MPB Demo
— numerical Maxwell eigensolver on MIT Server

For more information on MPB, see also the MPB web page at http://ab-initio.mit.edu/mpb (which includes a link to documentation). All of these files and links can also be found on the course web page.

The following are a sequence of MIT Server and Matlab commands that I will (hopefully) use in class to perform a simple calculation and analyze the output. Here, we analyze a simple 2d waveguide formed by an $\varepsilon=12$ region with thickness 1, surrounded by air.

The MPB input file, 2dwaveguide.ctl, is on the following page. The last page is the input file and commands for a more complicated problem—a 2d waveguide formed by a periodic sequence of dielectric cylinders.

A note on units: Since Maxwell's equations are scale-invariant, there is no need for MPB to settle on any particular units. Instead, you simply pick whatever units of distance you want — typically you select some characteristic lengthscale *a* in the system and set that equal to 1. Then, you specify a wavevector *k* to MPB in units of $2\pi/a$, and the frequencies ω are returned in units of $2\pi c/a$. See also the MPB manual for more details (it gets more complicated for 2d-periodic systems, where the lattice or reciprocal lattice vectors are used as a basis for vectors or wavevectors).

```
% foo = unix command
>> foo = Matlab command

% add mpb

% mpb 2dwaveguide.ctl > 2dwaveguide.out
% cat 2dwaveguide.out
% grep tmyevenfreqs: 2dwaveguide.out > foo.dat

>> foo = dlmread('foo.dat', ',', 1, 1)

>> kx = foo(:,2)
>> f = foo(:,6:end)

>> plot(kx, f, 'ro-')
>> hold on
>> plot(kx, kx, 'k', 'LineWidth', 2)

>> patch([kx;flipud(kx)], [kx; ones(size(kx))*max(kx)], [0.9,0.9,0.9])

% grep tmyoddfreqs: 2dwaveguide.out > foo.dat

>> foo = dlmread('foo.dat', ',', 1, 1)
>> fo = foo(:,6:end)
>> plot(kx, fo, 'bo-')

>> plot(kx, kx / sqrt(12), 'k')

% h5ls epsilon.h5

>> ep = hdf5read('epsilon.h5', 'data')
>> figure
>> plot(ep, 'k')

% h5ls e.k01.b01.z.tm.h5

>> ez1 = hdf5read('e.k01.b01.z.tm.h5', 'z.r') +
hdf5read('e.k01.b01.z.tm.h5','z.i') * i;
>> plot(real(ez1) * 100, 'r');

>> ez2 = hdf5read('e.k01.b02.z.tm.h5', 'z.r') +
hdf5read('e.k01.b02.z.tm.h5','z.i') * i;
>> plot(real(ez2) * 100, 'b');

>> ez3 = hdf5read('e.k01.b03.z.tm.h5', 'z.r') +
hdf5read('e.k01.b03.z.tm.h5','z.i') * i;
>> plot(real(ez3) * 100, 'm');

>> (ep .* ez1) * ez3'
```

```
; Example MPB input file for 18.325, illustrating a simple 2d dielectric
; waveguide along the x direction.  Run it with:
;       mpb 2dwaveguide.ctl > 2dwaveguide.out
; to produce an output file 2dwaveguide.out (as well as some .h5 data files).
; As described in the manual, you can extract the eigenfrequencies
; by doing "grep freqs: 2dwaveguide.out" at the Unix shell.

; (Note that anything after a ";" on a line is ignored by MPB.)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; First, we will define some parameters describing our structure.  Defining
; them symbolically here makes it easier to change them.  (e.g. we
; can change the dielectric constant from the command line via
; "mpb eps-hi=13 2dwaveguide.ctl".)  We then use these parameters below

(define-param eps-hi 12) ; the waveguide dielectric constant
(define-param eps-lo 1) ; the surrounding low-dielectric material
(define-param h 1) ; the thickness of the waveguide (arbitrary units)

(define-param Y 10) ; the size of the computational cell in the y direction

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Define the structure and the computational cell

; Here we define the size of the computational cell.  Since it is 2d,
; it has no-size in the z direction.  Because it is a waveguide in the
; x direction, then the eigenproblem at a given k has no-size in the
; x direction as well.
(set! geometry-lattice (make lattice (size no-size Y no-size)))

; the default-material is what fills space where we haven't placed objects
(set! default-material (make dielectric (epsilon eps-lo)))

; a list of geometric objects to create structures in our computational cell:
; (in this case, we only have one object, a block to make the waveguide)
(set! geometry
      (list (make block ; a dielectric block (a rectangle)
          (center 0 0 0) ; centered at origin
          (size infinity h infinity) ; block is finite only in y direction
          (material (make dielectric (epsilon eps-hi))))))

; MPB discretizes space with a given resolution.   Here, we set
; a resolution of 32 pixels per unit distance.  Thus, with Y=10
; our comptuational cell will be 320 pixels wide.  In general,
; you should make the resolution fine enough so that the pixels
; are much smaller than the wavelength of the light.
(set-param! resolution 32)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Tell MPB what eigenmodes we want to compute.

; Generally, we want omega(k) for a range of k values.  MPB
; can automatically interpolate a set of k values between any
; given bounds.  Here, we will interpolate 10 k's between 0 and 2.
(define-param kmin 0)
(define-param kmax 2)
(define-param k-interp 10)
; k-points is the list of k values that MPB computes eigenmodes at.
; (vector3 x y z) specifies a vector.  (k is in units of 2 pi/distance)
(set! k-points (interpolate k-interp
                          (list (vector3 kmin 0 0) (vector3 kmax 0 0))))

; we also need to specify how many eigenmodes we want to compute, given
; by "num-bands":
(set-param! num-bands 5)

; to compute *all* the modes, we now simply type (run).
; However, it is convenient to compute only one symmetry of mode
; at a time.  In particular, we will compute only TM (E in z direction)
; modes, and separately compute even and odd modes with respect to the
; y=0 mirror symmetry plane.
(run-tm-yeven)
(run-tm-yodd)
; (If we don't have y=0 mirror symmetry we should just use run-tm).

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; That's it!  We're done!  However, suppose we now want to get the
; *fields* at a given k.  To do this, we'll call the run function
; again, this time giving it an option to output the modes.

(define-param k 1) ; the k value where we'll output the modes
(set! k-points (list (vector3 k 0 0))) ; compute only a single k now

; output-efield-z does just what it says.  There are also options
; to output any other field component we care to examine.
(run-tm output-efield-z)
```

```
; Example MPB input file for 18.325, for a periodic (period = 1) sequence
; of dielectric cylinders in the x direction.  (This file is otherwise
; very similar to 2dwaveguide.ctl ... refer to that file for more details.)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Some parameters:

(define-param eps-hi 12) ; the waveguide dielectric constant
(define-param eps-lo 1) ; the surrounding low-dielectric material
(define-param h 1) ; the thickness of the waveguide (arbitrary units)

(define-param r 0.2) ; the radius of the cylinders

(define-param Y 10) ; the size of the computational cell in the y direction

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Define the structure and the computational cell

; note that now the size in the x direction is 1 (one period)
(set! geometry-lattice (make lattice (size 1 Y no-size)))

(set! default-material (make dielectric (epsilon eps-lo)))

(set! geometry
      (list (make cylinder ; cylinder oriented along z direction
          (center 0 0 0) ; centered at origin
          (radius r) (height infinity)
          (material (make dielectric (epsilon eps-hi))))))

(set-param! resolution 16)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Tell MPB what eigenmodes we want to compute.

(define-param kmin 0)
(define-param kmax 2)
(define-param k-interp 20)
; k-points is the list of k values that MPB computes eigenmodes at.
; (vector3 x y z) specifies a vector.  (k is in units of 2 pi/distance)
(set! k-points (interpolate k-interp
                        (list (vector3 kmin 0 0) (vector3 kmax 0 0))))

; we also need to specify how many eigenmodes we want to compute, given
; by "num-bands":
(set-param! num-bands 5)

; let's just stick with the even modes for now.
(run-tm-yeven)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; uncomment to output some fields at k=0.4:

(define-param k 0.4) ; the k value where we'll output the modes
(set! k-points (list (vector3 k 0 0))) ; compute only a single k now

; (run-tm-yeven output-efield-z)
```

```
% foo = Unix Command
>> foo = Matlab command

% add mpb

% mpb 2dwaveguide-periodic.ctl > 2dwaveguide-periodic.out

% grep tmyevenfreqs: 2dwaveguide.out > foo.dat

>> foo = dlmread('foo.dat', ',', 1, 1)

>> kx = foo(:,2)
>> fe = foo(:,6:end)

>> plot(kx, fe, 'ro-')
>> hold on
>> plot(kx, kx, 'k', 'LineWidth', 2)
>> plot(kx + 1, kx, 'k', 'LineWidth', 2)
>> plot(-kx + 1, kx, 'k', 'LineWidth', 2)
>> plot(-kx + 2, kx, 'k', 'LineWidth', 2)
>> axis([0 2 0 0.6])

Run again, uncommenting line to output fields...

>> ez1 = hdf5read('e.k01.b01.z.tmyeven.h5', 'z.r') +
         hdf5read('e.k01.b01.z.tmyeven.h5','z.i') * i;
>> pcolor(real(ez1))
>> axis image

% mpb-data -x 5 e.k01.b01.z.tmyeven.h5 epsilon.h5
% h5topng -S 4 -Zc bluered -C epsilon.h5:data-new -d z.r-new
                                            e.k01.b01.z.tmyeven.h5
% add graphics
% xv e.k01.b01.z.tmyeven.png &
```