In [6]:
```
"""
arj1
Arjun Srivastava
AMATH 301 B
"""

import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate
import pandas as pd
```

In [7]:
```
# Problem 1

mu = 3.39
std = 0.55

P = lambda x : (1 / np.sqrt(2 * np.pi * std**2)) * np.exp(-(x - mu)**2 / (2*st
d**2))

# a)

bounds = (2, 4)
Int, err = scipy.integrate.quad(P, bounds[0], bounds[1])
print("P:", Int)

"""
P = 0.8605569069798773
"""

# b)

LHR_err = []
for dx in range(2, 17):
    dx = 2**(-dx)
    x = np.arange(2, 4 + dx, dx)
    y = P(x)
    LHR = dx * np.sum(y[:-1])
    err = np.abs(LHR - Int)
    LHR_err.append(err)

# c)

RHR_err = []
for dx in range(2, 17):
    dx = 2**(-dx)
    x = np.arange(2, 4 + dx, dx)
    y = P(x)
    RHR = dx * np.sum(y[1:])
    err = np.abs(RHR - Int)
    RHR_err.append(err)

# d)

Trap_err = []
for dx in range(2, 17):
    dx = 2**(-dx)
    x = np.arange(2, 4 + dx, dx)
    y = P(x)
    trap = (dx / 2) * (y[0] + 2 * np.sum(y[1:-1]) + y[-1])
    err = np.abs(trap - Int)
    Trap_err.append(err)

# e)

data = {'Method': ['RHR', 'LHR', 'Trapezoidal']}
for k in range(len(RHR_err)):
    dx = 2**(-(k+2))
    data[dx] = [RHR_err[k], LHR_err[k], Trap_err[k]]

table = pd.DataFrame(data)
display(table)

# f)

"""
As expected, the trapezoidal rule yielded the smallest error while the RHR and
LHR yielded almost exactlty the same errors as
we decrease dx by a factor of 2^-1. By the final iteration (at 2^-16), the LHR
and RHR errors are at the 10^-6 magnitude while
the trapezoidal error is at 10^-11. We expect the trapezoidal error to be abou
t half of the LHR and RHR since it is based on an
average of those two measures, and the data supports this expectation. Since t
he LHR and RHR methods are first order, they're
error can be expressed to the magnitude of 10^k in the data. The trapezoidal m
ethod is second order, meaning it will be expressed
as (10^k)^2, which simplifies to 10^2k. Although the errors for the trapezoida
l method are not exactly half due to constant
factors and rounding approximations, various points in time confirm that this
 relationship between LHR/RHR's orders of accuracy
and that of the trapezoidal method is accurate.
"""
```

```
P: 0.8605569069798773
```

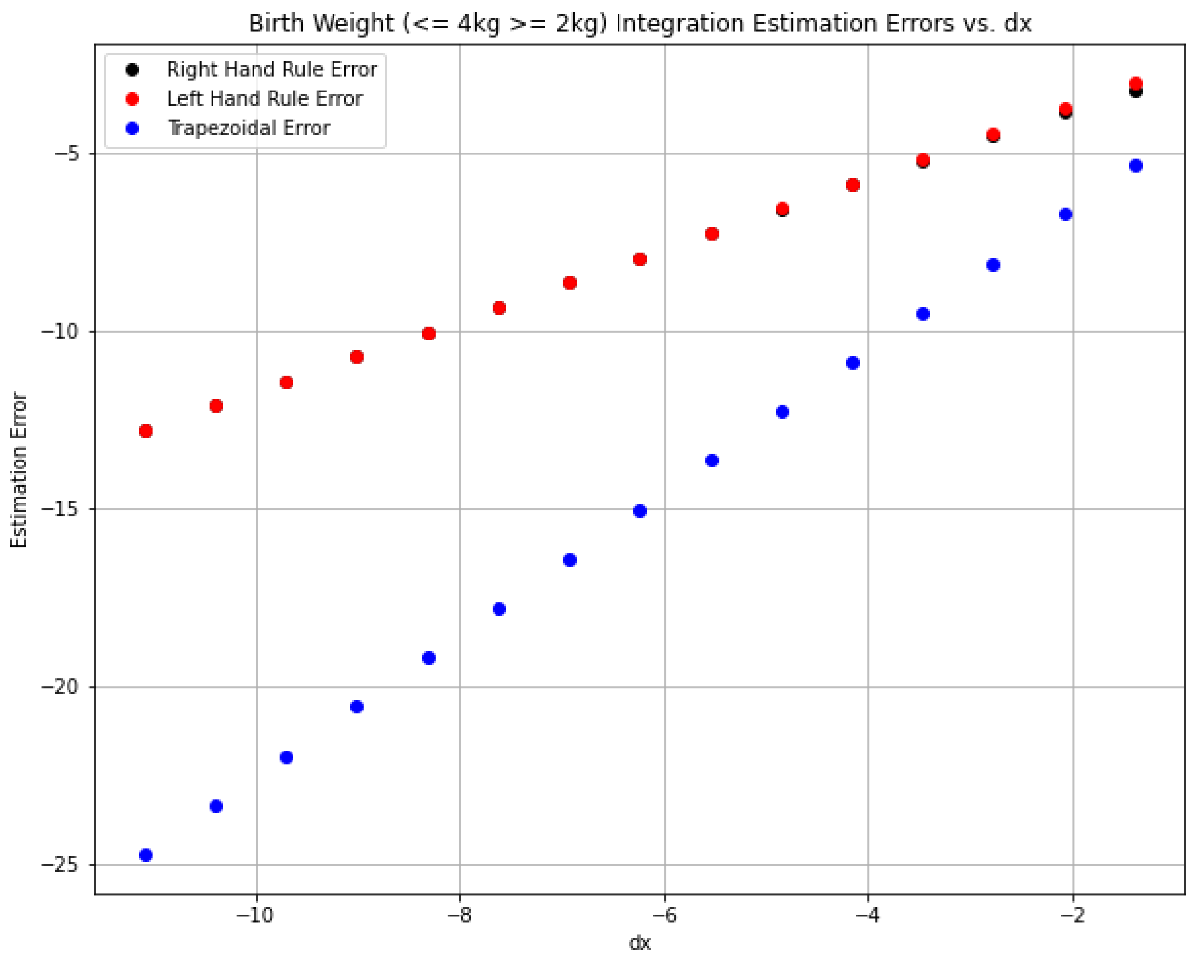
	Method	0.25	0.125	0.0625	0.03125	0.015625	0.0078125	0.00390625	0.00195
0	RHR	0.040450	0.021440	0.011022	0.005587	0.002812	0.001411	0.000707	3.535
1	LHR	0.050145	0.023857	0.011626	0.005738	0.002850	0.001420	0.000709	3.541
2	Trapezoidal	0.004848	0.001209	0.000302	0.000075	0.000019	0.000005	0.000001	2.948

Out[7]:

```
"\nAs expected, the trapezoidal rule yielded the smallest error while the RHR
and LHR yielded almost exactlty the same errors as\nwe decrease dx by a facto
r of 2^-1. By the final iteration (at 2^-16), the LHR and RHR errors are at t
he 10^-6 magnitude while\nthe trapezoidal error is at 10^-11. We expect the t
rapezoidal error to be about half of the LHR and RHR since it is based on an
\naverage of those two measures, and the data supports this expectation. Sinc
e the LHR and RHR methods are first order, they're\nerror can be expressed to
the magnitude of 10^k in the data. The trapezoidal method is second order, me
aning it will be expressed\nas (10^k)^2, which simplifies to 10^2k. Although
the errors for the trapezoidal method are not exactly half due to constant\nf
actors and rounding approximations, various points in time confirm that this
relationship between LHR/RHR's orders of accuracy\nand that of the trapezoida
l method is accurate.\n"
```

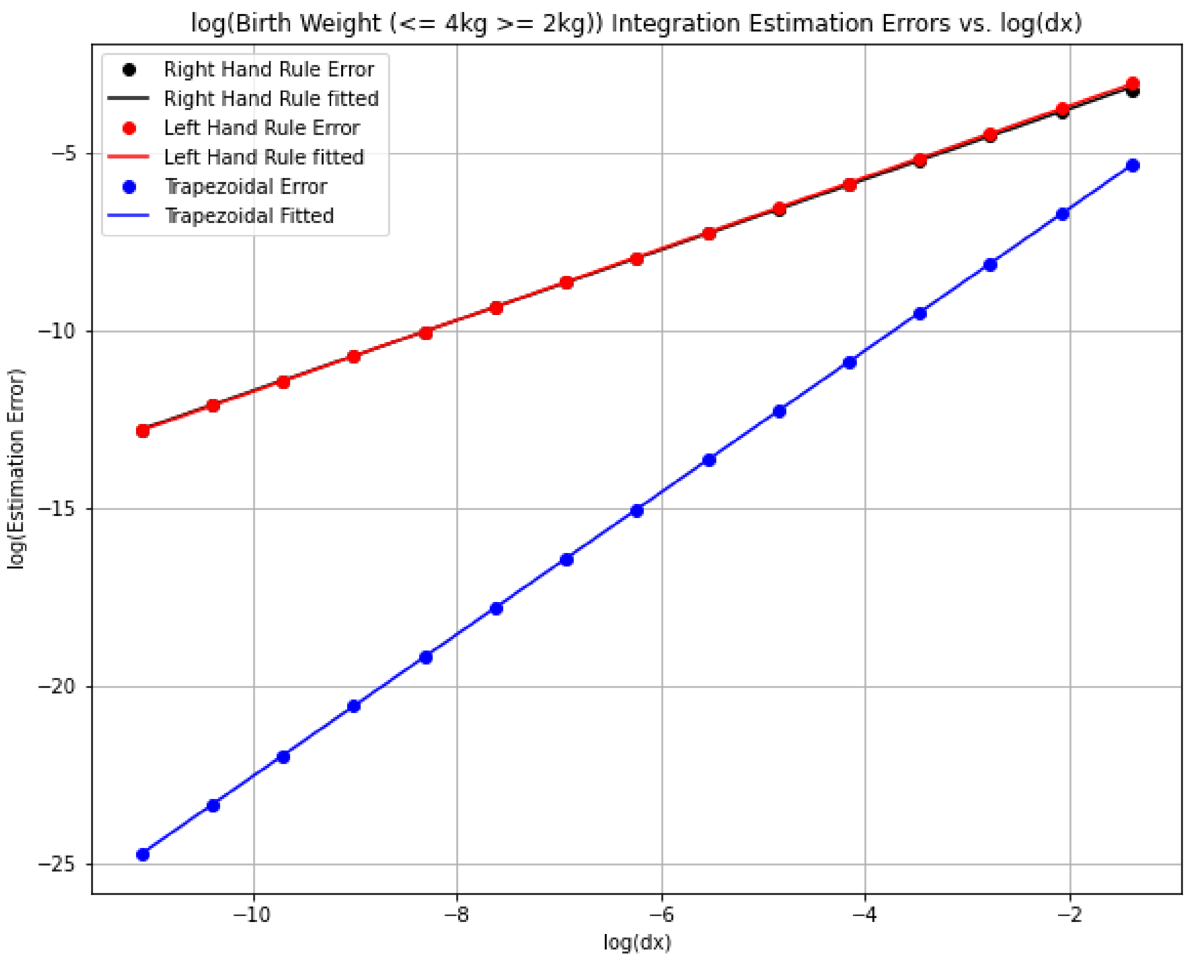
In [8]:
```
# Problem 2

x = []
for dx in range(2, 17):
    x.append(2**(-dx))

# a) done without log, deleted code

# b)

plt.figure(figsize=(10, 8))
plt.title('Birth Weight (<= 4kg >= 2kg) Integration Estimation Errors vs. dx')
plt.xlabel('dx')
plt.ylabel('Estimation Error')
plt.grid()
plt.plot(np.log(x), np.log(RHR_err), 'ko', np.log(x), np.log(LHR_err), 'ro', n
p.log(x), np.log(Trap_err), 'bo')
plt.legend(['Right Hand Rule Error', 'Left Hand Rule Error', 'Trapezoidal Erro
r'])
```

Out[8]:
```
<matplotlib.legend.Legend at 0x19b33647dc8>
```



In [9]:
```
# c)

logx = np.log(x)
log_rhr = np.log(RHR_err)
log_lhr = np.log(LHR_err)
log_trap = np.log(Trap_err)

# RHR
coeffs_rhr = np.polyfit(logx, log_rhr, 1)
rhrhat = np.polyval(coeffs_rhr, logx)

# LHR
coeffs_lhr = np.polyfit(logx, log_lhr, 1)
lhrhat = np.polyval(coeffs_lhr, logx)

# Trapezoidal
coeffs_trap = np.polyfit(logx, log_trap, 1)
traphat = np.polyval(coeffs_trap, logx)

plt.figure(figsize=(10, 8))
plt.title('log(Birth Weight (<= 4kg >= 2kg)) Integration Estimation Errors vs. 
log(dx)')
plt.xlabel('log(dx)')
plt.ylabel('log(Estimation Error)')
plt.grid()
plt.plot(logx, log_rhr, 'ko', logx, rhrhat, 'k', logx, log_lhr, 'ro', logx, lh
rhat, 'r', logx, log_trap, 'bo', logx, traphat, 'b')
plt.legend(['Right Hand Rule Error', 'Right Hand Rule fitted', 'Left Hand Rule 
Error', 'Left Hand Rule fitted', 'Trapezoidal Error', 'Trapezoidal Fitted'])

# d)

"""
The slopes of these lines represent the Estimation Error's rate of change rega
rding dx. As we can see, the earlier assumptions
we made about the trapezoidal error being about half that of the LHR and RHR h
olds true. As dx shrinks, the errors of LHR/RHR vs.
trapezoidal continue to shrink following this relationship. This reinforces an
d proves that the orders of accuracy are 1st
order for LHR and RHR and and 2nd order for trapezoidal.
"""
```

```
Out[9]: '\nThe slopes of these lines represent the \n'
```



```
In [ ]:
```