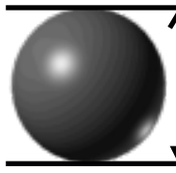
1. PM2.5について



排気フィルタ

吸気・オイルフィルタ

10^{-2}	cm	10	10	10	10	10	10	10
10^{-6}	μm	10	10	0.01	0.1	1	10	10
10^{-9}	nm	0.1	1	10	10	10	10	10
例		SO_2, NO_2分子 Fulleren分子		ディーゼル車のすす	Titania顔料 タバコの煙	土ぼこり		

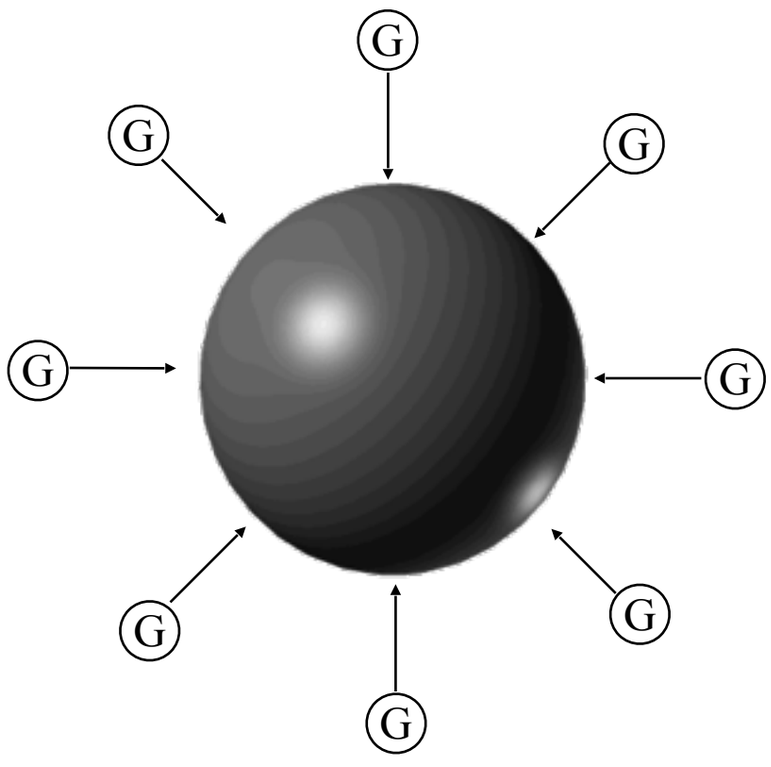
ブラウン運動

PM2.5: 空気1m^{3}あたり，粒径2.5μm以下の粒子が，何μg含まれるか

2. ブラウン運動とは何か

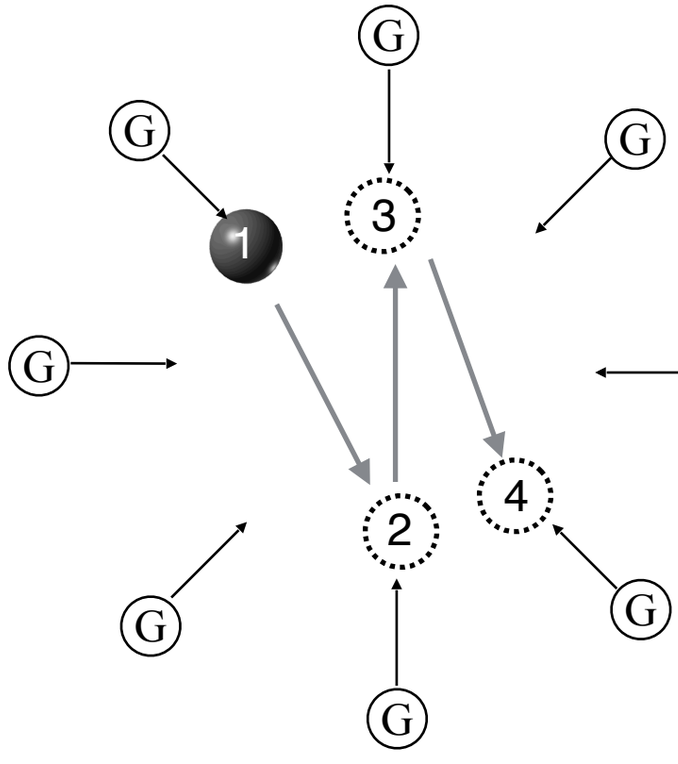
G 気体分子

粒子



粒子が
小さくなると

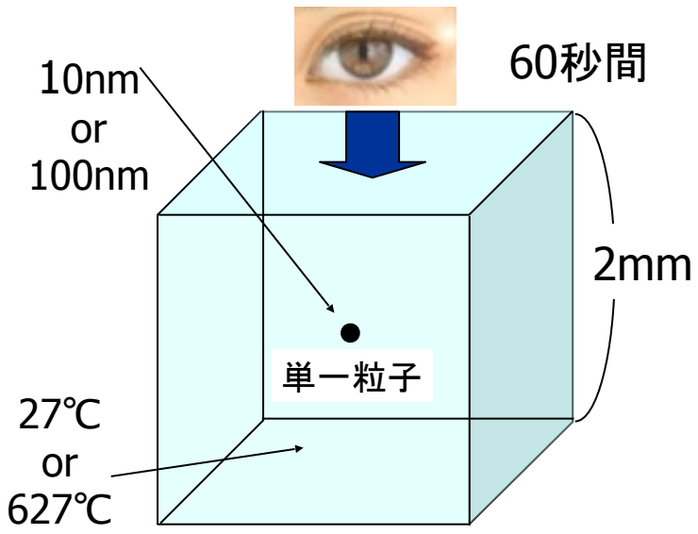
粒径<1μm



ブラウン運動!

ブラウン動力学法でシミュレーション可能

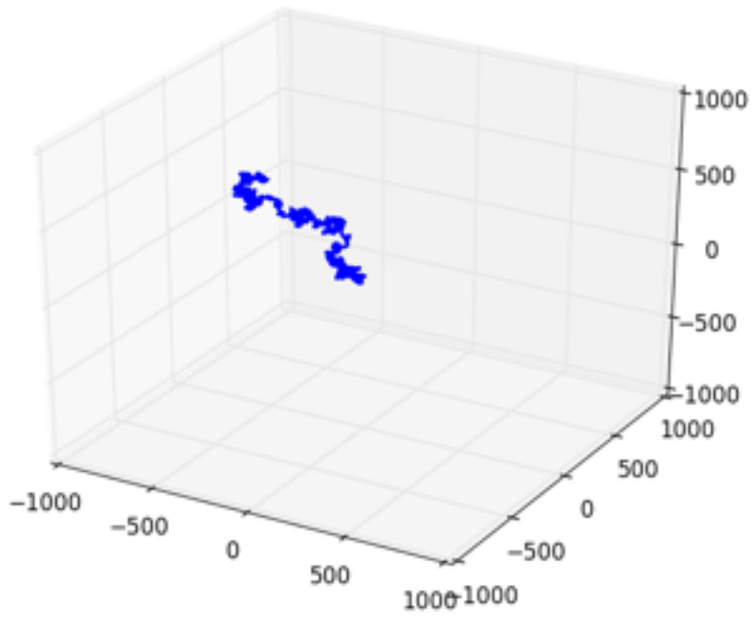
3. Brown動力学法について



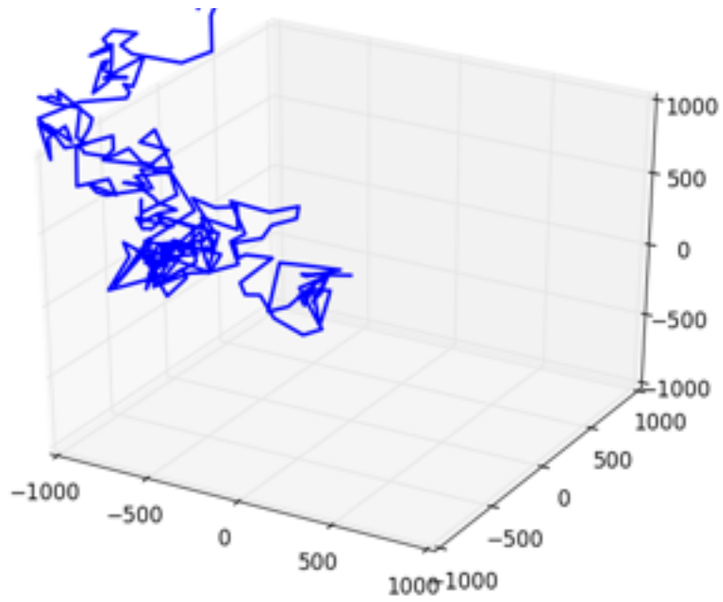
Brown運動の特徴(19世紀後半の実験)

- Brown運動は**乱雑**であり，その軌跡を描くと至る所で**微分不可能**な曲線となる.
- **粒子が小さい**ほど運動は活発である.
- **温度が高い**ほど運動が激しくなる.

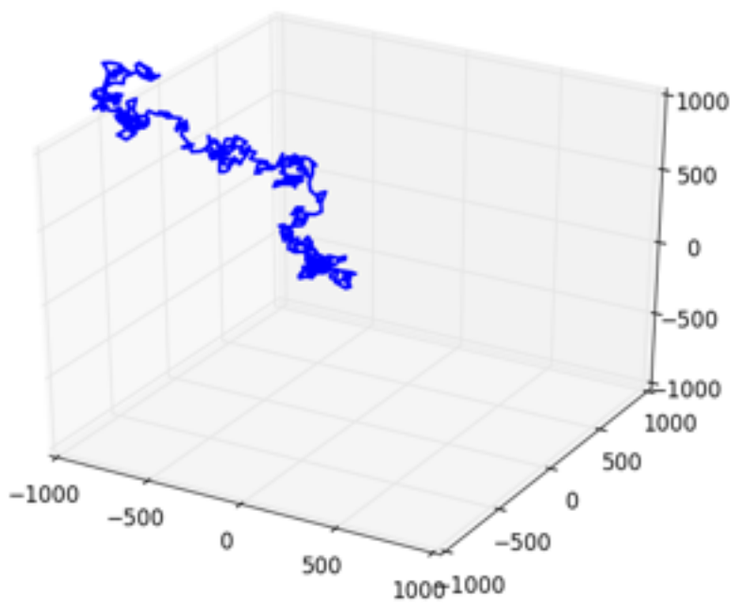
100 nm, 27 °C



10 nm, 27°C



100nm, 627°C



ブラウン運動の特徴を再現できる

4. f2pyについて

Fortranプログラム → f2py → pythonのmodule

```
f2py -c -m fsum sum.f90
```

```
subroutine sum(imax,val)

  implicit none
  integer::imax,i
  real*8::val
!f2py intent(in) i
!f2py intent(out) val

  val=0.d0
  do i=1,imax
    val=val+i
  end do

end subroutine sum
```

```
python
>>> import fsum
>>> fsum.sum(1000)
500500.0
```

Fortran言語を使う理由

- 数式の表現が読みやすい
- 既存の数値計算コードが多い
- 最適化しやすい → 高速
- f2pyがある

f2pyにより簡単にpython moduleが作れる

5. Brown動力学法のPythonモジュール作成

Fortranプログラムが確保した動的配列を
pythonスクリプトへどうやって渡すか？

```
f2py -c -m brown interface.f90 main.o
```

interface.f90

```
module part_param
  real*8::temp,dp
  real*8,allocatable,dimension(:)::px, py, pz
end module part_param
 ・ ・ ・
```

main.f90

```
 ・ ・ ・
open(10,file='param.data')
 read(10,*) nump
close(10)
 ・ ・ ・
allocate(px(nump),py(nump),pz(nump))
```

```
import brown
 ・ ・ ・
ptx=brown.part_param.px
pty=brown.part_param.py
ptz=brown.part_param.pz
```

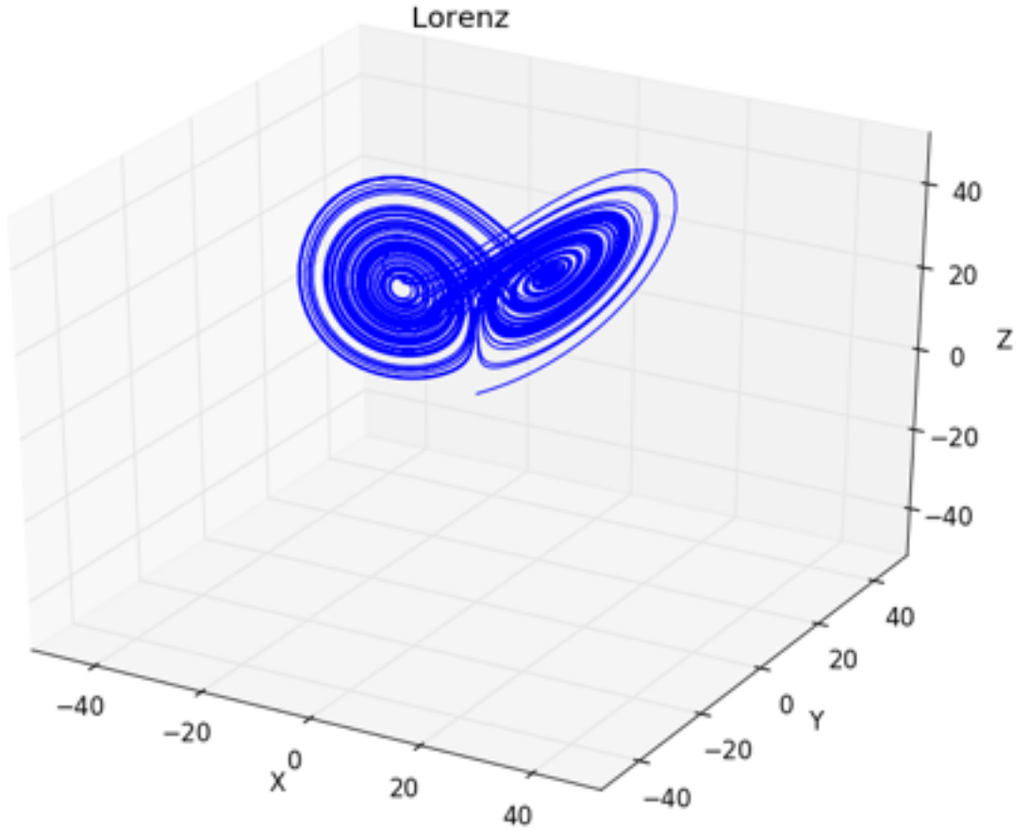
```
>>> type(ptx)
<class 'numpy.ndarray'>
>>> ptx.size
5000
```
(nump=5000の場合)

Fortranの動的配列はnumpyのarrayになる

6. matplotlibを用いたリアルタイム描画のやり方

例えば

Lorenzアトラクターの軌道を計算しながら
逐次描画させるにはどうすれば良いか？



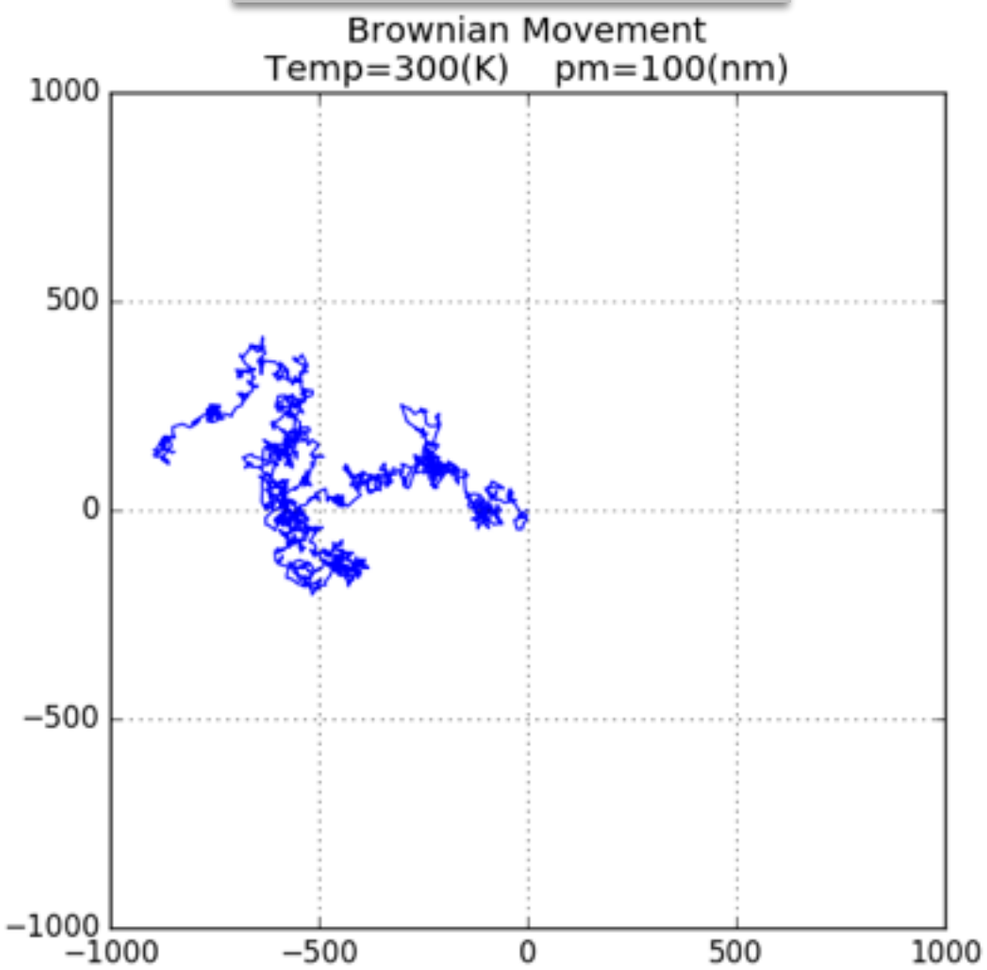
リアルタイム描画の方法

- ion, ioff関数
- pause関数
- event処理挿入
- timer関数
- animation関数

リアルタイム描画には複数の方法がある

7. matplotlibを用いたBrown運動のリアルタイム可視化

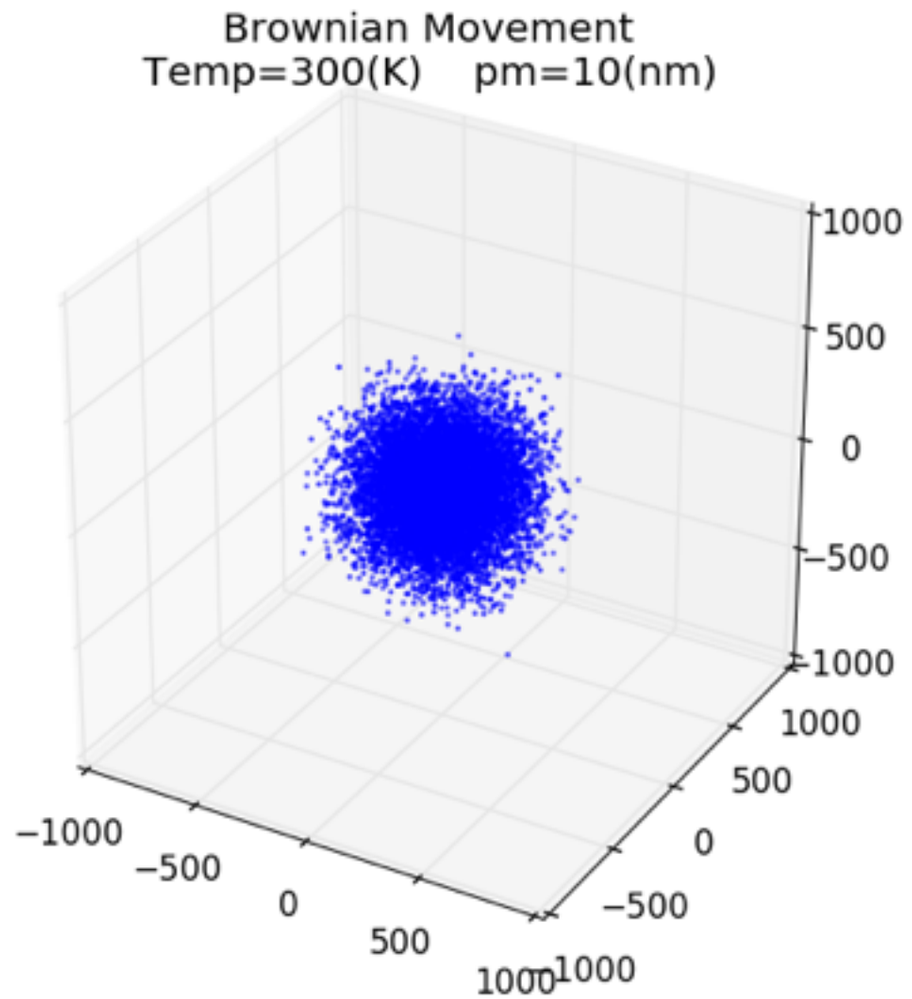
粒子1個の軌跡




```
line, = ax.plot([], [])
xdata, ydata = [],[]
. . .

pxyz=brown.pevolution(10000,0)
x, y = pxyz[0], pxyz[1]
xdata.append(x); ydata.append(y)
line.set_data(xdata, ydata)
ax.figure.canvas.draw()
```

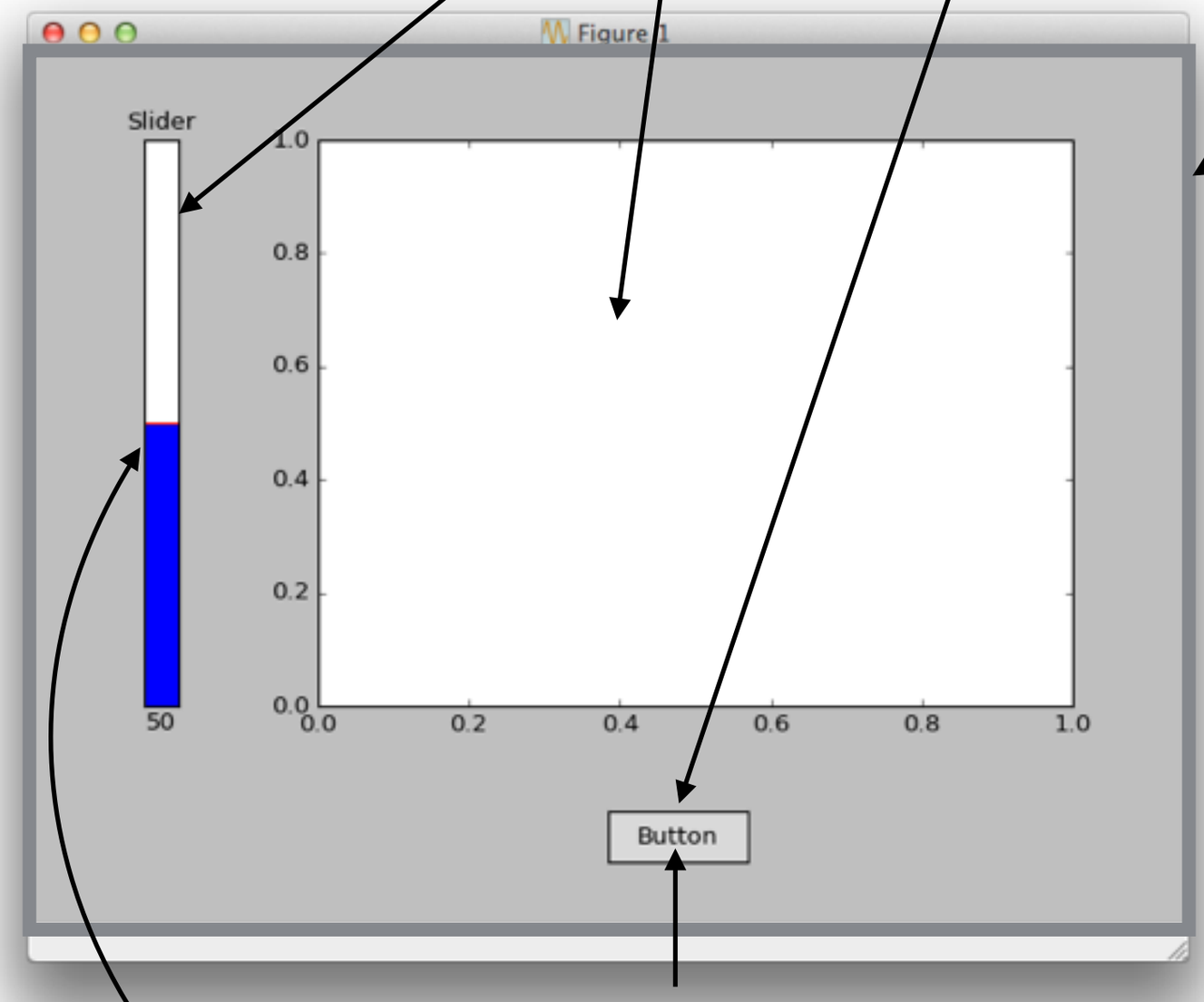
10^4個の粒子の位置




```
brown.pevolution(100,0)
ptx=brown.part_param.px
pty=brown.part_param.py
ptz=brown.part_param.pz
abc.remove()
abc=ax.scatter(ptx, pty, ptz)
```

8. matplotlibのイベント処理とwidget

fig.axisがグラフやwidgetの矩形領域を表現



四角の内側がfig.canvas

built-inイベントへの応答

```
fig.canvas.mpl_connect(
イベント名、応答関数)
```

‘button_press_event'

‘key_press_event'

・・・

```
fig.canvas.callbacks.process(
イベント名，データ)
```

でユーザ定義イベントの生成も可能

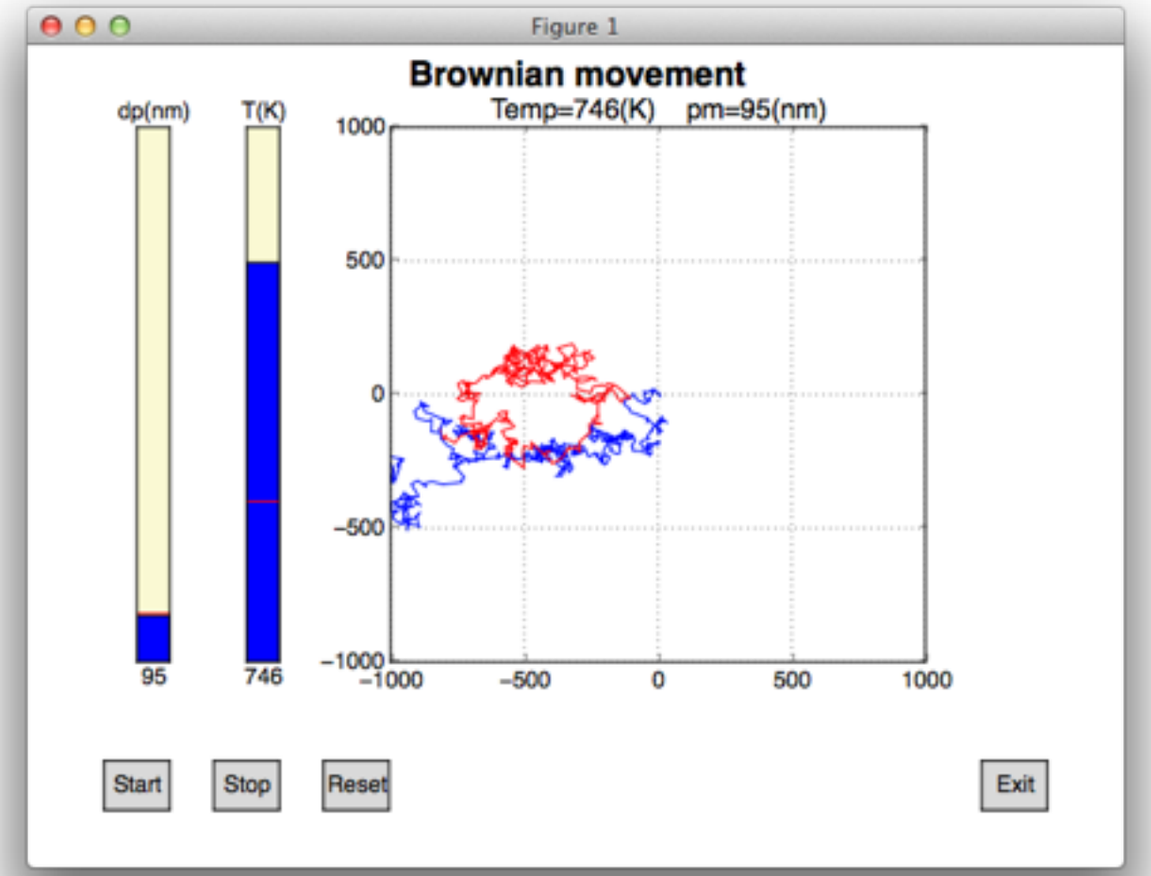
on_clicked関数で応答関数を指定

on_change関数で応答関数を指定

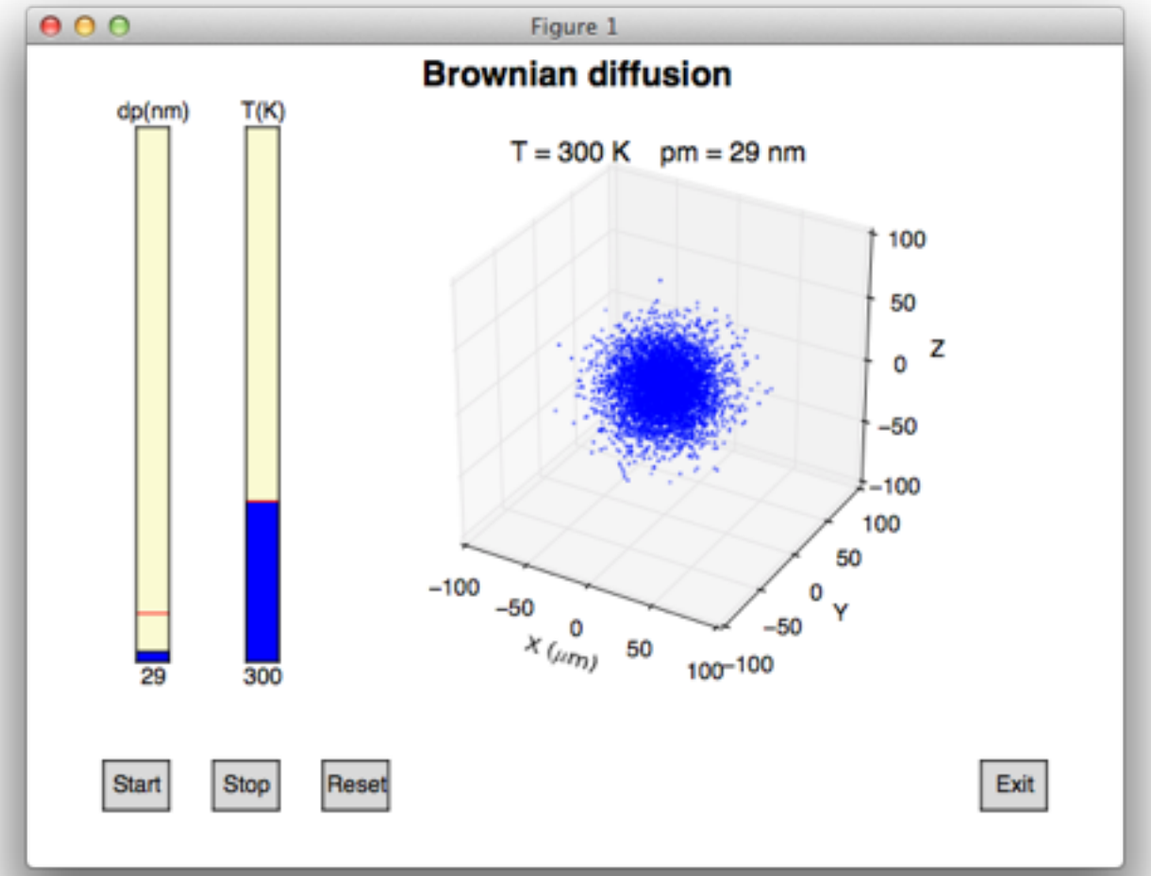
Built-in widgets Slider, Button, Check buttons, …

9. matplotlibを用いたGUIプログラム

粒子1個の軌跡



5x10^{3}個の粒子の位置



温度と粒径が変わると線の色が変わる

- ボタンクリックでstart & stopが可能
- Vertical Sliderはmatplotlibには含まれていない
- Resetで最初からやり直す
- Exitでプログラム終了