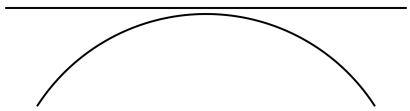
Simulation of bent waveguide coupler

> OS：Linux
> Program：Python 3.7.6
> Library：meep，numpy，pandas，matplotlib
> Date：2020-09-11

弯曲波导几何结构

本次仿真采用单弯曲侧向耦合波导。下图为计算元胞（cell）示意，弯曲的波导几何体（geometry）位于cell中部。波导与cell边缘有一定间隔（edge_x，edge_y），并且该间隔大于完美匹配层（pml_layers）厚度。图中横向为x轴，纵向为y轴，坐标原点（0, 0）位于图像中心。



参数设定：

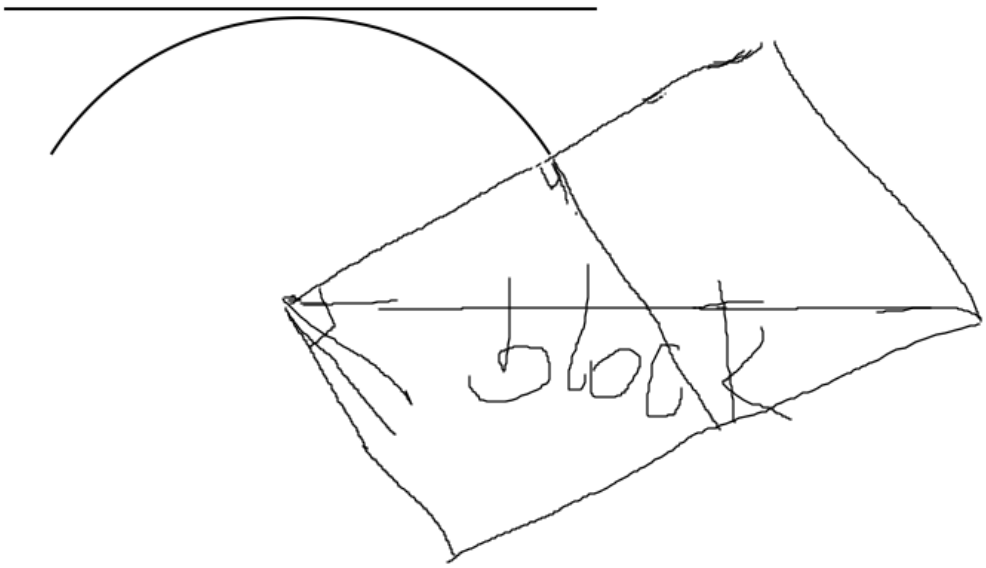
- 波导宽度：1
- 两个波导边缘间距：2
- 波导同等长度：200
- 计算分辨率：10（较常用的设定值）
- 弯曲半径：100
- 波导折射率：4
- 其余地方折射率：2.25（当然epsilon反映了折射率大小，但数值上并不等于折射率）

仿真方法

元胞与几何体定义

- 元胞对象：Cell类创建，传入三维；由于是二维仿真，第三维度设为0
- 几何体定义：由于最终传入SImulation定义属性的geometry是一个列表，列表中后定义的几何体属性会覆盖前面定义的，故依次创建geometry列表；另外由于Cell将计算空间限制再二维有限体积，则geometry中第三维度的参数数值没有影响。
    1. 对于整个Cell空间，设定折射率为2.25材料充斥的Block对象
    2. 加入创建直波导对象Block，折射率4
    3. 按照圆弧弯曲波导的外边缘为半径创建Cylinder对象（圆心坐标简单计算得到），折射率4
    4. 同样的圆心坐标，按照圆弧弯曲波导的内边缘为半径创建Cylinder对象，折射率2.25

5. 另外根据我们想要的弯曲波导几何长度再创建两个Block对象（折射率2.25），效果就是前步骤所创建切除弯曲波导的多余长出来的部分，如下图示意

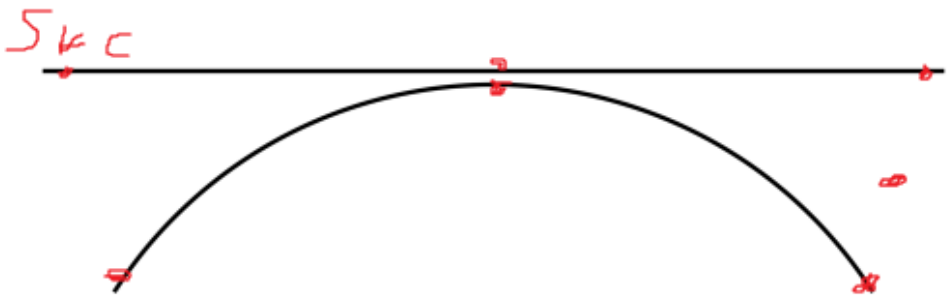


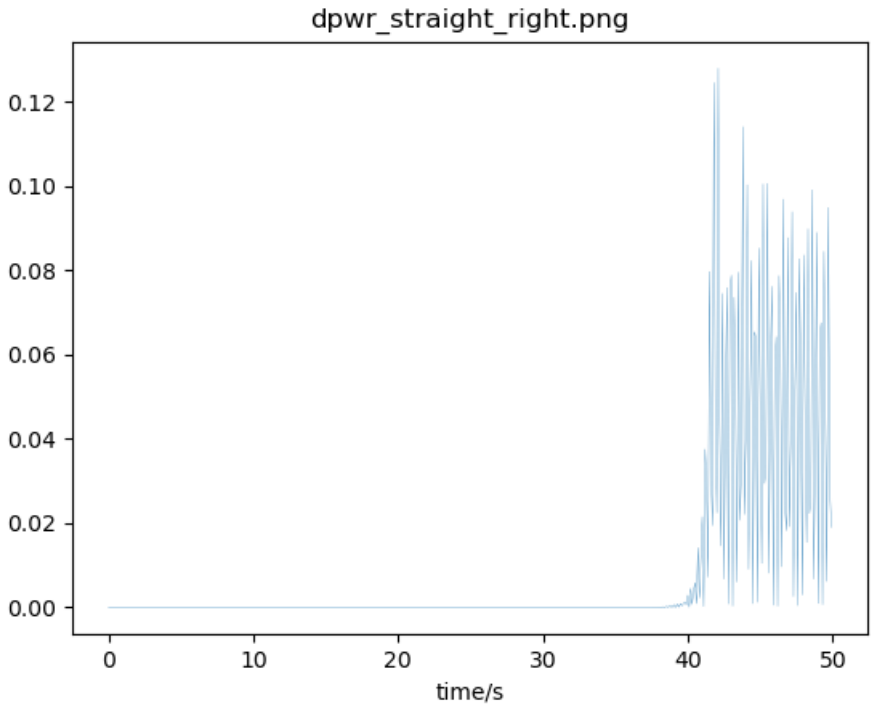
激发波源

- 连续波源
- y向偏振
- 中心位置为直形波导左端

监视点

即定义想要观察时域上能量密度和电场矢量大小变化的单个坐标点，示例程序中选取如下所示。最后可根据对应点计算所得的数据绘制时域曲线图，如下所示。





计算步骤

1. 创建SImulation仿真对象，必要的属性有resolution，pml_layers，geometry，cell，sources等
2. 执行run方法，进行计
   - `until`设定计算时间总步长
   - `mp.at_beginning(mp.in_volume(mp.Volume(center=mp.Vector3(), size=mp.Vector3(cell_x, cell_y, 0)), mp.output_epsilon))`意为初始`t`为0时输出全空间的电介质数据
   - 而后分别计算全空间`ey`、`dpwr`：`mp.to_appended("ey", mp.at_every(200.0, mp.in_volume(mp.Volume(center=mp.Vector3(), size=cell), mp.output_efield_y)))`、`mp.to_appended("dpwr", mp.at_every(200.0, mp.in_volume(mp.Volume(center=mp.Vector3(), size=cell), mp.output_dpwr)))`
   - 以及计算各个观察点的`ey`、`dpwr`
   - 输出均为HDF5格式文件，可用`unix% h5ls filename.h5`命令行查看其中数据维度及大小

结果展示

HDF5格式的文件利用Linux中h5utils软件转化那位png或txt数据文件，转换脚本参考meep在线文档或者此处附带的script.sh文件中内容。此处附带的对于单点计算可视化数据另外编写了python脚本h5topng.py，可参考调用。

源程序示例

wvg_wd.py

```
import meep as mp
import math
import scipy.constants as C
import matplotlib.pyplot as plt
import numpy as np
```

```
# Setup the main structure parameters and also the simulation size

scale_fac = 50*C.nano    # distance that one grid point represents
resolution = 10    # how many grid points per unit
width = 1.0           # waveguide width = 3microns

# 直-弯侧向耦合波导
separation = 2.0          # spacing between waveguides =4micron
# bend_r = 500        # bend radius = 500 micron
bend_r = 100
air_eps = 1.0
core_eps = 4.0  # 内层
clad_eps = 2.25  # 包层

# s_len = 500*2
s_len = 100*2
theta = s_len/2/bend_r  # unit: rad


edge_x = s_len*0.1
edge_y = edge_x

cell_x = s_len+edge_x*2
cell_y = (bend_r+separation/2+width/2+edge_y)*2
cell = mp.Vector3(cell_x, cell_y, 0)

geometry = [mp.Block(mp.Vector3(mp.inf, mp.inf, mp.inf),
                     center=mp.Vector3(), material=mp.Medium(epsilon=clad_eps))]
geometry.append(mp.Block(mp.Vector3(s_len, width, mp.inf), center=mp.Vector3(0,
(width+separation)/2), material=mp.Medium(epsilon=core_eps)))
geometry.append(mp.Cylinder(radius=bend_r+width/2, center=mp.Vector3(0, -
(separation/2+width/2+bend_r)), material=mp.Medium(epsilon=core_eps)))
geometry.append(mp.Cylinder(radius=bend_r-width/2, center=mp.Vector3(0, -
(separation/2+width/2+bend_r)), material=mp.Medium(epsilon=clad_eps)))
geometry.append(mp.Block(mp.Vector3(2*bend_r/np.tan(theta),bend_r*2),center=mp.V
ector3(bend_r/np.sin(theta),-(separation/2+width/2+bend_r)),
e1=mp.Vector3(np.cos(theta),-np.sin(theta)),
e2=mp.Vector3(np.sin(theta),np.cos(theta)),
material=mp.Medium(epsilon=clad_eps)))
geometry.append(mp.Block(mp.Vector3(2*bend_r/np.tan(theta),bend_r*2),center=mp.V
ector3(-bend_r/np.sin(theta),-(separation/2+width/2+bend_r)), e1=mp.Vector3(-
np.cos(theta),-np.sin(theta)), e2=mp.Vector3(-np.sin(theta),np.cos(theta)),
material=mp.Medium(epsilon=clad_eps)))


# monitoring points
mpt_straight_right = mp.Vector3(s_len/2-1,(separation+width)/2)
mpt_straight_center = mp.Vector3(0,(separation+width)/2)
mpt_bend_center = mp.Vector3(0,-(separation/2+width/2))
mpt_bend_left = mp.Vector3(-bend_r*np.sin(theta),-(bend_r*(1-np.cos(theta)) +
separation/2 + width/2))
mpt_bend_right = mp.Vector3(bend_r*np.sin(theta),-(bend_r*(1-np.cos(theta)) +
separation/2 + width/2))
mpt_vacancy = (mpt_straight_right + mpt_bend_right)/2


srcfrq = 1/((1550*C.nano)/(scale_fac*resolution))  # 1.03
sources = [mp.Source(mp.ContinuousSource(frequency=srcfrq),
```

```
                     component=mp.Ey,
                     center=mp.Vector3(-s_len/2-1, (separation+width)/2, 0))]
pml_layers = [mp.PML(1.0)]


sim = mp.Simulation(cell_size=cell,
                    boundary_layers=pml_layers,
                    geometry=geometry,
                    sources=sources,
                    resolution=resolution)


obs_size = mp.Vector3(s_len,width)
obs_center = mp.Vector3(0,(separation+width)/2)
# sim.init_sim()
sim.use_output_directory('wvg-out')
# sim.run(mp.at_beginning(mp.output_epsilon),until=10)

# sim.run(mp.at_beginning(mp.in_volume(mp.Volume(center=mp.Vector3(0,0,0),
size=mp.Vector3(cell_x,cell_y,0)), mp.output_png(mp.output_epsilon,"-Zc
dkbluered"))),
sim.run(mp.at_beginning(mp.in_volume(mp.Volume(center=mp.Vector3(),
size=mp.Vector3(cell_x, cell_y, 0)), mp.output_epsilon)),
       mp.to_appended("ey", mp.at_every(200.0,
mp.in_volume(mp.Volume(center=mp.Vector3(), size=cell), mp.output_efield_y))),
       mp.to_appended("dpwr", mp.at_every(200.0,
mp.in_volume(mp.Volume(center=mp.Vector3(), size=cell), mp.output_dpwr))),
        mp.to_appended("ey_straight_right", mp.at_every(1,
mp.in_point(mpt_straight_right, mp.output_efield_y))),
        mp.to_appended("ey_straight_center", mp.at_every(1,
mp.in_point(mpt_straight_center, mp.output_efield_y))),
        mp.to_appended("ey_bend_center", mp.at_every(1,
mp.in_point(mpt_bend_center, mp.output_efield_y))),
        mp.to_appended("ey_bend_left", mp.at_every(1, mp.in_point(mpt_bend_left,
mp.output_efield_y))),
        mp.to_appended("ey_bend_right", mp.at_every(1,
mp.in_point(mpt_bend_right, mp.output_efield_y))),
        mp.to_appended("ey_vacancy", mp.at_every(1, mp.in_point(mpt_vacancy,
mp.output_efield_y))),
        mp.to_appended("dpwr_straight_right", mp.at_every(1,
mp.in_point(mpt_straight_right, mp.output_dpwr))),
        mp.to_appended("dpwr_straight_center", mp.at_every(1,
mp.in_point(mpt_straight_center, mp.output_dpwr))),
        mp.to_appended("dpwr_bend_center", mp.at_every(1,
mp.in_point(mpt_bend_center, mp.output_dpwr))),
        mp.to_appended("dpwr_bend_left", mp.at_every(1,
mp.in_point(mpt_bend_left, mp.output_dpwr))),
        mp.to_appended("dpwr_bend_right", mp.at_every(1,
mp.in_point(mpt_bend_right, mp.output_dpwr))),
        mp.to_appended("dpwr_vacancy", mp.at_every(1, mp.in_point(mpt_vacancy,
mp.output_dpwr))),
        until=500)


obs_size = mp.Vector3(s_len,width)
obs_center = mp.Vector3(0,(separation+width)/2)
ez_data = sim.get_array(center=obs_center, size=obs_size, component=mp.Ey)
```

```
eps_data = sim.get_array(center=obs_center, size=obs_size,
component=mp.Dielectric)
eps_data.shape
plt.figure()
plt.imshow(eps_data.transpose(), interpolation='spline36', cmap='binary')
plt.imshow(ez_data.transpose(), interpolation='spline36', cmap='RdBu',
alpha=0.9)
plt.axis('off')
plt.show()
```

h5topng.py

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import argparse

parser = argparse.ArgumentParser(
    description="plot line figure based on one-dimension hdf5-format file ")
parser.add_argument('-h5', type=str, help='name of the hdf5-file')
parser.add_argument('-plot', type=str,
                    help='name of output figure name', default=None)

args = parser.parse_args()

h5file = args.h5
txtfile = args.h5[:-2]+'txt'
if args.plot == None:
    pngfile = args.h5[:-2]+'png'
else:
    pngfile = args.plot

os.system('/usr/bin/h5totxt' + ' ' + h5file + ' ' + '>' + ' ' + txtfile)

df = pd.read_csv(txtfile,header=None)
os.system('rm '+txtfile)
times = np.arange(df.size)*0.1
init_time = 0
# init_time = np.where(df!=0)[0][0]
# print(init_time)
plt.plot(times[init_time:],df.iloc[:, 0][init_time:],linewidth=0.2)
plt.xlabel('time/s')
plt.title(pngfile)
plt.title(pngfile)
# plt.show()
plt.savefig(pngfile)
```