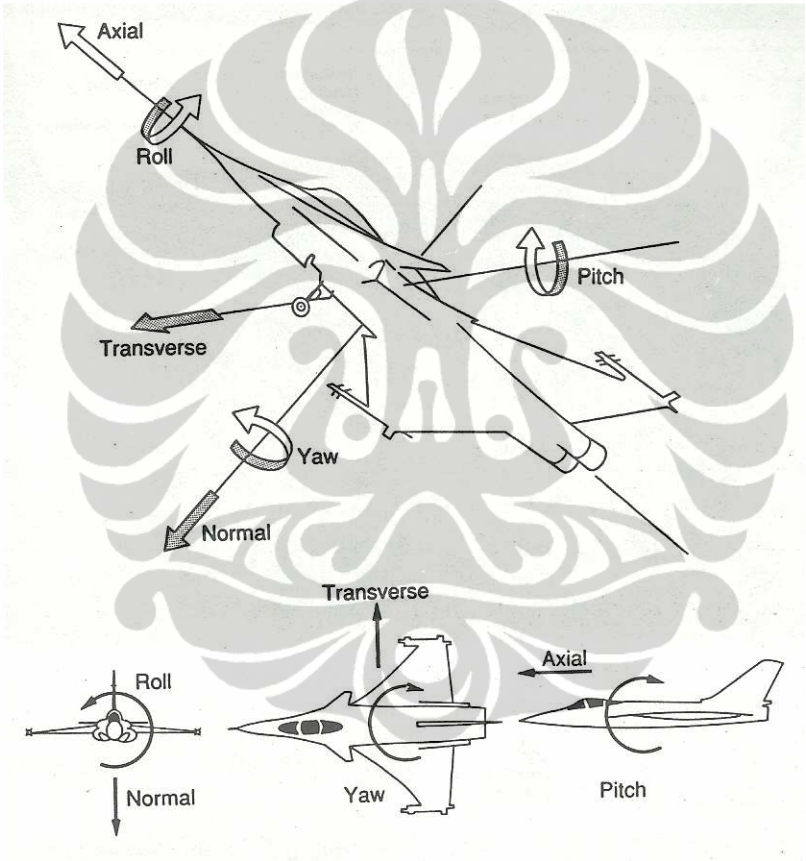
# BAB 2
# TINJAUAN PUSTAKA

## 2.1 Prinsip Dasar Kontrol Pesawat Terbang

Sebuah wahana terbang, yaitu pesawat terbang, memiliki bagian-bagian yang sangat menentukan untuk dapat terbang, sehingga memungkinkannya untuk bergerak dalam enam posisi derajat kebebasan (*six degree of freedom*), seperti terlihat pada gambar 2.1.




Gambar 2.1. Enam derajat kebebasan pada pesawat [1]

Untuk dapat bergerak dalam enam derajat kebebasan tersebut, pesawat terbang memiliki beberapa bidang kontrol gerak yang akan berpengaruh pada masing-masing derajat kebebasan. Beberapa bidang kontrol tersebut adalah sebagai berikut:

2.1.1 *Aileron*

*Aileron* adalah bidang kontrol gerak wahana terbang yang berfungsi untuk menggerakkan wahana dengan gerak *roll*. *Aileron* ini biasa terdapat pada sayap pesawat terbang, terletak pada bagian sebelah luar/ pinggir dari pesawat. Gerakan *aileron* ini akan saling berlawanan arah dari masing-masing *aileron* dengan besar sudut yang sama.

2.1.2 *Elevator*

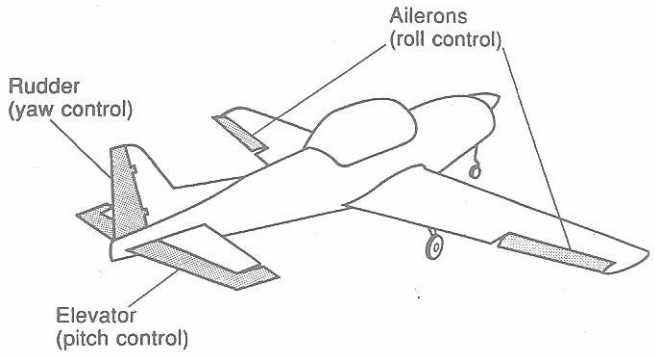
*Elevator* adalah bidang kontrol gerak pesawat terbang yang berfungsi untuk mengatur gerakan *pitch* pada pesawat. *Elevator* biasanya berada pada bagian ekor pesawat terbang dengan arah horisontal.

2.1.3 *Rudder*

*Rudder* adalah bidang kontrol gerak pesawat terbang yang berfungsi untuk mengatur gerakan *yaw* pada pesawat. *Rudder* biasanya berada pada bagian ekor pesawat terbang dengan arah vertikal.

2.1.4 *Throttle*

Selain ketiga bidang kontrol gerak seperti disebutkan di atas, ada satu bagian lain yang tidak dapat dipisahkan dari pesawat terbang, yang juga akan memberikan pengaruh dari perubahan sudut elevasi dari masing-masing bidang kontrol gerak tersebut, yaitu *throttle*. *Throttle* berfungsi untuk mengatur *thrust* / gaya dorong dari mesin pesawat, sehingga akan berpengaruh pada kecepatan pesawat. Posisi dari masing-masing bidang kontrol gerak tersebut seperti terlihat pada gambar 2 di bawah ini.

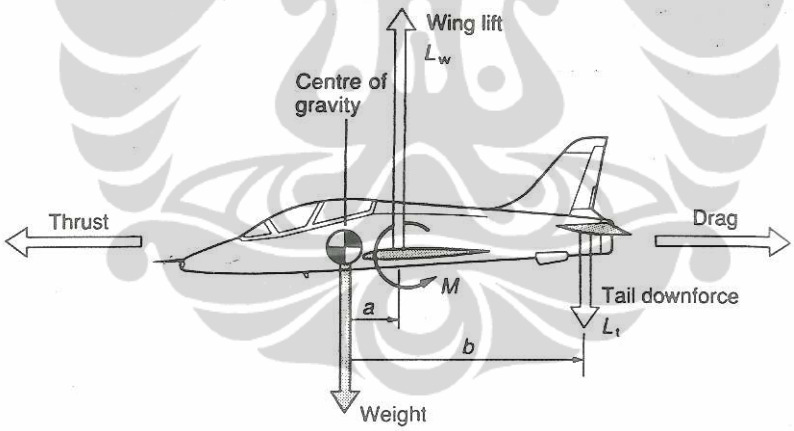



Gambar 2.2. Posisi bidang kontrol gerak pada pesawat konvensional [1]

## 2.2. *Static Stability*

Kestabilan terbang adalah sebuah harga mutlak pada sebuah wahana terbang, seperti pesawat terbang. Dalam keadaan terbang (*steady flight*), gaya-gaya yang bekerja pada pesawat harus seimbang, dan tidak boleh ada sedikitpun resultan gaya yang dapat menimbulkan momen putar pada tiap aksis. Jika hal ini telah terpenuhi, maka pesawat dapat dikatakan seimbang (*trimmed*).

Sebuah pesawat dapat dikatakan *statically stable*, apabila ia akan kembali menuju ke kondisi inisial terbang, dalam hal ini *attitude*, kecepatan, dan lain-lain, setelah mendapat gangguan dari luar, maupun dari gerakan *impuls* kecil dari bidang kontrol gerak. Secara umum, untuk dapat terbang normal, kondisi *trimmed* dan stabil sangat dibutuhkan [1]. Sebagaimana ditunjukkan pada gambar 2.3, pesawat yang pada posisi flight mengalami momen yang berlawanan arah jarum jam dari efek *wing lift*, maka akan diseimbangkan dengan momen searah jarum jam dari bidang kontrol *elevator*.




Gambar 2.3. Pesawat dalam kondisi *trimmed*

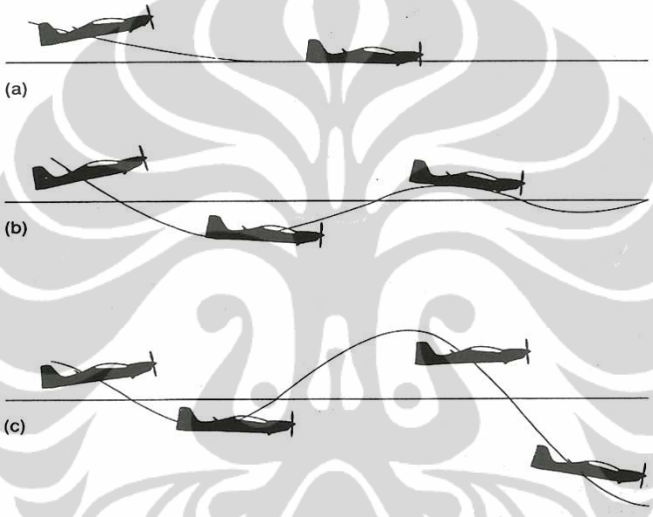
## 2.3. *Longitudinal dan Lateral Stability*

Dalam sub bab sebelumnya telah dijelaskan mengenai tiga gerakan pada pesawat, yaitu *pitch*, *yaw* dan *roll*. *Longitudinal stability* adalah kestabilan terbang pesawat yang dipengaruhi dari gerakan pitching (*pitching stability*). Sedangkan lateral stability adalah kestabilan terbang pesawat yang dipengaruhi dari gerakan *rolling* dan *yawing*. Gerak *rolling* dan *yawing* ini saling mempengaruhi antara satu dengan lainnya (terdapat kopel). [1]

Pada pesawat konvensional, kopel antara kestabilan longitudinal dan lateral hampir tidak ada, sehingga dalam pembahasannya maupun teknik kontrolnya dapat dipisahkan. Kecuali pada pesawat dengan gerak manuver yang sangat tinggi seperti pada pesawat tempur, maka kopel antara kestabilan longitudinal dan lateral tetap harus diperhitungkan.

## 2.4. *Dynamic Stability*

Kestabilan dinamik pada pesawat dapat diilustrasikan sebagaimana pada gambar 2.4 berikut ini.

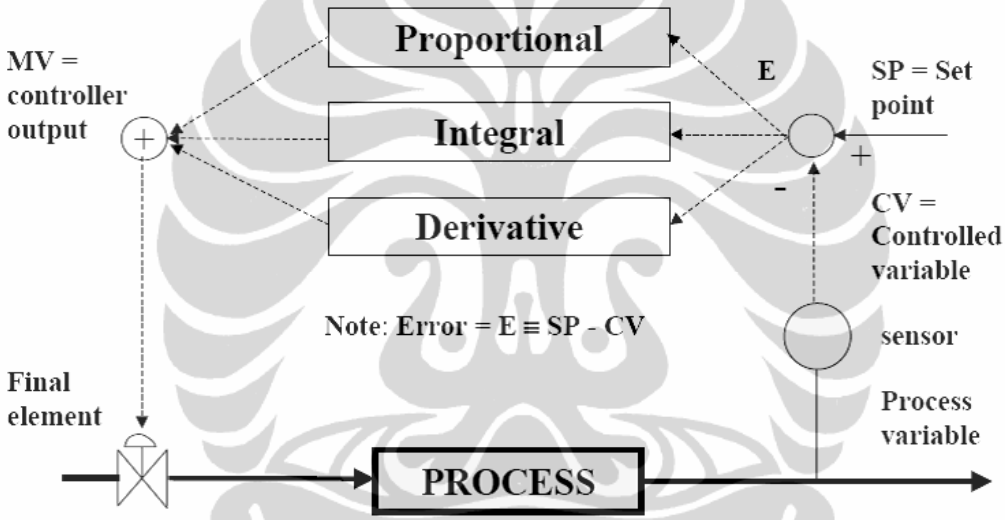



Gambar 2.4. Gerakan osilasi pada *longitudinal motion* [1]

Gambar 2.4 di atas mengilustrasikan konsep tentang *static* dan *oscillatory dinamic stability* pada pesawat yang sedang terbang pada sumbu longitudinal, tanpa ada gerakan *roll*. Saat pesawat diganggu dengan perubahan sudut terbang (*angle of attack*), maka ada tiga kemungkinan respon pesawat yang akan terjadi. Pada gambar 2.4.a, pesawat akan langsung menuju ke posisi inisialnya sebagaimana sebelum adanya gangguan, tidak terjadi osilasi, sehingga dapat dikatakan *heavily damped*. Pada gambar 2.4.b, terlihat gerakan osilasi, namun dengan amplitudo semakin kecil hingga akhirnya tidak terdapat osilasi kembali. Di sini dapat dikatakan bahwa pesawat dalam kondisi *dynamically* dan *statically stable*. Pada gambar 2.4.c, terlihat terjadi osilasi dengan amplitudo yang semakin membesar, sehingga dikatakan bahwa pesawat dalam kondisi *dynamically unstable negatively damped*.[1]

## 2.5. Kontrol PID

PID (*Proportional Integral Derivative*) adalah salah satu teknik kontrol yang telah lama dikenal luas. Kontroler PID telah banyak diimplementasikan pada banyak sektor, terutama industri sejak tahun 1940-an hingga sekarang. Kontroler PID adalah sebuah kontroler dengan satu input dan satu output (*single loop, SISO*), sehingga hanya dapat digunakan pada *plant* tunggal dengan satu *controlled variable (CV)* dan satu *manipulated variable (MV)*. Gambar 2.5 berikut adalah sebuah gambaran skematik tentang kontrol PID.




Gambar 2.5 Skematik kontrol PID [2]

Sebagaimana terlihat pada gambar 2.5, kontroler PID tersusun dari tiga mode kontrol, yaitu proporsional, integral dan *derivative*. Kontrol proporsional akan berfungsi untuk mempercepat respons dengan *gain* (Kc), kontroler integral akan berfungsi untuk memperkecil nilai *offset*, sedangkan kontroler *derivative* akan memperbaiki respon *transien*. Salah satu keunikan dari kontroler PID ini adalah kontroler ini dapat digunakan secara terpisah maupun gabungan dari 2 atau 3 kontroler, yaitu proporsional saja (P), proporsional-Integral (PI) atau gabungan ketiganya, proporsional-integral-*derivative* (PID).

2.5.1 Kontrol Proporsional

Kontroler Poporsional berfungsi sebagai gain yang secara proporsional 'searah' dengan besar nilai error yang terjadi. Secara umum kontrol proporsional dapat ditulis sebagai :

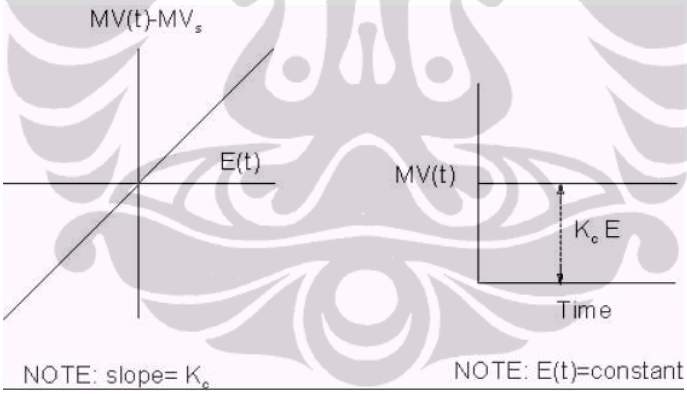
$$MV(t) = K_c E(t) + I_p \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.1)$$

$$G_C(s) = \frac{MV(s)}{E(s)} = K_C \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.2)$$

Kontroler *gain Kc* adalah parameter pertama dari tiga parameter dalam kontrol PID yang akan sangat mempengaruhi parameter lainnya.

Ip adalah konstanta inisialisasi awal dari kontroler PID ini untuk mengetahui kondisi awal dari suatu sistem, dimana konstanta ini akan selalu ditambahkan pada iterasi PID selanjutnya.

Gambar 2.6 berikut merupakan gambaran dari kontrol proporsional, yang merupakan grafik antara *manipulated variable* dengan *error*.




Gambar 2.6 Plot antara *manipulated variable* dengan *error* pada kontrol proporsional [2]

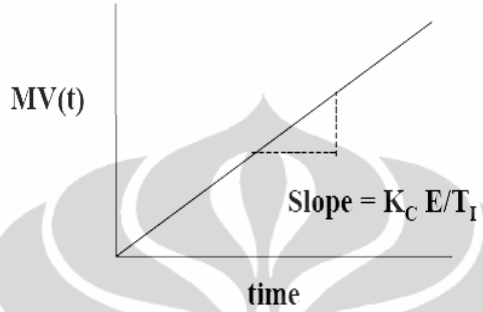
2.5.3 Kontrol Integral

Pada kontrol integral, dilakukan integrasi terhadap error dalam waktu integral, Ti, sehingga proses integral akan menghilangkan *steady state error*. Proses integral ini dapat ditulis pada persamaan sebagai berikut:

$$MV(t) = \frac{K_c}{T_I}\int_0^{\infty} E(t')dt' + I_I \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.3)$$

$$G_C(s) = \frac{MV(s)}{E(s)} = \frac{K_C}{T_I}\frac{1}{s} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.4)$$

Kontrol integral ini akan mempercepat sistem menuju ke *zero steady state offset*, sehingga menggabungkan mode kontrol integral ini dengan kontrol proporsional akan menjadi kontrol yang lebih optimal. Gambar 2.7 berikut merupakan gambaran dari kemampuan kontrol integral. Dengan *error* konstan, kontrol integral akan menaikkan nilai dari *manipulated variable* secara linear dengan *slope* = Kc E(t)/Ti.




Gambar 2.7 Sifat kontrol integral [2]

2.5.4 Kontrol Derivative

Kontrol derivative bekerja dalam konteks *rate error*, sehingga nilai dari kontroler ini akan naik jika *rate error* naik, dan apabila nilai *error* adalah konstan/ tidak berubah, maka nilai dari kontroller ini akan nol pula. Persamaan dari proses kontrol *derivative* ini dapat ditulis sebagai berikut:

$$MV(t) = K_c T_D \frac{dE(t)}{dt} + I_D \quad \text{.....} \quad (2.5)$$

$$G_C(s) = \frac{MV(s)}{E(s)} = K_c T_d s \quad \text{.....} \quad (2.6)$$

Persamaan 2.7 di atas memperlihatkan proses kontrol *derivative* dengan mengacu pada nilai *error* dari *set point*. Dengan cara yang berbeda namun dapat menghasilkan kontrol yang lebih baik, adalah dengan melakukan proses kontrol *derivative* dengan mengacu pada nilai dari *controlled variable* (CV) [10], yang dapat dilihat pada persamaan 2.8 berikut:

$$MV(t) = -K_c T_D \frac{d\,CV(t)}{dt} + I_D \quad \text{.....} \quad (2.7)$$

2.5.5 Kontroler PID

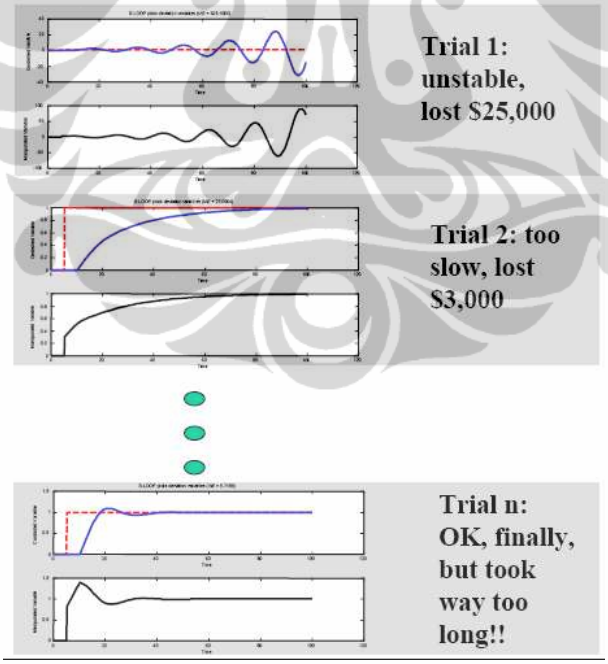
Kontroler PID merupakan gabungan dari tiga mode kontrol sebagaimana telah dijelaskan sebelumnya. Dengan demikian, persamaan dari kontroler PID dapat dituliskan sebagai berikut:

$$E(t) = SP(t) - CV(t)$$

$$MV(t) = K_c\left[E(t) + \frac{1}{T_I}\int_0^{\infty} E(t')dt' - T_d\frac{d\,CV}{dt}\right] + I \quad \text{..... (2.8)}$$

## 2.6 Kontroler PID *Tuning*

Dalam kontroler PID, penentuan konstanta-konstanta Kc, Ti dan Td adalah suatu hal yang sangat penting. Dengan nilai-nilai dari konstanta-konstanta tersebutlah suatu kontroler PID terlihat kemampuannya. Nilai konstanta Kc, Ti dan Td yang tidak tepat akan mengakibatkan kontrol yang kurang sempurna, bahkan dapat membuat suatu sistem menjadi tidak stabil. Ada beberapa cara dalam menentukan nilai dari konstanta-konstanta tersebut. Cara 'termudah' yang beberapa orang menggunakannya adalah dengan "*trial and error*". Namun hal ini dapat mengakibatkan suatu pemborosan, baik dari segi waktu maupun "*cost*" yang dikeluarkan. Sebagai sebuah gambaran, gambar 2.9 berikut menunjukkan sebuah proses pencarian nilai dari konstanta Kc, Ti dan Td dengan cara *trial and error*.



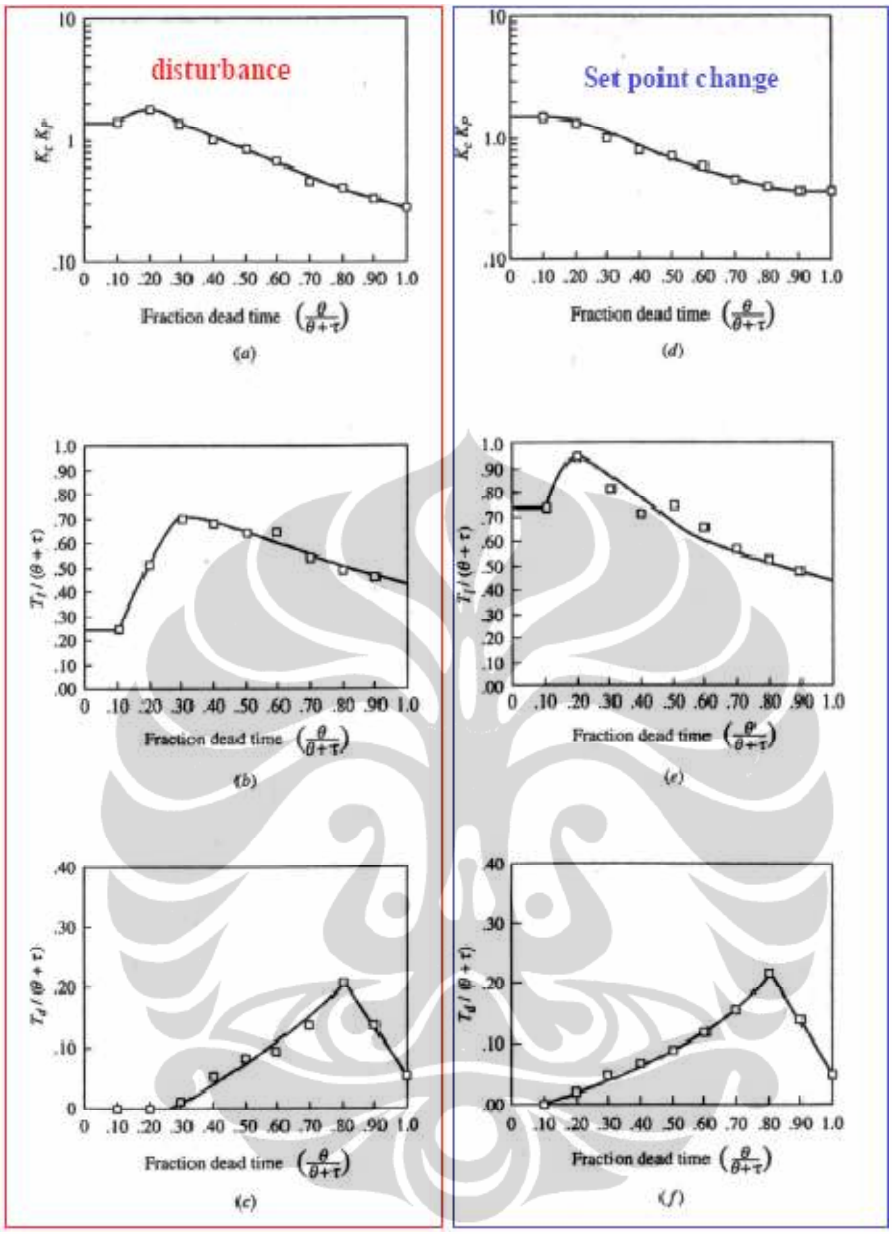

Gambar 2.8 Gambaran *cost* trial and error [3]

Cara lain untuk menentukan nilai konstanta dari Kc, Ti dan Td adalah dengan teknik PID *tuning*, dimana penentuan nilai konstanta-konstanta tersebut dengan berdasarkan dinamika kelakuan sistem. Beberapa masalah yang harus diperhatikan dalam proses *tuning* ini, yaitu:

- *Process dynamics*. Dinamika proses dari suatu sistem dapat dilihat dari respons sistem terhadap perubahan sinyal acuan (*step respons*).
- *Measured variable*. Variabel yang terukur dapat memperlihatkan respons dinamik dari suatu sistem, termasuk *noise* dari sensor dan gangguan luar dari proses sistem.
- *Model error*. Dengan mengetahui *range* nilai *error* dari model proses yang digunakan, maka dapat ditentukan nilai-nilai 'tengah' dari batasan-batasan *error* tersebut.
- *Input forcing*. *Step input disturbance* dan *step input* dari *set point*.
- *Controller*. Jenis kontroler yang digunakan, dalam hal ini PID atau PI.
- *Performance measure.* Mengetahui behavior dari *controlled variable* yang akan meminimaliskan IAE dan zero offset, behavior dari *manipulated variable* yang akan mengetahui batasan-batasan dari MV. [2]

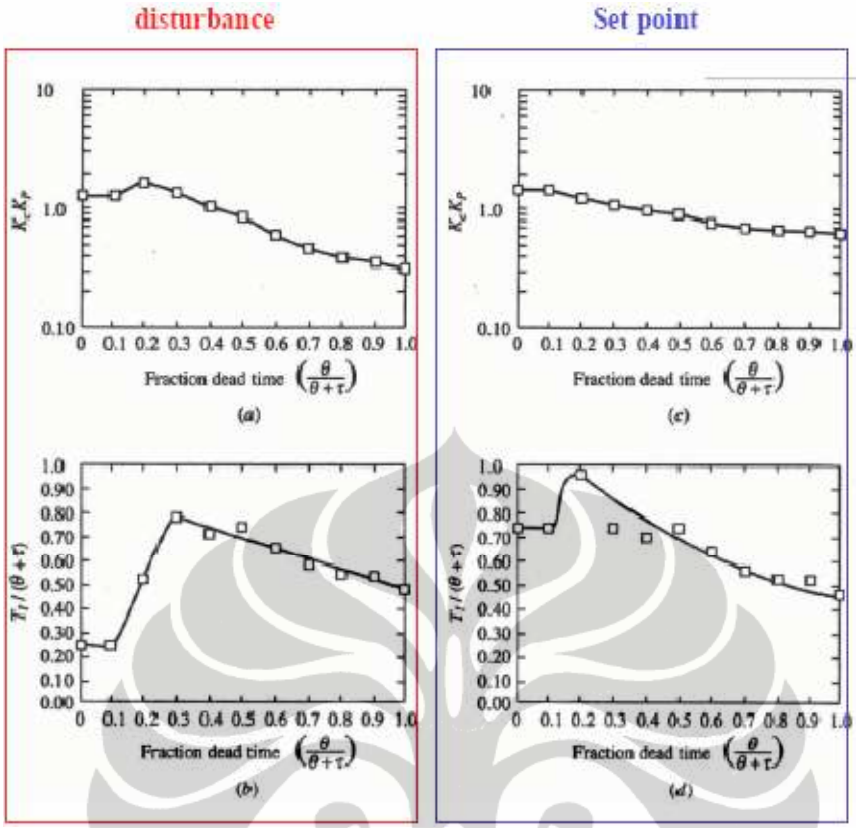
## 2.7 Ciancone Correlation

*Ciancone correlation* pertama kali dibangun oleh Ciancone dan Marlin (1992). *Ciancone correlation* ini menggunakan tabel *ciancone* untuk menentukan nilai-nilai dari Kc, Ti dan Td. Gambar 2.9 berikut adalah ciancone chart yang digunakan untuk menentukan nilai-nilai dari Kc, Ti dan Td untuk kontroler PID.




Gambar 2.9. Ciancone chart untuk kontroler PID. Untuk disturbance respons: (a) control system gain, (b) integral time, (c) derivative time. Untuk set point respons: (d) gain, (e) integral time, (f) derivative time. [3]

Sedangkan *ciancone chart* untuk menentukan nilai dari Kc, Ti dan Td untuk kontroler PI diperlihatkan pada gambar 2.10 berikut.



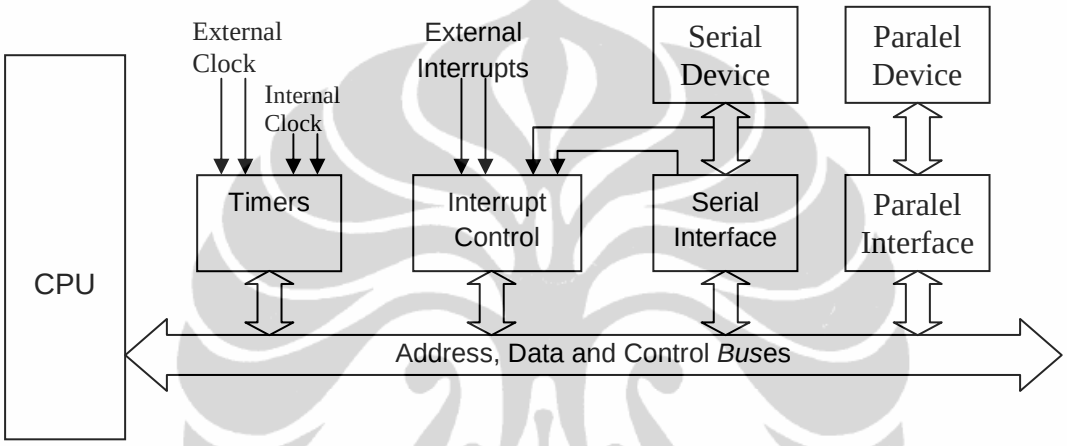

Gambar 2.10 Ciancone chart untuk kontroler PI. Untuk disturbance respons: (a) control system gain, (b) integral time. Untuk set point respons: (c) gain, (d) integral time. [3]

Langkah-langkah dalam melakukan tuning kontroler dengan teknik ciancone correlation adalah sebagai berikut:

1) Dapatkan nilai-nilai dari Kp, θ dan τ dari model dinamik sistem dengan menggunakan metode *empirical*.
2) Hitung *fraction dead time*, θ/(θ+τ).
3) Pilih tabel yang sesuai, dengan *disturbance respons* atau *set point respons*.
4) Tentukan nilai dari *dimensionless tuning* dari grafik untuk KcKp, Ti/(θ+τ) dan Td/(θ+τ).
5) Hitung *dimensional tuning controller*. Misal: Kc=(KcKp)/Kp.
6) Implementasikan ke dalam kontroler.[2]

## 2.7 Sistem Mikrokontroler ATMega32

Secara umum, sistem mikroprosesor akan terdiri dari beberapa komponen antara lain: *CPU* sebagai pemroses data/ program, *ROM* sebagai memori program, *RAM* sebagai memori data, dan *PIO* sebagai rangkaian perantara (*Interface*) untuk menghubungkan sistem ini dengan berbagai alat masukan dan keluaran. Dengan demikian sistem ini disebut sebagai sebuah sistem mikrokomputer. Gambar 2.11 berikut adalah blok diagram sebuah sistem minimum sebuah mikrokontroler.




Gambar 2.11 Blok diagram sistem minimum mikrokontroler [4]

Mikrokontroler ATMega32 merupakan salah satu keluarga mikrokontroler Atmel, dengan beberapa blok dan fungsi tambahan selain dari sistem minimum sebagaimana pada gambar 2.11 di atas. ATMega32 merupakan *low cost* mikrokontroler 8 bit, dengan kecepatan eksekusi 1 MIPS per MHz. Kecepatan maksimal dari mikrokontroler ini adalah 16 MHz. Beberapa fitur dari mikrokontroler ini yang sangat berguna bagi penelitian ini adalah fasilitas *In System Programming* (ISP) yang memudahkan ketika proses pemrograman berlangsung, kapasitas *flash* untuk program memori 32 Kbytes, dua buah *timer* 8 bit dan satu buah *timer* 16 bit, 8 kanal 10 bit ADC, *programmable serial* USART dan 32 *programmable I/O lines*.

### 2.7.1 Dasar Kerja Program *Flash* Mikrokontroller ATMega32

Program untuk mengendalikan kerja dari mikrokontroler disimpan di dalam memori program. Program pengendali tersebut merupakan kumpulan dari instruksi kerja mikrokontroller. Sepanjang mikrokontroller bekerja, instruksi

tersebut *byte* demi *byte* diambil ke *CPU* dan selanjutnya dipakai untuk mengatur kerja mikrokontroler. Proses pengambilan instruksi dari memori program dikatakan sebagai '*fetch cycles*' dan saat-saat CPU melaksanakan instruksi disebut sebagai '*execute cycles*'. Semua mikrokontroler maupun mikroprosesor dilengkapi sebuah register yang berfungsi khusus untuk mengatur '*fetch cycles*', register tersebut dinamakan sebagai *Program Counter*. Nilai *Program Counter* secara otomatis bertambah satu setiap kali selesai mengambil 1 *byte* isi memori program, dengan demikian isi memori program bisa berurutan diumpankan ke CPU.

Saat AVR direset, isi *Program Counter* direset menjadi 0000. Artinya sesaat setelah reset isi dari memori program nomor 0 dan seterusnya akan diambil ke *CPU* dan diperlakukan sebagai instruksi yang akan mengatur kerja mikrokontroler. Dengan demikian, awal dari program pengendali AVR harus ditempatkan di memori nomor 0, setelah reset AVR menjalankan program mulai dari memori-program nomor 0000, dengan melakukan proses '*fetch cycles*' dan '*execute cycles*' terus menerus tanpa henti.

Jika sarana *interupsi* diaktifkan, maka proses menjalankan program di atas akan dihentikan sebentar, mikrokontroler melayani dulu permintaan *interupsi*, selesai melayani permintaan *interupsi CPU* akan melanjutkan mengerjakan program utama lagi. Selesai melayani interupsi, nilai *Program Counter* yang tadi disimpan ke dalam *Stack* akan dikembalikan ke *Program Counter*, dengan demikian CPU bisa melanjutkan pekerjaan di program utama.[5]