ECE 695
Numerical Simulations
Lecture 4: Eigenproblems for Electro-optic Systems

Prof. Peter Bermel

January 18, 2017

Outline

- Electrostatic Potentials
- Solving $Ax = b$
- Spin arrays
- Solving eigenproblems
- Bandstructure problem
- Bloch's theorem
- Photonic bandstructures
  - 1D
  - 2D

Electrostatic Potential

- Consider an array of charges governed by Gauss' law:

$$\nabla \cdot E = \rho/\epsilon_o$$

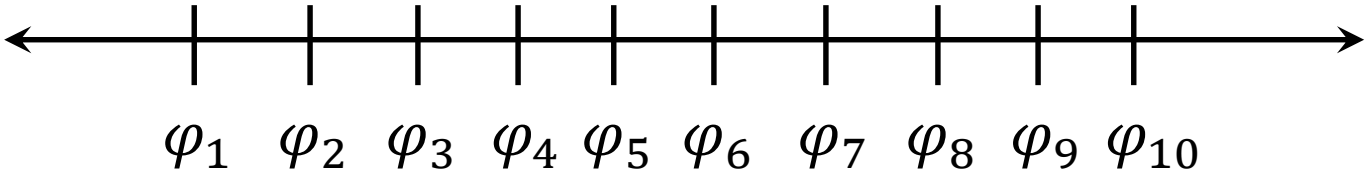
- Using the definition of potential yields Poisson's equation:

$$-\nabla^2 \varphi = \rho/\epsilon_o$$

- Consider solving this equation for an arbitrary set of charges – what to do?

Electrostatic Potential Example

- Strictly speaking, continuous fields have an uncountable number of possible values, and cannot be evaluated numerically
- Key is to transform from continuous field values to those on a grid
- Increase resolution as needed



Electrostatic Potential Solution: 1D

- Approximate Laplacian in 1D with:

$$\nabla^2\varphi \approx \frac{\varphi_{i-1} - 2\varphi_i + \varphi_{i+1}}{h^2}$$

- Where *h* is the grid spacing
- Sets up the linear algebra problem:

$$\begin{pmatrix} -2 & 1 & 0 & & & 0 & 0 & 0 \\ 1 & -2 & 1 & \cdots & & 0 & 0 & 0 \\ 0 & 1 & -2 & & & 0 & 0 & 0 \\ & \vdots & & \ddots & & & \vdots & \\ 0 & 0 & 0 & & -2 & 1 & 0 & \\ 0 & 0 & 0 & \cdots & 1 & -2 & 1 & \\ 0 & 0 & 0 & & 0 & 1 & -2 & \end{pmatrix} \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \vdots \\ \varphi_{N-2} \\ \varphi_{N-1} \\ \varphi_N \end{pmatrix} = \frac{h^2}{\epsilon_o} \begin{pmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ \vdots \\ \rho_{N-2} \\ \rho_{N-1} \\ \rho_N \end{pmatrix}$$

Electrostatic Potential Solution: 1D

- In MATLAB, use:

```
>> N=10; M=-2; J=1; A=diag(M*ones(N,1))+diag(J*ones(N-1,1),1)+diag(J*ones(N-1,1),-1)

A =

    -2     1     0     0     0     0     0     0     0     0
     1    -2     1     0     0     0     0     0     0     0
     0     1    -2     1     0     0     0     0     0     0
     0     0     1    -2     1     0     0     0     0     0
     0     0     0     1    -2     1     0     0     0     0
     0     0     0     0     1    -2     1     0     0     0
     0     0     0     0     0     1    -2     1     0     0
     0     0     0     0     0     0     1    -2     1     0
     0     0     0     0     0     0     0     1    -2     1
     0     0     0     0     0     0     0     0     1    -2

>> rho=rand(10,1)

rho =

    0.1190
    0.4984
    0.9597
    0.3404
    0.5853
    0.2238
    0.7513
    0.2551
    0.5060
    0.6991
```

How to Solve $Ax = b$?

- Direct matrix inversion: $x = A^{-1}b$
- Linear algebra software packages:
  - Linear Algebra Package (LAPACK)
  - MATLAB backslash operator

Linear Algebra Package (LAPACK)

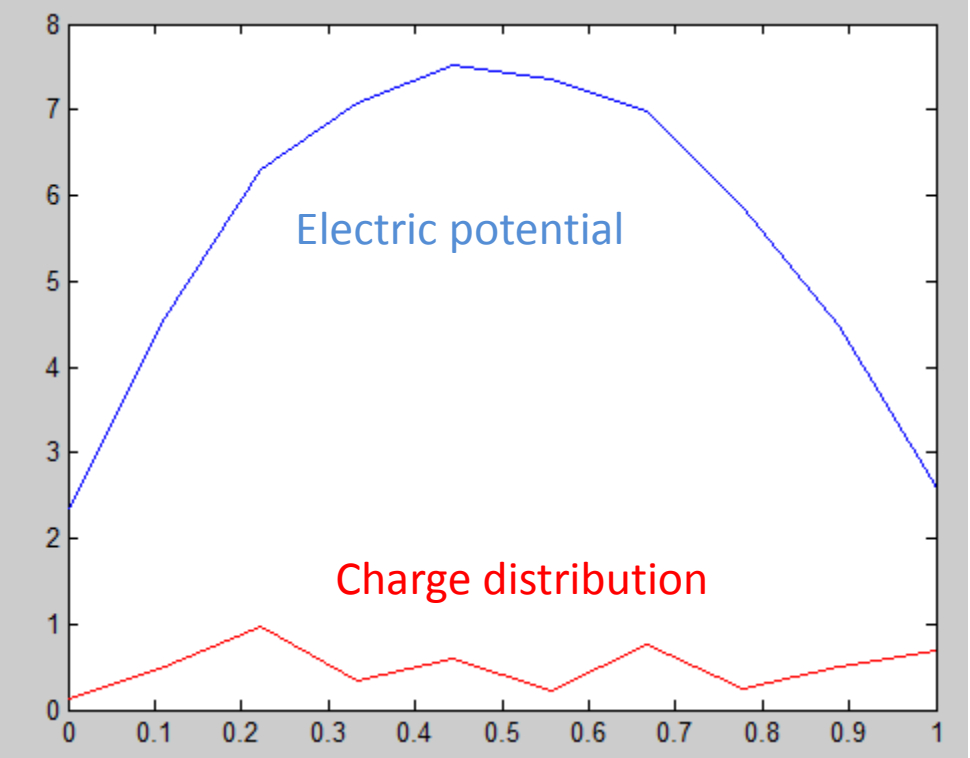
- Implements many linear algebra techniques:
  - Linear programming/least squares
  - Matrix decompositions/factorizations
  - Eigenvalues
- Designed in 1992 to deal with special cases efficiently

Matrix type	full	banded	packed	tridiag	generalized problem
general	ge	gb		gt	gg
symmetric	sy	sb	sp	st	
Hermitian	he	hb	hp		
SPD / HPD	po	pb	pp	pt	
triangular	tr	tb	tp		tg
upper Hessenberg	hs				hg
trapezoidal	tz				
orthogonal	or		op		
unitary	un		up		
diagonal	di				
bidiagonal	bd				

Electrostatic Potential Solution: 1D

- MATLAB: use backslash operator to solve problems of the form $A \cdot x = b$

```
>> phi=A\rho

phi =

   -2.3498
   -4.5806
   -6.3131
   -7.0858
   -7.5181
   -7.3651
   -6.9884
   -5.8604
   -4.4772
   -2.5882
```

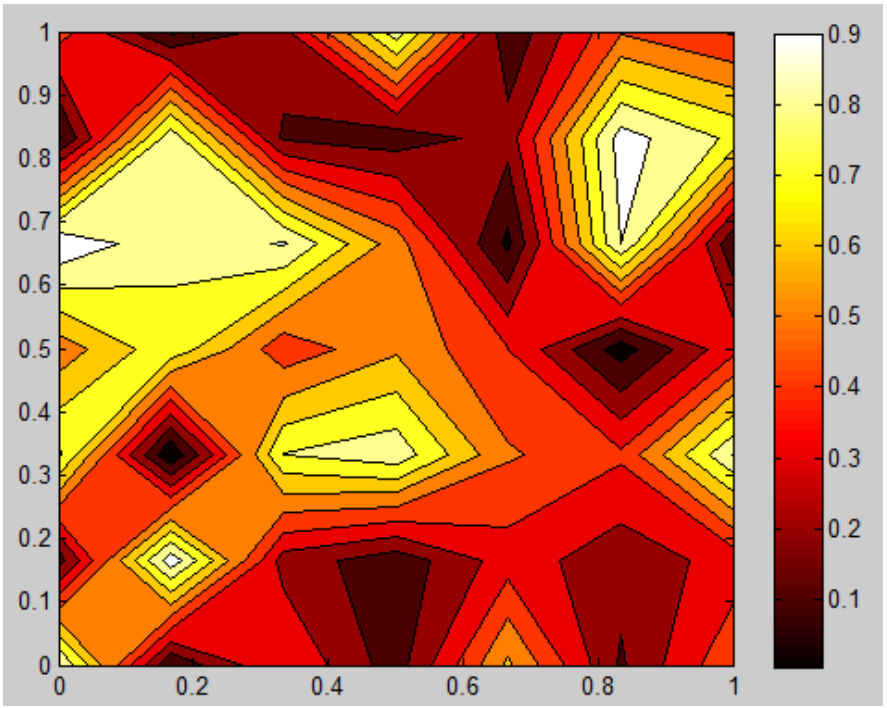


Electrostatic Potential Solution: 2D

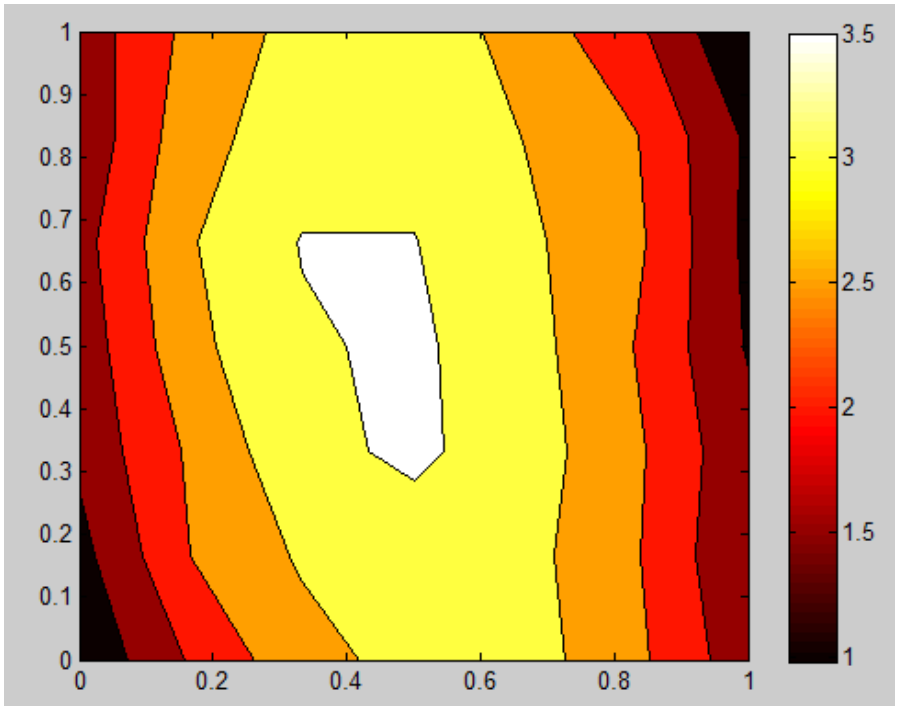
- Connections in two directions creates a total of 5 non-vanishing diagonals in our linear algebra problem:

$$\begin{pmatrix} -4 & 1 & 0 & & & 1 & 0 & 0 \\ 1 & -4 & 1 & \cdots & & 0 & 1 & 0 \\ 0 & 1 & -4 & & & 0 & 0 & 1 \\ & \vdots & & \ddots & & & \vdots & \\ 1 & 0 & 0 & & -4 & 1 & & 0 \\ 0 & 1 & 0 & \cdots & 1 & -4 & & 1 \\ 0 & 0 & 1 & & 0 & 1 & & -4 \end{pmatrix} \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \vdots \\ \varphi_{N-2} \\ \varphi_{N-1} \\ \varphi_N \end{pmatrix} = \frac{h^2}{\epsilon_o} \begin{pmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ \vdots \\ \rho_{N-2} \\ \rho_{N-1} \\ \rho_N \end{pmatrix}$$

Electrostatic Potential Solution: 2D



Charge distribution in 2D
(7x7 grid)



Electrostatic potential in 2D
(7x7 grid)

Spin Array Example

- Consider an array of spins $\{\sigma_i\}$, coupled by an exchange interaction
- Ising-model Hamiltonian is given by:

$$\mathcal{H} = \sum_i M_i \sigma_i + \sum_{\langle ij \rangle} J_{ij} \sigma_i \sigma_j$$

- The brackets often are interpreted to mean nearest-neighbor interactions only

Spin Array Solution: 1D

- Convert Hamiltonian into matrix, assuming nearest neighbor interaction only:

$$\mathcal{H} = \begin{pmatrix} M & J & 0 & & 0 & 0 & 0 \\ J & M & J & \cdots & 0 & 0 & 0 \\ 0 & J & M & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & M & J & 0 \\ 0 & 0 & 0 & \cdots & J & M & J \\ 0 & 0 & 0 & & 0 & J & M \end{pmatrix}$$

- Use Schrodinger equation to set up eigenproblem:

$$\mathcal{H}\Psi = E\Psi$$

- Choose basis spinor wavefunction :

$$\Psi = (\sigma_1, \sigma_2, \cdots, \sigma_N)^T$$

Spin Array Solution: 1D

- In MATLAB, use the following:

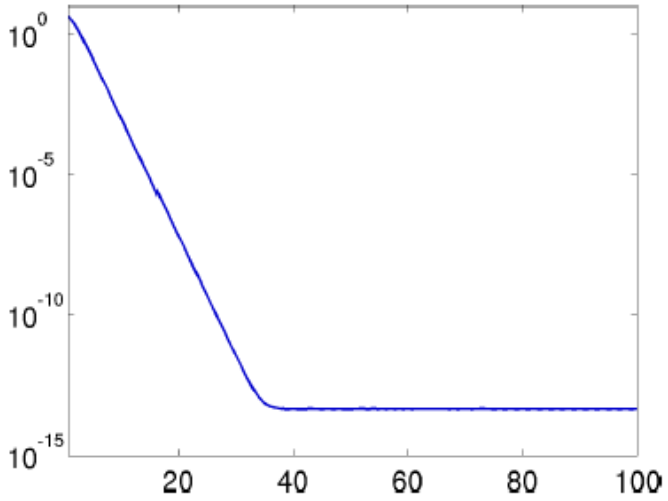
```
>> N=6; M=2; J=1; A=diag(M*ones(N,1))+diag(J*ones(N-1,1),1)+diag(J*ones(N-1,1),-1)

A =

     2     1     0     0     0     0
     1     2     1     0     0     0
     0     1     2     1     0     0
     0     0     1     2     1     0
     0     0     0     1     2     1
     0     0     0     0     1     2
```

How to Solve $Ax = \lambda x$?

- Eigenproblem Solution Techniques
  - Power Methods
  - Inverse Iteration
  - Atomic Transformations
  - Factorization Methods
- Linear algebra software packages
  - Linear Algebra Package (LAPACK)
  - MATLAB `eigs` package

Power Method

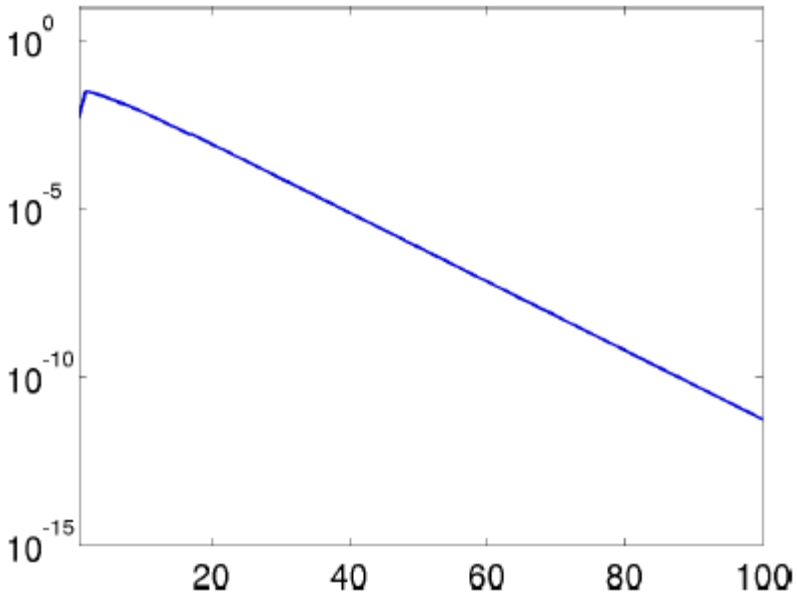
- Algorithm:
  - Initially guess dominant eigenvector v_o
  - Let $v_{k+1} = Av_k$ (optionally: normalize each step)
  - Dominant eigenvalue $\lambda_1 = \frac{{v_k}^T A v_k}{{v_k}^T v_k}$
- To find other eigenvalues:
  - Inverse power method
  - Shifted inverse power method



Error vs. iteration number

Inverse Iteration

- Formalizes concept of converging on a target eigenvalue & eigenvector
- Algorithm:
  - Start with approximate eigenvalue τ and random unit vector b_o
  - Let $x_k = (\boldsymbol{A} - \tau \mathbf{1})^{-1} b_{k-1}$
  - Let $b_k = \frac{x_k}{|x_k|}$
  - Repeat until tolerance reached
  - Our eigenvalue is given by $\lambda = {b_k}^T \cdot \boldsymbol{A} \cdot b_k$



Error vs. iteration number

Inverse Iteration Challenges

- For unlucky b_o, convergence too slow
- For multiple close roots, can only find one eigenvector
- For non-symmetric real matrices, can't find complex conjugate pairs

Transformation Methods

- General concept of similarity transformations: $A \rightarrow Z^{-1}AZ$
  - Atomic transformations: construct each Z explicitly
  - Factorization methods: QR and QL methods
- Keep iterating atomic transformations:
  - Until off-diagonal elements are small: then use Z matrix to read off eigenvectors
  - Otherwise: use factorization approach

Jacobi Transformations

- Atomic transformation:

$$\mathbf{P}_{pq} = \begin{bmatrix} 1 & & & & & & \\ & \cdots & & & & & \\ & & c & \cdots & s & & \\ & & \vdots & 1 & \vdots & & \\ & & -s & \cdots & c & & \\ & & & & & \cdots & \\ & & & & & & 1 \end{bmatrix}$$

- Generalizes rotation matrix: $\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$

$$\mathbf{A}' = \mathbf{P}_{pq}^T \cdot \mathbf{A} \cdot \mathbf{P}_{pq} \qquad S' = S - 2|a_{pq}|^2$$

Householder Transformation

- Basic approach discussed previously

$$\mathbf{P} = \mathbf{1} - 2\mathbf{w} \cdot \mathbf{w}^T \qquad \mathbf{A}' = \mathbf{P} \cdot \mathbf{A} \cdot \mathbf{P} = \left[\begin{array}{c|ccc} a_{11} & k & 0 & \cdots & 0 \\ \hline k & & & & \\ 0 & & & & \\ \vdots & & & \text{irrelevant} & \\ 0 & & & & \end{array}\right]$$

- Keys to this strategy:
  - Construct vector *w* to eliminate most elements: $w = [0, a_{21} - \alpha, a_{31}, \cdots, a_{N1}]^T$
  - Where $\alpha = -\operatorname{sgn} a_{21} \sqrt{\sum_{j=2}^{N} \left(a_{j1}\right)^2}$
  - Iterate recursively to tridiagonal form and solve: $R \cdot x = \lambda \prod_i {Q_i}^T x$

Factorization in Eigenproblems

- Most common approach known as QR method

$$\mathbf{A} = \mathbf{Q} \cdot \mathbf{R} \qquad \mathbf{A}' = \mathbf{R} \cdot \mathbf{Q} \qquad \mathbf{A}' = \mathbf{Q}^T \cdot \mathbf{A} \cdot \mathbf{Q}$$

- Can also do the same with $\mathbf{A} = \mathbf{Q} \cdot \mathbf{L}$
- QL algorithm:
  - Use Householder algorithm to construct Q_k
  - Factorize: $A_k = Q_k L_k$
  - Rearrange: $A_{k+1} = L_k Q_k = {Q_k}^T A_k Q_k$

Spin Array Solution: 1D

```
>> [V,D]=eigs(full(A),N)
Warning: For real symmetric problems, must have number of eigenvalues k < n.
Using eig(full(A)) instead.
> In eigs>checkInputs at 926
  In eigs at 94

V =

    0.2319   -0.4179    0.5211   -0.5211   -0.4179    0.2319
    0.4179   -0.5211    0.2319    0.2319    0.5211   -0.4179
    0.5211   -0.2319   -0.4179    0.4179   -0.2319    0.5211
    0.5211    0.2319   -0.4179   -0.4179   -0.2319   -0.5211
    0.4179    0.5211    0.2319   -0.2319    0.5211    0.4179
    0.2319    0.4179    0.5211    0.5211   -0.4179   -0.2319


D =

    3.8019         0         0         0         0         0
         0    3.2470         0         0         0         0
         0         0    2.4450         0         0         0
         0         0         0    1.5550         0         0
         0         0         0         0    0.7530         0
         0         0         0         0         0    0.1981
```

Spin Array Solution: 2D

- Convert Hamiltonian into matrix, assuming nearest neighbor interactions in 2 directions:

$$\mathcal{H} = \begin{pmatrix} M & J & 0 & & J & 0 & 0 \\ J & M & J & \cdots & 0 & J & 0 \\ 0 & J & M & & 0 & 0 & J \\ & \vdots & & \ddots & & \vdots & \\ J & 0 & 0 & & M & J & 0 \\ 0 & J & 0 & \cdots & J & M & J \\ 0 & 0 & J & & 0 & J & M \end{pmatrix}$$

- Use Schrodinger equation to set up eigenproblem:

$$\mathcal{H}\Psi = E\Psi$$

- Choose basis spinor wavefunction :

$$\Psi = (\sigma_1, \sigma_2, \cdots, \sigma_N)^T$$

Spin Array Solution: 2D

- In MATLAB, use the following:

```
>> N=9; M=2; J=1; A=diag(M*ones(N,1))+diag(J*ones(N-1,1),1)+diag(J*ones(N-1,1),-1);
>> A=A+diag(J*ones(N-sqrt(N),1),sqrt(N))+diag(J*ones(N-sqrt(N),1),-sqrt(N))

A =

     2     1     0     1     0     0     0     0     0
     1     2     1     0     1     0     0     0     0
     0     1     2     1     0     1     0     0     0
     1     0     1     2     1     0     1     0     0
     0     1     0     1     2     1     0     1     0
     0     0     1     0     1     2     1     0     1
     0     0     0     1     0     1     2     1     0
     0     0     0     0     1     0     1     2     1
     0     0     0     0     0     1     0     1     2
```

Spin Array Solution: 2D

```
>> [V,D]=eig(full(A))

V =

   -0.2137   -0.4253   -0.2629   -0.1941    0.5774   -0.1941    0.2629    0.4253    0.2137
    0.2985    0.4253   -0.2629    0.4011   -0.0000   -0.4011   -0.2629    0.4253    0.2985
   -0.3362   -0.2629    0.4253    0.3701   -0.0000    0.3701   -0.4253    0.2629    0.3362
    0.4011    0.2629    0.4253   -0.2985   -0.0000    0.2985    0.4253    0.2629    0.4011
   -0.4274    0.0000    0.0000   -0.3882   -0.5774   -0.3882   -0.0000    0.0000    0.4274
    0.4011   -0.2629   -0.4253   -0.2985    0.0000    0.2985   -0.4253   -0.2629    0.4011
   -0.3362    0.2629   -0.4253    0.3701    0.0000    0.3701    0.4253   -0.2629    0.3362
    0.2985   -0.4253    0.2629    0.4011   -0.0000   -0.4011    0.2629   -0.4253    0.2985
   -0.2137    0.4253    0.2629   -0.1941    0.5774   -0.1941   -0.2629   -0.4253    0.2137


D =

   -1.2742         0         0         0         0         0         0         0         0
         0    0.3820         0         0         0         0         0         0         0
         0         0    1.3820         0         0         0         0         0         0
         0         0         0    1.4710         0         0         0         0         0
         0         0         0         0    2.0000         0         0         0         0
         0         0         0         0         0    2.5290         0         0         0
         0         0         0         0         0         0    2.6180         0         0
         0         0         0         0         0         0         0    3.6180         0
         0         0         0         0         0         0         0         0    5.2742
```

Bandstructure Problem

- Amounts to solving an eigenvalue equation for a system with discrete translational symmetry
- Examples include:
  - Electronic bandstructure: $\left[-\frac{\hbar^2}{2m}\nabla^2 + V(x)\right]\Psi(x) = \hbar\omega\Psi(x)$
  - Photonic bandstructure: $\nabla \times [\epsilon^{-1}(\nabla \times H)] = \left(\frac{\omega}{c}\right)^2 H$
  - Phononic bandstructure: $\nabla \times [C(\nabla \times u)] = -\rho\omega^2 u$

Next Class

- Continue formulating eigenproblems for electro-optic systems, with emphasis on band structures
- Continue reviewing Joannopoulos, Chapter 2 and Appendix D