Nedbálek Jakub
VŠB – Technical University of Ostrava, Ostrava Poruba, Czech Republic

Neural networks for function approximation in dynamic modelling

Keywords

reliability, Monte Carlo, RBF neural network, simulation, temperature

Abstract

The paper demonstrates the comparsion of Monte Carlo simulation (MC) algorithm with the Radial Basis Function (RBF) neural network enhancement of the same algorithm in the reliability case study. In our test, we dispose of the tank containing liquid water – its temperature process variable evolves deterministicly according to the differential equation, which solution is known. All component failures are considered as a stochastic events. In the case of surpassing temperature treshhold of the liquid inside the tank, we interpret the situation as the system failure. With regard to process dynamics, we attempt to evaluate the tank system unreliability related to the initiative input parameters setting. The neural network is used in equation coeficients calculation, which is executed in each transient state. Due to the neural networks, for some of the initial component settings, we can achieve the results of computation faster than in classical way of coeficients calculating and substituting into the equation.

1. Introduction

Let us have the model of a dynamic system, in which the temperature is evolving according to the time and initial component settings. The target is to specify the probability of a system failure, which is defined as exceeding the temperature bounds. We are also interested in the time necessary for computing the result. It is proposed to enhance the simulation algorithm with neural network tools which will be used in calculating the differential equation coeficients *a* and *b* (chap. 3. relation (4)) being changed acording to k_i component states (on/off). After each k_i switching, which is invoked by either passing the temperature transition state or failure of k_i component, we must calculate new values of prameters *a* and *b* in equation (4) according to (2).
As a solution, it is appropriate to apply neural networks for the aproximation of parameters (2) dependent on the k_1, k_2 and k_3 component settings.
Optimal tool for constructing the simulation algorithm is the Monte Carlo (MC) method. This paper is derived from [2, 3].

2. The benchmark process description

We dispose of the tank with warmed water, which temperature is kept in the specific maximal or minimal bounds – in this range, we consider system as stable and reliable. The system also contains two electric components, responsible for water heating, and security valve, which decreases the temperature. In the bottom of the tank, there is a faucet for water supplying. We suppose, the volume of water in the tank is constant during our experiment.
Let us define variables:
T(t) – temperature of water at the time t;
Tempmax – maximal temperature of water in the tank;
Tempmax = 368,15K
Tempmin – minimal temperature of water, for T< Tempmin failure occurs
Tempmin = 338,15K
Tempbas – security level for the minimal temperature
Tempbas = 343,13K
Temphau – security level for the maximal temperature
Temphau = 363,15K
Secu – reserve for the maximum temperature, for T> (Tempmax + Secu) failure occurs, Secu = 2 K
M – water weight, M = 500kg
Te – external temperature, Te = 293K
A – tank surface, A = 6m^2
h – thermal exchange coeficient, h = 6 WK^{-1}m^{-2}
c_p – measure heat capacity, c_p = 4184 Jkg^{-1}K^{-1}
$W_1 = W_2$ – heating power, W = 5000W
tm – process duration, tm = 720 h

hazard rate – transition to on-state
$\lambda_{W1on} = \lambda_{W2on} = 6.10^{-4}\ h^{-1}$
hazard rate – transition to off-state
$\lambda_{W1off} = \lambda_{W2off} = 4.10^{-4}\ h^{-1}$
hazard rate – transition to on or off-state
$\lambda_{Vson} = \lambda_{Vsoff} = 1.10^{-3}\ h^{-1}$

3. The equation solution

To evaluate the probability failure, we need to write the differential equation, describing our system evolution. The equation obviously reflects the following points:
a/ decreasing the initial temperature due to heat penetration through the tank wall
b/ increasing the water temperature caused by two heating components, if activated
c/ the water temperature decrease invoked by the security valve activation
Our equation comes from [3] but it is altered for the behaviour of the system

$$\frac{dT}{dt} = aT + b \qquad (1)$$

where

$$a = -\left(\frac{A \cdot h}{M \cdot c_p} + \frac{Q_s \cdot c_p \cdot k_3}{M \cdot c_p}\right)$$

$$b = \left[\frac{1}{M \cdot c_p} \cdot \left(A \cdot h \cdot T_e + Q_s \cdot c_p \cdot T_e \cdot k_3 + W_1 \cdot k_1 + W_2 \cdot k_2\right)\right] \qquad (2)$$

and

$$Q_s = \frac{\dfrac{W_1 + W_2}{Temp\max - T_e} - A \cdot h}{c_p} \qquad (3)$$

The solution of (1) follows the equation

$$T = \left(T_0 + \frac{b}{a}\right)\exp^{a \cdot t} - \frac{b}{a} \qquad (4)$$

where T_0 is the starting simulation temperature.
The k_1, k_2 and k_3 coefficients equals 1 or 0 (the specific component is either on or off). For $k_1 = k_2 = 1$ the heating components are active and temperature of water in the tank is increasing, for $k_3 = 1$ the vent is unclosed and the temperature is decreasing, etc. We watch the process along the period of tm = 720 h. The initial temperature is set between Tempmin and Tempmax, that is – $T_0 = 353{,}15$ K.

4. Creating an algorithm

To construct the correct algorithm for our test case simulation, we take into account following points:

a) as mentioned before, for T< Tempmin and also for T> (Tempmax + Secu) failure occurs

b) the temperature passes by 5 stages generally – see the diagram:

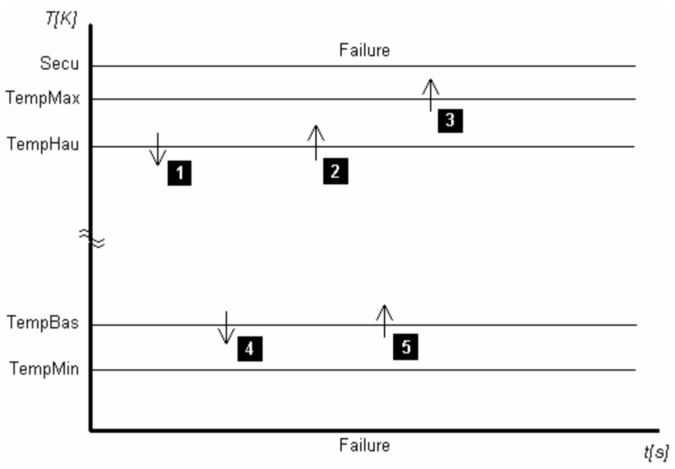

Figure 1. Dynamic rules of the system

For each of the temperature stages, the change (switch) of the specific component to the opposite state, that causes the required temperature turnover (see (1)) and stabilization in tolerable bounds. In case of random failure of the k_i component, we keep on monitoring evolution of the temperature, until it exceeds limits – we consider the system as disfunctional. (In the terms of the k_i failure definition, the whole system does not have to be failed yet. The temperature of water in the tank could be still between bounds.)

c) There are following rules for components changes at temperature borders crossing:
State 1: If T(t-1)>= Temphau and T(t) <= Temphau, then $k_3 = 0$ (vent will be closed)
2: If T(t-1) <= Temphau and T(t) >= Temphau, then $k_1 = 0$ (heating component num. 1 will be cut off)
3: If T(t-1) <= Tempmax and T(t) >= Tempmax, then $k_3 = 1$ (vent will be opened)
4: If T(t-1) >= Tempbas and T(t) <= Tempbas, then $k_1 = 1$ & $k_2 = 1$ (both heating components are active)
5: If T(t-1) <= Tempbas and T(t) >= Tempbas, then $k_2 = 0$ (heating component num. 2 will be cut off)

d) time step option

Considering fact, that we present the evolution of (4) at time *t* during the period of *tm*, it is necessary to select an appropriate time to explore all detail changes of the temperature bahaviour and also to reduce the inadequate number of cycles of numerical simulation. The optimal solution seems to be the one minute pattern, which reflects suitably all changes at temperature borders crossing. Longer patterns do not suit our solution due to inaccurancies – a "jump-over" of some of the states mentioned in c) occurs sometimes.

e) Switching the component to the opposite state could happen at any time in the simulation due to random failure.

f) Period of the process is set for 720 hours.

5. Application of the RBF

Our simulation algorithm contains cycle, running over the process duration, in which (4) evolves according to time. This equation has coefficients *a* and *b*, that depend on k_i component states (on/off) – see (2). In the simulation, the k_i state is influenced by either passing the temperature transition state (see *Figure 1.*) or failure of component itself. It means, that we must recalculate the *a* and *b* whenever the temperature transition or failure of the k_i occurs. Simply, we are able to write lines of code to enumerate new values of the *a* and *b* right in the body of process duration cycle, whenever it is necessary to do so. The second possibility is to apply the Radial Basis Function (RBF) neural network to approximate the function of *a* and *b* coefficients depending on k_i component states.
It is acceptable to use other types of neural network, nevertheless the RBF is obviously the best to solve the problem. This is the result of two main facts, firstly, we are not urged to design the network architecture (RBF has two layers standardly) and secondly, the RBF can not be trapped in a local minimum during training phase [1]. RBF complies our requirements on the function approximation [6]. Applying other types of neural network to unriddle this case study and to compare them with the used RBF network is the matter of a future research.
At the begining, we need to find out the convenient training set. This is obtained by simple computation of (2) for all combinations of the k_i states (see *Table 3.*). Then, before the process duration cycle, we are ready to create and train the standard RBF architecture – there are several implementations and function support of the RBF in programming languages – for example, the Matlab software provides large neural network toolbox.

Consequently, the *a* and *b* parameters in (4) everywhere in the cycle are replaced with the callback function of the RBF network.
We can generally summarize, that the main modification consist in using the RBF as an auxiliary tool for working with equation (4) during the time of a simulation cycle. In any case, the MC construction of the algorithm remains the same for both cases.

6. The results presentation

Table 1. contains the distibutional function of failure probability value averages for each initial components settiings. The results were obtained for 10^5 Monte Carlo simulations (1- the comp. active, 0 – comp. inactive at the beginning). The fifth column shows the computational time. All results are obtained in the state of *tm* = 720 h. The simulation was implemented in the Matlab software.

Table 1. The results for 10^5 cycles of Monte Carlo

k1	k2	K3	aver.F(tm)	aver. t[s]
0	0	0	0,3517	2315,0
0	0	1	0,5303	2174,0
0	1	0	0,5567	1928,6
0	1	1	0,5312	2170,7
1	0	0	0,3518	2332,1
1	0	1	0,5306	2194,0
1	1	0	0,5580	1920,4
1	1	1	0,5602	1915,6
aver.			0,4963	2118,8
sigma			0,0901	174,1

Table 2. The results for the same Monte Carlo algorithm with RBF neural network enhancement

k1	k2	k3	aver.F(tm)	aver. t[s]
0	0	0	0,3510	2383,1
0	0	1	0,5305	2037,1
0	1	0	0,5574	2002,9
0	1	1	0,5325	2042,2
1	0	0	0,3506	2390,6
1	0	1	0,5305	2040,5
1	1	0	0,5578	1975,2
1	1	1	0,5593	1968,9
aver.			0,4962	2105,1
sigma			0,0906	176,2

From comparsion of *Table 1.* with *Table 2.*, we can see the results of simulation at the time of 720 hours are very close – the RBF neural network is able to

aproximate with good accurancy (that was tested in the simulation code itself).
The results of computing time look more interesting – the average time necessary to simulate 720 hours long process is shorter by roughly 10 sec. This value seems to be neglectible, nevertheless the differences in results between the MC and the modification with RBF are larger when we look at the specific initial component settings.
Generally, we can express the presumption, that if the vent is opened and maximally one heating spiral is activated, it is more useful to enhance the MC algorithm with RBF network (the result is reached by 2- 2.5 min faster). In other cases, the Monte Carlo itself is faster (1 min. advance).
In this place, we should stress out the information, that the comparsion test on the MC and RBF network enhancement was executed on the computer, which had all applications, including hidden ones, and non-operation system processes not pertaining to simulation itself, halted. This measure is needed in order to provide the simulation the similar computing system capacity along the whole processing time and avert the distortion in result time values (operating system sometimes allocate to the other running applications the memory, as consequently leads to Matlab processing slow down).
With respect to the length of algorithm, the MC enhanced with the RBF is larger in creation and training of the network. In the simulation itself, the length of code remains the same.
In *Table 2.*, we also considered time necessary to train the RBF network.
The results from *Table 1.* and *2.* are presented in the figures. The x-axis denotes possible component states according to binary code, as it is shown in *Table 3.*

Table3. The k_i component states combination (1 -on, 0 -off)

X axis	k1	k2	k3
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
… etc.			

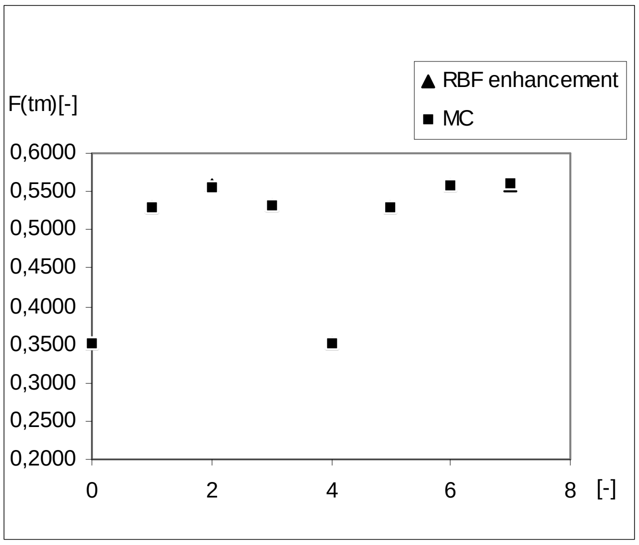

Figure 2. Failure probability comparsion of the MC and the RBF neural network enhancement at time *tm*

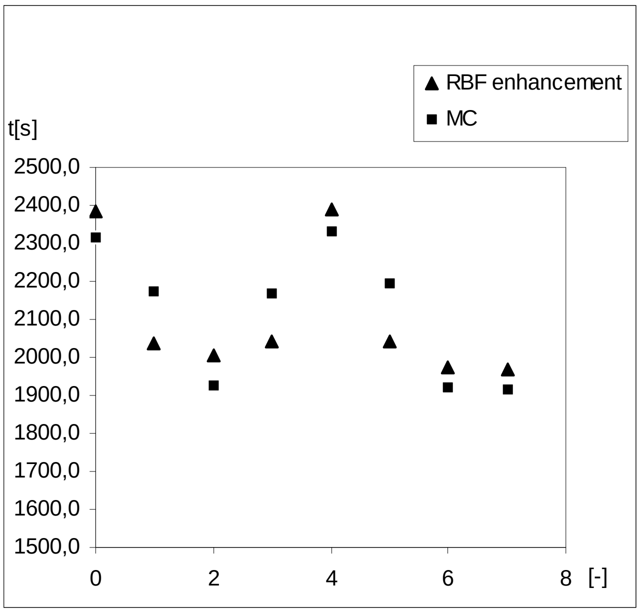

Figure 3. Computing time comparsion of the MC and the RBF neural network enhancement

7. Conclusion

For 10^5 cycles, the failure probability at time t = 720hrs equals to the value $\overline{F}_{(720)}$= 0,4963 $\pm$ 0,0901 (MC) or 0,4962 $\pm$ 0,0906 (RBF enhancement). The algorithm in chap. 4 is implemented in the Matlab software.
Out of the comparsion of the *Figure 1* and *Figure 2* follows, that the failure probability values are similar for both methods.
The whole computing time needed to obtain results for each initial component settings is shorter by approx. 10 sec. when we use enhancement with RBF network. The

greater differences in time consuption are evident for specific settings – we can state, that if the security vent is opened and maximally one heating spiral is activated than it is preferable to add the RBF in algorithm (the result is known by 2 – 2.5 min faster), in all other cases, the plain Monte Carlo method is more suitable (faster by about 1 min). Application of the RBF neural network can sometimes lead to obtain results faster. This information is likely to be applicable in other, not only dynamic simulation, test cases.

8. Acknowledgements

The author of this paper would like to thank for the financial support on behalf of the research and development project num. 1M06047 (CQR), which is subventioned by the Ministry of Education of the CR.

References

[1] Chen, S., Cowan, C. F. N. & Grant, P. M. (1991). Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks. *IEEE Transactions on Neural Networks* vol. 2, no. 2, March, 302-309.

[2] Nedbálek, J. (2007). The Temperature Stability of Liquid in the Tank. *Proc. Risk, Quality and Reliability*, Ostrava 131-133.

[3] Pasquet, S., Chatalet, E., Padovani, E. & Zio E. (1998). *Use of Neural Networks to evaluate the RAMS` parameters of dynamic systems*. Université de Technologie de Troyes France, Polytechnic of Milan Italy.

[4] Pasquet, S., Chatalet, E.,Thomas, P. & Dutuit, Y. (1997). Analysis of a Sequential, Non- Coherent and Looped System with Two Approaches. Petri Nets and Neural Networks. *Proc. of International cenference on safety and reliability,* ESREL’97, Lisabon, Portugal, 2257-2264.

[5] Virius, M. (1985*). Základy výpočetní techniky* (Metoda Monte Carlo), ČVUT, Praha.

[6] Yee, P. V. & Haykin, S. (2001). *Regularized Radial Basis Function Networks: Theory and Applications.* John Wiley.