Multistability control by local sensitivity analysis

Moreno-Morton Luis Rodrigo, Franci Alessio

In this document, we go through the implementation of the control algorithm used in the paper *"Control of multistability through local sensitivity analysis: application to cellular decision-making networks"*. A pseudocode summarizes the relevant steps and, when appropriate, points to specific lines of the source file `gradients.py`, where the reader can find commented python implementations of the various functions used by the algorithm.

As a preliminary step, it is necessary to define the parameterized model vector field in two ways: a numerical one (for simulation of the model dynamics and root-finding of its equilibria) and a symbolic one (for symbolic differentiation). The numerical definition needs to be compatible with the function `solve_ivp()` from the `Python` package `SciPy`. See lines 17 through 109 in `gradients.py` for the implementation of this step in the case of the EMT model of the paper. The symbolic definition needs to be formatted suitably for differentiation using `SymPy`. See lines 119 through 172 in `gradients.py` for the implementation of this step in the case of the EMT model of the paper.

In this preliminary step, we also define the variables (`sys_vars`, line 121), parameters (`sys_pars`, line 122), as well as the **controllable parameters** of the system (`parameters`, lines 141 until end of section), *i.e.* those parameters used to perform the sensitivity analysis leading to the desired parametric control signal. These definitions are used not only for the sensitivity analysis but also for the substitutions from symbolic to numerical values in functions `subs_pars()` (line 185) and `subs_state()` (line 204). We point out that in case of larger dynamical systems most of this preliminary step can be automatized through automatic differentiation tools.

For any questions regarding the use of the functions provided in the script, please contact Rodrigo Moreno at luiromormor@gmail.com.

Algorithm 1 Eigenvalue and saddle control to increase the basin of attraction of the stable equilibrium x_e.

Require: A dictionary with parameter values π. Keys must match symbols of `sys_pars`.
Require: A dictionary of equilibrium points X.
Require: $x_e \in X$, a stable equilibrium point whose basin of attraction is to be stabilized
Require: $s_1, \dots, s_l \in X$, saddle points in the border of the basin of attraction of x_e.
Require: $(\lambda_1, \dots, \lambda_l; v_1, \dots, v_l)$ Controlled eigenvalues and associated eigenvectors of $J^{\pi}_{x_e}$, corresponding to directions pointing from x_e to the surrounding saddle points (see Algorithm 2 below).
Require: δ_c, the desired change in eigenvalues or saddle distance.
Require: n_{iter}, the maximum number of iterations
Require: ϵ, step size in the parameter control direction

1: $X^* \leftarrow X$
2: $x_e^* \leftarrow x_e$
3: $(s_1^*, \dots, s_l^*) \leftarrow (s_1, \dots, s_l)$
4: $\pi^* \leftarrow \pi$
5: $c_d \leftarrow 0$
6: $c_e \leftarrow 0$
7: $quit \leftarrow False$ ▷ Boolean variable to report bifurcations.
8: $ii \leftarrow 0$
9: **while** $ii \leq n_{iter} \wedge c_d < \delta_c \wedge c_e < \delta_c \wedge \neg quit$ **do**
10: $\quad g_d \leftarrow$ Distance gradients ▷ `distance_gradient()`, line 285
11: $\quad g_e \leftarrow$ Eigenvalue gradients ▷ `eigenvar_gradient()`, line 308
12: $\quad g_o \leftarrow$ Optimum parameter control direction (through MGDA) ▷ `optimum_coefficients()`, line 430
13: $\quad \pi^* \leftarrow \pi + \epsilon(g_o)$
14: $\quad X^* \leftarrow \{x; F(x, \pi^*) = 0\}$
15: $\quad$ **if** $\text{length}(X^*) \neq \text{length}(X)$ **then** ▷ A bifurcation occurred
16: $\qquad quit \leftarrow True$
17: $\quad$ **end if**
18: $\quad x_e^* \leftarrow \text{argmin}_{x \in X^*} \|x - x_e^*\|$ ▷ Update stable equilibrium location
19: $\quad s_i^* \leftarrow \text{argmin}_{x \in X^*} \|x - s_i^*\|$, $i = 1, \dots, l$ ▷ Update saddle location
20: $\quad (\lambda_1, \dots, \lambda_l; v_1, \dots, v_l) \leftarrow$ Controlled eigenvalues and eigenvectors at (x_e^*, π^*) ▷ `get_eigenvalues()`, line 216
21: $\quad c_d \leftarrow \sum_{i=1}^{l} \|x_e^* - s_i^*\| - \|x_e - s_i\|$ ▷ Change in distance
22: $\quad c_e \leftarrow \sum_{i=1}^{l} \lambda_i - \lambda_i^*$ ▷ Change in eigenvalues
23: $\quad ii \leftarrow ii + 1$
24: **end while**

Lines 12 through 17 are shown separately so as to make the flow of ideas clear. However, they are encompassed by a single function `control()` in `gradients.py`.

Algorithm 2 Algorithm to select the controlled eigenvalue/eigenvector pairs

Require: A dictionary of equilibrium points

$t \leftarrow$ Threshold selection value
$(E, V) \leftarrow$ Eigenvalue-eigenvector pairs at x_e
$S \leftarrow$ Unstable equilibria that make up borders of basin of attraction
for $s \in S$ **do**
$\quad d \leftarrow x_e - s$
$\quad$ **for** $v \in V$ **do**
$\qquad \theta \leftarrow d \cdot v / \|d\| \|v\|$
$\qquad$ **if** $\theta > t$ **then**
$\qquad\quad$ Store eigenvalue for control
$\qquad$ **else**
$\qquad\quad$ Ignore eigenvalue for the pair (v, s)
$\qquad$ **end if**
$\quad$ **end for**
end for