Central Pattern Generator (CPG) document

Central Pattern Generator

The CPG model is a dynamic system used to generate a series of sine waves and make the phase difference, frequency and amplitude of these sine waves adjustable in real time. This article includes a CPG model written in python for ease of use. CPG enables the sine wave to produce a continuous transformation curve when switching amplitude, frequency, phase and offset. CPG can be described as

$$\dot{\varphi} = \omega + \mathbf{A} \cdot \varphi + \mathbf{B} \cdot \theta$$

$$\ddot{r} = a \cdot \left[\frac{a}{4}(R - r) - \dot{r}\right]$$

$$x = r \cdot sin(\varphi) + \delta$$

Where $\varphi \in \mathbb{R}^{n\times 1}$ and $r \in \mathbb{R}^{n\times 1}$ are internal states of CPG, n is the number of output sine wave channels.

$$\mathbf{A} = \begin{bmatrix} -\mu_1 & \mu_1 & & & \\ \mu_2 & -2\mu_2 & \mu_2 & & \\ & & \ddots & & \\ & \mu_{n-1} & & -2\mu_{n-1} & \mu_{n-1} \\ & & & \mu_n & -\mu_n \end{bmatrix} \in \mathbb{R}^{n\times n}$$

$$\mathbf{B} = \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ & -1 & \ddots & & \\ & & \ddots & 1 & \\ & & & -1 & 1 \\ & & & & -1 \end{bmatrix} \in \mathbb{R}^{n\times (n-1)}$$

μ_i are parameters to control the rate of convergence. $R \in \mathbb{R}^{n\times 1}$, $\theta \in \mathbb{R}^{(n-1)\times 1}$, $\omega \in \mathbb{R}^{n\times 1}$ and $\delta \in \mathbb{R}^{n\times 1}$ are inputs of CPG. $R \in \mathbb{R}^{n\times 1}$ controls the amplitudes of the n channels, $\omega \in \mathbb{R}^{n\times 1}$ are the frequencies of the channels, $\delta \in \mathbb{R}^{n\times 1}$ are the offsets of the channels and $\theta \in \mathbb{R}^{(n-1)\times 1}$ are the phase difference of the channels. $\theta_i = \varphi_i - \varphi_{i+1}$. For example, setting

$$R = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \omega = \begin{bmatrix} 3 \\ 4 \end{bmatrix}, \delta = \begin{bmatrix} 5 \\ 6 \end{bmatrix}, \theta = \begin{bmatrix} \frac{\pi}{2} \end{bmatrix}$$

will result two output channels by x, where

$$x_1 = 1 \cdot sin(3 \cdot t + \sigma) + 5$$

$$x_2 = 2 \cdot sin\left(4 \cdot t + \sigma - \frac{\pi}{2}\right) + 6$$

σ is determined by the initial value of φ.

Python Implementation

The class written in Python is CPG()

Arguments:

variables	meaning	Data type
n	Number of channels	Positive integer
R	Amplitudes $R \in \mathbb{R}^{n\times 1}$	Numpy array of size (n, 1)
omega	frequencies $\omega \in \mathbb{R}^{n\times 1}$	Numpy array of size (n, 1)
mu	Phase convergence rate	Numpy array of size (n, 1)
a	Amplitude convergence rate	Numpy array of size (n, 1)
dp	Phase shift between channels $\theta \in \mathbb{R}^{(n-1)\times 1}$	Numpy array of size (n-1, 1)
init_phi	Initial values of $\varphi \in \mathbb{R}^{n\times 1}$	Numpy array of size (n, 1)
offset	offsets $\delta \in \mathbb{R}^{n\times 1}$	Numpy array of size (n, 1)

Where typical choice of 'mu' is 0.5 and typical choice of 'a' is 0.1. Here shows an example of initialization of a CPG

```
n_joints = 5
cpg = CPG(n=n_joints,
          R=np.array([[1.0], [1.0], [1.0], [1.0], [1.0]]),
          omega=np.array([[0.1], [0.1], [0.1], [0.1], [0.1]]),
          mu=np.array([[0.5], [0.5], [0.5], [0.5], [0.5]]),
          a=np.array([[0.1], [0.1], [0.1], [0.1], [0.1]]),
          dp=np.array([[math.pi/5], [math.pi/5], [math.pi/5], [math.pi/5]]),
          init_phi=np.zeros((n_joints, 1)),
          offset=np.zeros((n_joints, 1)))
```

The member functions are

```
set_amp(R)
```

Function: set new amplitudes

Input:

R	Amplitudes $R \in \mathbb{R}^{n\times 1}$	Numpy array of size (n, 1)

Output: None

```
set_freq(omega)
```

Function: set new frequencies

Input:

omega	frequencies $\omega \in \mathbb{R}^{n\times 1}$	Numpy array of size (n, 1)

Output: None

set_phase_shift(dp)

Function: set new phase shifts

Input:

omega	frequencies $\omega \in \mathbb{R}^{n\times 1}$	Numpy array of size (n, 1)

Output: None

update()

Function: evolve the internal states to the next time step. For example you are controlling a system with 50Hz, you have to call this function at each step.

Input: None

Output: None

reset(init_phi)

Function: reset the CPG to initial state

Input:

init_phi	Initial values of $\varphi \in \mathbb{R}^{n\times 1}$	Numpy array of size (n, 1)

Output: None

```
output()
```

Function: output sine waves

Input: None

Output:

x	Output sine waves	Numpy array of size (n, 1)

A typical use of CPG is

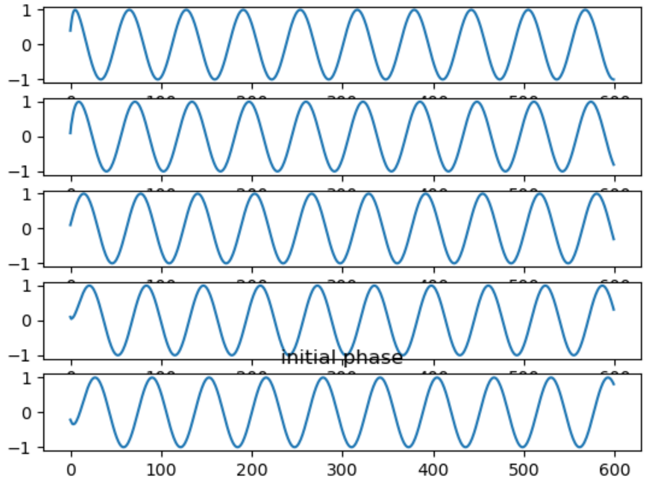
```
n_joints = 5
cpg = CPG(n=n_joints,
          R=np.array([[1.0], [1.0], [1.0], [1.0], [1.0]]),
          omega=np.array([[0.1], [0.1], [0.1], [0.1], [0.1]]),
          mu=np.array([[0.5], [0.5], [0.5], [0.5], [0.5]]),
          a=np.array([[0.1], [0.1], [0.1], [0.1], [0.1]]),
          dp=np.array([[math.pi/5], [math.pi/5], [math.pi/5], [math.pi/5]]),
          init_phi=np.zeros((n_joints, 1)),
          offset=np.zeros((n_joints, 1)))

for i in range(200):
        cpg.update()
        x = cpg.output()
```

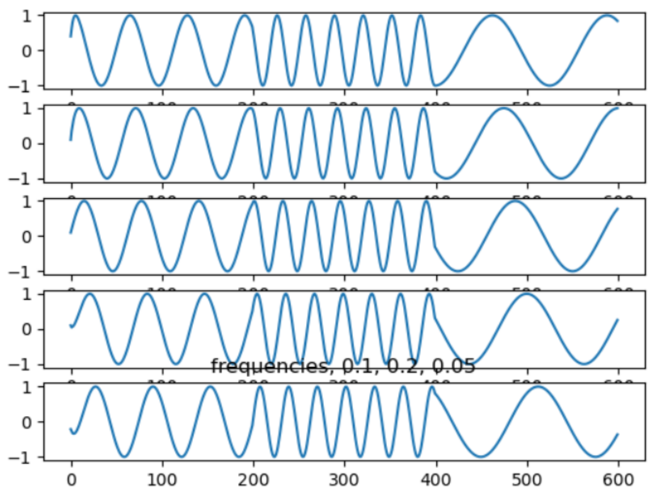
Then you can use cpg to control external systems

By setting the parameters of CPG, one can get different performances of sine waves

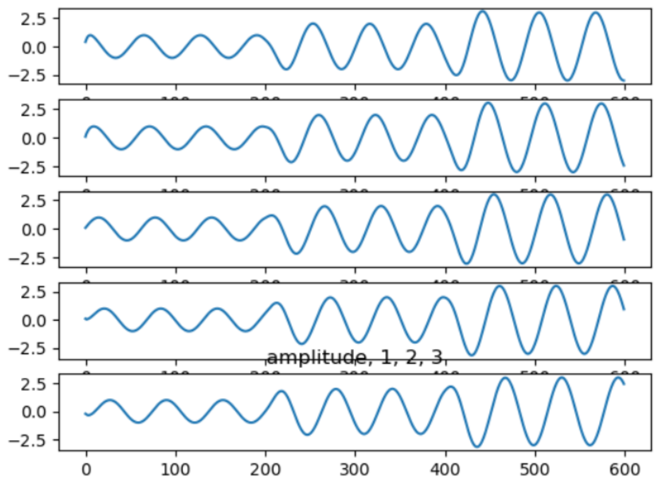
Phase shift by $\frac{\pi}{5}$



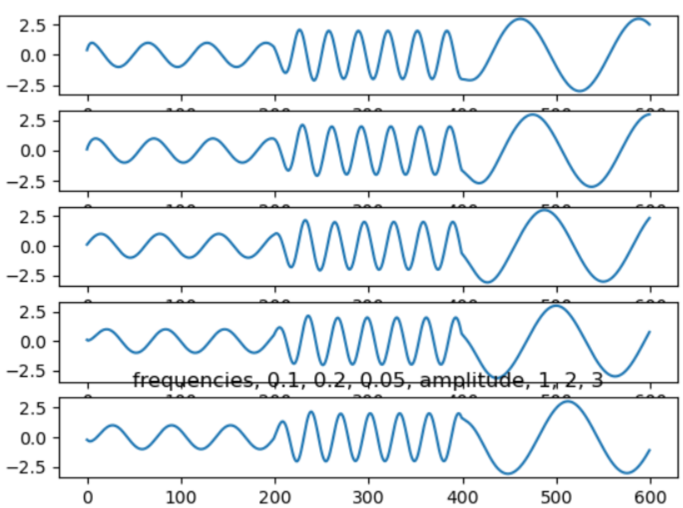
Frequency change



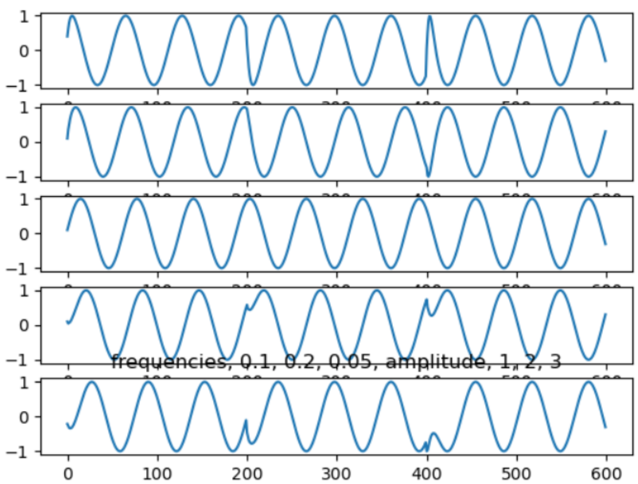
Amplitude change



Both frequency and amplitude change



Phase shift from $\frac{\pi}{5}$ to $\frac{\pi}{2}$ to π


frequencies, 0.1, 0.2, 0.05, amplitude, 1, 2, 3
0
100
200
300
400
500
600
1
0
−1