ECE 695
Numerical Simulations
Lecture 8: Photonic Bandstructures in MPB

Prof. Peter Bermel

January 27, 2017

Outline

- Reformulating the eigenproblem
- Square rod lattice bandstructure
- Triangular rod lattice bandstructure
- Plotting bandstructures
- Visualizing fields
- Maximizing bandgaps
- Diamond lattice
- Finding and tuning point defects

Reformulating the Eigenproblem

- Magnetic field in planewave basis:

$$\left|H_{\vec{k}}\right\rangle \cong \sum_{\{m_j\}} \vec{h}_{\{m_j\}} e^{i\sum_{j,k} m_j \vec{G}_j \cdot n_k \vec{R}_k / N_k} = \sum_{\{m_j\}} \vec{h}_{\{m_j\}} e^{2\pi i \sum_j m_j n_j / N_j}$$

- Operator scales like $\mathcal{O}(N \log N)$:

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

- Tensor-based averaging aids in convergence:

$$\widetilde{\varepsilon^{-1}} = \overline{\varepsilon^{-1}} P + \overline{\varepsilon}^{-1}(1 - P) \qquad P_{ij} = n_i n_j$$

- MPB performs conjugate-gradient minimization of Block Rayleigh quotient

First Bandstructure: Input

```
(set! num-bands 8) ; sets p, the number of bands

(set! k-points (list (vector3 0 0 0)      ; Gamma
                     (vector3 0.5 0 0)    ; X
                     (vector3 0.5 0.5 0) ; M
                     (vector3 0 0 0)))    ; Gamma

(set! k-points (interpolate 4 k-points))
; creates 4 intermediate values between each pair
```

First Bandstructure: Input

```
(set! geometry (list (make cylinder
   (center 0 0 0) (radius 0.2)
   (height infinity)
   (material (make dielectric (epsilon
12))))))

(set! geometry-lattice (make lattice (size 1
1 no-size)))

(set! resolution 32)
```

First Bandstructure: Output

```
unix% mpb sqrods.ctl
tefreqs:, k index, kx, ky, kz, kmag/2pi,
band 1, band 2, band 3, band 4, band 5, band
6, band 7, band 8
...
tefreqs:, 13, 0.3, 0.3, 0, 0.424264,
0.372604, 0.540287, 0.644083, 0.81406,
0.828135, 0.890673, 1.01328, 1.1124
Gap from band 1 (0.282623311147724) to band
2 (0.419334798706834), 38.951466088891%
Gap from band 4 (0.715673834754345) to band
5 (0.743682920649084), 3.838552265034%
```

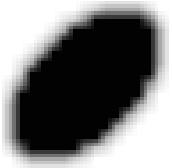
Triangular Lattice

```
(set! num-bands 8)

(set! geometry-lattice (make lattice (size 1 1 no-size)
                         (basis1 (/ (sqrt 3) 2) 0.5)
                         (basis2 (/ (sqrt 3) 2) -0.5)))
(set! geometry (list (make cylinder
                       (center 0 0 0) (radius 0.2) (height infinity)
                       (material (make dielectric (epsilon 12))))))
(set! k-points (list (vector3 0 0 0)           ; Gamma
                     (vector3 0 0.5 0)         ; M
                     (vector3 (/ -3) (/ 3) 0) ; K
                     (vector3 0 0 0)))         ; Gamma
(set! k-points (interpolate 4 k-points))
(set! resolution 32)
(run-tm (output-at-kpoint (vector3 (/ -3) (/ 3) 0)
                          fix-efield-phase output-efield-z))
(run-te)
```

Triangular Lattice

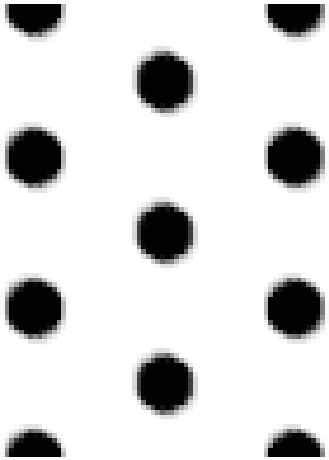
```
unix% mpb tri-rods.ctl >& tri-rods.out
unix% h5topng -S 3 epsilon.h5
```



Why does this look so distorted?

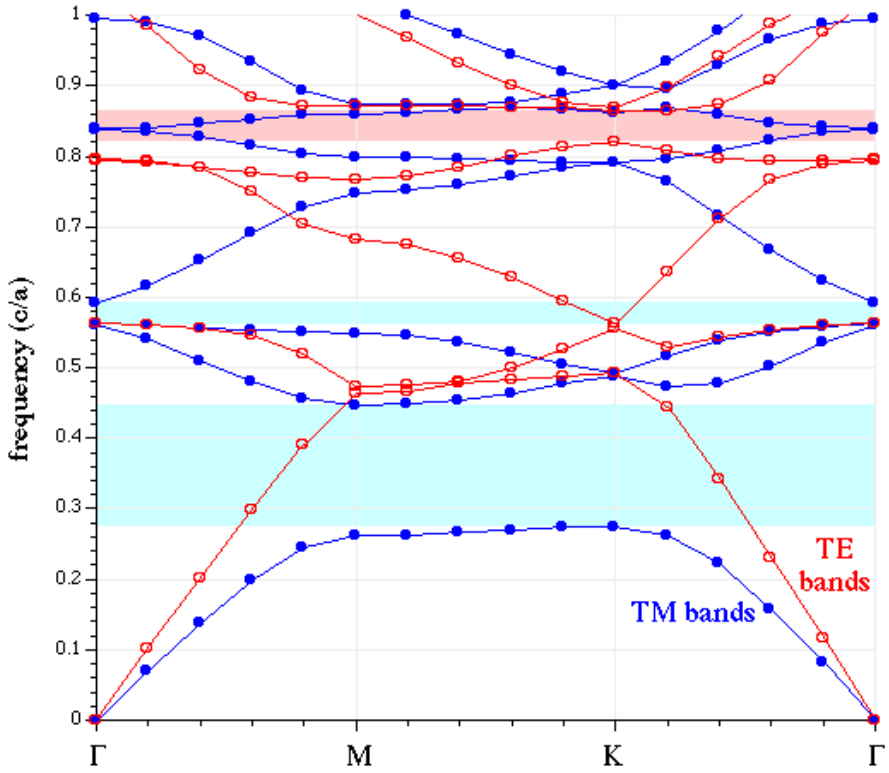
Triangular Lattice

```
unix% mpb-data -r -m 3 -n 32 epsilon.h5
unix% h5ls epsilon.h5
data                        Dataset {32, 32}
data-new                    Dataset {96, 83}
description                 Dataset {SCALAR}
lattice\ copies             Dataset {3}
lattice\ vectors            Dataset {3, 3}
unix% h5topng epsilon.h5:data-new
```



This fixes the problem.

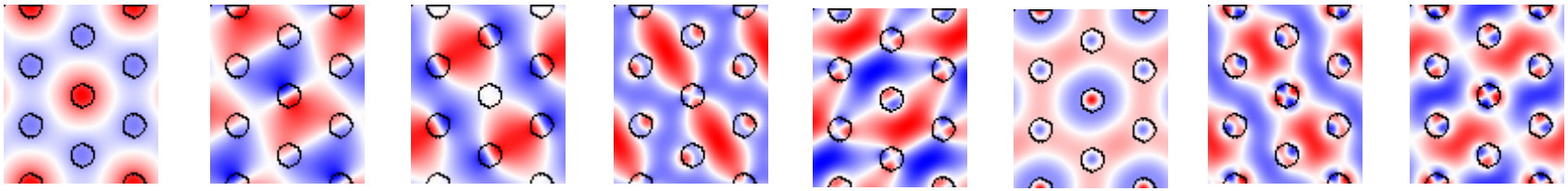
Plotting Bandstructures




```
unix% grep tmfreqs tri-rods.out > tri-rods.tm.dat
unix% grep tefreqs tri-rods.out > tri-rods.te.dat
```

Visualizing Fields

TM band 1 TM band 2 TM band 3 TM band 4 TM band 5 TM band 6 TM band 7 TM band 8

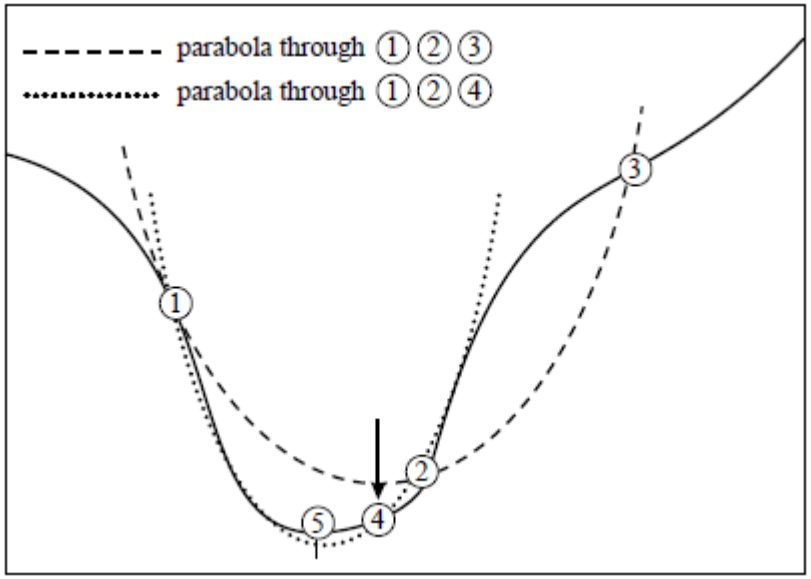


```
(run-tm output-efield-z)
(run-te (output-at-kpoint (vector3 0.5 0 0) output-hfield-z output-dpwr))

unix% mpb-data -r -m 3 -n 32 e.k11.b*.z.tm.h5
unix% h5topng -C epsilon.h5:data-new -c bluered -Z -d z.r-new e.k11.b*.z.tm.h5
```

Maximizing Bandgaps: Brent's Method

- Assumes a concave function
- Algorithm:
  - Evaluate function at bracket endpoints & center
  - Fit parabola
  - Find x_{min} & $f(x_{min})$
  - Keep two closest points for bracket and repeat until bracket is around $\sqrt{\varepsilon}$
- Infer optimum based



Maximizing Bandgaps

```
(define (first-tm-gap r)
(set! geometry (list (make cylinder
                       (center 0 0 0) (radius r) (height infinity)
                       (material (make dielectric (epsilon 12))))))
(run-tm)
(retrieve-gap 1)) ; return the gap from TM band 1 to TM band 2
(set! num-bands 2)
(set! mesh-size 7) ; increase from default value of 3
; libctl provides a built-in function (using Brent's algorithm).
; We just tell it to find the maximum between 0.1 to 0.5 (rtol=0.1):
(define result (maximize first-tm-gap 0.1 0.1 0.5))
(print "radius at maximum: " (max-arg result) "\n")
(print "gap size at maximum: " (max-val result) "\n")
```

Radius at maximum: 0.176393202250021

Gap size at maximum: 48.6252611051049

The tolerance of 0.1 that we specified means that the true maximum is within 0.1 * 0.176393202250021, or about 0.02, of the radius found here

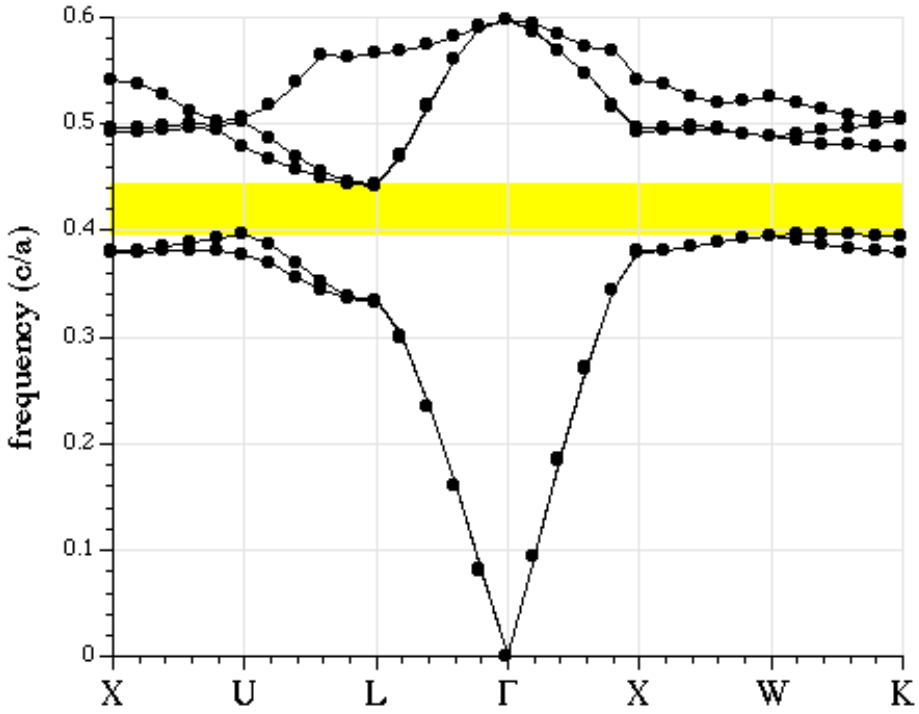
Diamond Lattice

```
(set! geometry-lattice (make lattice
                         (basis-size (sqrt 0.5) (sqrt 0.5) (sqrt 0.5))
                         (basis1 0 1 1)
                         (basis2 1 0 1)
                         (basis3 1 1 0)))
; Corners of the irreducible Brillouin zone for the fcc lattice
(set! k-points (interpolate 4 (list
                                (vector3 0 0.5 0.5)             ; X
                                (vector3 0 0.625 0.375)         ; U
                                (vector3 0 0.5 0)               ; L
                                (vector3 0 0 0)                 ; Gamma
                                (vector3 0 0.5 0.5)             ; X
                                (vector3 0.25 0.75 0.5)         ; W
                                (vector3 0.375 0.75 0.375))))   ; K
```

Diamond Lattice

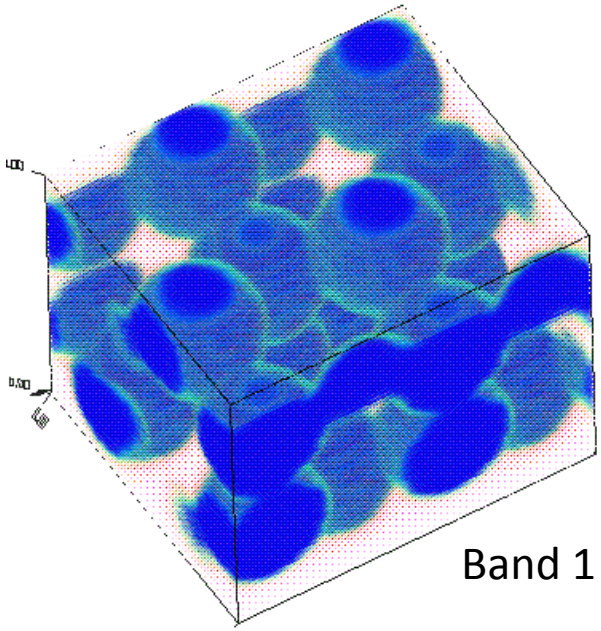
```
 ; define parameters (can also be set from command-line)
 (define-param eps 11.56) ; dielectric constant of spheres
 (define-param r 0.25)    ; the radius of the spheres
 (define diel (make dielectric (epsilon eps)))
 ; A diamond lattice has two "atoms" per unit cell:
 (set! geometry (list (make sphere (center 0.125 0.125 0.125)
(radius r) (material diel))
                      (make sphere (center -0.125 -0.125 -
0.125) (radius r) (material diel))))
 (set-param! resolution 16) ; use a 16x16x16 grid
 (set-param! mesh-size 5)
 (set-param! num-bands 5)
 ; run calculation, outputting electric-field energy density
at the U point:
 (run (output-at-kpoint (vector3 0 0.625 0.375) output-dpwr))
```

Diamond Lattice Bandstructure

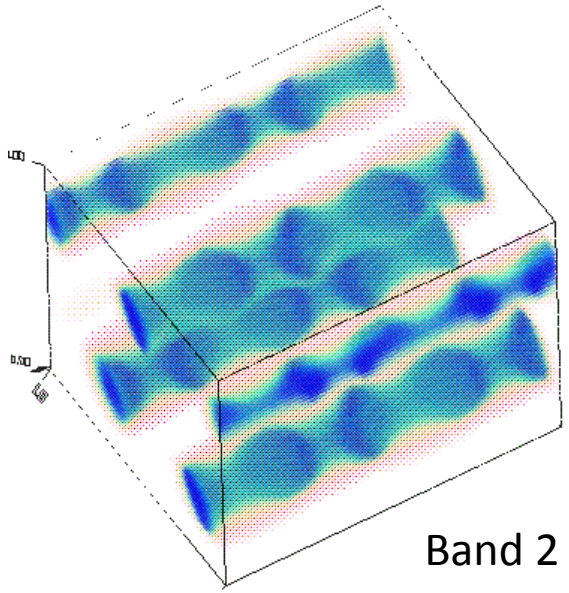



```
unix% nohup mpb diamond.ctl >& diamond.out &
unix% grep Gap diamond.out
Gap from band 2 (0.396348703007373) to band 3 (0.440813418580596),
10.6227251392791%
unix% grep freqs diamond.out > diamond.dat
```

Diamond Lattice: Output



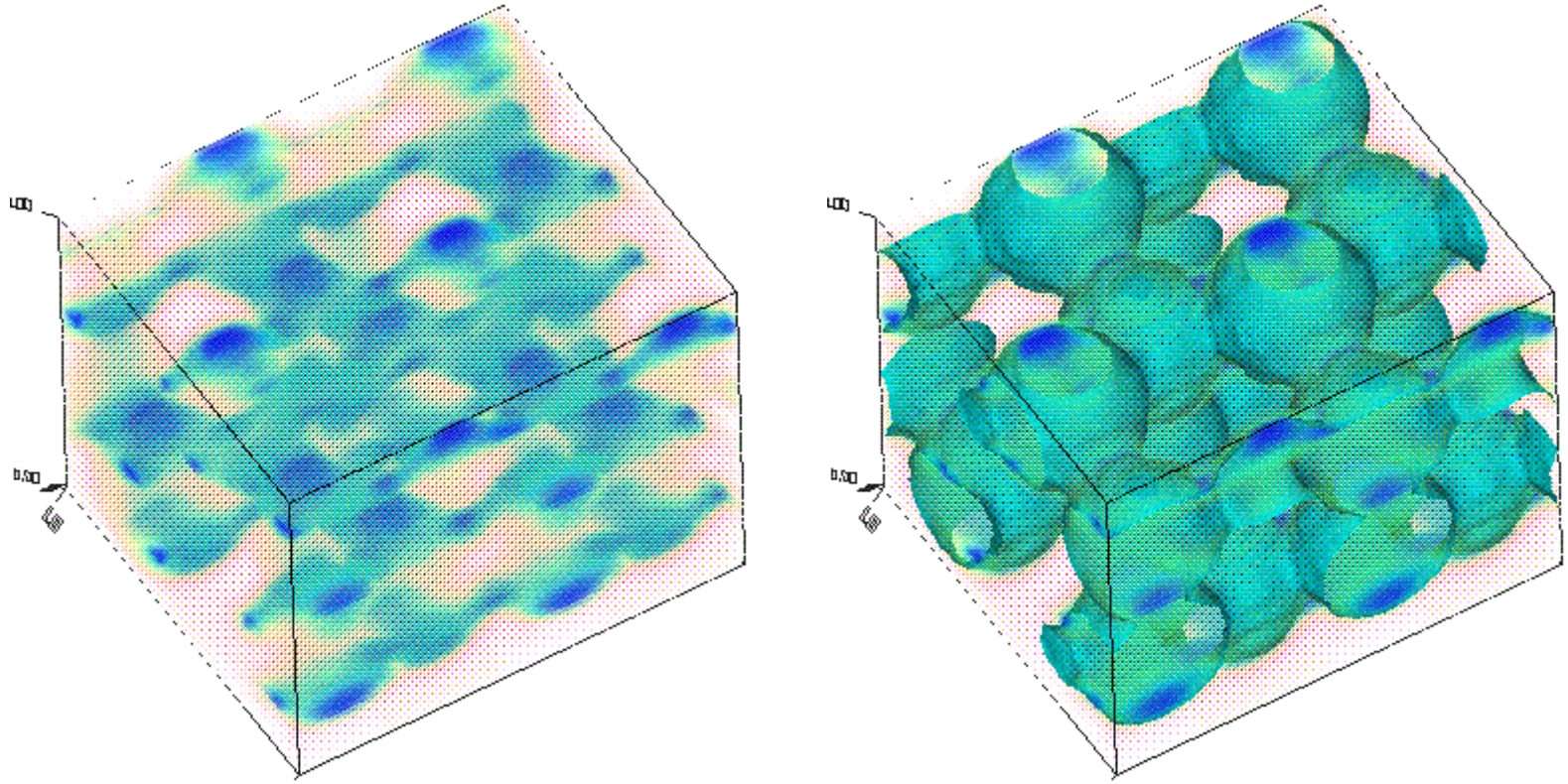
Band 1



Band 2

```
unix% mpb-data -m 2 -r -n 32 epsilon.h5
dpwr.k06.b*.h5
unix% h5tov5d -o diamond.v5d -d data-new epsilon.h5
dpwr.k06.b*.h5
unix% vis5d diamond.v5d &
```

Diamond Lattice: Output



Third band, just above the bandgap

Finding Point Defects

```
(set! geometry-lattice (make lattice (size 5 5 no-size)))

(set! geometry (list (make cylinder
                       (center 0 0 0) (radius 0.2) (height infinity)
                       (material (make dielectric (epsilon 12))))))
(set! geometry (geometric-objects-lattice-duplicates geometry))

(set! geometry (append geometry
                         (list (make cylinder (center 0 0 0)
                                     (radius 0.2) (height infinity)
                                     (material air)))))
```

Punches an air hole into a 5 x 5 lattice

0.624794702341156

Finding Point Defects

```
(set! resolution 16)
; Only a single k point is needed for a point-defect calculation
(set! k-points (list (vector3 0.5 0.5 0)))
;For a supercell, the original bands are folded many times over
;We need many more bands to reach the same frequencies
(set! num-bands 50)
(output-efield-z 25)
(get-dfield 25)  ; compute the D field for band 25
(compute-field-energy)  ; compute the energy density from D
(print "energy in cylinder: "
 (compute-energy-in-objects (make cylinder (center 0 0 0)
                                  (radius 1.0) (height infinity)
                                  (material air))))
```

0.624794702341156

Finding Point Defects

```
(set! num-bands 1)  ; only need to compute a single band, now!
(set! target-freq (/ (+ 0.2812 0.4174) 2))
(set! tolerance 1e-8)
(run-tm)
```

0.378166

Tuning Point Defect Mode

```
(define old-geometry geometry) ; save 5x5 grid with missing
rod
(define (rootfun eps)
  ; add the cylinder of epsilon = eps to the old geometry:
  (set! geometry (append old-geometry
                         (list (make cylinder (center 0 0 0)
                                     (radius 0.2) (height infinity)
                                     (material (make dielectric
                                                 (epsilon eps)))))))
  (run-tm)  ; solve for the mode (using the targeted solver)
  (print "eps = " eps " gives freq. = " (list-ref freqs 0))
  (- (list-ref freqs 0) 0.314159)) ; 1st band freq.-0.314159

(define rooteps (find-root rootfun 0.01 1 12))
(print "root (value of epsilon) is at: " rooteps "\n")
```

epsilon	frequency
1	0.378165893321125
12	0.283987088221692
6.5	0.302998920718043
5.14623274327171	0.317371748739314
5.82311637163586	0.309702408341706
5.41898003340128	0.314169110036439
5.62104820251857	0.311893530112625

Next Class

- Use Fast Fourier Transforms (including FFTW)
- Reference: Numerical Recipes and FFTW User Manual: http://fftw.org/fftw3_doc/