population_2Osc_nonlinear_stim

November 14, 2025

1 Two Nonlinear Two-Population Oscillators with Periodic Perturbation

1.1 Model

The two populations' temporal dynamics (change of state) is described by two first-order nonlinear differential equations. Here we have two coupled units:

$$\frac{dEx_1}{dt} = (h_{ex} - Ex_1 - c_2 * sigmoid(In_1) + c_{EE} * sigmoid((Ex_1 + frac_E * Ex_2)) + pert) * \tau_{ex}$$
$$\frac{dIn_1}{dt} = (h_{in} - In_1 - c_4 * sigmoid(In_1) + c_{EI} * sigmoid((Ex_1 + frac_I * Ex_2))) * \tau_{in}$$

$$\frac{dEx_2}{dt} = (h_{ex} - Ex_2 - c_2 * sigmoid(In_2) + c_{EE} * sigmoid((Ex_2 + frac_E * Ex_1))) * \tau_{ex}$$
$$\frac{dIn_2}{dt} = (h_{in} - In_2 - c_4 * sigmoid(In_2) + c_{EI} * sigmoid((Ex_2 + frac_I * Ex_1))) * \tau_{in}$$

Ex_i and In_i are variables that change with time t and represent the excitatory and inhibitory population, respectively.

h_{ex} and h_{in} are constant inputs to the populations.

c_2 and c_4 are model parameters representing internal inhibitory coupling.

τ_{ex} and τ_{in} are the poulation time constants.

c_{EE} and c_{EI} are self- and between population excitatory coupling parameters.

pert represents external perturbation.

Function *sigmoid* is a continuous step function, conmputationally implemented as tangens hyperbolicus.

[1]:
```
from scipy.integrate import odeint

from numpy import tanh, mod, linspace, around, zeros, mod, asarray
from numpy import sin, pi, var, ndarray, flip, arange
```

```
from numpy.random import default_rng, seed

from matplotlib.pyplot import subplots

import sk_dsp_comm.sigsys as ss

from scipy.signal import find_peaks

from itertools import product
```

to install `pip install scikit-dsp-comm`

[2]:
```
def sigmoid(u):
    return tanh(u)

def oscillator(y, t, N, h_ex, h_in, pars, frac_E, frac_I, sr, time_stop, pert):

    tau_ex, tau_in, c2, c4, c_EE, c_EI = pars

    if len(pert) == 0:

        dydt = (


            (h_ex - y[0] - c2*sigmoid(y[1]) +␣
 ↪c_EE*sigmoid(y[0]+frac_E*y[2]))*tau_ex,

            (h_in - y[1] - c4*sigmoid(y[1]) +␣
 ↪c_EI*sigmoid(y[0]+frac_I*y[2]))*tau_in,


            (h_ex - y[2] - c2*sigmoid(y[3]) +␣
 ↪c_EE*sigmoid(y[2]+frac_E*y[0]))*tau_ex,

            (h_in - y[3] - c4*sigmoid(y[3]) +␣
 ↪c_EI*sigmoid(y[2]+frac_I*y[0]))*tau_in
          )

        return dydt


    else:

        time_index = int(t*sr)

        if time_index >= time_stop*sr:
```

```
            dydt = zeros(2*N)

            return dydt


        else:

            dydt = (

                (pert[time_index] - y[0] - c2*sigmoid(y[1]) +␣
↪c_EE*sigmoid(y[0]+frac_E*y[2]))*tau_ex,

                (h_in             - y[1] - c4*sigmoid(y[1]) +␣
↪c_EI*sigmoid(y[0]+frac_I*y[2]))*tau_in,


                (h_ex             - y[2] - c2*sigmoid(y[3]) +␣
↪c_EE*sigmoid(y[2]+frac_E*y[0]))*tau_ex,

                (h_in             - y[3] - c4*sigmoid(y[3]) +␣
↪c_EI*sigmoid(y[2]+frac_I*y[0]))*tau_in
               )

            return dydt


def plot_bifdiagram_scan(results, freq_set):

    if freq_set[-1] < freq_set[0]:

        freq_set = flip(freq_set)

    freq_min, freq_max = freq_set[0], freq_set[-1]

    fig, ax  = subplots(nrows=1, figsize=(5, 4))

    for result in results:

        results_min_f, results_max_f = result[0], result[1]

        for xe, ye in zip(freq_set, results_max_f[0]):

            if not isinstance(ye, ndarray):
                ax.scatter(xe, ye, c='k', s=5)
            else:
                ax.scatter([xe] * len(ye), ye, c='r', s=5, marker='o')
```

```
        for xe, ye in zip(freq_set, results_min_f[0]):

            if not isinstance(ye, ndarray):
                ax.scatter(xe, ye, c='k', s=5)
            else:
                ax.scatter([xe] * len(ye), ye, c='b', s=5, marker='x')

        ax.set_xticks(linspace(freq_min, freq_max, 5));
        ax.set_xticklabels([]);

        ax.set_ylabel('Ex', fontsize=14)

        y_min, y_max = ax.get_ylim()

        # y_min, y_max = 0.5, 1.5

        ax.set_ylim(y_min, y_max)
        ax.set_yticks(linspace(y_min, y_max, 3))
        ax.set_yticklabels(around(linspace(y_min, y_max, 3)), fontsize=12)

    ax.set_xticks(linspace(freq_min, freq_max, 5));
    ax.set_xticklabels(around(linspace(freq_min, freq_max, 5), 2), fontsize=12);

    ax.set_xlabel('Parameter', fontsize=16)

    fig.tight_layout()

    return fig, ax


def plot_bifdiagram(results_min_f, results_max_f,
                    par_set):

    N = len(results_min_f)

    fig, ax = subplots(figsize=(6, 4))

    for xe, ye in zip(par_set, results_max_f[0]):

        if not isinstance(ye, ndarray):
            ax.scatter(xe, ye, c='r', s=6, marker='D')
        else:
            ax.scatter([xe] * len(ye), ye, s=3, c='k', marker='D')

    for xe, ye in zip(par_set, results_min_f[0]):

        if not isinstance(ye, ndarray):
```

```
            ax.scatter(xe, ye, c='r', s=6, marker='d')
        else:
            ax.scatter([xe] * len(ye), ye, s=3, c='gray', marker='d')


    ax.set_xticks(linspace(par_set[0], par_set[-1], 5));
    ax.set_xticklabels(around(linspace(par_set[0], par_set[-1], 5), 2),␣
↪fontsize=16);
    ax.set_xlabel('Parameter', fontsize=16)

    ax.set_ylabel('EX', fontsize=14)

    y_min, y_max = ax.get_ylim()

    ax.set_yticks(linspace(y_min, y_max, 3));
    ax.set_yticklabels(around(linspace(y_min, y_max, 3),2), fontsize=14);

    fig.tight_layout()

    return fig, ax
```

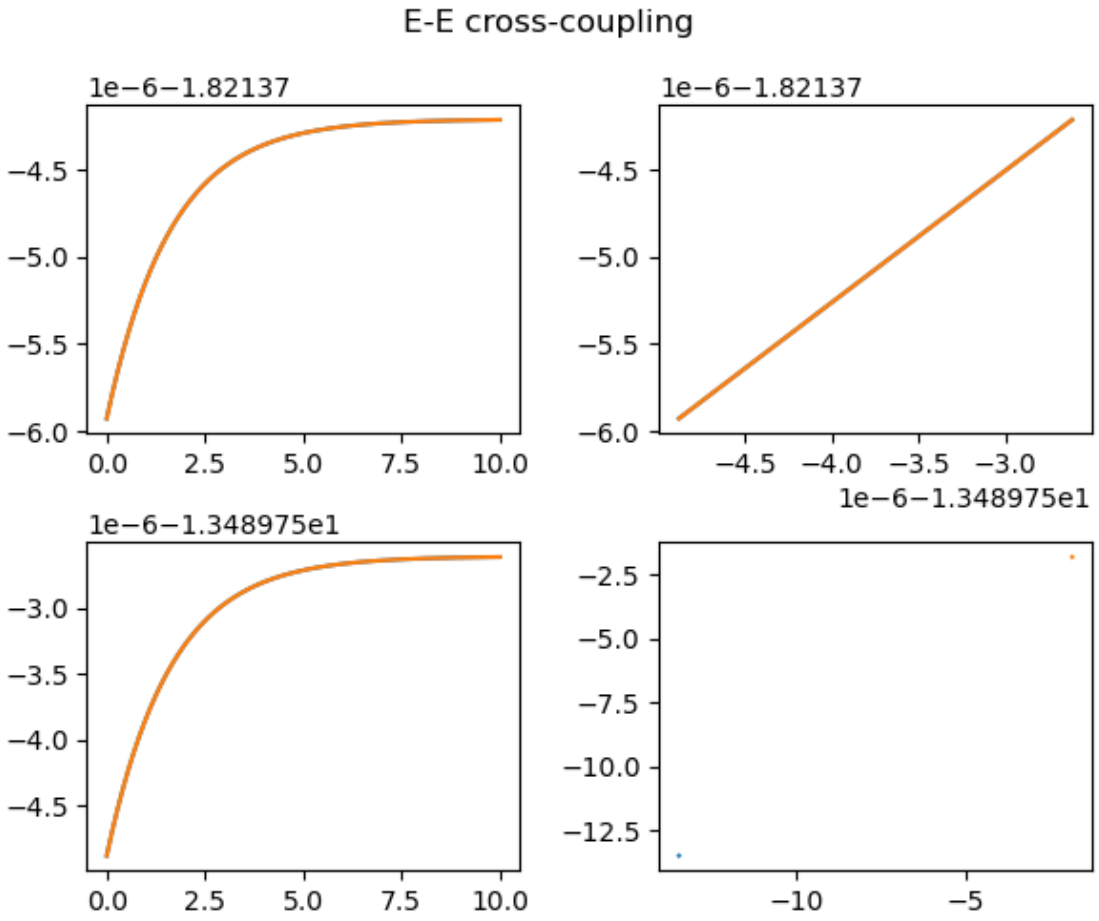
2 Time Series

[49]:
```
N = 2

# Input parameter
h_ex_0 = -7.
h_in_0 = -4.0

# Oscillator parameters
pars = (1, 2.5, 10, 0, 5, 10) # Homoclinic
# pars = (1, 1.5, 10, 0, 5, 10) # SNIC
# t_ex, t_in, c2, c4, cc_EE, cc_EI

# Strong synchronisation
# frac_E, frac_I = 0.5, 0.0 # < 1. 0 means no coupling

# Temporary desynchronisation
# frac_E, frac_I = 0.0, 0.5 # < 1. 0 means no coupling

frac_E, frac_I = 0.1, 0.0 # < 1. 0 means no coupling

# Initial conditions
SEED = 12345
# rng  = default_rng(SEED)
# y_ini = rng.uniform(size=2*N)
```

```
# Time array
time_stop = 10
sr        = 1000
time      = linspace(start=0, stop=time_stop, num=time_stop*sr)

pulse = []

# Simulation
y = odeint(func=oscillator, y0=y_ini, t=time,
          args=(N, h_ex_0, h_in_0, pars, frac_E, frac_I, sr, time_stop, pulse),
          hmax=0.1)

# Resetting of initial conditions
y_ini = y[-1, :]

# Show final values of all variables
print('End of run:', list(around(y[-1,:],3)))
print('')

fig, ax = subplots(ncols=2, nrows=2, figsize=(6, 5))

ax[0, 0].plot(time, y[:, [0, 2]]);
ax[0, 1].plot(y[:, [1, 3]], y[:, [0, 2]]);
ax[1, 0].plot(time, y[:, [1, 3]]);
ax[1, 1].plot(y[:, [1, 0]], y[:, [3, 2]]);

if frac_E > 0 and frac_I == 0:

    chars = 'E-E cross-coupling'
    fig.suptitle(chars)

elif frac_E == 0 and frac_I > 0:

    chars = 'E-I cross-coupling'
    fig.suptitle(chars)

elif frac_E > 0 and frac_I > 0:

    chars = 'E-E and E-I cross-coupling'
    fig.suptitle(chars)

else:

    chars = 'No coupling'
    fig.suptitle(chars)
```

```
# elif pars[3] == 10 and pars[6] > 0:

#     chars = 'Fast with E-E coupling'
#     fig.suptitle(chars)

# elif pars[3] == 10 and pars[7] > 0:

#     chars = 'Fast with E-I coupling'
#     fig.suptitle(chars)

fig.tight_layout()
```

```
End of run: [-1.821, -13.49, -1.821, -13.49]
```




```
[ ]:
```

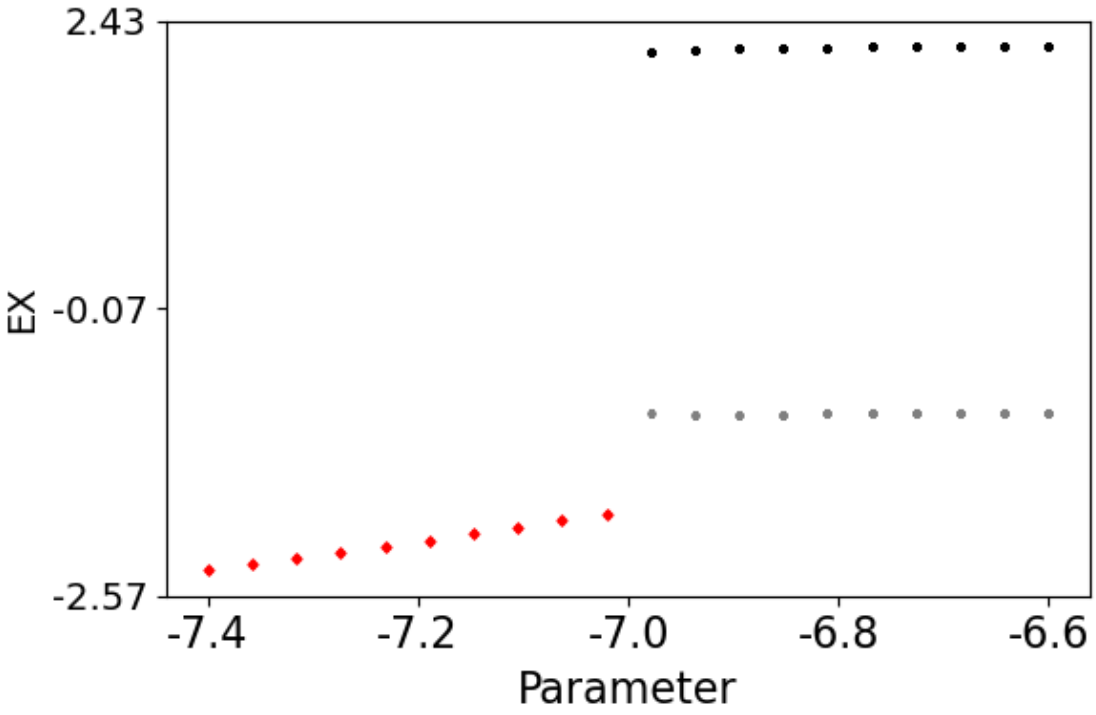
3 Bifurcation Diagram with Coupling

[45]:
```
# Bifurcation parameter set
steps = 20

par_min, par_max = -6.6, -7.4

par_set = linspace(par_min, par_max, steps)

# Time array
time_stop = 500
time = linspace(start=0, stop=time_stop, num=time_stop*sr)

results_min_f      = dict()
results_min_inds_f = dict()
results_max_f      = dict()
results_max_inds_f = dict()

rows = time.size

pulse = []

# Simulation "forward"
for par in par_set:

    y_pert = odeint(func=oscillator, y0=y_ini, t=time,
                    args=(N, par, h_in_0, pars, frac_E, frac_I, sr, time_stop,␣
 ↪pulse),
                    hmax=0.1)

    for num, series in enumerate(y_pert[rows//2:,:-1:2].T):

        if var(series) < 0.0001:

            if num not in results_min_f:

                results_min_f[num]      = [series[-1]]
                results_min_inds_f[num] = [0]

            else:
                results_min_f[num].append(series[-1])
                results_min_inds_f[num].append(0)

            if num not in results_max_f:

                results_max_f[num]      = [series[-1]]
                results_max_inds_f[num] = [0]
```

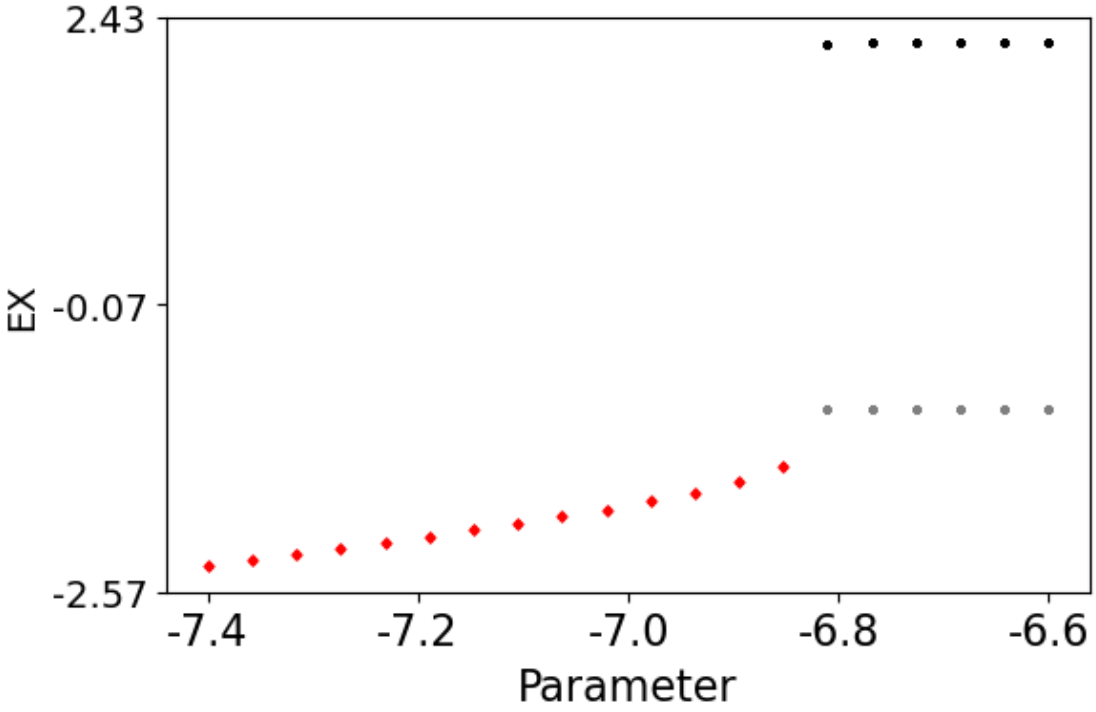
```
            else:
                results_max_f[num].append(series[-1])
                results_max_inds_f[num].append(0)

        else:

            y_f_max_inds = find_peaks(series, distance=100)
            y_f_maxs     = series[y_f_max_inds[0]]

            y_f_min_inds = find_peaks(-series, distance=100)
            y_f_mins     = series[y_f_min_inds[0]]

            if num not in results_min_f:

                results_min_f[num]      = [y_f_mins]
                results_min_inds_f[num] = [y_f_min_inds]

                results_max_f[num]      = [y_f_maxs]
                results_max_inds_f[num] = [y_f_max_inds]

            else:

                results_min_f[num].append(y_f_mins)
                results_min_inds_f[num].append(y_f_min_inds)

                results_max_f[num].append(y_f_maxs)
                results_max_inds_f[num].append(y_f_max_inds)


    if par != par_set[-1]:

        y_ini = y_pert[-1, :]

print('')
print('Scan complete!', list(around(y_pert[-1,:],3)))
print('')
```

```
Scan complete! [-2.343, -13.817, -2.343, -13.817]
```

[46]:
```
fig, ax = plot_bifdiagram(results_min_f, results_max_f, par_set)
```



[44]:
```
fig, ax = plot_bifdiagram(results_min_f, results_max_f, par_set)
```



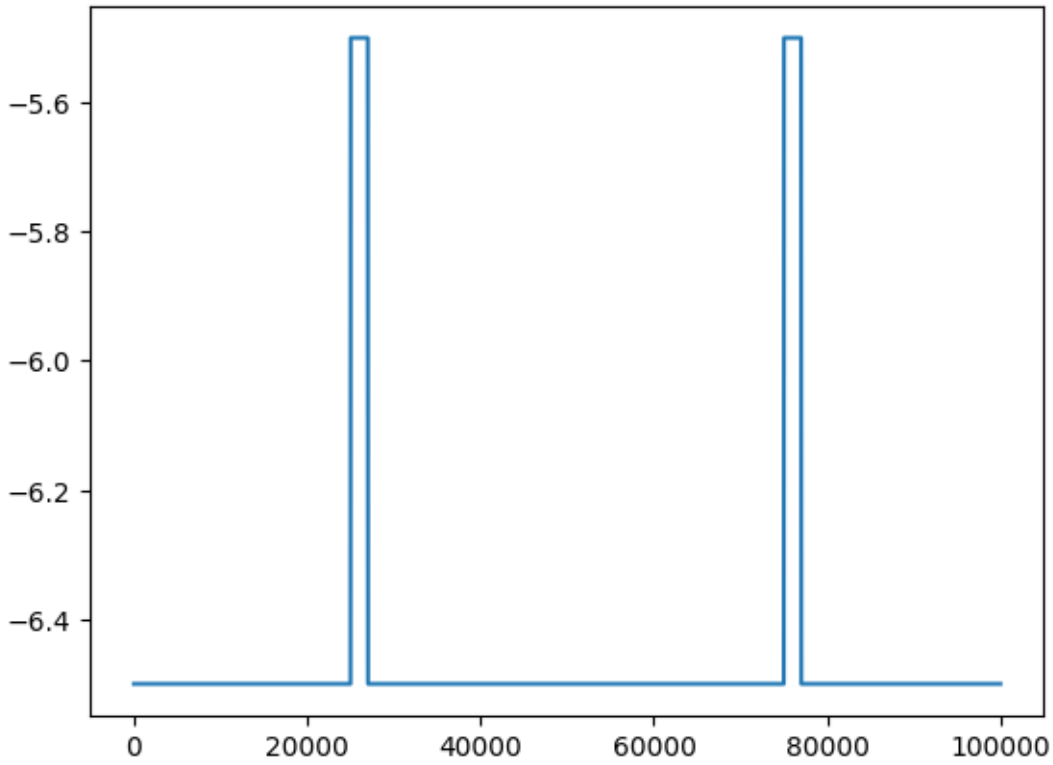
[ ]:

4 Periodic Pulse Perturbation

[20]:
```
# h_ex_0 = -6.5

# frac_E, frac_I = 0.5, 0.0 # < 1. 0 means no coupling

pulse_wid = 2.
pulse_amp = 1.
pulse_per = 50

# Initial conditions
SEED = 12345
rng  = default_rng(SEED)
y_in = rng.uniform(size=2*N)

# Time array
time_stop = 100
sr        = 1000
time      = linspace(start=0, stop=time_stop, num=time_stop*sr)

rows = time.size

pert = h_ex_0 + pulse_amp*ss.rect(mod(time, pulse_per)-(pulse_per)/2-pulse_wid/
 ↪2, pulse_wid)

# Simulation
y_pert = odeint(func=oscillator, y0=y_in, t=time,
              args=(N, h_ex_0, h_in_0, pars, frac_E, frac_I, sr, time_stop,␣
 ↪pert),
              hmax=0.1)

# Resetting of initial conditions
y_in = y_pert[-1, :]

# Show final values of all variables
print('End of run:', list(around(y_pert[-1,:],2)))
print('')
```

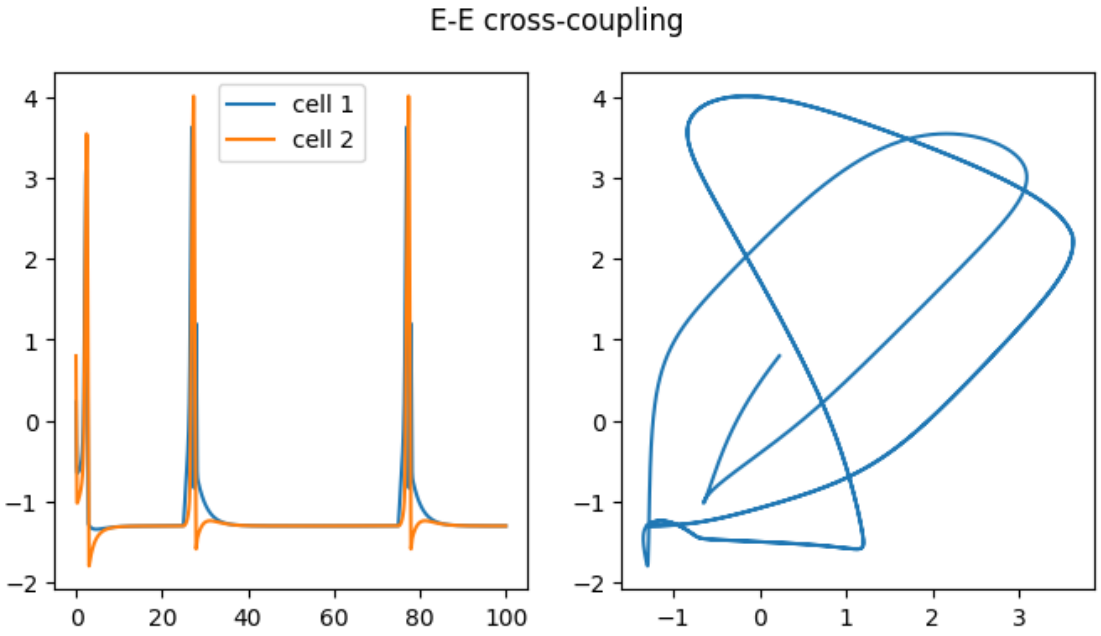
```
End of run: [np.float64(-1.3), np.float64(-12.63), np.float64(-1.3),
np.float64(-12.63)]
```

[22]:
```
fig, ax = subplots()

ax.plot(pert);
```



[24]:
```
fig, ax = subplots(ncols=2, nrows=1, figsize=(8, 4))

ax[0].plot(time, y_pert[:, 0], label="cell 1");
ax[0].plot(time, y_pert[:, 2], label="cell 2");
ax[0].legend()
ax[1].plot(y_pert[:, 0], y_pert[:, 2]);


if frac_E > 0 and frac_I == 0:

    chars = 'E-E cross-coupling'
    fig.suptitle(chars)

elif frac_E == 0 and frac_I > 0:

    chars = 'E-I cross-coupling'
    fig.suptitle(chars)
```

```
elif frac_E > 0 and frac_I > 0:

    chars = 'E-E and E-I cross-coupling'
    fig.suptitle(chars)

else:

    chars = 'No coupling'
    fig.suptitle(chars)
```



[ ]:

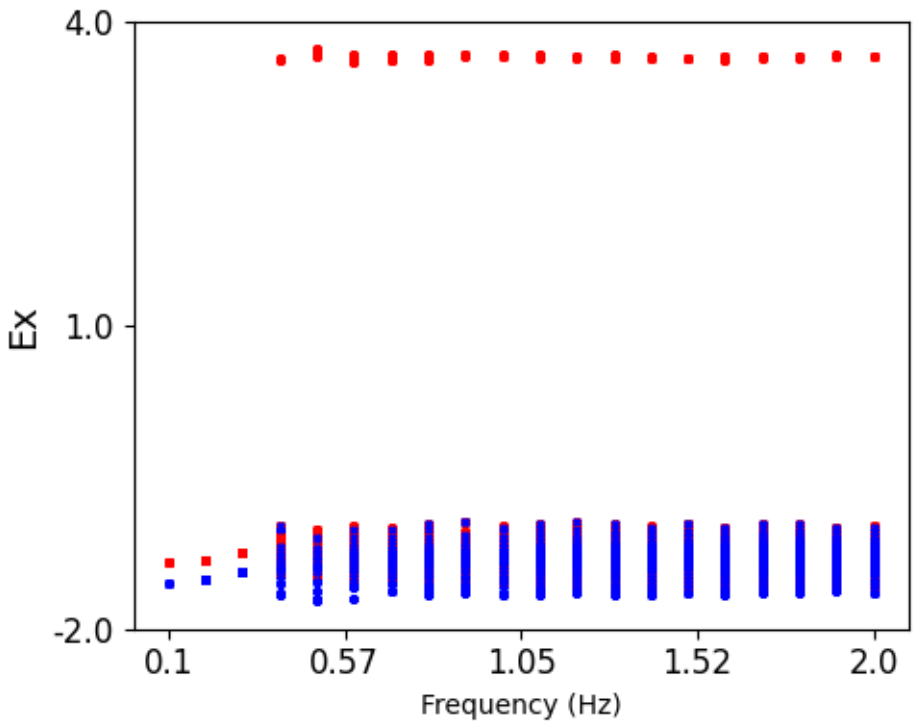
[26]:
```
# Bifurcation parameter set
steps = 20
freq_min, freq_max = 0.1, 2
freq_set = linspace(freq_min, freq_max, steps)

# Time array
time_stop = 100
sr        = 1000
time      = linspace(start=0, stop=time_stop, num=time_stop*sr)

rows = time.size

pulse_wid = 1.
```

```
pulse_amp = 0.25

results = list()


results_min_f      = dict()
results_min_inds_f = dict()
results_max_f      = dict()
results_max_inds_f = dict()

# Simulation "forward"
for freq in freq_set:

    pert = h_ex_0 + pulse_amp*ss.rect(mod(time, 1/freq)-(1/freq)/2-pulse_wid/2,␣
 ↪pulse_wid)

    y_pert = odeint(func=oscillator, y0=y_in, t=time,
                    args=(N, h_ex_0, h_in_0, pars, frac_E, frac_I, sr,␣
 ↪time_stop, pert),
                    hmax=0.1)

    for num, series in enumerate(y_pert[rows//2:,::2].T):

        if var(series) < 0.0001:

            if num not in results_min_f:

                results_min_f[num]      = [series[-1]]
                results_min_inds_f[num] = [0]

            else:
                results_min_f[num].append(series[-1])
                results_min_inds_f[num].append(0)

            if num not in results_max_f:

                results_max_f[num]      = [series[-1]]
                results_max_inds_f[num] = [0]

            else:
                results_max_f[num].append(series[-1])
                results_max_inds_f[num].append(0)

        else:

            y_f_max_inds = find_peaks(series, distance=100)
            y_f_maxs     = series[y_f_max_inds[0]]
```

```
            y_f_min_inds = find_peaks(-series, distance=100)
            y_f_mins     = series[y_f_min_inds[0]]

            if num not in results_min_f:

                results_min_f[num]      = [y_f_mins]
                results_min_inds_f[num] = [y_f_min_inds]

                results_max_f[num]      = [y_f_maxs]
                results_max_inds_f[num] = [y_f_max_inds]

            else:

                results_min_f[num].append(y_f_mins)
                results_min_inds_f[num].append(y_f_min_inds)

                results_max_f[num].append(y_f_maxs)
                results_max_inds_f[num].append(y_f_max_inds)

        # if par != par_set[-1]:

        #     y_in = y_pert[-1, :]

    results.append((results_min_f, results_max_f))

print('')
print('Scan complete!')
print('')
```

```
Scan complete!
```

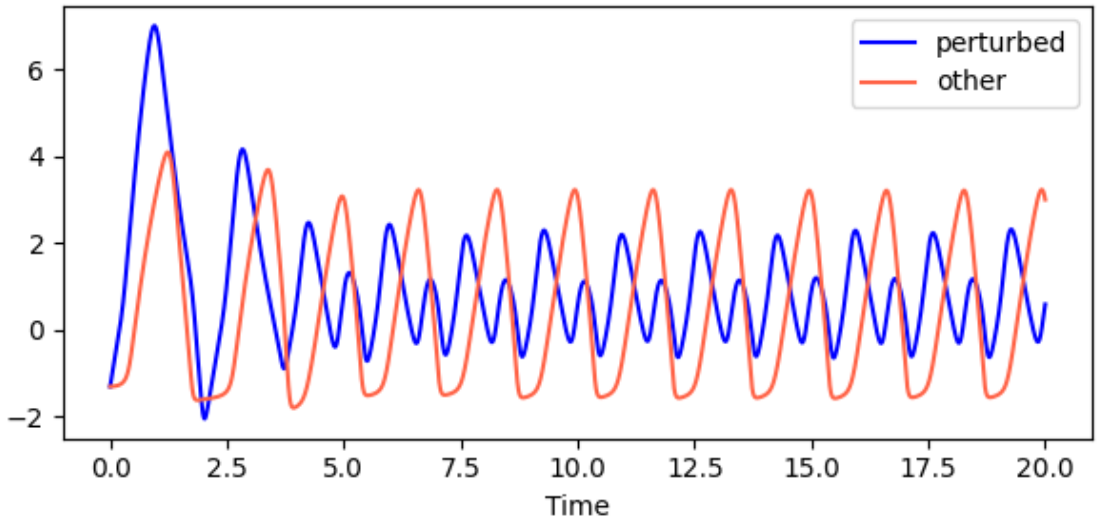
[28]:
```
# Plot
fig, ax = plot_bifdiagram_scan(results, freq_set)


ax.set_xlabel('Frequency (Hz)', fontsize=10);


fig.tight_layout()
```



[ ]:

[30]:
```
# Time array
time_stop = 20
sr        = 1000
time      = linspace(start=0, stop=time_stop, num=time_stop*sr)

rows = time.size

freq       = 0.5

pulse      = pulse_amp*ss.rect(mod(time, 1/freq)-(1/freq)/2-pulse_wid/2,␣
 ↪pulse_wid)

fig, ax = subplots(figsize=(6, 3))

y_pert = odeint(func=oscillator, y0=y_in, t=time,
                args=(N, h_ex_0, h_in_0, pars, frac_E, frac_I, sr, time_stop,␣
 ↪pulse),
                hmax=0.1)

ax.plot(time, y_pert[:, 0], c='b',      label='perturbed')
ax.plot(time, y_pert[:, 2], c='tomato', label='other')
```

```
ax.legend()

ax.set_xlabel('Time');

fig.tight_layout()
```



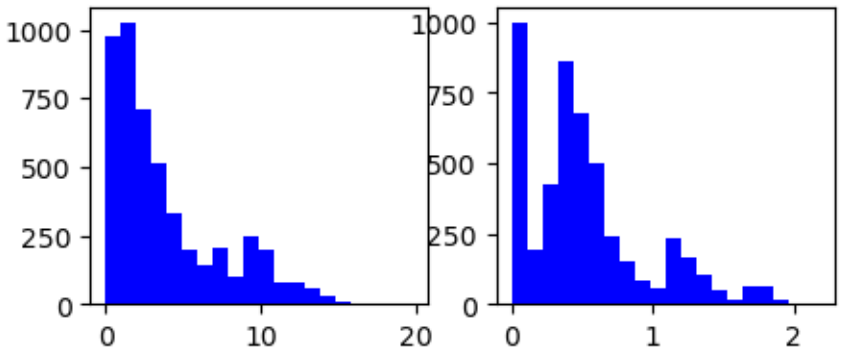
[ ]:

5 Pulse Perturbation 2D Parameter Scan

[50]:
```
# Initial conditions y_last
y_last = y_ini.copy()

time_stop_pert = 200
sr             = 1000
time           = linspace(start=0, stop=time_stop_pert, num=time_stop_pert*sr)

resolution = 70

pulse_width_min, pulse_width_max = 0.1, 4.1
pulse_ampli_min, pulse_ampli_max = 4, 0.0001

pulse_width = linspace(pulse_width_min, pulse_width_max, resolution)
pulse_ampli = linspace(pulse_ampli_min, pulse_ampli_max, resolution)

pulse_period = 200

results_pert = list()
```

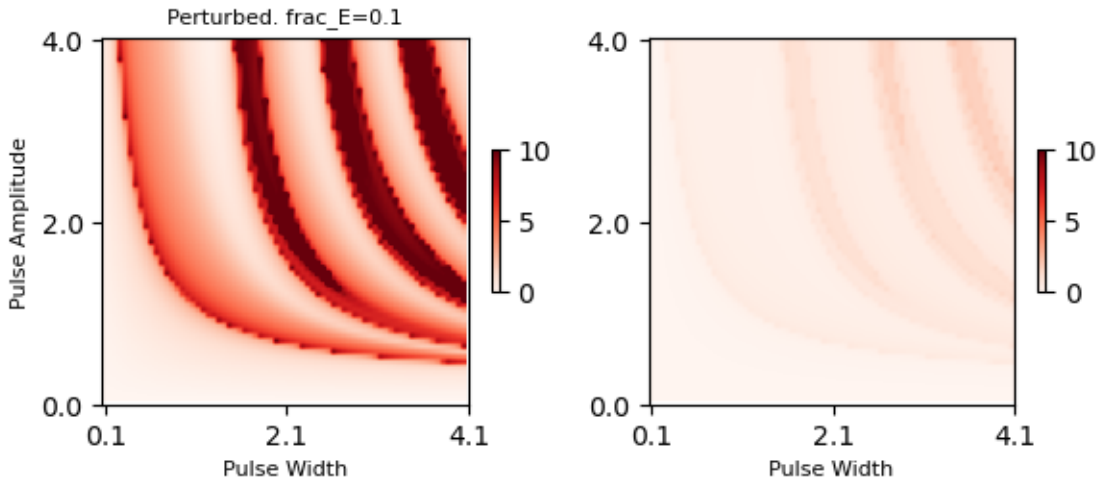
```
for pulse in product(pulse_width, pulse_ampli):

    h_ex_p = h_ex_0 + pulse[1]*ss.rect(mod(time, pulse_period)-pulse_period/
 ↪2+pulse[0]/2, pulse[0])

    y = odeint(func=oscillator, y0=y_last, t=time, args=(N, h_ex_0, h_in_0,␣
 ↪pars, frac_E, frac_I, sr, time_stop_pert, h_ex_p),
               hmax=0.1)


    results_pert.append((sum(y[time_stop_pert*sr//2:, 0] - y_last[0])/sr,
                         sum(y[time_stop_pert*sr//2:, 2] - y_last[2])/sr
                     ))

results_pert_ex1_matrix = asarray(results_pert)[:, 0].reshape(resolution,␣
 ↪resolution)
results_pert_ex2_matrix = asarray(results_pert)[:, 1].reshape(resolution,␣
 ↪resolution)

print('COMPLETE')
print(' ')

fig, ax = subplots(ncols=2, figsize=(5, 2))

ax[0].hist(results_pert_ex1_matrix.ravel(), bins=20, color='b');

ax[1].hist(results_pert_ex2_matrix.ravel(), bins=20, color='b');
```

```
COMPLETE
```



[56]:
```
vmin, vmax = 0, 10

fig, ax = subplots(figsize=(6, 4), ncols=2)

im1 = ax[0].imshow(results_pert_ex1_matrix.T, cmap='Reds', vmin=vmin, vmax=vmax)

ax[0].set_xlabel('Pulse Width', fontsize=8)
ax[0].set_ylabel('Pulse Amplitude', fontsize=8)
ax[0].set_xticks(linspace(0, resolution, 3))
ax[0].set_xticklabels(around(linspace(pulse_width_min, pulse_width_max, 3),1))
ax[0].set_yticks(linspace(0, resolution, 3))
ax[0].set_yticklabels(around(linspace(pulse_ampli_min, pulse_ampli_max, 3), 2))
ax[0].set_title('Perturbed. frac_E=0.1', fontsize=8)


im2 = ax[1].imshow(results_pert_ex2_matrix.T, cmap='Reds', vmin=vmin, vmax=vmax)

ax[1].set_xlabel('Pulse Width', fontsize=8)
# ax[1].set_ylabel('Pulse Amplitude', fontsize=8)
ax[1].set_xticks(linspace(0, resolution, 3))
ax[1].set_xticklabels(around(linspace(pulse_width_min, pulse_width_max, 3), 1))
ax[1].set_yticks(linspace(0, resolution, 3))
ax[1].set_yticklabels(around(linspace(pulse_ampli_min, pulse_ampli_max, 3), 2))


fig.colorbar(im1, ax=ax[0], shrink=0.2);
fig.colorbar(im2, ax=ax[1], shrink=0.2);

fig.tight_layout()

if pars[1] == 1.5:

    title_chars = 'Figs/N=1/Fig_2Cb_PulsePert_SNIC.png'

elif pars[1] == 2.5:

    title_chars = 'PulsePert_Homoclinic_frac_E=0.1.png'

# fig.savefig(title_chars, dpi=300, format='png', bbox_inches='tight')

title_chars
```

[56]:
```
'PulsePert_Homoclinic_frac_E=0.1.png'
```


Perturbed. frac_E=0.1
Pulse Amplitude
4.0
2.0
0.0
0.1
2.1
4.1
Pulse Width
10
5
0
4.0
2.0
0.0
0.1
2.1
4.1
Pulse Width
10
5
0

[ ]: