Computation of Physiologic Closed-Loop Controller Performance Metrics

Tool Reference

- RST Reference Number: RST26IP01.01
- Date of Publication: 7/14/2026
- Recommended Citation: U.S. Food and Drug Administration. (2026). *Computation of Physiologic Closed-Loop Controller Response Metrics* (RST26IP01.01). https://cdrh-rst.fda.gov/computation-physiologic-closed-loop-controller-response-metrics

For more information

Catalog of Regulatory Science Tools to Help Assess New Medical Devices

Disclaimer

About the Catalog of Regulatory Science Tools

The enclosed tool is part of the Catalog of Regulatory Science Tools, which provides a peer-reviewed resource for stakeholders to use where standards and qualified Medical Device Development Tools (MDDTs) do not yet exist. These tools do not replace FDA-recognized standards or MDDTs. This catalog collates a variety of regulatory science tools that the FDA's Center for Devices and Radiological Health's (CDRH) Office of Science and Engineering Labs (OSEL) developed. These tools use the most innovative science to support medical device development and patient access to safe and effective medical devices. If you are considering using a tool from this catalog in your marketing submissions, note that these tools have not been qualified as Medical Device Development Tools and the FDA has not evaluated the suitability of these tools within any specific context of use. You may request feedback or meetings for medical device submissions as part of the Q-Submission Program.

For more information about the Catalog of Regulatory Science Tools, email RST_CDRH@fda.hhs.gov.

User Manual

Overview

This software package provides tools to compute physiologic closed-loop controller (PCLC) performance metrics from time-series data of a controlled physiologic variable. The package is available in both MATLAB and Python.

Disclaimer: The mention of commercial products, their sources, or their use in connection with material reported herein is not to be construed as either an actual or implied endorsement of such products by the Department of Health and Human Services.

MATLAB package files

- **evaluate_CL_metrics.m**: the PCLC performance metric computation function
- **test_data.mat**: an example dataset for demonstration purposes
- **run_example.m**: an example script demonstrating how to load data, call the function, and visualize results

Python package files

- **evaluate_CL_metrics.py**: the PCLC performance metric computation function
- **test_data.mat**: the same example dataset, loaded via scipy.io.loadmat
- **run_example.py**: an example script demonstrating how to load data, call the function, and visualize results

> **NOTE:** *The Python implementation is an exact translation of the MATLAB function. All equations, thresholds, and logic are identical. ind_CL_start and ind_CL_end are passed as 1-based integers in both implementations, matching MATLAB convention.*

Data Requirements

Before calling the function, format input data as follows. While variable names below use MAP-specific terminology consistent with the provided example dataset, the function can be applied to other physiologic variables. Users applying the tool to other applications can substitute the appropriate controlled variable and disturbance signal in place of MAP_Values and Hemorrhage_Values, respectively.

- **MAP_Values** — full time-series of the controlled physiologic variable (e.g., MAP in mmHg)
- **MAP_Times** — corresponding time vector in minutes, same length as MAP_Values
- **Hemorrhage_Values** — full time-series of the disturbance signal (e.g., hemorrhage rate in l/min); set to a zero vector if no disturbance was applied
- **ind_CL_start** — scalar integer index (1-based) at which controller engagement begins
- **ind_CL_end** — scalar integer index (1-based) at which controller engagement ends
- **target** — scalar target value of the controlled variable (e.g., 70 for MAP in mmHg)

The function accepts the data as provided and does not apply any additional processing to the controlled variable signal. Data acquisition and preprocessing can be considered by the user prior to calling the function.

NOTE: *ind_CL_start and ind_CL_end use 1-based indexing in both MATLAB and Python, where index 1 corresponds to the first element of the vector. In Python, the function converts these to 0-based indices internally; the user does not need to subtract 1.*

Step-by-Step Instructions — MATLAB

1. **Step 1:** Place evaluate_CL_metrics.m, test_data.mat, and run_example.m in the same MATLAB working directory.
2. **Step 2:** Open MATLAB® R2023b or a compatible version.
3. **Step 3:** Run the example script run_example.m to verify correct installation and review the expected outputs using the provided example dataset.
4. **Step 4:** Review the output struct PCLC_char for the computed metric values. Access fields using dot notation, for example PCLC_char.MDPE. NaN values indicate metrics that could not be computed for the given subject.
5. **Step 5:** When reporting results across a cohort of multiple subjects, call evaluate_CL_metrics.m separately for each subject.

Step-by-Step Instructions — Python

Installation

Python 3.8 or later is required. Install the required packages by running the following command in a command line window:

```
pip3 install numpy pandas scipy matplotlib
```

1. **Step 1:** Place evaluate_CL_metrics.py, test_data.mat, and run_example.py in the same folder.
2. **Step 2:** Open a command line window and navigate to the folder containing the files. For example:

   ```
   cd "/path/to/your/folder"
   ```
3. **Step 3:** Run the example script to verify correct installation and review the expected outputs:

   ```
   python3 run_example.py
   ```
4. **Step 4:** Review the output dictionary PCLC_char for the computed metric values. Access fields using bracket notation, for example PCLC_char['MDPE']. NaN values (float('nan')) indicate metrics that could not be computed for the given subject. To check for NaN in Python use import math; math.isnan(PCLC_char['risetime']).
5. **Step 5:** When reporting results across a cohort of multiple subjects, call evaluate_CL_metrics separately for each subject.

NOTE: *If your .mat file was saved in MATLAB version 7.3 format (HDF5), scipy.io.loadmat will raise a NotImplementedError. In that case, use h5py: import h5py; f = h5py.File('my_data.mat', 'r').*

Output

MATLAB

The function returns a MATLAB struct named PCLC_char. Fields are accessed using dot notation: PCLC_char.MDPE, PCLC_char.risetime, etc.

Python

The function returns a Python dictionary named PCLC_char. Fields are accessed using bracket notation: PCLC_char['MDPE'], PCLC_char['risetime'], etc. All values are Python floats. NaN is represented as float('nan').

Output Metrics Table

The following metrics are returned in both implementations [1,2]. Units shown reflect the provided example dataset (MAP-based automated fluid resuscitation).

Metric	Symbol	Units	MATLAB field	Python key
Median Performance Error	MDPE	%	`PCLC_char.MDPE`	`PCLC_char['MDPE']`
Median Absolute Performance Error	MDAPE	%	`PCLC_char.MDAPE`	`PCLC_char['MDAPE']`
Wobble	Wob	%	`PCLC_char.Wobble`	`PCLC_char['Wobble']`
Percentage of Time Within 10% of Target	POT(10)	%	`PCLC_char.per10`	`PCLC_char['per10']`
Percentage of Time Within 20% of Target	POT(20)	%	`PCLC_char.per20`	`PCLC_char['per20']`
Percentage of Time Within 30% of Target	POT(30)	%	`PCLC_char.per30`	`PCLC_char['per30']`
Modified Rise Time	Tr	min	`PCLC_char.risetime`	`PCLC_char['risetime']`
Percentage Overshoot	%O.S.	%	`PCLC_char.overshoot`	`PCLC_char['overshoot']`
Settling Time	Ts	min	`PCLC_char.settlingtime`	`PCLC_char['settlingtime']`
Divergence	D	%/min	`PCLC_char.divergence`	`PCLC_char['divergence']`
Steady State Error	SSE	mmHg	`PCLC_char.SSE`	`PCLC_char['SSE']`

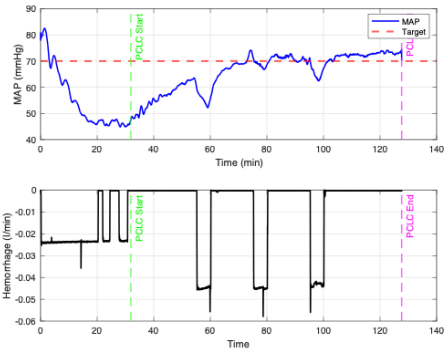
NaN is returned for risetime, overshoot, and settlingtime when the conditions for their computation are not met. When applying statistical analyses and reporting summary statistics across a cohort, methods for handling NaN values can be considered.

These metrics are designed for PCLC evaluation scenarios in which the controller may be subject to disturbances during the engagement period. In the representative use case of automated fluid resuscitation, these disturbances take the form of hemorrhagic events that perturb MAP during closed-loop controller engagement. Several metrics, including settling time and steady state error, include specific provisions to account for the presence of such disturbances. Users applying this function to other PCLC applications can supply the appropriate disturbance signal for their specific context, or a zero vector if no disturbance is applied.

Example Results

The following example results are generated by running run_example.m (MATLAB) or run_example.py (Python) with the provided example dataset (test_data.mat). The example dataset represents a MAP-based automated fluid resuscitation scenario in which a closed-loop controller regulates MAP toward a target of 70 mmHg in an anesthetized swine subject undergoing hemorrhage and fluid resuscitation, consistent with the experimental protocol described in [1]. This example is provided for demonstration and test purposes only. It illustrates the expected format and range of outputs and does not constitute a performance benchmark or acceptance criterion.

(a)



(b)

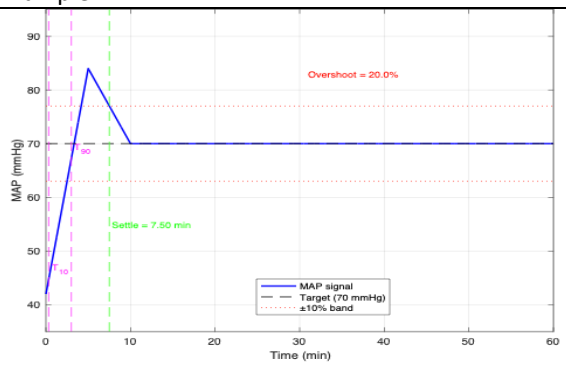
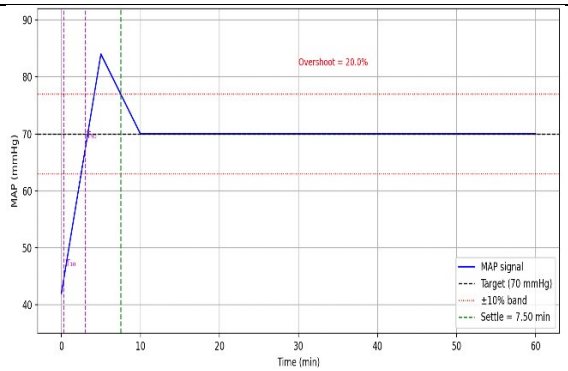
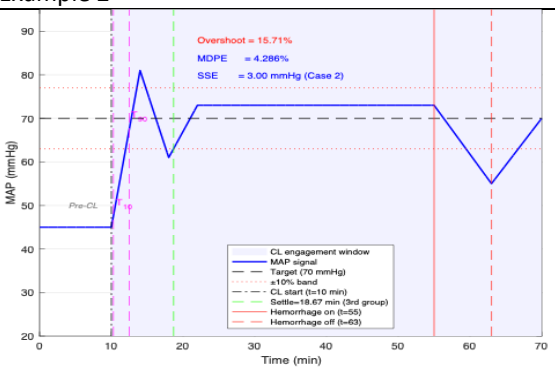
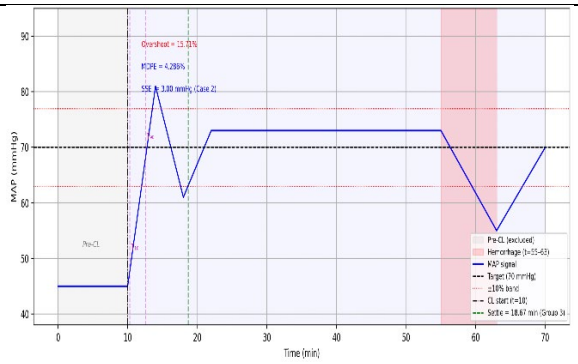
```
PCLC_char =
  Metric            Value
  ---------------   ------------
  MDPE              -1.4026
  MDAPE             4.1039
  Wobble            5.1025
  per10             68.5326
  per20             86.4648
  per30             97.9112
  risetime          32.3000
  overshoot         6.3673
  settlingtime      62.1500
  divergence        -0.2264
  SSE               3.1427
```

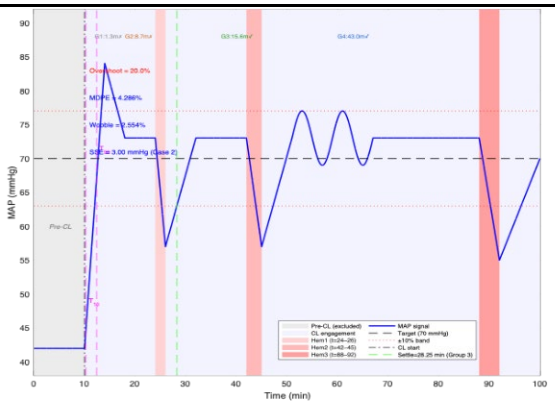
Figure 1: (a) MAP response and hemorrhage profile generated by running run_example.m or run_example.py with the provided example dataset. The top panel shows MAP over time with the target value, PCLC start, and PCLC end indicated. The bottom panel shows the hemorrhage disturbance profile with the same engagement markers; (b) PCLC performance metrics for the example.

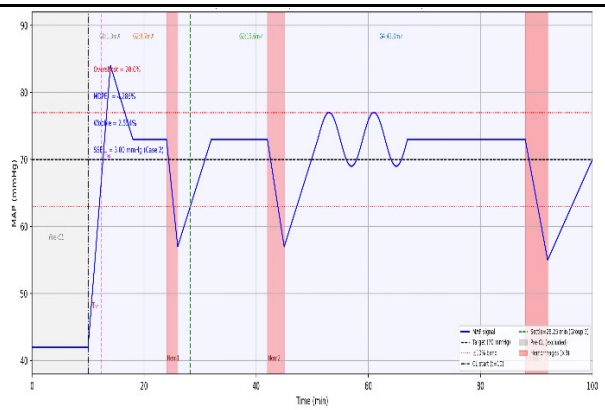
For applications other than MAP-based fluid resuscitation, the example results will differ based on the controlled variable, target value, disturbance profile, and controller behavior specific to the application.

Software Code Testing Examples

Manufactured data with known PCLC performance metrics are used for code testing. The tables below show results from both MATLAB and Python implementations across three manufactured signal examples, confirming equivalence between implementations.

MATLAB	Python
Example 1	
	
========== Metric Expected Computed Status ---------- MDPE (%) 0.0000 0.0000 PASS MDAPE (%) 0.0000 0.0000 PASS Wobble (%) 0.0000 0.0000 PASS per10 (%) 90.2516 90.2516 PASS per20 (%) 97.2005 97.2005 PASS per30 (%) 98.6002 98.6002 PASS risetime (min) 2.6700 2.6700 PASS overshoot (%) 20.0000 20.0000 PASS settlingtime (min) 7.5000 7.5000 PASS divergence (%/min) < 0 −0.1957 PASS (sign < 0) SSE (mmHg) 0.0000 0.0000 PASS ----------	========== Metric Expected Computed Status ---------- MDPE (%) 0.0000 0.0000 PASS MDAPE (%) 0.0000 0.0000 PASS Wobble (%) 0.0000 0.0000 PASS per10 (%) 90.2516 90.2516 PASS per20 (%) 97.2005 97.2005 PASS per30 (%) 98.6002 98.6002 PASS risetime (min) 2.6700 2.6700 PASS overshoot (%) 20.0000 20.0000 PASS settlingtime (min) 7.5000 7.5000 PASS divergence (%/min) < 0 −0.1957 PASS (sign < 0) SSE (mmHg) 0.0000 0.0000 PASS ----------
Example 2	
	
========== Metric Expected Computed Status ---------- MDPE (%) 4.2857 4.2857 PASS MDAPE (%) 4.2857 4.2857 PASS Wobble (%) 0.0000 0.0000 PASS per10 (%) 80.6366 80.6366 PASS per20 (%) 96.4339 96.4339 PASS per30 (%) 99.2501 99.2501 PASS risetime (min) 2.2200 2.2200 PASS overshoot (%) 15.7143 15.7143 PASS settlingtime (min) 18.6700 18.6700 PASS divergence (%/min) > 0 0.0207 PASS (sign > 0) SSE (mmHg) 3.0000 3.0000 PASS ----------	========== Metric Expected Computed Status ---------- MDPE (%) 4.2857 4.2857 PASS MDAPE (%) 4.2857 4.2857 PASS Wobble (%) 0.0000 0.0000 PASS per10 (%) 80.6366 80.6366 PASS per20 (%) 96.4339 96.4339 PASS per30 (%) 99.2501 99.2501 PASS risetime (min) 2.2200 2.2200 PASS overshoot (%) 15.7143 15.7143 PASS settlingtime (min) 18.6700 18.6700 PASS divergence (%/min) > 0 0.0207 PASS (sign > 0) SSE (mmHg) 3.0000 3.0000 PASS ----------
Example 3	





Metric	Expected	Computed	Status
MDPE (%)	4.2857	4.2857	PASS
MDAPE (%)	4.2857	4.2857	PASS
Wobble (%)	> 0	2.5542	PASS (sign > 0)
per10 (%)	80.3688	80.3688	PASS
per20 (%)	97.6558	97.6558	PASS
per30 (%)	99.2556	99.2556	PASS
risetime (min)	2.1300	2.1300	PASS
overshoot (%)	20.0000	20.0000	PASS
settlingtime (min)	28.2500	28.2500	PASS
divergence (%/min)	< 0	−0.0319	PASS (sign < 0)
SSE (mmHg)	3.0000	3.0000	PASS

Metric	Expected	Computed	Status
MDPE (%)	4.2857	4.2857	PASS
MDAPE (%)	4.2857	4.2857	PASS
Wobble (%)	> 0	2.5542	PASS (sign > 0)
per10 (%)	80.3688	80.3688	PASS
per20 (%)	97.6558	97.6558	PASS
per30 (%)	99.2556	99.2556	PASS
risetime (min)	2.1300	2.1300	PASS
overshoot (%)	20.0000	20.0000	PASS
settlingtime (min)	28.2500	28.2500	PASS
divergence (%/min)	< 0	−0.0319	PASS (sign < 0)
SSE (mmHg)	3.0000	3.0000	PASS

References

[1] Chalumuri et al. *Annals of Biomedical Engineering*, 54, pp. 857–868 (2026). https://doi.org/10.1007/s10439-025-03929-2

[2] Varvel et al. *Journal of Pharmacokinetics and Biopharmaceutics*, 20(1), 63–94 (1992). https://doi.org/10.1007/bf01143186