Gaussian process emulation of an individual-based model simulation of microbial communities

Oluwole Oyebamiji
School of Mathematics, Statistics and Physics,
Newcastle University, UK

February 11, 2018

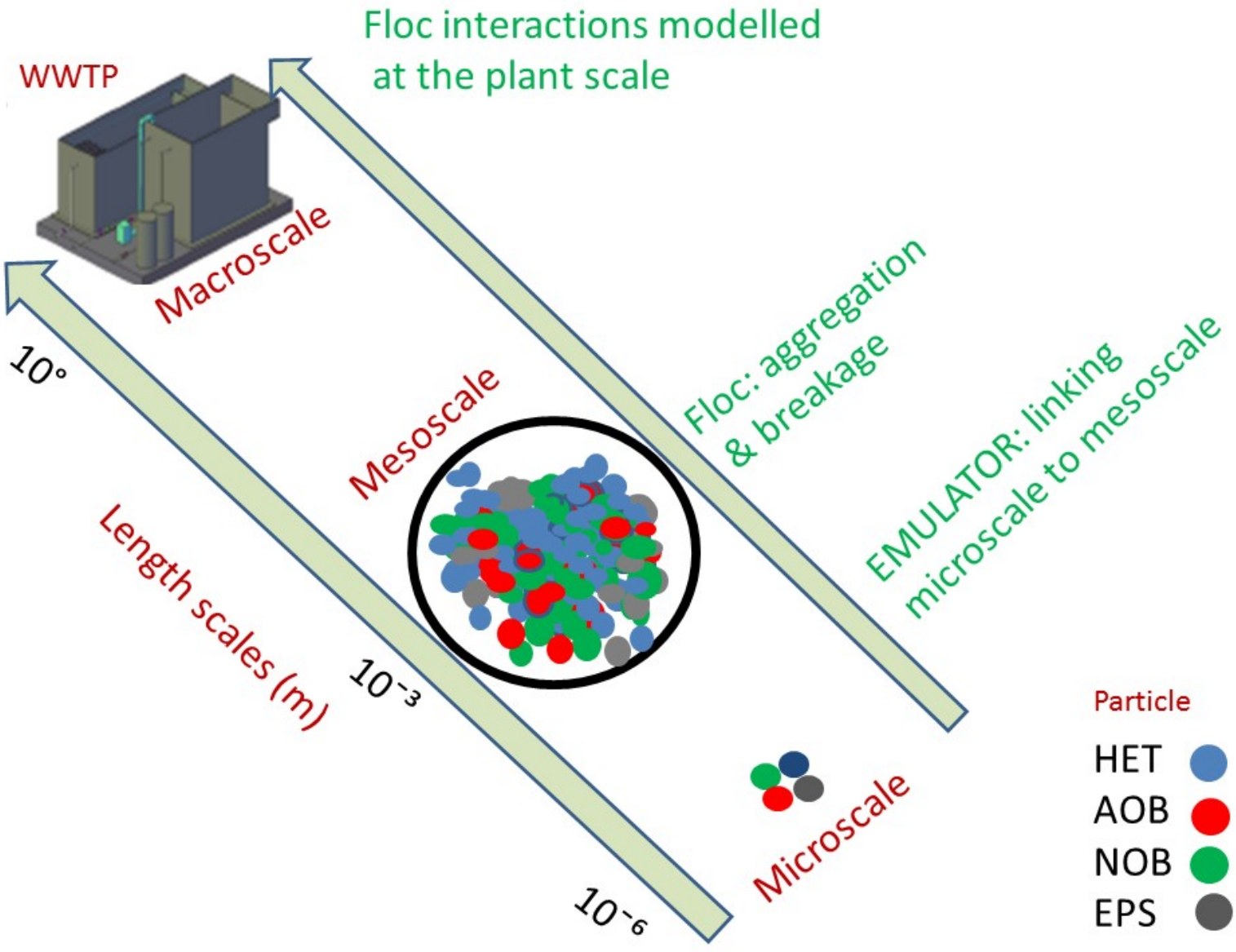
Aims of talk

- Key attributes of NUFEB biological model
- Procedure for building an emulator
- Gaussian process emulator
- NUFEB model outputs for emulation
- Results and conclusion

Conceptual framework

- Concerned with the multi-scale stochastic simulation of complex biological communities in an open environmental system, with a view to using simulation for designing engineering interventions to improve performance
- Wastewater treatment plants (WWTPs) are an ideal example of a multi-scale open engineered biological system
- $\approx 10^{18}$ individual bacteria in a typical plant
- Macro-scale characteristics of WWTPs are the consequence of micro-scale features of a vast number of individual bacteria acting together in complex ecological communities
- Model and understand the interaction of microbes at a fine scale to accurately capture macroscale responses
- Transfer this fine-scale information to the engineered macroscale process in a computationally efficient way

Schematic of different length scales for multiscale modelling of WWTP



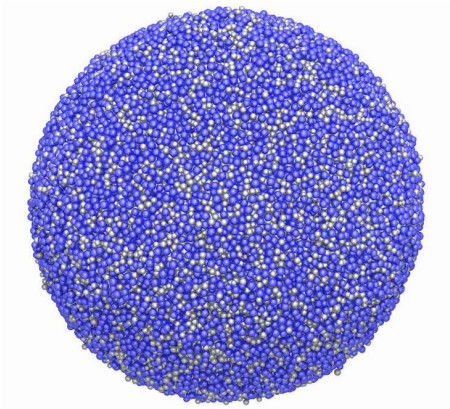
Biological modelling

- Each bacterial cell is an agent within an individual based model of cell growth, division, death and mechanical interactions
- First order growth kinetics, solved numerically using an explicit Euler scheme
- Growth rates depend on the nutrient environment of the agent, and nutrients are taken up from the fluid during growth
- Cell division occurs when cells reach a critical size stochastic elements in the division process excretes as a discrete particle/agent (placed randomly)
- The model can run up to order 10^6 bacteria
- A long way off the 10^{18} bacteria in a typical full-scale WWTP

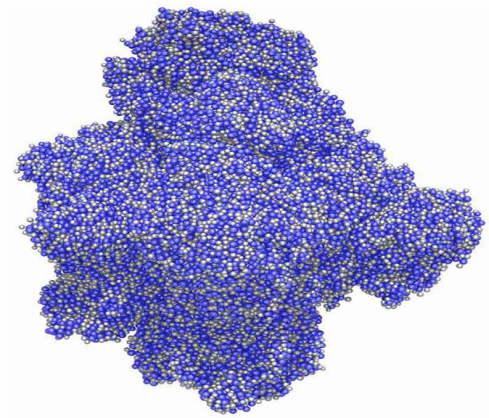
Emulation of complex computer models

- The micro-scale model is very computationally intensive approx 5 CPU hours to simulate 24 real hours
- There now exists a large literature on the use of statistical model emulators in the design and analysis of computer experiments
- Emulators essentially provide a fast approximate surrogate for the true model, with properly quantified uncertainty
- To build a good emulator, we want residual uncertainty to be small. In 1d this is easy, but in higher dimensions we need to choose design points to fill space efficiently so that there arent big gaps in parameter space for which we dont have a simulator run
- Latin hypercube designs (LHDs) are a good way to choose design points to fill space in an efficient way

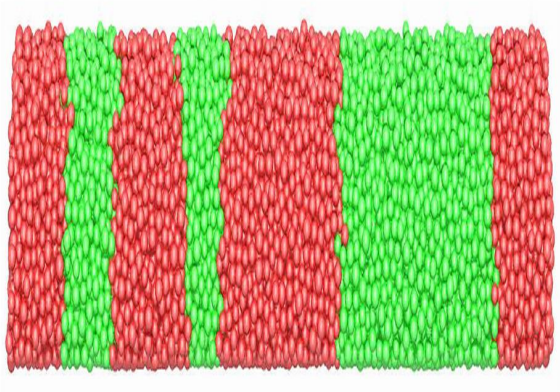
Flocs and biofilms growing under different nutrient conditions



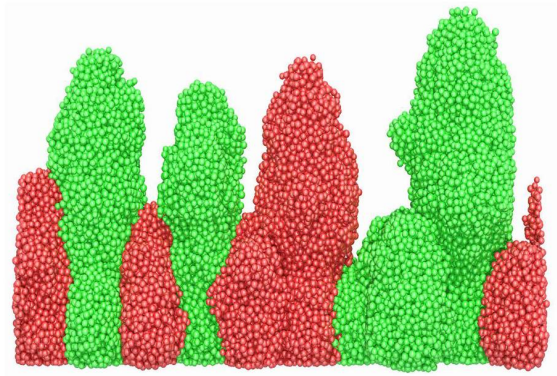
(a) Smooth surface floc: $\lim \Delta \to \infty$



(b) Rough surface floc: $\lim \Delta \to 0$



(c) Smooth surface biofilms: $\lim \Delta \to \infty$



(d) Rough surface biofilms: $\lim \Delta \to 0$

Gaussian process emulator

- A GP is a probability distribution on functions defined so that the marginal distribution of any finite number of points always has a multivariate normal (MVN) distribution
- Points close together in input space are typically more highly correlated than points far away
- Stationary GPs are defined by a covariance function many different possible choices an exponential kernel

$$Cov[f(x), f(x')] = K(x, x') = \sigma^2 \exp\{-(x-x')^T R(x-x')\} + \delta I, \tag{1}$$

where R is a diagonal matrix of correlation or scale hyperparameters. It determines how fast the spatial correlation decays throughout the input space. $\delta \geq 0$ is the nugget parameter and I is an indicator function which is 1 if $x = x'$ and 0 otherwise.

Emulation of microscale model

- Emulate important univariate summary statistics arising from the analysis of the raw computer model output (eg. Floc diameter, floc mass, biofilm height, surface roughness etc) based on numerous inputs (covariates)
- Consider output y for input x as a deterministic function

$$y = f(x). \tag{2}$$

- Outputs y_i at n design points x_i statistically modelled as

$$y \sim N(HB, A) \tag{3}$$

where H is a design matrix where the i^{th} row of H is a (deterministic) function of x_i, and A is the matrix of covariances determined by the exponential kernel described earlier and B is a vector of regression coefficients.

Multivariate emulation

- Univariate emulation ignores correlation between multiple outputs
- Often better to jointly model
- Covariance matrix for p outputs a $p \times p$ matrix Σ
- Assuming separability of the two covariance matrices, we get a matrix normal distribution for $n \times p$ output matrix Y

$$Y \sim MN(HB, A, \Sigma) \tag{4}$$

- ie. $Var(vec\ Y) = \Sigma A$

Dynamic emulation

- Our simulation model is a time evolving dynamical system
- Often desirable to emulate dynamical behaviour over time steps (much larger than the simulation time steps)
- Regard the simulator as a dynamic function

$$y_t = f(x_t, y_{t-1}), \tag{5}$$

where y_t is a state vector and x_t represents the model inputs at time t

- Statistically model and emulate f (,)

Dynamic emulation

- Construct a single step emulator $\mathbf{y}_1 = f(\mathbf{x}_1, \mathbf{y}_0)$ using a GP
- Proceed sequentially, feed back the entire output distribution from the GP model, such that at time step $t = 1$, for input $(\mathbf{x}_1, \mathbf{y}_0)$
- Sample from the distribution of $f(\mathbf{y}_0, \mathbf{x}_1)$, the model output is given as

$$\tilde{\mathbf{y}}_1^{(s)} \sim N\Big(\mu^{\bullet}(\mathbf{x}_1, \mathbf{y}_0), \mathbf{K}^{\bullet}(\mathbf{x}_1, \mathbf{y}_0)\Big)$$

- For time $t = 2$, the input data $\mathbf{x}_2$ is augmented by complete distribution $\mathbf{y}_1^{(s)}$ such that $\mathbf{X}_2 = [(\mathbf{x}_2, \tilde{\mathbf{y}}_1)]^T$
- Generate samples from the distribution of $f(\tilde{\mathbf{y}}_1^{(s)}, \mathbf{x}_2)$ and denote as $\tilde{\mathbf{y}}_2^{(s)}$
- Repeat until $T - 1$ steps are reached and rebuild the emulator

Dynamic emulation.....

$$\begin{pmatrix} \text{Original inputs} \\ \vdots \\ (\mathbf{y}_0, \mathbf{x}_1) \\ (\tilde{\mathbf{y}}_1, \mathbf{x}_2) \\ \vdots \\ (\tilde{\mathbf{y}}_{T-1}, \mathbf{x}_T) \end{pmatrix} = \begin{pmatrix} \text{Original outputs} \\ \vdots \\ \tilde{\mathbf{y}}_1^{(s)} \\ \tilde{\mathbf{y}}_2^{(s)} \\ \vdots \\ \tilde{\mathbf{y}}_T^{(s)} \end{pmatrix}.$$

- Repeat the entire process many times to obtain $\tilde{\mathbf{Y}}^N = \left[\tilde{\mathbf{y}}_1^{(s)}, \ldots, \tilde{\mathbf{y}}_{T-1}^{(s)}\right]^N$, for $s = 1, \ldots, N$
- $\tilde{\mathbf{Y}}^N$ is a sample from the joint distribution of $[\mathbf{y}_1, \ldots, \mathbf{y}_{T-1}]$ given the emulator training, N is the number of Monte Carlo (MC) sample

Normal approximation

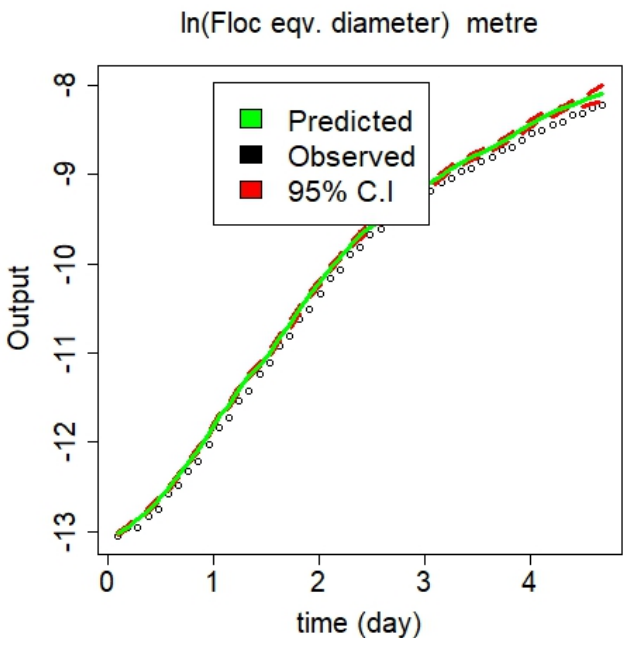
Follow Conti et al.(2009), the marginal distribution of $\mathbf{y}_t$ can be approximated as $\mathbf{y}_t \sim N\Big(\mu_t(.), \mathbf{K}_t(.)\Big)$ MC sampling to repeatedly revise the mean and variance

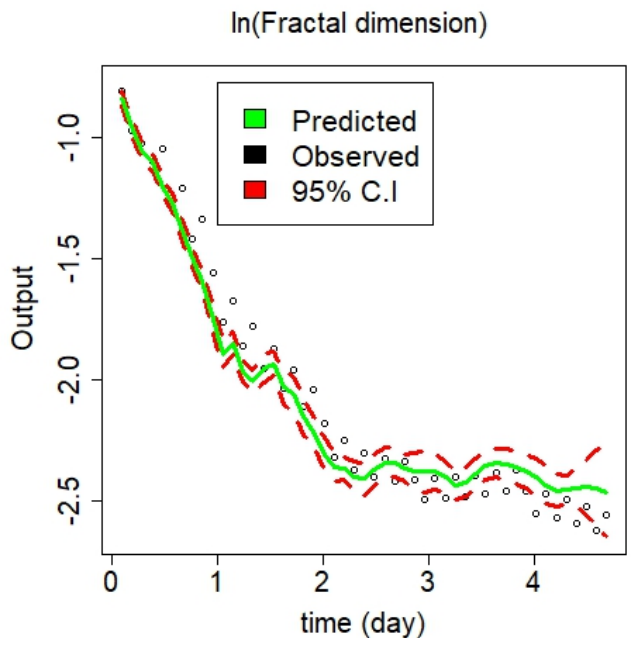
$$\hat{\mu}_{t+1} = \frac{1}{N}\sum_{s=1}^{N}\Big(\mu(\tilde{\mathbf{y}}_t^{(s)}, x_{t+1})|f(\mathbf{y})\Big),$$

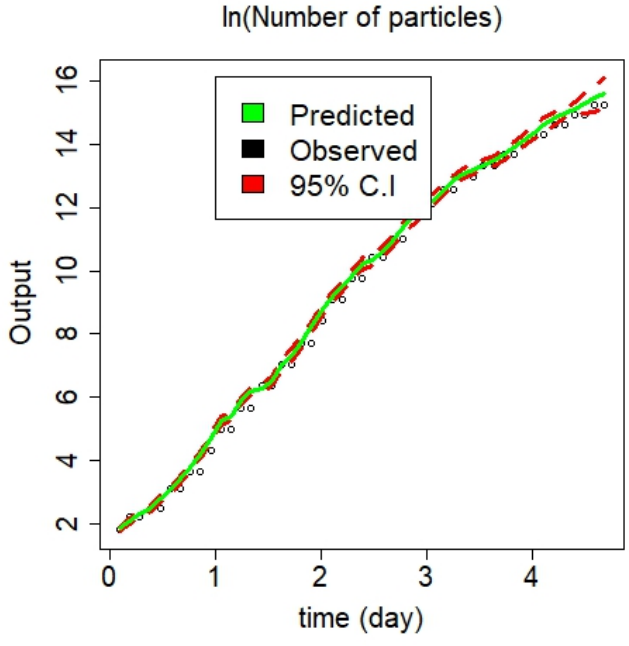
$$\widehat{\mathbf{K}}_{t+1} = \frac{1}{N}\sum_{s=1}^{N}\Big(\mathbf{K}(x_{t+1}, \tilde{\mathbf{y}}_t^{(s)}), (x_{t+1}, y_t)|f(\mathbf{y})\Big) + \frac{1}{N}\sum_{s=1}^{N}\Big(\mu(\tilde{\mathbf{y}}_t^{(s)}, x_{t+1})|f(\mathbf{y})\Big)^2,$$

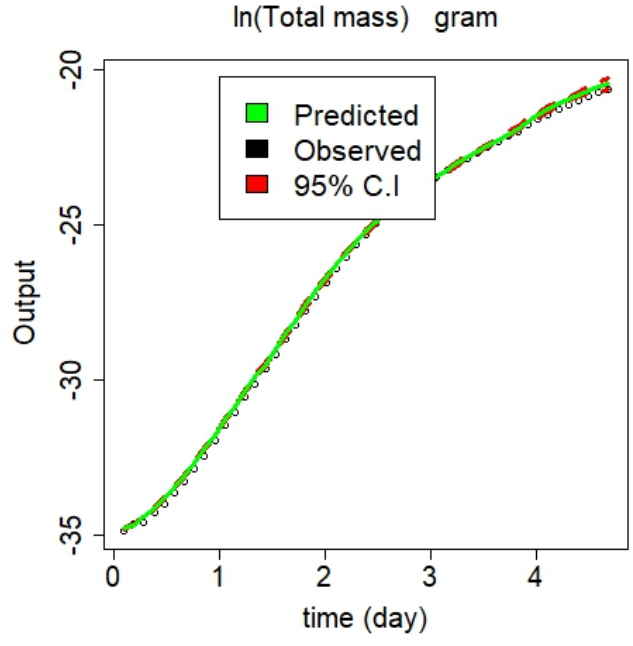
where $\tilde{\mathbf{y}}_t^{(s)}$ is a sample from $N\Big(\mu_t(.), \mathbf{K}_t(.)\Big)$.

Result comparison









Conclusion

- Emulator gives results almost instantaneously (~ 1 minute)
- This is 220-fold increase in computational efficiency


Oyebamiji OK, Wilkinson DJ, Jayathilake PG, Curtis TP, Rushton SP, Li B, Gupta P (2017). Gaussian process emulation of an individual-based model simulation of microbial communities. *Journal of Computational Science*, $1; 22 : 69 - 84$.


Jayathilake PG, Gupta P, Li B, Madsen C, Oyebamiji O, Gonzlez-CabaleiroR, Rushton S, Bridgens B, Swailes D, Allen B, McGough AS (2017). A mechanistic Individual-based Model of microbial communities. *PloS one*, $12(8) : e0181965$.


Conti, S., Gosling, J. P., Oakley, J. E., & O'Hagan, A. (2009). Gaussian process emulation of dynamic computer codes. *Biometrika*, $96(3), 663676$.