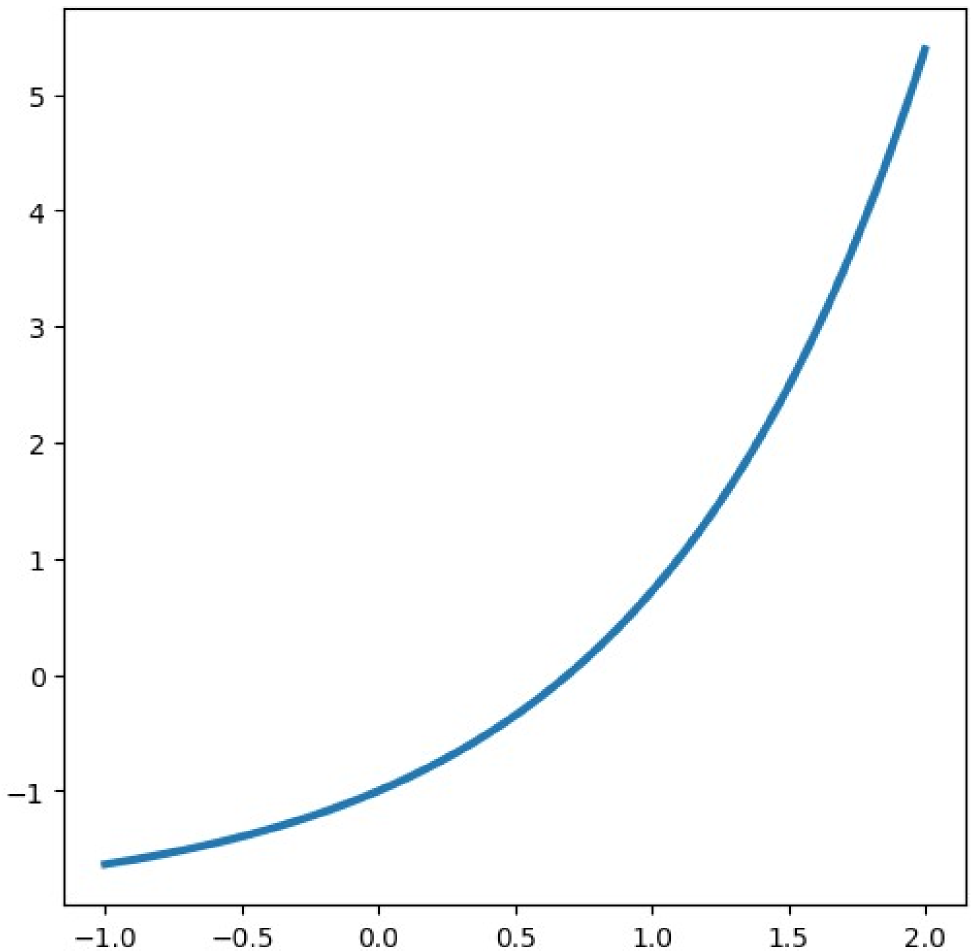
66310837

นายจิรัฐ ฟองดา

```python
import numpy as np
import numpy.linalg as la
import scipy.linalg as sla
import matplotlib.pyplot as plt

def f(x):
    return np.exp(x) - 2

def df(x):
    return np.exp(x)

x = np.linspace(-1, 2, 100)
plt.figure(figsize=(6,6))
plt.plot(x, f(x), lw=3)
```

```
[<matplotlib.lines.Line2D at 0x17dd2ad6930>]
```

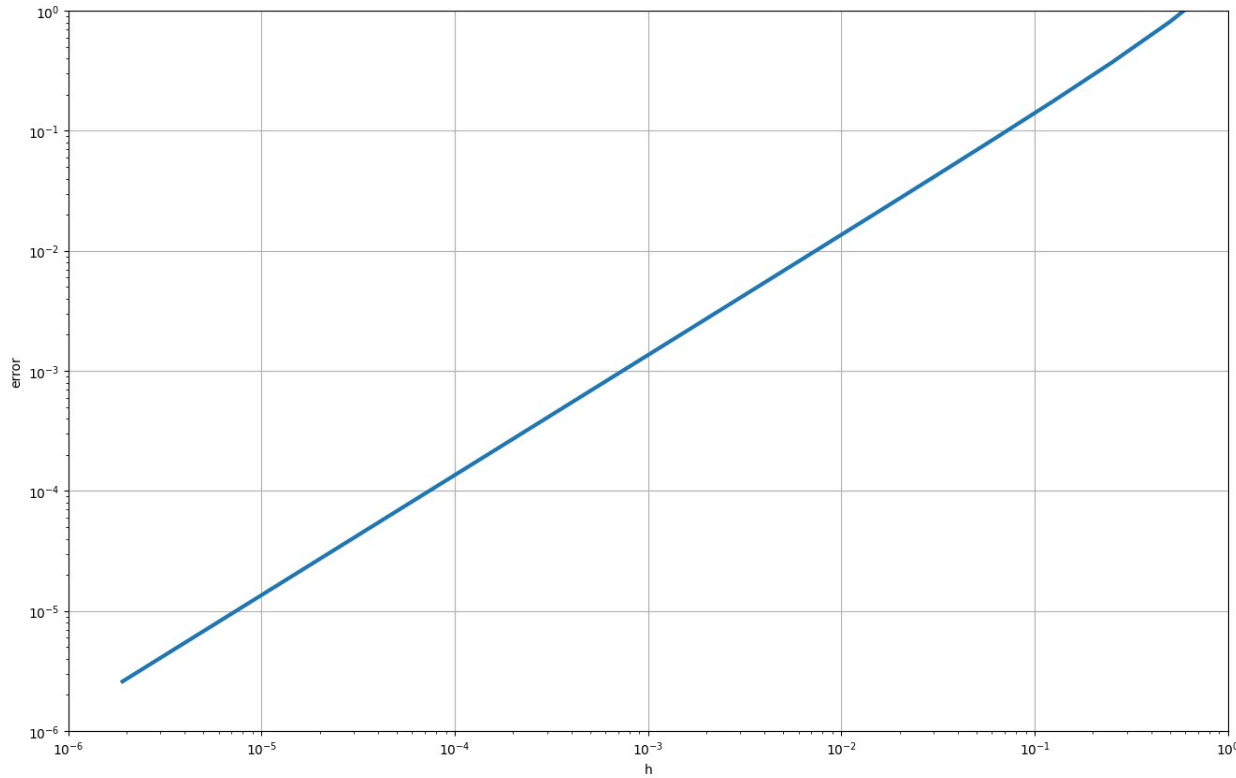



```
xhat = 1.0 # point where we want to find the approximation
h = 1.0 # initial perturbation
errors = []
hs = []

fval = f(xhat) # we only need to evaluate this once!

# in general, we don't have this value, but we will use it here to
visualize the error
dfexact = df(xhat)

for i in range(20):

    # one function evalution per each perturbation size
    dfapprox = ( f(xhat+h) - fval ) / h

    # get the error
    err = np.abs(dfexact - dfapprox)

    print(" %E \t %E " %(h, err) )
```

```
    hs.append(h)
    errors.append(err)

    h = h /2
```

```
 1.000000E+00    1.952492E+00
 5.000000E-01    8.085327E-01
 2.500000E-01    3.699627E-01
 1.250000E-01    1.771983E-01
 6.250000E-02    8.674402E-02
 3.125000E-02    4.291906E-02
 1.562500E-02    2.134762E-02
 7.812500E-03    1.064599E-02
 3.906250E-03    5.316064E-03
 1.953125E-03    2.656301E-03
 9.765625E-04    1.327718E-03
 4.882812E-04    6.637511E-04
 2.441406E-04    3.318485E-04
 1.220703E-04    1.659175E-04
 6.103516E-05    8.295707E-05
 3.051758E-05    4.147811E-05
 1.525879E-05    2.073897E-05
 7.629395E-06    1.036945E-05
 3.814697E-06    5.184779E-06
 1.907349E-06    2.592443E-06
```

```
plt.figure(figsize=(16,10))
plt.loglog(hs, errors, lw=3)
plt.xlabel('h')
plt.ylabel('error')
plt.xlim([1e-6,1])
plt.ylim([1e-6,1])
plt.grid()
```

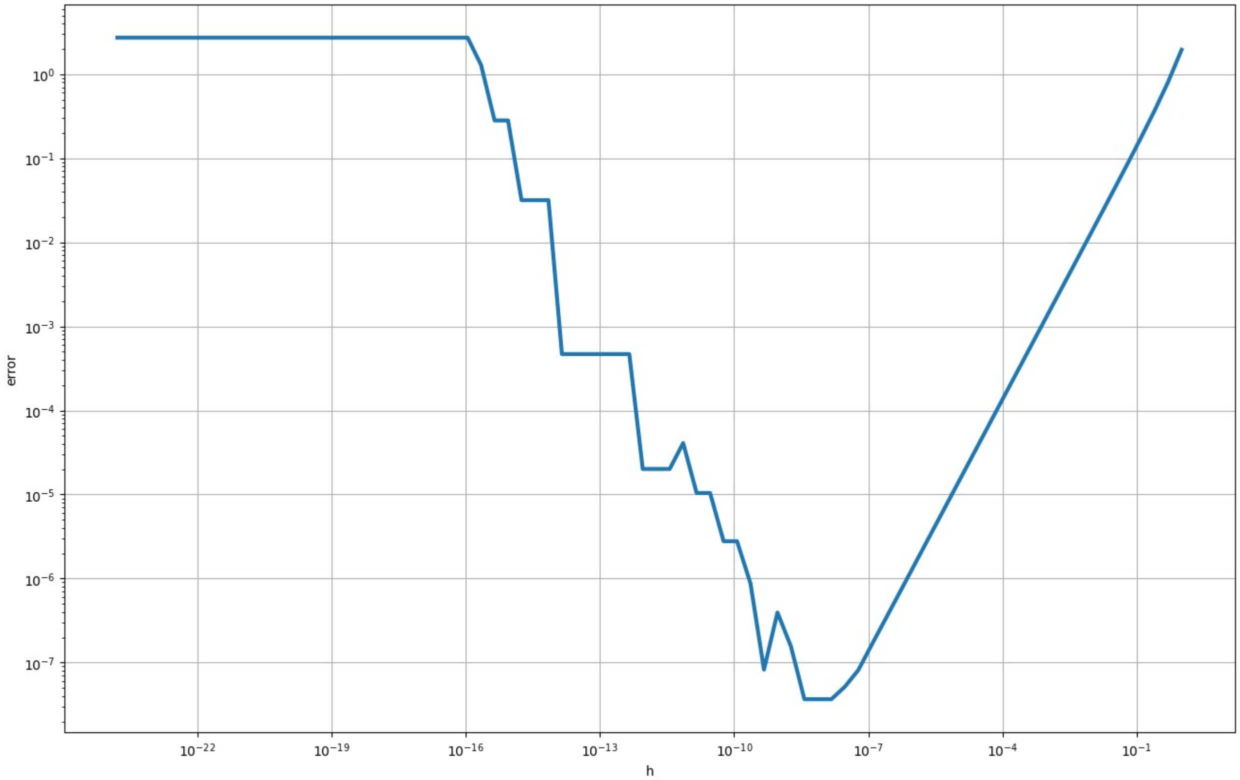



```
xhat = 1.0 # point where we want to find the approximation
h = 1.0 # initial perturbation
errors = []
hs = []

fval = f(xhat) # we only need to evaluate this once!

# in general, we don't have this value, but we will use it here to
visualize the error
dfexact = df(xhat)

for i in range(80):

    # one function evalution per each perturbation size
    dfapprox = ( f(xhat+h) - fval ) / h

    # get the error
    err = np.abs(dfexact - dfapprox)

    print(" %E \t %E " %(h, err) )

    hs.append(h)
    errors.append(err)

    h = h /2
```

```
1.000000E+00    1.952492E+00
5.000000E-01    8.085327E-01
2.500000E-01    3.699627E-01
1.250000E-01    1.771983E-01
6.250000E-02    8.674402E-02
3.125000E-02    4.291906E-02
1.562500E-02    2.134762E-02
7.812500E-03    1.064599E-02
3.906250E-03    5.316064E-03
1.953125E-03    2.656301E-03
9.765625E-04    1.327718E-03
4.882812E-04    6.637511E-04
2.441406E-04    3.318485E-04
1.220703E-04    1.659175E-04
6.103516E-05    8.295707E-05
3.051758E-05    4.147811E-05
1.525879E-05    2.073897E-05
7.629395E-06    1.036945E-05
3.814697E-06    5.184779E-06
1.907349E-06    2.592443E-06
9.536743E-07    1.296275E-06
4.768372E-07    6.485398E-07
2.384186E-07    3.253709E-07
1.192093E-07    1.633208E-07
5.960464E-08    8.136437E-08
2.980232E-08    5.156205E-08
1.490116E-08    3.666089E-08
7.450581E-09    3.666089E-08
3.725290E-09    3.666089E-08
1.862645E-09    1.558702E-07
9.313226E-10    3.942888E-07
4.656613E-10    8.254840E-08
2.328306E-10    8.711259E-07
1.164153E-10    2.778475E-06
5.820766E-11    2.778475E-06
2.910383E-11    1.040787E-05
1.455192E-11    1.040787E-05
7.275958E-12    4.092545E-05
3.637979E-12    2.010971E-05
1.818989E-12    2.010971E-05
9.094947E-13    2.010971E-05
4.547474E-13    4.681715E-04
2.273737E-13    4.681715E-04
1.136868E-13    4.681715E-04
5.684342E-14    4.681715E-04
2.842171E-14    4.681715E-04
1.421085E-14    4.681715E-04
7.105427E-15    3.171817E-02
3.552714E-15    3.171817E-02
1.776357E-15    3.171817E-02
```

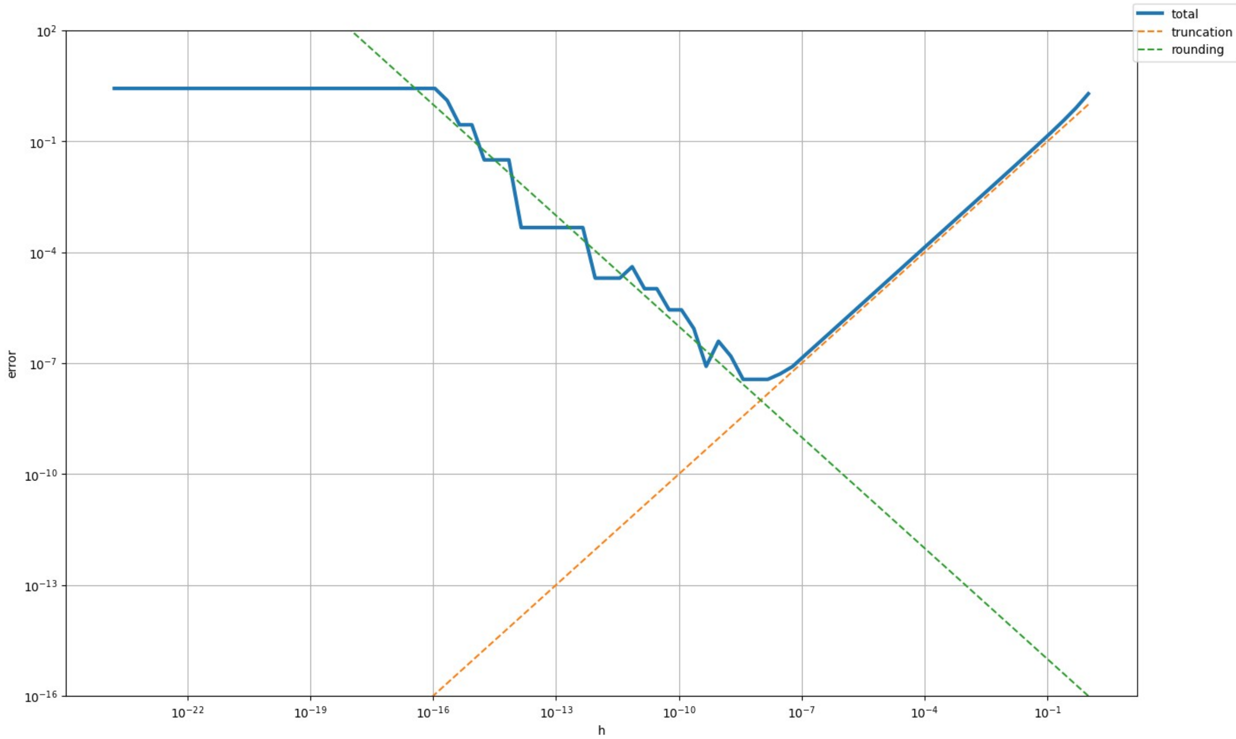
```
 8.881784E-16    2.817182E-01
 4.440892E-16    2.817182E-01
 2.220446E-16    1.281718E+00
 1.110223E-16    2.718282E+00
 5.551115E-17    2.718282E+00
 2.775558E-17    2.718282E+00
 1.387779E-17    2.718282E+00
 6.938894E-18    2.718282E+00
 3.469447E-18    2.718282E+00
 1.734723E-18    2.718282E+00
 8.673617E-19    2.718282E+00
 4.336809E-19    2.718282E+00
 2.168404E-19    2.718282E+00
 1.084202E-19    2.718282E+00
 5.421011E-20    2.718282E+00
 2.710505E-20    2.718282E+00
 1.355253E-20    2.718282E+00
 6.776264E-21    2.718282E+00
 3.388132E-21    2.718282E+00
 1.694066E-21    2.718282E+00
 8.470329E-22    2.718282E+00
 4.235165E-22    2.718282E+00
 2.117582E-22    2.718282E+00
 1.058791E-22    2.718282E+00
 5.293956E-23    2.718282E+00
 2.646978E-23    2.718282E+00
 1.323489E-23    2.718282E+00
 6.617445E-24    2.718282E+00
 3.308722E-24    2.718282E+00
 1.654361E-24    2.718282E+00
```

```
plt.figure(figsize=(16,10))
plt.loglog(hs, errors, lw=3)
plt.xlabel('h')
plt.ylabel('error')
plt.grid()
```




```
plt.figure(figsize=(16,10))
plt.loglog(hs, errors, lw=3, label='total')
plt.loglog(hs, np.array(hs), '--', label='truncation')
plt.loglog(hs, 1e-16/np.array(hs), '--', label='rounding')
plt.legend(bbox_to_anchor=(1.1, 1.05))
plt.xlabel('h')
plt.ylabel('error')
plt.grid()
plt.ylim(1e-16,1e2)
```

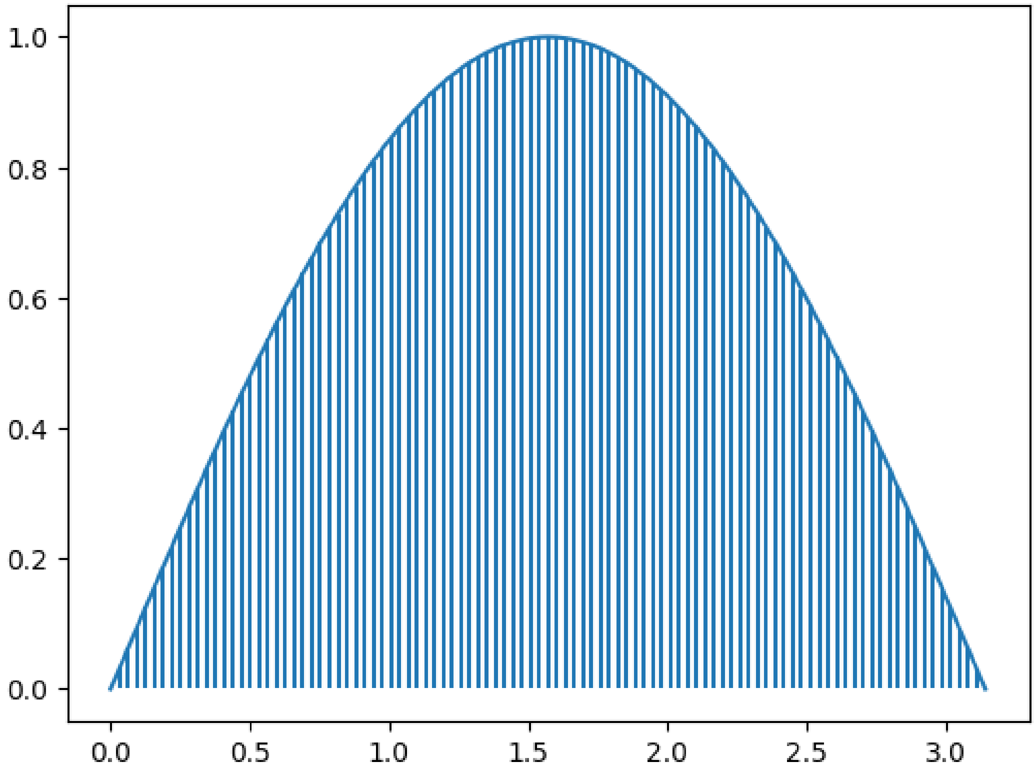
```
(1e-16, 100.0)
```



Riemanns Integral

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import collections as matcoll

a = 0
b = np.pi
n = 101
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)
f = np.sin(x)

lines = []
for i in range(len(x)):
    pair=[(x[i],0), (x[i], f[i])]
    lines.append(pair)

linecoll = matcoll.LineCollection(lines)
fig, ax = plt.subplots()
ax.add_collection(linecoll)

plt.plot(x,f)
```

```
[<matplotlib.lines.Line2D at 0x17dd2bbdca0>]
```




```
I_riemannL = h * sum(f[:n-1])
err_riemannL = 2 - I_riemannL

I_riemannR = h * sum(f[1:])
err_riemannR = 2 - I_riemannR

I_mid = h* sum(np.sin((x[:n-1] + x[1:])/2))
# x[:n-1] 0 1 2 ... 9
# x[1:]   1 2 3 ... 10
err_mid = 2 - I_mid

print(I_riemannL)
print(err_riemannL)

print(I_riemannR)
print(err_riemannR)

print(I_mid)
print(err_mid)
```

```
1.9998355038874436
0.00016449611255664226
1.9998355038874436
0.00016449611255664226
2.0000822490709864
-8.224907098641765e-05
```

Trapezoid Rule

```
a = 0
b = np.pi
n = 101
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)
f = np.sin(x)

I_trap = (h)*np.sum((f[:n-1]+f[1:])/2)

err_trap = 2 - I_trap

print(I_trap)
print(err_trap)
```

```
1.9998355038874436
0.0001644961125564226
```