Lab3

December 6, 2020

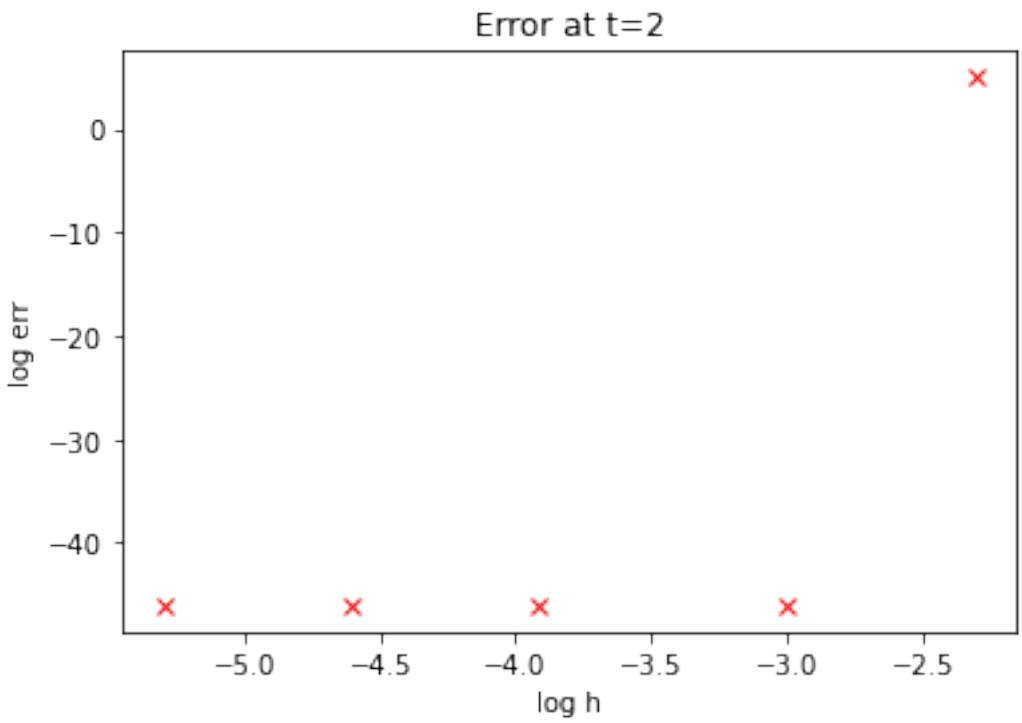
[330]:
```
#**********************#
#*          Q1        *#
#**********************#


# Cauchy Problem
# y'(x) = lambda y(x)
# y(0) = y0


def forward_euler(h,y0,lamb):
    y0 = y0 + h*np.dot(lamb, y0)
    return y0

def backward_euler(h, y0, lamb):
    # Implement Newton's Iteration
    y1 = forward_euler(h, y0, lamb)    # Initial value for Newton Iteration
    g = y1-y0-h*lamb*y1
    g_prime = 1-h*lamb**2*y1
    for i in range(0,1000):
        y0 = y1
        y1 = y0 - g/g_prime
        g = y1-y0-h*lamb*y1
        g_prime = 1-h*lamb**2*y1


    return y1
```

[331]:
```
#**********************#
#*        Q2          *#
#**********************#

import numpy as np
import matplotlib.pyplot as plt


# 2a) lamb = -23; h = 0.1
```

```
lamb = -23
h = 0.1
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = forward_euler(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e1 = err[x.shape[0]-1]


# 2a) h = 0.05
h = 0.05
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = forward_euler(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e2 = err[x.shape[0]-1]


# 2a) h = 0.02
h = 0.02
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(1,x.shape[0]):
    y[i+1] = forward_euler(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e3 = err[x.shape[0]-1]

# 2a) h = 0.01
h = 0.01
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = y[i] + h*lamb*y[i]
```

```
err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e4 = err[x.shape[0]-1]

# 2a) h = 0.005
h = 0.005
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = forward_euler(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e5 = err[x.shape[0]-1]



plt.plot(np.log([0.1,0.05,0.02,0.01,0.005]),[e1,e2,e3,e4,e5],'rx')
plt.xlabel("log h")
plt.ylabel("log err")
plt.title('Lambda = -23')
plt.title("Error at t=2")

## We can see that the approximations errors are getting small as
## stepsize h becomes small.
```

[331]: Text(0.5, 1.0, 'Error at t=2')



[ ]:

[ ]:

[ ]:

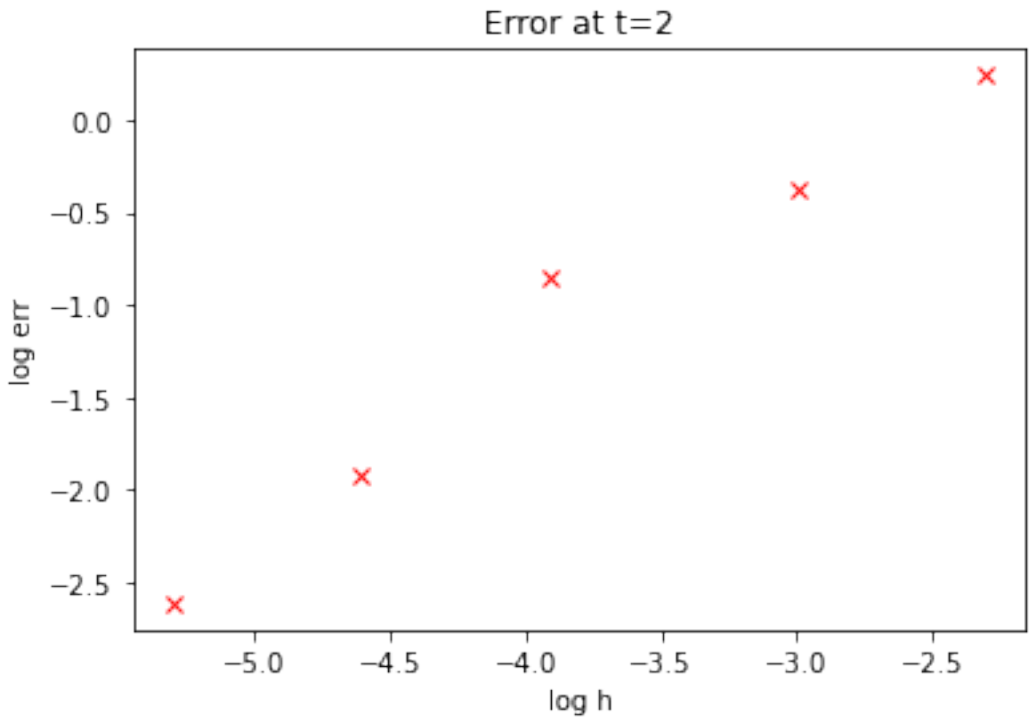
[ ]:

[332]:
```
# 2a) lamb = 1; h = 0.1
lamb = 1
h = 0.1
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = forward_euler(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e1 = err[x.shape[0]-1]


# 2a) lamb = 1; h = 0.05
```

```
h = 0.05
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = forward_euler(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e2 = err[x.shape[0]-1]


# 2a) lamb = 1; h = 0.02
h = 0.02
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(1,x.shape[0]):
    y[i+1] = forward_euler(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e3 = err[x.shape[0]-1]

# 2a) lamb = 1; h = 0.01
h = 0.01
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = y[i] + h*lamb*y[i]

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e4 = err[x.shape[0]-1]

# 2a) lamb = 1; h = 0.005
h = 0.005
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = forward_euler(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e5 = err[x.shape[0]-1]
```

```
plt.plot(np.log([0.1,0.05,0.02,0.01,0.005]),[e1,e2,e3,e4,e5],'rx')
plt.xlabel("log h")
plt.ylabel("log err")
plt.title('Lambda = 1')
plt.title("Error at t=2")


## We can see that the approximations errors are getting small as
## stepsize h becomes small.
```

[332]: Text(0.5, 1.0, 'Error at t=2')



[ ]:

[ ]:

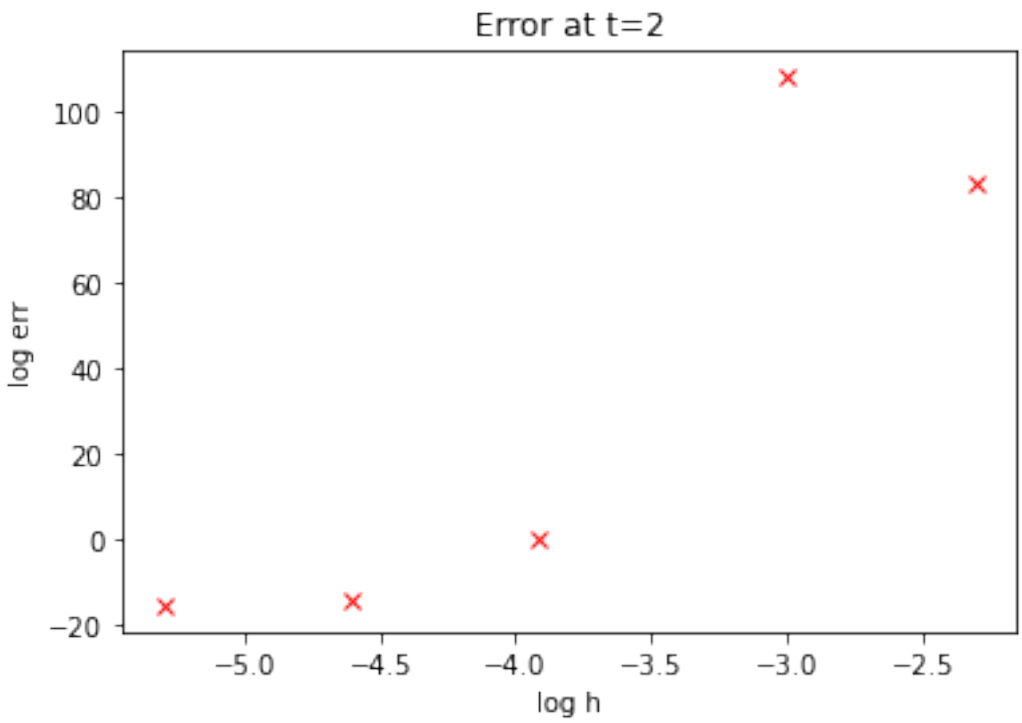
[ ]:

[ ]:

[333]:
```
# 2a) lamb = diag(-1,-100); h = 0.1
lamb = np.diag([+1,-100])
h = 0.1
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = forward_euler(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e1 = err[x.shape[0]-1]

# 2a) lamb = diag(-1,-100); h = 0.05
lamb = np.diag([-1,-100])
h = 0.05
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = forward_euler(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e2 = err[x.shape[0]-1]


# 2a) lamb = diag(-1,-100); h = 0.02
lamb = np.diag([-1,-100])
h = 0.02
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = forward_euler(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e3 = err[x.shape[0]-1]
```

```
# 2a) lamb = diag(-1,-100); h = 0.01
lamb = np.diag([-1,-100])
h = 0.01
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = forward_euler(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e4 = err[x.shape[0]-1]


# 2a) lamb = diag(-1,-100); h = 0.005
lamb = np.diag([-1,-100])
h = 0.005
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = forward_euler(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e5 = err[x.shape[0]-1]


plt.plot(np.log([0.1,0.05,0.02,0.01,0.005]),[e1,e2,e3,e4,e5],'rx')
plt.xlabel("log h")
plt.ylabel("log err")
plt.title("Error at t=2")


## We can see that the approximations errors are getting small as
## stepsize h becomes small.
```

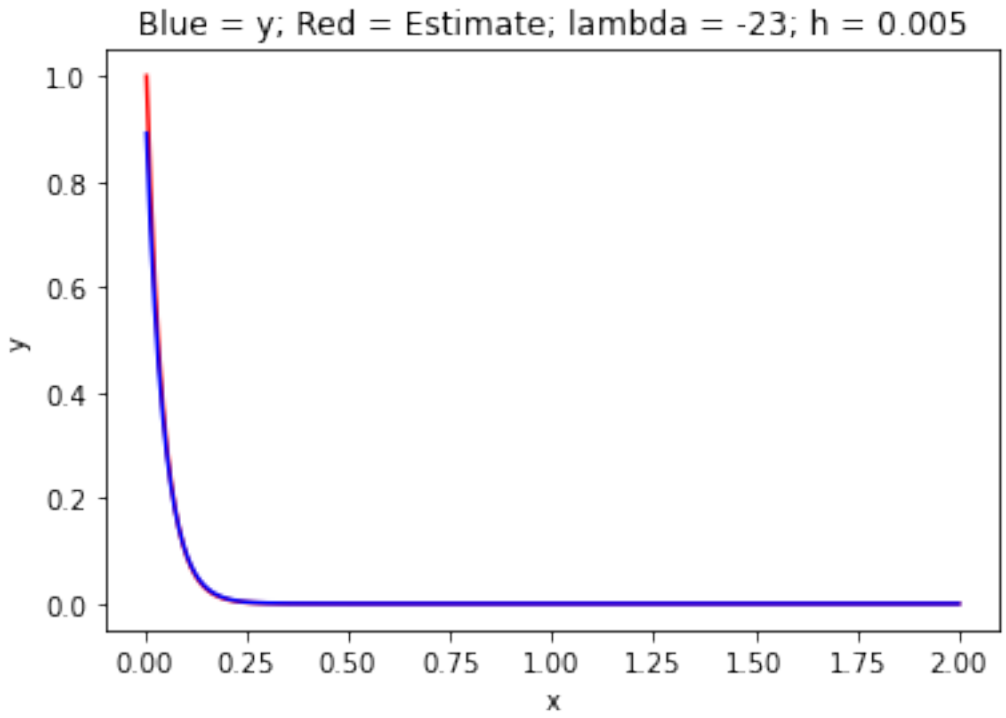
[333]: Text(0.5, 1.0, 'Error at t=2')



[334]:
```
#**********************
#*          Q2c         *#
#**********************

# Let h = 0.05, lambda = -23
lamb = -23
h = 0.005
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = forward_euler(h, y[i], lamb)

# err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))



plt.plot(x, y[:x.shape[0]],'r-',x,np.exp(lamb*x),'b-')
plt.xlabel("x")
plt.ylabel("y")
plt.title("Blue = y; Red = Estimate; lambda = -23; h = 0.005")
```

```
## We can see that the approximations are getting very close to the exact␣
 ↪values as
## stepsize h becomes small.
```

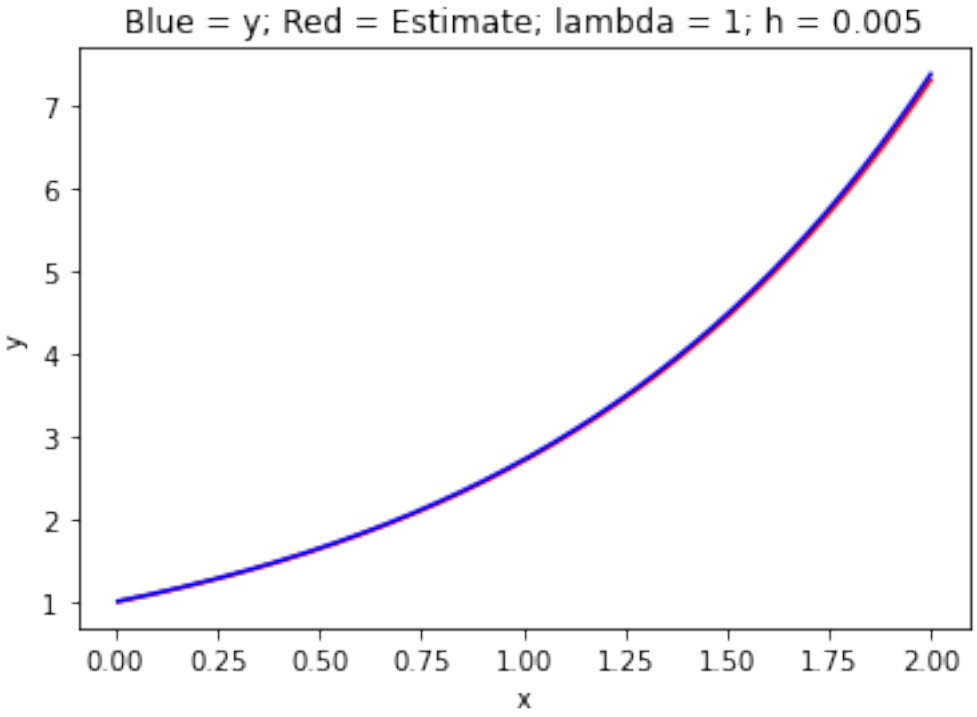
[334]:
```
Text(0.5, 1.0, 'Blue = y; Red = Estimate; lambda = -23; h = 0.005')
```



[335]:
```
##########################
##          Q2c          #
##########################

# Let h = 0.05, lambda = 1
lamb = 1
h = 0.005
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = forward_euler(h, y[i], lamb)

# err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
```

```
plt.plot(x, y[:x.shape[0]],'r-',x,np.exp(lamb*x),'b-')
plt.xlabel("x")
plt.ylabel("y")
plt.title("Blue = y; Red = Estimate; lambda = 1; h = 0.005")


## We can see that the approximations are getting very close to the exact␣
 ↪values as
## stepsize h becomes small.
```

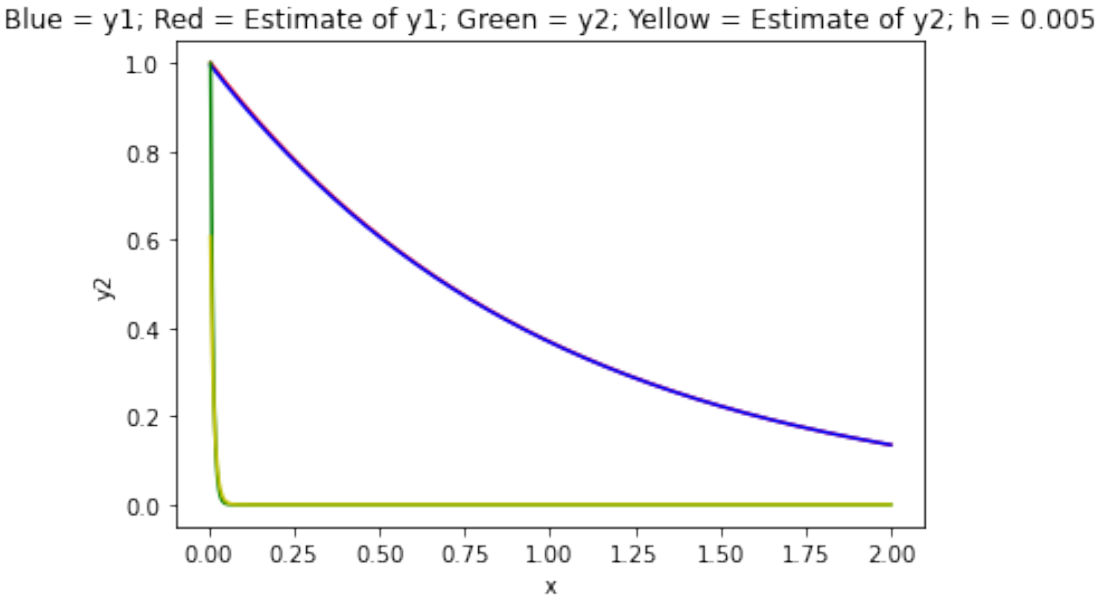
[335]: `Text(0.5, 1.0, 'Blue = y; Red = Estimate; lambda = 1; h = 0.005')`



[336]:
```
# 2c) lamb = diag(-1,-100); h = 0.005
lamb = np.diag([-1,-100])
h = 0.005
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = forward_euler(h, y[i,:], lamb)
```

```
plt.plot(x, y[:x.shape[0],0],'r-',x,np.exp(lamb[0,0]*x),'b-')
plt.xlabel("x")
plt.ylabel("y1")

plt.plot(x, y[:x.shape[0],1],'g-',x,np.exp(lamb[1,1]*x),'y-')
plt.xlabel("x")
plt.ylabel("y2")
plt.title("Blue = y1; Red = Estimate of y1; Green = y2; Yellow = Estimate of y2;
 ↪ h = 0.005")


## We can see that the approximations errors are getting small as
## stepsize h becomes small.
```

[336]:
```
Text(0.5, 1.0, 'Blue = y1; Red = Estimate of y1; Green = y2; Yellow = Estimate
of y2; h = 0.005')
```



[337]:
```
#**********************
#*          Q3          *#
#**********************


# Runge-Kutta
```

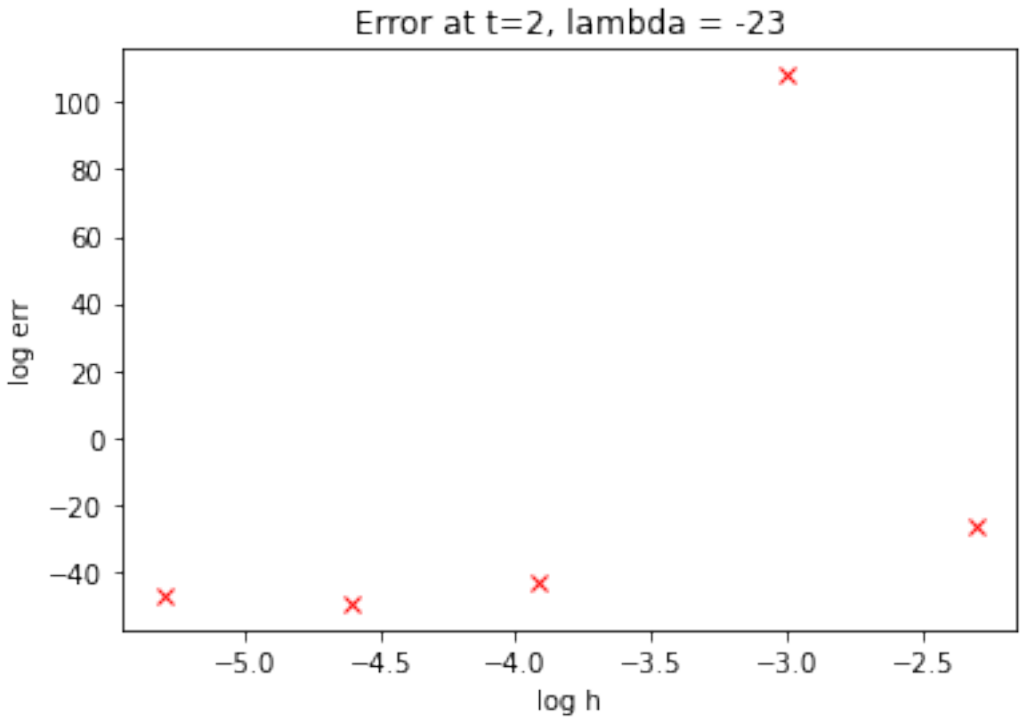
```
def RK(h,y0,lamb):
    b1 = 0
    c2 = 0.5
    b2 = 1
    K1 = np.dot(lamb,y0)
    K2 = np.dot(lamb,(y0+h*c2*K1))
    y1 = y0+h*(b1*K1+b2*K2)
    return y1



def RK4(h,y0,lamb):
    c1 = 0
    c2 = 0.5
    c3 = 0.5
    c4 = 1
    b1 = 1/6
    b2 = 1/3
    b3 = 1/3
    b4 = 1/6


    K1 = np.dot(lamb,y0)
    K2 = np.dot(lamb,(y0+h*c2*K1))
    K3 = np.dot(lamb,(y0+h*c3*K2))
    K4 = np.dot(lamb,(y0+h*c4*K3))
    y1 = y0+h*(b1*K1+b2*K2+b3*K3+b4*K4)
    return y1
```

[338]:
```
# 2-Stage Runge Kutta for Lambda = -23 with varying step size


lamb = -23

# 3a) h = 0.1
h = 0.1
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e1 = err[x.shape[0]-1]
```

```
# 3a) h = 0.05
h = 0.05
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)

for i in range(0,x.shape[0]):
    y[i+1] = RK(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e1 = err[x.shape[0]-1]


# 3a) h = 0.02
h = 0.02
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e3 = err[x.shape[0]-1]


# 3a) h = 0.01
h = 0.001
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e4 = err[x.shape[0]-1]


# 3a) h = 0.005
h = 0.005
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK(h, y[i], lamb)
```

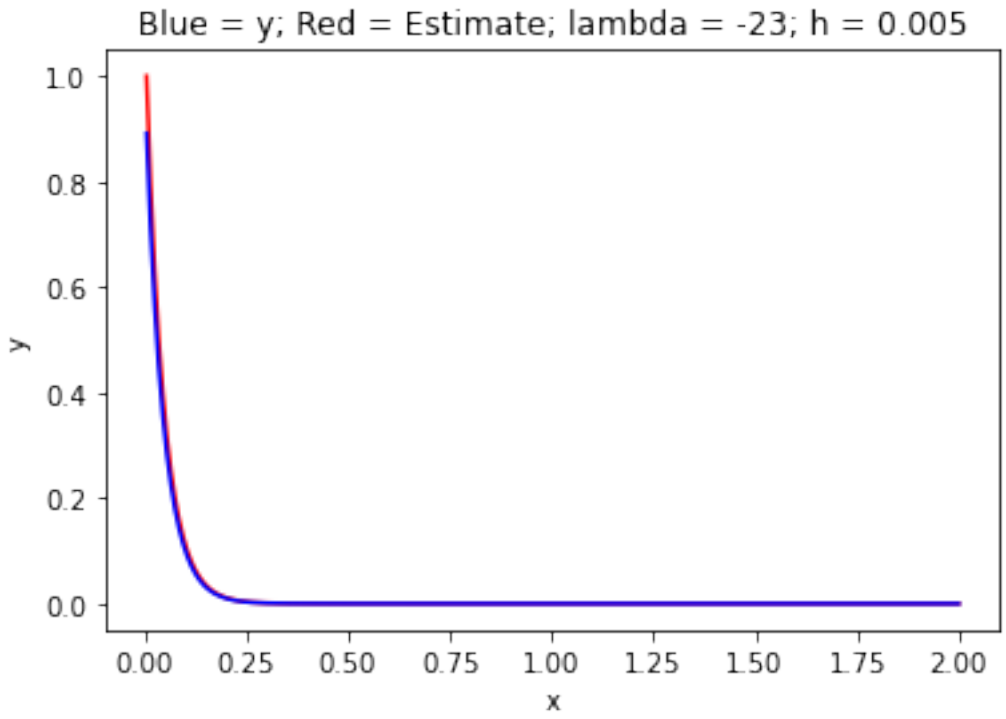
```
err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e5 = err[x.shape[0]-1]


plt.plot(np.log([0.1,0.05,0.02,0.01,0.005]),[e1,e2,e3,e4,e5],'rx')
plt.xlabel("log h")
plt.ylabel("log err")
plt.title("Error at t=2, lambda = -23")

## We can see that the approximations errors are getting small as
## stepsize h becomes small.
```

[338]:
```
Text(0.5, 1.0, 'Error at t=2, lambda = -23')
```



[339]:
```
plt.plot(x, y[:x.shape[0]],'r-',x,np.exp(lamb*x),'b-')
plt.xlabel("x")
plt.ylabel("y")

plt.title("Blue = y; Red = Estimate; lambda = -23; h = 0.005")

## We can see that the approximations are getting very close to the exact␣
 ↪values as
## stepsize h becomes small.
```

[339]: Text(0.5, 1.0, 'Blue = y; Red = Estimate; lambda = -23; h = 0.005')



[ ]:

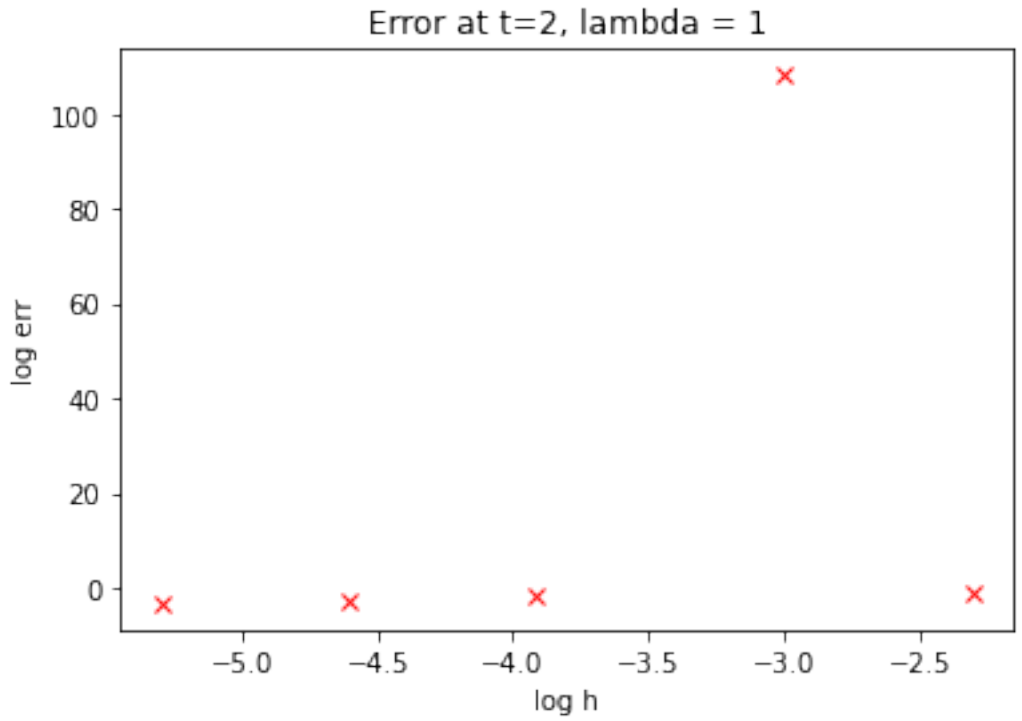
[ ]:

[340]:
```
# 2-Stage Runge Kutta for Lambda = 1 with varying step size


lamb = 1

# 3a) h = 0.1
h = 0.1
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e1 = err[x.shape[0]-1]
```

```
# 3a) h = 0.05
h = 0.05
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)

for i in range(0,x.shape[0]):
    y[i+1] = RK(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e1 = err[x.shape[0]-1]


# 3a) h = 0.02
h = 0.02
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e3 = err[x.shape[0]-1]


# 3a) h = 0.01
h = 0.01
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e4 = err[x.shape[0]-1]


# 3a) h = 0.005
h = 0.005
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK(h, y[i], lamb)
```

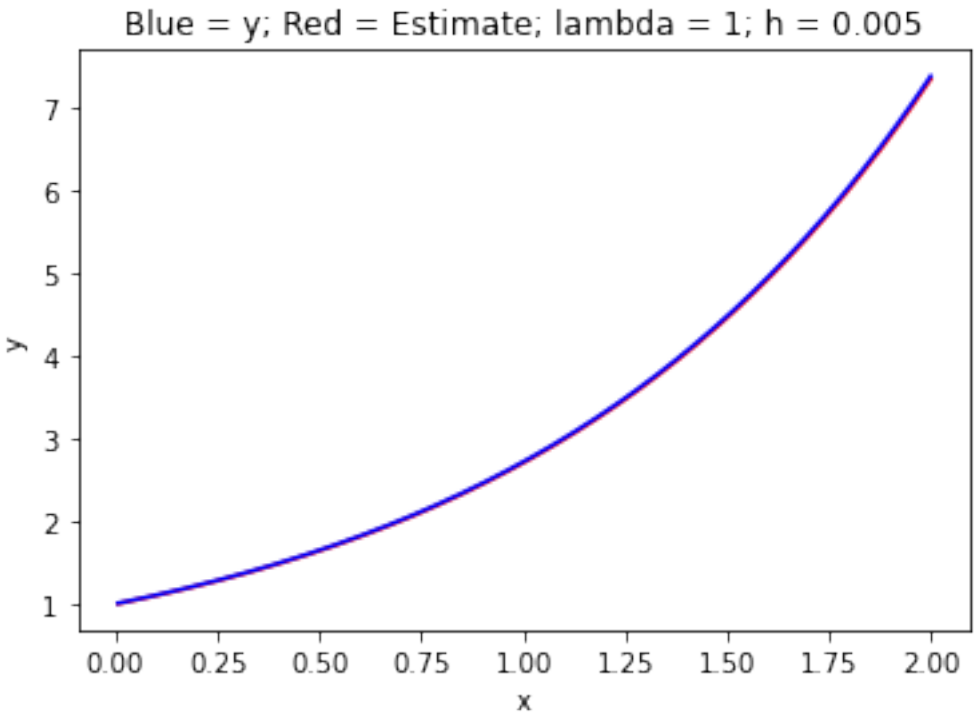
```
err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e5 = err[x.shape[0]-1]


plt.plot(np.log([0.1,0.05,0.02,0.01,0.005]),[e1,e2,e3,e4,e5],'rx')
plt.xlabel("log h")
plt.ylabel("log err")
plt.title("Error at t=2, lambda = 1")

## We can see that the approximations errors are getting small as
## stepsize h becomes small.
```

[340]:
```
Text(0.5, 1.0, 'Error at t=2, lambda = 1')
```



[341]:
```
plt.plot(x, y[:x.shape[0]],'r-',x,np.exp(lamb*x),'b-')
plt.xlabel("x")
plt.ylabel("y")

plt.title("Blue = y; Red = Estimate; lambda = 1; h = 0.005")

## We can see that the approximations are getting very close to the exact␣
 ↪values as
## stepsize h becomes small.
```

[341]: Text(0.5, 1.0, 'Blue = y; Red = Estimate; lambda = 1; h = 0.005')



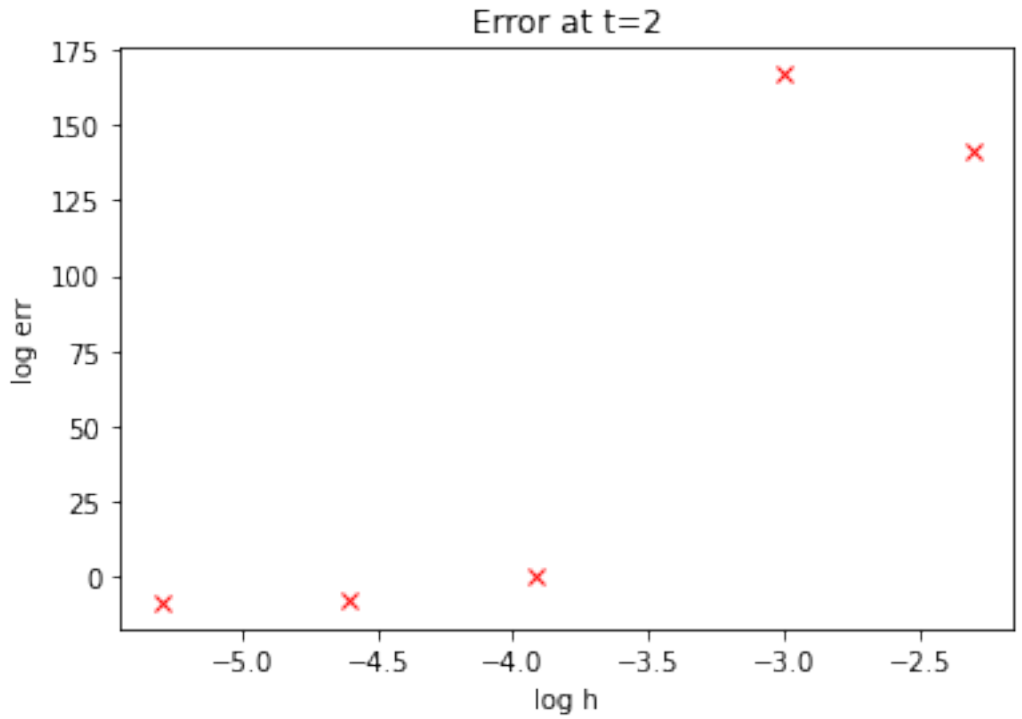
[ ]:

[342]:
```
# 3a) lamb = diag(-1,-100); h = 0.1
lamb = np.diag([-1,-100])
h = 0.1
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = RK(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e1 = err[x.shape[0]-1]

# 3a)  h = 0.05
lamb = np.diag([-1,-100])
h = 0.05
x = np.arange(0,2,h)+h
```

```
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = RK(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e2 = err[x.shape[0]-1]


# 3a)  h = 0.02
lamb = np.diag([-1,-100])
h = 0.02
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = RK(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e3 = err[x.shape[0]-1]


# 3a)  h = 0.01
lamb = np.diag([-1,-100])
h = 0.01
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = RK(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e4 = err[x.shape[0]-1]


# 3a)  h = 0.005
lamb = np.diag([-1,-100])
h = 0.005
```

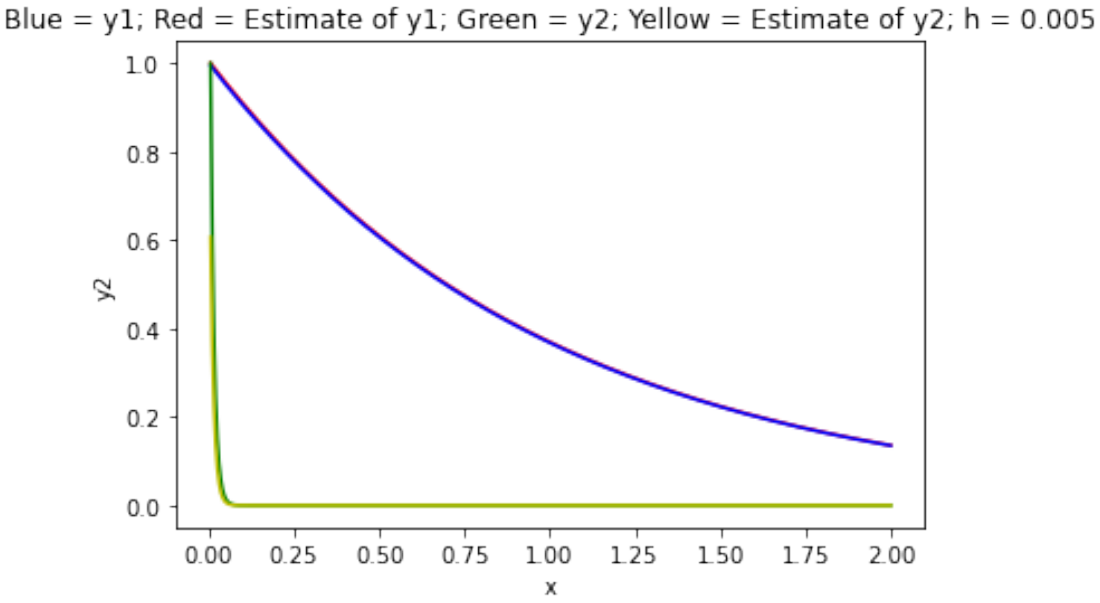
```
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = RK(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e5 = err[x.shape[0]-1]


plt.plot(np.log([0.1,0.05,0.02,0.01,0.005]),[e1,e2,e3,e4,e5],'rx')
plt.xlabel("log h")
plt.ylabel("log err")
plt.title("Error at t=2")



## We can see that the approximations errors are getting small as
## stepsize h becomes small.
```

[342]: Text(0.5, 1.0, 'Error at t=2')



[343]:
```
plt.plot(x, y[:x.shape[0],0],'r-',x,np.exp(lamb[0,0]*x),'b-')
plt.xlabel("x")
plt.ylabel("y1")

plt.plot(x, y[:x.shape[0],1],'g-',x,np.exp(lamb[1,1]*x),'y-')
plt.xlabel("x")
plt.ylabel("y2")
plt.title("Blue = y1; Red = Estimate of y1; Green = y2; Yellow = Estimate of y2;
 ↪ h = 0.005")




## We can see that the approximations are getting very close to the exact␣
 ↪values as
## stepsize h becomes small.
```

[343]:
```
Text(0.5, 1.0, 'Blue = y1; Red = Estimate of y1; Green = y2; Yellow = Estimate
of y2; h = 0.005')
```



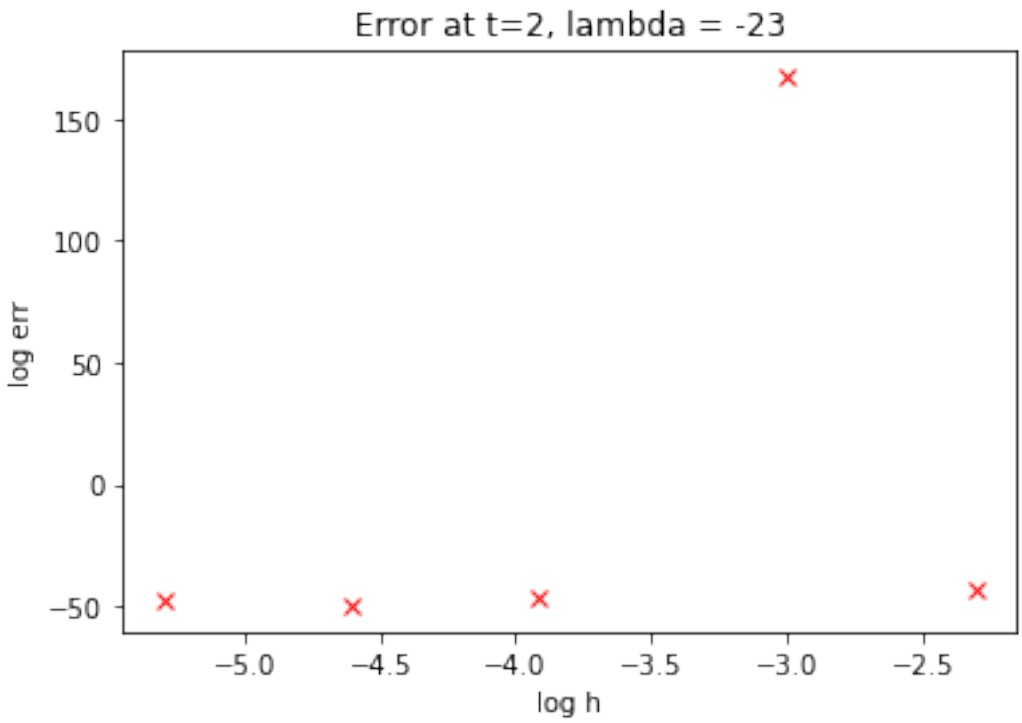
[ ]:

[344]:
```
# 4-Stage Runge Kutta for Lambda = -23 with varying step size
```

```
lamb = -23

# 3a) h = 0.1
h = 0.1
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK4(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e1 = err[x.shape[0]-1]

# 3a) h = 0.05
h = 0.05
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)

for i in range(0,x.shape[0]):
    y[i+1] = RK4(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e1 = err[x.shape[0]-1]


# 3a) h = 0.02
h = 0.02
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK4(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e3 = err[x.shape[0]-1]


# 3a) h = 0.01
h = 0.001
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK4(h, y[i], lamb)
```

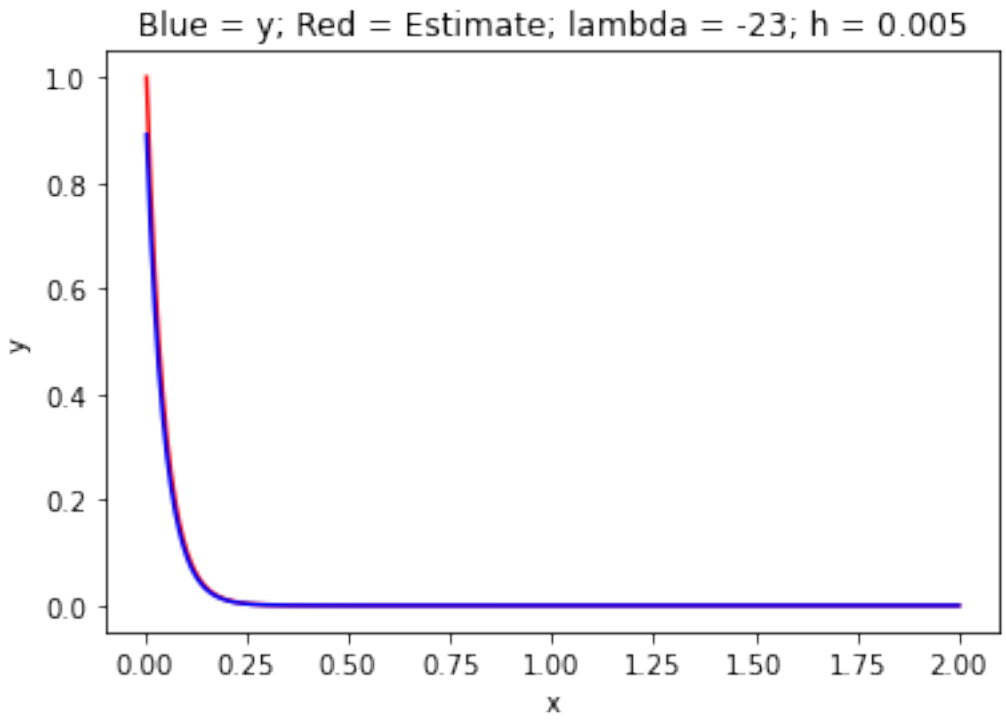
```
err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e4 = err[x.shape[0]-1]


# 3a) h = 0.005
h = 0.005
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK4(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e5 = err[x.shape[0]-1]


plt.plot(np.log([0.1,0.05,0.02,0.01,0.005]),[e1,e2,e3,e4,e5],'rx')
plt.xlabel("log h")
plt.ylabel("log err")
plt.title("Error at t=2, lambda = -23")

## We can see that the approximations errors are getting small as
## stepsize h becomes small.
```

[344]: Text(0.5, 1.0, 'Error at t=2, lambda = -23')



[345]:
```
plt.plot(x, y[:x.shape[0]],'r-',x,np.exp(lamb*x),'b-')
plt.xlabel("x")
plt.ylabel("y")

plt.title("Blue = y; Red = Estimate; lambda = -23; h = 0.005")

## We can see that the approximations are getting very close to the exact␣
 ↪values as
## stepsize h becomes small.
```

[345]:
```
Text(0.5, 1.0, 'Blue = y; Red = Estimate; lambda = -23; h = 0.005')
```



[ ]:

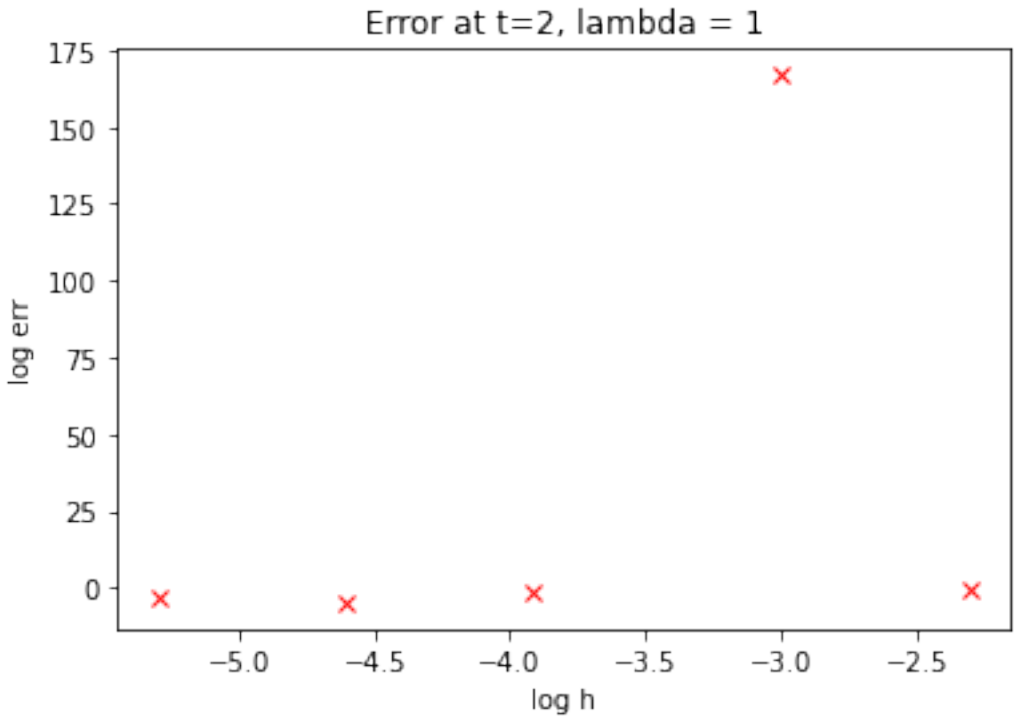
[ ]:

[350]:
```
# 4-Stage Runge Kutta for Lambda = 1 with varying step size


lamb = 1

# 3a) h = 0.1
h = 0.1
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK4(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e1 = err[x.shape[0]-1]

# 3a) h = 0.05
h = 0.05
```

```
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)

for i in range(0,x.shape[0]):
    y[i+1] = RK4(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e1 = err[x.shape[0]-1]


# 3a) h = 0.02
h = 0.02
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK4(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e3 = err[x.shape[0]-1]


# 3a) h = 0.01
h = 0.001
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK4(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e4 = err[x.shape[0]-1]


# 3a) h = 0.005
h = 0.005
x = np.arange(0,2,h)+h
y = np.ones(x.shape[0]+1)


for i in range(0,x.shape[0]):
    y[i+1] = RK4(h, y[i], lamb)

err = np.log(np.abs(y[0:x.shape[0]]-np.exp(lamb*x)))
e5 = err[x.shape[0]-1]
```

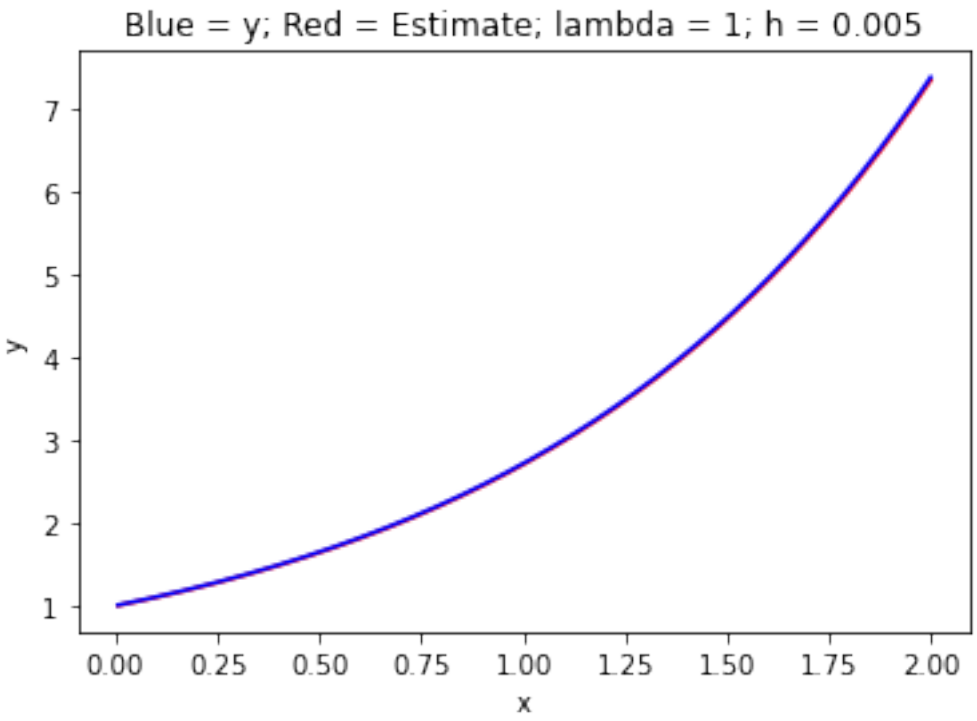
```
plt.plot(np.log([0.1,0.05,0.02,0.01,0.005]),[e1,e2,e3,e4,e5],'rx')
plt.xlabel("log h")
plt.ylabel("log err")
plt.title("Error at t=2, lambda = 1")

## We can see that the approximations errors are getting small as
## stepsize h becomes small.
```

[350]: Text(0.5, 1.0, 'Error at t=2, lambda = 1')



[351]:
```
plt.plot(x, y[:x.shape[0]],'r-',x,np.exp(lamb*x),'b-')
plt.xlabel("x")
plt.ylabel("y")

plt.title("Blue = y; Red = Estimate; lambda = 1; h = 0.005")

## We can see that the approximations are getting very close to the exact␣
 ↪values as
## stepsize h becomes small.
```

[351]: Text(0.5, 1.0, 'Blue = y; Red = Estimate; lambda = 1; h = 0.005')



[ ]:

[ ]:

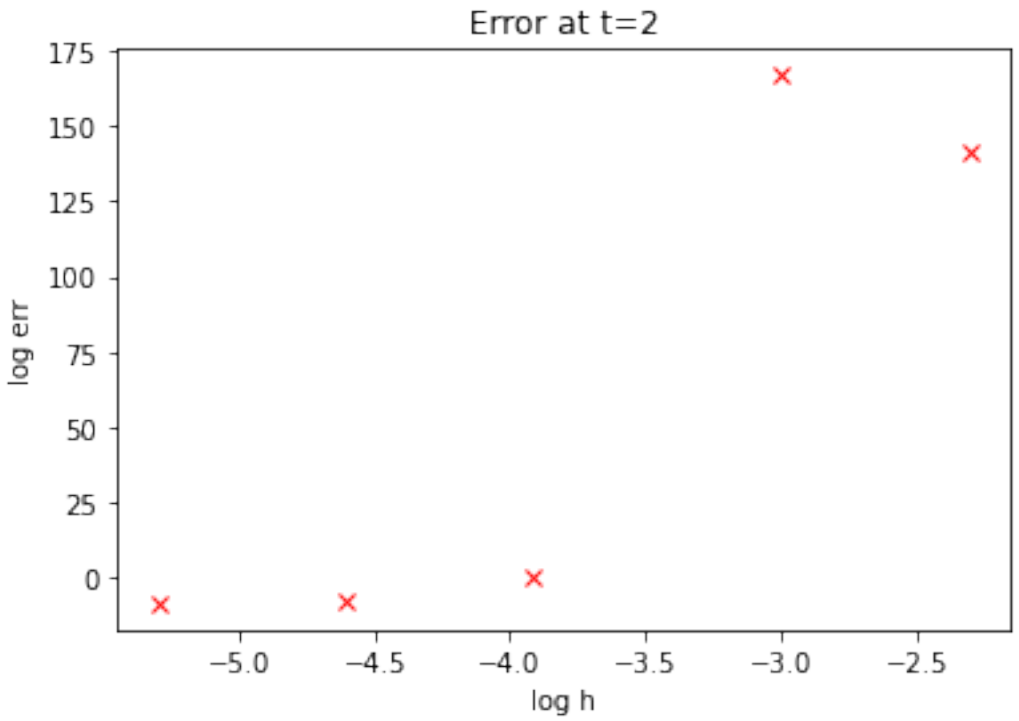
[ ]:

[352]:
```
# 4-Step Runge-Kutta Method

# 3a) lamb = diag(-1,-100); h = 0.1
lamb = np.diag([-1,-100])
h = 0.1
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = RK(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e1 = err[x.shape[0]-1]
```

```
# 3a)  h = 0.05
lamb = np.diag([-1,-100])
h = 0.05
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = RK(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e2 = err[x.shape[0]-1]


# 3a)  h = 0.02
lamb = np.diag([-1,-100])
h = 0.02
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = RK(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e3 = err[x.shape[0]-1]


# 3a)  h = 0.01
lamb = np.diag([-1,-100])
h = 0.01
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = RK(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e4 = err[x.shape[0]-1]
```

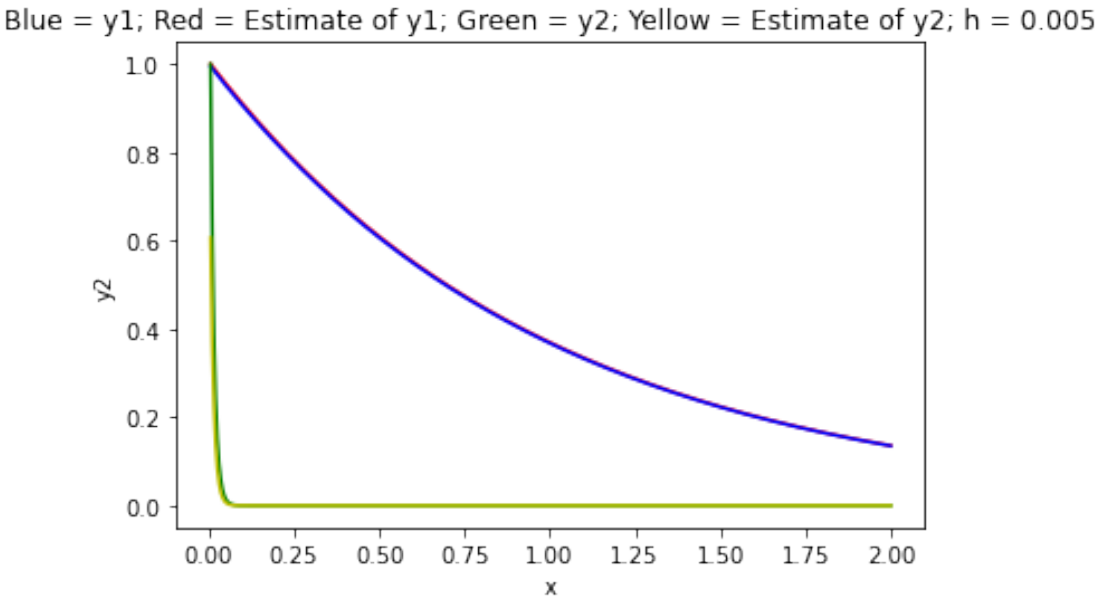
```
# 3a)  h = 0.005
lamb = np.diag([-1,-100])
h = 0.005
x = np.arange(0,2,h)+h
y = np.ones([x.shape[0]+1,2])


for i in range(0,x.shape[0]):
    y[i+1,:] = RK(h, y[i,:], lamb)


err = np.log(np.abs(np.sum(y[0:x.shape[0],:]**2,axis=1)-(np.exp(-2*x)+np.
 ↪exp(-200*x))))
e5 = err[x.shape[0]-1]


plt.plot(np.log([0.1,0.05,0.02,0.01,0.005]),[e1,e2,e3,e4,e5],'rx')
plt.xlabel("log h")
plt.ylabel("log err")
plt.title("Error at t=2")



## We can see that the approximations errors are getting small as
## stepsize h becomes small.
```

[352]: Text(0.5, 1.0, 'Error at t=2')



[353]:
```
plt.plot(x, y[:x.shape[0],0],'r-',x,np.exp(lamb[0,0]*x),'b-')
plt.xlabel("x")
plt.ylabel("y1")

plt.plot(x, y[:x.shape[0],1],'g-',x,np.exp(lamb[1,1]*x),'y-')
plt.xlabel("x")
plt.ylabel("y2")
plt.title("Blue = y1; Red = Estimate of y1; Green = y2; Yellow = Estimate of y2;
 ↪ h = 0.005")




## We can see that the approximations are getting very close to the exact␣
 ↪values as
## stepsize h becomes small.
```

[353]:
```
Text(0.5, 1.0, 'Blue = y1; Red = Estimate of y1; Green = y2; Yellow = Estimate
of y2; h = 0.005')
```


Blue = y1; Red = Estimate of y1; Green = y2; Yellow = Estimate of y2; h = 0.005
1.0
0.8
0.6
0.4
0.2
0.0
y2
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
x

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: