TUGAS AKHIR
NA 1701

PROGRAM FAIRING B-SPLINE PERENCANAAN BADAN KAPAL

Sonda Jeri Silalahi
Nrp. 4192.100.013

JURUSAN TEKNIK PERKAPALAN
FAKULTAS TEKNOLOGI KELAUTAN
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
S U R A B A Y A
1 9 9 9

JURUSAN TEKNIK PERKAPALAN

FAKULTAS TEKNOLOGI KELAUTAN ITS

SURAT KEPUTUSAN TUGAS AKHIR (NA 1701)

No. : 32 /PT12.FTK2/M/1998

Nama Mahasiswa	:	Sonda Jeri Silalahi
Nomor Pokok	:	4192100013
Tanggal diberikan tugas	:	16 . Maret . 1998
Tanggal selesai tugas	:	16 . Juli . 1998
Dosen Pembimbing	:	1. Ir. Petrus Eko Pamunggal, Ph.D
		2. Firmanto Hadi, ST

Uraian / judul tugas akhir yang diberikan :

#PROGRAM FAIRING B- SPLINE PERENCANAAN BADAN KAPAL#

sOn

Surabaya, 16 Maret 1998
Jurusan Teknik Perkapalan FTK-ITS
K e t u a,

Ir. Koestowo Sastro Wiyono
NIP. 130 687 430.

Tembusan :
1. Yth. Dekan FTK-ITS.
2. Yth. Dosen Pembimbing.
3. Arsip.

LEMBAR PENGESAHAN

Surabaya, 16 Februari 1999

Mengetahui dan Menyetujui

Dosen Pembimbing I

16/2-99.

Ir. P. Eko Panaunggal, PhD
NIP. 1303 286 963

Dosen Pembimbing II

16/02/99

Firmanto Hadi, S.T
NIP. 132 133 974

DEPARTEMEN PENDIDIKAN DAN KEBUDAYAAN
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI KELAUTAN
JURUSAN TEKNIK PERKAPALAN

LEMBAR PENGESAHAN

(Telah direvisi sesuai proses verbal Tugas Akhir)

Judul : Program Fairing B-Spline Perencanaan Badan Kapal

Penulis : Sonda Jeri Silalahi

NRP : 4192100013

Dosen Pembimbing : 1. Ir. P. Eko Panunggal, PhD

2. Firmanto Hadi, ST

Surabaya, 03 Maret 1999

Mengetahui dan Menyetujui

Dosen Pembimbing

Ir. P. Eko Panaunggal, PhD
NIP. 132 133 974

ABSTRAK

Pemodelan badan kapal menggunakan persamaan permukaan tiga dimensi merupakan suatu teknik pemodelan yang diharapkan memberikan hasil yang lebih baik daripada pemodelan menggunakan kurva-kurva dua dimensi. Dengan persamaan tiga dimensi analisa terhadap karakteristik permukaan relatif lebih mudah dilakukan.

Tugas Akhir ini memberikan suatu teknik untuk mendefinisikan dan melakukan fairing terhadap persamaan permukaan badan kapal yang selanjutnya dapat digunakan untuk pemodelan badan kapal untuk dianalisa lebih lanjut. Pendefinisian dan fairing tersebut diwujudkan dalam bentuk program interaktif untuk digunakan pada komputer jenis PC sehingga pemanfaatannya dapat lebih luas.

Hasil dari program yang disusun ini adalah berupa data persamaan permukaan. Dari permukaan tersebut dapat dihasilkan visualisasi tiga dimensi dari badan kapal berupa bentuk kurva-kurva potongan badan kapal yaitu body plan, water line dan buttock line dari berbagai potongan yang dikehendaki.

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat TYME yang telah melimpahkan rahmat-Nya. Semata-mata atas kehendak dan kuasa-Nya lah Tugas Akhir ini dapat penulis selesaikan dengan lancar.

Tugas Akhir dengan judul "PROGRAM FAIRING B-SPLINE PERENCANAAN BADAN KAPAL" ini disusun sebagai salah satu syarat untuk menyelesaikan studi di Fakultas Teknologi Kelautan Jurusan Teknik Perkapalan Institut Teknologi Sepuluh Nopember Surabaya, guna melengkapi prasyarat kesarjanaan.

Selanjutnya penulis menyampaikan rasa terima kasih yang teramat dalam kepada :

- Bapak, Ibu, Abang dan Adikku yang telah memberikan dukungan moral maupun material sehingga penulis bisa menyelesaikan studi dengan lancar.
- Bapak P. Eko Panunggal, PhD , selaku Dosen Pembimbing I Tugas Akhir.
- Bapak Firmanto Hadi, S.T , selaku Dosen Pembimbing II Tugas Akhir.
- Bapak Ir. Sjarief Widjaja, PhD, selaku Dosen Wali.
- Bapak Ir. Koestowo Sastro Wiyono, selaku Ketua Jurusan Teknik Perkapalan, FTK, ITS.
- Seluruh Staff Pengajar dan Katyawan FTK, khususnya Jurusan Teknik Perkapalan.
- Seluruh warga Wisper I/99.
- Rekan-rekan P-32, khususnya rekan Gung Wira, Parlindungan Manik, Ahmad Nurdin dan Anton Kristanto beserta keluarga.

- Semua pihak yang tidak bisa penulis sebutkan satu persatu yang telah membantu penulis.

Akhirnya penulis hanya bisa berharap semoga tulisan ini bermanfaat bagi pembaca. Dan dengan kerendahan hati penulis menerima segala saran dan kritik untuk perbaikan tulisan ini.

Surabaya, Febuari 1999

Penulis

DAFTAR ISI

DAFTAR GAMBAR

persamaan permukaan ini semua titik pada permukaan akan tercakup. David.F.Rogers dan J.Alan Adams(1) menyatakan bahwa dengan model matematis tiga dimensi memungkinkan analisa yang relatif lebih mudah terhadap karakteristik permukaan , misalnya volume, luas permukaan dan momen inersia.

Pemodelan matematis tiga dimensi yang telah dilakukan oleh Baihaqqi(2) untuk input data yang bagus permukaan yang dihasilkan relatif terlihat bagus(mulus) tetapi untuk data yang kurang baik pada permukaan yang dihasilkan akan terlihat bagian-bagian yang tidak mulus.

1.2 PERMASALAHAN

Permasalahan yang dikemukakan dalam Tugas Akhir ini adalah bagaimana menghasilkan persamaan permukaan tiga dimensi dari data setenagah lebar kapal dan bagaimana menghilangkan bagian yang tidak mulus pada permukaan yang dihasilkan persamaan tersebut.

1.3 BATASAN MASALAH

Batasan-batasan yang diambil antara lain :

- Metode representasi yang dipakai adalah B-Sline, tanpa membandingkannya dengan yang lain.
- Pembentukan permukaan hanya dilakukan dari data-data setengah lebar kapal yang diberikan, tanpa mengikutkan parameter-parameter bentuk.

- Proses fairing dilakukan tanpa suatu konstrain tertentu seperti displasemen, koefisien blok atau karakteristik kapal lainnya.

1.4 TUJUAN DAN MANFAAT

Tugas Akhir ini memberikan suatu teknik untuk mendefinisikan persamaan permukaan badan kapal yang disertai dengan proses fairing sehingga dihasilkan suatu permukaan yang lebih mulus yang selanjutnya dapat digunakan untuk pemodelan badan kapal dan analisa lebih lanjut. Pendefinisian dan fairing tersebut diwujudkan dalam bentuk program interaktif untuk digunakan pada komputer jenis PC sehingga pemanfaatannya dapat lebih luas.

Hasil dari program yang disusun ini adalah berupa data persamaan permukaan. Dari persamaan permukaan tersebut dapat dihasilkan visualisasi grafik tiga dimensi dari badan kapal dalam bentuk kurva potongan-potongan badan kapal yaitu body plan, water line dan buttock line dari berbagai potongan yang dikehendaki.

Dengan telah terdefinisikannya persamaan permukaan badan kapal dan telah dilakukan proses fairing terhadap persamaan tersebut maka model matematis permukaan tiga dimensi telah diperoleh. Dari model matematis ini berbagai analisa lanjut dapat dilakukan, seperti perhitungan hidrostatik, bonjean, peluncuran dan analisa lain yang memerlukan data bentuk badan kapal dengan hasil yang relatif lebih akurat.

BAB II
REPRESENTASI KURVA DAN PERMUKAAN DENGAN METODE B-SPLINE

2.1 PENDAHULUAN

Suatu proses perancangan yang dibantu dengan penggunaan komputer akan memberikan penyelesaian yang lebih cepat, akurat dan fleksibilitas yang tinggi.

Untuk dapat melakukan perancangan dengan komputer maka suatu objek design perlu dimodelkan. Salah satu pemodelan yang umum digunakan dalam perancangan dengan komputer adalah pemodelan secara matematis. Dengan pemakaian model matematis akan diperoleh keuntungan-keuntungan, antara lain :

- Tingkat presisi yang tinggi, sesuai dengan kemampuan komputer yang menanganinya.
- Koordinat setiap titik pada objek, baik berupa kurva atau permukaan dapat diperoleh secara akurat, sehingga berbagai macam analisa terhadap objek dapat dilakukan dengan hasil yang akurat.

Metode-metode yang umum digunakan dalam pemodelan kurva dan permukaan adalah Cubic Spline untuk kurva, Bezier Spline dan B-Spline untuk kurva dan permukaan.

Sesuai dengan judul dan batasan masalah maka dalam tulisan ini hanya dibahas mengenai metode B-Spline untuk mempresentasikan kurva dan permukaan secara matematis.

2.2. REPRESENTASI KURVA DENGAN B-SPLINE

B-Spline, kependekan umum dari Basis Spline, pertama kali diperkenalkan oleh Schoenberg. B-Spline merupakan suatu fungsi matematis dengan sifat-sifat khusus yang jika dibandingkan dengan polinomial atau fungsi spline lainnya, fungsi ini memiliki beberapa kelebihan yang menguntungkan dalam membentuk suatu kurva.

Horst Nowacky (3) mengemukakan bahwa B-Spline memenuhi syarat dari karakteristik suatu fungsi yang dikehendaki untuk digunakan dalam membentuk suatu kurva. Karakteristik yang dikehendaki itu antara lain :

1. Memungkinkan pendefinisian kurva tanpa memperhatikan orientasi kurva tersebut dalam ruang. Dengan memutar Object maupun sumbu koordinat yang dipakai akan dihasilkan bentuk kurva atau permukaan yang sama.
2. Adanya hubungan yang pasti antara variabel bebas dan variabel tak bebas.
3. Perhitungan dari masing-masing komponen koordinat dapat dilakukan terpisah.

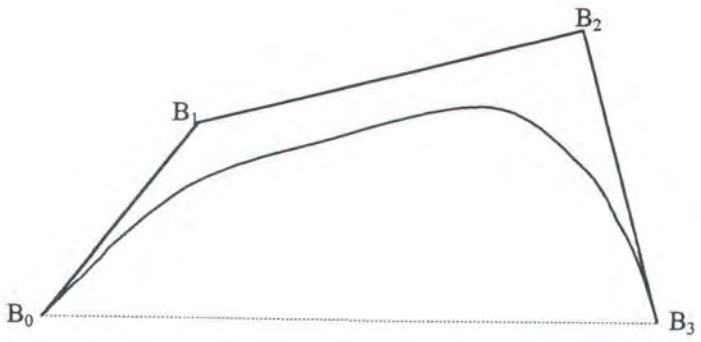
Ketiga syarat ini dapat dipenuhi oleh suatu persamaan parametris, dan B-Spline juga merupakan persamaan parametris, syarat selanjutnya :

4. Memungkinkan adanya diskontinuitas pada turunan orde tertentu pada suatu kurva.
5. Memungkinkan kontrol lokal terhadap bagian tertentu dari kurva, dan tidak mengakibatkan perubahan pada bagian yang lain.

6. Mampu menampung sejumlah batasan geometris.
7. Mudah digunakan tanpa fungsi matematis yang rumit yang membutuhkan dasar matematis yang kuat.
8. Perhitungan dapat dilakukan secara tepat.

2.2.1 TEORI DASAR

Penjelasan Teori Dasar berikut ini diambil dari David F. Rogers dan J.Alan Adams (1). Kurva B-Spline dibentuk dari suatu poligon dengan kordinat-koordinatnya telah ditentukan terlebih dahulu. Kurva yang terbentuk bukan kurva yang tepat melalui titik-titik poligon, tetapi yang terbentuk adalah kurva yang lebih "smooth" terhadap poligonnya. Istilah 'smooth' yang dipakai disini hanya mengacu pada pengertian kualitatif, yaitu suatu kurva yang 'enak dipandang' dan tidak tampak patah, tanpa mengacu pada suatu definisi matematis tertentu.



Gbr 2.1 Kurva B-Spline dan Poligon pembentuknya

Kurva B-Spline dibentuk dari persamaan :

$$p(t) = \sum_{i=1}^{n+1} B_i \, N_{i,k}(t) \; ; \; t_{min} \le t \le t_{max} \, , \, 2 \le k \le n+1 \qquad \text{pers(2-1)}$$

Dimana B_i adalah koordinat dari n+1 poligon dan $N_{i,k}$ merupakan fungsi basis B-Spline. k adalah order kuva B-Spline, yang berarti bahwa kurva B-Spline diwakili oleh kurva polinomial derajat k-1.Perhitungan fungsi basis B-Spline dilakukan berdasarkan rumus rekursi Cox-deBoor :

$$N_{i,1}(t) = \begin{cases} = 0 \text{ jika } x_i \le t < x_{i+1} \\ = 1 \text{ untuk hal lainnya} \end{cases} \qquad \text{pers(2-2a)}$$

dan

$$N_{i,k}(t) = \frac{(t - x_i)N_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} \qquad \text{pers(2-2b)}$$

Nilai x_i adalah elemen dari knot vektor yang harus memenuhi syarat $x_i \le x_{i+1}$. Harga parameter t berawal dari t_{min} sampai t_{max} sepanjang kurva. Dalam perhitungan fungsi basis ini diterapkan perjanjian 0/0 = 0.

Knot Vektor mengindikasikan range dari parameter t, yaitu suatu poligon mulai berpengaruh saat $t = x_i$ dan berakhir pengaruhnya pada $t = x_{i+1}$. Penentuan nilai knot vektor dapat dilakukan dengan tiga metode yaitu Uniform B-Spline, Open Uniform B-Spline dan Non Uniform B-Spline.

Masing-masing jarak knot vektor untuk uniform knot vektor adalah sama. Sebagai contoh adalah

[0 1 2 3 4]

Uniform knot vektor umumnya berawal dari 0 dan bertambah 1 sampai harga maksimum dan dapat juga dinormalkan antara 0 dan 1 dengan interval bilangan desimal yang sama, contoh [0 0.25 0.5 0.75 1].

Knot vektor Open Uniform mempunyai penggandaan nilai elemen knot sebanyak order k pada bagian ujung. Bagian dalam knot vektor mempunyai selisih yang sama. Beberapa contoh knot vektor Open Uniform :

k = 2 [0 0 1 2 3 4 4]

k = 3 [0 0 0 1 2 3 3 3]

k = 4 [0 0 0 0 1 2 2 2 2]

sehingga secara umum knot vektor Open Uniform dinyatakan dengan :

$x_i = 0$ untuk $1 \le i \le k$ pers(2-3a)

$x_i = i-k$ untuk $k+1 \le i \le n+1$ pers(2-3b)

$x_i = n-k+2$ untuk $n+2 \le i \le n+k+1$ pers(2-3c)

Knot vektor Non Uniform pada bagian ujungnya sama dengan knot vektor Open Uniform sedangkan bagian tengah knot vektor Non Uniform dibuat proporsional dengan jarak antar poligon sehingga knot vektor Non Uniform dapat diperoleh melalui :

$x_i = 0$ untuk $1 \le i \le k$ pers(2-4a)

$$x_{i+k} = \left(\frac{\left(\frac{i}{n-k+2} \right) c_{i+1} + \sum_{j=1}^{i} c_j}{\sum_{g=1}^{n} c_g} \right) (n-k+2) \quad \text{untuk } k+1 \le i \le n+1 \qquad \text{pers(2-4b)}$$

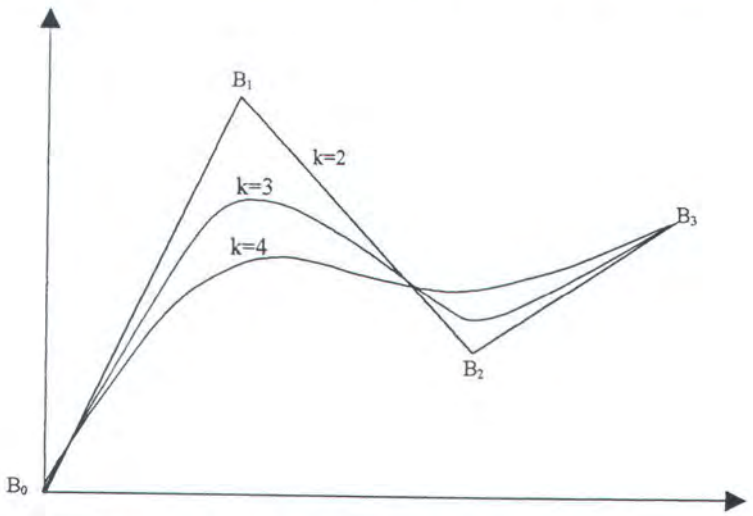
$x_i = n-k+2$ untuk $n+2 \le i \le n+k+1$ pers(2-4c)

Fleksibilitas kurva B-Spline dapat dihasilkan dengan beberapa cara :

- Merubah type knot vektor : Uniform, Open Uniform atau Non Uniform.

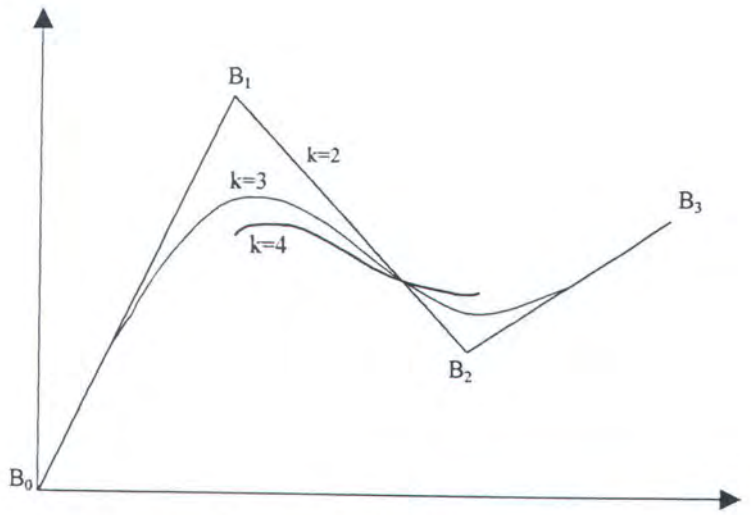
- Merubah order k fungsi basis B-Spline
- Merubah jumlah dan posisi poligon

Gambar 2.2 menunjukkan tiga kurva B-Spline dengan knot vektor open uniform dengan order yang berbeda. Kurva B-Spline order k=4 merupakan kurva polinomial derajat 3. Kurva B-Spline order k=3 dibentuk dari dua kurva parabola (kurva polinomial derajat 2). Sedangkan kurva B-Spline order k=2 merupakan segmen kurva linier yang berimpit dengan poligonnya.



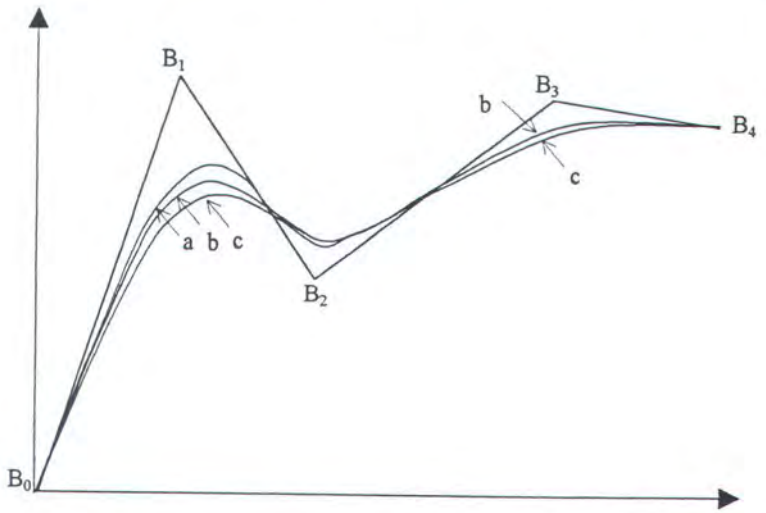
Gbr 2.2 Pengaruh variasi order k pada kurva B-Spline dengan knot vektor Open Uniform

Ujung-ujung kurva B-Spline dengan knot vektor uniform tidak berimpit dengan ujung-ujung poligonnya. Gambar 2.3 memperlihatkan tiga kurva B-Spline dengan knot vektor uniform dengan order yang berbeda. Untuk order k=2 kurva berimpit dengan poligonnya. Tetapi untuk k>2 ujung awal dan akhir kurva tidak berimpit dengan ujung-ujung poligon. Untuk order k=3 kurva berawal dari pertengahan poligon B_0 dan B_1, dan berakhir pada pertengahan poligon B_2 dan B_3. Untuk order k=4 kurva yang dihasilkan lebih pendek lagi.



Gbr 2.3 Pengaruh variasi order k pada kurva B-Spline dengan Uniform knot vektor

Kurva B-Spline Non Uniform dihasilkan dengan memperhitungkan jarak antar poligon. Untuk jarak poligon yang sama satu dengan yang lainnya maka



Gbr 2.4 Perbandingan Kurva B-Spline Open Uniform dan Non Uniform. (a). Knot vektor Uniform. (b). Non Uniform knot vektor . (c) Non Uniform knot vektor dengan vertex ganda pada B_3

kuva yang dihasilkan akan sama dengan kurva dengan knot vektor open uniform. Gambar 2.4 memperlihatkan perbandingan kurva B-Spline open uniform dan non uniform.

Derivatif dari kurva B-Spline pada sembarang titik pada kurva diperoeh dengan melakukan differensial terhadap persamaan B-Spline :

$$P(t) = \sum_{i=1}^{n+1} B_i \, N_{i,k}(t)$$

derivatif pertama :

$$P'(t) = \sum_{i=1}^{n+1} B_i \, N'_{i,k}(t) \qquad \text{pers(2-5)}$$

dan derivatif kedua :

$$P''(t) = \sum_{i=1}^{n+1} B_i \, N''_{i,k}(t) \qquad \text{pers(2-6)}$$

Derivatif fungsi basis B-Spline juga diperoleh dengan differensial terhadap persamaan fungsi basis B-Spline :

$$N_{i,k}(t) = \frac{(t - x_i)N_{i,k-.1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}}$$

Maka derivatif pertama :

$$N'_{i,k}(t) = \frac{(t - x_i)N'_{i,k-.1}(t) + N_{i,k-.1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)N'_{i+1,k-1}(t) + N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} \qquad \text{pers(2-7)}$$

dan derivatif kedua :

$$N'_{i,k}(t) = \frac{(t - x_i)N''_{i,k-.1}(t) + 2N'_{i,k-.1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)N''_{i+1,k-1}(t) + 2N'_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} \qquad \text{pers(2-8)}$$

2.2.2 B-SPLINE CURVE FITTING

Pada bagian sebelumnya telah didiskudikan bahwa kurva B-Spline dihasilkan dari poligon-poligon yang telah ditentukan sebelumnya. Pada bagian ini kurva B-Spline akan dibentuk dari sejumlah titik data.

Jika titik-titik data terletak pada kurva B-Spline, maka pers(2-1) harus dipenuhi :

$$D_1(t_1) = N_{1,k}(t_1)B_1 + N_{2,k}(t_1)B_2 + \ldots + N_{i,k}(t_1)B_i$$

$$D_2(t_2) = N_{1,k}(t_2)B_1 + N_{2,k}(t_2)B_2 + \ldots + N_{i,k}(t_2)B_i$$

$$\vdots$$

$$D_j(t_j) = N_{1,k}(t_j)B_1 + N_{2,k}(t_j)B_2 + \ldots + N_{i,k}(t_j)B_i$$

Bentuk persamaan dapat ditulis dalam bentuk matriks :

$$[D] = [N][B] \qquad \text{pers(2-9)}$$

dimana

$$[D]^T = [\, D_1(t_1) \;\; D_2(t_2) \;\; \ldots D_j(t_j) \,]$$

$$[B]^T = [\, B_1 \;\; B_2 \;\; \ldots \;\; B_i \,]$$

$$[N] = \begin{bmatrix} N_{1,k}(t_1) & N_{2,k}(t_1) & \cdots & N_{i,k}(t_1) \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ N_{1,k}(t_j) & N_{2,k}(t_j) & \cdots & N_{i,k}(t_j) \end{bmatrix}$$

Jika $2 \le k \le n+1 = j$, maka matriks [N] merupakan matriks bujursangkar dan poligon dapat diperoleh dengan inverse matriks, yaitu :

$$[B] = [N]^{-1}\,[D] \qquad \text{pers(2-10)}$$

Jika jumlah poligon yang akan dihasilkan lebih kecil dari jumlah titik data, yaitu $2 \leq k \leq n+1 \leq j$, maka matriks [N] bukan merupakan matriks bujursangkar. Penyelesaiannya dilakukan dengan bantuan Transpose matriks, yaitu :

$$[D] = [N]\,[B]$$

$$[N]^T\,[D] = [N]^T\,[N]\,[B]$$

$$[B] = [\,[N]^T\,[N]\,]^{-1}\,[N]^T\,[D]$$

Harga parameter t_j untuk setiap titik data diukur dari jarak titik data pada kurva B-Spline. Pendekatan untuk harga parameter t adalah menggunakan panjang chord antara titik data. Untuk sejumlah j titik data harga parameter t adalah :

$$t1 = 0 \qquad \text{pers(2-11a)}$$

$$\frac{t_l}{t_{\max}} = \frac{\sum_{s=2}^{l} \sum |D_s - D_{s-1}|}{\sum_{s=2}^{j} |D_s - D_{s-1}|} \qquad l \geq 2 \qquad \text{pers(2-11b)}$$

Harga parameter t_{max} biasanya diambil sebesar harga maksimum dari knot vektor.

2.3 REPRESENTASI PERMUKAAN DENGAN B-SPLINE

Representasi permukaan dengan metode B-Spline pada dasarnya sama dengan representasi kurva dengan metode B-Spline. Perbedaannya adalah jika pada representasi kurva hanya terdapat satu parameter t , maka untuk representasi permukaan terdapat dua parameter u dan w. Perhitungan knot vektor dan fungsi basis dilakukan dengan cara yang sama.

Permukaan B-Spline dinyatakan dengan persamaan :

$$Q(u,w)=\sum_{i=1}^{n+1}\sum_{j=1}^{m+1}B_{i,j}\,N_{i,k}(u)M_{j,l}(w) \qquad \text{pers(2-12)}$$

Dimana Bi,j adalah poligon dan $N_{i,k}(u)$ dan $M_{j,l}(u)$ adalah fungsi basis B-Spline pada arah biparametrik u dan w. Fungsi basis B-Spline dinyatakan dengan persamaan :

$$N_{i,1}(u)=\begin{cases}=0 \text{ jika } x_i \le u < x_{i+1}\\ =1 \text{ untuk hal lainnya}\end{cases} \qquad \text{pers(2-13a)}$$

$$N_{i,k}(u)=\frac{(u-x_i)N_{i,k-1}(u)}{x_{i+k-1}-x_i}+\frac{(u_{i+k}-t)N_{i+1,k-1}(u)}{x_{i+k}-x_{i+1}} \qquad \text{pers(2-13b)}$$

dan

$$M_{j,1}(w)=\begin{cases}=0 \text{ jika } y_i \le w < y_{i+1}\\ =1 \text{ untuk hal lainnya}\end{cases} \qquad \text{pers(2-14a)}$$

$$M_{j,1}(w)=\frac{(w-y_j)M_{j,l-1}(w)}{y_{j+l-1}-y_j}+\frac{(y_{j+1}-w)M_{j+1,l-1}(w)}{y_{j+1}-y_{j+1}} \qquad \text{pers(2-14b)}$$

Dimana xi dan yj adalah elemen knot vektor.

Derivatif parametrik dari permukaan B-Spline diperoleh dengan diferensial terhadap persamaan permukaan B-Spline :

$$Q_u(u,w)=\frac{\partial Q}{\partial u}=\sum_{i=1}^{n+1}\sum_{j=1}^{m+1}B_{i,j}\,N'_{i,k}(u)M_{j,l}(w) \qquad \text{pers(2-15a)}$$

$$Q_w(u,w)=\frac{\partial Q}{\partial w}=\sum_{i=1}^{n+1}\sum_{j=1}^{m+1}B_{i,j}\,N_{i,k}(u)M'_{j,l}(w) \qquad \text{pers(2-15b)}$$

$$Q_{uw}(u,w)=\frac{\partial^2 Q}{\partial u\partial w}=\sum_{i=1}^{n+1}\sum_{j=1}^{m+1}B_{i,j}\,N'_{i,k}(u)M'_{j,l}(w) \qquad \text{pers(2-15c)}$$

$$Q_{uu}(u,w) = \frac{\partial^2 Q}{\partial u^2} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} B_{i,j}\, N''_{i,k}(u) M_{j,l}(w) \qquad \text{pers(2-15d)}$$

$$Q_{ww}(u,w) = \frac{\partial^2 Q}{\partial w^2} \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} B_{i,j}\, N_{i,k}(u) M''_{j,l}(w) \qquad \text{pers(2-15e)}$$

2.3.1 B-SPLINE SURFACE FITTING

Pada bagian sebelumnya telah didiskudikan bahwa permukaan B-Spline dihasilkan dari poligon-poligon yang telah ditentukan sebelumnya. Pada bagian ini permukaan B-Spline akan dibentuk dari sejumlah titik data.

Jika titik-titik data terletak pada permukaan B-Spline, maka pers() harus dipenuhi :

$$D_{1,1}(u_1, w_1) = N_{1,k}(u_1)\,[M_{1,l}(w_1)\,B_{1,1} + M_{2,l}(w_1)B_{1,2} + \ldots + M_{m+1,l}(w_1)B_{1,m+1}]$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$+ N_{n+1,k}(u_1)\,[M_{1,l}(w_1)B_{n+1,1} + M_{2,l}(w_1)B_{n+1,2} + \ldots + M_{m+1,l}(w_1)B_{n+1,m+1}]$$

Bentuk persamaan di atas dapat ditulis dalam bentuk matriks :

$$[D] = [C]\,[B] \qquad \text{pers(2-16)}$$

dimana $C_{i,j} = N_{i,k}\, M_{j,l}$.

Jika [C] merupakan matriks bujursangkar maka poligon dapat diperoleh dengan inverse matriks, yaitu :

$$[B] = [C]^{-1}\,[D] \qquad \text{pers(2-17)}$$

Jika [C] bukan merupakan matriks bujursangkar maka poligon dapat diperoleh dengan menggunakan transpose matriks :

$$[D] = [C]\,[B]$$

$$[C]^T\,[D] = [C]^T\,[C]\,[B]$$

$$[B] = [\,[C]^T\,[C]\,]^{-1}\,[C]^T\,[D]$$

Harga parameter u dan w untuk setiap titik data diukur dari jarak titik data pada permukaan B-Spline. Pendekatan untuk harga parameter u dan w adalah menggunakan panjang chord antara titik data. Untuk sejumlah r titik data pada arah parametrik u harga parameter u adalah :

$$u_1 = 0 \qquad \text{pers(2-18a)}$$

$$\frac{u_l}{u_{\max}} = \frac{\sum_{g=2}^{l}\sum\left|D_{g,s} - D_{g-1,s}\right|}{\sum_{g=2}^{r}\left|D_{g,s} - D_{g-1,s}\right|} \qquad \text{pers(2-18b)}$$

dan untuk sejumlah r titik data pada arah parametrik w harga parameter w adalah:

$$w_1 = 0 \qquad \text{pers(2-19a)}$$

$$\frac{w_l}{w_{\max}} = \frac{\sum_{g=2}^{l}\sum\left|D_{r,g} - D_{r,g-1}\right|}{\sum_{g=2}^{s}\left|D_{r,g} - D_{r,g-1}\right|} \qquad \text{pers(2-19b)}$$

Harga parameter u_{max} dan w_{max} adalah harga maksimum dari knot vektor.

2.3.2 GAUSSIAN KURVATUR DAN FAIRING PERMUKAAN

Dalam perencanaan yang dibantu dengan komputer diperlukan suatu teknik untuk memperlihatkan atau memvisualisasikan kemulusan (smoothhnes/fairness) dari sebuah objek yang berbentuk permukaan. Walaupun suatu permukaan telah dihasilkan dengan metode-metode representasi permukaan yang ada, seperti Bezier Spline, B-Spline dan sebagainya, mungkin masih dapat ditemui bagian-bagian yang kurang mulus. Pada saat ini teknik matematis yang

paling baik untuk visualisasi kemulusan permukaan adalah dengan menggunakan kurvatur gaussian.

Kurvatur gaussian dinyatakan dengan rumus :

$$\kappa_g = \frac{AC - B^2}{|Q_u \times Q_w|^4} \qquad \text{pers(2-20)}$$

dimana

$$A = [\, Q_u \times Q_w \,] \cdot Q_{uu}$$

$$B = [\, Q_u \times Q_w \,] \cdot Q_{uw}$$

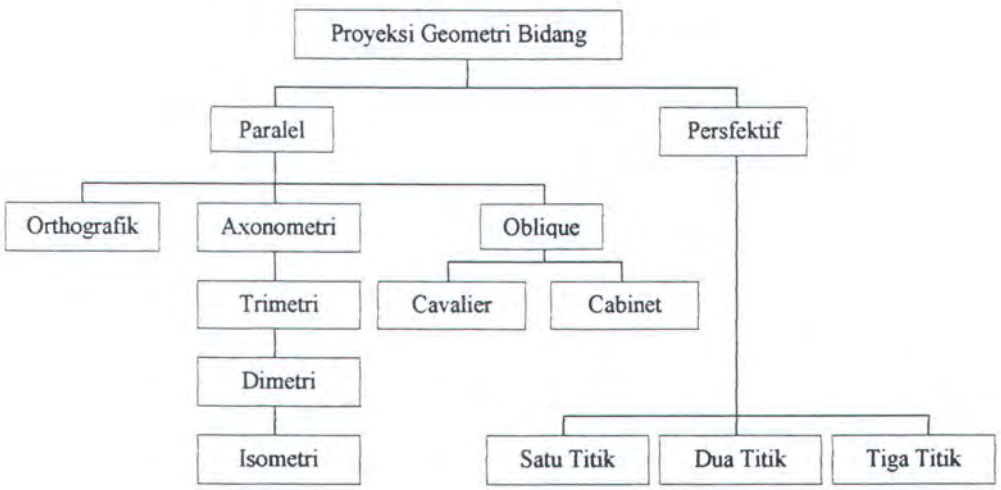
$$C = [\, Q_u \times Q_w \,] \cdot Q_{ww}$$

Kurvatur gaussian pada sebuah titik yang terletak pada sebuah permukaan mengindikasikan kondisi lokal permukaan pada titik tersebut, apakah permukaan tersebut elliptic, hyperbolic atau parabolic. (Kurvatur gaussian positif, negatif atau nol). Jika kurvatur gaussian nol maka permukaan dapat dibentangkan menjadi sebuah bidang datar.

2.4 TEKNIK MENAMPILKAN OBJEK TIGA DIMENSI

Teknik untuk menampilkan objek tiga dimensi ke bidang dua dimensi ada dua jenis yaitu transformasi paralel dan transformasi persfektif. Kedua transformasi ini merupakan transformasi tiga dimensi yaitu mentransformasikan objek tiga dimensi dari suatu ruang tiga dimensi ke ruang tiga dimensi lainnya. Untuk menampilkan hasil transformasi ke bidang dua dimensi diperlukan proyeksi dari ruang tiga dimensi ke bidang dua dimensi. Hasil tampilan yang

diperoleh melalui proyeksi tersebut disebut proyeksi geometri bidang. Gambar 2. Menunjukkan hirarki dari poyeksi geometri bidang.



Gbr 2. Hirarki Proyeksi Geometri Bidang

Proyeksi Persfektif memproyeksikan objek ke bidang gambar dengan garis proyeksi bersumber pada satu titik yang disebut titik pandang. Jadi gambar yang dihasilkan tidak proporsional dengan objek, yaitu bagian objek yang lebih dekat akan terlihat lebih besar dibandingkan dengan bagian objek yang lebih jauh dengan ukuran yang sama.

Proyeksi Orthografik diperoleh dengan memproyeksikan objek ke bidang $x=0$, bidang $y=0$ dan bidang $z=0$. Jadi diperoleh tiga bidang gambar orthogonal saja yaitu pandangan depan atau belakang, pandangan samping dan pandangan atas atau bawah.

Proyeksi Oblique memproyeksikan objek ke bidang gambar dengan garis proyeksi yang tidak tegak lurus dengan bidang gambar, sehingga hanya permukaan yang sejajar dengan bidang proyeksi yang akan ditampilkan sesuai

dengan bentuk dan ukuran aslinya. Kekurangan jenis proyeksi ini adalah pengamat akan mengalami kesulitan untuk menentukan bidang gambar dan sudut proyeksinya untuk mendapatkan gambar yang diinginkan.

Proyeksi Axonometri menghasilkan gambar proyeksi ke bidang gambar dengan memutar objek sebesar θ terhadap sumbu x, kemudian memutar objek sebesar ϕ terhadap sumbu y lalu memproyeksikannya ke bidang z=0. Proyeksi isometri memakai sudut putar θ dan ϕ yang tetap yaitu $\theta=45^0$ dan $\phi=35.26^0$. Dalam proyeksi Dimetri terdapat hubungan antara sudut θ dan ϕ sehingga gambar proyeksi diperoleh dengan memasukkan salah satu sudut putar tersebut. Untuk proyeksi Trimetri sudut putar θ maupun sudut putar ϕ dapat dipilih sesuai dengan keinginan kita.

Dengan uraian singkat mengenai macam-macam proyeksi di atas maka penulis memilih proyeksi Axonometri Trimetri untuk penulisan program dan karenanya hanya proyeksi tersebut yang akan dibahas selanjutnya.

Proyeksi Axonometri Trimetri memanipulasi objek tiga dimensi dengan melakukan rotasi sebesar θ terhadap sumbu x , kemudian rotasi sebesar ϕ terhadap sumbu y dan proyeksi ke bidang z=0.

Matriks rotasi sebesar θ terhadap sumbu x menurut David F.Rogers dan J.Alan Adams (1) :

$$[R_x] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

Matriks rotasi sebesar ϕ terhadap sumbu y:

$$[R_y] = \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix}$$

Matriks proyeksi ke bidang z=0 :

$$[P_z] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

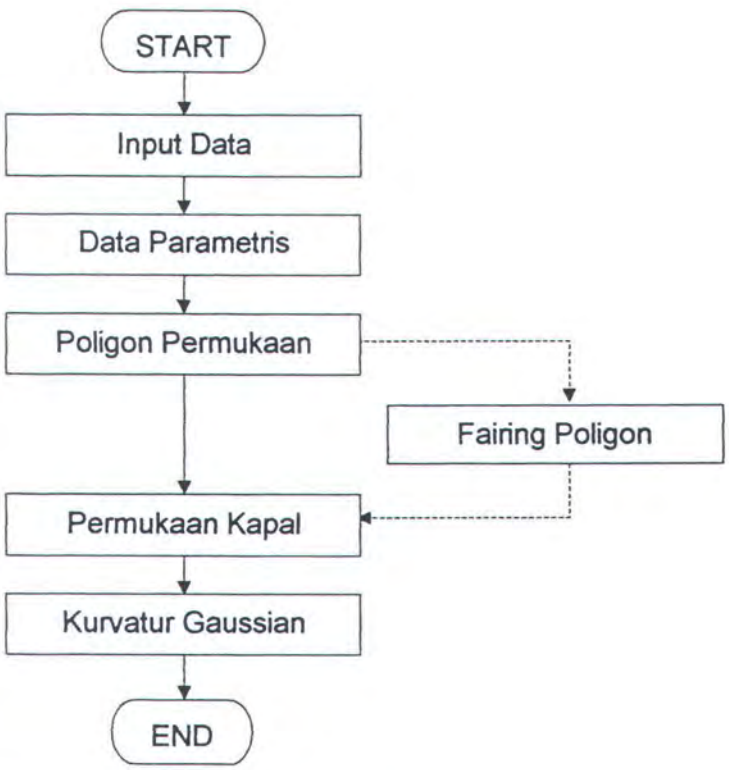
Jika [X] merupakan koordinat objek tiga dimensi maka dengan proyeksi Axonomtri Trimetri akan diperoleh proyeksi ke bidang dua dimensi dengan koordinat [X*] :

$$\{X^*] = [X]\ \{R_x]\ R[_y]\ [P_z]$$

BAB III
PROGRAM FAIRING PERMUKAAN KAPAL

3.1 BLOCK DIAGRAM PROGRAM

Program Fairing Permukaan kapal ini disusun sesuai dengan block diagram pada gbr.3.1. Block Diagram tersebut menggambarkan proses yang akan dikerjakan program secara garis besar. Panah dengan garis putus-putus



Gbr. 3.1 Block Diagram Program

menunjukkan suatu proses yang optional yang berarti proses yang tergantung pada pemakai program apakah proses tersebut dikerjakan atau tidak dikerjakan.

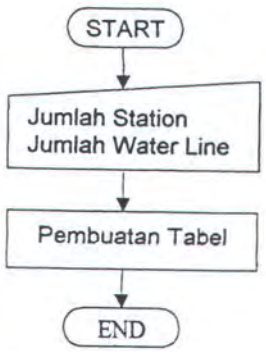
Program akan dimulai dengan penerimaan input data yang akan diolah menjadi data parametris. Data parametris tersebut akan digunakan untuk menghitung poligon permukaan. Data poligon permukaanlah yang akan dipakai untuk memperoleh permukaan kapal, dimana sebelum proses perhitungan permukaan, poligon dapat mengalami proses fairing terlebih dahulu. Perhitungan permukaan akan menghasilkan data dalam bentuk station, garis air dan buttock line. Proses perhitungan permukaan akan disertai dengan proses perhitungan kurvatur gaussian. Kurvatur gaussian hanya memberikan suatu indikasi bentuk permukaan.

Pada kenyataannya semua proses di atas tidak dilakukan sesuai dengan urutan seperti block diagram program. Karena semua data disimpan dalam media penyimpan (disket/hard disk) maka proses yang manapun dapat dilakukan terlebih dahulu. Sebagai contoh adalah dengan file data poligon yang telah tersimpan pemakai program dapat langsung menghitung permukaan ataupun melakukan proses fairing. Jadi Block Diagram di atas bukan menggambarkan urutan proses yang dilakukan program, tetapi menggambarkan proses yang dilakukan program terhadap input data sampai diperoleh data permukaan kapal.

Proses-proses di atas akan dijabarkan lebih lanjut pada sub-bab berikutnya.

3.2 DATA MASUKAN

Pada umumnya sebuah program komputer memerlukan data masukan (input) untuk diolah dan menghasilkan suatu keluaran (output). Program fairing permukaan kapal ini memerlukan data masukan berupa setengah lebar kapal, jarak masing-masing station dari AP, jarak masing-masing water line dari base line dan jarak stem-stern kapal pada masing-masing water line. Pembuatan unit pengolahan data masukan adalah seperti yang ditunjukkan gbr. 3.2.



Gbr. 3.2 Flowchart Pengolahan Data Masukan

Pemasukan data dimulai dengan pengisian jumlah station dan jumlah Water Line, kemudian akan dibuat tabel-tabel sesuai dengan jumlah station dan jumlah Water Line. Tabel yang akan dibuat terdiri dari tiga jenis yaitu tabel setengah lebar, tabel jarak masing-masing station dari AP, tabel jarak water line dari base line dan stem-stern. Tabel-tabel ini perlu ditetapkan jumlah baris, jumlah kolom dan judul masing-masing kolomnya. Tabel-tabel ini akan dibentuk seperti pada gbr.3.3 sampai gbr.3.5.

Jumlah baris tabel setengah lebar adalah sebanyak jumlah station, sedang jumlah kolomnya sebanyak jumlah water line tambah 1 untuk kolom station. Judul masing-masing kolom seperti pada gbr.3.3.

Station	WL 1	WL 2	.....	WL n
1			.....	
2			.....	
:	:	:	.....	:
:	:	:	.....	:
m			.....	

m = jumlah station
n = jumlah water line

Gbr. 3.3 Tabel setengah lebar kapal

Tabel jarak masing-masing station dari AP membutuhkan baris sebanyak jumlah station serta jumlah kolom sebanyak 2 yaitu untuk kolom station dan jarak masing-masing station dari AP seperti terlihat pada gbr 3.4.

Station	Jarak
1	
2	
:	:
:	:
m	

m = jumlah station

Gbr. 3.4 Tabel jarak masing-masing station dari AP

Tabel jarak masing-masing water line dari base line membutuhkan baris sebanyak jumlah water line serta jumlah kolom sebanyak 4 yaitu untuk kolom

water line, jarak masing-masing water line dari base line, jarak stem dari station terakhir dan jarak stern dari AP seperti terlihat pada gbr 3.5.

Water Line	Jarak	stem dari station m	stern dari AP
1			
2			
:	:	:	:
:	:	:	:
n			

n = jumlah water line

Gbr. 3.5 Tabel jarak masing-masing water line dari base line dan stem-stem

Setelah pembuatan tabel selesai maka masing-masing tabel siap menerima masukan. Setelah tabel diisi dengan data maka data siap untuk diolah menjadi data pembentuk permukaan badan kapal.

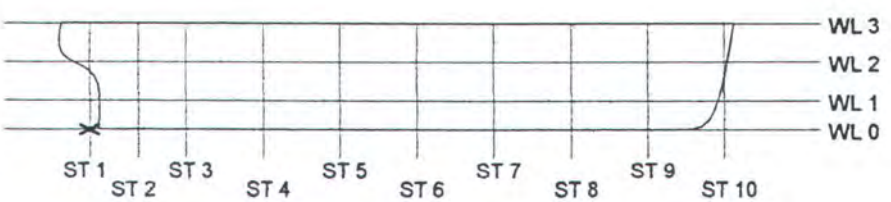
3.3 DATA PEMBENTUK PERMUKAAN KAPAL

Data masukan yang diperoleh sebelumnya belum dapat digunakan dalam perhitungan permukaan B-Spline karena data masukan tidak membagi masing-msing station dan masing-masing water line dalam jumlah bagian yang sama, seperti diilustrasikan dalam gbr. 3.6. Water line 0 hanya memiliki 8 titik data yaitu mulai station 2 sampai station 8, sedang water line 3 memiliki 10 titik data mulai station 1 sampai station 10. Station 1 dan station 10 hanya memiliki dua titik data yaitu water line 2 dan 3, sedang station 2 sampai station 9 memiliki 4 titik data yaitu mulai dari water line 0 sampai water line 4.

Untuk mendapatkan poligon permukaan B-Spline dibutuhkan data yang berbentuk matriks :

$$[D]=\begin{bmatrix} D_{1,1} & D_{1,2} & \cdots & D_{1,n} \\ D_{2,1} & D_{2,2} & \cdots & D_{2,n} \\ \vdots & \vdots & & \vdots \\ D_{m,1} & D_{m,2} & \cdots & D_{m,n} \end{bmatrix}$$

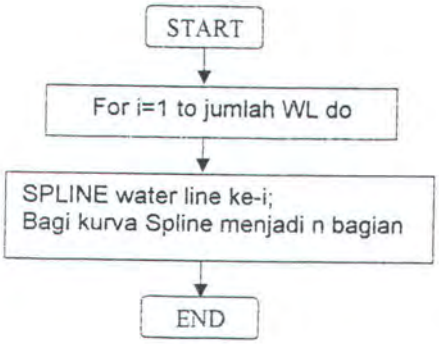
dimana m = jumlah station dan n = jumlah water line. Masing-masing elemen D yaitu Di,j dimana i=1 sampai m dan j=1 sampai n merupakan titik yang terletak pada permukaan. Jika data masukan yang dimasukkan dalam matriks [D] maka D1,1 (ditandai dengan 'x' pada gbr 3.6) bukan merupakan titik yang terletak pada permukaan.



Gbr. 3.6. Pembagian station dan water line kapal

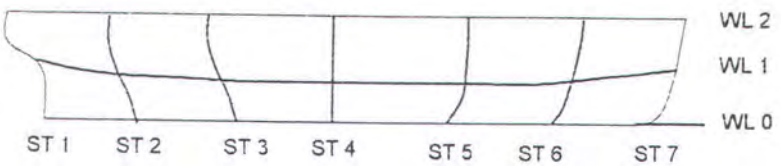
Jadi data masukan harus diubah terlebih dahulu menjadi data pembentuk permukaan dengan cara membagi masing-masing water line dan station sehingga memiliki jumlah bagian yang sama. Cara pembagiannya dilakukan menurut flowchart pada gbr. 3.7. Melalui titik-titik data masing-masing water line dibuat kurva spline dan kurva spline yang diperoleh dibagi menjadi jumlah bagian yang sama. Banyaknya bagian itu disebut jumlah station parametris. Melalui titik-titik data yang baru diperoleh juga dibuat kurva spline dan dibagi menjadi jumlah bagian yang sama yang disebut jumlah water line parametris.

Setelah pembagian kurva spline maka akan diperoleh bentuk station dan water line baru yang tidak merupakan garis lurus lagi seperti terlihat pada gbr.3.8.



Gbr. 3.7 Flowchart perhitungan data pembentuk permukaan

Station yang baru disebut station parametris dan water line yang baru disebut water line parametris.

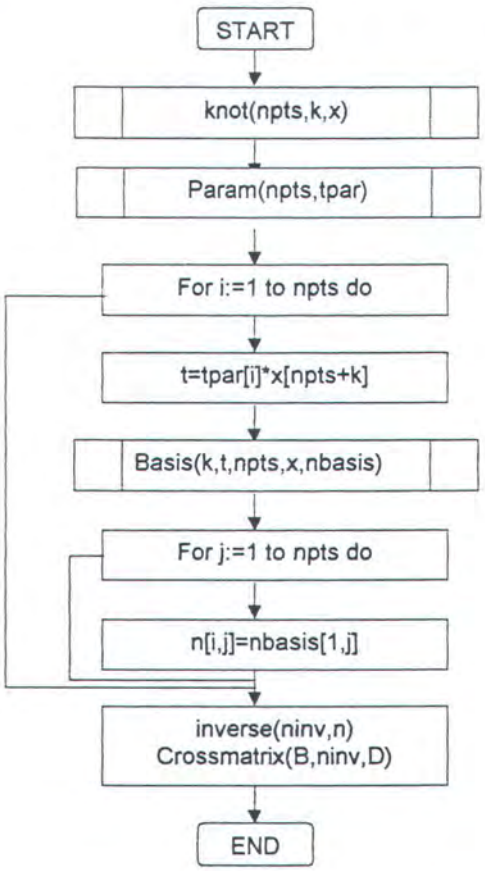


Gbr. 3.8. Kapal dengan station parametris dan water line parametris

Proses spline dan pembagian kurva spline yang terdapat pada flowchart gbr. 3.7 dilaksanakan melalui tiga tahap perhitungan yaitu pertama perhitungan poligon, kedua perhitungan kurva spline dan ketiga perhitungan titik data pada kurva spline yang diinginkan.

Perhitungan poligon ditunjukkan dalam flowchart gbr. 3.9. Perhitungan poligon dimulai dengan menentukan knot vektor dan parameter tpar. Perhitungan knot vektor dan parameter tpar akan dijelaskan pada bagian selanjutnya. Setelah

itu dihitung fungsi basis, dan fungsi basis ini dikumpulkan dalam satu variabel penampung yaitu variabel n. Sesuai dengan rumus perhitungan poligon $[B]=[N]^{-1}[D]$ dimana [B] koordinat poligon, [N] fungsi basis B-Spline dan [D]



Gbr. 3.9. Flowchart perhitungan Poligon

titik data , maka variabel n diinvers dan didapatkan hasil inversnya dalam variabel ninv. Hasil invers tersebut kemudian dikalikan dengan titik data [D] dan didapatkan koordinat poligon dalam variabel B.

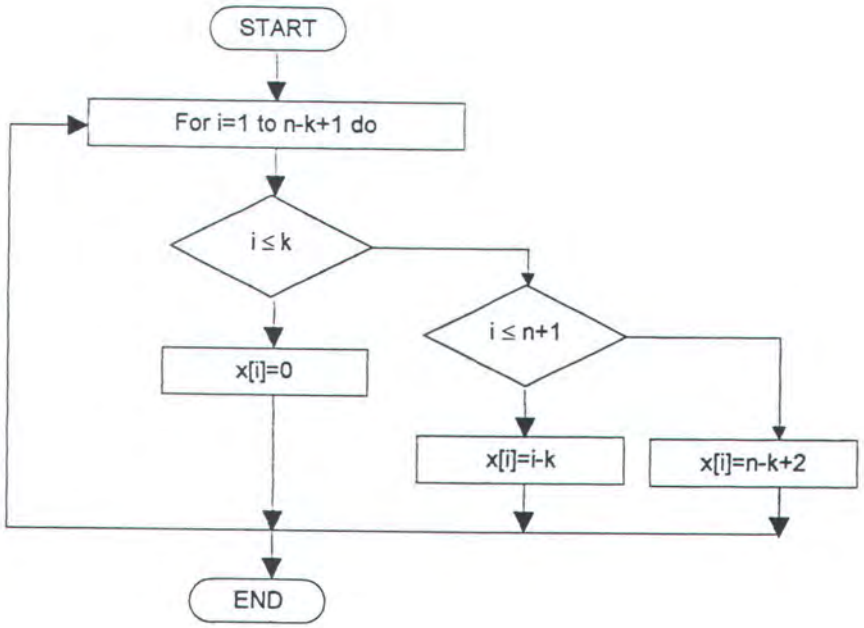
Perhitungan poligon di atas memanggil prosedur knot, param dan basis. Prosedur knot merupakan perhitungan knot vektor dan perhitungannya ditunjukkan pada Flowchart gbr.3.10. Perhitungan knot vektor dilakukan berdasarkan pers(2-3a) sampai pers(2-3c) :

$x_i = 0$ untuk $1 \leq i \leq k$

$x_i = i-k$ untuk $k+1 \leq i \leq n+1$

$x_i = n-k+2$ untuk $n+2 \leq i \leq n+k+1$

Dimana n+1 merupakan jumlah poligon dan k adalah order kurva spline. Langkah pertama perhitungan adalah melakukan loop sebanyak n+k+1 kali, Setiap loop



Gbr. 3.10. Flowchart perhitungan knot vektor

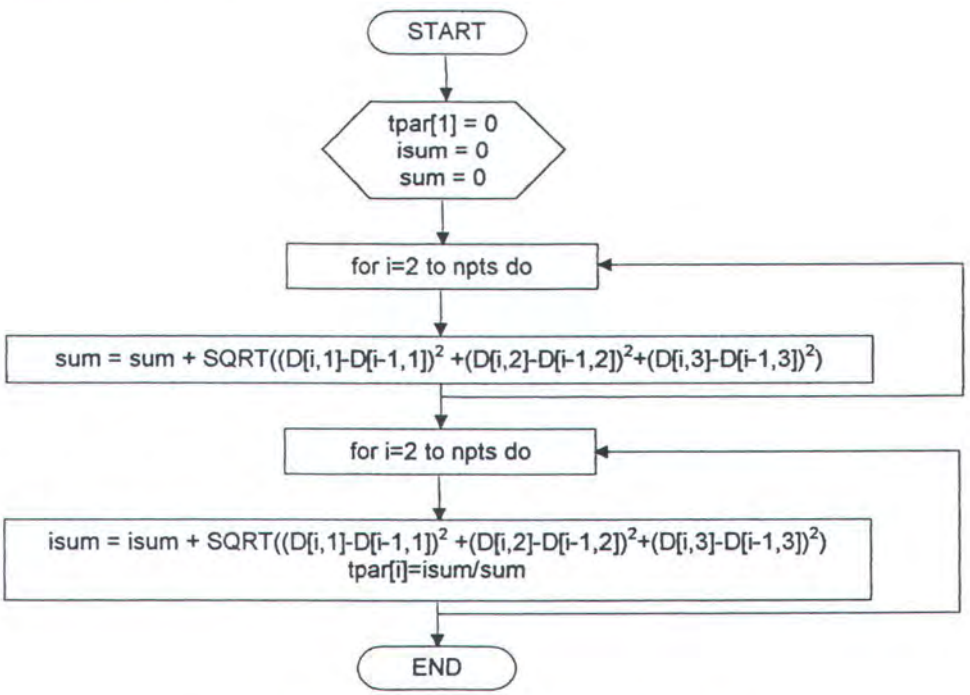
diuji apakah nilai i≤k. Jika i≤k maka x[i]=0 dan jika tidak maka diuji kembali apakah i≤n+1. Jika i≤n+1 maka x[i]=i-k dan jika tidak maka x[i]=n-k+2.

Prosedur param merupakan prosedur untuk menghitung harga parameter tpar berdasarkan pers(2-11a) dan pers (2-11b) :

$$t1 = 0$$

$$\frac{t_l}{t_{\max}} = \frac{\sum_{s=2}^{l} \sum |D_s - D_{s-1}|}{\sum_{s=2}^{j} |D_s - D_{s-1}|} \qquad l \geq 2$$

Flowchartnya dapat dibuat seperti gbr.3.11. Perhitungan diawali dengan pemberian harga tpar[1]=0, isum=0 dan sum=0. Sum merupakan penjumlahan seluruh panjang chord titik data, sedangkan isum merupakan penjumlahan panjang chord sampai titik data ke-l, dimana l merupakan indeks parameter yang sedang dihitung.



Gbr. 3.11. Flowchart perhitungan parameter tpar

Prosedur Basis merupakan prosedur untuk menghitung fungsi basis B-Spline berdasarkan pers(2-2a) dan pers(2-2b) :

$$N_{i,1}(t) = \begin{cases} = 0 \text{ jika } x_i \le t < x_{i+1} \\ = 1 \text{ untuk hal lainnya} \end{cases}$$

dan

$$N_{i,k}(t) = \frac{(t - x_i)N_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}}$$

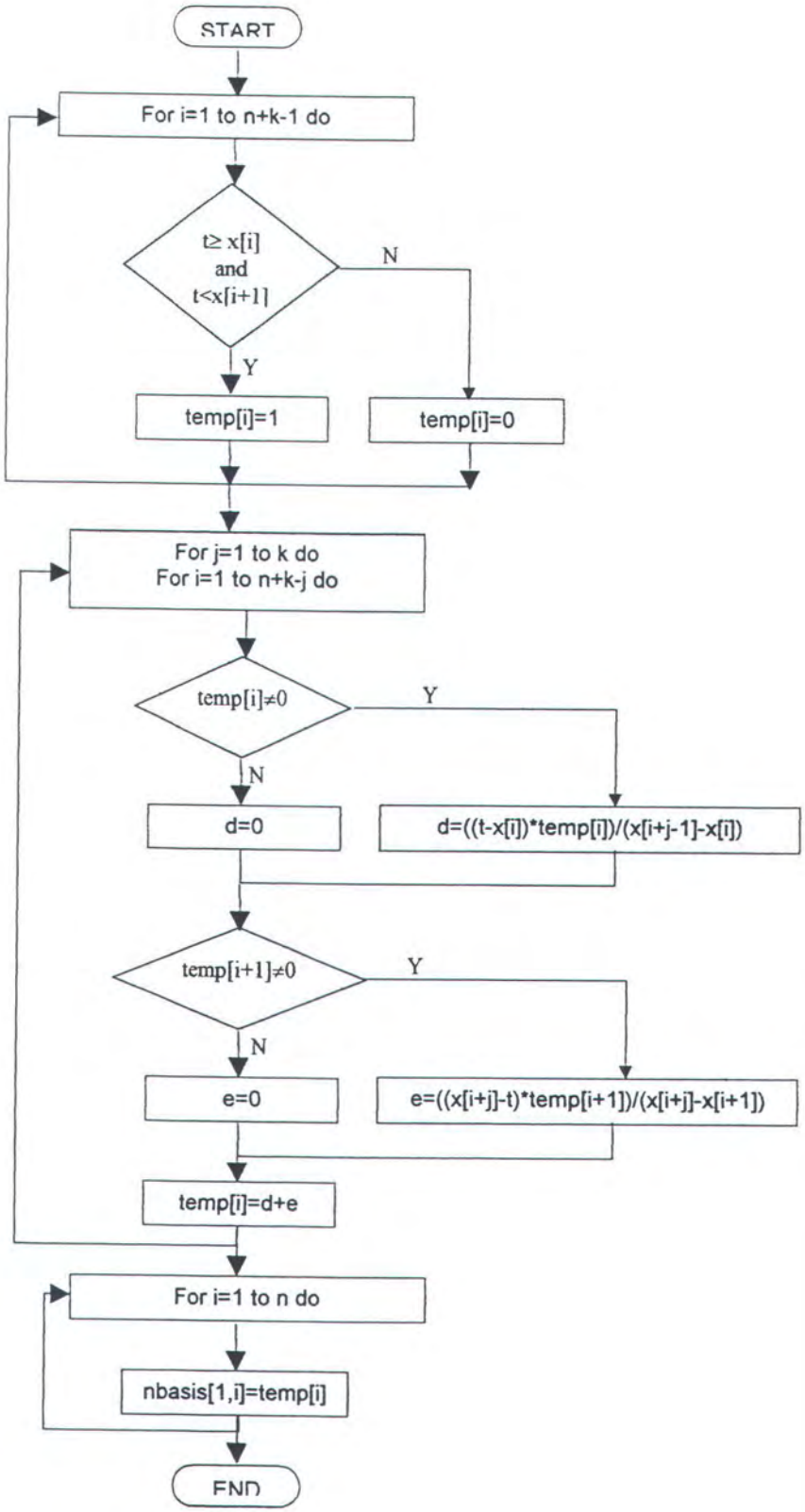
Sehingga flowchartnya dapat dibuat seperti gbr. 3.12.

Sampai sejauh ini telah diperoleh poligon dari titik data. langkah selanjutnya adalah perhitungan kurva spline berdasarkan pers(2-1) :

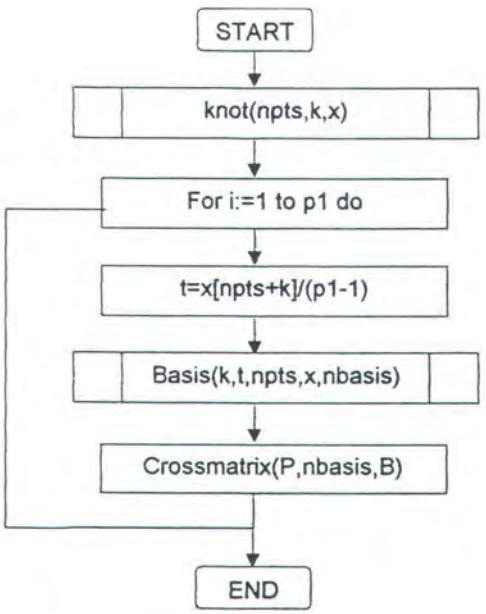
$$p(t) = \sum_{i=1}^{n+1} B_i \, N_{i,k}(t) \; ; \; t_{min} \le t \le t_{max} \, , \, 2 \le k \le n+1$$

Flowchart untuk perhitungan kurva spline adalah seperti gbr.3.13. Setelah knot vektor diperoleh maka panjang maksimum parameter kurva x[npts+k] dibagi (p1-1) dimana p1 merupakan jumlah titik yang akan didapat dari kuva spline. Kemudian dihitung fungsi basis dan titik pada kurva yaitu P diperoleh dengan mengalikan poligon dengan fungsi basis.

Seperti telah dijelaskan sebelumnya setelah dibuat kurva spline melalui water line dan station maka akan didapat titik data baru yaitu P. Titik data P ini akan digunakan untuk menghitung poligon permukaan.



Gbr. 3.12. Flowchart perhitungan fungsi basis B-Spline



Gbr. 3.13. Flowchart perhitungan kurva B-Spline

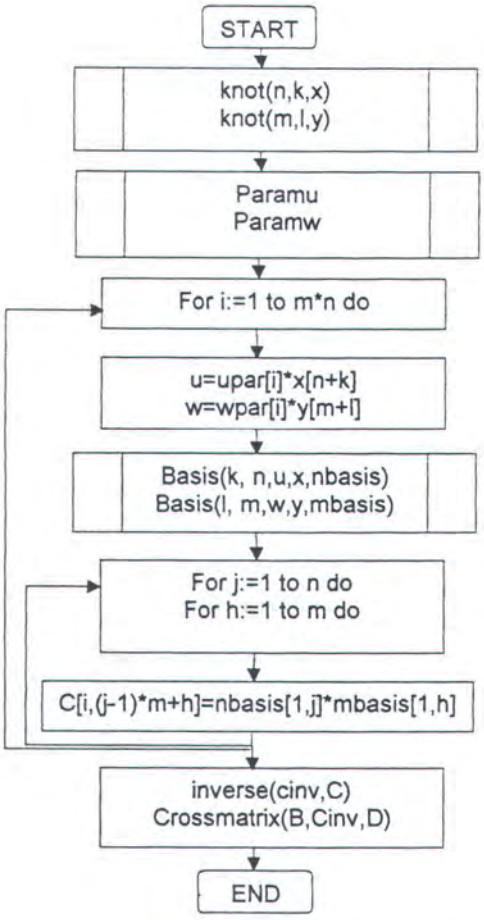
3.4 POLIGON PERMUKAAN

Poligon permukaan dibentuk dari data pembentuk permukaan kapal. Perhitungan poligon permukaan berdasarkan pers(2-17) :

$$[B] = [C]^{-1} [D]$$

dimana [B] merupakan koordinat poligon, [D] data pembentuk permukaan kapal yang diperoleh pada sub-bab 3.2. [C] adalah fungsi basis B-Spline. Flowchart perhitungan poligon dibuat seperti pada gbr. 3.14. m adalah jumlah water line, n jumlah station, k merupakan order permukaan ke arah u dan l order permukaan ke arah w. Perhitungan dimulai dengan penentuan knot vektor x dan y, kemudian dihitung parameter u dan parameter w untuk masing-masing titik data. Perhitungan fungsi basis dengan prosedur Basis menghasilkan fungsi basis Nbasis

untuk arah u dan Mbasis untuk arah W. C merupakan hasil perkalian antara Nbasis dengan Mbasis. Setelah C diinvers, diperoleh Cinv, lalu dikalikan dengan titik data D untuk mendapatkan poligon permukaan B. Proses Paramu dan Paramw adalah sama dengan proses Param pada sub-bab 3.2 dan proses Basis juga sama dengan proses Basis pada sub-bab 3.2.



Gbr. 3.14. Flowchart perhitungan Poligon permukaan kapal

3.5 PERHITUNGAN PERMUKAAN KAPAL

Poligon permukaan yang diperoleh pada sub-bab 3.3 dipergunakan untuk menghitung permukaan kapal. Persamaan yang dipakai adalah pers(2-12) :

3.5.1 Perhitungan Station Kapal

Suatu garis pada permukaan diwakili oleh beberapa titik dimana masing-masing titik yang berurutan dihubungkan dengan garis. Jadi untuk mendapatkan suatu station tertentu dihitung koordinat beberapa titik, dalam program ini 100 titik. Sumbu-sumbu koordinat ditentukan sebagai berikut. sumbu x adalah sumbu ke arah memanjang kapal, sumbu z adalah sumbu tegak kapal dan sumbu y adalah sumbu melintang kapal.

Koordinat ke arah x suatu station tertentu adalah tetap. Koordinat ke arah z didapat dengan membagi tinggi station dibagi dengan jumlah titik yang akan dicari kurang satu. Jadi suatu station tertentu telah diketahui koordinat x dan koordinat z. Jadi perhitungan station tertentu adalah bagaimana menentukan komponen Qy titik Q pada station tersebut dengan komponen Qx dan Qz telah diketahui. Persamaan permukaan B-Spline dapat ditulis :

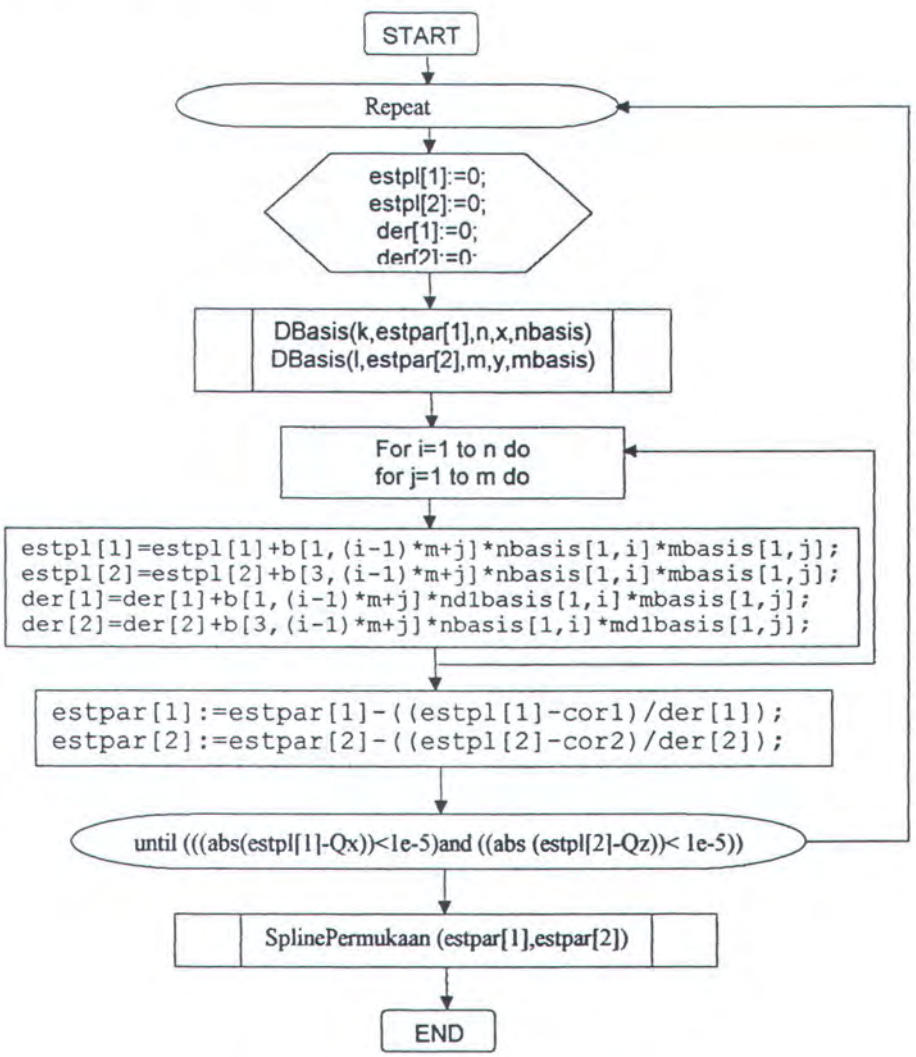
$$Qx = f(u,w)$$

$$Qy = g(u,w)$$

$$Qz = h(u,w)$$

Melalui dua persamaan Qx dan Qz dengan dua komponen Qx dan Qz telah diketahui maka secara matematis dapat dicari u dan w yang selanjutnya dapat dihitung komponen Qy. Tetapi karena fungsi f(u,w) dan h(u,w) merupakan fungsi yang tidak sederhana dalam u dan w maka penentuan u dan w sangat sulit untuk diselesaikan secara analitik. Jalan pemecahan yang diambil adalah dengan cara numerik menggunakan metode Newton Raphson. Flowchart perhitungannya disusun sepeti gbr.3.16.

estpar[1] dan estpar[2] mewakili u dan w yang dicari. estpl[1] dan estpl[2] mewakili komponen x dan komponen z pada permukaan untuk u=estpar[1] dan w=estpar[2].estpar[1] dan estpar[2] dihitung sampai selisih antara Qx dan estpl[1] lebih kecil dari 10^{-5} dan selisih antara Qz dan estpl[2] lebih kecil dari 10^{-5}. Jika estpar[1] dan estpar[2] telah didapat maka komponen Qy dapat dicari dengan proses Spline Permukaan. Proses Spline Permukaan adalah proses yang dijelaskan pada flowchart gbr. 3.15.



Gbr. 3.16. Flowchart perhitungan Station Kapal

3.5.2 Perhitungan Water Line Kapal

Untuk mendapatkan suatu water line tertentu ditempuh cara yang sama dengan mendapatkan station kapal. Perbedaannya adalah jika pada perhitungan station komponen Qx konstan maka pada perhitungan water line komponen Qz konstan. Koordinat ke arah z suatu water line tertentu adalah tetap. Koordinat ke arah x didapat dengan membagi panjang water line dengan jumlah titik yang akan dicari kurang satu. Jadi suatu water line tertentu telah diketahui koordinat x dan koordinat z. Flowchartnya sama dengan perhitungan station.

3.5.3 Perhitungan Buttock Line Kapal

Perhitungan untuk mendapatkan suatu Buttock line tertentu juga sama dengan perhitungan station dan water line kapal. Perbedaannya adalah jika pada perhitungan station dan water line komponen yang diketahui adalah komponen Qx dan Qz maka pada perhitungan buttock line komponen yang diketahui adalah komponen Qy dan Qx.

Koordinat ke arah y suatu buttock line tertentu adalah tetap. Koordinat ke arah x didapat dengan membagi panjang buttock line dengan jumlah titik yang akan dicari kurang satu. Jadi suatu buttock line tertentu telah diketahui koordinat x dan koordinat z. Jadi perhitungan buttock line tertentu adalah bagaimana menentukan komponen Qz titik Q pada buttock line tersebut dengan komponen Qy dan Qx telah diketahui.

Flowchart perhitungan buttock line juga sama dengan flowchart perhitungan station dan water line. Bagian yang berbeda adalah estpl[1] dan

estpl[2] mewakili komponen x dan komponen y pada permukaan untuk u=estpar[1] dan w=estpar[2]. estpar[1] dan estpar[2] dihitung sampai selisih antara Qx dan estpl[1] lebih kecil dari 10^{-5} dan selisih antara Qy dan estpl[2] lebih kecil dari 10^{-5}. Jika estpar[1] dan estpar[2] telah didapat maka komponen Qz dapat dicari dengan proses SplinePermukaan. Proses SplinePermukaan adalah proses yang dijelaskan pada flowchart gbr. 3.15.

3.6 PERHITUNGAN KURVATUR GAUSSIAN

Telah dijelaskan pada bab II bahwa kurvatur Gauss dapat digunakan untuk menggambarkan bentuk lokal permukaan. Pehitugan poligon berdasarkan pada pers(2-20) :

$$\kappa_g = \frac{AC - B^2}{|Q_u \times Q_w|^4}$$

dimana $A = [\, Q_u \times Q_w \,] \cdot Q_{uu}$ $B = [\, Q_u \times Q_w \,] \cdot Q_{uw}$ $C = [\, Q_u \times Q_w \,] \cdot Q_{ww}$

Derivatif Q_u, Q_w, Q_{uu}, Q_{ww} dan Q_{uw} berdasarkan pada pers(2-15a) s/d (2-15e) :

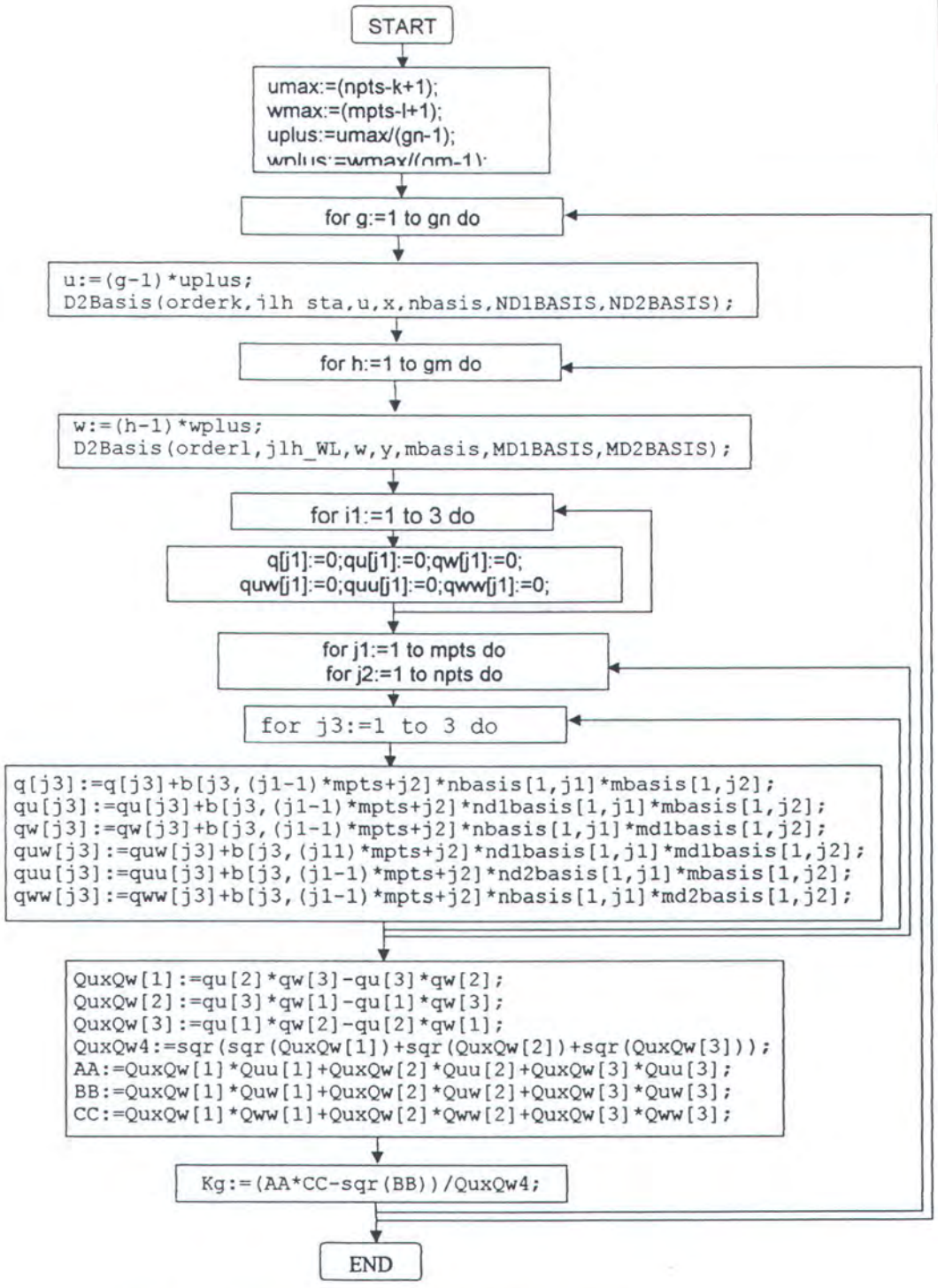
$$Q_u(u,w) = \frac{\partial Q}{\partial u} = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j}\, N'_{i,k}(u) M_{j,l}(w)$$

$$Q_w(u,w) = \frac{\partial Q}{\partial w} = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j}\, N_{i,k}(u) M'_{j,l}(w)$$

$$Q_{uw}(u,w) = \frac{\partial^2 Q}{\partial u \partial w} = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j}\, N'_{i,k}(u) M'_{j,l}(w)$$

$$Q_{uu}(u,w) = \frac{\partial^2 Q}{\partial u^2} = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j}\, N''_{i,k}(u) M_{j,l}(w)$$

$$Q_{ww}(u,w) = \frac{\partial^2 Q}{\partial w^2} = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j}\, N_{i,k}(u) M''_{j,l}(w)$$



Gbr. 3.17. Flowchart perhitungan kurvatur gaussian

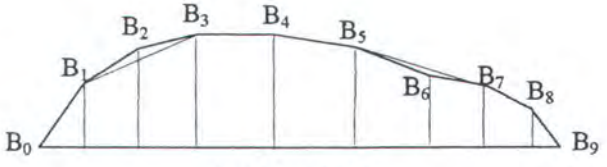
Maka Flowchart untuk menghitung kurvatur gaussian dapat disusun seperti gbr.3.17. Nilai maksimum parameter u adalah npts-k+1 dimana npts jumlah

poligon ke arah u dan nilai maksimum parameter w adalah mpts-l+1 dimana mpts merupakan jumlah poligon ke arah w. k dan l adalah order permukaan B-Spline ke arah u dan w. Kurvatur Gaussian akan dihitung sebanyak gn × gm titik pada permukaan kapal, dimana gn pembagian ke arah u dan gm pembagian ke arah w. Jadi Nilai maksimum parameter u yaitu umax dibagi menjadi gn-1 bagian dan Nilai maksimum parametr w yaitu wmax dibagi menjadi gm-1 bagian.

Untuk mendapatkan kurvatur Gaussian pada tiap-tiap titik yang direncanakan dicari harga derivatif fungsi basis B-Spline Q_u, Q_w, Q_{uu}, Q_{ww} dan Q_{uw} . Kemudian kurvatur gaussian diperoleh melalui pers(2-20).

3.7 FAIRING POLIGON

Sebelum permukaan kapal dicari melalui poligon permukaan, poligon tersebut perlu difairkan. Prosesnya dapat dijelaskan dengan bantuan gbr.3.18. Untuk setiap poligon diberikan suatu tanda '+' , '-' atau '0'. Tanda ini diperoleh



gbr 3.18. Poligon

dengan menarik garis antara 2 poligon yang mengapit poligon yang dicari tandanya. Jika poligon tersebut berada di atas garis tersebut maka tanda untuk poligon itu adalah '+', '-' jika berada di bawah garis tersebut dan '0' jika tepat berada pada garis tersebut. Pada gambar 3.16 poligon B_2 bertanda '+' karena

berada di atas garis yang menghubungkan antara poligon B_1 dan B_3, dan poligon B_6 bertanda '-' karena berada di bawah garis yang menghubungkan poligon B_5 dan B_7.

Jika poligon pada gambar 3.16 disusun tandanya akan didapatkan tanda :

B_0 B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8 B_9

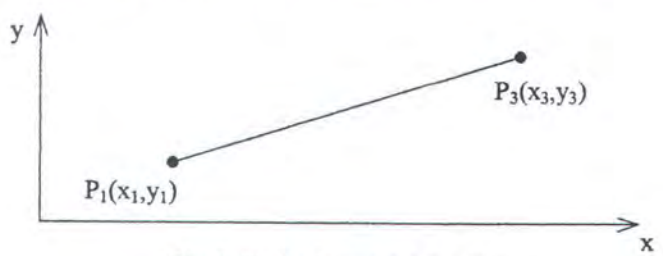
'+' '+' '+' '+' '+' '+' '-' '+' '+' '+'

Jika melalui poligon ini dibuat kurva maka kurva akan berganti kelengkungan pada sekitar poligon B_6 karena hanya B_6 yang mempunyai tanda '-'. Secara matematis tidak pernah ditemukan definisi kurva yang fair. Dalam hal ini penulis mendefinisikan kurva yang fair sebagai kurva yang enak dipandang yaitu kurva yang pergantian kelengkungannya tidak banyak terjadi dan tidak terdapat kelengkungan yang berbeda pada daerah yang sempit seperti poligon B_6 pada gbr.3.18. Jadi fairing poligon dilakukan dengan mengubah poligon-poligon tertentu untuk mendapatkan tanda-tanda yang tidak sering berganti dan tidak ada tanda yang diapit oleh dua tanda yang berbeda dari dua tanda tersebut.

Untuk memfairkan poligon penulis memilih cara manual. Dipilihnya cara manual dan bukan otomatis adalah karena untuk suatu poligon yang berganti tanda berulang kali sangat sulit untuk memperoleh tanda apa yang sebaiknya menggantikan tanda-tanda yang berulang kali berganti tersebut. Sebagai contoh adalah jika terdapat suatu tanda pada poligon '+','-','+','-','+','-','+' maka akan terdapat banyak alternatif untuk merubah poligon untuk mendapatkan tanda yang lebih baik. Alternatif pertama adalah poligon dirubah sehingga semua tanda '+' berubah menjadi tanda '-'. Alternatif kedua poligon dirubah sehingga semua

tanda '-' berubah menjadi tanda '+'. Alternatif ketiga poligon dirubah sehingga beberapa poligon pertama bertanda menjadi '+' dan beberapa poligon terakhir bertanda '-'. Alternatif keempat adalah poligon dirubah sehingga beberapa poligon pertama bertanda '-' dan beberapa poligon terakhir bertanda '+'. Alternatif kelima adalah bagian ujung bertanda sama sedang bagian tengah bertanda lain. Karena sedemikian banyaknya alternatif maka sangat sulit untuk menentukan alternatif jika fairing poligon dilakukan secara otomatis. Sedangkan dengan cara manual pemakai program dapat menentukan pilihan alternatif dengan melihat gambar poligon.

Persamaan garis yang melalui dua titik dapat dinyatakan dengan :



Gbr 3.19 Garis melalui dua titik

$P=P_1+(P_3-P_1)t$ dimana $0 \le t \le 1$. Jika suatu titik P_2 (x_2,y_2) berada di antara P_1 dan P_3 dengan $x_1 \le x_2 \le x_3$ maka titik pada garis P dengan absis x_2 dapat didapat melalui :

$$x_2 = x_1+(x_3-x_1)t$$

x_1, x_2 dan x_3 diketahui sehingga diperoleh t :

$$t = (x_2- x_1)/(x_3-x_1)$$

dan t dimasukkan ke dalam persamaan garis :

$$y_2^* = y_1+(y_3-y_1)t$$

BAB IV
SUSUNAN DAN PENGUJIAN PROGRAM

4.1 STRUKTUR DAN PEMBAGIAN PROGRAM

Program fairing permukaan badan kapal ini disusun dengan menggunakan pemrograman Delphi dengan referensi

Program ini dibentuk dalam satu program utama, dalam Delphi disebut Project dan beberapa sub-program yang disebut unit dalam Delphi. Program utama berisi nama-nama dan lokasi sub-sub program serta mengatur urutan-urutan sub-program yang akan ditampilkan. Sub-sub program program ini berisi rutin-rutin yang akan digunakan oleh sub program lainnya. Pemecahan menjadi beberapa sub-program ini dimaksudkan untuk pengelompokan suatu sub-sub rutin untuk suatu massalah tertentu dalam suatu sub-program tersendiri. Hal tersebut akan memudahkan dalam penyusunan program, dalam pengecekan dan penelusuran bila terjadi kesalahan serta dalam pengembangan program.

Sub-sub rutin yang disusun dalam program ini dikelompokkan dalam unit-unit, antara lain :

- Basknot.pas

  Unit Basknot.pas berisi rutin-rutin untuk menghitung fungsi basis B-Spline dan knot vektor.

- BSpl2D.Pas

  Unit BSpl2D.pas berisi rutin-rutin untuk menghitung B-Spline curve fitting melalui titik data.

- Bsplfit.pas

  Unit BSplfit.pas berisi rutin-rutin untuk menghitung B-Spline surface fitting melalui titik data.

- CalcPoly.pas dan Calcpol2.pas

  Kedua unit ini berisi rutin-rutin untuk menghitung poligon melalui data parametris.

- CalcSurf.pas,smurf.pas dan Precasu.pas

  Ketiga unit ini berisi rutin-rutin untuk menghitung surface dari data poligon.

- Gaussclr.pas

  Unit Gaussclr.pas berisi rutin-rutin untuk menampilkan jendela yang berisi warna-warna yang mewakili suatu nilai gauss tertentu.

- IOData.pas

  Unit IOData.pas berisi rutin-rutin untuk keperluan input output data.

- Mother.pas

  Unit Mother.pas berisi rutin-rutin untuk menampilkan jendela utama.

- Newton.pas

  Unit Newton berisi rutin-rutin untuk menghitung akar dengan metode newton raphson.

- Polyfair.pas, PLChange.pas dan Vfair.pas

  Ketiga unit ini berisi rutin-rutin untuk menghitung dan menampilkan menu untuk perhitungan fairing poligon.

- Ta_Decl.pas

  Unit Ta_Decl berisi deklarasi tipe, konstanta dan variabel yang digunakan secara global oleh seluruh unit .

- View3D.pas dan V3Dproc.pas

  Unit View3D berisi rutin-rutin untuk menampilkan data dalam ruang tiga dimensi.

- View2Dpas

  Unit View2D.pas berisi rutin-rutin untuk menampilkan data dalam bidang dua dimensi.

Sedang program utama diberi nama HSDF, singkatan dari Hull Surface Definition and Fairing .

4.2 KETERBATASAN PROGRAM

Program yang telah disusun dalam Tugas Akhir ini memiliki beberapa keterbatasan sehingga tidak bisa digunakan untuk semua jenis bentuk badan kapal. Keterbatasan ini timbul dari keterbatasan metode B-Spline dalam mendefinisikan permukaan. Program ini tidak dapat membentuk permukaan yang diskontinu. Permukaan yang diskontinu yang dimaksudkan di sini adalah permukaan yang mengandung suatu titik atau bidang yang diskontinu pada turunan tertentu, antara lain seperti suatu knuckle atau lompatan bidang kelengkunggan.

Knuckle atau suatu sudut tajam pada lambung kapal banyak dijumpai antara lain pada kapal cepat yang menggunakan chine, pada kapal-kapal dengan skeg pada bagian buritan dan pada kapal dengan bentuk buritan transom.

4.3 KEKURANGAN PROGRAM

Program HSDF yang disusun di sini masih memiliki beberapa kekurangan. Disebut kekurangan karena hal-hal tersebut dapat diatasi dengan menambah dan mengembangkan program. Hanya saja karena keterbatasan waktu penyusunan program ini dan keterbatasan-keterbatasan lain maka hal itu tidak teratasi dan menjadi kekurangan dalam program ini.

Kekurangan-kekurangan itu antara lain :

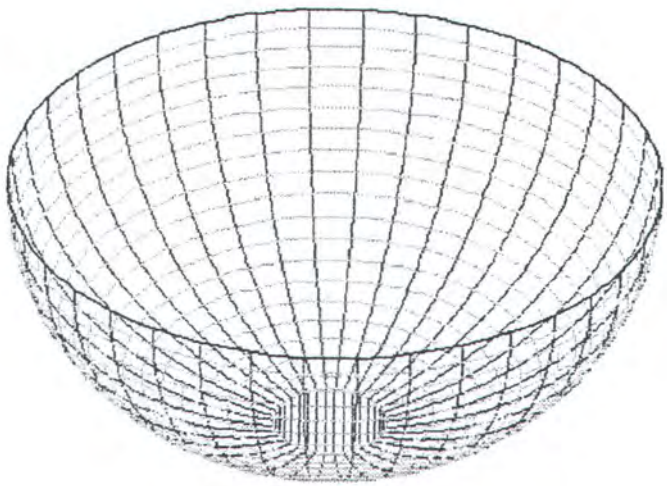
- Tidak tersedianya fasilitas untuk mencetak tampilan gambar dari program. Dalam hal ini juga tidak tersedianya fasilitas ekspor data gambar ke format grafik standar untuk dapat digunakan oleh software lain.

  Hal ini diabaikan dalam program ini karena mencetak tampilan bukanlah sesuatu yang vital dalam program ini. Program ini adalah program untuk keperluan analisa bukan program untuk keperluan design atau produksi sehingga pencetakan gambar tidak terlalu diperlukan sebagaimana pada program untuk keperluan design dan produksi.

- Kemampuan analisa dari persamaan permukaan yang telah terbentuk dengan program ini, hanya terbatas pada pembentukan potongan-potongan badan kapal yang berupa station, garis air dan buttock line.

4.4 PENGUJIAN PROGRAM

Pengujian program dilaksanakan untuk memperoleh dua hasil yaitu perubahan yang didapat setelah proses fairing dan ketepatan perhitungan kurvatur gaussian. Karena itu dalam pengujian program digunakan dua jenis data. Data pertama adalah data kapal "SEMERU' dari tugas rancang Lines Plan penulis dan data kedua adalah data setengah bola.

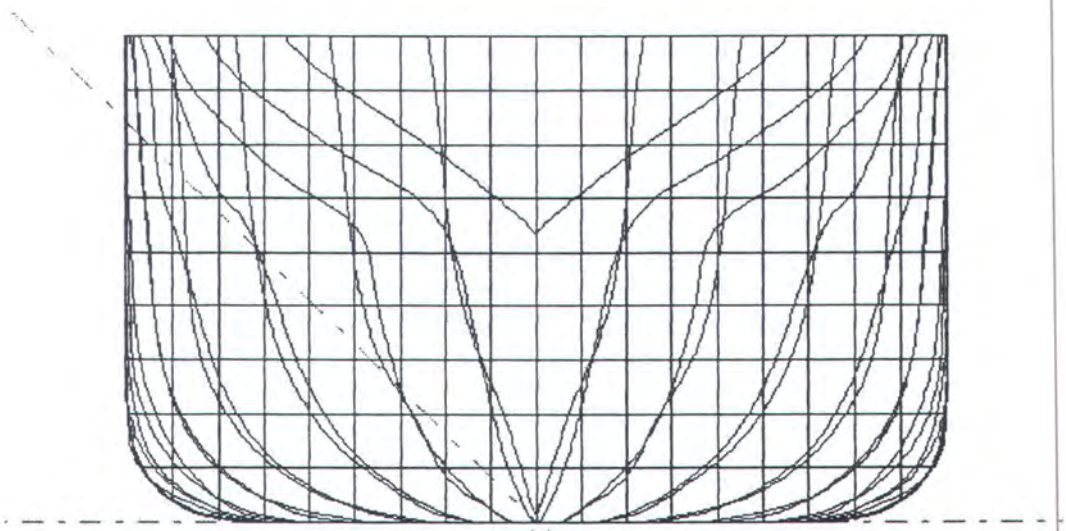


Gbr. 4. 1 Permukaan setengah bola

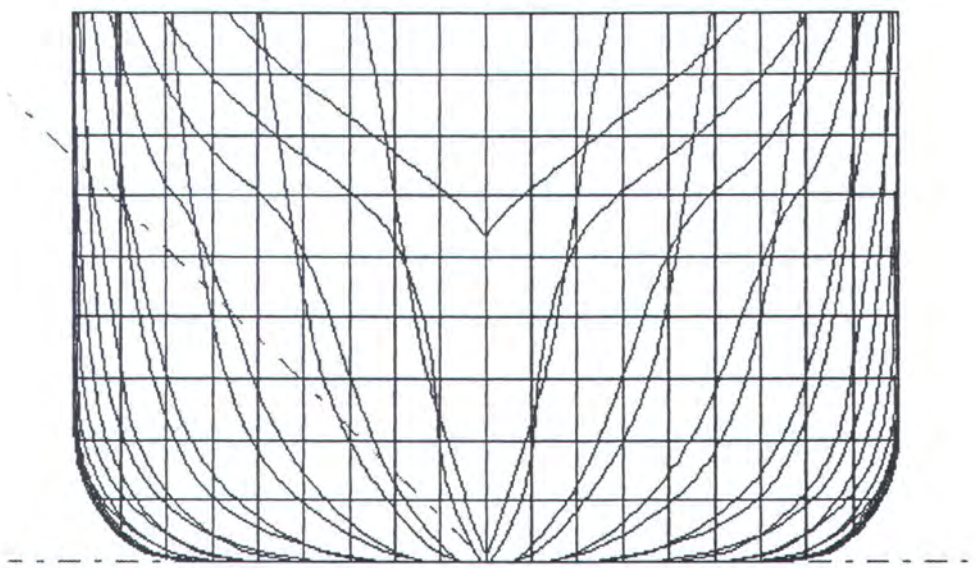
Data Setengah bola yang digunakan diukur pada setiap perubahan sudut azimuth sebesar 22.5° dan setiap perubahan sudut attitude 10° , dan permukaan yang dihasilkan ditunjukkan pada gbr 4.1. Untuk suatu permukaan bola setiap titik pada permukaannya memiliki kelengkungan yang sama sebesar 1/r dimana r jari-jari bola, maka kurvatur gaussian untuk setiap titik pada permukaan bola sesuai pers(2-) adalah $\kappa_{min} \cdot \kappa_{max} = 1/r^2$.

Data setengah bola yang digunakan mempunyai jari-jari 6 unit sehingga kurvatur gaussian permukaan setengah bola tersebut adalah 1/36 atau sebesar 0.02778. Hasil program memberikan harga kurvatur gaussian berkisar antara 0.026 s/d 0.029. Ketidaktepatan ini diakibatkan oleh ketidaktepatan metode B-Spline dalam menghasilkan permukaan yang benar-benar merupakan permukaan setengah bola.

Data Setengah lebar kapal digunakan untuk menghasilkan data poligon kapal. Data poligon ini dicoba untuk difairkan sehingga akan diperoleh data poligon baru. Melalui kedua data poligon ini yaitu poligon yang dihasilkan melalui perhitungan dan data poligon setelah difairkan, dibentuk persamaan permukaan B-Spline untuk menghasilkan permukaan kapal. Hasil fairing yang diperoleh tergantung dari perubahan poligon yang dilakukan oleh pemakai program karena proses fairing dilakukan secara manual. Gbr. 4.2 menunjukkan dua gambar badan kapal pandangan melintang dimana gbr.4.2(a) diperoleh dari poligon hasil perhitungan data setengah lebar kapal dan gbr.4.2(b) diperoleh melalui poligon yang telah difairkan.



(a)



(b)

Gbr.4.2 Body Plan dari data kapal "SEMERU" . (a) Tanpa fairing poligon. (b) Dengan fairing poligon

4.5 PENENTUAN KNOT VEKTOR

Kurva yang dihasilkan melalui fitting titik data relatif baik jika poligonyang diperoleh melalui perhitungan terletak tidak jauh dari titik data. Letak poligon yang diperoleh melalui perhitungan tergantung pada knot vektor yang dipilih. Jika digunakan knot vektor open uniform maka untuk data yang masing-masing jarak antar data yang berurutan relatif sama akan dihasilkan kurva yang baik tetapi untuk data yang masing-masing jarak antar data berurutan tidak beraturan, yang lebih sering dijumpai pada kenyataan, maka kemungkinan besar matriks fungsi basis [N] pada pers(2-10) kemungkinan besar pada diagonal utamanya mengandung nilai nol sehingga tidak dapat diinvers. Jadi knot vektor open uniform tidak dapat dipergunakan dalam penyusunan program . Demikian juga knot vektor periodik tidak dapat dipergunakan karena knot vektor ini digunakan untuk menghasilkan kurva-kurva periodik sedangkan kurva-kurva badan kapal bukan merupakan kurva periodik.

sehingga knot vektor yang digunakan dalam penyusunan program adalah knot vektor non uniform. Untuk n titik data dan order kurva k not vektornya adalah :

$$x = [\, x_1\ x_2\ .\ .\ x_k\ x_{k+1}\ x_{k+2}\ .\ .\ .\ x_{n+1}\ x_{n+2}\ x_{n+k-1}\,]$$

x_1 sampai dengan x_k bernilai 0 sedangkan x_{n+1} sampai dengan x_{n+k-1} bernilai maksimum yaitu n-k+1.

x_{k+1} sampai dengan x_n adalah sebagai berikut :

$$x_{k+i} = t_{j+i} + \frac{s}{2}\,(t_{j+i+1} - t_{j+i}) \qquad 1 \le i \le n-k$$

dimana

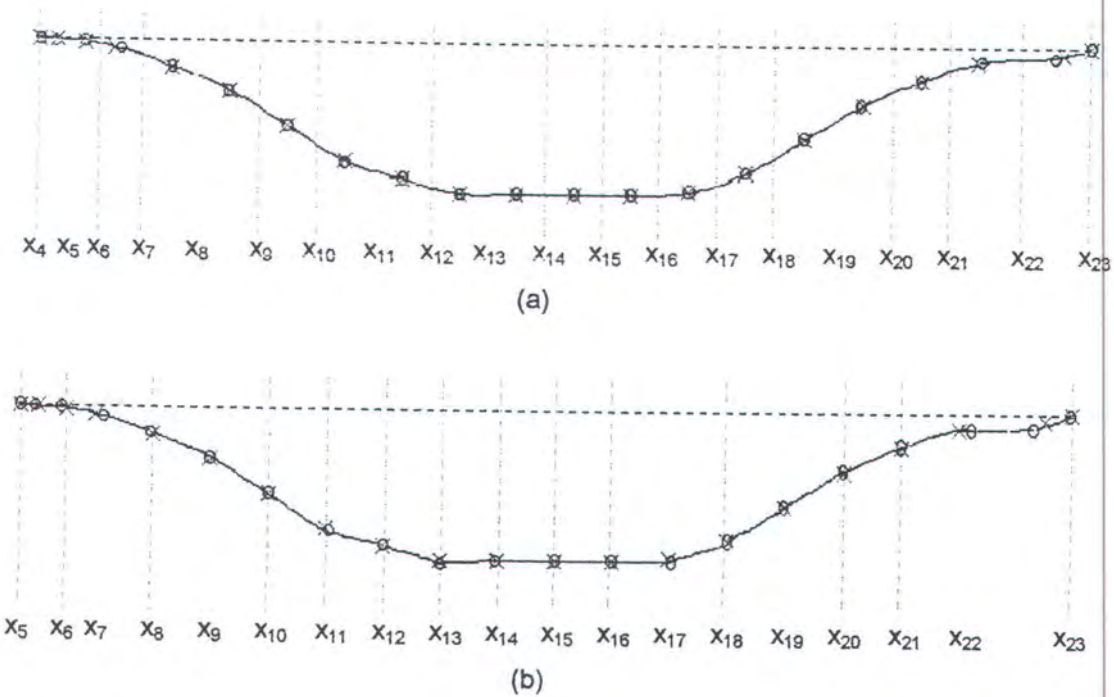
$j = k \text{ div } 2$

$s = k \bmod 2$

Operator div dan mod berarti hasil bagi dan sisa bagi.

t adalah parameter kurva pada titk data sesuai dengan pers (2-11)

Teknik penentuan knot vektor tersebut diperoleh dari pengamatan terhadap karakteristtk hubungan knot vektor dan poligon yang dihasilkan program pada



Gbr. 4.3 Titik data, Poligon dan knot vektor. Pada kurva , lingkaran menunjukkan titik data, tanda silang menunjukkan poligon dan garis putus-putus tegak menunjukkan knot vektor. (a) Dengan order k urva 3 (b). Dengan Order kurva 4

bermacam-macam jumlah poligon dan order kurva. Gbr. 4.3 menunjukkan hubungan poligon dan knot vektor untuk order kurva 3 dan 4.

Untuk order kurva 3 Poligon-poligon berada di tengah-tengah knot vektor dan hal ini juga berlaku untuk semua order kurva ganjil. Sedangkan untuk order kurva 4 dan genap maka poligon-poligon terletak tepat pada knot vektornya.

BAB V
KESIMPULAN DAN SARAN

5.1 KESIMPULAN

Penggunaan B-Spline surface fitting untuk membentuk persamaan permukaan badan kapal tidak bisa secara langsung diterapkan pada data setengah lebar kapal yang umum digunakan untuk pembentukan kurva-kurva badan kapal. Jika hal itu dilakukan maka bentuk permukaan akan tidak seperti yang diinginkan akibat adanya ketidaksesuaian antara data setengah lebar yang asli dengan knot vektor yang diambil.

Teknik yang berhasil membentuk persamaan permukaan badan kapal dengan baik adalah sebagai berikut :

- Menggunakan data parametris dua arah yang disusun dari data setengah lebar kapal yang sekaligus mencakup data bentuk haluan dan buritan.
- Menggunakan order 3
- Menggunakan knot vektor non uniform yang proporsional terhadap panjang lengkung kurva
- menggunakan jumlah poligon sama dengan jumlah titik data

Teknik ini hanya bisa diterapkan pada permukaan badan kapal yang tidak memiliki titik atau bidang yang diskontinu. Jadi tidak bisa untuk membentuk permukaan dengan chine yang dijumpai pada kapal planning, membentuk skeg atau membentuk buritan dengan transom.

Teknik fairing dipilih dengan memfairkan poligon permukaan secara manual. Dengan demikian hasil fairing yang akan diperoleh tergantung pada pemakai program. Fairing yang dilakukan bertujuan untuk memperbaiki bentuk permukaan tanpa suatu batasan tertentu seperti displasemen tetap, Koefisien blok tetap dan sebagainya.

5.2 SARAN PENGEMBANGAN

Program ini baru suatu program awal dari penggunaan persamaan permukaan untuk memodelkan badan kapal. Program ini hanya digunakan untuk membentuk model persamaan permukaan saja, belum menyertakan kemampuan untuk analisa dan manipulasi dari permukaan yang terbentuk. Oleh karena itu program-program kelanjutan dari program ini masih diperlukan yakni program-program yang mampu menganalisa persamaan permukaan yang terbentuk di sini untuk berbagai keperluan dalam proses design dan produksi kapal.

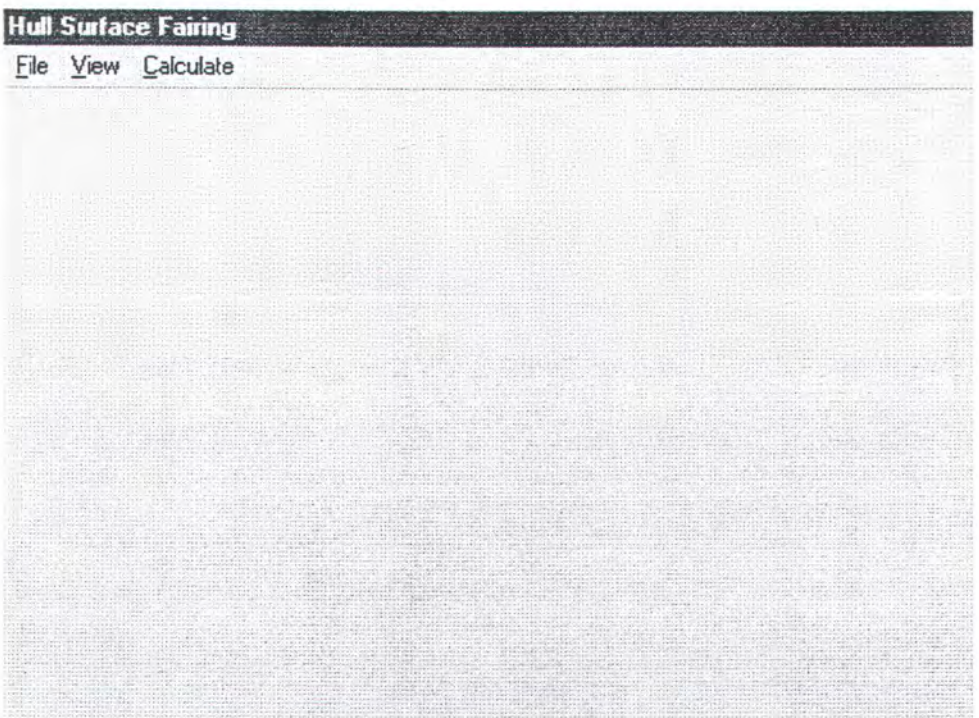
Pengembangan proses fairing secara otomatis dapat dilakukan dengan memberikan batasan berupa kecenderungan yang umum dijumpai pada badan kapal. Proses fairing juga dapat dikembangkan dengan batasan berupa konstrain pada suatu karakteristik kapal seperti displasemen, koefisien blok atau karakteristik kapal lainnya.

DAFTAR PUSTAKA

1. Rogers, David F. and Adams, J. Alan , **Mathematical Elements for Computer Graphics,** *McGraw-Hill,* New York, February 1989.
2. Baihaqqi, Mohammad, **Pendefinisian Persamaan Permukaan Badan Kapal dari Data Badan Kapal Dengan Metode B-Spline Menggunakan PC**, *Jurusan Teknik Perkapalan ITS,* Surabaya, 1995.
3. Nowacky, H., **Ship Lines Creation by Computer, Objectives, Methods and Results**, *Proceedings SCAHSD,* Anapolis, 1977
4. Hearn, Donald and Baker, M. Pauline, **Computer Graphics,** *Prentice-Hall,inc.* , New Jersey
5. Wozniewicz, Andrew J. , Shammas, Namir and Campbell, Tom, **Tech Your Self Delphi in 21 Days,** *Sam Publishing*,Indianapolis, 1995
6. Hill, Francis S., **Computer Graphics,** Macmilian Publishing, New York
7. Okianto, D, **Borland Delphi 2.0 for Windows 95,** *Elex Media Komputindo, Gramedia,* 1996.
8. Calvert, Charles, **Delphi 2 Unleashed,** *Sam Publishing,* Indianapolis, 1996

LAMPIRAN A : PETUNJUK PENGGUNAAN PROGRAM

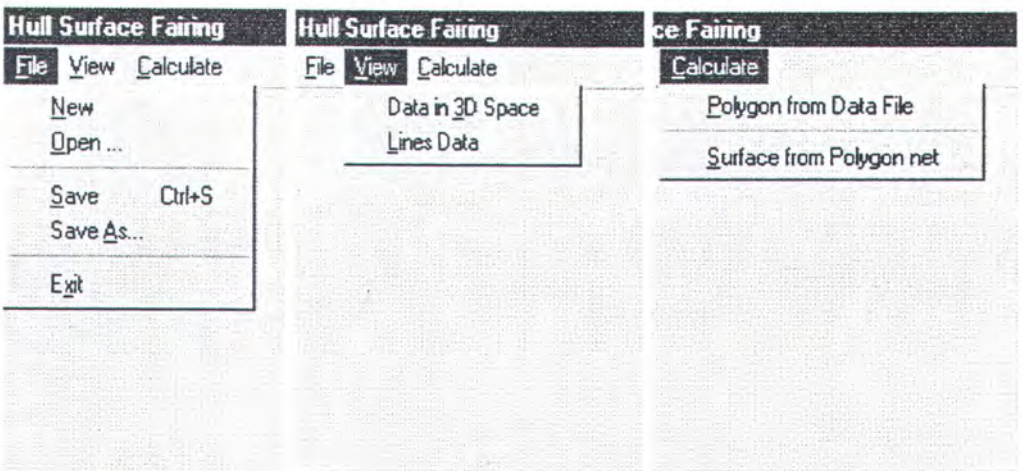
Setelah program dieksekusi (dijalankan) maka di layar akan terlihat jendela utama seperti pada gbr.A.1 . Jendela utama mempunyai menu utama File, View dan Calculate.



Gbr. A.1 Jendela utama program

Menu utama File mempunyai sub-menu New, Open, Save as, Save dan Exit seperti terlihat pada gbr.A.2.(a). Sub-menu New dipilih jika ingin membuat dan memasukkan data baru. Pemasukan data baru dimulai dengan mengisikan data jumlah station dan water line kemudian dilanjutkan dengan mengisi data half breadth.

Sub-menu Open berguna untuk membuka file data yang telah tersimpan dalam media penyimpan (hard disk atau disket).



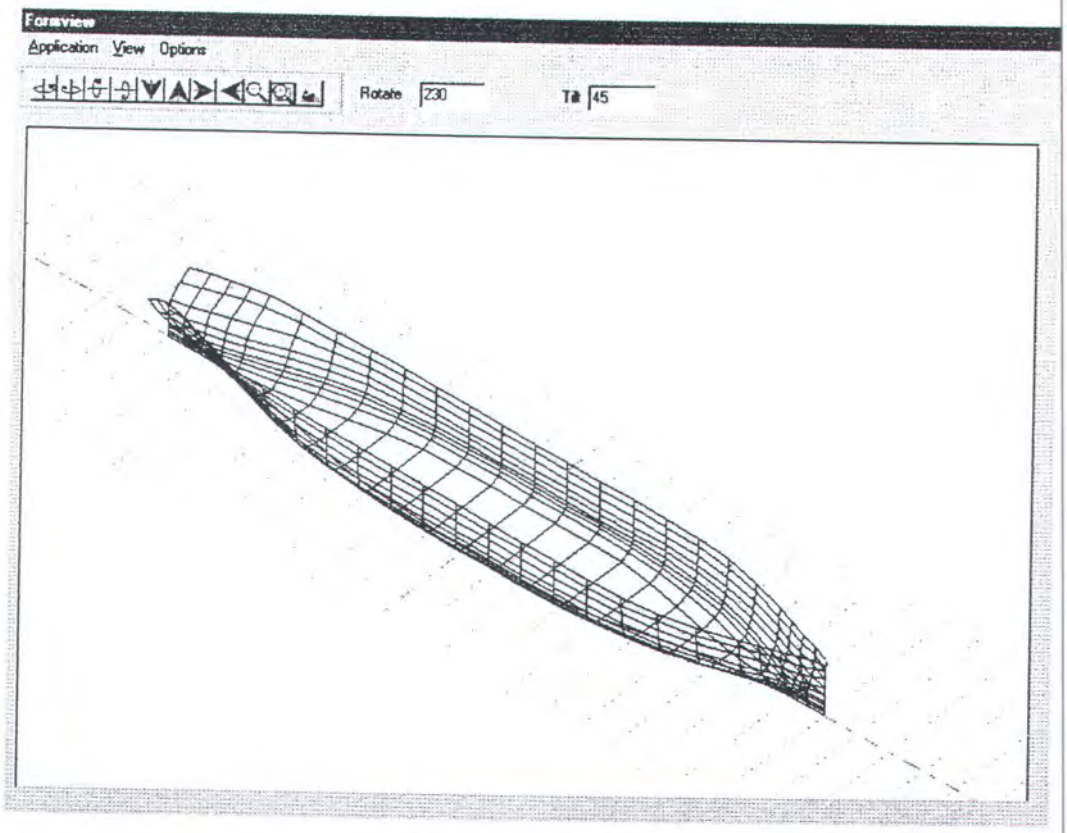
(a) (b) (c)

Gbr. A.2 Pembagian menu utama. (a). Menu File dan sub-menunya. (b) Menu View dan sub-menunya. (c). Menu Calculate dan sub-menunya.

Sub-menu Save as dan Save berguna untuk menyimpan data ke dalam media penyimpan. Sub-menu Save akan menyimpan data ke dalam nama file data yang sedang aktif, sedangkan sub-menu Save as akan menanyakan nama file tempat data akan disimpan. Sub-menu Exit digunakan untuk keluar dari program.

Menu utama View mempunyai dua sub-menu yaitu Data in 3D Space dan Lines seperti terlihat pada gbr.A.2.(b). Jika salah satu sub-menu ini dipilih maka akan ditampilkan jendela View seperti yang terlihat pada gbr.A.3. Jendela View memiliki 3 menu utama dan 11 bit button . Menu utama terdiri dari Application, View dan Options.

Menu utama jendela View application terdiri dari sub-menu Print dan sub-menu Exit. Sub-menu Print digunakan untuk mencetak gambar dengan printer, sedang sub-menu Exit dipakai untuk keluar dari jendela View.



Gbr.A.3 Jendela View

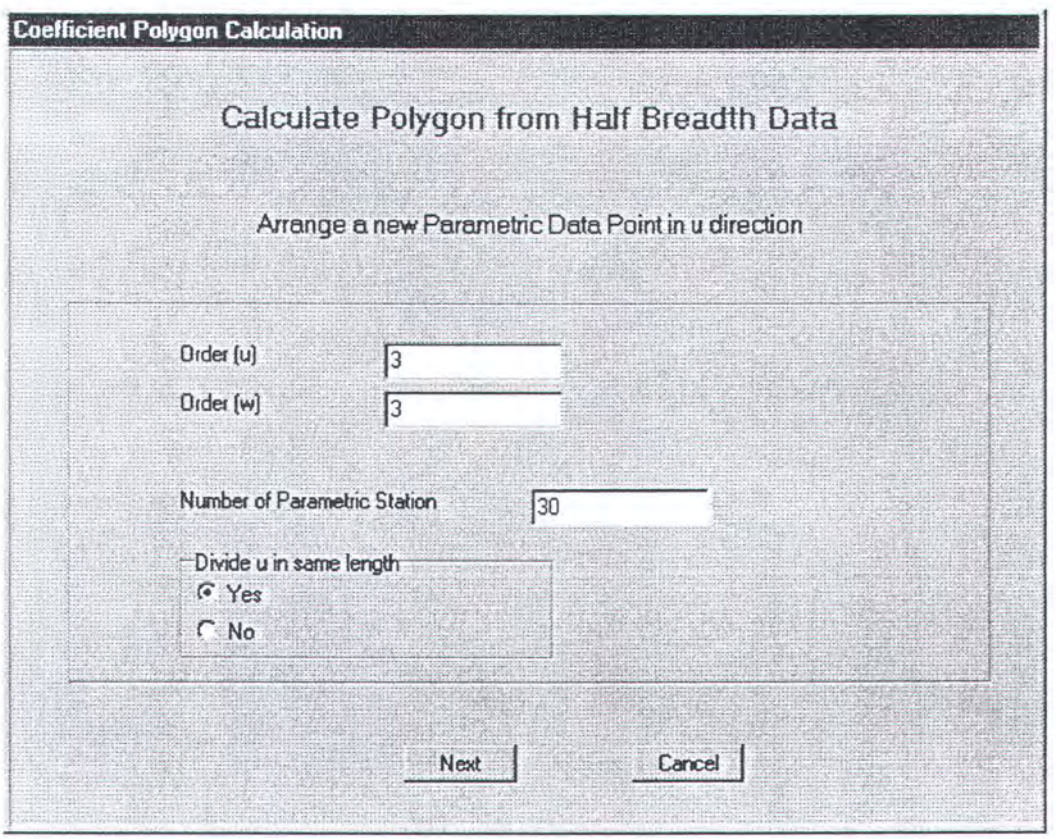
Menu utama jendela View View terdiri dari sub-menu Body Plan, Water Line, Buttock Line, Gaussian Curvature dan Polygon Vertices. Semua sub-menu ini dipakai untuk menampilkan dan menghilangkan body plan, Water Line, Buttock Line, Gaussian Curvature dan Polygon Vertices.

Bit button 1 dan 2 berguna untuk memutar gambar kapal dengan sumbu tegak kapal sebagai sumbu putar, sedang bit button 3 dan 4 berguna untuk memutar gambar kapal dengan sumbu memanjang kapal sebagai sumbu putar. Bit button 5, 6, 7 dan 8 digunakan untuk menggerakkan gambar sesuai dengan arah panah yang ditunjukkan bit button. Bit button 9 dan 10 digunakan untuk

memperkecil dan memperbesar gambar. Bit button 11 digunakan untuk menampilkan gambar setengah kapal atau penuh.

Menu utama Options terdiri dari sub-menu Select Polygon dan change poligon. Sub-menu select digunakan untuk memilih suatu daerah tertentu dari gambar. Setelah daerah tertentu dipilih, maka akan ditampilkan beberapa poligon yang berpengaruh pada daerah yang dipilih. Dengan sub-menu change pemakai program dapat memilih polygon yang akan dirubah.

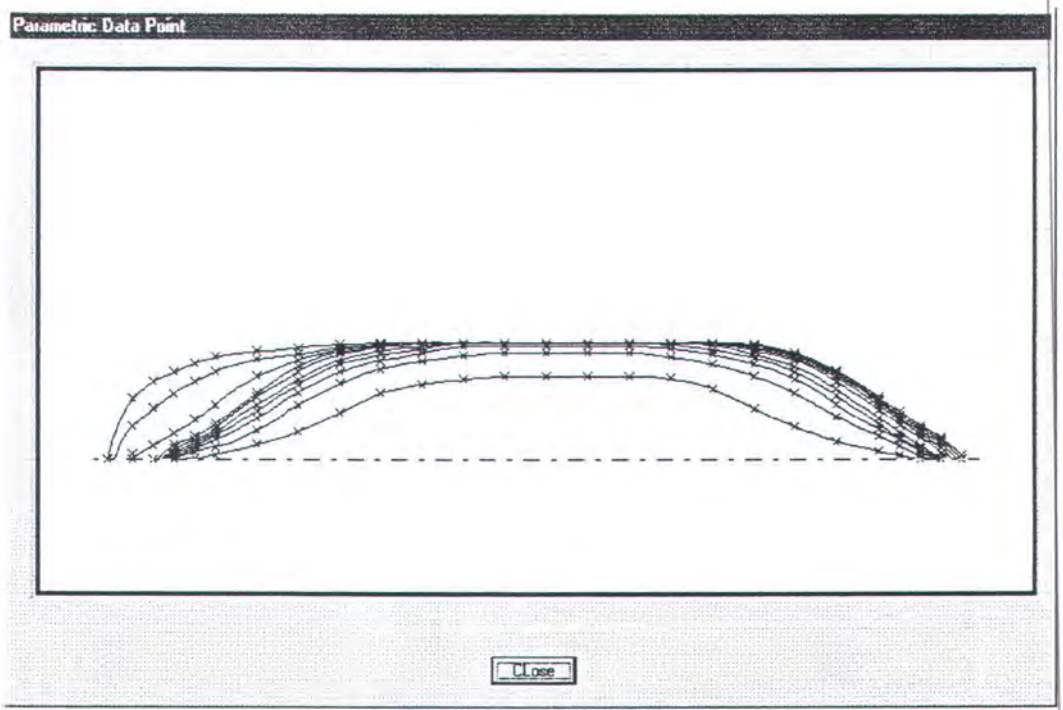
Menu utama Calculate dari jendela utama terdiri dari dua sub-menu yaitu Polygon from Data File dan Surface from Polygon seperti terlihat pada gbr.A.2.(c). Sub-menu from Data File dipilih jika pemakai program akan menghitung poligon dari data setengah lebar kapal. jika sub-menu ini dipilih maka



Gbr.A.4 Jendela perhitungan data parametris

akan ditampilkan jendela perhitungan data parametris seperti ditunjukkan gbr.A.4. Jendela ini berfungsi untuk menerima input yang akan dipakai dalam perhitungan data station parametris. Pada jendela ini terdapat tiga kotak input yaitu order (u) ,order (w) dan jumlah station parametris serta dua button yaitu button Next dan button Cancel. Button Next digunakan jika akan melanjutkan perhitungan sementara button Cancel untuk membatalkan perhitungan. Setelah kotak input diisi dan dipilih button Next maka akan ditampilkan jendela gambar data station parametris seperti terlihat pada gbr.A.5. Jendela ini hanya berfungsi untuk memperlihatkan gambar data station parametris karena itu hanya mempunyai satu button yaitu button Close untuk menutup jendela. Setelah jendela gambar data station parametris ditutup maka akan ditampilkan kembali jendela perhitungan data parametris seperti pada gbr.A.4 tetapi kotak input jumlah station parametris dirubah menjadi jumlah water line parametris.. Jendela ini berfungsi untuk menerima input untuk perhitungan data water line parametris. Setelah button Next dipilih maka akan ditampilkan kembali jendela pada gbr.A.5 tetapi gambarnya merupakan data waterline parametris. Jika jendela ini ditutup maka akan ditampilkan jendela perhitungan poligon seperti terlihat pada gbr.A.6. Jendela ini mempunyai dua button yaitu button Next untuk melanjutkan perhitungan dan button Cancel untuk membatalkan perhitungan. Jika dipilih button Next maka perhitungan poligon dimulai dan kemajuan proses perhitungan akan ditunjukkan dengan persen dalam kotak progress seperti terlihat pada gbr.A.7.

Jika perhitungan poligon telah selesai maka akan ditampilkan kotak Save dan pemakai program dapat memilih file untuk menyimpan data poligon.

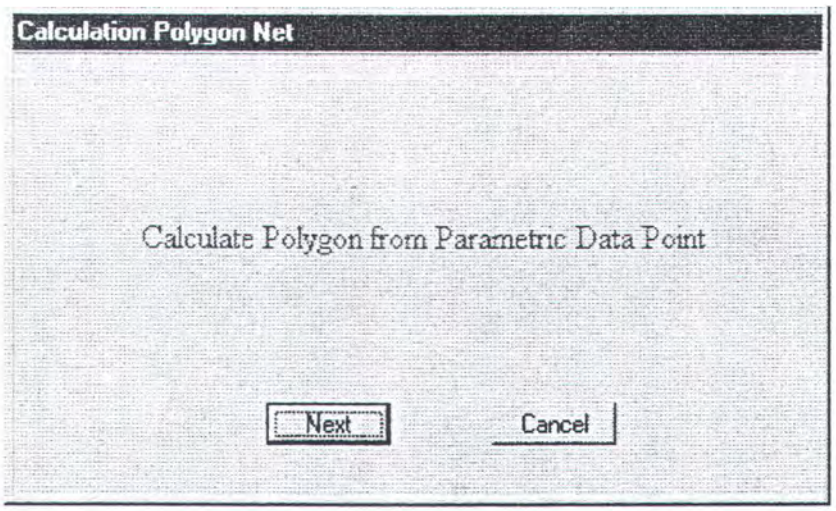


Gbr A.5 Data station parametris

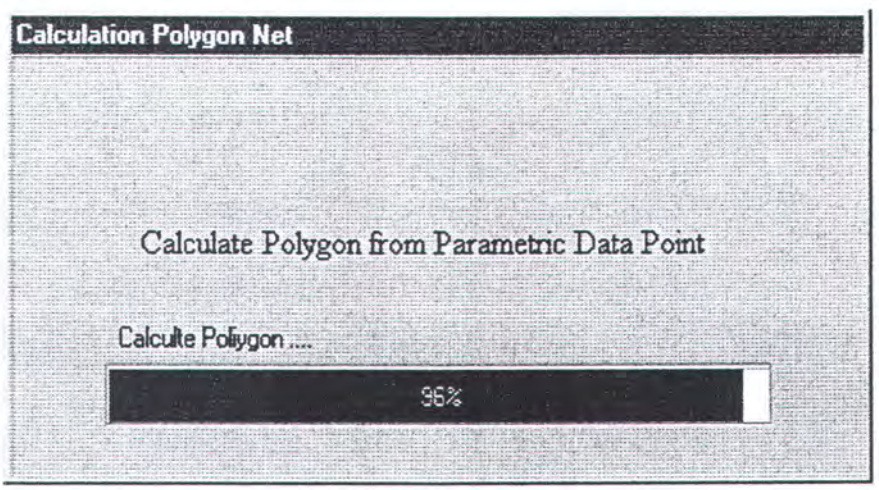
Sub menu Surface from Polygon pada jendela utama dipilih jika pemakai program akan menghitung permukaan dari file data poligon. Jika sub-menu ini dipilih maka akan ditampilkan kotak Open dan pemakai program dapat memilih file data poligon yang akan dipakai untuk perhitungan permukaan. Setelah file data poligon dipilih maka akan ditampilkan jendela perhitungan permukaan seperti terlihat pada gbrA.8.

Jendela perhitungan permukaan mempunyai 5 button. Button pertama adalah button Transverse Section (Body Plan). Button ini dipakai untuk melihat gambar salah satu potongan melintang (body plan). Jika button ini dipilih maka akan ditampilkan jendela Transverse Section seperti pada gbr.A.9.

Jendela Transverse Section mempunyai dua kotak input. Kotak input pertama adalah jarak ujung belakang kapal ke potongan melintang yang akan digambar, sedang kotak input kedua merupakan pembagian potongan melintang.



Gbr.A.6 Jendela Polygon Calculation

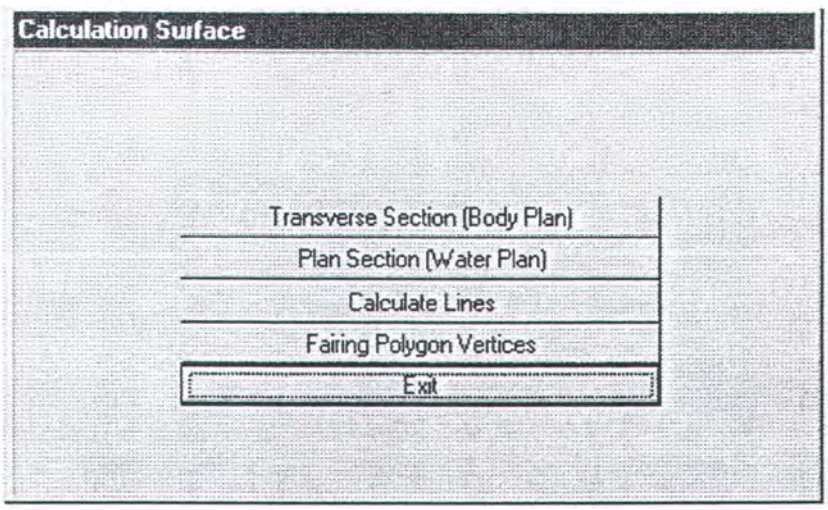


Gbr.A.7 Jendela Polygon Calculation dengan kotak kemajuan proses

Jendela Transverse Section juga memiliki dua button yaitu button OK untuk melihat gambar dan button Close untuk kembali ke jendela perhitungan permukaan.

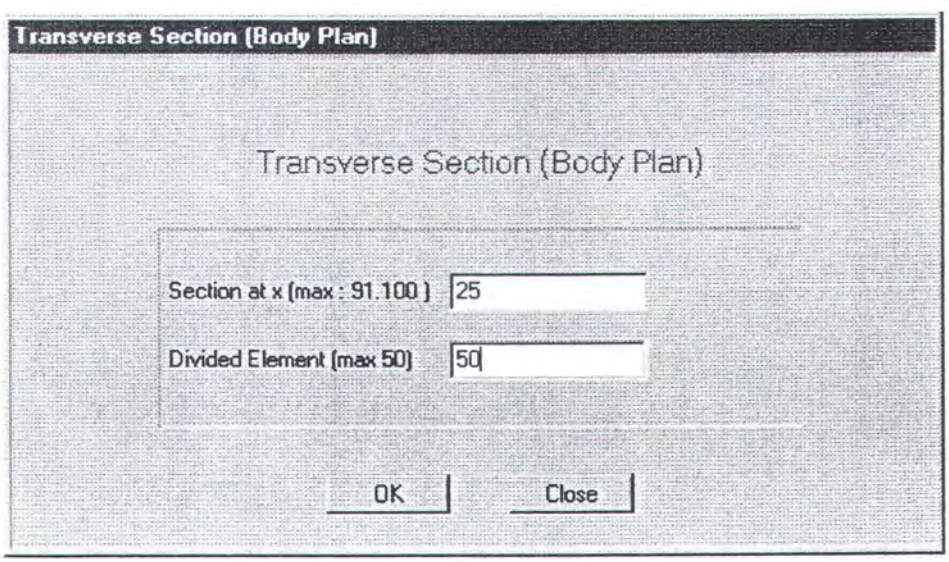
Button kedua dari jendela perhitungan permukaan adalah button Plan Section (Water Plan). Jika pemakai program memilih button ini maka akan ditampilkan jendela Plan Section. Bentuk jendela ini sama dengan bentuk jendela Transverse Section.

Button ketiga dari jendela perhitungan permukaan adalah button Calculate Lines yang berfungsi untuk menghitung lines dari kapal. Jika button ini dipilih maka akan ditampilkan jendela yang meminta input jumlah Station, jumlah Water Line dan jumlah Buttock Line. Setelah input ini dimasukkan maka akan ditampilkan jendela kemajuan perhitungan lines. Setelah perhitungan Lines selesai



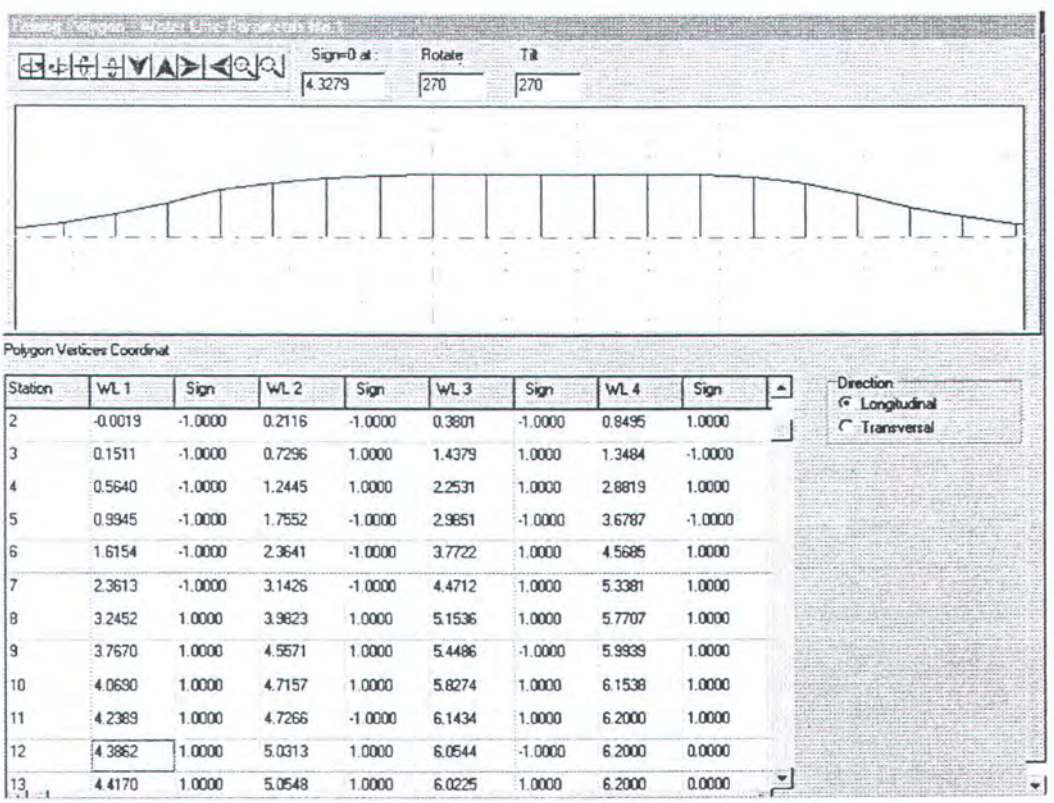
Gbr.A.8 Jendela perhitungan permukaan

maka akan ditampilkan kotak Save dan pemakai program dapat memilih File untuk menyimpan data Lines.



Gbr.A.9 Jendela Transverse Section

Button keempat dari jendela perhitungan permukaan adalah button Fairing Polygon Vertices yang berfungsi sebagai proses fairing poligon. Jika button ini dipilih maka akan ditampilkan jendela Fairing Polygon. Bentuk jendela ini adalah seperti gbr.A.10. Proses Fairing manual dapat dilakukan dengan mengedit



Station	WL 1	Sign	WL 2	Sign	WL 3	Sign	WL 4	Sign
2	-0.0019	-1.0000	0.2116	-1.0000	0.3801	-1.0000	0.8495	1.0000
3	0.1511	-1.0000	0.7296	1.0000	1.4379	1.0000	1.3484	-1.0000
4	0.5640	-1.0000	1.2445	1.0000	2.2531	1.0000	2.8819	1.0000
5	0.9345	-1.0000	1.7552	-1.0000	2.9851	-1.0000	3.6787	-1.0000
6	1.6154	-1.0000	2.3641	-1.0000	3.7722	1.0000	4.5685	1.0000
7	2.3613	-1.0000	3.1426	-1.0000	4.4712	1.0000	5.3381	1.0000
8	3.2452	1.0000	3.9823	1.0000	5.1536	1.0000	5.7707	1.0000
9	3.7670	1.0000	4.5571	1.0000	5.4486	-1.0000	5.9939	1.0000
10	4.0690	1.0000	4.7157	1.0000	5.8274	1.0000	6.1538	1.0000
11	4.2389	1.0000	4.7266	-1.0000	6.1434	1.0000	6.2000	1.0000
12	4.3862	1.0000	5.0313	1.0000	6.0544	-1.0000	6.2000	0.0000
13	4.4170	1.0000	5.0548	1.0000	6.0225	1.0000	6.2000	0.0000

Gbr.A.10 Jendela Fairing Polygon

langsung nilai koordinat poligon pada tabel gbr.A.10.

Button kelima dari jendela perhitungan permukaan adalah button Exit yang dipilih jika pemakai program akan keluar dari jendela tersebut dan kembali ke jendela utama.

LAMPIRAN B : LISTING PROGRAM

Program yang disusun dalam Tugas Akhir ini terdiri dari 19 unit dimana tiap unit terdiri dari ±700 baris. Karena program ini sangat besar dan sebagian besar dari isi program adalah untuk mengatur jendela dan tampilan, sehingga tidak semua unit disertakan dalam lampiran ini. Unit-unit yang disertakan adalah unit Basknot.pas untuk perhitungan knot vektor dan fungsi basisi B-Spline, unit Precasu.pas, untuk perhitungan permukaan dan kurvatur Gaussian, hanya disertakan bagian perhitungannya.

```
unit basknot;

interface
uses ta_decl;
Procedure OpenKnot(pts,kl:integer; var mknot:knottype);
Procedure knotc(pts,kl:integer; polytemp:polytt; var
cknot:knottype);
Procedure knotpar(pts,kl:integer; chordpar:stattype; var
cknot:knottype);
Procedure knotc2(pts,kl:integer; var cknot:knottype);
Procedure knotp(pts,kl:integer; var pknot:knottype);
Procedure Basis(kl,pts:integer ; t:single; mknot:knottype; var
bas:mattype);
Procedure Rbasis(kl,pts:integer ; t:single; mknot:knottype;
h:stattype; var bas:mattype);
Procedure DBasis(kl,pts:integer ; t:single; mknot:knottype; var
bas,dbas:mattype);
Procedure D2basis(kl,npts:integer;t:single;mknot:knottype;var
bas,dbas1,dbas2:mattype);

implementation
type
  co3dtype = array[1..3]of single;
var
  uarea : array[1..maxstat]of single;
  warea : array[1..maxwl]of single;

{========calculate open knot vector============}
Procedure OpenKnot(pts,kl:integer; var mknot:knottype);
var
  i:integer;
```

```
begin
   mknot[1]:=0;
   for i:=2 to pts+kl do
      if (i>kl) and (i<(pts+2)) then mknot[i]:= mknot[i-1] + 1
         else mknot[i] := mknot[i-1];
end;

{=========calculate non uniform open knot vector=================}
{=========proportional to the chord lengths======================}
Procedure knotc(pts,kl:integer; polytemp:polytt; var
cknot:knottype);
var
  i,j,n :integer;
  maxchord,sum : single;
  chord : array[1..maxstat] of single;

begin
   n:=pts-1;
  maxchord:=0;
  for i:=2 to pts do
  begin
    chord[i-1]:=sqrt(sqr(polytemp[1]^[i]-polytemp[1]^[i-1])
                +sqr(polytemp[2]^[i]-polytemp[2]^[i-1])
                +sqr(polytemp[3]^[i]-polytemp[3]^[i-1]));
    maxchord:=maxchord+chord[i-1];
  end;
  for i:=1 to kl do cknot[i]:=0;
  for i:=1 to n-kl+1 do
  begin
    sum:=0;
    for j:=1 to i do sum:=sum+chord[j];
    cknot[kl+i]:=((i/(n-kl+2))*chord[i+1]+sum)/maxchord*(n-kl+2);
  end;
  for i:=n+2 to pts+kl do cknot[i]:=n-kl+2;
end;

{============calculate non uniform open knot vector===============}
{==========proportional to the parameter chord lengths ===========}
Procedure knotpar(pts,kl:integer; chordpar:stattype; var
cknot:knottype);
var
  i,j,n :integer;
  maxchord,sum : single;
  chord : array[1..maxstat] of single;

begin
  n:=pts-1;
  maxchord:=1;
  for i:=2 to pts do chord[i-1]:=(chordpar[i]-chordpar[i-1]);
  for i:=1 to kl do cknot[i]:=0;
  for i:=1 to n-kl+1 do
  begin
```

```
    sum:=0;
    for j:=1 to i do sum:=sum+chord[j];
    cknot[kl+i]:=((i/(n-kl+2))*chord[i+1]+sum)/maxchord*(n-kl+2);
  end;
  for i:=n+2 to pts+kl do cknot[i]:=n-kl+2;
end;

{=====B-Spline basis function for open uniform knot vector===}
Procedure Basis(kl,pts:integer ; t:single; mknot:knottype; var
                bas:mattype);
var
   d,e : single;
   temp2 : array[1..maxstat] of single;
   pplusk,ic,c : integer;

begin
   for ic:=1 to pts+kl do temp2[ic]:=0;
   pplusk:=pts+kl;
   for ic:= 1 to pplusk-1 do
   if (t >= mknot[ic]) and (t < mknot[ic+1]) then temp2[ic] := 1
   else temp2[ic] := 0;
   for c:= 2 to kl do
   for ic := 1 to pplusk-c do
   begin
     if temp2[ic]<>0 then
     d:=((t-mknot[ic])*temp2[ic])/(mknot[ic+c-1]-mknot[ic])
     else d:= 0;
     if temp2[ic+1]<>0 then
     e:=((mknot[ic+c]-t)*temp2[ic+1])/(mknot[ic+c]-mknot[ic+1])
     else e:=0;
     temp2[ic]:=d+e;
   end;
   if t >= mknot[pplusk] then temp2[pts] := 1;
   for ic := 1 to pts do bas[1,ic]:=temp2[ic];
   if t >= mknot[pplusk] then bas[1,pts]:=1;
end;


{======Rational B-Spline basis function for open knot vector=======}
Procedure Rbasis(kl,pts:integer ; t:single; mknot:knottype;
                 h:stattype; var bas:mattype);
var
   sum,d,e : single;
   temp2 : array[1..maxstat] of single;
   pplusk,ic,c : integer;

begin
   for ic:=1 to pts+kl do temp2[ic]:=0;
   pplusk:=pts+kl;
   for ic:= 1 to pplusk-1 do
   if (t >= mknot[ic]) and (t < mknot[ic+1]) then temp2[ic]:=1
   else temp2[ic] := 0;
```

```
    for c:= 2 to kl do
    for ic := 1 to pplusk-c do
    begin
      if temp2[ic]<>0 then
      d:=((t-mknot[ic])*temp2[ic])/(mknot[ic+c-1]-mknot[ic])
      else d:= 0;
      if temp2[ic+1]<>0 then
      e:=((mknot[ic+c]-t)*temp2[ic+1])/(mknot[ic+c]-mknot[ic+1])
      else e:=0;
      temp2[ic]:= d+e;
    end;
    if t >= mknot[pplusk] then temp2[pts] := 1;
    sum:=0;
    for ic:=1 to pts do sum:=sum+(temp2[ic]*h[ic]);
    for ic:=1 to pts do
    if sum<>0 then bas[1,ic]:=temp2[ic]*h[ic]/sum
    else bas[1,ic]:=0;
    if t >= mknot[pplusk] then bas[1,pts]:=1*h[pts]/sum;
end;


{=======B-Spline basis function and the first derivative==========}
Procedure DBasis(kl,pts:integer ; t:single; mknot:knottype; var
                 bas,dbas:mattype);
var
    b1,b2,d1,d2,d3,d4     : single;
    temp1,temp2           : array[1..maxstat] of single;
    pplusk,ic,c           : integer;

begin
    for ic:=1 to pts+kl do
    begin
      temp1[ic]:=0;temp2[ic]:=0;
    end;
    pplusk:=pts+kl;
    for ic:= 1 to pplusk-1 do
    if (t >= mknot[ic]) and (t < mknot[ic+1]) then temp1[ic] := 1
    else temp1[ic] := 0;
    if t>=mknot[pplusk] then temp1[pts]:=1;
    for c:= 2 to kl do
    for ic := 1 to pplusk-c do
    begin
      if temp1[ic]<>0 then
      b1:=((t-mknot[ic])*temp1[ic])/(mknot[ic+c-1]-mknot[ic])
      else b1:= 0;
      if temp1[ic+1]<>0 then
      b2:=((mknot[ic+c]-t)*temp1[ic+1])/(mknot[ic+c]-mknot[ic+1])
      else b2:=0;
      if temp1[ic]<>0 then d1:=temp1[ic]/(mknot[ic+c-1]-mknot[ic])
      else d1:=0;
      if temp1[ic+1]<>0 then
      d2:=-temp1[ic+1]/(mknot[ic+c]-mknot[ic+1])
```

```
      else d2:=0;
      if temp2[ic]<>0 then
      d3:=((t-mknot[ic])*temp2[ic])/(mknot[ic+c-1]-mknot[ic])
      else d3:=0;
      if temp2[ic+1]<>0 then
      d4:=((mknot[ic+c]-t)*temp2[ic+1])/(mknot[ic+c]-mknot[ic+1])
      else d4:=0;
      temp1[ic]:= b1+b2;
      temp2[ic]:= d1+d2+d3+d4;
    end;
    if t>=mknot[pplusk] then temp1[pts]:=1;
    for ic:=1 to pts do bas[1,ic]:=temp1[ic];
    for ic := 1 to pts do dbas[1,ic]:=temp2[ic];
end;

{=======B-Spline basis function and the first derivative==========}
Procedure D2basis(kl,npts:integer;t:single;mknot:knottype;var
                  bas,dbas1,dbas2:mattype);
var
    b1,b2,d1,d2,d3,d4,s1,s2,s3,s4 : single;
    temp,temp1,temp2      : array[1..100] of single;
    pplusk,ic,c           : integer;

begin
    pplusk:=npts+kl;
    for ic:=1 to npts+kl do
    begin
      temp[ic]:=0;
      temp1[ic]:=0;
      temp2[ic]:=0;
    end;
    {calculate first order basis function}
    for ic:= 1 to pplusk-1 do
    if (t >= mknot[ic]) and (t < mknot[ic+1]) then temp[ic]:=1
    else temp[ic] := 0;
    if t>=mknot[pplusk] then temp[npts]:=1;
    {calculate higher order basis function and their derivatives}
    for c:= 2 to kl do
    begin
      for ic := 1 to pplusk-c do
      begin
        {calculate basis function}
        if temp[ic]<>0 then
        b1:= ((t-mknot[ic])*temp[ic])/(mknot[ic+c-1]-mknot[ic])
        else b1:= 0;
        if temp[ic+1]<>0 then
        b2:=((mknot[ic+c]-t)*temp[ic+1])/(mknot[ic+c]-mknot[ic+1])
        else b2:=0;
        {calculate first derivative}
        if temp[ic]<>0 then
        d1:=temp[ic]/(mknot[ic+c-1]-mknot[ic])
        else d1:=0;
```

```
        if temp[ic+1]<>0 then
        d2:=-temp[ic+1]/(mknot[ic+c]-mknot[ic+1])
        else d2:=0;
        if temp1[ic]<>0 then
        d3:=((t-mknot[ic])*temp1[ic])/(mknot[ic+c-1]-mknot[ic])
        else d3:=0;
        if temp1[ic+1]<>0 then
        d4:= ((mknot[ic+c]-t)*temp1[ic+1])/(mknot[ic+c]-mknot[ic+1])
        else d4:=0;
        {calculate SECOND derivative}
        if temp1[ic]<>0 then
        s1:=2*temp1[ic]/(mknot[ic+c-1]-mknot[ic])
        else s1:=0;
        if temp1[ic+1]<>0 then
        s2:=-2*temp1[ic+1]/(mknot[ic+c]-mknot[ic+1])
        else s2:=0;
        if temp2[ic]<>0 then
        s3:=(t-mknot[ic])*temp2[ic]/(mknot[ic+c-1]-mknot[ic])
        else s3:=0;
        if temp2[ic+1]<>0 then
        s4:= (mknot[ic+c]-t)*temp2[ic+1]/(mknot[ic+c]-mknot[ic+1])
        else s4:=0;
        temp[ic]:=b1+b2;
        temp1[ic]:=d1+d2+d3+d4;
        temp2[ic]:=s1+s2+s3+s4;
      end;
    end;
    {put in arrays}
    for ic:=1 to npts do
    begin
      bas[1,ic]:=temp[ic];
      dbas1[1,ic]:=temp1[ic];
      dbas2[1,ic]:=temp2[ic]
    end;
end;

end.



unit PreCaSu;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  ExtCtrls, StdCtrls,ta_decl, DBTables, Db, Grids,
DBGrids,newton,basknot, Gauges,bsplfit;
```

```
procedure calclines;
Procedure Bsplinegauss;


implementation

{==========Gaussian Curvature Calculation==========}
Procedure Bsplinegauss;
var
   g,h,j1,j2,j3,icon                  : integer;
   u,w,u1,u2,w1,w2,wplus,uplus : single;
   AA,BB,CC,QuxQw4                : single;
   Q,Qu,Qw,Quw,Quu,Qww,QuxQw      : array[1..3] of single;

begin
  new(gvcor);
  u1:=0;u2:=1;w1:=0;w2:=1;
  nplusc := jlh_sta + orderk;
  mplusc := jlh_WL + orderl;
  icon:=0;
  u1:=u1*(jlh_sta-orderk+1);u2:=u2*(jlh_sta-orderk+1);
  w1:=w1*(jlh_WL-orderl+1);w2:=w2*(jlh_WL-orderl+1);
  u:=u1;
  uplus:=(u2-u1)/(gnum-1);
  wplus:=(w2-w1)/(gprt-1);
  for g:=1 to gnum do
  begin
    u:=(g-1)*uplus;
    D2Basis(orderk,jlh_sta,u,x,nbasis,ND1BASIS,ND2BASIS);
    for h:=1 to gprt do
    begin
      num:=(g-1)*gprt+h;
      w:=(h-1)*wplus;
      for j1:=1 to 3 do
      begin
        q[j1]:=0;
        qu[j1]:=0;
        qw[j1]:=0;
        quw[j1]:=0;
        quu[j1]:=0;
        qww[j1]:=0;
      end;
      D2Basis(orderl,jlh_WL,w,y,mbasis,MD1BASIS,MD2BASIS);
      inc(icon,1);
      gvcor^[icon].paru:=u;
      gvcor^[icon].parw:=w;
      for j1:=1 to jlh_sta do
      for j2:=1 to jlh_WL do
      begin
        for j3:=1 to 3 do
        begin
          q[j3]:=q[j3]+b[j3]^[(j1-1)*jlh_WL
```

```
                +j2]*nbasis[1,j1]*mbasis[1,j2];
          qu[j3]:=qu[j3]+b[j3]^[(j1-1)*jlh_WL
                  +j2]*nd1basis[1,j1]*mbasis[1,j2];
          qw[j3]:=qw[j3]+b[j3]^[(j1-1)*jlh_WL
                  +j2]*nbasis[1,j1]*md1basis[1,j2];
          quw[j3]:=quw[j3]+b[j3]^[(j1-1)*jlh_WL
                   +j2]*nd1basis[1,j1]*md1basis[1,j2];
          quu[j3]:=quu[j3]+b[j3]^[(j1-1)*jlh_WL
                   +j2]*nd2basis[1,j1]*mbasis[1,j2];
          qww[j3]:=qww[j3]+b[j3]^[(j1-1)*jlh_WL
                   +j2]*nbasis[1,j1]*md2basis[1,j2];
        end;
      end;
      QuxQw[1]:=qu[2]*qw[3]-qu[3]*qw[2];
      QuxQw[2]:=qu[3]*qw[1]-qu[1]*qw[3];
      QuxQw[3]:=qu[1]*qw[2]-qu[2]*qw[1];
      QuxQw4:=sqr(sqr(QuxQw[1])+sqr(QuxQw[2])+sqr(QuxQw[3]));
      AA:=QuxQw[1]*Quu[1]+QuxQw[2]*Quu[2]+QuxQw[3]*Quu[3];
      BB:=QuxQw[1]*Quw[1]+QuxQw[2]*Quw[2]+QuxQw[3]*Quw[3];
      CC:=QuxQw[1]*Qww[1]+QuxQw[2]*Qww[2]+QuxQw[3]*Qww[3];
      if QuxQw4<>0 then
      begin
        gvcor^[icon].warna:=(AA*CC-sqr(BB))/QuxQw4;
      end
      else
      begin
        gvcor^[icon].warna:=0;
      end;
      gvcor^[icon].x:=q[1];
      gvcor^[icon].y:=q[2];
      gvcor^[icon].z:=q[3];
    end;
  end;
end;


{==================Lines Calculation====================}
procedure calclines;
var
  h,i,j,jmod,blprth : integer;
  fact2,bord,bordb,bordc,bordd,wpos,factor,tinggi,gaussect : single;
  totalhit,totalhit1:integer;
  callpaint:boolean;
  condition : char;
begin
  bdprt:=100;wlprt:=100;blprt:=100;
  blprth:=round(blprt/2);blprt:=blprth*2;
  for i:=1 to 3 do new(bdcor[i]);
  for i:=1 to bdnum do
  begin
    searchsk(2,bdsect[i],bordb);
```

```
  if bdsect[i]<p[1]^[1].pl then searchSS(1,1,bdsect[i],bord)
  else if bdsect[i]>p[1]^[(po1-1)*po2+1].pl then
       searchSS(1,2,bdsect[i],bord)
  else searchsk(1,bdsect[i],bord);
  for j:=1 to bdprt do
  begin
    num:=(i-1)*bdprt+j;
    bdcor[1]^[num]:=bdsect[i];
    bdcor[3]^[num]:=((bordb-bord)/(bdprt-1))*(j-1)+bord;
    changepo1po2:=false;
    Searchy(bdcor[1]^[num],bdcor[3]^[num],bdcor[2]^[num]);
    while totaliterasi<>0 do
    begin
      changepo1po2:=true;
      po1:=po1+2;po2:=po2+2;
      for h:=1 to 3 do dispose(p[h]);
      bspline;
      Searchy(bdcor[1]^[num],bdcor[3]^[num],bdcor[2]^[num]);
    end;
    if changepo1po2 then
    begin
      po1:=10;po2:=25;
      for h:=1 to 3 do dispose(p[h]);
      bspline;
    end;
  end;
end;
for i:=1 to 3 do new(wlcor[i]);
for i:=1 to wlnum do
begin
  searchSS(2,1,wlsect[i],bord);
  searchSS(2,2,wlsect[i],bordb);
  for j:=1 to wlprt do
  begin
    num:=(i-1)*wlprt+j;
    wlcor[3]^[num]:=wlsect[i];
    wlcor[1]^[num]:=((bordb-bord)/(wlprt-1))*(j-1)+bord;
    changepo1po2:=false;
    Searchy(wlcor[1]^[num],wlcor[3]^[num],wlcor[2]^[num]);
    while totaliterasi<>0 do
    begin
      changepo1po2:=true;
      po1:=po1+2;po2:=po2+2;
      for h:=1 to 3 do dispose(p[h]);
      bspline;
      Searchy(wlcor[1]^[num],wlcor[3]^[num],wlcor[2]^[num]);
    end;
    if changepo1po2 then
    begin
      po1:=10;po2:=25;
      for h:=1 to 3 do dispose(p[h]);
      bspline;
```