Math 104B HW 2

Qingze Lan 3046380

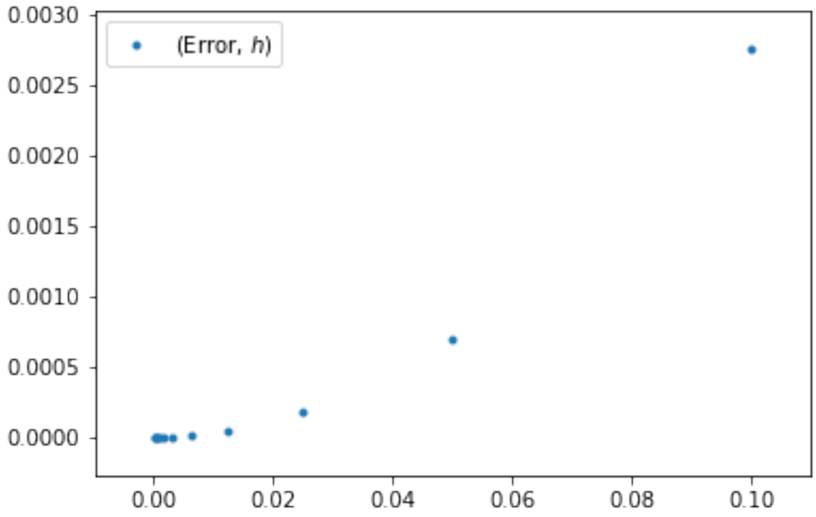
In [1]:
```
from numpy import exp, array, log, linspace, geomspace, log10
from pandas import DataFrame, options
import matplotlib.pyplot as plt
from scipy.stats import linregress
from math import pi
```

Problem 1

Let $f(x) = e^x$

(a) We'll compute the centered difference for $x_0 = 1/2$ and $h = 0.1/2^n$ for $n = 0, 1, \ldots, 10$. We'll verify too, the quadratic rate of convergence.

In [2]:
```
fun = lambda x: exp(x) # Our function
x_0 = 1/2 # x0
h_ar = array( [0.1/2**n for n in range(0,10+1)] ) # h

# Computing centered difference
# Input: f,x0,h
# Output: D_h^0 f(x0)

def cent_diff(f,x0,h):
    return ( f(x0 + h) - f(x0 - h) )/2/h

D = cent_diff(fun,x_0,h_ar)
error = abs(D-exp(1/2))

# Plot of error vs h
plt.scatter(h_ar, error,
            marker = '.', label = '(Error, $h$)')
plt.legend()
plt.ylim(-max(error)*0.1, max(error)*1.1)
plt.show()

# Data
data_1 = {'$h$':h_ar,'$D_h^0 f(1/2)$':D,'Error':error}
print("The actual value of the derivative is f'(1/2) = %.6f" % exp(
1/2))
DataFrame(data_1)
```




```
The actual value of the derivative is f'(1/2) = 1.648721
```

Out[2]:

	$D_h^0 f(1/2)$	h	**Error**
0	1.651471	0.100000	2.749243e-03
1	1.649408	0.050000	6.870531e-04
2	1.648893	0.025000	1.717472e-04
3	1.648764	0.012500	4.293579e-05
4	1.648732	0.006250	1.073388e-05
5	1.648724	0.003125	2.683467e-06
6	1.648722	0.001563	6.708665e-07
7	1.648721	0.000781	1.677166e-07
8	1.648721	0.000391	4.192926e-08
9	1.648721	0.000195	1.048235e-08
10	1.648721	0.000098	2.621470e-09

From the plot at the table we can verify the quadratic rate of convergence. We can confirm it by taking the $\log$ of the error and h, and finding the slope of a linear regression.

In [3]:

```
# Finding rate of convergence
l_error = log(error)
l_h = log(h_ar)
slope = linregress(l_h,l_error)[0]
print('We have found a slope = %.5f' % slope)
```

```
We have found a slope = 2.00002
```

(b)

We'll find now the optimal value of h considering round-off and truncation errors. If the difference is computed exactly, we'll have, for $|\delta_x|, |\delta_y| < \epsilon$:

$$\frac{f(x_0+h)(1+\delta_x) - f(x_0-h)(1-\delta_y)}{2h} = \frac{f(x_0+h) - f(x_0-h)}{2h} + r_h$$

with

$$|r_h| = \Big|\frac{f(x_0+h)\delta_x - f(x_0-h)\delta_y}{2h}\Big| \leq \big(|f(x_0+h)| + |f(x_0-h)|\big)\frac{\epsilon}{2h} \approx \frac{|f(x_0)|\epsilon}{h}$$

Now we can bound the total error with:

$$M_3 h^2 + \frac{|f(x_0)|\epsilon}{h}$$

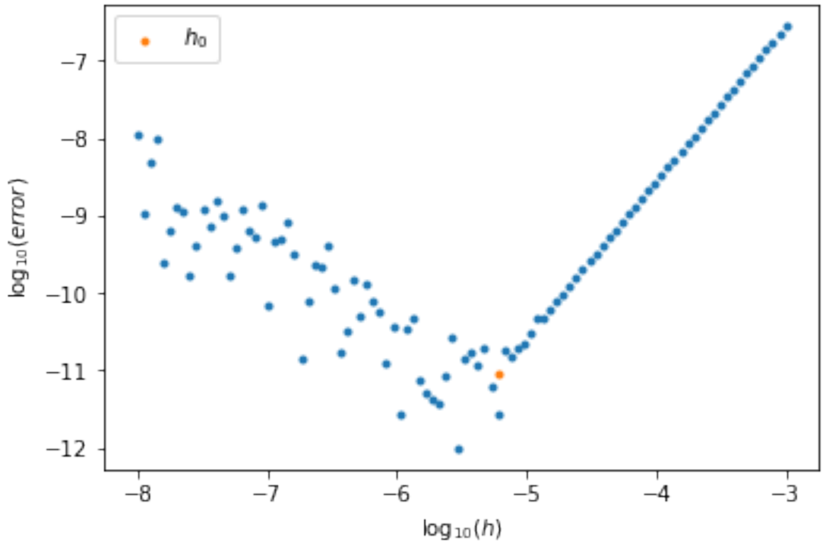
where $M_3 = |f'''(\eta)|/6$. This is minimized when $h = h_0$:

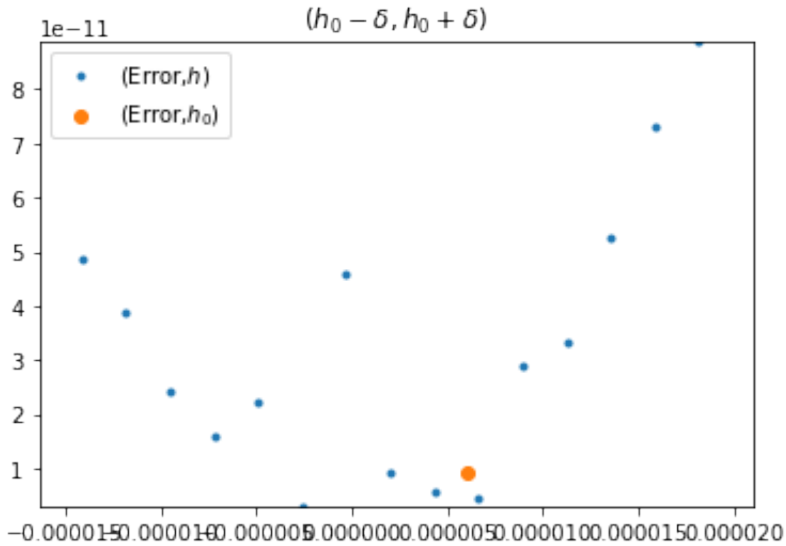
$$h_0 = \Big(\frac{3\epsilon|f(x_0)|}{M_3}\Big)^{1/3} = c\epsilon^{1/3} \sim \epsilon^{1/3}$$

We'll verify numerically that the error is minimum in this case.

In [4]:

```
# Verifying optimal value of h

eps = 2**(-52)
h_opt = (eps)**(1/3)

h_range_1 = geomspace(1e-8,1e-3,100)
D_21 = cent_diff(fun, 1/2, h_range_1)
error_21 = abs(D_21-exp(1/2))

# Taking logarithms for better visualization
l_h_1 = log10(h_range_1)
l_error_1 = log10(error_21)

h_range_2 = linspace(h_opt - h_opt/0.3, h_opt + h_opt/0.5,15)
D_22 = cent_diff(fun, 1/2, h_range_2)
error_22 = abs(D_22-exp(1/2))

# Large range of h
plt.scatter(l_h_1, l_error_1,marker = '.')
plt.scatter(log10(h_opt),
            log10(abs(cent_diff(fun,1/2,h_opt)-exp(1/2))),
           marker = '.',label = '$h_0$')
plt.legend()
plt.xlabel('$\log_{10}(h)$')
plt.ylabel('$\log_{10}(error)$')
plt.show()

# Plot near h_0
plt.title('$(h_0-\delta,h_0+\delta)$')
plt.scatter(h_range_2, error_22,
            marker = '.', label = '(Error,$h$)')
plt.scatter(h_opt,abs(cent_diff(fun,1/2,h_opt)-exp(1/2)),
           label = '(Error,$h_{0}$)')
plt.legend()
plt.xlim(h_range_2[0]*1.16,h_range_2[-1]*1.16)
plt.ylim(min(error_22),max(error_22))
plt.show()
```





In light of the above we see that we have the order of the minimum error when taking $h_0 = \epsilon^{1/3}$. Note that, the best error we can get is $\sim 10^{-11}$, which differs in 5 orders of magnitude from the machine precision. This is exactly $O(\epsilon^{2/3})$.

(c) We'll construct a fourth order approximation to $f'(1/2)$ by using Richardson extrapolation. We know that:

$$D_h^0 f(x_0) = f'(x_0) + c_2 h^2 + c_4 h^4 + \dots$$

By writing $D_{h/2}^0 f(x_0)$ and substracting we can get:

$$D_h^{ext} f(x_0) = \frac{4D_{h/2}^0 f(x_0) - D_h^0 f(x_0)}{3} = f'(x_0) + Ch^4 + \dots$$

Using this we can compute an approximation to $f'(1/2)$. We'll verify as in the previous case, the rate of convergence.

In [5]:
```
h_ar_r = array( [0.1/2**n for n in range(0,6+1)] ) # Avoid round_of
f
# Richardson extrapolation, centered difference
# Input: f,x0,h
# Output: D_h^ext f(x0)

def rich_cent_diff(f,x0,h):
    return 1/3*(4*cent_diff(f,x0,h/2)-cent_diff(f,x0,h))

# Results

D_r = rich_cent_diff(fun,x_0,h_ar_r)
error_r = abs(D_r-exp(1/2))

options.display.float_format = '{:,.15f}'.format
data_2 = {'$h$':h_ar_r,'$D_h^{ext} f(.5)$':D_r,'Error':error_r}
print("The actual value of the derivative is f'(1/2) = %.15f" % exp
(1/2))
DataFrame(data_2)
```

```
The actual value of the derivative is f'(1/2) = 1.648721270700128
```

Out[5]:

	$D_h^{ext} f(.5)$	h	Error
0	1.648720927114287	0.100000000000000	0.000000343585841
1	1.648721249230807	0.050000000000000	0.000000021469321
2	1.648721269358376	0.025000000000000	0.000000001341752
3	1.648721270616254	0.012500000000000	0.000000000083874
4	1.648721270694935	0.006250000000000	0.000000000005193
5	1.648721270699838	0.003125000000000	0.000000000000290
6	1.648721270700146	0.001562500000000	0.000000000000017

We can verify the rate of convergence in the same way as before

In [6]:
```
# Finding rate of convergence
l_error_r = log(error_r)
l_h_r = log(h_ar_r)
slope_r = linregress(l_h_r,l_error_r)[0]
print('We have found a slope = %.5f' % slope_r)
```

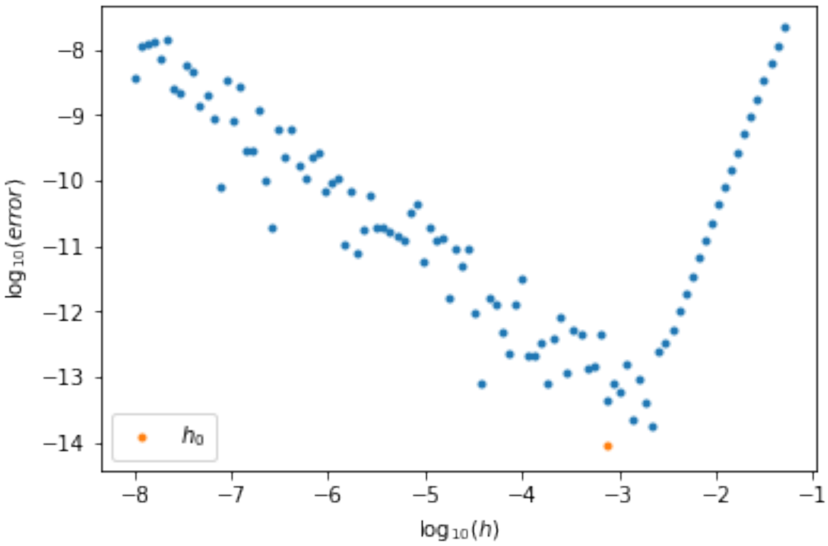
```
We have found a slope = 4.03879
```

We have found then that the error goes approximately as h^4 as we expected. In this case the total error is bounded by:

$$Ch^4 + \frac{|f(x_0)|\epsilon}{h}$$

Then we will find the minimum error when this two terms become comparable, this is when $h_0 \sim \epsilon^{1/5}$. This implies that the minimum error is approximately $\epsilon^{4/5}$. We can do a similar plot to verify it.

In [7]:
```
# Verifying optimal value of h

h_opt_r = (eps)**(1/5) # Close to optimal h

h_range_r = geomspace(1e-8,5e-2,100)
D_21_r = rich_cent_diff(fun, 1/2, h_range_r) # Centered differences
h_range
error_21_r = abs(D_21_r-exp(1/2)) # Error
# Taking logarithms for better visualization
l_h_r = log10(h_range_r)
l_error_1_r = log10(error_21_r)

# Large range of h
plt.scatter(l_h_r, l_error_1_r,marker = '.')
plt.scatter(log10(h_opt_r),
            log10(abs(rich_cent_diff(fun,1/2,h_opt_r)-exp(1/2))),
           marker = '.',label = '$h_0$')
plt.legend()
plt.xlabel('$\log_{10}(h)$')
plt.ylabel('$\log_{10}(error)$')
plt.show()
```



As seen in the plot, the minimum error we can make is $\sim 10^{-14}$.