Outline

• Maxwell eigenproblem
• Matrix decompositions
• Reformulating the eigenproblems
• Iterative eigensolvers
  – Conjugate gradient solvers
  – Preconditioners
  – Davidson solvers
  – Targeted solvers
Maxwell Eigenproblem

- Maxwell’s equations in QM notation:

\[ \vec{\nabla} \times \frac{1}{\varepsilon} \vec{\nabla} \times |H\rangle = -\frac{1}{c^2} \frac{\partial^2}{\partial t^2} |H\rangle \]

\[ \vec{\nabla} \cdot |H\rangle = 0. \]

- Choose:

\[ |H\rangle = e^{i(\vec{k} \cdot \vec{x} - \omega t)} |H_{\vec{k}}\rangle \]

- Such that:

\[ \hat{A}_{\vec{k}} |H_{\vec{k}}\rangle = (\omega/c)^2 |H_{\vec{k}}\rangle \]

\[ \hat{A}_{\vec{k}} = \left( \vec{\nabla} + i\vec{k} \right) \times \frac{1}{\varepsilon} \left( \vec{\nabla} + i\vec{k} \right) \times \]
Maxwell Eigenproblem

- Choose basis such that:
  \[ \langle H^{(n)}_k | H^{(m)}_k \rangle = \delta_{n,m} \]

- Generally can expand H-field in the basis:
  \[ |H_k^n\rangle \equiv \sum_{m=1}^{N} h_m |b_m\rangle. \]

- Which yields:
  \[ Ah = \left( \frac{\omega}{c} \right)^2 Bh \]
  \[ A_{\ell m} = \langle b_\ell | \hat{A}_k | b_m \rangle \text{ and } B_{\ell m} = \langle b_\ell | b_m \rangle \]

\( h \) is a column vector.
Maxwell Eigenproblem

• Best algorithm to use?
  
  – Power method: \( \lambda_1 = \frac{v_k^T A v_k}{v_k^T v_k} \)
  
  – Inverse iteration: \( \lambda = b_{k-1} (A - \tau 1)^{-1T} \cdot A \cdot (A - \tau 1)^{-1} b_{k-1} \)
  
  – Transformation method: \( A' = PAP \)
  
  – Decomposition: \( A_{k+1} = L_k Q_k = Q_k^T A_k Q_k \)
LU Decomposition

Rewrite input matrix $A$ as a product of lower-triangular and upper-triangular matrices, i.e.,

$$
\begin{bmatrix}
\alpha_{11} & 0 & 0 & 0 \\
\alpha_{21} & \alpha_{22} & 0 & 0 \\
\alpha_{31} & \alpha_{32} & \alpha_{33} & 0 \\
\alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44}
\end{bmatrix}
\cdot
\begin{bmatrix}
\beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} \\
0 & \beta_{22} & \beta_{23} & \beta_{24} \\
0 & 0 & \beta_{33} & \beta_{34} \\
0 & 0 & 0 & \beta_{44}
\end{bmatrix}
= 
\begin{bmatrix}
a_{11} & a_{12} & a_{13} & a_{14} \\
a_{21} & a_{22} & a_{23} & a_{24} \\
a_{31} & a_{32} & a_{33} & a_{34} \\
a_{41} & a_{42} & a_{43} & a_{44}
\end{bmatrix}
$$
LU Decomposition

To construct the LU matrices, use Crout’s algorithm to compute the decomposition in place:

\[ \beta_{ij} = a_{ij} - \sum_{k=1}^{i-1} \alpha_{ik} \beta_{kj} \]

\[ \alpha_{ij} = \frac{1}{\beta_{jj}} \left( a_{ij} - \sum_{k=1}^{j-1} \alpha_{ik} \beta_{kj} \right) \]
Band Diagonal Matrices

• Band diagonal means only k diagonals are non-zero

• Common special case: tridiagonal:

$$
\begin{bmatrix}
  b_1 & c_1 & 0 & \cdots \\
  a_2 & b_2 & c_2 & \cdots \\
  & \ddots & \ddots & \ddots \\
  & & a_{N-1} & b_{N-1} & c_{N-1} \\
  & & & \ddots & \ddots & \ddots \\
  & & & & a_N & b_N \\
  & & & & & 0 & a_{N} \\
\end{bmatrix}
\cdot
\begin{bmatrix}
  u_1 \\
  u_2 \\
  \vdots \\
  u_{N-1} \\
  u_N
\end{bmatrix}
=
\begin{bmatrix}
  r_1 \\
  r_2 \\
  \vdots \\
  r_{N-1} \\
  r_N
\end{bmatrix}
$$

• Can perform LU decomposition in kN operations
Singular Value Decomposition

• Based on theorem: any matrix $A = U \cdot W \cdot V^T$, where:

$$
\begin{pmatrix}
A \\
\end{pmatrix}
= 
\begin{pmatrix}
U \\
\end{pmatrix} \cdot 
\begin{pmatrix}
w_1 \\
w_2 \\
\vdots \\
w_N \\
\end{pmatrix} \cdot 
\begin{pmatrix}
V^T \\
\end{pmatrix}
$$

• $U$ and $V$ are both orthogonal: $U^T U = 1$; $V^T V = 1$
• Inversion is easy: $A^{-1} = V \cdot \frac{1}{W} \cdot U^T$
• Condition number is set by max $w_j$/min $w_j$; difficult to work with large values
Sparse Linear Systems

Band diagonal
Block triangular
Block tridiagonal
Single-bordered block diagonal
Double-bordered block diagonal
Single-bordered block triangular
Bordered band-triangular
Single-bordered band diagonal
Double-bordered band diagonal
Other
Cholesky Decomposition

• Can be thought of as taking the square root of a matrix $A$, such that $A = LL^T$

• Writing out explicitly yields following equations:

\[
L_{ii} = \left( a_{ii} - \sum_{k=1}^{i-1} L_{ik}^2 \right)^{1/2}
\]

\[
L_{ji} = \frac{1}{L_{ii}} \left( a_{ij} - \sum_{k=1}^{i-1} L_{ik}L_{jk} \right)
\]
QR Decomposition

• Write $A = Q \cdot R$, where $Q^T Q = 1$, and $R$ is upper triangular
• Can then solve $R \cdot x = Q^T b$
• Matrix from a series of Householder transformation, s.t. $Q = \prod_i Q_i$
  – each $Q_i = 1 - 2w \cdot w^T$
  – Choose vector $w$ to eliminate off-diagonal entries in one row + one column
Decomposition Timing

• Execution time:
  – total number of elements computed/stored: $N^2$
  – Total number of operations per element (average): $N/3$
  – Scaling: $O(N^3)$
Alternative approaches

- May want to focus on first $p$ bands, where $p \ll N$
- Now storage scales like $O(pN)$
- Work scales like $O(p^2N)$
Reformulating the Eigenproblem

• Choose planewave basis such that:

\[ |b_{m_1,m_2,m_3}\rangle = e^{i \sum_j m_j \vec{G}_j \cdot \vec{x}} \]

• Now we can rewrite

\[ |H_{ \hat{k}}\rangle \equiv \sum_{\{m_j\}} \tilde{h}_{\{m_j\}} e^{i \sum_{j,k} m_j \vec{G}_j \cdot \vec{n}_k \vec{R}_k / N_k} = \sum_{\{m_j\}} \tilde{h}_{\{m_j\}} e^{2\pi i \sum_j m_j n_j / N_j} \]

• This is in fact a discrete Fourier transform!
Reformulating the Eigenproblem

- While a normal DFT scales like $O(N^2)$, but
- A fast Fourier transform (FFT) scales like $O(N \log N)$
- A succession of FFT and inverse FFT’s, i.e.:

$$A_{\ell m} = -\left( \vec{k} + \vec{G}_{\ell} \right) \times \cdots \text{IFFT} \cdots \vec{e}^{-1} \cdots \text{FFT} \cdots \left( \vec{k} + \vec{G}_{m} \right) \times$$

...would have the same scaling
Representing Epsilon

• Discontinuities in F-series representing epsilon pose a risk for convergence
• Could address these through averaging motivated by Maxwell’s equations:

\[
\bar{\varepsilon}^{-1} = \overline{\varepsilon}^{-1}P + \overline{\varepsilon}^{-1}(1 - P)
\]

\[
P_{ij} = n_i n_j
\]
Iterative Eigensolvers

• Two classes of iterative eigensolvers to find lowest $p$ bands:
  – Conjugate-gradient minimization of Block Rayleigh quotient
  – Davidson method (based on Lanczos’ method)
Iterative Eigensolvers

- Rayleigh quotient defined as the minimum value of:
  \[ \lambda_0 = \min \frac{y_0^\dagger A y_0}{y_0^\dagger B y_0} \]

- Rayleigh-Ritz algorithm can solve first value, then choose a new \( y_1 \) orthogonal to \( y_0 \), and minimize again
Iterative Eigensolvers

- Block Rayleigh quotient minimization: Choose $Y$ to an $n \times p$ matrix, then minimize: $tr \left[ Y^\dagger AY \right]$
  - Where $Y^\dagger BY = I$

- For an unconstrained minimization, transform to $Z$ via: $Y = Z \left( Z^\dagger BZ \right)^{-1/2}$
  - Then minimize: $tr \left[ Z^\dagger AZU \right]$
  - Where: $U = \left( Z^\dagger BZ \right)^{-1}$
Iterative Eigensolvers

- Steepest descent method: use gradient $r_k$ to find direction of minimization
- Conjugate-gradient method: combine gradient with memory of prior steps via

$$p_k = r_k - \sum_{i<k} \frac{p_i^T A r_k}{p_i^T A p_i} p_i$$

$$x_{k+1} = x_k + \alpha_k p_k \quad \alpha_k = \frac{p_k^T b}{p_k^T A p_k} = \frac{p_k^T (r_{k-1} + A x_{k-1})}{p_k^T A p_k} = \frac{p_k^T r_{k-1}}{p_k^T A p_k}$$

- Best suited for convex problems: quadratic or near-quadratic in nature
Convexity

- Convex functions have certain properties that aid in finding an optimum:
  - Precisely one optimum in an open set of values
  - Continuous and at least twice differentiable
  - Midpoints always lower than edges — i.e.,
    \[ f[\delta x_1 + (1 - \delta) \delta x_2] < \delta f(x_1) + (1 - \delta)f(x_2) \]
- Examples include \(x^2\), sinh(x)
Iterative Eigensolvers

• Minimization direction:

\[ G = P_\perp A Z U \]
\[ P_\perp = 1 - B Z U Z^\dagger \]
\[ D = \hat{K} G + \gamma D_0. \]

\[
\gamma = \frac{\text{tr} \left[ G^\dagger \hat{K} G \right]}{\text{tr} \left[ G_0^\dagger \hat{K} G_0 \right]} \\
\gamma = \frac{\text{tr} \left[ (G - G_0)^\dagger \hat{K} G \right]}{\text{tr} \left[ G_0^\dagger \hat{K} G_0 \right]}
\]

Fletcher-Reeves  Polak-Ribiere

Pre-conditioning operator
Iterative Eigensolvers

Convergence rate for 3 variations of iterative eigensolvers to known solutions: conjugate gradient method clearly faster
Preconditioners

- Preconditioner matrix $\hat{K}$ given by:
  \[ \delta Z \approx \hat{K}G = \hat{A}^{-1}GU^{-1} \]

- Approximate inverse estimated either by:
  
  - Diagonal method:
    \[ \tilde{A}_{\ell m} = \left| \hat{\kappa} + \hat{G}_m \right|^2 \delta_{\ell,m} \]
  
  - Non-diagonal method:
    \[ \tilde{A} = \vec{\nabla} \times \hat{P}_T \frac{1}{\hat{\epsilon}} \hat{P}_T \vec{\nabla} \times \]
Iterative Eigensolvers

Benefit of two types of preconditioners: diagonal and non-diagonal
Davidson Eigensolver

• Build up subspace $V$ containing desired eigenvectors where: $V^\dagger BV = I$

• Find Ritz eigenvectors: $Y = \hat{V}Y_v$

• Compute residual: $R = BYL - AY$

• Add to $V$ using: $D = \hat{K}R$
Davidson Eigensolver

Relative performance of Davidson, versus conjugate-gradient methods
Targeted Eigensolvers

• To find a single defect state in a photonic crystal, use a supercell and shift the target of the eigensolver via:

\[ \hat{A}'_k = \left( \hat{A}_k - \frac{\omega_m^2}{c^2} \right)^2 \]

• This yields the same eigenvectors, but shifted eigenvalues—can easily return to original basis
Targeted Eigensolvers

Convergence of targeted eigensolver as the supercell grows still reasonable
Eigensolver Performance

![Graphs showing eigensolver performance](image-url)
Next Class

• Use MIT Photonic Bands
• Reference: MPB User Tutorial: