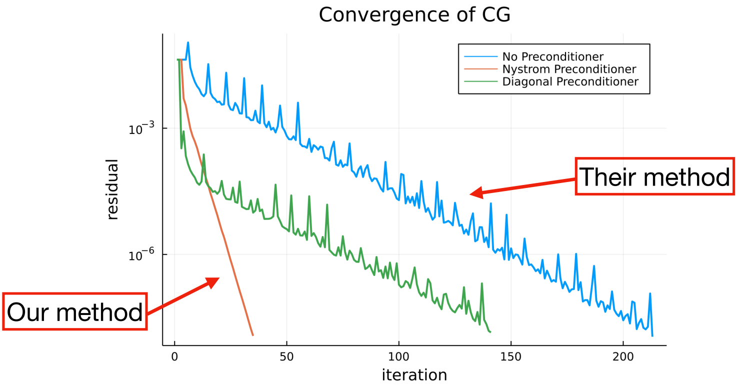
Speeding up x = A\b with RandomizedPreconditioners.jl

Theo Diamandis, Zachary Frangella

March 2022

Speedup x = A\b with this one easy trick



Outline

Preconditioning Linear Systems

Implementation: `RandomizedPreconditioners.jl`

Examples

Future Directions

We want to quickly solve $Ax = b$

- We focus on large systems of the form

$$(A + \mu I)x = b$$

where $A \in \mathbb{S}_+^n$ and $\mu \geq 0$.

We want to quickly solve $Ax = b$

- We focus on large systems of the form

$$(A + \mu I)x = b$$

where $A \in \mathbb{S}^n_+$ and $\mu \geq 0$.

- "Large" means a direct solve is not computationally feasible.

We want to quickly solve $Ax = b$

- We focus on large systems of the form

$$(A + \mu I)x = b$$

  where $A \in \mathbb{S}^n_+$ and $\mu \geq 0$.

- "Large" means a direct solve is not computationally feasible.

- Ideas can be extended to other systems.

We use the conjugate gradient method (CG)

- CG only requires matrix vector products: $v \mapsto Av$
- CG converges quickly when
  1. The condition number of A is small
  2. The eigenvalues of A are clustered

We use the conjugate gradient method (CG)

- CG only requires matrix vector products: $v \mapsto Av$
- CG converges quickly when
    1. The condition number of A is small
    2. The eigenvalues of A are clustered

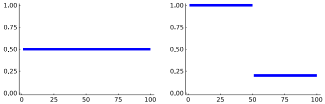


Figure: Easy for CG

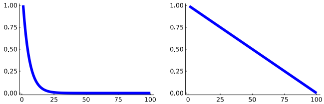


Figure: Hard for CG

A preconditioner can make the spectrum of A "nice"

Goal: Find a *preconditioner* P such that:

1. $v \mapsto P^{-1} v$ is easily evaluated
2. $P^{-1/2}(A + \mu I)P^{-1/2}$ has a "nice" spectrum for CG

Idea: figure out what you want, then approximate

We find the best possible preconditioner and instead of computing it exactly (slow), approximate it (fast).

We precondition using the dominant eigenspace

- Ideally, if we had access to the rank-k eigendecomposition $\lfloor A \rfloor_k = V_k \Lambda_k V_k^T$ and λ_{k+1} we would use

$$P = \frac{1}{\lambda_{k+1} + \mu} V_k (\Lambda_k + \mu I) V_k^T + I - V_k V_k^T.$$

- P admits an explicit cheap to apply inverse.

- Preconditioned system satisfies

$$\kappa_2(P^{-1/2} A_\mu P^{-1/2}) = \frac{\lambda_{k+1} + \mu}{\lambda_n + \mu}.$$

Approximate a decomposition via the Nystöm Sketch

- Computing exact partial eigendecompositions is expensive.
- The Nyström sketch gives an approximate eigendecomposition,

$$\hat{A}_{\mathrm{nys}} = (A\Omega)(\Omega^T A\Omega)^\dagger (A\Omega)^T = \hat{V}\hat{\Lambda}\hat{V}^T.$$

- $\Omega \in \mathbb{R}^{n\times k}$ is a random test matrix
  - A common choice is a standard normal Gaussian matrix.

The Nystöm Sketch comes from a best fit problem

- The Nyström sketch solves the optimization problem,

$$\hat{A}_{\text{nys}} = \underset{\text{range}(\hat{A}) \subset \text{range}(A\Omega)}{\text{argmin}} \|A - \hat{A}\|_F^2.$$

And sketching works well if the spectrum decays

- Approximation error depends on tail-eigenvalues [Tro+17]:

$$\mathbb{E}\|A - \hat{A}_r\| \leq \lambda_{r+1} + \tfrac{r}{k-r+1} \sum_{j>r} \lambda_j.$$

- System is well-conditioned in expectation [FTU21]:

$$\mathbb{E}\left[\kappa\left(P^{-1/2}(A + \mu I)P^{-1/2}\right)\right] < 28.$$

Outline

Preconditioners can be constructed easily

- It only takes two lines of code!

```
using RandomizedPreconditioners
Anys = NystromSketch(A, k, r)
P = NystromPreconditioner(Anys, μ)
```

Preconditioners can be constructed easily

- It only takes two lines of code!

```
using RandomizedPreconditioners
Anys = NystromSketch(A, k, r)
P = NystromPreconditioner(Anys, μ)
```

- And we can get P^{-1} as well:

```
Pinv = NystromPreconditionerInverse(Anys, μ)
```

Preconditioners have efficient operations for solvers

- We use multiple dispatch to implement efficient
  - `ldiv!` for `P` and
  - `mul!` for `Pinv`

Preconditioners have efficient operations for solvers

- We use multiple dispatch to implement efficient
  - `ldiv!` for `P` and
  - `mul!` for `Pinv`
- These preconditioners can be easily passed to interative solvers:

```
using Krylov
x, stats = cg(A+μ*I, b; M=Pinv)
```

```
using IterativeSolvers
x, ch = cg(ATA, b; Pl = P, log=true)
```

Several sketches are included

- Positive semidefinite matrices: Nyström Sketch

```
Â = NystromSketch(A, k, r)
```

Several sketches are included

- Positive semidefinite matrices: Nyström Sketch

```
Â = NystromSketch(A, k, r)
```

- Symmetric matrices: Eigen Sketch

```
Â = EigenSketch(A, k, r)
```

Several sketches are included

- Positive semidefinite matrices: Nyström Sketch

```
Â = NystromSketch(A, k, r)
```

- Symmetric matrices: Eigen Sketch

```
Â = EigenSketch(A, k, r)
```

- General matrices: Randomized SVD

```
Â = RandomizedSVD(A, k, r; q=10)
```

These sketches come with several utilities including

- Fast multiplication:

```
Â = NystromSketch(A, k, r)
Â * v .== Â.U * Â.Λ * Â.U'* v

Â = RandomizedSVD(A, k, r)
Â * v .== Â.U * Â.Λ * Â.V' * v
```

These sketches come with several utilities including

- Fast multiplication:

```
Â = NystromSketch(A, k, r)
Â * v .== Â.U * Â.Λ * Â.U'* v

Â = RandomizedSVD(A, k, r)
Â * v .== Â.U * Â.Λ * Â.V' * v
```

- Adaptive sketch size selection:

```
#Doubles sketch size until ||Â - A|| is small
Â = adaptive_sketch(A, r0, EigenSketch)
```

Outline

Preconditioning Linear Systems

Implementation: `RandomizedPreconditioners.jl`

Examples

Future Directions

CG is faster on regression for small overhead

Ridge regression with $\sim 4.3k$ features (guillermo dataset, OpenML)

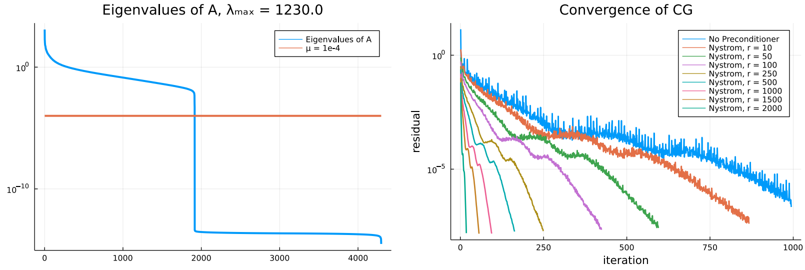


Figure: Spectrum (left) and convergence for various sketch sizes (right)

And it works on large examples too!

Ridge regression with 15k features, solved in <5s on a laptop

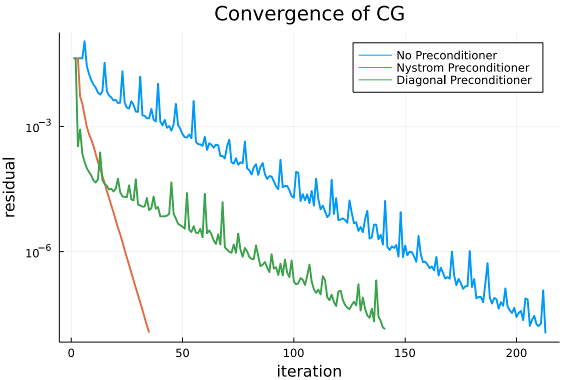


Figure: Nyström PCG vs Jacobi (diagonal) PCG vs vanilla CG

Where to go from here?

- **To use:** `RandomizedPreconditioners.jl`
  - Works with `LinearSolve.jl`
- **To learn:** Zach's paper on Nyström PCG [FTU21]
  - Also check out Martinsson & Tropp survey [MT21]

Outline

Preconditioning Linear Systems

Implementation: `RandomizedPreconditioners.jl`

Examples

Future Directions

Future Work

- Adding additional test matrices
  - *e.g.*, Subsampled Scrambled Fourier Transform
  - Providing better support for sparse matrices
- Adding general preconditioners for nonsymmetric systems
  - This is an open research question
- Performance and robustness
- Applications!

References

[FTU21] Zachary Frangella, Joel A Tropp, and Madeleine Udell. "Randomized Nyström Preconditioning". In: *arXiv preprint arXiv:2110.02820* (2021).

[MT21] PG Martinsson and JA Tropp. "Randomized numerical linear algebra: foundations & algorithms". In: *arXiv preprint arXiv:2002.01387* (2021).

[Tro+17] Joel A Tropp et al. "Practical sketching algorithms for low-rank matrix approximation". In: *SIAM Journal on Matrix Analysis and Applications* 38.4 (2017), pp. 1454–1485.

Thank you

- **Package:** `RandomizedPreconditioners.jl`
- **Contact:** `tdiamand@mit.edu`, `zjf4@cornell.edu`