Randomised least-squares: Blendenpik

[Avron-Maymounkov-Toledo 2010]

$$\min_x \|Ax - b\|_2, \qquad A \in \mathbb{R}^{m\times n}, \;\; m \gg n$$

- Traditional method: normal eqn $x = (A^TA)^{-1}A^Tb$ or $A = QR, x = R^{-1}(Q^Tb)$, both $O(mn^2)$ cost
- Randomised: generate random $G \in \mathbb{R}^{4n\times m}$, and $GA = \hat{Q}\hat{R}$

  (QR factorisation), then solve $\min_y \|(A\hat{R}^{-1})y - b\|_2$'s normal eqn via Krylov
  - $O(mn\log m + n^3)$ cost using fast FFT-type transforms for G
  - Successful because $A\hat{R}^{-1}$ is well-conditioned

Explaining Blendenpik via Marchenko-Pastur

Claim: $A\hat{R}^{-1}$ is well-conditioned with $\boxed{G}\,\boxed{A} = \boxed{\hat{Q}}\,\boxed{\hat{R}}$ (QR)

Show this for $G \in \mathbb{R}^{4n\times m}$ Gaussian:

Proof: Let $A = QR$. Then $GA = (GQ)R =: \tilde{G}R$

- $\boxed{\tilde{G}}$ is $4n \times n$ rectangular Gaussian, hence well-cond
- So by M-P, $\kappa_2(\tilde{R}^{-1}) = O(1)$ where $\tilde{G} = \tilde{Q}\tilde{R}$ is QR
- Thus $\tilde{G}R = (\tilde{Q}\tilde{R})R = \tilde{Q}(\tilde{R}R) = \tilde{Q}\hat{R}$, so $\hat{R}^{-1} = R^{-1}\tilde{R}^{-1}$
- Hence $A\hat{R}^{-1} = Q\tilde{R}^{-1}$, $\kappa_2(A\hat{R}^{-1}) = \kappa_2(\tilde{R}^{-1}) = O(1)$

Blendenpik: solving $\min_x \|Ax - b\|_2$ using $\hat{R}$

We have $\kappa_2(A\hat{R}^{-1}) =: \kappa_2(B) = O(1)$;
defining $\hat{R}x = y$, $\min_x \|Ax - b\|_2 = \min_y \|(A\hat{R}^{-1})y - b\|_2 = \min_y \|By - b\|_2$

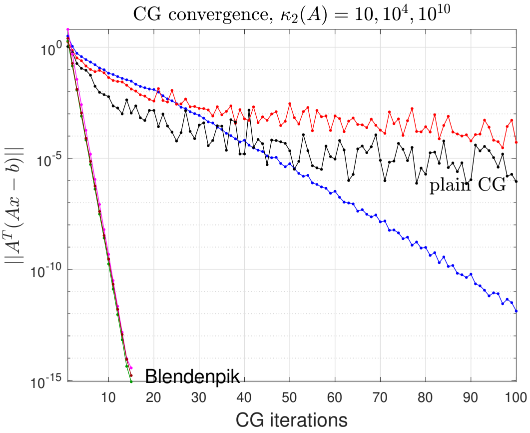
- B well-conditioned⇒in normal equation

$$B^T By = B^T b \tag{1}$$

  B well-conditioned $\kappa_2(B) = O(1)$;

- solve (1) via CG (or a tailor-made method LSQR; nonexaminable)
  - exponential convergence, $O(1)$ iterations! (or $O(\log \frac{1}{\epsilon})$ iterations for ϵ accuracy)
  - each iteration requires $w \leftarrow Bw$, consisting of $w \leftarrow \hat{R}^{-1}w$ ($n \times n$ triangular solve) and $w \leftarrow Aw$ ($m \times n$mat-vec multiplication); $O(mn)$ cost overall

Blendenpik experiments



CG for $A^TAx = A^Tb$ vs. Blendenpik $(AR^{-1})^T(AR^{-1})x = (AR^{-1})^Tb$, $m = 10000, n = 100$

In practice, Blendenpik gets $\approx \times 5$ speedup over classical (Householder-QR based) method when $m \gg n$

Randomised algorithm for $Ax = b$?

We've seen randomisation can be very successful for SVD and least-squares problems.

What about $Ax = b$? This is among the biggest open problems in (Randomised) NLA:

- Randomisation to find good preconditioner?
- Randomisation to 'deflate out' large/small singval components?
- ...

Major breakthrough needed—any ideas?

Important (N)LA topics not treated

- tensors [Kolda-Bader 2009]
- FFT (values$\leftrightarrow$coefficients map for polynomials) [e.g. Golub and Van Loan 2012]
- sparse direct solvers [Duff, Erisman, Reid 2017]
- multigrid [e.g. Elman-Silvester-Wathen 2014]
- functions of matrices [Higham 2008]
- generalised, polynomial eigenvalue problems [Guttel-Tisseur 2017]
- perturbation theory (Davis-Kahan etc) [Stewart-Sun 1990]
- compressed sensing [Foucart-Rauhut 2013]
- model order reduction [Benner-Gugercin-Willcox 2015]
- communication-avoiding algorithms [e.g. Ballard-Demmel-Holtz-Schwartz 2011]

C6.1 Numerical Linear Algebra, summary

1st half

- SVD and its properties (Courant-Fisher etc), applications (low-rank)
- Direct methods (LU) for linear systems and least-squares problems (QR)
- Stability of algorithms

2nd half

- Direct method (QR algorithm) for eigenvalue problems, SVD
- Krylov subspace methods for linear systems (GMRES, CG) and eigenvalue problems (Arnoldi, Lanczos)
- Randomised algorithms for SVD and least-squares

Where does this course lead to?

Courses with significant intersection

- C6.3 Approximation of Functions (Prof. Nick Trefethen, MT): Chebyshev polynomials/approximation theory
- C7.7 Random Matrix Theory (Prof. Jon Keating): for theoretical underpinnings of Randomised NLA
- C6.4 Finite Element Method for PDEs (Prof. Patrick Farrell): NLA arising in solutions of PDEs
- C6.2 Continuous Optimisation (Prof. Cora Cartis): NLA in optimisation problems

and many more: differential equations, data science, optimisation, machine learning,...
NLA is everywhere in computational maths

Thank you for your interest in NLA!