# BFGS based quasi-Newton method for acceleration of MM algorithm

Medha Agarwal and Jason Xu

September 2, 2020

**Abstract**

MM and consequently, EM algorithms are a powerful optimization technique due to both - its simplicity and stability. However, in many statistical applications, they suffer from slow convergence, especially when the dimension of parameter space is high. There has been some work for accelerating the MM algorithm using quasi-Newton methods. These methods attempt to update the approximation for the Jacobian of the algorithm map at each iteration. However, most of them are either too problem-specific or rely on certain conditions on starting values or MM update formula to give acceptable convergence. We propose a quasi-Newton method for accelerating MM algorithms based on the popular BFGS optimization method. However, we are not interested in the optimization problem of the objective function. Our algorithm targets to find the fixed point of the MM algorithm using no specific information about the objective function or its gradient whatsoever. We also propose an LBFGS based algorithm using the same update formulation of Jacobian matrices. The performance of both the algorithms is numerically examined and compared to other quasi-Newton acceleration schemes for MM.

## 1 Introduction

MM algorithm is a powerful iterative method for computation of maximum likelihood estimates. It is attractive because of two reasons: (1) MM algorithms are computationally simple to implement and (2) they are stable (monotonous increase in likelihood) (see Dempster et al. (1977); Laird (1978) and references thereof for discussion on its properties and relevant examples). MM algorithm is much simpler to execute than its alternatives like gradient-based methods (Newton-Raphson) and Fisher score method. A widely discussed drawback with MM algorithms is its slow linear convergence, especially when the amount of missing data is large. (see Wu (1983); Boyles (1983); Meng and Rubin (1994) for discussion on convergence) and dependence on initial value.

I like to give a gentler overview of the motivation and some approaches. The more detailed background with equations etc can be included later in a prior work section. Here is a

start: To address this shortcoming, there have been a number of works that design acceleration schemes for MM and EM algorithms. Broadly speaking, these methods seek additional information to better inform the search direction and/or step lengths of the unadorned algorithm. Often such information comes from second-order terms, and it is necessary to balance a tradeoff for additional computational cost. Some approaches directly work with the objective function that the MM algorithm originally seeks to optimize— in the case of EM, this is the observed data log-likelihood. [Citations to Lange and Jamshidian papers]. Approximate second order information can be obtained via Fisher scoring in the case of EM. Outside of the context of missing data, classical tools from the numerical optimization literature such as quasi-Newton and conjugate gradient methods can be applied to the objective to similar effect.

These methods may be limited when it is nontrivial to carry out operations such as differentiation with respect to the original objective, and typically rely on handling and storing the approximate Hessian which becomes costly for high-dimensional problems. Indeed, an advantage of MM algorithms lies in sidestepping an intractable objective in favor of operating on surrogates. An alternative approach is to apply acceleration schemes on the sequence of iterates produced by the MM algorithm itself—in other words, to consider acceleration of the MM *algorithm map* instead, largely ignoring the optimization objective. While these often preserve the simplicity and low computational complexity of the original algorithm, they are less firmly rooted in optimization theory.

[Overview the methods that do so but mention that in many cases they are more ad hoc and do not fully/formally leverage the wisdom in the classical literature and are instead loosely inspired by it.]

Here I would "close the introduction" to simply state our contributions and give a very high level description of our proposed approach, contextualizing it in the "methodological gap" that has been roughly outlined prior. Some things we can highlight is by bridging it to Broyden's method, we get convergence guarantees [hopefully], a way to incorporate additional secant conditions (like Hua's method), efficient low-memory variations, and an explicit decomposition of search direction and step length information. In practice we find that it performs similarly if not better than existing simpler methods in terms of actual runtime, and as suggested by the theory is more robust/reliable across a broad range of settings. Then with that road map in mind the reader can later be exposed to the finer details.

What you have written here has lots of good information. Eventually we may split it up so that some references come in the introduction above, some contextualizes relations to existing work after we present our method, and the rest can become part a more specific background/literature review section to follow the introduction. There has been wide discussion on methods for accelerating the EM algorithms in statistical community. Varadhan and Roland (2008) suggested a binary classification for EM accelerators - (1) monotone and (2) non-monotone acceleration schemes. Monotone accelerators target to increase the likelihood like EM algorithm. Prominent examples are parameter expansion EM (Liu et al. (1998)), Expectation Conditional Maximization (ECM) (Meng and Rubin (1993)), and ECME which

shares the monotone convergence properties of EM and ECM while offering a faster convergence (Liu and Rubin (1994)). Non-monotone methods tend to find the root of an objective function, for example, $G(x) = F(x) - x$ where $F(x)$ is the fixed point algorithm map. These methods employ EM iterative schemes to get closer to the root of an objective function which gives the required maximum likelihood estimates. The most widely used and cited method for EM acceleration is the multivariate Aitken's acceleration suggested by Louis (1982)

$$\Delta\theta_n = -(J - I)^{-1} g(\theta_n) \tag{1}$$

where $\Delta\theta_n$ is the acceleration step, $g(\theta_n)$ is the current step and $J$ denotes the approximation to the Jacobian of the EM step. The method proposed by Louis (1982) to find $(J - I)$ is essentially the Newton Raphson method to find the zero of $g(x)$ (see discussion by Laird et al. (1987); Meilijson (1989)). Other examples of monotone accelerators are conjugate gradient (CG) method (Jamshidian and Jennrich (1993)) and quasi Newton methods (Jamshidian and Jennrich (1997); Lange (1995)). The recent acceleration methods STEM and SQUAREM developed by Varadhan and Roland (2008) presents a different category which approximates the Newton's method using the idea of Steffenson's two point secant appoximation for the Jacobian. They build a vector form of this method for STEM and apply an idea of 'squaring' to obtain SQUAREM.

The monotone methods are fast and retain the ascent property of EM algorithms. However, they are highly dependent on the intrinsic model properties and hence cannot be broadly applied. Non-monotone methods are generally applicable but lack on the part of storage and globalised convergence. These methods store the Hessian of the algorithm map or require the observed information matrix which limits their use in high dimensional cases. They depend on the starting values and therefore, show poor results when started 'badly'. The SQUAREM requires only the algorithm map for EM and offer great results even for high dimensional problems. Following the work by Varadhan and Roland (2008), Zhou et al. (2011), built a quasi-Newton acceleration technique that retains the simplicity of SQUAREM. We will call this method ZAL (Zhou Alexander Lange) for future comparisons. As pointed out in Zhou et al. (2011), we are interested in finding the root of the equation $F(x) - x = 0$ unlike the previous quasi Newton methods (Jamshidian and Jennrich (1997); Lange (1995)) that require optimization of the objective function.

Both SQUAREM and ZAL do not approximate the full Hessian for optimizing the objective function using Newton's method due to storage and computational complications. ZAL is based on the assumption that we start near the fixed point whereas SQUAREM is based on the assumption that the Jacobian is of the form $\alpha_n I_p$ where $\alpha_n$ is a scalar and $I_p$ is a p-dimensional identity matrix. We propose a quasi Newton acceleration for EM algorithms that instead attempts to approximate the Jacobian of the equation $G(x) = F(x) - x$ for finding its root using Newton's method. The altered approach advocated by ZAL that targets to find the root of $G(x)$ helps us in ignoring the conditions of positive definiteness and makes the algorithm more robust against numerical instabilities. Our method differs from ZAL in

the quantity that is minimized given the same constraint. ZAL minimizes the Jacobian of $F(x)$ near the fixed point. This implies that every iterate for Jabobian approximation only depends on the current state and the algorithm map.

Our update for the inverse Jacobian approximation takes inspiration from the classical BFGS approach for quasi Newton method. We minimize the spectral norm of the difference between consecutive inverse Jacobian approximations. Our quasi Newton update formula is similar to SQUAREM and ZAL with an added advantage of BFGS robustness for finding the root of $G(x)$. This method surpasses both the assumptions of SQUAREM and ZAL.

In high dimensional cases where the number of variables may go up to an order of thousands or millions, the storage costs can be overwhelming. This can limit the application of BFGS algorithms to low dimensional cases only. Limited memory quasi Newton algorithms have been studied by many authors like memory-less quasi-Newton methods (Shanno (1978)) and limited memory BFGS method (Nocedal (1980)). Another concept of partitioned updating of Hessian matrix by introduced by Griewank and Toint (1982). Limited memory Hessian approximations can also be used as pre-conditioners in conjugate gradient methods (see Buckley and LeNir (1983); Gill and Leonard (2003)). Numerical supremacy of L-BFGS has been established by Liu and Nocedal (1989). They are simple to implement and do not require an understanding of the structure of Hessian matrix.

# 2 Quasi-Newton methods for MM acceleration

Building on work established by Varadhan and Roland (2008) and Zhou et al. (2011), we will use a Newton-type method to accelerate the convergence of an MM algorithm map to its fixed point. The popular BFGS update formula is a quasi-newton method that works towards optimizing a function by approximating its Hessian at each step. Consider the MM algorithm map $F : \mathbb{R}^p \to \mathbb{R}^p$. Instead of solving the parent optimization problem, we are interested in finding the roots of the objective function $G(x) = F(x) - x$. This altered approach of finding the root of $G(x)$ using quasi-Newton method requires one to only update the Jacobian approximation of $G(x)$ at each step instead of a Hessian matrix.

The Steffenson type EM (STEM) method suggested by Varadhan and Roland (2008) achieves this approximation using a scalar multiple of identity matrix. This scalar can be understood as the steplength for each update. The same steplength is used to arrive at the SQUAREM method. While SQUAREM outperforms many acceleration methods and is chiefly regarded for its simplicity, the loss of information due to approximation of Jacobian by an identity matrix can be severe, especially in high dimensional cases. We propose approximating the full Jacobian matrix at each iterate.

However, storage and calculation of the full $p \times p$ dimensional Jacobian at each iteration can be a computationally challenging task which will limit its application for a high-dimensional problem. To surpass this limitation, we will also present an L-BFGS based

quasi-newton algorithm which does not require to store the full dense $p \times p$ inverse Jacobian matrix. Instead, one is only required to store some pre-specified $m$ numbers of iterations to approximate the inverse Jacobian. In the next two subsections, we present both the algorithms and their theoretical comparison to other quasi-Newton algorithms like SQUAREM and ZAL.

## 2.1 BFGS based quasi-Newton method

Our method begins with a simple core idea: first, denote the algorithm map $F$, i.e. $F : x_k \mapsto x_{k+1}$ where $x_k$ is the $k$th iterate of the unaccelerated MM algorithm. An MM algorithm seeks to optimize some objective function, but can equivalently be expressed without reference to that objective as seeking the fixed points of the algorithm map. That is, we seek roots of the equation $G(x) = F(x) - x$. The observation allows us to employ quasi-Newton acceleration to $G$. The iterative update formulation using Newton's method is given by:

$$x^{n+1} = x^n - (dF(x^n) - I)^{-1} G(x^n)$$

Let $H^n$ be an approximation of $(dF(x^n) - I)^{-1}$, then the quasi-Newton update formula is given by

$$x^{n+1} = x^n - H^n G(x^n)\,.$$

The secant approximation required for quasi Newton method can be obtained by using two consecutive updates from the algorithm map starting from the current position. We will follow the same secant approximation recommended by Zhou et al. (2011), where the function $G(x)$ us assumed to be linear between $x_t$ and $F(x_t)$ with the inverse Jacobian given by $H_{t+1}$

$$H^{t+1}\left[G(F(x_t)) - G(x_t)\right] = F(x_t) - x_t. \tag{2}$$

Let us define $v_t = G(F(x_t)) - G(x_t) = F^2(x_t) - 2F(x_t) + x_t$ and $u_t = F(x_t) - x_t$, then the secant requirement is $H^{t+1} v_t = u_t$. I edited this section, but here we might add a clearer transition to emphasize the fact that one needs to specify additional information in order for the solution to be well-defined rather than underdetermined, and in particular we will instead make use of the objective that allows us to recover Broyden's method for accelerated nonlinear root-finding.

We propose a quasi Newton update which optimizes the next iterate of Jacobian, such that $\|H_{t+1} - H_t\|_F$ is minimum and also, the secant approximation constraint is met. In fact, for the best results we require several secant approximations $H_t(u_t^i - v_t^i) = u_t^i$ for $i \in \{1, ..., q\}$. These can be generated at the current iterate $x^t$ and previous $(q-1)$ iterates. Let $U_t = (u_1\ u_2 \dots u_q)$ and likewise $V_t = (v_1\ v_2 \dots v_q)$ be two $\times q$ matrices. The optimization problem becomes,

$$\begin{aligned} &\text{Minimize: } \|H_{t+1} - H_t\|_F \\ &\text{Constraint: } H_{t+1} V_t = U_t \quad . \end{aligned}$$

We take the partial derivative of the Lagrangian

$$\mathcal{L} = \frac{1}{2}\|H_{t+1} - H_t\|_F^2 + \Lambda^T(H_{t+1}V_t - U_t)$$

with respect to $h_{t+1}^{ij}$ and equate to 0. Here $h_{t+1}^{ij}$ denotes the $ij^{th}$ element of the matrix $H_{t+1}$. As a consequence, we get the Lagrange multiplier equation:

$$0 = h_{t+1}^{ij} - h_t^{ij} + \sum_{k=1}^{p} \lambda_{ik} v_{jk}$$

In matrix notation, we have,

$$H_{t+1} - H_t + \Lambda V_t^T = 0 \tag{3}$$

Post multiply equation 3 by $V_t$ and put $H_{t+1}V_t = U_t$

$$H_{t+1}V_t - H_tV_t + \Lambda V_t^T V_t = 0$$

This implies

$$\Lambda = (H_tV_t - U_t)(V_t^T V_t)^{-1}\,.$$

Therefore,

$$H_{t+1} = H_t\left(I - V_t(V_t^T V_t)^{-1}V_t^T\right) + U_t(V_t^T V_t)^{-1}V_t^T.$$

Although as the problem dimension increases, higher $q$ should give tigher convergence to the fixed point, however numerically we might face singularity for the matrix $V_t^T V_t$. The special case of $q = 1$ is the most successful where:

$$H_{t+1} = H_t - H_t\frac{v_t v_t^T}{v_t^T v_t} + \frac{u_t v_t^T}{v_t^T v_t}\,. \tag{4}$$

Equation 4 can be written as $H_{t+1} = H_t + A_t + B_t$ where both $A_t$ and $B_t$ are rank-1 matrices. Therefore, as expected Equation 4 is a rank-2 update. The symmetry condition on $H_t$ assumed in classical BFGS update can be dropped here because it approximates a Jacobian matrix instead of a Hessian matrix.

**Intuition and relation to prior work** The search direction at $t^{th}$ iteration is given by $p_t = -H_tG(x_t)$ and the update formula is

$$x_{t+1} = x_t + \gamma_t p_t$$

where $\gamma_t = \alpha_t/\|p_t\|$ is an appropriate scaling factor in the search direction and $\alpha_t$ is the steplength. Notice that the steplength for non-accelerated MM algorithm is $\|F(x_t) - x_t\| = \|u_t\|$. Suppose we use the steplength $\|H_t\|\|G(x_t)\|$ in the search direction $p_t = -H_tG(x_t)$. If $H_t$ is sufficiently close to $H_{t+1}$, using Equation 2 we can lowebound $\|H_t\|$ by $\|u_t\|/\|v_t\|$. We

will use this lowerbound of steplength giving $\alpha_t = \|u_t\|^2/\|v_t\|$. The SQUAREM algorithm provides three different possible steplengths along the direction $u_t$. Notice that our choice of $\alpha_t$ coincides with the third steplength. Likewise, we can construct $\alpha_t$ corresponding to the first and second steplengths of SQUAREM. In this paper, we will only focus on the $\alpha_t$ explicitly mentioned above. We can further flesh out this subsection, and emphasize exactly why this is more general than the SQUAREM methods and how it sort of unifies existing methods, and also why it is potentially a better formulation than ZAL. We should close this with a discussion on the drawback that our formulation, while closer to established wisdom, leads to a Jacobian that can be unwieldy, and the design of ZAL is more immediately applicable to high-dimensional problems. This then transitions into the next section

## 2.2 L-BFGS based quasi-Newton method

Our algorithm requires storage of $H_t$ before performing a rank-two update on it by Equation 4. The cost of storing the $n \times n$ inverse Hessian matrix for each update can limit the application of this method for high dimensional problems. Many limited memory quasi-Newton algorithms have been proposed (see Shanno (1978); Nocedal (1980); Griewank and Toint (1982)). Here we will focus on L-BFGS method by extending our BFGS based quasi-Newton accelerator to the L-BFGS analogue using the same Jacobian update formulation. Recall that $H_t$ is updated by the formula

$$H_{t+1} = H_t \left( I - \frac{v_t v_t^T}{v_t^T v_t} \right) + \frac{u_t v_t^T}{v_t^T v_t} = H_t W_t + . \frac{u_t v_t^T}{v_t^T v_t}$$

where $W_t = \left( I - v_t v_t^T / v_t^T v_t \right)$ and the BFGS update is given by $x_{t+1} = x_t - H_t(F(x_t) - x_t)$. Similar to the L-BFGS method, we will store previous $m$ (usually between 3 to 20) pairs of $\{u_i, v_i\}$, $i = t-1, ..., t-m$. The product responsible for update at each step, i.e. $H_t G(x_t)$ is obtained by performing a sequence of inner products and vector summations involving $G(x_t)$ and the pairs $\{u_i, v_i\}$. After the new iterate is computed, the oldest vector pair $\{u_{(t-m)}, v_{(t-m)}\}$ is replaced by the new pair $\{u_{t+1}, v_{t+1}\}$ obtained from the current step. In this way, the set of vector pairs includes curvature information from the $m$ most recent iterations.

An initial estimate of inverse Hessian at $t^{th}$ step is considered to be a scalar multiple of identity matrix. Let us denote this by $H_t^0$. This initial approximation can vary from iteration to iteration, unlike standard BFGS method. Following this $H_t^0$ is updated $m$ times via Equation 4 in a nested manner to obtain the following relation

$$
\begin{aligned}
H_t = & H_t^0(W_{t-m}...W_{t-1}) \\
& + \frac{u_{t-m}v_{t-m}^T}{v_{t-m}^T v_{t-m}}(W_{t-m+1}...W_{t-1}) \\
& + \frac{u_{t-m+1}v_{t-m+1}^T}{v_{t-m+1}^T v_{t-m+1}}(W_{t-m+2}...W_{t-1}) \\
& + ... \\
& + \frac{u_{t-1}v_{t-1}^T}{v_{t-1}^T v_{t-1}}.
\end{aligned}
$$

Let $H_t^0$ be written as $\nu_t I$ where $\nu_t$ is a scalar that varies from iteration to iteration. Chapter 6 from Nocedal and Wright (2006) explain the effectiveness of using

$$\nu_t = \frac{u_t^T v_t}{v_t^T v_t}.$$

As discussed there, $\nu_t$ is the scaling factor that attempts to estimate the size of the true Hessian matrix along the most recent search direction. One can observe that STEM method of EM acceleration by Varadhan and Roland (2008) is similar L-BFGS method described above in the sense that it corresponds to L-BFGS with $m = 0$. However it is worth to note that they arrived at the approximation of inverse Hessian matrix as $\nu_t I$ by minimizing the distance between the zeros of two linear secant-like approximations for $G(x)$; one centered around $x_t$ and another at $F(x_t)$.

### 2.3 Properties

Add remark that this respects/preserves any linear constraints that may have originally been enforced. Add convergence results or highlight existing theorems that can apply under appropriate assumptions

## 3 Examples and Performance

We will compare five methods for the examples below: (1) non-accelerated MM method, (2)ZAL for $q = 2$, (3) SQUAREM (all three variates) (4) BFGS based QN method, and (5) L-BFGS based QN method. Both ZAL and SQUAREM offer scalar approximations for the Jacobian method. As a consequence, they are pretty fast to implement. For a high-dimensional problem, BFGS based QN method offer extremely slow convergence. However, we are interested in ascertaining the fast convergence of BFGS method for a moderately dimensional problem, and study the performance of L-BFGS as compared to BFGS method for high dimensional cases. We observe striking results for our algorithms in most of the examples. At other times, they perform as good as the other aforementioned acceleration

algorithms. The termination criterion for each example is to stop at $x^*$ such that $\|F(x^*) - x^*\| \leq \epsilon$ where $\epsilon$ is fixed at $10^{-7}$ for each example.

### 3.0.1 Truncated Beta Binomial

| Household | Number of cases | | | | MLE | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | $\hat{\pi}$ | $\hat{\alpha}$ |
| (a) | 15 | 5 | 2 | 2 | 0.0000 | 0.6151 |
| (b) | 12 | 6 | 7 | 6 | 0.1479 | 1.1593 |
| (c) | 10 | 9 | 2 | 7 | 0.0000 | 1.6499 |
| (d) | 26 | 15 | 3 | 9 | 0.0001 | 1.0594 |

Table 1: The Lidwell and Somerville (1951) cold data on households of size 4 and corresponding MLEs under the truncated beta-binomial model

We consider the cold incidence dataset by Lidwell and Sommerville (1951) and fit a zero-truncated beta binomial model because the households reported have at least one cold incidence. Table 1 summarizes the dataset where the households were classified as: (a) adults only; (b) adults and school children; (c) adults and infants; and (d) adults, school children, and infants. Suppose $n$ is the total number of independent observations (households) and $x_i$ denotes the number of cold cases in the $i^{th}$ household. This is a discrete probability model with likelihood given by

$$L(\theta|X) = \prod_{i=1}^{n} \frac{g(x_i|\theta)}{1 - g(0|\theta)}\,.$$

Here $g(x|\theta)$ is the probability density function for a beta binomial distribution with parameter vector $\theta$ and maximum count of $m = 4$. Here $\theta = (\alpha, \pi)$ such that $\pi \in (0, 1)$ and $\alpha > 0$. The MM algorithm updates are given by

$$\alpha_{t+1} = \frac{\sum_{k=0}^{m-1}(\frac{s_{1k}k\alpha_t}{\pi_t + k\alpha_t} + \frac{s_{2k}k\alpha_t}{1 - \pi_t + k\alpha_t})}{\sum_{k=0}^{m-1} \frac{r_k k}{1 + k\alpha_t}}$$

$$\pi_{t+1} = \frac{\sum_{k=0}^{m-1} \frac{s_{1k}\pi_t}{\pi_t + k\alpha_t}}{\sum_{k=0}^{m-1}(\frac{s_{1k\pi_t}}{\pi_t + k\alpha_t} + \frac{s_{2k}(1 - \pi_t)}{1 - \pi_t + k\alpha_t})}$$

where $s_{1k}$, $s_{2k}$, $r_k$ are pseudo counts given by

$$
\begin{aligned}
s_{1k} &= \sum_{i=1}^{n} 1_{x_i \geq k+1} \\
s_{2k} &= \sum_{i=1}^{n} \left[ 1_{x_i \leq m-k-1} + \frac{g(0|\pi_t, \alpha_t)}{1 - g(0|\pi_t, \alpha_t)} \right] \\
r_k &= \sum_{i=1}^{n} \left[ 1 + \frac{g(0|\pi_t, \alpha_t)}{1 - g(0|\pi_t, \alpha_t)} \right] 1_{t \geq k+1} \, .
\end{aligned}
$$

Table 2 lists the final negative log-likelihood, number of MM evaluations, number of algorithm iterations, and running times until convergence for each algorithm is achieved for all the four cases. The starting point is $(\pi, \alpha) = (0.5, 1)$. In the last three cases, SQUAREM-2 fails to converge to the truth. It can be observed that BFGS with $q = 2$ provides significant improvement over $q = 1$ case for the third and fourth case. Except for second case, BFGS with $q = 2$ either outperforms or performs at least as good as other acceleration schemes. Figure 1 shows the progress path of all the algorithms on a contour plot for the first data type. Notice that in this case, our algorithm provides the fastest convergence.

### 3.0.2 Generalized eigenvalues

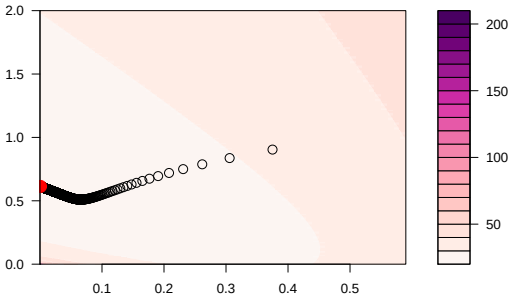
Because of the zigzag nature of steepest ascent, naive acceleration performs poorly. If $x_{n+1} = F(x_n)$ is the algorithm map, the Naive and Zhou's method is calculated using $F_s(x)$ as the algorithm map which is its s-fold functional composition. This substitution preserves the ascent property. Table 3 shows the results of accelerating two-step (s = 2) steepest ascent and steepest descent.

### 3.0.3 Multivariate t-distribution

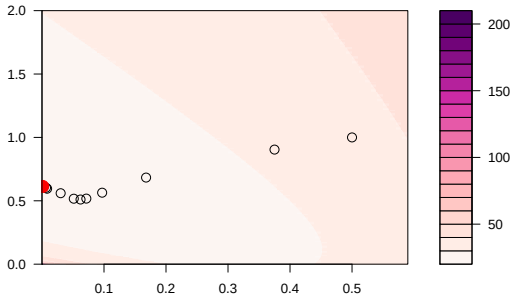
Here we use a slightly high dimensional problem and compare the performance of BFGS based quasi Newton acceleration scheme to other quasi newton methods discussed before. This example has been used by Varadhan and Roland (2008) to compare their SQUAREM algorithm to standard EM and PX-EM (essentially the efficient data augmentation method implemented by Meng and Van Dyk (1997)). We will also call their method PX-EM. The reason why their data augmentation method fits into parameter expansion paradigm can be seen in the discussion section of Varadhan and Roland (2008).

Suppose we have p dimensional data $Y = (y_1, ..., y_N)$ and we wish to fit a multivariate t distribution of unknown degrees of freedom. The density of t-distribution with $\nu$ degrees of freedom is given by

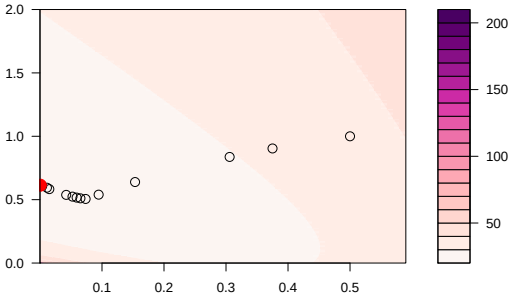
$$
f(y|\mu, \Sigma) \propto |\Sigma|^{-1/2} \left( \nu + (y - \mu)^T \Sigma^{-1} (y - \mu) \right)^{(\nu+p)/2}
$$



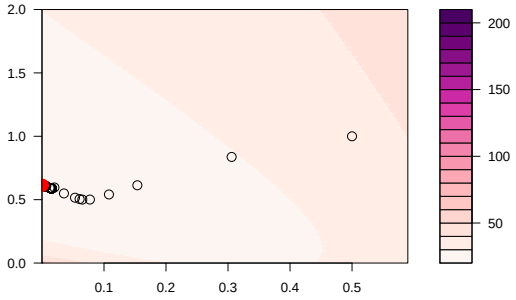

(a) MM



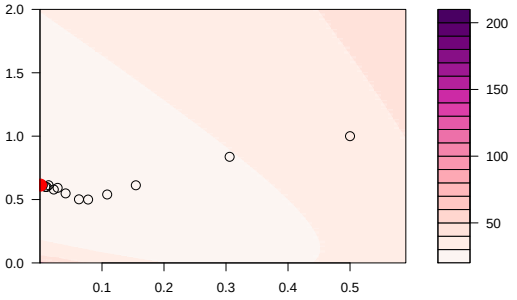

(b) BFGS, $q = 1$



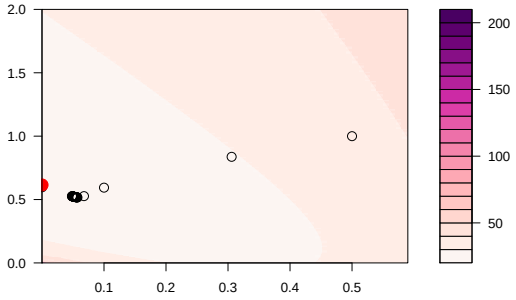

(c) BFGS, $q = 2$




(d) SqS1




(e) SqS3




(f) ZAL, $q = 1$

Figure 1: Truncated Beta Binomial. Ascent of different methods for Somerville household type (a) data. Starting points are $(\pi_0, \alpha_0) = (0.5, 1)$. The tolerance is $\epsilon = 10^{-7}$.

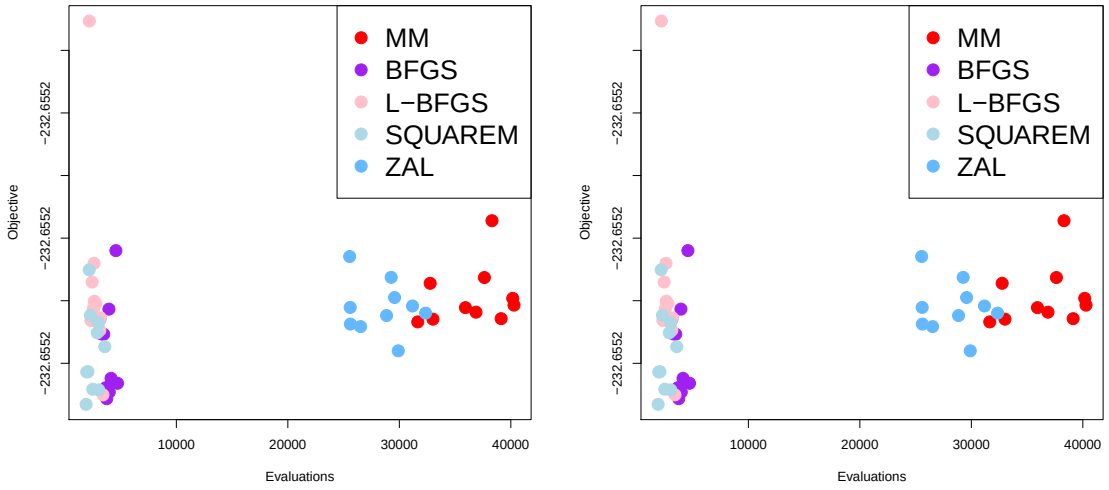
| Data | Algorithm | -ln L | Fevals | Iterations | Time |
|---|---|---|---|---|---|
| (a) | MM | 25.2282 | 17898 | 17989 | 2.647 |
| | BFGS, $q = 1$ | 25.2287 | 26 | 14 | 0.043 |
| | BFGS, $q = 2$ | 25.2276 | 29 | 16 | 0.039 |
| | SqS1 | 25.2282 | 1343 | 661 | 0.208 |
| | SqS2 | 25.2280 | 51 | 18 | 0.0279 |
| | SqS3 | 25.2282 | 54 | 19 | 0.0289 |
| | ZAL, $q = 1$ | 25.2270 | 66 | 33 | 0.035 |
| (b) | MM | 41.7286 | 5492 | 5492 | 0.748 |
| | BFGS, $q = 1$ | 41.7286 | 928 | 470 | 0.270 |
| | BFGS, $q = 2$ | 41.7286 | 1107 | 555 | 0.316 |
| | SqS1 | 41.7286 | 254 | 115 | 0.039 |
| | SqS3 | 41.7286 | 99 | 34 | 0.027 |
| | ZAL, $q = 1$ | 41.7286 | 524 | 262 | 0.105 |
| | ZAL, $q = 2$ | 41.7286 | 593 | 297 | 0.141 |
| (c) | MM | 37.358 | 61843 | 61843 | 5.891 |
| | BFGS, $q = 1$ | 37.358 | 1870 | 936 | 0.624 |
| | BFGS, $q = 2$ | 37.358 | 63 | 33 | 0.061 |
| | SqS1 | 37.358 | 2799 | 1388 | 0.406 |
| | SqS3 | 37.358 | 120 | 41 | 0.035 |
| | ZAL, $q = 1$ | 37.358 | 1292 | 646 | 0.495 |
| | ZAL, $q = 2$ | 37.358 | 60 | 32 | 0.036 |
| (d) | MM | 65.042 | 25026 | 25026 | 2.797 |
| | BFGS, $q = 1$ | 65.043 | 268 | 135 | 0.117 |
| | BFGS, $q = 2$ | 65.041 | 57 | 30 | 0.087 |
| | SqS1 | 65.041 | 1197 | 587 | 0.200 |
| | SqS3 | 65.042 | 60 | 21 | 0.029 |
| | ZAL, $q = 2$ | 65.040 | 49 | 25 | 0.031 |

Table 2: Comparison of algorithms for the Lidwell and Somerville Data. The starting point is $(\pi, \alpha) = (0.5, 1)$, the stopping criterion is $\epsilon = 10^{-7}$, and the number of parameters is two.

Therefore the likelihood for the observed data is $\prod_{i=1}^{N} f(y_i|\mu, \Sigma)$. We need to find $(\mu, \Sigma)$ which maximize the likelihood. There is no general closed form solution for the MLE, but if we augment y such that $y_comp = \{(y_i, q_i); i = 1, ..., N\}$, where q are i.i.d. from $\chi^2_\nu/\nu$, then the MLE follows from a weighted least-squares procedure. The E-step finds the expectation of the complete-data log-likelihood, conditionally on Y and $(\mu_n, \Sigma_n)$ from the previous iteration. The missing data $q_i$ conditionally on Y, and $(\mu_n, \Sigma_n)$ is distributed as: $q_i \sim \chi^2_{\nu+p}/(\nu + d_i^{(n)})$ where $d_i^{(n)} = (y_i - \mu_n)^T \Sigma_n^{-1} (y_i - \mu_n); i = 1, ..., N$. As the complete-data log-likelihood is

| Algorithm | Largest Eigenvalue | | Smallest Eigenvalue | |
|---|---|---|---|---|
| | Evals | Time | Evals | Time |
| MM | 29654 | 13.9217 | 33822 | 23.2785 |
| BFGS | 1966 | 2.6628 | 3016 | 4.7921 |
| LBFGS | 2022 | 2.1103 | 2557 | 2.9211 |
| SqS1 | 2571 | 1.3733 | 1971 | 1.7423 |
| SqS2 | 2571 | 1.5673 | 1971 | 1.9398 |
| SqS3 | 2571 | 1.3833 | 1971 | 1.6266 |
| ZAL | 19077 | 13.7742 | 23875 | 17.7076 |

Table 3: Generalized eigenvalues. Number of $F(x)$ evaluations and running times random matrices A and B of dimension $100 \times 100$. Here the stopping criterion is $\epsilon = 10^{-7}$, and the number of parameters is 100. The starting vector is randomly chosen.




(a) Objective vs Number of F evaluations

(b) Objective vs Time

Figure 2: Generalized eigenvalues. Convergence value of objective vs no. of function evaluations scatter plot for $N = 10$ replications. The stopping criteria is $\epsilon = 10^{-7}$. Lower objective is better

linear in $q_i$, the E-step is:

$$w_i = E[q_i|y_i, \mu_n, \Sigma_n] = (\nu + p)/(\nu + d_i^{(n)}); \qquad i = 1, ..., N$$

The M-step yields:

$$\mu_{n+1} = \sum_i w_i y_i \Big/ \sum_i w_i$$
$$\Sigma_{n+1} = \frac{1}{N} w_i (y_i - \mu)(y_i - \mu)^T$$

| Algorithm | Evals | Time | -ve Likelihood |
|---|---|---|---|
| EM | 560.5 (485.0, 619.0) | 3.131 (2.814, 4.507) | -6230.447 |
| PX-EM | 44 (43, 49) | 0.095 (0.079, 0.138) | -6230.447 |
| ZAL | 637.5 (60.0, 834) | 1.508 (0.202, 2.808) | -6230.447 |
| BFGS, $q = 1$ | 104 (64, 174) | 0.474 (0.270, 1.709) | -6230.447 |
| BFGS, $q = 2$ | 178 (125, 263) | 0.811 (0.552, 1.223) | -6230.447 |
| L-BFGS | 72 (60, 93) | 0.373 (0.236, 0.678) | -6230.447 |
| SqS1 | 63 (53, 93) | 0.175 (0.119, 0.324) | -6230.447 |
| SqS2 | 60 (48, 84) | 0.134 (0.095, 0.234) | -6230.447 |
| SqS3 | 64 (48, 84) | 0.141 (0.097, 0.196) | -6230.447 |

Table 4: Multivariate t-distribution. Parameter estimation of a 30-dimensional multivariate t-distribution, with 1 degree of freedom, $\mu = 0$, and a randomly generated covariance matrix. The problem dimension is 495. Minimum, median, and maximum for 10 such simulations.

The PX-EM method of Meng and Van Dyk (1997) differs only in update of $\Sigma$ by replacing the denominator N by $\sum_i w_i$. We will use a randomly generated data for $\nu = 1$ (multivariate Cauchy distribution) with parameters $\mu = 0$ and $\Sigma = V$ where V is a random symmetrical randomly generated matrix. We will also use $p = 10$ which accounts for 65 parameters (10 for $\mu$ and 55 for $\Sigma$). We report the results for 100 simulations with randomly generated data and use the starting values suggested by Meng and Van Dyk (1997).

$$\mu_0 = \frac{1}{N} \sum_{i=1}^{N} y_i$$
$$\Sigma_0 = \frac{1}{N} \sum_{i=1}^{N} (y_i - \overline{y})(y_i - \overline{y})^T$$

### 3.0.4 Landweber's method for quadratic minimization

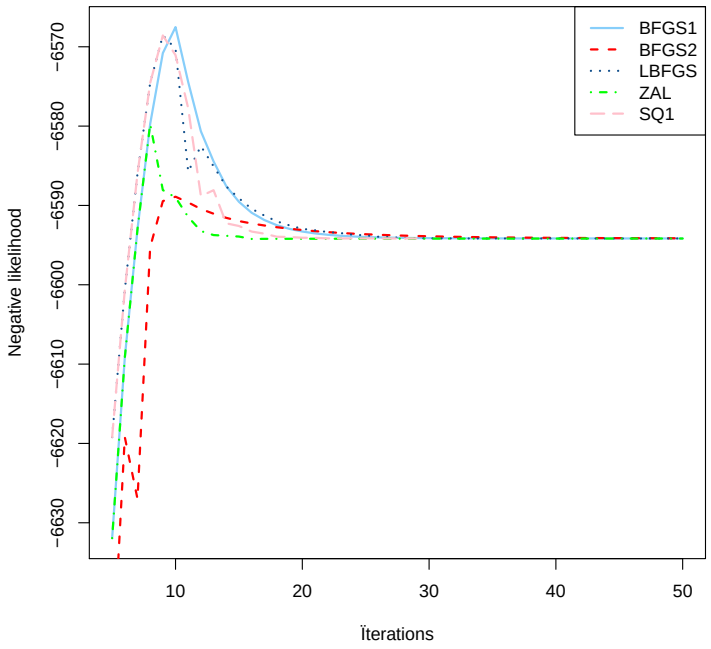



Figure 3: Multivariate t-distribution. Running plot of negative likelihood vs number of iterations need to fix plot so that likelihood is only decreasing with iteration?

If we do derive some theory and we assume conditions such as Lipschitz, then we may move this example up first and say that it is meant to confirm the theory in practice, i.e. we observe superlinear rate compared to the unaccelerated algorithm.

In this example, we deal with a simple problem of minimizing a quadratic function $f(\theta)$ using an MM iterative scheme. Landweber's method identifies a maximizing function for the quadratic equation using the Lipchitz property of gradient of $f(\theta)$. Suppose the quadratic function is of the form

$$f(\theta) = \frac{1}{2}\theta^T A\theta + b^T\theta$$

where $A$ is a $p \times p$ positive definite matrix. The exact solution is available by solving the linear equation $A\theta = -b$. This operation has an order of complexity of $\mathcal{O}(p^3)$. Therefore, we will explore the iterative scheme explained in Lange (2016). We know that $\nabla f(\theta) = A\theta + b$. Therefore, we can write the gradient inequality

$$\|\nabla f(\theta) - \nabla f(\Phi)\| = \|A(\theta - \Phi)\| \leq \|A\|\|\theta - \Phi\|$$

As a consequence, spectral norm of A is the Lipchitz constant for $\nabla f(\theta)$. Consider $L > \|A\|$ to be the Lipschitz constant for $\nabla f(\theta)$, then Lendweber's method gives the following maximization for $f(\theta)$

$$f(x) \leq f(y) + df(y)(x - y) + \frac{L}{2}\|x - y\|^2$$

Minimizing the above maximizer gives the following update formula

$$\theta_{n+1} = \theta_n - \frac{1}{L}\nabla f(\theta_n) = \theta_n - \frac{1}{L}(A\theta_n + b)$$

We consider a high dimensional problem with $p = 1000$. The starting value for all the replications are

| Algorithm | Fevals | Time | Likelihood |
|---|---|---|---|
| MM | 2290 | 0.529 | -0.6427 |
| BFGS1 | 106 | 0.009 | -0.6427 |
| BFGS2 | 35 | 0.004 | -0.6427 |
| LBFGS | 25 | 0.001 | -0.6427 |
| SqS1 | 60 | 0.002 | -0.6427 |
| SqS2 | 210 | 0.006 | -0.6427 |
| SqS3 | 72 | 0.001 | -0.6427 |
| ZAL | 231 | 0.20 | -0.6427 |

Table 5: Quadratic Minimization. Minimizing the quadratic function $f(\theta) = \theta^T A\theta/2 + b^T\theta$. The number of parameters is 5 and the stopping criterion is $\epsilon = 10^{-7}$. The starting value is generated randomly.

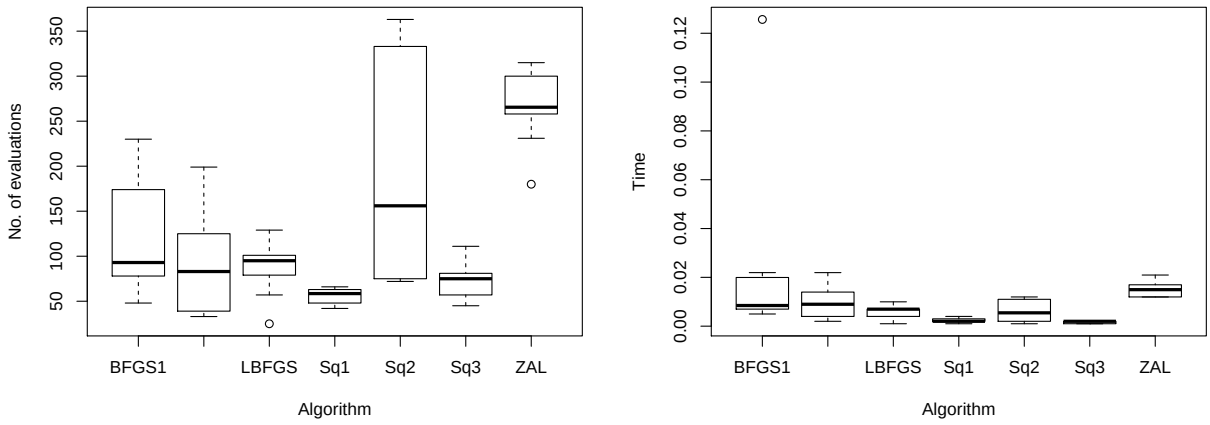



Figure 4: Quadratic minimization. Boxplots for objective value, number of updates, and time taken for all the compared algorithms.

# References


Boyles, R. A. (1983). On the convergence of the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 45(1):47–50.

Buckley, A. and LeNir, A. (1983). QN-like variable storage conjugate gradients. *Mathematical programming*, 27(2):155–175.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Gill, P. E. and Leonard, M. W. (2003). Limited-memory reduced-Hessian methods for large-scale unconstrained optimization. *SIAM Journal on Optimization*, 14(2):380–401.

Griewank, A. and Toint, P. L. (1982). Partitioned variable metric updates for large structured optimization problems. *Numerische Mathematik*, 39(1):119–137.

Jamshidian, M. and Jennrich, R. I. (1993). Conjugate gradient acceleration of the EM algorithm. *Journal of the American Statistical Association*, 88(421):221–228.

Jamshidian, M. and Jennrich, R. I. (1997). Acceleration of the EM algorithm by using quasi-newton methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(3):569–587.

Laird, N. (1978). Nonparametric maximum likelihood estimation of a mixing distribution. *Journal of the American Statistical Association*, 73(364):805–811.

Laird, N., Lange, N., and Stram, D. (1987). Maximum likelihood computations with repeated measures: application of the EM algorithm. *Journal of the American Statistical Association*, 82(397):97–105.

Lange, K. (1995). A quasi-Newton acceleration of the EM algorithm. *Statistica sinica*, pages 1–18.

Lange, K. (2016). *MM optimization algorithms*. SIAM.

Lidwell, O. and Sommerville, T. (1951). Observations on the incidence and distribution of the common cold in a rural community during 1948 and 1949. *Epidemiology & Infection*, 49(4):365–381.

Liu, C. and Rubin, D. B. (1994). The ECME algorithm: a simple extension of EM and ECM with faster monotone convergence. *Biometrika*, 81(4):633–648.

Liu, C., Rubin, D. B., and Wu, Y. N. (1998). Parameter expansion to accelerate EM: the PX-EM algorithm. *Biometrika*, 85(4):755–770.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Louis, T. A. (1982). Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 44(2):226–233.

Meilijson, I. (1989). A fast improvement to the EM algorithm on its own terms. *Journal of the Royal Statistical Society: Series B (Methodological)*, 51(1):127–138.

Meng, X.-L. and Rubin, D. B. (1993). Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80(2):267–278.

Meng, X.-L. and Rubin, D. B. (1994). On the global and componentwise rates of convergence of the EM algorithm. *Linear Algebra and its Applications*, 199:413–425.

Meng, X.-L. and Van Dyk, D. (1997). The EM algorithm——an old folk-song sung to a fast new tune. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(3):511–567.

Nocedal, J. (1980). Updating quasi-Newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782.

Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Shanno, D. F. (1978). Conjugate gradient methods with inexact searches. *Mathematics of operations research*, 3(3):244–256.

Varadhan, R. and Roland, C. (2008). Simple and globally convergent methods for accelerating the convergence of any EM algorithm. *Scandinavian Journal of Statistics*, 35(2):335–353.

Wu, C. J. (1983). On the convergence properties of the EM algorithm. *The Annals of statistics*, pages 95–103.

Zhou, H., Alexander, D., and Lange, K. (2011). A quasi-Newton acceleration for high-dimensional optimization algorithms. *Statistics and computing*, 21(2):261–273.