USE OF CONJUGATE GRADIENT METHODS FOR THE GREAT-CIRCLE REDUCTION IN FUTURE SPACE ASTROMETRY MISSIONS

A.E. Ilin

Pulkovo Observatory, Russian Academy of Sciences, St.Petersburg 196140, Russia

ABSTRACT

Over the past few years it has been demonstrated that iterative methods for finding the solution of linear systems of equations offer major advantages over the 'direct' methods in various physical and numerical problems. We discuss here improvements in conjugate gradients algorithms applied to the reduction on the reference great circles for the Hipparcos or GAIA-like type of observations. We consider the CPU time and the storage requirements for various conjugate gradients methods. We recommend use of the stabilized version of the bi-conjugate-gradients algorithm with left Jacobi preconditioning. This algorithm requires 67 per cent greater storage than the simplest CG algorithm without preconditioning, but is typically three times faster and has a better convergence. Our results can be used in future space astrometric projects.

Key words: space astrometry; data reduction.

1. THE PROBLEM STATEMENT

The problem of the great-circle reduction reduces to that of finding, in a least-squares sense, $\mathbf{x}$ the vector of the small unknown corrections to the approximate values satisfying a linearized observation equation (Marel 1988, Marel & Petersen 1992):

$$\mathbf{y} = \mathbf{A}\mathbf{x} = \mathbf{A}_\mathrm{a}\ \mathbf{x}_\mathrm{a} + \mathbf{A}_\mathrm{s}\mathbf{x}_\mathrm{s} + \mathbf{A}_\mathrm{i}\mathbf{x}_\mathrm{i} \qquad (1)$$

with the $m \times n$ ($m \gg n$) design matrix $\mathbf{A}$ of partial derivatives, and $\mathbf{y}$ the vector with linearized observations, i.e. the observed value minus a value computed from approximate data on the unknown parameters. The unknowns $\mathbf{x}$ and design matrix $\mathbf{A}$ are partitioned into an attitude, star and instrument part, respectively denoted by indices a, s and i. The submatrices $\mathbf{A}_\mathrm{a}$ and $\mathbf{A}_\mathrm{s}$ are very sparse, each of them containing only one non-zero element per row. On the other hand $\mathbf{A}_\mathrm{i}$ is almost completely filled (Figure 1).

The solution $\mathbf{x}$ of the least-squares problem can be computed from the so-called normal equations:

$$\begin{pmatrix} \mathbf{N}_{aa} & \mathbf{N}_{as} & \mathbf{N}_{ai} \\ \mathbf{N}_{sa} & \mathbf{N}_{ss} & \mathbf{N}_{si} \\ \mathbf{N}_{ia} & \mathbf{N}_{is} & \mathbf{N}_{ii} \end{pmatrix} \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \mathbf{x}_\mathrm{s} \\ \mathbf{x}_\mathrm{i} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_\mathrm{a} \\ \mathbf{b}_\mathrm{s} \\ \mathbf{b}_\mathrm{i} \end{pmatrix} \qquad (2)$$

with $\mathbf{N}_{pq} = \mathbf{A}_\mathrm{p}^\mathrm{T}\mathbf{W}_{yy}\mathbf{A}_\mathrm{q}$ and $\mathbf{b}_\mathrm{p} = \mathbf{A}_\mathrm{p}^\mathrm{T}\mathbf{W}_{yy}\mathbf{y}$ for $p, q = a, s, i$, and with weight matrix $\mathbf{W}_{yy}$, where:

$$\sigma_0{}^2\mathbf{W}_{yy}^{-1} = \mathbf{C}_{yy} = E\{(\mathbf{y} - E\{\mathbf{y}\})^\mathrm{T}(\mathbf{y} - E\{\mathbf{y}\})\} \qquad (3)$$

and σ_0 is the variance of unit weight.

2. CHOICE OF A SOLUTION METHOD

Iterative methods for the solution of huge and sparse linear systems of equations require less storage and can use simpler data structure then direct methods. They fully exploit the sparsity in the system of equations. Moreover in many problems the iterative solution can be computed faster than the direct solution.

In the last several years several high quality packages became available (Templates, NSPCG, ITPACK, PIM(Cunha & Hopkins 1996)). After examining several options (Ilin 1997) we have decided to use the PIM package.

3. THE PARALLEL ITERATIVE METHODS

The Parallel Iterative Methods (PIM) (Cunha & Hopkins 1996) is a collection of FORTRAN 77 routines designed to solve systems of linear equations on parallel computers using a variety of iterative methods. PIM offers a number of iterative methods, including:

- conjugate-gradients (CG);
- conjugate-gradients for normal equations with minimization of the residual norm (CGNR);
- conjugate-gradients for normal equations with minimization of the error norm (CGNE);
- bi-conjugate-gradients (Bi-CG);
- conjugate-gradients squared (CGS);
- the stabilized version of Bi-Conjugate-Gradients (Bi- STAB) the restarted, stabilized version of Bi-Conjugate-Gradients (RBi-CGSTAB);

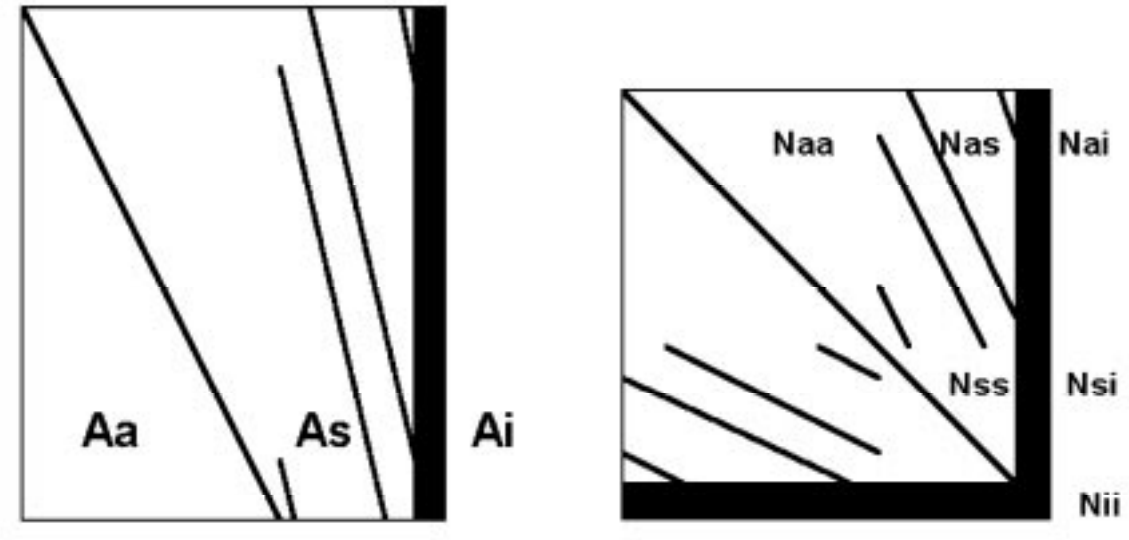

Figure 1. Non-zero structure of the design and normal matrices.

Table 1. CPU-time, number of iterations and storage.

	preconditioning				storage
Method	no	left	right	symmetric	matrix+
CG	102.2(699)	* (2000)	* (2000)	34.4(207)	6n
CGEV	* (2000)	* (2000)	* (2000)	34.0(207)	6n
BiCG	* (2000)	* (2000)	* (2000)	* (2000)	8n
CGS	* (2000)	41.8 (175)	42.4(180)	43.8(179)	10n
Bi-CGSTAB	251.0(1168)	32.0(138)	51.4(220)	42.6(176)	10n
RBi-CGSTAB	172.7(491)	27.8 (72)	50.5(129)	38.5 (94)	(6+2k)n
RGMRES	* (2000)	46.6 (46)	* (2000)	1396(1333)	(4+k)n
RGMRESEV	* (2000)	46.7(46)	* (2000)	1400(1333)	(4+k)n
RGCR	* (2000)	4745.0(1984)	* (2000)	* (2000)	(5+2k)n
CGNR	* (2000)	* (2000)	* (2000)	* (2000)	6n
CGNE	* (2000)	* (2000)	* (2000)	* (2000)	6n
QMR	* (2000)	* (2000)	* (2000)	* (2000)	11n
TFQMR	* (2000)	* (2000)	* (2000)	* (2000)	10n

- the restarted, generalized minimal residual (RGMRES);
- the restarted, generalized conjugate residual (RGCR);
- the quasi-minimal residual with coupled two-term recurrences (QMR);
- the transpose-free quasi-minimal residual (TFQMR) and Chebyshev acceleration.

The routines allow the use of preconditioning; the user may choose to use left-, right- or symmetric-preconditioning. Several stopping criteria are also available.

4. PRECONDITIONING

We used the simplest preconditioning that consist of just the diagonal of the coefficient matrix (Jacobi preconditioning):

$$\mathbf{Q}_1\mathbf{N}\mathbf{Q}_2\mathbf{x} = \mathbf{Q}_1\mathbf{b}, \tag{4}$$

where $\mathbf{Q}_1 = \mathrm{diag}(\mathbf{A})^{-1}$ for left preconditioning, $\mathbf{Q}_2 = \mathrm{diag}(\mathbf{A})^{-1}$ for right preconditioning, and $\mathbf{Q}_1 = \mathbf{Q}_2 = \mathrm{diag}(\mathbf{A})^{-1/2}$ for symmetric preconditioning.

5. STOPPING CRITERIA

Ideally we would like to stop when the magnitudes of the error $\mathbf{e}^{(\mathrm{i})} = \mathbf{x}^{(\mathrm{i})} - \mathbf{x}$ fall below a user-supplied threshold. But $\mathbf{e}^{(\mathrm{i})}$ is hard to estimate directly, so we use the *residual* $\mathbf{r}^{(\mathrm{i})} = \mathbf{A}\mathbf{x}^{(\mathrm{i})} - \mathbf{b}$ instead, which is more readily computed. We used following criterion:

$$\|\mathbf{r}^{(i)}\| \leq \varepsilon_{\mathrm{stop}} \cdot \|\mathbf{b}\| \qquad (5)$$

This criterion yields the forward error bound:

$$\|\mathbf{e}^{(\mathrm{i})}\| \leq \|\mathbf{A}^{-1}\| \cdot \|\mathbf{r}^{(\mathrm{i})}\| \leq \varepsilon_{\mathrm{stop}} \cdot \|\mathbf{A}^{-1}\| \cdot \|\mathbf{b}\| \qquad (6)$$

6. TEST MATRIX

We used matrices generated by our package of observations simulation for the space astrometry project Struve. We consider here the case with 3000 stars within the circular scan of 1 degree width, with 11 instrument parameters and with 4050 frames. The number of observation equations is 67487. The 'input catalogue' was obtained by adding systematic and accidental noise to the true star coordinates (see Ilin 1997 for details).

The special compressed format was used to storage only nonzero elements of the sparse matrices $\mathbf{A}$ and $\mathbf{N}$.

7. NUMERICAL RESULTS

Table 1 presents CPU time in seconds (sequential Pentium- 133), the number of iterations (in brackets) and the storage requirements for several CG-like methods. Star indicates that the method did not converge in the maximum allowed number (2000) of iterations or that the method failed to converge. For restarted methods the basis number $k = 10$ was used. We took $\varepsilon_{\mathrm{stop}} = 10^{-8}$ as the stopping criterion.

8. PARALLELISM

Everyone can easily implement CG-like algorithms for multiprocessor computers. The basic time-consuming kernels of iterative schemes are:

- inner product;
- vector updates;
- matrix-vector products;
- preconditioning solves.

A number of computational schemes in the PIM library have been specially reorganized for parallelism to reduce the idle time for some processors.

9. COMPARISION WITH LSQR METHOD

The LSQR algorithm (Paige & Saunders 1982) works directly on the design matrix, using column scaling as a preconditioning method. We have solved the problem (1) with our test matrix by the LSQR method. This method requires 1086 iterations and 1135 s of CPU time. Therefore it works slower than most of CG-like methods. However Marel 1988 notes that the LSQR algorithm is not significantly faster than Choleski factorization realized in the Hipparcos great-circle reduction. Therefore considering CG-like methods could give a considerable gain for the data reduction in the future space astrometric missions.

10. SUMMARY

To solve adequately the problem of the great-circle reduction in the future space astrometric projects involving several million programme stars we recommend use of nonstationary iterative (conjugate-gradients-like) methods.

The left or symmetrical Jacobi preconditioning reduces the number of iterations needed, and improves the convergence of all conjugate gradients methods.

We recommend use of the stabilized version of the biconjugate-gradient algorithm with left Jacobi preconditioning. This algorithm requires 67 per cent greater storage than the simplest CG algorithm without preconditioning, but is typically three times faster and has a better convergence.

REFERENCES

da Cunha, R., Hopkins T. 1996, Appl. Numer. Math., Pim 2.0 the parallel iterative method package for systems of linear equations user's guide (fortran 77 version). Technical Report 1-96, Computing Laboratory, University of Kent, Canterbury, Kent, UK

Ilin A.E. 1997, in 'Struve' Space Astrometric System. Scientific Grounds of the Project, Part 2 (in press)

van der Marel H. 1988, PhD Thesis

van der Marel H., Peteresen C. 1992, A&A, 258, 60

Paige C.C., Saunders M.A. 1982, ACM Trans. Math. Soft., 8, 43