Eigenproblems

Linear Algebra 2

Eigenvalue Problems

- The eigenvalues $\lambda^{(i)}$ and eigenvectors $\underline{u}^{(i)}$ of a matrix A are given by solutions of

$$(A - \lambda I)\underline{u} = 0$$

- Many examples in Physics :
  - Atomic spectra
  - Coupled oscillators
  - Radios
  - Musical instruments

- And many numerical methods for finding solutions…

Finding Eigenvalues/Eigenvectors

- A method or finding eigenvalues that you will have met is to expand the determinant :

$$p(t) = \det(A - tI)$$

- For an $n \times n$ matrix this results in a n^{th}-degree polynomial known as the characteristic polynomial
  - The roots of this polynomial are the eigenvalues
- In fact, solving eigenvalue problems is intrinsically related to polynomial root finding
  - It can be shown that for n>4, rational solutions do not generally exist
  - Algorithms to find eigenvalues (or roots of polynomials) are generally *iterative*

Power Iteration

- This is a very simple iterative method for finding eigenvectors
  - Algorithm starts from an initial estimate/guess
  - Output is closer to the true eigenvector
  - Iteration can then provide the result to required precision

- The method is described by the recurrence relation :

$$v_{i+1} = \frac{Av_i}{|Av_i|}$$

  - ie. at each iteration, v_i is multiplied by A and normalised

- *Used by Google for page ranking, and Twitter for follow recommendations…*

Power Iteration

- The *Eigenproblems notebook* includes an example
- Attempt to find the largest eigenvector $\hat{\underline{u}}^{(0)}$ of matrix A

$$A = \begin{pmatrix} -2 & -4 & 2 \\ -2 & 1 & 2 \\ 4 & 2 & 5 \end{pmatrix} \qquad \hat{\underline{u}}^{(0)} = \begin{pmatrix} 0.058 \\ 0.351 \\ 0.935 \end{pmatrix}$$

- Starting from random guess for $\hat{\underline{u}}^{(0)}$
- You can see the algorithm converges on the eigenvector, although ~40 iterations required to get 3 decimal places

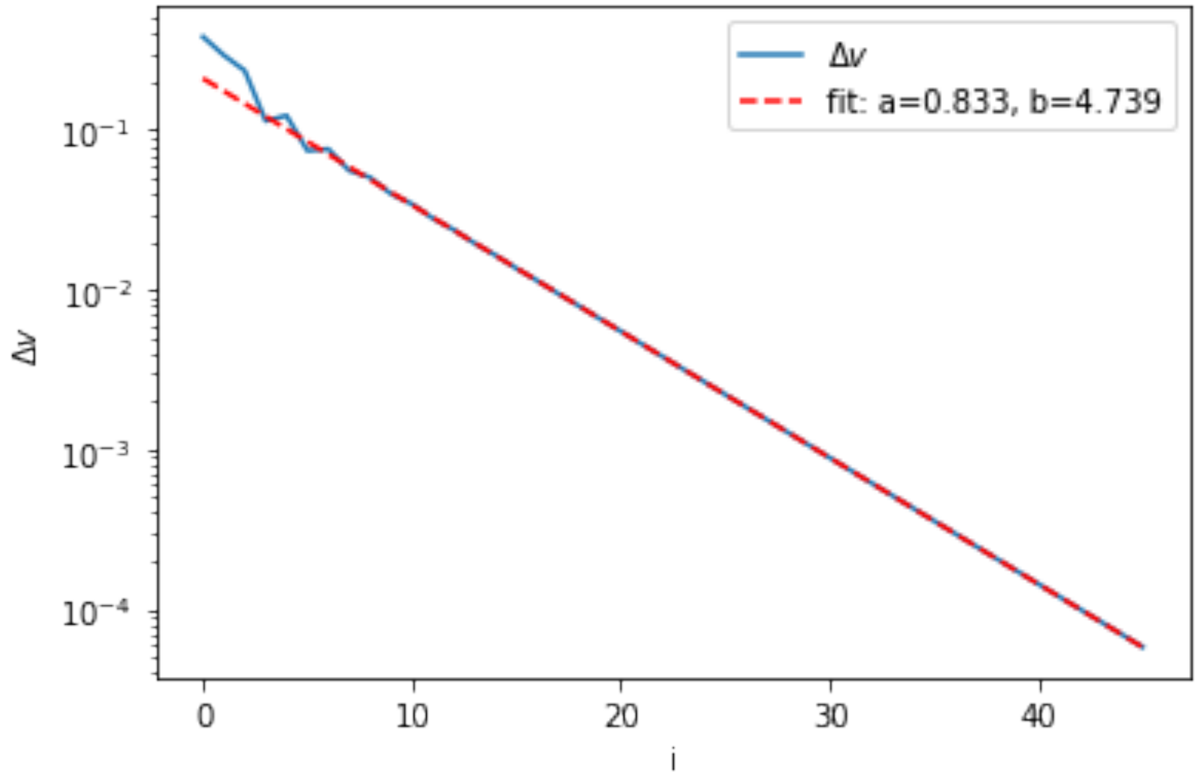
randomly generated guess

3 dp correct

u_0	u_1	u_2
[0.4359949	0.02592623	0.54966248]
[0.02715432	0.05562669	0.99808232]
[0.29446291	0.3420972	0.89233463]
[-0.02651875	0.23621606	0.97133863]
[0.18192338	0.38639215	0.90421513]
[-0.01604317	0.29088117	0.95662467]
[0.13465607	0.38515724	0.91297406]
[0.00258742	0.31387726	0.94946004]
[0.10900307	0.37703536	0.91976229]
[0.05852015	0.35058088	0.93470233]
[0.05833769	0.35047613	0.93475301]
[0.05848974	0.35056342	0.93471078]
[0.05836303	0.35049068	0.93474598]]

Power Iteration

- The notebook also explores the numerical error in the method
  - Plot shows max error wrt true eigenvector
  - Clear relationship : $\Delta v \sim 0.83^i$
- When using iterative methods, we often talk about the *convergence rate*
  - How fast does the algorithm converge on the result
  - *Not to be confused with algorithmic complexity !*



Power Iteration - Eigenvalues

- The notebook also includes a method to obtain an estimate of the eigenvalue at each step - by computing the Rayleigh Quotient :

$$\mu_i = \frac{v_i^* A v_i}{v_i^* v_i}$$

- Note that, in the Power Iteration method, this does not impact the next iteration
  - ie. v_{i+1} does not depend on μ_i
- A similar analysis of the eigenvalues shows they converge in the same way as the eigenvectors

Rayleigh Quotient Iteration

- An improved version of Power Iteration uses the Rayleigh Quotient in each iteration

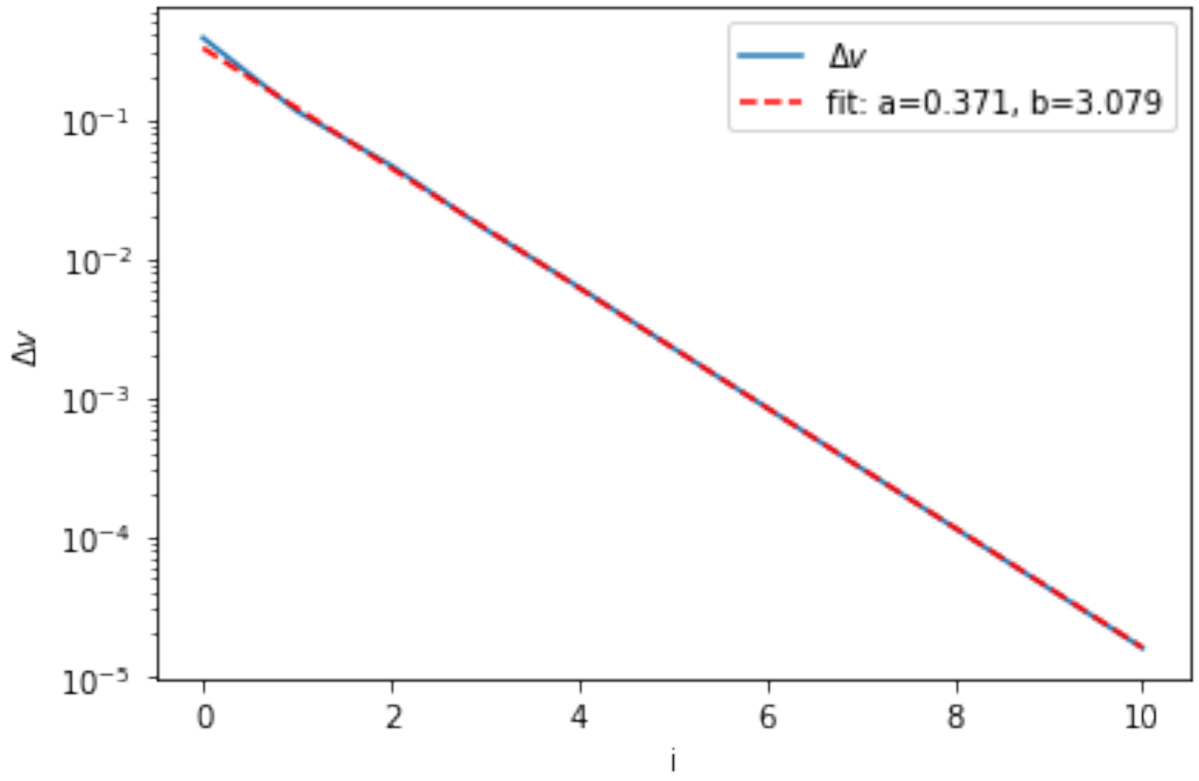
- The recurrence relation :

power iteration $v_{i+1} = \frac{Av_i}{|Av_i|}$ becomes $v_{i+1} = \frac{(A - \mu_i I)^{-1} v_i}{|(A - \mu_i I)^{-1} v_i|}$ Rayleigh quotient iteration

- Instead of repeatedly applying the matrix A on each iteration, we apply the full eigenvalue equation

Rayleigh Quotient Iteration

- A key point to note here, is that the Rayleigh quotient iteration converges much faster
  - Instead of $\Delta v \sim 0.83^i$
  - We now have $\Delta v \sim 0.37^i$
- However, Rayleigh quotient iteration is a lot more sensitive to the initial estimate
  - It will converge on *AN* eigenpair
  - But not necessarily the largest…



Deflation

- Iteration methods mentioned above only find one eigenvalue
  - Usually the largest - but dependent on the initial eigenvector guess!
  - They can be extended to find all eigenpairs

- Deflation is a technique to construct a new matrix which has the same eigenpairs as the original, with one removed (ie. set to zero)

- The method is then :
  1. Find an eigenpair
  2. “Deflate” the matrix to remove the known eigenpair
  3. Repeat until all eigenpairs are found

Hotelling Deflation

- Perhaps the most simple deflation method
- The deflation step to “remove” an eigenpair is :

$$A' = A - \mu^{(i)} \underline{v}^{(i)} \underline{v}^{*(i)}$$

- Note that this method has some limitations
  - Only works for symmetric matrices
  - Introduces an error in remaining eigenvalues, related to the ratio $\lambda^{(0)}/\lambda^{(1)}$
  - If this ratio is large, the method can become unstable
- Example given in the *Eigenproblem notebook*, with application to a system of coupled oscillators

Summary

- Eigenproblems are a class of problem that occur widely in Physics
  - Closely related to root-finding problems
- Numerical methods are generally iterative, and find one eigenpair at a time, eg :
  - Power iteration
  - Rayleigh quotient iteration
- We can extent these techniques to find all eigenpairs using deflation
  - Hotelling's deflation is a simple, albeit limited, method
- **Next** - study the Eigenproblems notebook which goes through this material in more detail, with examples