RC 21862 (98393) October 25, 2000
Computer Science/Mathematics

IBM Research Report

An Experimental Comparison of Some Direct Sparse Solver Packages

Anshul Gupta

IBM Research Division
T.J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598

anshul@watson.ibm.com

Yanto Muliadi

Scientific Computing and Computational Mathematics Department
Stanford University
2140, S.C.C.M.
Stanford, CA, 94305

cylee@stanford.edu

Research Division
Almaden · Austin · China · Delhi · Haifa · Tokyo · Watson · Zurich

An Experimental Comparison of Some Direct Sparse Solver Packages

Anshul Gupta
IBM T.J.Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598
anshul@watson.ibm.com

Yanto Muliadi
Scientific Computing and Computational Mathematics Department
Stanford University
2140, S.C.C.M., Stanford, CA, 94305
cylee@stanford.edu

IBM Research Report RC 21862 (98393)
October 25, 2000

Abstract

This paper reports the results of an empirical comparison of a set of commonly used software packages for solving general sparse systems of linear equations. The packages included in this study are MUMPS, S+, SPOOLES, SuperLU, UMFPACK, and WSMP. The study includes results from distributed-memory and shared-memory parallel versions of the packages, where applicable. At the time of writing this report, MUMPS, S+, SPOOLES, and SuperLU had MPI-based distributed-memory parallel versions, SPOOLES and SuperLU had Pthreads-based shared-memory versions, and UMFPACK and WSMP included only single-CPU codes. All experiments were conducted on a 24-CPU IBM RS6000 S/80. To the best of our knowledge, this is a first extensive comparison of all these software packages on the same hardware and the same set of real industrial problems. Therefore, we hope that this report will serve as a useful reference for practitioners who are often faced with the decision of choosing one of the packages for their applications.

1 Introduction

In this paper, we present the results of an experimental comparative study of general sparse matrix factorization in a set of commonly used software packages for solving sparse systems of linear equations. The packages included in the experiments are MUMPS [1], S+ [11], SPOOLES [2], SuperLU [4, 5, 10], UMFPACK [3], and WSMP[7]. The study includes results from distributed-memory and shared-memory parallel versions of the packages, where applicable. At the time of writing this report, MUMPS, S+, SPOOLES, and SuperLU had MPI-based distributed-memory parallel versions, SPOOLES and SuperLU had Pthreads-based shared-memory versions, and UMFPACK and WSMP included only single-CPU codes. All experiments were conducted on a 24-CPU IBM RS6000 S/80. To the best of our knowledge, this is a first extensive comparison of all these software packages on the same hardware and the same set of real industrial problems. Therefore, we hope that this report will serve as a useful reference for practioners who are often faced the decision of choosing one of the packages for their applications.

2 Experimental Setup

The experiments were conducted on a set of large unsymmetric sparse matrices (or at least among the largest available at the time of writing this report) collected from various sources. Table 1 lists all the test matrices, their dimensions, number of nonzeros, and application areas where they came from. The matrix "tib" was provided by Kai Shen at UC santa Barbara, "bbmat," "ecl32," and "nasasrb" were provided by Xiaoye Li at NERSC, and the others were obtained from Tim Davis' collection (*www.cise.ufl.edu/~davis/sparse*) and Matrix Market (*math.nist.gov/MatrixMarket*).

For each run, we report the factorization time, operation count, Megaflops, the number of nonzeros in the factor (as reported by the program's symbolic analysis phase), and the amount of memory used by factorization. We do not report the reordering and symbolic factorization times because we believe that in most practical settings, these symbolic steps are performed only once for solving several sparse systems with the coefficient matrices of the same nonzero pattern. Moreover, the symbolic analysis times of different packages vary within a much smaller range than the factorization times, except perhaps for UMFPACK, whose reordering and symbolic factorization steps are merged with numerical factorization, and hence there is hardly any preprocessing time.

One of the ground rules for the experiments was that all input parameters that may influence the behavior of the program were fixed and were not modified to accommodate the demands of individual matrices. For example, we tested all the packages on all matrices in our test suite for various values of the pivoting threshold, and for each, chose the smallest value for which all completed runs yielded solutions (after iterative refinement) that were within the acceptable range of error for all matrices. The acceptable forward error was different for each matrix depending on it condition number, but we demanded forward and the backward errors to be of the roughly the same order of magnitude for a given marix for all packages. Table 2 shows the pivoting thresholds used with various softwares. We could not find an easy way to introduce threshold pivoting in S+ and ran the code as is, which is equivalent to a threshold of 1. The distributed-memory version of SuperLU (SuperLU_MPI) does not have a provision for partial pivoting; hence the threshold is 0. Other parameters easily accessible in the software, such as the various block sizes in SuperLU, were also fixed once to values that appeared to be the best on average.

All experiments were performed on a 24-CPU RS6000 S/80, although at most 16 CPUs were used in any experiment. Each CPU was a 600 Mhz. PowerPC with a peak Megaflop rate of 1200. The system had 64 Gbytes of RAM, 8 Mbytes of direct mapped level 2 cache and each CPU had 64 Kbytes of 2-way associative level 1 cache. All codes were compiled with the -O3 optimization level of the xlC (C/C++) or xlf (Fortran) compilers. All compilation and execution was performed in 32-bit

Matrix	*N*	*NNZ*	Application
af23560	23560	484256	Fluid dynamics
av41092	41092	1683902	F.E.M.
bayer01	57735	277774	Chemistry
bbmat	38744	1771722	Fluid dynamics
comp2c	16783	578665	Linear programming
cry10000	10000	49699	Material science
e40r0000	17281	553956	Fluid dynamics
e40r1000	17281	553956	Fluid dynamics
e40r5000	17281	553956	Fluid dynamics
ecl32	51993	380415	Electrical engineering
epb3	84617	463625	Thermodynamics
fidap011	16614	1091362	F.E.M.
fidap035	19716	218308	F.E.M.
fidapm11	22294	623554	F.E.M.
lhr34c	35152	764014	Chemical engineering
lhr71c	70304	1528092	Chemical engineering
memplus	17758	126150	Circuit simulation
nasasrb	54870	2677324	F.E.M.
onetone1	36057	341088	Circuit simulation
onetone2	36507	227628	Circuit simulation
pre2	659033	5959282	Circuit simulation
raefsky3	21200	1488768	Fluid dynamics
raefsky4	19779	1316789	Fluid dynamics
rma10	46835	2374001	Fluid dynamics
tib	18510	145149	Circuit simulation
twotone	120750	1224224	Circuit simulation
venkat50	62424	1717792	Fluid dynamics
wang3	26064	177168	Circuit simulation
wang4	26068	177196	Circuit simulation

Table 1: Test matrices with their order (N), the number of nonzeros (NNZ), and the application area of origin.

Package	*Alpha*
MUMPS	0.01
S+	1.00
Spooles	0.01
SuperLU	0.01
Spooles_MT	0.01
SuperLU_MT	0.01
Spooles_MPI	0.01
SuperLU_MPI	0.00
UMFPACK	0.25
WSMP	0.05

Table 2: The list of test packages and the pivoting threshold used with each of them.

mode, which means that at most 2 Gbytes of memory was available to a process. Therefore, only the distributed-memory parallel versions could use more than 2 Gbytes of aggregate memory.

3 Single CPU Results

Tables 3 through 7 list the the serial LU factorization time, floating-point operation count for factorization, serial factorization speed in Megaflops, the number of nonzeros in the factor, and the amount of memory used by factorization, respectively, for all packages except WSMP. The tables also have column entries of multi-threaded and distributed-memory versions of SPOOLES and SuperLU_MT run on a single CPU. In each row of these tables, the best number is shown in boldface. In many cases, a software would fail to complete the factorization and would either crash or return with an error. Such cases are marked by an F with a subscript $_C$ for an outright crash, $_M$ for a failure due to inadequate memory, and $_N$ for failing due to a small or zero pivot. Columns for S+ are missing from Tables 4, 5, and 6 from because we could not find a way to collect the operation-count and fill-in statistics from S+.

Except Table 7, the data for all the other tables was either reported by the programs themselves or was obtained by inserting simple timing functions at appropriate locations in the code. The amount of memory used was obtained by inserting calls to the C *mallinfo()* function just before and after the LU factorization routine in each code. This may not be a very accurate measure because a code may allocate more memory than it uses. In that case, the extra memory may never be paged in and the program may work just as efficiently with a smaller physical memory than the reported value in Table 7. We know for sure that this is the case in SuperLU and SuperLU_MT. In fact, this is the reason why there are a number of F_M entries for SuperLU and SuperLU_MT in these tables. SuperLU allocates memory for factorization in two large chunks—one each for the lower and upper triangular part of the factor. If it allocates one large block, then it may fail to allocate an adequate second block, even if a triangular portion of the factor does not use the entire first block and a smaller first block would

	Serial			*Multithread*		*MPI*			
	UMF	*SUP*	*SPO*	*SUP*	*SPO*	*SUP*	*SPO*	*S+*	*MUM*
af23560	45.45	30.60	11.19	31.96	11.31	14.77	10.49	81.86	**8.930**
av41092	186.2	F_M	F_M	772.8	F_M	F_N	F_M	327.6	**30.59**
bayer01	**1.761**	1.940	F_M	2.400	F_M	3.230	F_M	90.71	2.262
bbmat	682.6	F_M	112.3	214.1	111.9	F_M	**97.72**	621.3	113.0
comp2c	120.0	F_M	197.3	3403.	196.9	42.04	287.6	**12.34**	29.32
cry10000	0.425	F_N	0.452	8.460	0.516	0.490	0.502	5.200	**0.362**
e40r0000	29.90	F_M	2.504	42.24	2.566	1.470	2.645	24.85	**1.184**
e40r1000	29.43	F_M	6.420	42.35	6.460	1.500	6.711	24.57	**1.188**
e40r5000	29.67	F_M	341.2	43.95	342.4	2.080	395.7	24.45	**1.186**
ecl32	F_M	F_M	445.6	F_M	445.0	F_M	562.1	F_M	**144.9**
epb3	29.76	23.47	**4.778**	24.24	4.907	5.670	4.998	114.1	5.689
fidap011	168.3	39.91	13.19	F_M	13.21	16.94	**12.25**	124.0	18.68
fidap035	1.215	0.850	1.129	0.990	1.197	1.130	1.235	11.53	**0.820**
fidapm11	944.6	F_M	16.30	88.89	16.37	F_N	**15.10**	348.6	25.26
lhr34c	**3.464**	F_M	F_M	F_M	F_M	11.68	F_M	46.04	5.526
lhr71c	**6.802**	11.35	F_M	12.45	F_M	22.97	F_M	130.2	11.71
memplus	**0.310**	F_M	0.481	F_M	0.545	0.260	0.561	8.498	0.320
nasasrb	81.85	102.3	**26.49**	F_M	26.62	F_M	25.04	338.8	26.79
onetone1	12.22	185.7	104.8	184.0	104.8	10.70	112.8	72.80	**9.998**
onetone2	**1.793**	26.92	18.88	28.29	19.06	3.550	20.69	14.20	2.815
pre2	F_M	F_C	F_M	F_C	F_M	F_M	F_M	F_M	F_C
raefsky3	39.05	146.8	10.31	145.9	10.31	**6.860**	10.04	78.54	8.754
raefsky4	109.5	1983.	191.8	F_M	190.3	28.11	156.8	313.2	**27.50**
rma10	15.69	F_M	10.24	F_M	10.28	**5.780**	10.73	131.1	6.624
tib	**0.524**	265.4	1.598	266.1	1.694	1.470	1.749	164.7	0.618
twotone	**30.00**	F_M	839.5	F_C	827.1	637.3	724.0	F_M	124.7
venkat50	16.22	F_M	11.36	F_M	11.40	**8.110**	11.56	196.7	9.432
wang3	106.7	3352.	91.65	3226.	90.98	36.87	62.70	699.9	**32.33**
wang4	97.30	313.2	20.97	318.5	21.14	23.68	**16.26**	599.3	25.66

Table 3: LU Factorization time on a single CPU (in seconds). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination. UMF=UMFPACK, SUP=SuperLU, SPO=Spooles, MUM=MUMPS.

	Serial			*Multithread*	*MPI*	
	UMF	*SUP*	*SPO*	*SUP*	*SUP*	*MUM*
af23560	16132	4037.	2566.	4036.	4943.	**2557.**
av41092	59271	F_M	F_M	69431	F_N	**8416.**
bayer01	**31.07**	45.52	F_M	88.22	34.54	124.7
bbmat	237e3	F_M	**22859**	27766	F_M	41353
comp2c	7562.	F_M	7967.	272e3	**440.4**	4222.
cry10000	70.18	F_N	22.52	765.6	**19.84**	24.89
e40r0000	8701.	F_M	**158.1**	6581.	248.7	172.1
e40r1000	8980.	F_M	**158.1**	6582.	267.3	172.1
e40r5000	8643.	F_M	**158.1**	6609.	460.6	172.1
ecl32	F_M	F_M	**21274**	F_M	F_M	64553
epb3	8221.	2312.	596.8	2308.	**460.7**	1168.
fidap011	57832	5378.	**3080.**	F_M	6438.	7008.
fidap035	212.3	47.08	72.03	**47.07**	66.71	55.45
fidapm11	342e3	F_M	**3115.**	11244	F_N	9667.
lhr34c	302.5	F_M	F_M	F_M	**244.9**	641.0
lhr71c	**542.0**	895.8	F_M	895.5	520.6	1430.
memplus	1.813	F_M	5.779	F_M	**1.624**	2.135
nasasrb	28565	13122	**6082.**	F_M	F_M	9452.
onetone1	3309.	14780	85936	14750	**1349.**	2288.
onetone2	**159.2**	2921.	1406.	3022.	225.7	510.3
pre2	F_M	F_C	F_M	F_C	F_M	F_M
raefsky3	13222	21323.	**2341.**	21427	2682.6	2905.
raefsky4	37789	376e3	22136	F_M	**10530**	10902
rma10	4790.	F_M	**1354.**	F_M	1602.	1396.
tib	54.2	38949	89.16	38932	**31.76**	44.64
twotone	5507.	F_M	30631	F_C	**4831.**	29338
venkat50	4912.	F_M	**1954.**	F_M	2280.	2310.
wang3	38803	458e3	**13455**	447e3	14126	13782
wang4	35924	34876	**5289.**	34854	8776.	10514

Table 4: Number of floating point operations (in millions). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination. UMF=UMFPACK, SUP=SuperLU, SPO=Spooles, MUM=MUMPS. All versions of Spooles generate the same number of operations.

	Serial			*Multithread*		*MPI*		
	UMF	*SUP*	*SPO*	*SUP*	*SPO*	*SUP*	*SPO*	*MUM*
af23560	**355.0**	131.9	229.3	125.5	226.8	335.1	244.5	286.4
av41092	**318.4**	F_M	F_M	89.52	F_M	F_N	F_M	275.1
bayer01	17.64	45.52	F_M	36.58	F_M	10.66	F_M	**55.14**
bbmat	346.6	F_M	203.6	129.7	204.3	F_M	233.8	**365.8**
comp2c	63.00	F_M	40.37	80.19	40.47	10.48	27.70	**144.0**
cry10000	**165.1**	F_N	49.77	90.52	43.67	40.48	44.88	68.67
e40r0000	**291.0**	F_M	63.13	155.6	61.60	170.3	59.76	71.26
e40r1000	**305.1**	F_M	24.62	155.2	24.47	177.04	23.55	145.3
e40r5000	**291.3**	F_M	0.463	150.1	0.462	221.5	0.399	145.1
ecl32	F_M	F_M	47.74	F_M	47.81	F_M	37.83	**445.5**
epb3	**276.3**	98.51	124.9	94.21	121.6	115.7	119.4	205.3
fidap011	343.6	134.8	233.5	F_M	233.2	**380.1**	251.5	375.1
fidap035	**174.7**	55.38	63.77	47.67	60.16	59.56	58.34	67.67
fidapm11	362.2	F_M	191.1	126.0	190.2	F_N	206.3	**382.7**
lhr34c	87.30	F_M	F_M	F_M	F_M	20.86	F_M	**116.0**
lhr71c	79.67	78.92	F_M	71.51	F_M	22.60	F_M	**122.1**
memplus	5.846	F_M	**12.02**	F_M	10.61	6.246	10.30	6.68
nasasrb	349.0	128.3	229.6	F_M	228.5	F_M	242.9	**352.8**
onetone1	**270.8**	79.59	82.00	79.59	82.00	126.1	76.17	228.9
onetone2	88.76	108.5	74.49	106.2	73.78	63.57	67.95	**181.3**
pre2	F_M	F_C	F_M	F_C	F_M	F_M	F_M	F_C
raefsky3	338.6	145.3	227.2	146.9	227.1	**389.9**	233.2	331.8
raefsky4	345.2	190.0	115.4	F_M	115.7	374.3	141.1	**396.4**
rma10	**305.4**	F_M	132.2	F_M	131.6	276.7	126.1	210.7
tib	**103.3**	146.8	55.80	74.32	52.64	21.75	50.99	72.29
twotone	183.5	F_M	36.49	F_C	37.03	7.578	42.31	**235.2**
venkat50	**302.9**	F_M	172.0	F_M	171.5	280.4	169.1	244.9
wang3	363.7	136.8	146.8	138.3	147.9	382.0	214.6	**426.2**
wang4	369.2	111.3	252.2	109.2	250.1	370.6	325.3	**409.8**

Table 5: Megaflops (Million floating point operations per second). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination. UMF=UMFPACK, SUP=SuperLU, SPO=Spooles, MUM=MUMPS.

	Serial			*Multithread*	*MPI*	
	UMF	*SUP*	*SPO*	*SUP*	*SUP*	*MUM*
af23560	22.69	14.21	**9.107**	14.21	11.391	8.340
av41092	41.27	F_M	F_M	76.69	F_N	**14.13**
bayer01	**1.124**	3.033	F_M	**3.021**	1.336	2.823
bbmat	1250.	F_M	366.7	**43.95**	F_M	46.00
comp2c	10.36	F_M	14.32	82.16	3.005	7.046
cry10000	0.712	F_N	0.502	3.950	**0.374**	0.462
e40r0000	15.51	F_M	2.358	15.00	1.982	**1.722**
e40r1000	17.81	F_M	3.359	15.00	2.028	**1.722**
e40r5000	15.48	F_M	10.47	15.03	2.531	**1.722**
ecl32	F_M	F_M	43.34	F_M	F_M	**42.86**
epb3	21.16	23.91	7.851	23.91	**4.696**	6.899
fidap011	39.48	12.91	**8.829**	F_M	11.90	12.46
fidap035	2.117	1.298	1.463	1.297	**1.224**	1.237
fidapm11	109.8	F_M	**10.71**	21.79	F_M	14.01
lhr34c	3.813	F_M	F_M	F_M	**3.621**	5.581
lhr71c	**7.350**	10.99	F_M	10.98	7.387	11.53
memplus	**0.157**	F_M	0.359	F_M	0.158	0.164
nasasrb	44.49	35.01	**21.10**	F_M	F_M	24.18
onetone1	5.401	29.69	30.96	29.69	**3.135**	4.723
onetone2	**1.337**	12.93	13.76	13.14	1.384	2.260
pre2	F_M	F_C	F_M	F_C	F_M	F_M
raefsky3	18.74	29.87	**8.097**	29.89	8.154	8.435
raefsky4	29.70	105.3	28.89	F_M	**15.19**	15.74
rma10	16.28	F_M	9.489	F_M	9.366	**8.868**
tib	0.511	29.52	1.277	29.52	**0.462**	0.691
twotone	**8.766**	F_M	84.78	F_C	10.16	22.12
venkat50	17.68	F_M	12.44	F_M	**11.70**	11.97
wang3	26.95	114.1	19.45	113.0	13.68	**1.378**
wang4	24.91	45.35	**9.331**	45.35	10.49	11.61

Table 6: Number of nonzeros in factor (in millions). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination. UMF=UMFPACK, SUP=SuperLU, SPO=Spooles, MUM=MUMPS. All versions of Spooles generate factors of the same size.

	Serial			*Multithread*		*MPI*			
	UMF	*SUP*	*SPO*	*SUP*	*SPO*	*SUP*	*SPO*	*S+*	*MUM*
af23560	375.4	1272.	**84.70**	1511.	84.72	103.1	94.86	207.0	104.1
av41092	891.9	F_M	F_M	1854.	F_M	F_N	F_M	419.5	**196.5**
bayer01	38.11	1861.	F_M	1704.	F_M	**23.53**	F_M	247.7	41.85
bbmat	1588.	F_M	**312.3**	1183.	**312.3**	F_M	621.3	796.6	580.8
comp2c	689.8	F_M	143.2	1475.	143.2	**38.38**	164.2	56.58	211.6
cry10000	12.32	F_N	6.890	1209.	6.896	**5.523**	7.861	23.30	7.390
e40r0000	264.0	F_M	23.40	1723.	23.40	**19.19**	32.60	92.45	25.17
e40r1000	257.5	F_M	31.58	1723.	31.58	**19.57**	40.80	92.27	25.17
e40r5000	256.4	F_M	89.11	1723.	89.12	**23.94**	98.67	92.01	25.17
ecl32	F_M	F_M	**366.3**	F_M	**366.3**	F_M	386.6	F_M	612.1
epb3	314.3	1250.	76.83	1547.	76.84	**57.71**	86.06	313.8	96.94
fidap011	740.5	1481.	77.08	F_M	**77.03**	100.9	97.05	228.5	167.4
fidap035	32.37	1147.	16.28	1316.	16.28	**13.83**	20.09	56.59	17.80
fidapm11	1563.	F_M	94.92	1960.	**94.86**	F_N	107.6	431.3	196.9
lhr34c	68.40	F_M	F_M	F_M	F_M	**42.50**	F_M	161.2	75.19
lhr71c	129.2	1330.	F_M	1207.	F_M	**85.82**	F_M	389.1	151.8
memplus	8.885	F_M	6.280	F_M	6.266	**4.650**	8.568	37.37	5.026
nasasrb	676.6	1595.	**183.7**	F_M	**183.7**	F_M	231.5	625.4	295.9
onetone1	142.7	1792.	273.3	1147.	273.3	**35.56**	284.5	125.8	78.62
onetone2	34.31	1197.	128.9	1422.	128.9	**18.88**	134.6	50.11	36.45
pre2	F_M	F_C	F_M	F_C	F_M	F_M	F_M	F_M	F_C
raefsky3	335.1	1225.	70.66	1492.	70.67	**68.69**	96.32	199.8	113.8
raefsky4	499.2	1545.	240.96	F_M	241.0	**129.2**	275.0	376.8	211.1
rma10	277.0	F_M	85.98	F_M	85.99	**82.38**	125.9	311.5	115.9
tib	16.12	1525.	18.06	1702.	18.07	**10.17**	20.94	213.7	12.64
twotone	207.0	F_M	735.1	F_C	735.3	**133.4**	767.9	F_M	442.3
venkat50	266.9	F_M	113.2	F_M	113.2	**105.1**	143.2	470.3	155.3
wang3	527.7	1861.	167.0	1759.	167.0	**124.7**	176.9	602.1	227.2
wang4	430.7	1884.	**84.96**	1246.	84.97	97.81	91.23	549.1	186.6

Table 7: Memory usage (in Megabytes). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination. UMF=UMFPACK, SUP=SuperLU, SPO=Spooles, MUM=MUMPS.

	Time		*Memory*		*NNZ*		*Opcount*		*MFLOPS*	
	α=0.0	α=.05	α=0.0	α=.05	α=0.0	α=.05	α=0.0	α=.05	α=0.0	α=.05
af23560	5.91	6.31	86	86	9.198	10.65	3053.	3060.	516.6	485.2
av41092	F_N	8.47	F_N	92	F_N	9.678	F_N	2312.	F_N	271.9
bayer01	1.15	1.34	17	17	1.527	1.746	39.59	39.86	34.42	29.52
bbmat	35.63	37.3	299	300	31.64	36.30	19438	19502	545.6	522.5
comp2c	1.46	4.11	24	28	2.707	3.602	459.1	855.8	314.3	208.5
cry10000	0.19	0.19	4	5	0.490	0.576	25.67	26.31	135.9	136.5
e40r0000	0.77	0.92	22	23	2.013	2.334	231.8	236.9	300.3	256.6
e40r1000	0.78	0.80	22	23	2.037	2.336	238.6	239.2	305.7	298.4
e40r5000	0.99	1.63	26	28	2.425	2.865	348.0	384.3	349.8	236.4
ecl32	39.8	41.7	231	232	25.77	29.32	22442	22459	564.1	538.6
epb3	2.16	2.19	48	49	4.834	5.510	413.7	416.3	191.4	190.1
fidap011	5.88	6.42	85	86	8.581	10.03	3038.	3052	516.7	475.1
fidap035	0.37	0.38	12	13	1.202	1.394	49.91	51.47	136.0	136.9
fidapm11	F_N	11.5	F_N	123	F_N	15.12	F_N	5492.	F_N	475.9
lhr34c	1.18	1.49	31	33	28.15	3.403	143.1	162.6	120.9	109.0
lhr71c	2.32	3.05	62	67	5.575	6.870	285.3	340.2	122.7	111.3
memplus	0.28	0.24	2	3	0.171	0.222	3.971	4.105	14.14	17.13
nasasrb	10.9	11.4	194	194	19.23	23.32	5628.	5634.	513.6	492.9
onetone1	5.77	7.26	37	39	4.015	4.520	1710.	1903.	296.6	262.3
onetone2	0.99	1.11	14	15	1.344	1.558	165.3	177.9	166.4	160.0
pre2	F_N	373.	F_N	1230	F_N	95.91	F_N	123277	F_N	330.2
raefsky3	4.47	4.66	81	81	7.863	9.065	2270.	2270.	507.9	488.3
raefsky4	7.52	7.78	101	101	10.21	11.74	4029.	4031.	535.5	518.2
rma10	3.62	3.77	105	106	9.494	10.84	1447.	1456.	399.4	386.0
tib	0.31	0.30	4	5	0.421	0.506	21.01	21.38	67.96	71.24
twotone	30.3	40.2	100	104	10.86	12.39	10103	10716	333.4	266.4
venkat50	4.35	4.51	117	118	11.53	13.06	1827.	1831.	420.1	405.5
wang3	13.5	13.8	87	88	10.22	13.13	7225.	7234.	533.8	525.7
wang4	11.8	12.3	91	91	10.24	12.38	6653.	6665.	565.5	542.4

Table 8: Performance statistics for WSMP for pivoting threshold α equal tp 0.0 and 0.05. Factorization time is in seconds, memory usage is in Megabytes, and the number of nonzeros in the factor (NNZ) and the factorization operation count are in millions.

have sufficed. On the other hand, if it allocates too small a first block, it can run out of space in this block during factorization. So, although it appears that SuperLU and SuperLU_MT run out of memory quite often, this is an artifact of the implementation with static allocation and does not really reflect the amount of memory that SuperLU actually uses. We believe that this can be easily fixed in the SuperLU software and should not be held against the robustness of its algorithms.

The most striking observation in Table 3 is the range of times that different packages take to factor the same matrix. It is not uncommon to see the fastest software being faster than the slowest one by one to two orders of magnitude. On the whole, MUMPS appears to be the fastest on our test suite, followed by UMFPACK. SuperLU_MPI and all versions of SPOOLES share the third place, although the absense of partial pivoting made the job of SuperLU_MPI easier. S+ appears to be slowest, however, it ran under a big disadvantage that it performed partial pivoting without any threshold and suffered a large degree of fill-in and extra operations as a result.

Table 4 shows that SPOOLES usually has the least operation count; i.e., it does a got job of ordering. However, it fails to translate the small operation count into a small running time as it operates at a slower Megaflop rate than its counterparts. Table 5 shows that MUMPS and UMPACK obtain the highest Megaflops. It is to UMPACK's credit that it is able to sustain a high rate of floating point operations in spite of performing ordering and symbolic factorization alongside numerical factorization. Both MUMPS and UMPACK employ the multifrontal technique for factorization and therefore obtain a higher level of performance through the use of level-3 BLAS on larger dense blocks.

Table 7 shows the difference between the amount of memory allocated between the beginning and the end of LU factorization. The amount of memory used in factorization appears to be the least for SuperLU_MPI. However, as we remarked earlier, a software may allocate more memory than it uses, and Table 7 may noy truly reflect the actual amount of memory used by each software.

Table 8 shows the performance and other statistics for WSMP. Note that in most cases, WSMP factorization time is smaller (often significantly) than the smallest time in Table 3. Most of this advantage is a result of better ordering, which is also indicated by smaller operation counts and number of factor nonzeros for WSMP. Ordering is probably also responsible for the fact that among all the packages tested, WSMP was the only one that was able to solve the largest problem "pre2" in the test suite. Often, WSMP is also able to achieve much higher Megaflops than other packages, especially when the matrix permits calls to level-3 BLAS on large supernodes. Note that Table 8 shows the results from WSMP both with (threshold = 0.05) and without (threshold = 0) partial pivoting. Numerical failure is observed in certain cases where pivoting is not used, which is not completely unexpected. However, what is noteworthy is that partial pivoting with threshold does not appear to have a significant impact on performance in most cases. This is largely because of the use of a preprocessing step that permutes the rows of the matrix such that the product of the absolute values of its diagonal entries is maximized [6, 9, 8]. MUMPS and SuperLU_MPI also use the same preprocessing technique.

	1	*2*	*4*	*6*	*8*	*10*	*12*	*16*	Max S
af23560	31.96	19.10	10.01	7.12	6.13	5.38	5.47	5.58	**5.94**
av41092	772.8	494.3	332.0	268.4	228.1	203.7	183.4	F_M	4.21
bayer01	2.400	4.27	3.76	3.58	3.57	3.61	3.71	3.64	0.67
bbmat	214.1	121.6	86.55	73.96	67.81	64.32	62.13	60.01	3.57
comp2c	3403.	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
cry10000	8.460	4.500	2.730	2.230	2.090	2.060	2.000	2.010	**4.23**
e40r0000	42.24	21.28	10.95	7.480	5.750	4.780	4.190	4.350	**10.1**
e40r1000	42.35	21.35	10.98	7.490	5.780	4.790	4.180	4.320	**10.1**
e40r5000	43.95	22.25	11.47	7.830	6.080	5.050	4.490	4.530	**9.79**
ecl32	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
epb3	24.24	15.02	9.510	9.750	9.750	10.12	10.45	10.79	2.55
fidap011	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
fidap035	0.990	0.620	0.490	0.470	0.490	0.570	0.490	0.630	2.11
fidapm11	88.89	48.75	29.49	23.74	F_M	F_M	F_C	F_M	3.74
lhr34c	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
lhr71c	12.45	**6.59**	4.58	4.26	4.13	3.96	4.09	4.36	3.14
memplus	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
nasasrb	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
onetone1	184.0	120.4	83.48	67.26	64.87	61.16	51.66	45.63	**4.03**
onetone2	28.29	21.14	13.83	11.54	10.91	10.59	9.49	9.26	3.06
pre2	F_C	F_C	F_C	F_C	F_C	F_C	F_C	F_C	-
raefsky3	145.9	73.04	36.97	25.14	18.86	15.38	13.02	10.12	**14.4**
raefsky4	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
rma10	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
tib	266.1	2525.	1310.	894.0	706.5	591.5	540.1	391.3	0.68
twotone	F_C	F_C	F_C	F_C	F_C	F_C	F_C	F_C	-
venkat50	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
wang3	3226.	1963.	1007.	686.5	530.4	433.1	367.1	282.9	**11.4**
wang4	318.5	165.5	84.81	58.85	44.75	36.85	31.47	25.49	**12.5**

Table 9: Parallel run times of SuperLU_MT (in seconds). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination. The last column indicates the maximum speedup for each matrix.

	1	*2*	*4*	*6*	*8*	*10*	*12*	*16*
af23560	1511.	1511.	1511.	1511.	1511.	1511.	1511.	1511.
av41092	1854.	1854.	1854.	1854.	1854.	1854.	1854.	1854.
bayer01	1704.	1704.	1704.	1704.	1704.	1704.	1704.	1704.
bbmat	1183.	1183.	1183.	1183.	1183.	1183.	1183.	1183.
comp2c	1475.	F_M	F_M	F_M	F_M	F_M	F_M	F_M
cry10000	1209.	1209.	1209.	1209.	1209.	1209.	1209.	1209.
e40r0000	1723.	1723.	1723.	1723.	1723.	1723.	1723.	1723.
e40r1000	1723.	1723.	1723.	1723.	1723.	1723.	1723.	1723.
e40r5000	1723.	1723.	1723.	1723.	1723.	1723.	1723.	1723.
epb3	1546.	1546.	1546.	1546.	1546.	1546.	1546.	1546.
fidap011	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
fidap035	1316.	1316.	1316.	1316.	1316.	1316.	1316.	1316.
fidapm11	1960.	1960.	1960.	1960.	F_M	F_M	F_C	F_M
lhr34c	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
lhr71c	1207.	1207.	1207.	1207.	1207.	1207.	1207.	1207.
memplus	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
onetone1	1147.	1147.	1147.	1147.	1147.	1147.	1147.	1147.
onetone2	1422.	1422.	1422.	1422.	1422.	1422.	1422.	1422.
pre2	F_C	F_C	F_C	F_C	F_C	F_C	F_C	F_C
raefsky3	1942.	1942.	1942.	1942.	1942.	1942.	1942.	1942.
raefsky4	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
rma10	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
tib	1702.	1702.	1702.	1702.	1702.	1702.	1702.	1702.
twotone	F_C	F_C	F_C	F_C	F_C	F_C	F_C	F_C
venkat50	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
wang3	1759.	1759.	1759.	1759.	1759.	1759.	1759.	1759.
wang4	1246.	1246.	1246.	1246.	1246.	1246.	1246.	1246.

Table 10: Parallel memory use by SuperLU_MT (in Megabytes). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination.

	1	*2*	*4*	*6*	*8*	*10*	*12*	*16*	Max S
af23560	11.31	8.465	6.866	6.190	5.641	5.525	5.452	5.467	2.07
av41092	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
bayer01	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
bbmat	111.9	77.26	55.53	49.49	47.97	49.52	47.54	48.07	2.36
comp2c	196.9	167.9	144.9	143.7	140.8	142.2	141.5	140.4	1.40
cry10000	0.516	0.392	0.354	0.357	0.364	0.364	0.389	0.424	1.45
e40r0000	2.566	1.809	1.508	1.441	1.366	1.310	1.335	1.355	1.96
e40r1000	6.460	4.256	3.475	3.264	2.855	2.753	2.683	2.445	2.64
e40r5000	342.4	315.2	291.0	278.2	270.2	266.4	263.2	265.2	1.29
ecl32	445.0	344.3	312.1	284.7	282.5	280.7	276.6	279.6	1.61
epb3	**4.907**	3.649	2.635	2.361	2.312	2.236	2.298	2.341	2.13
fidap011	13.21	9.511	7.387	6.596	6.287	6.312	5.910	5.967	2.23
fidap035	1.197	0.911	0.725	0.700	0.703	0.707	0.709	0.748	1.71
fidapm11	16.37	11.87	**9.083**	**7.707**	**6.556**	**6.384**	6.100	5.969	2.75
lhr34c	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
lhr71c	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
memplus	0.545	0.422	0.372	0.364	0.374	0.377	0.404	0.462	1.50
nasasrb	26.62	19.69	14.65	14.10	13.41	12.31	12.41	11.68	2.28
onetone1	104.8	69.59	53.16	48.29	47.33	44.75	42.55	42.18	2.50
onetone2	19.06	12.74	8.316	6.783	5.818	5.346	5.126	4.709	4.05
pre2	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
raefsky3	10.31	6.888	4.846	4.457	4.152	4.103	4.073	4.129	2.53
raefsky4	191.3	131.8	103.5	101.9	103.0	113.3	113.5	111.0	1.87
rma10	10.28	7.837	6.384	5.754	5.495	5.262	5.189	5.121	1.87
tib	1.694	1.281	1.111	1.060	1.055	1.054	1.028	1.128	1.65
twotone	827.1	552.0	511.3	507.9	513.4	508.1	467.6	471.8	1.77
venkat50	11.40	8.428	5.913	5.547	5.075	5.099	5.005	4.958	2.30
wang3	90.98	67.83	60.12	57.33	56.53	57.18	55.40	55.79	1.61
wang4	21.14	16.06	14.58	13.72	13.73	13.92	13.86	13.52	1.56

Table 11: Parallel run times of Spooles_MT (in seconds). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination. The last column indicates the maximum speedup for each matrix.

	1	*2*	*4*	*6*	*8*	*10*	*12*	*16*
af23560	**84.72**	**84.72**	**84.72**	**84.72**	**84.72**	**84.71**	**84.70**	**84.72**
av41092	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
bayer01	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
bbmat	**312.3**	**312.4**	**312.3**	**312.3**	**312.4**	**312.3**	**312.3**	**312.3**
comp2c	143.2	143.2	143.2	143.2	143.2	143.2	143.2	143.2
cry10000	6.896	6.896	6.896	6.896	6.888	6.896	6.896	6.896
e40r0000	23.40	23.40	**23.40**	**23.39**	**23.40**	**23.39**	**23.39**	**23.40**
e40r1000	31.58	31.58	31.58	31.58	31.58	31.58	31.59	31.58
e40r5000	89.12	89.12	89.12	89.12	89.13	89.12	89.12	89.13
ecl32	**366.3**	**366.3**	**366.3**	**366.3**	**366.4**	**366.3**	**366.4**	**366.3**
epb3	76.84	76.84	76.84	**76.85**	**76.84**	**76.85**	**76.84**	**76.85**
fidap011	**77.03**	**77.01**	**77.03**	**77.03**	**77.03**	**77.01**	**77.02**	**77.02**
fidap035	**16.28**	**16.28**	**16.28**	**16.28**	**16.28**	**16.28**	**16.28**	**16.28**
fidapm11	**94.86**	**94.87**	**94.85**	**94.85**	**94.87**	**94.85**	**94.84**	**94.87**
lhr34c	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
lhr71c	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
memplus	6.266	6.266	**6.266**	**6.266**	**6.266**	**6.266**	**6.266**	**6.266**
nasasrb	**183.7**	**183.7**	**183.7**	**183.7**	**183.7**	**183.7**	**183.7**	**183.7**
onetone1	273.3	273.2	273.2	273.3	273.1	273.1	273.2	273.1
onetone2	128.9	128.9	128.9	128.9	128.9	128.9	128.9	128.9
pre2	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
raefsky3	70.67	70.68	**70.67**	**70.66**	**70.68**	**70.68**	**70.67**	**70.69**
raefsky4	241.0	241.0	241.0	241.0	241.0	241.0	241.0	241.0
rma10	85.99	85.99	**85.98**	**86.00**	**85.99**	**85.98**	**85.99**	**85.99**
tib	18.07	18.06	18.06	18.07	**18.07**	**18.06**	**18.07**	**18.07**
twotone	735.3	735.3	735.0	**734.7**	**734.5**	**735.0**	**735.1**	**735.0**
venkat50	113.2	113.2	113.2	113.2	**113.3**	**113.2**	**113.3**	**113.2**
wang3	167.0	167.0	167.0	167.0	167.0	167.0	167.0	167.0
wang4	**84.97**	**84.97**	**84.97**	**84.95**	**84.95**	**84.95**	**84.96**	**84.97**

Table 12: Parallel memory use by Spooles_MT (in Megabytes). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination.

	1	*2*	*4*	*6*	*8*	*10*	*12*	*16*	Max S
af23560	14.77	9.140	6.130	5.530	4.960	4.240	4.680	3.640	4.06
av41092	F_N	F_N	F_N	F_N	F_M	F_N	F_N	F_N	-
bayer01	3.230	3.570	2.820	5.390	F_M	4.350	5.080	F_M	1.15
bbmat	F_M	F_M	60.87	43.10	32.75	27.93	23.99	18.82	-1
comp2c	42.04	21.47	10.92	**7.510**	**6.180**	**3.990**	**4.450**	**3.540**	**11.9**
cry10000	0.490	0.810	0.660	1.400	1.300	1.040	1.350	0.760	0.74
e40r0000	1.470	1.200	1.030	1.360	1.420	1.230	1.300	1.010	1.46
e40r1000	1.500	1.240	1.050	1.580	1.440	1.360	1.620	0.860	1.74
e40r5000	2.080	1.500	1.190	1.570	1.550	1.400	1.480	1.190	1.75
ecl32	F_M	106.4	55.50	**39.63**	F_M	**26.30**	**23.09**	F_M	-
epb3	5.670	6.740	5.800	11.07	F_M	9.060	11.09	F_M	0.98
fidap011	16.94	**9.240**	**5.390**	**4.070**	**3.480**	**2.860**	**2.920**	**2.700**	**6.27**
fidap035	1.130	1.190	1.140	2.110	1.770	1.520	2.090	1.180	0.96
fidapm11	F_M	F_N	F_N	F_N	F_N	F_N	F_N	F_N	-
lhr34c	11.68	7.240	4.480	F_M	F_M	4.63	F_M	F_M	2.61
lhr71c	22.97	14.13	9.210	10.82	F_M	8.780	9.900	F_M	2.61
memplus	**0.260**	0.560	0.540	1.400	1.260	0.870	1.460	0.830	0.48
nasasrb	F_M	F_N	**9.240**	**8.460**	**6.860**	**6.080**	**6.010**	**4.510**	-
onetone1	10.70	**6.940**	**4.520**	**5.580**	F_M	4.370	4.810	F_M	2.45
onetone2	3.550	3.270	2.620	4.270	F_M	3.290	3.780	F_M	1.35
pre2	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
raefsky3	**6.860**	**4.190**	**2.520**	**2.440**	**1.920**	**1.700**	**1.880**	**1.300**	5.28
raefsky4	28.11	**15.14**	**8.340**	**6.120**	**5.190**	**4.350**	**4.140**	**3.220**	**8.73**
rma10	**5.780**	4.200	3.010	3.700	3.310	3.090	3.380	2.230	2.59
tib	1.47	1.82	1.80	2.84	2.81	2.70	3.17	2.47	0.82
twotone	637.3	335.4	171.8	F_M	F_M	F_M	F_M	F_M	3.70
venkat50	**8.110**	5.820	4.290	5.580	4.990	4.350	4.730	3.530	2.30
wang3	36.87	20.27	**11.13**	**9.430**	**7.990**	**7.120**	**6.730**	**5.110**	7.24
wang4	23.68	13.55	**8.250**	**7.510**	F_M	**5.830**	**5.990**	**4.210**	5.64

Table 13: Parallel run times of SuperLU_MPI (in seconds). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination. The last column indicates the maximum speedup for each matrix.

	1	*2*	*4*	*6*	*8*	*10*	*12*	*16*
af23560	103.1	104.5	107.4	111.2	114.1	117.0	119.8	123.0
av41092	F_N	F_N	F_N	F_N	F_M	F_N	F_N	F_N
bayer01	**23.53**	**27.22**	**34.37**	**47.58**	F_M	**58.35**	**70.74**	F_M
bbmat	F_M	F_M	331.0	336.0	340.3	344.6	348.7	354.9
comp2c	**38.38**	**39.42**	**41.51**	**50.24**	**46.07**	**52.52**	**50.18**	**52.99**
cry10000	**5.523**	**6.255**	**7.591**	10.42	12.08	12.28	13.67	15.05
e40r0000	**19.19**	**20.09**	**21.79**	24.99	27.10	27.69	31.27	32.50
e40r1000	**19.57**	**20.48**	**22.19**	**24.06**	**25.72**	**26.99**	**29.01**	**31.57**
e40r5000	**23.94**	**24.86**	**26.59**	**28.92**	**30.14**	**31.49**	**33.46**	**36.04**
ecl32	F_M	397.5	404.1	412.5	F_M	422.5	432.0	F_M
epb3	**57.71**	**63.45**	**74.03**	89.38	F_M	105.1	122.4	F_M
fidap011	100.9	101.6	103.1	104.6	106.0	107.3	109.0	111.4
fidap035	13.83	15.02	17.20	19.94	21.61	23.48	26.42	28.77
fidapm11	F_N	F_N	F_N	F_N	F_N	F_N	F_N	F_N
lhr34c	42.50	44.77	49.16	F_M	F_M	71.38	F_M	F_M
lhr71c	**85.82**	**90.34**	**99.08**	**109.2**	F_M	**122.8**	**134.1**	F_M
memplus	**4.650**	**5.707**	7.669	11.14	12.23	14.20	16.89	18.89
nasasrb	F_M	F_N	216.7	221.6	226.1	230.7	235.6	243.8
onetone1	**35.56**	**38.08**	**42.65**	**49.81**	F_M	**56.63**	**64.69**	F_M
onetone2	**18.88**	**21.39**	**25.86**	**32.45**	F_M	**39.75**	**44.23**	F_M
pre2	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
raefsky3	**68.69**	**69.58**	71.32	73.16	74.77	76.41	78.30	81.32
raefsky4	**129.2**	**130.1**	**131.9**	**133.6**	**135.3**	**136.8**	**138.7**	**141.7**
rma10	**82.38**	**84.52**	88.63	93.38	97.31	101.4	105.5	112.2
tib	**10.17**	**11.58**	**14.22**	**17.91**	20.53	21.82	25.86	28.50
twotone	**133.4**	**142.3**	**158.7**	F_M	F_M	F_M	F_M	F_M
venkat50	**105.1**	**108.1**	113.9	120.4	125.3	130.8	136.7	145.7
wang3	**124.7**	**126.4**	**129.8**	**134.6**	**137.3**	**140.5**	**143.7**	**147.6**
wang4	97.81	99.65	103.1	109.2	F_M	114.1	120.9	123.7

Table 14: Parallel memory use by SuperLU_MPI (in Megabytes). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination.

	1	*2*	*4*	*6*	*8*	*10*	*12*	*16*	Max S
af23560	10.49	8.573	7.1162	6.646	6.285	6.117	5.952	5.789	1.81
av41092	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
bayer01	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
bbmat	**97.72**	76.96	62.39	55.59	55.16	58.58	58.43	59.85	1.67
comp2c	287.6	283.0	274.2	255.5	236.6	249.8	245.7	239.4	1.20
cry10000	0.502	0.316	**0.230**	**0.227**	**0.210**	**0.208**	**0.200**	**0.199**	2.50
e40r0000	2.645	1.775	1.429	1.333	1.252	1.187	1.163	1.146	2.31
e40r1000	6.711	4.283	3.480	3.165	2.837	2.792	2.740	2.490	2.70
e40r5000	395.7	353.1	353.2	327.2	324.1	323.5	320.1	323.2	1.24
ecl32	562.1	443.7	424.8	388.2	376.6	391.9	379.0	381.4	1.49
epb3	4.998	3.574	**2.332**	**1.897**	**1.742**	**1.577**	**1.592**	**1.433**	**3.49**
fidap011	**12.25**	9.335	8.262	7.618	7.342	6.949	7.059	6.447	1.90
fidap035	1.235	0.841	0.583	0.507	0.471	0.461	**0.439**	**0.430**	**2.87**
fidapm11	**15.10**	**11.50**	9.124	7.848	7.345	6.520	**6.082**	**5.801**	2.60
lhr34c	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
lhr71c	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
memplus	0.561	0.396	0.298	0.276	0.273	0.255	0.272	0.284	**2.20**
nasasrb	**25.04**	19.34	15.48	14.63	13.75	13.32	13.06	12.02	2.08
onetone1	112.8	101.8	76.83	69.70	70.54	70.30	64.90	63.79	1.77
onetone2	20.69	13.38	8.084	5.873	5.095	4.350	4.172	3.455	**5.99**
pre2	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
raefsky3	10.04	7.127	5.428	5.185	4.504	4.406	4.560	4.422	2.27
raefsky4	156.8	127.1	105.0	106.2	102.5	111.6	102.4	110.0	1.53
rma10	10.73	8.338	6.818	6.109	5.899	5.865	5.476	5.229	2.05
tib	1.749	1.135	0.873	0.823	0.770	0.820	0.753	0.765	2.32
twotone	724.0	584.2	543.0	490.2	471.5	530.3	491.1	467.8	1.55
venkat50	11.56	8.648	5.893	5.454	5.027	4.870	4.901	4.673	2.47
wang3	62.70	49.93	45.34	43.49	42.35	43.42	43.85	45.96	1.44
wang4	**16.26**	**12.93**	12.41	12.37	12.45	12.81	13.33	13.76	1.31

Table 15: Parallel run times for Spooles_MPI (in seconds). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination. The last column indicates the maximum speedup for each matrix.

	1	*2*	*4*	*6*	*8*	*10*	*12*	*16*
af23560	94.86	96.83	101.1	105.2	109.1	113.2	117.4	125.9
av41092	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
bayer01	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
bbmat	349.3	353.3	361.2	369.3	378.1	386.5	394.5	412.8
comp2c	164.2	168.4	177.8	185.9	194.6	203.7	212.5	230.5
cry10000	7.861	8.233	9.066	**9.782**	**10.62**	**11.47**	**12.28**	**13.98**
e40r0000	32.60	33.25	34.52	35.73	37.02	38.34	39.65	42.24
e40r1000	40.80	41.44	42.74	43.98	45.31	46.66	47.96	50.59
e40r5000	98.67	99.35	100.9	102.3	103.9	105.4	106.8	109.9
ecl32	386.6	391.1	400.7	411.0	421.5	432.2	442.6	463.5
epb3	86.06	87.95	91.91	95.77	99.83	103.9	108.0	116.1
fidap011	97.05	98.38	101.1	104.0	106.6	109.3	112.3	117.8
fidap035	20.09	20.79	22.07	23.41	24.85	26.25	27.64	30.50
fidapm11	107.6	109.4	113.1	117.2	121.1	125.1	129.1	136.4
lhr34c	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
lhr71c	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
memplus	8.568	9.041	9.948	10.88	11.84	12.82	13.80	15.79
nasasrb	231.5	234.2	239.9	245.7	251.7	257.6	263.5	275.4
onetone1	284.5	288.1	296.3	304.1	312.2	319.9	327.8	344.0
onetone2	134.6	136.9	141.3	146.2	150.9	155.0	159.8	169.1
pre2	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
raefsky3	96.32	97.42	99.66	102.2	104.3	106.7	108.8	113.9
raefsky4	275.0	277.7	283.3	289.1	295.0	300.8	306.5	318.7
rma10	125.9	127.4	130.8	134.0	137.4	140.7	143.9	150.7
tib	20.94	21.76	23.48	25.28	26.90	28.60	30.39	33.97
twotone	767.9	775.5	792.4	809.6	825.4	843.1	859.2	891.2
venkat50	143.2	145.3	149.7	153.7	158.1	162.6	167.0	175.8
wang3	176.9	179.1	184.0	188.8	193.7	198.7	203.7	213.3
wang4	91.23	93.02	96.53	99.94	103.6	106.9	110.7	118.4

Table 16: Parallel memory use by Spooles_MPI (in Megabytes). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination.

	1	*2*	*4*	*6*	*8*	*10*	*12*	*16*	Max S
af23560	81.86	75.29	47.12	37.92	33.15	29.18	27.16	26.06	3.14
av41092	327.6	305.2	173.1	124.4	102.0	89.90	77.53	68.00	**4.82**
bayer01	90.71	80.85	62.73	100.3	90.54	84.85	81.88	80.09	1.45
bbmat	621.3	602.0	343.0	245.5	200.9	178.5	141.8	141.1	4.40
comp2c	**12.34**	**9.724**	**7.426**	11.29	10.30	9.698	13.93	10.99	1.27
cry10000	5.200	5.575	4.850	5.389	5.152	4.943	5.486	6.092	1.07
e40r0000	24.85	20.89	11.08	12.00	10.07	8.852	9.210	8.863	2.81
e40r1000	24.57	20.83	11.05	11.93	9.877	8.791	9.188	8.826	2.78
e40r5000	24.45	20.64	10.96	12.00	9.985	8.887	9.216	8.797	2.78
ecl32	F_M	2293.	1257.	836.5	684.4	581.1	468.3	444.6	-
epb3	114.1	108.3	81.78	126.1	114.5	105.9	101.8	101.9	1.39
fidap011	124.0	114.2	65.81	45.04	38.29	33.84	28.25	26.99	4.59
fidap035	11.53	11.41	8.86	13.75	12.78	11.92	12.51	12.31	1.30
fidapm11	348.6	333.2	183.7	123.3	100.3	89.01	69.02	66.63	**5.23**
lhr34c	46.06	43.84	32.57	38.95	35.47	33.56	33.76	31.41	1.47
lhr71c	130.2	122.0	88.10	127.3	112.2	104.5	103.5	96.64	1.35
memplus	8.498	7.864	5.274	7.047	6.359	6.136	6.604	6.949	1.61
nasasrb	338.8	325.9	195.5	160.8	137.5	125.0	105.2	104.1	**3.25**
onetone1	72.80	71.21	45.61	35.03	31.31	29.04	27.38	26.91	2.71
onetone2	14.20	15.36	13.52	15.22	14.42	14.13	15.54	16.09	1.05
pre2	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
raefsky3	78.54	76.09	46.89	37.05	32.54	29.72	25.98	24.10	3.26
raefsky4	313.2	302.2	171.0	113.4	92.93	81.26	64.00	62.38	5.02
rma10	131.1	116.26	73.69	77.39	66.48	59.43	54.81	48.24	2.72
tib	164.7	160.2	88.27	63.35	50.04	41.70	36.73	32.51	**5.08**
twotone	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
venkat50	196.7	184.3	122.1	134.6	117.6	108.0	98.79	92.50	2.13
wang3	699.9	694.2	390.1	258.4	210.3	181.3	145.3	144.0	4.86
wang4	599.3	594.2	334.8	227.3	184.2	158.6	128.0	124.3	4.82

Table 17: Parallel run times S+ (in seconds). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination. The last column indicates the maximum speedup for each matrix.

	1	*2*	*4*	*6*	*8*	*10*	*12*	*16*
af23560	207.0	212.0	222.2	232.1	242.1	252.2	262.2	282.6
av41092	419.5	430.1	451.6	**473.1**	**494.5**	**515.8**	**536.5**	**579.9**
bayer01	247.7	276.3	333.7	381.3	438.4	495.4	550.0	667.0
bbmat	796.6	815.1	852.3	889.0	926.1	963.0	999.9	1075.
comp2c	56.58	61.15	70.35	79.33	88.41	97.54	106.5	124.9
cry10000	23.30	24.45	26.76	28.89	31.19	33.46	35.63	40.38
e40r0000	92.45	93.71	96.27	98.69	101.2	103.8	106.2	111.5
e40r1000	92.27	93.53	96.07	98.51	101.1	103.6	106.0	111.3
e40r5000	92.01	93.27	95.81	98.26	100.8	103.3	105.7	111.0
ecl32	F_M	1531.	1577.	1622.	1668.	1715.	1761.	1854.
epb3	313.8	335.5	379.0	421.8	465.0	508.2	551.0	637.9
fidap011	228.5	231.0	236.2	241.1	246.1	251.3	256.2	266.5
fidap035	56.59	60.08	67.08	73.96	80.94	87.89	94.78	108.9
fidapm11	431.3	437.8	450.9	463.7	476.7	489.7	502.4	528.9
lhr34c	161.2	170.3	188.5	206.5	224.6	242.7	260.5	297.1
lhr71c	389.1	418.0	475.8	532.7	590.2	647.8	705.2	820.6
memplus	37.37	39.70	44.40	48.95	53.61	58.28	62.77	72.36
nasasrb	625.4	638.3	664.2	690.0	715.8	741.8	767.4	819.6
onetone1	125.8	130.1	138.8	147.1	155.6	164.1	172.4	189.7
onetone2	50.11	53.83	61.28	68.39	75.68	**83.02**	90.11	**105.0**
pre2	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
raefsky3	199.8	202.7	208.6	214.3	220.2	226.0	231.8	243.7
raefsky4	376.8	381.5	391.0	400.1	409.5	418.9	428.1	447.5
rma10	311.5	324.0	349.2	374.1	399.2	424.3	449.3	499.8
tib	213.7	218.0	226.7	235.0	243.6	252.1	260.6	278.3
twotone	F_M	F_M	F_M	F_M	F_M	F_M	F_M	F_M
venkat50	470.3	495.7	546.5	597.1	647.8	698.4	749.1	850.8
wang3	602.1	610.3	627.0	643.2	659.6	676.1	692.3	725.8
wang4	549.1	556.5	571.6	586.2	601.1	616.0	630.6	660.9

Table 18: Parallel memory use by S+ (in Megabytes), F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination.

	1	*2*	*4*	*6*	*8*	*10*	*12*	*16*	Max S
af23560	**8.930**	**5.439**	**4.019**	**3.770**	**3.503**	**3.538**	**3.314**	**3.278**	2.72
av41092	**30.59**	**19.52**	**12.29**	**12.59**	**8.648**	**9.738**	**11.29**	**10.33**	3.54
bayer01	**2.262**	**1.265**	**1.049**	**1.082**	**1.084**	**1.116**	**1.127**	**1.143**	**2.16**
bbmat	113.0	**61.76**	**39.65**	**30.55**	**28.13**	**24.56**	**25.46**	**21.08**	**5.36**
comp2c	29.32	17.78	15.56	13.07	12.50	13.74	11.95	12.55	2.45
cry10000	**0.362**	**0.244**	0.253	0.256	0.254	0.256	0.258	0.260	1.48
e40r0000	**1.184**	**0.738**	**0.538**	**0.539**	**0.549**	**0.572**	**0.576**	**0.586**	2.20
e40r1000	**1.188**	**0.742**	**0.539**	**0.535**	**0.550**	**0.575**	**0.577**	**0.582**	2.22
e40r5000	**1.186**	**0.748**	**0.537**	**0.537**	**0.551**	**0.574**	**0.579**	**0.583**	2.21
ecl32	**144.9**	**80.68**	**54.73**	44.68	**42.97**	36.83	35.33	**31.84**	**4.55**
epb3	5.689	**3.490**	2.409	2.064	2.018	1.922	1.940	1.735	3.28
fidap011	18.68	10.50	10.09	10.50	8.079	8.428	8.800	7.651	2.44
fidap035	**0.820**	**0.444**	**0.448**	**0.447**	**0.456**	**0.454**	0.460	0.473	1.83
fidapm11	25.26	15.95	11.56	11.31	10.48	10.29	9.688	9.573	2.64
lhr34c	**5.526**	**2.977**	**1.886**	**1.602**	**1.303**	**1.199**	**1.087**	**1.093**	**5.08**
lhr71c	**11.71**	**7.242**	**3.814**	**3.077**	**2.643**	**2.188**	**2.015**	**1.757**	**6.66**
memplus	0.320	**0.175**	**0.198**	**0.171**	**0.170**	**0.170**	**0.170**	**0.170**	1.88
nasasrb	26.79	**18.33**	14.90	14.55	13.28	13.23	13.08	12.61	2.13
onetone1	**9.998**	7.498	5.538	6.421	**5.920**	**6.320**	**5.649**	**5.871**	1.77
onetone2	**2.815**	**1.882**	**1.575**	**1.504**	**1.523**	**1.544**	**1.583**	**1.632**	1.87
pre2	F_C	F_M	F_M	F_M	F_M	F_M	F_M	F_M	-
raefsky3	8.754	5.878	3.625	3.454	3.647	3.069	2.967	2.753	3.18
raefsky4	**27.50**	21.57	13.17	11.39	10.77	10.83	9.082	9.156	3.03
rma10	6.624	**3.887**	**2.645**	**2.304**	**1.959**	**1.994**	**1.951**	**1.964**	**3.40**
tib	**0.618**	**0.628**	**0.635**	**0.636**	**0.635**	**0.642**	**0.653**	**0.654**	0.98
twotone	**124.7**	**86.40**	**47.90**	**36.21**	**33.35**	**36.38**	**36.11**	**35.70**	**3.74**
venkat50	9.432	**5.536**	**3.651**	**2.850**	**2.745**	**2.315**	**2.495**	**2.326**	**4.07**
wang3	**32.33**	**19.50**	11.17	10.46	10.26	8.875	9.121	7.822	4.13
wang4	25.66	15.66	10.03	8.579	**7.673**	7.482	7.284	6.669	3.85

Table 19: Parallel run times for MUMPS (in seconds). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination. The last column indicates the maximum speedup for each matrix.

	1	*2*	*4*	*6*	*8*	*10*	*12*	*16*
af23560	104.1	122.0	149.3	173.9	199.1	222.8	246.9	295.7
av41092	**196.5**	**271.8**	**373.5**	495.1	568.4	643.6	736.7	903.5
bayer01	41.85	46.52	55.34	64.87	**73.97**	83.08	92.18	**110.4**
bbmat	580.8	730.9	776.8	863.4	876.8	1008.	1189.	1400.
comp2c	211.6	309.9	429.2	496.4	603.5	694.8	794.5	949.7
cry10000	7.390	9.068	11.69	14.01	16.34	18.66	20.99	25.63
e40r0000	25.17	31.81	40.44	48.17	56.34	64.74	71.11	83.84
e40r1000	25.17	31.81	40.44	48.17	56.34	64.74	71.11	83.84
e40r5000	25.17	31.81	40.44	48.17	56.34	64.74	71.11	83.84
ecl32	612.1	1029.	1058.	1119.	1125.	1136.	1181.	1353.
epb3	96.94	119.8	146.9	194.4	225.2	257.8	282.0	347.5
fidap011	167.4	223.7	284.7	369.6	444.2	506.3	581.2	711.8
fidap035	17.80	19.40	22.55	25.70	28.85	32.00	35.15	41.44
fidapm11	196.9	309.3	313.6	367.0	431.6	499.5	562.8	691.8
lhr34c	**75.19**	**87.10**	**103.1**	**118.3**	**131.2**	**145.1**	**162.5**	**189.5**
lhr71c	151.8	160.8	193.4	207.6	**219.8**	259.8	284.1	**322.8**
memplus	5.026	6.652	9.760	12.80	15.84	18.88	21.92	28.01
nasasrb	295.9	327.4	403.9	465.7	518.3	590.1	646.7	772.5
onetone1	78.62	144.8	222.7	289.4	367.8	444.5	500.5	635.2
onetone2	36.45	53.95	84.04	108.0	133.0	156.6	183.5	230.0
pre2	F_C	F_M	F_M	F_M	F_M	F_M	F_M	F_M
raefsky3	113.8	156.4	189.6	236.7	279.0	312.7	350.0	420.1
raefsky4	211.1	311.0	390.5	463.0	568.0	582.7	694.1	878.6
rma10	115.9	127.5	145.5	161.5	178.3	196.0	211.6	241.9
tib	12.64	14.13	17.10	20.07	23.05	26.02	29.0	34.92
twotone	442.3	883.7	877.3	900.9	944.5	890.1	988.4	1249.
venkat50	155.3	179.8	223.2	261.0	295.7	325.6	354.8	426.4
wang3	227.2	431.4	483.8	445.6	453.2	473.4	548.4	689.3
wang4	186.6	352.9	408.4	360.3	375.7	428.6	559.7	714.2

Table 20: Parallel memory use by MUMPS (in Megabytes). F_N = failed numerically, F_M = failed due to running out of memory, F_C = abnormal termination.

4 Results of Parallel Runs

Tables 9 through 20 give the parallel run times and memory usage for all the parallel versions of the softwares on 1 to 16 processors. Note that the multithreaded codes (Spooles_MT and SuperLU_MT) could use only 2 Gbytes of memory, even on multiple CPU's. All distributed memory codes could use 2 Gbytes of RAM on each MPI process. The memory usage tables for these codes are constructed by adding the memory used by each process during factorization.

In Tables 9 through 20, the best entry for a given matrix on a given number of processors across all parallel codes is highlighted in boldface. For example, on 4 CPUs, S+ factored the matrix "comp2c" in the smallest time, while SuperLU_MPI used the smallest amount of memory. The last column in a table showing parallel factorization time of a particular package gives the highest speedup achieved for each matrix by that package. The entries in boldface in this column indicate the highest speedup achieved for a matrix among all the packages. For example, the maximum speedup obtained by Spooles_MT on "af23560" is 2.07 (which happens to be at 12 CPUs). However, the highest speedup obtained on "af23560" is 5.94 by SuperLU_MT (which happens to be 10 CPUs). Comparing the maximum speedup column among different parallel factorization-time tables can provide a rough indication of the scalability of different packages. However, if serial factorization times for a matrix vary significantly among different packages, then comparing their relative scalability on that matrix is not very useful.

A glance through the parallel factorization time tables reveals that MUMPS yields the fastest factorization most often, followed by SuperLU_MPI. Since MUMPS performs partial pivoting and SuperLU_MPI doesn't, it probably won't be be unfair to say that MUMPS was observed to be the fastest parallel sparse LU factorization code in our experimental setup on the suite of matrices chosen in this report. In terms of scalability, SuperLU_MT has the most number of boldface entries in the maximum speedup column, followed closely by MUMPS. However, even with an advantage in speedup, SuperLU_MT was not able to make up for the slow serial factorization.

A comparison of the parallel memory use tables reveals that Spooles_MT is most memory efficient followed by SuperLU_MPI, at least in terms of the memory allocated.

5 Concluding Remarks

In this paper, we have reported the results of an empirical comparison of general sparse LU factorization by a set of commonly used software packages, namely, MUMPS, S+, SPOOLES, SuperLU, UMFPACK, and WSMP. Our results in Section 3 indicate WSMP [7] to have the fastest and most memory efficient serial implementation. Among parallel sparse factorization codes, the experimenatl results in Section 4 show that MUMPS [1] has the best overall performance.

References

[1] Patrick R. Amestoy, Iain S. Duff, and J. Y. L'Execellent. MUMPS: Multifrontal massively parallel solver version 2.0. Technical Report PA–98/02, CERFACS, Toulouse Cedex, France, 1998.

[2] Cleve Ashcraft and Roger G. Grimes. SPOOLES: An object-oriented sparse matrix library. In *Proceedings of the Nith SIAM Conference on Parallel Processing for Scientific Computing*, March 1999.

[3] Timothy A. Davis and Iain S. Duff. An unsymmetric-pattern multifrontal method for sparse LU factorization. *SIAM Journal on Matrix Analysis and Applications*, 18(1):140–158, January 1997.

[4] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W.-H. Liu. A supernodal approach to sparse partial pivoting. Technical Report UCB//CSD-95-883, Computer Science Division, U.C. Berkeley, July 1995.

[5] James W. Demmel, John R. Gilbert, and Xiaoye S. Li. An asynchronous parallel supernodal algorithm for sparse Gaussian elimination. Technical Report UCB//CSD-97-943, Computer Science Division, U.C. Berkeley, February 1997.

[6] Iain S. Duff and Jacko Koster. The design and use of algorithms for permuting large entries to the diagonal of sparse matrices. Technical Report RAL-TR-97-059, Rutherford Appleton Laboratory, November 1997.

[7] Anshul Gupta. WSMP: Watson sparse matrix package. Technical Report (In preparation), IBM T. J. Watson Research Center, Yorktown Heights, NY, 2000.

[8] Anshul Gupta and Lexing Ying. Algorithms for finding maximum matchings in bipartite graphs. Technical Report RC 21576 (97320), IBM T. J. Watson Research Center, Yorktown Heights, NY, October 19, 1999.

[9] Xiaoye S. Li and James W. Demmel. Making sparse Gaussian elimination scalable by static pivoting. In *Supercomputing '98 Proceedings*, 1998.

[10] Xiaoye S. Li and James W. Demmel. A scalable sparse direct solver using static pivoting. In *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, 1999.

[11] Kai Shen, Tao Yang, and Xiangmin Jiao. S+: Efficient 2D sparse LU factorization on parallel machines. *SIAM Journal on Matrix Analysis and Applications*, To be published.