04_calcium_2V_oscillations

November 17, 2025

1 Oscillator: Calcium-IP3 Oscillations

- Two variable set-up as X<->Y oscillator. X = cytosolic calcium; Y = IP3
- Steady state to oscillation transition as calcium supply is increased
- Result similar to S->P oscillation with forward and feedback inhibition
- Basis of a spatio-temporal reaction-diffusion model

[1]:
```python
from scipy.integrate import solve_ivp
from matplotlib.pyplot import subplots
from numpy import linspace, around, var, ndarray
from scipy.signal import find_peaks
```

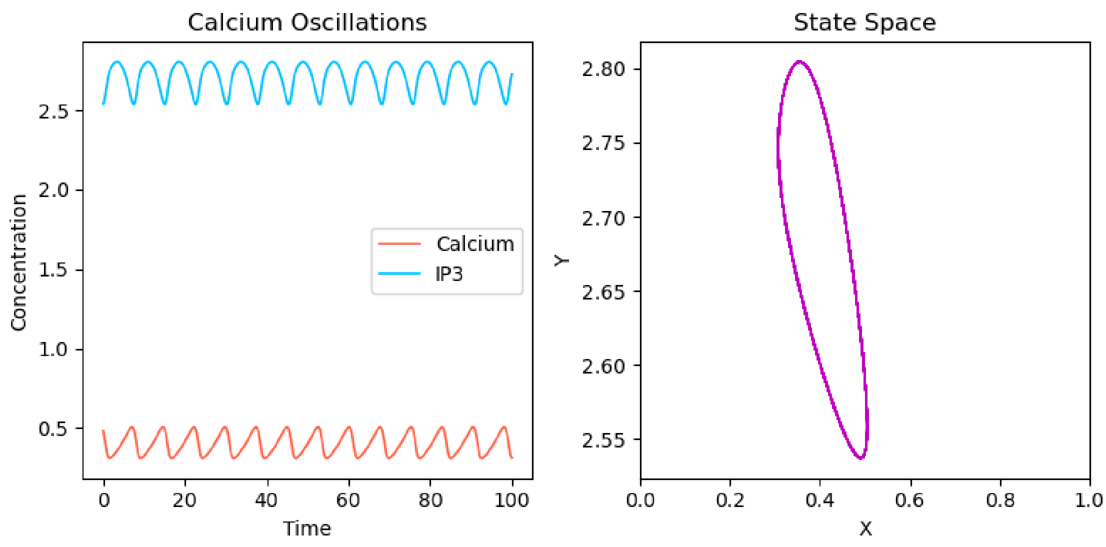
1.1 Model of Calcium - IP3 interaction

[2]:
```python
def model(t, variables, a, m2, m3, ka, k, k1):
    """Coupled system with feedback inhibition"""
    X, Y = variables

    dXdt = a - m2*X/(1+X) + (m3*Y/(k1+Y))*X**2/(ka+X**2) + Y - k*X
    dYdt =     m2*X/(1+X) - (m3*Y/(k1+Y))*X**2/(ka+X**2) - Y

    return [dXdt, dYdt]
```

1.2 Time Series

[3]:
```python
X_0 = 2.5
Y_0 = 1.0

y0 = [X_0, Y_0]
y0 = [0.48, 2.54]

a         = 0.32
m2, m3    = 20, 23
ka, k, k1 = 0.8, 0.8, 0.8

t_span = (0, 100)

```

```
solution = solve_ivp(model, t_span, y0, args=(a, m2, m3, ka, k, k1), method='BDF', max_step=0.1)

t = solution.t

X = solution.y[0]
Y = solution.y[1]

fig, ax = subplots(ncols=2, figsize=(8, 4))

ax[0].plot(t, X, label='Calcium', linewidth=1.2, color='tomato')
ax[0].plot(t, Y, label='IP3', linewidth=1.2, color='deepskyblue')

ax[0].set_xlabel('Time')
ax[0].set_ylabel('Concentration')
ax[0].legend()
ax[0].set_title('Calcium Oscillations')

ax[1].plot(X, Y, linewidth=1, color='m');
# ax[1].plot(c2[:300], c3[:300], linewidth=1, color='k');
ax[1].set_xlabel('X')
ax[1].set_ylabel('Y')
ax[1].set_title('State Space')
ax[1].set_xlim(0, 1)
# ax[1].set_ylim(4, 14)


fig.tight_layout()
```



[4]:
```
around((X[-1], Y[-1]), 2)
```

[4]: array([0.31, 2.73])

1.3 Bifurcation Diagram

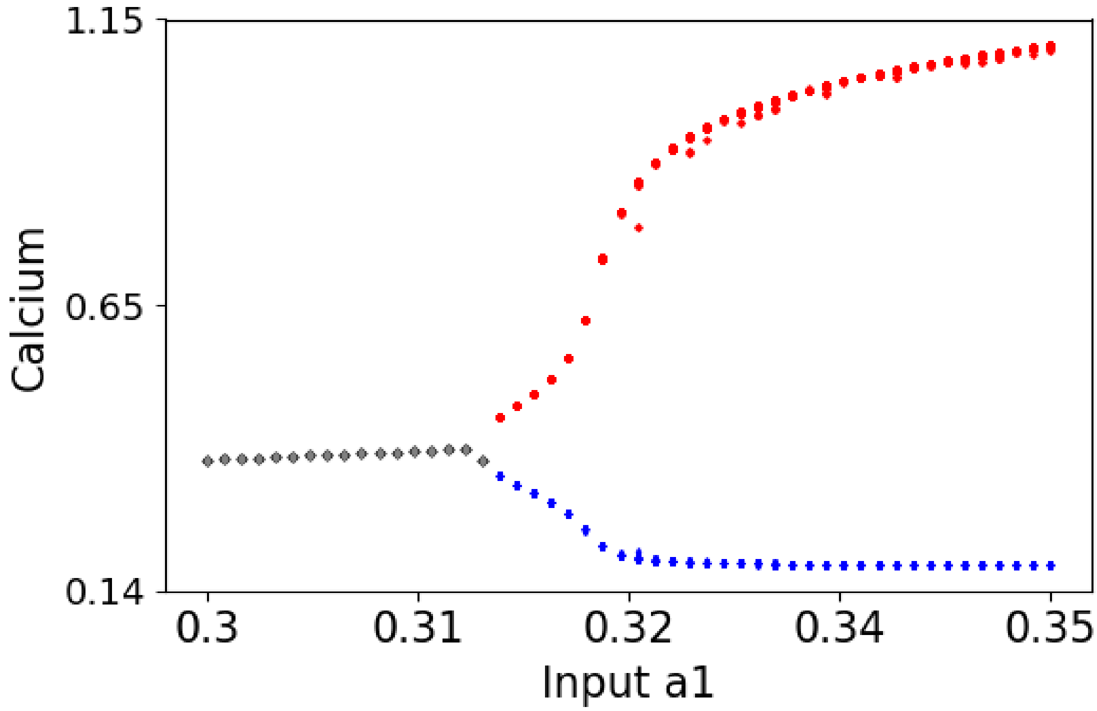
[8]:
```
# Bifurcation parameter set
par_min, par_max, steps = 0.3, 0.35, 50

par_set = linspace(par_min, par_max, steps)

# Time array
t_span = (0, 600)

y0 = (X[-1], Y[-1])

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

# Simulation "forward"
for par in par_set:

    solution = solve_ivp(model, t_span, y0, args=(par, m2, m3, ka, k, k1),␣
 ↪method='BDF', max_step=0.1)

    X = solution.y[0]
    Y = solution.y[1]

    rows = X.size//2

    series = X[rows//2:]

    num = 0

    if var(series) < 0.001:

        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)
```

```python
        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]:

        y0 = solution.y[:, -1]

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

```
Scan complete! [np.float64(0.522), np.float64(2.652)]
```

[9]:
```
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='k', s=6, marker='D')
        else:
            ax.scatter([xe] * len(ye), ye, s=3, c='r', marker='D')

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

        if not isinstance(ye, ndarray):
            ax.scatter(xe, ye, c='gray', s=6, marker='d')
        else:
            ax.scatter([xe] * len(ye), ye, s=3, c='b', 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
```

[10]:
```
fig, ax = plot_bifdiagram(results_min_f, results_max_f, par_set)

title_chars = 'Input a1'

ax.set_xlabel(title_chars, fontsize=16);
ax.set_ylabel('Calcium', fontsize=16);
```


1.15
0.65
0.14
Calcium
0.3
0.31
0.32
0.34
0.35
Input a1

```
[ ]:
```