

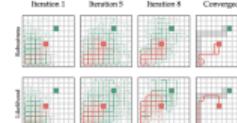
Julia in Academia

TEXTBOOKS, STANFORD COURSES, AND THE FUTURE

The cover of the book 'Algorithms for Validation' features a dark background with a blue horizontal bar at the bottom. The title 'ALGORITHMS FOR VALIDATION' is centered in white capital letters. Below the title, the authors' names are listed: MYKEL J. KOCHENDERFER, SYDNEY M. KATZ, ANTHONY L. CORSO, and 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 true 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 same algorithm that quickly moves towards the objective. A red dot marks the final objective and a blue dot that stays close to the red dot, only moving toward it at the end of the run.



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

Another common objective for event likely failure analysis is

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

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, smoothness=0.0, lambda=0.0)
    v = validate(v, w)
    v = log(v)
    v = -lambda*gradient(v)*log(rho(v))
    return rho(v), w, smoothness
end
```

Algorithm 4.4c. Objective function that weighs the smoothness between robustness and likelihood. The function takes a trajectory v , a system w , and a weighting parameter λ . It also takes a smoothness parameter ϵ that controls the rate of the smoothed robustness function. The smoothness parameter is set to 0.0 and the negative likelihood function is the smooth 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.cmu.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

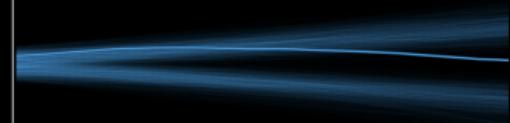


Recent Stanford PhD Graduate
Used Julia throughout my research

A video thumbnail from JuliaCon 2015. It shows a man in a grey shirt and 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

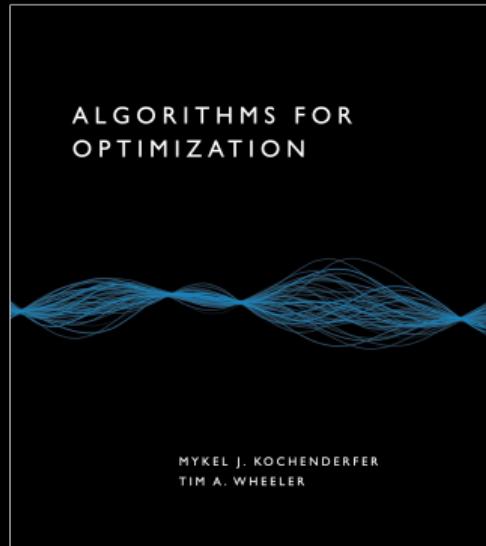


MYKEL J. KOCHENDERFER
SYDNEY M. KATZ
ANTHONY L. CORSO
ROBERT J. MOSS

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

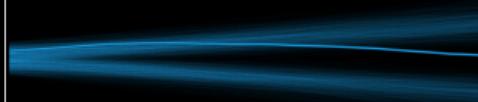
ALGORITHMS FOR
DECISION MAKING



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. A woman is visible in the foreground, looking at a screen. The video frame has a caption below it: "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, 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

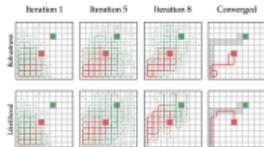


Figure 4.3. A comparison of optimization-based falsification on the grid world using the robustness objective and the weighted objective. The plots show the progression of the weighted optimization algorithm, which will be discussed in the next section. The top row of plots in each column in the final column represents the robustness objective, while the bottom row represents the weighted objective. Both objectives find a failure that stays close to the start, but the weighted objective finds a failure that stays closer to the goal, and both converge very quickly toward the solution 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/(1 + exp(-(y - c))) + log(1/(1 + exp(-(y - c))) + log(pdf(x, z)))
    return softmax(x, w, formula, n, smoothness) - k + log(pdf(x, z))
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 violation between solutions and likelihood. The function takes a solution x , a target solution y , a specification z , a system w , a formula n , a smoothness parameter l , and a weighting parameter k . It returns a weighted combination of the weighted likelihood for the solution if $smoothness < 0$ or the weighted likelihood for the robustness if $smoothness > 0$. The function also handles wider the normal trajectory distribution.

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

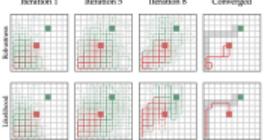


Figure 4.3 shows four 4x4 grid plots illustrating the progress of an optimization-based falsification algorithm over four iterations. The first three plots show the 'Robustness' objective, where the red dot (failure) moves from the top-left towards the bottom-right. The fourth plot, labeled 'Converged', shows the 'Weighted' objective, where the red dot has moved to the bottom-right corner.

Iteration 1 Iteration 5 Iteration 8 Converged

Robustness

Weighted

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 robustness-based falsification algorithm, which will be discussed in the next section. The first three plots in each row in the final column represents the robustness objective, while the last plot represents the weighted objective (this finds much faster). Note that the robustness objective finds failure that quickly moves towards the obstacles, while the weighted objective finds a failure that stays close to the start 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

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

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 4.10. Objective function that weights the robustness between robustness and likelihood. The function takes a system sys , a nominal trajectory y , and a specification b . It returns a weighted combination of the robustness objective for the robustness if w represents the nominal distribution and the likelihood based on the nominal trajectory distribution.

M.J. Kochenderfer and T.A. Whalen, Algorithms for Optimization, MIT Press, 2020.

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

© 2020 Kochenderfer, Kate, Corra, and Mooney under a Creative Commons CC-BY-NC-ND license. <https://creativecommons.org/licenses/by-nd/4.0/>. algbook@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;isImportanceSamplingEstimation,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(u, r) for (w, r) in zip(ws, rs))  
385 end  
386 %end{juliaverbatim}  
387 %caption{\texttt{Isceis\_estimation}}: The importance sampling estimation algorithm for estimating the  
probability of failure. The algorithm generates  $\mathcal{N}(v|m)$  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  
either \texttt{Xref{eq}{fail1}} or \texttt{Xref{eq}{fail2}}.  
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 
$$\text{Variance}_{\text{IS}}(\hat{p}_{\text{fail}}) = \frac{1}{N} \sum_{i=1}^N \left( \hat{p}_{\text{fail}} - \bar{p}_{\text{fail}} \right)^2$$
  
394 where  $\hat{p}_{\text{fail}} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}_{\{x_{t+1} \in \text{fail}\}}$ .  
395 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.  
396 It is evident from \texttt{Xref{eq}{is_var}} that we can achieve a variance of zero when  
397 %begin{equation}  
398 %label{eq:optimal_proposal}  
399 \mathbb{E}_{x \sim q}[\mathbb{P}_{p \sim p}(x_{t+1} \in \text{fail})] = \mathbb{P}_{p \sim p}(x_{t+1} \in \text{fail})  
400 %end{equation}  
401 This distribution corresponds to the failure distribution  $p(x_{t+1} \in \text{fail})$ . As noted in  
402 \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  $\mathbb{P}_{p \sim p}(x_{t+1} \in \text{fail})$  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.
```

L^AT_EX code example

Integrated Julia code using [pythontex](#)

JULIA INTEGRATED INTO LATEX CODE

```

368 where the weights are  $w_i = p(\tau_{\text{tar},i}) / q(\tau_{\text{tar},i})$ . These weights are sometimes referred to as vtext{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, v) for i in 1:m]
381       ps = [pdf(q, v) for v in rs]
382       qs = [pdf(q, v) for v in rs]
383       ws = ps ./ qs
384       return mean(w * isfailure(u, v) for (u, v) in zip(ws, rs))
385     end
386   end(JuliaVerbatim)
387   caption(alg.estimate) The importance sampling estimation algorithm for estimating the probability of failure. The algorithm generates \|v(m) samples from the proposal distribution \{q(v)\}. It then computes the importance weights for the samples and applies vref(eqis.estimate) to compute vbar_u.vtext{fail}(u).
388   end(algorithm)
389
390   subsection(Optimal Proposal Distribution)
391   label(SectionOptimalProposal)
392   The accuracy and efficiency of importance sampling approaches is highly dependent on the proposal distribution and variance of the estimator in vref(eqis.estimate) is
393   begin(Equation)
394     label(eqis_var)
395     vtext{Var}(\hat{p}_f) = \frac{1}{m} \sum_{i=1}^m \left( \frac{p(\tau_{\text{tar},i})}{q(\tau_{\text{tar},i})} - \hat{p}_f \right)^2
396   end(Equation)
397   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.
398
399 It is evident from vref(eqis_var) that we can achieve a variance of zero when
400 begin(Equation)
401   label(eqis_optimal_proposal)
402   
$$q(\tau_{\text{tar}}) = \frac{p(\tau_{\text{tar}})}{\int p(\tau_{\text{tar}}) d\tau_{\text{tar}}} = \frac{p(\tau_{\text{tar}})}{p_{\text{fail}}}$$

403 end(Equation)
404 This distribution corresponds to the failure distribution np_tau_and_tau_norm(psi). As noted in vref(eqis.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_f 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 `vtext{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 `\|v(m)` samples from the proposal distribution `\{q(v)\}`. It then computes the importance weights for the samples and applies equation (7.9) to compute `p_f`.

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

$$\text{Var}[\hat{p}_f] = \frac{1}{m} \sum_{i=1}^m \left[\frac{(p(\tau_{\text{tar},i})/q(\tau_{\text{tar},i}) - \hat{p}_f)^2}{q(\tau_{\text{tar},i})} \right] \quad (7.10)$$

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

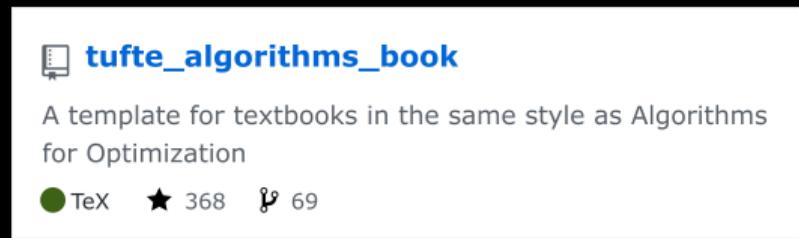
$$q^*(\tau) = \frac{p(\tau)}{\int p(\tau) d\tau} \quad (7.11)$$

This distribution corresponds to the failure distribution $p(\tau) | \tau \in \mathcal{F}$. 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_f 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

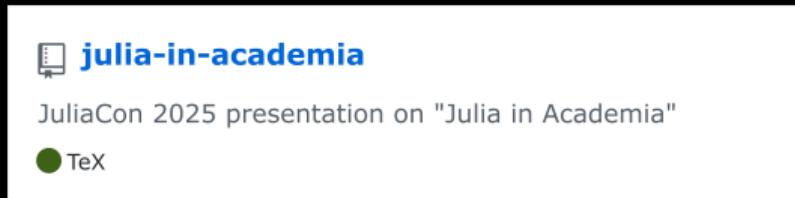
OPEN-SOURCE TEXTBOOK TEMPLATE REPOSITORY



github.com/sisl/tufte_algorithms_book

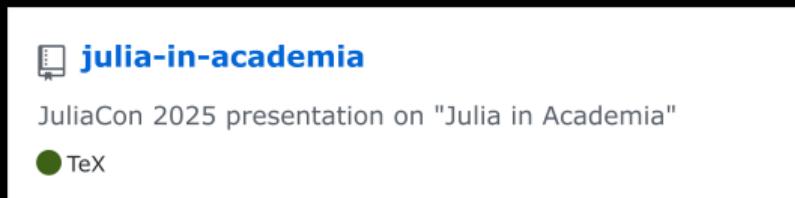
OPEN-SOURCE SLIDES REPO (`BEAMER` + `PYTHONTEX`)

OPEN-SOURCE SLIDES REPO (BEAMER + PYTHONTEX)

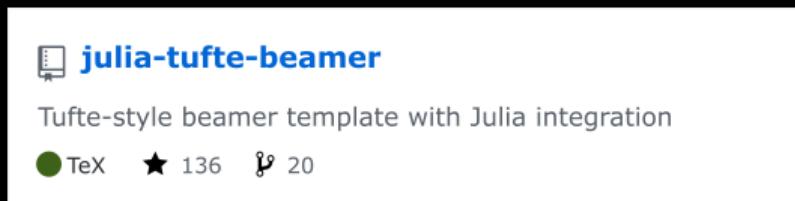


github.com/mossr/julia-in-academia

OPEN-SOURCE SLIDES REPO (BEAMER + PYTHONTEX)



github.com/mossr/julia-in-academia



github.com/mossr/julia-tufte-beamer

INTEGRATED JULIA EXAMPLE

We can compute Julia code *directly in the slides/pages.*

INTEGRATED JULIA EXAMPLE

We can compute Julia code *directly in the slides/pages.*

```
julia> using SignalTemporalLogic
```

INTEGRATED JULIA EXAMPLE

We can compute Julia code *directly in the slides/pages*.

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ₁ = @formula ◊(sₜ → sₜ > 0);

τ # \tau<TAB>
ψ₁ # \psi<TAB>\_1<TAB>
◊ # \lozenge<TAB>
```

INTEGRATED JULIA EXAMPLE

We can compute Julia code *directly in the slides/pages*.

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ₁ = @formula ◊(sₜ → sₜ > 0);
julia> ρ₁ = ρ(τ, ψ₁)
3.0
```

```
τ # \tau<TAB>
ψ₁ # \psi<TAB>\_1<TAB>
◊ # \lozenge<TAB>
ρ # \rho<TAB>
```

INTEGRATED JULIA EXAMPLE

We can compute Julia code *directly in the slides/pages*.

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ₁ = @formula ◊(sₜ → sₜ > 0);
julia> ρ₁ = ρ(τ, ψ₁)
3.0
julia> ψ₂ = @formula □(sₜ → sₜ > 0);
```

```
τ # \tau<TAB>
ψ₁ # \psi<TAB>\_1<TAB>
◊ # \lozenge<TAB>
ρ # \rho<TAB>
□ # \square<TAB>
```

INTEGRATED JULIA EXAMPLE

We can compute Julia code *directly in the slides/pages*.

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ₁ = @formula ◊(sₜ → sₜ > 0);
julia> ρ₁ = ρ(τ, ψ₁)
3.0
julia> ψ₂ = @formula □(sₜ → sₜ > 0);
julia> ρ₂ = ρ(τ, ψ₂)
-4.0

τ # \tau<TAB>
ψ₁ # \psi<TAB>\_1<TAB>
◊ # \lozenge<TAB>
ρ # \rho<TAB>
□ # \square<TAB>
```

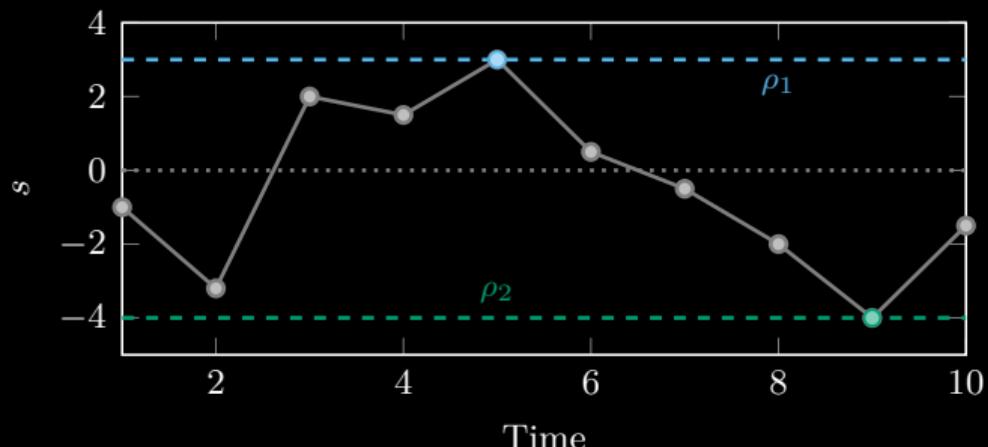
INTEGRATED JULIA EXAMPLE

We can compute Julia code *directly in the slides/pages*.

```
julia> using SignalTemporalLogic
julia> τ = [-1.0, -3.2, 2.0, 1.5, 3.0, 0.5, -0.5, -2.0, -4.0, -1.5];
julia> ψ₁ = @formula ◊(sₜ → sₜ > 0);
julia> ρ₁ = ρ(τ, ψ₁)
3.0
julia> ψ₂ = @formula □(sₜ → sₜ > 0);
julia> ρ₂ = ρ(τ, ψ₂)
-4.0

τ # \tau<TAB>
ψ₁ # \psi<TAB>\_1<TAB>
◊ # \lozenge<TAB>
ρ # \rho<TAB>
□ # \square<TAB>
```

$$\psi_1 = \diamond(s_t > 0) \quad \psi_2 = \square(s_t > 0)$$



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
- Integrated tooling to run and plot Julia code during PDF compilation

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
- Integrated tooling to run and plot Julia code during PDF compilation
- 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
- Integrated tooling to run and plot Julia code during PDF compilation
- 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`

SHOWCASE: `IntervalArithmetic.jl` and `LazySets.jl`

SHOWCASE: `IntervalArithmetic.jl` and `LazySets.jl`

```
struct NaturalInclusion <: ReachabilityAlgorithm
    h # time horizon
end

function r(sys, x)
    s, x = extract(sys.env, x)
    τ = rollout(sys, s, x)
    return τ[end].s
end

to_hyperrectangle(I) = Hyperrectangle(low=[i.lo for i in I],
                                         high=[i.hi for i in I])

function reachable(alg::NaturalInclusion, sys)
    I's = []
    for d in 1:alg.h
        I = intervals(sys, d)
        push!(I's, r(sys, I))
    end
    return UnionSetArray([to_hyperrectangle(I') for I' in I's])
end
```

SHOWCASE: `IntervalArithmetic.jl` and `LazySets.jl`

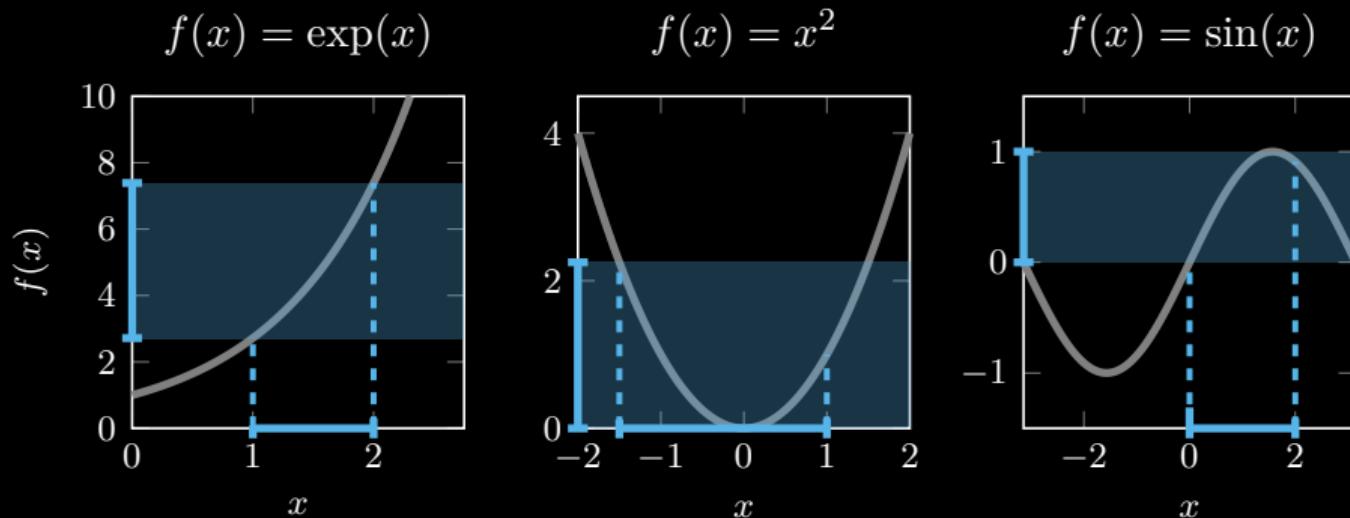


Figure: Example of the interval counterparts for the \exp , square, and \sin functions.

SHOWCASE: `IntervalArithmetic.jl` and `LazySets.jl`

```
function (env::InvertedPendulum)(s, a)
    θ, ω = s[1], s[2]
    dt, g, m, l = env.dt, env.g, env.m, env.l
    ω = ω + (3g / (2 * l) * sin(θ) + 3 * a / (m * l^2)) * dt
    θ = θ + ω * dt
    return [θ, ω]
end
```

SHOWCASE: `IntervalArithmetic.jl` and `LazySets.jl`

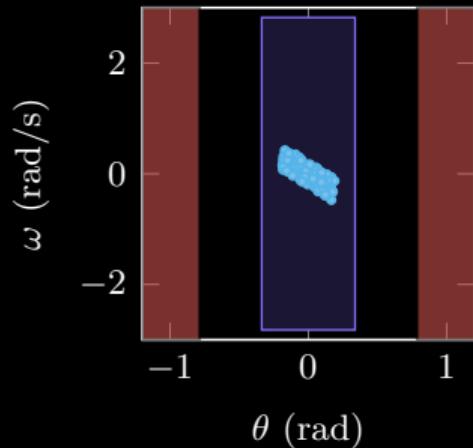


Figure: Inverted pendulum reachable set using `IntervalArithmetic.jl` and `LazySets.jl`.

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

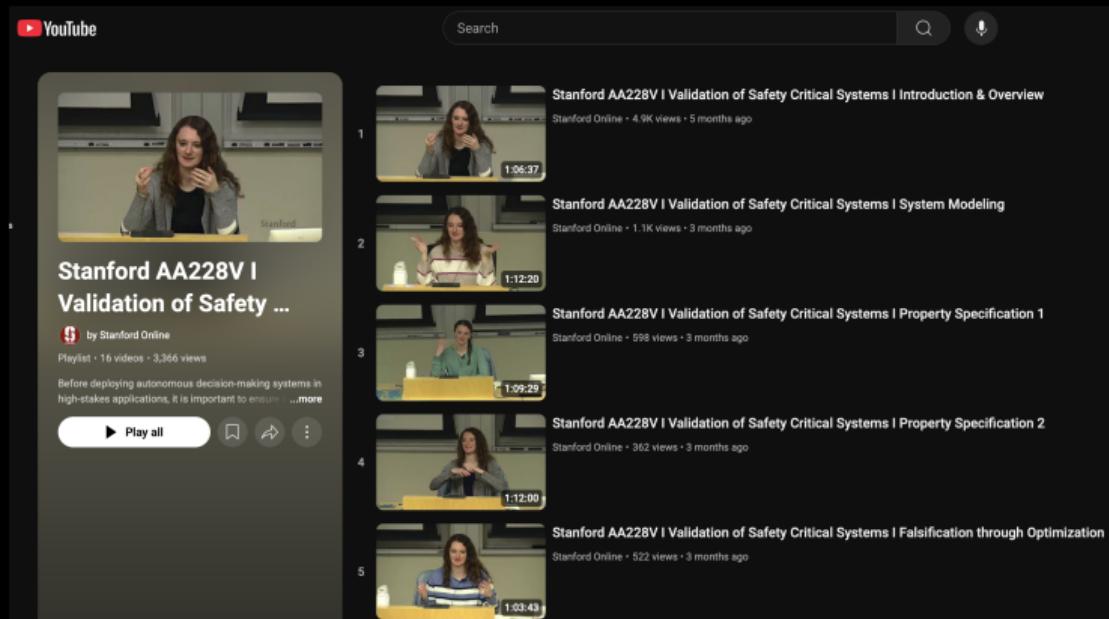
Grad-level course at Stanford

Follows *Algorithms for Validation* textbook

First offered in Winter 2025

LECTURES

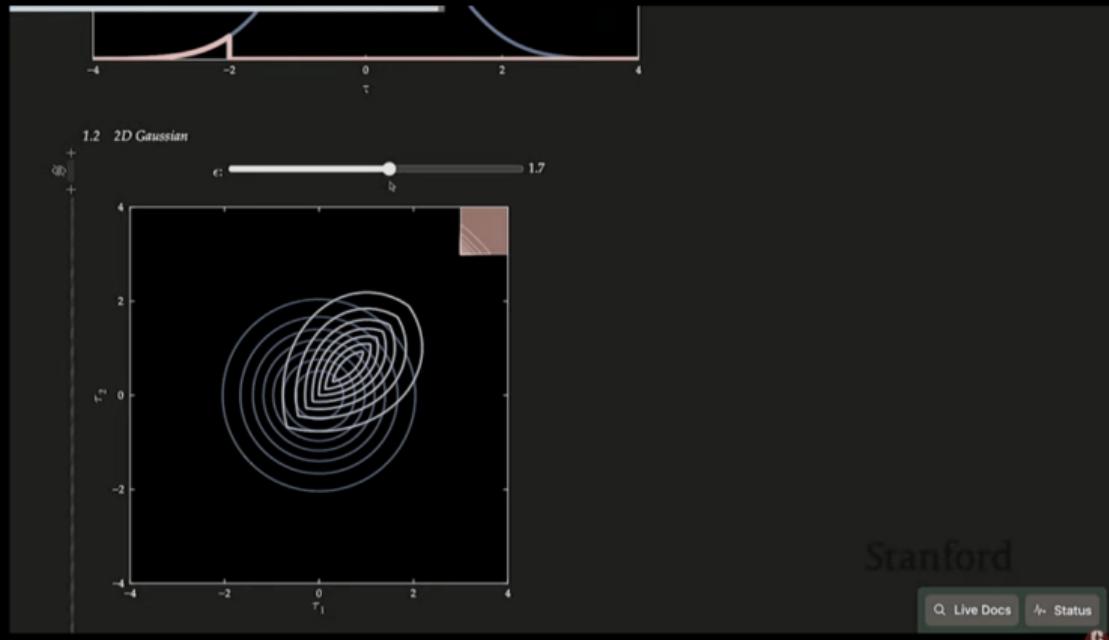
LECTURES



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

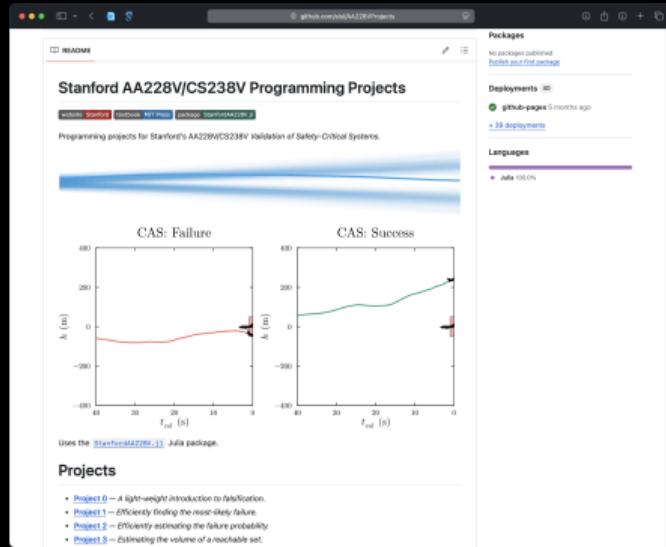
Uses the [StanfordMAster_11](#) Julia package.

Projects

- Project 0 — A light-weight introduction to telefication.
- Project 1 — Efficiently finding the near-shy 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

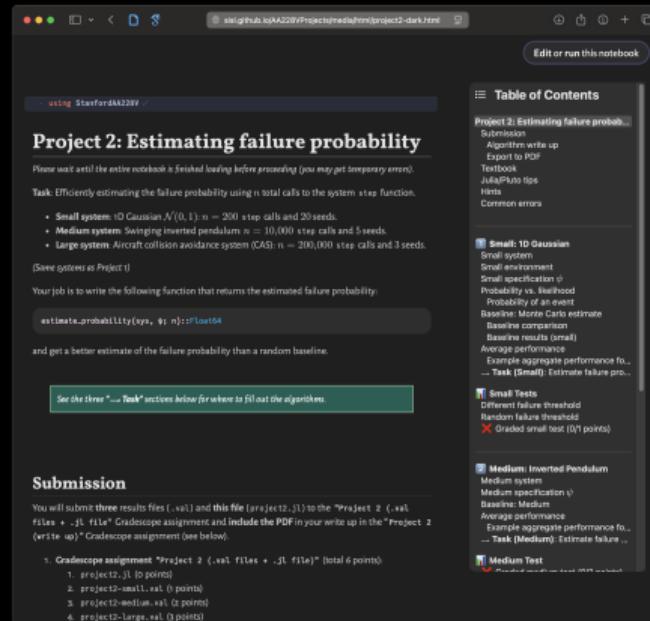
CAS: Success

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 ..
- 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 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.
- Code:** This will re-run most_likely_failure(:largeSystem, s) and re-save project3-large.vat. Uncheck this to load results from the file.
- Output:** 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 falling back towards the end point. Text above the plot indicates $\ell = -97.37071722814194$ and $n_{step} = 9.963$.
- 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
 - Deadline checker threshold
 - Random failure threshold
 - Slider to control threshold
 - Graded small test (1/1 points)
 - Algorithm for SmallSystem
- Bottom:** Live Docs, Status

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.
- Notes:** This will re-run most_likely_failure(::LargeSystem, s) and re-save project3-Large.vat. Uncheck this to load results from the file.
- Content Area:** CollisionAvoidance tests passed! (green background)
 - You found a passing trajectory!
 - $\delta = -97.37071722814194$ (failure log-likelihood)
 - $n_{step} = 9.963$ (step calls ≤ 10,000)
 - Most-likely failure found
 - A plot showing position A (m) vs time t_end (s). The curve starts at approximately (-100, -100), rises to a peak around (30, 100), and then levels off towards the end of the plot.
- Results:** Results saved for CollisionAvoidance
- File:** /Users/moser/Code/sis1/AA228V/Student/AA228VProjects/projects/project3/project3-Large.vat
- Instructions:** Please submit the file listed above to Gradescope.
- Bottom 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) (checked)
- Instructions:** Click to re-run the LargeSystem evaluation.
- Note:** 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 showing a red curve representing the most-likely failure trajectory over time t_{end} (s) from -10 to 40, and height A (m) from -300 to 300. The plot is titled "Most-likely failure found".
- 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
 - Baseline: 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 Imple...
- Large Tests:**
 - Large System Threshold
 - Random Failure threshold
 - Slider to control threshold
 - Graded small test (1/1 points) (checked)
 - Algorithm for SmallSystem
- Bottom:** 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

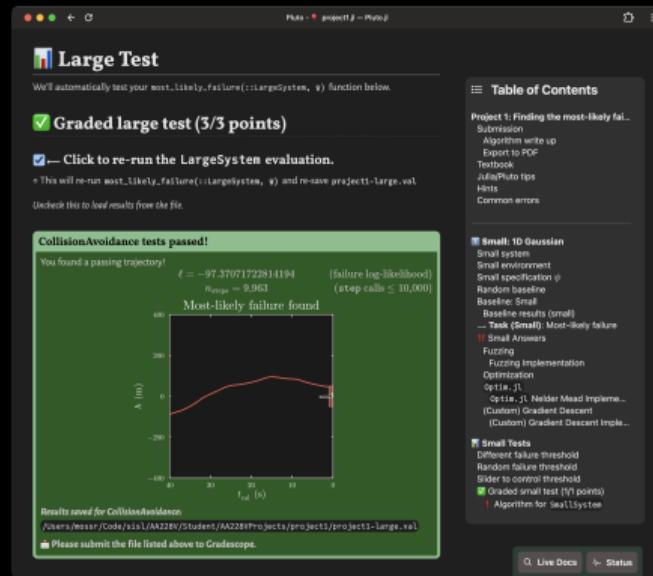
The screenshot shows a Pluto notebook interface with the following details:

- 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

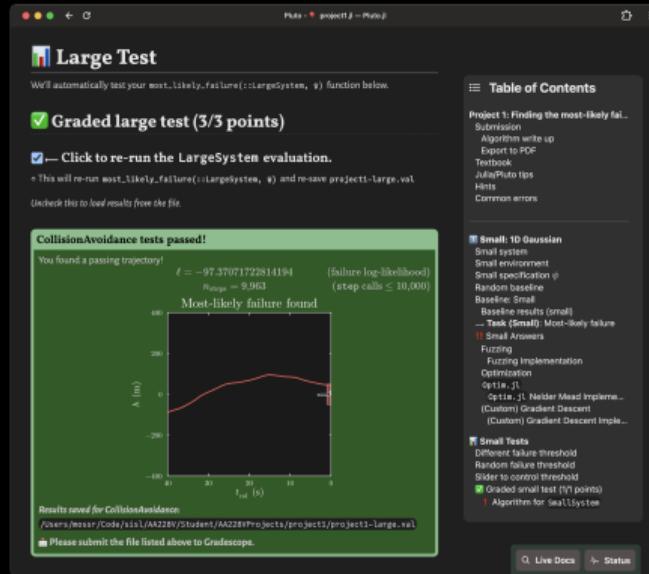
ASSIGNMENTS IN JULIA



- 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'.

The 'Autograder' section has a dropdown menu showing 'autograder_project1.zip' with options to 'Replace Autograder (.zip)' or 'Download Autograder'.

Below this are dropdown menus 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.'

At the bottom of this section are two buttons: '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 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 'Docker Image Status' (built on Jan 31, 2025). 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 a link to the Gradescope autograder specification. The repository has 7 stars, 2 forks, and 7 watching. It also lists releases, packages, and languages used.

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.ipynb[UntitledCell.ipynb]". Below the title, the page header reads "Algorithms for Validation". A "Lecture Introduction" section follows, containing a "1 Introduction" heading. This section discusses the importance of validation in high-stakes settings like autonomous vehicles and robotics. It highlights the challenges of validation as systems become more complex and the spectrum of behaviors widens. The text ends with a note about the need for validation from a historical perspective and its societal consequences.

Below the introduction, there's a "2 Probability Distributions" section. It starts with a brief description of the univariate normal distribution, followed by its mathematical formula:

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

Next is a figure titled "Figure 1: The normal (Gaussian) distribution", which shows a bell-shaped curve centered at $\mu = 0$ with a standard deviation of $\sigma = 1$. The x-axis ranges from -4 to 4. Below the figure are two sliders: one for μ set to 0.1 and one for σ^2 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

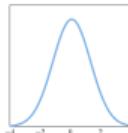
Before designing decision-making processes in high-stakes settings, it is important to ensure that they will operate as intended. We refer to the process of analyzing the behavior of these systems as validation. Validation is a critical component of the development process for decision-making systems 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.

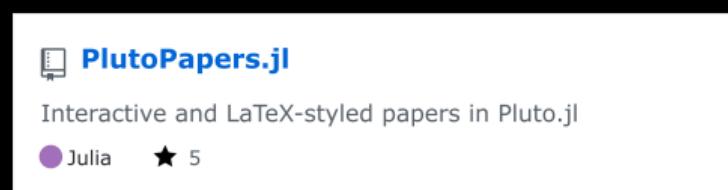
2 Probability Distributions

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

$\mu =$
 $\sigma^2 =$



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

Interactive research papers
Run code directly in the paper

THANK YOU!



QUESTIONS?

mossr@cs.stanford.edu