L’écosystème Julia *control-toolbox* pour le contrôle optimal

Olivier Cots – Toulouse INP, IRIT, Inria

Jean-Baptiste Caillau • Joseph Gergaud • Pierre Martinon

Café Julia, 20 novembre 2025 (https://github.com/ocots/cafe-julia)

Contexte

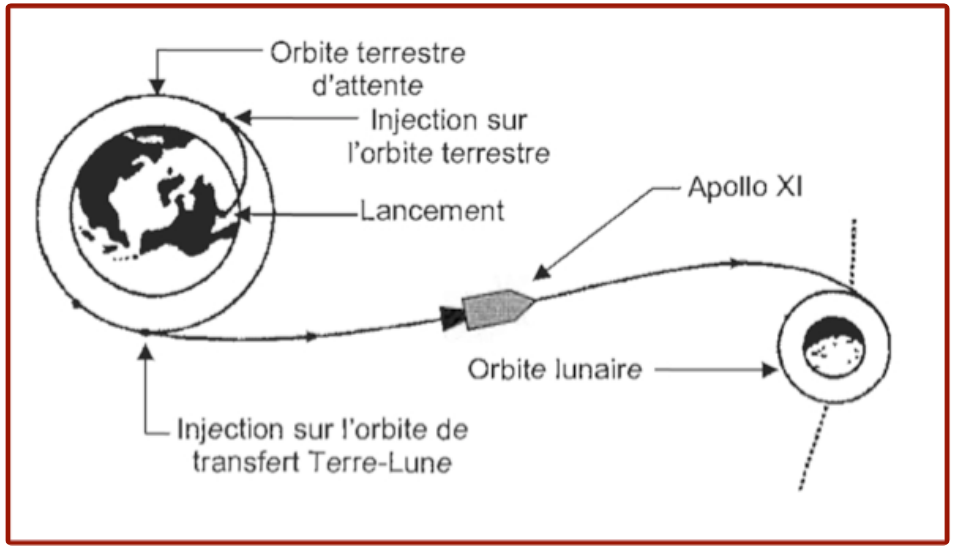
Le projet **control-toolbox** est une organisation GitHub qui rassemble plusieurs packages Julia pour modéliser, résoudre et analyser des problèmes de contrôle optimal dans les équations différentielles ordinaires déterministes. Son package principal est **OptimalControl.jl**, dont la première version a été publiée le 4 février 2023.

Dans cette présentation, nous verrons :

- Une introduction au contrôle optimal ;
- Une présentation de l'architecture du projet ;
- Un panorama de l'écosystème Julia pour le contrôle optimal ;
- Une démonstration pratique sur un exemple simple et une application plus complexe ;
- Une étude comparative de performances sur CPU et GPU ;
- Une présentation des principaux choix de conception logicielle.

1. Introduction

- Un problème de **contrôle optimal** consiste à déterminer, pour un système dynamique contrôlé, une trajectoire *optimale* et la commande associée. La paire trajectoire-commande peut être soumise à des contraintes. Une paire est dite optimale si elle minimise un critère donné, parmi toutes celles *admissibles*.



Transfert Terre–Lune – Mission Apollo XI

- **Domaine** : optimisation dans les équations différentielles ordinaires déterministes.
- **Applications** : aéronautique, aérospatial, biologie, énergie, finance, robotique, santé...

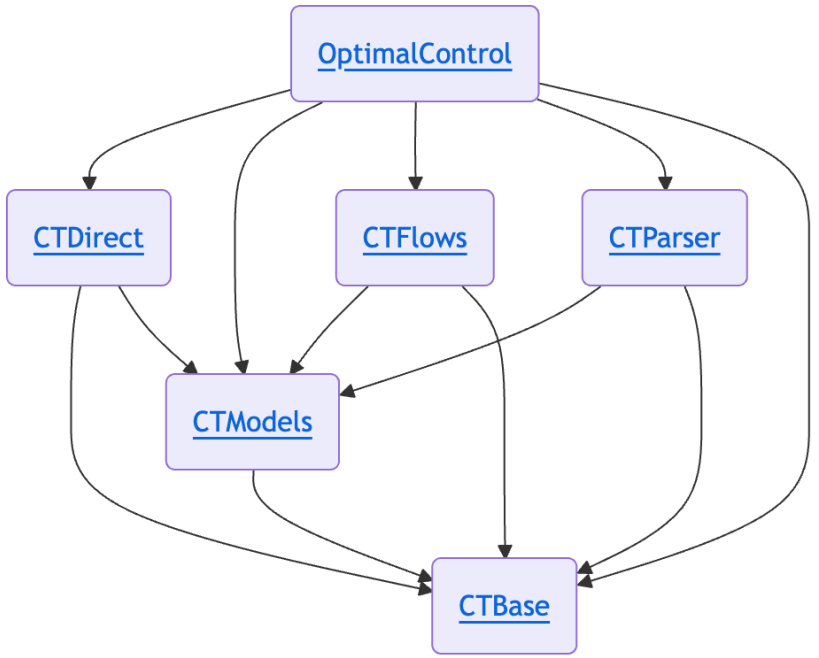
2. Panorama de *control-toolbox*

Packages principaux

- **OptimalControl.jl** : permet de modéliser, résoudre et analyser des OCPs (directes/indirectes, CPU/GPU)
- **OptimalControlProblems.jl** : bibliothèque de problèmes (modélisations JuMP, ADNLPModel, ExaModel), prête pour le benchmarking

Briques internes clés et architecture

- **CTBase.jl** : exceptions, fonctions utilitaires
- **CTModels.jl** : types des modèles, solutions, setters, getters et visualisation
- **CTDirect.jl** : discrétisation (OCP → NLP) et résolution du problème NLP
- **CTFlows.jl** : flots de systèmes dynamiques
- **CTParser.jl** : définition abstraite (Domain-Specific-Language) et parsing



💡 *Architecture modulaire*

3. Pourquoi Julia ?

Julia est un langage performant, de haut niveau.

- **Performances** : compilation JIT et fonctions *type stable* → code machine optimisé
- **Syntaxe expressive** : proche des notations mathématiques, avec prise en charge d'Unicode

```
julia> f(α, β) = α^2 + 3β^2

julia> ∇f(α, β) = [
  2α,
  6β
]

julia> ∇f(1.0, 2.0)
2-element Vector{Float64}:
  2.0
 12.0
```

🌐 Écosystème riche et spécialisé

- **AD & EDO** : ForwardDiff.jl, Zygote.jl, DifferentialEquations.jl
- **Optimisation** : JuMP.jl, JuliaSmoothOptimizers, MadNLP.jl, ExaModels.jl, ADNLPModels.jl
- **GPU** : CUDA.jl, KernelAbstractions.jl, CUDSS.jl
- **DSL** : MLStyle.jl, Moshi.jl pour le pattern matching

Avantages clés : modélisation intuitive, parallélisme SIMD/GPU, extensibilité, différentiation automatique...

4. Exemple minimaliste : double intégrateur

Problème : Trouver la paire trajectoire-commande optimale pour amener le système du double intégrateur ($\ddot{x} = u$) depuis la position $(-1, 0)$ jusqu'à la position $(0, 0)$ en minimisant l'énergie de la commande.

Formulation mathématique

$$\min_{x(\cdot),\, u(\cdot)} \quad \frac{1}{2}\int_0^1 u^2(t)\,\mathrm{d}t$$

$$\text{s.c. :} \quad \dot{x}(t) = \begin{bmatrix} x_2(t) \\ u(t) \end{bmatrix},$$

$$x(0) = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad x(1) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Utiliser l'IA →

Avec OptimalControl.jl

```
using OptimalControl

ocp = @def begin
    t ∈ [0, 1], time
    x ∈ R², state
    u ∈ R, control

    x(0) == [-1, 0]
    x(1) == [0, 0]

    ∂(x)(t) == [x₂(t), u(t)]

    0.5∫(u(t)^2) → min
end
```

Résolution et visualisation

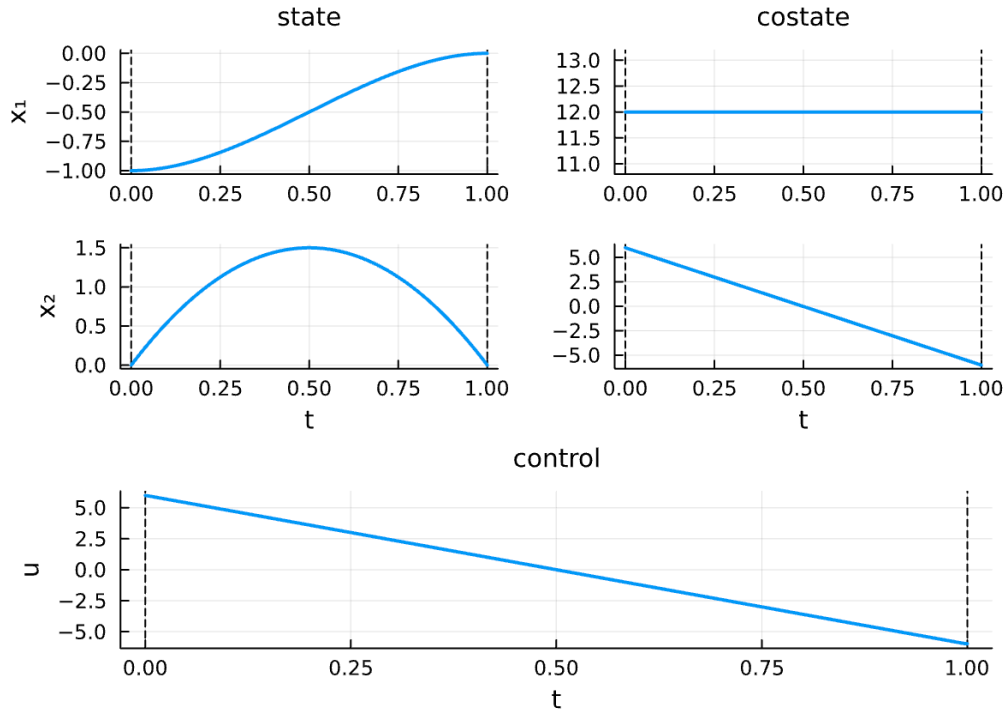
Résolution

```
using NLPModelsIpopt
sol = solve(ocp)
```

```
▫ This is OptimalControl version v1.1.1 running with: direct, adnlp, ipopt.

▫ The optimal control problem is solved with CTDirect version v0.16.2.

    ┌ The NLP is modelled with ADNLPModels and solved with NLPModelsIpopt.
    │
    ├ Number of time steps•: 250
    └ Discretisation scheme: trapeze

▫ This is Ipopt version 3.14.17, running with linear solver MUMPS 5.8.0.

Number of nonzeros in equality constraint Jacobian...:     3005
Number of nonzeros in inequality constraint Jacobian.:        0
Number of nonzeros in Lagrangian Hessian.............:      251

Total number of variables............................:     1004
                     variables with only lower bounds:        0
                variables with lower and upper bounds:        0
                     variables with only upper bounds:        0
Total number of equality constraints.................:      755
Total number of inequality constraints...............:        0
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0  1.0000000e-01 1.10e+00 3.11e-14   0.0 0.00e+00    -  0.00e+00 0.00e+00   0
```

Visualisation

```
using Plots
plot(sol)
```



Architecture SIMD (Single Instruction on Multiple Data) et Performance

- Discrétisation du problème de contrôle optimal
- Profile de perfomance sur CPU
- Problème de Goddard : Temps de résolution CPU et Temps de résolution GPU

Autres exemples d'applications

- Double intégrateur à temps min : direct et indirect
- Magnetic Resonance Imaging : direct, indirect et crochets de Poisson pour le calcul du contrôle singulier

5. Architecture logicielle et bonnes pratiques

🏗️ Séparation des responsabilités

- **Modèles** : définition, manipulation et visualisation
- **Algorithmes** : méthodes de transcription, intégrateurs
- **Interfaces** : DSL proche des mathématiques

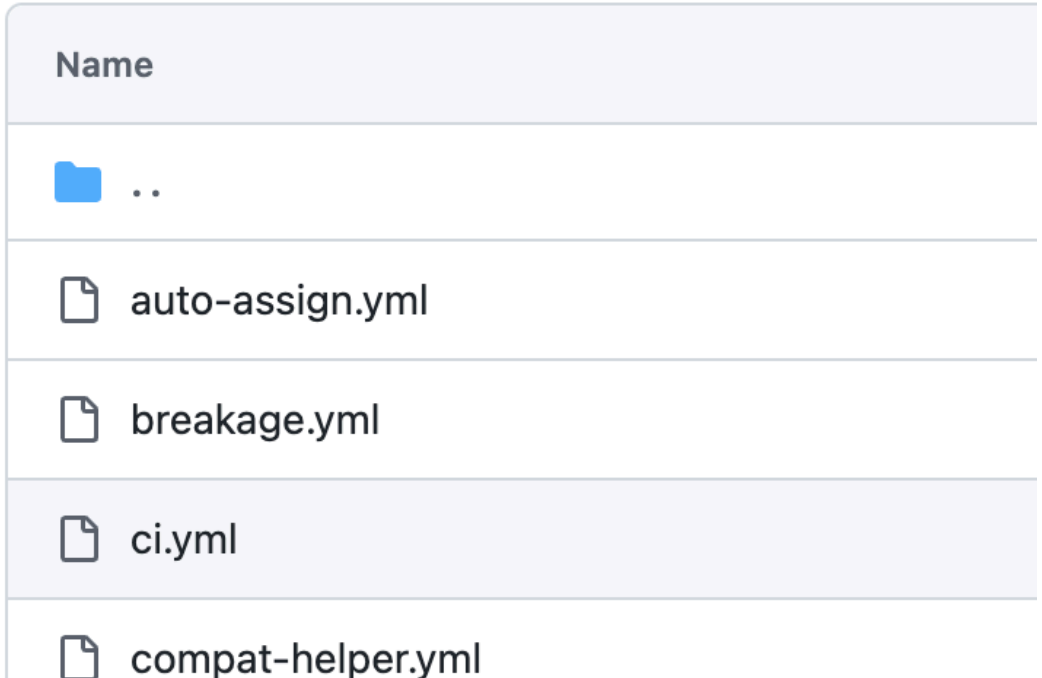
⚡ Performance

- **Différentiation automatique** et compilation Julia
- **Structure creuse** des problèmes discrétisés
- Support natif **CPU et GPU** pour le calcul haute performance

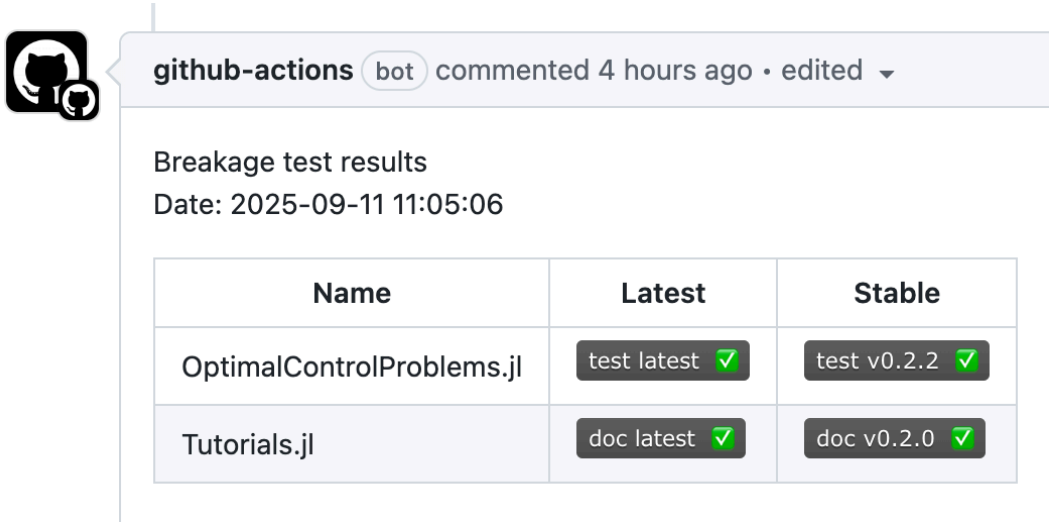
Qualité logicielle

- **Intégration continue** : tests, couverture, documentation
- **Benchmarks** : suivi des performances (à améliorer → Guix)
- **Détection d'incompatibilités** en amont
- **Workflows GitHub** centralisés : CTActions

CTActions workflows



Détection d'incompatibilités



📖 Ouverture et communauté

- Documentation complète sur control-toolbox.org : Manuels pour OptimalControl.jl, tutoriels avancés, catalogue de problèmes modélisés.
- Applications phares de la communauté (template) :
  - PWL models of gene regulatory networks (+ Binder)
  - Loss control regions in optimal control problems
  - Optimal control in Medical Resonance Imaging
  - Minimum time orbit transfer

Reproductibilité

You can download the exact environment used to build this documentation:

- 📦 Project.toml - Package dependencies
- 📋 Manifest.toml - Complete dependency tree with versions

▶ ℹ️ Version info

▶ 📦 Package status

▶ 📚 Complete manifest

Communauté

- Issues et discussions GitHub
- Contributions bienvenues

Conclusion & Perspectives

Principaux atouts

- **Unifié** : Approche unifiée pour les méthodes directes et indirectes
- **Modulaire** : Architecture flexible et extensible
- **Performant** : Exploitation des capacités de Julia
- **Communautaire** : Documentation complète et écosystème en croissance

Prochaines étapes

- Proposer de nouveaux algorithmes : méthode homotopique, raffinement de grille, multi-phases, hybride...
- Extension de la collection de problèmes modélisés et benchmarking
- Extension du DSL control-toolbox (CTDSL) : transformations abstraites, compilateur CTDSL $\longleftrightarrow$ MTK (ModelingToolKit)

Ressources

- 📚 Documentation : control-toolbox.org
- 💻 Codes sources : github.com/control-toolbox
- 📧 Contact : Olivier Cots, olivier.cots@irit.fr

control-toolbox

The control-toolbox ecosystem gathers Julia packages for mathematical control and applications. It is an outcome of a research initiative supported by the Inria Centre at Université Côte d'Azur and the Labex CIMI (Centre International de Mathématiques et Informatique de Toulouse) at Université de Toulouse and a sequel to previous developments, notably Bocop and Hampath. See also: ct gallery. The root package is OptimalControl.jl which aims to provide tools to solve optimal control problems by direct and indirect methods, both on CPU and GPU.

Getting started

To solve your first optimal control problem using OptimalControl.jl package, please check the documentation, or simply try our basic example tutorial.