CasADi introduction

Course on Numerical Optimal Control, 4-13 August 2014

Joel Andersson

Freiburg, 4 August 2014

Outline

What is CasADi?

A general-purpsose software framework for quick, yet efficient, implementation of algorithms for numeric optimization

In particular

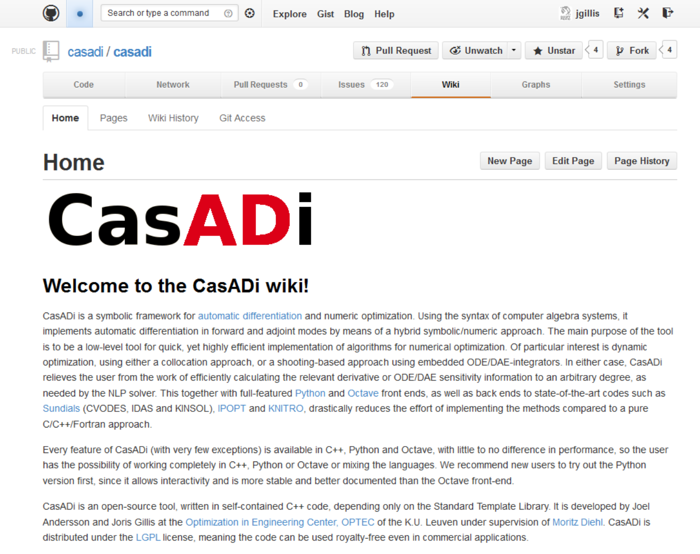
Facilitates the solution of optimal control problems (OCPs) using a variety of different methods

- OCPs: Wednesday morning
- *Facilitates*, not actually *solve* the OCPs

CasADi

Where CasADi lives

casadi.org → github.com



Home

CasADi

Welcome to the CasADi wiki!

CasADi is a symbolic framework for automatic differentiation and numeric optimization. Using the syntax of computer algebra systems, it implements automatic differentiation in forward and adjoint modes by means of a hybrid symbolic/numeric approach. The main purpose of the tool is to be a low-level tool for quick, yet highly efficient implementation of algorithms for numerical optimization. Of particular interest is dynamic optimization, using either a collocation approach, or a shooting-based approach using embedded ODE/DAE-integrators. In either case, CasADi relieves the user from the work of efficiently calculating the relevant derivative or ODE/DAE sensitivity information to an arbitrary degree, as needed by the NLP solver. This together with full-featured Python and Octave front ends, as well as back ends to state-of-the-art codes such as Sundials (CVODES, IDAS and KINSOL), IPOPT and KNITRO, drastically reduces the effort of implementing the methods compared to a pure C/C++/Fortran approach.

Every feature of CasADi (with very few exceptions) is available in C++, Python and Octave, with little to no difference in performance, so the user has the possibility of working completely in C++, Python or Octave or mixing the languages. We recommend new users to try out the Python version first, since it allows interactivity and is more stable and better documented than the Octave front-end.

CasADi is an open-source tool, written in self-contained C++ code, depending only on the Standard Template Library. It is developed by Joel Andersson and Joris Gillis at the Optimization in Engineering Center, OPTEC of the K.U. Leuven under supervision of Moritz Diehl. CasADi is distributed under the LGPL license, meaning the code can be used royalty-free even in commercial applications.

More about CasADi

- Free & open-source (LGPL), also for commercial use
- Use from C++ or Python
- Project started in December 2009
  - Original motivation: Solve OCPs with models from *Modelica*
  - Joris Gillis joined spring 2010
- Since 2012, a growing number of users
- Now a mature project at version 2.0, released 25 July 2014

Today's exercise

- Symbolic framework
- How to solve nonlinear programs (NLPs)

Outline

What you need to know

- CasADi allows you to symbolic expressions using syntax similar to e.g. *Symbolic Math Toolbox* for MATLAB or *SymPy*.

```
from casadi import *
x = SX.sym("x")           Variable x with display name "x"
f = sqrt(x**2 + 10)
g = sin(x)
```

$f = \sqrt{x^2 + 10}$

$g = \sin(x)$

- These functions are then used to define *functions* ...

```
F = SXFunction([x],[f,g])   Defines F :
F.init()
```

$$F: \begin{array}{ccc} \mathbb{R} & \rightarrow & \mathbb{R} \times \mathbb{R} \\ (x) & \mapsto & (f, g) \end{array}$$

- ...that can e.g. be automatically differentiated using *algorithmic differentiation* (AD) $\Rightarrow$ Exercise 4, tomorrow

```
J = F.jacobian()   Defines J :
```

$$J: \begin{array}{ccc} \mathbb{R} & \rightarrow & \mathbb{R} \times \mathbb{R} \times \mathbb{R} \\ (x) & \mapsto & \left(\frac{\partial f}{\partial x}, f, g\right) \end{array}$$

Matrices in CasADi

- CasADi is *everything-is-a-matrix* (cf. MATLAB)
- All matrices are *sparse*
- Syntax is MATLAB inspired

```
SX.sym("x",2,3)      2-by-3 symbolic primitive
SX.zeros(4,5)        dense 4-by-5 matrix with all zeros
SX.sparse(4,5)       sparse (empty) 4-by-5 matrix
SX.eye(4)            4-by-4 identity matrix
```

Two symbolic types with (almost) the same syntax

- SX: *Expression graph* with scalar-valued operations
- Low overhead, for simple functions

```
x = SX.sym("x",2,2)
f = sin(x**2 + 10)
print f.shape
print x, f
```

$(2,2)$

$$\begin{bmatrix} x_0 & x_2 \\ x_1 & x_3 \end{bmatrix}, \begin{bmatrix} \sin x_0^2 + 10 & \sin x_2^2 + 10 \\ \sin x_1^2 + 10 & \sin x_3^2 + 10 \end{bmatrix}$$

- MX: *Expression graph* with matrix-valued operations
- Larger overhead, but more generic

```
x = MX.sym("x",2,3)
f = sin(x**2 + 10)
print f.shape
print f
```

$(2,2)$

$x, \quad \sin x^2 + 10$

(NB: sin *and power elementwise)*

Why?

By mixing, construct expressions (functions) that are both fast and generic

Outline

Parametric NLPs in CasADi

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x,p) \\ \text{subject to} & \begin{array}{ccc} x_{\text{lb}} \leq & x & \leq x_{\text{ub}}, \\ g_{\text{lb}} \leq & g(x,p) & \leq g_{\text{ub}}, \end{array} \end{array}$$

- $x \in \mathbb{R}^n$ is decision variable
- $p \in \mathbb{R}^m$ is fixed (and known) parameter vector
- Equality constraints: $g_{\text{lb}}^{(k)} = g_{\text{ub}}^{(k)}$ for some k.

NLP solvers in CasADi

Are *functions*: $(x_{guess}, p) \mapsto (x_{optimal}, \ldots)$