InfiniteOpt

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

Dr. Joshua Pulsipher

FAST • DYNAMIC
COMPOSABLE • OPEN SOURCE
the julia programming language

julia

JUMP

Today's Topics

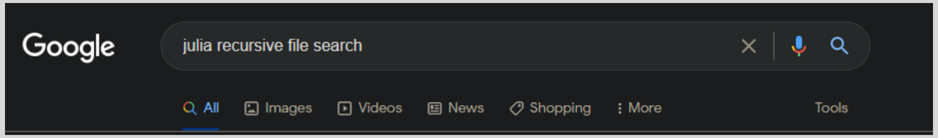
CODING IN JULIA

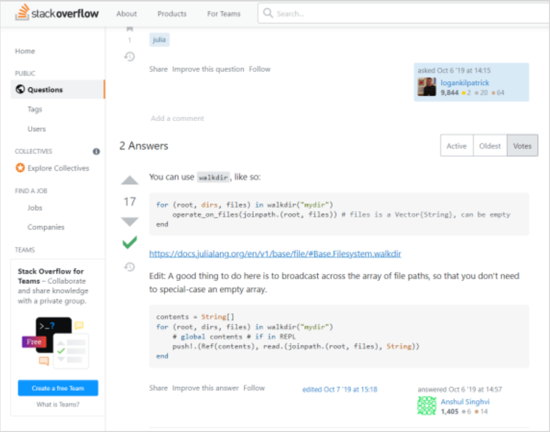
MATHEMATICAL OPTIMIZATION VIA JUMP.JL

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

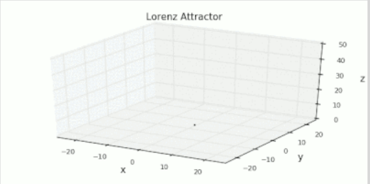
- Parallel Computing
  - Designed for it from the ground up

- Portability
  - Works on any O.S.

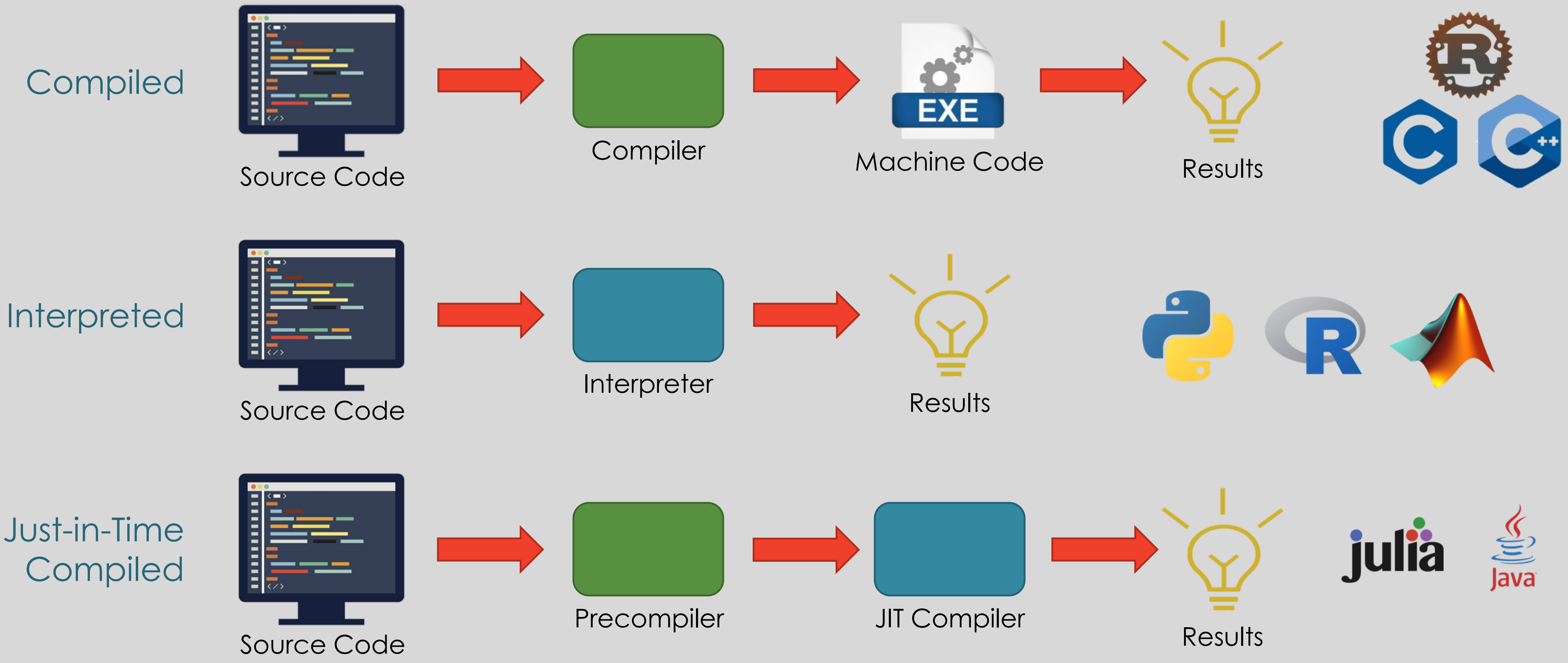
- Generality
  - Can be used for scripts, apps, metaprogramming, etc.

- Open Source
  - It's free!

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



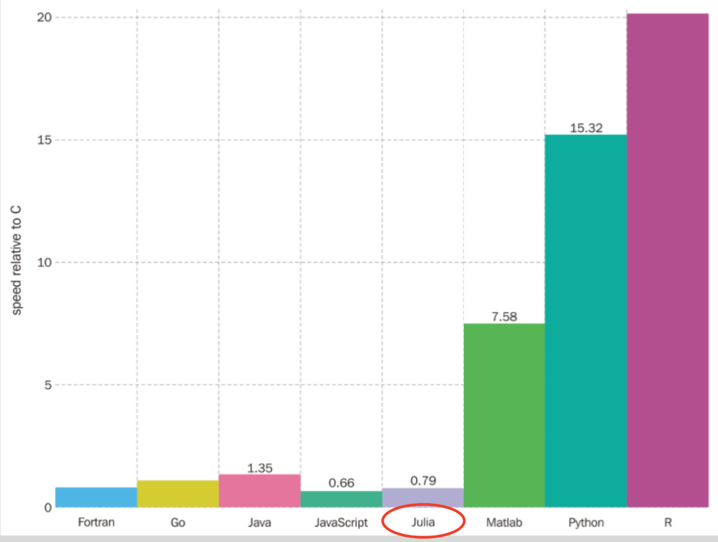
Programming Language Paradigms

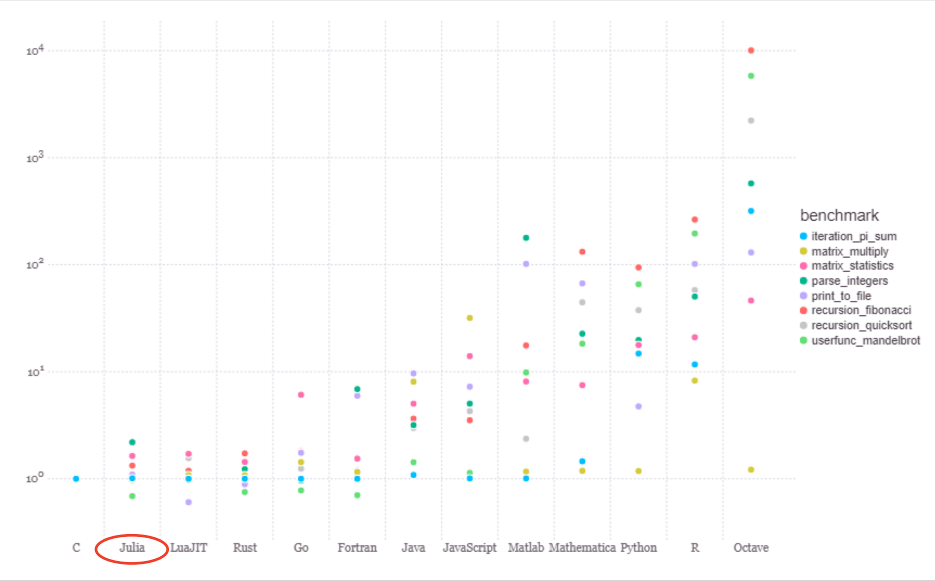


*Over simplified

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*



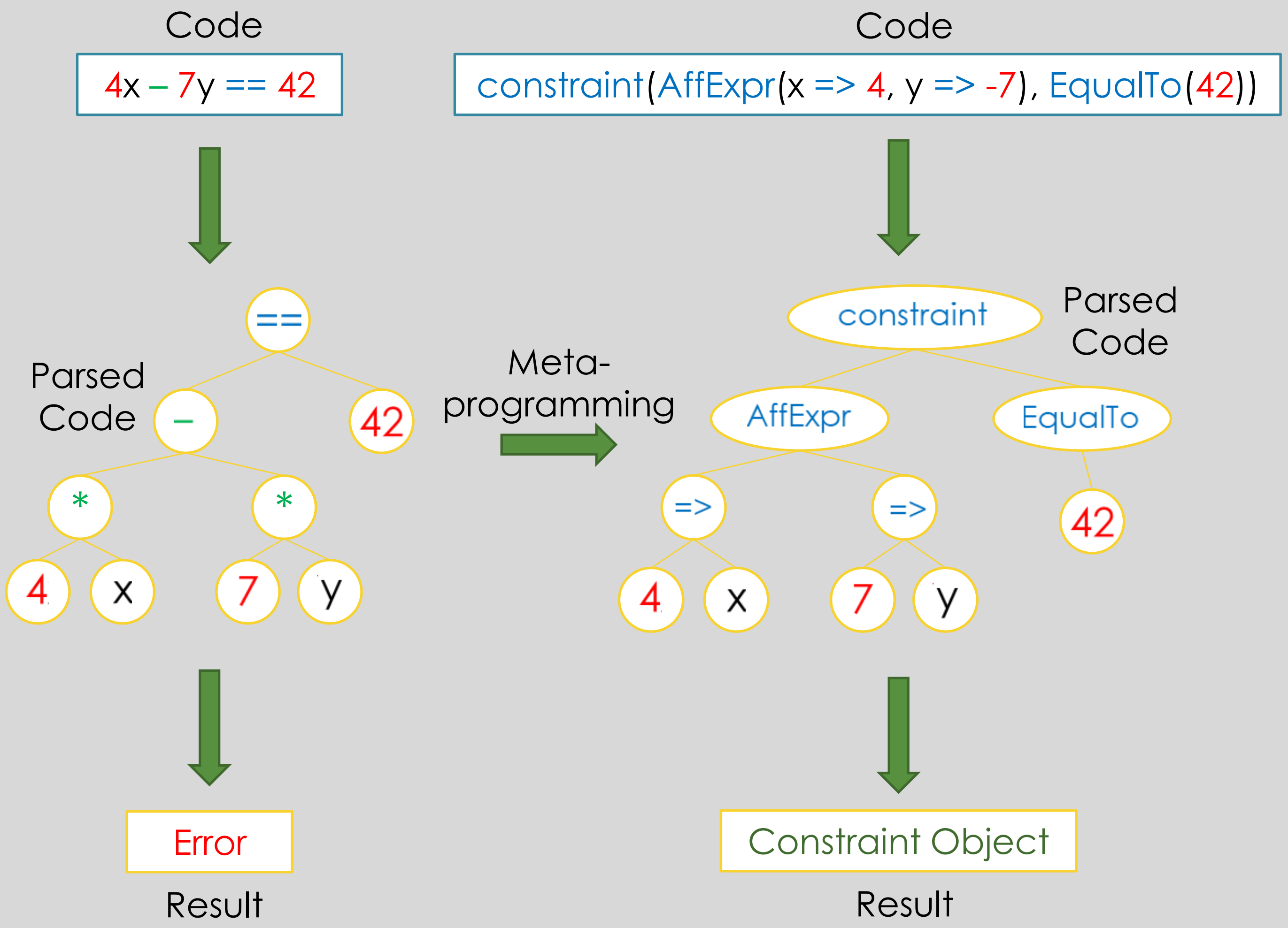


Metaprogramming (Macros)

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

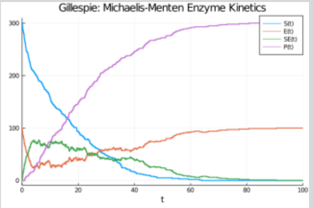
```
@constraint(4x – 7y == 42)
```

```
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)
```

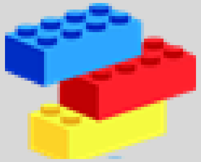
- DataFrames.jl
  - Efficient datasets (like Pandas)
- Flux.jl
  - Efficient/interpretable machine learning
- DistributedArrays.jl (GPU parallelzation)
  - Easy parallelization for CPU cores/threads and GPUs

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)
```

- Extensible
  - Can be extended for advanced techniques

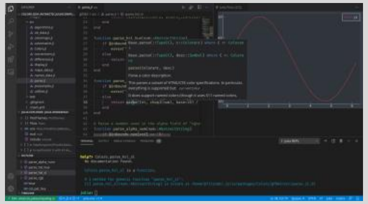


- Solver Library
  - Rapidly try different solvers

GUROBI OPTIMIZATION · COIN|OR · SCIP · IBM CPLEX · KNITRO

- Open Source
  - It's free!

- Programmable
  - Easy embed in a data script



- 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

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.

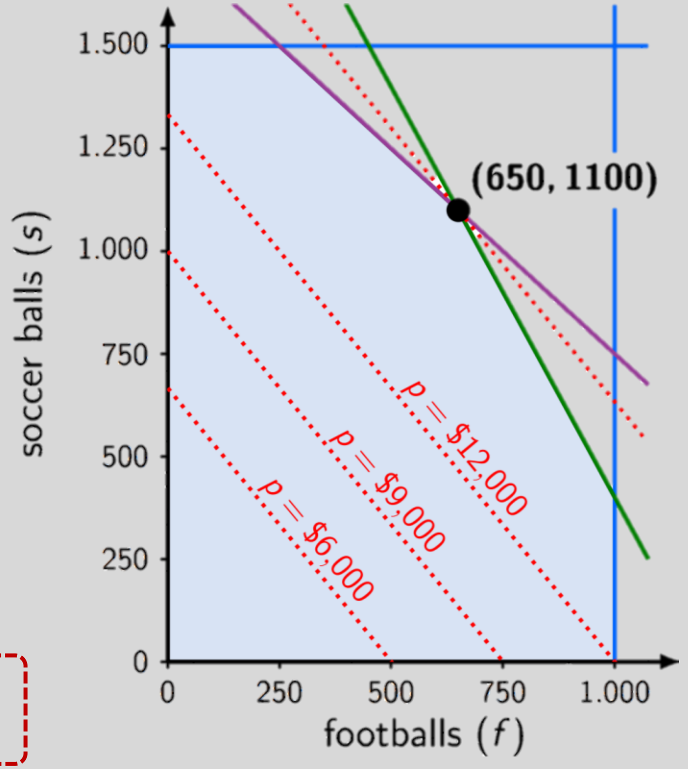
Aspects

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

Sports Store Example

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

Finite # of **decisions**



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**

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

- Define **variables**

```julia
@variable(model, f)
@variable(model, s)
```

- Define **objective**

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

- Define **constraints**

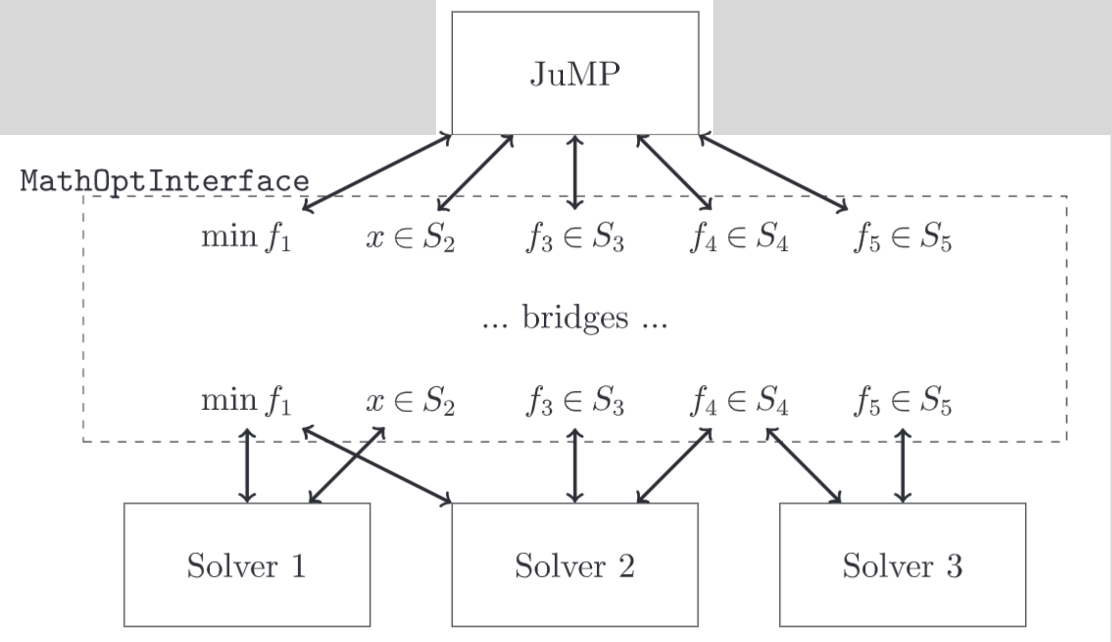
```julia
@constraint(model, 4f + 2s <= 4800)
@constraint(model, f + s <= 1750)
@constraint(model, 0 <= f <= 1000)
@constraint(model, 0 <= s <= 1500)
```

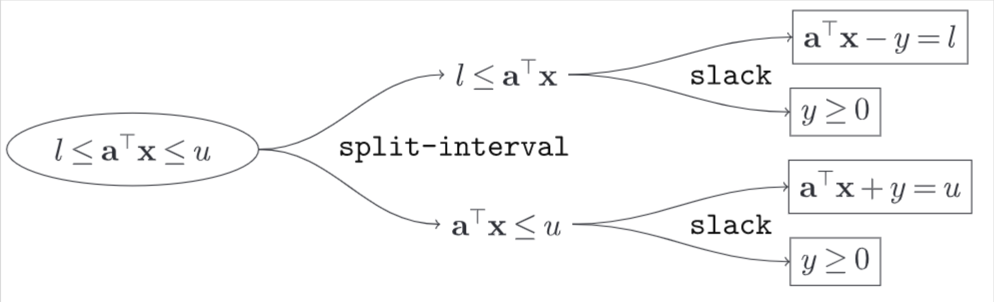
- **Solve**

```julia
optimize!(model)
profit = objective_value(model)
f_best = value(f)
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

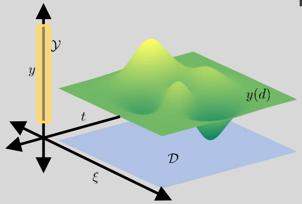




Legat, Benoît, et al. "MathOptInterface: a data structure for mathematical optimization problems." *INFORMS Journal on Computing* 34.2 (2022): 672-689.

Why InfiniteOpt.jl?

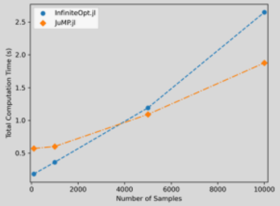
- Unifying Abstraction
  - Captures a wide breadth of problem classes



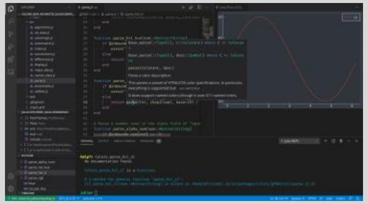
- High-level Syntax
  - Compactly express problems

```
@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

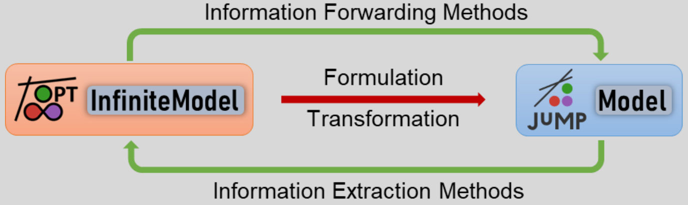


- Built on JuMP.jl
  - Draw from an extensive suite of solvers



- Open Source
  - It's free!

- Enables Accessible Research
  - Implement advanced methods for general users

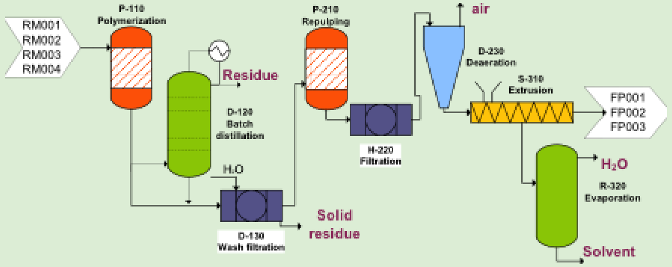


Applications over Infinite Domains

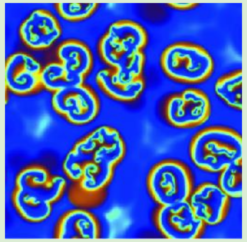
Idea: Many engineering applications are parameterized over **infinite (continuous) domains** (e.g., space & time)

Illustrative Applications

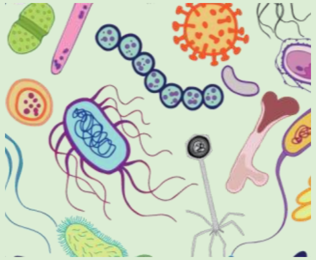
Autonomous Vehicles
(time)



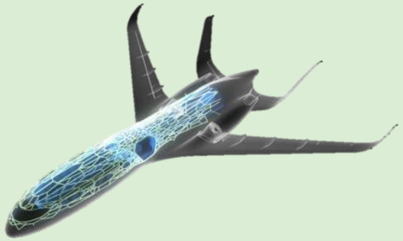
Process Systems
(time)



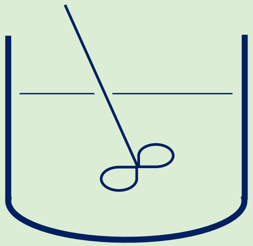
Diffusive Processes
(time & space)



Microbial Communities
(time)

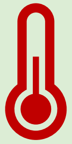
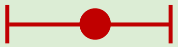


Structural Design
(space)



Reactive Systems
(time)

Uncertainty

- Environment 
- Fitted parameters 
- Forecasting 
- Small length scales 

What is Infinite Optimization?

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

New Aspects

- Infinite Variables

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

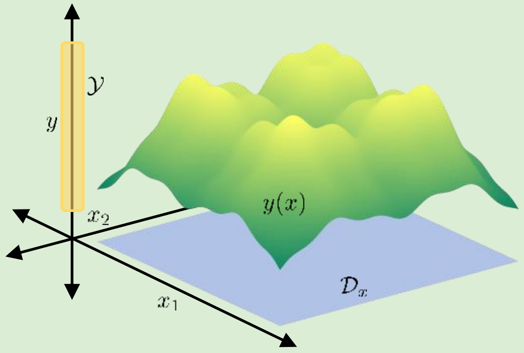
- Measures

$$\mathbb{R}\left[y(\xi)\right] \qquad \int_{[-1,1]^3} \int_0^T y(t, x) dt dx$$

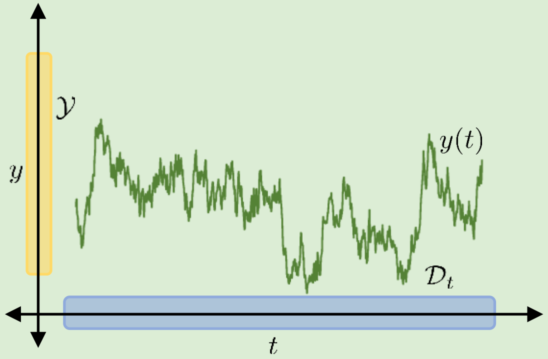
- Derivatives

$$\frac{dy(t)}{dt} \qquad \nabla_x y(t, x)$$

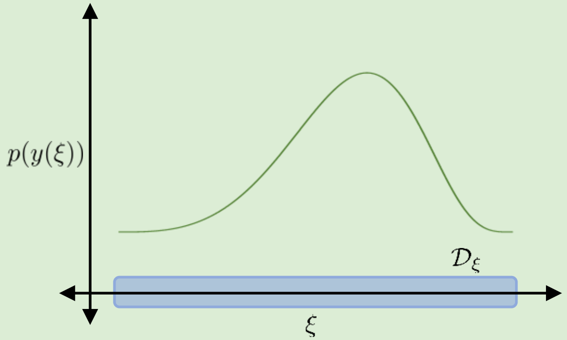
Application Fields



PDE-Constrained



Dynamic



Stochastic

Infinite # of **decisions**

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

Motivating Example: Stochastic Optimal Control

Goal: Determine **optimal social distancing policy** to control spread of a contagion and minimize economic impact

Problem Setup

- **SEIR** disease model

$$\frac{dy_s}{dt} = (y_u - 1)\beta y_s y_i$$
$$\frac{dy_e}{dt} = (1 - y_u)\beta y_s y_i - \xi y_e$$
$$\frac{dy_i}{dt} = \xi y_e - \gamma y_i$$
$$\frac{dy_r}{dt} = \gamma y_i$$

s: susceptible pop.
e : exposed pop.
i : infectious pop.
r: recovered pop.
u: social distancing

- Incubation constant ξ is **uncertain** (statically)

Formulation

$$\begin{aligned}
\min \quad & \int_{t \in \mathcal{D}_t} y_u(t)dt \\
\text{s.t.} \quad & \frac{\partial y_s(t,\xi)}{\partial t} = (y_u(t) - 1)\beta y_s(t,\xi) y_i(t,\xi), && \forall t \in \mathcal{D}_t, \xi \in \mathcal{D}_\xi \\
& \frac{\partial y_e(t,\xi)}{\partial t} = (1 - y_u(t))\beta y_s(t,\xi) y_i(t,\xi) - \xi y_e(t,\xi), && \forall t \in \mathcal{D}_t, \xi \in \mathcal{D}_\xi \\
& \frac{\partial y_i(t,\xi)}{\partial t} = \xi y_e(t,\xi) - \gamma y_i(t,\xi), && \forall t \in \mathcal{D}_t, \xi \in \mathcal{D}_\xi \\
& \frac{\partial y_r(t,\xi)}{\partial t} = \gamma y_i(t,\xi), && \forall t \in \mathcal{D}_t, \xi \in \mathcal{D}_\xi \\
& y_s(0,\xi) = s_0, y_e(0,\xi) = e_0, y_i(0,\xi) = i_0, y_r(0,\xi) = r_0, && \forall \xi \in \mathcal{D}_\xi \\
& y_i(t,\xi) \leq i_{max}, && \forall t \in \mathcal{D}_t, \xi \in \mathcal{D}_\xi \\
& y_u(t) \in [0, \overline{u}] \\
& \xi \sim \mathcal{U}(\underline{\xi}, \overline{\xi})
\end{aligned}$$

Infinite Domains & Parameters

Infinite Domains

Definition

Domains of **infinite cardinality** (e.g., continuous spaces)

Notation

$$\mathcal{D}_\ell \subseteq \mathbb{R}^{n_\ell}$$

where $|\mathcal{D}_\ell| = \infty$

Infinite Parameters

Definition

Parameters **living on infinite domains**.

Notation

$$d_\ell \in \mathcal{D}_\ell$$

Problem Domain

Definition

Domain of infinite-dimensional problem.

Notation

$$\mathcal{D} := \prod_{\ell \in \mathcal{L}} \mathcal{D}_\ell$$

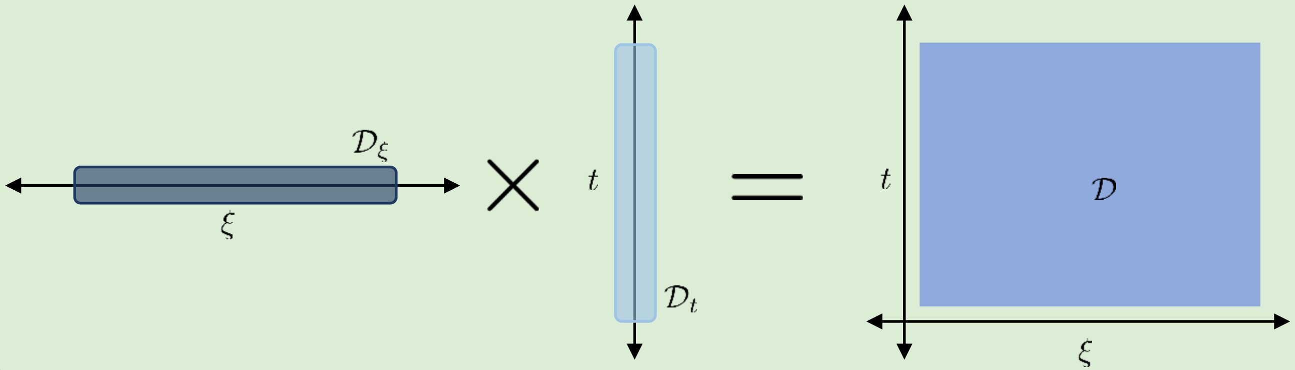
Examples

$$t \in \mathcal{D}_t = [0, 10]$$

$$\xi \in \mathcal{D}_\xi = (-\infty, \infty)^n$$

$$\mathcal{D} = \mathcal{D}_t \times \mathcal{D}_\xi = [0, 10] \times (-\infty, \infty)^n$$

Illustration

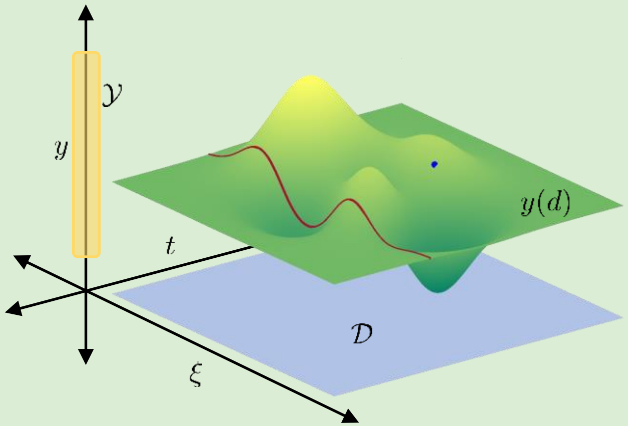


J. L. Pulsipher, W. Zhang, T. J. Hongisto, and V. M. Zavala. A *Unifying Modeling Abstraction for Infinite-Dimensional Optimization*. Computers & Chem. Eng. 2022

Decision Variables

Infinite Variables

Illustration



Definition

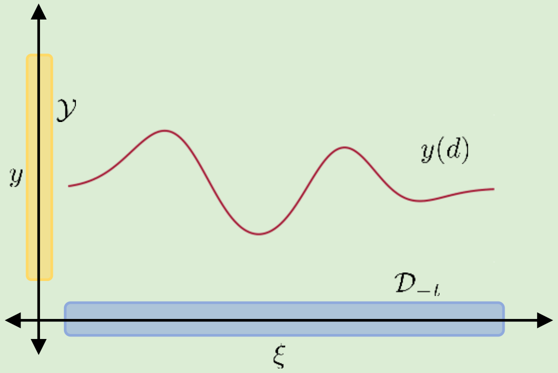
Functions w/ infinite parameter inputs → manifolds to shape

Notation

$$y : \mathcal{D} \mapsto \mathcal{Y} \subseteq \mathbb{R}^{n_y}$$

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

Semi-Infinite Variables

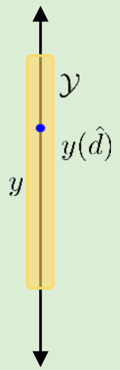


Partially restricted infinite variable

$$y : \mathcal{D}_{-\ell} \mapsto \mathcal{Y} \subseteq \mathbb{R}^{n_y}$$

$$y(0, x, \xi)$$

Point Variables

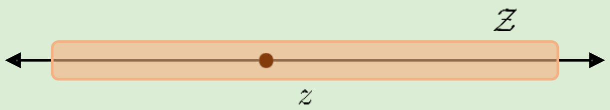


Fully restricted infinite variable

$$y(\hat{d}) \in \mathcal{Y} \subseteq \mathbb{R}^{n_y}$$

$$y(0, 1, 0.43)$$

Finite Variables



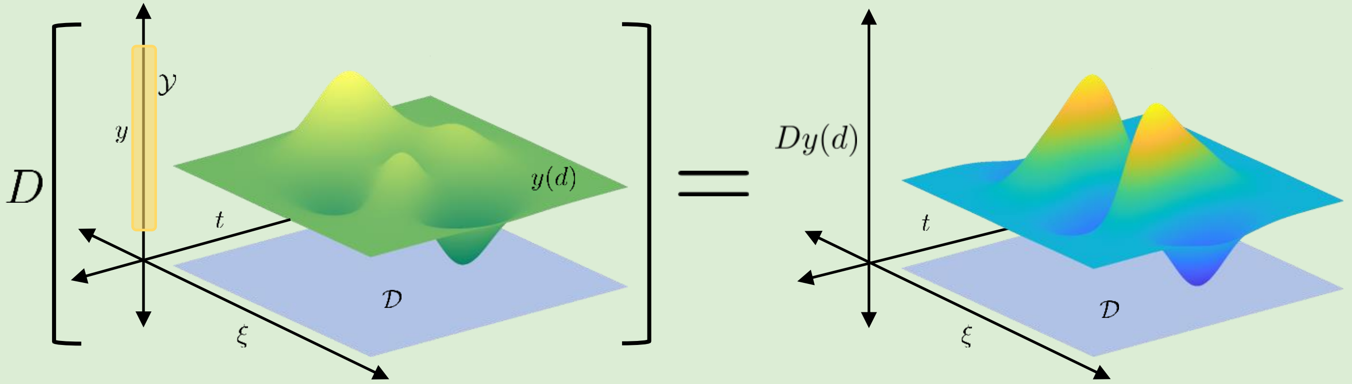
Indexed over finite domains

$$z \in \mathcal{Z} \subseteq \mathbb{R}^{n_z}$$

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

Operators

Differential Operators



Definition

Capture **how a variable changes** over its infinite domain

Notation

$$Dy : \mathcal{D} \mapsto \mathbb{R}^{n_d}$$

$$\frac{dy(t)}{dt} \qquad \nabla_x y(t, x)$$

Measure Operators

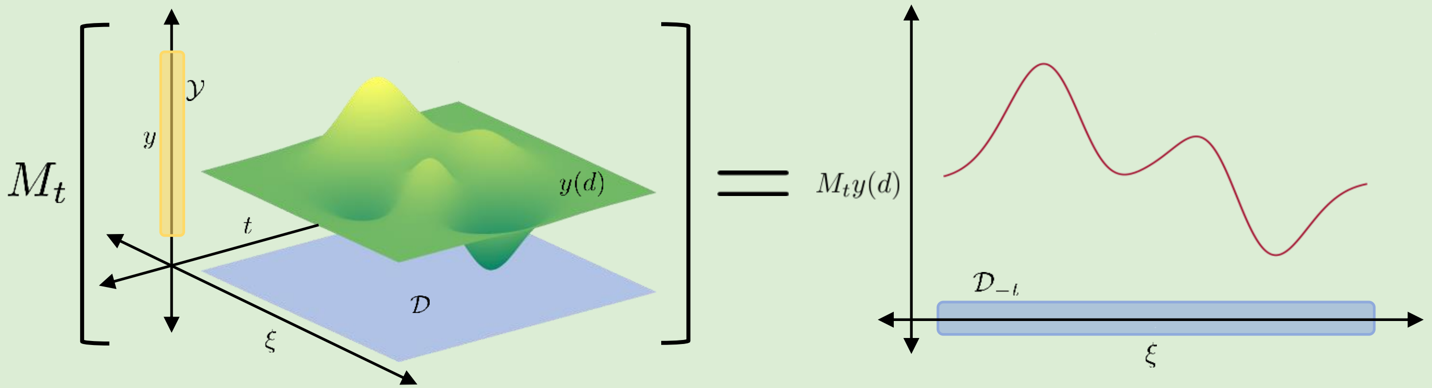
Definition

Summarize/collapse **functions** over an infinite domain

Notation

$$M_\ell y : \mathcal{D}_{-\ell} \mapsto \mathbb{R}^{n_m}$$

$$\int_{(t,x)\in\mathcal{D}_{t,x}} y(t, x) dt dx \qquad \mathbb{R}\left[y(\xi)\right]$$



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

Objectives and Constraints

Objectives

Definition

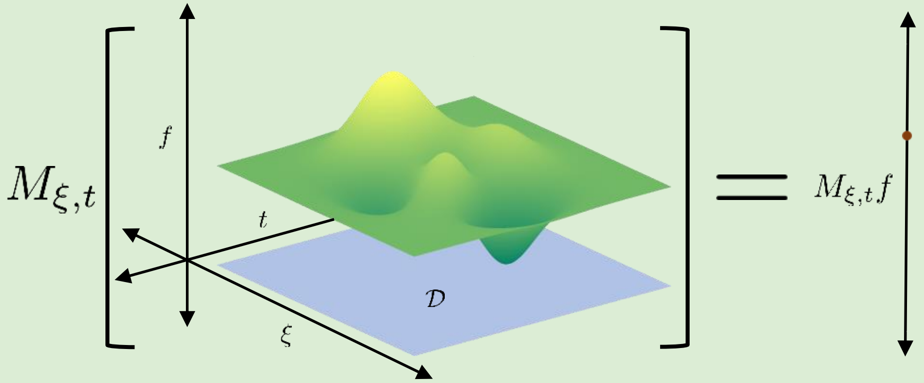
Scalarize cost function via **measure operators**.

Notation

$$M f(Dy, y(d), z, d)$$

Example

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



Constraints

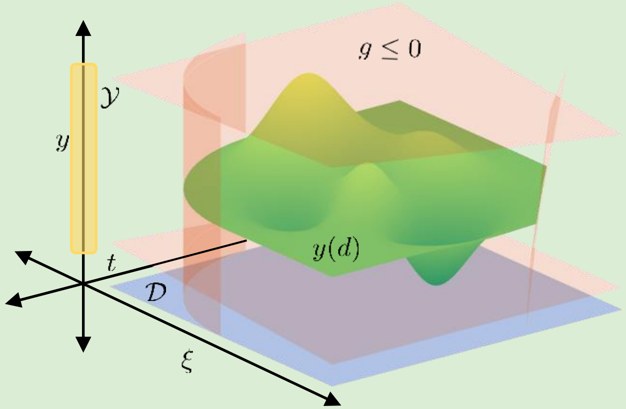
Algebraic Constraints

Enforce condition on function g for all values of d.

Notation

$$g(Dy, y(d), z, d) \leq 0, \;\; d \in \mathcal{D}$$

$$y^2(t,\xi) + y(t,\xi) \leq 0, \;\; t \in \mathcal{D}_t, \;\; \xi \in \mathcal{D}_\xi$$



Measure Constraints

Enforce constraint on measure M of functions h.

Notation

$$M h(Dy, y(d), z, d) \geq 0$$

$$\mathbb{P}_\xi(y(t,\xi) \leq 0) \geq \alpha, \;\; t \in \mathcal{D}_t$$

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

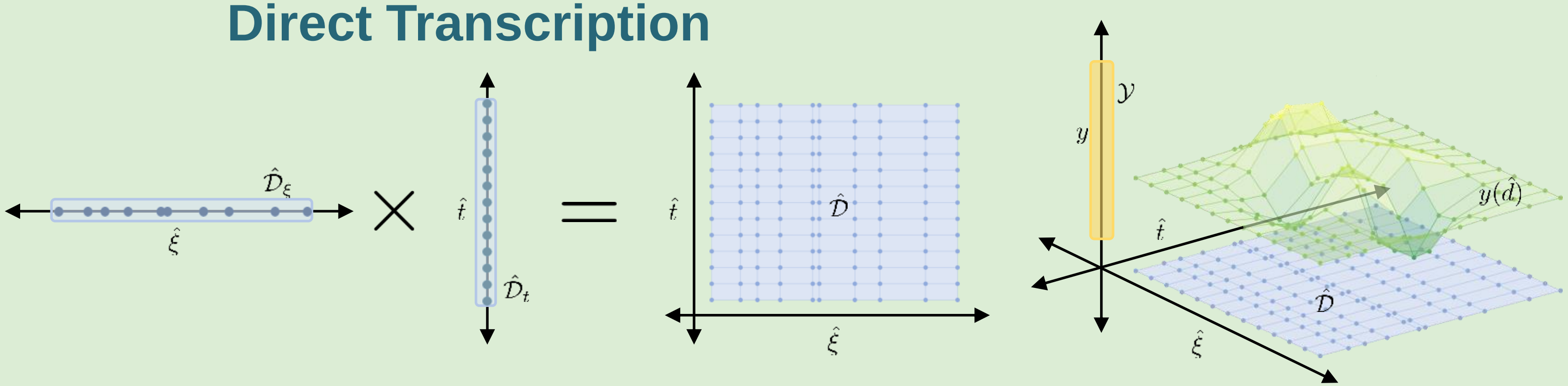
Transformations

Idea: **Transform** infinite model into a **finite formulation** that is amendable to **conventional optimization solvers**.

Direct Transcription

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

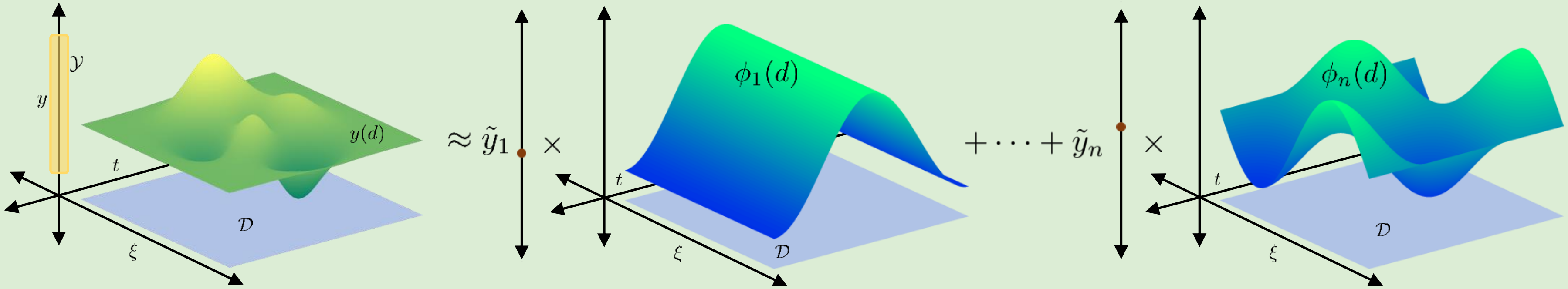
$$\hat{\mathcal{D}} := \prod_{\ell \in \mathcal{L}} \{\hat{d}_{\ell,i} : \hat{d}_{\ell,i} \in \mathcal{D}_\ell, i \in \mathcal{I}_\ell\}$$



Method of Weighted Residuals

Project onto set of known **basis functions**

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



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

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

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

```
@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**

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

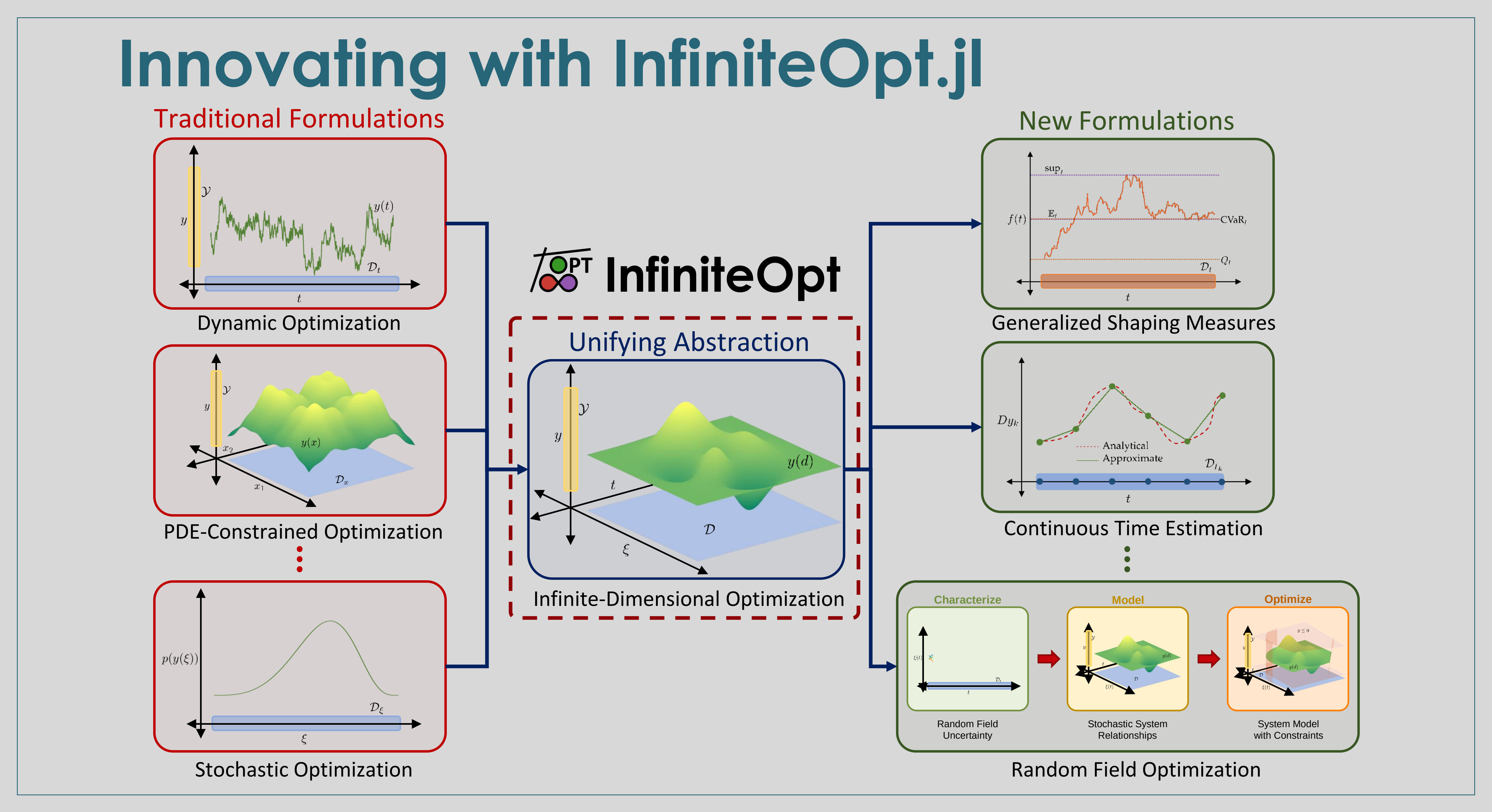
$$\mathbb{P}\left(y_b(t,\xi) \leq 0\right) \geq \alpha, \quad \forall t \in \mathcal{D}_t$$

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

```
@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)
```

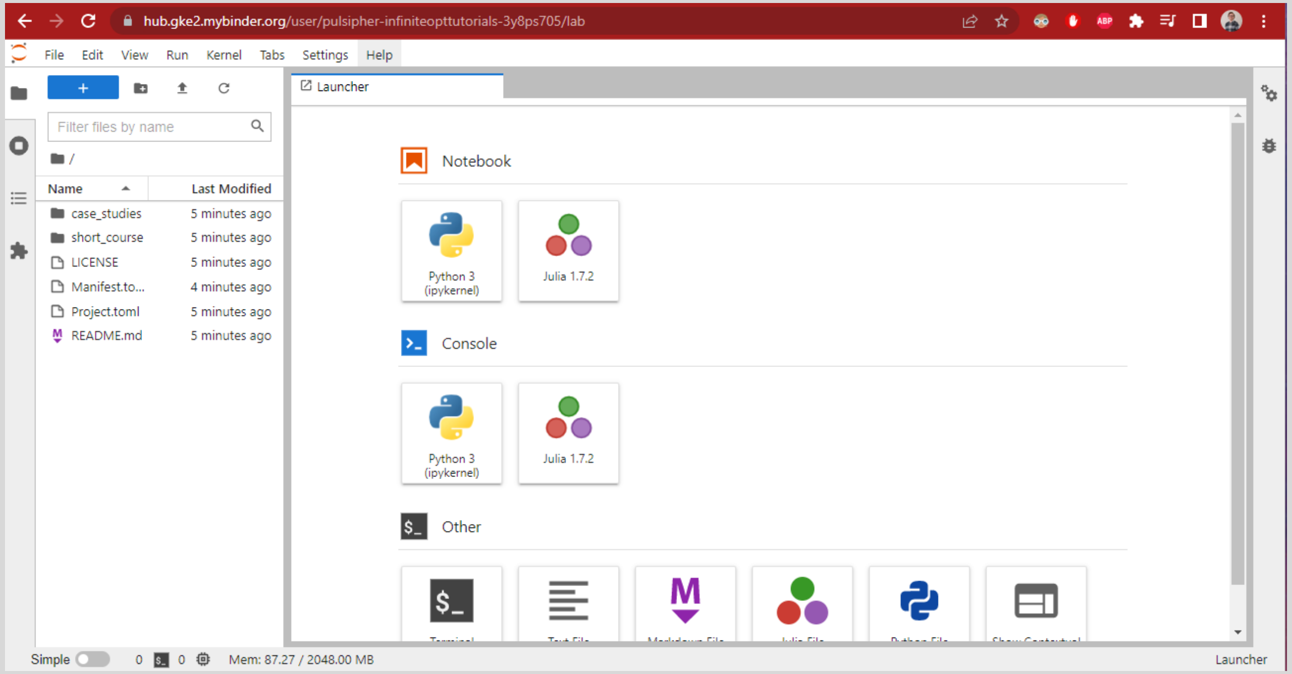


Innovating with InfiniteOpt.jl
Traditional Formulations
Dynamic Optimization
PDE-Constrained Optimization
Stochastic Optimization
InfiniteOpt
Unifying Abstraction
Infinite-Dimensional Optimization
New Formulations
Generalized Shaping Measures
Continuous Time Estimation
Analytical
Approximate
Characterize
Model
Optimize
Random Field Uncertainty
Stochastic System Relationships
System Model with Constraints
Random Field Optimization

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 1.7 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

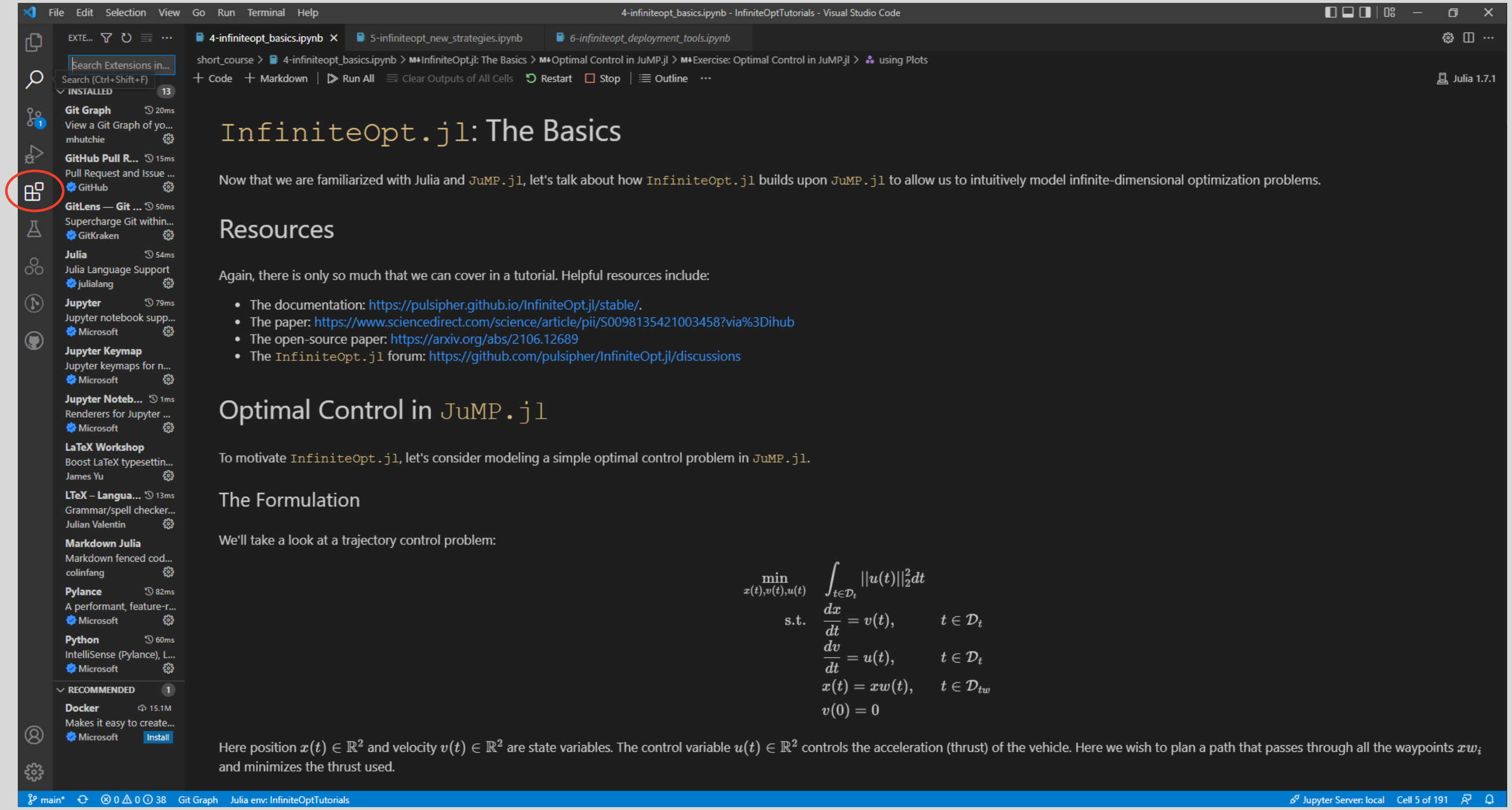
Current stable release: v1.7.2 (Feb 6, 2022)

Checksums for this release are available in both MD5 and SHA256 formats.

Windows [help]	64-bit (installer), 64-bit (portable)	32-bit (installer), 32-bit (portable)	
macOS x86 (Intel or Rosetta) [help]	64-bit (.dmg), 64-bit (.tar.gz)		
macOS ARM (M-series Processor) [help]	64-bit (experimental)		
Generic Linux on x86 [help]	64-bit (glibc) (GPG), 64-bit (musl)[1] (GPG)	32-bit (GPG)	
Generic Linux on ARM [help]	64-bit (AArch64) (GPG)	32-bit (ARMv7-a hard float) (GPG)	
Generic FreeBSD on x86 [help]	64-bit (GPG)		
Source	Tarball (GPG)	Tarball with dependencies (GPG)	GitHub

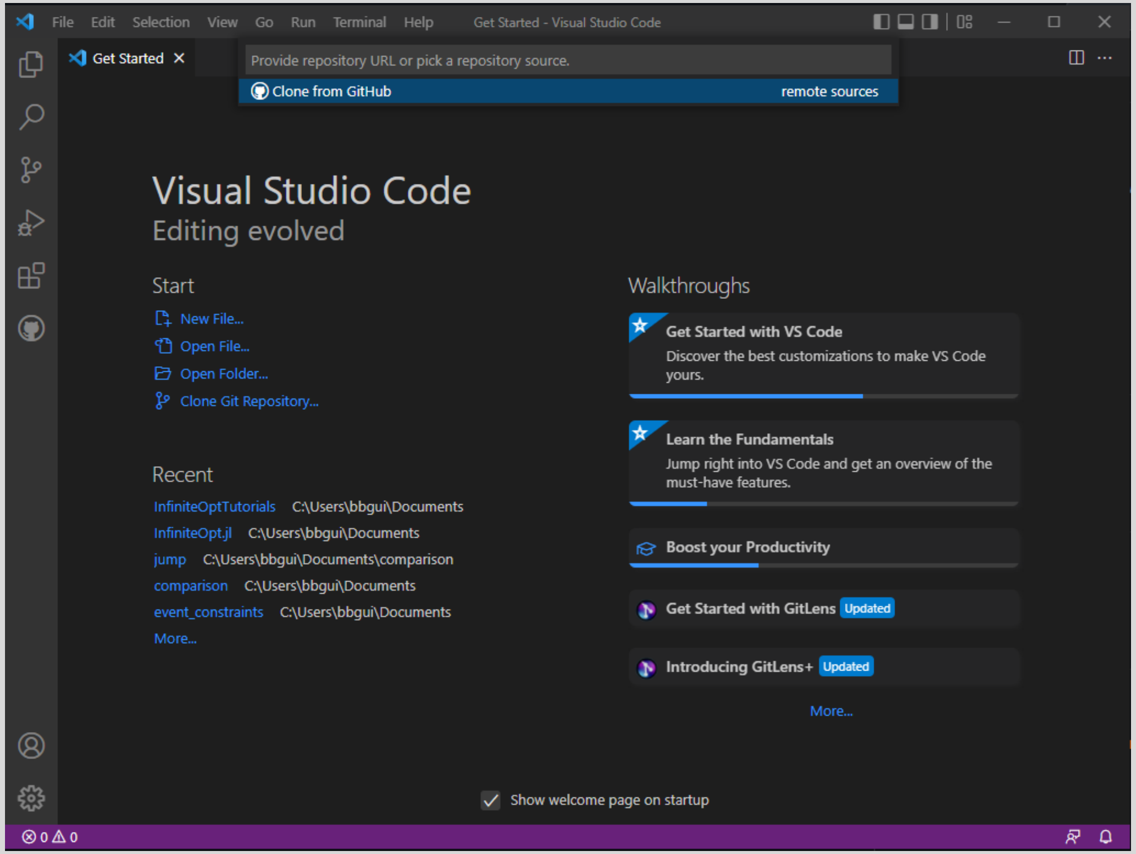
Install Visual Studio Code

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



Download the Course Material

- Located at https://github.com/pulsipher/InfiniteOptTutorials
- Download via VS Code
  - 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



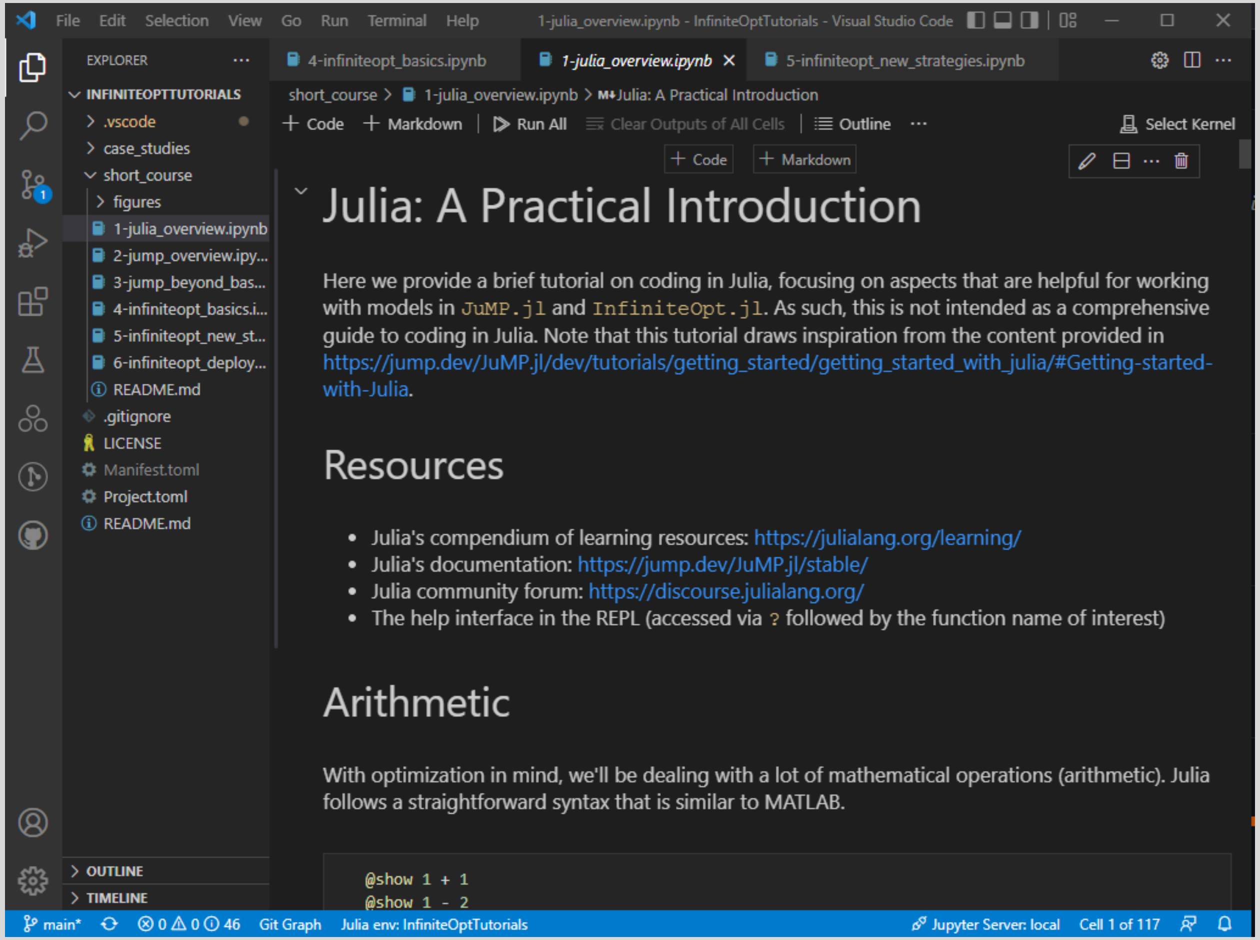
Setup the Package Environment

- 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
  - Once it's done, you can close the terminal window

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

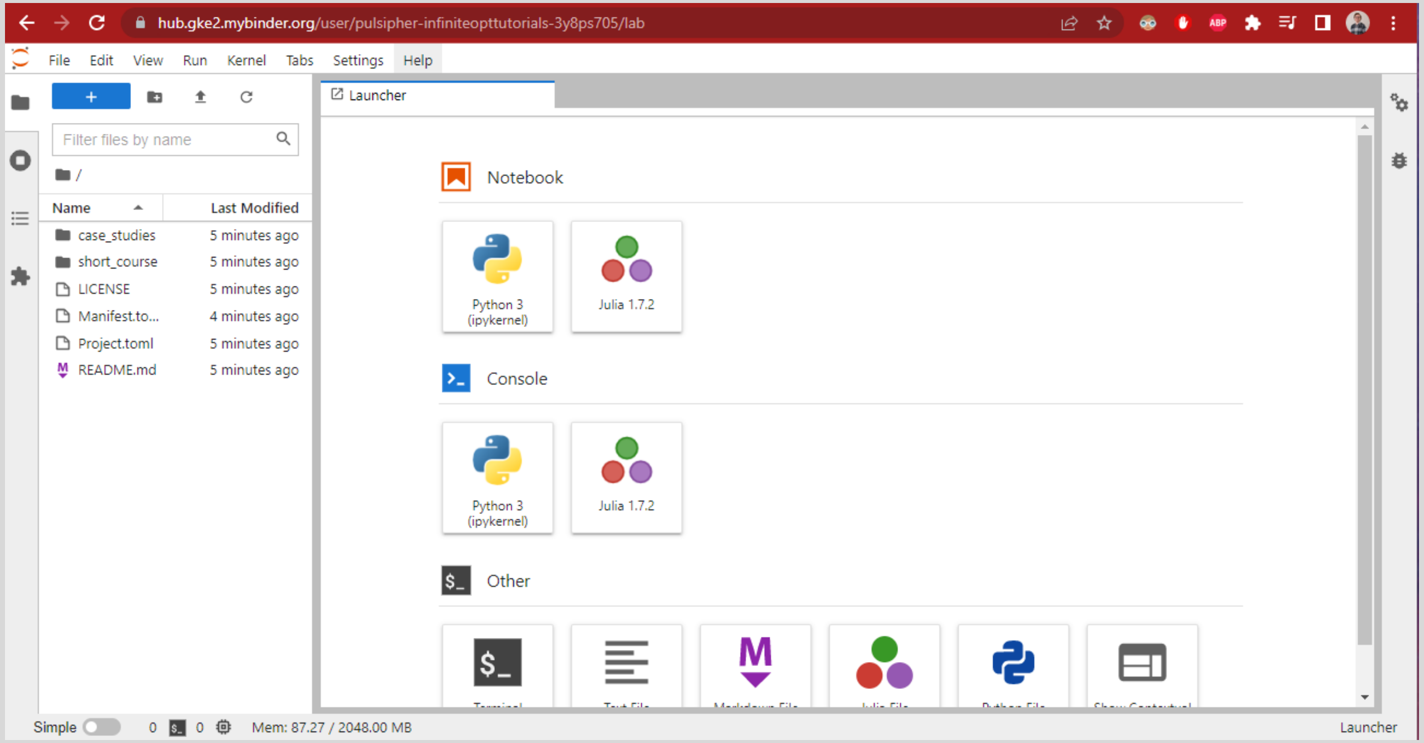
Running Jupyter Notebook

- 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



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



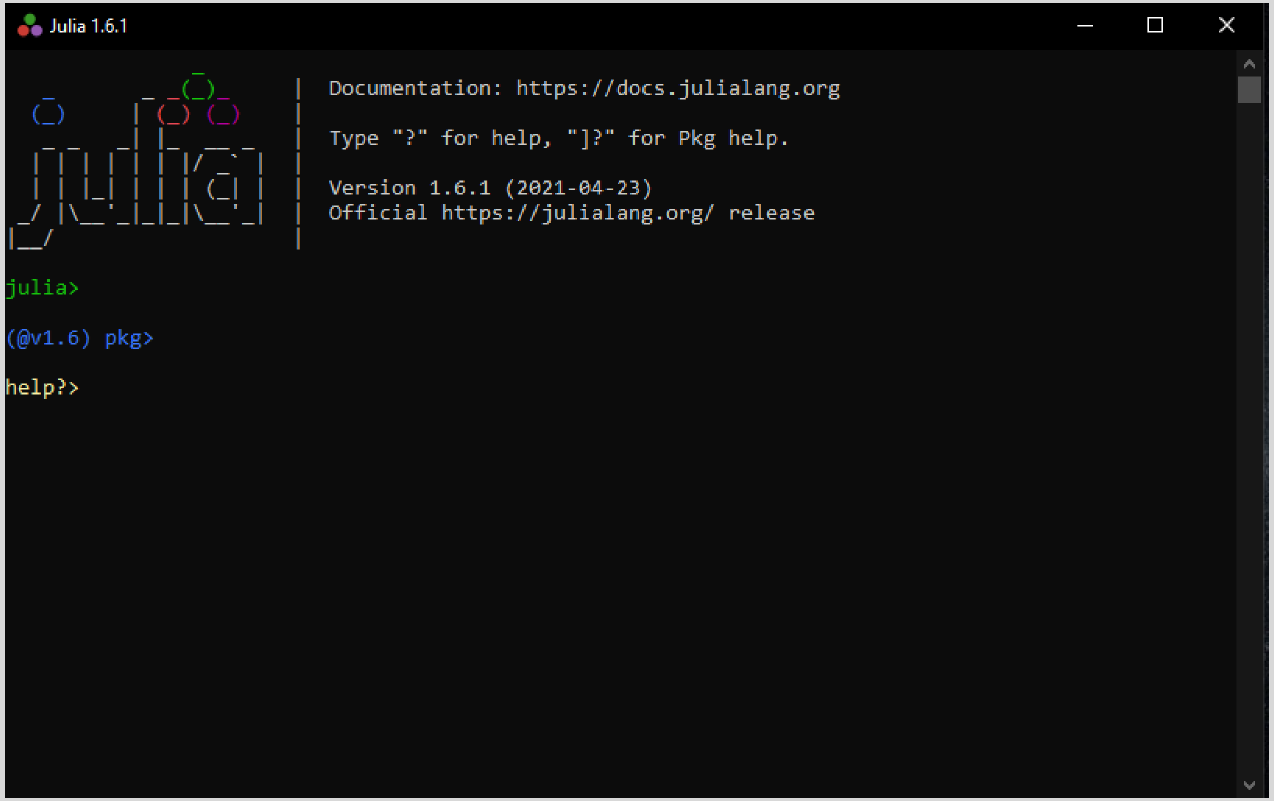
15 Minute Break Time

julia

A Practical Introduction

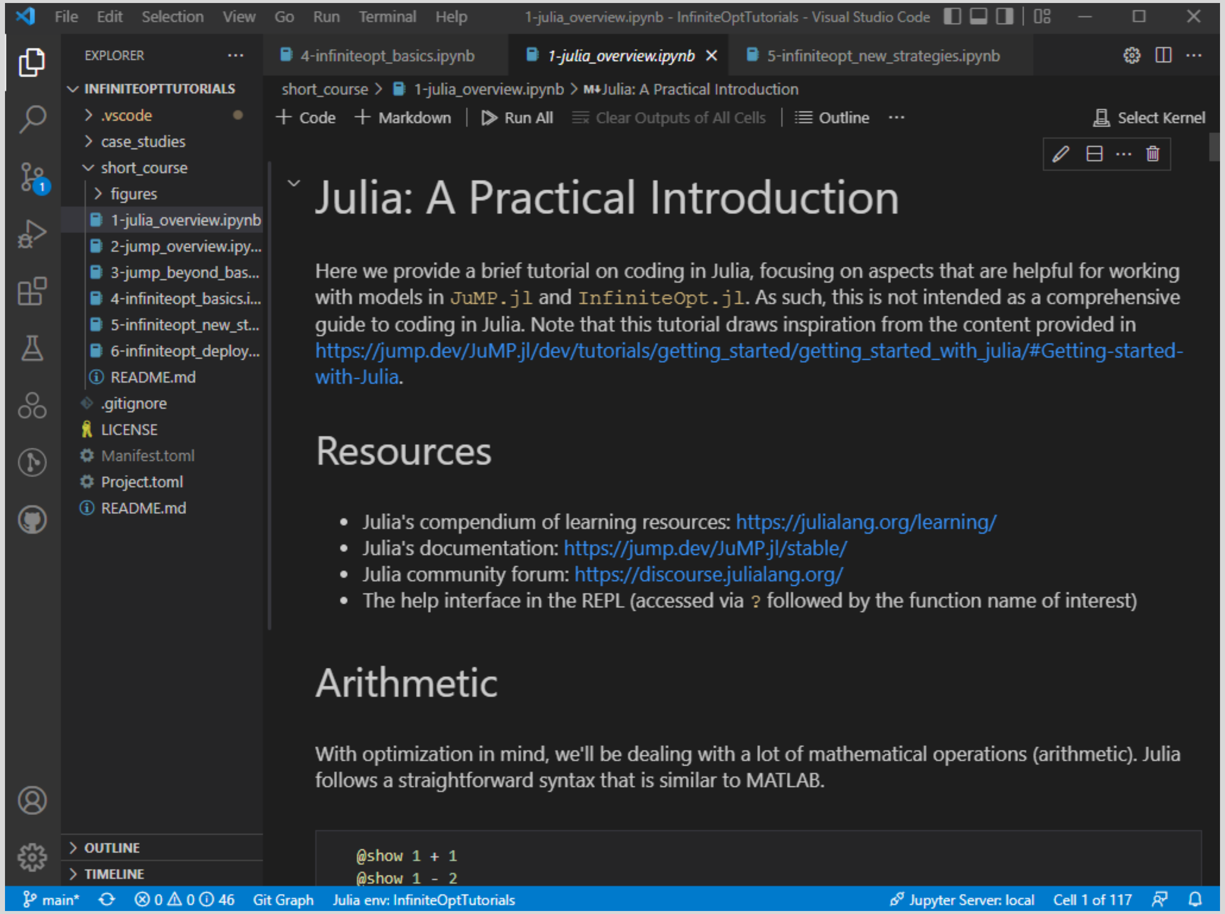
The REPL

- 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



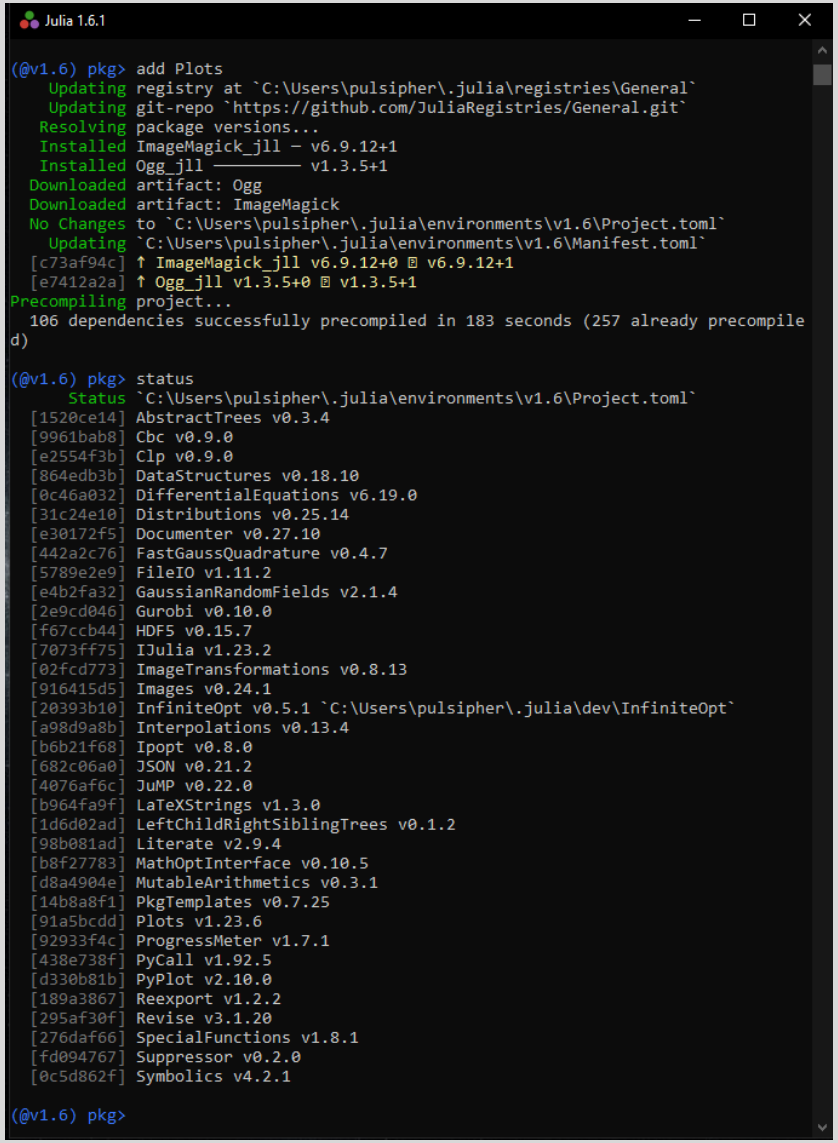
To the Tutorial!

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



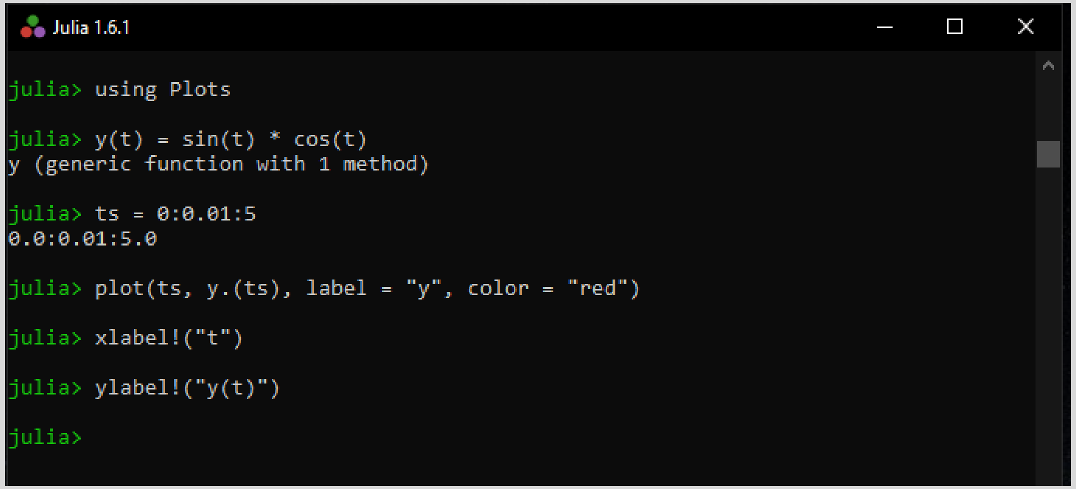
Packages

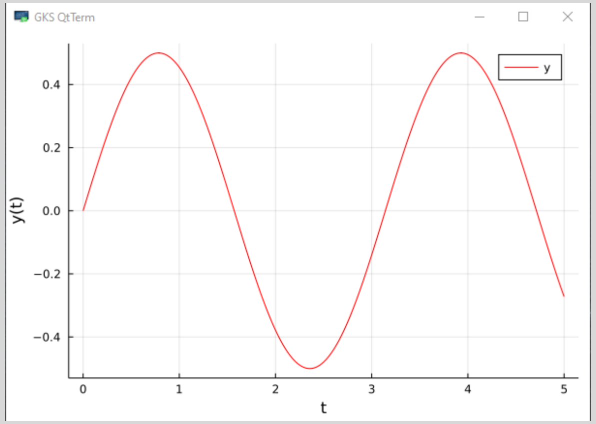
- Libraries of abilities
  - Nearly 7,000 packages are available
  - Access a wealth of functions and capabilities
- Package manager
  - Use the Pkg.jl interface in the REPL
  - Manages dependencies
- Environments
  - An independent collection of packages
  - Stops packages from stepping on each others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



Plotting

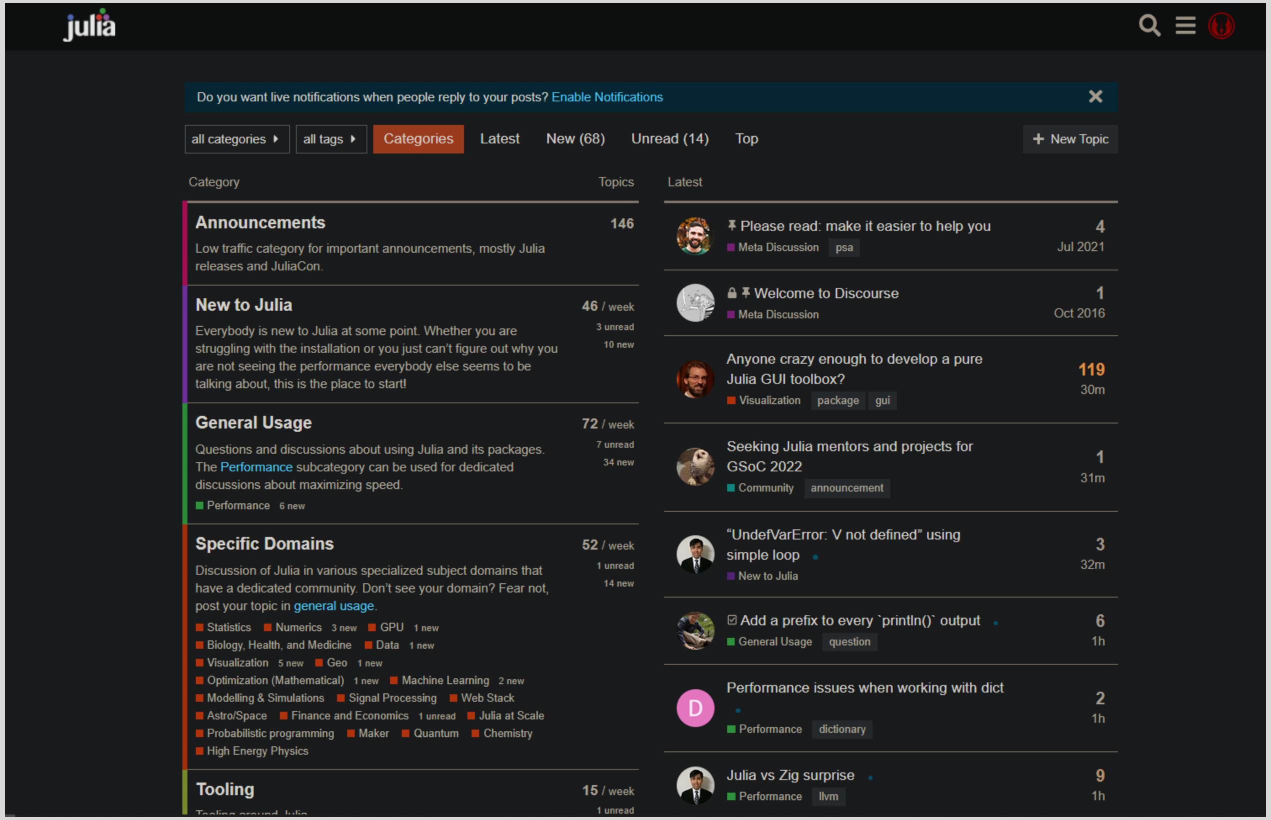
- Plots.jl
  - Efficient go to
  - The documentation is terrible ☹
- PyPlot.jl
  - Simply maps to matplotlib.pyplot
  - Imports all the all the pyplot functions
- Makie.jl
  - Native to Julia
  - Good documentation
  - Can be slow sometimes
- Others
  - GadFly.jl (pretty good 2D library)
  - PGFPlots.jl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





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**

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

- Define **variables**

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

- Define **objective**

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

- Define **constraints**

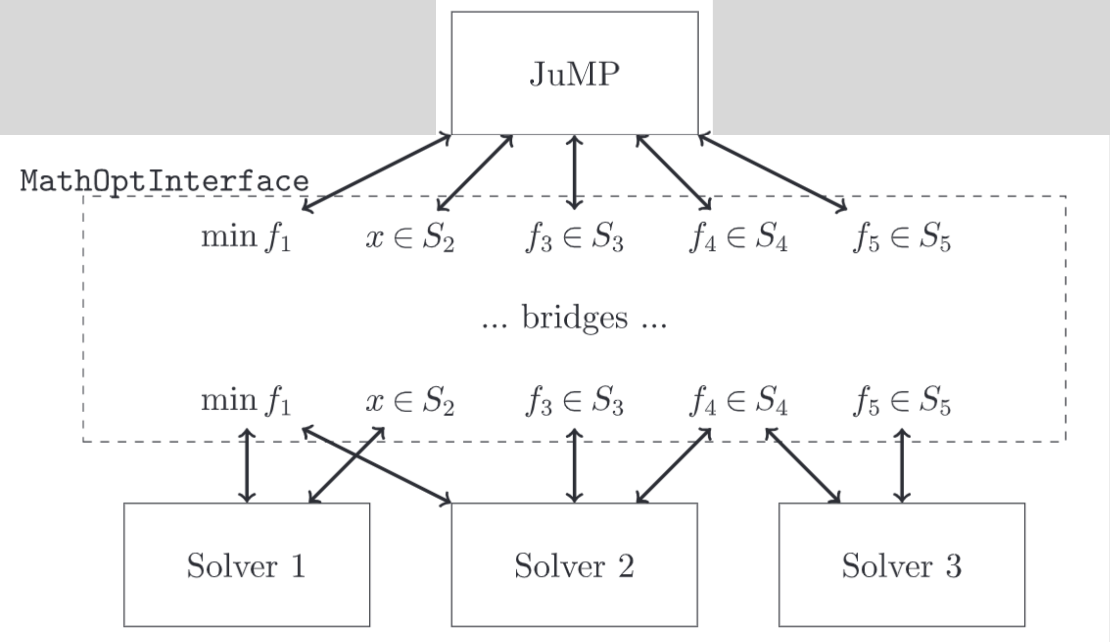
```
@constraint(model, 4f + 2s <= 4800)
@constraint(model, f + s <= 1750)
@constraint(model, 0 <= f <= 1000)
@constraint(model, 0 <= s <= 1500)
```

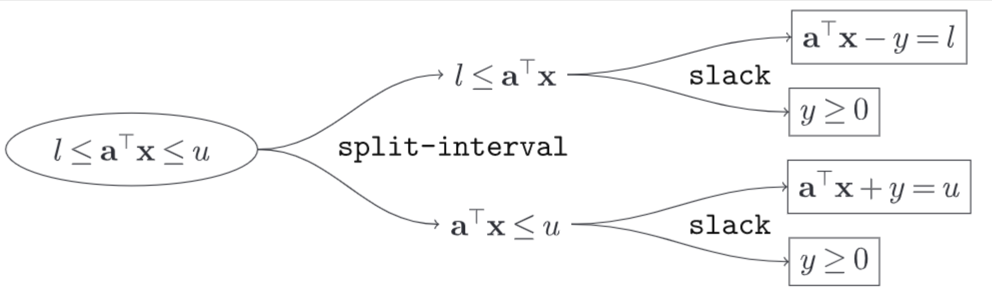
- **Solve**

```
optimize!(model)
profit = objective_value(model)
f_best = value(f)
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

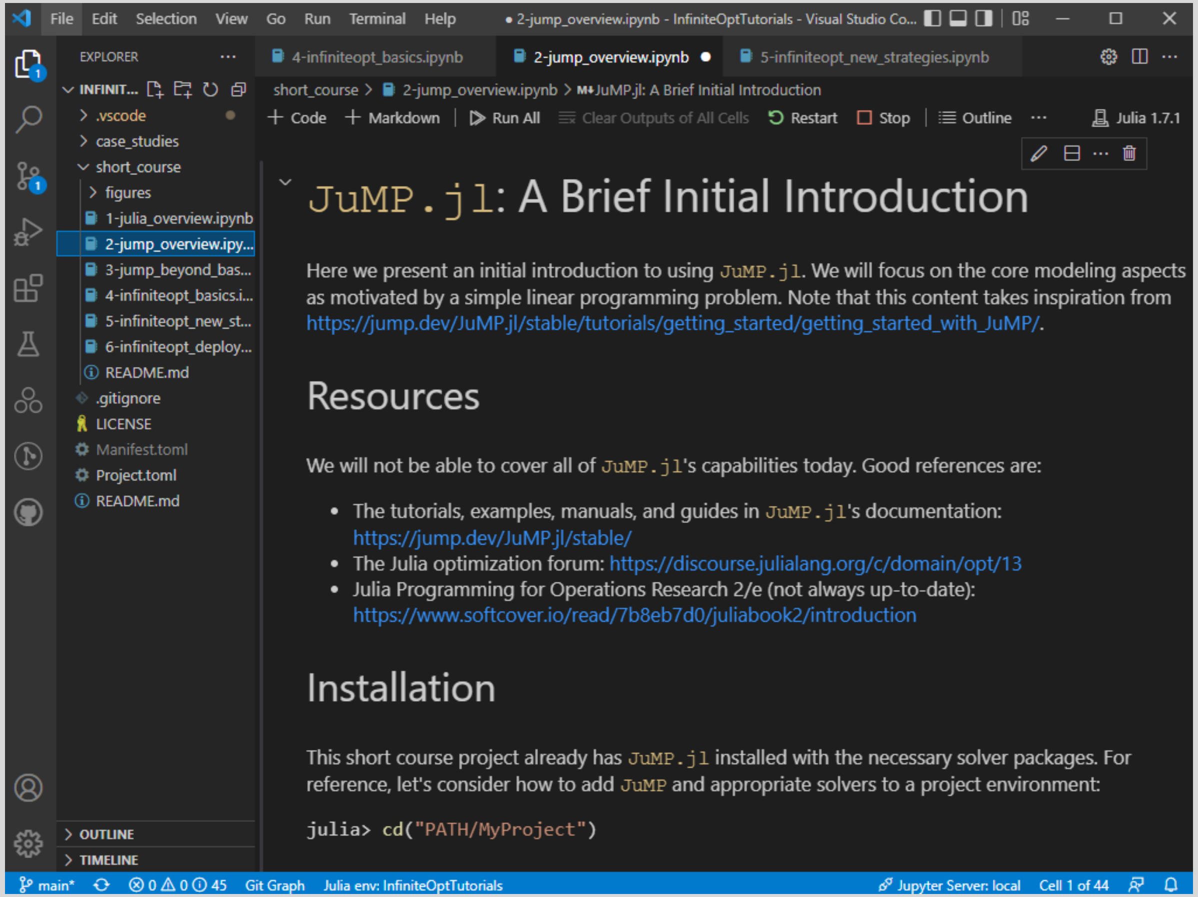




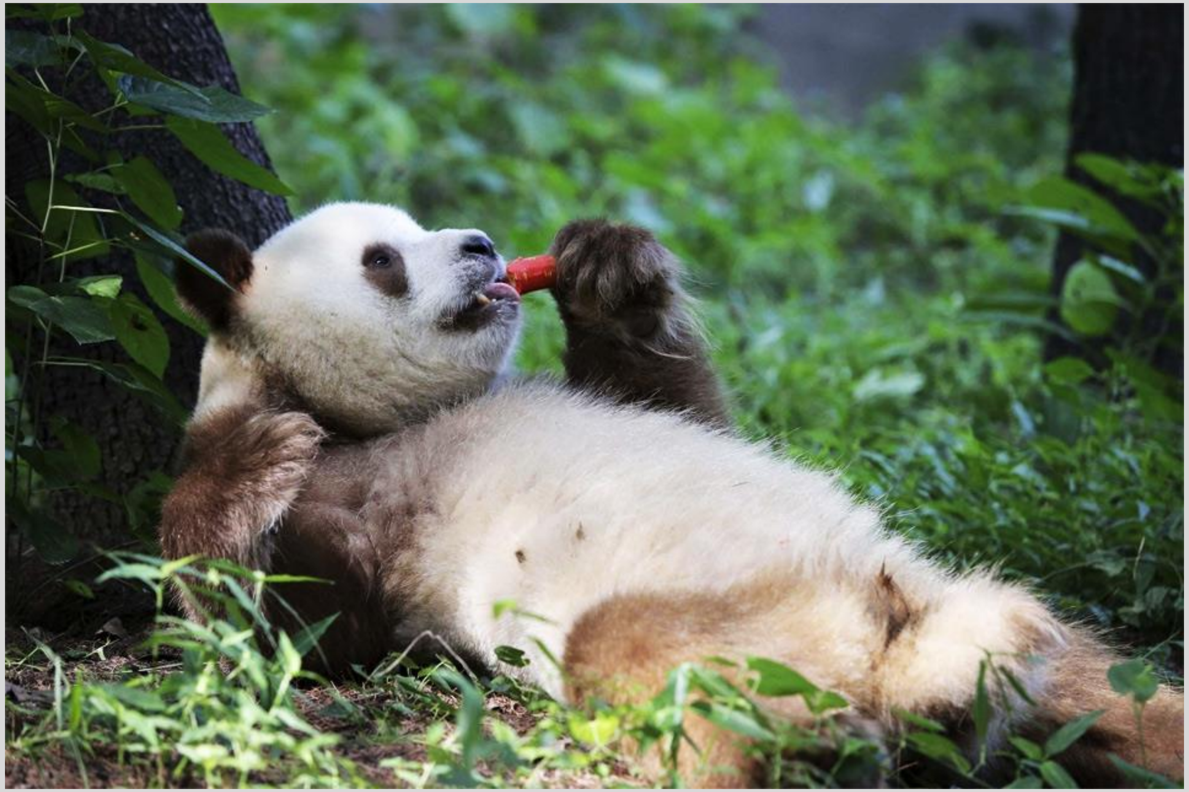
Legat, Benoît, et al. "MathOptInterface: a data structure for mathematical optimization problems." *INFORMS Journal on Computing* 34.2 (2022): 672-689.

To the Tutorial!

- Open the “2-jump_overview.ipynb” jupyter notebook



1 Hour Lunch Break

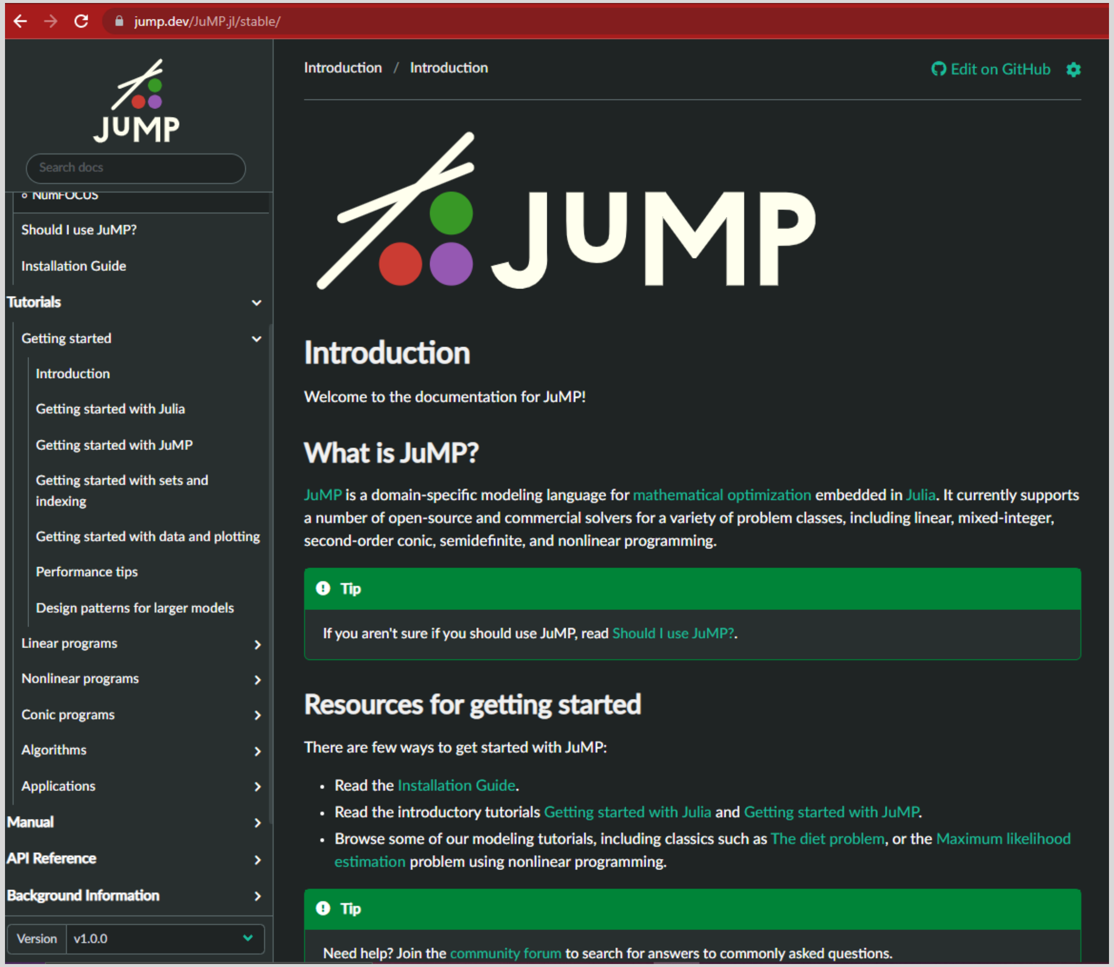


JuMP

Beyond the basics

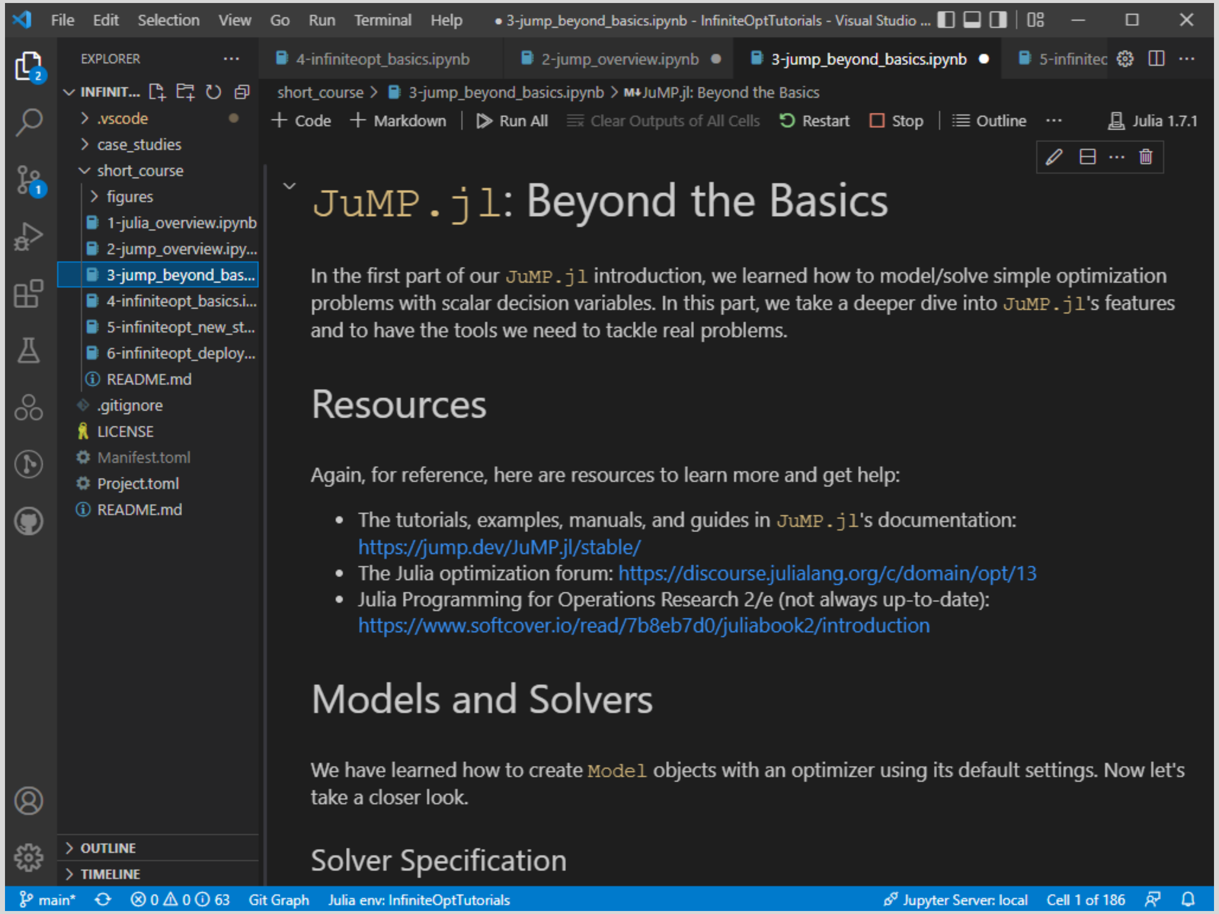
Learning JuMP.jl

- 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

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

```
@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**

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

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

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

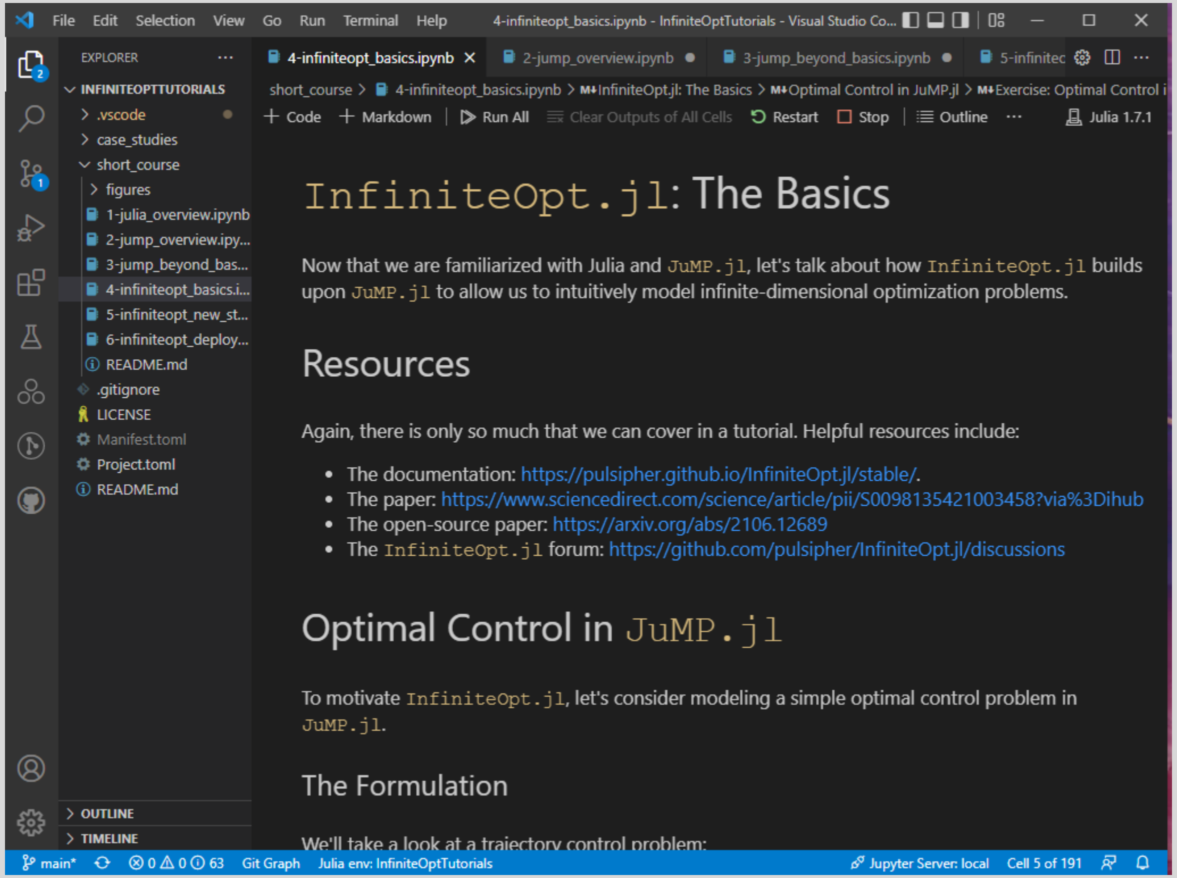
```
@constraint(m, ∂(yb, t) == 2yb^2 + ya - z[1])
@constraint(m, yb ≤ yc * U)
@constraint(m, 𝔼(yc, ξ) ≥ α)
@constraint(m, ya(0) + z[2] == β)
```

- **Solve**

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

To the Tutorial!

- Open the “4-infiniteopt_basics.ipynb” jupyter notebook



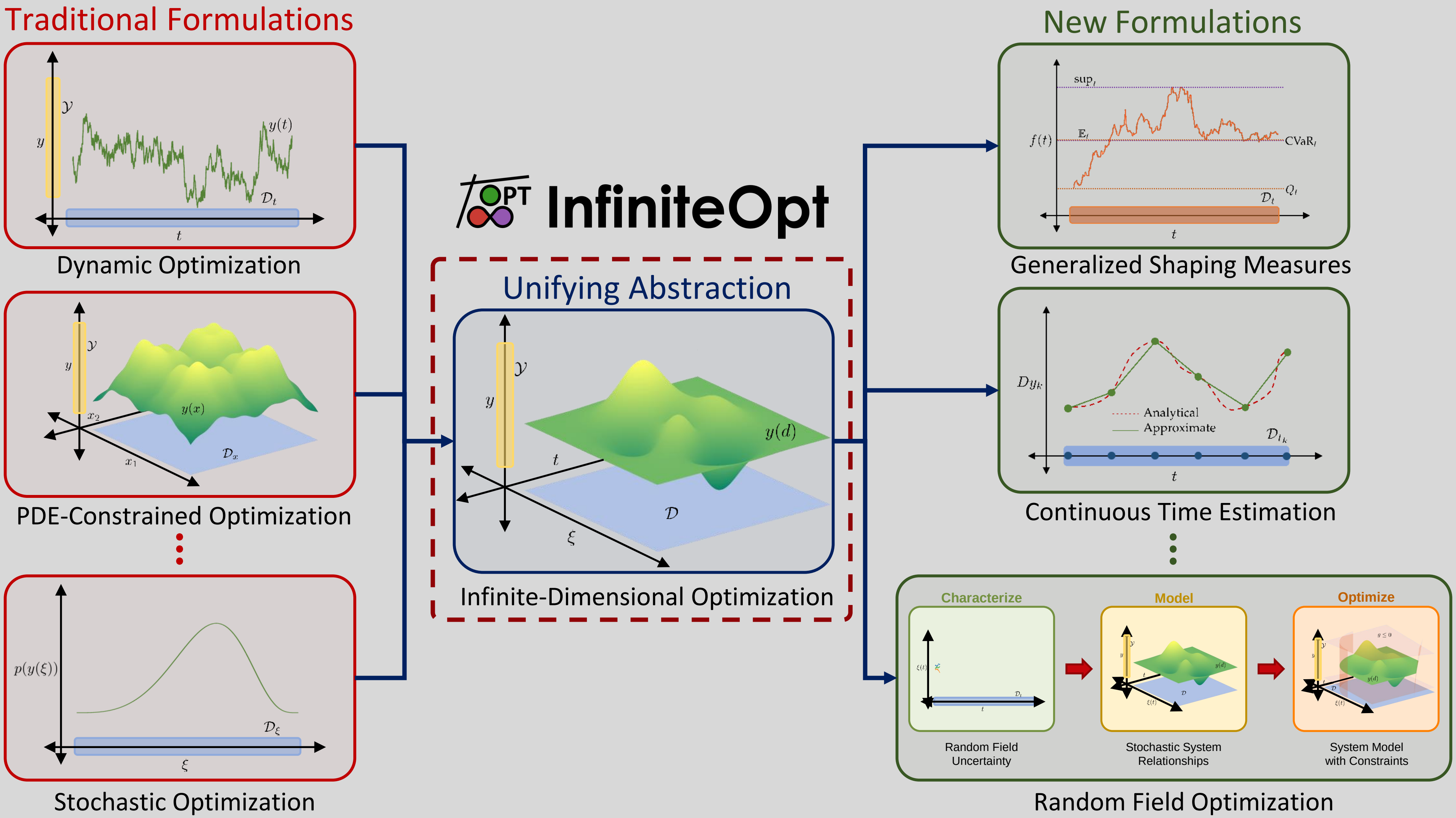
15 Minute Break Time

Why is InfiniteOpt.jl so awesome?

InfiniteOpt

New Modeling Strategies

Innovating with InfiniteOpt.jl

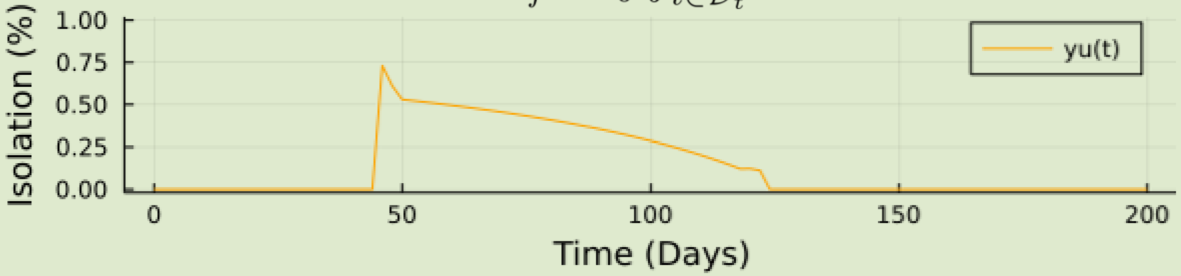


Measure Comparison

Expectation (Uniform pdf)

- Weight the cost **uniformly**

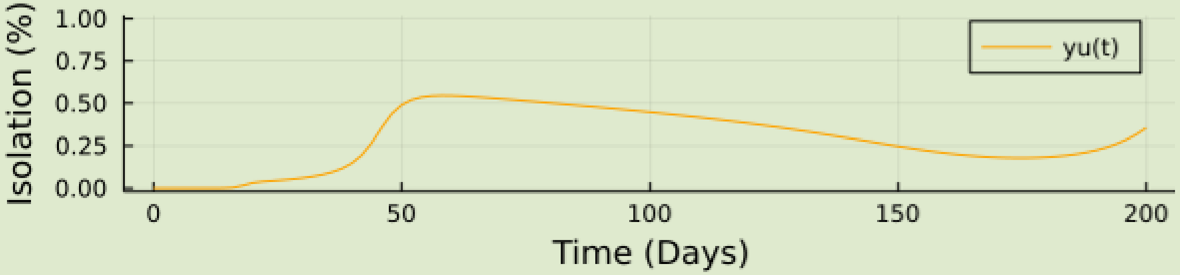
$$\mathbb{E}_t[y_u(t)] = \frac{1}{t_f - t_0}\int_{t\in\mathcal{D}_t} y_u(t)dt$$



Expectation (Exponential pdf)

- Place emphasis on the **initial times**

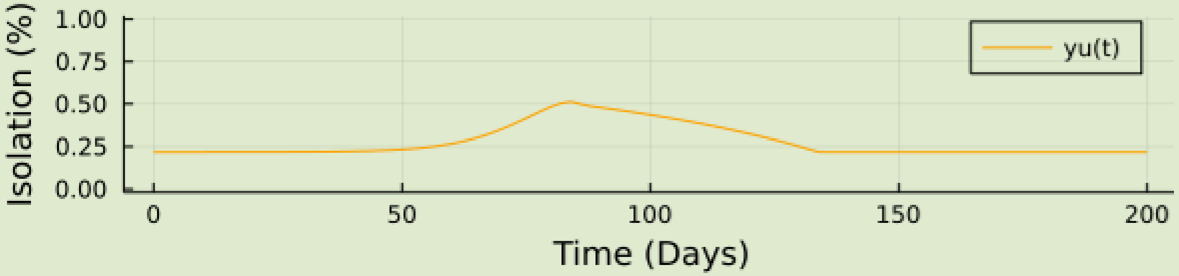
$$\mathbb{E}_t[y_u(t)] = \int_{t\in\mathcal{D}_t} y_u(t)e^{-t}dt$$



Mean-Variance

- Penalize **fluctuations** in the trajectory

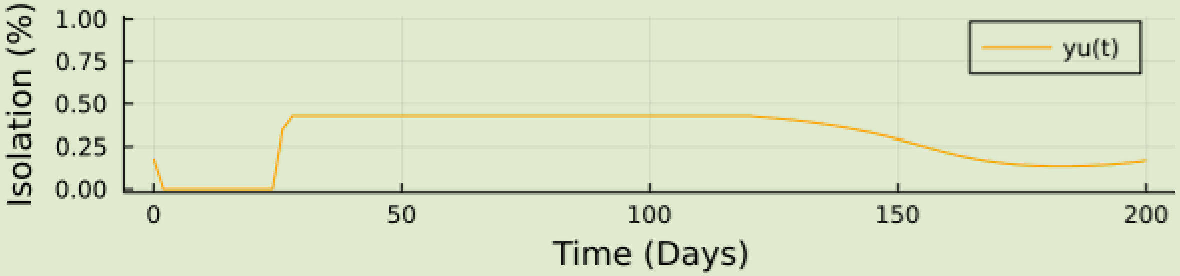
$$\mathbb{E}\text{-}\mathbb{V}_t = \mathbb{E}_t[y_u(t)] + \lambda\mathbb{V}_t[y_u(t)]$$



CVaR

- Only penalize the **high cost peaks**

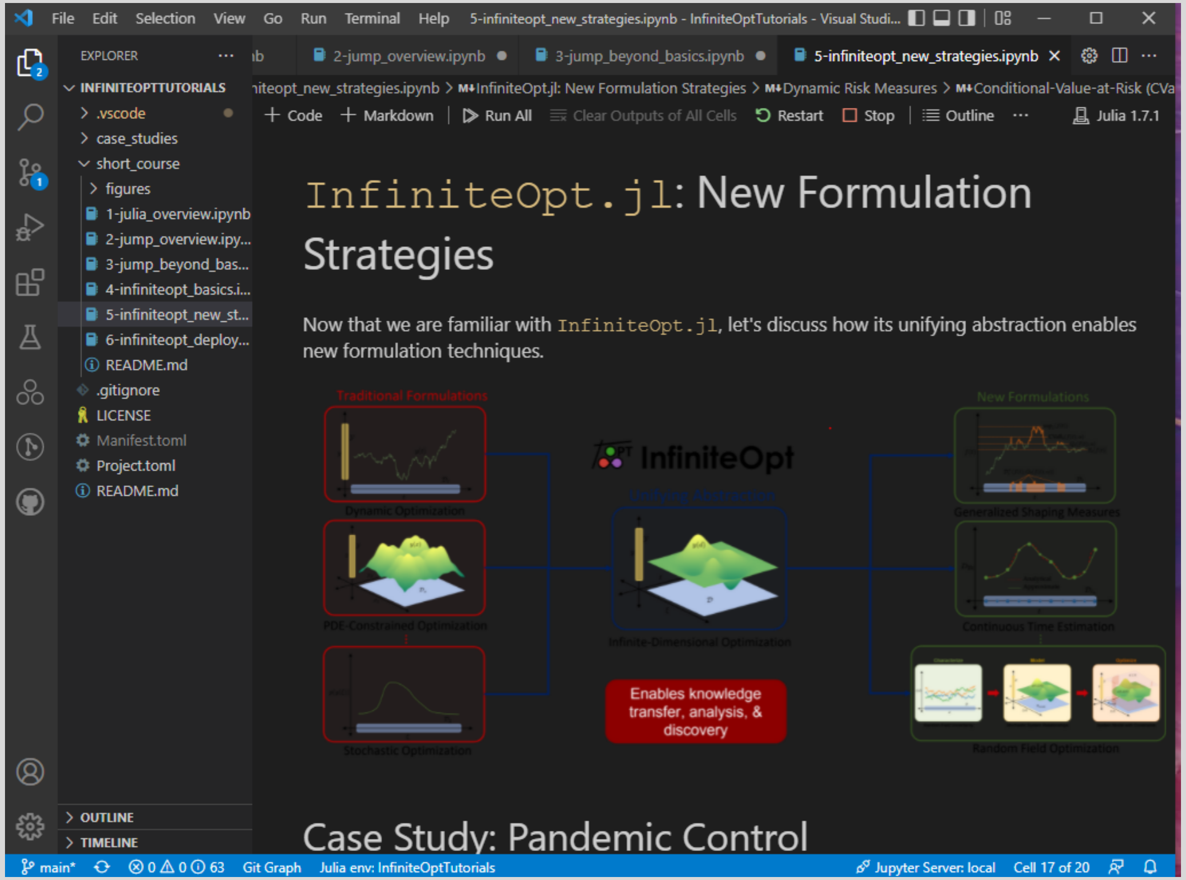
$$\text{CVaR}_t(y_u(t);\alpha)$$



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

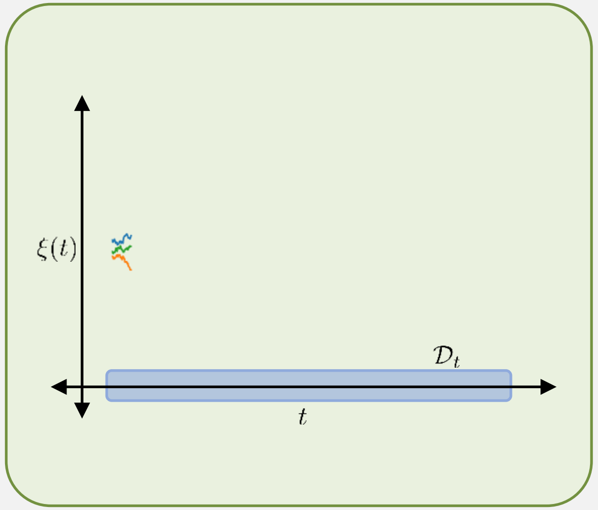
To the Tutorial!

- Open the “5-infiniteopt_new_strategies.ipynb” jupyter notebook



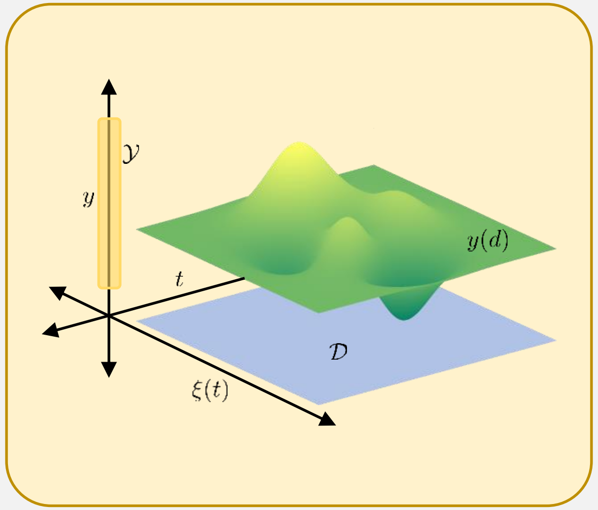
Random Field Optimization

Characterize



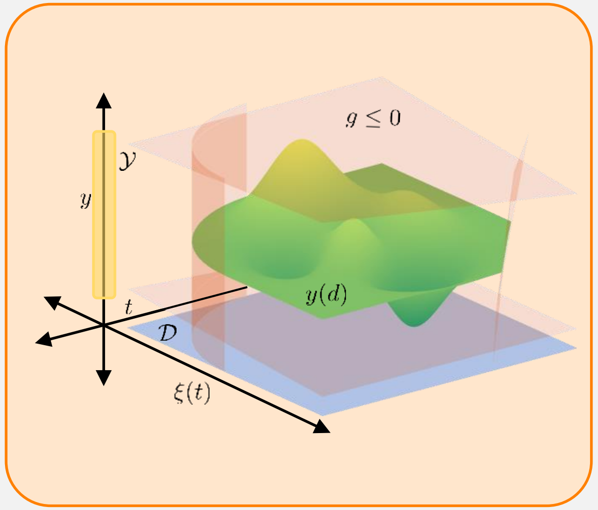
Random Field
Uncertainty

Model



Stochastic System
Relationships

Optimize



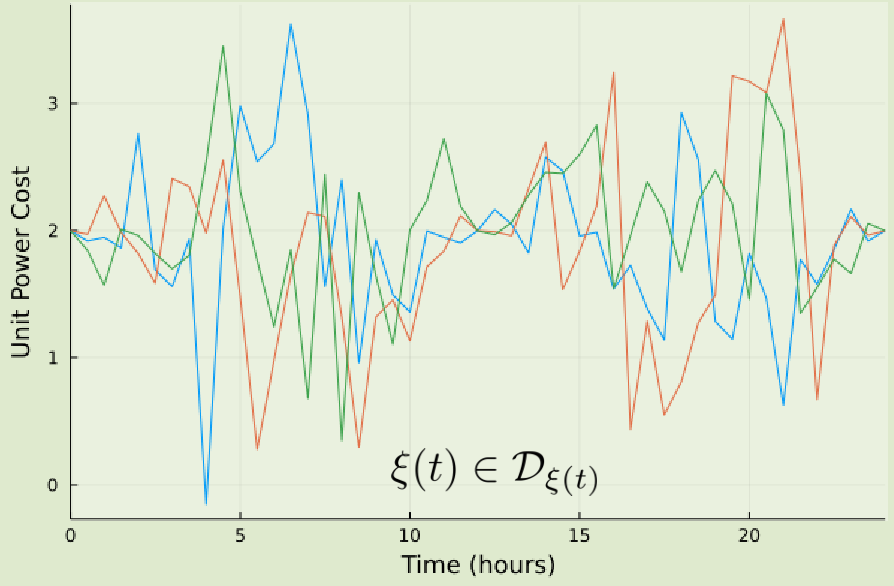
System Model
with Constraints

Illustrative Examples

Idea: Motivate framework with decoupled examples in **time** and **space**

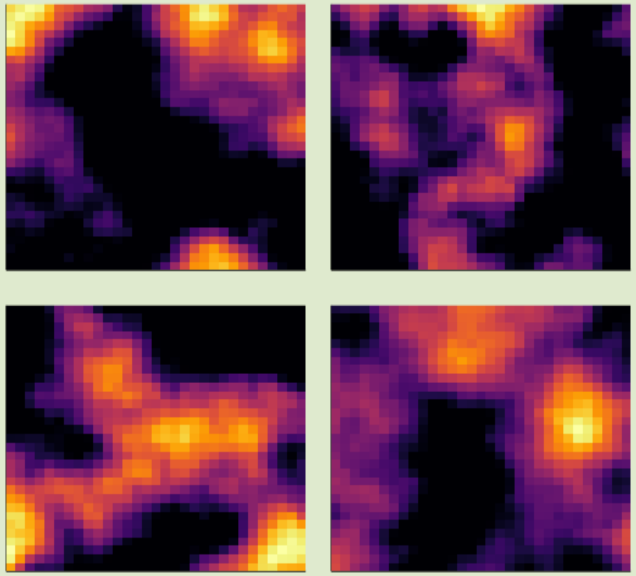
Dynamic Electricity Market

- Random field for **pricing forecast**



Spatial Thermal Diffusivity

- Random field for **uncertain diffusivity**



$\xi(x) \in \mathcal{D}_{\xi(x)}$

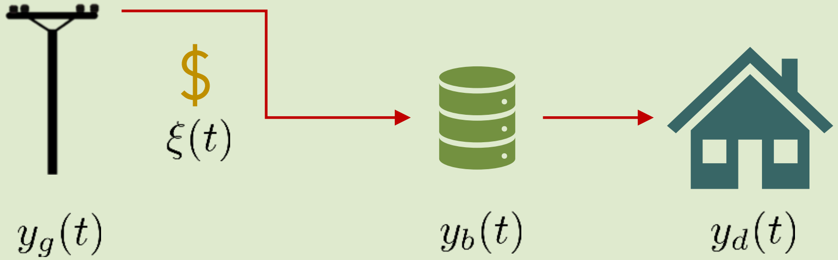
Simplifying Assumption: Fields are continuous and differentiable → avoid stochastic calculus

Pulsipher, J.L., Davidson, B.R., and Zavala, V.M. *Random Field Optimization*. Under Review. 2022

Illustrative Examples

Battery-Aided Electricity System

- **Store power** via battery to avoid high costs

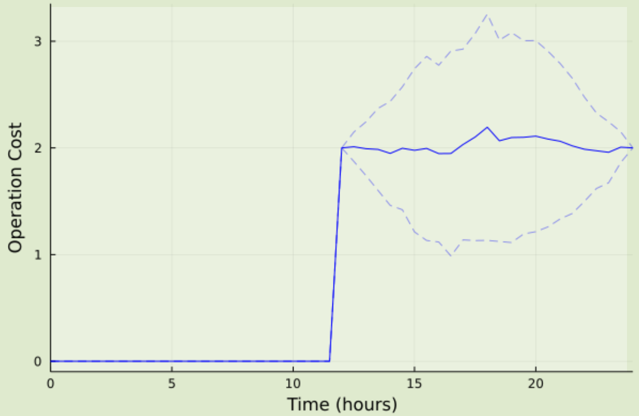


- **Simulate** with predetermined purchasing

$$\frac{dy_b(t)}{dt} = y_g(t) - y_d(t)$$

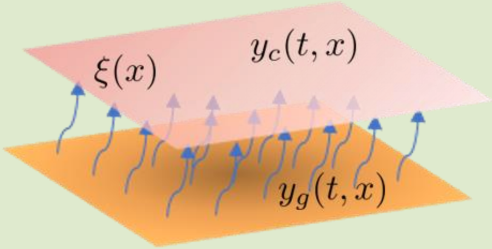
$$f(t) = \xi(t) y_g(t)$$

$$y_b(0) = y_{b0}$$



Transient Thermal Diffusion

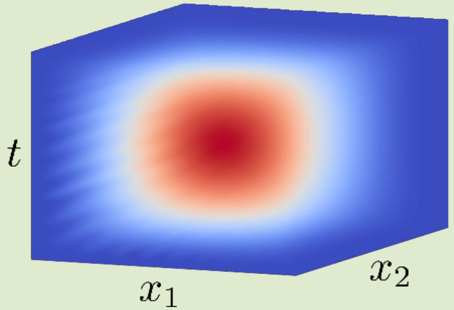
- Control a plate's **temperature distribution**



- **Simulate** with uniform heating

$$\frac{\partial y_c(t,x)}{\partial t} = \xi(x)\left(\frac{\partial^2 y_c(t,x)}{\partial x_1^2} + \frac{\partial^2 y_c(t,x)}{\partial x_2^2}\right) + y_g(t,x)$$

$$y_c(0,x), y_c(t, \text{boundary}) = 0$$



Pulsipher, J.L., Davidson, B.R., and Zavala, V.M. *Random Field Optimization*. Under Review. 2022

Illustrative Examples

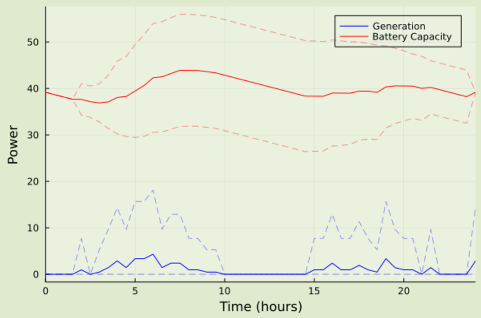
Optimal Battery Design

- Optimally choose the **battery size**

$$
\begin{aligned}
\min \quad & \mathbb{E}_{\xi(t)}\left[\int_{t\in\mathcal{D}_t} \xi(t) y_g(t,\xi(t))dt\right] + 0.1z \\
\text{s.t.} \quad & \frac{dy_b(t,\xi(t))}{dt} = y_g(t,\xi(t)) - 1, \quad t\in\mathcal{D}_t,\ \xi(t)\in\mathcal{D}_\xi(t) \\
& 0.2z \le y_b(t,\xi(t)) \le z, \qquad t\in\mathcal{D}_t,\ \ \xi(t)\in\mathcal{D}_\xi(t) \\
& y_b(0,\xi(0)) = 0.5z, \qquad \xi(0)\in\mathcal{D}_\xi(0) \\
& y_b(24,\xi(24)) = 0.5z, \qquad \xi(24)\in\mathcal{D}_\xi(24)
\end{aligned}
$$

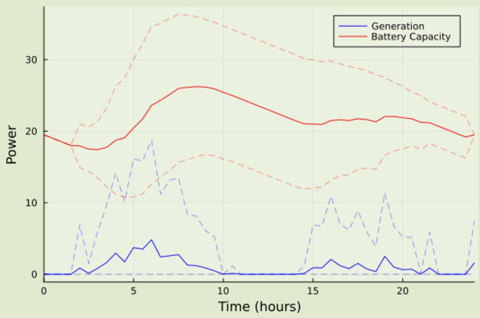
- **Results**

Deterministic



$z = 78.3,\ \text{cost} = 14.7$

Stochastic



$z = 39,\ \boxed{\text{cost} = 12}$

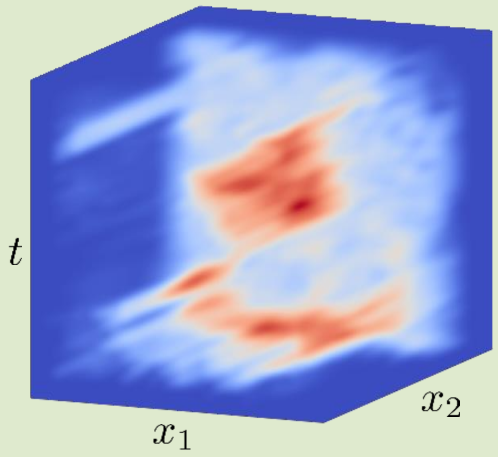
Optimal Temperature Distribution

- Optimally **control heating** to reach setpoint

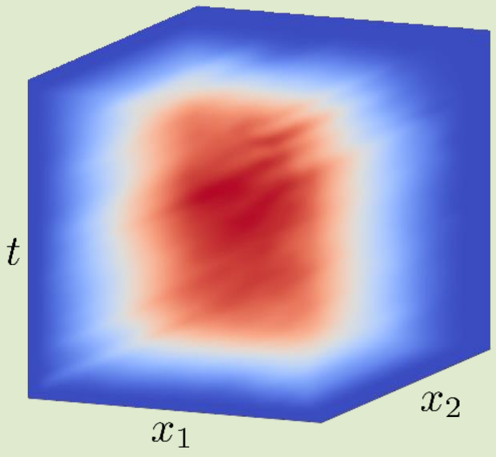
$$
\begin{aligned}
\min \quad & \mathbb{E}_{\xi(x)}\left[\int_{(t,x)\in\mathcal{D}_{t,x}} (y_c(t,x,\xi(x)) - \bar{y}_c(x))^2 dtdx\right] \\
\text{s.t.} \quad & \frac{\partial y_c(t,x,\xi(x))}{\partial t} = \xi(x)\nabla_x^2 y_c(t,x,\xi(x)) + y_g(t,x),\ (t,x)\in\mathcal{D}_{t,x},\ \xi(x)\in\mathcal{D}_{\xi(x)} \\
& y_c(0,x,\xi(x)), y_c(t,\text{boundary},\xi(\text{boundary})) = 0,\ (t,x)\in\mathcal{D}_{t,x},\ \xi(x)\in\mathcal{D}_{\xi(x)}
\end{aligned}
$$

- **Results**

Heating



Average Temperature



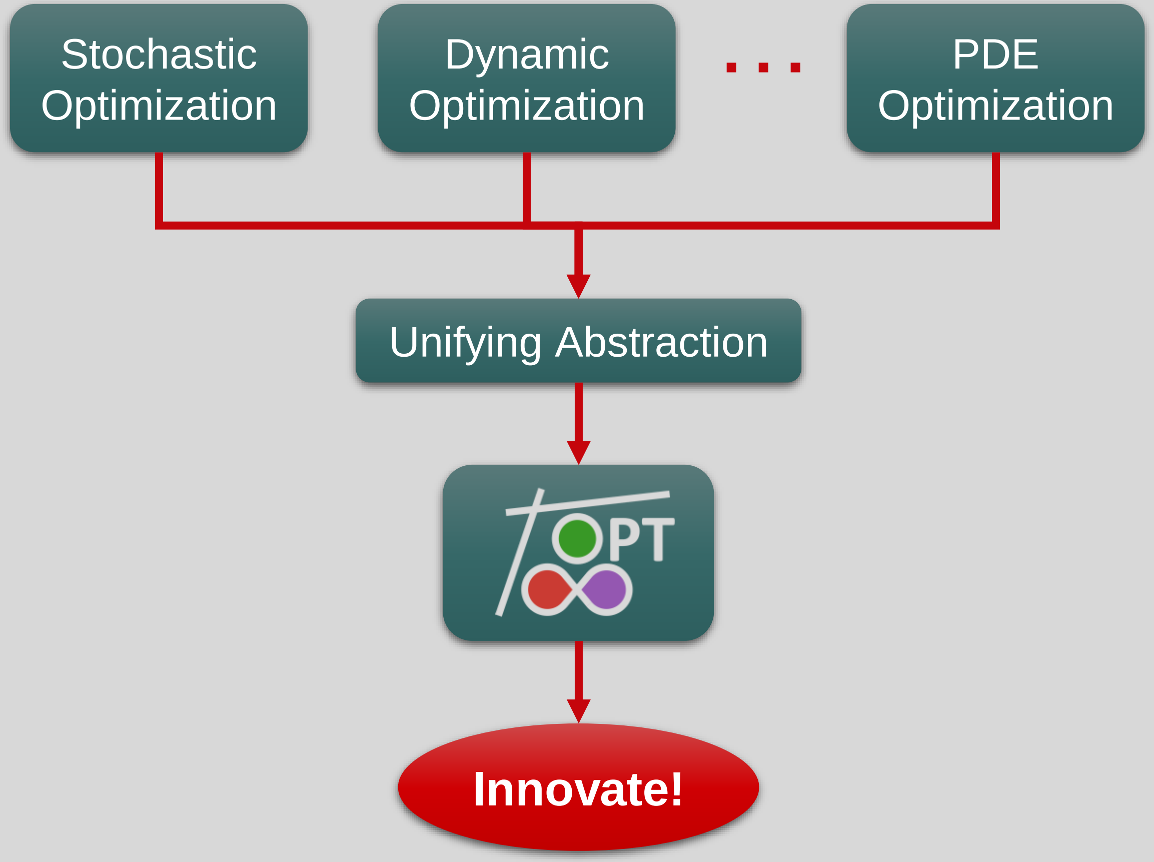
Pulsipher, J.L., Davidson, B.R., and Zavala, V.M. *Random Field Optimization*. Under Review. 2022

InfiniteOpt

Deployment Tools

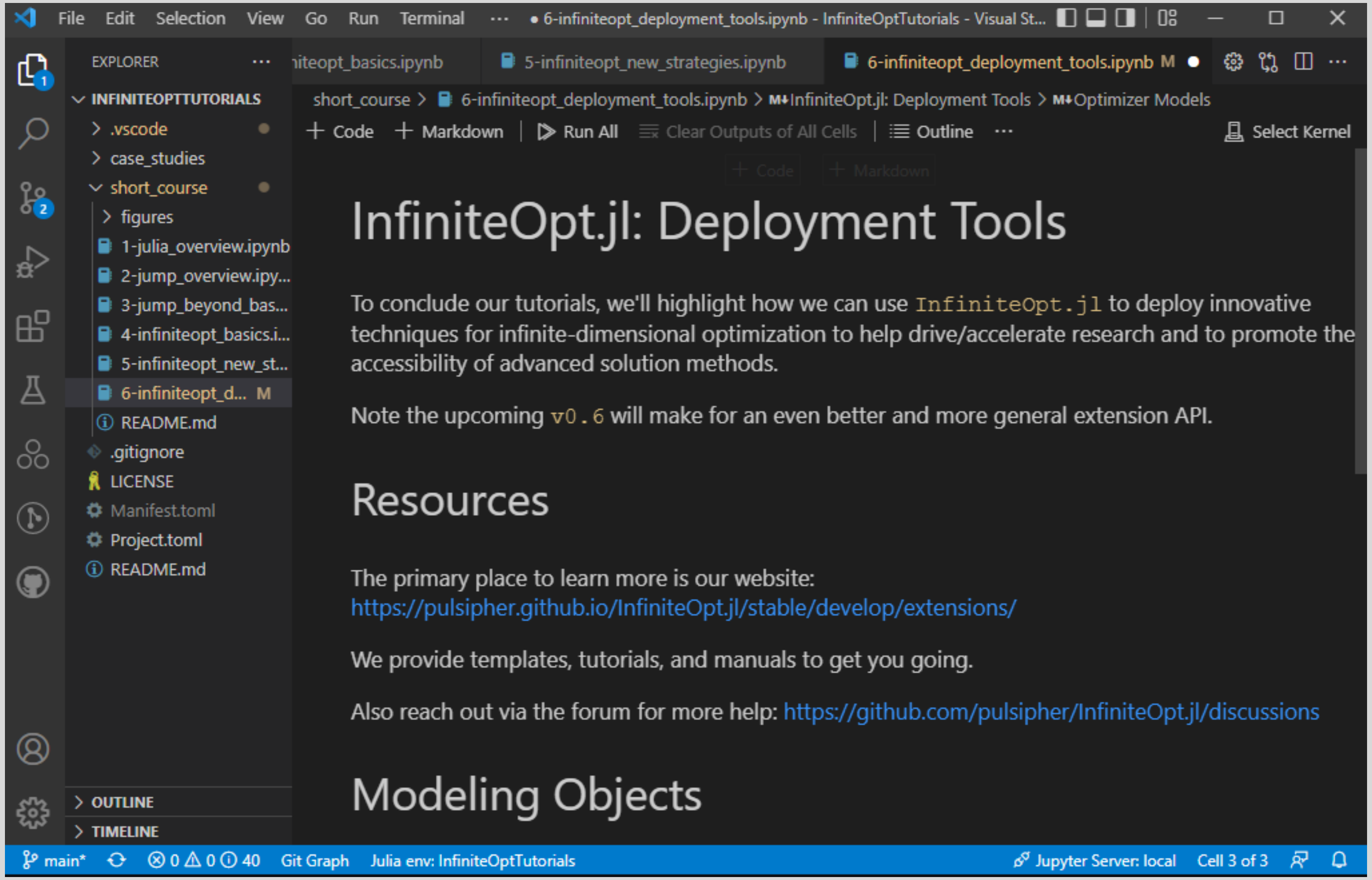
Want to deploy your cool techniques?

- 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

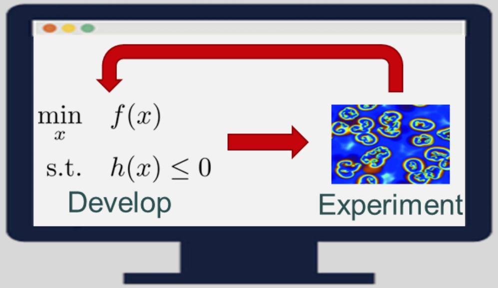


InfiniteOpt

Final Thoughts

When should I use InfiniteOpt.jl?

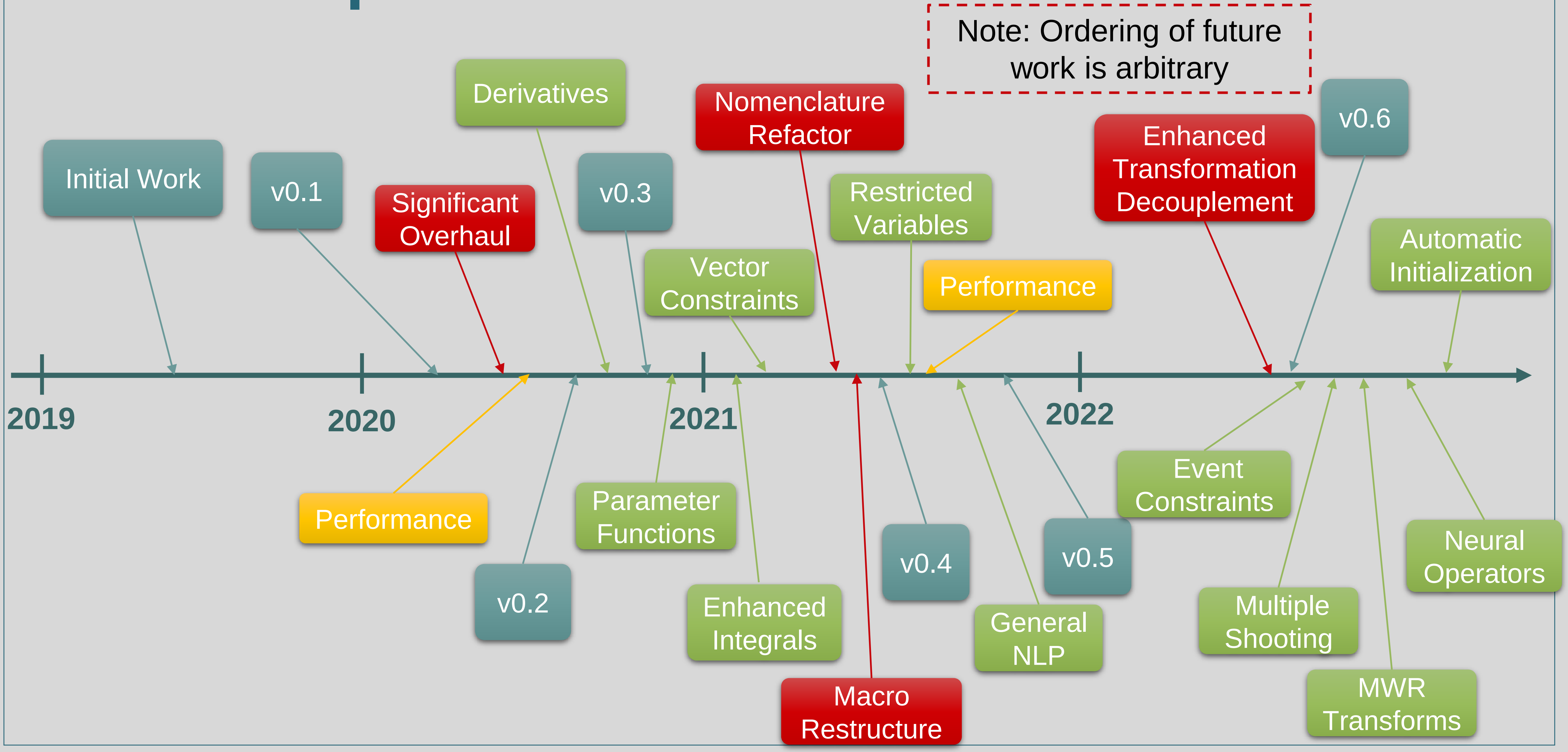
- Research
  - Sandbox InfiniteOpt problem approaches
  - Deploy your innovations → accessibility
  - Rapid comparisons



- Learning
  - High-level interface for novices
  - Features to interrogate and learn

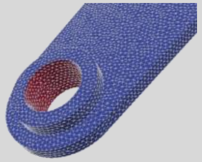
- Application
  - Offline decision making (e.g., optimal control)
  - Any InfiniteOpt problem class

Development Timeline

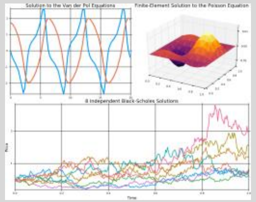


Future Development: Modeling Objects

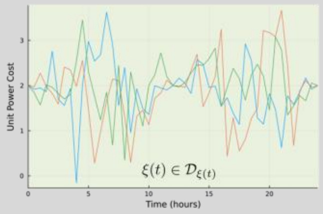
- Non-rectangular domains
  - Leverage packages like Gridap.jl



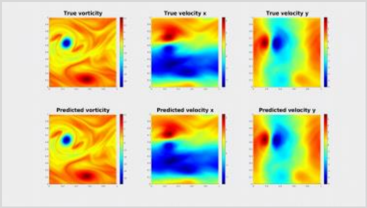
- Optimal Control Toolbox
  - Automatic initialization via DifferentialEquations.jl
  - Simulation



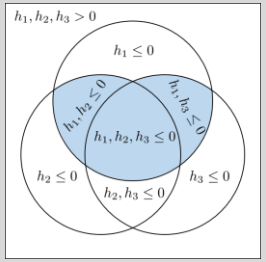
- Infinite Parameter Functions
  - Embed random fields uncertainty



- Infinite-Dimensional Surrogates
  - Embed neural operator models

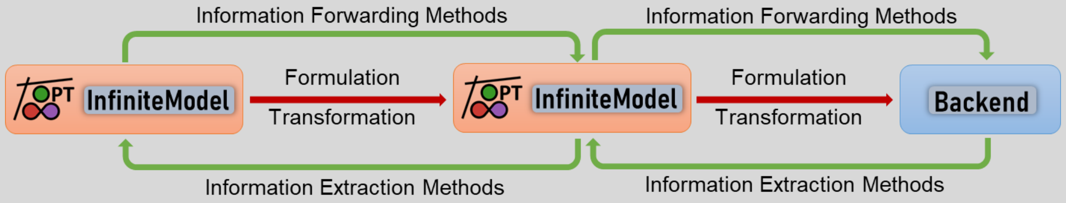


- Event Constraints
  - Generalization of chance constraints
  - Connection to generalized disjunctive programming

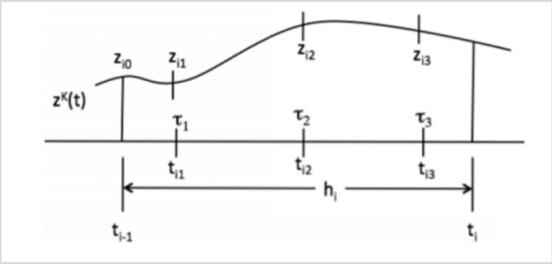


Future Development: Transformations

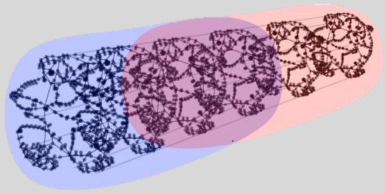
- Transformation Decouplement Rewrite
  - Fully decouple TranscriptionOpt
  - Allow arbitrary (non-JuMP) models



- Enhanced Transcription
  - Comprehensive collocation support
  - Efficient multivariate quadrature



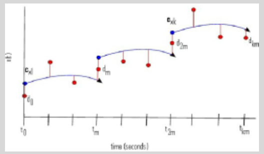
- Automatic Decomposition
  - Connection to Plasmo.jl/MadNLP.jl
  - Connection to Parapint



- Symbolic Simplification via ModelingToolkit.jl
  - Automatic order reduction

ModelingToolkit.jl

- DAE Solver Transformations
  - Multiple shooting
  - Adjoint methods via DifferentialEquations.jl



Future Development: Enhanced Nonlinear

- Leverage JuMP.jl's NLP Overhaul
  - Vector-valued NLP
  - Modular AD backends
  - Replace our `NLPExpr`s with JuMP.jl objects
  - Support black-box functions

NumFOCUS signs agreement with LANL to improve nonlinear support in JuMP

announcements · 21 Feb 2022

The JuMP Steering Committee is pleased to announce that we, through NumFOCUS, have signed an agreement with Los Alamos National Laboratory (LANL) to improve nonlinear support in JuMP.

The agreement runs until September 2023.

At a high-level, the agreement seeks to improve nonlinear support in JuMP on two key fronts:

- improving the automatic differentiation performance in JuMP by adding the ability to use different automatic differentiation backends
- improving the modeling ergonomics of JuMP by adding the ability to create nonlinear expressions using the `@constraint` and `@objective` macros, and identifying ways to add vector-valued nonlinear functions.

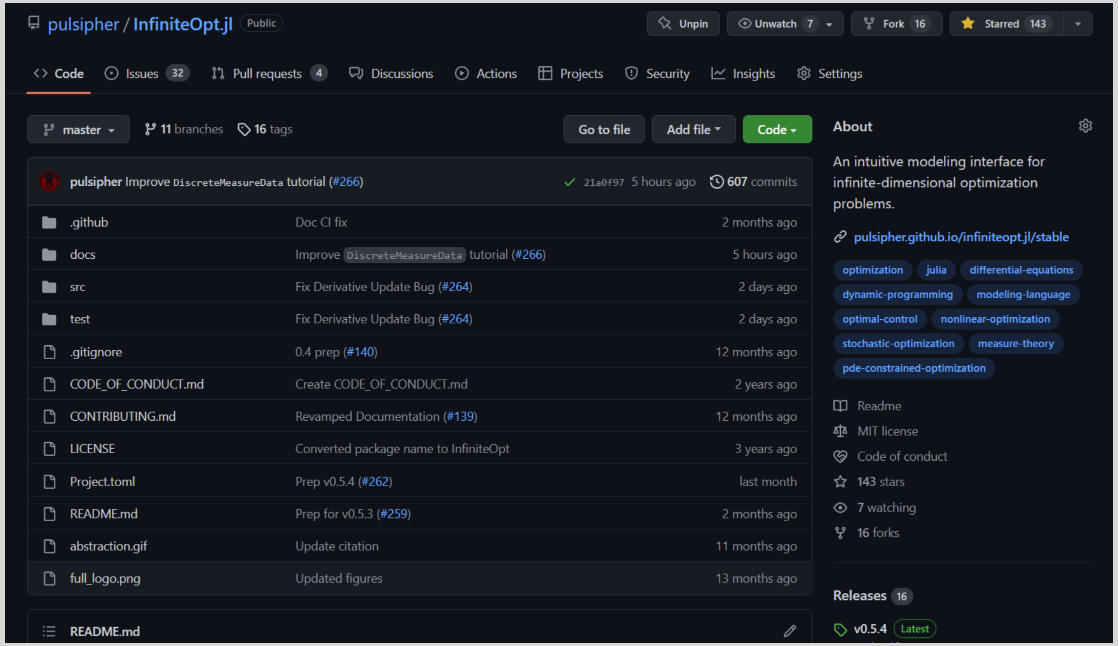
In addition, there is scope to work on and improve other aspects of nonlinear programming in JuMP.

To discuss our plans or to get involved by suggesting improvements:

- Read and comment on our work-in-progress Google doc "Developing state-of-the-art nonlinear support in JuMP"
- Join the monthly developer calls
- Join the developer chatroom for updates and an invitation to semi-regular nonlinear developer calls.

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!