Ch 9.3 part 2: variants of the power method

- Accelerating convergence
- Finding more than just the dominant eigenvalue

Friday, November 21, 2025 10:40 AM

Accelerating Convergence

- Book mentions Aitken's Δ^2 method (2025, we didn't cover)

which I've never seen used in practice in this situation, as it only helps w/ eigenvalues not eigenvectors. So we won't discuss.

Shifts

Suppose $A \in \mathbb{R}^{3\times 3}$ has eigenvalues $\{100, 99, 97\}$

then $B = A - qI$ has eigenvalues $\{100-q, 99-q, 97-q\}$

and same eigenvectors as A

Power method on A converges linearly w/ rate $\frac{99}{100}$. Slow!

Choose $q = 97$ so B has eigenvalues $\{3, 2, 0\}$

Power method on B (which is as easy as power method on A)

has rate $\frac{2}{3}$. Much better!

But be careful! you'd think $q = 99.1$ is even better!

eigs $\{0.9, 0.1, -2.1\}$

ratio $\frac{0.1}{0.9}$ is great! But... power method now converges to the -2.1 eigenvalue since $|-2.1| > |0.9|$, so you're getting a different eigenpair as before.

So if $\lambda_1 > \lambda_2 > \dots > \lambda_n$ you can use shifts to make $|\lambda_1 - q| > |\lambda_n - q|$

or $|\lambda_1 - q| < |\lambda_n - q|$

(but won't help us find "interior" eigenvalues)

... so slightly helpful, i.e. for a few use cases

Saturday, November 29, 2025 11:12 AM

Inversion

If A has eigs $\{\lambda_1, \dots, \lambda_n\}$ w/ eigenvectors $\{\vec{v}_1, \dots, \vec{v}_n\}$

then $B = A^{-1}$ has eigs $\{\frac{1}{\lambda_1}, \dots, \frac{1}{\lambda_n}\}$ w/ same eigenvectors

So if λ's are $\{4, 1, -3\}$

then A^{-1} has eigenvalues $\{\frac{1}{4}, 1, -\frac{1}{3}\}$ ↑ now dominant!

and power method on A^{-1} converges to this interior eigenvalue.

Obvious disadvantage: must calculate $A^{-1}\cdot\vec{x}$ in power method.

You'd do a LU factorization in $O(n^3)$, save it, then $A^{-1}\vec{x}$ costs $O(n^2)$ per step.

So overall cost is $O(n^3)$ + #iterations $\cdot O(n^2)$

(vs normal power method: cost is #iterations $\cdot O(n^2)$ or less if sparse ...)

Shift-and-invert

Combining the above ideas, let $B = (A - qI)^{-1}$

then it has same eigenvectors, and eigenvalues $\frac{1}{\lambda_1 - q}, \dots, \frac{1}{\lambda_n - q}$

Choose q to target any eigenvalue and to accelerate convergence

Ex: $A \in \mathbb{R}^{3\times 3}$ has eigenvalues $\{10, 9, 3\}$

Use case 1: we run power method for 2 iterations and crudely estimate $\lambda_1 = 10.1$ Set $q = 10.1$

$B = (A - 10.1\cdot I)^{-1}$ has eigenvalues $\left\{\frac{1}{10-10.1}, \frac{1}{9-10.1}, \frac{1}{3-10.1}\right\}$

i.e. $\left\{\frac{-1}{-1} = -10, \frac{1}{-1.1} \approx -1, \frac{1}{-7.3}\right\}$

power method on B converges to λ_1 w/ rate $\frac{1}{10}$. Super fast!

Saturday, November 29, 2025 11:23 AM

Use case 2 (still $\{10, 9, 3\}$)

We want to find λ_2 eigenpair. Crudely estimate $\lambda_2 \approx 9.1$
(e.g. use a deflation strategy we'll discuss next)

then set $q = 9.1$, $B = (A - 9.1 \cdot I)^{-1}$

which has eigs $\left\{ \frac{1}{.9}, \frac{1}{-.1}, \frac{1}{-6.1} \right\}$

$\approx \{ 1.1, -10, -.18 \}$

↑ that's dominant (i.e. has largest abs. value)

So power method on B converges to λ_2 (at rate $\approx \frac{1.1}{10}$ good!!)

Adaptive shift-and-invert: Rayleigh Quotient Iteration

Idea: allow q to change every iteration using your best, most recent guess for the target eigenvalue!

guess is from Rayleigh quotient

In some cases, this works very well and you have cubic convergence!

↖ ridiculously fast

Make sure to adjust it for previous offsets

Downside: If $B_1 = A - q_1 \cdot I$ and you have its LU factorization, that doesn't help you get the LU factorization of $B_2 = A - q_2 I$ (if $q_2 \neq q_1$)

So you need an $O(n^3)$ LU factorization per step.

(if $q_1 \approx q_2$, $B_1^{-1} \approx B_2^{-1}$ as formalized by Neumann Series but that is an approximation, not exact, so it's (once you truncate) complicated/messy)

Saturday, November 29, 2025 11:34 AM

Finding more than 1 eigenpair

Suppose you found an eigenpair λ_1, w/ eigenvector $\vec{v}_1$, and now you want to find more.

Idea 1:

Redo power method but starting at $\vec{w}$ which is orthogonal to $\vec{v}_1$

Fails in practice due to roundoff error

Idea 2: "annihilation"

Use $B = A - \lambda_1 \cdot I$ (a shift) A little better. Run pwr method on B

You could find another eigenvalue λ_2 this way,

then $C = (A - \lambda_1 \cdot I)(A - \lambda_2 I)$ Now it's not as nice

Idea 3: "deflation"

use $B = A - \lambda_1 \cdot \vec{v}_1 \cdot \vec{w}_1^T$ for any vector $\vec{w}_1$ s.t. $\vec{v}_1^T \cdot \vec{w}_1 = 1$

Ex: Hotelling deflation uses the "obvious" choice $\vec{w}_1 = \frac{1}{\|\vec{v}_1\|^2} \cdot \vec{v}_1$

Wielandt deflation uses $\vec{w}_1$ from a scaled row of A which lets you treat B as a $(n-1) \times (n-1)$ matrix. Sounds nice in theory since you could continue the process and keep getting smaller.

then do power method on B

Ideas 2 & 3 kind of work...

to find a few dominant eigenvalues.

But if you want all eigenvalues (for a large matrix)

they're too numerically unstable.

Idea 4: Simultaneous iteration aka Simultaneous Orthogonal iteration

Recall: Simple normalization

Initialize at $\vec{x}_0$ w/ unit norm

For $k = 1, 2, 3, \ldots$

$$\vec{y}_k = A \cdot \vec{x}_{k-1}$$

$$\mu_{k-1} = \vec{x}_{k-1}^T \cdot \vec{y}_k$$

$$\vec{x}_k = \vec{y}_k / \|\vec{y}_k\|_2$$

Saturday, November 29, 2025 11:48 AM

Instead of $\vec{x}_k \in \mathbb{R}^{n\times 1}$ let's do $X \in \mathbb{R}^{n\times r}$ and try to find top r eigenpairs simultaneously:

for $k = 1, 2, 3, \ldots$

$$Y_k = A \cdot X_{k-1} \in \mathbb{R}^{n\times r}$$

(Before, $\vec{x}_k = \vec{y}_k / \|\vec{y}_k\|$, we normalized. Now, let's orthonormalize)

Compute QR factorization, $Y_k = Q_k R_k$

$X_k = Q_k$

In fact, we could rewrite with Q's not x's, look for all $r = n$ eigenvalues, and initialize $Q_1 = I$:

so $Y_1 = A$

$Y_1 = A$

for $k = 1, 2, \ldots$

$Q_k R_k = Y_k$

$Y_{k+1} = A \cdot Q_k$

If we define A_1 and A_2

$A_1 = Y_1 = A$

$A_2 = Q_1^* \cdot Y_2 = Q_1^* \cdot A_1 \cdot Q_1 = Q_1^* A \cdot Q_1$

$Y_2 = A \cdot Q_1$

$Y_3 = A \cdot Q_2$

then since $Q_2 R_2 = Y_2$

it holds that $\underbrace{Q_1^* Q_2}_{\tilde{Q}_2} R_2 = A_2$ a qr decomp!

try

$$A_3 = \tilde{Q}_2^* A_2 \tilde{Q}_2$$

$$= \tilde{Q}_2^* \cdot Q_1^* A Q_1 \tilde{Q}_2$$

$$= Q_2^* \cancel{Q_1 Q_1^*} A \cancel{Q_1 Q_1^*} Q_2 = Q_2^* \cdot A \cdot Q_2$$

$$= Q_2^* \cdot Y_3$$

in general:

$A_{k+1} = Q_k^* Y_{k+1}$

Saturday, November 29, 2025 12:25 PM

but the advantage is that we have the recurrence

$A_1 = A$

for $k = 1, 2, \ldots$

$Q_k R_k = A_k$ "Q_k" is now really "$\tilde{Q}_k$"

$A_{k+1} = Q_k^* A_k Q_k$

$= Q_k R_k$ ... so $A_{k+1} = Q_k^* Q_k R_k Q_k$

giving us the

QR Algorithm

$A_1 = A$

for $k = 1, 2, \ldots$

$Q_k R_k = A_k$ // qr factorization, $O(n^3)$

$A_{k+1} = R_k Q_k$

Same as orthogonal iteration if you ignore floating pt. error and do $r = n$ and $Q_1 = I$.

This is stable but costly, $O(n^3)$ per iteration.

Next time we'll see how to make it cheaper,

both for the QR

and in case we want to do shift-and-invert,

by preprocessing to Upper Hessenberg.

SOURCES: our book isn't great.

Mostly based on Eduardo Corona and Stephen Becker's own notes, with some wikipedia, and some Golub and van Loan (definitive resource)