In [46]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
from scipy.integrate import RK45, odeint, solve_ivp
from scipy.special import expit
import math
plt.style.use(['science','notebook','grid'])
```

In [47]:

```
i = 0.0+1.0j; k1 = 1.3*10**12; k2 = 1.3*10**12;
w1 = 3*10**12;
w3 = 5*10**12; w2 = w3 - w1; d=10; c = 3*10**8;
A3 = 10**3+0.0j
kappa = np.sqrt(4*d**2*w1**2*w2**2/(k1*k2*c**4)*np.abs(A3)**2)
k3 = 2.60*10**12+4*kappa;

K1 = 2*1j*w1**2*d/(k1*c**2)
K2 = 2*1j*w2**2*d/(k2*c**2)
K3 = 2*1j*w3**2*d/(k3*c**2)




delta_k1 = 0.0203
delta_k2 = 0.75*delta_k1
delta_k3 = 0.95*delta_k1
delta_k4 = 1.05*delta_k1
delta_k5 = 1.25*delta_k1
```

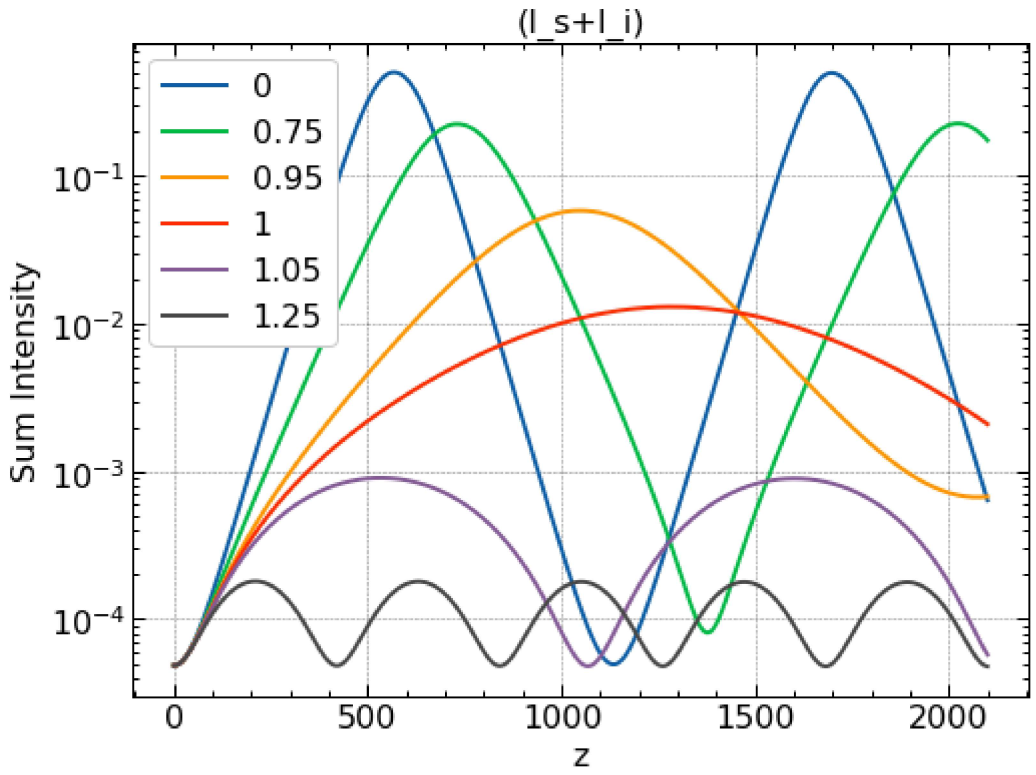
In [48]:

```
def dA2dz(z,S):
    #Constants above. I made them simple, but will probably need to know reasonable val
    A1, A2, A3 = S
    return (K1*A3*np.conj(A2), K2*A3*np.conj(A1), K3*A1*A2)

def dA2dz1(z, S): #delta_k != 0
    A1, A2, A3 = S
    return (K1*A3*np.conj(A2)*np.exp(1j*delta_k1*z),K2*A3*np.conj(A1)*np.exp(1j*delta_k
            K3*A1*A2*np.exp(-1j*delta_k1*z))

def dA2dz2(z, S): #delta_k != 0
    A1, A2, A3 = S
    return (K1*A3*np.conj(A2)*np.exp(1j*delta_k2*z),K2*A3*np.conj(A1)*np.exp(1j*delta_k
            K3*A1*A2*np.exp(-1j*delta_k2*z))

def dA2dz3(z, S): #delta_k != 0
    A1, A2, A3 = S
    return (K1*A3*np.conj(A2)*np.exp(1j*delta_k3*z),K2*A3*np.conj(A1)*np.exp(1j*delta_k
            K3*A1*A2*np.exp(-1j*delta_k3*z))

def dA2dz4(z, S): #delta_k != 0
    A1, A2, A3 = S
    return (K1*A3*np.conj(A2)*np.exp(1j*delta_k4*z),K2*A3*np.conj(A1)*np.exp(1j*delta_k
            K3*A1*A2*np.exp(-1j*delta_k4*z))

def dA2dz5(z, S): #delta_k != 0
    A1, A2, A3 = S
    return (K1*A3*np.conj(A2)*np.exp(1j*delta_k5*z),K2*A3*np.conj(A1)*np.exp(1j*delta_k
            K3*A1*A2*np.exp(-1j*delta_k5*z))
```

```
z_max = 2100

delta = 1
A3 = 10+0.0j
A1 = 0.1+0.0j

y02 = [A1,0,A3]

sol1 = solve_ivp(dA2dz,np.array([0,z_max]),y02, t_eval = np.linspace(0,z_max,1000))
sol2 = solve_ivp(dA2dz1,np.array([0,z_max]),y02, t_eval = np.linspace(0,z_max,1000))
sol3 = solve_ivp(dA2dz2,np.array([0,z_max]),y02, t_eval = np.linspace(0,z_max,1000))
sol4 = solve_ivp(dA2dz3,np.array([0,z_max]),y02, t_eval = np.linspace(0,z_max,1000))
sol5 = solve_ivp(dA2dz4,np.array([0,z_max]),y02, t_eval = np.linspace(0,z_max,1000))
sol6 = solve_ivp(dA2dz5,np.array([0,z_max]),y02, t_eval = np.linspace(0,z_max,1000))


z_OPA = sol1.t
A1_OPA = 4*c*8.9**-12*np.abs(sol1.y[0]**2)
A2_OPA = 4*c*8.9**-12*np.abs(sol1.y[1]**2)
A3_OPA = 4*c*8.9**-12*np.abs(sol1.y[2]**2)
z2_OPA = sol2.t
A1_OPA2 = 4*c*8.9**-12*np.abs(sol2.y[0]**2)
A2_OPA2 = 4*c*8.9**-12*np.abs(sol2.y[1]**2)
A3_OPA2 = 4*c*8.9**-12*np.abs(sol2.y[2]**2)
z3_OPA = sol3.t
A1_OPA3 = 4*c*8.9**-12*np.abs(sol3.y[0]**2)
A2_OPA3 = 4*c*8.9**-12*np.abs(sol3.y[1]**2)
A3_OPA3 = 4*c*8.9**-12*np.abs(sol3.y[2]**2)
z4_OPA = sol4.t
A1_OPA4 = 4*c*8.9**-12*np.abs(sol4.y[0]**2)
A2_OPA4 = 4*c*8.9**-12*np.abs(sol4.y[1]**2)
A3_OPA4 = 4*c*8.9**-12*np.abs(sol4.y[2]**2)
z5_OPA = sol5.t
A1_OPA5 = 4*c*8.9**-12*np.abs(sol5.y[0]**2)
A2_OPA5 = 4*c*8.9**-12*np.abs(sol5.y[1]**2)
A3_OPA5 = 4*c*8.9**-12*np.abs(sol5.y[2]**2)
z6_OPA = sol6.t
A1_OPA6 = 4*c*8.9**-12*np.abs(sol6.y[0]**2)
A2_OPA6 = 4*c*8.9**-12*np.abs(sol6.y[1]**2)
A3_OPA6 = 4*c*8.9**-12*np.abs(sol6.y[2]**2)

A_sum1 = np.add(A1_OPA,A2_OPA)
A_sum2 = np.add(A1_OPA2,A2_OPA2)
A_sum3 = np.add(A1_OPA3,A2_OPA3)
A_sum4 = np.add(A1_OPA4,A2_OPA4)
A_sum5 = np.add(A1_OPA5,A2_OPA5)
A_sum6 = np.add(A1_OPA6,A2_OPA6)
```

In [49]:

```
plt.plot(z_OPA,A_sum1)
plt.plot(z3_OPA,A_sum3)
plt.plot(z4_OPA,A_sum4)
plt.plot(z2_OPA,A_sum2)
plt.plot(z5_OPA,A_sum5)
plt.plot(z6_OPA,A_sum6)

plt.xlabel("z")
plt.ylabel("Sum Intensity")
plt.yscale('log')
plt.legend(["0","0.75","0.95","1","1.05","1.25"])
plt.title("(I_s+I_i)")
plt.show()
```



Reference: Jeffrey Moses and Shu-Wei Huang, "Conformal profile theory for performance scaling of ultrabroadband optical parametric chirped pulse amplification," J. Opt. Soc. Am. B 28, 812-831 (2011)