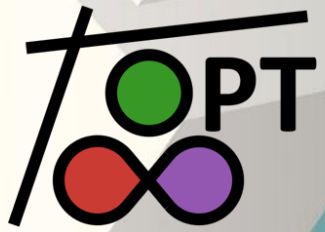
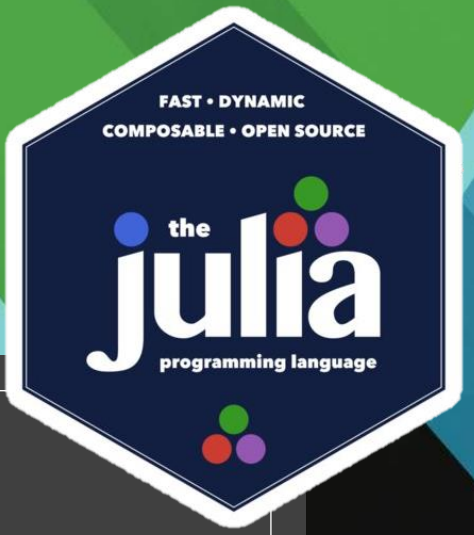


InfiniteOpt



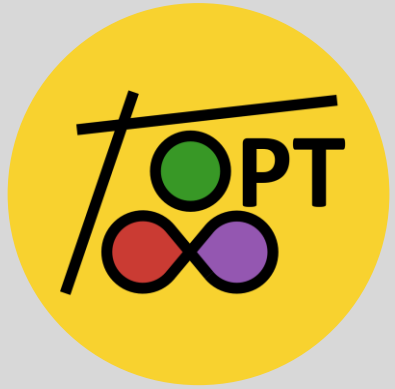
INFINITEOPT.JL: A JULIA PACKAGE FOR INFINITE-DIMENSIONAL OPTIMIZATION

Dr. Joshua Pulsipher

Today's Topics

CODING IN JULIA

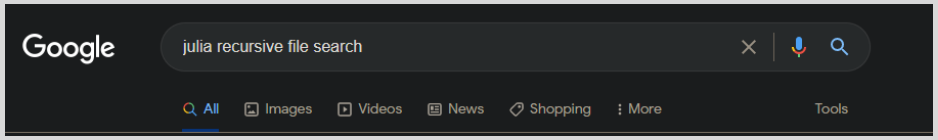
MATHEMATICAL OPTIMIZATION VIA JUMP.JL

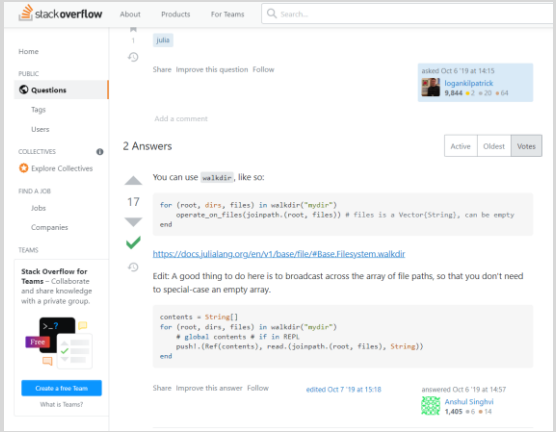


INFINITE-DIMENSIONAL OPTIMIZATION VIA INFINITEOPT.JL

Learning Outcomes

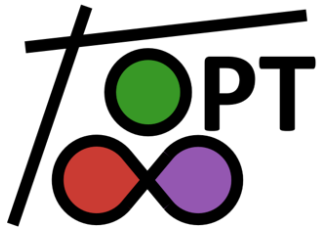
- Topic familiarity
  - Can briefly describe what Julia, JuMP.jl, and InfiniteOpt.jl are (and what they might be useful for)
  - Can "Google" the right things
- Programming expertise
  - Can setup basic Julia, JuMP.jl, and InfiniteOpt.jl workflows with appropriate reference materials
- Continued learning
  - Know what/where comes next in developing expertise with these tools
  - Identify resources to facilitate this learning





Course Schedule

Times	Topic	Duration
8:30 a.m. – 9:45 a.m.	*Introduction* - The why and what of Julia, JuMP.jl, and InfiniteOpt.jl.	45 min.
9:45 a.m. – 10:15 a.m.	*Installation and Setup* - Configure software on personal laptop. Online interface provided as an alternative.	30 min.
10:15 a.m. – 10:30 a.m.	Break	15 min.
10:30 a.m. – 11:30 a.m.	*Julia: A Practical Introduction* – Overview of core types, programmatic syntax, and package management.	60 min.
11:30 a.m. – 11:45 a.m.	Break	15 min.
11:45 a.m. – 12:30 p.m.	*JuMP.jl: A Brief Introduction* – The basics of modeling and solving mathematical optimization problems in JuMP.jl.	45 min.
12:30 p.m. – 1:30 p.m.	Lunch	60 min.
1:30 p.m. – 2:30 p.m.	*JuMP.jl: Beyond the Basics* – A deeper dive into the core modeling/solution strategies including variables, constraints, containers, and more.	60 min.
2:30 p.m. – 2:45 p.m.	Break	15 min.
2:45 p.m. – 3:45 p.m.	*InfiniteOpt.jl: The Basics* – An introduction on how to compactly model and solve complex infinite-dimensional optimization problems.	60 min.
3:45 p.m. – 4:00 p.m.	Break	15 min.
4:00 p.m. – 4:45 p.m.	*InfiniteOpt.jl: New Modeling Strategies* – A tutorial on how InfiniteOpt.jl enables new formulation/solution approaches.	45 min.
4:45 p.m. – 5:15 p.m.	*InfiniteOpt.jl: Deployment Tools* – An overview of the API to enable rapid deployment of new modeling/solution techniques.	30 min.
5:15 p.m. – 5:30 p.m.	*Final Thoughts* – Summary of key points and panned future development. Provide resources for further learning.	15 min.



InfiniteOpt

Introduction

Why Julia?

- Speed
  - Up to C speeds

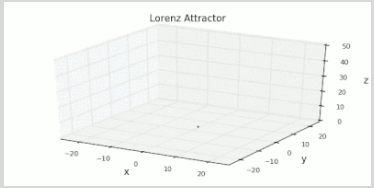
- Parallel Computing
  - Designed for it from the ground up

- Portability
  - Works on any O.S.

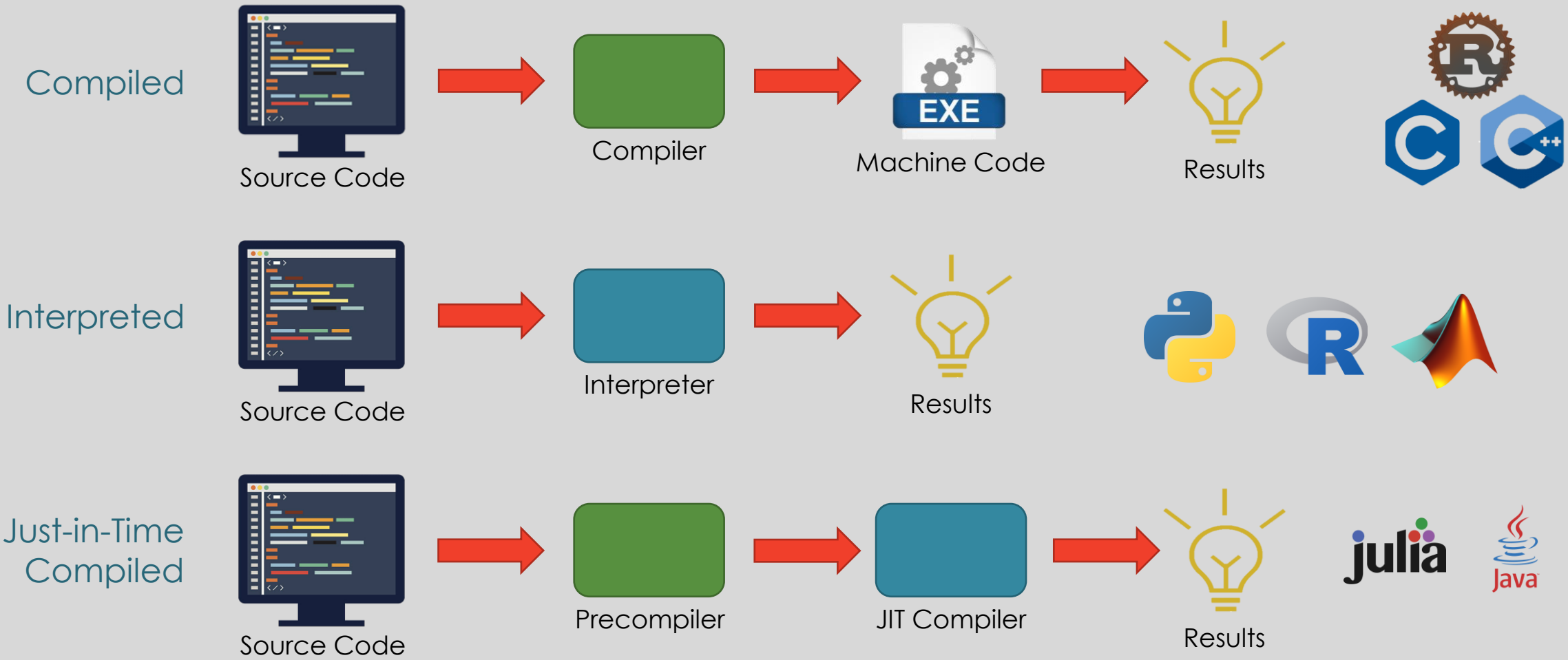
- Generality
  - Can be used for scripts, apps, metaprogramming, etc.

- Open Source
  - It's free!

- Scientific Computing
  - MATLAB-like linear algebra, symbolic math, scripting



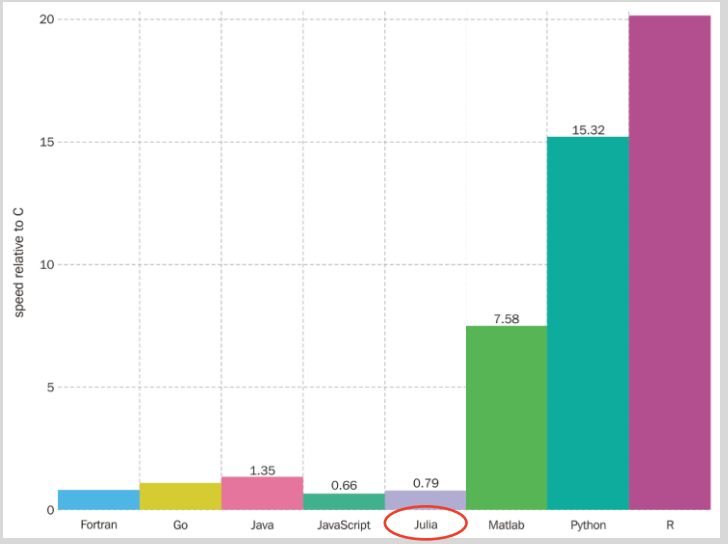
Programming Language Paradigms



*Over simplified

Performance

- Compiled Languages
  - Difficult "low-level" code
  - Cannot be modified while running
  - Not very portable between computers
  - Fast
- Interpreted Languages
  - Easy "high-level" syntax
  - Can be modified while running
  - Highly portable
  - Slow
- JIT Compiled Languages
  - Can be "high-level" (only Julia)
  - Highly portable
  - Can be modified while running
  - Fast*



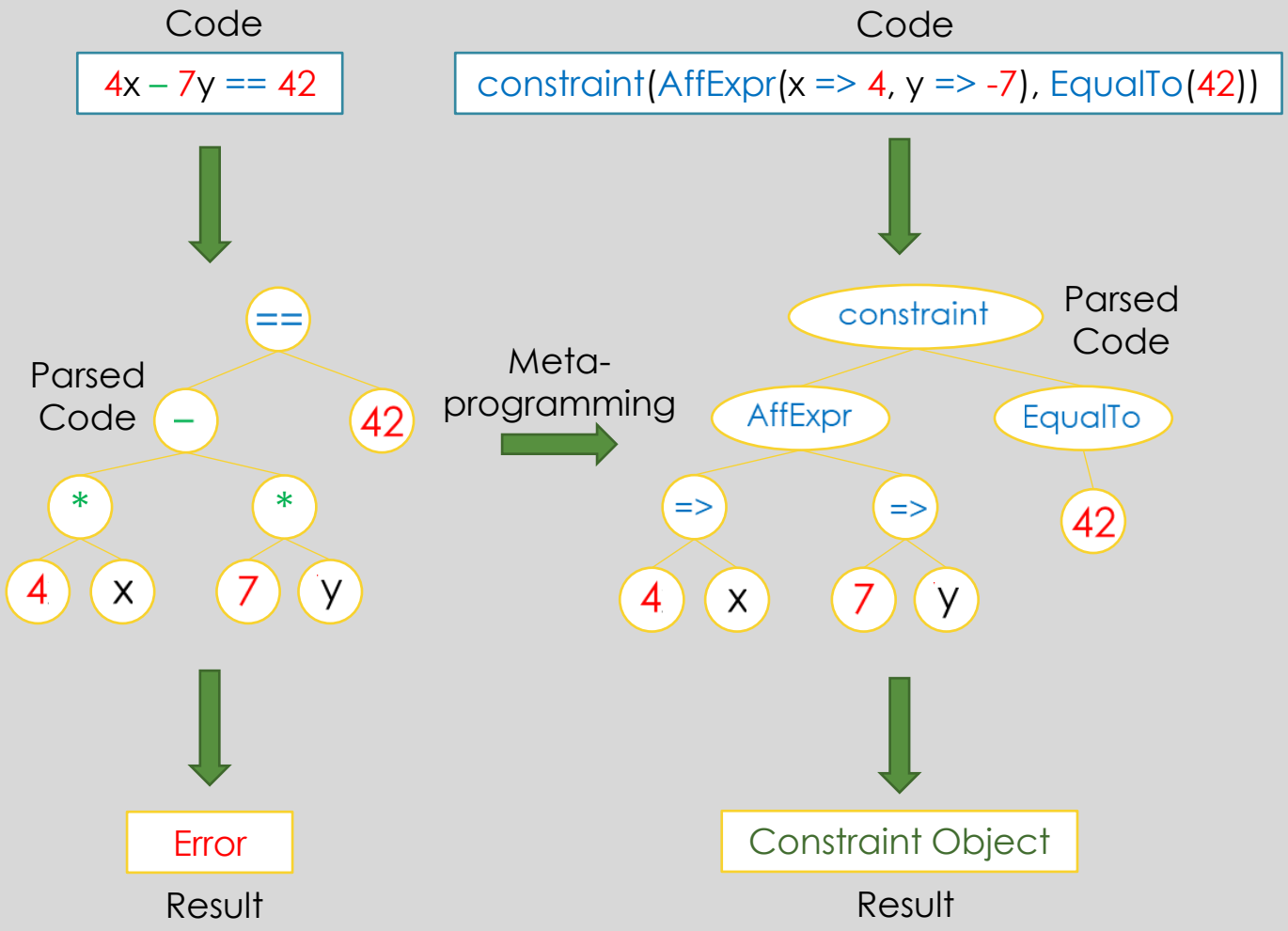
Metaprogramming (Macros)

- Intercept code before it is run
- Write Julia code that writes Julia code
- **Enables easy symbolic coding for users**

```
@constraint(4x – 7y == 42)
```

⬇

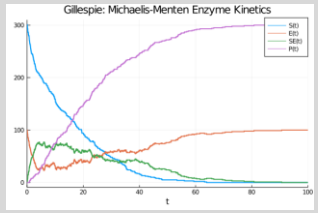
```
constraint(AffExpr(x => 4, y => -7), EqualTo(42))
```



Packages

- ModelingToolkit.jl
  - Symbolic math ~1000 times faster than SymPy

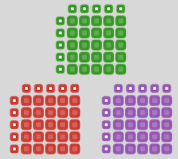
- DifferentialEquations.jl
  - Solve ODEs, PDEs, SDEs, DAEs, more
  - Fastest implementation available



- JuMP.jl
  - Symbolic interface for optimization

```
@constraint(model, 4f + 2s <= 4800)
@constraint(model, f + s <= 1750)
@constraint(model, 0 <= f <= 1000)
@constraint(model, 0 <= s <= 1500)
```

- DataFrames.jl
  - Efficient datasets (like Pandas)



- Flux.jl
  - Efficient/interpretable machine learning

- DistributedArrays.jl (GPU parallelzation)
  - Easy parallelization for CPU cores/threads and GPUs

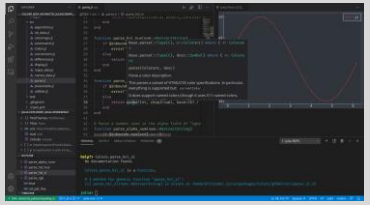
Why JuMP.jl?

- High-level Syntax
  - Novice friendly

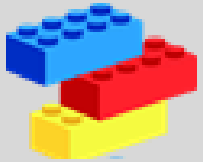
```
@constraint(model, 4f + 2s <= 4800)
@constraint(model, f + s <= 1750)
@constraint(model, 0 <= f <= 1000)
@constraint(model, 0 <= s <= 1500)
```

- Solver Library
  - Rapidly try different solvers

- Programmable
  - Easy embed in a data script



- Extensible
  - Can be extended for advanced techniques

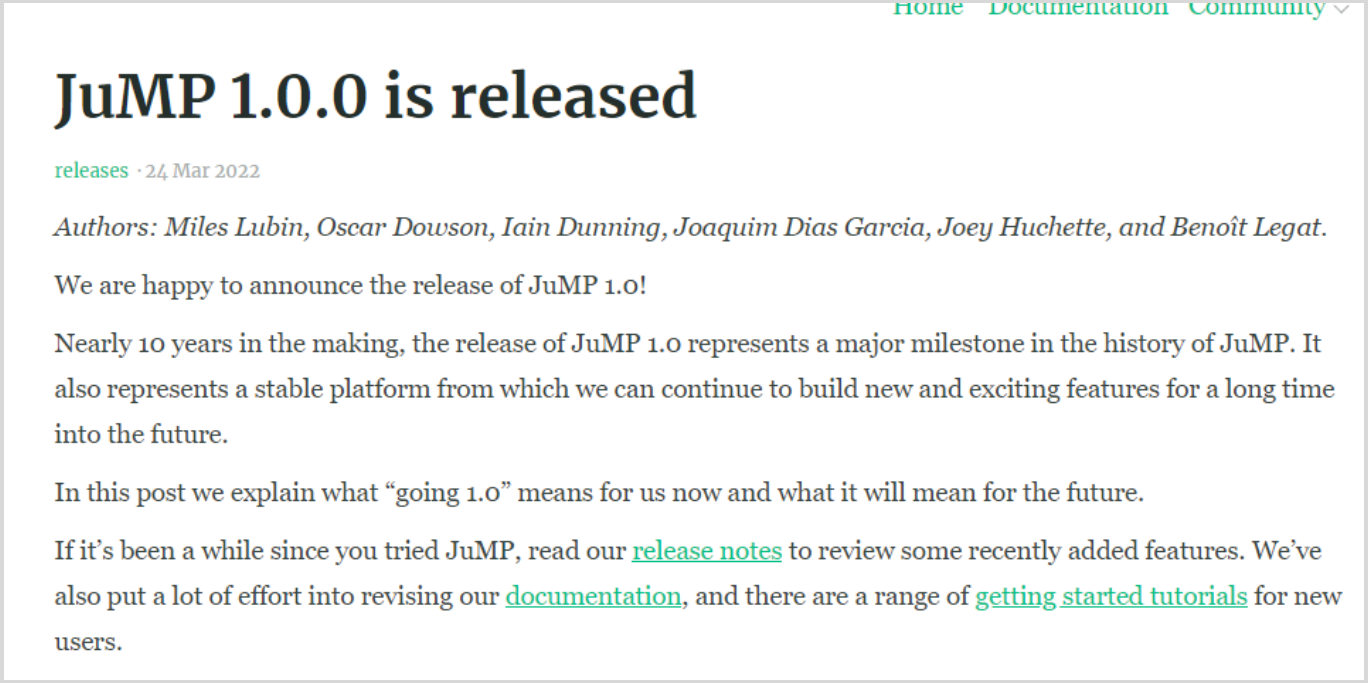


- Open Source
  - It's free!

- Performance
  - Highly performant (it's Julia)

JuMP.jl 1.0

- After nearly 10 years, JuMP.jl is now at version 1.0
- "By releasing JuMP 1.0.0, the core contributors are ensuring that all code you write using a 1.x.y release of JuMP will continue to work" until 2.0.0



Home Documentation Community

JuMP 1.0.0 is released

releases · 24 Mar 2022

Authors: Miles Lubin, Oscar Dowson, Iain Dunning, Joaquim Dias Garcia, Joey Huchette, and Benoît Legat.

We are happy to announce the release of JuMP 1.0!

Nearly 10 years in the making, the release of JuMP 1.0 represents a major milestone in the history of JuMP. It also represents a stable platform from which we can continue to build new and exciting features for a long time into the future.

In this post we explain what "going 1.0" means for us now and what it will mean for the future.

If it's been a while since you tried JuMP, read our release notes to review some recently added features. We've also put a lot of effort into revising our documentation, and there are a range of getting started tutorials for new users.

What is Finite Optimization?

Idea: Determine the "best" set of **decisions** for a given problem.

Aspects

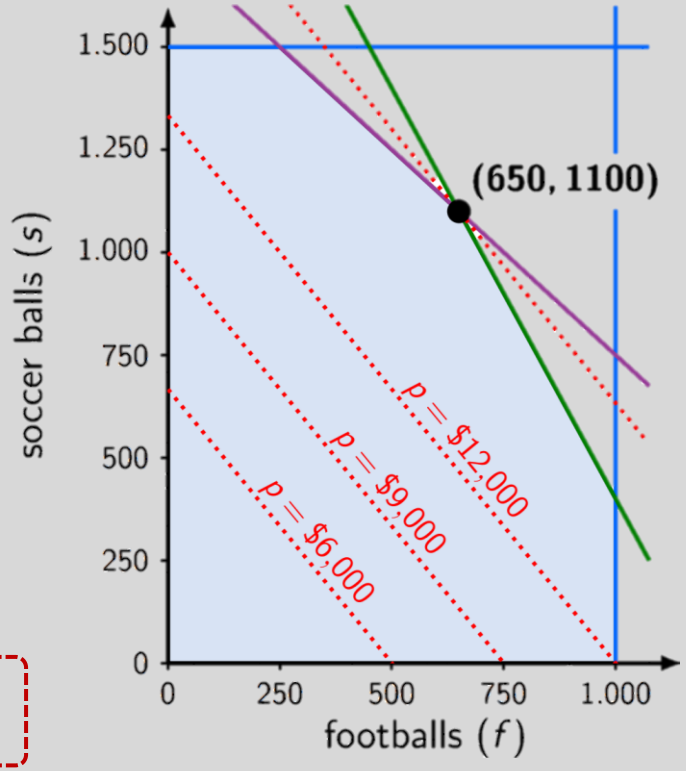
- Parameters
  - Fixed quantities/data
- Decision Variables
  - Scalar values to optimize
- Objective
  - Decision criteria
- Constraints
  - Modeling equations
  - Decision requirements

$$\max_{f,s} \; 12f + 9s$$

$$\text{s.t.} \quad 4f + 2s \leq 4800$$
$$f + s \leq 1750$$
$$0 \leq f \leq 1000$$
$$0 \leq s \leq 1500$$

Finite # of **decisions**

Sports Store Example



Modeling in JuMP.jl

$$\begin{aligned} \max_{f,s} \quad & 12f + 9s \\ \text{s.t.} \quad & 4f + 2s \leq 4800 \\ & f + s \leq 1750 \\ & 0 \leq f \leq 1000 \\ & 0 \leq s \leq 1500 \end{aligned}$$

- Initialize **model**

```
1  using JuMP, Gurobi
2  model = Model(Gurobi.Optimizer)
```

- Define **variables**

```
3  @variable(model, f)
4  @variable(model, s)
```

- Define **objective**

```
5  @objective(model, Max, 12f + 9s)
```

- Define **constraints**

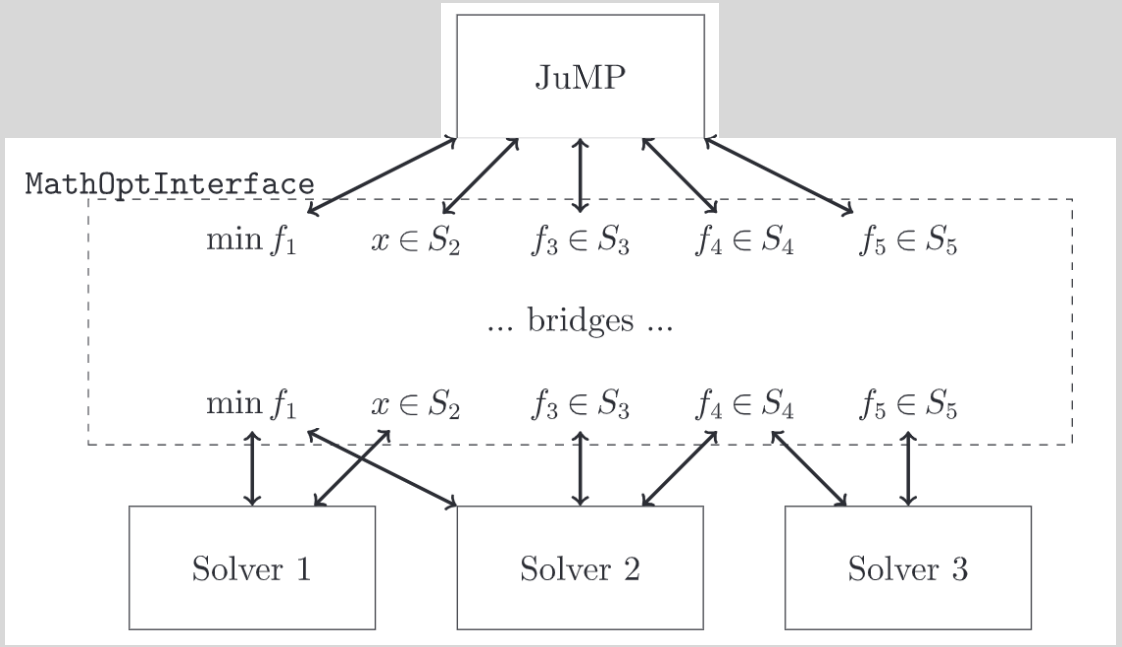
```
6  @constraint(model, 4f + 2s <= 4800)
7  @constraint(model, f + s <= 1750)
8  @constraint(model, 0 <= f <= 1000)
9  @constraint(model, 0 <= s <= 1500)
```

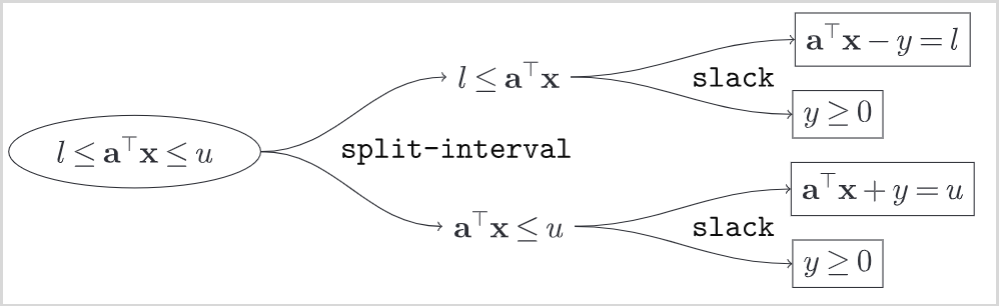
- **Solve**

```
10 optimize!(model)
11 profit = objective_value(model)
12 f_best = value(f)
13 s_best = value(s)
```

JuMP.jl Architecture

- JuMP.jl provides the high-level interface (the "macro sugar")
- Model is stored in MathOptInterface.jl model
  - Constraints are of the form *function* in *set*
- Bridges convert constraints into form that solvers support
  - Solvers don't explicitly support every *function-set* combination
- MathOptInferface.jl preserves the constraint mappings
- Solvers interface with MathOptInterface.jl

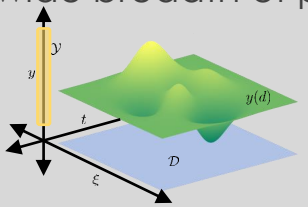




Legat, Benoît, et al. "MathOptInterface: a data structure for mathematical optimization problems." *INFORMS Journal on Computing* 34.2 (2022): 672-689.

Why InfiniteOpt.jl?

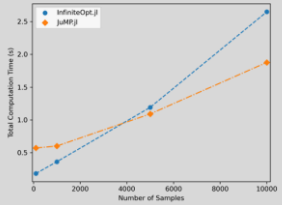
- Unifying Abstraction
  - Captures a wide breadth of problem classes



- High-level Syntax
  - Compactly express problems

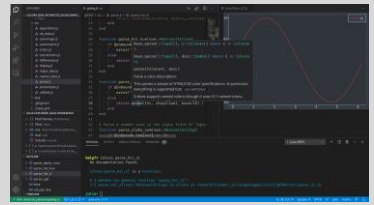
```
@constraint(m, ∂(yb, t) == 2yb^2 + ya - z[1])
@constraint(m, yb ≤ yc * U)
@constraint(m, 𝔼(yc, ξ) ≥ α)
@constraint(m, ya(0) + z[2] == β)
```

- Performance
  - Quickly build complex models at scale



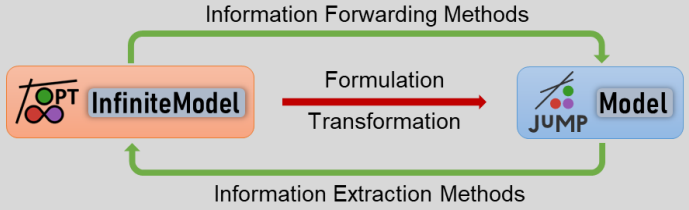
- Built on JuMP.jl
  - Draw from an extensive suite of solvers

GUROBI OPTIMIZATION · COIN-OR · IBM CPLEX · KNITRO



- Open Source
  - It's free!

- Enables Accessible Research
  - Implement advanced methods for general users



KEY TAKEAWAYS

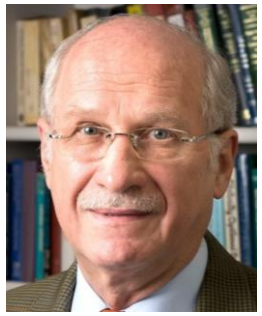
- InfiniteOpt's intuitive interface makes optimal control problems **easier to model** and enables **new modeling objects**

- InfiniteOpt's transformation API automates and **accelerates advanced solution techniques** on CPUs and GPUs

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

UNIVERSITY OF WATERLOO
Department of Chemical Engineering

Carnegie Mellon University
Center for Advanced Process Decision-making

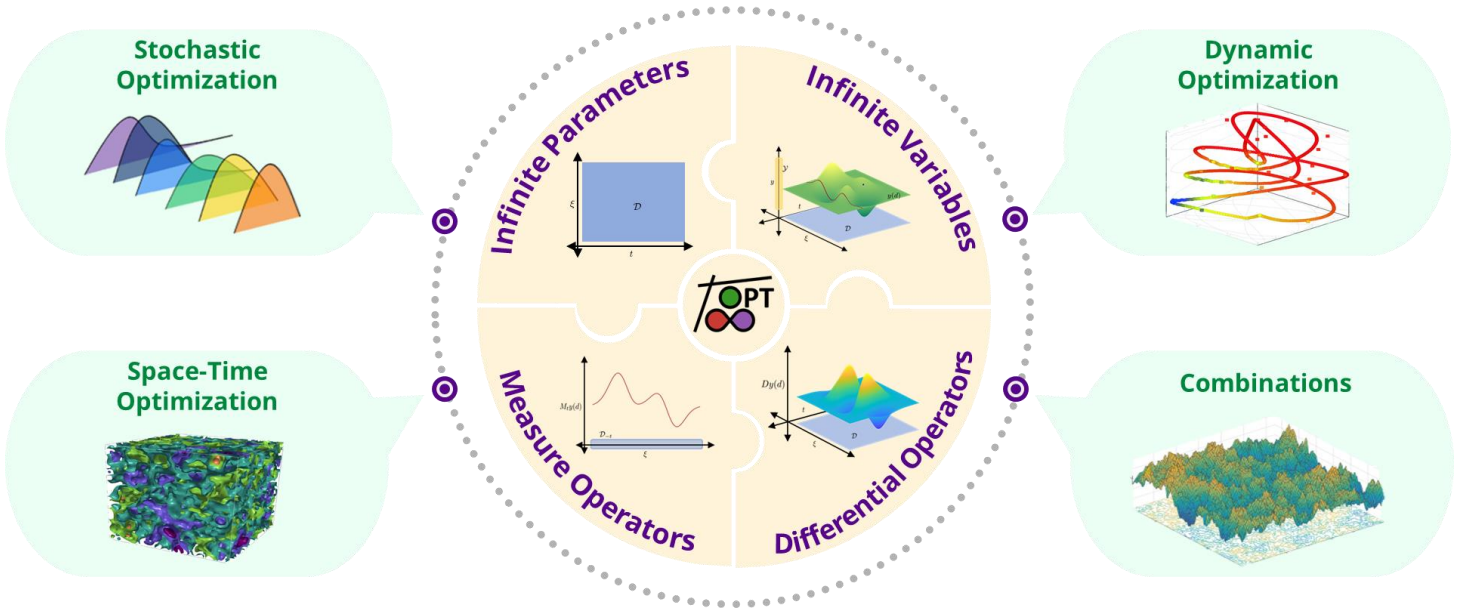
UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

OUTLINE

- Unifying Abstraction
- InfiniteOpt
- New Modeling Objects
- Accelerated Solution Methods

OUTLINE

- **Unifying Abstraction**
- InfiniteOpt
- New Modeling Objects
- Accelerated Solution Methods



WHAT IS AN ABSTRACTION?

Definition: A **general representation** of events/objects comprised of essential **attributes.**

Example: 3 Act Storyline

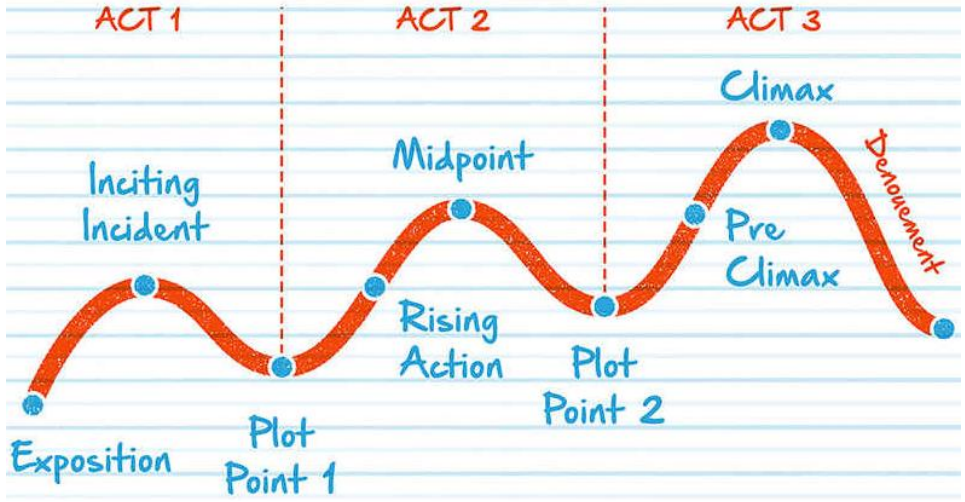
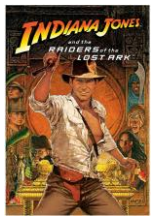


Image from blog.reedsy.com



Power of Abstraction

- Better fundamental understanding
- Facilitates **scientific discovery**
- Enable **accessible modeling** tools

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

MATLAB

DYNAMIC OPTIMIZATION

Dynamic Optimization

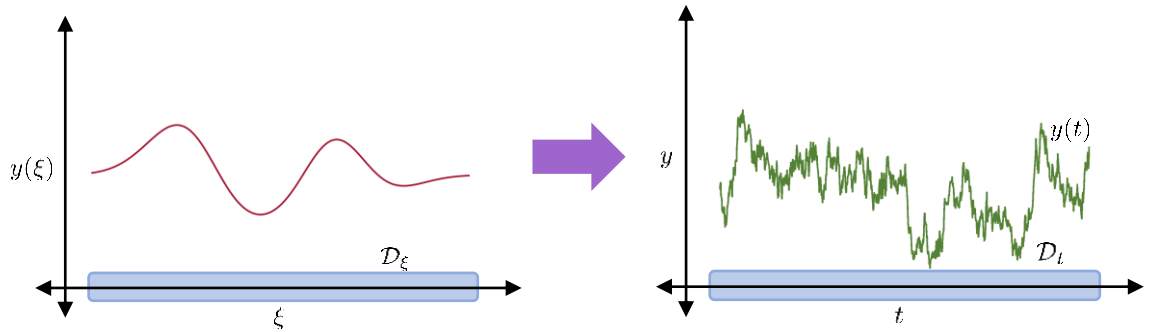
$$\min_{y(t)} \quad \int_{t\in\mathcal{D}_t} f(y(t),t)dt$$
$$\text{s.t.} \quad g(y(t),t) \le 0, \quad t \in \mathcal{D}_t$$

Stochastic Optimization

$$\min_{y(\xi)} \quad \int_{\xi\in\mathcal{D}_\xi} f(y(\xi),\xi)p(\xi)d\xi$$
$$\text{s.t.} \quad g(y(\xi),\xi) \le 0, \quad \xi \in \mathcal{D}_\xi$$

Different problem, similar features

- Replace $\xi \in \mathcal{D}_\xi$ with time $t \in \mathcal{D}_t$
- Let $p(\xi) = 1$
- Shape $y(t)$ instead of $y(\xi)$



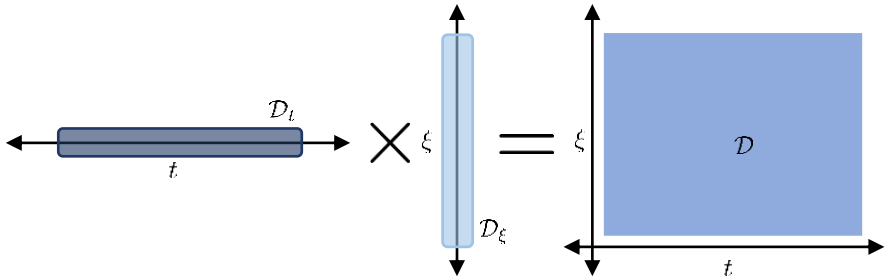
UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

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

INFINITE-DIMENSIONAL OPTIMIZATION

Infinite Parameters

- Index over **continuous domains**



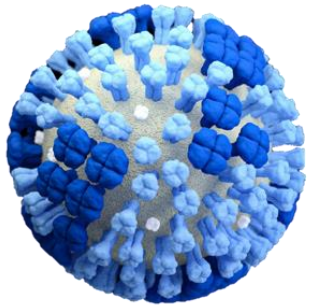
$$t \in \mathcal{D}_t \qquad \xi \in \mathcal{D}_\xi \qquad d \in \mathcal{D} = \mathcal{D}_t \times \mathcal{D}_\xi$$

- **Example:** Disease Control
  - Population dynamics

$$t \in [0, t_f]$$

  - Uncertain infection rates

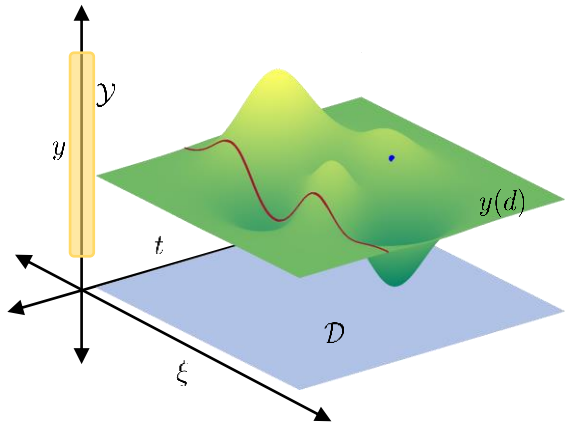
$$\xi \in (-\infty, \infty) \sim \mathcal{N}(\mu, \Sigma)$$



Infinite Variables

- **Decisions** indexed by infinite parameters

$y(d)$



- **Example:** Disease Control
  - Population of infected at a particular time and infection rate

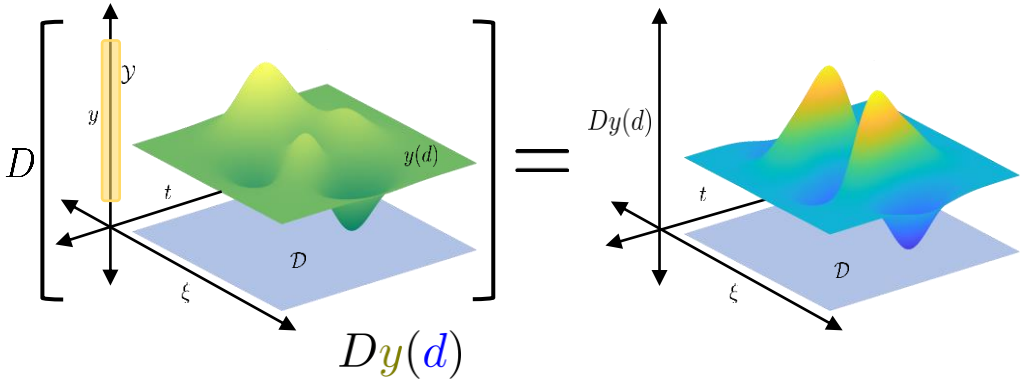
$$y_i(t, \xi)$$

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

INFINITE-DIMENSIONAL OPTIMIZATION

Differential Operators

- Capture of **rate of change** in variables

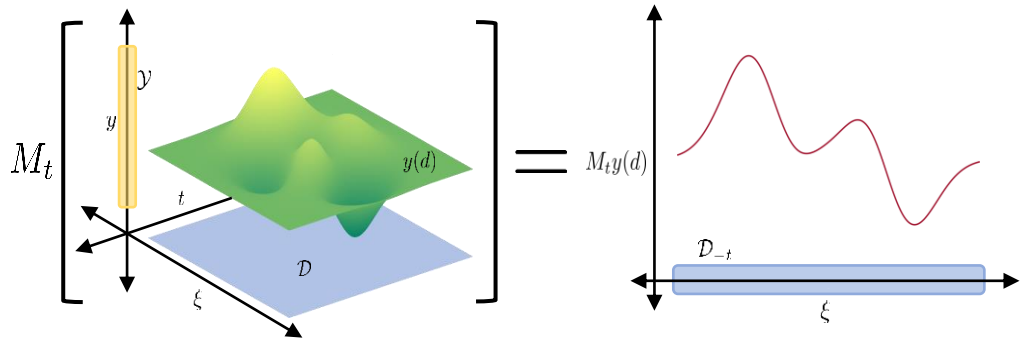


$$Dy(d)$$

- **Example:** Disease Control
  - Time derivative $\dfrac{\partial y_i(t,\xi)}{\partial t}$
  - SEIR model $\dfrac{\partial y_i(t,\xi)}{\partial t} = \xi y_e(t) - \gamma y_i(t)$

Measure Operators

- **Summarize variables** over continuous domains



$$M_d y(d)$$

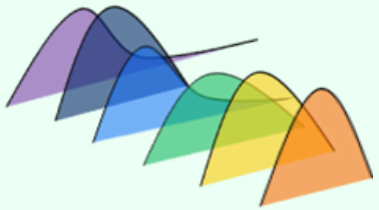
- **Example:** Disease Control
  - Summarize overall infections

$$\int_{t\in\mathcal{D}_t} \mathbb{E}_\xi[y_i(t,\xi)]dt \qquad \mathbb{E}_\xi\left[\int_{t\in\mathcal{D}_t} y_i(t,\xi)dt\right]$$

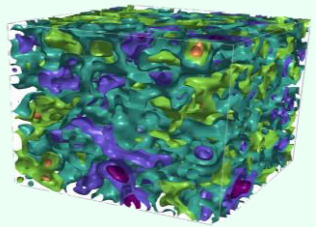
J. L. Pulsipher, W. Zhang, T. J. Hongisto, and V. M. Zavala. "A unifying modeling abstraction for infinite-dimensional optimization." 2022

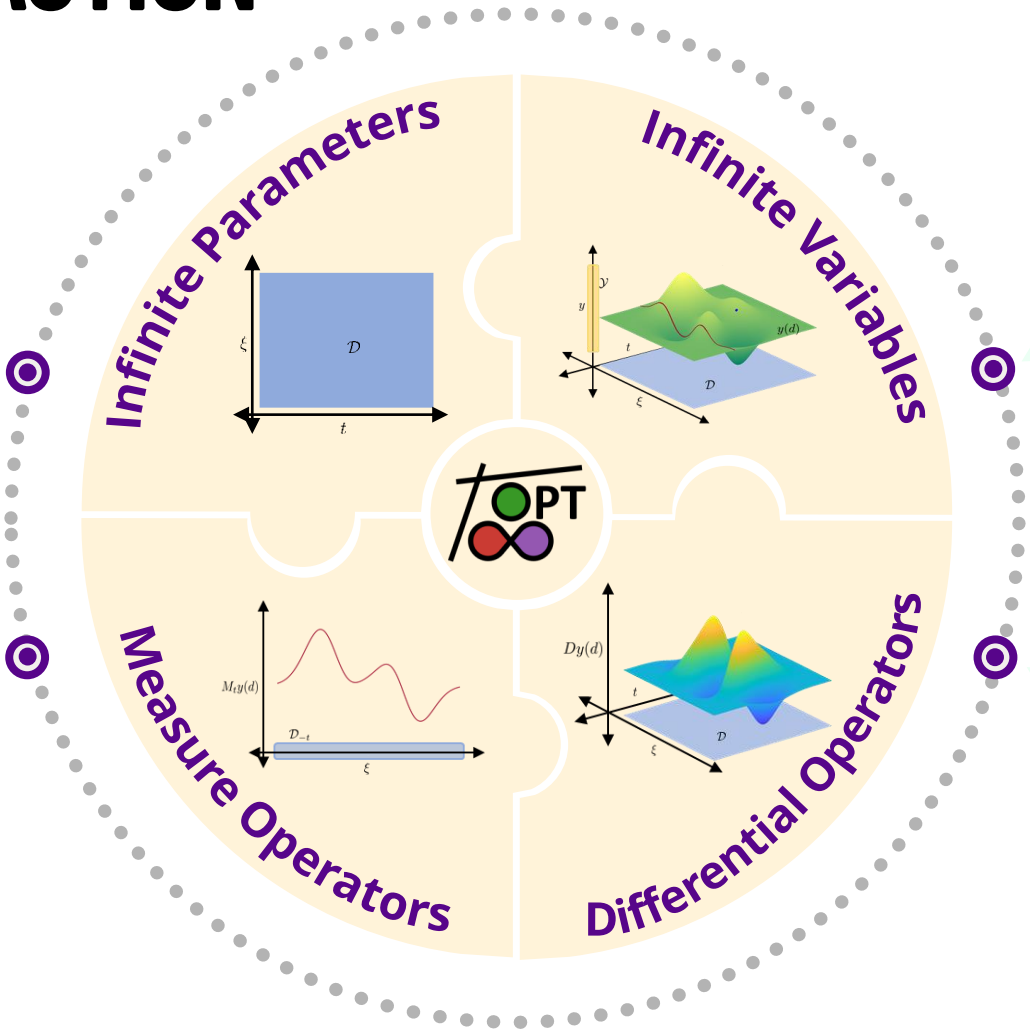
UNIFYING ABSTRACTION

Stochastic Optimization

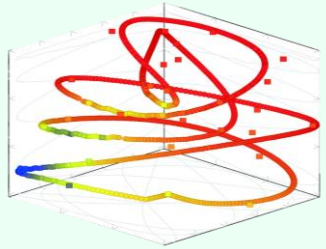


Space-Time Optimization

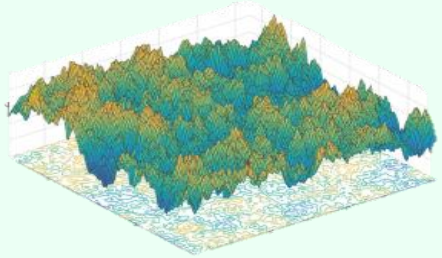




Dynamic Optimization



Combinations



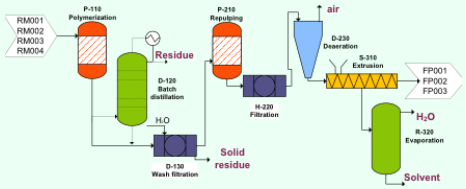
J. L. Pulsipher, W. Zhang, T. J. Hongisto, and V. M. Zavala. "A unifying modeling abstraction for infinite-dimensional optimization." 2022

APPLICATIONS

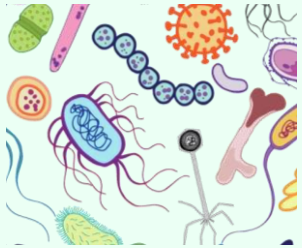
Unifying Abstraction

Dynamic Optimization
(Time)

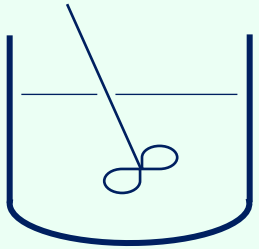
Autonomous Vehicles



Process Control

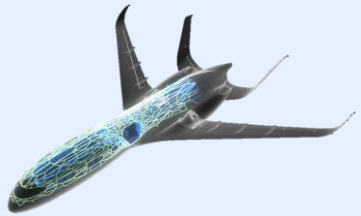


Microbial Communities

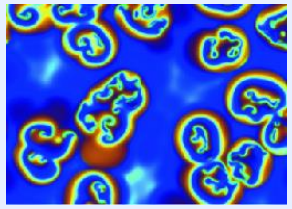


Reaction Systems

PDE-Constrained Optimization
(Space-Time)

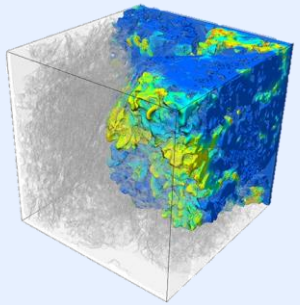


Structural Design



Diffusive Systems

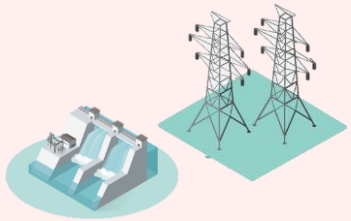
Wildfire Simulation



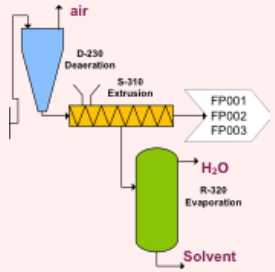
Complex Fluids

Stochastic Optimization
(Uncertainty)

Pharmaceutical Production



Optimal Power Flow



Process Design

Investment Planning

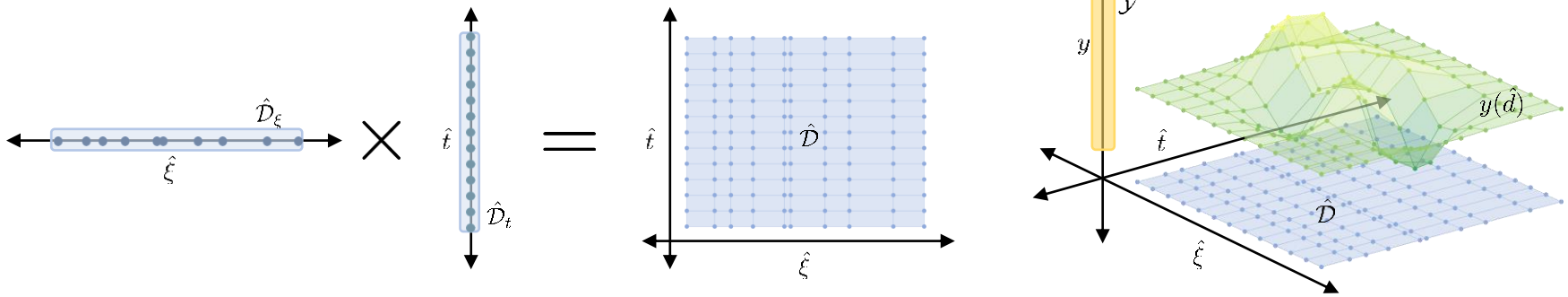
J. L. Pulsipher, W. Zhang, T. J. Hongisto, and V. M. Zavala. "A unifying modeling abstraction for infinite-dimensional optimization." 2022

TRANSFORMING INFINITEOPT PROBLEMS INTO FINITE ONES

Direct Transcription

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

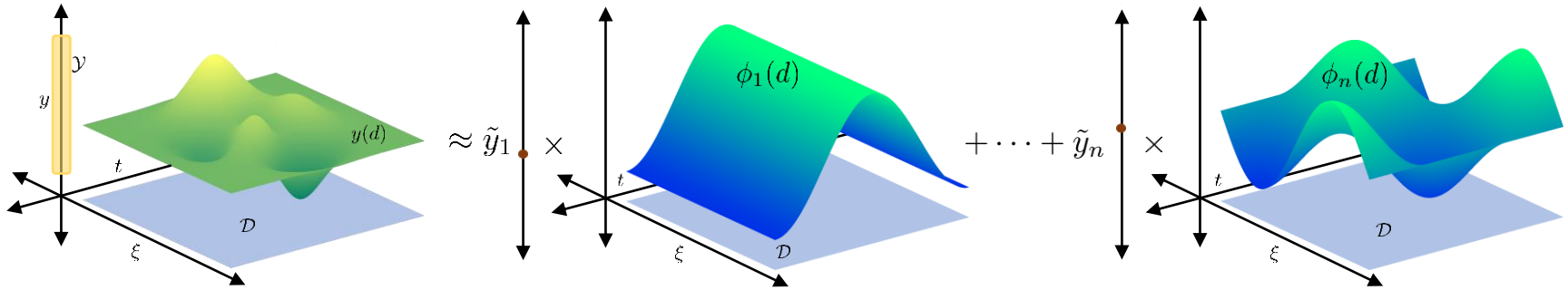
$$\hat{\mathcal{D}} := \prod_{\ell \in \mathcal{L}} \{\hat{d}_{\ell,i} : \hat{d}_{\ell,i} \in \mathcal{D}_\ell, i \in \mathcal{I}_\ell\}$$



Method of Weighted Residuals

Project onto set of known **basis functions**

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



UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

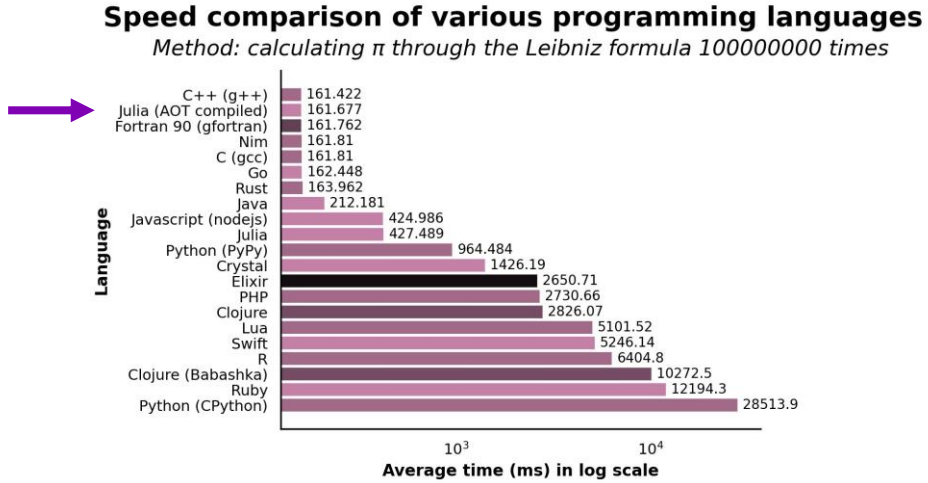
J. L. Pulsipher, W. Zhang, T. J. Hongisto, and V. M. Zavala. "A *Unifying Modeling Abstraction for Infinite-Dimensional Optimization*." Computers & Chem. Eng. 2022

OUTLINE

- Unifying Abstraction
- **InfiniteOpt**
- New Modeling Objects
- Accelerated Solution Methods

WHY JULIA?

- **Fast** as C/C++/Fortran, but as **simple** as Python/MATLAB/R



github.com/coin-or/Ipopt.git
220k lines of code

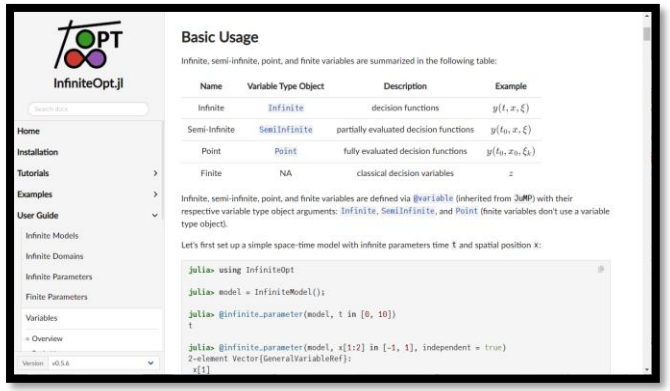
github.com/MadNLP/MadNLP.jl.git
20k lines of code

- **JuMP** provides intuitive and powerful platform for optimization
  - **50+ solver library** + all AMPL and GAMS solvers
  - Extensive amount of **constraint types** supported

InfiniteOpt

Why is it Different?

- Implements **unifying abstraction**
  - Models a wide range of problems
  - Leverages structure to **accelerate solutions**
- Implemented in julia
  - Enables **intuitive** symbolic expressions
  - Highly **performant**
- **Extensive resources**
  - Documentation, tutorials, examples, forum, short courses, videos



Intuitive Modeling API

$$\frac{\partial y_b(t,\xi)}{\partial t} = 2y_b(t,\xi)^2 + y_a(t) - z_1$$
$$\mathbb{E}_\xi\left[y_c(t,\xi)\right] \geq \alpha$$
$$y_a(0) + z_2 = \beta$$

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

Impact

- 1000s of downloads
- Use cases in **diverse disciplines**
  - e.g., evolutionary biology, rocketry, economics, autonomous vehicles

Try it @ https://github.com/infiniteopt/InfiniteOpt.jl

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

WHAT MODELING OBJECTS DOES IT SUPPORT?

InfiniteOpt Specific

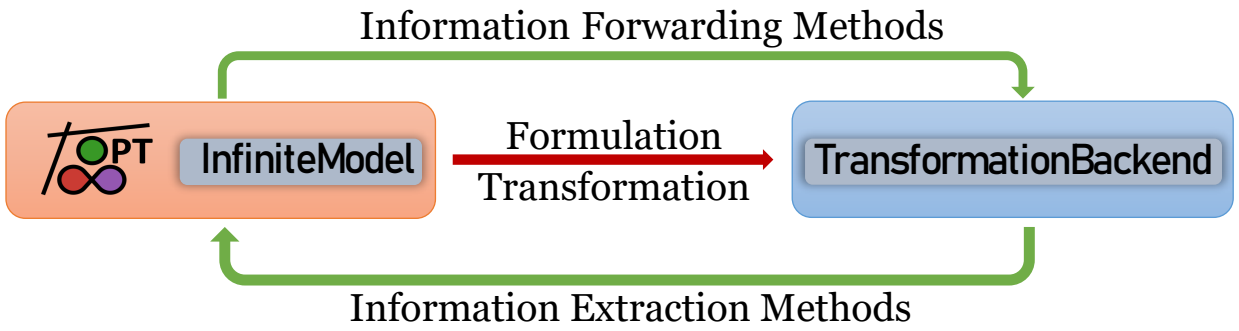
- Infinite parameters
  - Any distribution from Distributions.jl
  - Cartesian domains (e.g., time/space)
- Integrals
- Expectations
- Ordinary differential equations
- Partial differential equations
- Chance constraints

Inherited from JuMP

- Affine/quadratic/nonlinear/mixed-integer models
- Constraint programming
- Vector-valued constraints
- Conic constraints
- Complementarity constraints
- Multi-objectives
- Indicator constraints
- Many many more…

TRANSFORMING INFINITEOPT MODELS

Transformation Paradigm

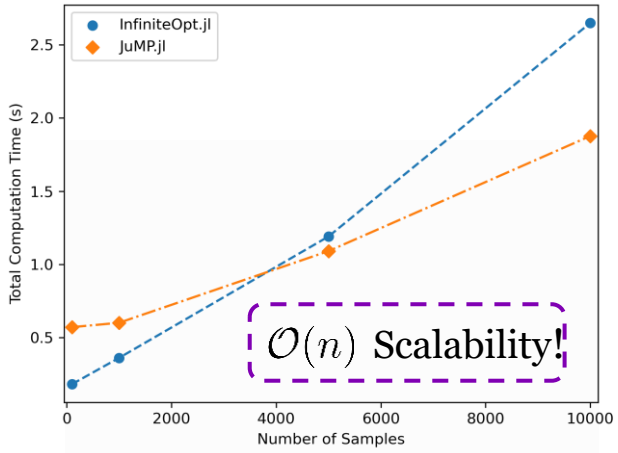


Transformation API

- Highly extensible to **make advanced solution techniques accessible/automated**
- Detailed templates, tutorials, and docs

Automated Transcription (via TranscriptionOpt)

- Many **derivative/measure approximations**
  - Orthogonal collocation, Gauss quadrature, etc.
- **Performant**



InfiniteOpt.jl
↓
Automatic Transcription

JuMP.jl
↓
Manual Transcription

MODELING SYNTAX

- Initialize the **model**

```
using InfiniteOpt, Distributions, Ipopt
m = InfiniteModel(Ipopt.Optimizer)
```

- Add the **infinite parameters**

$$t \in [t_0, t_f] \qquad \xi \sim \mathcal{N}(\mu, \Sigma)$$

```
@infinite_parameter(m, t ∈ [t0, tf], num_supports = 10)
@infinite_parameter(m, ξ[1:10] ~ MvNormal(μ, Σ))
```

- Add **variables** and their domain constraints

$$y_a(t) \in \mathbb{R}_+ \quad y_b(t, \xi) \in \mathbb{R}_+ \quad y_c(\xi) \in \{0, 1\} \quad z \in \mathbb{Z}^2$$

```
@variable(m, ya ≥ 0, Infinite(t))
@variable(m, yb ≥ 0, Infinite(t, ξ))
@variable(m, yc, Infinite(ξ), Bin)
@variable(m, z[1:2], Int)
```

- Define the **objective**

$$\min_{y_a(t), y_b(t,\xi), y_c(\xi), z} \int_{t \in \mathcal{D}_t} y_a(t)^2 + 2\mathbb{E}_\xi[y_b(t, \xi)]dt$$

```
@objective(m, Min, ∫(ya^2 + 2 * 𝔼(yb, ξ), t))
```

- Add the **constraints**

$$\frac{\partial y_b(t,\xi)}{\partial t} = 2y_b(t,\xi)^2 + y_a(t) - z_1, \quad \forall t \in \mathcal{D}_t,\ \xi \in \mathcal{D}_\xi$$

$$\mathbb{P}\left(y_b(t,\xi) \leq 0\right) \geq \alpha, \qquad \forall t \in \mathcal{D}_t$$

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

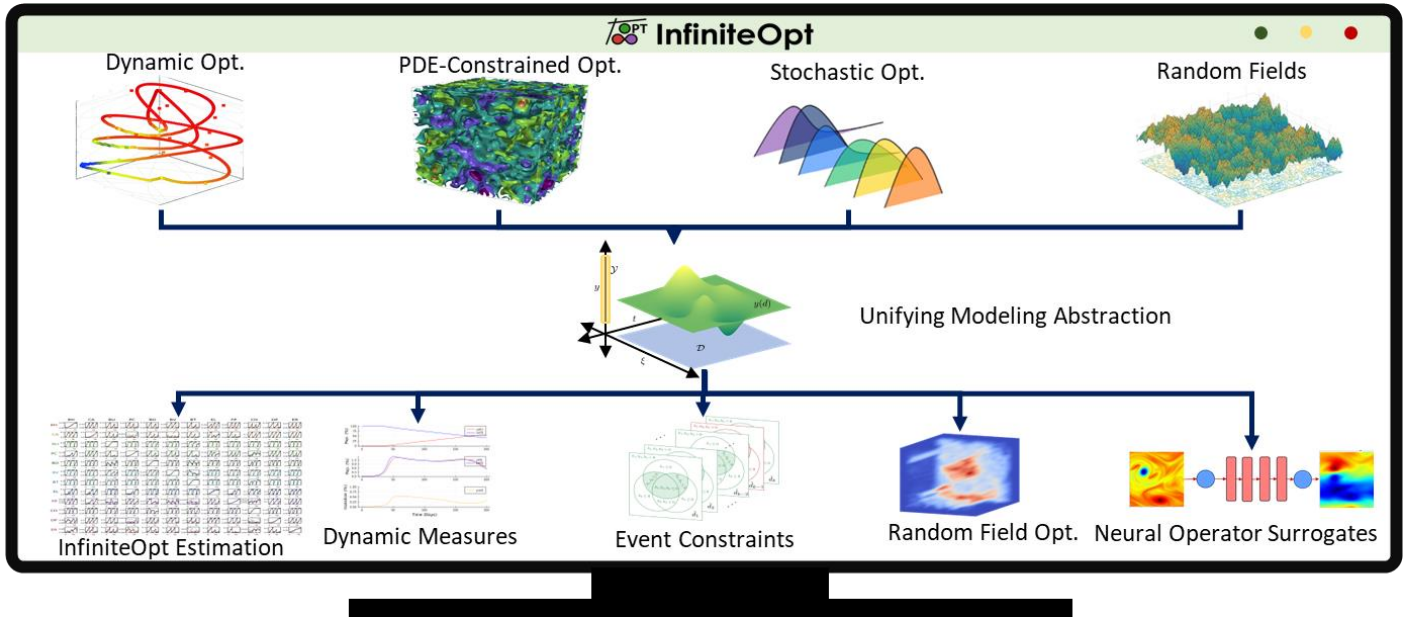
```
@constraint(m, ∂(yb, t) == 2yb^2 + ya - z[1])
@constraint(m, yb ≤ yc * U)
@constraint(m, 𝔼(yc, ξ) ≥ α)
@constraint(m, ya(0) + z[2] == β)
```

- **Transform and Solve**

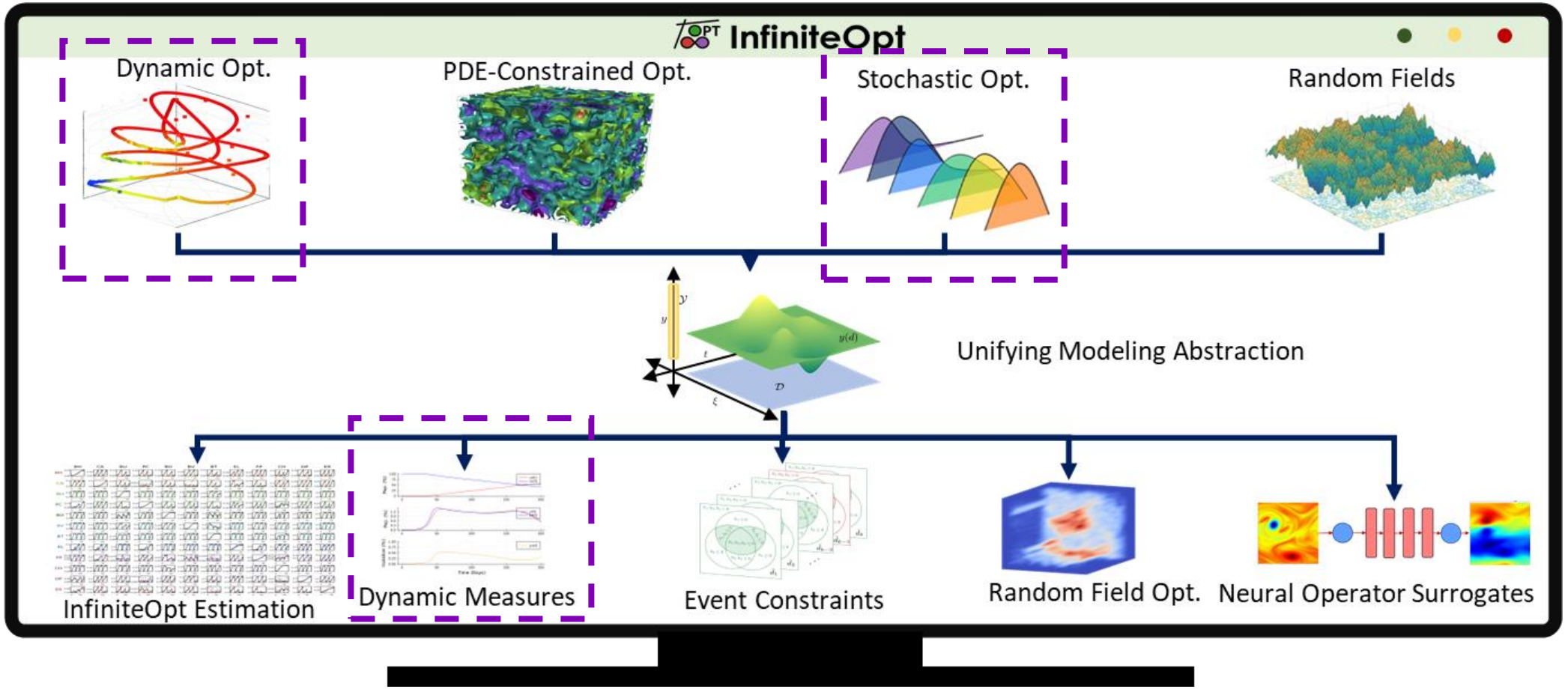
```
optimize!(m)
opt_objective = objective_value(m)
```

OUTLINE

- Unifying Abstraction
- InfiniteOpt
- **New Modeling Objects**
- Accelerated Solution Methods



INNOVATION VIA CROSS POLLINATION



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

J. L. Pulsipher, B. Davidson, and V. M. Zavala. "New Measures for Shaping Trajectories in Dynamic Optimization." 2022

J. L. Pulsipher, Hongisto, and V. M. Zavala. "Random Field Optimization." 2023

D. Ovalle, H. Perez, I. E. Grossmann, C. D. Laird, **J. L. Pulsipher**. "Event Constrained Programming". In progress

STOCHASTIC OPTIMIZATION

Formulation

$$\min_{z\in\mathcal{Z},y(\xi)\in\mathcal{Y}} \quad M_\xi f(z, y(\xi), \xi)$$
$$\text{s.t.} \quad g(z, y(\xi), \xi) \leq 0, \quad \xi \in \mathcal{D}_\xi$$

- Random infinite parameter

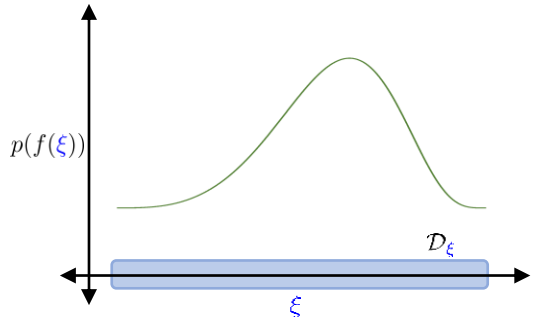
$$\xi : \Omega \mapsto \mathcal{D}_\xi \qquad (\Omega, \mathscr{F}, F)$$

- Measured cost function of 1st/2nd stage

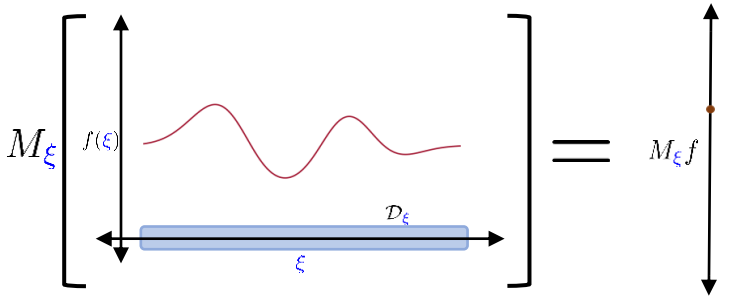
- 2nd stage recourse constraints

Risk Measures

- Shape the random **cost density function**



- Directly shape the random **cost function**



J. L. Pulsipher, B. Davidson, and V. M. Zavala. "New Measures for Shaping Trajectories in Dynamic Optimization." 2022

STOCHASTIC RISK MEASURES

Expectation

- Integral weighted via **density function**

$$\mathbb{E}_{\xi}[f(\xi)] := \int_{\xi \in \mathcal{D}_{\xi}} f(\xi) p(\xi) d\xi$$

Mean-Variance

- Penalize the cost spread/**fluctuation**

$$\mathbb{E}\text{-}\mathbb{V}_{\xi}[f(\xi)] := \mathbb{E}_{\xi}[f(\xi)] + \lambda \mathbb{V}_{\xi}[f(\xi)]$$

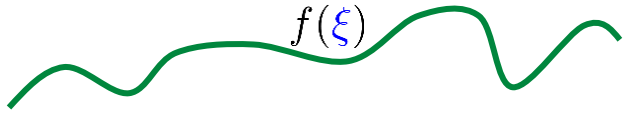
CVaR

- Penalize the $1-\alpha$ **large cost values**

$$\mathrm{CVaR}_{\xi}(f(\xi);\alpha) := \min_{\hat{f} \in \mathbb{R}} \left\{ \hat{f} + \frac{1}{1-\alpha} \mathbb{E}_{\xi}[f(\xi) - \hat{f}]_{+} \right\}$$

Disutility

- Penalize the **unfavorable cost outcomes**



$$\tilde{D}_{\xi}(f(\xi)) := \inf_{\hat{f} \in \mathbb{R}} \mathbb{E}_{\xi}[f(\xi) + s(f(\xi) - \hat{f})]$$

A. Ruszczynski and A. Shapiro. "Value-at-risk vs. conditional value-at-risk in risk management and optimization." 2008

TIME-VALUED RISK MEASURES

Expectation

- Integral with **weighting function**

$$\mathbb{E}_t[f(t)] := \int_{t \in \mathcal{D}_t} f(t)p(t)dt$$

Mean-Variance

- Penalize the cost **fluctuation**
- Uses weighting function

$$\mathbb{E}\text{-}\mathbb{V}_t[f(t)] := \mathbb{E}_t[f(t)] + \lambda \mathbb{V}_t[f(t)]$$

CVaR

- Penalize the $1 - \alpha$ fraction **largest cost values**
- Uses weighting function

$$\text{CVaR}_t(f(t); \alpha) := \min_{\hat{f} \in \mathbb{R}} \left\{ \hat{f} + \frac{1}{1-\alpha} \mathbb{E}_t[f(t) - \hat{f}]_+ \right\}$$

Disutility

- Penalize the **unfavorable cost outcomes**
- Uses weighting function

$$\tilde{D}_t(f(t)) := \inf_{\hat{f} \in \mathbb{R}} \mathbb{E}_t[f(t) + s(f(t) - \hat{f})]$$

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

J. L. Pulsipher, B. Davidson, and V. M. Zavala. "New Measures for Shaping Trajectories in Dynamic Optimization." 2022

OPTIMAL DISEASE CONTROL

Formulation

- Limit infected and **minimize closures**
- Use the SEIR disease model
- Let cost function be control action

$$
\begin{aligned}
\min \quad & M_t y_u(t) \\
\text{s.t.} \quad & \dot{y}_s(t) = (y_u(t) - 1)\,\beta y_s(t) y_i(t),\ t \in \mathcal{D}_t \\
& \dot{y}_e(t) = (1 - y_u(t))\,\beta y_s(t) y_i(t) - \xi y_e(t),\ t \in \mathcal{D}_t \\
& \dot{y}_i(t) = \xi y_e(t) - \gamma y_i(t),\ t \in \mathcal{D}_t \\
& \dot{y}_r(t) = \gamma y_i(t),\ t \in \mathcal{D}_t \\
& y_s(0) = s_0, y_e(0) = e_0, y_i(0) = i_0, y_r(0) = r_0 \\
& y_i(t) \le \overline{y_i},\ t \in \mathcal{D}_t \\
& y_u(t) \in [0, \overline{y_u}],\ t \in \mathcal{D}_t
\end{aligned}
$$

Implementation

InfiniteOpt

```
using InfiniteOpt, Ipopt

# Set the parameters
γ, β, ξ = 0.303, 0.727, 0.3
s0, e0, i0, r0 = 1 - 1e-5, 1e-5, 0, 0

# Define the model
m = InfiniteModel(Ipopt.Optimizer)

# Define the time parameter
@infinite_parameter(m, t ∈ [0, 200], num_supports = 101)

# Add the variables
@variable(m, ys, Infinite(t))
@variable(m, ye, Infinite(t))
@variable(m, yi ≤ 0.02, Infinite(t))
@variable(m, yr, Infinite(t))
@variable(m, 0 ≤ yu ≤ 0.8, Infinite(t))

# Set the time expectation objective
@objective(m, Min, E(yu, t))

# Define the SEIR equations
@constraint(m, ∂(ys, t) == -(1 - yu) * β * ys * yi)
@constraint(m, ∂(ye, t) == (1 - yu) * β * ys * yi - ξ * ye)
@constraint(m, ∂(yi, t) == ξ * ye - γ * yi)
@constraint(m, ∂(yr, t) == γ * yi)
@constraint(m, ys(0) == ys0)
@constraint(m, ye(0) == ye0)
@constraint(m, yi(0) == yi0)
@constraint(m, yr(0) == yr0)

# Solve the model and retrieve results
optimize!(m)
u_opt = value(yu)
ts = value(t)
```

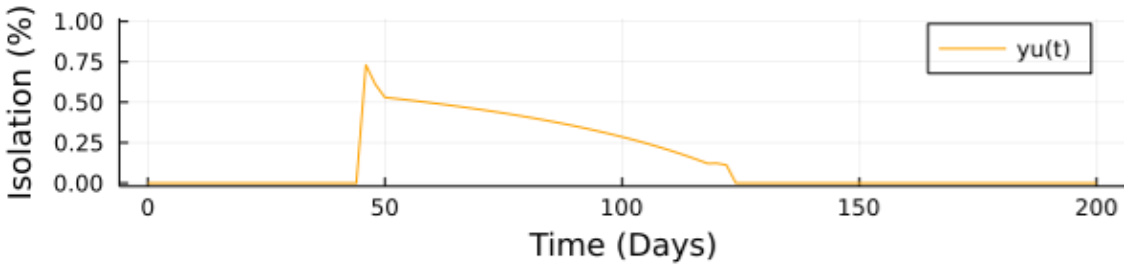
UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

J. L. Pulsipher, B. Davidson, and V. M. Zavala. "New Measures for Shaping Trajectories in Dynamic Optimization." 2022

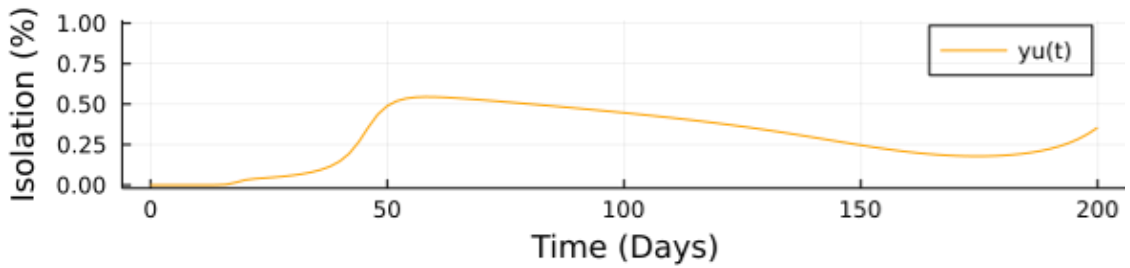
OPTIMAL COST/CONTROL TRAJECTORIES

Expectation (Uniform pdf)

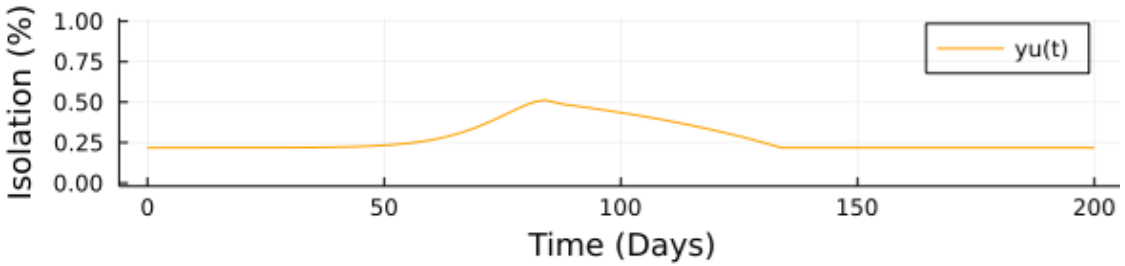
(Same as traditional Bolza objective)



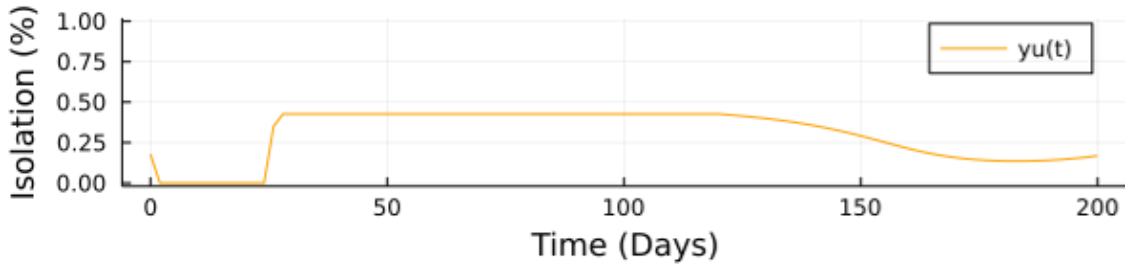
Expectation (Exponential pdf)



Mean-Variance ($\lambda = 8$)



CVaR ($\alpha = 0.9$)

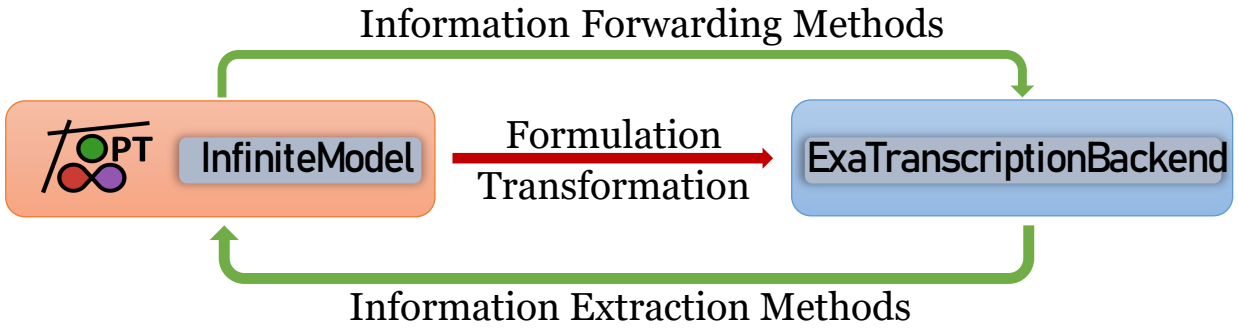


UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

J. L. Pulsipher, B. Davidson, and V. M. Zavala. "New Measures for Shaping Trajectories in Dynamic Optimization." 2022

OUTLINE

- Unifying Abstraction
- InfiniteOpt
- New Modeling Objects
- **Accelerated Solution Methods**

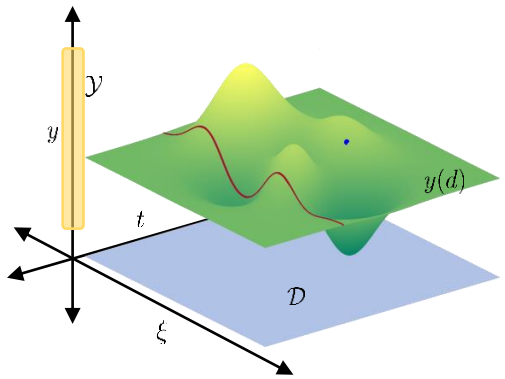


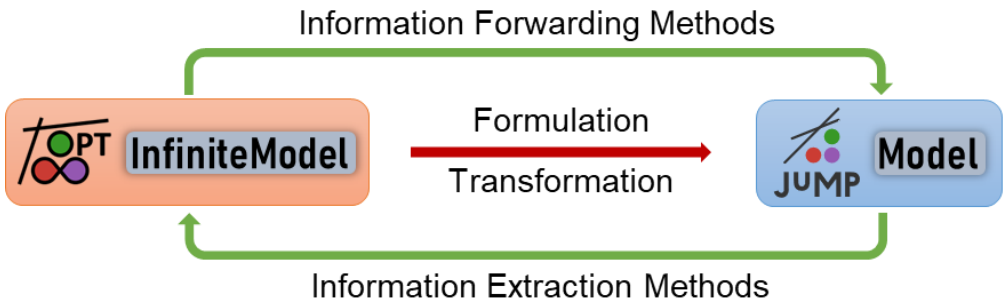
SOLVING INFINITEOPT PROBLEMS VIA TRANSCRIPTIONOPT

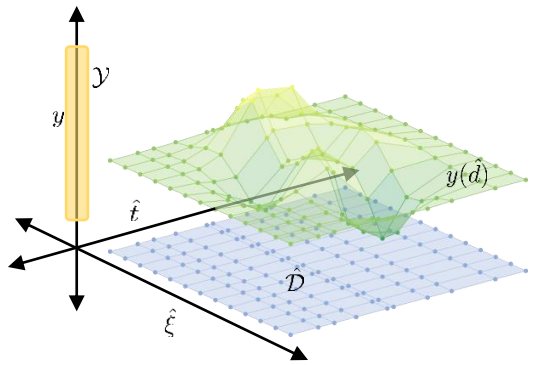
- Apply **transformation** to obtain finite JuMP model that can be solved
- InfiniteOpt has a large suite of **discretization** techniques
- Discretized InfiniteOpt problems have **repeated structure**
- Traditional modeling languages like JuMP do not leverage repeated structure

$$\sin^2(y(t)) \leq 42, \ t \in \mathcal{D}_t$$

$$\sin^2(y_k) \leq 42, \ k \in \mathcal{K}$$



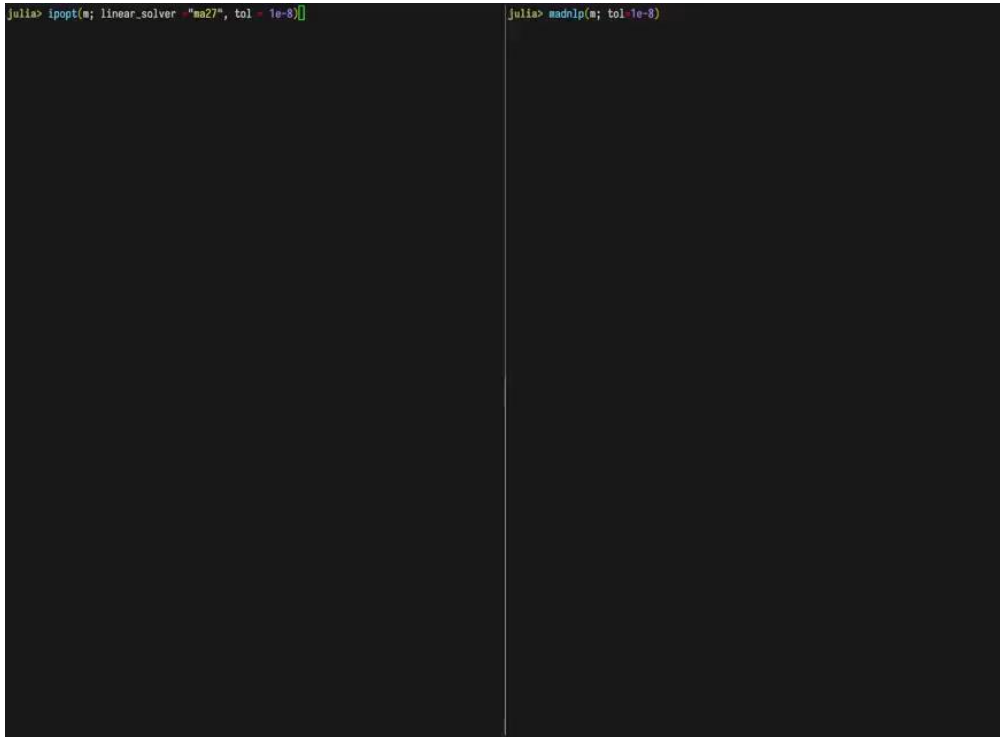




Can we leverage the repeated structure to **accelerate solution performance**?

ACCELERATING NLP PERFORMANCE ON CPUS AND GPUS

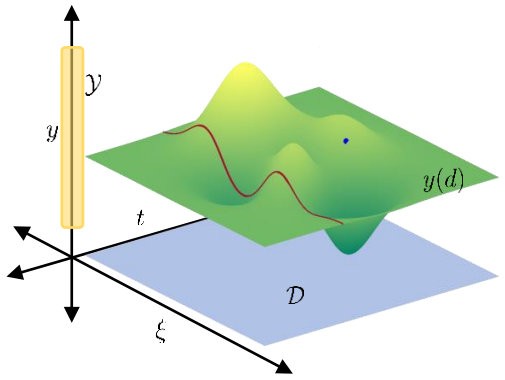
- ExaModels uses a single instruction multiple data (SIMD) abstraction
  - Well-suited for optimization problems with **highly repeated structures**
  - Efficiently **compute gradients** for each structure
- MadNLP solves NLPs on GPUs
  - Depends on CUDASS linear solver
  - Obtain solutions **10-20 times faster**
- Translating InfiniteOpt problems to SIMD is nontrivial

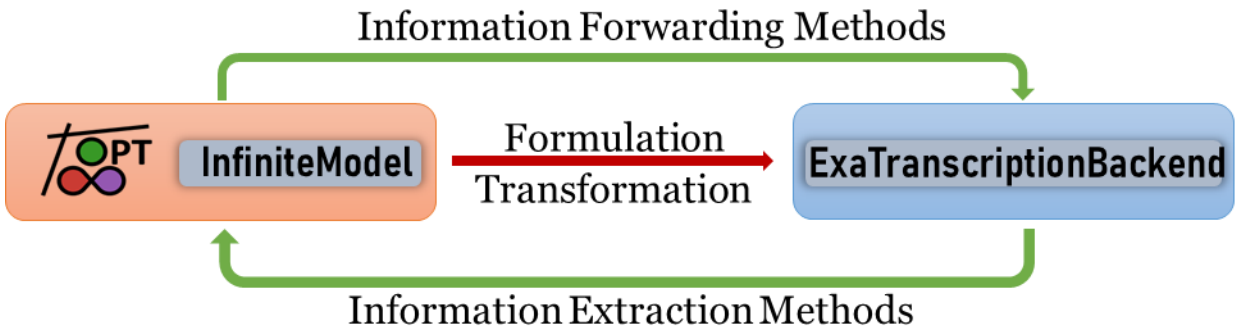


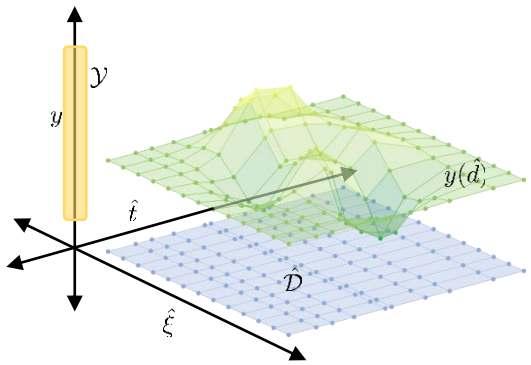
S. Shin, F. Pacaud, and M. Anitescu. "Accelerating optimal power flow with gpus: Simd abstraction of nonlinear programs and condensed-space interior-point methods." 2023

INFINITEEXAMODELS.JL

- Bridges the gap between InfiniteOpt & ExaModels
- **Automates transcription** via established transformation interface
- Leverages repeated structure to **drastically reduce model creation time**
  - More efficient than manual transcription directly given to ExaModels







J. Pulsipher and S. Shin. "Scalable Modeling of Infinite-Dimensional Nonlinear Programs with InfiniteExaModels.jl." 2023

BENCHMARK PROBLEMS

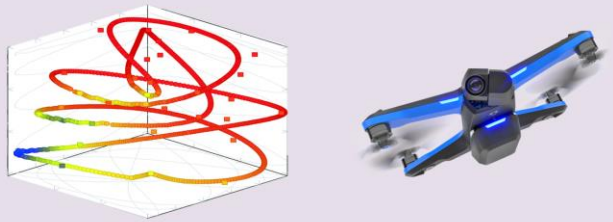
- Compare performance with JuMP, AMPL, ExaModels, and InfiniteExaModels
- Run on CPU with Ipopt and GPU with MadNLP

2-Stage Stochastic Program

- Stochastic optimal power flow
- 1,000 to 16,000 random scenarios

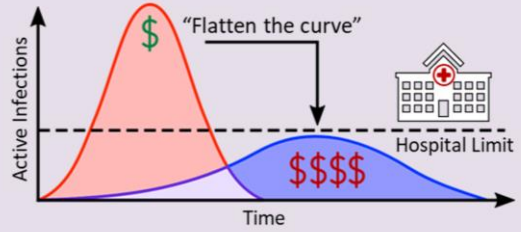
Optimal Control

- Model predictive control of quadcopter
- Track trajectory setpoint and vary grid size



Stochastic Optimal Control

- Control isolation policy to combat disease
- Uncertain transmission rate

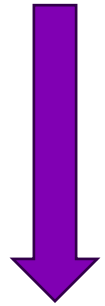


J. Pulsipher and S. Shin. "Scalable Modeling of Infinite-Dimensional Nonlinear Programs with InfiniteExaModels.jl." 2023

NUMERICAL RESULTS (CPU W/ IPOPT)

- AD is **5 – 20 times faster**
- Model build time is **1 – 2 orders-of-magnitude faster**

	Stochastic 2-Stage				Optimal Control				Stochastic Control			
Approach	**Build**	**AD**	**Solve**	**Tot.**	**Build**	**AD**	**Solve**	**Tot.**	**Build**	**AD**	**Solve**	**Tot.**
JuMP.jl	87.1	21.4	63.7	151	143	6.3	28.6	172	10.9	60.7	386	397
AMPL	99.3	11.5	90.3	190	153	5	26.4	179	10.6	23.4	364	375
ExaModelsMOI.jl	86.6	3.05	56.3	143	139	1.1	23.1	162	8.63	3.45	368	376
InfiniteExaModels.jl	1.94	2.03	43	45	9.34	1	22.6	31.9	0.12	3.01	369	369



```
1    using InfiniteOpt, InfiniteExaModels, NLPModelsIpopt
2    model = InfiniteModel(ExaTranscriptionBackend(IpoptSolver))
```

J. Pulsipher and S. Shin. "Scalable Modeling of Infinite-Dimensional Nonlinear Programs with InfiniteExaModels.jl." 2023

NUMERICAL RESULTS (GPU W/ MADNLP)

- All AD and solve times are up to **~20 faster on GPU**
- InfiniteExaModels.jl builds models **orders-of-magnitude faster** than ExaModels

	Stochastic 2-Stage				Optimal Control				Stochastic Control			
Approach	**Build**	**AD**	**Solve**	**Tot.**	**Build**	**AD**	**Solve**	**Tot.**	**Build**	**AD**	**Solve**	**Tot.**
ExaModelsMOI.jl	83	0.09	3.15	86.1	133	0.1	6.55	140	8.21	0.51	20.4	28.6
InfiniteExaModels.jl	1.8	0.14	3.03	4.82	8.63	0.1	6.44	15.1	0.06	0.74	21	21

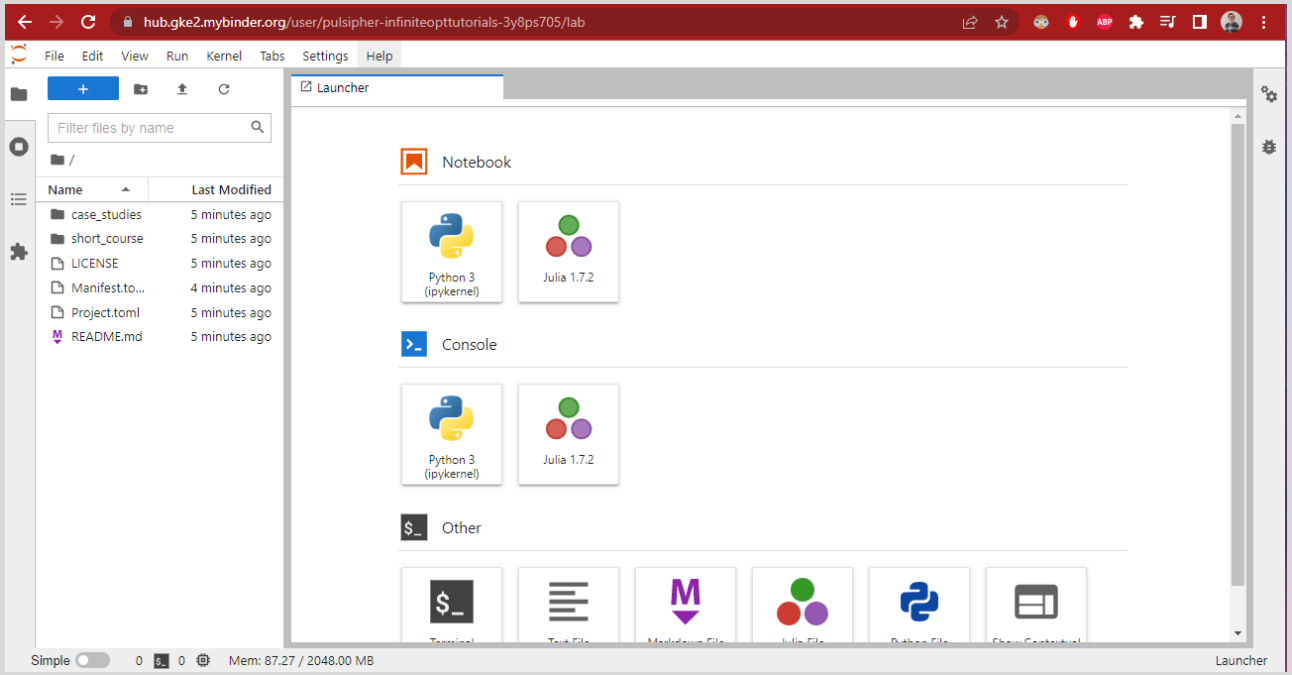
(151 on CPU) *(172 on CPU)* *(397 on CPU)*

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

Installation and Setup

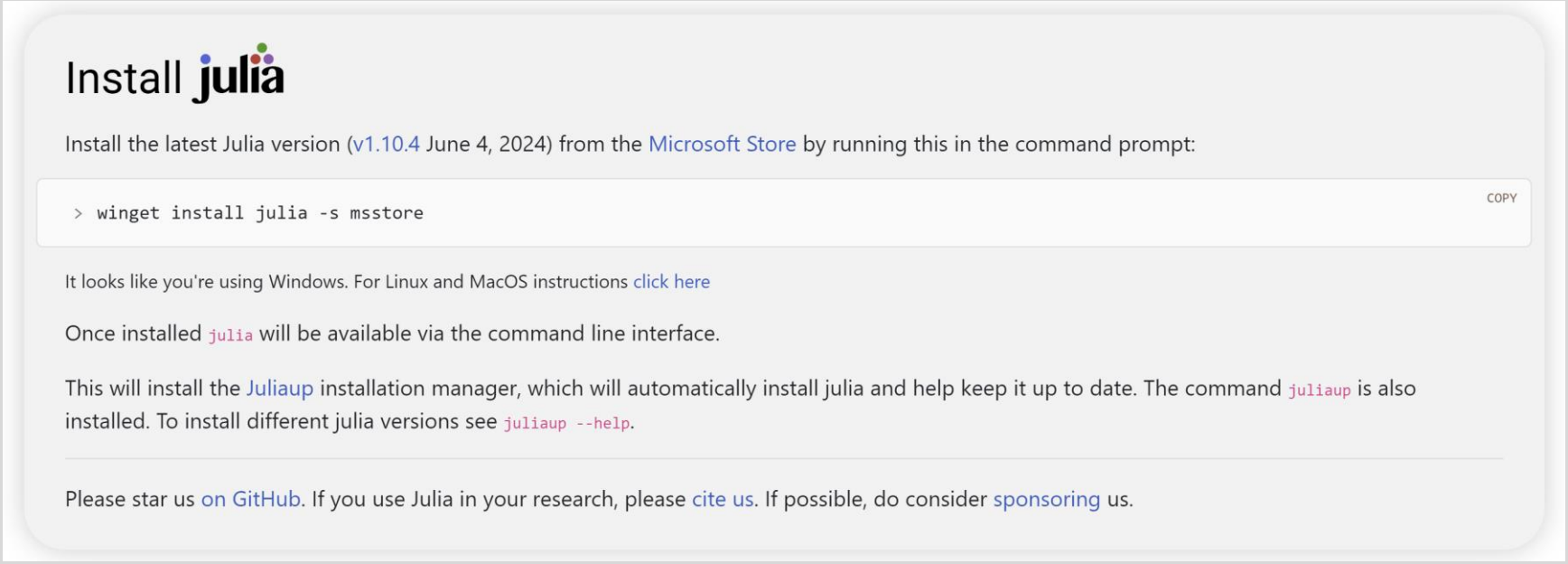
Alternative: Binder

- Online interface via Binder: https://mybinder.org/v2/gh/pulsipher/InfiniteOptTutorials/main
- This is free, but slow to load
- Limited computing resources for optimizing models
- If you are using ChromeOS or Android then you'll have to use this



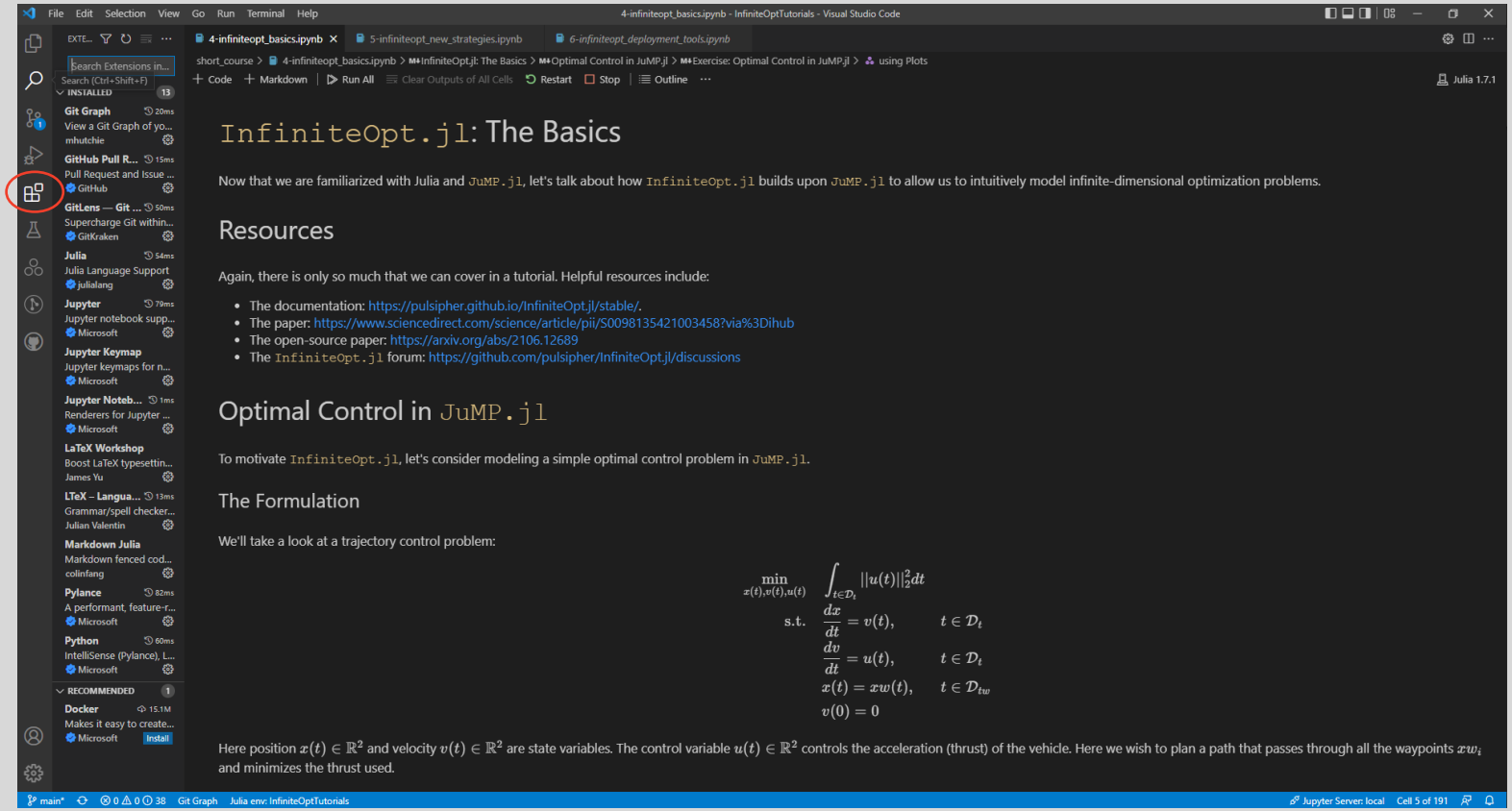
Install Julia

- Download latest Julia 1.10 (need at least 1.6) from https://julialang.org/downloads/
  - *Windows:* Choose 64-bit installer
  - *macOS:* Choose the x86 installer, do not use the ARM build even if you have an M-series processor
  - *Linux:* Choose the appropriate one for your distribution of Linux
  - Be sure to add it to the PATH (not the default)



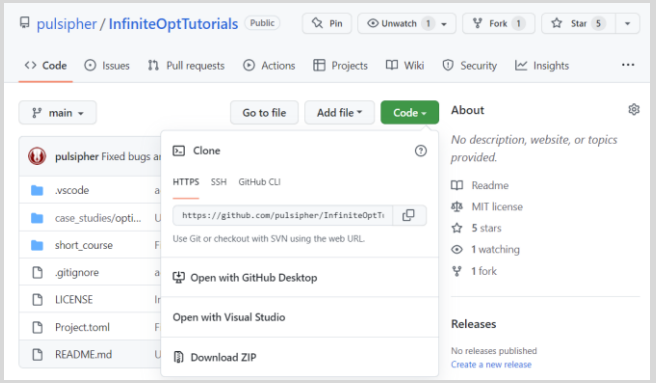
Install Visual Studio Code

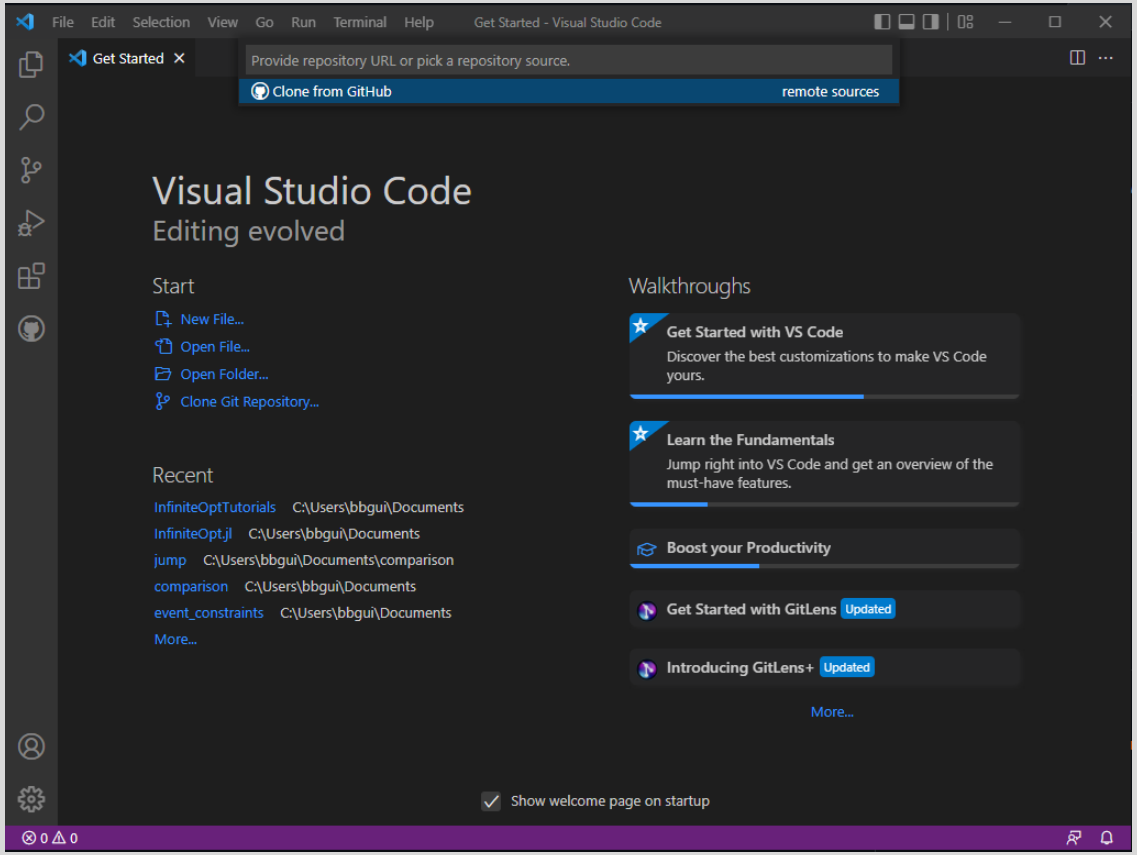
- Download from https://code.visualstudio.com/
  - The default options are fine
- Install the extensions
  - Julia
  - Jupyter



Download the Course Material

- Located at https://github.com/pulsipher/InfiniteOptTutorials
  - Can Google "InfiniteOptTutorials" + "pulsipher"
- Download via VS Code (if git installed)
  - Click "Clone Git Repository"
  - Paste in the link
- Or download it from GitHub
  - Clone it
  - Or download the zip file
  - Or click the "Open with Visual Studio" button
  - Then open the folder in VS Code





Setup the Package Environment

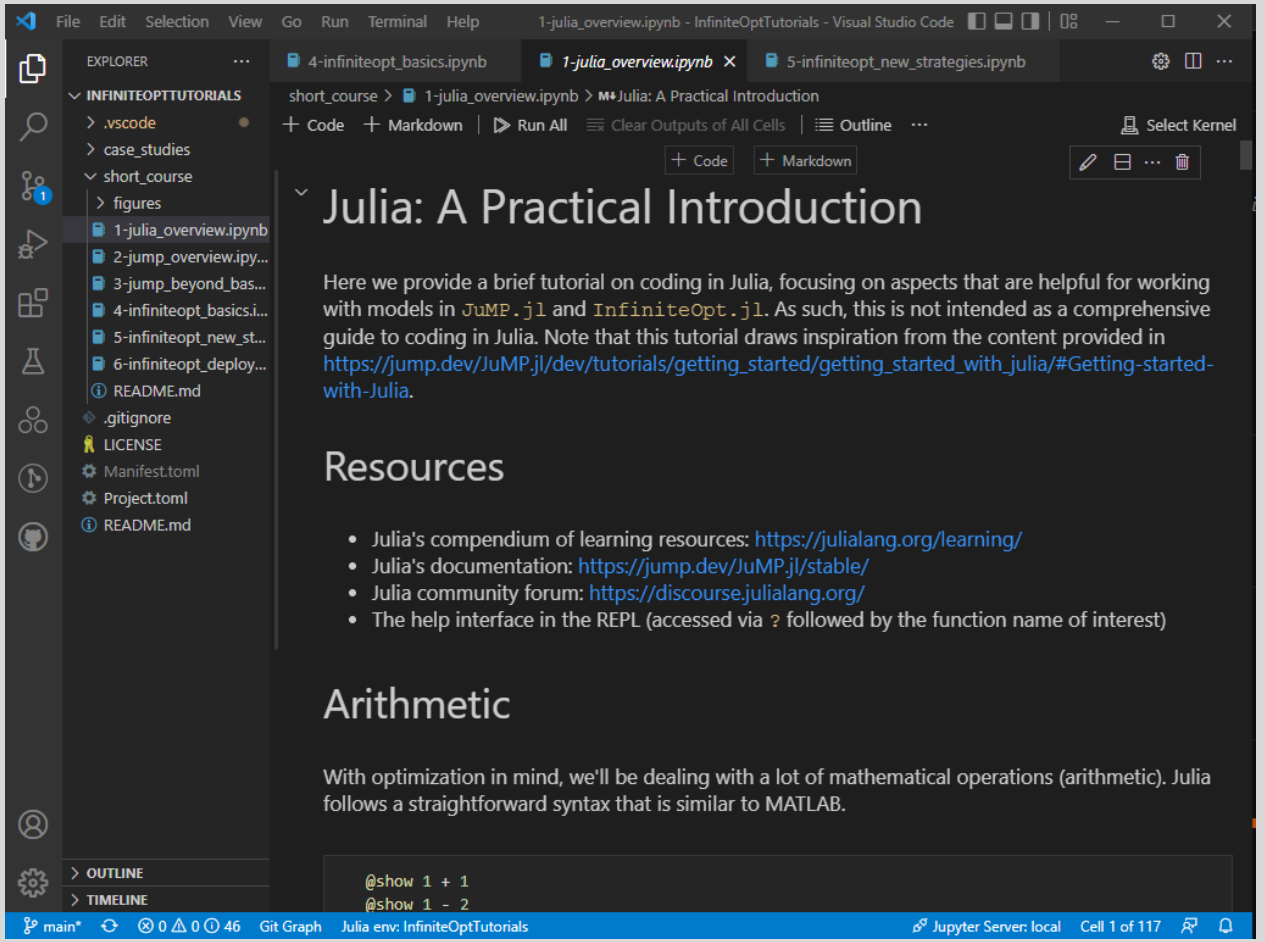
- Now that everything is installed let's setup the Julia environment
- Make sure the open the "InfiniteOptTutorials" folder in VS Code first
- Steps
  - Click on "Julia env: …" on the button bar
  - Select "InfiniteOptTutorials" from the drop-down bar
  - Now under "View" select "Command Palette"
  - Type "Julia: start repl" and push enter
  - In the Julia REPL type "]" and press enter
  - Now you should see "(InfiniteOptTutorials) pkg>"
  - Type "instantiate" and press enter
- Load the packages
  - Return to `julia>` via backspace
  - Load the packages via `using InfiniteOpt, Plots, HiGHS, Ipopt`

```
(InfiniteOptTutorials) pkg> instantiate
```

```
julia> using InfiniteOpt, Plots, HiGHS, Ipopt
```

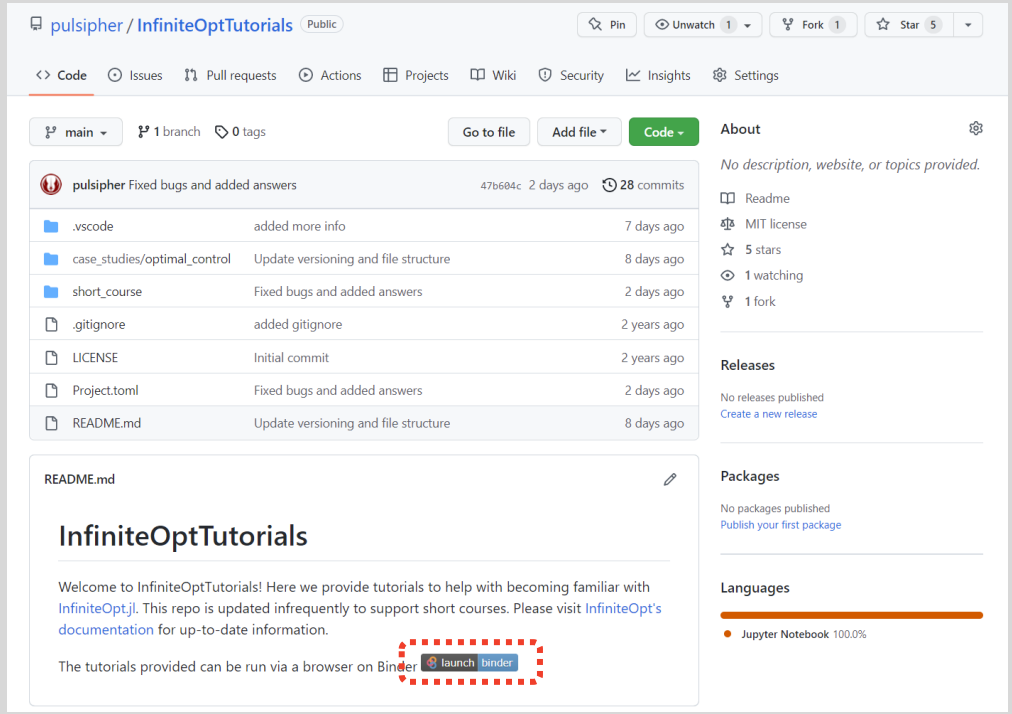
Running Jupyter Notebook

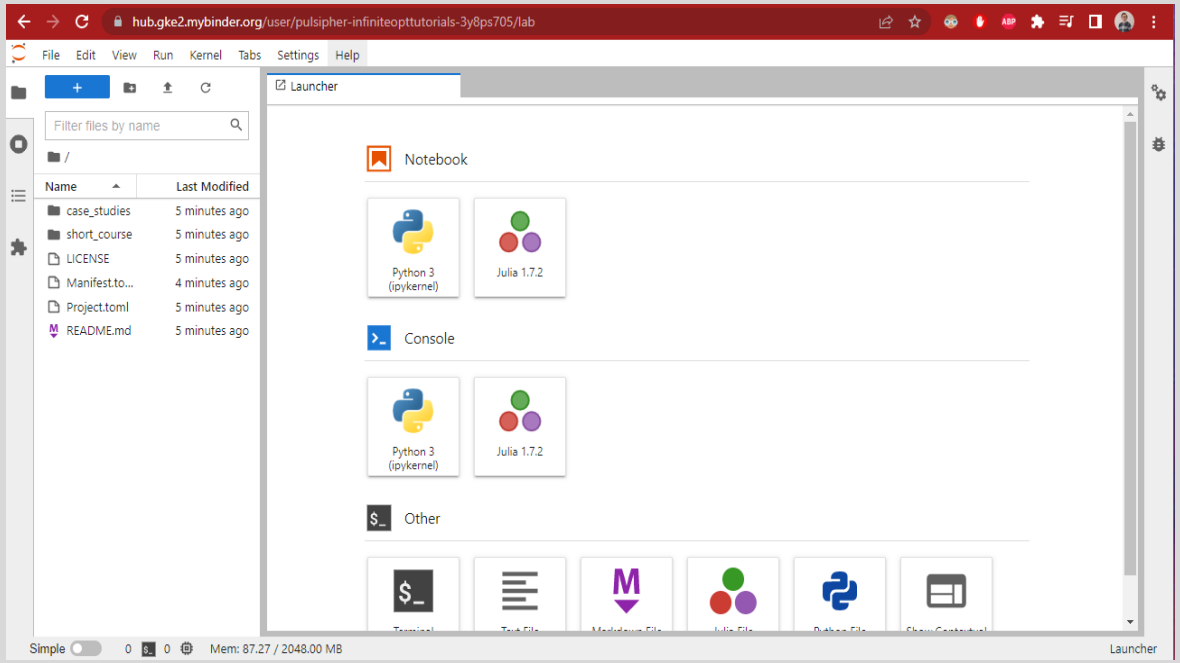
- Origins
  - Iterative scripting for Julia, Python, and R
  - That's where the "Ju" comes from
- IJulia
  - Supported via the IJulia package
- VS Code
  - VS code supports Jupyter notebooks directly
  - No setup needed
  - Select the Julia kernel when prompted



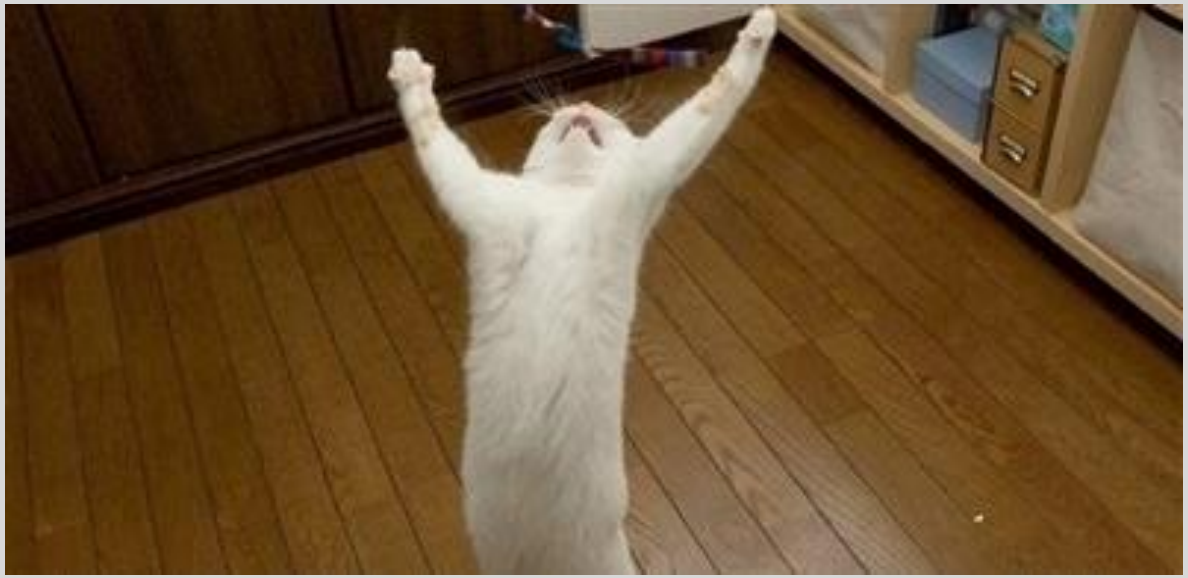
Having Trouble? Just use Binder Instead

- Online interface via Binder: https://mybinder.org/v2/gh/pulsipher/InfiniteOptTutorials/main
- This is free, but slow to load
- Limited computing resources for optimizing models





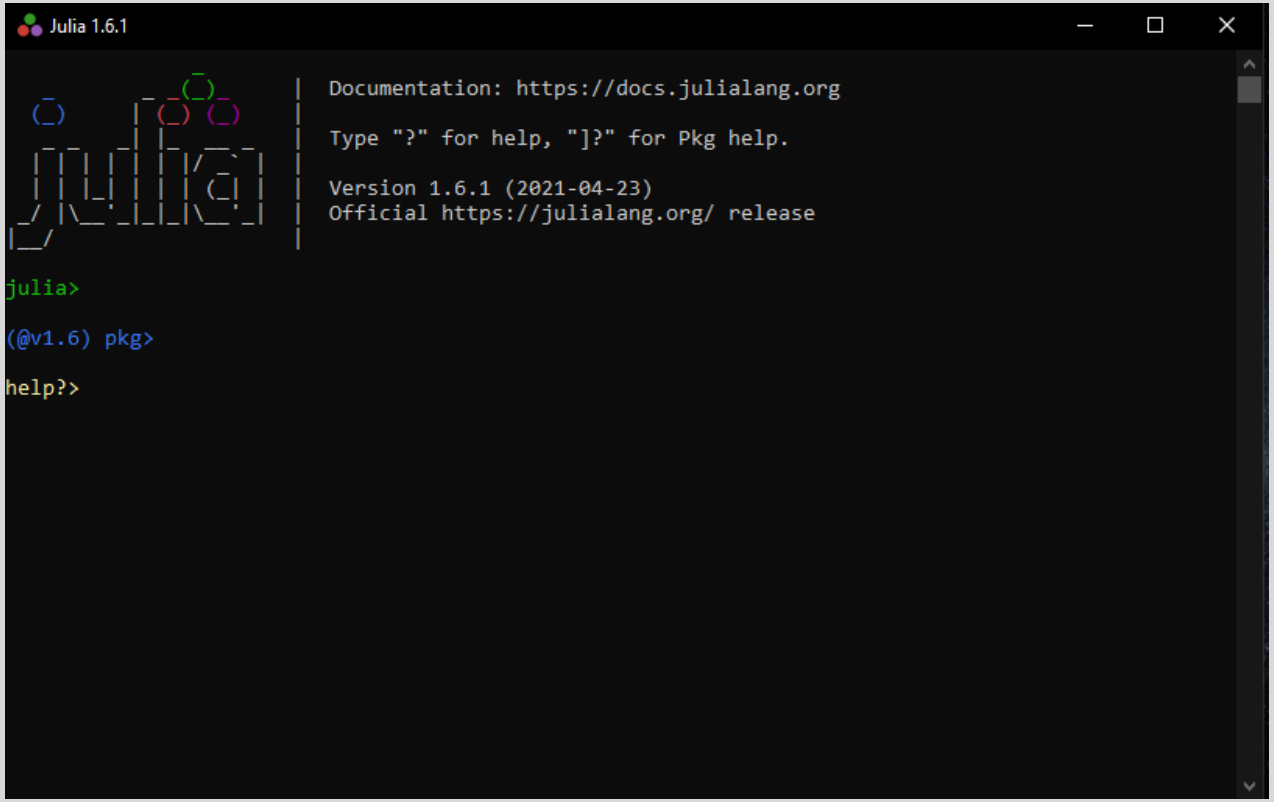
15 Minute Break Time



A Practical Introduction

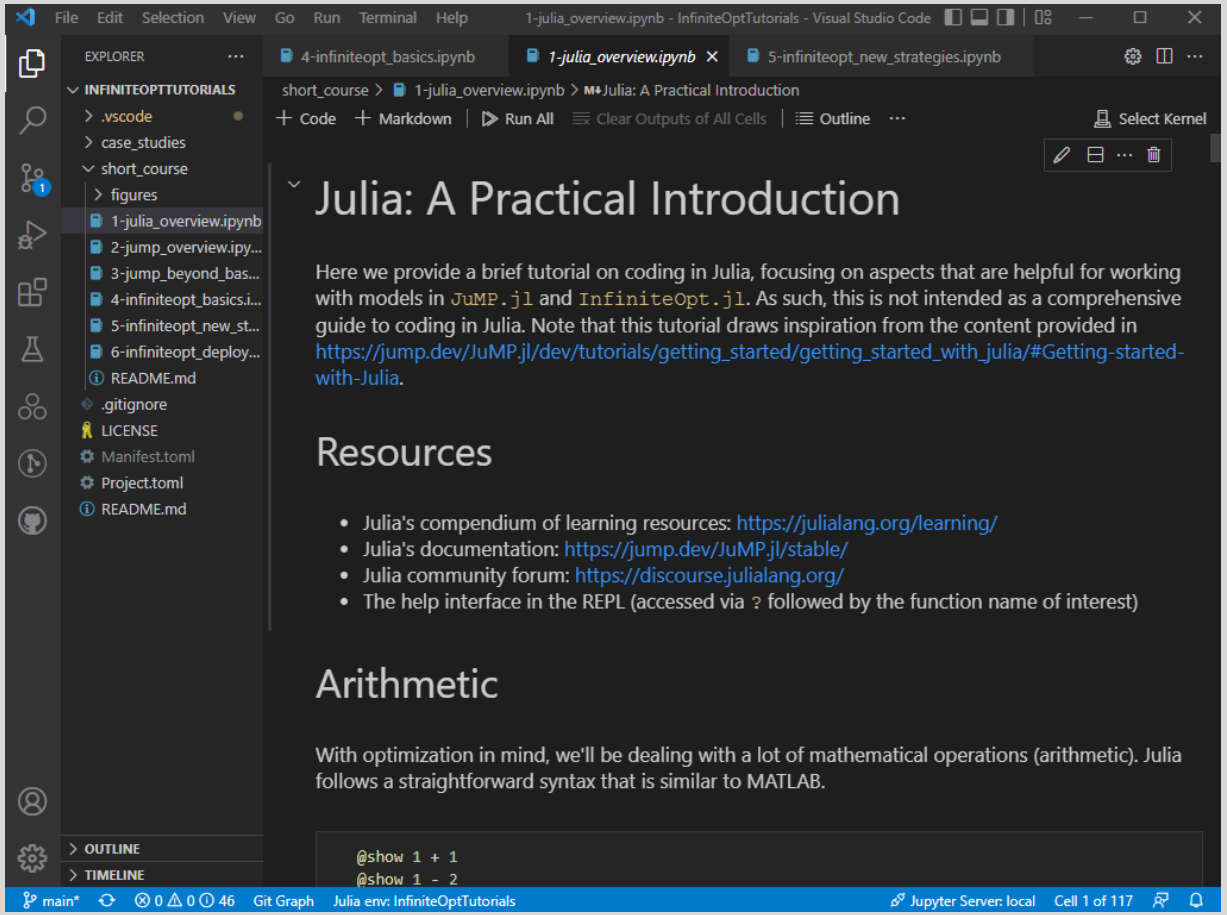
The REPL

- What is it?
  - Read-Evaluate-Print-Loop
  - Executes commands, code, scripts
- What can I do with
  - Run quick commands
  - Run entire scripts
  - Play around
  - Get help
  - This is analogous to the Python kernel
- Package management
  - Use the package mode
  - More on this later



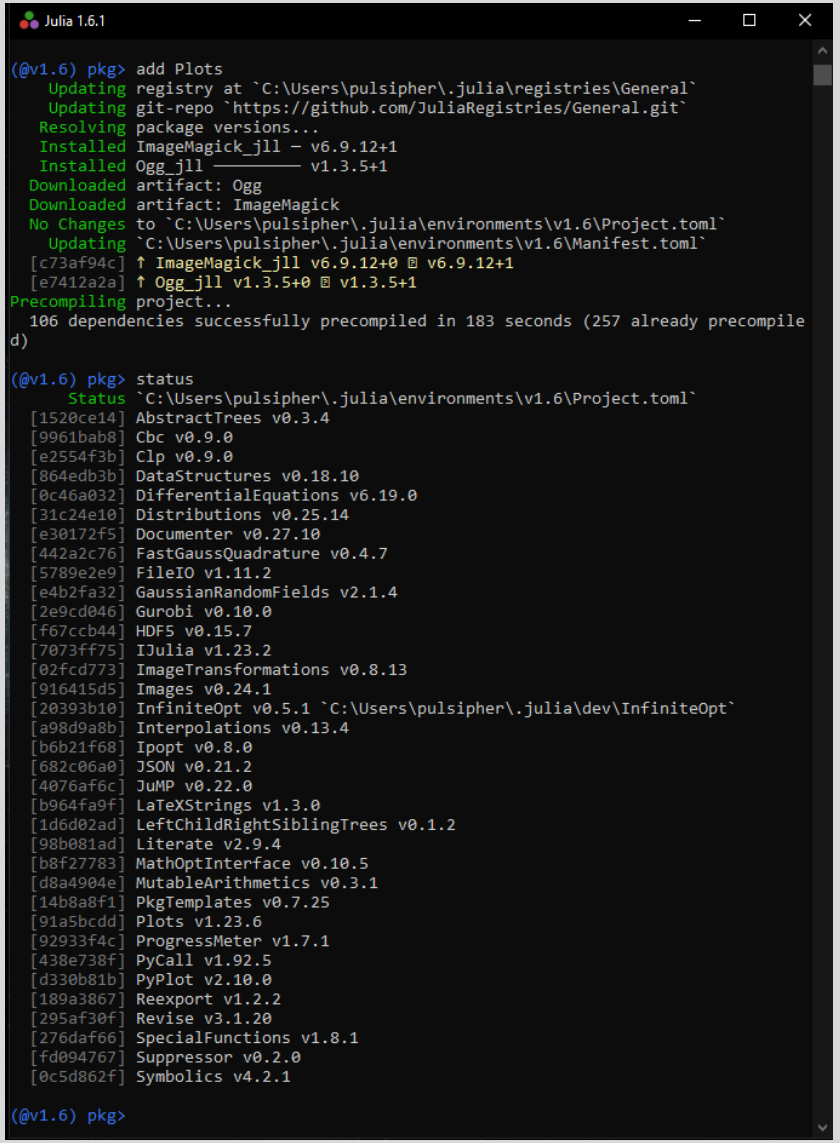
To the Tutorial!

- Open the "1-julia_overview.ipynb" jupyter notebook



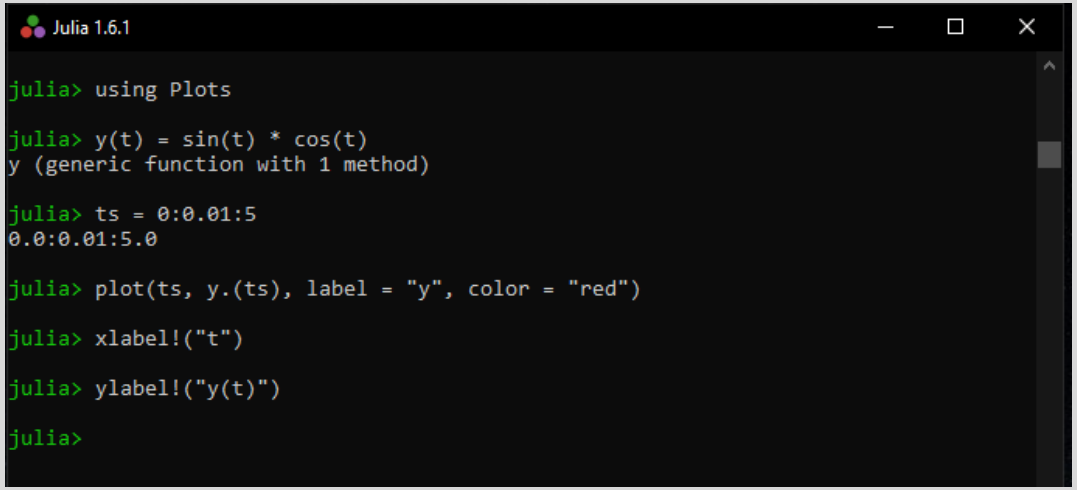
Packages

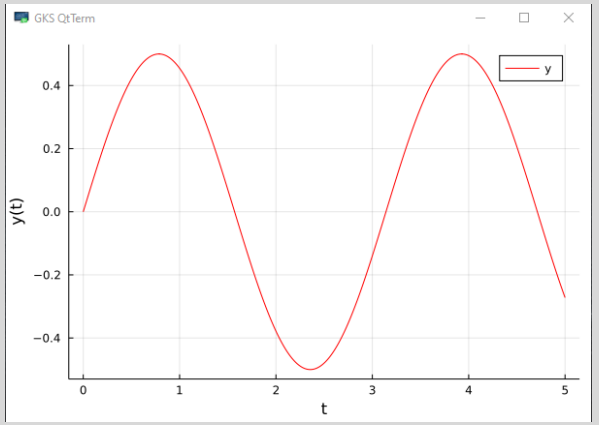
- Libraries of abilities
  - Nearly 7,000 packages are available
  - Access a wealth of functions and capabilities
- Package manager
  - Use the Pkg.jl interface in the REPL
  - Manages dependencies
- Environments
  - An independent collection of packages
  - Stops packages from stepping on each others toes
  - See documentation for details
- Importing
  - Use the `using` or `import` keywords



Plotting

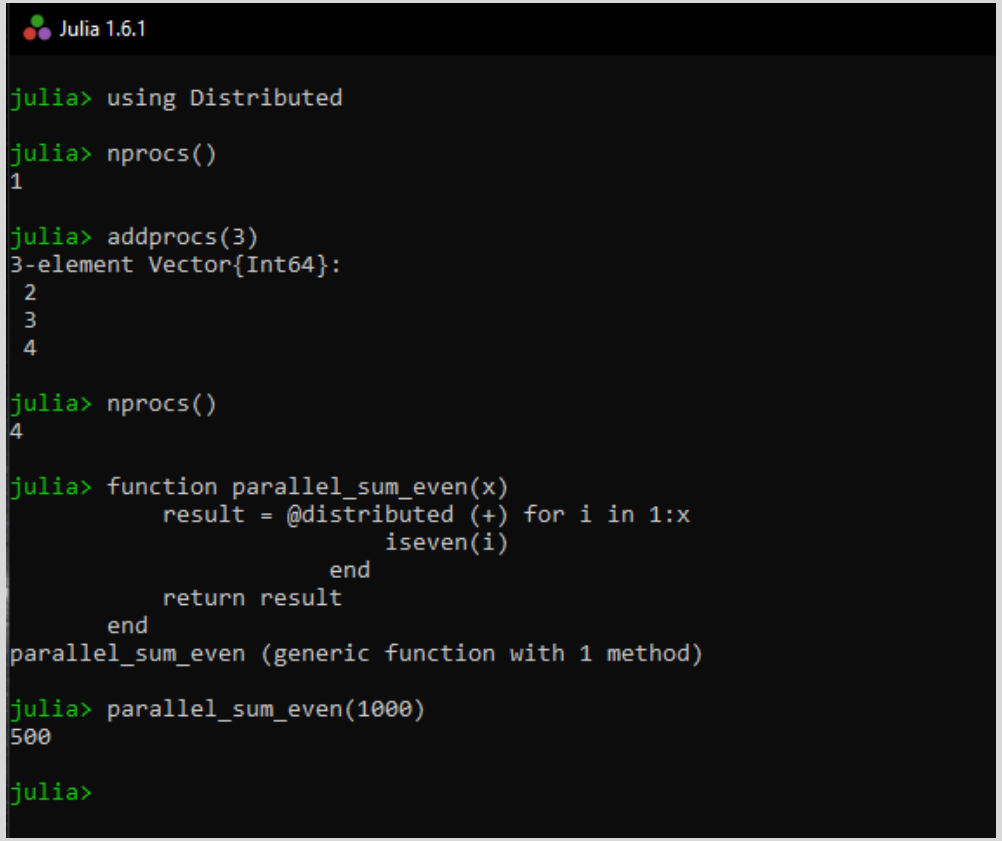
- Plots.jl
  - Efficient go to
  - The documentation is terrible ☹
- PyPlot.jl
  - Simply maps to matplotlib.pyplot
  - Imports all the all the pyplot functions
- Makie.jl
  - Native to Julia
  - Good documentation
  - Can be slow sometimes
- Others
  - GadFly.jl (pretty good 2D library)
  - PGFPlots.jl (plots for LaTeX)
  - UnicodePlots.jl (plot directly in the REPL)





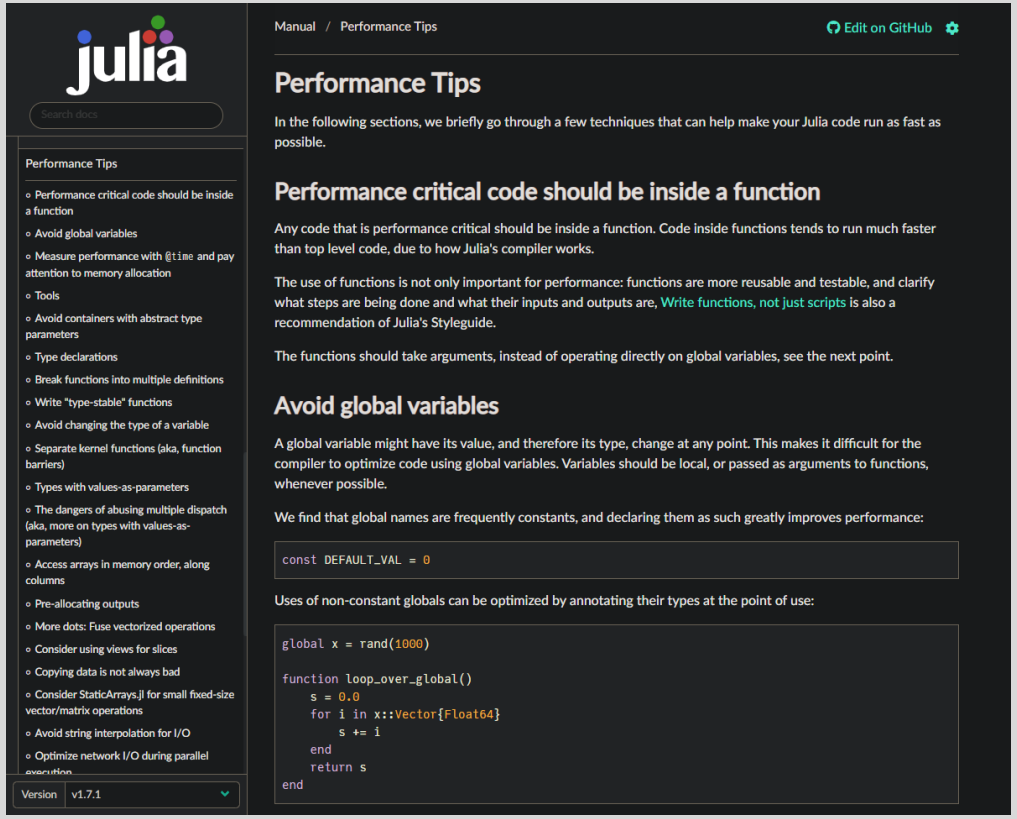
Basic Parallel Computing

- Distributed.jl
  - Provides the basic CPU parallel computing methods
  - Add parallel resources via `addprocs`
  - Parallel for loops using `@distributed`
  - Other packages like MPI.jl take this further
  - To use arrays look at DistributedArrays.jl
  - Good for "larger jobs" (increased memory)
- Multi-threading via Threads
  - Operate on for loops with `Threads.@threads`
  - Must start Julia with the # of threads `$ Julia -threads 4`
  - Use cautiously (shared memory)
- GPU computing
  - Run Julia code on GPUs
  - See juliagpu.org for info



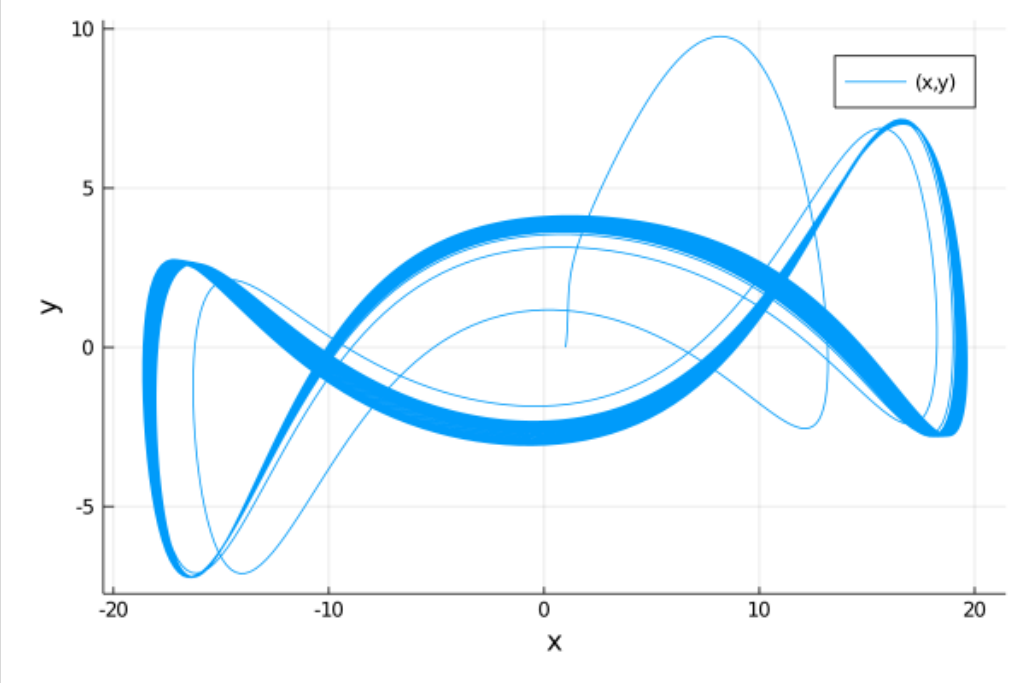
Performance

- The performance tips page
  - https://docs.julialang.org/en/v1/manual/performance-tips/
- Julia code *can be* C-fast
- Can be slow if we have poorly structured code
- Common practices
  - Preallocate instead of appending to arrays
  - Access arrays in memory order
  - Avoid ambiguous types
  - Use function overloading



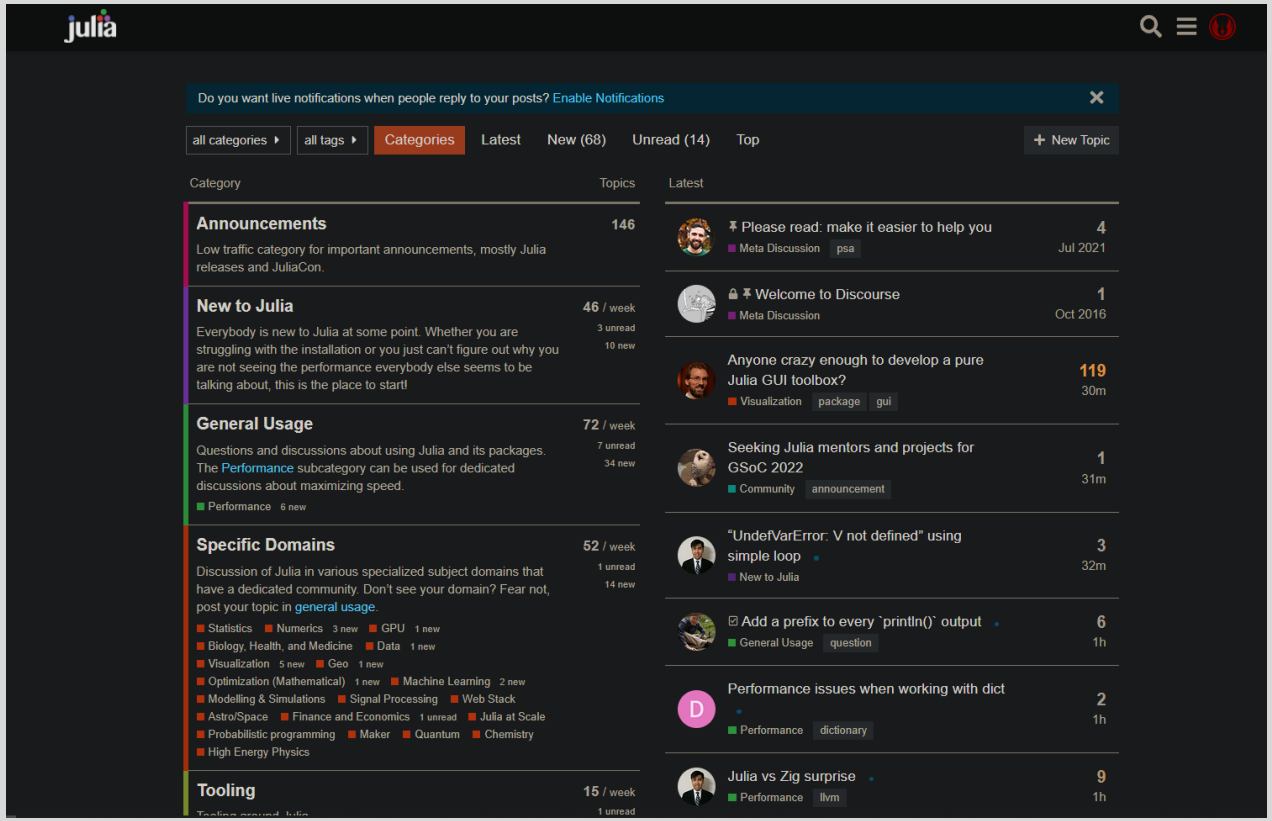
Other Useful Things

- File reading and writing
  - Delimited files via DelimitedFiles (built-in)
    - Text files and basic CSVs
  - CSVs via CSV.jl
  - Store/read dictionaries via JSON.jl
  - Tables via DataFrames.jl
- Math stuff
  - Symbolic math with Symbolics.jl and ModelingToolkit.jl
  - Differential equations with DIfferentialEquations.jl
  - Statistics with Statistics.jl and Distributions.jl
- Machine learning
  - Flux.jl
  - Tensorflow.jl and ScikitLearn.jl
  - Knet.jl



Resources

- Learn Julia in Y Minutes
  - https://learnxinyminutes.com/docs/julia/
- Julia's documentation
  - https://docs.julialang.org/en/v1/
- Julia's Discourse forum
  - https://discourse.julialang.org/
- Julia's YouTube channel
  - https://www.youtube.com/user/julialanguage
  - Check out the tutorials and JuliaCon talks
- Google it!



15 Minute Break Time

JUMP

A Brief Introduction

Modeling in JuMP.jl

$$\begin{aligned} \max_{f,s} \quad & 12f + 9s \\ \text{s.t.} \quad & 4f + 2s \leq 4800 \\ & f + s \leq 1750 \\ & 0 \leq f \leq 1000 \\ & 0 \leq s \leq 1500 \end{aligned}$$

- Initialize **model**

```
1  using JuMP, Gurobi
2  model = Model(Gurobi.Optimizer)
```

- Define **variables**

```
3  @variable(model, f)
4  @variable(model, s)
```

- Define **objective**

```
5  @objective(model, Max, 12f + 9s)
```

- Define **constraints**

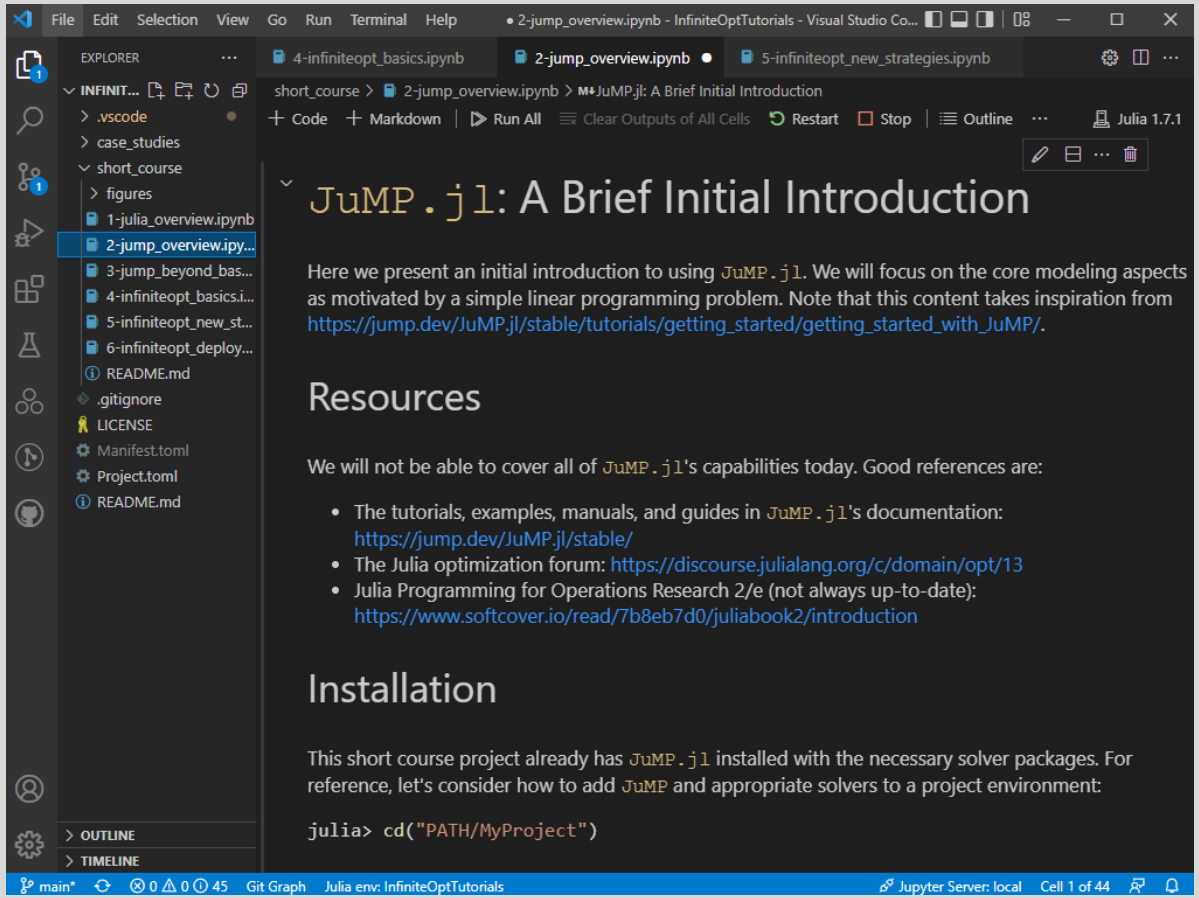
```
6  @constraint(model, 4f + 2s <= 4800)
7  @constraint(model, f + s <= 1750)
8  @constraint(model, 0 <= f <= 1000)
9  @constraint(model, 0 <= s <= 1500)
```

- **Solve**

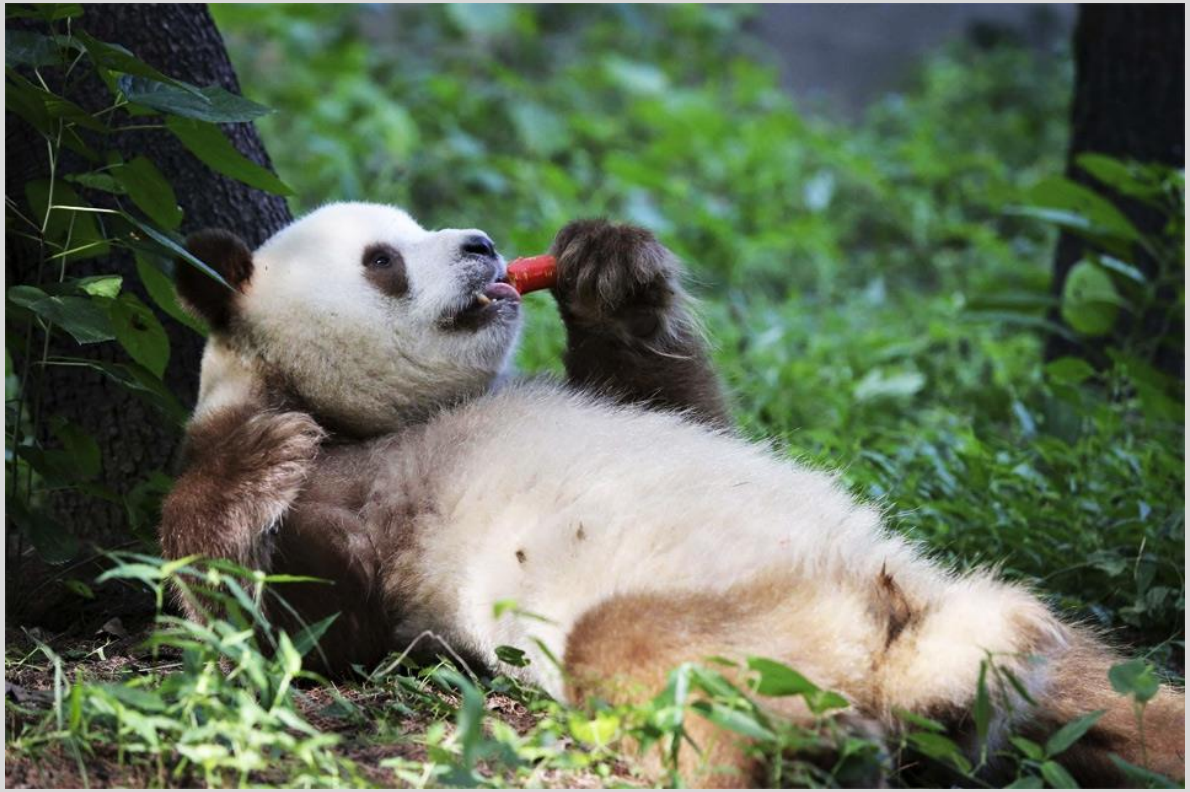
```
10 optimize!(model)
11 profit = objective_value(model)
12 f_best = value(f)
13 s_best = value(s)
```

To the Tutorial!

- Open the "2-jump_overview.ipynb" jupyter notebook



1 Hour Lunch Break

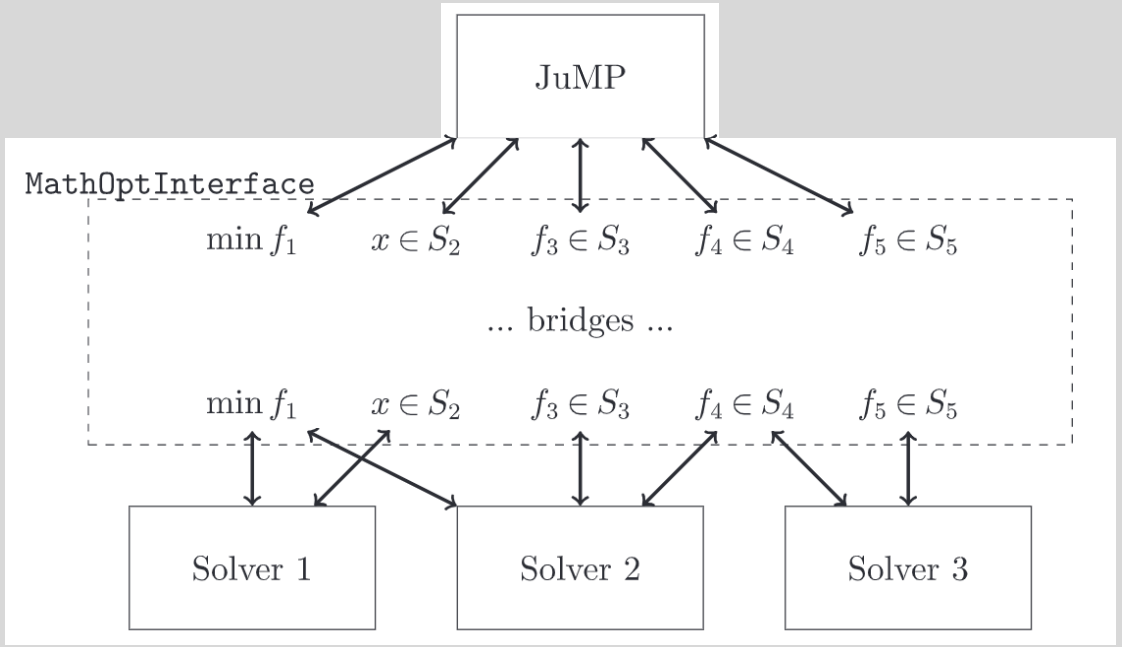


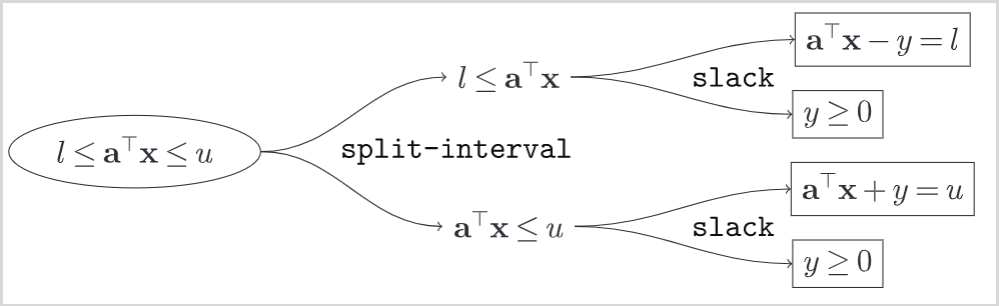
JUMP

Beyond the basics

JuMP.jl Architecture

- JuMP.jl provides the high-level interface (the "macro sugar")
- Model is stored in MathOptInterface.jl model
  - Constraints are of the form *function* in *set*
- Bridges convert constraints into form that solvers support
  - Solvers don't explicitly support every *function-set* combination
- MathOptInferface.jl preserves the constraint mappings
- Solvers interface with MathOptInterface.jl

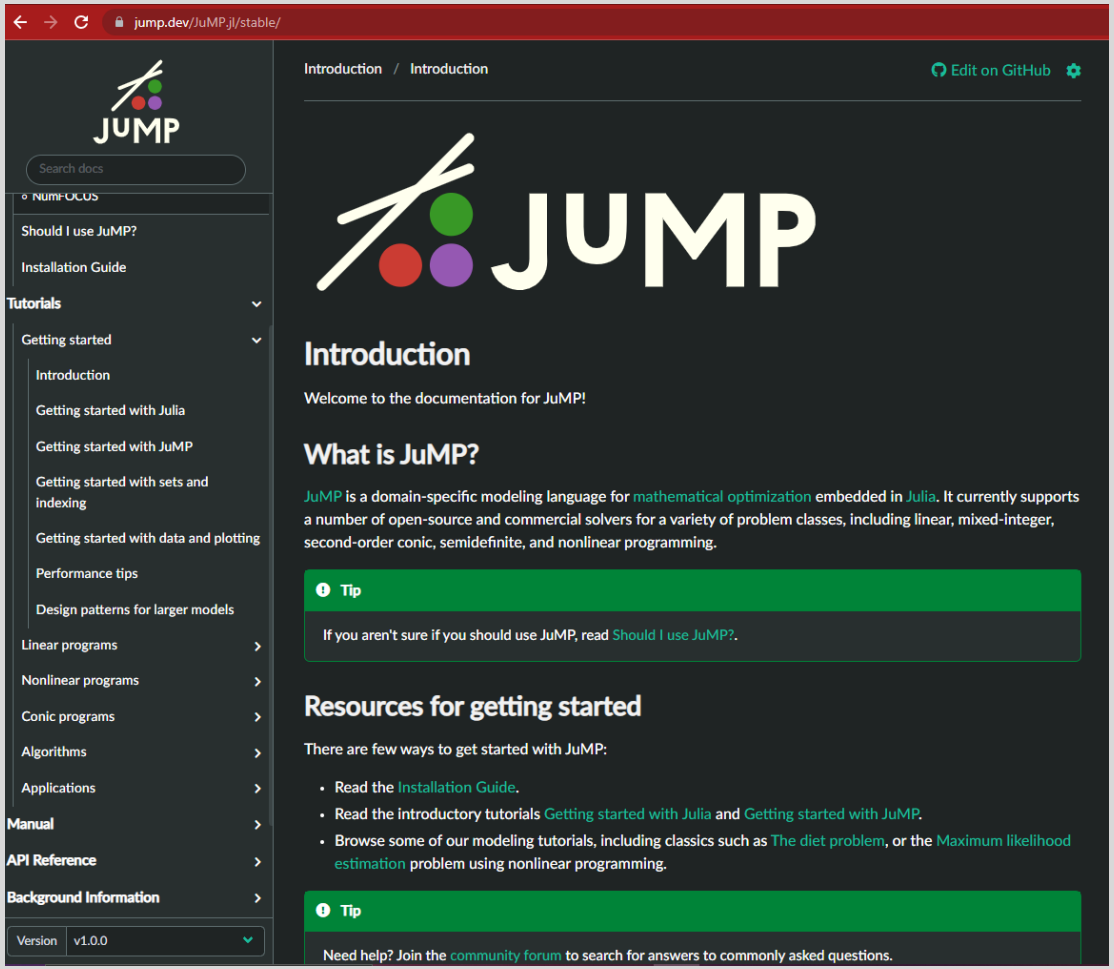




Legat, Benoît, et al. "MathOptInterface: a data structure for mathematical optimization problems." *INFORMS Journal on Computing* 34.2 (2022): 672-689.

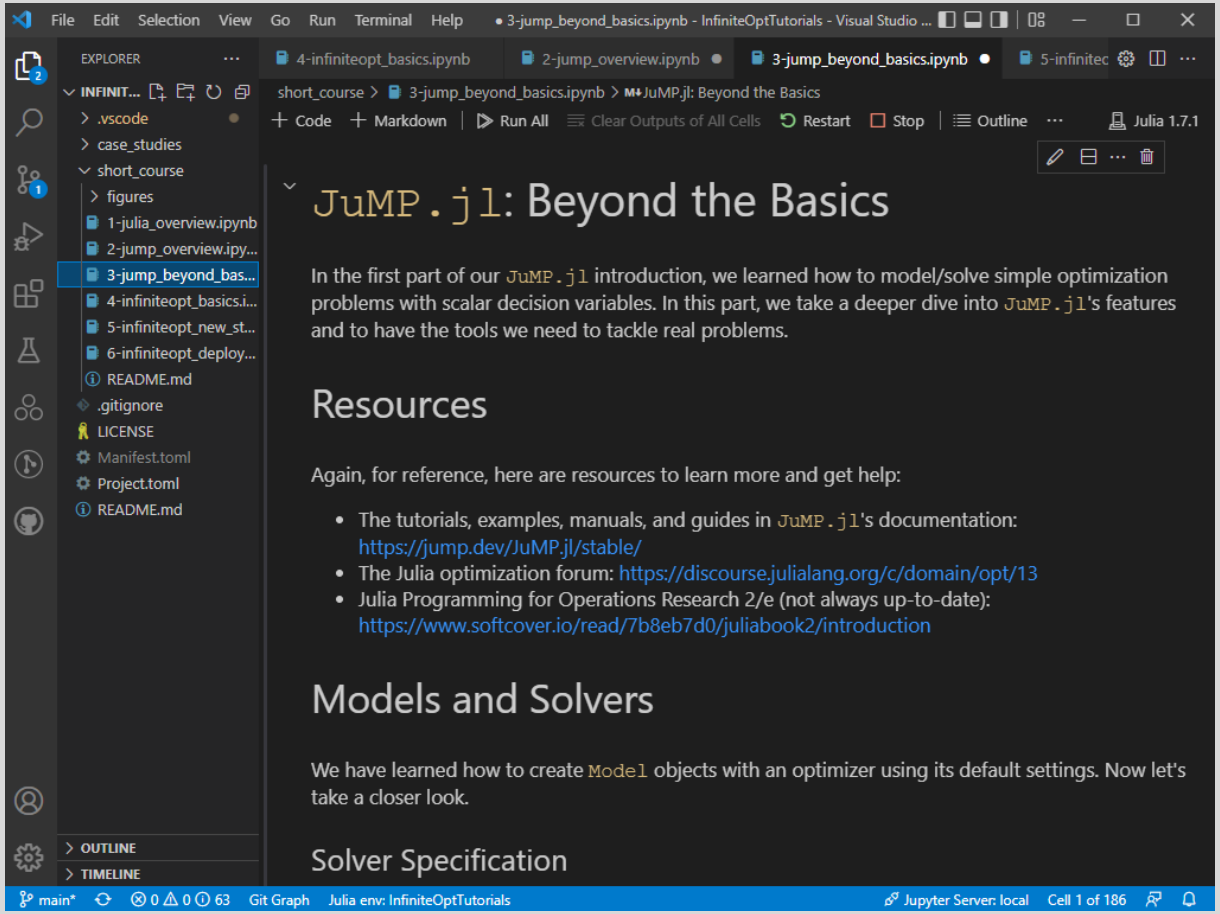
Learning JuMP.jl

- The documentation is full of very useful tutorials

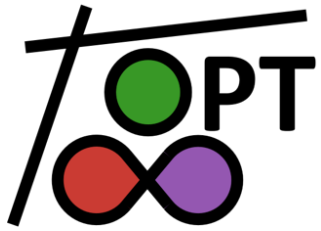


To the Tutorial!

- Open the "3-jump_beyond_basics.ipynb" jupyter notebook



15 Minute Break Time



InfiniteOpt

The Basics

Modeling in InfiniteOpt.jl

- Initialize the **model**

```
using InfiniteOpt, Distributions, Ipopt
m = InfiniteModel(Ipopt.Optimizer)
```

- Add the **infinite parameters**

$$t \in [t_0, t_f] \qquad \xi \sim \mathcal{N}(\mu, \Sigma)$$

```
@infinite_parameter(m, t ∈ [t0, tf], num_supports = 10)
@infinite_parameter(m, ξ[1:10] ~ MvNormal(μ, Σ))
```

- Add **variables** and their domain constraints

$$y_a(t) \in \mathbb{R}_+ \quad y_b(t,\xi) \in \mathbb{R}_+ \quad y_c(\xi) \in \{0,1\} \quad z \in \mathbb{Z}^2$$

```
@variable(m, ya ≥ 0, Infinite(t))
@variable(m, yb ≥ 0, Infinite(t, ξ))
@variable(m, yc, Infinite(ξ), Bin)
@variable(m, z[1:2], Int)
```

- Define the **objective**

$$\min_{y_a(t), y_b(t,\xi), y_c(\xi), z} \int_{t \in \mathcal{D}_t} y_a(t)^2 + 2\mathbb{E}_\xi[y_b(t,\xi)]dt$$

```
@objective(m, Min, ∫(ya^2 + 2 * 𝔼(yb, ξ), t))
```

- Add the **constraints**

$$\frac{\partial y_b(t,\xi)}{\partial t} = 2y_b(t,\xi)^2 + y_a(t) - z_1, \quad \forall t \in \mathcal{D}_t,\ \xi \in \mathcal{D}_\xi$$

$$\mathbb{P}\left(y_b(t,\xi) \le 0\right) \ge \alpha, \quad \forall t \in \mathcal{D}_t$$

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

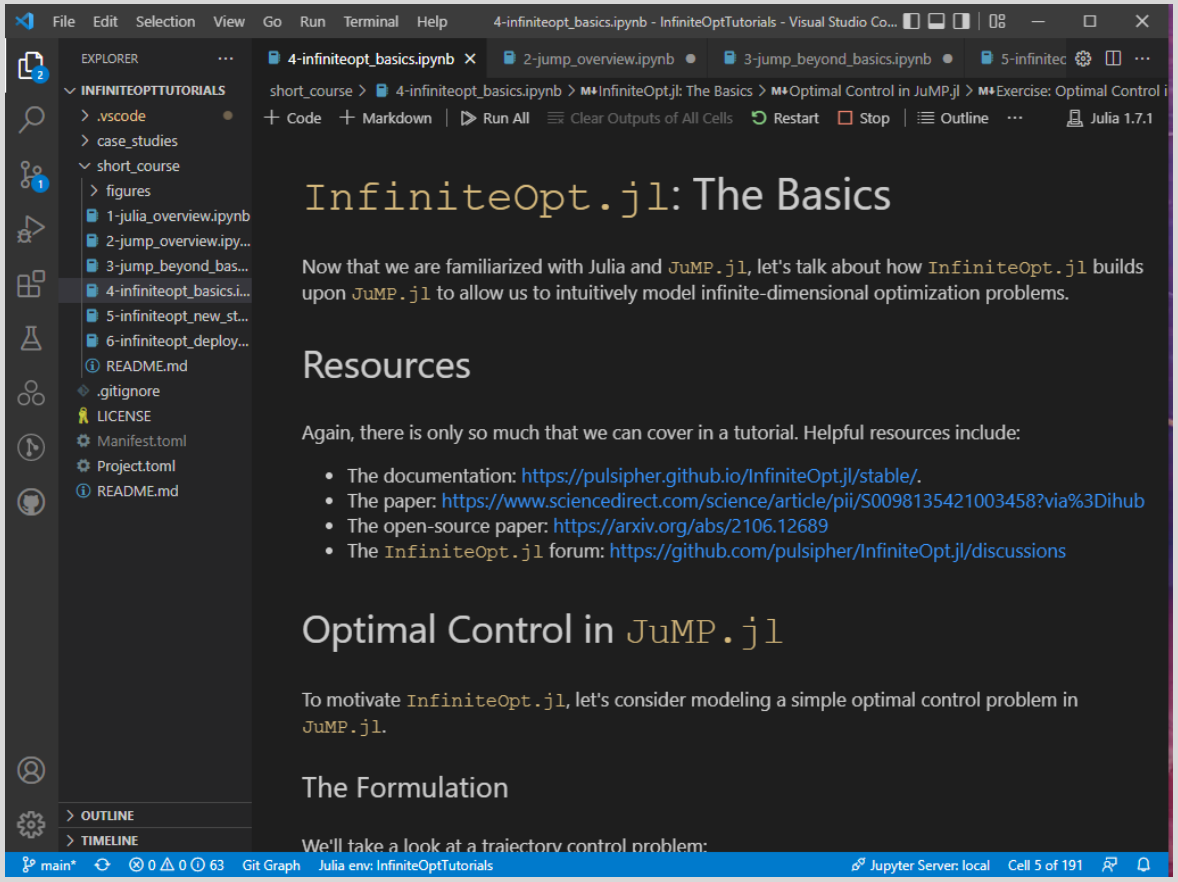
```
@constraint(m, ∂(yb, t) == 2yb^2 + ya - z[1])
@constraint(m, yb ≤ yc * U)
@constraint(m, 𝔼(yc, ξ) ≥ α)
@constraint(m, ya(0) + z[2] == β)
```

- **Solve**

```
optimize!(m)
opt_objective = objective_value(m)
```

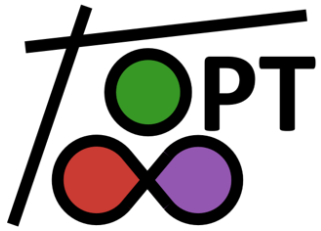
To the Tutorial!

- Open the "4-infiniteopt_basics.ipynb" jupyter notebook



15 Minute Break Time

Why is InfiniteOpt.jl so awesome?

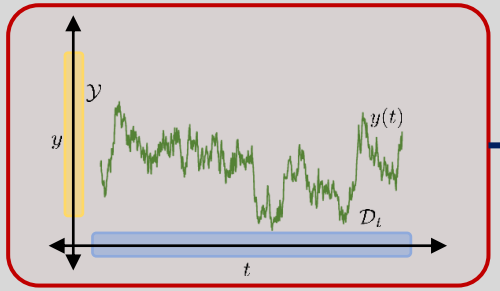


InfiniteOpt

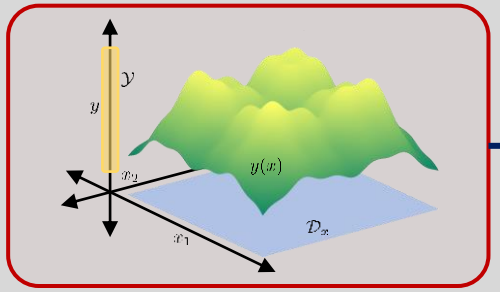
New Modeling Strategies

Innovating with InfiniteOpt.jl

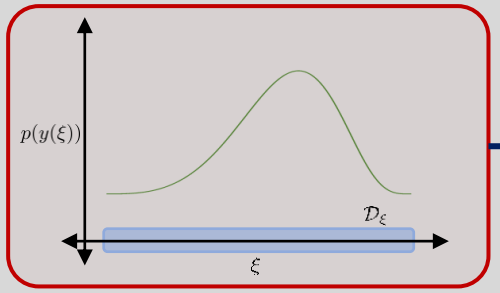
Traditional Formulations



Dynamic Optimization



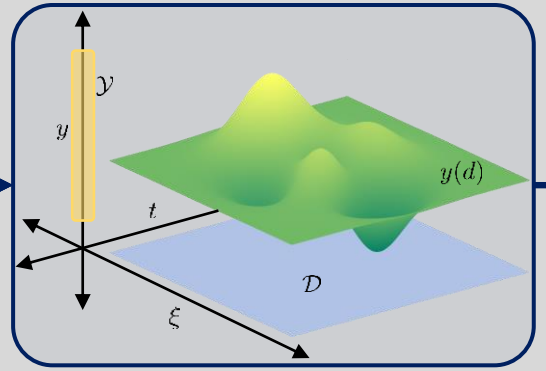
PDE-Constrained Optimization



Stochastic Optimization

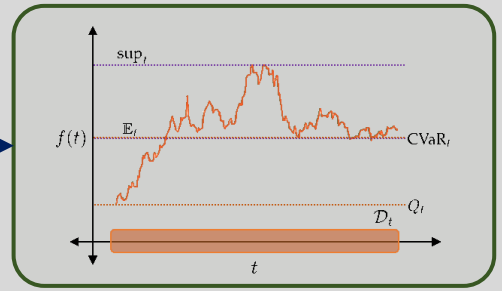
InfiniteOpt

Unifying Abstraction

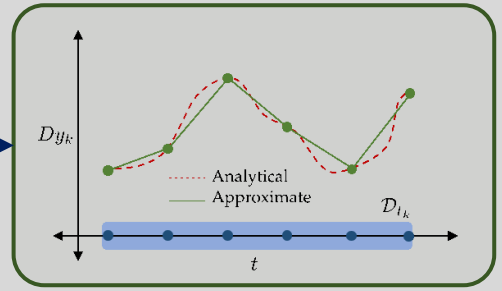


Infinite-Dimensional Optimization

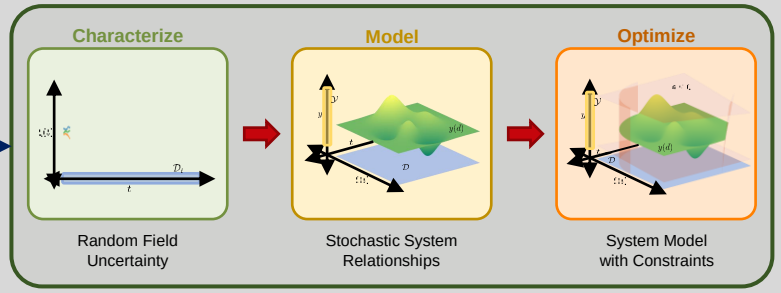
New Formulations



Generalized Shaping Measures



Continuous Time Estimation



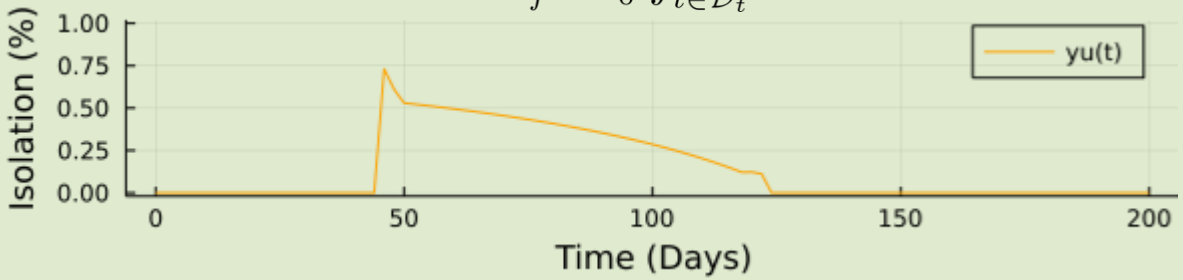
Random Field Optimization

Measure Comparison

Expectation (Uniform pdf)

- Weight the cost **uniformly**

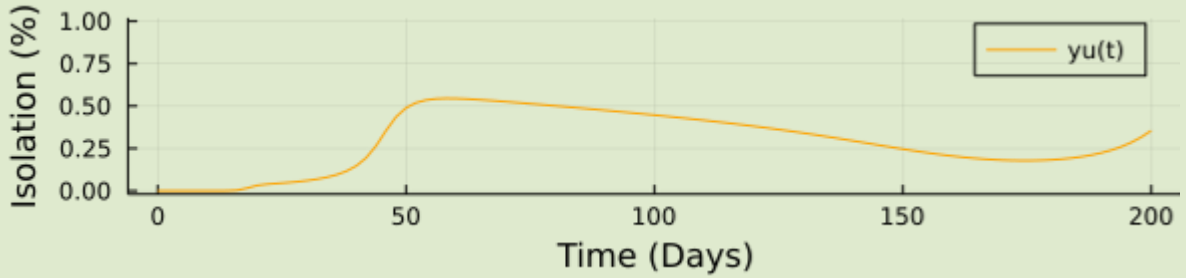
$$\mathbb{E}_t[y_u(t)] = \frac{1}{t_f - t_0} \int_{t \in \mathcal{D}_t} y_u(t) dt$$



Expectation (Exponential pdf)

- Place emphasis on the **initial times**

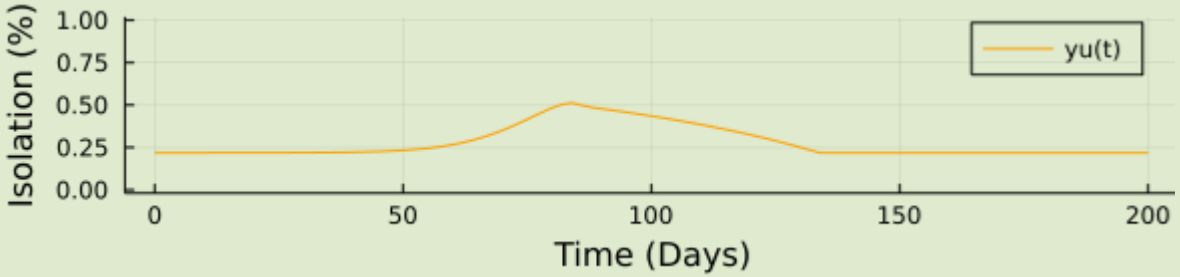
$$\mathbb{E}_t[y_u(t)] = \int_{t \in \mathcal{D}_t} y_u(t) e^{-t} dt$$



Mean-Variance

- Penalize **fluctuations** in the trajectory

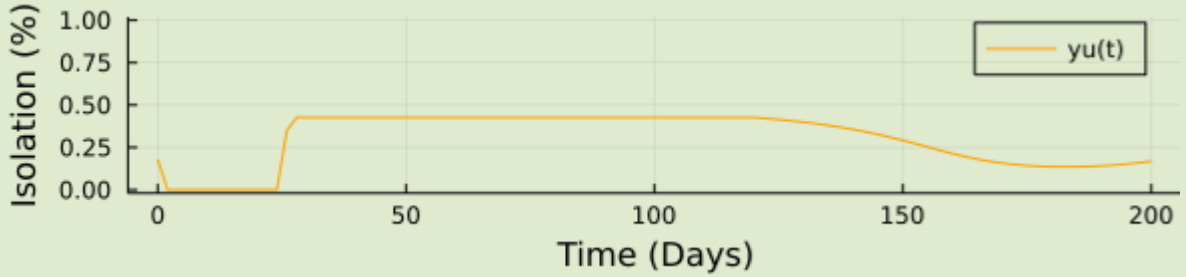
$$\mathbb{E}\text{-}\mathbb{V}_t = \mathbb{E}_t[y_u(t)] + \lambda \mathbb{V}_t[y_u(t)]$$



CVaR

- Only penalize the **high cost peaks**

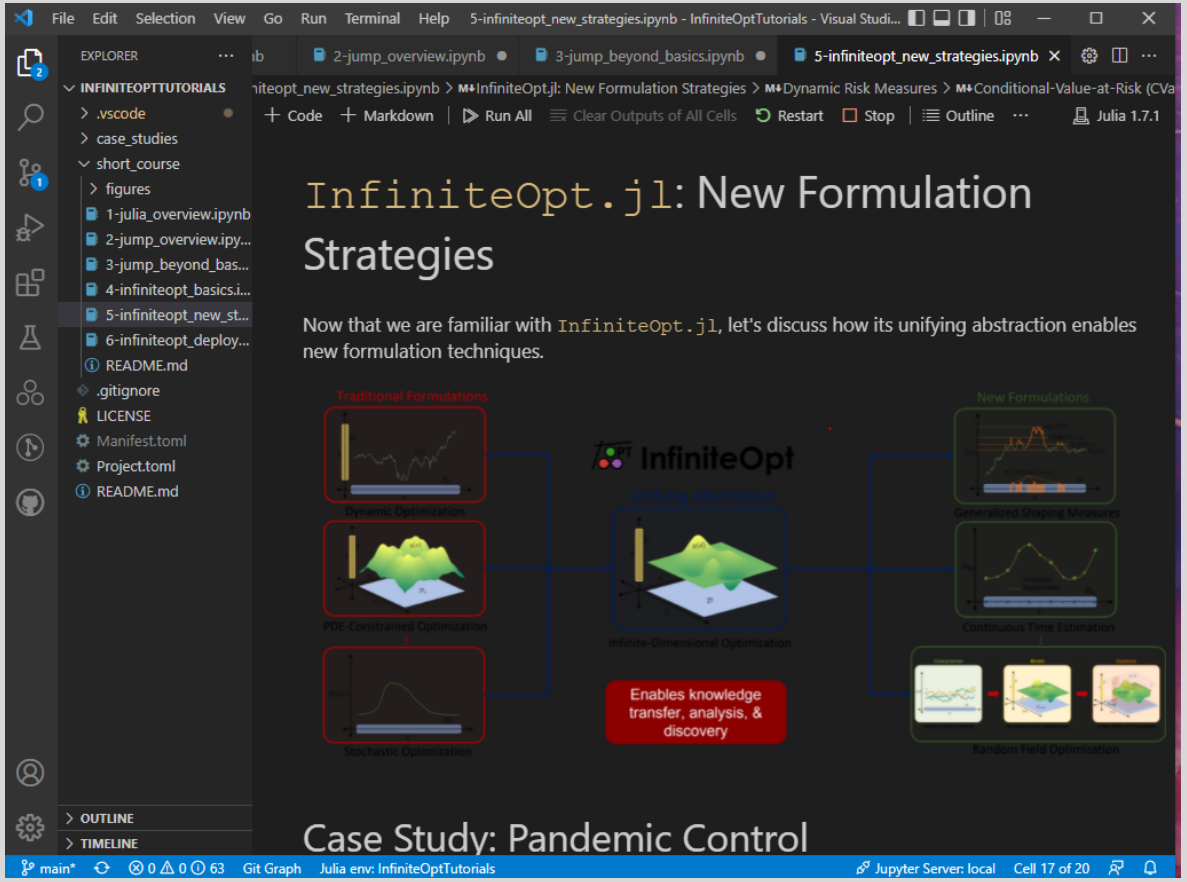
$$\mathrm{CVaR}_t(y_u(t); \alpha)$$

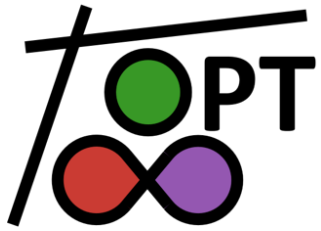


Pulsipher, J.L., Davidson, B.R., and Zavala, V.M. *New Measures for Shaping Trajectories in Dynamic Optimization*. Under Review. 2021

To the Tutorial!

- Open the "5-infiniteopt_new_strategies.ipynb" jupyter notebook





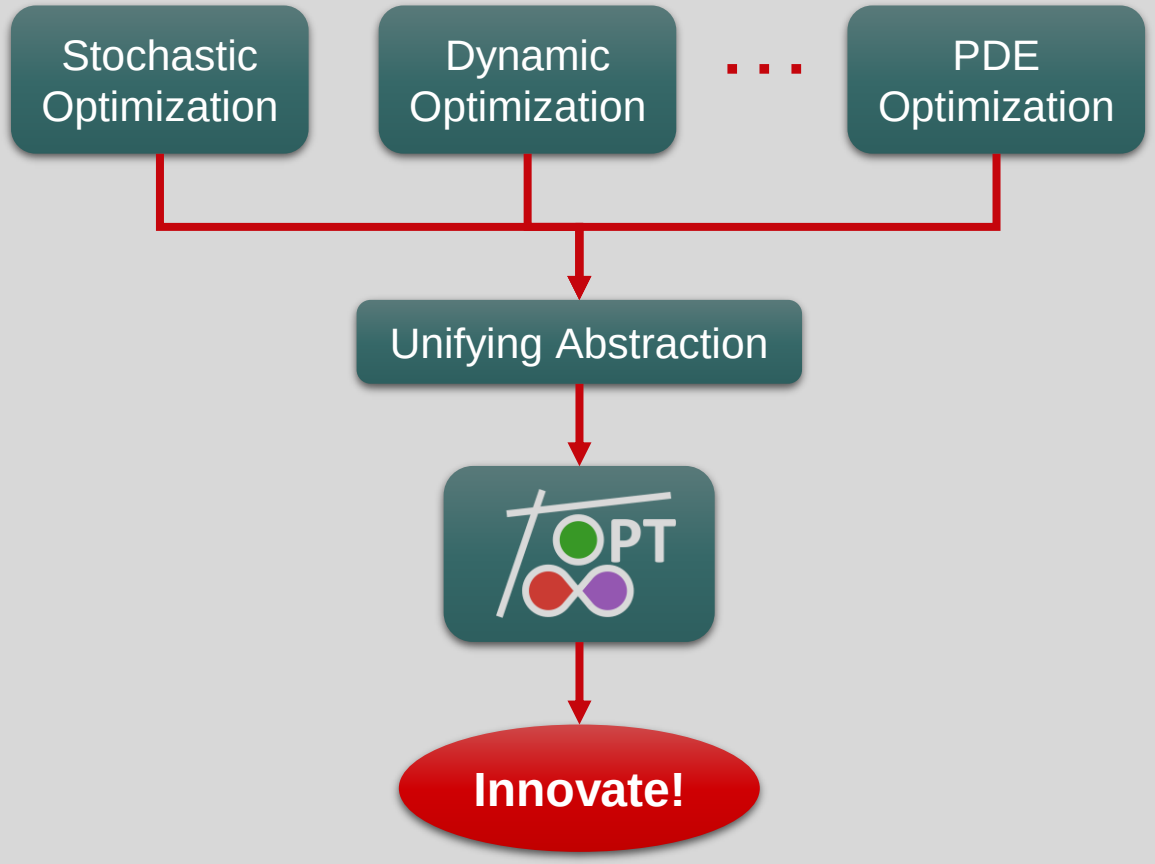
InfiniteOpt

Deployment Tools

Want to deploy your cool techniques?

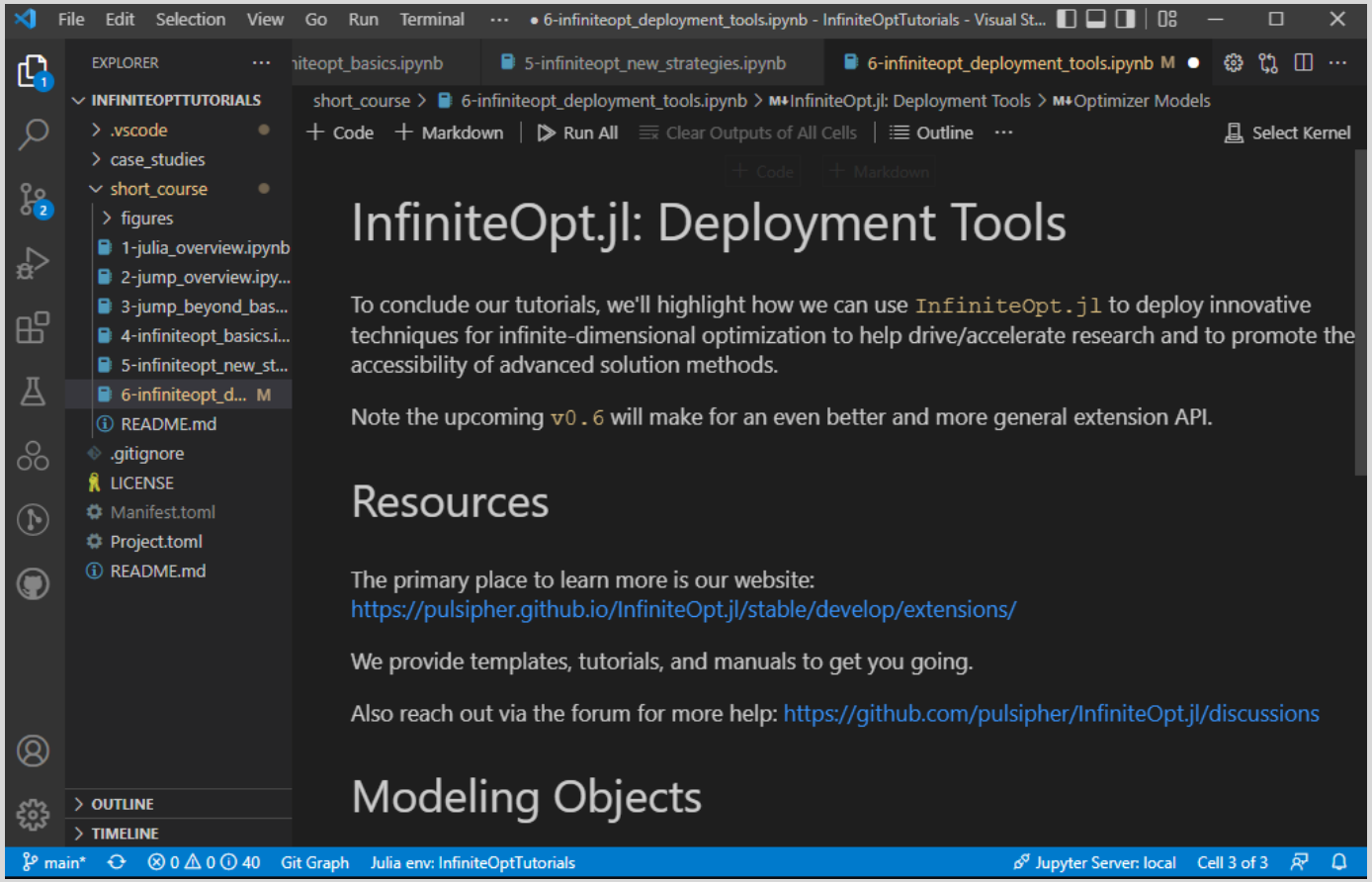
- Almost all modeling aspects can be extended
  - **Infinite domain**
  - **Support generation**
  - **Measure** representation/evaluation
  - **Derivative** evaluation methods
  - Model **transformation**
  - More!

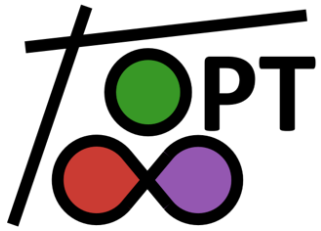
- Extensive **documentation**, **tutorials**, & **templates**



To the Tutorial!

- Open the "6-infiniteopt_deployment_tools.ipynb" jupyter notebook



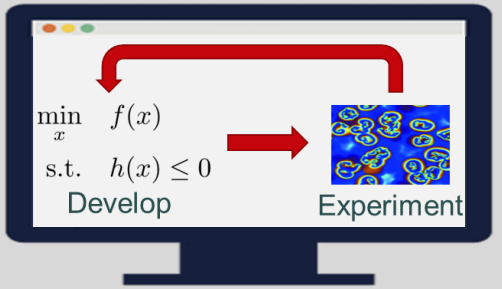


InfiniteOpt

Final Thoughts

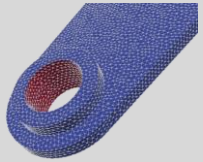
When should I use InfiniteOpt.jl?

- Research
  - Sandbox InfiniteOpt problem approaches
  - Deploy your innovations → accessibility
  - Rapid comparisons
- Learning
  - High-level interface for novices
  - Features to interrogate and learn
- Acceleration
  - Solve problems faster on CPU/GPU via InfiniteExaModels.jl
- Application
  - Offline/online decision making (e.g., optimal control)
  - Any InfiniteOpt problem class

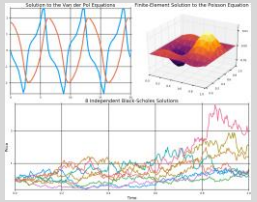


Future Development: Modeling Objects

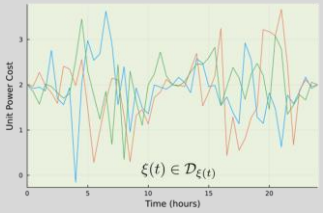
- Non-rectangular domains
  - Leverage packages like Gridap.jl



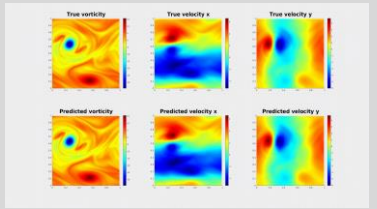
- Optimal Control Toolbox
  - Automatic initialization via DifferentialEquations.jl
  - Simulation



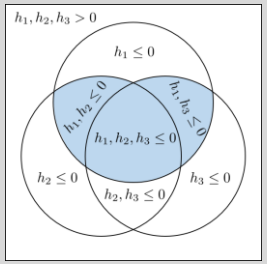
- Infinite Parameter Functions
  - Embed random fields uncertainty



- Infinite-Dimensional Surrogates
  - Embed neural operator models

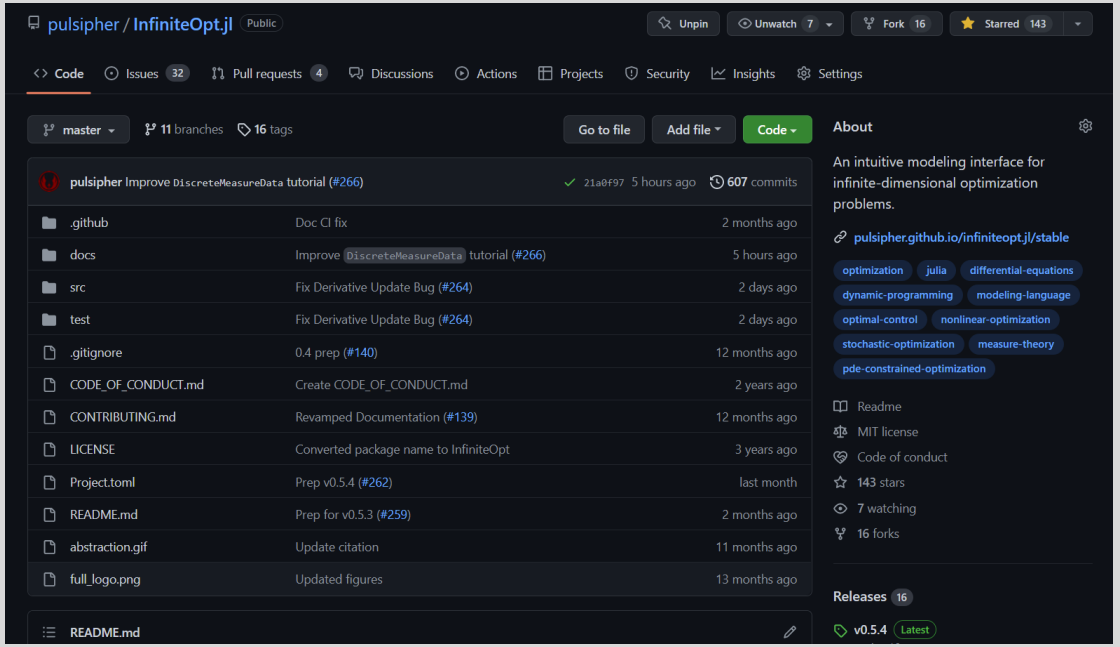


- Event Constraints
  - Generalization of chance constraints
  - Connection to generalized disjunctive programming



How can I help?

- Get started with our tutorials: https://pulsipher.github.io/InfiniteOpt.jl/stable/
- Check out the contribution guide: https://pulsipher.github.io/InfiniteOpt.jl/stable/develop/start_guide/
- Reach out on the forum: https://github.com/pulsipher/InfiniteOpt.jl/discussions
- Give InfiniteOpt.jl a star!



What were those resources, again?

- Julia
  - Julia learning: https://julialang.org/learning/
  - Documentation: https://jump.dev/JuMP.jl/stable/
  - Forum: https://discourse.julialang.org/
  - YouTube: https://www.youtube.com/c/TheJuliaLanguage
- JuMP.jl
  - Documentation: https://jump.dev/JuMP.jl/stable/
  - Forum: https://discourse.julialang.org/c/domain/opt/13
  - Free Textbook: https://www.softcover.io/read/7b8eb7d0/juliabook2/introduction
- InfiniteOpt.jl
  - Documentation: https://pulsipher.github.io/InfiniteOpt.jl/stable/
  - Forum: https://github.com/pulsipher/InfiniteOpt.jl/discussions

That’s all!