Overview of slides 09

Recap of methods for Boundary Value Problems

- Direct matrix method: Convert boundary value problem to sparse matrix equation
  - direct elimination, or
  - conjugate gradient-type algorithm
- Relaxation: convert static boundary value problem to initial value problem
  - start with some guess
  - solve diffusion equation in computer time
  - find stationary (late-time) solution

These slides:

- Spectral method: Transform PDE/ODE to algebraic equation via Fourier transform

Why fourier transform?

1. Physics
   - Fourier transform takes you from time to frequency,
     or from space to wave number
     — waves/oscillations are naturally formulated in 'fourier space'
   - Quantum mechanics relates energy and frequency [$E = \hbar\omega$]
     and momentum and wave number [$\lambda = h/p \Longleftrightarrow p = \hbar k$]
     $\rightarrow$ FT takes you from space/time to momentum/energy representation
2. Mathematics
   - Differential equations (boundary value problems) are easier to solve

   Poisson equation $$\nabla^2 \varphi(x) = \rho(x) \quad \Longrightarrow \quad k^2 \Phi(k) = \tilde{\rho}(k)$$
3. Computing, data analysis
   - Fourier methods are used to filter noise
   - and for pattern recognition, trend finding

The Fourier transform

Fourier series

For a function defined on the domain $x \in [-\frac{L}{2}, \frac{L}{2}]$ with $f(-\frac{L}{2}) = f(\frac{L}{2})$,

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{2\pi i n x/L} \quad \text{with} \quad c_n = \frac{1}{L}\int_{-L/2}^{L/2} e^{-2\pi i n x/L} f(x) dx$$

Fourier transformation

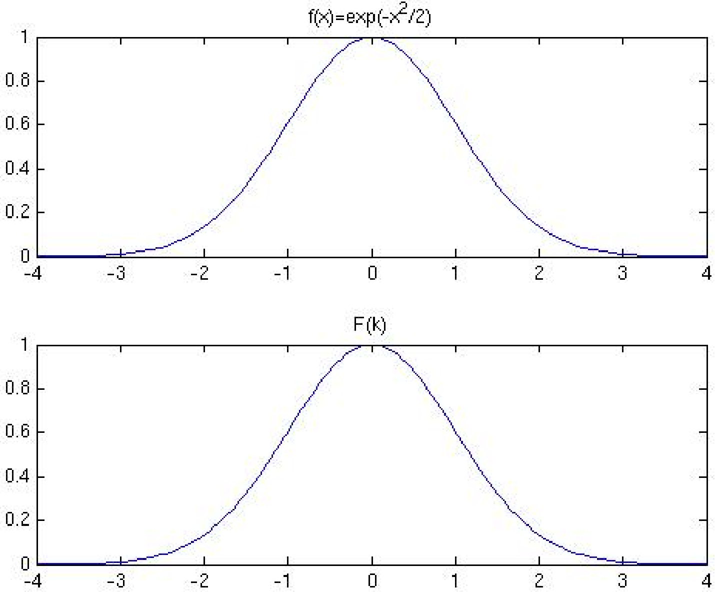
The Fourier series can be written as

$$c_n = \frac{1}{L}\hat{f}_n \equiv \frac{1}{L}\hat{f}(\frac{2\pi n}{L}) \quad \text{with} \quad \hat{f}(k) = \int_{-L/2}^{L/2} e^{-ikx} f(x) dx\,.$$

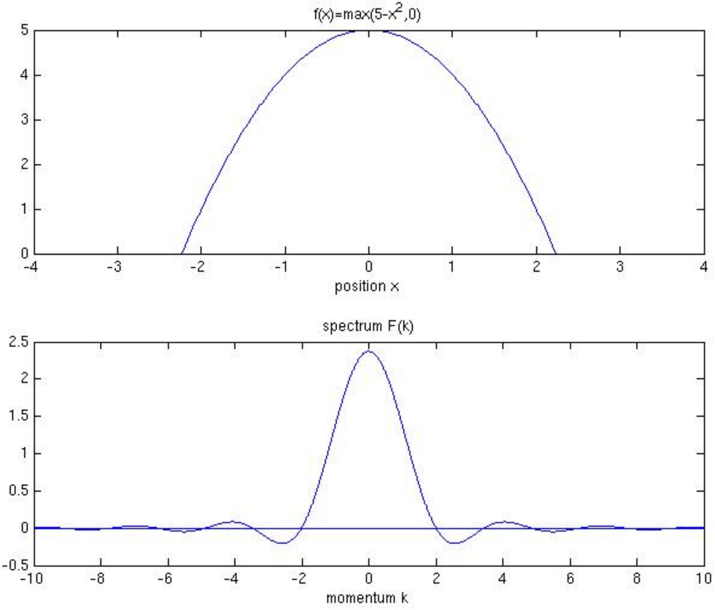
Taking $L \to \infty$ we get

$$f(x) = \sum_{n=-\infty}^{\infty} \frac{1}{L}\hat{f}(k_n) e^{ik_n x} = \frac{1}{2\pi}\sum_{n=-\infty}^{\infty} \hat{f}(k_n)\frac{2\pi}{L} \longrightarrow \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{ikx}\hat{f}(k) dk$$

Example: Gaussian

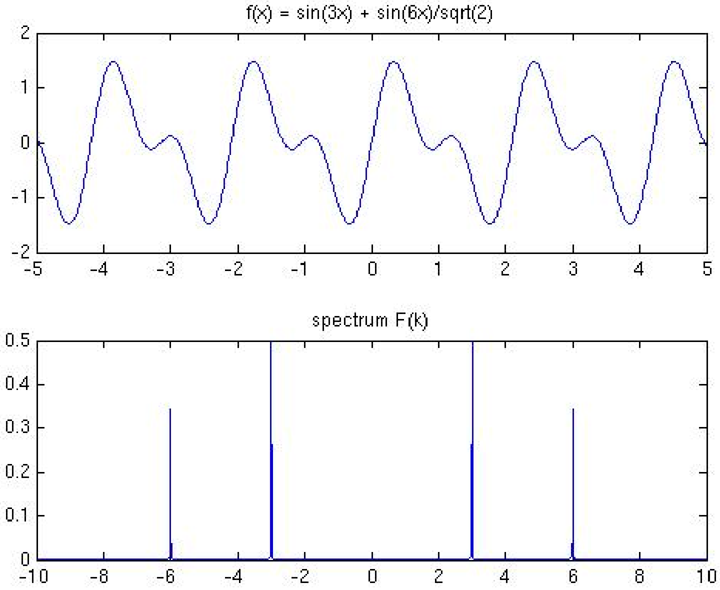


Example 2

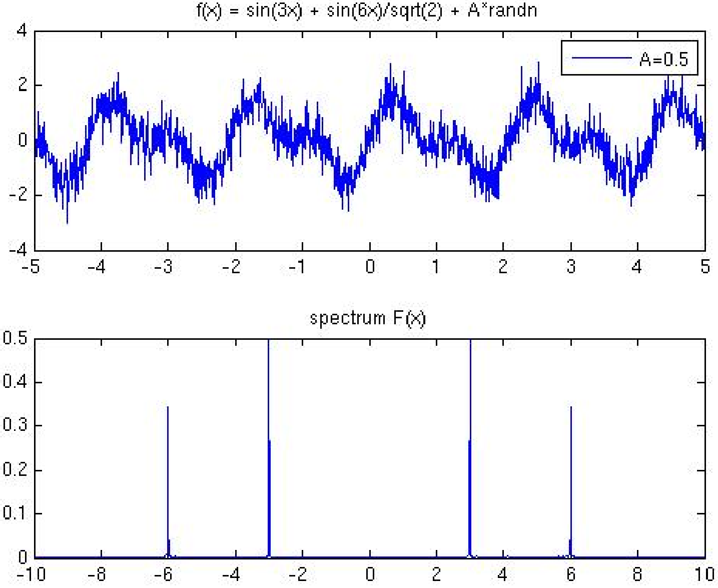


f(x)=max(5-x^2,0)
5
4
3
2
1
0
-4
-3
-2
-1
0
1
2
3
4
position x
spectrum F(k)
2.5
2
1.5
1
0.5
0
-0.5
-10
-8
-6
-4
-2
0
2
4
6
8
10
momentum k

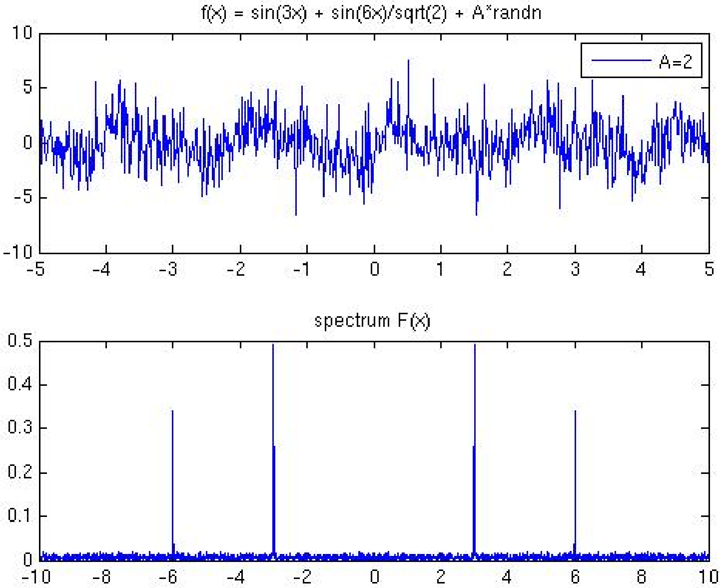
Example: harmonics



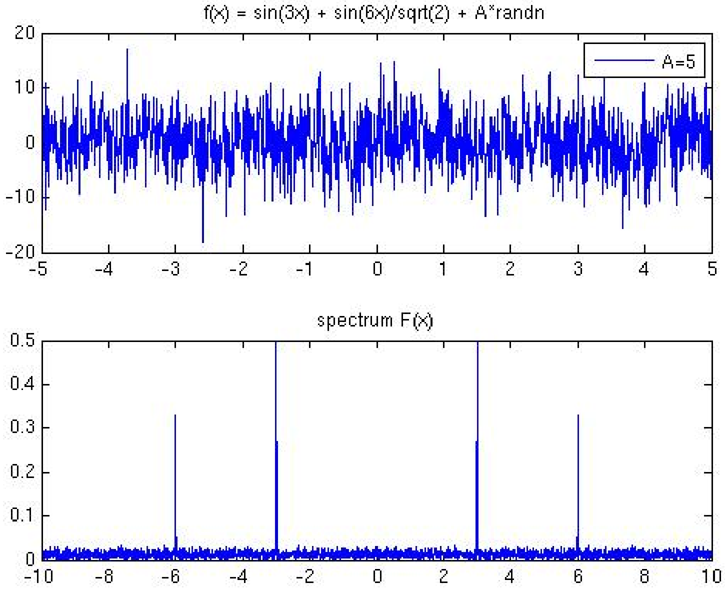
Add some noise...



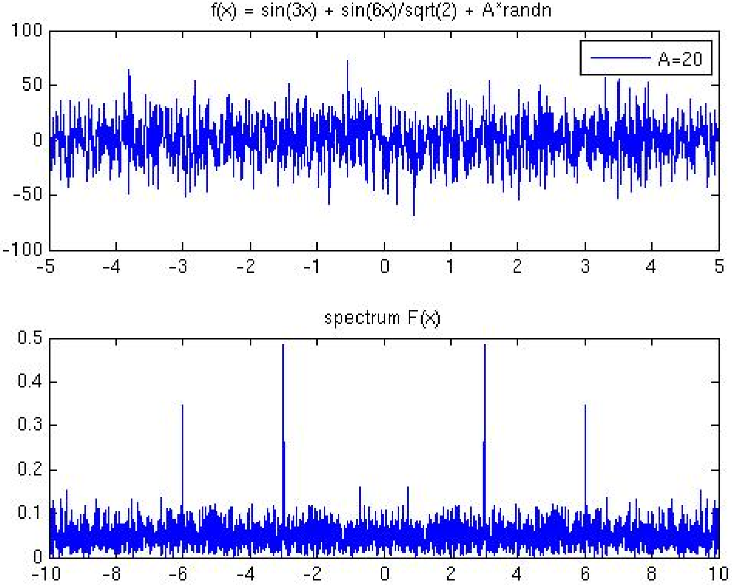
Add more noise



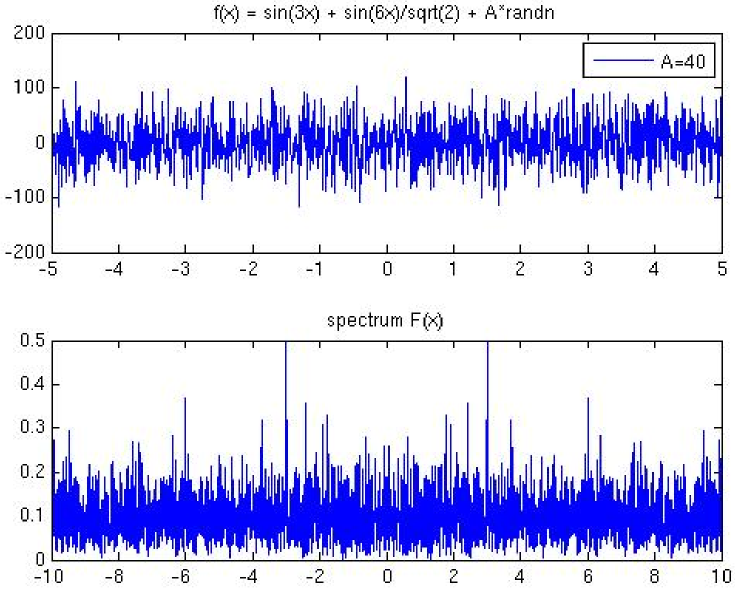
And more noise...



More noise...


f(x) = sin(3x) + sin(6x)/sqrt(2) + A*randn
A=20
100
50
0
-50
-100
-5
-4
-3
-2
-1
0
1
2
3
4
5
spectrum F(x)
0.5
0.4
0.3
0.2
0.1
0
-10
-8
-6
-4
-2
0
2
4
6
8
10

Even more noise!



Properties

If $f(x) \longleftrightarrow F(k)$ (i.e., if $F(k)$ is the F.T. of $f(x)$), then

$$
\begin{aligned}
f(ax) &\quad\longleftrightarrow\quad \frac{1}{|a|}F\left(\frac{k}{a}\right) \\
f(x+a) &\quad\longleftrightarrow\quad e^{ika}F(k) \\
\frac{df}{dx} &\quad\longleftrightarrow\quad -ikF(k) \\
f * g \equiv \int_{-\infty}^{\infty} f(\xi)g(x-\xi)d\xi &\quad\longleftrightarrow\quad F(k)G(k) \quad \text{[convolution]} \\
\int_{-\infty}^{\infty} f(\xi+x)g(\xi)d\xi &\quad\longleftrightarrow\quad F(k)G(-k) \quad \text{[correlation]}
\end{aligned}
$$

$$
\begin{aligned}
&f \text{ is real} &&\Longrightarrow\quad F(-k) = F^*(k) \\
&f \text{ is imaginary} &&\Longrightarrow\quad F(-k) = -F^*(k) \\
&f \text{ is even} &&\Longrightarrow\quad F(-k) = F(k) \\
&f \text{ is odd} &&\Longrightarrow\quad F(-k) = -F(k)
\end{aligned}
$$

Differential equations converted to algebraic

The Fourier transform of a derivative

Assume we know $\hat{f}(k)$, the Fourier transform of $f(x)$.
What is the Fourier transform of $f'(x)$?

$$\int_{-\infty}^{\infty} e^{-ikx} f'(x) dx = \left[e^{-ikx} f(x) \right]_{-\infty}^{\infty} - \int_{-\infty}^{\infty} (-ik) e^{-ikx} f(x) dx = ik\hat{f}(k)$$

Differential equations

$$Af''(x) + Bf'(x) + Cf(x) = g(x)$$

Take the Fourier transform of this:

$$-k^2 A\hat{f}(k) + ikB\hat{f}(k) + C\hat{f}(k) = \hat{g}(k) \quad \Longrightarrow \quad \hat{f}(k) = \frac{\hat{g}(k)}{-k^2 A + ikB + C}$$

The differential equation becomes an algebraic equation!

Many dimensions

The D-dimensional Fourier transform is simply

$$\hat{f}(\vec{k}) = \int e^{-i\vec{k}\cdot\vec{x}} f(\vec{x}) d^D x \,, \qquad f(\vec{x}) = \frac{1}{(2\pi)^D} \int e^{i\vec{k}\cdot\vec{x}} \hat{f}(\vec{k}) d^D k$$

The Fourier transform of the partial derivative is

$$\widehat{\partial_i f}(\vec{k}) = i k_i \hat{f}(\vec{k}) \quad \Longrightarrow \quad \widehat{\nabla^2 f}(\vec{k}) = -k^2 \hat{f}(\vec{k})$$

Any partial differential equation becomes an algebraic equation!

Example

The Poisson equation, $\nabla^2 \Phi(\vec{x}) = \rho(\vec{x})$, becomes

$$-k^2 \hat{\Phi}(\vec{k}) = \hat{\rho}(\vec{k}) \quad \Longleftrightarrow \quad \hat{\Phi}(\vec{k}) = -\frac{\hat{\rho}(\vec{k})}{k^2}$$

But so what?

We can solve any linear PDE with constant coefficients in Fourier (wave number/frequency) space. Great.

How do we transform real space to/from Fourier space?

Fourier integrals can be costly.

Discrete Fourier transform

1. Finite volume $\longrightarrow$ back to Fourier series, discrete $k_n = \frac{2\pi n}{L}$
2. Discrete set of points, spacing $a \longrightarrow k_{\max} = \frac{2\pi}{a}$ (minimum wavelength $\lambda_{\min} = a$)

$$\hat{f}_n \equiv \sum_{j=0}^{N-1} e^{-2\pi ijn/N} f_j \,, \qquad f_j = \frac{1}{N} \sum_{n=0}^{N-1} e^{2\pi ijn/N} f_n$$

Slow fourier transform

Naive implementation

The discrete fourier transform is

$$F_k = \sum_{j=0}^{N-1} e^{-2\pi ijk/N} f_j$$

You need to compute all the phases $e^{-2\pi ijk/N}$ $\forall j, k$

This is really really slow!

If you still want to kill yourself:

Do not compute all of them separately:

1. There are only N independent phases
2. Compute $z \equiv e^{-2\pi i/N} = \cos\frac{2\pi}{N} + i\sin\frac{2\pi}{N} = a + ib$
3. Compute the other phases by multiplication:
   $e^{-4\pi i/N} = z^2 = (a+ib)(a+ib) = a^2 - b^2 + 2iab$
   $e^{-6\pi i/N} = z^3 = z \cdot z^2$ etc.

Fast Fourier Transform

[Gauss (1805), Danielson & Lanczos (1942), Cooley & Tukey (1965)]

Danielson–Lanczos Lemma

If $N = 2n$ the discrete fourier transform is the sum of an even and an odd half-transform

$$
\begin{aligned}
F_k &= \sum_{j=0}^{N-1} e^{-2\pi ijk/N} f_j \\
&= \sum_{j=0}^{N/2-1} e^{-2\pi i(2j)k/N} f_{2j} + \sum_{j=0}^{N/2-1} e^{-2\pi i(2j+1)k/N} f_{2j+1} \\
&= \sum_{j=0}^{N/2-1} e^{-2\pi ijk/(N/2)} f_{2j} + z^k \sum_{j=0}^{N/2-1} e^{-2\pi ijk/(N/2)} f_{2j+1} \\
&= F_k^e + z^k F_k^o
\end{aligned}
$$

Fast Fourier Transform

The Danielson–Lanczos Lemma can be applied recursively!

$N = 2^m$

1. Best case: minimize calculation of phase factors.

More general case

We can generalise the Danielson–Lanczos Lemma for factors of 3, 5, etc.
$\longrightarrow$ more computation for higher prime factors

If N is prime the FFT becomes the slow fourier transform!

Main lesson

Use $N = 2^m$ if you can.
Otherwise, make sure the prime factors of N are as small as possible

Python implementation: numpy.fft package

`F = fft(f)`	Perform the fast fourier transform on f
`f = ifft(F)`	Perform the inverse fourier transform
`g = fftshift(f)`	Shift the data cyclically by $N/2$
`F = fftn(f)`	Perform n-dimensional FFT
`f = ifftn(F)`	Perform inverse n-dimensional FFT

FFT window vs normal window

We often display both $f(x)$ and $F(k)$ symmetrically about zero:

$$x \in [-\frac{L}{2}, \frac{L}{2}], \qquad k \in [-\frac{\pi}{a}, \frac{\pi}{a}]$$

The FFT takes $x \in [0, L\rangle$ as input and returns $k \in [0, \frac{2\pi}{a}\rangle$

Use `fftshift` to move between the two representations!

The discretised Poisson equation

How can we use the discrete Fourier transform to solve a PDE?
Take the Poisson equation, now on a discrete $M \times N$ grid:

$$\nabla^2 \Phi(x, y) \longrightarrow \frac{\Phi_{j+1,k} + \Phi_{j-1,k} + \Phi_{j,k-1} + \Phi_{j,k+1} - 4\Phi_{j,k}}{a^2} = \rho_{jk}$$

Taking the discrete Fourier transform of this we get

$$\frac{e^{2\pi im/M} + e^{-2\pi im/M} + e^{2\pi in/M} + e^{-2\pi in/M} - 4}{a^2} \hat{\Phi}_{mn} = \hat{\rho}_{mn}$$

$$\Longrightarrow \quad -\frac{4}{a^2}\left(\sin^2 \frac{\pi m}{M} + \sin^2 \frac{\pi n}{N} \right) \hat{\Phi}_{mn} = \hat{\rho}_{mn}$$

Solution

$$\hat{\Phi}_{mn} = \hat{\Phi}(\hat{k}_x, \hat{k}_y) = -\frac{\hat{\rho}_{mn}}{\hat{k}_x^2 + \hat{k}_y^2}\,, \qquad \hat{k}_i = \frac{2}{a} \sin \frac{\pi n_i}{N_i}\,.$$

Numerical implementation

Poisson equation with periodic boundary conditions

1. Shift the source ρ so that x, y both start at 0
2. Fourier transform ρ: `Rho = fftn(rho,2)`
3. Calculate the discrete wave numbers $\hat{k}^2$
4. Find $\hat{\Phi} = -\hat{\rho}/\hat{k}^2$
5. Fourier transform back: `phi = ifftn(Phi,2)`
6. Shift the solution back to the original window

What about Dirichlet or Neumann boundary conditions?

Dirichlet boundary conditions

Use the discrete sine transform:

$$\hat{\Phi}_{mn} = \sum_{jk} \sin\frac{\pi jm}{M} \sin\frac{\pi kn}{N} \Phi_{jk}\,, \quad \Phi_{jk} = \frac{2}{MN} \sum_{mn} \sin\frac{\pi jm}{M} \sin\frac{\pi kn}{N} \hat{\Phi}_{mn}$$

This is zero on the boundaries. Boundary values can be moved to the rhs.

Summary: Spectral Methods

- Linear PDEs with constant coefficients on regular grids can be solved using Fourier (spectral) methods
- Differential equations become algebraic equations in Fourier space
- Fast Fourier Transform: an efficient method for discrete fourier transforms.
- Method applies naturally to periodic boundary conditions, but can be extended to Dirichlet or Neumann using the discrete sine or cosine transform.

Done with Fourier methods

Next: Eigenvalue problems

Given a square matrix A, find all/some of its eigenvalues

... and maybe the corresponding eigenvectors.

Once eigenvalues are obtained, calculating corresponding eigenvectors is some extra numerical work. (We limit discussion to eigenvalues.)

Eigenvalue algorithms

Full vs sparse

Broadly, two classes of algorithms:

1. Not huge sizes; matrix stored in full format
   $\rightarrow$ natural to perform full diagonalization
   - obtain all eigenvalues
   - On desktop, applicable for sizes $\lesssim 10^4$
2. Larger sizes: matrix stored in sparse format, or not at all
   - Algorithms to obtain some eigenvalues
   - Extremal eigenvalues easiest, internal eigenvalues harder

Eigenvalue algorithms

Full diagonalization

- Algorithm: iterative QR decomposition
- Define $A^{(0)} = A$.

$$\begin{aligned} A^{(k-1)} &= Q_k R_k \qquad &\text{(QR decomp of } A^{(k-1)}\text{)} \\ A^{(k)} &= R_k Q_k \qquad &\text{(inverting order: next matrix defined)} \end{aligned}$$

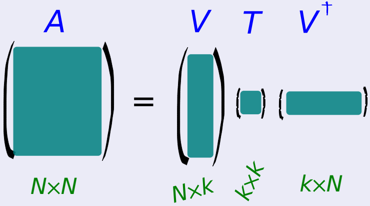
  until $A^{(k)}$ is sufficiently diagonal.

- QR decomposition done by Gram-Schmidt, Givens rotations, or Householder transformations.
- Modern variants include many refinements to basic idea.

Eigenvalue algorithms

Sparse matrices

- Algorithms based on repeated application of A to vectors.
- Simplest: power method (rather primitive)
- Better: Rayleigh quotient iteration
- More sophisticated: Krylov subspace methods



Lanczos algorithm: diagonalize the T matrix.