CME 338 Final Project

Matthew Zahr

June 14, 2013

Abstract

For the final project, I enhanced the PDCO code on the SOL website by adding Method = 22, which is identical to the SQD Method = 21 with the original sparse LU (lu.m) replaced with the MA57 LDL^T factorization (ldl.m). Method 22 is then compared to Methods 1, 2, 3, 21 in a very similar study that was performed in Homework 5.

Results

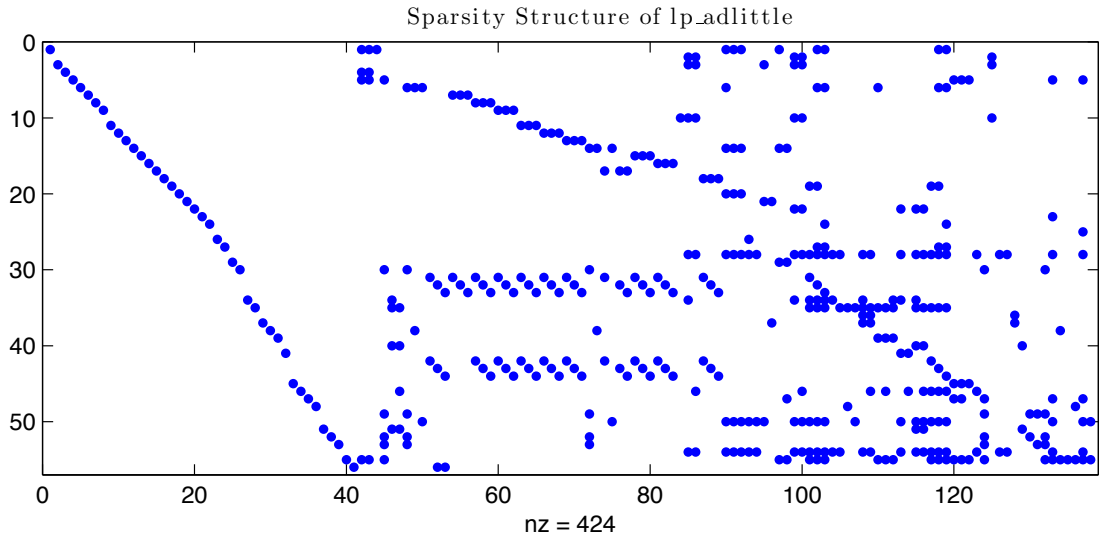
Figure 1 contains the sparsity structure of the first problem considered, lp_adlittle from Tim Davis' sparse matrix collection. Table 1 contains the PDCO results for this problem (lp_adlittle.mat). All of the direct solvers (Cholesky, QR, indefinite-Cholesky type - both lu and ldl factorization) for computing the search direction give exactly the same number of PDCO iterations (as expected since they should be computing identical search directions, modulo round-off error). The direct solvers required 21 PDCO. Also, LSMR gives the same number of PDCO iterations (21) as the direct methods when the tolerances are small. When the tolerances for LSMR are large, more PDCO iterations (24) are required because the search directions are only computed approximately. It was also interesting to note that the number of LSMR iterations increased quite rapidly when small tolerances were used; the increase in LSMR iterations was slower when large tolerances were used and an upper bound of about 108 iterations was hit. As far as CPU times go, method 21 and 22 were the fastest (Method 22 was a bit faster than Method 21, but the CPU times are so small that uncertainty in the timing may be responsible for the differences) with the Cholesky and QR methods about tied for second fastest. The LSMR methods were the slowest, but it was beneficial to use larger tolerances (accept more PDCO iterations for a slower increase in LSMR iterations).

Table 1: PDCO Results: (lp_adlittle.mat)

Method	**(ATol1,ATol2)**	**# PDCO**	**Time (sec)**
Chol	-	21	0.182
QR	-	21	0.190
LSMR	Default	21	0.523
LSMR	(1e-6,1e-6)	24	0.377
SQD	-	21	0.0542
SQD (MA57)	-	21	0.0377

Figure 2 contains the sparsity structure of the second problem considered, lp_israel from Tim Davis' sparse matrix collection. Notice that this problem has a few rather dense columns. Table 2 contains the PDCO results for this problem (lp_israel.mat). First, we observe that all of the direct solvers (Cholesky, QR, indefinite-Cholesky type - both lu and ldl factorization) returned sub-optimal solutions at "convergence," while the LSMR based algorithm required fewer PDCO iterations (32 instead of 36) and obtained a better solution. This is a rather strange and suggesting that this may be an issue with optimality tolerances (the direct solver algorithms generate a point that barely satisfies optimality while LSMR generates one that satisfies the optimality condition by a wide margin). This is, in fact, the case since I re-ran all of

Figure 1: LP Sparsity (lp_adlittle.mat)



the tests with the optimality tolerance set to 10^{-8} instead of 10^{-6} and all methods converged to the same value of the objective function (-4.7971602×10^5); the direct algorithms required 44 iterations with this new tolerance and LSMR (default atol1 and atol2) required 39. Notice that the using LSMR with relaxed tolerances fails to converge (due to linesearch failures).

Figure 2: LP Sparsity (lp_israel.mat)

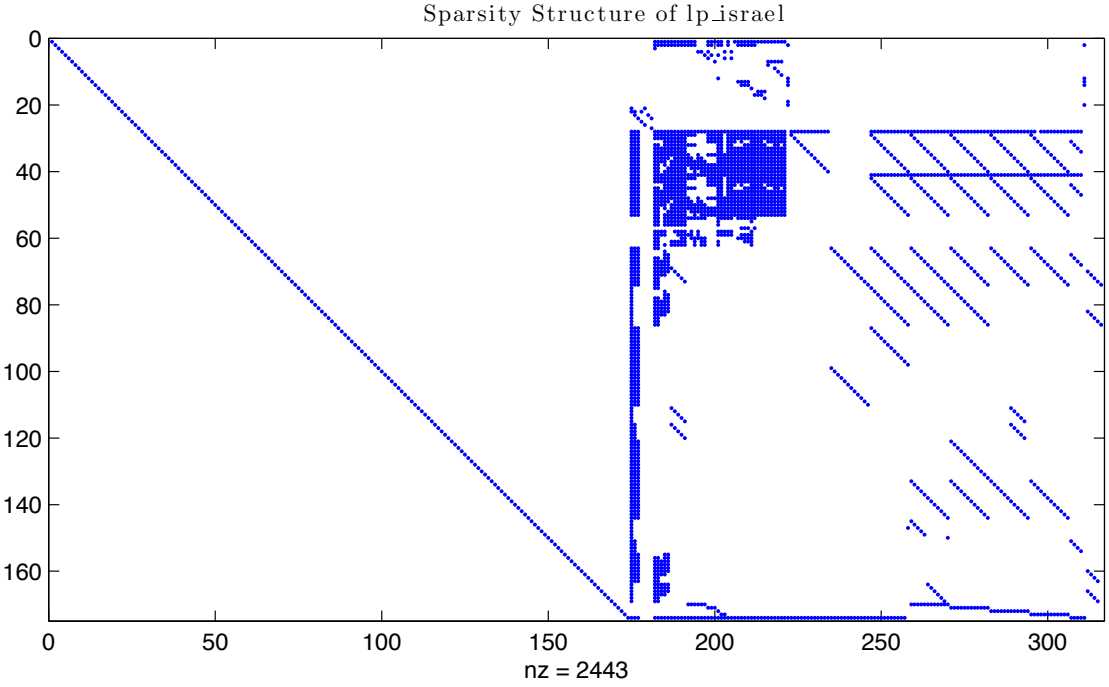


Figure 3 contains the sparsity structure of the third problem considered, lp_sc105 from Tim Davis' sparse matrix collection. Table 3 contains the PDCO results for this problem (lp_sc105mat). All solvers that

Table 2: PDCO Results: Small Problem (lp_israel.mat)

Method	**(ATol1,ATol2)**	**# PDCO**	**Time (sec)**
Chol	-	36	0.395
QR	-	36	0.431
LSMR[1]	Default	32	0.985
LSMR	(1e-6,1e-6)	Failed	Failed
SQD	-	36	0.128
SQD (MA57)	-	36	0.261

LSMR found point with lower objective function (-4.227×10^5) than Chol, QR, and SQD (-3.817×10^5).

successfully found the optimal solution required exactly 16 PDCO iterations to do so. LSMR with relaxed tolerances again failed to converge due to linesearch failures. For this problem, the Cholesky and SQD with MA57 were the fastest algorithms with LSMR the slowest. The CPU times are again very small which means that timing uncertainty could be responsible for the differences.

Figure 3: LP Sparsity (lp_sc105.mat)

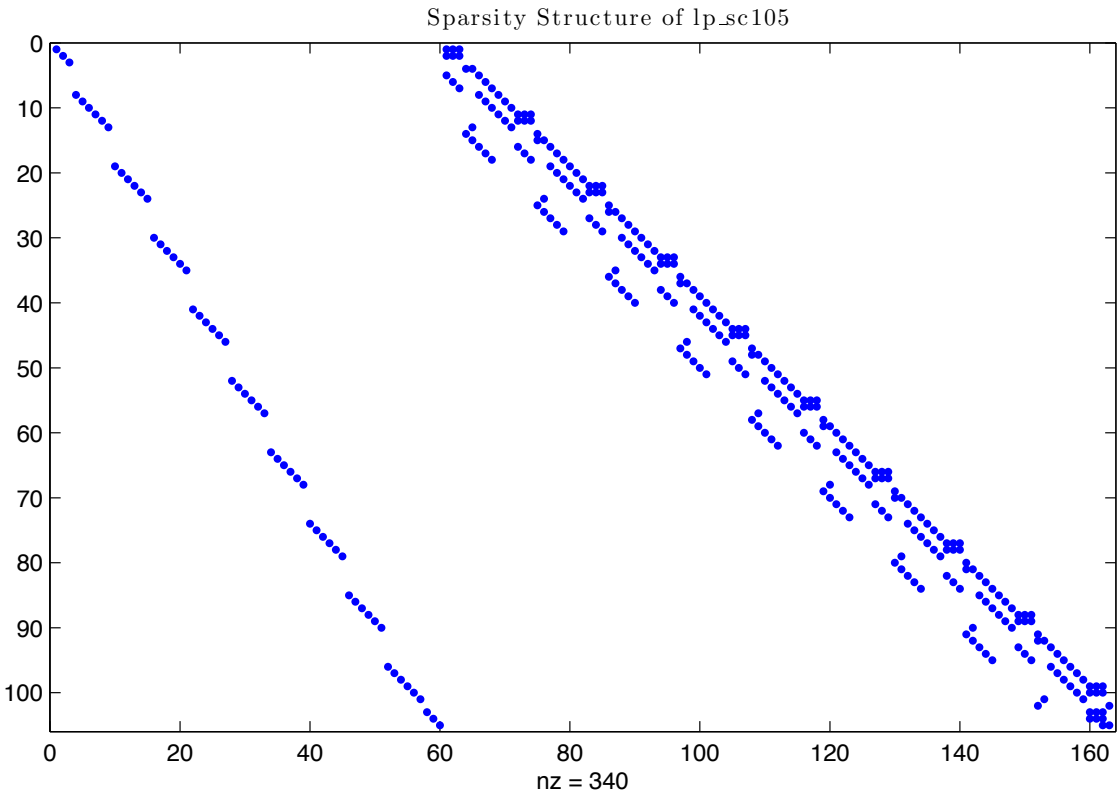


Figure 4 contains the sparsity structure of the fourth problem considered, lp_ship12l from Tim Davis' sparse matrix collection. Table 4 contains the PDCO results for this problem (lp_sc105mat). All solvers that successfully found the optimal solution required exactly 28 PDCO iterations to do so. LSMR with relaxed tolerances again failed to converge due to linesearch failures. For this problem, the sparse Cholesky algorithm was the fastest with LSMR by far the slowest. For all direct algorithms, the CPU times are again very small which means that timing uncertainty could be responsible for the differences. Table 5 contains the PDCO results for the lpi_ceria3d. All of the direct solvers (Cholesky, QR, indefinite-Cholesky type) for computing the search direction give exactly the same number of PDCO iterations

Table 3: PDCO Results: Small Problem (lp_sc105.mat)

Method	**(ATol1,ATol2)**	**# PDCO**	**Time (sec)**
Chol	-	16	0.0249
QR	-	16	0.0595
LSMR	Default	16	0.452
LSMR	(1e-6,1e-6)	Fail	Fail
SQD	-	16	0.0448
SQD (MA57)	-	16	0.0359

Figure 4: LP Sparsity (lp_ship12l.mat)

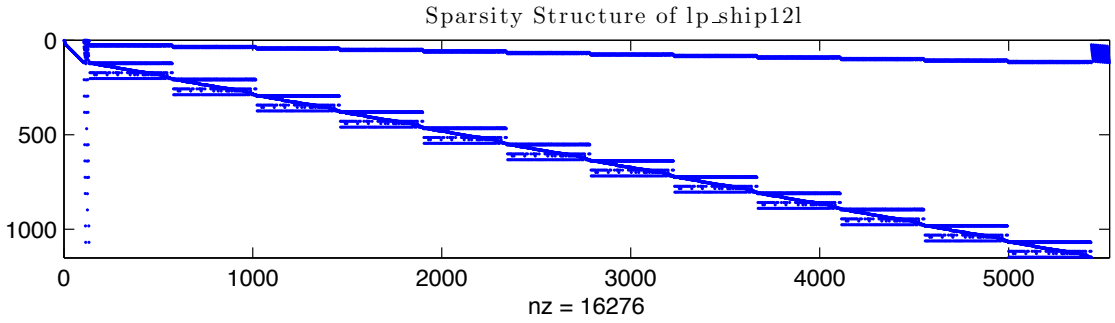


Table 4: PDCO Results: Small Problem (lp_ship12l.mat)

Method	**(ATol1,ATol2)**	**# PDCO**	**Time (sec)**
Chol	-	28	0.302
QR	-	28	0.454
LSMR	Default	28	3.155
LSMR	(1e-6,1e-6)	Fail	Fail
SQD	-	28	0.533
SQD (MA57)	-	28	0.819

(as expected since they should be computing identical search directions, modulo round-off error). The direct solvers required 43 PDCO. For this problem, even with small tolerances, LSMR required more iterations than the direct solvers; this is due to the fact that during the later PDCO iterations, the LSMR algorithm was not converging to the specified tolerance before reaching its maximum number of iterations (178800) causing the later search directions to differ from the directions computed with the direct solvers. I did not determine the exact number of PDCO iterations required for the large problem (with small tolerances) because I killed the run after 70 iterations and over an hour of compute time. When the tolerances for LSMR were large, the optimization failed due to a large number of linesearch failures (likely due to an inaccurate search direction for the relaxed LSMR tolerances). As far as CPU times go, method 21 and 22 were the fastest (by a wide margin) with the Cholesky method second fastest and the QR method third fastest (Method 21 outperformed Method 22 by nearly a factor of 4 in this case). The LSMR methods were the slowest; in this case, using relaxed LSMR tolerances resulted in a failure to converge.

Figure 5: LP Sparsity (lpi_ceria3d.mat)

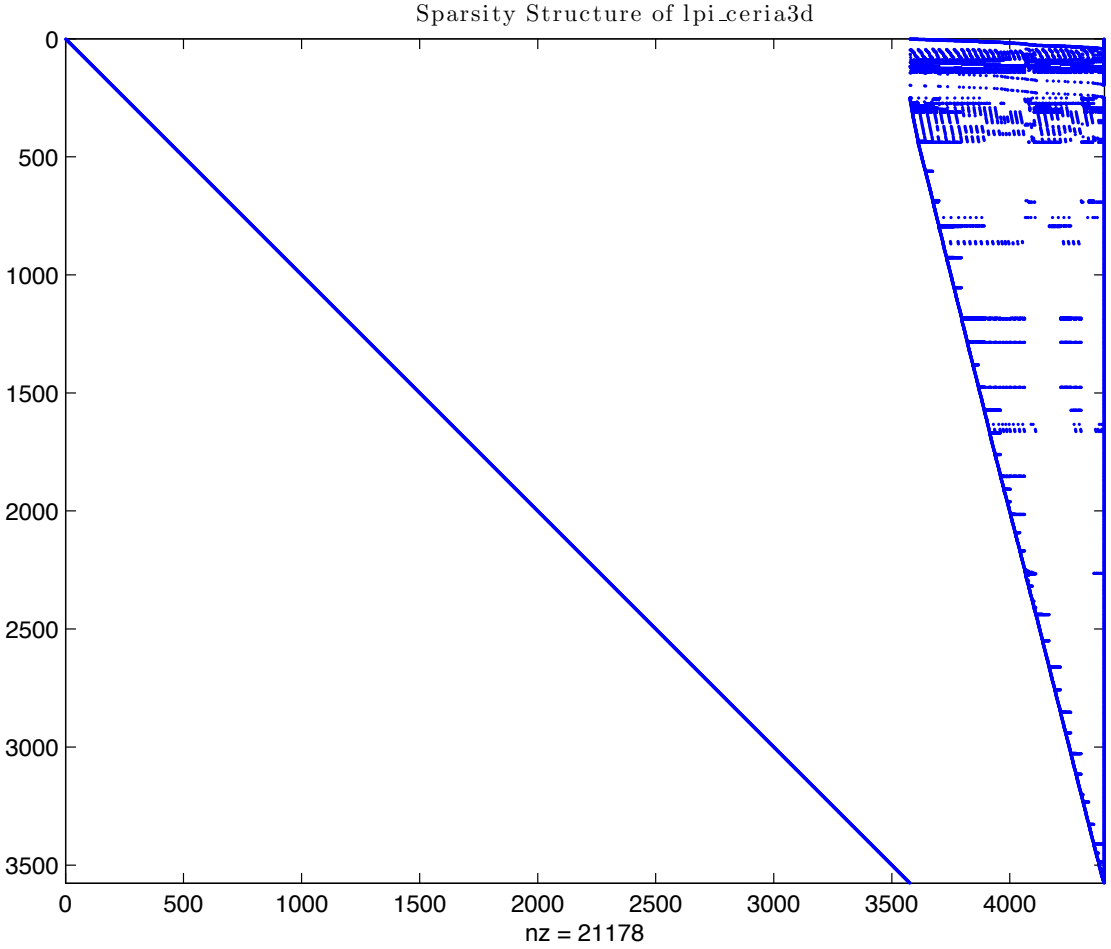


Table 5: PDCO Results: Large Problem (lpi_ceria3d.mat)

Method	**(ATol1,ATol2)**	**# PDCO**	**Time (sec)**
Chol	-	43	78.9395
QR	-	43	352.8506
LSMR	Default	> 70	$>$ 1hr
LSMR	(1e-6,1e-6)	Failed	Failed
SQD	-	43	1.2471
SQD (MA57)	-	43	4.072