ELINVO (Electronics, Informatics, and Vocational Education), November 2019; 4(2):169-175
ISSN 2580-6424 (printed), ISSN 2477-2399 (online,) DOI: 10.21831/elinvo.v4i2. 28347

Studi Komparasi Simulasi Sistem Kendali PID pada MATLAB, GNU *Octave*, *Scilab,* dan *Spyder*

Ardy Seto Priambodo[1], **Arya Sony**[2]
[1,2]Program Studi Pendidikan Teknik Elektronika Fakultas Teknik Universitas Negeri Yogyakarta
E-mail: ardyseto@uny.ac.id

ABSTRACT

The speed of the process in a simulation process is essential to determine the number of iterations that can be performed. In this study, a comparison between MATLAB, GNU Octave, Scilab, and Spyder software in a PID Control System simulation was observed. The process that is the object of this research is the PID control system used to control the plant in the form of a DC motor model. The transfer function of a DC motor is a model of a DC motor derived from the 2nd Newton's law and the voltage Kirchoff law. Simulations performed on software using two ways, namely program code and block programming. The shortest number of lines of program code is MATLAB with 17 lines, and the longest is Spyder with 20 lines of program code. Block programming can only be done on MATLAB and Scilab because other software does not have this feature. Program code execution time is observed for each software, and Spyder is the fastest with a time of 0.0682 seconds. The plot results of each software do not have much difference. MATLAB is the best software with its features, but a license fee is required to use it.

***Keyword**: PID, MATLAB, GNU Octave, Scilab, Spyder*

ABSTRAK

Kecepatan proses dalam sebuah proses simulasi adalah suatu yang penting untuk menentukan jumlah proses iterasi yang dapat dilakukan. Dalam penelitian ini perbandingan antara perangkat lunak MATLAB, GNU *Octave*, *Scilab*, dan *Spyder* dalam simulasi Sistem Kendali PID diamati. Proses yang menjadi obyek dalam penelitian ini adalah sistem kendali PID yang digunakan untuk mengendalikan *plant* yang berupa model motor DC. Fungsi alih dari motor DC merupakan model dari motor DC yang diturunkan dari hukum Newton ke-2 dan hukum Kirchoff tegangan. Simulasi dilakukan pada perangkat lunak menggunakan 2 cara yaitu kode program dan *block programming*. Jumlah baris kode program yang terpendek adalah MATLAB dengan 17 baris dan yang terpanjang adalah *Spyder* dengan 20 baris kode program. Untuk *block programming* hanya dapat dilakukan pada MATLAB dan *Scilab* karena perangkat lunak yang lain tidak memiliki fitur tersebut. Waktu eksekusi kode program diamati padan masing-masing perangkat lunak dan *Spyder* adalah yang tercepat dengan waktu 0.0682 detik. Hasil plot dari masing-masing perangkat lunak tidak banyak memiliki perbedaan. MATLAB adalah perangkat lunak yang terbaik dengan fiturnya namun diperlukan biaya lisensi dalam menggunakannya.

Kata Kunci: PID, MATLAB, GNU *Octave, Scilab, Spyder*

PENDAHULUAN

Sistem kendali merupakan salah satu bidang dalam ilmu teknik yang memiliki peran penting dalam kehidupan manusia baik dalam kehidupan sehari-hari maupun dunia industri [1]. Mulai dari peralatan seperti mesin cuci, pendingin ruangan (AC), bahkan hingga robot pada industri manufaktur, pesawat tanpa awak semua ditanamkan sistem kendali agar peralatan tersebut dapat bekerja dengan baik [2],[3]. Sehingga penguasaan terhadap pemahaman sistem kendali menjadi sangat penting baik dari sisi teori maupun praktiknya.

Ada beberapa *tools* yang dapat digunakan untuk melakukan simulasi sistem kendali. *Tools* tersebut adalah MATLAH, GNU *Octave*, *Scilab*,

dan *Spyder* yang merupakan IDE dari bahasa pemograman Python.

MATLAB adalah sebuah bahasa pemrograman tingkat tinggi dan tertutup yang ditujukan untuk komputasi numeris yang dikembangkan oleh *MathWorks, INC* [4]. MATLAB sangat memiliki banyak *tools* yang dapat digunakan banyak disiplin ilmu. Selain itu MATLAB juga memiliki *library* yang dapat digunakan untuk menjalankan simulasi model matematika, pengolahan sinyal digital dan sistem kendali. Selain menggunakan kode-kode program dalam melakukan simulasi dapat pula menggunakan Simulink yaitu pemrograman yang menggunakan blok.

Selain MATLAB terdapat perangkat lunak GNU *Octave*, yaitu aplikasi *free* dan *open source* yang ditujukan untuk komputasi numerik baik linier maupun non-linier [5]. GNU *Octave* dikembangkan oleh GNU yang mengusung perangkat lunak *open source.* Hampir semua sintaks yang ada pada MATLAB kompatibel dengan GNU *Octave* sehingga kode-kode yang dituliskan pada MATLAB dapat dijalankan pula di GNU *Octave*. Lisensi dari GNU *Octave* adalah GNU *General Public License.*

Scilab adalah perangkat lunak *open source* yang digunakan untuk menyelesaikan permasalahan dalam komputasi numerik dan visualisasi data [6]. Pada awalnya pengembang aplikasi ini adalah INRIA dan ENPC namun sekarang pengembangan dari perangat lunak ini ada dibawah konsorsium *Scilab*. Perangkat lunak dengan lisensi GNU GPL ini sangat mirip dengan MATLAB Simulink karena juga terdapat fitur pemograman menggunakan blok yaitu xcos.

Spyder merupakan singkatan dari *The Scientific Python Development Environment.* Sesuai dengan kepanjangannya, Sypder dibuat untuk permasalahan-permasalahan terkait saintifik yang mana bahasa pemrograman yang digunakan adalah Python [7]. Tampilan dari *Spyder* mirip sekali dengan MATLAB sehingga ada beberapa yang menganggap *Spyder* dapat digunakan sebagai pengganti MATLAB.

Pada penelitian yang dilakukan oleh A.P. Leros [8] yang melakukan perbandingan dan benchmarking antara MATLAB dan GNU *Octave* berdasarkan waktu eksekusi dan tes performa didapatkan hasil bahwa GNU *Octave* adalah sebuah software yang dapat digunakan untuk proses saintifik yang gratis sebagai pengganti MATLAB.

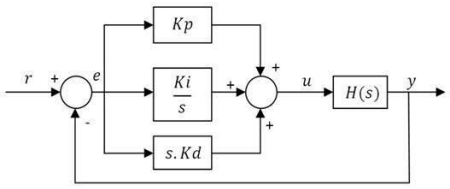
Tujuan dari penelitian ini adalah membandingkan hasil dari simulasi sistem kendali PID yang akan digunakan untuk menstabilkan sebuah sistem dengan model berbentuk fungsi transfer yang akan diaplikasikan dengan menggunakan Matlab, GNU *Octave*, *Scilab*, dan *Spyder*.

METODE

Pada penelitian ini membandingkan Matlab, GNU *Octave*, *Scilab* dan *Spyder* dalam menjalankan simulasi sistem kendali PID. Ada beberapa hal yang akan dibandingkan yaitu: 1) Kode baris program, 2) Hasil plot grafik respon, 3) Waktu eksekusi program, 4) Fitur pemrograman menggunakan blok diagram.

Sistem kendali PID yang dijalankan bekerja dengan sebuah sistem dengan model yang berbentuk fungsi transfer dari sebuah model motor DC. Sistem kendali PID adalah sebuah sistem kendali yang terdiri dari 3 komponen. Ketiga komponen tersebut adalah Kendali Proporsional, Integral dan Derivatif yang kemudian disingkat menjadi kendali PID [9].

Sistem kendali PID merupakan sistem kendali konvensional yang masih banyak digunakan pada dunia industri karena mudah diimplementasikan, mudah dilakukan penalaan parameter yang digunakan, dan terbukti handal dalam mengendalikan berbagai jenis obyek sistem [10]. Diagram blok dari sistem kendali PID ditunjukkan pada gambar 1.



Gambar 1. Diagram Blok dari Sistem Kendali PID (Proporsional-Integral-Derivatif)

Pada gambar 1, $H(s)$ merupakan model dari *plant* yang ingin dikendalikan, dan dalam penelitian ini adalah model matematika dari motor DC yang direpresentasikan menggunakan fungsi alih. Blok sistem kendali terlihat pada blok sebelum *plant* yaitu Proporsional, Integral, dan Derivatif yang ditandai dengan adanya K_p, K_i, K_d. Simbol e atau *error* adalah selisih antara r yang merupakan set poin dengan pembacaan umpan balik dari sistem yang dalam aktual biasanya menggunakan sensor.

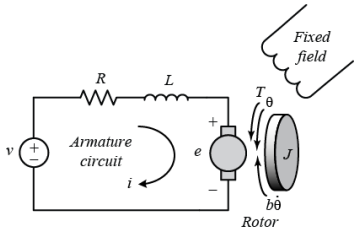
Keluaran dari sistem kendali PID atau u yang merupakan masukan dari *plant* dapat dihitung pada domain waktu yang ditunjukkan pada persamaan (1).

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de}{dt} \qquad (1)$$

Dimana $u(t)$ merupakan keluaran dari sistem kendali PID. K_p, K_i, K_d adalah koefisien untuk Proporsional, Integral dan Derivatif. $e(t)$ merupakan kesalahan yang dihitung dari selisih antara set poin dengan nilai yang terukur. Transformasi laplace dari persamaan (1) ditunjukkan pada persamaan (2).

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \qquad (2)$$

Penelitian ini menggunakan *plant* yang berupa model sebuah model motor DC. Sistem kendali PID yang dirancang bertujuan untuk mengatur kecepatan dari motor DC. Rangkaian elektronika dari sebuah motor DC ditunjukkan pada gambar 2.



Gambar 2. Rangkaian elektronika dari sebuah Motor DC

Pada gambar 2 ditunjukkan v adalaj tegangan sumber yang bekerja pada *armature circuit* dan keluaran dari motor dc ini adalah kecepatan putar $\dot{\theta}$. Dengan asumsi rotor dan *shaft* adalah rigid kita dapat menentukan bahwa torsi yang dihasilkan sebading dengan kecepatan putar dari *shaft*. Tabel 1 menunjukkan parameter fisik yang digunakan.

Tabel 1. Parameter fisik dari motor DC [11]

Simbol	Keterangan	Nilai
J	Momen inersia	$0.01\ kg.m^2/s^2$
b	Konstanta gaya gesek mekanikal	$0.1\ Ns/m$
K_e	Konstanta gaya gerak listrik	$0.01\ Nm/A$
K_t	Konstanta torsi	$0.01\ Nm/A$
R	Resistansi	$1\ \Omega$
L	Induktansi	$0.5\ H$

Secara umum, torsi yang dihasilkan oleh motor DC sebanding dengan arus jangar dan besar medan magnet yang terjadi. Dalam kasus ini berasumsi besar medan magnet adalah konstan sehinga torsi sebanding dengan arus i dengan faktor konstanta K_t yang persamaannya ditunjukkan pada persamaan (3). Besar konstanta *back emf* atau e adalah sebanding dengan kecapatan putar dari *shaft* dengan faktor konstanta K_e yang persamaannya ditunjukkan pada persamaan (4). dalam satuan SI, torsi motor dan konstanta *back emf* adalah sama sehingga $K_t = K_e$ oleh karena itu selanjutnya akan digunakan simbol K untuk merepresentasikan keduanya.

$$T = K_t i \quad (3)$$
$$e = K_e \dot{\theta} \quad (4)$$

Dari gambar 2 kita dapat memperoleh persamaan berdasarkan hukum newton ke-2 dan hukum kirchoff tegangan yang ditunjukkan pada persamaan (5) dan (6).

$$J\ddot{\theta} + b\dot{\theta} = K_i \quad (5)$$
$$L\frac{di}{dt} + Ri = V - K\dot{\theta} \quad (6)$$

Dengan mengaplikasikan transformasi laplace pada persamaan (5) dan (6) kita dapat menuliskan persamaan tersebut sehingga memiliki nilai s yang ditunjukkan pada persamaan (7) dan (8).

$$s(Js + b)\Theta(s) = KI(s) \quad (7)$$
$$(Ls + R)I(s) = V(s) - Ks\Theta(s) \quad (8)$$

Persamaan (7) dan (8) dapat dibuat menjadi sebuah *open-loop transfer function* dengan menghilangkan $I(s)$ antara kedua persamaan tersebut dimana kecepatan putar motor DC adalah keluaran dan tegangan jangkar adalah inputnya seperti ditunjukkan pada persamaan (9).

$$P(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K}{(Js + b)(Ls + R) + K^2} \quad (9)$$

Persamaan (2) dan persamaan (9) akan digunakan dalam simulasi sistem kendali PID dengan obyek *plant* adalah sebuah motor DC. Model dari motor DC pada simulasi menggunakan parameter sesuai dengan table 1 sehingga fungsi alihnya menjadi seperti yang ditunjukkan pada persamaan (10).

$$\frac{\dot{\Theta}(s)}{V(s)} = \frac{0.01}{0.005s^2 + 0.06s + 0.1001} \quad (10)$$

```
import time
start_time = time.time()
from control.matlab import *
import matplotlib.pyplot as plt
J = 0.001;
b = 0.1;
K = 0.01;
R = 1;
L = 0.5;
Kp = 2;
Ki = 1;
Kd = 1;
motor_model = tf([K],[J*L, (J*R)+(b*L), (b*R)+K**2])
controller = tf([Kd, Kp, Ki],[1,0])
system = feedback(controller*motor_model,1)
y,t = step(system,range(0, 200, 1))
plt.plot(t,y)
plt.title('Response of Motor DC with PID Control')
print("Execute Time: %s seconds" % (time.time() - start_time))
plt.show()
```

Gambar 5. Kode Python untuk Simulasi sistem kendali PID untuk pengendalian Motor DC

Persamaan (2) dan (10) tersebut kita bisa tuliskan kedalam sebuah kode MATLAB seperti yang ditunjukkan pada gambar 3. Pada GNU *Octave*, kode yang pada MATLAB juga dapat digunakan namun perlu memanggil paket / modul *control* terlebih dahulu dengan perintah *pkg load control*. Selain kode utama dari simulasi sistem kendali PID ada tambahan kode di awal dan akhir program untuk mencatat waktu eksekusi program sebagai pembanding antar masing-masin *software*.

```
tic
J = 0.001;
b = 0.1;
K = 0.01;
R = 1;
L = 0.5;
Kp = 2;
Ki = 1;
Kd = 1;
s = tf('s');
motor_model = K/((J*s+b)*(L*s+R)+K^2);
controller = Kp + Ki/s + Kd*s;
system = feedback(controller*motor_model,1);
[y,t] = step(system,[0:1:200]);
plot(t,y)
title('Response of Motor DC with PID Control')
ExecuteTime = toc
```

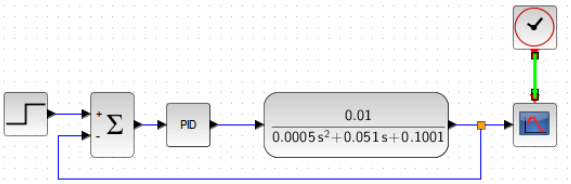
Gambar 3. Kode MATLAB dan GNU *Octave* untuk Simulasi sistem kendali PID untuk pengendalian Motor DC

Gambar 4 dan 5 menunjukkan kode program pada perangkat lunak *Scilab* dan *Spyder*. Ada sedikit perbedaan sintaks yang digunakan pada *Scilab* yaitu terutama pada pembentuk fungsi alih dan dalam membuat *feedback* dari sistem.

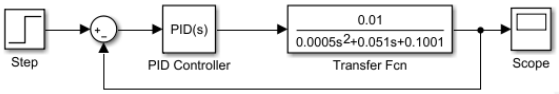
```
tic();
J = 0.001;
b = 0.1;
K = 0.01;
R = 1;
L = 0.5;
Kp = 2;
Ki = 1;
Kd = 1;
s = poly(0,'s');
motor_model = syslin('c', (K / (((J*s+b)*(L*s+R))+K^2)))
controller = syslin('c', (Kp + Ki/s + Kd*s))
system = (controller*motor_model) /. 1
t=0:1:200;
y=csim('step',t,system);
plot2d(t,y);
xtitle('Response of Motor DC with PID Control');
t=toc();
printf("Execute Time: %f \n",t);
```

Gambar 4. Kode *Scilab* untuk Simulasi sistem kendali PID untuk pengendalian Motor DC

Selain menggunakan kode program, simulasi sistem kendali PID dapat dilakukan dengan menggunakan penggambaran blok-blok diagram. Namun tidak semua perangkat lunak dapat melakukan hal ini, hanya MATLAB dengan simulink-nya dan *Scilab* dengan xcos-nya yang dapat melakukan simulasi dengan menggunakan blok diagram. Blok diagram pada simulink dan xcos ditunjukkan pada gambar 6 dan 7.



Gambar 6. Blok diagram simulasi kendali PID pada *Scilab* xcos



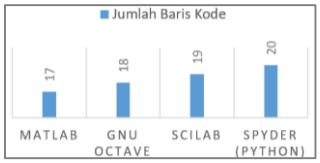
Gambar 7. Blok diagram simulasi kendali PID pada MATLAB simulink

HASIL DAN PEMBAHASAN

Komputer yang digunakan pada penelitian ini memiliki spesifikasi seperti yang tertera pada tabel 2. Hasil dari masing-masing perangkat lunak akan dibandingkan satu dengan lainnya. MATLAB merupakan program berbayar sedangkan 3 lainnya adalah program gratis dan *open source.*

Jumlah baris kode dari MATLAB adalah yang paling sedikit yaitu 17 sedangkan GNU *Octave* karena diharuskan memanggil modul *control*, jumlah barisnya adalah 18. Untuk *Scilab* jumlah kode baris adalah 19 dan untuk kode yang dituliskan pada *Spyder* adalah yang terbanyak yaitu 20 baris. Grafik jumlah baris kode pada masing-masing perangkat lunak ditunjukkan pada gambar 8.

Hasil plot dari progam ditunjukkan pada gambar 9. Gambar 9a adalah hasil plot untuk *software* MATLAB, gambar 9b adalah hasil plot untuk GNU *Octave*, gambar 9c adalah hasil plot untuk *Scilab* dan gambar 9d adalah hasil plot untuk *Spyder*.



Gambar 8. Jumlah baris kode pada masing-masing perangkat lunak (MATLAB, GNU *Octave*, *Scilab* dan *Spyder* (Python))

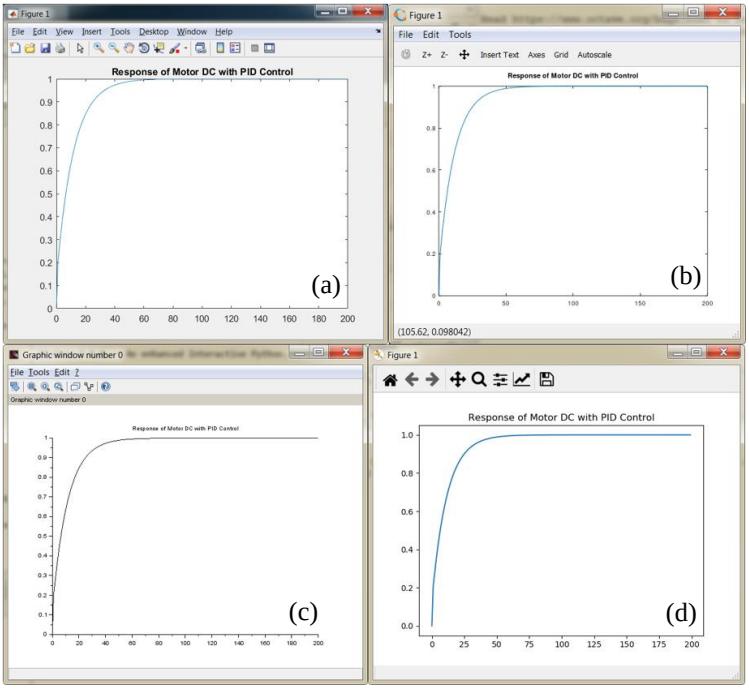
Tabel 2. Spesifikasi Komputer

No	Komponen	Spesifikasi
1.	CPU	AMD FX-7500
2.	RAM	16384MB
3.	DISK	SSD 240GB

Sebelum dieksekusi untuk *Spyder* perlu memanggil beberapa *library control* dan *matplotlib.* Dengan memanggil *from control.matlab import* *, perintah baris kode yang ada sama dengan MATLAB karena pengembang dari *library* tersebut membuat perintah-perintahnya sama dengan MATLAB. Sehingga untuk membuat fungsi alih sama dengan MATLAB menggunakan perintah *tf*. *Library matplotlib* digunakan untuk membuat plot dari hasil perhitungan yang dijalankan saat simulasi.

Secara standar hasil plot dari masing-masing *software* terlihat sangat identik dengan beberapa hal yang sedikit terlihat berbeda. Terlihat hasil yang ditampilkan pada *Spyder* jarak skala yang tertampil sedikit melebihi nilai minimal dan maksimal dari hasil respon

sedangkan untuk ketiga lainnya jarak adalah sesuai dengan nilai minimal dan maksimal dari hasil respon. Warna garis yang tertampil pada hasil plot untuk *Scilab* tidak berwarna biru berbeda dibandingkan ketiga lainnya. Hasil plot juga terdapat kotak dibagian luar grafik untuk MATLAB, GNU *Octave* dan *Spyder* sedangkan tidak ada pada *Scilab*.

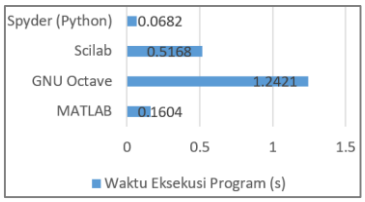


Gambar 9. Gambar plot dari respon sistem dengan set poin sinyal step pada software

(a) MATLAB (b) GNU *Octave* (c) *Scilab* (d) *Spyder*

Selain dari hasil plot yang diamati adalah lama waktu eksekusi program pada masing-masing perangkat lunak. Pada MATLAB, GNU *Octave* dan *Scilab* ada perintah *tic* dan *toc* untuk mencatat waktu eksekusi yan ditambahkan diawal dan akhir program. Untuk *Spyder* yang menggunakan bahasa pemograman Python adalah dengan menggunakan *library time* yang kemudian mencatat waktu awal dan akhir dari program dieksekusi dan waktu eksekusi dihitung dari pengurangan waktu akhir dan awal. Untuk proses penghitungan eksekusi program pada *Spyder* dapat dilihat pada baris 1-2 dan 19. Program dijalankan sebanyak 10 kali untuk masing-masing perangkat lunak dan kemudian diambil nilai rata-rata dari 10 kali tersebut. Hasil dari waktu eksekusi pada masing-masing perangkat lunak dapat dilihat pada grafik yang ditunjukkan pada gambar 10.

Dari grafik pada gambar 10 terlihat *Spyder* merupakan perangkat lunak yang tercepat dalam eksekusi program yaitu 0.0682 detik disusul setelahnya yaitu MATLAB dengan waktu eksekusi 0.1604 detik dan kemudian 2 terakhir adalah *Scilab* dan GNU *Octave* dengan waktu eksekusi 0.5168 detik dan 1.2421 detik.



Gambar 10. Rerata waktu eksekusi program simulasi kendali PID

Perbandingan terakhir adalah terkait dengan simulasi dengan menggunakan penggambaran diagram blok. *Scilab* dan *Spyder* merupakan perangkat lunak yang tidak bisa melakukan hal tersebut sehingga hanya MATLAB dengan Simulink dan *Scilab* dengan xcos yang dapat melakukan. Penggambaran diagram blok pada penelitian ini juga sederhana, pada Simulink terdapat 5 komponen sedangkan

xcos terdapat 6 komponen. Yang membedakan antara Simulink dan xcos adalah pada bagian *scope*, pada xcos membutuhkan pewaktu eksternal yang berupa timer sedangkan pada Simulink tidak membutuhkan.

SIMPULAN

Terdapat beberapa aspek yang diamati dalam perbandingan penggunaan perangkat lunak MATLAB, GNU *Octave*, *Scilab* dan *Spyder* dalam menjalankan simulasi sistem kendali PID. Pada aspek sisi biaya langganan, MATLAB adalah yang paling besar sedangkan ketiga perangkat lunak lainnya dapat digunakan secara gratis. Berkaitan dengan *block programming*, MATLAB menjadi yang terbaik karena dukungan modul yang sangat memudahkan pengguna karena lengkapnya dokumentasi dan contoh blok yang tersedia. Selain MATLAB, *Scilab* memiliki xcos yang bentuknya sangat mirip sekali dengan Simulink yang ada pada MATLAB. Penggunaan xcos untuk simulasi sistem kendali PID sangat mirip dengan Simulink yang ada pada MATLAB sehingga tidak ada perbedaan yang signifikan. Dalam pengamatan kecepatan eksekusi program, *Spyder* adalah yang tercepat dengan waktu proses 0.0682 detik dan yang paling lama waktu proses adalah GNU *Octave* dengan waktu 1.2421 detik. Dengan mengamati beberapa aspek pada ke-4 software tersebut kita dapat berkesimpulan bahwa penulis merekomendasikan MATLAB apabila pembiayaan bukan suatu masalah sedangkan GNU *Octave* merupakan software open source yang dapat dipilih selanjutnya karena dukungan komunitas yang sangat baik.

REFERENSI

[1] R. Pratiwi, A. Waris, and S. Salengke, "Rancang Bangun Sistem Kendali Kecepatan Putaran Motor Dc Berbasis Logika Fuzzy Untuk Mesin Pengaduk Hasil Pertanian (Studi Kasus Pengadukan Biji Kedelai)," *J. Agritechno*, 2019, doi: 10.20956/at.v12i1.185.

[2] J. Zhao, L. Han, L. Wang, and Z. Yu, "The fuzzy PID control optimized by genetic algorithm for trajectory tracking of robot arm," in *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, 2016, doi: 10.1109/WCICA.2016.7578443.

[3] J. SUN and L. L. CHENG, "Robust PID Controller for AR Drone," 2017, doi: 10.1142/9789813146426_0138.

[4] M. L. Rajaram, E. Kougianos, S. P. Mohanty, and U. Choppali, "Wireless Sensor Network Simulation Frameworks: A Tutorial Review: MATLAB/Simulink bests the rest," *IEEE Consum. Electron. Mag.*, 2016, doi: 10.1109/MCE.2016.2519051.

[5] R. Brum, M. C. S. De Castro, and F. A. B. Silva, "A whole-cell simulator in GNU *Octave*," in *AIP Conference Proceedings*, 2017, doi: 10.1063/1.5012406.

[6] Priyadarshni and J. S. Sohal, "Improvement of artificial neural network based character recognition system, using *Scilab*," *Optik (Stuttg).*, 2016, doi: 10.1016/j.ijleo.2016.05.106.

[7] Z. Hussain and M. Siyab Khan, "Introducing Python Programming for Engineering Scholars," 2018.

[8] A. P. Leros, A. Andreatos, and A. Zagorianos, "Matlab - *Octave* science and engineering benchmarking and comparison," in *International Conference on Computers - Proceedings*, 2010.

[9] K. Ogata and J. W. Brewer, "Modern Control Engineering," *J. Dyn. Syst. Meas. Control*, 1971, doi: 10.1115/1.3426465.

[10] R. De Keyser, C. M. Ionescu, and C. I. Muresan, "Comparative evaluation of a novel principle for PID autotuning," in *2017 Asian Control Conference, ASCC 2017*, 2018, doi: 10.1109/ASCC.2017.8287335.

[11] G. Tan, W. Shu, Y. Guo, and M. Liu, "The application of fuzzy control in brushless DC motor for pure electric vehicle," in *2015 IEEE International Conference on Communication Problem-Solving, ICCP 2015*, 2016, doi: 10.1109/ICCPS.2015.7454229.