计算物理期末作业

祝茗 2024202020022

Final Project

1. Write a program to solve the time-dependent partial differential equations for classical or quantum physics, i.e., Time-Dependent Maxwell Equation (TDME) or Time-Dependent Schrödinger Equation (TDSE). You can choose either Trotter-Suzuki Algorithm or Chebyshev Polynomial Algorithm for the implementation of the time-evolution operator, and consider only the solution in two-dimension. More details about the numerical methods can be found in the attached TDME.pdf and TDSE.pdf.

2. Use the program to simulate the propagation of a Gaussian wave package in two-dimension.

(a) For TDME, the preparation of the initial package is described in the TDME_Appendix.pdf. Reproduce the results shown in Fig. A-2 using the same parameters indicated there. For the initial wave package prepared in Eq. 29, take account the modes up to $m = n = 50$.

(b) For TDSE, the initial Gaussian wave package with vector k along x direction can be constructed as

$$\Psi(x, y) = \frac{1}{A} \exp\left[-\frac{x^2}{4\sigma_x^2} - \frac{y^2}{4\sigma_y^2} + ikx\right], \tag{1}$$

where A is a normalization factor, σ_x and σ_y are the width of the package. Simulate splitting of a Gaussian wave package through a small wave guide as the one shown in Fig. 1 of TDSE.pdf. The initial wave package is prepared on the left side, and all the parameters are given in page 18 and Fig. 1.

我计算了电磁场的随时演化。

In [1]:

```python
# 导入必要的库
from matplotlib import pyplot as plt
import numpy as np


# 设置参数
Lx = 18
Ly = 12
dx = 0.1
dy = 0.1
x0 = 3.05
y0 = 6.0
c = 1
sigmax = 2.75
sigmay = 2.0
k = 5
dt = 0.01
```

初始波包可以由以下公式表示

$$f(x,y,t)=\sin(k(x-x_0-ct))\exp[-((x-x_0-ct)/\sigma_x)^{10}-((y-y_0)/\sigma_y)^2]$$

$$a_{nm}=\frac{4}{\omega L_xL_y}\int_0^{L_x}\mathrm{d}x\int_0^{L_y}\mathrm{d}y\sin(k_nx)\sin(k_my)\left.\frac{\partial}{\partial t}f(x,y,t)\right|_{t=0}$$

$$b_{nm}=\frac{4}{L_xL_y}\int_0^{L_x}\mathrm{d}x\int_0^{L_y}\mathrm{d}y\sin(k_nx)\sin(k_my)\left.f(x,y,t)\right|_{t=0}$$

$$E_z(x,y,t)=\sum_{n,m}\sin(k_nx)\sin(k_my)[a_{nm}\sin(\omega t)+b_{nm}\cos(\omega t)]$$

$$H_x(x,y,t)=\sum_{n,m}\frac{ck_m}{\omega}\sin(k_nx)\cos(k_my)[a_{nm}\cos(\omega t)-b_{nm}\sin(\omega t)]$$

$$H_y(x,y,t)=-\sum_{n,m}\frac{ck_n}{\omega}\cos(k_nx)\sin(k_my)[a_{nm}\cos(\omega t)-b_{nm}\sin(\omega t)]$$

In [2]:

```
# 生成格点
x_even = np.arange(dx/2, Lx, dx)
y_even = np.arange(dy/2, Ly, dy)
x_odd = np.arange(0, Lx+dx, dx)
y_odd = np.arange(0, Ly+dy, dy)

X_mn, Y_mn = np.meshgrid(x_odd, y_odd, indexing='ij')
X_Ez, Y_Ez = np.meshgrid(x_even, y_even, indexing='ij')
X_Hx, Y_Hx = np.meshgrid(x_even, y_odd, indexing='ij')
X_Hy, Y_Hy = np.meshgrid(x_odd, y_even, indexing='ij')

n = np.arange(1, 51, 1)
m = np.arange(1, 51, 1)
N, M = np.meshgrid(n, m, indexing='ij')

nx = int(Lx/dx)*2 + 1  # x 方向格点数
ny = int(Ly/dy)*2 + 1  # y 方向格点数

omega = c*np.sqrt((N*np.pi/Lx)**2+(M*np.pi/Ly)**2)


def f(x, y, t) -> np.ndarray:
    """初始波包

    Args:
        x: x坐标
        y: y坐标
        t: 时间

    Returns:
        初始波包
    """
    return np.sin(k*(x-x0-t))*np.exp(-((x-x0-t)/sigmax)**10-((y-y0)/sigmay)**2)
```

按照规则，交替把 $H^{(x)}, H^{(y)}$ 作用于 Ψ，更新电场与磁场

In [3]:

```
def update_H(Psi, dt):
    i_forward_Psi = np.zeros_like(Psi)
    i_forward_Psi[1:] = Psi[:-1]
    i_backward_Psi = np.zeros_like(Psi)
    i_backward_Psi[:-1] = Psi[1:]
```

```
    j_forward_Psi = np.zeros_like(Psi)
    j_forward_Psi[:, 1:] = Psi[:, :-1]
    j_backward_Psi = np.zeros_like(Psi)
    j_backward_Psi[:, :-1] = Psi[:, 1:]
    Psi[1::2, ::2] += -dt * (j_backward_Psi - j_forward_Psi)[1::2, ::2] / dx  #
    Psi[::2, 1::2] += dt * (i_backward_Psi - i_forward_Psi)[::2, 1::2] / dy  # H
    Psi[1, ::2] = Psi[-2, ::2] = Psi[::2, 1] = Psi[::2, -2] = 0


def update_E(Psi, dt):
    i_forward_Psi = np.zeros_like(Psi)
    i_forward_Psi[1:] = Psi[:-1]
    i_backward_Psi = np.zeros_like(Psi)
    i_backward_Psi[:-1] = Psi[1:]
    j_forward_Psi = np.zeros_like(Psi)
    j_forward_Psi[:, 1:] = Psi[:, :-1]
    j_backward_Psi = np.zeros_like(Psi)
    j_backward_Psi[:, :-1] = Psi[:, 1:]
    Psi[1::2, 1::2] += dt * (i_backward_Psi - i_forward_Psi)[1::2, 1::2] / dx -
    Psi[1, ::2] = Psi[-2, ::2] = Psi[::2, 1] = Psi[::2, -2] = 0
```

计算波包的电磁场

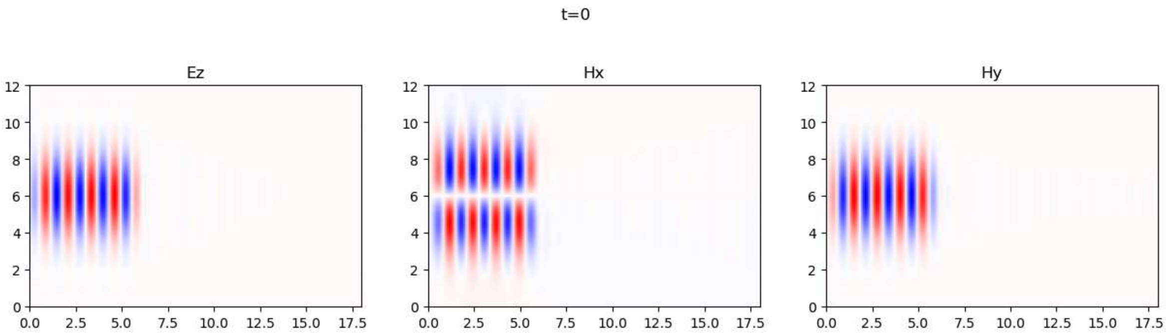
In [4]:
```
def get_anm(n, m) -> np.ndarray:
    """ a_nm
    """
    return np.sum(np.sin(n*np.pi/Lx*X_mn)*np.sin(m*np.pi/Ly*Y_mn)*((f(X_mn, Y_mn


def get_bnm(n, m) -> np.ndarray:
    """ b_nm
    """
    return np.sum(np.sin(n*np.pi/Lx*X_mn)*np.sin(m*np.pi/Ly*Y_mn)*f(X_mn, Y_mn,


# 向量化 a_nm, b_nm 函数
v_anm = np.vectorize(get_anm)
v_bnm = np.vectorize(get_bnm)


# 计算 a_nm, b_nm
anm = v_anm(N, M)
bnm = v_bnm(N, M)


def get_Ez(x, y, t) -> np.ndarray:
    """电场 z 分量
    """
    return np.sum(np.sin(N*np.pi/Lx*x)*np.sin(M*np.pi/Ly*y)*(anm*np.sin(omega*t)


def get_Hx(x, y, t) -> np.ndarray:
    """磁场 x 分量
    """
    return np.sum(c*M*np.pi/Ly/omega*np.sin(N*np.pi/Lx*x)*np.cos(M*np.pi/Ly*y)*(


def get_Hy(x, y, t) -> np.ndarray:
    """磁场 y 分量
```

```
    """
    return np.sum(-c*N*np.pi/Lx/omega*np.cos(N*np.pi/Lx*x)*np.sin(M*np.pi/Ly*y)*

# 向量化电磁场函数
v_Ez = np.vectorize(get_Ez)
v_Hx = np.vectorize(get_Hx)
v_Hy = np.vectorize(get_Hy)


# 制备初态电磁场
Ez_init = v_Ez(X_Ez, Y_Ez, t=0)
Hx_init = v_Hx(X_Hx, Y_Hx, t=0)
Hy_init = v_Hy(X_Hy, Y_Hy, t=0)

# 制备末态电磁场
Ez_final = v_Ez(X_Ez, Y_Ez, t=10)
Hx_final = v_Hx(X_Hx, Y_Hx, t=10)
Hy_final = v_Hy(X_Hy, Y_Hy, t=10)
```

绘制计算出的波包

In [6]:

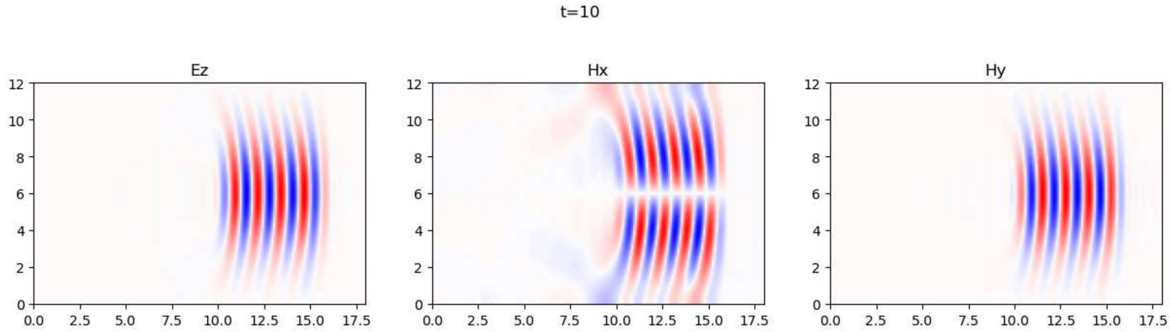
```
# 绘制 t=0 时刻的波包
fig, axes = plt.subplots(1, 3, figsize=(15, 4))

axes[0].imshow(Ez_init.T, cmap='bwr', origin='lower', extent=[0, 18, 0, 12])
axes[1].imshow(Hx_init.T, cmap='bwr', origin='lower', extent=[0, 18, 0, 12])
axes[2].imshow(Hy_init.T, cmap='bwr', origin='lower', extent=[0, 18, 0, 12])

axes[0].set_title('Ez')
axes[1].set_title('Hx')
axes[2].set_title('Hy')

fig.suptitle('t=0')

plt.show()
```



In [7]:

```
# 绘制 t=10 时刻的波包
fig, axes = plt.subplots(1, 3, figsize=(15, 4))

axes[0].imshow(Ez_final.T, cmap='bwr', origin='lower', extent=[0, 18, 0, 12])
axes[1].imshow(Hx_final.T, cmap='bwr', origin='lower', extent=[0, 18, 0, 12])
axes[2].imshow(Hy_final.T, cmap='bwr', origin='lower', extent=[0, 18, 0, 12])

axes[0].set_title('Ez')
axes[1].set_title('Hx')
axes[2].set_title('Hy')

fig.suptitle('t=10')
```

```
plt.show()
```

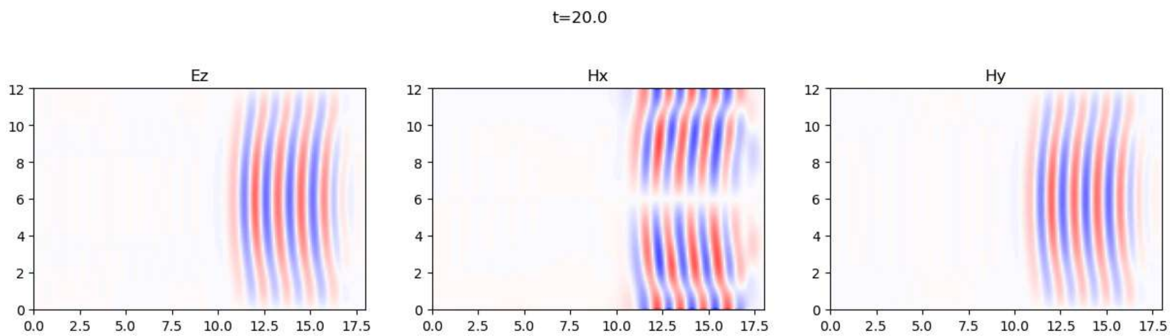


考虑波包的演化，按照 TM-Mode 的格点排列方式，把 E_z, H_x 和 H_y 置入 Ψ 中

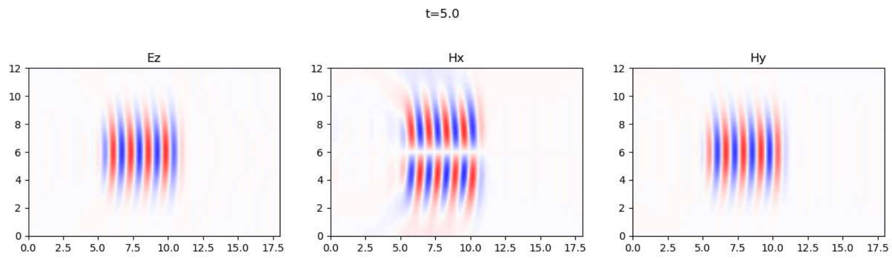
In [8]:
```
Psi = np.zeros((nx, ny))
Psi[1::2, 1::2] = Ez_init  # 偶数项是 E_z
Psi[1::2, ::2] = Hx_init  # i 偶 j 奇是 H_x
Psi[::2, 1::2] = Hy_init  # i 奇 j 偶是 H_y
```

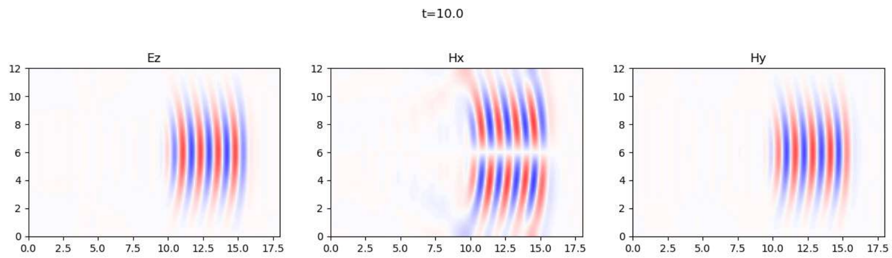
将 `update_H` 和 `update_E` 两个函数交替作用在 Ψ 上, 然后绘图

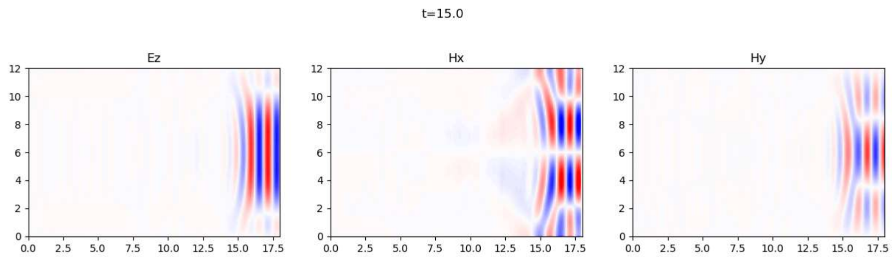
In [10]:
```
Psi_odd = np.copy(Psi)

fig, axes = plt.subplots(1, 3, figsize=(15, 4))

for n_frame in range(100):  # 生成 100 张图像
    for i in range(20):  # 每张时间间隔 dt*20 = 0.2
        update_H(Psi, dt)
        update_E(Psi, dt)

    axes[0].imshow(Psi[1::2, 1::2].T, cmap='bwr', origin='lower', extent=[0, 18,
    axes[1].imshow(Psi[1::2, ::2].T, cmap='bwr', origin='lower', extent=[0, 18,
    axes[2].imshow(Psi[::2, 1::2].T, cmap='bwr', origin='lower', extent=[0, 18,

    axes[0].set_title('Ez')
    axes[1].set_title('Hx')
    axes[2].set_title('Hy')

    fig.suptitle('t=%.1f' % (0.2*(n_frame+1)))

    fig.savefig(fname='./picpic/t=%.1f.png' % (0.2*(n_frame+1)))
```

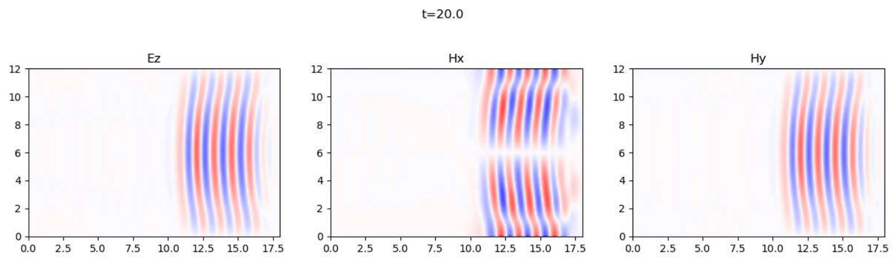


其中一些时刻的电磁场：









使用 `ffmpeg` 把输出的一系列时刻的 `png` 图片, 合成为 `output.mp4`