In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
```

# Power of Signal Averaging

Recall the Spectral Line problem and its dataset. We had a model with a predicted signal $S$ given by:

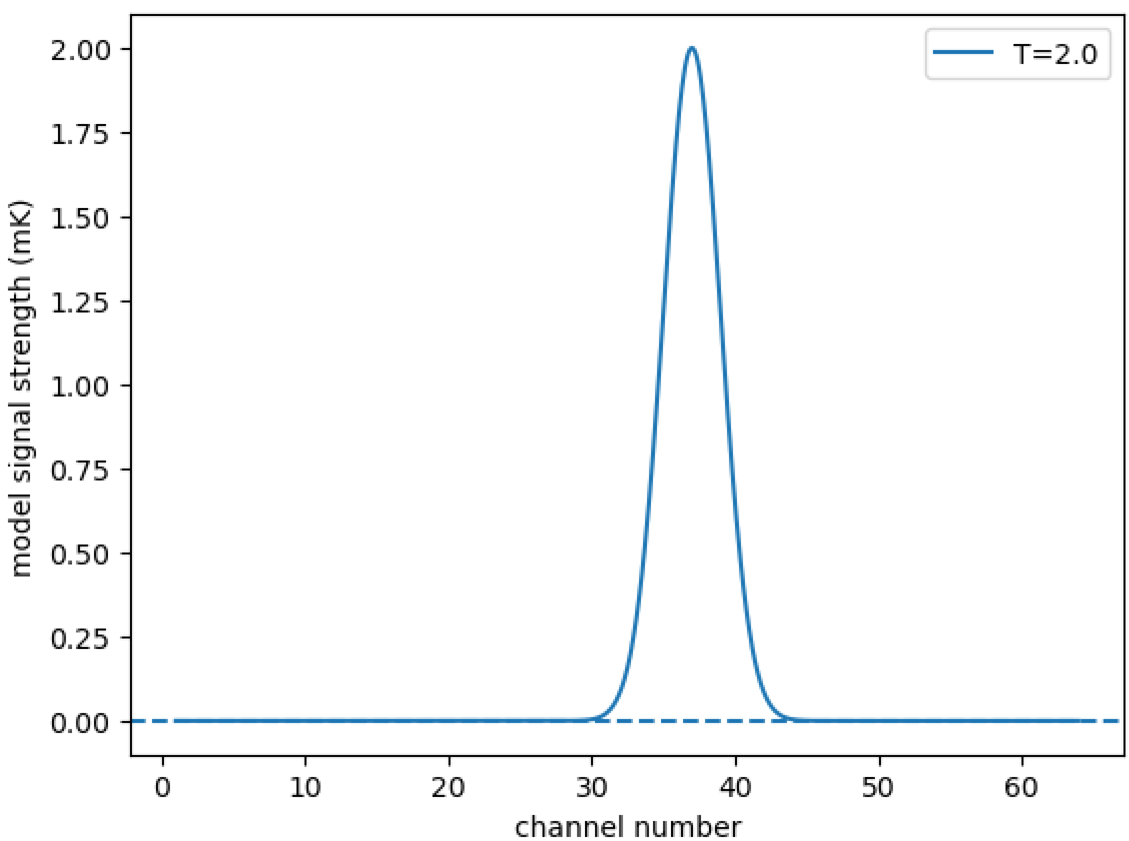
$$S(\text{model } 1) = T\ e^{-(\nu_i - \nu_0)^2/(2\sigma_L^2)} = T\ f_i$$

where $T$ is a free parameter that corresponds to the amplitude (height) of the gaussian, $\sigma_L$ is its intrinsic width, $\nu_0$ is its central channel, and $\nu_i$ is the channel number.

In [2]:
```python
def model1(nu, T, nu0=37.0, sigL=2.0):
    S = T*np.exp(-((nu-nu0)**2)/(2*sigL*sigL))
    return S
```

In [3]:
```python
testchannel = np.arange(1, 64.1, 0.1)
testmodel1  = model1(testchannel, 2.0)
plt.plot(testchannel, testmodel1, label='T=2.0')
plt.axhline(y=0, linestyle="--")
plt.xlabel("channel number")
plt.ylabel("model signal strength (mK)")
plt.legend()
```

Out[3]: <matplotlib.legend.Legend at 0x1115e88e0>

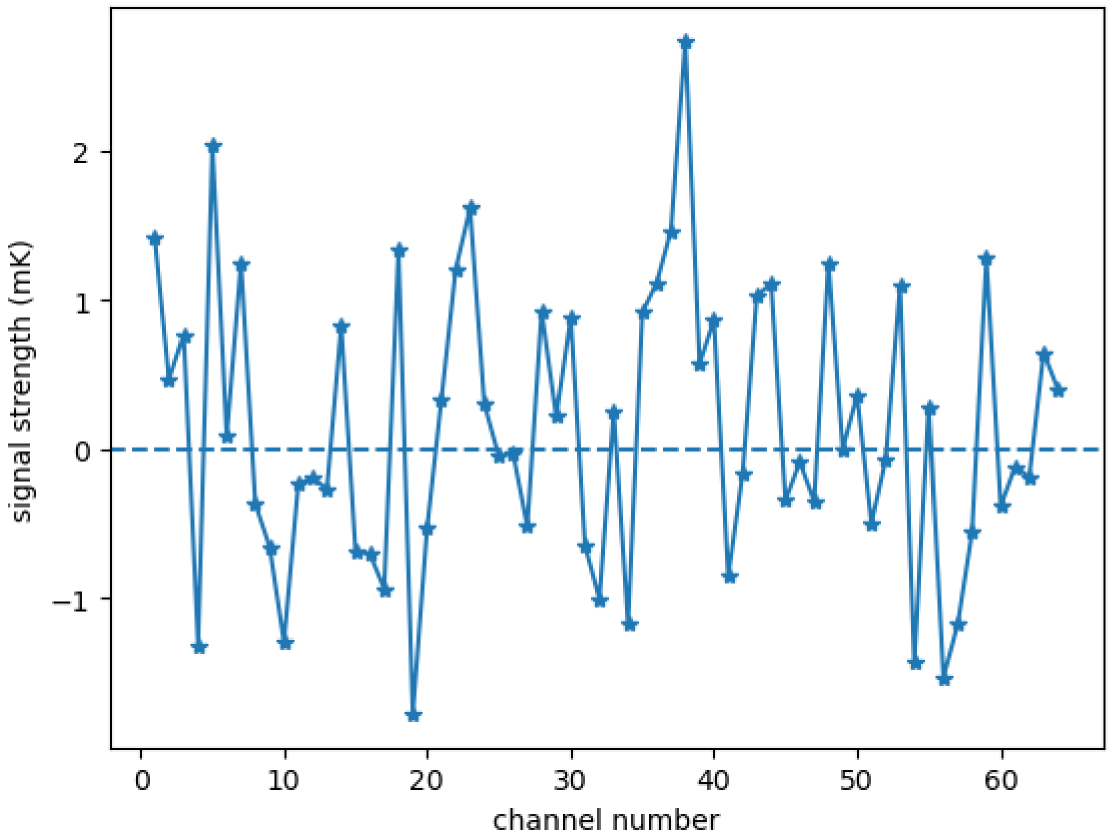


And the dataset looks like the following.

In [4]:
```
data1   = np.genfromtxt('SpectralLine1.dat')
channel = data1[:, 0]
signal  = data1[:, 1]
```

In [5]:
```
plt.plot(channel, signal, marker='*')
plt.axhline(y=0, linestyle="--")
plt.xlabel("channel number")
plt.ylabel("signal strength (mK)")
```

Out[5]: `Text(0, 0.5, 'signal strength (mK)')`

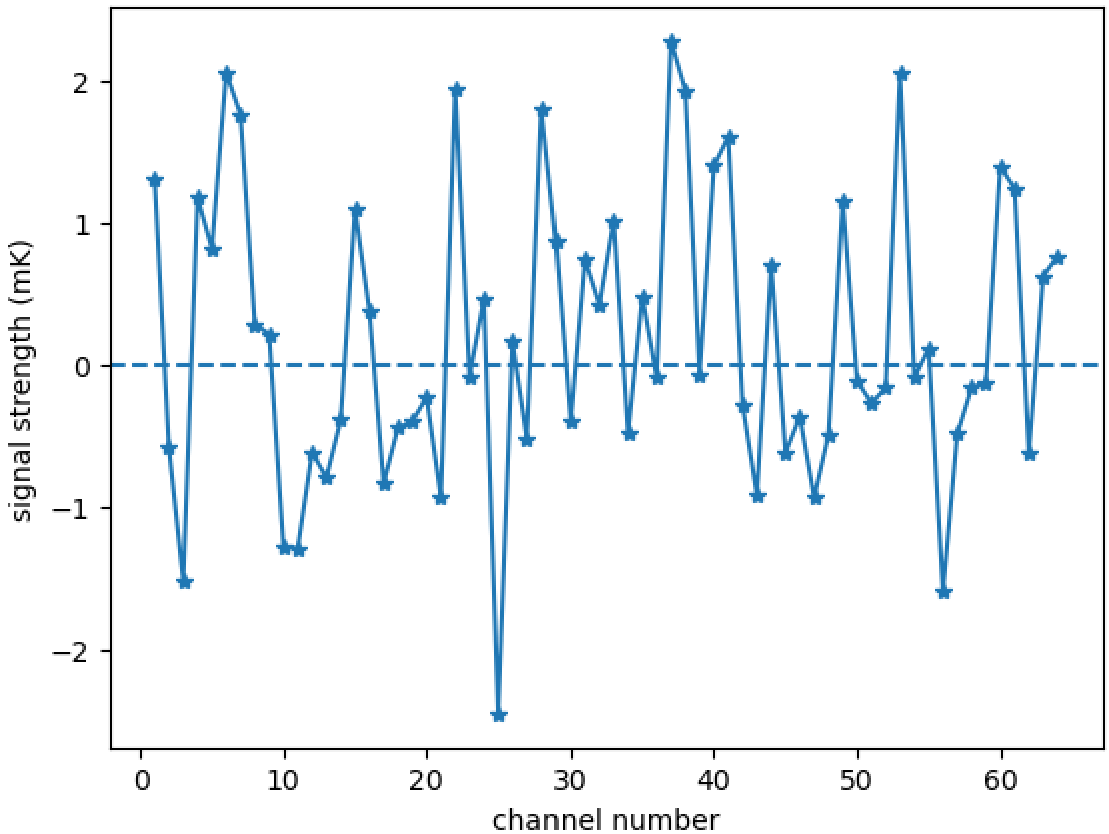



The most likely value of $T$ was something like 1.57 mK, but in the following let's assume $T = 2.0$ mK and simulate another dataset.

In [6]:
```
newmodel = model1(channel, 2.0)
```

In [7]:
```
np.random.seed(239847)
newsignal = np.random.normal(newmodel, scale=1.0)
plt.plot(channel, newsignal, marker='*')
plt.axhline(y=0, linestyle="--")
plt.xlabel("channel number")
plt.ylabel("signal strength (mK)")
```

Out[7]: `Text(0, 0.5, 'signal strength (mK)')`



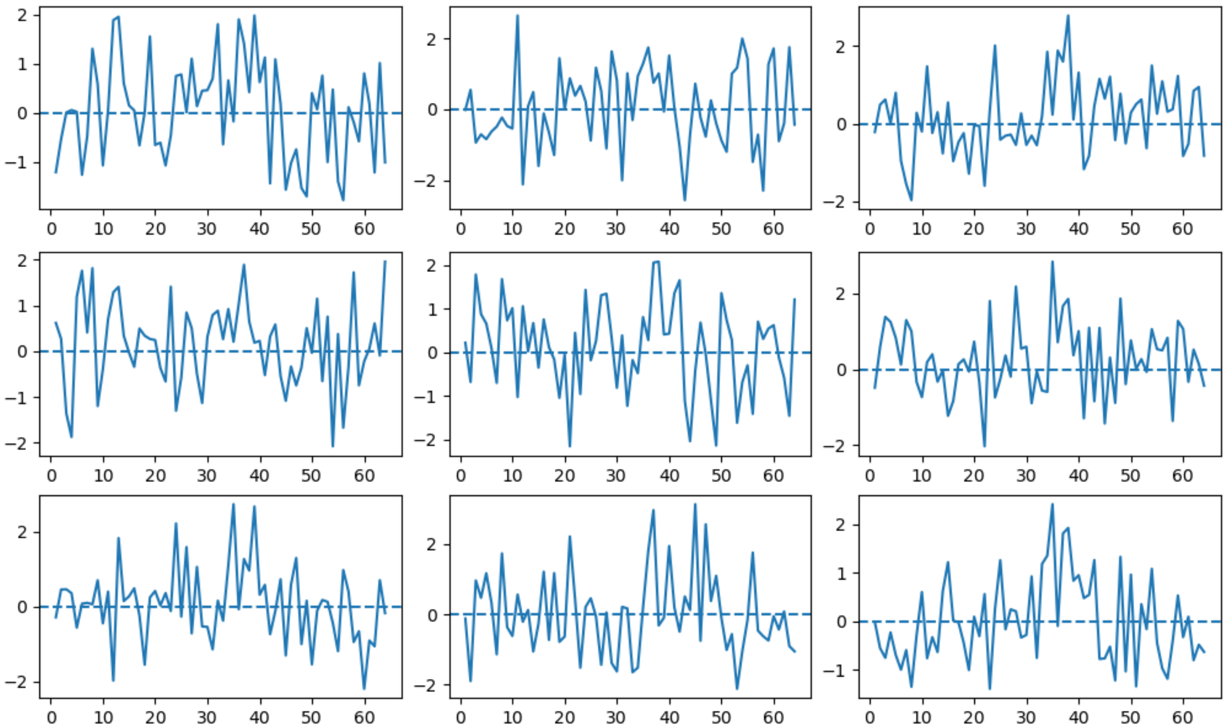

It is quite noisy primarily because the signal is $T = 2.0$ mK while the noise is $\sigma = 1.0$ mK. Let's simulate 9 more realizations from the same model. This can be thought of as 9 additional independent experiments measuring the same thing.

In [8]:
```
newsignal1 = np.random.normal(newmodel, scale=1.0)
newsignal2 = np.random.normal(newmodel, scale=1.0)
newsignal3 = np.random.normal(newmodel, scale=1.0)
newsignal4 = np.random.normal(newmodel, scale=1.0)
newsignal5 = np.random.normal(newmodel, scale=1.0)
newsignal6 = np.random.normal(newmodel, scale=1.0)
newsignal7 = np.random.normal(newmodel, scale=1.0)
newsignal8 = np.random.normal(newmodel, scale=1.0)
newsignal9 = np.random.normal(newmodel, scale=1.0)
```

In [9]:
```
fig, axs = plt.subplots(3, 3, figsize=(10, 6),
                       constrained_layout=True)
axs[0,0].plot(channel, newsignal1)
axs[0,1].plot(channel, newsignal2)
axs[0,2].plot(channel, newsignal3)
axs[1,0].plot(channel, newsignal4)
axs[1,1].plot(channel, newsignal5)
axs[1,2].plot(channel, newsignal6)
axs[2,0].plot(channel, newsignal7)
axs[2,1].plot(channel, newsignal8)
axs[2,2].plot(channel, newsignal9)
axs[0,0].axhline(y=0, linestyle="--")
axs[0,1].axhline(y=0, linestyle="--")
axs[0,2].axhline(y=0, linestyle="--")
axs[1,0].axhline(y=0, linestyle="--")
```

```
axs[1,1].axhline(y=0, linestyle="--")
axs[1,2].axhline(y=0, linestyle="--")
axs[2,0].axhline(y=0, linestyle="--")
axs[2,1].axhline(y=0, linestyle="--")
axs[2,2].axhline(y=0, linestyle="--")
```

Out[9]: <matplotlib.lines.Line2D at 0x1119cf700>

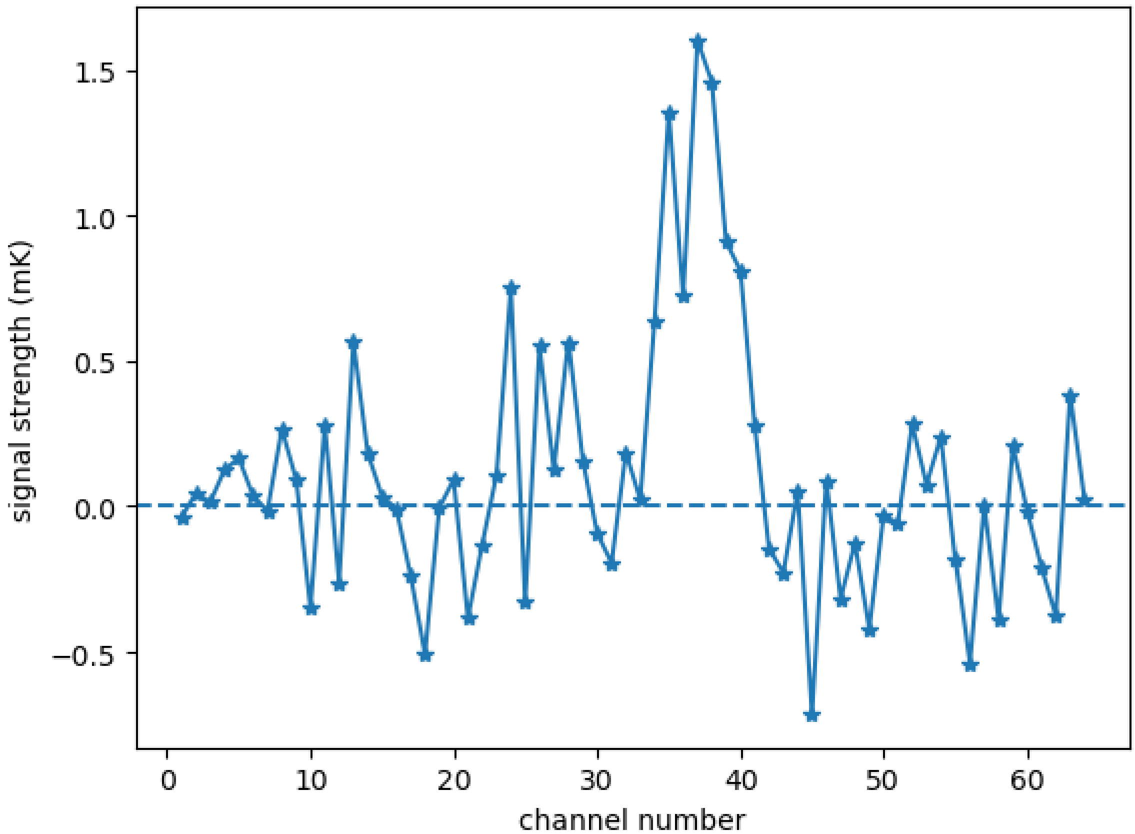


They all look similar and very noisy. However, if we take the average, you can see that the signal-to-noise ratio increases because the noise decreases as $\sigma/\sqrt{n}$.

In [10]:
```
avesignal = 0.1*(newsignal + newsignal1
                 + newsignal2 + newsignal3
                 + newsignal4 + newsignal5
                 + newsignal6 + newsignal7
                 + newsignal7 + newsignal9)
```

In [11]:
```
plt.plot(channel, avesignal, marker='*')
plt.axhline(y=0, linestyle="--")
plt.xlabel("channel number")
plt.ylabel("signal strength (mK)")
```

Out[11]: Text(0, 0.5, 'signal strength (mK)')




Let's simulate $n = 100$ realizations and see the further improvement.

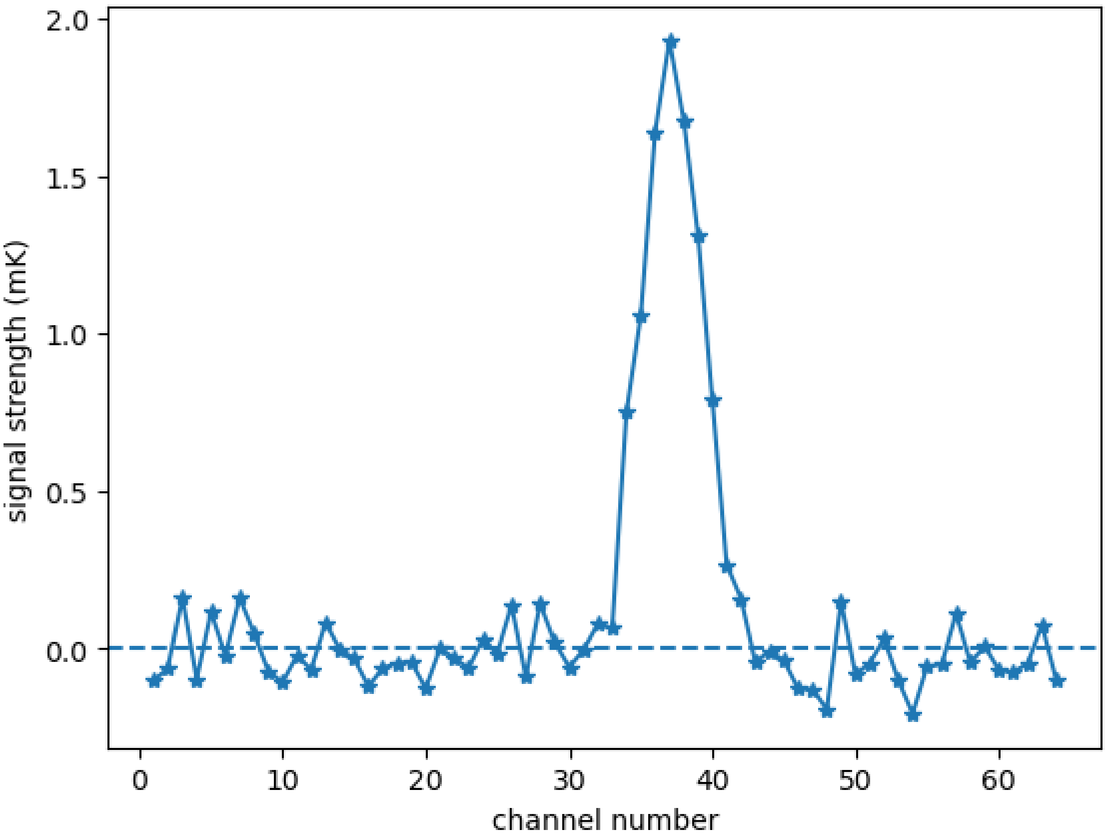
In [12]:
```
n = 100
signals = np.empty([n, 64])
np.random.seed(239847)
for i in np.arange(n):
    signals[i] = np.random.normal(newmodel, scale=1.0)

avesignal = np.sum(signals, axis=0)/n
```

In [13]:
```
print(signals.shape)
```

```
(100, 64)
```

In [14]:
```
plt.plot(channel, avesignal, marker='*')
plt.axhline(y=0, linestyle="--")
plt.xlabel("channel number")
plt.ylabel("signal strength (mK)")
```

Out[14]:
```
Text(0, 0.5, 'signal strength (mK)')
```


2.0
1.5
1.0
0.5
0.0
signal strength (mK)
0
10
20
30
40
50
60
channel number

In [ ]: