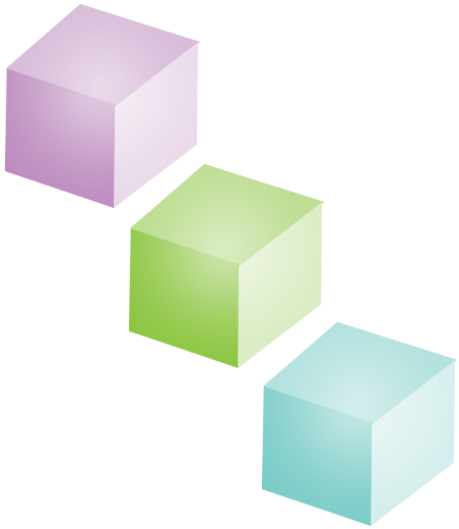
KEMP User Guide

Version 1.00

석명수
고려대학교 물리학과 나노광학연구실

Contents



▶Python-script 작성

1) FDTD 공간 설정
2) FDTD 경계 구조 설정
3) FDTD 물질 구조 설정
4) FDTD source 설정
5) FDTD 시간 루프 + DATA 취합

▶ 계산 실행

1) Script 실행(일반 PC)
2) MPI 설정(클러스터) 및 실행

1.1. FDTD 공간 설정

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import h5py as h5
4
5  import KEMP
6
7  nm = 1.e-9
8  dx, dy, dz = 10*nm, 10*nm, 10*nm
9  nx, ny, nz = 100, 100, 500 # 1 um, 1 um, 5 um
10 x = np.ones(nx, dtype=np.float32)*dx
11 y = np.ones(ny, dtype=np.float32)*dy
12 z = np.ones(nz, dtype=np.float32)*dz
13
14 grids = (x,y,z)
15
16 fdtd = KEMP.Basic_FDTD('3D', grids, dtype=np.float32, engine='nvidia_cuda', device_id=0, MPI_extension=False)
17
```

1.1.1. Python module 'KEMP' 로드

1.1.2. 공간 parameter 설정

1.1.3. 3D FDTD 생성

1.1.1. Python module 'KEMP' 로드

KEMP를 파이썬 스크립트에서 사용하고자 할 경우, 간단히 KEMP 모듈을 import 한다는 명령어 한 줄로 KEMP를 사용할 수 있게 된다.

1.1.2. 공간 parameter 설정

FDTD 공간을 설정하기 위해, FDTD 공간의 각 축(x,y,z)을 구성하는 길이 단위(dx, dy, dz)와 각 칸의 숫자(nx, ny, nz)를 설정함으로써 FDTD 공간을 구성한다.

1.1.3. FDTD 생성

앞에서 설정한 FDTD 공간 정보(grids)를 포함하여 여러 설정을 반영한 FDTD instance(fdtd)를 생성한다.

1.2. FDTD 경계 구조 설정

```
16  fdtd = KEMP.Basic_FDTD('3D', grids, dtype=np.float32, engine='nvidia_cuda', device_id=0, MPI_extension=False)
17
18  pml_apply = {'x':'', 'y':'', 'z':'+-'}
19
20  fdtd.apply_PML(pml_apply)
21
22  pbc_apply = {'x':True, 'y':False, 'z':False}
23
24  fdtd.apply_PBC(pbc_apply)
25
```

→ 1.2.1. PML 경계면 설정 및 적용

→ 1.2.2. PBC 경계면 설정 및 적용

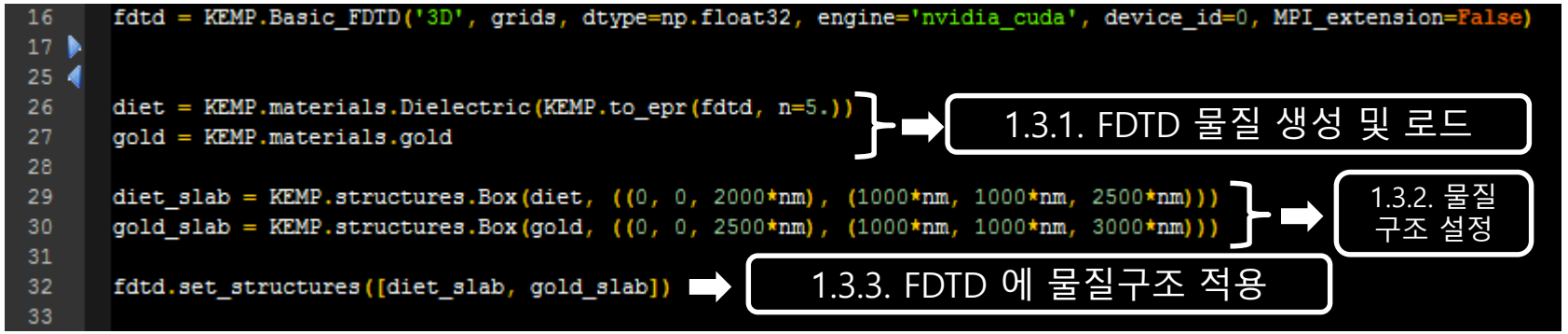
1.2.1. PML(Perfectly Matched Layer) 설정

PML은 FDTD에서 구현할 수 있는 무반사 흡수 특성을 갖는 경계면으로, 유한한 FDTD 공간을 실질적으로 열린 공간과 같이 계산할 수 있게 설정할 수 있다. KEMP에서 생성되는 3D-FDTD 공간의 세 축의 끝부분인 6개의 경계면에 대해 각각 PML을 적용할 수 있다.

1.2.2. PBC(Periodic Boundary Condition) 설정

끝없이 펼쳐지는 주기적인 물질 구조를 계산해내기 위해, KEMP에서는 FDTD 공간에 Periodicity를 부여하는 PBC를 3D-FDTD 공간의 세 축(x,y,z axis)에 대해 각각 적용할 수 있다. 필요할 경우 Bloch-Boundary Condition을 구현할 수 있다.

1.3. FDTD 물질 구조 설정



1.3.1. 물질 생성 및 로드

FDTD에서 구현 가능한 여러 물질을 생성하거나, 불러올 수 있다. KEMP에서는 Dielectric(ε), Dimagnetic(μ), Dielectromagnetic(ε, μ), dispersive meterials(Drude, CP, ...) 의 여러 가지 물질을 구현할 수 있도록 지원하고 있다.

1.3.2. 물질 구조 설정

1.3.1 에서 설정한 물질이 FDTD 공간에서 갖는 입체적 위치 및 구조 정보를 구현할 수 있다. KEMP에서는 2D 및 3D에서 여러 입체적 구조(다각형, 직육면체, 타원체, 기둥, 뿔 등)를 지원한다.

1.3.3. FDTD에 물질 구조 적용

설정한 물질 구조에 적용 순서를 부여하고, 순서에 따라 각 구조가 FDTD 공간에 적용되도록 한다.

1.4. FDTD Source wave 설정

```
16  fdtd = KEMP.Basic_FDTD('3D', grids, dtype=np.float32, engine='nvidia_cuda', device_id=0, MPI_extension=False)
17
33
34  from scipy.constants import c as c0
35  wavelength = 450*nm # Blue
36  angle_freq = 2*np.pi*c0/wavelength
37  delta_t    = KEMP.to_SI(fdtd, 'time', fdtd.dt)
38  period = wavelength/c0
39  period_step = int(period/delta_t)
40  tmax = period_step*30 # 30 periods
41
42  src = fdtd.apply_direct_source('ex', ((0,0,100),(-1,-1,100)))
43
```

1.4.1. Source 설정

1.4.2. FDTD Source 생성

1.4.1. Source 설정

FDTD 공간에 들어갈 incident wave의 주파수를 포함한 여러 값을 설정한다.

1.4.2. FDTD Source 생성

FDTD Source instance를 생성한다. Source instance에는 FDTD 공간에서 incident wave가 들어가는 위치 범위를 설정한다.

1.5. FDTD Time Loop + DATA 취합

```
16  fdtd = KEMP.Basic_FDTD('3D', grids, dtype=np.float32, engine='nvidia_cuda', device_id=0, MPI_extension=False)
17
43
44  ex_trs = np.zeros(tmax, dtype=np.float32)
45
46  for tstep in xrange(tmax):
47      src.set_source(np.sin(angle_freq*tstep*delta_t))
48
49      fdtd.updateE()
50      fdtd.updateH()
51
52      ex_trs[tstep] = fdtd.ex[:,:,100].mean()
53
54  f = h5.File('ex_data.h5', 'w')
55  f.create_dataset('ex_data', data=ex_trs)
56  f.close()
57
```

1.5.1. Time Loop

1.5.2. DATA 취합 및 파일 저장

1.5.1. FDTD Time Loop

FDTD 공간에 들어갈 incident wave의 주파수를 포함한 여러 값을 설정한다.

1.5.2. FDTD DATA 취합 및 저장

FDTD의 각 field는 numpy의 array처럼 동작한다. 따라서 FDTD의 각 field에서 DATA로 필요한 위치 영역을 지정하여 저장하는 것이 가능하다.

2.1. Script 실행(일반 PC)

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import h5py as h5
4
5   import KMES as KEMP
6
7   nm = 1.e-9
8   dx, dy, dz = 10*nm, 10*nm, 10*nm
9   nx, ny, nz = 100, 100, 500 # 1 um, 1 um, 5 um
10  x = np.ones(nx, dtype=np.float32)*dx
11  y = np.ones(ny, dtype=np.float32)*dy
12  z = np.ones(nz, dtype=np.float32)*dz
13
14  grids = (x,y,z)
15
16  fdtd = KEMP.Basic_FDTD('3D', grids, dtype=np.float32, engine='nvidia_cuda', device_id=[0,1,2], MPI_extension=False)
17
```

1.1.1. Python module 'KEMP' 로드

1.1.2. 공간 parameter 설정

1.1.3. 3D FDTD 생성

2.1.1. 계산 device 설정 (GPU 한정)

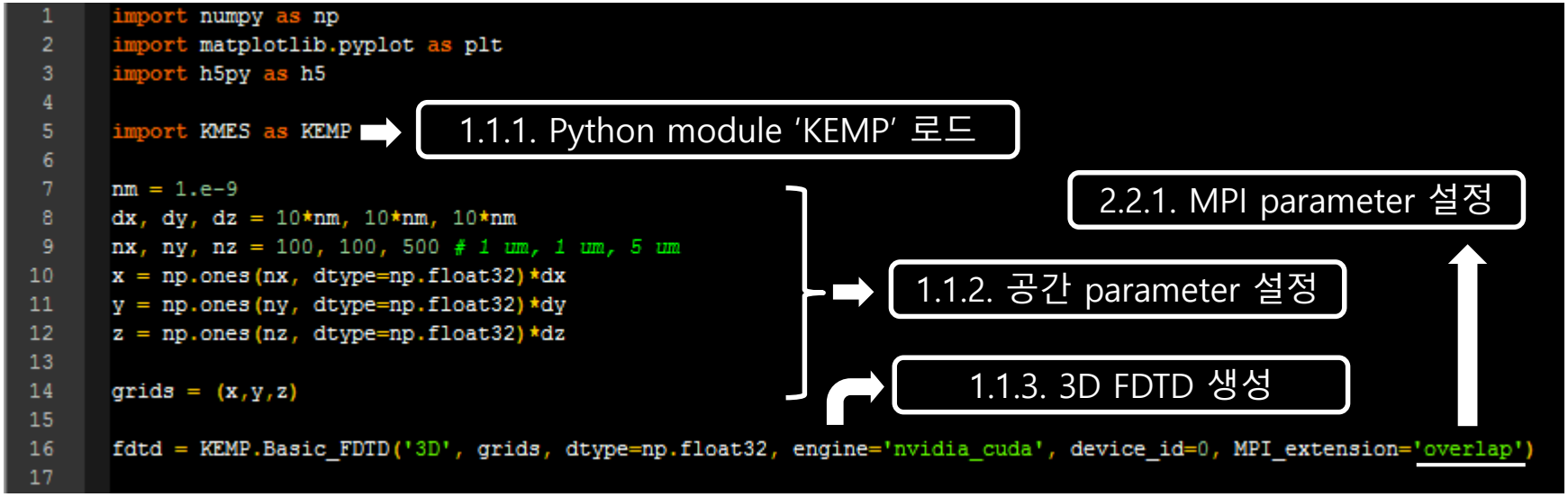
2.1.1. 계산 device 설정

FDTD를 계산할 GPU device를 설정한다. 단, CPU 계산의 경우(engine='intel_cpu') 이 옵션은 무시된다.
한 PC에 2대 이상의 GPU가 장착될 수 있으므로, 여러 device를 위 그림과 같이 설정할 수 있다. 이 경우 multi-devices 모드로 FDTD instance가 생성되며, 계산성능이 크게 향상된다. 단, FDTD 공간을 균등하게 분할하여 각 GPU에 할당하므로 각 GPU의 계산성능이 동일한 것을 권장한다.

2.1.2. Script 실행

Script 실행의 경우 보통의 python script 실행방법과 같다.
Python KEMP_example.py

2.2. MPI 설정(클러스터) 및 실행



2.2.1. MPI parameter 설정

FDTD instance를 생성할 때 초기값으로 주는 MPI_extension 항목을 기본값인 False 에서 다음 값들 중 하나로 바꾼다.

1. 'block' : FDTD 계산에서 사용할 MPI 통신 모드를 블로킹 통신으로 설정한다.
2. 'nonblock': FDTD 계산에서 사용할 MPI 통신 모드를 논블로킹 통신으로 설정한다. 1번의 블로킹 통신에 비해 계산속도가 향상된다.
3. 'overlap': FDTD 계산에서 통신에 필요한 경계면 부분을 CPU에 할당하고 통신 과정을 FDTD 공간 영역 계산과정과 병렬로 실행한다. MPI 병렬 계산에서 요구되는 통신 부하를 최소화할 수 있으므로 계산속도가 가장 높게 향상된다. 단 계산과정에 CPU 및 GPU를 전부 사용해야 하고, 상황에 따라 2번 모드에 비해 계산속도 향상이 제한적일 수 있다.

2.2.2. script 실행

Shell 환경에서 다음과 같이 명령어를 작성 및 실행한다.

mpirun –np 3 –host y201 y202 y203 python KEMP_example.py