BAB III

METODE PENELITIAN

3.1 Perancangan

Pada *software Artificial Intelligent* ini menggunakan dua jenis *controller* yaitu *controller* PID dan ANFIS. Kedua jenis *controller* ini penggunaannya tidak secara bersamaan. *User* harus memilih salah satu *controller* yang dikehendaki.

Jika *user* memilih *controller* PID maka *user* akan diberi kesempatan mengisi kotak *setpoint*, *Konst.Prop* untuk *controller* P, *Konst.Int* untuk *controller* I, *Konst.Der* untuk *controller* D. Berbeda dengan pemilihan ANFIS *user* hanya mengisi kotak *setpoint* saja.

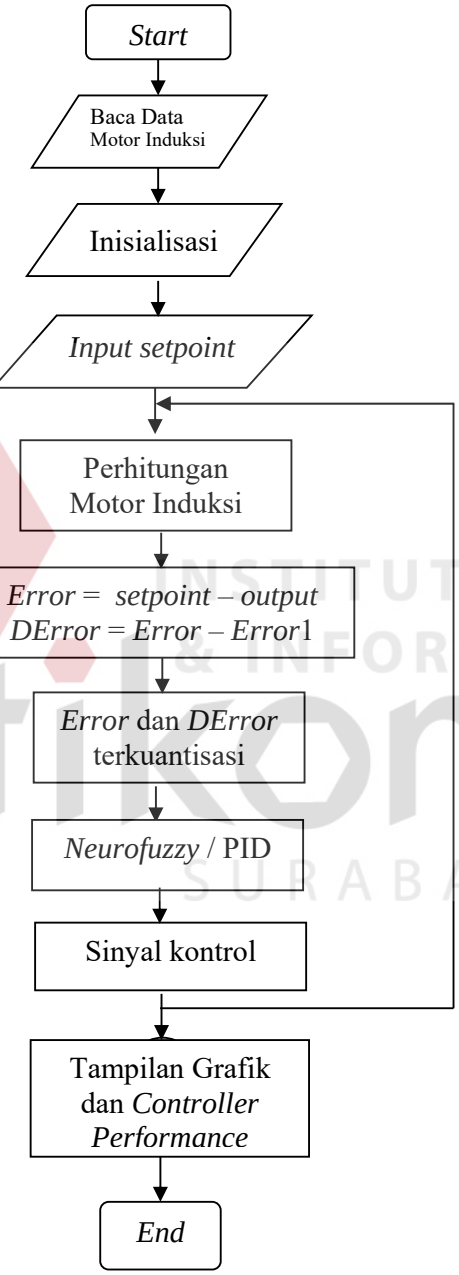
Hasil respon sistem akan ditampilkan bagian *output*, berupa data *overshoot, Settling Time, Rise Time, Error, Error S S, Kp, Ki, Kd*. Data ini akan tampil untuk kedua jenis *controller.*

Tampilan yang akan diberikan berupa informasi dengan grafik yang terdiri dari 3 jenis. Grafik untuk tampilan *Error & Delta Error,* Respon Sistem dan Konstanta PID. Untuk kedua *controller* akan sama – sama memberikan informasi tersebut.

Dan untuk tampilan grafik tersebut dapat disimpan dalam bentuk ektensi BMP *Image*. Penentuan grafik yang akan disimpan dapat disesuaikan dengan keinginan *user.*

3.1.1 Perancangan Aplikasi Perangkat Lunak *Artificial Intelligent Controller*

A. Diagram Alir



Gambar 3.1. Diagram Alir proses simulasi

Uraian dari diagram alir diatas yaitu sebagai berikut

- Proses diawali dengan pemberian *setpoint*
- Pembacaan data karakteristik dari motor induksi.
- Proses Inisialisasi .
- Perhitungan motor induksi secara matematis yang hasilnya merupakan *output* dari motor induksi tersebut.
- Setelah didapatkan *error* antara *setpoint* dengan *output* dan juga didapatkan *delta error*, maka selanjutnya menghitung besar sinyal kontrol dengan menggunakan *Neurofuzzy* PID.
- *Error* dan *delta error* terkuantisasi menjadi *input controller Neurofuzzy* PID untuk diproses sehingga menghasilkan nilai K_p, K_i, K_d dan setelah di-*input* kedalam persamaan berikut :

$$m(t) = Kp\, e(t) + K_i T_s \sum_{i=1}^{n} e(t) + \frac{K_d}{T_s} de(t) \qquad (3.1)$$

maka akan menghasilkan sinyal *control* yang akan di-*input*kan ke dalam *plant*.

- Dengan perubahan kecepatan motor akan menimbulkan *error* dan *delta error* yang baru yang akan menjadi *input controller Neurofuzzy* PID. Dari perubahan *error* dan *delta error* tersebut menyebabkan *update* parameter dari *controller Neurofuzzy* PID.

3.2 Pembuatan *Software* Simulator *Artificial Intelligent*

3.2.1 Algoritma ANFIS

Pertama kali *neurofuzzy* PID menerima *input* dari *error* dan *delta error* sebagai *input* untuk diproses sehingga didapatkan nilai K_p, K_i, K_d .

- Lapisan *input*

  Lapisan *input* digunakan untuk membentuk fungsi kenggotaan *error* dan *delta error* sebagai akibat dari *input error* dan *delta error* serta meyebarkan informasi pada lapisan berikutnya untuk diproses. Penggalan *listing membership* untuk *error*, programnya adalah sebagai berikut :

```
// Perhitungan KP
// layer 1 : membership Function Error
      if x[1] >= 1.5 then
      Begin
          mf1[7]:=1;
          for s:=1 to 6 do
          mf1[s]:=0;
          End;

          if ((x[1] >= -1.5) and (x[1] < 1.5)) then
          Begin
          for s:=1 to 7 do
          mf1[s]:=exp(-(sqr((x[1]-a11[s])/b11[s])));
          End;

          if x[1] < -1.5 then
          Begin
          mf1[1]:=1;
          for s:=2 to 7 do
          mf1[s]:=0;
      End;
```

  Untuk *membership delta error*, prinsipnya sama dengan penggalan *listing* diatas yang nantinya akan menghasilkan suatu variable mf2[s].

- Lapisan dalam

  Lapisan ini berfungsi mengolah informasi yang diberikan oleh lapis sebelumnya, serta menyebar informasi pada lapis berikutnya untuk diproses kembali.

```
     for r:=1 to 7 do
Begin
 for s:=1 to 7 do
  cz1[r,s]:=c1[r,s]*z1[r,s];
End;
```

- Lapisan *output*

  Lapisan ini berfungsi untuk mengeluarkan hasil yang telah diproses pada lapisan dalam sehingga menghasilkan nilai K_p, K_i, K_d .

```
     g1_bukanArray:=0;
for r:=1 to 7 do
begin
 for s:=1 to 7 do
  begin
   g1[r,s]:=g1_bukanArray + cz1[r,s];
   g1_bukanArray:=g1[r,s];
  end;
 end;
x1:=g1_BukanArray/h1_BukanArray;

for r:=1 to 7 do
  Begin
   for s:=1 to 7 do
    cz1[r,s]:=c1[r,s]*z1[r,s];
  End;

dst …..
```

Penggalan program diatas hanya mencakup pencarian nilai K_p saja dan pada prinsipnya untuk mencari nilai K_i dan K_d sama seperti dalam pencarian nilai K_p.

Normalisasi nilai K_p, K_i dan K_d

```
// normaslisasi dari rumus PID
 kpn:=((3-0.5)*kp)+0.5;
 kin:=(((1e-6)-(1e-7))*ki)+1e-7;
 kdn:=(((8e-3)-(1e-6))*kd)+1e-6;

 sig_er:=sig_er1+er;
```

realisasi dari rumus PID yang akan menjadi *input* untuk *plant*

```
// sinyal yang dikirmkan ke plant
 ter:=ter_1 + kpn * er  + kin * tsc *sig_er + (kdn * der) / tsc;

 sig_er1:=sig_er;
 End;
```

3.2.2 Algoritma Proportional Integral Derivative Controller

Pada *controller* ini saat *user* mengisikan data konstanta *controller* maka hasil ketiga *controller* pada kotak *output* akan sama dengan yang diisikan.

Namun untuk hasil *performance output* akan sama prosesnya dengan ANFIS.

3.2.3 Performansi *Output*

Pada tampilan data *output* dapat kita lihat informasi yang diberikan yaitu berupa *overshoot, settling time, rise time, error, error S S, Kp, Ki* dan *Kd.* Adapun hasil tersebut didapatkan adalah merupakan hasil pengolahan dari *input* yang diberikan oleh *user*.

Untuk *overshoot* atau lewatan maksimum (persen) yang merupakan harga puncak maksimum dari kurva respon tidak sama dengan satu, maka biasa digunakan persen lewatan maksimum.

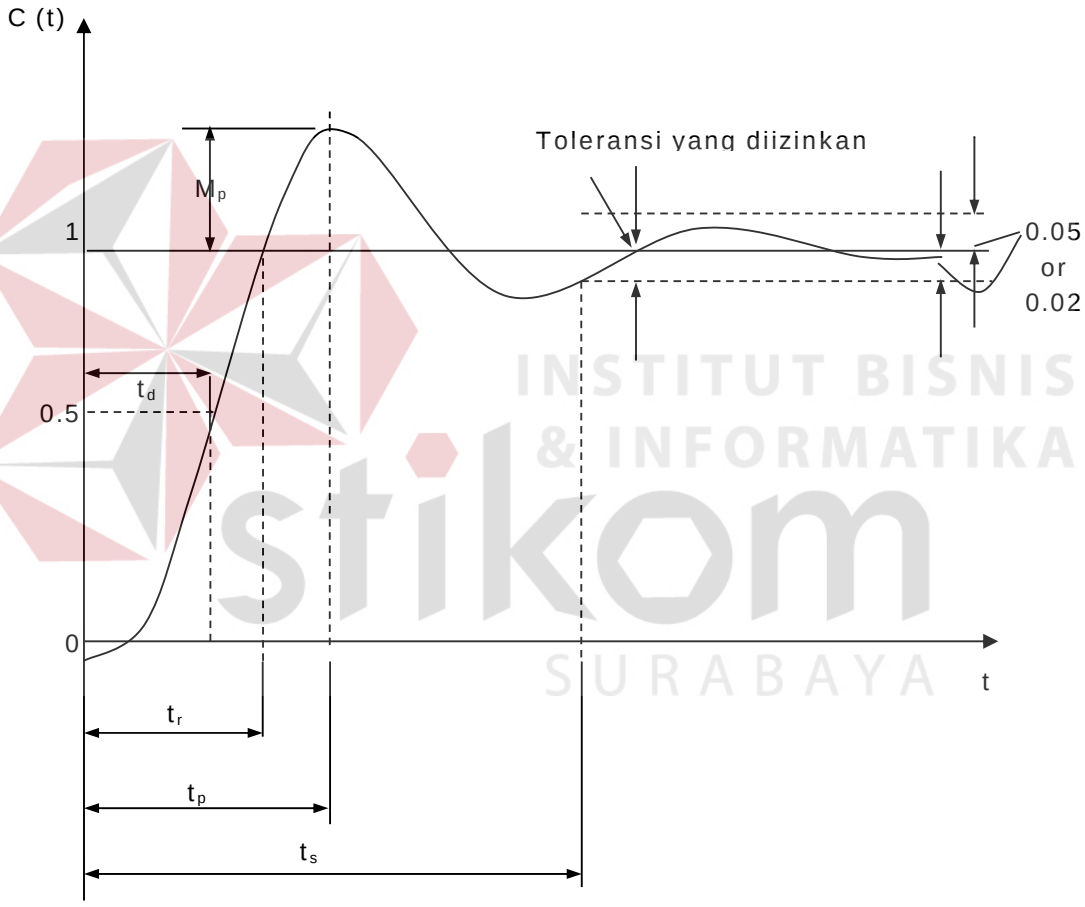
$$\text{Persen lewatan maksimum} = \frac{c_{(waktu\ puncak)} - c(\infty)}{c(\infty)} \times 100\,\% \qquad (3.2)$$

Besarnya *overshoot* atau persen lewatan maksimum secara langsung menunjukkan relatif sistem.

Waktu yang diperlukan kurva respon untuk mencapai dan menetap dalam daerah sekitar harga akhir dan ukurannya ditentukan dengan presentase mutlak harga akhirnya (biasanya 2% atau 5%) yang merupakan suatu *settling time* respon sistem. Waktu penetapan ini dikaitkan dengan konstanta waktu terbesar dari

sistem kontrol. Kriteria presentase kesalahan yang digunakan ditentukan dari sasaran disain sistem yang ditanyakan.

Pada *rise time* yang merupakan waktu yang diperlukan oleh respon untuk naik dari 10% menjadi 90%, 5% menjadi 95% atau 0% menjadi 100% dari nilai akhir yang biasa digunakan. Untuk sistem diatas *rise time* yang biasa digunakan 10% menjadi 90%



Gambar 3.2. Kurva respon sistem yang menunjukan *overshoot (Mp), Settling time(t_s), rise time(t_r)*

Seperti yang telah dijelaskan diatas pada tampilan tabel *output* akan memberikan beberapa jenis data untuk melengkapi informasi yang yang dibutuhkan oleh *user*.

Dan penggalan program yang ada pada simulator pada tugas akhir ini adalah sebagai berikut :

```
// arus torsi
 It:=(lr/(p*lm*lm*Im))*ter;
 // kecepatan slip
 wsl:=(rr/(lr*Im))*It;
 // tranformasi arus stator ke sumbu abc
 ir_dq2abc(IM,It,y_kecil[7]);
 is_dq2abc(y_kecil[1], y_kecil[2], y_kecil[5]);
 // switching PWM Inverter
  PWM_Inv;
 IF PIDNeuroFuzzy1.Checked = True then Update_NFPID;
   //hasil rise time
  rt:=rt2-rt1;
 // mencari omega_r ( respon sistem) terbesar
  if looping = 1 then
   omega_r_terbesar := omega_r
  else
  begin
   if omega_r_terbesar < omega_r Then  omega_r_terbesar := omega_r
  end;
 // output
  // Perhitungan overshoot
 mov := ((omega_r_terbesar-setpoint)/setpoint)*100;
 // pemilihan overshoot
 if mov > 0 then
  lblOvershoot.Caption := ': ' + floattostr(mov) +  ' %'
 else
  lblOvershoot.Caption := ': ' + 'Tidak Terjadi';
 // harga mutlak error yang terjadi
 Error := abs(er);
```

Untuk menampilakan data yang ada pada kolom *output* digunakan cara sederhana. Dengan menuliskan langsung pada kolom yang sudah dibuat. Perhatikan tampilan program berikut:

```
// output performa sistem
lblSettlingtime.Caption  := ': ' + floattoStr(st) + ' dt';
lblriseTime.Caption      := ': ' + floattostr(rt) + ' dt';
```

```
 lblError.Caption         := ': ' + floattostr(Error)+ ' rpm';
 lblErrorSS.Caption       := ': ' + floattostr(pe) + ' rpm';
 lblKp.Caption            := ': ' + floattostr(Kpn);
 lblKi.Caption            := ': ' + Floattostr(Kin); // * 1000000 );
 lblKd.Caption            := ': ' + floattostr(Kdn); // * 100 );
 lblTime.Caption            := ': ' + floattostr(Waktu) + ' dt';
 // waktu dalam millisecond
 waktu := Looping * Ts;
 // banyak iterasi
 inc(looping);
end;
```