A new algorithm for computing quadrature-based bounds in conjugate gradients

Petr Tichý

Czech Academy of Sciences
Charles University in Prague

joint work with

Gérard Meurant and Zdeněk Strakoš

June 08–13, 2014
Householder Symposium XIX,
Spa, Belgium

Problem formulation

Consider a system

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

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

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**

$$\gamma_{k-1} = \frac{r_{k-1}^T r_{k-1}}{p_{k-1}^T \mathbf{A}\, p_{k-1}}$$

$$x_k = x_{k-1} + \gamma_{k-1} p_{k-1}$$

$$r_k = r_{k-1} - \gamma_{k-1} \mathbf{A}\, p_{k-1}$$

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

$$p_k = r_k + \delta_k p_{k-1}$$

test quality of x_k

end for

$$\mathbf{D}_k = \begin{bmatrix} \gamma_0^{-1} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \gamma_{k-1}^{-1} \end{bmatrix}$$

$$\mathbf{L}_k = \begin{bmatrix} 1 & & & \\ \sqrt{\delta_1} & \ddots & & \\ & \ddots & \ddots & \\ & & \sqrt{\delta_{k-1}} & 1 \end{bmatrix}$$

How to measure quality of an approximation in CG?

A practically relevant question

- **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,**
  - **the $\mathbf{A}$-norm of the error**,
  - 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 $\mathbf{A}$-norm of the error plays an important role in stopping criteria [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**

$$\begin{aligned} \alpha_k &= v_k^T \mathbf{A} v_k \\ w &= \mathbf{A} v_k - \alpha_k v_k - \beta_{k-1} v_{k-1} \\ \beta_k &= \|w\| \\ v_{k+1} &= w/\beta_k \end{aligned}$$

end for

$$\mathbf{T}_k$$

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

CG versus Lanczos

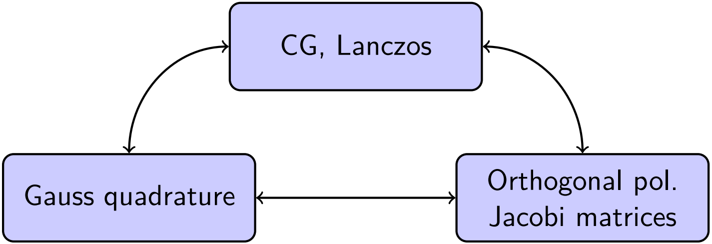
Let A be symmetric, positive definite

Lanczos		CG
$\mathbf{T}_k$	$\longleftrightarrow$	$\mathbf{D}_k,\ \mathbf{L}_k$

- Both algorithms generate an **orthogonal basis** of $\mathcal{K}_k(\mathbf{A}, b)$.
- Lanczos using a **three-term** recurrence $\rightarrow$ $\mathbf{T}_k$.
- CG using a **coupled two-term** recurrence $\rightarrow$ $\mathbf{D}_k,\ \mathbf{L}_k$.

$$\mathbf{T}_k \;=\; \mathbf{L}_k \, \mathbf{D}_k \, \mathbf{L}_k^T \, .$$

CG, Lanczos and Gauss quadrature



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

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

Gauss quadrature for $f(\lambda) \equiv \lambda^{-1}$

- **Gauss quadrature**

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

- **Lanczos**

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

- **CG**

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

Important : $\mathcal{R}_k[\lambda^{-1}] > 0\,.$

Gauss-Radau quadrature for $f(\lambda) = \lambda^{-1}$

μ is prescribed

$$\int_\zeta^\xi f(\lambda)\, d\omega(\lambda) \;=\; \underbrace{\sum_{i=1}^{k} \widetilde{\omega}_i f\left(\tilde{\theta}_i\right) + \widetilde{\omega}_{k+1} f(\mu)}_{\left(\widetilde{\mathbf{T}}_{k+1}^{-1}\right)_{1,1} \equiv \widetilde{\tau}_{k+1}} \;+ \mathcal{R}_k[f]\,,$$

where

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

and μ is an eigenvalue of $\widetilde{\mathbf{T}}_{k+1}$.

Important: if $0 < \mu \le \lambda_{\min}$, then $\mathcal{R}_k[\lambda^{-1}] < 0\,.$

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

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

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

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

Then

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

Gauss quadrature: $\hat{\mathcal{R}}_{k+d} > 0 \rightarrow$ **lower bound**,
Radau quadrature: $\hat{\mathcal{R}}_{k+d} < 0 \rightarrow$ **upper bound**.

How to compute efficiently

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

How to compute efficiently $\widehat{\tau}_{k+d} - \tau_k$?

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

For **numerical reasons**, it is not convenient to compute τ_k and $\widehat{\tau}_{k+d}$ explicitly. Instead,

$$\begin{aligned}\widehat{\tau}_{k+d} - \tau_k \;&=\; \sum_{j=k}^{k+d-2} \left(\tau_{j+1} - \tau_j\right) + \left(\widehat{\tau}_{j+d} - \tau_{j+d-1}\right)\\ &\equiv\; \sum_{j=k}^{k+d-2} \Delta_j + \widehat{\Delta}_{k+d-1}\,,\end{aligned}$$

and update the Δ_j's **without subtractions**. Recall $\tau_j = \left(\mathbf{T}_j^{-1}\right)_{1,1}$.

Golub and Meurant approach

[Golub & Meurant 1994, 1997]: Use **tridiagonal matrices**

$$\boxed{\text{CG}} \;\rightarrow\; \boxed{\mathbf{T}_k} \;\rightarrow\; \boxed{\mathbf{T}_k - \mu\mathbf{I}} \;\rightarrow\; \boxed{\widetilde{\mathbf{T}}_k}$$

and compute Δ's using **updating** strategies,
no need to store tridiagonal matrices.

Use the formulas

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

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

CGQL (Conjugate Gradients and Quadrature via Lanczos)

input $\mathbf{A}$, b, x_0, μ

$r_0 = b - \mathbf{A}x_0$, $p_0 = r_0$

$\delta_0 = 0$, $\gamma_{-1} = 1$, $c_1 = 1$, $\beta_0 = 0$, $d_0 = 1$, $\tilde{\alpha}_1^{(\mu)} = \mu$,

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

Our approach

[Meurant & T. 2013]: Update LDL^T **decompositions** of $\mathbf{T}_k$ and $\widetilde{\mathbf{T}}_k$

$$\boxed{\text{CG}} \rightarrow \boxed{\mathbf{L}_k\mathbf{D}_k\mathbf{L}_k^T} \rightarrow \boxed{\widetilde{\mathbf{L}}_k\widetilde{\mathbf{D}}_k\widetilde{\mathbf{L}}_k^T}$$

- We use **tridiagonal matrices** only **implicitly**.
- We get **very simple formulas** for updating Δ_{k-1} and $\Delta_k^{(\mu)}$.
- In [Meurant & T. 2013], this idea is used also for **other types of quadratures** (Gauss-Lobatto, Anti-Gauss).

CGQ (Conjugate Gradients and Quadrature)

[Meurant & T. 2013]

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

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

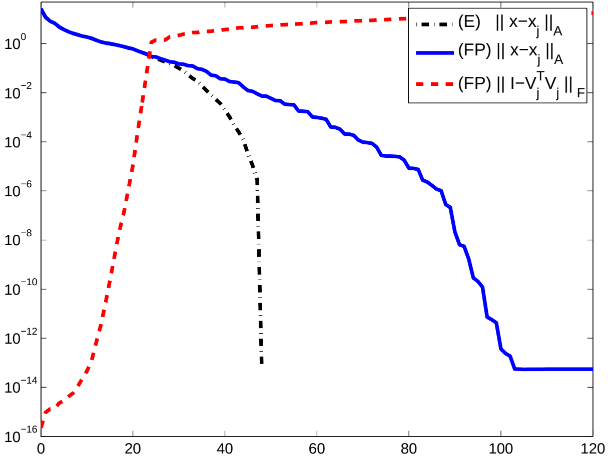
Estimates(k,d)
end for

Conclusions (theoretical part)

- **Simple formulas** for computing **bounds** on $\|x - x_k\|_{\mathbf{A}}$.
- Almost **for free**.
- Work well also with **preconditioning**.
- Behaviour in **finite precision arithmetic**?

CG in finite precision arithmetic

Orthogonality is lost, **convergence is delayed**!



Identities need not hold in finite precision arithmetic!

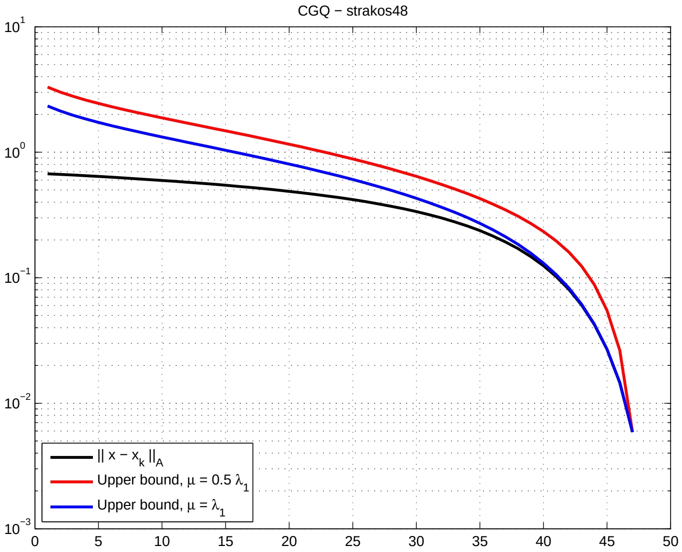
Bounds in finite precision arithmetic

- **Observation**: CGQL and CGQ give the **same results** (up to a small inaccuracy).
- Do the bounds **correspond to** $\|x - x_k\|_{\mathbf{A}}$?
- **Gauss quadrature** lower bound $\rightarrow$ **yes** [Strakoš & T. 2002].
- What about the **Gauss-Radau** upper bound?

$$\begin{aligned} \|x - x_k\|_{\mathbf{A}}^2 &= \Delta_k^{(\mu)} + \mathcal{R}_{k+1}^{(R)}, \\ \|x - x_k\|_{\mathbf{A}} &\leq \sqrt{\Delta_k^{(\mu)}}. \end{aligned}$$

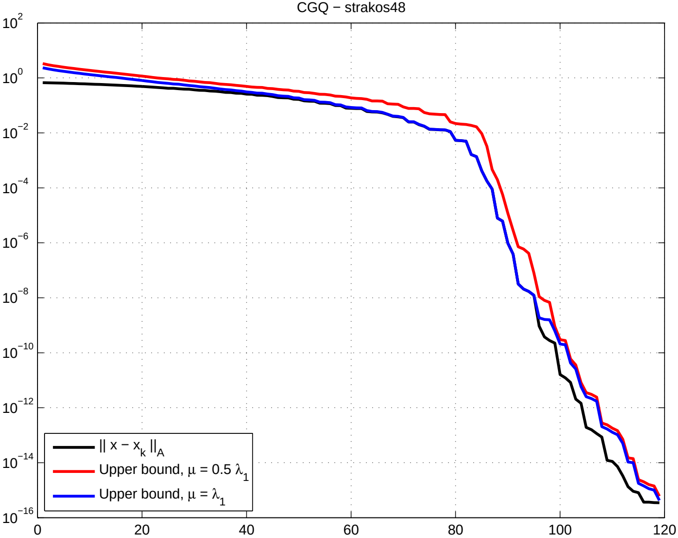
Gauss-Radau upper bound, exact arithmetic

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



Gauss-Radau upper bound, finite precision arithmetic

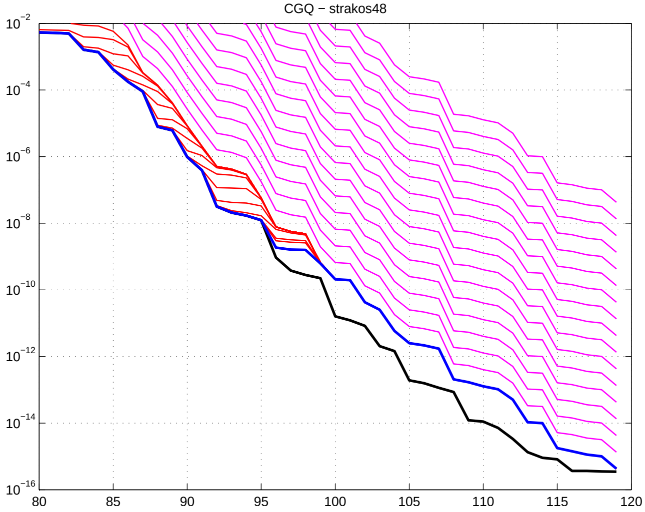
Strakoš matrix, $n = 48$, $\lambda_1 = 0.1$, $\lambda_n = 1000$, $\rho = 0.9$, $d = 1$



Gauss-Radau upper bound, finite precision arithmetic

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

$\boxed{\mu \ = \ \alpha\,\lambda_1}$, $\alpha \in (0, 1]$, $\alpha \approx 1$ (red), $\alpha \approx 0$ (magenta)

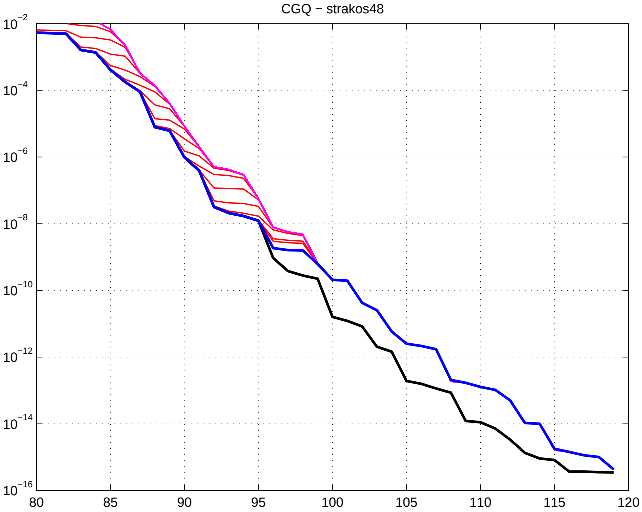


$\sqrt{\Delta_k^{(\mu)}}$

Gauss-Radau upper bound, finite precision arithmetic

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

$\boxed{\mu \ = \ \alpha\,\lambda_1}$, $\alpha \in (0, 1]$, $\alpha \approx 1$ (red), $\alpha \approx 0$ (magenta)



$\sqrt{\alpha\,\Delta_k^{(\mu)}}$

Conclusions (numerical observation)

Gauss-Radau upper bound

- It seems that $\sqrt{\varepsilon}$ **is a limiting level** for the accuracy of the **Gauss-Radau** upper bound.
- We **cannot avoid subtractions** in computing this bound. If $\mu \approx \lambda_1$, then $\mathbf{T}_k - \mu \mathbf{I}$ **may be ill conditioned**.
- Simple formulas $\rightarrow$ **investigation** of numerical behaviour.
- **Understanding** can help
  - in suggesting **another approach**,
  - in **improving Gauss quadrature** lower bound (adaptive choice of d).

Related papers

- G. Meurant and P. Tichý, [On computing quadrature-based bounds for the $\mathbf{A}$-norm of the error in CG, Numer. Algorithms, 62 (2013), pp. 163–191.]
- 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 CG and why it works in finite precision computations, ETNA, 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.]

Thank you for your attention!