Computing the trace of the inverse of sparse matrices using modified moments

Gérard MEURANT

June 24, 2008

Introduction

Let A be symmetric (positive definite for the sake of simplicity)

There are applications (QCD, study of fractals, GCV and its applications) where it is desired to compute bounds or estimates of the trace of the inverse $tr(A^{-1})$ and/or the determinant $\det(A)$ of large sparse matrices

Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of A
We have

$$tr(A^r) = \sum_{i=1}^{n} \lambda_i^r$$

We are interested in $r = -1$

The sum $\sum_{i=1}^{n} \lambda_i^r$ can be written as a Riemann–Stieltjes integral

$$tr(A^r) = \mu_r = \int_a^b \lambda^r \, d\alpha, \quad a \le \lambda_1, \ \lambda_n \le b$$

where the (unknown) measure α is given as

$$\alpha(\lambda) = \sum_{j=1}^{n} H(\lambda - \lambda_j)$$

H is the unit step function, $H(\lambda) = 0, \ \lambda < 0, \ \ H(\lambda) = 1, \ \lambda \ge 0$

The values μ_r are the moments related to α

We wish to compute

$$\mu_{-1} = \int_a^b \frac{1}{\lambda} \, d\alpha$$

Bai and Golub results

The main idea is to use Gauss quadrature to estimate or bound the integral

Bai and Golub (1997) use three moments $r = 0, 1, 2$

$$\mu_0 = n, \quad \mu_1 = \mathrm{tr}(A) = \sum_{i=1}^{n} a_{i,i}, \quad \mu_2 = \mathrm{tr}(A^2) = \sum_{i,j=1}^{n} a_{i,j}^2 = \|A\|_F^2$$

to analytically compute the nodes and weights of a Gauss–Radau rule and then bounds for the integral of $1/\lambda$

Bai and Golub results 2

Their result is

$$\begin{pmatrix}\mu_1 & n\end{pmatrix}\begin{pmatrix}\mu_2 & \mu_1 \\ b^2 & b\end{pmatrix}^{-1}\begin{pmatrix}n \\ 1\end{pmatrix} \leq \mathrm{tr}(A^{-1}) \leq \begin{pmatrix}\mu_1 & n\end{pmatrix}\begin{pmatrix}\mu_2 & \mu_1 \\ a^2 & a\end{pmatrix}^{-1}\begin{pmatrix}n \\ 1\end{pmatrix}$$

This result is nice since the moments μ_0, μ_1, μ_2 are easy to compute, but in many cases, the bounds are far from being sharp

Gauss quadrature rules

Associated with the measure α there exist orthonormal polynomials p_k

$$\int_a^b p_i(\lambda)p_j(\lambda)\ d\alpha(\lambda) = \begin{cases} 1 & \text{if i=j} \\ 0 & \text{otherwise} \end{cases}$$

They satisfy a three-term recurrence

$$\gamma_j p_j(\lambda) = (\lambda - \omega_j)p_{j-1}(\lambda) - \gamma_{j-1}p_{j-2}(\lambda), \quad j = 1, 2, \dots, n$$

The Jacobi matrix is

$$J_k = \begin{pmatrix} \omega_1 & \gamma_1 & & & \\ \gamma_1 & \omega_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{k-2} & \omega_{k-1} & \gamma_{k-1} \\ & & & \gamma_{n-1} & \omega_k \end{pmatrix}$$

Gauss quadrature rules 2

The eigenvalues of J_k (which are also the zeros of p_k) are the nodes t_j of the Gauss quadrature rule, the weights w_j are the squares of the first elements of the normalized eigenvectors of J_k

The main question is:

Can we compute the Jacobi matrix from the moments?

If we can do this, we have the algorithm:

moments ($r \geq 0$) $\Rightarrow$ Jacobi matrix $\Rightarrow$ eigensystem $\Rightarrow$ nodes and weights $\Rightarrow$ estimate

The Chebyshev algorithm

An answer to our question has been given by Chebyshev (1859) who described an algorithm to obtain the coefficients of the orthogonal polynomials from the moments

One needs $2k$ moments to compute J_k

The algorithm (as it is described today) uses the Cholesky factorization of the (Hankel) moment matrix $m_{i,j} = \mu_{i+j-2}$

However, the map from the moments to the coefficients is ill-conditioned (see Gautschi) and the Chebyshev algorithm is often unstable

The modified Chebyshev algorithm

This algorithm was developed by J. Wheeler in 1974, see also Sack and Donovan (1972)

Let π_k be a family of known orthogonal polynomials satisfying

$$b_{k+1}\pi_{k+1}(\lambda) = (\lambda - a_{k+1})\pi_k(\lambda) - c_k\pi_{k-1}(\lambda)$$

The modified moments are

$$m_l = \int_a^b \pi_l(\lambda)\, d\alpha$$

which have to be known

The algorithm uses mixed moments which are

$$\sigma_{k,l} = \int_a^b p_k(\lambda)\pi_l(\lambda)\, d\alpha(\lambda)$$

The modified Chebyshev algorithm 2

To obtain relations between ω_i, γ_i and the mixed moments, we use the orthogonality properties of the polynomials

To compute m coefficients:

$$\sigma_{-1,l} = 0,\ l = 1, \dots, 2m-2, \quad \sigma_{0,l} = m_l,\ \ l = 0, 1, \dots, 2m-1$$

$$\omega_1 = a_1 + b_1 \frac{m_1}{m_0},\ \ \eta_0 = m_0$$

for $k = 1, \dots, m-1$

for $l = k, \dots, 2m-k-1$

$$\sigma_{k,l} = b_{l+1}\sigma_{k-1,l+1} + (a_{l+1} - \omega_k)\sigma_{k-1,l} + c_l \sigma_{k-1,l-1} - \eta_{k-1}\sigma_{k-2,l}$$

then,

$$\omega_{k+1} = a_{k+1} + b_{k+1}\frac{\sigma_{k,k+1}}{\sigma_{k,k}} - b_k \frac{\sigma_{k-1,k}}{\sigma_{k-1,k-1}}$$

$$\eta_k = b_k \frac{\sigma_{k,k}}{\sigma_{k-1,k-1}}, \quad \gamma_k = \sqrt{\eta_k}$$

Implementation

As auxiliary polynomials, we use the shifted Chebyshev polynomials:

$$C_0(\lambda) \equiv 1, \quad \left(\frac{\lambda_n - \lambda_1}{2}\right) C_1(\lambda) = \lambda - \left(\frac{\lambda_n + \lambda_1}{2}\right)$$

$$\left(\frac{\lambda_n - \lambda_1}{4}\right) C_{k+1}(\lambda) = \left(\lambda - \frac{\lambda_n + \lambda_1}{2}\right) C_k(\lambda) - \left(\frac{\lambda_n - \lambda_1}{4}\right) C_{k-1}(\lambda)$$

Computing the modified moment m_l is computing the trace of the matrix $C_l(A)$

Pb: we have to compute the product of "sparse " matrices, but we have to store only the last 2 of them

Numerical experiments

Example: Poisson equation

$n = 36$, $tr(A^{-1}) = 13.7571$, Bai and Golub bounds

$$10.2830 \leq tr(A^{-1}) \leq 24.3776$$

Moments

k	bound
1	9.0000
2	11.3684
3	12.5714
4	13.1581
5	13.4773
6	13.6363
7	13.7139
8	13.7452
9	13.7550
10	13.7568

After $k = 10$ the moment matrices are no longer positive definite

Modified Moments

k	bound
1	9.0000
2	11.3684
3	12.5714
4	13.1581
5	13.4773
6	13.6363
7	13.7139
8	13.7452
9	13.7550
10	13.7568
11	13.7571

$n = 900$, $tr(A^{-1}) = 512.6442$, Bai and Golub bounds

$$261.003 \leq tr(A^{-1}) \leq 8751.76$$

The Chebyshev algorithm breaks down after $k = 10$

Modified Moments

k	bound
5	400.0648
10	463.2560
15	489.5383
20	502.0008
25	508.0799
30	510.9301
35	512.1385
40	512.5469

- We can avoid computing the matrices $C_l(A)$ and then their traces by using a Monte Carlo technique:
  Use random vectors z whose components are 1 and -1 with probability $1/2$, then $z^T Bz$ is an estimator of $tr(B)$
  One just needs to compute $C_l(A)z$ and an inner product
- The same techniques can be used to estimate $\det(A)$ by remarking that $\det(A) = exp[tr(\ln(A))]$