Software Package Development for the Active Traffic Management Module Self-oscillation Regime Investigation

Tatyana R. Velieva[1,3(✉)], Anna V. Korolkova[1,3], Anastasiya V. Demidova[1,3], and Dmitry S. Kulyabov[1,2,3]

[1] Department of Applied Probability and Informatics, Peoples' Friendship University of Russia (RUDN University), 6 Miklukho-Maklaya Street, 117198 Moscow, Russian Federation
{velieva_tr,korolkova_av,demidova_av,kulyabov_ds}@rudn.university
[2] Laboratory of Information Technologies, Joint Institute for Nuclear Research, 6 Joliot-Curie Street, Dubna, 141980 Moscow Region, Russian Federation
[3] Bogoliubov Laboratory of Theoretical Physics, Joint Institute for Nuclear Research, 6 Joliot-Curie Street, Dubna, 141980 Moscow Region, Russian Federation

Abstract. Self-oscillating modes in control systems of computer networks quite negatively affect the characteristics of these networks. The problem of finding the areas of self-oscillations is actual and important as the study of parameters of self-oscillations. These studies are extremely labor-intensive because of the substantial non-linear nature of the mathematical model. To investigate the self-oscillation parameters, the authors used the method of harmonic linearization. However, the previously used model with symmetric oscillations showed its limitations. In this paper, a more complex model with asymmetric oscillations is applied. For this purpose, the authors developed the program complex with both numerical analysis elements and elements of symbolic calculations.

Keywords: Traffic active management · Control theory
Self-oscillating mode

1 Introduction

While modeling technical systems with control it is often required to study not only characteristics of these systems, but also the influence of system parameters on these characteristics.

In systems with control there is a parasitic phenomenon as the self-oscillating mode. Earlier, we conducted the study to determine the regions of occurrence of self-oscillations in the system with the RED Active Queue Management algorithm [8]. To do this, we applied the method of harmonic linearization. We used a simplified model with unbiased oscillations. However, the asymmetry of the

W. Zamojski et al. (Eds.): DepCoS-RELCOMEX 2018, AISC 761, pp. 515–525, 2019.
https://doi.org/10.1007/978-3-319-91446-6_48

drop function generates a constant bias of self-oscillations. Thus this case has limited use.

In this paper, the method of harmonic linearization with asymmetric oscillations is applied. For this more labor-intensive method (than the one considered earlier) the authors have developed the software package for symbolic formulas obtaining and programs for numerical computation generating.

The structure of the paper is as follows. In the Sect. 2, we provide a brief description of the RED algorithm. In the Sect. 3, we describe the method of harmonic linearization for nonsymmetric oscillations. In the Sect. 4, we calculate the harmonic linearization coefficients for nonsymmetric oscillations. In the Sect. 5 we describe the main elements of the developed software package. In Sect. 6 a concrete example of calculation of self-oscillation parameters is given.

2 The RED Congestion Adaptive Control Mechanism

The RED algorithm uses a weighted queue length as a factor determining the probability of packets drop. As the average queue length grows, the probability of packets drop also increases. The algorithm uses two threshold values of the average queue length to control the drop function.

The RED algorithm is quite effective due to simplicity of its implementation in the network hardware, but it has a number of drawbacks. In particular, for some parameters values there is a steady oscillatory mode in the system, which negatively affects the quality of service (QoS) indicators [4,10,17]. Unfortunately there are no clear selection criteria for RED parameters values, at which the system does not enter in self-oscillating mode.

To describe the RED algorithm we will use the continuous model (see [1,2,6,11,12,18]) with some simplifying assumptions: the model is written in the moments; the model describes only the phase of congestion avoidance for TCP Reno protocol; in the model the drop is considered only after reception of 3 consistent ACK confirmations.

3 Harmonic Linearization Method

The method of harmonic linearization was proposed by Bogolyubov, Krylov [7] and Nyquist [13]. The content of this method is reduced to separating the 'slow' variables from the 'fast' variables. The harmonic-linearized system depends on the amplitudes and frequencies of the periodic processes. This is an essential difference between harmonic linearization and the usual linearization method, leading to purely linear expressions, which allows us to investigate the basic properties of nonlinear systems.

The method of harmonic linearization is used for systems of a certain structure (see Fig. 1). The system consists of a linear link H_l and a nonlinear link H_{nl}, given by the function $f(x)$. A static nonlinear element is usually considered.

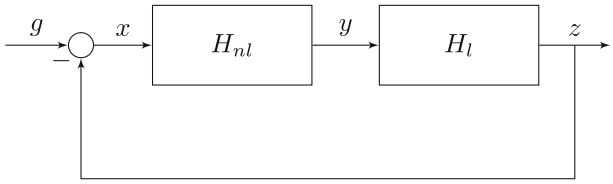

Fig. 1. Block structure of the system for the harmonic linearization method

The free harmonic oscillations are applied to the input of the nonlinear element:

$$x(t) = x_0 + \tilde{x} := x_0 + A\,\sin(\omega t).$$

On the output of the nonlinear element $f(x)$ we get a periodic signal. Let's expand it in a Fourier series:

$$y = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k\,\sin(k\omega t) + b_k\,\cos(k\omega t)).$$

where the coefficients of the Fourier series have the following form:

$$a_0 = \frac{1}{\pi}\int_0^{2\pi} f(x_0 + A\sin(\omega t))\,\mathrm{d}(\omega t)\,;$$
$$a_k = \frac{1}{\pi}\int_0^{2\pi} f(x_0 + A\sin(\omega t))\sin(k\omega t)\,\mathrm{d}(\omega t)\,;$$
$$b_k = \frac{1}{\pi}\int_0^{2\pi} f(x_0 + A\sin(\omega t))\cos(k\omega t)\,\mathrm{d}(\omega t)\,; \quad k = \overline{1,\infty}.$$

The linear element is a low-pass filter, that is, when k is increasing the linear elements suppress higher harmonics.

Let's write the signal after the non-linear element:

$$y = y_0 + \tilde{y} \approx \varkappa_0(A,\omega,x_0) + [\varkappa(A,\omega,x_0) + \mathrm{i}\varkappa'(A,\omega,x_0)]\tilde{x}, \tag{1}$$

$\varkappa_0$ is a constant shift, $\varkappa$ and $\varkappa'$ are the harmonic linearization coefficients:

$$\varkappa_0(A,\omega,x_0) = \frac{1}{2\pi}\int_0^{2\pi} f(x_0 + A\sin(\omega t))\,\mathrm{d}(\omega t)\,;$$
$$\varkappa(A,\omega,x_0) = \frac{a_1}{A} = \frac{1}{A\pi}\int_0^{2\pi} f(x_0 + A\sin(\omega t))\sin(\omega t)\,\mathrm{d}(\omega t)\,; \tag{2}$$
$$\varkappa'(A,\omega,x_0) = \frac{b_1}{A} = \frac{1}{A\pi}\int_0^{2\pi} f(x_0 + A\sin(\omega t))\cos(\omega t)\,\mathrm{d}(\omega t)\,.$$

In addition to (1), we will write

$$z = z_0 + \tilde{z} = (y_0 + \tilde{y})H_l(\omega),$$
$$x = x_0 + \tilde{x} = g(\omega) - (z_0 + \tilde{z}).$$

Then we may obtain the harmonic linearization equation:

$$\left[x_0 + H_l(\omega)\Big|_{\omega=0} \varkappa_0(A,\omega,x_0)\right] + \\ + [1 + H_l(\varkappa(A,\omega,x_0) + \mathrm{i}\varkappa'(A,\omega,x_0))]\tilde{x} = g(\omega) := g_0(\omega) + \tilde{g}(\omega).$$

Separating for constant and harmonic components, we may write:

$$\left[x_0 + H_l(\omega)\Big|_{\omega=0} \varkappa_0(A,\omega,x_0)\right] = g_0(\omega),$$
$$[1 + H_l(\varkappa(A,\omega,x_0) + \mathrm{i}\varkappa'(A,\omega,x_0))]\tilde{x} = \tilde{g}(\omega).$$

In the study of self-oscillatory mode it is assumed that there is no external signal ($g = 0$).

4 RED Model Harmonic Linearization

The linearization of the RED model and the derivation of the function H_l are described in detail in the article [8]:

$$H_l(\omega) = -\frac{C^4 T_f^3 w_q}{2N(Cw_q + \mathrm{i}\omega)(\mathrm{i}T_f\omega + 1)\left(\mathrm{i}CT_f^2\omega - \mathrm{i}NT_f\omega + 2N\right)}.$$

The linearized function P_{RED} has the form shown in Fig. 2

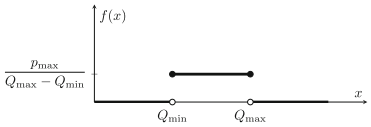

Fig. 2. The function P_{RED}

Let us compute the coefficients of harmonic linearization $\varkappa_0(A,\omega,x_0)$, $\varkappa(A,\omega,x_0)$ and $\varkappa'(A,\omega,x_0)$ (2) for the static nonlinearity P_{RED}:

$$\varkappa_0(A,\omega,x_0) = \frac{1}{2\pi}\int_0^{2\pi} P_{\mathrm{RED}}(x_0 + A\sin(\omega t))\,\mathrm{d}(\omega t)\,;$$
$$\varkappa(A,\omega,x_0) = \frac{1}{A\pi}\int_0^{2\pi} P_{\mathrm{RED}}(x_0 + A\sin(\omega t))\sin(\omega t)\,\mathrm{d}(\omega t)\,;$$
$$\varkappa'(A,\omega,x_0) = \frac{1}{A\pi}\int_0^{2\pi} P_{\mathrm{RED}}(x_0 + A\sin(\omega t))\cos(\omega t)\,\mathrm{d}(\omega t)\,.$$

Depending on the relations between the thresholds $Q_{\min}$, $Q_{\max}$, the shift x_0 and the amplitude A, it is possible to obtain different limits of integration. For example, let's consider the case when

$$Q_{\min} < x_0 < Q_{\max}, \quad x_0 - A > Q_{\min}, \quad x_0 + A > Q_{\max}.$$

Then we will obtain:

$$\begin{gathered}
\varkappa_0(A, x_0) = \frac{1}{2\pi} \frac{p_{\max}}{Q_{\max} - Q_{\min}} \int_0^{\alpha_{\max}} \mathrm{d}(\omega t) + \\
+ \int_{\pi - \alpha_{\max}}^{2\pi} \mathrm{d}(\omega t) = \frac{1}{2\pi} \frac{p_{\max}}{Q_{\max} - Q_{\min}} [2\alpha_{\max} + \pi]; \\
\varkappa(A, x_0) = \frac{1}{A\pi} \frac{p_{\max}}{Q_{\max} - Q_{\min}} \left[(-\cos(\omega t)) \Big|_0^{\alpha_{\max}} + (-\cos(\omega t)) \Big|_{\pi - \alpha_{\max}}^{2\pi} \right]; \\
\varkappa'(A, x_0) = \frac{1}{A\pi} \frac{p_{\max}}{Q_{\max} - Q_{\min}} \left[\sin(\omega t) \Big|_0^{\alpha_{\max}} + \sin(\omega t) \Big|_{\pi - \alpha_{\max}}^{2\pi} \right].
\end{gathered} \tag{3}$$

The values of the integration limits are follows:

$$\sin \alpha_{\max} = \frac{Q_{\max} - x_0}{A}; \quad \cos \alpha_{\max} = \sqrt{1 - \frac{(Q_{\max} - x_0)^2}{A^2}}. \tag{4}$$

Thus, from (3) with the help of (4) we may get:

$$\begin{gathered}
\varkappa_0(A, x_0) = \frac{p_{max}}{2\pi \left(Q_{max} - Q_{min}\right)} \left[2 \operatorname{asin} \left(\frac{Q_{max} - x_0}{A} \right) + \pi \right]; \\
\varkappa(A, x_0) = -\frac{2 p_{max}}{\pi A \left(Q_{max} - Q_{min}\right)} \sqrt{1 - \frac{\left(Q_{max} - x_0\right)^2}{A^2}}; \\
\varkappa'(A, x_0) = 0.
\end{gathered}$$

The resulting harmonic linearization coefficients are used to generate the program by means of a computer algebra system.

5 Software Implementation of the RED Model

The software implementation of the RED model was carried out in two stages. At the first stage, a computer algebra system was employed. With the help of this system the whole time-consuming processing of the formulas is carried out. The resulting expressions are used both in the generation of numerical programs and in the transfer of formulas to the text of articles. Then the resulting formulas are used to generate computational programs. We suggest to use the Julia language [5] as a numerical programming language. It is unlikely that this language is really a silver bullet. However, it has a number of interesting features. This language is positioned as a modern reincarnation of

the FORTRAN language. It supports both the stage of prototyping and writing the final version of the program. This language is intensively developing. All this attracted our attention to this language.

The most interesting system of symbolic calculations for us is the SymPy system [9]. This system appeared as a library of symbolic calculations for the Python language. But the Python language has become a universal glue language. Its application in a variety of projects led to explosive growth of related tools and libraries. Therefore, SymPy developed along with it. Now this is a fairly powerful system of computer algebra. SymPy suits us for the following reasons:

- As an interactive shell, it is convenient to use the Jupyter notepad, which is a component of the system iPython [16], with the REPL ideology.
- Python is actually used as a glue language, that allows you to integrate different software products. In addition, within the SciPy library [14] is supported a large number of output formats.
- The output of SymPy can be naturally passed for further numerical calculations to the NumPy [15] library and various programming languages.

Consider the fragments of the developed software. First, the SymPy library is included.

```
from sympy import *
init_printing(pretty_print=True, use_latex=True,   ↩
     use_unicode=True, fontsize='14pt')
```

In the following fragment the calculations for the nonlinear transfer function H_{nl} are performed.

```
Q_min = symbols('Q_min',real=True)
Q_max = symbols('Q_max',real=True)
p_max = symbols('p_max',real=True)
A = symbols('A',real=True)
x_0 = symbols('x_0',real=True)

sQ_min,sQ_max = symbols('sQ_min,sQ_max',real=True)
sQ_subst = [(sQ_min,sqrt(1- (x_0 -   ↩
     Q_min)**2/A**2)),(sQ_max,sqrt(1- (Q_max -x_0)**2/A**2))]

k0 = 1/(2*pi) *p_max/(Q_max - Q_min) *(2 * asin((Q_max-x_0)/A) + pi)
print(latex(k0))
k1 = 1 /(A * pi) * p_max/(Q_max - Q_min) * (- 2 * sQ_max)
print(latex(k1.subs(sQ_subst)))
k2 = 1 /(A * pi) * p_max/(Q_max - Q_min) * 2 * (Q_max - x_0)/A
print(latex(k2.subs(sQ_subst)))

Hnl = k1 + I * k2
```

For further calculations, the numerator and denominator of the nonlinear transfer function H_{nl} are obtained, as well as the real and imaginary parts.

```
reHnl = re(Hnl)
imHnl = im(Hnl)
nreHnl,dreHnl=fraction(reHnl)
nimHnl,dimHnl=fraction(imHnl)
```

In the following fragment, the initial model is linearized and the transfer function H_l is obtained. This operation is quite labor-intensive.

```
T_p, N, C, w_q = symbols('T_p, N, C, w_q',real=True)
t, p, W, W_T, Q, Qh  = symbols('t, p, W, W_T, Q, \hat{Q}',real=True)
T_f, W_f, p_f, Q_f = symbols('T_f, W_f, p_f, Q_f',real=True)
s, omega = symbols('s, \omega',real=True)
dW,dW_T,dQ,dQh,dp =  ↩
     symbols('\delta{W},\delta{W_T},\delta{Q},\delta{\hat{Q}},  ↩
     \delta{p}', real=True)

T = T_p + Q / C
L_W = 1 / T - W * W_T * p / (2 * T)
L_Q = W * N / T - C
L_Qh = -w_q * C * Qh + w_q * C * Q

W_f = C * T_f / N
p_f = 2 / W_f**2
Qh_f = Q_f

f_subst = [(T, T_f), (2*T, 2*T_f), (W_T, W_f), (W, W_f), (p, p_f)]
s_subst = [(dW_T,dW*(1-s*T_f))]
```

We calculate the variations with automatic substitution of the coupling equations.

```
dL_WdW = L_W.diff(W).subs(f_subst)
dL_WdW_T = L_W.diff(W_T).subs(f_subst)
dLWdQ = L_W.diff(Q).subs(f_subst)
dL_Wdp = L_W.diff(p).subs(f_subst)
dL_QdW = L_Q.diff(W).subs(f_subst)
dL_QdQ = L_Q.diff(Q).subs(f_subst)
d_QhdQh = L_Qh.diff(Qh).subs(f_subst)
dL_QhdQ = L_Qh.diff(Q).subs(f_subst)
dDotW = dL_WdW * dW + dL_WdW_T * dW_T + dLWdQ * dQ + dL_Wdp *dp
dDotQ = dL_QdW * dW + dL_QdQ * dQ
dDotQh = d_QhdQh * dQh + dL_QhdQ * dQ
dWfin = solve((dDotW - dW*s).subs(s_subst),dW)
dQfin = solve(dDotQ - dQ*s,dQ)
dQhfin = solve(dDotQh - dQh*s,dQh)
dW_subst = [(dW,dWfin[0])]
dQfin2 = dQfin[0].subs(dW_subst)
dQ_subst = [(dQ,dQfin2),(dp,1)]
```

We extract the real and imaginary parts.

```
nHl,dHl=fraction(Hl)
```

The export of the resulting expressions to the Julia system.

```
x = symbols('x')
julia_x_subst = [(A,x)]
julia_code(eq_re_fin_omega2.subs(julia_x_subst),assign_to='f(x)')

x1,x2,x0 = symbols('x[1],x[2],x[3]')
julia_subst = [(A,x1),(omega,x2),(x_0,x0)]

print(julia_code(eq_re_fin.subs(julia_subst),assign_to='F[1]'))
print(julia_code(eq_im_fin.subs(julia_subst),assign_to='F[2]'))

omega0_subst = [(omega,0)]
Hl0 = Hl.subs(omega0_subst)
H0 = x_0 + Hl0 * k0
print(julia_code(H0.subs(julia_subst),assign_to='F[3]'))
```

For numerical calculations we use the Julia language. Here is the fragment of the program code, most of which was obtained by using a computer algebra system.

```
using NLsolve

include("parameters.jl")

function f!(F, x::Vector{Float64})
    F = Array{Float64}(length(x))
    F[1] = 2*C.^4.*T_f.^3.*p_max.*w_q.*sqrt(1 - (Q_max -
     x[3]).^2./x[1].^2) + 2*pi*N.*x[1].*(Q_max -
     Q_min).*(-C.^2.*T_f.^3.*w_q.*x[2].^2 +
     C.*N.*T_f.^2.*w_q.*x[2].^2 + 2*C.*N.*w_q -
     C.*T_f.^2.*x[2].^2 - N.*T_f.*x[2].^2)
    F[2] = sqrt((C.^2.*T_f.^2.*w_q + C.*N.*T_f.*w_q +
     2*N)./(C.*T_f - N))./T_f
    F[3] = -C.^3.*T_f.^3.*p_max.*(2*asin((Q_max - x[3])./x[1]) -
     pi)./(8*pi*N.^2.*(Q_max - Q_min)) + x[3]
end

res = nlsolve(f!, x_init).zero
```

This program uses the library for solving systems of nonlinear equations NLsolve. The parameters are set in a separate file. The results are output in the form of a text file.

6 Influence of Parameters on Occurrence of Self-oscillations

Based on the developed algorithm, one can investigate the dependence of self-oscillation regions on the parameters of the RED algorithm. Naturally, we can consider other variants of RED-like algorithms.

As an illustration, we give a concrete example. Let's set the following parameters of the RED algorithm: the number of sessions $N = 60$, round-trip time $T_p = 0.5\,\text{s}$, thresholds $Q_{\min} = 75$ packages and $Q_{\max} = 150$ packets, drop probability $p = 0.1$, parameter $w_q = 0.002$. Let us investigate the dependence of self-oscillation on the link capacity C. We obtain that the transition to the self-oscillatory regime occurs at $C_a = 15\,\text{Mbps}$. That is, for $C \geqslant C_a$ the system will be in self-oscillation mode. This can be demonstrated using the network simulation tool NS-2 [3] (see Figs. 3 and 4). For the transition point to the self-oscillating regime, we obtain the following self-oscillation parameters: the frequency of self-oscillations $\nu = \omega/2\pi = 0.64$ Hz, the amplitude of the oscillations $A = 145$ packets.

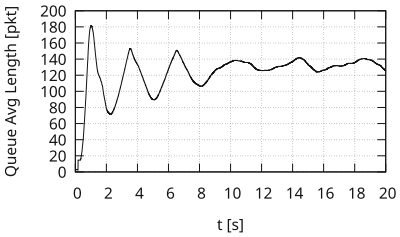

Fig. 3. Average queue length at link capacity $C = 5\,\text{Mbps}$

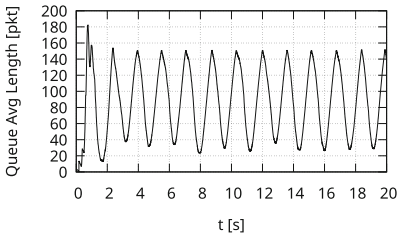

Fig. 4. Average queue length at link capacity $C = 15\,\text{Mbps}$

7 Conclusion

The authors developed the software package for RED-like Active Queue Management algorithms investigation. The software package includes components for symbolic and numerical calculations. In constructing the calculation technique, the asymmetric nature of self-oscillations arising in the system was taken into account.

Acknowledgments. The publication has been prepared with the support of the "RUDN University Program 5–100" and funded by Russian Foundation for Basic Research (RFBR) according to the research project No 16-07-00556.

References

1. Brockett, R.: Stochastic analysis for fluid queueing systems. In: Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304), vol. 3, pp. 3077–3082. IEEE (1999). https://doi.org/10.1109/CDC.1999.831407
2. Hollot, C.V.V., Misra, V., Towsley, D., Gong, W.-B.: On designing improved controllers for AQM routers supporting TCP flows. In: Proceedings of the IEEE INFOCOM 2001 Conference on Computer Communications, Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213), vol. 3, pp. 1726–1734. IEEE (2001). https://doi.org/10.1109/INFCOM.2001.916670
3. Issariyakul, T., Hossain, E.: Introduction to Network Simulator NS2. Springer, Boston (2012)
4. Jenkins, A.: Self-oscillation. Phys. Rep. **525**(2), 167–222 (2013). https://doi.org/10.1016/j.physrep.2012.10.007
5. Joshi, A., Lakhanpal, R.: Learning Julia. Packt Publishing, Birmingham (2017)
6. Korolkova, A.V., Velieva, T.R., Abaev, P.A., Sevastianov, L.A., Kulyabov, D.S.: Hybrid simulation of active traffic management. In: Proceedings of the 30th European Conference on Modelling and Simulation, pp. 685–691 (2016). https://doi.org/10.7148/2016-0685
7. Kryloff, N., Bogolïuboff, N.: Les methodes symboliques d l Mecanique non Lineaire dans leur application a l'etude de la resonance dans l'oscillateur. Bulletin del'Academie des Sciences de l'URSS. Classe des sciences mathematiques et na (1), 7–34 (1934)
8. Kulyabov, D.S., Korolkova, A.V., Velieva, T.R., Eferina, E.G., Sevastianov, L.A.: The methodology of studying of active traffic management module self-oscillation regime. In: Zamojski, W., Mazurkiewicz, J., Sugier, J., Walkowiak, T., Kacprzyk, J. (eds.) DepCoS-RELCOMEX 2017. Advances in Intelligent Systems and Computing, vol. 582, pp. 215–224. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-59415-621
9. Lamy, R.: Instant SymPy Starter. Packt Publishing, Birmingham (2013)
10. Lautenschlaeger, W., Francini, A.: Global synchronization protection for bandwidth sharing TCP flows in high-speed links. In: Proceedings of the 16-th International Conference on High Performance Switching and Routing, IEEE HPSR 2015, Budapest, Hungary (2015)
11. Misra, V., Gong, W.B., Towsley, D.: Stochastic differential equation modeling and analysis of TCP-windowsize behavior. In: Proceedings of PERFORMANCE, vol. 99 (1999)
12. Misra, V., Gong, W.B., Towsley, D.: Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. ACM SIGCOMM Comput. Commun. Rev. **30**(4), 151–160 (2000). https://doi.org/10.1145/347057.347421
13. Nyquist, H.: Regeneration theory. Bell Syst. Tech. J. **11**(1), 126–147 (1932). https://doi.org/10.1002/j.1538-7305.1932.tb02344.x
14. Oliphant, T.E.: Python for scientific computing. Comput. Sci. Eng. **9**(3), 10–20 (2007). https://doi.org/10.1109/MCSE.2007.58
15. Oliphant, T.E.: Guide to NumPy, 2nd edn. CreateSpace Independent Publishing Platform, Santa Monica (2015)
16. Perez, F., Granger, B.E.: IPython: a system for interactive scientific computing. Comput. Sci. Eng. **9**(3), 21–29 (2007). https://doi.org/10.1109/MCSE.2007.53

17. Ren, F., Lin, C., Wei, B.: A nonlinear control theoretic analysis to TCP-RED system. Comput. Netw. **49**(4), 580–592 (2005). https://doi.org/10.1016/j.comnet.2005.01.016
18. Velieva, T.R., Korolkova, A.V., Kulyabov, D.S.: Designing installations for verification of the model of active queue management discipline RED in the GNS3. In: 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), pp. 570–577. IEEE Computer Society (2015). https://doi.org/10.1109/ICUMT.2014.7002164