(2)

AD-A265 891

IMENTATION PAGE

Form Approved
OMB No. 0704-0188

2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
April 1993	Professional paper

4. TITLE AND SUBTITLE
NEC2, NEC3, AND NEC4 ON A CONVEX MINI-SUPERCOMPUTER

5. FUNDING NUMBERS
In house funding

6. AUTHOR(S)
L. Koyama

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152-5001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152-5001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A methodology was desired for optimizing the Numerical Electromagnetics Code (NEC) on a given platform. The platform chosen was the Convex mini-supercomputer. The matrix fill and factor times were the guages of optimizing for speed. The software tool for choosing where to optimize was the profiler that comes with the FORTRAN compiler.

NEC2, NEC3, and NEC4 were evaluated. The test cases were models of 44, 300, 722, and 2286 segments. Three levels of built-in compiler optimizations were used. Additional optimizations were sought. The greatest speedup in runtime came with the use of LINPACK library routines specifically optimized of the Convex.

93 6 10 123

93-13691

Published in *9th Annual Review of Progress in Applied Computational Electromagnetics*, Mar 22-26, 1993, pp 45-52.

14. SUBJECT TERMS			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

21a. NAME OF RESPONSIBLE INDIVIDUAL	21b. TELEPHONE *(include Area Code)*	21c. OFFICE SYMBOL
L. Koyama	(619) 553-3784	Code 824

NEC2, NEC3, AND NEC4 ON A CONVEX MINI-SUPERCOMPUTER

Lance Koyama
NCCOSC RDTE DIV
Code 824
SAN DIEGO, CA 92152-7304

Abstract

A methodology was desired for optimizing the Numerical Electromagnetics Code (NEC) on a given platform. The platform chosen was the Convex mini-supercomputer. The matrix fill and factor times were the gauges of optimizing for speed. The software tool for choosing where to optimize was the profiler that comes with the FORTRAN compiler.

NEC2, NEC3, and NEC4 were evaluated. The test cases were models of 44, 300, 722, and 2286 segments. Three levels of built-in compiler optimizations were used. Additional optimizations were sought. The greatest speedup in runtime came with the use of LINPACK library routines specifically optimized for the Convex.

INTRODUCTION

This study gives a methodology for optimizing the NEC codes (or any method of moments code) for a given platform, in this case, a Convex mini-supercomputer.

The Convex Computer Corporation mini-supercomputers have become very popular because of their high power for the dollar. The model used for this study was the Convex C240 which is commonly classified as a mini-supercomputer. Its vector architecture makes it a supercomputer and it is smaller than a Cray, making it a mini. Its cogent features are as follows:

- 4 processors - 50 MegaFLOPS each
- Each processor includes scalar and vector processing units
- Peak performance - 200 MegaFLOPS
- LINPACK1000 benchmark: 162 MegaFLOPS
  (Cray Y-MP): 305 MegaFLOPS
- Whetstone benchmark: 38 MIPS
  (Cray Y-MP): 26 MIPS
- MULTIunits benchmark: 4900
  (Cray Y-MP): 6000

Each processor has

- 8 vector registers of 128 elements each
- Each element (word) consists of 64 bits
- There are 3 independent functional unit controllers:
  - Load and Store
  - Multiply and Divide
  - Add and Logical

SCOPE

NEC runs were made on various combinations of the following parameters:

Codes

Name	Number of Lines	Number of Routines
NEC2	8,734	81
NEC3	9,780	99
NEC4	16,039	207

NEC2 was chosen for its complete documentation; NEC4, for being the latest and greatest and NEC3 to round out the family.

Compiler Optimizations - None, Scalar, Vector, Parallel -

There are three types of automatic optimizations that come with the FORTRAN compiler. The scalar optimization performs a great many types of both machine dependent and machine independent optimizations on the scalar level. The vector optimization seeks loops that are actually dealing with arrays. As much as possible, the entire loop is converted to vector operations. The parallel optimization operates only within individual routines. It tries to spread the processing among the four processors if it would be more efficient.

In the following discussion, the optimizations are labeled as follows.

Optimization	Types
0	None
1	Scalar
2	Scalar + Vector
3	Scalar + Vector + Parallel

Models - 44 segment, 300 segment, 722 segment, 2286 segment -

The 44 segment model is a one wavelength loop. The 300 segment model is a monopole on a ground plane. The 722 segment model is the US Navy's Spruance (DD-963) class destroyer segmented for up to 6 MHz problems. The 2286 segment model is the same ship segmented for 30 MHz problems.

Information Gathered

For each run, the following information was gathered:

- Matrix fill, matrix factor, and total run-time
- A profile of the run listing each routine and operation used and for each:
    - Percentage of total run-time used in calls to the routine or operation
    - The number of calls to the routine or operation
    - The time used in a call to the routine or operation

For some of the runs, some additional information was gathered: the percent used of all the processors, the amount of memory used, the physical reads and writes, the number of page faults, and the number of page faults paged out to disk.

Manual Optimization

Looking at the profiles of the runs, routines were chosen to be optimized beyond the automatic optimizations of the quite intelligent compiler.

RESULTS

Verification

The impedance of an antenna on each of the models was used to verify that a run was valid.

Profiles

To gauge the performance of each run, the profiler that comes with the FORTRAN compiler was used. There is some overhead in its use as it performs its counts and timings as seen in the examples below for a 722 segment model.

	Fill	Factor	Total
NEC3, optim.2, w/o profiler	106.943	37.999	148.343
with profiler	120.596	32.316	157.349
NEC4, optim.3, w/o profiler	128.441	49.368	187.582
with profiler	155.841	45.500	213.442

All times in the following data and discussion presume the use of the profiler. You will see in the following profiles an item called "mcount". This is one of the profiler overhead items.

The following series of profiles shows the differences between NEC2, NEC3, and NEC4 for a 722 segment model using compiler optimization 2 in all cases. The routines or functions that take up more than 5% of the total runtime are shown.

NEC2		Fill	Factor	Total
		177.541	32.436	213.422
%time	cumsecs	#call	ms/call	name
15.7	34.84	521284	0.07	_efld_
15.4	69.07	1021498	0.03	_eksc_
15.2	102.82	1	33750.00	_factr_
14.7	135.38	4901342	0.01	_gf_
9.8	157.13		21750ms	mcount
9.1	177.24	1021498	0.02	_intx_
7.5	194.00	2042996	0.01	_gx_
6.5	208.49	722	20.07	_cmww_

NEC3		Fill	Factor	Total
		120.596	32.316	157.349
%time	cumsecs	#call	ms/call	name
20.5	33.62	1	33620.00	_factr_
17.1	61.58	887915	0.03	_eksclr_
15.7	87.28	521284	0.05	_efld_
11.5	106.19		18910ms	mcount
10.6	123.57	722	24.07	_cmww_
5.5	132.50	1351794	0.01	_mth$c_exp

```
NEC4                Fill     Factor      Total
                 146.313     32.366    190.199

%time   cumsecs     #call     ms/call   name
 17.1     33.82       286      118.25   _factr_
 14.9     63.24    887913        0.03   _eksclr_
 13.5     89.95                26710ms  mcount
 10.8    111.23   1042568        0.02   _efldsg_
  8.6    128.22       722       23.53   _cmww_
```

The following show the differences in profiles as the size of the model changes. (NEC4 is used with optimization 3).

```
300 segments        Fill     Factor      Total
                  23.873      4.436     30.287

%time   cumsecs     #call     ms/call   name
 21.0      6.00    177310        0.03   _eksclr_
 20.9     11.96                5965ms   mcount
 10.4     14.95    180000        0.02   _efldsg_
  9.0     17.52       300        8.60   _cmww_
  6.3     19.32         1     1800.00   _factr_
  5.8     20.98    544795        0.00   _mth$c_div
```

```
722 segments        Fill     Factor      Total
                  155.841     45.500    213.442

%time   cumsecs     #call     ms/call   name
 19.8     38.00                38000ms  mcount
 15.5     67.62    887913        0.03   _eksclr_
  9.7     86.29   1042568        0.02   _efldsg_
  9.2    103.96       286       61.78   _factr_
  9.2    121.52       722       24.32   _cmww_
  5.0    131.03   3178604        0.00   _mth$c_div
```

```
2286 segments       Fill     Factor      Total
                 1437.447   5678.125   7220.752

%time   cumsecs     #call     ms/call   name
 52.3   1747.24       756     2311.16   _factr_
 10.0   2082.34                335105ms  mcount
  9.8   2410.93   9918973        0.03   _eksclr_
  5.6   2596.86  10451592        0.02   _efldsg_
  5.4   2776.95      2286       78.78   _cmww_
```

```
2286 segments   LINPACK routines
                    Fill     Factor      Total
                 1451.863    700.771   2304.245

%time   cumsecs     #call     ms/call   name
 18.4    334.04                334040ms  mcount
 18.3    665.41   9918973        0.03   _eksclr_
 10.6    858.13  10451592        0.02   _efldsg_
 10.4   1046.93                188800ms  _cgefa_
  9.7   1223.06      2286       77.05   _cmww_
  5.3   1319.97  10451592        0.01   _ekscsz_
  5.0   1410.89  31597174        0.00   _mth$c_div
```

The last profile was of a run using LINPACK routines, discussed next.

Manual Optimization

A widely available set of routines for solving linear equations, called LINPACK, was available specifically optimized for the Convex hardware. Two routines were chosen to replace the matrix factor and solve portions of the NEC codes.

Function	NEC routine	LINPACK routine
Factor matrix	factr	cgefa
Solve matrix	solve	cgesl

In both cases, because of the way NEC stores matrices, the interaction matrix had to be transposed before and after the LINPACK routines were used.

Next, routines high in the profile list were sought that could benefit from manipulation so that the compiler could vectorize them. In NEC2: *efld*, *eksc*, *gf*, *intx*, and *test* all had no loops. In NEC4: *eksclr*, *efldsg*, and *ekscsz* all had no loops. In both: *cmww* was already automatically 70% vectorized by the compiler. All other routines consumed less than 5% of the total runtime. It was not considered worthwhile to continue the optimization effort.

Runtimes

The following lists the impedances and runtimes of the significant runs.

				Impedance		Times (seconds)		
Segments	NEC	Optim	LinPack	R	X	FILL	FACTOR	TOTAL
44	2	2		100.6	-139.1	0.36	0.02	0.47
44	4	3		100.3	-140.6	0.47	0.05	0.69
44	4	3	yes	100.3	-140.6	0.47	0.01	0.66
300	2	0		232.3	-36.4	9.54	19.28	29.86
300	2	1		232.3	-36.4	9.20	12.97	22.98
300	2	2		232.3	-36.4	10.23	2.36	13.30
300	2	3		232.3	-36.4	10.63	4.39	15.80
300	4	3		240.3	-37.1	23.87	4.44	30.29
300	4	3	yes	240.3	-37.1	24.12	1.71	27.86
722	2	0		20.3	-0.5	168.27	268.14	441.56
722	2	1		20.3	-0.5	176.80	180.83	361.73
722	2	2		20.3	-0.4	177.54	32.44	213.42
722	2	2	yes	20.3	-0.4	176.06	22.08	202.21
722	2	3		20.3	-0.4	179.58	45.46	228.81
722	3	2		19.7	-2	120.60	32.32	157.35
722	4	2		23	0.5	146.31	32.37	190.20
722	4	3		23	0.5	155.84	45.50	213.44
722	4	3	yes	23	0.5	157.37	22.18	193.51
2286	4	3		34	20.8	1437.45	5678.13	7220.75
2286	4	3	yes	34	20.8	1451.86	700.77	2304.25

In matrix format

FILL RUNTIMES (sec)

Segments	NEC	Optimization					
		0	1	2	LinP	3	LinP
44	2			0.36			
	4					0.47	0.47
300	2	10	9	10		11	
	4					24	24
722	2	168	177	178	176	180	
	3			121			
	4				146	156	157
2286	4					1437	1452

FACTOR RUNTIMES (sec)

Segments	NEC	Optimization					
		0	1	2	LinP	3	LinP
44	2			0.02			
	4					0.05	0.01
300	2	19	13	2		4	
	4					4	2
722	2	268	181	32	22	45	
	3			32			
	4			32		46	22
2286	4					5678	701

TOTAL RUNTIMES (sec)

Segments	NEC	Optimization					
		0	1	2	LinP	3	LinP
44	2			0.47			
	4					0.69	0.66
300	2	30	23	13		16	
	4					30	28
722	2	442	362	213	202	229	
	3			157			
	4			190		213	194
2286	4					7221	2304

Figures 1, 2, 3, 4, and 5 graphically show the trends in the data.

CONCLUSIONS

Just as important as the speed of the machine is the way the software utilizes its resources. For the case of the Convex computer used in this study, the automatic optimizations supplied with its FORTRAN compiler cut the time for a NEC run in half for a 722 segment model.

Library routines optimized for a machine's hardware should be used whenever possible to replace existing code. Two routines especially appropriate for a method of moments code are the matrix factor and matrix solve routines from the LINPACK library. These had been optimized for the Convex hardware. For a 2,286 segment model, they cut the factor time by a factor of 8 resulting in an overall runtime improvement of a factor of 3.

REFERENCES

1. G. J. Burke and A. J. Poggio, *Numerical Electromagnetics Code (NEC) - Method of Moments*, Lawrence Livermore National Laboratory, Report UCID-18834, January 1981

2. A. Ralston, *A First Course in Numerical Analysis*, McGraw-Hill Book Company, 1965

3. "VECLIB Programmer's Reference", CONVEX Computer Corporation, August 1991

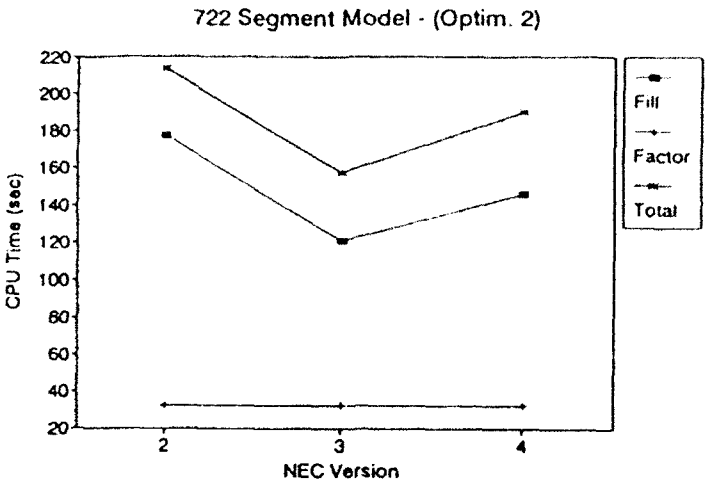

Figure 1. Runtime Variation with NEC Version Number

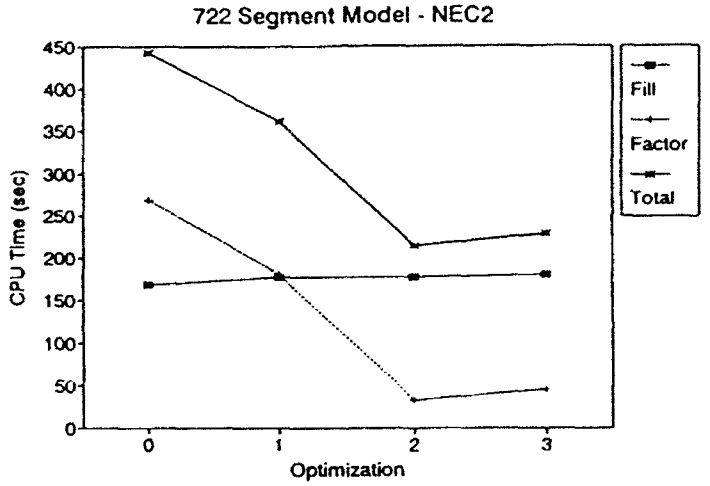

Figure 2. Runtime Variation with Optimization Level

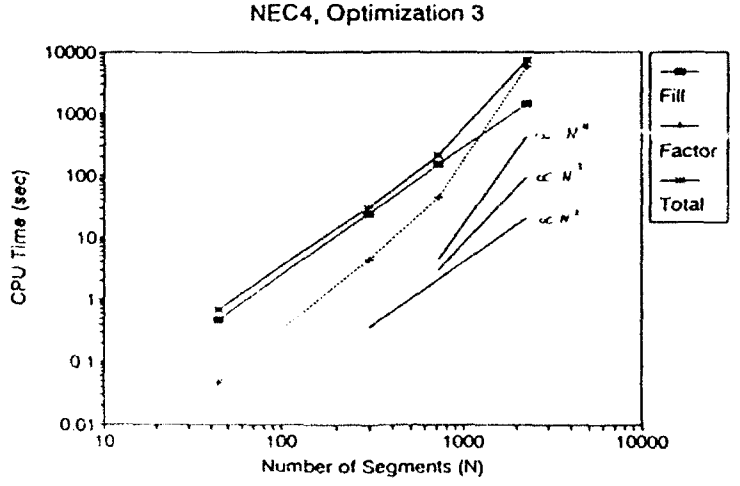

Figure 3. Runtime Variation with Number of Segments

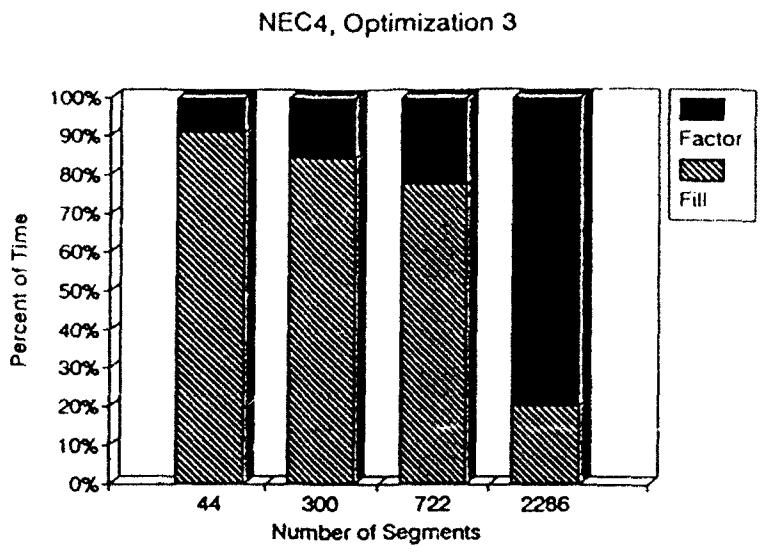

Figure 4. Percentage of Time in Fill and Factor for Different Model Sizes

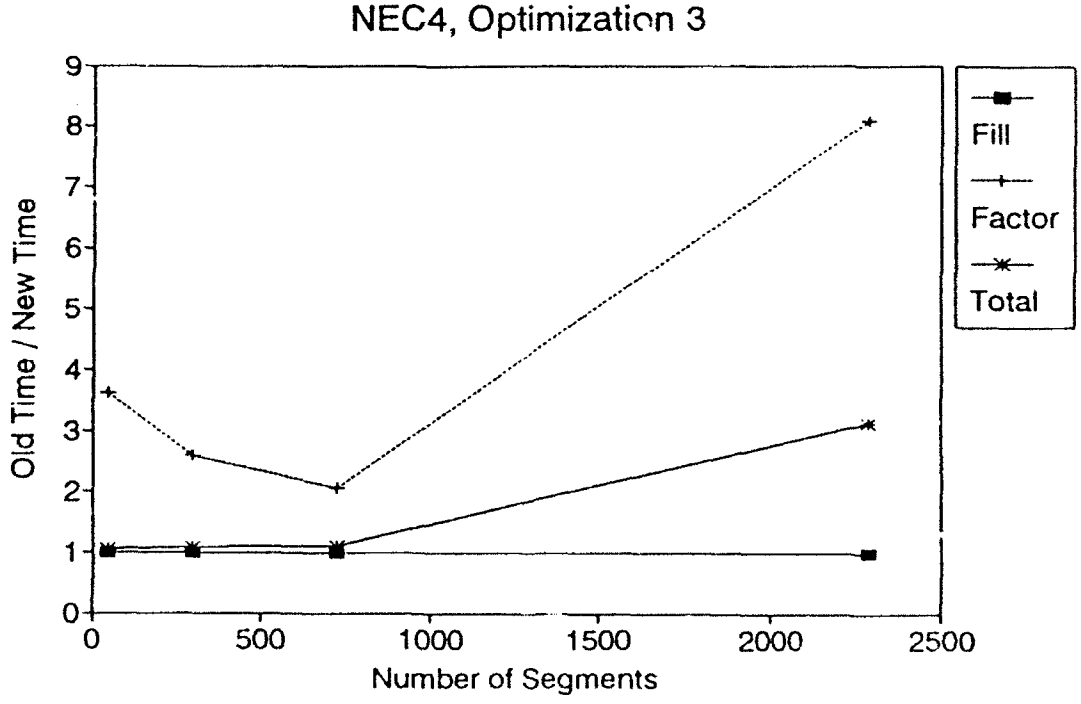

Figure 5. Improvement by Using LINPACK Routines