

Fast-tracking closed-loop reaction with Stuntman

Introduction

Optimizing a reaction, formulation, or process to its best conditions usually means running experiment after experiment: design a set of conditions, execute them, measure the outcome, learn from the result, and repeat. Done manually, each cycle depends on a scientist reviewing the last round's data, deciding what to try next, and setting up the following experiment by hand. It's slow, and it pulls scientists away from higher-value work.

Even standard lab automation does not solve this. Hands-free hardware can execute a fixed protocol, but design, execution, and analysis often run on separate, disconnected systems, leaving a scientist to stitch together fragmented data, review the results, and decide what to run next by hand before the following cycle can start. Closed-loop automation closes that gap by letting the results of one experiment automatically shape the next.¹ Stuntman (Figure 1) combines built-in AI with fully modular hardware to support closed-loop experimentation from workflow development through execution, analysis, and optimization. Stuntman AI can generate and modify the Python scripts that connect these steps, while validated scripts and the selected optimizer control the experiment during execution.

This App Note describes how Stuntman's closed loop was used to optimize the enzymatic hydrolysis of sucrose. It shows how a Bayesian optimizer built with scikit-optimize finds and holds high-scoring reaction conditions across a full run. It also shows how the same architecture extends to other workflows across small molecules, materials, and biologics.

Methods

Stuntman was configured to optimize the hydrolysis of sucrose catalyzed by invertase (β -fructofuranosidase; EC 3.2.1.26).^{2,3} The closed-loop cycle



Figure 1: Stuntman: native AI-driven, fundamentally flexible automation for closed-loop experimentation.

was handled using a series of short python scripts (Figure 2). Stuntman AI was used to generate and refine the Python scripts that coordinated the closed-loop workflow. Starting from natural-language descriptions of the experimental objective and workflow, Stuntman AI helped create the scripts for configuration handling, design generation, execution, analysis, evaluation, and optimization. The scripts were reviewed and validated before the experiment was initiated. During the run, the validated scripts coordinated each cycle, while the Gaussian-process Bayesian optimizer selected the experimental conditions.

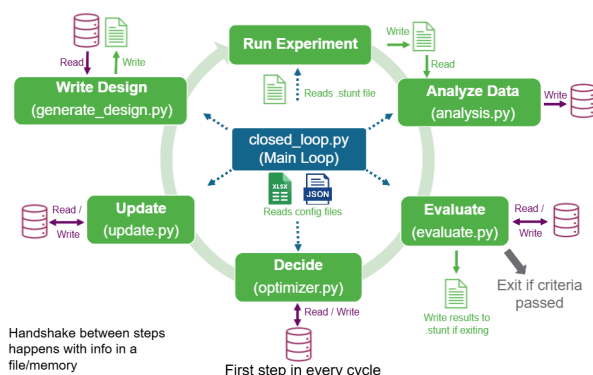


Figure 2: Stuntman's closed-loop workflow architecture.

Item	Name	Source Value	Unit	Sample Type	Constant Sample Value	Source Secondary Value	Sample Secondary Value	Secondary Unit
1	Water	55.5	M	Diluent				
2	Enzyme	333	μM	Variant				
3	NaCl	2500	mM	Variant				
4	Phosphate pH 6.0	6	pH	Variant		500	50	mM
5	Phosphate pH 9.0	9	pH	Variant		500	50	mM
6	Sucrose	250	mg/mL	Constant	25			

Table 1: Materials, roles and their properties for sucrose hydrolysis experiment.

Item	Name	Value	Unit
1	Samples per cycle	12	
2	Points per sample	4	
3	Amount/volume per sample	500	μL
4	Reaction library	Reaction Mixtures	
5	Measurement library	Stunner Plate	
6	Custom amount calculation	FALSE	
7	Pause after every cycle	FALSE	
8	Simulation	FALSE	
9	Substrate extinction	1992	$\mu\text{M}/\text{unit}$
10	Quench library	Quenched Reactions	

Table 2: Experiment settings for sucrose hydrolysis experiment.

Item	Name	Value	Unit	Sign
1	Maximum cycles	12		=
2	Velocity	10	$\mu\text{M}/\text{sec}$	>=
3	Linearity	0.99		>=
4	Score	1		>=

Table 3: Optimization criteria for sucrose hydrolysis experiment.

Every cycle runs through the same lightweight architecture – write design, run experiment, analyze data, evaluate outcomes, decide next experiment parameters, update parameters and loop. First a script called `closed_loop.py` reads Stuntman's configuration file, which defines the experiment and its parameters.

The configuration file can be created in XLSX or JSON format. For the sucrose hydrolysis experiment, it specified the materials, their roles and properties (Table 1), the experimental settings (Table 2), and the optimization criteria (Table 3). The materials included invertase at 333 μM , NaCl at 2500 mM, and phosphate buffers at pH 6.0 and 9.0, and sucrose at 250 mg/mL. Water served as the diluent, and sucrose was held at a constant experimental concentration of 25 mg/mL.

Optimization criteria were defined for a maximum of 12 cycles, requiring reaction velocity $\geq 10 \mu\text{M}/\text{sec}$ and linearity ≥ 0.99 , with each candidate scored on a composite scale. A composite score ≥ 1 was required to pass.

Experimental settings included running twelve reactions in each cycle, with sampling at four time points – 0, 10, 20, 30 minutes – and analyzing on an integrated Stunner.

The deck (Figure 3) was configured with three types of libraries: a reaction plate holding the Reaction Mixtures library, a quench plate holding the Quenched Reactions library to stop each sampled time point, and an analysis plate (Stunner Plate) where UV/Vis measurements determined reaction progress using a substrate extinction coefficient of 1992 $\mu\text{M}/\text{unit}$.



Figure 3: The Stuntman deck staged with reaction, quench, and analysis plates during a closed-loop run.

Next, `closed_loop.py` passes the configuration details to `optimizer.py` which decides what experimental conditions to run. Candidate conditions for each cycle were proposed by a Gaussian-process Bayesian optimizer implemented with Scikit-Optimize.⁴ The optimizer was set to run 12 candidates per cycle for up to 12 cycles. Each cycle contained 12 candidates: one fixed control, the best-performing non-control condition identified so far, and 10 new conditions proposed by the optimizer. The optimizer used a three-phase strategy: exploration rate was set at 90% for cycles 1–4, decreased to 50% for cycles 5–8, and decreased again to 10% for cycles 9–12 (Figure 4). This stepwise schedule favored broad exploration early in the run, balanced exploration and exploitation in the middle cycles, and focused primarily on high-scoring regions during the final cycles.

The conditions selected by `optimizer.py` are written to `update.py`, and passed to `generate_design.py`, which generates a python script that is converted into `.stunt` file for execution. The experiment was initiated remotely through the Stuntman Application Programming Interface (API). Stuntman automatically executed aspiration and dispense of the various reagents, mixing the reactions, collecting and quenching timepoint samples, transferring the samples to Stunner analysis plates and moving the plates to Stunner for analysis. Stuntman execution data and Stunner absorbance results were captured in a single `.stunt` file.

After execution, `analysis.py` takes over. In this workflow, analysis script converted the timepoint

absorbance data for each sample into reaction velocity and linearity values. The `evaluate.py` script then compared the results with the stopping criteria defined in the initial configuration file. For this experiment, the stopping criterion was the maximum number of cycles, although thresholds for reaction velocity and linearity could also have been specified. If the stopping criteria were met, the loop ended and saved the results to the `.stunt` file. Otherwise, the loop returned to `optimizer.py` to select the next set of experimental conditions. This cycle continued until the stopping criteria were met or the user terminated the experiment.

```
# Exploration schedule: (1-4) 90.0 (5-8) 50.0 (9-12) 10.0

import pickle
from typing import Optional
from scipy.stats import qmc
from skopt import Optimizer
from skopt.learning import GaussianProcessRegressor
from skopt.learning.gaussian_process.kernels import Matern
from skopt.space import Real

class OptimizationMain:

    # -----
    # Constants
    # -----

    CANDIDATES_PER_CYCLE = 12
    OPTIMIZER_CANDIDATES = 10

    CONTROL = {'Invertase': 30, 'pH': 7.5, 'NaCl': 100}

    ORIGINAL_BOUNDS = {
        'Invertase': (5.0, 100.0),
        'pH': (6.0, 9.0),
        'NaCl': (0.0, 2500.0),
    }

    PHASE_KAPPA = {1: 8.0, 2: 4.05, 3: 0.1}

    PHASE_STEPS = {
        1: {'Invertase': 10, 'pH': 0.3, 'NaCl': 100},
        2: {'Invertase': 3, 'pH': 0.1, 'NaCl': 30},
        3: {'Invertase': 1, 'pH': 0.03, 'NaCl': 10},
    }

    ZOOM_DELTAS = {
        2: {'Invertase': 15, 'pH': 0.5, 'NaCl': 420},
        3: {'Invertase': 5, 'pH': 0.165, 'NaCl': 140},
    }
```

Figure 4: Snippet of `optimizer.py` showing the exploration-exploitation schedule, variable ranges and step sizes.

Results

Rapid and sustained convergence

The optimizer rapidly identified high-scoring reaction conditions and continued selecting them throughout the remainder of the run (Figure 5). Scores initially spanned a broad, predominantly low-scoring range, reflecting exploration of the condition space. By approximately sample 45 during the fourth cycle, the scores converged into a narrow, high-performing band and remained there. This sustained performance indicated that the optimizer had identified and consistently



$$\text{Score} = 0.6 \times \frac{\text{Velocity}}{\text{Pass Velocity}} + 0.4 \times \frac{\text{Linearity}}{\text{Pass Linearity}}$$

Figure 5: Composite score by sample across the closed-loop run. Scores converged into a consistently high-performing range by approximately sample 45.

exploited a promising region of the experimental space rather than finding a single high-scoring condition by chance.

End-to-end autonomous execution

The run also demonstrated autonomous iteration across the complete design–execute–measure–learn cycle. No scientist was required to review results or select conditions between cycles. After each cycle, the analysis script processed the absorbance data and calculated reaction velocity and linearity. The evaluation script scored the results, and the optimizer used them to select the next batch of conditions. The workflow generated the corresponding design, which Stuntman executed automatically. This autonomous cycle continued until the pre-defined stopping criterion was reached. Inputs, selected conditions, execution data, and results were retained in the .stunt file for subsequent review.

Configuration-driven operation

The run demonstrated configuration-driven control of the sucrose hydrolysis experiment. The materials and their roles and properties, optimization criteria, experiments per cycle, and maximum number of cycles were defined in XLSX or JSON configuration files and read by closed_loop.py. This kept key experimental inputs separate from the workflow scripts, allowing them to be reviewed and adjusted without changing the sequence of the closed loop.

The Python scripts continued to provide the workflow-specific optimization, analysis, evaluation, and execution logic.

Extending the closed loop to other workflows

The same design–execute–measure–learn framework demonstrated here can be adapted to other applications by changing the experimental configurations, execution methods, analytical measurements, and evaluation logic. In small molecules and chemistry, it can support reaction screening, optimization, and preformulation studies. In materials and energy, it can be applied to electrolyte formulation, polymer development, and catalyst screening.⁵ In biologics & gene therapy, Stuntman’s closed loop can drive multi-parameter developability screens – identifying the best structure/formulation pair across quantification, sizing, and stability readouts, including anti-body-drug conjugate (ADC) developability screens that additionally track drug-to-antibody ratio, or DAR.

Conclusions

Stuntman closed the loop across experimental design, execution, measurement, and optimization without requiring a scientist to review results or select conditions between cycles. In this sucrose hydrolysis case study, a Gaussian-process Bayesian optimizer converged on a consistently high-performing region by approximately sample 45—during the fourth cycle—and continued selecting high-scoring conditions throughout the remainder of the run. Configuration files kept key experimental inputs separate from the workflow-specific optimization, analysis, evaluation, and execution logic, making the experiment easier to adjust and review. Stuntman AI supported development of the workflow scripts. During the run, the validated scripts controlled execution, and the Bayesian optimizer selected the experimental conditions. This study demonstrates a foundation for autonomous experimentation that can be adapted to reaction and formulation development and other workflows across small molecules and chemistry, materials and energy, and biologics and gene therapy.

References

1. Tom, G., et al., Aspuru-Guzik, A. Self-Driving Laboratories for Chemistry and Materials Science, *Chem. Rev.*, 2024; 124(16):9633–9732.
2. International Union of Biochemistry and Molecular Biology. EC 3.2.1.26: β -Fructofuranosidase. Enzyme Nomenclature. <https://iubmb.qmul.ac.uk/enzyme/EC3/2/1/26.html> (accessed August 26, 2026).
3. Faurschou, N.V., Taaning, R.H., Pedersen, C.M. Substrate specific closed-loop optimization of carbohydrate protective group chemistry using Bayesian optimization and transfer learning, *Chem. Sci.*, 2023; 14(23):6319.
4. Desimpel, S., Dorbec, M., Van Geem, K.M., Stevens, C.V. Bayesian optimization for chemical reactions, *Chem. Soc. Rev.*, 2026; 55(5):2731–2775.
5. Schilter, O., Pacheco Gutierrez, D., Folkmann, L.M., et al., Laino, T. Combining Bayesian optimization and automation to simultaneously optimize reaction conditions and routes, *Chem. Sci.*, 2024; 15(20):7732–7741.



Unchained Labs
 4747 Willow Road
 Pleasanton, CA 94588
 Phone: 1.925.587.9800
 Toll-free: 1.800.815.6384
 Email: info@unchainedlabs.com

© 2026 Unchained Labs. All rights reserved. The Unchained Labs logo, Stunner, and the Stunner logo, Stuntman, and the Stuntman logo are trademarks and/or registered trademarks of Unchained Labs. All other brands or product names mentioned are trademarks owned by their respective organizations.

Rev C