

KNITRO Performance on Large Nonlinear Problems

Todd Plantenga, Richard A. Waltz (August 2006)

This study examines the performance of KNITRO 5 on large nonlinear optimization problems, and describes simple tuning techniques for improving performance. Application problems come from the Constrained Optimization Problem Set (COPS) [1]. Results show that KNITRO can satisfactorily solve problems with over 100,000 variables and constraints on a standard workstation.

All tests were made with the 32-bit Linux version of KNITRO 5.0.3 running on a 3.0 GHz Pentium 4 workstation with 1 Gbyte physical memory. Test problems in AMPL [2] format come from the COPS 3.0 test set, and may be downloaded from

<http://www-unix.mcs.anl.gov/~more/cops/>

or by request from Ziena. COPS problems were executed as AMPL models, and solved by KNITRO using default options except where noted. Only the COPS sizing parameter was modified in the AMPL models to vary problem size.

Problem “gasoil”. The COPS problem determines the reaction coefficients for the catalytic cracking of gas oil into gas and other byproducts. The nonlinear programming problem (NLP) minimizes a quadratic objective subject to a large number of nonlinear equality constraints plus some linear constraints. A more accurate physical model is obtained by increasing the collocation interval n_h , which increases the size of the optimization problem.

The table below shows KNITRO performance for increasing problem size. KNITRO default options were used, which means the Interior-Point/Direct algorithm (IP-direct) was automatically chosen. Results show the number of function evaluations (*num FC*) and CPU time (*CPU sec*) necessary to solve to default tolerances. CPU time includes time spent in AMPL evaluating the functions and their partial derivatives; in this problem, it is less than 10% of total time. Execution time jumps significantly for the largest version of the problem because memory requirements exceed RAM capacity of the test machine.

n_h	variables	constraints	IP-direct (alg=1)	
			num FC	CPU sec
100	1,001	998	19	0.2
500	5,001	4,998	24	1.2
1,000	10,001	9,998	21	3.2
5,000	50,001	49,998	25	54.4
10,000	100,001	99,998	44	211.0
25,000	250,001	249,998	32	1730.0

Problem “robot”. The COPS problem computes control parameters for a robotic arm that minimizes the time required to travel between two points. The NLP minimizes a simple objective subject to nonlinear and linear equality constraints, plus bound constraints on variables. A more accurate physical solution is obtained by increasing the number of time step intervals n_h , which increases the size of the optimization problem.

The table below shows KNITRO performance for increasing problem size. Again, KNITRO default options were used, except in this case the algorithm was specified as either Interior-Point/CG (IP-CG) or Active Set. As the table shows, Interior-Point/CG is the best performing algorithm in terms of CPU time (*CPU sec*), while Active-Set seems best in terms of function evaluations (*num FC*). CPU time includes time spent in AMPL evaluating the functions and their partial derivatives; in this problem, it is less than 10% of total time.

n_h	<i>variables</i>	<i>constraints</i>	IP-CG (alg=2)		Active-Set (alg=3)	
			<i>num FC</i>	<i>CPU sec</i>	<i>num FC</i>	<i>CPU sec</i>
400	3,598	2,400	18	1.0	17	2.5
800	7,198	4,800	22	2.2	13	16.0
1,000	8,998	6,000	21	3.1	13	25.6
5,000	44,998	30,000	28	17.7	<i>not attempted</i>	
10,000	89,998	60,000	26	43.8	<i>not attempted</i>	
20,000	179,998	120,000	30	159.7	<i>not attempted</i>	

Problem “elec”. The COPS problem finds an equilibrium state distribution of n_p electrons positioned on a conducting sphere. The NLP minimizes a nonlinear objective subject to quadratic equality constraints. The problem has an exponentially large number of local minima, and KNITRO merely finds one of them.

The table below shows KNITRO performance for increasing problem size. KNITRO default options were used, except for the choice of algorithm. Experiments on smaller versions of the problem show the Interior-Point/CG algorithm (IP-CG) running significantly faster than Interior-Point/Direct (IP-Direct); this is because the Hessian matrix is nearly dense (*nnz H* is the number of nonzero elements). The Direct algorithm factorizes the Hessian matrix, while CG (conjugate gradient) simply multiplies it by a series of vectors.

n_p	<i>vars</i>	<i>cons</i>	<i>nnz H</i>	IP-CG (alg=2)		IP-Direct (alg=1)	
				<i>num FC</i>	<i>CPU sec</i>	<i>num FC</i>	<i>CPU sec</i>
100	300	100	45,150	94	1.5	94	4.1
500	1,500	500	1,125,750	144	78.6	144	431.4
1,000	3,000	1,000	4,501,500	244	581.4	<i>not attempted</i>	

Even the faster Interior Point/CG algorithm is taking too much time as the size of this problem increases. Examination of the execution time summary printed at the end of each KNITRO run reveals that the dominant cost is evaluation of AMPL quantities. A more detailed timing breakdown is given in the file `kdbg_profileIP.log` with user option `debug=1`. It shows that 31% to 45% of CPU time is spent evaluating the Hessian matrix in AMPL (see the table below). The default Hessian option (`hessopt=1`) in KNITRO requests the full matrix from AMPL at each major iteration. Two alternative Hessian options are worth considering to try and reduce execution time.

Hessopt=5 requests that AMPL evaluate Hessian-vector products instead of the full Hessian matrix. However, the table below shows that this is even less efficient. The table reports the number of evaluations of full Hessians (*num H*) and Hessian-vector products (*num Hv*), and the percentage of CPU time spent in AMPL evaluating each of these (*% CPU*).

n_p	IP-CG, hessopt=1 (full Hessian evals)				IP-CG, hessopt=5 (Hessian-vector prods)			
	<i>num FC</i>	<i>CPU sec</i>	<i>num H</i>	<i>% CPU</i>	<i>num FC</i>	<i>CPU sec</i>	<i>num Hv</i>	<i>% CPU</i>
100	94	1.5	26	31%	94	3.9	723	86%
500	144	78.6	49	45%	144	164.2	1,278	87%
1,000	244	581.4	65	38%	222	903.3	1,749	84%

Hessopt=6 instructs KNITRO to use a limited memory BFGS approximation in place of the real Hessian. This causes the algorithm to take more iterations, but Hessian computations become much cheaper. The table below shows there are some savings in time in comparison with hessopt=1.

n_p	IP-CG (hessopt=1)		IP-CG (hessopt=6)	
	<i>num FC</i>	<i>CPU sec</i>	<i>num FC</i>	<i>CPU sec</i>
100	94	1.5	614	4.0
500	144	78.6	842	125.1
1,000	244	581.4	862	503.8

References

- [1] Elizabeth D. Dolan, Jorge J. Moré and Todd S. Munson. “Benchmarking Optimization Software with COPS 3.0.” Technical Report ANL/MCS-TM-273, Mathematics and Computer Science Division, Argonne National Laboratory, 2004.
- [2] Robert Fourer, David M. Gay, Brian W. Kernighan. *AMPL, a Modeling Language for Mathematical Programming*. 2nd ed. Toronto, Ontario: Thomson Brooks/Cole, 2003.

KNITRO is a premier solver for nonlinear optimization problems, handling bound constraints, nonlinear equalities and inequalities (both convex and nonconvex), and complementarity constraints. KNITRO solves large-scale NLPs, LPs, QPs, MPCCs, nonlinear systems of equations, and least squares problems. KNITRO is available as a thread-safe, embeddable software library on multiple platforms, with programmatic APIs and interfaces to major modeling languages. Full support and continued development of KNITRO is provided by Ziena Optimization, Inc. For more information, visit <http://www.ziena.com>.