Instructions to install rGEP on a computer (under windows)

# **A- Preparation of the simulation environment**

**1) Prepares python 3.8 and needed tools**

Install anaconda 3 from [*https://www.anaconda.com*](https://www.anaconda.com)

Normally it comes with the required tools, but check they are present using the Anaconda prompt window with Administrator privileges (launch from Windows or Anaconda) using these commands:

*conda install conda=23.5.0 (last version at this moment, just specify it if changed)*

*conda install PyQtgraph*

*conda install pandas*

*conda install seaborn*

*conda install scipy*

*conda install matplotlib*

Note: If previous versions of those tools are present, use “conda update” instead of “conda install”

You will also need a more specific tool that is not in the Anaconda package. cma is the name of an optimization tool (Covariance Matrix Adaptation Evolution Strategy developed by [Hansen & Ostermeier 2001](http://www.cmap.polytechnique.fr/~nikolaus.hansen/publications.html#hansenaost2001)). To install this tool, use the pip command:

*pip install cma*

In this environment, we will use **Spyder** to run the Python script files (you can launch from Windows or Anaconda Navigator).

**2)You will also need the neuromechanical simulation tool AnimatLab** from [*https://animatlab.com*](https://animatlab.com).

Download the last version and install it.

AnimatLab is a software tool that combines biomechanical simulation and biologically realistic neural networks. Using an intuitive GUI environment, it allows to create a body with bones and muscles, and sensory receptors (proprioceptors such as GTO and muscle spindles; exteroceptors…). After installation, you will find it at:

*C:/Program Files (x86)/NeuroRobotic Technologies/AnimatLab*

**3) Download the complete series of python files from GitHub**

([*https://gitub.u-bordeaux.fr/hybrid\_team/2\_controle\_model\_musculosquel/2\_1\_spinal\_like\_regulator*](https://gitub.u-bordeaux.fr/hybrid_team/2_controle_model_musculosquel/2_1_spinal_like_regulator))

Place them in the following directory that you will create:

*D:/SimulationScripts/Python38/2\_1\_spinal\_like\_regulator*

You supposed to have 2 directories and several .py files in this directory.

You should NOT have the same directory inside (i.e “2\_1\_spinal\_like\_regulator” or “2\_1\_spinal\_like\_regulator-master”)

Make a new directory and copy all files from the previous directory to:

*D:/SimulationScripts/Python38/**AnimatLab\_Python*

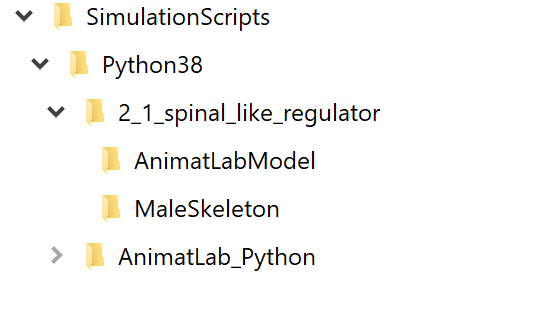
*AnimatLab\_Python is supposed to be the directory you will use for the simulations.*

*2\_1\_spinal\_like\_regulator is supposed to be the backup directory.*

*When you unzip the file downloaded from the GitHub, you might have a duplication of the 2\_1\_spinal\_like\_regulator directory. Make sure to have only ONE directory named like that.*

*NB this directory has to be different from the one containing “Animatlab.exe”.*

Finally, the content of the *D:/SimulationScripts/Python38/* directory should be as follow:

**

**4) Prepare the directories in which GEP process will save data**:

1. With window explorer create a folder: D:/Sim\_SLR/ (*if D not available create this folder in C:)*

Download MaleSkeleton.rar from *GitHub*

(<https://gitub.u-bordeaux.fr/hybrid_team/2_controle_model_musculosquel/2_1_spinal_like_regulator/-/tree/master/MaleSkeleton>)

Unpack it and place the “MaleSkeleton” directory (containing the meshes used to build the bones) in /Sim\_SLR/ sub-directory as follows:

*D:/Sim\_SLR/MaleSkeleton*

*Note: the “Sim\_SLR” sub-directories will contain the results of all simulations. This is a folder different from the “AnimatLab\_Python” that contains all python scripts of this project.*

1. With windows explorer, create a root directory (simple directory) that will contain the simulation results of a given model. For simplicity, let’s say this directory will be NS24:

*D:/Sim\_SLR/*NS24

The term “NS24” indicates that we mill work with an AnimatLab model based on the structure “NS24”. This model was built in AnimatLab. Indeed, the model NS24 contains all neurons and stimulations that can be used for a 14 parameters model (in this case many elements are disabled in the model), or for a 36 parameters model, or for a 50 parameters model. Here we present the case of the NS24\_14

1. Once the root directory is created, create a model sub-directory (*NS24\_14\_model*) to place the AnimatLab model files in:

*D:/Sim\_SLR/*NS24/NS24\_14\_model

*These model files are: “NS24\_14.aproj”, “LineChart.aform” and “SecondLineChart.aform”. These* files can be downloaded from *GitHub*

(<https://gitub.u-bordeaux.fr/hybrid_team/2_controle_model_musculosquel/2_1_spinal_like_regulator/-/tree/master/AnimatLabModel>)

in “model” folder (here *D:/Sim\_SLR/*NS24/NS24\_14\_model).

*Note1: you will certainly build different models to change some characteristics of the initial model. I highly recommend, each time a modification is made to the model, to create a new subdirectory from Sim\_SLR/ (for example “Sim\_SLR/NS24NG/NS24NG\_14\_model” if the modified model is the same except for the absence of gravity (NG = No Gravity). Try to keep names short because there will exist many subdirectories created by the processes…*

Now, you can run **AnimatLab** and open the file “*NS24\_14.aproj”*, in the “NS24\_14\_model” folder (model folder). When you open the aproj file, several windows will pop up: a big one with several parameters and graphs, and a small one where you will have to give two parameters.

In the big window, on the right, you will see two chart windows containing the predefined variables to be plot against time when you run the simulation.

Before running the simulation, check that the arm skeleton is correctly loaded. You can see the skeleton of the arm on the second window from left, top (blue one). If you do not see it, zoom in by maintaining the right click and pull the mouse (maintaining the left click and moving left or right will rotate the arm). If the arm appears with no bones, but with boxes, this means the mesh files were not found. In this case the meshes used to build the bones are not at the correct address in the “ArmNS24.aproj” file. This can be easily corrected using the. ***GUI\_AnimatPar.py*** script*(see d- below).*

1. Quit AnimatLab.

Within **Spyder** environment, go to the Python directory:

*D:/SimulationScripts/Python38/AnimatLab\_Python*

Run the ***GUI\_AnimatPar.py*** script**.** Be sure to run it in a dedicated console.

Run this GUI to change the address of the mesh files in the “NS24\_14.aproj” file. The meshes are used to build the bones in the AnimatLab model. Since AnimatLab uses the complete path to these files, most of the time it will not load the meshes (due to this path being specific to your computer).

To change the paths in the “NS24\_14.aproj” file, simply press “Change mesh path” button (Bottom, left of the GUI). Two windows will pop up in a row. Indicate for each windows respectively:

1. NS24\_14\_model (the folder where is the NS24\_14.aproj),

*D:/Sim\_SLR/NS24/NS24\_14\_model*

1. the *MaleSkeleton* folder:

*D:/Sim\_SLR/MaleSkeleton*

Once this is made, quit ***GUI\_AnimatPar.py*** by pressing the “Quit” button.

Run **AnimatLab** and open the file “*NS24\_14.aproj”*, in the “NS24\_14\_model” folder again.

Check the mesh by zooming on the blue window. If the mesh is correctly loaded, you will see the bones appears in places of the boxes.

Run a simulation by clicking the “Play” button (►).

1. **If not already opened, run the AnimatLab executable** (AnimatLab.exe) and open the NS24\_14.aproj” file to create the “NS24\_14\_StandAlone.asim“ file *(top menu “File” -> “Export Standalone asim”).*

*Note 3: It is essential to recreate the asim file each time the mesh path is modified in the aproj file, in order to give the asim file the correct paths since the python control of the model uses the asim file.*

*WARNING!!! Read Note 4 about the Chart timing (Tool Viewers -> Data Toolxx -> Line Chartxx in the AnimatLab GUI “workspace” (first column left)).*

*WARNING!!! when the Export Standalone Simulation window opens you must uncheck the option “Show Graphic Window”. When unchecked, press OK to save the new asim file.*

Now, you should see a new “NS24\_14.asim” file in the model sub-directory (“NS24\_14\_model”)

1. **Precautions before running** *NS24\_14.aproj* in **AnimatLab executable**:

The arm representation should now be correct (with arm skeleton meshes loaded) in the AnimatLab window. If this is not the case, repeat the “d-e sections”.

In this simplified model, the arm skeleton is moved by two muscles (biceps and triceps) in relation with a simplified spinal network. Three files are required with the extensions: \*.aproj and \*.aform. The \*.aproj is a xml file that contains all the information used to run the model with animatLab. The two \*.aform files are used to define the charts, in which simulation results are stored in text format. The \*.asim file is a xml file that is used in python to run the model with specific parameter values of the spinal network. It is created in AnimatLab *(top menu “File” -> “Export Standalone asim”).*

Descending commands to the various neurons of the spinal network must be prepared. This ensemble of stimulations is already created in the example file. They point on the various neuron targets. We use preparatory stimuli commands (SET) and GO stimuli commands. The SET stim commands start at t=0 and end at t=10s, the GO stim commands start at t=5s and end at t=10s. In AnimatLab these stimulations will be used to move the arm (normally). You can modify the intensity of these stimulations, but not the timing, and check the result by running the model in AnimatLab (the “>” button in simulation control panel). The python scripts (GEP\_GUI.py, controlScriptGEP.py) will use these stimulations to handle arm movements, and explore the effect of varying their values.

**Run a simulation by clicking the “Play” button (►).**

*Note 4: WARNING!!!!!* ***If you have run the model in AnimatLab****, the Chart timing is reset by default to 0.02ms. For space reason, we use a reduced sampling rate of 10ms. You can change it manually in AnimatLab:*

*GO in Workspace window -> “ToolViewers” ->DataToolxx -> LineChart*

*When you press on “LineChart” the corresponding parameter are presented in the bottom window.*

*- Put “Auto collectInterval” to False*

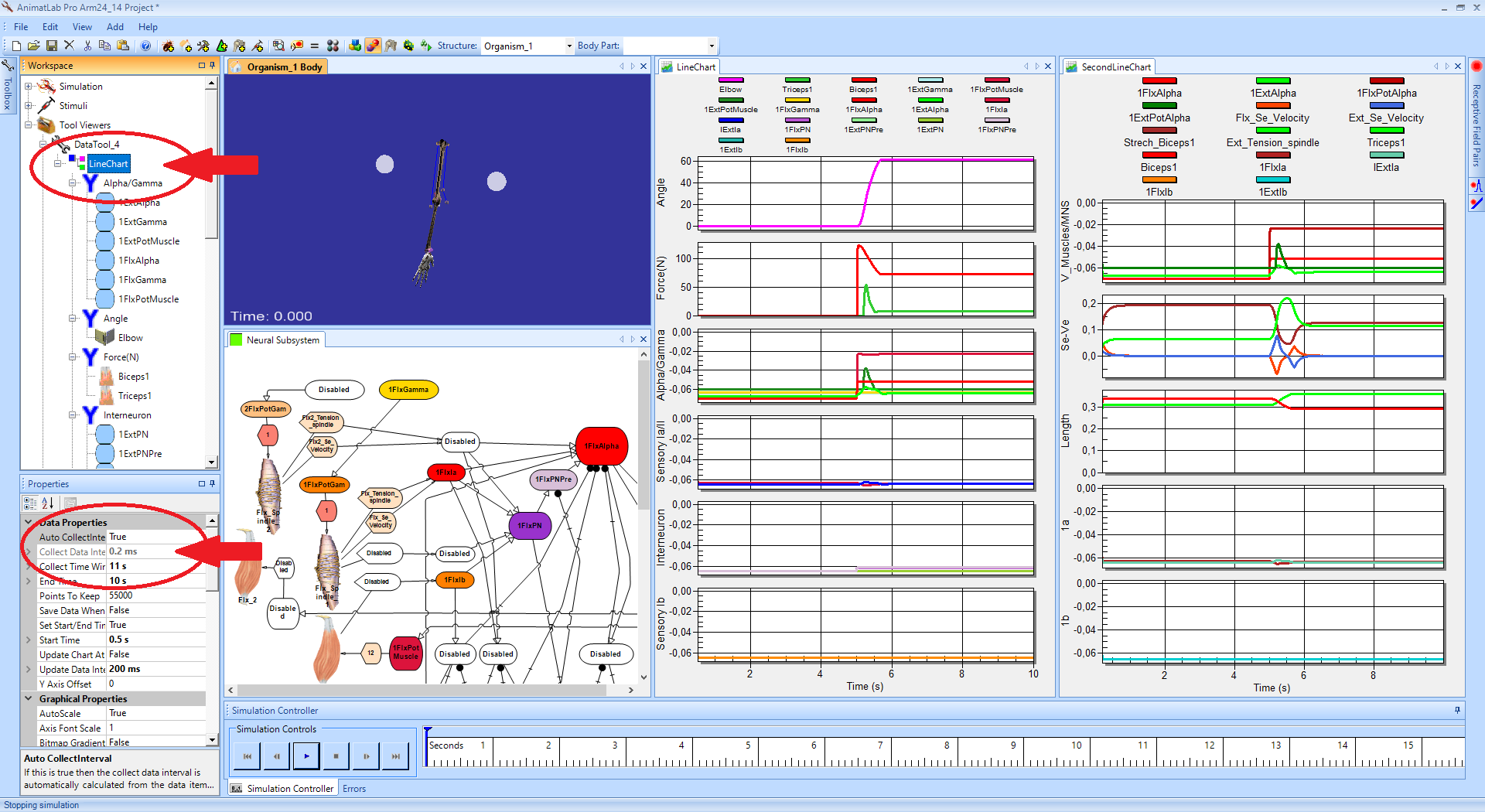
*- Once it is done you can change the “Collect Data Interval” to 10 ms. (in the menu “Collect Data Interval”, change the variable “value”).*

*Repeat this procedure for the second “LineChart”.*

*Only when this is done, you can save the modified NS24\_14.aproj file*

*You must also export the new standalone asim file. Top bar menu -> File -> Export Standalone Sim*

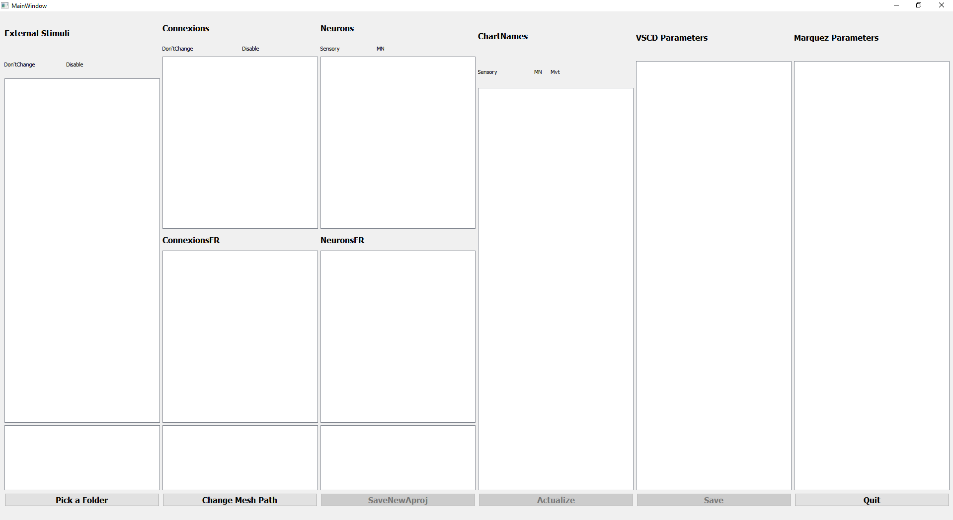
*WARNING!!! when the Export Standalone Simulation window opens you must uncheck the option “Show Graphic Window”. When unchecked press OK to save the new asim file.*

****

**5) Run Spyder environment window that allows to run python scripts.**

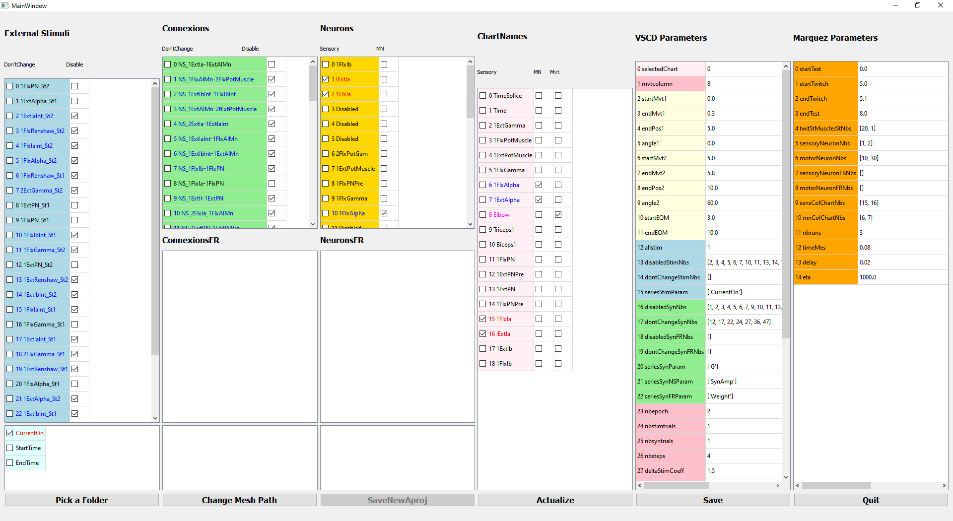
In the **Spyder** environment, run the **GUI\_AnimatPar.py** script.

A large graphic window opens, but a smaller window asks first to select the linechart that will be used to read simulation data. Select the first one:

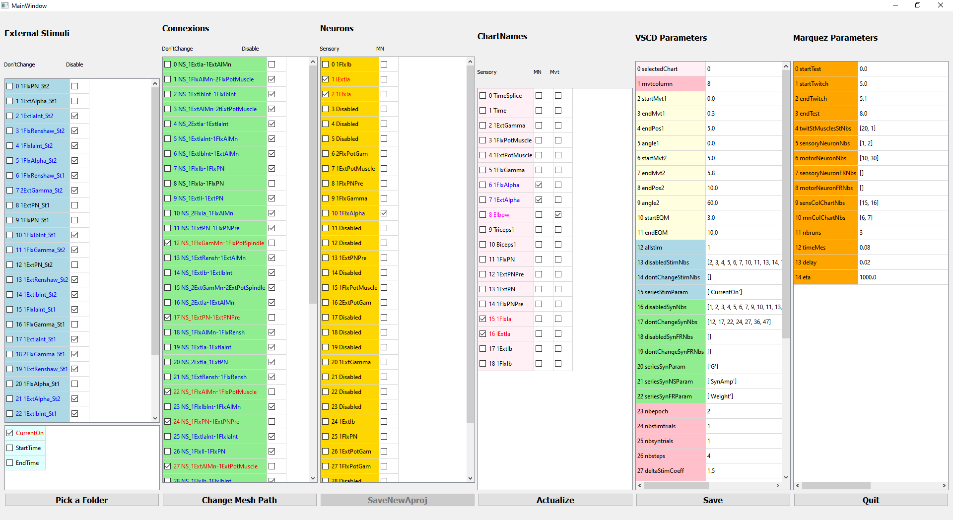


1. Press “pick a folder” button to select the directory containing the prepared AnimatLab model (the model sub-directory): “*NS24\_14\_model*”. A smaller window opens and asks for the chart file to choose. Select the First Line Chart (“0 LineChart”).

The script will read the .asim file to get information on stimulations, neurons and synapses of the model and the selected chart (that are the four first left columns of the GUI).



You can lower the bottom part of the 2nd and 3rd column to have full vision of the Connections and Neurons Parameters:



1. In the first column (blue one), specify what to do with the stimulations:

i: “Don’t change” its value (left case): *NB normally you should not use “don’t change” here.*

ii: “disable” the stimulation (right case): *Note that many are already disabled (as they were in the AnimatLab aproj file). Some stimulations end with “\_St1” (SET commands) while others end with “St2” (GO commands). Here we disable all “\_St2” stimulations except the two that target the PN neurons: 1FlxtPN\_St2 and 1ExtPN\_St2). Some St1 must be disabled too (see below).*

iii: if none of these actions is made, the value of the stimulus will be allowed to vary in the scripts. *We should get the following list as active (black) stimuli (= 8 active stim):*

|  |  |
| --- | --- |
| *1FlxPN\_St2* |  |
| *1ExtAlpha\_St1* |  |
| *1ExtPN\_St1* |  |
| *1FlxPN\_St1* |  |
| *1ExtPN\_St2* |  |
| *1FlxGamma\_St1,* |  |
| *1FlxAlpha\_St1* |  |
| *1ExtGamma\_St1* |  |

iv: In the bottom window, unselect StartTime and EndTime. Keep CurrentOn selected.

1. In the second column (green one), specify what to do with the synapses
   * 1. Disable ALL the synapses (right case): *Note that many are already disabled (due to their status in AnimatLab).* They will turn in blue.
     2. “Don’t change” values (left case): *You must check the “don’t change” case for (= 6 values, in red):*

|  |  |
| --- | --- |
| *NS\_1FlxGamMn-1FlxPotSpindle* |  |
| *NS\_1ExtPN-1ExtPNPre* |  |
| *NS\_1FlxAlMn-1FlxPotMuscle* |  |
| *NS\_1ExtAlMn-1ExtPotMuscle* |  |
| *NS\_1ExtGamMn-1ExtPotSpindle* |  |
| *NS\_1FlxPN-1FlxPNPre* |  |

These synapses don't need to be optimized because the presynaptic element is the only one that needs it.

* + 1. *At the end the list of active synapses must be as follows (= 6 values unchecked, in black):*

|  |  |
| --- | --- |
| *NS\_1ExtIa-1ExtAlMn,* |  |
| *NS\_1FlxIa-1FlxPN,* |  |
| *NS\_1ExtIa-1ExtPN,* |  |
| *NS\_1FlxPNPre-1FlxAlMn,* |  |
| *NS\_1FlxIa-1FlxAlMn,* |  |
| *NS\_1ExtPNPre-1ExtAlMn.* |  |

1. In the third column (yellow one), what are the neurons types (this is not used presently… it is still in development to use muscle twitches to set the strength of proprioceptive feedback… see Marquez et al).
   * 1. In the sensory neuron (left case, check “1FlxIa” and “1ExtIa”)
     2. In the motoneurons (right case, check “1FlxAlpha” and “1ExtAlpha”)
2. In the fourth column (light pink one) explain what represent columns in chart files. Specify where are the MNs, sensory neurons and movement (elbow joint). These elements are essential for the script to run.
   * 1. the sensory neuron (left case, check “1FlxIa” and “1ExtIa”)
     2. the motoneurons (middle case, check “1FlxAlpha” and “1ExtAlpha”)

As soon as you check the case for the MN, a box popup. Click on the MN and a scroll menu pop again. Select the St1 that corresponds to the MN you select (e.g., 1FlxAlpha\_St1 for 1FlxAlpha)

* + 1. the joint that makes the movement (right case, check “Elbow”)

1. The fifth column (multicolor one) defines parameters used in optimization (VSCD, CMAes…). Rows 0 to 11 define movement aim (elbow flexion amplitude and speed (or duration here)). The other parameter rows summarize the choices made in columns 1 to 3 (rows 12-15 for the stimulations on neurons and 16-22 for the synapses). They are updated automatically when changes are made in the three first columns. In addition, row 23-41 give indications on process settings used in the optimization and GEP processes. The contain of the fifth column will be saved (when pressing the “save” button) in a “*.pkl*” format in a sub-directory created automatically by the GUI -> “ResultFiles”. Other directories needed in the optimization and GEP tools are also created automatically when pressing “save” button: “AprojFiles”, “FinalModel”, “SimFiles”, and the needed files are also created in these folders automatically.

**Don’t forgot to press “save” button**

Once this is made Close the **GUI\_AnimatPar** (press the “Quit” button).

At this stage, you have two possibilities to go on: either with **GEP\_GUI.py** or with **ControlScriptGEP.py**

**GEP\_GUI.py** is a GUI in which you control every script to be run, it needs more manipulations but allow to better understand how scripts work.

**ControlScriptGEP.py** is a script running commands from a text file. Needs much less manipulations.

# **B- Instructions to build a Control Script**

**ControlScriptGEP.py** is a program that launches various simulation process, and handles automatically the folders, using instructions written in a text file. There are several steps in the whole process of a such a text file.

**1) First step: create a text script to run CMAes (or VSCD)**

We can write the control script file by hand, but to avoid errors, we have written a python script that does it for us. It is fast and secure.

Under Spyder, run “**buildControlScript.py**”.

1. The window that opens asks for the model sub-directory (“NS24\_14/NS24\_14\_model”) that contains the *aproj, asim* and *aform* files of the AnimatLab model.
2. A warning window reminds you important information about the model directory (press “show details) to read the explanation
3. A dialog box opens and asks if you want to go on or not. If you forgot something (i.e., the meshes path was not corrected in the aproj file) you can stop here (by pressing Escape). But if OK then press OK.
4. Next window asks you to choose between “Optimization” or “GEP”. **Choose “Optimization”**. During optimizations valid behaviors will be produced. From these behaviors we will select seeds for running rGEP (in Third step).
5. Next window asks for “*Base Directory*”. Since no simulation was performed, the 1st time just click the “cancel” button. For the next trials you can choose the appropriate directory.
6. Short identity name definition. You can change the proposed name to: “ID124A”. As soon as you click “OK”, another window will pop.

*Explanation of the name “ID124A”:*

*“124” is just a number that will appear in all graphs and subsequent sub directories to facilitate the identification of the simulation.*

*[(if present) “NG” signifies that the model runs in gravity=0]*

*“A” is just a letter that allows to have several variants of the same model (see below)*

1. Next window proposes model characteristics to be included in the name of the simulation root dir.

*Here, these characteristics indicate how spindle neurons 1a were built:*

*“ID124A\_WM1aDyn20\_B150”*

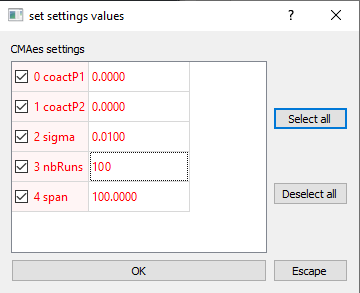
To accept, press OK.

*These characteristics will be included in the full Name of the* ***Simulation Directory****, with other indications (suvh as ordinate “MSpeed” (max speed) and coactivation penalties ”cP0\_0” - see below):*

*“ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0”*

1. Next window asks for gravity value. Since we are in not in No Gravity (NG), set gravity = **- 9.81**. Press OK
2. Next window asks for the movement template characteristics. Press OK
3. Net window asks for the ordinate variable in the behavior window. Select “MaxSpeed” and press OK
4. Next window asks the type of optimization you want to run (CMAes or VSCD). Select CMAes and press OK.
5. The next window asks the values of parameters of the Optimization method you chose. Among those parameters there are: span=100, CoactPenality1=0, CoactPenality2=0. These values will also be included in the ***Simulation directory*** and in the *Script file name*.

In order to test the installation process, put “nbRuns” at 100. Press OK



1. The next window asks to validate the constructed name of the control script text file: it should be:

“CMAes\_ID124A\_span100\_cP0\_0”, and in the python folder containing the python files used in GEP, a new sub directory “Script\_files” should contain the **script file** we have created:

“CMAes\_ID124A\_span100\_cP0\_0.txt”

If you want an explanation on the content of the control script text File, see C - ANNEXES 1)

**2) Second step: run ControlScriptGEP to execute our text script file.**

Run ControlScriptGEP.py. The window that opens asks you to select a script File. Go in “Script\_files” directory:

D:/SimulationScripts/Python38/AnimatLab\_Python/Script\_files

Select the control script file “CMAes\_ID124A\_span100\_cP0\_0.txt”. A window appears indicating the content of the file you selected. Click on “Accept” to validate.

If this is the first time we run simulations, a window will appear: “Give Computer Information”. Follow instructions:

* Choose AnimatLab Program Directory (by default “C:/Program Files (x86)/NeuroRobotic Technologies/AnimatLab/bin)
* Set the number of processors you got. (It can be obtained in the Windows Device Manager)

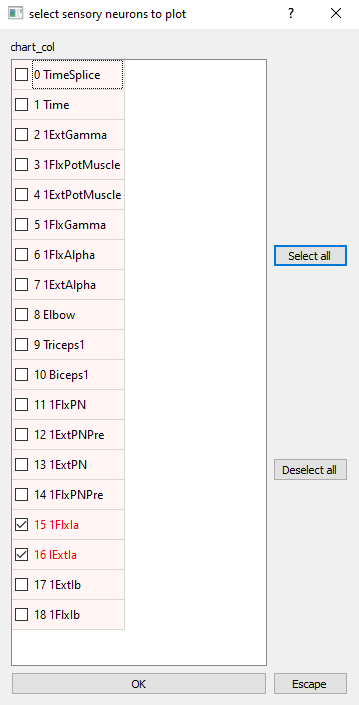
The text file is read and lines are executed. It opens the GEP\_GUI windows and run the simulations.

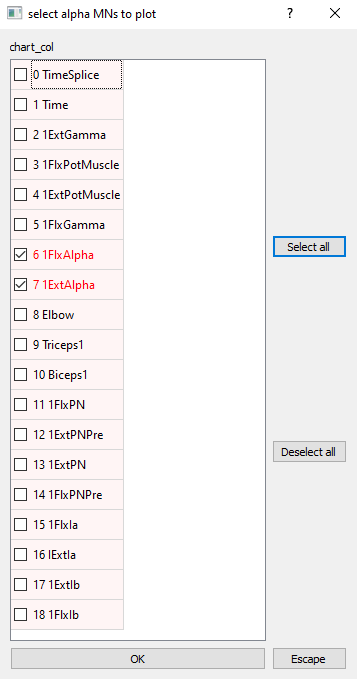
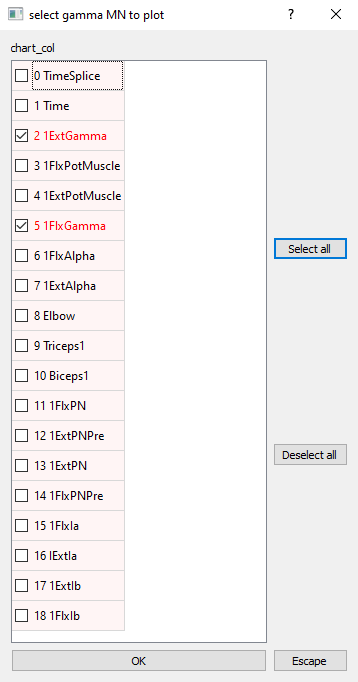
Process can take several hours depending on computer performances.

As soon as the CMAes finishes to process 3 new windows will open in a row asking for columns in the charts:

1. Sensory neurons
2. Alpha MNs
3. Gamma MNs

Be sure to have the following configuration then click “OK”





When finished, it transfers the content of the workDir into appropriate folders that are created.

D:/Sim\_SLR/*NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0*/1\_CMAES\_ID124A\_span100/*ang0-60\_dur1740/trial-0*.

When finished, close the GEP\_GUI window, close the Spyder console in which ControlScriptGEP.py was running.

**3) Third step: run GEP\_GUI to get seeds from optimization results.**

We have run **ControlScriptGEP.py** and we are supposed to have obtained valid behaviors. To get the seeds launch **GEP\_GUI**

The **GEP\_GUI** allows to run all python scripts processes used to explore the descending commands effects onto the spinal network. We will load the result of the CMAes optimization process to get seeds.

1. Once the **GEP\_GUI** script is launched, a first window asks for selecting a folder for the simulation. Select D:/Sim\_SLR/NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/1\_CMAes\_ID124A\_span100\_cP0\_0/ang0-90\_dur1220*/trial-0*.

If you have produced several times the optimization process, you can choose */trial-1* or */trial-2*…

Then other windows are created on the screen.

1. A small window asks: “choose Ordinate for bhv Window”: *Select “maxSpeed”.*

Python script has loaded the elements from the “.aproj” and “.asim” files. You can have a look on the list of elements in the console of Spyder. Note that gravity=-9.81 (this value was read from the “.asim” file).

1. In the GEP\_GUI main window (rightmost) press the “Open a File” button (on the right).
2. In the selection windows that opens select *GEPData00.par*.

The data are loaded. If a dot is present in the “bhvWindow” go directly to **g-**. Otherwise continue here. No dot is drawn in the “bhvWindow” because the optimization process failed to find valid behaviors (i.e., with a varmse < 1). The varmse is a MSE (Mean Square Error) calculated from the actual movement to be tested (with its amplitude and maximum speed, a template of movement with minimum jerk is built, and the movement is compared to this adapted template). Because the template varies with each movement tested, this MSE was named “*varmse*”.

1. To overcome this limitation, we will loosen the mse criterion by pressing the “Filter/mse” button in the bhvWindow.
2. A window proposes several options. Check the “*mseThreshold2 = 10”* (note that, if necessary, you can change the value). If you select several *mseThresholds* (i.e., *mseThreshold1, mseThreshold2, mseThreshold3*…), The behaviors (dot) fulfilling criteria will appear with a color specific to each criterion. Note that in the final selection, the script will concatenate the results. For this example, let’s take the only criterion *mseThreshold2 = 10.* A single dot should be visible in the bhvWindow. If this is not the case increase the value to 12, 15… until you get at least one behavior.
3. Select the dot (simply by left-clicking on it with the mouse arrow).
4. Once a behavior is selected, press the “Get seeds” button in the Main Window (rightmost). The selected dots are identified and the name of a directory is proposed (the directory will be “this name” and will automatically add “\_seeds00”. For this example, it will create “*trial-0\_seeds00*”). From the corresponding parameters, the script runs the model and stores all data about these seeds in the new directory. Note that to allow these “false” seeds to be considered in the rGEP process, their varmse is artificially forced to 0.
5. The program re-runs all selected behavior parameters, and a new directory should be created:

*D:/…/Sim\_SLR/NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/1\_CMAES\_ID124A\_span100/ang0-60\_dur1740/****trial-0\_seeds00***

1. It creates also a new sub-directory in the Simulation directory:

*D:/…/Sim\_SLR/NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/****0\_ID124\_seeds00***

1. Once finished, close the GEP-GUI graphic window (press the “Quit” button), and close the console in which GEP\_GUI.py was running.

**4) Fourth step: create a text script to run GEPrand**

We create a new control script by using “**buildControlScript.py**” (in Spyder environment).

Run the “**buildControlScript.py**” file. The 1st directory you have to choose is the model directory:

*D:/Sim\_SLR/NS24/NS24\_14\_model/*

A box pop, if the check is OK, click OK. Another box pop to ask if you want to go on, click OK.

A 3rd box opens asking to choose between Optimization or GEP. This time, instead of choosing “Optimize”, **you will choose** ‘**GEP**’. The directory to choose is the base directory:

*D:/Sim\_SLR/NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0* */0\_ID124A\_base/*

A series of 4 boxes will pop. Control the rightness and then click OK

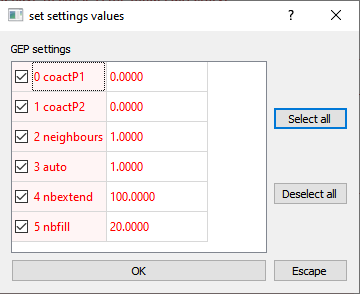
The next box will ask for the Graph ordinate. Prefer Max Speed.

You now have to select the seeds directory:

*D:/Sim\_SLR/NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/0\_ID124A\_seeds00/*

At the next point, you must choose between two possibilities for span. **Span** determines the amount of variation allowed around parent (seed) parameters to produce a new parameter set to be tested. Either you choose a single span value that will be applied for all parameters, or you can choose a list of span values adapted for each parameter. Such span lists are produced automatically when you select seeds (see 3)) in the case valid behaviors exist. For testing the program, prefer “unique span value” and set it to 3.

The following box pop:



**WARNING**: At this point, in order to test the installation, you will have to change “nbextend” to 10 and “nbfill” to 1 by clicking directly on the number inside the window

Then press OK to all Dialog windows to validate proposed choices… you can reduce the number of extends if you want to run a short trial.

A new control script file is created in the Script\_files directory:

“rGEP\_ID124A\_span05\_cP0\_0.txt” if you chose single span

“rGEP\_ID124A\_spanAuto\_cP0\_0.txt” if you chose list of span values

This text script could be:

create\_workDir Path=D: /…/Sim\_SLR/ DirName=workDir\_animatlab

copy\_to\_workDir PathSrc= D: /…/Sim\_SLR/NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/0\_ ID124A \_seeds00

gravity g=-9.81

angle 0 60 max\_speed 75

bhvCriteria bhv\_ordinate=maxSpeed

GEPrand xCoactPenality1=0 xCoactPenality2=0 neighbours=1 auto=1 nbextend=1000 nbfill=50

span=5 or span=from\_file (depending on the choice made for span)

GEPMetrixGraphs

transfert\_from\_workDir PathDest= D: /…/Sim\_SLR/NS24/NS24\_14 \_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/2\_rGEP\_ ID124A\_span02

*Explanations of the script*

*line 2: In this text file, we copy the content of the “0\_ID124A\_seeds00” folder into the workDir in which GEPrand will run*

*line 6: “GEPrand” instruction will launch GEPrand with the settings:*

*xCoactPenality1=0 xCoactPenality2=0 neighbours=1 auto=1 nbextend=1000 nbfill=50*

*line 7: span=5 is applied to the GEPrand*

*Note9: In GEPrand, span is used differently from what is done in CMAes (where it really limits parameter space). Here span is relative to the behaviors that are used to get new valid behaviors. Indeed, in GEPrand, span is the range of variation (%) around the parameter value of the parent behavior. We use a small value of span to facilitate the discovery of valid behavior. But as new valid movements are discovered, the span applies to these new parameters… Thereby, the parameters of the 1000th run set may be very far from the initial ones (i.e., much farer than the span value span=2%).*

**5) Fifth step: run controlScriptGEP.py to execute the second text script file**

Run ControlScriptGEP.py, and select the last file created in the following directory:

*D:/SimulationScripts/Python38/AnimatLab\_Python/Script\_files/*

*Press OK to all Dialog windows*

*Simulations are now running. Just wait (it can take several minutes depending on the computer you have).*

The script is executed, new directories are created and the processed launched, and the final results are saved in workDir and then transferred into the specified directory (that is created automatically)

C:/Sim\_SLR/NS24/*S24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0*/2\_GEP\_ID124A\_span002.

Check if this directory exists to validate the full installation.

# **C- ANNEXES**

1. **Structure of the first Control script text file (for CMAes):**

We start from the folder “*0\_NS24\_14\_base*” containing the required subdirectories and valid files (See A4, A5). The first step consists in creating a script text file that will organize and run the simulations, using the information contained in “*0\_NS24\_14\_base*” folder to obtain some valid behaviors with CMAes or VSCD (or at least some behaviors that are not too far from valid behaviors.

Script:

create\_workDir Path=D: /…/Sim\_SLR/ DirName=workDir\_animatlab

copy\_to\_workDir PathSrc= D: /…/Sim\_SLR/NS24/*NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0*/*0\_NS24\_14\_base*

gravity g=-9.81

angle 0 60 max\_speed 75

bhvCriteria bhv\_ordinate=maxSpeed

cmaes xCoactPenality1=0 xCoactPenality2=0 threshold=Var cmaes\_sigma=0.005 nbTotCMAesRuns=10

span=100

#GEPMetrixGraphs

transfert\_from\_workDir PathDest= D: /…/Sim\_SLR/NS24/*NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0*/*2\_CMAES\_NS24\_14\_span100*

Explanations of the script

line 1: “*create\_workDir*” instruction will create a folder named “*workDir\_animatlab*” using the path location (Path=…):

D: /…/Sim\_SLR/ workDir\_animatlabwill be created

line 2: “*copy\_to\_workDir*” instruction will copy the content of PathSrc=… to the *workDir* folder

The content of “*0\_NS24\_14\_base*” will be copied into D: /…/Sim\_SLR/ workDir\_animatlab)

line 3: “*gravity*” instruction indicates the value of gravity that is given by next instruction (here g=0).

*NB: For a No gravity replace by g=0*

line 4: “*angle 0 60 max\_speed 75*” instructions indicate that the following operations form a complete block that will be executed before the next block is executed. For an optimization process (CMAes or VSCD), any produced movement will be compared to the template defined by amplitude = 60 (starting form angle 0 i.e., vertical resting position), max\_speed = 75.

line 5: “*bhvCriteria*” instruction indicates which ordinate to choose for the behavior space: bhv\_ordinate=maxSpeed

line 6: “*cmaes*” instruction indicates that the script will execute a CMAes with the following configuration:

xCoactPenality1=0 xCoactPenality2=0 threshold=Var cmaes\_sigma=0.005 nbTotCMAesRuns=100

Explanation on cmaes configuration:

*xCoactPenality1* and *xCoactPenality2* are the penalty factors by which coactivation of MNs is multiplied during the preparatory phase and the stabilization phase after movement from 0 to 60 degrees, respectively.

*threshold* indicate which movement chart to save (together with its asim and aproj files). Here “*threshold=Var*” indicates that each time a better movement is found (i.e., each time MSE decreases), it is saved. Another possibility is to give the threshold MSE below which movements are saved (for example: “*threshold=1*” or “*threshold=1.5*” …)

*cmaes\_sigma* is the sigma used to start CMAes. Here *cmaes\_sigma=0.005*).

*nbTotCMAesRuns* is the total number of simulations runs. Usually, CMAes processes by bouts of 10 runs before preparing the next set of runs. Here *nbTotCMAesRuns=100* indicates that there will be 10 bouts of 10 runs (=100runs)

line 7: *span=100* is indeed also associated with the preceding line. “*Span*” is the percentage of parameter space in which parameters are allowed to vary. Here the value of 100% indicate that parameters are allowed to be modified up to the limits ([0, 1], see NB2).

NB1: *When the script reads the lines it reads the first term (here “cmaes”) to know what to do with the remaining of the line. In the case of cmaes, or VSCD or GEPrand, the next line is also read to get the span value for this process.*

NB2: *In all simulations, all parameters have been normalized between 0 and 1. Therefore, span applies to this normalized space.*

NB3: *span can be either a global value applied to all parameters (this is the case here). But it can be specific to each parameter. In the process to get seeds, there is a search for individual span values for each parameter. If such individual spans were found, they are saved in the “datastructure” file (GEPdataa00.par) located in the GEPdata sub-directory. If this is the case you can specify to use this specific span collection (span=from\_datastructure). But sometime this is not possible; in such occurrences you must specify the span value to be used (span=2; or span=10 …)*

line 8: “*GEPMetrixGraphs”* instruction will produce several graphs useful to appreciate the processes. These graphs are saved in a sub-directory named “graphs” that is automatically created. However, here this line has been commented by insertion of a “#” and will not be executed.

line 9: “*transfert\_from\_workDir”* instruction will displace the contend of the working directory (workDir\_animatlab) into a directory specified by:

PathDest= D: /…/Sim\_SLR/NS24/ *NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0*//*2\_CMAES\_NS24\_14\_span100*

This directory will be automatically created.

*Note8: When using a control script (with* ***ControlScriptGEP****), the “transfert\_from\_workDir” command will create a more complex directory structure than what we did in section B, because you may try an optimization for various aim movements (angle of 20, 30, 60, 120 degrees, with other max\_speed values), and you may repeat the process several times. Therefore, the ControlScriptGEP will save the result of the process (*“*transfert\_from\_workDir”) not directly into the PathDest, but in a subdirectory of this folder: ang0-60\_dur1740/trial-0. In this name, ang0-60 indicate the start and end angles of the template movement, and dur1740 is the corresponding duration of the movement phase for a maxSpeed of 75 with an amplitude of 60°. The sub-sub-directory (/trial-0) is there to separate several repetitions of the same process. Indeed, if int the text script, lines 2 to 9 (i.e., copy\_to\_workDir to transfert\_from\_workDir lines). The whole process will run as many times as the sequence is repeated and saved in /trial-1, /trial-2… sub-sub-directories under the /ang0-60\_dur1740 sub-directory (if the same angle line is used). In case of repeated processes, you can separate these process lines by empty lines to clarify the human reading. This has no incidence on the execution of the script.*

*Example:*

create\_workDir Path=D: /…/Sim\_SLR/ DirName=workDir\_animatlab

copy\_to\_workDir PathSrc= D: /…/Sim\_SLR/*NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0*/0\_NS24\_14\_base

gravity g=-9.81

angle 0 60 max\_speed 75

bhvCriteria bhv\_ordinate=maxSpeed

cmaes xCoactPenality1=0 xCoactPenality2=0 threshold=Var cmaes\_sigma=0.005 nbTotCMAesRuns=10

span=100

#GEPMetrixGraphs

transfert\_from\_workDir PathDest= *D: /…/Sim\_SLR/NS24*/*NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0*/1\_CMAES\_NS24\_14\_span100

copy\_to\_workDir PathSrc= D: /…/Sim\_SLR/*NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0*/0\_NS24\_14\_base

gravity g=-9.81

angle 0 60 max\_speed 75

bhvCriteria bhv\_ordinate=maxSpeed

cmaes xCoactPenality1=0 xCoactPenality2=0 threshold=Var cmaes\_sigma=0.005 nbTotCMAesRuns=10

span=100

#GEPMetrixGraphs

transfert\_from\_workDir PathDest= D: /…/Sim\_SLR/NS24*/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0*/1\_CMAES\_NS24\_14\_span100

…

*But you can alco replace CMAes by VSCD:*

copy\_to\_workDir PathSrc= D: /…/Sim\_SLR*/NS24*/*NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0*/0\_NS24\_14\_base

gravity g=-9.81

angle 0 60 max\_speed 75

bhvCriteria bhv\_ordinate=maxSpeed

VSCD xCoactPenality1=0 xCoactPenality2=0 deltacoeff=0.01 nbsteps=3 nbepoch=4

span=100

#GEPMetrixGraphs

transfert\_from\_workDir PathDest= D: /…/Sim\_SLR*/NS24*/*NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/*1\_VSCD\_NS24\_14\_span100

*In this case for explanations of the parameters indicated after VSCD instruction, see B4d. Note that PathDest is modified accordingly*

**2) Instructions to use GEP\_GUI (in case we do not use Control Script Files)**

### a- Prepare the sub-directories in which CMAes will run.

If you do not use the BuildControlScript.py script, you will have to handle folders creation…

The **GUI\_AnimatLabOptimization.py** script, was used to build the subdirectories of the simulation directory *NS24\_14\_model*. If you do not use the BuildControlScript.py script, you will copy the content of this directory in *0\_ID124A\_14\_base* directory. It is necessary to preserve *0\_ID124A\_14\_base* subdirectory because it will serve as base for all subsequent simulations. So never run GEP\_GUI in this directory, but copy its content into the working folder.

We will first try CMAes to get some valid behaviors (see below definition of valid behavior). But before runnn CMAes, we need to create the working directory:

i - Create a new directory “*1\_NS24\_14\_CMAes\_span100*”

It should be placed like this:

*…/Sim\_SLR/NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/1\_ ID124A \_14\_CMAes\_span100*

*Note: The term “span100” is just to indicate that CMAes can explore the whole range (100%) of parameters (limits were set in* **GUI\_AnimatLabOptimization.py***).*

ii - In this new directory, we copy all the files and directories from the *0\_ ID124A \_14\_base* directory.

*Definition: A Valid Behavior is defined as a flexion of the elbow starting from fully extended position and reaching a stable flexed position with an angle >10 degrees (without oscillations; and following a minimum jerk profile). These valid behaviors will be searched using the CMAes tool.* The “aim” movement was defined in the preceding step using **GUI\_AnimatLabOptimization.py** in thefifth column of the GUI. A template of this “aim” movement has been created (“Template.txt”) in the sub directory “ResultFiles” of the *0\_ID124A\_14\_base* directory.

### b- In the spyder environment, run the GEP\_GUI.py python script to run CMAes

The **GEP\_GUI** allows to run all python scripts processes used to explore the descending commands effects onto the spinal network.

1. Once the script is launched, a first window asks for selecting a folder for the simulation. Select *…/Sim\_SLR/NS24/ NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/1\_ ID124A \_14\_CMAes\_span100* folder.
2. Then other windows are created on the screen.
3. A small window asks: “select the ordinate for behavior”: *Select “maxSpeed”.*

Python script has loaded the elements from the “.aproj” and “.asim” files. You can have a look on the list of elements in the console of spyder. Note that gravity=0 (this value was red from the “.asim” file)

1. In the main window of the GEP\_GUI (rightmost window) prepare the parameters of the CMAes process to be run:
2. **In the last line**: set Gravity=0 (already done); CoactPenality1=0; CoactPenality2=0. *Note that Coactpenality1,2 are the multiplication factors for coactivity in Flex MN and Ext MN during the preparatory phase (2s<t<5s) and (7s<t<10s) for CoactPenality1 and CoactPenality2, respectively. By setting them to 0 we allow coactivity during these two phases.*
3. **In the penultimate line, activate the “Use Global span (%)” with a value of 100.** This will allow the CMAes process to use to whole range (100%) of parameter domains.
4. **Launch the CMAes for 100 runs.** The number of runs is specified in the box, close to the CMAes button. To launch CMAes, press the **CMAes** button. The simulations start. It will take at least one hour (but it depends on your computer performances). *When finished, the script asks to select three sets of parameters to be plotted against time. Validate the two first proposed sets. For the third set, select the parameters you want to check. For example: 1ExtPN, 1FxlPN, 1ExtPNPre, 1FlxPNPre to check the pathway PN -> MN. Plots are made and saved in* ***ResultFiles*** *sub-directory.*
5. Once finished, before launching another process, quit the GUI (press the “Quit” button) and close the console in which GEP\_GUI was running.

**3) Prepare the organization of the folders to run the VSCD optimization process.**

We will use again the *0\_NS24\_14\_base* and copy its content in a new directory.

1. Create a new directory:

*…/Sim\_SLR/NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/1\_ID124A\_VSCD\_Span100\_Delta0.01*

1. Copy the content of *0\_ID124A\_14\_base* directory in the new folder.

*Note6: Indeed, the “base” directory is useful to easily create new directories for running various Optimization processes, starting always from the same configuration. This is important, because during the process of optimization, the best optimized set of parameters is saved in the asim file in the subdirectory “FinalModel”. Each time you start a new optimization process, this is in this subdirectory that the starting parameter values are taken from. Therefore, to start with the same configuration, we always copy the directory content from 0\_ID124A\_14\_base.*

*Note7: As you can see, the directories are organized from the root directory (…/Sim\_SLR/ NS24\_14\_model/) with the following rules:*

* *0\_ID124\_base*
* *0\_ID124\_Seeds00*
* *0\_ID124\_Seeds01*
* *…*
* *1\_ID124\_CMAes\_Span100*
* *1\_ID124\_VSCD\_Span100\_delta0.01*
* *1\_ID124\_VSCD\_Span020\_delta0.001*
* *…*
* *2\_ID124\_rGEP\_Span002*

*Sub-directories starting with ”0\_” are starting points for simulations (we have already seen 0\_ID124\_base used to copy the model files into the “1\_ Arm24\_14\_CMAes\_Span100” subdirectory before running CMAes.*

*Sub-directories starting with ”1\_” are Subdirectories in which an optimization process has been run. Seeds will be selected from such sub directories (see below).*

*Sub-directories starting with ”2” are Subdirectories in which rGEP has been run (see below).*

*It is interesting to give information on the settings used in the process. For example, the span value (amount of space (%) in which parameter are allowed to change around their actual value), or the delta value for VSCD (delta is the starting amount of change operated in each of the 14 parameters of the NS24\_14\_model model). Delta is adapted after each single run to accelerate the convergence to the solution.*

**4) In the spyder environment, run GEP\_GUI.py to run VSCD**

The **GEP\_GUI** allows to run all python scripts processes used to explore the descending commands effects onto the spinal network.

1. Once the script is launched, a first window asks for selecting a folder for the simulation. Select *…/Sim\_SLR/NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/1\_ID124A\_VSCD\_span100\_delta0.01* folder.
2. Then other windows are created on the screen.
3. A small window asks: “select the ordinate for behavior”: *Select “maxSpeed”.*

Python script has loaded the elements from the “.aproj” and “.asim” files. You can have a look on the list of elements in the console of spyder. Note that gravity=0 (this value was red from the “.asim” file)

1. In the main window of the GEP\_GUI (rightmost window) prepare the parameters of the CMAes process to be run:
2. **In the last line**: set Gravity=0 (already done); CoactPenality1=0; CoactPenality2=0. *Note that Coactpenality1,2 are the multiplication factors for coactivity in Flex MN and Ext MN during the preparatory phase (2s<t<5s) and (7s<t<10s) for CoactPenality1 and CoactPenality2, respectively. By setting them to 0 we allow coactivity during these two phases.*
3. **In the penultimate line, activate the “Use Global span (%)” with a value of 100.** This will allow the CMAes process to use to whole range (100%) of parameter domains.
4. **Launch the VSCD for 3 epochs and 3 steps per parameter.** The number of epochs is specified in the box, close to the VSCD button. In an **epoch** each single parameter is adjusted a number of times (= **nb of steps**). As a consequence, with 14 parameters, **nbsteps**=3 and **nbepochs**=3, a total of 126 runs will be made. Note that the order of parameter optimization is random and different at each epoch. To launch VSCD, press the **VSCD** button. The simulations start. It will take at least one hour (but it depends on your computer performances). *When finished, the script asks to select three sets of parameters to be plotted against time. Validate the two first proposed sets. For the third set, select the parameters you want to check. For example: 1ExtPN, 1FxlPN, 1ExtPNPre, 1FlxPNPre to check the pathway PN -> MN. Plots are made and saved in* ***ResultFiles*** *sub-directory.*
5. Once finished, before launching another process, quit the GUI (press the “Quit” button) and close the console in which GEP\_GUI was running.

NB: It is important to quit the GUI before changing of activity (even if it seems strange because the commands seem to be accessible). This is to start with a clean background. If we do not quit and close the console, the script may crash…

**5) In the spyder environment, run GEP\_GUI.py to Get Seeds**

The **GEP\_GUI** allows to run all python scripts processes used to explore the descending commands effects onto the spinal network. We will load the result of a CMAes optimization method to get seeds.

1. Once the **GEP\_GUI** script is launched, a first window asks for selecting a folder for the simulation. Select *…/Sim\_SLR/NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/1\_ID124A\_CMAes\_span100* folder.
2. Then other windows are created on the screen.
3. A small window asks: “select the ordinate for behavior”: *Select “maxSpeed”.*

Python script has loaded the elements from the “.aproj” and “.asim” files. You can have a look on the list of elements in the console of spyder. Note that gravity=0 (this value was red from the “.asim” file).

1. In the GEP\_GUI main window (rightmost) press the “Open a File” button (on the right).
2. In the selection windows that opens select *GEPData00.par*.

The data are loaded. But no dot is drawn in the “bhvWindow”. This is because the optimization process failed to find valid behaviors (i.e., with a varmse < 1). The varmse is a MSE (Mean Square Error) calculated from the actual movement to be tested (with its amplitude and maximum speed, a template of movement with minimum jerk is built, and the movement is compared to this adapted template). Because the template varies with each movement tested, this MSE was named “*varmse*”.

1. To overcome this limitation, we will loosen the mse criterion by pressing the “Filter/mse” button in the bhvWindow.
2. A window proposes several options. Check the “*mseThreshold2 = 10”* (note that, if necessary, you can change the value). If you select several *mseThresholds* (i.e., *mseThreshold1, mseThreshold2, mseThreshold3*…), The behaviors (dot) fulfilling criteria will appear with a color specific to each criterion. Note that in the final selection, the script will concatenate the results. For this example, let’s take the only criterion *mseThreshold2 = 10.* A single dot should be visible in the bhvWindow. If this is not the case increase the value to 12, 15… until you get at least one behavior.
3. Select the dot (simply by left-clicking on it with the mouse arrow).
4. Once a behavior is selected, press the “Get seeds” button in the Main Window (rightmost). The selected dots are identified and the name of a directory is proposed (the final directory will be “this name\_seeds00” (here, it will be “*1\_ID124A\_CMAes\_span100\_seeds00*”). From the corresponding parameters, the script runs the model and stores all data about these seeds in the new directory. Note that to allow these “false” seeds to be considered in the rGEP process, their varmse is artificially forced to 0. The new seeds are stored in a new directory: *0\_ID124A\_seeds00*

**6) Prepare the organization of the folders adding a folder to run rGEP with seeds00.**

In order to preserve the automatically created *0\_ID124A\_seeds00* folder, we will run rGEP in another folder:

1. Create a new directory:

*…/Sim\_SLR/ NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/2\_rGEP\_ID124A\_seeds00\_span002*

1. Copy the content of *0\_ID124A\_seeds00* directory in the new folder

**7) In the Spyder environment, run GEP\_GUI.py to run GEP**

1. Once the **GEP\_GUI** script is launched, a first window asks for selecting a folder for the simulation. Select *…/Sim\_SLR/NS24/NS24\_14\_ID124A\_WM1aDyn20\_B150\_bhvMSpeed\_cP0\_0/2\_rGEP\_ID124A\_seeds00\_span002* folder.
2. Then other windows are created on the screen.
3. A small window asks: “select the ordinate for behavior”: *Select “maxSpeed”.*
4. In the GEP\_GUI main window (rightmost) press the “Open a File” button (on the right).
5. In the selection windows that opens select *GEPData00.par*.
6. In the bhvWindow, a dot appears (because its varmse was forced to 0)
7. In the main window of the GEP\_GUI (rightmost window) prepare the parameters of the rGEP process to be run:
8. **In the last line**: set Gravity=0 (already done); CoactPenality1=0; CoactPenality2=0.
9. **In the penultimate line, activate the “Use Global span (%)” with a value of 2.** This will allow the **GEPrand** process to modify the parameters of the valid behaviors up to 2% of their parameter domains.
10. In the line of **GERrand**, activate “*Auto aim*”, set “*neighbrs*” to 1, set “*Nb ext*” to 500, set “*Nb fill*” to 50

In “*Auto aim”* mode, GEP will automatically search aims (arm movements) in the periphery of the already found behaviors (arm movements with a varmse<1).

*The number of neighbors “neighbrs” specifies how many valid behaviors in the vicinity of the automatically defined aim will be used to set the modified parameters. This option is still under development. So, it must be left to 1.*

The two elements “*Nb ext*” and “*Nb fill*” indicate the number of times all peripheral behaviors will be used (“*Nb ext*”) to extend the behavioral domain, and the number of times empty spaces within the domain will be used to attempt filling them (“*Nb fill*”).

1. **Launch the GEPrand ...** by pressing the **GERrand** button.

The GEPrand script run for several hours depending of computer performances. The scripts use parallel runs. Hence, the greater number of processors, the faster.