Fast Methods for Solving High Accuracy Surface Modeling

Na Zhao and Tian Xiang Yue

reprinted from

Journal of
Algorithms & Computational Technology

Volume 7 · Number 2

June 2013

Multi-Science Publishing Co. Ltd

Fast Methods for Solving High Accuracy Surface Modeling

Na Zhao[1] and Tian Xiang Yue[2,*]
[1,2]Institute of Geographical Sciences and
Natural Resources Research,
Chinese Academy of Sciences, Beijing 100101

Received 27/02/2012; Accepted 25/05/2012

ABSTRACT
High accuracy surface modeling (HASM) is a novel surface modeling method. The well known preconditioned conjugate gradient (PCG) method is used to solve the equations produced by HASM. In this paper, in order to improve the convergence rate of PCG, we use two preconditioners: incomplete Cholesky decomposition conjugate gradient method (ICCG) and symmetric successive over relaxation-preconditioned conjugate gradient method (SSORCG), which have not previously been available for use with HASM. Furthermore, we give adequate storage scheme of the large sparse matrix and optimize the performance of sparse matrix-vector multiplication. We test the proposed method on a Dell OP990 machine. Numerical tests show that ICCG has the fastest convergence rate of HASM. We also find that both ICCG and SSORCG have much faster convergence rates than some available solvers.

Keywords: Surface Modeling, Preconditioned Conjugate Gradient Method, Incomplete Cholesky Factorization, Successive Over-Relaxation Algorithm, HASM

1. INTRODUCTION

As an innovative method, high accuracy surface modeling (HASM) is based on the fundamental theorem of surface. The fundamental theorem of surface makes sure that a surface is uniquely defined by the first and the second fundamental coefficients [1]. HASM method, combined with Gauss-Codazzi equations, divides the simulated areas in an uniformly orthogonal way and establishes the corresponding difference equations with finite difference method. We then

*Corresponding author. yue@lreis.ac.cn, [1]zhaon@lreis.ac.cn

solve this problem under the restriction of the sample data [2]. The whole calculation process of HASM can be divided into three stages: deriving finite difference approximations to differential equations, establishing the sampling points equations and solving the algebra equation. Numerical tests show that the simulation result of HASM is much better than classical surface modeling methods [3].

In order to solve the time-consuming calculation problem and the overabundance of data problem, Yue and others gave the optimum formulation of HASM [4, 5]. However, several problems need to be settled, such as error problem and low computational speed. Effective use of this model requires that the matrix equation produced in HASM be solved efficiently, that is, that a correct solution is produced using as little computer processing time as possible. Effective use of the model also requires that the amount of computer storage be minimized to allow for solution on small computers and to avoid over-burdening large computers.

The purpose of this paper is to document PCG, a numerical code which uses the preconditioned conjugate-gradient method to solve the matrix equations produced by HASM. Two preconditioning strategies are included which have not previously been available for use with HASM, and which perform better than some existing PCG algorithm.

2. PRECONDITIONED CONJUGATE GRADIENT METHOD

In this work, selected numerical methods are presented for solving the matrix equations that arise when the finite difference method is applied. The finite difference model produces a set of linear equations which can be expressed in matrix notation as:

$$Ax = b, \tag{1}$$

where the coefficient matrix $A \in R^{n \times n}$ is symmetric and positive definite.

Linear equations such as (1) can be solved using direct or iterative methods. In most direct methods the matrix A is factored exactly and the true solution is obtained by one backward and one forward substitution. In most iterative methods, an initial estimate of the solution is refined iteratively using an approximately factored matrix. Direct methods are for the small size problem, but iterative methods are more efficient for large problems and require less computer storage [6]. The conjugate gradient method (CG) [7] is an iterative method which can be used to solve matrix equations while its coefficient matrix

is symmetric and positive definite. CG method has been the subject of considerable interest in recent years because of its efficiency and ability to solve difficult problems. Observe that the method of conjugate gradient works well on matrices that are either well-conditioned or have just a few distinct eigenvalues. However, the matrix A in HASM is ill-conditioned. Hence we further introduce the preconditioned conjugate gradient method which have a fast convergence rate through modifying the condition of A.

Alogrithm 1: Preconditioned Conjugate Gradients [8]:

Given an initial x_0,
$k = 0$, $r_0 = b - Ax_0$,
while ($r_k \neq 0$)
  solve $Mz_k = r_k$
    $k = k + 1$
If $k = 1$
  $p_1 = z_0$
else
  $\beta_k = r_{k-1}^T z_{k-1} / r_{k-2}^T z_{k-2}$
  $p_k = z_{k-1} + \beta_k p_{k-1}$
end
  $\alpha_k = r_{k-1}^T z_{k-1} / p_k^T A p_k$
  $x_k = x_{k-1} + \alpha_k p_k$
  $r_k = r_{k-1} - \alpha_k A p_k$
end
  $x = x_k$

The major amount of calculation is from the linear system $Mz_k = r_k$. The choice of a good preconditioner can have a dramatic effect upon the rate of convergence. Next, we will find different matrix M such that Algorithm 1 above can have high computational speed and the computational cost of $Mz_k = r_k$ is low.

2.1. Incomplete Cholesky Preconditioners

The linear system (1) can be converted into the following system:

$$\overline{A}x = \overline{b}, \tag{2}$$

where $\overline{A} = MA$, $\overline{b} = Mb$. If $k_2(\overline{A}) \ll k_2(A)$ ($k_2(A)$ denotes the 2-norm condition of A), system (2) can have faster computational speed than (1) by using the same algorithm if a adequate preconditioner is used.

The incomplete Cholesky preconditioner (ICCG) has been very popular [9]. Suppose the incomplete Cholesky factorization of A is as follows:

$$A = M + R = LL^T + R.$$

Where L is a lower triangular matrix with the property that $M = LL^T$ is close to A and L has the same sparsity structure with A. However, for the incomplete Cholesky factorization, $A - R = LL^T$. The structure of L can be controlled by the matrix R. This avoids the complete Cholesky factorization which destroys the sparsity structure of A. Consider that there exist many zero elements of R and the nonzero elements value of R are small in actual computation. In this paper, we consider the non-filled Cholesky factorization of A, that is, the position of L is zero if the corresponding position of A is zero. The code of this algorithm is as follows:

```
A(k, k) = √A(k,k)
      for i = k + 1: n
      if A(i, k) ≠ 0
      A(i, k) = A(i, k) / A(k, k)
      end
   end
for j = k + 1: n
   for i = j : n
      if A(i, j) = 0
      A(i, j) = A(i, j) – A(i, k) A (k, j)
      end
   end
end
end
```

Since $k_2(A) = \dfrac{\sigma_{\max}(A)}{\sigma_{\min}(A)} (k_2(A) \geq 1)$ is the 2-norm condition of a matrix A, where, $\sigma_{\max}(A)$ is the largest singular value and $\sigma_{\min}(A)$ is the smallest singular value of A, matrix $M^{-1}A$ can be modified to close an unit matrix by setting the element values of R and thus has clustered singular values. In this case, $k_2(M^{-1}A) \approx 1$. It is the purpose of precondition which makes $k_2(M^{-1}A) << k_2(A)$. In the actual computation process, to take advantage of the large number of zeros in matrices A and L, we adopt the block compressed sparse row (BCSR) format [10]. $Mz_k = r_k$ in the program can be transformed to $L^T z_k = y_k$ and $Ly_k = r_k$ and the computational cost of these is $0(n^2)$ or less.

2.2. Symmetric Successive Over-Relaxation (SSOR) Preconditioner

The major computational cost of HASM are the sparse matrix-vector multiplication and the matrix inverse. The computational cost of matrix inverse is larger than matrix-matrix multiplication. SSOR preconditioning can directly compute the inverse matrix of M which reduces the computational cost. Morever, this method only requires the sparse matrix-vector multiplication and it is easy to parallelize.

In terms of matrix splitting, if $A = L + D + L^T$, where D is diagonal and L is strictly lower triangular, the SSOR preconditioner is given by [10]:

$$M = KK^T, \tag{3}$$

where, $K = 1/\sqrt{2-w}\left(\frac{1}{w}D + L\right)\left(\frac{1}{w}D\right)^{-1/2}, 0 < w < 2.$

The inverse of K is as follows:

$$K^{-1} = \sqrt{2-w}(1/wD)^{1/2}(I + wD^{-1}L)^{-1}1/wD^{-1},$$

Denote $1/wD = \bar{D}$,
since $(I + \bar{D}^{-1}L)^{-1} = I - \bar{D}^{-1}L + (\bar{D}^{-1}L)^2 - \ldots$. So

$$K^{-1} \approx \sqrt{2-w}\bar{D}^{-1/2}(I - \bar{D}^{-1}L)\bar{D}^{-1} = \sqrt{2-w}\bar{D}^{-1/2}(I - L\bar{D}^{-1}) \equiv \bar{K}.$$

Then the approximate inverse matrix of A is $\bar{M} = \bar{K}^T\bar{K}$. In this paper, we use SSOR preconditioner [12] to improve the efficiency of HASM where $z_k = \bar{M}r_k$ in the PCG algorithm can be changed into $y_k = \bar{K}r_k$ and $z_k = \bar{K}^T y_k$ because of the same sparsity structure between A and K.

3. NUMERICAL TEST

In this section, we employ Gaussian synthetic surface (Figure 1) to validate the efficiency of PCG for solving HASM. The formulation of Gaussian synthetic surface is expressed as:

$$f(x, y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2-y^2} - \frac{e^{-(x+1)^2-y^2}}{3}.$$

The computational area is $[-3, 3] \times [-3, 3]$, and $-6.5510 < f(x, y) < 8.1062$.

The first test is that we fix the sampling interval ($m = 4$), outer iterative times (1 times) and inner convergence criteria $\left(\left\|f^{n+1} - f^n\right\|_2 < 10^{-6}\right)$. We change the grid number of computational area to compare the computational efficiency of

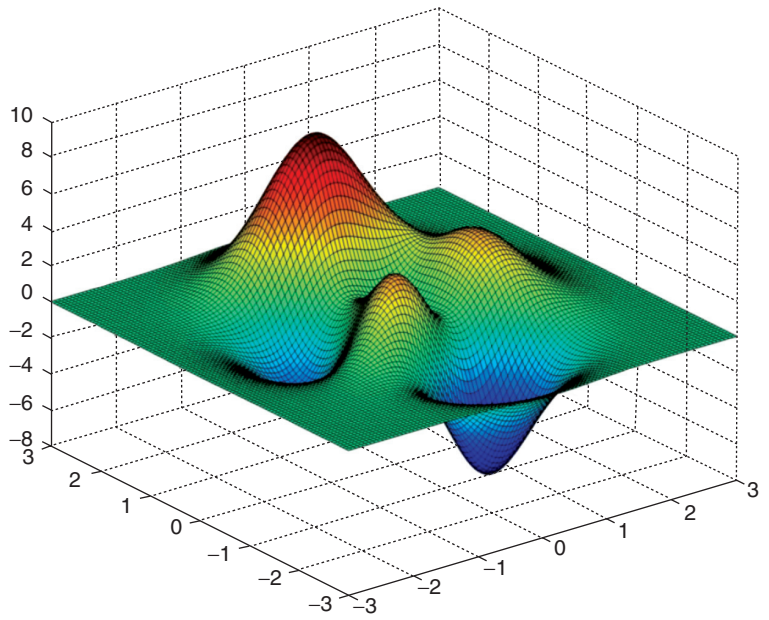

Figure 1. Gaussian synthetic surface.

PCG for different preconditioned strategies in HASM. Meanwhile, we present the relationship between the computational time and the grid number for the ICCG preconditioner and the diagonal preconditioner. In this paper, the diagonal preconditioner denotes the preconditioner is a diagonal matrix whose diagonal elements are the diagonal elements of A. The numerical results are showed in Table 1 and Figure 2. Here, S in Table 1 denotes the diagonal preconditioned conjugate gradient method (DCG) while T denotes the tridiagonal preconditioned conjugate gradient method (TCG). From Table 1 and Figure 2, we can see that the computational time and the inner iterative numbers of ICCG and SSORCG methods are less than the DCG and TCG methods. The difference against the computational time of ICCG and DCG has good linear relationship with the grid numbers, that is, the efficiency of ICCG becomes sharper with the number of grids increasing. The relationship is as follows:

$$gn = -2.3729 \times 10^3 t^2 + 1.5062 \times 10^5 t + 1.5211 \times 10^4,$$

where t is the time difference of the two methods, gn is the grid number.

Table 1. The comparison of computational efficiency with different pre-processing methods.

	Compute time (s)				t (s)	Number of inner iteration			
Number of grid	**ICCG**	**SSORCG**	**S**	**T**	**S-ICCG**	**ICCG**	**SSORCG**	**S**	**T**
101 × 101	0.0797	0.0922	0.1178	0.2418	0.0381	23	42	74	66
301 × 301	0.6826	0.7896	1.0882	1.7557	0.4056	23	38	69	61
501 × 501	1.9908	2.4962	3.6256	5.2110	1.6348	22	37	69	59
1001 × 1001	8.1942	10.4086	15.6115	21.3498	7.4173	22	37	69	59

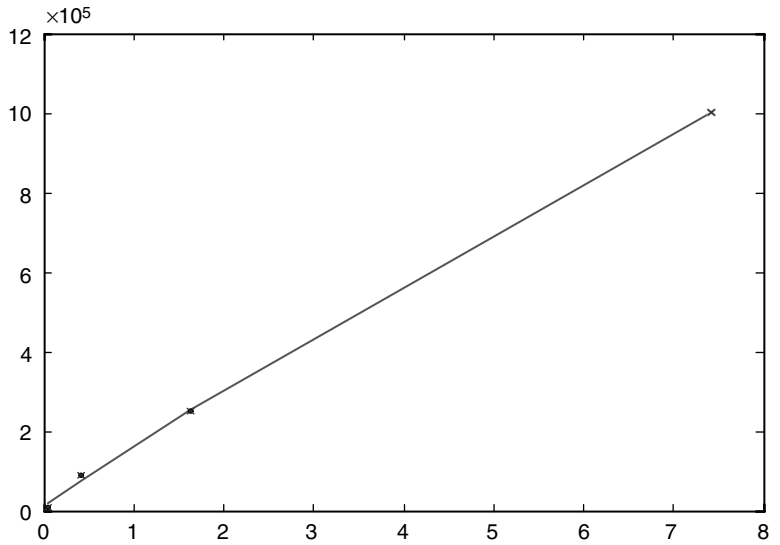

Figure 2. The relationship of the time interval and the number of grid between S and ICCG.

Results show that when we apply the PCG method in HASM, ICCG and SSORCG methods are more effective both in computational time and iterative numbers than former preconditioned conjugate gradient methods.

The second test is that we fix the sampling interval (m = 4), number of grids (101 × 101) and outer iterations (5 times). We compare the convergence accuracy of the different PCG methods. In the third test we fix the sampling interval (m = 4), the number of grid (501 × 501) and the inner iterative numbers (10 times). We then compare the convergence accuracy under different outer iterative numbers. The test results are shown in Tables 2 and 3. The results show

Table 2. The comparison of simulation accuracy with different inner-iteration numbers.

Inner iteration	**ICCG**	**SSORCG**	**S**	**T**
5	2.3482	7.4335	6.4740	7.9532
10	0.0395	0.8129	4.7385	1.9689
20	0.000026	0.0145	0.4666	0.0876
50	0	0	0.000513	0.000044

Table 3. The comparison of simulation accuracy with different outer iteration numbers.

Outer iteration	ICCG	SSORCG	S	T
2	0.001163	0.017035	0.113418	0.094594
4	0.000138	0.002427	0.013710	0.007262
6	0.000064	0.001169	0.006030	0.003271
8	0.000042	0.000821	0.007483	0.002054

that when we apply PCG method in HASM, ICCG method is the fastest followed by SSORCG method.

From the numerical tests above, we see that we can apply ICCG method and SSORCG method in HASM, and use the SSORCG method when we propose a parallel implementation of HASM.

4. CONCLUSIONS

HASM is an important tool for surface modeling. However, the low computational efficiency of HASM restricts the widely applications of it. In this paper, considering that the coefficient matrix of HASM is ill-conditioned, we apply ICCG method and SSORCG method to improve the efficiency of HASM. In actual computation, we consider the implementation details of the two methods regards of the computational time and the data storage. We use Gaussian synthetic surface to validate the methods on DELL OP990 machine. The results show that ICCG method and SSORCG method are better than other preconditioned conjugate-gradient methods when they are used in HASM.

REFERENCES

[1] Henderson, D.W., *Differential Geometry*, Prentice-Hall, Inc., London, 1998.

[2] Yue, T.X., Du, Z.P., and Liu, J.Y., High Precision Surface Modeling and Error Analysis, *Progress in Natural Science*, 2004, 14(2), 83–89.

[3] Yue, T.X. and Du, Z.P., High Accuracy Surface Modeling and Comparative Analysis of its Errors, *Progress in Natural Science*, 2007, 16(8), 986–991.

[4] Yue, T.X. and Du, Z.P., Numerical Test for Optimum Formulation of High Accuracy Surface Modeling, *Geo-Information Science*, 2006, 8(3), 83–87.

[5] Yue, T.X., *Surface Modeling: High Accuracy and High Speed Methods*, CRC Press, 2011.

[6] Remson, I., Hornberger, G.M. and Molz, F.J., *Numerical Methods in Subsurface Hydrology*, John Wiley, 1971.

[7] Hestenes, M.R. and Stiefel, E.F., Methods of Conjugate Gradients for Solving Linear Systems, *Journal of Research of the Natural Bureau of Standards*, 1952, 49, 409–436.

[8] Golub, G.H. and Van Loan, C.F., *Matrix Computations*, Posts & Telecompress, 2009.

[9] Meijerink, J.A. and van der Vorst, H.A., An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is A Symmetric M-Matrix, *Mathematics of Computation*, 1977, 31, 148–162.

[10] Yuan, E., Zhang, Y.Q., Liu, F.F. and Sun, X.Z. Automatic Performance Tuning of Sparse Matrix-Vector Multiplication: Implementation Techniques and its Application Research, *Journal of Computer Research and Development*, 2009, 46(7), 1117–1126.

[11] Evans, D.J. and Forrington, C.V.D., An Iterative Process for Optimizing Symmetric Successive Over-Relaxation. *The Computer Journal*. 1963, 6(3), 271–273.

[12] Helfenstein, R. and Koko, J., Parallel Preconditioned Conjugate Gradient Algorithm on GPU. *Journal of Computational and Applied Mathematics*. Homepage: http:\\www.elsevier.com/locate/cam.