TUGAS AKHIR – MN141581

Alat Bantu Edukasi (Digital) Untuk Mempermudah Pemahaman Arti Fisik Frekuensi Natural Dan Mode Shape Getaran Longitudinal Dan Torsional Sistem Propulsi Kapal

PURNOMO ADHI WICAKSONO
NRP. 4111 100 034

Ir. ASJHAR IMRON, M.Sc, MSE, PED.

JURUSAN TEKNIK PERKAPALAN
Fakultas Teknologi Kelautan
Institut Teknologi Sepuluh Nopember
Surabaya
2015

FINAL PROJECT – MN141581

The Educational Tools (Digital) To Facilitate Understanding a Physical Meaning of Natural Frequency and Mode Shape Longitudinal And Torsional Vibration of Ship Propulsion System

PURNOMO ADHI WICAKSONO
NRP. 4111 100 034

Ir. ASJHAR IMRON, M.Sc, MSE, PED.

DEPARTMENT OF NAVAL ARCHITECTURE & SHIPBUILDING ENGINEERING
Faculty of Marine Technology
Sepuluh Nopember Institute of Technology
Surabaya
2015

LEMBAR PENGESAHAN

Alat Bantu Edukasi (Digital) Untuk Mempermudah Pemahaman Arti Fisik Frekuensi Natural Dan Mode Shape Getaran Longitudinal Dan Torsional Sistem Propulsi Kapal

TUGAS AKHIR
Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
pada
Bidang Keahlian Rekayasa Perkapalan – Konstruksi
Program S1 Jurusan Teknik Perkapalan
Fakultas Teknologi Kelautan
Institut Teknologi Sepuluh Nopember

Oleh:

PURNOMO ADHI WICAKSONO
NRP. 4111 100 034

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Ir. Asjhar Imron, M.Sc, MSE, PED
NIP.19510918 197803 1 003

SURABAYA, 2015

Alat Bantu Edukasi (Digital) Untuk Mempermudah Pemahaman Arti Fisik Frekuensi Natural Dan Mode Shape Getaran Longitudinal Dan Torsional Sistem Propulsi Kapal

Nama Mahasiswa : Purnomo Adhi Wicaksono

NRP : 4111.100.034

Jurusan : Teknik Perkapalan, Fakultas Teknologi Kelautan

Institut Teknologi Sepuluh Nopember

Dosen Pembimbing: Ir. Asjhar Imron, M.Sc, MSE, PED.

ABSTRAK

Masalah getaran longitudinal dan torsional sistem propulsi kapal dianalisa dengan menggunakan model diskrit sistem getaran dengan banyak derajat kebebasan. Beberapa cara untuk menentukan parameter-parameter dalam perhitungan getaran longitudinal dan torsional dibahas dan cara-cara penentuan banyaknya derajat kebebasan juga dibicarakan. Metode numerik yang efektif dipilih dengan mempertahankannya se-general mungkin sehingga bisa digunakan tidak hanya untuk masalah getaran saja, namun juga dalam masalah lain yang sebanding. Komputer algoritma dikembangkan menggunakan metode iterasi yang seefektif mungkin sehingga konvergensi bisa didapat dengan cepat tanpa harus mengorbankan ketelitian. Hasil yang didapatkan berupa besarnya frekuensi natural, mode shape dan animasi yang bertujuan agar mempermudah dalam pemahaman frekuensi natural dan mode shape. Sebagai tambahan, program ditambahkan perhitungan dari massa, momen inersia massa dan kekakuan pegas.

Kata Kunci: Getaran longitudinal dan torsional sistem propulsi, Frekuensi Natural, Mode Shape.

The Educational Tools (Digital) To Facilitate Understanding a Physical Meaning of Natural Frequency and Mode Shape Longitudinal And Torsional Vibration of Ship Propulsion System

Student Name: Purnomo Adhi Wicaksono

ID. No. : 4111.100.034

Departement :Naval Architecture, Ocean Technology Faculty, Institute of Technology Sepuluh Nopember

Supervisor : Ir. Asjhar Imron, M.Sc, MSE, PED.

ABSTRACT

The Longitudinal and torsional vibration problems of ship propulsion system is analyzed with a discrete model with many degrees of freedom. Several methods to determine the parameters in the calculation of longitudinal and torsional vibrations and the number of degrees of freedom are discussed. The effective numerical method choosen is not limited to solve ship propulsion vibration problems. But can also be used for common vibration problem with minor modification. The algorithm is developed using an effective iterative method that it converges quickly determined without sacrificing precision. The resulting natural frequencies and mode shapes is animated to facilitate better understanding of the concept. As a bonus, pre-processor software is added to facilitate the calculation of mass of the object, mass moment of inertia, and spring stiffness.

Keywords: Longitudinal and Torsional vibration of propulsion system, natural frequencies, Mode Shapes

KATA PENGANTAR

Bismillahirrahmanirrahiim.

Puji syukur kepada Tuhan Yang Maha Esa karena atas karunianya Tugas Akhir ini dapat selesai dengan baik.

Pada kesempatan ini Penulis ingin mengucapkan terima kasih kepada pihak-pihak yang membantu penyelesaian Tugas Akhir ini, yaitu:

1. Bapak Ir. Ashjar Imron, M.Sc, MSE, PED. selaku dosen pembimbing yang dengan sabar telah memberikan bimbingan ilmu dan arahan dalam menyelesaikan tugas akhir ini.
2. Tim penguji ujian tugas akhir Jurusan Perkapalan yang telah memberikan ide / masukan dalam penyelesaian tugas akhir ini.
3. Bapak Prof. Ir. I.K.A.P Utama, M.Sc., Ph. D selaku Ketua Jurusan Teknik Perkapalan yang memberikan inspirasi dan motivasi kepada penulis.
4. Bapak Prof. Ir. Djauhar Manfaat, M.Sc, Ph.D. selaku dosen wali yang sejak awal perkuliahan banyak membantu penulis.
5. Dosen – dosen Jurusan Teknik Perkapalan khususnya Bidang Studi Rekayasa Perkapalan,terima kasih saya ucapkan atas bimbingan, ilmu serta motivasi yang telah diberikan selama di bangku perkuliahan.
6. Kedua Orang Tua (Ir. Didik Soesanto dan Indah Nursanti) , Adik (Dinda Amalia Candela dan Dinda Al-Khwarizmi curie) dan Keluarga yang selalu memberikan dorongan semangat, doa yang tulus ikhlas serta memberikan kesempatan penulis untuk melanjutkan studi di bangku perkuliahan ini.
7. Teman – teman “P-51“ pada umumnya, yang selalu saling memberikan semangat dan motivasi dalam terselesaikannya tugas akhir ini.
8. Semua pihak yang telah membantu dalam penyelesaian Tugas akhir ini yang tidak dapat penulis sebutkan satu persatu.

Penulis sadar bahwa Tugas Akhir ini masih jauh dari kesempurnaan sehingga kritik dan saran yang bersifat membangun sangat diharapkan. Akhir kata semoga tulisan ini dapat bermanfaat bagi banyak pihak.

Surabaya, Juni 2015

Purnomo Adhi Wicaksono

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Dengan semakin cepatnya kemajuan teknologi khususnya dibidang perkapalan menyebabkan semakin besar pula tuntutan akan kenyamanan, kekuatan, dan ketahanan suatu kapal. Getaran pada kapal merupakan salah satu dari sejumlah hal-hal penting yang perlu mendapatkan perhatian khusus dalam proses desain kapal.

Apabila dilihat dari segi kekuatan kapal, salah satu kegagalan sturktur yang terjadi pada kapal disebebkan karena getaran yang berlebihan yang terjadi pada sistem propulsi kapal. Getaran yang berlebihan terjadi karena adanya resonansi antara RPM propeller dengan sistem propulsi pada kapal. Apabila resonansi terjadi saat kapal dalam operasi, maka akan menimbulkan banyak kerusakan pada struktur konstruksi pada kapal. Hal ini disebabkan oleh kurangnya optimalisasi proses desain kapal dalam hal pengaruh getaran terhadap struktur kapal khususnya pada sistem propulsai kapal. Untuk mengetahui kemungkinan terjadinya resonansi, maka besarnya frekuensi natural dari sistem propulsi kapal tersebut harus dihitung dengan teliti.

Oleh karena itu untuk mencegah terjadinya kerusakan tersebut maka perlu adanya pertimbangan mengenai pengaruh getaran pada saat proses desain kapal. Dan juga perlu adanya pemahaman konsep getaran mengenai frekuensi natural dan mode shape perlu dipahami secara mendalam. Dengan diketahuinya frekuensi natural dan mode shapes maka terjadinya resonansi bias dihindari. Dalam tugas akhir ini akan dihasilkan sebuah software aplikasi untuk proses edukasi yang bertujuan untuk mempermudah proses pemahaman mengenai arti dari frekuensi natural dan mode shapes pada konsep getaran.

1.2 Perumusan Masalah

Sehubungan dengan latar belakang tersebut di atas permasalahan yang akan dibahasdalam tugas akhir ini adalah :

1. Apa saja faktor –faktor yang mempengaruhi besarnya frekuensi natural dan mode shapes?
2. Bagaimana cara mempermudah pemahaman frekuensi natural dan mode shape ?

1.3 Batasan Masalah

Agar permasalahan yang dibahas tidak keluar dari permasalahan yang dibahas maka perlu dirumuskan batasan masalah. Batasan masalah dalam penulisan tugas akhir ini adalah sebagai berikut:

1. Lingkup pengerjaan hanya sampai menghitung frekuensi natural dan mode shape
2. Animasi yang dihasilkan sesuai dengan mode shapes yang dihasilkan
3. Pemodelan massa dan inersia massa untuk getaran longitudinal dan torsional mengacu pada sistem propulsi
4. Bahasa pemrograman yang digunakan dalam perhitungan ini adalah *Java*

1.4 Tujuan

Adapun tujuan dari pengerjaan Tugas Akhir ini adalah untuk :

1. Mengidentifikasi faktor-faktor yang dapat mempengaruhi besarnya frekuensi natural dan mode shapes .
2. Membuat suatu program aplikasi yang user friendly dan interaktif untuk mempermudah pemahaman konsep frekuensi natural dan mode shapes.

1.5 Manfaat

Adapun manfaat yang dapat diperoleh adalah :

1. Dari penelitian ini diharapkan akan dapat diaplikasikan sebuah program aplikasi dalam proses edukasi.
2. Mempermudah pemahaman konsep frekuensi natural dan mode shape dengan cara permodelan software.

1.6 Hipotesis Awal

Jika penelitian ini dilakukan akan dihasilkan sebuah program aplikasi yang bermanfaat untuk mempermudah pemahaman konsep frekuensi natural dan mode shape.

1.7 Metodologi

Pengerjaan penelitian tugas akhir ini dilakukan secara sistematis berdasarkan urutan kerja yang dilakukan oleh penulis:

1.7.1 Study Literatur

Study literatur dilakukan untuk mengetahui teori-teori dasar yang menunjang dalam penulisan tugas akhir dan sebagai acuan dalam menyusun semua hipotesa dan kesimpulan yang akan diambil. Studi literatur berfungsi sebagai bekal dan pengetahuan awal untuk menentukan arah pengerjaan tugas akhir ini.

1.7.2 Pemasukan Data Primer

Tugas Akhir ini menggunakan bantuan Software JAVA untuk pembuatan aplikasi yang akan dilakukan. Untuk itu pertama kali yang harus dilakukan adalah membuat rumus dari metode perhitungan yang ada dan membuat langkah-langkah perhitungannya.

1.7.3 Pembuatan Program

Pembuatan program yang dilakukan dengan pembuatan obyek, prosedur, fungsi dan pengkelasan. Obyek dapat dibuat dengan 2 cara yaitu melalui komponen pallete (obyek visual) dan menuliskan langsung pada program (non-visual). Obyek visual berguna sebagai tempat pemasukan data, penampilan hasil proses dan pengaturan interface program. Sedang obyek non visual berguna untuk pemrosesan data dan pengaturan properti yang ditampilkan melalui fungsi dan prosedur yang dimilikinya.

1.7.4 Validasi Model

Untuk menjamin bahwa *software* yang dilakukan sudah benar maka validasi perlu dilakukan. Validasi dilakukan dengan membandingkan hasil dari *software* dengan perhitungan manual yang dilakukan.

1.8 Sistematika Penulisan

Untuk memperoleh hasil laporan tugas akhir yang sistematis dan tidak keluar dari pokok permasalahan maka dibuat sistematika sebagai berikut:

BAB 1. PENDAHULUAN

Bab ini berisi konsep dasar penyusunan tugas akhir yang meliputi latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, sistematika penulisan.

BAB 2. DASAR TEORI DAN TINJAUAN PUSTAKA

Bab ini berisi uraian mengenai landasan-landasan teori yang digunakan untuk menyelesaikan tugas akhir ini. Dasar teori yang digunakan dalam bab ini digunakan untuk lebih memahami konsep dasar getaran longitudinal dan torsional sistem propulsi kapal, frekuensi natural dan mode shape.

BAB 3. METODOLOGI

Bab ini berisi mengenai langkah – langkah dalam pembuatan *software*. Berisi mengenai detail dan pengertian tentang langkah – langkah yang dilakukan dalam pembuatan *software* pengambilan keputusan.

BAB 4. OVERVIEW METODE PENENTUAN FREKUENSI NATURAL DAN MODE SHAPE

Bab ini membahas tentang beberapa metode penentuan frekuensi natural, terutama dipilih yang paling sesuai apabila manipulasi computer akan dilakukan. Hal ini perlu karena bukan rahasia lagi bahw abanyak teori yang elegan masih belum mampu menghasilkan hasil yang bermanfaat untuk perencanaan karena terbatasnya analisa numerik yang bisa digunakan.

BAB 5. PENYELESAIAN NUMERIK DAN IMPLEMENTASI PROGRAM

Dan akhirnya pada bab ini dibahas mengenai metode yang digunakan. Metode yang digunakan adalah anlisa eigenproblem dan transformasi. Pada bab ini juga dilakukan verifikasi terhadap program komputr dengan membandingkannya dengan data yang bisa yang bisa dimanipulasi dengan menggunakan closed form, dan juga bisa dihitung secara manual. Dengan demikian pemilihan pemilihan data hanya terbatas pada jumlah derajat kebebasan yang tidak terlalu besar karena sulitnya manipulasi dengan tangan apabila hal itu harus dilakukan.

BAB 6. KESIMPULAN DAN SARAN

Berisikan kesimpulan dari hasil analisadan saran-saran untuk pengembangan lebih lanjut yang berkaitan dengan materi yang terdapat dalam tugas akhir ini.

DAFTAR PUSTAKA

LAMPIRAN

BAB 2
TINJAUAN PUSTAKA

2.1 Getaran

Getaran adalah peristiwa gerak bolak-balik sebuah benda terhadap suatu titik kesetimbangan. Getaran kapal merupakan salah satu bagian dari keseluruhan masalah yang tercakup dalam dinamika kapal (ship dynamics).

Secara umum ada 2 kelompok getaran yaitu getaran bebas dan getaran paksa. Getaran bebas terjadi jika sistem berosilasi karena bekerjanya gaya yang ada dalam sistem itu sendiri (inherent) dan jika tidak ada gaya luar yang bekerja. Sistem yang bergetar bebas akan bergetar pada satu atau lebih frekuensi naturalnya yang merupakan sifat sistem dinamika yang dibentuk oleh distribusi massa dan kekakuannya.

Sedangkan getaran paksa adalah getaran yang terjadi karena rangsangan gaya luar artinya rangsangan dari luar berisolasi dengan system sehingga sistem dipaksa untuk bergetar pada frekuensi rangsangan. Bila sebuah system dipengaruhi oleh eksitasi harmonik paksa, maka respon getarannya akan berlangsung pada frekuensi yang sama dengan frekuensi eksitasinya.

2.1.1 Konsep Dasar

Suatu sistem agar dapat bergetar harus mempunyai tiga elemen pokok, yaitu input, system dan output. Input merupaka gaya luar yang bekerja dan dikenakan pada sistem, yang dengan sendirinya harus berupa gaya dinamis yaitu merupaka fungsi waktu. Secara garis besar input ini bisa dibedakan menjadi beberapa golongan, sebagai berikut,

Input Harmonik

Ini adalahjenis input yang paling sederhana sepanjang menyangkut masalah getaran. Semua input yang bisa ditulis dalam bentuk Fcos(wt) atau Fsin(wt) dikategorikan sebagai input harmonik (F merupakan amplitude gaya eksitasi dan w adalah frekuensi eksitasi).

Input Periodik

Input harmonik adalah input periodik yang paling sederhana. Namun demikian tidak setiap fungsi periodik harus berupa fungsi harmonik. Input jenis ini bisa berupa smbarang input, satu-satunya batasan adalah bahwa input tersebut harus periodik untuk waktu t yang tak terhingga. Input jenis ini tidak terlalu sulit ditangani. Secara konsepsi, yaitu dengan merubahnya menjadi penjumlahan bentuk harmonik. Teknik ini dikenal dengan nama deret Fourier.

Input Sembarang

Input ini benar-benar sembarang, yaitu tidak harus mengikuti pattern tertentu. Input jenis ini sering disebut statistical input, karena orang harus menggunakan analisa statistk untuk menanganinya.

Input yang merupakan gaya interaktif

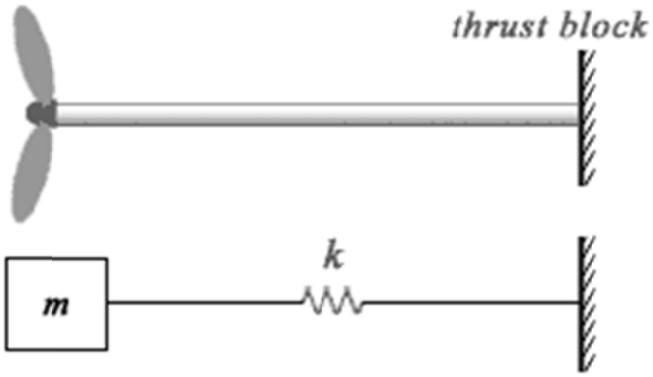
Sejauh ini, input jenis ini yang terjelek karena inout itu sendiri akan sangat bergantung dari getaran yang dihasilkan. Padahal getarannya sendiri sangat bergantung pada input.

Dalam kaitannya dengan getaran yang terjadi pada sistem propulsi kapal, input yang bisa terjadi adalah input karean bekerjanya mesin, input karena bekerjanya propeller dismping input gelombang. Namun demikian input terakhir ini lebih bersifat input sembarang dan akibatnya bukan hanya pada getaran kapal. input yang paling dominan adalah karena bekerjanya propeller.

2.1.2 Getaran Longitudinal

Getaran longitudinal pada sistem propulsi kapal merupakan salah satu getaran dengan koordinat gerak sejajar dengan sumbu poros propeller. Getaran ini timbul akibat putaran propeller serta adanya gaya radial yang ditimbulkan main engine [Priatmoko 2003].

Gaya aksial propeller (thrust) ditahan oleh thrust block yang kemudian diteruskan ke konstruksi kapal. Karena gaya aksial ini, maka thrust block dan pondasinya akan mengalami pergeseran secara longitudinal. Untuk analisa getaran longitudinal, sistem propulsi kapal dapat dimodelkan sebagaisuatu sistem pegas massa seperti gambar dibawah.



Gambar 2.1 getaran longitudinal sistem propulsi kapal

Parameter sistem getaran longitudinal adalah massa (m), kekekuan pegas (k), dan peredam (c). gabungan massa pegas, pegas, dan peredam menggambarkan sifat fisik dari suatu sistem. Energy yang akan diterima oleh sistem melalui gaya luar yang dikenakan, atau melalui gangguan yang merubah system dari posisi keseimbangannya.

Dalam menentukan besarnya massa suatu benda menggunakan persamaan sebagai berikut

$$m = \frac{W}{g} \qquad \text{II.(1)}$$

dengan W adalah berat benda

g adalah percepatan gravitasi

Untuk menentukan besarnya kekakuan pegas menggunakan persamaan sebagai berikut [GOODIERE, 1894]

$$k = \frac{A\,E}{l} \qquad \text{II.(2)}$$

dengan A adalah luas penampang

E adalah modulus young

l adalah panjang

Besarnya output ditentukan dengan menggunakan referensi terhadap massa ini diukur dari posisi keseimbangannya. Hal ini akan sangat kelihatan pada sistem diskrit, yaitu sistem dengan n massa.

Pegas bisa diasumsikan mempunyai massa maupun tidak mempunyai massa. State of the art analisa getaran adalah menganggap bahwa pegas ini tidak memiliki massa. Besarnya gaya pegas sebanding dengan displasemen yang dikenakan pada pegas, yang bisa berbanding linear maupun non linear.

Peredam selalu diasumsikan tidak mempunyai massa. Besarnya gaya peredam sebanding dengan kecepatan yang dikenakan pada peredam tersebut, yang bisa linear maupun non linear.

2.1.3 Getaran Torsional

Getaran torsi adalah getaran sudut periodik poros elastis dengan rotor bulat yang dikaitkan kepadanya. Pada sistem propulsi getaran yang terjadi diakibatkan bekerjanya eksitasi torsi (momen) [Priatmoko 2003].

Pada propeller akan bekerja enam komponen gaya/momen osilasi (unsteady force/moment) yaitu tiga komponen gaya dan tiga komponen momen. Gaya dan momen tersebut terjadi karena propeller berputar pada daerah wake yang tidak uniform.

Parameter sistem getaran longitudinal adalah momen inersia massa (J), kekekuan pegas torsional (K), dan peredam (C). gabungan massa, pegas, dan peredam menggambarkan sifat fisik dari suatu sistem. Energy yang akan diterima oleh sistem melalui gaya luar yang dikenakan, atau melalui gangguan yang merubah system dari posisi keseimbangannya.

Dalam menentukan besarnya massa suatu benda menggunakan persamaan sebagai berikut

$$J = \rho \, J_A \, L \qquad \text{II.3}$$

dengan ρ adalah kerapatan benda

J_A adalah momen inersia penampang

L adalah panjang

Untuk menentukan besarnya kekakuan pegas torsional menggunakan persamaan sebagai berikut [GOODIERE, 1894]

$$k = \frac{J_A\, G}{L} \qquad \text{II.4}$$

dengan J_A adalah momen inersia penampang

E adalah modulus young

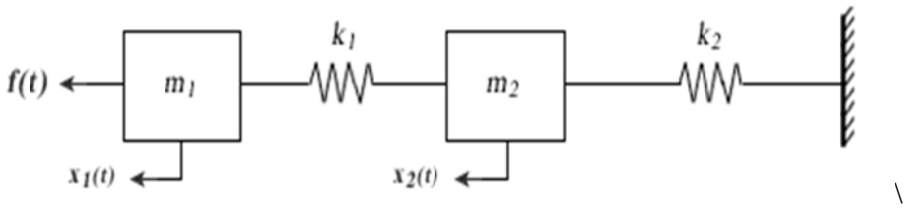
L adalah panjang

Dapat dilihat bahwa dalam system getaran longitudinal dan getran torsional terdapat hal yang identik. Massa pada getaran longitudinal memiliki sifat yang sama pada momen inersia massa pada getaran torsional. Kekakuan pegas pada getaran longitudinal dan torsional memiliki sifat yang sama

Satu hal yang perlu dicatat adalah bahwa pemisahan seperti kekakuan pegas dan redaman yang diasumsikan tidak memiliki massa dalam prakteknya tidak semudah itu. Semakin kompleks suatu sistem makin sulit menentukan parameter-parameter tersebut secara pasti. Bahkan tidak berlebihan bila dikatakan bahwa sebagian besar masalah getaran telah terselesaikan kalau besarnya parameter tersebut bisa ditentukan dengan akurat.

2.1.4 Formulasi matriks

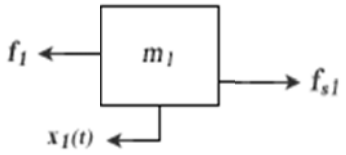
Untuk sistem dengan banyak derajat kebebasan maka cara paling baik untuk menanganinya adalah dengan menulis persamaan geraknya dengan bentuk matriks. Kita ambil contoh sistem getaran longitudinal dengan dua derajat kebebasan.



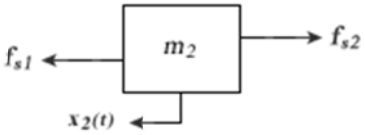
Gambar 2.2 Getaran Longitudinal dengan dua derajat kebebasan

Derajat kebebasan didefinisikan sebagai jumlah koordinat independen yang diperlukan untuk mendeskripsikan gerakan suatu sistem. Dengan demikian sistem kontinue mempuyai jumlah derajat kebebasan yang tidak terhingga.

Secara skematis persamaan gerak dari sistem getaran longitudinal dengan dua derajat kebebasan dapat di ekspresikan sebagai berikut,



Gambar 2.3 freebody diagram massa 1



Gambar 2.4 *freebody diagram massa 2*

$\sum F = m_1 \ddot{x}_1$ $\qquad\qquad$ $\sum F = m_2 \ddot{x}_2$

$f_1 - f_{s1} = m_1 \ddot{x}_1$ $\qquad\qquad$ $f_{s1} - f_{s2} = m_2 \ddot{x}_2$

Eliminasi

$f_{s1} = k_1 (x_1 - x_2)$ II.5

$f_{s2} = k_2 x_2$ II.6

Subtitusi

$f_1 - k_1 (x_1 - x_2) = m_1 \ddot{x}_1$ II.7

$k_1 (x_1 - x_2) - k_2 x_2 = m_2 \ddot{x}_2$ II.8

Dari persamaan-persamaan diatas agar mudah dalam mengerjakan lebih baik ditransformasikan kedalam bentuk matriks,

$$[m]\{\ddot{x}\} + [k]\{x\} = \{f(t)\} \quad \text{II.9}$$

Dengan $[m]$ sebagai matriks massa

$[k]$ sebagai matriks kekakuan pegas

$\{x\}$ sebagai vector output displasemen, serta

$\{f(t)\}$ sebagai vector eksitasi dinamis

2.1.5 Frekuensi Natural Dan Mode Shape

Apabila sistem diganggu dari keseimbangannya maka sistem akan bergetar dengan suatu frekuensi tertentu. Frekuensi ini, yaitu frekuensi getaran dari sistem yang bergetar tanpa adanya gaya luar (free vibration), frekuensi natural. Kalau lebih jauh lagi dianggap bahwa perbandingan antara konstanta peredam dan massa sistem cukup kecil, maka frekuensi natural bisa didefinisikan sebagai frekuensi sistem yang bergetar tanpa eksitasi dan tanpa damping

Sistem dengan satu derajat kebebasan dengan demikian hanya akan mempunyai satu frekuensi natural dan satu konfigurasi displasemen saja. Untuk frekuensi natural sistem dengan satu derajat kebebasan dapat dengan mudah ditentukan, yaitu,

$$\omega = \sqrt{\frac{k}{m}} \qquad \text{II.10}$$

untuk getaran longitudinal dan untuk getaran torsional, yaitu

$$\omega = \sqrt{\frac{K}{\mathrm{J}}} \qquad \text{II.11}$$

Untuk sistem dengan banyak derajat kebebasan frekuensi natural ini tidak bisa ditentukan dengan semudah itu, namun ada teknik-teknik tertentu yang bisa digunakan dalam memecahkan permasalahan tersebut. Beberapa teknik tersebut akan dibahas pada Bab IV.

Disamping itu, ada konsep baru yang tidak muncul pada sistem dengan hanya satu derajat kebebasan. Pada sistem dengan banyak derajat kebebasan akan terjadi konfigurasi displasemen yang berbeda apabila sistem bergetar (tanpa eksitasi tanpa damping) pada frekuensi natural yang berbeda. Konfigurasi demikian yang tidak memperhatikan harga mutlak dari displasemen, disebut mode shape. Jadi konfigurasi pada mode shape hanyalah menunjukkan perbandingan dan arah antara displasemen satu dengan displasemen lainnya.

Secara matematis ekspresi untuk untuk menentukan frekuensi natural adalah sebagai berikut. Apabila dimisalkan,

$$x_t = Re\ A\ e^{iwt} \qquad \text{II.12}$$
$$\dot{x}_t = Re\ iwA\ e^{iwt}$$

$$\ddot{x}_t = Re - w^2 A\, e^{iwt}$$

sehingga persamaan gerak dari getaran dapat disubtitusi menjadi

$$(-\omega^2[m] + [k])\{A\} = 0 \qquad \text{II.13}$$

atau

$$[D(\omega_n)]\{A\} = 0 \qquad \text{II.14}$$

sehingga ω dihitung dari

$$\det[D(\omega_n)] = 0 \qquad \text{II.15}$$

ekspresi ini didasarkan pada anggapan yang telah disebutkan diatas dan juga bahwa system bergetar sesuai dengan fungsi harmonik.

Dari persamaan diatas terlihat bahwa besarnya $\{A\}$ tidak bisa ditentukan. Namun demikian mode shape bisa ditentukan. Karena $\det[D(w_n)] = 0$ pada waktu w=wj (wj adalah frekuensi natural yang ke-j) maka persamaan diatas disebut linearly dependent, yaitu bahwa kita tidak bisa mendapatkan semua harga $\{A\}$. Dalam hal ini hanya terdapat N-1 persamaan yang independen untuk setiap N frekuensi natural. Kalau persamaan ke N di hilangkan, maka mode shape didapatkan dari ekspresi berikut,

$$\begin{bmatrix} D_{11} & \cdots & D_{1,N-1} \\ \vdots & \ddots & \vdots \\ D_{N-1,1} & \cdots & D_{N-1,N-1} \end{bmatrix} \begin{Bmatrix} A_1 \\ : \\ A_{N-1} \end{Bmatrix} = \begin{Bmatrix} 0 \\ : \\ 0 \end{Bmatrix} \qquad \text{II.16}$$

sehingga mode shape secara simbolis ditulis,

$$\{A\} = \begin{Bmatrix} A_{1j} \\ A_{2j} \\ : \\ A_{N-1,j} \end{Bmatrix} \qquad \text{II.17}$$

Persamaan diatas secara teoritis bisa digunakan untuk menentukan besarnya frekuensi natural dan mode shape. Namun demikian cara ini sangat tidak efisien dan bahkan untuk ukuran matriks yang besar hamper tidak mungkin dilakukan dengan manual. Pada Bab IV akan membahas mengenai beberapa cara yang sesuai untuk memanipulasi dengan computer.

2.2 Pendekatan Parameter Getaran Sistem Propulsi Kapal

Sebagaimana dimaklumi, langkah pertama dalam menganalisa suatu system adalah mendekati system tersebut dengan model matematis. Tepat atau tidaknya suatu analisa sangat tergantung dari model yang digunakan, yaitu apakah model dimaksud telah mewakili sifat yang esensial dari sistem yang dianalisa.

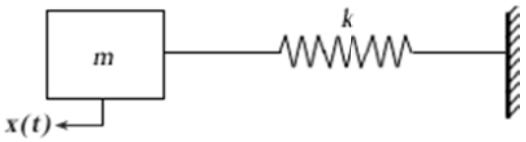
Dalam hal getaran sistem propulsi kapal kendati persamaan matematisnya telah dikenal dengan baik. Namun tanpa harga parameter-parameter yang mendekati kenyataan maka analisa yang dihasilkan mungkin tidak mencerminkan keadaan sebenarnya, tidak perduli bagaimana canggihnya model maupun cara analisa yang digunakan. Karena itu gambaran tentang penentuan parameter menjadi sesuatu yang sangat penting. Berikut adalah beberapa usulan penentuan penentuan beberapa parameter, masing-masing untuk sistem propulsi dengan poros pendek dan sistem dengan poros panjang.

2.2.1 Sistem dengan Poros Pendek

Tidak ada ketentuan yang pasti kapan suatu sistem dianggap mempunyai poros pendek dan kapan dianggap mempunyai poros panjang. Penentuan ini semata-mata merupakan “judgement” individu yang melakukan analisa. Namun demikian ada petunjuk yang berlaku universal untuk menentukan apakah suatu sistem dianggap mempunyai porors pendek atau panjang, meski disinipun “individual judgement” masih memerankan peranan yang sangat penting.

Seperti diketahui bahwa getaran longitudinal terjadi karena gaya aksial yang berfluktuasi akibat putaran propeller yang ditahan oleh thrust bearing dan kemudian diteruskan ke konstruksi kapal. karena gaya aksial ini maka thrust bearing beserta pondasinya akan mengalami pergeseran longitudinal (atau aksial). Getaran torsi adalah getaran sudut periodik poros elastis dengan rotor bulat yang dikaitkan kepadanya. Pada sistem propulsi getaran yang terjadi diakibatkan bekerjanya eksitasi torsi (momen). Apabila cukup beralasan untuk menganggap bahwa tidak terjadi gerakan relatif antara mesin sampai propeller, maka sistem dapat dikategorikan sebagai sistem dengan poros pendek. Dalam hal demikian maka sistem akan bergetar pada posisi thrust bearing.

Asumsi ini beranggapan bahwa seluruh sistem, dari propeller sampai mesin, merupakan satu kesatuan benda tegar yang bergetar dengan displasemen sama. Secara skematis dapat digambarkan sebagai berikut

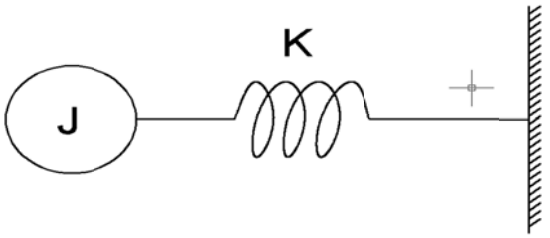


Gambar 2.5 Sistem dengan poros pendek getaran longitudinal

Dimana,

m adalah massa propeller ditambah dengan massa poros dan sebagian massa pondasi

k adalah kekakuan pegas sistem secara keseluruhan



Gambar 2.6 Sistem dengan poros pendek getaran torsional

Dimana,

J adalah inersia massa propeller ditambah inersia massa poros ditambah inersia massa flywheel dan inersia massa dari silinder-silinder mesin induk.

K adalah kekakuan pegas dari system secara keseluruhan

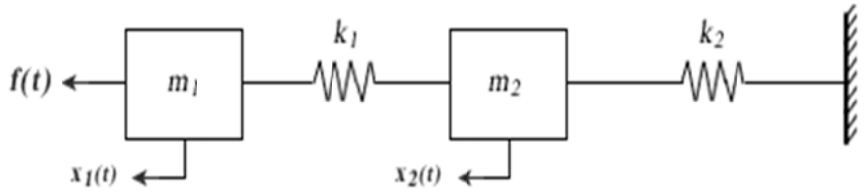
Karena sistem dengan poros pendek ini bisa dianalisa sebagai sistem dengan satu derajat kebebasan, maka praktis tidak ada masalah dalam menentukan besarnya frekuensi natural dan mode shape.

2.2.2 Sistem dengan Poros Panjang

Kalau asumsi bahwa sistem merupakan satu kesatuan benda tegar tak bisa dipertahankan lagi, maka secara otomatis analisa diatas tidak bisa digunakan. Sistem dengan demikian harus didekati dengan banyak massa. Berapa jumlah massa yang digunakan sekali lagi tidak bisa ditentukan dengan eksak, karena ini sangat tergantung dari kondisi sistem itu sendiri. Semakin

banyak massa yang digunakan berarti akan semakin akurat hasil yang didapatkan, namun dipihak lain juga semakin mahal biaya yang harus dikeluarkan.

Contoh untuk sistem dengan poros panjang atau sistem dengan banyak derajat kebebasan adalah sebagai berikut



Gambar 2.7 Sistem dengan poros panjang getaran longitudinal

Gambar diatas merupakan model getaran longitudinal dengan dua massa. Parameter-parameter pada model ini adalah seperti dijelaskan berikut ini.

m_1 adalah massa propeller yang meliputi:

- Massa dari propeller sendiri
- Propeller added hydrodynamic mass (tambahan massa pada propeller karena pengaruh hirodinamis). Besarnya tambahan massa ini masih memerlukan penelitian lebih lanjut, kendati beberapa penelitian sudah dilakukan yang ada kaitannya dengan added mass ini. Yang biasanya digunakan adalah sekitar 60% dari massa propeller [HARRINGTON, 1971].
- Sebagian massa poros, yaitu 1/3 bagian dari seluruh massa poros. Hal ini sebagai konsekuensi anggapan bahwa poros dianggap tidak mempunyai massa. Angka diatas didapat berdasarkan analisa bahwa frekuensi natural kearah longitudinal antara poros dengan massa uniform tidak berbeda dengan frekuensi natural poros tanpa berat kalau 1/3 dari total massanya diletakkan di ujung [THOMSON, 1976].

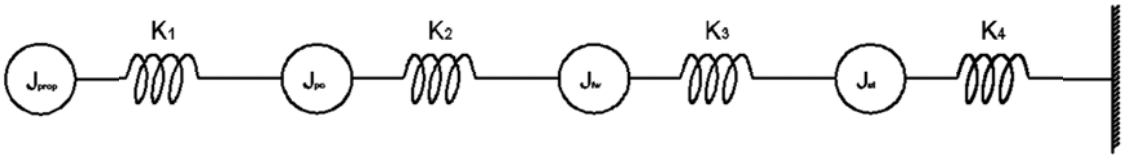
m_2 adalah massa permesinan yang terdiri dari:

- Massa mesin induknya sendiri (termasuk massa reduction gear)
- Sisa dari massa poros, yaitu 2/3 dari total massa karena 1/3 bagiannya telah diikutkan pada massa m_1.

k_1 adalah kekakuan pegas dari poros utama

k_2 adalah kekakuan pegas thrust bearing, meliputi kekakuan thrust collar, thrust bearing housing dan thrust foundation.

Apabila kita ingin mendekati model sistem propulsi dengan tiga massa maka akan memberikan hasil yang lebih akurat dibandingkan dengan pendekatan dua massa. Pendekatan semacam ini diterapkan pada kasus dimana antara reduction gear dan mesin induk terdapat gerakan relatif, yang sering terjadi apabila penggerak utamanya adalah turbin. Dalam kasus ini maka massa reduction gear (yang dalam hal ini cukup besar) harus dipisahkan tersendiri. Sehingga terdapat 3 massa yaitu massa propeller dan 1/3 poros, massa reduction gear, dan massa permesinan. Dan apabila kita ingin mendekati model dengan banyak massa, maka hasil yang didapatkan akan semakin maksimal. Dan penentuan jumlah massa yang digunakan dan komponen-komponen dari tiap massa ditentukan berdasarkan judgement dari seorang engineer.



Gambar 2.8 Sistem dengan poros panjang getaran torsional

Gambar diatas merupakan model getaran torsional pada system propulsi kapal. untuk getaran torsional kali ini yang menjadi parameter derajat kebebasannya adalah jumlah silinder mesin diesel yang digunakan sebagai main engine kapal. sehingga variasi derajat kebebasan ditentukan dari jumlah silinder mesin. Parameter-parameter pada model ini adalah seperti dijelaskan berikut ini

J_{prop} adalah inersia massa propeller yang meliputi:

- Inersia massa dari propeller sendiri, dan lazimnya terdapat tambahan 25% dari inersia massa propeller (tambahan massa air yang seolah-olah ikut berputar)

- Sebagian inersia massa poros, yaitu 1/3 bagian dari seluruh massa poros. Hal ini sebagai konsekuensi anggapan bahwa poros dianggap tidak mempunyai massa. Angka diatas didapat berdasarkan analisa bahwa frekuensi natural kearah longitudinal antara poros dengan massa uniform tidak berbeda dengan frekuensi natural poros tanpa berat kalau 1/3 dari total massanya diletakkan di ujung [THOMSON, 1976].

J_{poros} adalah inersia massa poros propeller yang terdiri dari:

- Sisa dari inersia massa poros, yaitu 2/3 dari total inersia massa karena 1/3 bagiannya telah diikutkan pada massa J_{Prop}. Dan dengan anggapan bahwa poros adalah uniform tidak ada perubahan diameter dari sisi ke sisi yang lain.

$J_{flyweel}$ adalah inersia massa flywheel yang terdiri dari:

- Merupakan inersia massa dari flywheel (roda gigi)

$J_{silinder}$ adalah inersia massa silinder mesin yang terdiri dari:

- Merupakan inersia massa dari silinder-silinder mesin induk sendiri
- Sebagian inersia massa dari poros engkol

K_1dan K_2 adalah kekakuan pegas dari poros utama.

K_3 adalah kekakuan pegas dari thrust bearing, meliputi kekakuan thrust collar, thrust bearing housing dan thrust foundation.

K_4dan K_n adalah kekakuan pegas dari poros engkol mesin induk.

2.3 JAVA

2.3.1 Pengertian

Java pada dasarnya adalah sebuah bahasa pemrograman komputer. Bahasa pemrograman adalah perintah-perintah atau instruksi yang dimengerti oleh komputer untuk melakukan tugas-tugas tertentu. Java selain disebut sebagai sebuah bahasa pemrograman, juga sering disebut sebagai sarana (tool) untuk menghasilkan program-program aplikasi berbasiskan Windows [LYRACC, 2009]. Beberapa kemampuan atau manfaat dari Java, yaitu:

1. Untuk membuat program aplikasi berbasis Windows.
2. Untuk membuat objek-objek pembantu program seperti misalnya control ActiveX, file Help, aplikasi Internet.
3. Menguji program dan menghasilkan program berakhiran EXE yang bersifat executable atau dapat langsung dijalankan.

2.3.2 Sejarah JAVA

Java dipelopori oleh James Gosling, Patrick Naughton, Chris Warth, Ed Frank, dan Mike Sheridan dari Sun Microsystems, Inc pada tahun 1991. Mereka membutuhkan kurang lebih 18 bulan untuk membuat versi pertamanya. Bahasa ini pada awalnya disebut “Oak” tapi kemudian diubah menjadi “Java” pada tahun 1995 karena nama Oak telah dijadikan hak cipta dan digunakan sebagai bahasa pemrograman lainnya. Antara pembuatan Oak pada musim gugur 1992 hingga diumumkan ke publik pada musim semi 1995, banyak orang yang terlibat dalam desain dan evolusi bahasa ini. Bill Joy, Arthur van Hoff, Jonathan Payne, Frank Yellin, dan Tim Lindholm merupakan kontributor kunci yang mematangkan prototipe aslinya.

2.3.3 Kelebihan JAVA

Bahasa pemrograman lain yang telah ada sebelum Java lahir sudah merupakan bahasa yang baik dan mudah dipelajasi oleh programmer profesional. Akan tetapi para programmer ini menginginkan sesuatu yang baru yang memiliki banyak hal yang menyelesaikan masalah mereka. Utamanya adalah keamanan kode mereka. Hal ini melahirkan pikiran yang revolusioner untuk menemukan bahasa pemrograman lain yang disebut Java. Tidak hanya keamanan tapi juga beberapa hal yang sering disebut sebagai Java-Buzzwords. Kata-kata ini menjelaskan berbagai fitur tambahan dan beberapa hal yang membuat Java demikian sukses

dan diterima oleh dunia perangkat lunak. Berikut ini adalah penjelasan serta keuntungan dari kata-kata tersebut.

2.3.4 Sederhana Dan Berorientasi Objek

Seperti diuraikan sebelumnya, Java lahir dari suatu pemikiran mendalam akan bahasa pemrograman yang ada pada saat itu, seperti C dan C++. Hal ini akan memudahkan programmer profesional untuk dapat mengerti lebih jelas tentang Java, fungsionalitas, dan lain sebagainya apabila ia memiliki pengetahuan dasar tentang C++ dan konsep pemrograman berorientasi objek. Tujuannya agar konsep dasar dari teknologi Java dapat dimengerti dengan mudah, dan programmer dapat segera menghasilkan sesuatu sedini mungkin. Tidak hanya ini, penemu Java memastikan bahwa Java juga bermula dari bahasa pemrograman dasar yang sudah ada pada saat itu. Kemudian mereka membuang berbagai fitur yang rumit dan membingungkan.

Bahasa pemrograman Java didesain sejak awal untuk menjadi bahasa yang berorientasi objek. Setelah kira-kira 30 tahun, akhirnya teknologi objek menjadi kenyataan dan diterima oleh sebagian besar komunitas pemrograman. Konsep berorientasi objek memungkinkan pembuatan software yang kompleks, berbasis network, sehingga dapat disimpulkan bahwa teknologi Java menghasilkan platform pembuatan perangkat lunak yang baik dan efisien serta berorientasi objek.

2.3.5 Keuntungan Yang Anda Dapat Dari JAVA

- **Mulai dengan cepat**: Java merupakan bahasa pemrograman berorientasi objek, mudah dipelajari, terutama untuk programmer yang sudah menguasai C atau C++
- **Tulis lebih sedikit program**: Jumlah kelas, jumlah metode, dll, menunjukkan bahwa program yang ditulis dalam bahasa pemrograman Java memiliki jumlah 4 kali lipat lebih kecil dari program sama yang ditulis dalam bahasa C++
- **Tulis program lebih baik**: Bahasa pemrograman Java menganjurkan praktek membuat program yang baik, dan automatic garbage collection membantu Anda untuk menghindari kebocoran memori. Orientasi objeknya, arsitektur komponen JavaBeans, dan jangkauannya yanga luas, API yang mudah diperluas, memungkinkan Anda menggunakan kode yang ada.

- **Membuat program dengan lebih cepat**: Bahasa pemrograman Java lebih mudah dari C++, pemrograman akan menjadi 2 kali lipat lebih cepat, dengan jumlah baris yang jauh lebih sedikit.
- **Menghindari kebergantungan pada platform tertentu**: Anda dapat menjalankan program Anda pada banyak platform dengan TIDAK menggunakan library yang ditulis spesifik untuk platform tertentu.
- **Tulis sekali, jalankan di mana saja**: Karena aplikasi yang ditulis dalam bahasa Java dikompilasi ke dalam kode byte yang bebas platform, aplikasi yang ditulis dapat jalan secara konsisten pada platform apa saja.
- **Distribusikan software Anda dengan mudah**: Dengan Java Web Start, pengguna program Anda akan dapat menggunakan aplikasi Anda dengan mudah. Sistem pengecekan versi otomatis pada saat program dimulai menjamin pengguna Anda selalu menjalankan versi terkini. Apabila versi baru tersedia, Java Web Start akan melakukan instalasi secara otomatis.

BAB 3
METODOLOGI

Metodologi yang dilakukan dalam penelitian ini meliputi identifikasi masalah, studi literatur, pengumpulan data, membuat langkah-langkah pengerjaan, tahap perancangan program, validasi program, penyempurnaan interface dan pembuatan laporan.

3.1 Identifikasi Masalah

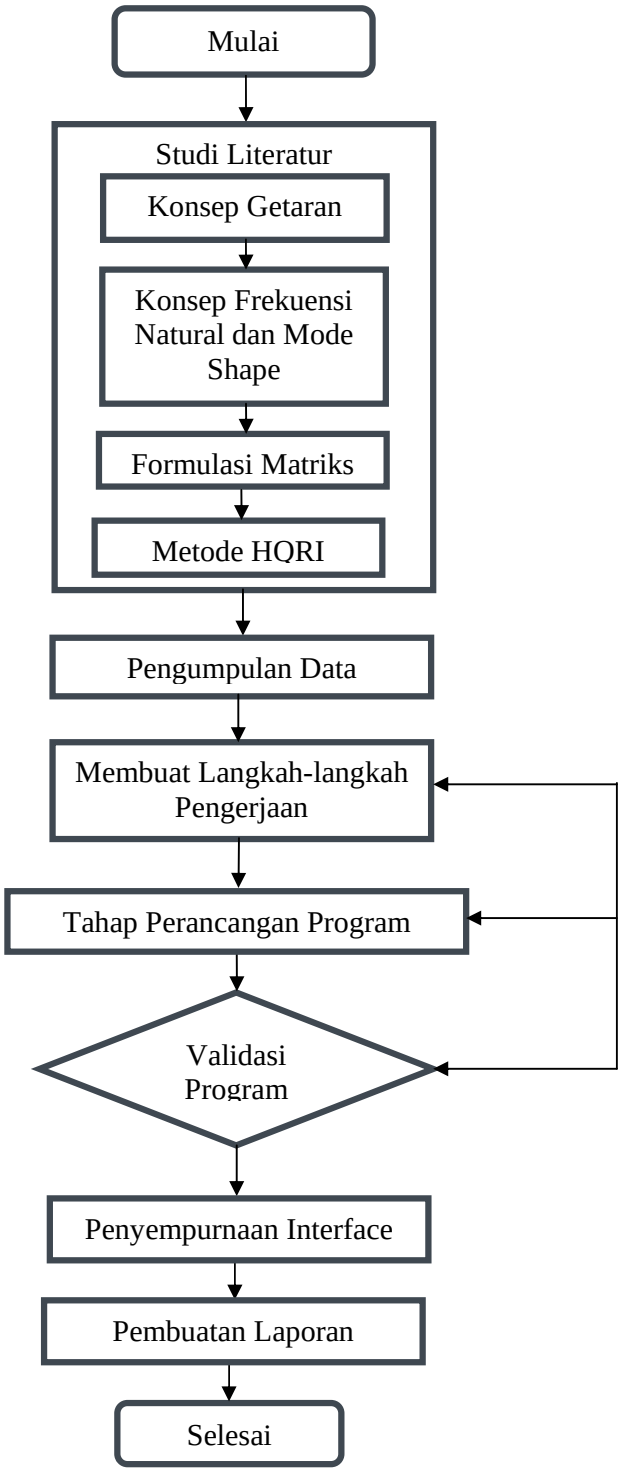
Identifikasi masalah merupakan tahapan awal dari penelitian yang dilakukan dan menjelaskan bidang masalah yang dihadapi serta memberikan asumsi dan batasan – batasan terhadap permasalahan yang ada, selain itu identifikasi masalah juga memberikan gambaran mengenai hal – hal yang diperlukan dalam menjelaskan permasalahan yang ada.

3.2 Studi literatur

Dilakukan untuk mengumpulkan referensi – referensi yang relevan dengan teori yang digunakan dalam penelitian dan dapat mendukung analisis perancangan sistem.

3.3 Pengumpulan data

Pengumpulan data mengenai perhitungan massa, kekakuan pegas, momen inersia massa dan kekakuan pegas torsional yang akan digunakan sebagai variabel dalam perhitungan frekuensi natural dan mode shape getaran. Proses perhitungan yang digunakan adalah menggunakan metode Householder-QR-Iteration. Pengumpulan data meliputi semua perhitungan householder dan iterasi QR dalam pemecahan eigenproblem.



Gambar 3.1 Flowchart Metode Pelaksanaan

3.4 Membuat Langkah-Langkah Pengerjaan

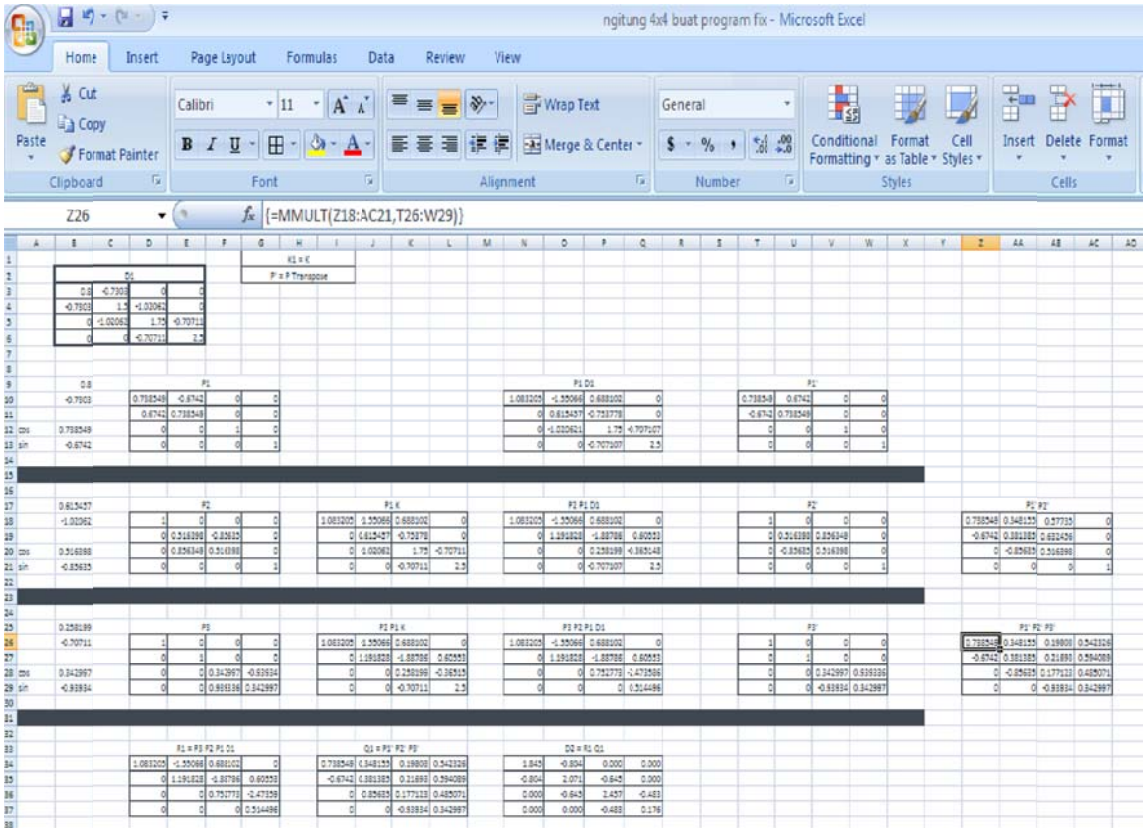
Pembuatan langkah-langkah pengerjaan ini dilakukan dengan tujuan untuk mempermudah dalam pelaksanaan perancangan sistem agar dapat merancang sistem dengan sistematis. Dan apabila terdapat kesalahan dalam hasil saat perencanaan sistem dapat dengan mudah diketahui letak kesalahannya.

3.5 Tahap Perancangan Program

Perancangan sistem berusaha untuk membangun sistem pendukung keputusan sesuai dengan kebutuhan sistem yang telah dianalisis, sehingga diperoleh sistem usulan baru yang lebih baik.

3.5.1 Pembuatan Kerangka Awal Sistem

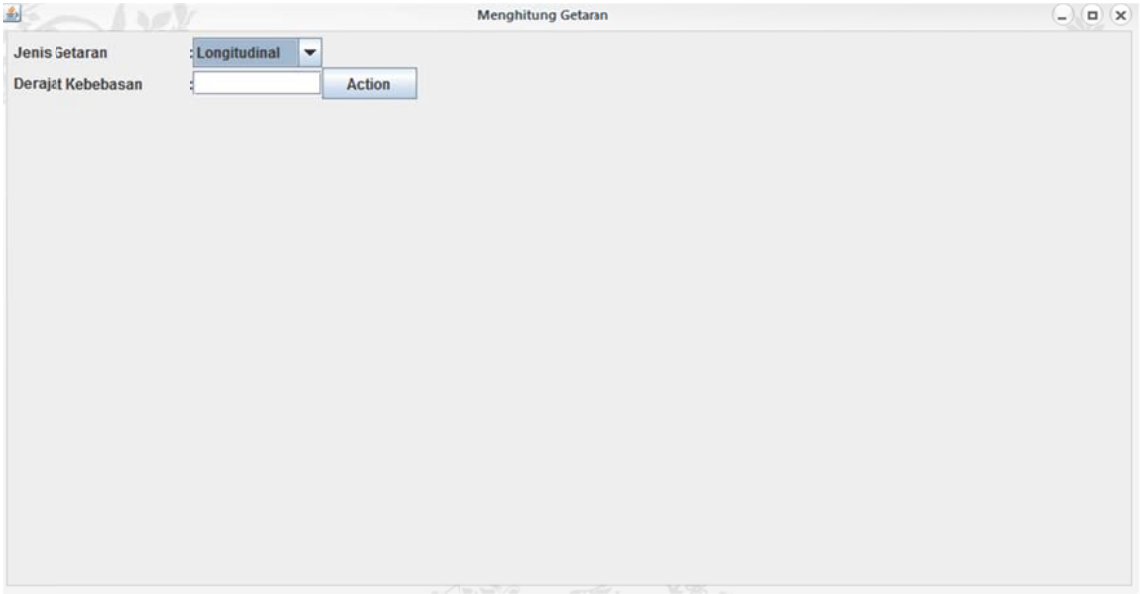
Setelah dipahami kebutuhan sistem yang ingin dibangun kemudian dilakukan pembuatan kerangka awal sistem pendukung keputusan yang berisikan serangkaian langkah awal dalam penyusunan sistem pengambilan keputusan, kerangka awal ini bertujuan untuk mengarahkan dalam pembuatan sistem pengambilan keputusan sehingga dapat mengarah pada tujuan yang ingin dicapai. Dalam pembuatan *software* ini kerangka awal sistem melalui program Microsoft excel, kerangka awal dibuat untuk memudahkan *desainer* sebagai tuntunan dan acuan dalam perhitungan dan pembuatan bahasa pemrograman yang akan dimasukkan kedalam *field logic* yang ada di dalam aplikasi JAVA. Hasil dari pembuatan kerangka awal sistem dapat dilihat pada gambar berikut.



Gambar 3.2 Kerangka awal sistem pada *Microsoft excel*

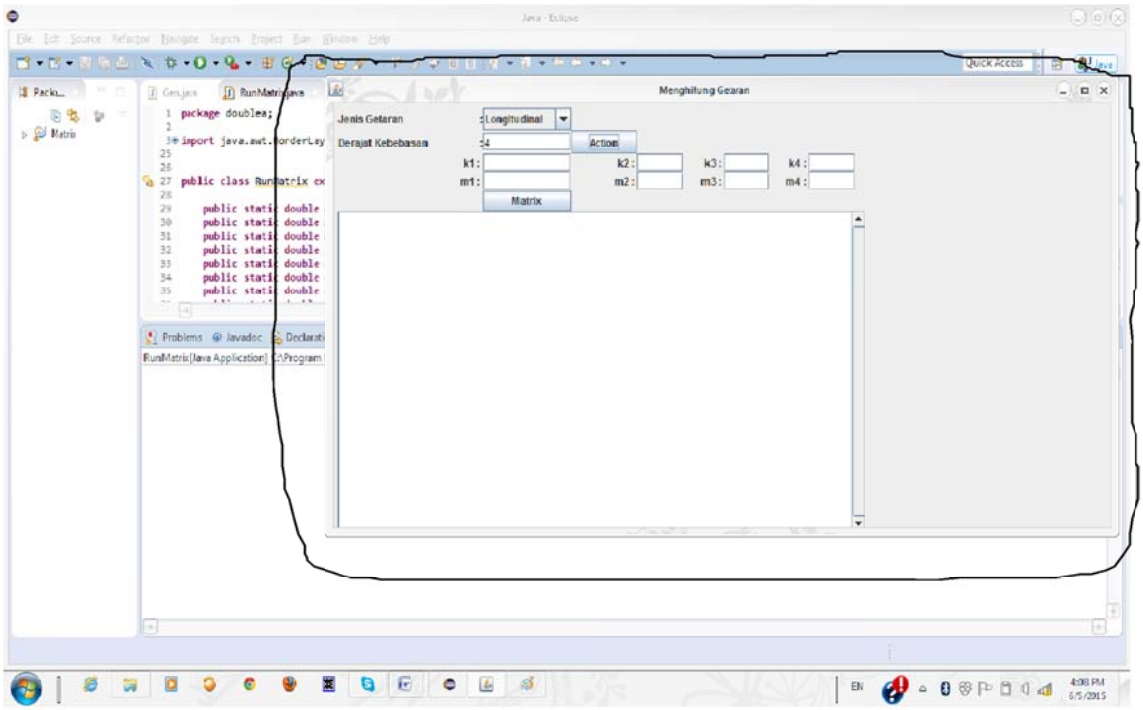
3.5.2 Perancangan Detail

Perancangan detail menjelaskan spesifikasi secara terperinci sistem pengambilan keputusan yang akan dibuat terdiri atas sistem basis data dan sistem basis model. Proses awal dalam tahap perancangan detail adalah pembuatan nama*logic* pada setiap *textbox* dan *buttons* yang tersedia. Penamaan bertujuan untuk memudahkan dalam memasukkan bahasa pemrograman yang akan dibuat, sehingga dengan begitu pembuatan *software* menjadi lebih tersusun dan terarah dalam bentuk aplikasi JAVA. Proses pertama dilakukan dengan membuat *userform*. Contoh userform dapat dilihat pada gambar berikut.



Gambar 3.3 Contoh *userform* awal

Userform awal akan diberi *textbox* dan*label* untuk memudahkan informasi bagi pengguna dan menjadi *field* dalam memasukkan bahasa pemrograman. Proses ini dapat dilihat pada gambar berikut.



Gambar 3.4 Pembuatan *textbox* dan *label* pada *userform*

3.6 Validasi Program

Langkah validasi dan reabilitas model berusaha membandingkan apakah sistem baru yang diusulkan sesuai dengan sistem nyata yang akan diwakili atau tidak, jika sistem yang dibentuk tidak sesuai dengan keadaan nyata yang diwakili maka dilakukan perancangan ulang sistem sehingga didapat sistem yang mampu mewakili kondisi sistem nyata yang ingin digambarkan.

3.7 Penyempurnaan Interface

Penyempurnaan interface ini dilakukan setelah sistem telah selesai dan telah divalidasi. Hal ini bertujuan agar pengguna sistem ini dapat dengan mudah mempelajari dan memakai aplikasi tersebut

BAB 4
METODE PENENTUAN FREKUENSI NATURAL DAN MODE SHAPE

Terdapat banyak metode untuk menentukan frekuensi natural dan mode shape. Meskipun demikian tidak ada satu metode yang sesuai untuk segala persoalan. Metode yang efisien untuk persoalan tertentu mungkin kurang atau tidak efisien untuk persoalan yang lain. Berikut ini adalah overview dari tiga metode yang dianggap mempunyai potensi sebagai kandidat karena dimungkinkan dimanipulasi menggunakan komputer.

Ketiga metode yang dimaksud ketiganya bisa digunakan untuk analisa dalam permasalahan ini, karena itu pertimbangan lain akan diberikan dalam memilih metode yang paling efisien umtuk persoalan ini.

Metode Holzer dan metode transfer matriks menggunakan prinsip ‘trial and eror’ dalam menyelesaikan persoalan ini. Andaikan tidak ada alternatif lain dalam menyelesaikan persoalan ini maka metode tersebut sebenarnya cukup memadai, terutama apabila konvergensi bisa terjadi dengan cepat. Celakanya adalah bahwa hampir tidak mungkin untuk mengetahui sebelumnya apakah konvergensi akan terjadi dengan cepat atau tidak (untuk kondisi ekstrim sering bahwa cara ini tidak menghasilkan konvergensi sama sekali).

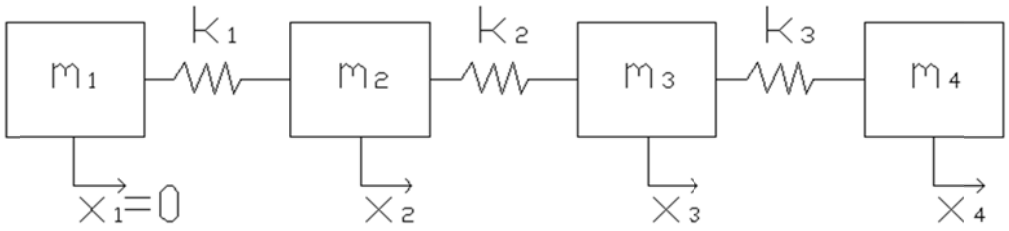
Kemudian untuk salah satu cara untuk menyelesaikan permasalahan ini adalah menggunakan metode eigenproblem. Dengan menggunakan metode matriks eigenproblem ini keberatan-keberatan diatas tidak akan terjadi. Disamping itu sama sekali tidak ada aspek ‘trial and error’ sehingga waktu kommputer dapat dikurangi, dan harga yang didapatkanakan lebih akurat karena bisa secara langsung didapatkan.

Namun demikian terdapat masalah dalam implementasi numeriknya. Kunci utamanya terletak pada pemilihan manipulasi numeric yang sesuai. Hal ini akan dibahas dengan lebih mendetail pada bab mendatang.

4.1 Metode Holzer

Metode ini dimulai dengan memperkirakan besarnya frekuensi natural secara sembarang. Frekuensi natural perkiraan ini akan menjadi frekuensi sebenarnya (yaitu salah satu frekuensi sebenarnya) apabila memenuhi persyaratan tertentu. Dalam banyak hal percobaan ini harus dilakukan beberapa kali untuk menentukan satu frekuensi natural. Untuk mencari frekuensi natural yang lainnya maka proses tersebut diatas harus diulang kembali. Dan apabila untuk derajat kebebasan yang besar akan mengalami kesulitan yang berat. Cara ini bisa dilakukan dengan bantuan tabel seperti ditunjukkan oleh tabel 4.1.

Cara akan diilustrasikan dengan bantuan gambar 4.1, yaitu sistem dengan empat derajat kebebasan, tanpa damping dan tanpa eksitasi.



Gambar 4. 1 Sistem tanpa eksitasi dan tanpa damping

Persyaratan yang dimaksud diatas adalah persamaan ulang IV-1 dan persamaan uji IV-2 berikut

$$x_n = x_{n-1} - \sum_{i=1}^{n-1} \frac{(w^2 m_i - jwc_i)}{(k_{n-1} + jwq_i)} x_i \qquad \text{IV.1}$$

$$\sum_{i=1}^{n} (w^2 m_i - jwc_i) x_i = 0 \qquad \text{IV.2}$$

apabila pengaruh redama diabaikan dalam menghitung frekuensi natural dan mode shapes serta dari definisi eksitasi sama dengan nol, maka persamaan IV-1 dan IV-2 menjadi

$$x_n = x_{n-1} - \sum_{i=1}^{n-1} \frac{w^2 j_i x_i}{k_{n-1}} \qquad \text{IV.3}$$

$$\sum_{i=1}^{n} w^2 j_i x_i = 0 \qquad \text{IV.5}$$

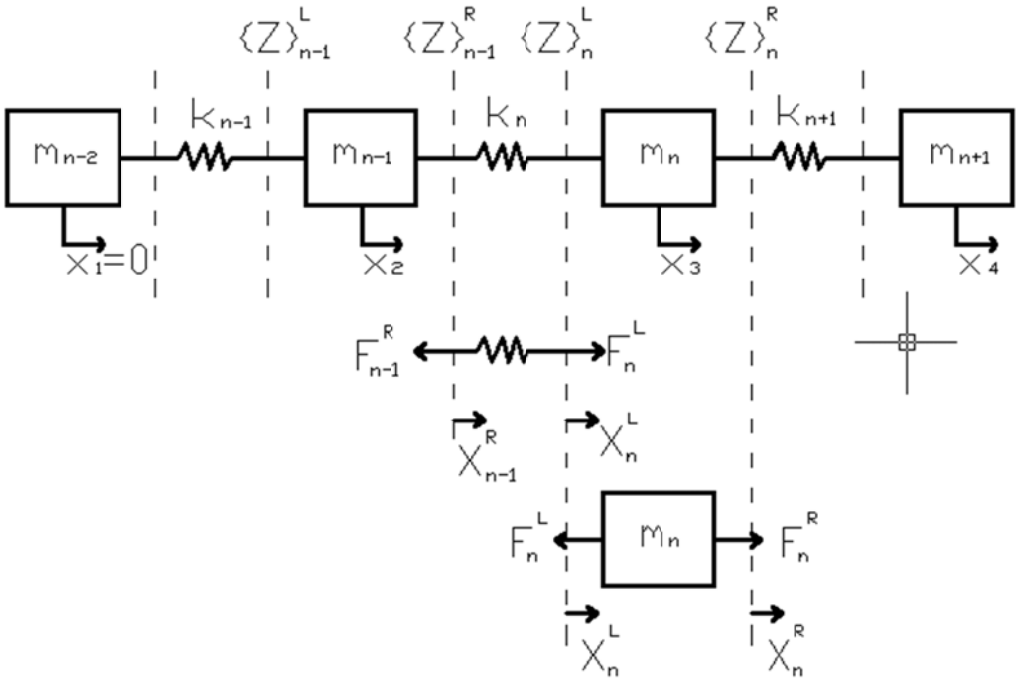
kedua persamaan diatas dapat ditulis dalam bentuk tabulasi sebagai berikut

	1	2	3	4	5	6
	m	x	$m\ x\ w^2$	$\sum m\ x\ w^2$	k	$\frac{1}{k}\sum m\ x\ w^2$
1	m_1	1,00			k_1	
2	m_2				k_2	
N-1	m_{N-1}				k_{N-1}	
N	m_N					

Tabel 4.1 Tabulasi Holzer

4.2 Metode Transfer Matriks

Perhatikan sistem pada Gambar 4.2 berikut



Gambar 4.2 Metode Transfer Matriks

apabila hukum newton digunakan untuk massa m_n, maka didapat

$$m_n\ddot{x}_n = F_n^R - F_n^L \qquad \text{IV.6}$$

dan untuk gerakan harmonikpersamaan IV-5 bisa ditulis menjadi

$$F_n^R = -w^2 m_n x_n + F_n^L \qquad \text{IV.7}$$

dengan asumsi bahwa m_n benda tegar, maka displasemen sebelah kiri akan sama dengan displasemen sebelah kanan, sehingga

$$x_n^R = x_n^L \qquad \text{IV.8}$$

Persamaan IV-6 dan IV-7 dapat ditulis dalam bentuk matriks

$$\begin{Bmatrix} x \\ F \end{Bmatrix}_n^R = \begin{bmatrix} 1 & 0 \\ -wm & 1 \end{bmatrix} \begin{Bmatrix} x \\ F \end{Bmatrix}_n^L \qquad \text{IV.9}$$

Vektor disebelah kanan disebut state vektor, sedangkan matriksnya sendiri disebut point matriks.

Pada pegas k_n bekerja gaya pada ujung-ujungnya yang sama besar, yaitu

$$F_{n-1}^R = F_n^L \qquad \text{IV.10}$$

atau

$$x_n^L - x_{n-1}^R = \frac{F_{n-1}^R}{k_n} \qquad \text{IV.11}$$

persamaan IV-9 dan IV-10 dalam bentuk matriks ditulis

$$\begin{Bmatrix} x \\ F \end{Bmatrix}_n^L = \begin{bmatrix} 1 & 1/k \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ F \end{Bmatrix}_{n-1}^R \qquad \text{IV.12}$$

matriks persegi pada persamaan IV-11 disebut field matrix.

Hubungan antara state vector pada station n dengan state vector pada station n-1 adalah

$$\begin{Bmatrix} x \\ F \end{Bmatrix}_n^R = \begin{bmatrix} 1 & 1/k \\ -wm & 1 - wm/k \end{bmatrix} \begin{Bmatrix} x \\ F \end{Bmatrix}_{n-1}^R \qquad \text{IV.13}$$

atau

$$\{Z\}_n^R = H_n \ \{Z\}_{n-1}^R \qquad \text{IV.14}$$

dengan H_n

$$H_n = \begin{bmatrix} 1 & 1/k \\ -wm & 1 - wm/k \end{bmatrix} \qquad \text{IV.15}$$

sebagai transfer matriks.

Dengan persamaan IV-12 state vector pada station n dapat dihubungkan dengan state vector pada ujung sistem, sebagai berikut

$$\{Z\}_n^R = [H_n \quad H_{n-1} \quad \ldots\ldots \quad H_2 \quad H_1]\{Z\}_0^R \qquad \text{IV.16}$$

Dengan diketahuinya state vector pada station n menggunakan asumsi harga w^2, maka state vector pada station n dapat ditentukan. Tergantung pada kondisi batas, x_n atau F_n dapat diplotkan sebagai fungsi w^2. Harga w^2 ini adalah frekuensi natural bila kondisi batas terpenuhi.

4.3 Metode Matriks Eigenvalue

Seperti diketahui masalah eigenvalue ditulis sebagai berikut

$$[L]\{x\} = \lambda\{x\} \qquad \text{IV.17}$$

dengan harga semua konstan yang memenuhi persamaan IV-16 disebut eigenvalues dan vektor $\{x\}$ yang memenuhi persamaan IV-16 untuk setiap harga λ disebut eigenvector atau eigenfunction. Dengan demikian setiap perkalian dengan konstanta sembarang terhadap eigenvector juga merupakan eigenvector. Atau dengan kata lain bahwa harga eigenvector bukan harga absolut.[BATHE, 1976]

Bila n adalah ordo dari matriks $[L]$, maka terdapat n penyelesaian non trivial. Pernyelesaian non trivial ke i memberikan hasil eigenvalue λ dan eigenvector $\{x\}$.

Persamaan getaran tanpa damping tanpa eksitasi (dengan anggapan harmonis) akan menghasilkan satu persamaan yang akan membentuk kedalam masalah eigenvalue, yaitu

$$[k]\{A\} = \omega^2[m]\{A\} \qquad \text{IV.18}$$

atau,

$$[L]\{A\} = \lambda\{A\} \qquad \text{IV.19}$$

dengan $\lambda = w^2$ adalah harga eigenvaluesnya.

Apabila matriks $[L]$ pada persamaan IV-19 mempunyai bentuk simetris, maka dikatakan masalah eigenvalue ini mempunya bentuk standar (standard eigenvalue problem). Sebaliknya disebut ’non standard eigenvalue problem’.

Banyak masalah-masalah teknik yang akhirnya muncul sebagai masalah eigenvalue. Meskipun tidak selalu dalam bentuk standar, namun selalu dapat ditransformasikan kembali kedalam bentuk standar dengan tanpa mengalami kesulitan yang berarti.

Terdapat 3 cara dalam menyelesaikan persamaan IV-18 yaitu dengan matriks dinamik. Inverse matriks dinamik dan standar matriks simetris.

4.3.1 Matriks dinamik

Apabila persamaan IV-18 dikalikan dengan inverse dari matriks $[k]$, maka didapatkan hasil

$$\frac{1}{\omega^2}\{A\} = [Q]\{A\} \qquad \text{IV.20}$$

dengan

$$[Q] = [k]^{-1}[m] \qquad \text{IV.21}$$

matriks $[A]$ disebut matriks dinamik dan merupakan matriks simetris. Namun apabila ordo matriks menjadi besar maka untuk menghitung inverse dari matriks $[k]$ tidak terlalu mudah untuk dilakukan.

4.3.2 Inverse Matriks Dinamik

Apabila persamaan IV-17 dikalikan dengan inverse dari matriks $[m]$ maka akan didapatkan

$$[Q]^{-1}\{x\} = \omega^2\{x\} \qquad \text{IV.22}$$

dengan

$$[Q]^{-1} = [m]^{-1}[k] \qquad \text{IV.23}$$

matrik $[A]^{-1}$ merupakan inverse dari matriks $[A]$ dan bukan merupakan matriks simetris. Sehingga untuk matriks dengan ordo besar akan menimbulkan masalah seperti sebelumnya.

4.3.3 Matriks Simetris Standar

Karena manipulasi yang tidak terlalu sulit, metode ini paling banyak dipakai. Dengan metode ini setiap masalah eigenvalue tidak standar harus dirubah dahulu kedalam bentuk eigenvalue standar menggunakan transformasi sebagai berikut

$$\{\tilde{A}\} = [m]^{1/2}\{A\} \qquad \text{IV.24}$$

atau

$$\{A\} = [m]^{-1/2}\{\tilde{A}\} \qquad \text{IV.25}$$

substitusi persamaan IV-23 pada persamaan IV-17, maka akan didapatkan

$$[k]\,[m]^{-1/2}\,\{\tilde{A}\} = \omega^2[m][m]^{-1/2}\,\{\tilde{A}\} \qquad \text{IV.26}$$

perkalian persamaan IV-24 dengan matriks $[m]^{-1/2}$, maka akan didapatkan persamaan seperti permasalahan eigenvalue standar

$$[D]\{\tilde{A}\} = \omega^2\{\tilde{A}\} \qquad \text{IV.27}$$

dengan

$$[D] = [m]^{-1/2}[k][m]^{-1/2} \qquad \text{IV.28}$$

Matriks $[D]$ pada persamaan IV-27 diatas sekarang mempunya bentuk simetris. Matriks simetris selalu menguntungkan untuk dimanipulasi dengan komputer, baik dari segi memory yang digunakan maupun dari algoritmanya.

Eigenvector yang didapatkan dari eigenvalue standar diatas, $\{\tilde{A}\}$ harus ditransformasikan kembali untuk mendapatkan penyelesaian yang diinginkan menggunakan persamaan

$$\{A\} = \quad [m]^{-\frac{1}{2}}\ \{\tilde{A}\} \qquad \text{IV.29}$$

dan $\{A\}$ adalah penyelesaian persamaan IV-18.

BAB 5
PENYELESAIAN NUMERIK DAN IMPLEMENTASI PROGRAM

5.1 Penyelesaian Numerik

5.1.1 Perhitungan Pre-Processor

Dalam persamaan getaran longitudinal, variabel yang digunakan untuk menghitung frekuensi natural dan mode shapes adalah besarnya massa dan kekakuan pegas. Dan untuk getaran torsional variabel yang digunakan untuk menghitung frekuensi natural dan mode shapes adalah besarnya momen inersia massa dan kekakuan pegas.

- Getaran longitudinal
  - dalam menentukan besarnya massa benda menggunakan persamaan sebagai berikut

$$m = \frac{W}{g} \qquad \text{V.1}$$

dengan: W adalah berat benda

g adalah percepatan gravitasi (10m/s^2)

  - dalam menentukan besarnya kekakuan pegas menggunakan persamaan sebagai berikut

$$k = \frac{A\,E}{L} \qquad \text{V.2}$$

dengan: A adalah luasan penampang

E adalah Modulus Young benda

L adalah panjang

- Getaran torsional

dalam menentukan besarnya inersia massa adalah menggunakan rumus

$$J = \iiint_v r^2\, dm \qquad \text{V.3}$$

Namun untuk saat ini besarnya inersia massa menggunakan persamaan pendekatan sebagai berikut

- Inersia massa propeller

$$J_{Prop} = \frac{5\,R^2\,m_{prop}}{16} + \frac{\pi\,\rho\,r^4\,L}{6} \qquad \text{V.4}$$

dengan: R adalah jari-jari propeller
m adalah massa propeller
ρ adalah kerapatan massa poros
r adalah jari-jari poros
L adalah panjang poros

- Inersia massa poros

$$J_{poros} = \frac{\pi \rho r^4 L}{6} \quad \text{V.5}$$

dengan: ρ adalah kerapatan massa poros
r adalah jari-jari poros
L adalah panjang poros

- Inersia massa flywheel

$$J_{flywheel} = \frac{\pi \rho r^4 L}{2} \quad \text{V.6}$$

dengan: ρ adalah kerapatan massa flywheel
r adalah jari-jari flywheel
L adalah panjang flywheel

- Inersia massa silinder mesin induk [HARTOG, 1956]

$$J_{Silinder} = \frac{(\pi \rho r^4 L + m r_{piston}^2)}{2} \quad \text{V.7}$$

dengan: R adalah jari-jari propeller
m adalah massa propeller
ρ adalah kerapatan massa poros
r adalah jari-jari poros
L adalah panjang poros

- Kekakuan pegas

$$K = \frac{\pi r^4 E}{4 L (1+\partial)} \quad \text{V.8}$$

dengan: r adalah jari-jari
E adalah Modulus Young
∂ adalah angka poison
L adalah panjang

5.1.2 Penyelesaian masalah eigenvalue standart

Algoritma yang digunakan untuk menyelesaikan masalah eigenvalue standar disini adalah metode Jacobi. Iterasi ke k untuk penyelesaian $[D]\{x\} = -w^2\{x\}$ didefinisikan sebagai.

$$D_{k+1} = P_k^T D_k P_k \qquad \text{V.9}$$

dengan P_k sebagai matriks orthogonal dan P_k^T sebagai transpose matriks dari matriks P_k.

Matriks P_k disebut sebagai matriks rotasi yang digunakan untuk mentransformasi semua elemen pada maktriks D_k, kecuali elemen pada diagonal utama, menjadi nol. Untuk mentransformasi elemen (i,j) menjadi nol, maka matriks orthogonal P_k adalah sebagai berikut.

$$P_k = \begin{bmatrix} 1 & & & & & & & & \\ & . & & & & & & & \\ & & 1 & & & & & & \\ & & & \cos x & \sin x & & & & \\ & & & -\sin x & \cos x & & & & \\ & & & & & 1 & & & \\ & & & & & & . & & \\ & & & & & & & 1 & \end{bmatrix} \begin{matrix} \\ \\ \\ i \\ j \\ \\ \\ \\ \end{matrix}$$

dimana

$$\sin x = \frac{\bar{k}_{ji}}{(\bar{k}_{ii}^2 + \bar{k}_{ji}^2)^{1/2}} \qquad \text{V.10}$$

dan

$$\cos x = \frac{\bar{k}_{ii}}{(\bar{k}_{ii}^2 + \bar{k}_{ji}^2)^{1/2}} \qquad \text{V.11}$$

Selama proses iterasi berlangsung matriks D_k selalu simetris untuk setiap harga k. karena itu yang digunakan hanya segitiga atas atau bawah saja dari matriks tersebut (upper or lower triangle). Hal ini adalah keuntungan utama dari metode Jacobi.

5.1.3 Metode QR Iteration

Penamaan "QR Iteration" diambil dari notasi yang digunakan dalam algoritma tersebut [BATHE,1976]. Langkah-langkah yang digunakan adalah dengan memecah matriks $[D]$ kedalam bentuk

$$[D] = [Q]\,[R] \qquad \text{V.12}$$

dimana $[Q]$ merupakan matriks orthogonal dan $[R]$ merupakan "upper or lower triangular" matriks, kemudian kita bentuk

$$[R][Q] = [Q]^T[D][Q] \qquad \text{V.13}$$

Dengan menghitung $[R][Q]$ pada hakekatnya kita juga sudah menyelesaikan transformasi dari bentuk

$$\begin{aligned} D_2 &= P_1^T D_1 P_1 \\ D_3 &= P_2^T D_2 P_2 \\ &\vdots \\ D_{k+1} &= P_k^T D_k P_k \\ &\vdots \end{aligned} \qquad \text{V.14}$$

faktorisasi dari persamaan V.14 dapat dibentuk dengan mengaplikasikan metode Gram-Schmidt triangular menggunakan matriks rotasi Jacobi, dengan

$$P_{n,n-1}^T \ldots . P_{3,2}^T P_{2,1}^T = R \qquad \text{V.15}$$

matriks rotasi $P_{j,i}^T$ dipilih untuk membuat nol pada elemen (j,i) pada matriks $[D]$. Dan untuk Q adalah

$$[Q] = P_{2,1}^T P_{3,2}^T \ldots P_{n,n-1}^T \qquad \text{V.16}$$

QR iterasi dibentuk melalui pengulangan dari proses yang diberikan pada persamaan 1 dan 2. Menggunakan notasi $D_1 = D$, kita bentuk

$$D_k = Q_k R_k \qquad \text{V.17}$$

dan kemudian

$$D_{k+1} = R_k Q_k \qquad \text{V.18}$$

5.1.4 Penentuan eigenvalue dan eigenvector

Pada prosedur QR iterasi, dari proses iterasi hanya didapatkan hasil eigenvalue dan eigenvector dari matriks tridiagonal $[D]$. Yang diinginkan adalah menentukan besarnya frekuensi natural dan besarnya mode shapes. Untuk mencari besarnya frekuensi natural dapat ditentukan melalui hasil eigenvalue yang didapatkan dari proses QR iterasi tersebut.

$$\lambda = \omega^2 \qquad \text{V.19}$$

sehingga untuk menentukan besarnya frekuensi natural (ω) adalah

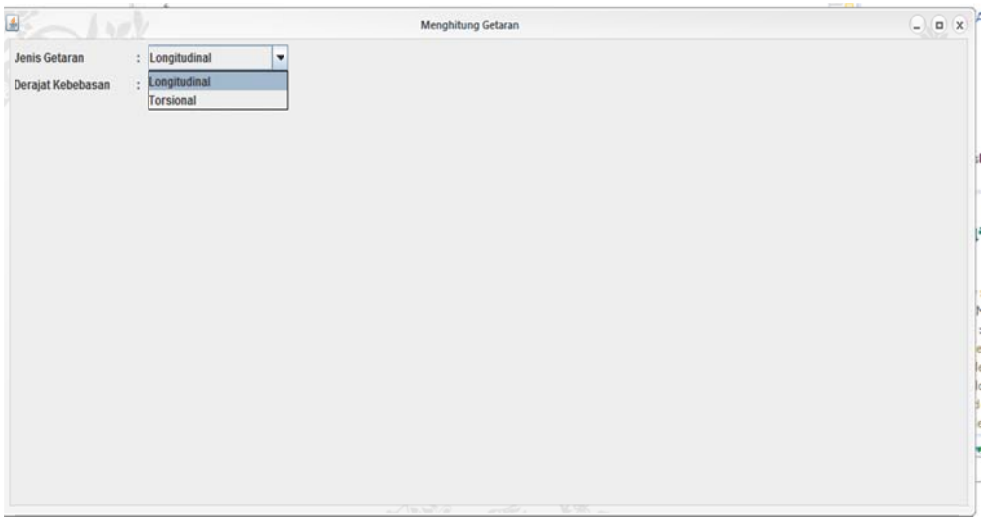
$$\omega = \sqrt{\lambda} \qquad \text{V.20}$$

untuk mencari besarnya mode shapes dapat ditentukan melalui hasil eigenvector yang didapatkan dari proses QR iterasi tersebut. Dari proses iterasi didapatkan eigenvector $\{\tilde{x}\}$, maka untuk mendapatkan hasil mode shapes yang asli kita harus mentransformasikannya kembali dengan persamaan IV.25.

$$\{A\} = [m]^{-\frac{1}{2}} \{\tilde{A}\} \qquad \text{V.21}$$

5.2 Implementasi Program

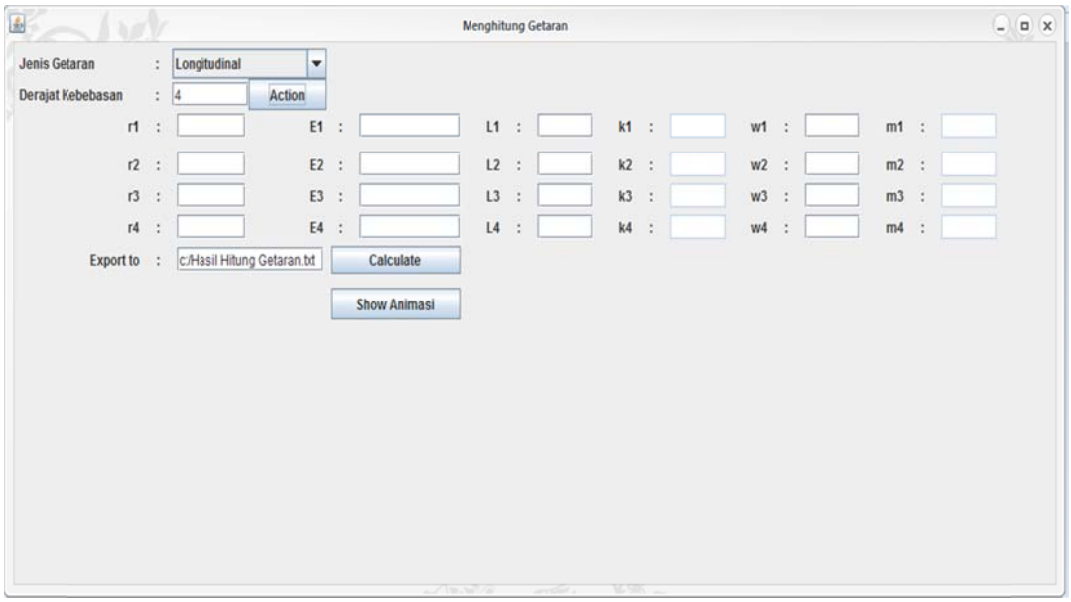
Pada proses *interface* awal, hal pertama yang dilakukan adalah dengan memilih opsi apakah kita akan menghitung frekuensi dan mode shapes pada getaran longitudinal ataukah pada getaran torsional. Gambar interface awal pada program terlihat seperti Gambar 5.1.



Gambar 5.1 Interface awal program

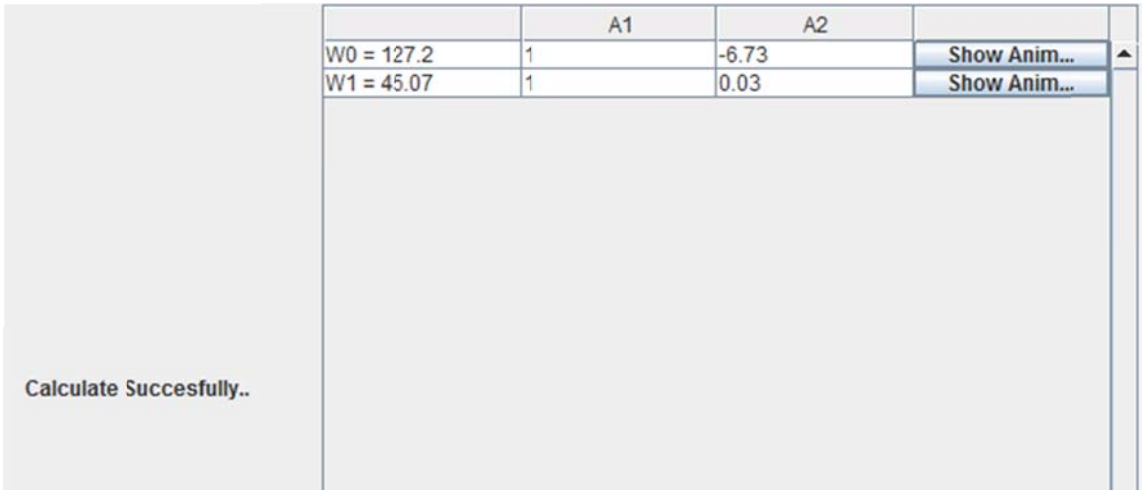
5.2.1 Getaran Longitudinal

Setelah kita memilih getaran longitudinal pada interface awal yang akan dihitung adalah besarnya massa dan kekakuan pegas. Lalu yang dilakukan adalah menginput harga-harga dari setiap parameter yang telah tertera pada program. Parameter-parameter yang diinput antara lain adalah besarnya berat dari benda untuk mengetahui besarnya massa, kemudian besarnya jari-jari, modulus young, dan panjang untuk mengitung besarnya harga kekakuan pegas. Interface dapat dilihat pada Gambar 5.2.

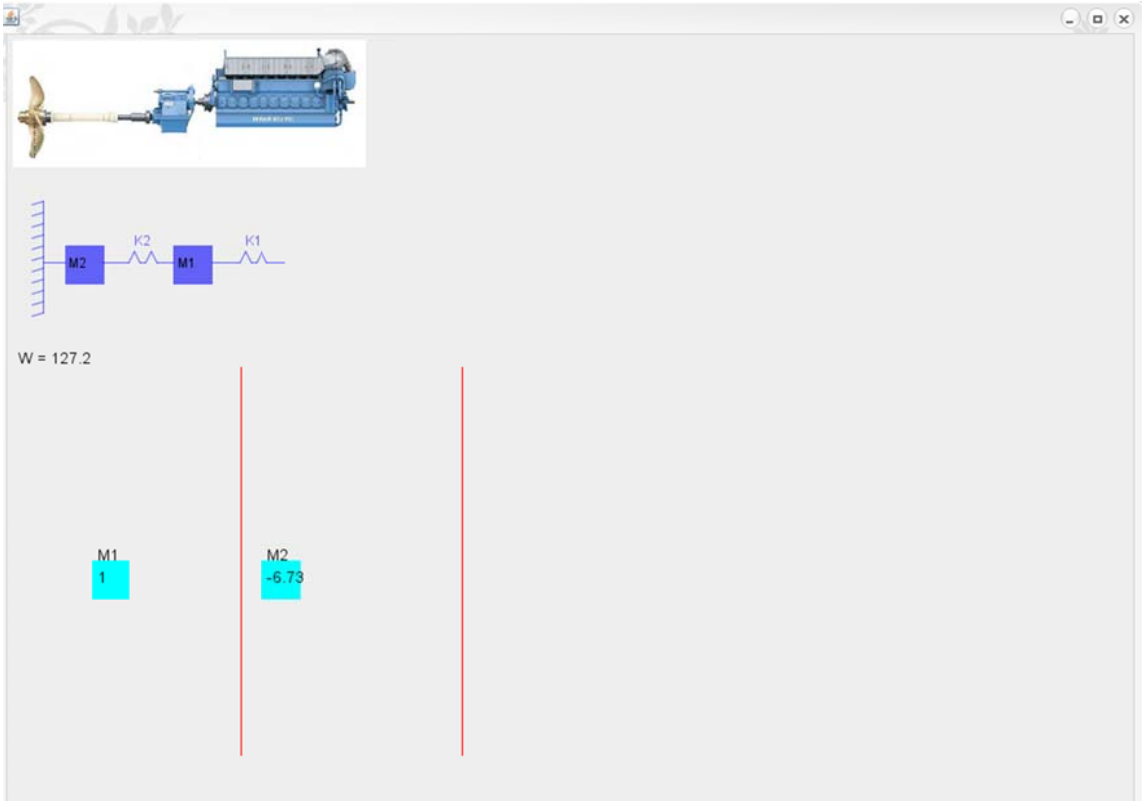


Gambar 5.2 Interface perhitungan massa dan kekakuan pegas

Apabila kita telah menginput seluruh parameter, kemudian kita tekan tombol calculate untuk memulai perhitungan. Apabila perhitungan telah sukses maka kita dapat melihat hasil dari perhitungan yang telah dilakukan dengan cara menekan tombol open file. Dan kita akan mengetahui berapa besarnya frekuensi natural dan mode shapesnya. Dan apabila kita ingin melihat animasi dari mode shapes yang dihasilkan kita tekan tombol show animasi. Hasil dari perhitungan dan animasi dapat dilihat pada Gambar 5.3 dan Gambar 5.4.



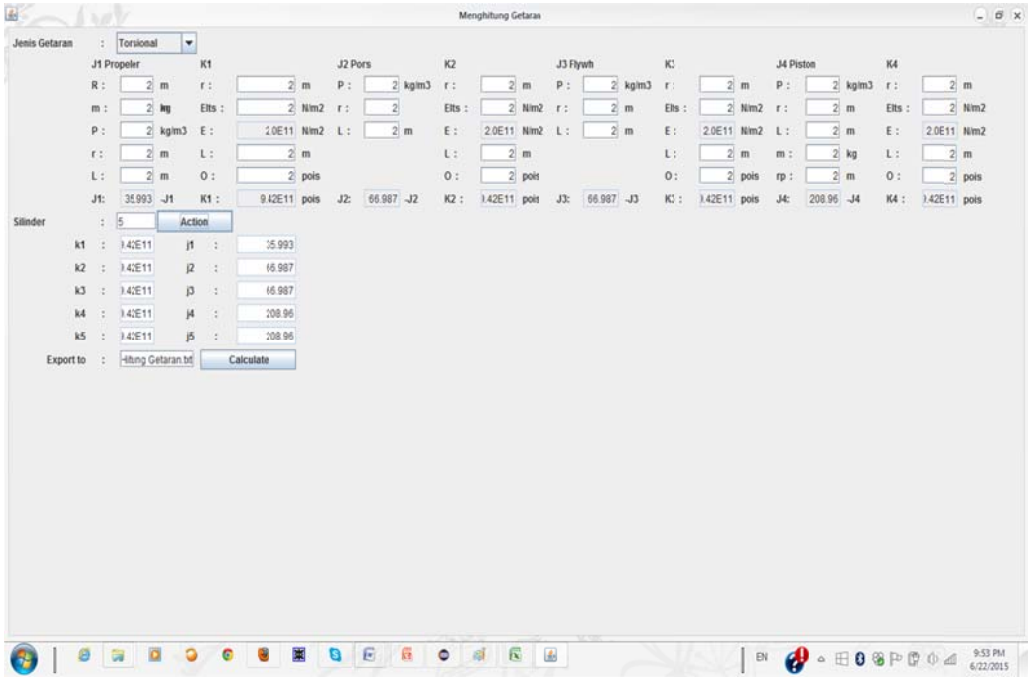
Gambar 5.3 Hasil Frekuensi natural dan Mode shapes getaran longitudinal



Gambar 5.4 Animasi yang dihasilkan pada getaran longitudinal

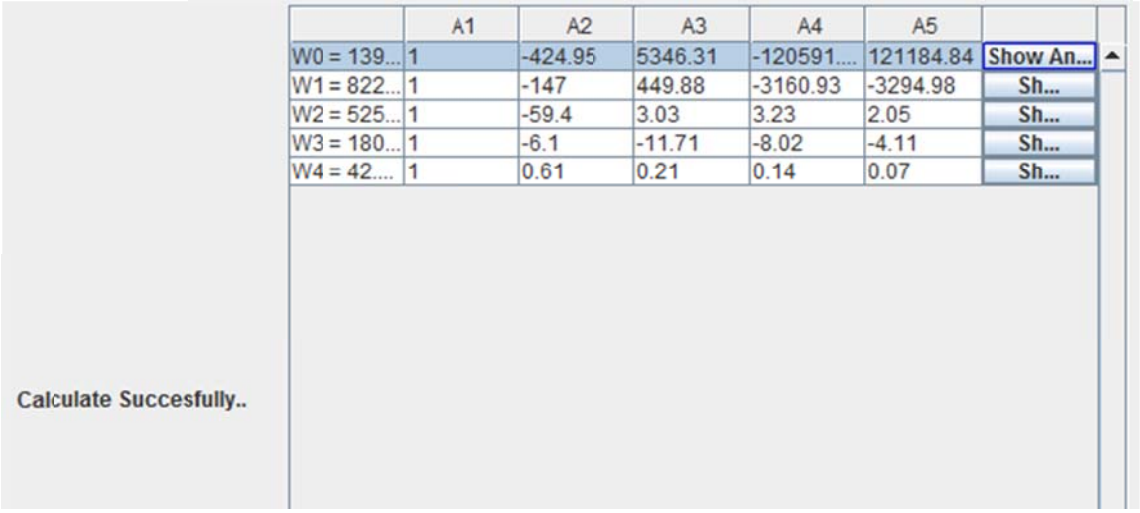
5.2.2 Getaran Torsional

Apabila kita memilih getaran torsional pada interface awal yang akan dihitung adalah besarnya inersia massa dan kekakuan pegas torsional. Lalu yang dilakukan adalah menginput harga-harga dari setiap parameter yang telah tertera pada program. Parameter-parameter yang diinput antara lain adalah besarnya massa propeller, jari-jari propeller, jari-jari poros, kerepatan massa poros dan panjang poros untuk menentukan besarnya inersia massa propeller dan inersia massa poros. Lalu kerapatan massa flywheel, jari-jari flywheel dan panjang flywheel untuk menentukan inersia massa flywheel. Jari-jari poros engkol mesin, kerapatan massa poros engkol, panjang poros engkol, jari-jari piston dan massa piston untuk menentukan inersia massa silinder mesin. kemudian besarnya jari-jari, modulus young, angka poison, dan panjang untuk mengitung besarnya harga kekakuan pegas torsional. Dan terdapat input mengenai jumlah silinder mesin induk yang bertujuan untuk menentukan besarnya J_5, $J_{6,}$ dan seterusnya. Interface dapat dilihat pada Gambar 5.5.

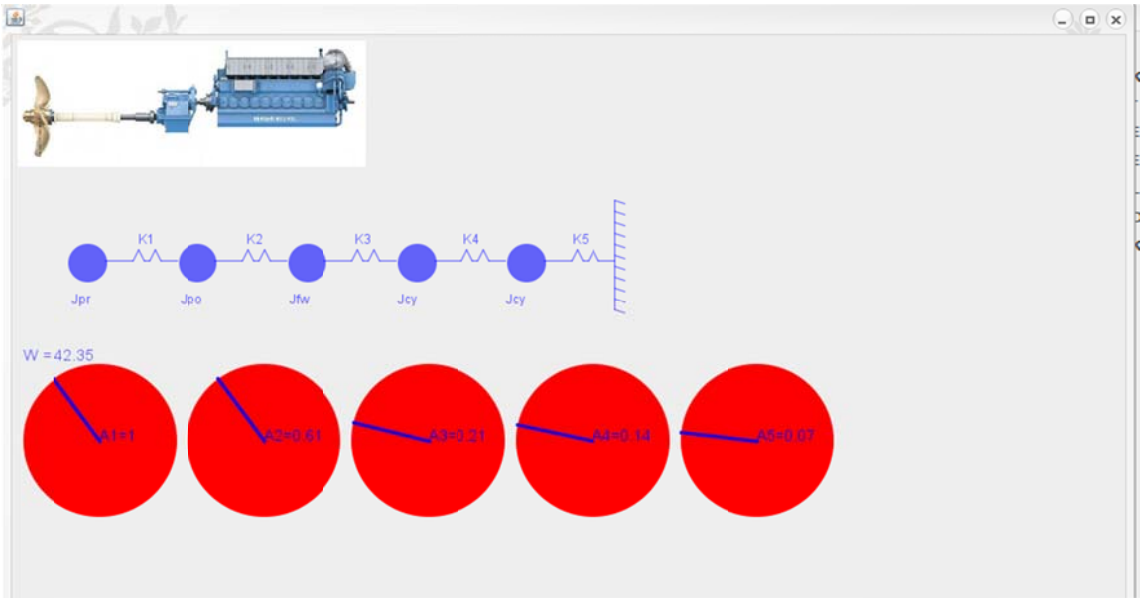


Gambar 5.5 Interface perhitungan inersia massa dan kekakuan pegas torsional

Apabila kita telah menginput seluruh parameter, kemudian kita tekan tombol calculate untuk memulai perhitungan. Apabila perhitungan telah sukses maka kita dapat melihat hasil dari perhitungan yang telah dilakukan dengan cara menekan tombol open file. Dan kita akan mengetahui berapa besarnya frekuensi natural dan mode shapesnya. Dan apabila kita ingin melihat animasi dari mode shapes yang dihasilkan kita tekan tombol show animasi. Hasil dari perhitungan dan animasi dapat dilihat pada Gambar 5.6 dan Gambar 5.7.



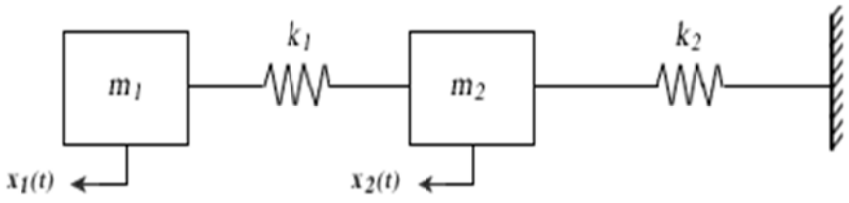
Gambar 5.6 Hasil Frekuensi natural dan Mode shapes getaran torsional



Gambar 5.7 Animasi yang dihasilkan pada getaran torsional

5.3 Validasi Program

Validasi dilakukan dengan membandingkan hasil yang bisa dihitung secara analitis. Agar hal ini bisa dilakukan secara manual, tentu saja order dari matriks harus dibatasi. Untuk validasi ini diambil system dengan dua derajat kebebasan yang dengan mudah bisa dihitung menggunakan analisa yang ada. Gambar 5.8 adalah skema dari sistem dengan dua massa pada getaran longitudinal yang digunakan untuk validasi.



Gambar 5.8 Sistem dengan dua massa getaran longitudinal

5.3.1 Perhitungan Manual

jari-jari (r_1)	0,2
Modulus young (E_1)	200.000.000
panjang (L_1)	6
sehingga	
kekakuan pegas (k_1)	4186666,667

jari-jari (r_2)	0,5
Modulus young (E_2)	200.000.000
panjang (L_2)	1
sehingga	
kekakuan pegas (k_2)	157000000

berat (W_1)	20000
sehingga	
m1	2000

berat (W_2)	100000
sehingga	
m2	10000

$m_1 = 2.000$

$m_2 = 10.000 = 5m_1$

$k_1 = 4.186.666,667$

$k_2 = 157.000.000 = 37,5\ k_1$

matriks yang dihasilkan,

$-\omega^2[m] = \begin{bmatrix} -\omega^2 m_1 & \\ & -\omega^2 m_2 \end{bmatrix}$ dan $[k] = \begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 + k_2 \end{bmatrix}$

Sehingga,

$$[D(\omega_n)] = \begin{bmatrix} -\omega^2 m_1 + k_1 & -k_1 \\ -k_1 & -\omega^2\ 5m_1 + 38{,}5k_1 \end{bmatrix}$$

Untuk menghitung frekuensi natural,

$$det[D(\omega_n)] = 0$$

Atau $5{m_1}^2\omega^4 - 43{,}5\ m_1 k_1 \omega^2 + 37{,}5\ {k_1}^2 = 0$

Untuk mencari frekuensi natural dengan menggunakan rumus

$$\omega_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

sehingga didapatkan

ω_1 = 45,0679677971483

ω_2 = 127,204081236516

untuk menentukan mode shapes menggunakan persamaan, dan hanya menggunakan n-1 persamaan

$$[D(\omega_n)]\{A\} = 0$$

persamaan yang digunakan

$$(-\omega^2 m_1 + k_1)\, A_1 - k_1 A_2 = 0$$

sehingga mode shapes ke 1

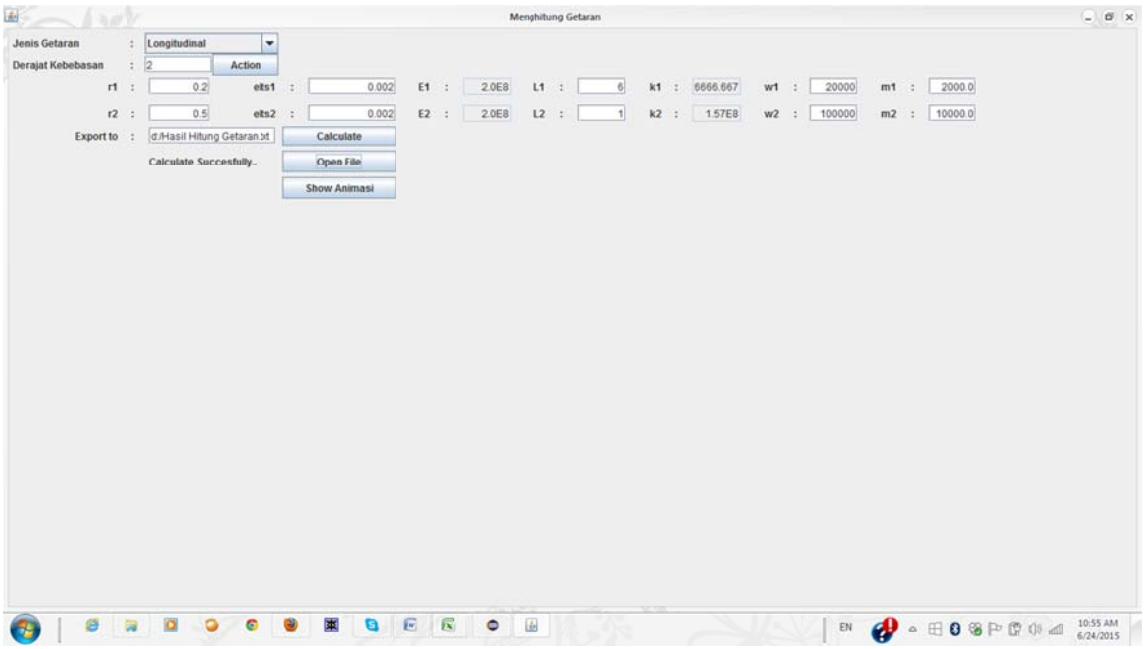
$$A_{11} = 1$$
$$A_{21} = \ 0{,}029718975$$

sehingga mode shapes ke 2

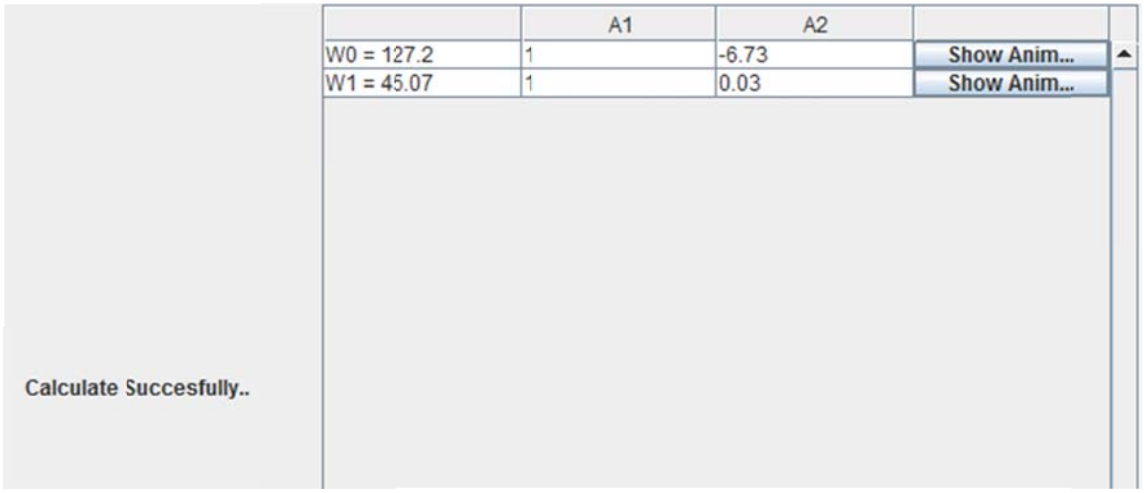
$$A_{12} = 1$$
$$A_{22} = \ -6{,}72970707401215$$

Hasil computer run pada halaman berikut menunjukkan bahwa untuk contoh ini baik harga frekuensi natural dan mode shapes tidak ada perbedaan, dengan kata lain kesalahannya adalah 0%.

5.3.2 Hasil computer run



Gambar 5.9 Input pada program



Gambar 5.10 hasil frekuensi natural dan mode shapes

BAB 6
KESIMPULAN DAN SARAN

6.1 Kesimpulan

1. Program yang digunakan untuk menghitung frekuensi natural dan mode shapes sudah terlaksanadalam bentuk software dengan menggunakan aplikasi JAVA.

2. Pembuatan model diskrit menjadi n massa akan sangat berpengaruh terhadap hasil analisa. Semakin banyak memecah massa pada system, maka diharapkan hasil yang didapatkan semakin akurat

3. Informasi yang disediakan dalam *software* telah meliputi harga frekuensi natural dan mode shapes sehingga dapat digunakan sebagai sarana pembelajaran dalam mempermudah pemahaman.

6.2 Saran

1. Perlu dilakukan analisa lebih lanjut dalam menentukan besarnya massa, inersia massa, kekakuan pegas dan kekakuan pegas torsional pada system propulsi kapal.

2. Pembuatan *software* masih terbatas sampai hasil perhitungan dan animasi 2D, disarankan mengembangkan program lagi sehingga didapatkan hasil visual 3D untuk memudahkan pemahaman.

DAFTAR PUSTAKA

Bathe, J, 1996, **Finite Element Procedures**, Prentice-Hall, New Jersey

Goodiere, J. N, 1894, **Teori Elastisitas**, Edisi Ketiga, Sapdodadi, Jakarta.

Hartog, J. P. D, 1956, **Mechanical Vibrations,** Edisi Keempat, McGraw-Hill Book Company, New York.

Harrington, D. L, 1971, **Marine Engineering**, The Society of Naval Architects and Marine Engineers, New York.

Imron, A, 1994, **Getaran Kapal 1**, Jurusan Teknik Perkapalan, Institut Teknologi Sepuluh Nopember, Surabaya.

Priatmoko, D, 2003, **Analisa Getaran Dan Sistem Perporosan Pada Reduction Gear KM.Kumala,** Institut Teknologi Sepuluh Nopember, Surabaya

Thomson, W. T, 1976, **Theory of Vibration with Application**, Prentice Hall Inc, New Jersey.

Online Reference, http://java.lyracc.com/java-untuk-pemula/bab-i-pendahuluan, diakses pada tanggal 6 juni 2015 pukul 23.31.

```
package doublea;


import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Collection;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.JTextField;


public class RunMatrix extends JComponent{

	public static double matrixPD[][];
	public static double matrixPTN[][];
	public static double matrixPT[][];
	public static double R[][];
	public static double Q[][];
	public static double Qawal[][];
	public static double Qakhir[][];
	public static double D[][];
	public static int[] k;
	public static int[] m;
	public static boolean valid;
	public static int ik=1;
	public static int iteration=2;
	private static boolean INFO = true;
```

```
private static boolean DEBUG = true;
private static Color m_tBlue = new Color(0, 0, 255, 150);
final static JFrame frame = new JFrame();

JTextField txtOrdoLongitudinal = new JTextField(6);
JTextField txtOrdoTorsional = new JTextField(3);
JPanel jpContent = new JPanel(new FlowLayout());
final JPanel jp = new JPanel();
JPanel mainPanel = new JPanel(new GridLayout(2, 1));
final GridBagConstraints gbc = new GridBagConstraints();
private Collection<JTextField> txtvalk = new ArrayList<JTextField>();
private Collection<JTextField> txtvalm = new ArrayList<JTextField>();
JButton btnCalculate = new JButton("Calculate");
public static JTextArea textArea = new JTextArea();
public static JLabel sccs = new JLabel("");
public static JButton btnOpenFile = new JButton("Open File");
public static JButton btnShowAnimasi = new JButton("Show Animasi");
public static JTextField expto = new JTextField(3);
final JComboBox jComboBox1 = new JComboBox();
private static String notasi;

final int txtwidth = 5;

final JLabel lbl_ordo_T = new JLabel("Derajat Kebebasan");
final JLabel sp_T = new JLabel(":");
final JButton btnActLongitudinal = new JButton("Action");
final JLabel lbl_ordo_J = new JLabel("Silinder");
final JLabel spt_J = new JLabel(":");
final JButton btnActTorsional = new JButton("Action");

//
final JLabel lbltorpropelerJ1info = new JLabel("1,25(0,25 R2m)+1/3 P 3.14/2 r4 L");
final JLabel lbltorporosJ2info = new JLabel("3.14/3 Pr4L");
final JLabel lbltorflywheelJ3info = new JLabel("P 3.14/2 r4 L");
final JLabel lbltorpistonJ4info = new JLabel("P 3.14/2 r4 L + mr2p/2");


final JLabel lbl_J1 = new JLabel("J1 Propeler");
final JLabel lbl_J1_R = new JLabel("R  :");
final JTextField txt_J1_R = new JTextField(txtwidth);
final JLabel lbl_J1_m = new JLabel("m  :");
final JTextField txt_J1_m = new JTextField(txtwidth);
final JLabel lbl_J1_P = new JLabel("P  :");
final JTextField txt_J1_P = new JTextField(txtwidth);
final JLabel lbl_J1_r = new JLabel("r  :");
final JTextField txt_J1_r = new JTextField(txtwidth);
final JLabel lbl_J1_L = new JLabel("L  :");
final JTextField txt_J1_L = new JTextField(txtwidth);
final JLabel lbl_J1_Result = new JLabel("J1:");
final JTextField txt_J1_Result = new JTextField(txtwidth);
```

```
final JLabel lbl_J2 = new JLabel("J2 Pors");
final JLabel lbl_J2_P = new JLabel("P  :");
final JTextField txt_J2_P = new JTextField(txtwidth);
final JLabel lbl_J2_r = new JLabel("r  :");
final JTextField txt_J2_r = new JTextField(txtwidth);
final JLabel lbl_J2_L = new JLabel("L  :");
final JTextField txt_J2_L = new JTextField(txtwidth);
final JLabel lbl_J2_Result = new JLabel("J2:");
final JTextField txt_J2_Result = new JTextField(txtwidth);

final JLabel lbl_J3 = new JLabel("J3 Flywh");
final JLabel lbl_J3_P = new JLabel("P  :");
final JTextField txt_J3_P = new JTextField(txtwidth);
final JLabel lbl_J3_r = new JLabel("r  :");
final JTextField txt_J3_r = new JTextField(txtwidth);
final JLabel lbl_J3_L = new JLabel("L  :");
final JTextField txt_J3_L = new JTextField(txtwidth);
final JLabel lbl_J3_Result = new JLabel("J3:");
final JTextField txt_J3_Result = new JTextField(txtwidth);

final JLabel lbl_J4 = new JLabel("J4 Piston");
final JLabel lbl_J4_P = new JLabel("P  :");
final JTextField txt_J4_P = new JTextField(txtwidth);
final JLabel lbl_J4_r = new JLabel("r  :");
final JTextField txt_J4_r = new JTextField(txtwidth);
final JLabel lbl_J4_L = new JLabel("L  :");
final JTextField txt_J4_L = new JTextField(txtwidth);
final JLabel lbl_J4_m = new JLabel("m  :");
final JTextField txt_J4_m = new JTextField(txtwidth);
final JLabel lbl_J4_rp = new JLabel("rp  :");
final JTextField txt_J4_rp = new JTextField(txtwidth);
final JLabel lbl_J4_Result = new JLabel("J4:");
final JTextField txt_J4_Result = new JTextField(txtwidth);


//Satuan Torsional K
String lbl_K_inf_r="r";
String lbl_K_inf_elts="elts";
String lbl_K_inf_E="E";
String lbl_K_inf_L="L";
String lbl_K_inf_O="O";

final JLabel lbl_K1 = new JLabel("K1");
final JLabel lbl_K1_r = new JLabel("r  :");
final JTextField txt_K1_r = new JTextField(txtwidth);
final JLabel lbl_K1_inf_r= new JLabel(lbl_K_inf_r);
final JLabel lbl_K1_elts = new JLabel("Elts  :");
final JTextField txt_K1_elts = new JTextField(2);
final JLabel lbl_K1_inf_elts= new JLabel(lbl_K_inf_elts);
final JLabel lbl_K1_E = new JLabel("E  :");
final JTextField txt_K1_E = new JTextField(txtwidth);
```

```
final JLabel lbl_K1_inf_E= new JLabel(lbl_K_inf_E);
final JLabel lbl_K1_L = new JLabel("L  :");
final JTextField txt_K1_L = new JTextField(txtwidth);
final JLabel lbl_K1_inf_L= new JLabel(lbl_K_inf_L);
final JLabel lbl_K1_O = new JLabel("O  :");
final JTextField txt_K1_O = new JTextField(txtwidth);
final JLabel lbl_K1_inf_O= new JLabel(lbl_K_inf_O);
final JLabel lbl_K1_Result = new JLabel("K1  :");
final JTextField txt_K1_Result = new JTextField(txtwidth);
final JLabel lbl_K1_inf_Result= new JLabel(lbl_K_inf_O);

final JLabel lbl_K2 = new JLabel("K2");
final JLabel lbl_K2_r = new JLabel("r  :");
final JTextField txt_K2_r = new JTextField(txtwidth);
final JLabel lbl_K2_inf_r= new JLabel(lbl_K_inf_r);
final JLabel lbl_K2_elts = new JLabel("Elts  :");
final JTextField txt_K2_elts = new JTextField(2);
final JLabel lbl_K2_inf_elts= new JLabel(lbl_K_inf_elts);
final JLabel lbl_K2_E = new JLabel("E  :");
final JTextField txt_K2_E = new JTextField(txtwidth);
final JLabel lbl_K2_inf_E= new JLabel(lbl_K_inf_E);
final JLabel lbl_K2_L = new JLabel("L  :");
final JTextField txt_K2_L = new JTextField(txtwidth);
final JLabel lbl_K2_inf_L= new JLabel(lbl_K_inf_L);
final JLabel lbl_K2_O = new JLabel("O  :");
final JTextField txt_K2_O = new JTextField(txtwidth);
final JLabel lbl_K2_inf_O= new JLabel(lbl_K_inf_O);
final JLabel lbl_K2_Result = new JLabel("K2  :");
final JTextField txt_K2_Result = new JTextField(txtwidth);
final JLabel lbl_K2_inf_Result= new JLabel(lbl_K_inf_O);

final JLabel lbl_K3 = new JLabel("K3");
final JLabel lbl_K3_r = new JLabel("r  :");
final JTextField txt_K3_r = new JTextField(txtwidth);
final JLabel lbl_K3_inf_r= new JLabel(lbl_K_inf_r);
final JLabel lbl_K3_elts = new JLabel("Elts  :");
final JTextField txt_K3_elts = new JTextField(2);
final JLabel lbl_K3_inf_elts= new JLabel(lbl_K_inf_elts);
final JLabel lbl_K3_E = new JLabel("E  :");
final JTextField txt_K3_E = new JTextField(txtwidth);
final JLabel lbl_K3_inf_E= new JLabel(lbl_K_inf_E);
final JLabel lbl_K3_L = new JLabel("L  :");
final JTextField txt_K3_L = new JTextField(txtwidth);
final JLabel lbl_K3_inf_L= new JLabel(lbl_K_inf_L);
final JLabel lbl_K3_O = new JLabel("O  :");
final JTextField txt_K3_O = new JTextField(txtwidth);
final JLabel lbl_K3_inf_O= new JLabel(lbl_K_inf_O);
final JLabel lbl_K3_Result = new JLabel("K3  :");
final JTextField txt_K3_Result = new JTextField(txtwidth);
final JLabel lbl_K3_inf_Result= new JLabel(lbl_K_inf_O);
```

```
final JLabel lbl_K4 = new JLabel("K4");
final JLabel lbl_K4_r = new JLabel("r  :");
final JTextField txt_K4_r = new JTextField(txtwidth);
final JLabel lbl_K4_inf_r= new JLabel(lbl_K_inf_r);
final JLabel lbl_K4_elts = new JLabel("Elts  :");
final JTextField txt_K4_elts = new JTextField(2);
final JLabel lbl_K4_inf_elts= new JLabel(lbl_K_inf_elts);
final JLabel lbl_K4_E = new JLabel("E  :");
final JTextField txt_K4_E = new JTextField(txtwidth);
final JLabel lbl_K4_inf_E= new JLabel(lbl_K_inf_E);
final JLabel lbl_K4_L = new JLabel("L  :");
final JTextField txt_K4_L = new JTextField(txtwidth);
final JLabel lbl_K4_inf_L= new JLabel(lbl_K_inf_L);
final JLabel lbl_K4_O = new JLabel("O  :");
final JTextField txt_K4_O = new JTextField(txtwidth);
final JLabel lbl_K4_inf_O= new JLabel(lbl_K_inf_O);
final JLabel lbl_K4_Result = new JLabel("K4  :");
final JTextField txt_K4_Result = new JTextField(txtwidth);
final JLabel lbl_K4_inf_Result= new JLabel(lbl_K_inf_O);


//Satuan Torsional

//Satuan J1 Propeler

final JLabel lbl_J1_Inf_R = new JLabel("-R");
final JLabel lbl_J1_Inf_m = new JLabel("-m");
final JLabel lbl_J1_Inf_P = new JLabel("-P");
final JLabel lbl_J1_Inf_r = new JLabel("-r");
final JLabel lbl_J1_Inf_L = new JLabel("-L");
final JLabel lbl_J1_Inf_Result  = new JLabel("-J1");

//Satuan J2 Pors

final JLabel lbl_J2_Inf_P = new JLabel("-P");
final JLabel lbl_J2_Inf_r= new JLabel("-r");
final JLabel lbl_J2_Inf_L= new JLabel("-L");
final JLabel lbl_J2_Inf_Result  = new JLabel("-J2");

//Satuan J3 Flywheel

final JLabel lbl_J3_Inf_P = new JLabel("-P");
final JLabel lbl_J3_Inf_r= new JLabel("-r");
final JLabel lbl_J3_Inf_L= new JLabel("-L");
final JLabel lbl_J3_Inf_Result  = new JLabel("-J3");

//Satuoan J4 Piston
final JLabel lbl_J4_Inf_P = new JLabel("-P");
final JLabel lbl_J4_Inf_r = new JLabel("-r");
final JLabel lbl_J4_Inf_L = new JLabel("-L");
final JLabel lbl_J4_Inf_m = new JLabel("-m");
```

```
final JLabel lbl_J4_Inf_rp = new JLabel("-rp");
final JLabel lbl_J4_Inf_Result  = new JLabel("-J4");



public Component createCompunent(){

        GridBagLayout gridBagLayout = new GridBagLayout();
        GridLayout gridLayout = new GridLayout();
        JPanel ordoPanel = new JPanel();
        jp.setLayout(gridBagLayout);


        //Jenis Getaran
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 20;
        gbc.gridx = 0;
        gbc.gridy = 0;
        JLabel lbljg = new JLabel("Jenis Getaran");
        jp.add(lbljg,gbc);

        gbc.gridx = 1;
        gbc.gridy = 0;
        JLabel spjg = new JLabel(":");
        spjg.setHorizontalAlignment(JLabel.CENTER);
        jp.add(spjg,gbc);


        gbc.ipadx = 10;
        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.gridwidth =2;
   jComboBox1.addItem("Longitudinal");
   jComboBox1.addItem("Torsional");
        jp.add(jComboBox1,gbc);



        //Derajat Kebebasan Longitudionaal
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 20;
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.gridwidth = 1;
        jp.add(lbl_ordo_T,gbc);

        gbc.gridx = 1;
        gbc.gridy = 1;
        sp_T.setHorizontalAlignment(JLabel.CENTER);
        jp.add(sp_T,gbc);
```

```
                gbc.ipadx = 10;
                gbc.gridx = 2;
                gbc.gridy = 1;
                jp.add(txtOrdoLongitudinal,gbc);

                gbc.gridx = 3;
                gbc.gridy = 1;

                //btnActLongitudinal.setPreferredSize(new Dimension(60, 23));
                jp.add(btnActLongitudinal, gbc);


                //Derajat Kebebasan Torsional
                gbc.gridx = 0;
                gbc.gridy = 9;
                gbc.gridwidth = 1;
                lbl_ordo_J.setVisible(false);
                jp.add(lbl_ordo_J,gbc);

                gbc.gridx = 1;

                spt_J.setHorizontalAlignment(JLabel.CENTER);
                spt_J.setVisible(false);
                jp.add(spt_J,gbc);

                gbc.gridx = 2;
                txtOrdoTorsional.setVisible(false);
                jp.add(txtOrdoTorsional,gbc);

                gbc.gridwidth = 1;
                gbc.gridx = 3;
                gbc.gridwidth = 2;
                btnActTorsional.setVisible(false);
                //btnActLongitudinal.setPreferredSize(new Dimension(60, 23));
                jp.add(btnActTorsional, gbc);


                gbc.insets = new Insets(3,5,3,3);

        //==================Torsional=========================================
==

                //==================Propeler
j1=====================================

                //==================Propeler R=============================


                gbc.gridx = 1;
                gbc.gridy = 1;
                lbl_J1.setHorizontalAlignment(JLabel.LEFT);
```

```
                jp.add(lbl_J1,gbc);

                gbc.gridx = 1;
                gbc.gridy = 2;
                gbc.gridwidth = 1;
                lbl_J1_R.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J1_R,gbc);

                gbc.gridx = 2;
                txt_J1_R.setHorizontalAlignment(JLabel.RIGHT);
                jp.add(txt_J1_R,gbc);

                gbc.gridx = 3;
                lbl_J1_Inf_R.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J1_Inf_R,gbc);

                //==================Propeler
m====================================

                gbc.gridx = 1;
                gbc.gridy = 3;
                lbl_J1_m.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J1_m,gbc);

                gbc.gridx = 2;
                txt_J1_m.setHorizontalAlignment(JLabel.RIGHT);
                jp.add(txt_J1_m,gbc);

                gbc.gridx = 3;
                lbl_J1_Inf_m.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J1_Inf_m,gbc);

                //==================Propeler
P====================================

                gbc.gridx = 1;
                gbc.gridy = 4;
                lbl_J1_P.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J1_P,gbc);

                gbc.gridx = 2;
                txt_J1_P.setHorizontalAlignment(JLabel.RIGHT);
                jp.add(txt_J1_P,gbc);

                gbc.gridx = 3;
                lbl_J1_Inf_P.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J1_Inf_P,gbc);

                //==================Propeler
r====================================
```

```
		gbc.gridx = 1;
		gbc.gridy = 5;
		lbl_J1_r.setHorizontalAlignment(JLabel.LEFT);
		jp.add(lbl_J1_r,gbc);

		gbc.gridx = 2;
		txt_J1_r.setHorizontalAlignment(JLabel.RIGHT);
		jp.add(txt_J1_r,gbc);

		gbc.gridx = 3;
		lbl_J1_Inf_r.setHorizontalAlignment(JLabel.LEFT);
		jp.add(lbl_J1_Inf_r,gbc);

		//==================Propeler L=================================

		gbc.gridx = 1;
		gbc.gridy = 6;
		lbl_J1_L.setHorizontalAlignment(JLabel.LEFT);
		jp.add(lbl_J1_L,gbc);

		gbc.gridx = 2;
		txt_J1_L.setHorizontalAlignment(JLabel.RIGHT);
		jp.add(txt_J1_L,gbc);

		gbc.gridx = 3;
		lbl_J1_Inf_L.setHorizontalAlignment(JLabel.LEFT);
		jp.add(lbl_J1_Inf_L,gbc);

		//==================Result J1=================================

		gbc.gridx = 1;
		gbc.gridy = 7;
		lbl_J1_Result.setHorizontalAlignment(JLabel.LEFT);
		jp.add(lbl_J1_Result,gbc);

		gbc.gridx = 2;
		txt_J1_Result.setHorizontalAlignment(JLabel.RIGHT);
		txt_J1_Result.setEditable(false);
		jp.add(txt_J1_Result,gbc);

		gbc.gridx = 3;
		lbl_J1_Inf_Result.setHorizontalAlignment(JLabel.LEFT);
		jp.add(lbl_J1_Inf_Result,gbc);

		//==============  J1 End
==========================================


		//====================Torsional
K1=================================
```

```
//====================K1 Torsional r==================

gbc.gridx = 4;
gbc.gridy = 1;
lbl_K1.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K1,gbc);

gbc.gridx = 4;
gbc.gridy = 2;
lbl_K1_r.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K1_r,gbc);

gbc.gridx = 5;
txt_K1_r.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K1_r,gbc);

gbc.gridx = 6;
lbl_K1_inf_r.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K1_inf_r,gbc);

//====================K1 Torsional Elastisitas==================

gbc.gridx = 4;
gbc.gridy = 3;
lbl_K1_elts.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K1_elts,gbc);

gbc.gridx = 5;
txt_K1_elts.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K1_elts,gbc);

gbc.gridx = 6;
lbl_K1_inf_elts.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K1_inf_elts,gbc);

//====================K1 Torsional E==================

gbc.gridx = 4;
gbc.gridy = 4;
lbl_K1_E.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K1_E,gbc);


gbc.gridx = 5;
txt_K1_E.setEditable(false);
txt_K1_E.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K1_E,gbc);

gbc.gridx = 6;
lbl_K1_inf_E.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K1_inf_E,gbc);
```

```
//=================== K1 Torsional L==================

gbc.gridx = 4;
gbc.gridy = 5;
lbl_K1_L.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K1_L,gbc);

gbc.gridx = 5;
txt_K1_L.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K1_L,gbc);

gbc.gridx = 6;
lbl_K1_inf_L.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K1_inf_L,gbc);

//===================K1 Torsional O==================

gbc.gridx = 4;
gbc.gridy = 6;
lbl_K1_O.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K1_O,gbc);

gbc.gridx = 5;
txt_K1_O.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K1_O,gbc);

gbc.gridx = 6;
lbl_K1_inf_O.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K1_inf_O,gbc);

//===========Result K1 Torsional================

gbc.gridx = 4;
gbc.gridy = 7;
lbl_K1_Result.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K1_Result,gbc);

gbc.gridx = 5;
txt_K1_Result.setHorizontalAlignment(JLabel.RIGHT);
txt_K1_Result.setEditable(false);
jp.add(txt_K1_Result,gbc);

gbc.gridx = 6;
lbl_K1_inf_Result.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K1_inf_Result,gbc);
//===========End K1 Torsional================
```

```
                //==================Pors
j2=======================================

                //==================Pors P=============================


                gbc.gridx = 9;
                gbc.gridy = 1;
                gbc.gridwidth = 2;
                lbl_J2.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J2,gbc);

                gbc.gridx = 9;
                gbc.gridy = 2;
                gbc.gridwidth = 1;
                lbl_J2_P.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J2_P,gbc);

                gbc.gridx = 10;
                txt_J2_P.setHorizontalAlignment(JLabel.RIGHT);
                jp.add(txt_J2_P,gbc);

                gbc.gridx = 11;
                lbl_J2_Inf_P.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J2_Inf_P,gbc);


                //==================Pors r=====================================

                gbc.gridx = 9;
                gbc.gridy = 3;
                lbl_J2_r.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J2_r,gbc);

                gbc.gridx = 10;
                txt_J2_r.setHorizontalAlignment(JLabel.RIGHT);
                jp.add(txt_J2_r,gbc);

                gbc.gridx = 3;
                lbl_J2_Inf_r.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J2_Inf_r,gbc);

                //==================Pors
L=======================================

                gbc.gridx = 9;
                gbc.gridy = 4;
                lbl_J2_L.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J2_L,gbc);

                gbc.gridx = 10;
```

```
                txt_J2_L.setHorizontalAlignment(JLabel.RIGHT);
                jp.add(txt_J2_L,gbc);

                gbc.gridx = 11;
                lbl_J2_Inf_L.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J2_Inf_L,gbc);

                //==================Result J2=================================

                gbc.gridx = 9;
                gbc.gridy = 7;
                lbl_J2_Result.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J2_Result,gbc);

                gbc.gridx = 10;
                txt_J2_Result.setHorizontalAlignment(JLabel.RIGHT);
                txt_J2_Result.setEditable(false);
                jp.add(txt_J2_Result,gbc);

                gbc.gridx = 11;
                lbl_J2_Inf_Result.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_J2_Inf_Result,gbc);

                //==============  J2 End
==========================================


                //====================Pors k2=================================

                //====================K2 Pors r==================

                gbc.gridx = 12;
                gbc.gridy = 1;
                lbl_K2.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_K2,gbc);

                gbc.gridx = 12;
                gbc.gridy = 2;
                lbl_K2_r.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_K2_r,gbc);

                gbc.gridx = 13;
                txt_K2_r.setHorizontalAlignment(JLabel.RIGHT);
                jp.add(txt_K2_r,gbc);

                gbc.gridx = 14;
                lbl_K2_inf_r.setHorizontalAlignment(JLabel.LEFT);
                jp.add(lbl_K2_inf_r,gbc);

                //==================K2 Pors Elastisitas==================
```

```
gbc.gridx = 12;
gbc.gridy = 3;
lbl_K2_elts.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K2_elts,gbc);

gbc.gridx = 13;
txt_K2_elts.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K2_elts,gbc);

gbc.gridx = 14;
lbl_K2_inf_elts.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K2_inf_elts,gbc);

//====================K2 Pors E====================

gbc.gridx = 12;
gbc.gridy = 4;
lbl_K2_E.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K2_E,gbc);


gbc.gridx = 13;
txt_K2_E.setHorizontalAlignment(JLabel.RIGHT);
txt_K2_E.setEditable(false);
jp.add(txt_K2_E,gbc);

gbc.gridx = 14;
lbl_K2_inf_E.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K2_inf_E,gbc);

//==================== K2 Pors L====================

gbc.gridx = 12;
gbc.gridy = 5;
lbl_K2_L.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K2_L,gbc);

gbc.gridx = 13;
txt_K2_L.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K2_L,gbc);

gbc.gridx = 14;
lbl_K2_inf_L.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K2_inf_L,gbc);

//====================K2 Pors O====================

gbc.gridx = 12;
gbc.gridy = 6;
lbl_K2_O.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K2_O,gbc);
```

```
gbc.gridx = 13;
txt_K2_O.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K2_O,gbc);

gbc.gridx = 14;
lbl_K2_inf_O.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K2_inf_O,gbc);

//===========Result K2 Torsional================

gbc.gridx = 12;
gbc.gridy = 7;
lbl_K2_Result.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K2_Result,gbc);

gbc.gridx =13;
txt_K2_Result.setHorizontalAlignment(JLabel.RIGHT);
txt_K2_Result.setEditable(false);
jp.add(txt_K2_Result,gbc);

gbc.gridx = 14;
lbl_K2_inf_Result.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K2_inf_Result,gbc);
//===========End K2 Pors================



//==================Flywheel j3====================

// Flywheel P

gbc.gridx = 15;
gbc.gridy = 1;
gbc.gridwidth = 2;
lbl_J3.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J3,gbc);

gbc.gridx = 15;
gbc.gridy = 2;
gbc.gridwidth = 1;
lbl_J3_P.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J3_P,gbc);

gbc.gridx = 16;
txt_J3_P.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_J3_P,gbc);

gbc.gridx = 17;
lbl_J3_Inf_P.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J3_Inf_P,gbc);
```

```
// Flywheel r

gbc.gridx = 15;
gbc.gridy = 3;
lbl_J3_r.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J3_r,gbc);

gbc.gridx = 16;
txt_J3_r.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_J3_r,gbc);

gbc.gridx = 17;
lbl_J3_Inf_r.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J3_Inf_r,gbc);

// Flywheel L

gbc.gridx = 15;
gbc.gridy = 4;
lbl_J3_L.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J3_L,gbc);

gbc.gridx = 16;
txt_J3_L.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_J3_L,gbc);

gbc.gridx = 17;
lbl_J3_Inf_L.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J3_Inf_L,gbc);

//==================Result J3=================================

gbc.gridx = 15;
gbc.gridy = 7;
lbl_J3_Result.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J3_Result,gbc);

gbc.gridx = 16;
txt_J3_Result.setHorizontalAlignment(JLabel.RIGHT);
txt_J3_Result.setEditable(false);
jp.add(txt_J3_Result,gbc);

gbc.gridx = 17;
lbl_J3_Inf_Result.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J3_Inf_Result,gbc);

//-----------------
```

```
//====================Pors k3==================================

//====================K3 Flywheel r====================

gbc.gridx = 18;
gbc.gridy = 1;
lbl_K3.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K3,gbc);

gbc.gridx = 18;
gbc.gridy = 2;
lbl_K3_r.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K3_r,gbc);

gbc.gridx = 19;
txt_K3_r.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K3_r,gbc);

gbc.gridx = 20;
lbl_K3_inf_r.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K3_inf_r,gbc);

//====================K3 Pors Elastisitas====================

gbc.gridx = 18;
gbc.gridy = 3;
lbl_K3_elts.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K3_elts,gbc);

gbc.gridx = 19;
txt_K3_elts.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K3_elts,gbc);

gbc.gridx = 20;
lbl_K3_inf_elts.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K3_inf_elts,gbc);

//====================K3 Pors E====================

gbc.gridx = 18;
gbc.gridy = 4;
lbl_K3_E.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K3_E,gbc);


gbc.gridx = 19;
txt_K3_E.setHorizontalAlignment(JLabel.RIGHT);
txt_K3_E.setEditable(false);
jp.add(txt_K3_E,gbc);

gbc.gridx = 20;
```

```
lbl_K3_inf_E.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K3_inf_E,gbc);

//=================== K3 Pors L==================

gbc.gridx = 18;
gbc.gridy = 5;
lbl_K3_L.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K3_L,gbc);

gbc.gridx = 19;
txt_K3_L.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K3_L,gbc);

gbc.gridx = 20;
lbl_K3_inf_L.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K3_inf_L,gbc);

//==================K3 Pors O==================

gbc.gridx = 18;
gbc.gridy = 6;
lbl_K3_O.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K3_O,gbc);

gbc.gridx = 19;
txt_K3_O.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K3_O,gbc);

gbc.gridx = 20;
lbl_K3_inf_O.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K3_inf_O,gbc);

//===========Result K3 Torsional================

gbc.gridx = 18;
gbc.gridy = 7;
lbl_K3_Result.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K3_Result,gbc);

gbc.gridx =19;
txt_K3_Result.setHorizontalAlignment(JLabel.RIGHT);
txt_K3_Result.setEditable(false);
jp.add(txt_K3_Result,gbc);

gbc.gridx = 20;
lbl_K3_inf_Result.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K3_inf_Result,gbc);
//===========End K3 Pors================
```

```
//=================piston j4======================

// piston P

gbc.gridx = 21;
gbc.gridy = 1;
gbc.gridwidth = 2;
lbl_J4.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J4,gbc);

gbc.gridx = 21;
gbc.gridy = 2;
gbc.gridwidth = 1;
lbl_J4_P .setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J4_P ,gbc);

gbc.gridx = 22;
txt_J4_P.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_J4_P,gbc);

gbc.gridx = 23;
lbl_J4_Inf_P.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J4_Inf_P,gbc);

// piston r

gbc.gridx = 21;
gbc.gridy = 3;
lbl_J4_r.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J4_r,gbc);

gbc.gridx = 22;
txt_J4_r.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_J4_r,gbc);

gbc.gridx = 23;
lbl_J4_Inf_r.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J4_Inf_r,gbc);

// piston L

gbc.gridx = 21;
gbc.gridy = 4;
lbl_J4_L.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J4_L,gbc);

gbc.gridx = 22;
txt_J4_L.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_J4_L,gbc);

gbc.gridx = 23;
```

```
lbl_J4_Inf_L.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J4_Inf_L,gbc);



// piston m

gbc.gridx = 21;
gbc.gridy = 5;
lbl_J4_m.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J4_m,gbc);

gbc.gridx = 22;
txt_J4_m.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_J4_m,gbc);

gbc.gridx = 23;
lbl_J4_Inf_m.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J4_Inf_m,gbc);

// piston rp

gbc.gridx = 21;
gbc.gridy = 6;
lbl_J4_rp.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J4_rp,gbc);

gbc.gridx = 22;
txt_J4_rp.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_J4_rp,gbc);

gbc.gridx = 23;
lbl_J4_Inf_rp.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J4_Inf_rp,gbc);

//================== Result J4====================
gbc.gridx = 21;
gbc.gridy = 7;
lbl_J4_Result.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J4_Result,gbc);

gbc.gridx = 22;
txt_J4_Result.setHorizontalAlignment(JLabel.RIGHT);
txt_J4_Result.setEditable(false);
jp.add(txt_J4_Result,gbc);

gbc.gridx = 23;
lbl_J4_Inf_Result.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_J4_Inf_Result,gbc);

//------------------
```

```
//====================Piston k4==================================

//====================K4 Piston r==================

gbc.gridx = 24;
gbc.gridy = 1;
lbl_K4.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K4,gbc);

gbc.gridx = 24;
gbc.gridy = 2;
lbl_K4_r.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K4_r,gbc);

gbc.gridx = 25;
txt_K4_r.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K4_r,gbc);

gbc.gridx = 26;
lbl_K4_inf_r.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K4_inf_r,gbc);

//====================K4 Pors Elastisitas==================

gbc.gridx = 24;
gbc.gridy = 3;
lbl_K4_elts.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K4_elts,gbc);

gbc.gridx = 25;
txt_K4_elts.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K4_elts,gbc);

gbc.gridx = 26;
lbl_K4_inf_elts.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K4_inf_elts,gbc);

//====================K4 Pors E==================

gbc.gridx = 24;
gbc.gridy = 4;
lbl_K4_E.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K4_E,gbc);


gbc.gridx = 25;
txt_K4_E.setHorizontalAlignment(JLabel.RIGHT);
txt_K4_E.setEditable(false);
jp.add(txt_K4_E,gbc);
```

```
gbc.gridx = 26;
lbl_K4_inf_E.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K4_inf_E,gbc);

//=================== K4 Pors L==================

gbc.gridx = 24;
gbc.gridy = 5;
lbl_K4_L.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K4_L,gbc);

gbc.gridx = 25;
txt_K4_L.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K4_L,gbc);

gbc.gridx = 26;
lbl_K4_inf_L.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K4_inf_L,gbc);

//==================K4 Pors O==================

gbc.gridx = 24;
gbc.gridy = 6;
lbl_K4_O.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K4_O,gbc);

gbc.gridx = 25;
txt_K4_O.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txt_K4_O,gbc);

gbc.gridx = 26;
lbl_K4_inf_O.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K4_inf_O,gbc);

//===========Result K4 Torsional================

gbc.gridx = 24;
gbc.gridy = 7;
lbl_K4_Result.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K4_Result,gbc);

gbc.gridx =25;
txt_K4_Result.setHorizontalAlignment(JLabel.RIGHT);
txt_K4_Result.setEditable(false);
jp.add(txt_K4_Result,gbc);

gbc.gridx = 26;
lbl_K4_inf_Result.setHorizontalAlignment(JLabel.LEFT);
jp.add(lbl_K4_inf_Result,gbc);
//===========End K3 Pors================
```

```
		showHideElementTorsional(false);

		final Gen gen = new Gen();

		//Calculate J1
		txt_J1_R.addKeyListener(new KeyAdapter() {
			public void keyReleased(KeyEvent e) {
				if(!txt_J1_R.getText().isEmpty() && !txt_J1_m.getText().isEmpty()
&& !txt_J1_P.getText().isEmpty() && !txt_J1_r.getText().isEmpty()&& !txt_J1_L.getText().isEmpty()){

txt_J1_Result.setText(gen.torsionalPropelerJ(Double.valueOf(txt_J1_R.getText()),Double.valueOf(txt_J1
_m.getText()),Double.valueOf(txt_J1_P.getText()),Double.valueOf(txt_J1_r.getText()),Double.valueOf(t
xt_J1_L.getText())));
				}

			}
			});
		txt_J1_m.addKeyListener(new KeyAdapter() {
			public void keyReleased(KeyEvent e) {
				if(!txt_J1_R.getText().isEmpty() && !txt_J1_m.getText().isEmpty()
&& !txt_J1_P.getText().isEmpty() && !txt_J1_r.getText().isEmpty()&& !txt_J1_L.getText().isEmpty()){

txt_J1_Result.setText(gen.torsionalPropelerJ(Double.valueOf(txt_J1_R.getText()),Double.valueOf(txt_J1
_m.getText()),Double.valueOf(txt_J1_P.getText()),Double.valueOf(txt_J1_r.getText()),Double.valueOf(t
xt_J1_L.getText())));
				}

			}
			});
		txt_J1_P.addKeyListener(new KeyAdapter() {
			public void keyReleased(KeyEvent e) {
				if(!txt_J1_R.getText().isEmpty() && !txt_J1_m.getText().isEmpty()
&& !txt_J1_P.getText().isEmpty() && !txt_J1_r.getText().isEmpty()&& !txt_J1_L.getText().isEmpty()){

txt_J1_Result.setText(gen.torsionalPropelerJ(Double.valueOf(txt_J1_R.getText()),Double.valueOf(txt_J1
_m.getText()),Double.valueOf(txt_J1_P.getText()),Double.valueOf(txt_J1_r.getText()),Double.valueOf(t
xt_J1_L.getText())));
				}

			}
			});
		txt_J1_r.addKeyListener(new KeyAdapter() {
			public void keyReleased(KeyEvent e) {
				if(!txt_J1_R.getText().isEmpty() && !txt_J1_m.getText().isEmpty()
&& !txt_J1_P.getText().isEmpty() && !txt_J1_r.getText().isEmpty()&& !txt_J1_L.getText().isEmpty()){

txt_J1_Result.setText(gen.torsionalPropelerJ(Double.valueOf(txt_J1_R.getText()),Double.valueOf(txt_J1
_m.getText()),Double.valueOf(txt_J1_P.getText()),Double.valueOf(txt_J1_r.getText()),Double.valueOf(t
xt_J1_L.getText())));
				}
```

```
		}
			});
		 txt_J1_L.addKeyListener(new KeyAdapter() {
			public void keyReleased(KeyEvent e) {
				if(!txt_J1_R.getText().isEmpty() && !txt_J1_m.getText().isEmpty()
&& !txt_J1_P.getText().isEmpty() && !txt_J1_r.getText().isEmpty()&& !txt_J1_L.getText().isEmpty()){

txt_J1_Result.setText(gen.torsionalPropelerJ(Double.valueOf(txt_J1_R.getText()),Double.valueOf(txt_J1
_m.getText()),Double.valueOf(txt_J1_P.getText()),Double.valueOf(txt_J1_r.getText()),Double.valueOf(t
xt_J1_L.getText())));
				}

		}
			});

		 //Calculate J2
		 txt_J2_P.addKeyListener(new KeyAdapter() {
			public void keyReleased(KeyEvent e) {
				if(!txt_J2_P.getText().isEmpty() && !txt_J2_r.getText().isEmpty() &&
!txt_J2_L.getText().isEmpty()){

txt_J2_Result.setText(gen.torsionalPorsJ(Double.valueOf(txt_J2_P.getText()),Double.valueOf(txt_J2_r.g
etText()),Double.valueOf(txt_J2_L.getText())));
				}

		}
			});
		 txt_J2_r.addKeyListener(new KeyAdapter() {
			public void keyReleased(KeyEvent e) {
				if(!txt_J2_P.getText().isEmpty() && !txt_J2_r.getText().isEmpty() &&
!txt_J2_L.getText().isEmpty()){

txt_J2_Result.setText(gen.torsionalPorsJ(Double.valueOf(txt_J2_P.getText()),Double.valueOf(txt_J2_r.g
etText()),Double.valueOf(txt_J2_L.getText())));
				}

		}
			});
		 txt_J2_L.addKeyListener(new KeyAdapter() {
			public void keyReleased(KeyEvent e) {
				if(!txt_J2_P.getText().isEmpty() && !txt_J2_r.getText().isEmpty() &&
!txt_J2_L.getText().isEmpty()){

txt_J2_Result.setText(gen.torsionalPorsJ(Double.valueOf(txt_J2_P.getText()),Double.valueOf(txt_J2_r.g
etText()),Double.valueOf(txt_J2_L.getText())));
				}

		}
			});
```

```
		//Calculate J3
		txt_J3_P.addKeyListener(new KeyAdapter() {
			public void keyReleased(KeyEvent e) {
				if(!txt_J3_P.getText().isEmpty() && !txt_J3_r.getText().isEmpty() &&
!txt_J3_L.getText().isEmpty()){

txt_J3_Result.setText(gen.torsionalPorsJ(Double.valueOf(txt_J3_P.getText()),Double.valueOf(txt_J3_r.g
etText()),Double.valueOf(txt_J3_L.getText())));
				}

		}
			});
		txt_J3_r.addKeyListener(new KeyAdapter() {
			public void keyReleased(KeyEvent e) {
				if(!txt_J3_P.getText().isEmpty() && !txt_J3_r.getText().isEmpty() &&
!txt_J3_L.getText().isEmpty()){

txt_J3_Result.setText(gen.torsionalPorsJ(Double.valueOf(txt_J3_P.getText()),Double.valueOf(txt_J3_r.g
etText()),Double.valueOf(txt_J3_L.getText())));
				}

		}
			});
		txt_J3_L.addKeyListener(new KeyAdapter() {
			public void keyReleased(KeyEvent e) {
				if(!txt_J3_P.getText().isEmpty() && !txt_J3_r.getText().isEmpty() &&
!txt_J3_L.getText().isEmpty()){

txt_J3_Result.setText(gen.torsionalPorsJ(Double.valueOf(txt_J3_P.getText()),Double.valueOf(txt_J3_r.g
etText()),Double.valueOf(txt_J3_L.getText())));
				}

		}
			});

		//Calculate J4
		txt_J4_P.addKeyListener(new KeyAdapter() {
			public void keyReleased(KeyEvent e) {
				if(!txt_J4_P.getText().isEmpty() && !txt_J4_r.getText().isEmpty() &&
!txt_J4_L.getText().isEmpty() && !txt_J4_m.getText().isEmpty()&& !txt_J4_rp.getText().isEmpty()){

txt_J4_Result.setText(gen.torsionalPistonJ(Double.valueOf(txt_J4_P.getText()),Double.valueOf(txt_J4_r.
getText()),Double.valueOf(txt_J4_L.getText()),Double.valueOf(txt_J4_m.getText()),Double.valueOf(txt_
J4_rp.getText())));
				}

		}
			});

		txt_J4_r.addKeyListener(new KeyAdapter() {
			public void keyReleased(KeyEvent e) {
```

```
                                        if(!txt_J4_P.getText().isEmpty() && !txt_J4_r.getText().isEmpty() &&
!txt_J4_L.getText().isEmpty() && !txt_J4_m.getText().isEmpty()&& !txt_J4_rp.getText().isEmpty()){

txt_J4_Result.setText(gen.torsionalPistonJ(Double.valueOf(txt_J4_P.getText()),Double.valueOf(txt_J4_r.
getText()),Double.valueOf(txt_J4_L.getText()),Double.valueOf(txt_J4_m.getText()),Double.valueOf(txt_
J4_rp.getText())));
                                        }

                        }
                                });
                        txt_J4_L.addKeyListener(new KeyAdapter() {
                                public void keyReleased(KeyEvent e) {
                                        if(!txt_J4_P.getText().isEmpty() && !txt_J4_r.getText().isEmpty() &&
!txt_J4_L.getText().isEmpty() && !txt_J4_m.getText().isEmpty()&& !txt_J4_rp.getText().isEmpty()){

txt_J4_Result.setText(gen.torsionalPistonJ(Double.valueOf(txt_J4_P.getText()),Double.valueOf(txt_J4_r.
getText()),Double.valueOf(txt_J4_L.getText()),Double.valueOf(txt_J4_m.getText()),Double.valueOf(txt_
J4_rp.getText())));
                                        }

                        }
                                });
                        txt_J4_m.addKeyListener(new KeyAdapter() {
                                public void keyReleased(KeyEvent e) {
                                        if(!txt_J4_P.getText().isEmpty() && !txt_J4_r.getText().isEmpty() &&
!txt_J4_L.getText().isEmpty() && !txt_J4_m.getText().isEmpty()&& !txt_J4_rp.getText().isEmpty()){

txt_J4_Result.setText(gen.torsionalPistonJ(Double.valueOf(txt_J4_P.getText()),Double.valueOf(txt_J4_r.
getText()),Double.valueOf(txt_J4_L.getText()),Double.valueOf(txt_J4_m.getText()),Double.valueOf(txt_
J4_rp.getText())));
                                        }

                        }
                                });
                        txt_J4_rp.addKeyListener(new KeyAdapter() {
                                public void keyReleased(KeyEvent e) {
                                        if(!txt_J4_P.getText().isEmpty() && !txt_J4_r.getText().isEmpty() &&
!txt_J4_L.getText().isEmpty() && !txt_J4_m.getText().isEmpty()&& !txt_J4_rp.getText().isEmpty()){

txt_J4_Result.setText(gen.torsionalPistonJ(Double.valueOf(txt_J4_P.getText()),Double.valueOf(txt_J4_r.
getText()),Double.valueOf(txt_J4_L.getText()),Double.valueOf(txt_J4_m.getText()),Double.valueOf(txt_
J4_rp.getText())));
                                        }

                        }
                                });

                        //Calculate K1
                        txt_K1_elts.addKeyListener(new KeyAdapter() {
                                public void keyReleased(KeyEvent e) {
                                        if(!txt_K1_elts.getText().isEmpty()){
```

```
txt_K1_E.setText(gen.torsionalKEval(Double.valueOf(txt_K1_elts.getText())));
                                        }

                    }
                            });

                     txt_K1_r.addKeyListener(new KeyAdapter() {
                            public void keyReleased(KeyEvent e) {
                                    if(!txt_K1_r.getText().isEmpty() && !txt_K1_elts.getText().isEmpty()
&& !txt_K1_E.getText().isEmpty() && !txt_K1_L.getText().isEmpty()&&
!txt_K1_O.getText().isEmpty()){

txt_K1_Result.setText(gen.torsionalK(Double.valueOf(txt_K1_r.getText()),Double.valueOf(txt_K1_E.ge
tText()),Double.valueOf(txt_K1_L.getText()),Double.valueOf(txt_K1_O.getText())));
                                        }

                    }
                            });

                     txt_K1_elts.addKeyListener(new KeyAdapter() {
                             public void keyReleased(KeyEvent e) {
                                    if(!txt_K1_r.getText().isEmpty() && !txt_K1_elts.getText().isEmpty()
&& !txt_K1_E.getText().isEmpty() && !txt_K1_L.getText().isEmpty()&&
!txt_K1_O.getText().isEmpty()){

txt_K1_Result.setText(gen.torsionalK(Double.valueOf(txt_K1_r.getText()),Double.valueOf(txt_K1_E.ge
tText()),Double.valueOf(txt_K1_L.getText()),Double.valueOf(txt_K1_O.getText())));
                                        }

                    }
                            });

                     txt_K1_E.addKeyListener(new KeyAdapter() {
                             public void keyReleased(KeyEvent e) {
                                    if(!txt_K1_r.getText().isEmpty() && !txt_K1_elts.getText().isEmpty()
&& !txt_K1_E.getText().isEmpty() && !txt_K1_L.getText().isEmpty()&&
!txt_K1_O.getText().isEmpty()){

txt_K1_Result.setText(gen.torsionalK(Double.valueOf(txt_K1_r.getText()),Double.valueOf(txt_K1_E.ge
tText()),Double.valueOf(txt_K1_L.getText()),Double.valueOf(txt_K1_O.getText())));
                                        }

                    }
                            });
                     txt_K1_L.addKeyListener(new KeyAdapter() {
                             public void keyReleased(KeyEvent e) {
                                    if(!txt_K1_r.getText().isEmpty() && !txt_K1_elts.getText().isEmpty()
&& !txt_K1_E.getText().isEmpty() && !txt_K1_L.getText().isEmpty()&&
!txt_K1_O.getText().isEmpty()){
```

```
txt_K1_Result.setText(gen.torsionalK(Double.valueOf(txt_K1_r.getText()),Double.valueOf(txt_K1_E.ge
tText()),Double.valueOf(txt_K1_L.getText()),Double.valueOf(txt_K1_O.getText())));
                                        }

                        }
                                });
                         txt_K1_O.addKeyListener(new KeyAdapter() {
                                public void keyReleased(KeyEvent e) {
                                        if(!txt_K1_r.getText().isEmpty() && !txt_K1_elts.getText().isEmpty()
&& !txt_K1_E.getText().isEmpty() && !txt_K1_L.getText().isEmpty()&&
!txt_K1_O.getText().isEmpty()){

txt_K1_Result.setText(gen.torsionalK(Double.valueOf(txt_K1_r.getText()),Double.valueOf(txt_K1_E.ge
tText()),Double.valueOf(txt_K1_L.getText()),Double.valueOf(txt_K1_O.getText())));
                                        }

                        }
                                });

                         //Calculate K2
                                        txt_K2_elts.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K2_elts.getText().isEmpty()){

txt_K2_E.setText(gen.torsionalKEval(Double.valueOf(txt_K2_elts.getText())));
                                                        }

                                        }
                                                });

                                        txt_K2_r.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K2_r.getText().isEmpty() &&
!txt_K2_elts.getText().isEmpty() && !txt_K2_E.getText().isEmpty() &&
!txt_K2_L.getText().isEmpty()&& !txt_K2_O.getText().isEmpty()){

txt_K2_Result.setText(gen.torsionalK(Double.valueOf(txt_K2_r.getText()),Double.valueOf(txt_K2_E.ge
tText()),Double.valueOf(txt_K2_L.getText()),Double.valueOf(txt_K2_O.getText())));
                                                        }

                                        }
                                                });

                                        txt_K2_elts.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K2_r.getText().isEmpty() &&
!txt_K2_elts.getText().isEmpty() && !txt_K2_E.getText().isEmpty() &&
!txt_K2_L.getText().isEmpty()&& !txt_K2_O.getText().isEmpty()){
```

```
txt_K2_Result.setText(gen.torsionalK(Double.valueOf(txt_K2_r.getText()),Double.valueOf(txt_K2_E.ge
tText()),Double.valueOf(txt_K2_L.getText()),Double.valueOf(txt_K2_O.getText())));
                                                }

                                        }
                                                });

                                        txt_K2_E.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K2_r.getText().isEmpty() &&
!txt_K2_elts.getText().isEmpty() && !txt_K2_E.getText().isEmpty() &&
!txt_K2_L.getText().isEmpty()&& !txt_K2_O.getText().isEmpty()){

txt_K2_Result.setText(gen.torsionalK(Double.valueOf(txt_K2_r.getText()),Double.valueOf(txt_K2_E.ge
tText()),Double.valueOf(txt_K2_L.getText()),Double.valueOf(txt_K2_O.getText())));
                                                }

                                        }
                                                });
                                        txt_K2_L.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K2_r.getText().isEmpty() &&
!txt_K2_elts.getText().isEmpty() && !txt_K2_E.getText().isEmpty() &&
!txt_K2_L.getText().isEmpty()&& !txt_K2_O.getText().isEmpty()){

txt_K2_Result.setText(gen.torsionalK(Double.valueOf(txt_K2_r.getText()),Double.valueOf(txt_K2_E.ge
tText()),Double.valueOf(txt_K2_L.getText()),Double.valueOf(txt_K2_O.getText())));
                                                }

                                        }
                                                });
                                        txt_K2_O.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K2_r.getText().isEmpty() &&
!txt_K2_elts.getText().isEmpty() && !txt_K2_E.getText().isEmpty() &&
!txt_K2_L.getText().isEmpty()&& !txt_K2_O.getText().isEmpty()){

txt_K2_Result.setText(gen.torsionalK(Double.valueOf(txt_K2_r.getText()),Double.valueOf(txt_K2_E.ge
tText()),Double.valueOf(txt_K2_L.getText()),Double.valueOf(txt_K2_O.getText())));
                                                }

                                        }
                                                });
                                        //Calculate K3
                                        txt_K3_elts.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K3_elts.getText().isEmpty()){

txt_K3_E.setText(gen.torsionalKEval(Double.valueOf(txt_K3_elts.getText())));
                                                }
```

```
                                        }
                                                });

                                         txt_K3_r.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K3_r.getText().isEmpty() &&
!txt_K3_elts.getText().isEmpty() && !txt_K3_E.getText().isEmpty() &&
!txt_K3_L.getText().isEmpty()&& !txt_K3_O.getText().isEmpty()){

txt_K3_Result.setText(gen.torsionalK(Double.valueOf(txt_K3_r.getText()),Double.valueOf(txt_K3_E.ge
tText()),Double.valueOf(txt_K3_L.getText()),Double.valueOf(txt_K3_O.getText())));
                                                        }

                                        }
                                                });

                                         txt_K3_elts.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K3_r.getText().isEmpty() &&
!txt_K3_elts.getText().isEmpty() && !txt_K3_E.getText().isEmpty() &&
!txt_K3_L.getText().isEmpty()&& !txt_K3_O.getText().isEmpty()){

txt_K3_Result.setText(gen.torsionalK(Double.valueOf(txt_K3_r.getText()),Double.valueOf(txt_K3_E.ge
tText()),Double.valueOf(txt_K3_L.getText()),Double.valueOf(txt_K3_O.getText())));
                                                        }

                                        }
                                                });

                                         txt_K3_E.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K3_r.getText().isEmpty() &&
!txt_K3_elts.getText().isEmpty() && !txt_K3_E.getText().isEmpty() &&
!txt_K3_L.getText().isEmpty()&& !txt_K3_O.getText().isEmpty()){

txt_K3_Result.setText(gen.torsionalK(Double.valueOf(txt_K3_r.getText()),Double.valueOf(txt_K3_E.ge
tText()),Double.valueOf(txt_K3_L.getText()),Double.valueOf(txt_K3_O.getText())));
                                                        }

                                        }
                                                });
                                         txt_K3_L.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K3_r.getText().isEmpty() &&
!txt_K3_elts.getText().isEmpty() && !txt_K3_E.getText().isEmpty() &&
!txt_K3_L.getText().isEmpty()&& !txt_K3_O.getText().isEmpty()){

txt_K3_Result.setText(gen.torsionalK(Double.valueOf(txt_K3_r.getText()),Double.valueOf(txt_K3_E.ge
tText()),Double.valueOf(txt_K3_L.getText()),Double.valueOf(txt_K3_O.getText())));
                                                        }
```

```
                                        }
                                                });
                                         txt_K3_O.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K3_r.getText().isEmpty() &&
!txt_K3_elts.getText().isEmpty() && !txt_K3_E.getText().isEmpty() &&
!txt_K3_L.getText().isEmpty()&& !txt_K3_O.getText().isEmpty()){

txt_K3_Result.setText(gen.torsionalK(Double.valueOf(txt_K3_r.getText()),Double.valueOf(txt_K3_E.ge
tText()),Double.valueOf(txt_K3_L.getText()),Double.valueOf(txt_K3_O.getText())));
                                                        }

                                        }
                                                });
                                         //Calculate K4
                                         txt_K4_elts.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K4_elts.getText().isEmpty()){

txt_K4_E.setText(gen.torsionalKEval(Double.valueOf(txt_K4_elts.getText())));
                                                        }

                                        }
                                                });

                                         txt_K4_r.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K4_r.getText().isEmpty() &&
!txt_K4_elts.getText().isEmpty() && !txt_K4_E.getText().isEmpty() &&
!txt_K4_L.getText().isEmpty()&& !txt_K4_O.getText().isEmpty()){

txt_K4_Result.setText(gen.torsionalK(Double.valueOf(txt_K4_r.getText()),Double.valueOf(txt_K4_E.ge
tText()),Double.valueOf(txt_K4_L.getText()),Double.valueOf(txt_K4_O.getText())));
                                                        }

                                        }
                                                });

                                         txt_K4_elts.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K4_r.getText().isEmpty() &&
!txt_K4_elts.getText().isEmpty() && !txt_K4_E.getText().isEmpty() &&
!txt_K4_L.getText().isEmpty()&& !txt_K4_O.getText().isEmpty()){

txt_K4_Result.setText(gen.torsionalK(Double.valueOf(txt_K4_r.getText()),Double.valueOf(txt_K4_E.ge
tText()),Double.valueOf(txt_K4_L.getText()),Double.valueOf(txt_K4_O.getText())));
                                                        }

                                        }
                                                });
```

```
                                        txt_K4_E.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K4_r.getText().isEmpty() &&
!txt_K4_elts.getText().isEmpty() && !txt_K4_E.getText().isEmpty() &&
!txt_K4_L.getText().isEmpty()&& !txt_K4_O.getText().isEmpty()){

txt_K4_Result.setText(gen.torsionalK(Double.valueOf(txt_K4_r.getText()),Double.valueOf(txt_K4_E.ge
tText()),Double.valueOf(txt_K4_L.getText()),Double.valueOf(txt_K4_O.getText())));
                                                        }

                                        }
                                                });
                                        txt_K4_L.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K4_r.getText().isEmpty() &&
!txt_K4_elts.getText().isEmpty() && !txt_K4_E.getText().isEmpty() &&
!txt_K4_L.getText().isEmpty()&& !txt_K4_O.getText().isEmpty()){

txt_K4_Result.setText(gen.torsionalK(Double.valueOf(txt_K4_r.getText()),Double.valueOf(txt_K4_E.ge
tText()),Double.valueOf(txt_K4_L.getText()),Double.valueOf(txt_K4_O.getText())));
                                                        }

                                        }
                                                });
                                        txt_K4_O.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txt_K4_r.getText().isEmpty() &&
!txt_K4_elts.getText().isEmpty() && !txt_K4_E.getText().isEmpty() &&
!txt_K4_L.getText().isEmpty()&& !txt_K4_O.getText().isEmpty()){

txt_K4_Result.setText(gen.torsionalK(Double.valueOf(txt_K4_r.getText()),Double.valueOf(txt_K4_E.ge
tText()),Double.valueOf(txt_K4_L.getText()),Double.valueOf(txt_K4_O.getText())));
                                                        }

                                        }
                                                });

                        jComboBox1.addActionListener (new ActionListener () {
                            public void actionPerformed(ActionEvent e) {
                                Object cmboitem = jComboBox1.getSelectedItem();
                                        if(cmboitem.equals("Longitudinal")){

                                                showHideElementLongitudinal(true);
                                                showHideElementTorsional(false);

                                        }else{
                                                showHideElementTorsional(true);
                                                showHideElementLongitudinal(false);
                                        }
                        }
```

```
});

btnActLongitudinal.addActionListener(new ActionListener() {

	@Override
	public void actionPerformed(ActionEvent e) {
		//showGrapLongitudinal();
		final Gen gen = new Gen();
		int ordo = Integer.valueOf(txtOrdoLongitudinal.getText());
		Object cmboitem = jComboBox1.getSelectedItem();
		if(cmboitem.equals("Longitudinal")){
			notasi = "m";
		}else{
			notasi = "j";
		}

		gbc.ipadx = 20;
		gbc.gridx = 0;
		gbc.gridy = 2;
		JLabel lblk = new JLabel("");
		jp.add(lblk, gbc);

		gbc.ipadx = 20;
		gbc.gridx = 0;
		gbc.gridy = 3;
		JLabel lblm = new JLabel("");
		jp.add(lblm, gbc);


		int klabel=2;
		int gridy=2;
		for (int i = 1; i <= ordo; i++) {

			//r

			gbc.gridx = 0;
			gbc.gridy = gridy;
			JLabel lblrn = new JLabel("r"+i);
			lblrn.setHorizontalAlignment(JLabel.RIGHT);
			jp.add(lblrn,gbc);

			gbc.gridx = 1;
			gbc.gridy = gridy;
			JLabel lblrn1 = new JLabel(":");
			lblrn1.setHorizontalAlignment(JLabel.CENTER);
			jp.add(lblrn1,gbc);

			gbc.gridx = 2;
			gbc.gridy = gridy;
			final JTextField txtr = new JTextField(3);
			txtr.setHorizontalAlignment(JLabel.RIGHT);
```

```
jp.add(txtr,gbc);

//Elastisitas

gbc.gridx = 3;
gbc.gridy = gridy;
JLabel lblEl = new JLabel("elts"+i);
lblEl.setHorizontalAlignment(JLabel.RIGHT);
jp.add(lblEl,gbc);

gbc.gridx = 4;
gbc.gridy = gridy;
JLabel lblEln1 = new JLabel(":");
lblEln1.setHorizontalAlignment(JLabel.CENTER);
jp.add(lblEln1,gbc);

gbc.gridx = 5;
gbc.gridy = gridy;
final JTextField txtEl = new JTextField(3);
txtEl.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txtEl,gbc);

//E

gbc.gridx = 6;
gbc.gridy = gridy;
JLabel lblEn = new JLabel("E"+i);
lblEn.setHorizontalAlignment(JLabel.RIGHT);
jp.add(lblEn,gbc);

gbc.gridx = 7;
gbc.gridy = gridy;
JLabel lblEn1 = new JLabel(":");
lblEn1.setHorizontalAlignment(JLabel.CENTER);
jp.add(lblEn1,gbc);

gbc.gridx = 8;
gbc.gridy = gridy;
final JTextField txtE = new JTextField(3);
txtE.setEditable(false);
txtE.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txtE,gbc);

//L

gbc.gridx = 9;
gbc.gridy = gridy;
JLabel lblLn = new JLabel("L"+i);
lblLn.setHorizontalAlignment(JLabel.RIGHT);
jp.add(lblLn,gbc);
```

```
gbc.gridx = 10;
gbc.gridy = gridy;
JLabel lblLn1 = new JLabel(":");
lblLn1.setHorizontalAlignment(JLabel.CENTER);
jp.add(lblLn1,gbc);

gbc.gridx = 11;
gbc.gridy = gridy;
final JTextField txtL = new JTextField(3);
txtL.setHorizontalAlignment(JLabel.RIGHT);
jp.add(txtL,gbc);

//k

gbc.gridx = 12;
gbc.gridy = gridy;
JLabel lblkn = new JLabel("k"+i);
lblkn.setHorizontalAlignment(JLabel.RIGHT);
jp.add(lblkn,gbc);

gbc.gridx = 13;
gbc.gridy = gridy;
JLabel lblkn1 = new JLabel(":");
lblkn1.setHorizontalAlignment(JLabel.CENTER);
jp.add(lblkn1,gbc);

gbc.gridx = 14;
gbc.gridy = gridy;
final JTextField txtk = new JTextField(3);
txtk.setEditable(false);
//txtk.setBackground(getBackground().WHITE);
txtk.setHorizontalAlignment(JLabel.RIGHT);
txtvalk.add(txtk);
jp.add(txtk,gbc);

// w

gbc.gridx = 15;
gbc.gridy = gridy;
JLabel lblwn = new JLabel("w"+i);
lblwn.setHorizontalAlignment(JLabel.RIGHT);
jp.add(lblwn,gbc);

gbc.gridx = 16;
gbc.gridy = gridy;
JLabel lblwn2 = new JLabel(":");
lblwn2.setHorizontalAlignment(JLabel.CENTER);
jp.add(lblwn2,gbc);

gbc.gridx = 17;
gbc.gridy = gridy;
```

```
                                        final JTextField txtw = new JTextField(3);
                                        txtw.setHorizontalAlignment(JLabel.RIGHT);
                                        jp.add(txtw,gbc);


                                        //m

                                        gbc.gridx = 18;
                                        gbc.gridy = gridy;
                                        JLabel lblmn = new JLabel(notasi+i);
                                        lblmn.setHorizontalAlignment(JLabel.RIGHT);
                                        jp.add(lblmn,gbc);

                                        gbc.gridx = 19;
                                        gbc.gridy = gridy;
                                        JLabel lblkn2 = new JLabel(":");
                                        lblkn2.setHorizontalAlignment(JLabel.CENTER);
                                        jp.add(lblkn2,gbc);

                                        gbc.gridx = 20;
                                        gbc.gridy = gridy;
                                        final JTextField txtm = new JTextField(3);
                                        txtm.setHorizontalAlignment(JLabel.RIGHT);
                                        txtvalm.add(txtm);
                                        jp.add(txtm,gbc);

                                        txtr.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txtr.getText().isEmpty() &&
!txtE.getText().isEmpty() && !txtL.getText().isEmpty()){

txtk.setText(gen.longitudionalK(Double.valueOf(txtr.getText()),Double.valueOf(txtE.getText()),Double.v
alueOf(txtL.getText())));
                                                        }

                                        }
                                                });

                                        txtEl.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txtEl.getText().isEmpty()){

txtE.setText(gen.longitudionalKEval(Double.valueOf(txtEl.getText())));
                                                        }
                                        }
                                                });
                                        txtL.addKeyListener(new KeyAdapter() {
                                                public void keyReleased(KeyEvent e) {
                                                        if(!txtr.getText().isEmpty() &&
!txtE.getText().isEmpty() && !txtL.getText().isEmpty()){
```

```
txtk.setText(gen.longitudionalK(Double.valueOf(txtr.getText()),Double.valueOf(txtE.getText()),Double.v
alueOf(txtL.getText())));
                                                    }
                                                }
                                                    });

                                                    txtw.addKeyListener(new KeyAdapter() {
                                                     public void keyReleased(KeyEvent e) {

txtm.setText(gen.longitudionalM(Double.valueOf(txtw.getText())));
                                                }
                                                    });


                                                    jp.revalidate();
                                                    validate();
                                                    klabel=klabel+2;
                                                    gridy=gridy+2;
                                                }

                                                gbc.gridx = 0;
                                                gbc.gridy = klabel;
                                                JLabel lblexpto = new JLabel("Export to");
                                                lblexpto.setHorizontalAlignment(JLabel.RIGHT);
                                                jp.add(lblexpto,gbc);

                                                gbc.gridx = 1;
                                                gbc.gridy = klabel;
                                                JLabel lblexpto1 = new JLabel(":");
                                                lblexpto1.setHorizontalAlignment(JLabel.CENTER);
                                                jp.add(lblexpto1,gbc);

                                                gbc.gridx = 2;
                                                gbc.gridy = gridy;
                                                expto.setText("c:/Hasil Hitung Getaran.txt");
                                                gbc.gridwidth=2;
                                                jp.add(expto,gbc);

                                                gbc.gridx = 4;
                                                gbc.gridy = gridy;
                                                btnCalculate.setPreferredSize(new Dimension(50, 23));
                                                jp.add(btnCalculate, gbc);


                                                gbc.gridx = 2;
                                                gbc.gridy = gridy+1;
                                                sccs.setHorizontalAlignment(JLabel.LEFT);
                                                jp.add(sccs,gbc);

                                                gbc.gridx = 4;
```

```
                        gbc.gridy = gridy+1;
                        btnOpenFile.setVisible(false);
                        jp.add(btnOpenFile,gbc);

                        gbc.gridx = 4;
                        gbc.gridy = gridy+2;
                        btnShowAnimasi.setVisible(true);
                        jp.add(btnShowAnimasi,gbc);


                        btnOpenFile.addActionListener(new ActionListener() {

                                @Override
                                public void actionPerformed(ActionEvent e) {
                                        // TODO Auto-generated method stub
                                        try {
                                                Runtime.getRuntime().exec("notepad
"+expto.getText());
                                        } catch (IOException e1) {
                                                // TODO Auto-generated catch block
                                                e1.printStackTrace();
                                        }
                                }
                        });

                        btnShowAnimasi.addActionListener(new ActionListener() {

                                @Override
                                public void actionPerformed(ActionEvent e) {
                                        // TODO Auto-generated method stub
                                        try {
                                                showGrapLongitudinal();
                                        } catch (InterruptedException e1) {
                                                // TODO Auto-generated catch block
                                                e1.printStackTrace();
                                        }
                                }
                        });


                }
        });

        btnActTorsional.addActionListener(new ActionListener() {

                @Override
                public void actionPerformed(ActionEvent e) {
                        //showGrapTorsional();

                        final Gen gen = new Gen();
                        int ordof = Integer.valueOf(txtOrdoTorsional.getText());
```

```
int gridy=10;
for (int i = 1; i <= ordof+3; i++) {

		//k

		gbc.gridx = 0;
		gbc.gridy = gridy;
		JLabel lblkn = new JLabel("k"+i);
		lblkn.setHorizontalAlignment(JLabel.RIGHT);
		jp.add(lblkn,gbc);

		gbc.gridx = 1;
		gbc.gridy = gridy;
		JLabel lblkn1 = new JLabel(":");
		lblkn1.setHorizontalAlignment(JLabel.CENTER);
		jp.add(lblkn1,gbc);

		gbc.gridx = 2;
		gbc.gridy = gridy;
		final JTextField txtk = new JTextField(3);
		txtk.setEditable(false);
		txtk.setBackground(getBackground().WHITE);
		txtk.setHorizontalAlignment(JLabel.RIGHT);
		txtvalk.add(txtk);
		jp.add(txtk,gbc);

		//j

		gbc.gridx = 3;
		gbc.gridy = gridy;
		JLabel lblmn = new JLabel("j"+i);
		lblmn.setHorizontalAlignment(JLabel.RIGHT);
		jp.add(lblmn,gbc);

		gbc.gridx = 4;
		gbc.gridy = gridy;
		JLabel lblkn2 = new JLabel(":");
		lblkn2.setHorizontalAlignment(JLabel.CENTER);
		jp.add(lblkn2,gbc);

		gbc.gridx = 5;
		gbc.gridy = gridy;
		final JTextField txtm = new JTextField(6);
		txtm.setEditable(false);
		txtm.setBackground(getBackground().WHITE);
		txtm.setHorizontalAlignment(JLabel.RIGHT);
		txtvalm.add(txtm);
		jp.add(txtm,gbc);

		if(i==1){
```

```
                txtk.setText(txt_K1_Result.getText());
                txtm.setText(txt_J1_Result.getText());
            }else if(i==2){
                txtk.setText(txt_K2_Result.getText());
                txtm.setText(txt_J2_Result.getText());
            }else if(i==3){
                txtk.setText(txt_K3_Result.getText());
                txtm.setText(txt_J3_Result.getText());
            }else if(i==4){
                txtk.setText(txt_K4_Result.getText());
                txtm.setText(txt_J4_Result.getText());
            }else{
                txtk.setText(txt_K4_Result.getText());
                txtm.setText(txt_J4_Result.getText());
            }


            jp.revalidate();
            validate();
            gridy=gridy+2;
    }

    gbc.gridx = 0;
    gbc.gridy = gridy;
    JLabel lblexpto = new JLabel("Export to");
    lblexpto.setHorizontalAlignment(JLabel.RIGHT);
    jp.add(lblexpto,gbc);

    gbc.gridx = 1;
    gbc.gridy = gridy;
    JLabel lblexpto1 = new JLabel(":");
    lblexpto1.setHorizontalAlignment(JLabel.CENTER);
    jp.add(lblexpto1,gbc);

    gbc.gridx = 2;
    gbc.gridy = gridy;
    expto.setText("c:/Hasil Hitung Getaran.txt");
    gbc.gridwidth=2;
    jp.add(expto,gbc);

    gbc.gridx = 4;
    gbc.gridy = gridy;
    btnCalculate.setPreferredSize(new Dimension(50, 23));
    jp.add(btnCalculate, gbc);


    gbc.gridx = 2;
    gbc.gridy = gridy+1;
    sccs.setHorizontalAlignment(JLabel.LEFT);
    jp.add(sccs,gbc);
```

```
                                gbc.gridx = 4;
                                gbc.gridy = gridy+1;
                                btnOpenFile.setVisible(false);
                                jp.add(btnOpenFile,gbc);

                                gbc.gridx = 4;
                                gbc.gridy = gridy+2;
                                btnShowAnimasi.setVisible(false);
                                jp.add(btnShowAnimasi,gbc);

                                btnOpenFile.addActionListener(new ActionListener() {

                                        @Override
                                        public void actionPerformed(ActionEvent e) {
                                                // TODO Auto-generated method stub
                                                try {
                                                        Runtime.getRuntime().exec("notepad
"+expto.getText());
                                                } catch (IOException e1) {
                                                        // TODO Auto-generated catch block
                                                        e1.printStackTrace();
                                                }
                                        }
                                });

                                btnShowAnimasi.addActionListener(new ActionListener() {

                                        @Override
                                        public void actionPerformed(ActionEvent e) {
                                                // TODO Auto-generated method stub
                                                try {
                                                        showGrapTorsional();
                                                } catch (InterruptedException e1) {
                                                        // TODO Auto-generated catch block
                                                        e1.printStackTrace();
                                                }
                                        }
                                });


                        }
                });

                btnCalculate.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent evt) {
                                int i=0;
                                int ordo;
                                Object cmboitem = jComboBox1.getSelectedItem();
                                if(cmboitem.equals("Longitudinal")){
                                        ordo =
Integer.valueOf(txtOrdoLongitudinal.getText().toString());
```

```
                                        }else{
                                                ordo =
Integer.valueOf(txtOrdoTorsional.getText().toString())+3;
                                        }


                                        double k[] = new double[ordo];
                                        double m[] = new double[ordo];
                                         for(JTextField field: txtvalk){
                                                double f = Double.valueOf(field.getText().toString());
                                         k[i] = f;
                                         i++;
                                        }
                                         int r=0;
                                         for(JTextField field: txtvalm){
                                                double f = Double.valueOf(field.getText().toString());
                                         m[r] = f;
                                         r++;
                                        }

                                         try {
                                                calculateMatrix(k, m, ordo);
                                        } catch (Exception e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        }
                                }
                        });

                        jpContent.add(jp);
                        return  jpContent;
        }

        public void showHideElementLongitudinal(boolean sh){
                txtOrdoLongitudinal.setVisible(sh);
                lbl_ordo_T.setVisible(sh);
                sp_T.setVisible(sh);
                btnActLongitudinal.setVisible(sh);
        }



        public void showHideElementTorsional(boolean sh){

                /*lbl_ordo_J.setVisible(sh);
                spt_J.setVisible(sh);
                txtOrdoTorsional.setVisible(sh);
                btnActTorsional.setVisible(sh);
*/
                lbl_J1.setVisible(sh);
                lbl_J2.setVisible(sh);
```

```
lbl_J3.setVisible(sh);
lbl_J4.setVisible(sh);

lbl_K1.setVisible(sh);
lbl_K2.setVisible(sh);
lbl_K3.setVisible(sh);
lbl_K4.setVisible(sh);
lbl_K1.setVisible(sh);
lbl_K2.setVisible(sh);
lbl_K3.setVisible(sh);
lbl_K4.setVisible(sh);

lbl_J1_R.setVisible(sh);
lbl_J1_m.setVisible(sh);
lbl_J1_P.setVisible(sh);
lbl_J1_r.setVisible(sh);
lbl_J1_L.setVisible(sh);
lbl_J1_Result.setVisible(sh);


//label
lbl_K1_r.setVisible(sh);
lbl_K1_elts.setVisible(sh);
lbl_K1_E.setVisible(sh);
lbl_K1_L.setVisible(sh);
lbl_K1_O.setVisible(sh);
lbl_K1_Result.setVisible(sh);


//label
lbl_J3_P.setVisible(sh);
lbl_J3_r.setVisible(sh);
lbl_J3_L.setVisible(sh);
lbl_J3_Result.setVisible(sh);


//label
lbl_K3_r.setVisible(sh);
lbl_K3_elts.setVisible(sh);
lbl_K3_E.setVisible(sh);
lbl_K3_L.setVisible(sh);
lbl_K3_O.setVisible(sh);
lbl_K3_Result.setVisible(sh);

txt_J1_R.setVisible(sh);
txt_J1_m.setVisible(sh);
txt_J1_P.setVisible(sh);
txt_J1_r.setVisible(sh);
txt_J1_L.setVisible(sh);
txt_J1_Result.setVisible(sh);
```

```
//text
txt_K1_r.setVisible(sh);
txt_K1_elts.setVisible(sh);
txt_K1_E.setVisible(sh);
txt_K1_L.setVisible(sh);
txt_K1_O.setVisible(sh);
txt_K1_Result.setVisible(sh);

//text
txt_J3_P.setVisible(sh);
txt_J3_r.setVisible(sh);
txt_J3_L.setVisible(sh);
txt_J3_Result.setVisible(sh);


//text
txt_K3_r.setVisible(sh);
txt_K3_elts.setVisible(sh);
txt_K3_E.setVisible(sh);
txt_K3_L.setVisible(sh);
txt_K3_O.setVisible(sh);
txt_K3_Result.setVisible(sh);


lbl_J1_Inf_R.setVisible(sh);
lbl_J1_Inf_m.setVisible(sh);
lbl_J1_Inf_P.setVisible(sh);
lbl_J1_Inf_r.setVisible(sh);
lbl_J1_Inf_L.setVisible(sh);
lbl_J1_Inf_Result.setVisible(sh);


//label  info
lbl_K1_inf_r.setVisible(sh);
lbl_K1_inf_elts.setVisible(sh);
lbl_K1_inf_E.setVisible(sh);
lbl_K1_inf_L.setVisible(sh);
lbl_K1_inf_O.setVisible(sh);
lbl_K1_inf_Result.setVisible(sh);



//label  info
lbl_J3_Inf_P.setVisible(sh);
lbl_J3_Inf_r.setVisible(sh);
lbl_J3_Inf_L.setVisible(sh);
lbl_J3_Inf_Result.setVisible(sh);

//label  info
lbl_K3_inf_r.setVisible(sh);
```

```
lbl_K3_inf_elts.setVisible(sh);
lbl_K3_inf_E.setVisible(sh);
lbl_K3_inf_L.setVisible(sh);
lbl_K3_inf_O.setVisible(sh);
lbl_K3_inf_Result.setVisible(sh);

lbl_J2_P.setVisible(sh);
lbl_J2_r.setVisible(sh);
lbl_J2_L.setVisible(sh);
lbl_J2_Result.setVisible(sh);

//label
lbl_K2_r.setVisible(sh);
lbl_K2_elts.setVisible(sh);
lbl_K2_E.setVisible(sh);
lbl_K2_L.setVisible(sh);
lbl_K2_O.setVisible(sh);
lbl_K2_Result.setVisible(sh);

//label
lbl_J4_P.setVisible(sh);
lbl_J4_r.setVisible(sh);
lbl_J4_L.setVisible(sh);
lbl_J4_m.setVisible(sh);
lbl_J4_rp.setVisible(sh);
lbl_J4_Result.setVisible(sh);

//label
lbl_K4_r.setVisible(sh);
lbl_K4_elts.setVisible(sh);
lbl_K4_E.setVisible(sh);
lbl_K4_L.setVisible(sh);
lbl_K4_O.setVisible(sh);
lbl_K4_Result.setVisible(sh);

txt_J2_P.setVisible(sh);
txt_J2_r.setVisible(sh);
txt_J2_L.setVisible(sh);
txt_J2_Result.setVisible(sh);

//text
txt_K2_r.setVisible(sh);
txt_K2_elts.setVisible(sh);
txt_K2_E.setVisible(sh);
txt_K2_L.setVisible(sh);
txt_K2_O.setVisible(sh);
txt_K2_Result.setVisible(sh);

//text
txt_J4_P.setVisible(sh);
txt_J4_r.setVisible(sh);
```

```
        txt_J4_L.setVisible(sh);
        txt_J4_m.setVisible(sh);
        txt_J4_rp.setVisible(sh);
        txt_J4_Result.setVisible(sh);

        //text
        txt_K4_r.setVisible(sh);
        txt_K4_elts.setVisible(sh);
        txt_K4_E.setVisible(sh);
        txt_K4_L.setVisible(sh);
        txt_K4_O.setVisible(sh);
        txt_K4_Result.setVisible(sh);

        lbl_J2_Inf_P.setVisible(sh);
        lbl_J2_Inf_r.setVisible(sh);
        lbl_J2_Inf_L.setVisible(sh);
        lbl_J2_Inf_Result.setVisible(sh);

        //label  info
        lbl_K2_inf_r.setVisible(sh);
        lbl_K2_inf_elts.setVisible(sh);
        lbl_K2_inf_E.setVisible(sh);
        lbl_K2_inf_L.setVisible(sh);
        lbl_K2_inf_O.setVisible(sh);
        lbl_K2_inf_Result.setVisible(sh);

        //label  info
        lbl_J4_Inf_P.setVisible(sh);
        lbl_J4_Inf_r.setVisible(sh);
        lbl_J4_Inf_L.setVisible(sh);
        lbl_J4_Inf_m.setVisible(sh);
        lbl_J4_Inf_rp.setVisible(sh);
        lbl_J4_Inf_Result.setVisible(sh);

        //label  info
        lbl_K4_inf_r.setVisible(sh);
        lbl_K4_inf_elts.setVisible(sh);
        lbl_K4_inf_E.setVisible(sh);
        lbl_K4_inf_L.setVisible(sh);
        lbl_K4_inf_O.setVisible(sh);
        lbl_K4_inf_Result.setVisible(sh);

        lbl_ordo_J.setVisible(sh);
        spt_J.setVisible(sh);
        txtOrdoTorsional.setVisible(sh);
        btnActTorsional.setVisible(sh);

}


public void calculateMatrix(double k[], double m[],int ordo) throws Exception{
```

```
		Gen gen = new Gen();

		double matrixK[][] = gen.createMatrixK(k, ordo);
		double matrixM[][] = gen.createMatrixM(m, ordo);
		double matrixMSqrt[][] = gen.createMatrixMSqrt(m, ordo);
		double matrixMInverse[][] = gen.Inverse(matrixMSqrt);
		double matrixMinvXK[][]= gen.MultiplyMatrix(matrixMInverse, matrixK);
		double matrixD[][] = gen.MultiplyMatrix(matrixMinvXK, matrixMInverse);


		File file = new File(expto.getText());
		StringBuilder sb = new StringBuilder();

		sb.append(System.getProperty("line.separator"));
		sb.append(gen.showMatixToText(matrixK, "==========Matrix K========\n"));
		sb.append(gen.showMatixToText(matrixM, "==========Matrix
"+notasi+"========\n"));
		sb.append(gen.showMatixToText(matrixMSqrt,"==========Matrix
"+notasi+"^1/2========\n"));
		sb.append(gen.showMatixToText(matrixMInverse,"==========Matrix "+notasi+"^(-
1/2)========\n"));
		sb.append(gen.showMatixToText(matrixD,"==========Matrix D========\n"));


		long itr = 1;
		int itrPerSheet = 1;
		long itrSheet = 1;
		for (long i = 1; i <= iteration; i++) {
			if(DEBUG){

				sb.append(System.getProperty("line.separator"));
				sb.append("################################## SHEET
"+itrSheet+"   Iteration "+itrPerSheet +" ############################");

	sb.append(System.getProperty("line.separator"));sb.append(System.getProperty("line.separator"))
;

				MatrixMdl matrixMdl = new MatrixMdl();

				if(matrixPD!=null){
					matrixD = D;
				}
				if(itrPerSheet==1){

	sb.append(gen.showMatixToText(matrixD,"==========Matrix D========\n"));

					matrixMdl = gen.pickedNumber(matrixD, itrPerSheet);

					matrixMdl.setCos(gen.cos(matrixMdl.getPickCosVal(),
matrixMdl.getPickSinVal()));
```

```
                                        matrixMdl.setSin(gen.sin(matrixMdl.getPickCosVal(),
matrixMdl.getPickSinVal()));

                                        double[][] matrixP = gen.createMatrixP(matrixMdl,ordo);


        sb.append(gen.showMatixToText(matrixP,"==============MATRIX
P"+itr+"==============="));

                                        matrixPD =gen.MultiplyMatrix(matrixP, matrixD);


        sb.append(gen.showMatixToText(matrixPD,"==============MATRIX P"+itr+"P"+(itr-
1)+"D=============="));

                                        matrixPT = gen.createMatrixPTranspose(matrixP);


        sb.append(gen.showMatixToText(matrixPT,"==============MATRIX
P"+itr+"'================="));

                                }else{

        sb.append(gen.showMatixToText(matrixPD,"=============Matrix PD==========="));

                                        matrixMdl = gen.pickedNumber(matrixPD, itrPerSheet);
                                        matrixMdl.setCos(gen.cos(matrixMdl.getPickCosVal(),
matrixMdl.getPickSinVal()));
                                        matrixMdl.setSin(gen.sin(matrixMdl.getPickCosVal(),
matrixMdl.getPickSinVal()));

                                        double[][] matrixP = gen.createMatrixP(matrixMdl,ordo);


        sb.append(gen.showMatixToText(matrixP,"==============MATRIX
P"+itr+"==============="));

                                        matrixPD =gen.MultiplyMatrix(matrixP, matrixPD);


        sb.append(gen.showMatixToText(matrixPD,"==============MATRIX P"+itr+"P"+(itr-
1)+"D=============="));

                                        matrixPT = gen.createMatrixPTranspose(matrixP);


        sb.append(gen.showMatixToText(matrixPT,"==============MATRIX
P"+itr+"'================="));
                                }

                                if(itrPerSheet==1){
```

```
                matrixPTN = matrixPT;
            }else{
                matrixPTN =gen.MultiplyMatrix(matrixPTN, matrixPT);
            }


            if(itrPerSheet==ordo-1){
                itrSheet++;
                itrPerSheet=1;

                R = matrixPD;
                Q = matrixPTN;
                D =gen.MultiplyMatrix(R, Q);


        sb.append(gen.showMatixToText(R,"====================Matrix
R================="));


        sb.append(gen.showMatixToText(Q,"====================Matrix Q=P1'PN'
================="));


        sb.append(gen.showMatixToText(D,"====================Matrix
D================"));

                if(Qawal==null){
                    Qawal = matrixPTN;

        sb.append(gen.showMatixToText(Qawal,"====================Matrix Q
Awal================="));
                }else{
                    Qakhir = gen.MultiplyMatrix(Qawal, Q);

        sb.append(gen.showMatixToText(Qawal,"====================Matrix Q
Awal================="));

        sb.append(gen.showMatixToText(Qakhir,"====================Matrix Q
Akhir================="));

                    Qawal = Qakhir;
                }

                boolean validFinal = gen.checkThreeDiagonal(D, ordo);
                if(validFinal){
                    sb.append(gen.ShowfinalQ(R, Qakhir, ordo,
matrixMInverse,"====================Final W================"));
                    break;
                }else{
                    i--;
                }
```

```
                                        ik=1;
                                }else{
                                        itrPerSheet++;
                                        i--;
                                }
                        }

                }

                try (PrintWriter fop = new PrintWriter(file)) {

                        // if file doesn't exists, then create it
                        if (!file.exists()) {
                                file.createNewFile();
                        }

                        fop.write(sb.toString());

                        fop.flush();
                        fop.close();

                        //System.out.println("Done");
                        sccs.setText("Calculate Succesfully..");
                        btnOpenFile.setVisible(true);
                        btnShowAnimasi.setVisible(true);

                } catch (IOException e) {
                        e.printStackTrace();
                        final JPanel panel = new JPanel();
                   JOptionPane.showMessageDialog(panel, e.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);

                }

        }


        public void showGrapTorsional() throws InterruptedException{
                JFrame frameTorsional = new JFrame("Animasi Torsional");
                int ordoVal = Integer.valueOf(txtOrdoTorsional.getText());
                frameTorsional.getContentPane().add(new AnimasiTorsional(ordoVal+3));
                frameTorsional.pack();
                frameTorsional.setVisible(true);
                frameTorsional.addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {
                                System.exit(0);
                        }
                });
```

```
	}

	public void showGrapLongitudinal() throws InterruptedException{
		JFrame frameLongitudinal = new JFrame("Animasi Torsional");

		double xarr[][]={{2, 0.5, 0.75,0.95},{-1, 0.5, 0.75,0.95}};
		frameLongitudinal.getContentPane().add(new AnimasiLongitudinal(xarr));
		frameLongitudinal.pack();
		frameLongitudinal.setVisible(true);
		frameLongitudinal.addWindowListener(new WindowAdapter() {
			public void windowClosing(WindowEvent e) {
				System.exit(0);
			}
		});

	}


	public static void main(String args[]) throws Exception {

		frame.setTitle("Menghitung Getaran");

		RunMatrix runMatrix = new RunMatrix();
		Component content = runMatrix.createCompunent();

		frame.getContentPane().add(content, BorderLayout.LINE_START);

		frame.setSize(1380,600);
		frame.setVisible(true);
		frame.setLocationRelativeTo(null);

		frame.addWindowListener(new WindowAdapter() {
			public void windowClosing(WindowEvent e) {
				System.exit(0);
			}
		});


	}

}
```

Purnomo Adhi Wicaksono, itulah nama lengkap penulis yang telah berhasil menyelesaikan tugas akhir ini yang berjudul “Alat Bantu Edukasi (Digital) Untuk Mempermudah Pemahaman Arti Fisik Frekuensi Natural Dan Mode Shape Getaran Longitudinal Dan Torsional Sistem Propulsi Kapal”. Penulis dilahirkan di Surabaya pada tanggal 17 januari 1995 silam. Penulis merupakan anak pertama dari tiga bersaudara dalam keluarga. Penulis menempuh pendidikan formal sejak tingkat dasar pada TK Salman Al-Farisy Cibinong 1998-2000. Lalu melanjutkan studi ke SDN Ciriung 06 Cibinong 2000-2006. Lalu melanjutkan ke jenjang SMP Puspanegara 2006-2009. Kemudian melanjutkan pada SMAN 3 Kota Bogor 2009-2011. Setelah dinyatakan lulus dalam mengenyam pendidikan SMA, penulis melanjutkan studi di Jurusan Teknik Perkapalan FTK ITS pada tahun 2011 melalui jalus SNMPTN Undangan.

Di Jurusan Teknik Perkapalan penulis mengambil bidang studi Rekayasa Perkapalan – Konstruksi Kapal. Selama masa studi di ITS, selain kegiatan perkuliahan penulis juga pernah menjadi Kepala Divisi Departemen Pendidikan dan Keprofesian HIMATEKPAL (Himpunan Teknik Perkapalan 2012-2014). Penulis juga pernah menjuarai Olimpiade Matematika saat SD (Peringkat I se-Gugus V, Peringkat I se-Kecamatan Cibinong, Peringkat II se-Kabupaten Bogor) dan SMP (Peringkat II Olimpiade Matematika Primagama).

Email : purnomoadhiw@gmail.com
purnomoaw2504@gmail.com