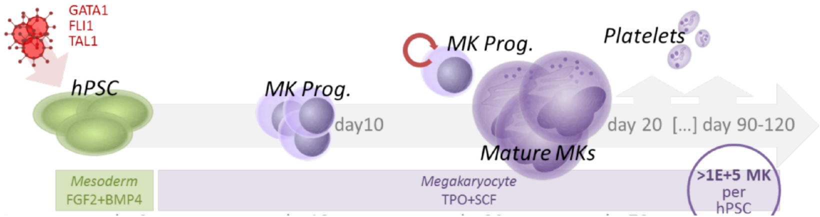
Improving protocols for cell differentiation through reinforcement learning

Lorenz Wernisch, John Reid, Cedric Ghevaert, Thomas Moreau

MRC-Biostatistics Unit and Cambridge Blood Centre
Cambridge, UK

Warsaw, 27 Sep 2018

Stem cells to Megakaryocytes

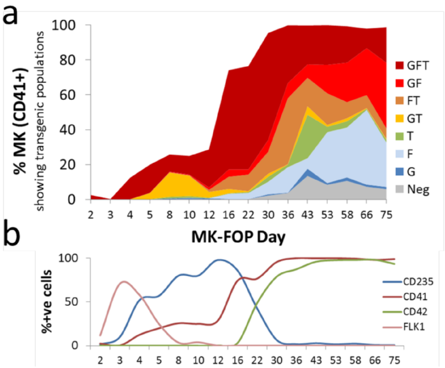


Human pluriopotent stem cells to Megakaryocytes via induction of genes GATA1, FLI1, TAL1

Increase yield and maturity of MK cells

Problem: When and how much induction is required

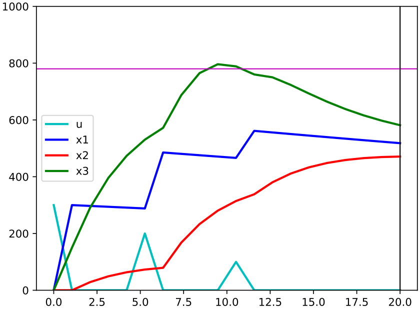
Flow cytometry measurements



Cell surface markers and gene transcription profile indicate maturity

Yield of MK cells from cell cultures

Toy abstraction of problem

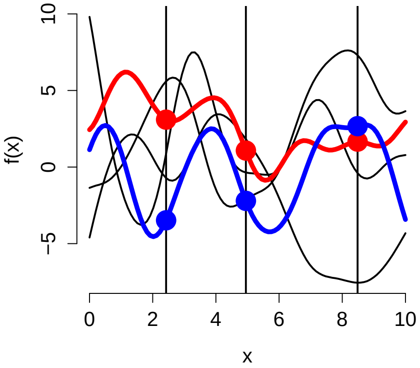


System of three variables $x_{1,t}, x_{2,t}, x_{3,t}$ measured daily over 20 days

We control input u_t to push $x_{3,20}$ (green) to a target value on day 20

$$\begin{pmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \end{pmatrix} = \begin{pmatrix} x_{1,t-1} \\ x_{2,t-1} \\ x_{3,t-1} \end{pmatrix} + \begin{pmatrix} f_1(x_{1,t-1}, x_{2,t-1}, x_{3,t-1}, u_{t-1}) \\ f_2(x_{1,t-1}, x_{2,t-1}, x_{3,t-1}, u_{t-1}) \\ f_3(x_{1,t-1}, x_{2,t-1}, x_{3,t-1}, u_{t-1}) \end{pmatrix}$$

Gaussian process prior



Family of functions via covariance K on input points x

$y \sim N(0, K_{xx})$

Prediction for x^* from (x, y)

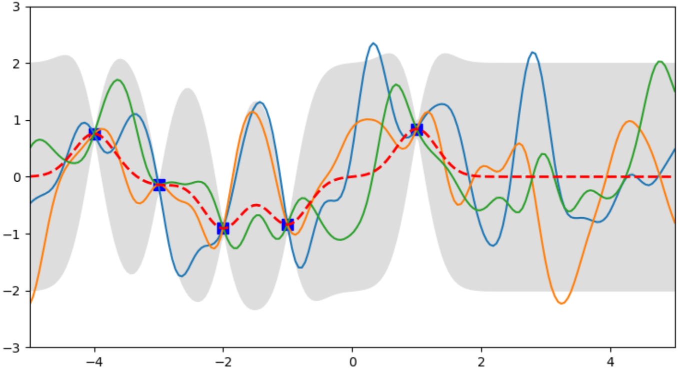
$y^* \sim N(K_{x^*x} K_{xx}^{-1} y, \Sigma)$

$\Sigma = K_{x^*x^*} - K_{x^*x} K_{xx}^{-1} K_{xx^*}$

Gaussian $\mathrm{cov}(x, x^*) = \theta \exp((x - x^*)^2 / \ell)$

Matern $\mathrm{cov}(x, x^*) = \theta(1 + |x - x^*|/\ell) \exp(-\theta_2 |x - x^*|/\ell)$

GP uncertainty and samples



Regions of uncertainty, amount controlled by θ

Smoothness controlled by lengthscale ℓ

Estimate by ML or MAP

Multidimensional input

For $x, x^* \in R^m$

$$\text{cov}_G(x, x^*) = \theta \exp(-\sum (x_d - x_d^*)^2/\ell_{G,d})$$

with Gaussian (nonlinear) relevance parameters $1/\ell_{G,1}, \dots, 1/\ell_{G,m}$

Linear covariance part

$$\text{cov}_L(x, x^*) = \sum_d x_d^2/\ell_{L,d}$$

with linear relevance parameters $1/\ell_{L,1}, \dots, 1/\ell_{L,m}$

$$\text{cov}(x, x^*) = \mu + \text{cov}_L(x, x^*) + \text{cov}_G(x, x^*) + \sigma^2 I(x = x^*)$$

Automatic relevance determination (ARD)

Maximize marginal log likelihood for data x, y:

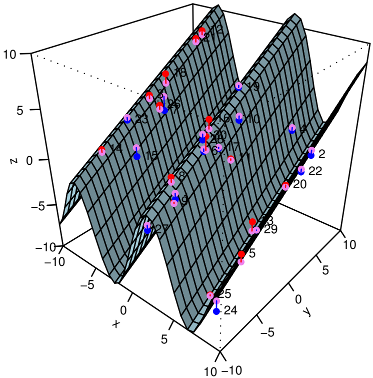
$$\mathrm{argmax}_{\theta,\ell,\sigma^2} - y^T K_{xx}(\theta)^{-1} y - \log |K_{xx}(\theta)|$$

Relevance $1/\ell_d \to 0$, ie $\ell_d \to \infty$

GP mean becomes flat in input dimension x_d: x_d has no influence on y it is **irrelevant**

Often suitable priors on ℓ, σ^2 required to achieve ARD sparsity effect

Automatic relevance determination



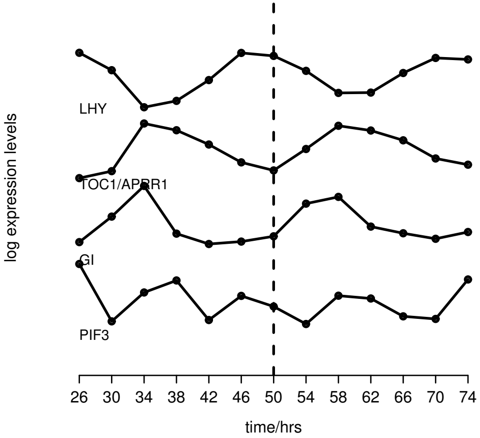
Relevance parameters $1/\ell_d$:

	x_1	x_2	x_3
nonlinear	0.21	0	0
linear	0	0.35	0

estimated σ 0.92

30 data point generated from with
$f(x_1, x_2, x_3) = 5\sin(0.7x_1) + 0.5x_2 + \epsilon$
where $\epsilon \sim N(0, 1)$

Circadian clock in A. thaliana



Constant light:
13 time points every 4 hours from 26 to 74 hrs

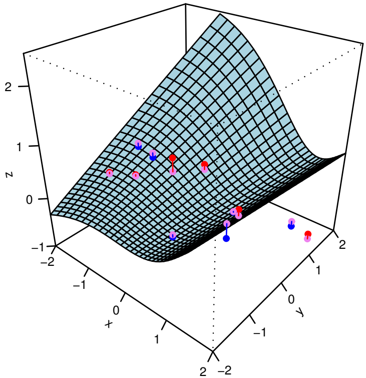
Optimise GP lenghtscale parameters

GP autoregression

$$\begin{pmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \\ x_{4,t} \end{pmatrix} = \begin{pmatrix} x_{1,t-1} \\ x_{2,t-1} \\ x_{3,t-1} \\ x_{4,t-1} \end{pmatrix} + \begin{pmatrix} f_1(x_{1,t-1}, x_{2,t-1}, x_{3,t-1}, x_{4,t-1}) \\ f_2(x_{1,t-1}, x_{2,t-1}, x_{3,t-1}, x_{4,t-1}) \\ f_3(x_{1,t-1}, x_{2,t-1}, x_{3,t-1}, x_{4,t-1}) \\ f_4(x_{1,t-1}, x_{2,t-1}, x_{3,t-1}, x_{4,t-1}) \end{pmatrix}$$

GP with constant, linear, Gaussian and noise component

Gene network: LHY regulation

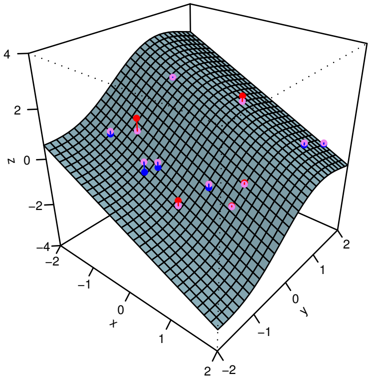


	LHY	TOC1	GI	PIF3
nlin	0.01	0.01	0.78	0.01
lin	0.81	1.13	0.45	0

No dependence of LHY on PIF3

Nonlinear dependency of LHY on TOC1 and GI

Gene network: GI regulation

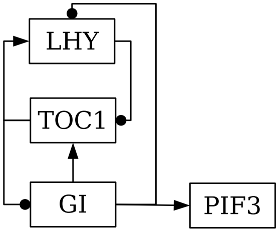


	LHY	TOC1	GI	PIF3
nlin	0.01	0	0.78	0
lin	0	0.82	0.17	0

No dependence of GI on LHY and PIF3

Linear (negative) dependency of GI on TOC1
Nonlinear (positive) dependency of GI on itself

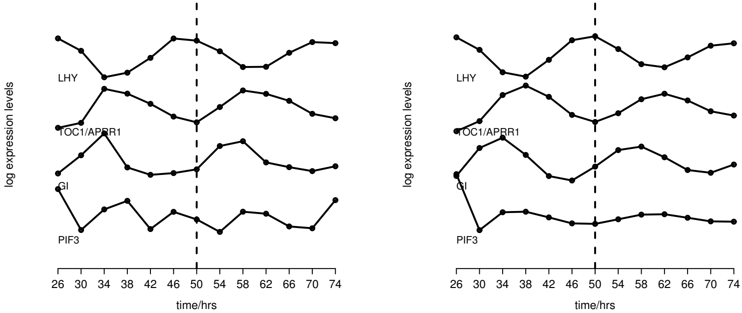
Circadian clock network



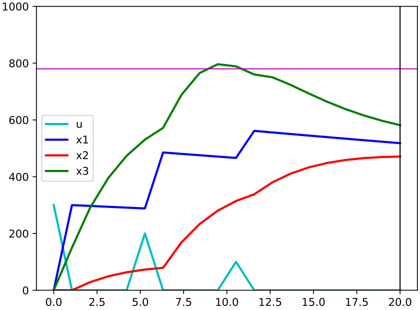
LHY in negative feedback with TOC1

Second negative feedback loop involving GI

In agreement with current knowledge



Control problem



System of three variables $x_{1,t}, x_{2,t}, x_{3,t}$ measured daily over 20 days

We control input u_t to push $x_{3,20}$ to a target value on day 20

$$\begin{pmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \end{pmatrix} = \begin{pmatrix} x_{1,t-1} \\ x_{2,t-1} \\ x_{3,t-1} \end{pmatrix} + \begin{pmatrix} f_1(x_{1,t-1}, x_{2,t-1}, x_{3,t-1}, u_{t-1}) \\ f_2(x_{1,t-1}, x_{2,t-1}, x_{3,t-1}, u_{t-1}) \\ f_3(x_{1,t-1}, x_{2,t-1}, x_{3,t-1}, u_{t-1}) \end{pmatrix}$$

Dynamical system via Gaussian processes

Idea: use data to approximate the difference

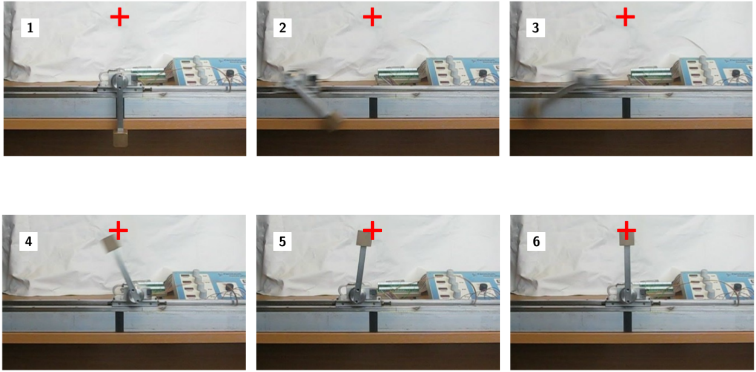
$$x_{i,t} - x_{i,t-1} = f_i(x_{1,t-1}, \ldots, x_{d,t-1}, u_{t-1})$$

via d GPs $f_i \sim GP(\mu(x,u), \Sigma(x,u))$, $i = 1, \ldots, d$

Iterative cycle for *optimising control* towards target:

- Approximate response x to u by GPs
- Optimize control input u towards target using GP approximation
- Acquire new data applying control to real system

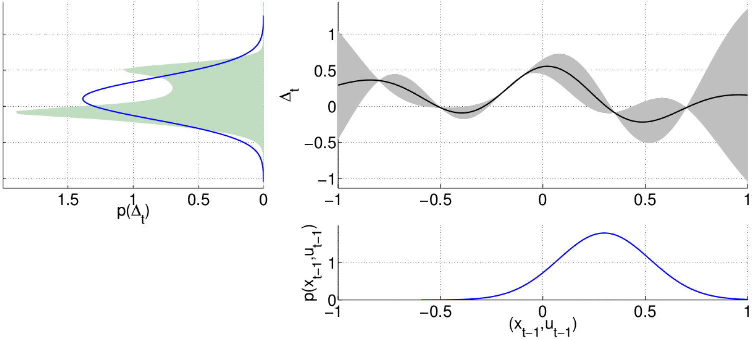
Pilco: probabilistic learning of control



Learn to move cart left/right to swing up pole

Deisenroth and Rasmussen, 2011: PILCO: A Model-Based and Data-Efficient Approach to Policy Search (Probabilistic Inference for Learning COntrol)

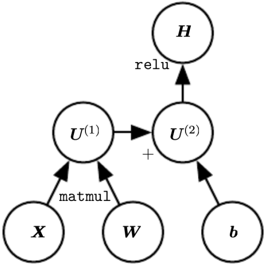
Transmitting uncertainty



(Gaussian) uncertainty in *inputs* results in (non-Gaussian) distribution in output

Pilco: approximate by Gaussian via moment matching

Alternative: dataflow graphs



Tensorflow works with *stateful dataflow graphs*

Data flow between nodes in a directed graph

Nodes represent eg: arithmetic operations, control clauses (if else), matrix manipulations, random number generators

Automatic differentiation

Optimize u using GP approximation

Minimize cost $c(u)$, eg for trajectory $x_3(u_0, \ldots, u_{19})$

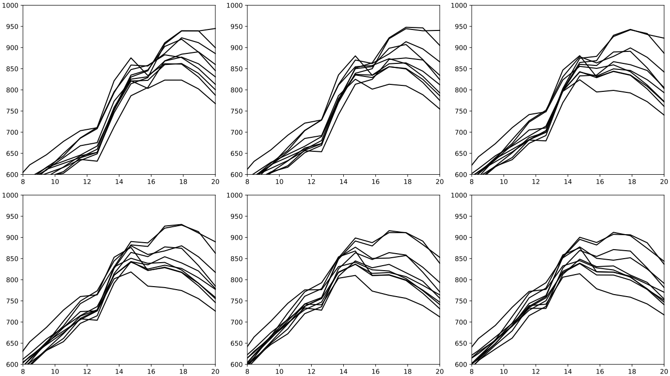
$$c(u) = (x_{3,20}(u_0, \ldots, u_{19}) - x_{\mathrm{target}})^2 + \lambda \sum_t |u_t|$$

How to define trajectory using GPs?

Bad idea: use means of GP for $f_i(x_{t-1}, u_{t-1})$

Better idea: sample several *random* trajectories using GP uncertainty and optimise eg mean

Random trajectories



Expected loss $C(u) = 1/m \sum_k c(x^{(k)})$

Reparametrisation trick $p(x) = g(u, \epsilon)$:
$\nabla_u C(u) = 1/m \sum_k \nabla_u c(g(u, \epsilon_k))$
for a fixed sample $\epsilon_k \sim N(0, I)$

Reparameterisation for Gaussian

$p_N(z \mid \mu(u), \Sigma(u))$

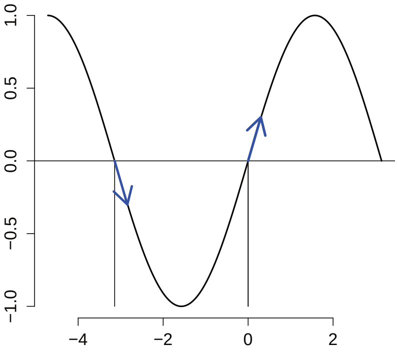
Choleski factorisation $\Sigma(u) = C(u)C(u)^T$

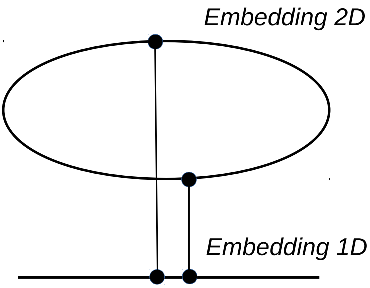
With $\epsilon \sim N(0, I)$

$$z(u) = g(\epsilon, \mu(u), \Sigma(u)) = \mu(u) + C(u)\epsilon$$

$\mu(u), C(u)$ from GPs trained on data and test input u

Key model parameter: latent dimension





$\dot{y} = f(y)$ or $x_t = x_{t-1} + f(t_{t-1})$ not representable in 1D:

Identical y mapped to different $f(y)$

Takens' theorem

Observe one variable $y(1), y(2), y(3), \ldots, y(L)$ from dynamical system

Find lag d with (eg by small autocorrelation)

Choose m and form delay embedding (eg $d = 2$, $m = 3$):

$$\begin{pmatrix} y(1) \\ y(3) \\ y(5) \end{pmatrix}, \begin{pmatrix} y(2) \\ y(4) \\ y(6) \end{pmatrix}, \begin{pmatrix} y(3) \\ y(5) \\ y(7) \end{pmatrix}, \ldots$$

Choose m with few "false" neighbors (monitor distance increase when adding $m + 1$st expansion)

Rules of the game

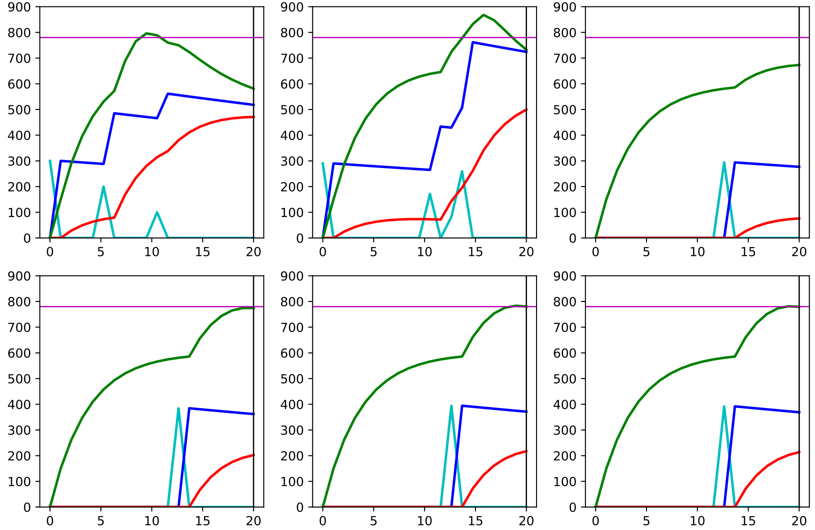
Given: black box dynamical system, cost function $c(x(u))$ for input u to the system

Choose some initial input $u = (u_0, \ldots, u_{T-1})$, iterate

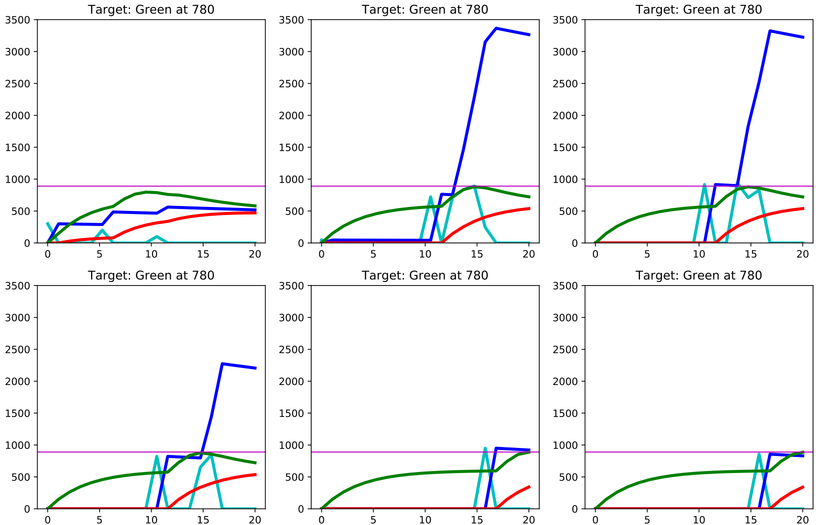
- Obtain $x_1, \ldots, x_t$ for input u into black box system
- Construct a model of the system and optimize u to achieve minimum $c(x(u))$

Can the true minimum of $c(x(u))$ be achieved? How many iterations?

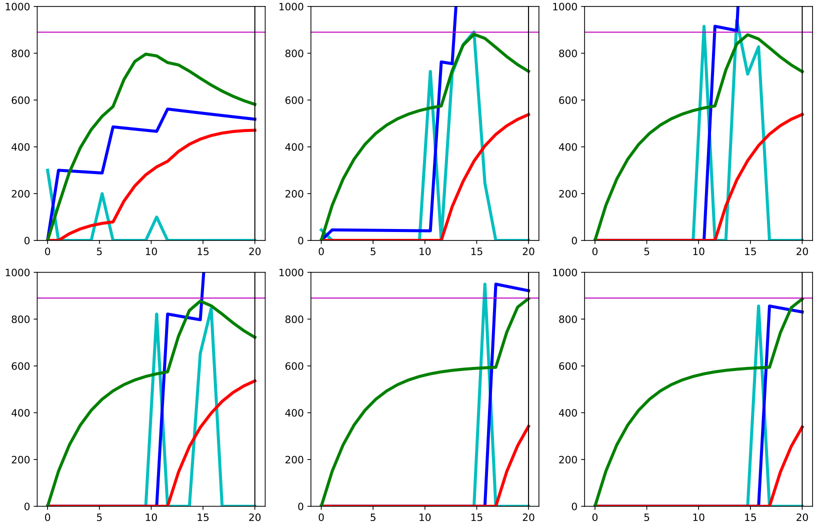
Control aim: green to 780



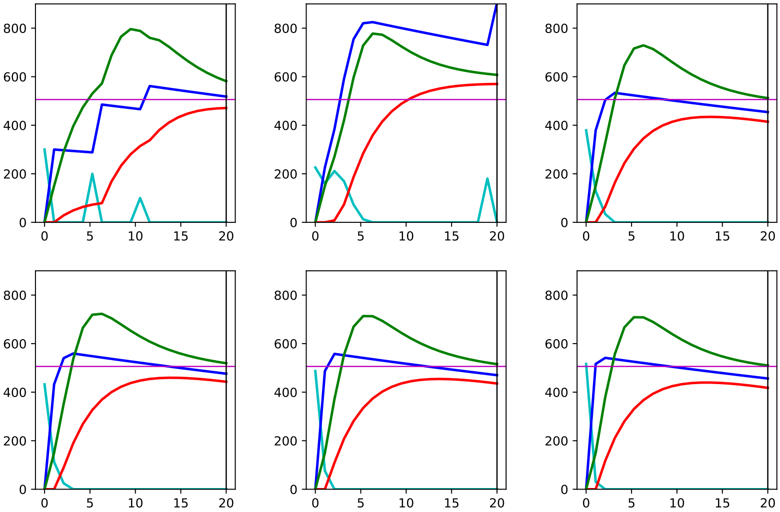
Control aim: green to maximum



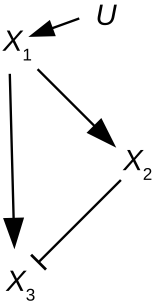
Green to maximum

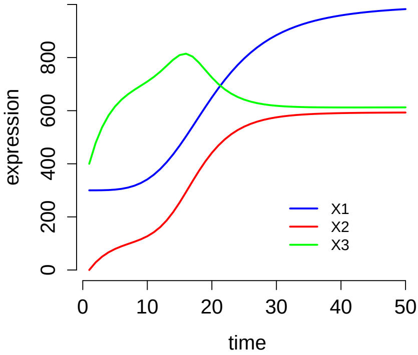


Control aim: green to minimum



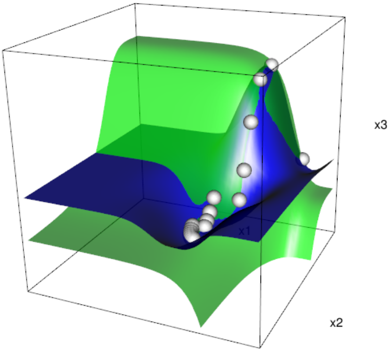
Feedforward loop

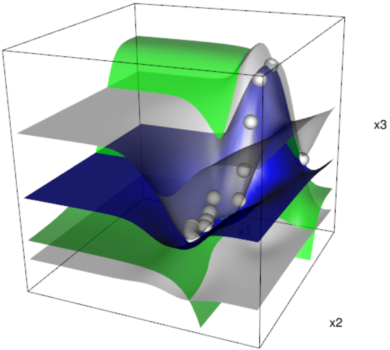




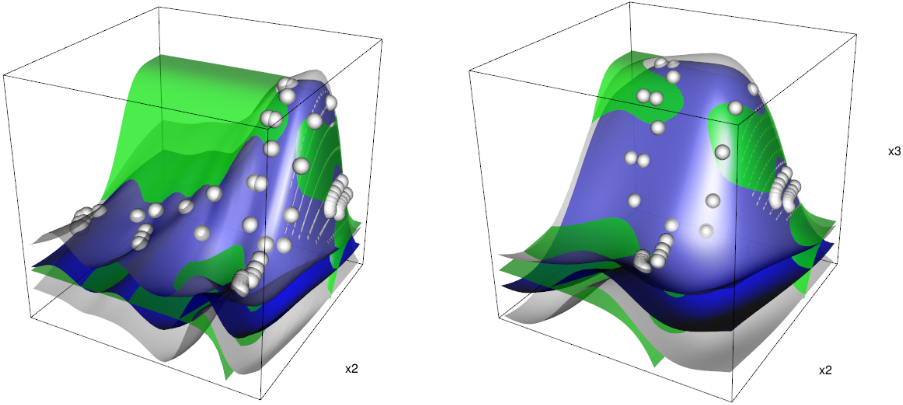
$$X_1(t) = (1 - \lambda_1)X_1(t-1) + U_{\text{activate}}(t)$$
$$X_2(t) = (1 - \lambda_2)X_2(t-1) + h^+(X_1(t-1))$$
$$X_3(t) = (1 - \lambda_3)X_3(t-1) + h^+(X_1(t-1)) + h^-(X_2(t-1))$$

GP for $f_3(x_1, x_2)$ initial experiment





After four maximization experiments



NANOG, SOX, OCT4 network

$$\frac{d[O]}{dt} = \frac{\eta_1 + a_1[A_+] + a_2[OS] + a_3[OS][N]}{1 + \eta_2 + b_1[A_+] + b_2[OS] + b_3[OS][N]} - \gamma_1[O] - k_{1c}[O][S] + k_{2c}[OS]$$

$$\frac{d[S]}{dt} = \frac{\eta_3 + c_1[A_+] + c_2[OS] + c_3[OS][N]}{1 + \eta_4 + d_1[A_+] + d_2[OS] + d_3[OS][N]} - \gamma_2[S] - k_{1c}[O][S] + k_{2c}[OS]$$

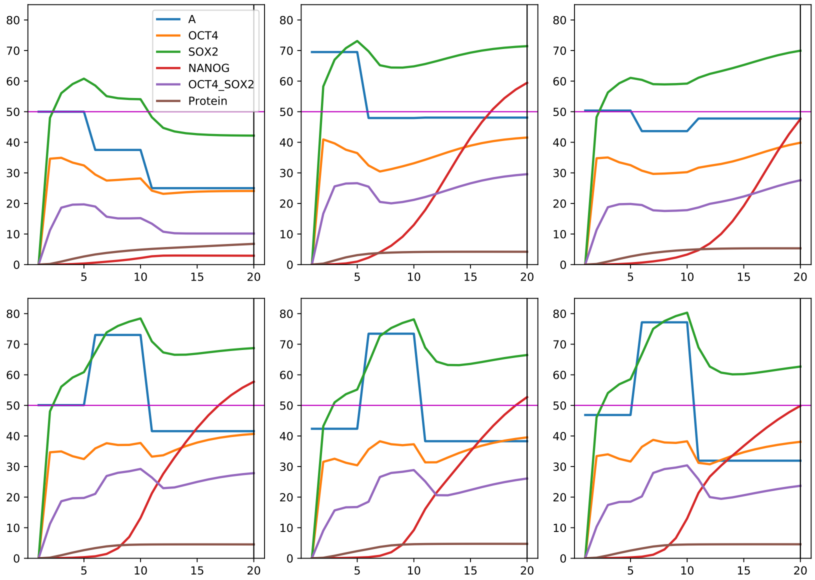
$$\frac{d[OS]}{dt} = k_{1c}[O][S] - k_{2c}[OS] - k_{3c}[OS]$$

$$\frac{d[N]}{dt} = \frac{\eta_5 + e_1[OS] + e_2[OS][N]}{1 + \eta_6 + f_1[OS] + f_2[OS][N] + f_3[B_-]} - \gamma_3[N].$$

Chickarmane et al., Transcriptional Dynamics of the Embryonic Stem Cell Switch, PLoS Comp. Biol.

SMBL Biomodel BIOMD0000000203

Control aim: NANOG (red) to 50



Afterthoughts

Taking uncertainty into account is essential: deterministic version (eg via GP mean function) does not work

Strong regularising effect of using expectation of cost

Converges surprisingly quickly

Dynamic control problems widely applicable

Tensorflow (or similar frameworks) enable straightforward, flexible implementation, can be easily adapted and expanded