Supplementary Materials: Component Characterization in a Growth-Dependent Physiological Context: Optimal Experimental Design

Nathan Braniff [1], Matthew Scott [1], and Brian Ingalls [1]*

Contents

1. Derivation of the Physiological Gene Expression Model

1.1. Protein fraction of cell mass

The total cell mass M_{Tot} can be partitioned into fractions of protein and other constituents. The protein fraction of the cellular mass, Φ_{pr}, can be fit to data from [1] with a linear function:

$$\Phi_{pr} = \kappa_{pr}\lambda + \Phi_{pr0} \tag{1}$$

The fit shown in Fig. S1 provides estimates of $\kappa_{pr} = -6.47$ min and $\Phi_{pr0} = 0.65$.

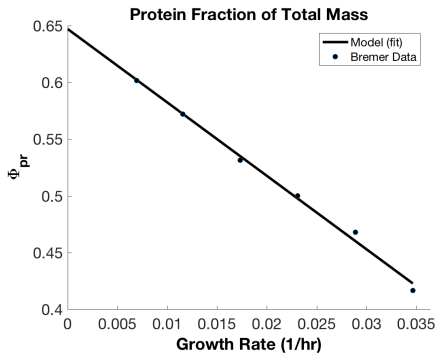

Figure S1. Fit of protein fraction of the cell mass using data from Table 2 of [1] .

1.2. Growth Dependence of the Total RNAP Population

The total RNAP fraction of the overall protein mass, Φ_p, exhibits an approximately linear relationship with growth rate:

$$\Phi_p = \kappa_p\lambda + \Phi_{p0}. \tag{2}$$

We fit this to data provided in [1], yielding estimates $\kappa_p = 0.30$ min and $\Phi_{p0} = 0.0074$, shown in Figure S2.

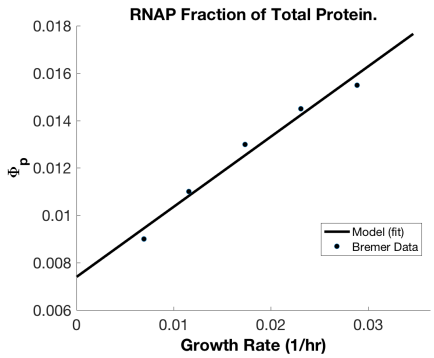

Figure S2. Fit of RNAP fraction of protein mass using data from Table 3 of [1].

1.3. Available RNAP

The total RNAP population can be partitioned by state: freely diffusing; weakly DNA bound at a non-specific site; actively transcribing other genes; paused and non-functioning during transcription (paused); or immature [2–4]. We are interested in those RNAP that can initiate transcription. It has been assumed in past works that transcriptional initiation is proportional to the free fraction [4]. However, our thermodynamic equilibrium model of the promoter accounts for competition between non-specific binding sites and the specific promoter site. We therefore define the combined pool of free and non-specifically bound RNAPs as the *available* RNAP pool, P_a.

We use a course-grained partitioning of total RNAPs, P_{Tot}: dividing them into i) the available RNAPs, P_a, ii) the bound RNAPs, P_b. We will neglect the immature population as this has been measured to be a small fraction ($< 10\%$) of the total [2]. We can therefore write the total as

$$P_{Tot} \approx P_a + P_b \tag{3}$$

The available subgroup, P_a, includes those RNAPs freely diffusing in the nucleoid as well as those non-specifically bound to the DNA. These two sub-groups within the available pool, P_a, have been observed to be in rapid equilibrium [2]. We assume that the bound RNAPs, P_b, include all those bound to the DNA that are actively transcribing or paused in transcription. We write

$$P_a = P_{Tot} - P_b = P_{Tot}(1 - \Phi_b) = P_{Tot}\Phi_a, \tag{4}$$

where Φ_b is the fraction of transcription-occupied RNAPs unavailable for initiation of transcription and Φ_a is the faction of those that are available.

The growth-dependent fraction of RNAPs that are available to initiate transcription (non-transcribing) is still poorly understood. Work by Klumpp and Hwa [4] as well as Bremer and colleagues [3,5] have attempted to describe the partitioning of RNAP into free and occupied fractions across growth rates without consensus. However, all agree the concentration of free RNAP increases with growth rate. Recent spatial imaging experiments of fluorescently-tagged RNAP suggest these previous theories may be partially inaccurate; specifically, this new data suggest much larger fractions of RNAP are busy in transcription, and smaller fractions are non-specifically bound or paused, than previously expected [2,6].

Current direct measurements of the dependence of Φ_b or Φ_a on growth rate are sparse. Bakshi *et al.* examine only a single growth rate (doubling time of approximately 42 min) at which they estimate the partitioning of RNAP using spatial tracking of tagged molecules [2]. Stracy *et al.* have since compared RNAP partitioning between growth on minimal and rich media in a spatial tracking study [6]. A plot of these three data points is shown in Figure S3. It should be noted that the studies use somewhat different experimental methodologies, and future work with multiple growth rates in the same strain and conditions is needed to provide a confident description. Growth rates for strains in

Stracy *et al.* were previously reported in [7]. From the limited available data (Figure S3) we hypothesize

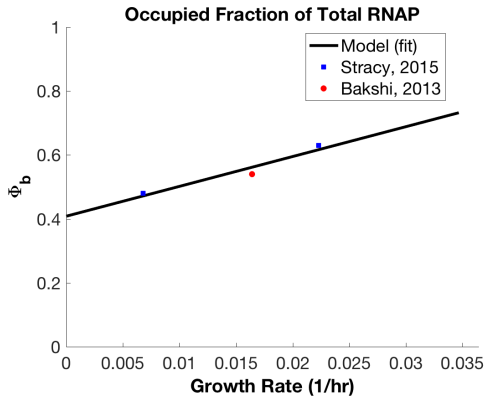

Figure S3. Fraction of total RNAP occupied in transcription, as it depends on growth rate. Data from [6], [7] and [2].

a linear dependence for Φ_b:

$$\Phi_b = \kappa_b \lambda + \Phi_{b0}. \tag{5}$$

Fitting yields the estimates $\kappa_b = 9.3$ min and $\Phi_{b0} = 0.41$. Then $\Phi_{a0} = 1 - \Phi_{b0} = 0.59$ and $\kappa_a = -\kappa_b = -9.3$ min.

We can then re-write this relationship as

$$\Phi_a = \kappa_a \lambda + \Phi_{a0}. \tag{6}$$

Using this relation we can construct an expression for the available RNAP by multiplying P_{Tot} by Φ_a;

$$P_a = \frac{\rho V_0}{m_{rnap}} \left(\kappa_a \lambda + \Phi_{a0}\right) \left(\kappa_p \lambda + \Phi_{p0}\right) \left(\kappa_{pr} \lambda + \Phi_{pr0}\right) e^{(C+D)\lambda}. \tag{7}$$

Dividing the expression for the available RNAP by the expression for cell volume yields the concentration of available RNAP, which our model predicts will decrease with growth rate. Past works have observed an increasing transcription rate per gene (particularly moving from slow to moderate growth rates) [8]. This suggests that the free RNAP concentration must therefore be increasing with the growth rate. However, although our model predicts a decreasing RNAP concentration, the transcription rate is predicted to increase (from slow to moderate growth) as the RNAP density along the genomic DNA increases. This is consistent with the fact that DNA-binding proteins like RNAP and transcription factors (TFs) are mostly confined to the nucleoid DNA [2,9] where they diffuse along the DNA strand. As a result, in our model, total RNAP concentration is less relevant than the RNAP density along the genomic DNA. We hypothesize this may cause the non-monotonic relation for transcription rate per gene that has previously been observed by Liao *et al.* for constitutive promoters [10].

1.4. Transcription Rate

Following [11,12], our model of transcription involves interactions between RNA polymerases (RNAPs), transcription factors (TFs), promoter copies and non-specific binding sites along the genomic DNA. As described in the main text, at each time point we suppose that there are P_a available RNAP copies and T_a active transcription factor copies diffusing along the genomic DNA, and that the DNA contains N_s non-specific binding sites to which the DNA binding proteins may weakly attach and g copies of the regulated promoter of interest. Further we assume that $N_s \gg P_a, T_a, g$ and that each binding of an RNAP or a TF to a non-specific site or a promoter can be characterized by an associated

binding energy: ϵ_{rn} and ϵ_{rg} for RNAP to the non-specific sites and promoters respectively, and ϵ_{tn} and ϵ_{tg} for transcription factor to non-specific sites and promoters respectively (all ϵ are negative [13]).

We use these species and site counts to enumerate the possible arrangements of RNAP and TF across the genome, and we use the binding energies to derive Boltzmann weights for each arrangement [11]. This allows us to construct a partition function. For example if all the P_a RNAPs and u TFs are bound to nonspecific sites, the weighted enumeration for this group of possible micro-states (the partition function) can be written as

$$Z(P_a,u)=\underbrace{\frac{N_s!}{P_a!T_a!(N_s-P_a-T_a)!}}_{\text{\# of micro states}}\underbrace{e^{-P_a\frac{\epsilon_{rn}}{k_BT}}e^{-T_a\frac{\epsilon_{tn}}{k_BT}}}_{\text{energetic favorability}}\approx\frac{N_s^{(P_a+T_a)}}{P_a!T_a!}e^{-P_a\frac{\epsilon_{rn}}{k_BT}}e^{-T_a\frac{\epsilon_{tn}}{k_BT}}, \tag{8}$$

where the approximation holds because N_s is large compared with the other quantities [11]. We can then construct the partition function for the total number of arrangements of a single promoter copy ($g=1$) as

$$\begin{aligned}Z^{Tot}_{g=1}(P_a,T_a)=\;&\underbrace{Z(P_a,T_a)}_{\text{Empty Promoters}}+\underbrace{Z(P_a-1,T_a)e^{-\frac{\epsilon_{pg}}{k_BT}}}_{\text{RNAP on Promoter}}\\&+\underbrace{Z(P_a,T_a-1)e^{-\frac{\epsilon_{tg}}{k_BT}}}_{\text{TF on Promoter}}+\underbrace{Z(P_a-1,T_a-1)e^{-\frac{\epsilon_{pg}+\epsilon_{tg}+\epsilon_{pt}}{k_BT}}}_{\text{RNAP and TF on Promoter}}.\end{aligned} \tag{9}$$

Here ϵ_{pt} is the binding energy between RNA polymerase and transcription factor when both are bound to the same promoter. We can use this expression to write the equilibrium probability of the single promoter being occupied by an RNAP by taking the ratio of the partition functions for the RNAP-bound states to the total partition function;

$$\begin{aligned}p_{bnd}&=\frac{Z^{Bnd}_1(P_a,T_a)}{Z^{Tot}_1(P_a,T_a)}\\p_{bnd}&=\frac{\frac{P_a}{N_s}e^{-\frac{\Delta\epsilon_r}{k_BT}}+\frac{P_aT_a}{N_s^2}e^{-\frac{(\Delta\epsilon_r+\Delta\epsilon_t+\epsilon_{pt})}{k_BT}}}{1+\frac{P_a}{N_s}e^{-\frac{\Delta\epsilon_r}{k_BT}}+\frac{T_a}{N_s}e^{-\frac{\Delta\epsilon_t}{k_BT}}+\frac{P_aT_a}{N_s^2}e^{-\frac{(\Delta\epsilon_r+\Delta\epsilon_t+\epsilon_{pt})}{k_BT}}}\end{aligned} \tag{10}$$

Here the $\Delta\epsilon$ values are the differences between the energy involved in binding the promoter and the background non-specific binding: $\Delta\epsilon_t=\epsilon_{tg}-\epsilon_{tn}$ and $\Delta\epsilon_r=\epsilon_{rg}-\epsilon_{rn}$. Note, $\epsilon_{tn}>\epsilon_{tg}$ and $\epsilon_{pn}>\epsilon_{pg}$ so that $\Delta\epsilon_t$ and $\Delta\epsilon_r$ are both negative [13]. Denoting the Boltzmann weights as $K_r=e^{-\frac{\Delta\epsilon_r}{k_BT}}$, $K_t=e^{-\frac{\Delta\epsilon_t}{k_BT}}$ and $K_{rt}=e^{-\frac{(\Delta\epsilon_r+\Delta\epsilon_t+\epsilon_{pt})}{k_BT}}$ yields the following simplified form;

$$p_{bound}=\frac{\frac{P_a}{N_s}K_r+\frac{P_aT_a}{N_s^2}K_{rt}}{1+\frac{P_a}{N_s}K_r+\frac{T_a}{N_s}K_t+\frac{P_aT_a}{N_s^2}K_{rt}} \tag{11}$$

Next, with RNAP bound to a certain fraction of the promoters (or on average a certain fraction of the time over the relevant times scale of initiation), we assume open complex and promoter escape occurs at a fixed rate α (NATE cite for open complex and escape rate), giving

$$\text{Initiation Rate (for } g\text{=1)}=\alpha\frac{\frac{P_a}{N_s}K_r+\frac{P_aT_a}{N_s^2}K_{rt}}{1+\frac{P_a}{N_s}K_r+\frac{T_a}{N_s}K_t+\frac{P_aT_a}{N_s^2}K_{rt}} \tag{12}$$

So far, we have described a single promoter. To address the case of multiple promoters we would need to account for the cross-correlation between their occupancy by the transcription factor or RNAP. This has little effect at high TF copy numbers (although at low copy numbers the occupancy of one promoter significantly decreases the odds of another being occupied). We assume that the RNAP and TF populations are considerably larger than g, which allows us to assume the promoters function approximately independently [12]. In that case, we can scale equation (12) to arrive at the initiation rate as a function of g.

$$\text{Initiation Rate} = \alpha g \frac{\frac{P_a}{N_s}K_r + \frac{P_a T_a}{N_s^2}K_{rt}}{1 + \frac{P_a}{N_s}K_r + \frac{T_a}{N_s}K_t + \frac{P_a T_a}{N_s^2}K_{rt}} \tag{13}$$

We assume that the initiation rate is the limiting step in transcription, as elongation rates are generally faster [14–16]. We can therefore write the overall transcript production rate as follows;

$$\text{Transcript Production Rate} = \alpha g \frac{\frac{P_a}{N_s}K_r + \frac{P_a T_a}{N_s^2}K_{rt}}{1 + \frac{P_a}{N_s}K_r + \frac{T_a}{N_s}K_t + \frac{P_a T_a}{N_s^2}K_{rt}}$$

Where:

$$\begin{aligned} P_a &= \frac{\rho V_0}{m_{rnap}} \left(\kappa_a \lambda + \Phi_{a0}\right) \left(\kappa_p \lambda + \Phi_{p0}\right) \left(\kappa_{pr} \lambda + \Phi_0^{Prot}\right) e^{(C+D)\lambda}, \\ N_s &= \frac{\eta}{\lambda C}\left(e^{(C+D)\lambda} - e^{D\lambda}\right) \\ g &= e^{((C+D)-l_{ori}C)\lambda} \end{aligned} \tag{14}$$

1.5. Total Ribosome Population

From [17], we have a linear relation for the fraction of protein mass that is composed of ribosomal protein:

$$\Phi_r = \kappa_r \lambda + \Phi_{r0}. \tag{15}$$

Fitting the model to data from [1] yields estimates of $\kappa_r = 5.5$ min and $\Phi_{r0} = 0.030$, as shown in Figure S4.

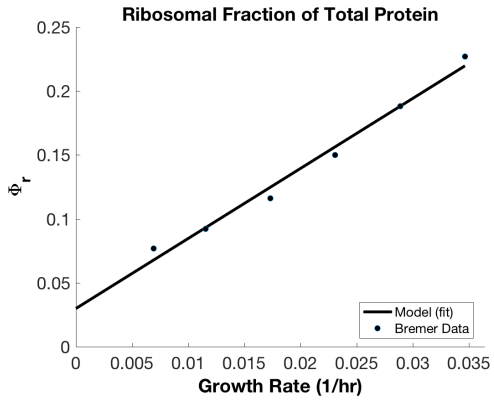

Figure S4. Ribosomal protein fraction of the total protein mass as a function of growth rate. Data from Table 3 of [1].

1.6. Translation Rate

Both Klumpp *et al.* and Liang *et al.* note that the translation rate per transcript is roughly constant and speculate that this could be due to a constant concentration of free ribosomes maintained

by regulatory feedback [4,18]. Dai's more recent results suggest that the fraction of inactive (non-translating) ribosomes, Φ_{inact}, is constant at the moderate to fast growth rates we consider here [19]. The total ribosome concentration increases with growth rate, because total ribosome copy number scales faster than the cell volume:

$$\begin{aligned}\frac{R_{Tot}}{V} &= \left(\frac{\rho V_0}{m_{rib}}\left(\kappa_r\lambda+\Phi_{r0}\right)\left(\kappa_{pr}\lambda+\Phi_{pr0}\right)e^{(C+D)\lambda}\right)\left(V_0e^{(C+D)\lambda}\right)^{-1}\\ &= \frac{\rho}{m_{rib}}\left(\kappa_r\lambda+\Phi_{r0}\right)\left(\kappa_{pr}\lambda+\Phi_{pr0}\right)\end{aligned} \tag{16}$$

Therefore, the inactive ribosome concentration $[R_{inact}]$ also increases, because Φ_{inact} is constant:

$$\frac{R_{inact}}{V} = \Phi_{inact}\frac{R_{Tot}}{V} = \Phi_{inact}\frac{\rho}{m_{rib}}\left(\kappa_r\lambda+\Phi_{r0}\right)\left(\kappa_{pr}\lambda+\Phi_{pr0}\right) \tag{17}$$

The inactive ribosomes R_{inact} are either non-functioning (stalled or assembling), R_{nf}, or free, R_f, so

$$R_{inact} = R_{nf} + R_f \tag{18}$$

Following [4,18], we presume that the concentration of free ribosomes, $[R_f]$ is constant. Thus the fraction of inactive ribosomes that are free, Φ_f must scale inversely with the ribosomal fraction of the mass:

$$\Phi_f = \frac{1}{\left(\kappa_r\lambda+\Phi_{r0}\right)\left(\kappa_{pr}\lambda+\Phi_{pr0}\right)} \tag{19}$$

and therefore the fraction of inactive ribosomes, Φ_{nf} is

$$\Phi_{nf} = 1 - \frac{1}{\left(\kappa_r\lambda+\Phi_{r0}\right)\left(\kappa_{pr}\lambda+\Phi_{pr0}\right)} \tag{20}$$

This then yields an expression for the free ribosome concentration that is constant across growth rates:

$$\frac{R_{free}}{V} = \Phi_f\Phi_{inact}\frac{R_{Tot}}{V} \tag{21}$$

Using a mass action expression for translation:

$$\text{Translation Rate (in copy \#)} = \beta\frac{R_f}{V}X_{rna} \tag{22}$$

This implies the translation efficiency per mRNA, α_p, is constant:

$$\text{Translation Rate (per mRNA)} = \beta\frac{R_f}{V} \tag{23}$$

This result conflicts with our assumption of $\Phi_f \approx \Phi_{inact}$ and R_{nf} being negligible. However, in a related study of translation in *Bacillus subtilis*, Borkowski *et al.* observe a decreasing translation efficiency (per mRNA) with increasing growth rates and they infer that this is due to a decreasing free ribosome concentration [20]. The authors use this varying free ribosome concentration to test the mass action (linear) translation model used by Klumpp *et al.* [8]. With a varying free ribosome concentration,

R_f/V, the ratio of efficiencies between two different RBS is constant if the mass action (linear) model of Klumpp is used:

$$\text{Ratio of Translation Efficiencies} = \frac{\beta_1 \frac{R_f}{V}}{\beta_2 \frac{R_f}{V}} = \frac{\beta_1}{\beta_2} \tag{24}$$

However, Borkowski *et al.* observe that the ratio of translation efficiencies between different transcripts varies across growth rates. This suggests that a different model of translation initiation may be more appropriate, and that constant translation rate may not be caused by constant free ribosome concentrations (applying the same argument to both *B. subtilis* and *E. coli*).

Borkowski *et al.* propose a Michaelis-Menten model of translation initiation in terms of the free ribosome concentration to explain the non-constant translation efficiency ratios [20]:

$$\text{Translation Rate (copy \#)} = \frac{\beta \frac{R_f}{V}}{K_M + \frac{R_f}{V}} X_{rna} \tag{25}$$

with translation efficiency then expressed as;

$$\text{Translation Rate (per mRNA)} = \frac{\beta \frac{R_f}{V}}{K_M + \frac{R_f}{V}} \tag{26}$$

In this model, the mRNA's RBS is characterized by two constants, β, the maximal translation initiation rate per mRNA, and K_M, a half-saturating constant specific to the given RBS. This model is also justified by the mechanisms of translation initiation in which the mRNA species may be in limiting quantities and become saturated. This Michaelis-Menten formulation agrees with the observation of constant translation efficiency in Klumpp and Liao [8,18], under the assumption that the RBS of the gene has a low K_M value and is near saturation over the relevant growth rates. We have also assumed that initiation of translation is the limiting step in protein production, and that it is slower than translation elongation [21,22].

2. Details of the Multiple-shooting Algorithm and Optimization

Our OED algorithm can be classified as a direct optimal control approach – following a discretize-then-optimize procedure [23], where the system dynamics are implemented numerically, the control variables are discretized and selected, the system response is simulated, and then the controls are adjusted to improve the objective. However, for our dynamic, non-linear system we found this approach performed poorly if implemented in a naive manner. Multiple shooting and collocations methods provide improvements by discretizing the simulation along with the controls [23,24]. In multiple shooting specifically, the simulation is partitioned into a series of initial value problems [23]. This process increases the dimensionality of the problem but also improves the problem structure, giving the optimization algorithm more information about how each control contributes to the objective function [25]. This process has been referred to as 'lifting', where the problem is lifted into a higher-dimensional, but more easily navigated space [25]. The problem structure can be further improved by including derivative information for the object and constraints, which can be done in a straightforward manner using algorithmic differentiation tools available in CasADi [26].

Below, we provide a brief overview of the multiple shooting algorithm used in this work for optimal experimental design (OED). Further details on the implementation of similar algorithms can be found in [27–29]. We will describe the algorithm implementation in pseudo-code, outlining the use of CasADi's symbolic interface. CasADi uses symbolics (on the front-end) to create mathematical expressions; details of the back-end implementation can be found in [30]. An example of some (pseudo) syntax for the CasADi MATLAB interface is given in Algorithm 1. In this example we define two

Algorithm 1 CasADi Example

```
x=SX.sym('x')
y=SX.sym('y')
z=x+y
f=function('f',{x,y},{z})
f(1,2)
»3
w1=SX.sym('w1')
w2=SX.sym('w2')
f(w1,w2)
»'w1+w2'
```

symbolic variables 'x' and 'y' with 'SX.sym()'. We then create the symbolic expression 'z=x+y', where 'z' now symbolically means 'x+y'. To be able to evaluate that expression on new inputs, we define a function 'f' that maps 'x' and 'y' to the output described by 'z'. Below this, we see that we can use the function 'f' to map specific numbers to a numerical output, but we can also use it to create new symbolic expressions (i.e. 'w1+w2'). These in turn could be used to build layers of symbolics as we will do in the OED algorithm. The reader is referred to CasADi manual for further details on the internal functions [30]. We mix use of both the MX and SX symbolic classes, which have different computational properties [30], however the reader can ignore this for general understanding (MX symbolics are created like SX symbols but with 'MX.sym()').

For simplicity we will restate the system dynamics for a single sub-experiment as follows

$$\frac{d\boldsymbol{Y}}{dt} = \boldsymbol{G}(\boldsymbol{Y}, \boldsymbol{\theta}, \lambda, u(t), W(t)) \tag{27}$$

Here the state $\boldsymbol{Y}$ contains $\boldsymbol{X}$ (2 components), all $\bar{\boldsymbol{X}}_{\theta_i}$ for each of 6 parameters (12 components), $\hat{w}$ (2 components), and the unique elements of $\mathcal{I}$ (21 components), for a total of 37 components. The details of the dependence of $\boldsymbol{Y}$ on λ, $u(t)$ and $W(t)$ can be found in the main text. Our algorithm begins by defining this right-hand side as a symbolic expression in CasADi, via Algorithm 2. In defining

Algorithm 2 RHS function definition

```
1: Y=SX.sym('Y',37)                    ▷ Define symbolic variables
2: θ=SX.sym('Y',6)
3: λ=SX.sym('λ')
4: u=SX.sym('u')
5: w=SX.sym('w',2)                     ▷ One each for w_rna, w_prot
6: RHS=G(Y,θ,λ,u,w)                    ▷ Implement G algebraically
7: g= function('g',{Y,λ,u,w},{RHS})    ▷ Create RHS CasADi function
```

$\boldsymbol{G}(\boldsymbol{Y}, \boldsymbol{\theta}, \lambda, u, w_{rna}, w_{prot})$ algebraically (line 7 above, details omitted), we used CasADi's algorithmic differentiation and matrix algebra abilities. This can be done by first defining the RHS for the state variables $\boldsymbol{X}_{rna}$ and $\boldsymbol{X}_{prot}$ symbolically. Then the sensitivities and FIM RHS functions can be determined from the corresponding RHS expression by using the `jacobian` function, the `jtimes` function and the matrix product operator, among others.

Using the symbolic function for the overall RHS, $\boldsymbol{G}$, we can construct a symbolic function, $\bar{\boldsymbol{G}}_1$, for a single step of an explicit, fixed-step-size numerical integration scheme. We used a fourth-order Runge-Kutta scheme as described in the CasADi examples [31]. External ODE solvers, like the Sundials suite [32], can be used, but defining an explicit integrator in CasADi has the advantage that the integrator itself can be algorithmically differentiated. This is useful for providing first and second order integral information to the NLP solver. In contrast, the use of conditional statements by variable step-size or implicit solvers generally precludes algorithmic differentiation. The single step of the RK4 integrator is given as shown in Algorithm 3, see [31] for further details.

Algorithm 3 Define RK4 Scheme

Define Symbolic $\boldsymbol{Y}_{input}$ ▷ Define an input $\boldsymbol{Y}$ vector
$k_1 = g(\boldsymbol{Y}_{input}, \boldsymbol{\theta}, \lambda, u, w)$ ▷ Implement the RK4 sub-steps
$k_2 = g(\boldsymbol{Y}_{input} + \Delta t k_1/2, \boldsymbol{\theta}, \lambda, u, w_{rna}, w_{prot})$
$k_3 = g(\boldsymbol{Y}_{input} + \Delta t k_2/2, \boldsymbol{\theta}, \lambda, u, w)$
$k_4 = g(\boldsymbol{Y}_{input} + \Delta t k_3, \boldsymbol{\theta}, \lambda, u, w)$
$\boldsymbol{Y}_{output} = \boldsymbol{Y}_{input} + \Delta t(k_1 + 2k_2 + 2k_3 + k_4)/6$ ▷ Create a symbolic expression for the output
$\bar{\boldsymbol{G}}_1$=function('$\bar{\boldsymbol{G}}_1$',{$\boldsymbol{Y}_{input}$,$\boldsymbol{\theta}$,λ,u,w},{$\boldsymbol{Y}_{output}$}) ▷ Create a function mapping from $\boldsymbol{Y}_{input}$ to $\boldsymbol{Y}_{output}$

Recall that for a single sub-experiment, the input $u(t)$ is piecewise-constant over six 100 min intervals, $W(t)$ (i.e. $w_{rna}(t)$ and $w_{prot}(t)$) is piecewise-constant over forty-eight 12.5 min intervals, and the growth rate λ is constant. We label these discretized controls by their corresponding intervals as follows: growth controls, $\lambda^{(i)}$, where $i \in \{1, , 2, 3\}$ (each sub-experiment); induction controls, $u^{(j,i)}$, where $j \in \{1, \ldots, 6\}$ (each induction interval & sub-experiment); and sampling controls, $w_{rna}^{(k,j,i)}$ and $w_{rna}^{(k,j,i)}$, where $k \in \{1, \ldots, 48\}$ (each sampling interval, induction interval & sub-experiment). Because the 48 sampling intervals are the shortest of the piecewise constant intervals, each has constant controls (in λ, u and $w_{(species)}$) over its duration. We can therefore iterate the single RK4 step function, $\hat{\boldsymbol{G}}_1$, to create an integrator, $\hat{\boldsymbol{G}}$, that maps the state at the beginning of the sampling interval to the end, 12.5 min later, with a constant set of controls (see Algorithm 4).

Algorithm 4 Iterate RK4 over the sampling (smallest) control interval

$\boldsymbol{Y}_o$=MX.sym('$\boldsymbol{Y}_o$')
$\boldsymbol{Y}_{iter}$=$\boldsymbol{Y}_o$
for $12.5/\Delta t$ **do** ▷ Iterate, advancing Δt time units each loop
    $\boldsymbol{Y}_{iter} = \bar{\boldsymbol{G}}_1(\boldsymbol{Y}_{iter}, \boldsymbol{\theta}, \lambda, u, w_{rna}, w_{prot})$ ▷ Apply $\bar{\boldsymbol{G}}_1$ to the state $\boldsymbol{Y}_{iter}$ each loop
end for
$\bar{\boldsymbol{G}}$=function('$\bar{\boldsymbol{G}}$',{$\boldsymbol{Y}_o$,$\boldsymbol{\theta}$,λ,u,w},{$\boldsymbol{Y}_{iter}$}) ▷ Create function, maps interval start, $\boldsymbol{Y}_o$, to end, $\boldsymbol{Y}_{iter}$

To determine the D-optimality score we need to integrate the RHS over the total time (0 to 600 min) for each sub-experiment. The final objective value can be computed from the Fisher information entries (the 17th to 37th components) at the final time, $Y(t = t_f)_{17\ldots37}$, in each of the sub-experiments. To apply multiple-shooting, we partitioned each sub-experiment's duration into six shooting intervals. We used intervals of 100 min, corresponding to the six constant-u induction intervals. We treat each of these shooting intervals as a separate initial value problem, with its own initial conditions $\boldsymbol{Y}_o^j$. Algorithm 5 shows how we use $\bar{\boldsymbol{G}}$ to propagate these initial conditions through the series of shooting intervals, linking the initial value problems with constraints to enforce continuity. There are seven initial conditions $\boldsymbol{Y}_o^j$ for each sub-experiment because the final time is treated as a (dummy) initial condition; this increases the sparsity of the problem. Using the initial conditions, $\boldsymbol{Y}_o^j$, as optimization variables, along with the discretized controls, provides a number of benefits. It gives the NLP solver direct access to the system state at regular intervals throughout the simulation time. This improves the problem structure as the NLP solver can alter the states directly. The continuity constraints then propagate this information to the control variables. Moreover, these states increase the sparsity of the NLP problem because the coupling of the controls and the objective across the simulation is partitioned by the additional optimization variables. The system dynamics in each shooting interval only depend on controls in the other intervals via the continuity constraints.

As shown in Algorithm 5, we loop over each sub-experiment, induction/shooting interval and sampling interval, iteratively building up a symbolic expression for the objective function and for the nonlinear constraint functions. At the beginning of the experiment, we create vectors 'CtrlVec' and 'CnstrnVec' which, as we move through the three nested loops, are filled with symbolic terms for each of the NLP optimization variables and the non-linear constraints, respectively. The elements of

Algorithm 5 Construct control problem

CtrlVec={} ▷ Empty vector for OED control symbols
lbw=[] ▷ Empty vector lower bound of OED control symbols
ubw=[] ▷ Empty vector upper bound of OED control symbols
CnstrnVec={} ▷ Empty vector for nonlinear constraint symbols
lbc=[] ▷ Empty vector lower bound of nonlinear constraints
ubc=[] ▷ Empty vector upper bound of nonlinear constraints
FIM$= \bar{0}$
for $i = 1 : 3$ **do** ▷ Loop over sub-experiments
    λ_i = MX.sym('$\lambda^{(i)}$') ▷ Create λ control, one for each loop
    CtrlVec={CtrlVec, $\lambda^{(i)}$} ▷ Add $\lambda^{(i)}$ to control vector
    lbw = [lbw; λ_{min}]; ▷ Restrict growth rates to feasible range
    ubw = [ubw; λ_{max}];

    $\boldsymbol{Y}_o^{(0,i)}$ = MX.sym('$Y_o^{(0,i)}$', 37); ▷ Create initial condition state for each sub-experiment
    CtrlVec={CtrlVec, $\boldsymbol{Y}_o^{(0,i)}$ } ▷ Add it to the control vector
    lbw = [lbw; $\bar{0}$];
    ubw = [ubw; $\bar{\text{Inf}}$];

    CnstrnVec={CnstrnVec, $\boldsymbol{Y}_o^{(0,i)} -$SteadyState($\lambda^{(i)}$,$\boldsymbol{\theta}$) } ▷ Constrain IC to be at steady state
    lbc = [lbc; $\bar{0}$]; ▷ Bounds for constraint are 0, implying equality
    ubc = [ubc; $\bar{0}$];

    for $i = 1 : 6$ **do** ▷ Loop over shooting/induction interval
        $u^{(j,i)}$ = MX.sym('$u^{(j,i)}$') ▷ Create u control, one for each sub-exp. & induction intrvl.
        CtrlVec={CtrlVec, $u^{(j,i)}$} ▷ Add $u^{(j,i)}$ to control vector
        lbw = [lbw; u_{min}]; ▷ Restrict u to feasible range
        ubw = [ubw; u_{max}];

        for $j = 1 : 48$ **do**
            $w^{(k,j,i)}$ = MX.sym('$w^{(k,j,i)}$',2) ▷ Create w for each sub-exp., induction & samp. intrvl
            CtrlVec={CtrlVec, $w^{(k,j,i)}$} ▷ Add $w^{(k,j,i)}$ to control vector
            lbw = [lbw; $\bar{0}$]; ▷ Restrict w to feasible range
            ubw = [ubw; w_{max}];

            $\boldsymbol{Y}_o^{(j-1,i)} = \bar{\boldsymbol{G}}(\boldsymbol{Y}_o^{(j-1,i)}, \boldsymbol{\theta}, \lambda^{(i)}, u^{(j,i)}, w^{(k,j,i)})$ ▷ Advance (symbolic) state vector
        end for

        $Y_o^{(j,i)}$ = MX.sym('$Y_o^{(j,i)}$', 37); ▷ Create new shooting interval IC
        CtrlVec={CtrlVec, $Y_o^{(j,i)}$ } ▷ Add it to the control vector
        lbw = [lbw; $\bar{0}$];
        ubw = [ubw; $\bar{\text{Inf}}$];

        CnstrnVec={CnstrnVec, $Y_o^{(j,i)} - Y_o^{(j-1,i)}$ } ▷ Constrain $Y_o^{(j,i)}$ for continuity with $Y_o^{(j-1,i)}$
        lbc = [lbc; $\bar{0}$]; ▷ Bounds for constraints are 0, implying equality
        ubc = [ubc; $\bar{0}$];
    end for
    CnstrnVec={CnstrnVec, $c_{max} - Y_o^{(6,i)}(15..16)$ } ▷ Constrain integral of samp. density to leq. 12
    lbc = [lbc; $\bar{0}$]; ▷ Bounds for constraint is 0, implying equality
    ubc = [ubc; $\bar{0}$];

    FIM=FIM+$Y_o^{(6,i)}(17..37)$ ▷ Sum FIM terms for each sub-exp.
end for
Objective$= -\log(\det(\text{sym}(FIM)))$ ▷ Define the overall objective

'CtrlVec' are individual symbols representing the controls and the shooting initial conditions, which the NLP solver will optimize. The elements of 'CnstrnVec' contain non-linear symbolic expression that evaluate to the constraint functions. The vectors 'lbc' and 'ubc' are the lower and upper bounds for the non-linear constraint functions. If we want two symbolic expressions to be equal, we insert an expression for their difference into 'CnstrnVec', and then set both 'lbc' and 'ubc' to zero to enforce equality. The vectors 'lbw' and 'ubw' likewise constrain the 'CnstrnVec' optimization variable vector to feasible ranges. We start with both 'CtrlVec', 'CnstrnVec', 'lbw', 'ubw', 'lbc' and 'ubc' empty and fill them as we loop over the problem structure.

At line 21 we create a symbol, $\lambda^{(i)}$ for the growth rate control. This is done once for each sub-experiment at the start of the outer loop. We then add it to the control vector, 'CtrlVec', and constrain its range. At line 26 we create a symbol vector for the initial conditions for the sub-experiment. This too is then added to the control vector and constrained to a feasible range. However at line 31, we insert the additional nonlinear constraint that the initial condition must be at steady state (defined by the CasADi function 'SteadyState($\lambda^{(i)}$, $\boldsymbol{\theta}$)', definition not given and must be provided by the user). We enforce equality in the following lines. At line 36 we create an induction control variable $u^{(j,i)}$ and add it to 'CtrlVec', once for each sub-experiment and shooting/induction interval. This also marks the beginning of a shooting interval. In the following lines we constrain the induction to its feasible range. At line 42 we create symbols for the sampling density controls, add them to 'CtrlVec', and then constrain them. At line 47 we use $\bar{\boldsymbol{G}}$ to propagate the symbol for the initial condition, $Y_o^{(j,i)}$, forward, storing it in the same variable. (This does not erase the original contents of $Y_o^{(j,i)}$ from the start of the shooting interval, as those symbols are stored in 'CtrlVec'.) The inner-most loop calls $\bar{\boldsymbol{G}}$ with the corresponding control symbols for the given interval and iteratively advances the state vector. After completion of the inner loop, at line 51, a new shooting initial condition is created and added to 'CtrlVec'. At line 56 we constrain the new initial condition to be equal to the final value of the state vector on the previous shooting interval (the product of the iterated $\bar{\boldsymbol{G}}$), adding it to 'CnstrnVec'. At the end of the sub-experiment loop, line 60, we enforce the integral constraints on the sampling density. At line 64 we add the FIM entries in the final sub-experiment state vector to the running totals across the sub-experiments. Finally, we form the objective expression 'Objective', which contains a (very large) symbolic expression for the objective of the entire experiment, in line 66.

To improved numerical stability, we compute the determinant of the Fisher information matrix using QR factorization: $\mathcal{I}_{Tot} = QR$. The entries in 'FIM' are the unique values of $\mathcal{I}_{Tot}$. The function 'sym()' in Algorithm 5 reforms the complete $\mathcal{I}_{Tot}$ from the vector 'FIM'. Because $\mathcal{I}_{Tot}$ is positive semi-definite $\det(\mathcal{I}_{Tot}) \geq 0$. Further

$$|\det(\mathcal{I}_{Tot})| = |\det(Q)||\det(R)| \tag{28}$$

The factorization is such that $|\det(Q)| = 1$. Because R is an upper triangular matrix, the determinant is the product of its diagonal entries:

$$\det(\mathcal{I}_{Tot}) = \prod_m R_{(m,m)} \tag{29}$$

and so

$$-\ln(\Theta_D(\mathcal{I}_{Tot})) = -\ln(\det(\mathcal{I}_{Tot})) = -\sum_m \ln(R_{(m,m)}) \tag{30}$$

The expression in 'Objective' is a mathematical function of the symbols listed in 'CtrlVec'. Likewise, the vector of expressions in 'CnstrnVec' are also mathematical functions of the symbols in 'CtrlVec'. Because these functions are symbolic, they can be differentiated with respect to any (or all) of the entries in 'CtrlVec' (or the parameter vector $\boldsymbol{\theta}$). This property allows CasADi to automatically generate Jacobians and Hessians for the objective and the constraints when passing the problem to IPOPT.

Although generation of the derivatives is automated once the symbolic expression is constructed, choosing a problem structure that achieves maximal sparsity is critical. The degree of sparsity in the Hessian and Jacobians make a significant difference in the computation time. Once the the symbolic expressions for the OED problem have been created, passing them to IPOPT is straightforward using CasADi's interface. Algorithm 6 shows the creation of the solver and its call in CasADi's MATLAB interface. Starting the solver requires an initial guess for the control vector. We generate this by simulating one of the null experiments and storing the state variables and controls at the appropriate times to construct 'CtrlVec'.

Algorithm 6 Calling IPOPT

```
prob = struct( Objective, CtrlVec, CnstrnVec)   ▷ Package symbol vectors for passage to IPOPT
solver = nlpsol( 'ipopt', prob)                 ▷ Create a solver instance
solver( CtrlVec_o, lbw, ubw, lbc, ubc)          ▷ Call solver with initial guess; CtrlVec_o, pass upper/lower bounds
```

Parameter Estimation: We also implemented our weighted least-squares parameter estimation algorithm in CasADi. This was also implemented as a multiple-shooting algorithm and was structured in a similar manner to the OED algorithm above, where the parameters are treated as time-constant controls in an optimal control problem [33,34]. The weights in our fitting algorithm were taken as the inverse of the sampling variances, $\sigma^2_{rna} = (0.05)X_{rna}$ and $\sigma^2_{prot} = (0.05)X_{prot}$. In our numerical fitting experiments, for each experimental design (null, null variants, optimal and perturbed optimal), we initialized the parameter estimation algorithm to a random parameter vector drawn uniformly from the feasible parameter range. For each experimental design, a small subset of the 30 fittings either did not converge (max number of iterations or other stopping condition was reached) or converged to clearly erroneous estimates (relative error exceeding several orders of magnitude). We removed these outliers before computing covariances. The number of outliers in each design were as follows: null experiment, 3; growth variant, 2; sampling variant, 1; induction variant, 3; true optimal, 0; perturbed optimal, 0 (for all six).

Timing: Our OED algorithm normally took between 70 and 400 iterations to converge in IPOPT (depending on the number and range of the parameters). The wall-clock time was on the order of an hour. Our parameter estimating algorithm normally took between 10 and 70 iterations to converge. The wall-clock timing was on the order of 10s of minutes. All experiments were done on a Mac mini machine with a 2.6 GHz Intel Core i5 and 16 GB of RAM.

References

1. Bremer, H.; Dennis, P. Modulation of Chemical Composition and Other Parameters of the Cell at Different Exponential Growth Rates. *EcoSal Plus* **2008**, *3*.
2. Bakshi, S.; Dalrymple, R.M.; Li, W.; Choi, H.; Weisshaar, J.C. Partitioning of RNA polymerase activity in live Escherichia coli from analysis of single-molecule diffusive trajectories. *Biophysical journal* **2013**, *105*, 2676–2686.
3. Patrick, M.; Dennis, P.P.; Ehrenberg, M.; Bremer, H. Free RNA polymerase in Escherichia coli. *Biochimie* **2015**, *119*, 80–91.
4. Klumpp, S.; Hwa, T. Growth-rate-dependent partitioning of RNA polymerases in bacteria. *Proceedings of the National Academy of Sciences* **2008**, *105*, 20245–20250.
5. Bremer, H.; Dennis, P.; Ehrenberg, M. Free RNA polymerase and modeling global transcription in Escherichia coli. *Biochimie* **2003**, *85*, 597–609.
6. Stracy, M.; Lesterlin, C.; De Leon, F.G.; Uphoff, S.; Zawadzki, P.; Kapanidis, A.N. Live-cell superresolution microscopy reveals the organization of RNA polymerase in the bacterial nucleoid. *Proceedings of the National Academy of Sciences* **2015**, *112*, E4390–E4399.
7. Endesfelder, U.; Finan, K.; Holden, S.J.; Cook, P.R.; Kapanidis, A.N.; Heilemann, M. Multiscale spatial organization of RNA polymerase in Escherichia coli. *Biophysical journal* **2013**, *105*, 172–181.

8. Klumpp, S.; Zhang, Z.; Hwa, T. Growth rate-dependent global effects on gene expression in bacteria. *Cell* **2009**, *139*, 1366–1375.
9. de Leon, F.G.; Sellars, L.; Stracy, M.; Busby, S.J.; Kapanidis, A.N. Tracking low-copy transcription factors in living bacteria: the case of the lac repressor. *Biophysical journal* **2017**, *112*, 1316–1327.
10. Liang, S.T.; Bipatnath, M.; Xu, Y.C.; Chen, S.L.; Dennis, P.; Ehrenberg, M.; Bremer, H. Activities of constitutive promoters in Escherichia coli1. *Journal of molecular biology* **1999**, *292*, 19–37.
11. Bintu, L.; Buchler, N.E.; Garcia, H.G.; Gerland, U.; Hwa, T.; Kondev, J.; Phillips, R. Transcriptional regulation by the numbers: models. *Current opinion in genetics & development* **2005**, *15*, 116–124.
12. Rydenfelt, M.; Cox III, R.S.; Garcia, H.; Phillips, R. Statistical mechanical model of coupled transcription from multiple promoters due to transcription factor titration. *Physical Review E* **2014**, *89*, 012702.
13. Phillips, R. Napoleon is in equilibrium. *Annu. Rev. Condens. Matter Phys.* **2015**, *6*, 85–111.
14. Häkkinen, A.; Ribeiro, A.S. Characterizing rate limiting steps in transcription from RNA production times in live cells. *Bioinformatics* **2015**, *32*, 1346–1352.
15. Muthukrishnan, A.B.; Kandhavelu, M.; Lloyd-Price, J.; Kudasov, F.; Chowdhury, S.; Yli-Harja, O.; Ribeiro, A.S. Dynamics of transcription driven by the tetA promoter, one event at a time, in live Escherichia coli cells. *Nucleic acids research* **2012**, *40*, 8472–8483.
16. Kandhavelu, M.; Mannerström, H.; Gupta, A.; Häkkinen, A.; Lloyd-Price, J.; Yli-Harja, O.; Ribeiro, A.S. In vivo kinetics of transcription initiation of the lar promoter in Escherichia coli. Evidence for a sequential mechanism with two rate-limiting steps. *BMC systems biology* **2011**, *5*, 149.
17. Scott, M.; Gunderson, C.W.; Mateescu, E.M.; Zhang, Z.; Hwa, T. Interdependence of cell growth and gene expression: origins and consequences. *Science* **2010**, *330*, 1099–1102.
18. Liang, S.T.; Xu, Y.C.; Dennis, P.; Bremer, H. mRNA composition and control of bacterial gene expression. *Journal of Bacteriology* **2000**, *182*, 3037–3044.
19. Dai, X.; Zhu, M.; Warren, M.; Balakrishnan, R.; Patsalo, V.; Okano, H.; Williamson, J.R.; Fredrick, K.; Wang, Y.P.; Hwa, T. Reduction of translating ribosomes enables Escherichia coli to maintain elongation rates during slow growth. *Nature microbiology* **2017**, *2*, 16231.
20. Borkowski, O.; Goelzer, A.; Schaffer, M.; Calabre, M.; Mäder, U.; Aymerich, S.; Jules, M.; Fromion, V. Translation elicits a growth rate-dependent, genome-wide, differential protein production in Bacillus subtilis. *Molecular systems biology* **2016**, *12*, 870.
21. Kudla, G.; Murray, A.W.; Tollervey, D.; Plotkin, J.B. Coding-sequence determinants of gene expression in Escherichia coli. *science* **2009**, *324*, 255–258.
22. Laursen, B.S.; Sørensen, H.P.; Mortensen, K.K.; Sperling-Petersen, H.U. Initiation of protein synthesis in bacteria. *Microbiology and molecular biology reviews* **2005**, *69*, 101–123.
23. Betts, J.T. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics* **1998**, *21*, 193–207.
24. Kelly, M. An introduction to trajectory optimization: how to do your own direct collocation. *SIAM Review* **2017**, *59*, 849–904.
25. Diedam, H.; Sager, S. Global optimal control with the direct multiple shooting method. *Optimal Control Applications and Methods* **2018**, *39*, 449–470.
26. Andersson, J.A.E.; Gillis, J.; Horn, G.; Rawlings, J.B.; Diehl, M. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation* **In Press, 2018**.
27. Janka, D.; Körkel, S.; Bock, H.G. Direct multiple shooting for nonlinear optimum experimental design. In *Multiple Shooting and Time Domain Decomposition Methods*; Springer, 2015; pp. 115–141.
28. Telen, D.; Vercammen, D.; Logist, F.; Van Impe, J. Robustifying optimal experiment design for nonlinear, dynamic (bio) chemical systems. *Computers & Chemical Engineering* **2014**, *71*, 415–425.
29. Hoang, M.; Barz, T.; Merchan, V.; Biegler, L.; Arellano-Garcia, H. Simultaneous solution approach to model-based experimental design. *AIChE Journal* **2013**, *59*, 4169–4183.
30. LLC, M. Andersseon, Joel and Gillis, Joris and Diehl, Mortiz, 2018.
31. Andersseon, J.; Gillis, J.; Horn, G. CasADi Examples Package, 2018.
32. Hindmarsh, A.C.; Brown, P.N.; Grant, K.E.; Lee, S.L.; Serban, R.; Shumaker, D.E.; Woodward, C.S. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)* **2005**, *31*, 363–396.

282 33. Bock, H.G.; Kostina, E.; Schlöder, J.P. Numerical methods for parameter estimation in nonlinear differential
283 algebraic equations. *GAMM-Mitteilungen* **2007**, *30*, 376–408.
284 34. Bock, H.G.; Körkel, S.; Schlöder, J.P. Parameter estimation and optimum experimental design for differential
285 equation models. In *Model based parameter estimation*; Springer, 2013; pp. 1–30.