Sparse Hessian Factorization in Curved Trajectories for Minimization

ALBERTO J JIMÉNEZ

The Curved Trajectories Algorithm (CTA) for the minimization of unconstrained functions of several variables follows polynomial space curves at each step. The space curves result from truncations of a Taylor series expansion of the solution. A critical ingredient in the efficiency of CTA is the factorization of the sparse hessian matrix and handling a non positive hessian. This paper describes a new approach for non-positive hessians that has given outstanding robust results that compare very favorably with current state of the art minimization algorithms using a sample of the Cuter problems.


Additional Key Words and Phrases: Variable order convergence, Curved trajectory search, Accurate third and fourth derivative terms, Robust convergence to minimum.

1. INTRODUCTION

We focus on the problem:

\[ \text{Minimize } f(x), \quad f : \mathbb{R}^n \to \mathbb{R} \quad (1) \]

Where \( f \) is a nonlinear continuous real scalar function, with all derivatives continuous, of \( n \) variables represented by the vector \( x \). We write the derivatives as follows:

\[ f'(x), \quad f''(x), \quad f'''(x), \quad f^{(4)}(x), \quad \text{etc} \]

Derivative \( f'(x) \) is called the Gradient, an \( n \times 1 \) vector, \( f''(x) \) is called the Hessian, an \( n \times n \) symmetric matrix; and the higher derivatives are vectors of matrices and matrices of matrices, etc.; and all are symmetric (Clairut’s Theorem applies to the continuous functions considered). A minimum point \( x^* \) is a critical point of the function, which means we look for a solution to the equation of the Gradient being the zero vector:

\[ f'(x) = 0 \quad (2) \]

A necessary condition to be a minimum is that the Hessian \( f''(x) \) must be positive semi definite at the critical point (e.g. the Hessian has no negative eigenvalues at \( x^* \)).
otherwise the critical point is not a minimum. A minimum point may not be a global minimum. This paper addresses calculating a local minimum point.

Typical methods for approximating the minimum start from a guess \( x_k \), and a direction \( v_k \), selected such that directional derivative is negative along \( v_k \). The sub-problem to solve at each step is the scalar (line) search:

\[
\text{Minimize } f(x_k + pv_k)
\]

This scalar minimization is a well known problem to solve (see for examples Himmelblau [1972] Section 2.6 and Kincaid and Cheney [2002] Chapter 11). There are well known strategies for approximating the solution. The scalar search yields \( p_k \) and the next guess of the solution is then \( x_{k+1} = x_k + p_kv_k \), and the iterations continue until the Gradient is as close to zero as desired, or there is lack of progress. Convergence (see for example Luenberger [1965] page 125, or Kincaid and Cheney [2002] page 33) requires sufficient descent at each step (in addition to other requirements), that is:

\[
f(x_{k+1}) = f(x_k + p_kv_k) < f(x_k)
\]

2. BRIEF DESCRIPTION OF CTA ALGORITHM

The CTA algorithm (see Jiménez [2008]) adds higher order terms to the Newton method. The higher order terms are derived by expanding the solution, or the inverse function, in a Taylor series, and retaining up to two additional terms after the Newton term. The CTA algorithm then selects the number of terms to use at each iteration, depending on tests for convergence of the series terms. The net result is that at each step of the algorithm:

\[ x_{k+1} = x_k + h(p_k) \]

Where \( h(p) \) is a vector function representing the direction CTA selects at the step, and \( p_k \) is calculated to insure (4) is satisfied. The forms of the vector function are given by:

<table>
<thead>
<tr>
<th>Order</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd</td>
<td>( h(p) = -pd_2 )</td>
</tr>
<tr>
<td>3rd</td>
<td>( h(p) = -\frac{p}{2}(3-p)d_2 - p^2d_4 )</td>
</tr>
<tr>
<td>4th</td>
<td>( h(p) = -\frac{1}{6}p(p^2 - 6p + 11)d_2 - p^2(2-p)d_4 - p^3d_4 )</td>
</tr>
</tbody>
</table>
Where the vectors $d_2$, $d_3$, and $d_4$ are calculated from the Hessian and Gradient at $x_k$, and from Gradients on the neighborhood of $x_k$ by solving the linear systems:

$$\left[f^*(x_k)\right]d_2 = f'(x_k)$$  \hspace{1cm} (5)

$$\left[f^*(x_k)\right]d_3 \approx \frac{1}{\epsilon^2}\left[f'(x_k - \epsilon d_2) + (\epsilon - 1)f'(x_k)\right]$$  \hspace{1cm} (6)

$$\left[f^*(x_k)\right]d_4 \approx \frac{1}{\epsilon^2}\left[f'(x_k - \epsilon^2 d_2 - \epsilon^2 d_3) + (\epsilon - 1)f'(x_k)\right]$$  \hspace{1cm} (7)

When $3^{rd}$ order or $4^{th}$ order is selected, the vector function is a polynomial curved trajectory, or space curve, for $p \geq 0$ with $h(0) = 0$. This algorithm converges rapidly when near the solution when the hessian is positive semi-definite. When far from the solution, the hessian can be negative definite and the trajectory may not be a descent direction. A recent paper (see Jiménez [Oct 2008]) describes adding rotations to CTA when far from the solution resulting in reducing the number of hessians needed. In this paper we describe the factorization of the Hessian, handling it when stored sparsely, and when it is negative definite.

3. FACTORIZATION OF THE HESSIAN

Effective Newton direction based methods use a modified factorization of the Hessian (see for example Nocedal and Wright [2006]). One of the earliest, and fairly effective, modifications to the symmetric Cholesky factorization is described by Murray [1972]. The idea is to never calculate the Newton direction, $d_2$ of equation (5), using a Hessian that is negative semi-definite by making positive any diagonal that becomes very small or negative during the factorization. Jimenez [1976] added strong diagonal pivoting to Murray’s method to achieve good results. Gill, Murray and Wright [1981] also describe adding diagonal pivoting to this method. More recently, Schnabel and Eskow [1991] describe a new modified method based on Gershgorin eigenvalue disk estimates that achieved encouraging results.

The new method described here was discovered because the CTA algorithm calculates Taylor series terms that achieve theoretical properties when the unmodified Hessian is used in equations (5) to (7). This means that initially, the Hessian is not modified when negative diagonal terms are encountered during the factorization. Only diagonal elements close to zero are set to a small constant (default is $10^{-14}$) to avoid singularities. This
approach means a strong diagonal pivoting Gaussian factorization is used since the square root present in Cholesky factorization is not possible.

The implications of this factorization would seem daunting. The new approach means losing some the advantages of the symmetrical Hessian, since the factorization is no longer symmetric. Fortunately, the CTA algorithm recovers this disadvantage by a reduction in the total work required to find the solution. It is surprising how well this factorization change coupled with modifications of the \( d_2, d_3, \) and \( d_4 \) correction terms work as described in the rest of this paper.

3. CTA FACTORIZATION OF THE HESSIAN

Diagonal pivoting is employed resulting in an integer vector called \( E \) in the software that encodes a pre– and post–permutation matrix. Each step of the factorization selects the strongest diagonal defined to be the one with the smallest ratio of the absolute value of the largest off-diagonal element divided by the diagonal. Negative diagonals are clipped at \( \beta \) so that they are not selected too early. This selection reduces the elements growing in value as Gaussian elimination proceeds. This pivoting strategy was compared with the selection of the largest diagonal at each step; and it made very significant difference, especially for poorly scaled problems. While negative diagonal elements are not changed during this initial factorization, a beta constant is calculated as described by Murray [1972] with the Jimenez [1976] modification. An initial beta is calculated as follows:

\[
\beta = \max(10^{-14}, \min(\sqrt{l_{\text{ij}}}, \text{weakii}) )
\]

where \( l_{\text{ij}} \) is the largest absolute value element, and \( \text{weakii} \) is the weakest diagonal measured by the diagonal minus the sum of all the absolute value off-diagonals of its row. The value of \( \beta \) is updated at each row step during the factorization. A vector \( D \) is also determined. The vector \( D \) represents the diagonal matrix that would be added to the Hessian in order to make it positive definite; \( \beta \) plays a role in this vector being the positive value that a negative diagonal would become in the event that a positive definite Hessian is needed. Using the vector \( D \) and \( \beta \), the factorized Hessian may be fixed—modified to be positive definite—if a step does not achieve descent.

After calculating the correction vectors \( d_2, d_3, \) and \( d_4 \) using equations (5) to (7), the CTA algorithm may modify these vectors when the Hessian is not positive definite. Each
modification strategy is labeled for reference. The rows which have negative diagonals are identified with a non-zero entry in the D vector described above.

1. **NOFLIP**: Leave the correction vectors alone. This can be a descent direction and in some cases leads to significant descent.

2. **NEGFLIP**: Flip the sign of entries that correspond to non-zero D vector entries. We interpret this strategy shortly. It often is the best descent direction.

3. **POSFLIP**: Flip the sign of entries that correspond to zero D vector entries. Sometimes this strategy works well when NEGFLIP fails.

4. **OPPFLIP**: Flip the sign of all entries. Rarely is the opposite direction the best.

5. **FORCEPOS**: Fix the Hessian factorization so that it is positive definite. This strategy is very similar to the original Murray [1972] Hessian modification.

If the Hessian is positive definite, then the CTA algorithm proceeds well. When the Hessian is not positive definite, then the strategy selection is as follows:

If the trajectory points to no descent, try OPPFLIP. If this direction provides insufficient descent or the trajectory points in a descent direction initially, try NEGFLIP, then POSFLIP, NOFLIP, and finally FORCEPOS in that order. CTA also can select different orders at each step in its effort to explore a significant descent at each step.

To visualize the interpretation of each of these strategies, consider the two variable example of Rosenbrock’s [1960] problem commonly known as the banana-shaped valley function:

\[ f(x, y) = 100(y - x^2)^2 + (1 - x)^2 \]  

At the point:

\[ x_0 = [x, y]^T = [0.111613, 0.158564]^T, \quad \text{solution is } [1, 1]^T \]

The Hessian is:

\[
\begin{bmatrix}
-41.1136 & -46.2301 \\
-46.2301 & 200
\end{bmatrix}
\]

After factorization, the D vector is \([65.9419, 0]^T\). This Hessian has eigenvalues -49.6737 and 208.56; and of course this is a negative definite matrix. The CTA algorithm selects order 3 at this step, and the correction vectors are:

\[ d_2 = \begin{bmatrix} 0.034148 \\ 0.142393 \end{bmatrix}, \quad d_3 = \begin{bmatrix} 0.035466 \\ 0.007032 \end{bmatrix} \]

The trajectory vector function is given by:
The effect of the FORCEPOS strategy is to change the Hessian by \textit{adding} the diagonal matrix:

\[
\begin{bmatrix}
65.9419 & 0 \\
0 & 0
\end{bmatrix}
\]

The effect of the NEGFLIP, POSFLIP, and OPPFLIP strategies are to \textit{multiply} the Hessian by the matrices:

\[
\begin{bmatrix}
-1 & 0 \\
0 & 1
\end{bmatrix}, \quad
\begin{bmatrix}
1 & 0 \\
0 & -1
\end{bmatrix}, \quad \text{and} \quad
\begin{bmatrix}
-1 & 0 \\
0 & -1
\end{bmatrix}
\]
respectively. Given the relative uncertainty depending on parameters in the calculation of beta, as described by Murray [1972] and others, the relative precision of the other strategies is attractive. The well established and desirable goal is to leave as much of the Hessian undisturbed when encountering a negative definite point. The NEGFLIP and POSFLIP strategies, for example, change the Hessian into skew-symmetric matrices with identical absolute values; but the modified Hessian is not guaranteed positive definite, and thus CTA implements the FORCEPOS strategy as a last recourse. The positive eigenvalues are changed in small ways since the Trace of the matrix stays the same.

When FORCEPOS is needed, the factorization is fixed starting with the row where the first negative diagonal is located, thus saving operations compared to starting from the first row.

3. THE HESSIAN STORED IN SPARSE FORM

As documented in the earlier implementation of CTA (Jimenez [2008]), CTA is very efficient for problems of less than 500 variables, when compared with several of the state of the art algorithms for unconstrained minimization. One advantage of CTA is solving very difficult problems with poor scaling. CTA has just been extended to exploit a sparse Hessian; and as the results of the next section show, CTA now handles large problems just as well as the small ones.

The Hessian is expected in component form, and only the diagonal and the upper or lower off-diagonals are supplied. The Hessian is then stored with a row index vector, and each row column index vector parallel to the element values in a floating point vector. The factorization can pivot as before for the strongest diagonal, which is best for poorly scaled problems. CTA also includes the approximate minimum degree (AMD) ordering algorithm of Amestoy, Davis and Duff [1996, and 2004]. For large, not poorly scaled problems, this ordering minimizes the number of fill-ins during the factorization, and results in the fastest times. A heuristic has been added to switch to strong diagonal pivoting when off-diagonal growth is large, and progress towards the solution slows down.

While CTA has a growth check on off-diagonal values during the factorization, and switches to strong diagonal pivoting when the growth becomes large and progress is
slow, this may come too late for some poorly scaled problems. An example illustrates the differences. Consider the 1,000 variable version of the poorly scaled and nearly singular problem SCOSINE in the Cuter environment (more on Cuter tests later). Table 1 shows the impact of strong diagonal pivoting on this problem. Strong diagonal pivoting can be requested by setting a parameter in CTA.

<table>
<thead>
<tr>
<th>Pivoting Strategy</th>
<th>NF</th>
<th>NG</th>
<th>NH</th>
<th>F</th>
<th>GNORM</th>
<th>CODE</th>
<th>SEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD</td>
<td>1467</td>
<td>153</td>
<td>6</td>
<td>-995</td>
<td>1.307E-04</td>
<td>2</td>
<td>0.532</td>
</tr>
<tr>
<td>Growth Check</td>
<td>2275</td>
<td>237</td>
<td>11</td>
<td>-999</td>
<td>2.495E-07</td>
<td>0</td>
<td>1.328</td>
</tr>
<tr>
<td>Strong Diagonal</td>
<td>3603</td>
<td>340</td>
<td>6</td>
<td>-999</td>
<td>3.669E-07</td>
<td>2</td>
<td>1.016</td>
</tr>
</tbody>
</table>

Using AMD on the SCOSINE problem of Table 1 to come up with an ordering that minimizes fillins during the factorization process keeps the original 0.3% number of non-zeros the same. Using strong diagonal pivoting to determine the factorization order results in as high as 0.48% number of non-zeros towards the ending iterations. The full Hessian results with Strong Diagonal pivoting require over 16 seconds.

3. RESULTS FOR LARGE PROBLEMS FROM CUTER GOULD ET AL. [2003]

UNCONSTRAINED PROBLEMS

The Curved Trajectories Algorithm (CTA) has been implemented to run in the Cuter environment, and is available from the author for academic use. CTA solves all the error-free unconstrained problems, including ones with interval constraints on the variables. Stopping criteria has been selected to achieve solutions as good as published results, typically with Gradient norms less than $5 \times 10^{-4}$. Table 2 summarizes the results run in a Dell pc, Windows XP, running VMware with Ubuntu using the gnu C development environment. The column labeled CODE is returned by CTA; it is a judgment on the solution found, with 0 being a local minimum, 1 a minimum on a boundary, 2 probably a minimum of a poorly scaled problem, and 5 ending large gradient but probably a minimum of a poorly scaled problem. There are six more returned codes explained in the Readme file. As shown below, the CTA package performs well when compared with other packages, especially in robustness; that is, calculating a solution to all the problems.
We compared results with the Lancelot (Conn, et al. [2000]) package, the CG Descent (Hager and Zhang [2005]) package, and the L-BFGS-B (Byrd, Lu and Nocedal [1995]). CTA was the only package to solve the very difficult and poorly scaled SBRYBND,
SCOSINE, SCURLY10, SCURLY20, and SCURLY30. CTA was also about as fast or faster than the other packages for the problems all packages solved.

Performance Profile graphs show performance comparisons well, since they take into consideration problems not solved correctly. Dolan and Moré [2002] describe these graphical cumulative distributions of performance graphs. CPU-time is a good performance measure, with other measures being numbers of Gradients or any other counter. Figure 2 is the CPU-time profile graph. A package is deemed to have succeeded if the function value is within 10% of its correct value, or the Gradient is less than $5 \times 10^{-4}$. This strategy allows for a package to solve for different local minima.

![CPU-time Profile (51 problems >= 500 variables)](image)

Figure 2. CPU-time Performance Profile on 51 problems with more than 499 variables.

Note this profile has logarithm scale on the x-axis. The interpretation of this graph is as follows: the first bin says that CTA solved about 38% and L-BFGS solved about 24% of the problems as fast as or faster than the other packages tested. CTA flattens at higher percent values because all of the 51 problems were solved correctly, and the other packages solved from 69% to 80% of the problems.
7. CONCLUSIONS

This paper presents a new implementation for handling a negative semi-definite sparsely stored Hessian on the highly efficient CTA algorithm. It has been implemented in the already very efficient CTA algorithm, and has made the CTA package competitive for large problems, and kept its robust ability to solve even very poorly scaled problems.

REFERENCES


Received November 2008
Figures 4 through 6 show Performance Profiles of the counters.

Figure 4  Number of Function evaluations Performance Profile on 51 problems with 499 variables or more.

Figure 5  Number of Gradient evaluations Performance Profile on 51 problems with 499 variables or more.
Figure 6  Number of Hessian evaluations Performance Profile on 51 problems with 499 variables or more.