training_set.txt		
-4.195979899497488	0.527638190954774	-0.194370042271220
-2.939698492462311	-3.190954773869347	-0.005885116036303
-0.075376884422110	-4.195979899497488	-0.225181757558616
⋮	⋮	⋮

Table 1: Form of the provided `training_set.txt` file.

		model_data.txt
Number of neurons	--→	5
Lower bound $\mu_{\min}$	--→	-5
Upper bound $\mu_{\max}$	--→	5
Lower bound $\sigma_{\min}$	--→	0.1
Upper bound $\sigma_{\max}$	--→	2
Lower bound $\theta_{\min}$	--→	-20
Upper bound $\theta_{\max}$	--→	20

Table 2: Form of the `model_data.txt` file.

We offer a simple application example that comprehensively describes the workings of the proposed software and can be used as a guide for applying the VCB algorithm to more complicated problems.

We assume that the user has copied the folder `SOMO-VCB/` of our software into a desirable folder and has changed the working directory of Matlab® to that folder. For example, if the SOMO-VCB folder is copied in the `/home/user/software/` folder, the user shall change the working directory by giving the following command in the Matlab® command window:

```
>> cd /home/user/software/SOMO-VCB/
```

Firstly, the user shall select its training set and store it at file `training_set.txt`. For the current example, we considered the approximation of the 2-dimensional *Mexican-Hat* test function, given by:

$$f(x) = \frac{\sin\left(x_1^2 + x_2^2\right)}{\sqrt{x_1^2 + x_2^2}}, \tag{1}$$

Where:

$$x = (x_1, x_2) \in X \triangleq [-5, 5] \times [-5, 5].$$

Each file line contains a 2-dimensional point followed by its function value, i.e., three decimal numbers given in 15 decimal digits as depicted in Table 1.

Having decided on the experimental setting to run, the user shall take the following steps:

(S1) Copy the corresponding `training_set.txt` file into the `data/` folder.

		vcb_data.txt
Number of mini-batches	--→	10
Percentage of patterns used per mini-batch set	--→	0.1
Number of VCB cycles	--→	10
Number of single-pattern network evaluations	--→	10000000

Table 3: Form of the `vcb_data.txt` file.

		so_data.txt
Number of experiments	--→	2
Number of optimizer iterations	--→	100
Line search iterations	--→	100
Lower bound $\lambda_{\min}$	--→	0.1
Upper bound $\lambda_{\max}$	--→	70
Minimum relative improvement	--→	1.e-4
Minimum gradient-norm	--→	1.e-4
Parameter ρ_1	--→	1.e-4
Parameter ρ_2	--→	0.1

Table 4: Form of the `so_data.txt` file.

(S2) In the `data/` folder, the user shall modify the `model_data.txt` file which provides the necessary network-related information. In our example, the function is approximated using an RBF neural network. The file for our 5-neurons RBF network is depicted in Table 2 and implies that it had all its center vector components in the range $[-5, 5]$, its standard deviations in $[0.1, 2]$, and its weights in $[-20, 20]$.

(S3) The user shall revise the `vcb_data.txt` in the `data/` folder. In our experiments, we used the file depicted in Table 3, i.e., each mini-batch set consisted of 10 mini-batches, each one formed by randomly selecting 10% of the training patterns from the training set. Also, the number of VCB cycles in the specific instance is fixed and equal to 10, while the available computation budget is 10000000 single-pattern evaluations.

(S4) The user shall adjust the parameter file for the selected optimization approach into the `data/` folder. Tables 4 and 5 report the files used in our experiments for the single-objective and the multi-objective case, respectively. We performed 2 independent experiments with the BFGS algorithm in the single-objective case. The maximum number of iterations of the algorithm was 100, used as a termination condition in case the user does not specify the number of VCB cycles. Moreover, line search was given a maximum of 100 iterations, while the domain of the penalty parameter λ was $[0.1, 70]$. The minimum relative improvement and the gradient-norm tolerance were set to 10^{-4}. Finally, the parameters of the Wolfe-Powell conditions were set to 10^{-4} and 0.1, respectively.

		mo_data.txt
Number of objective functions	--→	2
Number of experiments	--→	2
Swarm size	--→	10
Repository size	--→	10
MOPSO iterations	--→	100
Iterations with unchanged repository	--→	20
Inertia coefficient	--→	0.729
Parameter ϕ_1	--→	1.49
Parameter ϕ_2	--→	1.49
Intervals for adaptive grid	--→	20
Velocity clamping percentage	--→	0.2
Mutation parameter	--→	0.25
Pareto set evaluation strategy	--→	0

Table 5: Form of the `mo_data.txt` file.

Screen output (`screenOutput = ‘‘off’’`)

```
SO-VCB ALGORITHM
All data has been read, and output files are ready.
Proceeding to experiments:

Running Experiment 1...
...finished (elapsed time:  7.71 sec).

Running Experiment 2...
...finished (elapsed time:  7.14 sec).
```

Table 6: Screen output running `so_vcb_main` with the parameter `screenOutput = ‘‘off’’` in `so_vcb_main.m`.

For the multi-objective case, the number of objectives was set to 2, and the number of experiments was 2. Both the swarm and repository sizes were equal to 10. MOPSO was allowed to run for 100 iterations in case of an unspecified number of VCB cycles, while after every 20 iteration with an unchanged repository, the swarm was restarted. The velocity-update parameters were 0.729 and 1.49, while the adaptive grid assumed a partitioning of 20 intervals per dimension. The velocity clamping percentage was 20%, and the mutation parameter was equal to 0.25. Eventually, the evaluation strategy was set to 0 for evaluating the whole Pareto optimal set.

(S5) Having set all the aforementioned files, the user can directly run the algorithm by running the `so_vcb_main` or the `mo_vcb_main` command in the Matlab® command window.
The user has two options regarding the screen output during the run. Setting the parameter `screenOutput` to “off” in the `so_vcb_main.m` or `mo_vcb_main.m` file, the user receives only essential information on the

Screen output (`screenOutput = ''on''`)

```
SO-VCB ALGORITHM
All data has been read, and output files are ready.
Proceeding to experiments:

Running Experiment 1...
    Cycle 1 ...  f*=20.4209072115 ...  NNeval=153600/10000000 (15.4%)
    Cycle 2 ...  f*=5.4099053397 ...  NNeval=283200/10000000 (28.3%)
    Cycle 3 ...  f*=1.5491444294 ...  NNeval=391200/10000000 (39.1%)
    Cycle 4 ...  f*=1.3728664490 ...  NNeval=552000/10000000 (55.2%)
    Cycle 5 ...  f*=1.2921087204 ...  NNeval=712800/10000000 (71.3%)
    Cycle 6 ...  f*=0.9590738342 ...  NNeval=780000/10000000 (78.0%)
    Cycle 7 ...  f*=0.8961949619 ...  NNeval=876000/10000000 (87.6%)
    Cycle 8 ...  f*=0.8403128739 ...  NNeval=928800/10000000 (92.9%)
    Cycle 9 ...  f*=0.8059666415 ...  NNeval=1089600/10000000 (109.0%)
...finished (elapsed time:  7.57 sec).
                        .
                        .
                        .
```

Table 7: Screen output running `so_vcb_main` with the parameter `screenOutput = ''on''` in `so_vcb_main.m`.

`*_report`

1	0.8059666415	9	1089600	7.57
2	0.1431546082	8	1106400	7.73

Table 8: Form of the `*_report` file.

screen, such as the start and the end of each experiment and the elapsed time. Table 6 reports this type of output. On the other hand, setting `screenOutput` to "on", which is also the default value, the program prints a line per VCB cycle, reporting the best solution value `f*` up to that cycle, as well as the neural network evaluations spent (in number and percentage of the total available computation budget). Table 7 illustrates this type of output. In the multi-objective approach, the number of Pareto optimal solutions is also reported per cycle.

(S6) After the end of all experiments, the three output files are written in the `results/` folder. The `*_log` files (where "`*`" stands for either "`so`" or "`mo`") differ between the single-objective and the multi-objective approach because they report the corresponding algorithm parameters along with the VCB and network parameters. However, the other two files, namely `*_report` and `*_solution`, contain the same information for both approaches. Table 8 illustrates such a `*_report` file. Each line consists of the experiment number, the total MSE of the detected solution over the whole training set, the number of VCB cycles, the total number of network evaluations, and the elapsed time. For instance, in the second experiment (2-nd line), the detected solution had

	*_solution				
1	2.6873326262	-13.3158690724	9.7231230055	.....	3.6376235658
2	-10.2043261403	-10.2745952288	0.0004279571	.....	0.7325618163

Table 9: Form of the `*_solution` file. The vectors are trimmed due to length limitation.

a total MSE equal to 0.1431546082, and the algorithm spent 1106400 network evaluations while running for 7.73 seconds (wall-clock time). The corresponding solution vectors are reported in Table 9.