

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

Olivier Cots – CNRS, Toulouse INP, IRIT

Jean-Baptiste Caillau • Joseph Gergaud • Pierre Martinon • Sophia Sed

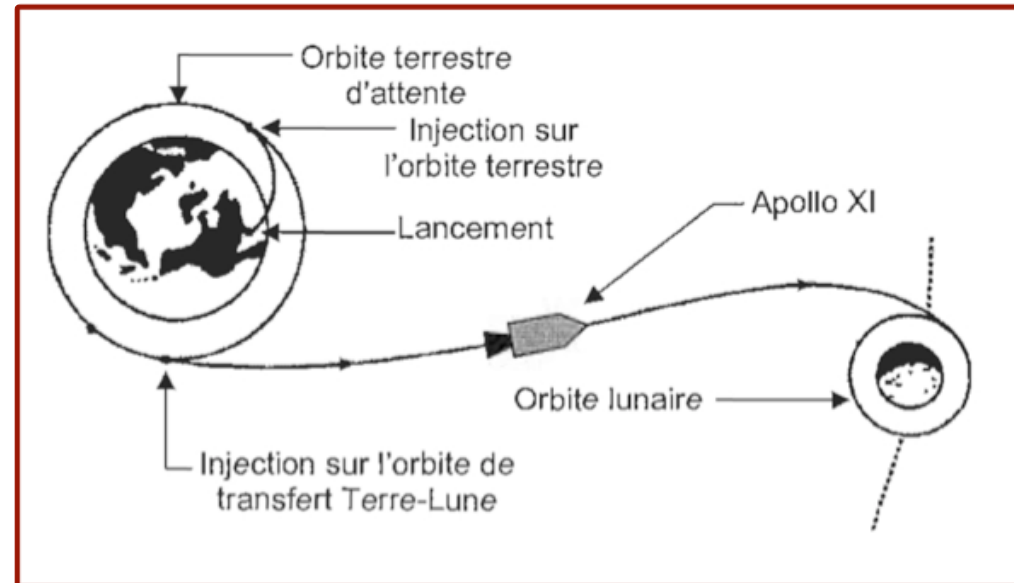
Contexte

Le projet **control-toolbox** rassemble plusieurs packages Julia pour modéliser et résoudre des problèmes de contrôle optimal.

- Package central : **OptimalControl.jl**
- Architecture modulaire et performante
- Calcul **CPU et GPU**
- Connexion fluide entre :
 - formulation mathématique,
 - simulation,
 - optimisation avancée

1. Introduction

- **Contrôle optimal** = trajectoire optimale d'un système dynamique contrôlé sous contraintes
- Domaine : math appliquées, optimisation, simulation numérique
- Applications : robotique, aéronautique, finance, énergie



Transfert Terre-Lune – Mission Apollo XI

2. Pourquoi Julia ?

Julia est un langage de haut niveau, rapide et dynamique, idéal pour le calcul scientifique et le contrôle optimal.

- **Performances** : compilation JIT et fonctions stables en type → code machine optimisé
- **Syntaxe expressive** : proche des notations mathématiques, support Unicode

```
julia> f(x1, x2) = x12 + 3x22
```

```
julia> ∇f(x1, x2) = [  
    2x1,  
    6x2  
]
```

```
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

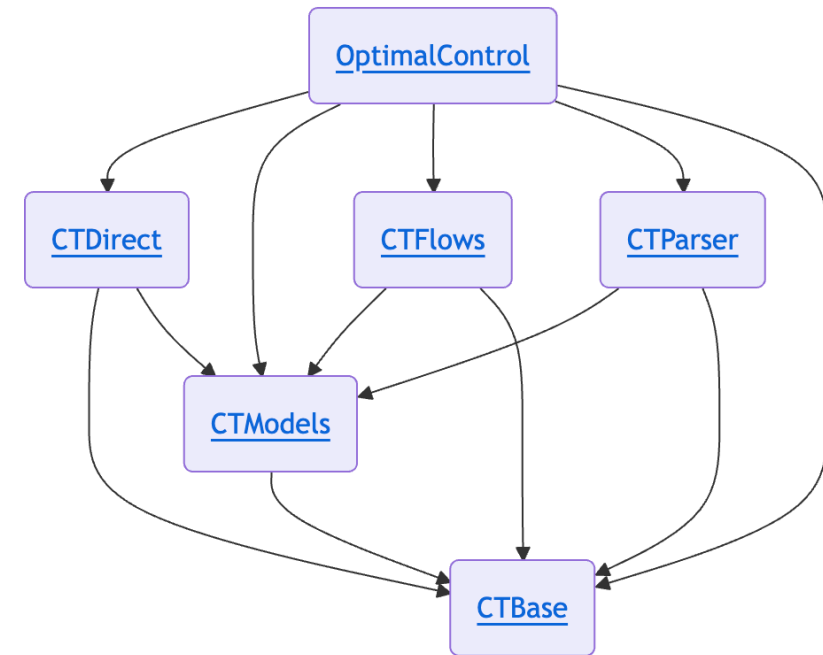
3. Panorama de *control-toolbox*

Packages principaux

- **OptimalControl.jl** : DSL pour modéliser et résoudre des OCPs (directes/indirectes, CPU/GPU)
- **OptimalControlProblems.jl** : bibliothèque de problèmes de référence, prête pour benchmarking et comparaisons

Briques internes clés et architecture

- **CTBase.jl** : exceptions, fonctions utilitaires
- **CTModels.jl** : types des modèles, solutions, setters, getters et visualisation
- **CTDirect.jl** : discrétisation et résolution
- **CTFlows.jl** : systèmes hamiltoniens et flots
- **CTParser.jl** : définition abstraite et parser



4. Exemple minimaliste : double intégrateur

Problème : Trouver le contrôle optimal pour amener un système de la position $(-1, 0)$ à $(0, 0)$ en minimisant l'énergie du contrôle.

Formulation mathématique

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

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

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

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]

    ẋ(t) == [x₂(t), u(t)]

    0.5f(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

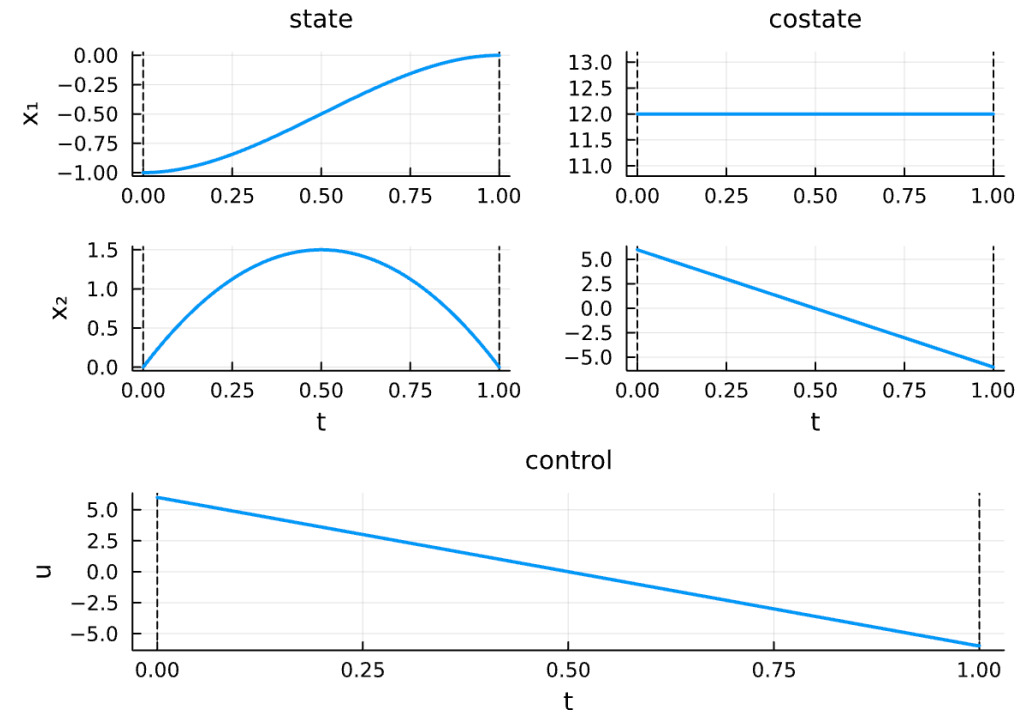
inequality constraints with lower and upper bounds: 0

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
1	5.0000000e-02	7.36e-01	2.00e-15	11.0	6.00e-02	1.00e+00	1.00e+00	1.00e+00	1

Visualisation

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



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
- **Tests unitaires** : modèles, solveurs, API
- **Benchmarks** : suivi des performances
- **Détection d'incompatibilités** avec les dépendances

Actions CI/CD

Name	
 ..	
 auto-assign.yml	
 breakage.yml	
 ci.yml	
 compat-helper.yml	

Détection d'incompatibilités



github-actions bot commented 4 hours ago • edited ▾

Breakage test results

Date: 2025-09-11 11:05:06

Name	Latest	Stable
OptimalControlProblems.jl	test latest ✓	test v0.2.2 ✓
Tutorials.jl	doc latest ✓	doc v0.2.0 ✓

Ouverture et communauté

- Documentation complète sur control-toolbox.org : Manuels pour [OptimalControl.jl](#), [Tutoriels](#) avancés, Catalogue de [problèmes modélisés](#).
- Applications phares de la communauté :
 - [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

Reproducibility

- ▶ The documentation of this package was built using these direct dependencies,
 - ▶ and using this machine and Julia version.
 - ▶ A more complete overview of all dependencies and their versions is also provided.
- You can also download the [manifest](#) file and the [project](#) file.

Communauté active

- Issues et discussions GitHub
- Contributions bienvenues
- Environnements reproductibles

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

- Extension de l'écosystème : Méthodes indirectes, Méthodes homotopiques
- Renforcement de la communauté : applications, tutoriels, algorithmes...

Ressources

-  Documentation : control-toolbox.org
-  Code source : github.com/control-toolbox
-  Contact : Olivier Cots, olivier.cots@irit.fr



ct 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](#).