Projection methods for estimating the diagonal of a matrix inverse
honoring Claude Brezinski and Sebastiano Saetzu

E. Gallopoulos\textsuperscript{1}
in collaboration with
Vasilis Kalantzis\textsuperscript{1}, Costas Bekas\textsuperscript{2} and Alessandro Curioni\textsuperscript{2}

\textsuperscript{1}Computer Engineering & Informatics Dept.
University of Patras, Greece
and \textsuperscript{2}IBM Zurich

SC’11
Oct. 14, 2011
Stochastic estimation

Solving systems with mrhs

Seed methods

Block seed approach

The projBCG approach

Numerical experiments

Conclusions - plans
Sought values

\[ A \rightarrow \text{diag}(P(A)) \]

\[ A \rightarrow \text{trace}(P(A)) \]

where \( P(A) = A^{-1} \).

- geology
- signal processing
- portfolio management
- uncertainty quantification (SAN09, RW06)
- network science (Benzi, Estrada, Hatano, Higham, Tsourakakis, ...)

Could have chosen simpler or more complicated functions - 
\( A^3, e^A, \sinh(A), ... \) [M. Benzi, L. Reichel, M. Redivo-Zaglia, C. Brezinski, M. Mitrouli, P. Fika, ...]
Sought values

\[ A \rightarrow \text{diag}(P(A)) \]

\[ A \rightarrow \text{trace}(P(A)) \]

where \( P(A) = A^{-1} \).

- geology
- signal processing
- portfolio management
- uncertainty quantification (SAN09, RW06)
- network science (Benzi, Estrada, Hatano, Higham, Tsourakakis, ...)

Could have chosen simpler or more complicated functions -
\((A^3, e^A, \sinh(A), \ldots)\) (M. Benzi, L. Reichel, M. Redivo-Zaglia, C. Brezinski, M. Mitrouli, P. Fika, ...)
Direct approaches

\[ A \xrightarrow{\text{invert}} A^{-1} \xrightarrow{\text{x-diagonal}} D(A^{-1}) \]

For \( A \) spd:

\[ A \xrightarrow{\text{Cholesky}} (LL^T)^{-1} \xrightarrow{\text{x-diagonal}} \{ e_i^T (LL^T)^{-1} e, i = 1, \ldots, n \} \]

Observations

- **Pros**: straightforward, BLAS-3 based
- **Cons**: cubic ops, quadratic storage, "matrix bound"
- **Research**: cost reduction efforts based on matrix structure (TS11, L^+11)
Observations

- Important matrix problem: Estimate diagonal or trace of matrix inverse
- Recent scheme “reducing complexity from cubic to quadratic” (BCF09)
- Based on (BKS07) on estimating matrix diagonal.

Low Cost High Performance Uncertainty Quantification

C. Bekas, A. Curioni
IBM Research - Zurich
Säumerstrasse 4
Rüschlikon, Switzerland
bek,cur@zurich.ibm.com

I. Fedulova
IBM Russian Systems and Technology Laboratory - Moscow
Obrucheva 30/1, 117485, Russia
irina@ru.ibm.com
Key ingredients

- Numerical – Iterative refinement
- Numerical – Iterative matrix method
- Stochastic – Diagonal estimator
- Computation – Parallel processing

Complexity down from cubic to quadratic
730 TFLOPS on 72 Blue Gene/P Racks
Approach appears successful (more ways than one!)
Diagonal of inverse estimation (BCF09)

Working hypotheses

Matrix is spd and typically well-conditioned.

- Estimates diagonal entries of inverse matrix by means of few, relative to matrix size, $\text{MV}'s$ of inverse matrix with $s \ll n$ carefully selected vectors.
- Replaces Cholesky with CG + mixed precision iterative refinement;
- Low precision CG implementation, $P_{\text{low}} \Rightarrow$ exploits computer systems with fast low-precision arithmetic (Cell, GPU’s);
- Low precision also leads to smaller communication overheads;
- High utilization of computational resources and scalability.
Stochastic estimation of inverse diagonal and trace

Accept slightly abusive notation \( P(A, z) := A^{-1}z \).

Given \( z_1, \ldots, z_s \) Rademacher vectors, where \( s \leq n \). Then

\[
H_s(P(A)) = \frac{1}{s} \sum_{i=1}^{s} (z_i^\top P(A, z_i))
\]

is minimum variance estimator for \( \text{tr} \( \hat{A} \) \) (Hut89).

If \( z_i \)'s are as above then the inverse diagonal can be estimated as

\[
D_s(P(A)) = [\sum_{i=1}^{s} (z_i \odot P(A, z_i))] \odot [\sum_{i=1}^{s} z_i \odot z_i]
\]

where \( \odot, \odot \) are Hadamard operations (BKS07).

Once \( P(A, z) \) is set, algorithms can easily be constructed
Statistical diagonal estimation

**Output:** Approximate inverse diagonal $\mathcal{D}(A^{-1})$

For $k = 1, \ldots, s$ do

Generate $z^{(k)}$

end for

$$t = \sum_{k=1}^{s} P(A, z^{(k)}) \odot z^{(k)}$$

$$q = \sum_{k=1}^{s} z^{(k)} \odot z^{(k)}$$

$$\mathcal{D}_s(A^{-1}) = t \odot q$$

**Bound on samples for trace (AT11)**

For spd $A$, an upper bound on $s$ to achieve relative error $\epsilon$ with probability greater than $1 - \delta$ is $6\epsilon^{-2} \ln(2\text{rank}(A)/\delta)$. 
Choosing \([z_1, \ldots, z_s]\)

It holds that

\[
(D_s(A^{-1}))_i = (\text{diag}(A^{-1}))_i + \sum_{i \neq j} (A^{-1})_{ij} \frac{\sum_{k=1}^{s} z_i^{(k)} z_j^{(k)}}{\sum_{k=1}^{s} (z_i^{(k)})^2}
\]

**Note:** If \(\sum_{k=1}^{s} z_i^{(k)} z_j^{(k)}\) converges to zero for any non-zero \((A^{-1})_{ij}, i \neq j\), then each coefficient of entry \((A^{-1})_{ij}\) will converge to zero.

If the \(i\)th row of \(Z\) is orthogonal to all those rows \(j\) of \(Z\) for which \((A^{-1})_{ij} \neq 0\), then the estimator for the diagonal of \(A^{-1}\) will yield an exact result for \((A^{-1})_{ii}\). (BKS07)
A library of trace estimators (AT11)

<table>
<thead>
<tr>
<th>Trace estimator</th>
<th>notes on $z_i$, etc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>i. random, i.i.d. entries</td>
</tr>
<tr>
<td>$G_s(P(A)) = \frac{1}{s} \sum_{i=1}^{s} (z_i^T P(A, z_i))$</td>
<td>from $\mathcal{N}(0, 1)$</td>
</tr>
<tr>
<td>normalized Rayleigh quotient</td>
<td>i. random, $z_i^\top Z_i = n$ and</td>
</tr>
<tr>
<td>$R_s(P(A)) = \frac{1}{s} \sum_{i=1}^{s} (z_i^T P(A, z_i))$</td>
<td>$\mathbb{E}(z_i^\top A z_i) = \text{Tr}(A)$</td>
</tr>
<tr>
<td>Hutchinson</td>
<td>i. random, i.i.d. Rademacher</td>
</tr>
<tr>
<td>$H_s(P(A)) = \frac{1}{s} \sum_{i=1}^{s} (z_i^T P(A, z_i))$</td>
<td></td>
</tr>
<tr>
<td>uniform vector</td>
<td>drawn independently and</td>
</tr>
<tr>
<td>$U_s(P(A)) = \frac{n}{s} \sum_{i=1}^{s} (z_i^T P(A, z_i))$</td>
<td>uniformly from ${e_1, \ldots, e_n}$</td>
</tr>
<tr>
<td>mixed unit vector</td>
<td>$z_i$ as before; $\mathcal{F} := FD$</td>
</tr>
<tr>
<td>$T_s(P(A)) = \frac{n}{s} \sum_{i=1}^{s} (z_i^T \mathcal{F} P(A, \mathcal{F}^\top z_i))$</td>
<td>$D$ diagonal, unitary w. Rademacher</td>
</tr>
<tr>
<td></td>
<td>entries; $F$ ``special seed'' matrix</td>
</tr>
</tbody>
</table>
Fast method (BCF09)

Input \( A, \{ z_i, i = 1, \ldots, s \}, k, \text{tol} \)

Step 1 Solve \( Ax_i = z_i, i = 1, \ldots, s \) w. low precision CG + iterative refinement.

Step 2 Apply stochastic estimator diagonal of inverse

Low precision CG + iterative refinement solver

for \( i = 1, \ldots, s \)

Given \( x_0^{(i)} \), compute \( r_0^{(i)} = z^{(i)} - Ax_0^{(i)} \)

\( k = 1 \)

while \( \| r_k^{(i)} \| > \text{tol} \)

\( a_k^{(i)} \leftarrow P_{\text{low}}(A, r_k^{(i)}, k) \) % performs \( k \) CG steps (l.p.)

\( x_{k+1}^{(i)} = x_k^{(i)} + a_k^{(i)} \) (h.p.)

\( r_{k+1}^{(i)} = z^{(i)} - Ax_{k+1}^{(i)} \) (h.p.)

\( k = k + 1 \)

endwhile

endfor
Objective

Solve $AX = Z$ where $A \in \mathbb{R}^{n \times n}$ is spd and $Z \in \mathbb{R}^{n \times s}$ with $s > 1$.

When possible: Factorize and solve: $LL^T X = Z$

Costs: Solve at a rate of $O\left(\frac{n^3}{s} + n^2\right)$ per rhs

Cubic cost amortized as $s$ increases.

Typically

Matrix $A$ large iterative methods
Linear systems with multiple right-hand sides

Objective
Solve $AX = Z$ where $A \in \mathbb{R}^{n \times n}$ is spd and $Z \in \mathbb{R}^{n \times s}$ with $s > 1$.

When possible: Factorize and solve: $LL^TX = Z$

Costs: Solve at a rate of $O\left(\frac{n^3}{s} + n^2\right)$ per rhs

Cubic cost amortized as $s$ increases.

Typically
Matrix $A$ large iterative methods

Can the iterative method amortize costs over the rhs?
Iterative methods for mrhs: Rough taxonomy

Seed methods: Exploit information from one Krylov subspace
Parlett’80, Saad’87, PapadrakakisSmerou, vdVorst, SmithPetersonMittra’89, Fisher, SimonciniG, ChanWan’97, GuennouniJbilou, Gu, LotstedtNilsson, AbdelRehimMorganWilcox, ...

Block methods: Generate and use block Krylov subspace
O’Leary, Vital, NikishinYeremin, SimonciniG, CalvettiReichel, FreundMalhotra, Jbilou, JbilouMessaoudiSadok, JbilouSadok, GuennouniJbilou, BakerDennisJessup, Gutknecht, ...

Hybrid approaches: Block seeds, deflation, multiple matrices, diagonal estimation
SimonciniG, ChanWan, SaadErhel, ErhelGyomar’ch, ChanNg, deSturler, KilmerMillerRappoport, Morgan, GolubRuizTouhami, OrginosStathopoulos..., AnitescuChenWang,Chen, BurrageWilliamsErhelPohl, GolubvonMatt, SidjeWilliamsBurrage, ...
Seed methods: What & why? (Par80, Saa87, SPM89, CW97)

- **Iterate:** Select & build seed Krylov subspace $K_m(A, r_0^{(1)})$; use to approximate remaining systems.

- In Lanczos formulation, when $s = 2$, solve $\Pi_m(z^{(2)} - Ag) = 0$ for $g \in x_0^{(2)} + K_m(A, r_0^{(1)})$, thus

$$\bar{g} = x_0^{(2)} + V_m T_m^{-1} V_m r_0^{(2)}.$$  

- Seed Krylov subspace captures well extremal eigenvectors.

- Projection deflates their contribution from remaining initial residuals

- (Saa87) If $\Pi_m r_0^{(2)} = \sum_{j=1}^{m} \eta_j A^{j-1} v_1$ then

$$\|b^{(2)} - A\bar{g}\|_{A^{-1}} = \|(I - \Pi_m) r_0^{(2)}\|_{A^{-1}} + \epsilon,$$

where $\epsilon \leq \frac{\|\eta_1 v_1\|_{A^{-1}}}{C_m(\gamma)}$, $\gamma = \frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1}$.
Basic steps for **SeedCG**

<table>
<thead>
<tr>
<th>CG step</th>
<th>Galerkin projection step</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>(iteration i, seed k)</em></td>
<td>*(non-seed k ( \neq j \in {1, \ldots, s} ))</td>
</tr>
<tr>
<td>( \delta_i^{(k,k)} = \frac{(r_i^{(k,k)})^\top r_i^{(k,k)}}{(r_{i-1}^{(k,k)})^\top r_{i-1}^{(k,k)}} )</td>
<td></td>
</tr>
<tr>
<td>( p_i^{(k,k)} = r_i^{(k,k)} + \delta_i^{(k,k)} p_{i-1}^{(k,k)} )</td>
<td></td>
</tr>
<tr>
<td>( \sigma_i^{(k,k)} = \frac{(r_i^{(k,k)})^\top r_i^{(k,k)}}{(p_i^{(k,k)})^\top A p_i^{(k,k)}} )</td>
<td>( \eta_i^{(k,j)} = \frac{(p_i^{(k,k)})^\top r_i^{(k,j)}}{(p_i^{(k,k)})^\top A p_i^{(k,k)}} )</td>
</tr>
<tr>
<td>( x_{i+1}^{(k,k)} = x_i^{(k,k)} + \sigma_i^{(k,k)} p_i^{(k,k)} )</td>
<td>( x_{i+1}^{(k,j)} = x_i^{(k,j)} + \eta_i^{(k,j)} p_i^{(k,k)} )</td>
</tr>
<tr>
<td>( r_{i+1}^{(k,k)} = r_i^{(k,k)} - \sigma_i^{(k,k)} A p_i^{(k,k)} )</td>
<td>( r_{i+1}^{(k,j)} = r_i^{(k,j)} - \eta_i^{(k,j)} A p_i^{(k,k)} )</td>
</tr>
</tbody>
</table>
Clear advantage for low-rank mrhs

- If $B \in \mathbb{R}^{n \times s}$ and $\text{rank}(B) = 1$, then traditional approach will run CG $s$ times while SeedCG only once.
- If $\text{rank}(B) = k$, then SeedCG needs only $k - 1$ restarts ((ChanWan)).

Classic "massive" seeding idea:

- project on seed Krylov subspace

\[ x_0^{(k)} = \hat{x}_0^{(k)} + V_m T_m^{-1} V_m^\top \hat{r}_0^{(k)}, \quad \forall 1 \geq k \leq s \]

- if initial guesses not satisfactory, refine by selecting another system as the seed and repeat above procedure (SPM89, CW97)
- "massive seeding" works well when rhs are "similar"
- idea can be extended to "block seed" (CW97)
Basic steps and costs for **SeedCG**

<table>
<thead>
<tr>
<th>CG step (iteration (i), seed (k))</th>
<th>Galerkin projection step (non-seed (k \neq j \in {1, \ldots, s}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \delta_{i}(k,k) = \frac{(r_{i}^{(k,k)})^\top r_{i}^{(k,k)}}{(r_{i-1}^{(k,k)})^\top r_{i-1}^{(k,k)}} )</td>
<td>( \eta_{i}(k,j) = \frac{(p_{i}^{(k,k)})^\top r_{i}^{(k,j)}}{(p_{i}^{(k,k)})^\top A p_{i}^{(k,k)}} )</td>
</tr>
<tr>
<td>( p_{i}^{(k,k)} = r_{i}^{(k,k)} + \delta_{i} p_{i-1} )</td>
<td>( x_{i+1}^{(k,j)} = x_{i}^{(k,j)} + \eta_{i}(k,j) p_{i}^{(k,k)} )</td>
</tr>
<tr>
<td>( \sigma_{i}(k,k) = \frac{(r_{i}^{(k,k)})^\top r_{i}^{(k,k)}}{(p_{i}^{(k,k)})^\top A p_{i}^{(k,k)}} )</td>
<td>( \eta_{i}(k,j) = \frac{(p_{i}^{(k,k)})^\top r_{i}^{(k,k)}}{(p_{i}^{(k,k)})^\top A p_{i}^{(k,k)}} )</td>
</tr>
<tr>
<td>( x_{i+1}^{(k,k)} = x_{i}^{(k,k)} + \sigma_{i} p_{i} )</td>
<td>( x_{i+1}^{(k,j)} = x_{i}^{(k,j)} + \eta_{i}(k,j) p_{i}^{(k,k)} )</td>
</tr>
<tr>
<td>( r_{i+1}^{(k,k)} = r_{i}^{(k,k)} - \sigma_{i} A p_{i}^{(k,k)} )</td>
<td>( r_{i+1}^{(k,j)} = r_{i}^{(k,j)} - \eta_{i}(k,j) A p_{i}^{(k,k)} )</td>
</tr>
</tbody>
</table>

| 1MV + 2DOT + 3SAXPY | 1DOT + 2SAXPY / rhs |
Generate initial guesses I

Cons to massive seeding

- entails lots of (hidden) overhead
- ... caused by inner products - due to projections

Lighter projection scheme

- solve seed system
- compute (initial) approximations to remaining systems
- correct any if needed using CG

Numerical reasons to avoid massive seeding

- loss of orthogonality, seed residual not exactly orthogonal to previously deflated eigenvectors
- initial guesses for subsequent rhs adversely affected (possible reemergence of extreme eigenvectors)
- however, interior eigenvectors might be better reduced
Generate initial guesses

Cons to massive seeding
- entails lots of (hidden) overhead
- ... caused by inner products - due to projections

Lighter projection scheme
- solve seed system
- compute (initial) approximations to remaining systems
- correct any if needed using CG
Generate initial guesses I

**Cons to massive seeding**

- entails lots of (hidden) overhead
- ... caused by inner products - due to projections

**Lighter projection scheme**

- solve seed system
- compute (initial) approximations to remaining systems
- correct any if needed using CG

**Numerical reasons to avoid massive seeding**

- loss of orthogonality, seed residual not exactly orthogonal to previously deflated eigenvectors
- initial guesses for subsequent rhs adversely affected (possible reemergence of extreme eigenvectors)
- however, interior eigenvectors might be better reduced
Let $A$ be the $20,000 \times 20,000$ matrix whose entries are zero everywhere except for the primes 2, 3, 5, 7, \ldots, 224737 along the main diagonal and the number 1 in all the positions $A(i,j)$ with $|i-j| = 1, 2, 4, 8, \ldots, 16384$. What is the $(1,1)$ entry of $\text{inv}(A)$?

Problem 7

$A$ is a $20000 \times 20000$ integer matrix

$A(i,i) = \text{prime}(i)$

$A(i,j) = 1$ if $|i-j| = 1, 2, 4, 8, \ldots, 16384$

$= 0$ else

What is $(A^{-1})_{(11)}$?

Solution:

$0.72507834626840116746868771925116096886918059447950895787816\ldots$

Method:

Solve $A^T = [1, 0, 0, 0, \ldots, 0]$; solution is $x_1$. Conjugate gradient works well since $A$ is sparse symmetric positive definite. Neither $A$ nor $A^{-1}$ need be stored.
$A = \text{Trefethen2000}$

**Figure:** Eigenvector contribution into non-seed residual while solving $Ax^{(1)} = z^{(1)}$, where $z^{(k)}$ are $\mathcal{N}(0, 1)$. Matrix $A = \text{Trefethen2000}$ with $
\lambda_1 = 1.1207, \; \lambda_2 = 2.6268 \text{ and } \lambda_6 = 13.1808, \; \lambda_7 = 16.7443, \; \lambda_8 = 19.2067.$
Figure: Eigenvector contribution into non-seed residual while solving $A x^{(1)} = z^{(1)}$, where $z^{(k)}$ are $\mathcal{N}(0, 1)$ and $A = \text{diag}([1 : 10000]/10000)$.
Block approach

Goal: reduce contribution of eigenvectors eigenvector contribution
Increase efficiency by expanding search space to capture
eigenvectors

Pros:
- each subsequent rhs projected on a block Krylov subspace
- better captures the extremal eigenvectors, especially if eigenvectors converge later than linear systems.
- independence in columns of Z is a desirable feature in block algorithms
Block approach

**Goal:** reduce contribution of eigenvectors eigenvector contribution
Increase efficiency by expanding search space to capture eigenvectors

- Expand subspace to contain capture further interior eigenvectors
  ⇒ Block CG (BCG) (O’L80)
Block approach

Goal: reduce contribution of eigenvectors eigenvector contribution
Increase efficiency by expanding search space to capture eigenvectors

- Expand subspace to contain capture further interior eigenvectors
  ⇒ Block CG (BCG) (O’L80)
- Massive block seed approach (CW97)
Block approach

**Goal:** reduce contribution of eigenvectors eigenvector contribution
Increase efficiency by expanding search space to capture eigenvectors

- Expand subspace to contain capture further interior eigenvectors
  ⇒ Block CG (BCG) (O’L80)
- Massive block seed approach (CW97)
- Block seed only once
- partition \( Z = [Z^{(1)}, \ldots, Z^{(ν)}] \), \( p = \frac{s}{ν} \) and set \( Z^{(1)} \) as the block seed.

**Pros:**
- each subsequent rhs projected on a block Krylov subspace
- better captures the extremal eigenvectors, especially if eigenvectors converge later than linear systems.
- independence in columns of \( Z \) is a desirable feature in block algorithms
Figure: One rhs as seed; $A = \text{diag}([1 : 10000]/10000)$
Figure: 4 rhs as (block) seed; $A = \text{diag}([1:10000]/10000)$
BlockSeedCG - version with only one seed

input : $A, Z, tol, \hat{X}_0, \nu$

output : $[X^{(1)}, X^{(2)}_0 \ldots, X^{(\nu)}_0]$  

$R^{(1)}_0 = Z^{(1)}$  

$P^{(1)}_0 = R^{(1)}_0 \zeta_0$

for $i = 0, \ldots, $ convergence do

\[ \alpha = (P^{(1)}_i AP^{(1)}_i)^{-1} \zeta_i^T (R^{(1)^T}_i R^{(1)}_i) \]

\[ X^{(1)}_{i+1} = X^{(1)}_i + P^{(1)}_i \alpha \]

\[ R^{(1)}_{i+1} = R^{(1)}_i - AP^{(1)}_i \alpha \]

\[ \beta = \zeta_i^{-1} (R^{(1)^T}_i R_i)^{-1} (R^{(1)^T}_{i+1} R_{i+1}) \]

\[ P^{(1)}_{i+1} = (R^{(1)}_{i+1} + P^{(1)}_i \beta) \zeta_{i+1} \]

for $j = 2, \ldots, \nu$ do

\[ \eta^{(j)}_i = ((P^{(1)}_i)^T AP^{(1)}_i)^{-1} (P^{(1)}_i)^T \hat{R}^{(j)}_i \]

\[ \hat{X}^{(1,j)}_{i+1} = \hat{X}^{(1,j)}_i + P^{(1)}_i \eta^{(1,j)}_i \]

\[ \hat{R}^{(j)}_{i+1} = \hat{R}^{(1,j)}_i - AP^{(1)}_i \eta^{(1,j)}_i \]

end for

end for
projBCG - version 1

input : $A$, $Z$, $\hat{X}_0$, $tol_1$, $tol_2$, $\nu$
output : $d(A^{-1})$
partition $Z$ as $Z = [Z^{(1)}, \ldots, Z^{(\nu)}]$
$[X^{(1)}, X_0^{(2)} \ldots, X_0^{(\nu)}] = \text{BlockSeedCG}(A, Z, \hat{X}_0, tol_1, \nu)$
for $k = 2, \ldots, \nu$ do
    $[X^{(k)}] = \text{BCG}(A, Z^{(k)}, X_0^{(k)}, tol_2)$
end for

t = \sum_{j=1}^{s} x^{(j)} \odot z^{(j)}$
$q = \sum_{j=1}^{s} z^{(j)} \odot z^{(j)}$
$D_s(A^{-1}) = t \oslash q$
Reorthogonalization

- full:
  - cost runs at $O(m^2 n)$
  - secondary storage is enabled

- partial, periodic:
  - cost can be better amortized
  - suitable if memory swaps are rare

Alternative approach

- re-project on part of the Krylov subspace
  - re-projection refines the initial guesses
  - extremal eigenvectors are now reduced again, as the seed converged at an earlier phase

- two possibilities:
  - keep in secondary storage part of $K_m(A, Z^{(1)})$
  - solve again with $Z^{(1)}$ to lower accuracy
projBCG - version 2

**input**: $A$, $Z$, $\hat{X}_0$, $\text{tol}_1$, $\text{tol}_2$, $\nu$

**output**: $d(A^{-1})$

partition $Z$ as $Z = [Z^{(1)}, \ldots, Z^{(\nu)}]$

$[X^{(1)}, X_0^{(2)} \ldots, X_0^{(\nu)}] = \text{BlockSeedCG}(A, Z, \hat{X}_0, \text{tol}_1, \nu)$

$\hat{X}_0(:, 1:p) = \text{zeros}(n, p)$;

$[X^{(1)}, X_0^{(2)} \ldots, X_0^{(\nu)}] = \text{BlockSeedCG}(A, Z, \hat{X}_0, \text{tol}_2, \nu)$

for $k = 2, \ldots, \nu$ do

$[X^{(k)}] = \text{BCG}(A, Z^{(k)}, X_0^{(k)}, \text{tol}_2)$

end for

$t = \sum_{j=1}^{s} x^{(j)} \odot z^{(j)}$

$q = \sum_{j=1}^{s} z^{(j)} \odot z^{(j)}$

$D_s(A^{-1}) = t \odot q$
$A = \text{diag}([1 : 10000]/10000), \rho = 5$
$A = \text{diag}([1 : 10000]/10000), \rho = 5$
Some practical considerations

- size of block $p$, chosen as fraction of $s$
- smaller $tol_1$ implies larger Krylov subspace
- avoid singularity (but if it happens, deflate and repartition)
- what if eigenvalues at one end are better separated?
  - corresponding eigenvectors can easily be found by a Lanczos process
  - $projBCG$ works on the orthogonal complement of these eigenvectors
Dense and well-conditioned
MV’s are the major computational cost. Storage can be a problem. Iterative methods converge relatively fast. No need for aggressive preconditioning.

Independence of rhs
Convergence at similar rates so block methods are better tolerated. Simple seed methods might be less efficient.

Low precision is adequate
Block methods well tolerated with little danger of block singularity even at convergence.
Computational environment

Hardware

- 702 Blade, 2 socket system
- 8 core Power 7, @ 3GHz, 192Gflop/s peak per chip,
- 64GB memory, L2 cache: 256KB per core, L3 cache: 4MB per core

Software

- OS: AIX; Compiler: IBM XL Fortran 90 for AIX, V13.1
- Numerical libraries: ESSL (SMP)
Some comparisons

Figure: model covariance matrix $A$ with $n = 10000$ and $\text{cond}(A) = 10^4$. $s = 40$, rhs i. random, i.i.d. Rademacher; Compared $\text{BCG}$, $\text{eigCG}$, $\text{projBCG}$
Measurements

- Model covariance matrices with quadratic decay
- spd $A$ with entries $\alpha_{ij}$ determined as

\[
\alpha_{ij} = \begin{cases} 
1 + \sqrt{i} & \text{if } i = j \\
1/|i - j|^2 & \text{if } i \neq j
\end{cases}
\]

- Convergence set to $\|r\| \leq 10^{-6}$
- Dimensions $n = 1000 : 1000 : 30,000$, $s = 300$
Numerical results: Error vs. rhs

**Figure:** model covariance matrix $A$ with $n = 10000$. rhs i. random, i.i.d. Rademacher, block size = $s/10$
Comparisons: MV’s vs. rhs

Figure: model covariance matrix $A$ with $n = 10000$. rhs i. random, i.i.d. Rademacher; block size = $s/10$
Figure: model covariance matrix $A$ with $n = 10000$. rhs i. random, i.i.d. Rademacher; block size = $s/10$
Figure: MV per rhs for different block sizes vs. matrix size
Figure: Total time for different blocksizes vs. matrix size
Performance

Figure: Gflop/s for different block sizes vs. matrix size (192 Gflop/s system peak)
Leverage for preconditioning

- in many cases, $A$ is "heavy" on the diagonal
- Jacobi preconditioner
- deploy stochastic estimation to approximate $\text{diag}(A)$:

$$\text{Var}(\text{tr}(A)) = 2(\|A\|_F^2 - \sum_{i=1}^{n} A_{ii}^2)$$

- cost: $s$ MV products
Example

- spd $A$ with entries $\alpha_{ij}$ determined as

\[
\alpha_{ij} = \begin{cases} 
1 + \sqrt{i} & \text{if } i = j \\
\frac{1}{|i-j|^u} & \text{if } i \neq j 
\end{cases}
\]

- convergence set to $\|r\| \leq 10^{-6}$

<table>
<thead>
<tr>
<th></th>
<th>CG</th>
<th>PCG</th>
<th>BCG</th>
<th>PBCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u=1$</td>
<td>2100</td>
<td>720</td>
<td>1100</td>
<td>520</td>
</tr>
<tr>
<td>$u=2$</td>
<td>1680</td>
<td>560</td>
<td>960</td>
<td>400</td>
</tr>
</tbody>
</table>

Table: MV products, $n=20000$, $s=20$
Conclusions

- effective method for special problems of interest to applications
- matrix-free
- replaces Cholesky and multiple CG
- performance and scalability appear good
- MRHS is useful tool for this problem
Conclusions

- effective method for special problems of interest to applications
- matrix-free
- replaces Cholesky and multiple CG
- performance and scalability appear good
- MRHS is useful tool for this problem

Plans

- finish article and submit paper by deadline
- parallel processing (SIAM PP’2012)
- alternative statistical estimators
- better statistical postprocessing (Mitrouli et al.?)
- methodology for general matrix functions / pros/cons
- your applications? financial applications? THE DEBT problem?
On the subject of **debt**, the **spiritual** one should be more important!
H. Avron and S. Toledo.
Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix.

C. Bekas, A. Curioni, and I. Fedulova.
Low-cost data uncertainty quantification.

C. Bekas, E. Kokiopoulou, and Y Saad.
An estimator for the diagonal of a matrix.

T.F. Chan and W.L. Wan.
Analysis of projection methods for solving linear systems with multiple right-hand sides.

M. Hutchinson.
A stochastic estimator for the trace of the influence matrix for laplacian smoothing splines.
*Communications in Statistics - Simulation and Computation, 1989.*

L. Lin et al.
Selinv---an algorithm for selected inversion of a sparse symmetric matrix.
D.P. O’Leary.
The block conjugate gradient algorithm and related methods.

B. N. Parlett.
A new look at the Lanczos algorithm for solving symmetric systems of linear equations.

C. Rasmussen and C. Williams.
Gaussian Processes for Machine Learning.

Y. Saad.
On the Lanczos method for solving symmetric systems with several right hand sides.

Efficient sampling for spatial uncertainty quantification in multibody system dynamics applications.
C. F. Smith, A. F. Peterson, and R. Mittra.
A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields.

J. Tang and Y. Saad.
A probing method for computing the diagonal of the matrix inverse.
*Numerical Linear Algebra with Applications*, 2011.
To appear.