Optimal Control for Constrained Hybrid System Computational Libraries and Applications

L.T. Paiva [1]

[1] *Department of Electrical and Computer Engineering*
University of Porto, Faculty of Engineering

✍ *Rua Dr. Roberto Frias, s/n, 4200–465 Porto, Portugal*
✉ *ltpaiva@fe.up.pt*
☎ *22 508 1450*

February 28, 2013

Abstract

This final report briefly describes the work carried out under the project PTDC/EEA-CRO/116014/2009 – "Optimal Control for Constrained Hybrid System".
The aim was to build and maintain a software platform to test an illustrate the use of the conceptual tools developed during the overall project: not only in academic examples but also in case studies in the areas of robotics, medicine and exploitation of renewable resources. The grand holder developed a critical hands–on knowledge of the available optimal control solvers as well as package based on non–linear programming solvers.

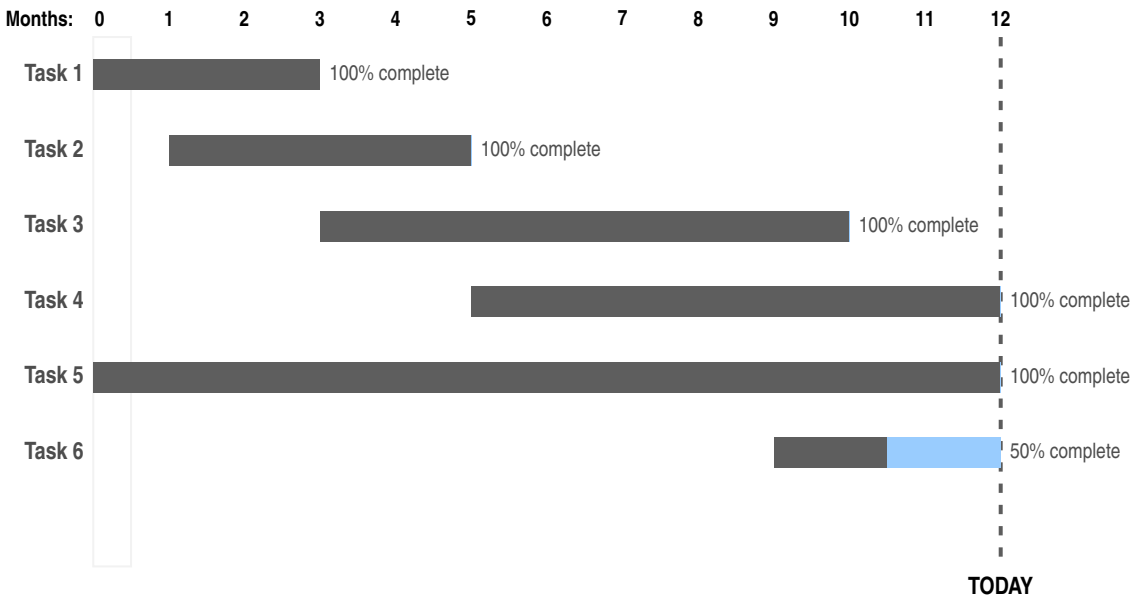
Contents

Work Plan

Project PTDC/EEA-CRO/116014/2009 – "Optimal Control for Constrained Hybrid System"

The main tasks to be carried out are:

1. To understand and obtain a hands–on knowledge of non–linear programming models, current tools to address them (AMPL, MATLAB, . . .) and current solvers (KNITRO, WORHP, SNOPT, . . .). Comparative study of different solvers.
2. To understand optimal control problems and how to transcribe them into NLP to be implemented in AMPL or MATLAB.
3. To obtain a hands–on knowledge of current optimal control packages such DIDO, ACADO, OCPID-DAE, ASTOS, BOCOP and others. To perform a comparative study.
4. To help to implement and solve case studies and tests to illustrate theoretical developments.
5. To maintain the project web–page and documentation accessible to the project team.
6. (If time permits) To construct a platform to easily solve sequences of optimal control problems in a receding horizon fashion in order to implement model predictive controllers.



Progress reports on tasks just concluded and ongoing tasks should be produced at the end of months 3, 5, 7, 10 and 12.

Chapter 1

Optimal Control and Nonlinear Programming Interfaces

1.1 Introduction

Optimal control and nonlinear programming interfaces are software used to solve optimal control problems. A interface is a software that help us to prepare all data as input to the solver. We can divide the interfaces in two groups: optimal control interfaces and nonlinear programming interfaces. The first ones handle with the discretisation and transcription procedures, while the other ones leave this work to the programmer/researcher (see Figure 1.1). Several interfaces will be presented in this chapter, evolving open–source, freeware and commercial software, working under different operating systems.

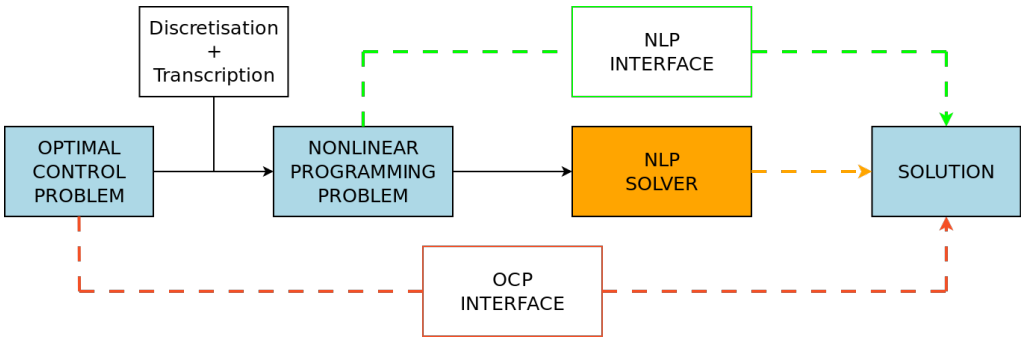


Figure 1.1

1.2 AMPL

"AMPL – A Mathematical Programming Language – is a comprehensive and powerful algebraic modeling language for linear and nonlinear optimization problems, in discrete or continuous variables." The AMPL library can be found in the official site[1].

1.2.1 Basic information

Developed by Robert Fourer, David Gay and Brian Kernighan at Bell Laboratories;

[1] `http://www.anmpl.com`

Operating system(s): AMPL can be used on Linux, Unix, Mac OS X and Microsoft Windows;

Code language(s): AMPL is written in C++ and it is released as open source code under the Eclipse Public License (EPL).

1.2.2 Installing AMPL on Linux

Download your AMPL executable by selecting the Intel (Pentium-compatible) PCs running Linux on AMPL website[2] or type in a Terminal window:

```
wget http://netlib.sandia.gov/ampl/student/linux/ampl.gz
```

The downloaded file must be decompressed and it must be made executable, after which it may be executed from any directory in your search path.

```
gzip -d ampl.gz
chmod +x ampl
sudo mkdir -p /opt/ampl
sudo mv ampl /opt/ampl/
echo 'EXPORT PATH=${PATH}:/opt/ampl' >> ${HOME}/.bashrc
```

To complete your AMPL installation, you can also download solvers to a directory in your search path. To use a downloaded solver, set AMPL's solver option to the name of the solver's executable file.

The student edition of AMPL is limited to 300 variables and 300 constraints and objectives (after presolve) for nonlinear problems.

1.2.3 Getting started with AMPL

The first step in solving a problem is to load the model: variables, objective function and constraints. It is usual the create a `.mod` file containing these informations. First open a AMPL command window by typing `ampl` in a Terminal window.

AMPL commands end with semicolon (`;`). Some AMPL basic commands:

- to declare a parameter: `param <par_name>;`
- to load a model: `model <file_name>.mod;`
- to run a AMPL script: `include <script_name>.run;`
- to load a data file: `data <file_name>.dat;`
- to view the content of a variable: `display <var_name>;`
- to exit a AMPL command window `exit;`

1.3 ACADO – Automatic Control And Dynamic Optimization

"ACADO Toolkit is a software environment and algorithm collection for automatic control and dynamic optimization. It provides a general framework for using a great variety of algorithms for direct optimal control, including model predictive control, state and parameter estimation and robust optimization." The ACADO library can be found in the official site[3].

[2] `http://netlib.sandia.gov/ampl/student/linux/ampl.gz`
[3] `http://sourceforge.net/projects/acado/`

1.3.1 Basic information

Developed under the direction of Moritz Diehl;

Operating system(s): ACADO can be used on Linux, Unix, Mac OS X and Microsoft Windows;

Code language(s): ACADO Toolkit is implemented as self-contained C++ code, it comes along with user-friendly MATLAB interface and is released under the LGP License.

1.3.2 Installing ACADO on Linux

In order to install ACADO Toolkit under LINUX/UNIX operating systems make sure that a C++ compiler is correctly installed. For the case that a graphical output is desired, Gnuplot should be installed in addition. However, ACADO Toolkit will also run without Gnuplot - in this case the graphical output is simply turned off. There are no further dependencies.

First, download ACADO Toolkit and its licence from the offical website[4] and extract the files:

```
tar xfvz ACADOtoolkit-1.0.2613beta.tar.gz
sudo mv ACADOtoolkit-1.0.2613beta /opt/acado
```

Then, go to the directory `/opt/acado` and compile the package:

```
cd /opt/acado
sudo make
sudo make doc (optional)
```

Finally, check whether the installation was successful:

```
cd examples/getting_started
./simple_ocp
```

The following documentation[5] is available:

- ACADO Toolkit User's Manual
- ACADO for Matlab User's Manual
- ACADO Toolkit Introductory Talk

1.4 BOCOP – The optimal control solver

"The BOCOP project aims to develop an open-source toolbox for solving optimal control problems." The BOCOP library can be found in the official site[6].

1.4.1 Basic information

Developed by V. Grelard, P. Martinon and F. Bonnans at Inria-Saclay;

Operating system(s): BOCOP is available for linux precompiled packages (Mac and Windows versions are still under testing);

Code language(s): BOCOP requires SciLab and it is released under the Eclipse Public License (EPL).

[4] http://www.acadotoolkit.org/download.php
[5] http://sourceforge.net/p/acado/wiki/Documentation/
[6] http://www.bocop.org/

1.4.2 Installing BOCOP on Linux

In order to install BOCOP software under LINUX/UNIX operating system, download BOCOP software from the official website[7] and extract the files:

```
tar xvf bocop-1.0.3-beta-linux32.tar.gz
mv bocop-1.0.3-beta-linux32 /opt/bocop
cd /opt/bocop
make
sudo chmod +x bocop.sh
```

Add the following to your `/.bashrc` or `/.bash_profile`:

```
BOCOPDIR=/opt/bocop
export PATH=${PATH}:${BOCOPDIR}
```

1.4.3 Getting started with BOCOP

Read the user's guide[8] and run the examples shown in the `/opt/bocop/problems` folder.

1.5 DIDO – Automatic Control And Dynamic Optimization

"DIDO, the leading optimal control software, powers users by offering the easiest and direct solutions to the most complex problems." The DIDO software can be found in the official site[9].

1.5.1 Basic information

Developed at Elissar Global;

Operating system(s): DIDO can be used on Microsoft Windows;

Code language(s): DIDO requires MATLAB and it is released under academic and commercial licenses.

1.5.2 Installing DIDO

DIDO's foundation is pseudospectral theory and is the only pseudospectral solution with mathematically proven convergence properties. DIDO is a MATLAB program for solving hybrid optimal control problems. The general–purpose program is named after Dido, the legendary founder and first queen of Carthage who is famous in mathematics for her remarkable solution to a constrained optimal control problem even before the invention of calculus.

To install DIDO on your computer you should consider the system requirements:

- WINDOWS
- MATLAB

and the following steps:

[7] `http://www.bocop.org`
[8] `http://bocop.saclay.inria.fr/?download=10`
[9] `http://www.elissarglobal.com/industry/products/software-3/`

STEP 1: Obtain a license for DIDO from Elissar Global at the official site[10].

STEP 2: After you receive an email with the link to a compressed version of DIDO, download the `.zip` file and uncompress it. It will automatically uncompress with the right folder structure.

STEP 3: Right click on the DIDO icon and click on PROPERTIES. In the Target box, type location of your licensed MATLAB file. In the Start in box, type the full path name of your DIDO folder.

STEP 4: Double–click on the DIDO ICON (this should start MATLAB and automatically initialize DIDO).

(a) In the MATLAB command window type `TestDIDO`.
A successful installation should produce the final output on the screen:
"CONGRATULATIONS! DIDO TEST WAS SUCCESSFUL."

(b) A further test may be performed by typing `LanderProblem`.
A successful installation should produce a pretty graph.

1.5.3 Getting started with DIDO

In the MATLAB command window run:

```
[cost, primal] = dido(Problem)
```

where `Problem` is a structure array with fields cost, dynamics, events and path that point to the input files:

```
Problem.cost = 'MyCostFile';
Problem.dynamics = 'MyDynamicsFile';
Problem.events = 'MyEventsFile';
Problem.path = 'MyPathFile';
```

The variable `primal` is used in nearly all DIDO files to pass the states, controls, parameters and other useful quantities. To print out the state– and control–trajectory, in the MATLAB command window run:

```
primal.states primal.controls
```

This generality allows for:

- Fairly complex interior point constraints;
- Pre-defined segments;
- Differentially-flat segments;
- Transition conditions;
- Mid–maneuver changes in dynamics;
- Multi-dynamical systems;
- Mid–maneuver changes in the cost function;
- Switches;
- Discrete events.

[10] `http://www.elissarglobal.com/academic/get-dido/try-dido/`

1.6 ICLOCS – Imperial College London Optimal Control Software

"The code allows users to define and solve optimal control problems with general path and boundary constraints and free or fixed final time. It is also possible to include constant design parameters as unknowns." The ICLOCS software can be found in the official site[11].

1.6.1 Basic information

Developed by Paola Falugi, Eric Kerrigan and Eugene van Wyk;

Operating system(s): ICLOCS can be used on Linux, Unix, Mac OS X and Microsoft Windows;

Code language(s): ICLOCS is implemented in MATLAB and it is released as open source code under the BSD License.

1.6.2 Installing ICLOCS on Linux

In order to install ICLOCS software under LINUX/UNIX operating system, download ICLOCS software from the official site[12] and extract the files:

```
wget http://www.ee.ic.ac.uk/ICLOCS/ICLOCS_0.2_NI.tar.zip
unzip ICLOCS_0.2_NI.tar.zip
tar xvf ICLOCS_0.2_NI.tar
sudo mv ICLOCS_0.2_NI /opt/iclocs
```

In order to use it in MATLAB, you need to tell MATLAB where to find it. To do this just type `addpath(genpath('/opt/iclocs/'))` in the MATLAB command window.

1.6.3 Getting started with ICLOCS

Read the user's guide[13] and run the examples shown in the `/opt/iclocs/examples` folder.

1.7 TACO – Toolkit for AMPL Control Optimization

"TACO is the Toolkit for AMPL Control Optimization. It defines some add-ons to the AMPL modeling language that allow the elegant formulation of ODE/DAE optimal control problems in AMPL." The TACO toolkit can be found in the official site[14].

1.7.1 Basic information

Developed by Christian Kirches and Sven Leyffer ;

Operating system(s): TACO can be used on Linux, Unix, Mac OS X and Microsoft Windows;

Code language(s): TACO is written in C.

[11] `http://www.ee.ic.ac.uk/ICLOCS/`
[12] `http://www.ee.ic.ac.uk/ICLOCS/`
[13] `http://www.ee.ic.ac.uk/ICLOCS/user_guide.pdf`
[14] `http://www.iwr.uni-heidelberg.de/~Christian.Kirches/software.html`

1.7.2 Installing TACO on Linux

In order to install TADO Toolkit under LINUX/UNIX operating systems make sure that CMAKE and a C++ compiler is correctly installed. To install CMAKE type in a Terminal window:

```
wget https://launchpad.net/ubuntu/+archive/primary/+files/cmake_2.8.8.orig.tar.gz
gunzip cmake_2.8.8.orig.tar.gz
tar xvf cmake_2.8.8.orig.tar
cd cmake-2.8.8
sudo ./bootstrap
sudo make
sudo make install
```

First, download TACO Toolkit from the offical site[15] and extract the files:

```
wget https://www.iwr.uni-heidelberg.de/groups/agbock/FILES/taco_source.tar.gz
gunzip taco_source.tar.gz
tar xfvz taco_source.tar
sudo ln -s /opt/CoinIpopt/ThirdParty/ASL/libamplsolver.a libamplsolver.a
```

1.8 Pseudospectral Methods in Optimal Control

1.8.1 PSOPT

PSOPT is an open source optimal control software package written in C++ that uses direct collocation methods, including pseudospectral and local discretizations, available in the office site[16]. Pseudospectral methods solve optimal control problems by approximating the time-dependent variables using global polynomials, such as Legendre or Chebyshev functions. Local discretisation methods approximate the time dependent functions using local splines, and can be seen as implementations of implicit Runge-Kutta integrators. With both global and local methods, differential equations, continuous constraints and integrals associated with the problem are discretised over a grid of nodes. Sparse nonlinear programming is then used to find local optimal solutions.

PSOPT is able to deal with problems with the following characteristics:

- Single or multiphase problems;
- Continuous time nonlinear dynamics;
- Nonlinear path constraints;
- General event constraints;
- Integral constraints;
- Interior point constraints;
- Bounds on controls and state variables;
- General cost function with Lagrange and Mayer terms;

[15] http://www.iwr.uni-heidelberg.de/~Christian.Kirches/software.html
[16] http://www.psopt.org

- Linear or nonlinear linkages between phases;
- Fixed or free initial phase time;
- Fixed or free final phase time;
- Optimization of static parameters;
- Optimal parameter estimation given sampled observations.

The implementation has the following features:

- Choice between Legendre, Chebyshev, central differences, trapezoidal or Hermite-Simpson discretisation;
- Large scale nonlinear programming using IPOPT and (optionally) SNOPT;
- Estimation of the discretisation error;
- Automatic mesh refinement;
- Automatic scaling;
- Automatic differentiation using the ADOL-C library;
- Numerical differentiation by using sparse finite differences;
- Automatic identification of the sparsity of the derivative matrices;
- DAE formulation, so that differential and algebraic constraints can be implemented in the same C++ function;
- Easy to use interface to GNUplot to produce graphical output, including 2D plots, 3D curves and surfaces, and polar plots;
- Automatic generation of LaTeX code to produce a table that summarizes the mesh refinement process.

Full details on PSOPT and its features can be found in its documentation[17].

1.8.2 GPOPS-II - MATLAB Optimal Control Software

GPOPS-II is a general purpose optimal control software available in the office site[18]. GPOPS-II is a new open-source MATLAB optimal control software that implements a brand new hp-adaptive Legendre-Gauss-Radau quadrature integral pseudospectral method for solving general nonlinear optimal control problems. Using GPOPS-II, the optimal control problem is transcribed to a nonlinear programming problem (NLP). The NLP is then solved using either the solver SNOPT or the solver IPOPT.

The following are some the key features of GPOPS-II:

- Allows for an extremely general formulation of the optimal control problem.
- Allows for inclusion of integral constraints and highly general boundary conditions.
- Complete first and second sparse finite-differencing of optimal control problem to compute all derivatives required by the NLP solver.

[17] `http://code.google.com/p/psopt/downloads/detail?name=PSOPT_Manual_R3.pdf&can=2&q=`
[18] `http://www.gpops.org`

- The latest advances in mesh refinement including hp–adaptive pseudospectral methods.
- Gaussian quadrature integration methods for rapid convergence.
- Highly accurate costate estimation.
- Inclusion of the NLP solver SNOPT (for Academic Users) and IPOPT (for Not-for-Profit and Commercial Users).
- No third-party products other than MATLAB are required.

More information about the methodology used in GPOPS-II can be found by reading the relevant articles in the open literature[19].

The fees for using GPOPS-II are as follows:

- For students and all others employed at academic institutions: NO CHARGE
- U.S. Federal, State, or Local Government: NO CHARGE
- Users at not-for-profit institutions (including non-U.S. Government agencies) or commercial institutions: LICENSING FEES APPLY

[19] `http://vdol.mae.ufl.edu/`

Chapter 2

Nonlinear Programming Solvers

2.1 Introduction

A solver is a software used to compute numerical solution of an optimal control problem. In general, an optimal control/nonlinear programming interface is needed to prepare all data as input to the solver (see Figure 2.1). Several solvers will be presented in this chapter, evolving open–source, freeware and commercial software, working under different operating systems. Since all solvers require configuration, lists of parameters will be presented for some of them.

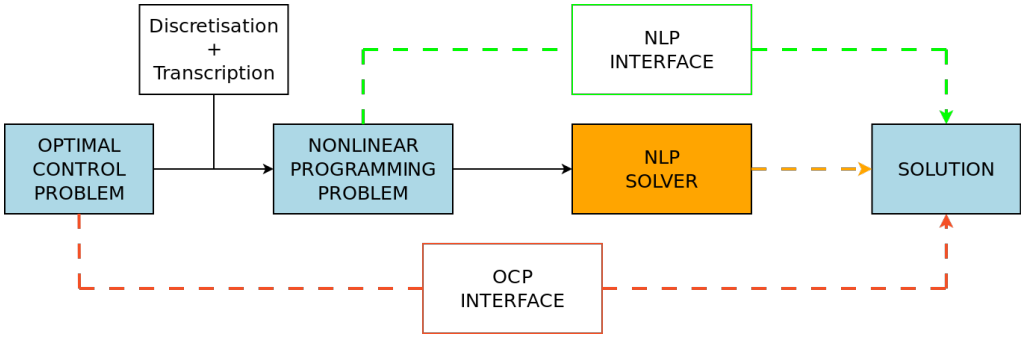


Figure 2.1

2.2 IPOPT – Interior Point OPTimizer

"IPOPT is a software package for large–scale nonlinear optimisation. It is designed to find (local) solutions of mathematical optimisation problems." The IPOPT library can be found in http://www.artelys.com/[1].

2.2.1 Basic information

Developed by Andreas Wächter and Carl Laird;

Operating system(s): IPOPT can be used on Linux, Unix, Mac OS X and Microsoft Windows;

Code language(s): IPOPT is written in C++ and is released as open source code under the Eclipse Public License (EPL).

[1] `http://www.artelys.com/index.php?page=knitro&hl=en_EN`

Modeling language interface(s): IPOPT can be used as a library that can be linked to C++, C or Fortran code, as well as a solver executable for the AMPL modelling environment. The package includes interfaces to CUTEr optimisation testing environment, as well as the MATLAB and R programming environments.

2.2.2 Installing IPOPT

How to install IPOPT on Linux

Getting the IPOPT code via subversion:

```
sudo svn co https://projects.coin-or.org/svn/Ipopt/stable/3.10 /opt/CoinIpopt
```

This will create a directory named `CoinIpopt` in your `/opt/` folder.

BLAS routines that provide standard building blocks for performing basic vector and matrix operations;

LAPACK routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems;

HSL state-of-the-art packages for large-scale scientific computation;

ASL AMPL Solver Library allows to read the `.nl` files and provides the automatic differentiation functionality.

MUMPS MUltifrontal Massively Parallel sparse direct Solver;

METIS programs for partitioning graphs, partitioning finite element meshes, and producing fill reducing orderings for sparse matrices.

To install IPOPT just run:

```
cd /opt/CoinIpopt/ThirdParty/Blas
./get.Blas
cd ../Lapack
./get.Lapack
cd ../ASL
./get.ASL
cd ../Mumps
./get.Mumps
cd ../Metis
./get.Metis
```

IPOPT requires a sparse symmetric linear solver. There are different possibilities. The user's guide of IPOPT suggests that you use HSL subroutines:

1. Go to HSL website[2] and find the package list;
2. Click on MA27, read the license and submit the registration form;
3. Go back to the package list;
4. Click on MC19, read the license and submit the registration form.

[2] `http://hsl.rl.ac.uk/archive/hslarchive.html`

You will receive in your email the links to obtain the HSL packages. Download them to your $HOME directory and run:

```
gunzip ma27-1.0.0.tar.gz
tar xvf ma27-1.0.0.tar
gunzip mc19-1.0.0.tar.gz
tar xvf mc19-1.0.0.tar
cd /opt/CoinIpopt/ThirdParty/HSL/
sudo mv -r $HOME/mc19-1.0.0 .
sudo mv -r $HOME/ma27-1.0.0 .
cd mc19-1.0.0
sudo ./configure
sudo make install
sudo make check
cd ../ma27-1.0.0
sudo ./configure
sudo make install
sudo make check
```

The `make check` command will check if the installation of the HSL packages was successful.
Return to the `/opt/CoinIpopt` directory and continue the IPOPT installation:

```
cd ../..
sudo mkdir build
cd build
sudo ../configure
sudo make test
sudo make install
```

If you get the error message "`error:'NULL' was not declared in this scope`" when running `sudo make`, then you have to edit the following files:
`/opt/CoinIpopt/Ipopt/src/Common/IpSmartPtr.hpp`
`/opt/CoinIpopt/Ipopt/src/Algorithm/LinearSolvers/IpTripletToCSRConverter.cpp`
by adding `#include <cstddef>`.

How to install IPOPT on Mac OS X

In a Terminal window, install `homebrew`:
`/usr/bin/ruby -e "$(/usr/bin/curl -fsSL https://raw.github.com/mxcl/homebrew/master/Library/Contributions/install_homebrew.rb)"`

With this software you can easily install dependency packages like `wget` and `gfortran`:

```
brew install wget
brew install gfortran
sudo wget http://www.coin-or.org/download/source/Ipopt/Ipopt-3.10.2.zip
sudo unzip Ipopt-3.10.2.zip
cd Ipopt-3.10.2
```

Download Netlib Blas/Lapack:

```
cd /opt/CoinIpopt/ThirdParty/Blas
sudo ./get.Blas
```

```
cd ../Lapack
sudo ./get.Lapack
cd ../ASL
sudo ./get.ASL
cd ../Mumps
sudo ./get.Mumps
cd ../Metis
sudo ./get.Metis
```

Make a build for a 64bit target:

```
sudo mkdir build64
cd build64
sudo ../configure --disable-shared --with-blas=BUILD --with-lapack=BUILD F77=gfortran
FFLAGS="-fexceptions -m64 -fbackslash" CFLAGS="-fno-common -no-cpp-precomp -fexceptions
-arch x86_64 -m64" CXXFLAGS="-fno-common -no-cpp-precomp -fexceptions -arch x86_64
-m64"
```

Compile, test and install:

```
make
make test
make install
```

Add the following to your `/.bashrc` or `/.bash_profile`:

```
IPOPTDIR=/opt/CoinIpopt/build
export PATH=$PATH:$IPOPTDIR:$IPOPTDIR/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$IPOPTDIR/ThirdParty/ASL
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$IPOPTDIR/ThirdParty/Blas/.libs
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$IPOPTDIR/ThirdParty/HSL/.libs
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$IPOPTDIR/ThirdParty/Lapack/.libs
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$IPOPTDIR/ThirdParty/Metis/.libs
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$IPOPTDIR/ThirdParty/Mumps/.libs
```

How to install the Matlab interface on Linux

First of all, be sure

```
cd /opt/matlabR2008a/sys/os/glnx86
mv libgcc_s.so.1 libgcc_s.so.1.bck
ln -s /lib/i386-linux-gnu/libgcc_s.so.1 libgcc_s.so.1
mv libstdc++.so.6.0.8 libstdc++.so.6.0.8.bck
ln -s /usr/lib/i386-linux-gnu/libstdc++.so.6 libstdc++.so.6.0.8
```

Go to the subdirectory `/opt/CoinIpopt/build/Ipopt/contrib/MatlabInterface/src` and open the Makefile file.

```
sudo apt-get install hardening-wrapper
```

You need to edit this file to suit the installation for your system setup. You may need to modify `MATLAB_HOME`, `MEXSUFFIX` and `MEX` as explained in the comments of the Makefile. I set these variables as follows:

```
MATLAB_HOME = /opt/matlabR2008a
MEXSUFFIX = mexglx
MEX = /opt/matlabR2008a/bin/mex
```

In my case, I removed some of the compiler options as it follows:

```
prefix = /opt/CoinIpopt
CXX = /opt/intel/composer_xe_2011_sp1.9.293/bin/ia32/icpc
CCXXFLAGS = -O3 -pipe -DNDEBUG -Wparentheses -Wreturn-type -Wcast-qual -Wall
-Wpointer-arith -Wwrite-strings -Wconversion -Wno-unknown-pragmas -DMATLAB_MEXFILE
LDFLAGS = $(CXXFLAGS) -Wl, --rpath -Wl, ${prefix}/lib/coin -Wl, --rpath -Wl,
--rpath -Wl,${prefix}/lib/coin/ThirdParty
```

In my case, I had to set explicitly the include Matlab folder:

```
%.o:  %.cpp
$(CXX) $(CXXFLAGS) $(INCL) -I/opt/matlabR2008a/extern/include -o $ -c $^
```

Once you set up the Makefile properly, type `sudo make install` in the same directory as the Makefile. If the installation procedure was successful, you will end up with a MEX file called `ipopt.mexglx`. In order to use it in MATLAB, you need to tell MATLAB where to find it. To do this just type `addpath /opt/CoinIpopt/lib` in the MATLAB command window after you check if the `ipopt.mexglx` file is in this folder.

Documentation

More informations can be found in the following documentation:

Online documentation: http://www.coin-or.org/Ipopt/documentation/[3]
A PDF version of this documentation can be downloaded here[4].

A short IPOPT tutorial: can be downloaded here[5].

MATLAB interface: Instructions specific to the MATLAB interface can be found at the Matlab interface page[6].

2.2.3 Getting started with IPOPT and AMPL

```
option solver ipopt ;
option ipopt_options 'option1=value1 option2=value2' ;
solve ;
```

[3] http://www.coin-or.org/Ipopt/documentation/
[4] https://projects.coin-or.org/Ipopt/browser/stable/3.9/Ipopt/doc/documentation.pdf?format=raw
[5] http://drops.dagstuhl.de/volltexte/2009/2089/pdf/09061.WaechterAndreas.Paper.2089.pdf
[6] https://projects.coin-or.org/Ipopt/wiki/MatlabInterface

Table 2.1: IPOPT directives for AMPL

Option	Description
acceptable_compl_inf_tol	Acceptance threshold for the complementarity conditions
acceptable_constr_viol_tol	Acceptance threshold for the constraint violation
acceptable_dual_inf_tol	Acceptance threshold for the dual infeasibility
acceptable_tol	Acceptable convergence tolerance (relative)
alpha_for_y	Step size for constraint multipliers
bound_frac	Desired minimal relative distance of initial point to bound
bound_mult_init_val	Initial value for the bound multipliers
bound_push	Desired minimal absolute distance of initial point to bound
bound_relax_factor	Factor for initial relaxation of the bounds
compl_inf_tol	Acceptance threshold for the complementarity conditions
constr_mult_init_max	Maximal allowed least-square guess of constraint multipliers
constr_viol_tol	Desired threshold for the constraint violation
diverging_iterates_tol	Threshold for maximal value of primal iterates
dual_inf_tol	Desired threshold for the dual infeasibility
expect_infeasible_problem	Enable heuristics to quickly detect an infeasible problem
file_print_level	Verbosity level for output file
halt_on_ampl_error	Exit with message on evaluation error
hessian_approximation	Can enable Quasi-Newton approximation of hessian
honor_original_bounds	If no, solution might slightly violate bounds
linear_scaling_on_demand	Enables heuristic for scaling only when seems required
linear_solver	Linear solver to be used for step calculation
linear_system_scaling	Method for scaling the linear systems
ma27_pivtol	Pivot tolerance for the linear solver MA27
ma27_pivtolmax	Maximal pivot tolerance for the linear solver MA27
ma57_pivtol	Pivot tolerance for the linear solver MA57
ma57_pivtolmax	Maximal pivot tolerance for the linear solver MA57

Table 2.2: IPOPT directives for AMPL

Option	Description
max_cpu_time	CPU time limit
max_iter	Maximum number of iterations
max_refinement_steps	Maximal number of iterative refinement steps per linear system solve
max_soc	Maximal number of second order correction trial steps
maxit	Maximum number of iterations (same as max_iter)
min_refinement_steps	Minimum number of iterative refinement steps per linear system solve
mu_init	Initial value for the barrier parameter
mu_max	Maximal value for barrier parameter for adaptive strategy
mu_oracle	Oracle for a new barrier parameter in the adaptive strategy
mu_strategy	Update strategy for barrier parameter
nlp_scaling_max_gradient	Maximum gradient after scaling
nlp_scaling_method	Select the technique used for scaling the NLP
obj_scaling_factor	Scaling factor for the objective function
option_file_name	File name of options file (default: `ipopt.opt`)
outlev	Verbosity level (same as print_level)
output_file	File name of an output file (leave unset for no file output)
pardiso_matching_strategy	Matching strategy for linear solver Pardiso
pardiso_out_of_core_power	Enables out-of-core version of linear solver Pardiso
print_level	Verbosity level
print_options_documentation	Print all available options (for ipopt.opt)
print_user_options	Toggle printing of user options
required_infeasibility_reduction	Required infeasibility reduction in restoration phase
slack_bound_frac	Desired minimal relative distance of initial slack to bound
slack_bound_push	Desired minimal absolute distance of initial slack to bound
tol	Desired convergence tolerance (relative)
wantsol	solution report without -AMPL: sum of 1 : write `.sol` file 2 : print primal variable values 4 : print dual variable values 8 : do not print solution message
warm_start_bound_push	Enables to specify how much should variables should be pushed inside the feasible region
warm_start_init_point	Enables to specify bound multiplier values
warm_start_mult_bound_push	Enables to specify how much should bound multipliers should be pushed inside the feasible region
watchdog_shortened_iter_trigger	Trigger counter for watchdog procedure
wsmp_num_threads	Number of threads to be used in WSMP
wsmp_pivtol	Pivot tolerance for the linear solver WSMP
wsmp_pivtolmax	Maximum pivot tolerance for the linear solver WSMP
wsmp_scaling	Determines how the matrix is scaled by WSMP

2.3 KNITRO

"KNITRO is an optimization software library for finding solutions of both continuous (smooth) optimization models (with or without constraints), as well as discrete optimization models with integer or binary variables (i.e. mixed integer programs). KNITRO is primarily designed for finding local optimal solutions of large-scale, continuous nonlinear problems." The KNITRO library can be found in http://www.artelys.com/[7].

2.3.1 Basic information

KNITRO is a software package for solving smooth optimization problems, with or without constraints.

Developed at Ziena Optimization;

Operating system(s): KNITRO can be used on Linux, Unix, Mac OS X and Microsoft Windows;

Code language(s): KNITRO is written C, C++, Fortran and Java;

Programming interfaces(s): Fortan, C/C++, Java and Microsoft Excel;

Modeling language interface(s): KNITRO offers a total of five interfaces: Matlab, AMPL, Mathematica, AIMMS, GAMS and MPL.

2.3.2 Installing KNITRO on Linux

How to get a license

KNITRO is developed by Ziena Optimization LLC and marketed and supported by Artelys. The first step is to request a license by selecting the download tab[8]. Along with the academic and commercial trial versions, there is a student version limited in problem size (300 variables and 300 constraints) available for 6 months. Each license is personalised, hence, among other personal information, you should inform about

- the operating system (OS);
- the OS bit architecture;
- the Machine ID of the computer on which KNITRO will run.
  Download, extract an script available bellow the Machine ID field.

After reading and accepting the End User License Agreement, the download of the KNITRO will be available and you will get a license key on your email. Then, you should put the license file, whose name begins with `ziena_lic`, in your `$HOME` directory or in the `$HOME/.ziena/` directory.

How to install

The KNITRO software package for Unix is delivered as a gzipped tar file. After downloading KNITRO, you need to unpack it. Type in a Terminal window:

```
gunzip knitro-8.x-platformname.tar.gz
tar -xvf knitro-8.x-platformname.tar
sudo mv knitro-8.x-z /opt/
```

[7] http://www.artelys.com/index.php?page=knitro&hl=en_EN
[8] http://www.artelys.com/index.php?page=knitro&hl=en_EN#downloads-tab

This will create a directory named `knitro-8.x-z` in your `/opt/` folder.

In order to run KNITRO binary or executable files from anywhere on your Linux computer, it is necessary to set several environment variables. In particular, you must update the `PATH` environment variable so that it indicates the location of the `knitroampl` directory. You must also update the `LD_LIBRARY_PATH` environment variable so that it indicates the location of the KNITRO lib directory. Setting the `PATH` and `LD_LIBRARY_PATH` environment variables on a Linux system can be done as follows:

```
echo 'export PATH=$PATH:/opt/knitro-8.x-z' >> ${HOME}/.bashrc
echo 'export PATH=$PATH:/opt/knitro-8.x-z/knitroampl' >> ${HOME}/.bashrc
echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/knitro-8.x-z/lib' >> ${HOME}/.bashrc
```

Each Matlab release expects the KNITRO library file on Linux to have a specific name. Therefore, to use KNITRO 8.x or later with Matlab release R2008a through 2011b, you must create a symbolic link to the expected name. Open a Terminal window and, in the KNITRO library directory, issue the command:

- for Matlab releases R2008a, R2008b or R2009a
  `ln -s libknitro.so.8.0.0 libknitro.so.5`
- for Matlab releases R2009b, R2010a or R2010b
  `ln -s libknitro.so.8.0.0 libknitro.so.6`
- for Matlab releases R2011a, R2011b or R2012a
  `ln -s libknitro.so.8.0.0 libknitro.so.7`

Documentation

More informations can be found in the following documentation[9]:

KNITRO 8.0 User Manual: Knitro80_UserManual.pdf
Version: December 2011 (816 KB)

Specific KNITRO/Matlab interface documentation can be found on the Matlab Optimization Toolbox webpages[10].

2.3.3 Getting started with KNITRO and AMPL

```
option solver knitroampl ;
option knitro_options 'option1=value1 option2=value2' ;
solve ;
```

[9] `http://www.artelys.com/uploads/pdfs/Knitro80_UserManual.pdf`
[10] `http://www.mathworks.com/access/helpdesk/help/toolbox/optim/`

Table 2.3: KNITRO directives for AMPL (1)

Option	Description
alg	Algorithm (0=auto, 1=direct, 2=cg, 3=active, 5=multi)
algorithm	Algorithm (0=auto, 1=direct, 2=cg, 3=active, 5=multi)
bar_directinterval	Frequency for trying to force direct steps
bar_feasible	Emphasize feasibility
bar_feasmodetol	Tolerance for entering stay feasible mode
bar_initmu	Initial value for barrier parameter
bar_initpt	Barrier initial point strategy
bar_maxbacktrack	Maximum number of linesearch backtracks
bar_maxcrossit	Maximum number of crossover iterations
bar_maxrefactor	Maximum number of KKT refactorizations allowed
bar_murule	Rule for updating the barrier parameter
bar_penaltycons	Apply penalty method to constraints
bar_penaltyrule	Rule for updating the penalty parameter
bar_switchrule	Rule for barrier switching alg
blasoption	Which BLAS/LAPACK library to use
blasoptionlib	Name of dynamic BLAS/LAPACK library
cplexlibname	Name of dynamic CPLEX library
debug	Debugging level (0=none, 1=problem, 2=execution)
delta	Initial trust region radius
feastol	Feasibility stopping tolerance
feastol_abs	Absolute feasibility tolerance
feastolabs	Absolute feasibility tolerance
gradopt	Gradient computation method
hessopt	Hessian computation method
honorbnds	Enforce satisfaction of the bounds
infeastol	Infeasibility stopping tolerance
linsolver	Which linear solver to use
lmsize	Number of limited-memory pairs stored for LBFGS
lpsolver	LP solver used by Active Set algorithm
ma_maxtime_cpu	Maximum CPU time when 'alg=multi', in seconds
ma_maxtime_real	Maximum real time when 'alg=multi', in seconds
ma_outsub	Enable subproblem output when 'alg=multi'
ma_terminate	Termination condition when option 'alg=multi'
maxcgit	Maximum number of conjugate gradient iterations
maxit	Maximum number of iterations
maxtime_cpu	Maximum CPU time in seconds, per start point
maxtime_real	Maximum real time in seconds, per start point
mip_branchrule	MIP branching rule
mip_debug	MIP debugging level (0=none, 1=all)
mip_gub_branch	Branch on GUBs (0=no, 1=yes)
mip_heuristic	MIP heuristic search
mip_heuristic_maxit	MIP heuristic iteration limit
mip_implications	Add logical implications (0=no, 1=yes)
mip_integer_tol	Threshold for deciding integrality
mip_integral_gap_abs	Absolute integrality gap stop tolerance
mip_integral_gap_rel	Relative integrality gap stop tolerance
mip_knapsack	Add knapsack cuts (0=no, 1=ineqs, 2=ineqs+eqs)
mip_lpalg	LP subproblem algorithm

Table 2.4: KNITRO directives for AMPL (2)

Option	Description
mip_maxnodes	Maximum nodes explored
mip_maxsolves	Maximum subproblem solves
mip_maxtime_cpu	Maximum CPU time in seconds for MIP
mip_maxtime_real	Maximum real in seconds time for MIP
mip_method	MIP method (0=auto, 1=BB, 2=HQG)
mip_outinterval	MIP output interval
mip_outlevel	MIP output level
mip_outsub	Enable MIP subproblem output
mip_pseudoinit	Pseudo-cost initialization
mip_rootalg	Root node relaxation algorithm
mip_rounding	MIP rounding rule
mip_selectrule	MIP node selection rule
mip_strong_candlim	Strong branching candidate limit
mip_strong_level	Strong branching tree level limit
mip_strong_maxit	Strong branching iteration limit
mip_terminate	Termination condition for MIP
ms_enable	Enable multistart
ms_maxbndrange	Maximum unbounded variable range for multistart
ms_maxsolves	Maximum KNITRO solves for multistart
ms_maxtime_cpu	Maximum CPU time for multistart, in seconds
ms_maxtime_real	Maximum real time for multistart, in seconds
ms_num_to_save	Feasible points to save from multistart
ms_outsub	Enable subproblem output for parallel multistart
ms_savetol	Tol for feasible points being equal
ms_seed	Seed for multistart random generator
ms_startptrange	Maximum variable range for multistart
ms_terminate	Termination condition for multistart
newpoint	Use newpoint feature
objno	objective number: 0 = none, 1 = first (default), 2 = second (if _nobjs $>$ 1), etc.
objrange	Objective range
objrep	Whether to replace minimize obj: v; with minimize obj: f(x) when variable v appears linearly in exactly one constraint of the form s.t. c: $v \geq f(x)$; or s.t. c: $v = f(x)$; Possible objrep values: 0 = no 1 = yes for $v \geq f(x)$ (default) 2 = yes for $v = f(x)$ 3 = yes in both cases

Table 2.5: KNITRO directives for AMPL (3)

Option	Description
opttol	Optimality stopping tolerance
opttol_abs	Absolute optimality tolerance
opttolabs	Absolute optimality tolerance
outappend	Append to output files (0=no, 1=yes)
outdir	Directory for output files
outlev	Control printing level
outmode	Where to direct output (0=screen, 1=file, 2=both)
par_blasnumthreads	Number of parallel threads for BLAS
par_numthreads	Number of parallel threads
pivot	Initial pivot tolerance
presolve	KNITRO presolver level
presolve_dbg	KNITRO presolver debugging level
presolve_tol	KNITRO presolver tolerance
relax	whether to ignore integrality: 0 (default) = no, 1 = yes
scale	Automatic scaling option
soc	Second order correction options
timing	Whether to report problem I/O and solve times: 0 (default) = no 1 = yes, on stdout
version	Report software version
wantsol	solution report without -AMPL: sum of 1 : write .sol file 2 : print primal variable values 4 : print dual variable values 8 : do not print solution message
xpresslibname	Name of dynamic Xpress library
xtol	Stepsize stopping tolerance

2.4 WORHP – WORHP Optimises Really Huge Problems

"WORHP is a software library for mathematical nonlinear optimization, suitable for solving problems with thousands or even millions of variables and constraints." The WORHP library can be found in http://www.worhp.de/[11].

2.4.1 Basic information

Developed under the direction of Christof Büskens with Matthias Gerdts;

Operating system(s): WORHP runs on Linux, Unix, Mac OS X and Microsoft Windows;

Code language(s): WORHP is written Fortan and C;

Modeling language interface(s): WORHP offers a total of nine interfaces: 3 for Fortran, 3 for C/C++, Matlab, ASTOS and AMPL, for different programming languages and communication paradigms.

2.4.2 Installing WORHP on Linux

How to get a license

The first step is to check the availability of WORHP for your favourite platform. WORHP is available for several platforms, although some restrictions apply:

- Minimum supported `gcc` version is 4.3.
- Windows Visual Studio builds are only available for 32-bit systems.
  (64-bit builds should be available by mid-2012)
- The MATLAB interface for Windows is available for 32-bit systems only.
- Builds of the MATLAB interface may have compatibility issues between different MATLAB versions. The developers can generally supply builds for a number of different MATLAB versions upon request.
- Builds for Apple Macs are subject to platform availability.

Then, you should request a license by logging in the WORHP website[12]. Licenses are available for academic and commercial users. You should fill–up the form available in Get WORHP > For Academic > Request a license.

Each license is personalised, hence, among other personal information, you should inform about

- the operating system (OS);
- the OS version;
  (in a Terminal window run: `lsb_release -cs`)
- the OS bit architecture;
- the compiler family (C or Fortran) to be used;
  (in a Terminal window run: `gfortran --version` or `gcc --version`)
- the MAC address of the computer on which WORHP will run.

[11] `http://www.worhp.de/`
[12] `http://www.worhp.de/`

Documentation

The following documentation[13] is available:

WORHP User Manual: Worhp_User_Manual.pdf
Documentation of WORHP from first steps to usage of different interfaces.
Version: 2012-03-02 (749.98 KB)

WORHP Tutorial: Worhp_Tutorial.pdf
Introduction to using WORHP, with detailled installation instructions.
Version: 2012-03-08 (423.6 KB)

WORHP Technical Datasheet: Worhp_Datasheet.pdf
7-page datasheet about the key technical properties of WORHP.
Version: 2011-08-18 (788.65 KB)

WORHP Conference Publication: Worhp_Conference_Paper.pdf
Prepared for 62nd International Astronautical Congress in Cape Town, South Africa (reformatted with generic layout).
Version: 2011-11-01 (959.46 KB)

[13] http://www.worhp.de/content/publications

2.5 Other Commercial Packages

2.5.1 SOCS – Sparse Optimal Control Software

"The Sparse Optimal Control Family, developed by The Boeing Company, contains two advanced software packages, available separately or together."
Sparse Optimal Control Software (SOCS) is general-purpose software for solving optimal control problems. Applications include trajectory optimization, chemical process control and machine tool path definition. The SNOPT library can be found in http://www.boeing.com/[14].

Developed by The Boeing Company;

Operating system(s): SOCS is supported on most UNIX and Windows systems. The software is supported on most major platforms with at least 14 decimal digits of precision, which on most systems means double precision.

Code language(s): This software and all lower-level support routines are written in ANSI-Standard FORTRAN 77;

Interface(s): FORTRAN 77.

2.5.2 SNOPT – Sparse Nonlinear OPTimizer

SNOPT is a software package for solving large-scale optimization problems (linear and non–linear programs). It is especially effective for non–linear problems whose functions and gradients are expensive to evaluate. The functions should be smooth but need not be convex. The SNOPT library can be found in http://www.sbsi-sol-optimize.com/[15].

Developed by Philip Gill, Walter Murray and Michael Saunders;

Operating system(s): It is intended for any machine with a reasonable amount of memory and a FORTRAN compiler;

Code language(s): SNOPT is implemented in FORTRAN 77 and distributed as source code;

Interface(s): It may be called from a driver program, typically in Fortran, C or MATLAB.

[14] http://www.boeing.com/phantom/socs/
[15] http://www.sbsi-sol-optimize.com/asp/sol_product_snopt.htm

Chapter 3

Project webpage

One of main tasks of this project was to develop and to maintain the project web–site, making documentation accessible to the project team. All documentation that is supporting this project is available in the project web–site[1]. This web site was written in HTML language and it can be easily updated.

Figure 3.1: Screenshot of the project web–site

[1] http://paginas.fe.up.pt/~faf/ProjectFCT2009/

Chapter 4

Optimal Control Toolbox

One of tasks to be carried out was to construct a platform to easily solve sequences of optimal control problems in a receding horizon fashion in order to implement model predictive controllers. We took this goal a little further and we thought to develop a Matlab platform that would be able to solve optimal control problems using different solvers and evolving several ODE methods as in Figure 4.1. This would require a huge time investment and we knew from the beginning it would be a hard task to finish during this project. Still, we archived a good work that could be the starting point for a new project (See appendix A.1).

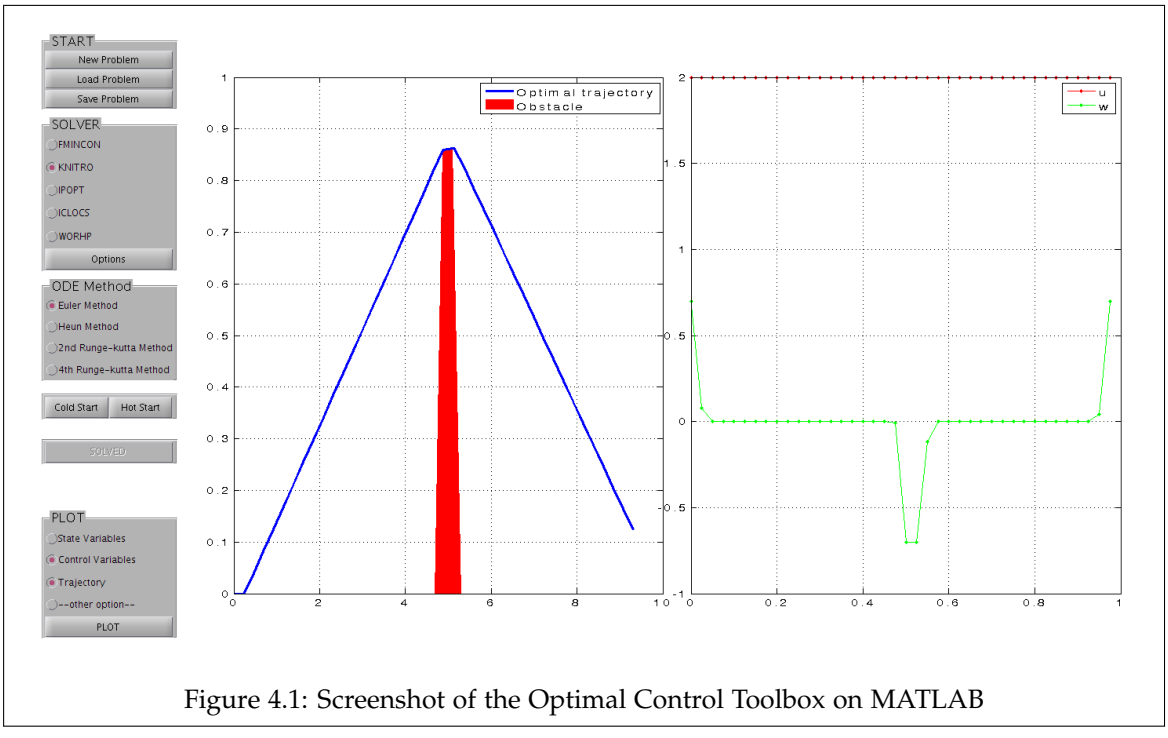


Figure 4.1: Screenshot of the Optimal Control Toolbox on MATLAB

Chapter 5

Applications

Another task of this project was too help to implement and to solve case studies and tests. In this chapter, several problems are presented using some of the interfaces and solvers previously discussed.

5.1 Car–Like

The Car–like problem can be started as:

$$
\begin{aligned}
\text{Minimize} \quad & T \\
\text{subject to} \quad & \dot{x} = u\cos(\theta) \\
& \dot{y} = u\,\text{sen}(\theta) \\
& \dot{\theta} = u\,w \\
\\
& u_{\min} \leq u \leq u_{\max} \\
& w_{\min} \leq w \leq w_{\max} \\
& y \geq \bar{y} - k(x - \bar{x}) \\
& x(0) = x_0\,, \quad x(T) = x_f \\
\\
& y(0) = x_0\,, \quad y(T) = y_f \\
& \theta(0) = \theta_0\,, \quad \theta(T) = \theta_f
\end{aligned}
$$

Change of variables:

$$
\begin{aligned}
&\tau \in [0,1] \\
&t(\tau) : [0,1] \to [0,T] \\
&t(0) = 0\,, \quad t(1) = T
\end{aligned}
$$

$$
\begin{aligned}
&\frac{\mathrm{d}t}{\mathrm{d}\tau} = v\,, \quad v > 0 \\
&\frac{\mathrm{d}x}{\mathrm{d}\tau} = \frac{\mathrm{d}x}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\tau} = v\frac{\mathrm{d}x}{\mathrm{d}t} \\
&\frac{\mathrm{d}y}{\mathrm{d}\tau} = \frac{\mathrm{d}y}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\tau} = v\frac{\mathrm{d}y}{\mathrm{d}t} \\
&\frac{\mathrm{d}\theta}{\mathrm{d}\tau} = \frac{\mathrm{d}\theta}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\tau} = v\frac{\mathrm{d}\theta}{\mathrm{d}t}
\end{aligned}
$$

$$
\begin{aligned}
\underset{v\in\mathbb{R},\, u_1,\, u_2\in\mathbb{U}}{\text{Minimize}} \quad & t(1) \\
\text{subject to} \quad & \frac{\mathrm{d}x}{\mathrm{d}\tau} = v\,u\cos\left(\theta(\tau)\right) && \forall\tau \in [0,1] \\
& \frac{\mathrm{d}y}{\mathrm{d}\tau} = v\,u\,\mathrm{sen}\left(\theta(\tau)\right) && \forall\tau \in [0,1] \\
& \frac{\mathrm{d}\theta}{\mathrm{d}\tau} = v\,u\,w && \forall\tau \in [0,1] \\
& \frac{\mathrm{d}t}{\mathrm{d}\tau} = v && \forall\tau \in [0,1] \\
& 0 \le u \le 2 && \forall\tau \in [0,1] \\
& -0.7 \le w \le 0.7 && \forall\tau \in [0,1] \\
& 1 \le v \le 2\sqrt{(x_n - x_0)^2 + (y_n - y_0)^2} && \forall\tau \in [0,1] \\
& y \ge \bar{y} - k(x - \bar{x})^2 && \forall\tau \in [0,1]
\end{aligned}
$$

Initial conditions:

$$
\begin{aligned}
&x_0 = 0\,, \quad x_f = 10 \\
&y_0 = 0\,, \quad y_f = 0 \\
&\theta_0 = 0\,, \quad \theta_f = 0 \\
&\left(x_n - x_f\right)^2 + \left(y_n - y_f\right)^2 + \left(\theta_n - \theta_f\right)^2 \le 0.5
\end{aligned}
$$

$$
n = 40\,, \quad (\bar{x},\, \bar{y}) = (5,\, 1)\,, \quad k = 10
$$

Interface: AMPL (see appendix A.2)

Solver: IPOPT

Solver settings:

```
option solver ipopt ;
```

```
ipopt_options "max_iter=999 acceptable_tol=1e 8" ;
```

Solution: $v = 4.7236725474$ The problem solved in: 324 iterations
Time taken: 1.23 s

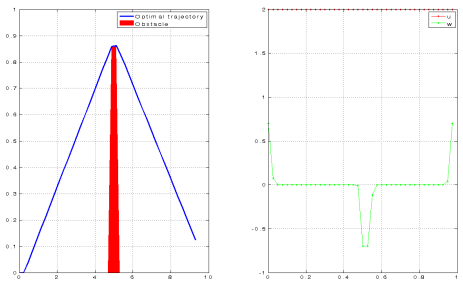


Figure 5.1: Optimal trajectory and controls

5.2 Goddard Problem

The Goddad problem can be started as:

$$
\begin{aligned}
\text{Maximize} \quad & m(T) \\
\text{subject to} \quad & \dot{r} = v && \forall t \in [0, T] \\
& \dot{v} = -\frac{1}{r^2} + \frac{1}{m}\left(T_{max} u - D(r, v)\right) && \forall t \in [0, T] \\
& \dot{m} = -b T_{max} u && \forall t \in [0, T] \\
& 0 \leq u \leq 1 && \forall t \in [0, T] \\
& D(r(\cdot), v(\cdot)) \geq C && \forall t \in [0, T]
\end{aligned}
$$

Initial conditions:

$$
\begin{aligned}
& r(0) = 1\,, \quad r(T) = 1.01 \\
& v(0) = 0 \\
& m(0) = 1 \\
& \text{where} \quad D(r, v) = A v^2 \rho(r), \quad \rho(r) = e^{-k(r - r_0)} \\
& \text{and} \quad b = 7,\ T_{max} = 3.5 \\
& \qquad A = 310,\ k = 500,\ r_0 = 1
\end{aligned}
$$

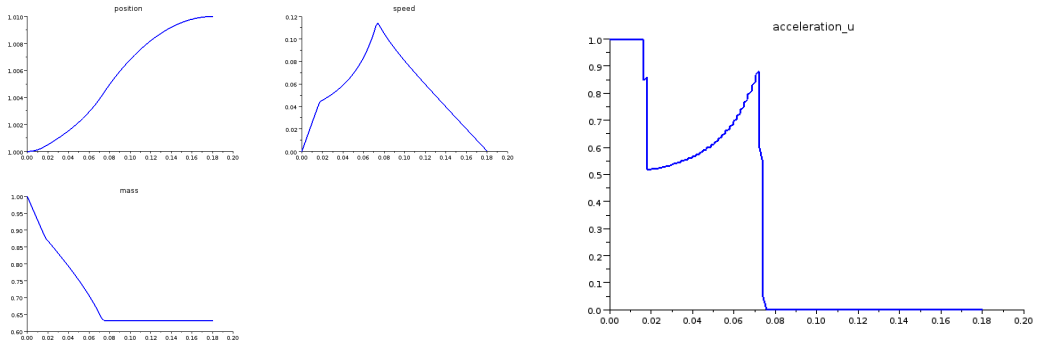
Interface: BOCOP

Solver: IPOPT

Solver settings:

```
max_iter=1000
tol=1.0000000000e-6
mu_strategy=adaptive
```

Solution: $m = -6.327366 \times 10^{-1}$
The problem solved in: 20 iterations
Time taken: 2.24 s



(a) State variables (b) Control Variables

Figure 5.2: Optimal trajectories and control

5.3 HIV

For most of the present HIV chemotherapy drugs, the state system would be:

$$\begin{aligned}
\frac{\mathrm{d}\,T}{\mathrm{d}\,t} &= \frac{s}{1+V} - \mu_T T + rT\left(1 - \frac{T+T^*+T^{**}}{T_{\max}}\right) - k_1 VT \\
\frac{\mathrm{d}\,T^*}{\mathrm{d}\,t} &= k_1 VT - \mu_{T^*} T^* - k_2 T^* \\
\frac{\mathrm{d}\,T^{**}}{\mathrm{d}\,t} &= k_2 T^* - \mu_b T^{**} \\
\frac{\mathrm{d}\,V}{\mathrm{d}\,t} &= (1-\mathbf{u}(t))\, N\mu_b T^{**} - k_1 VT - \mu_V V
\end{aligned}$$

where

$T(t)$	concentration of uninfected CD4$^+$ T cells
$T^*(t)$	concentration of latently infected CD4$^+$ T cells
$T^{**}(t)$	concentration of actively infected CD4$^+$ T cells
$V(t)$	concentration of free infectious virus particles
$u(t)$	control, rate of chemotherapy, $0 \leq u(t) \leq 1$
	$u(t) = 1$: maximal chemotherapy
	$u(t) = 0$: no chemotherapy

The objective function:

$$\max \int_0^{T_f} \left(T(t) - \frac{1}{2} B\mathbf{u}(t)^2\right) \mathrm{d}\,t$$

Initial conditions, parameters and constants:

$$\begin{aligned}
T(0) &= 982.8 \\
T^*(0) &= 0.05155 \\
T^{**}(0) &= 6.175 \times 10^{-4} \\
V(0) &= 0.07306
\end{aligned}$$

	Parameters and constants	Values
B	parameter	100
μ_T	death rate of ininfected CD4$^+$ T cell population	$0.02\,\mathrm{d}^{-1}$
μ_{T^*}	death rate of latently infected CD4$^+$ T cell population	$0.2\,\mathrm{d}^{-1}$
$\mu_{T^{**}}$	death rate of actively infected CD4$^+$ T cell population	$0.24\,\mathrm{d}^{-1}$
μ_V	death rate of free virus	$2.4\,\mathrm{d}^{-1}$
k_1	rate CD4$^+$ T cells becomes infected by free virus	$2.4 \times 10^{-5}\,\mathrm{mm}^3\mathrm{d}^{-1}$
k_2	rate T^* cells convert to actively infected	$3 \times 10^{-3}\,\mathrm{mm}^3\mathrm{d}^{-1}$
r	rate of growth for the CD4$^+$ T cell population	$0.03\,\mathrm{d}^{-1}$
N	number of free virus produced by T^{**} cells	1200
$T_{\max}$	maximum CD4$^+$ T cell population level	$1.5 \times 10^3\,\mathrm{mm}^{-3}$
s	source term for uninfected CD4$^+$ T cells	$10\,\mathrm{d}^{-1}\mathrm{mm}^{-3}$

Table 5.1: Parameters and constants

Interface: ICLOCS (see appendix A.3)

Solver: IPOPT

Solver settings:

```
options.transcription='hermite';
options.derivatives='numeric';
options.hessianFD='central';
options.NLPsolver='ipopt';
options.ipopt.tol=1e-9;
options.ipopt.print_level=5;
options.ipopt.max_iter=200;
options.ipopt.mu_strategy ='adaptive';
options.ipopt.hessian_approximation='exact';
options.fmincon=optimset;
options.scaling=1;
options.print.time=1;
options.print.relative_local_error=1;
options.print.cost=1;
options.plot.states=1;
options.plot.inputs=1;
options.plot.multipliers=1;
options.nodes=1000;
options.tau=0;
options.ODEsolver='cvodes';
Method='Adams';
Solver='Newton';
```

Solution: -4.9204×10^5
The problem solved in: 17 iterations
Time taken: 11.26 s

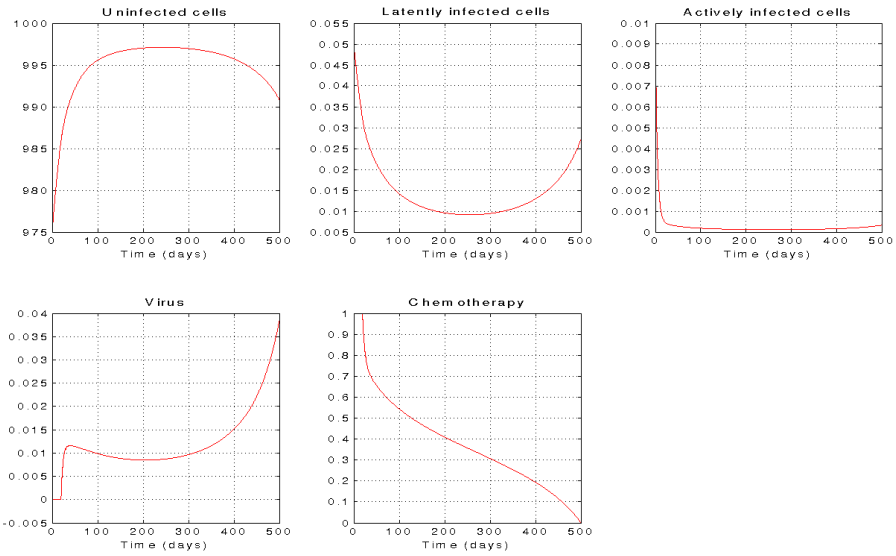


Figure 5.3: Treatment period of 500 days starting after 800 days of infection

Appendix A

Programming code

A.1 Optimal Control Toolbox: MATLAB Code

```
function OptimalControlToolbox

global start solver ODE solve plots initsol;

% Auxiliary vars
ss = get(0,'ScreenSize');
FontSize.panel.title = 14;
FontSize.axes.title = 16;
Length.panel = 0.12;
Length.axes = Length.panel + 0.07;
Button.height = 0.04;


%%% FIGURE %%%
hfig = figure('Position',ss);
set(hfig,'Visible','off');
set(hfig,'Name','Optimal Control Toolbox','NumberTitle','off');
%set(hfig,'MenuBar','none');


%%% FIGURE MENU %%%
figmenu = uimenu('Label','Workspace');
uimenu(figmenu,'Label','New Figure','Callback','figure');
uimenu(figmenu,'Label','ODE Solver','Callback','SolveODEmethod');
uimenu(figmenu,'Label','Save','Callback','save');
uimenu(figmenu,'Label','Quit','Callback','exit','Separator','on','Accelerator','Q');


%%% AXES %%%
haxes = gca;
set(haxes,'Position',[Length.axes .1 0.75 0.8]);
% set(get(haxes,'Title'),'String','RESULTS','FontSize',FontSize.axes.title);


%%% START PANEL %%%
% Create the button group.
start.buttons = 3;
bsize = ButtonSize(start.buttons);
start.panel = uibuttongroup('Position',[.02 .85 Length.panel Button.height*start.buttons
    ]);
set(start.panel,'Title','START','FontSize',FontSize.panel.title);
% Create push buttons in the button group.
```

```
start.new  = uicontrol('Parent',start.panel,'Style','PushButton','Units','normalized','
    String','New Problem', 'Position',bsize(1,:),'Callback',@start_callback);
start.load = uicontrol('Parent',start.panel,'Style','PushButton','Units','normalized','
    String','Load Problem','Position',bsize(2,:),'Callback',@start_callback);
start.save = uicontrol('Parent',start.panel,'Style','PushButton','Units','normalized','
    String','Save Problem','Position',bsize(3,:),'Callback',@start_callback);
% Initialize some button group properties.
set(start.save,'Enable','Off');


%%% SOLVER PANEL %%%
% Create the button group.
solver.buttons = 6;
bsize = ButtonSize(solver.buttons);
solver.panel = uibuttongroup('Position',[.02 .6 Length.panel Button.height*solver.
    buttons],'SelectionChangeFcn',@solver_select);
set(solver.panel,'Visible','Off');
set(solver.panel,'Title','SOLVER','FontSize',FontSize.panel.title);
% Create radio buttons in the button group.
solver.fmincon = uicontrol('Parent',solver.panel,'Style','Radio','Units','normalized','
    String','FMINCON','Position',bsize(1,:));
solver.knitro  = uicontrol('Parent',solver.panel,'Style','Radio','Units','normalized','
    String','KNITRO', 'Position',bsize(2,:));
solver.ipopt   = uicontrol('Parent',solver.panel,'Style','Radio','Units','normalized','
    String','IPOPT',  'Position',bsize(3,:));
solver.iclocs  = uicontrol('Parent',solver.panel,'Style','Radio','Units','normalized','
    String','ICLOCS', 'Position',bsize(4,:));
solver.worhp   = uicontrol('Parent',solver.panel,'Style','Radio','Units','normalized','
    String','WORHP',  'Position',bsize(5,:));
% Create push buttons in the button group to define options.
solver.options = uicontrol('Parent',solver.panel,'Style','PushButton','Units','
    normalized','String','Options','Position',bsize(solver.buttons,:));
% Initialize some button group properties.
set(solver.panel,'SelectedObject',[]);  % No selection
set(solver.options,'Enable','off');


%%% ODE METHOD PANEL %%%
% Create the button group.
ODE.buttons = 4;
bsize = ButtonSize(ODE.buttons);
ODE.panel = uibuttongroup('Position',[.02 .43 Length.panel Button.height*ODE.buttons]);
set(ODE.panel,'Visible','Off');
set(ODE.panel,'Title','ODE Method','FontSize',FontSize.panel.title,'SelectionChangeFcn',
    @method_select);
% Create radio buttons in the button group.
ODE.button_Euler = uicontrol('Parent',ODE.panel,'Style','Radio','Units','normalized','
    String','Euler Method',          'Position',bsize(1,:));
ODE.button_Heun  = uicontrol('Parent',ODE.panel,'Style','Radio','Units','normalized','
    String','Heun Method',           'Position',bsize(2,:));
ODE.button_RK2   = uicontrol('Parent',ODE.panel,'Style','Radio','Units','normalized','
    String','2nd Runge-kutta Method','Position',bsize(3,:));
ODE.button_RK4   = uicontrol('Parent',ODE.panel,'Style','Radio','Units','normalized','
    String','4th Runge-kutta Method','Position',bsize(4,:));
% Initialize some button group properties.
set(ODE.panel,'SelectedObject',[]);  % No selection
%set(ODE.panel,'SelectedObject',ODE.button_Euler);  % default selection


%%% INITIAL CONDITION PANEL %%%
% Create the button group.
initsol.buttons = 1;
bsize = ButtonSize(initsol.buttons);
```

```
initsol.panel = uibuttongroup('Position',[.02 .37 Length.panel Button.height*initsol.
    buttons]);
set(initsol.panel,'Visible','Off');
% Create radio buttons in the button group.
bsize(1,3) = 0.49;
initsol.cold = uicontrol('Parent',initsol.panel,'Style','PushButton','Units','normalized
    ','String','Cold Start','Position',bsize,'Callback',@initsol_callback);
bsize(1,1) = 0.5;
initsol.hot  = uicontrol('Parent',initsol.panel,'Style','PushButton','Units','normalized
    ','String','Hot Start', 'Position',bsize,'Callback',@initsol_callback);
% Initialize some button group properties.


%%% SOLVE BUTTON %%%
% Create the button group.
solve.buttons = 1;
bsize = ButtonSize(solve.buttons);
solve.panel = uibuttongroup('Position',[.02 .3 Length.panel Button.height]);
set(solve.panel,'Visible','Off');
% Create radio buttons in the button group.
solve.button = uicontrol('Parent',solve.panel,'Style','PushButton','Units','normalized',
    'String','SOLVE','Position',bsize,'Enable','On');
set(solve.button,'Enable','On','Callback',@singlebutton_callback);


%%% PLOT PANEL %%%
% Create the button group.
plots.buttons = 5;
bsize = ButtonSize(plots.buttons);
plots.panel = uibuttongroup('Position',[.02 .03 Length.panel Button.height*plots.buttons
    ]);
set(plots.panel,'Visible','Off');
set(plots.panel,'Title','PLOT','FontSize',FontSize.panel.title);
% Create radio buttons in the button group.
plots.state   = uicontrol('Parent',plots.panel,'Style','Radio','Units','normalized','
    String','State Variables',  'Position',bsize(1,:));
plots.control = uicontrol('Parent',plots.panel,'Style','Radio','Units','normalized','
    String','Control Variables','Position',bsize(2,:));
plots.traject = uicontrol('Parent',plots.panel,'Style','Radio','Units','normalized','
    String','Trajectory',       'Position',bsize(3,:));
plots.other   = uicontrol('Parent',plots.panel,'Style','Radio','Units','normalized','
    String','--other option--', 'Position',bsize(4,:));
plots.button  = uicontrol('Parent',plots.panel,'Style','PushButton','Units','normalized'
    ,'String','PLOT','Position',bsize(5,:),'Interruptible','On','Callback',
    @singlebutton_callback);
% Initialize some button group properties.
% set(plots.panel,'SelectionChangeFcn',@selcbk);
set(plots.panel,'SelectedObject',[]);  % No selection


%%% CONTEXT MENU %%%
cmenu = uicontextmenu('Parent',hfig,'Position',[10 215]);
mh1 = uimenu(cmenu,'Label','Item 1','Callback',@uimenu_callback);
mh2 = uimenu(cmenu,'Label','Item 2','Callback',@uimenu_callback);
mh3 = uimenu(cmenu,'Label','Item 3','Callback',@uimenu_callback);
% set(hfig,'UIContextMenu',cmenu);
set(haxes,'UIContextMenu',cmenu);
set(cmenu,'Visible','on');

set(hfig,'Visible','on');
end


%%% SINGLE BUTTONS CALLBACKS %%%
```

```
function singlebutton_callback(hObject,event)
% HELP ...
%
    global start solver ODE solve plots;
    switch lower(get(hObject,'String'))

    %%% SOLVE BUTTON CALLBACK
    case {'solve'}
        set(hObject, 'String', 'SOLVED');
        set(hObject, 'Enable', 'Off');
        set(plots.panel,  'Visible','On');

    %%% LOAD PROBLEM BUTTON CALLBACK
    case {'plot'}


    %%% OTHERWISE
    otherwise
        disp('Unknown button.')
    end
end


%%% START PANEL CALLBACKS %%%
function start_callback(hObject,event)
% HELP ...
%
    global start solver ODE solve plots initsol;
    switch lower(get(hObject,'String'))

    %%% NEW PROBLEM BUTTON CALLBACK
    case {'new problem'}
        dee;
        set(start.save,   'Enable','On');
        set(solver.panel, 'Visible','On');
        set(ODE.panel,    'Visible','On');
        set(solve.panel,  'Visible','On');
        set(initsol.panel,'Visible','On');

    %%% LOAD PROBLEM BUTTON CALLBACK
    case {'load problem'}
        [load_file,load_path] = uigetfile({'*.oct;*.dat','Data files (*.oct,*.dat)'; ...
            '*.m;*.fig;*.mat;*.mdl','MATLAB Files (*.m,*.fig,*.mat,*.mdl)';'*.*','All
                Files (*.*)'},get(hObject,'String'));
        if isequal(load_file,0) || isequal(load_path,0)
            disp('User selected Cancel');
        else
            set(start.load,'UserData',fullfile(load_path,load_file));
            set(start.save,'UserData',fullfile(load_path,load_file));
            set(start.save,   'Enable','On');
            set(solver.panel, 'Visible','On');
            set(ODE.panel,    'Visible','On');
            set(solve.panel,  'Visible','On');
            set(initsol.panel,'Visible','On');
            load(fullfile(load_path,load_file),'-mat');
        end;

    %%% SAVE PROBLEM BUTTON CALLBACK
    case {'save problem'}
        if isempty(get(hObject,'UserData'))
            [save_file,save_path] = uiputfile({'*.oct;*.dat','Data files (*.oct,*.dat)';
                ...
                '*.m;*.fig;*.mat;*.mdl','MATLAB Files (*.m,*.fig,*.mat,*.mdl)';'*.*','
                    All Files (*.*)'},get(hObject,'String'));
```

```
205             else
                [save_file,save_path] = uiputfile({'*.oct;*.dat','Data files (*.oct,*.dat)';
                    ...
207                 '*.m;*.fig;*.mat;*.mdl','MATLAB Files (*.m,*.fig,*.mat,*.mdl)';'*.*','
                        All Files (*.*)'},get(hObject,'String'),get(hObject,'UserData'));
            end
209
            if isequal(save_file,0) || isequal(save_path,0)
211             disp('User selected Cancel')
            else
213             set(start.load,'UserData',fullfile(save_path,save_file));
                set(start.save,'UserData',fullfile(save_path,save_file));
215             save(fullfile(save_path,save_file),'-mat','-v7');
            end
217
        %%% OTHERWISE
219     otherwise
            disp('Unknown button.')
221     end

223 end


225
    %%% SOLVER PANEL CALLBACKS %%%
227 function solver_select(hObject,event)
    % HELP ...
229 %
        global start solver ODE solve plots initsol;
231     switch lower(get(get(hObject,'SelectedObject'),'String'))

233     %%% FMINCON BUTTON CALLBACK
        case {'fmincon'}
235         disp('fmincon')
            set(solver.options,'Enable','On');
237
        %%% HOT START BUTTON CALLBACK
239     case {'knitro'}
            disp('knitro')
241         set(solver.options,'Enable','On');

243     %%% COLD START BUTTON CALLBACK
        case {'ipopt'}
245         disp('ipopt')
            set(solver.options,'Enable','On');
247
        %%% HOT START BUTTON CALLBACK
249     case {'worhp'}
            disp('worhp')
251         set(solver.options,'Enable','On');

253     %%% OTHERWISE
      otherwise
255         disp('Unknown button.')
        end
257 end


259
    %%% SOLVER PANEL CALLBACKS %%%
261 function odemethod_select(hObject,event)
    % HELP ...
263 %
        global start solver ODE solve plots initsol;
265     switch lower(get(get(hObject,'SelectedObject'),'String'))
```

```
267     %%% FMINCON BUTTON CALLBACK
        case {'fmincon'}
269         disp('fmincon')
            set(solver.push_fmincon,'Enable','On');
271
        %%% KNITRO BUTTON CALLBACK
273     case {'knitro'}
            disp('knitro')
275
        %%% IPOPT BUTTON CALLBACK
277     case {'ipopt'}
            disp('ipopt')
279
        %%% ICLOCS BUTTON CALLBACK
281     case {'iclocs'}
            disp('iclocs')
283
        %%% WORHP BUTTON CALLBACK
285     case {'worhp'}
            disp('worhp')
287
        %%% OTHERWISE
289   otherwise
            disp('Unknown button.')
291     end
    end
293

295 %%% INITIAL SOLUTION PANEL CALLBACKS %%%
    function initsol_callback(hObject,event)
297 % HELP ...
    %
299     global start solver ODE solve plots initsol;
        switch lower(get(hObject,'String'))
301
        %%% COLD START BUTTON CALLBACK
303     case {'cold start'}

305     %%% HOT START BUTTON CALLBACK
        case {'hot start'}
307         [initsol.hotstart_file, initsol.hotstart_path] = uigetfile({'*.dat','Data files
                (*.dat)'; ...
                '*.m;*.fig;*.mat;*.mdl','MATLAB Files (*.m,*.fig,*.mat,*.mdl)'; '*.*','All
                    Files (*.*)'},get(hObject,'String'));
309         if isequal(initsol.hotstart_file,0) || isequal(initsol.hotstart_path,0)
                disp('User selected Cancel');
311         else
                set(solve.panel, 'Visible','On');
313         end;

315     %%% OTHERWISE
      otherwise
317         disp('Unknown button.')
        end
319 end


321
    %%% CONTEXT MENU CALLBACKS %%%
323 function uimenu_callback(hObject,event)
    % HELP ...
325 %
    global start solver ODE solve plots initsol;
327     switch lower(get(hObject,'Label'))
```

```
329     %%% NEW PROBLEM BUTTON CALLBACK
        case {'item 1'}
331
333     %%% LOAD PROBLEM BUTTON CALLBACK
        case {'item 2'}
335
337     %%% SAVE PROBLEM BUTTON CALLBACK
        case {'item 3'}
339
341     %%% OTHERWISE
        otherwise
343         disp('Unknown option.')
        end
345
    end
347
349
    function bsize = ButtonSize(buttons)
351 % HELP ...
    %
353     bsize = zeros(buttons,4);
        bsize(:,1) = 0.01;
355     bsize(:,3) = 0.98;
        bsize(:,4) = 1/buttons;
357     for i = 1:buttons;
            bsize(i,2) = (buttons-i)/buttons;
359     end;
    end
```

../MatlabOCT/OptimalControlToolbox.m

A.2 Car–Like: AMPL Code

```
#!/opt/ampl/ampl
# (c) 2012 - L.T.Paiva and F.A.C.C Fontes

reset;
shell "clear";

### PARAMETERS ###
param tf := 1.0 ;
param n  := 40 ;
param h  := tf/n ;

param k     := 10.0 ;
param xbar  :=  5.0 ;
param ybar  :=  1.0 ;
param rfmax :=  0.5 ;
param pi    := 4*atan(1.0) ;

param x0     :=  0.0 ;
param y0     :=  0.0 ;
#param theta0 :=  pi/2.0 ;
param theta0 :=  0.0 ;
param xf     := 10.0 ;
param yf     :=  0.0 ;
#param thetaf := -pi/2.0 ;
param thetaf := 0.0 ;

param umax := 2.0 ;
param umin := 0.0 ;
param wmax := 0.7 ;
param wmin := -wmax ;
param vmin := 1.0 ;
param vmax := 2*sqrt((xf-x0)^2+(yf-y0)^2) ;

param t0     := 0.0 ;

### STATE VARIABLES ###
var x {i in 0..n} ;
var y {i in 0..n} ;
var theta {i in 0..n} ;
var t {i in 0..n} ;

### CONTROL VARIABLES ###
var u {i in 0..n};
var v ;
var w {i in 0..n};

### OBJECTIVE FUNCTION ###
minimize obj: t[n] ;

#### CONSTRAINTS ####

### INITIAL VALUES ###
s.t. ivx     : x[0] = x0 ;
s.t. ivy     : y[0] = y0 ;
s.t. ivtheta : theta[0] = theta0 ;
s.t. ivt     : t[0] = t0 ;

### OBSTACULO ###
var y2 {i in 0..n}  = ybar - k *(x[i] - xbar)^2 ;
s.t. mu_y  {i in 0..n} : y[i]  >= y2[i] ;

### FINAL RESTRICTION ###
```

```
63 var rf = (x[n]-xf)^2 + (y[n]-yf)^2 + (theta[n]-thetaf)^2 ;
   s.t. mu_rf : rf <= rfmax ;
65
   # auxiliary functions for improved Euler
67 var fx {i in 0..n}     = v*u[i]*cos(theta[i]) ;
   var fy {i in 0..n}     = v*u[i]*sin(theta[i]) ;
69 var ftheta {i in 0..n} = v*u[i]*w[i] ;

71 ### EULER Method ###
   s.t. lx {i in 0..n-1} : x[i+1]=x[i] + h*fx[i] ;
73 s.t. ly {i in 0..n-1} : y[i+1]=y[i] + h*fy[i] ;
   s.t. ltheta {i in 0..n-1} : theta[i+1]=theta[i] + h*ftheta[i] ;
75 s.t. lt {i in 0..n-1} : t[i+1]=t[i] + h*v ;

77 ### Control constraints
   s.t. mu_v {i in 0..n} : vmin <= v <= vmax ;
79 s.t. mu_u {i in 0..n} : umin <= u[i] <= umax ;
   s.t. mu_w {i in 0..n} : wmin <= w[i] <= wmax ;
81
   ### SOLVER ###
83 #option solver knitroampl ;
   #option knitro_options "maxit=1000 opttol=1e-8 debug=2 feastol_abs=1e-5";
85
   option solver ipopt;
87 option ipopt_options "max_iter=999 acceptable_tol=1e-8";

89 solve;

91 ### SAVE RESULTS ###
   printf {i in 0..n} "%18.10f %18.10f\n", x[i], y[i]    > 'traj.dat';
93 printf {i in 0..n} "%18.10f %18.10f\n", i*h, x[i]     > 'x.dat';
   printf {i in 0..n} "%18.10f %18.10f\n", i*h, y[i]     > 'y.dat';
95 printf {i in 0..n} "%18.10f %18.10f\n", i*h, y2[i]    > 'y2.dat';
   printf {i in 0..n} "%18.10f %18.10f\n", i*h, theta[i] > 'theta.dat';
97 printf {i in 0..n} "%18.10f %18.10f\n", i*h, t[i]     > 't.dat';

99 printf "%18.10f\n", v   > 'v.dat';
   printf {i in 0..n-1} "%18.10f %18.10f\n", i*h, u[i]  > 'u.dat';
101 printf {i in 0..n-1} "%18.10f %18.10f\n", i*h, w[i]  > 'w.dat';

103 printf {i in 0..n-1} "%18.10f", obj  > 'obj.dat';
```

A.3 HIV: ICLOCS Code

```
clear all; close all; clc; format compact;

% Fetch the problem definition
[problem,guess] = OptChemotherapeuticStrat;
% Get options and solver settings
options = settings;

% Format for NLP solver
[infoNLP,data] = transcribeOCP(problem,guess,options);
% Solve the NLP
[solution,status] = solveNLP(infoNLP,data);
% Output solutions
output(solution,options,data);

plotHIV;
```

../Applications/OptChemotherapeuticStrat/iclocs/main.m

```
function [problem,guess] = OptChemotherapeuticStrat

%Initial Time. t0<tf
problem.time.t0 = 0;

% Final time. Let tf_min=tf_max if tf is fixed.
problem.time.tf_min = 500;
problem.time.tf_max = problem.time.tf_min;
guess.tf = [];

% Parameters bounds. pl=< p <=pu
problem.parameters.pl = [];
problem.parameters.pu = [];
guess.parameters = [];

% some parameters
m = [0.02 0.02 0.24 2.4 2.4E-5 3E-3];
r = 0.03; N = 1200; tmax = 1.5E3; s = 10; % miu_T*tmax= 30>10 =s

% Initial conditions for the system.
%%% for infection by free virus
% Ta0 = tmax/2.0 * (1 - m(1)/r + sqrt((1-m(1)/r)^2 + 4*s/(r*tmax))); Tl0=0; Ti0=0; V0=1E
    -3;
%%% for infection by both infected cells and virus
Ta0 = 975; Tl0 = 0.05; Ti0 = 0.01; V0 = 1E-3;
problem.states.x0 = [Ta0 Tl0 Ti0 V0];

% Initial conditions for system. Bounds if x0 is free s.t. x0l=< x0 <=x0u
problem.states.x0l = [Ta0 Tl0 Ti0 V0];
problem.states.x0u = [Ta0 Tl0 Ti0 V0];

% State bounds. xl=< x <=xu
problem.states.xl = [-inf -inf -inf -inf];
problem.states.xu = [inf inf inf inf];

% Terminal state bounds. xfl=< xf <=xfu
Taf = 950; Tlf = 5e-5; Tif = 5E-5; Vf = 0.5;
problem.states.xf = [Taf Tlf Tif Vf];
problem.states.xfl = [-inf -inf -inf -inf];
problem.states.xfu = [inf inf inf inf];

% Guess the state trajectories with [x0 xf]
guess.states = [problem.states.x0 ; problem.states.xf];
```

```
43
% Number of control actions N
45 problem.inputs.N = 0;

47 % Input bounds (usual these are arrays)
problem.inputs.ul = 0;
49 problem.inputs.uu = 1;

51 % Guess the input sequences with [u0 uf]
guess.inputs = [];
53
% Choose the set-points if required
55 problem.setpoints.states = [];
problem.setpoints.inputs = [];
57
% Bounds for path constraint function gl =< g(x,u,p,t) =< gu
59 problem.constraints.gl = [];
problem.constraints.gu = [];
61
% Bounds for boundary constraints bl =< b(x0,xf,u0,uf,p,t0,tf) =< bu
63 problem.constraints.bl = [];
problem.constraints.bu = [];
65
% store the necessary problem parameters used in the functions
67 problem.data = [m r N tmax s];

69 % Get function handles and return to Main.m
problem.functions={@L,@E,@f,@g,@b};
71
%---------- END OF CODE ----------
73


75 function stageCost=L(x,xr,u,ur,p,t,data)

77 B = 100;
coef = 0.5;
79
stageCost = -x(:,1) + coef * B * u(:,1).*u(:,1);
81
%---------- END OF CODE ----------
83


85 function boundaryCost=E(x0,xf,u0,uf,p,tf,data)

87 boundaryCost=tf;

89 %---------- END OF CODE ----------
91


function dx = f(x,u,p,t,data)
93
x1 = x(:,1); x2 = x(:,2); x3 = x(:,3); x4 = x(:,4);
95 u1 = u(:,1);

97 m = data(1:6);
r = data(7); N = data(8); tmax = data(9); s = data(10);
99
dx(:,1) = s./(1+x4) - m(1).*x1 + r.*x1.*(1 - (x1+x2+x3)./tmax) - m(5).*x4.*x1 ;
101
dx(:,2) = m(5).*x4.*x1 - m(2).*x2 - m(6).*x2;
103
dx(:,3) = m(6).*x2 - m(3).*x3;
105
dx(:,4) = (1-u1).*N.*m(3).*x3 - m(5).*x4.*x1 - m(4).*x4;
```

```
% dx(:,4) = u1.*N.*m(3).*x3 - m(5).*x4.*x1 - m(4).*x4;

%---------- END OF CODE ----------


function c=g(x,u,p,t,data)

c=[];

%---------- END OF CODE ----------

function bc=b(x0,xf,u0,uf,p,tf,data)

bc=[];

%---------- END OF CODE ----------
```

../Applications/OptChemotherapeuticStrat/iclocs/OptChemotherapeuticStrat.m

```
function options = settings

options.transcription='hermite';
options.derivatives='numeric';
options.hessianFD='central';

options.perturbation.H=[];
options.perturbation.J=[];

% NLP solver
options.NLPsolver='ipopt';

% IPOPT settings (if required)
options.ipopt.tol=1e-9;
options.ipopt.print_level=5;
options.ipopt.max_iter=200;
options.ipopt.mu_strategy ='adaptive';
options.ipopt.hessian_approximation='exact';

% fmincon settings (if required)
 options.fmincon=optimset;

% Automatic scaling (recommended)
options.scaling=1;

% Display computation time
options.print.time=1;

% Display relative local discretization error
options.print.relative_local_error=1;

% Display cost
options.print.cost=1;

% Plot states
options.plot.states=1;

% Plot inputs
options.plot.inputs=1;

% Plot Lagrange multipliers
options.plot.multipliers=1;

```

```
% Direct transcription settings
options.nodes=1000;
options.tau=0;
options.ODEsolver='cvodes';

% CVODES settings (if required)
Method='Adams';
Solver='Newton';
```

../Applications/OptChemotherapeuticStrat/iclocs/settings.m

```
hfig1  = figure(1);
haxis1 = get(hfig1,'CurrentAxes');
hplot1 = get(haxis1,'Children');

T  = [get(hplot1(4),'Xdata')' get(hplot1(4),'Ydata')'];
Tl = [get(hplot1(3),'Xdata')' get(hplot1(3),'Ydata')'];
Ti = [get(hplot1(2),'Xdata')' get(hplot1(2),'Ydata')'];
V  = [get(hplot1(1),'Xdata')' get(hplot1(1),'Ydata')'];

hfig2  = figure(2);
haxis2 = get(hfig2,'CurrentAxes');
hplot2 = get(haxis2,'Children');

u = [get(hplot2,'Xdata')' get(hplot2,'Ydata')'];

data = [T Tl Ti V u];
titles = char('Uninfected cells','Latently infected cells','Actively infected cells','
    Virus','Chemotherapy');
sz = size(titles);  titlesize = sz(1)*sz(2);  titlen = sz(1);
legends = char('S(t)','E(t)','I(t)','R(t)','u(t)');
sz = size(legends); legendsize = sz(1)*sz(2); legendn = sz(1);
locations = char('NorthWest','NorthEast','NorthEast','NorthEast','NorthWest','NorthEast'
    );
sz = size(locations); locationsize = sz(1)*sz(2); locationn = sz(1);

hfig = figure();
for i = 1:size(data,2)/2
    subplot(2,3,i);
    hplot = plot(data(:,2*i-1),data(:,2*i),'r-');
    title(strtrim(titles(i:titlen:titlesize)),'FontSize',20);
    xlabel('Time (days)','FontSize',18);
    set(gca,'FontSize',17);
    % legend(strtrim(legends(i:legendn:locations)),'Location',locations(i:locationn:
        locationsize));
    set(gca,'Xlim',[0 500]);
    grid;
end
```

../Applications/OptChemotherapeuticStrat/iclocs/plotHIV.m

```
function [tf,Xf] = solveODEsys

close all; clc;
format long;

tf = 800;
Ta0 = 1000; Tl0 = 0; Ti0 = 0; V0 = 1E-3;

options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-4 1e-4]);
[t,X] = ode45(@ODEsystem,[0 tf],[Ta0 Tl0 Ti0 V0],options);

hfig = figure();
```

```
13 plot(t,X(:,1),'r-',t,X(:,2),'g-.',t,X(:,3),'b.',t,X(:,4),'k.');

15 Xf = X(length(X),:);
   disp(Xf);
17 end

19 function dx = ODEsystem(t,x)
       dx = zeros(4,1);
21
       m = [0.02 0.02 0.24 2.4 2.4E-5 3E-3];
23     r = 0.03; N = 1200; tmax = 1.5E3; s = 10;
       u = 0;
25
       dx(1) = s/(1+x(4)) - m(1)*x(1) + r*x(1)*(1-(x(1)+x(2)+x(3))/tmax) - m(5)*x(4)*x(1);
27     dx(2) = m(5)*x(4)*x(1) - m(2)*x(2) - m(6)*x(2);
       dx(3) = m(6)*x(2) - m(3)*x(3);
29     dx(4) = (1-u)*N*m(3)*x(3) - m(5)*x(4)*x(1) - m(4)*x(4);
   end
```

../Applications/OptChemotherapeuticStrat/iclocs/solveODEsys.m