RENATO POLI (rep2656) February, 13th 2024

**PGE382L – CP3 (Spring, 2024)**

Solve:

**CASE 1 - Crank Nicholson, Central differences:**

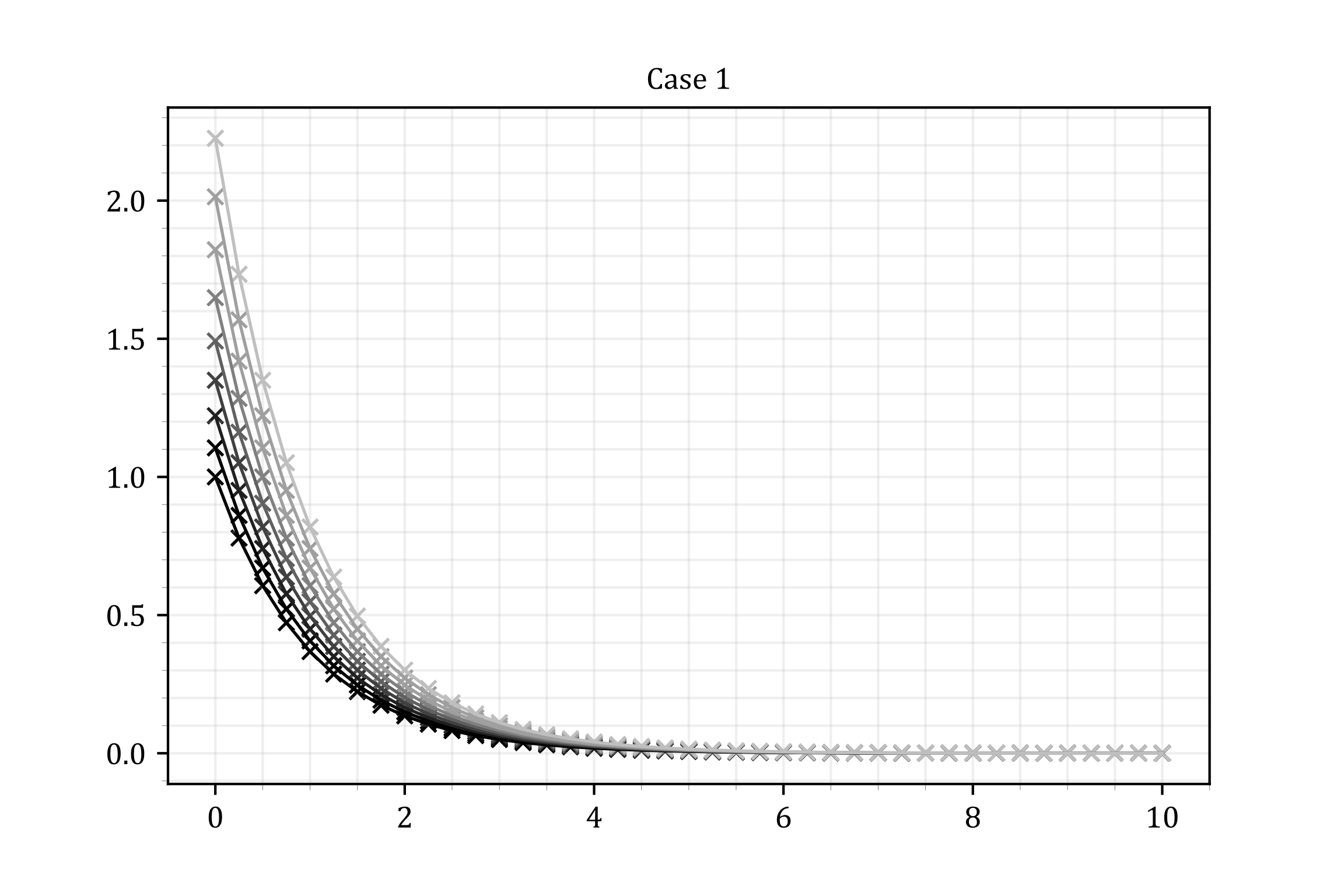
Factor and reorganize LHS/RHS:

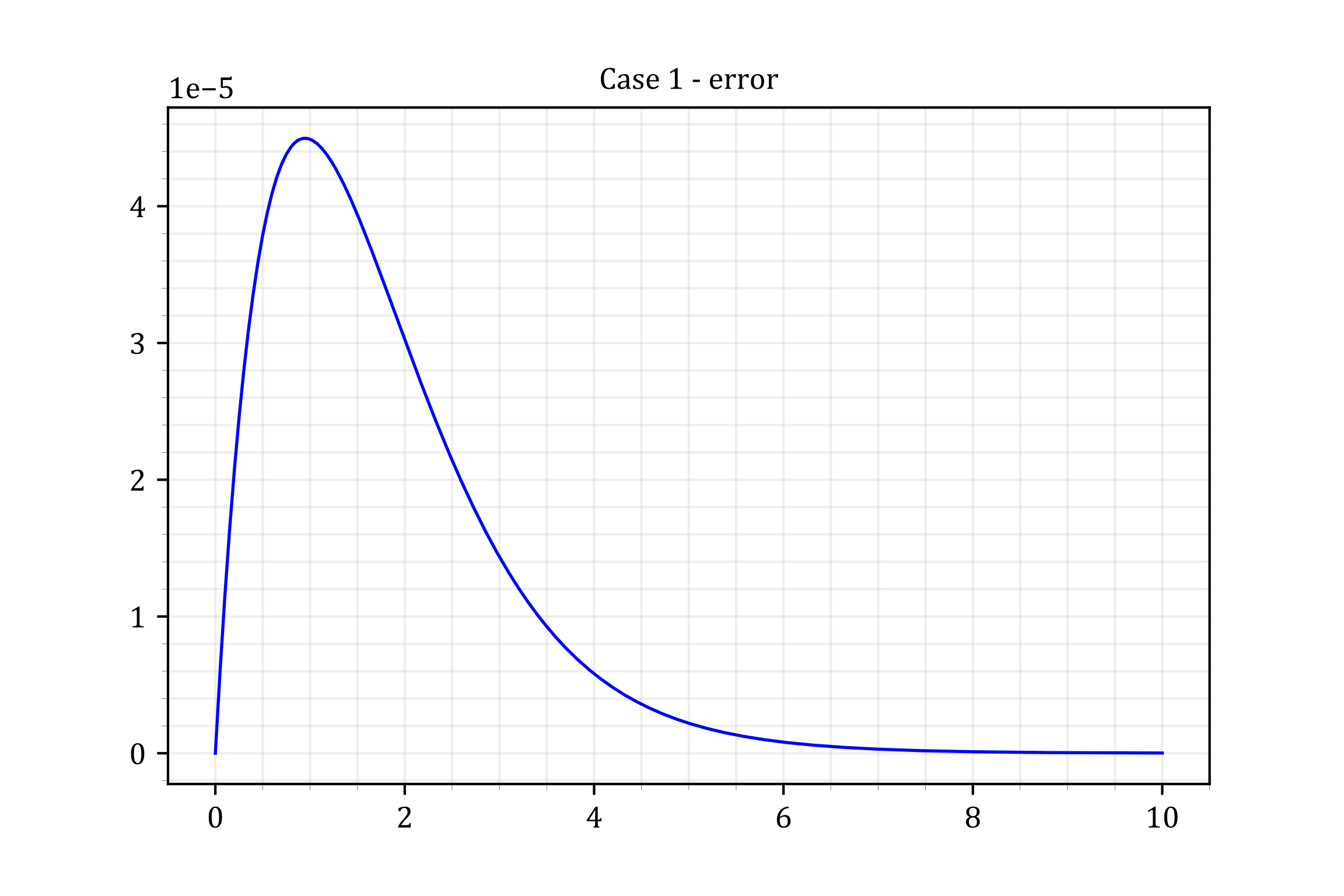
Multiply by 2 and reorganize:

BC @ defines the equation at

BC @

**SOLUTION – CASE 1**





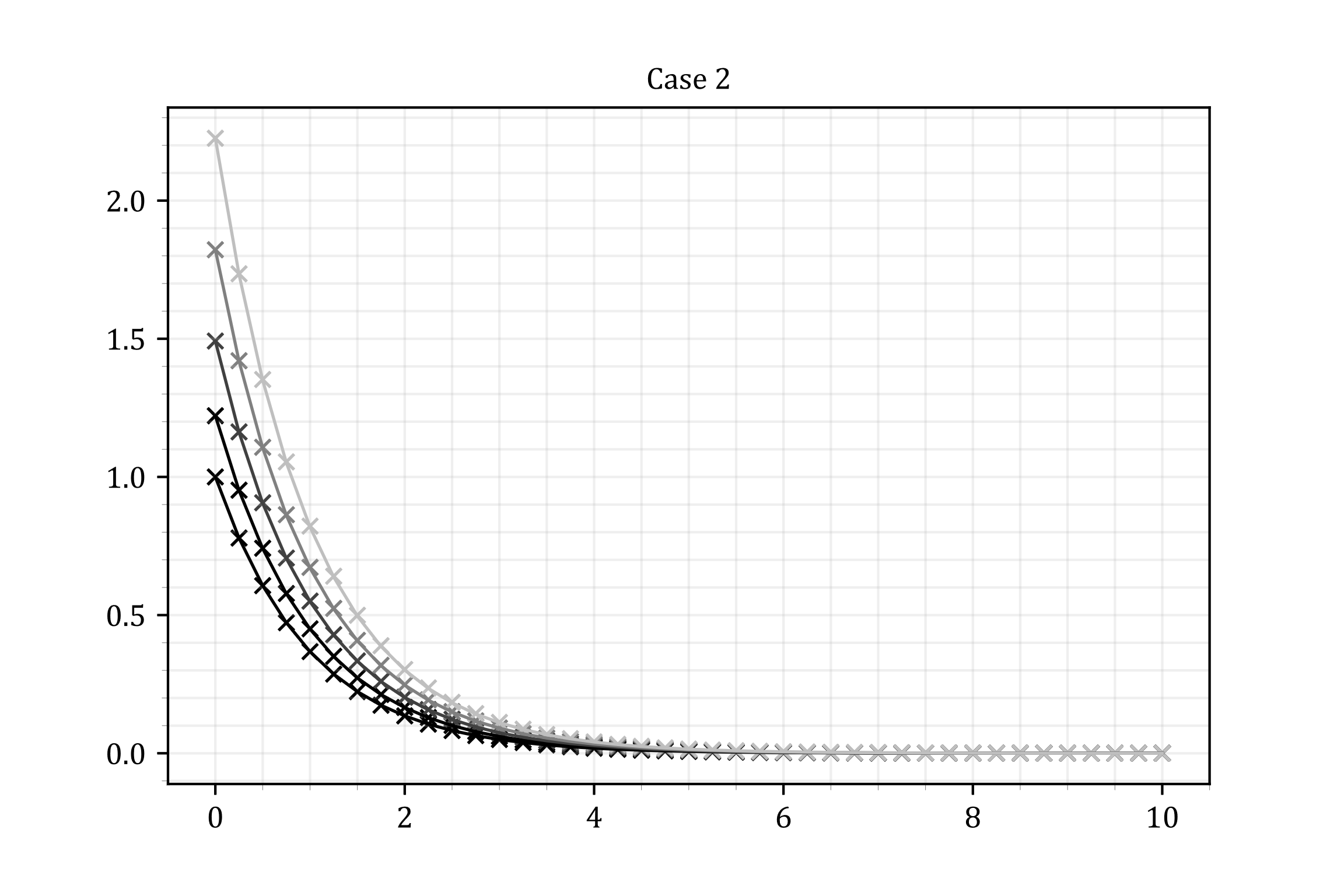
**CASE 2 - Crank Nicholson, Central differences for and Backward difference for :**

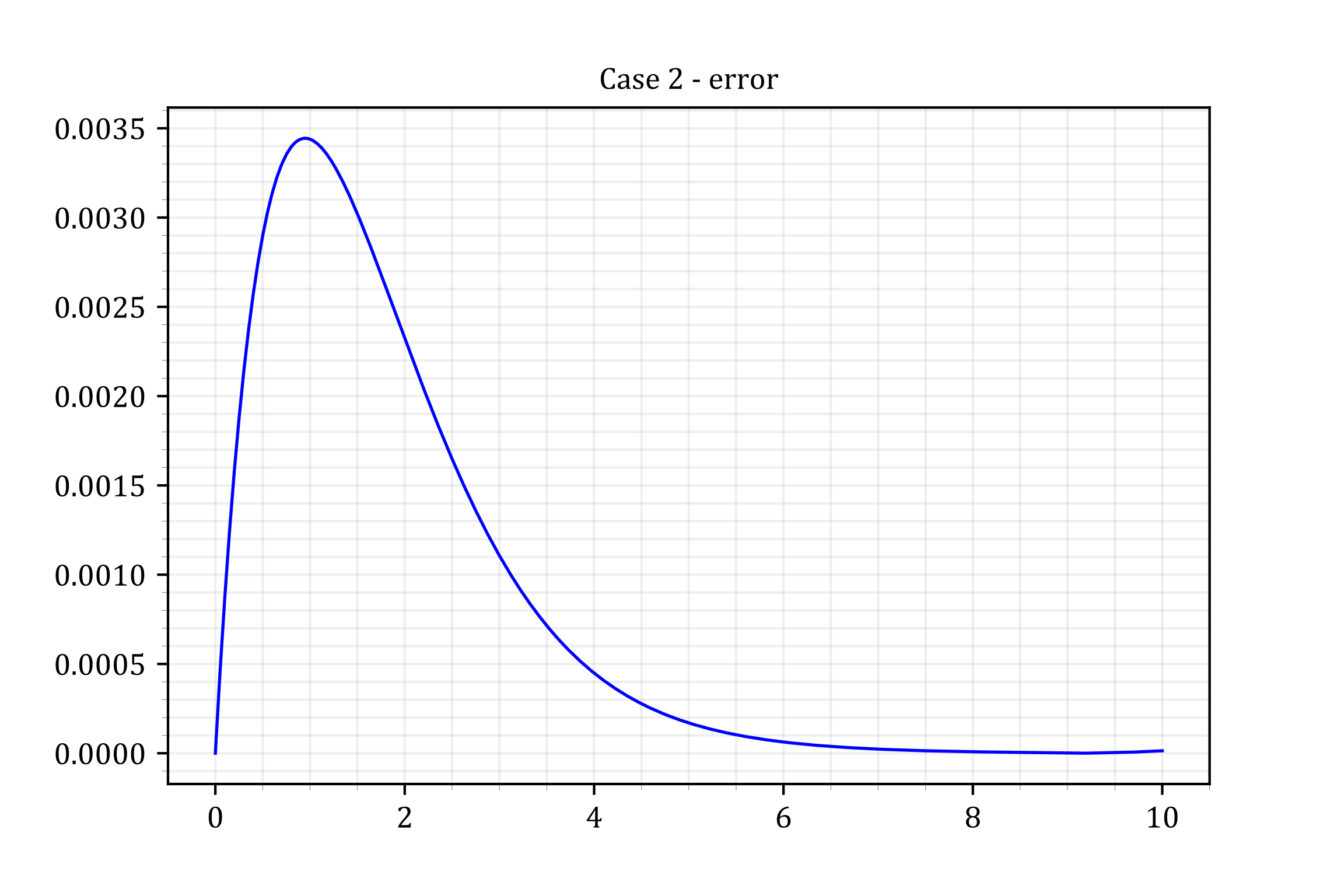
Factor and reorganize LHS/RHS:

BC @ defines the equation at

BC @

**SOLUTION – CASE 2**





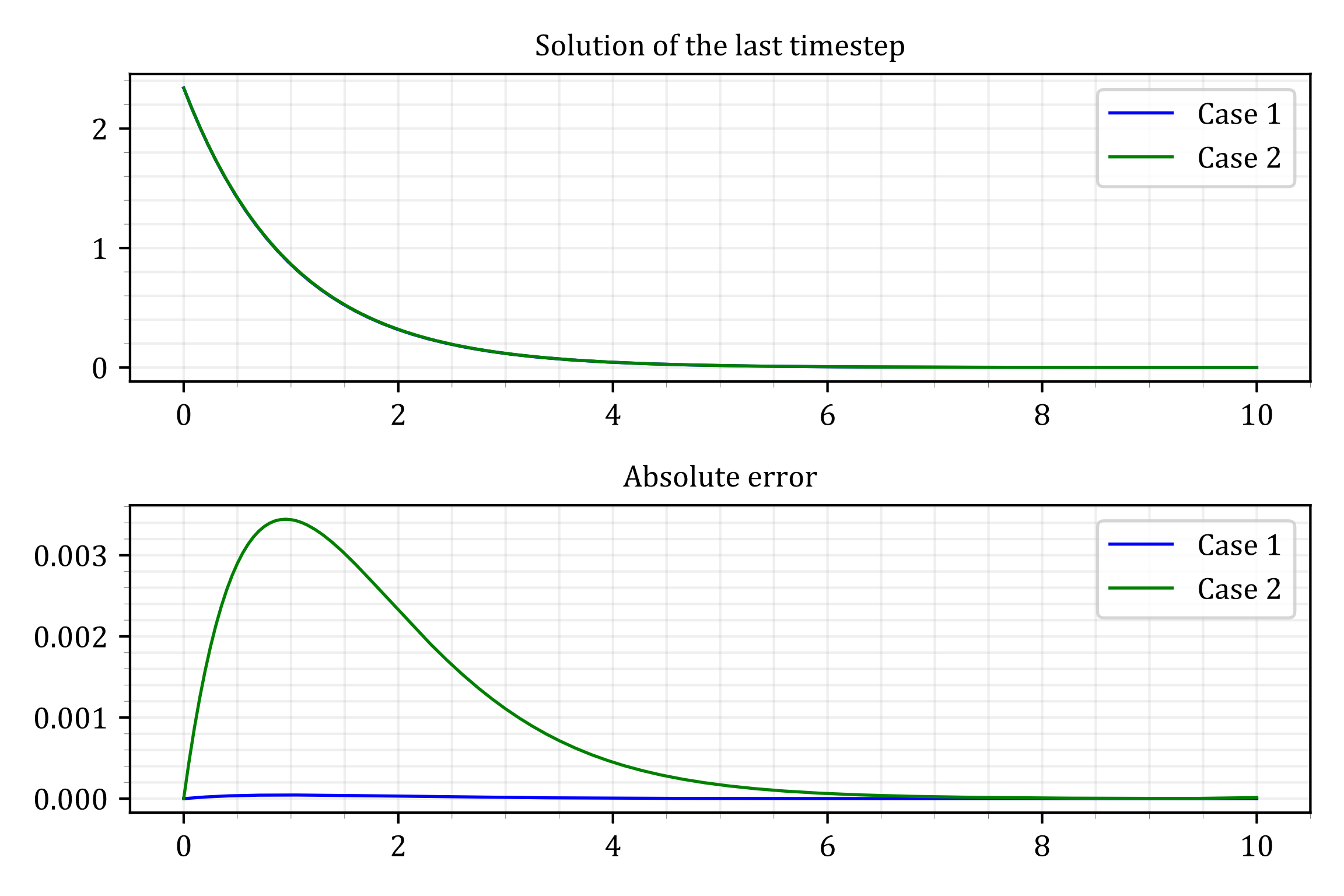
**MAXIMUM ABSOLUTE ERRORS (CASE 1 vs CASE 2)**

C1 Max err C2 Max err

==================================================

4.49684e-05 3.44407e-03

==================================================



**Code (Python)**

MAXX = 10

MAXT = 0.85

DX = 1 / 40

DT = 1 / 100

# CASE 1

from math import factorial, pi, sin, ceil

import numpy as np

from numpy import exp, linspace, vectorize

import matplotlib.pyplot as plt

plt.style.use('paper.mplstyle')

X = np.arange(0, 10 + DX, DX)

T = np.arange(0, 0.85 + DT, DT)

nx = len(X)

nt = len(T)

# Set initial condition and BC@X=0

Uni = np.zeros( (nt, nx) )

Uni[0,:] = np.exp( - X )

Uni[:,0] = np.exp( T )

EXACT\_Uni = np.zeros( (nt, nx) )

for n in np.arange( 0, nt ) :

EXACT\_Uni[n,:] = np.exp( T[n] - X )

for n in np.arange( 0, nt-1 ) :

K = np.zeros( (nx, nx) )

F = np.zeros( nx )

for i in np.arange( 1, nx ) :

K[i, i-1] += - 1/2/DX - 1/DX/DX

K[i, i] += 2/DT + 2/DX/DX + 1

F[i] += Uni[n,i-1] \* ( 1/2/DX + 1/DX/DX )

F[i] += Uni[n,i ] \* ( 2/DT - 2/DX/DX - 1 )

# BC @ i=0

if i == 1 :

F[i] -= (- 1/2/DX - 1/DX/DX) \* Uni[n+1,i-1]

if i < nx-1 :

K[i, i+1] += 1/2/DX - 1/DX/DX

F[i] += Uni[n,i+1] \* ( -1/2/DX + 1/DX/DX )

# BC @ i=N

else :

K[i, i-1] += 1/2/DX - 1/DX/DX

K[i, i] += (-2\*DX) \* ( 1/2/DX - 1/DX/DX )

F[i] += ( Uni[n,i-1] - 2\*DX\*Uni[n,i] ) \* ( - 1/2/DX + 1/DX/DX )

# Remove i=0

K=K[1:nx,1:nx]

F=F[1:nx]

U = np.linalg.solve(K,F)

Uni[n+1,1:nx] = U

# PLOT

import matplotlib.pyplot as plt

import numpy as np

from matplotlib import cm

range = np.arange(nt-1)[::10]

# range = np.arange(2)

colors = cm.get\_cmap('gray', len(range))

print(colors)

i=0

for n in range :

evr = 10

c=colors(n/nt\*.8)

plt.plot( X[::evr], EXACT\_Uni[n,::evr], color=c )

plt.scatter( X[::evr], Uni[n,::evr], color=c, s=25, marker='x' )

plt.title("Case 1")

plt.savefig("Case\_1.svg")

C1\_LAST = Uni[-1,:]

C1\_ERR = abs(EXACT\_Uni[-1,:] - Uni[-1,:])

plt.plot(X,C1\_ERR)

plt.title("Case 1 - error")

plt.savefig("Case\_1-Err.svg")

# CASE 2

from math import factorial, pi, sin, ceil

import numpy as np

from numpy import exp, linspace, vectorize

import matplotlib.pyplot as plt

plt.style.use('paper.mplstyle')

X = np.arange(0, 10 + DX, DX)

T = np.arange(0, 0.85 + DT, DT)

nx = len(X)

nt = len(T)

# Set initial condition and BC@X=0

Uni = np.zeros( (nt, nx) )

Uni[0,:] = np.exp( - X )

Uni[:,0] = np.exp( T )

EXACT\_Uni = np.zeros( (nt, nx) )

for n in np.arange( 0, nt ) :

EXACT\_Uni[n,:] = np.exp( T[n] - X )

for n in np.arange( 0, nt-1 ) :

K = np.zeros( (nx, nx) )

F = np.zeros( nx )

for i in np.arange( 1, nx ) :

K[i, i-1] += - 1/DX - 1/DX/DX

K[i, i] += 1/DX + 2/DT + 2/DX/DX + 1

F[i] += Uni[n,i-1] \* ( 1/DX + 1/DX/DX )

F[i] += Uni[n,i ] \* ( -1/DX + 2/DT - 2/DX/DX - 1 )

# BC @ i=0

if i == 1 :

F[i] += (1/DX/DX + 1/DX) \* Uni[n+1,i-1]

if i < nx-1 :

K[i, i+1] += - 1/DX/DX

F[i] += Uni[n,i+1] \* ( 1/DX/DX )

# BC @ i=N

else :

K[i, i] += (1-DX) \* ( - 1/DX/DX )

F[i] += Uni[n,i] \* ( 1 - DX ) \* ( - 1/DX + 1/DX/DX )

# Remove i=0

K=K[1:nx,1:nx]

F=F[1:nx]

U = np.linalg.solve(K,F)

Uni[n+1,1:nx] = U

# PLOT

import matplotlib.pyplot as plt

import numpy as np

from matplotlib import cm

range = np.arange(nt-1)[::20]

colors = cm.get\_cmap('gray', len(range))

print(colors)

i=0

for n in range :

evr = 10

c=colors(n/nt\*.8)

plt.plot( X[::evr], EXACT\_Uni[n,::evr], color=c )

plt.scatter( X[::evr], Uni[n,::evr], color=c, s=25, marker='x' )

plt.title("Case 2")

plt.savefig("case2.svg")

C2\_LAST = Uni[-1,:]

C2\_ERR = abs(EXACT\_Uni[-1,:] - Uni[-1,:])

plt.plot(X,C2\_ERR)

plt.title("Case 2 - error")

plt.savefig("Case\_2-Err.svg")

# COMPARE 1 VS 2

fig, [ax1,ax2] = plt.subplots(2,1)

ax1.set\_title("Solution of the last timestep")

ax1.plot(X,C1\_LAST, label='Case 1')

ax1.plot(X,C2\_LAST, label='Case 2')

ax1.legend()

ax2.set\_title("Absolute error")

ax2.plot(X,C1\_ERR, label='Case 1')

ax2.plot(X,C2\_ERR, label='Case 2')

ax2.legend()

fig.tight\_layout()

C1\_MAX = max(C1\_ERR)

C2\_MAX = max(C2\_ERR)

print(f"{'C1 Max err':20s}{'C2 Max err':20s}")

print(50\*"=")

print(f"{C1\_MAX:10.5e}{C2\_MAX:-20.5e}")

print(50\*"=")

fig.savefig("err.svg")