0.1 Second step: Fit a cubic to the wrist force data (18 rows each successful/failed)

Split by successful and failed.

TODO: See slides for what the final plots should look like. I've handled the for loop and the plot indexing for you.

At this point, you should have the data in the form ts, ys for both the successful and the failed data.

Where you're headed (pseudo code)

for 18 rows of data

```
Fit a cubic polynomial to the data

Plot the original data plus the fitted polynomial

   First three rows: successful, last three rows: failed
```

Implementation steps: You could write a function for doing the polynomial fit, but since there already is one (polynomial.polyval) you probably don't need one.

Steps - Fit polynomial for one row of data - Plot it and make sure it looks correct - Copy that code into the for loop, adjusting for which row (p) and which subplot (axs[r, c]) - Repeat for the failed data

Note: It can take a few seconds to plot.

In [ ]:
```
n_rows = 6
n_cols = 6
fig, axs = plt.subplots(n_rows, n_rows, figsize=(16, 16))
ch_name = "Wrist force"
for p in range(0, (n_rows // 2) * n_cols):
    r, c = p // n_cols, p % n_cols
    axs[r, c].plot(ts, data_wrist_force_successful[p])
    ...
    axs[r, c].set_title(f"{ch_name} summed" + f" suc {p}", fontsize=10)

for p in range(0, (n_rows // 2) * n_cols):
    r, c = 3 + p // n_cols, p % n_cols
    axs[r, c].plot(ts, data_wrist_force_failed[p])
    ...
    axs[r, c].set_title(f"{ch_name} summed" + f" fail {p}", fontsize=10)

plt.tight_layout()
```