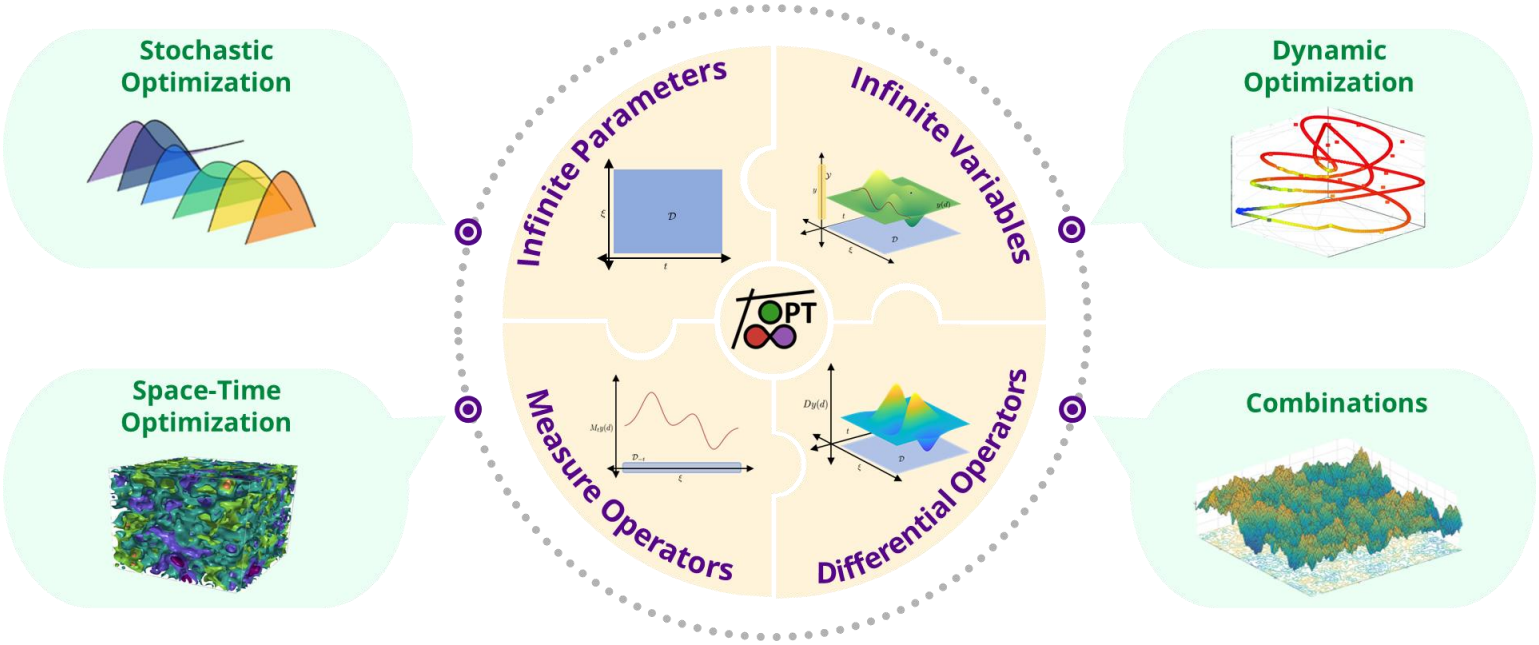
THE STATE OF InfiniteOpt

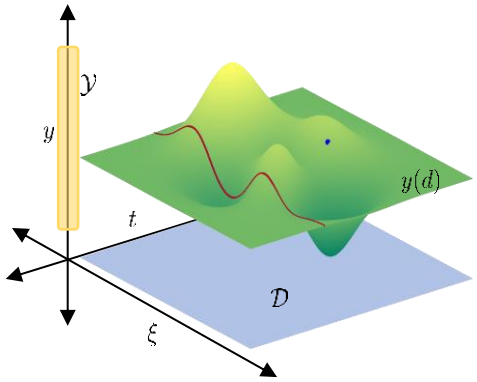
Joshua Pulsipher

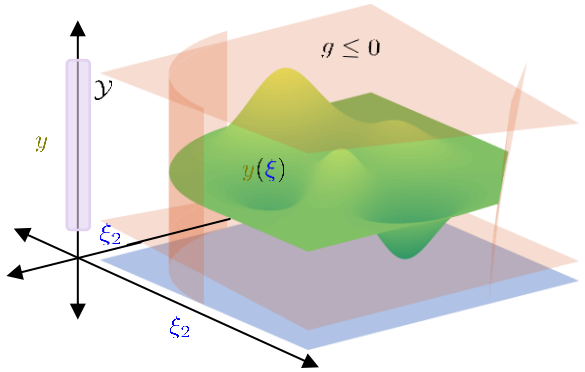


UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

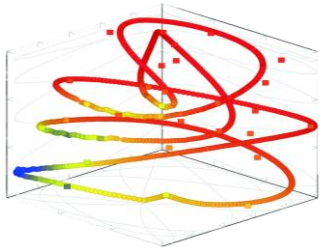
WHAT IS INFINITE-DIMENSIONAL OPTIMIZATION?

- Variables and/or constraints **indexed over continuous domains**

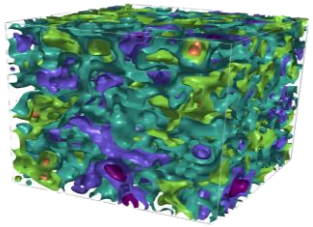




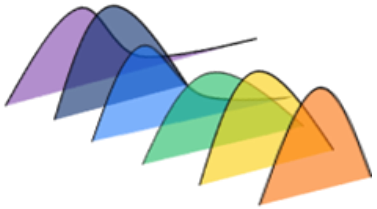
- Three common domains



Time



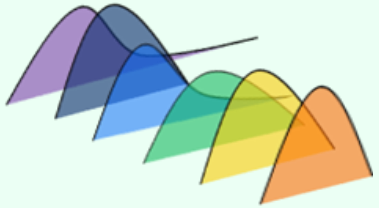
Space



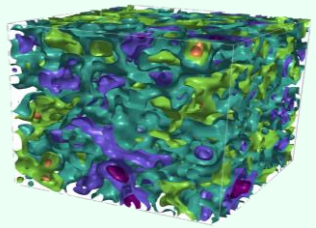
Uncertainty

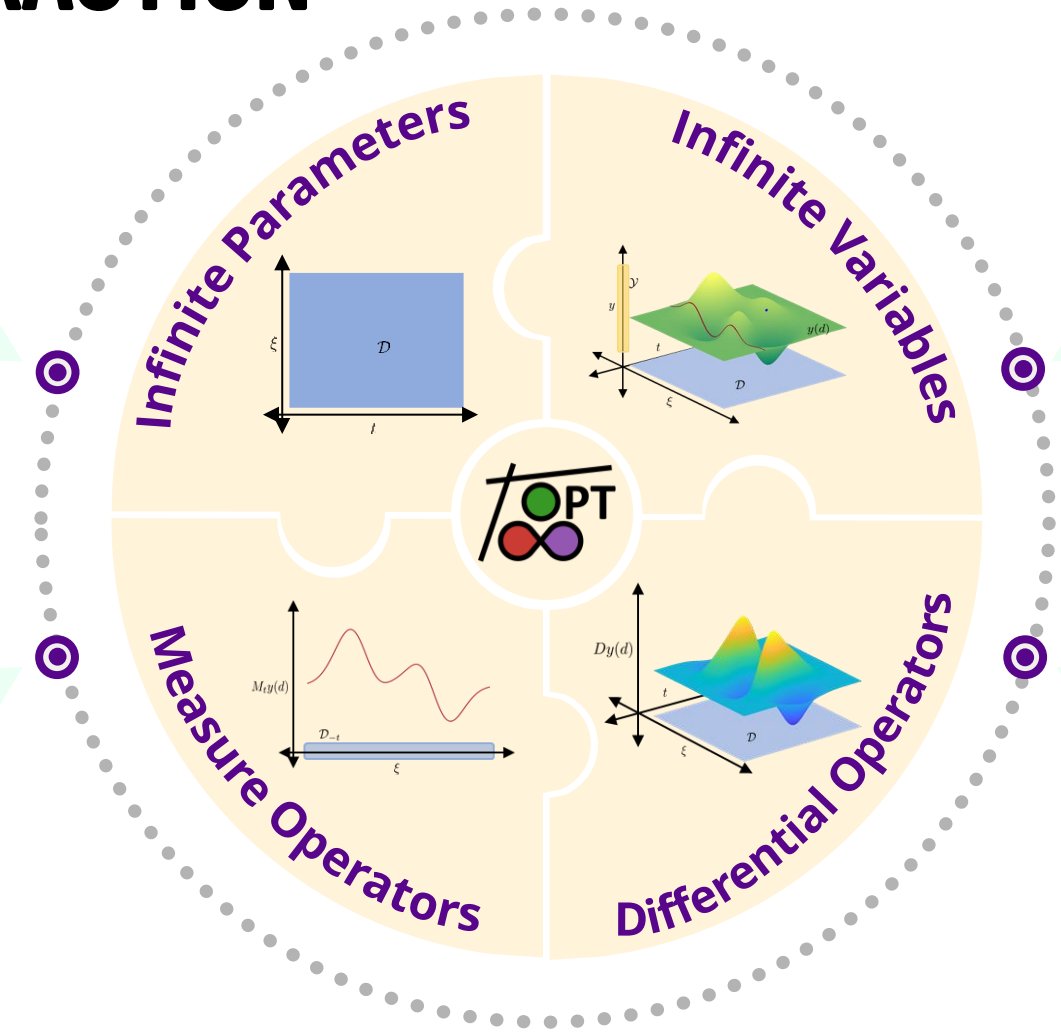
UNIFYING ABSTRACTION

Stochastic Optimization

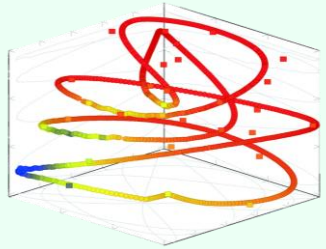


Space-Time Optimization

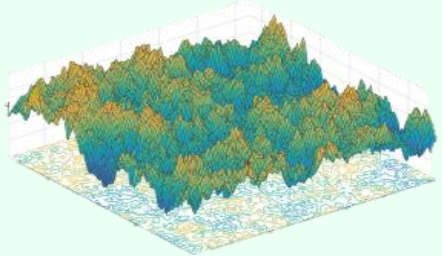




Dynamic Optimization



Combinations



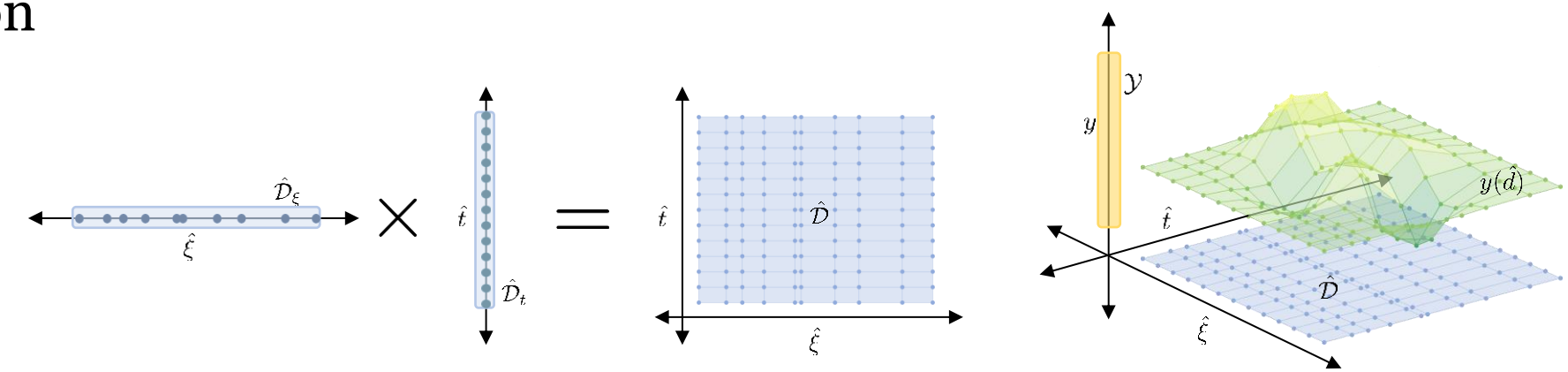
J. L. Pulsipher, W. Zhang, T. J. Hongisto, and V. M. Zavala. "A unifying modeling abstraction for infinite-dimensional optimization." 2022

SOLUTION VIA TRANSFORMATION

- Direct Transcription

Project onto set of **finite points** $\hat{\mathcal{D}}$

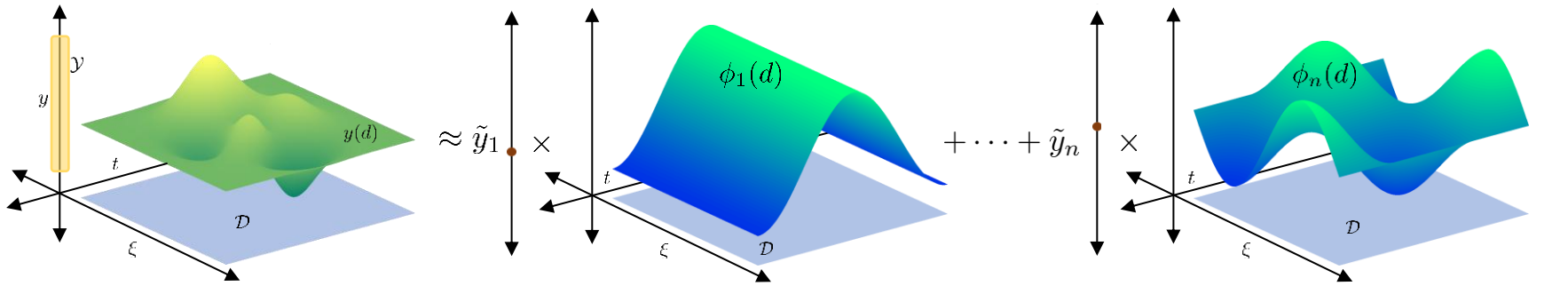
$$\hat{\mathcal{D}} := \prod_{\ell \in \mathcal{L}} \{\hat{d}_{\ell,i} : \hat{d}_{\ell,i} \in \mathcal{D}_\ell, i \in \mathcal{I}_\ell\}$$



- Alternative Methods

Project onto set of known **basis functions**

$$y(d) \approx \sum_{i \in \mathcal{I}} \tilde{y}_i \phi_i(d)$$

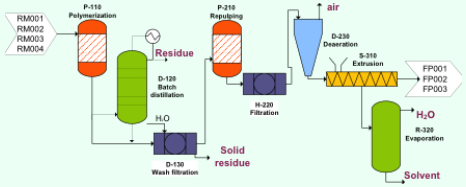


APPLICATIONS

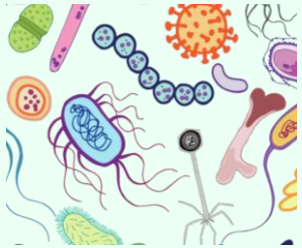
Unifying Abstraction

Dynamic Optimization
(Time)

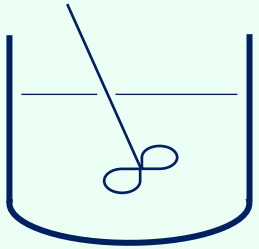
Autonomous Vehicles



Process Control

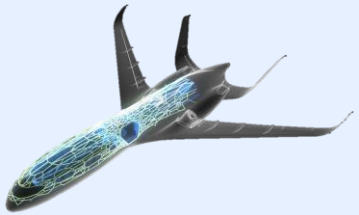


Bioprocessing

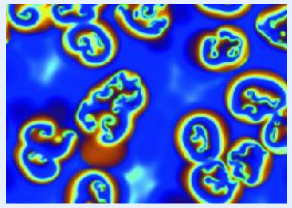


Reaction Systems

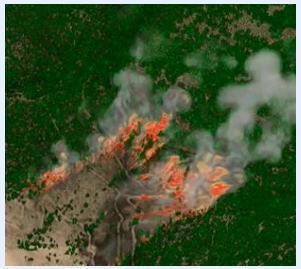
PDE-Constrained Optimization
(Space-Time)



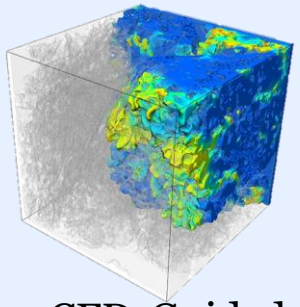
Structural Design



Diffusive Systems



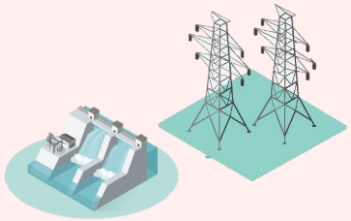
Wildfire Management



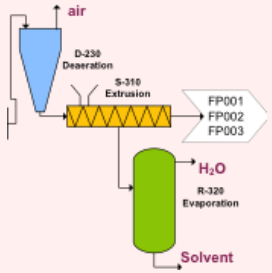
CFD-Guided Design

Stochastic Optimization
(Uncertainty)

Pharmaceutical Production



Optimal Power Flow



Process Design

Investment Planning

InfiniteOpt

Unifying Abstraction

- Captures **wide envelope** of problems
- Automates **transformations**
- Inspires **new modeling** approaches

Compact and Performant Modeling

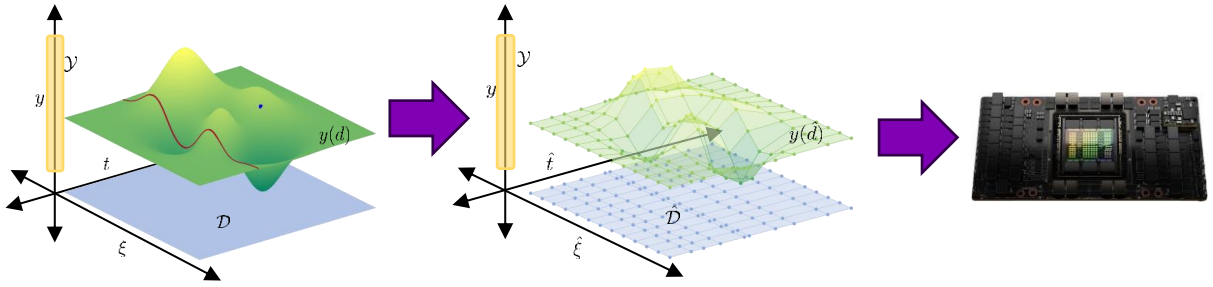
$$\frac{\partial y_b(t,\xi)}{\partial t} = 2y_b(t,\xi)^2 + y_a(t) - z_1$$

$$\mathbb{E}_\xi\left[y_c(t,\xi)\right] \geq \alpha$$

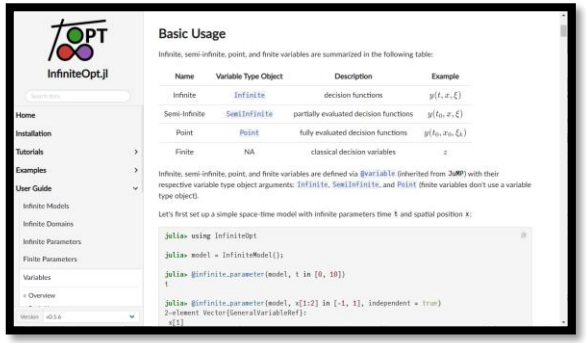
$$y_a(0) + z_2 = \beta$$

```
@constraint(m, ∂(yb, t) == 2yb^2 + ya - z[1])
@constraint(m, 𝔼(yc, ξ) ≥ α)
@constraint(m, ya(0) + z[2] == β)
```

Accelerated Solutions



Extensive Documentation



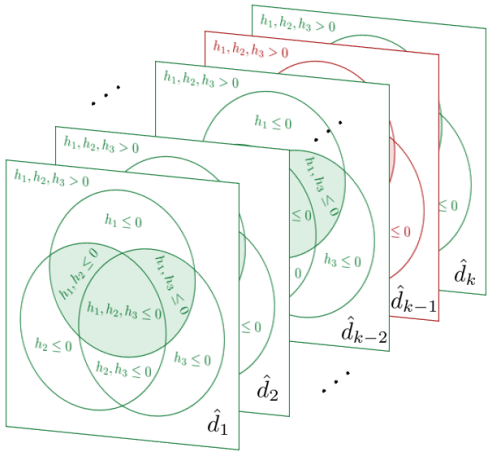
UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

WHAT ABOUT OTHER TOOLS?

Tool	ODEs	PDEs	Stochastic	Free	Extensible	ML Models	GPU	Measures
InfiniteOpt	✓	✓	✓	✓	✓	✓	✓	✓
PyomoDAE	✓	✓	✗	✓	✓	~	✗	~
CasADi	✓	✗	✗	✓	✗	✗	✗	✗
GEKKO	✓	~	✗	✓	✗	✓	✗	✗
et	✓	✗	✗	✓	✓	✗	✓	~
gPROMS	✓	✗	✗	✗	✗	✗	✗	~
	✓	✓	✗	✓	✓	~	~	~

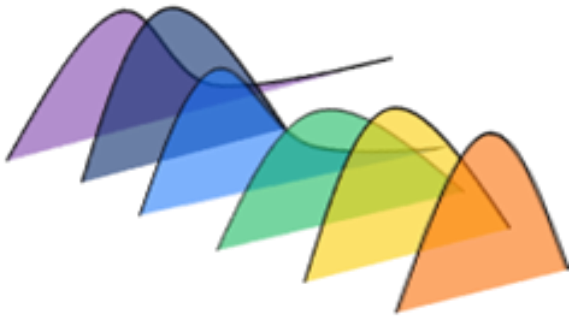
INSPIRING NEW MODELING OBJECTS

Event Constraints



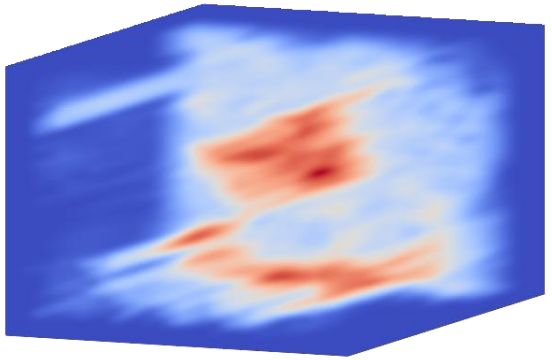
$$\mathbb{P}_d(g(y(d), z, d) \leq 0) \geq \alpha$$

Risk-Inspired Measures



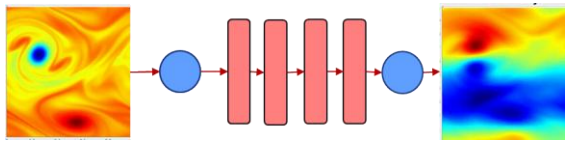
$$R(f(y(d), d))$$

Random Field Optimization



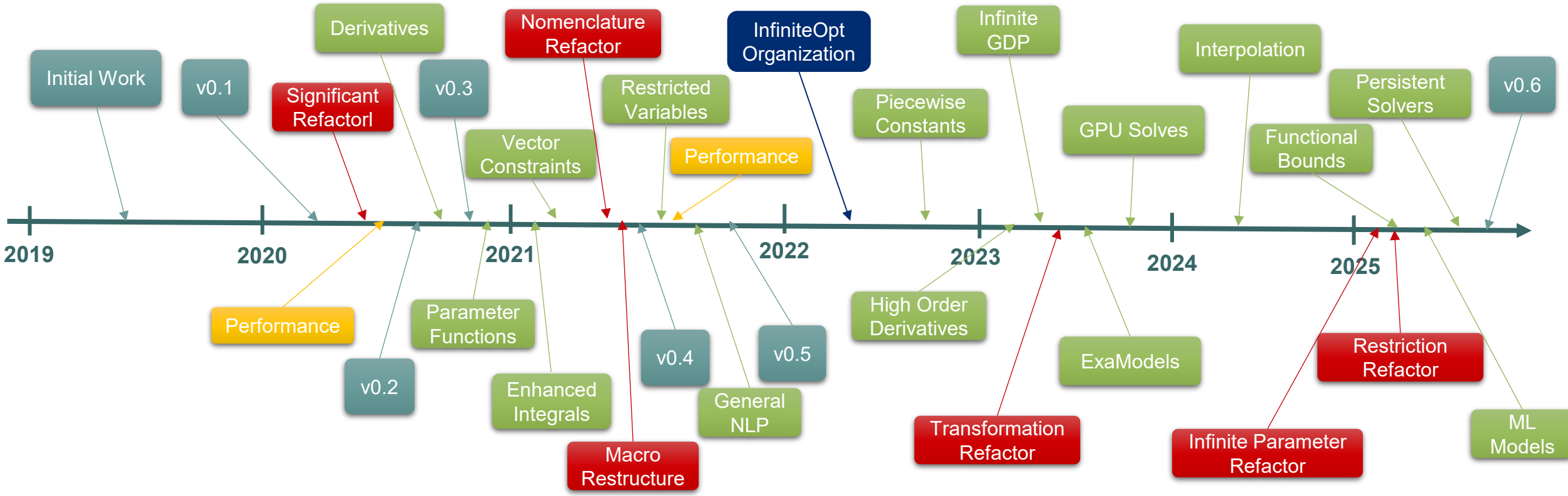
$$y(t, x, \xi(t, x))$$

Infinite-Dimensional ML Surrogates



$$NO(u(d)) = x(d)$$

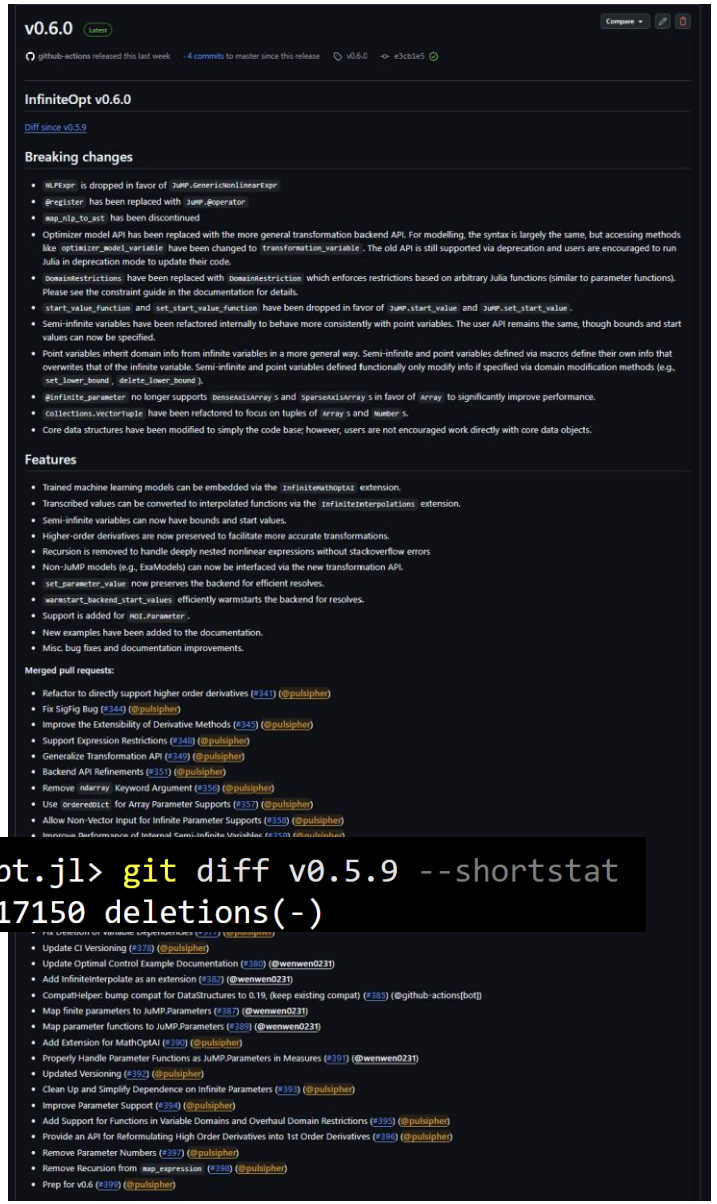
DEVELOPMENT HISTORY



WHAT'S NEW WITH VERSION 0.6?

- Extensive features/changes over past 3 years
  - User interface is largely unchanged
  - New nonlinear expressions
  - New **transformation** API
  - Refactor reduced variables
  - Functional bounds
  - **GPU accelerated solution**
  - **ML model** embedding
  - Infinite GDPs
  - Much much more

```
PS C:\Users\Pulsipher\Documents\InfiniteOpt.jl> git diff v0.5.9 --shortstat
 137 files changed, 15594 insertions(+), 17150 deletions(-)
```



v0.6.0 Latest

Compare

github-actions released this last week · 4 commits to master since this release · v0.6.0 · e3cb1e5

InfiniteOpt v0.6.0

Diff since v0.5.9

Breaking changes

- `NLPExpr` is dropped in favor of `JuMP.GenericNonlinearExpr`
- `@register` has been replaced with `JuMP.@operator`
- `map_nlp_to_ast` has been discontinued
- Optimizer model API has been replaced with the more general transformation backend API. For modelling, the syntax is largely the same, but accessing methods like `optimizer_model_variable` have been changed to `transformation_variable`. The old API is still supported via deprecation and users are encouraged to run Julia in deprecation mode to update their code.
- `DomainRestrictions` have been replaced with `DomainRestriction` which enforces restrictions based on arbitrary Julia functions (similar to parameter functions). Please see the constraint guide in the documentation for details.
- `start_value_function` and `set_start_value_function` have been dropped in favor of `JuMP.start_value` and `JuMP.set_start_value`.
- Semi-infinite variables have been refactored internally to behave more consistently with point variables. The user API remains the same, though bounds and start values can now be specified.
- Point variables inherit domain info from infinite variables in a more general way. Semi-infinite and point variables defined via macros define their own info that overwrites that of the infinite variable. Semi-infinite and point variables defined functionally only modify info if specified via domain modification methods (e.g., `set_lower_bound`, `delete_lower_bound`).
- `@infinite_parameter` no longer supports `DenseAxisArray`s and `SparseAxisArray`s in favor of `Array` to significantly improve performance.
- `Collections.VectorTuple` have been refactored to focus on tuples of `Array`s and `Number`s.
- Core data structures have been modified to simply the code base; however, users are not encouraged work directly with core data objects.

Features

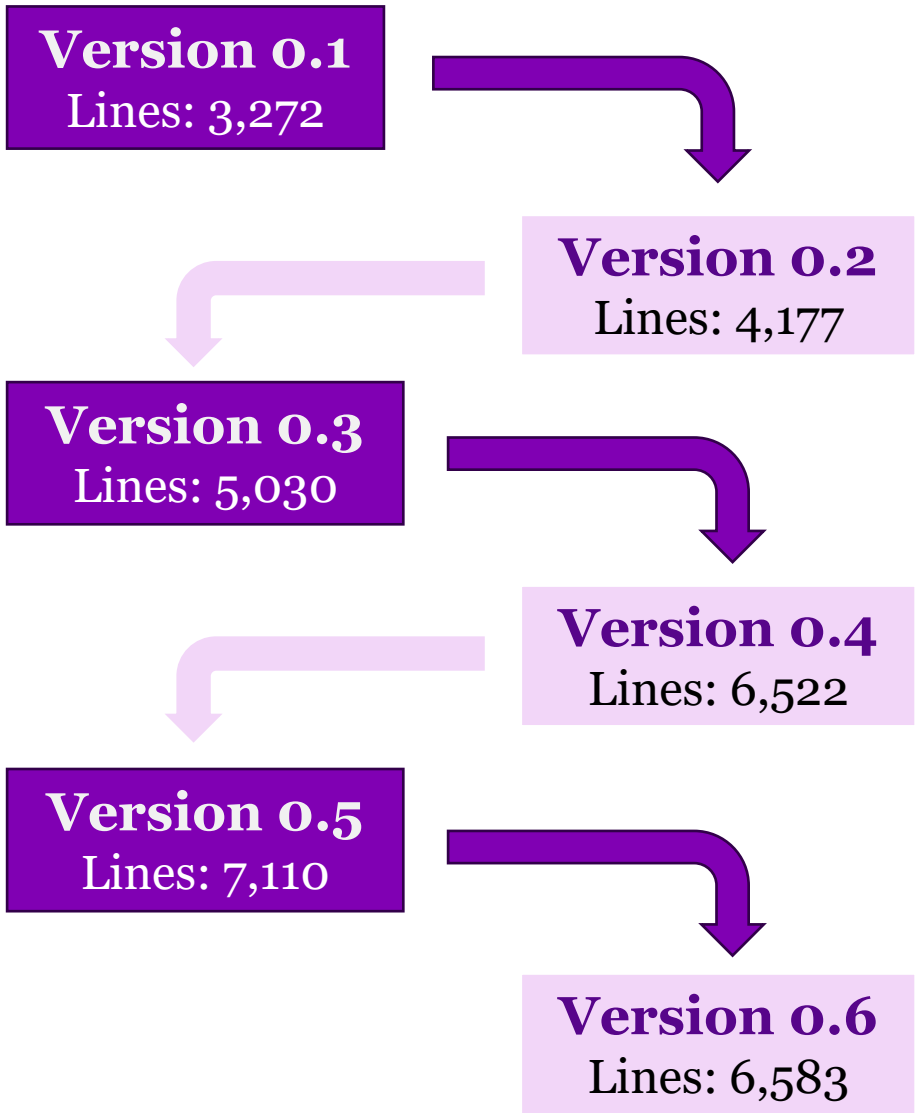
- Trained machine learning models can be embedded via the `InfiniteMathOptAI` extension.
- Transcribed values can be converted to interpolated functions via the `InfiniteInterpolations` extension.
- Semi-infinite variables can now have bounds and start values.
- Higher-order derivatives are now preserved to facilitate more accurate transformations.
- Recursion is removed to handle deeply nested nonlinear expressions without stackoverflow errors
- Non-JuMP models (e.g., ExaModels) can now be interfaced via the new transformation API.
- `set_parameter_value` now preserves the backend for efficient resolves.
- `warmstart_backend_start_values` efficiently warmstarts the backend for resolves.
- Support is added for `MOI.Parameter`.
- New examples have been added to the documentation.
- Misc. bug fixes and documentation improvements.

Merged pull requests:

- Refactor to directly support higher order derivatives (#341) (@pulsipher)
- Fix SigFig Bug (#344) (@pulsipher)
- Improve the Extensibility of Derivative Methods (#345) (@pulsipher)
- Support Expression Restrictions (#348) (@pulsipher)
- Generalize Transformation API (#349) (@pulsipher)
- Backend API Refinements (#351) (@pulsipher)
- Remove `ndarray` Keyword Argument (#356) (@pulsipher)
- Use `OrderedDict` for Array Parameter Supports (#357) (@pulsipher)
- Allow Non-Vector Input for Infinite Parameter Supports (#358) (@pulsipher)
- Update CI Versioning (#378) (@pulsipher)
- Update Optimal Control Example Documentation (#380) (@wenwen0231)
- Add InfiniteInterpolate as an extension (#382) (@wenwen0231)
- CompatHelper: bump compat for DataStructures to 0.19, (keep existing compat) (#385) (@github-actions[bot])
- Map finite parameters to JuMP.Parameters (#387) (@wenwen0231)
- Map parameter functions to JuMP.Parameters (#389) (@wenwen0231)
- Add Extension for MathOptAI (#390) (@pulsipher)
- Properly Handle Parameter Functions as JuMP.Parameters in Measures (#391) (@wenwen0231)
- Updated Versioning (#392) (@pulsipher)
- Clean Up and Simplify Dependence on Infinite Parameters (#393) (@pulsipher)
- Improve Parameter Support (#394) (@pulsipher)
- Add Support for Functions in Variable Domains and Overhaul Domain Restrictions (#395) (@pulsipher)
- Provide an API for Reformulating High Order Derivatives into 1st Order Derivatives (#396) (@pulsipher)
- Remove Parameter Numbers (#397) (@pulsipher)
- Remove Recursion from `map_expression` (#398) (@pulsipher)
- Prep for v0.6 (#399) (@pulsipher)

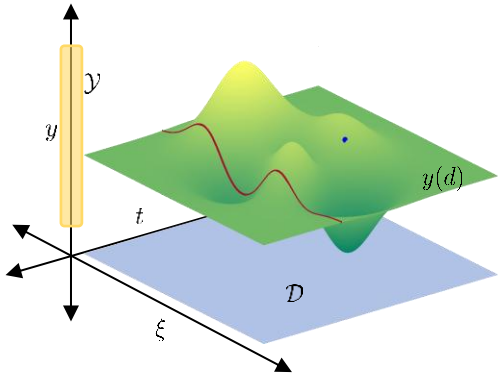
SIMPLIFIED CODE BASE

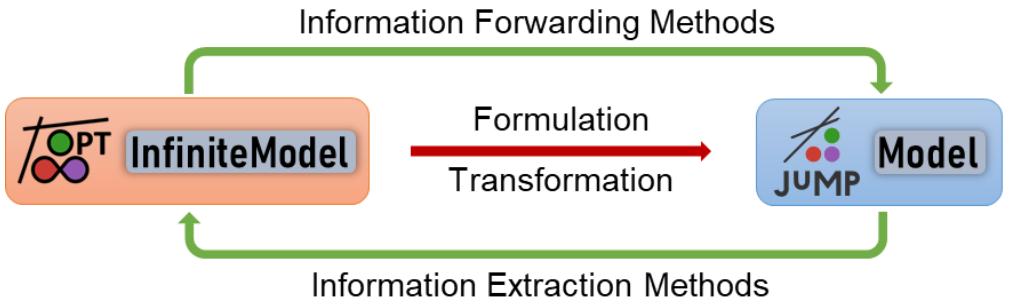
- Features and performance refactors steadily increased tracked source code lines
- **Reduced code** and increased performance
- Removed unnecessary complexity

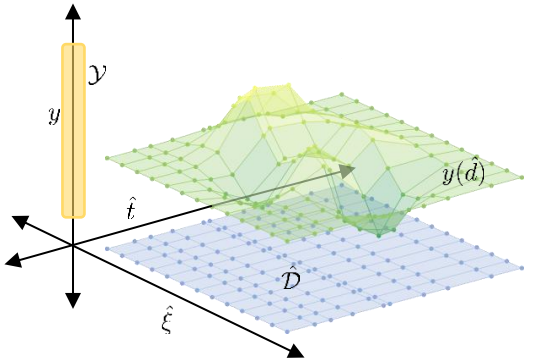


OPTIMIZER MODELS → TRANSFORMATION BACKENDS

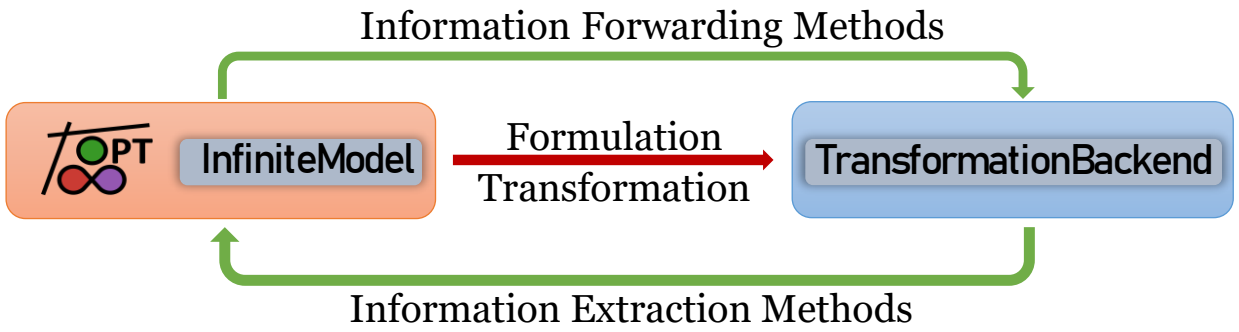
- *Before v0.6:* Optimizer model API







- *v0.6:* Transformation backend API

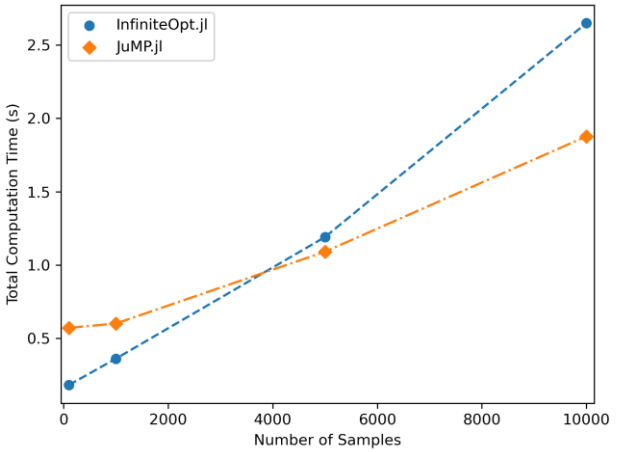


JuMP

ExaModels

IMPROVED TRANSCRIPTION

- TranscriptionOpt provides large library of discretization methods
  - E.g., finite difference, orthogonal collocation, quadrature, trapezoid, more
  - v0.6 expands library to better support higher order derivatives
- All outputs now are given as N-dimensional arrays
  - Where N is the # of infinite parameters



InfiniteOpt.jl
↓
Automatic Transcription

JuMP.jl
↓
Manual Transcription

INFINITEINTERPOLATIONS.JL

- Use interpolation to return continuous function solutions

- Extension of Interpolations.jl

- Works on variables, constraints, and expressions

```julia
using InfiniteOpt, Interpolations

# DEFINE AND SOLVE INFINITEMODEL HERE

discrete_y = value(y)
continuous_y = value(y, Constant()) # piecewise constant
continuous_y = value(y, Linear()) # linear spline
continuous_y = value(y, Cubic()) # cubic spline
```

FUNCTIONAL BOUNDS AND DOMAIN RESTRICTIONS

- Infinite variables support **function** bounds
- Semi-infinite variables now support bounds/starts
- Domain restrictions now take **arbitrary function**
  - Previously limited to subintervals
  - This is quite useful for PDEs

```
1   using InfiniteOpt
2   model = InfiniteModel()
3   @infinite_parameter(model, t ∈ [0, 10])
4   @infinite_parameter(model, x ∈ [-1, 1])
5
6   upper(t, x) = sin(t) * x
7   @variable(model, 0 ≤ y ≤ upper, Infinite(t, x))
8   set_upper_bound(y(0, x), 42)
9
10  restrict(t) = !iszero(t)
11  restriction = DomainRestriction(restrict, t)
12  @constraint(model, 3y^2 + 2y ≥ 2, restriction)
```

IMPROVED PARAMETER SUPPORT

- Previously parameter updates required rebuilding the backend
- New **persistent API** added to transformation backends
- Can update finite parameter and parameter functions (new)

```
using InfiniteOpt, Ipopt
model = InfiniteModel(
    Ipopt.Optimizer,
    update_parameter_functions = true
)
@infinite_parameter(model, t ∈ [0, 10])

@finite_parameter(model, p == 42)
@parameter_function(model, setpoint == t -> t > 5 ? 20 : 10)

# DEFINE THE MODEL

optimize!(model)

set_parameter_value(p, 10)
set_parameter_value(setpoint, t -> t > 3 ? 20 : 10)
```

WARMSTARTING

- Update start values via set_start_values
  - Uses last solution
  - **Requires Interpolations.jl** be imported
  - Backend will be rebuilt
- Update start values via warmstart_backend_start_values
  - Supports **persistent backend**
  - Updates all possible variables
  - The start values in the InfiniteModel are not updated

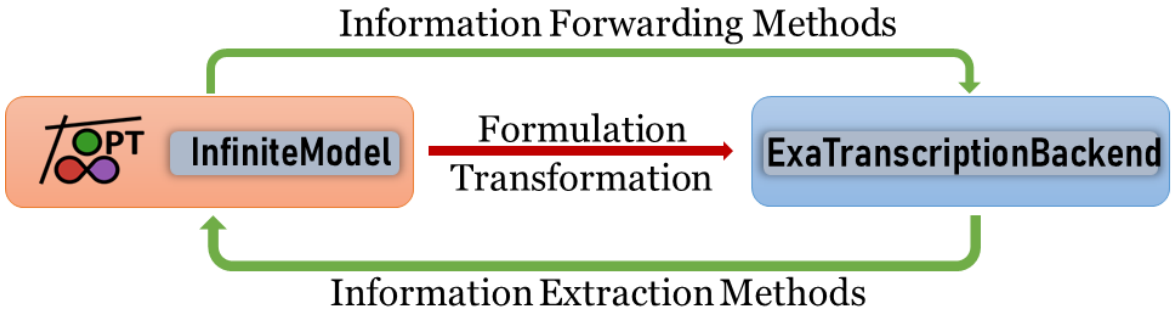
```
1   using InfiniteOpt, Interpolations, Ipopt
2   model = InfiniteModel(Ipopt.Optimizer)
3
4   # DEFINE THE MODEL
5
6   optimize!(model) # initial solve
7
8   set_start_values(model)
9
10  optimize!(model) # backend is rebuilt
11
12  warmstart_backend_start_values(model)
13
14  optimize!(model) # backend is not rebuilt
```

INFINITEEXAMODELS.JL: ACCELERATING SOLUTION

- Bridges the gap between InfiniteOpt & ExaModels
- **Automates transcription** through intuitive interface
- Leverages repeated structure to reduce **model creation time**
- Supports CPU (via Ipopt) and **GPU** (via MadNLP)

v0.1 is now released!

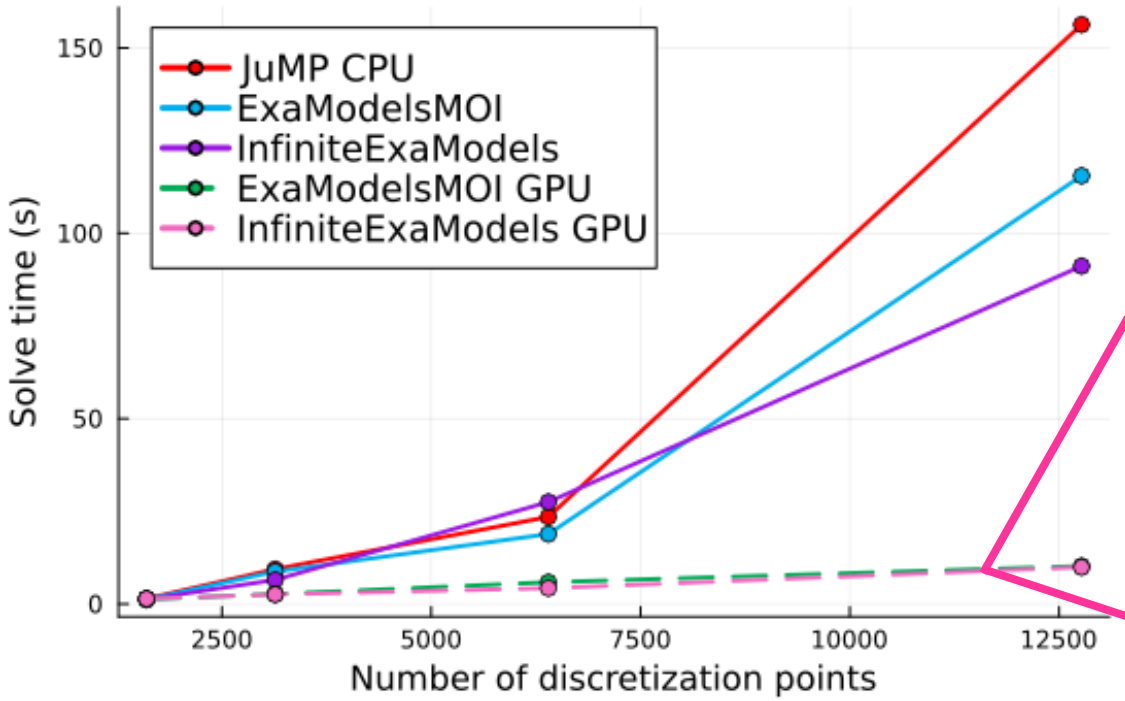
```
using InfiniteOpt, InfiniteExaModels, MadNLPGPU, CUDA
transform_backend = ExaTranscriptionBackend(MadNLPSolver, backend = CUDABackend())
model = InfiniteModel(transform_backend)
```

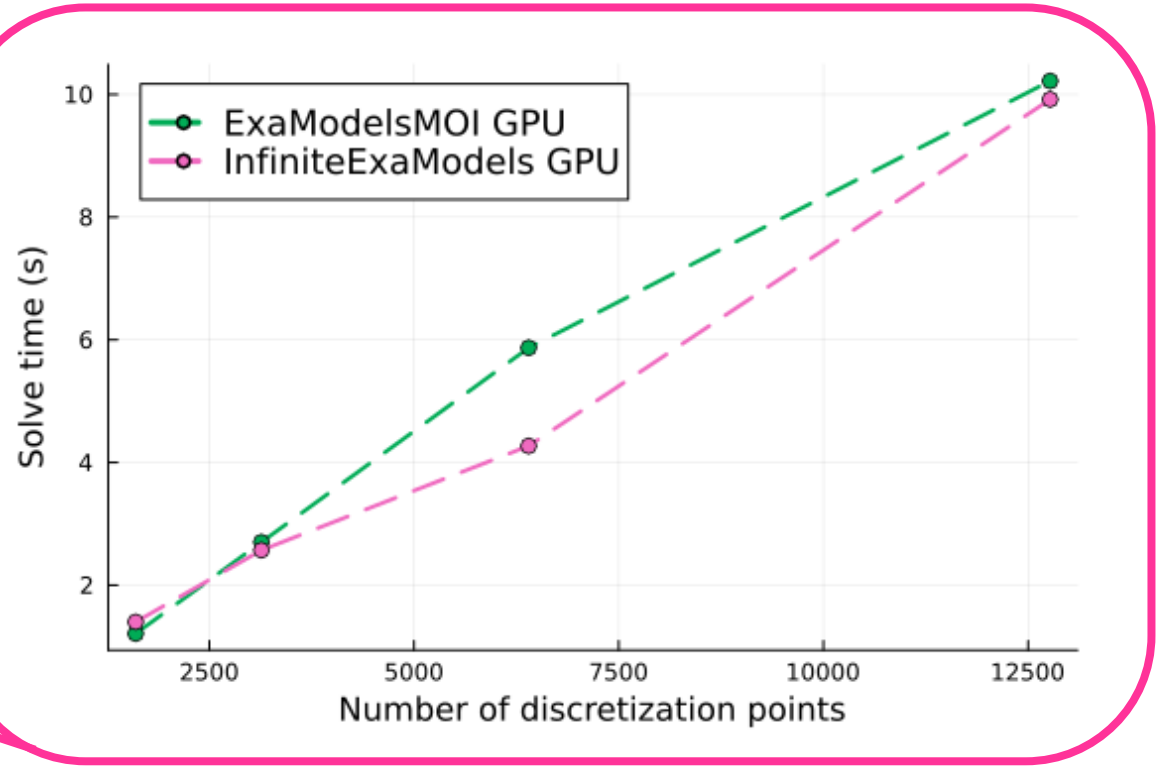


INFINITEEXAMODELS.JL: OPTIMAL CONTROL

Paper

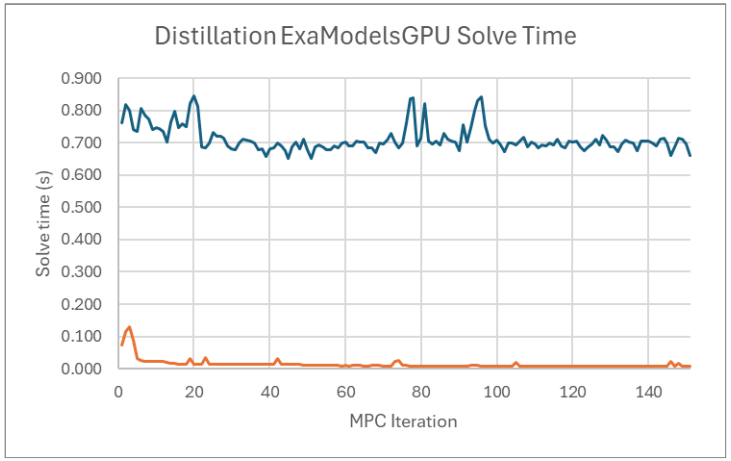
- GPU performs & scales better than CPU workflows
  - Significant time spent for **solver initialization**





RAPID NONLINEAR MODEL PREDICTIVE CONTROL

- New persistent backend + InfiniteExaModels.jl + GPU → **fastest known NMPC**



0.71 s per iteration

0.01 s per iteration

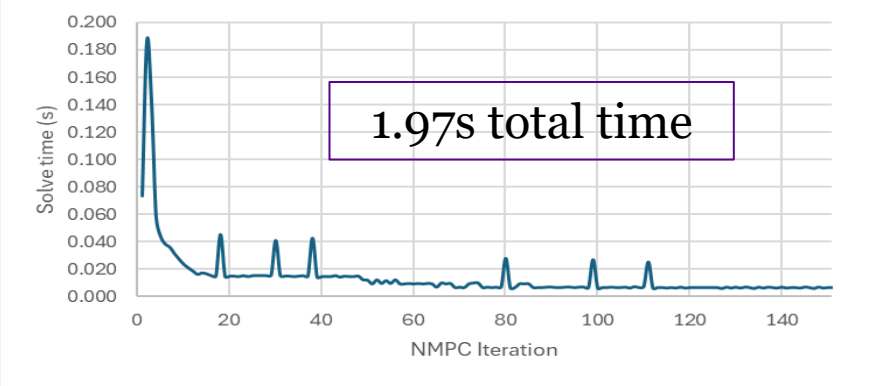
```julia
using InfiniteOpt, Ipopt
model = InfiniteModel(Ipopt.Optimizer)
# DEFINE THE MODEL

for i in 1:N
    optimize!(model)
    warmstart_backend_start_values(model)
    set_parameter_value(x0, value(x(0)))
    set_parameter_value(setpoint, set[i])
end
```

	JuMP		ExaModelsCPU		ExaModelsGPU	
Case Study	**Base**	**WS + PU**	**Base**	**WS + PU**	**Base**	**WS + PU**
Distillation	137.53	39.09	61.45	17.08	30.94	1.62
PDE Heated Plate	143.95	135.19	136.35	80.31	21.53	1.42

OPTIMALCONTROL.JL VS. INFINITEOPT.JL FOR NMPC

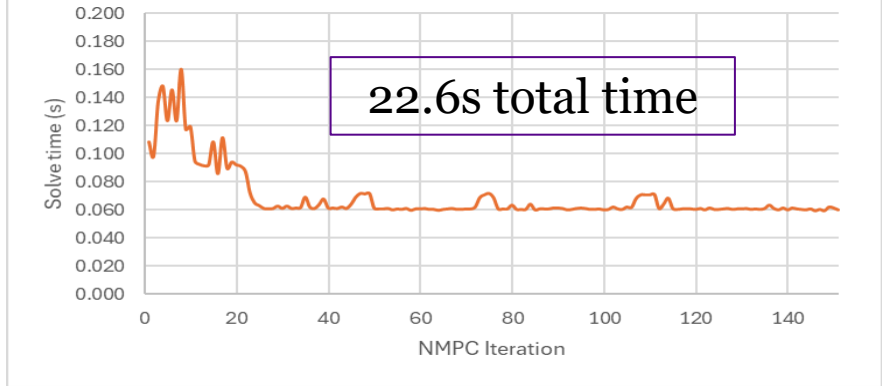
InfiniteOpt.jl

- Interfaces with ExaModels → GPU NMPC
- Supports ODEs and **PDEs**
- Supports **orthogonal collocation** and finite difference on GPU
- Support **set notation** (benchmark 22 lines)
- Supports **persistent solves**



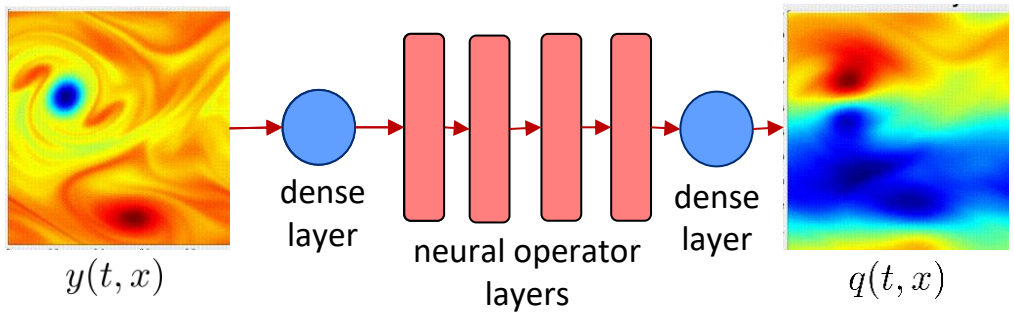
OptimalControl.jl

- Interfaces with ExaModels → GPU NMPC
- Supports ODEs
- Supports **only finite difference** on GPU
- Doesn't support set notation (benchmark **63 lines**)
- **Rebuilds solver** instance



INFINITEMATHOPTAI.JL: EMBEDDING ML MODELS

- Acts as a bridge between InfiniteOpt & MathOptAI.jl
- **Embed ML models** directly in InfiniteOpt problems
- Enables use of **neural ODEs**
- Opens possibility for infinite ML models
  - Gaussian processes
  - Neural operators




```julia
using InfiniteOpt, Flux, MathOptAI
NN = predictor(
    Flux.Chain(
        Flux.Dense(2, 16, Flux.relu),
        Flux.Dense(16, 16, Flux.relu),
        Flux.Dense(16, 2)
    )
)
model = InfiniteModel()
@infinite_parameter(model, t ∈ [0, 10])
@variable(model, x[1:2], Infinite(t))
y, formulation = add_predictor(model, NN, x)
@constraint(model, [i in 1:2], ∂(x[i], t) == y[i])
```

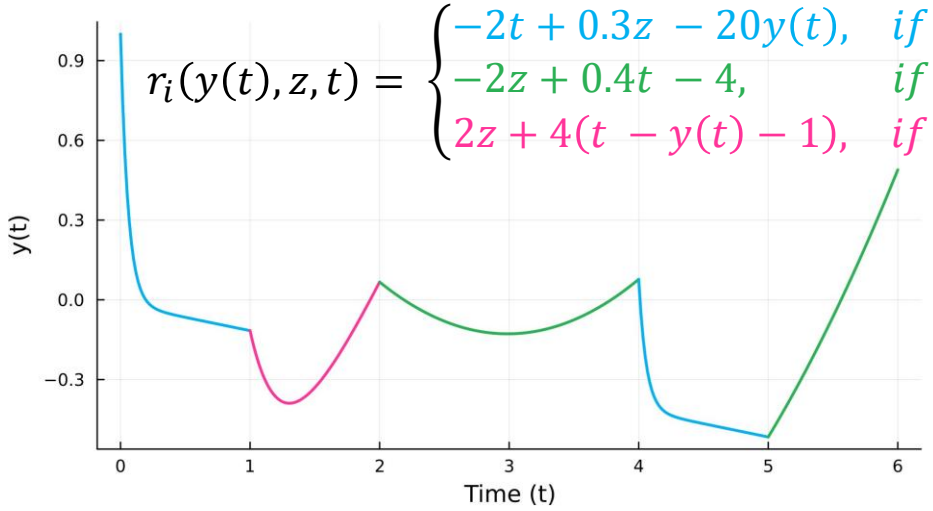
INFINITEDISJUNCTIVEPROGRAMMING.JL: MODELING LOGIC

- Enables infinite-dimensional GDP
- Model ODE/constraint switching

Paper

InfiniteOpt + JuMP

$$r_i(y(t), z, t) = \begin{cases} -2t + 0.3z - 20y(t), & if\ i = 1 \\ -2z + 0.4t - 4, & if\ i = 2 \\ 2z + 4(t - y(t) - 1), & if\ i = 3 \end{cases}$$




```julia
using DisjunctiveProgramming, InfiniteOpt, HiGH
model = InfiniteGDPModel(HiGHS.Optimizer)
I = 1:4; J = 1:2
@infinite_parameter(model, t ∈ [0, 1], num_supports = 100)
@variable(model, 0 ≤ y[I] ≤ 10, Infinite(t))

# Add the disjunctions and their indicator variables
@variable(model W[I, J], InfiniteLogical(t))
@constraint(model, [i ∈ I, j ∈ J], 0 ≤ y[i], Disjunct(W[i, 1]))
@constraint(model, [i ∈ I, j ∈ J], y[i] ≤ 0, Disjunct(W[i, 2]))
@disjunction(model, [i ∈ I], W[i, :])

# Add the logical propositions
@constraint(model, W[1, 1] ∨ W[2, 1] ∧ W[3, 1] := true)
optimize!(model, gdp_method = Hull())
```

MTKINFINITEOPTEXT.JL: INTEGRATING MTK MODELS

- Ports ModelingToolkit models to InfiniteOpt
- Enables structural simplifications
- More work is needed to increase performance and better use InfiniteOpt's extension API

```julia
using ModelingToolkit, InfiniteOpt, Ipopt

# Double integrator minimum time
t = M.t_nounits
D = M.D_nounits
@variables x(..) v(..)
@variables u(..) [bounds = (-1.0, 1.0), input = true]
@parameters tf
constr = [v(tf) ~ 0.0, x(tf) ~ 0]
cost = [tf] # Maximize the final distance.
@named block = ODESystem(
    [D(x(t)) ~ v(t), D(v(t)) ~ u(t)], t; costs = cost, constraints = constr)
block, input_idxs = structural_simplify(block, ([u(t)], []))

u0map = [x(t) => 1.0, v(t) => 0.0]
tspan = (0.0, tf)
parammap = [u(t) => 0.0, tf=>1.0]
jprob = JuMPDynamicOptProblem(block, u0map, tspan, parammap; steps = 51)
jsol = solve(jprob, Ipopt.Optimizer, :Verner8)
```

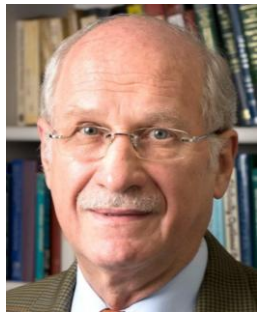
DEVELOPMENT ROADMAP (A PARTIAL SNAPSHOT)

- Add more features that JuMP supports
  - Multi-objective, generic precision, complex values, etc.
- Enhance PDE support (e.g., support finite elements)
- Expand envelope of persistent backend API
- Develop GPU-accelerated parameter estimation workflows
- Infinite ML models
  - Add support for larger collection of infinite-dimensional ML models
  - Use simultaneous method to train neural ODEs
- Infinite GDP: Finish release and develop tailored solver

ACKNOWLEDGEMENTS

Victor Zavala
UW-Madison
Professor

Carl Laird
CMU
Professor



Ignacio Grossmann
CMU
Professor

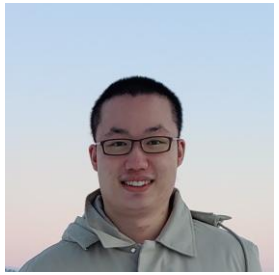
Hector Perez
RelationalAI
Researcher

Daniel Ovalle Varela
CMU
PhD Candidate

Stefan Mazzadi
UWaterloo
Incoming MASc

Sungho Shin
MIT
Asst. Professor

Evelyn Gondosiswanto
UWaterloo
MASc Student



Weiqi Zhang
Uber
Former PhD Student

Daniel Nguyen
UWaterloo
MASc Student

Oscar Dowson
JuMP-dev
Developer

UNIVERSITY OF WATERLOO
Department of Chemical Engineering

CAPD

DEPARTMENT OF ENERGY · UNITED STATES OF AMERICA

NSERC CRSNG

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

TRY IT OUT!

InfiniteExaModels.jl

InfiniteOpt

DisjunctiveProgramming.jl