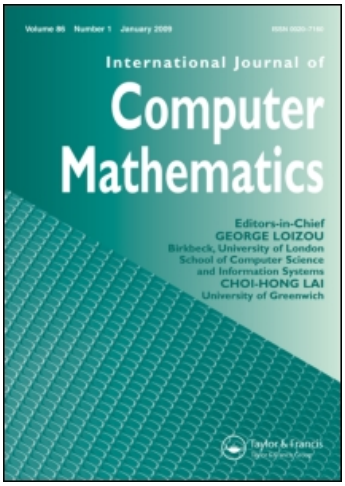

International Journal of Computer Mathematics
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713455451

Applications of the Conjugate Gradient method in optimal surface parameterizations

Zhongxiao Jia [a]
[a] Department of Mathematical Sciences, Tsinghua University, Beijing, People's Republic of China

First published on: 10 May 2010

To cite this Article Jia, Zhongxiao(2010) 'Applications of the Conjugate Gradient method in optimal surface parameterizations', International Journal of Computer Mathematics, 87: 5, 1032 — 1039, First published on: 10 May 2010 (iFirst)
To link to this Article: DOI: 10.1080/00207160802275951
URL: http://dx.doi.org/10.1080/00207160802275951

International Journal of Computer Mathematics
Vol. 87, No. 5, April 2010, 1032–1039

Taylor & Francis
Taylor & Francis Group

Applications of the Conjugate Gradient method in optimal surface parameterizations

Zhongxiao Jia*

Department of Mathematical Sciences, Tsinghua University, Beijing, People's Republic of China

(*Received 11 September 2007; revised version received 20 March 2008; accepted 18 May 2008*)

This paper concerns the solutions of very large symmetric semipositive definite (singular) linear systems involved in the problem of optimal surface parameterizations using inverse curvature mapping. Two approaches are presented that transform the singular linear systems into two kinds of symmetric positive definite linear systems, so that the famous Conjugate Gradient (CG) method can be used for solving them. Numerical experiments are run on two practical large problems to illustrate that the CG algorithm works very efficiently.

Keywords: symmetric seimidefinite; positive definite; CG; Krylov subspace; surface parameterization

2000 AMS Subject Classification: 65F10; 65D17; 68U07

1. The problem description

Surface parameterization refers to a process that maps the surface to a planar region. It has broad applications in graphics and the process introduces distortions between the original surface and its planar image. The distortions are divided into angle distortions and area distortions. A conformal parameterization is efficient for eliminating angle distortions, while an optimal conformal parameterization aims at reducing area distortions best.

In graphics, surfaces are usually represented as triangular meshes. The Riemannian metric of a surface is approximated by the polygonal metric, which can be represented as the discrete metric, i.e. edge lengths. The Gauss curvature of the surface is approximated by the discrete curvature at each vertex, which measures the difference between the summation of corner angles adjacent to the vertex and 2π. Computing a mesh parameterization amounts to finding a discrete metric such that all interior vertices have zero curvature.

A continuous conformal parameterization maps infinitesimal circles on the surface to the infinitesimal circles on the plane, and it preserves the intersection angles among them. This fact inspires circle packing and circle pattern methods for approximating conformal maps in discrete setting. The circle packing algorithm covers the mesh with circles centred at each vertex and different circles intersect each other. The edge lengths are determined by the circle radii and the

*Email: jiazx@tsinghua.edu.cn

ISSN 0020-7160 print/ISSN 1029-0265 online

DOI: 10.1080/00207160802275951
http://www.informaworld.com

intersection angles. Such discrete metric is called a circle packing metric. In the process of the conformal parameterization, the circle radii are adjusted while the intersection angles are fixed, so that the final vertex curvatures are zeros for almost all vertices except for some singularities. The circle packing metrics with the same intersection angles are called *conformal*. Collins and Stephenson [1] consider the circle packing problem that is a configuration of circles realizing a specified pattern of tangencies. They develop an efficient iterative algorithm for computing radii of packings in the Euclidean and hyperbolic planes. Kharevych *et al.* [4] discuss conformal mappings and present a novel approach for constructing discrete conformal mapping from surface meshes of arbitrary topology to the plane.

For a given mesh, the conformal circle packing metric determines discrete curvatures. It is shown [7] that the converse is true too, i.e. the curvature determines uniquely a conformal circle packing metric. Suppose that the vertex set is $\{v_1, v_2, \ldots, v_n\}$ and denoted by the vector $x = (x_1, x_2, \ldots, x_n)^T$ a conformal circle packing metric, where x_i is the logarithm of the radius at v_i, by $k = (k_1, k_2, \ldots, k_n)^T$ a curvature distribution on the mesh, where k_i is the curvature of v_i. Since scaling do not affect curvature, we normalize the conformal circle packing metric, such that $\sum_{i=1}^{n} x_i = 0$.

For a given triangle mesh, all the configurations of all edge lengths (discrete metrics) form a metric space and all the configurations of vertex curvatures (curvature functions) form a curvature space. The discrete metric determines curvature functions by the curvature mapping that is restricted to a subspace of the metric space, where all metrics are conformal to each other. The curvature mapping is a diffeomorphism when it is constrained on a subspace. Given a prescribed curvature function, the inverse curvature function can be directly used to compute the corresponding conformal metric. The inverse curvature mapping is then applied for discrete parameterizations for surfaces with arbitrary topologies by prescribing the target curvature such that all the curvatures are concentrated on several cone singularities and most vertices have zero curvatures. All such curvatures form a special subspace of the curvature space, the admissible space. In order to achieve optimal parameterizations uniformly, one needs to first define the uniformity energy on the total curvature space and then optimize the energy on the total curvature space.

This is a basic idea of optimal surface parameterizations using inverse curvature mapping. The details are tedious, and we omit them and refer to [7]. In the optimal surface parameterization algorithm, the bottleneck and the most time-consuming work are to solve a very large sparse symmetric semidefinite positive linear system at each iteration step. A coefficient matrix A is based on a three-dimensional model $M(V, E, F)$, where V is the set of mesh vertices, E the set of edges and F the set of surfaces. The number of vertices is large and can be up to a few hundred thousands. We are required to solve the function $f : v_i \to x_i$ defined on mesh vertices. It is known that on each vertex v_i of the discrete meshes the function f satisfies the discrete Poisson equation

$$b_i = \sum_{[v_i, v_j] \in E} w_{ij}(x_i - x_j), \tag{1}$$

where $[v_i, v_j]$ denotes an adjacent edge and w_{ij}s are weight functions, and $\sum_{i=1}^{n} b_i = 0$.

Equation (1) can be condensed as the following large sparse-constrained symmetric semipositive definite (singular) linear system

$$Ax = b, \tag{2}$$

where $A = A^T \in \mathcal{R}^{n \times n}$ with rank $n - 1$, $x = (x_1, x_2, \ldots, x_n)^T$, $b = (b_1, b_2, \ldots, b_n)^T \in \mathcal{R}^n$, the last row of A equals the minus sum of its first $n - 1$ rows so that the row sums of A are zeros, the non-zero entries in the ith row of A are the diagonal entries a_{ii} and a_{ij} for $[v_i, v_j] \in E$ and a

basis of the null space of A is $u = (1, 1, \ldots, 1)^T \in \mathcal{R}^n$. The constraints are

$$x_1 + x_2 + \cdots + x_n = u^T x = 0, \tag{3}$$

$$b_1 + b_2 + \cdots + b_n = u^T b = 0. \tag{4}$$

Since A is singular, all standard direct and iterative solvers cannot be used directly. Note that A is very large sparse. Due to storage requirements and computational costs, direct solvers are generally impractical and iterative solvers are the only viable choices.

The paper is organized as follows. In Section 2 we review the Conjugate Gradient (CG) method for solving the symmetric positive definite linear system; in Section 3 we present two approaches to equivalently transforming Equation (2) into symmetric positive definite systems, so that CG can be used for solving Equation (2); finally we report numerical results on two practical problems to illustrate that CG works very well.

Some notations to be used are introduced. Denote by the superscript T the transpose of a matrix or vector, by $(f, g) = f^T g = g^T f$ the usual Euclidean inner product of two real vectors f and g, by $\|\cdot\|$ the spectral norm of a matrix and the vector 2-norm, by $\mathcal{R}^n$ the n-dimensional real space and by I the identity matrix with order clear from the context.

2. The CG method

CG is the most popular iterative method for solving the large symmetric positive definite system [2,6]. For a large symmetric indefinite system, one often uses SYMMLQ, mathematically equivalent to CG, and the Minimal Residual method (MINRES) [5,6]. The CG method can converge very fast and may obtain an approximate solution with prescribed accuracy when iterations are much smaller than the problem size. We briefly review the CG algorithm for an $n \times n$ symmetric positive definite system $Bz = d$.

The CG algorithm

1. Choose an initial guess z_0 to z, and set $r_0 = d - Bz_0$, $p_1 = r_0$.
2. For $k = 1, 2, \ldots$ until convergence

$$\alpha_k = \frac{r_{k-1}^T r_{k-1}}{p_k^T (Bp_k)};$$

$$z_k = z_{k-1} + \alpha_k p_k;$$

$$r_k = r_{k-1} - \alpha_k (Bp_k);$$

$$\beta_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}};$$

$$p_{k+1} = r_k + \beta_k p_k.$$

The CG algorithm only needs a matrix by vector product Bp_k and $12n$ floating point arithmetic operations (flops) at each iteration k and stores four vectors p_k, Bp_k, x_k, r_k possibly plus the non-zero entries of B. In exact arithmetic, the algorithm finds the exact solution z in at most n steps. Even if B is dense, each iteration only costs $2n^2 + 12n$ flops. If B is sparse and has $O(n)$ non-zero entries, each iteration only needs $O(n)$ flops; so total flops are only $O(n^2)$ even if the algorithm is run n iterations. An attractive property of CG is that it may converge after $m \ll n$

iterations [6]. This means that the computational cost is only $O(mn)$ flops when B is sparse, much less than $O(n^3)$ flops needed by a direct solver, a reduction by at least one and even two orders.

If B is ill-conditioned, i.e. the condition number of B is large, CG may converge in many iterations. To speed up the convergence, it is necessary to combine preconditioning techniques with CG, leading to the Preconditioned CG (PCG) method. The aim of preconditioning is to reduce the condition number of the preconditioned system. One of the most popular preconditioners is the incomplete Cholesky factorization of B and its variants [2,6]. For more details on CG and PCG as well as convergence theory, see [2,6].

3. Two equivalent symmetric positive definite forms of Equation (2)

CG is not directly applicable to the singular system (2). We present two approaches to transforming it into equivalent symmetric positive definite linear systems, so that CG can work on them to solve Equation (2).

The first approach needs the following result.

LEMMA 3.1 *The leading* $(n-1)\times(n-1)$ *principal matrix* $\tilde{A}$ *of* A *is positive definite.*

Proof For any non-zero vector $\tilde{x}\in\mathcal{R}^{n-1}$, define $x=(\tilde{x}^T,0)^T\in\mathcal{R}^n$. Then

$$(\tilde{x},\tilde{A}\tilde{x})=(x,Ax).$$

Since the rank of A is $n-1$, A has only one simple zero eigenvalue. So the basis vector u of the null space of A is just the (unnormalized) eigenvector of A associated with the zero eigenvalue. Note that the last entry of x is zero while that of u is not. Therefore, x and u are linearly independent. Let $\hat{u}$ be the resulting vector that orthogonalizes u against x, i.e. $u^T\hat{u}=0$. Then because of the linear independence of x and u, we have

$$\hat{u}=u-\frac{x^Tu}{x^Tx}x=\left(I-\frac{xx^T}{x^Tx}\right)u\neq 0.$$

If $u^Tx\neq 0$, we have

$$x=\frac{x^Tx}{u^Tx}(u-\hat{u}).$$

Note that by construction, $\hat{u}$ is a linear combination of the eigenvectors of A associated with its all positive eigenvalues. Then we obtain

$$\begin{aligned}(\tilde{x},\tilde{A}\tilde{x})=(x,Ax)&=\frac{(x^Tx)^2}{(u^Tx)^2}(u-\hat{u},A(u-\hat{u}))\\&=\frac{(x^Tx)^2}{(u^Tx)^2}(\hat{u},A\hat{u})>0.\end{aligned}$$

If $u^Tx=0$, x itself is a linear combination of the eigenvectors of A associated with all the positive eigenvalues. Therefore, we get

$$(\tilde{x},\tilde{A}\tilde{x})=(x,Ax)>0.$$

■

Combining with Lemma 3.1, we prove next that the singular system (2) can be equivalently reduced to an $(n-1)\times(n-1)$ symmetric positive definite linear system, so that CG or PCG can work.

THEOREM 3.2 *Partition $Ax = b$ of Equation* (2) *as*

$$\begin{pmatrix} \tilde{A} & c \\ c^T & a \end{pmatrix} \begin{pmatrix} \tilde{x} \\ x_n \end{pmatrix} = \begin{pmatrix} \tilde{b} \\ b_n \end{pmatrix}, \tag{5}$$

where

$$\tilde{A} \in \mathcal{R}^{(n-1)\times(n-1)},\ c, \tilde{x} = (x_1, x_2, \ldots, x_{n-1})^T, \tilde{b} = (b_1, b_2, \ldots, b_{n-1})^T \in \mathcal{R}^{n-1}.$$

Let $\tilde{y}$ be the solution of the symmetric positive definite linear system

$$\tilde{A}\tilde{y} = \tilde{b}. \tag{6}$$

Then

$$\tilde{x} = \tilde{y} - n(\tilde{u}^T\tilde{y})\tilde{u}, \tag{7}$$

$$x_n = -\sum_{i=1}^{n-1} x_i, \tag{8}$$

where $\tilde{u} = (1, 1, \ldots, 1)^T \in \mathcal{R}^{n-1}$.

Proof From the properties of A in Equation (2) and the constraints (3) and (4), we have

$$c = -\tilde{A}\tilde{u}, \quad a = -c^T\tilde{u}, \quad x_n = -\tilde{u}^T\tilde{x}, \quad b_n = -\tilde{b}^T\tilde{u}.$$

Note that Equation (5) can be written as

$$\tilde{A}\tilde{x} + cx_n = \tilde{b},$$
$$c^T\tilde{x} + ax_n = b_n.$$

Substituting the expressions of c and x_n into the first equation of the above, we then get

$$\begin{aligned} \tilde{A}\tilde{x} - \tilde{A}\tilde{u}(-\tilde{u}^T\tilde{x}) &= \tilde{A}\tilde{x} + \tilde{A}\tilde{u}\tilde{u}^T\tilde{x} \\ &= \tilde{A}(I + \tilde{u}\tilde{u}^T)\tilde{x} \\ &= \tilde{A}\tilde{y} = \tilde{b}, \end{aligned}$$

where

$$\tilde{y} = (I + \tilde{u}\tilde{u}^T)\tilde{x}.$$

Since from Lemma 3.1 $\tilde{A}$ is positive definite, $\tilde{y}$ exists and is unique.

We can easily verify that $I + \tilde{u}\tilde{u}^T$ has an $n-2$ multiple eigenvalue one and a simple eigenvalue n.

So it is invertible and

$$\tilde{x} = (I + \tilde{u}\tilde{u}^T)^{-1}\tilde{y}.$$

Exploiting the Shermann–Morrison formula [2, p. 50], we get

$$\begin{aligned} \tilde{x} &= (I + \tilde{u}\tilde{u}^T)^{-1}\tilde{y} \\ &= (I - \tilde{u}(1 + \tilde{u}^T\tilde{u})\tilde{u}^T)\tilde{y} \\ &= (I - n\tilde{u}\tilde{u}^T)\tilde{y} \\ &= \tilde{y} - n(\tilde{u}^T\tilde{y})\tilde{u}. \end{aligned}$$

Equation (8) follows from Equation (4) trivially. ■

So we can solve Equation (2) by applying CG to Equation (6) and recovering the solution by Equations (7) and (8).

We now present the second approach.

Note that the vector $e = 1/\sqrt{n}u \in \mathcal{R}^n$ is a normalized (unit length) eigenvector of A associated with the zero eigenvalue. It follows from Equations (3) and (4) that $e^T x = 0$ and $e^T b = 0$. So there hold

$$(I - ee^T)x = x,\ (I - ee^T)b = b. \tag{9}$$

The solution of $Ax = b$ satisfies the projected linear system

$$(I - ee^T)A(I - ee^T)x = (I - ee^T)b = b. \tag{10}$$

Note that $e^T x = 0$ means $x \in e^\perp$, the orthogonal complement of e with respect to $\mathcal{R}^n$.

The matrix $(I - ee^T)A(I - ee^T)$ is obviously singular. However, since A has a simple zero eigenvalue and e is the associated normalized eigenvector, the eigenvalues of $(I - ee^T)A(I - ee^T)|_{e^\perp}$, $(I - ee^T)A(I - ee^T)$ restricted to the subspace $e^\perp$, are just all the $n-1$ positive ones of A. So $(I - ee^T)A(I - ee^T)|_{e^\perp}$ is symmetric positive definite. When constrained on $x \in e^\perp$, Equation (10) has a unique solution and satisfies Equation (2). Note that $b \in e^\perp$ too. So the m-dimensional Krylov subspace $\mathcal{K}_m((I - ee^T)A(I - ee^T), (I - ee^T)b) = \mathcal{K}_m$ generated by $(I - ee^T)b$ and $(I - ee^T)A(I - ee^T)$ is contained in $e^\perp$, i.e. it only contains the information on the eigenvectors corresponding to the positive eigenvalues of A. As a result, CG works for Equation (10) on $\mathcal{K}_m$. This can be done once the initial approximate solution $x_0 = 0$ in CG so that b is the starting vector for the Krylov subspace.

We should emphasize that, unlike a general CG, here we can only take the initial approximate solution $x_0 = 0$ in CG for Equation (10) and this guarantees that $\mathcal{K}_m \subset e^\perp$ and otherwise not. When m iterations are run, CG generates an approximate solution $x_m \in \mathcal{K}_m \subset e^\perp$. Furthermore, the residual

$$r_m = (I - ee^T)b - (I - ee^T)A(I - ee^T)x_m = (I - ee^T)(b - Ax_m) = b - Ax_m,$$

i.e. the residual for Equation (10) by CG is just that for $Ax = b$.

In the CG algorithm for Equation (10), at iteration k we need to compute the matrix by vector product

$$(I - ee^T)A(I - ee^T)p_k.$$

Obviously, it is prohibitive to form $(I - ee^T)A(I - ee^T)$ explicitly as this is too expensive and results in a large dense matrix. A much cheaper and correct way is to compute it in the form

$$(I - ee^T)A(I - ee^T)p_k = Ax - (e^T p_k)Ae - e(e^T Ap_k) - (e^T p_k)(e^T Ae)e.$$

It is easily verified that, besides Ap_k, we need a matrix by vector product Ae and $12n$ flops. Compared with the computational cost of CG in Section 2, each iteration of CG for Equation (10) costs twice that of CG for Equation (6). However, observe that e is fixed at all iterations. So we only need to compute Ae once before iteration starts and store it for use in iterations. In such a way, the number of matrix by vector products in CG for Equation (10) is only one more than those in CG for Equation (6), assuming that the same iterations of CG are run. We can thus save several flops.

A few words are about convergence of CG for Equations (6) and (10). It is impossible to know which one is faster in advance. In fact, based on the eigenvalue interlacing theorem on a symmetric matrix, the eigenvalues of $\tilde{A}$ interlace those of A, so the largest eigenvalue of $\tilde{A}$ is no more than that of A and the smallest one of $\tilde{A}$ is no more that the smallest positive one of the singular A. So the

spectral condition number of $\tilde{A}$ is either bigger or smaller than that of $(I - ee^T)A(I - ee^T)|_{e^\perp}$. As a consequence, in terms of iterations, either one of them may be faster than the other.

Theoretically, the eigenvalues of $A_{e^\perp}$ are the same as those of $(I - ee^T)A(I - ee^T)|_{e^\perp}$. Furthermore, when b has no component in the direction of e, the Krylov subspace $\mathcal{K}_m(A, b)$ generated by b and A does too and thus contains only the information on the eigenvectors associated with the $n - 1$ positive eigenvalues of A. So in exact arithmetic, CG works for $Ax = b$ directly as if the zero eigenvalue of A disappears and computes the same x_m from $\mathcal{K}_m(A, b)$. However, this CG approach is not reliable and may converge much more slowly than the CG for Equation (10). The reasons are twofold. First, b and A may have errors that are far bigger than the unit roundoff, so A is not exactly singular and b has a small component in the eigenvector of A associated with its smallest eigenvalue, which is small but numerically not zero. So CG for $Ax = b$ actually works on a highly ill-conditioned positive definite linear system. The second is that even if A and b are exact, in finite precision, CG for a positive definite linear system is unstable though it still converges. As iterations proceed, the information on the eigenvector associated with the zero eigenvalue of A accumulates, becomes substantial in $\mathcal{K}_m(A, b)$ and cannot be ignored. As a consequence, CG for $Ax = b$ will gradually work on a singular system and the zero eigenvalue of A will gradually play an essential role to possibly make CG fail or converge extremely slowly. In contrast, CG for the explicitly projected system (10) restricted to $e^\perp$ removes the eigenvector associated with the smallest eigenvalue of A both theoretically and numerically, and the smallest eigenvalue of A does not affect the convergence, as CG for Equation (10) orthogonalizes against e explicitly at each iteration.

Finally, we should point out that the MINRES method [5] and the mathematically equivalent Conjugate Residual (CR) method [6] can be used to solve Equations (6) and (10) too.

4. Numerical examples

We apply CG to Equations (6) and (10) on two practical problems [7]. Numerical experiments were run on an Intel Pentium IV with CPU 2.4 GHz and RAM 512 MB under the Window XP system using Matlab 7.1 with the machine precision $\epsilon = 2.22 \times 10^{-16}$. We modified pcg.m in Matlab 7.1 slightly for our use.

CG stops when the relative residual norms satisfy

$$\frac{\|\tilde{A}\tilde{y}_m - \tilde{b}\|}{\|\tilde{b}\|}, \frac{\|Ax_m - b\|}{\|b\|} \leq tol$$

where $\tilde{y}_m$ and x_m are the approximate solutions obtained by CG for Equations (6) and (10), respectively, and tol is a user-prescribed tolerance.

In the following tables, *time* denotes the CPU time (seconds), m denotes the iteration steps of CG when the convergence occurs and $error$ denotes the actual relative residual norms.

Example 4.1 The matrix A is of order $n = 51{,}038$ and has 4,06,435 non-zero entries [7]. The largest eigenvalue and the second smallest eigenvalue of A are computed by the implicitly restarted refined Arnoldi algorithm in [3] and they are approximately 6.1218 and 1.6803×10^{-4}. So the spectral condition number of $(I - ee^T)A(I - ee^T)|_{e^\perp}$ is 3.6443×10^4. The largest and smallest eigenvalues of $\tilde{A}$ are 6.1218 and 6.4715×10^{-6}, and its spectral condition number is 9.4597×10^6. So it is expected that CG for Equation (10) uses fewer iterations than CG for Equation (6) does. The results listed in Table 1 confirm this. Both CGs were very efficient and converged in no more than $n/30$ iterations, much smaller than n. We found that the sums of entries of x_ms obtained by CG for Equation (10) meet the constraint (3) very accurately with errors no more than 10^{-8}.

Table 1. *Left*: Conjugate Gradient CG for Equation (6) in Example 4.1; *Right*: CG for Equation (10) in Example 4.1.

tol	Time	*m*	Error	*tol*	Time	*m*	$\sum_{i=1}^{n} x_i$	Error
10^{-5}	16.08	801	9.8×10^{-6}	10^{-5}	26.23	529	-6.1×10^{-9}	1.0×10^{-6}
10^{-6}	17.74	893	9.9×10^{-7}	10^{-6}	31.72	646	-7.0×10^{-9}	9.9×10^{-7}
10^{-7}	19.55	985	9.8×10^{-8}	10^{-7}	35.70	736	-7.6×10^{-9}	9.7×10^{-8}
10^{-8}	20.92	1055	9.8×10^{-9}	10^{-8}	41.28	839	-8.3×10^{-9}	9.7×10^{-9}
10^{-9}	22.20	1108	9.9×10^{-10}	10^{-9}	44.36	913	-8.7×10^{-9}	9.7×10^{-9}

Table 2. *Left*: CG for Equation (6) in Example 4.2; *Right*: CG for Equation (10) in Example 4.2.

tol	Time	*m*	Error	*tol*	Time	*m*	$\sum_{i=1}^{n} x_i$	Error
10^{-5}	212	1521	9.99×10^{-6}	10^{-5}	210	895	5.4×10^{-7}	9.97×10^{-6}
10^{-6}	240	1706	9.97×10^{-7}	10^{-6}	295	1270	8.4×10^{-7}	9.99×10^{-7}
10^{-7}	267	1898	9.98×10^{-8}	10^{-7}	343	1479	9.5×10^{-7}	9.99×10^{-8}
10^{-8}	297	2114	9.88×10^{-9}	10^{-8}	379	1691	1.0×10^{-6}	9.95×10^{-9}
10^{-9}	322	2293	9.95×10^{-10}	10^{-9}	406	1848	1.0×10^{-6}	9.97×10^{-9}

Although CG for Equation (10) used fewer iterations than CG for Equation (6), it used more CPU time. This is because A is very sparse and the cost of matrix by vector products is not dominant any longer, so that the cost of each iteration of CG for Equation (10) is nearly twice that of CG for Equation (6). CPU timings in Table 1 reflect this.

Example 4.2 The matrix A is of order $n = 3{,}00{,}343$ and has 2,100,462 non-zero entries [7]. The largest eigenvalue and the second smallest eigenvalue of A are computed by the implicitly restarted refined Arnoldi algorithm in [3] and they are approximately 6.6167 and 6.200×10^{-5}. So the spectral condition number of $(I - ee^T)A(I - ee^T)|_{e^\perp}$ is 9.9463×10^4. The largest and smallest eigenvalues of $\tilde{A}$ are 6.6167 and 402675×10^{-6}, and its spectral condition number is 1.445×10^6. So it is expected that CG for Equation (10) uses fewer iterations than CG for Equation (6). Table 2 reports the results. CG for both Equations (6) and (10) converged in no more than $n/150$ iterations, far smaller than n. Compared with the results of Example 4.1, CG for the larger problem works more efficiently. We also observed all phenomena similar to those for Example 4.1.

Acknowledgements

The author thanks two referees for their valuable comments and suggestions that helped in improving this paper considerably. Thanks also go to the authors of [7] for providing the problem and data. Supported by the National Natural Science Foundation (No. 10771116) and the Doctoral Program of the Ministry of Education (No. 20060003003).

References

[1] C. Collins and K. Stephenson, *A circle packing algorithm*, Comput. Geom. 25 (2003), pp. 233–256.
[2] G.H. Golub and C.F. van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, London, 1996.
[3] Z. Jia, *Polynomial charaterizations of the approximate eigenvectors by the refined Arnoldi method and an implicitly restarted refined Arnoldi algorithm*, Linear Algebra Appl. 287 (1999), pp. 191–214.
[4] L. Kharevych, B. Springborn, and P. Schröder, *Discrete conformal mapping via circle patterns*, ACM Trans. Graphics 25 (2006), pp. 412–438.
[5] C.C. Paige and M.A. Saunders, *Solution of a sparse indefinite systems of linear equations*, SIAM J. Numer. Anal. 12 (1975), pp. 617–629.
[6] Y. Saad, *Iterative Solutions of Large Sparse Linear Systems*, 2nd ed., SIAM, PA, 2003.
[7] Y. Yang, J. Kim, F. Luo, S. Hu, and X. Gu, *Optimal surface parameterization using inverse curvature map*, IEEE Trans. Visu. Comput. Graphics, accepted.