

# Julia in Academia

## TEXTBOOKS, STANFORD COURSES, AND THE FUTURE

### ALGORITHMS FOR VALIDATION



MYKEL J. KOCHENDERFER  
SYDNEY M. KATZ  
ANTHONY L. CORSO  
ROBERT J. MOSS

92 CHAPTER 4. PREDICTION THROUGH OPTIMIZATION

Figure 4.3 A comparison of optimization-based localization on a 10x10 grid. The top row shows the initial objective and the likelihood function for the first iteration of a population-based optimization algorithm which will be discussed in the next section. The shaded gray path on the plate is the most likely path for the system. The bottom row shows the final objective and likelihood function after 8 iterations. The algorithm quickly moves towards the objective and a failure that stays close to the nominal path, only moving toward the outside at the end.



issues, other objective functions may lead to the discovery of more likely failure trajectories.

Another common objective for event likely failure analysis is

$$f(\mathbf{v}) = \mu(\mathbf{v}, \theta) - \lambda \log(\mathcal{C}(\mathbf{v})) \quad (4.8)$$

where  $\lambda$  is a weighting parameter selected by the user (Algorithm 4.4c). This objectives is search and encourages the optimization algorithm to search simultaneously for trajectories that are both likely and close to failure.

```
function weighted_likelihood_objective(v, mu, w, w_likelihood, l, l-1, d)
    v = v[1:l]
    l = l+1
    l-1 = l-1 + d
    p = bimodal_trajectory_probability(mu, logit(l))
    return w_likelihood(l, v, mu) + w * logit(l) * p
end
```

Algorithm 4.4c Objective function that weighs the likelihood between robustness and likelihood. The function takes in a vector  $v$ , a mean  $\mu$ , a weight  $w$ , a likelihood function  $w_{likelihood}$ , a lower bound  $l$ , an upper bound  $l-1$ , and a dimension  $d$ . The function returns the weighted robustness and likelihood. The likelihood function is set to 0 if the lower likelihood is set to 0 and the negative likelihood is set to the nominal trajectory distribution.

### 4.6 Optimization Algorithms

We can search for failures by applying a variety of optimization algorithms to the optimization problems in equation (4.5).<sup>14</sup> Algorithm 4.5 implements

© 2014 Kochenderfer, Katz, Corso, and Moss shared under a Creative Commons CC-BY-NC-ND license.  
<http://www.cs.cmu.edu/~kochenderfer/>, comments to [koch@cs.cmu.edu](mailto:koch@cs.cmu.edu)

ROBERT MOSS, PHD  
STANFORD UNIVERSITY | JULIACON 2025  
[mossr@cs.stanford.edu](mailto:mossr@cs.stanford.edu)

WHOAMI

# WHOAMI



Recent Stanford PhD Graduate  
Used Julia throughout my research

# WHOAMI



A video thumbnail featuring a man in a grey shirt and khaki pants standing in front of a chalkboard, gesturing with his hands. The thumbnail is framed by a blue border. Below the thumbnail, the text reads "Robert Moss" and "Using Julia as a Specification Language for the Next-Generation Airborne Collision Avoidance System". To the left of the thumbnail, the Julia logo and "JuliaCon 2015" are visible.

Prev. at MIT Lincoln Laboratory  
Julia as a specification language

Recent Stanford PhD Graduate  
Used Julia throughout my research

# WHOAMI



Recent Stanford PhD Graduate  
Used Julia throughout my research

A video thumbnail from JuliaCon 2015. It shows a man in a grey shirt and khaki pants standing at a podium, gesturing with his hands. The background includes a chalkboard and a wooden panel. The thumbnail has a blue border. Below the thumbnail, the text reads: "Robert Moss" and "Using Julia as a Specification Language for the Next-Generation Airborne Collision Avoidance System".

Prev. at MIT Lincoln Laboratory  
Julia as a specification language

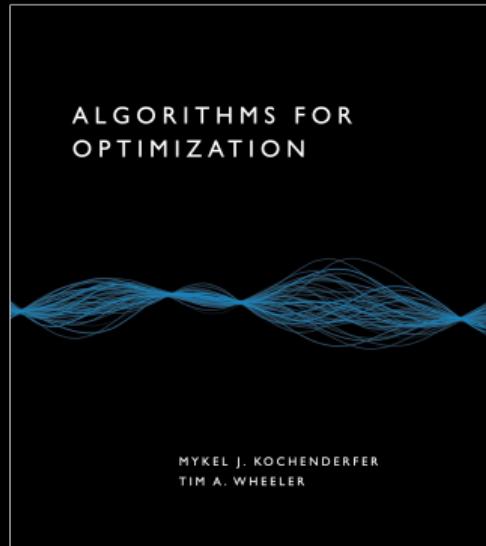
## ALGORITHMS FOR VALIDATION

MYKEL J. KOCHENDERFER  
SYDNEY M. KATZ  
ANTHONY L. CORSO  
ROBERT J. MOSS

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



MYKEL J. KOCHENDERFER  
TIM A. WHEELER

ALGORITHMS FOR  
DECISION MAKING



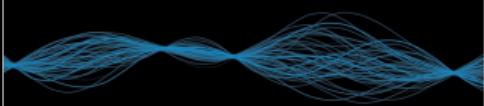
MYKEL J. KOCHENDERFER  
TIM A. WHEELER  
KYLE H. WRAY

Algorithms for Optimization  
MIT Press, 2019

Algorithms for Decision Making  
MIT Press, 2022

# TEXTBOOKS USING JULIA

ALGORITHMS FOR  
OPTIMIZATION



MYKEL J. KOCHENDERFER  
TIM A. WHEELER

Algorithms for Optimization  
MIT Press, 2019

ALGORITHMS FOR  
DECISION MAKING



MYKEL J. KOCHENDERFER  
TIM A. WHEELER  
KYLE H. WRAY

Algorithms for Decision Making  
MIT Press, 2022

ALGORITHMS FOR  
VALIDATION



MYKEL J. KOCHENDERFER  
SYDNEY M. KATZ  
ANTHONY L. CORSO  
ROBERT J. MOSS

Algorithms for Validation  
MIT Press, 2025

# TEXTBOOKS USING JULIA

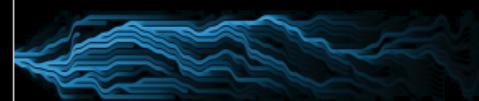
ALGORITHMS FOR  
OPTIMIZATION



MYKEL J. KOCHENDERFER  
TIM A. WHEELER

Algorithms for Optimization  
MIT Press, 2019

ALGORITHMS FOR  
DECISION MAKING



MYKEL J. KOCHENDERFER  
TIM A. WHEELER  
KYLE H. WRAY

Algorithms for Decision Making  
MIT Press, 2022

ALGORITHMS FOR  
VALIDATION



MYKEL J. KOCHENDERFER  
SYDNEY M. KATZ  
ANTHONY L. CORSO  
ROBERT J. MOSS

Algorithms for Validation  
MIT Press, 2025

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

# JULIACON 2019



The slide features a white background with a purple footer bar at the bottom. In the center, there is a video frame showing a presentation. The video frame has a white header bar with the JuliaCon Baltimore 2019 logo. The main content of the video frame shows a man standing at a podium with a laptop, speaking to an audience. The video frame is captioned with "Tim Wheeler" and "Kitty Hawk". To the left of the video frame, there is text: "HOW WE WROTE A TEXTBOOK USING JULIA" in bold black capital letters, and "CODE, CONTENT, AND TOOLING" in red capital letters below it. At the bottom left of the slide, there is smaller text: "TIM WHEELER" and "JULY 2019".

**HOW WE WROTE A TEXTBOOK USING JULIA**  
**CODE, CONTENT, AND TOOLING**

TIM WHEELER  
JULY 2019

**Tim Wheeler**  
Kitty Hawk

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

## TEXTBOOKS WITH INTEGRATED JULIA

# TEXTBOOKS WITH INTEGRATED JULIA

§8 CHAPTER 4: FALSIFICATION THROUGH OPTIMIZATION

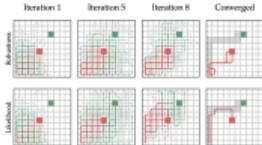


Figure 4.3. A comparison of optimization-based falsification on the grid world using the robustness objective and the weighted robustness objective. The plots show the progress of the optimization-based falsification algorithm, which will be discussed in the next section. The first three columns in each row in the final column represents the robustness objective, while the last column represents the weighted robustness objective (this follows the convention of the previous section). The robustness objective finds failures that quickly move towards the obstacles, while the weighted robustness objective find a failure that stays close to the target and only moves toward the obstacles 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

$$f(\tau) = p(\tau, \eta) - \lambda \log(p(\tau)) \quad (4.8)$$

where  $\lambda$  is a weighting parameter selected by the user (algorithm 4.10). This objective is search and converges the optimization algorithm to search stimulus ready for trajectories that are both likely and close to failure.

```
function weighted_likelihood_objective(x, xref, w1, smoothness, d, N, 2-1.0)
    y = zeros(N)
    v = rollnorm(xref, v, x)
    c = 1.0e-10
    for i=1:N
        x = [x[i]; v]
        p = logpdf(mvnrnd(xref, cov(x)))
        return smoothness(x, w, formula, w1) + -1 * log(pdf(x, c))
    end
end
```

## 4.6 Optimization Algorithms

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

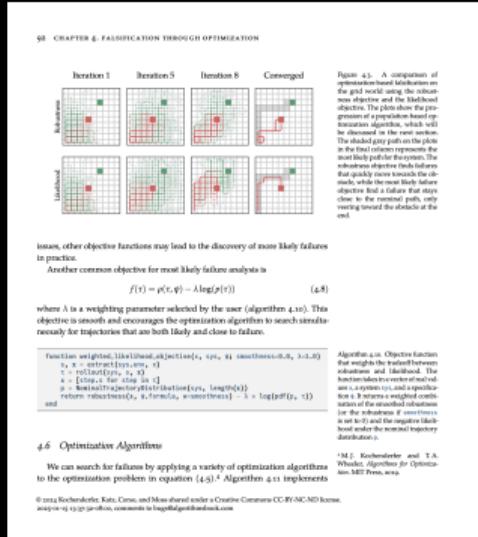
© 2012 Kochenderfer, Kate, Corra, and Mooney under a Creative Commons CC-BY-NC-ND license.  
aaron-01-13-12-jarrell, comments to log@jaguar1.us

Algorithm 4.11. Objective function that weights the robustness between solutions and likelihood. The function takes a solution  $x$ , a reference trajectory  $x_{ref}$ , and a specification  $v$ . It returns a weighted combination of the robustness measure for the solution if  $v$  is robust, or a weighted combination of the likelihood of the solution under the normal trajectory distribution.

<sup>4</sup>M. J. Kochenderfer and T.A. Wheeler, *Algorithms for Optimization*. MIT Press, 2020.

## Algorithms for Validation Figures and Julia code

# TEXTBOOKS WITH INTEGRATED JULIA



## Algorithms for Validation Figures and Julia code

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

## Example algorithm

Importance sampling estimation of failure probability

# JULIA INTEGRATED INTO L<sup>A</sup>T<sub>E</sub>X CODE

```
368 where the weights are  $w_{i,l} = p(x_{t+1}) / q(x_{t+1})$ . These weights are sometimes referred to as importance weights. Trajectories that are more likely under the nominal trajectory distribution have  
higher importance weights.  
369  
370 %begin{algorithm} % importance sampling estimation  
371 %begin{juliaverbatim}  
372 struct ImportanceSamplingEstimation  
373     p # nominal distribution  
374     q # proposal distribution  
375     n # number of samples  
376 end  
377  
378 function estimate(p,q;isproposalSamplingEstimation,sys,v)  
379     b, u = a_bias(b, u), a_bias  
380     rs = [rollout(sys, q) for i in 1:n]  
381     ps = [pdf(p, r) for r in rs]  
382     qs = [pdf(q, r) for r in rs]  
383     ws = ps ./ qs  
384     return mean(ws * isfailure(b, r) for (b, r) in zip(ws, rs))  
385 end  
386 %end{juliaverbatim}  
387 %caption{\texttt{IsProposalSamplingEstimation}} The importance sampling estimation algorithm for estimating the  
probability of failure. The algorithm generates  $\mathcal{N}(v|0)$  samples from the proposal distribution  $\mathcal{N}(v|q)$ . It then computes the importance weights for the samples and applies \texttt{Xref{eq}{is\_estimator}} to compute  
Xref{eq}{is\_estimator}{fail}.  
388 %end{algorithm}  
389  
390 %subsection{Optimal Proposal Distribution}  
391 %label{SectionOptimal_Proposal}  
392 The accuracy and efficiency of importance sampling approaches is highly dependent on the proposal  
distribution. The variance of the estimator in \texttt{Xref{eq}{is_estimator}} is  
393 %begin{equation}  
394 \text{Var}(\text{Estimate}) = \frac{1}{N} \sum_{i=1}^N \left( \frac{p(x_i)}{q(x_i)} - \frac{p(x_i)}{q(x_i)} \right)^2 \text{Var}(q(x_i))  
395 %end{equation}  
396 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.  
397  
398 It is evident from \texttt{Xref{eq}{is_var}} that we can achieve a variance of zero when  
399 %begin{equation}  
400 \text{Var}(q(x_i)) = 0 \Leftrightarrow p(x_i) = q(x_i) \forall i \in \{1, \dots, N\}   
401 %end{equation}  
402 This distribution corresponds to the failure distribution  $p_f(v) / q(v)$ . As noted in  
403 \texttt{Xref{char}{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  $\int p_f(v) dv$  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.
```

## L<sup>A</sup>T<sub>E</sub>X code example

Integrated Julia code using [pythontex](#)

# JULIA INTEGRATED INTO LATEX CODE

```

368 where the weights are  $w_i = p(x_{t+1}) / q(x_{t+1})$ . These weights are sometimes referred to as xtext (importance weights). Trajectories that are more likely under the nominal trajectory distribution have higher importance weights.
369
370 begin(algorithm) % importance sampling estimation
371   begin(JuliaVerbatim)
372     struct ImportanceSamplingEstimation
373       p # nominal distribution
374       q # proposal distribution
375       n # number of samples
376     end
377
378     function estimate(alg:ImportanceSamplingEstimation, sys, q)
379       u, w = alg.u, alg.w, alg.q
380       rs = [(rollout(q, q) for i in 1:n]
381       ps = [pdf(q, r) for r in rs]
382       qs = [pdf(q, r) for r in rs]
383       ws = ps ./ qs
384       ws = ws ./ sum(ws)
385       return measure(w * isfailure(u, r) for (u, r) in zip(ws, rs))
386     end
387   end(JuliaVerbatim)
388   caption(alg:ImportanceSamplingEstimation) The importance sampling estimation algorithm for estimating the probability of failure. The algorithm generates n samples from the proposal distribution q/q( $\cdot$ ). It then computes the importance weights for the samples and applies xref{eq:is_estimator} to compute xref{xtext}(fail).
389   end(algorithm)
390
391   subsection(Optimal Proposal Distribution)
392   label(OptimalProposal)
393   The accuracy and efficiency of importance sampling approaches is highly dependent on the proposal distribution and the variance of the estimator in xref{eq:is_estimator} is
394   begin(OptimalProposal)
395     label(eq_is_var)
396     xtext(Var){what p.xtext(fail)} = xref{t1}(e) * what E[(xtext(w)) * (xtext(w))'] / left(\frac{p(x_{t+1})}{q(x_{t+1})} * what E[(xtext(w)) * (xtext(w))'])^2xref{t1}(e) * what E[(xtext(w)) * (xtext(w))'] / left(\frac{p(x_{t+1})}{q(x_{t+1})} * what E[(xtext(w)) * (xtext(w))'])^2
397     end(OptimalProposal)
398     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.
399
400 It is evident from xref{eq:is_var} that we can achieve a variance of zero when
401 begin(Equation)
402   label(eq_optimal_proposal)
403   
$$q(x_{t+1}) = \frac{p(x_{t+1})}{q(x_{t+1})} \cdot \frac{q(x_{t+1})}{p(x_{t+1})} = \frac{p(x_{t+1})}{q(x_{t+1})}$$

404   end(Equation)
405   This distribution corresponds to the failure distribution p(x_{t+1} | x_{t+1} \in \text{failure}). As noted in xref{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(x_{t+1} | \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  $w_i = p(x_i)/q(x_i)$ . These weights are sometimes referred to as `xtext` (importance weights). Trajectories that are more likely under the nominal trajectory distribution have higher importance weights.

**Algorithm 7.3.** The importance sampling estimation algorithm for estimating the probability of failure. The algorithm generates `n` samples from the proposal distribution `q`. It then computes the importance weights for the samples and applies equation (7.9) to compute `xref{xtext}(fail)`.

7.2. IMPORTANCE SAMPLING 147

**7.2.2 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

$$\text{Var}[\hat{p}_{\text{fail}}] = \frac{1}{n} \mathbb{E}_{i \sim q(\cdot)} \left[ \frac{(p(x_i) / q(x_i) - \bar{w})^2}{q(x_i)} \right] \quad (7.10)$$

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 (7.9) that we can achieve a variance of zero when

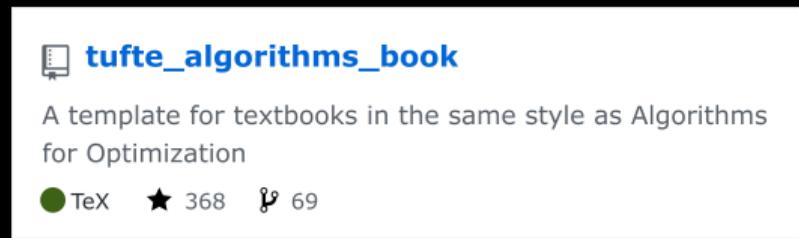
$$q^*(x) = \frac{p(x)}{\int p(x) / q(x) dx} \quad (7.11)$$

This distribution corresponds to the failure distribution  $p(x' | x \in \text{fail})$ . 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  $p(x_{t+1} | \text{fail})$  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.

Compiled LATEX PDF  
Julia algorithms and LATEX math

# OPEN-SOURCE TEXTBOOK TEMPLATE REPOSITORY

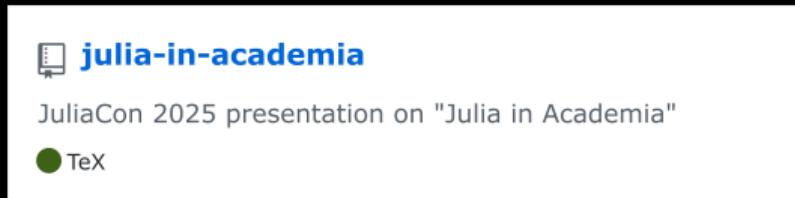
# OPEN-SOURCE TEXTBOOK TEMPLATE REPOSITORY



[github.com/sisl/tufte\\_algorithms\\_book](https://github.com/sisl/tufte_algorithms_book)

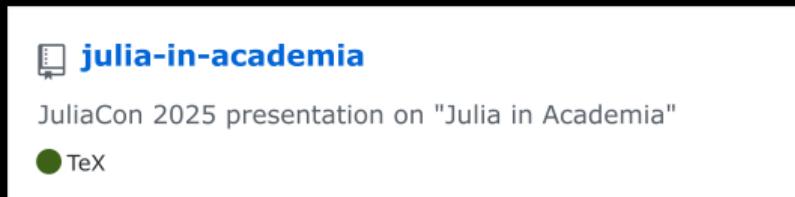
OPEN-SOURCE SLIDES REPO (`BEAMER` + `PYTHONTEX`)

# OPEN-SOURCE SLIDES REPO (BEAMER + PYTHONTEX)

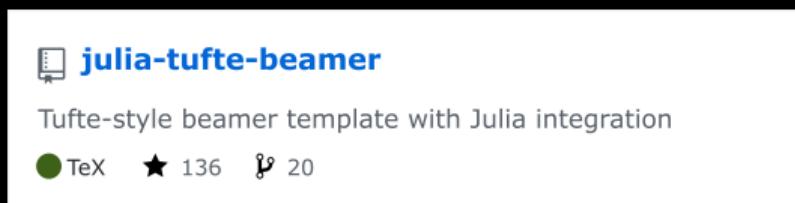


[github.com/mossr/julia-in-academia](https://github.com/mossr/julia-in-academia)

# OPEN-SOURCE SLIDES REPO (BEAMER + PYTHONTEX)



[github.com/mossr/julia-in-academia](https://github.com/mossr/julia-in-academia)



[github.com/mossr/julia-tufte-beamer](https://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.,  $\lambda$  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.,  $\lambda$  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

Stanford University

**AA228V/CS238V**

Validation of Safety Critical Systems

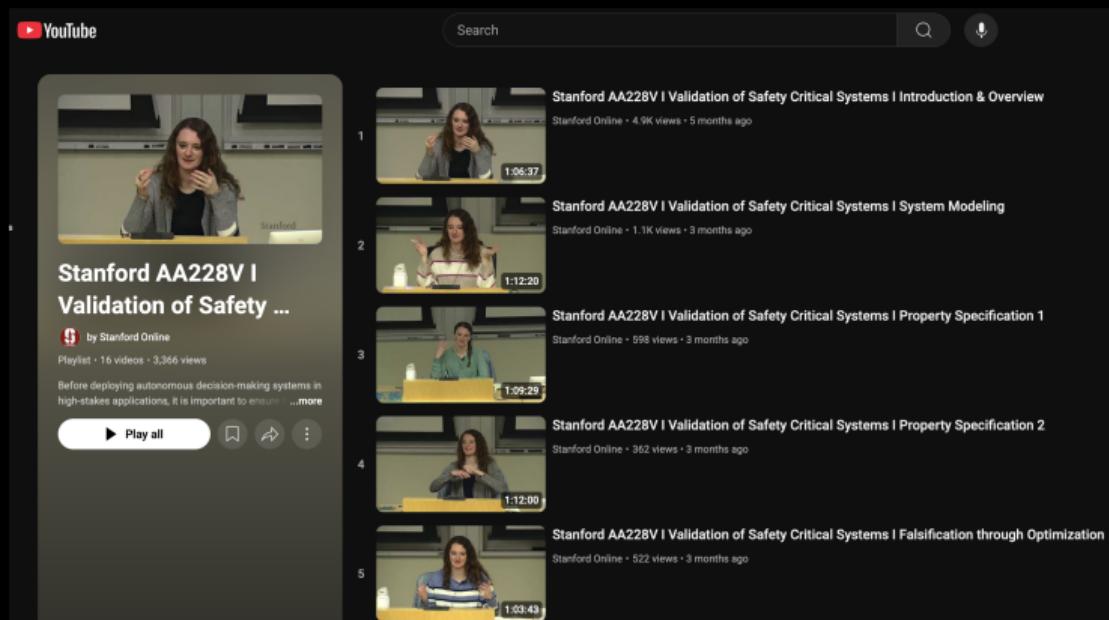
Grad-level course at Stanford

Follows *Algorithms for Validation* textbook

First offered in Winter 2025

# LECTURES

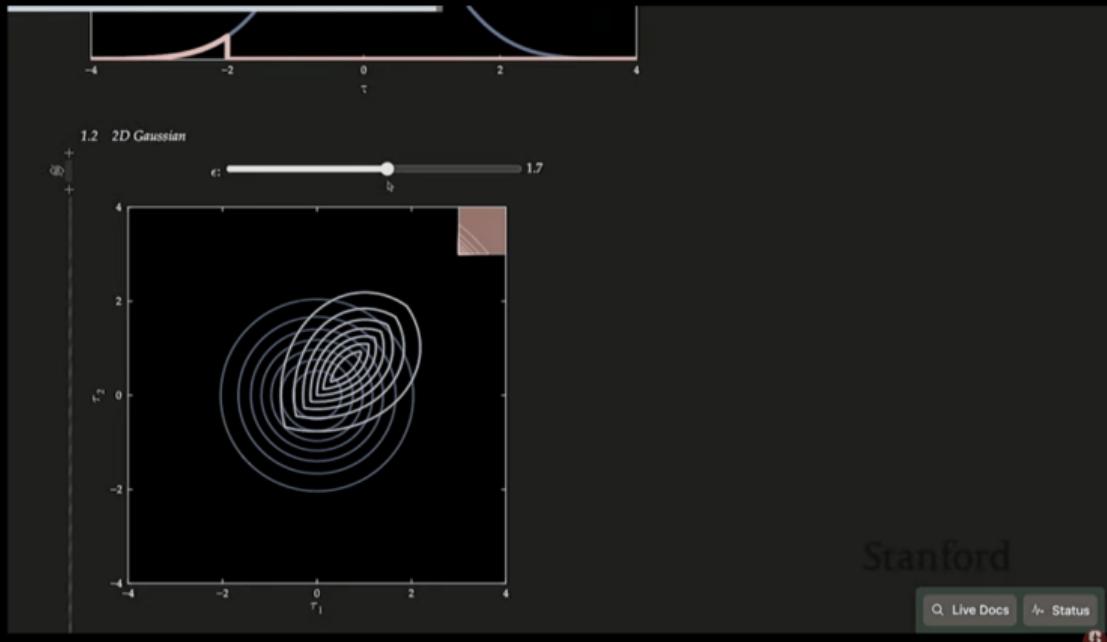
# LECTURES



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

Stanford AA228V/CS238V Programming Projects

Programming projects for Stanford's AA228V/CS238V Validation of Safety-Critical Systems.

CAS: Failure

CAS: Success

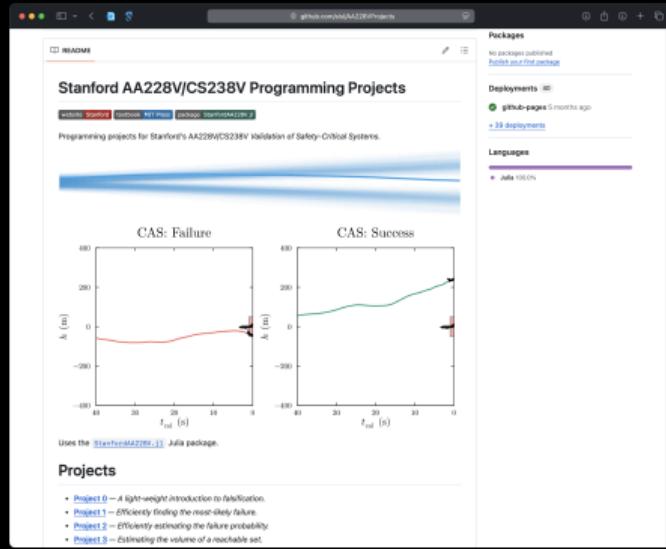
Uses the [StanfordMAster.jl](#) Julia package.

Projects

- Project 0 — A light-weight introduction to telefication.
- Project 1 — Efficiently finding the next-step failure.
- Project 2 — Efficiently estimating the failure probability.
- Project 3 — Estimating the volume of a reachable set.

Assignments repository  
Student-facing code

# ASSIGNMENTS IN JULIA



Stanford AA228V/CS238V Programming Projects

Programming projects for Stanford's AA228V/CS238V Validation of Safety-Critical Systems.

CAS: Failure

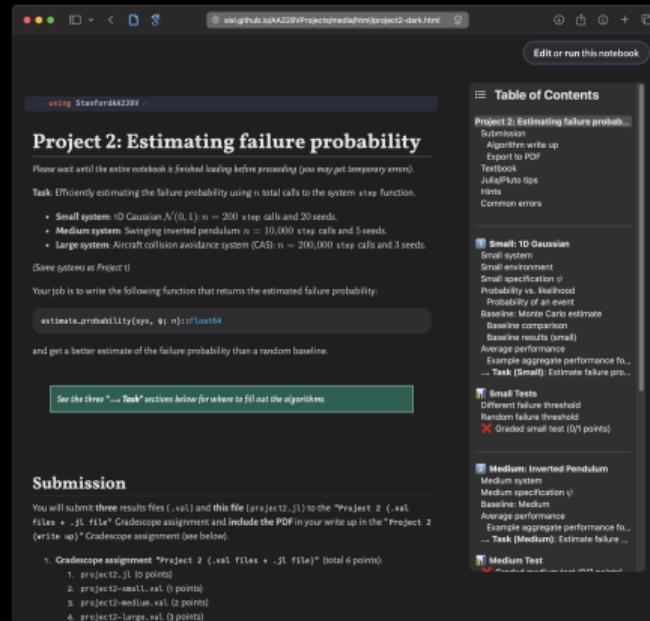
CAS: Success

Uses the [StanfordMATH.jl](#) Julia package.

Projects

- Project 0 – A light-weight introduction to Gradscale.
- Project 1 – Efficiently finding the near-shley failure.
- Project 2 – Efficiently estimating the failure probability.
- Project 3 – Estimating the volume of a macheable set.

Assignments repository  
Student-facing code



## Project 2: Estimating failure probability

Please wait until the entire notebook is finished loading before proceeding (you may get temporary errors).

Task: Efficiently estimating the failure probability using  $n$  total calls to the system step function.

- Small system: 1D Gaussian  $\mathcal{N}(0, 1)$ ;  $n = 200$  step calls and 20 seeds.
- Medium system: Swinging inverted pendulum;  $n = 10,000$  step calls and 5 seeds.
- Large system: Aircraft collision avoidance system (CAS);  $n = 200,000$  step calls and 3 seeds.

(Same system as Project 1)

Your job is to write the following function that returns the estimated failure probability:

```
estimate_probability(ys, ys; n)::Float64
```

and get a better estimate of the failure probability than a random baseline.

See the three “...Task” sections below for where to fill out the algorithms.

### Submission

You will submit three results files (.val) and this file (project2.jl) to the “Project 2 (.val files + .jl file” Gradscale assignment and include the PDF in your write up in the “Project 2 (write up)” Gradscale assignment (see below).

Gradscale assignment “Project 2 (.val files + .jl file)” (total 6 points)

- project2.jl (0 points)
- project2-small.val (0 points)
- project2-medium.val (2 points)
- project2-large.val (3 points)

### Table of Contents

- Project 2: Estimating failure probability ..
- Submission ..
- Algorithm write up ..
- Export to PDF ..
- Testfile ..
- JuliaPkgInfo.ipynb ..
- Hints ..
- Common errors ..

- Small: 1D Gaussian ..
- Small system ..
- Small environment ..
- Small specification  $\psi$  ..
- Probability vs. likelihood ..
- Probability of an event ..
- Baseline: Monte Carlo estimate ..
- Baseline comparison ..
- Baseline results (small) ..
- Average performance ..
- Example aggregate performance fo..
- ... Task (small): Estimate failure pro..

- Small Tests ..
- Different failure threshold ..
- Random failure threshold ..
- Created small test (0? points)

- Medium: Inverted Pendulum ..
- Medium system ..
- Medium specification  $\psi$  ..
- Baseline: Medium ..
- Average performance ..
- Example aggregate performance fo..
- ... Task (medium): Estimate failure ..

- Medium Test ..

Pluto assignments  
Includes local tests

# ASSIGNMENTS IN JULIA

The screenshot shows a Pluto notebook interface with the title "Large Test". The notebook content includes:

- A green checkmark indicating a "Graded large test (3/3 points)" has passed.
- A note: "Click to re-run the LargeSystem evaluation." followed by a link to "most\_likely\_failure(::LargeSystem, s)".
- A link to "LargeSystem" with a note: "This will re-run most\_likely\_failure(::LargeSystem, s) and re-save project3-Large.vat. Uncheck this to load results from the file."
- A section titled "CollisionAvoidance tests passed!" containing:
  - Text: "You found a passing trajectory!"
  - Equation:  $\ell = -97.37071722814194$  (failure log-likelihood)
  - Text:  $n_{step} = 9.963$  (step calls ≤ 10,000)
  - Text: "Most-likely failure found"
  - A plot showing position  $A$  (m) versus time  $t_{rel}$  (s). The plot shows a red curve starting at approximately (-100, -100), rising to a peak around (30, 100), and then falling back towards the end point. A vertical red line marks the "Most-likely failure".
- A "Results saved for CollisionAvoidance" message with the path: "/Users/moser/Code/sis1/AA228/Student/AA228/Projects/project3/project3-Large.vat".
- A note: "Please submit the file listed above to Gradescope."

The sidebar on the right contains a "Table of Contents" with sections like "Project 1: Finding the most-likely fail...", "Small: 1D Gaussian", and "Small Tests".

Interactive Pluto tests  
Get feedback instantly

# ASSIGNMENTS IN JULIA

The screenshot shows a Pluto notebook interface with the following details:

- Title:** Large Test
- Description:** We'll automatically test your most\_likely\_failure(::largeSystem, s) function below.
- Status:** Graded large test (3/3 points)
- Feedback:** Click to re-run the LargeSystem evaluation.
- Text:** This will re-run most\_likely\_failure(::LargeSystem, s) and re-save project3-Large.vat. Uncheck this to load results from the file.
- Plot:** CollisionAvoidance tests passed! A plot titled "Most-likely failure found" showing position A (m) vs time t\_end (s). The plot shows a red curve starting at approximately (-100, -100), rising to a peak around (30, 100), and then falling back towards the end point.
- Text:** You found a passing trajectory!  
 $\delta = -97.37071722814194$  (failure log-likelihood)  
 $n_{step} = 9.963$  (step calls ≤ 10,000)
- Table of Contents:**
  - Project 1: Finding the most-likely fail...
  - Submission
  - Algorithm write up
  - Export to PDF
  - Textbook
  - JuliaPluto tips
  - Hints
  - Common errors
- Small Tests:**
  - Small: 1D Gaussian
    - Small system
    - Small environment
    - Small specification φ
    - Random baseline φ
    - Baselines Small
    - Baseline results (small)
    - Task (Small): Most-likely failure
  - Small Answers
  - Fuzzing
    - Fuzzing Implementation
    - Optimization
    - Optim.jl
    - Optim.jl, Nelder-Mead Implement...
    - (Custom) Gradient Descent
    - (Custom) Gradient Descent Implement...
  - Small Tests
    - Default tolerance threshold
    - Random failure threshold
    - Slider to control threshold
    - Graded small test (1/1 points)
    - Algorithm for SmallSystem
- Buttons:** Live Docs, Status

- Separated assignment code from core library ([StanfordAA228V.jl](#))

Interactive Pluto tests  
Get feedback instantly

# ASSIGNMENTS IN JULIA

The screenshot shows a Pluto notebook interface with the following details:

- Title:** Large Test
- Description:** Will automatically test your most\_likely\_failure(::largeSystem, s) function below.
- Status:** Graded large test (3/3 points)
- Feedback:** Click to re-run the LargeSystem evaluation.
- Plot:** CollisionAvoidance tests passed! A plot titled "Most-likely failure found" shows position A (m) versus time t\_end (s). The plot shows a red curve starting at approximately (-100, -100), rising to a peak around (30, 100), and then settling near (50, -50).
- Results:** Results saved for CollisionAvoidance.
- File Path:** /Users/moser/Code/sis1/AA228V/Student/AA228VProjects/projects/project3/project3-Large.val
- Instructions:** Please submit the file listed above to Gradescope.

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

Interactive Pluto tests  
Get feedback instantly

# ASSIGNMENTS IN JULIA

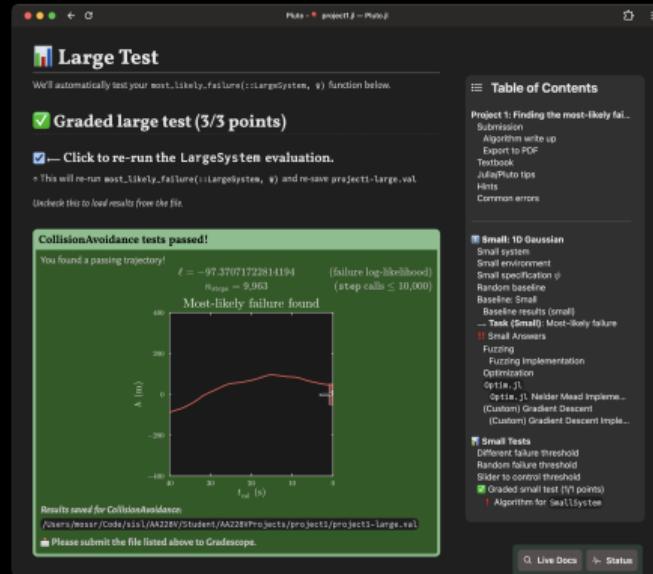
The screenshot shows a Pluto notebook interface with the following details:

- Title:** Large Test
- Description:** Will automatically test your most\_likely\_failure(::largeSystem, s) function below.
- Status:** Graded large test (3/3 points)
- Feedback:** Click to re-run the LargeSystem evaluation.
- Output:** CollisionAvoidance tests passed! (A plot showing position vs time with a red curve and a green shaded region indicating uncertainty.)
- Log:** You found a passing trajectory!  $\delta = -97.37071722814194$  (failure log-likelihood)  $n_{step} = 9.963$  (step calls ≤ 10,000)
- Table of Contents:** A sidebar listing various project tasks and their status (e.g., Small: 1D Gaussian, Large: Most likely failure, etc.).
- Bottom:** Status bar with 'Live Docs' and 'Status' buttons.

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

Interactive Pluto tests  
Get feedback instantly

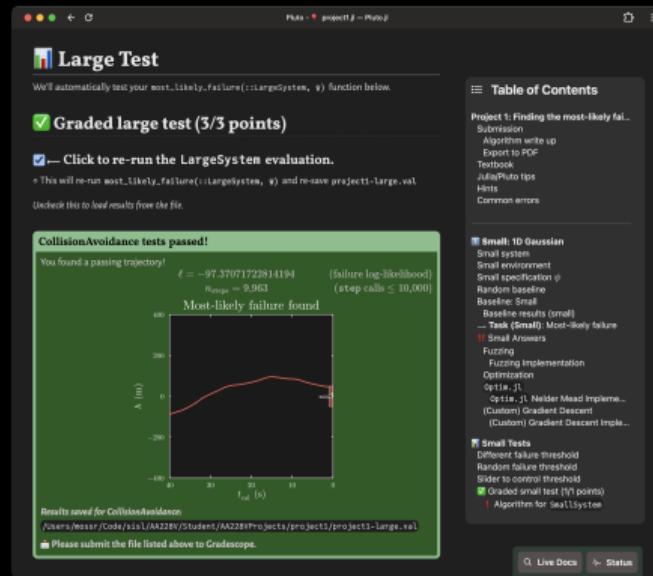
# ASSIGNMENTS IN JULIA



- 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

Interactive Pluto tests  
Get feedback instantly

# 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

The screenshot shows the 'Configure Autograder' page on Gradescope. At the top, there's a navigation bar with three dots, a back arrow, and a forward arrow. The title 'Configure Autograder | Gradescope' is displayed, along with the Gradescope logo.

The main section is titled 'Configure Autograder'. It includes a note: 'Upload your autograder code and change settings here. You can also come back to this step later, but submissions will not be automatically graded until then. Please follow our [guidelines](#) for structuring your autograder.' Below this, there's a note: 'Note: Uploading an autograder zip file will automatically update your Dockerhub image name once it is built successfully.'

There are two radio button options for 'Autograder Configuration': 'Zip file upload' (selected) and 'Manual Docker configuration'.

The 'Autograder' section contains a file input field with the value 'autograder\_project1.zip', a 'Replace Autograder (.zip)' button, and a 'Download Autograder' button.

Configuration dropdowns include 'Base Image OS' set to 'Ubuntu', 'Base Image Version' set to '22.04', and 'Base Image Variant' set to 'Base'. A note below these says: 'Choose the [base image](#) that will be used to build your autograder. This determines the operating system version and packages available in your autograder.'

Buttons at the bottom of this section are 'Update Autograder' and 'Test Autograder'.

The 'Docker Image Status' section shows: 'built as of Jan 31, 2025 at 8:58:58 PM PST'. It has links for 'Build Output' and 'Build Errors'.

At the bottom right of the main content area is a green button labeled 'Manage Submissions'.

Autograde via Gradescope  
Students upload Pluto notebook

# GRADING ASSIGNMENTS

The screenshot shows the 'Configure Autograder | Gradescope' page. It includes fields for 'Autograder Configuration' (with 'Zip file upload' checked), 'Autograder (.zip)' (containing 'autograder\_project1.zip'), and 'Docker Image Status' (built on Jan 31, 2025 at 8:58:58 PM PST). A 'Manage Submissions' button is at the bottom.

Autograde via Gradescope  
Students upload Pluto notebook

The screenshot shows the GitHub repository 'Gradescope.jl'. It displays the repository's README, which includes the title 'Gradescope.jl', a description of the interface for grading Julia code through Gradescope autograders, and links to the specification and contact information. The repository has 7 stars and 2 forks.

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



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

The screenshot shows a web-based interface for an interactive research paper. At the top, there's a navigation bar with the title 'PlutoPapers.jl' and a file path 'PlutoPapers\_11/contents/lectures/1'. Below the title, the page title is 'Algorithms for Validation'.

The main content area has a header 'Lecture Introduction' and a section titled '1 Introduction'. The text in this section discusses the importance of validation in high-stakes settings like autonomous vehicles, robotics, and healthcare, noting that validation is a critical component of the development process. It highlights that as systems become more complex, the spectrum of possible behaviors becomes more challenging and requires a rigorous validation process. The text also mentions the need for validation from a historical perspective and the societal consequences of validation failures.

Below the introduction, there's a section titled '2 Probability Distributions' with a sub-section on the 'univariate normal distribution'. A mathematical formula for the normal distribution is shown:

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

A figure titled 'Figure 1: The normal (Gaussian) distribution' shows a bell-shaped curve on a coordinate system with the x-axis labeled 'x' and numerical ticks at -4, -2, 0, 2, 4. The curve is centered at 0.

At the bottom of the page, there are two sliders for parameters  $\mu$  and  $\sigma^2$ . The  $\mu$  slider is set to 0.1, and the  $\sigma^2$  slider is set to 1.00.

A small note on the right side of the page says: 'Here is a longer sidebar explaining more things in detail. This could influence links to other material.'

Interactive research papers  
Run code directly in the paper

# PlutoPapers.jl

Pluto.jl

Algorithms for Validation

Lecture Introduction

1 Introduction

Before designing decision-making processes 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 systems become more complex, the range of behaviors they can exhibit increases, and the 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 validation that can be used to help validate a broad range of systems.

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:

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

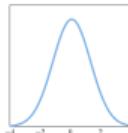
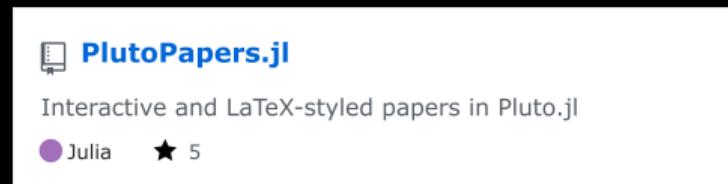


Figure 1: The normal (Gaussian) distribution

$\mu =$   0.1  
 $\sigma^2 =$   1.0



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

Interactive research papers  
Run code directly in the paper

THANK YOU!



QUESTIONS?

[mossr@cs.stanford.edu](mailto:mossr@cs.stanford.edu)