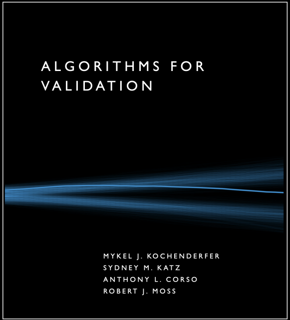
Julia in Academia

Textbooks, Stanford Courses, and the Future



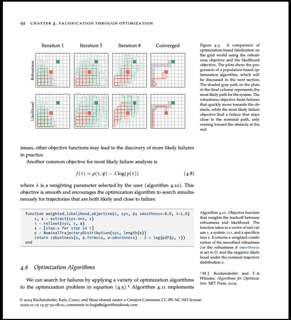

92 CHAPTER 4. FALSIFICATION THROUGH OPTIMIZATION

Figure 4.3. A comparison of optimization-based falsification on the grid world using the robustness objective and the likelihood objective. The plots show the progression of a population-based optimization algorithm, which will be discussed in the next section. The shaded gray path on the plots in the final column represents the most likely path for the system. The robustness objective finds failures that quickly move towards the obstacle, while the most likely failure objective find a failure that stays close to the nominal path, only veering toward the obstacle at the end.

issues, other objective functions may lead to the discovery of more likely failures in practice.

Another common objective for most likely failure analysis is

$$f(\tau) = \rho(\tau, \psi) - \lambda \log(p(\tau)) \tag{4.8}$$

where λ is a weighting parameter selected by the user (algorithm 4.10). This objective is smooth and encourages the optimization algorithm to search simultaneously for trajectories that are both likely and close to failure.

```julia
function weighted_likelihood_objective(x, sys, ψ; smoothness=0.0, λ=1.0)
    s, x = extract(sys.env, x)
    τ = rollout(sys, s, x)
    s = [step.s for step in τ]
    p = NominalTrajectoryDistribution(sys, length(x))
    return robustness(s, ψ.formula, w=smoothness) - λ * log(pdf(p, τ))
end
```

Algorithm 4.10. Objective function that weights the tradeoff between robustness and likelihood. The function takes in a vector of real values `x`, a system `sys`, and a specification `ψ`. It returns a weighted combination of the smoothed robustness (or the robustness if `smoothness` is set to 0) and the negative likelihood under the nominal trajectory distribution `p`.

4.6 Optimization Algorithms

We can search for failures by applying a variety of optimization algorithms to the optimization problem in equation (4.5).[4] Algorithm 4.11 implements

[4] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for Optimization*. MIT Press, 2019.

Robert Moss, PhD
Stanford University | JuliaCon 2025
mossr@cs.stanford.edu

WHOAMI

WHOAMI

Recent Stanford PhD Graduate

Used Julia throughout my research

WHOAMI

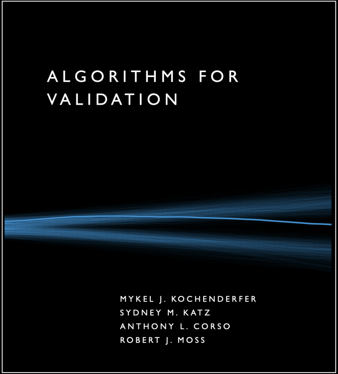
Recent Stanford PhD Graduate
Used Julia throughout my research

Prev. at MIT Lincoln Laboratory
Julia as a specification language

WHOAMI

Recent Stanford PhD Graduate
Used Julia throughout my research

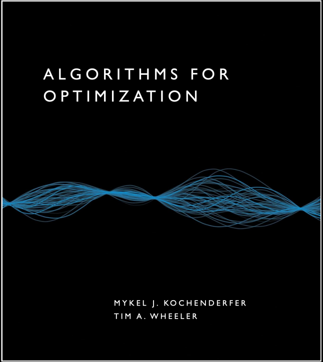
Prev. at MIT Lincoln Laboratory
Julia as a specification language



Textbook Co-Author/Head TA
Algorithms written in Julia

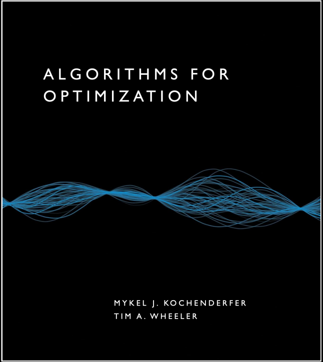
Textbooks using Julia

Textbooks using Julia

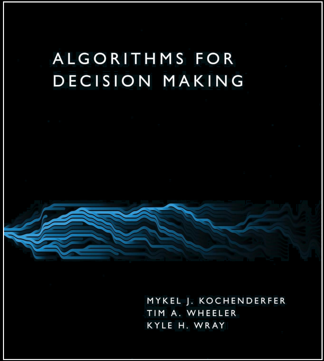


Algorithms for Optimization
MIT Press, 2019

Textbooks using Julia

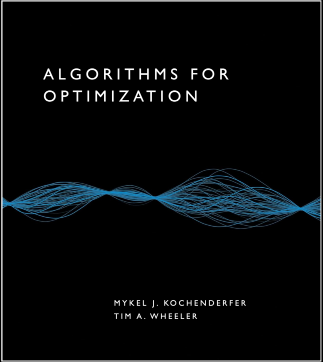


Algorithms for Optimization
MIT Press, 2019

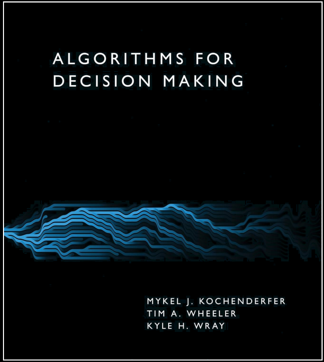


Algorithms for Decision Making
MIT Press, 2022

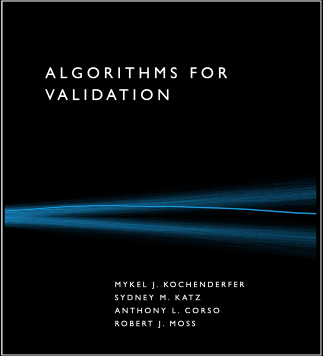
Textbooks using Julia



Algorithms for Optimization
MIT Press, 2019

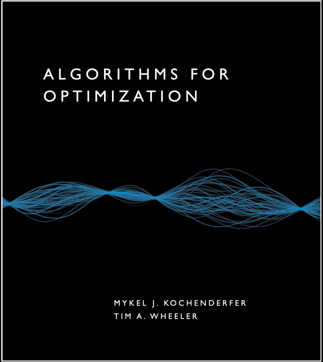


Algorithms for Decision Making
MIT Press, 2022

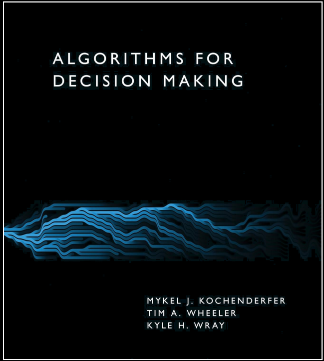


Algorithms for Validation
MIT Press, 2025

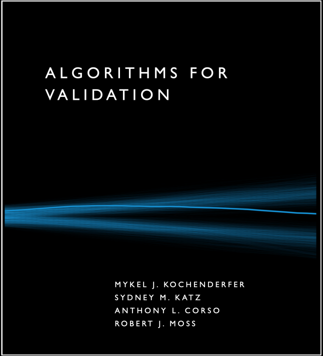
Textbooks using Julia



Algorithms for Optimization
MIT Press, 2019



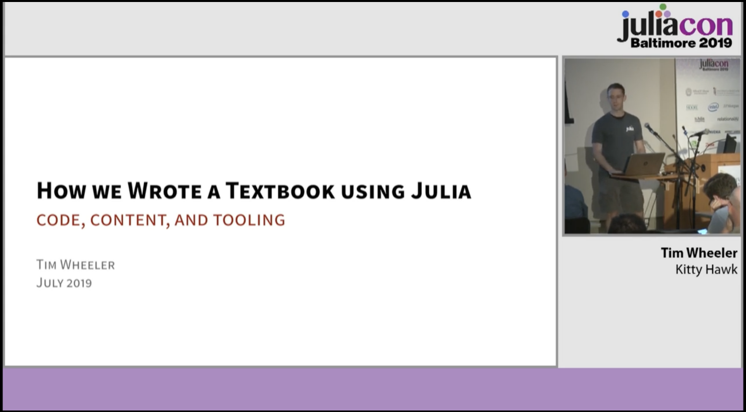
Algorithms for Decision Making
MIT Press, 2022



Algorithms for Validation
MIT Press, 2025

All PDFs available for free at `https://algorithmsbook.com`

JuliaCon 2019



How We Wrote a Textbook using Julia
Tim Wheeler, JuliaCon 2019

Textbooks with integrated Julia

Textbooks with integrated Julia

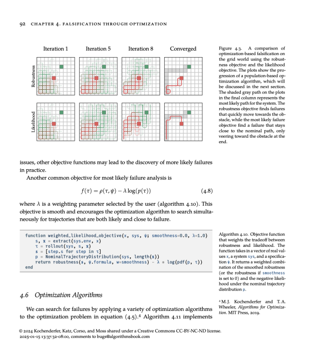


Figure 4.3. A comparison of optimization-based falsification on the grid world using the robustness objective and the likelihood objective. The plots show the progression of a population-based optimization algorithm, which will be discussed in the next section. The shaded gray path on the plots in the final column represents the most likely path for the system. The robustness objective finds failures that quickly move towards the obstacle, while the most likely failure objective find a failure that stays close to the nominal path, only veering toward the obstacle at the end.

issues, other objective functions may lead to the discovery of more likely failures in practice.

Another common objective for most likely failure analysis is

$$f(\tau) = \rho(\tau, \psi) - \lambda \log(p(\tau)) \tag{4.8}$$

where λ is a weighting parameter selected by the user (algorithm 4.10). This objective is smooth and encourages the optimization algorithm to search simultaneously for trajectories that are both likely and close to failure.

```
function weighted_likelihood_objective(x, sys, ψ; smoothness=0.0, λ=1.0)
    s, x = extract(sys.env, x)
    τ = rollout(sys, s, x)
    s = [step.s for step in τ]
    p = NominalTrajectoryDistribution(sys, length(x))
    return robustness(s, ψ.formula, w=smoothness) - λ * log(pdf(p, τ))
end
```

Algorithm 4.10. Objective function that weights the tradeoff between robustness and likelihood. The function takes in a vector of real values `x`, a system `sys`, and a specification `ψ`. It returns a weighted combination of the smoothed robustness (or the robustness if `smoothness` is set to 0) and the negative likelihood under the nominal trajectory distribution `p`.

4.6 Optimization Algorithms

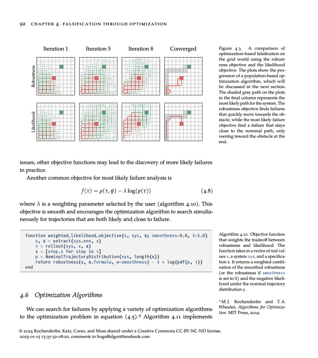
We can search for failures by applying a variety of optimization algorithms to the optimization problem in equation (4.5).[4] Algorithm 4.11 implements

[4] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for Optimization*. MIT Press, 2019.

Algorithms for Validation

Figures and Julia code

Textbooks with integrated Julia



92 CHAPTER 4. FALSIFICATION THROUGH OPTIMIZATION

Figure 4.3. A comparison of optimization-based falsification on the grid world using the robustness objective and the likelihood objective. The plots show the progression of a population-based optimization algorithm, which will be discussed in the next section. The shaded gray path on the plots in the final column represents the most likely path for the system. The robustness objective finds failures that quickly move towards the obstacle, while the most likely failure objective find a failure that stays close to the nominal path, only veering toward the obstacle at the end.

issues, other objective functions may lead to the discovery of more likely failures in practice.

Another common objective for most likely failure analysis is

$$f(\tau) = \rho(\tau, \psi) - \lambda \log(p(\tau)) \tag{4.8}$$

where λ is a weighting parameter selected by the user (algorithm 4.10). This objective is smooth and encourages the optimization algorithm to search simultaneously for trajectories that are both likely and close to failure.

```julia
function weighted_likelihood_objective(x, sys, ψ; smoothness=0.0, λ=1.0)
    s, x = extract(sys.env, x)
    τ = rollout(sys, s, x)
    s = [step.s for step in τ]
    p = NominalTrajectoryDistribution(sys, length(x))
    return robustness(s, ψ.formula, w=smoothness) - λ * log(pdf(p, τ))
end
```

Algorithm 4.10. Objective function that weights the tradeoff between robustness and likelihood. The function takes in a vector of real values `x`, a system `sys`, and a specification `ψ`. It returns a weighted combination of the smoothed robustness (or the robustness if `smoothness` is set to 0) and the negative likelihood under the nominal trajectory distribution `p`.

4.6 Optimization Algorithms

We can search for failures by applying a variety of optimization algorithms to the optimization problem in equation (4.5).[4] Algorithm 4.11 implements

[4] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for Optimization*. MIT Press, 2019.

Algorithms for Validation

Figures and Julia code

```julia
struct ImportanceSamplingEstimation
    p # nominal distribution
    q # proposal distribution
    m # number of samples
end

function estimate(alg::ImportanceSamplingEstimation, sys, ψ)
    p, q, m = alg.p, alg.q, alg.m
    τs = [rollout(sys, q) for i in 1:m]
    ps = [pdf(p, τ) for τ in τs]
    qs = [pdf(q, τ) for τ in τs]
    ws = ps ./ qs
    return mean(w * isfailure(ψ, τ) for (w, τ) in zip(ws, τs))
end
```

Example algorithm

Importance sampling estimation of failure probability

Julia integrated into LaTeX code

```latex
where the weights are $w_i = p(\tau_i) / q(\tau_i)$. These weights are sometimes referred to as \textit{importance weights}. Trajectories that are more likely under the nominal trajectory distribution have higher importance weights.

\begin{algorithm} % importance sampling estimation
\begin{juliaverbatim}
struct ImportanceSamplingEstimation
    p # nominal distribution
    q # proposal distribution
    m # number of samples
end

function estimate(alg::ImportanceSamplingEstimation, sys, ψ)
    p, q, m = alg.p, alg.q, alg.m
    τs = [rollout(sys, q) for i in 1:m]
    ps = [pdf(p, τ) for τ in τs]
    qs = [pdf(q, τ) for τ in τs]
    ws = ps ./ qs
    return mean(w * isfailure(ψ, τ) for (w, τ) in zip(ws, τs))
end
\end{juliaverbatim}
\caption{\label{alg:is_estimation} The importance sampling estimation algorithm for estimating the probability of failure. The algorithm generates \jlv{m} samples from the proposal distribution \jlv{q}. It then computes the importance weights for the samples and applies \cref{eq:is_estimator} to compute $\hat p_\text{fail}$.}
\end{algorithm}

\subsection{Optimal Proposal Distribution}
\label{sec:optimal_proposal}
The accuracy and efficiency of importance sampling approaches is highly dependent on the proposal distribution. The variance of the estimator in \cref{eq:is_estimator} is
\begin{equation}
    \label{eq:is_var}
    \text{Var}[\hat p_\text{fail}] = \frac{1}{m} \mathbb E_{\tau \sim q(\cdot)}\left[\frac{(p(\tau) \mathds 1\{\tau \notin \psi\} - q(\tau)p_\text{fail})^2}{q(\tau)}\right]
\end{equation}
In general, we want to select a proposal distribution that makes this variance low, and the optimal proposal distribution is the one that minimizes this variance.

It is evident from \cref{eq:is_var} that we can achieve a variance of zero when
\begin{equation}
    \label{eq:optimal_proposal}
    q^\ast(\tau) = \frac{p(\tau) \mathds 1\{\tau \notin \psi\}}{p_\text{fail}}
\end{equation}
This distribution corresponds to the failure distribution $p(\tau \mid \tau \notin \psi)$. As noted in \cref{cha:failure_distribution}, computing this distribution is not possible in practice since we often do not know the full set of failure trajectories and the normalizing constant $p_\text{fail}$ is the quantity we are trying to estimate. Our goal is therefore to select a proposal distribution that is as close as possible to the failure distribution.
```

LaTeX code example

Integrated Julia code using `pythontex`

Julia integrated into $\LaTeX$ code

```latex
where the weights are $w_i = p(\tau_i) / q(\tau_i)$. These weights are sometimes referred to as \textit{importance weights}. Trajectories that are more likely under the nominal trajectory distribution have higher importance weights.

\begin{algorithm} % importance sampling estimation
\begin{juliaverbatim}
struct ImportanceSamplingEstimation
    p # nominal distribution
    q # proposal distribution
    m # number of samples
end

function estimate(alg::ImportanceSamplingEstimation, sys, ψ)
    p, q, m = alg.p, alg.q, alg.m
    τs = [rollout(sys, q) for i in 1:m]
    ps = [pdf(p, τ) for τ in τs]
    qs = [pdf(q, τ) for τ in τs]
    ws = ps ./ qs
    return mean(w * isfailure(ψ, τ) for (w, τ) in zip(ws, τs))
end
\end{juliaverbatim}
\caption{\label{alg:is_estimation} The importance sampling estimation algorithm for estimating the probability of failure. The algorithm generates \jlv{m} samples from the proposal distribution \jlv{q}. It then computes the importance weights for the samples and applies \cref{eq:is_estimator} to compute $\hat p_\text{fail}$.}
\end{algorithm}

\subsection{Optimal Proposal Distribution}
\label{sec:optimal_proposal}
The accuracy and efficiency of importance sampling approaches is highly dependent on the proposal distribution. The variance of the estimator in \cref{eq:is_estimator} is
\begin{equation}
    \label{eq:is_var}
    \text{Var}[\hat p_\text{fail}] = \frac{1}{m} \mathbb E_{\tau \sim q(\cdot)}\left[\frac{(p(\tau)
    \mathds 1\{\tau \notin \psi\} - q(\tau)p_\text{fail})^2}{q(\tau)}\right]
\end{equation}
In general, we want to select a proposal distribution that makes this variance low, and the optimal proposal distribution is the one that minimizes this variance.

It is evident from \cref{eq:is_var} that we can achieve a variance of zero when
\begin{equation}
    \label{eq:optimal_proposal}
    q^\ast(\tau) = \frac{p(\tau) \mathds 1\{\tau \notin \psi\}}{p_\text{fail}}
\end{equation}
This distribution corresponds to the failure distribution $p(\tau \mid \tau \notin \psi)$. As noted in \cref{cha:failure_distribution}, computing this distribution is not possible in practice since we often do not know the full set of failure trajectories and the normalizing constant $p_\text{fail}$ is the quantity we are trying to estimate. Our goal is therefore to select a proposal distribution that is as close as possible to the failure distribution.
```

$\LaTeX$ code example

Integrated Julia code using `pythontex`

where the weights are $w_i = p(\tau_i)/q(\tau_i)$. These weights are sometimes referred to as *importance weights*. Trajectories that are more likely under the nominal trajectory distribution have higher importance weights.

```julia
struct ImportanceSamplingEstimation
    p # nominal distribution
    q # proposal distribution
    m # number of samples
end

function estimate(alg::ImportanceSamplingEstimation, sys, ψ)
    p, q, m = alg.p, alg.q, alg.m
    τs = [rollout(sys, q) for i in 1:m]
    ps = [pdf(p, τ) for τ in τs]
    qs = [pdf(q, τ) for τ in τs]
    ws = ps ./ qs
    return mean(w * isfailure(ψ, τ) for (w, τ) in zip(ws, τs))
end
```

Algorithm 7.3. The importance sampling estimation algorithm for estimating the probability of failure. The algorithm generates `m` samples from the proposal distribution `q`. It then computes the importance weights for the samples and applies equation (7.8) to compute $\hat{p}_{\text{fail}}$.

7.2.1 *Optimal Proposal Distribution*

The accuracy and efficiency of importance sampling approaches is highly dependent on the proposal distribution. The variance of the estimator in equation (7.8) is

$$\text{Var}[\hat{p}_{\text{fail}}] = \frac{1}{m}\mathbb{E}_{\tau\sim q(\cdot)}\left[\frac{(p(\tau)\mathbb{1}\{\tau \notin \psi\} - q(\tau)p_{\text{fail}})^2}{q(\tau)}\right] \tag{7.10}$$

In general, we want to select a proposal distribution that makes this variance low, and the optimal proposal distribution is the one that minimizes this variance.

It is evident from equation (7.10) that we can achieve a variance of zero when

$$q^*(\tau) = \frac{p(\tau)\mathbb{1}\{\tau \notin \psi\}}{p_{\text{fail}}} \tag{7.11}$$

This distribution corresponds to the failure distribution $p(\tau \mid \tau \notin \psi)$. As noted in chapter 6, computing this distribution is not possible in practice since we often do not know the full set of failure trajectories and the normalizing constant p_{fail} is the quantity we are trying to estimate. Our goal is therefore to select a proposal distribution that is as close as possible to the failure distribution.

Compiled $\LaTeX$ PDF

Julia algorithms and $\LaTeX$ math

Open-Source Textbook Template Repository

Open-Source Textbook Template Repository

tufte_algorithms_book

A template for textbooks in the same style as Algorithms for Optimization

TeX ★ 368 69

github.com/sisl/tufte_algorithms_book

Open-Source Slides Repo (beamer + pythontex)

Open-Source Slides Repo (beamer + pythontex)

julia-in-academia

JuliaCon 2025 presentation on "Julia in Academia"

TeX

github.com/mossr/julia-in-academia

Open-Source Slides Repo (beamer + pythontex)

julia-in-academia

JuliaCon 2025 presentation on "Julia in Academia"

TeX

github.com/mossr/julia-in-academia

julia-tufte-beamer

Tufte-style beamer template with Julia integration

TeX ★ 136 20

github.com/mossr/julia-tufte-beamer

Julia for Textbooks: The Good

Julia for Textbooks: The Good

- Concise algorithm descriptions (fits within single page)

Julia for Textbooks: The Good

- Concise algorithm descriptions (fits within single page)
- Multiple dispatch makes it easy to design common interface

Julia for Textbooks: The Good

- Concise algorithm descriptions (fits within single page)
- Multiple dispatch makes it easy to design common interface
- Auto-differentiation allows for clean algorithms that just work

Julia for Textbooks: The Good

- Concise algorithm descriptions (fits within single page)
- Multiple dispatch makes it easy to design common interface
- Auto-differentiation allows for clean algorithms that just work
- Full Unicode support means math matches code (e.g., λ in math and `λ` in code)

Julia for Textbooks: The Good

- Concise algorithm descriptions (fits within single page)
- Multiple dispatch makes it easy to design common interface
- Auto-differentiation allows for clean algorithms that just work
- Full Unicode support means math matches code (e.g., λ in math and `λ` in code)

Shout-out to the following packages:
`Distributions.jl`, `LazySets.jl`, `IntervalArithmetic.jl`, `Flux.jl`, `Optim.jl`, and `JuMP.jl`

Julia for Textbooks: The Not-So-Good

Julia for Textbooks: The Not-So-Good

- Somewhat new to readers: Requires learning Julia in the process

Julia for Textbooks: The Not-So-Good

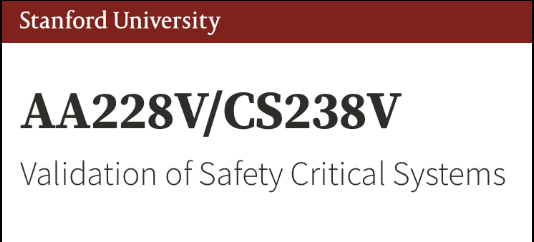
- Somewhat new to readers: Requires learning Julia in the process
- Potential obscure idioms: broadcasting, anonymous functions, multiple dispatch

Julia for Textbooks: The Not-So-Good

- Somewhat new to readers: Requires learning Julia in the process
- Potential obscure idioms: broadcasting, anonymous functions, multiple dispatch
- Textbook tooling: Needed to be built out from scratch

Julia in the Classroom

Julia in the Classroom



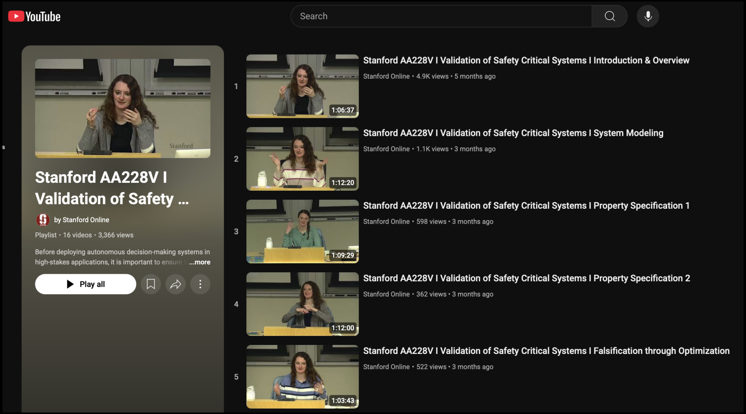
Grad-level course at Stanford

Follows *Algorithms for Validation* textbook

First offered in Winter 2025

Lectures

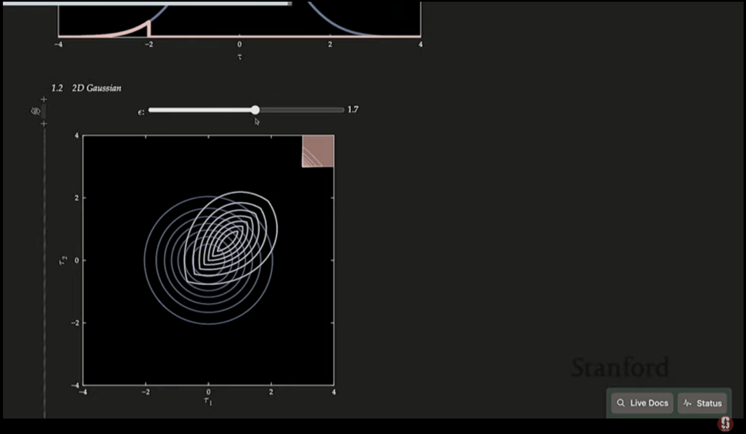
Lectures



All lectures available on YouTube

Led by Sydney Katz, textbook co-author and award-winning lecturer

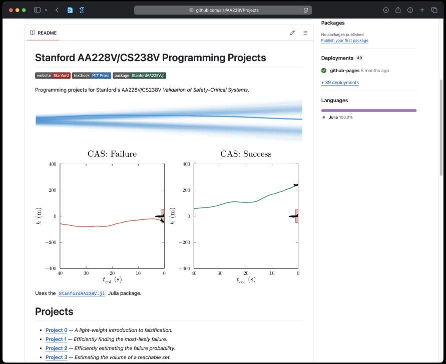
Interactive Lectures



Interactive lectures using `Pluto.jl`

Assignments in Julia

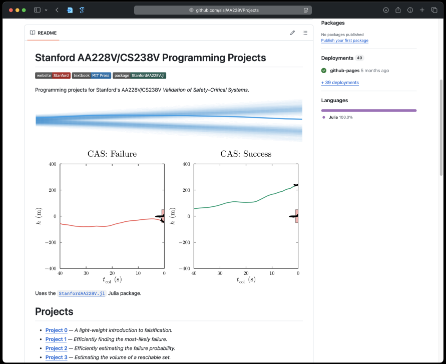
Assignments in Julia



Assignments repository

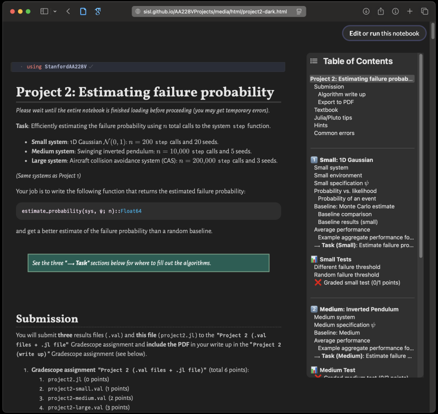
Student-facing code

Assignments in Julia



Assignments repository

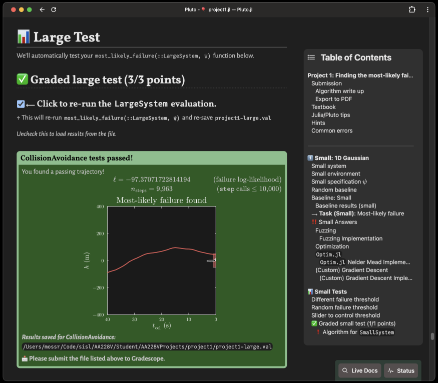
Student-facing code



Pluto assignments

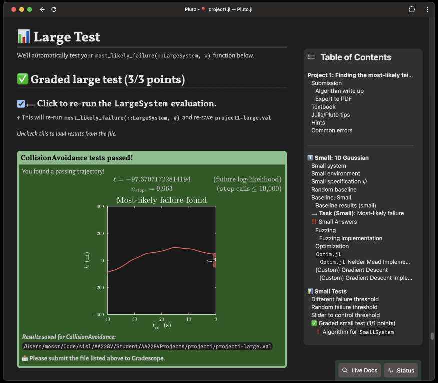
Includes local tests

Assignments in Julia



Interactive Pluto tests
Get feedback instantly

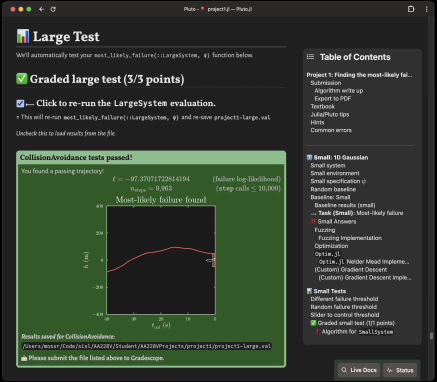
Assignments in Julia



Interactive Pluto tests
Get feedback instantly

- Separated assignment code from core library (`StanfordAA228V.jl`)

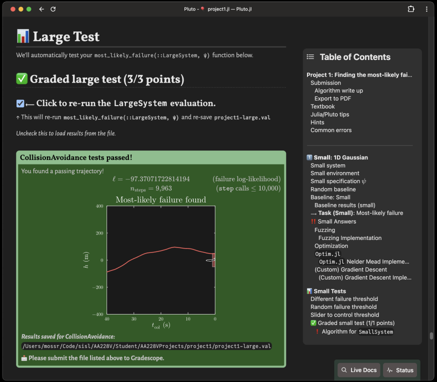
Assignments in Julia



Interactive Pluto tests
Get feedback instantly

- Separated assignment code from core library (`StanfordAA228V.jl`)
- Local tests exactly match graded tests

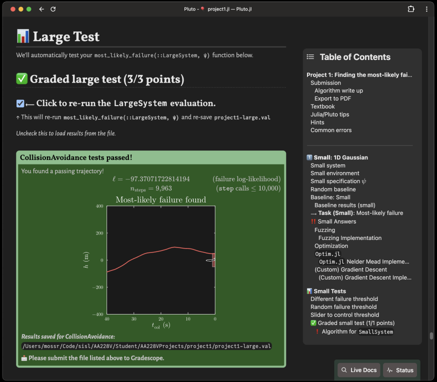
Assignments in Julia



Interactive Pluto tests
Get feedback instantly

- Separated assignment code from core library (`StanfordAA228V.jl`)
- Local tests exactly match graded tests
- Interactive nature allows students to play with their algorithms

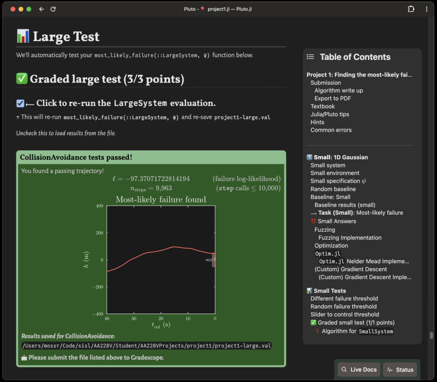
Assignments in Julia



Interactive Pluto tests
Get feedback instantly

- Separated assignment code from core library (`StanfordAA228V.jl`)
- Local tests exactly match graded tests
- Interactive nature allows students to play with their algorithms
- No "hidden state" that may confuse students

Assignments in Julia

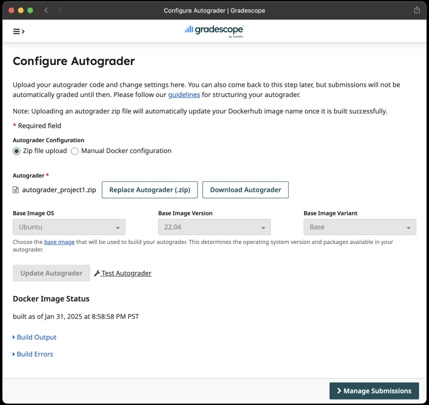


Interactive Pluto tests
Get feedback instantly

- Separated assignment code from core library (`StanfordAA228V.jl`)
- Local tests exactly match graded tests
- Interactive nature allows students to play with their algorithms
- No "hidden state" that may confuse students
- Open-source nature requires clever obfuscation of solution code

Grading Assignments

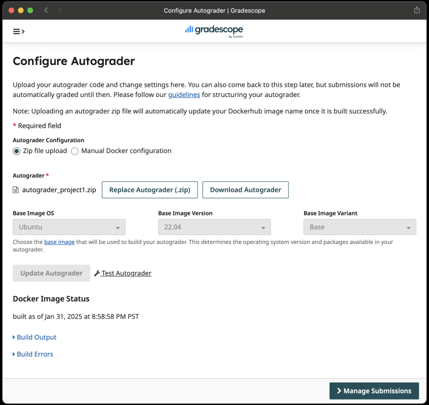
Grading Assignments



Autograde via Gradescope

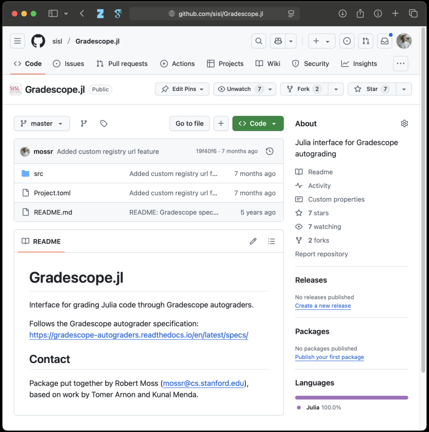
Students upload Pluto notebook

Grading Assignments



Autograde via Gradescope

Students upload Pluto notebook



Light-weight `Gradescope.jl` package

Manage deps and create autograders

Future use of Julia in Academia

Future use of Julia in Academia

- Teach fundamentals like probability and statistics in pure Julia

Future use of Julia in Academia

- Teach fundamentals like probability and statistics in pure Julia
- Unify curriculum across compounding courses

Future use of Julia in Academia

- Teach fundamentals like probability and statistics in pure Julia
- Unify curriculum across compounding courses
- Build cohesive teaching materials using Pluto (e.g., through JuliaAcademy)

Future use of Julia in Academia

- Teach fundamentals like probability and statistics in pure Julia
- Unify curriculum across compounding courses
- Build cohesive teaching materials using Pluto (e.g., through JuliaAcademy)
- Build general grading frameworks to work across courses

Future use of Julia in Academia

- Teach fundamentals like probability and statistics in pure Julia
- Unify curriculum across compounding courses
- Build cohesive teaching materials using Pluto (e.g., through JuliaAcademy)
- Build general grading frameworks to work across courses
- Compile Julia code to binaries to avoid obfuscation

Future use of Julia in Academia

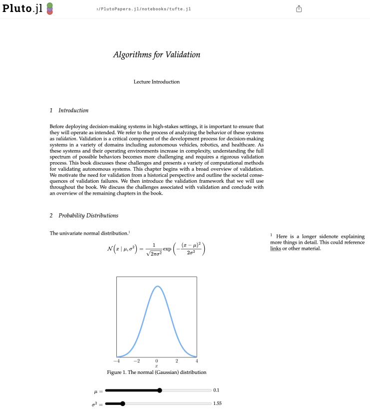
- Teach fundamentals like probability and statistics in pure Julia
- Unify curriculum across compounding courses
- Build cohesive teaching materials using Pluto (e.g., through JuliaAcademy)
- Build general grading frameworks to work across courses
- Compile Julia code to binaries to avoid obfuscation
- Write interactive research papers in Julia/Pluto

PlutoPapers.jl

PlutoPapers.jl

What if we could interact directly with the code in research papers?

PlutoPapers.jl



Pluto.jl

/PlutoPapers.jl/notebooks/tufte.jl

Algorithms for Validation

Lecture Introduction

1 *Introduction*

Before deploying decision-making systems in high-stakes settings, it is important to ensure that they will operate as intended. We refer to the process of analyzing the behavior of these systems as *validation*. Validation is a critical component of the development process for decision-making systems in a variety of domains including autonomous vehicles, robotics, and healthcare. As these systems and their operating environments increase in complexity, understanding the full spectrum of possible behaviors becomes more challenging and requires a rigorous validation process. This book discusses these challenges and presents a variety of computational methods for validating autonomous systems. This chapter begins with a broad overview of validation. We motivate the need for validation from a historical perspective and outline the societal consequences of validation failures. We then introduce the validation framework that we will use throughout the book. We discuss the challenges associated with validation and conclude with an overview of the remaining chapters in the book.

2 *Probability Distributions*

The univariate normal distribution.[1]

$$\mathcal{N}\left(x \mid \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

[1] Here is a longer sidenote explaining more things in detail. This could reference links or other material.

Figure 1. The normal (Gaussian) distribution

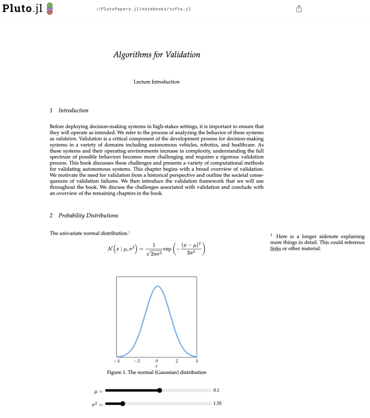
$\mu =$ 0.1

$\sigma^2 =$ 1.55

Interactive research papers

Run code directly in the paper

PlutoPapers.jl


Pluto.jl ⁄PlutoPapers.jl/notebooks/tufte.jl

Algorithms for Validation

Lecture Introduction

1 *Introduction*

Before deploying decision-making systems in high-stakes settings, it is important to ensure that they will operate as intended. We refer to the process of analyzing the behavior of these systems as *validation*. Validation is a critical component of the development process for decision-making systems in a variety of domains including autonomous vehicles, robotics, and healthcare. As these systems and their operating environments increase in complexity, understanding the full spectrum of possible behaviors becomes more challenging and requires a rigorous validation process. This book discusses these challenges and presents a variety of computational methods for validating autonomous systems. This chapter begins with a broad overview of validation. We motivate the need for validation from a historical perspective and outline the societal consequences of validation failures. We then introduce the validation framework that we will use throughout the book. We discuss the challenges associated with validation and conclude with an overview of the remaining chapters in the book.

2 *Probability Distributions*

The univariate normal distribution.[1]

$$\mathcal{N}\left(x \mid \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

[1] Here is a longer sidenote explaining more things in detail. This could reference links or other material.

Figure 1. The normal (Gaussian) distribution

$\mu =$ 0.1

$\sigma^2 =$ 1.55

Interactive research papers

Run code directly in the paper

PlutoPapers.jl

Interactive and LaTeX-styled papers in Pluto.jl

Julia ★ 5

https://github.com/mossr/PlutoPapers.jl

Thank you!

Questions?

mossr@cs.stanford.edu