IOP Publishing Mach. Learn.: Sci. Technol. 5 (2024) 025022 https://doi.org/10.1088/2632-2153/ad3cb6

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
14 December 2023

REVISED
6 March 2024

ACCEPTED FOR PUBLICATION
9 April 2024

PUBLISHED
29 April 2024

PAPER

Review and experimental benchmarking of machine learning algorithms for efficient optimization of cold atom experiments

Oliver Anton[1,5,*], **Victoria A Henderson**[1,5], **Elisa Da Ros**[1], **Ivan Sekulic**[2,3], **Sven Burger**[2,3], **Philipp-Immanuel Schneider**[2,3] **and Markus Krutzik**[1,4]

[1] Institut für Physik and IRIS, Humboldt-Universität zu Berlin, Newtonstr. 15, 12489 Berlin, Germany
[2] JCMwave GmbH, Bolivarallee 22, 14050 Berlin, Germany
[3] Zuse Institute Berlin (ZIB), Takustraße 7, 14195 Berlin, Germany
[4] Ferdinand-Braun-Institut (FBH), Gustav-Kirchhoff-Str.4, 12489 Berlin, Germany
[5] O A and V A H are joint first authors.
[*] Author to whom any correspondence should be addressed.

E-mail: oliver.anton@physik.hu-berlin.de

Keywords: machine learning, cold atoms, optimization, Bayesian optimization, benchmarking, noisy expected improvement

Abstract

The generation of cold atom clouds is a complex process which involves the optimization of noisy data in high dimensional parameter spaces. Optimization can be challenging both in and especially outside of the lab due to lack of time, expertise, or access for lengthy manual optimization. In recent years, it was demonstrated that machine learning offers a solution since it can optimize high dimensional problems quickly, without knowledge of the experiment itself. In this paper we present results showing the benchmarking of nine different optimization techniques and implementations, alongside their ability to optimize a rubidium (Rb) cold atom experiment. The investigations are performed on a 3D ^{87}Rb molasses with 10 and 18 adjustable parameters, respectively, where the atom number obtained by absorption imaging was chosen as the test problem. We further compare the best performing optimizers under different effective noise conditions by reducing the signal-to-noise ratio of the images via adapting the atomic vapor pressure in the 2D+ magneto-optical trap and the detection laser frequency stability.

1. Introduction

Cold atom systems are an essential part of the quantum revolution, facilitating new generations of quantum technologies. Although the theory behind laser and evaporative cooling techniques is well understood [1–4], in practice each apparatus is unique and optimal parameters must be found experimentally, typically in a lengthy manual optimization routine involving a large number of inter-dependent parameters. Such a labour intensive approach requires a large amount of technical expertise and patience, and will often only find a local efficiency maximum due to the size and complexity of the parameter space. Thus, by instead optimizing via machine learning, one can improve the productivity of cold atom experiments and also take steps towards autonomous operation or control by non-specialist operators. This will be particularly essential for mobile quantum technologies such as gravimeters [5–9] and lattice clocks [10–14], where the environmental conditions vary, or space missions on manned and unmanned spacecraft [15–22]. However, the optimization of problems involving noisy data and large parameter sets is computationally heavy and requires the use of an appropriate optimization algorithm.

The topic of optimization has been of significant interest in the community, with a variety of approaches being taken. Earlier examples utilised heuristic optimization algorithms, such as differential evolution algorithms, to maximize atom number in a magnetic trap loaded from a magneto-optical trap (MOT) [23, 24] and demonstrated a higher atom number than that achieved by manual optimization. Here up to 21 parameters were optimized in 5 h 45 min or 5000 evaluations. Later, neural networks were used to optimize optical depth in an MOT by optimizing 63 parameters [25], and to optimize an evaporation ramp for ^{87}Rb Bose–Einstein condensate (BEC) using optical depth (OD) as indicator for the phase space density [26].

Reinforcement learning is used in [27] to optimize atom number and temperature simultaneously, as well as preparing a defined number of atoms in the MOT. The optimisation technique is performed on both a simulation and a real world experiment, using laser detuning as a single optimization parameter. Currently, the most popular machine learning (ML) technique within the field is Bayesian optimization (BO) using Gaussian process regression [28–32]. Such routines were, for example, used to optimize the evaporation ramp for a crossed optical dipole trap (cODT) via a 16 parameter ramp. In this case, the minimization of the number of atoms in the wings of the cloud was used as an optimization goal [28]. They were also able to identify which parameters are important to optimize and which could be excluded from the optimization as unimportant. Another work benchmarks a different Bayesian optimizer against a random search, and shows the relative importance of different stages of an evaporation ramp for atoms in a cODT [29]; here 8 parameters are used to optimize the number of atoms in a circular area of interest, with 300 trials requiring 3 h, and repeating the optimization 16 times to reduce the effects of noise. Further benchmarking work has occurred in [30], comparing BO to neural networks and differential evolution by optimizing 37 parameters for cooling and subsequent evaporation to Bose–Einstein condensate (BEC) in a time-averaged orbiting potential trap; this showed the significant advantage of BO over neural networks and differential evolution.

While these results are promising in terms of applying BO for the preparation of ultra-cold atom systems, more systematic comparisons between different methods are required to facilitate the choice of the most suitable algorithms. BO as well as other ML methods can have a significant computational overhead for calculating the next parameter sample due to the required training and evaluation of the underlying ML model. On one hand, this calls for an efficient implementation of ML methods. But on the other hand, a quantification of the overhead versus the runtime of the cold-atom experiment is required to decide between ML methods and less computationally expensive heuristic methods.

Another major challenge that influences the choice of the best optimization method is the inherently noisy data that is produced in changing environments, especially for mobile experiments. Atom numbers and temperatures are typical optimization objectives. They are calculated from the fitting of the atom cloud absorption profiles. This fitting can be very sensitive to fluctuations, especially at low atom numbers. This can degrade the performance of algorithms to varying degrees, depending on the underlying strategy. This difficulty can be tackled in a range of ways. For example, one can take the average of several optimization cycles [29]. An alternative approach is to adjust the algorithm, for example by resampling old population members in a differential evolution algorithm [24], or by using a cost function which is less sensitive to noise [28–30]. Many of the alternative cost functions have the additional benefit of requiring less computation time by using parameters such as OD as an indicator for atom number or phase space density. It has also been recently shown that one can overcome the problem of a changing environment by using reinforcement learning, as performed in [33].

In this paper, we study and benchmark how noise and dimensionality influence the performance of a large set of heuristic and ML optimization methods. We consider the heuristic methods particle swarm optimization (PSO) [34, 35] and its adaptation LILDE [24], differential evolution (DE), covariance matrix adaptation evolution strategy (CMA-ES) [36] and downhill simplex, also known as Nelder–Mead search [37]. As a baseline uninformed method, we include random sampling (RS) from a uniform distribution. In addition, ML-based BO in various implementations [38, 39] is considered, with particular emphasis on algorithms that can handle noise in an explicit manner. Moreover, an extension of BO is studied, which is explicitly designed to cope with noisy data without the need for repeated optimization or resampling [40]. Although the extended BO method is more elaborate, we developed an efficient implementation that facilitates the optimization of a large number of parameters using a fitted atom number as a cost function with a small computational overhead [41].

2. Optimization algorithms for tuning experimental parameters

There is a large number of algorithms that are suitable for minimizing different kinds of objective functions. The laboratory experiments considered in this work can be regarded as a function $f: \boldsymbol{p} \in \mathcal{X} \subset \mathbb{R}^d \mapsto \mathbb{R}$ that maps a d-dimensional vector of experimental control parameters $\boldsymbol{p}$ to the objective value $f(\boldsymbol{p})$. The training data fed to the minimization methods $\tilde{f}(\boldsymbol{p}) = f(\boldsymbol{p}) + \epsilon$ is corrupted by a random noise ϵ that is assumed to have zero mean. In many cases, its probability distribution is well modeled by a normal distribution with a constant variance η^2_{noise}, i.e. $\epsilon \sim \mathcal{N}(0, \eta^2_{\text{noise}})$. The goal is to find the parameter vector $\boldsymbol{p}_{\text{min}}$ that minimizes $\mathrm{E}[\tilde{f}(\boldsymbol{p})] = f(\boldsymbol{p})$ in the search space $\mathcal{X}$. The function $f(\boldsymbol{p})$ is in general non-convex and derivative information is unattainable, rendering gradient-based methods inappropriate. Since d has a moderately large value (here $d = 10$ or $d = 18$) and the objective is expensive to evaluate, an exhaustive search in the full space $\mathcal{X}$ is also infeasible. In such a case, one often resorts to heuristic minimization strategies. These do not guarantee

convergence to a global minimum, however they can explore larger parameter spaces and are to a certain degree robust towards noise.

2.1. Heuristic minimization methods

All considered heuristic strategies start from a random population of N vectors which is updated in each iteration.

PSO is a global minimization strategy that draws its inspiration from the field of sociobiology and the intelligence of animal groups (e.g. a flock of birds) [42]. The swarm members move with individual velocities through the parameter space. These velocities are updated by a randomized weighted sum of the current velocities, velocities directed to the individual best seen parameters $\boldsymbol{p}_{\text{best},i}$, $i = 1,\ldots,N$, and the best seen parameters of the whole swarm $\boldsymbol{p}_{\text{best, swarm}}$.

Differential evolution (DE) is an evolutionary optimization algorithm inspired by the mutation, crossover and selection processes occurring in nature [43]. In the mutation step, for each member $\boldsymbol{p}_i$ of the population, a mutated genome is created as $\boldsymbol{p}_{\text{mut}} = \boldsymbol{a} + F(\boldsymbol{b} - \boldsymbol{c})$, where F is the differential weight and $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ are distinct randomly selected population members. Random entries of $\boldsymbol{p}_i$, selected according to a crossover probability, are replaced with the mutated genome $\boldsymbol{p}_{\text{mut}}$, forming a new candidate. The candidate replaces $\boldsymbol{p}_i$ in the next population if its objective value is lower.

The covariance matrix adaptation evolution strategy (CMA-ES) is another evolutionary algorithm [44]. It draws its N population members from a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ with mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$. In contrast to DE, it does not replace members on an individual basis, but uses a weighted subset of $M < N$ members with the lowest objective values to update the mean $\boldsymbol{\mu}$ and covariance matrix Σ for the next iteration.

Downhill simplex (simplex) or Nelder-Mead search also updates a population of vectors in each iteration [37]. However, this population is a simplex of only $N = d + 1$ vectors (i.e. a triangle in $\mathbb{R}^2$, a tetrahedron in $\mathbb{R}^3$ etc) which is much smaller than typical populations of DE and CMA-ES. New candidates are created through processes called *reflection*, *expansion* and *contraction* of the worst-performing vector of the simplex. If those three candidates do not lead to a specific improvement, the simplex shrinks towards its best performing vector $\boldsymbol{p}_{\text{best, simplex}}$. Due to the small population, simplex is a more local minimization method that typically converges faster than other heuristic methods as in the worst case $d + 2$ evaluations, or in the best case only one evaluation of the objective is required per iteration.

None of the above algorithms handle noise in an explicit manner. However, the population updates are often based on many vectors and corresponding objective evaluations (e.g. N individual comparisons for DE, averaging over M vectors for CMA-ES) such that the impact of noise is diminished. A different approach is taken in the LILDE extension of DE. This algorithm re-evaluates members that have survived a specific number of generations [24], further reducing the impact of noise by removing population members whose fitness value is only good due to noise. We speculate that PSO and simplex can be most severely misled by noise. For these methods a single vector and corresponding noisy evaluation can act as an attractor of the population ($\boldsymbol{p}_{\text{best, swarm}}$ for PSO and $\boldsymbol{p}_{\text{best, simplex}}$ for simplex). As long as no lower value is observed, these vectors are not updated.

2.2. Model-based BO Method

As described in the previous section, heuristic methods use the state of the current population to generate members of the next generation. Information on members of previous populations has no direct effect. Due to this sparse use of information on previous evaluations of $\tilde{f}(\mathbf{p})$, heuristic methods require often many trials and there is a risk they will only converge to local and not necessarily global minima. Model-based optimization seeks to avoid these problems by using all previous evaluations to fit some kind of regression model in each iteration. One of the most common model-based optimization methods is BO [45].

BO is a sequential method that uses previous observations $\mathcal{D} = \{\tilde{f}(\boldsymbol{p}_1),\ldots,\tilde{f}(\boldsymbol{p}_k)\}$ to train a stochastic ML model which is used to determine a new parameter vector $\boldsymbol{p}_{k+1}$ to evaluate. The stochastic model, a Gaussian process, can be regarded as a posterior distribution over random functions given the observations [46]. For each parameter vector $\boldsymbol{p}^*$ it predicts a normal distribution of possible function values $y(\boldsymbol{p}^*) \sim \mathcal{N}\left(\mu(\boldsymbol{p}^*), \sigma^2(\boldsymbol{p}^*)\right)$, where

$$\mu\left(\boldsymbol{p}^*\right) = \mu_0 + \sigma_0^2 \sum_{i,j=1}^{k} \kappa\left(\boldsymbol{p}^*, \boldsymbol{p}_i\right) \Sigma_{ij}^{-1} \left[f\left(\boldsymbol{p}_j\right) - \mu_0\right], \tag{1}$$

$$\sigma^2\left(\boldsymbol{p}^*\right) = \sigma_0^2 - \sigma_0^4 \sum_{i,j=1}^{k} \kappa\left(\boldsymbol{p}^*, \boldsymbol{p}_i\right) \Sigma_{ij}^{-1} \kappa\left(\boldsymbol{p}_j, \boldsymbol{p}^*\right). \tag{2}$$

Here, μ_0 and σ_0^2 are hyperparameters for the mean and variance of the objective function. The covariance matrix $[\Sigma]_{ij} = \sigma_0^2 \kappa(\boldsymbol{p}_i, \boldsymbol{p}_j) + \eta^2 \delta_{ij}$ depends on a positive definite covariance function $\kappa(\cdot,\cdot)$ and on an additional hyperparameter for the noise variance η^2. The best choice of the covariance function depends on the assumed differentiability of the objective $f(\boldsymbol{p})$ [46]. In this work, the considered BO optimizers use the Matérn-5/2 covariance function,

$$\kappa(\boldsymbol{p}_1, \boldsymbol{p}_2) = \left(1 + \sqrt{5}d + \frac{5}{3}d^2\right) \exp\left(-\sqrt{5}d\right) \tag{3}$$

$$\text{with } d = \sqrt{\sum_{i=1}^{d} \frac{(p_{1,i} - p_{2,i})^2}{l_i^2}}\,. \tag{4}$$

The length scales $l_1, \ldots, l_d$, at which the covariance decreases, are another set of hyperparameters. The value of all hyperparameters is determined by their maximum likelihood estimate [46].

The next sampling point $\boldsymbol{p}_{k+1}$ is chosen by some infill criterion. A common choice is to maximise the expected improvement of the predicted objective value distribution

$$\mathrm{EI}(\boldsymbol{p}^*) = \mathrm{E}\left[\max\left(0, y_{\min} - y(\boldsymbol{p}^*)\right)\right] \tag{5}$$

with respect to the lowest seen objective value $y_{\min}$.

In the noiseless case ($\eta^2 = 0$), the predicted variance $\sigma^2(\boldsymbol{p}^*)$, and thus $\mathrm{EI}(\boldsymbol{p}^*)$, tends to zero if the number of neighbouring observations increases. Therefore, a local minimum will always be eventually escaped, since $\mathrm{EI}(\boldsymbol{p}^*)$ will be larger in parameter regions with less data and more predicted variance. In fact, it has been shown that the expected improvement strategy converges in the noiseless case at a near optimal rate to the *global* minimum if the objective belongs to the reproducing kernel Hilbert space with kernel $\sigma_0^2 \kappa(\boldsymbol{p}_1, \boldsymbol{p}_2)$ [47].

In the noisy case, $\sigma^2(\boldsymbol{p}^*)$ and thus $\mathrm{EI}(\boldsymbol{p}^*)$ do not tend to zero, even if $\boldsymbol{p}^*$ is an observed point. This can lead to an oversampling of local minima. Moreover, the best observed value $y_{\min}$ might be corrupted by noise. Several extensions of BO have been proposed for noisy settings [40, 48]. In this work, we focus on the noisy expected improvement strategy (NEI) [40]. For this, F sets of random function values $\mathcal{F}_i = \{y_{i,1}, \ldots, y_{i,k}\}$, $i = 1, \ldots, F$ are drawn from the noisy Gaussian process posterior at the observed points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k$. The fantasies $\mathcal{F}_i$ are thus possible values of $f(\boldsymbol{p})$ that are compatible with the noisy observations $\mathcal{D}$. Each $\mathcal{F}_i$ is used to train a noiseless Gaussian process and to determine a noiseless expected improvement $\mathrm{EI}_i(\boldsymbol{p}^*)$ with respect to the corresponding minimum $y_{\min,\,\mathrm{i}} = \min(\mathcal{F}_i)$. The NEI is then defined as the average over the fantasies

$$\mathrm{NEI}(\boldsymbol{p}^*) = \frac{1}{F} \sum_{i=1}^{F} \mathrm{EI}_i(\boldsymbol{p}^*)\,. \tag{6}$$

2.3. Different implementations of BO

BO is a relatively complex method for which implementation details can have a large influence on the convergence and computational overhead. Therefore, in this work three different implementations of BO are considered, the open-source package FMFN [39], the open-source AX framework [38] provided by Meta Platforms Inc. and the BO optimizer (JCM) included in the commercial software JCMsuite, developed by some of the authors.

One important implementation difference between the three methods is the strategy used to maximise acquisition functions. Both, $\mathrm{EI}(\boldsymbol{p}^*)$ and $\mathrm{NEI}(\boldsymbol{p}^*)$ can be very hard to maximise since at each sample position the acquisition value is zero. This can lead on one hand to a large number of local maxima between the samples, and on the other hand to extremely small acquisition values for large parts of the parameter space. Therefore, the chosen maximization strategy can greatly impact the performance of the BO algorithm.

The open-source package FMFN [39] uses the best result from 10 000 random samples of the acquisition function and 10 local L-BFGS-B optimizations started from random initial positions. Random sampling can result in samples of low quality for larger dimensionality d of the parameter space. For example, for $d = 18$ parameters, 10 000 samples correspond to effectively about 1.7 samples per dimension which can be considered extremely sparse. The AX framework first evaluates the acquisition function at, by default, 1000 random positions. The best 20 positions are further optimized by the local L-BFGS-B or SLSQP method. We speculate, that 1000 initial random samples can be again too sparse to find good initial positions for a local optimization. The JCM optimiser performs an initial DE maximisation of the acquisition function. We believe that this global heuristic maximisation approach is more appropriate given the increasing number of local maxima of the acquisition function. The obtained maximum and the previous sample with the lowest objective value are then tuned by a local L-BFGS-B optimization.

Another important difference in implementation concerns the handling of noise. AX and JCM can detect the noise level η^2 in order to perform a maximisation of NEI($\boldsymbol{p}^*$). FMFN assumes by default that the objective function is noiseless. The same holds for the JCM optimizer with disabled noise detection.

2.4. Efficient implementation of noisy BO

BO is often applied to optimize very expensive problems such as tuning the hyperparameters of other complex ML algorithms [49]. For these applications, the computational overhead of training and evaluating Gaussian processes is often negligible in comparison with testing a hyperparameter set by, e.g. training a deep neural network for several minutes or even hours. In contrast, the few-seconds runtime of a cold atom experiment is much shorter. Within our collaboration [41], we have therefore developed different strategies with the goal of limiting the computational overhead of BO based on noisy experimental data. For this work, the methods were implemented into the JCM optimizer.

Principally, the NEI approach increases the computational effort of BO by the factor F representing the number of noiseless Gaussian processes. In order to decrease this overhead, we do not optimize the length scale hyperparameters of each noiseless Gaussian process, but instead use the length scale hyperparameters of the noisy Gaussian process as proposed in [40]. Moreover, due to identical length scales, the noiseless covariance matrix $[\Sigma]_{ij}^{\text{noiseless}} = \sigma_0^2 \kappa(\boldsymbol{p}_i, \boldsymbol{p}_i)$ is identical for each of the F Gaussian processes up to different optimal prefactors σ_0^2. Therefore, the expensive step of the matrix inversion (i.e. Cholesky decomposition) only has to be done once.

To further decrease the waiting time for the next sample, the numerical effort of the sample computation (e.g. the maximum number of DE iterations) is adapted such that the computational overhead is not larger than the runtime of the experiment itself [50]. Finally, we follow the batch optimization strategy in [40] to compute the next sampling point already during a pending evaluation of the objective. Taking all approaches together, it is often possible to provide good new sample values with negligible waiting time between evaluations of the experiment.

3. Experimental set-up

The apparatus used within this paper is described in figure 1. It is based on a commercial RuBECi vacuum system, alongside a 2D+ MOT platform, magnetic coils [51], and an optical system for the 3D MOT beams. A rendering can be seen in the box labelled 'physics package' in figure 1. The vacuum chamber consists of two glass cells separated by a differential pumping stage. One chamber is used to create a 2D+ MOT which acts as an atom source. The second chamber acts as a 'physics package', where atoms are trapped in a 3D-MOT before further cooling. Here, two separately controllable coils, aligned along the x-axis, provide a magnetic gradient. Additionally, two pairs of Helmholtz coils, aligned along the y- and z-axis of the experiment, are used to generate offset fields for cancelling stray magnetic fields. The laser light for trapping in the 3D-MOT is fed into the system via three collimators providing beams with a diameter of 13.2 mm which are retro-reflected. The atoms are imaged using absorption imaging.

As shown in the 'optical setup' box of figure 1, cooling ($F = 2 \rightarrow F' = 3$) and repumping ($F = 1 \rightarrow F' = 2$) light is generated via two micro-integrated DFB-MOPA modules [52]. The lasers are frequency stabilized via offset locks to a reference laser locked to the ^{85}Rb $F = 3 \rightarrow F' = 3/4$ crossover. The optical power delivered to the experiment is modulated and switched via acousto-optic modulators (AOM), then combined on a 50:50 fibre splitter, before additional extinction is provided by shutters. No active intensity stabilization is present in the setup.

In short, the experimental sequence starts with the loading of the 3D-MOT by a 2D+ MOT, followed by the compression and molasses phases, and ends with the imaging of the atoms. A selection of experimental parameters from this sequence will be used in order to perform the benchmarking.

During the MOT phase, the detuning and optical power of the cooling and repumping lasers can be varied, as well as the currents delivered to both gradient coils and the z-offset coils.

Within the molasses phase, the power and detuning of cooling and repumping lasers can be ramped over a variable duration or switched, and the coil currents can also be modified as in the MOT phase. These tunable molasses parameters are the start point of the ramp for the optical power of the repumping, the endpoints of the ramps for the optical power of cooling and repumping, the current applied to all coils responsible for generating offset fields, and the overall length of the molasses phase. Since the molasses phase is short compared to the rest of the sequence, this parameter does not significantly influence the run-time and thus the result of the comparison. This sequence is used for all measurements presented in the following benchmarking measurements. A list of the parameters including their limits can be found in table A1 in appendix A.

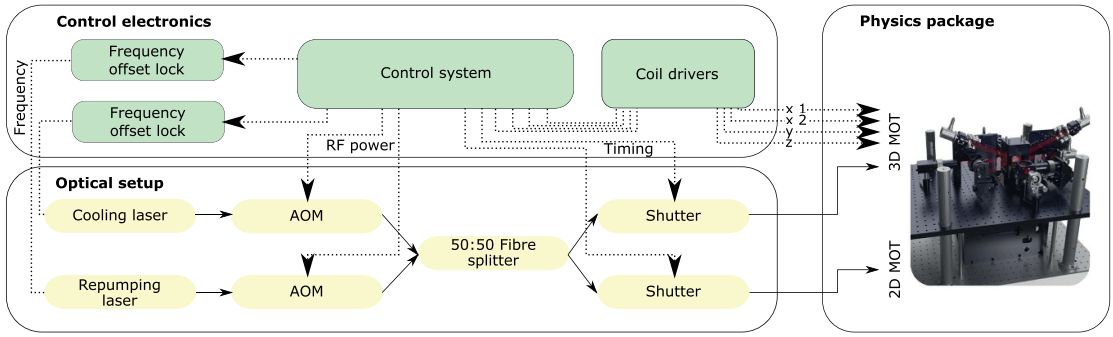

Figure 1. A block diagram showing the relevant parts of the experimental setup used within this paper, alongside a rendering of the apparatus (on the right). Dashed lines indicate electrical and solid lines optical signals, similarly green shading represents electronics components and yellow shading optical components.

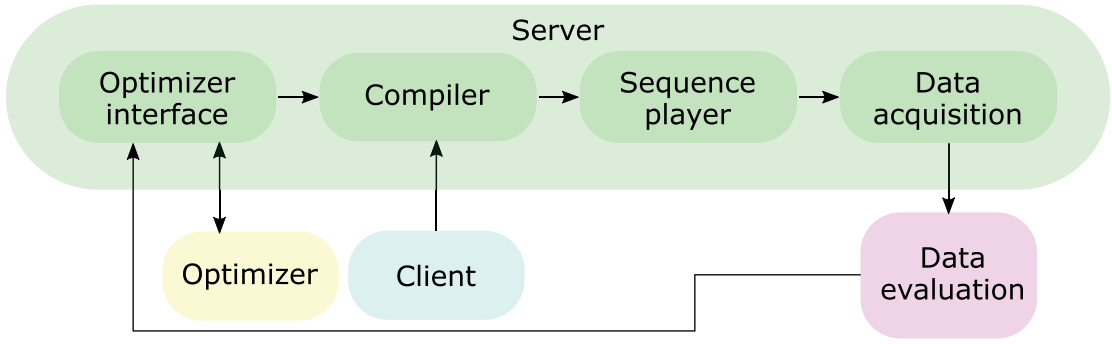

Figure 2. Schematic (block diagram) of the control software showing the server's workflow running the experiment with its sub-processes. Using the client, a sequence is created to be used by the server for compilation and later used by the sequence player. Feedback for the optimizer is provided using the acquired data via the data evaluation software. An adapted sequence is then fed into the compiler.

After a sequence is performed by the experiment, absorption images are evaluated and post-processed in tailored Python analysis code [53]. Atom number and optical density can be evaluated via either a full 2D Gaussian fit, or via two 1D fits of row and column sums. This information is then fed into the optimization algorithm which uses the result to generate a new sequence. This is shown in figure 2.

The offset frequencies, optical powers, coil currents, and timings can be controlled via the Sinara system, a schematic of which is shown in the 'Control electronics' box of figure 1. The Sinara system is a field programmable gate array (FPGA) based real-time system, developed by Quartiq [54]. The FPGA board is augmented with digital in/out breakout boards as well as DDS and DAC cards for controlling the experiment. By using these cards we control our frequency offset locks, coil drivers, shutters and AOMs. The control software of the experiment is a server-client based system using Python. It is described in appendix D.

4. Results

In the following sections we present benchmarking results of the performance of several optimizers using post-molasses atom number after 10 ms of free expansion as the test problem. The problem is chosen due to its resilience to false positives and the potential to adjust the noisiness of the data by worsening the imaging lasers frequency lock and reducing the rubidium gas pressure in the 2D+ MOT. The fitness function for this problem is $f = -N_{\text{atoms}}$, which converts the maximisation of atom number into a minimization problem as required by the optimizers used. We also note that this experiment typically takes approximately 2.5 s per shot.

In the 10-dimensional case we vary parameters that influence the number of trapped atoms, such as laser frequencies, optical powers and magnetic fields. A complete list of the parameters chosen including the limits can be found in the table in appendix A. The MOT loading time was excluded due to the effect it would have on the duration of the experiment and thus the comparability of the results. In addition, we included the start of the molasses phase: the frequencies of the cooling and repumping lasers, as well as the starting point of the power ramp of the cooling laser.

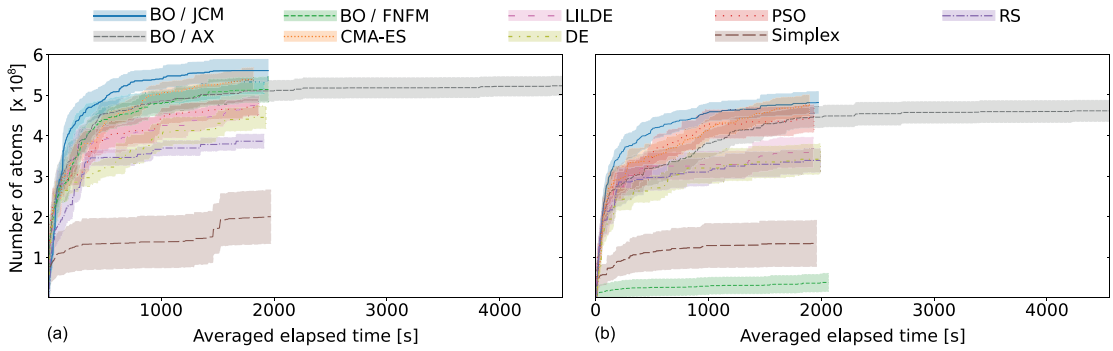

Figure 3. Development of the fitness function (maximization of atom number) over time for different optimizers for (a) a 10-dimensional parameter space and (b) a 18-dimensional parameter space. Each algorithm runs for 400 iterations. The lines represent the mean value, while the shaded areas show the standard deviation over 10 repetitions.

For the 18-dimensional case we extended this list of parameters to include all remaining variables influencing the molasses, namely the endpoint of the cooler power ramp, the start and end point of the repumping power ramp, the offset fields during molasses, and the temporal length of the molasses phase. Since the length of the molasses phase is short, including this as an optimization parameter does not noticeably affect the results of the comparison. The power ramps are linear and frequency is instantaneously switched.

The optimizers were tested using the following procedure. Each algorithm is run using the default settings for 400 iterations and repeated 10 times to obtain an average performance. The choice of 400 iterations allows for many algorithms to reach a plateau whilst also limiting the time needed for data acquisition to a reasonable amount. In order to remove systematic errors due to experimental drift, the algorithms are interleaved rather than run sequentially, that is, each algorithm is run once, one after another, and then this process repeated 10 times. Parameter limits are chosen such as to reduce the size of the parameter space where no atoms are measured.

In the following figures, we show the evolution of the mean fitness (i.e. atom number) over time, with the shaded area representing the standard error of the 10 runs. The optimizers received no initial set and were initialized randomly within the defined parameter space.

As described in section 2, in order to provide comprehensive benchmarking, we compare at least one algorithm per family: CMA-ES; the BO methods JCM with noise detection, FMFN and AX; PSO; variations on differential evolution LILDE and DE; RS; and simplex. We benchmark for both 10- and 18- dimensional space as well as two different noise levels.

4.1. 10-dimensional optimization

Results for 10-dimensional optimization are shown in figure 3. RS is considered a baseline optimization that does not take the information from previous evaluations into account. In our benchmarking, it swiftly finds sets of parameters producing a signal, which indicates that a signal can be found throughout the majority of the parameter space. Although RS is considered a baseline method, we see that simplex consistently performs worse, reaching a maximum of $(2.0 \pm 0.7) \times 10^8$ atoms compared to $(3.9 \pm 0.2) \times 10^8$ atoms for RS, as well as taking longer to reach these values. As described in section 2.1, simplex is a local minimization method and does not incorporate strategies to limit effects of noise. Noisy observations can deteriorate the algorithm by attracting population members to regions with relatively low performance.

Following simplex and RS in performance are DE, LILDE and PSO. All three reach similar fitness values ($(4.5 \pm 0.3) \times 10^8$, $(4.7 \pm 0.2) \times 10^8$ and $(4.7 \pm 0.2) \times 10^8$ atoms respectively) but do not plateau within the given number of iterations. The graphs show jumps in performance at certain times for all three. This is most visible in the LILDE algorithm at 400 s and 1000 s. The jumps are due to these optimizers relying on a generational approach where gained knowledge is collected and combined after a defined number of iterations to generate new sets. The updated process produces a sharp increase in the reached fitness value. This indicates that an adjustment of the default generation size may increase the performance.

Bayesian optimizers FMFN and AX lie in third and fourth place. Both perform surprisingly equally in terms of the reached fitness with $(5.1 \pm 0.3) \times 10^8$ and $(5.2 \pm 0.2) \times 10^8$ atoms, respectively. In terms of optimization speed, they perform similarly to begin with, however, over the full iteration number a clear advantage of FMFN over AX is visible. We attribute the significant computational overhead of AX to the more elaborate computation of expected improvement which requires predictions from F (default $F = 20$)

Gaussian processes instead of only one. Based on the plateauing of fitness it seems that neither optimizer would significantly benefit from a higher number of iterations.

CMA-ES performs second best. Here we see a fitness of $(5.4 \pm 0.3) \times 10^8$ atoms which is close to the best reachable value. Upon reaching 400 iterations, the fitness is still increasing. As a result, it is possible that a higher fitness value could be reached with further iterations and it may perform similarly to the best algorithm given enough time. It is surprising that such a simple optimizer still yields such good performance compared to more elaborate approaches. As such, we conclude that the described approach of updating the mean vector from a set of points is robust against the noise levels in this problem.

The best performing optimizer in our list is the JCM Bayesian optimizer, which was extended for this work. It outperforms the others from start to end. The computation time is short compared to AX due to strategies like precomputing next samples and sharing information between the F Gaussian process regressions (see the theory section 2.3). It reaches the best fitness ($(5.6 \pm 0.3) \times 10^8$ atoms) after ≈1500 s reaching a plateau afterwards.

Analyzing parameters for optimal fitness gives us an indication of the experiment's sensitivity to their variation and their significance within the sequence. In the 10-dimensional case, appendix B displays the final parameters and their standard deviation for the 4 best performing optimizers. Values for other optimizers are omitted due to lower relevance, potentially being far from the absolute maximum.

4.2. 18-dimensional optimization

In the 18-dimensional case we find that all optimizers perform consistently worse compared to the 10-dimensional case. Since all limits stayed the same as those used in the 10-dimensional case, one would expect the optimizers to reach the same, or possibly better, fitness as before. We assume that the larger parameter space makes the optimization problem much harder such that only lower fitness values are found within the same number of iterations. However, we note that we are unable to rule out systematic drifts in the experiment between the time the 10- and 18-dimensional data was taken.

When we look at each optimizer individually, we find that FMFN is now the worst performing optimizer, notably worse even than simplex and RS. We attribute this behaviour to its very sparse search for good samples in higher dimensional problems, as described in section 2.3.

LILDE and DE now barely perform better than RS. By default, in these algorithms, the population size is 15 times the number of dimensions. Therefore, for $d = 18$ dimensions, there are 270 evaluations per iteration meaning that these optimization approaches reduce to effectively random sampling within 400 iterations.

As in the 10-dimensional example, the high overhead of AX results in it taking far longer than any other optimizer to finish 400 iterations. Despite the higher computational effort and thus long optimization times, AX still performs well for higher dimensional problems, reaching a plateau of $(4.6 \pm 0.3) \times 10^8$ atoms.

PSO performs similarly to AX in terms of fitness (reaching $(4.4 \pm 0.3) \times 10^8$ atoms) but takes much less time.

The best performing optimizers are again CMA-ES and the JCM optimizer reaching similar fitness values of $(4.7 \pm 0.3) \times 10^8$ and $(4.8 \pm 0.3) \times 10^8$ atoms, respectively. CMA-ES takes marginally less time for 400 iterations, and does not reach a plateau value, therefore its fitness may eventually exceed the JCM optimizer. Despite this, the JCM optimizer reaches a close-to-optimal fitness value far more quickly than any other optimizer.

In conclusion, one can see the commercial Bayesian optimizer, JCM, performing best, closely followed by CMA-ES representing the best open-source optimizer available for this use case. Other optimizers perform consistently worse and degrade in performance, when compared to the lower dimensional problem. We see this most prominently for FMFN. As with the 10-dimensional case, table appendix C presents the final parameters and their corresponding standard deviations for the top four performing optimizers. Values for other optimizers are once again omitted due to lower relevance, as they may significantly deviate from the global maximum.

4.3. Influence of noise on the experiment optimization

In order to explore the robustness of the optimizers to noise, we tested the algorithms at two different noise levels for the 10-dimensional problem described above. The different noise levels were achieved in two ways: by changing the dispenser current in the 2D+ MOT we can reduce the number of atoms loaded into the 3D MOT; and by changing the PID parameters of the imaging laser we can introduce instabilities in its frequency and thus in the number of atoms measured. The lower atom number reached in the higher noise test case is due to the experimental conditions (i.e. a reduced loading rate) rather than the performance of the optimizers. The two noise levels have a fluctuation in atom number of ±5.6% and ±19.5%, where we note that the lower noise level is the typical performance of our experiment. We compared the highest performing

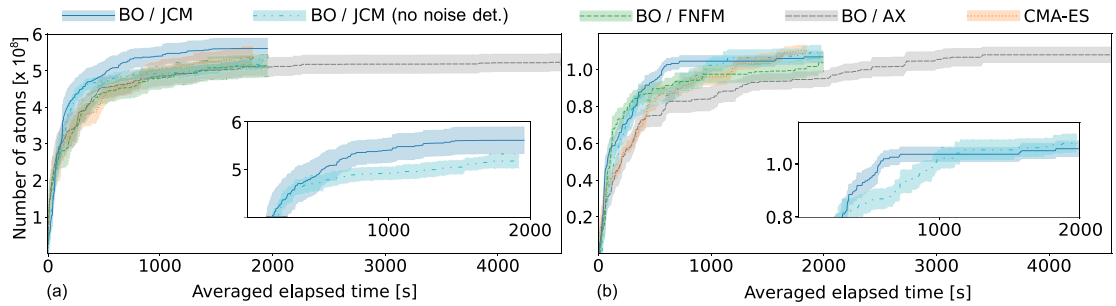

Figure 4. Development of the fitness function (maximization of atom number) over time for the highest performing optimizers with different noise levels. Figure (a) shows optimization with an atom number noise level of ±5.6%, while (b) shows the same for a noise level of ± 19.5%. The JCM optimizer is tested with and without noise detection activated, with this comparison highlighted in the insets. The reduced atom number reached in (b) is due to the experimental conditions choosen to increase the noise level. Each algorithm runs for 400 iterations. The lines represent the mean value, while the shaded area shows the standard deviation over 10 repetitions.

optimizers, namely AX, FMFN, CMA-ES and JCM, alongside the JCM optimizer without its noise detection feature.

In the low noise case (figure 4), the same trends are visible as discussed in section 4.1. However, by comparing the JCM algorithm with and without noise detection we can see that the additional feature does not increase early optimization speeds, instead it results in a slightly higher fitness value of $(5.6 \pm 0.3) \times 10^8$ (with noise detection) compared to $(5.2 \pm 0.2) \times 10^8$ (without noise detection) atoms and in that higher fitness being reached faster.

In the high noise case, all five algorithms tested perform relatively similarly in terms of fitness. FMFN reaches the lowest fitness value ($(1.03 \pm 0.06) \times 10^8$ in 2000 s), followed by AX at $(1.08 \pm 0.04) \times 10^8$ in 4800 s, JCM with and without noise detection reach $(1.07 \pm 0.03) \times 10^8$ and $(1.09 \pm 0.04) \times 10^8$ atoms in 2000 s, and finally CMA-ES reaches a fitness value of $(1.10 \pm 0.03) \times 10^8$ atoms in 1900 s. Although we have ranked the algorithms here according to the mean fitness value, the differences between them are almost all within error margins. The main differences in the algorithms can actually be seen in the speed of optimization. For example, AX is the slowest optimizer at all points in the process, and FMFN slows dramatically after performing fastest in very early stages.

Due to the additional noise detection capabilities of the algorithm developed here, it was expected to perform the best in a high noise environment. The benchmarking data shows that it does indeed optimize faster than all other optimizers and to a higher fitness than all optimizers except CMA-ES. In fact, it reaches close to its final value within 720 s. This is roughly half the time taken for CMA-ES to reach the same value. We assume that CMA-ES performs so well in terms of final fitness due to averaging over many vectors which limits the effect of noise (section 2.1). Nevertheless, the noise detection function provided by JCM significantly improves the speed of optimization compared to the algorithm without it, and enables it to perform better than the other BOs tested. In general we find that most optimizers do not reach a plateau in the available iterations.

In summary, the influence of noise extends the iterations and thus time needed for the optimization. It also generally reduces the initial speed of optimization considerably, which means that more time is required to achieve a 'good enough' value. There is a trade-off in performance between speed and final fitness, with CMA-ES providing a marginally better final fitness, and the JCM with noise detection algorithm reaching a high fitness value much quicker than other algorithms.

5. Outlook

In conclusion, we tested different optimizers and showed their performance for different dimensionalities. Additionally, the developed implementation of JCM's efficient noise detection feature was benchmarked against the three best performing algorithms at two different noise levels. The best performing optimizers were found to be JCM (with noise detection) and the open source CMA-ES, with JCM providing faster optimization.

Optimization of the problem using 18 dimensions rather than 10 resulted in a worse performance for all optimizers. This indicates that optimizing more parameters at once is not always better when it comes to the optimization of real world experiments. We also tested the influence of noise on optimizer performance, finding that with increased noise AX, JCM, FMFN and CMA-ES reach similar fitness, however, JCM and

CMA-ES continued to offer the best performance in terms of speed and fitness. We were able to confirm that the NEI strategy implemented into the JCM optimizer for this work improves the speed of optimization and in some cases also the final fitness value.

When choosing a suitable optimizer for an experiment, one must make a trade-off between speed, final fitness, and cost. The open source algorithm CMA-ES provided a marginally better final fitness, whereas the JCM optimizer with noise detection reached a high fitness value much quicker than other algorithms. Therefore, CMA-ES would be better suited to an experiment which would be optimized once or infrequently. However, for experiments requiring very regular optimization, such as mobile experiments, the speed benefits of the JCM optimizer would enable more efficient operation.

The results presented could be extended further via investigations at longer iteration numbers or in more noisy conditions.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Acknowledgments

This work is supported by the German Space Agency (DLR) with funds provided by the BMWK under Grant Numbers 50WM2067, 50WM2175, 50WM2253 and 50WM2055, as well as the German Federal Ministry of Education and Research (BMBF Forschungscampus MODAL, Project Number 05M20ZBM).

Appendix A. Experimental parameters

Table A1. Optimization parameters and their limits. For the 10 parameter optimization, the parameters up to the line were used, for 18 parameter optimization, all parameters were used. The optimizer values are the values used by our control system to run the experiment. The physical frequencies are quoted as offsets from the ($F = 2 \rightarrow F' = 3$) and repumping ($F = 1 \rightarrow F' = 2$) transition respectively. The conversion factors of the voltage to magnetic fields are not given for the gradient coils. They can generate offset fields as well as gradient fields depending on the voltage of the other gradient coils so no limit independent of the setpoint of the other coil can be given.

		Optimizer values		Physical quantities	
Stage	Parameter	Lower	Upper	Lower	Upper
MOT	Cool—frequency	143.0 MHz	146.5 MHz	2.16 MHz	−25.84 MHz
MOT	Repump—frequency	317 MHz	323 MHz	−32.3 MHz	69.7 MHz
MOT	Cool—power	0 dB	20 dB	30 mW	4 mW
MOT	Repump—power	0 dB	30 dB	3.4 mW	0.1 mW
MOT	Gradient coils—X1	−4 V	−2.5 V	—	—
MOT	Gradient coils—X2	2.5 V	4 V	—	—
MOT	Offset coil—Z	0.6 V	1.0 V	1.4 G	2.3 G
Molasses	Cool—frequency	165 MHz	175 MHz	−174 MHz	254 MHz
Molasses	Repump—frequency	310 MHz	317 MHz	−152 MHz	−33 MHz
Molasses	Cool—power start	0 dB	20 dB	30 mW	4 mW
Molasses	Repump—power start	10 dB	20 dB	1.9 mW	0.2 mW
Molasses	Cool—power end	18 dB	30 dB	5 mW	0.1 mW
Molasses	Repump—power end	18 dB	30 dB	0.4 mW	0 mW
Molasses	Gradient coils—X1	−0.1 V	0.1 V	—	—
Molasses	Gradient coils—X2	0 V	0.2 V	—	—
Molasses	Offset coil—Z	−0.3 V	0 V	−6.8 G	0.0 G
Molasses	Offset coil—Y	−0.1 V	0.1 V	−6.8 G	6.8 G
Molasses	Duration	10 ms	30 ms	10 ms	30 ms

Appendix B. Best parameters reached in the 10 dimensional case

As discussed in section 4.1, parameters for optimal fitness gives us an insights into the experiment's sensitivity to their variation and their significance within the sequence. In the 10-dimensional case, table B1 displays the final parameters and their standard deviation for the 4 best performing optimizers. Values for other optimizers are omitted due to lower relevance, potentially being far from the absolute maximum.

Obviously, all optimisers converge into the same global maximum. Therefore, a smaller standard deviation between all 10 runs of the optimisers is associated with a steeper gradient around the found

Table B1. Mean values of the input parameters, including their standard deviation, used to reach the best fitness. The parameters of the four best performing optimizers are shown. A big standard deviation shows either little influence of this parameter on the result or the optimizer having difficulties in finding the optimal set.

Stage	Parameter	BO / JCM	CMA-ES	FMFN	AX
MOT	Cool—frequency (MHz)	145.08 ± 0.08	145.09 ± 0.07	145.0 ± 0.1	145.0 ± 0.1
MOT	Repump—frequency (MHz)	319.1 ± 0.4	319.0 ± 0.2	319.0 ± 0.4	318.9 ± 0.3
MOT	Cool—power (dB)	5 ± 5	6 ± 3	7 ± 2	6 ± 2
MOT	Repump—power (dB)	2 ± 2	4 ± 2	8 ± 5	4 ± 2
MOT	Gradient coils—X1 (V)	-3.72 ± 0.08	-3.80 ± 0.08	-3.8 ± 0.2	-3.7 ± 0.1
MOT	Gradient coils—X2 (V)	3.96 ± 0.05	3.98 ± 0.02	3.97 ± 0.05	3.9 ± 0.2
MOT	Offset coil—Z (V)	0.82 ± 0.08	0.80 ± 0.09	0.8 ± 0.1	0.82 ± 0.04
Molasses	Cool—frequency (MHz)	172 ± 2	171 ± 2	171 ± 2	171 ± 2
Molasses	Repump—frequency (MHz)	312 ± 2	313 ± 3	313 ± 2	312 ± 3
Molasses	Cool—power start (dB)	10 ± 3	15 ± 4	11 ± 4	10 ± 6

maximum. This highlights the parameter as influential. For shallow maxima, the scattering will be bigger since it is harder to differentiate between noise and actual improvement. Such a parameter would be less important, since a deviation does not affect the maximal achievable fitness much.

We evaluate the landscape around the maximum of three selected parameters, the detunings of the cooling and the repumping lasers, and the attenuation of the RF power provided to the cooling AOM during the MOT phase. These parameters were chosen as examples that exhibit certain features. The detuning of the cooling laser describes a precisely defined sharp maximum; the detuning of the repumping laser presents a shallow maximum; and finally, the RF attenuation scales non-linearly with the optical power and presents a large plateau. The attenuation parameter is important outside the constant region, but it loses any significance in the plateau, as no further improvement can be achieved. Since we saw that both JCM and CMA-ES found the global maximum, we evaluate their behavior first to assess the shape of the landscape.

First, we look at the frequency of the cooling laser. It presents a similarly small standard deviation for both optimizers (0.08 MHz for JCM and 0.07 MHz for CMA-ES) and thus a sharp maximum. As expected, this parameter has a significant influence on the final atom number since it is a key parameter for the achievable trap depth.

In contrast, the repumping laser has a significantly smaller influence, as highlighted by the substantially higher standard deviation of 0.4 MHz for JCM and 0.2 MHz for CMA-ES. The presence of repumping is necessary to re-excite the small fraction of atoms that fall into a dark state during the trap loading, but the detuning itself does not contribute significantly to the final number of trapped atoms. This leads to a shallow maximum in the landscape and thus to a bigger standard deviation of the optimized parameter.

Finally, we can look at the attenuation of RF power supplied to the cooling AOM during the MOT phase. Both JCM and CMA-ES identify the plateauing area between 0 dB and 10 dB: JCM finds a mean value of (5 ± 5) dB, while CMA-ES (6 ± 3) dB. At attenuation values that are higher than 10 dB the atom number falls since the trap depth is reduced.

It is also interesting to analyse the parameter behaviour for AX and FMFN, which perform well, but do not match the high fitness levels reached by JCM and CMA-ES. The standard deviation of the frequency of the cooling laser (0.1 MHz) is slightly bigger for both compared to JCM/CMA-ES. The actual parameter values differ by 0.08 MHz, which, when tested experimentally, leads to a 0.8% decrease in atom number. A similar value and standard deviation is reached for the repumping detuning. The main differences occur in the case of the attenuation of the RF power to the AOM. The standard deviation is significantly smaller with 2 dB compared to JCM (5 dB) and slightly smaller compared to CMA-ES(3 dB). The smaller standard deviations indicate that FNFM, AX and CMA-ES focus more on this parameter than JCM, despite the existance of other more influential parameters. If an algorithm focusses on the wrong parameter, more iterations overall are needed to reach the same fitness value. Evidence for this can be seen in the lack of fitness plateau for some of the optimizers. Additionally, small deviations add up. For example, if a single parameter (the cooling detuning) causes barely visible deviation of 0.8%, then across 10 parameters, this is a loss of 8%.

We can see exactly this behavior for AX and FMFN.

Appendix C. Best parameters reached in the 18 dimensional case

As in the 10-dimensional case for JCM, CMA-ES, PSO and AX, we can examine the experiment's sensitivity to parameter variation and thier significance through the standard deviations displayed in table C1. In this case, we examine the detunings of the cooling and repumping lasers, as well as the attenuation of the cooling laser during the MOT phase.

Table C1. Mean values of the input parameters, including their standard deviation, used to reach the best fitness. The parameters of the four best performing optimizers are shown. A big standard deviation shows either little influence of this parameter on the result or the optimizer having difficulties in finding the optimal set.

Stage	Parameter	BO / JCM	CMA-ES	PSO	AX
MOT	Cool—frequency (MHz)	145.0 ± 0.1	145.0 ± 0.1	145 ± 0.2	145 ± 0.1
MOT	Repump—frequency (MHz)	319.1 ± 0.4	319.2 ± 0.4	318.5 ± 0.6	319 ± 0.2
MOT	Cool—power (dB)	6 ± 3	7 ± 2	6 ± 5	7 ± 2
MOT	Repump—power (dB)	5 ± 2	6 ± 3	3 ± 3	6 ± 2
MOT	Gradient coils—X1 (V)	−3.7 ± 0.2	−3.8 ± 0.2	−3.8 ± 0.3	−3.5 ± 0.2
MOT	Gradient coils—X2 (V)	3.90 ± 0.07	3.95 ± 0.01	3.9 ± 0.2	3.6 ± 0.2
MOT	Offset coil—Z (V)	0.9 ± 0.1	0.81 ± 0.09	0.8 ± 0.1	0.80 ± 0.06
Molasses	Cool—frequency (MHz)	170 ± 2	170 ± 2	169 ± 3	170 ± 1
Molasses	Repump—frequency (MHz)	312 ± 1	312 ± 2	313 ± 3	313 ± 1
Molasses	Cool—power start (dB)	9 ± 5	9 ± 5	11 ± 8	11 ± 4
Molasses	Repump—power start (dB)	15 ± 2	16 ± 3	15 ± 15	16 ± 1
Molasses	Cool—power end (dB)	24 ± 2	23 ± 4	24 ± 4	23 ± 2
Molasses	Repump—power end (dB)	22 ± 2	22 ± 3	24 ± 5	23 ± 1
Molasses	Gradient coils—X1 (V)	0.02 ± 0.05	0.04 ± 0.04	−0.03 ± 0.06	0.00 ± 0.03
Molasses	Gradient coils—X2 (V)	0.08 ± 0.05	0.13 ± 0.05	0.11 ± 0.08	0.09 ± 0.03
Molasses	Offset coil—Z (V)	−0.18 ± 0.07	−0.19 ± 0.08	−0.2 ± 0.1	−0.16 ± 0.03
Molasses	Offset coil—Y (V)	−0.01 ± 0.03	0.02 ± 0.02	0.02 ± 0.07	0.00 ± 0.02
Molasses	Duration (ms)	18 ± 4	18 ± 5	14 ± 6	20 ± 3

The standard deviations of the cooling frequency for all considered optimizers is comparable to the one achieved by AX and FMFN in the 10-dimensional case. For JCM and CMA-ES this highlights a deterioration compared to the result in the 10-dimensional case.

The results for the detuning of the repumping laser do not seem to be affected by the increase of the dimensionality for JCM, CMA-ES and AX, while PSO not only converges to a different value but also presents a larger standard deviation. This shift in the detuning also leads to a lower achieved fitness value.

This behaviour repeats for the attenuation of the cooling AOM: JCM, CMA-ES, and AX now result in a standard deviation of 2–3 dB, a value similar to that found by AX in the 10-dimensional case. The PSO optimizer instead presents a larger standard deviation of 5 dB, probably due to the swarm approach in a noisy environment.

In general we see that for multiple parameters the standard deviation became slightly bigger compared to the 10 dimensional case. This, combined with the fact that the optimizers no longer reach the previous fitness values, shows how these seemingly insignificant changes in total significantly affect the final fitness. This underlines how optimizing many parameters at once is not necessarily beneficial for finding the best parameter set in the case of noisy environments.

If we consider the significance of the additional parameters considered in the 18-dimensional case by examining their standard deviations, we realise that some are more crucial than others. We find that, for all optimizers, the parameters controlling gradient and offset coils have rather small standard deviations. This shows their importance during the molasses phase: such a result is expected since the canceling of stray magnetic fields during molasses is crucial. All the other additional parameters exhibit rather large standard deviations, and are therefore of lower significance.

It is worth highlighting how the start and end points of the attenuation ramps of the AOMs, as well as the overall duration of the molasses phase, exhibit large standard deviations in both CMA-ES and PSO. While these variations do not significantly influence the atom number, which is chosen as the fitness function, they are key parameters for another property of the cloud: its temperature. A fully optimised atomic ensemble is ideally characterised by a high atom number and low temperature. Even though we rely on the atom number as fitness fuction, we could partially take into account the atom temperature by applying the optimisation algorithms to the cloud after some time of flight. Hotter atoms leave the detection zone faster as soon as the trap is released, which will translate into a lower atom number whenever the atoms are hot enough to leave the field of view before the detection. This is reflected in the smaller standard deviations associated with these parameters (i.e. start and end points of the attenuation ramps of the AOMs, and duration of the molasses phase) within JCM and AX.

Appendix D. Experimental control software

The control software of the experiment is a server-client-based system using Python. In our system the client handles the graphical user interface and user-input to the experiment.

The server runs on the experiment control PC. It connects to all devices needed to run the experiment like cameras as well as Sinara, handles the communication between them and external software for data evaluation, storage and optimization. The workflow within the software for running the experiment can be seen in the server box of figure 2. Once the server receives instructions for a sequence, this table is compiled into instructions executable by the Sinara system before being queued for execution in the sequence player. After the sequence is played, the Sinara system signals to the server that data can be read out from devices. The acquired data is sent for evaluation in external software and the results are sent back to the server. Back at the server the data can be used for optimization processes which generate a new sequence using the suggestions from the optimization algorithm selected.

A single sequence can consist of multiple time steps addressing many channels of the Sinara hardware. The resulting table can become rather complex and thus laboratory routines can be optimized by the saving and loading of previously played sequences. Here, each unique, played sequence is stored in a MySQL database with a hash generated from its content for identification and to avoid duplication. Similarly, each time a sequence is played, a measurement hash is created, which includes the time of execution and any volatile settings like calibrations and camera settings. The measurement hashes are linked to the sequence hash and can be used to identify saved data.

Absorption images are evaluated and post-processed in tailored Python analysis software we call Pical (Picture analysis for cold atoms). Atom number and optical density can be evaluated via either a full 2D Gaussian fit, or via two 1D fits of row and column sums. The software is also capable of analysing sequences of pictures, which allows for the determination of the expansion rate or temperature of the cloud. One can implement a range of other post-processing calculations such as calculating the phase-space density of the cloud, or by evaluating an alternative cost function.

ORCID iDs

Oliver Anton https://orcid.org/0009-0009-8494-5071
Victoria A Henderson https://orcid.org/0000-0002-9233-589X
Elisa Da Ros https://orcid.org/0000-0003-0267-0350
Ivan Sekulic https://orcid.org/0009-0000-2414-5595
Sven Burger https://orcid.org/0000-0002-3140-5380
Philipp-Immanuel Schneider https://orcid.org/0000-0002-9949-9483

References

[1] Dalibard J and Cohen-Tannoudji C 1989 *J. Opt. Soc. Am.* B **6** 2023
[2] Metcalf H J and Van der Straten P 1999 *Laser Cooling and Trapping* (Springer) (https://doi.org/10.1007/978-1-4612-1470-0)
[3] Ketterle W and Druten N V 1996 Evaporative cooling of trapped atoms *Advances In Atomic, Molecular and Optical Physics* vol 37 (Elsevier) pp 181–236
[4] Ketterle W, Durfee D S and Stamper-Kurn D M 1999 Making, probing and understanding Bose–Einstein condensates (arXiv:cond-mat/9904034)
[5] Freier C, Hauth M, Schkolnik V, Leykauf B, Schilling M, Wziontek H, Scherneck H G, Müller J and Peters A 2016 *J. Phys.: Conf. Ser.* **723** 1
[6] Stray B *et al* 2022 *Nature* **602** 590–4
[7] Wu X, Pagel Z, Malek B S, Nguyen T H, Zi F, Scheirer D S and Müller H 2019 *Sci. Adv.* **5** 1–10
[8] Ménoret V, Vermeulen P, Le Moigne N, Bonvalot S, Bouyer P, Landragin A and Desruelle B 2018 *Sci. Rep.* **8** 12300
[9] Antoni-Micollier L, Carbone D, Ménoret V, Lautier-Gaud J, King T, Greco F, Messina A, Contrafatto D and Desruelle B 2022 *Geophys. Res. Lett.* **49** e2022GL097814
[10] Poli N, Schioppo M, Vogt S, Falke S, Sterr U, Lisdat C and Tino G 2014 *Appl. Phys.* B **117** 1107–16
[11] Grotti J *et al* 2018 *Nat. Phys.* **14** 437–41
[12] Takamoto M, Ushijima I, Ohmae N, Yahagi T, Kokado K, Shinkai H and Katori H 2020 *Nat. Photon.* **14** 411–5
[13] Gellesch M, Jones J, Barron R, Singh A, Sun Q, Bongs K and Singh Y 2020 *Adv. Opt. Technol.* **9** 313–25
[14] Beloy K *et al* (Boulder Atomic Clock Optical Network (BACON) Collaboration) 2021 *Nature* **591** 564–9
[15] Elliott E R, Krutzik M C, Williams J R, Thompson R J and Aveline D C 2018 *npj Microgravity* **4** 16
[16] Frye K *et al* 2021 *EPJ Quantum Technol.* **8** 1
[17] Devani D *et al* 2020 *CEAS Space J.* **12** 539–49
[18] Aveline D C *et al* 2020 *Nature* **582** 193–7
[19] Sidhu J S *et al* 2021 *IET Quantum Commun.* **2** 182–217
[20] Ahlers H *et al* 2022 arXiv:2211.15412
[21] Alonso I *et al* 2022 *EPJ Quantum Technol.* **9** 30
[22] Da Ros E, Kanthak S, Sağlamyürek E, Gündoğan M and Krutzik M 2023 *Phys. Rev. Res.* **5** 033003
[23] Rohringer W, Bücker R, Manz S, Betz T, Koller C, Göbel M, Perrin A, Schmiedmayer J and Schumm T 2008 Stochastic optimization of a cold atom experiment using a genetic algorithm *Appl. Phys. Lett.* **93** 264101
[24] Geisel I, Cordes K, Mahnke J, Jöllenbeck S, Ostermann J, Arlt J, Ertmer W and Klempt C 2013 *Appl. Phys. Lett.* **102** 21
[25] Tranter A D, Slatyer H J, Hush M R, Leung A C, Everett J L, Paul K V, Vernaz-Gris P, Lam P K, Buchler B C and Campbell G T 2018 *Nat. Commun.* **9** 4360

[26] Wu Y, Meng Z, Wen K, Mi C, Zhang J and Zhai H 2020 *Chin. Phys. Lett.* **37** 10
[27] Reinschmidt M, Fortágh J, Günther A and Volchkov V 2023 Reinforcement learning in ultracold atom experiments (arXiv:2306.16764)
[28] Wigley P B *et al* 2016 *Sci. Rep.* **6** 1–6
[29] Nakamura I, Kanemura A, Nakaso T, Yamamoto R and Fukuhara T 2019 *Opt. Express* **27** 20435
[30] Barker A J, Style H, Luksch K, Sunami S, Garrick D, Hill F, Foot C J and Bentine E 2020 *Mach. Learn.: Sci. Technol.* **1** 015007
[31] Davletov E T, Tsyganok V V, Khlebnikov V A, Pershin D A, Shaykin D V and Akimov A V 2020 *Phys. Rev.* A **102** 11302
[32] Ma J *et al* 2023 arXiv:2303.05358
[33] Milson N, Tashchilina A, Ooi T, Czarnecka A, Ahmad Z F and LeBlanc L J 2023 *Mach. Learn.: Sci. Technol.* **4** 045057
[34] Poli R, Kennedy J and Blackwell T 2007 *Swarm Intell.* **1** 33–57
[35] Kaveh A and Zolghadr A 2014 *Comput. Struct.* **130** 10–21
[36] CMA-ES 2019 Covariance matrix adaptation evolution strategy for non-linear numerical optimization in Python (available at: https://pypi.org/project/cma/)
[37] Nelder J A and Mead R 1965 *Comput. J.* **7** 308–13
[38] Meta Platforms, Inc 2023 Bayesian optimization (available at: https://ax.dev/)
[39] Nogueira F 2014 Bayesian optimization: open source constrained global optimization tool for Python (available at: https://github.com/bayesian-optimization/BayesianOptimization)
[40] Letham B, Karrer B, Ottoni G and Bakshy E 2019 *Bayesian Anal.* **14** 495–519
[41] OptimalQT 2023 (available at: https://jcmwave.com/company/projects/item/1052-optimal-qt)
[42] Zhang Y, Wang S, Ji G *et al* 2015 A comprehensive survey on particle swarm optimization algorithm and its applications *Math. Probl. Eng.* **2015** 1–38
[43] Das S and Suganthan P N 2010 *IEEE Trans. Evolution. Comput.* **15** 4–31
[44] Hansen N 2006 The CMA evolution strategy: a comparing review *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms* (Springer) pp 75–102
[45] Shahriari B, Swersky K, Wang Z, Adams R P and De Freitas N 2015 *Proc. IEEE* **104** 148–75
[46] Rasmussen C E *et al* 2006 *Gaussian Processes for Machine Learning* vol 1 (Springer) (https://doi.org/10.7551/mitpress/3206.001.0001)
[47] Bull A D 2011 *J. Mach. Learn. Res.* **12** 10
[48] Picheny V, Wagner T and Ginsbourger D 2013 *Struct. Multidisc. Optim.* **48** 607–26
[49] Snoek J, Larochelle H and Adams R P 2012 *Advances in Neural Information Processing Systems* vol 25
[50] Schneider P I, Garcia Santiago X, Soltwisch V, Hammerschmidt M, Burger S and Rockstuhl C 2019 *ACS Photonics* **6** 2726–33
[51] ColdQuanta, Inc., DBA Infleqtion 2023 RuBECi (available at: www.shopcoldquanta.com/rubeci)
[52] Wicht A, Bawamia A, Krüger M, Kürbis C, Schiemangk M, Smol R, Peters A and Tränkle G 2017 *Proc. SPIE* **10085** 100850F
[53] Anton O 2023 Pical—Picture analysis for cold atoms (available at: https://git.physik.hu-berlin.de/pical/pical-picture-analysis-for-cold-atoms)
[54] M-Labs 2023 Sinara hardware (available at: https://m-labs.hk/experiment-control/sinara-core/)