In [41]:
```python
import numpy as np
import matplotlib.pyplot as plt
```

In [9]:
```python
f = lambda x : x**4 - 2*x + 1

def trap_int( f , a  , b , n ):
    h = ( b - a )/n
    x = np.linspace( a , b,  n )
    y = f( x )
    result = (.5)*( y[0] + y[-1])*h
    result += np.sum( y[1:-1] )*h
    return result

for n in range( 7 ):
    result = trap_int( f , 0 , 2 , 10**n )
    print( 'for n = ' , 10**n , ' result = ' ,\
        result , 'error = ' , round((result/4.4 - 1)*100 , 5 ) ,\
            '%' )
```

```
for n =  1  result =  2.0 error =  -54.54545 %
for n =  10  result =  4.078372199359853 error =  -7.30972 %
for n =  100  result =  4.357077430084268 error =  -0.97551 %
for n =  1000  result =  4.395610677342941 error =  -0.09976 %
for n =  10000  result =  4.3995601066773355 error =  -0.01 %
for n =  100000  result =  4.399956001066678 error =  -0.001 %
for n =  1000000  result =  4.399995600010665 error =  -0.0001 %
```

Assignment 6

Question 1

In [10]:
```python
def trapint( f , a,   b , res = 10000 ):
    x = np.linspace( a , b , res )
    y = f( x )
    steps = ( b - a )/res
    result = ( y[0] + y[-1])/2
    result += np.sum( y[1:-1] )
    return result*steps
# f -> function to integrate
# a -> lower limit
# b -> upper limit
# res -> number of points between the range a , b
```

In [11]:
```python
trapint( lambda x : x**4 - 2*x + 1, 0 , 2 )
```

Out[11]:
```
4.3995601066773355
```

In [22]:
```python
def blackbody( z , c ):
    temp = np.divide( c*z , 1 - z )
    result = ( c**4 )*( np.divide( np.power(z , 3 )*np.exp( - temp ) , np
    return result
```

In [23]:
```
def trapint( f , a,   b , c , res = 10000 ):
    x = np.linspace( a , b , res )
    y = f( x , c  )
    steps = ( b - a )/res
    result = ( y[0] + y[-1])/2
    result += np.sum( y[1:-1] )
    return result*steps
```

In [32]:
```
for c in range(1 ,  7 ):
    result = trapint( blackbody , 0  + 0.000001 , 1 - 0.00001 , c , res =
    acctual = np.pi**4/15
    error = ( result/acctual -1 )*100
    print(f'with c = {c}; acctual result = {acctual}; int result = {resul
```

```
with c = 1; acctual result = 6.493939402266828; int result = 6.4939387528
72889; Error = -9.999999972531981e-06%
with c = 2; acctual result = 6.493939402266828; int result = 6.4939387528
72886; Error = -1.0000000028043132e-05%
with c = 3; acctual result = 6.493939402266828; int result = 6.4939387528
72897; Error = -9.999999850407448e-06%
with c = 4; acctual result = 6.493939402266828; int result = 6.4939387528
72894; Error = -9.9999999059186e-06%
with c = 5; acctual result = 6.493939402266828; int result = 6.4939387528
72892; Error = -9.99999993922529e-06%
with c = 6; acctual result = 6.493939402266828; int result = 6.4939387528
728965; Error = -9.999999861509679e-06%
```

Question 2

$$\int_0^x e^{-t^2} dt$$

In [37]:
```
f = lambda x : np.exp( - np.square( x ) )

def SimpSon(f,a,b,steps=100000):
    if steps%2 == 1:
        raise ValueError( "n should be even!")
    x = np.linspace( a, b , steps )
    y = f( x )
    result = y[0] + y[-1]
    temp = y[1:-1]
    result += 4*np.sum(y[::2]) + 2*np.sum(y[1::2])
    return result*( ( b - a )/steps )/3
```

In [40]:
```
x = np.linspace( 0 , 3 , int(3/0.1) )
y = np.zeros( x.size )
for i in range( x.size ):
    y[i] = SimpSon( f , 0 , x[i] )
```

In [71]:

```
fig, ax = plt.subplots( figsize = ( 13 , 4 ))
plt.plot( x , y )
plt.grid()
plt.yticks( np.linspace( 0 ,1 , 11))
plt.ylabel(r'$\int_0^x e^{-t^2}dt$')
plt.xlabel(r'$x$')
plt.xticks( np.linspace( 0 , 3 , 11 ))
plt.show()
```