Namn och CID på gruppmedlemmar:

Blend Ahmed Omar (blend), Ebbe Ledin (ebbel), Albin Östling (ostlinga)

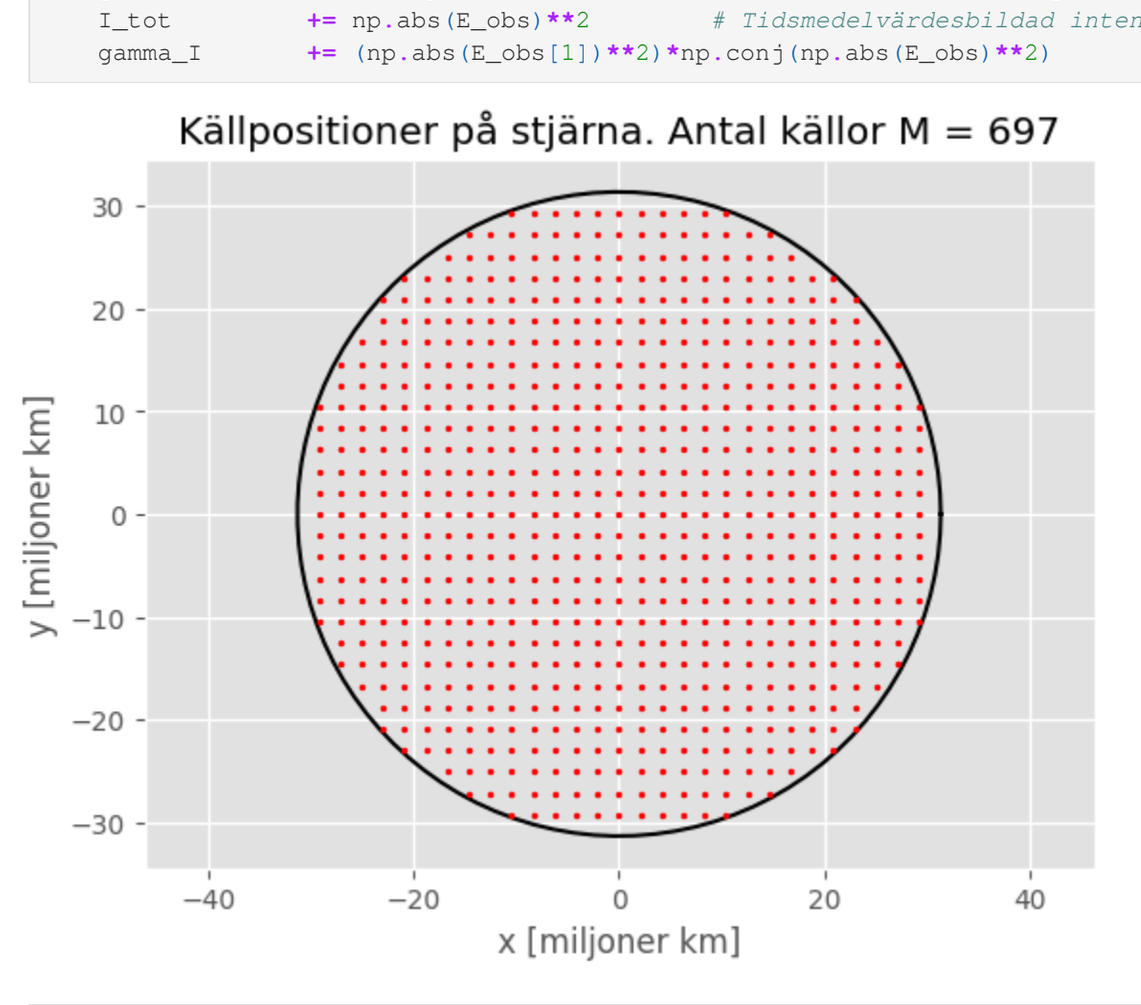
In [1]:
```python
import numpy as np
from numpy.matlib import repmat # Vi lånar funktionen repmat från matlib
from scipy import constants # Scipy constants innehåller alla möjliga fysikaliska konstanter
c = constants.speed_of_light # Vi använder det för att importera ljusets hastighet
import matplotlib.pyplot as plt
plt.style.use("ggplot")
```

In [2]:
```python
# Funktioner för HUPP:en

def xy_source(N, D_star, seperation):
    '''

    N:          Antalet observationspunkter (punkter längs u-axeln där fältet beräknas)
    D_star:     Stjärnans diameter [m]
    separation: Avstånd mellan punktkällorna på stjärnan [m],
                kan lämpligen anges som bråkdel av D_star,
                t.ex. separation=D_star/30

    x: matris med punktkällornas x-positioner [m]
    y: matris med punktkällornas y-positioner [m]
    M: punktkällornas antal

    '''

    x = np.arange(-D_star/2, D_star/2 + seperation, seperation) # Vektor med källpunkter i x-led
    y = x                                                        # och i y-led

    X, Y = np.meshgrid(x, y)
    R    = np.sqrt(X**2 + Y**2)                                  # Längd från origo till källpunkter

    element_inuti_diameter = R < (D_star/2)                      # Element innanför D_star

    x = X[element_inuti_diameter]                                # Plocka ut x-koordinater som är innanför D_star
    y = Y[element_inuti_diameter]                                # och plocka ut y-koordinater

    M = np.sum(element_inuti_diameter)                           # Totalt antal källor innanför D_star

    return (x, y, M)

def plot_xy_source(x ,y, M):
    '''
    Skapar en figur som visar hur xy-source är placerade

    '''
    theta = np.linspace(0, 2*np.pi, 100)                       # Rita en cirkel med diameter D_star
    x_circumference = D_star/2*np.cos(theta)
    y_circumference = D_star/2*np.sin(theta)


    miljon_km = 1e6*1e3                                        # Skala om axlar till miljoner km

    x_miljon_km = x/miljon_km
    y_miljon_km = y/miljon_km

    x_circumference_miljon_km = x_circumference/miljon_km
    y_circumference_miljon_km = y_circumference/miljon_km

    plt.figure()
    plt.axis('equal')

    plt.plot(x_circumference_miljon_km, y_circumference_miljon_km, 'black')
    plt.plot(x_miljon_km, y_miljon_km, 'ro', markersize=1.4)

    plt.title(r'Källpositioner på stjärna. Antal källor M = ' + str(M))
    plt.xlabel('x [miljoner km]')
    plt.ylabel('y [miljoner km]')
```

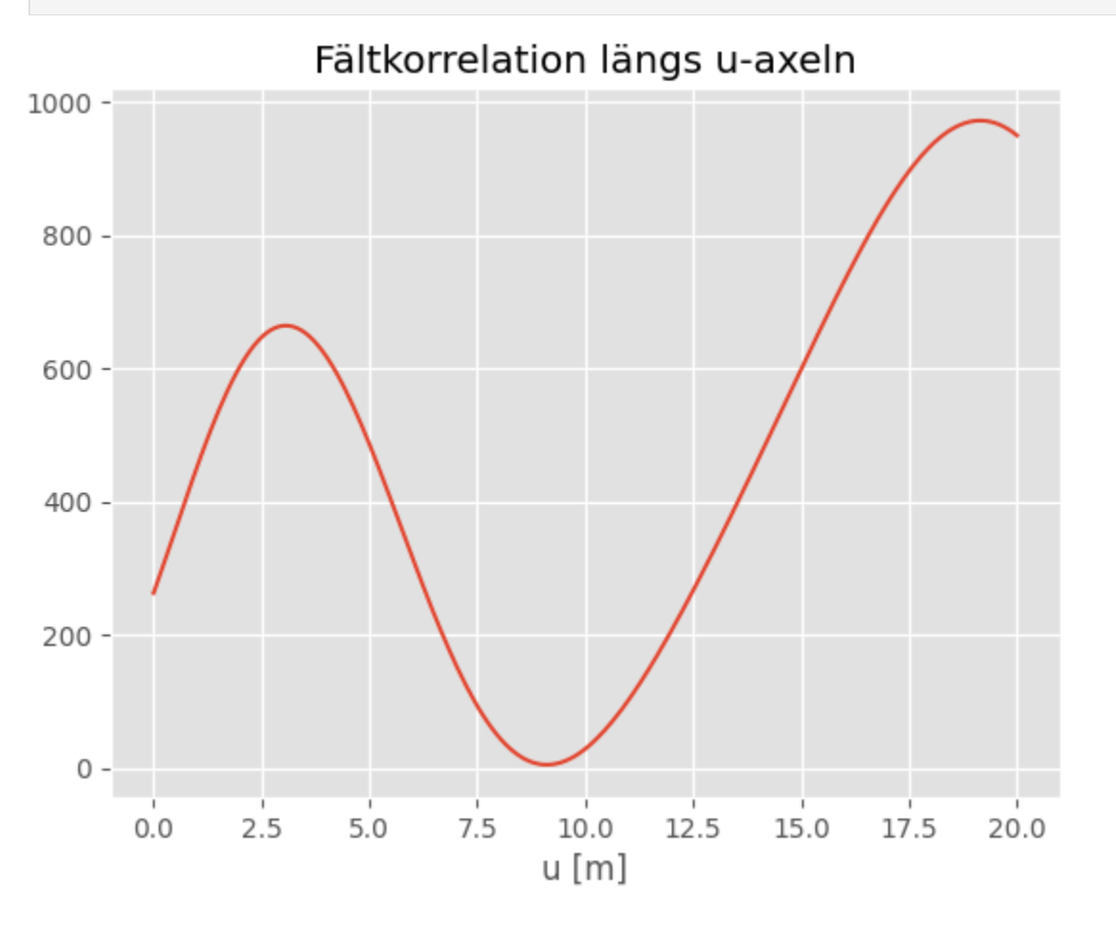
Uppgift a

Beräkna korrelationen $\Gamma_{AB}(u)$ för positioner hos punkt B från $u = 0$ upp till $u = 20$ meter.

In [3]:
```python
#%% Definiera variabler och generera källor %%#

sek_per_ar = 365.24*24*60*60                           # Sekunder på ett år
L_70  = 70*sek_per_ar*c                                # 70 ljusår  [m]
L_300 = 300*sek_per_ar*c                               # 300 ljusår [m]

lam0  = 650e-9                                         # Våglängd [m]
k0    = 2*np.pi/lam0                                   # Vågtal   [1/m]

### Genererar källor från stjärna ###
D_sol         = 1392700e3
D_star        = 45*D_sol                               # Stjärnas diameter [m]
separation    = D_star/30                              # Separation mellan källor på stjärna

N = 200                                                # Punkter längs u-axeln
x, y, M = xy_source(N, D_star, separation)             # Generera x och y positioner för källor
plot_xy_source(x ,y, M)                                # Plotta stjärna med källor

### Punkter som observeras ###
u = np.linspace(0, 20, N)
u = u.reshape(N, 1)                                    # Reshape till kolonvektor

### Skapa repeterade matriser för att undvika loopar ###
x_repeterad = repmat(x, N, 1)
y_repeterad = repmat(y, N, 1)
u_repeterad = repmat(u, 1, M)

### Distans till observations punkter ###
r = -(x_repeterad/L_70)*u_repeterad                                        # Ledning finns i HUPP-beskrivning

#%% Summera koherenstider %##
### Initiera variabler ###
gamma   = 0                # Korrelation <E(0,0)conj(E(u,0))>
I_tot   = 0                # Tidsmedelvärdesbildad intensitet
gamma_I = 0                # Intensitetskorrelation <I(0,0)conj(I(u,0))>

### Loopa över w koherenstider ###
iterationer = 1000          # Antal iterationer/koherenstider. Använda >200 minst!
for i in range(iterationer):
    ### Generera slumpmässig fas ###
    fas           = 2*np.pi*np.random.rand(M, 1)
    fas_repeterad = np.transpose(repmat(fas, 1, N))

    ### Observerade värden ###
    E_k_obs       = np.exp(1j*(fas_repeterad + k0*r))        # Given i HUPP-beskrivning
    E_obs         = np.sum(E_k_obs,1)        # Given i HUPP-beskrivning
    inst_produkt  = E_obs[0]*np.conj(E_obs)        # Given i HUPP-beskrivning

    gamma         += inst_produkt        # E-fältets korrelation (amplitud o fas), summera inst_produkt i varje iteration!
    I_tot         += np.abs(E_obs)**2        # Tidsmedelvärdesbildad intensitet, summera np.abs(E_obs)**2 över alla koherenstider
    gamma_I       += (np.abs(E_obs[1])**2)*np.conj(np.abs(E_obs)**2)        # Intensitetskorrelation, samma som Gamma fast med I = np.abs(E_obs(1))**2 och np.abs(E_obs)**2
```

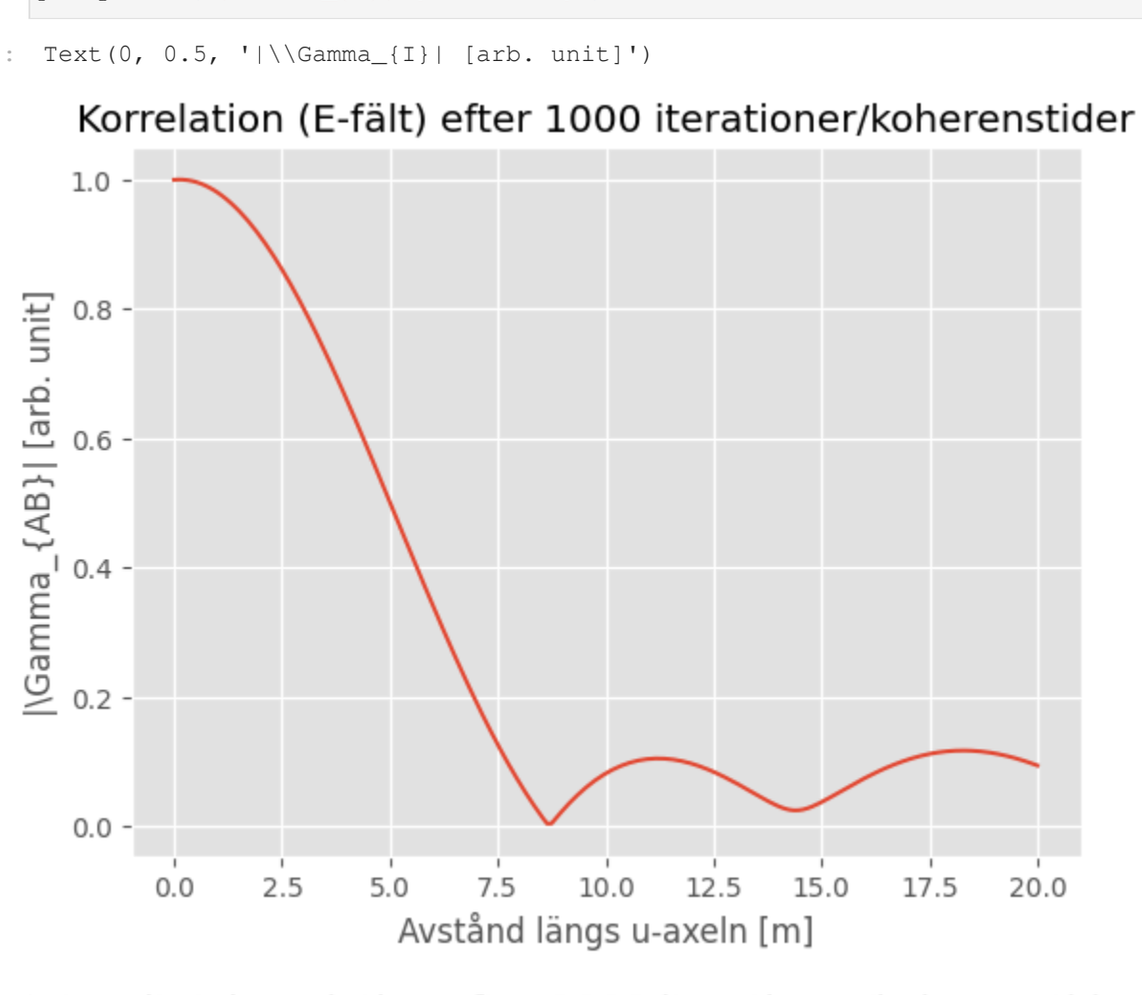


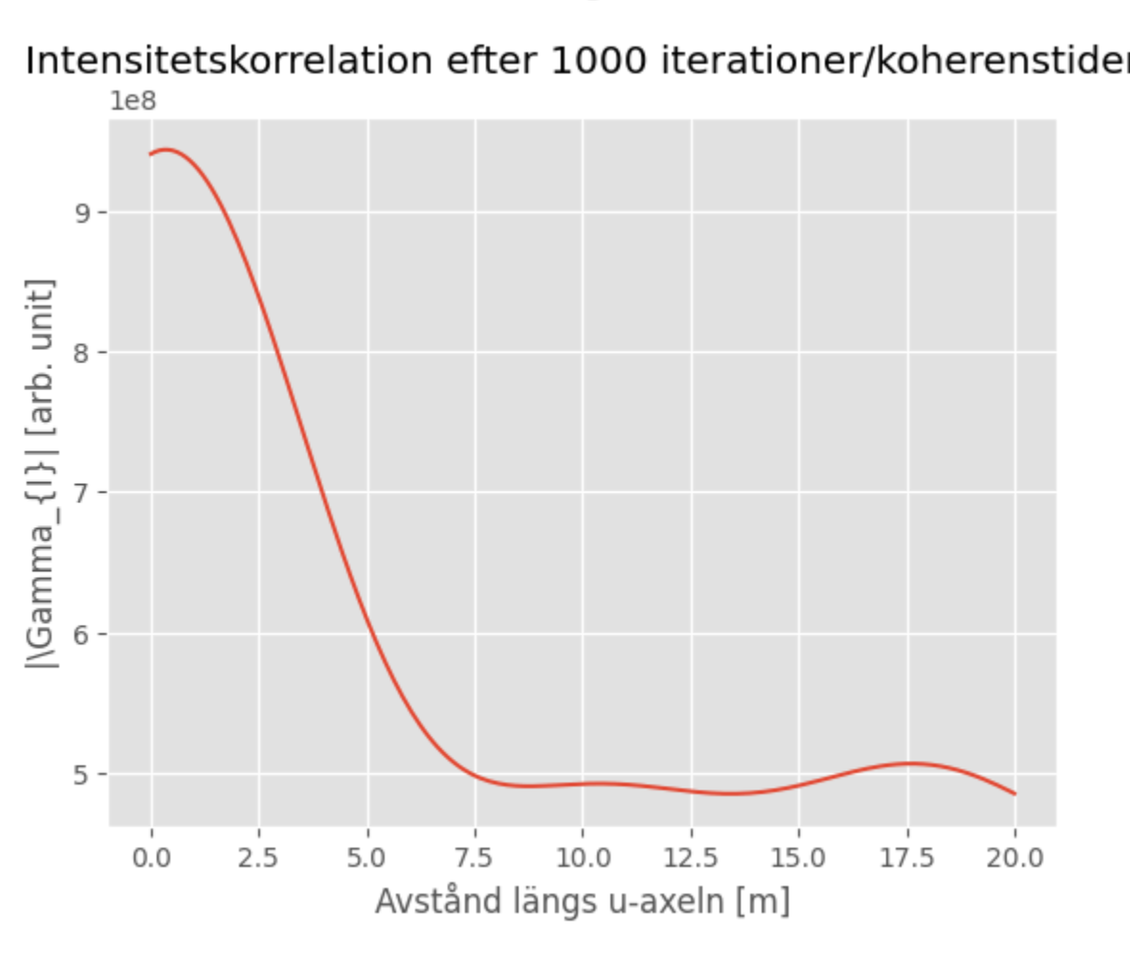
In [4]:
```python
I_obs_inst = np.abs(E_obs)**2

plt.figure()
plt.plot(u.flatten(), np.real(I_obs_inst))
plt.xlabel('u [m]')
plt.title('Fältkorrelation längs u-axeln')
plt.grid(True)
plt.show()
```



In [5]:
```python
gamma_norm = np.abs(gamma/np.max(gamma))

plt.figure()
plt.plot(u, gamma_norm)
plt.title(r'Korrelation (E-fält) efter ' + str(iterationer) + ' iterationer/koherenstider')
plt.xlabel (r'Avstånd längs u-axeln [m]')
plt.ylabel(r'|\Gamma_{AB}| [arb. unit]')

gamma_I_norm = np.abs(gamma_I/np.max(gamma_I))

plt.figure()
plt.plot(u, gamma_I)
plt.title('Intensitetskorrelation efter ' + str(iterationer) + ' iterationer/koherenstider' )
plt.xlabel ('Avstånd längs u-axeln [m]')
plt.ylabel('|\Gamma_{I}| [arb. unit]')
```

Out[5]:
```
Text(0, 0.5, '|\\Gamma_{I}| [arb. unit]')
```



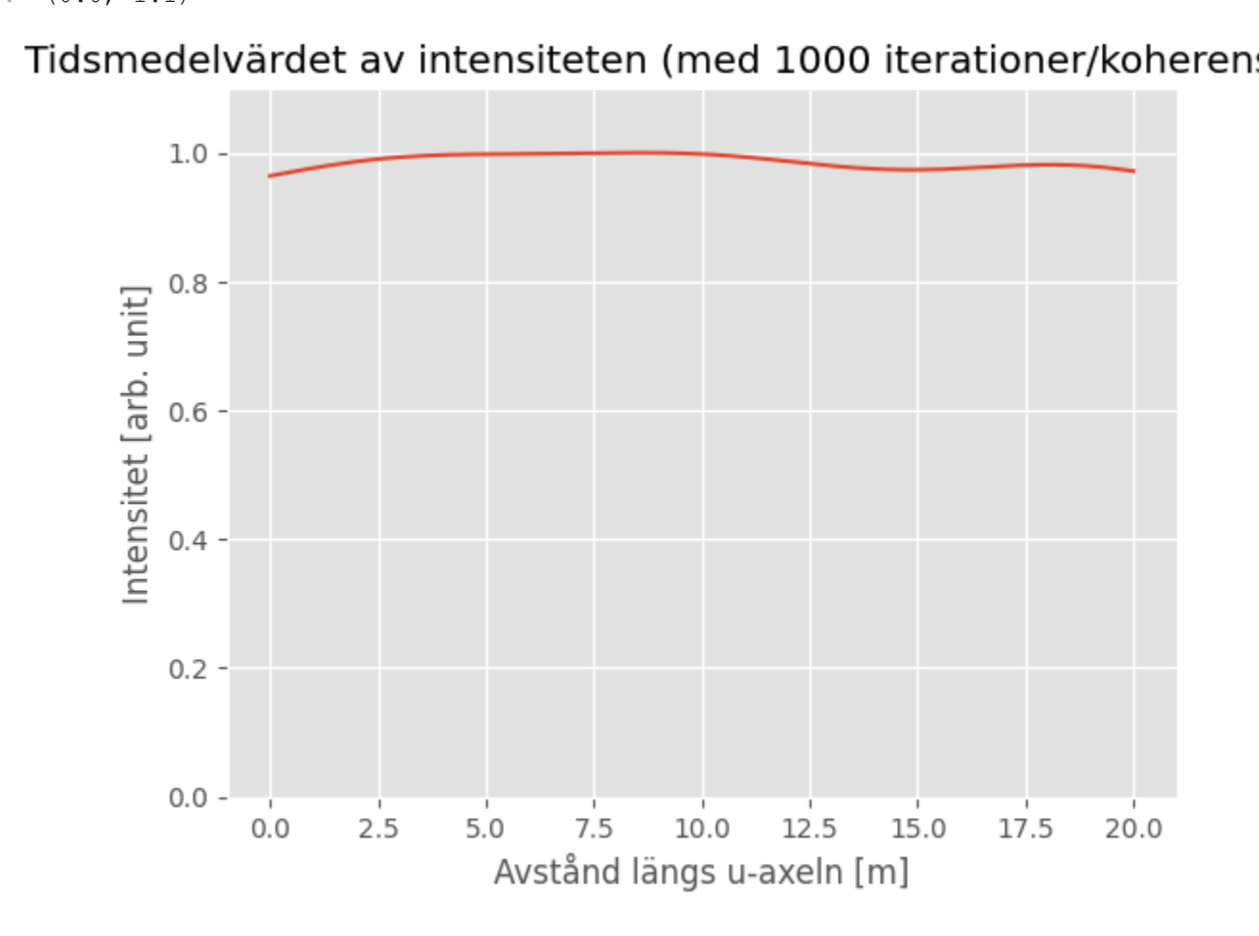


Uppgift b

Som en "sanity check", beräkna också intensiteten, i vanlig mening, i observationspunkterna (sånär som på en ointressant konstant).

In [6]:
```python
I_tot_norm = np.abs(I_tot/np.max(I_tot))

plt.figure()
plt.plot(u, I_tot_norm)
plt.title('Tidsmedelvärdet av intensiteten (med ' + str(iterationer) + ' iterationer/koherenstider)' )
plt.xlabel ('Avstånd längs u-axeln [m]')
plt.ylabel('Intensitet [arb. unit]')

plt.ylim([0, 1.1])
```

Out[6]:
```
(0.0, 1.1)
```



Är intensitetens variation längs u-axeln vad du förväntar dig?

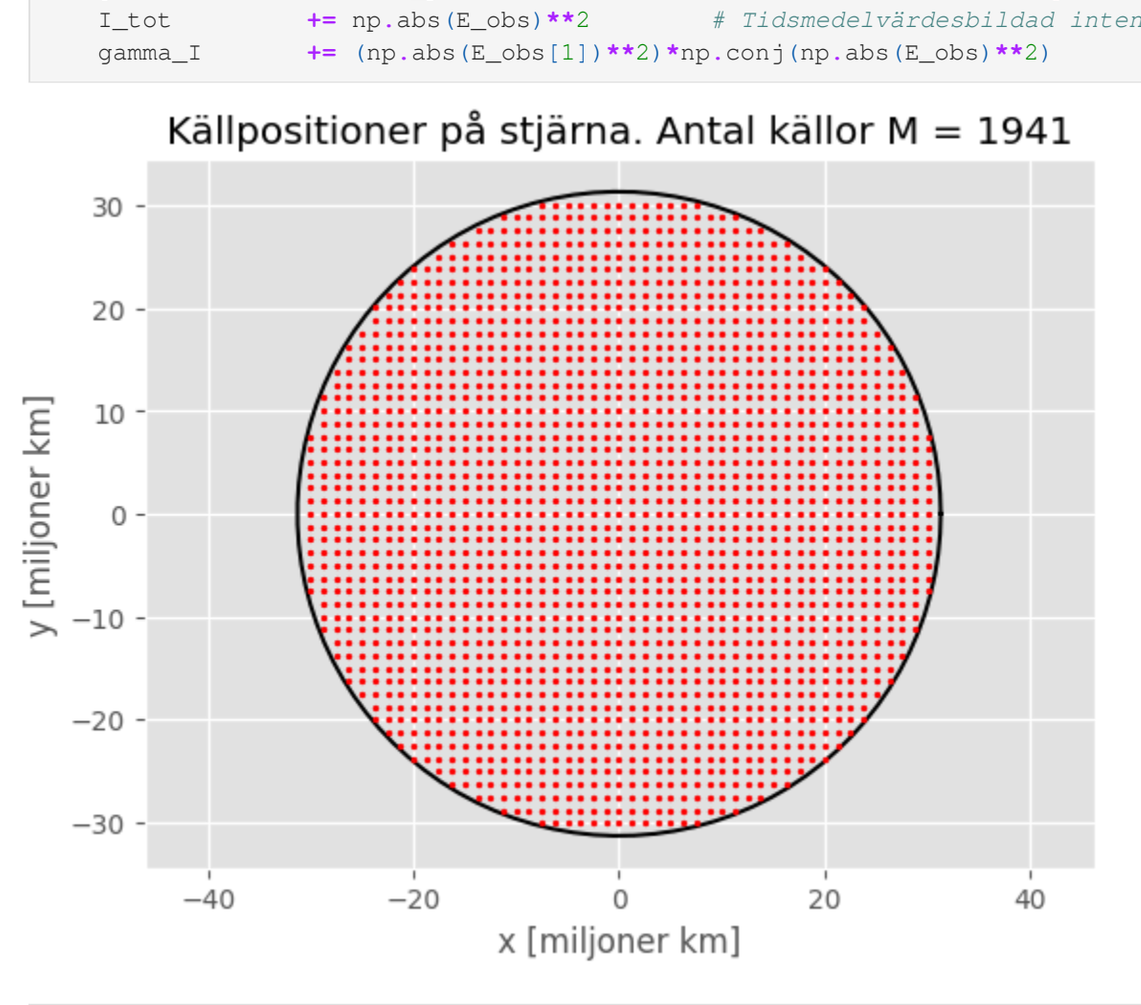
Ja, vi förväntade oss att det inte skulle variera så mycket på grund av att vi har många oberoende punktkällor med slumpmässig fas.

Uppgift c

Kolla att du samplat punktkällorna tillräckligt tätt för att representera stjärnan på ett bra sätt. Gör detta genom att kolla att du får liknande resultat som i **(a)** även om du använder ett väsentligt annorlunda antal punktkällor.

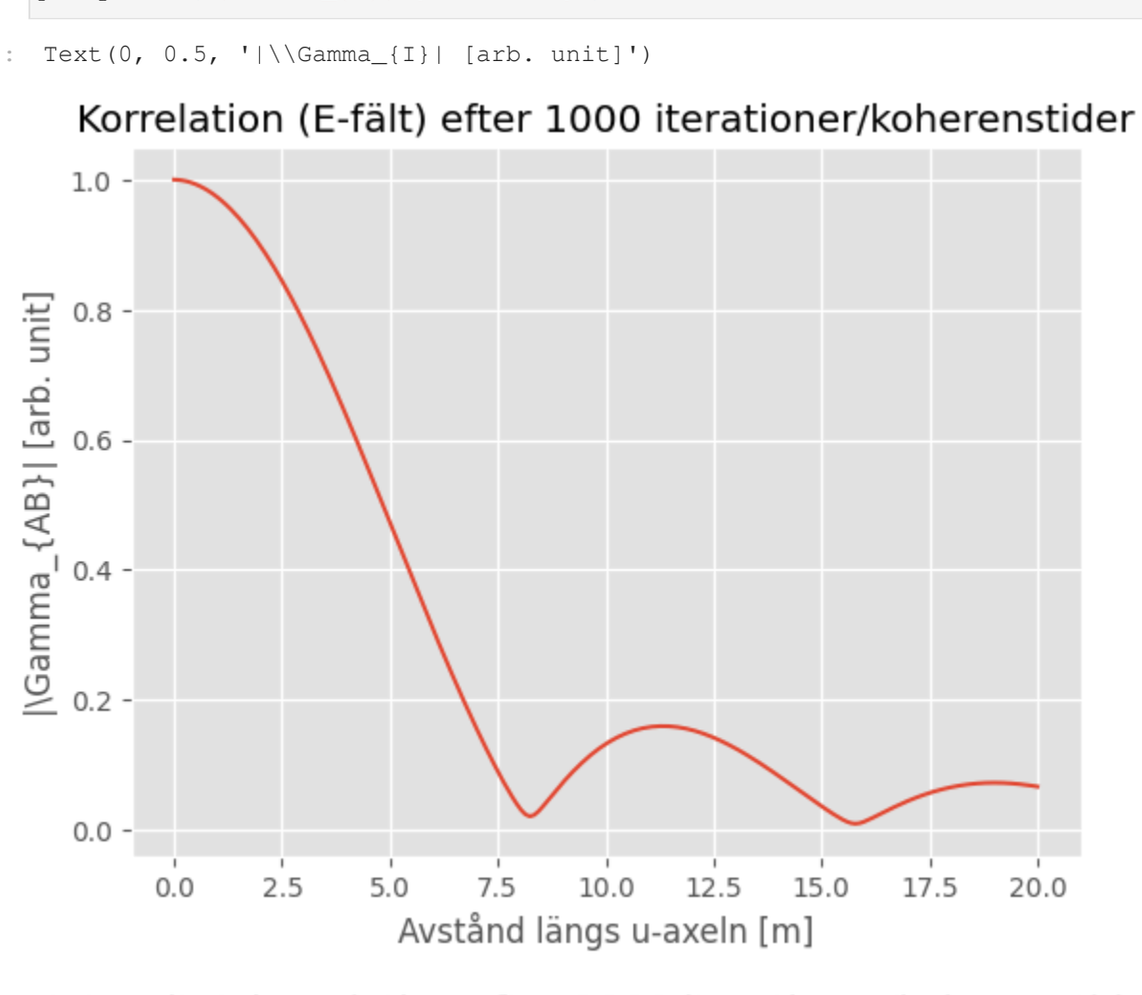
In [7]:
```python
# KOD
```

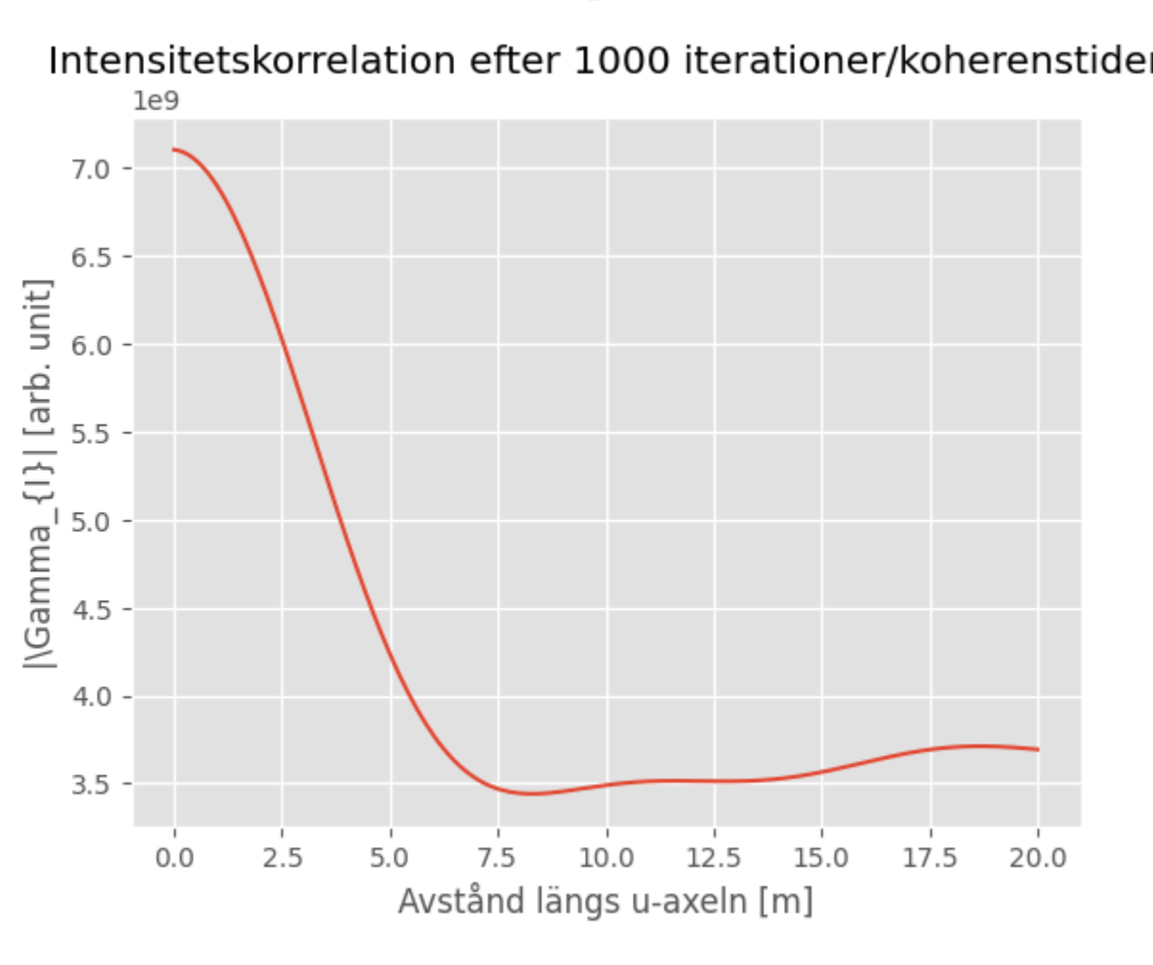
In [8]:
```python
#%% Definiera variabler och generera källor %%#

sek_per_ar = 365.24*24*60*60                           # Sekunder på ett år
L_70  = 70*sek_per_ar*c                                # 70 ljusår  [m]
L_300 = 300*sek_per_ar*c                               # 300 ljusår [m]

lam0  = 650e-9                                         # Våglängd [m]
k0    = 2*np.pi/lam0                                   # Vågtal   [1/m]

### Genererar källor från stjärna ###
D_sol         = 1392700e3
D_star        = 45*D_sol                               # Stjärnas diameter [m]
separation    = D_star/50                              # Separation mellan källor på stjärna

N = 200                                                # Punkter längs u-axeln
x, y, M = xy_source(N, D_star, separation)             # Generera x och y positioner för källor
plot_xy_source(x ,y, M)                                # Plotta stjärna med källor

### Punkter som observeras ###
u = np.linspace(0, 20, N)
u = u.reshape(N, 1)                                    # Reshape till kolonvektor

### Skapa repeterade matriser för att undvika loopar ###
x_repeterad = repmat(x, N, 1)
y_repeterad = repmat(y, N, 1)
u_repeterad = repmat(u, 1, M)

### Distans till observations punkter ###
r = -(x_repeterad/L_70)*u_repeterad                                        # Ledning finns i HUPP-beskrivning

#%% Summera koherenstider %##
### Initiera variabler ###
gamma   = 0                # Korrelation <E(0,0)conj(E(u,0))>
I_tot   = 0                # Tidsmedelvärdesbildad intensitet
gamma_I = 0                # Intensitetskorrelation <I(0,0)conj(I(u,0))>

### Loopa över w koherenstider ###
iterationer = 1000          # Antal iterationer/koherenstider. Använda >200 minst!
for i in range(iterationer):
    ### Generera slumpmässig fas ###
    fas           = 2*np.pi*np.random.rand(M, 1)
    fas_repeterad = np.transpose(repmat(fas, 1, N))

    ### Observerade värden ###
    E_k_obs       = np.exp(1j*(fas_repeterad + k0*r))        # Given i HUPP-beskrivning
    E_obs         = np.sum(E_k_obs,1)        # Given i HUPP-beskrivning
    inst_produkt  = E_obs[0]*np.conj(E_obs)        # Given i HUPP-beskrivning

    gamma         += inst_produkt        # E-fältets korrelation (amplitud o fas), summera inst_produkt i varje iteration!
    I_tot         += np.abs(E_obs)**2        # Tidsmedelvärdesbildad intensitet, summera np.abs(E_obs)**2 över alla koherenstider
    gamma_I       += (np.abs(E_obs[1])**2)*np.conj(np.abs(E_obs)**2)        # Intensitetskorrelation, samma som Gamma fast med I = np.abs(E_obs(1))**2 och np.abs(E_obs)**2
```



In [9]:
```python
gamma_norm = np.abs(gamma/np.max(gamma))

plt.figure()
plt.plot(u, gamma_norm)
plt.title(r'Korrelation (E-fält) efter ' + str(iterationer) + ' iterationer/koherenstider')
plt.xlabel (r'Avstånd längs u-axeln [m]')
plt.ylabel(r'|\Gamma_{AB}| [arb. unit]')

gamma_I_norm = np.abs(gamma_I/np.max(gamma_I))

plt.figure()
plt.plot(u, gamma_I)
plt.title('Intensitetskorrelation efter ' + str(iterationer) + ' iterationer/koherenstider' )
plt.xlabel ('Avstånd längs u-axeln [m]')
plt.ylabel('|\Gamma_{I}| [arb. unit]')
```

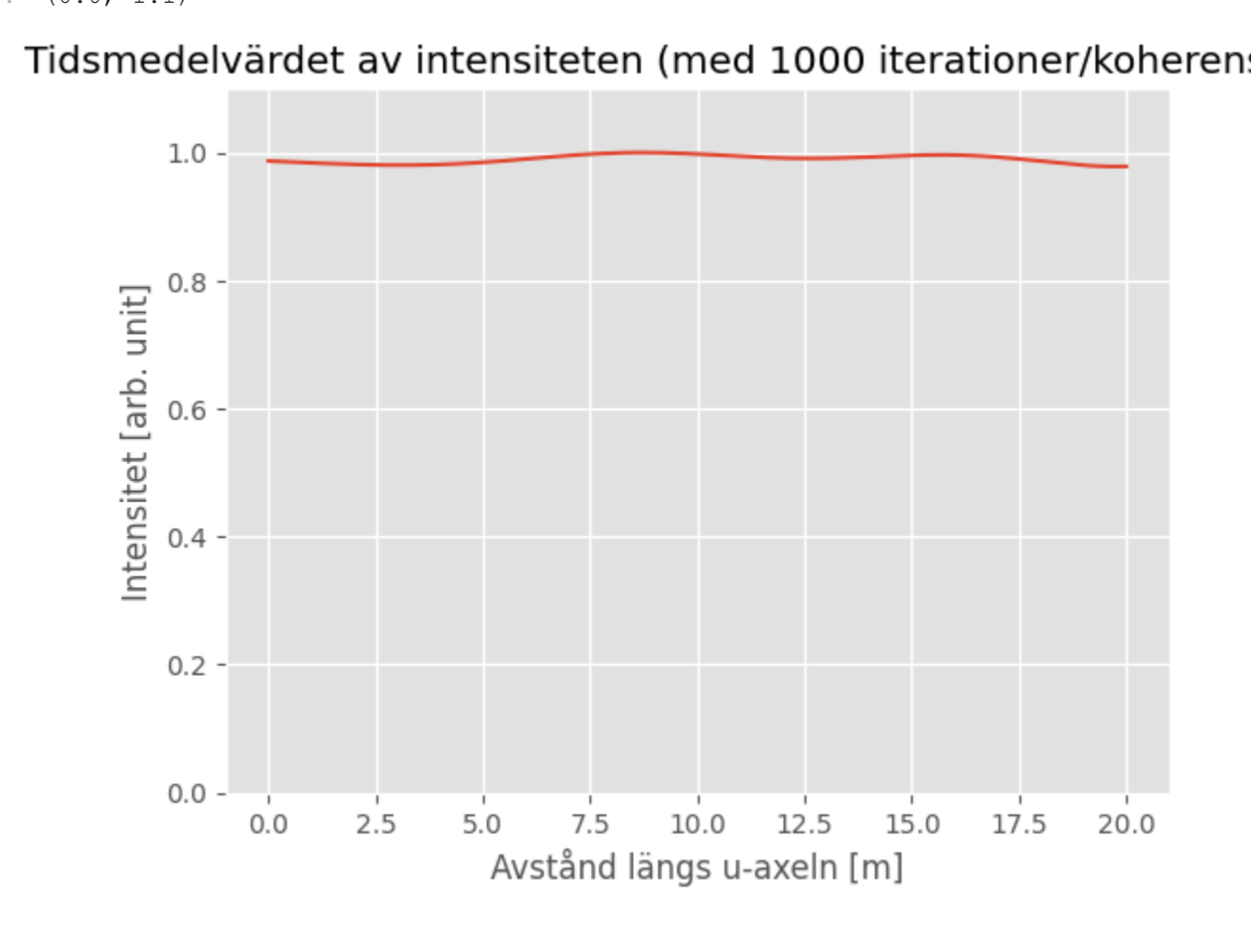
Out[9]:
```
Text(0, 0.5, '|\\Gamma_{I}| [arb. unit]')
```





In [10]:
```python
I_tot_norm = np.abs(I_tot/np.max(I_tot))

plt.figure()
plt.plot(u, I_tot_norm)
plt.title('Tidsmedelvärdet av intensiteten (med ' + str(iterationer) + ' iterationer/koherenstider)' )
plt.xlabel ('Avstånd längs u-axeln [m]')
plt.ylabel('Intensitet [arb. unit]')

plt.ylim([0, 1.1])
```

Out[10]:
```
(0.0, 1.1)
```



Uppgift d

Om spatiella koherenslängden l_s i detta specialfall (d.v.s. cirkulär inkoherent ljuskälla) mer precist definieras som avståndet mellan två punkter på u-axeln då korrelationen mellan deras fält blir noll (första nollstället), vad ger din simulering att $const$ har för värde i formeln:

$$l_s = const \frac{\lambda}{D_{källa}} L$$

Testa också ett annat värde på diametern hos ljuskällan, $D_{källa}$ för att bekräfta formeln.

In [11]:
```python
# KOD
def const(D_star, L_70, lam0, k0):
    N = 200                                                # Punkter längs u-axeln
    x, y, M = xy_source(N, D_star, separation)             # Generera x och y positioner för källor                              # Plotta stjärna med källor

    ### Punkter som observeras ###
    u = np.linspace(0, 20, N)
    u = u.reshape(N, 1)                                    # Reshape till kolonvektor

### Skapa repeterade matriser för att undvika loopar ###
    x_repeterad = repmat(x, N, 1)
    y_repeterad = repmat(y, N, 1)
    u_repeterad = repmat(u, 1, M)

### Distans till observations punkter ###
    r = -(x_repeterad/L_70)*u_repeterad                                        # Ledning finns i HUPP-beskrivning

#%% Summera koherenstider %##
### Initiera variabler ###
    gamma   = 0                # Korrelation <E(0,0)conj(E(u,0))>
    I_tot   = 0                # Tidsmedelvärdesbildad intensitet
    gamma_I = 0                # Intensitetskorrelation <I(0,0)conj(I(u,0))>

### Loopa över w koherenstider ###
    iterationer = 1000          # Antal iterationer/koherenstider. Använda >200 minst!
    for i in range(iterationer):
    ### Generera slumpmässig fas ###
        fas           = 2*np.pi*np.random.rand(M, 1)
        fas_repeterad = np.transpose(repmat(fas, 1, N))

    ### Observerade värden ###
        E_k_obs       = np.exp(1j*(fas_repeterad + k0*r))        # Given i HUPP-beskrivning
        E_obs         = np.sum(E_k_obs,1)        # Given i HUPP-beskrivning
        inst_produkt  = E_obs[0]*np.conj(E_obs)        # Given i HUPP-beskrivning

        gamma         += inst_produkt        # E-fältets korrelation (amplitud o fas), summera inst_produkt i varje iteration!
        I_tot         += np.abs(E_obs)**2        # Tidsmedelvärdesbildad intensitet, summera np.abs(E_obs)**2 över alla koherenstider
        gamma_I       += (np.abs(E_obs[1])**2)*np.conj(np.abs(E_obs)**2)


    gamma_norm = np.real(gamma/np.max(gamma))
    index = np.where(gamma_norm <= 0)[0][0]
    l_s = u.flatten()[index]

    const = l_s*D_star/(lam0*L_70)

    return const

print(const(D_star,L_70,lam0,k0))
print(const(D_star/2,L_70,lam0,k0))
```

```
1.2291421688700173
1.214509524002517
```

Uppgift e

In [12]:
```python
# koherenslängden l_s = ? är utläst från bilden i texten till HUPP 4
# första nollstället när ränderna är svåra att urskilja, finns två ställen
# på bilden i HUPP-beskrivningen

#Både l_s=2.98 och l_s=3.36 har både mycket svårt urskiljda ränder, vi väljer att ta ett l_s =3.36, då den är mycket svårare att se ränderna på. Däremot skulle vi gissa på att ett mer exakt värde ligger lite under dett

l_s1 = 3.36

D_star_e = 1.20*lam0*L_300/l_s1
# D_star i antal soldiametrar
D_star_e/(2*696342e3)

print(D_star_e)
print(D_star_e/(2*696342e3))
```

```
658854261828.5474
473.08238037383023
```

Vad fick man för värde på stjärndiametern? Jämför med solens diameter.

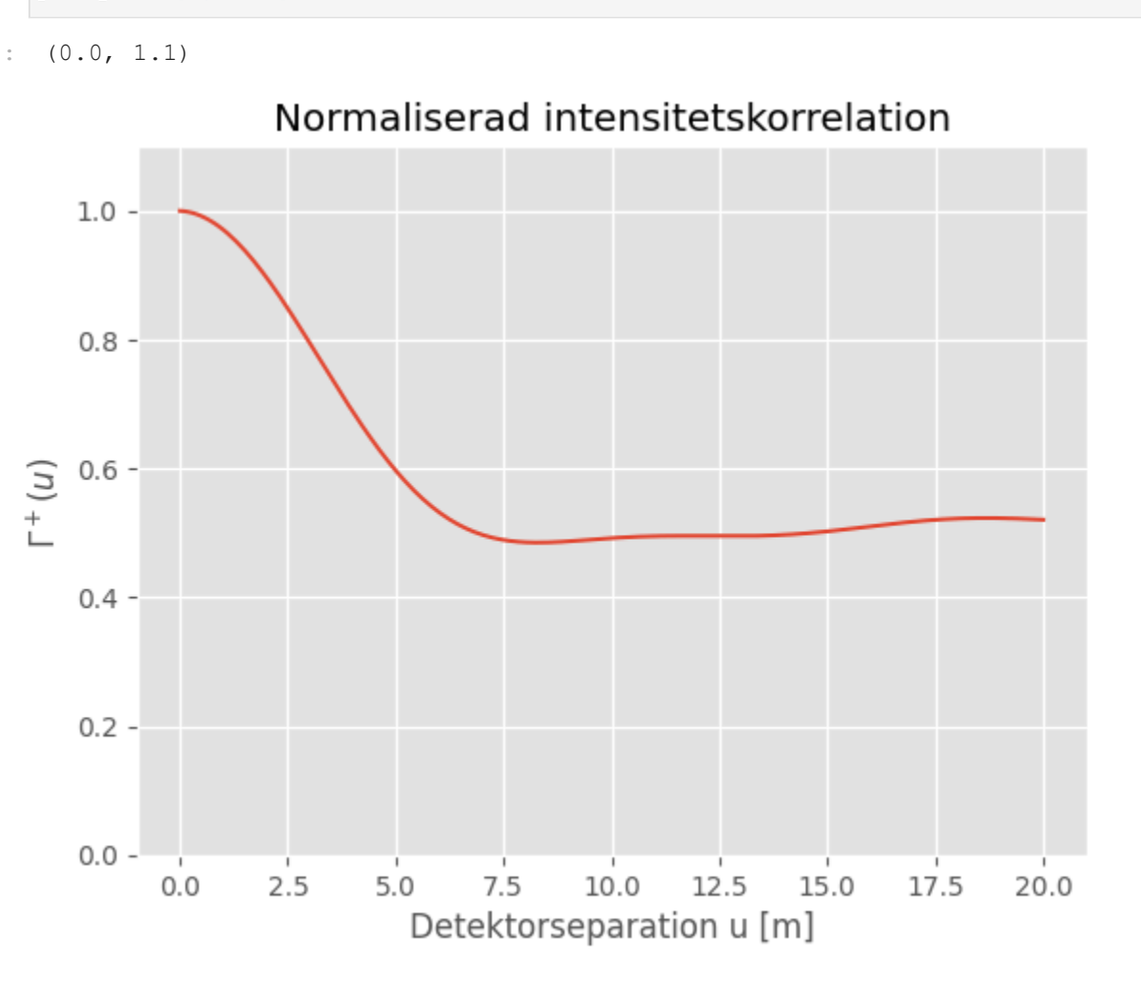
Med vår uppskattning av l_s så är stjärndiametern ungefär 470 gånger större än solens diameter! En diameter på 658854261828.5474 meter.

Uppgift f

Visa att de upprörda fysikerna hade fel! Återanvänd koden från **(b)**

In [13]:
```python
# kod

plt.figure()
plt.plot(u, gamma_I_norm)
plt.title('Normaliserad intensitetskorrelation' )
plt.xlabel ('Detektorseparation u [m]')
plt.ylabel(r'$\Gamma^+(u)$ ')

plt.ylim([0, 1.1])
```

Out[13]:
```
(0.0, 1.1)
```



Man kan tydligt se att intensitetskorrelationen faktiskt varierar med detektorseparationen u, vilket motstrider kvantfysikernas påståenden.