Libralis Software Package user's guide

http://libralis.dii.unisi.it

Contents

1. Installation

1.1. Disclaimer

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

1.2. System requirements

- Microsoft Windows (XP/Vista)
- 1 MB free disk space
- Haptik Library 1.0 [1]
- Haptic Device Drivers.

1.3. Downloading

The Libralis Package can be downloaded from the Libralis Project website:
http://libralis.dii.unisi.it

To start using *Libralis* just launch setup.exe and install the application.

Please, be sure that Haptik Library 1.0 (http://www.haptiklibrary.org) and the device drivers are installed.

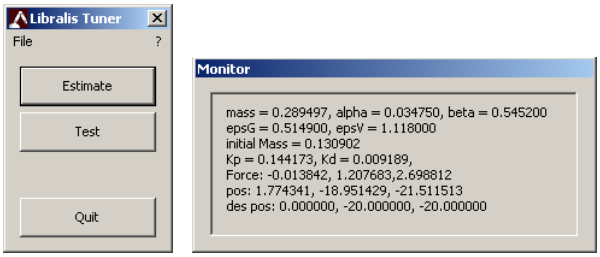
[1] http://www.haptiklibrary.org

2. Introduction to Libralis Software Package

Libralis Tuner allows to perform the gravity estimation with a 3DoF impedance haptic device in a user-defined cubic grid. This software also allows the testing of the performed estimations.

Libralis Library is a *C++* library that allows the developers to simply include the gravity compensation in their interactive applications.

3. *Libralis Tuner*



(a) Main Window. (b) Data Monitor.

Figure 1: Software Main Window.

In Figure 1 is shown the software Main Window, which appears at the program launching. From the dialog on the left (see Figure 1a) the main functions can be chosen.
Estimate runs the offline gravity estimation, Test starts the online gravity compensation used to test the previously estimation.
Quit terminates the application.

The same operations can be performed by the commands on the *File* menu. By the *Help (?)* menu the About window and the Online Help can be opened.

In Figure 1b is shown the Monitor Dialog, which reports technical information during the software execution.

3.1. Offline Gravity Estimation

Click [Estimate] or select the menu *File* → *Estimate* in the Main Window (see Figure 1a) to start the gravity estimation procedure.
Select the haptic device from those available [2] in the Choose Dialog (see Figure 2) and click the [OK] button. The Configuration Dialog (see Figure 4) is shown, from this window all the user-defined parameters [3] can be modified as specified in the following.

3.1.1. Configuration Files

The panel (4.A) allows to manage the configuration files:

[Load] - load the values previously saved or edited from a user-specified *.ini* file.

[Default] - reload the default values.

[Apply] - check and apply the values inserted in the boxes, if they are not valid an error message is shown.

[Save] - check and save the values in a user-specified $.ini$ file.

By default *simple-mode* configuration is selected. If you want use *expert-mode* configuration see Appendix D.

[2] If your device is not shown in the list see the Appendix B1.
[3] Those data are read from the *config.ini* file, to edit this file see Appendix B2.

3.1.2. simple-mode configuration

The box 4.(1).a shows the name of the haptic device in use. Eventually it is possible to select a different device by clicking Change Device.

The box 4.(1).b reports the default maximum force that the haptic device is capable to exert. ***Attention:*** editing improperly this value can lead to actuator saturation, degrading performance.

Select the grid bounding box 4.(2).a and choose the side-lenght of each internal cube 4.(2).b. [4]

Act on the sliders Accuracy and Spring (4.* and 4.*) to set the duration and the accuracy of the estimation. ***Attention:*** note that increasing the desired accuracy will increase also the estimation duration. If the haptic device has low encoder resolution, it will not be able to match high levels of accuracy. This may lead the iterative estimation to get into a deadlock. Moreover, a low spring value means that end-effector motion will be slow and smooth, while high values will result in quick and impulsive motion, eventually causing vibration and actuator saturation.

To complete the set-up, a comment can be inserted in 4.(6). This data are not necessary for the estimate procedure. The text will be eventually saved in the estimate and configuration files and shown during the online gravity compensation.

If some data inserted in the boxes are not correct an error message, which explains the wrong typed values, is shown.

[4] The cube side length influences the number of the vertices to estimate, in Appendix C we can see how a wrong choice for this value can reduce the dimensions of the effectively estimated grid. Another problem connected with the choice of the cube side dimension exists: if it is too big we could meet some trouble during the transition from a vertex to the next one. In this case the haptic device end-effector could be too fast and the device motors could reach the saturation.

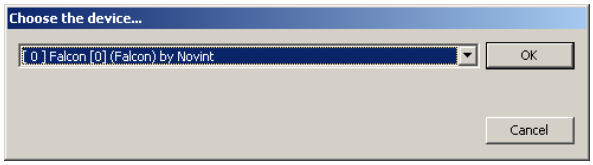


Figure 2: Choose Device Dialog.

Figure 3: Warning message.

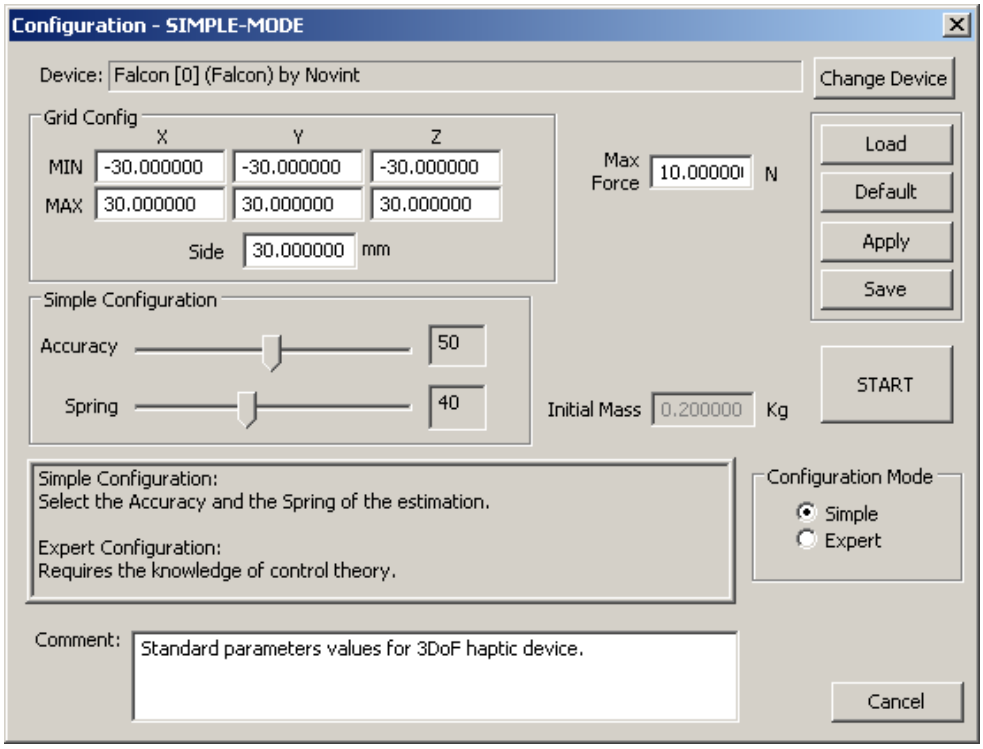


Figure 4: Offline Estimate Configuration Dialog.

3.1.3. Estimation phase

The estimation starts clicking on the Estimate button. The procedure is fully automatic. The first step is the execution of a procedure that calculates some parameters for the estimation algorithm.
During the procedure the Estimate Dialog is shown (see Figure 5). Click Cancel to stop the estimate procedure and go back to the software Configuration Dialog.

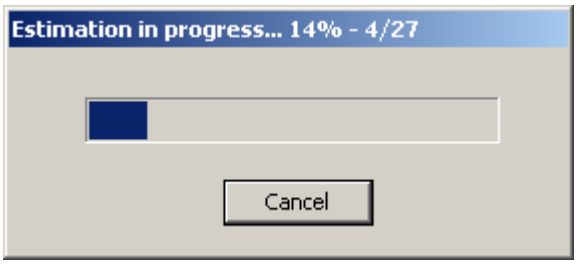


Figure 5: Offline gravity estimation.

The Monitor Dialog (see Fig. 1b) reports information about the device (e.g. the position and the force exerted by the end-effector) and the estimate procedure (e.g. α, β, vertex to estimate, etc.)

At the end of this phase choose the filename *.dat* to save the estimate results.
Now it is possible to test the gravity estimation just performed.

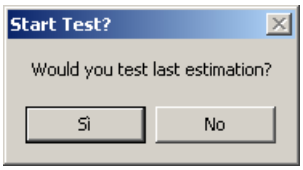


Figure 6: Test after estimation.

If the offline calibration provides accurate estimations in a reasonable time duration, we suggest to save the configuration values used in an *.ini* file with the command Save, eventually typing a reminder comment in the apposite box 4.(6).

3.2. Online Gravity Compensation

This function is used for the *manual* testing of the gravity estimation results.

The testing phase (or online gravity compensation) can be started from the Main Dialog or it can be started once the offline estimate phase has ended and its results have been saved.

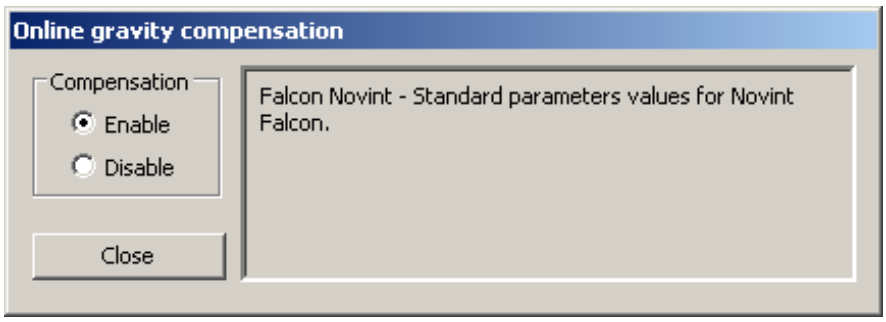


Figure 7: Online Gravity Compensation.

The test dialog is shown in Figure 7, in the box on the right some information about the loaded data are shown (device model, orientation and the saved comment). By clicking on Enable or Disable, we can enable or disable the gravity compensation during the test. The Close button quit the testing phase.

During this phase, the Monitor Dialog (see Figure 1b) reports information such as the end-effector position, the cube detection, the eventually projected position and the gravity term computed at each position of the device workspace.

4. *Libralis Library*

Libralis Library is a static library which allows developers to use online gravity compensation in their haptic applications.

The library exposes only two methods: the first can be used to load gravity data from files and initialize the compensation controller, while the second is used in the haptic servo-loop to compute the desired compensation term at the current end-effector position.

4.1. Getting started

This section illustrate how to get started using Libralis with Haptik Library.

- Add the files libralis.h and libralis.cpp to your existing C++ project.
- In your **Project properties**, add the following path
  [ProgramFolder] \ Siena Robotics and System Lab\Libralis Library\Include
  to **Additional Include Directories**, and the following path
  [ProgramFolder] \ Siena Robotics and System Lab\Libralis Library\Lib
  to **Additional Library Directories**.
- Include the header libralis.h to C++ file in which you have implemented the haptic servo-loop.
- In libralis.h, you have to define the correct path and filename to access the gravity data file, with NO file extension.
- Link the library Libralis.lib to your project.

Please refer to the **sampleLibralis** project, in your Libralis Library folder.

4.2. Extending Libalis to any existing API

Libralis Library can easily be extended to any API used to implement the haptic servo-loop and to access the haptic device. To that end, it is sufficient to edit header libralis.h and the implementation libralis.cpp accordingly. These files originally comes with the inclusion of Haptik Library headers, and use the data type **Vector3** to represent end-effector position and force-feedback vectors.

The default libralis.h file is the following:

```
#include <windows.h>

#include "LibralisFunctions.h"

#include <RSLib\Haptik.hpp>

using namespace RSLib;

using namespace RSLib::Math;


#define GRAVITY_COMPENSATION_FILE __TEXT("./path/filename")

bool InitLibralis();

Vector3 Compensate(Vector3 position);
```

If you desire to replace Haptik with a different API, the file libralis.h should be edited as shown in the following example:

```
#include <windows.h>

#include "LibralisFunctions.h"

#include <myHapticAPI.h>

using namespace my_Haptic_API;


#define GRAVITY_COMPENSATION_FILE __TEXT("./path/filename")

bool InitLibralis();

my_3D_vector_type Compensate(my_3D_vector_type position);
```

Consequently, also the implementation in file libralis.cpp should be edited accordingly:

```
#include "libralis.h"

    ...

my_3D_vector_type Compensate(my_3D_vector_type position)
{
if (Initialized)
{
ProbePose.x = position.x;
ProbePose.y = position.y;
ProbePose.z = position.z;

CompensationTerm = gravityCompensation(ProbePose, fv3Max_WS,
                                       fv3Min_WS, &hCompensationTable,
                                       &hEstimateTable, fSide).gravity;
}
else
{
CompensationTerm.x = 0.0f;
CompensationTerm.y = 0.0f;
CompensationTerm.z = 0.0f;
}

// ***** Replace here the return type *****
//
//
return my_3D_vector_type(CompensationTerm.x,
                         CompensationTerm.y,
                         CompensationTerm.z);

}


bool InitLibralis()
{
    ...
}
```

A License

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

B Configuration Files

B1. Haptik.config.txt

```
Omega.GravityCompensation = FALSE
Omega.Brake = FALSE
Omega.MaxForce = 10.000000
Falcon.GravityCompensation = FALSE
Falcon.Brake = FALSE
Delta.LeftHanded = FALSE
Delta.MaxForce = 10.000000
Delta.MaxTorque = 10.000000
Delta.GravityCompensation = FALSE
Delta.GripperMass = 0.0
Delta.BaseAngleDeg = 0.0
Delta.XOffset = 0
Delta.YOffset = 0
Delta.ZOffset = 0
HaptikLibrary.numberOfPlugins = 6
HaptikLibrary.plugin0_0 = Haptik.Falcon.dll
HaptikLibrary.plugin1_0 = Haptik.Phantom42OH.dll
HaptikLibrary.plugin1_1 = Haptik.Phantom42.dll
HaptikLibrary.plugin1_2 = Haptik.Phantom40.dll
HaptikLibrary.plugin1_3 = Haptik.Phantom31.dll
HaptikLibrary.plugin2_0 = Haptik.Delta30.dll
HaptikLibrary.plugin2_1 = Haptik.Delta31.dll
HaptikLibrary.plugin3_0 = Haptik.Freedom.dll
HaptikLibrary.plugin4_0 = Haptik.Remote.dll
HaptikLibrary.plugin5_0 = Haptik.Spectre.dll
...
```

Figure 8: Content of the *Haptik.config.txt* file.

Please, check the *Haptik.Config.txt* file (see Fig. 8), it is required by Haptik Library[5]. If yours device is supported from Haptik Library, but is not in the list, you must add it to the file by a simple text editor and you must also update the *HaptikLibrary.numberOfPlugins* entry (please refer to Haptik Library User's Guide).

[5] http://sirslab.dii.unisi.it/haptiklibrary/

B2. config.ini

29 1 0	// rows, columns, version		
0.023	// 00 alpha/1000 :: PD stiffness (kp)	0.01	// 15 AlphaMin
0.37	// 01 beta/1000 ::PD damping (kd)	.5	// 16 BetaMax
10	// 02 max device force [N]	.1	// 17 BetaMin
0.3	// 03 lambda	2	// 18 EpsMMax
40	// 04 WS_X MAX [mm]	.0001	// 19 EpsMMin
40	// 05 WS_Y MAX mm	2	// 20 EpsVMax
40	// 06 WS_Z MAX mm	.0001	// 21 EpsVMin
40	// 07 side [mm]	30	// 22 mass delta [kg]
0.20	// 08 initial standard mass [kg]	0	// 23 ::NOT USED::
.05	// 09 mass accuracy [N]	-40	// 24 WS_X min [mm]
20	// 10 mass samples	-40	// 25 WS_Y min [mm]
.08	// 11 velocity accuracy [mm/s]	-40	// 26 WS_Z min [mm]
150	// 12 velocity samples	50	// 27 Accuracy
.1	// 13 ::NOT USED::	40	// 28 Velocity
0.1	// 14 AlphaMax		Omega - Standard orientation.

Figure 9: Content of the *config.ini* file.

The *config.ini* file contains all the parameter values. The default values are suitable for all the 3DoF haptic devices supported by Haptik, but in the *config* folder are present some *.ini* files specifically designed for the most common devices.

We can load the parameters values from a *.ini* file and these files can be modified by a text editor.

C Grid creation

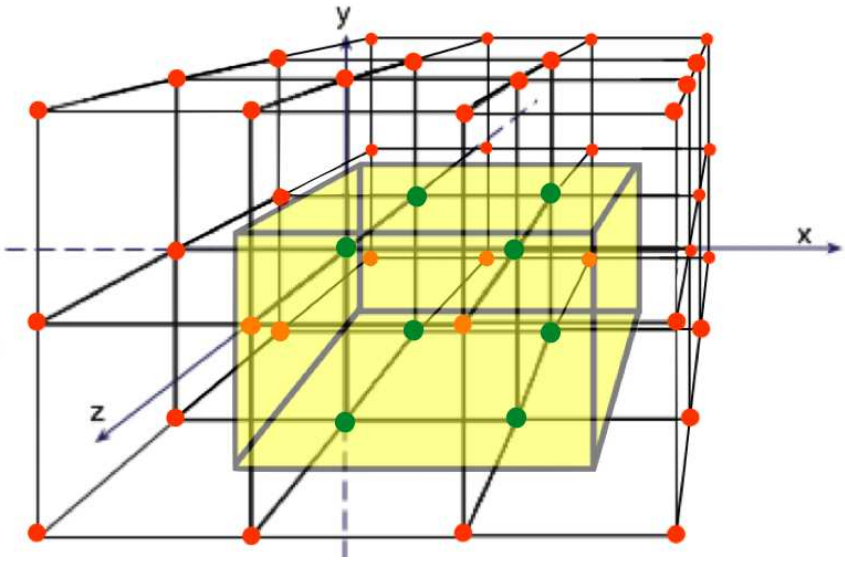


Figure 10: Grid creation.

Libralis Tuner builds a grid starting from a parallelepiped contained in the device workspace and the cube side defined by the user. This cubic grid is the smallest one that contain the parallelepiped and the workspace center $(0, 0, 0)$ is a vertex of this grid.

In the inner vertices (the green ones in Fig. 10) the estimation is performed, instead in the extern vertices (the red ones), gravity and mass are calculated by a function.

To obtain an accurate grid we recommend to choose a value for the cube side that minimize the gap between the parallelepiped and the inner vertices.

In *expert-mode* configuration (see Appendix D) the grid can be loaded from a file.

D Expert-mode configuration

In the Configuration Dialog it is possible choose the *expert-mode* configuration. This method allows to set low-level control parameters. ***Attention:*** editing low-level parameters requires the knowledge of the control architecture described in detail in:

1 A. Formaglio, S. Mulatto, and D. Prattichizzo. *Iterative estimation of the end-effector apparent gravity force for 3DoF impedance haptic devices*. In Proc. EUCA European Conference on Control 2009, Budapest, Hungary, August 2008.

2 A. Formaglio, M. Fei, S. Mulatto, M. de Pascale, and D. Prattichizzo. *Autocalibrated gravity compensation for 3 DoF impedance haptic devices*. Eurohaptics 2008. Lecture Notes in Computer Science. Volume LNCS 5024, pages 43–52. Springer Verlag, Madrid, Spain, June 2008.

In the following, a brief description of the advanced settings is reported. Referring to the Figure 11:

- The grid can be loaded by a file (see Fig. 11.(1)).
- After check the apposite box 11.(2).a, the mass value used for parameters initialization can be set in 11.(2).b. In this way the procedure used to determine a value for the initial mass is skipped.
- The parameters α and β 11.(3) can be set directly. If during the estimation we detect an anomalous behavior of the device (i.e. sudden shots, not accurate vertices etc), we can modify α and β remembering how they influence K_P and K_D, i.e. the parameters of the PD control implemented to drive the haptic device.
- The stop conditions for velocity and gravity in terms of n and ϵ can be exactly defined in 11.(4). Increasing the samples number and decreasing the confidence interval, the estimate accuracy raises, but at the same time also the estimate duration raises. For best set up of those parameters it is needed the knowledge of the haptic encoders accuracy.
- In 11.(5) set the percentage λ of the distance between the casual initial position and the first vertex to estimate to use for the stop condition of the automatic paramenters initialization. If the distance from the initial position and the first vertex is large, lower this value. Instead if that distance is small, increase λ.

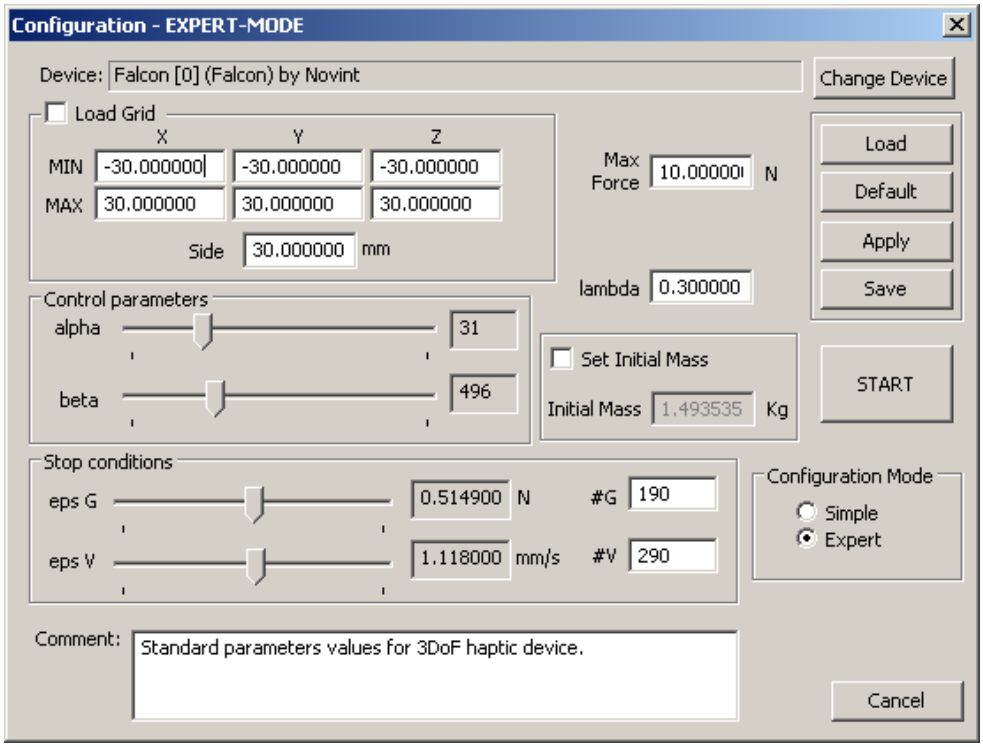


Figure 11: Configuration Dialog - Expert mode.