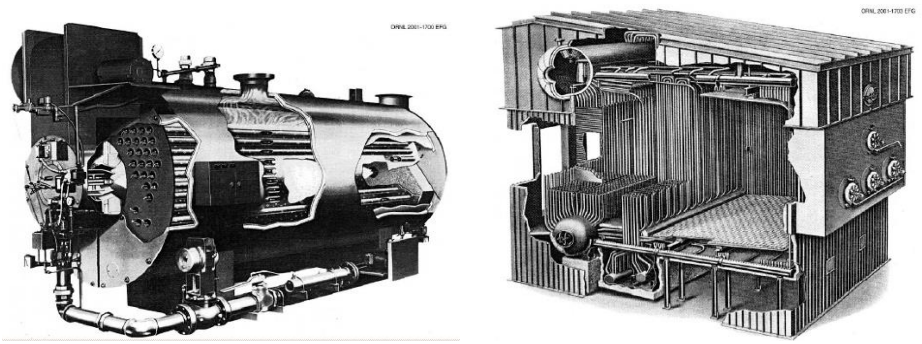
BAB II

TINJAUAN PUSTAKA

2.1 *Boiler*

Boiler adalah suatu alat yang biasanya berupa tanki/drum/vessel tertutup yang terbuat dari baja dan berfungsi sebagai penghantar panas yang dihasilkan dari proses pembakaran bahan bakar terhadap air sehingga menghasilkan *steam* dengan tekanan dan temperatur tertentu.[4:2)]*Steam* yang dihasilkan dapat digunakan sebagai media pemanas (tekanan rendah), penggerak *steam turbin* pada pompa (tekanan sedang), dan tenaga penggerak *steam turbine* pada pembangkit tenaga listrik (tekanan tinggi).[2:1)]



(a) **(b)**

Gambar 2.1 a. *Fire Tube Boiler* b.*Water Tube Boiler*

Klasifikasi *boiler* berdasarkan bentuk konstruksinya dibedakan menjadi dua, yaitu *fire tube boiler* (*boiler* pipa api)[3:5)] dan *water tube boiler* (*boiler* pipa air[3:6)]. *Fire tube boiler* adalah *boiler* yang produksi *steam* berada diluar *tube*, desain ini memiliki kapasitas kecil, tekanan rendah, *steam saturated*, dans*team* di *shell side*. Sedangkan *water tube boiler* adalah *boiler* yang produksi *steam* berada

di dalam *tube*, desain ini memiliki kapasitas besar, tekanan operasi tinggi, dan *steam saturated / superheated*.[4:4-6)]*Fire tube boiler* dan *water tube boiler* dapat dilihat pada gambar 2.1.

Secara umum, tujuan sistem kontrol pada *boiler* adalah menghasilkan produk steam yang sesuai dengan spesifikasi dan menjaga *boiler*agar beroperasi dengan efisien dan aman. Secara garis besar, sistem kontrol pada *boiler* ini terdiri dari [11)] :

1) *Drum level control*;
2) *Combustion control*;
3) *Atomizing control*;
4) *Blowdown control*; dan
5) *Steam temperature control*.

2.2 *Drum Level Control*

Tujuan *drum level contro1*[13)] adalah menjaga agar *level drum* tetap pada *setpoint*-nya walaupun terjadi perubahan beban ataupun gangguan (*disturbance*) lainnya.*Level drum* yang terlalu rendah bisa menyebabkan terjadinya panas berlebih (*overheated*) pada *boiler tubes* sehingga *tubes* bisa menjadi rusak/bengkok/bocor. Sebaliknya *level drum* yang terlalu tinggi akan menyebabkan pemisahan air dan *steam* dalam drum tidak sempurna sehingga kualitas *steam* yang dihasilkan kurang (banyak mengandung air/basah).

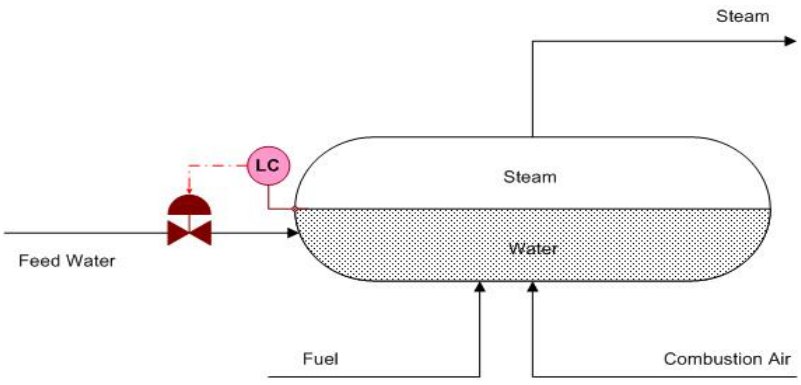
Ada tiga alternatif/jenis *drum level control*, yaitu:

1) *Single element drum level control*;

2) *Two-element drum level control*;

3) *Three-element drum level control*.

Single-element drum level control. Ini merupakan konfigurasi *drum level control* yang paling sederhana, yaitu hanya menggunakan *feedback level control*. Disebut *single-element* karena hanya *level drum* saja yang dikontrol. Konfigurasi kontrol ini umumnya digunakan pada *boiler* berkapasitas rendah (<150,000 pounds-per-hour), *pressure* rendah (<250 pounds-per-square-inch), dan dengan beban yang relatif tetap/stabil. Kekurangan konfigurasi kontrol ini adalah sulit mempertahankan level pada *setpoint*nya jika terjadi perubahan beban secara terus menerus.



Gambar 2.2*Single Element Drum Level Control*

2.3 Dasar Pengendalian Proses

Sistem pengendalian proses merupakan gabungan dari beberapa alat dengan fungsi tertentu yang digunakan untuk mempertahankan variabel yang dikendalikan (*process variable*)pada nilai tertentu (*set point*). Pada awalnya, suatu pengendalian proses dilakukan secara manual. Namun, seiring dengan

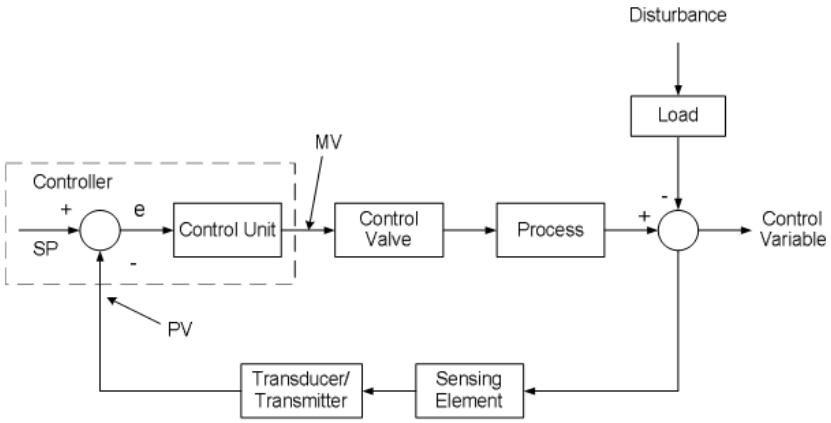
perkembangan teknologi pengendalian manual mulai ditinggalkan dan beralihmenggunakan sistem pengendalian otomatis yang menggunakan suatu unit kontroler dengan suatu algoritma tertentu untuk menghitung besarnya nilai *manipulated variable* berdasarkan *error*.

Tujuan utama dari sistem pengendalian adalah menjaga atau mengendalikan *process variable* agar selalu sama dengan *set point* walaupun kondisi ideal tidak pernah tercapai sepenuhnya karena adanya *load* ataupun *disturbance*. Sehingga sistem harus diatur sedemikian rupa agar respon dari kontroler dapat stabil (tidak berosilasi pada semua kondisi operasi), cepat (tidak terjadi *delay* dalam menanggapi perubahan) , dan tepat (mendekati nilai *set point*).

Dalam sistem pengendalian terdapat beberapa istilah yang perlu diketahui :

1. *Set Point* adalah besarnya nilai *prosesvariable* yang dikehendaki. Sebuah *controller* akan selalu berusaha untuk menyamakan *proses variable* dengan *set point.*
2. *ProsesVariable*(*Measurement Variable*) adalah besaran yang nilainya dikendalikan.
3. *Manipulated Variable* adalah input dari suatu proses yang dapat dimanipulasi atau diubah-ubah besarnya agar *proses variable* besarnya sama dengan *set point.*
4. *Disturbance* adalah besaran lain selain manipulated variable yang dapat menyebabkan berubahnya *proses variable.*

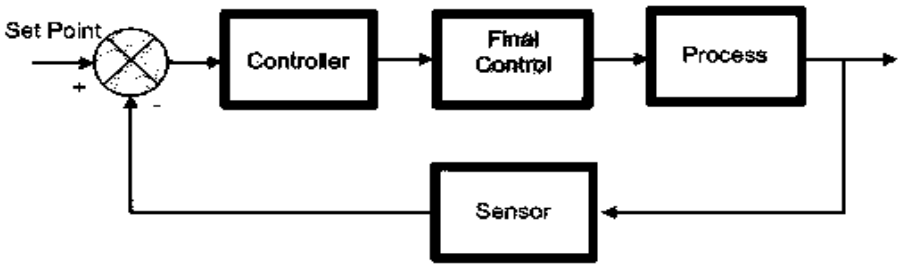
5. *Error* adalah selisih antara *set point* dikurangi *measurement variable*(*proses variable*). *Error* bisa bernilai positif dan bisa juga negatif. Bila *measurement variable* lebih besar dari *set point* maka *error* menjadi positif dan sebaliknya.
6. *Sensing Element*, adalah bagian paling ujung suatu sistem pengukuran. Contoh *sensing elemen*t yang banyak dipakai adalah *Thermocouple*. Bagian ini juga biasa disebut *Sensor* atau *Primary Element.*
7. *Transmitter,* adalah alat yang berfungsi untuk membaca sinyal *sensing element,* dan mengubahnya menjadi sinyal yang dapat dimengerti oleh *controller.*
8. *Controller,* adalah element yang mengerjakan tiga dari empat tahap langkah pengendalian yaitu membandingkan *set point* dengan *measurement variable*, menghitung berapa banyak koreksi yang dilakukan, dan mengeluarkan sinyal koreksi sesuai dengan hasil perhitungan tadi. *Controller* sepenuhnya menggantikan peran manusia dalam mengendalikan suatu proses.
9. *Final Control element*, adalah bagian akhir dari instrument sistem pengendalian yang berfungsi untuk mengubah *measurement variable*dengan cara memanipulasi besarnya *manipulated variable* berdasarkan perintah *controller.*



Gambar 2.3 Blok Diagram Sistem Kontrol Proses

Dalam pelaksanaannya[1:129)], sistem pengendalian memiliki banyak klasifikasi *control loop*.Klasifikasi *control loop* meliputi *single control loop* (*feedback control* dan *feedforward control*) dan *complicated control loop* (*cascade, split range, ratio,* dan*auto selector control*). Pada pembahasan kali ini penulis akan mengulas *feedback control*, *cascade*,dan *feedforward control*.

2.2.1 Rangkaian Kontrol Umpan Balik (*FeedBack*)

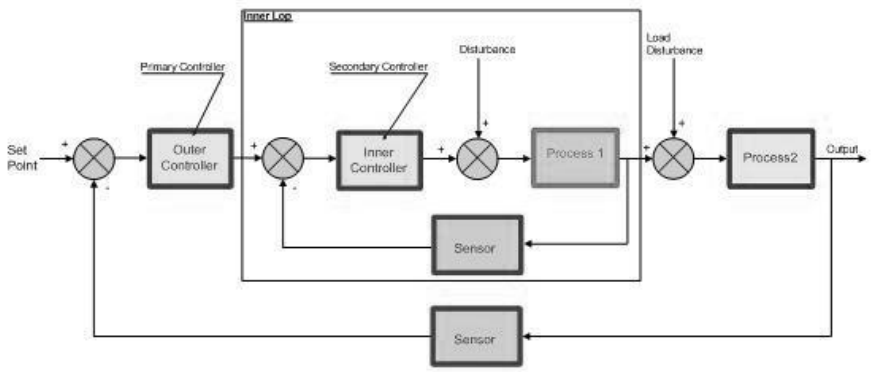


Gambar 2.4 Diagram blok *Feedback Control* dalam industri proses

Feedback control[1:140)] adalah suatu sistem pengontrolan dimana *control action* tergantung pada *output* proses. Tipe sistem kontrol ini mengukur *process variable* pada *output* proses. Setiap terjadi perubahan pengukuran pada *outlet* proses akibat adanya efek *disturbances* (*load*), maka sistem kontrol *feedback* bereaksi memberikan *corrective action* untuk menghilangkan kesalahan (*error*).

Jadi sistem *control feedback*akan bereaksi setelah efek dari *disturbances* dirasakan pada *output* proses.

2.2.2 Rangkaian Kontrol Kombinasi (*Cascade*)



Gambar 2.5 Diagram blok *Cascade Control* dalam industri proses

Kontrol *cascade* adalah sebuah metode kontrol yang memiliki (minimal) dua buah *loop*pengontrolan :*loop* pengontrolan primer atau *master loop* dan *loop* pengontrolan sekunder atau *slave loop*. Dalam skema kontrol ini, output kontroler pada sisi master secara fungsional digunakan untuk memanipulasi *set point* bagi *loop* pengontrolan sekundernya. [10:186)]

Hasil dari penggabungan kontrol ini dapat menghasilkan keluaran yang mempunyaai respons perbaikan terhadap kesalahan dengan lebih cepat.Dan kelebihan lainnya, karena kontrol ini terdiri atas dua unit *sensing element*, menjadikannya lebih sensitif daripada kontrol tunggal.[10:187)]

Penerapan pengendalian *cascade* dapat merugikan apabila elemen proses di *primary loop* lebih cepat dari elemen proses pada *secondary loop*, karena sistem akan cederung berosilasi akibat timbulnya interaksi antara *primary loop* dan *secondary loop*. Jadi sistem pengendalian *cascade* hanya dapat diterapkan pada proses dengan elemen primer yang jauh lebih lambat dari elemen *secondary*-nya.[1:142)]

2.4 *Transfer Function* komponen kontrol [7)]

2.4.1 *Control Valve*

Model matematika *control valve* diperoleh dengan persamaan :

$$\frac{M_b(s)}{U(s)} = \frac{Kv}{\tau_v s + 1} \quad \text{(2.1)}$$

$$Kv = \frac{Span\ Output\ (Flow)}{Span\ Input\ (Arus)} \times \frac{Span\ Output\ I/P(Pneumatik)}{Span\ Input\ I/P(Arus)} \quad \text{(2.2)}$$

Dimana :

M_b (s)= Laju aliran BFW (kg/s)

U (s)= sinyal masukan ke control valve (Amp)

K_v = Gain *control valve*

τ_v = *time constan control valve* (s)

2.4.2 *Controller*

Untuk *gain* dari *controller*

$$mv_{(s)} = Kp\left[1 + \frac{1}{T_i s} + T_d s\right] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.3)$$

Dimana:

Kp= Konstanta Proportional

Ti= *Integral time*

Td= *Derivative time*

2.4.3 Pemodelan matematika pada kontrol level *Steam Drum*

Hukum kesetimbangan massa menyatakan bahwa jumlah massa yang masuk ke dalam sistem sebanding dengan jumlah massa yang keluar dari sistem dan massa yang terakumulasi dalam sistem itu sendiri. Hal ini juga berlaku untuk *"boiler steam drum"* yang mempunyai masukan berupa air dari *feedwater system* dan keluarannya berupa steam yang nantinya digunakan pada *Superheater System.*

Hukum kesetimbangan massa pada *"boiler-steam drum"* sesuai dengan persamaan berikut :

$$\begin{bmatrix} laju\ massa \\ di\ dalam \\ steamdrum \end{bmatrix} = \begin{bmatrix} laju\ massa \\ feedwater \end{bmatrix} - \begin{bmatrix} laju\ massa \\ steam \end{bmatrix} - \begin{bmatrix} laju\ massa \\ blowdown \end{bmatrix} \dots\dots\dots\dots (2.4)$$

Laju perpindahan massa pada *blowdown system* dapat diabaikan karena pada *blowdown system* fluida cair akan dialirkan menuju *furnace* di dalam boiler kemudian dipompakan kembali ke dalam *boiler steam drum* dan proses ini terjadi secara kontinyu, sehingga persamaan di atas menjadi :

$$\begin{bmatrix} laju\ massa \\ di\ dalam \\ steamdrum \end{bmatrix} = \begin{bmatrix} laju\ massa \\ feedwater \end{bmatrix} - \begin{bmatrix} laju\ massa \\ steam \end{bmatrix}$$

$$\rho_w \frac{dV_L}{dt} + \rho_v \frac{dV_v}{dt} = \dot{m}_w - \dot{m}_v \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (2.5)$$

Dimana :

m_v = Massa *Vapor* (kg)

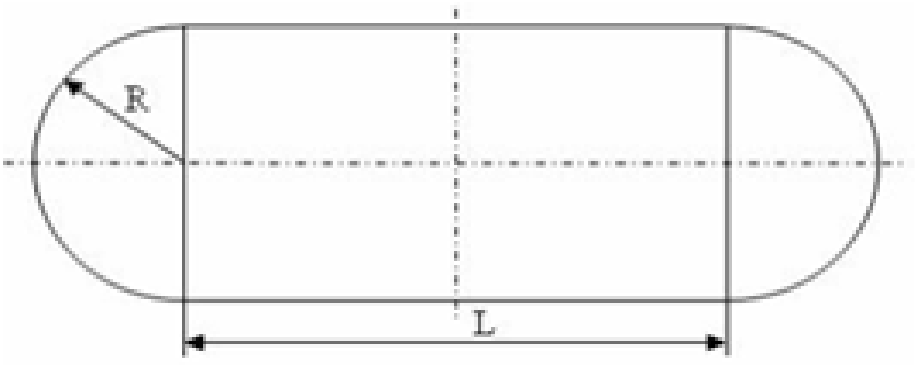
m_w = Massa *liquid* (kg)

ρ_v = Massa Jenis *Vapo*r (kg/m^3)

ρ_w= Massa Jenis air (kg/m^3)

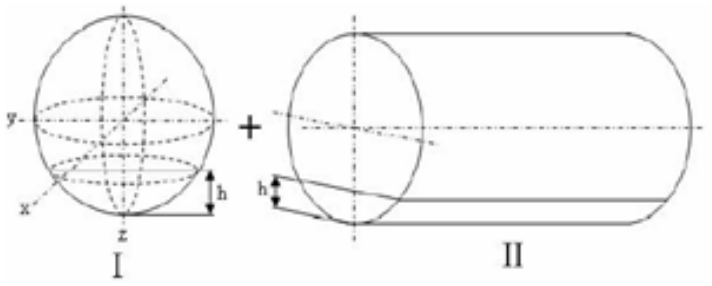
V_v = Volume *Vapor* (m^3)

V_l= Volume *Liquid* (m^3)

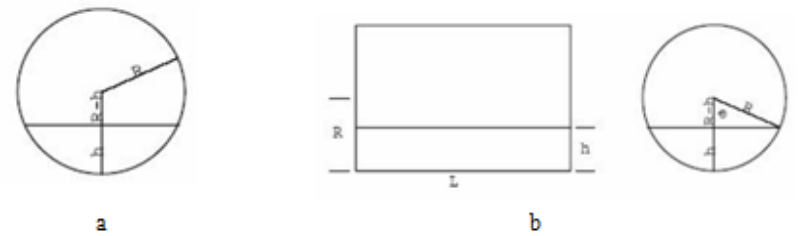


Gambar 2.6 Penampang melintang '*boiler steam-drum*'

Untuk memodelkan *boiler steam drum* dalam hal ini dilakukan dengan menganggapnya sebagai gabungan antara sebuah bola dan sebuah tabung yang memiliki diameter yang sama yang di dalamnya dianggap hanya berupa ruang kosong. Sehingga pendekatan matematisnya dapat dituliskan :



Gambar 2.7 Pendekatan *boiler steam drum*



Gambar 2.11 (*a*) penampang bola, (*b*) penampang tabung

Dengan melakukan perhitungan matematis maka:

$$V_{bola} = \left(\frac{\pi}{3}\right) h^2 (3R - h) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots . (2.6)$$

$$V_{tabung} = \frac{1}{2} R^2 L \left[2acrCos \frac{R-h}{R} - Sin\left(2arcCos \frac{R-h}{R} \right) \right] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots . (2.7)$$

Sehingga :

$$V_{Total} = V_{bola} + V_{tabung}$$

$$= \left(\frac{\pi}{3}\right) h^2 (3R - h) + \frac{1}{2} R^2 L \left[2acrCos \frac{R-h}{R} - Sin\left(2arcCos \frac{R-h}{R} \right) \right] \ldots .(2.8)$$

Dimana:

h = level fluida cair (m)

R = Radius *”Boiler Steam Drum”(m)*

L = Tinggi *”Boiler Steam Drum”* tanpa bagian bola (m)

2.5 Kontrol PID

Kontroler automatik membandingkan harga yang sebenarnya dari keluaran “proses” dengan harga yang diinginkan, menentukan deviasi, dan menghasilkan

suatu sinyal kontrol untuk memperkecil deviasi sampai nol atau suatu harga yang kecil, dengan cepat, tepat, dan stabil.[9:39)]

Cara kontroler automatik menghasilkan keluaran sinyal kontrol disebut mode pengontrolan (*control mode*). Kontroler automatik di industri dapat diklasifikasikan sesuai dengan mode pengontrolannya sebagai berikut[9:39-40)] :

1. Mode kontrol*on off*
2. Mode kontrol *propotional*
3. Mode kontrol *proportionalintegral*
4. Mode kontrol *proportionalintegral derivative*

Pada proses kontrol terdapat dua jenis proses yaitu proses*servo*, sering mengalami perubahan titik proses (SV) dan proses*regulator*, sering mengalami perubahan pada beban. Blok diagram kedua proses dapat dilihat pada gambar 2.5 untuk *regulator* dan 2.6 untuk *servo*. Dalam rumus penghitungan pada mode pengontrolan (persamaan matematis kontroler PID), untuk proses dengan jenis *servo* perlu ditambahkan elemen bias.

2.5.1 Kontroler "*On-Off*"

Dengan mode kontrol *on-off*, pada intinya pengontrol merupakan sebuah saklar yang diaktivasi oleh sinyal *error* dan hanya menyuplai sinyal pengoreksi *on-off*.[5:104)]Keluaran pengontrol hanya mempunyai dua nilai yang mungkin, ekuivalen dengan kondisi *on* dan *off*.Karena aksi kontrolnya yang bersifat diskontinu, serta terdapat ketertinggalan waktu di dalam sistem, maka akan

muncul kondisi osilasi dari nilai variabel yang dikontrol disekitar nilai yang diharapkan.

Kontrol *on-off* merupakan mode kontrol yang sederhana dan murah, dan sering kali digunakan dengan osilasi yang direduksi hingga level yang dapat diterima.

2.5.2 Kontroler proportional

Kontrol proporsional dirancang untuk menghilangkan osilasi disekitar nilai *set point* yang diakibatkan oleh kontrol *on-off*, sehingga kontrol proposional memiliki respond yang lebih stabil dibandingkan dengan kontrol *on-off*.

Pada kontrol proposional, besar *manipulated variable* adalah proposional terhadap besar *error* yang terjadi.[5:106)] Persamaan matematis kontroler proportional dapat dituliskan sebagai berikut [9:40)] :

$$mv_{(t)} = Kp.e_{(t)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.9)$$

Dimana :

$Kp = \frac{100\ \%}{PB}$, PB (%) :*propotional band*

Band proportional merupakan angka presentase terhadap skala penuh indikator, dimana output kontroler membuka (atau menutup) *control valve* dari posisi 0% sampai 100%. [9:40)]

Pengontrolan proposional memiliki keterbatasan berupa *steady state error* (*error* keadaan tunak) atau *offset proportional*. Semakin tinggi *gain* penguat, maka semakin rendah *offset* karena sistem akan bereaksi dengan lebih cepat. Mode kontrol proporsional cenderung digunakan dalam proses-proses dimana *gain*Kp dapat dibuat cukup besar untuk mereduksi *error* keadaan tunak hingga level yang dapat diterima. Namun, semakin besar *gain* , maka semakin besar pula peluang sistem berosilasi. Osilasi terjadi karena ketertinggalan atau jeda waktu pada sistem, dimana semakin besar *gain*, maka semakin besar aksi pengontrolan untuk suatu nilai *error* tertentu, sehingga akan semakin besar peluang bahwa sistem akan melewati nilai pengaturan dan osilasi terjadi. [5:109)]

2.5.3 Kontroler *proportionalintegral*

Dalam aplikasi dilapangan, kontrol proporsional integral biasa digunakan untuk pengontrolan proses yang memiliki dinamika relatif cepat (seperti aliran, tekanan, dan level). Berdasarkan sebuah survei dinyatakan bahwa hampir 80% kontroler PID yang terinstal di industri menggunakan kontrol PI dalam operasinya.[10:84)]

Untuk sebuah modul kontrol PID, kontrol PI dapat diperoleh dengan cara men-*setting* nilai gain (waktu) *derivative* sama dengan nol, sehingga secara matematis dapat ditulis sebagai berikut :

$$mv_{(t)} = Kp\left[e_{(t)} + \frac{1}{T_i}\int_0^t e_{(t)}\,dt\right] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.10)$$

Penambahan integrator dalam kontroler pada dasarnya dimaksudkan untuk menggantikan sinyal bias manual (atau manual reset) yang berfungsi untuk menghilangkan error *steady*. Sehingga selama masih ada error atau selisih antara *setpoint* dengan *variable proses*, output kontroler PI akan terus membesar atau mengecil.

2.5.4 Kontroler *proportionalintegral derivative*

Aksi kontrol PID pada dasarnya bertujuan untuk menggabungkan kelebihan – kelebihan komponen – komponen dasar kontrol PID :

- **Kontrol proposional**, berfungsi untuk mempercepat respon.
- **Kontrol integral**, berfungsi untuk menghilangkan *error steady*.
- **Kontrol derivatif**, berfungsi untuk memperbaiki sekaligus mempercepat respon transien.[10:93)]

Secara matematis kontrol proposional integral derivatif dapat dituliskan sebagai berikut :

$$mv_{(t)} = Kp\left[e_{(t)} + \frac{1}{T_i}\int_0^t e_{(t)}\,dt + T_d\frac{de_{(t)}}{dt}\right] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.11)$$

2.6 Tuning Kontroler

Salah satu latar belakang penggunaan kontrol PID digunakan hampir pada setiap industry yang bergerak dibidang proses adalah kesederhaan struktur kontrolnya. Selain hanya memiliki tiga parameter kontrol yang perlu diatur atau dilakukan usaha tuning (penalaan), pengaruh perubahan setiap parameter PID

terhadap dinamika pengontrolan secara intuitif mudah dipahami oleh operator.[10:70)]

Tuning kontroler (penalaan) merupakan suatu prosespengaturannilai *proportional band* (PB) atau Kp (100, *integral/reset time* (Ti), dan *derivative/rate time* (Td)pada nilai yang menghasilkan kinerja sistem pengoperasian proses optimal. [8:50)]

Kondisi optimal tuning dapat tercapai bila kinerja dari sistem kontrol dapat stabil (tidak fluktuasi), tepat (*steady state error*), dan cepat (*trasnsient respons*).Kondisi ini dapat kita lihat berdasar pada beberapa parameter.[8:50)] Berikut ini adalah tabel mengenai pengaruh tuning pada salah satu parameter PID terhadap unjuk kerja proses. [10:74)]

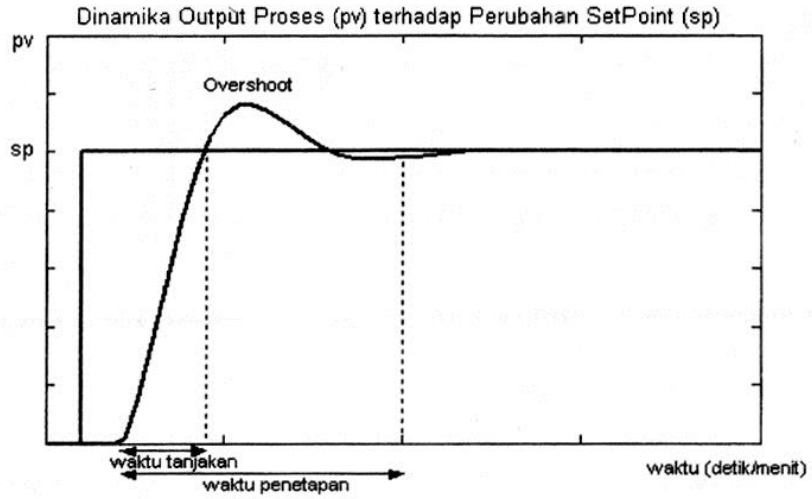
Tabel 2.1 Pengaruh tuning parameter PID terhadap unjuk kerja proses

	Waktu Tanjakan	Overshoot	Waktu Penetapan	Kestabilan
Pembesaran Kp	Berkurang	Bertambah	Sedikit Bertambah	Menurun
Pembesaran Ki (Pengecilan Ti)	Sedikit Berkurang	Bertambah	Bertambah	Menurun
Pembesaran Kd (Pembesaran Td)	Sedikit Berkurang	Berkurang	Berkurang	Meningkat

Berikut ini adalah penjelasan mengenai parameter – parameter unjuk kerja yang nampak pada tabel 3.1. [10:75)]

- **Waktu tanjakan**, waktu yang diperlukan respon (deviasi output variabel proses) untuk naik dari 0 sampai 100% harga akhirnya.

- ***Overshoot***, lonjakan maksimum yang dialami oleh respon proses.
- **Waktu penetapan**, waktu yang diperlukan respon untuk mencapai dan menetap disekitar 95%-98% dari harga akhirnya.

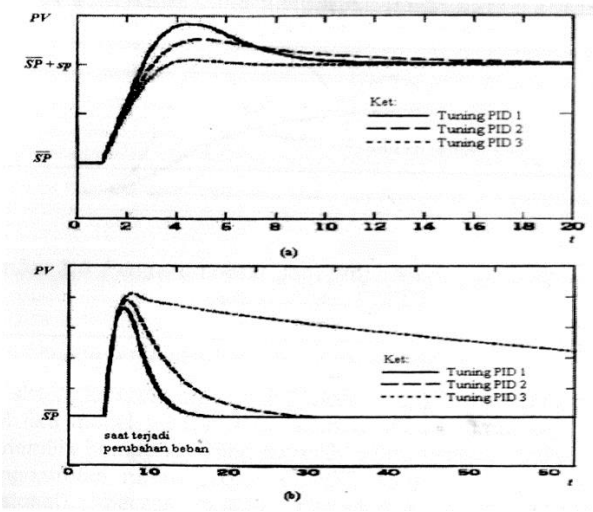


Gambar 2.9 Respon Proses sebagai akibat perubahan SV [10:75)]

Nilai parameter PID yang optimal pada dasarnya dapat dicari secara mudah berdasarkan metode – metode tuning yang telah dikembangkan oleh sejumlah peneliti. Beberapa hal yang perlu diperhatikan sebelum melakukan tuning, antara lain [10:2 & 75)]:

- **Struktur PID**, kontrol P, PI, atau PID.
- **Karakteristik proses**, walaupun pada tuning metode Ziegler & Nichols terdapat cara tuning bila tidak diketahui karakteristik prosesnya. Karakteristik proses ini dapat dicari dengan menggunakan model FOPDT dan IPDT.

- **Problem proses**, pengaruh tuning pada problem *servo* maupun *regulator* dapat dilihat pada gambar 3.13.



Gambar 2.10 (a) Problem *servo*, respon proses dalam menanggapi perubahan *setpoint*, (b) Problem *regulator*, respon proses dalam menanggapi perubahan beban (gangguan)[10:76].

2.6.1 Karakteristik Proses dengan FOPDT

Berkaitan dengan masalah pengontrolan, salah satu kunci utama keberhasilan adalah pengetahuan mengenai karakteristik dinamik atau model proses yang akan dikontrol. Pengetahuan model sangat penting mengingat secara teknis terdapat hubungan antara proses yang akan dikontrol parameter PID yang harus di-*tuning*.[10:9-17)]

Dalam banyak kasus, karakteristik dari suatu proses dapat diketahui melalui dua model matematis berikut :

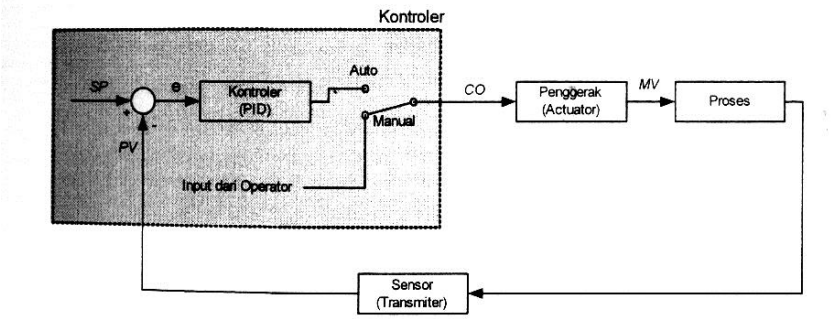
- **Model proses self regulating**, model proses yang bersifat stabil. Untuk kepentingan perancangan dan tuning parameter kontrol PID-nya, model

proses ini dapat didekati oleh sebuah model matematis yang dikenal dengan nama **FOPDT** (***F**irst **O**rder **P**lus **D**ead **T**ime*) yang hanya dicirikan oleh tiga buah parameter : Keterlambatan transportasi (**L**), Konstanta waktu (**T**), dan Gain statis proses (**K**).

- **Model proses non self regulating**, model proses yang tidak stabil. Salah satu yang sering dijumpai di industri adalah model **IPDT** (***I**ntegrating **P**lus **D**ead **T**ime*) yang hanya dicirikan oleh dua buah parameter, yaitu Keterlambatan transportasi (**L**) dan Gain integratif proses (**K***).

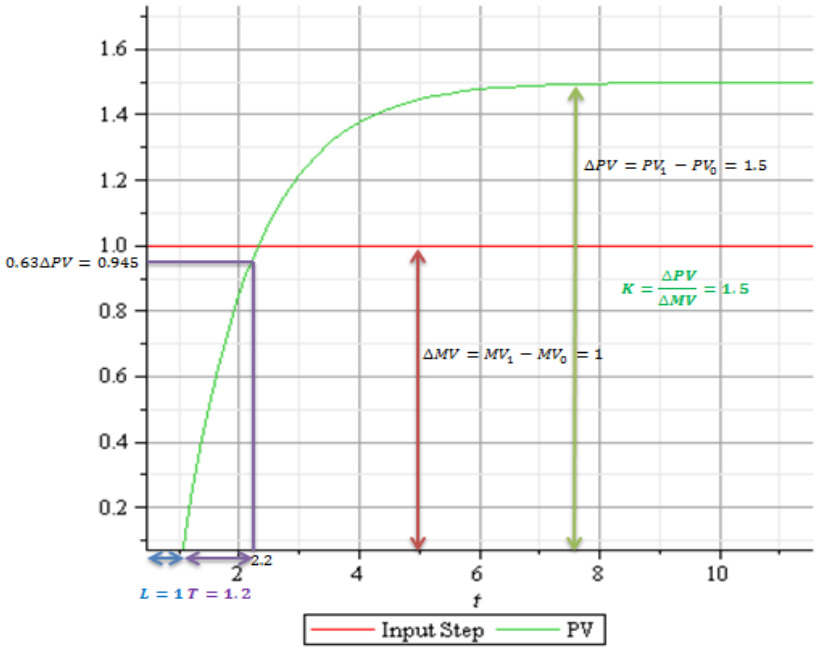
Pada sub-bab ini akan lebih banyak mengulas mengenai metode FOPDT yang paling umum dijumpai di industri proses.

Seperti telah sedikit disinggung di atas, model *self regulating process* pada dasarnya dapat didekati dengan metode matematis**FOPDT** (***F**irst **O**rder **P**lus**D**ead **T**ime*) yang hanya dicirikan oleh tiga buah parameter. Ketiga parameter yang menggambarkan dinamika proses tersebut, secara praktis dapat diperoleh atau diidentifikasi melalui eksperimen sederhana *Bump Test* atau tes sinyal tangga secara *open loop* pada mode kontrol manual, seperti pada gambar 2.14.



Gambar 2.11 Eksperimen *Bump Test* pada mode kontrol manual (*open loop*)

Secara teknis, eksperimen *bump test* dilakukan dengan cara memberi perubahan tangga (*step*) sinyal ouput kontroler oleh operator pada saat proses telah mengalami *steady* (menetap) disekitar titik kerja nominalnya. Respon variabel output (PV) kemudian direkam dan dianalisis dengan menggunakan perangkat lunak tertentu atau dapat juga dianalisis secara manual oleh operator yang bertanggung jawab pada proses tersebut.



Gambar 2.12 Respon tangga pada eksperimen *bump test* untuk model FOPDT

Dengan mengacu grafik respon tangga pada gambar 2.15 parameter – parameter proses FOPDT dapat dicari/dihitung sebagai berikut :

- Keterlambatan transportasi proses, L , waktu keterlambatan transportasi atau waktu ketidakpastian yang terjadi pada proses dihitung sejak terjadi

perubahan tangga pada MV sampai PV yang dikontrol mulai menanggapi perubahan input MV tersebut.

- Konstanta waktu proses (*time constan*), T , waktu yang diperlukan agar nilai PV mencapai kurang lebih 63% dari keadaan *steady* akhirnya. Perhitungan nilai *time constan* dimulai setelah waktu tunda berlalu. Selain dengan cara mengamati respon dari grafik, kontanta waktu proses dapat dihitung berdasarkan *gradient*atau *slpoe max* yang terjadi pada saat masa transien.

$$T = \frac{\Delta PV}{slope\ max} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (2.12)$$

Besar kecil konstanta waktu tersebut pada dasarnya menunjukkan kecepatan respon proses, semakin kecil nilai konstanta waktu, respon semakin cepat. Nilai ini pada dasarnya ditentukan oleh tiga hal utama, yaitu dimensi *plant* tempat kontrol proses berlangsung, jenis material dan beban yang terlihat pada kontrol proses, dan kekuatan atau daya penggerak.

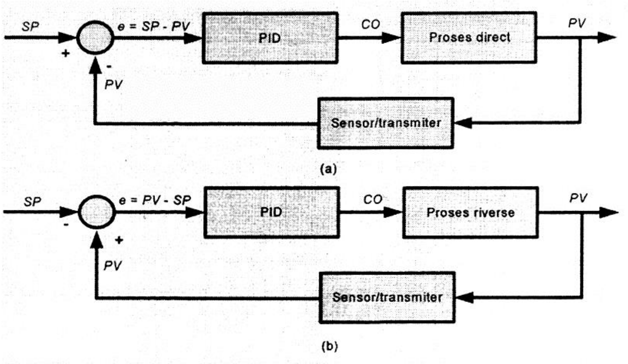
- Gain statis proses, K , perbandingan perubahan PV terhadap perubahan MV dalam keadaan *steady*.

$$K = \frac{\Delta PV}{\Delta MV} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (2.13)$$

Nilai gain proses ini secara langsung menunjukkan kesensitifan dari proses, semakin besar gain statis maka proses semakin sensitif.

Gain statis proses pada dasarnya dapat bernilai positif dan negatif tergantung dari proses dan sifat *control valve* (hal ini berbeda dengan dua

parameter proses sebelumnya yaitu, L dan T yang selalu bernilai positif). Bila K bernilai positif, maka perubahan PV terhadap perubahan MV bersifat *direct*. Sedangkan saat K bernilai negatif, maka perubahan PV terhadap perubahan MV bersifat *reverse*. Sifat *direct* dan *reverse* disebut juga sebagai mode aksi kontroler.



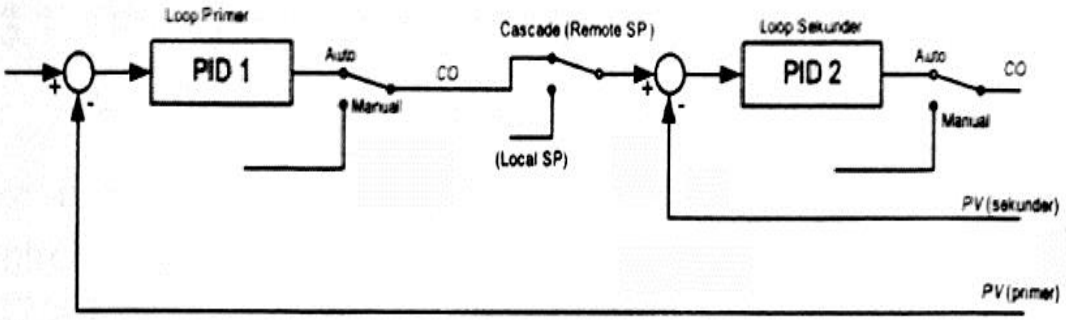
Gambar 2.13 Prinsip kerja mode aksi *reverse* (a) dan mode aksi *direct* (b)

Berdasarkan nilai ketiga parameter diatas, persamaan diferensial yang menggambarkan prilaku proses diatas secara umum dapat direpresentasikan kedalam bentuk fungsi alih proses seperti yang direlasikan oleh persamaan berikut ini :

$$H_{(s)} = \frac{K}{Ts+1} e^{-sL} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.14)$$

$H_{(s)}$: fungsi alih kontrol proses FOPDT

2.6.2 Tuning *cascade control*



Gambar 2.14Aliran Sinyal Kontrol Lengkap pada mode *Cascade*

Berikut adalah langkah-langkah untuk melakukan tuning pada sistem kontrol *cascade* :

1 Letakkan kedua loop pada mode manual.

2 Lakukan tuning terlebih dahulu pada loop sekunder, seperti pada tuning sistem kontrol PID loop tunggal. Jika parameter proses sekunder telah diketahui secara pasti, parameter kontrol PID dapat dihitung secara mudah dengan menggunakan metode tuning yang telah dijelaskan diatas.

3 Setelah setting di loop sekunder menghasilkan respon yang cukup mantap, ubah mode kontrol loop sekunder ke auto. Kemudian tuning dilakukan pada loop primer, yaitu dengan terlebih dahulu mengubah mode loop kontrol sekunder dari posisi auto ke cascade. Dengan menganggap sistem kontrol memiliki *bumplesstransfer*, secara otomatis output kontrol primer akan diinisialisasi dengan nilai setpoint dari loop sekunder.

4 Lakukan tuning dengan menggunakan metode-metode yang ada diatas.[10:194)]

2.7 Algoritma Firefly

Algoritma Firefly[11)] adalah algoritma yang terinspirasi dari kunang-kunang, mereka akan berkedip ketika berkomunuikasi dengan kunang-kunang lain dan ketika menarik calon mangsa. Algoritma ini dikembangkan oleh Xin-She Yang pada 2007, dan menggunakan 3 peraturan sebagai berikut.

1. Semua kunang-kunang berjenis kelamin satu sehingga seekor kunang-kunang akan tertarik pada kunang-kunang lain terlepas dari jenis kelamin.
2. Daya tarik sebanding dengan tingkat kecerahan cahaya kedip kunang-kuang. Oleh karena itu kunang-kunang dalam tingkat kecerahan lebih rendah akan tertarik dan bergerak ke arah kunang-kunang yangt lebih tinggi tingkat kecerahannya. Kecerahan dapat berkurang seiring dengan bertambahnya jarak dan adanya penyerapan cahaya akibat faktor udara. Jika tidak ada yang paling terang dari populasi tersebut, semua kunang-kunang bergerak acak.
3. Kecerahan dari seekor kunang-kunang dipengaruhi oleh nilai fungsi tujuan dari masalah yang diberikan. Untuk masalah maksimalisasi, intensitas cahaya sebanding dengan nilai tujuan.

Ada dua hal yang sangat penting dalam algoritma firefly yaitu intensitas cahaya dan fungsi keatraktifan, daya tarik seekor kunang-kunang ditentukan oleh seberapa terang cahaya yang dimilikinya dan cahaya itu dipengaruhi oleh fungsi obyektif. Kunang-kunang lain bergerak mendekat kepada kunang-kunang yang memiliki cahaya paling terang (kunang-kunang yang mempunyai nilai paling maksimal) dan mempunyai persamaan intensitas cahaya sebagai berikut.

$I_i = f(x)$ ... ... (2.15)

Keterangan : I_i = Intensitas Cahaya Firefly $i(i = 1, 2, \ldots..,n)$

$f(x)$ = Fungsi Obyektif

intensitas cahaya menyatakan tingkat kecerahan dari kunang- kunang, semakin cerah seekor kunang-kunang maka semakin bagus solusi kunang-kunang tersebut. pada kunang-kunang yang berdekatan maka akan timbul daya tarik menarik yang dilambangkan dengan *beta* (β). Daya tarik yang diindikasikan dengan cerah tidaknya cahaya yang dihasilkan oleh kunang-kunang.besar kecil daya tarik bersifat relative yang bergantug pada jarak kunang-kunang. Karena daya tarik kunang-kunang sebanding dengan intensitas cahaya yang dilihat kunang-kunang didekatnya maka rumus daya tarik kunang-kunang menggunakan persamaan.

$\beta = \beta_0 e^{-\gamma r^2}$ ... ... (2.16)

keterangan : β = keatraktifan

β_0 = Nilai daya tarik kunang-kunang (base beta)

γ = Koefisien penyerapan cahaya (gama)

r = Jarak

persamaan pertama yang terdiri dari variabel β_0 dengan variasi nilai antara 0 sampai 1, namun disini variabel β_0 bernilai 1 karena merupakan nilai keatraktifan awal pada kunang-kunang, dan variabel $\gamma = 0{,}01$ merupakan koefisien penyerapan cahaya awal yang mengontrol penurunan intensitas cahaya. cahaya

akan semakin berkurang (lemah) apabila jarak semakin jauh, gama (γ) mempunyai nilai bervariasi dari 0 sampai 10.

Untuk menghitung sebuah jarak dibutuhkan 2 buah objek atau lebih, karena sifat dari kunang-kunang yaitu kunang-kuang bergerak mendekat kepada kunang-kunang yang lebih terang dan bergerak secara acak, jika tidak terdapat kunang-kunang yang paling terang diantara kelompoknya, cahaya kunang-kunang akan berkurang (redup) siring bertambahnya jarak. Jarak antara kunang-kunang *i* dan *j*, diwakili oleh x_i dan x_j adalah jarak Euclidean, dimana selisih kordinat lokasi kunang-kunang i terhadap kunang-kunag j merupakan jarak diantara keduanya, dan mempunyai persamaan.

$$rij = \sqrt{\sum_{k=1}^{d}(xi,k - xj,k)^2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.17)$$

pergerakan dari kunang-kunang *i* yang bergerak kea rah kunang-kunang *j* yang lebih terang atau menuju ke tingkat intensitas cahaya yang terbaik, dtentukan dengan persamaan.

$$x'i = xi + \beta = \beta_0 e^{-\gamma r^2}(xj\text{-}xi) + \alpha\,(rand - \tfrac{1}{2}) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.18)$$

keterangan: r*ij* = jarak kunang-kunang *i* dan *j*

x'*i* = posisi kunang-kunang *i* yang baru

x*i* = posisi kunang-kunang *i* sekarang (*i* = 1, 2, …..,n)

x*j* = posisi kunang-kunang *j* sekarang (*i* = 1, 2, …..,n)

d = jumlah kordinat dimensi

x*i*,*k* = posisi kunang-kunang *i* pada dimensi *k*

x*j*,*k* = posisi kunang-kunang *j* pada dimensi *k*

β_0 = daya tarik terhadap r = 0

γ = koefisien penyerapan cahaya [0-10]

α = Parameter pengacak [0-1]

rand = nilai random [0-1]

pergerakan terjadi jika intensitas cahaya antara satu kunang-kunang lebih besar dibanding kunang-kunang yang lain. daya tarik kunang-kunang dijelaskan pada persamaan (2.16), sedangkan pergerakan acak kunang-kunang dijelaskan pada persamaan (2.17). konstanta *alpha* dan *rand* adalah sebuah vector dari nilai acak yang memiliki kisaran antara 0 sampai 1 yang diambil dari distribusi *Gaussian* atau distribusi seragam.

Adapun gambaran umum tahap metode algoritma firefly dalam menyelesaikan optimasi adalah:

1. Inisialisasi parameter dan populasi kunang-kunang.

   Proses inisialisasi dilakukan dengan cara menentukan jumlah populasi dan iterasi yang akan dilakukan serta menentukan nilai untuk variabel β_0 (*base beta*), γ (*gama*), α (*alpha*) dan rand untuk menghitung ketertarikan, jarak dari pergerakan kunang-kunang, dimana jumlah populasi yaitu jumlah calon kandidat solusi yang sudah ditentukan di penentuan kombinasi kandidat solusi pada tahapan pembuatan program *linear*.

2. Menentukan dimensi *firefly* sebagai posisi awal kunang-kunang.

   Pembangkitan bilangan acak sebagai nilai posisi awal kunang-kunang, nilai ini adalah kombinasi kandidat solusi yang sudah dibangkitkan sebelumnya, dimana terdapat dua dimensi yaitu dimensi *i* dan dimensi *j*. Banyaknya dimensi *i* ditentukan oleh banyaknya populasi *firefly* yang

merupakan jumlah kandidat solusi yang diinginkan sedangkan banyaknya dimensi *j* ditentukan oleh nilai dari kandidat tersebut.

3. Menghitung intensitas cahaya dari setiap posisi kunang-kunang.

   Untuk memperoleh nilai intensitas cahaya dibutuhkan nilai hasil dari evaluasi fungsi obyektif. Maka nilai *fitness* yang sudah didapatkan sebelumnya dijadikan sebagai nilai intensitas cahaya setiap kunang-kunang. Karena tujuan permasalahan adalah mencari nilai maksimal, jadi semakin tinggi nilai fungsi, maka semakin tinggi nilai intensitasnya, dijelaskan pada persamaan (2.15).

4. Menentukan *Gbest* (*Global best*).

   Nilai *Gbest* merupakan nilai intensitas cahaya paling terang dari semua kunang-kunang, setelah didapat nilai intensitasnya, setiap kunang-kunang dibandingkan untuk mencari intensitas cahaya paling tinggi, kemudian posisi kunang-kunang yang memiliki intensitas cahaya paling tinggi akan dijadikan *Gbest*, dijelaskan pada persamaan (2.18).

5. Menghitung jarak posisi antar kunang-kunang.

   Perhitungan jarak setiap kunang-kunang terhadap kunang-kunang yang intensitas cahayanya paling tinggi (*Gbest*) dengan menggunakan metode *eucledean*, dijelaskan pada persamaan (2.17).

6. Menghitung nilai keatraktifan antar kunang-kunang.

   Melakukan perhitungan nilai beta menggunakan konstanta base beta, gama, dan jarak yang sudah diketahui sebelumnya, dijelaskan pada persamaan (2.16).

7. Melakukan pergerakan pada kunang-kunang.

   Kunang-kunang akan bergerak menuju kunang-kunang yang lebih besar intensitas cahayanya sehingga diperoleh posisi baru, kecuali untuk *Gbest* posisinya tidak berubah karena tidak melakukan pergerakan.

8. Memperbaruhi posisi kunang-kunang yang sudah bergerak.

   Perubahan posisi kunang-kunag akan diurutkan dari yang terkecil hingga yang terbesar melalui aturan *Smallest Position Value* (*SPV*), untuk mengubah setiap dimensi dalam nilai posisi kunang-kunang menjadi bentuk urutan permutasi sehingga terbentuk kandidat solusi yang baru.

9. Mengevaluasi nilai *fitness* kunang-kunang pada posisi baru.

   Setelah mendapatkan dan ditempatkan pada posisi baru, nilai *fitness* yang dihasilkan dievaluasi kembali oleh kandidat solusi baru.

10. Menentukan level kunang-kunang berdasarkan intensitas cahaya baru.

    Nilai *fitness* yang dihasilkan oleh kandidat solusi yang baru tersebut akan digunakan sebagai nilai intensitas cahaya kunang-kunag yang baru.

11. Mengecek parameter konvergen pada kandidat solusi.

    Jika posisi kunang-kunang mencapai posisi tujuan atau posisi terbaik maka nilai kunang-kunang dikatakan konvergent, apabila nilai baru melebihi batas minimal dan maksimal pada masing-masing dimensi atau belum mencapai iterasi maksimum, kembali ke nilai posisi awal dan ulangi langkah awal dengan mencari kunang-kunang *Gbest* baru sampai menemukan solusi terbaik atau iterasi selesai.