Reading Report of Figueiredo et al.

(ECE 695, Reading Assignment 01, Spring 2018)

Overview of the Paper

The paper addresses the issue of solving ℓ_1 regularized optimization problems for tasks such as sparse signal reconstruction and compressed sensing. The authors formulates the problem as a quadratic program where $\boldsymbol{x}$ is separated into positive and negative parts. Due to the constraints on the two parts, the problem then becomes a standard bounded-constrained quadratic program(BCQP).

Then the BCQP is solved using gradient projection algorithm, where we calculate the gradient of the objective function at the current solution from the previous iteration and use backtracking line search to generate a solution for the current iteration. The author also proposed an alternative approach using the concept by Barzilai and Borwein. For this approach the search direction is calculated in a different way and a standard line search is perform to calculate step size.

Prior Work

Because of the nondifferentiable nature of ℓ_1 regularized problems, generic methods such as ellipsoid method and subgradient methods can be used while being slow. And since ℓ_1 LSP can also be converted into a convex quadratic problem with linear inequality constraints, it can be solved by convex optimization solvers such as interior point method like MOSEK for small or medium sized problems. Specialized methods that exploits matrix-vector operations of $\boldsymbol{A}$ and $\boldsymbol{A}^T$ are able to handle large scale problems. Homotopy based methods such as LARS were also proposed for solving the same problem. Of course, interior points method such as the one from our first assignment is also discussed. There are also matching pursuit and orthogonal matching pursuit algorithms that can find the sparse representation of a signal on a dictionary of functions.

Key Ideas of the Paper

The first key idea of this paper is the formulation of the original ℓ_1 problem as a BCQP by introducing this positive-part operator. Using this positive-part operator, the ℓ_1 term can be simply written as the sum of two vector inner products. Since the inner products are calculated with the vector of one's, the ℓ becomes just the sum of elements of the two newly introduced variable $\boldsymbol{u}$ and $\boldsymbol{v}$. Then by careful construction, the objective function can be written as the standard BCQP.

The second key idea is about the analysis of the dimensions of BCQP. Although the BCQP formulation seems to double the dimensions of the original problem, It can be shown that the operations can be simplified to have the same dimensions as the original problem. For example, $\boldsymbol{Bz}$ can be calculated by applying $\boldsymbol{A}$ and $\boldsymbol{A}^T$ each once on the difference between $\boldsymbol{u}$ and $\boldsymbol{v}$. $\boldsymbol{z}^T\boldsymbol{Bz}$ can be reduced into applying $\boldsymbol{A}$ once followed by a vector norm operation.

Another key point is the adoption of the following thresholding operation for the search direction.

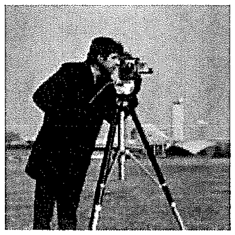
$$g_i^k = \begin{cases} (\Delta F(\boldsymbol{z}^k))_i, & if z_i^k > 0 \text{ or } (\Delta F(\boldsymbol{z}^k))_i < 0, \\ 0 & otherwise. \end{cases}$$

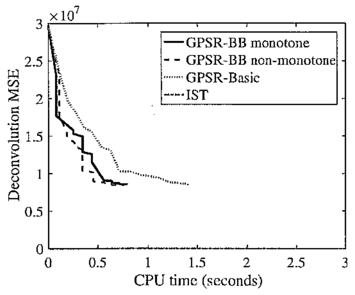
Basically, the algorithm will update the value of an element if and only if the element satisfies two cases:1) value for the current solution is positive ;2) the gradient for this element is negative. If we look at the second case, since α is positive, that means the new value for this element is always increased(more positive) due to the negative gradient. The first case means the element value will not be updated if the current value is already negative unless the it satisfy the second case is satisfied so that the element will become more positive. As a result, both cases prefer positive solutions, which matches our purpose due to the BCQP formulation.

A fourth key point is the various usage of the $()_+$ operator used in both the basic version and the BB version. This operator will also bias the solution toward the positive quadrant.

(a)Original

(b)Blurred


(c)Deblurred



(d)MSE vs time

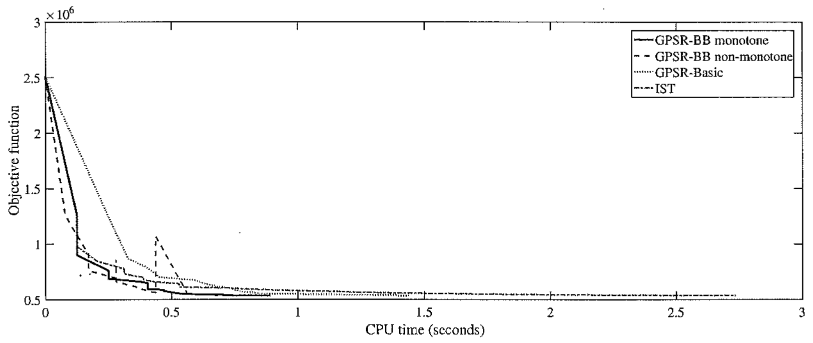


Figure 1: Experiment results for image deblurring

Comments

I ran their code package first for a image deblurring case. The input image is the cameraman. Based on Fig 2, the three approaches proposed by the paper have similar convergence rate, while the IST method takes a longer time to converge to the optimum of the objective function. We can notice that for the case of GPSR-BB non-monotone, the solution path is quite turbulent, this means the solutions can fluctuates a lot from the current iteration to the next iteration. This is very reasonable since the non-monotone case choose a constant step size of one and could well land on the bad solution along the search direction. The turbulence becomes worse then the τ is smaller, which means a higher weight on the data term(forward model). This is reasonable, since the level surfaces(surface of the same cost) for the data term is much more turbulent than the prior term, and then data term dominates the fixed step size will just cause more turbulence. The monotone approach however uses a line search to guarantee that the cost of the objective function will at least decrease throughout iterations. Another thing we can notice is that while the proposed method performs better than IST in terms of the cost of the objective function, the IST approach performs better with quite some margin in the MSE with the ground truth. This is due to the fact that ISt uses a diagonal matrix to bound the $\boldsymbol{A}^T\boldsymbol{A}$, and in the case of deconvolution the bound can be tight. As a result, although the proposed methods converges faster in terms of objective function, they might actually never converge to a prefered solution that is closer to the ground truth.

I also compared the l1-ls method with the proposed method on compressed sensing experiment. The number of observation is 1024 and the signal length is 4096. For a small τ, the l1-ls method actually outperforms the proposed methods with fixed τ. Both the GPSR methods took about 6 sec to solve the problem, while l1-ls took less than 4 sec. However, with the continuation scheme for tuning the τ parameter on the run, the GPSR methods are able to solve the problem in less than 2 sec. This almost feels like cheating considering the continuation scheme actually changes the objective function on the run. However, in practice this smart cheat might actually be prefered. For the case where τ is relatively large(signal very sparse), the GPSR methods outperforms l1-ls even without using the continuation scheme. The GPSR methods in this case take about half of the runtime of l1-ls, which is about 2 sec.

Reading Report of Fugueiredo et al.

10/10

(ECE 695, Reading Assignment 02, Spring 2018)

Overview of the Paper

The paper talks about the algorithm, Gradient Projection for Sparse Reconstruction (GPSR), to solve the famous l1-regularized least square minimization problem (1).

$$\min_{x} \frac{1}{2} \parallel \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} \parallel^2 + \tau \parallel \boldsymbol{x} \parallel_1 \qquad (1)$$

The algorithm is inspired by Gradient Projection Method. It derives the search direction, applies the line search method to find the step size, and does the iteration until it satisfies the termination criterion. The author provide two kinds of iterative algorithm, GPSR-BASIC and GPSR-BB, with detailed steps and analysis. In the experiment results, it clearly proves that the proposed algorithms outperforms the existing methods on efficiency (iteration and CPU time).

Prior Work

This paper mentions and compares with several previous approaches regarding the l1-regularized least square equation. For example, the famous solution is the least square absolute shrinkage and selection operator (LASSO), even though it's not efficiency to solve the large scale problem. The author talked about the *l1_ls* approach, the algorithm which we read in the Assignment 01. *l1_ls* approach uses Interior-Point Method to design the algorithm and applies the PCG searching direction to save the computation cost. In the experiment result, it clearly shows that the proposed algorithm in this paper does outperform those previous approaches significantly.

Key Ideas of the Paper

In this paper, the author focuses on the well-known l1-regularized, convex unconstrained optimization problem (euqation (1)) with GPSR approach, which is faster and more efficient. Through the substitution of equation (6), the equation(8) is the standard bound-constrained quadratic program (BCQP) which can represent the equation (1). The Gradient Projection Algorithm would be the good choice to solve BCQP problem.

To solve the equation (8), the GPSR approach search the negative gradient of object function as the projection direction. It performs "the backtracking line search" to find the reasonable value for the decrease in the object function. After that, it stops the iteration if the convergence test is satisfied, otherwise do the iteration. This approach is called GPSR-BASIC in this paper.

The author provides another improvement of GPSR approach, Barzilai-Borweign Gradient Projection (GPSR-BB). Barzilai-Borweign approach provide the simple approximation to calculate the search direction $\delta^{(k)}$. The GPSR-BB exploits the exact line search instead of the backtracking line search.

The termination condition is extremely important for the iteration algorithm. The well-designed termination criterion can get the good enough result without excessive calculation. The author provide four different condition for the termination criterion (equation (16), (17), (19), (20)). It doesn't analyze the performance between these termination criterion but choose the equation (17) as the one for the simulation.

Debiasing is applied after the approximate solution is derived by the proposed algorithm. It uses the CG algorithm to get the explanatory result in least-square sense. With debiasing, it can get the better solution according to the experiment results.

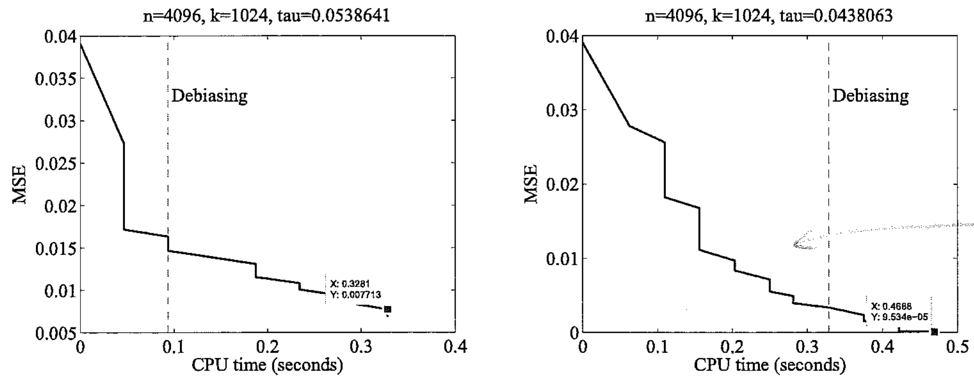
Warm Starting is important in the iterative algorithm. "Warming Starting" means that the initial point of the second iteration is the solution of the first iteration. With the selected warming starting point , the efficiency (iteration and CPU time) will be largely improved.

Comments

I tried the code package with "the signal dimension=4096" and "the number of observation=1024". The number of non-zero signal is 160 unless otherwise noted.

The termination criterion. For the iterative algorithm, it's difficult to decide whether the approximate solution is good enough without excessive compution cost. I run the simulation to compare between different termination criterion. Both case with *tolp* = 0.01. The first termination criterion is the equation (16). And the second one is the equation (19). We can see the simulation results in Figure 1. With the first termination criterion, the algorithm stops around (MSE, CPU Time)=$(0.1, 0.015)$ before debiasing and terminate at (MSE, CPU Time)=$(0.3281, 0.0077)$ after debiasing. With the second one, it stops around (MSE, CPU Time)=$(0.32, 0.0033)$ before debiasing and terminate at (MSE, CPU Time)=$(0.4688, 9.5 * 10^{-5})$ after debiasing. We can see that the second termination criterion leads to the better performance with slightly more time. The *tolp* plays an important role in the termination criterion. However, "the meaning of *tolp*" would be different between different termination criterion, so the termination criterion could only be selected based on the experiment results.

The debiasing effect. The author proposes the debiasing step which chooses the optimal value for the components according to the least-squares criterion. Here I define the ratio between Debiased MSE and Biased MSE (MSE before debiasing) and plot the figure of this ratio versus the number of non-zero signals, n_{spike}. In Figure 2, we can observe that the debiasing step can improve the performance significantly when n_{spike} is below 240 (which means sparse enough). However, the improvement due to the debiasing step drops quickly when the n_{spike} is over 240. When $n_{spike} = 300$, the ratio is almost 1. It means that the debiasing doesn't help in this scenario. I conclude that the debiasing step improves the signal reconstruction performance only when the signal is sparse enough $(\frac{n_{spike}}{n} < \frac{300}{4096})$. $\leftarrow$ Good!



Why staircase shape?

Figure 1: MSE versus CPU time with different termination criterion. (Left plot) equation 16, (Right plot) equation 19

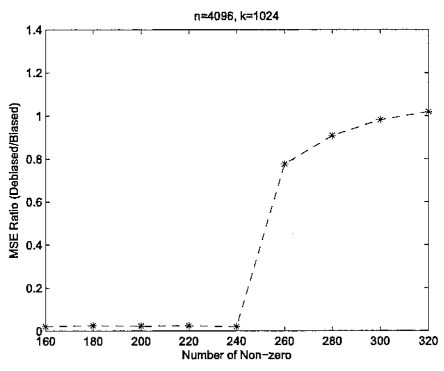


Figure 2: MSE ratio between debiased and biased signal

10/10

Reading Report of Figueiredo et al.

(ECE 695, Reading Assignment 02, Spring 2018)

Overview of the Paper

In this paper, the following l_1-regularized least square minimization problem is considered.

$$\min_x \quad \frac{1}{2}\|Ax-y\|_2^2+\lambda\|x\|_1 \tag{1}$$

To solve (1), the authors first transform it to an equivalent bound-constrained quadratic program (BCQP), and propose gradient projection (GP) algorithms to solve the BCQP. Analysis of computational complexity of each iteration of the proposed GP algorithms is provided. Finally, numerical results are provided to show that the proposed GP algorithms outperform existing ones in terms of computation time.

Prior Work

Problem (1) is convex but non-smooth, and has no closed-form solution. Hence, several algorithms have been proposed (before this work) to solve (1) or its equivalent convex problems. Specifically, homotopy algorithms, which require the evaluation of (sub-matrices) A^TA, have been proposed for solving (1) or its equivalent QP. The standard interior-point (IP) algorithm, which also require the evaluation of A^TA, can also be adopted to solve the QP formulation of (1). Since the computational complexity of evaluating A^TA is high for large-dimension problems, several algorithms have been proposed without such evaluation for solving (1) or its equivalent convex problems, which includes customized IP algorithms, majorization-minimization (MM) based algorithms, and orthogonal matching pursuit (OMP) algorithms.

Key Ideas of the Paper

In this paper, the authors derive an equivalent BCQP form of (1). For such constrained problems, pure gradient descent algorithm cannot be applied directly since it may lead to infeasible point in the next iteration. To cope with the feasibility issue, a natural idea is to adopt projection (to the feasible set) operation. Using this idea, the authors propose a GP algorithm for such problem. Since the computational complexity of such GP algorithm mainly comes from the evaluation of the objective function and its gradient, which does not require the evaluation of A^TA, its computational complexity is generally lower than the ones requiring such operation. To further reduce the computational complexity and improve performance, the authors also propose several variants of the GP algorithm. First, to speed up line search, GP algorithm with BarzilaiBorwein type approaches are adopted. Second, an additional debiasing step is proposed for some ✓ applications to further refine the solution. Finally, for *computationally hard* problems, e.g., (1) with large (BB) dimension and small τ, the authors propose a warm start approach: first solving a *computationally easy* problem which is *far away* from the original problem (e.g., large τ in (1)), and then use the solution as a starting point for solving a harder but closer problem (e.g., smaller τ in (1)) to the original one, and so on.

Comments

It is a well-written paper: the introduction clearly captures most of the existing works, and their differences from this work; the description of the proposed algorithm and the idea behind it are succinct and easy to follow; it also provides some guidelines for different applications, including the selection of terminal criterion, and the use of debiasing and warm starting. However, I think this paper could be better, and I will list my comment in the following.

First, it is not clear on the theoretical advantage of the proposed algorithm compared to state-of-the-art ones, i.e., for those do not require the evaluation of A^TA. In this paper, analysis of computational complexity of each iteration is provided, but the authors do not compare it with existing ones. Furthermore, although this paper provides numerical comparison among different algorithms, the metric (i.e., CPU time) is not very representative since the algorithms are run in MATLAB, since CPU time in MATLAB is not an accurate

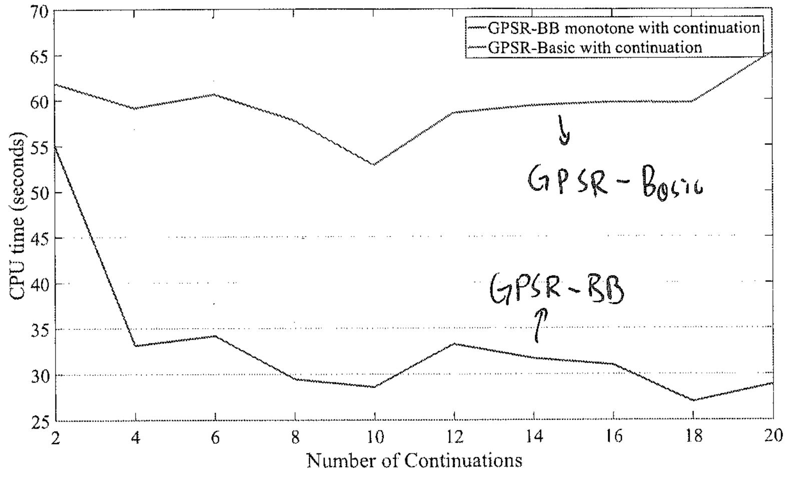


Figure 1: Average CPU time versus number of continuations for GPSR-Basic and GPSR-BB.

measure of computational complexity. It would be better to show the CPU time when implementing using C++, or flops in MATLAB.

Second, since the authors propose several variant of GP algorithms, it would be better if they provide more guideline of the selection of the algorithms, i.e., engineers working on a specific application would like to know which GP algorithm should they implement in order to have the best performance; should they apply BB methods? should they apply debiasing?

In the following, I will provide numerical results as well as discussions on a related problem I am interested in. In the paper, the authors state that for computationally hard problem, it would be better to adopt the warm starting and continuation strategy. For example, for (1) with small τ, the strategy is to start with a large τ, solve the problem and then reduce τ and use the previous solution as a the initial point of the new problem, and so on. This paper also provides numerical results to show that this strategy indeed works. However, this paper does not provide any clue on the selection of intermediate τ. For example, if we would like to solve (1) with $\tau = 0.05\|A^T y\|_\infty$ and we know (1) with $\tau = 0.1\|A^T y\|_\infty$ is easy to solve, should we solve (1) with two different τ, i.e., $\tau = 0.1\|A^T y\|_\infty$ and $\tau = 0.05\|A^T y\|_\infty$? or solve more, say six different τ, i.e., $\tau = \{0.1, 0.09, 0.08, 0.07, 0.06, 0.05\}\|A^T y\|_\infty$? In the simulation, the parameters adopted is listed in the following: A, x is generated according to the method of Section IV-D in the paper, with dimension of A, x being $1024 \times 4096, 4096 \times 1$, $y = Ax$. Target and initial τ are $0.005\|A^T y\|_\infty$ and $0.8\|A^T y\|_\infty$, respectively. The results are averaged over 10 realizations, as shown in the figure.

In the figure, we can observe the following results. First, the CPU time of GPSR-Basic, which is the proposed GP algorithm with backtracking line search, is lower than that of GPSR-BB for all number of continuations, which is GP algorithm with BB method. The above result is due to the fact that BB method greatly reduces the computational complexity of line search, no matter what τ is. Second, we can observe the existence of minimum in the figure, as expected, with the reason listed below. As number of continuations increases, the time for each continuation decreases due to warm starting from a solution corresponding to a problem with closer τ. Since CPU time is the product of the time for each continuation and the number of continuations, when the number of continuations is small, increasing number of continuations should lead to decrease of CPU time due to significant reduction of the time for each continuation. However, as the number of continuations getting larger, the marginal improvement of the time for each continuation becomes smaller, which causes the increase of CPU time. This result is more clear in GPSR-Basic (10 is the minimum point), and not less clear in GPSR-BB, because I think the increasing of CPU time may be more clear for number of continuations greater than 20.

Good discussion on warm start and run time.

10/10

Reading Report of: *Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems* by Figueiredo et al.

(ECE 695, Reading Assignment 01, Spring 2018)

Overview of the Paper

This paper deals with the efficient solution of the well known ℓ_1-regularized LS problem. The problem has a wide range of applications including sparse signal recovery in ill-/underdetermined linear systems. The authors attempt to improve upon existing algorithms by reformulating the problem as a bound-constraint quadratic program (BCQP) which requires only one level of iteration. To speed up the computation time the algorithm makes use of an approximations of the Hessian matrix of the objective function and uses a *warm start* technique to compute an array of solutions with different regularization.

Prior Work

The body relevant prior work mostly overlaps with that of `l1_ls` (HW1) and the paper emphasizes explicit comparisons to that algorithm. By referencing `l1_ls`'s superiority to existing algorithm the authors use it as a state-of-the art benchmark for comparison. Further, it mentions homotopy algorithms that are useful when the number of nonzeros is small and the overall problem size is big. Also, there were iterative shrinkage/thresholding and matching pursuit algorithms which are effective in certain problem conditions.

Key Ideas of the Paper

One of the key ideas of this paper is that the minimization problem of equation 1 in the paper is transformed into the BCQP

$$\min_{\mathbf{u},\mathbf{v}} \frac{1}{2}\|\mathbf{y}-\mathbf{A}(\mathbf{u}-\mathbf{v})\|_2^2+\tau\mathbf{1}_n^\top\mathbf{u}+\tau\mathbf{1}_n^\top\mathbf{v} \tag{1}$$
$$\text{s.t.} \quad \mathbf{u}\geq\mathbf{0} \quad \text{and} \quad \mathbf{v}\geq 0$$

where the solution to the problem is $\mathbf{x}=(\mathbf{u}-\mathbf{v})$. Using the vector $\mathbf{z}^\top=[\mathbf{u}^\top\mathbf{v}^\top]$ yields the property that the constraint can simply be maintained by projecting a proposed update direction onto the positive orthant. That is, if $\mathbf{z}_0$ satisfies the constraints then $\mathbf{z_1}=[\mathbf{z}_0+\lambda(\mathbf{d})_+]$ also satisfies it for any vector $\mathbf{d}$ and any positive scaler, λ. The authors present two algorithms that differ in the choice of the expressions for $\mathbf{d}$ and λ.

The GPSR-Basic algorithm uses as an initial guess the step size that minimizes the unconstrained objective function from equation 1 in the negative gradient direction. Then the final step size and direction is found using projection on the positive orthant and a backtracking line search approach.

In the GPSR-BB algorithm the update is determined using a variant of the BARZILAI-BORWEIN approach. Here, the update pattern vaguely resembles the NEWTON method while approximating the Hessian as a multiple of the identity.

Another key aspect of the paper is the use of a *continuation* method which enables the fast computation of the solution even when the regularization parameter, τ, is small or to obtain the solution for an array of τ's. The authors show how by solving the problem for initially large τ's, the estimates, $\mathbf{z}$, can be used as an initial starting points for gradually smaller values of τ. In practice the intermediate estimates are often useful by themselves, however the continuation technique may often even speed up the computation in comparison to directly aiming for the small τ.

Comments

In my experiments I was particularly interested in the properties of the *continuation* method. Also, I was interested in how the GPSR algorithm performs in comparison with the `l1_ls` algorithm from the previous homework. For this I used the simple `demo_continuation.m` example as a basis. I needed to ensure that the GPSR and `l1_ls` terminated at approximately the same MSE, thus I modified the `l1_ls` code so that it uses the ground truth to compute the MSE after each iteration while artificially terminating the solvers in post-analysis.

As a metric of comparison I chose the CPU time required to reach the same, final MSE. To consider a wide range of problems I stuck with the given problem size ($2^{10} \times 2^{12}$) while changing the number of nonzero ± 1 spikes in the unknown signal, $\mathbf{x}$. I also changed the regularization parameter $\tau = \{0.1, 1, 10\}\, \tau_0$ where $\tau_0 = 0.005 \left\|\mathbf{A}^\top y\right\|_\infty$. I only focus on the GPSR-BB algorithms as the results were similar to the GPSR-Basic ones.

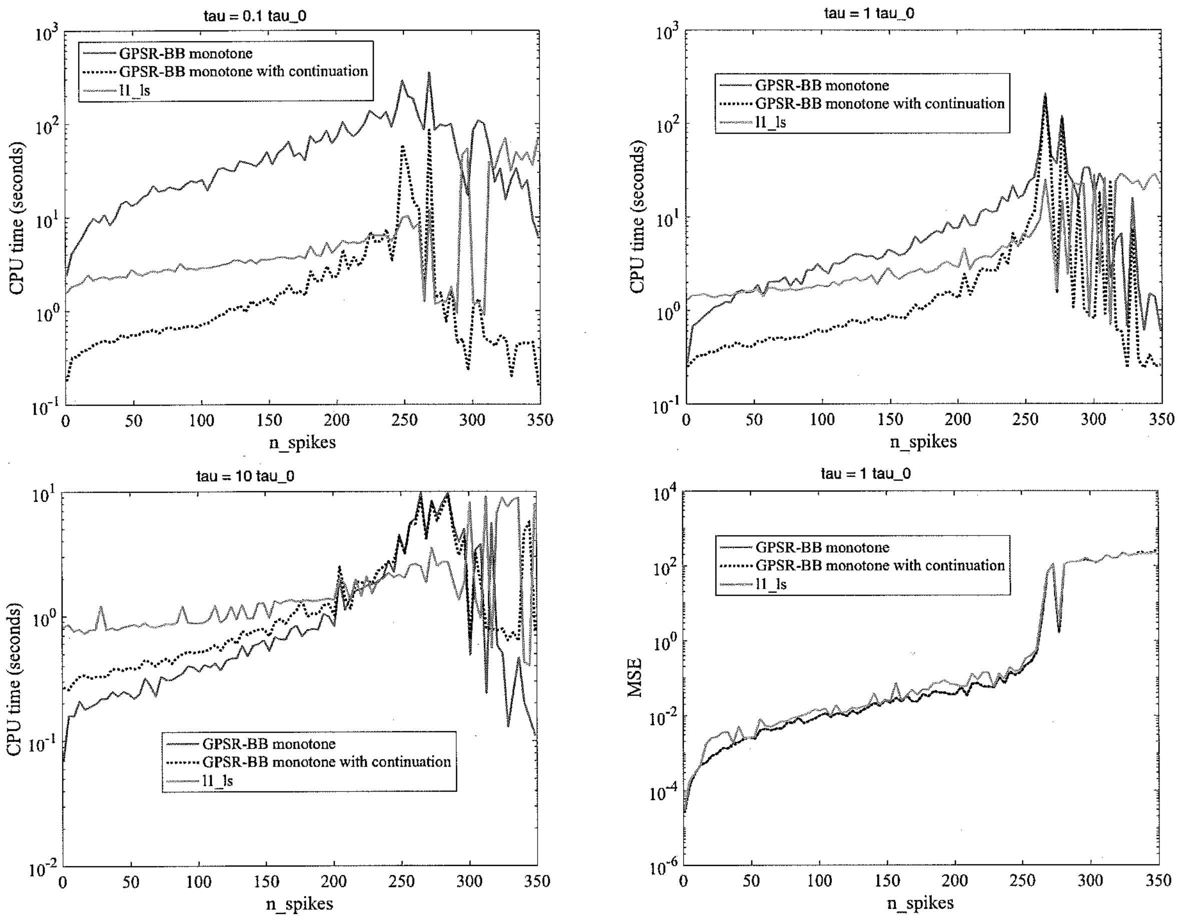


Figure 1: CPU time for different values of τ and MSE for the mid τ value over the number of nonzero spikes

Figure 1 shows the plots of the CPU time till convergence for 3 different τ's and the MSE of the algorithms for the mid τ at the time they were stopped.

Some Observations:

very good observation

As the number of nonzero values grows, the MSE grows smoothly resulting in moderate error until a sort of critical number of nonzeros is reached (about at 250 spikes) beyond which the error is 1 to 2 orders of magnitude higher. At that point the convergence times of either of the 3 algorithms is unpredictable (compare the 2 graphs with $\tau = \tau_0$) when given a problem of similar size but different instances of the random signals and matrix. In these cases the values of τ is simply too small to result in a good prediction of the unknown.

When focusing on the parts below the critical number of nonzeros ($\approx$ 10 to 200) the GPSR algorithms seem to increase in CPU time faster than the `l1_ls` algorithm (slope of the CPU time graphs). At a low number of nonzeros the `l1_ls` may be slower then the GPSR algorithm whereas at a higher number of nonzeros it may be faster than the GPSR-BB algorithm and comparable to GPSR-BB-continued algorithm. Further, at high τ's the GPSR-BB algorithm is fastest and the `l1_ls` slowest while at low τ's the GPSR-continued algorithm is fastest and the GPSR algorithm slowest.