though the LINPACK benchmark measures just one thing—the speed with which a particular computer can solve a system of simultaneous linear equations—we've found that it is a pretty fair indication of the machine's overall speed for technical computation involving floating-point arithmetic. If I can have just one performance number, I'd still make it the LINPACK number.

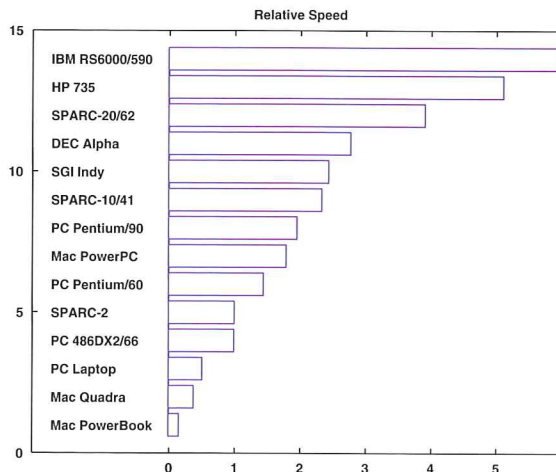
But, I'd much prefer to have more than one performance number. So, we've developed the MATLAB benchmark, which is in the demo directory with version 4.0. For this benchmark, we've chosen five basic tasks which represent different aspects of overall MATLAB performance. Each of the five tasks is scaled, so that it requires one second of computation on the SPARC-2 that was our primary development machine a couple of years ago. And, each task is usually done ten times, so the "chunk" being timed takes several hundred clock ticks. The five tasks are:

1. **Loops:** A combination of `for` loops and matrix allocation using `zeros`. This measures the speed of the MATLAB interpreter and of the underlying operating system's dynamic memory allocation.
2. **LU:** The triangular factorization of a 167-by-167 matrix. The order was chosen to take one second on the SPARC-2. This is like the LINPACK benchmark and measures the speed of floating point arithmetic.
3. **Sparse:** Solve a sparse system of linear equations. Although this is not a typical computation, it does involve

a combination of integer, floating-point, logical, and indexing operations that is representative of a broad cross-section of MATLAB functions.

4. **3-D:** `surf(peaks(24))`. This measures the speed of polygonal fill graphics.
5. **2-D:** `plot(fft(eye(52)))`. This measures the speed of line drawing graphics.

We've recently run the MATLAB benchmark program on all of the different computers we have available at The MathWorks, and asked a few people to run it on other, new computers that we don't have yet. The execution times obtained on 14 of these machines are given in the table on the next page. Smaller values represent faster times. The resulting speeds are summarized in the bar graph below. Since speed is proportional to the reciprocals of the times, longer bars represent faster speeds.



We tried to include the "top of the line" machine from each of several Unix workstation manufacturers, as well as a couple of PCs, a couple of Macs, and a couple of laptops. The Power Mac version of MATLAB had not yet been released when we made these runs (because the conversion was not complete) but the portion used by the benchmark was in good shape. We haven't included any of the Cray or Convex supercomputers because they are time-sharing machines that don't do their own graphics.

I want to emphasize that this is just one snapshot of a constantly changing scene. It so happens that the IBM RS/6000 model 590, and the Hewlett-Packard model 735 are on the top of the list today. However, as I write this column, we're working on the port to the MIPS R8000 in the SGI Challenge series, which appears to be a really fast machine. We're having some trouble with 64-bit integers, so I don't have benchmark results yet, but this might well top the next version of the list. And then Sun, or DEC, or somebody else, could be the champion after that.

continued on next page

We've recently run the MATLAB benchmark program on all of the different computers we have available, and asked others to run it on newer machines we don't have yet.



The performance gap between workstations and micros is closing, but it's still true that all of the workstations on the list are faster than all of the micros.

Benchmark Execution Time

Platform	Loops	LU	Sparse	3-D	2-D
IBM RS6000/590	1.38	0.67	1.93	2.60	1.78
HP 735	1.35	1.34	2.52	2.43	2.14
SPARC-20/62	2.18	1.94	3.41	2.73	2.53
DEC Alpha, 3500	2.88	2.58	2.68	5.67	4.22
SGI Indy, R4000	3.55	2.40	4.12	5.20	5.25
SPARC-10/41	3.48	2.89	4.78	5.06	5.23
PC Pentium/90	4.69	3.96	3.45	7.24	6.28
Mac PowerPC, 8100	4.78	3.99	3.53	7.77	7.91
PC Pentium/60	6.45	5.26	4.80	9.17	8.99
SPARC-2	10.00	10.00	10.00	10.00	10.00
PC 486DX2/66	5.59	9.96	8.26	13.70	12.77
PC Laptop, 486DX2/40	14.10	15.40	13.20	30.50	25.00
Mac Quadra, 700	26.30	18.40	20.60	36.50	29.50
Mac PowerBook, 165C	34.00	84.60	74.80	65.40	65.40

It's interesting to look at the individual execution times. Some machines are faster with memory management and integer operations, some are faster with floating-point computations, and some are faster with graphics.

A couple of the desktop PCs and Macs we used have fast graphics boards, which improve the 2-D and 3-D numbers. The SPARC-20/62 is a dual-processor machine. When MATLAB runs on one processor and the X-server runs on the other, it also improves the graphics times. On the other hand, for the two laptops on the list, the graphics are relatively slow.

The performance gap between workstations and micros is closing, but it's still true that all of the workstations on the list are faster than all of the micros.

The two hot new chips for the micros, the Pentium and the Power PC, are roughly comparable.

The LU number continues to be a pretty good measure of overall performance, but it doesn't tell the whole story.

Processor clock speed doesn't tell the whole story, either. The 90 Mhz Pentium is only 30-40% faster than the 60 Mhz Pentium.

The matrix order, $n=167$, for the LU test is a good size for today's micros, but it's getting to be too small for today's workstations. We'll have to make it larger for the benchmark in MATLAB Version 5.


Some of the workstation manufacturers offer optimized math libraries. We use such libraries on the supercomputers, but haven't used them on the workstations for two reasons.

First, we can't assume everybody has these libraries and we can't distribute them with MATLAB. Second, these libraries only affect the LU portion of the benchmark.

We'll include these results in the version of bench.m that's part of MATLAB 4.2c, which should be available by the time this appears in print.

Please don't use these numbers as the only basis for choosing a particular computer. There are many other aspects of selecting a computer that are more important than benchmarks.

How do the 1994 MATLAB benchmark results compare with the 1977 LINPACK results? The times required by the various college campus mainframes on our original test site list to solve a 100-by-100 linear system ranged from .506 seconds on the CDC Cyber 175 at the University of Illinois to 17.1 seconds on the KA-10 at Yale. If we scale these times by $(167/100)^3$ to compare them with the LU times in today's table, we find them falling in the lower half of the chart. Today's workstations and personal computers are faster, have more memory, and far better graphics capabilities, than the central computer center mainframes of 17 years ago.

But the most important change in 17 years is that I don't have to go down to the comp center after dinner to pick up my output anymore. 

Cleve Moler is chairman and co-founder of The MathWorks. His email address is moler@mathworks.com