SynTwin: Host-aware digital twin framework for quantitative and context-aware biopart characterization
–User Manual–

Jesús Picó[a], Andrés Arboleda-García[a], David R. Penas[b], Julio R. Banga[b], Alejandro Vignoni[a], Yadira Boada[a]

[a]Synthetic Biology and Biosystems Control Lab, Institut ai2, Universitat Politècnica de València, Camí de Vera s/n, València, 46022, Spain
[b]Computational Biology Lab, MBG-CSIC, Pazo de Salcedo, Carballeira 8, Pontevedra, 36143, Spain

Abstract

This document describes SynTwin, the host-aware digital twin framework used in the study "**Host-aware Identification of Intrinsic Gene Expression Biopart Parameters using Combinatorial Libraries**". SynTwin is designed as a general framework for host-aware characterization and predictive modeling of genetic parts beyond the specific case study presented here. The present release corresponds to the implementation employed to generate all computational results reported in the manuscript and its Supplementary Information.

*Jesús Picó
Email addresses: jpico@upv.edu.es (Jesús Picó), maarbgar@upvnet.upv.es (Andrés Arboleda-García), david.rodriguez.penas@csic.es (David R. Penas), j.r.banga@csic.es (Julio R. Banga), alvig2@upv.es (Alejandro Vignoni), yaboa@upv.es (Yadira Boada)

Contents

S1. General Overview of SynTwin

SynTwin implements the host-aware, model-based methodology introduced in the associated manuscript for the quantitative characterization of genetic bioparts embedded within combinatorial libraries of transcriptional units (TUs). The framework integrates experimental data processing, host-aware mechanistic modeling, digital twin inference, and global parameter estimation into a coherent computational workflow.

The approach enables the estimation of biologically meaningful parameters—plasmid copy number, promoter transcription rate, and ribosome binding site (RBS) translational efficiency—expressed in standardized units with direct biophysical interpretation. Figure S1.1 summarizes the complete workflow implemented in SynTwin.

Combinatorial TU library and experimental characterization. Each TU combines an origin of replication, promoter, and RBS driving expression of *GFPmut3*. The combinatorial design ensures that every biopart appears embedded in multiple distinct genetic and physiological contexts (Fig. S1.1A), which is essential for resolving parameter lack of identifiability that arise when TUs are studied in isolation. For each construct, high-resolution measurements of optical density and GFP fluorescence are used to compute time-resolved specific growth rates and TU synthesis rates (Fig. S1.1B).

Host-aware modeling and digital twin. To capture the coupling between synthetic gene expression and cell physiology, we employ a host-aware modeling framework combining: (i) an endogenous Host Equivalent Model (HEM) describing resource allocation and growth, and (ii) a mechanistic TU synthesis model parameterized by biopart-specific quantities.

To improve predictive capacity and mitigate uncertainty inherent to purely mechanistic descriptions, we introduce a hybrid *E. coli* digital twin (Fig. S1.1C) that integrates real-time experimental growth-rate measurements. The specific growth rate provides an informative proxy for the cellular resource state—reflecting substrate availability, endogenous metabolism, and gene expression burden—and thus constrains the host–circuit interaction in a mechanistically consistent manner **??**. This model-in-the-loop strategy enables the digital twin to infer internal resource fluxes and deliver context-aware predictions of TU synthesis rates.

Context-independent parameterization of bioparts. The TU synthesis model explicitly separates: (a) global physiological variables (growth rate, resource flux, substrate availability), determined by the digital twin, from (b) intrinsic biopart parameters (copy number, promoter strength, RBS nominal strength and sensitivity).

While the host state modulates translation capacity, the biopart-specific parameters remain context-independent. This separation enables a quantitative mapping from DNA sequences to biophysical parameters and from these to circuit-level phenotypes.

Parameter estimation via global optimization. Predicted synthesis rates from the digital twin are matched to experimental TU data through a global heuristic optimization procedure (Fig. S1.1D). This allows simultaneous estimation of all biopart parameters using the full combinatorial dataset and propagates uncertainty naturally through Monte Carlo sampling of the solution landscape. The resulting parameter distributions (Fig. S1.1E) are then used to compute confidence intervals for TU synthesis-rate predictions over the full range of growth conditions.

Role of the combinatorial library. The combinatorial structure of the TU library is essential for resolving structural and practical identifiability limitations inherent to individual constructs. Because each promoter, RBS, and origin appears in multiple distinct contexts, the global sensitivity (Jacobian) matrix becomes structured and effectively full-rank. This enables reliable, context-independent characterization of all bioparts and supports incremental expansion of the library by adding new parts whose parameters can then be inferred using previously characterized bioparts as anchors.

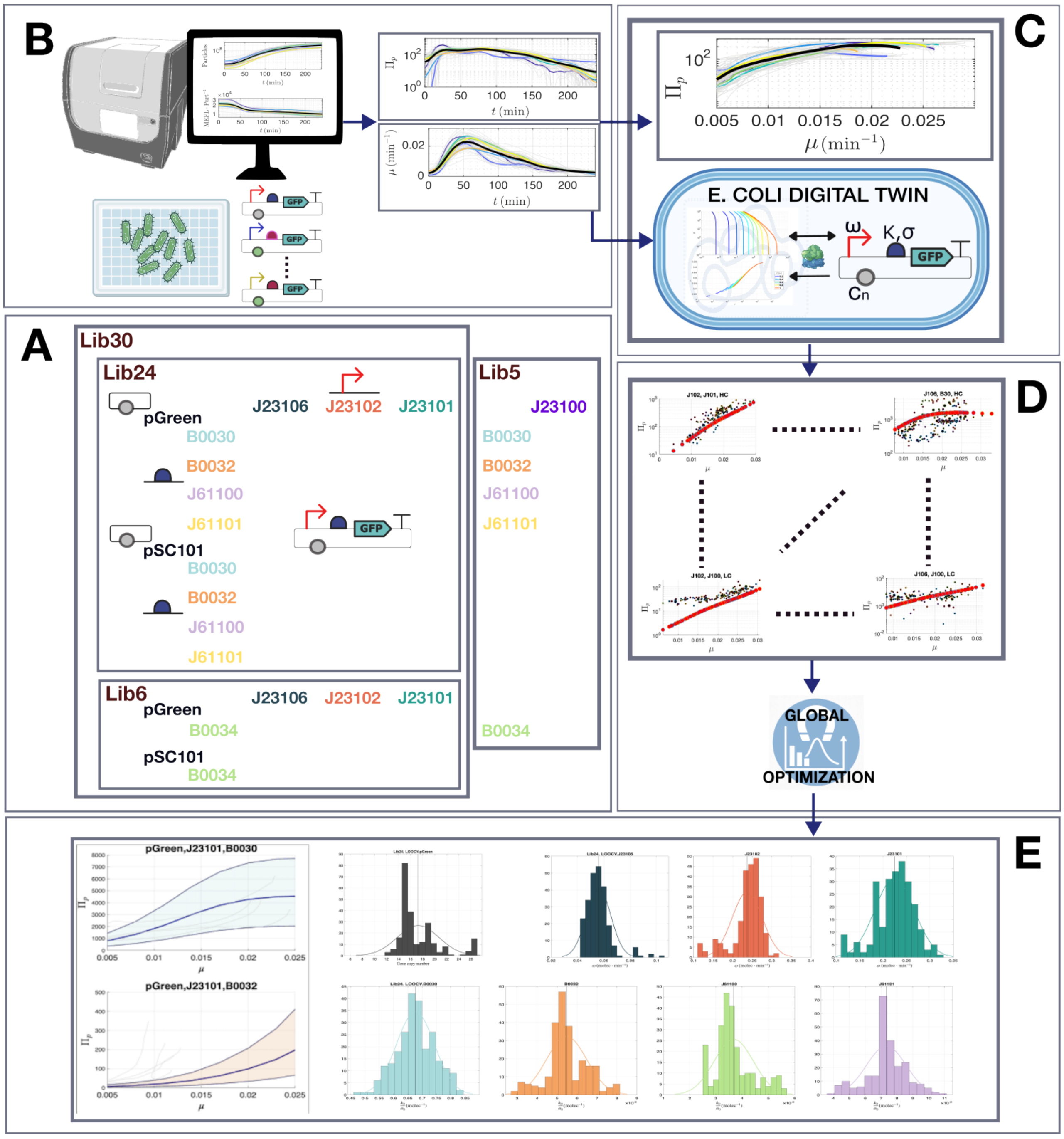

Figure S1.1: Overview of the host-aware biopart characterization workflow. (A) Combinatorial library of constitutive transcriptional units (TUs) constructed from multiple plasmid origins, promoters, and ribosome binding sites (RBSs), ensuring that each biopart appears across diverse genetic contexts. (B) High-resolution experimental characterization: time-series of optical density and GFP fluorescence are processed to obtain specific growth rate $\mu(t)$ and TU synthesis rate $\Pi(t)$ for every construct. (C) Host-aware *E. coli* digital twin integrating a mechanistic resource-allocation model with real-time experimental growth-rate measurements. The digital twin provides context-dependent estimates of cellular resource fluxes and predictions of growth-coupled TU synthesis dynamics. (D) Global optimization loop combining digital-twin predictions with experimental synthesis-rate data to estimate biologically interpretable biopart parameters (copy number, promoter transcription rate, and RBS translation efficiency). (E) Posterior parameter distributions and TU synthesis-rate fits across the library. These distributions quantify uncertainty and support context-independent, mechanistically grounded characterization of genetic bioparts.

S2. Software Arquitecture and Directory Structure

The SynTwin root directory is organized as follows:

```
SynTwin/
    Scripts_base/
    Generate_HEM/
        HEM_Surrogate/
        Third_party_data/
        Estimate_WT/
        Nut_estimate/
        Mass_estimate/
    Experimental_Data/
    Matlab/
        MEIGO/
        bads-master/
    Estimation_Pi/
        Lib30_L10_full_model/
        Lib30_ALL_full_model/
        Lib30_L10_reduced_model/
        Lib30_ALL_reduced_model/
        Lib24_ALL_full_model/
        Lib24_L10_reduced_model/
        Lib24_ALL_reduced_model/
        Lib6_L10_reduced_model/
        Lib6_ALL_reduced_model/
        Lib5_L10_reduced_model/
        Lib5_ALL_reduced_model/
    Jacobian_analysis/
```

Scripts_base/ Core functions for data preprocessing, synthesis-rate computation, and model utilities.

Generate_HEM/ Implementation of the Host Equivalent Model (HEM) and generation of surrogate models used by the digital twin.

Experimental_Data/ Processed experimental datasets used in the manuscript. Raw measurements are provided in structured MATLAB format for reproducibility.

Matlab/ Contains third-party optimization solvers (MEIGO and BADS).

Estimation_Pi/ Parameter estimation workflows for specific libraries. Each subfolder contains scripts for estimation, result generation, visualization, and stored parameter distributions.

Jacobian_analysis/ Assessmet of practical identifiability of SynTwin library parameters by computing the rank of the experimental Jacobian (sensitivity) matrix.

S2.1. Data availability, third-party data attribution, and third-party software

SynTwin is distributed together with the processed experimental datasets required to reproduce the computational results reported in the manuscript and this Software Supplementary. In addition, SynTwin includes third-party datasets that were used to fit and validate the *Host Equivalent Model (HEM)* component. SynTwin does not claim ownership of any third-party datasets; any reuse should comply with the original license/terms of the upstream sources.

Third-party datasets used for HEM fitting (Excel copies of upstream CSV files). For reproducibility and ease of inspection, SynTwin ships Excel (`.xlsx`) copies converted from two datasets originally distributed in the *Flux Parity* repository by (**?**): (i) collated *E. coli* ribosomal mass fractions and (ii) collated *E. coli* peptide elongation (translation) rates. These files are verbatim re-encodings of the upstream CSV content (no reanalysis is intended by the file-format change). The original data are publicly available from the authors via the Flux Parity GitHub repository and its Zenodo archive (DOI: `10.5281/zenodo.5893799`). When using SynTwin-derived results that depend on these datasets, users should cite the corresponding (**?**) work and/or the Zenodo record as required by their venue.

Additional upstream sources. The (**?**) collated datasets incorporate measurements from earlier literature. In particular, some of the compiled values (and, in some cases, additional values used by SynTwin scripts for host fitting) originate from: (**?**) *Modulation of chemical composition and other parameters of the cell at different exponential growth rates*. EcoSal Plus, 3, DOI: `10.1128/ecosal.5.2.3`. Users reusing values that originate from this source (directly or via the (**?**) compilation) are encouraged to cite (**?**) in addition to the (**?**) dataset source.

Where to find the files in the repository. These Excel copies are stored under folder `Generate_HEM/`. Search for filenames containing `ecoli_ribosomal_mass_fractions` and `ecoli_peptide_elongation_rates`.

Third-party software. SynTwin interfaces with and/or redistributes third-party optimization software used for parameter estimation: **MEIGO (eSS)** (Enhanced Scatter Search metaheuristic) and **BADS** (Bayesian Adaptive Direct Search). These packages retain their original authorship and licenses; users are responsible for complying with the corresponding license terms when using or redistributing these tools. When reporting results obtained with SynTwin that rely on these optimizers, users are encouraged to cite the original MEIGO and/or BADS publications in addition to citing the SynTwin paper.

Repository-level statement. A repository-level `Data Availability & Attribution Statement` is provided in the SynTwin root directory with the recommended attribution details and file pointers.

S2.2. Portable path initialization (MATLAB paths)

All SynTwin scripts are intended to run without hard-coded absolute paths. Before running any script, the SynTwin project path must be initialized once per MATLAB session by calling: `init_SynTwin.m` (located at the repository root). This function adds the core SynTwin folders (e.g., `Scripts_base/` and `Generate_HEM/HEM_Surrogate/`) to the MATLAB search path.

Running scripts from any working directory. If a script located in a subfolder is launched while MATLAB's current folder is the SynTwin root (or any other directory), MATLAB may not automatically include the folder containing that script. To ensure portability, each runnable script should add its own folder to the path using: `mfilename('fullpath')`. A minimal template is:

```
% --- Portable project initialization (no absolute paths) ---
ROOT = init_SynTwin();  % Adds Scripts_base + HEM_Surrogate by default

% Absolute path to the folder where this script lives (model folder)
SCRIPT_DIR = fileparts(mfilename('fullpath'));
addpath(SCRIPT_DIR);

% Example: load results stored next to this script
load(fullfile(SCRIPT_DIR,'Results_Tensor_Lib30_ALL_full_model_Instances.mat'));
```

This strategy guarantees that scripts can be executed either from their own folder or from the SynTwin root, while keeping all file operations portable via `fullfile()`.

Project-relative paths. Additionally, SynTwin provides the helper `SynTwin_path()` to construct absolute paths inside the repository from relative components, e.g. `SynTwin_path('Experimental_Data','file.mat')`.

S3. Optimization algorithms

SynTwin relies on global optimization routines for parameter estimation. In the present release, two alternative solvers are provided:

- **MEIGO (eSS) ?**, developed by J.R. Banga and collaborators, implementing the Enhanced Scatter Search (eSS) metaheuristic.
- **BADS (Bayesian Adaptive Direct Search) ?**, a hybrid Bayesian optimization framework.

Both solvers are included within the SynTwin distribution for user convenience. They are third-party software packages and retain their original licenses and authorship. SynTwin interfaces with these solvers through modular objective-function wrappers specific to the host-aware digital twin model.

In the results reported in the associated manuscript, both eSS and BADS were used interchangeably for the sake of comparison and assessment of estimation robustness.

S4. Parameter estimation, Results Tensor Construction, and Visualization Workflow

SynTwin parameters are inferred by minimizing the mismatch between experimentally estimated synthesis-rate trajectories and digital-twin predictions. All estimations are performed using the Bayesian Adaptive Direct Search (BADS) optimizer under deterministic model evaluations.

Unless otherwise specified, the objective function is defined in base-10 logarithmic space with magnitude-dependent weighting:

$$w_{\log} = \left|\log_{10}(P_i^{\text{pred}}) \cdot \left(\log_{10}(P_i^{\text{pred}}) - \log_{10}(P_i^{\text{exp}})\right)\right|, \tag{1}$$

where P_i^{pred} and P_i^{exp} denote predicted and experimental synthesis rates, respectively. The scalar cost is obtained by aggregating this weighted error across time points and transcriptional units (TUs).

All workflows are implemented in modular MATLAB scripts and operate on structured tensor containers storing experimental observables, model parameters, and Monte Carlo samples.

S4.1. Experimental Data

S4.1.1. Minimun set of experimental data

The minimum experimental data required by SynTwin consists of the experimental values of specific growth rate $\mu(t)$ and synthesis rate $\Pi(t)$ evaluated, for each TU in the library, between the time at which $\mu_{\max}$ is reached, and the time at which the maximum particle number is attained (see Supplementary Information of the companion paper for more details).Therefore, the vector lengths equal the number of sampled time points within this post-$\mu_{\max}$ regime, which defines the host-aware characterization window, obtained at each data aggregation level.

The data structure consists of a three-dimensional MATLAB cell array of size (num_plasmids $\times$ num_promoters $\times$ num_rbss). Each tensor entry (cell) corresponds to a single transcriptional unit (TU), defined by a unique combination of origin (plasmid), promoter, and RBS. Three data aggregation levels may be considered:

`Global`	Averaged aggregated values over all the experimental data
`Instances`	(<NumInstances>$\times 1$) cell array: averaged aggregated values per experiment (*Instance*).
`Wells`	(<NumWells>$\times 1$) cell array: raw well-level data. By default <NumWells>=10

The files `ExpData_Tensor_<Library>_micro.mat` provided with SynTwin contain the minimum required processed experimental data used as input for the host-aware parameter estimation framework for libraries <Library>=Lib24, Lib30, and Lib5: essential Transcriptional Unit metadata, and growth and synthesis data at three data aggregation levels. A detailed description of the data structure in given in Appendix A.

S4.1.2. Complete set of experimental data

In addition to the minimun set data structures, the files `ExpData_Tensor_<Library>.mat` contain an extended version with the complete processed experimental data obtained as general input for the host-aware parameter estimation framework. In contrast to the `_micro` version, this extended structure includes:

- Transcriptional Unit Metadata,
- Global statistics of synthesis rate and load at the maximum specific growth rate,
- Fold-level data and statistics of Particles, MEFL, ratio Particles/MEFL, specific growth rate,synthesis rate, cell load, and
- Experiments Metadata.

A detailed description of the complete experimental data structure in given in Appendix B.

S4.2. Overview of the provided libraries

SynTwin includes several combinatorial libraries that play distinct roles within a hierarchical inference strategy. Rather than representing independent case studies, these libraries form a structured identification pipeline composed of:

- **Core libraries** (L24, L30): used to establish foundational, context-independent parameter distributions.
- **Patchwork (incremental) libraries** (L6, L5): used to extend the characterization to new bioparts by inheriting previously inferred parameters and estimating only the remaining degrees of freedom.

L24 (core, reduced model). Primary reference library used to infer intrinsic RBS initiation capacities under a reduced translation formulation. Provides the baseline parameter distributions inherited by downstream analyses.

L30 (general analysis). Expanded combinatorial library spanning a broader TU space. Used to evaluate model generalization and consistency across contexts.

L6 (patchwork extension). Six-construct sublibrary designed to estimate a shared RBS parameter while inheriting promoter-, origin-, and copy-number distributions from L24.

L5 (patchwork extension). Five-construct sublibrary isolating a single promoter across multiple RBS variants within a fixed origin context. Only promoter strength is estimated, while RBS and copy-number parameters are inherited from L24 and L6.

The detailed estimation workflows, inheritance logic, cross-validation schemes, and uncertainty propagation mechanisms are described in Section S4.3.

S4.3. Parameter estimation Workflow

S4.3.1. Estimation framework and objective function

Parameter inference in SynTwin is performed by minimizing the mismatch between experimentally estimated synthesis-rate trajectories and digital-twin predictions.

All optimizations are carried out using the Bayesian Adaptive Direct Search (BADS) algorithm under deterministic model evaluations. Unless otherwise specified, inference is conducted in base-10 logarithmic space with magnitude-dependent weighting of prediction errors. For predicted and experimental synthesis rates P_i^{pred} and P_i^{exp}, respectively, the weighted mismatch at each time point is defined as:

$$w_{\log} = \left|\log_{10}(P_i^{\text{pred}}) \left(\log_{10}(P_i^{\text{pred}}) - \log_{10}(P_i^{\text{exp}})\right)\right|. \tag{2}$$

The scalar objective function is obtained by aggregating this quantity across time points and transcriptional units (TUs).

All workflows are implemented as modular MATLAB scripts operating on structured tensor containers that store experimental observables, model parameters, and Monte Carlo samples.

The estimation framework differs between:

- **Core libraries**, used to establish foundational parameter distributions under reduced or full model formulations.
- **Patchwork (incremental) libraries**, which inherit previously inferred parameters and estimate only a restricted subset of remaining unknowns.

S4.3.2. Identifiability-driven hierarchical design

The parameter-estimation workflows implemented in SynTwin are compatible with identifiability-based design principles for incremental and patchwork expansion of biopart libraries.

Rather than prescribing a fixed sequence of library constructions, the framework allows users to design new combinatorial libraries guided by structural and practical identifiability considerations. In particular, core libraries can be constructed to isolate primary parameters under well-conditioned contexts, while subsequent incremental libraries may selectively inherit previously identified parameters and estimate only the remaining degrees of freedom.

The theoretical design rules and identifiability criteria underlying this hierarchical expansion strategy are detailed in the main paper. The present software documentation focuses on the computational implementation of such workflows.

S4.3.3. Core libraries (L24 and L30)

The core inference stage is performed on the combinatorial libraries L24 and L30. In practice, L24 constitutes the primary reference library for parameter identification under the reduced-model formulation, while L30 was used for broader general analysis and validation.

Reduced-model formulation. For the core library L24, parameter inference is conducted under a reduced RBS model in which:

- The intrinsic initiation capacity $\kappa^0 = \frac{k^0}{\sigma^0}$ (parameter `k0_sigma0`) of each ribosome binding site (RBS) is estimated.
- The sensitivity-related parameter $\rho^0 = \frac{1}{\sigma^0}$ (`inv_sigma0`) is fixed to a predefined value.

This formulation isolates the dominant translational strength parameter while maintaining numerical stability and identifiability.

Cross-validation. Leave-one-out (L1O) cross-validation is used to assess robustness. For each left-out construct, multiple independent optimization runs are performed. The resulting parameter samples are aggregated to compute:

- Raw parameter samples,
- Mean and standard deviation statistics,
- Monte Carlo samples for downstream uncertainty propagation.

The output of the L24 workflow constitutes the foundational parameter tensor that is subsequently inherited by incremental libraries.

Lib30. The L30 library follows the same objective-function definition and tensor-based workflow structure but spans a larger combinatorial space. It is primarily used to evaluate model generalization across a broader set of transcriptional-unit combinations and to verify consistency of the reduced-model assumptions.

S4.3.4. Patchwork libraries: incremental parameter inheritance (L6 and L5)

Libraries L6 and L5 implement an incremental ("patchwork") inference strategy in which previously inferred parameters are inherited and only a restricted subset of parameters is estimated.

This hierarchical design allows uncertainty propagation across library layers while avoiding redundant re-estimation of already identified parameters.

Lib6. Lib6 isolates a shared RBS across multiple promoter and plasmid-origin contexts. In this workflow:

- A single shared RBS intrinsic initiation capacity `k0_sigma0` is estimated.
- Promoter- and origin-dependent parameters are inherited from L24.
- Effective gene copy number distributions are inherited from L24.

Uncertainty propagation is implemented by pairing each optimization run with a consistent Monte Carlo sample of inherited parameters. For each run index, the same inherited sample is used across all TUs within the library, ensuring coherent uncertainty transmission.

Lib5. Lib5 isolates a single promoter (e.g., J23100) across five RBS variants within a fixed plasmid-origin context. In this case:

- Only the promoter strength parameter ω^0 is estimated.
- RBS intrinsic initiation capacities are inherited:
  - From L24 for four RBS variants,
  - From L6 for the B0034 RBS.
- Effective gene copy number distributions are inherited from L6.

Leave-one-out cross-validation across RBS contexts is used to assess robustness and transferability of promoter inference.

Hierarchical uncertainty propagation. In both L6 and L5, Monte Carlo sampling of inherited parameters is explicitly propagated into the estimation process. Each optimization run is associated with a specific inherited Monte Carlo sample, thereby preserving correlations between upstream and downstream inference layers. Posterior predictive distributions are generated from these propagated samples.

S4.3.5. Executable workflow and script structure

All estimation workflows follow a standardized directory and script structure under:

```
Estimation_Pi/<Library>_<EstimationScheme>_<ModelVariant>_model/
```

where:

`<Library>`	{Lib24, Lib30, Lib6, Lib5, ...}
`<EstimationScheme>`	{ALL, L1O}
`<ModelVariant>`	{full, reduced}

Each folder contains a complete, self-contained workflow composed of four main scripts:

1. **Estimation script** `Estimate_<Library>_<EstimationScheme>_<ModelVariant>_model.m`

   Executes the optimization loop (BADS or MEIGO), loads the required experimental tensors and inherited parameter structures (if any), and stores raw optimization outputs in the subfolder `Estimated_results/`.

2. **Objective function** `J4_LogPI_<Library>_<EstimationScheme>_<ModelVariant>.m`

   Implements the scalar objective function. This function:

   - Receives the free parameter vector,
   - Calls the digital twin prediction routines,
   - Compares predicted and experimental Π values,
   - Returns the aggregated logarithmic mismatch.

3. **Results tensor construction** `Generate_Results_<Library>_<EstimationScheme>_<ModelVariant>_model.m`

   Processes the raw optimization outputs and constructs `Results_Tensor_<Library>_<EstimationScheme>_<ModelVarian`
   a structured object including:

   - Local parameter estimates,
   - Aggregated statistics,
   - Deterministic digital twin predictions,
   - Monte Carlo uncertainty propagation.

A complete description of `Results_Tensor_<Library>_<EstimationScheme>_<ModelVariant>` is given in Appendix C.

4. **Visualization script** `Show_Results_<Library>_<EstimationScheme>_<ModelVariant>_model.m`

   Generates the plots and summary visualizations used in the manuscript and Supplementary Information.

Execution sequence.

A complete estimation workflow is executed in three steps:

1. Run `Estimate_...` to perform optimization.
2. Run `Generate_Results_...` to construct the results tensor.
3. Run `Show_Results_...` for visualization and validation.

Data dependencies.

All workflows rely on:

- Experimental tensors in `Experimental_Data/ExpData_Tensor_<Library>_micro.mat`
- The host digital twin surrogate in `Generate_HEM/HEM_Surrogate/HEM_Surrogate.mat`
- Previously inferred parameter tensors (for patchwork libraries) located in their corresponding folders under `Estimation_Pi/`

Inheritance logic in patchwork libraries.

For incremental libraries (L6, L5), the estimation script explicitly loads previously generated results tensors, extracts Monte Carlo samples of inherited parameters, and associates each optimization run with a consistent inherited sample index. This guarantees hierarchical uncertainty propagation while estimating only the locally free parameters.

This standardized structure ensures that all libraries — regardless of size or inference scheme — share an identical execution logic, differing only in: (i) the number of free parameters, (ii) the inheritance structure, and (iii) the experimental tensor dimensions.

S5. Practical identifiability analysis

To evaluate the practical identifiability of inferred parameters, we compute the Jacobian of model predictions with respect to the estimated parameter vector at the optimum.

For a parameter vector θ and synthesis-rate predictions $P_i(\mu;\theta)$ evaluated at experimentally observed growth rates μ, the sensitivity matrix is defined as:

$$J_{ij} = \frac{\partial P_i}{\partial \theta_j}. \tag{3}$$

The rank and singular value spectrum of J are used to assess local parameter identifiability.

S5.1. Rank analysis

If $\mathrm{rank}(J) < \dim(\theta)$, the parameter set is locally non-identifiable under the given data. Full column rank indicates local structural identifiability.

S5.2. Conditioning and practical identifiability

Even when full rank is achieved, poor conditioning (large singular value spread) indicates practical non-identifiability. We therefore compute:

- Singular value spectra,
- Numerical rank under tolerance thresholds,
- Condition numbers.

These diagnostics are implemented in the `Jacobian_analysis` module of SynTwin.

S5.3. Interpretation in incremental libraries

For incremental libraries (Lib5 and Lib6), Jacobian analysis is performed both:

- On locally inferred parameters (L1O folds),
- On pooled parameter distributions.

This allows quantifying whether parameter inheritance introduces identifiability constraints or redundancy across contexts.

Appendix A. Structure of the Experimental Data Tensor (micro version)

Generality Across Libraries

The files `ExpData_Tensor_<Library>_micro.mat` contain the minimum required processed experimental data used as input for the host-aware parameter estimation framework.
The structures:

- `ExpData_Tensor_Lib24_micro`
- `ExpData_Tensor_Lib30_micro`
- `ExpData_Tensor_Lib5_micro`

share an identical internal schema. Only the tensor dimensions differ, reflecting the number of TUs in each combinatorial library. This uniform data model enables direct reuse of estimation and digital twin routines across libraries without structural modification.

Tensor Structure

The data structure consists of a three-dimensional MATLAB cell array of size (num_plasmids $\times$ num_promoters $\times$ num_rbss). Each tensor entry (cell) corresponds to a single transcriptional unit (TU), defined by a unique combination of origin (plasmid), promoter, and RBS.

Three data aggregation levels are considered:

`Global`	Averaged aggregated values over all the experimental data
`Instances`	(<NumInstances>$\times$1 cell array): averaged aggregated values per experiment (*Instance*).
`Wells`	(<NumWells>$\times$1 cell array) raw well-level data. By default <NumWells>=10

The data vectors at each aggregation level store the specific growth rate $\mu(t)$ and TU synthesis rate $\Pi(t)$ evaluated between the time at which $\mu_{\max}$ is reached, and the time at which the maximum particle number is attained (see Supplementary Information for more details).Therefore, the vector lengths equal the number of sampled time points within this post-$\mu_{\max}$ regime, which defines the host-aware characterization window, obtained at each data aggregation level.

Global hierarchy

Each tensor cell is a MATLAB struct containing:

`TU_Ori`	Plasmid/origin identifier (e.g., pGreen).
`TU_Promoter`	Promoter identifier (e.g., J23106).
`TU_RBS`	RBS identifier (e.g., B0030).
`TU_Bioparts`	Comma-separated string encoding the triplet (Ori, Promoter, RBS).
`TU_Name`	Internal construct label (lab dependent)
`TU_color_code`	RGB visualization code
`Mu_mumax_pmax_global_mean`	Time-resolved global mean value of growth rate evaluated from $\mu_{\max}$ to $P_{\max}$ averaged across all Instances.
`Mu_mumax_pmax_global_std`	Standard deviation associated to Mu_mumax_pmax_global_mean
`Pi_mumax_pmax_global_mean`	Time-resolved global mean value of synthesis rate evaluated from $\mu_{\max}$ to $P_{\max}$ averaged across all Instances.
`Pi_mumax_pmax_global_std`	Standard deviation associated to Pi_mumax_pmax_global_mean
`Instances`	<NumInstances>$\times$1 cell struct containing all experimental observables

`Mu_mumax_pmax_instance_mean`	Time-resolved mean value of growth rate evaluated from $\mu_{\max}$ to $P_{\max}$ averaged across the <NumWells> wells of the Instance.
`Mu_mumax_pmax_instance_std`	Standard deviation associated to Mu_mumax_pmax_instance_mean
`Pi_mumax_pmax_instance_mean`	Time-resolved mean value of synthesis rate evaluated from $\mu_{\max}$ to $P_{\max}$ averaged across the <NumWells> wells of the Instance.
`Pi_mumax_pmax_instance_std`	Standard deviation associated to Pi_mumax_pmax_instance_mean
`Wells`	<NumWells>$\times$1 cell struct containing Well experimental observables

`Mu_mumax_pmax`	Time-resolved growth rate evaluated from $\mu_{\max}$ to $P_{\max}$ for the Well

	`Pi_mumax_pmax`	Time-resolved synthesis rate evaluated from $\mu_{\max}$ to $P_{\max}$ for the Well

Appendix B. Structure of the Experimental Data Tensor (complete version)

The files `ExpData_Tensor_<Library>.mat` contain the complete processed experimental data obtained as general input for the host-aware parameter estimation framework. The same internal organization applies to all libraries; only the tensor dimensions change according to the number of transcriptional units (TUs) in each library. The tensor is a 3D cell array:(`num_plasmids` $\times$ `num_promoters` $\times$ `num_rbss`). Each cell contains a struct with the following hierarchy:

1. Global hierarchy

`TU_Ori`	Origin of replication
`TU_Promoter`	Promoter identifier
`TU_RBS`	RBS identifier
`TU_Bioparts`	Concatenated construct string
`TU_Name`	Internal construct label (lab dependent)
`TU_color_code`	RGB visualization code
`Data`	1×1 struct containing all experimental observables

`Construct_name`	Internal construct label (lab dependent, e.g. the same as TU_Name)
`Global_stats_Mumax_Pi_p`	2×2 matrix containing the global averaged maximum specific growth rate $\mu_{\max}$, the synthesis rate $\Pi_{\max}(\mu_{\max})$ and their standard deviations
`Global_stats_Mumax_Phi_h`	2×2 matrix containing the global averaged maximum specific growth rate $\mu_{\max}$, the host cell burden $\Phi_{,h\max}(\mu_{\max})\%$ and their standard deviations (see Supplementary Information S3.4)
`Global_stats_Mumax_Phi_s`	2×2 matrix containing the global averaged maximum specific growth rate $\mu_{\max}$, the strain cell burden $\Phi_{s,\max}(\mu_{\max})\%$ and their standard deviations (see Supplementary Information S3.4)
`Stats_Particles`	1×1 struct containing the cell count measurements as *Particles* (see Supplementary Information S2.2) and additional experimental data and metadata (see point 2 below)
`Stats_MEFL`	1×1 struct containing fold-level fluorescence measurements as *MEFL* (see Supplementary Information S2.2 and point 3 below)
`Stats_MEFL_Particles`	1×1 struct containing fold-level derived values of *MEFL/Particles* (see Supplementary Information S2.2 and point 4 below)
`Stats_Mu`	1×1 struct containing fold-level derived values of the specific growth rate μ (see Supplementary Information S2.3) and related singular values and statistics (see point 5 below)
`Stats_Pi_p`	1×1 struct containing fold-level derived values of the background-compensated synthesis rate Π (see Supplementary Information S2.3) and related statistics (see point 6 below)
`Stats_Phi_s`	1×1 struct containing fold-level derived values of the strain cell burden and related statistics (see point 7 below)
`Stats_Phi_h`	1×1 struct containing fold-level derived values of the host cell burden and related statistics (see point 7 below)
`Global_data_time`	$n_T \times 1$ vector containing the global time sampling instants, with $n_T = \min_i(n_i)$ obtained as the minimum length of all the experimental instances of the TU.

2. Stats_Particles hierarchy Stats_Particles is a 1×1 struct containing the cell count measurements as *Particles* and additional experimental data and metadata, with fields:

`List_instances`	1×Num_instances cell containing single experiment (*Instance*) data and metadata. Each cell contains:

`Experiment_IDnum`	Experiment identification number (lab dependent)
`Experiment_date`	Experiment date
`Colony_number`	Colony number (if appropriate)

`Status_exp`	Experiment status. Set to 0 to indicate experiments not to be included in the evaluations (e.g. because of experimental issues)
`Data_time`	$n_t \times 1$ vector with the sampling time instants of the experiment
`Data_raw`	$n_t \times n_{\text{wells}}$ matrix with the *Particles* samples for each well
`Data_mean`	$n_t \times 1$ vector with the mean *Particles* samples averaged across wells
`Data_std`	$n_t \times 1$ vector with the standard deviation of *Particles* samples averaged across wells
`Data_Mumax_Pi_p`	$2 \times n_{\text{wells}}$ matrix containing the maximum specific growth rate $\mu_{\max}$ for each well (upper row) and the synthesis rates $\Pi_{\max}(\mu_{\max})$ evaluated at $\mu_{\max}$ (lower row).
`Data_Mumax_Phi_s`	$2 \times n_{\text{wells}}$ matrix containing the maximum specific growth rate $\mu_{\max}$ for each well (upper row) and the strain burden $\Phi_{s,\max}(\mu_{\max})$ evaluated at $\mu_{\max}$ (%, lower row).
`Data_Mumax_Phi_h`	$2 \times n_{\text{wells}}$ matrix containing the maximum specific growth rate $\mu_{\max}$ for each well (upper row) and the host burden $\Pi_{h,\max}(\mu_{\max})$ evaluated at $\mu_{\max}$ (%, lower row).
`Data_stats_Mumax_Pi_p`	2×2 matrix containing the $\mu_{\max}, \Pi_{\max}(\mu_{\max})$ mean values averaged across the n_{wells} wells (upper row) and their corresponding standard deviations (lower row).
`Data_stats_Mumax_Phi_s`	2×2 matrix containing the $\mu_{\max}, \Phi_{s,\max}(\mu_{\max})$ mean values averaged across the n_{wells} wells (upper row) and their corresponding standard deviations (lower row).
`Data_stats_Mumax_Phi_h`	2×2 matrix containing the $\mu_{\max}, \Phi_{h,\max}(\mu_{\max})$ mean values averaged across the n_{wells} wells (upper row) and their corresponding standard deviations (lower row).
`Value_Particlesmax_wells`	$1 \times n_{\text{wells}}$ vector containing the maximum value of Particles $P_{\max}$ attained in each well
`Index_Time_Particlesmax_wells`	$1 \times n_{\text{wells}}$ vector containing the time samples at which the maximum value of Particles is attained
`Value_Particlesmax_mean`	Mean value of $P_{\max}$ across the n_{wells} wells.
`Index_Time_Particlesmax_mean`	Average time sample at which the maximum value of Particles $P_{\max}$ is attained
`Data_mumax_pmax_mean`	$n_r \times 1$ vector with the time-resolved value of Particles, averaged across the n_{wells} wells of the Instance (experiment), evaluated at the n_r-length samples interval between μ_max and $P_{\max}$.
`Data_mumax_pmax_std`	$n_r \times 1$ vector with the standard deviations for Data_mumax_pmax_mean
`Data_Mumax_Pi_p_All`	$4 \times (n_{\text{wells}} * \text{Num\_instances})$ matrix containing the maximum specific growth rate $\mu_{\max}$ for each well (first row), synthesis rates $\Pi_{\max}(\mu_{\max})$ (second row), Experiment_IDnum (third row), and Colony_number (fourth row).
`Data_Mumax_Phi_s_All`	$4 \times (n_{\text{wells}} * \text{Num\_instances})$ matrix containing the maximum specific growth rate $\mu_{\max}$ for each well (first row), strain burden $\Phi_{s,\max}(\mu_{\max})$ (%, second row), Experiment_IDnum (third row), and Colony_number (fourth row).
`Data_Mumax_Phi_h_All`	$4 \times (n_{\text{wells}} * \text{Num\_instances})$ matrix containing the maximum specific growth rate $\mu_{\max}$ for each well (first row), host burden $\Phi_{h,\max}(\mu_{\max})$ (%, second row), Experiment_IDnum (third row), and Colony_number (fourth row).
`Global_data_mean`	Time-resolved mean value of Particles averaged across experiments
`Global_data_std`	Standard deviation associated to Global_data_mean
`Global_value_Particlesmax_mean`	Global mean value of $P_{\max}$ averaged across experiments.
`Global_index_Time_Particlesmax_mean`	Average time sample at which Global_value_Particlesmax_mean is attained
`Mumax_pmax_global_mean`	Time-resolved global mean value of Particles evaluated from $\mu_{\max}$ to $P_{\max}$ (Data_mumax_pmax_mean) averaged across all Instances (experiments).
`Mumax_pmax_global_std`	Standard deviation associated to Mumax_pmax_global_mean

3. Stats_MEFL hierarchy Stats_MEFL is a 1×1 struct containing the fluorescence measurements as *MEFL* and related statistics, with fields:

`List_instances` $1 \times$ Num_instances cell containing single experiment (*Instance*) data. Each cell contains:

`Data_raw`	$n_t \times n_{\text{wells}}$ matrix with the *MEFL* samples for each well
`Data_mean`	$n_t \times 1$ vector with the mean *MEFL* samples averaged across wells
`Data_std`	$n_t \times 1$ vector with the standard deviation of *MEFL* samples averaged across wells
`Data_mumax_pmax_mean`	$n_r \times 1$ vector with the time-resolved value of *MEFL*, averaged across the n_{wells} wells of the Instance (experiment), evaluated at the n_r-length samples interval between μ_max and $P_{\max}$.
`Data_mumax_pmax_std`	$n_r \times 1$ vector with the standard deviations for Data_mumax_pmax_mean

`Global_data_mean`	Time-resolved mean value of *MEFL* averaged across experiments
`Global_data_std`	Standard deviation associated to Global_data_mean
`Mumax_pmax_global_mean`	Time-resolved global mean value of *MEFL* evaluated from $\mu_{\max}$ to $P_{\max}$ (Data_mumax_pmax_mean) averaged across all Instances (experiments).
`Mumax_pmax_global_std`	Standard deviation associated to Mumax_pmax_global_mean

4. Stats_MEFL_Particles hierarchy Stats_MEFL_Particles is a 1×1 struct containing the cell specific fluorescence measurements as *MEFL/Particles* and related statistics, with fields:

`List_instances` $1 \times$ Num_instances cell containing single experiment (*Instance*) data. Each cell contains:

`Data_raw`	$n_t \times n_{\text{wells}}$ matrix with the *MEFL/Particles* samples for each well
`Data_mean`	$n_t \times 1$ vector with the mean *MEFL/Particles* samples averaged across wells
`Data_std`	$n_t \times 1$ vector with the standard deviation of *MEFL/Particles* samples averaged across wells

`Global_data_mean`	Time-resolved mean value of *MEFL/Particles* averaged across experiments
`Global_data_std`	Standard deviation associated to Global_data_mean

5. Stats_Mu hierarchy Stats_Mu is a 1×1 struct containing the specific growth rate data (μ) and related statistics, with fields:

`List_instances` $1 \times$ Num_instances cell containing single experiment (*Instance*) data. Each cell contains:

`Data_raw`	$n_t \times n_{\text{wells}}$ matrix with the growth rate samples for each well
`Data_mean`	$n_t \times 1$ vector with the mean growth rate samples averaged across wells
`Data_std`	$n_t \times 1$ vector with the standard deviation of growth rate samples evaluated across wells
`Values_Mumax_wells`	$1 \times n_{\text{wells}}$ vector containing the maximum value of growth rate $\mu_{\max}$ attained in each well
`Indices_Times_Mumax_wells`	$1 \times n_{\text{wells}}$ vector containing the time samples at which $\mu_{\max}$ is attained
`Value_Mumax_mean`	Mean value of $\mu_{\max}$ across the n_{wells} wells.
`Index_Time_Mumax_mean`	Average time sample at which $\mu_{\max}$ is attained
`Data_mumax_pmax_mean`	$n_r \times 1$ vector with the time-resolved value of growth rate μ, averaged across the n_{wells} wells of the Instance (experiment), evaluated at the n_r-length samples interval between μ_max and $P_{\max}$.
`Data_mumax_pmax_std`	$n_r \times 1$ vector with the standard deviations for Data_mumax_pmax_mean

`Global_data_mean`	Time-resolved mean value of growth rate averaged across experiments
`Global_data_std`	Standard deviation associated to Global_data_mean
`Mumax_pmax_global_mean`	Time-resolved global mean value of Data_mumax_pmax_mean (growth rate evaluated from $\mu_{\max}$ to $P_{\max}$) averaged across all Instances (experiments).
`Mumax_pmax_global_std`	Standard deviation associated to Mumax_pmax_global_mean
`Global_value_Mumax_mean`	Global mean value of $\mu_{\max}$ averaged across experiments.

`Global_index_Time_Mumax_mean`	Average time sample at which Global_value_Mumax_mean is attained

6. Stats_Pi hierarchy Stats_Pi is a 1×1 struct containing the synthesis rates Π and related statistics, with fields:

`List_instances` 1×Num_instances cell containing single experiment (*Instance*) data. Each cell contains:

`Data_raw`	$n_t \times n_{\text{wells}}$ matrix with the time-resolved evaluation of Π for each well
`Data_mean`	$n_t \times 1$ vector with the mean Π samples averaged across wells
`Data_std`	$n_t \times 1$ vector with the standard deviation of Π samples averaged across wells
`Data_mumax_pmax_mean`	$n_r \times 1$ vector with the time-resolved value of Π, averaged across the n_{wells} wells of the Instance (experiment), evaluated at the n_r-length samples interval between μ_max and $P_{\max}$.
`Data_mumax_pmax_std`	$n_r \times 1$ vector with the standard deviations for Data_mumax_pmax_mean

`Global_data_mean`	Time-resolved mean value of Π averaged across experiments
`Global_data_std`	Standard deviation associated to Global_data_mean
`Mumax_pmax_global_mean`	Time-resolved global mean value of Π evaluated from $\mu_{\max}$ to $P_{\max}$ (Data_mumax_pmax_r averaged across all Instances (experiments).
`Mumax_pmax_global_std`	Standard deviation associated to Mumax_pmax_global_mean

7. Stats_Phi_s hierarchy Stats_Phi_s is a 1×1 struct containing the strain burden estimation $\Phi_s(\%)$ and related statistics, with fields:

`List_instances` 1×Num_instances cell containing single experiment (*Instance*) data. Each cell contains:

`Data_raw`	$n_t \times n_{\text{wells}}$ matrix with the time-resolved Φ_s samples for each well
`Data_mean`	$n_t \times 1$ vector with the mean Φ_s samples averaged across wells
`Data_std`	$n_t \times 1$ vector with the standard deviation of Φ_s samples averaged across wells

`Global_data_mean`	Time-resolved mean value of Φ_s averaged across experiments
`Global_data_std`	Standard deviation associated to Global_data_mean

strut

8. Stats_Phi_h hierarchy Stats_Phi_h is a 1×1 struct containing the host burden estimation $\Phi_s(\%)$ and related statistics, with fields:

`List_instances` 1×Num_instances cell containing single experiment (*Instance*) data. Each cell contains:

`Data_raw`	$n_t \times n_{\text{wells}}$ matrix with the time-resolved Φ_h samples for each well
`Data_mean`	$n_t \times 1$ vector with the mean Φ_h samples averaged across wells
`Data_std`	$n_t \times 1$ vector with the standard deviation of Φ_h samples averaged across wells

`Global_data_mean`	Time-resolved mean value of Φ_h averaged across experiments
`Global_data_std`	Standard deviation associated to Global_data_mean

Appendix C. Structure of the Results Tensor Data Type

The data files *Results_Tensor_<Library>_<EstimationScheme>_<ModelVariant>_<DataAggregation>.m* contain the data structure `Results_Tensor_<Library>_<EstimationScheme>_<ModelVariant>` encoding the complete prediction results of the host-aware characterization framework, together with all associated metadata, parameter-estimation summaries, deterministic digital twin simulations, sensitivity analyses, and Monte Carlo (MC) uncertainty propagation outputs.

This naming convention explicitly specifies the experimental library, the parameter-estimation scheme, the model class, and the data aggregation level used during estimation. Notice the data aggregation level <DataAggregation> appears in the name of the Matlab *.mat* file, but not in that of the data structure. For this one, the data aggregation level is included as a field (see point 4 below).

The components of the identifier are defined as follows:

Library Identifier. *<Library>* = {Lib24, Lib30, ...}
Specifies the combinatorial library used for parameter estimation. The number denotes the total number of transcriptional units (TUs) included in the library.

Estimation Scheme. *<EstimationScheme>* = {L1O, ALL}

`L1O`	Parameters are estimated using cross-validation (Leave-One-Out), where one TU is excluded per fold.
`ALL`	Parameters are estimated using all TUs simultaneously, without cross-validation.

Model Variant. *<ModelVariant>* = {complete, reduced}

`complete`	Full translation model.
`reduced`	Reduced translation model with fixed translational parameter $1/\sigma^0$.

Data Aggregation Level. *<DataAggregation>* = {Global, Instances, Wells}
Indicates the level at which experimental data were aggregated for parameter estimation:

`Global`	Data aggregated across all experiments and wells.
`Instances`	Data aggregated per experimental instance.
`Wells`	Well-resolved experimental data used directly.

Number of Parameters. The number of TUs in the dataset equals the size of the estimated library (<Library>). In addition, the maximum number of parameters to be estimated (<NumPars>) depends on the number of intrinsecally different bioparts (ORIs, promoters and RBSs) in the library. ORIs and promoters contribute with one parameter each one. Each RBS contributes with 2 parameters in case <ModelVariant> = *complete*, and 1 parameter in case <ModelVariant> = *reduced*. Thus, for instance, Lib30 (2 ORIS, 3 promoters, 5 RBSs) in which the copy number of one of the ORIS is known *a priori*, contains 14 effective parameters for <ModelVariant> = *complete*, and 9 in case <ModelVariant> = *reduced*.

Tensor Structure
The data structure consists of a three-dimensional MATLAB cell array of size (num_plasmids × num_promoters × num_rbss). Each entry (cell) corresponds to a single transcriptional unit (TU), defined by a specific combination of origin (plasmid), promoter, and RBS.

Each TU cell contains:

1. Transcriptional Unit Metadata

`TU_Ori`	Plasmid/origin identifier (e.g., pGreen).
`TU_Promoter`	Promoter identifier (e.g., J23106).
`TU_RBS`	RBS identifier (e.g., B0030).
`TU_Bioparts`	Comma-separated string encoding the triplet (Ori,Promoter, RBS).
`TU_Name`	Internal TU label.
`TU_color`	RGB visualization code

2. Global experimental data
The following vectors store global statistics (i.e. means and standard deviations across all the wells in all the experimental instances) for the specific growth rate and the synthesis rate (Mu and Pi). The values span the range from the sample in which the maximum growth rate is attained until that in which the maximum number of Particles is reached.

`Mu_mumax_pmax_global_mean`

`Mu_mumax_pmax_global_std`

`Pi_mumax_pmax_global_mean`

`Pi_mumax_pmax_global_std`

3. Instance- and Wells-Level Experimental Data and Parametric Sensitivities

The field Instances encodes experiment-level information for each independent experiment in which a given transcriptional unit (TU) was evaluated. The number of elements in Instances therefore equals the number of independent experiments performed for that TU. For example, if a TU was characterized in five separate experiments, Instances will contain five elements. Each Instances{i} object stores the complete information corresponding to that TU within experiment i. Within each experiment, measurements were performed across <NumWells> wells (e.g. <NumWells>=10); accordingly, each instance contains a Wells field of size (<NumWells>×1).

3.1 Biological Interpretation of Vector Lengths

The lengths of the Mu and Pi vectors (e.g., 13×1) correspond to the number of time samples recorded between:
1. The time at which the culture reaches the maximum specific growth rate (μ_max), and
2. The time at which the maximum particle (cell) number is reached.

Thus, each element of these vectors represents a measurement within the post-μ_max growth regime, which is the regime used for host-aware characterization and sensitivity analysis as described in the Supplementary Information of the associated publication.

3.2 Instance-Level Fields

Each `Instances{i}` struct contains the (n×1) vectors:

`Mu_mumax_pmax_instance_mean`

`Mu_mumax_pmax_instance_std`

`Pi_mumax_pmax_instance_mean`

`Pi_mumax_pmax_instance_std`

where n denotes the number of sampled time points between μ_max and the maximum particle number $P\max$. These quantities represent the mean and standard deviation across the <NumWells> wells of that experiment.

Additionally, the following instance-level sensitivity mappings are provided:

`S_Pi_NA_instance_values`

`S_Pi_Omega_instance_values`

`S_Pi_RBS_k0_sigma0_instance_values`

`S_Pi_RBS_inv_sigma0_instance_values`

These (n×1) vectors quantify the sensitivities of the experimental synthesis rate Π with respect to the bioparts parameters at each sampled operating point evaluated using the estimated parameters described in point 6. The definitions of these sensitivities follow the formalism described in the Supplementary Information of the host-aware characterization framework.

3.3 Wells-Level Structure

`Instances{i}.Wells` is a <NumWells>×1 cell array. Each `Wells{j}` element corresponds to one experimental well within experiment i.Each `Wells{j}` struct contains well-resolved quantities:

`Mu_mumax_pmax`

`Pi_mumax_pmax`

`S_Pi_NA_well_values`

`S_Pi_Omega_well_values`

`S_Pi_RBS_k0_sigma0_well_values`

`S_Pi_RBS_inv_sigma0_well_values`

These represent the primary experimental data and associated sensitivity evaluations at single-well resolution. The instance-level mean and standard deviation fields are computed from these well-level values.

4. Parameter estimation data aggregation flag

- `Use_mean`: Flag indicating how experimental data aggregation is handled (e.g., 'Instances').

5. Local Parameter Estimates under Leave-One-Out Cross-Validation

A Leave-One-Out (L1O) cross-validation procedure was performed across the TUs in the dataset.For each TU:

- One TU is excluded.

- The remaining TUs are used to estimate the <NumPars> parameters of the model.

The estimation procedure is repeated <NumRuns> times (independent optimization runs, usually set to <NumRuns>=10). Thus, for each leave-one-out configuration, <NumRuns> parameter vectors of length <NumPars> are obtained. These results are stored as:

`Parameters_local_raw`	(<NumRuns>×<NumPars>) matrix containing the <NumPars> estimated parameter vectors obtained when the corresponding TU was left out.
`Parameters_local_mean`	(1×<NumPars>) vector with the mean of the <NumRuns> parameter estimates for that leave-one-out configuration.
`Parameters_local_std`	(1×<NumPars>) vector with the corresponding standard deviation of the <NumRuns> parameter estimates.

These quantities describe the variability of the global model parameters under the specific L1O configuration associated with this TU cell. Notice under Leave-One-Out Cross-Validation each parameter is estimated a total number of <NumEstimates> = <NumRuns>×<NumTUs>

6. TU-Relevant Biophysical Quantities Aggregated Across All Estimations

Across the estimation procedure <NumEstimates> are obtained per parameter vector, being <NumEstimates> = <NumRuns> in case <EstimationScheme> = ALL, and <NumEstimates> = <NumTUs>×<NumRuns>in case <EstimationScheme> = L1O. From these <NumEstimates> global parameter vectors, only the parameters that are biophysically relevant for the specific TU corresponding to this cell are extracted and retained. These TU-relevant parameters are mapped into interpretable biophysical quantities, which are stored both in raw sampled form and as aggregated statistics:

`Gene_cn_raw`	(<NumEstimates> ×1)
`Gene_cn_mean`	
`Gene_cn_std`	
`Omega_raw`	(<NumEstimates> ×1)
`Omega_mean`	
`Omega_std`	
`RBS_k0_sigma0_raw`	(<NumEstimates> ×1)
`RBS_k0_sigma0_mean`	
`RBS_k0_sigma0_std`	
`RBS_inv_sigma0_raw`	(<NumEstimates> ×1). Only for <ModelVariant>=complete
`RBS_inv_sigma0_mean`	Only for<ModelVariant>=`complete`
`RBS_inv_sigma0_std`	Only for <ModelVariant>=`ccomplete`

Notice under Leave-One-Out Cross-Validation, the *_raw vectors therefore contain <NumEstimates> samples derived from the full set of L1O estimations (<NumTUs> folds × <NumRuns> runs).
These values represent the distribution of TU-specific inferred quantities under cross-validated parameter uncertainty. The corresponding mean and standard deviation fields provide compact summary statistics of this distribution. This separation between: (1) Parameters_local_* (fold-specific global model parameters), and (2) TU-specific biophysical quantities aggregated across all folds, ensures that both fold-level estimation variability and global cross-validated uncertainty are explicitly preserved in the data structure.

7. Global Pi-Dependent Mapping Values

These vectors characterize global sensitivity evaluations (see also point 3):

`S_Pi_NA_global_values`

`S_Pi_Omega_global_values`

`S_Pi_RBS_k0_sigma0_global_values`

`S_Pi_RBS_inv_sigma0_global_values`

8. Deterministic Synthesis Predictions Across Growth-Rate Slices

The field Mu_slices_values (1×10 double) defines a discrete set of growth-rate values (μ) used to interrogate the digital twin of the host-aware model. These values span the post-μ_max regime considered relevant for host-aware characterization and resource-allocation analysis. Given that global model parameters have already been estimated, the calibrated digital twin can be simulated at each prescribed μ slice. This yields deterministic predictions of synthesis rates and internal physiological quantities for the transcriptional unit (TU) represented in the current tensor cell.

These predictions are stored in the struct `Synthesis_predictions` (1×1 struct).
For each μ slice, the following <NumWells>×1 vectors are provided:

`Mu_values`	The discrete growth-rate values at which simulations are evaluated (identical to Mu_slices_values).
`Pi_pred_values`	Predicted synthesis rate (Pi) at each μ slice.
`fs_pred_values`	Predicted normalized substrate fraction (f_s), representing resource allocation to growth-related processes.
`varphi_pred_values`	Predicted effective resource flux (φ), describing intracellular resource allocation dynamics.
`load_pred_values`	Estimated translational load imposed by the TU at each μ value.
`KA_t_pred_values`	Effective translation rate (ETR)-related prediction, reflecting translation efficiency under the inferred parameterization.
`KRBS_c_values`	Effective RBS-related kinetic coefficient derived from the model.
`Svarphi_KA_t_values`	Composite quantity capturing the interaction between resource flux (φ) and translation rate (K_A^t).

Together, these quantities describe the mechanistic state of the digital twin under varying growth conditions. They provide a deterministic mapping from growth rate (μ) to synthesis rate (Π) and to internal host-aware resource variables.

8.1. Relative Parametric Sensitivities Across Growth-Rate Slices

In addition to deterministic predictions, relative parametric sensitivities are computed at each μ slice and stored in `Relative_sensitivities` (1×1 struct)
For each μ value (<NumWells>×1 vectors), the following sensitivities are provided:

`S_Pi_NA_values`	Sensitivity of predicted Π with respect to gene copy number.
`S_Pi_Omega_values`	Sensitivity of predicted Π with respect to the promoter ω parameter governing resource/translation coupling.
`S_Pi_RBS_k0_sigma0_values`	Sensitivity of predicted Π with respect to the RBS intrinsic initiation capacity $\kappa^0 = k^0/\sigma^0$.
`S_Pi_RBS_inv_sigma0_values`	Sensitivity of predicted Π with respect to the inverse sensitivity RBS-related parameter.

These sensitivities are computed using the calibrated digital twin and quantify the local dependence of synthesis rate on mechanistic parameters at each growth condition. Importantly, these values provide a growth-rate-resolved mechanistic sensitivity landscape, complementing:

- Instance- and Wells-level sensitivities (experiment-specific),
- Global Π-dependent experimental mappings (condition-aggregated),
- Monte Carlo uncertainty propagation (distribution-level).

9. Monte Carlo (MC) Uncertainty Propagation

To quantify how parameter uncertainty propagates into synthesis-rate predictions and host-aware physiological variables, a Monte Carlo (MC) uncertainty propagation procedure is performed for each transcriptional unit (TU).

9.1 Parameter Sampling

From the parameter distributions associated with the TU (derived from the <NumEstimates> parameter estimated values), <NumRndSamples>=1000 random samples are drawn. These sampled values are stored as:

`Gene_cn_MC_samples`	(<NumRndSSamples>×1 double)
`Omega_MC_samples`	(<NumRndSSamples>×1 double)
`RBS_k0_sigma0_MC_samples`	(<NumRndSamples>×1 double)
`RBS_inv_sigma0_MC_samples`	(<NumRndSSamples>×1 double). Only for <ModelVariant>=`complete`

Each index (k = 1,. . . ,<NumRndSamples>) defines one complete sampled parameter realization for the TU.

9.2 Forward Propagation Through the Digital Twin

For each of the <NumRndSamples> sampled parameter sets, the calibrated digital twin is evaluated exactly as described in Section 8:

- The predefined Mu_slices_values (<NumMuSlices> growth-rate values) are used.
- For each μ slice, synthesis rate (Π) is predicted.
- Internal host-aware quantities (f_s, φ, load, ETR-related variables, RBS terms) are computed.

- Relative parametric sensitivities are evaluated.

Each realization is stored as one element of the (1×<NumRndSamples> struct) `MC_samples`. Every `MC_samples{k}` struct contains:

`Synthesis_predictions`	Deterministic predictions for that sample
`Relative_sensitivities`	Growth-rate-resolved sensitivities

Thus, `MC_samples` stores <NumRndSamples> full digital-twin realizations, each evaluated across <NumMuSlices> growth-rate slices.

9.3 Distribution Construction Across Growth-Rate Slices

After computing all <NumRndSamples> realizations, the results are reorganized by growth-rate slice. For each μ value, the distribution of predicted synthesis rates across the <NumRndSamples> samples is constructed. These distributions are stored in the 1×<NumMuSlices> struct `MC_mu_slices`. Each slice j contains:

`Mu_slice`	The corresponding growth-rate value.
`Pi_pdf`	Kernel density estimate (probability distribution) of predicted syntesis rate Π across the <NumRndSamples> MC realizations.
`Pi_pred_mean`	Mean predicted synthesis rate.
`Pi_pred_std`	Standard deviation of predicted synthesis rate.
`Pi_pred_q50`	Median prediction.
`Pi_pred_q25, Pi_pred_q75`	25th and 75th percentiles.
`Pi_pred_q2p5, Pi_pred_q97p5`	2.5th and 97.5th percentiles (approximate 95% credible interval).

9.4 Interpretation

The Monte Carlo procedure therefore provides:

1. Distribution-level uncertainty in synthesis-rate predictions at each growth condition.
2. Growth-rate-resolved credible intervals.
3. Full propagation of cross-validated parameter uncertainty through the mechanistic digital twin.

This complements:

- Deterministic predictions (Section 8),
- Local growth-conditioned sensitivities (Section 8.1),
- Fold-level parameter variability (Sections 5–6),

and yields a comprehensive uncertainty-aware characterization of each transcriptional unit within the library.

Supplemental References

Acerbi, L., and Ma, W. J. (2017). Practical bayesian optimization for model fitting with bayesian adaptive direct search. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (p. 1834–1844). Curran Associates, Inc. volume 30.

Bremer, H., and Dennis, P. P. (2008). Modulation of chemical composition and other parameters of the cell at different exponential growth rates. *EcoSal Plus*, *3*, 10.1128/ecosal.5.2.3.

Ceroni, F., Algar, R., Stan, G. B., and Ellis, T. (2015). Quantifying cellular capacity identifies gene expression designs with reduced burden. *Nature Methods*, *12*, 415–418.

Chure, G., and Cremer, J. (2023). An optimal regulation of fluxes dictates microbial growth in and out of steady state. *eLife*, *12*, e84878.

Egea, J. A., Henriques, D., Cokelaer, T., Villaverde, A. F., MacNamara, A., Danciu, D.-P., Banga, J. R., and Saez-Rodriguez, J. (2014). MEIGO: an open-source software suite based on metaheuristics for global optimization in systems biology and bioinformatics. *BMC Bioinformatics*, *15*, 136.

Scott, M., Gunderson, C. W., Mateescu, E. M., Zhang, Z., and Hwa, T. (2010). Interdependence of cell growth and gene expression: Origins and consequences. *Science*, *330*, 1099–1102.