Supplementary Document 2

Computational reconstruction of pancreatic islets as a tool for structural and functional analysis

G. J. Félix-Martínez, A. Nicolás-Mata and J. R. Godínez-Fernández

Table of contents

Installation

1. Installing IsletLab in Linux

1. Verify that the gcc compiler is installed. Open a Terminal and enter: `gcc --version` and press Enter. If installed, the version of the gcc compiler must be displayed, otherwise, install it by type the following commands (sudo privileges are needed):

   ```
   sudo apt update
   ```

   ```
   sudo apt install build-essential
   ```

   Verify the installation by typing:

   ```
   gcc --version
   ```

2. Verify that the nvcc compiler is installed. Open a Terminal and enter: `nvcc --version` and press Enter. If installed, the version of the nvcc compiler must be displayed. Otherwise, download the toolkit and follow the installation instructions. Once the installation is finished, open a terminal and enter `nvcc --version` to verify the installation. (Note: If your computer does not have a capable GPU, the nvcc compiler will not be available and you will not be able to perform functional simulations in IsletLab. However, it will be still possible to reconstruct islets).

3. Download and install Anaconda (Python 3.8 or 3.9)

4. Download the **Application file** or clone the Isletlab repository from https://github.com/gjfelix/IsletLab. If you downloaded the repository as a zip file, extract the files.
5. Open a Terminal in the base environment and go to the repository folder.
6. Create a new environment using the **isletlabgui_v1.0.yml** file. All the python modules needed will be installed automatically.

```
conda env create -f isletlabgui_v1.0.yml
```

7. Activate the new environment

```
conda activate isletlab_v1.0
```

8. Run Isletlab:

```
python isletlabgui_v1.0.py
```

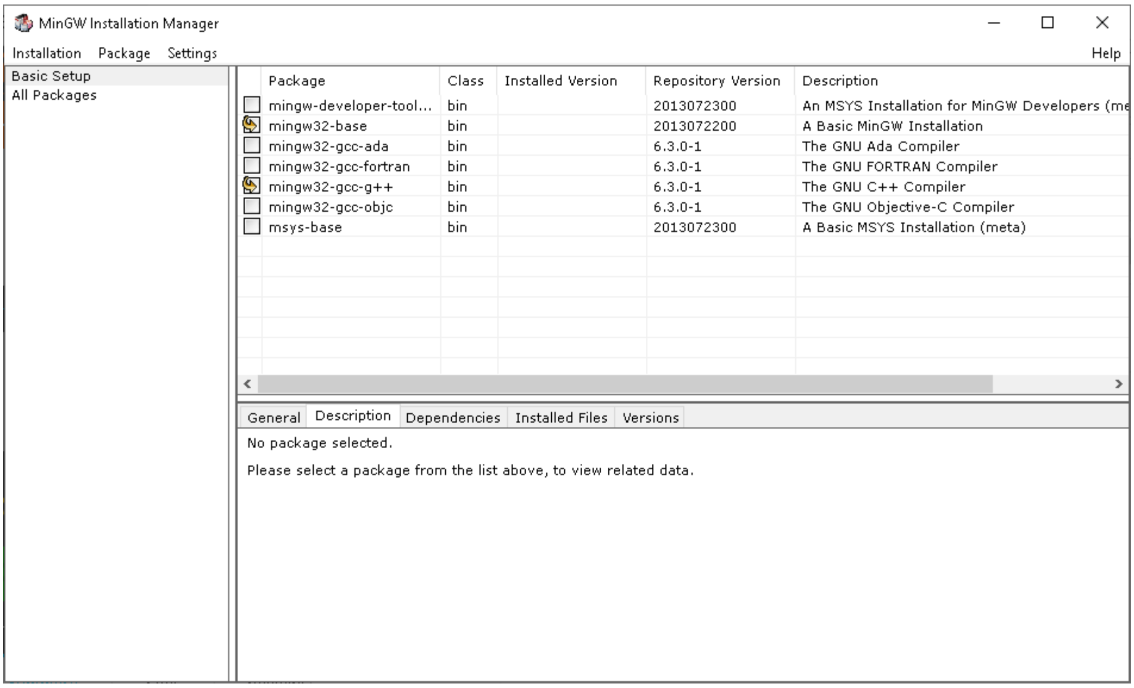
2. Installing IsletLab in Windows

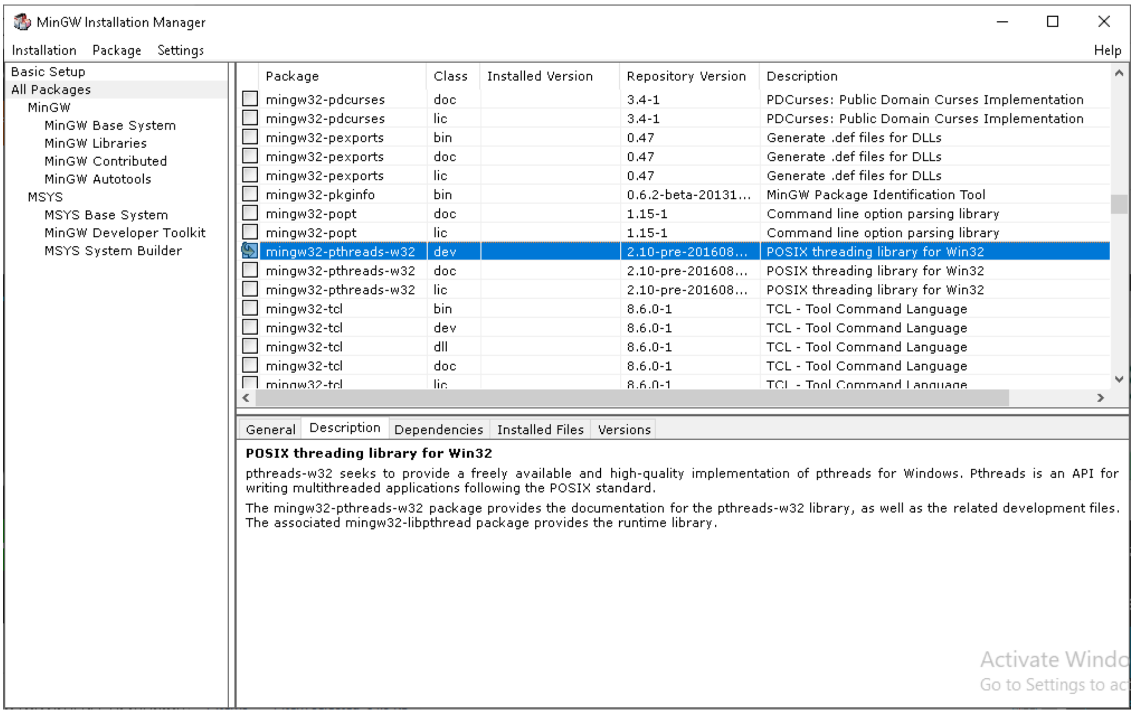
Third-party compilers are required:

- GCC (MinGW)
- NVCC
- MSVC (Cl.exe)

First open a Command Prompt and type the following commands: `gcc --version` , `nvcc --version` and `cl.exe`. If any of these commands is not recognized by the system, follow the corresponding steps below to install it and configure it before using IsletLab.

1. **GCC**. Download and install MinGW from https://sourceforge.net/projects/mingw/ (a GNU Compiler Collection needed by IsletLab). Be sure to mark for installation the **mingw32-base** and the **mingw32-gcc-g++** packages from the **Basic Setup** list and the **mingw32-pthreads-w32** package (dev) from the **All Packages** list. Once selected, go to the **Installation Menu** and select **Apply Changes**. Click the **Apply** button to finalize the installation.

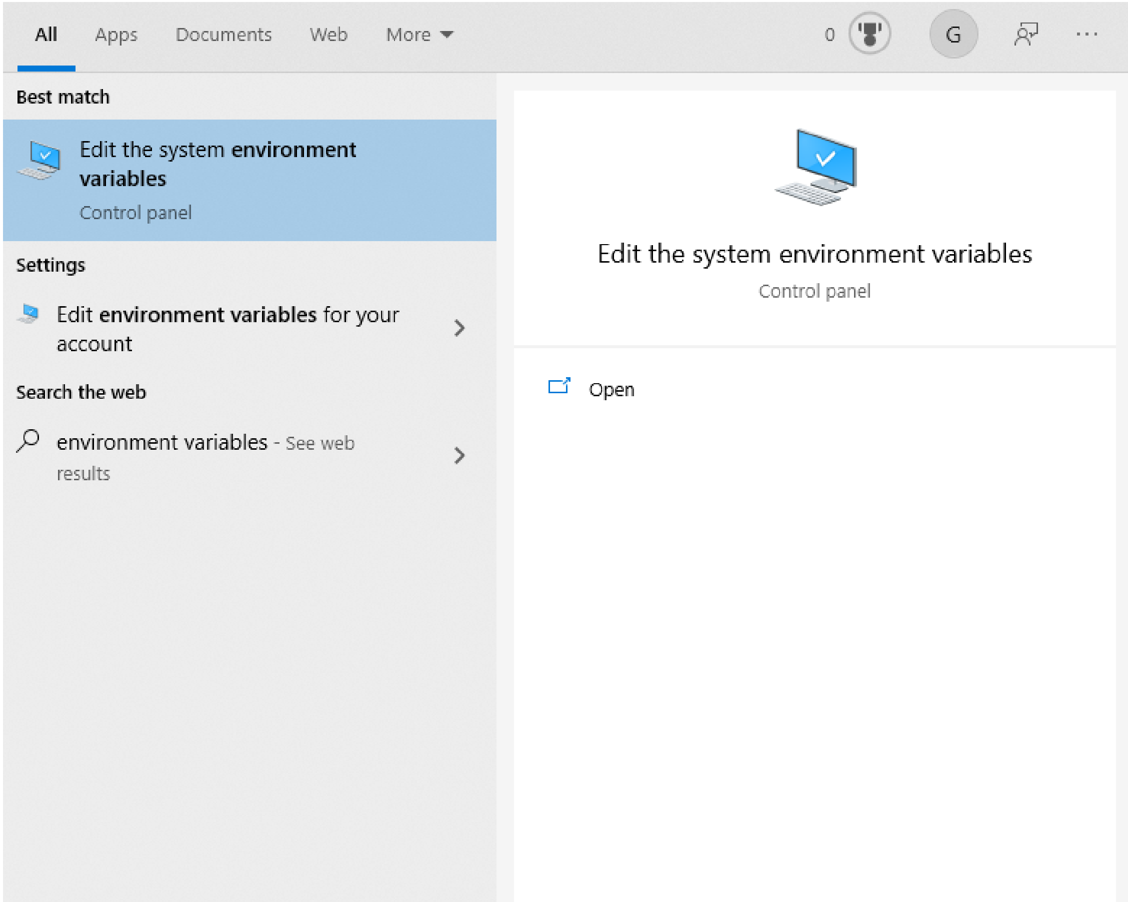




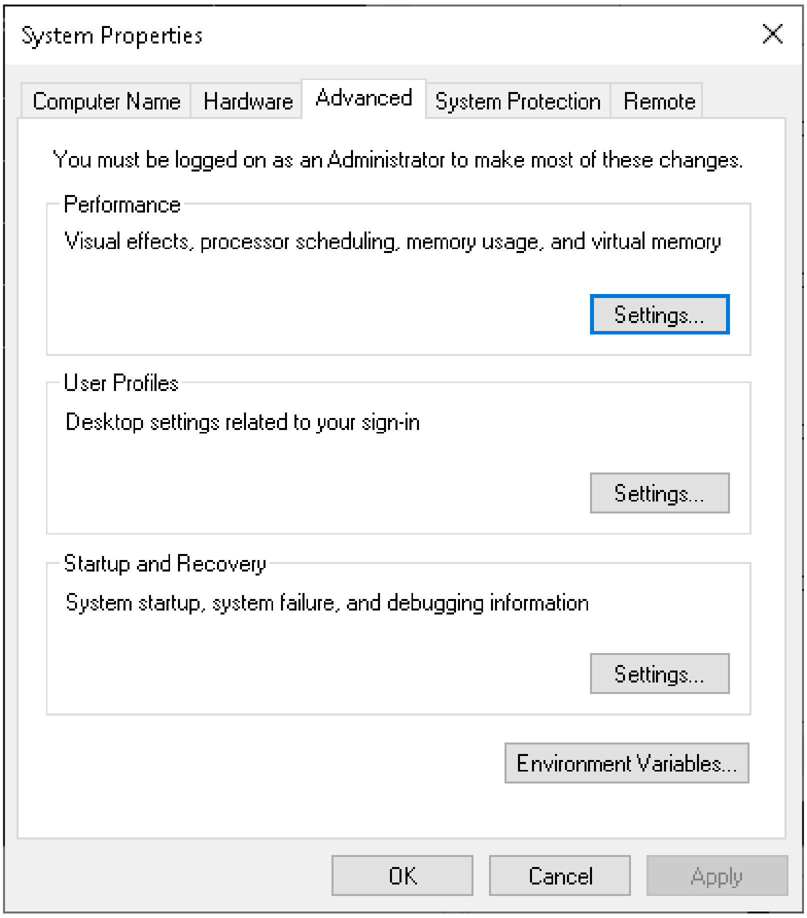
2. **MSVC**. [Download and install the **Build Tools for Visual Studio** package](). Be sure to select **Desktop development with C++** and click **Install**.
3. **NVCC**. [Download and install the CUDA Toolkit]().

Now, the paths to these compilers must be added to the Environment Variables:

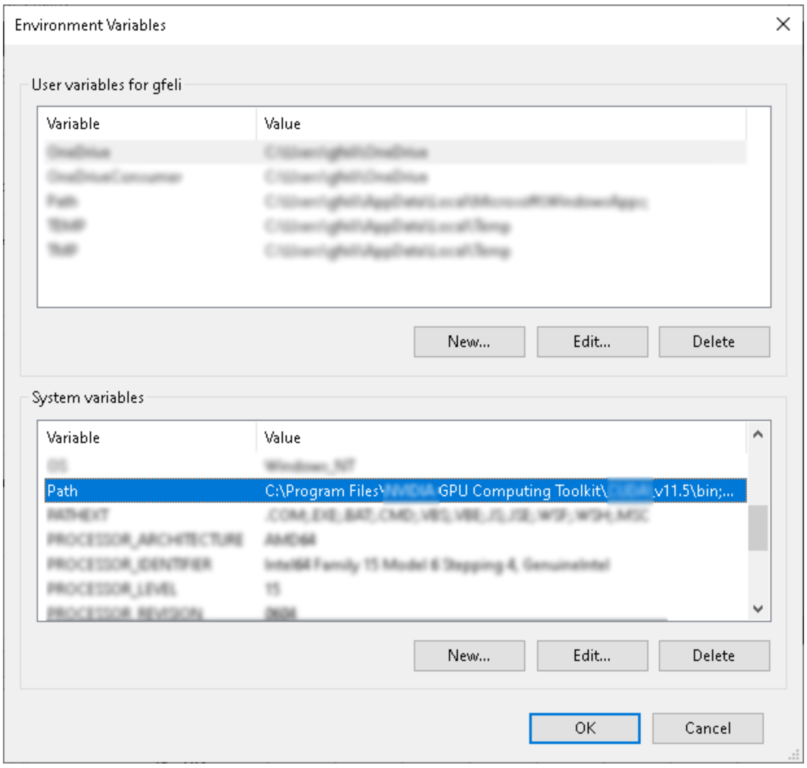
4. Search for **Environment Variables** in the search bar and select **Edit the system environment variables**.



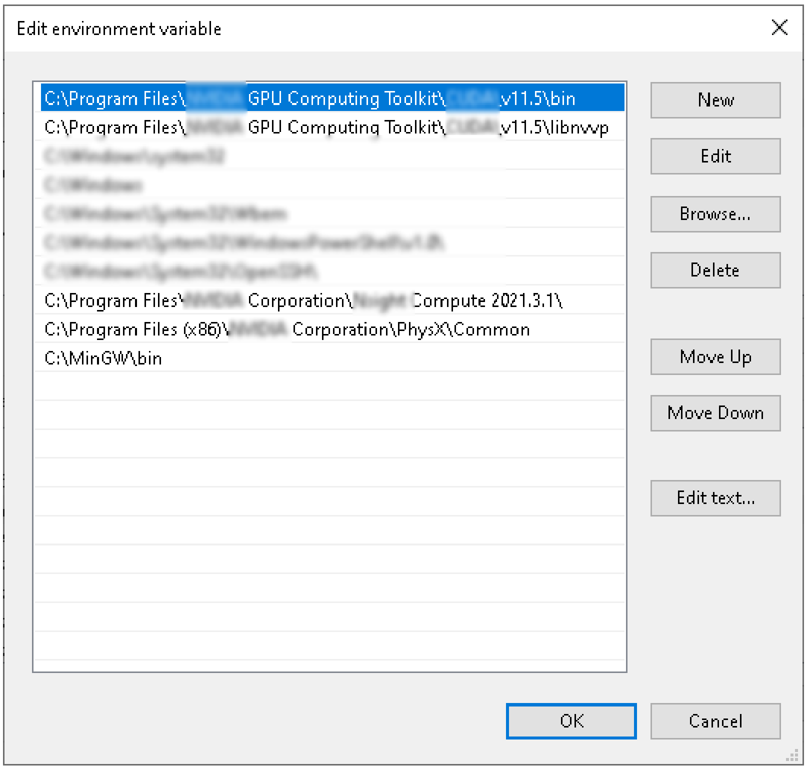
- Click the **Environment Variables** button.



- Look for the **Path** variable in the **System variables** box and click **Edit**.



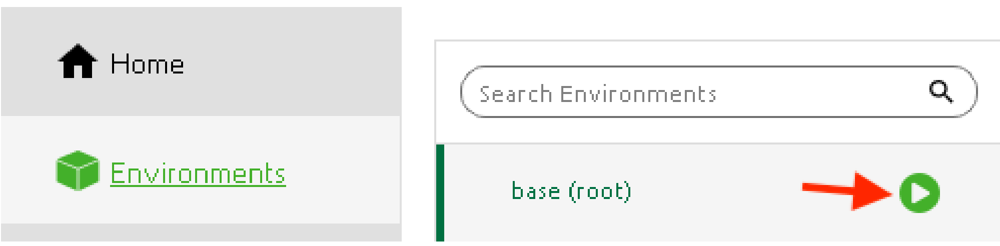
- Click on an empty row and then click **Browse**. Go to the **bin** folder inside the **MinGW** folder (installed usually in C:\MinGW\bin). Click **OK**.



- Click **Browse** again and look for the **cl.exe** file, commonly located in the Visual Studio Folder (for instance, **C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools\VC\MSVC\14.3030705\bin\Hostx64\x64\cl.exe**, although this path could be different. In such case, look for the folder containing the **cl.exe** file). Click **OK** and close the **Environment Variables** window.
- Verify that the paths to the GPU Toolkit directories are listed (first to rows in the image above). Otherwise, [download and install the Toolkit]() (if a capable device is available in your computer).

  Note that these paths could vary depending on the version, installation path, operative system version, etc.

4. [Download and install Anaconda]() (Python 3.8 or 3.9 will work).
5. Reboot your computer.
6. Download the **Application file** or clone the IsletLab repository from https://github.com/gjfelix/IsletLab. If you download the repository as a zip file, extract the files.
7. Open the **Navigator**, select **Environments** from the left panel, **click on the "Play" button** of the base (root) environment, and **select Open Terminal** to open a Command Prompt.



8. In the command prompt, go to the repository folder (where you extracted the IsletLab repository files). There should be a file named **isletlabgui_v1.0.yml**.
9. Create a conda environment using the **isletlabgui_v1.0.yml** file. All the python modules needed will be installed automatically.

```
conda env create -f isletlabgui_v1.0.yml
```

10. Close the Command Prompt and go back to the enviroments window. You should see the environment called **isletlab_v1.0** listed in the Environments tab (see the image below). Launch the isletlab_v1.0 environment by clicking the **Play** button and selecting **Open Terminal**.

11. In the Command Prompt, go to the IsletLab repository folder and run the application by typing the following command:

```
python isletlabgui_v1.0.py
```

Note: Functional simulations involve GPU computing which requires a capable GPU. plus the driver and the Toolkit. Check if the NVCC compiler is installed by opening a Command Prompt and typing `nvcc --version`. If the version of the NVCC compiler is not displayed, it is likely that the Toolkit is not installed. If these requirements are not met, you still will be able to reconstruct and analyze pancreatic islets (cell-to-cell contacts, network analysis).

3. Installing IsletLab in macOS

1. Download and install Anaconda.
2. Verify that the gcc-10 compiler is installed. Open a Terminal and enter: `gcc-10 --version` and press Enter. If installed, the version of the gcc-10 compiler must be displayed, otherwise, it must be installed as follows:

- Install MacPorts (https://www.macports.org/install.php).
- Open a Terminal and enter: sudo port install gcc10.
- Verify the installation by entering `gcc-10 --version` in a Terminal.

3. Download the **Application file** or clone the Isletlab repository (https://github.com/gjfelix/IsletLab). If you downloaded the repository as a zip file, extract the files.
4. Open a Terminal in the base environment and go to the repository folder (where the repository files were extracted).
5. Create a new environment using the **isletlabgui_v1.0.yml** file. All the python modules needed will be installed automatically.

```
conda env create -f isletlabgui_v1.0.yml
```

6. Activate the new environment. This step can be perform either typing the following command in the Terminal:

```
conda activate isletlab_v1.0
```

7. Run Isletlab:

```
python isletlabgui_v1.0.py
```

Note: Due to hardware incompatibilities with Isletlab, it is currently not possible to perform functional simulations in macOS.

4. CPU Threads

The number of threads available varies between computers depending on the characteristics of the CPU. Broadly speaking, the number of threads is the number of parallel calculations that IsletLab will use during the reconstruction process. It is really important to determine the maximum number of threads available in order to reduce the computing time. Note that you will need the number of threads available in your computer when performing an islet reconstruction.

- In Windows, in the current version of IsletLab it is only possible to use a single CPU thread. Therefore, the **Threads** parameter must be set to **1**.
- In Linux, open a Terminal and enter the `lscpu` command. Look for the number of **CPU(s)** and the number of **Thread(s) per core**. Then calculate the **total umber of threads** by multiplying the number of CPUs by the number of Thread(s) per core.
- In macOS, run the following command in a Terminal in order to determine the number of logical cores: `sysctl hw.logicalcpu`.

5. Reconstructing an islet

The reconstruction parameters needed for the protocol are related to the optimization algorithm used to reconstruct the islets proposed by Félix-Martínez et al. (Félix-Martínez, Gerardo J., Aurelio N. Mata, and J. Rafael Godínez-Fernández. "Reconstructing human pancreatic islet architectures using computational optimization." Islets 12.6 (2020): 121-133.). The interested reader is referred to the article for further details.

First, an input data file must be provided by the user. The input file must be composed of four columns with the cell types in column 1, the spatial coordinates (X, Y, Z) in columns 2-4, and a single row for each cell in the islet as in the example given below:

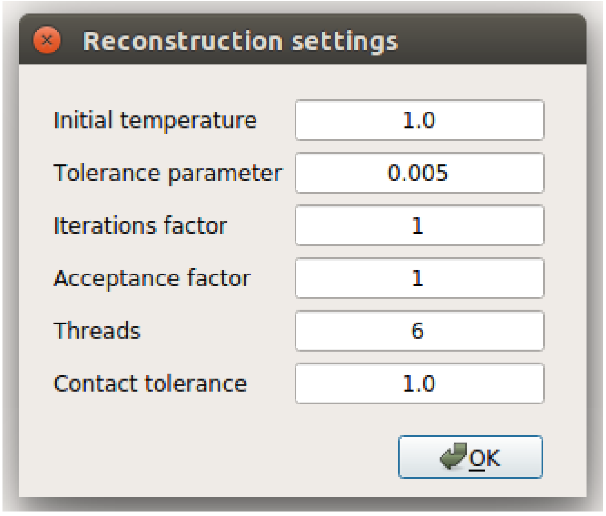
```
11    172.22000   -144.21000     -1.00000
12    165.72000   -136.07000     -4.00000
13    155.32000   -144.21000     -4.00000
.
.
.
```

Based on the input file an initial islet is automatically proposed by assigning initial radii to the islet cells (Islet initialization). Then, the number of overlapped cells in the initial islet is calculated .

Starting from the value of the Initial Temperature parameter defined by the user, the algorithm proposes a new islet and calculate the new number of overlapped cells, accepting the new islet if the number of overlapped cells is reduced, or either accepting it or rejecting it based on a Temperature dependent probability (the probability of accepting an islet with a higher number of overlapped cells decreases as the temperature decreases). The Temperature is reduced by half and the process is repeated until a predefined stop criteria (Tolerance parameter) is met.

Follow this step-by-step guide to reconstruct an islet.

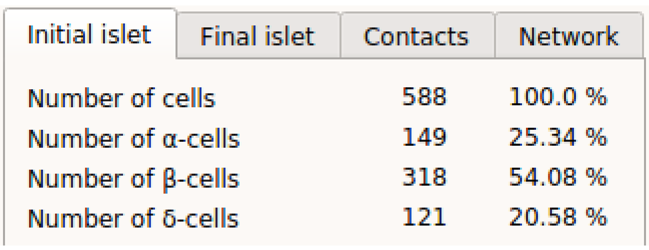
1. Determine how many CPU threads your computer have available.
2. Click the **Reconstruction settings** button and enter the number of CPU threads you want to use in the **Threads** field.



3. Modify the reconstruction parameters (**Initial temperature**, **Tolerance**, **Iterations and Acceptance factors**) if needed and click **OK**. It is worth remembering that the number of new islet tested (i.e. iterations) **for each** value of the Temperature parameter is given either by `Acceptance factor * Number of cells` or `Iterations factor  * Number of cells` (the one reached first). Once the iterations for a given Temperature value are completed, the Temperature value is reduced by half and the process is repeated until a predefined convergence criteria, defined by the **Tolerance parameter**, is reached.
4. Click the **Load initial islet** button and select your input file (If you don't have any data, you can use the Input_test_file.txt file included in the **Application File**). Note that the input file must be composed of four columns: 1. Cell type (coded as 11:α, 12: β, 13: δ); 2-4: nucleus coordinates X, Y and Z, respectively. For instance, an input file of an islet composed of three cells would look as follows:

```
12 172.22 -144.21 -1.00
11 165.72 -136.07 -4.00
13 155.32 -144.21 -4.00
.
.
.
```

   When a valid input file is selected, A 3-D visualization of the initial islet must be shown in the **Initial Islet** tab of the **Plots panel** and the number and percentages of the different cell populations (α, β and δ) must be displayed in the **Initial Islet** tab of the **Statistics panel**.



5. Click the **Reconstruct islet** button. The **Reconstruction Log** should appear. Click the **Run** button to start the reconstruction process and close the **Reconstruction Log** when indicated. For details, see the following section. A 3D representation of the reconstructed islet is then presented in the **Final Islet** tab or the **Plots panel** and the statistics related to the reconstructed islets are shown in the **Final Islet** tab of the **Statistics panel**.

6. Reconstruction Log

The **Reconstruction Log** allows the user to monitor the islet reconstruction process.

The first lines of the **Reconstruction Log** show the IsletLab version and gives the reference to the paper where the details about the reconstruction algorithm can be consulted.

```
IsletLab v.1.0

Pancreatic islet reconstruction based on the algorithm by
Felix-Martinez et al. DOI: 10.1080/19382014.2020.1823178
```

Then, the number of overlapped cells in the initial islet is given (the number to be minimized during the optimization procedure).

```
Overlapped cells in initial islet: 760.000000
```

Afterwards, the CPU threads to be used for the reconstructions are tested and initialized (2 threads used in this example).

```
Initializing thread: 0

Initializing thread: 1
```

Then, information about the reconstruction/optimization process is displayed.

First, the current value of the Temperature parameter (**T**) is given, along with the number of overlapped cells (**OC**) for the current **Temperature** value. It is worth remembering that for each Temperature value several iterations are performed (determined by the **Iterations factor** parameter in the **Reconstruction settings**, see **Section 5**) . The minimal (**min(OC)**) and maximum (**max(OC)**) values of the number of overlapped cells for the current Temperature value are also shown. Finally, the total number of iterations performed (**Total**) as well as the number of iterations accepted (**Accepted**) are given.

This information is displayed for each Temperature value until the convergence criteria is reached. The process stops automatically and the total **Computing time** is shown.

```
T = 1.0000000000
Overlapped cells (OC) = 595.000000
[min(OC) max(OC)] = [594.000000 759.000000]
[Accepted Total] = [438 588]


T = 0.5000000000
Overlapped cells (OC) = 464.000000
[min(OC) max(OC)] = [464.000000 595.000000]
[Accepted Total] = [347 588]

.
.
.

T = 0.0000000009
Overlapped cells (OC) = 96.000000
[min(OC) max(OC)] = [96.000000 96.000000]
[Accepted Total] = [154 588]

Computing time: 76 seconds
```

```
Please close this window to continue.
```

Note that the **Initial Temperature**, the total number of iterations per temperature value and the maximum number of accepted iterations can be modified in the **Reconstruction settings**. See the details in **Section 5**.

7. Reconstruction results

As a result of the islet reconstruction process, IsletLab provides the user with both basic graphical visualizations as well as data files, which can be used to perform further analyses.

Firstly, the IsletLab window shows a 3D representation of the reconstructed islet (**Final Islet** tab in the **Plots panel**), with α, β and δ-cells shown in red, green and blue, respectively. The data behind this visualization is saved in the file "**Filename_postprocessed.txt**", named automatically after the file containing the input data ("**Filename.txt**" in this example). This file contains the cells' radii in column 1, a color value in column 2 (used to visualize the islet), the cells' type in column 3 (coded as 11: α, 12: β and 13: δ cells) and the X, Y and Z coordinates of each cell in columns 4-6. An example of this structure is shown below.

```
4.775805    0.400000    12.000000   172.220000  -144.210000 2.328237
5.274468    0.400000    12.000000   165.720000  -136.070000 -0.570117
.
.
.
4.658718    0.400000    12.000000   159.492397  -144.210000 -4.000000
4.547464    0.400000    12.000000   181.310000  -176.883972 -6.000000
```

Other files related to the reconstruction process are also generated and are briefly described below (assuming that the name of the file containing the initial data is "**Filename.txt**"):

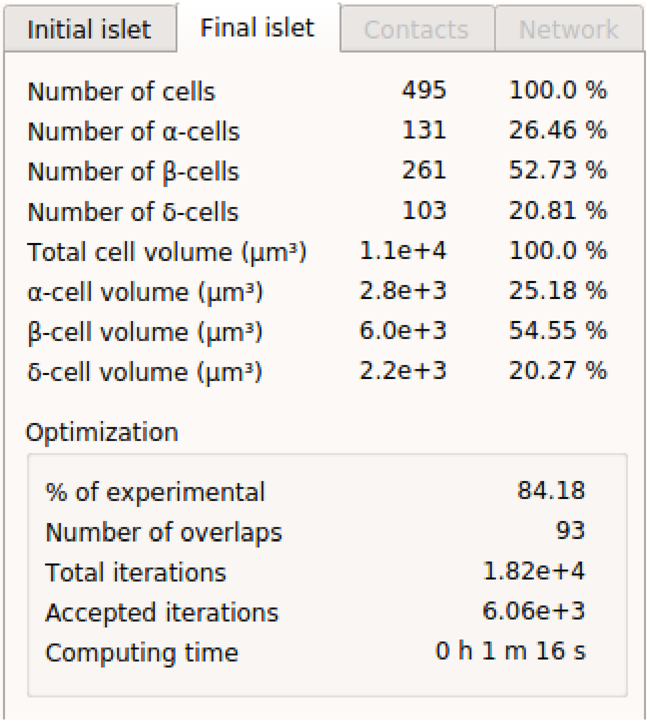
"**Filename_reconstructed.txt**".- This file has the same structure described above, although it also contains the remaining overlapped cells removed during the postprocessing step.

"**Filename_initial.txt**".- This file has the same structure described above and contains the initial islet generated. It is used to generate the corresponding 3D visualization.

"**Filename_overlapped_cells.txt**".- Contains the list of cells of the initial islet deleted during the postprocessing step of the reconstruction.

"**Filename_process_log.txt**".- Contains the information showed to the user in the Reconstruction Log.

Statistics related to the reconstructed islet are shown in the **Statistics panel**:



Number of cells	495	100.0 %
Number of α-cells	131	26.46 %
Number of β-cells	261	52.73 %
Number of δ-cells	103	20.81 %
Total cell volume (μm³)	1.1e+4	100.0 %
α-cell volume (μm³)	2.8e+3	25.18 %
β-cell volume (μm³)	6.0e+3	54.55 %
δ-cell volume (μm³)	2.2e+3	20.27 %

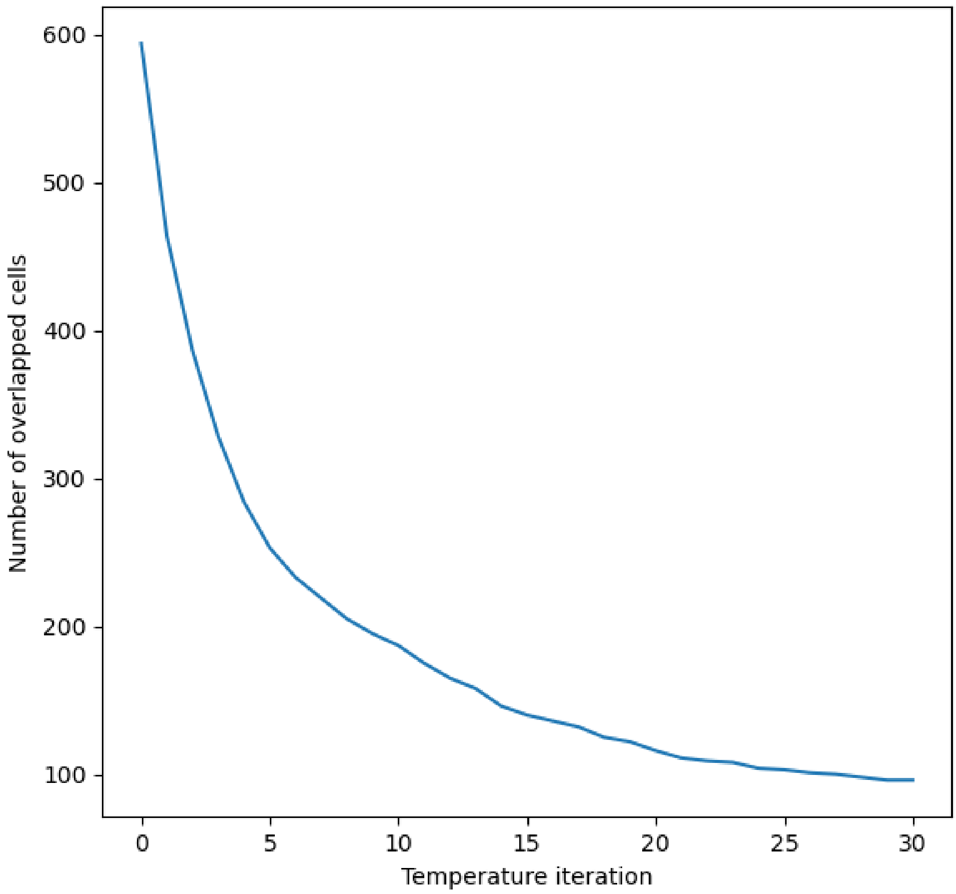
Optimization

% of experimental	84.18
Number of overlaps	93
Total iterations	1.82e+4
Accepted iterations	6.06e+3
Computing time	0 h 1 m 16 s

In addition to the number and percentages of the different types of cells, the cell volume is also calculated. In the bottom part, the optimization stats are displayed, including:

- **% of experimental**. Shows the percentage of the cells of the initial islet included in the reconstructed islet.
- **Number of overlaps**. Indicate the number of cells deleted from the islet during the postprocessing step of the reconstruction algorithm.
- **Total iterations**. Is the total number of iterations performed during the iterative optimization procedure.
- **Accepted iterations**. Indicate the number of iterations accepted during the reconstruction process.
- **Computing time**. Total computing time of the reconstruction.

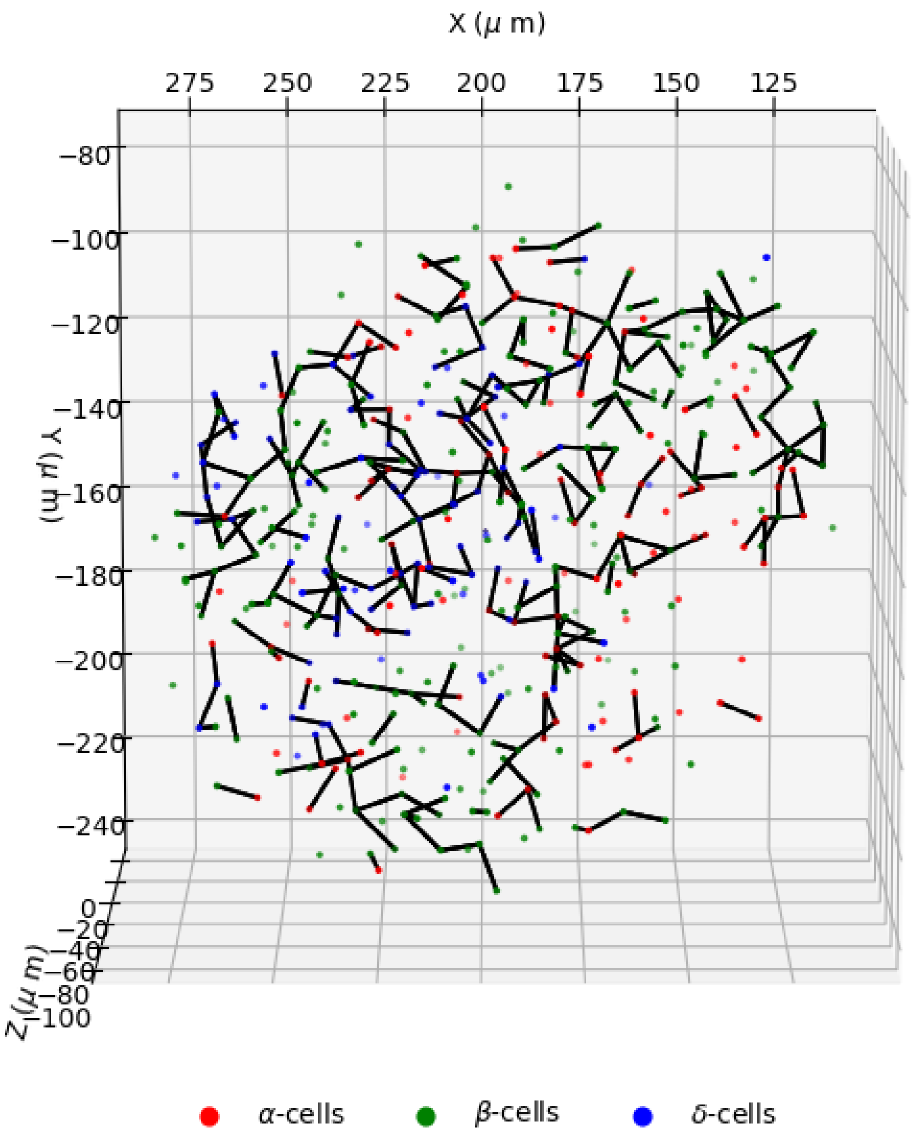
Finally, the **Convergence Plot**, created in the **Plots panel**, reflects the evolution of the number of overlapped cells during the reconstruction process. For instance, in the **Convergence Plot** shown below, the initial islet included nearly 600 overlapped cells while at the end of the reconstruction the reconstructed islet (before the postprocessing step) included ~100 overlapped cells.



This **Convergence Plot** is useful to determine if the **Reconstruction settings** must be modified to improve the reconstruction results.

8. Cell-to-cell contacts results

Once an islet has been reconstructed, cell-to-cell contacts are identified by looking for neighbor cells whose membranes are closer than the distance given by the user in the **Reconstruction settings** panel via the **Contact tolerance** parameter. When the **Cell-to-cell contacts** button is pressed, a graphical representation of the contacts is shown in the **Contacts** tab of the **Plots panel** (see the image below) where only the centers of the α, β an δ cells (red, green and blue, respectively) are shown, and the cell-to-cell contacts are indicated by black lines.



The contacts statistics, that is, the number and percentages of the different contacts and type of contacts, are also shown in the **Statistics panel**.

Initial islet | Final islet | Contacts | Network

Total contacts	257.0	100 %
Homotypic	151.0	58.75 %
Heterotypic	106.0	41.25 %
α - α	31.0	12.06 %
β - β	91.0	35.41 %
δ - δ	29.0	11.28 %
α - β	43.0	16.73 %
α - δ	23.0	8.95 %
β - δ	40.0	15.56 %

Note that **homotypic contacts** include all the contacts between cells of the same type (α-α, β-β, δ-δ), while the **heterotypic contacts** include all the contacts between cells of different types (α-β, β-δ, α-δ).

Several files related to the identification of **Cell-to-cell contacts** are created: **Filename_all_contacts.txt**, **Filename_aa_contacts.txt**, **Filename_ab_contacts.txt**, **Filename_ad_contacts.txt**, **Filename_bb_contacts.txt**, **Filename_bd_contacts.txt**, **Filename_dd_contacts.txt**, **Filename_bbbd_contacts.txt** are the files where **Adjacency**

matrices are saved either for further analysis by the user or to perform **Functional simulations** in IsletLab. All these files share the same structure. For instance, imagining an hypothetical islet composed of 5 cells, the contacts data files would contain a 5 by 5 matrix as:

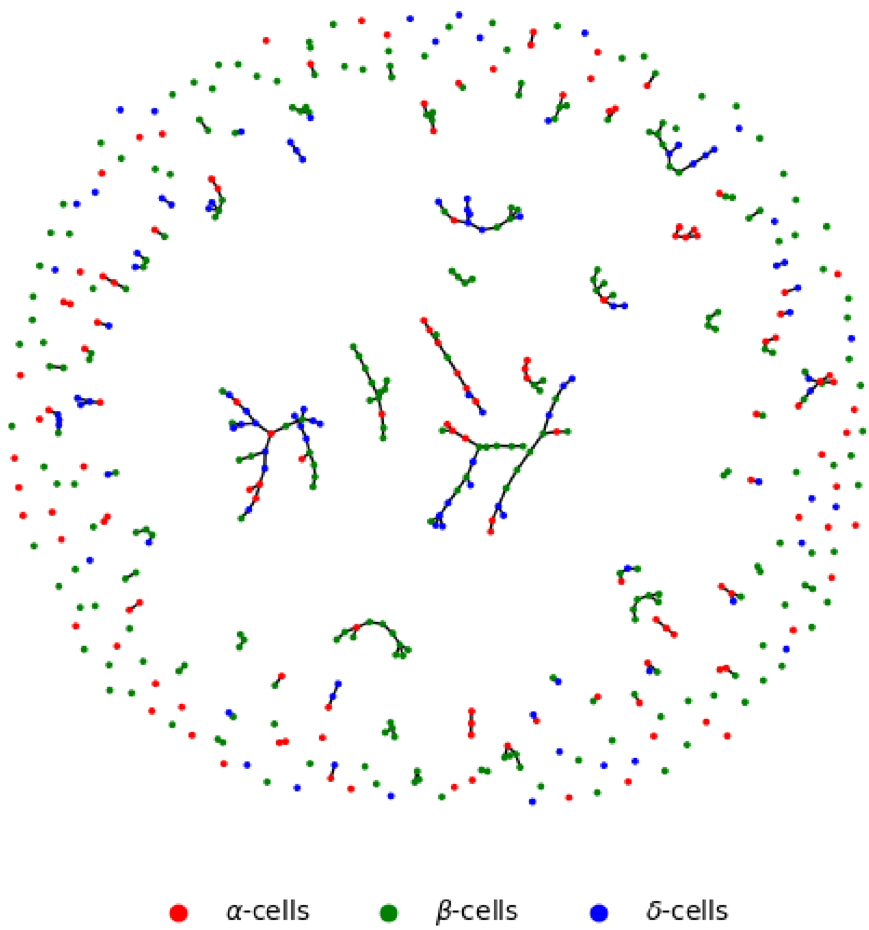
```
0 1 0 0 1
1 0 1 1 0
0 1 0 0 0
0 1 0 0 1
1 0 0 1 0
```

Each row and column of this matrices represent the contacts between all the cells in the islet. For instance, cell 1 (row 1 or column 1) would be in contact with cell 2 and cell 5; cell 3 (row 3 or column 3) would be in contact only with cell 2, etc. Note that the type of cells are obtained from the files created during the reconstruction process.

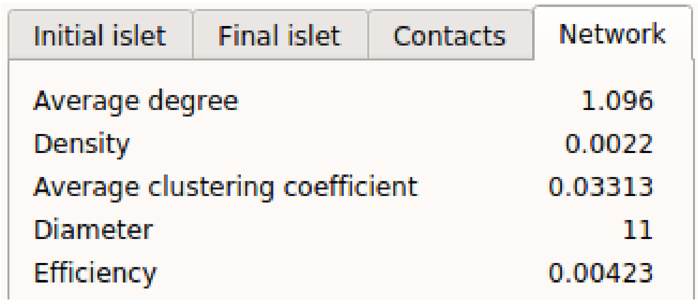
9. Islet network results

When the islet network is generated from the **cell-to-cell contacts**, a 2D visualization of the network is shown in the **Network** tab of the **Plots panel** where α, β and δ cells are shown in red, blue and green, respectively.

An example of an islet network is shown below:



In addition to the network plot, related metrics described below are shown in the **Statistics** panel:



Average degree. It's the average number of links per node in the islet network. In this context, each cell in the reconstructed islet is a node and a link is formed between cells in close contact.

Density. It is a measure of connectedness of the islet network calculated as the ratio of cell-to-cell contacts to all the possible contacts.

Average clustering coefficient. It is a measure of interconnection of the cells' neighborhood.

Efficiency. It is a measure of global integration of the network.

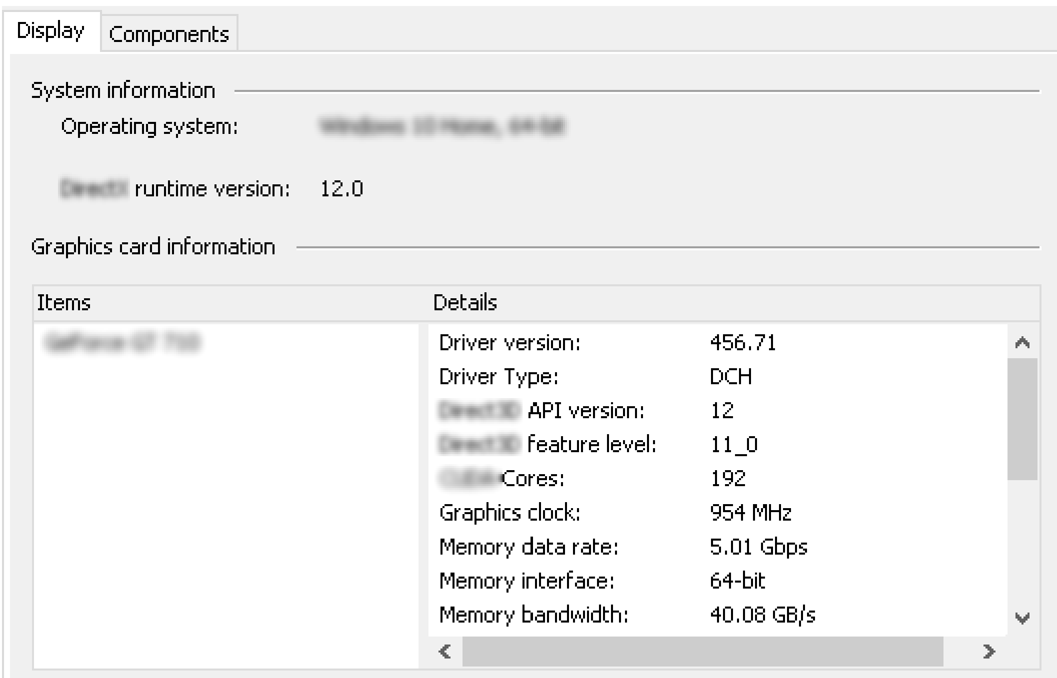
Diameter. It's a measure of the network's size and it's given by the longest short path between all the nodes in the islet network.

More details about the network metrics in the context of islet networks can be found in [Félix-Martínez, Gerardo J., and J. R. Godínez-Fernández. "Comparative analysis of reconstructed architectures from mice and human islets." Islets 14.1 (2022): 23-35.].

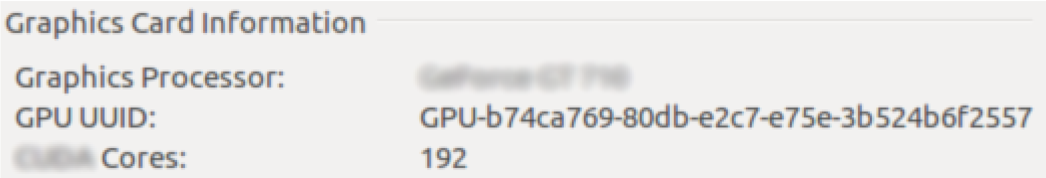
10. GPU blocks and threads

When performing functional simulations it is necessary and extremely important to know how many cores the GPU has available. Determining the number of GPU blocks and threads is a complex subject and directly related to the GPU hardware characteristics. In the following you will find instructions to obtain the number of GPU Cores available.

- In Windows, open the **NVIDIA Control Panel**, then click in **System Information** in the bottom left corner and look for the number of cores (192 in the image below). It is a complex topic to determine the number of blocks and threads, but you could assume that each block has 32 threads (not ideal) and so determine the number of blocks by calculating the number of blocks necessary to use the number of cores available. In the example shown below, I would use 6 blocks and 32 threads (6 x 32 = 192).



- In Linux, open the **NVIDIA X Server Settings** and select your GPU. Find the number of **Cores** (192 in this case) and calculate the number of blocks and threads as described in the instructions given above.



- As previously mentioned, it is not yet possible to perform functional simulations in macOS.

NOTE: Ideally, the number of blocks and threads per block should be determined in accordance with the characteristics of the GPU available. For instance, with a GPU with 2304 Cores, 36 Multiprocessors (MP) and 64 Cores per MP, a better selection of parameters would be 36 blocks and 64 threads.

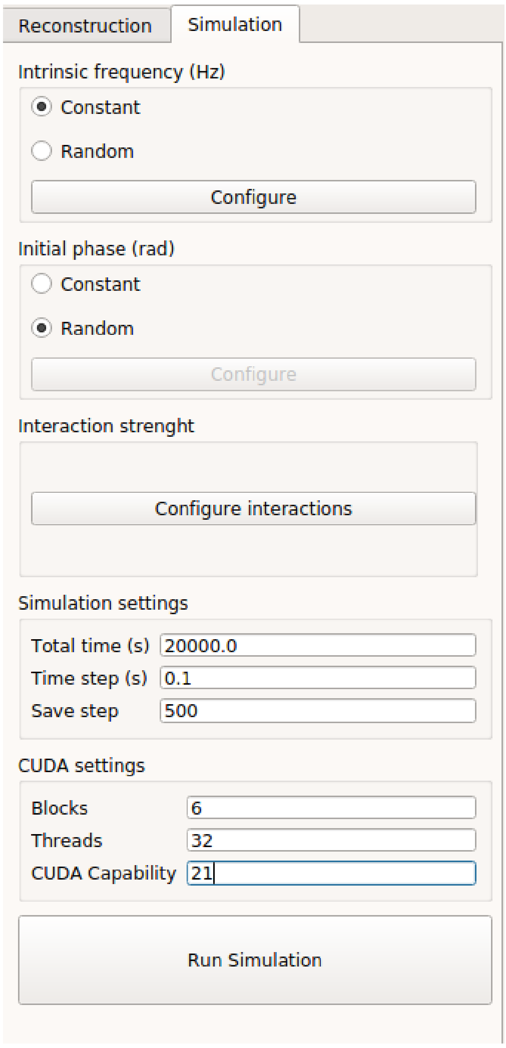
11. Configuring simulations

Functional simulations described in the protocol are implemented by adopting the methodology proposed by Hoang et al. (Hoang, Danh-Tai, Manami Hara, and Junghyo Jo. "Design principles of pancreatic islets: glucose-dependent coordination of hormone pulses." PloS one 11.4 (2016): e0152446.), which treats each cell as an oscillator representing its pulsatile secretory activity. In short, a system of differential equations given by:

$$\frac{d\theta_i}{dt} = \omega_i + \sum_{j\in\Lambda_i} K_{\sigma_i\sigma_j} sin(\theta_j - \theta_i)$$

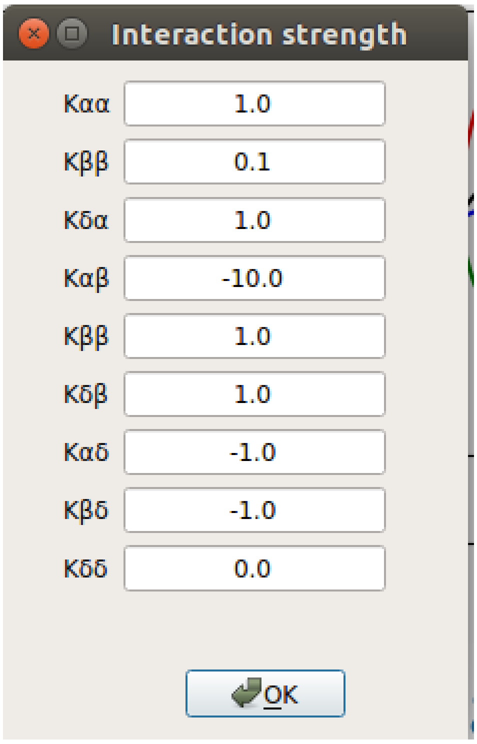
is solved, where *i* represents each cell of the islet, *j* are the cells in contact with cell *i*, θ_i represents the phase of iterator (cell) *i*, ω_i is the intrinsic frequency of oscillator (cell) *i*, $K_{\sigma_i\sigma_j}$ is the interaction strength between cells *i* and *j*, and σ_i and σ_j indicate the corresponding type of cells (α, β, δ). In summary, the phase of each cell of the islet is affected by the phase of the cells in contact with it (i.e. $j\in\Lambda_i$), in accordance with the connectivity given by the reconstruction process.

In order to be solved, the user have to give the **Initial phase** (initial value), the **Initial frequency**, and **Interaction strengths** in the **Simulation tab** of the configuration panel of IsletLab.



When a **Constant intrinsic frequency** is selected, the same frequency is assigned to **all** the cells in the islet. The value of the **intrinsic frequency** is assigned by clicking the **Configure** button. When a **Random intrinsic frequency** is selected, frequencies drawn from a normal distribution are assigned; thus, when the **Configure** button is pressed, the user can enter the mean and standard deviation of the distribution of frequencies.

The **interactions strengths** can be defined by pressing the **Configure interactions** button.



These parameters in practice represent the extent of the influcence of a cell's phase to its neighbor cells' phase. Details about these parameters can be found in the original article by Hoang et al. (Hoang, Danh-Tai, Manami Hara, and Junghyo Jo. "Design principles of pancreatic islets: glucose-dependent coordination of hormone pulses." PloS one 11.4 (2016): e0152446.)

Simulations are preformed using parallel computing through the GPU (graphical processing unit). If a capable graphic card is not available, it will not be possible to perform functional simulations. If a capable graphics card is available, the user needs to define the number of blocks and threads (**Nblocks** and **Nthreads** in the **settings** section of the configuration panel of the **Simulation tab**, see also the GPU blocks and threads section). Note that these parameters depend on the hardware used by the user (see here for a list of capable devices).

12. Simulation Log

The **Simulation Log** (an excerpt is shown below) allows the user to monitor the initialization and evolution of the functional simulations.

First, the islet is initialized by reading and configuring the connectivity properties according to the cell-to-cell contacts previously identified. In the Simulation Log, this is shown as a list of three columns, where each row corresponds to cell of the islets and the first column shows the **Cell id**, the second column the **Number of neighbors** and the third column the **the Neighbors' ID**.

Once the simulations starts, the simulation time is printed in the **Simulation Log** until the **Total time** determined by the user is reached. At the end of the Simulation Log the **Computing time** is displayed.

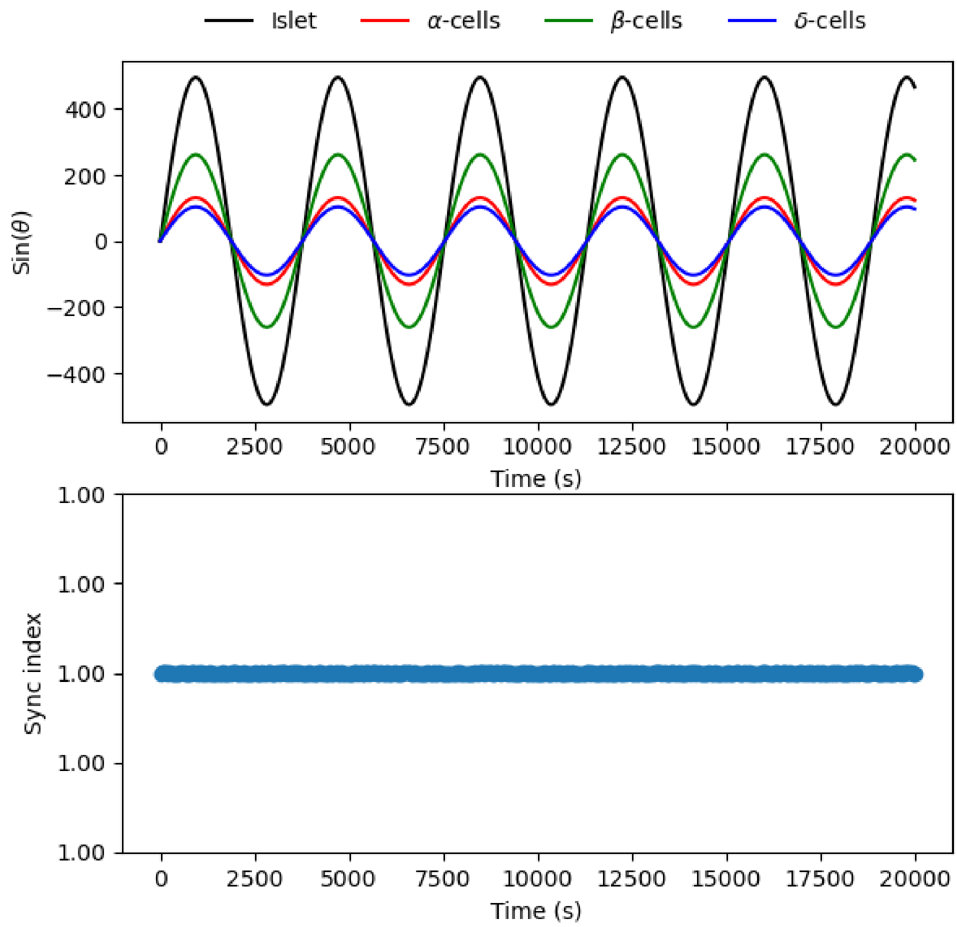
```
IsletLab v1.0


Initializing islet connectivity:

Cell ID: 0    Neighbors: 1   Neighbors ID: 1
.
.
.
Cell ID: 203      Neighbors: 4   Neighbors ID: 204 236 331 462
.
.
.
Cell ID: 332      Neighbors: 3   Neighbors ID: 241 331 333
.
.
.
Cell ID: 494      Neighbors: 1   Neighbors ID: 340



Simulating:

t = 0.000000
t = 50.000000
t = 100.000000
.
.
.
t = 19900.000000
```

```
t = 19950.000000
t = 20000.000000


Computing time: 21 seconds

Please close this window to continue.
```

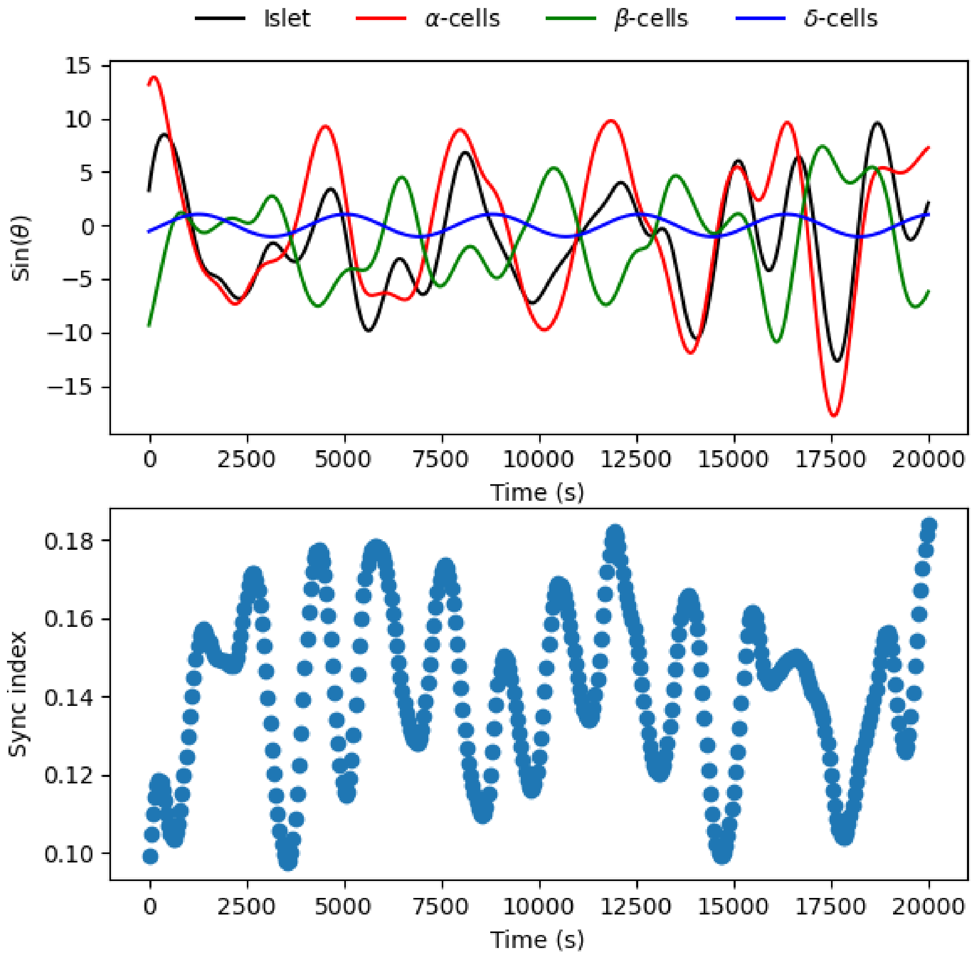
13. Simulation results

Basic visualizations are given as a result of the functional simulations. In the top panel, the summed oscillations of the whole islet (black line), β-cells (green line), α-cells (red line) and δ-cells (blue line) are presented. In the bottom panel, the synchronization index that summarizes the phase coherence of all the cells in the islet is shown. Two different examples are shown below.

First, a simulation performed with **Constant initial frequency** and **Constant initial phase**. Note that all the cells are in phase, and therefore, the synchronization index is equal to 1 during the whole simulation. This is because the interactions between the different type of cells depend on the differences between their phases. Therefore, the synchronization index reflects the phase coherence between all the cells of the islet, having a value of zero when the cells are completely out of phase, and 1, when the cells are completely in phase.



Completely different results are obtained when the **Intrinsic Frequency** is set to **Constant** and the **Initial phase** is set to **random**. In this case, since the initial phase of the cells is defined randomly, and therefore, there is a different in phase between several islet cells, the synchronization index shows a complex behavior, generated by both the phases differences and the interactions between the cells.



In addition to the graphical representations of the results, in the file **File_kuramoto_angles.data** the phases of all the cells in the islets are saved.