Understanding *A Fast Contour-Integral Eigensolver for Non-Hermitian Matrices*

Austin Gill
Karen Braman

SDSM&T

April 10, 2018

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Understanding *A Fast Contour-Integral Eigensolver for Non-Hermitian Matrices*

Austin Gill
Karen Braman

SDSM&T

April 10, 2018

Outline

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Outline
1. Introduction
2. Overview
3. The Theory
4. Implementation
5. The Other Puzzle Pieces
6. Results

What's an eigenvalue?

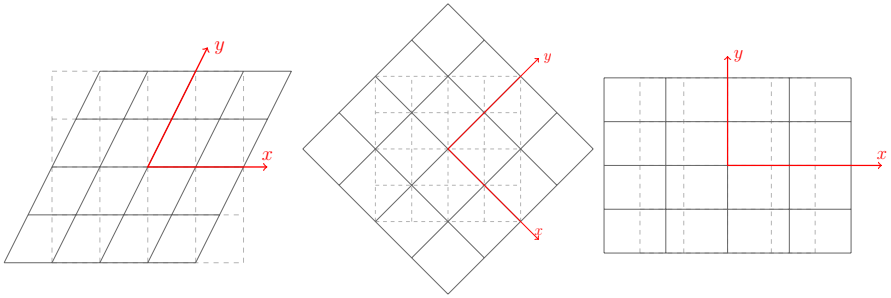
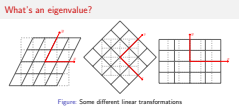


Figure: Some different linear transformations

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY



If we're going to talk about solving the eigenvalue problem, we should probably talk about what an eigenthingy *is*.

We know that if you multiply a vector by a matrix, you get a new vector. So matrices are *transformations*. Wouldn't it be cool if we wanted to know about the vectors that *didn't* change when you apply a transformation to them? We've probably all seen this, but here it is again at twice the speed and three times the fun.

What's an eigenvalue?

"Given a transformation, what vectors map to scaled versions of themselves?"

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

What's an eigenvalue?

"Given a transformation, what vectors map to scaled versions of themselves?"

If we're going to talk about solving the eigenvalue problem, we should probably talk about what an eigenthingy *is*.

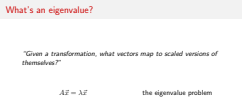
We know that if you multiply a vector by a matrix, you get a new vector. So matrices are *transformations*. Wouldn't it be cool if we wanted to know about the vectors that *didn't* change when you apply a transformation to them? We've probably all seen this, but here it is again at twice the speed and three times the fun.

What's an eigenvalue?

"Given a transformation, what vectors map to scaled versions of themselves?"

$$A\vec{x} = \lambda\vec{x} \qquad \text{the eigenvalue problem}$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY



If we're going to talk about solving the eigenvalue problem, we should probably talk about what an eigenthingy *is*.

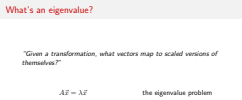
We know that if you multiply a vector by a matrix, you get a new vector. So matrices are *transformations*. Wouldn't it be cool if we wanted to know about the vectors that *didn't* change when you apply a transformation to them? We've probably all seen this, but here it is again at twice the speed and three times the fun.

What's an eigenvalue?

"Given a transformation, what vectors map to scaled versions of themselves?"

$$A\vec{x} = \lambda\vec{x} \qquad \text{the eigenvalue problem}$$
$$A\vec{x} - \lambda\vec{x} = 0$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY



If we're going to talk about solving the eigenvalue problem, we should probably talk about what an eigenthingy *is*.

We know that if you multiply a vector by a matrix, you get a new vector. So matrices are *transformations*. Wouldn't it be cool if we wanted to know about the vectors that *didn't* change when you apply a transformation to them? We've probably all seen this, but here it is again at twice the speed and three times the fun.

What's an eigenvalue?

"Given a transformation, what vectors map to scaled versions of themselves?"

$$A\vec{x} = \lambda\vec{x} \qquad \text{the eigenvalue problem}$$
$$A\vec{x} - \lambda\vec{x} = 0$$
$$(A - \lambda)\vec{x} = 0$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

What's an eigenvalue?
"Given a transformation, what vectors map to scaled versions of themselves?"
$A\vec{x} = \lambda\vec{x}$ the eigenvalue problem
$A\vec{x} - \lambda\vec{x} = 0$
$(A - \lambda)\vec{x} = 0$

If we're going to talk about solving the eigenvalue problem, we should probably talk about what an eigenthingy *is*.

We know that if you multiply a vector by a matrix, you get a new vector. So matrices are *transformations*. Wouldn't it be cool if we wanted to know about the vectors that *didn't* change when you apply a transformation to them? We've probably all seen this, but here it is again at twice the speed and three times the fun.

What's an eigenvalue?

"Given a transformation, what vectors map to scaled versions of themselves?"

$$A\vec{x} = \lambda\vec{x} \qquad \text{the eigenvalue problem}$$
$$A\vec{x} - \lambda\vec{x} = 0$$
$$(A - \lambda)\vec{x} = 0$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

What's an eigenvalue?
"Given a transformation, what vectors map to scaled versions of themselves?"
$A\vec{x} = \lambda\vec{x}$ the eigenvalue problem
$A\vec{x} - \lambda\vec{x} = 0$
$(A - \lambda)\vec{x} = 0$

If we're going to talk about solving the eigenvalue problem, we should probably talk about what an eigenthingy *is*.

We know that if you multiply a vector by a matrix, you get a new vector. So matrices are *transformations*. Wouldn't it be cool if we wanted to know about the vectors that *didn't* change when you apply a transformation to them? We've probably all seen this, but here it is again at twice the speed and three times the fun.

What's an eigenvalue?

"Given a transformation, what vectors map to scaled versions of themselves?"

$$A\vec{x} = \lambda\vec{x} \qquad \text{the eigenvalue problem}$$
$$A\vec{x} - \lambda\vec{x} = 0$$
$$(A - \lambda I)\vec{x} = 0$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

What's an eigenvalue?
"Given a transformation, what vectors map to scaled versions of themselves?"
$A\vec{x} = \lambda\vec{x}$ the eigenvalue problem
$A\vec{x} - \lambda\vec{x} = 0$
$(A - \lambda I)\vec{x} = 0$

If we're going to talk about solving the eigenvalue problem, we should probably talk about what an eigenthingy *is*.

We know that if you multiply a vector by a matrix, you get a new vector. So matrices are *transformations*. Wouldn't it be cool if we wanted to know about the vectors that *didn't* change when you apply a transformation to them? We've probably all seen this, but here it is again at twice the speed and three times the fun.

What's an eigenvalue?

"Given a transformation, what vectors map to scaled versions of themselves?"

$$
\begin{aligned}
A\vec{x} &= \lambda\vec{x} && \text{the eigenvalue problem} \\
A\vec{x} - \lambda\vec{x} &= 0 \\
(A - \lambda I)\vec{x} &= 0 \\
\det(A - \lambda I) &= 0 && \text{the characteristic polynomial } p(\lambda)
\end{aligned}
$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

What's an eigenvalue?
"Given a transformation, what vectors map to scaled versions of themselves?"
$A\vec{x} = \lambda\vec{x}$ the eigenvalue problem
$A\vec{x} - \lambda\vec{x} = 0$
$(A - \lambda I)\vec{x} = 0$
$\det(A - \lambda I) = 0$ the characteristic polynomial $p(\lambda)$

If we're going to talk about solving the eigenvalue problem, we should probably talk about what an eigenthingy *is*.

We know that if you multiply a vector by a matrix, you get a new vector. So matrices are *transformations*. Wouldn't it be cool if we wanted to know about the vectors that *didn't* change when you apply a transformation to them? We've probably all seen this, but here it is again at twice the speed and three times the fun.

I declare us done!

We know how to find roots, so we're done, right?

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

I declare us done!
We know how to find roots, so we're done, right?

We've reduced our problem to one we know how to solve, so we're done, right? *Well*, not actually.

Finding the characteristic polynomial requires taking the determinant, which ranges in complexity from $\mathcal{O}(n!)$ to $\sim \mathcal{O}(n^3)$ in special cases.

Then, even if we did magically get the characteristic polynomial, we have to find its roots. To do this we would have to deal with instability, even when the underlying eigenvalue problem is relatively stable.

I declare us done!

We know how to find roots, so we're done, right?

- Just *finding* the characteristic polynomial is hard.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

I declare us done!
We know how to find roots, so we're done, right?
- Just *finding* the characteristic polynomial is hard.

We've reduced our problem to one we know how to solve, so we're done, right? *Well*, not actually.

Finding the characteristic polynomial requires taking the determinant, which ranges in complexity from $\mathcal{O}(n!)$ to $\sim \mathcal{O}(n^3)$ in special cases.

Then, even if we did magically get the characteristic polynomial, we have to find its roots. To do this we would have to deal with instability, even when the underlying eigenvalue problem is relatively stable.

I declare us done!

We know how to find roots, so we're done, right?

- Just *finding* the characteristic polynomial is hard.
- Even if we had the characteristic polynomial, root finding problems are ill-conditioned, even when the underlying eigenvalue problem isn't [1].

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

I declare us done!
We know how to find roots, so we're done, right?
- Just finding the characteristic polynomial is hard.
- Even if we had the characteristic polynomial, root finding problems are ill-conditioned, even when the underlying eigenvalue problem isn't [1].

We've reduced our problem to one we know how to solve, so we're done, right? *Well*, not actually.

Finding the characteristic polynomial requires taking the determinant, which ranges in complexity from $\mathcal{O}(n!)$ to $\sim \mathcal{O}(n^3)$ in special cases.

Then, even if we did magically get the characteristic polynomial, we have to find its roots. To do this we would have to deal with instability, even when the underlying eigenvalue problem is relatively stable.

Outline

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Outline
Introduction
Overview
The Theory
Implementation
The Other Puzzle Pieces
Results

A proposed solution

There are established methods for solving the eigenvalue problem, but in 2003 Sakurai and Sugiura [2] proposed a *new* kind of eigensolver (ab)using ideas from functional and complex analysis.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

A proposed solution
There are established methods for solving the eigenvalue problem, but in 2003 Sakurai and Sugiura [2] proposed a new kind of eigensolver (ab)using ideas from functional and complex analysis.

The paper we looked at is an extension on previous work in this area. Other eigensolvers in the same vein required special structure before they could work. The algorithm we looked at solves a more general problem for *Non-Hermitian* matrices.

Hermitian matrices are the Complex equivalent to symmetric matrices. You take the conjugate of everything, transpose it, and if what you get is the same as what you had when you started, that's what a Hermitian matrix is.

A proposed solution

There are established methods for solving the eigenvalue problem, but in 2003 Sakurai and Sugiura [2] proposed a *new* kind of eigensolver (ab)using ideas from functional and complex analysis.

The paper we researched, *A Fast Contour-Integral Eigensolve for Non-Hermitian Matrices* [3] is an extension of previous work in this area.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

A proposed solution

There are established methods for solving the eigenvalue problem, but in 2003 Sakurai and Sugiura [2] proposed a *new* kind of eigensolver (ab)using ideas from functional and complex analysis.

The paper we researched, *A Fast Contour-Integral Eigensolve for Non-Hermitian Matrices* [3] is an extension of previous work in this area.

The paper we looked at is an extension on previous work in this area. Other eigensolvers in the same vein required special structure before they could work. The algorithm we looked at solves a more general problem for *Non-Hermitian* matrices.

Hermitian matrices are the Complex equivalent to symmetric matrices. You take the conjugate of everything, transpose it, and if what you get is the same as what you had when you started, that's what a Hermitian matrix is.

30000 foot view

Our `FastEig` eigensolver will operate in two stages.

30000 foot view
Our FastEig eigensolver will operate in two stages.

Our eigensolver breaks the problem into two stages. In the first stage, it combs through the complex plane with a fairly coarse comb looking for regions where eigenvalues are "dense".

The second stage is a much finer comb through of the Complex plane. It takes those regions, along with our best guess as to how many eigenvalues there are in each of them, and attempts to find the eigenvalues in that region.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

30000 foot view

Our `FastEig` eigensolver will operate in two stages.

Stage 1 Search the complex plane for regions that are "dense" with eigenvalues

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

30000 foot view
Our FastEig eigensolver will operate in two stages.
Stage 1 Search the complex plane for regions that are "dense" with eigenvalues

Our eigensolver breaks the problem into two stages. In the first stage, it combs through the complex plane with a fairly coarse comb looking for regions where eigenvalues are "dense".

The second stage is a much finer comb through of the Complex plane. It takes those regions, along with our best guess as to how many eigenvalues there are in each of them, and attempts to find the eigenvalues in that region.

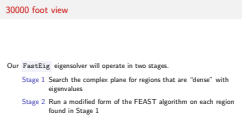
30000 foot view

Our `FastEig` eigensolver will operate in two stages.

Stage 1 Search the complex plane for regions that are "dense" with eigenvalues

Stage 2 Run a modified form of the FEAST algorithm on each region found in Stage 1

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY



Our eigensolver breaks the problem into two stages. In the first stage, it combs through the complex plane with a fairly coarse comb looking for regions where eigenvalues are "dense".

The second stage is a much finer comb through of the Complex plane. It takes those regions, along with our best guess as to how many eigenvalues there are in each of them, and attempts to find the eigenvalues in that region.

Outline

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Outline
Introduction
Overview
The Theory
Implementation
The Other Puzzle Pieces
Results

Before we can count. . .

Suppose λ_j for $j = 1, \ldots, s$ are the s eigenvalues inside the contour Γ in the complex plane.

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Before we can count. . .
Suppose λ_j for $j = 1, \ldots, s$ are the s eigenvalues inside the contour Γ in the complex plane.

I'm going to handwave my way through most of this, because it turns out that it's not too terribly important for implementing the numerical algorithm. So we "build" $\mathbf{\Phi}$. . . except we don't actually. . .

Before we can count. . .

Suppose λ_j for $j = 1, \ldots, s$ are the s eigenvalues inside the contour Γ in the complex plane.

Now consider the residue

$$\phi(z) = \frac{1}{2\pi \mathbf{i}} \int_\Gamma \frac{1}{\mu - z} \mathrm{d}\mu$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Before we can count. . .
Suppose λ_j for $j = 1, \ldots, s$ are the s eigenvalues inside the contour Γ in the complex plane.
Now consider the residue
$\phi(z) = \frac{1}{2\pi \mathbf{i}} \int_\Gamma \frac{1}{\mu - z} \mathrm{d}\mu$

I'm going to handwave my way through most of this, because it turns out that it's not too terribly important for implementing the numerical algorithm. So we "build" $\mathbf{\Phi}$. . . except we don't actually. . .

Before we can count. . .

Suppose λ_j for $j = 1, \ldots, s$ are the s eigenvalues inside the contour Γ in the complex plane.

Now consider the residue

$$\phi(z) = \frac{1}{2\pi \mathbf{i}} \int_\Gamma \frac{1}{\mu - z} \mathrm{d}\mu$$

We now want to build the spectral projector $\mathbf{\Phi}$ to the eigenspace

$$\mathrm{span}\{\vec{x}_1, \ldots, \vec{x}_s\}$$

spanned by the corresponding s eigenvectors.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Before we can count. . .
Suppose λ_j for $j = 1, \ldots, s$ are the s eigenvalues inside the contour Γ in the complex plane.
Now consider the residue
$\phi(z) = \frac{1}{2\pi \mathbf{i}} \int_\Gamma \frac{1}{\mu - z} \mathrm{d}\mu$
We now want to build the spectral projector $\mathbf{\Phi}$ to the eigenspace
$\mathrm{span}\{\vec{x}_1, \ldots, \vec{x}_s\}$
spanned by the corresponding s eigenvectors.

I'm going to handwave my way through most of this, because it turns out that it's not too terribly important for implementing the numerical algorithm. So we "build" $\mathbf{\Phi}$. . . except we don't actually. . .

Before we can count. . .

So, we build $\mathbf{\Phi}$ using magic

$$\mathbf{\Phi} \equiv \phi(A) = \frac{1}{2\pi \mathbf{i}} \int_{\Gamma} (\mu I - A)^{-1} \mathrm{d}\mu$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Before we can count. . .
So, we build $\mathbf{\Phi}$ using magic
$\mathbf{\Phi} \equiv \phi(A) = \frac{1}{2\pi \mathbf{i}} \int_{\Gamma} (\mu I - A)^{-1} \mathrm{d}\mu$

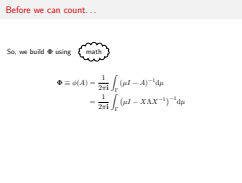
I'm going to handwave my way through most of this, because it turns out that it's not too terribly important for implementing the numerical algorithm. So we "build" $\mathbf{\Phi}$. . . except we don't actually. . .

Before we can count. . .

So, we build $\mathbf{\Phi}$ using 

$$\mathbf{\Phi} \equiv \phi(A) = \frac{1}{2\pi \mathbf{i}} \int_{\Gamma} (\mu I - A)^{-1} \mathrm{d}\mu$$
$$= \frac{1}{2\pi \mathbf{i}} \int_{\Gamma} \left(\mu I - X \Lambda X^{-1}\right)^{-1} \mathrm{d}\mu$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY



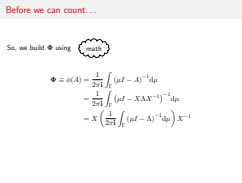
I'm going to handwave my way through most of this, because it turns out that it's not too terribly important for implementing the numerical algorithm. So we "build" $\mathbf{\Phi}$. . . except we don't actually. . .

Before we can count. . .

So, we build $\mathbf{\Phi}$ using 

$$\begin{aligned}\mathbf{\Phi} \equiv \phi(A) &= \frac{1}{2\pi\mathbf{i}} \int_\Gamma (\mu I - A)^{-1} \mathrm{d}\mu \\ &= \frac{1}{2\pi\mathbf{i}} \int_\Gamma \left(\mu I - X\Lambda X^{-1}\right)^{-1} \mathrm{d}\mu \\ &= X \left(\frac{1}{2\pi\mathbf{i}} \int_\Gamma (\mu I - \Lambda)^{-1} \mathrm{d}\mu \right) X^{-1}\end{aligned}$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY



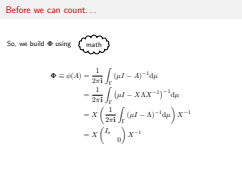
I'm going to handwave my way through most of this, because it turns out that it's not too terribly important for implementing the numerical algorithm. So we "build" $\mathbf{\Phi}$. . . except we don't actually. . .

Before we can count. . .

So, we build $\mathbf{\Phi}$ using 

$$\begin{aligned}\mathbf{\Phi} \equiv \phi(A) &= \frac{1}{2\pi\mathbf{i}} \int_\Gamma (\mu I - A)^{-1} \mathrm{d}\mu \\ &= \frac{1}{2\pi\mathbf{i}} \int_\Gamma \left(\mu I - X\Lambda X^{-1}\right)^{-1} \mathrm{d}\mu \\ &= X \left(\frac{1}{2\pi\mathbf{i}} \int_\Gamma (\mu I - \Lambda)^{-1} \mathrm{d}\mu \right) X^{-1} \\ &= X \begin{pmatrix} I_s & \\ & 0 \end{pmatrix} X^{-1}\end{aligned}$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY



I'm going to handwave my way through most of this, because it turns out that it's not too terribly important for implementing the numerical algorithm. So we "build" $\mathbf{\Phi}$. . . except we don't actually. . .

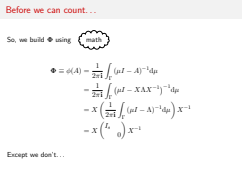
Before we can count...

So, we build $\mathbf{\Phi}$ using 

$$\begin{aligned}\mathbf{\Phi} \equiv \phi(A) &= \frac{1}{2\pi\mathbf{i}} \int_\Gamma (\mu I - A)^{-1} \mathrm{d}\mu \\ &= \frac{1}{2\pi\mathbf{i}} \int_\Gamma \left(\mu I - X\Lambda X^{-1}\right)^{-1} \mathrm{d}\mu \\ &= X \left(\frac{1}{2\pi\mathbf{i}} \int_\Gamma (\mu I - \Lambda)^{-1} \mathrm{d}\mu \right) X^{-1} \\ &= X \begin{pmatrix} I_s & \\ & 0 \end{pmatrix} X^{-1}\end{aligned}$$

Except we don't...

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY



I'm going to handwave my way through most of this, because it turns out that it's not too terribly important for implementing the numerical algorithm. So we "build" $\mathbf{\Phi}$... except we don't actually...

Randomization is cool

Instead we approximate it using randomization. [3]

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Randomization is cool
Instead we approximate it using randomization. [3]

We don't actually build $\mathbf{\Phi}$, we approximate it. Here's what the paper we looked at had to say. Warning bells and flashing lights should be going off right now. What the heck does "appropriately chosen" even mean?

The paper didn't tell us. All it said was "Let Y be a random matrix."

So we form this Z thing, *except* we actually don't...

Randomization is cool

Instead we approximate it using randomization. [3]

> *Instead, the basis of the eigenspace can be extracted with randomization, where the product of $\mathbf{\Phi}$ and an appropriately chosen random matrix Y is computed*

Randomization is cool
Instead we approximate it using randomization. [3]
Instead, the basis of the eigenspace can be extracted with randomization, where the product of $\mathbf{\Phi}$ and an appropriately chosen random matrix Y is computed

We don't actually build $\mathbf{\Phi}$, we approximate it. Here's what the paper we looked at had to say. Warning bells and flashing lights should be going off right now. What the heck does "appropriately chosen" even mean?

The paper didn't tell us. All it said was "Let Y be a random matrix."

So we form this Z thing, *except* we actually don't...

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Randomization is cool

Instead we approximate it using randomization. [3]

> *Instead, the basis of the eigenspace can be extracted with randomization, where the product of $\mathbf{\Phi}$ and an appropriately chosen random matrix Y is computed*

Instead we form Z

$$Z = \mathbf{\Phi} Y = \frac{1}{2\pi \mathbf{i}} \int_{\Gamma} (\mu I - A)^{-1} Y \mathrm{d}\mu \tag{1}$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Randomization is cool
Instead we approximate it using randomization. [3]
Instead, the basis of the eigenspace can be extracted with randomization, where the product of $\mathbf{\Phi}$ and an appropriately chosen random matrix Y is computed
Instead we form Z
$Z = \mathbf{\Phi} Y = \frac{1}{2\pi \mathbf{i}} \int_{\Gamma} (\mu I - A)^{-1} Y \mathrm{d}\mu$ (1)

We don't actually build $\mathbf{\Phi}$, we approximate it. Here's what the paper we looked at had to say. Warning bells and flashing lights should be going off right now. What the heck does "appropriately chosen" even mean?

The paper didn't tell us. All it said was "Let Y be a random matrix."

So we form this Z thing, *except* we actually don't...

Randomization is cool

Instead we approximate it using randomization. [3]

> *Instead, the basis of the eigenspace can be extracted with randomization, where the product of $\mathbf{\Phi}$ and an appropriately chosen random matrix Y is computed*

Instead we form Z

$$Z = \mathbf{\Phi} Y = \frac{1}{2\pi \mathbf{i}} \int_{\Gamma} (\mu I - A)^{-1} Y \mathrm{d}\mu \tag{1}$$

Except we don't...

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Randomization is cool
Instead we approximate it using randomization. [3]
Instead, the basis of the eigenspace can be extracted with randomization, where the product of $\mathbf{\Phi}$ and an appropriately chosen random matrix Y is computed
Instead we form Z
$$Z = \mathbf{\Phi} Y = \frac{1}{2\pi \mathbf{i}} \int_{\Gamma} (\mu I - A)^{-1} Y \mathrm{d}\mu \tag{1}$$
Except we don't...

We don't actually build $\mathbf{\Phi}$, we approximate it. Here's what the paper we looked at had to say. Warning bells and flashing lights should be going off right now. What the heck does "appropriately chosen" even mean?

The paper didn't tell us. All it said was "Let Y be a random matrix."

So we form this Z thing, *except* we actually don't...

Before we can run we have to ~~walk~~ build HSS approximations

The `FastEig` algorithm makes heavy use of Hierarchically Semi Separable matrix approximations as a precomputation step in both stages.

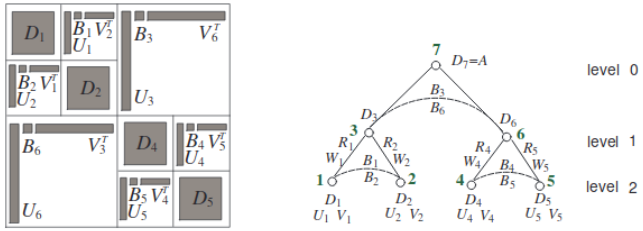


FIG. 2.1. *A 3-level HSS matrix and its corresponding HSS tree.*

Figure: An HSS approximation of A from Figure 2.1 of [4]

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Before we can run we have to ~~walk~~ build HSS approximations
The `FastEig` algorithm makes heavy use of Hierarchically Semi Separable matrix approximations as a precomputation step in both stages.
Figure: An HSS approximation of A from Figure 2.1 of [4]

Before we can dive into the meat of the algorithm, I have to make a side note here about Hierarchically Semi Separable matrix approximations.

I don't really know that much about them, other than they approximate any matrix with one that is rank-structured.

It turns out to not be that big of a deal because

1. There's a library for that, and
2. The code runs *much* faster when we remove HSS approximations all together.

Before we can run we have to ~~walk~~ build HSS approximations

The `FastEig` algorithm makes heavy use of Hierarchically Semi Separable matrix approximations as a precomputation step in both stages.

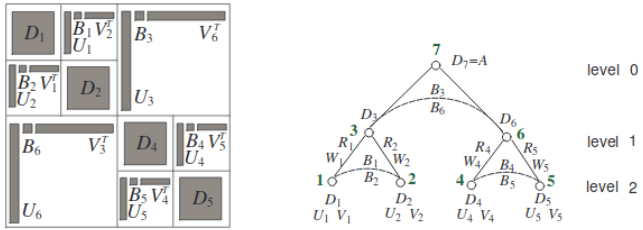


FIG. 2.1. *A 3-level HSS matrix and its corresponding HSS tree.*

Figure: An HSS approximation of A from Figure 2.1 of [4]

There's a library [5] for that...

Before we can dive into the meat of the algorithm, I have to make a side note here about Hierarchically Semi Separable matrix approximations.

I don't really know that much about them, other than they approximate any matrix with one that is rank-structured.

It turns out to not be that big of a deal because

1. There's a library for that, and
2. The code runs *much* faster when we remove HSS approximations all together.

So how do we count eigenvalues?

This is what we spent our time implementing

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

So how do we count eigenvalues?
This is what we spent our time implementing

Stage 1 consists of building a count estimate for the number of eigenvalues inside a given region. So how is that going to work?

Let's start with an example.

So how do we count eigenvalues?

This is what we spent our time implementing

Let's start with an example

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

So how do we count eigenvalues?
This is what we spent our time implementing
Let's start with an example

Stage 1 consists of building a count estimate for the number of eigenvalues inside a given region. So how is that going to work?

Let's start with an example.

Counting eigenvalues

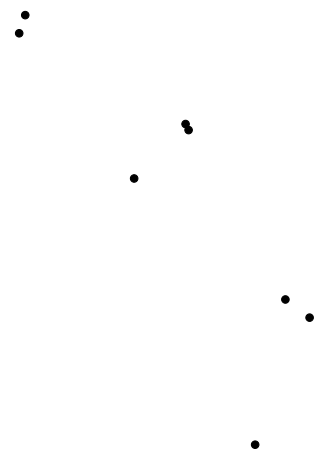


Figure: Quadsection with threshold $k = 2$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Counting eigenvalues

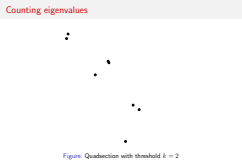


Figure: Quadsection with threshold $k = 2$

So we want to filter out regions of the plane where there are eigenvalues, and we want to do so with a threshold $k = 2$. So we pick a region and quadsect it. Then on each of the four regions, we continue to build a count estimate and quadsect until that count estimate hits our threshold.

Counting eigenvalues

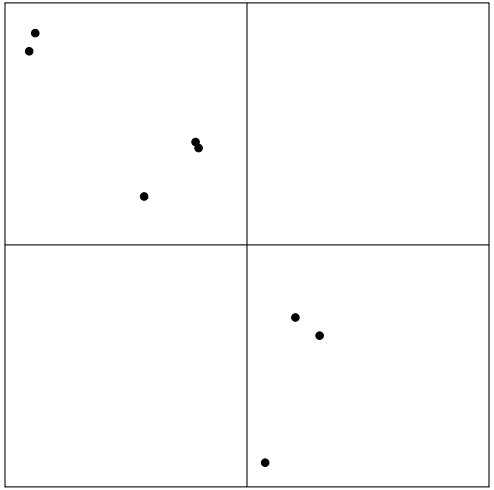


Figure: Quadsection with threshold $k = 2$

Counting eigenvalues

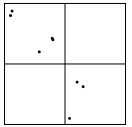


Figure: Quadsection with threshold $k = 2$

So we want to filter out regions of the plane where there are eigenvalues, and we want to do so with a threshold $k = 2$. So we pick a region and quadsect it. Then on each of the four regions, we continue to build a count estimate and quadsect until that count estimate hits our threshold.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Counting eigenvalues

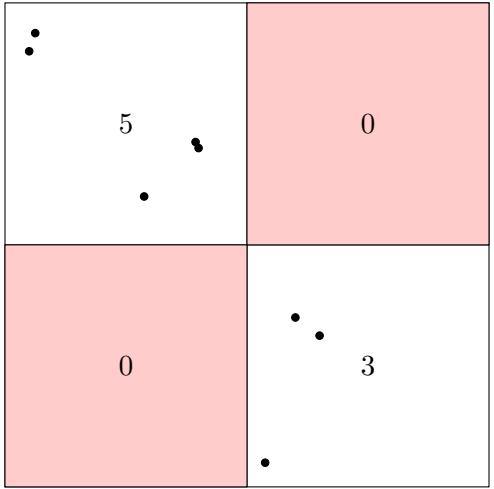


Figure: Quadsection with threshold $k = 2$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Counting eigenvalues

Figure: Quadsection with threshold $k = 2$

So we want to filter out regions of the plane where there are eigenvalues, and we want to do so with a threshold $k = 2$. So we pick a region and quadsect it. Then on each of the four regions, we continue to build a count estimate and quadsect until that count estimate hits our threshold.

Counting eigenvalues

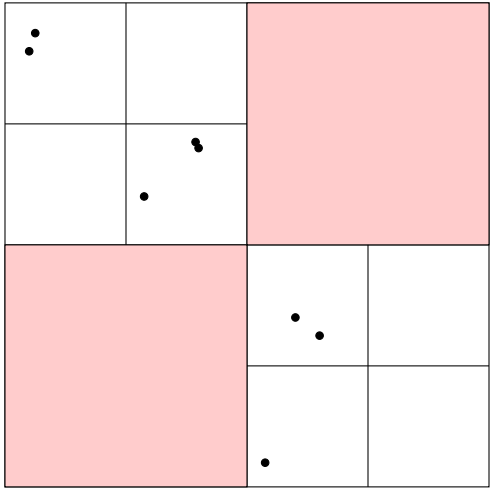


Figure: Quadsection with threshold $k = 2$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Counting eigenvalues

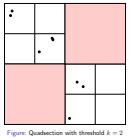


Figure: Quadsection with threshold $k = 2$

So we want to filter out regions of the plane where there are eigenvalues, and we want to do so with a threshold $k = 2$. So we pick a region and quadsect it. Then on each of the four regions, we continue to build a count estimate and quadsect until that count estimate hits our threshold.

Counting eigenvalues

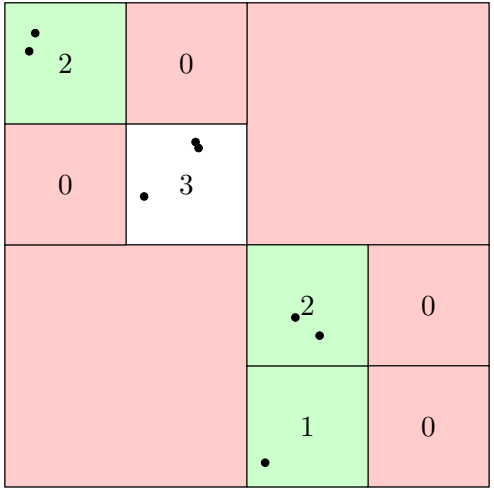


Figure: Quadsection with threshold $k = 2$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Counting eigenvalues

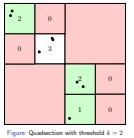


Figure: Quadsection with threshold $k = 2$

So we want to filter out regions of the plane where there are eigenvalues, and we want to do so with a threshold $k = 2$. So we pick a region and quadsect it. Then on each of the four regions, we continue to build a count estimate and quadsect until that count estimate hits our threshold.

Counting eigenvalues

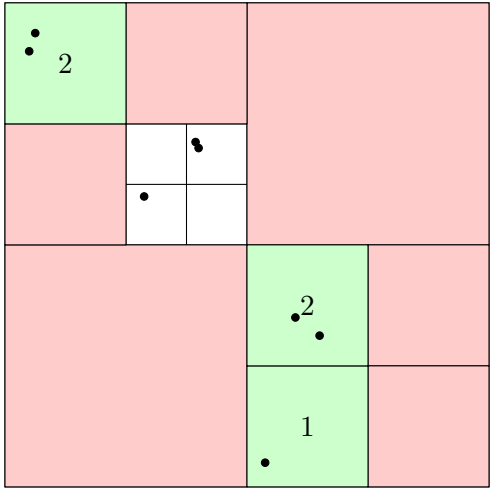


Figure: Quadsection with threshold $k = 2$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Counting eigenvalues

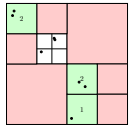


Figure: Quadsection with threshold $k = 2$

So we want to filter out regions of the plane where there are eigenvalues, and we want to do so with a threshold $k = 2$. So we pick a region and quadsect it. Then on each of the four regions, we continue to build a count estimate and quadsect until that count estimate hits our threshold.

Counting eigenvalues

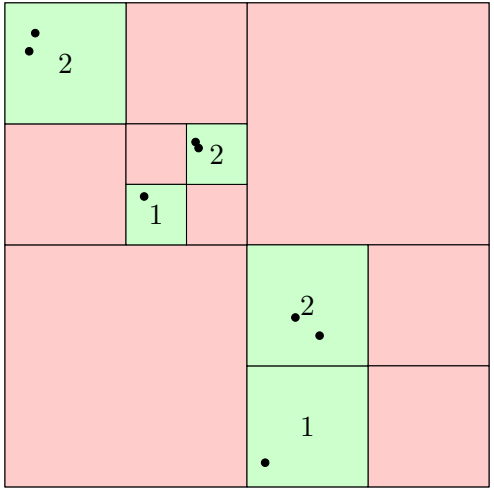


Figure: Quadsection with threshold $k = 2$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Counting eigenvalues

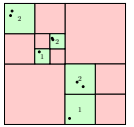


Figure: Quadsection with threshold $k = 2$

So we want to filter out regions of the plane where there are eigenvalues, and we want to do so with a threshold $k = 2$. So we pick a region and quadsect it. Then on each of the four regions, we continue to build a count estimate and quadsect until that count estimate hits our threshold.

Outline

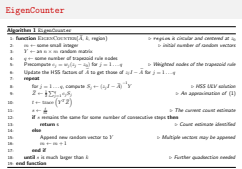
SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Outline
Introduction
Overview
The Theory
Implementation
The Other Puzzle Pieces
Results

EigenCounter

Algorithm 1 `EigenCounter`

function EigenCounter($\widetilde{A}$, k, region) ▷ *region is circular and centered at* z_0
  $m \leftarrow$ some small integer ▷ *initial number of random vectors*
  $Y \leftarrow$ an $n \times m$ random matrix
  $q \leftarrow$ some number of trapezoid rule nodes
  Precompute $c_j = w_j(z_j - z_0)$ for $j = 1 \dots q$ ▷ *Weighted nodes of the trapezoid rule*
  Update the HSS factors of $\widetilde{A}$ to get those of $z_j I - \widetilde{A}$ for $j = 1 \dots q$
  repeat
    for $j = 1 \dots q$, compute $S_j \leftarrow (z_j I - \widetilde{A})^{-1} Y$ ▷ *HSS ULV solution*
    $\widetilde{Z} \leftarrow \frac{1}{2} \sum_{j=1}^{q} c_j S_j$ ▷ *An approximation of* (1)
    $t \leftarrow \operatorname{trace}\left(Y^T \widetilde{Z}\right)$
    $s \leftarrow \frac{t}{m}$ ▷ *The current count estimate*
    if s remains the same for some number of consecutive steps **then**
      return s ▷ *Count estimate identified*
    else
      Append new random vector to Y ▷ *Multiple vectors may be appened*
      $m \leftarrow m + 1$
    end if
  until s is much larger than k ▷ *Further quadsection needed*
end function



So here's the algorithm that counts the number of eigenvalues inside some contour.

First we initialize some values. More warning bells should be going off. "Let m be a small integer"?! Over the last year we've found quite a few places where the authors aren't really specific, so we've either made blind guesses or scraped together enough contextual information to figure out something reasonable.

The next thing we do is some precomputations, and then we iteratively approximate that Z creature from before, each time hopefully getting a better and better approximation.

Trapezoid nodes

We parameterize the q nodes along a circle centered at z_0 with radius r as

$$z_j = z_0 + re^{\mathbf{i}\pi t_j}$$

where

$$t_j = -1 + \frac{2(j-1)}{q}$$

for $j = 1, \ldots, q$. Then the corresponding weights w_j for each node is just $\frac{2}{q}$, and we have

$$c_j = w_j(z_j - z_0)$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Trapezoid nodes
We parameterize the q nodes along a circle centered at z_0 with radius r as
$z_j = z_0 + re^{\mathbf{i}\pi t_j}$
where
$t_j = -1 + \frac{2(j-1)}{q}$
for $j = 1, \ldots, q$. Then the corresponding weights w_j for each node is just $\frac{2}{q}$, and we have
$c_j = w_j(z_j - z_0)$

This actually turned out to be one of the things we guessed at initially, and had not so good results with. After we dig more digging, and found this information spread throughout the paper's proof of the optimality of the trapezoid rule.

Trapezoid nodes

```
j = 1:num_points;
% t_j = -1 + 2(j-1)/q
tj = -1 + 2 * (j - 1) / num_points;

% z_j = z_0 + re^{i\pi t_j}
nodes = center + radius * exp(pi * 1i * tj);
weights = (2 / num_points);

% c_j = w_j(z_j - z_0)
cj = weights * (nodes - center);
```


So here's what that ends up looking like implemented in Matlab.

Updating the HSS factors

The algorithm then states to update the HSS factors D, U, R, B, W, and V of $\widetilde{A}$ to get those of $z_j I - \widetilde{A}$ for $j = 1, \ldots, q$.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Then we have to precompute and save some shifted factors. The paper was fairly handwavy about this. They didn't show how to shift the HSS factors for the problem we needed out of "notational convenience".

It ended up being fairly reasonable. We just had to shift the D subblocks and negate everything else. (Look at picture?)

Updating the HSS factors
The algorithm then states to update the HSS factors D, U, R, B, W, and V of $\widetilde{A}$ to get those of $z_j I - \widetilde{A}$ for $j = 1, \ldots, q$.

Updating the HSS factors

The algorithm then states to update the HSS factors D, U, R, B, W, and V of $\widetilde{A}$ to get those of $z_jI - \widetilde{A}$ for $j = 1, \ldots, q$.

All generators of $z_jI - \widetilde{A}$ are the same as $\widetilde{A}$, except for the diagonal D_i subblocks and negating each entry in the rest of the subblocks U_i, R_i, B_i, W_i, and V_i

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Then we have to precompute and save some shifted factors. The paper was fairly handwavy about this. They didn't show how to shift the HSS factors for the problem we needed out of "notational convenience".

It ended up being fairly reasonable. We just had to shift the D subblocks and negate everything else. (Look at picture?)

Updating the HSS factors

The algorithm then states to update the HSS factors D, U, R, B, W, and V of $\widetilde{A}$ to get those of $z_jI - \widetilde{A}$ for $j = 1, \ldots, q$.

All generators of $z_jI - \widetilde{A}$ are the same as $\widetilde{A}$, except for the diagonal D_i subblocks and negating each entry in the rest of the subblocks U_i, R_i, B_i, W_i, and V_i

So we update $D_i \leftarrow zI - D_i$ and negate everything else

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Updating the HSS factors
The algorithm then states to update the HSS factors D, U, R, B, W, and V of $\widetilde{A}$ to get those of $z_jI - \widetilde{A}$ for $j = 1, \ldots, q$.
All generators of $z_jI - \widetilde{A}$ are the same as $\widetilde{A}$, except for the diagonal D_i subblocks and negating each entry in the rest of the subblocks U_i, R_i, B_i, W_i, and V_i
So we update $D_i \leftarrow zI - D_i$ and negate everything else

Then we have to precompute and save some shifted factors. The paper was fairly handwavy about this. They didn't show how to shift the HSS factors for the problem we needed out of "notational convenience".

It ended up being fairly reasonable. We just had to shift the D subblocks and negate everything else. (Look at picture?)

Computing S_j

Computing

$$S_j = \left(z_j I - \widetilde{A}\right)^{-1} Y$$

is equivalent to solving

$$\left(z_j I - \widetilde{A}\right) S_j = Y$$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Computing S_j
Computing $S_j = \left(z_j I - \widetilde{A}\right)^{-1} Y$
is equivalent to solving $\left(z_j I - \widetilde{A}\right) S_j = Y$

Then we need to compute a bunch of these S matrices. This means solving a matrix-matrix system, which can be done columnwise.

Then we need to save the results for later computation.

Computing S_j

Computing

$$S_j = \left(z_j I - \widetilde{A}\right)^{-1} Y$$

is equivalent to solving

$$\left(z_j I - \widetilde{A}\right) S_j = Y$$

which can be done column-wise. We can solve the system $AS = Y$ as follows in pseudocode

for $\vec{y}$ in Y **do**
  Solve $A\vec{s} = \vec{y}$
  Append $S \leftarrow \vec{s}$
end for

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Computing S_j

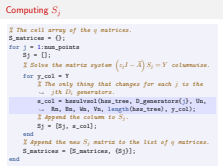
Computing
$$S_j = \left(z_j I - \widetilde{A}\right)^{-1} Y$$
is equivalent to solving
$$\left(z_j I - \widetilde{A}\right) S_j = Y$$
which can be done column-wise. We can solve the system $AS = Y$ as follows in pseudocode
for $\vec{y}$ in Y **do**
Solve $A\vec{s} = \vec{y}$
Append $S \leftarrow \vec{s}$
end for

Then we need to compute a bunch of these S matrices. This means solving a matrix-matrix system, which can be done columnwise.

Then we need to save the results for later computation.

Computing S_j

```
% The cell array of the q matrices.
S_matrices = {};
for j = 1:num_points
    Sj = [];
    % Solve the matrix system (z_j I - Ã) S_j = Y columnwise.
    for y_col = Y
        % The only thing that changes for each j is the
        ↪ jth D_i generators.
        s_col = hssulvsol(hss_tree, D_generators{j}, Un,
        ↪ Rn, Bn, Wn, Vn, length(hss_tree), y_col);
        % Append the column to S_j.
        Sj = [Sj, s_col];
    end
    % Append the new S_j matrix to the list of q matrices.
    S_matrices = [S_matrices, {Sj}];
end
```

Here's what that looks like in code. The highlighted line is where the heavy work gets done.

Computing S_j

Profiling the code revealed that this was *slow*.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Computing S_j
Profiling the code revealed that this was slow.

Unfortunately, I had the bright idea to profile our code because I was bored and didn't want to write my research paper for all this.

It turns out computing each S_j is pretty slow. We'll talk about this some more.

Computing S_j

Profiling the code revealed that this was *slow*.

Eliminating the HSS approximations resulted in a speedup by two orders of magnitude.

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Computing S_j
Profiling the code revealed that this was slow.
Eliminating the HSS approximations resulted in a speedup by two orders of magnitude.

Unfortunately, I had the bright idea to profile our code because I was bored and didn't want to write my research paper for all this.

It turns out computing each S_j is pretty slow. We'll talk about this some more.

Computing S_j

Profiling the code revealed that this was *slow*.

Eliminating the HSS approximations resulted in a speedup by two orders of magnitude.

Performance vs complexity.

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Computing S_j
Profiling the code revealed that this was slow.
Eliminating the HSS approximations resulted in a speedup by two orders of magnitude.
Performance vs complexity.

Unfortunately, I had the bright idea to profile our code because I was bored and didn't want to write my research paper for all this.

It turns out computing each S_j is pretty slow. We'll talk about this some more.

Approximating Z

We approximate $Z = \Phi Y$ as

$$\widetilde{Z} = \frac{1}{2} \sum_{j=1}^{q} c_j S_j$$

```
Z = 0;
for j = 1:num_points
    Z = Z + (coefficients(j) * S_matrices{j});
end
Z = 0.5 * Z;
```


Approximating Z
We approximate $Z = \Phi Y$ as
$\widetilde{Z} = \frac{1}{2} \sum_{j=1}^{q} c_j S_j$
Z = 0;
for j = 1:num_points
Z = Z + (coefficients(j) * S_matrices{j});
end
Z = 0.5 * Z;

Now we can finally approximate the Z creature from before. This is where we actually perform the integration.

Current step estimate

Now we compute our current-step eigenvalue count estimate $s = \frac{t}{m}$ where $t = \text{trace}\left(Y^T \widetilde{Z}\right)$.

```
total_trace = trace(Y' * Z);
count_estimate = total_trace / num_rand_vectors;
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Current step estimate
Now we compute our current-step eigenvalue count estimate $s = \frac{t}{m}$ where $t = \text{trace}\left(Y^T \widetilde{Z}\right)$.
total_trace = trace(Y' * Z);
count_estimate = total_trace / num_rand_vectors;

Here's somewhere where we deviated from the paper. We strongly believe it's a typo, but they said to increment

$$t = t + \text{trace}(Y^T \widetilde{Z})$$

in the algorithm description, but then in the body of the paper they wouldn't...

Current step estimate

Now we compute our current-step eigenvalue count estimate $s = \frac{t}{m}$ where $t = \text{trace}\left(Y^T \widetilde{Z}\right)$.

```
total_trace = trace(Y' * Z);
count_estimate = total_trace / num_rand_vectors;
```

But s turns out to be a complex number...

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Here's somewhere where we deviated from the paper. We strongly believe it's a typo, but they said to increment

$$t = t + \text{trace}(Y^T \widetilde{Z})$$

in the algorithm description, but then in the body of the paper they wouldn't...

Current step estimate

Now we compute our current-step eigenvalue count estimate $s = \frac{t}{m}$ where $t = \text{trace}\left(Y^T \widetilde{Z}\right)$.

```
total_trace = trace(Y' * Z);
count_estimate = total_trace / num_rand_vectors;
```

But s turns out to be a complex number... So we assume $s = |s|$.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Current step estimate
Now we compute our current-step eigenvalue count estimate $s = \frac{t}{m}$ where $t = \text{trace}\left(Y^T \widetilde{Z}\right)$.
total_trace = trace(Y' * Z);
count_estimate = total_trace / num_rand_vectors;
But s turns out to be a complex number... So we assume $s = |s|$.

Here's somewhere where we deviated from the paper. We strongly believe it's a typo, but they said to increment

$$t = t + \text{trace}(Y^T \widetilde{Z})$$

in the algorithm description, but then in the body of the paper they wouldn't...

Current step estimate

Now we compute our current-step eigenvalue count estimate $s = \frac{t}{m}$ where $t = \text{trace}\left(Y^T \widetilde{Z}\right)$.

```
total_trace = trace(Y' * Z);
count_estimate = total_trace / num_rand_vectors;
```

But s turns out to be a complex number... So we assume $s = |s|$.

The paper states $t = t + \text{trace}\left(Y^T \widetilde{Z}\right)$. We think this is a typo.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Current step estimate
Now we compute our current-step eigenvalue count estimate $s = \frac{t}{m}$ where $t = \text{trace}\left(Y^T \widetilde{Z}\right)$.
total_trace = trace(Y' * Z);
count_estimate = total_trace / num_rand_vectors;
But s turns out to be a complex number... So we assume $s = |s|$.
The paper states $t = t + \text{trace}\left(Y^T \widetilde{Z}\right)$. We think this is a typo.

Here's somewhere where we deviated from the paper. We strongly believe it's a typo, but they said to increment

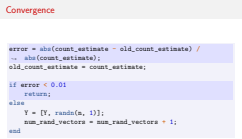
$$t = t + \text{trace}(Y^T \widetilde{Z})$$

in the algorithm description, but then in the body of the paper they wouldn't...

Convergence

```
error = abs(count_estimate - old_count_estimate) /
↪  abs(count_estimate);
old_count_estimate = count_estimate;

if error < 0.01
    return;
else
    Y = [Y, randn(n, 1)];
    num_rand_vectors = num_rand_vectors + 1;
end
```

The paper was very hand-wavy about convergence. All they said continue until s repeated several times or grew to be much larger than k. We had problems getting s to "repeat", but it did appear to converge to reasonable values.

So, what we do is say we're done if the relative error between iterations dips below 0.01. This isn't the best, but the calculations are slow enough that this was all I was patient enough for.

Outline

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Outline
Introduction
Overview
The Theory
Implementation
The Other Puzzle Pieces
Results

We've finished Stage 1

Our `FastEig` eigensolver will operate in two stages.

Stage 1 Search the complex plane for regions that are "dense" with eigenvalues

Stage 2 Run a modified form of the FEAST algorithm on each region found in Stage 1

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

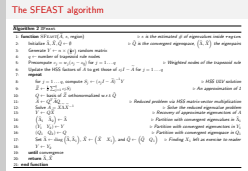
So this is all we had time for. We kept running into enough issues and hitting our heads against the table getting the eigencounter algorithm to run that we never had a chance to look at any of the other pieces.

Our goals started off pretty lofty, and were ultimately whittled down to where we're at now.

The SFEAST algorithm

Algorithm 2 SFeast

function SFEAST($\widetilde{A}$, s, region) ▷ *s is the estimated # of eigenvalues inside `region`*
  Initialize $\widehat{\Lambda}, \widehat{X}, \widehat{Q} \leftarrow \emptyset$ ▷ *$\widehat{Q}$ is the convergent eigenspace, $\left(\widehat{\Lambda}, \widehat{X}\right)$ the eigenpairs*
  Generate $Y \leftarrow n \times \left(\frac{2}{3}s\right)$ random matrix
  $q \leftarrow$ number of trapezoid rule nodes
  Precompute $c_j = w_j(z_j - z_0)$ for $j = 1 \dots q$ ▷ *Weighted nodes of the trapezoid rule*
  Update the HSS factors of $\widetilde{A}$ to get those of $z_j I - \widetilde{A}$ for $j = 1 \dots q$
  repeat
    for $j = 1 \dots q$, compute $S_j \leftarrow (z_j I - \widetilde{A})^{-1} Y$ ▷ *HSS ULV solution*
    $\widetilde{Z} \leftarrow \frac{1}{2}\sum_{j=1}^{q} c_j S_j$ ▷ *An approximation of 1*
    $Q \leftarrow$ basis of $\widetilde{Z}$ orthonormalized w.r.t $\widehat{Q}$
    $\widehat{A} \leftarrow Q^T \widetilde{A} Q$ ▷ *Reduced problem via HSS matrix-vector multiplication*
    Solve $\widehat{A} = \widetilde{X}\widetilde{\Lambda}\widetilde{X}^{-1}$ ▷ *Solve the reduced eigenvalue problem*
    $Y \leftarrow Q\widetilde{X}$ ▷ *Recovery of approximate eigenvectors of A*
    $\begin{pmatrix}\widehat{\Lambda}_1 & \widehat{\Lambda}_2\end{pmatrix} \leftarrow \widehat{\Lambda}$ ▷ *Partition with convergent eigenvalues in $\widehat{\Lambda}_1$*
    $(Y_1 \quad Y_2) \leftarrow Y$ ▷ *Partition with convergent eigenvectors in Y_1*
    $(Q_1 \quad Q_2) \leftarrow Q$ ▷ *Partition with convergent eigenspace in Q_1*
    Set $\widehat{\Lambda} \leftarrow \operatorname{diag}\left(\widehat{\Lambda}, \widehat{\Lambda}_1\right)$, $\widehat{X} \leftarrow \begin{pmatrix}\widehat{X} & X_1\end{pmatrix}$, and $\widehat{Q} \leftarrow \begin{pmatrix}\widehat{Q} & Q_1\end{pmatrix}$ ▷ *Finding X_1 left as exercise to reader*
    $Y \leftarrow Y_2$
  until convergence
  return $\widehat{\Lambda}, \widehat{X}$
end function



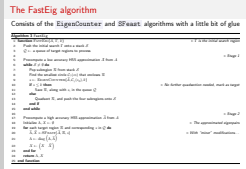
So we never looked at the other algorithms, but here they are for completeness. The SFEAST algorithm is what extends the restrictions of previous work. We run it on a region, along with its estimated eigenvalue count.

The FastEig algorithm

Consists of the `EigenCounter` and `SFeast` algorithms with a little bit of glue

Algorithm 3 `FastEig`

function FastEig(A, Γ, k) ▷ *Γ is the initial search region*
  Push the initial search Γ onto a stack $\mathcal{S}$
  $\mathcal{Q} \leftarrow$ a queue of target regions to process
  ▷ *Stage 1*
  Precompute a low accuracy HSS approximation $\widetilde{A}$ from A
  while $\mathcal{S} \neq \emptyset$ **do**
    Pop subregion $\mathcal{R}$ from stack $\mathcal{S}$
    Find the smallest circle $\mathcal{C}_\gamma(z_0)$ that encloses $\mathcal{R}$
    $s \leftarrow$ EigenCounter$(\widetilde{A}, \mathcal{C}_\gamma(z_0), k)$
    if $s \leq k$ **then** ▷ *No further quadsection needed, mark as target*
      Save $\mathcal{R}$, along with s, in the queue $\mathcal{Q}$
    else
      Quadsect $\mathcal{R}$, and push the four subregions onto $\mathcal{S}$
    end if
  end while
  ▷ *Stage 2*
  Precompute a high accuracy HSS approximation $\widetilde{A}$ from A
  Initialize $\Lambda, X \leftarrow \emptyset$ ▷ *The approximated eigenpairs*
  for each target region $\mathcal{R}$ and corresponding s in $\mathcal{Q}$ **do**
    $\widehat{\Lambda}, \widehat{X} \leftarrow$ SFeast$(\widetilde{A}, \mathcal{R}, s)$ ▷ *With "minor" modifications...*
    $\Lambda \leftarrow \operatorname{diag}\begin{pmatrix}\Lambda, \widehat{\Lambda}\end{pmatrix}$
    $X \leftarrow \begin{pmatrix}X & \widehat{X}\end{pmatrix}$
  end for
  return Λ, X
end function



Then we glue the eigencounter and the SFEAST algorithms together to get an eigensolver.

Outline

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Outline
Introduction
Overview
The Theory
Implementation
The Other Puzzle Pieces
Results

Numerical results

Definition

We define a **Cauchy-like** matrix A to be a matrix of the form

$$A_{ij} = \frac{u_i v_j}{s_i - t_j}$$

where $s_i = e^{\frac{2i\pi\mathbf{i}}{n}}$ and $t_j = e^{\frac{(2j+1)\pi\mathbf{i}}{n}}$ are on the unit circle, and $\{u_i\}_{i=1}^n$ and $\{v_j\}_{j=1}^n$ are random. Matrices of this form are known to be rank structured.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Numerical results
Definition
We define a **Cauchy-like** matrix A to be a matrix of the form
$A_{ij} = \frac{u_i v_j}{s_i - t_j}$
where $s_i = e^{\frac{2i\pi}{n}}$ and $t_j = e^{\frac{(2j+1)\pi}{n}}$ are on the unit circle, and $\{u_i\}_{i=1}^n$ and $\{v_j\}_{j=1}^n$ are random. Matrices of this form are known to be rank structured.

So what kind of results have we gotten? First, we ran our eigencounter only on a special kind of matrix because that's what the paper did.

Our implementation isn't the fastest, so we focused on relatively small matrices and a fixed region in the complex plane picked randomly.

Numerical results

Definition

We define a **Cauchy-like** matrix A to be a matrix of the form

$$A_{ij} = \frac{u_i v_j}{s_i - t_j}$$

where $s_i = e^{\frac{2i\pi\mathbf{i}}{n}}$ and $t_j = e^{\frac{(2j+1)\pi\mathbf{i}}{n}}$ are on the unit circle, and $\{u_i\}_{i=1}^n$ and $\{v_j\}_{j=1}^n$ are random. Matrices of this form are known to be rank structured.

Using 100×100 of these matrices on a region centered at $4 - 7\mathbf{i}$ with radius 3 gave an average of 60% accuracy on a 20 matrix batch.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Numerical results
Definition
We define a **Cauchy-like** matrix A to be a matrix of the form
$A_{ij} = \frac{u_i v_j}{s_i - t_j}$
where $s_i = e^{\frac{2i\pi}{n}}$ and $t_j = e^{\frac{(2j+1)\pi}{n}}$ are on the unit circle, and $\{u_i\}_{i=1}^n$ and $\{v_j\}_{j=1}^n$ are random. Matrices of this form are known to be rank structured.
Using 100×100 of these matrices on a region centered at $4 - 7\mathbf{i}$ with radius 3 gave an average of 60% accuracy on a 20 matrix batch.

So what kind of results have we gotten? First, we ran our eigencounter only on a special kind of matrix because that's what the paper did.

Our implementation isn't the fastest, so we focused on relatively small matrices and a fixed region in the complex plane picked randomly.

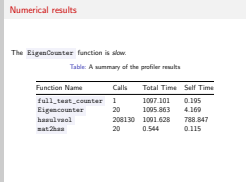
Numerical results

The `EigenCounter` function is *slow*.

Table: A summary of the profiler results

Function Name	Calls	Total Time	Self Time
`full_test_counter`	1	1097.101	0.195
`Eigencounter`	20	1095.863	4.169
`hssulvsol`	208130	1091.628	788.847
`mat2hss`	20	0.544	0.115

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY



As I mentioned earlier, our eigencounter is really slow, especially with matrices this size. That was pretty disappointing to me, so I made the mistake of profile our code.

It turns out that nearly all of the time taken is in solving the matrix-matrix systems using the HSS library that the authors provided.

After removing all HSS approximations we got a speedup two orders of magnitude faster. I think this is for two reasons

1. The HSS library isn't the most efficient or best written.
2. We only get savings on really large problems where the complexity benefits outweigh the performance hits.

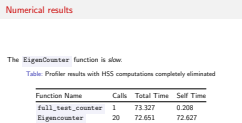
Numerical results

The `EigenCounter` function is *slow*.

Table: Profiler results with HSS computations completely eliminated

Function Name	Calls	Total Time	Self Time
`full_test_counter`	1	73.327	0.208
`Eigencounter`	20	72.651	72.627

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY



As I mentioned earlier, our eigencounter is really slow, especially with matrices this size. That was pretty disappointing to me, so I made the mistake of profile our code.

It turns out that nearly all of the time taken is in solving the matrix-matrix systems using the HSS library that the authors provided.

After removing all HSS approximations we got a speedup two orders of magnitude faster. I think this is for two reasons

1. The HSS library isn't the most efficient or best written.
2. We only get savings on really large problems where the complexity benefits outweigh the performance hits.

What did I learn?

- How to make 

in LaTeX

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

What did I learn?
- How to make clouds in LaTeX

So at the end, what did I learn? Ultimately, I had fun. It was extremely frustrating, but very rewarding.

I learned that eigenvalues are cool, and that they're a really deep subject.

What did I learn?

- How to make 

in LaTeX
- Eigenvalues are cool!

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

What did I learn?
- How to make clouds in LaTeX
- Eigenvalues are cool!

So at the end, what did I learn? Ultimately, I had fun. It was extremely frustrating, but very rewarding.

I learned that eigenvalues are cool, and that they're a really deep subject.

What did I learn?

- How to make 

in LaTeX
- Eigenvalues are cool!
- Finding eigenvalues is harder than it looks, even when you have a roadmap

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

What did I learn?
- How to make clouds in LaTeX
- Eigenvalues are cool!
- Finding eigenvalues is harder than it looks, even when you have a roadmap

So at the end, what did I learn? Ultimately, I had fun. It was extremely frustrating, but very rewarding.

I learned that eigenvalues are cool, and that they're a really deep subject.

What did I learn?

- How to make 

in LaTeX
- Eigenvalues are cool!
- Finding eigenvalues is harder than it looks, even when you have a roadmap
- Research involves a lot of confusion

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

What did I learn?
- How to make clouds in LaTeX
- Eigenvalues are cool!
- Finding eigenvalues is harder than it looks, even when you have a roadmap
- Research involves a lot of confusion

So at the end, what did I learn? Ultimately, I had fun. It was extremely frustrating, but very rewarding.

I learned that eigenvalues are cool, and that they're a really deep subject.

What did I learn?

- How to make 

in LaTeX
- Eigenvalues are cool!
- Finding eigenvalues is harder than it looks, even when you have a roadmap
- Research involves a lot of confusion
- Optimizations are meaningless until you profile your code.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

What did I learn?
- How to make clouds in LaTeX
- Eigenvalues are cool!
- Finding eigenvalues is harder than it looks, even when you have a roadmap
- Research involves a lot of confusion
- Optimizations are meaningless until you profile your code.

So at the end, what did I learn? Ultimately, I had fun. It was extremely frustrating, but very rewarding.

I learned that eigenvalues are cool, and that they're a really deep subject.

The Future. . .

There's a lot that could be explored further. (Eigenvalues are a deep topic!)

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

The Future. . .
There's a lot that could be explored further. (Eigenvalues are a deep topic!)

There's a lot that could be done in the future. I'm hoping to turn what I've done so far into my CSR masters research topic, but am having a hard time picking exactly what I want to do.

First, we hand waved our way around the spectral projector business. Then there are these HSS creatures that I don't understand very well. I don't understand how they're implemented, or really what benefit they have.

The Future. . .

There's a lot that could be explored further. (Eigenvalues are a deep topic!)

- Explore the spectral projector $\mathbf{\Phi}$ business from a more theoretical perspective

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

The Future. . .
There's a lot that could be explored further. (Eigenvalues are a deep topic!)
- Explore the spectral projector $\mathbf{\Phi}$ business from a more theoretical perspective

There's a lot that could be done in the future. I'm hoping to turn what I've done so far into my CSR masters research topic, but am having a hard time picking exactly what I want to do.

First, we hand waved our way around the spectral projector business. Then there are these HSS creatures that I don't understand very well. I don't understand how they're implemented, or really what benefit they have.

The Future. . .

There's a lot that could be explored further. (Eigenvalues are a deep topic!)

- Explore the spectral projector $\mathbf{\Phi}$ business from a more theoretical perspective
- Explore HSS approximations. (how, why, what?)

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

The Future. . .
There's a lot that could be explored further. (Eigenvalues are a deep topic!)
- Explore the spectral projector $\mathbf{\Phi}$ business from a more theoretical perspective
- Explore HSS approximations. (how, why, what?)

There's a lot that could be done in the future. I'm hoping to turn what I've done so far into my CSR masters research topic, but am having a hard time picking exactly what I want to do.

First, we hand waved our way around the spectral projector business. Then there are these HSS creatures that I don't understand very well. I don't understand how they're implemented, or really what benefit they have.

The Future. . .

There's a lot that could be explored further. (Eigenvalues are a deep topic!)

- Explore the spectral projector $\mathbf{\Phi}$ business from a more theoretical perspective
- Explore HSS approximations. (how, why, what?)
- Finish the `FastEig` implementation

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

The Future. . .
There's a lot that could be explored further. (Eigenvalues are a deep topic!)
- Explore the spectral projector $\mathbf{\Phi}$ business from a more theoretical perspective
- Explore HSS approximations. (how, why, what?)
- Finish the `FastEig` implementation

There's a lot that could be done in the future. I'm hoping to turn what I've done so far into my CSR masters research topic, but am having a hard time picking exactly what I want to do.

First, we hand waved our way around the spectral projector business. Then there are these HSS creatures that I don't understand very well. I don't understand how they're implemented, or really what benefit they have.

The Future. . .

There's a lot that could be explored further. (Eigenvalues are a deep topic!)

- Explore the spectral projector $\mathbf{\Phi}$ business from a more theoretical perspective
- Explore HSS approximations. (how, why, what?)
- Finish the `FastEig` implementation
- Improve the `FastEig` implementation (opportunities for parallelism abound)

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

The Future. . .
There's a lot that could be explored further. (Eigenvalues are a deep topic!)
- Explore the spectral projector $\mathbf{\Phi}$ business from a more theoretical perspective
- Explore HSS approximations. (how, why, what?)
- Finish the `FastEig` implementation
- Improve the `FastEig` implementation (opportunities for parallelism abound)

There's a lot that could be done in the future. I'm hoping to turn what I've done so far into my CSR masters research topic, but am having a hard time picking exactly what I want to do.

First, we hand waved our way around the spectral projector business. Then there are these HSS creatures that I don't understand very well. I don't understand how they're implemented, or really what benefit they have.

References I

L. Trefethen and D. Bau, *Numerical Linear Algebra*.

Society for Industrial and Applied Mathematics, 1997.

T. Sakurai and H. Sugiura, "A projection method for generalized eigenvalue problems," *Journal of Computational and Applied Mathematics*, vol. 159, no. 1, pp. 119–128, 2003.

X. Ye, J. Xia, R. H. Chan, S. Cauley, and V. Balakrishnan, "A fast contour-integral eigensolver for non-hermitian matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 38, no. 4, pp. 1268–1297, 2017.

Y. Xi and J. Xia, "On the stability of some hierarchical rank structured matrix algorithms," *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 3, pp. 1279–1303, 2016.

References I

L. Trefethen and D. Bau, *Numerical Linear Algebra*.
Society for Industrial and Applied Mathematics, 1997.

T. Sakurai and H. Sugiura, "A projection method for generalized eigenvalue problems," *Journal of Computational and Applied Mathematics*, vol. 159, no. 1, pp. 119–128, 2003.

X. Ye, J. Xia, R. H. Chan, S. Cauley, and V. Balakrishnan, "A fast contour-integral eigensolver for non-hermitian matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 38, no. 4, pp. 1268–1297, 2017.

Y. Xi and J. Xia, "On the stability of some hierarchical rank structured matrix algorithms," *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 3, pp. 1279–1303, 2016.

References II

J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, "A hierarchically semiseparable (hss) matrix package." `https://www.math.purdue.edu/~xiaj/work/hss.zip`, 2010.

All code, LaTeX source, and relevant papers can be found at `https://research.agill.xyz/`.

References II

J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, "A hierarchically semiseparable (hss) matrix package." `https://www.math.purdue.edu/~xiaj/work/hss.zip`, 2010.

All code, LaTeX source, and relevant papers can be found at `https://research.agill.xyz/`.