CEPAGE: a toolbox for Central Pattern Generator analysis

Matteo Lodi*, Andrey Shilnikov†, Marco Storace*
*DITEN, University of Genoa, Via Opera Pia 11a, I-16145 Genova, Italy
E-mail: matteo.lodi@edu.unige.it, marco.storace@unige.it
†Neuroscience Institute, Georgia State University, 100 Piedmont Ave, Atlanta GA 30303, USA
E-mail: ashilnikov@gsu.edu

***Abstract*—This paper is focused on a new object-oriented toolbox, called CEPAGE, devoted to simulation and analysis of Central Pattern Generators (CPGs). A CPG is a little group of neurons producing periodical patterns, which control rhythmic activities of animals. CEPAGE is conceived to carry out brute-force bifurcation analysis, but can also generate data for subsequent continuation analysis through other widely-used packages, such as AUTO or MATCONT. Two case studies are considered, with three and four neurons, with the twofold purpose of illustrating the main CEPAGE functionalities and provide new analysis results.**

I. Introduction

Among the many types of biological neural networks, Central Pattern Generators (CPGs) are little groups of neurons that produce rhythmic patterned outputs without sensory feedback or central input [1]–[3]. CPGs play many roles in animals, generating rhythms for locomotion, swimming, breathing, heart beating, swallowing, and other oscillatory functions. CPG models are widely used in robotics applications, to control a variety of different types of robots and different modes of locomotion mimicking the nature [3]. This is one of the main reasons why, in the last years, a growing attention has been devoted to the study of stable rhythmic patterns (or *motifs*) generated by this kind of networks [4].

A CPG model requires to specify the general architecture (type and number of oscillators/neurons), the type and topology of couplings and the waveforms to be generated. All these elements concur to determine the conditions for synchronization between oscillators and the resulting gaits, i.e., the stable phase relations between oscillators.

In this paper we propose a package (called CEPAGE[1]) for simulation and analysis of CPG models. CEPAGE has a two-layer organisation: the outer layer is a MATLAB interface that makes it easy the CPG configuration and offers tools for data analysis and visualization; the inner layer is used for numerical integrations and is based on Boost C++ libraries and on MEX files.[2] The MATLAB layer provides flexibility to CEPAGE, since it makes it easy to add new neuron and synapse models to be simulated and new functionalities to the package by extending the base classes. Moreover, MATLAB allows the user to write very concise and clear scripts, which nonetheless retain the full power and speed of the underlying C/C++ code. The main advantages of CEPAGE (besides the simplicity of usage provided by the MATLAB interface) can be then summarized as follows. *(i) Computational efficiency*. The inner part of CEPAGE is written in compiled languages (C and C++) and MEX files, then applications run much faster than equivalent codes in interpreted languages, such as Python or Matlab. Moreover, it is based on efficient and actively developed libraries, which can run also in parallel on multi-core units. *(ii) Flexibility*. The object-oriented programming nature of both the layers greatly facilitates reusability and development. Moreover, it is easy to vary parameters (of single neurons or of the synaptic connections) to obtain brute-force bifurcation diagrams. *(iii) Compatibility with other packages*. CEPAGE is intrinsically complementary to numerical continuation packages such as AUTO [5] and MATCONT [6] and can be used in combination with these packages.

These features are at least partially shared by other packages, of course, but CEPAGE is tailored for the analysis and simulation of CPGs. The only other package with similar characteristics is motifToolbox,[3] which runs under LINUX.

Two case studies are used to illustrate the CEPAGE functionalities. The first one is an already studied three-neuron CPG: in this case we obtain a new bifurcation diagram. The second case study is a four-neuron CPG not yet analyzed in literature, at the best of our knowledge, that can be viewed as a CPG controlling the gaits of a quadruped. We will show how the gaits change by varying a bifurcation parameter without altering the network structure: this corresponds to changing the locomotion rhythm, e.g., from walk, to trot, to gallop [7].

II. CPG description

The toolbox models a CPG composed by N neurons through the following dynamical system:

$$\dot{\mathbf{z}}_i = \begin{bmatrix} \dot{V}_i \\ \dot{\mathbf{x}}_i \end{bmatrix} = \begin{bmatrix} f_i(\mathbf{z}, I_{syn}^{(i)}) \\ \mathbf{p}_i(\mathbf{z}_i) \end{bmatrix} \tag{1}$$

[1]The French word "cépage" means "grape variety".

[2]A MEX file is a type of computer file that provides an interface between MATLAB and functions written in C, C++ or Fortran. It stands for "MATLAB executable". When compiled, MEX files are dynamically loaded and allow external functions to be invoked from within MATLAB as if they were built-in functions.

[3]Motiftoolbox is freely available at https://github.com/jusjusjus/Motiftoolbox.

978-1-4673-6853-7/17/$31.00 ©2017 IEEE

For the i-th neuron, V_i is the membrane voltage, $\mathbf{x}_i$ is a vector containing the other state variables (whose dynamics are described by the vector field $\mathbf{p}_i$) and $I_{syn}^{(i)}$ is the incoming synaptic current, containing the following contributions:

$$I_{syn}^{(i)} = \sum_{j=0}^{N-1} g_{i,j}^{in} h_{in}(V_i, V_j) + \sum_{j=0}^{N-1} g_{i,j}^{ex} h_{ex}(V_i, V_j) + \sum_{j=0}^{N-1} g_{i,j}^{el} h_{el}(V_i, V_j) \quad (2)$$

where h_{in}, h_{ex} and h_{el} describe chemical inhibitory, chemical excitatory and electrical synapses actions, respectively, whereas the coefficients $g_{i,j}^{in}$, $g_{i,j}^{ex}$ and $g_{i,j}^{el}$ represent chemical inhibitory, chemical excitatory and electrical synapses strengths, respectively, between neuron i and neuron j; $g_{i,j}^{xx} = 0$ means that neurons i and j are not connected by synapses of type xx. The synaptic actions are modeled as follows:

$$h_{el}(V_i, V_j) = V_j - V_i; \quad h_{in}(V_i, V_j) = \frac{E_{in} - V_i}{1 + e^{\nu(V_j - \theta)}}$$
$$h_{ex}(V_i, V_j) = \frac{E_{ex} - V_i}{1 + e^{\nu(V_j - \theta)}} \quad (3)$$

where E_{in} and E_{ex} are the inhibitory and excitatory synapses reverse potentials, respectively, whereas ν and θ act on the chemical synapses activation function shape.

A. Phase difference representation

A very useful approach in CPG analysis is the so-called phase difference representation [8]–[11], which allows using nonlinear systems' stability analysis tools on the rhythmic patterns generated by the network. Assuming that the state of the i-th neuron describes a periodic orbit $\hat{\mathbf{z}}_i(t)$ of period T_i, this orbit can be mapped (through the modulo function) to a phase variable $\phi_i \in [0, 1)$ so that ϕ_i is reset to 0 when V_i grows over a threshold V_{th}.

The phase difference representation uses $N - 1$ state variables $\Delta\phi_{1,i} = (\phi_1 - \phi_i) \mod 1$ $(i = 2, \ldots, N)$ and, generally speaking, should be computed by numerically integrating system (1), being not known *a priori*. In order to use continuation analysis and also for reducing the simulation times, under proper assumptions it is possible to approximate the phase difference vector field through a *Phase Resetting Curve* (PRC) [13].

III. Toolbox description

CEPAGE is an object-oriented toolbox for simulation and analysis of CPGs. Figure 1 shows the functional relationships between classes (gray boxes), main methods (solid ellipses) and corresponding output data (white boxes). The dashed ellipses denote external analysis tools that can be applied to the obtained data. Parallel computation, MEX files and the Boost C++ libraries are used to reduce the simulation times. The *neuron* class describes a single neuron and the *CPG* class represents a CPG. The main toolbox functionalities are:

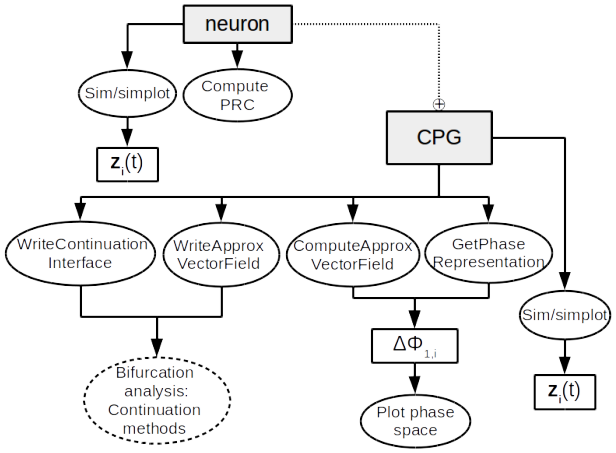


Fig. 1. Relationships between CEPAGE objects.

-) simulation of CPGs: by using method *sim* of class *CPG*, the user can easily obtain the time evolution of the state variables describing the network; it is also possible to start parallel simulations from different initial conditions. If only one initial condition is considered, it is possible to use the *simplot* method, which also plots the state evolution;
-) limit cycle continuation; this functionality is useful when one wants to detect limit cycle bifurcations; through the method *writeContinuationInterface*, it is possible to generate AUTO or MATCONT files for the limit cycle continuation;
-) CPG phase difference simulation: the method *getPhaseRepresentation* of class *CPG* allows obtaining the evolution of the phase differences for the CPG neurons; also in this case, parallel computations can be exploited to integrate the system starting from different initial conditions. The simulation results can then be plotted through the *plotPhaseSpace* method. This functionality can be used to obtain a brute-force bifurcation diagram of the phase differences, but turns out to be very time consuming for relatively large networks;
-) CPG approximate phase difference simulation: the method *computeApproxVectorField* of class *CPG* is useful to carry out brute-force (i.e., based on numerical integrations and Poincaré sections [14]) analysis of the phase differences between neurons reducing the simulation times. The approximate solution works accurately only for weakly-coupled networks and is computed starting from PRCs [15], which can be computed through the method *computePRC* of class *neuron model*;
-) phase difference continuation: the approximate formulation allows also knowing the vector field that describes the phase difference evolution, making it possible a continuation analysis of the patterns generated by the network. CEPAGE can automatically generate files through the method *writeApproxVectorField*, which can be used to carry out continuation analysis with AUTO or MATCONT.

In order to perform a complete bifurcation analysis of a CPG, all the tools described above should be employed, as shown in the next section.

IV. Case studies

A. 3-neuron CPG

As a first test bench, we consider a CPG already analyzed in [13]. This CPG is composed by 3 neurons (see Fig. 2a), described by a modified FitzHugh-Nagumo model, with parameters $E_{in} = -1.5V$, $E_{ex} = 0V$, $\nu = 100V^{-1}$, $\theta = 0V$ and synaptic coupling only inhibitory with[4]

$$g^{in} = 10^{-3} \begin{bmatrix} 0 & 1 & g \\ g & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

where g is the bifurcation parameter and varies in the range $[1, 4]$. Figure 3 shows the obtained bifurcation diagram in the control space: the solid curves have been obtained by CEPAGE through brute-force analysis, whereas the dashed curves have been obtained through AUTO-07P, by using the PRC approximation and the starting data generated by CEPAGE. It is apparent that the two analysis results match better for lower values of the bifurcation parameter, since for stronger couplings the PRC approximation looses its accuracy. This is an original result, which corroborates the analysis carried out in [13]. In the same figure, the plane for $g = 1$ shows a state portrait with five stable equilibrium points, corresponding to coexisting motifs. The basins of attractions of the five stable equilibria are shown with the same color code. For larger values of g, four of them undergo fold (SN) bifurcations with as many unstable equilibria and the CPG generates only one stable (gray) pattern.

(a) (b)

Fig. 2. Analyzed CPGs.

B. 4-neuron CPG

In this section we analyze the 4-neuron CPG (a variant of the CPG studied in [10]) shown in Fig. 2b, in order to highlight the main features of CEPAGE. The proposed results are new. The four neurons are described by the reduced leech heart interneuron model [16], with synapses parameters $E_{in} = -0.0625$V, $E_{ex} = 0$V, $\nu = 1000$V^{-1}, $\theta = -0.03$V and synaptic coupling matrices as follows:

$$g^{in} = 2.5 \cdot 10^{-3} \begin{bmatrix} 0 & 1 & 0 & 0.25 \\ 1 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & g \\ 0 & 0 & g & 0 \end{bmatrix}$$

[4]Each conductance is expressed in μS/cm^2.

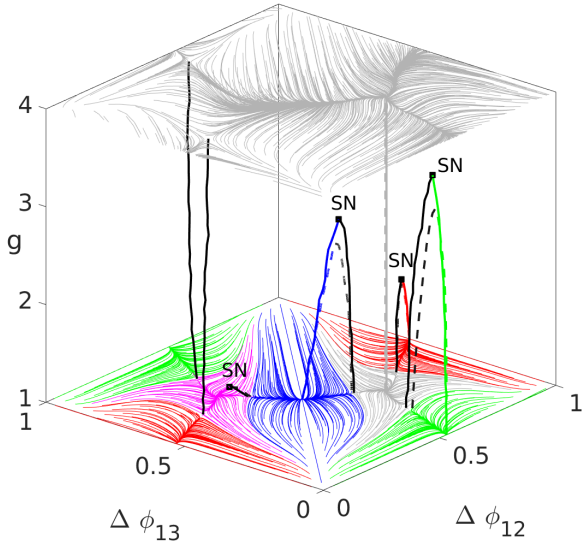


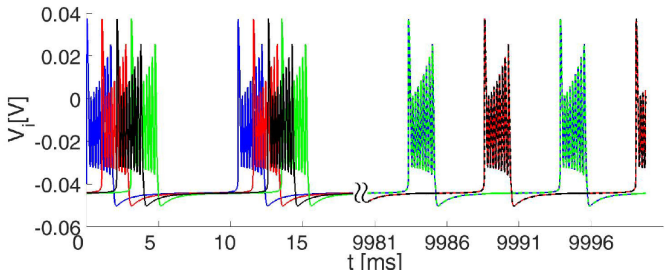
Fig. 3. Bifurcation diagram of the 3-neuron CPG obtained by brute-force (solid lines) and AUTO-07P continuation (dashed lines).

$$g^{ex} = 2.5 \cdot 10^{-3} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.25 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 \end{bmatrix}$$

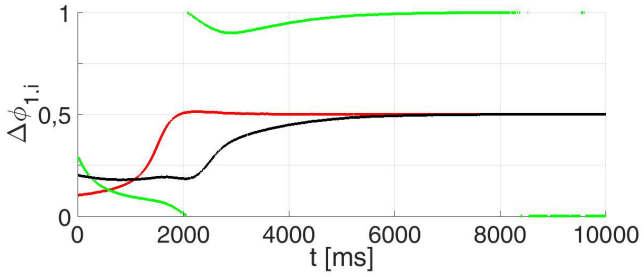
$$g^{el} = 2.5 \cdot 10^{-3} \begin{bmatrix} 0 & 0.25 & 0 & 0 \\ 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0 \end{bmatrix}$$

where the bifurcation parameter g ranges in $[1.4, 4]$.

Figure 4a shows the asymptotic membrane voltages $V_i(t)$ ($i = 1, \ldots, 4$, the line colors correspond to the neuron colors in Fig. 2b) obtained by simulating the CPG for $g = 1.68$. The corresponding phase differences $\Delta\phi_{1,i}$ ($i = 2, \ldots, 4$) are shown in panel b. The behavior of the invariant sets in a projection of the control space is shown in Fig. 5, where the brute-force bifurcation diagram (black and blue curves) is obtained by simulating the CPG for different values of g. $\Delta\phi_{1,2}$ remains fixed to 0.5 and is not shown. By increasing g, the stable motif (black curve) asymptotically reached in Fig. 4b undergoes a fold bifurcation for $g = 3.65$ and disappears. For higher values of g, the CPG loose phase locking (blue points), i.e., it is no more in an Arnold tongue. The transition between these stable invariant sets is shown in Fig. 6, where the gray torus represents the "circular" state variables $\Delta\phi_{1,3}$ and $\Delta\phi_{1,4}$. A similar behavior is observed by decreasing g: in this case the fold bifurcation happens at $g = 1.43$. The stable equilibrium in Fig. 5 has been continued through AUTO-07P by exploiting the files provided by CEPAGE with the approximate vector field. The result is shown in Fig. 5 and is accurate only for low values of g, due to the PRC approximation.



(a)



(b)

Fig. 4. $V_i(t)$ (a) and $\Delta\phi_{1,i}(t)$ (b) for $g = 1.68$. The line colors correspond to the neuron colors in Fig. 2b. The network starts from random phase differences and converges to a phase-locked state.

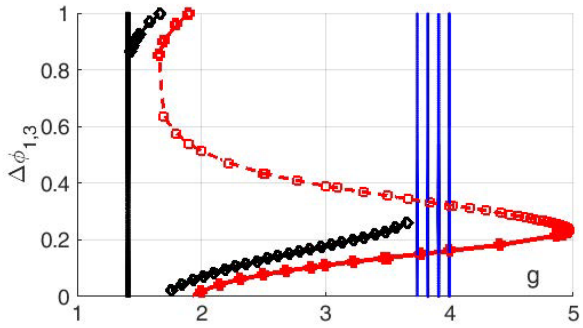
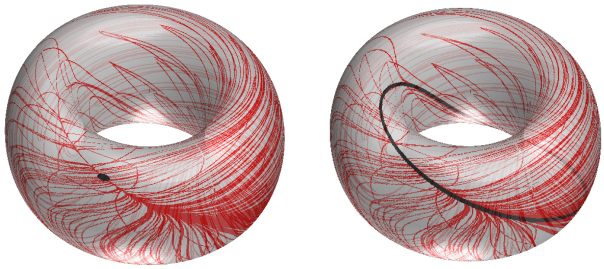


Fig. 5. CEPAGE (black and blue curves) and AUTO-07P (red curves) bifurcation diagrams in a projection of the control space showing stable (solid curves with filled markers) and unstable (dashed red curves) motifs and absence of phase locking (blue points).



(a) (b)

Fig. 6. Phase difference evolution over the torus for $g = 3.5$ (a) and $g = 3.7$ (b). Phase lags converge to (a) a stable fixed point (phase-locking state) or (b) an invariant cycle (phase-slipping state).

V. Concluding Remarks

The results proposed in this paper point out the CEPAGE good features for the analysis of CPGs. From the efficiency standpoint, the simulation of the 3-neuron CPG for 10s starting from 900 initial conditions (to generate the gray state portraits shown in Fig. 3) takes about 15s on a processor intel i7 8-core equipped with 12GB RAM. The brute-force bifurcation diagram of Fig. 3 can be obtained in few minutes (31 values of g). The simulation of the 4-neuron CPG for 10s starting from 1000 initial conditions takes few minutes and the bifurcation diagram (brute-force part) shown in Fig. 5 about 3 hours (50 values of g). The toolbox is still in development and some capabilities will be added, e.g., parallel simulation through GPU computing. From the analysis standpoint, if we imagine that each neuron of the 4-neuron CPG controls the gaits of a quadruped's locomotion, the proposed results could be interpreted as follows: with the same CPG configuration, for lower values of synaptic strength (e.g., $g = 1.68$) the quadruped walks; by increasing g (e.g., for $g = 3.4$), it breaks into a gallop; by further increasing the synaptic strength ($g > 3.7$), we would have an uncoordinated movement.

Acknowledgments

Work partially supported by the University of Genoa, by NSF BIO-DMS (grant IOS-1455527), and by Lobachevsky University of Nizhny Novgorod (RSF grant 14-41-000440).

References

[1] S. L. Hooper, *Central Pattern Generators*. John Wiley & Sons, 2001.
[2] E. Marder and D. Bucher, "Central pattern generators and the control of rhythmic movements," *Curr. Biol.*, **11**, pp. 986 – 996, 2001.
[3] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neur. Netw.*, **21**, pp. 642–653, 2008.
[4] D. M. Abrams, L. M. Pecora, and A. E. Motter, "Introduction to focus issue: Patterns of network synchronization," *Chaos*, **26**, p. 094601, 2016.
[5] E. J. Doedel, "AUTO: A program for the automatic bifurcation analysis of autonomous systems," *Congr. Numer*, **30**, pp. 265–284, 1981.
[6] A. Dhooge, W. Govaerts, and Y. A. Kuznetsov, "MATCONT: a MATLAB package for numerical bifurcation analysis of ODEs," *ACM Trans. Math. Softw.*, **29**, pp. 141–164, 2003.
[7] S. Coros *et al*, "Locomotion skills for simulated quadrupeds," in *ACM Trans. Graph.*, **30**, 2011, p. 59.
[8] L. Zhao and A. Nogaret, "Experimental observation of multistability and dynamic attractors in silicon central pattern generators," *Phys. Rev. E*, **92**, p. 052910, 2015.
[9] R. Barrio, M. Rodríguez, S. Serrano, and A. Shilnikov, "Mechanism of quasi-periodic lag jitter in bursting rhythms by a neuronal network," *Europhys. Lett.*, **112**, p. 38002, 2015.
[10] S. Jalil, D. Allen, J. Youker, and A. Shilnikov, "Toward robust phase-locking in melibe swim central pattern generator models," *Chaos*, **23**, p. 046105, 2013.
[11] J. Wojcik, J. Schwabedal, R. Clewley, and A. L. Shilnikov, "Key bifurcations of bursting polyrhythms in 3-cell central pattern generators," *PloS one*, **9**, p. e92918, 2014.
[12] J. Wojcik, R. Clewley, and A. Shilnikov, "Order parameter for bursting polyrhythms in multifunctional central pattern generators," *Phys. Rev. E*, **83**, p. 056209, 2011.
[13] J. T. Schwabedal, D. E. Knapper, and A. L. Shilnikov, "Qualitative and quantitative stability analysis of penta-rhythmic circuits," *Nonlinearity*, **29**, pp. 3647–3676, 2016.
[14] D. Linaro and M. Storace, "BAL: A library for the brute-force analysis of dynamical systems," *Comp. Phys. Comm.*, **201**, pp. 126–134, 2016.
[15] V. Novičenko and K. Pyragas, "Computation of phase response curves via a direct method adapted to infinitesimal perturbations," *Nonlinear Dyn.*, **67**, pp. 517–526, 2012.
[16] A. Shilnikov, "Complete dynamical analysis of a neuron model," *Nonlinear Dyn.*, **68**, pp. 305–328, 2012.