STABILISATOR SISTEM TENAGA BERBASIS JARINGAN SYARAF TIRUAN BERULANG UNTUK SISTEM MESIN TUNGGAL

Widi Aribowo
Fakultas Teknik, Universitas Negeri Surabaya
Kampus Unesa Jalan Ketintang, Surabaya 60231
Telp. (031) 8280009, Pes 500, 510, Fax (031) 8280796
e-mail: w1ed1e1@yahoo.com

Abstract

In this paper, recurrent neural network (RNN) is used to design power system stabilizer (PSS) due to its advantage on the dependence not only on present input but also on past condition. A RNN-PSS is able to capture the dynamic response of a system without any delays caused by external feedback, primarily bythe internal feedback loop in recurrent neuron. In this paper, RNNPSS consists of a RNN-identifier and a RNN-controller. The RNN-Identifier functions as the tracker of dynamics characteristics of the plant, while the RNN-controller is used to damp the system's low frequency oscillations. Simulation results using MATLAB demonstrate that the RNNPSS can successfully damp out oscillation and improve the performance of the system.

Keywords*: controller, identifier, power system stabilizer, recurrent neural network, RNNPSS*

Abstrak

Pada penelitian ini, jaringan syaraf tiruan berulang (RNN) digunakan untuk mendesain stabilisator sistem tenaga (PSS) karena mempunyai keunggulan bahwa keluarannya tidak hanya tergantung pada masukan saat ini, tetapi juga tergantung pada kondisi masukan waktu sebelumnya. RNNPSS dapat menangkap respon dinamik dari sistem tanpa waktu tunda umpan balik eksternal karena neuron berulang mempunyai umpan balik internal. RNNPSS yang akan digunakan dalam penelitian terdiri dari dua komponen utama, yaitu RNN-identifier dan RNN-controller. RNN-identifier berfungsi untuk mengetahui karekteristik dinamika sistem, sedangkan RNN-controller berfungsi untuk meredam osilasi frekuensi rendah. Dari hasil simulasi RNNPSS mampu meredam osilasi frekuensi rendah dan mampu memperbaiki performansi sistem. Simulasi dilakukan dengan perangkat lunak MATLAB.

Kata kunci*: controller, identifier, jaringan syaraf berulang, RNNPSS, stabilisator sistem tenaga*

1. PENDAHULUAN

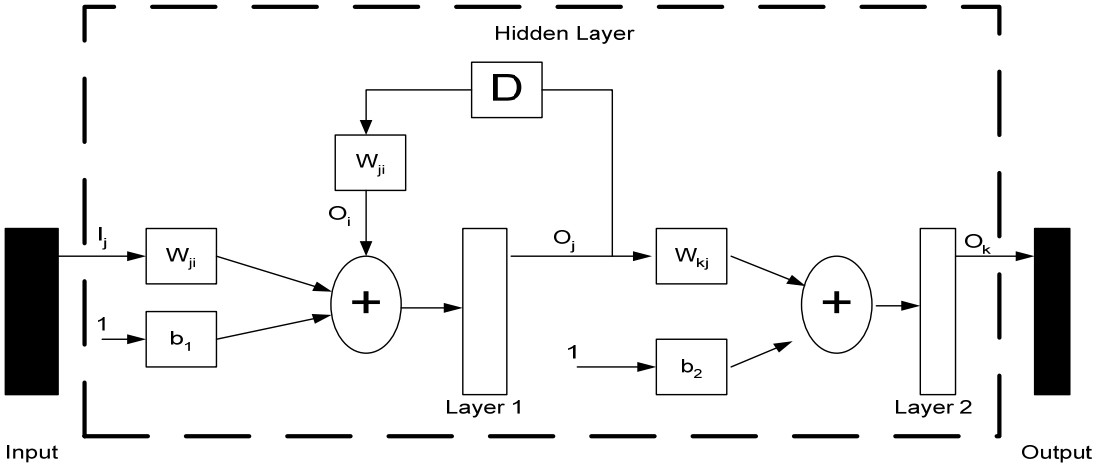
Stabilitas sistem tenaga listrik didefinisikan sebagai kemampuan suatu sistem tenaga listrik atau bagian komponen untuk mempertahankan sinkronisasi dan keseimbangan sistem. Dari keadaan operasional yang stabil dari sistem tenaga listrik, terdapat keseimbangan antara daya input mekanis pada primer mover dengan daya output listrik pada sistem. Dalam keadaan ini semua generator berputar pada keadaan sinkron [1]. Penggunaan AVR *(automatic voltage regulator)* dengan penguatan yang tinggi untuk menambah kestabilan sistem tenaga listrik dapat menimbulkan ketidakstabilan saat kondisi-kondisi khusus (kondisi yang mendekati limit daya penyaluran). Di saat kondisi-kondisi tersebut bila ada perubahan kecil beban maka umpan balik yang ada dapat menyebabkan redaman negatif [2].

PSS adalah piranti dengan fungsi transfer tertentu yang dapat diatur besaran dan fasa. Kedalam PSS ini diinjeksikan sinyal yang antara lain sefasa dengan daya. Kemudian keluaran PSS dimasukkan ke rangkaian eksitasi [3]. Dengan mengatur fasa PSS, sinyal keluaran eksitasi akan menghasilkan redaman positif yang berfungsi mengkompensir redaman negatif. Agar redaman postif PSS cukup untuk mengkompensir redaman negatif, komponen penguat *(amplifier)* yang ada di dalam PSS harus diatur [4].

Perkembangan sistem tenaga listrik yang semakin cepat dan masalah yang semakin beragam, menuntut penyelesaian waktu nyata. Oleh karena itu, pada makalah ini diusulkan desain jaringan syaraf tiruan berulang (*recurrent neural network*, RNN) untuk stabilisator sistem tenaga (PSS) yang dapat merespon perubahan performa sistem secara langsung. Pada penelitian ini RNNPSS diaplikasikan (disimulasikan dengan perangkat lunak MATLAB) pada sistem mesin tunggal. Pengaplikasian RNNPSS pada sistem mesin tunggal dititik beratkan pada kinerja RNNPSS terhadap osilasi frekuensi rendah dan perbaikan performansi sistem.

2. METODE PENELITIAN

Kelemahan *backpropagation neural network* adalah terbatasnya pada pelatihan fungsi statis dan keluaran hanya tergantung pada kondisi masukan saat ini, sehingga tidak mampu kalau ada perubahan pola data masukan. Kelemahan tersebut mendorong dikembangkannya menjadi RNN (Gambar 1). RNN adalah NN dengan fasilitas umpan balik menuju neuron itu sendiri maupun neuron yang lain, sehingga aliran informasi dari masukan mempunyai arah jamak (*multidirectional*). Keluaran RNN tidak hanya tergantung pada masukan saat itu saja, tetapi juga tergantung pada kondisi masukan NN untuk waktu lampau. Kondisi ini dimaksudkan untuk menampung kejadian lampau diikutkan pada proses komputasi. Hal ini penting untuk problematika yang cukup rumit, dan tanggapan keluaran NN berkaitan dengan variasi waktu (*time-varying*), sehingga NN memiliki kepekaan terhadap waktu dengan memori kondisi lampau [5].



Gambar 1. Struktur jaringan syaraf tiruan berulang (RNN)

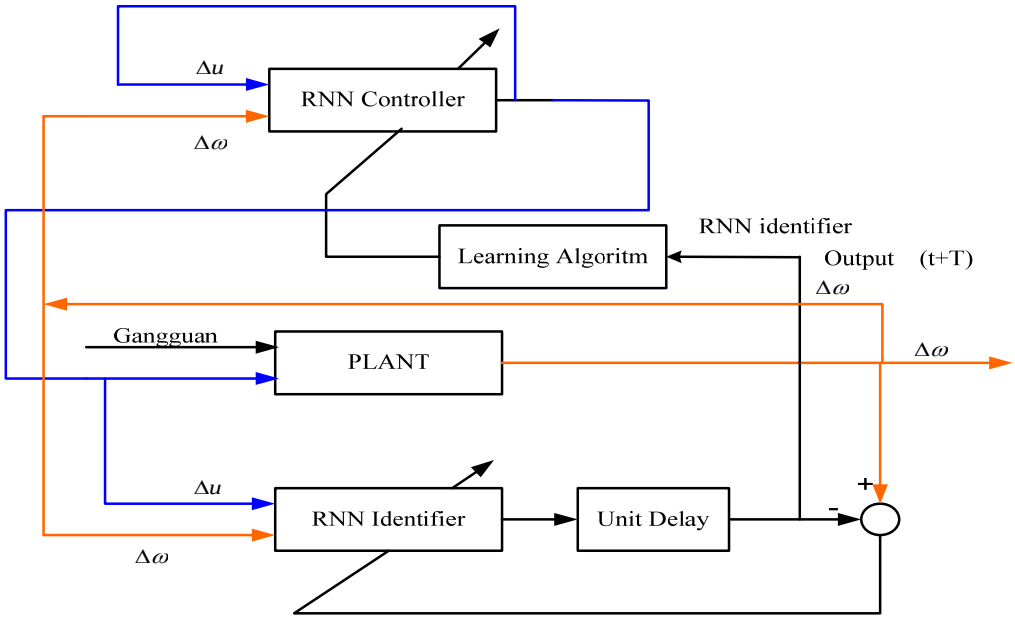
Model RNNPSS yang dipasang *parallel* antara *RNN-identifier* dan *RNN–Controller* adalah seperti terlihat pada Gambar 2.

2.1. RNN *Identifier* (RNN-i)

Skema dari RNN *Identifier* menggunakan model *forward*, yang dipasang paralel dengan sistem dapat dilihat pada Gambar 2, dan strukturnya ditunjukkan pada Gambar 3. RNN-i ini membaca keluaran dari *plant* $\omega(t)$ dan berusaha untuk meniru bentuk gelombang dari keluaran *plant* dengan cara membandingkan keluaran *plant* dengan output RNN-i. Jika terjadi perbedaan galat, maka sinyal galat tersebut dikirim ke RNN -i untuk dilakukan proses pembelajaran untuk memperkecil galat. RNN-i ini mempunyai dua masukan, yaitu $\Delta\omega$ dan Δu . $\Delta\omega$ adalah keluaran *plant* dan Δu adalah keluaran PSS. Sebagai masukan awal RNN-i menggunakan keluaran dari PSS konvensional untuk proses pelatihan. Secara matematis dapat ditulis sebagaimana persamaan (1).

$$Xi(t) = [\Delta w, \Delta u] \quad (1)$$

$\Delta\omega$ diambil dari nilai kecepatan mesin sinkron yang terakhir yang disensor dengan interval waktu konstan 200 ms .



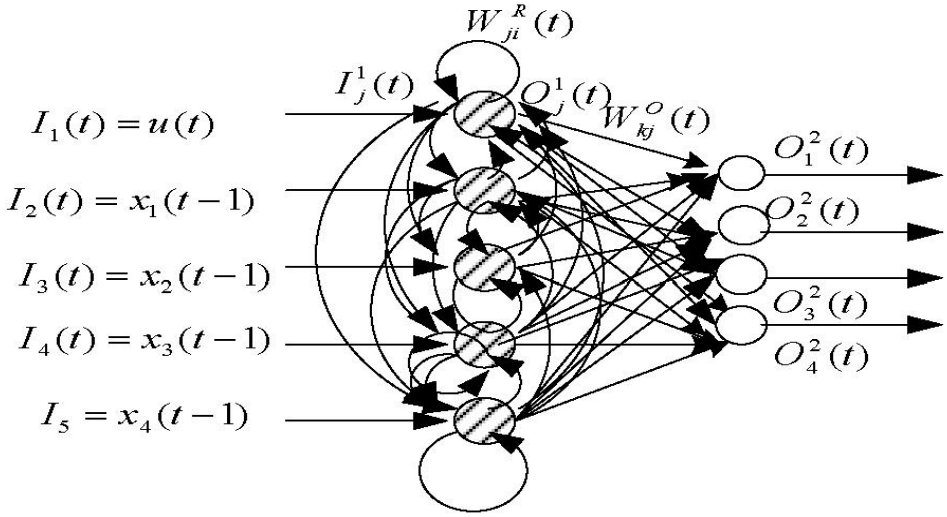
Gambar 2. Model RNNPSS

$$\Delta w = [w(t), w(t-T), w(t-3T)] \quad (2)$$

Δu diambil dari tiga aksi kendali terakhir yang telah dilakukan (simulasi K-PSS).

$$\Delta u = [u(t-T), u(t-2T), u(t-3T)] \quad (3)$$

T adalah sampling periode, ω adalah deviasi kecepatan anguler terhadap kecepatan sinkron dalam rad/s. *u* adalah output dari *controller*.



Gambar 3. Struktur RNN-i [6]

Keluaran fungsi agregat *(output aggregation function)* adalah dihitung dengan persamaan (4*)* dan (5)*.*

$$Oi_j(t) = f(Pi_j(t)) = \frac{e^{Pi_j(t)} - e^{-Pi_j(t)}}{e^{Pi_j(t)} + e^{-Pi_j(t)}} \quad (4)$$

$$Pi_j(t) = \sum_{i=1}^{j} Ii_j \cdot Wi_{ji} + bi_1 + \sum_{i=1}^{j} Oi_i \cdot Wi_{ji} \quad (5)$$

Keluaran pada bagian RNN-i, adalah dihitung dengan persamaan (6)

$$Pi_k(t) = \sum_{i=1}^{k} Oi_j(t) \cdot Wi_{kj} + bi_2 \quad (6)$$

$$O_k(t) = P_k(t) \quad (7)$$

Keluaran RNN *Identifier* ini kemudian dibandingkan dengan keluaran *plant*, sehingga didapatkan galat Galat antara sistem dan keluaran *RNN identifier* pada unit tunda digunakan sebagai *training RNN identifier.* Galat kuadrat sebanding dengan *performance index* sebagaimana persamaan (8)

$$E_i(t) = \frac{1}{2}\sum_{k=1}^{n}(wi_k(t) - Oi_k(t))^2 \quad (8)$$

Bobot W_{ji} dan W_{kj} dapat diatur dengan menggunakan *steepest descent algorithm* sebagaimana persamaan (9).

$$\mathrm{W_{ji}}(t+1) = \mathrm{W_{ji}}(t) + \Delta\mathrm{W_{ji}}(t) = \mathrm{W_{ji}}(t) - \eta_I \frac{\partial Ei(t)}{\partial \mathrm{W_{ji}}(t)} \quad (9)$$

$$\mathrm{W_{kj}}(t+1) = \mathrm{W_{kj}}(t) + \Delta\mathrm{W_{kj}}(t) = \mathrm{W_{kj}}(t) - \eta_I \frac{\partial Ei(t)}{\partial \mathrm{W_{kj}}(t)} \quad (10)$$

Dengan η_I adalah *learning rate* dari RNN-I, *gradient error E(t)* dari *weight* W_{ji} dan *weight* W_{kj} adalah sebagai berikut:

$$\frac{\partial Ei(t)}{\partial \mathrm{W_{kj}}(t)} = -(wi_k(t) - Oi_k(t))Oi_j(t) \quad (11)$$

$$\frac{\partial Ei(t)}{\partial \mathrm{W_{ji}}(t)} = -\sum[wi_k(t) - Oi_k(t)Wi_{kj}(t)(1 - Oi_j(t)^2)Oi_j(t-1)] \quad (12)$$

2.2. RNN-*Controller* (RNN-c)

Struktur RNN-c ditunjukkan pada Gambar 3. Secara matematis, lapis input RNN-c dapat dihitung sebagaimana persamaan (13) dan (14), sedangkan lapis keluaran dapat dihitung dengan persamaan (15).

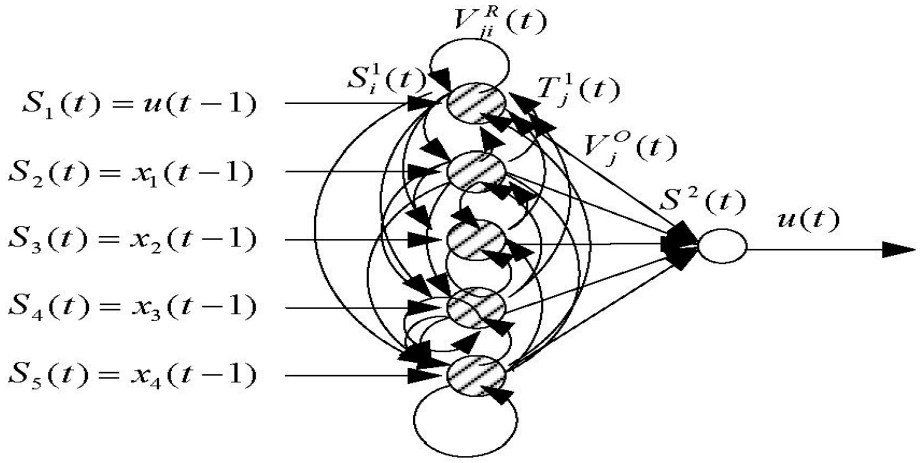
$$S_j^1(t) = S_j(t) + \sum_{i=1}^{Mc} Vc_{ji}^r(t) T_i^1(t-1), \qquad j = 1,...Mc \tag{13}$$

dan

$$T_j^1(t) = f(S_j^1(t)) = \frac{e^{S_j^1(t)} - e^{-S_j^1(t)}}{e^{S_j^1(t)} + e^{-S_j^1(t)}} \qquad j = 1,...Mc \tag{14}$$

$$u(t) = S^2(t) = \sum_{j=1}^{Mc} Vc_j^o(t) T_j^1(t) \tag{15}$$

dengan $Wc(t)$ adalah matriks bobot *neural controller* pada *time instant* t dan $u(t)$ adalah de-normalisasi untuk mendapatkan aksi control actual dan kemudian dikirim ke *plant* dan RNN-i.



Gambar 4. Struktur RNN-c [6]

Indeks performansi dari *neurocontroller* adalah dihitung dengan persamaan (16).

$$E_c(t) = \frac{1}{2}\sum_{k=1}^{n}(r_k(t) - x_k(t))^2 \tag{16}$$

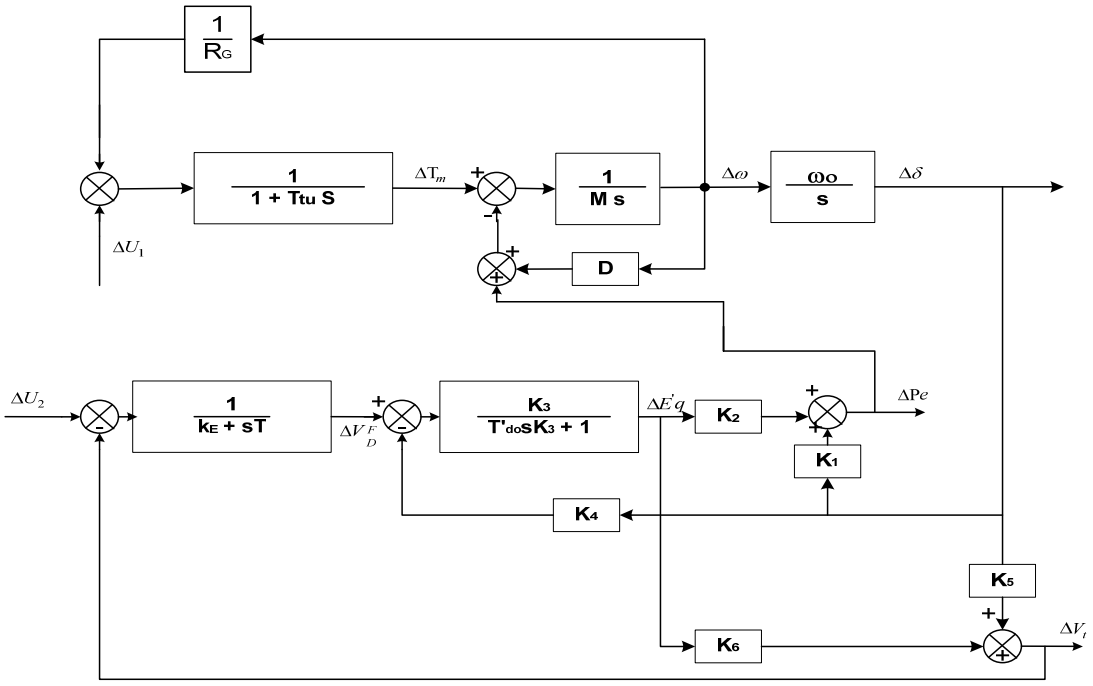
dengan $x_k(t)$ adalah keluaran *plant* dan $r_k(t)$ adalah referensi yang dinginkan, sedangkan bobot dari pemutakhiran RNN *controller* adalah dihitung dengan persamaan (17).

$$Vc_{ji}(t+1) = Vc_{ji}(t) + \Delta Vc_{ji}(t) = Vc_{ji}(t) - \eta_I \frac{\partial Ec(t)}{\partial Vc_{ji}(t)} \tag{17}$$

dengan ηc adalah laju pembelajaran untuk *RNN controller*

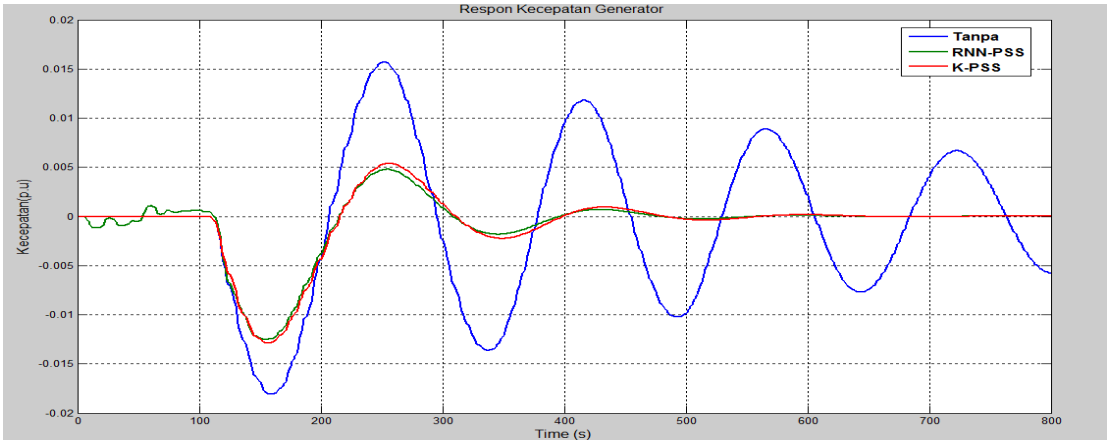
3. SIMULASI DAN ANALISIS

Generator dimodelkan dalam bentuk model *Philips Heffron*, dan dapat dilihat pada Gambar 5.



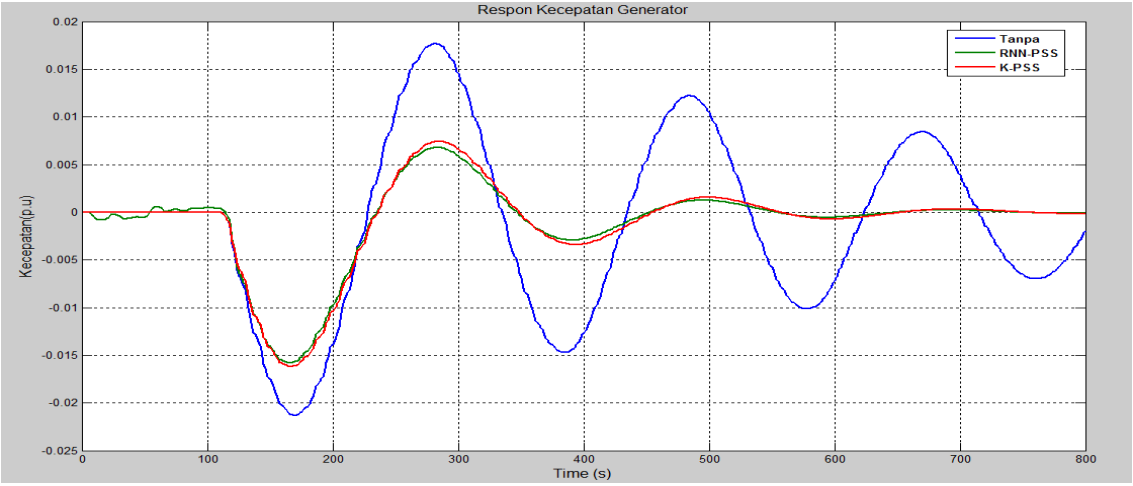
Gambar 5. Diagram Blok Dinamika Single Mesin [1].

Data pelatihan untuk meredam osilasi kecepatan adalah data keluaran *plant* berupa kecepatan dengan variasi gangguan antara 0.1-1,0 pu. Fungsi aktifasi yang digunakan adalah *tansig*, dan *purelin*. Jumlah neuron yang digunakan adalah 5. Fungsi pelatihan yang diterapkan adalah *trainlm*. Struktur jaringan yang digunakan pada pelatihan ini adalah terdiri dari tiga lapis (layer), yaitu lapis masukan, lapis tersembunyi dan lapis keluaran Setelah dilakukan proses pemetaan, maka langkah selanjutnya adalah memasang RNNPSS di sistem. Pada penelitian ini pembebanan diasumsikan sebesar: P=1.0 pu; Vt=1.0 pu; Pf=0.85 pu, P=0.5 pu; Vt=1.0 pu; Pf=0.85 pu, dan P=0.1 pu; Vt=1.0 pu; Pf=0.85 pu. Gangguan sebesar 0.1 p.u diinjeksikan kedalam *plant*, dan diperoleh keluaran kecepatan untuk *plant* seperti Gambar 6.

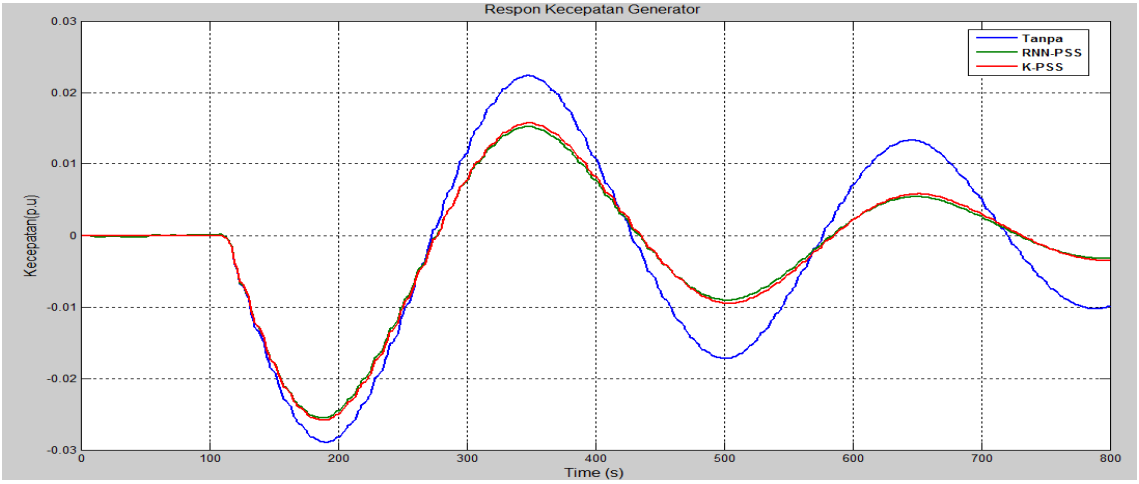


Gambar 6. Grafik kcepatan mesin tunggal kondisi P=1.0 pu; Vt=1.0 pu; Pf=0.85 pu.

Dari Gambar 6, RNNPSS dapat menurunkan *overshoot* kecepatan menjadi 0,0049 p.u dari kondisi semula yaitu 0,016 p.u dan lebih baik dari konvensional PSS yang hanya mampu menurunkan sebesar 0.006 p.u.



Gambar 7. Grafik kecepatan mesin tunggal kondisi P=0.5 pu; Vt=1.0 pu; Pf=0.85 pu.



Gambar 8. Grafik kecepatan mesin tunggal kondisi P=0.1 pu; Vt=1.0 pu; Pf=0.85 pu.

Dari Gambar 7, RNNPSS dapat menurunkan *overshoot* kecepatan menjadi 0,006 p.u dari kondisi semula yaitu 0,018 p.u dan lebih baik dari konvensional PSS yang hanya mampu menurunkan sebesar 0.0075 p.u. Dari Gambar 8, RNNPSS dapat menurunkan *overshoot* kecepatan menjadi 0,015 p.u dari kondisi semula yaitu 0,024 p.u dan lebih baik dari konvensional PSS yang hanya mampu menurunkan sebesar 0.016 p.u.

4. SIMPULAN

RNNPSS mampu memperbaiki performansi sistem generator di tempat RNNPSS dipasang. Keberhasilan dari desain stablisator sistem tenaga jaringan syaraf tiruan berulang (RNNPSS) ini sangat bergantung data dan proses pembelajaran yang benar.

DAFTAR PUSTAKA

[1]. Xu D, and He R. *ANN Based Multiple Power System Stabilizers Adaptive and Coordinates Control*. PowerCon 2002. International Conference Proceeding. 2002; 1: 361-364.
[2]. Rogers GY. The application of Power System Stabilizers to a Multigenerator Plant. *IEEE Trans. Power Syst*. 2000; 15(1): 350–355.

[3]. Chen CJ, Chen TC. Design of a Power System Stabilizer using a New Recurrent Network. *International Journal of Innovative Computing, Information and Control*. 2007; 3(4): 907-918.
[4]. You R, Eghbali HJ, Nehrir M. An Online Adaptive Neuro-Fuzzy Power System Stabilizer for Multi-machine Systems. *IEEE Transaction on Power Systems*. 2003; 18(1): 128-135.
[5]. Chaturvedi DK and Malik OP. Neural Network Controller for Power System Stabilizer. *Journal of the Institution of Engineers*. 2004; 85(1): 138-145.
[6]. Lin CH. Adaptive Recurrent Fuzzy Neural Network Control for Synchronous Reluctance Motor Servo Drive. *IEE Proc. Power Applications*. 2004; 151(6): 711-724.