Electronic Supplementary Material (ESI) for RSC Advances.
This journal is © The Royal Society of Chemistry 2017

# SUPPLEMENTARY INFORMATION

## Prediction of Concrete Corrosion in Sewers with Hybrid Gaussian Processes Regression Model

Yiqi Liu[a], Yarong Song[b], Jurg Keller[b], Philip Bond[b], Guangming Jiang* [b]

[a] School of Automation Science & Engineering, South China University of Technology, Wushang Road, Guangzhou 510640, China, Email: aulyq@scut.edu.cn

[b] Advanced Water Management Centre, The University of Queensland, St. Lucia, Brisbane QLD 4072, Australia, Email: g.jiang@awmc.uq.edu.au

**Corresponding author:** Guangming Jiang; Level 4 Gehrmann Building, Advanced Water Management Centre, The University of Queensland, St. Lucia, Brisbane, QLD 4072, Australia

Phone: +61 7 3346 3219; Fax: +61 7 3365 4726; E-mail: g.jiang@awmc.uq.edu.au

**Supplementary Material A**

The lower bound of $logp(y)$ can be formulated as follows:

$$logp(y) \geq \int q(u,v)logp(y|u,v)dudv - \sum_{j=1}^{Q} KL[q(u_j||p(u_j))] - \sum_{h=1}^{H} KL[q(v_h||p(v_h))]$$

(A.1)

which is derived using Jensen's inequality and the fact that both of $q(u_j)$, $q(v_h)$, $p(u_j)$, $p(v_h)$ are all multivariate Gaussian distribution. The KL divergence terms are analytically trackable. To compute the expected likelihood term,

$$logp(y|u,v) \geq \langle logp(y|g,t)\rangle_{p(g,t|u,v)} = \sum_{h=1}^{H}\sum_{i=1}^{N}\langle\ logp(y_{hi}|g_i,t_{hi})\rangle_{p(g|u)p(t_h|v_h)}$$

(A.2)

where $g_i = \{g_{ji} = (g_j)_i\}_{j=1}^{Q}$. After calculation of $\langle logp(y_{hi}|g_i,t_{hi})\rangle$, the resulting lower bound can be derived by substituting into eq. (A.1) as follows:

$$\begin{aligned} L = \sum_{h,i}\Big(&logN(y_{hi};\tilde{\mu}_{hi},\beta^{h^{-1}}) - \frac{1}{2}\beta^h\sum_{j=1}^{Q} w_{hj}^2\tilde{k}_{jii} \\ &- \frac{1}{2}\beta^h\tilde{k}_{hii}^t - \frac{1}{2}\beta^h\sum_{j=1}^{Q} trw_{hj}^2 s_j\Lambda_{ji} - \frac{1}{2}\beta^h trs_h^t\Lambda_{hi}\Big) \\ &- \sum_{j=1}^{Q}\Big(\frac{1}{2}log|K_{jzz}s_j^{-1}| + \frac{1}{2}trK_{jzz}^{-1}(\mu_j\mu_j^T + S_j)\Big) \\ &- \sum_{h=1}^{H}\Big(\frac{1}{2}log|K_{hzz}(s_h^t)^{-1}| + \frac{1}{2}trK_{hzz}^{-1}(\mu_h^t(\mu_h^t)^T + s_h^t)\Big) \end{aligned} \quad \text{(A.3)}$$

where $K_{jzz} = k(Z_j, Z_j)$, $K_{hzz} = k(Z_h^t, Z_h^t)$, and

$$\tilde{\mu}_{hi} = \sum_{j=1}^{Q} w_{hj}A(i,:)m_j + A_h^h(i,:)m_h^t \quad \text{(A.4)}$$

$$\Lambda_{ji} = A_j(i,:)^T A_j(i,:) \quad \text{(A.5)}$$

$$\Lambda_{hi} = A_h^t(i,:)^T A_h^t(i,:) \quad \text{(A.6)}$$

with $\tilde{k}_{jii} = (\tilde{K}_j)_{ii}$; $\tilde{k}_{hii} = (\tilde{K}_h)_{ii}$; $\mu_{ji} = (\mu_j)_i$; $\mu_{hi}^t = (\mu_h^t)_i$; $A_j = k(X, Z_j)K_{jzz}^{-1}$; $A_h^t = k(X, Z_h^t)K_{hzz}^{-1}$ and $A_j(i,:)$ is used to denote the *i*-th row vector of $A_j$. All parameters $l$ can be derived by calculation of $\frac{\partial L}{\partial \mu_j} = 0$, $\frac{\partial L}{\partial s_j} = 0$, $\frac{\partial L}{\partial \mu_h^t} = 0$, $\frac{\partial L}{\partial s_h^t} = 0$, $\frac{\partial L}{\partial Z} = 0$, $\frac{\partial L}{\partial Z^t} = 0$, $\frac{\partial L}{\partial \beta^h} = 0$.

**Supplementary Material B: HA definition**

An automaton is a formal model for a dynamic system with discrete and continuous components. A hybrid automaton is a tuple *H* = (*X, Q, Inv, Flow, E, Jump, Reset, Event, Init*) where:

- *X* is a finite set of *n* real-valued variables that model the continuous dynamics;
- *Q* is a finite set of control locations (mode);
- *Inv* is a mapping, which assigns an invariant condition to each location $q \in Q$. *Inv*(q) is a predicate over the variables in *X*. The control of a hybrid automaton remains at a location $q \in Q$, as long as *Inv*(*q*) holds;
- *Flow* is a mapping, which assigns a flow condition to each control location $q \in Q$. The flow condition Flow(*q*) is a predicate over *X* that defines how the variables in X evolve over the time *t* at location *q*;
- $E \subseteq Q \times Q$ is the discrete transition relation over the control locations;
- *Jump* is a mapping, which assigns a jump condition (guard) to each transition $e \in E$. The jump condition *jump*(e) is a predicate over *X* that must hold to fire *e*.

Omitting a jump condition on a transition means that the jump condition is always true and it can be taken at any point of time. Conventionally, writing *Jump*($e$)[$v$] means that the jump condition on a transition $e$ holds, if the variations of variables on the transition $v$;

- *Reset*($e$) is a predicate over $X$ that defines how the variables are reset;
- *Event* is a finite set $\Sigma$ of events, and an edge labelling function $event: E \rightarrow \Sigma$ that assigns to each control switch an event;
- *Init* is the initial state of the automaton. It defines the initial location together with the initial values of the variables $X$.

**Supplementary Material C: An example for hybrid automata model building**

To further illustrate hybrid automata, a pump station model based on autonomous hybrid automata is shown as follows. The hybrid automaton of Figure S1 models a pump station, which turns on and off according to the sensed water level. The variable $x$ represents the water level. In control mode OFF, the pump station is off, and the water level rises according to the flow condition (*Flow*) $\dot{x} = \frac{inflow(t)}{S}$, where $S$ is pump station wet well area, $t$ is the time, and *inflow* is incoming flow into pump station. In control mode ON, the pump station is on, and the water level falls or rises according to the Flow condition $\dot{x} = \frac{inflow(t) - outflow}{S}$, where outflow is constant due to constant pump speed operation. In this example, initially, the pump station is off and the initialized water level is 0.5. According to the jump condition $x > UL$ (water level upper limit), the pump station may go ON as soon as the water level reaches $UL$.

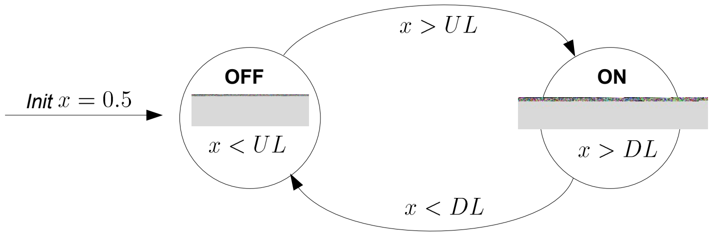


**Figure S1.** Pump station autonomous hybrid automaton

According to the invariant condition (*Inv*) $x < UL$ in the OFF circle, the pump station will stay OFF when the water level is lower than *UL*. Similar behaviour will occur once water level is lower than *DL* (water level down limit) in the ON circle, if pump station is ON.

**Supplementary Material D: Multivariate linear regression**

Multivariate regression is usually performed to predict several response variables simultaneously. The usual description of MLR is multivariate linear regression and is formulated as follows:

$$y^h = \beta^h x + \varepsilon^h \tag{D.1}$$

where $y^h$, $\beta^h$ and $\varepsilon^h$ represent noisy output, parameters and noises with respect to the *h*th response, $h = 1,\cdots,H$ and *N* is the number of observations, the dimension of input variables *x* is *M*. To identify the parameters $\theta = \{(\beta^h, D^h)\}_{h=1}^{H}$, covariance-weighted least squares estimation is used, which is formulated in the Matlab Toolbox[33]. To compute the predictive distribution of $y^h$ at a new testing input $x^*$, the mean values and variances are given as follows:

$$\mu_*^h = \beta^h x^* \tag{D.2}$$

$$s_*^h = D^h \tag{D.3}$$

where $D^h = diag\{(\sigma_1^h)^2, \cdots, (\sigma_M^h)^2\}$, $\sigma_M$ represents the variance with respect to the *M*th input variable.

**Supplementary Material E: Radial basis function networks**

A radial basis function network is an artificial neural network that uses radial basis functions as activation. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters. Radial basis function (RBF) networks typically have three layers: an input layer, a hidden layer with a non-linear RBF activation function and a linear output layer. The input can be modeled as a vector of real numbers $x \in R^n$. The output of the network is then a scalar function of the input vector and is given by

$$y = \sum_{i=1}^{N} a_i \varphi(x_i) \quad \text{(E.1)}$$

where $\varphi(x) = exp[-\beta \|x - c_i\|^2]$, *N* is the number of neurons in the hidden layer, $c_i$ is the center vector for the *i*th neuron , and $a_i$ is the weight of neuron *i* in the linear output neuron. Functions that depend only on the distance from a center vector are radially symmetric about that vector, hence the name radial basis function. In the basic form all inputs are connected to each hidden neuron. The norm is typically taken to be the Euclidean distance (although the Mahalanobis distance appears to perform better in general) and the radial basis function is commonly taken to be Gaussian. Parameters of one neuron has only a small effect for input values that are far away from the center of that neuron. Given certain mild conditions on the shape of the activation function, an RBF network with enough hidden neurons can approximate any continuous function with arbitrary precision. The parameters $a_i, c_i, \beta_i$are determined in a manner that optimizes the fit between $\varphi(x)$ and the data.

**Supplementary Material F: Kernel selection**

A kernel (also called a covariance function) is a positive-definite function of two inputs $x_i$, $x_j$, and is defined as $k(x_i, x_j)$ to represent the similarity between two objects. The most few basic kernels are shown in Fig. S2, suggesting that each covariance function is able to make a different set of assumptions about the function we wish to model. Even if the kind of structure is not expressed by any known kernel, kernels can be combined to create new ones with different properties (Fig.1 Line 2 and Line 3)[34].

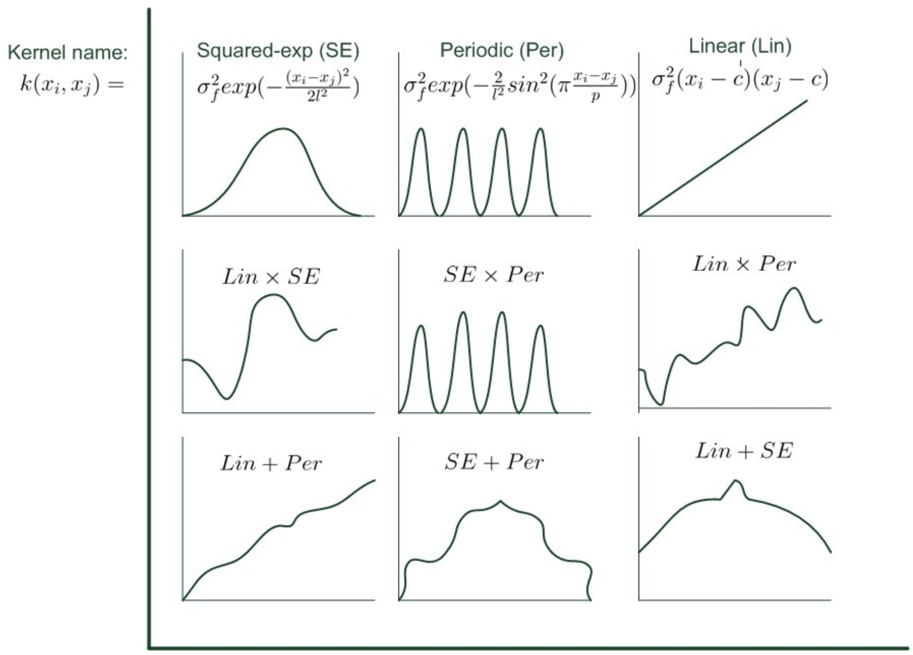


**Fig. S2.** Kernel functions for the GPR model

Different from the general covariance function, we also can use a new proposed addictive covariance function together with Squared-Exp (SE) as a base covariance function shown as follows:

$$k_{add}(x_i,x_j) = \sigma_f^2 exp\{ - \frac{1}{2}\sum_{q = 1}^{d}\frac{(x_{iq} - x_{jq})^2}{2l_q^2}\} \qquad \text{(F.1)}$$

where $k_{ij}$ is the base kernel Squared-Exp (SE), $d$ is the column number of training samples. This model, in fact, is a sum of functions of all possible combinations of input variables. This model can be specified by a weighted sum of all possible products of one-dimensional kernels. In our model, the only design choice necessary to specify an additive kernel is the selection of a one-dimensional base kernel for each input dimension. Parameters of the base kernels (such as length-scales ($l_1,l_1,\ldots,l_d$) can be learned as per usual by maximizing the marginal likelihood of the training data.

Table S1. Empirical reliability with different confidence levels

| | Corrosion initiation time (months) | | Corrosion rate (mm/y) | |
|---|---|---|---|---|
| | GPR-ex, GP | GPR-ex, PS | GPR-ex, GP | GPR-ex, PS |
| 95% Empirical reliability | 20/90<br>(78%) | 10/94<br>(90%) | 8/110<br>(92.7%) | 15/127<br>(88%) |

Notes: *x*/*y* (*z*%), *x* and *y* represent the number of samples out of 90% confident control limit and the total number of samples, respectively. *z* is the actual percentage of samples which failed to be predicted.