On computing quadrature-based bounds for the A-norm of the error in conjugate gradients

Petr Tichý

joint work with

Gerard Meurant and Zdeněk Strakoš

Institute of Computer Science,
Academy of Sciences of the Czech Republic

June 7, 2012, Dolní Maxov
Programy a algoritmy numerické matematiky 16 (PANM 16)

Problem formulation

Consider a system

$$\mathbf{A}x = b$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is **symmetric, positive definite**.

- $\mathbf{A}$ is large and sparse ,
- we do not need exact solution ,
- we are able to perform $\mathbf{A}v$ effectively (v is a vector) .

Without loss of generality, $\|b\| = 1$, $x_0 = 0$.

The conjugate gradient method

input $\mathbf{A}$, b

$r_0 = b$, $p_0 = r_0$

for $k = 1, 2, \ldots$ **do**

$$\begin{aligned}
\gamma_{k-1} &= \frac{r_{k-1}^T r_{k-1}}{p_{k-1}^T \mathbf{A}\, p_{k-1}} \\
x_k &= x_{k-1} + \gamma_{k-1} p_{k-1} \\
r_k &= r_{k-1} - \gamma_{k-1} \mathbf{A}\, p_{k-1} \\
\delta_k &= \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}} \\
p_k &= r_k + \delta_k p_{k-1}
\end{aligned}$$

test quality of x_k

end for

Mathematical properties of CG

optimality property

The kth Krylov subspace,

$$\mathcal{K}_k(\mathbf{A}, b) \equiv \operatorname{span}\{b, \mathbf{A}b, \ldots, \mathbf{A}^{k-1}b\}.$$

CG $\rightarrow x_k$, r_k, p_k

- residuals $r_0, \ldots, r_{k-1}$ form an orthogonal basis of $\mathcal{K}_k(\mathbf{A}, b)$,
- vectors $p_0, \ldots, p_{k-1}$ form an $\mathbf{A}$-orthogonal basis of $\mathcal{K}_k(\mathbf{A}, b)$,
- CG finds the solution of $\mathbf{A}x = b$ in at most n steps.
- The CG approximation x_k is optimal

$$\|x - x_k\|_{\mathbf{A}} = \min_{y \in \mathcal{K}_k} \|x - y\|_{\mathbf{A}}.$$

A practically relevant question

How to measure quality of an approximation?

- **using residual information,**
  - normwise backward error,
  - relative residual norm.

  "Using of the residual vector r_k as a measure of the "goodness" of the estimate x_k is not reliable" [Hestenes & Stiefel 1952]

- **using error estimates,**
  - estimate of the $\mathbf{A}$-norm of the error,
  - estimate of the Euclidean norm of the error.

  "The function $(x - x_k, \mathbf{A}(x - x_k))$ can be used as a measure of the "goodness" of x_k as an estimate of x." [Hestenes & Stiefel 1952]

The (relative) $\mathbf{A}$-norm of the error plays an important role in stopping criteria in many problems [Deuflhard 1994], [Arioli 2004], [Jiránek, Strakoš, Vohralík 2006]

The Lanczos algorithm

Let $\mathbf{A}$ be symmetric, compute orthonormal basis of $\mathcal{K}_k(\mathbf{A}, b)$

input $\mathbf{A}$, b
$v_1 = b/\|b\|$, $\delta_1 = 0$
$\beta_0 = 0$, $v_0 = 0$
for $k = 1, 2, \ldots$ **do**
  $\alpha_k = v_k^T \mathbf{A} v_k$
  $w = \mathbf{A} v_k - \alpha_k v_k - \beta_{k-1} v_{k-1}$
  $\beta_k = \|w\|$
  $v_{k+1} = w/\beta_k$
end for

$$\mathbf{T}_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & & \\ & & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix}$$

$$\mathbf{A} v_k = \beta_k v_{k+1} + \alpha_k v_k + \beta_{k-1} v_{k-1} \,.$$

The Lanczos algorithm can be represented by

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_k \mathbf{T}_k + \beta_k v_{k+1} e_k^T, \qquad \mathbf{V}_k^* \mathbf{V}_k = \mathbf{I} \,.$$

CG versus Lanczos

Let $\mathbf{A}$ be symmetric, positive definite

The CG approximation is the given by

$$x_k = \mathbf{V}_k \, y_k \quad \text{where} \quad \mathbf{T}_k \, y_k = \|b\| e_1,$$

and

$$v_{k+1} = (-1)^k \frac{r_k}{\|r_k\|} \, .$$

CG generates LDL^T factorization of $\mathbf{T}_k = \mathbf{L}_k \mathbf{D}_k \mathbf{L}_k^T$ where

$$\mathbf{L}_k \equiv \begin{bmatrix} 1 & & & \\ \sqrt{\delta_1} & \ddots & & \\ & \ddots & \ddots & \\ & & \sqrt{\delta_{k-1}} & 1 \end{bmatrix}, \quad \mathbf{D}_k \equiv \begin{bmatrix} \gamma_0^{-1} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \gamma_{k-1}^{-1} \end{bmatrix}.$$

CG versus Lanczos

Summary

- Both algorithms generate an orthogonal basis of the Krylov subspace $\mathcal{K}_k(\mathbf{A}, b)$.
- Lanczos generates an orthonormal basis $v_1, \ldots, v_k$ using a three-term recurrence $\rightarrow \mathbf{T}_k$.
- CG generates an orthogonal basis $r_0, \ldots, r_{k-1}$ using a coupled two-term recurrence $\rightarrow$ LDL^T factorization of $\mathbf{T}_k$.
- It holds that

$$v_{k+1} = (-1)^k \frac{r_k}{\|r_k\|} \,.$$

Orthogonal vectors $\to$ orthogonal polynomials

- residuals $r_0, \ldots, r_{k-1}$ form an orthogonal basis of $\mathcal{K}_k(\mathbf{A}, b)$,
- "CG is a polynomial method",

$$v \in \mathcal{K}_k(\mathbf{A}, b) \quad \Rightarrow \quad v = \sum_{j=0}^{k-1} \zeta_j \mathbf{A}^j b = q(\mathbf{A}) b$$

where q is a polynomial of degree at most $k-1$.

- Notation: $r_k = q_k(\mathbf{A}) b$, $\mathbf{A} = \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^T$, $b = \mathbf{U} \omega$. For $i \neq j$

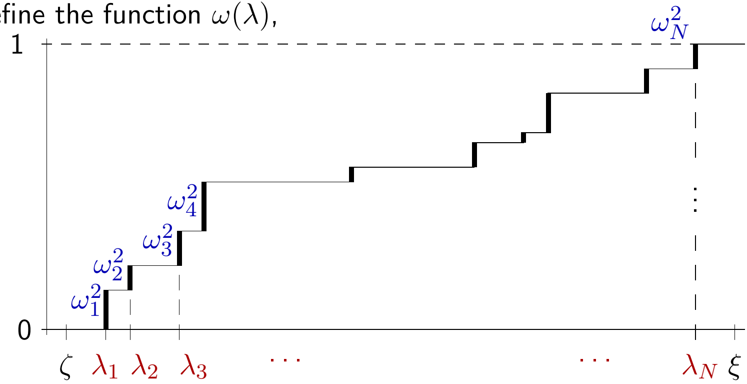
$$\begin{aligned} 0 &= r_i^T r_j = b^T q_i(\mathbf{A}) q_j(\mathbf{A}) b = \omega^T q_i(\boldsymbol{\Lambda}) q_j(\boldsymbol{\Lambda}) \omega \\ &= \sum_{\ell=1}^{N} \omega_\ell^2 q_i(\lambda_\ell) q_j(\lambda_\ell) \equiv \langle q_i, q_j \rangle_{\omega, \Lambda} . \end{aligned}$$

- CG implicitly constructs a sequence of orthogonal polynomials.

Distribution function $\omega(\lambda)$

$$\mathbf{A},\ b\ \rightarrow\ \langle\cdot,\cdot\rangle_{\omega,\Lambda}: \qquad \langle f,g\rangle_{\omega,\Lambda} = \sum_{\ell=1}^{N} \omega_\ell^2 f(\lambda_\ell) g(\lambda_\ell)\,.$$

Define the function $\omega(\lambda)$,



Then,

$$\langle f,g\rangle_{\omega,\Lambda} = \int_\zeta^\xi f(\lambda)g(\lambda)\,d\omega(\lambda)\,.$$

Quadrature formula

$$\int_{\zeta}^{\xi} f(\lambda)\, d\omega(\lambda) \;=\; \sum_{i=1}^{k} w_i f(\nu_i) \;+\; \mathcal{R}_k[f]\,.$$

Gauss Quadrature formula:

- Maximal degree of exactness $2k-1$
- Weights and nodes can be computed using orthogonal polynomials (e.g. ν_i are the roots).
- Orthogonal polynomial can be generated by a three-term recurence. Coefficients $\rightarrow$ Jacobi matrix.
- Gauss quadrature weight and nodes can be computed from the corresponding Jacobi matrix.

CG, Lanczos and Gauss quadrature

At any iteration step k, CG (implicitly) determines weights and nodes of the k-point Gauss quadrature

$$\int_{\zeta}^{\xi} f(\lambda)\, d\omega(\lambda) \;=\; \sum_{i=1}^{n} \omega_i^{(k)} f(\theta_i^{(k)}) \;+\; \mathcal{R}_k[f]\,.$$

$\mathbf{T}_k$ ... Jacobi matrix, $\theta_i^{(k)}$ ... eigenvalues of $\mathbf{T}_k$, $\omega_i^{(k)}$ ... scaled and squared first components of the normalized eigenvectors of $\mathbf{T}_k$.

$f(\lambda) \equiv \lambda^{-1}$. **Lanczos-related** quantities:

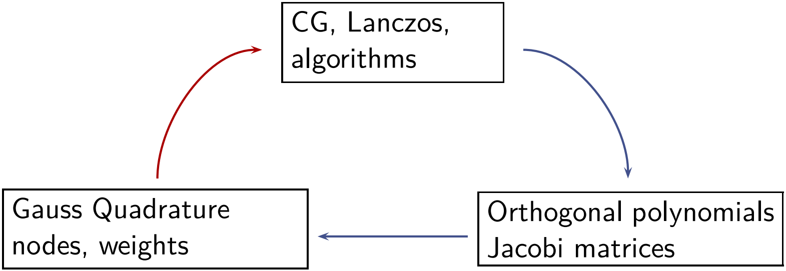
$$\left(\mathbf{T}_n^{-1}\right)_{1,1} \;=\; \left(\mathbf{T}_k^{-1}\right)_{1,1} + \mathcal{R}_k[\lambda^{-1}].$$

CG-related quantities

$$\|x\|_{\mathbf{A}}^2 \;=\; \sum_{j=0}^{k-1} \gamma_j \|r_j\|^2 + \|x - x_k\|_{\mathbf{A}}^2\,.$$

CG, Orthogonal polynomials, and Quadrature

Overview



$$\left(\mathbf{T}_n^{-1}\right)_{1,1} = \left(\mathbf{T}_k^{-1}\right)_{1,1} + \mathcal{R}_k[\lambda^{-1}]$$

$$\|x\|_{\mathbf{A}}^2 = \sum_{j=0}^{k-1} \gamma_j \|r_j\|^2 + \|x - x_k\|_{\mathbf{A}}^2 \, .$$

So why we need quadrature approach?

More general quadrature formulas

$$\int_{\zeta}^{\xi} f\, d\omega(\lambda) = \sum_{i=1}^{k} w_i f(\nu_i) + \sum_{j=1}^{m} \widetilde{w}_j f(\widetilde{\nu}_j) + \mathcal{R}_k[f],$$

the weights $[w_i]_{i=1}^k$, $[\widetilde{w}_j]_{j=1}^m$ and the nodes $[\nu_i]_{i=1}^k$ are unknowns, $[\widetilde{\nu}_j]_{j=1}^m$ are prescribed outside the open integration interval.

$m = 1$: **Gauss-Radau** quadrature. Algebraically: Given $\mu \equiv \widetilde{\nu}_1$, find $\widetilde{\alpha}_{k+1}$ so that μ is an eigenvalue of the extended matrix

$$\widetilde{\mathbf{T}}_{k+1} = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \beta_{k-1} & \\ & & \beta_{k-1} & \alpha_k & \beta_k \\ & & & \beta_k & \widetilde{\alpha}_{k+1} \end{bmatrix}.$$

Quadrature for $f(\lambda) = \lambda^{-1}$ is given by $\left(\widetilde{\mathbf{T}}_{k+1}^{-1}\right)_{1,1}$.

Quadrature formulas for $f(\lambda) = \lambda^{-1}$ take the form

$$\begin{aligned} \left(\mathbf{T}_n^{-1}\right)_{1,1} &= \left(\mathbf{T}_k^{-1}\right)_{1,1} + \mathcal{R}_k^{(G)} \,, \\ \left(\mathbf{T}_n^{-1}\right)_{1,1} &= \left(\widetilde{\mathbf{T}}_k^{-1}\right)_{1,1} + \mathcal{R}_k^{(R)} \,, \end{aligned}$$

and $\mathcal{R}_k^{(G)} > 0$ while $\mathcal{R}_k^{(R)} < 0$ if $\mu \leq \lambda_{\min}$. Equivalently

$$\begin{aligned} \|x\|_{\mathbf{A}}^2 &= \tau_k + \|x - x_k\|_{\mathbf{A}}^2 \,, \\ \|x\|_{\mathbf{A}}^2 &= \widetilde{\tau}_k + \mathcal{R}_k^{(R)} \,. \end{aligned}$$

where $\tau_k \equiv \left(\mathbf{T}_k^{-1}\right)_{1,1}$, $\widetilde{\tau}_k \equiv \left(\widetilde{\mathbf{T}}_k^{-1}\right)_{1,1}$.

[Golub & Meurant 1994, 1997, 2010, Golub & Strakoš 1994]

Idea of estimating the $\mathbf{A}$-norm of the error

Consider two quadrature rules at steps k and $k+d$, $d>0$,

$$\begin{aligned} \|x\|_{\mathbf{A}}^2 &= \tau_k + \|x-x_k\|_A^2 \,, \\ \|x\|_{\mathbf{A}}^2 &= \widehat{\tau}_{k+d} + \widehat{\mathcal{R}}_{k+d} \,. \end{aligned} \tag{1}$$

Then

$$\|x-x_k\|_{\mathbf{A}}^2 = \widehat{\tau}_{k+d} - \tau_k + \hat{\mathcal{R}}_{k+d} \,.$$

Gauss quadrature: $\hat{\mathcal{R}}_{k+d} = \mathcal{R}_{k+d}^{(G)} > 0 \to$ lower bound,
Radau quadrature: $\hat{\mathcal{R}}_{k+d} = \mathcal{R}_{k+d}^{(R)} < 0 \to$ upper bound.

How to compute efficiently

$$\widehat{\tau}_{k+d} - \tau_k \,?$$

Estimate based on Gauss quadrature rule

Evaluation

$$\|x - x_k\|_{\mathbf{A}}^2 \;=\; \tau_{k+d} - \tau_k \;+\; \|x - x_{k+d}\|_{\mathbf{A}}^2$$

We use a simple formula

$$\tau_{k+d} - \tau_k = \sum_{j=k}^{k+d-1} (\tau_{j+1} - \tau_j) \equiv \sum_{j=k}^{k+d-1} \Delta_j \,.$$

The quantity

$$\Delta_j = \left(\mathbf{T}_{j+1}^{-1}\right)_{1,1} - \left(\mathbf{T}_j^{-1}\right)_{1,1}$$

can be computed by an algorithm by Golub and Meurant, or simply using the formula

$$\Delta_j \;=\; \gamma_j \|r_j\|^2 \,.$$

Estimate based on Gauss-Radau quadrature rule

Given a node $\mu \leq \lambda_{\min}$,

$$\|x - x_k\|_{\mathbf{A}}^2 \;=\; \widetilde{\tau}_{k+d} - \tau_k \;+\; \mathcal{R}_{k+d}^{(R)}, \qquad \mathcal{R}_{k+d}^{(R)} < 0\,.$$

Reduction to the problem of computing

$$\Delta_j^{(\mu)} \;\equiv\; \widetilde{\tau}_{j+1} - \tau_j \;=\; \left(\widetilde{\mathbf{T}}_{j+1}^{-1}\right)_{1,1} - \left(\mathbf{T}_j^{-1}\right)_{1,1}\,.$$

First, we need to determine $\widetilde{\alpha}_{j+1}$ so that μ is an eigenvalue of

$$\widetilde{\mathbf{T}}_{j+1} = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \beta_{j-1} & \\ & & \beta_{j-1} & \alpha_j & \beta_j \\ & & & \beta_j & \widetilde{\alpha}_{j+1} \end{bmatrix}.$$

Second, compute $\Delta_j^{(\mu)}$ using the Golub-Meurant algorithm.

Golub and Meurant approach

[Golub & Meurant 1994, 1997]

- CG iteration $\rightarrow \gamma_{k-1}, \delta_k$.
- Compute Lanczos coefficients α_k, β_k.
- Compute rank one modification of $\mathbf{T}_{k+1} \rightarrow \tilde{\alpha}_{k+1}^{(\mu)}$.
- Compute the differences

$$\begin{aligned} \Delta_{k-1} &\equiv \left(\mathbf{T}_k^{-1}\right)_{1,1} - \left(\mathbf{T}_{k-1}^{-1}\right)_{1,1} \\ \Delta_k^{(\mu)} &\equiv \left(\widetilde{\mathbf{T}}_{k+1}^{-1}\right)_{1,1} - \left(\mathbf{T}_k^{-1}\right)_{1,1} \end{aligned}$$

- For $k > d$, use formulas

$$\|x - x_{k-d}\|_{\mathbf{A}}^2 = \sum_{j=k-d}^{k-1} \Delta_j + \|x - x_k\|_{\mathbf{A}}^2$$

$$\|x - x_{k-d}\|_{\mathbf{A}}^2 = \sum_{j=k-d}^{k-1} \Delta_j + \Delta_k^{(\mu)} + \mathcal{R}_k^{(R)}$$

for estimating.

CGQL (Conjugate Gradients and Quadrature via Lanczos)

input A, b, x_0, μ
$r_0 = b - Ax_0$, $p_0 = r_0$
$\delta_0 = 0$, $\gamma_{-1} = 1$, $c_1 = 1$, $\beta_0 = 0$, $d_0 = 1$, $\tilde{\alpha}_1^{(\mu)} = \mu$,
for $k = 1, \ldots,$ until convergence **do**
CG-iteration (k)

$$\begin{aligned}
\alpha_k &= \frac{1}{\gamma_{k-1}} + \frac{\delta_{k-1}}{\gamma_{k-2}}, \ \beta_k^2 = \frac{\delta_k}{\gamma_{k-1}^2} \\
d_k &= \alpha_k - \frac{\beta_{k-1}^2}{d_{k-1}}, \ \Delta_{k-1} = \|r_0\|^2 \frac{c_k^2}{d_k}, \\
\tilde{\alpha}_{k+1}^{(\mu)} &= \mu + \frac{\beta_k^2}{\alpha_k - \tilde{\alpha}_k^{(\mu)}}, \\
\Delta_k^{(\mu)} &= \|r_0\|^2 \frac{\beta_k^2 c_k^2}{d_k \left(\tilde{\alpha}_{k+1}^{(\mu)} d_k - \beta_k^2\right)}, \quad c_{k+1}^2 = \frac{\beta_k^2 c_k^2}{d_k^2}
\end{aligned}$$

Estimates(k,d)
end for

Meurant - Tichý approach

[Meurant & T. 2012]

- CG iteration $\rightarrow \gamma_{k-1}$, δ_k.
- Avoid the explicit use of tridiagonal matrices.
- CG provides LDL^T factorization of $\mathbf{T}_{k+1}$.
- We have shown how to update LDL^T factorization of $\widetilde{\mathbf{T}}_{k+1}$.
- Quite complicated algebraic manipulations.
- Δ_{k-1} and $\Delta_k^{(\mu)}$ can be computed using very simple formulas.

CGQ (Conjugate Gradients and Quadrature)

input A, b, x_0, μ,
$r_0 = b - Ax_0$, $p_0 = r_0$
$\Delta_0^{(\mu)} = \frac{\|r_0\|^2}{\mu}$,
for $k = 1, \ldots,$ until convergence **do**
 CG-iteration(k)

$$\begin{aligned} \Delta_{k-1} &= \gamma_{k-1}\|r_{k-1}\|^2, \\ \Delta_k^{(\mu)} &= \frac{\|r_k\|^2\left(\Delta_{k-1}^{(\mu)} - \Delta_{k-1}\right)}{\mu\left(\Delta_{k-1}^{(\mu)} - \Delta_{k-1}\right) + \|r_k\|^2} \end{aligned}$$

 Estimates(k,d)
end for

Preconditioning

The CG-iterates are thought of being applied to

$$\hat{\mathbf{A}}\hat{x} = \hat{b}\,.$$

We consider symmetric preconditioning

$$\hat{\mathbf{A}} = \mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}, \qquad \hat{b} = \mathbf{L}^{-1}b\,.$$

$\mathbf{P} \equiv \mathbf{L}\mathbf{L}^T$, change of variables

$$x_k \equiv \mathbf{L}^{-T}\hat{x}_k\,, \quad r_k \equiv \mathbf{L}\,\hat{r}_k\,, \quad z_k \equiv \mathbf{L}^{-T}\hat{r}_k\,, \quad p_k \equiv \mathbf{L}^{-T}\hat{p}_k\,.$$

It holds that

$$\begin{aligned} \|\hat{x} - \hat{x}_k\|^2_{\hat{\mathbf{A}}} &= \|x - x_k\|^2_{\mathbf{A}} \\ \|\hat{r}_k\|^2 &= z_k^T r_k\,. \end{aligned}$$

One can compute the quadratures-based estimates of the $\mathbf{A}$-norm of the error using the PCG coefficients $\hat{\gamma}_{k-1}$ and inner products $z_k^T r_k$ (instead of using $\|\hat{r}_k\|^2$).

Preconditioning - PCGQ

input $\mathbf{A}$, b, x_0, $\mathbf{P}$, μ

$r_0 = b - \mathbf{A}x_0$, $z_0 = \mathbf{P}^{-1}r_0$, $p_0 = z_0$, $\Delta_0^{(\mu)} = \frac{z_0^T r_0}{\mu}$

for $k = 1, \ldots, n$ until convergence **do**

$\hat{\gamma}_{k-1} = \frac{z_{k-1}^T r_{k-1}}{p_{k-1}^T \mathbf{A} p_{k-1}}$

$x_k = x_{k-1} + \hat{\gamma}_{k-1} p_{k-1}$

$r_k = r_{k-1} - \hat{\gamma}_{k-1} \mathbf{A} p_{k-1}$

$z_k = \mathbf{P}^{-1} r_k$

$\hat{\delta}_k = \frac{z_k^T r_k}{z_{k-1}^T r_{k-1}}$

$p_k = z_k + \hat{\delta}_k p_{k-1}$

$$\Delta_{k-1} = \hat{\gamma}_{k-1} z_{k-1}^T r_{k-1}$$

$$\Delta_k^{(\mu)} = \frac{z_k^T r_k \left(\Delta_{k-1}^{(\mu)} - \Delta_{k-1}\right)}{\mu \left(\Delta_{k-1}^{(\mu)} - \Delta_{k-1}\right) + z_k^T r_k}$$

Estimates(k,d)

Practically relevant questions

The estimation is based on formulas

$$\|x - x_k\|_{\mathbf{A}}^2 = \sum_{j=k}^{k+d-1} \Delta_j + \|x - x_{k+d}\|_{\mathbf{A}}^2$$

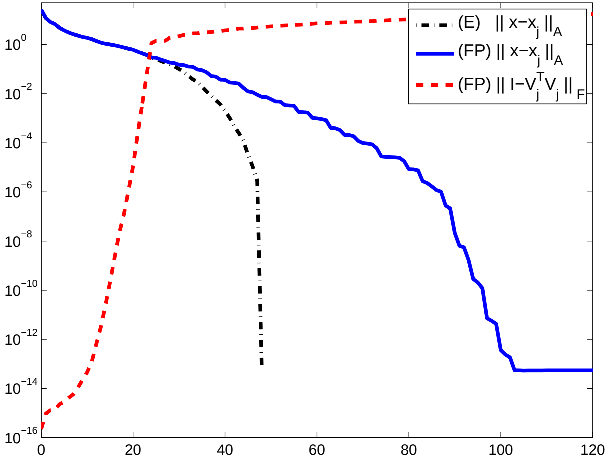
$$\|x - x_k\|_{\mathbf{A}}^2 = \sum_{j=k}^{k+d-1} \Delta_j + \Delta_{k+d}^{(\mu)} + \mathcal{R}_k^{(R)}$$

We are able to compute Δ_j and $\Delta_j^{(\mu)}$ almost for free.

Practically relevant questions:

- What happens in finite precision arithmetic ?
- How to choose d ?
- How to choose μ ?

Orthogonality is lost, convergence is delayed!



Identities need not hold in finite precision arithmetic!

Rounding error analysis

- Lower bound formula [Strakoš & T. 2002, 2005]: The equality

$$\|x - x_k\|_{\mathbf{A}}^2 \;=\; \sum_{j=k}^{k+d-1} \Delta_j \;+\; \|x - x_{k+d}\|_{\mathbf{A}}^2$$

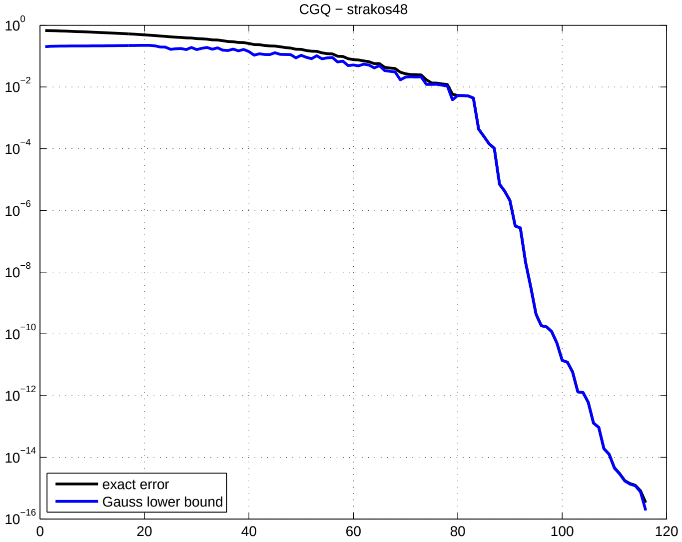
  holds (up to a small inaccuracy) also in finite precision arithmetic for computed vectors and coefficients.

- Upper bound formula: There is no rounding error analysis of the formula

$$\|x - x_k\|_{\mathbf{A}}^2 = \sum_{j=k}^{k+d-1} \Delta_j + \Delta_{k+d}^{(\mu)} \;+\; \mathcal{R}_{k+d}^{(R)}\,.$$

The choice of d - Experiment 1

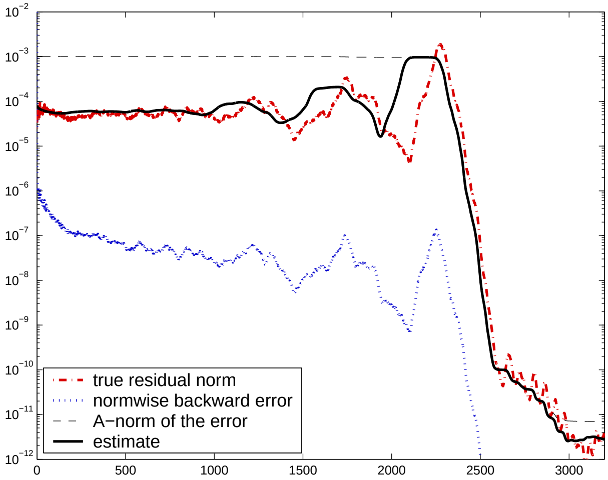
Strakos matrix, $n = 48$, $\lambda_1 = 0.1$, $\lambda_n = 1000$, $\rho = 0.9$, $d = 4$



The choice of d - Experiment 2

R. Kouhia: Cylindrical shell (Matrix Market), matrix s3dkt3m2

PCG, $\kappa(\mathbf{A}) = 3.62e+11\,,\ n = 90499\,,\ d = 200\,,$ cholinc$(\mathbf{A}, 0)\,.$



The choice of d

$$\|x - x_k\|_{\mathbf{A}}^2 = \sum_{j=k}^{k+d-1} \Delta_j \; + \; \|x - x_{k+d}\|_{\mathbf{A}}^2$$

We get a tight lower bound if

$$\|x - x_k\|_{\mathbf{A}}^2 \; \gg \; \|x - x_{k+d}\|_{\mathbf{A}}^2 \, .$$

How to detect a reasonable decrease of the $\mathbf{A}$-norm od the error?

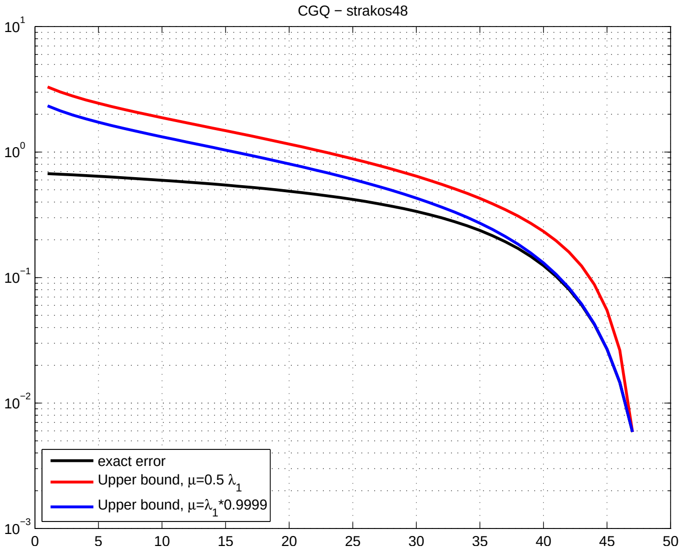
Theoretically, one could use the upper bound,

$$\frac{\|x - x_{k+d}\|_{\mathbf{A}}^2}{\|x - x_k\|_{\mathbf{A}}^2} \; \leq \; \frac{\Delta_{k+d}^{(\mu)}}{\sum_{j=k}^{k+d-1} \Delta_j} \; < \; \mathrm{tol} \, .$$

But, **can we trust the upper bound**?

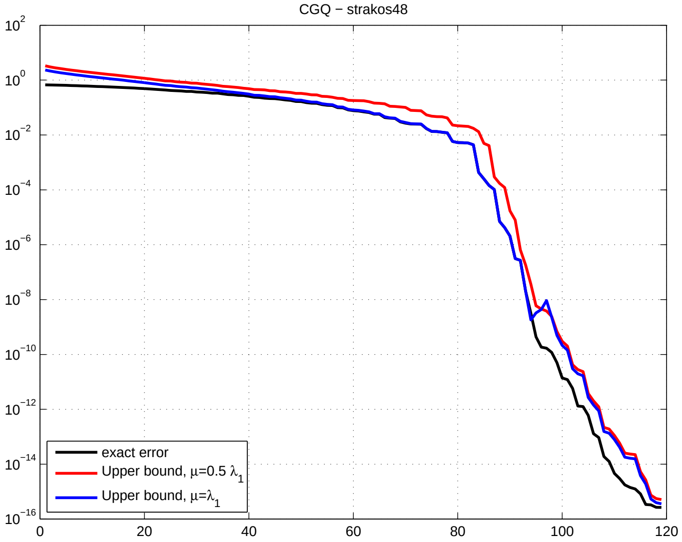
The choice of μ, upper bound, exact arithmetic

Strakos matrix, $n = 48$, $\lambda_1 = 0.1$, $\lambda_n = 1000$, $\rho = 0.9$, $d = 1$



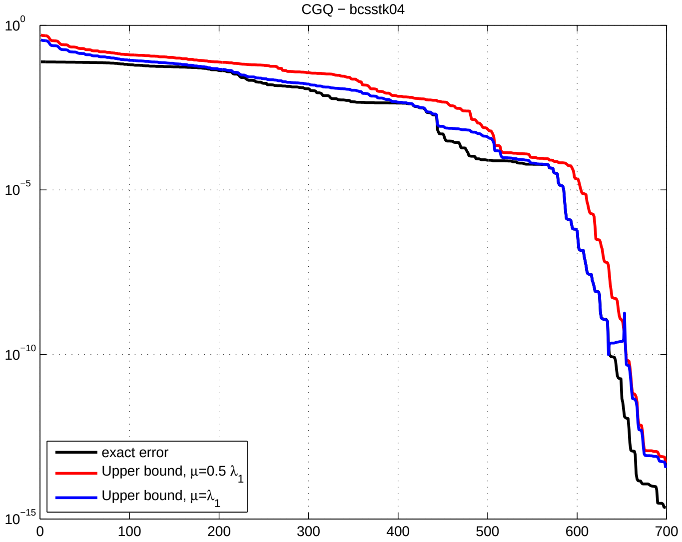
The choice of μ, upper bound, finite precision arithmetic

Strakos matrix, $n = 48$, $\lambda_1 = 0.1$, $\lambda_n = 1000$, $\rho = 0.9$, $d = 1$



The choice of μ, upper bound, finite precision arithmetic

bcsstk04 (Matrix Market), $n = 132$, $d = 1$



Numerical troubles with the upper bound

Given μ, we look for $\widetilde{\alpha}_{k+1}$ (explicitly or implicitly) so that μ is an eigenvalue of the extended matrix

$$\widetilde{\mathbf{T}}_{k+1} = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \beta_{k-1} & \\ & & \beta_{k-1} & \alpha_k & \beta_k \\ & & & \beta_k & \widetilde{\alpha}_{k+1} \end{bmatrix}.$$

To find such a $\widetilde{\alpha}_{k+1}$, we need to solve the system

$$(\mathbf{T}_k - \mu\mathbf{I})y = e_k\,.$$

If μ is close to the smallest eigenvalue of $\mathbf{T}_k$, we can get into numerical troubles!

Conclusions and questions

- The upper bound as well as the lower bound on the $\mathbf{A}$-norm of the error can be **computed in a simple way**.
- Unfortunately, the computation of the upper bound is **not always numerically stable**.
  - μ is far from λ_1 $\rightarrow$ overestimation,
  - μ is close to λ_1 $\rightarrow$ numerical troubles.
- The estimation of the $\mathbf{A}$-norm of the error **should be based** on the numerical stable **lower bound**.
- **How to detect** a reasonable decrease of the $\mathbf{A}$-norm of the error? (How to choose d adaptively?).
- Is there any way how to **involve** the upper bound?

Related papers

- G. Meurant and P. Tichý, [On computing quadrature-based bounds for the $\mathbf{A}$-norm of the error in conjugate gradients, Numer. Algorithms, (2012)]
- G. H. Golub and G. Meurant, [Matrices, moments and quadrature with applications, Princeton University Press, USA, 2010.]
- Z. Strakoš and P. Tichý, [On error estimation in the conjugate gradient method and why it works in finite precision computations, Electron. Trans. Numer. Anal., 13 (2002), pp. 56–80.]
- G. H. Golub and G. Meurant, [Matrices, moments and quadrature. II. BIT, 37 (1997), pp. 687–705.]
- G. H. Golub and Z. Strakoš, [Estimates in quadratic formulas, Numer. Algorithms, 8 (1994), pp. 241–268.]

2012 - 1952 = 60

Thank you for your attention!