An implementation of Newton’s method for Keating’s potential optimization problems

Anton Anikin, Alexander Gornov
Institute for System Dynamics and Control Theory
of Siberian Branch of Russian Academy of Sciences,
Irkutsk, Russia
anton.anikin@gmail.com
gornov@icc.ru

Abstract. A modification of Newton’s method for solving multidimensional problems of Keating’s potential optimization is addressed. The proposed approach is based on two main futures: information about the special structure of Hessian matrix and sparse matrix technology. Developed data structures for store sparse Hessian matrix and modification of the linear conjugate method for the Hessian matrices with non-positive determinant are considered. The results of computational experiments and comparable tests are presented.

Keywords: Keating’s potential, Newton’s method, sparse matrix

1. Introduction

Currently, the development of new materials for optoelectronic devices used heterostructures “Ge-Si” with germanium quantum dots [YSDe00, YDN06]. To simulate the interactions between atoms in such hybrid materials used Keating’s potential function [Kea66], in which the interaction energy can be written as:

\[ E = \sum_{i=1}^{n} \left[ \frac{3}{16} \sum_{j=1}^{4} \frac{\alpha_{ij}}{d_{ij}^2} \left( (r_i - r_j)^2 - d_{ij}^2 \right)^2 \right] + \]
\[ + \frac{3}{8} \sum_{j=1}^{4} \sum_{k=j+1}^{4} \frac{\beta_{ijk}}{d_{ij} \cdot d_{ik}} \left\{ (r_i - r_j) \cdot (r_i - r_k) + \frac{d_{ij} \cdot d_{ik}}{3} \right\}^2 \]

Here:
- \( n \) - the number of atoms in lattice;
- \( d_{ij}, d_{ik}, \alpha_{ij}, \beta_{ijk} \) - given constants;
- \( r_i = (x_{1i}, x_{2i}, x_{3i}) \),
- \( r_j = (x_{1j}, x_{2j}, x_{3j}) \),
- \( r_k = (x_{1k}, x_{2k}, x_{3k}) \) - optimized variables.

Thus, Keating’s potential minimization problem can be reduced to the standard form:

\[ f(x) \rightarrow \min_{x \in \mathbb{R}^n} \]

Minimization of this potential is necessary to simulate the properties of designed materials because it provides a steady state of a physical medium (lattice). To solve the problems of function minimization developed many methods and approaches, but the high requirements to the solutions of problems of Keating’s potential minimization, unfortunately, not able to apply the known methods directly. The problems, which now are under consideration, have a higher dimension - at least \( 10^5 \) variables. Current practical problems have about \( 10^6 \) atoms in lattice and respectively above \( 3 \cdot 10^6 \) variables. In addition, the solution must be obtained with accuracy comparable to machine \( \varepsilon \) (\( \approx 10^{-20} \) for double).

Traditionally, for high-precision solutions for unconstrained minimization problems used Newton’s method, which, having a quadratic rate of convergence, can, if you have good information about the first and second derivatives, to get the most correct digits in the result. In current practical importance Keating’s potential optimization problems number of atoms in the crystal lattice is more than \( 10^5 \), respectively, the number of variables (the coordinates of individual nodes in three dimensional space) - \( 3 \cdot 10^5 \). These dimensions make it impossible to direct use second-order methods for due to memory constraints of modern computer systems, since there is no physical possibility to store the matrix
of second derivatives (Hessian), which requires about \( n^2 \) memory (\( n \) - the dimension of the problem).

The paper deals with the modification of Newton’s method, based on the sparse structure of Hessian matrix. In section 2, sparse and dense views of Hessian matrix are shown. Section 3 outlines some popular methods of sparse matrices storing and describes proposed method for Hessian storage. Section 4 contains description of developed algorithm of Newton’s method with using sparse matrix technology. Section 5 shows results of computational experiments and some comparative tests.

2. The structure of Hessian

In general, matrix of second derivatives is dense and has the form shown in Figure 1. In many problems, including the considered Keating’s potential energy function, we have an analytic function presentation, which allows us to represent a matrix as shown in Figure 2.

```
\[
\begin{array}{cccc}
\frac{\partial f}{\partial x_1 \partial x_1} & \frac{\partial f}{\partial x_1 \partial x_2} & \frac{\partial f}{\partial x_1 \partial x_3} & \cdots & \frac{\partial f}{\partial x_1 \partial x_n} \\
\frac{\partial f}{\partial x_2 \partial x_1} & \frac{\partial f}{\partial x_2 \partial x_2} & \frac{\partial f}{\partial x_2 \partial x_3} & \cdots & \frac{\partial f}{\partial x_2 \partial x_n} \\
\frac{\partial f}{\partial x_3 \partial x_1} & \frac{\partial f}{\partial x_3 \partial x_2} & \frac{\partial f}{\partial x_3 \partial x_3} & \cdots & \frac{\partial f}{\partial x_3 \partial x_n} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial f}{\partial x_n \partial x_1} & \frac{\partial f}{\partial x_n \partial x_2} & \frac{\partial f}{\partial x_n \partial x_3} & \cdots & \frac{\partial f}{\partial x_n \partial x_n}
\end{array}
\]
```

Figure 1: Dense Hessian matrix

3. Sparse matrix storing methods

There are various methods of storing sparse matrices, which have different properties and used depending on the matrix and the method-sof treatment [Pis84]. Such storage methods include:

- Dictionary of keys (DOK).
\[
\begin{pmatrix}
\frac{\partial f}{\partial x_1 \partial x_1} & \frac{\partial f}{\partial x_1 \partial x_2} & 0 & \cdots & \frac{\partial f}{\partial x_1 \partial x_n} \\
\frac{\partial f}{\partial x_2 \partial x_1} & \frac{\partial f}{\partial x_2 \partial x_2} & 0 & \cdots & 0 \\
0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial f}{\partial x_n \partial x_1} & 0 & 0 & \cdots & \frac{\partial f}{\partial x_n \partial x_n}
\end{pmatrix}
\]

Figure 2: Sparse Hessian matrix

- List of lists (LIL).
- Coordinate list (COO).
- Compressed sparse row (CSR or CRS).
- Compressed sparse column (CSC or CCS).

and some other methods and their modifications.

In this paper, we apply a special storage format that is optimized for these tasks that have guaranteed the maximum number of known non-zero elements in row of Hessian. We denote this number of \( L \). Facing this fact, the following structure of the sparse matrix storage is used:

<table>
<thead>
<tr>
<th>Indexes</th>
<th>( I_{1,1}, I_{1,2} \ldots I_{1,L} )</th>
<th>( I_{2,1}, I_{2,2} \ldots I_{2,L} )</th>
<th>( \ldots )</th>
<th>( I_{n,1} \ldots I_{n,L} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td>( V_{1,1}, V_{1,2} \ldots V_{1,L} )</td>
<td>( V_{2,1}, V_{2,2} \ldots V_{2,L} )</td>
<td>( \ldots )</td>
<td>( V_{n,1} \ldots V_{n,L} )</td>
</tr>
</tbody>
</table>

Figure 3: The data structure for storing sparse Hessian

The indexes vector has length \( n \cdot L \), and contains lists of column numbers, which are non-zero derivatives for each row of the matrix. If the number of such derivatives is less than \( L \), the remaining cells for a given row filled with \((-1)\), which is a sign of the end of the corresponding list. Values vector also has a length \( n \cdot L \) and contains the values of the second derivatives, corresponding to a specific row and column of the Hessian matrix:

\[
V_{j,k} = \frac{\partial f}{\partial x_j \partial x_m}, m = I_{j,k}
\]
Using such structure for storing a sparse matrix requires about $2 \cdot L \cdot n$ cells of memory, which is significantly less than the amount needed to store the complete (dense) matrix - $n^2$ cells, in case $L \ll n$.

Since not all lines of the Hessian matrix have exactly $L$ of nonzero elements, this storage structure is not completely optimal. Nevertheless, in addressing the problem of energy Keating’s optimization of number of blank cells is small and for large dimensions ($10^5$ and higher), their number shall not exceed $4 - 5\%$ of the total number of cells. This method, when the cells are given with some excess allows simplifying code and speed up access to the custom line because it does not require a separate storage index of beginning of this line. In addition, in order to optimize access to the main diagonal, each its element is first in list, corresponding to the selected row.

While working with the Keating’s function, the fact that the number of nonzero elements of the rows of Hessian does not exceed 51 was revealed. Thus, for the storage of sparse form of this matrix the amount of $2 \cdot n \cdot 51$ cells, allowing us to keep it whole in memory during the solving problems with required dimensions, is enough.

4. Algorithm

When using the classical Newton’s method the existence of a sparse Hessian cannot be effectively used, because of solving the linear algebra problem using, for example, the Gauss method, will convert matrix from sparse format into triangular, with the number of nonzero elements approximately $(n^2/2)$, which critically limits the dimension of the task. Therefore, in the current study we used a modification of Newton’s method, based on the approximate solution of systems of linear equations using the conjugate gradient method [WE71].

In implementing the conjugate gradient method for its correct operation it is necessary to note, that the current Hessian matrix must have a positive determinant. In case of violation of this condition the main diagonal elements are increasing their values for some $\omega$, which is determined by a binary search algorithm in such a way as to be minimal (up to predefined $\varepsilon_\omega$). It was experimentally determined that the reliability
of the developed method is enhanced if a number of internal parameters of the algorithm satisfies additional conditions those are controlled by the parameter $Q$. All calculations of the first and second derivatives in the algorithm are produced by analytical formulas.

The developed method algorithm involves the following steps:

1) $x = x_0; Q = 0; \ f_{\text{min}} = f(x)$.

2) In order to solve the current system of linear equations the conjugate gradient method is used.

3) One-dimensional search is performed on the selected direction, calculated value of the selected point $f_k = f(x_k)$.

4) If $f_k < f_{\text{min}}$, then $f_{\text{min}} = f_k; x = x_k$, the transition to the next iteration (step 2).

5) Repeat steps 2-4 for different $Q > 0$ from a predetermined set of values.

6) In case of failure to achieve improved value of $f_{\text{min}}$, or when a certain boundary conditions ($\|\nabla f\| \leq \varepsilon$ and so on) - the completion of the algorithm.

5. Computational experiments

The algorithm was implemented with C language and tested on Linux OS (32 and 64-bit kernels) with different versions of GCC and ICC compilers. During testing most of the computing resources consumed to perform the procedure of multiplication of sparse Hessian matrix on dense vector $x$. To improve performance of created algorithm was used OpenMP technology [CJP07], and further calculations were performed on a computing system with 12 cores of Intel Xeon X5670, experiments were made with different numbers of used cores. The resulting parallel version of function, which perform multiplication of sparse matrix on a dense vector, effectively uses the available computing resources, significantly (up to 3-4 times compared to one CPU core) improving the performance of the final version of algorithm. We are also developing a parallel version of the algorithm with using NVidia CUDA technology [JE10].
Newton’s method

For example, in table 1 we present the results of solving the minimization problem with various sizes of “Ge-Si” crystals on computational system with using 10 cores.

Table 1: Newton’s method modification with sparse matrix. Big-scale problems.

<table>
<thead>
<tr>
<th>n (dimension)</th>
<th>Function value before</th>
<th>Function value after</th>
<th>t, sec.</th>
<th>Number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>98304</td>
<td>1.744335 · 10^{-3}</td>
<td>2.678759 · 10^{-25}</td>
<td>190</td>
<td>11</td>
</tr>
<tr>
<td>139968</td>
<td>1.744335 · 10^{-3}</td>
<td>1.240275 · 10^{-22}</td>
<td>260</td>
<td>11</td>
</tr>
</tbody>
</table>

A comparative tests of the developed method and A.Yu. Gornov’s implementation of classic Newton method with dense matrix were performed. Testing was executed on the computer system with Intel Core 2 Duo T8300 CPU, using 1 core by algorithms. The results presented in tables 2 and 3.

Table 2: Newton’s method with dense matrix.

<table>
<thead>
<tr>
<th>n (dimension)</th>
<th>Function value before</th>
<th>Function value after</th>
<th>t, sec.</th>
<th>Number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>3.534246 · 10^{-3}</td>
<td>1.739865 · 10^{-31}</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>192</td>
<td>1.744335 · 10^{-3}</td>
<td>1.529105 · 10^{-29}</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>648</td>
<td>3.564625 · 10^{-3}</td>
<td>1.295851 · 10^{-34}</td>
<td>87</td>
<td>6</td>
</tr>
<tr>
<td>1536</td>
<td>1.744335 · 10^{-3}</td>
<td>1.671099 · 10^{-24}</td>
<td>3635</td>
<td>7</td>
</tr>
<tr>
<td>3000</td>
<td>3.564625 · 10^{-3}</td>
<td>1.341495 · 10^{-4}</td>
<td>5021</td>
<td>6</td>
</tr>
</tbody>
</table>

As we can see from the tables 2 and 3, the developed method shows a significant acceleration relative to the Newton method with dense matrix with increasing dimension of the problem, caused by significant reduction of computations associated with the Hessian.

6. Conclusion

The proposed approach allows the use of the ideology of Newton’s method for solving large-scale problems, whose solving on modern
Table 3: Newton’s method modification with sparse matrix.

<table>
<thead>
<tr>
<th>n (dimension)</th>
<th>Function value before</th>
<th>Function value after</th>
<th>$t_{\text{sec}}$</th>
<th>Number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>$3.534246 \cdot 10^{-3}$</td>
<td>$1.298171 \cdot 10^{-30}$</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>192</td>
<td>$1.744335 \cdot 10^{-3}$</td>
<td>$4.244047 \cdot 10^{-28}$</td>
<td>3</td>
<td>37</td>
</tr>
<tr>
<td>648</td>
<td>$3.564625 \cdot 10^{-3}$</td>
<td>$1.295851 \cdot 10^{-4}$</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>1536</td>
<td>$1.744335 \cdot 10^{-3}$</td>
<td>$1.063583 \cdot 10^{-28}$</td>
<td>27</td>
<td>42</td>
</tr>
<tr>
<td>3000</td>
<td>$3.564625 \cdot 10^{-4}$</td>
<td>$1.341494 \cdot 10^{-4}$</td>
<td>30</td>
<td>11</td>
</tr>
</tbody>
</table>

computers with traditional methods are impossible. The presented algorithm is based on the the special structure of the problem, allowing the use of sparse matrix technology for storage of the Hessian matrix. For finding the descent directions the considered algorithm uses the modification of the linear conjugate gradient method, which allows us to maintain operability in the absence of positive determinant of the Hessian matrix.

Numerical experiments have demonstrated the principle operability of the proposed method. Despite the high sensitivity of the algorithm to the quality of algorithmic parameters setting, it made possible to solve some urgent problems of Keating’s potential minimization with dimensions which exceed $10^5$ variables.

Acknowledgements

This work was supported by the interdisciplinary integration project of SB RAS No. 43 and by RFBR grant No. 10-01-00595.

References


Studia Informatica Universalis.