visualization_multi_a

March 12, 2023

1 my range [20.5,20.65,10]

[28]:
```
# This file aims to draw conclusion from DN.npz including multi A
import numpy as np
import matplotlib.pyplot as plt
from astropy.modeling import models, fitting
import math
from astropy.constants import au, R_sun
from calculating_DN_2048 import wavelength_point_num, wavelength_list,␣
 ↪angle_point_num_alpha
from calculating_DN_2048 import offaxis_angle_x_alpha, offaxis_angle_y_alpha

transunit = ((au/R_sun).value)**2 / 1000
DN = np.load("output_DN/DN_my_range_2048.npz")

DN_alpha_list = DN["DN_alpha_list"]  # DN_alpha_list.shape= (10, 61, 15)
a_list = DN['a_list']
a_num = len(a_list)

# Cruciformscan in alpha direction
offaxis_angle_x_min_alpha=offaxis_angle_x_alpha*180*60/math.pi

# Fit data in DN ??.npz
wavelength_shift_alpha = np.zeros((a_num, angle_point_num_alpha))
fit_alpha = []   # List of Gaussian1D
for i in range(a_num):
    for j in range(angle_point_num_alpha):
        g_init = models.Gaussian1D(amplitude=1E9, mean=0.05, stddev=0.0424)
        # initial value for fitting
        fit_g = fitting.LevMarLSQFitter()
        g = fit_g(g_init, wavelength_list, DN_alpha_list[i][j])
        wavelength_shift_alpha[i][j] = g.mean.value
        fit_alpha.append(g)
```

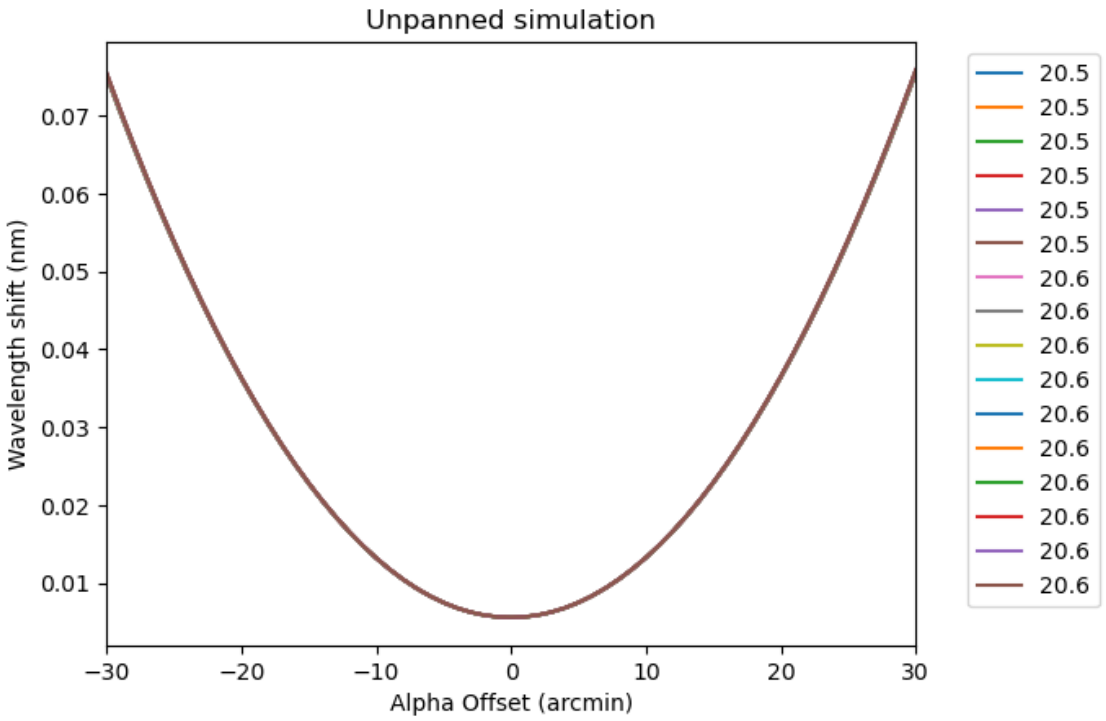
[29]:
```
a_list/transunit
```

[29]:
```
array([20.5 , 20.51, 20.52, 20.53, 20.54, 20.55, 20.56, 20.57, 20.58,
       20.59, 20.6 , 20.61, 20.62, 20.63, 20.64, 20.65])
```

1.1 Multi A

[30]:
```
fig, ax = plt.subplots()
for i in range(a_num):
    # ax.plot(offaxis_angle_x_alpha,
    #         wavelength_shift_alpha[i]\
    #             -wavelength_shift_alpha[i][int(angle_point_num_alpha/2)])
    ax.plot(offaxis_angle_x_min_alpha, wavelength_shift_alpha[i],
            label="{:.1f}".format(a_list[i]/transunit)  )

ax.set_xlim(-30,30)
ax.set_xlabel("Alpha Offset (arcmin)")
ax.set_ylabel('Wavelength shift (nm)')
ax.set_title("Unpanned simulation")
ax.legend(loc='upper left', bbox_to_anchor=(1.05, 1))
```

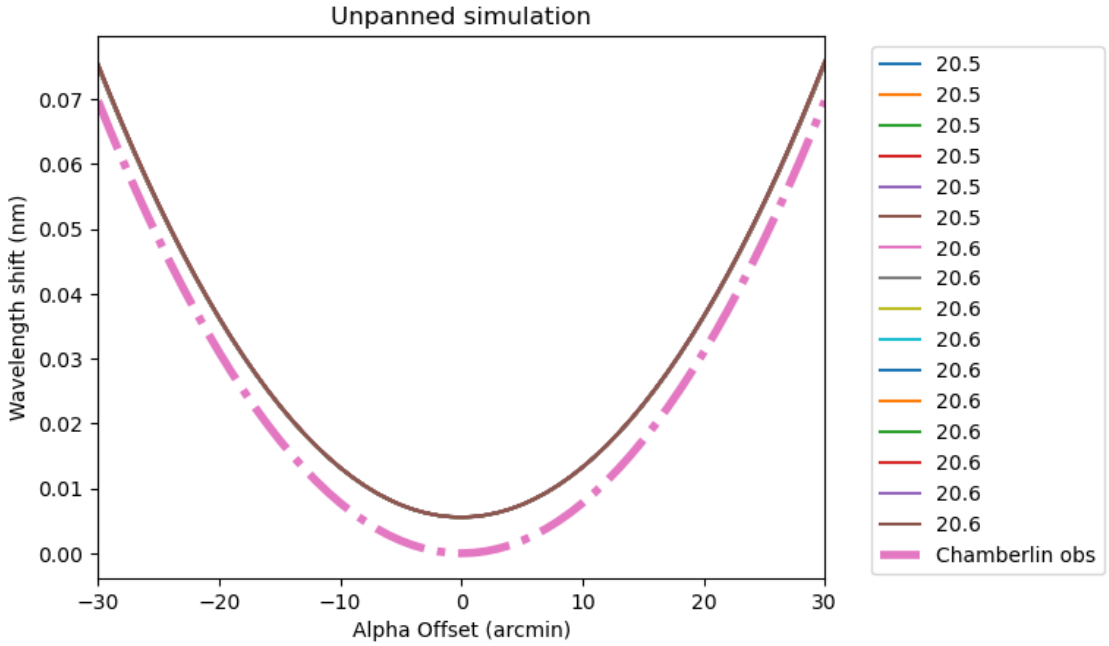
[30]:
```
<matplotlib.legend.Legend at 0x17bb517f280>
```



[31]:
```
fig, ax = plt.subplots()
for i in range(a_num):
```

```
    # ax.plot(offaxis_angle_x_alpha,
    #         wavelength_shift_alpha[i]\
    #             -wavelength_shift_alpha[i][int(angle_point_num_alpha/2)])
    ax.plot(offaxis_angle_x_min_alpha, wavelength_shift_alpha[i],
            label="{:.1f}".format(a_list[i]/transunit))

ax.set_xlim(-30, 30)
ax.set_xlabel("Alpha Offset (arcmin)")
ax.set_ylabel('Wavelength shift (nm)')
ax.plot(offaxis_angle_x_min_alpha,19.
 ↪8*transunit*offaxis_angle_x_alpha**2,label="Chamberlin obs",
       linestyle='dashdot',linewidth=4)
ax.set_title("Unpanned simulation")
ax.legend(loc='upper left', bbox_to_anchor=(1.05, 1))
```

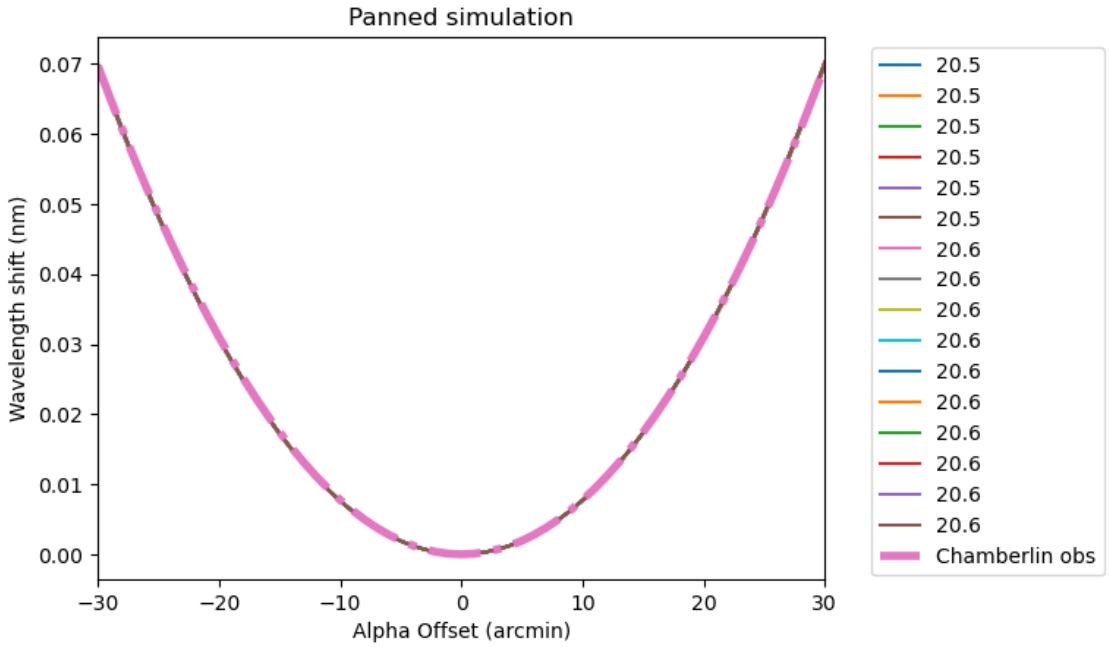
[31]: <matplotlib.legend.Legend at 0x17bb5316f10>



[32]:
```
fig, ax = plt.subplots()
for i in range(a_num):
    ax.plot(offaxis_angle_x_min_alpha,
            wavelength_shift_alpha[i]\
                -wavelength_shift_alpha[i][int(angle_point_num_alpha/2)],
            label="{:.1f}".format(a_list[i]/transunit))
    # ax.plot(offaxis_angle_x_min_alpha, wavelength_shift_alpha[i],
    #         label="{:.2f}".format(a_list[i]/transunit))
```

```
ax.set_xlim(-30, 30)
ax.plot(offaxis_angle_x_min_alpha,19.
 ↪8*transunit*offaxis_angle_x_alpha**2,label="Chamberlin obs",
        linestyle='dashdot',linewidth=4)
ax.set_xlabel("Alpha Offset (arcmin)")
ax.set_ylabel('Wavelength shift (nm)')
ax.set_title("Panned simulation")
ax.legend(loc='upper left', bbox_to_anchor=(1.05, 1))
```

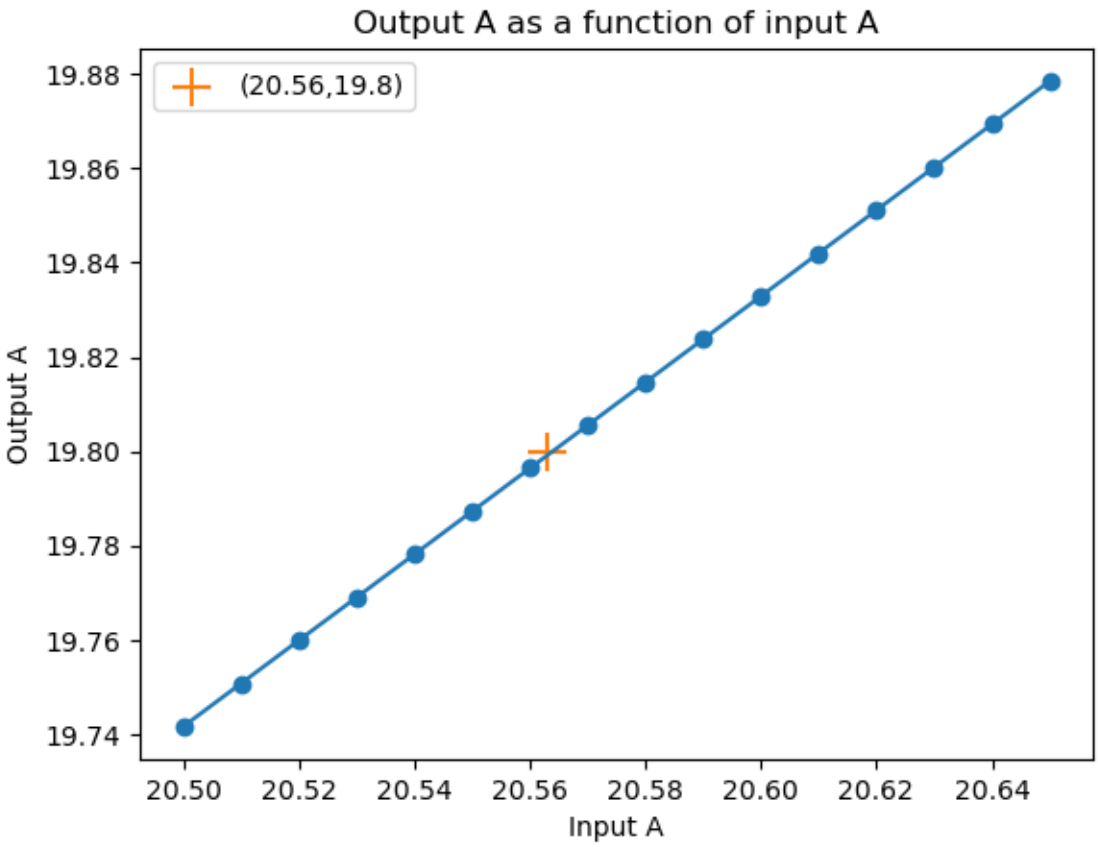
[32]: <matplotlib.legend.Legend at 0x17bb52e5f70>



[33]:
```
# %% Fit the output A

# Define the polynomial model
model = models.Polynomial1D(degree=2)

# Define the fitter
fitter = fitting.LevMarLSQFitter()

fitted_model = []
for i in range(a_num):
    # Fit the model to the data
    fitted_model.append(
        fitter(model, offaxis_angle_x_alpha, wavelength_shift_alpha[i]))
```

[34]:
```
c2_list = []
for i in range(a_num):
    c2_list.append(fitted_model[i].c2.value)
fig, ax = plt.subplots()
ax.plot(a_list/transunit, c2_list/transunit)
ax.scatter(a_list/transunit, c2_list/transunit)
ax.scatter(20.563,19.8,marker='+',s=200,label="(20.56,19.8)")
ax.set_xlabel("Input A")
ax.set_ylabel("Output A")
ax.set_title("Output A as a function of input A")
ax.legend()
```

[34]:
```
<matplotlib.legend.Legend at 0x17bb549b910>
```



1.1.1 A=20.72

2 10a (18.4 18.9 10)

[35]:
```
# This file aims to draw conclusion from DN.npz including multi A
import numpy as np
import matplotlib.pyplot as plt
from astropy.modeling import models, fitting
import math
from astropy.constants import au, R_sun

transunit = ((au/R_sun).value)**2 / 1000
DN = np.load("output_DN/DN_10a_2048.npz")
DN_alpha_list = DN["DN_alpha_list"]  # DN_alpha_list.shape= (10, 61, 15)
a_list = DN['a_list']
a_num = len(a_list)

# Initialize
wavelength_point_num = 15
wavelength_list = np.linspace(-0.1, 0.15, wavelength_point_num)

# Cruciformscan in alpha direction
angle_point_num_alpha = 61
DN_alpha = np.zeros((angle_point_num_alpha, wavelength_point_num))
offaxis_angle_x_alpha = np.linspace(-math.pi /
                                    360, math.pi/360, angle_point_num_alpha)
offaxis_angle_x_min_alpha=offaxis_angle_x_alpha*180*60/math.pi
offaxis_angle_y_alpha = np.zeros(angle_point_num_alpha)

# Cruciformscan in beta direction
angle_point_num_beta = 61
DN_beta = np.zeros((angle_point_num_beta, wavelength_point_num))
offaxis_angle_x_beta = np.zeros(angle_point_num_beta)
offaxis_angle_y_beta = np.linspace(-math.pi /
                                   360, math.pi/360, angle_point_num_beta)
offaxis_angle_y_min_beta=offaxis_angle_y_beta*180*60/math.pi

# Fit data in DN ??.npz
wavelength_shift_alpha = np.zeros((a_num, angle_point_num_alpha))
fit_alpha = []   # List of Gaussian1D
for i in range(a_num):
    for j in range(angle_point_num_alpha):
        g_init = models.Gaussian1D(amplitude=1E9, mean=0.05, stddev=0.0424)
        # initial value for fitting
        fit_g = fitting.LevMarLSQFitter()
        g = fit_g(g_init, wavelength_list, DN_alpha_list[i][j])
        wavelength_shift_alpha[i][j] = g.mean.value
```

```
        fit_alpha.append(g)
```

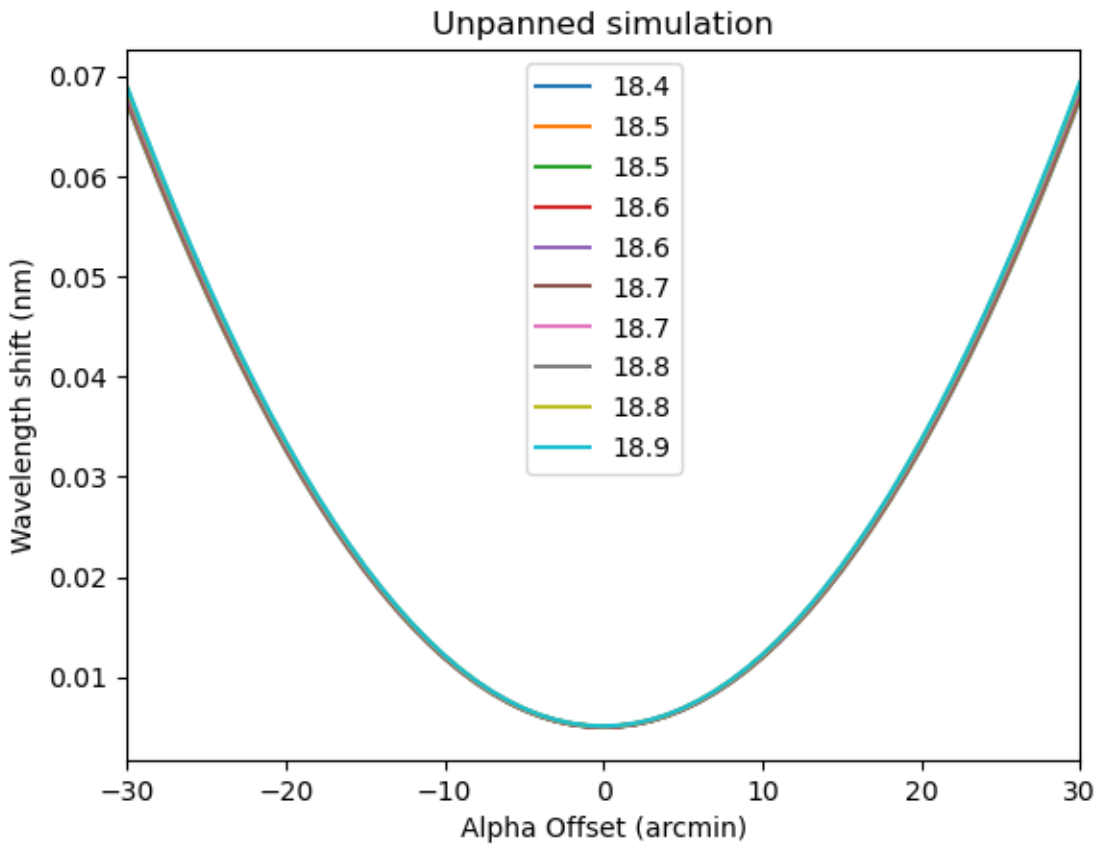
[36]:
```
a_list/transunit
```

[36]:
```
array([18.4       , 18.45555556, 18.51111111, 18.56666667, 18.62222222,
       18.67777778, 18.73333333, 18.78888889, 18.84444444, 18.9       ])
```

2.1 Multi A

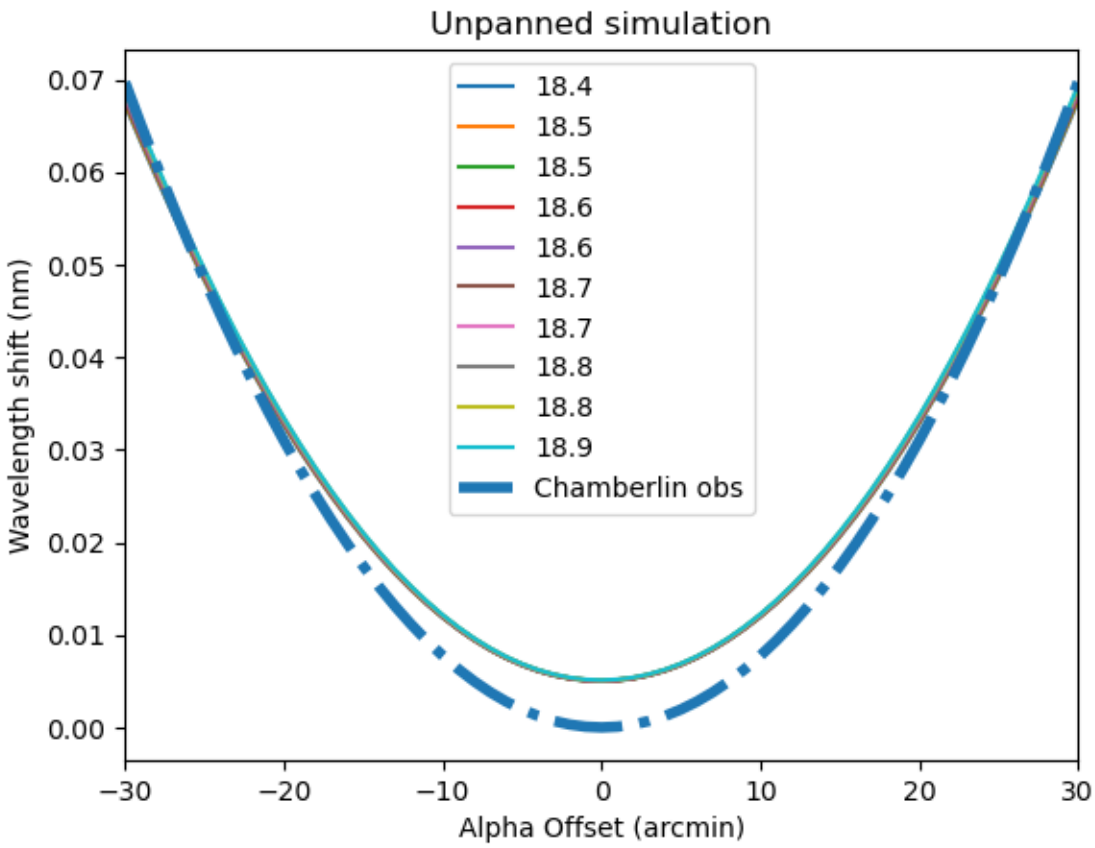
[37]:
```
fig, ax = plt.subplots()
for i in range(a_num):
    # ax.plot(offaxis_angle_x_alpha,
    #         wavelength_shift_alpha[i]\
    #             -wavelength_shift_alpha[i][int(angle_point_num_alpha/2)])
    ax.plot(offaxis_angle_x_min_alpha, wavelength_shift_alpha[i],
            label="{:.1f}".format(a_list[i]/transunit)  )

ax.set_xlim(-30,30)
ax.set_xlabel("Alpha Offset (arcmin)")
ax.set_ylabel('Wavelength shift (nm)')
ax.set_title("Unpanned simulation")
ax.legend()
```

[37]:
```
<matplotlib.legend.Legend at 0x17bbacf31c0>
```



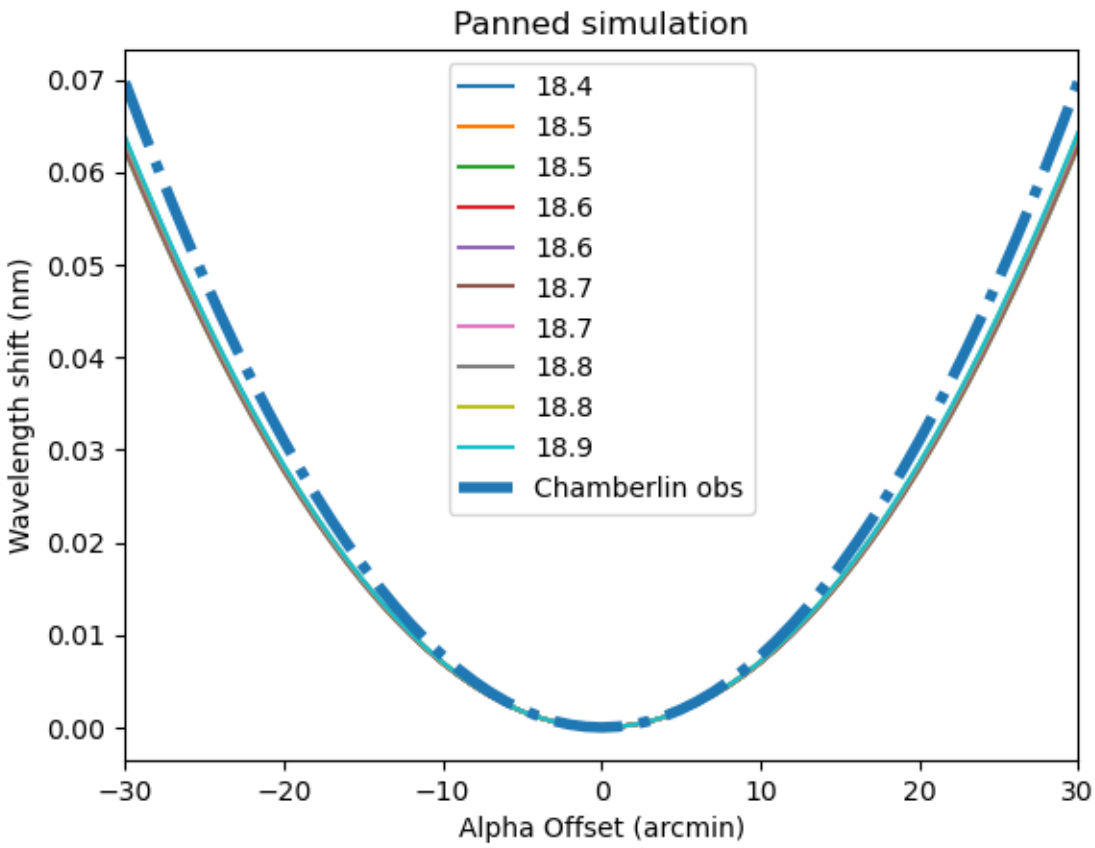
[38]:
```
fig, ax = plt.subplots()
for i in range(a_num):
    # ax.plot(offaxis_angle_x_alpha,
    #         wavelength_shift_alpha[i]\
    #             -wavelength_shift_alpha[i][int(angle_point_num_alpha/2)])
    ax.plot(offaxis_angle_x_min_alpha, wavelength_shift_alpha[i],
            label="{:.1f}".format(a_list[i]/transunit))

ax.set_xlim(-30, 30)
ax.set_xlabel("Alpha Offset (arcmin)")
ax.set_ylabel('Wavelength shift (nm)')
ax.plot(offaxis_angle_x_min_alpha,19.
 ↪8*transunit*offaxis_angle_x_alpha**2,label="Chamberlin obs",
       linestyle='dashdot',linewidth=4)
ax.set_title("Unpanned simulation")
ax.legend()
```

[38]: <matplotlib.legend.Legend at 0x17bbad8bf40>



[39]:
```
fig, ax = plt.subplots()
for i in range(a_num):
    ax.plot(offaxis_angle_x_min_alpha,
            wavelength_shift_alpha[i]\
                -wavelength_shift_alpha[i][int(angle_point_num_alpha/2)],
            label="{:.1f}".format(a_list[i]/transunit))
    # ax.plot(offaxis_angle_x_min_alpha, wavelength_shift_alpha[i],
    #         label="{:.2f}".format(a_list[i]/transunit))

ax.set_xlim(-30, 30)
ax.plot(offaxis_angle_x_min_alpha,19.
 ↪8*transunit*offaxis_angle_x_alpha**2,label="Chamberlin obs",
        linestyle='dashdot',linewidth=4)
ax.set_xlabel("Alpha Offset (arcmin)")
ax.set_ylabel('Wavelength shift (nm)')
ax.set_title("Panned simulation")
ax.legend()
```

[39]:
```
<matplotlib.legend.Legend at 0x17bbb32df70>
```



2.1.1 18.66

3 A: [0,50,26]

[40]:
```
# This file aims to draw conclusion from DN.npz including multi A
import numpy as np
import matplotlib.pyplot as plt
from astropy.modeling import models, fitting
import math
from astropy.constants import au, R_sun
from calculating_DN_2048 import wavelength_point_num, wavelength_list,␣
 ↪angle_point_num_alpha
from calculating_DN_2048 import offaxis_angle_x_alpha, offaxis_angle_y_alpha

transunit = ((au/R_sun).value)**2 / 1000
DN = np.load("output_DN/DN_dense_lambda_0to50_2048.npz")
DN_alpha_list = DN["DN_alpha_list"]  # DN_alpha_list.shape= (10, 61, 15)
a_list = DN['a_list']
a_num = len(a_list)
```

```
# Cruciformscan in alpha direction
offaxis_angle_x_min_alpha=offaxis_angle_x_alpha*180*60/math.pi

# Fit data in DN ??.npz
wavelength_shift_alpha = np.zeros((a_num, angle_point_num_alpha))
fit_alpha = []   # List of Gaussian1D
for i in range(a_num):
    for j in range(angle_point_num_alpha):
        g_init = models.Gaussian1D(amplitude=1E9, mean=0.05, stddev=0.0424)
        # initial value for fitting
        fit_g = fitting.LevMarLSQFitter()
        g = fit_g(g_init, wavelength_list, DN_alpha_list[i][j])
        wavelength_shift_alpha[i][j] = g.mean.value
        fit_alpha.append(g)
```

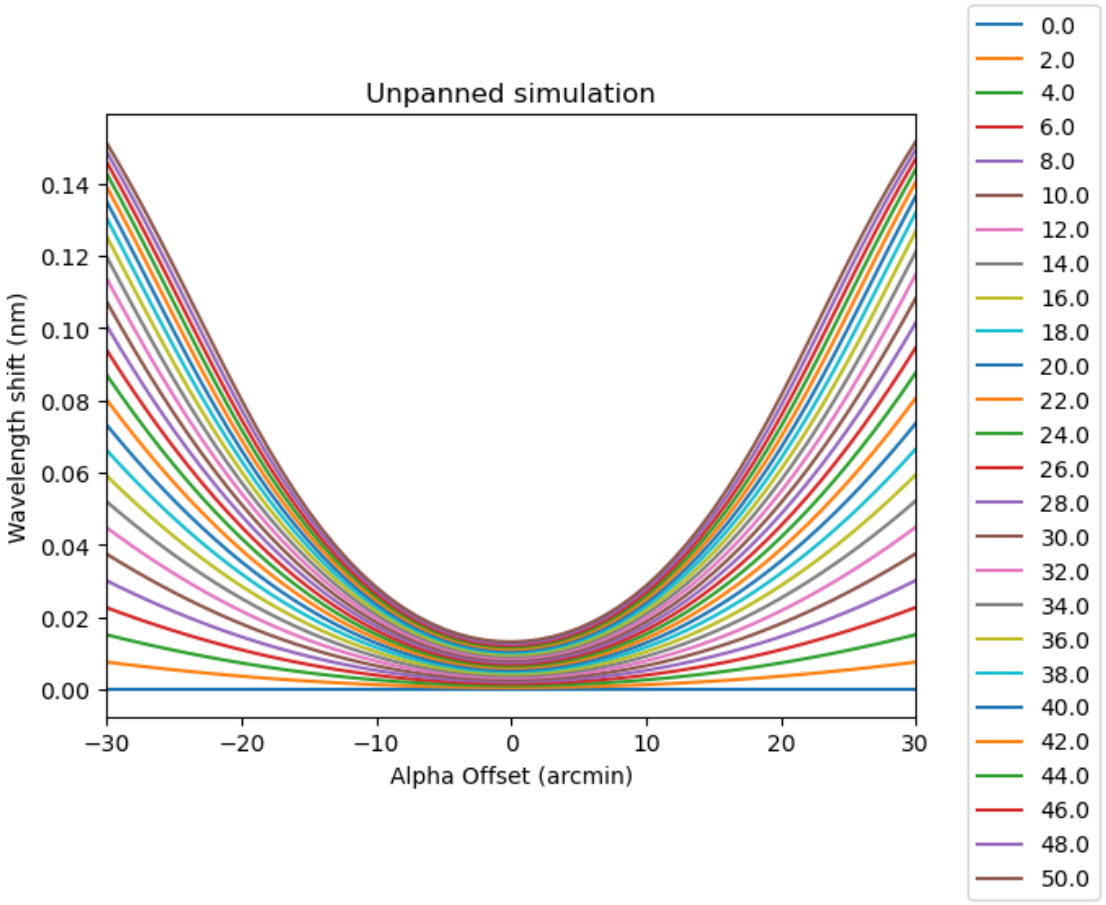
[41]:
```
a_list/transunit
```

[41]:
```
array([ 0.,  2.,  4.,  6.,  8., 10., 12., 14., 16., 18., 20., 22., 24.,
       26., 28., 30., 32., 34., 36., 38., 40., 42., 44., 46., 48., 50.])
```

3.1 Multi A

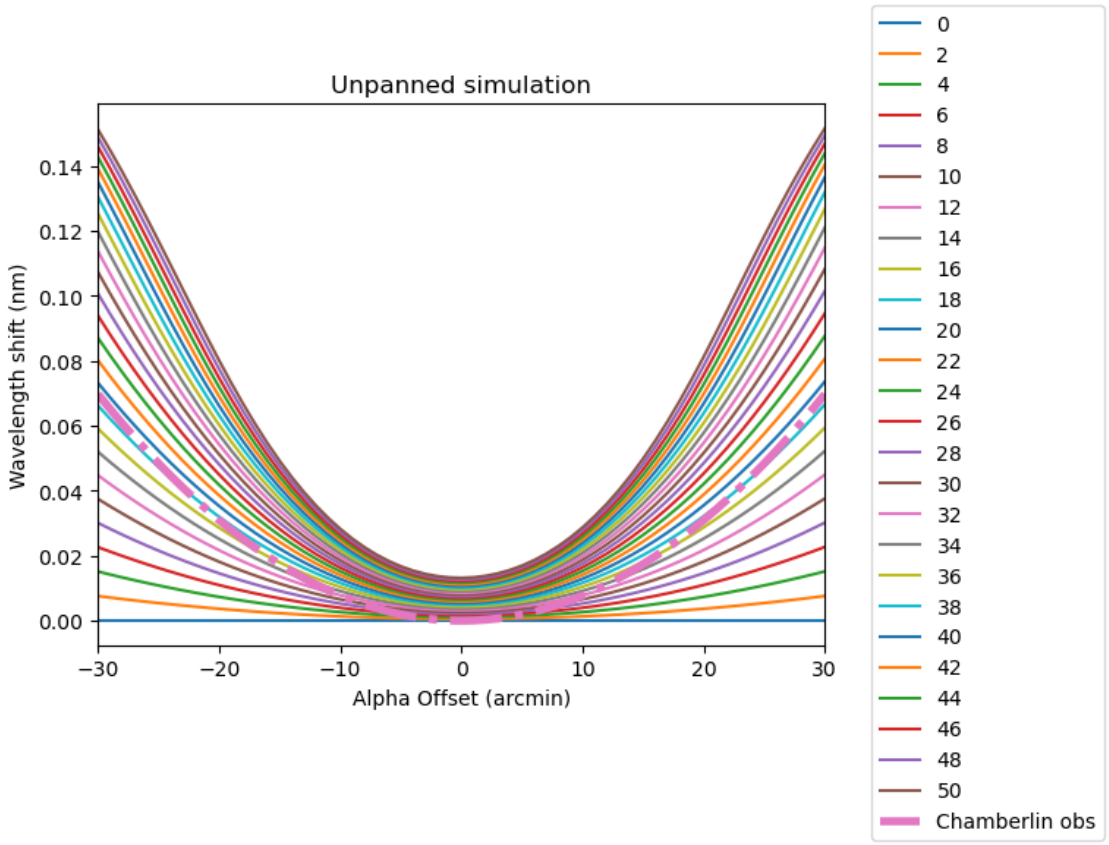
[42]:
```
fig, ax = plt.subplots()
for i in range(a_num):
    # ax.plot(offaxis_angle_x_alpha,
    #         wavelength_shift_alpha[i]\
    #             -wavelength_shift_alpha[i][int(angle_point_num_alpha/2)])
    ax.plot(offaxis_angle_x_min_alpha, wavelength_shift_alpha[i],
            label="{:.1f}".format(a_list[i]/transunit)  )

ax.set_xlim(-30,30)
ax.set_xlabel("Alpha Offset (arcmin)")
ax.set_ylabel('Wavelength shift (nm)')
ax.set_title("Unpanned simulation")
ax.annotate('Label', xy=(3, 6), xytext=(4, 8),
            arrowprops=dict(arrowstyle='->', lw=2))
ax.legend(loc='upper left', bbox_to_anchor=(1.05, 1.2))
```

[42]:
```
<matplotlib.legend.Legend at 0x17bbb577760>
```



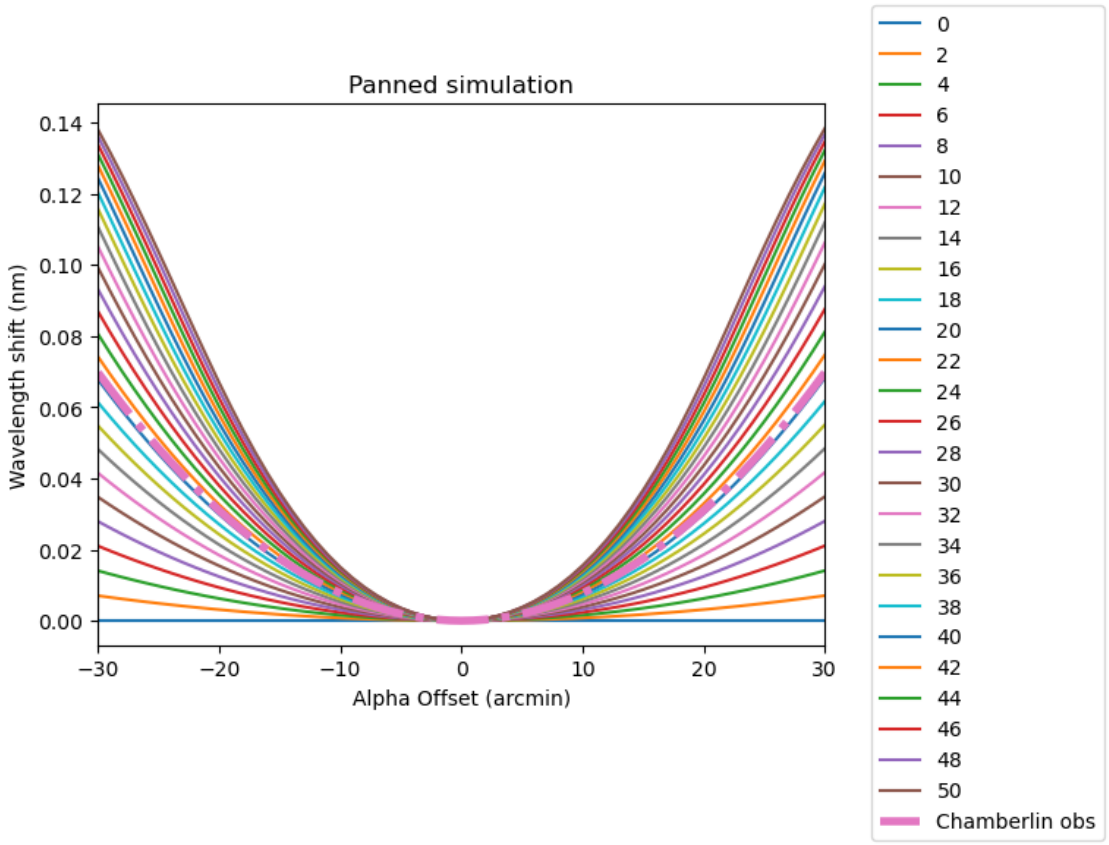
[43]:
```
fig, ax = plt.subplots()
for i in range(a_num):
    # ax.plot(offaxis_angle_x_alpha,
    #         wavelength_shift_alpha[i]\
    #             -wavelength_shift_alpha[i][int(angle_point_num_alpha/2)])
    ax.plot(offaxis_angle_x_min_alpha, wavelength_shift_alpha[i],
            label="{:.0f}".format(a_list[i]/transunit))

ax.set_xlim(-30, 30)
ax.set_xlabel("Alpha Offset (arcmin)")
ax.set_ylabel('Wavelength shift (nm)')
ax.plot(offaxis_angle_x_min_alpha,19.
 ↪8*transunit*offaxis_angle_x_alpha**2,label="Chamberlin obs",
        linestyle='dashdot',linewidth=4)
ax.set_title("Unpanned simulation")
ax.legend(loc='upper left', bbox_to_anchor=(1.05, 1.2))
```

[43]:
```
<matplotlib.legend.Legend at 0x17bb53ed6a0>
```



[44]:
```
fig, ax = plt.subplots()
for i in range(a_num):
    ax.plot(offaxis_angle_x_min_alpha,
            wavelength_shift_alpha[i]\
                -wavelength_shift_alpha[i][int(angle_point_num_alpha/2)],
            label="{:.0f}".format(a_list[i]/transunit))
    # ax.plot(offaxis_angle_x_min_alpha, wavelength_shift_alpha[i],
    #         label="{:.2f}".format(a_list[i]/transunit))

ax.set_xlim(-30, 30)
ax.plot(offaxis_angle_x_min_alpha,19.
 ↪8*transunit*offaxis_angle_x_alpha**2,label="Chamberlin obs",
       linestyle='dashdot',linewidth=4)
ax.set_xlabel("Alpha Offset (arcmin)")
ax.set_ylabel('Wavelength shift (nm)')
ax.set_title("Panned simulation")
ax.legend(loc='upper left', bbox_to_anchor=(1.05, 1.2))
```

[44]:
```
<matplotlib.legend.Legend at 0x17bba187f10>
```

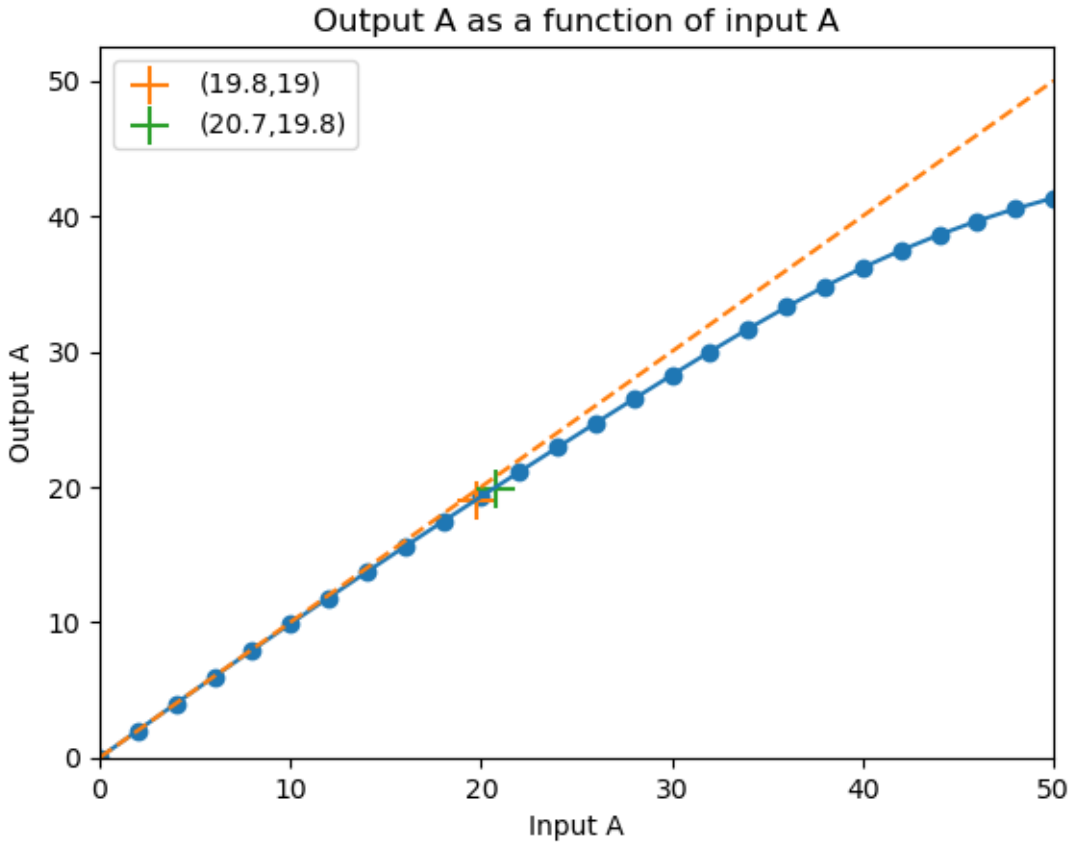


[45]:
```
# %% Fit the output A

# Define the polynomial model
model = models.Polynomial1D(degree=2)

# Define the fitter
fitter = fitting.LevMarLSQFitter()

fitted_model = []
for i in range(a_num):
    # Fit the model to the data
    fitted_model.append(
        fitter(model, offaxis_angle_x_alpha, wavelength_shift_alpha[i]))
```

[46]:
```
c2_list = []
for i in range(a_num):
    c2_list.append(fitted_model[i].c2.value)
fig, ax = plt.subplots()
ax.plot(a_list/transunit, c2_list/transunit)
ax.scatter(a_list/transunit, c2_list/transunit)
```

```
ax.scatter(19.8,19,marker='+',s=200,label="(19.8,19)")
ax.scatter(20.72,19.8,marker='+',s=200,label="(20.7,19.8)")
ax.plot(a_list/transunit,a_list/transunit,linestyle="dashed")
# ax.scatter(a_list/transunit,a_list/transunit)
ax.set_xlim(0,50)
ax.set_ylim(0,)
ax.set_xlabel("Input A")
ax.set_ylabel("Output A")
# ax.annotate('(19.8,19)', xy=(19.8,19), xytext=(2, 40),
            # arrowprops=dict(facecolor='black', shrink=0.05))
ax.set_title("Output A as a function of input A")

ax.legend()
```

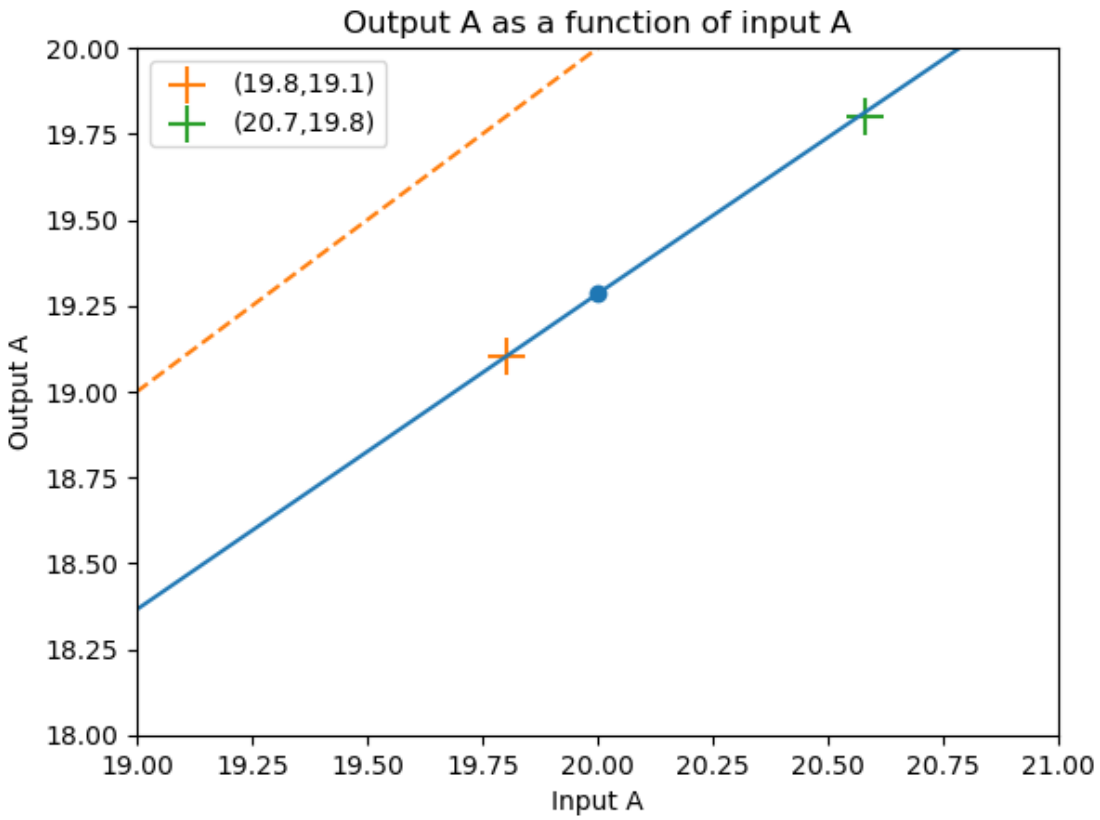
[46]: <matplotlib.legend.Legend at 0x17bba29bfd0>



[47]:
```
c2_list = []
for i in range(a_num):
    c2_list.append(fitted_model[i].c2.value)
fig, ax = plt.subplots()
```

```
ax.plot(a_list[8:12]/transunit, c2_list[8:12]/transunit)
ax.scatter(a_list[8:12]/transunit, c2_list[8:12]/transunit)
ax.scatter(19.8,19.1,marker='+',s=200,label="(19.8,19.1)")
ax.scatter(20.58,19.8,marker='+',s=200,label="(20.7,19.8)")
ax.plot(a_list/transunit,a_list/transunit,linestyle="dashed")
# ax.scatter(a_list/transunit,a_list/transunit)
ax.set_xlim(19,21)
# ax.set_ylim(0,)
ax.set_ylim(18,20)
ax.set_xlabel("Input A")
ax.set_ylabel("Output A")
# ax.annotate('(19.8,19)', xy=(19.8,19), xytext=(2, 40),
#             arrowprops=dict(facecolor='black', shrink=0.05))
ax.set_title("Output A as a function of input A")

ax.legend()
```

[47]: <matplotlib.legend.Legend at 0x17bbabfbdf0>



[ ]: