Duffing-stuffing parameter estimation

Goals:

- Estimate parameters for a Duffing-like model such that it describes the behavior of the system with low error in different experimental schemes (varying resonnance frequencies, degradation, etc.).
- Simulate the system and perform various analyses (sensitivity, stability, etc.)

Table of contents

Empirical data

Data is measured through frequency scans at lab.

Read data from `.mat` -files as a dict of numpy arrays. We focus primarily on the XY-trace data, containing a stable-state period of 100 observations per frequency, for 5 experiments total with variying resonnance frequencies.

```
Reading first experiment
Variables (rows x observations):  ('x', 'y', 's', 'f', 'v', 't', 'XResAmp', 'XResfFreq', 'YResAmp',
'YResfFreq')
Resonnance frequencies: (7600, 8100)
Resonnance amplitudes: (1.03, 1.10)
T = 55.78
t in [0.00, 55.78]
f in [0.0, 10000.0]
x shape: 101 x 100
y shape: 101 x 100
```

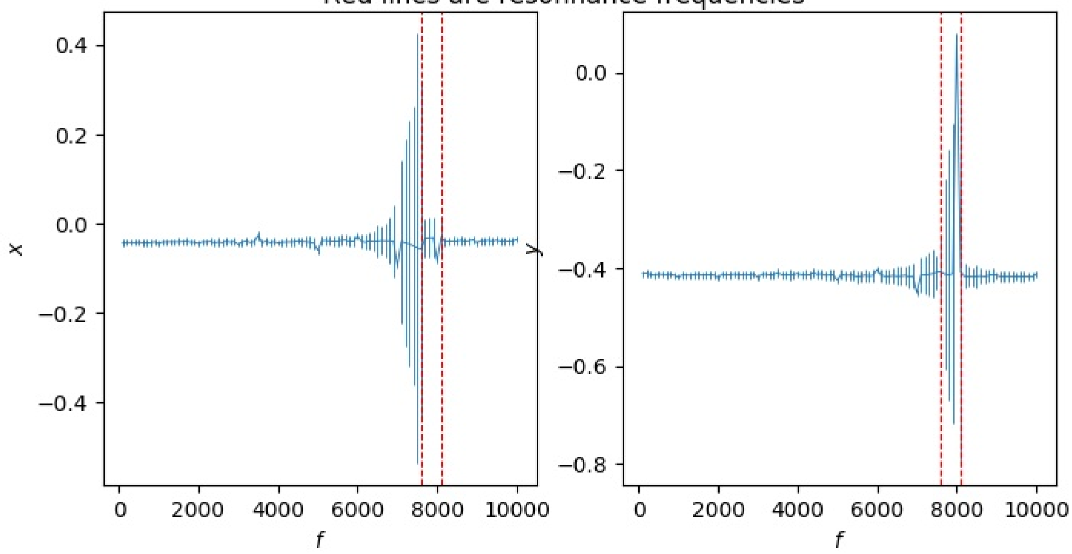
Plotting

Three types of plots:

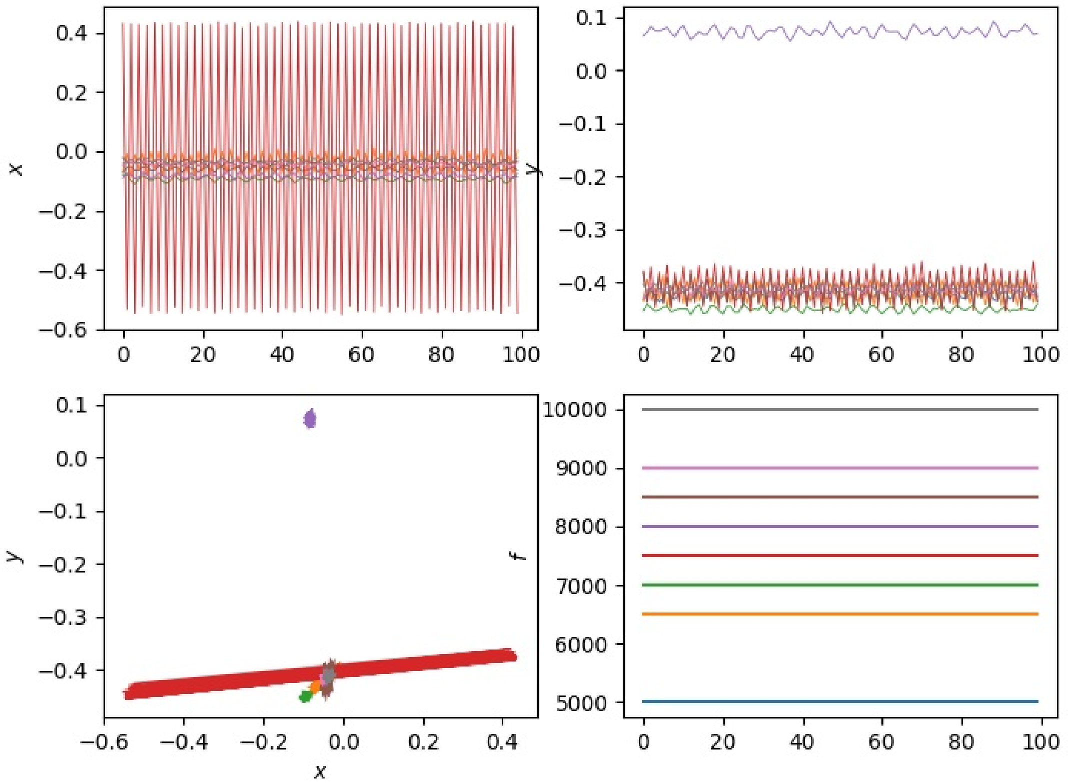
- Frequency scan with amplitude mean/std.
- Trajectory plot in (x, y)-plane.
- Trajectory over time for x and y.

```
Variables (rows x observations):  ('x', 'y', 's', 'f', 'v', 't', 'XResAmp', 'XResfFreq', 'YResAmp',
'YResfFreq')
Resonnance frequencies: (7600, 8100)
Resonnance amplitudes: (1.03, 1.10)
T = 55.78
t in [0.00, 55.78]
f in [0.0, 10000.0]
x shape: 101 x 100
y shape: 101 x 100
```

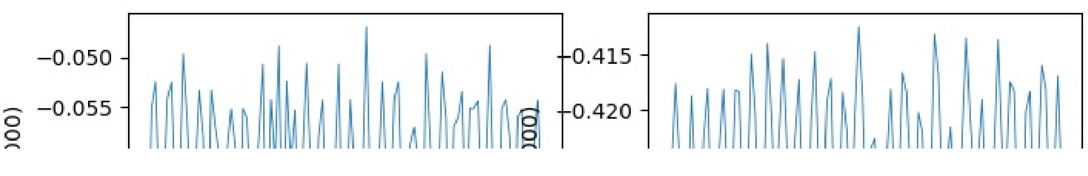
Amplitude mean and standard deviation per frequency
Red lines are resonnance frequencies

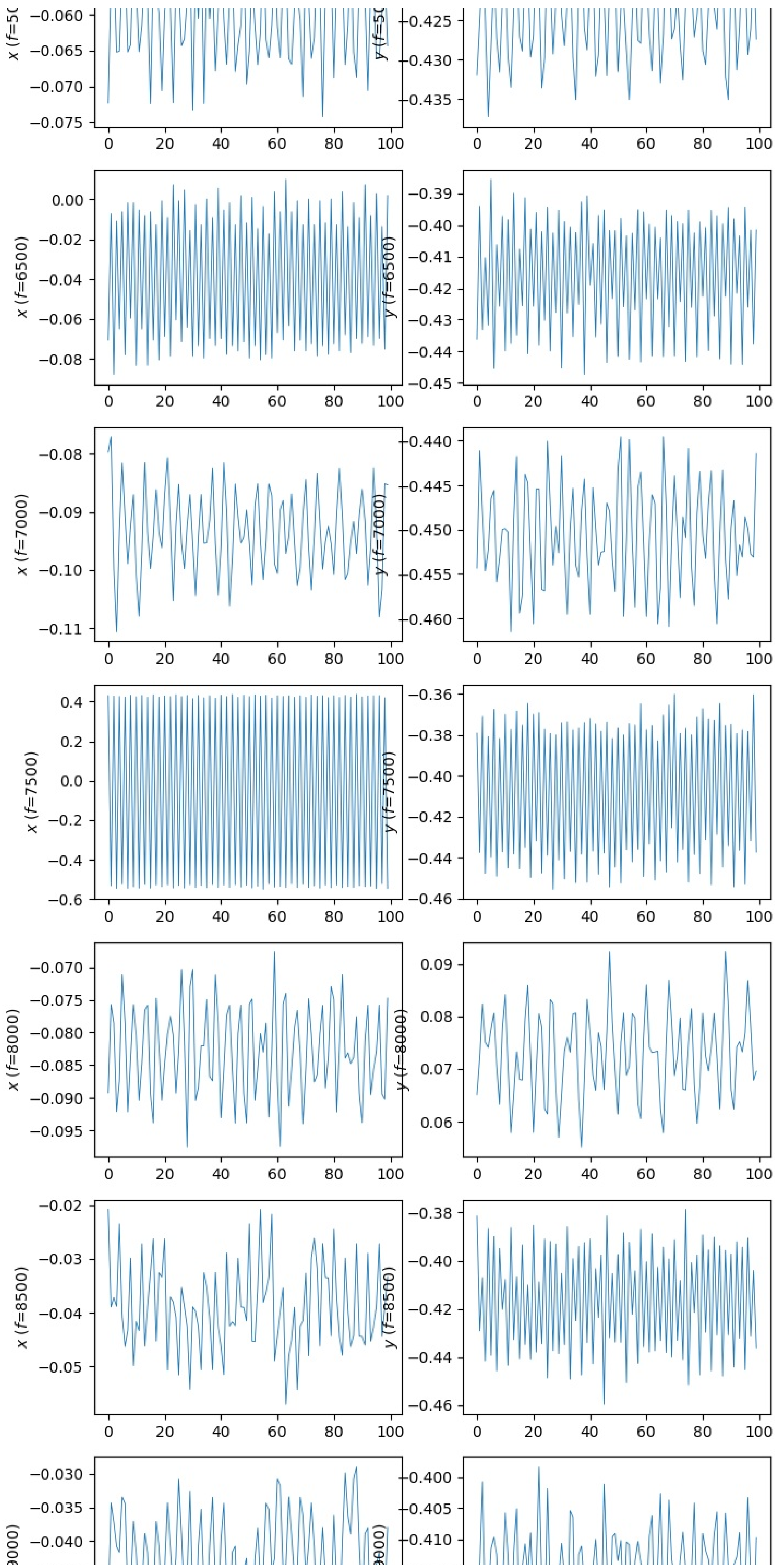


XY-data plots for given frequencies

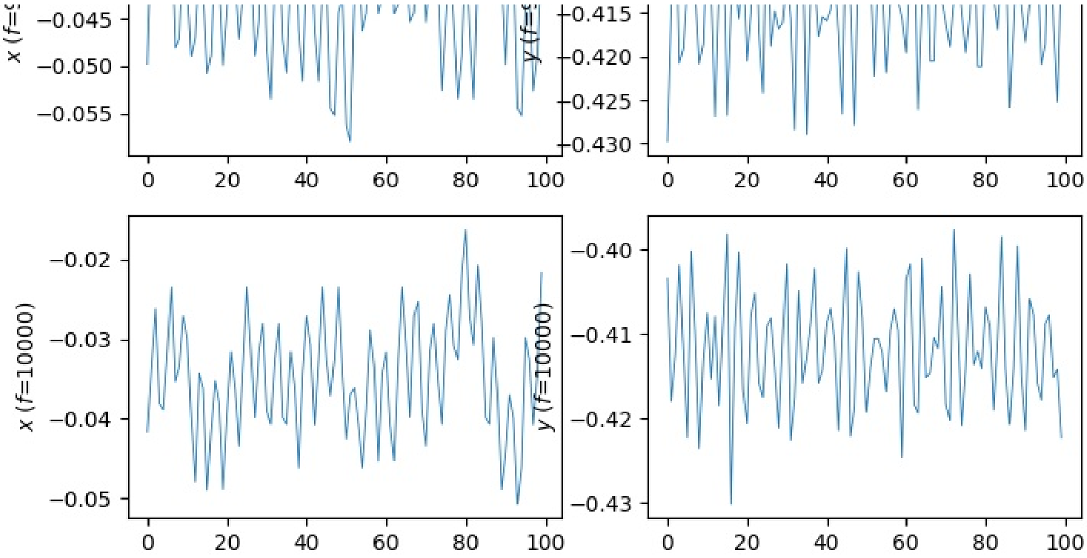


Stable-state XY-data plots for given frequencies





x (f=6500)
y (f=6500)
x (f=7000)
y (f=7000)
x (f=7500)
y (f=7500)
x (f=8000)
y (f=8000)
x (f=8500)
y (f=8500)
0
20
40
60
80
100
−0.060
−0.065
−0.070
−0.075
−0.425
−0.430
−0.435
0.00
−0.02
−0.04
−0.06
−0.08
−0.39
−0.40
−0.41
−0.42
−0.43
−0.44
−0.45
−0.08
−0.09
−0.10
−0.11
−0.440
−0.445
−0.450
−0.455
−0.460
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.36
−0.38
−0.40
−0.42
−0.44
−0.46
−0.070
−0.075
−0.080
−0.085
−0.090
−0.095
0.09
0.08
0.07
0.06
−0.02
−0.03
−0.04
−0.05
−0.38
−0.40
−0.42
−0.44
−0.46
−0.030
−0.035
−0.040
−0.400
−0.405
−0.410



Model

Model derived from Duffing equations:

$$m_1\ddot{y}_1 = F_1 - \dot{y}_1(c_1 + c_3) + \dot{y}_2 c_3 - y_1(k_1 + k_3) + y_2 k_3 - \alpha_1 y_1^3 + \alpha_3(y_2 - y_1)^3$$
$$m_2\ddot{y}_2 = F_2 - \dot{y}_2(c_2 + c_3) + \dot{y}_1 c_3 - x_2(k_2 + k_3) + y_1 k_3 - \alpha_2 y_2^3 + \alpha_3(y_2 - y_1)^3$$

where $F = Ce^{i\omega t}$. Transform to first-order form by variable substitutions $x_3 = \dot{y}_1, x_1 = y_1$ and $x_4 = \dot{y}_2, x_2 = y_2$:

$$\dot{x}_1 = x_3$$
$$\dot{x}_2 = x_4$$
$$m_1\dot{x}_3 = F_1 - x_3(c_1 + c_3) + x_4 c_3 - x_1(k_1 + k_3) + x_2 k_3 - \alpha_1 x_1^3 + \alpha_3(x_2 - x_1)^3$$
$$m_2\dot{x}_4 = F_2 - x_4(c_2 + c_3) + x_3 c_3 - x_2(k_2 + k_3) + x_1 k_3 - \alpha_2 x_2^3 + \alpha_3(x_2 - x_1)^3$$

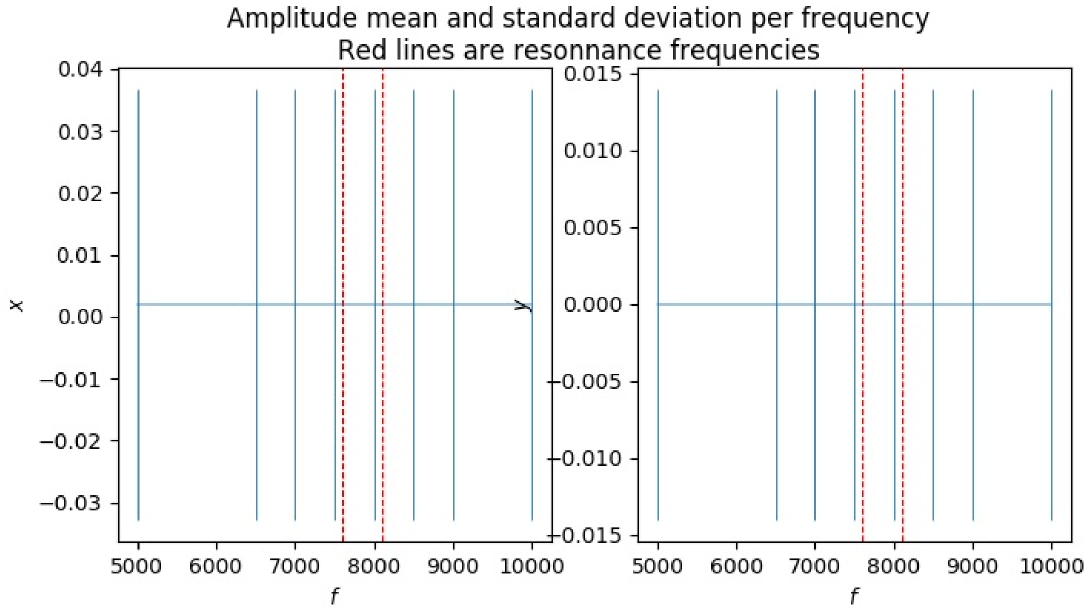
For some reason, this model doesn't work out when working backwards from resonnance frequencies. I may be missing something obvious, otherwise the fact that mass goes into the estimations may mess things up. See plots of simulation below.

Notes on solvers

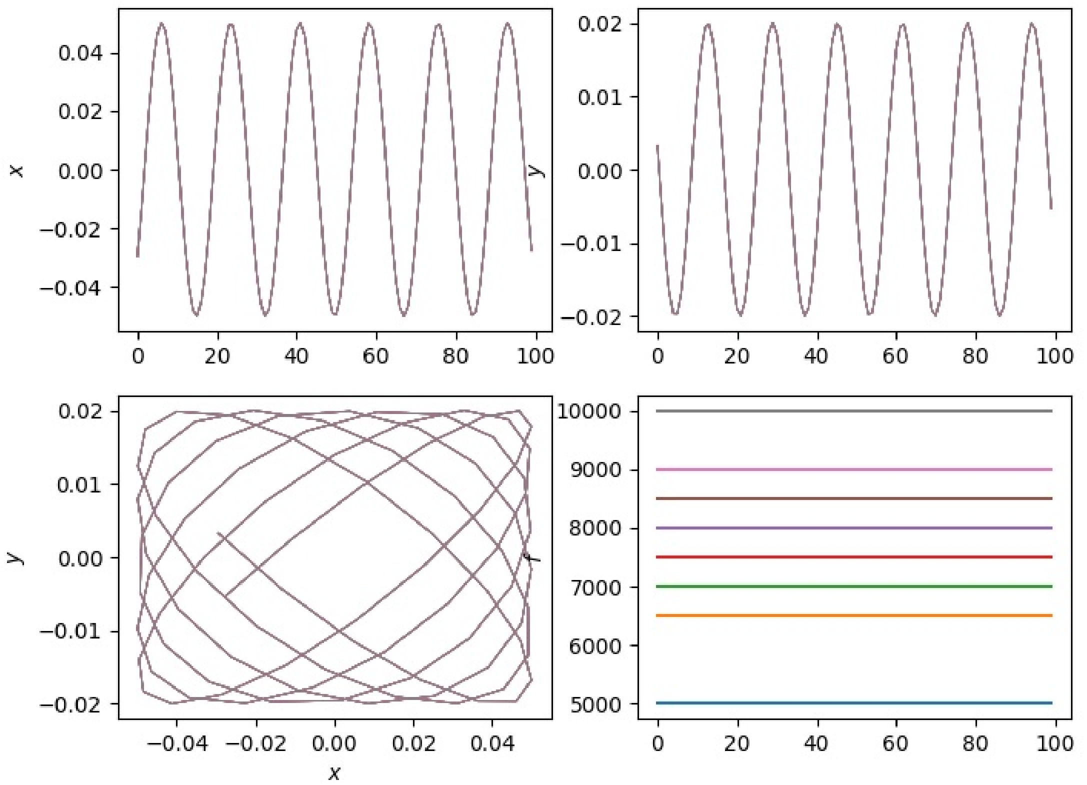
- We use SciPy's `solve_ivp` to simulate the system. Different methods (RK45, LSODA, Radeau) has been tested with no noticable differences.
- The standard `odeint` from SciPy is super shit. It easily diverges and is unstable. They claim to use the standard LSODA solver (same as `solve_ivp` with method='LSODA') but the results are entirely different.
- Once we hit the right parameters, the simulation is considerable slower because these are adaptive solvers.
- There is an ODE implementation from the PyDSTool package (https://github.com/robclewley/pydstool) which compiles to C and is much much faster.

TODO: Solve ODE with PyDSTool solver. Simulations takes 10-30 seconds per frequency, which results in way too long test cycles. With PyDSTool we can expect a magnitudal speedup.

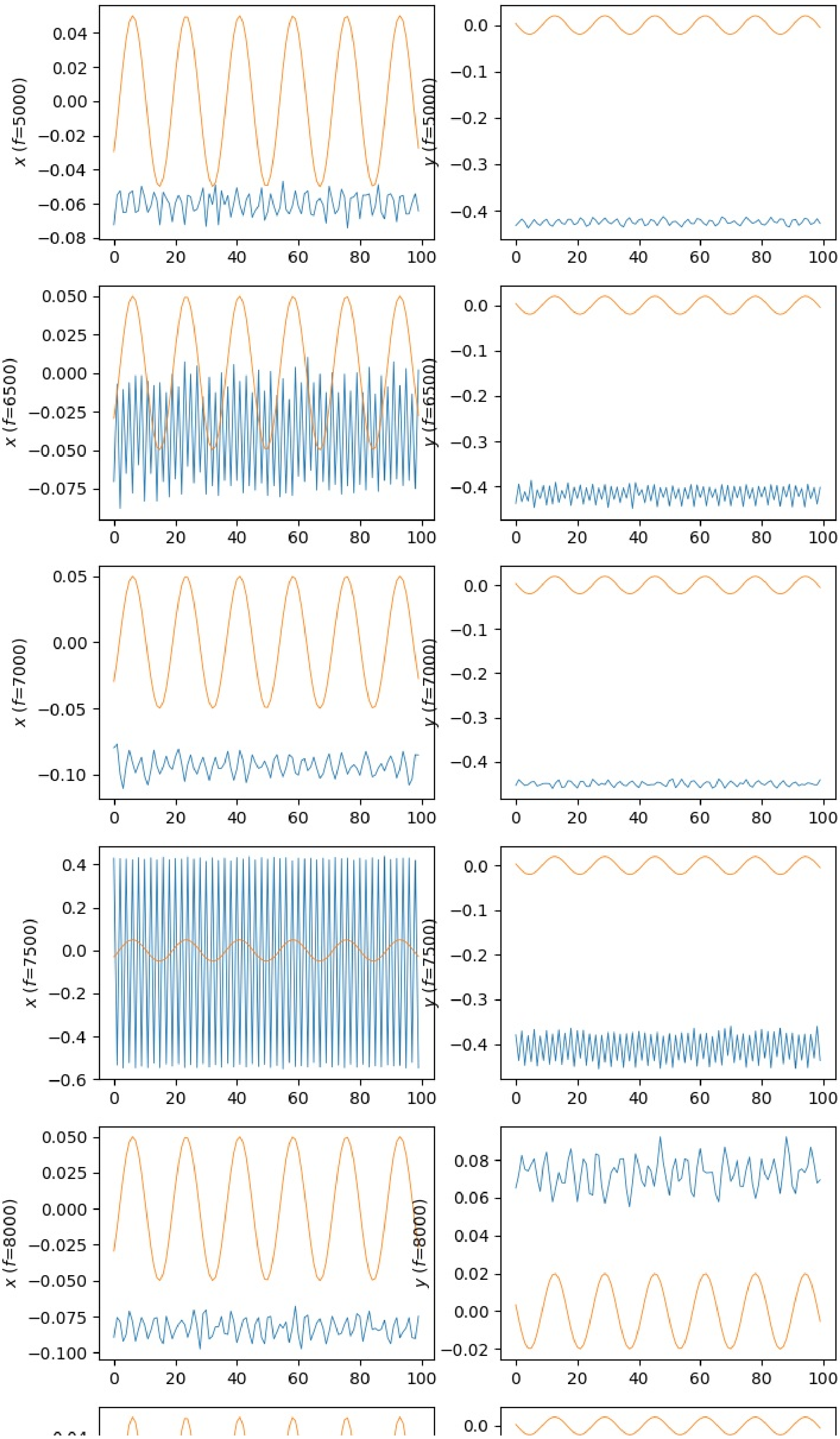
```
Omega0 1 = 47763.0
Omega0 2 = 50905.3
k1 = 3421950.0
k2 = 3887017.6
c1 = 2.1
c2 = 2.3
Variables (rows x observations):  ('x', 'y', 's', 'f', 'v', 't', 'XResAmp', 'XResfFreq', 'YResAmp',
'YResfFreq')
Resonnance frequencies: (7600, 8100)
Resonnance amplitudes: (1.03, 1.10)
T = 55.78
t in [0.00, 55.78]
f in [0.0, 10000.0]
x shape: 101 x 100
y shape: 101 x 100
```



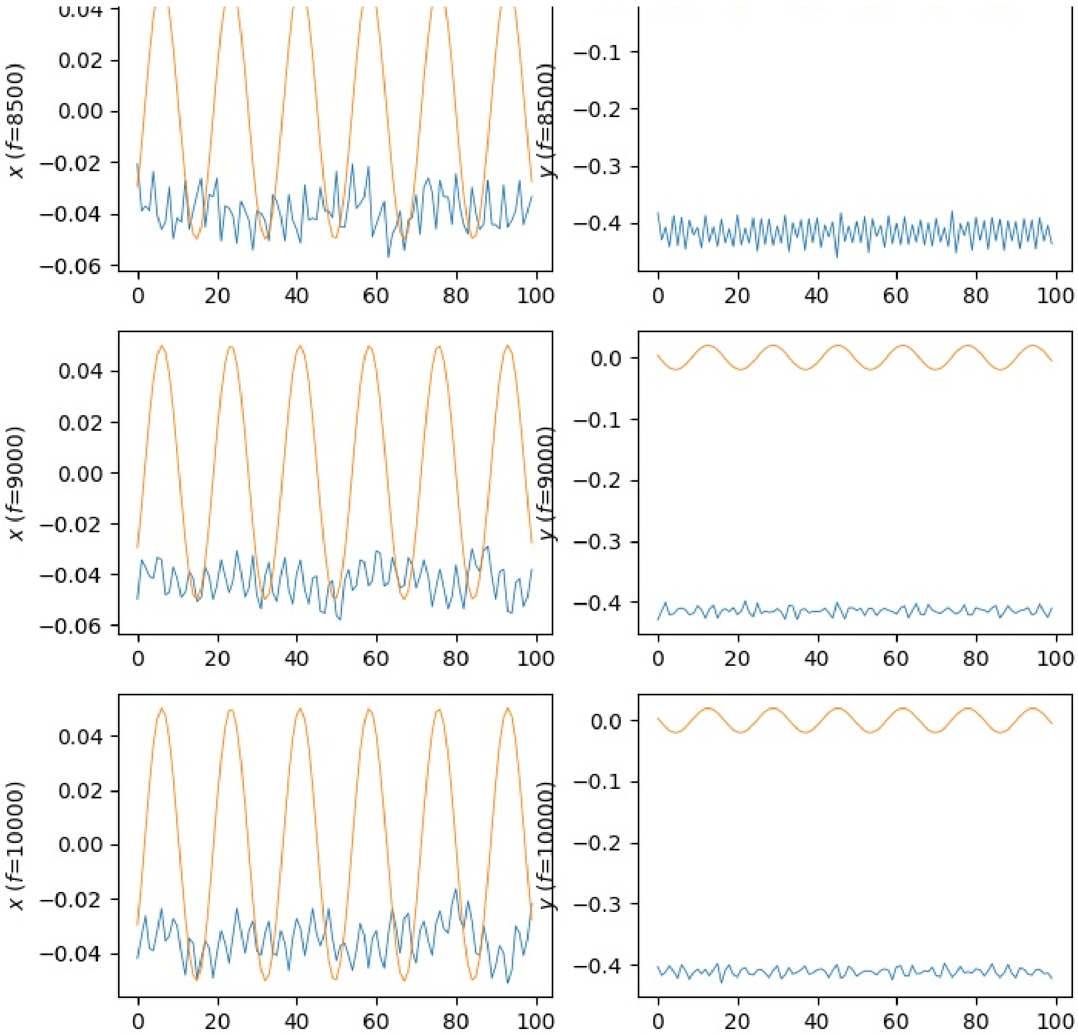
XY-data plots for given frequencies



Stable-state XY-data plots for given frequencies



x (f=5000)
y (f=5000)
x (f=6500)
y (f=6500)
x (f=7000)
y (f=7000)
x (f=7500)
y (f=7500)
x (f=8000)
y (f=8000)



x (f=8500)
y (f=8500)
x (f=9000)
y (f=9000)
x (f=10000)
y (f=10000)

Rewritten model

Eliminate mass, + easier to reason about physical constants:

$$\begin{aligned}
\dot{x}_1 &= x_3 \\
\dot{x}_2 &= x_4 \\
\dot{x}_3 &= \frac{1}{m_1}F_1 - x_3(c_1 + c_3) + x_4 c_3 - x_1(k_1 + k_3) + x_2 k_3 - \alpha_1 x_1^3 + \alpha_3 (x_2 - x_1)^3 \\
\dot{x}_4 &= \frac{1}{m_2}F_2 - x_4(c_2 + c_3) + x_3 c_3 - x_2(k_2 + k_3) + x_1 k_3 - \alpha_2 x_2^3 + \alpha_3 (x_2 - x_1)^3
\end{aligned}$$

With harmonic oscillator identities

- Undamped angular frequency:

$$\omega_0 = \sqrt{\frac{k}{m}}$$

- Damping ratio:

$$\zeta = \frac{c}{2\sqrt{mk}}$$

- Resonant freqency:

$$\omega_r = \omega_0 \sqrt{1 - 2\zeta^2}, \zeta < \frac{1}{\sqrt{2}}$$

express the constants subject to estimation as

$$\begin{aligned}
c_1 &= 2\zeta_1 \omega_{0,1} \\
c_2 &= 2\zeta_2 \omega_{0,2} \\
c_3 &= g_c(c_1, c_2) \\
k_1 &= \omega_{0,1}^2 \\
k_2 &= \omega_{0,2}^2 \\
k_3 &= g_k(k_1, k_2) \\
\alpha_1 &= f_1(\mathbf{X}; \theta) \\
\alpha_2 &= f_2(\mathbf{X}; \theta) \\
\alpha_3 &= g_c(\alpha_1, \alpha_2)
\end{aligned}$$

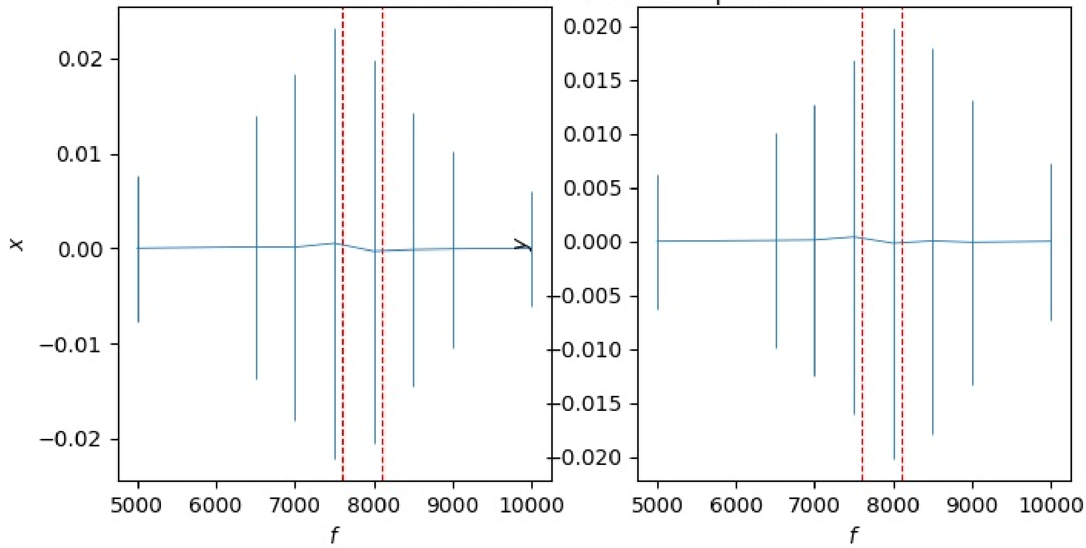
With the model expressed this way, things make sense and we get resonnance where it should be.

Harmonic oscillator

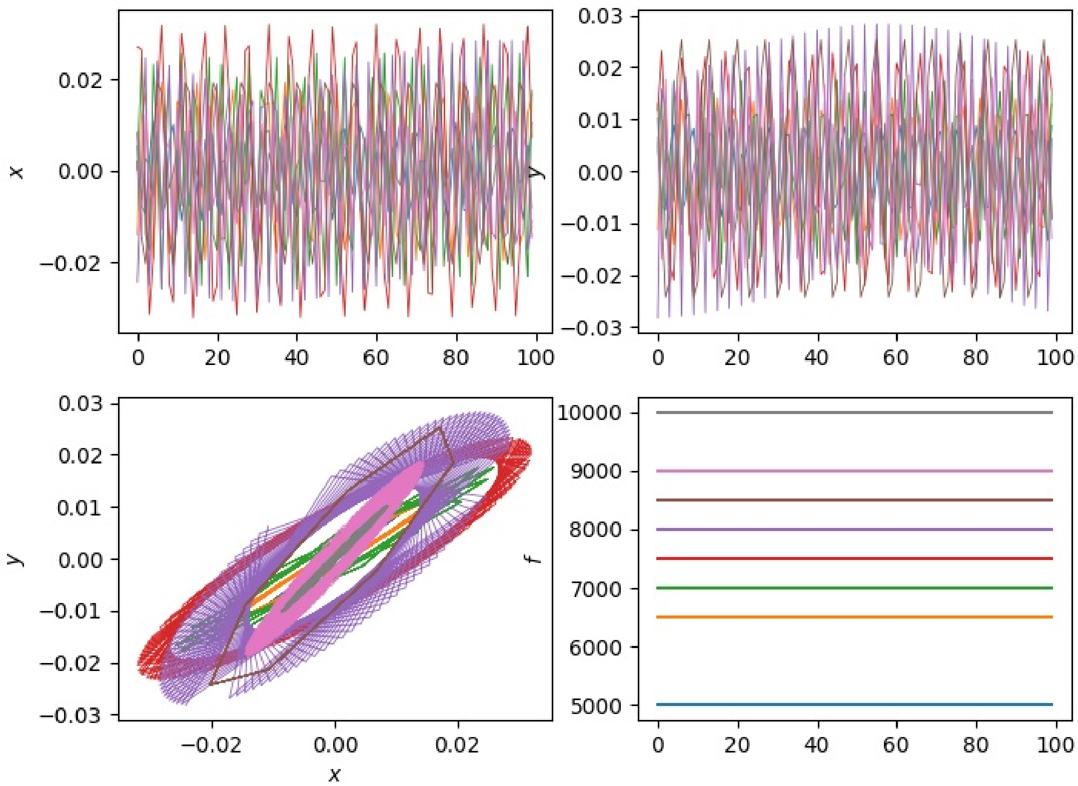
Set $c_3 = k_3 = \alpha_1 = \alpha_2 = \alpha_3 = 0$ for simulating a standard driven harmonic oscillator with no coupling between x- and y-components. Assume damping $\zeta_1 = \zeta_2 = 0.1$ and use resonnance frequencies from lab data to estimate parameters.

```
Read data, experiment 0
Variables (rows x observations):  ('x', 'y', 's', 'f', 'v', 't', 'XResAmp', 'XResfFreq', 'YResAmp',
'YResfFreq')
Resonnance frequencies: (7600, 8100)
Resonnance amplitudes: (1.03, 1.10)
T = 55.78
t in [0.00, 55.78]
f in [0.0, 10000.0]
x shape: 101 x 100
y shape: 101 x 100
Parameters:
Omega_r 1 = 47752.2
Omega_r 2 = 50893.8
Omega0 1 = 48237.0
Omega0 2 = 51410.5
c1 = 9647.4
c2 = 10282.1
c3 = 0.0
k1 = 2326809592.7
k2 = 2643039774.5
k3 = 0.0
a1 = 0.0
a2 = 0.0
a3 = 0.0
```

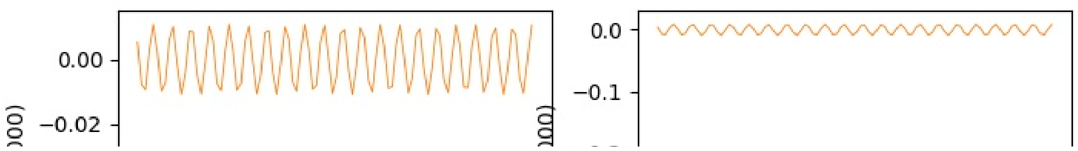
Amplitude mean and standard deviation per frequency
Red lines are resonnance frequencies

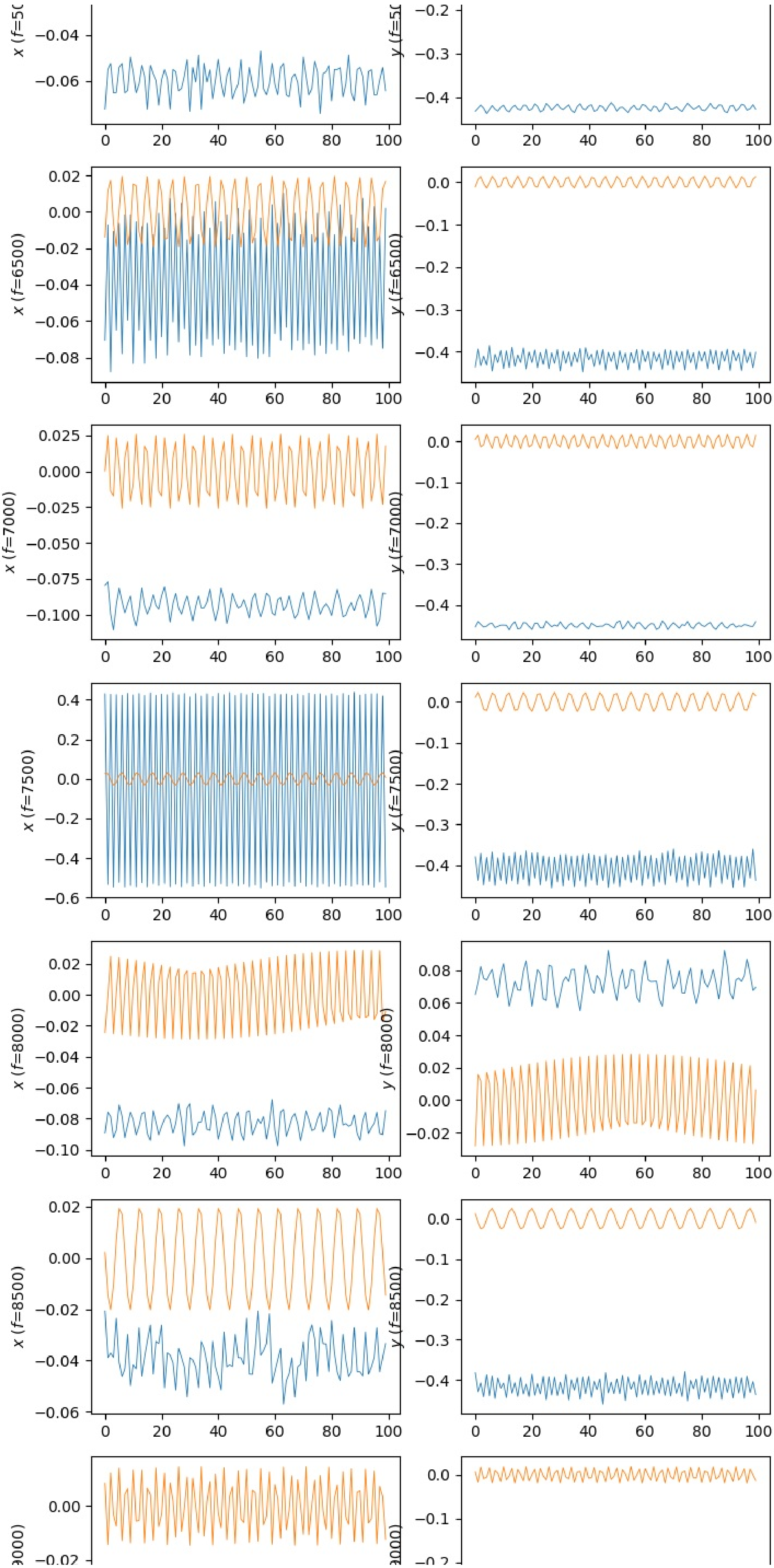


XY-data plots for given frequencies

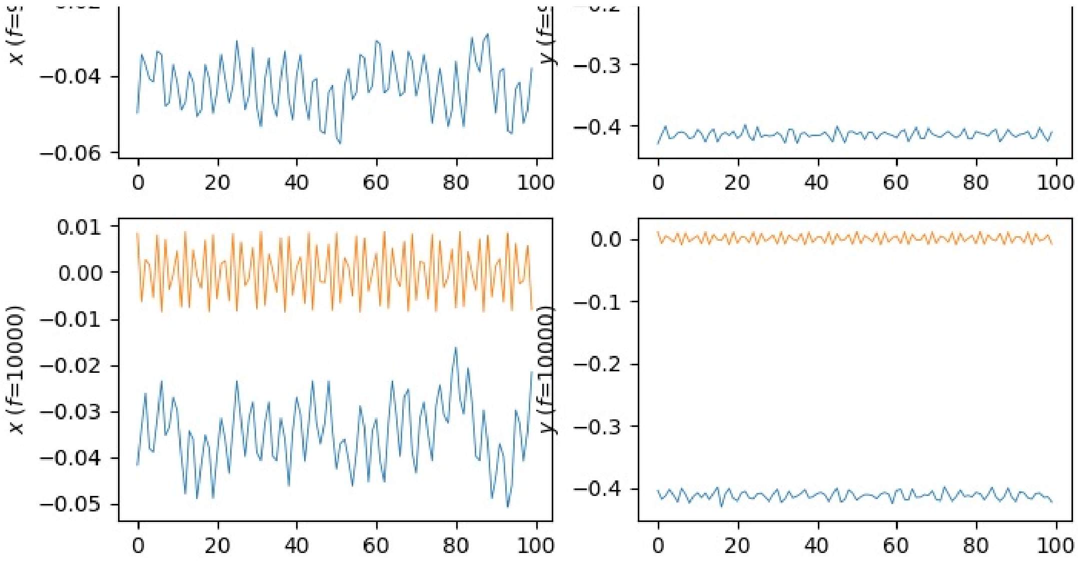


Stable-state XY-data plots for given frequencies





x (f=6500)
y (f=6500)
x (f=7000)
y (f=7000)
x (f=7500)
y (f=7500)
x (f=8000)
y (f=8000)
x (f=8500)
y (f=8500)

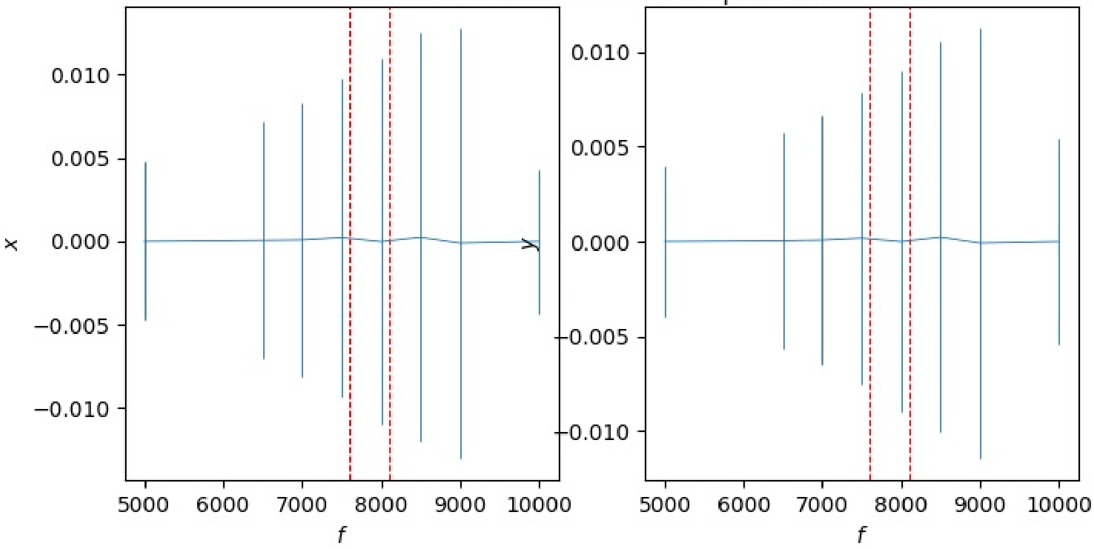


With duffing term

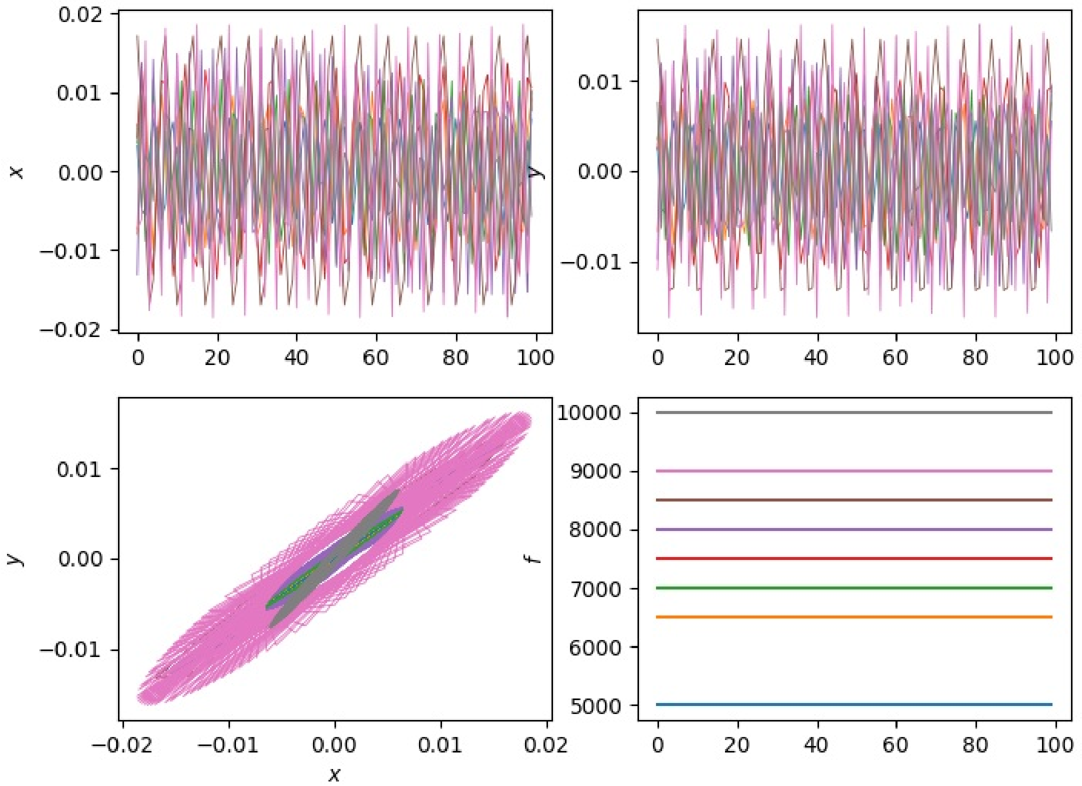
The duffing term has to be pretty large to see any stiffening effect. Set $\alpha_1 = 1500 k_1$ and $\alpha_2 = 1500 k_2$ (still without coupling).

```
Read data, experiment 0
Variables (rows x observations):  ('x', 'y', 's', 'f', 'v', 't', 'XResAmp', 'XResfFreq', 'YResAmp',
'YResfFreq')
Resonnance frequencies: (7600, 8100)
Resonnance amplitudes: (1.03, 1.10)
T = 55.78
t in [0.00, 55.78]
f in [0.0, 10000.0]
x shape: 101 x 100
y shape: 101 x 100
Parameters:
Omega_r 1 = 47752.2
Omega_r 2 = 50893.8
Omega0 1 = 48237.0
Omega0 2 = 51410.5
c1 = 9647.4
c2 = 10282.1
c3 = 0.0
k1 = 2326809592.7
k2 = 2643039774.5
k3 = 0.0
a1 = 3490214389022.0
a2 = 3964559661768.2
a3 = 0.0
```

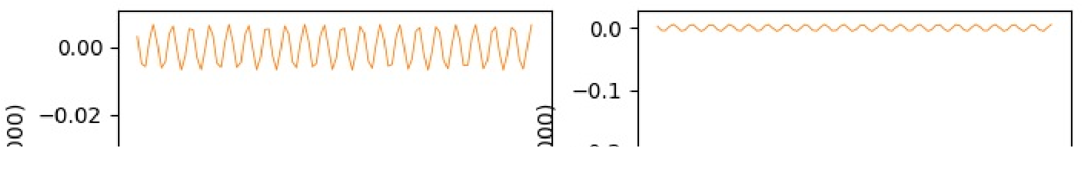
Amplitude mean and standard deviation per frequency
Red lines are resonnance frequencies

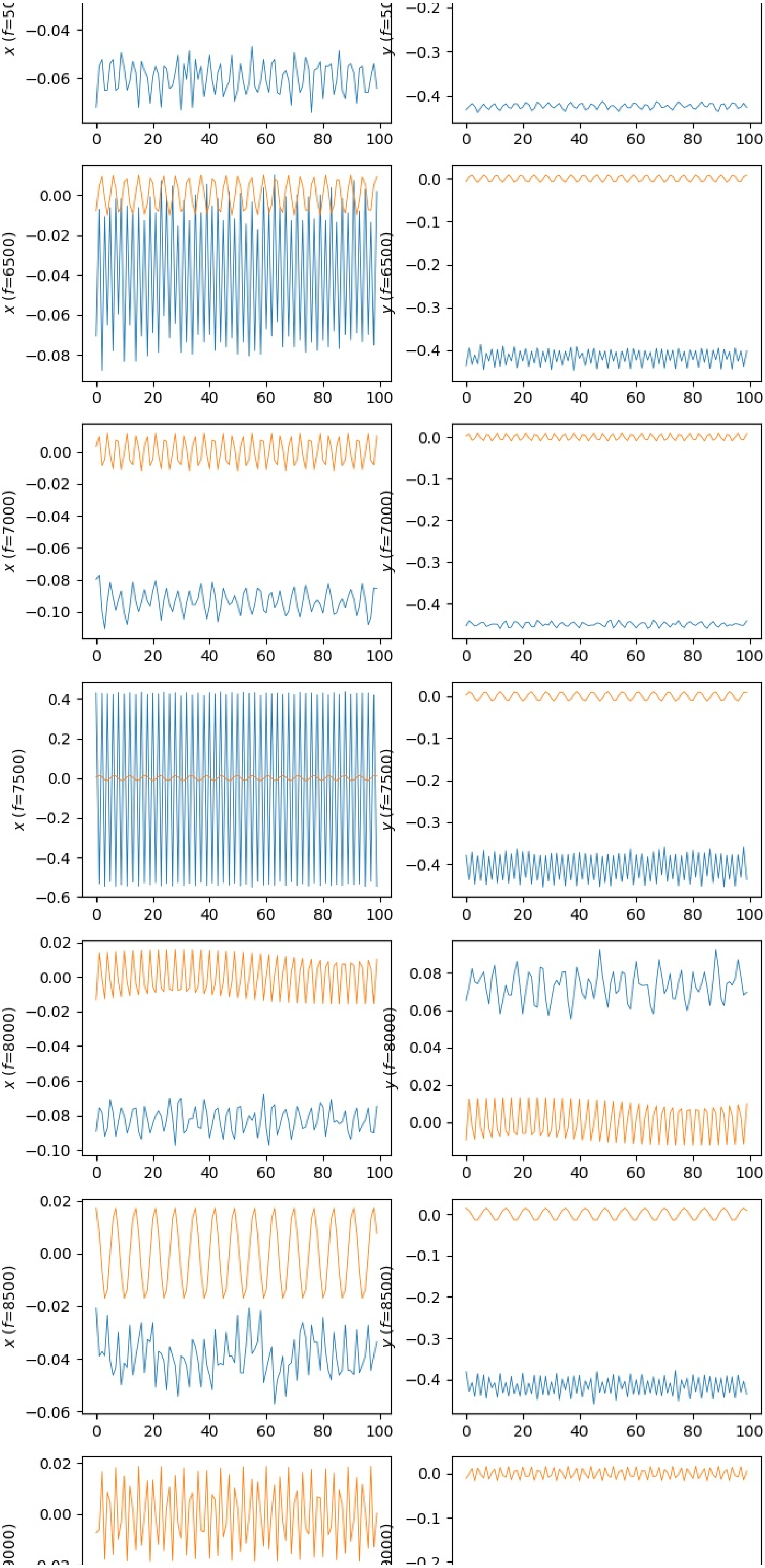


XY-data plots for given frequencies

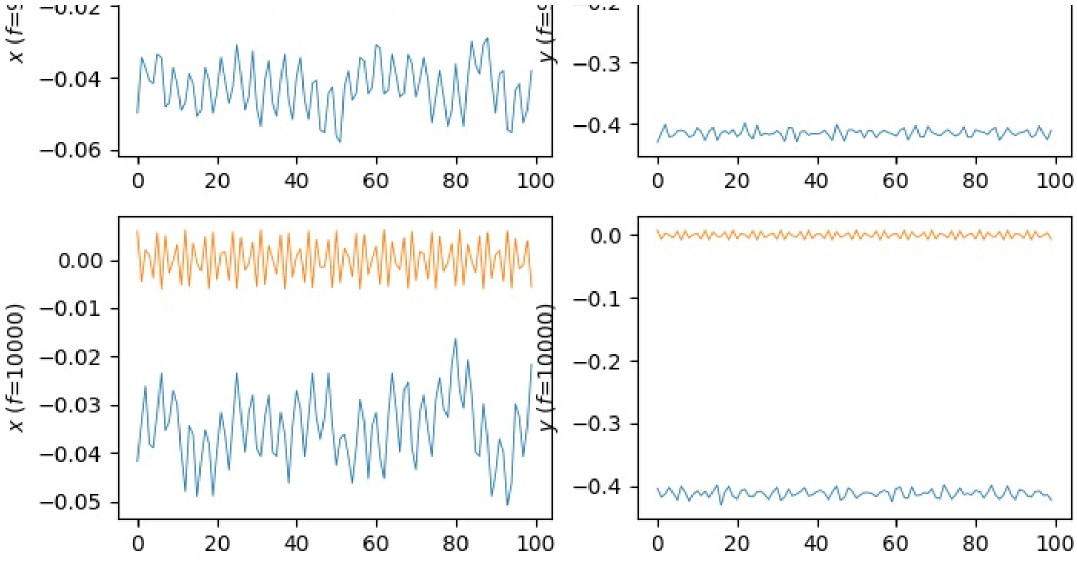


Stable-state XY-data plots for given frequencies





x (f=6500)
y (f=6500)
x (f=7000)
y (f=7000)
x (f=7500)
y (f=7500)
x (f=8000)
y (f=8000)
x (f=8500)
y (f=8500)



Loss function

Given the two multivariate signals, one empirical and one simulated, we need a distance metric $d(\mathbf{S}(\omega), \hat{\mathbf{S}}(\omega))$ that quantifies the error of our simulation.

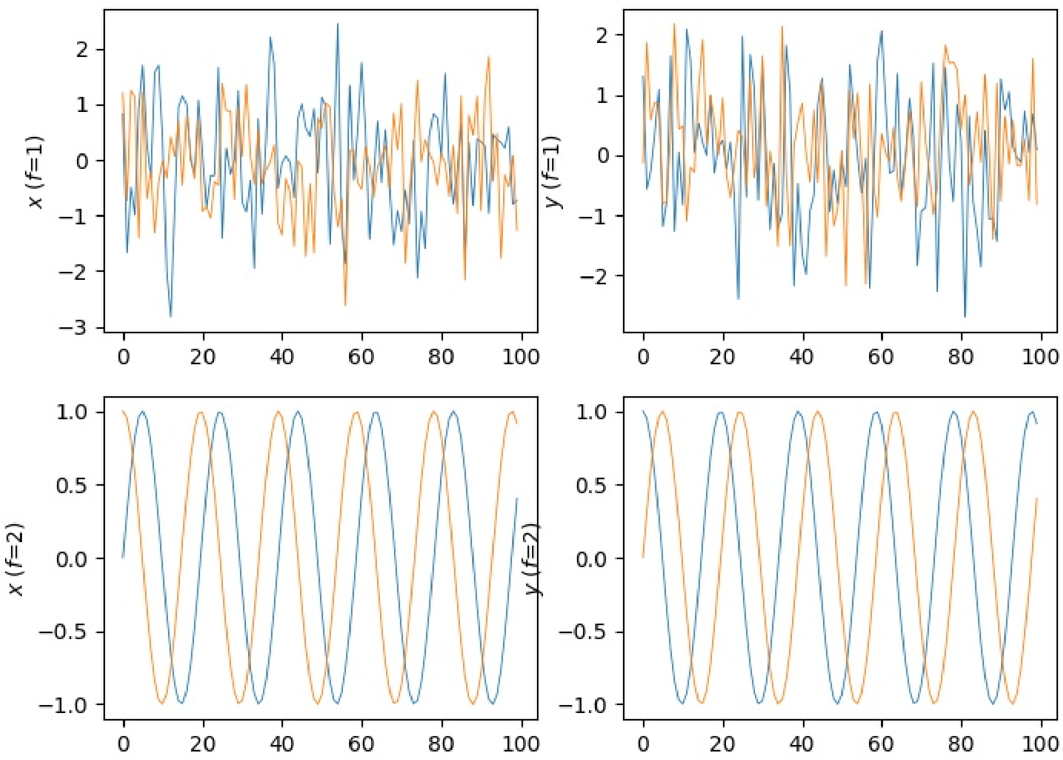
Consider first a single frequency $\mathbf{S}(\omega = x) \in \mathbb{R}^2$. Treat each component individually, perform autocorrelation do find the shift, then simply use mean squared error as the distance metric between the two common periods of $\mathbf{S}$ and $\hat{\mathbf{S}}$. We extend this to the multivariate case by simply averaging the loss for each frequency.

TODO: Assert that both components of $\mathbf{S}(\omega = x)$ have the same shift.

Loss function test

Random signals and sines.

Stable-state XY-data plots for given frequencies



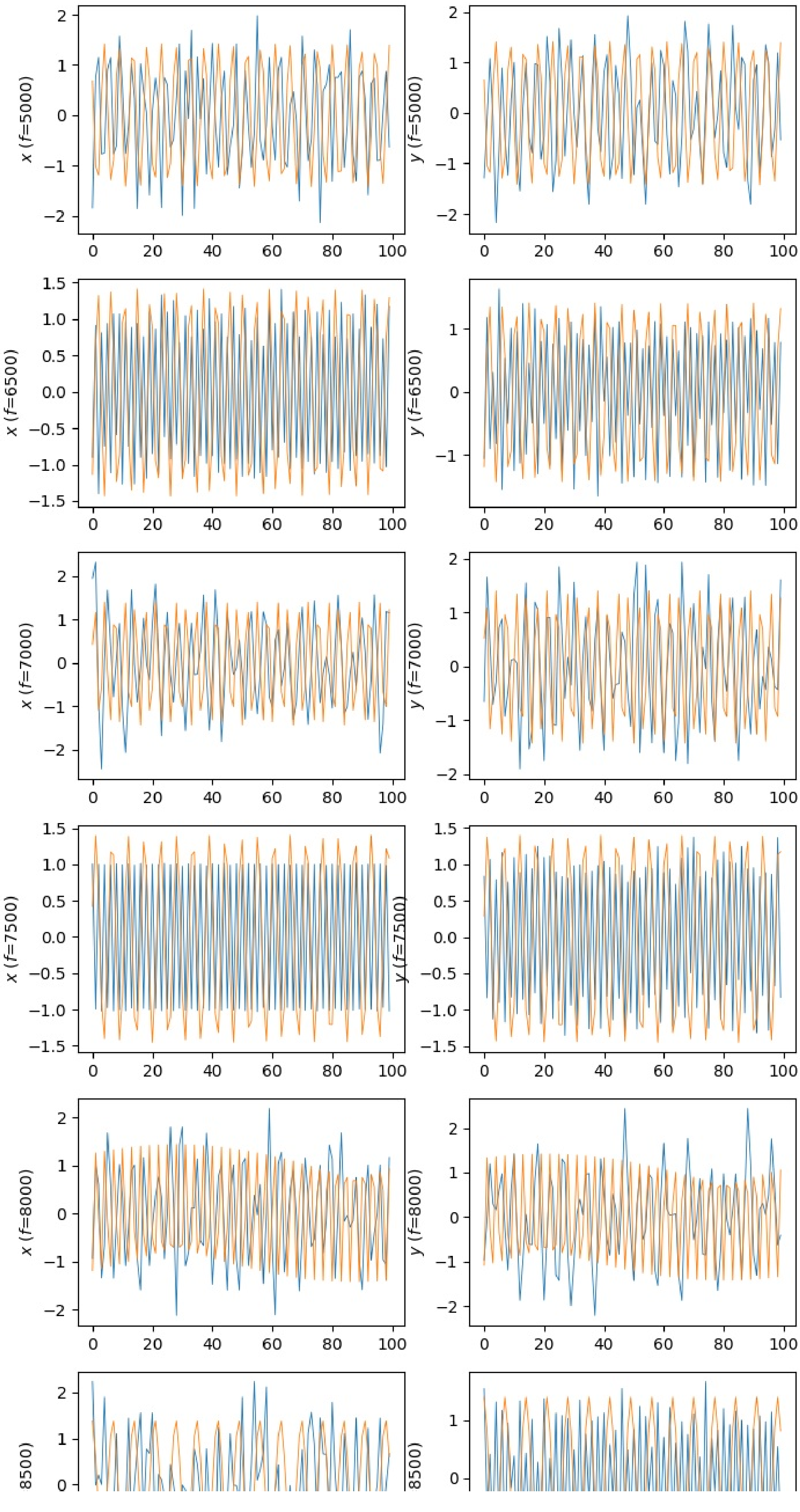

```
X idx: 32.0000 mean, 0.0000 std
Y idx: 1.0000 mean, 0.0000 std
X coeffs: 0.1348 mean, 0.0000 std
Y coeffs: 0.1414 mean, 0.0000 std
X MSEs: 1.4426 mean, 0.0000 std
Y MSEs: 1.7080 mean, 0.0000 std
Loss: 3.1506

X idx: 5.0000 mean, 0.0000 std
Y idx: 15.0000 mean, 0.0000 std
X coeffs: 0.9416 mean, 0.0000 std
Y coeffs: 0.8535 mean, 0.0000 std
X MSEs: 0.0022 mean, 0.0000 std
Y MSEs: 0.0127 mean, 0.0000 std
Loss: 0.0149

X idx: 18.5000 mean, 13.5000 std
Y idx: 8.0000 mean, 7.0000 std
X coeffs: 0.5382 mean, 0.4034 std
Y coeffs: 0.4975 mean, 0.3560 std
X MSEs: 0.7224 mean, 0.7202 std
Y MSEs: 0.8603 mean, 0.8476 std
Loss: 1.5827
```

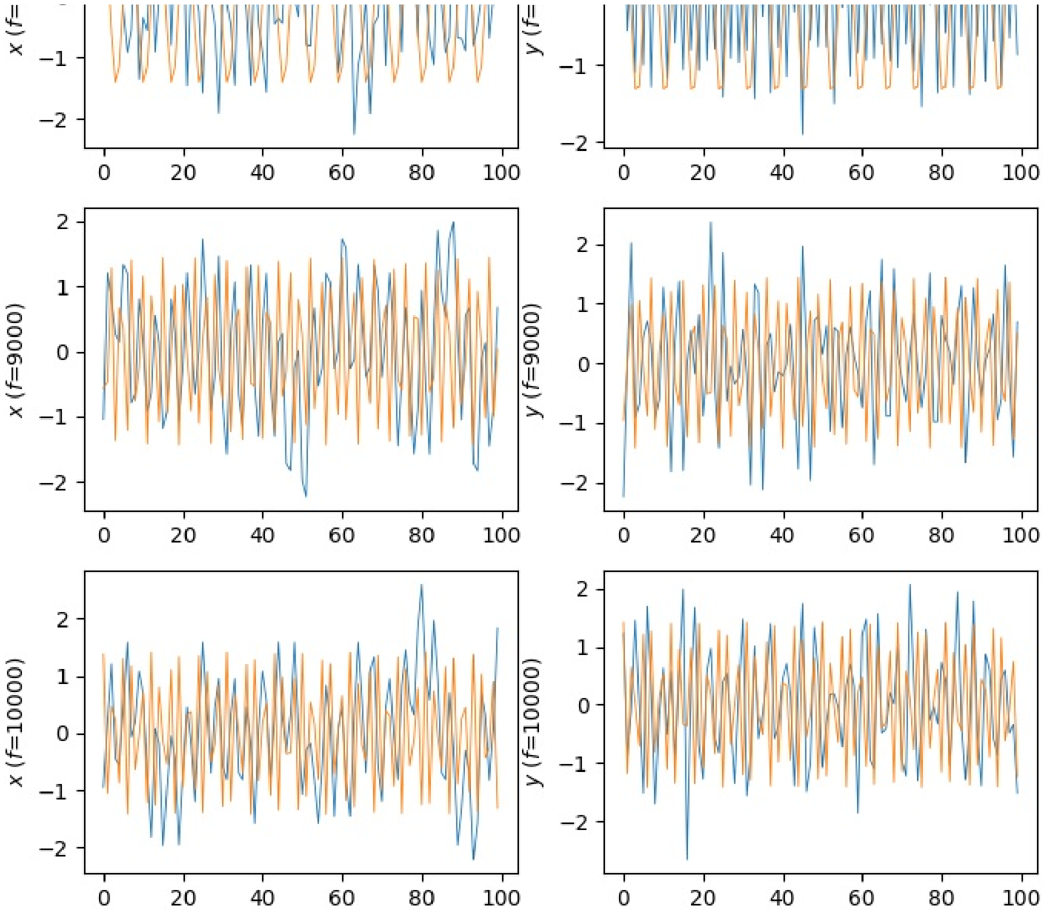
Loss for model

Plotting normalized signals.

```
X idx: 25.0000 mean, 17.5357 std
Y idx: 15.2500 mean, 9.2432 std
X coeffs: 0.0720 mean, 0.0817 std
Y coeffs: 0.0795 mean, 0.0590 std
X MSEs: 1.8373 mean, 0.1778 std
Y MSEs: 1.8084 mean, 0.1515 std
Loss: 3.6457
```

Stable-state XY-data plots for given frequencies


x (f=5000)
y (f=5000)
x (f=6500)
y (f=6500)
x (f=7000)
y (f=7000)
x (f=7500)
y (f=7500)
x (f=8000)
y (f=8000)
8500)
8500)
0
20
40
60
80
100


x (f=
y (f=
−1
−2
0
20
40
60
80
100
x (f=9000)
y (f=9000)
2
1
0
−1
−2
x (f=10000)
y (f=10000)