

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 bottom row shows the same after the most likely path on the plate has been found. The bottom row also shows that our likely moves towards the objective, and a failure that stays close to the nominal path, only moving toward the obstacle 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

$$f(\mathbf{v}) = \rho(\mathbf{v}, \mathbf{g}) - \lambda \log(\rho(\mathbf{v})) \quad (4.8)$$

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

```
function weighted_likelihood_objective(v, rho, w, w_smooth=0.0, l=1.0)
    g = v - v_nominal(v, w)
    l = l * max(1.0, 1.0 - g[1])
    g = nominal_trajectory(w, logit(l))
    return rho(v, g) + w_smooth * logit(l) * (l - logit(l)) * l
end
```

Algorithm 4.4c. Objective function that weights the tradeoff between robustness and likelihood. The function takes a trajectory v , a nominal trajectory w , a weighting parameter w , a smoothness parameter w_{smooth} , and a weighting factor l . The smoothness parameter is set to 0.0 and the negative likelihood is set to 1.0. The negative likelihood is 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).¹⁴ 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.stanford.edu

ROBERT MOSS, PHD
STANFORD UNIVERSITY | JULIACON 2025
mossr@cs.stanford.edu

WHOAMI

WHOAMI



Recent Stanford PhD Graduate
Used Julia throughout my research

WHOAMI



Robert Moss
Using Julia as a Specification Language for the
Next-Generation Airborne Collision Avoidance System

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 dark trousers standing at a podium, gesturing with his hands as he speaks. 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

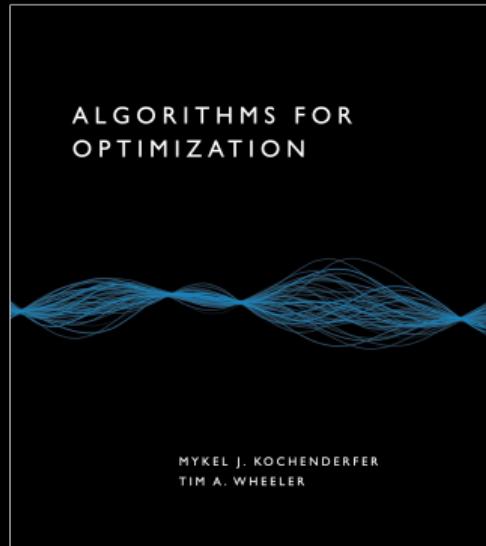
ALGORITHMS FOR VALIDATION

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



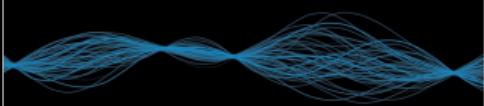
MYKEL J. KOCHENDERFER
TIM A. WHEELER
KYLE H. WRAY

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. To the right of the video frame, there is a caption with the speaker's name and affiliation.

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

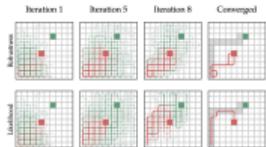


Figure 4.3. A comparison of optimization-based falsification on the grid world using the robustness objective and the weighted objective. The plots show the progression of the weighted optimization algorithm, which will be discussed in the next section. The first column of plots in the final column represents the robustness objective, while the second column represents the weighted objective (this follows that quickly moves towards the obstacles). The weighted objective finds a failure that stays close to the target, while the robustness objective stays far away from 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 λ is a weighting parameter selected by the user (algorithm 4.10). This objective is smooth and converges the optimization algorithm to search stimulus ready for trajectories that are both likely and close to failure.

```
function weighted_likelihood_objective(x, y, z, w, b, n, l=1.0)
    y = rollout(y, z, x)
    c = 1.0/n * sum(rollout(y, z, x))
    k = l * (n-1) / (l * (n-1) + 1)
    p = log(k) + log(1 - k) + log(rollout(y, z, length(z)))
    return softmax(x, w, formula, n) * p - l * log(pdf(x, c))
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).⁴ 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 target set y , a specification z , a system w , a formula n , a weighting parameter l , a step size c , and a specification b . It returns a weighted combination of the robustness and likelihood for the solutions if $n > 1$ (otherwise it returns the robustness if $n = 1$). It also handles vectorized solutions (i.e., multiple parallel solutions) by calculating the weighted robustness for each solution and then summing the results.

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

Algorithms for Validation Figures and Julia code

TEXTBOOKS WITH INTEGRATED JULIA

58 CHAPTER 4. FALSIFICATION THROUGH OPTIMIZATION

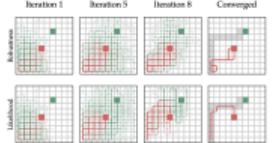


Figure 4.3 shows four 4x4 grid plots labeled 'Iteration 1', 'Iteration 5', 'Iteration 8', and 'Converged'. The first three iterations show a red dot moving through the grid, with green dots representing obstacles. The final 'Converged' plot shows the red dot at a specific location with a green dot nearby, indicating convergence.

Figure 4.3. A comparison of optimization-based falsification on the grid world using the robustness objective and the weighted objective. The plots above show the progress of the weighted objective algorithm, which will be discussed in the next section. The last plot shows the grid in the final column represents the result of the robustness objective. The robustness objective finds failures that quickly move towards the obstacles. The weighted objective finds a failure that stays close to the nominal trajectory while moving 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

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

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

```
function weighted_likelihood_objective(x, y, w, b, smoothness=0.0, N=2-1.0)
    y = rollout(sys, y, x)
    c = rollout(sys, y, x)
    x = [x[i:i+smoothness] for i in 1:N]
    p = [pdf(w, x[i]) for i in 1:N]
    q = [pdf(b, x[i]) for i in 1:N]
    return softmax(x, w, formula, smoothness) - b + log(p/pdf(x, c))
end
```

Algorithm 4.8. Objective function that weights the violation between robustness and likelihood. The function takes a system, a nominal trajectory y , and a specification b . It returns a weighted combination of the weighted constraint for the robustness if w is not zero, or the weighted likelihood for the nominal trajectory distribution.

© 2013 Kochenderfer, Kate, Corra, and Mooney under a Creative Commons CC-BY-NC-ND license.
algorithm-4.8.jl (julia), comments to kochenderfer@mit.edu

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^AT_EX 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^AT_EX code example

Integrated Julia code using [pythontex](#)

JULIA INTEGRATED INTO LATEX CODE

```

361 where the weights are  $ws_i = \langle \mathbf{v}(\text{fail}_i) \rangle / \langle \mathbf{v}(\text{true}_i) \rangle$ . These weights are sometimes referred to as Vortex
362 (importance) weights. Trajectories that are more likely under the nominal trajectory distribution have
363 higher importance weights.
364
365 %begin(julialinear)
366 begin(julialinear)
367 struct ImportanceSamplingEstimation
368     p::nominal_distribution
369     q::proposal_distribution
370     n::Int # number of samples
371 end
372
373 function estimate!(alg::ImportanceSamplingEstimation, sys, v)
374     p, q, n = alg.p, alg.q, alg.n
375     ts = Vector{TimeSeries}(undef, n)
376     for t in ts
377         ps = [pdf(p, v) for t in ts]
378         qs = [pdf(q, v) for t in ts]
379         ws = ps ./ qs
380         return mean(ws * isfailure(v, t) for (w, t) in zip(ws, ts))
381     end
382 end
383 send(julialinear)
384 #caption(julialgic,estimation) The importance sampling estimation algorithm for estimating the
385 probability of failure. The algorithm generates  $\langle \mathbf{v}(\text{v}_n) \rangle$  samples from the proposal distribution  $\langle \mathbf{v}(\text{q}) \rangle$ .
386 It then computes the importance weights for the samples and applies  $\langle \mathbf{v}(\text{eq}(\text{q})) \rangle$  to compute
387  $\langle \mathbf{v}(\text{f}, \text{stoch}) \rangle$ .
388 #end(julialgic)
389
390 #subsection(Optional_Proposal_Distribution)
391 #label(Optional_Proposal_Distribution)
392 #caption(Optional_Proposal_Distribution)
393 #note The accuracy and efficiency of importance sampling approaches is highly dependent on the proposal
394 distribution, and variance of the estimator in vref(eqis_estimator) is
395 #begin(eqis_variance)
396 #label(eqis_variance)
397 Vortex(E, vref(eqis_var)) = Vortex(1/(E)) * Vortex(E * (vref(eqis_mean) - left) * (frac(p, tau) * Vortex(E) - Vortex(p * tau))) + Vortex((p * tau) * Vortex(E) - Vortex(p * tau)) * right)
398 #end(eqis_variance)
399 #caption(eqis_variance)
400 #note In general, we want to select a proposal distribution that makes this variance low, and the optimal
401 proposal distribution is the one that minimizes this variance.
402
403 It is evident from vref(eqis_var) that we can achieve a variance of zero when
404 #begin(eqis_mean)
405 #label(eqis_mean)
406 #caption(eqis_mean)
407 #note  $\langle \mathbf{v}(\text{eq}(\text{q})) \rangle = \langle \mathbf{v}(\text{frac}(\text{p}, \text{tau})) \rangle \cdot \mathbf{v}(\text{mean}(\text{p} * \text{tau}))$ 
408 #end(eqis_mean)
409 #caption(eqis_mean)
410 #note This distribution corresponds to the failure distribution  $\langle \mathbf{v}(\text{v}_n) \rangle$  and  $\langle \mathbf{v}(\text{v}_n) \rangle$ . As noted in
411 vref(eqis_failure_distribution), computing this distribution is not possible in practice since we often
412 do not know the failure distribution. Our goal is therefore to select a proposal constant  $\langle \mathbf{v}(\text{p} * \text{tau}) \rangle$  that is
413 the quantity we are trying to estimate. Our goal is therefore to select a proposal constant that is as
414 close as possible to the failure distribution.
```

L^AT_EX code example

Integrated Julia code using `pythontex`

7.2. IMPORTANCE SAMPLING 14

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

```

structure ImportanceSamplingEstimation {
    p # nominal distribution
    q # proposed distribution
    n # number of samples
}

function estimate(aq::Alg{ImportanceSamplingEstimation}, sys, v)
    g, g_0 = aq.g, aq.g_0
    ws = [rollout(sys, q) for i in 1:n]
    gs = [g(ws[i]) for i in 1:n]
    g0s = [g0(ws[i]) for i in 1:n]
    ws = ws ./ ws
    return mean(g == falsevals(g, 1) & (ws .>= v))
}

```

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)

$$\text{Var}[\beta_{\text{bd}}] = \frac{1}{n} \mathbb{E}_{T \sim g(\cdot)} \left[\frac{(p(\tau) \mathbb{1}\{\tau \notin \psi\} - q(\tau)p_{\text{bd}})^2}{s(\tau)} \right] \quad (7.10)$$

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

$$q^*(\tau) = \frac{p(\tau) \mathbb{1}\{\tau \notin \phi\}}{p_{\text{out}}} \quad (7.15)$$

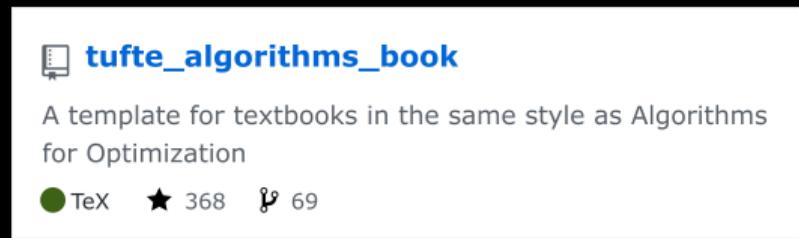
This distribution corresponds to the failure distribution $p(\tau \mid t \notin \mathcal{S})$. As noted in chapter 6, computing this distribution is not possible in practice since we often do not know the full set of failure trajectories and the normalizing constant p_{fail} is the quantity we are trying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

© 2024 Kochenderfer, Katz, Cono, and Moss shared under a Creative Commons CC-BY-NC-ND license
2023-91-05 13:27:33-08:00; comments to bear@californiathesbook.com

Compiled L^AT_EX PDF

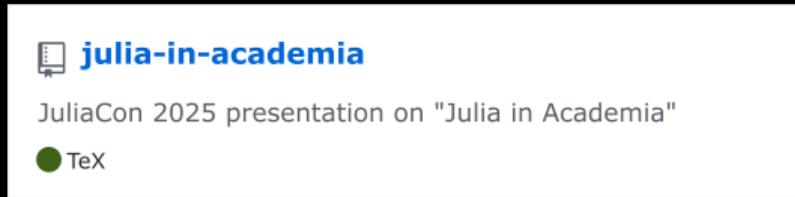
OPEN-SOURCE TEXTBOOK TEMPLATE REPOSITORY

OPEN-SOURCE TEXTBOOK TEMPLATE REPOSITORY



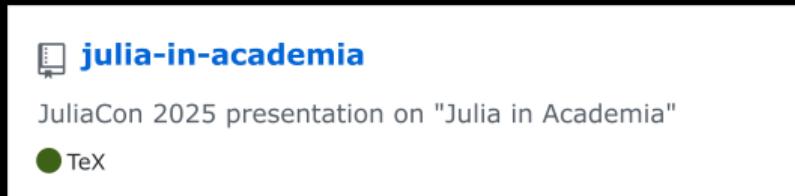
github.com/sisl/tufte_algorithms_book

OPEN-SOURCE SLIDES REPO (BEAMER + PYTHONTEX)

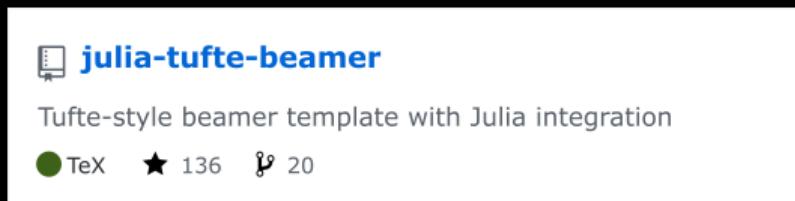


github.com/mossr/julia-in-academia

OPEN-SOURCE SLIDES REPO (BEAMER + PYTHONTEX)



github.com/mossr/julia-in-academia



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 `\lambda` 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 `\lambda` 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

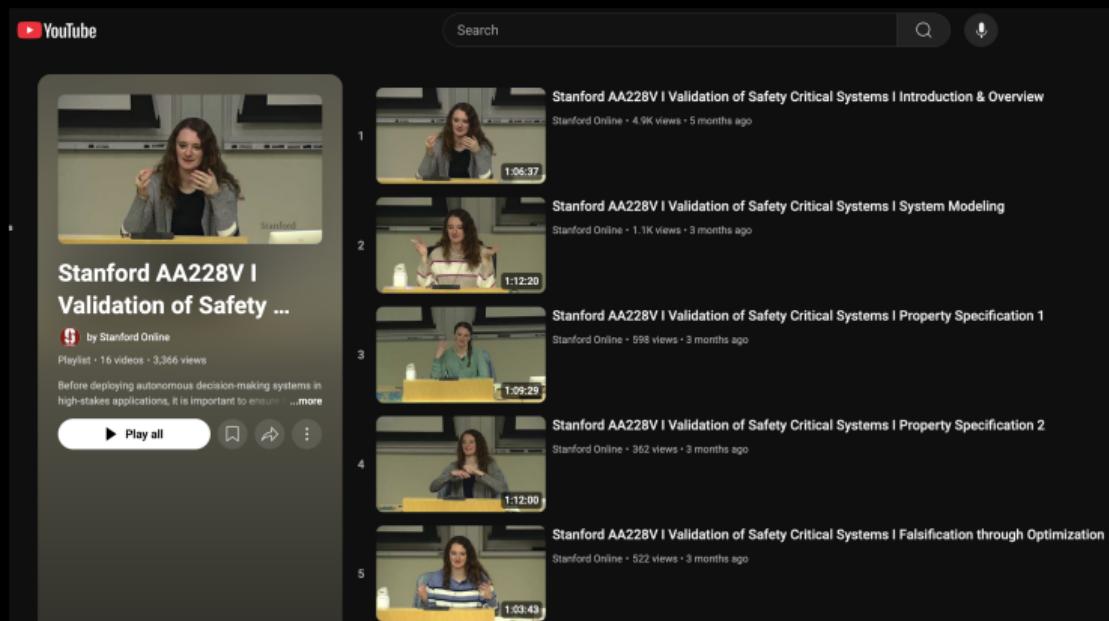
Grad-level course at Stanford

Follows *Algorithms for Validation* textbook

First offered in Winter 2025

LECTURES

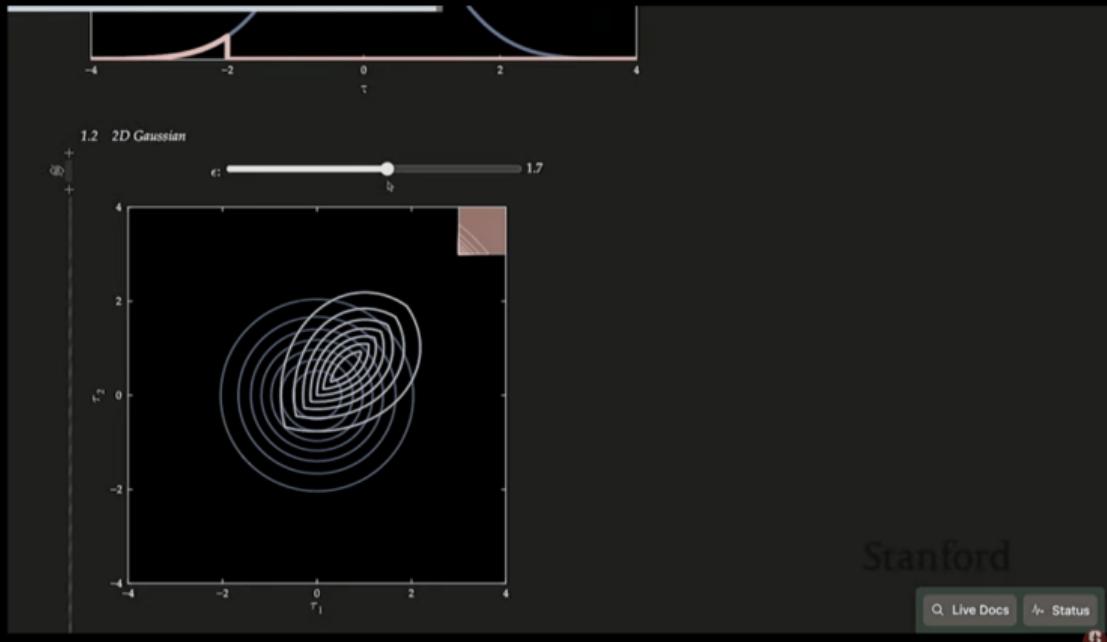
LECTURES



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

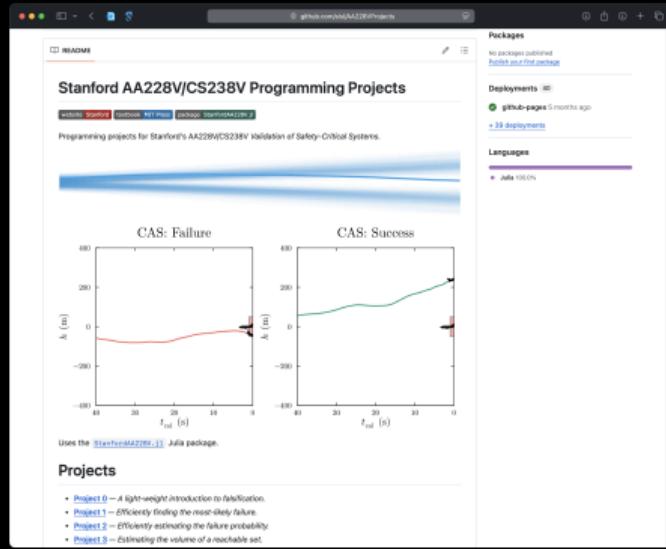
Uses the [StanfordMAster.jl](#) Julia package.

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

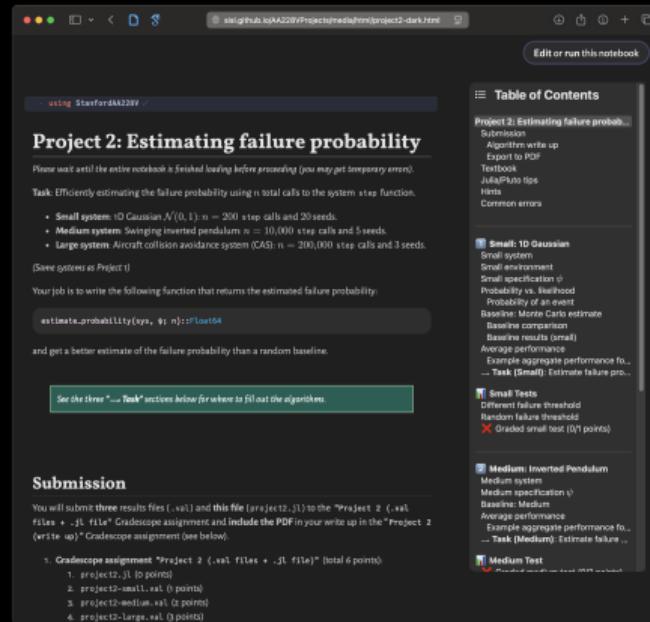
CAS: Success

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 ψ ..
- 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 ψ ..
- 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 instructions to re-run the function and save the project.
- A plot titled "Most-likely failure found" showing a red curve over time t_{rel} (s) from -100 to 100, and altitude A (m) from -300 to 300. The plot shows a trajectory starting at approximately (-100, -250), rising to a peak around (0, 100), and then descending towards the end.
- Text output: "CollisionAvoidance tests passed!" followed by "You found a passing trajectory!" and "Most-likely failure found".
- Output values: $\ell = -97.37071722814194$, $n_{step} = 9.963$, and "(step calls ≤ 10,000)".
- A sidebar "Table of Contents" listing various project sections and a "Small: 1D Gaussian" section.
- A footer message: "Results saved for CollisionAvoidance" and "Please submit the file listed above to Gradescope."

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.
- Results:** Results saved for CollisionAvoidance
- File Path:** /Users/moser/Code/sis1/AA228V/Student/AA228VProjects/projects/project3/project3-Large.vat
- Instructions:** Please submit the file listed above to Gradescope.
- Table of Contents:** A sidebar listing various JuliaPluto tips and common errors, including sections on Small: 1D Gaussian, Small Tests, and Small Answers.
- Buttons:** Live Docs and 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

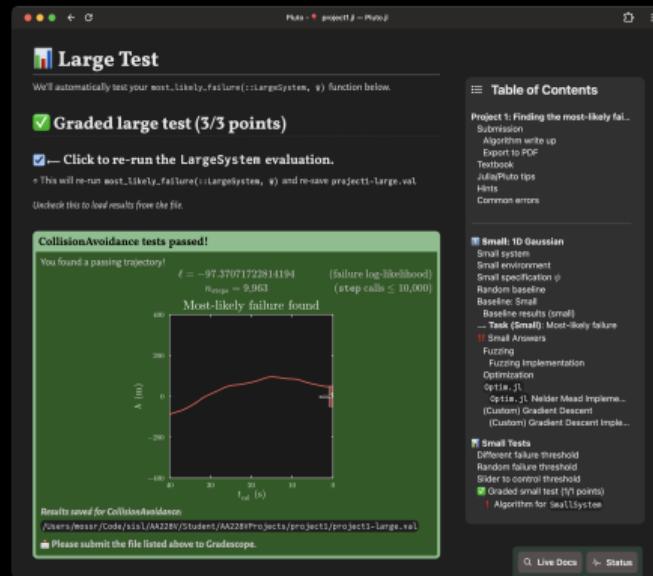
The screenshot shows a Pluto notebook interface with the following details:

- 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

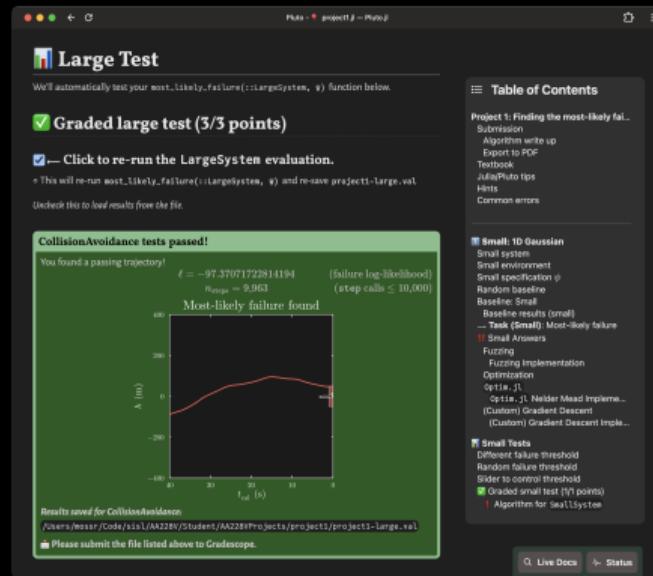
ASSIGNMENTS IN JULIA



- 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 content area has a heading 'Configure Autograder'. Below it, a note says: '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.' A note below that says: 'Note: Uploading an autograder zip file will automatically update your Dockerhub image name once it is built successfully.'

A section titled 'Autograder Configuration' contains a note: 'Required field'. It includes a radio button for 'Zip file upload' (which is selected) and another for 'Manual Docker configuration'. Below this is a file input field containing 'autograder_project1.zip', with buttons to 'Replace Autograder (.zip)' and 'Download Autograder'.

Below the file input are dropdown menus for 'Base Image OS' (Ubuntu), 'Base Image Version' (22.04), and 'Base Image Variant' (Base). A note next to these dropdowns 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.'

At the bottom of this section are two buttons: 'Update Autograder' and 'Test Autograder' (with a play icon).

The 'Docker Image Status' section shows: 'built as of Jan 31, 2025 at 8:58:58 PM PST'. It has sections 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 'Replace Autograder (.zip)'. It also has dropdowns for 'Base Image OS' (Ubuntu), 'Base Image Version' (22.04), and 'Base Image Variant' (Base). A note 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.' Below this are 'Docker Image Status' (built at Jan 31, 2025 at 8:58:58 PM PST) and 'Build Output/Errors' sections. At the bottom is a 'Manage Submissions' button.

Autograde via Gradescope
Students upload Pluto notebook

The screenshot shows the GitHub repository 'Gradescope.jl' by 'sisl'. The repository page includes a code editor for 'Gradescope.jl' (Public), commit history (e.g., 'mossr Added custom registry url feature'), and repository details like 'About' (Julia interface for Gradescope autograding), 'Releases' (No releases published), 'Packages' (No packages published), and 'Languages' (Julia 100.0%).

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?

Pluto.jl   

[Algorithms for Validation](#)

Lecture Introduction

1 Introduction

Before deploying decision-making options 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. As the complexity of the systems increases, so does the complexity of validating them. 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 chapter provides an overview of validation, its importance, and the challenges involved 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.³

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

³ Here is a longer sidebar explaining more things in detail. This could reference links or other material.

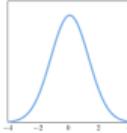


Figure 1. The normal (Gaussian) distribution

$\mu =$ 0.1

$\sigma^2 =$ 1.0

Interactive research papers
Run code directly in the paper

PLUTO PAPERS.JL

Pluto.jl    

Algorithms for Validation

Lecture Introduction

1 Introduction

Before deploying decision-making options 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. As the complexity of the systems increases, so does the complexity of validating them. 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 chapter provides an overview of validation methods and their application 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

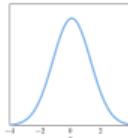
$$N(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$


Figure 1. The normal (Gaussian) distribution

$\mu =$ 0.1

$\sigma^2 =$ 1.0

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

PlutoPapers.jl

Interactive and LaTeX-styled papers in Pluto.jl

Julia  5

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

Interactive research papers
Run code directly in the paper

THANK YOU!

QUESTIONS?

mossr@cs.stanford.edu