Universiteit Leiden
The Netherlands

Opleiding Informatica

Optimizing Automated, Low-Volume Liquid Transfers

Rick Wierenga (3354687)

Supervisors:
Thomas Moerland & Fons Verbeek & Kevin Esvelt

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

July 11, 2024

Optimizing Automated, Low-Volume Liquid Transfers

Rick Wierenga

July 2024

Abstract

While liquid handling robots are ubiquitous in high-throughput molecular biology assays, their precision and accuracy at low-volume transfers limit their utility in many small-scale yet repetitive manual processes in the laboratory. Here, we present *lil optimizer* (low-volume incremental liquid-transfer optimizer), an automated, hardware-agnostic system that can efficiently optimize the precision and accuracy of low-volume liquid transfers in a closed loop system based on an arbitrary performance evaluation method. The system is compatible with any stochastic, continuous, n-dimensional, black box optimizer and has been validated with Bayesian optimization, policy gradient methods, and hierarchical optimistic optimization, concluding that the last of those should generally be preferred. With gravimetric evaluation, we efficiently optimize 500 nanoliter transfers, an order of magnitude lower than conventionally recommended in the field, and reach superhuman performance on 1uL transfers. We hope *lil optimizer* will accelerate biological and biomedical research by enabling the automation of common, low-throughput liquid handling tasks. Source code is freely available at `https://github.com/rickwierenga/lil-optimizer`.

Contents

1 Introduction

Pipetting, the fundamental operation in basic biological research essentially consists of four steps: mounting a disposable tip onto a pipette, aspirating liquid into the tip, dispensing the liquid out of the tip, and then discarding the tip. Liquid handling robots automate this pipetting process, typically using a head on a 2D gantry on which one or multiple pipetting channels are mounted. The head moves over a deck where tips, microplates, tubes and other containers are placed. Like manual pipetting, robotic channels often use an air displacement mechanism and mount disposable plastic tips.

Liquid handling robots are most commonly used for automating high-throughput assays, primarily in industrial and clinical settings [1, 2]. More advanced experiments use closed-loop control systems for dynamically updating protocols mid-run, for example to maintain bacterial cell cultures or to autonomously explore protein fitness landscapes [3, 4, 5]. At the same time, many biologists still spend a considerable amount of time manually pipetting. One reason for this is that liquid handling robots are notoriously hard to program and often require 'lab automation engineers' as specially trained middlemen between scientists and robots. This layer of humans decreases iteration time for scientists considerably, thus lowering their ability to automate protocols. PyLabRobot [6] aims to make programming robots easier and more accessible to a wide audience while simultaneously giving programmers more control.

Another reason why liquid handling robots are not widely used for low-throughput, day to day tasks is that by default many liquid handling robots have insufficient accuracy and precision at the many required low-volume transfers (¡5uL). This problem is recognized and robot manufacturers like Hamilton conveniently sell optimization platforms at an estimated cost of \$20.000 dollars, (just 25% of which is for hardware; the remaining \$15k are software costs)[1]. This proprietary program is inflexible because it only works with Hamilton robots and a specific Mettler-Toledo scale, it does not support volume-based parameter adjustments, and it does not support channel-specific calibrations. Manual optimization and Design of Experiments (DOE) are alternative strategies for optimizing low-volume transfers[7]. Due to higher iteration time and low walkaway time, these methods only afford a limited number of iterations which can be insufficient for attaining the required accuracy and precision.

The problem of expensive, stochastic black-box optimization, such as the problem of liquid transfer calibration, is well-studied in computer science and many approaches exist, such as hierchical optimistic optimization (HOO) [8], vanilla policy gradients (VPG) [9], Bayesian optimization (BO) [10, 11, 12], genetic algorithms (GA) [13], simulated annealing (SA) [14], particle swarm optimization (PSO) [15], and covariance matrix adaptation evolution strategy (CMA-ES) [16]. In essence, all of those methods evaluate an expensive, non-differentiable

[1] `https://www.hamiltoncompany.com/automated-liquid-handling/small-devices/liquid-verification-kit`

function and suggest the most informative next trial. While it is intuitive that those methods can be used to iteratively improve low-volume liquid transfers, this has not been done due to the historically closed nature and developer-unfriendly ecosystem of lab automation software. In this project we evaluate whether these methods can be used by answering the research question *which stochastic black-box optimization algorithms, if any, can be used to reliably and efficiently optimize automated, low-volume liquid transfers?*

Here, we introduce 'low-volume iterative liquid-transfer optimizer' (*lil optimizer*): an open source, efficient, closed-loop, and hardware agnostic low-volume liquid transfer optimizer. By iteratively optimizing precision using any arbitrary blackbox optimizer, like the ones described above, and optimizing accuracy using linear translations, transfers of arbitrary performance can be reached, including on volumes as low as 500nL. Thanks to its flexible nature, and being built on PyLabRobot, *lil optimizer* overcomes limitations of proprietary software and can be used for volume-based and channel-specific parameterization.

Lil optimizer has been tested with HOO, VPG and BO, all of which lead to acceptable results. While training with HOO is generally the most stable and the fastest, VPG is architecturally preferred because it is most easily extensible to non-context-free settings, and Bayesian optimization has the best validation performance. We further find, both using full experiments and a cheap heuristic, that doing more transfers per estimation of transfer precision, a hyperparameter of *lil optimizer*, can improve performance while not meaningfully impacting running time. After optimization with *lil optimizer*, the tested liquid handling robot no longer fails at microliter transfers but instead reaches statistically significant superhuman performance at those transfers. We hope that *lil optimizer* accelerates biological discoveries by improving the efficiency and reproducibility of low-throughput experiments through more viable automation (GitHub: `https://github.com/rickwierenga/lil-optimizer`).

2 Related work

Little has been published on the problem of liquid transfer optimization itself, presumably due to the historically closed nature of lab automation software. We do, however, start by introducing automated liquid transfers and how to define performance objectives. We then provide the theoretical, mathematical context around this problem, describe some of its characteristics, and how related problems are addressed elsewhere.

2.1 Background on liquid transfers

Each liquid transfer is fundamentally parameterized by a number of discrete and continuous parameters, such as the aspiration and dispense flow rates and the volume of air that is pre-aspirated to provide additional blow out on dispense. For automated transfers, the exact parameters depend on the capabilities of the hardware used. This currently ill-defined set of continuous parameters form the input space. The goal is find a point in this space, a 'parameter set', for which liquid transfers are both precise (close together) and accurate (close to the desired value).

The output of liquid transfers can be measured in various ways, most commonly with a scale (gravimetric analysis) or a microplate reader (e.g. photometric analysis), both yielding real-valued scalars. The difference of the measurement before and after the dispense, w_1 and w_2 respectively, is the 'dispense score'. As mentioned above, even though this score is measured as a scalar, there are two components to a liquid transfer performance. First, the accuracy measures the closeness to the target value to the mean as the absolute difference of the measured mean and the target mean. In contexts where the volume of a transfer is variable, the 'trueness' is a volume-independent quantity defined by the average absolute error divided by the target volume (known as the R-score, often a percentage). Second, the precision refers to the repeatability of measurements as given by the standard deviation of a number of transfers. A volume-independent quantify of precision is the coefficient of variation (CV), defined by the ratio of standard deviation to the mean. Both the accuracy and precision need to be optimized [17], making this a double-objective optimization problem.

2.2 Stochastic black-box optimization

The problem of liquid transfer optimization has several interesting mathematical properties that make it difficult. First, there is no known function that defines the mapping of the input space (transfer parameterization) to the output space (accuracy and precision), meaning no gradients can be computed (making this a 'black box problem'). Next, both the pipetting operations and measurements are noisy (making this problem 'exotic' and 'stochastic'). Further challenges are posed by the parameters not being independently optimizable: the parameters interact to change the final output (making this problem 'non-separable'). Liquid transfers in a given tip depend on the history of liquids in that tip, giving rise to a problem ‘context’, although this effect is not further studied here; each transfer is done in a ‘context-free’ manner. Intuitively, one might expect the output landscape to be reasonable convex, but this has not been validated and cannot be assumed. Finally, the input space is potentially unconstrained, and almost certainly consists of a number of continuous dimension, making table-based optimization unpractical.

Problem with these characteristics (stochastic, non-separable, non-convex, black box optimization) are common in reinforcement learning and are abstractly known as ‘context-free continuum-armed bandits’ [18]. A real-world analogy of such problems is sketched with a casino with an infinite grid of slot machines, where the challenge is to find which machine to pull for the highest average reward. Note that individual pulls might be considerably below or above the average reward for a given machine [9] (stochastic). The inherent characteristic of the machine, its true mean value, is not known and cannot be analyzed except by ‘pulling’ (black-box). The problem being ‘context-free’ means that the inherent, unknown mean output of each machine is constant over time.

In some settings, a fixed pulling budget is given and the cumulative reward over the entire optimization process needs to be maximized. These are known as ‘regret minimization problems’. This introduces an interesting exploration vs exploitation trade-off because one might either aggressively pull levers presumed to be good (exploitation) or risk bad pulls for the chance of finding a better lever (exploration). In other settings, one is interested in having maximum confidence in having found the best arm with minimal function evaluations. While the exploration vs exploitation trade-off persists in the sense of determining whether the algorithm should look at entirely unexplored regions or look near regions known to be promising, the algorithm is always exploring new inputs (there is no reason to repeat an input) which some describe as ‘pure exploration’[19]. The problem of liquid transfer optimization studied here is of the latter type.

3 Baseline hardware characterization

In this section, we quantify relevant characteristics of the hardware used to provide intuition for the maximal performance we can reach with this setup. Additionally, we measure the baseline accuracy and precision of automated liquid transfers using default parameters, and the accuracy and precision of expert human pipetting. These are targets which we aim to surpass.

3.1 Characterization of the analytical scale

The scale used in the following experiments is the Mettler Toledo WXS205SDU. This scale has a readability of 0.01 mg and an advertised repeatability (std) of 0.07 mg at nominal load. This scale was chosen because it is used in the Hamilton Liquid Verification Kit (LVK), meaning many prospective *lil' optimizer* users are likely to already have one. Second, this scale has good specifications. Its performance was verified locally using the Rice Lake '12504 20 Piece Stainless Steel Calibration Metric Test Weight Set', which meets the National Institute of Standards and Technology (NIST) Class F specification for commercial "Legal-for-Trade" weighing operations[2]. The results are given in Table 1. The standard deviation of some tens of milligrams (tens of nanoliters of water) provide the lower bound on the precision that can be reached. In all experiments, PyLabRobot was used as the interface to the scale.

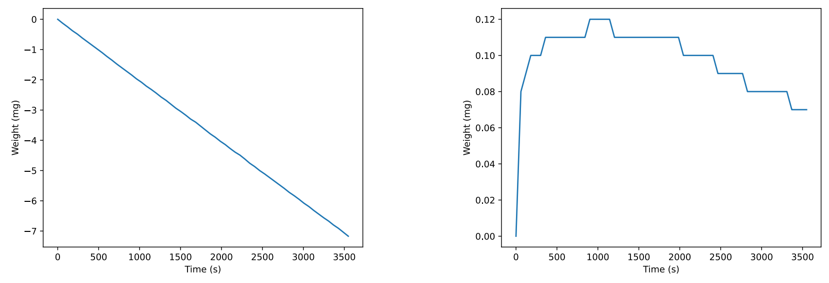
Weight (mg)	Mean (mg)	STD (mg)	Min (mg)	Max (mg)
1	1.057	0.085	0.930	1.220
2	2.042	0.026	2.000	2.100
5	5.055	0.047	4.980	5.130
10	10.017	0.039	9.960	10.070
100	100.050	0.013	100.030	100.070
1000	999.831	0.032	999.790	999.900

Table 1: Empirical characterization of the Mettler Toledo WXS205SDU scale using Rice Lake test weights at various loads. Each weight was measured N=10 times. Weights were handled using the tweezers to avoid contamination.

3.2 Evaporation rate and scale drift

Evaporation of water in the test tube was manually verified by taking a weight measurement every 60 seconds over an hour long window, during which -7.170 mg of water evaporated. To correct for scale drift, the same program was run for another hour without liquid on the scale, during which the scale drifted by +0.07mg (Fig 1). Ignoring the negligible drift in the scale, 0.03 mg of water evaporates per 15 seconds (an estimated average duration of a dispense). This value could be relevant for long dispenses, or when target precision or accuracy are near this magnitude, in which case a correction can simply be added

[2]`https://www.amazon.com/gp/product/B006MWG13U/`



(a) Evaporation (b) Scale drift

Figure 1: The evaporation rate of water on the scale is 7.170 mg / h; the scale drifts by +0.07mg over this time frame. Evaporation could impact results when a lot of time passes between measurements, or when targeting extremely low thresholds, in which case it is easily corrected for. Such correction was not applied in experiments described here. Drift is unlikely to impact measurements.

back into the post-dispense measurement. In the following experiments, such correction was not performed.

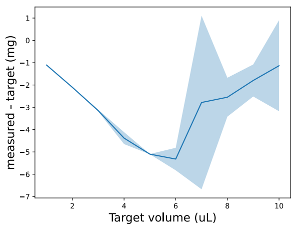
3.3 Characterization of liquid transfers using before optimization

VENUS, the default proprietary program typically used for programming the liquid handling robot used in these experiments, has pre-optimized transfer parameters for a number of liquids, including water, and for the tips one might use. These liquid transfer parameters were ported to PyLabRobot, using which we performed transfers at various volumes to provide baseline pipetting accuracy. These experiments were conducted using official Hamilton Co-Re II tips. The results are given in Figure 2 and the tables in Appendix A, providing a reference for deciding when further optimization is necessary.

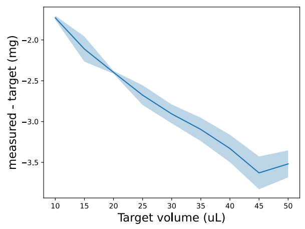
It is noteworthy that for $\leq 5\mu L$ transfers using 10 μL tips, the dispense consistently fails entirely (Fig. 2a). This is seen in the error being the same size as the target volume with a low standard deviation. Between 5 and 10 μL, the accuracy error ranges from approximately 80% to 10%, decreasing with target volume, while the standard deviation remains around 10%. This demonstrates that transfers at volume around this order of magnitude are impossible without optimization surpassing default performance.

3.4 Inter-channel variance in accuracy and precision

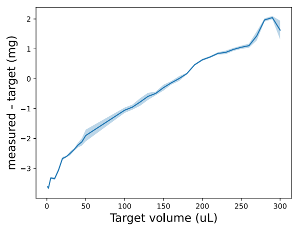
The robot used in these experiments has 8 independent channels. There is a significant difference between certain channels in terms of accuracy (between channel 0 and 7: $p < 0.0005$) (Fig. 3). For this reason, transfers for channels



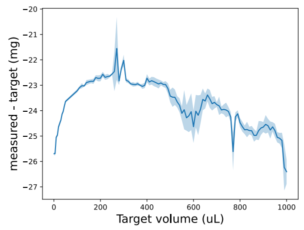
(a) 10uL



(b) 50uL



(c) 300uL



(d) 1000uL

Figure 2: Characterization of liquid transfers using default parameters for 10, 50, 300 and 1000uL tips. For 10uL transfers, the transfer fails consistently (as seen by the error being of the same size as the target volume).

have to be optimized independently. This is noteworthy because this is impossible with Hamilton's liquid transfer optimizer program LVK and proprietary control software VENUS, even though it is clearly needed. The fine-grained control granted by PyLabRobot does allow channel-specific parameters, and *lil optimizer* does one channel at a time.

3.5 Human pipetting accuracy and precision

To compare optimized robotic pipetting against human pipetting, 9 human scientists performed 10 1uL transfers using the pipette they would normally use for transfers of this volume. The results are given in Table 2. The observed mean and STD provide target values for *lil' optimizer*.

Let it be highlighted that even though the macro-averaged mean is close to 1, implying high accuracy, the means of the 10 transfers performed by individual scientists might differ significantly from this mean. For example, one scientist had a mean of $0.876mg$ which is significantly lower than the expected mean of $1mg$ ($p < 10^{-5}$). While these differences are easily correctable with pipette calibrations, and this difference is likely the result of incorrect calibration and not a skill issue, this result shows that in practice pipettes often go uncalibrated and that human-performed transfers might have significant error in accuracy.

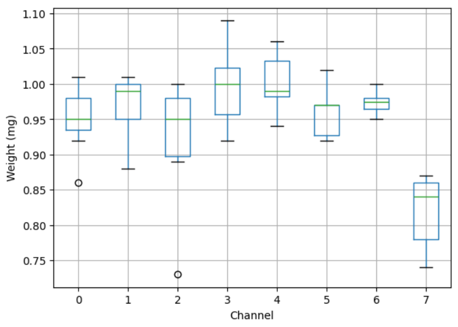


Figure 3: There is a significant between the mean in channels pipetting channels when performing the "same" transfer (between channel 0 and 7: $p < 0.0005$).

Statistic	Value
N	9
Macro-averaged mean	0.97 mg
Macro-averaged STD	0.08 mg
Min mean	0.88 mg
Max mean	1.12mg

Table 2: Statistics of 9 expert human scientists performing 10 1uL transfers each. The macro-averaged mean and standard deviation are given, as well as the most extreme mean observed for individual scientists.

4 Automatic optimization

Optimization of liquid transfers requires simultaneous optimization of accuracy and precision. In *lil optimizer*, this is done using a double loop where the inner loop minimizes the standard deviation across N transfers by sequentially evaluating and updating a transfer parameter set, and the outer loop applies a linear transformation to shift the accuracy into an expected acceptable range. In this way, the double-objective optimization, which is difficult, is effectively split into two separate optimization problems: most parameters affect precision and are optimized with an arbitrary STD minimizer, and accuracy is simply corrected with translations to the volume requested on the machine, which is expected to map onto the observed mean linearly. Combining both objectives into a single objective is theoretically possible, but would introduce more hyperparameters (how is each weighed?) and restricts user freedom to specify thresholds for each. The program ends when both accuracy and precision reach the user-defined threshold (Algorithm 1; Fig. 4).

By default, liquid transfers are optimized for a single pipetting channel and a single volume, but the program can simply be run multiple times for extending results to other volumes and channels. Here, the search space of future optimizations can easily be constrained using results found in a previous optimization to accelerate the process. This can also be done to fine-tune previous calibrations for changed environmental conditions, where necessary.

4.1 Accuracy maximization

The score accuracy is the absolute difference between the average measured value m_m of the previous set of N liquid transfers (where the standard deviation is less than or equal to the threshold $m_s \leq \delta$) and the target value v_t, i.e. $|m_m - v_t|$. This quantity is minimized until some threshold ϵ is reached by simply adjusting the volume sent to the machine v_m which is expected, by physical principles, to linearly shift the mean of observations m_m. This design choice proved to work well in practice (see Results). Since optimizing precision after an accuracy-shift might re-introduce precision-inefficiencies, the precision optimizer is always run after an accuracy adjustment. In the best case, the precision check exits in one STD check (if still $m_s \leq \delta$) and the result is immediately used by the accuracy maximization algorithm. However, if $m_s > \delta$, it is first adjusted to provide a more accurate mean measurement (see 'Precision maximization' below). The accuracy shift might not have led to a desired mean measurement, possibly necessitating multiple translations.

The translation needed is computed differently based the number of iterations done. With one recorded mean m_m (iteration 1), the volume sent to the machine is corrected using the absolute error $(v_t - v_o)$ (intuitively: how much is missed) and the expected efficiency $\frac{v_m}{m_m}$ (intuitively: how much of the requested liquid is expected to be dispensed), forming the first update rule: $v_m + (v_t - v_o) \cdot \frac{v_m}{m_m}$. With two or more recorded values, a linear model is fitted over the list of past

Algorithm 1 Double optimization loop for the simultaneous optimization of accuracy and precision, until thresholds ϵ and δ respectively. v_t is the target volume measurement, v_m is the current volume sent to the machine, starting at v_t; m_m and m_s are the mean and standard deviation of the latest STD check; p is the set of transfer parameters.

```
 1: $v_m \leftarrow v_t$
 2: $p \leftarrow ...$
 3: $m_s \leftarrow \infty$
 4: while $|m_m - v_t| > \epsilon$ or $m_s > \delta$ do
 5:     while $m_s > \delta$ do
 6:         $p \leftarrow$ std_minimizer.next_trial()          ▷ Arbitrary STD minimizer
 7:         $m_m, m_s \leftarrow$ perform_transfers($v_m, p$)
 8:         $p \leftarrow$ std_minimizer.score(m_s)
 9:     end while
10:     if $|m_s - v_t| > \epsilon$ then          ▷ Linear shift for correcting accuracy.
11:         linear_model.update(($v_m, m_m$))
12:         if first iteration then
13:             $v_m \leftarrow v_m + (v_t - m_m) \cdot \frac{v_m}{m_m}$          ▷ Ratio-adjusted volume update
14:         else
15:             $v_m \leftarrow$ linear_model.predict_y($v_t$)          ▷ Update from linear model
16:         end if
17:         $m_s \leftarrow \infty$
18:     end if
19: end while
```

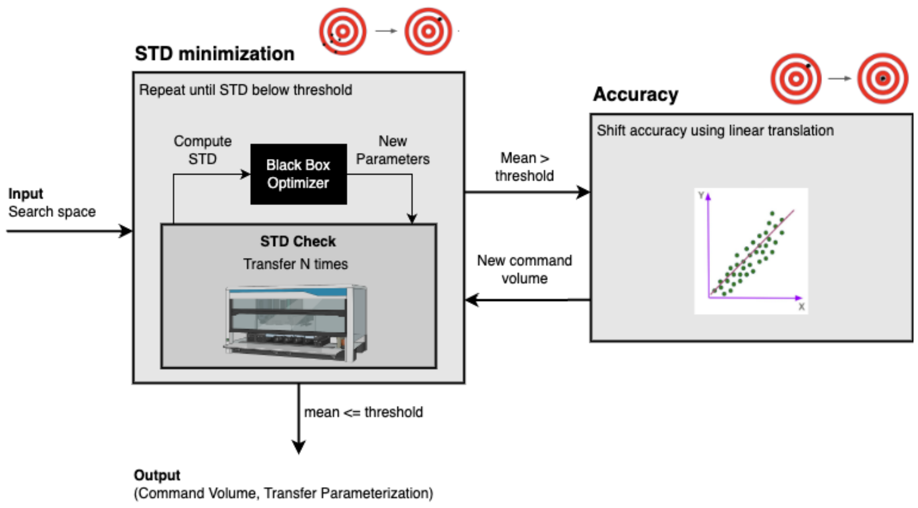


Figure 4: Systematic representation of *lil optimizer*. The algorithm repeatedly minimizes STD until it is below a certain threshold using an arbitrary black-box optimizer on a user-defined parameter space. STD is evaluated using an 'STD Check', consisting of N transfers (a hyperparameter). After the desired STD is reached, the mean of the last STD check is evaluated. If it is above the threshold, the volume requested on the machine is adjusted using a linear translation, following which STD is minimized again, starting from the current parameter-set. When the mean is sufficient, the *lil optimizer* returns the found parameter-set as well as the volume correction that needs to be applied.

(v_m, m_m) tuples, enabling fast convergence on the prediction of any following v_m. A linear curve is expected to be the most realistic model in this setting, and proves more robust to noisy measurements than the simple update rule that only looks at the last data point (data not shown).

4.2 Precision maximization

Maximizing precision is done by minimizing the standard deviation over a set of N transfers (an STD check). Contrary to the linear shift for adjusting accuracy, which is done in a single step, precision maximization might require a number of function evaluations (STD checks). Each new check requires a new parameterization, which can be provided by any optimizer compatible with continuous n-dimensional inputs and a single scalar objective in a non-differentiable, stochastic setting, with the goal of achieving an STD score below the threshold value in as few STD checks as possible. More formally, each iteration at time step t has past observations of STDs $r_{t'<t} \in \mathbb{R}$ and corresponding parameterizations $a_{t'<t} \in \mathbb{A} \subseteq \mathbb{R}^n$, and now the optimizer has to suggest the next a_t to reach the lowest possible r_t until it reaches a threshold δ. The number of time steps is unconstrained but should be minimal:

Parameter	Range
Dispense flow rate	$5.0\mu L/s$ - $40.0\mu L/s$
Blow out air volume	$3.0\mu L$ - $10.0\mu L$
Dispense mix speed	$5.0\mu L/s$ - $19.0\mu L/s$
Dispense swap speed	$1.0\mu L/s$ - $8.0\mu L/s$
Dispense settling time	$0s$ - $8s$

Table 3: The parameters of the liquid transfer that are optimized in the listed experiments, unless mentioned otherwise. The set of parameters that are optimized and their corresponding ranges are configurable.

$$\min_t r_t \leq \delta \quad \text{where } r_t = f(a_t) \tag{1}$$

where $f : \mathbb{A} \to \mathbb{R}$ is the unknown stochastic function mapping the transfer parameters to the standard deviation of measurements when multiple transfers are done at this parameterization.

Several of such optimizers were evaluated here: hierarchical heuristic optimization [8], vanilla policy gradients on a continuous action space [9], and Bayesian optimization[10, 11, 12]. The parameters that are optimized are given in Table 3. Below we describe the three STD minimizers in more detail. In each case, the *reward* or objective function was either the STD of N transfers ('STD check') directly, or its negative, depending on whether the algorithm maximizes or minimizes (VPG and HOO maximize).

Discretization is a common method of handling continuous parameters in reinforcement learning (e.g. [20]). However, it is impractical to apply this technique here and it was not evaluated for a couple of reasons. First, discretization necessarily leads to a loss of precision, ignoring values in between bins. This is particularly important when slight differences lead to large impacts, as might be the case here. Second, large but unknown parts of the parameter space are hypothesized to have insufficient performance and it is wasteful to 'assign' bins to these parts.

4.2.1 Hierarchical heuristic optimization

Hierarchical heuristic optimization (HOO) is a binary tree based optimization algorithm based on uncertainty estimates. At its core, HOO maintains a directed binary tree where each node corresponds to the subspace spanned by its descendants. Iteratively, 1) a leaf node is selected for evaluation, based on an optimistic value estimate, 2) the point in the center of the optimization space corresponding to the selected node is evaluated, 3) the node is expanded by adding two child nodes, and 4) the statistics in the tree are updated. HOO's adaptive exploration of the landscape can be construed as a variable-resolution discrete encoding of a hyperdimensional continuous space[8].

More formally, a general parameter n is used to track the number of evaluations. Each node maintains a visit count T, its depth in the tree h, the index in that depth $1 \leq i \leq 2^h$, its cumulative reward so far `cum_reward`, the mean reward $\hat{\mu} = \frac{\texttt{cum\_reward}}{T}$, as well as an estimate of the maximum mean-payoff in the subspace it associated with:

$$U = \begin{cases} \hat{\mu} + \sqrt{\frac{2 \ln n}{T}} + \nu_1 \rho^h, & \text{if } T > 0; \\ +\infty, & \text{otherwise.} \end{cases} \tag{2}$$

The actual upper bound is recursively constrained by the maximum mean-payoff of a node's children, if defined:

$$B = \begin{cases} \min \{U, \max \{B_{\text{left}}, B_{\text{right}}\}\}, & \text{if } B_{\text{left}}, B_{\text{right}} \text{ defined;} \\ +\infty, & \text{otherwise.} \end{cases} \tag{3}$$

ν_1 and ρ are hyperparameters to control the maximum variation in the region of a node; both are set to 0 in these experiments. Further, in these experiments, the term $\ln n$ is the numerator was replaced by $n_e = 10$, which the authors of the original work find to achieve the same regret as the original equation when the time horizon is known, while the reducing the computational complexity of the update step from $O(n^2)$ to $O(\log n)$. Even though the time horizon is not technically known here, it is empirically extremely low compared to the original context, and setting it to a fixed value works well.

In the selection step, the tree is traversed from the root down by repeatedly selecting the child with the highest B value until a leaf is reached. Ties are broken randomly. In the update step, T, `cum_reward`, and all quantities thereof, are updated, as well as the global parameter n.

Using a binary tree to discretize a single dimension is trivial. By defining a $\{0, 1\}^h \to \mathbb{R}^n$ mapping for each depth level h, going from a node in the binary tree to a point in the search space, HOO can be used on n-dimensional hypercubes. Note that $\mathbb{R}^n$ here refers to the user-defined optimization space and is independent of number of evaluations, also labeled n. In this thesis, the mapping at depth h is defined by cutting the remaining subspace in half along its longest dimension h times (Fig. 5), though any deterministic discretization of action space $A \subseteq \mathbb{R}^n$ into h subspaces (partitions) may be used.

4.2.2 Vanilla policy gradients

Many reinforcement learning algorithms learn values for discrete actions, something that is not feasible or possible with large or continuous action spaces[9]. One way of applying reinforcement learning in continuous action spaces, suggested by Barto and Sutton, is by using policy gradient methods to learn the statistics of a probability distribution over the action space instead of values for

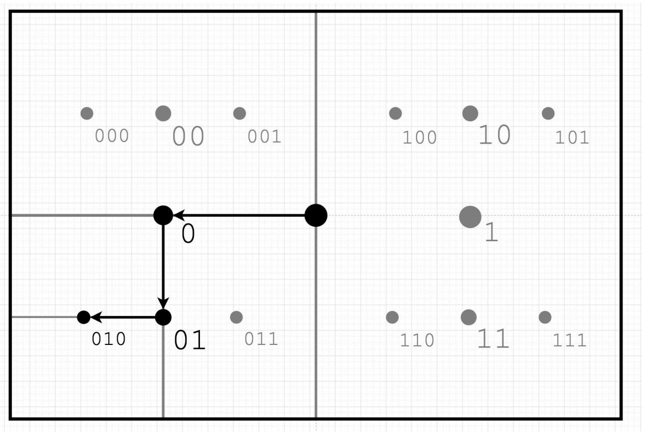


Figure 5: Hierarchical heuristic optimization (HOO) divides the remaining space of a hypercube in half along the longest remaining dimension. Each possible splitting point at depth h has a unique identifier, forming a $\{0,1\}^h \rightarrow \mathbb{R}^n$ bijective mapping connecting nodes in the binary tree to the n-dimensional continuous search space. While higher h affords higher resolution in the entire space, selection in HOO optimistically prioritizes regions of high expected value and explores only certain regions.

actions directly[9]. To get an action, one simply samples from the parameterized distribution.

For a Gaussian distribution, for example, the probability density function is defined as follows:

$$p(x) \doteq \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{4}$$

The mean μ and standard deviation σ can be parameterized functions instead of constants:

$$\pi(a|\theta) \doteq \frac{1}{\sigma(\boldsymbol{\theta})\sqrt{2\pi}} \exp\left(-\frac{(a-\mu(\boldsymbol{\theta}))^2}{2\sigma(\boldsymbol{\theta})^2}\right) \tag{5}$$

Continuing Barto and Sutton's example, the parameterized functions could be as below. Note that parameters are stateless constants here, but could easily be functions of some state, even neural networks. The exponent of $\boldsymbol{\theta}_\sigma$ is taken to ensure $\sigma(\boldsymbol{\theta})$ is positive.

$$\mu(\boldsymbol{\theta}) \doteq \boldsymbol{\theta}_\mu \quad \sigma(\boldsymbol{\theta}) \doteq \exp\left(\boldsymbol{\theta}_\sigma\right) \tag{6}$$

In order to use this parameterization with policy gradient methods, eligibility vectors have to be formulated. These vectors, based on the $\nabla \ln x = \frac{\nabla x}{x}$ identity, give the 'direction in the parameter space that most increases the probability of repeating the action in the future'[9].

$$\nabla \ln \pi(a, \boldsymbol{\theta}_\mu) = \frac{\nabla \pi(a, \boldsymbol{\theta}_\mu)}{\pi(a, \boldsymbol{\theta})} = \frac{1}{\sigma(\boldsymbol{\theta})^2}(a - \mu(\boldsymbol{\theta})), \quad \text{and} \tag{7}$$

$$\nabla \ln \pi(a, \boldsymbol{\theta}_\sigma) = \frac{\nabla \pi(a, \boldsymbol{\theta}_\sigma)}{\pi(a, \boldsymbol{\theta})} = \left(\frac{(a - \mu(\boldsymbol{\theta}))^2}{\sigma(\boldsymbol{\theta})^2} - 1 \right). \tag{8}$$

From here, any policy optimization algorithm can be used. We used 'vanilla policy gradients' (VPG), for 1-timestep this is the same as REINFORCE, which simply moves the parameters in the direction of the eligibility vectors with step-sizes proportional to the return $G_t = \sum_t r_t = r_0$.

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t, \theta_t)}{\pi(A_t, \theta_t)}. \tag{9}$$

As seen in equations 7 and 8, the denominator of the eligibility vector is the current action. This causes large updates to small parameters, which in simulation lead to unstable convergence (data not shown). Normalizing all dimensions worked better in simulation, and was used when applying this algorithm in reality.

4.2.3 Bayesian optimization

Bayesian optimization is a popular technique for sequential, non-differentiable ("black-box") optimization problems. A probability distribution is defined over all possible functions that match the observed data points using the 'surrogate model' (often a Gaussian process), approximating the objective function. This surrogate model is continuously updated as the function is evaluated. Points are sampled as suggested by the 'acquisition function', a function of both the values (exploitation) and corresponding uncertainties (exploration) the surrogate model. Each time a new data point is observed, the prior distribution (the initial belief about the function) is updated to form the posterior distribution (the updated belief about the function), which is again represented as a Gaussian process, following Bayes' rule $P(f|D) = \frac{P(D|f)P(f)}{P(D)}$ where f is a possible fitting function and D is the data [10, 11, 12].

A Gaussian process is fundamentally a stochastic process where each random variable, or any linear combination thereof, is normally distributed. Different from multivariate Gaussian distributions, a Gaussian process (GP) models not over finite vectors, but over entire functions. Some view this as an infinite-dimensional extension of multivariate Gaussian distributions. This is important, because we want to get estimates over the entire continuous output space (in this case, standard deviation of transfers as a function of an arbitrary input space), rather than estimates of certain points (as you would get with a finite, multivariate Gaussian distribution). Looking at how this information is represented mathematically, we find that the other key difference to simple multivariate distributions is that the mean vector μ is replaced by a mean function $m(x)$ of the input values, and that the covariance matrix Σ is replaced by a kernel function $k(x_1, x_2)$ where x_1, x_2 are arbitrary parameters between which the covariance needs to be known[21].

5 Results

In this section, we demonstrate *lil optimizer*'s ability to optimize 500nL transfers, an order of magnitude lower than previously recommended. We also investigate the effects of parameter N for each STD minimizer (HOO, VPG and BO).

5.1 Optimization of 500nL transfers

Lil optimizer achieved 500nL transfers to an accuracy and precision of 50nL with both HOO and Bayesian optimization (Fig. 6, Table 4). This precision is near the minimum achievable with the evaluation setup used (the analytical scale). Policy gradient methods were not tested on 500uL transfers due to time constraints.

For both algorithms, the experiment was repeated twice. In all cases except HOO repeat 2, the first iteration saw a slight loss of liquid in the first iteration, due to liquid sticking to the side of the tip. To compensate, the following iterations aspirated the difference, resulting in considerably higher requested machine volumes v_m of 0.922, 0.838, 1.046, and 0.903 μL.

It is noteworthy that both algorithms sometimes converge faster (2nd repeat) than other times (1st repeat). This difference is ascribed to randomness inherent to both optimization algorithms, or potentially stochasticity in the physical environment.

	Precision (STD in mg)	Accuracy (Mean - Target in mg)
HOO (1)	-0.0337	0.0954,
HOO (2)	0.0462	0.0524,
Bayesian (1)	0.013	0.0628,
Bayesian (2)	-0.06	0.0998

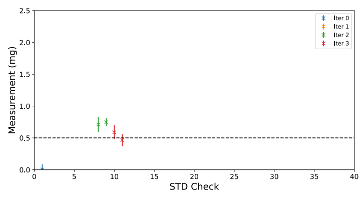
Table 4: Results after optimizing a 500nL transfer with HOO and BO. Two technical repeats.

5.2 The optimal number of transfers per STD check

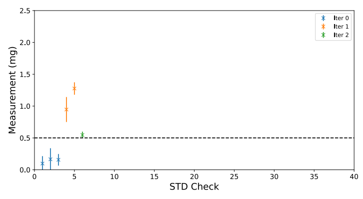
Intuitively, the number of transfers done per STD check (N) yields higher quality data at the cost of a longer running time per evaluation. In this section, we study the effects of changing this hyperparameter for all algorithms and we provide a heuristic for efficiently finding an optimal value.

5.2.1 Hierarchical heuristic optimization

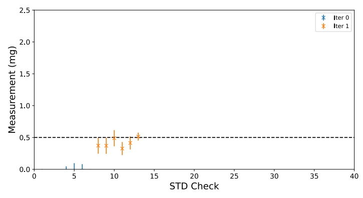
For all $N \in \{3, 5, 7, 9, 12, 15\}$, *lil' optimizer* with HOO reached the desired accuracy and precision thresholds of 50nL for 1uL transfer in 2 iterations (Fig. 7). The number of STD checks per iteration was generally small. With $N = 9$,



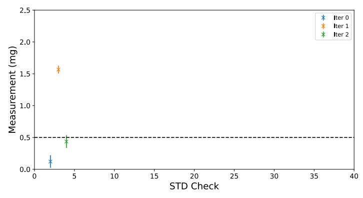
(a) HOO Repeat 1



(b) HOO Repeat 2



(c) Bayesian Repeat 1



(d) Bayesian Repeat 2

Figure 6: Optimization of a 500nL transfer with HOO (a and b) and Bayesian optimization (c and d). Each new color represents a new iteration of the outer loop, on which an accuracy adjustment is made.

the number of STD checks in the first iteration was high compared to other runs (7), which seems to have been an outlier. In the second iteration of this run, only 1 STD checks was required, meaning the STD was still below the threshold value after the accuracy adjustment.

Results of the first STD check during the entire run are generally excellent: the STD is low and the mean is close to the target. This effect is very much dependent on the user-specified range of optimizable parameters: HOO always bisects those in every dimension. Note that with good initial parameters (determined by their range) as shown here, results can be good from the start, whereas with worse parameters, it may take a number of additional STD checks, proportional to the base 2 logarithm of the search space size (HOO bisects).

The stability of the results was verified by performing 16 transfers with the parameters found with each N. It is seen that the higher N, the more precise and accurate the result (Fig. 8). For $N = 3$, the STD during validation is particularly high and it seems that the algorithm reached threshold values just by chance: at such a low number of transfers per STD check, the standard deviation and mean were coincidentally very low, and the algorithm stopped because thresholds were reached. In such cases, obviously, a larger number of transfers using the same parameterization is not expected to have the same mean and STD; the results do not generalize.

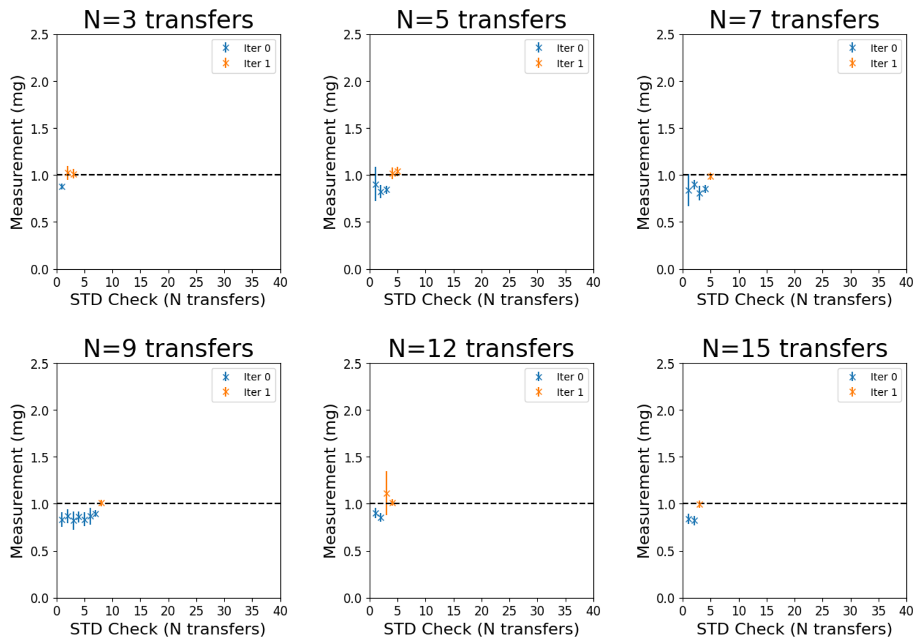


Figure 7: Optimization of 1uL transfers to accuracy and precision thresholds of 50nL with HOO for a different number of transfers N per STD check. Iterations refers to iterations of the outer loop and are color-coded (precision and accuracy adjustments); the inner loop (precision adjustments) may consist of multiple STD checks, depending on when the threshold is reached. In both cases fewer is better. Per STD check, the measured mean is visualized by a cross and one standard deviation by the error bar; the target is visualized with a dashed line.

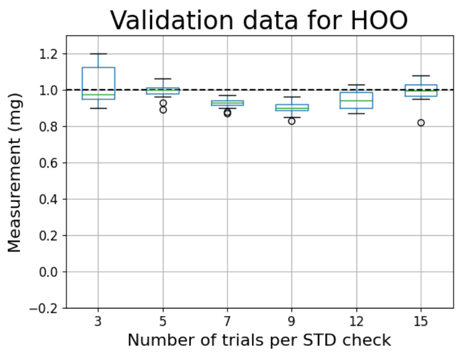


Figure 8: Unbiased estimates of real-world performance after HOO optimization with different N. 16 transfers were performed for each of the transfer parameters obtained from runs in Figure 7.

5.2.2 Vanilla policy gradients

The optimization process with VPG was successful for $N \in \{3, 5, 7, 9, 12\}$ (Fig. 9). Similar to with HOO, the program converges in at most 2 or 3 iterations with few STD checks per iteration, with one interesting exception for $N = 3$. Here, most iterations only require one, or very few, STD checks to reach good precision. However, a great number of iterations is needed which is likely the result of non-representative means causing inaccurate predictions in the linear regression model. Optimization with higher N do not suffer from this problem because the observed means are more representative of the population mean. Validation with VPG has not yet been performed due to time constraints.

The reason for fast convergence and good initial results with VPG is similar to HOO as was described above. Explicitly, VPG likely converges fast because the initial means of the Gaussian cloud are centered in the center of the search space, causing sampled parameters to be close to this center, and as seen with HOO, values close to the center provide good results. This is an optimistic signal because it shows that with some insight and carefully designed search spaces, fast results can be achieved with VPG (and HOO). However, contrary to HOO, it is harder to predict that VPG will swiftly arrive at acceptable parameters given a larger search space because parameters are sampled from a normal distribution; not bisected. This is potentially addressable by tuning the initial standard deviation.

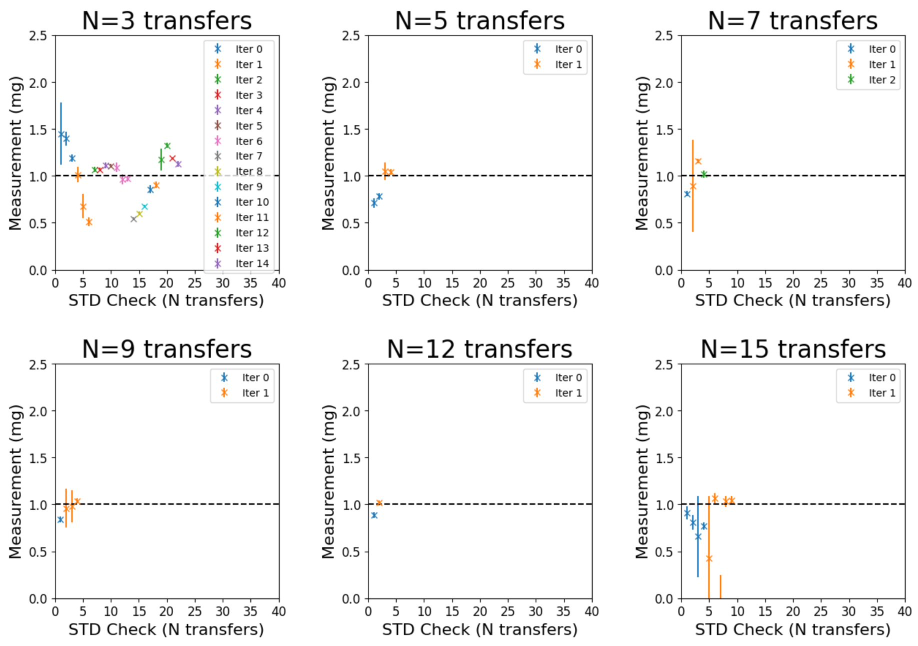


Figure 9: Optimization of 1uL transfers to accuracy and precision thresholds of 50nL with VPG for a different number of transfers N per STD check. Iterations refers to iterations of the outer loop and are color-coded (precision and accuracy adjustments); the inner loop (precision adjustments) may consist of multiple STD checks, depending on when the threshold is reached. In both cases fewer is better. Per STD check, the measured mean is visualized by a cross and one standard deviation by the error bar; the target is visualized with a dashed line.

5.2.3 Bayesian optimization

Compared to HOO and VPG, *lil' optimizer* configured with Bayesian optimization takes many iterations to converge, and for $N \in \{3, 7, 12\}$, it takes (much) more than 2 iterations. $N = 5$, needing few iterations, seems to be an outlier compared to other runs (Fig. 10).

The longer optimization time with Bayesian optimization can be explained by the inherent preference for exploration in, and non-systematic nature of, Bayesian optimization. Out of the three algorithms evaluated here, it is the only algorithm that does not start with parameters close to (VPG) or in the center of (HOO) the search space. This means many points have to be tried before it is established that certain parts of the search space are unproductive.

The fact that many iterations are needed means the accuracy maximizer (by ratio-adjusted or linear-regression based translations) is failing to converge. This is likely explained by the high STD during the earlier trials: measured means are extremely noisy and unrepresentative, making it hard to use those measurements to predict what good machine volumes are.

The validation data looks similar to HOO. It is notable that with $N \in \{3, 5\}$, many transfers fail (0 mg difference between w_1 and w_2) (Fig. 11).

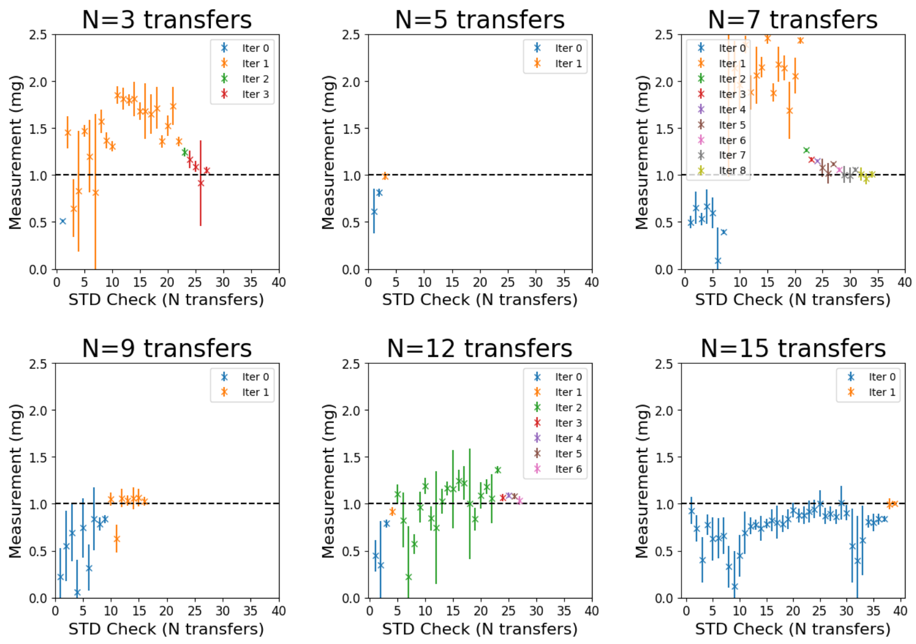


Figure 10: Optimization of 1uL transfers to accuracy and precision thresholds of 50nL with BO for a different number of transfers N per STD check. Iterations refers to iterations of the outer loop and are color-coded (precision and accuracy adjustments); the inner loop (precision adjustments) may consist of multiple STD checks, depending on when the threshold is reached. In both cases fewer is better. Per STD check, the measured mean is visualized by a cross and one standard deviation by the error bar; the target is visualized with a dashed line.

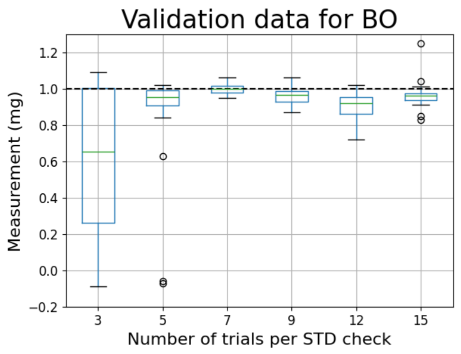


Figure 11: Unbiased estimates of real-world performance after BO optimization with different N. 16 transfers were performed for each of the transfer parameters obtained from runs in Figure 10.

5.2.4 A heuristic for N

Doing full optimizations for many different values of hyperparameter N is expensive in terms of time and number of pipetting tips required. As an alternative, simply doing a small number of transfers $N' > N$, can give insight into the quality of variance estimates for a given N, without having to run full optimization runs for each. With this approach, we might quickly estimate the required number of transfers to get a reasonable STD estimate, which can be used during the actual optimization process.

To investigate this, $N' = 96$ transfers were performed assuming that variance here is approximately equal to variance for $N' = \infty$. This dataset was shuffled 50 times. For all subsets consisting of the first $2 \leq N \leq N'$ items of a random permutation, the STD was computed. This gives insight into how good a sample of N estimates the variance compared to the full N' (Fig. 12).

At $N = 8$, the average observed standard deviation was $61.73nL \pm 29.15nL$ (95% confidence) compared to the 63.94 nL standard deviation over all 96 samples (Fig 12). While higher N continued to lead to more precise results, the rate of progression in terms of range slowed after N=8. Pragmatically, a value of 8 is convenient because it is equal to the number of tips in a standard tip rack (8x12).

Looking at the experiments (Fig. 7, 9, 10), no clear trend is observable in the number of STD checks that are required until the threshold is reached and N. It is notable, however, that with training with VPG (Fig. 9), the run with $N = 3$ takes considerably more iterations than runs with $N > 3$. This can be connected with Figure 12, where the range of values (max-min) rapidly drops until approximately $N = 3$. It seems advisable to use an N at least after this range of rapid improvement. Looking at the validation data (Fig. 8, 11), $N = 3$ is far worse than $N > 3$ in terms of STD achieved. With HOO, there is a trend in validation data getting better, both in terms of accuracy and precision, until $N = 15$, the highest value tested. With BO, some transfers fail after optimization with $N = 5$ (seen by the 0mg measurements), but the validation results are generally good in terms of accuracy and precision with $N > 5$. These results further support the idea that this heuristic measure can provide some insight into an appropriate N, but it is not a conclusive signal.

5.2.5 Comparing the overall runtime for different N

Revisiting the original question of this subsection of how many transfers should be done for a predictive STD check, we looked at the total number of STD checks needed to reach threshold values (Fig. 13) as well as the number of liquid transfers done before convergence (Fig. 14). As hypothesized, the number of STD checks required decreases gradually with the number of transfers per STD check N (VPG), or peaks around $N = 9$ (HOO), presumably due to HOO and VPG being able to use this higher-quality data. The number of STD checks required with BO appears independent of N, suggesting that Bayesian

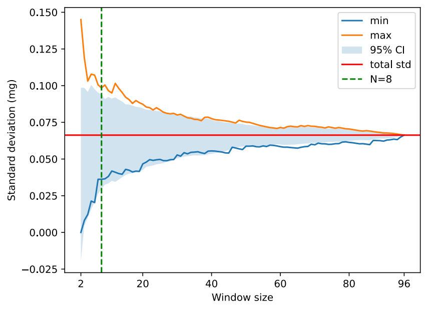


Figure 12: Determination of hyperparameter N at $1\mu L$ transfers. To estimate how many transfers are needed in each evaluation (N) for a good estimate of the standard deviation over an infinite time horizon, a total of 96 $1\mu L$ transfers were performed before optimization. The results were shuffled 50 times, and each time the standard deviation of the first $2 \leq N \leq 96$ items was computed. Plotted here are the minimum and maximum std for each window size N, along with the total standard deviation and an estimated optimal value.

optimization is unable to make use of higher-quality data efficiently.

The total number of transfers, the product of N and the number of STD checks, is a more practical measurement of performance (Fig. 14). Here, BO's inability to use data at higher N efficiently is even more pronounced as can be seen by the huge number of transfers requried compared to the other algorithms. For VPG, the number of transfers required still roughly decreases, although this could also be due to noise (the pattern is not seen consistently). For HOO, the number of transfers required is roughly constant after $N = 9$ (around 50), meaning that high-quality data is effectively used to reduce the number of evaluations required.

An unpredicted but fully explainable result is that with low N, a certain trial can be 'lucky' and have characteristics (mean or standard deviation) under the thresholds. This leads to fast 'convergence'. However, the parameters obtained in this manner lead to transfers that do not generalize well. This was demonstrated by running $N = 16$ transfers given the resulting parameters and observing the mean and standard deviation. Given that this effect does not occur at higher N, this effect is a parameter-specific limitation, easily fixed by setting N at an appropriate value, and not a limitation of *lil optimizer* as a system.

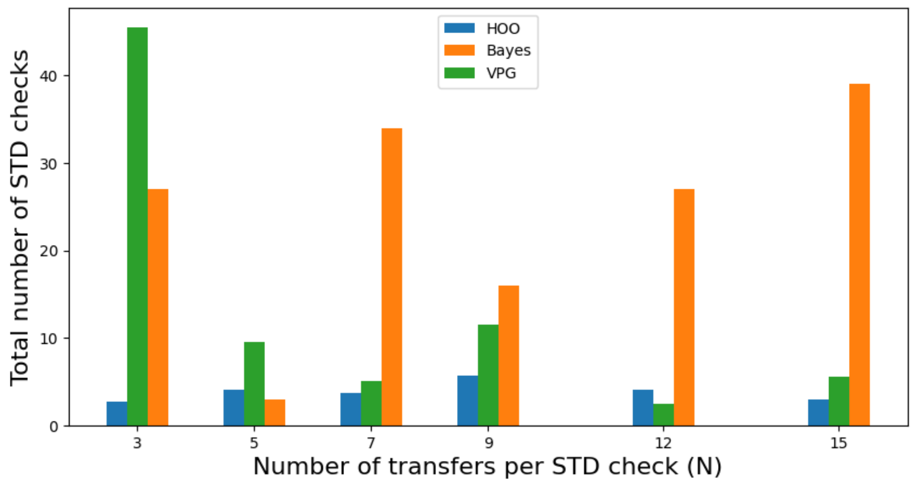


Figure 13: Number of STD checks until convergence; the number of times the inner loop of the double-optimization algorithm ran; the number of data points used by the STD minimizer during the run. Lower is better.

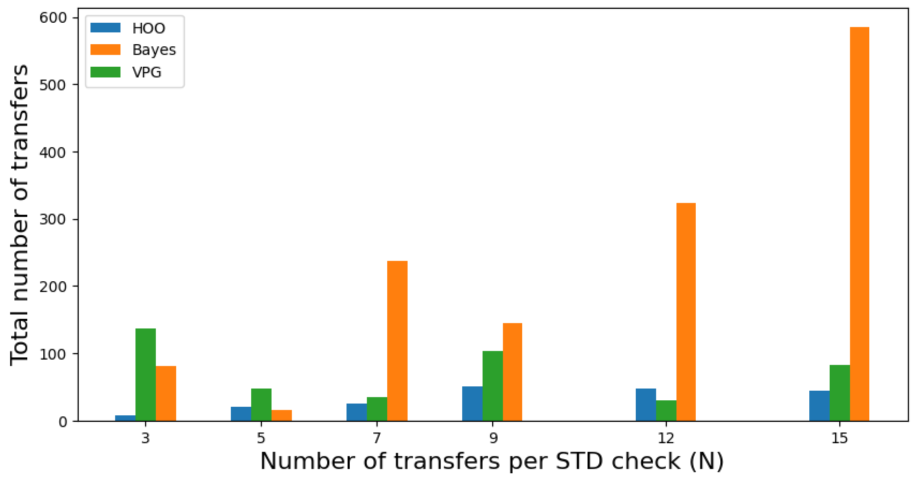


Figure 14: Number of liquid transfers until convergence. Lower is better.

6 Discussion

In this section, we reflect on a select number of technical decisions made in *lil optimizer* and why those may not be optimal. We additionally highlight known limitations of *lil optimizer* while making simultaneous suggestions for future work to address these limitations.

6.1 Evaluation of technical decisions

Single-pass accuracy adjustments. One assumption that was made in the current version of *lil optimizer* is that accuracy adjustments, by linear translation, would shift the mean of measurements into an acceptable range in one pass. This led to the design decision of implementing the accuracy in a single step, rather than a `while not threshold_reached` loop. In any case, a precision check needs to be run to check for the precision after the accuracy adjustment and make any adjustments if necessary (the mean may be significantly higher after an accuracy adjustment, requiring different parameters), but it was not tested whether this precision check and potential correction needs to happen after every accuracy adjustment, as is the case now, or only after an acceptable accuracy was reached (i.e. running accuracy adjustment in a while-loop). The benefit of the current approach is that data fed into the linear regression model for accuracy is all guaranteed to be of acceptable precision, which is not the case when not running intermediate precision-updates. This comes at the risk of optimizing precision for an accuracy (machine volume) that is not actually used. Whether or not such optimization is redundant or useful remains unclear.

Discretization. Discretization was originally thought to provide insufficient resolution. However, the HOO algorithm worked surprisingly well even at few STD checks: the depth of the tree after running so briefly is still low, and so the resolution of HOO's discretization is still very rough. This suggests that perhaps discretization would be a viable strategy in this problem. However, it is not clear how this would impact the users' ability to define their own optimization spaces. The obvious benefit of discretization with HOO is that the resolution is dynamic, and can be arbitrarily high in regions of interest.

Black box optimizers. Only 3 black box optimizers were evaluated here (HOO, VPG, BO) while many more exist in the literature (Section 2.2). While it is possible other optimizers might reach threshold values even faster, it is unlikely to make significant improvements over HOO and VPG, both of which convergence using less than 10 STD checks in most cases (Fig. 13). Other STD minimizers may be preferred for their inherent qualities, however (e.g. as described 'Each optimization is done from scratch' below).

6.2 Limitations and future directions

No non-water liquids were tested. All tests were performed using sterile water. While significant improvements in pipetting precision and accuracies were made compared to previous methods, many experiments additionally use other liquids such as DSMO, protein solutions, DNA solutions, etc. It would be good to evaluate how well *lil' optimizer* optimizes transfers of those liquids.

Sensitivity to noisy mean-volume measurements. The accuracy optimizer uses a ratio-adjusted volume update for the second iteration, and linear regression after that. As is seen in the optimization history with Bayesian optimization (Sec. 5.2.3), a number of noisy measurements can prolong the accuracy maximization considerably. In the future, the linear regression process may weigh recent measurements more, hopefully leading to faster convergence in the outer loop.

Each optimization is done from scratch. While providing the search range for each parameter can be a way to carry over information from one optimization to the next, this is a manual process and such data may not always be available. It would be interesting to explore if a model of the inherent liquid properties can be learned based on past parametersets and observations (a 'latent space' in this context), which could provide more targeted parameter suggestions based on past observations than these from-scratch, problem-independent optimizers discussed here. One particularly promising and simple extension of the current implementation would be to make VPG stateful (context-aware) so that it may use a liquid's density or viscosity in optimizing transfer parameters for that liquid. Alternative contextual bandit optimizers might also be evaluated.

Stateless transfers. The liquid transfers here are 'stateless' in the sense that they are performed with clear tips each time, and only a single aspiration and subsequent dispense are optimized. Many protocols consist of serial dispenses, following a single aspiration, or re-use tips. Both of these settings can be optimized easily by concatenating parameters and optimization the composite-operation as a normal run, or the optimization algorithms (particularly the STD minimizer) can incorporate a tip's past operations to optimize sequential transfers.

7 Conclusion

Lil optimizer enables efficient, closed-loop optimization of automated low-volume liquid transfers, which we hope will accelerate the adoption of automation for low-throughput biology experiments. A precision maximizer ('STD minimizer') pulls all transfers close together along the target axis, gravimetric measurements in these experiments, while a linear translation shifts the mean of the expected distribution into a desirable accuracy range. The precision and accuracy of the current parameter set are evaluated by doing N transfers (a hyperparameter) and observing its mean and standard deviation.

Using iterative precision and accuracy updates, it is possible to reach significantly superhuman performance. The macro-averaged standard deviation of 9 humans, together doing 90 $1uL$ transfers (Section. 3.5), is 0.08 mg. Transfers optimized with HOO and BO, independently validated, reach significant superhuman performance with standard deviations of 0.039 and 0.033 mg respectively ($p < 0.005$; $p < 0.0005$) while having a mean of 0.99125 and 0.998125 (an accuracy error less than $10\mu L$). Optimization, from scratch, can be done in around 30 minutes.

The precision maximizer is an interchangeable component, and any optimizer suitable for continuous, non-differentiable, stochastic, non-seperable problems can be used. Here, three such models were evaluated: hierchical optimistic optimization (HOO), vanilla policy gradients (VPG), and Bayesian optimization. It is observed that HOO and VPG require considerably less trials than Bayesian optimization, presumably because both initially start at values around the center of the optimization space. This leads to successful initial trails, and the algorithm quickly adjusts to find an acceptable parameter set. With Bayesian optimization, on the other hand, the initial trial is effectively random, and measurements can be extremely noisy. This means a high number of trials is required before a suitable parameter set is found.

It is also observed that with low standard deviation in the measurements, accuracy adjustments are often precise, leading to a small number of iterations (HOO and VPG). However, for Bayesian optimization, a comparatively large number of iterations is needed. This could be the result of inaccurate data, where the observed mean of transfers, on which the next iteration is dependent, is not the real mean of those transfers if the sample size had been larger. This hypothesis is based on the large standard deviation observed at those STD checks. The regression model, and particularly ratio-adjusted volume updates, are sensitive to these inaccuracies.

There is a trade-off between high-quality data at high N and faster optimization time of a parameter set at lower N. The overall number of transfers required needs to be minimal to save tips and time. It was hypothesized that higher N will lead to lower total run time, until a certain level because the higher-quality (less noisy) data will allow better predictions by the STD minimizer. This hypothesis might be true for HOO and VPG, but was rejected for BO which appeared unable to use higher-quality data for improving next evaluations and instead just had a greater overall runtime.

With the correct input parameter space, one example is given in Table 3, *lil optimizer* very quickly optimizes transfers to accuracy and precision far beyond the default transfer parameters in PyLabRobot, and even reaches superhuman performance. This makes it a useful tool that enables the use of laboratory automation in low-throughput settings. Further improvements can be made by sharing information between optimization runs of different liquids and optimizing sequential transfers, both of which are relatively simple architectural changes to *lil optimizer*.

Acknowledgements

I would like to thank my advisors Thomas Moerland, Fons Verbeek and Kevin Esvelt for their supervision.

I would like to thank Camillo Moschner of T-Therapeutics and Boqiang Tu of MIT for additional feedback and guidance throughout the project.

I would like to thank Ben Ray of Retro Biosciences for lending crucial hardware used in the experiments.

References

[1] Shaleem I. Jacob, Spyridon Konstantinidis, and Daniel G. Bracewell. "High-Throughput Process Development for the Chromatographic Purification of Viral Antigens". In: *Vaccine Delivery Technology: Methods and Protocols*. Ed. by Blaine A. Pfeifer and Andrew Hill. New York, NY: Springer US, 2021, pp. 119–182. ISBN: 978-1-0716-0795-4. DOI: 10.1007/978-1-0716-0795-4_9. URL: https://doi.org/10.1007/978-1-0716-0795-4_9.

[2] Sean C. Taylor, Beth Hurst, Ian Martiszus, Marvin S. Hausman, Samar Sarwat, Jeffrey M. Schapiro, Sarah Rowell, and Alexander Lituev. "Semi-quantitative, high throughput analysis of SARS-CoV-2 neutralizing antibodies: Measuring the level and duration of immune response antibodies post infection/vaccination". In: *Vaccine* 39.39 (2021), pp. 5688–5698. ISSN: 0264-410X. DOI: https://doi.org/10.1016/j.vaccine.2021.07.098. URL: https://www.sciencedirect.com/science/article/pii/S0264410X21010136.

[3] Emma J Chory, Dana W Gretton, Erika A DeBenedictis, and Kevin M Esvelt. *Enabling high-throughput biology with flexible open-source automation.* 2021. DOI: https://doi.org/10.15252/msb.20209942. eprint: https://www.embopress.org/doi/pdf/10.15252/msb.20209942. URL: https://www.embopress.org/doi/abs/10.15252/msb.20209942.

[4] Erika A. DeBenedictis, Emma J. Chory, Dana W. Gretton, Brian Wang, Stefan Golas, and Kevin M. Esvelt. "Systematic molecular evolution enables robust biomolecule discovery". In: *Nature Methods* 19.1 (2022), pp. 55–64. DOI: 10.1038/s41592-021-01348-4. URL: https://doi.org/10.1038/s41592-021-01348-4.

[5] Jacob T. Rapp, Bennett J. Bremer, and Philip A. Romero. "Self-driving laboratories to autonomously navigate the protein fitness landscape". In: *Nature Chemical Engineering* 1.1 (2024), pp. 97–107. DOI: 10.1038/s44286-023-00002-4. URL: https://doi.org/10.1038/s44286-023-00002-4.

[6] Rick Wierenga, Stefan Golas, Wilson Ho, Connor Coley, and Kevin Esvelt. "PyLabRobot: An Open-Source, Hardware Agnostic Interface for Liquid-Handling Robots and Accessories". In: *Device* (2023). DOI: 10.1016/j.device.2023.100111. URL: https://www.cell.com/device/fulltext/S2666-9986(23)00170-9.

[7] Steven A. Weissman and Neal G. Anderson. "Design of Experiments (DoE) and Process Optimization. A Review of Recent Publications". In: *Organic Process Research & Development* 19.11 (Nov. 2015), pp. 1605–1633. DOI: 10.1021/op500169m. URL: https://doi.org/10.1021/op500169m.

[8] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. "X-Armed Bandits". In: *CoRR* abs/1001.4475 (2010). arXiv: 1001.4475. URL: http://arxiv.org/abs/1001.4475.

[9] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[10] Jonas Mockus. "Bayesian Approach to Global Optimization". In: vol. 37. Jan. 2006, pp. 473–481. ISBN: 978-94-010-6898-7. DOI: 10.1007/BFb0006170.

[11] Jonas Mockus. "On Bayesian Methods for Seeking the Extremum and their Application". In: *IFIP Congress*. 1977. URL: https://api.semanticscholar.org/CorpusID:27071011.

[12] J. Močkus. "On bayesian methods for seeking the extremum". In: *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*. Ed. by G. I. Marchuk. Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 400–404. ISBN: 978-3-540-37497-8.

[13] Melanie Mitchell. *An introduction to genetic algorithms*. Cambridge, Mass.: Mit Press, 1996. ISBN: 9780262280013.

[14] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing". In: *Science* 220.4598 (1983), pp. 671–680. DOI: 10.1126/science.220.4598.671. eprint: https://www.science.org/doi/pdf/10.1126/science.220.4598.671. URL: https://www.science.org/doi/abs/10.1126/science.220.4598.671.

[15] Mohammad Reza Bonyadi and Zbigniew Michalewicz. "Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review". In: *Evolutionary Computation* 25.1 (Mar. 2017), pp. 1–54. ISSN: 1063-6560. DOI: 10.1162/EVCO_r_00180. eprint: https://direct.mit.edu/evco/article-pdf/25/1/1/1535702/evco_r_00180.pdf. URL: https://doi.org/10.1162/EVCO%5C_r%5C_00180.

[16] Nikolaus Hansen and Andreas Ostermeier. "Completely Derandomized Self-Adaptation in Evolution Strategies". In: *Evolutionary Computation* 9.2 (2001), pp. 159–195. DOI: 10.1162/106365601750190398.

[17] Maria E. DiLorenzo, Conal F. Timoney, and Robin A. Felder. "Technological Advancements in Liquid Handling Robotics". In: *JALA: Journal of the Association for Laboratory Automation* 6.2 (2001), pp. 36–40. DOI: 10.1016/S1535-5535-04-00123-6. eprint: https://doi.org/10.1016/S1535-5535-04-00123-6. URL: https://doi.org/10.1016/S1535-5535-04-00123-6.

[18] Rajeev Agrawal. "The Continuum-Armed Bandit Problem". In: *SIAM Journal on Control and Optimization* 33.6 (1995), pp. 1926–1951. DOI: 10.1137/S0363012992237273. eprint: https://doi.org/10.1137/S0363012992237273. URL: https://doi.org/10.1137/S0363012992237273.

[19] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. *Pure Exploration for Multi-Armed Bandit Problems*. 2010. arXiv: 0802.2655 [math.ST]. URL: https://arxiv.org/abs/0802.2655.

[20] Yunhao Tang and Shipra Agrawal. "Discretizing Continuous Action Space for On-Policy Optimization". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (Apr. 2020), pp. 5981–5988. DOI: 10.1609/aaai.v34i04.6059. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6059.

[21] Jochen Görtler, Rebecca Kehlbeck, and Oliver Deussen. “A Visual Exploration of Gaussian Processes”. In: *Distill* (2019). https://distill.pub/2019/visual-exploration-gaussian-processes. DOI: `10.23915/distill.00017`.

A Default pipetting statistics

The tables below show the default transfer accuracy and precision in PyLabRobot. These can be used to determine whether further optimization with *lil optimizer* is necessary in a certain context. Initially, it also provided baselines for *lil optimizer* to beat. All results are using official Hamilton Co-Re II tips.

A.0.1 10 uL Tips

Target	STD (mg)	Mean error (mg)	Precision (%CV)	Trueness (%R)
1.000000	0.007440	-1.106250	-7.002577	-10.625000
2.000000	0.009258	-2.105000	-8.817334	-5.250000
3.000000	0.043239	-3.148750	-29.068482	-4.958333
4.000000	0.267475	-4.380000	-70.388149	-9.500000
5.000000	0.006409	-5.098750	-6.489822	-1.975000
6.000000	0.504506	-5.318750	74.055997	11.354167
7.000000	3.890130	-2.783750	92.265168	60.232143
8.000000	0.871940	-2.547500	15.991555	68.156250
9.000000	0.717391	-1.797500	9.960307	80.027778
10.000000	2.040014	-1.136250	23.015243	88.637500

A.0.2 50 uL Tips

Target	STD (mg)	Mean error (mg)	Precision (%CV)	Trueness (%R)
10.000000	0.025877	-1.728750	0.312860	82.712500
15.000000	0.154220	-2.111250	1.196550	85.925000
20.000000	0.017678	-2.396250	0.100420	88.018750
25.000000	0.121413	-2.676250	0.543873	89.295000
30.000000	0.115380	-2.906250	0.425854	90.312500
35.000000	0.143222	-3.096250	0.448919	91.153571
40.000000	0.168014	-3.330000	0.458177	91.675000
45.000000	0.200958	-3.628750	0.485742	91.936111
50.000000	0.165308	-3.518750	0.355645	92.962500

A.0.3 300 uL Tips

Target	STD (mg)	Mean error (mg)	Precision (%CV)	Trueness (%R)
1.000000	0.038079	-3.617500	-1.454780	-261.750000
2.000000	0.045336	-3.663750	-2.724931	-83.187500
5.000000	0.024495	-3.320000	1.458030	33.600000
10.000000	0.041662	-3.342500	0.625789	66.575000
15.000000	0.060460	-3.056250	0.506202	79.625000
20.000000	0.055404	-2.671250	0.319725	86.643750
25.000000	0.021876	-2.607500	0.097695	89.570000

30.000000	0.066748	-2.496250	0.242689	91.679167
35.000000	0.043239	-2.371250	0.132519	93.225000
40.000000	0.079899	-2.221250	0.211493	94.446875
45.000000	0.123375	-2.112500	0.287672	95.305556
50.000000	0.196905	-1.900000	0.409365	96.200000
100.000000	0.089273	-1.053750	0.090224	98.946250
110.000000	0.084853	-0.950000	0.077811	99.136364
120.000000	0.127139	-0.777500	0.106640	99.352083
130.000000	0.120409	-0.591250	0.093045	99.545192
140.000000	0.061281	-0.491250	0.043926	99.649107
150.000000	0.097943	-0.297500	0.065425	99.801667
160.000000	0.059702	-0.137500	0.037346	99.914062
170.000000	0.081766	0.005000	0.048096	99.997059
180.000000	0.027999	0.171250	0.015540	99.904861
190.000000	0.028754	0.466250	0.015097	99.754605
200.000000	0.034200	0.633750	0.017046	99.683125
210.000000	0.031820	0.728750	0.015100	99.652976
220.000000	0.038707	0.848750	0.017526	99.614205
230.000000	0.060415	0.882500	0.026167	99.616304
240.000000	0.050639	0.982500	0.021013	99.590625
250.000000	0.054756	1.053750	0.021810	99.578500
260.000000	0.069642	1.107500	0.026672	99.574038
270.000000	0.153710	1.436250	0.056628	99.468056
280.000000	0.054100	1.968750	0.019186	99.296875
290.000000	0.054494	2.046250	0.018660	99.294397
300.000000	0.314299	1.628750	0.104201	99.457083

A.0.4 1000 uL Tips

Target	STD (mg)	Mean error (mg)	Precision (%CV)	Trueness (%R)
1.000000	0.047790	-25.696250	-0.193513	-2469.625000
5.000000	0.077724	-25.691250	-0.375639	-413.825000
10.000000	0.065192	-25.047500	-0.433242	-150.475000
15.000000	0.045806	-24.968750	-0.459499	-66.458333
20.000000	0.069269	-24.633750	-1.494883	-23.168750
25.000000	0.086644	-24.502500	17.415833	1.990000
30.000000	0.046904	-24.375000	0.833852	18.750000
35.000000	0.068400	-24.142500	0.629980	31.021429
40.000000	0.056061	-24.030000	0.351041	39.925000
45.000000	0.072494	-23.816250	0.342214	47.075000
50.000000	0.050071	-23.637500	0.189934	52.725000
100.000000	0.062436	-23.218750	0.081316	76.781250
110.000000	0.058797	-23.095000	0.067657	79.004545
120.000000	0.099058	-23.018750	0.102141	80.817708
130.000000	0.029155	-23.027500	0.027254	82.286538

140.000000	0.099391	-22.892500	0.084872	83.648214
150.000000	0.086685	-22.865000	0.068183	84.756667
160.000000	0.078819	-22.841250	0.057466	85.724219
170.000000	0.099857	-22.845000	0.067858	86.561765
180.000000	0.091016	-22.696250	0.057860	87.390972
190.000000	0.118736	-22.728750	0.070984	88.037500
200.000000	0.124377	-22.721250	0.070159	88.639375
210.000000	0.096474	-22.567500	0.051471	89.253571
220.000000	0.098959	-22.687500	0.050153	89.687500
230.000000	0.106024	-22.658750	0.051135	90.148370
240.000000	0.047340	-22.641250	0.021780	90.566146
250.000000	0.034949	-22.557500	0.015366	90.977000
260.000000	0.778496	-22.456250	0.327727	91.362981
270.000000	1.240550	-21.552500	0.499321	92.017593
280.000000	0.159860	-22.831250	0.062162	91.845982
290.000000	0.097358	-22.337500	0.036373	92.297414
300.000000	0.225859	-22.021250	0.081251	92.659583
310.000000	0.080000	-22.800000	0.027855	92.645161
320.000000	0.075593	-22.845000	0.025439	92.860938
330.000000	0.052355	-22.943750	0.017051	93.047348
340.000000	0.087668	-22.965000	0.027653	93.245588
350.000000	0.065014	-22.976250	0.019880	93.435357
360.000000	0.078456	-22.941250	0.023277	93.627431
370.000000	0.041726	-22.996250	0.012025	93.784797
380.000000	0.096649	-23.043750	0.027076	93.935855
390.000000	0.107138	-23.007500	0.029194	94.100641
400.000000	0.159860	-22.721250	0.042372	94.319687
410.000000	0.134104	-22.871250	0.034641	94.421646
420.000000	0.151658	-22.820000	0.038184	94.566667
430.000000	0.177799	-22.858750	0.043670	94.684012
440.000000	0.169953	-22.903750	0.040747	94.794602
450.000000	0.176473	-22.950000	0.041324	94.900000
460.000000	0.095879	-22.907500	0.021936	95.020109
470.000000	0.097358	-22.967500	0.021779	95.113298
480.000000	0.133142	-22.971250	0.029132	95.214323
490.000000	0.205339	-23.127500	0.043982	95.280102
500.000000	0.246805	-23.426250	0.051787	95.314750
510.000000	0.381742	-23.468750	0.078462	95.398284
520.000000	0.303727	-23.472500	0.061170	95.486058
530.000000	0.224436	-23.655000	0.044325	95.536792
540.000000	0.274994	-23.787500	0.053271	95.594907
550.000000	0.276043	-24.105000	0.052490	95.617273
560.000000	0.774651	-23.963750	0.144515	95.720759
570.000000	0.504713	-24.282500	0.092486	95.739912
580.000000	0.642089	-24.202500	0.115526	95.827155
590.000000	0.721218	-24.041250	0.127433	95.925212

600.000000	0.640753	-24.632500	0.111364	95.894583
610.000000	0.641125	-24.021250	0.109411	96.062090
620.000000	0.783371	-24.181250	0.131478	96.099798
630.000000	0.560249	-23.942500	0.092442	96.199603
640.000000	0.428117	-23.556250	0.069449	96.319336
650.000000	0.296621	-23.621250	0.047355	96.365962
660.000000	0.254527	-23.381250	0.039981	96.457386
670.000000	0.353066	-23.536250	0.054615	96.487127
680.000000	0.269228	-23.726250	0.041024	96.510846
690.000000	0.242355	-23.672500	0.036372	96.569203
700.000000	0.094074	-23.762500	0.013911	96.605357
710.000000	0.361781	-23.810000	0.052723	96.646479
720.000000	0.131882	-23.972500	0.018948	96.670486
730.000000	0.196977	-23.950000	0.027898	96.719178
740.000000	0.290563	-23.973750	0.040580	96.760304
750.000000	0.176266	-24.028750	0.024280	96.796167
760.000000	0.195991	-24.251250	0.026638	96.809046
770.000000	0.744339	-25.611250	0.099993	96.673864
780.000000	0.176387	-24.243750	0.023339	96.891827
790.000000	0.138764	-24.123750	0.018118	96.946361
800.000000	0.180693	-24.472500	0.023299	96.940938
810.000000	0.196355	-24.631250	0.025002	96.959105
820.000000	0.169706	-24.755000	0.021340	96.981098
830.000000	0.305509	-24.742500	0.037939	97.018976
840.000000	0.147739	-24.783750	0.018123	97.049554
850.000000	0.191642	-24.791250	0.023223	97.083382
860.000000	0.248592	-24.973750	0.029771	97.096076
870.000000	0.162739	-24.973750	0.019258	97.129454
880.000000	0.376070	-24.785000	0.043974	97.183523
890.000000	0.268804	-24.683750	0.031064	97.226545
900.000000	0.198494	-24.645000	0.022676	97.261667
910.000000	0.366038	-24.538750	0.041339	97.303434
920.000000	0.270383	-24.592500	0.030197	97.326902
930.000000	0.337382	-24.763750	0.037270	97.337231
940.000000	0.186198	-24.638750	0.020341	97.378856
950.000000	0.267258	-24.783750	0.028886	97.391184
960.000000	0.203540	-25.050000	0.021770	97.390625
970.000000	0.211778	-25.087500	0.022412	97.413660
980.000000	0.302581	-25.171250	0.031690	97.431505
990.000000	0.911200	-26.235000	0.094546	97.350000
1000.000000	0.482551	-26.403750	0.049564	97.359625

A.0.5 1000 uL tips with jet dispense

Target	STD (mg)	Mean error (mg)	Precision (%CV)	Trueness (%R)
1.000000	2.595058	-0.311250	376.777952	68.875000
2.000000	1.187109	-2.285000	-416.529591	-14.250000
5.000000	2.282705	1.460000	35.335993	70.800000
10.000000	1.626626	-0.656250	17.408708	93.437500
15.000000	2.571375	-0.386250	17.595587	97.425000
20.000000	1.127081	-0.463750	5.769180	97.681250
25.000000	2.247214	1.670000	8.426000	93.320000
30.000000	3.477735	3.561250	10.362353	88.129167
35.000000	2.280936	0.431250	6.437639	98.767857
40.000000	1.959989	-1.118750	5.040961	97.203125
45.000000	2.553842	0.882500	5.566047	98.038889
50.000000	0.599208	-1.267500	1.229586	97.465000
100.000000	1.872508	-0.165000	1.875603	99.835000
110.000000	2.106758	0.905000	1.899606	99.177273
120.000000	2.418151	0.223750	2.011376	99.813542
130.000000	2.331266	0.290000	1.789290	99.776923
140.000000	1.372318	-0.110000	0.980998	99.921429
150.000000	2.041872	2.168750	1.341847	98.554167
160.000000	2.153534	3.492500	1.317206	97.817188
170.000000	2.884153	0.076250	1.695800	99.955147
180.000000	2.829651	1.801250	1.556453	98.999306
190.000000	0.495680	-0.921250	0.262155	99.515132
200.000000	2.417324	0.320000	1.206731	99.840000
210.000000	1.703219	-0.031250	0.811177	99.985119
220.000000	0.667521	-1.148750	0.305011	99.477841
230.000000	2.423987	0.710000	1.050664	99.691304
240.000000	0.585526	-1.081250	0.245073	99.549479
250.000000	1.975959	0.418750	0.789062	99.832500
260.000000	1.874549	-0.132500	0.721348	99.949038
270.000000	2.824474	1.766250	1.039303	99.345833
280.000000	2.674261	1.931250	0.948551	99.310268
290.000000	0.774697	4.701250	0.262875	98.378879
300.000000	0.649654	4.162500	0.213588	98.612500
310.000000	0.688243	4.127500	0.219097	98.668548
320.000000	0.540628	4.202500	0.166756	98.686719
330.000000	0.615676	4.190000	0.184229	98.730303
340.000000	0.464095	4.158750	0.134849	98.776838
350.000000	0.580085	4.021250	0.163856	98.851071
360.000000	0.527013	3.465000	0.144997	99.037500
370.000000	0.795362	3.305000	0.213059	99.106757
380.000000	0.539045	3.566250	0.140535	99.061513
390.000000	0.876567	2.986250	0.223053	99.234295
400.000000	0.813019	2.725000	0.201879	99.318750
410.000000	0.658071	2.855000	0.159395	99.303659

420.000000	2.352151	0.081250	0.559928	99.980655
430.000000	1.980851	-0.656250	0.461367	99.847384
440.000000	0.541570	-2.088750	0.123671	99.525284
450.000000	2.380216	1.246250	0.527476	99.723056
460.000000	1.745048	-0.882500	0.380087	99.808152
470.000000	2.477049	-0.310000	0.527380	99.934043
480.000000	0.629603	-1.985000	0.131712	99.586458
490.000000	1.467368	-1.263750	0.300237	99.742092
500.000000	1.793160	-0.572500	0.359043	99.885500
510.000000	1.681698	-1.007500	0.330397	99.802451
520.000000	1.442201	1.405000	0.276599	99.729808
530.000000	2.181611	0.250000	0.411431	99.952830
540.000000	1.824365	-1.312500	0.338668	99.756944
550.000000	2.711048	-1.178750	0.493977	99.785682
560.000000	1.818516	-2.240000	0.326039	99.600000
570.000000	1.443641	-2.041250	0.254181	99.641886
580.000000	2.834899	-3.462500	0.491711	99.403017
590.000000	2.053045	-2.672500	0.349557	99.547034
600.000000	2.855806	-3.533750	0.478787	99.411042
610.000000	2.243151	-4.871250	0.370690	99.201434
620.000000	1.722701	-2.528750	0.278993	99.592137
630.000000	1.665565	-5.002500	0.266491	99.205952
640.000000	2.183450	-3.796250	0.343200	99.406836
650.000000	0.667913	-2.397500	0.103136	99.631154
660.000000	0.631099	-2.325000	0.095959	99.647727
670.000000	0.557001	-2.027500	0.083387	99.697388
680.000000	0.589708	-1.451250	0.086907	99.786581
690.000000	0.763245	-1.645000	0.110880	99.761594
700.000000	0.571908	-1.852500	0.081918	99.735357
710.000000	0.646439	-1.926250	0.091295	99.728697
720.000000	0.562391	-1.913750	0.078318	99.734201
730.000000	0.419436	-2.141250	0.057626	99.706678
740.000000	0.551148	-1.857500	0.074667	99.748986
750.000000	0.548921	-1.600000	0.073346	99.786667
760.000000	0.665989	-1.826250	0.087841	99.759704
770.000000	0.589358	-2.195000	0.076759	99.714935
780.000000	0.742058	-2.097500	0.095392	99.731090
790.000000	0.674641	-2.006250	0.085615	99.746044
800.000000	0.678296	-2.120000	0.085012	99.735000
810.000000	0.401711	-1.825000	0.049706	99.774691
820.000000	0.706782	-2.023750	0.086406	99.753201
830.000000	0.240115	-1.616250	0.028986	99.805271
840.000000	0.480994	-1.571250	0.057369	99.812946
850.000000	0.406307	-1.240000	0.047871	99.854118
860.000000	0.597680	-1.277500	0.069601	99.851453
870.000000	0.597040	-1.665000	0.068757	99.808621

880.000000	0.716200	-1.720000	0.081546	99.804545
890.000000	0.933427	-1.505000	0.105057	99.830899
900.000000	2.209699	-1.915000	0.246046	99.787222
910.000000	0.085189	-0.570000	0.009367	99.937363
920.000000	0.668533	-1.192500	0.072761	99.870380
930.000000	0.751987	-1.433750	0.080984	99.845833
940.000000	1.557019	-2.297500	0.166046	99.755585
950.000000	0.549570	-2.206250	0.057984	99.767763
960.000000	2.041239	-2.600000	0.213207	99.729167
970.000000	0.814458	-0.526250	0.084010	99.945747
980.000000	1.971323	-2.160000	0.201600	99.779592
990.000000	2.707700	-3.361250	0.274437	99.660480
1000.000000	0.661901	0.703750	0.066144	99.929625