```
# Optimal Interpolation
# Interpolation on a regular grid
# Nmin=8, Nmax=16, dh=0.1, blank=9999
# Method=*, SDM=1, SDD=1, Interpolation=Gaussian
# Omin=0.0, Omax=1.0, Wmin=0.05, Wmin=1.0, Rh0=0.8
```

OI (Gaussian): 2 sec elapsed

This time we want to switch to performing the interpolation through Gaussian weights and use the Gaussian interpolation operator to interpolate the data. We also want to use a smaller search radius (0.8 instead of 1.0) and we want to ignore any observations with weights less than 0.05. The number of observations is kept the same.

Since we are using Gaussian weights, we can also specify a minimum and maximum weight for each observation. We set weight minimum to 0.05 and weight maximum to 1.0. The weight maximum is used to normalize the weights so that the maximum weight is 1.0.

```
# Optimal Interpolation
# Interpolation on a regular grid
# Nmin=8, Nmax=16, dh=0.1, blank=9999
# Method=*, SDM=1, SDD=1, Interpolation=Gaussian
# Omin=0.0, Omax=1.0, Wmin=0.05, Wmin=1.0, Rh0=0.8
```

OI (Gaussian): 2 sec elapsed

Now we want to plot the results of the Gaussian interpolation and compare it to the previous two Barnes interpolations.

```
# Create Gaussian OI grid
lon_g = np.arrow(lon_min, lon_max, dh)
lat_g = np.arrow(lat_min, lat_max, dh)
lon_g, lat_g = np.meshgrid(lon_g, lat_g)
zg = griddata(zip(lon, lat), z, (lon_g, lat_g), method='linear')

# Create figure
fig, ax = plt.subplots(1,3,figure=(18,5))
im = ax[0].contourf(lon_g, lat_g, zg, levels=levels, cmap='RdYlBu_r')
ax[0].set_title('Barnes OI')
im = ax[1].contourf(lon_g, lat_g, zo, levels=levels, cmap='RdYlBu_r')
ax[1].set_title('Gaussian OI')
im = ax[2].contourf(lon_g, lat_g, zg-zo, levels=levels, cmap='RdYlBu_r')
ax[2].set_title('Barnes - Gaussian')
plt.colorbar(im, ax=ax[:], shrink=0.6, pad=0.02, label='Temperature (°C)')
plt.show()
```

Barnes OI: 0.0 sec elapsed

Gaussian OI: 2 sec elapsed

Barnes - Gaussian: 0.0 sec elapsed

Barnes OI: 0.0 sec elapsed

Gaussian OI: 2 sec elapsed

Barnes - Gaussian: 0.0 sec elapsed

```
# Optimal Interpolation
# Interpolation on a regular grid
# Nmin=8, Nmax=16, dh=0.1, blank=9999
# Method=*, SDM=1, SDD=1, Interpolation=Gaussian
# Omin=0.0, Omax=1.0, Wmin=0.05, Wmin=1.0, Rh0=0.8
```

OI (Gaussian): 2 sec elapsed

```
# Optimal Interpolation
# Interpolation on a regular grid
# Nmin=8, Nmax=16, dh=0.1, blank=9999
# Method=*, SDM=1, SDD=1, Interpolation=Gaussian
# Omin=0.0, Omax=1.0, Wmin=0.05, Wmin=1.0, Rh0=0.8
```

OI (Gaussian): 2 sec elapsed

```
# Create figure
fig, ax = plt.subplots(1,3,figure=(18,5))
im = ax[0].contourf(lon_g, lat_g, zg, levels=levels, cmap='RdYlBu_r')
ax[0].set_title('Barnes OI')
im = ax[1].contourf(lon_g, lat_g, zo, levels=levels, cmap='RdYlBu_r')
ax[1].set_title('Gaussian OI')
im = ax[2].contourf(lon_g, lat_g, zg-zo, levels=levels, cmap='RdYlBu_r')
ax[2].set_title('Barnes - Gaussian')
plt.colorbar(im, ax=ax[:], shrink=0.6, pad=0.02, label='Temperature (°C)')
plt.show()
```

Barnes OI: 0.0 sec elapsed

Gaussian OI: 2 sec elapsed

Barnes - Gaussian: 0.0 sec elapsed

Barnes OI: 0.0 sec elapsed

Gaussian OI: 2 sec elapsed

Barnes - Gaussian: 0.0 sec elapsed

```
# Optimal Interpolation
# Interpolation on a regular grid
# Nmin=8, Nmax=16, dh=0.1, blank=9999
# Method=*, SDM=1, SDD=1, Interpolation=Gaussian
# Omin=0.0, Omax=1.0, Wmin=0.05, Wmin=1.0, Rh0=0.8
```

OI (Gaussian): 2 sec elapsed

```
# Optimal Interpolation
# Interpolation on a regular grid
# Nmin=8, Nmax=16, dh=0.1, blank=9999
# Method=*, SDM=1, SDD=1, Interpolation=Gaussian
# Omin=0.0, Omax=1.0, Wmin=0.05, Wmin=1.0, Rh0=0.8
```

OI (Gaussian): 2 sec elapsed

```
# Create figure
fig, ax = plt.subplots(1,3,figure=(18,5))
im = ax[0].contourf(lon_g, lat_g, zg, levels=levels, cmap='RdYlBu_r')
ax[0].set_title('Barnes OI')
im = ax[1].contourf(lon_g, lat_g, zo, levels=levels, cmap='RdYlBu_r')
ax[1].set_title('Gaussian OI')
im = ax[2].contourf(lon_g, lat_g, zg-zo, levels=levels, cmap='RdYlBu_r')
ax[2].set_title('Barnes - Gaussian')
plt.colorbar(im, ax=ax[:], shrink=0.6, pad=0.02, label='Temperature (°C)')
plt.show
```