Rotations in Curved Trajectories for Unconstrained Minimization

Alberto J Jimenez

Mathematics Department, California Polytechnic University, San Luis Obispo, CA, USA 93407

Abstract Curved Trajectories Algorithm (CTA) is a package for the minimization of unconstrained functions of several variables with intervals on the variables. The core algorithm is novel in that steps may follow polynomial space curves instead of straight lines. The space curves result from truncations of a Taylor series expansion of the Gradient inverse function. When the series is convergent and the current guess of the solution is far, appropriate rotations of the space curve may allow further progress at a step. Improvements are significant for some functions, reducing the number of Hessians required or improving the accuracy of the solution. This idea may be useful to other minimization packages.

Keywords Curved Trajectory Minimization, Rotations of Space Curves, Unconstrained Optimization.

1. Introduction

We focus on the problem:

Minimize $f(x)$, $f: \mathbb{R}^n \rightarrow \mathbb{R}$, (1)

where $f$ is a nonlinear continuous real scalar function, with continuous derivatives, of $n$ variables represented by the vector $x$. We write the derivatives simply by:

$f'(x)$, $f''(x)$, $f'''(x)$, $f^{(4)}(x)$, etc.

The first derivative $f'(x)$ is called the Gradient, which is an $n \times 1$ vector function; $f''(x)$ is called the Hessian, which is an $n \times n$ symmetric matrix. The higher derivatives are tensors of higher dimension. A minimum point $x^*$ is a critical point of the function satisfying:

$f'(x^*) = 0$. (2)

A necessary condition for 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 only.

Typical methods for calculating a minimum start from a guess $x_0$. Then a direction $v_k$ is selected at the $k^{th}$ iteration step such that the directional derivative is negative along $v_k$. Now a search along this direction is made to find a suitable value of $p > 0$ that yields $f(x_k + pv_k) << f(x_k)$. One can approximate $p$ by solving the one-dimensional scalar problem, a line search:

Minimize $f(x_k + pv_k)$. (3)

This line search can require significant computation time and is usually unnecessary. Instead, a limited number of trial $p$ values are tested until one is selected that produces sufficient reduction in the function value. This scalar minimization is a well known problem (see for example [3] Section 2.6, or [7] Chapter 11). The scalar search yields $p_k$ and the next guess of the solution is $x_{k+1} = x_k + p_k v_k$ and the iterations continue until the Gradient is as close to zero as desired, or there is lack of progress. Recent work establishes a Trust Region (see [11] or [1]) for calculating $x_{k+1}$ using a quadratic model of the function. Convergence (see for example [8] page 125, or [7] page 33) requires sufficient descent at each step (in addition to other requirements), that is:

$f(x_{k+1}) = f(x_k + p_k v_k) << f(x_k)$. (4)

2. Brief Description of CTA Algorithm

A core component of the CTA optimization package is the addition of higher order terms to Newton's method (see [5] and [6]). The higher order terms are derived by expanding the unknown minimum $x^*$ equal to the Gradient inverse function in a Taylor series, and retaining up to two additional terms after the Newton term. The derivation begins from

$f'(x_k) = z_k^T \Rightarrow x_k = (f'(x_k)^{-1}) (z_k^T)$. (5)

After differentiation and much tedious algebra, the first four terms of the infinite Taylor series expanded at $z_k$ and evaluated at zero giving $x^* = (f'(x^*)^{-1})(0)$ are
\[ x^k = x_0 - \left[ p d_1 \right] + \frac{1}{2} p(p - 1)d_1 - \frac{1}{6} p(p - 1)(p - 2)d_1 - \cdots, \]

where the vectors \( d_2, d_3, \) and \( d_4 \) are approximated from the Hessian and Gradient at \( x_0 \), and from Gradients in the neighborhood of \( x_0 \) by solving the linear systems:

\[ f'(x_0) d_2 = f'(x_0), \]
\[ f'(x_0) d_3 = \frac{1}{2} f'(x_0) d_1 d_2, \]
\[ f'(x_0) d_4 = \frac{1}{6} f'(x_0) d_1 d_2 d_3, \]

The constant \( \varepsilon \) is a small positive constant. This results in an iteration, and at each step a \( p_k > 0 \) is selected such that

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

satisfies (4), where \( h_k(p_k) \) is a vector function representing the trajectory direction selected at the \( k^{th} \) step. The number of terms to use at each iteration depends on tests for convergence of the series. The forms of the vector function are given by

- \( 2^{nd} \) order trajectory
  \[ h(p) = -p d_2 \quad \text{(Newton’s Method)} \]
- \( 3^{rd} \) order trajectory
  \[ h(p) = -(0.5)p(3 - p)d_2 - p^2 d_3 \]
- \( 4^{th} \) order trajectory
  \[ h(p) = -p \left( p^2 - 6p + 11 \right)d_2 / 6 - p^2 \left( 2 - p \right)d_1 - p^3 d_4 \]

When the \( 3^{rd} \) order or \( 4^{th} \) order is selected, the vector function \( h(p) \) is a polynomial curved trajectory, or space curve, for \( p > 0 \) with \( h(0) = 0 \). This algorithm converges rapidly in the neighborhood of the solution when the Hessian is positive semi-definite. When far from the solution, the Hessian can be indefinite and \( h(p) \) may not be a descent direction for \( p > 0 \). An earlier paper by this author [4] describes the factorization of the Hessian, its modification when it is not positive definite, and storing it in sparse form. In this paper we show that an appropriate rotation of the curved trajectory can often make additional progress when far away from the solution. This rotation often trades additional Gradient and Function evaluations to reduce the number of Hessian evaluations. This tradeoff would be beneficial when the evaluation of the Hessian is very costly. In some cases while there is no reduction in the number of Hessian evaluations, improvements in the accuracy of the solution for badly scaled problems have been observed.

3. Exploring Rotations For A Two Variable Problem

We begin by focusing on a two variable problem. A rotation of the CTA trajectory curve is probably best described by an angle \( \theta \) which when measured positive counterclockwise, inserts a matrix multiplying the iteration trajectory as follows:

\[ x_{k+1} = x_k + R(\theta)h(p), \]

Where

\[ R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \]

Observe that when \( \theta = 0 \), \( R(0) \) is the unit matrix. Adding this rotation complicates the \( k^{th} \) step, if we were to search for the best \( p_k \) and the best \( \theta_k \) simultaneously. Perhaps this produces the best overall results for some algorithms. For this paper, since we are adding rotations to an already optimal algorithm, we first calculate the best \( p_k \) using the algorithm as is. Then we consider adding a rotation to the trajectory using the calculated \( p_k \) to determine if greater progress can be made towards the solution along a rotated trajectory; this can be viewed as rotating the trajectory and calculating a new value for \( p \geq p_k \) along the rotated trajectory. Before describing the calculation of a productive angle of rotation, let us look at an example.

Consider Rosenbrock’s [10] well known banana-shaped valley function:

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

with the usual starting guess point, and solution:

\[ x_0 = [0 \ 1]^T, \quad x = [1 \ 1]^T \]

The CTA package first iteration is given by:

\[ x = [\begin{array}{c} -1.2 \\ 1.0 \end{array}], \quad f = 24.2, \quad f' = [\begin{array}{c} -215.6 \\ -88.0 \end{array}], \quad \text{and} \ \ f'' = \begin{bmatrix} 1330 & 480 \\ 480 & 200 \end{bmatrix} \]

The first iteration vector function \( h(p) \) is given by:

\[ h(p) = \begin{bmatrix} p(0.04532 + p(0.02416 + 0.003879p)) \\ p(0.6979 - p(0.4968 - 0.06462p)) \end{bmatrix} \]
The CTA algorithm calculates \( p_1 = 5 \), yielding: 
\[ x_1 = [0.1156, 0.1479] \], \( f = 2.59 \)

Figure 1 shows contours of \( f \) (pasted from Maple) and this curved trajectory. This figure shows Gradient vectors along the trajectory. The trajectory angle with the Gradient vector plays a significant role in the implementation of rotation as shown later. Figure 2 shows Rosenbrock’s function value along this trajectory as a function of \( p \).

**Figure 1.** Contours of Rosenbrock [10] Banana Shaped Valley showing the original curved trajectory from the starting point \( x_0 = (-1.2, 1) \), and a rotated trajectory.

**Figure 2.** Contours of Rosenbrock [10] Banana Shaped Valley showing the first curved trajectory from the starting point \( x_0 = (-1.2, 1) \). And showing \( x_1 \) for \( p = 5 \). Also shown are Gradient vector directions at some points along the trajectory.

**Figure 3 Rosenbrock [10] Banana Shaped Valley function along the first step trajectory for the CTA algorithm as a function of \( p \).

The motivation for rotations is that the Gradient points in the direction where the trajectory should be rotated to obtain smaller values of the function being minimized. Given that the trajectory is a truncation of an infinite series, it is conjectured that a productive rotation accounts for some of the excluded terms. Figure 3 shows the original trajectory, and a more productive rotated trajectory beyond the step 1 point.

### 4. Calculating Angle Of Rotation For A Two Variable Problem

After calculating the best \( p = p_k \) at the \( k^{th} \) step with no rotation (this means \( \theta = 0 \)), we pursue calculating a suitable value for \( \theta = \theta_k \) that allows for further progress along the rotated trajectory. A general strategy is that calculating a productive rotation angle should not require calculation of the Hessian. The goal is to reduce the number of overall Hessians required, and using a new Hessian to determine a rotation angle would likely not help this goal. Using an additional Function evaluation or two in the neighborhood of \( p = p_k \) also seems reasonable, but no additional Gradients will be used in the calculation of a rotation angle.

There are three useful angles that can be calculated. We begin the derivations with the objective function as a function of \( \theta \) and \( p \):

\[
 f(x) = f(\theta, p) = f(x_1 + R(\theta)h(p)) .
\]

And the partial derivative with respect to \( \theta \) at \( p = p_k \) is given by:

\[
 \frac{\partial f}{\partial \theta} = f^T R'(h(p)) \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} -\sin \theta & 0 \\ -\cos \theta & -\sin \theta \end{bmatrix} \begin{bmatrix} h_1(p) \\ h_2(p) \end{bmatrix}.
\]

The above equation simplifies to:

\[
 \frac{\partial f}{\partial \theta} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} h_1(p) \\ h_2(p) \end{bmatrix} \cos \theta \\
- \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} h_1(p) \\ h_2(p) \end{bmatrix} \sin \theta.
\]

And at \( \theta = 0 \) and \( p = p_k \), this partial derivative value indicates the importance of \( \theta \):
\[
\frac{\partial f}{\partial \theta} = \frac{\partial f}{\partial x_2} h(p_i) - \frac{\partial f}{\partial x_1} h(p_i)
\]

At the minimum with respect to \( \theta \), this derivative is zero, but calculating this \( \theta' \) for \( \frac{\partial f}{\partial \theta} = 0 \) is not easy because the Gradient is changing with \( \theta \) and we want to avoid more Gradient evaluations. With the poor assumption that the Gradient does not change much, we obtain a poor approximation of \( \theta' \) which we label \( \theta_1 \):

\[
\theta_1 = \tan^{-1}\left( \frac{\frac{\partial f}{\partial x_2} h(p_i) - \frac{\partial f}{\partial x_1} h(p_i)}{\frac{\partial f}{\partial x_1} h(p_i) + \frac{\partial f}{\partial x_2} h(p_i)} \right), \quad -\frac{\pi}{2} \leq \theta_1 \leq \frac{\pi}{2}
\]

This \( |\theta_1| \) overshoots (it is larger than) the actual minimum, since the Gradient terms are becoming smaller with a productive rotation angle. As shown later, we use a fraction of it as part of calculating an initial guess of a productive \( \theta \).

A second useful angle is defined as \( \theta_2 \), and this is the angle at \( p = p_i \), where the partial derivative with respect to \( p \) is zero given by:

\[
\frac{\partial f}{\partial p} = f^{\ast\ast} R' h(p) = \left[ \frac{\partial f}{\partial x_1} h'(p_i) - \frac{\partial f}{\partial x_2} h'(p_i) \right] \cos \theta - \left[ \frac{\partial f}{\partial x_1} h'(p_i) + \frac{\partial f}{\partial x_2} h'(p_i) \right] \sin \theta
\]

The above equation simplifies to:

\[
\left. \frac{\partial f}{\partial p} \right|_{p=p_i} = \left[ \frac{\partial f}{\partial x_1} h'(p_i) - \frac{\partial f}{\partial x_2} h'(p_i) \right] \cos \theta
\]

At the minimum with respect to \( p \), the derivative is zero, and again with the poor assumption that the Gradient does not change much, we obtain:

\[
\theta_2 = \tan^{-1}\left( \frac{\frac{\partial f}{\partial x_1} h'(p_i) - \frac{\partial f}{\partial x_2} h'(p_i)}{\frac{\partial f}{\partial x_1} h'(p_i) + \frac{\partial f}{\partial x_2} h'(p_i)} \right), \quad -\frac{\pi}{2} \leq \theta_2 \leq \frac{\pi}{2}
\]

A third important angle is defined as \( \theta_3 \), and this is the angle between the trajectory and the Gradient at \( \theta = 0 \) and \( p = p_i \). This angle is between the tangent vector to the trajectory, \( h(p_i) \), and the Gradient. The angle between two vectors is:

\[
\theta_3 = \cos^{-1}\left( \frac{f^{\ast\ast} h'(p_i)}{\| f^{\ast\ast} h(p_i) \|} \right), \quad 0 \leq \theta_3 \leq \pi.
\]

With these three angles, we select an initial guess for \( \theta \) to be the following:

\[
\theta_0 = 0.4 \max(\theta_2 - \frac{\pi}{2}, |\theta_1|, \min(|\theta_1|, \frac{\pi}{2} - |\theta_1|))
\]

Typically at \( p = p_i \), the trajectory is no longer a descent direction and therefore \( \theta_0 \) exceeds 90 degrees. Recall that descending trajectories must have less than 90 degree angles with the Gradient (see for example [9] pages 20-21).

And the sign of \( \theta_0 \) is equal to the sign of \( \frac{\partial f}{\partial \theta} \bigg|_{\theta=0} \).

When the CTA iteration was successful with \( p_i \geq 1 \), then we set \( \theta_i = \theta_0 \). Otherwise evaluate the objective function along the rotated trajectory with \( \theta = \theta_i \) and \( p = p_i \) and then use a quadratic fit to obtain an improved \( \theta_i \). When this \( \theta_i \) is significantly larger than the initial theta, further improvement is obtained with another function evaluation along the rotated trajectory with \( \theta = \theta_i \) and a second quadratic fit. Thus up to two function evaluations may be required to obtain a productive theta, and some steps require none.

These calculations yield the productive rotation angle of \(-0.282835 \) radians shown in Figure 3 for Rosenbrock function with no additional function samples. The angles described above (in radians) are: \( \theta_0 = 0.70711 \), \( \theta_1 = 0.4954 \) and \( \theta_2 - \pi / 2 = 0.4954 \). The partial derivative is:

\[
\frac{\partial f}{\partial \theta} \bigg|_{\theta=0} = -28.539
\]

Figures 4 and 5 show the rotation angle contours and a 3D graph for Rosenbrock’s Banana function versus \( \theta \) and \( p \).
After implementing rotations in the CTA package, and testing hundreds of functions, it was determined that rotations should not be attempted unless the following is satisfied:

- CTA trajectory was the result of a converging series (successive terms have smaller norm),
- Hessian is positive definite when the order selected is two (Newton’s direction),
- Current point is far from the solution (Gradient norm is greater than 2),
- CTA’s $k^{th}$ step worked well ($p_k$ was not too small),
- Calculated $\theta_k$ is not too small (greater than 0.001),
- Function value on rotated trajectory is smaller than on the original trajectory.

The last item to discuss is the search for a new $p = p_k$ along the rotated trajectory. We want to start searching at the $p$ value on the trajectory which corresponds to the point closest to the point for $p_k$ on the non-rotated trajectory. This is done by calculating smallest distance between the two trajectories without needing any objective function evaluations. Figure 6 shows the results for Rosenbrock’s function.

5. More Than Two Variable Problems

Including more variables into rotating the trajectory starts by determining the most important coordinates to include in a rotation calculation. We argue that they are the coordinates that impact the Gradient angle with the trajectory the most. This means sorting the dot product terms of $f'(p_k)$.

Initially we only focused on the terms that are positive in the dot product thinking those coordinates are the ones making the angle the largest. However after experimentation, sorting on the absolute value of the dot product terms proved to be the best determinant of the most important coordinates. Also experimentally we determined that terms that are smaller than 15% of the largest one are not worth considering, and no more than the top 4 important coordinates needed to be considered.

After ranking the coordinates, the top two coordinates are used to calculate the first angle of rotation $\theta_0$, as explained in the previous section. The only change to the previous section is the calculation of $\theta_i$ when there are three important coordinates:

$$
\theta_i = \max(0.5 \max(\theta_i - \pi / 2, |\theta_i|), 0.39 \min(|\theta_i|, \pi / 2 - \theta_i)),
$$

and when there are four important coordinates:

$$
\theta_i = \max(0.6 \max(\theta_i - \pi / 2, |\theta_i|), 0.38 \min(|\theta_i|, \pi / 2 - \theta_i)),
$$

where the factors were optimized experimentally. Then all the important coordinates are added to the rotation as follows. Each of these coordinates in order is paired one at a time, with all the preceding ones, and the angle of rotation is calculated as described above. For example, the rotation matrix for a three variable problem where all are important variables is effectively the product of these three rotation matrices:

$$
R(\theta) = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \theta & -\sin \theta \\
0 & \sin \theta & \cos \theta
\end{bmatrix} \begin{bmatrix}
\cos \theta & 0 & -\sin \theta \\
0 & 1 & 0 \\
\sin \theta & 0 & \cos \theta
\end{bmatrix} \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \theta & \sin \theta \\
0 & -\sin \theta & \cos \theta
\end{bmatrix}
$$

Figure 5. A 3D graph of Rosenbrock [10] Banana Shaped Valley function along the first step trajectory for the CTA algorithm as functions of $p$ and $\theta$.

Figure 6. Contours of Rosenbrock [10] Banana Shaped Valley showing the original curved trajectory and rotated trajectory and starting guess to search for a better point leading to $p = 6.17$. 

The new better point on rotated trajectory is with $p = 6.17$.

Closest point on rotated trajectory is with $p = 5.44$.
The implementation always handles rotation between two variables at a time; and in fact, it never operates with any matrix larger than $2 \times 2$.

6. Results Of Some Cuter Unconstrained Problems [2]

The Curved Trajectories Algorithm (CTA) package, including the option of rotation, has been implemented to run in the Cuter environment, and is available from the author for academic use. The CTA package solves all the error-free unconstrained problems, including ones with interval constraints on the variables. Table 1 shows details on some problems running with and without rotation. Stopping criteria has been selected to achieve solutions as good as published results, typically with Gradient norms less than $5 \times 10^{-14}$.

Table 1. Summary of CTA package performance on a sample of small unconstrained problems from Cuter. Column labels: Rot is number of iterations that used rotations; It is number of iterations, NF, NG, and NH are the number of Function, Gradient and Hessian evaluations; f and Gradient columns are the ending Function and Gradient norm values.

<table>
<thead>
<tr>
<th>Problem size</th>
<th>Problem</th>
<th>Rot</th>
<th>It</th>
<th>NF</th>
<th>NG</th>
<th>NH</th>
<th>f</th>
<th>Gradient norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROSENBR 4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>26</td>
<td>14</td>
<td>4</td>
<td>2.799E-14</td>
<td>1.75E-06</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>12</td>
<td>14</td>
<td>3</td>
<td>5.480E-12</td>
<td>7.79E-05</td>
<td></td>
</tr>
<tr>
<td>WOOD 4</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>26</td>
<td>23</td>
<td>6</td>
<td>1.143E-10</td>
<td>1.48E-04</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>17</td>
<td>12</td>
<td>4</td>
<td>2.860E-19</td>
<td>1.34E-08</td>
<td></td>
</tr>
<tr>
<td>BRYBND 10</td>
<td>0</td>
<td>20</td>
<td>179</td>
<td>84</td>
<td>19</td>
<td>3.904E-06</td>
<td>2.01E-04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>16</td>
<td>149</td>
<td>77</td>
<td>16</td>
<td>3.770E-06</td>
<td>1.74E-04</td>
<td></td>
</tr>
<tr>
<td>CHNROSB 10</td>
<td>0</td>
<td>16</td>
<td>90</td>
<td>57</td>
<td>16</td>
<td>1.415E-08</td>
<td>1.04E-04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>14</td>
<td>128</td>
<td>62</td>
<td>14</td>
<td>4.656E-11</td>
<td>1.03E-05</td>
<td></td>
</tr>
<tr>
<td>DJMAANJ 15</td>
<td>0</td>
<td>10</td>
<td>120</td>
<td>64</td>
<td>10</td>
<td>6.31E-07</td>
<td>3.51E-05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4</td>
<td>43</td>
<td>23</td>
<td>4</td>
<td>4.31E-05</td>
<td>3.51E-05</td>
<td></td>
</tr>
<tr>
<td>ENGVAL2 3</td>
<td>0</td>
<td>8</td>
<td>70</td>
<td>35</td>
<td>8</td>
<td>1.855E-12</td>
<td>5.24E-05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>6</td>
<td>53</td>
<td>25</td>
<td>6</td>
<td>2.997E-11</td>
<td>4.10E-05</td>
<td></td>
</tr>
<tr>
<td>GROWSH5 5</td>
<td>0</td>
<td>48</td>
<td>444</td>
<td>218</td>
<td>44</td>
<td>1.004E+04</td>
<td>1.27E-08</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>48</td>
<td>712</td>
<td>239</td>
<td>39</td>
<td>1.004E+04</td>
<td>9.88E-05</td>
<td></td>
</tr>
<tr>
<td>HEART8 6</td>
<td>0</td>
<td>131</td>
<td>2675</td>
<td>1108</td>
<td>114</td>
<td>2.006E+04</td>
<td>4.21E-04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>146</td>
<td>8170</td>
<td>972</td>
<td>80</td>
<td>7.166E-02</td>
<td>4.82E-04</td>
<td></td>
</tr>
<tr>
<td>HEART8 8</td>
<td>0</td>
<td>69</td>
<td>1275</td>
<td>503</td>
<td>57</td>
<td>1.151E-14</td>
<td>2.61E-06</td>
<td></td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>39</td>
<td>657</td>
<td>278</td>
<td>35</td>
<td>6.530E-16</td>
<td>6.69E-07</td>
<td></td>
</tr>
<tr>
<td>MARATOSB 5</td>
<td>0</td>
<td>378</td>
<td>3797</td>
<td>1809</td>
<td>556</td>
<td>2.516E-07</td>
<td>3.07E-05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>361</td>
<td>3732</td>
<td>1756</td>
<td>345</td>
<td>1.027E-07</td>
<td>1.27E-05</td>
<td></td>
</tr>
<tr>
<td>MEYER3 3</td>
<td>0</td>
<td>25</td>
<td>317</td>
<td>128</td>
<td>23</td>
<td>87.9459</td>
<td>4.35E-04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>19</td>
<td>285</td>
<td>113</td>
<td>18</td>
<td>87.9459</td>
<td>2.53E-05</td>
<td></td>
</tr>
<tr>
<td>PFITLS 3</td>
<td>0</td>
<td>17</td>
<td>126</td>
<td>69</td>
<td>17</td>
<td>1.107E+04</td>
<td>2.14E-04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>13</td>
<td>149</td>
<td>61</td>
<td>5</td>
<td>1.485E+04</td>
<td>2.82E-04</td>
<td></td>
</tr>
<tr>
<td>PFITLS 3</td>
<td>0</td>
<td>35</td>
<td>418</td>
<td>191</td>
<td>35</td>
<td>1.503E+06</td>
<td>2.08E-04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>24</td>
<td>294</td>
<td>116</td>
<td>20</td>
<td>1.225E+05</td>
<td>4.92E-04</td>
<td></td>
</tr>
<tr>
<td>PFITLS 3</td>
<td>0</td>
<td>24</td>
<td>313</td>
<td>133</td>
<td>24</td>
<td>6.534E-06</td>
<td>3.30E-04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>52</td>
<td>727</td>
<td>332</td>
<td>51</td>
<td>6.631E-06</td>
<td>4.97E-04</td>
<td></td>
</tr>
<tr>
<td>PFITLS 3</td>
<td>0</td>
<td>35</td>
<td>354</td>
<td>162</td>
<td>25</td>
<td>6.483E-07</td>
<td>2.18E-04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>16</td>
<td>195</td>
<td>94</td>
<td>15</td>
<td>3.991E-06</td>
<td>4.37E-04</td>
<td></td>
</tr>
<tr>
<td>YFITU 3</td>
<td>0</td>
<td>26</td>
<td>158</td>
<td>96</td>
<td>26</td>
<td>6.670E-13</td>
<td>7.25E-07</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>31</td>
<td>281</td>
<td>132</td>
<td>30</td>
<td>7.470E-13</td>
<td>1.14E-04</td>
<td></td>
</tr>
</tbody>
</table>

7. Conclusion

This paper presents the derivation and implementation of adding a rotation option to a minimization algorithm. It has been implemented in the already very efficient CTA package and produced encouraging results, though in a few cases it caused more work. Using rotations changes the path taken to arrive at the solution and this can sometimes end up in a region that is not smooth causing more work to obtain the solution. Rotations reduced the work for the vast majority of the problems and saved Hessian evaluations, and this would be of benefit when the Hessian is costly to evaluate.

Table 2 summarizes the results on 127 problems with less than 300 variables, giving a view of the likely impact of rotations for a large number of difficult problems. For sure, many of the problems do not use rotations. Running times are about the same. Rotations typically save Hessian evaluations at the expense of additional Gradient and Function evaluations. This would be attractive when evaluating the Hessian is very costly.

As the samples in the Tables show, rotation makes significant improvements for some problems, no change to other problems, and causes a few problems to take a longer path to the solution; this is not an unexpected result with many difficult problems. In many cases rotations improve the solution found. The singular and poorly scaled problem SCOSINE in Table 3 is worth highlighting: rotations ended up needing one more Hessian, but the solution is more accurate.

Table 3. Summary of CTA package performance on a sample of large unconstrained problems from Cuter. Column labels: Rot is number of iterations that used rotations; It is number of iterations, NF, NG, and NH are the number of Function, Gradient and Hessian evaluations; f and Gradient columns are the ending Function and Gradient norm values.

<table>
<thead>
<tr>
<th>Problem size</th>
<th>Rot</th>
<th>It</th>
<th>NF</th>
<th>NG</th>
<th>NH</th>
<th>f</th>
<th>Gradient norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAINWOO 0</td>
<td>7</td>
<td>568</td>
<td>137</td>
<td>7</td>
<td>4.37E-04</td>
<td>3.57E-06</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>6</td>
<td>1471</td>
<td>357</td>
<td>5</td>
<td>4.37E-04</td>
<td>4.64E-04</td>
</tr>
<tr>
<td>SERBND 0</td>
<td>5</td>
<td>9226</td>
<td>1020</td>
<td>5</td>
<td>3.705E-19</td>
<td>3.58E-04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1,000</td>
<td>5</td>
<td>10127</td>
<td>1048</td>
<td>4</td>
<td>3.075E-11</td>
<td>2.01E-05</td>
</tr>
<tr>
<td>SCOSINE 0</td>
<td>6</td>
<td>375</td>
<td>49</td>
<td>4</td>
<td>3.97E-03</td>
<td>2.46E-07</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10,000</td>
<td>6</td>
<td>2284</td>
<td>215</td>
<td>5</td>
<td>9.99E-03</td>
<td>7.99E-07</td>
</tr>
</tbody>
</table>

Table 2. Totals for 127 unconstrained Cuter problems with less than 300 variables.

<table>
<thead>
<tr>
<th>Package</th>
<th>Iterations</th>
<th>NF</th>
<th>NG</th>
<th>NH</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTAM</td>
<td>1530</td>
<td>29082</td>
<td>9706</td>
<td>1401</td>
</tr>
<tr>
<td>CTA with rotation</td>
<td>1403</td>
<td>34722</td>
<td>9406</td>
<td>1291</td>
</tr>
</tbody>
</table>
REFERENCES


