L1-Magic Benchmarks and Potential Improvements

Stephen Conover

September 2, 2015

With Thanks to Chris Turnes and Adam Charles

Georgia Institute of Technology

CSIP
Center for Signal & Image Processing

Introduction

1. Improvements to Optimization Library L1-Magic
2. A Few Techniques
    1. Don't waste memory or allocate unnecessarily
    2. Avoid big operations when small ops work
    3. Do smart block operations when you can to exploit sparsity and structure

L1-Magic

1. **P_1: Basis Pursuit (L1 with Equality Constraints)**
   $$\boldsymbol{min\|x\|_1 \text{ subject to } Ax = b} \ (P_1)$$
2. P_A: Decode
   $$\min_x \|y - Ax\|_1$$
3. **P_2: L1 with Quadratic Constraints**
   $$\boldsymbol{min\|x\|_1 \ \text{ subject to } \|Ax - b\|_2 \le \epsilon}$$
4. P_D: Dantzig Selector
   $$min\|x\|_1 \ \text{ subject to } \|A^*(Ax - b\|_\infty \ \le \gamma$$
5. **TV1: Total Variation with Equality Constraints**
   $$\boldsymbol{\min TV(x) \text{ subject to } Ax = b}$$
6. TV2: Total Variation with Quadratic Constraints
   $$\min TV(x) \text{ subject to } \|Ax - b\|_2 \le \epsilon$$
7. TV_D: Dantzig TV

Goal

- Improve Performance
  - Reduce Execution Time
  - Reduce Memory Requirements
- Improvement Philosophy
  - No new fundamentals
  - Avoid MEX
  - CPU Now, GPU Later

Potential Improvements

1. Memory Allocation and Sparse Data Structures
2. Woodbury Identity
3. Block Matrix for Sparse + Dense Matrices

Problem P1: Basis Pursuit

Definition: $min\|x\|_1$ subject to $Ax = b$

Improvement: Memory Usage with Sparse Storage

Profiler Results

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time
139	`H11p = A*(sparse(diag(1./sigx)...`	10	0.066 s	73.3%
148	`Adx = A*dx;`	10	0.003 s	3.3%
141	`[dv,hcond] = linsolve(H11p, w1...`	10	0.003 s	3.3%
138	`w1p = -(w3 - A*(w1./sigx - w2....`	10	0.002 s	2.2%
87	`u = (0.95)*abs(x0) + (0.10)*ma...`	1	0.002 s	2.2%

Amdahl's Law

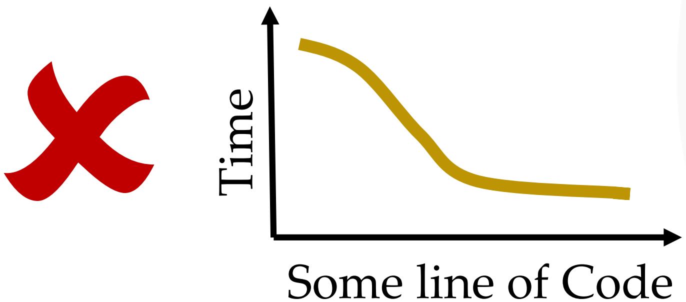
- The program can be divided into fraction of the code we can improve, P, and the faction of code we cannot, $(1 - P)$, to calculate the overall speedup of the program, $S(N)$, resulting from a factor N speedup of P

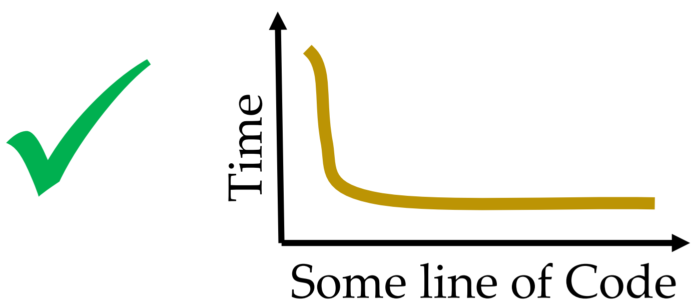
$$S(N) = \frac{1}{(1-P_t)+\frac{P_t}{N}}$$

- P may be defined in terms of P_t, the fraction of total execution time, rather than lines of code
- If $P_t \cong 1$ and corresponds to a small number of lines of code, then then rate of return can be significant

Concentration and Speedup

$$S(N) = \frac{1}{(1-P_t)+\frac{P_t}{N}}$$



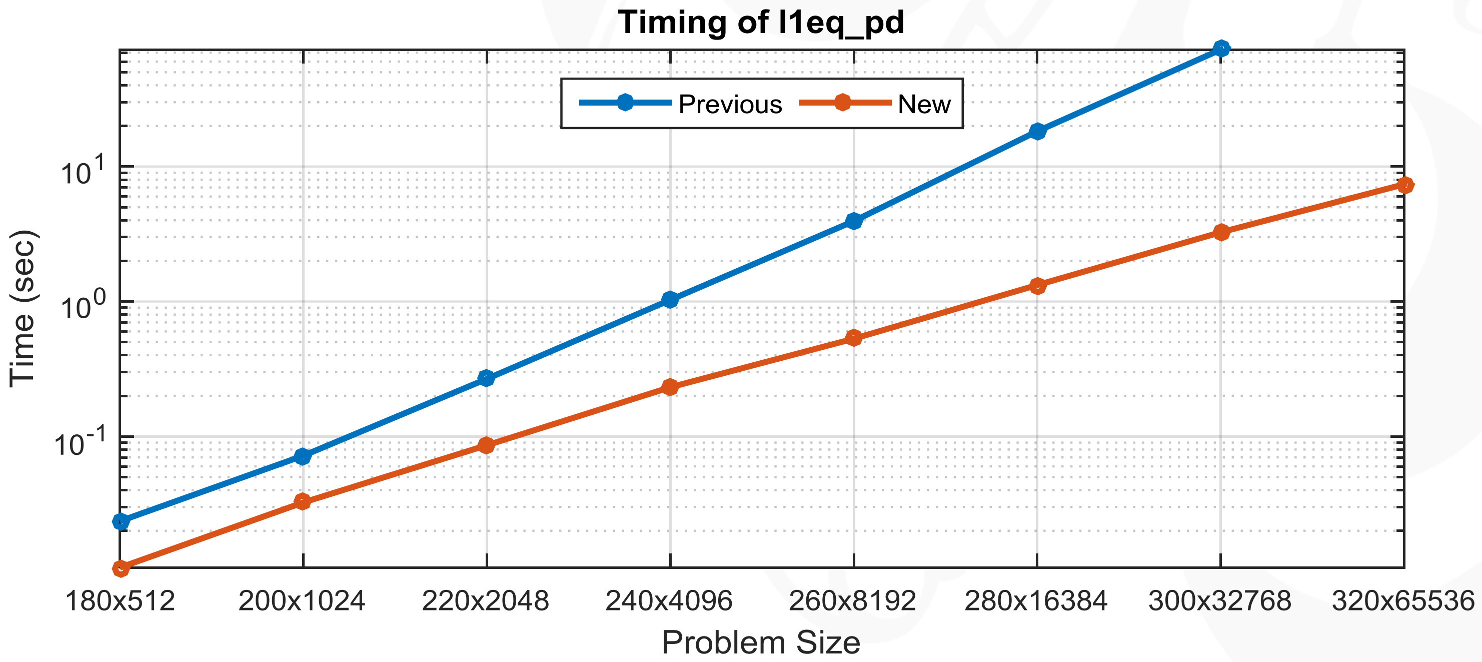


	Speedup		
P_t	Max	N=100	N=50
25%	1.3	1.3	1.3
50%	2	2	2
90%	10	9	8
95%	20	17	14
98%	50	34	25
99%	100	50	34

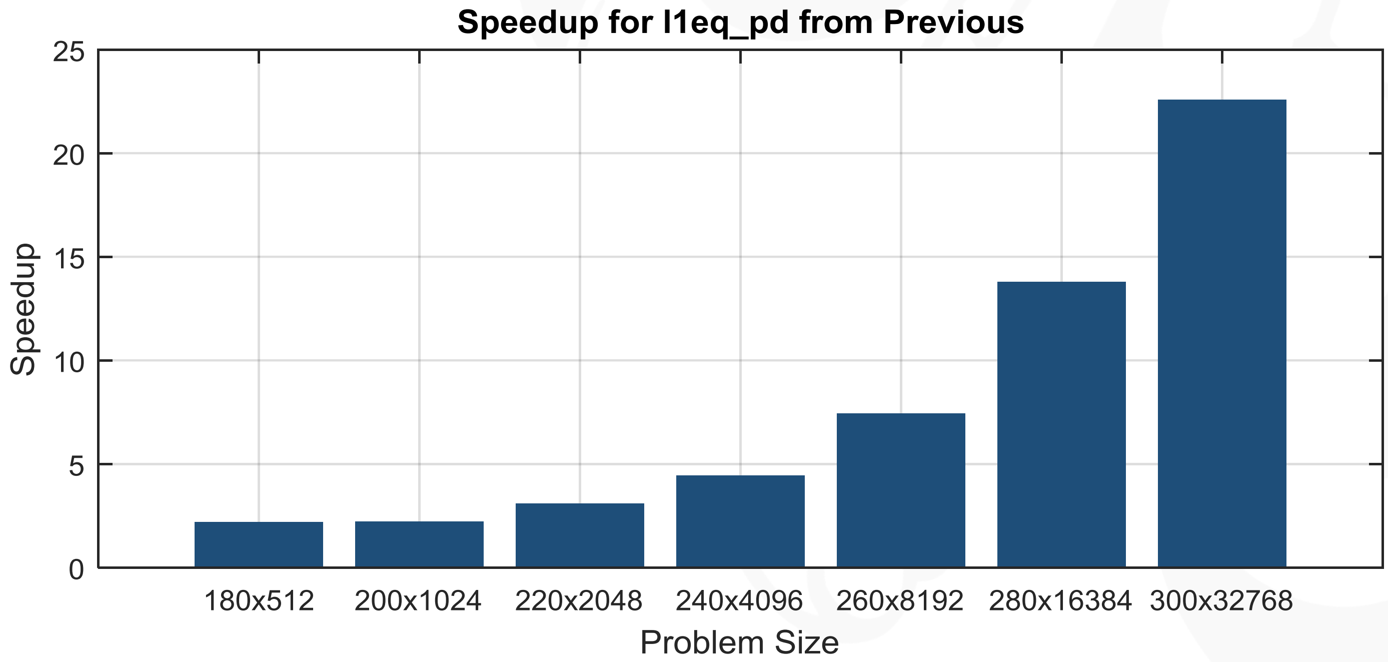
CODE

1. Original
   1. `w1p = -(w3 - A*(w1./sigx - w2.*sig2./(sigx.*sig1)));`
   2. `H11p = A*(sparse(diag(1./sigx))*A');`
   3. `opts.POSDEF = true; opts.SYM = true; dv,hcond] = linsolve(H11p, w1p, opts);`
2. Memory Allocation Improvement
   1. `w1p = -(w3 - A*(w1./(sigx) - w2.*sig2./(sigx.*sig1)));`
   2. **`SS = (sparse((1:N), (1:N), (1./(sigx))', N,N));`**
   3. **`H11p =  A*(SS*A');`**
   4. `opts.POSDEF = true; opts.SYM = true; [dv,hcond] = linsolve(H11p, w1p, opts);`

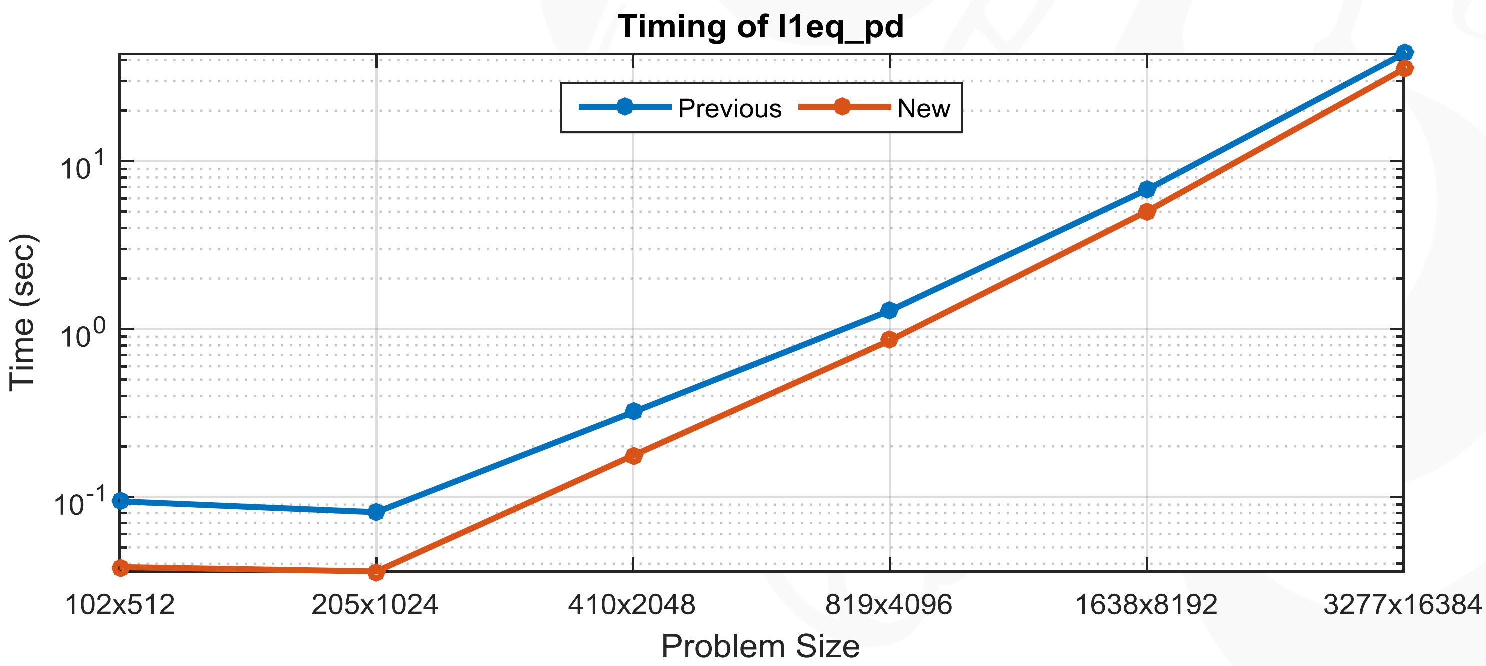
RESULTS PART I



RESULTS II

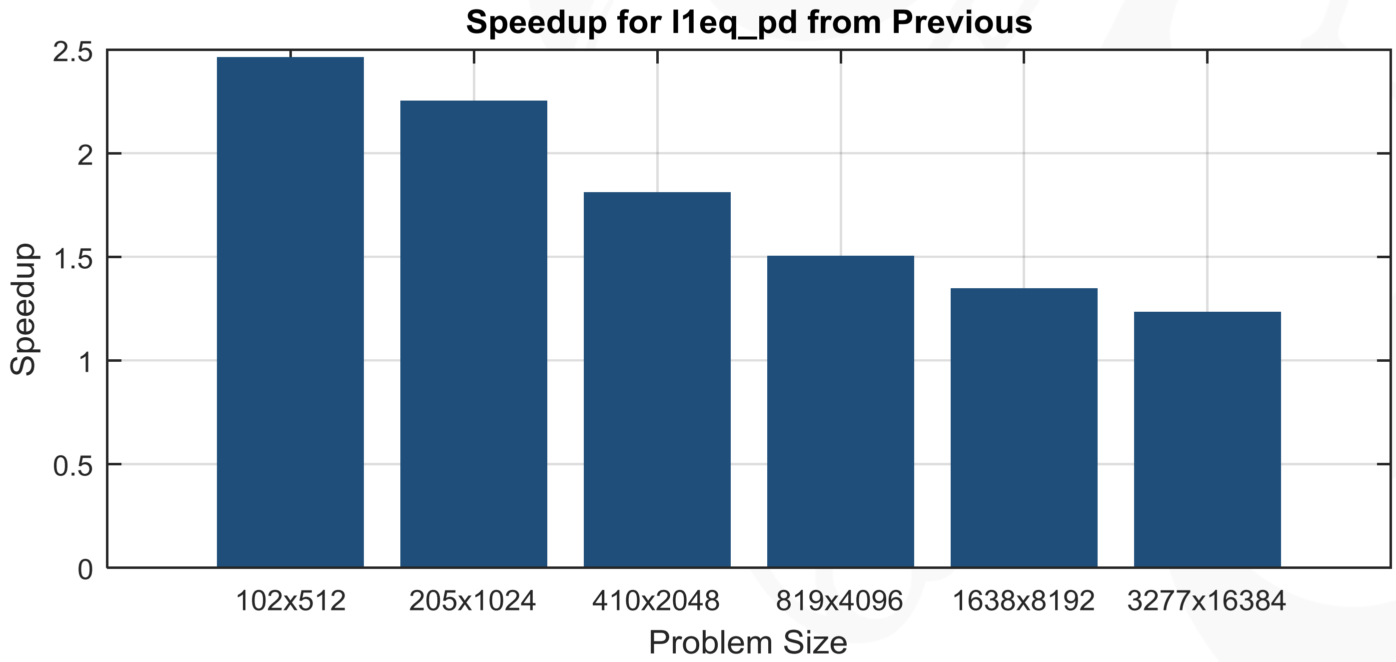

Speedup for l1eq_pd from Previous
Speedup
0
5
10
15
20
25
180x512
200x1024
220x2048
240x4096
260x8192
280x16384
300x32768
Problem Size

RESULTS III



Timing of l1eq_pd
Previous
New
Time (sec)
10^1
10^0
10^-1
102x512
205x1024
410x2048
819x4096
1638x8192
3277x16384
Problem Size

RESULTS IV



Speedup for l1eq_pd from Previous
Speedup
2.5
2
1.5
1
0.5
0
102x512
205x1024
410x2048
819x4096
1638x8192
3277x16384
Problem Size

Problem P2: Quadratic Constraints

Definition: $min\|x\|_1$ subject to $\|Ax-b\|_2 \leq \epsilon$

Improvement: Sparse Memory & Low Rank Updates

THE CODE

```
N = length(sigx);
SX = (sparse((1:N), (1:N), ((sigx))', N,N));
H11p = SX - (1/fe)*AtA + (1/fe)^2*(atr*atr');
opts.POSDEF = true; opts.SYM = true;
[dx,hcond] = linsolve(H11p, w1p, opts);
```

$$dx = \left(SX - \frac{1}{fe} A^T A + \left(\frac{1}{fe^2} A^T r r^T A \right) \right)^{-1} w1p$$

Two Problems

1. **Code is slower than it could be**
2. **Has stability issues**

Low Rank Updates

- Woodbury Matrix Identity

$$(S + UCV)^{-1} =$$
$$S^{-1} - S^{-1}U(C^{-1} + VS^{-1}U)^{-1}VS^{-1}$$

- Sherman-Morrison Formula

$$(M - uv^T)^{-1} = M^{-1} - \frac{M^{-1}uv^TM^{-1}}{1 + v^TM^{-1}u}$$

Rearrangement

1. $dx = \mathrm{H}^{-1} * \mathrm{w1}p = \left(SX - \frac{1}{fe} A^T A + \left(\frac{1}{fe^2} A^T r r^T A \right) \right)^{-1} w1p$
2. $H = S_x - \frac{1}{f_e} A^T A + \frac{1}{f_e^2} A^T r r^T A = S_x + \frac{1}{f_e} A^T \left(-I + \frac{1}{f_e} r r^T \right) A$
3. Apply to the Woodbury Matrix Identity with
   $S = S_x = diag(sig_x) \qquad U = \frac{1}{f_e} A^T \qquad V = A \;\; C = -I + \frac{1}{f_e} r r^T$
4. $H^{-1} = S^{-1} - S^{-1} U (C^{-1} + V S^{-1} U)^{-1} V S^{-1}$

$$= S_x^{-1} - \frac{1}{fe} S_x^{-1} A^T \left(C^{-1} + \frac{1}{f_e} A S_x^{-1} A^T \right)^{-1} A S_x^{-1}$$

More Rearrangement

Use the Sherman-Morrison Formula:

$$C^{-1} = (M + uv^T)^{-1} = M^{-1} - \frac{M^{-1}uv^TM^{-1}}{1 + v^TM^{-1}u}$$

$and\ set\ M = M^{-1} = -I \quad and \quad u = v = \frac{1}{\sqrt{f_e}}\mathrm{r}$

to get

$$C^{-1} = \frac{rr^T}{r^Tr - f_e} - I$$

Finally

Altogether, this gives us

$$H^{-1}y = S_x^{-1}y - \frac{1}{fe}S_x^{-1}A^T(G)^{-1}AS_x^{-1}y$$

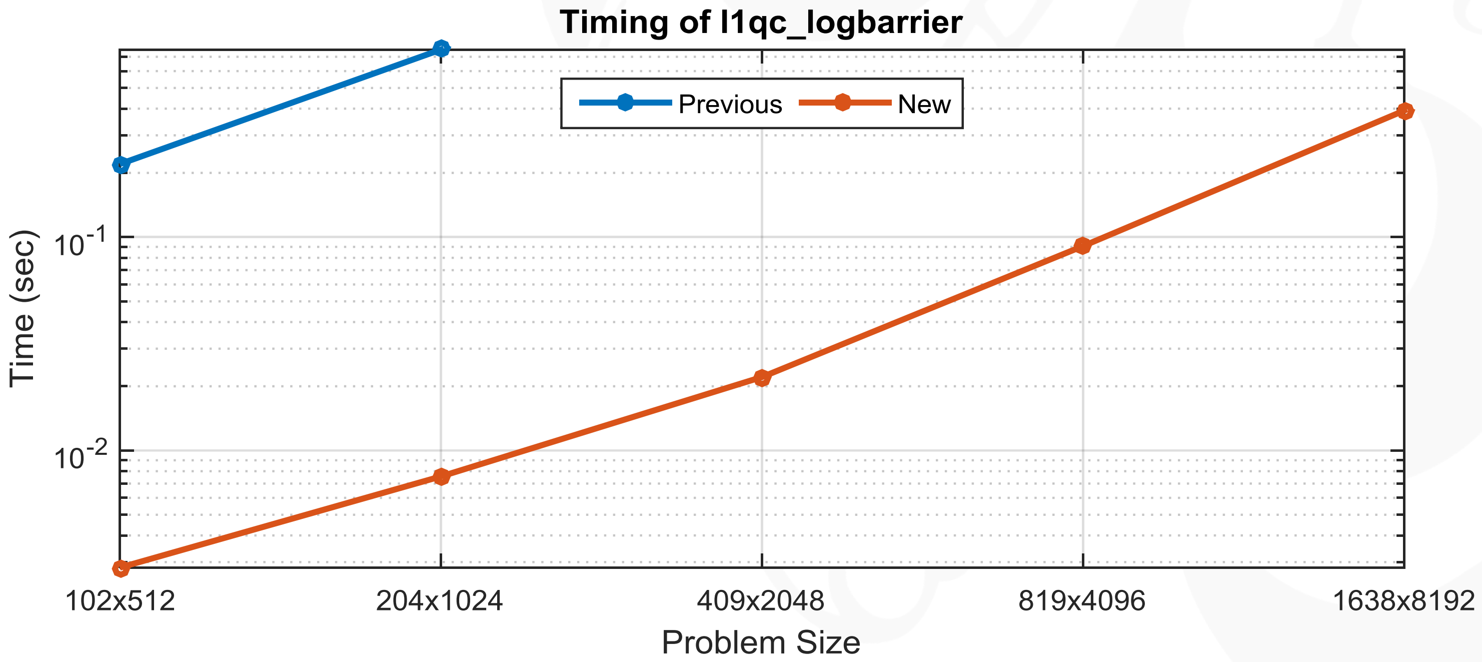
Where

$$G = \frac{rr^T}{r^T r - f_e} - I + \frac{1}{f_e}AS_x^{-1}A^T$$

And so

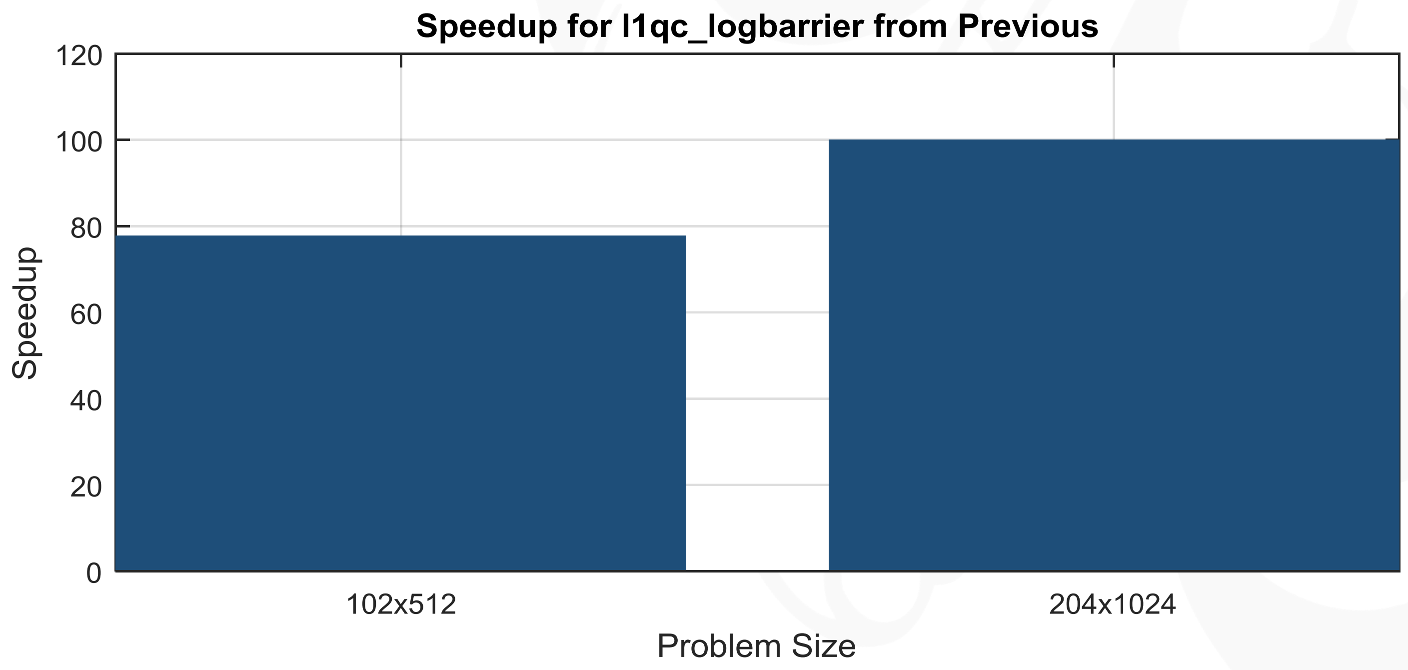
```
B = bsxfun(@times,A',sqrt(1./sigx));
dx = w1p./sigx - 1/fe*A'*(-linsolve(-(r*r'/(r'*r-fe)-eye(k) + (1/fe)*(B'*B)), A*(w1p./sigx)))./sigx;
```

RESULTS



Timing of l1qc_logbarrier
Previous
New
Time (sec)
10^{-1}
10^{-2}
102x512
204x1024
409x2048
819x4096
1638x8192
Problem Size

RESULTS



Speedup for l1qc_logbarrier from Previous
120
100
80
60
40
20
0
Speedup
102x512
204x1024
Problem Size

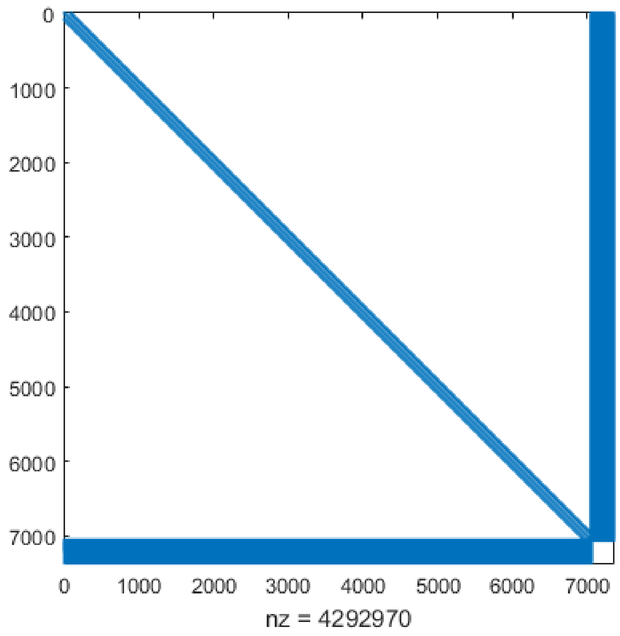
PROBLEM TV: TOTAL VARIATION

Definition: $\min TV(x)$ subject to $Ax = b$

Improvement: Block Matrix Inverse

The Setup

- The Newton Step Involves both the operator A and the TV operator



$$A = \begin{bmatrix} H & \sigma A \\ \sigma A' & 0 \end{bmatrix}$$

The Code

```
H11p =  Dh'*sparse(diag(-1./ft + sigb.*Dhx.^2))*Dh + ...
Dv'*sparse(diag(-1./ft + sigb.*Dvx.^2))*Dv + ...
Dh'*sparse(diag(sigb.*Dhx.*Dvx))*Dv + ...
Dv'*sparse(diag(sigb.*Dhx.*Dvx))*Dh;
afac = max(diag(H11p));
Hp = full([H11p afac*A'; afac*A zeros(K)]);
opts.SYM = true;
    [dxv, hcond] = linsolve(Hp, wp, opts);
```

THE CODE WITH SPARSE

```
S1 = sparse((1:N), (1:N), -1./ft + sigb.*Dhx.^2, N,N);
 S2 = sparse((1:N), (1:N), -1./ft + sigb.*Dvx.^2, N,N);
 S3 = sparse((1:N), (1:N), sigb.*Dhx.*Dvx, N,N);
 S4 = sparse((1:N), (1:N), sigb.*Dhx.*Dvx, N,N);
 H11p =  Dh'*S1*Dh + ...
   Dv'*S2*Dv + ...
   Dh'*S3*Dv + ...
   Dv'*S4*Dh;
 afac = max(diag(H11p));
Hp = full([H11p afac*A'; afac*A zeros(K)]);
     opts.SYM = true;
     [dxv, hcond] = linsolve(Hp, wp, opts);
```

Results Coming

- Never call full
- Do Block Inverse on H
- But H is poorly conditioned!
- Calculate **C = (B'*A)*B;** and do a full SVD on this kxk matrix and determine how many extra rows and columns H needs.
- Tentatively Up to x10 Faster but highly depended on K<<N.
- Results Pending

Arrowhead Solver

```
1. function [blockwiseSolution,
   conditionWarning] =
   arrowHeadSolver(A, B, w)
2. k0 = size(A,1);
3. C = (B'*A)*B;
   [U, Sc] = svd(full(C)); % grab
   the singular vectors, too
   Sc = diag(Sc);  % make them a
   vector
   U = fliplr(U); % put worst
   singular vectors first
   Sc = Sc / sum(Sc);
   Sc = cumsum(Sc);
   del = size(C,1) - nnz(Sc < (1-1e-
   4));
   % now, solve a different syste
4. w((k0+1):end) = U'*w((k0+1):end);
5. % Amod = [ A      B*U ]
   %          [U'*B'    0  ]
   Bmod = B*U;
   A = [A Bmod(:, 1:del); Bmod(:,
   1:del)', zeros(del)];
   B = [Bmod(:, (del+1):end); zeros(del,
   size(B, 2)-del)];
6. k = k0 + del;
   w1  = w(1:k);
   w2  = w((k+1):end);
7. [L,D,P,Sx] = ldl(A);
8. % S*P*(L*D*L')^(-1)*P'*S*y
   om =  (Sx*P)*(L'\((L*D)\((P'*Sx)*[w1,
   B])));
   mu = (B'*om(:,1) - w2);
   nu = (-B'*om(:,2:end)) \ mu;
9. blockwiseSolution = [om*[1; nu]; -nu];
   % re-calibrate the solution:
10.blockwiseSolution((k0+1):end,:) =
   U*blockwiseSolution((k0+1):end,:);
```

L1-Magic Comparison

How Does L1-Magic Compare with Alternatives FISTA and NESTA?

FISTA Notes

- FISTA Had Memory Issues > 2^{16} size signal because it calculates G=A'A for use in backtracking, some stopping criterion

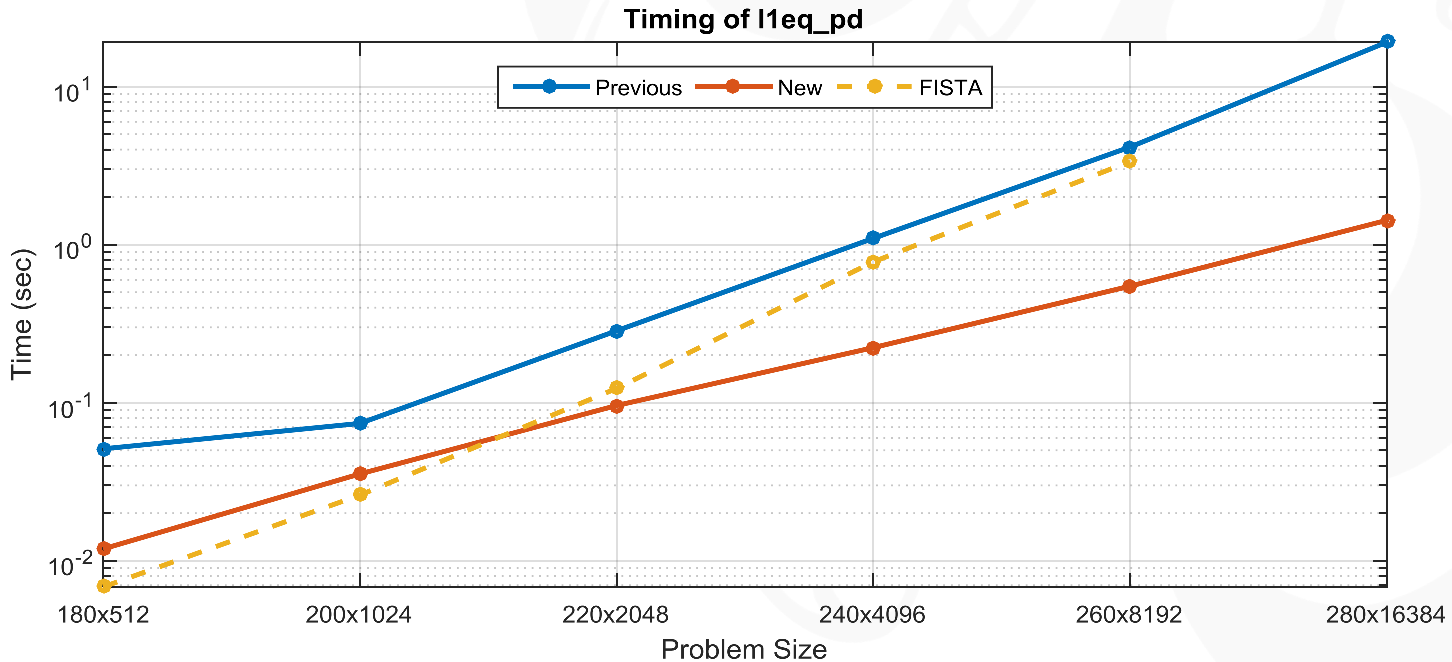
```
Creating measurment matrix...
Done.
Warning: Requested 65536x65536 (32.0GB) array exceeds maximum
array size preference. Creation of arrays greater than this
limit may take a long time and cause
MATLAB to become unresponsive. See array size limit or
preference panel for more information.
```

- Mixed Performance – Fixed by not calculating G=A'A

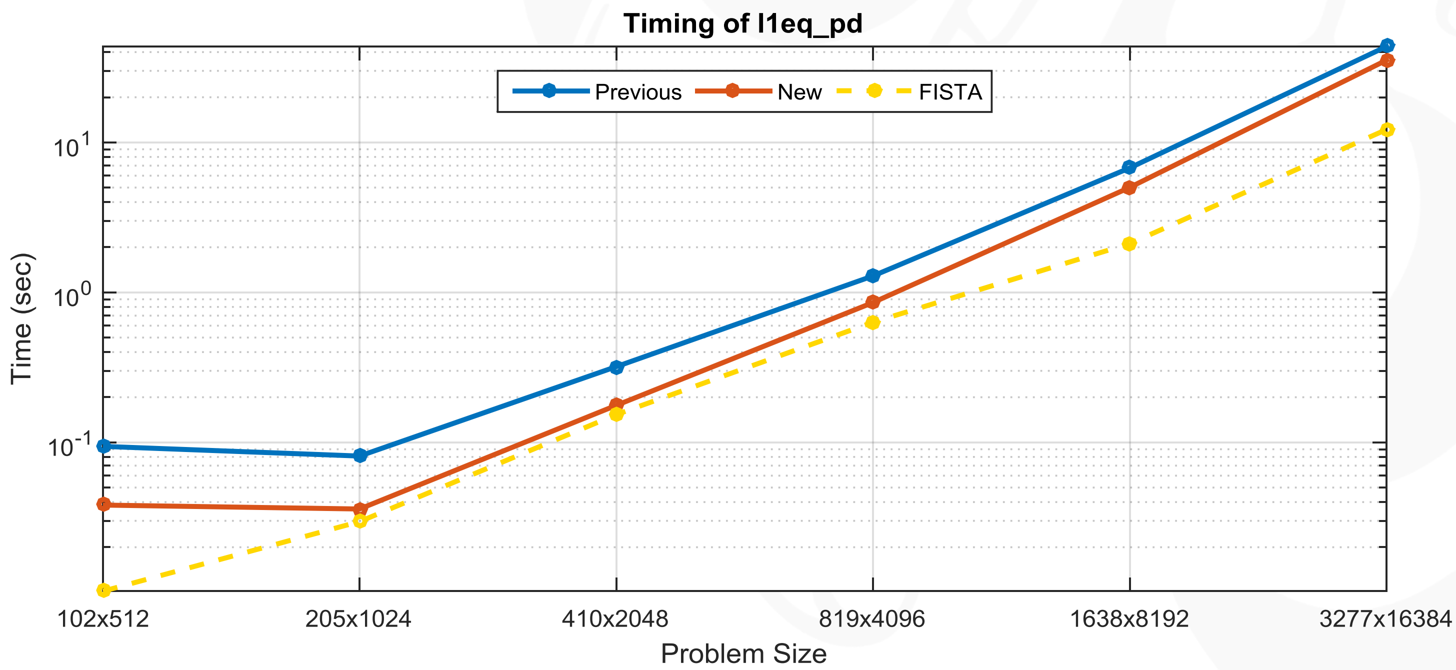
```
% G=A'*A; temp1 = G*yk - c ;
temp = A'*(A*yk) - c ; % gradient of f at yk
```

- Some convergence issues so may need to go back and tweek configuration for fair comparison

FISTA @ Small N

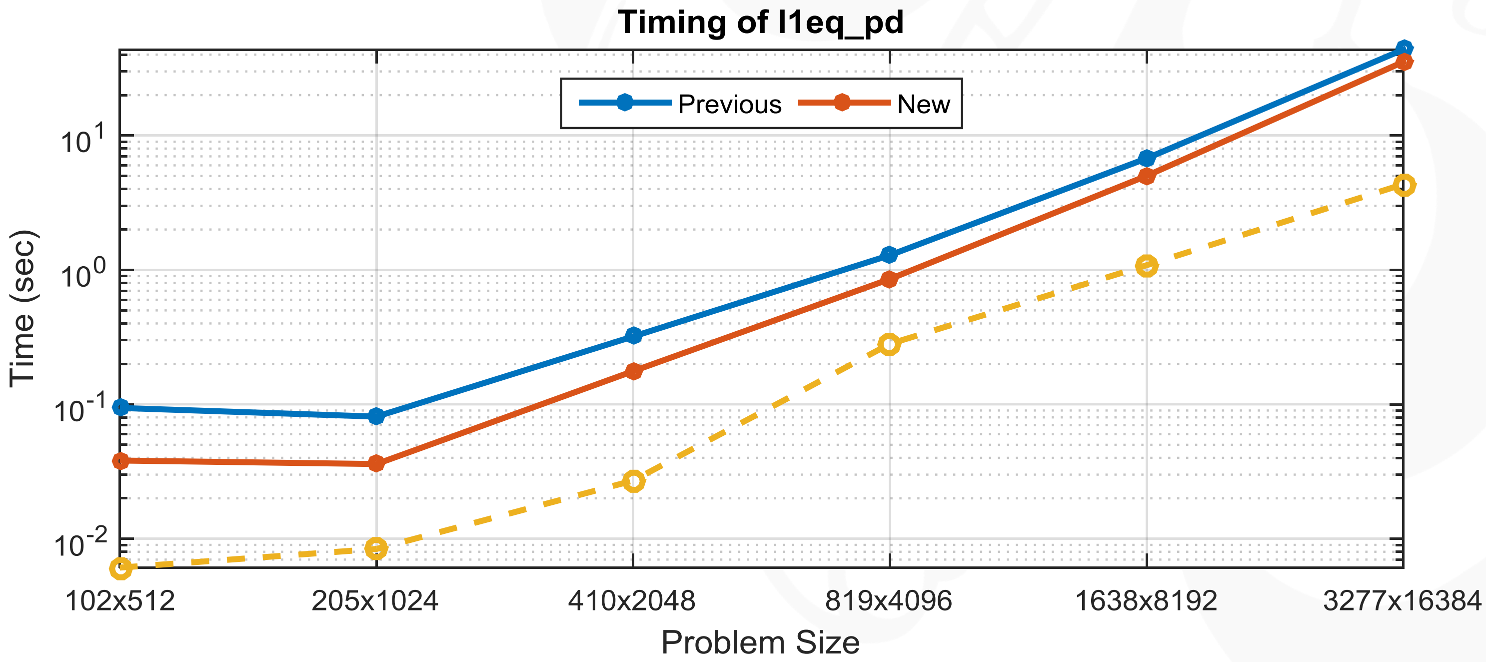


FISTA @ K=N/5

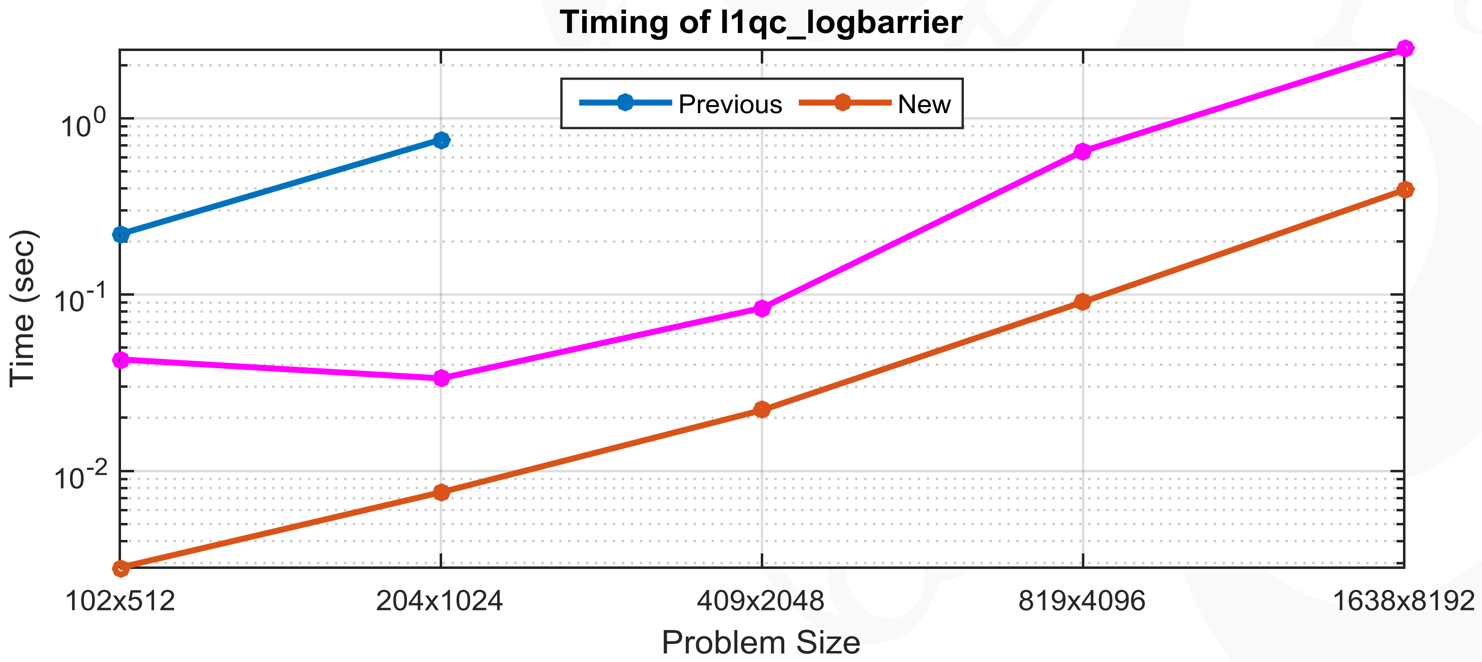


Timing of l1eq_pd
Previous
New
FISTA
Time (sec)
10^1
10^0
10^{-1}
102x512
205x1024
410x2048
819x4096
1638x8192
3277x16384
Problem Size

FISTA @ κ =N/5 with Improvement



Preliminary NESTA Results



Conclusion

- **Memory**
  - Memory Handling is important
  - If you don't run out of memory, you'll eventually run out of improvement
- **Low Rank Updates with Woodbury Identity**
  - Better Stability – kxk instead of NxN updates
  - Letter Memory Usage
  - Dependent on K<<N
- **Block Matrix**
  - Results Pending … but we don't just have to call full(…) and linsolve(…) blindly for best results and order of magnitude speed increases

END.

REFERENCES

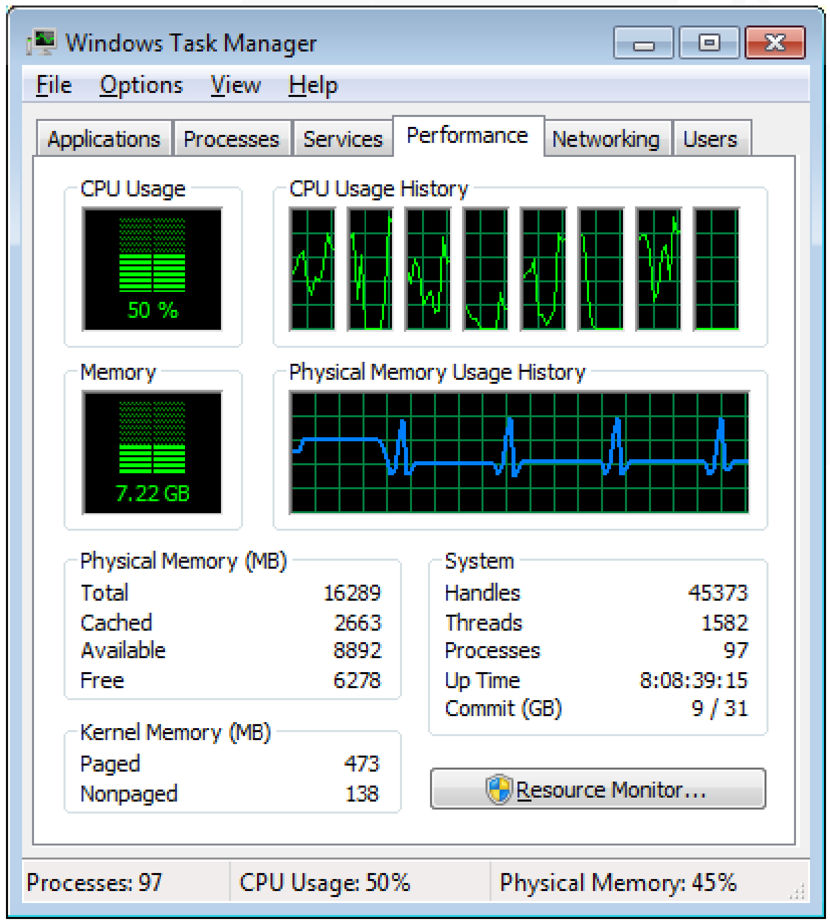
1. S. Boyd and L. Vandenberghe. ***Convex Optimization***. Cambridge University Press, 2004.
2. S. Conover. "L_1-Magic Benchmarks and Potential Improvements." Pending Report. September 2015.
3. FISTA: eecs.berkeley.edu/~yang/software/l1benchmark/
4. NESTA: statweb.stanford.edu/~candes/nesta/
5. J. Romberg. L1-Magic.
   Website: l1-magic.org
   Mirror: statweb.stanford.edu/~candes/l1magic/

Machine: ICanHaswell

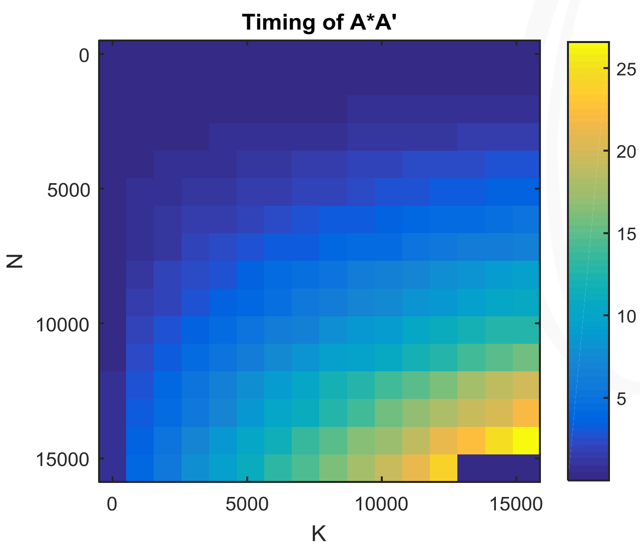
All Benchmarks Performed on the ICanHaswell development machine.

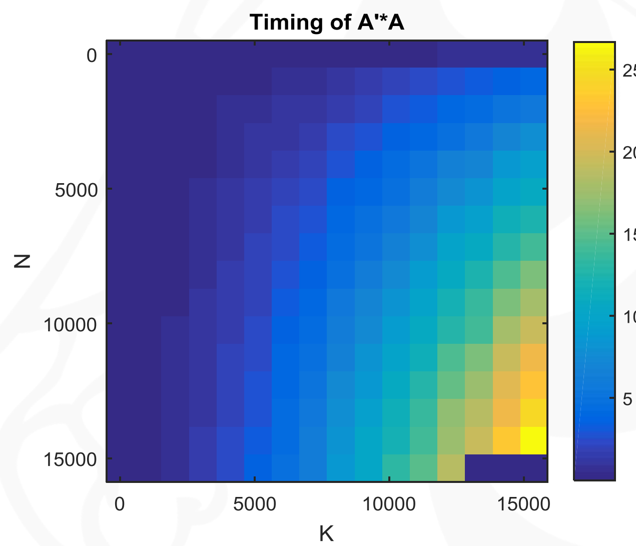
Field	Value
Name	ICanHaswell
OS	Windows 7 Pro 6.1, 7601
Motherboard	Asus
Processor	Intel Core i7-4770K @ 3.5Hz (8CPUs)
RAM	16GB
MATLAB	8.5.0.197613 (R2015a)
GPU	NVIDIA GeForce GTX 750, 4GB

L1EQ_PD(…) MEMORY SPIKES



MATRIX-MATRIX MULTIPLY I





Matrix-Matrix Multiply II