UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA

ENGENHARIA MECÂNICA

PROJETO MECÂNICO

TRABALHO DE FORMATURA – 1998

MOTOR – FOGUETE

(Parte II)

Projeto unidimensional e bidimensional de bocal supersônico

Orientador : Prof. Dr. Marcos Pimenta

Aluno: Ulisses Maximiano Nº USP : 1763621

Resumo

Desenvolvimento de um programa para análise bidimensional do escoamento supersônico em um bocal de foguete através do método das características. Comparação com a análise unidimensional e isoentrópica em um projeto de motor-foguete.

Agradecimentos

Agradeço aos professores Marcos Pimenta e Euryale Zerbini pelos conselhos, por todo o incentivo e motivação na área de Energia e Fluidos e pelo apoio nas horas difíceis.

Aos amigos e familiares que me apoiaram, incentivaram e suportaram durante a jornada deste e dos outros anos no curso de Engenharia Mecânica, o meu eterno agradecimento.

Em memória de minha avó Olimpia de Jesus Ferreira

Índice

Simbologia

a - velocidade do som no meio em questão(gás de combustão);

acel - aceleração do foguete;

A - área;

$\mathbf{C_p}$ - calor específico a pressão constante;

$\mathbf{C_v}$ - calor específico a volume constante;

D - diâmetro de uma seção;

F - força de empuxo;

g - constante gravitacional;

H - altura alcançada;

$\mathbf{K^{\pm}}$ - constantes características; constantes ao longo das linhas de Mach;

m - massa foguete;

$\mathbf{mw^{\pm}}$ - linhas de Mach ou linhas características;

M - número de Mach (M=V/a);

P - pressão absoluta do gás;

R - constante do gás;

r - raio;

T - temperatura do gás;

$\mathbf{V}$ - velocidade do gás em relação ao bocal;

$\mathbf{x,y}$ - coordenadas cartesianas;

$\mathbf{w}$ - vazão de gás em massa por unidade de tempo;

μ - ângulo de Mach;

ρ - densidade do gás;

θ - ângulo entre o vetor velocidade V e o eixo x, aqui considerado como eixo de simetria para o escoamento;

ν - função de Prandtl-Meyer

γ - razão de calor específico($\gamma=C_p/C_v$), aqui chamada de constante específica do gás;

I) Introdução

A motivação para este trabalho foi a idéia de fazer um pequeno foguete para a agricultura. Este foguete levaria uma carga de sal que, espalhada nas nuvens, poderia precipitar o processo de chuva em locais e/ou épocas com pequena incidência ou simplesmente evitar chuva de granizo sobre a lavoura.

Dois componentes são básicos num projeto de motor-foguete:

- a câmara de combustão e o grão de combustível no seu interior
- o bocal de escoamento dos gases de combustão

Na câmara de combustão se processa a queima do combustível, gerando os gases de propulsão em pressão e temperatura muito altas. É um processo em regime não permanente que depende muito da geometria do bocal para poder dimensionar uma câmara que resista às pressões acumuladas durante a queima. Para se ter uma idéia, com pressões de projeto de 100bar podemos ter picos de até 300bar dentro da câmara. Entretanto este estudo não será alvo deste projeto e para maior simplicidade adotaremos uma pressão de projeto interna à câmara como constante e a usaremos como pressão de estagnação à entrada do bocal.

No bocal se processa a aceleração dos gases de combustão que, ao deixarem o foguete em alta velocidade, geram uma força propulsora (principio da ação e reação). Simplificadamente, trocamos as altas pressões e temperaturas(velocidade dos gases quase nula) por alta velocidade na saída (e menores pressões e temperaturas). O bocal é composto de duas partes básicas. A primeira é um convergente onde a área de escoamento

diminui até um mínimo, chamado de garganta. Nesse ponto temos um limite de vazão em função das condições de entrada do bocal ;quando atingimos esse limite dizemos que o bocal está blocado, a velocidade dos gases chega à velocidade do som(chamado de Mach=1) e a vazão é máxima para as condições de entrada. A segunda parte do bocal é um divergente, que quando o bocal está blocado atua como um multiplicador de Mach, fazendo com que a velocidade dos gases atinja valores bem maiores que a velocidade do som(Mach>1). Em contrapartida, a pressão e a temperatura sofrem as maiores quedas nessa parte, chegando a pressão de saída a ser menor que a pressão atmosférica. O alvo principal deste estudo será o dimensionamento do perfil do divergente. É conveniente lembrar que embora possamos ter pressões de saída mais altas que a atmosférica e auxiliando no processo de empuxo do foguete, vamos procurar fazer essas pressões praticamente iguais.

Inicialmente projetamos o bocal unidimensional, usando diretamente as relações isoentrópicas entre temperatura, pressão, velocidade, mach, área e densidade. Um dos requisitos é o alcance vertical do foguete ou, diretamente, o empuxo do bocal. Foi usado o software MathCad nesta etapa. Em seguida foi desenvolvido um programa em Delphi Pascal para o projeto bi-dimensional, usando o método das características. Os resultados foram então comparados com a bibliografia.

II) Objetivos:

Pretende-se fornecer um programa e uma sistemática de projeto que permita construir foguetes ou túneis de vento supersônicos. Os túneis de vento exigem perfis de escoamento paralelos na saída para os projetos aerodinâmicos. Já os bocais de foguete devem ter o mínimo comprimento (mínima massa) para um máximo empuxo.

A bibliografia oferece extenso material sobre o método das características, mas não existe um programa que possa mostrar a influência do Mach de saída sobre o tamanho do perfil do bocal em termos dimensionais e visuais. É este o objetivo deste trabalho: fornecer um programa como uma das ferramentas de um projeto completo de foguete.

O perfil bidimensional do bocal abre espaço para uma análise mais precisa do escoamento usando o método das diferenças finitas. Da mesma forma permite a construção de um protótipo de foguete com auxilio de ferramentas computadorizadas de usinagem.

Na verdade, a operação de um foguete é toda um transitório, desde a queima do grão de combustível até o vôo. Um projeto futuro de foguete completo deve levar em conta esse transitório e adaptar o programa aqui desenvolvido para um cálculo progressivo em marcha de tempo. São muitas as questões a serem resolvidas num projeto de tal envergadura: aerodinâmica do foguete, controle, operação da ogiva de carga, variação do centro de massa do foguete, material de construção, variação das condições na câmara de combustão e consequentemente das condições na entrada do bocal, variação das condições do meio de saída dos gases(atmosfera), etc.

Como já dito, vamos apenas desenvolver e apresentar a ferramenta que auxilia no projeto do bocal. Embora o cálculo unidimensional de projeto que será apresentado com o auxílio do programa Mathcad da MathSoft seja válido para o bocal completo, o desenvolvimento do programa Bocal em Delphi Pascal, pelo método das características, se aplica apenas à parte supersônica do escoamento, a partir da garganta do bocal.

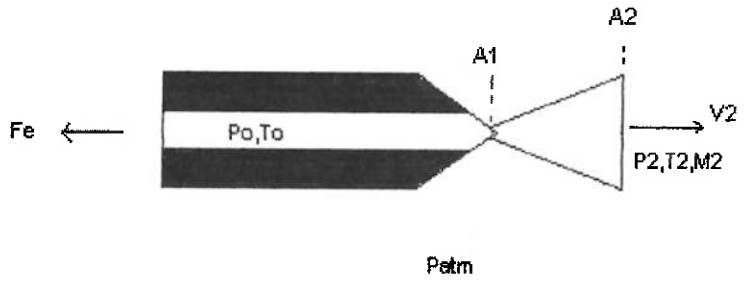
III)Parte A: Projeto Unidimensional

Para este parte assumimos as seguintes simplificações:

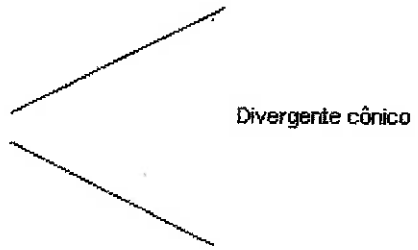
- bocal adiabático e isoentrópico;
- escoamento contínuo e permanente;
- condições de entrada constantes e conhecidas;
- pressão atmosférica constante;
- incremento de área **dA** constante para o perfil convergente e para o perfil divergente;
- não há escoamento através das paredes do bocal;
- não há reações químicas dos gases durante o escoamento no bocal;
- o gás de combustão atende à lei dos gases perfeitos;

A altura alcançada pelo foguete é apenas uma referência, levando em conta que consideramos a trajetória perfeitamente vertical, desprezamos a resistência do ar e usamos uma massa total de foguete estimada. Não levamos em conta também a perda de massa total pela subtração gradual do combustível queimado. O vôo de um foguete deste tipo é dividido em duas partes : propelido e inercial. A primeira se dá durante a queima do combustível e não costuma durar mais que 5 segundos. A segunda usa a velocidade inercial da primeira etapa contra a aceleração da gravidade.

Simplificadamente, a forma do motor-foguete é :



A parte escura representa o grão combustível.



As fórmulas usadas aqui que usamos aqui são derivadas das relações de escoamento isoentrópico de um gás perfeito, da continuidade, conservação do momento e da cinemática.

As grandezas envolvidas nos cálculos são:

T_0 – temperatura do gás de combustão na câmara, admitida como temperatura de entrada no bocal;

T_2 – temperatura do gás na saída do bocal;

P_0 – pressão do gás na câmara de combustão, admitida como a pressão de entrada no bocal;

P_2 – pressão do gás na saída do bocal;

a_0 – velocidade do som para o gás na câmara de combustão;

a_2 – velocidade do som para o gás na saída do bocal;

ρ_0 - densidade do gás na câmara de combustão;

ρ_2 - densidade do gás na saída do bocal;

V_2 – velocidade do gás na saída do bocal;

M_2 – número de mach do gás na saída do bocal;

w – vazão em massa;

D_1 – diâmetro da garganta do bocal (diâmetro mínimo do bocal);

D_2 – diâmetro do bocal na seção de saída;

A_1 – área da seção de garganta do bocal;

A_2 – área da seção de saída do bocal;

F_I – força de empuxo no caso unidimensional;

F_{II} – força de empuxo aproximada para o caso bidimensional;

acel – aceleração do foguete;

P_0, T_0, a_0 e ρ_0 são chamados de propriedades de estagnação onde V=0 define o estado de estagnação.

Na garganta temos M=1 e o estado crítico, com P^*, T^*, a^* e ρ^* (P_1, T_1, a_1 e ρ_1).

$$\frac{p}{p_0} = \left(\frac{\rho}{\rho_0}\right)^{\gamma} \quad (3.1)$$

$$\frac{T_o}{T} = 1 + \frac{\gamma - 1}{2} M^2 \quad (3.2)$$

$$\frac{p_0}{p} = \left(1 + \frac{\gamma - 1}{\gamma} M^2\right)^{\frac{\gamma}{\gamma - 1}} \quad (3.3)$$

$$\frac{T}{T_0} = \left(\frac{p}{p_0}\right)^{\frac{\gamma}{\gamma-1}} \quad (3.4)$$

$$\frac{\rho_0}{\rho} = \left(1 + \frac{\gamma - 1}{\gamma} M^2\right)^{\frac{\gamma}{\gamma-1}} \quad (3.5)$$

$$a = \sqrt{\gamma . R.T} \quad (3.6)$$

$$w = A^* . \frac{p_0}{\sqrt{T_0}} \sqrt{\frac{\gamma}{R}\left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}} \quad (3.7) \text{ vazão em massa}$$

$$w_2 = w^* = w = \rho_2 V_2 A_2 \quad (3.8)$$

$$A_2 = \frac{A_1}{M_2} \sqrt{\left[\frac{\phi_2}{1 + \left(\frac{\gamma-1}{2}\right)}\right]^{\frac{\gamma+1}{\gamma-1}}} \quad (3.9)$$

$$F = (P_2 - P_3)A + wV_2 \quad (3.10) \text{ empuxo do bocal}$$

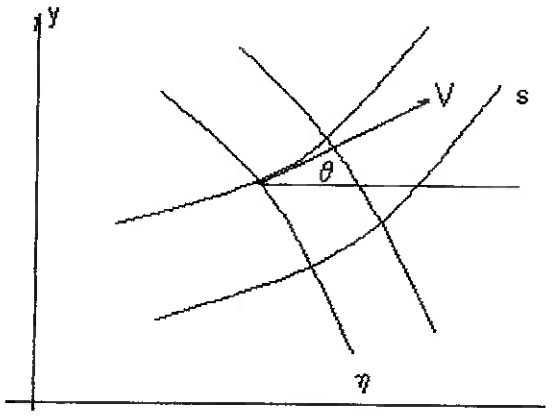
$$acel = \frac{F}{m} - g \quad (3.11) \text{ aceleração do foguete durante o vôo propelido;}$$

Não levamos em conta o decréscimo dm na massa total do foguete (em virtude da saída de gás) por se tratar apenas de um cálculo estimativo, no que tange a altura alcançada pelo foguete.

O memorial de cálculo feito com o programa Mathcad se encontra no capítulo VII. As relações usadas são derivadas diretamente das equações (3.1) a (3.11) e da cinemática. Nesse cálculo tambem se faz uma simples comparação entre o empuxo de um bocal unidimensional e o de um bocal bidimensional, levando-se em conta um diâmetro de seção de saída extraido do cálculo pelo método das características (programa Bocal).

IV)Parte B: Projeto Bidimensional

Consideremos um campo de escoamento bidimensional:



(Figura 1)

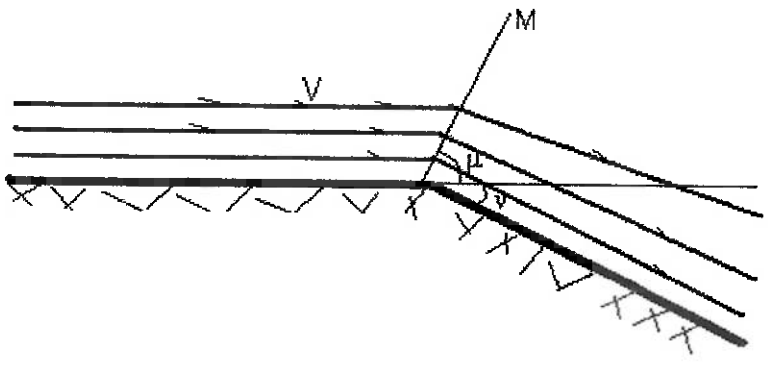
Adotando as hipóteses de escoamento uniforme e isoentrópico e aplicando as leis de conservação de massa, momento e energia, chegamos ao seguinte sistema de equações:

$$\left(M^2-1\right)\frac{1}{V}\frac{\partial V}{\partial s}+\frac{\partial \theta}{\partial \eta}=\frac{\operatorname{sen}\theta}{r} \quad (4.1)$$

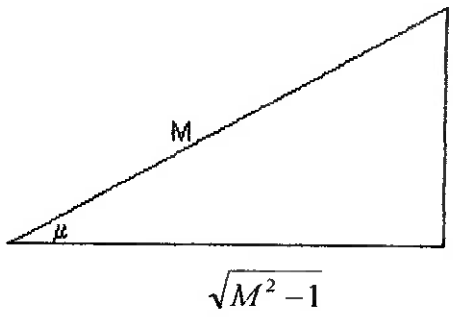
$$\frac{\partial V}{\partial \eta}+V\frac{\partial \theta}{\partial s}=0 \quad (4.2)$$

Onde r representa o raio no sistema de coordenadas cilíndricas.

Observando os diagramas:



(Figura 2)



(Figura 3)

onde,

μ = ângulo de Mach e $\text{sen}(\mu) = 1/M$ (definição);

ν = função de Prandtl-Meyer (ou ângulo de Prandtl-Meyer) e depende do gás e do número de mach,

$$\nu(M) = \sqrt{\frac{\gamma+1}{\gamma-1}}.arctg\left(\sqrt{\frac{\gamma-1}{\gamma+1}(M^2-1)}\right) - arctg\left(\sqrt{(M^2-1)}\right) \quad (4.3)$$

estabelecemos as relações

$$tg(\mu) = \frac{1}{\sqrt{M^2-1}} \quad (4.4) \qquad \text{e} \qquad \frac{dV}{V} = \frac{d\nu}{\sqrt{M^2-1}} \quad (4.5)$$

Aplicando (4.4) em (4.1) e (4.2) obtemos:

$$tg(\mu)\frac{\sqrt{M^2-1}}{V}\frac{\partial V}{\partial \eta} + \frac{\partial \theta}{\partial s} = 0 \quad (4.6)$$

$$\frac{\sqrt{M^2-1}}{V} + \frac{\partial V}{\partial s} + tg(\mu)\frac{\partial \theta}{\partial \eta} = \frac{tg(\mu)\,\text{sen}(\theta)}{r} \quad (4.7)$$

Finalmente, aplicando (4.5) em (4.6) e (4.7), simplificamos o sistema:

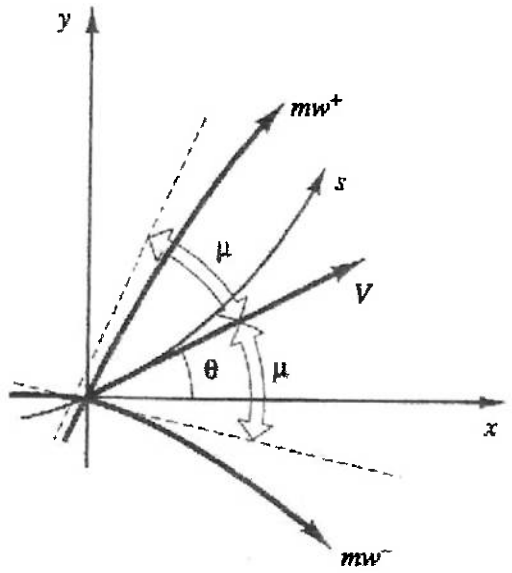
$$tg(\mu)\frac{\partial \nu}{\partial \eta}+\frac{\partial \theta}{\partial s}=0 \quad (4.8a)$$

$$\frac{\partial \nu}{\partial s}+tg(\mu)\frac{\partial \theta}{\partial \eta}=\frac{\text{sen}(\theta)tg(\mu)}{r} \quad (4.8b)$$

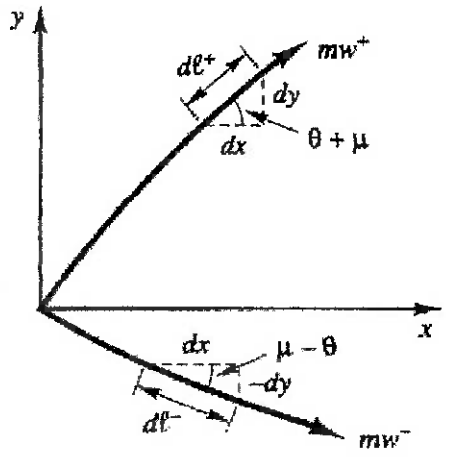
Manipulando-se esse sistema e fazendo uma mudança para coordenadas cartesianas, chegaremos a:

$$\begin{cases} \cos(\theta-\mu)\left(\frac{\partial}{\partial x}+tg(\theta-\mu)\frac{\partial}{\partial y}\right)(\theta+\nu)=\frac{\text{sen}(\mu)\text{sen}(\theta)}{r} & (4.9a) \\ \cos(\theta+\mu)\left(\frac{\partial}{\partial x}+tg(\theta+\mu)\frac{\partial}{\partial y}\right)(\theta-\nu)=-\frac{\text{sen}(\mu)\text{sen}(\theta)}{r} & (4.9b) \end{cases}$$

Agora, observemos a figura abaixo:



The Relationship among the Plus and Minus Mach Waves and the Streamline through a Point in Space



Differential Geometry along Mach Waves

Podemos deduzir a seguinte expressão:

(4.10) $\frac{dy}{dx} = tg(\theta \pm \mu)$ ao longo de $mw^{\pm}$

E através de mais manipulações algébricas chegamos à forma final:

$$\begin{cases} d(\theta - \nu) = -\dfrac{\text{sen}(\theta)\,\text{sen}(\mu)}{\text{sen}(\theta + \mu)}\dfrac{dr}{r} & \text{(4.11a)} \\ d(\theta + \nu) = +\dfrac{\text{sen}(\theta)\,\text{sen}(\mu)}{\text{sen}(\theta - \mu)}\dfrac{dr}{r} & \text{(4.11b)} \end{cases}$$

Aplicado a escoamentos bidimensionais (planar ou axisimétrico), permanente, supersônico, isoentálpico e isoentrópico.

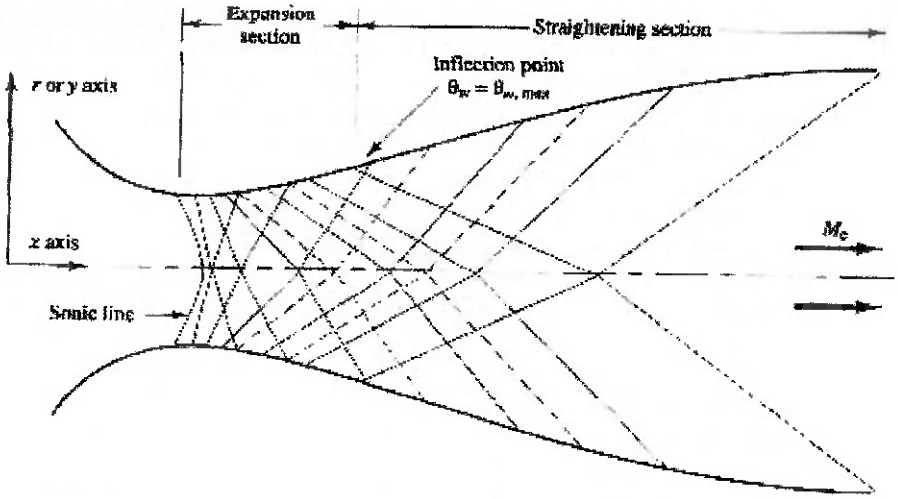
ESCOAMENTO PLANO

Se assumirmos um escoamento plano, $r \to \infty$, teremos:

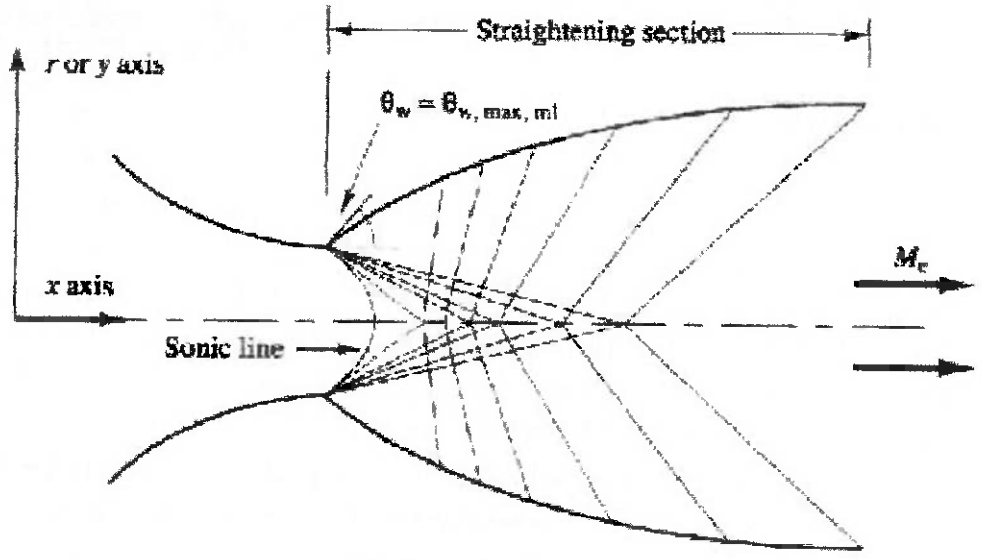
$$\begin{cases} \theta - \nu = cte = K^{+} & \text{(4.12a) ao longo das curvas} \quad \dfrac{dy}{dx} = tg(\theta + \mu) \\ \theta + \nu = cte = K^{-} & \text{(4.12b) ao longo das curvas} \quad \dfrac{dy}{dx} = tg(\theta - \mu) \end{cases}$$

V)Método das Características

Quando projetamos bocais de foguetes, o tamanho passa a ser um fator muito importante em virtude da massa. Aqui estão duas soluções de design para o método das características:



Bocal convergente-divergente com zona de expansão



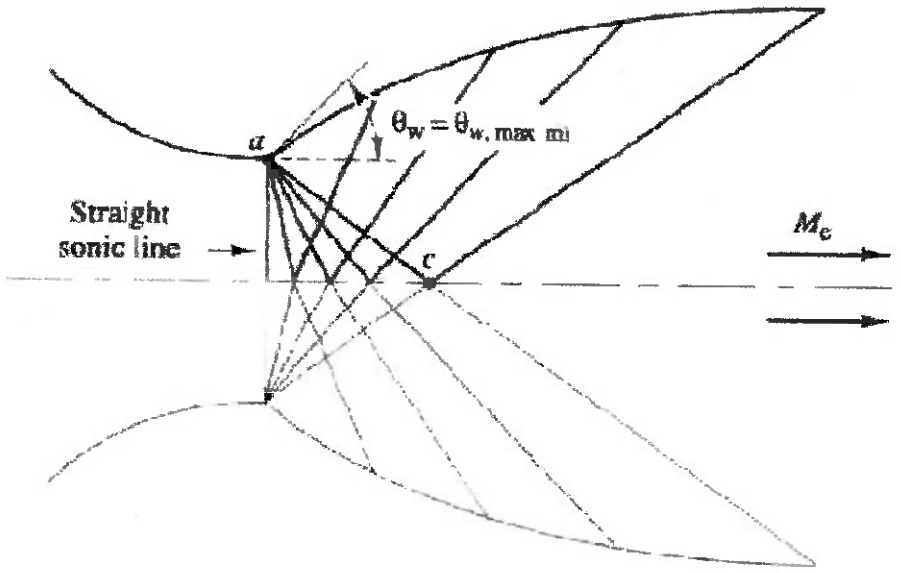
Bocal sem zona de expansão (Minimum-Lenght-Nozzle)

A segunda solução é clássica na literatura, principalmente quando usamos a solução pelo método das características. Tambem representa uma economia significativa de material.

Para o mesmo mach de saída, $\theta_{w,\max,ML} > \theta_{w,\max}$.

Condições:

No ponto c : $\begin{cases} \theta = 0 \\ M = M_{saida} \\ \nu = \nu(M_{saida}) \end{cases}$



Determinação do ângulo inicial para um bocal de comprimento mínimo

A última curva característica C^- de expansão na parte superior (da simetria) é C_{ac}^- , onde $\theta = \theta_{w,\max,ML}$ e $M = M_a$.

Como $M_a = 1 \Rightarrow \nu_a = \theta_{w,\max,ML}$

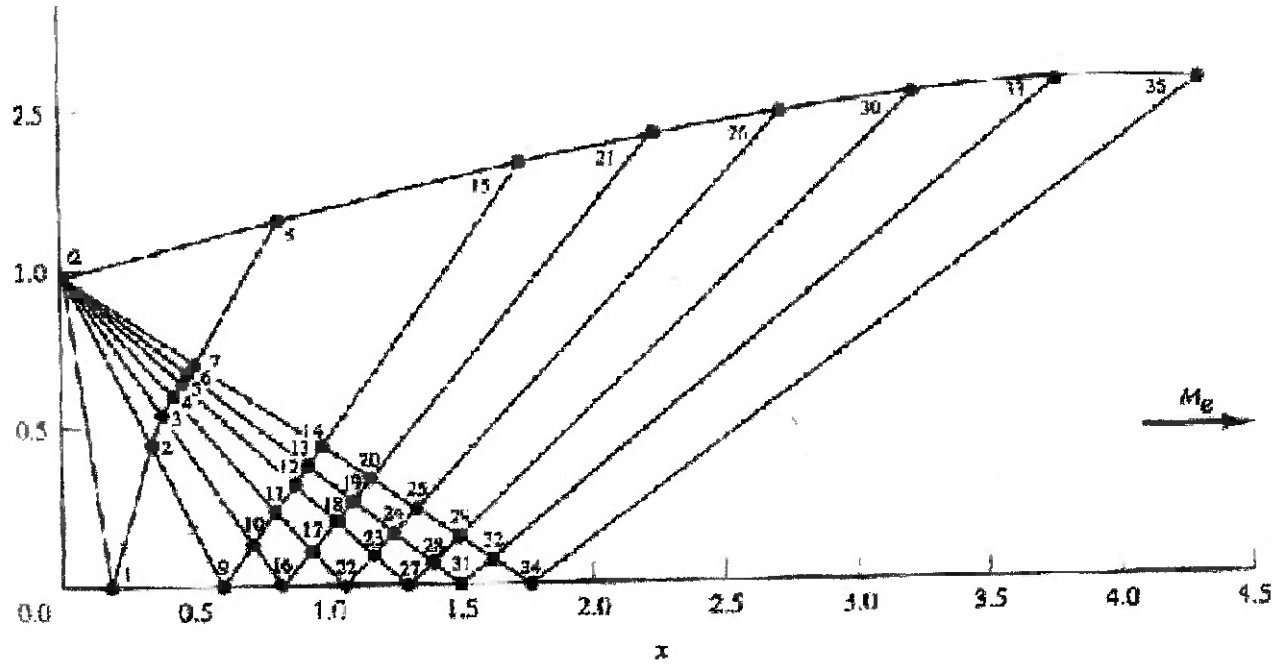
Então $K_a^- = \theta_a + \nu_a = 2.\theta_{w,\max,ML}$ (5.1)

e $K_c^- = \theta_c + \nu_c = 0 + \nu(M_{saida}) = \nu(M_{saida})$ (5.2)

Como $K_a^- = K_c^-$ (cte ao longo de C_{ac}^-)

Então $\theta_{w,\max,ML} = \frac{1}{2}\nu(M_{saida})$ (5.3)

Assim sendo, a partir do mach de saída, temos definido o ângulo θ necessário aos cálculos do perfil.



Exemplo de grid pelo método das características para mach de saída 1.92 e usando sete divisões iniciais

O passo seguinte adotado é a divisão desse ângulo máximo em partes iguais. O número de divisões definem a precisão dos resultados. As linhas características mw^+ e mw^- formam um grid de células com 3 ou 4 vértices. O algoritmo consiste na aplicação reiterada das equações (4.4), (4.5), (4.10) e (4.12) aos pontos das células.

VI)Programa "Bocal"

O núcleo do programa é o cálculo de cada ponto do grid, nas variáveis θ, μ, ν, K^+, K^-, x, y e Mach. Consideramos o perfil axisimétrico, o que facilita o cálculo nas células que fazem divisa com a linha de simetria.

Os dados iniciais capturados pela interface gráfica com o usuário são o Mach de saída do bocal, a constante específica γ do gás, o diâmetro da garganta em milímetros e o número de pontos do perfil (limitamos em 149 por questão de memória) que é igual ao números de divisões do máximo ângulo theta do perfil.

O algoritmo principal parte de um bocal de garganta unitária, sem unidade. O perfil achado serve para qualquer escala. O programa apenas multiplica o perfil pelo valor do raio de garganta a partir do diâmetro inicial fornecido. Não são necessários quaisquer outros dados para a obtenção do perfil. Entretanto como desejamos comparar o perfil bidimensional com o perfil unidimensional em termos de pressão de saída e empuxo final do bocal, vamos precisar de outros parâmetros de entrada e saída. O usuário deve então fornecer a temperatura(Kelvin) e a pressão(Pascal) de entrada, bem como a constante R do gás (em J/Kg.K).

Com os dados iniciais calculamos isoentropicamente os valores de pressão, temperatura, densidade e velocidade do som para o gás na região da garganta. Calculamos também a vazão em massa que, pela continuidade, é constante ao longo do bocal.

A rotina CIPAKC calcula os pontos iniciais junto à garganta. IRANP monta as células para todo o perfil, alocando espaço de memória para os 3 ou 4 pontos de cada célula e suas variáveis.

Em seguida a rotina SECANT calcula Mach e o ângulo de mach(μ) para todos os pontos do grid. CCXY calcula as coordenadas x,y de todos os

pontos do grid. A saída gráfica separa os pontos do grid que correspondem ao perfil do bocal.

Duas outras rotinas complementam o núcleo do programa: CX2 que percorre os pontos do grid através das linhas características, calculando para cada uma um valor médio de ρ (densidade) do gás. Utilizamos a fórmula

$$\omega = \rho_n \sum_i^j V \frac{\text{sen}(\mu - \theta)}{\text{sen}(\mu)} \pi (y_{i+1}^2 - y_i^2) \quad , \quad (6.1)$$

onde a velocidade V do gás é obtida através das relações isoentrópicas já apresentadas:

$$V = M\sqrt{\gamma RT} \quad (6.2) \qquad \text{e} \qquad T = \frac{T^*\left(\frac{\gamma+1}{2}\right)}{\left(1+\frac{\gamma-1}{2}M^2\right)} \quad (6.3)$$

A equação (6.1) apenas reflete a lei da continuidade aplicada ao escoamento.

E a rotina CalForça que calcula o empuxo em cada seção do bocal, pela relação

$$F_n = \sum_i^j \pi (y_{i+1}^2 - y_i^2) \left[(P_i - P_0) + \rho_n V_i \frac{\text{sen}(\mu - \theta)\cos(\theta)}{\text{sen}(\mu)} \right] \quad (6.4)$$

onde $P_i = \rho_n R T_i$

A equação (6.4) é um arranjo da relação unidimensional para o cálculo do empuxo já mostrada no capítulo III).

VII)Memorial de cálculo com o programa MATHCAD

CALCULO UNIDIMENSIONAL:

$\gamma := 1.2$ Constante específica do gás queimado

$T_0 := 2500 \cdot K$ Temperatura na entrada do bocal (K)

$P_0 := (150 \cdot 10^5) \cdot Pa$ Pressão na entrada do bocal (Pa)

$D_1 := 0.008 \cdot m$ Diâmetro da Garganta-D* (m)

$M_2 := 3$ Mach de saida do bocal

$Rbarra := 8314 \cdot \frac{joule}{kg \cdot K}$ Constante universal dos gases perfeitos

$Mol := 40$ Massa molecular dos gases de combustão

$R := \frac{Rbarra}{Mol}$ $R = 207.85 \cdot \frac{joule}{kg \cdot K}$

$a_0 := \sqrt{\gamma \cdot R \cdot T_0}$ $a_0 = 789.65 \cdot \frac{m}{sec}$ Velocidade do som(seção de entrada)

$\rho_0 := \frac{P_0}{R \cdot T_0}$ $\rho_0 = 28.87 \cdot \frac{kg}{m^3}$ Densidade do gás na entrada do bocal

$A_1 := \frac{(D_1)^2 \cdot \pi}{4}$ $A_1 = 5.03 \cdot 10^{-5} \cdot m^2$ Área da garganta-(A*)

$\omega_1 := \frac{A_1 \cdot P_0 \cdot \gamma}{a_0} \cdot \sqrt{\left(\frac{2}{\gamma + 1}\right)^{\frac{\gamma + 1}{\gamma - 1}}}$ $\omega_1 = 0.68 \cdot \frac{kg}{sec}$ **Vazão na garganta**

$\phi_2 := 1 + \left(\frac{\gamma - 1}{2}\right) \cdot (M_2)^2$ $\phi_2 = 1.9$

$T_2 := \frac{T_0}{\phi_2}$ $T_2 = 1.32 \cdot 10^3 \cdot K$ Temperatura na saida

$a_2 := \sqrt{\gamma \cdot R \cdot T_2}$ $a_2 = 572.87 \cdot \frac{m}{sec}$ Velocidade do som na saída

$V_2 := M_2 \cdot a_2$ $V_2 = 1.72 \cdot 10^3 \cdot \frac{m}{sec}$ Velocidade dos gases na saída

$P_2 := \frac{P_0}{(\phi_2)^{\left(\frac{\gamma}{\gamma - 1}\right)}}$ $P_2 = 3.19 \cdot 10^5 \cdot Pa$ **Pressão na saida**

$$\rho_2 := \frac{\rho_0}{\left(\phi_2\right)^{\frac{1}{\gamma-1}}} \qquad \rho_2 = 1.17 \cdot \frac{kg}{m^3}$$ Densidade na saida

$$A_2 := \frac{A_1}{M_2} \cdot \sqrt{\left[\frac{\phi_2}{1 + \left(\frac{\gamma-1}{2}\right)}\right]^{\frac{\gamma+1}{\gamma-1}}} \qquad A_2 = 3.39 \cdot 10^{-4} \cdot m^2$$ **Área da seção de saída**

$$D_2 := \sqrt{\frac{4 \cdot A_2}{\pi}} \qquad D_2 = 20.76 \cdot mm$$ Diâmetro da seção de saída

$P_3 := 10^5 \cdot Pa$ Pressão Atmosférica

$$F_1 := V_2 \cdot \omega_1 + \left(P_2 - P_3\right) \cdot A_2 \qquad F_1 = 1.24 \cdot 10^3 \cdot newton$$ **Empuxo do motor-foguete**

$$\omega_2 := \rho_2 \cdot V_2 \cdot A_2 \qquad \omega_2 = 0.68 \cdot \frac{kg}{sec}$$

$massa := 20 \cdot kg$ Massa do foguete

$$g := 9.8 \cdot \frac{m}{sec^2} \qquad acel := \frac{F_1}{massa} - g \qquad acel = 52.19 \cdot \frac{m}{sec^2}$$

$$\Delta t1 := \frac{3 \cdot kg}{\omega_1} \qquad \Delta t1 = 4.42 \cdot sec$$ Tempo de escoamento do gás (vôo propelido) para uma massa de combustível de 3Kg

$$\Delta S1 := \frac{acel \cdot \Delta t1^2}{2} \qquad \Delta S1 = 510.44 \cdot m$$

$$Vec1 := acel \cdot \Delta t1 \qquad Vec1 = 230.84 \cdot \frac{m}{sec} \qquad \Delta S2 := \frac{Vec1^2}{2 \cdot g} \qquad \Delta S2 = 2.72 \cdot 10^3 \cdot m$$

$$H := \Delta S1 + \Delta S2 \qquad H = 3.23 \cdot 10^3 \cdot m$$ Alcance Vertical do Foguete

NA GARGANTA:

valores críticos de P, T e ρ

$$P_1 := P_0 \cdot \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma}{\gamma-1}} \qquad P_1 = 8.47 \cdot 10^6 \cdot Pa$$

$$\rho_1 := \rho_0 \cdot \left(\frac{2}{\gamma+1}\right)^{\frac{1}{\gamma-1}} \qquad \rho_1 = 17.92 \cdot \frac{kg}{m^3} \qquad T_1 := T_0 \cdot \left(\frac{2}{\gamma+1}\right) \qquad T_1 = 2.27 \cdot 10^3 \cdot K$$

Cálculo da massa de combustível

$DD := 80 \cdot 10^{-3} \cdot m$ Diâmetro interno do foguete e da entrada do bocal

$dd := 15 \cdot 10^{-3} \cdot m$ Diâmetro interno do grão de combustível

$\rho c := 1300 \cdot \frac{kg}{m^3}$ densidade do combustível(propelente)

$L := 0.5 \cdot m$ Altura do grão propelente

$$Wcomb := \rho c \cdot L \cdot \pi \cdot \left(\frac{DD^2 - dd^2}{4} \right)$$

$Wcomb = 3.15 \cdot kg$ Massa total de combustível
Foi usado o valor de 3Kg para estimativa da altura alcançada

CALCULO BI-DIMENSIONAL:

$y := 6.74$ valor extraido de cálculo pelo método das características

$$Raio := \frac{D_1}{2} \cdot y \qquad Raio = 26.96 \cdot mm$$

$$Area := \pi \cdot Raio^2 \qquad Area = 22.83 \cdot cm^2 \qquad \rho_2 := \frac{\omega_2}{Area \cdot V_2} \qquad \rho_2 = 0.17 \cdot \frac{kg}{m^3}$$

$$P_2 := \rho_2 \cdot R \cdot T_2 \qquad P_2 = 4.73 \cdot 10^4 \cdot Pa$$

$F_2 := V_2 \cdot \omega_2 + (P_2 - P_3) \cdot Area$ $F_2 = 1.05 \cdot 10^3 \cdot newton$ Empuxo do bocal bidimensional (aproximado)

$\eta := \frac{F_2}{F_1}$ $\eta = 0.84$ razão de eficiência

VIII)Grid de Resultados (programa Bocal)

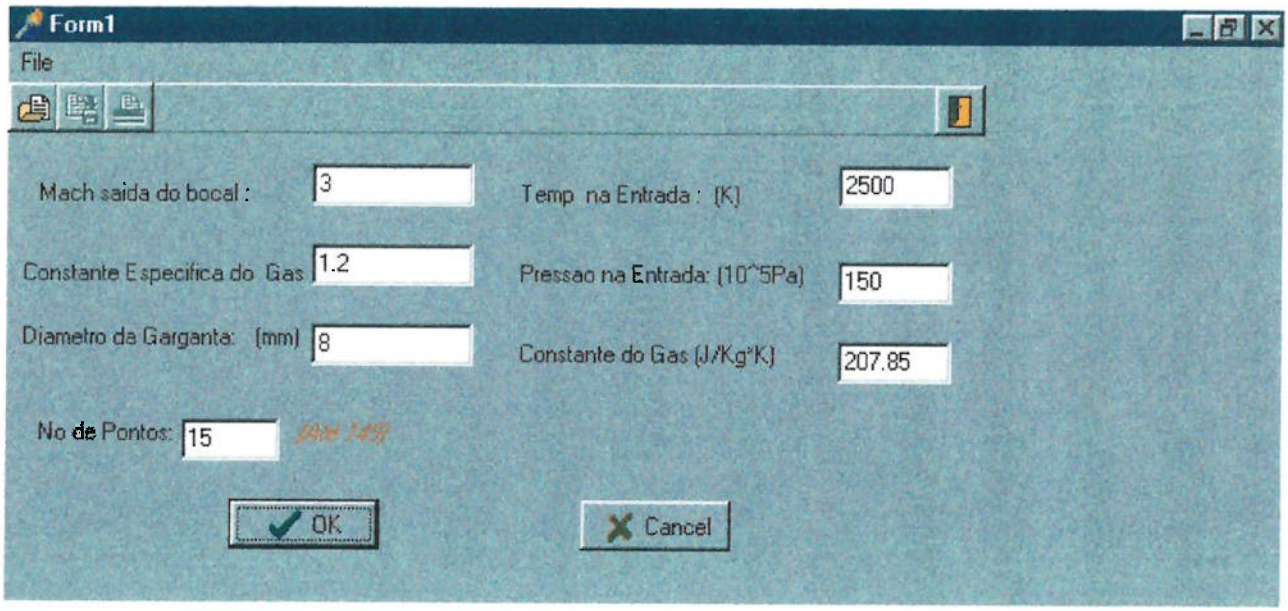
Mach de saida: **3,0**
Constante específica do gás (Cp/Cv) : **1,2**
Número de pontos de divisão iniciais(número de pontos do perfil): **15**

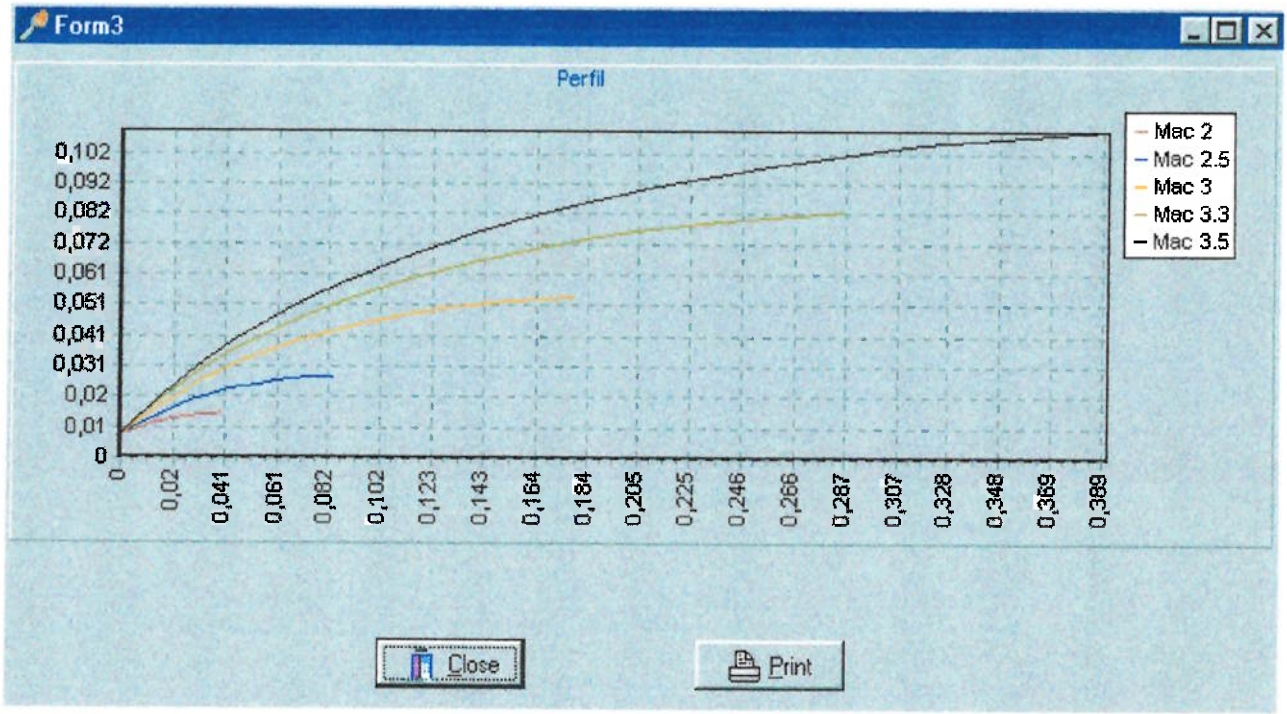
no	TH(θ)	NU(ν)	Mach	MU(μ)	XX (m)	YY (m)
1	2.122	2.122	1.129	62.348	0.000	1.000
2	4.244	4.244	1.210	55.708	0.000	1.000
3	6.365	6.365	1.282	51.254	0.000	1.000
4	8.487	8.487	1.349	47.838	0.000	1.000
5	10.609	10.609	1.413	45.047	0.000	1.000
6	12.731	12.731	1.475	42.681	0.000	1.000
7	14.853	14.853	1.536	40.624	0.000	1.000
8	16.974	16.974	1.596	38.802	0.000	1.000
9	19.096	19.096	1.655	37.168	0.000	1.000
10	21.218	21.218	1.714	35.685	0.000	1.000
11	23.340	23.340	1.773	34.328	0.000	1.000
12	25.462	25.462	1.832	33.078	0.000	1.000
13	27.583	27.583	1.891	31.919	0.000	1.000
14	29.705	29.705	1.951	30.839	0.000	1.000
15	**31.827**	**31.827**	**2.010**	**29.828**	**0.000**	**1.000**
16	2.122	2.122	1.129	62.348	0.572	0.000
17	4.244	4.244	1.210	55.708	0.661	0.170
18	6.365	6.365	1.282	51.254	0.726	0.277
19	8.487	8.487	1.349	47.838	0.781	0.360
20	10.609	10.609	1.413	45.047	0.829	0.432
21	12.731	12.731	1.475	42.681	0.874	0.497
22	14.853	14.853	1.536	40.624	0.916	0.558
23	16.974	16.974	1.596	38.802	0.956	0.617
24	19.096	19.096	1.655	37.168	0.995	0.675
25	21.218	21.218	1.714	35.685	1.034	0.733
26	23.340	23.340	1.773	34.328	1.071	0.792
27	25.462	25.462	1.832	33.078	1.109	0.852
28	27.583	27.583	1.891	31.919	1.145	0.913
29	29.705	29.705	1.951	30.839	1.182	0.977
30	31.827	31.827	2.010	29.828	1.218	1.043
31	**31.827**	**31.827**	**2.010**	**29.828**	**1.797**	**2.116**
32	0.000	8.487	1.349	47.838	0.805	0.000
33	2.122	10.609	1.413	45.047	0.903	0.106
34	4.244	12.731	1.475	42.681	0.985	0.195
35	6.365	14.853	1.536	40.624	1.059	0.274
36	8.487	16.974	1.596	38.802	1.128	0.349
37	10.609	19.096	1.655	37.168	1.195	0.421
38	12.731	21.218	1.714	35.685	1.259	0.492
39	14.853	23.340	1.773	34.328	1.322	0.564
40	16.974	25.462	1.832	33.078	1.384	0.638
41	19.096	27.583	1.891	31.919	1.446	0.713
42	21.218	29.705	1.951	30.839	1.508	0.791
43	23.340	31.827	2.010	29.828	1.571	0.873
44	25.462	33.949	2.071	28.878	1.634	0.959
45	27.583	36.071	2.131	27.982	1.697	1.049

46	**27.583**	**36.071**	**2.131**	**27.982**	**2.833**	**2.707**
47	0.000	12.731	1.475	42.681	1.018	0.000
48	2.122	14.853	1.536	40.624	1.116	0.091
49	4.244	16.974	1.596	38.802	1.206	0.174
50	6.365	19.096	1.655	37.168	1.290	0.253
51	8.487	21.218	1.714	35.685	1.371	0.331
52	10.609	23.340	1.773	34.328	1.451	0.410
53	12.731	25.462	1.832	33.078	1.529	0.489
54	14.853	27.583	1.891	31.919	1.607	0.571
55	16.974	29.705	1.951	30.839	1.686	0.656
56	19.096	31.827	2.010	29.828	1.765	0.745
57	21.218	33.949	2.071	28.878	1.845	0.839
58	23.340	36.071	2.131	27.982	1.926	0.938
59	25.462	38.192	2.193	27.134	2.009	1.044
60	**25.462**	**38.192**	**2.193**	**27.134**	**3.557**	**3.068**
61	0.000	16.974	1.596	38.802	1.230	0.000
62	2.122	19.096	1.655	37.168	1.334	0.085
63	4.244	21.218	1.714	35.685	1.433	0.167
64	6.365	23.340	1.773	34.328	1.530	0.249
65	8.487	25.462	1.832	33.078	1.625	0.332
66	10.609	27.583	1.891	31.919	1.719	0.417
67	12.731	29.705	1.951	30.839	1.814	0.505
68	14.853	31.827	2.010	29.828	1.910	0.598
69	16.974	33.949	2.071	28.878	2.007	0.697
70	19.096	36.071	2.131	27.982	2.106	0.801
71	21.218	38.192	2.193	27.134	2.208	0.912
72	23.340	40.314	2.255	26.330	2.311	1.031
73	**23.340**	**40.314**	**2.255**	**26.330**	**4.343**	**3.424**
74	0.000	21.218	1.714	35.685	1.453	0.000
75	2.122	23.340	1.773	34.328	1.568	0.083
76	4.244	25.462	1.832	33.078	1.680	0.167
77	6.365	27.583	1.891	31.919	1.791	0.254
78	8.487	29.705	1.951	30.839	1.903	0.343
79	10.609	31.827	2.010	29.828	2.015	0.438
80	12.731	33.949	2.071	28.878	2.130	0.537
81	14.853	36.071	2.131	27.982	2.247	0.643
82	16.974	38.192	2.193	27.134	2.367	0.757
83	19.096	40.314	2.255	26.330	2.490	0.879
84	21.218	42.436	2.317	25.565	2.617	1.012
85	**21.218**	**42.436**	**2.317**	**25.565**	**5.223**	**3.785**
86	0.000	25.462	1.832	33.078	1.698	0.000
87	2.122	27.583	1.891	31.919	1.826	0.085
88	4.244	29.705	1.951	30.839	1.955	0.174
89	6.365	31.827	2.010	29.828	2.084	0.267
90	8.487	33.949	2.071	28.878	2.216	0.365
91	10.609	36.071	2.131	27.982	2.351	0.471
92	12.731	38.192	2.193	27.134	2.490	0.584
93	14.853	40.314	2.255	26.330	2.633	0.706
94	16.974	42.436	2.317	25.565	2.781	0.839
95	19.096	44.558	2.381	24.836	2.935	0.984
96	**19.096**	**44.558**	**2.381**	**24.836**	**6.224**	**4.152**
97	0.000	29.705	1.951	30.839	1.972	0.000
98	2.122	31.827	2.010	29.828	2.119	0.090
99	4.244	33.949	2.071	28.878	2.268	0.185

100	6.365	36.071	2.131	27.982	2.421	0.287
101	8.487	38.192	2.193	27.134	2.578	0.397
102	10.609	40.314	2.255	26.330	2.741	0.516
103	12.731	42.436	2.317	25.565	2.910	0.647
104	14.853	44.558	2.381	24.836	3.086	0.789
105	16.974	46.680	2.445	24.140	3.270	0.946
106	**16.974**	**46.680**	**2.445**	**24.140**	**7.372**	**4.526**
107	0.000	33.949	2.071	28.878	2.285	0.000
108	2.122	36.071	2.131	27.982	2.455	0.096
109	4.244	38.192	2.193	27.134	2.631	0.201
110	6.365	40.314	2.255	26.330	2.813	0.315
111	8.487	42.436	2.317	25.565	3.002	0.439
112	10.609	44.558	2.381	24.836	3.200	0.577
113	12.731	46.680	2.445	24.140	3.408	0.728
114	14.853	48.801	2.511	23.473	3.626	0.896
115	**14.853**	**48.801**	**2.511**	**23.473**	**8.695**	**4.903**
116	0.000	38.192	2.193	27.134	2.648	0.000
117	2.122	40.314	2.255	26.330	2.849	0.106
118	4.244	42.436	2.317	25.565	3.058	0.222
119	6.365	44.558	2.381	24.836	3.277	0.351
120	8.487	46.680	2.445	24.140	3.508	0.495
121	10.609	48.801	2.511	23.473	3.751	0.655
122	12.731	50.923	2.577	22.834	4.007	0.833
123	**12.731**	**50.923**	**2.577**	**22.834**	**10.225**	**5.279**
124	0.000	42.436	2.317	25.565	3.077	0.000
125	2.122	44.558	2.381	24.836	3.316	0.118
126	4.244	46.680	2.445	24.140	3.568	0.250
127	6.365	48.801	2.511	23.473	3.835	0.399
128	8.487	50.923	2.577	22.834	4.118	0.566
129	10.609	53.045	2.644	22.221	4.419	0.755
130	**10.609**	**53.045**	**2.644**	**22.221**	**11.999**	**5.645**
131	0.000	46.680	2.445	24.140	3.588	0.000
132	2.122	48.801	2.511	23.473	3.877	0.134
133	4.244	50.923	2.577	22.834	4.184	0.286
134	6.365	53.045	2.644	22.221	4.512	0.459
135	8.487	55.167	2.713	21.630	4.864	0.657
136	**8.487**	**55.167**	**2.713**	**21.630**	**14.061**	**5.992**
137	0.000	50.923	2.577	22.834	4.207	0.000
138	2.122	53.045	2.644	22.221	4.560	0.154
139	4.244	55.167	2.713	21.630	4.939	0.332
140	6.365	57.289	2.783	21.062	5.348	0.537
141	**6.365**	**57.289**	**2.783**	**21.062**	**16.463**	**6.305**
142	0.000	55.167	2.713	21.630	4.964	0.000
143	2.122	57.289	2.783	21.062	5.402	0.180
144	4.244	59.410	2.854	20.513	5.875	0.391
145	**4.244**	**59.410**	**2.854**	**20.513**	**19.264**	**6.565**
146	0.000	59.410	2.854	20.513	5.904	0.000
147	2.122	61.532	2.926	19.983	6.452	0.214
148	**2.122**	**61.532**	**2.926**	**19.983**	**22.538**	**6.747**
149	0.000	63.654	3.000	19.471	7.085	0.000
150	**0.000**	**63.654**	**3.000**	**19.471**	**26.370**	**6.818**

IX)Apêndice B: Listagem do programa Bocal





```
 1: program DVP;          {** PROGRAMA BOCAL **}
 2:
 3: uses
 4:   Forms,
 5:   U1U in 'U1U.pas' {Form1},
 6:   tresult in 'tresult.pas' {Form2},
 7:   UG in 'UG.pas' {Form3},
 8:   U4 in 'U4.pas' {Form4};
 9:
10: {$R *.RES}
11:
12: begin
13:   Application.Initialize;
14:   Application.CreateForm(TForm1, Form1);
15:   Application.CreateForm(TForm2, Form2);
16:   Application.CreateForm(TForm3, Form3);
17:   Application.CreateForm(TForm4, Form4);
18:   Application.Run;
19: end.
```

```
 1: unit U1U;
 2:
 3: interface
 4:
 5: uses
 6:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 7:   StdCtrls, Buttons, Spin, Menus, ExtCtrls, Grids,math, Db, DBTables;
 8:
 9: type
10:   TForm1 = class(TForm)
11:     Edit1: TEdit;
12:     Label1: TLabel;
13:     Label2: TLabel;
14:     Edit2: TEdit;
15:     Label3: TLabel;
16:     BitBtn1: TBitBtn;
17:     BitBtn2: TBitBtn;
18:     Edit3: TEdit;
19:     Label4: TLabel;
20:     Label5: TLabel;
21:     MainMenu1: TMainMenu;
22:     File1: TMenuItem;
23:     Save1: TMenuItem;
24:     Load1: TMenuItem;
25:     N1: TMenuItem;
26:     Exit1: TMenuItem;
27:     SaveDialog1: TSaveDialog;
28:     OpenDialog1: TOpenDialog;
29:     Panel1: TPanel;
30:     SpeedButton1: TSpeedButton;
31:     SpeedButton2: TSpeedButton;
32:     SpeedButton3: TSpeedButton;
33:     SpeedButton4: TSpeedButton;
34:     Print1: TMenuItem;
35:     Edit4: TEdit;
36:     Label6: TLabel;
37:     Label7: TLabel;
38:     Edit5: TEdit;
39:     Label8: TLabel;
40:     Edit6: TEdit;
41:     Label9: TLabel;
42:     Edit7: TEdit;
43:     procedure BitBtn2Click(Sender: TObject);
44:     procedure BitBtn1Click(Sender: TObject);
45:     procedure Exit1Click(Sender: TObject);
46:     procedure Save1Click(Sender: TObject);
47:     procedure Load1Click(Sender: TObject);
48:     procedure SpeedButton1Click(Sender: TObject);
49:     procedure SpeedButton4Click(Sender: TObject);
50:     procedure FormCreate(Sender: TObject);
51:     procedure ResetTable(Sender: TObject);
52:
53:   private
54:     { Private declarations }
55:   public
56:      { Procedure CIPAKC(N: Integer;var agpn:gpno); }
57:        { Public declarations }
58:   end;
59: type
60:  XXYY = record
61:   X,Y : single;
62:   end;
63:   Keeper = array[1..150] of XXYY;
64:  GPNO=record
65:        KI,KII,THT,NU,MC,MU,XX,YY : single;
66:        end;
```

```
 67:  FGPO = file of GPNO;
 68:  TCoord = Array[1..11500] of single;
 69:  GridPoint=Record
 70:    cellno,flag,Point1,Point2,Point3,Point4 : Integer;
 71:    Ki1,kii1,th1,nu1,mc1,mu1,xx1,yy1, Ki2,kii2,th2,nu2,mc2,mu2,xx2,yy2,
 72:    Ki3,kii3,th3,nu3,mc3,mu3,xx3,yy3, Ki4,kii4,th4,nu4,mc4,mu4,xx4,yy4 :single;
 73:    end;
 74:
 75:  FGRiadpoint = file of Gridpoint;
 76:  BigVetor = array[1..11500] of single;
 77:  SmallVetor = array[1..150] of integer;
 78: var
 79:   Form1: TForm1;
 80:  Procedure CIPAKC(N: Integer;Thetamax,Defang:single;var aGPN:gpno;NU:BigVetor);
 81:  Procedure IRANP(ROW,icl:integer;var aGPN:GPNO; aGR:gridpoint);
 82:  Procedure CALCULA3c(k,count3c,FLA3c:integer;pnt1,pnt2,pnt3:SmallVetor;
 83:                      GPN3c:GPNO; var GR3c:gridpoint);
 84:  Procedure CALCULA4c(k,count4c,FLA4c:integer;pnt1,pnt2,pnt3,pnt4:SmallVetor;GPN4c:G
 85:                      var GR4c:gridpoint);
 86:  Procedure Secant(bb:single;mm:integer;var aGPN:gpno;NU :BigVetor);
 87:  Procedure C1050(L,H,sNU,B:single;it:integer; var aGPN:gpno);
 88:  Procedure MyStore(aGPN:gpno;aGR:gridpoint;co:integer);
 89:  Procedure CCXY(Acell,icl :integer;Thetamax,Defang:single;
 90:                 var aGPN:GPNO; aGR:gridpoint);
 91:  Procedure FinalStore(aGPN:gpno;aGR:gridpoint;rr:integer);
 92:  Function LinhaDoPerfil(atable:Ttable;aw:single):single;
 93:  Function CX2(aNoPoint,adesp:integer):TCoord;
 94:  Function CalForza(aNoPoint:integer;aro:Tcoord):TCoord;
 95:
 96: implementation
 97:
 98: uses tresult, UG;
 99:
100:  const
101:  EPS = 0.00001;
102:   _105=100000;
103:  var
104:  Fgp : FGPO;
105:  FGr : FGRiadpoint;
106:  T0,P0,R,T,P,ro,a0,w,aa,aaa : single;
107:  ed1,ed2,fm : single;
108:  Icl:word;
109:  Values1,Values2,Tc,Vc,Forza,Pp:Tcoord;
110:  agpno : array[1..11500] of gpno;
111: {$R *.DFM}
112:
113: Function CX2(aNoPoint,adesp:integer):TCoord;
114: var
115: cont,contaux,ainicio,afinal,i,n: integer ;
116: Aval:single;
117: afile:FGPO;
118: myvar: GPNO;
119:
120: thr,mur,auf,auf1,vant,aumach,auy: single;
121: Begin
122:
123: cont:=1;
124: contaux:=1;
125: assign(afile,'c:gp.dat');
126: reset(afile);
127:    Aval:=0;
128:    vant:=0;
129:    contaux:=1;
130:    n:=1;
131: {adesp:=adesp div 2;  }
132:
```

```
133: seek(afile,adesp);
134:
135: ANopoint:=Anopoint+1;
136: ainicio :=ANopoint;
137: afinal:=ainicio+ANoPoint-1;
138:
139:
140: repeat
141:   begin
142:   seek(afile,ainicio);
143:   Read(afile,myvar);
144:   aumach:=myvar.MC;
145:   auy:=myvar.YY;
146:   for i:=(ainicio+1) to afinal do
147:       begin
148:
149:       seek(afile,i);
150:       Read(afile,myvar);
151:       thr:=myvar.THT*pi/180;
152:       mur:=myvar.Mu*pi/180;
153:       auf:=(1+(((ed2-1)/2))*sqr(aumach));
154:       auf1:=(t*((ed2+1)/2));
155:       Tc[i]:=auf1/auf;
156:
157:       Vc[i]:=aumach*sqrt(ed2*R*Tc[i]);
158:       Values1[i]:=Vc[i]*(pi/sin(mur+thr))*(sqr(myvar.YY*fm)-sqr(auy*fm));
159:       auy:=myvar.YY;
160:       aumach:=myvar.MC;
161:       AVal:=Values1[i]+Vant;
162:       Vant:=Values1[i];
163:       end;
164:
165: {if contaux >(adesp div 2) then
166:                                 begin }
167:                                 CX2[n]:=(w/AVal);
168:
169:                                 inc(n);
170:                          {       end;    }
171:
172:
173: ainicio:=afinal+1;
174: dec(adesp);
175: afinal:=afinal+(aNopoint-cont);
176: inc(cont);
177: inc(contaux);
178: Aval:=0;
179:         end
180:         until afinal=ainicio;
181:
182:
183:  End;
184:
185: Function CalForza(aNoPoint:integer;aro:tcoord):TCoord;
186: var
187: cont,contaux,ainicio,afinal,i,n,tm: integer ;
188: Aval,Sp,P3:single;
189: afile:FGPO;
190: myvar: GPNO;
191: thr,mur,auy,auf,auf1,vant: single;
192: Begin
193: P3:=_105;
194: cont:=1;
195: contaux:=1;
196: assign(afile,'c:gp.dat');
197: reset(afile);
198:    Aval:=0;
```

```
199:     vant:=0;
200:     sp:=0;
201:     contaux:=1;
202:     auy:=0;
203:     n:=1;
204:
205:
206: seek(afile,Anopoint);
207:
208: ANopoint:=Anopoint+1;
209: ainicio :=ANopoint;
210: afinal:=ainicio+ANoPoint-1;
211:
212: repeat
213:  begin
214:  for i:=(ainicio+1) to afinal do
215:      begin
216:      seek(afile,i);
217:      Read(afile,myvar);
218:      thr:=myvar.THT*pi/180;
219:      mur:=myvar.Mu*pi/180;
220:      auf:=(1+(((ed2-1)/2))*sqr(myvar.MC));
221:      auf1:=(t*((ed2+1)/2));
222:      Tc[i]:=auf1/auf;
223:      Pp[i]:=aro[n]*R*Tc[i];
224:    {  Pp[i]:=p*power(auf,(ed2/(ed2-1))); }
225:      SP:=((Pp[i]-P3)*pi*(sqr(myvar.YY*fm)-sqr(auy*fm)))+Sp;
226:      Vc[i]:=myvar.MC*sqrt(ed2*R*Tc[i]);
227:      Values2[i]:=sqr(Vc[i])*(sqr(cos(thr))/sin(mur+thr))*pi*
228:                                          (sqr(myvar.YY*fm)-sqr(auy*fm));
229:
230:      auy:=myvar.YY;
231:
232:      AVal:=Values2[i]+Vant;
233:      Vant:=Values2[i];
234:      end;
235: { if contaux >(ANoPoint div 2) then
236:     begin   }
237:
238:     CalForza[n]:=Sp+(aro[n]*Aval);
239:     inc(n);
240:   {  end;   }
241:
242:  ainicio:=afinal+1;
243: {dec(adesp);  }
244:  afinal:=afinal+(aNopoint-cont);
245:  inc(cont);
246:  inc(contaux);
247:  Aval:=0;
248:  sp:=0;
249:  end
250:  until afinal=ainicio;
251:
252:
253:  End;
254:
255:
256:  Function LinhaDoPerfil(atable:Ttable;aw:single):single;
257:  var
258:   cont:integer;
259: Begin
260: {cont:=1;
261: atable.first;
262: while not aFGP0.EOF do
263:   begin
264:    {LinhadoPerfil[cont]:=aw/ }
```

```
265: {   atable.next;
266:    inc(cont);
267:   end;
268: }End;
269:
270: Procedure FinalStore(aGPN:gpno;aGR:gridpoint;rr:integer);
271:  var
272:  posi:longint;
273:  Begin
274:   {GET #5, RR, GR}
275:  seek(FGR,rr);
276:  read(FGR,aGR);
277:
278:  aGPN.KI := aGR.KI1;
279:  aGPN.KII := aGR.KII1;
280:  aGPN.THT := aGR.TH1 ;
281:  aGPN.NU := aGR.NU1;
282:  aGPN.MC := aGR.MC1;
283:  aGPN.MU := aGR.MU1;
284:  aGPN.XX := aGR.XX1;
285:  aGPN.YY := aGR.YY1;
286:  posi:=round(aGR.point1);    { PUT #4, GR.POINT1, GPN}
287:  seek(FGP,posi);
288:  write(FGP,aGPN);
289:
290:  aGPN.KI := aGR.KI2;
291:  aGPN.KII := aGR.KII2;
292:  aGPN.THT := aGR.TH2;
293:  aGPN.NU := aGR.NU2;
294:  aGPN.MC := aGR.MC2;
295:  aGPN.MU := aGR.MU2;
296:  aGPN.XX := aGR.XX2;
297:  aGPN.YY := aGR.YY2;
298:  posi:=round(aGR.point2);    {PUT #4, GR.POINT2, GPN  }
299:  seek(FGP,posi);
300:  write(FGP,aGPN);
301:
302:  aGPN.KI := aGR.KI3;
303:  aGPN.KII := aGR.KII3;
304:  aGPN.THT := aGR.TH3;
305:  aGPN.NU := aGR.NU3;
306:  aGPN.MC := aGR.MC3;
307:  aGPN.MU := aGR.MU3;
308:  aGPN.XX := aGR.XX3;
309:  aGPN.YY := aGR.YY3;
310:  posi:=round(aGR.point3);    {PUT #4, GR.POINT3, GPN }
311:  seek(FGP,posi);
312:  write(FGP,aGPN);
313:
314:  IF aGR.POINT4 <> 0 THEN
315:     begin
316:     aGPN.KI := aGR.KI4;
317:     aGPN.KII := aGR.KII4;
318:     aGPN.THT := aGR.TH4;
319:     aGPN.NU := aGR.NU4;
320:     aGPN.MC := aGR.MC4;
321:     aGPN.MU := aGR.MU4;
322:     aGPN.XX := aGR.XX4;
323:     aGPN.YY := aGR.YY4;
324:     posi:=round(aGR.point4);      {PUT #4, GR.POINT4, GPN }
325:     seek(FGP,posi);
326:     write(FGP,aGPN);
327:     end;
328: end;
329:
330: Procedure myStore(aGPN:gpno;aGR:gridpoint;co:integer);
```

```
331: var posi:longint;
332: Begin
333: seek(FGR,co);
334: read(FGR,aGR);
335:
336: posi:=round(aGR.point1);
337: seek(FGP,posi);
338: read(FGP,aGPN);
339: aGR.mc1:=aGPN.mc;
340: aGR.mu1:=aGPN.mu;
341:
342: posi:=round(aGR.point2);
343: seek(FGP,posi);
344: read(FGP,aGPN);
345: aGR.mc2:=aGPN.mc;
346: aGR.mu2:=aGPN.mu;
347:
348: posi:=round(aGR.point3);
349: seek(FGP,posi);
350: read(FGP,aGPN);
351: aGR.mc3:=aGPN.mc;
352: aGR.mu3:=aGPN.mu;
353:
354: if aGR.point4 <> 0 then
355:    begin
356:    posi:=round(aGR.point4);
357:    seek(FGP,posi);
358:    read(FGP,aGPN);
359:    aGR.mc4:=aGPN.mc;
360:    aGR.mu4:=aGPN.mu;
361:    end;
362:
363: seek(FGR,co);
364: write(FGR,aGR);
365:  {exit; }
366: End;
367:
368: Procedure C1050(L,H,sNU,B:single;it:integer; var aGPN:gpno);
369: var teste:boolean;
370:     MC,MU:BigVetor;
371:     A1,A2,C1,C2,FL,F,DX,XL,XH,AH,CH,NUR,NUM,DEN,MUR:single;
372:     ITR:integer;
373: Begin
374: ITR:=0;
375: XL := L;
376: XH := H;
377: NUR := sNU * (pi/ 180);
378:
379: A1 := SQRT(((1 / B)*(sqr(L) - 1)));
380: C1 := SQRT(sqr(L) - 1);
381: A2 := SQRT((1 / B) *(sqr(H) - 1));
382: C2 := SQRT(sqr(H) - 1);
383: FL := (SQRT(B) * Arctan(A1)) - (Arctan(C1)) - NUR;
384: F := (SQRT(B) * Arctan(A2)) - (Arctan(C2)) - NUR;
385: teste:=false;
386: WHILE (ITR <= 2500) and (teste=false)  do
387:       begin
388:       DX := (XL - XH) * (F / (F - FL));
389:       XL := XH;
390:       FL := F;
391:       XH := XH + DX;
392:       AH := SQRT((1 / B) * (sqr(XH) - 1));
393:       CH := SQRT(sqr(XH) - 1);
394:       F := (SQRT(B) * Arctan(AH)) - (Arctan(CH)) - NUR;
395:       IF (ABS(DX) < EPS) OR (F = 0) THEN
396:          begin
```

```
397:          MC[it] := XH;
398:          aGPN.MC := MC[it];
399:          NUM := (1 / MC[it]);
400:          DEN := SQRT(1 - sqr(NUM));
401:          MUR := Arctan(NUM / DEN);
402:          MU[it] := MUR * (180 /pi);
403:          aGPN.MU := MU[it];
404:          seek(FGP,it);     {PUT #4, MM, GPN }
405:          write(FGP,aGPN);
406:          exit;
407:          teste:=true;
408:          end;
409:       ITR := ITR + 1;
410:       end;
411: End;
412:
413: Procedure Secant(bb:single;mm:integer;var aGPN:gpno;NU:BigVetor);
414:
415: begin
416:
417:    {GET #4, MM, GPN}
418:    seek(FGP,mm);
419:    read(FGP,aGPN);
420:    NU[MM] := aGPN.NU;
421:
422:    IF NU[MM]<= 26.38 THEN
423:    C1050(1,2,NU[MM],bb,mm,aGPN)
424:    ELSE IF NU[MM] <= 65.78 THEN
425:         C1050(2,4,NU[MM],bb,mm,aGPN)
426:      ELSE IF NU[MM] <= 84.96 THEN
427:           C1050(4,6,NU[MM],bb,mm,aGPN)
428:        ELSE IF NU[MM] <= 95.62 THEN
429:             C1050(6,8,NU[MM],bb,mm,aGPN)
430:          ELSE IF NU[MM] <= 99.32 THEN
431:               C1050(8,9,NU[MM],bb,mm,aGPN)
432:            ELSE IF NU[MM] <= 102.3 THEN
433:                 C1050(9,10,NU[MM],bb,mm,aGPN)
434:              ELSE IF NU[MM] <= 104.8 THEN
435:                   C1050(10,11,NU[MM],bb,mm,aGPN)
436:                ELSE IF NU[MM] <= 106.9 THEN
437:                     C1050(11,12,NU[MM],bb,mm,aGPN)
438:                  ELSE IF NU[MM] <= 108.7 THEN
439:                       C1050(12,13,NU[MM],bb,mm,aGPN)
440:                    ELSE IF NU[MM] <= 110.2 THEN
441:                         C1050(13,14,NU[MM],bb,mm,aGPN)
442:                      ELSE IF NU[MM] <= 111.5 THEN
443:                           C1050(14,15,NU[MM],bb,mm,aGPN)
444:                        ELSE IF NU[MM] <= 116.2 THEN
445:                             C1050(15,20,NU[MM],bb,mm,aGPN)
446:                          ELSE IF NU[MM] <= 118.6 THEN
447:                               C1050(20,24,NU[MM],bb,mm,aGPN)
448:                            ELSE IF NU[MM] <= 120.9 THEN
449:                                 C1050(24,30,NU[MM],bb,mm,aGPN)
450:                              ELSE IF NU[MM] <= 123.3 THEN
451:                                   C1050(30,40,NU[MM],bb,mm,aGPN)
452:                                   ELSE
453:                                   C1050(40,50,NU[MM],bb,mm,aGPN);
454:
455:
456: end;
457:
458: Procedure CCXY(Acell,icl :integer;Thetamax,Defang:single;
459:                var aGPN:GPNO; aGR:gridpoint);
460: var
461:  posi:longint;
462:  R, pnt1,pnt2,pnt3,pnt4: integer;
```

```
463:  A1Id,A2Id,A3Id,A4Id,A1IId,A2IId,A3IId,A4IId, CON,
464:  A1I,A2I,A3I,A4I,A1II,A2II,A3II,A4II,
465:  S12,S13,S23,S24,S32,S34,X2,Y2,X3,Y3,X4,Y4,
466:  THT1,THT2,THT3,THT4,TH1,TH2,TH3,TH4: single ;
467: Begin
468: FOR R:= 1 TO Acell  do
469:     begin
470:     seek(FGR,R);              {GET #5, R, GR}
471:     read(FGR,aGR);
472:     Pnt1 := aGR.POINT1;
473:     Pnt2 := aGR.POINT2;
474:     Pnt3 := aGR.POINT3;
475:     Pnt4 := aGR.POINT4;
476:
477:     A1ID := (0.5 * (aGR.TH1 - aGR.MU1));
478:     A2ID := (0.5 * (aGR.TH2 - aGR.MU2));
479:     A3ID := (0.5 * (aGR.TH3 - aGR.MU3));
480:     A4ID := (0.5 * (aGR.TH4 - aGR.MU4));
481:     A1IID := (0.5 * (aGR.TH1 + aGR.MU1));
482:     A2IID := (0.5 * (aGR.TH2 + aGR.MU2));
483:     A3IID := (0.5 * (aGR.TH3 + aGR.MU3));
484:     A4IID := (0.5 * (aGR.TH4 + aGR.MU4));
485:
486:     CON := (Pi / 180);
487:     A1I := A1ID * CON;
488:     A2I := A2ID * CON;
489:     A3I := A3ID * CON;
490:     A4I := A4ID * CON;
491:     A1II := A1IID * CON;
492:     A2II := A2IID * CON;
493:     A3II := A3IID * CON;
494:     A4II := A4IID * CON;
495:
496:     if r=1 then
497:        begin
498:        S13 := TAN(2 * A3I);
499:        Y3 := 0;
500:        aGR.YY3 := Y3;
501:        X3 := (-1 / S13);
502:        aGR.XX3 := X3;
503:        aGPN.YY := Y3;
504:        aGPN.XX := X3;
505:        posi:= round(pnt3);   {PUT #4, PT3, GPN }
506:        seek(FGP,posi);
507:        write(FGP,aGPN);
508:        S24 := TAN(2 * A4I);
509:        S34 := TAN(A3II + A4II);
510:        X4 := (1 + (S34 * aGR.XX3)) / (S34 - S24);
511:        Y4 := -(S34 * X3) + (S34 * X4);
512:        aGPN.XX := X4;
513:        aGPN.YY := Y4;
514:        posi:= round(pnt4);   {PUT #4, PT4, GPN}
515:        seek(FGP,posi);
516:        write(FGP,aGPN);
517:        aGR.XX4 := X4;
518:        aGR.YY4 := Y4;
519:        seek(FGR,r);    {PUT #5, R, GR}
520:        write(FGR,aGR);
521:        end;
522:
523:     if (r<icl) and (r<>1) then
524:        begin
525:        S24 := TAN(2 * A4I);
526:        S34 := TAN(A3II + A4II);
527:        posi:= round(pnt3);  {GET #4, PT3, GPN}
528:        seek(FGP,posi);
```

```
529:        read(FGP,aGPN);
530:        X3 := aGPN.XX;
531:        Y3 := aGPN.YY;
532:        aGR.XX3 := X3;
533:        aGR.YY3 := Y3;
534:        X4 := (1- Y3 + (S34 * X3)) / (S34 - S24);
535:        Y4 := 1 + (S24 * X4);
536:        aGPN.XX := X4;
537:        aGPN.YY := Y4;
538:        posi:= round(pnt4);    {PUT #4, PT4, GPN }
539:        seek(FGP,posi);
540:        write(FGP,aGPN);
541:        aGR.XX4 := X4;
542:        aGR.YY4 := Y4;
543:        seek(FGR,r);     {  PUT #5, R, GR}
544:        write(FGR,aGR);
545:        end;
546:
547:     if r=icl then
548:        begin
549:        THT1 := Thetamax * CON;
550:        THT2 := aGR.TH2 * CON;
551:        S12 := TAN(0.5 * (THT1 + THT2));
552:        S32 := TAN(A3II + A2II);
553:        posi:= round(pnt3);  {GET #4, PT3, GPN }
554:        seek(FGP,posi);
555:        read(FGP,aGPN);
556:        X3 := aGPN.XX;
557:        Y3 := aGPN.YY;
558:        aGR.XX3 := X3;
559:        aGR.YY3 := Y3;
560:        X2 := (1 - Y3 + (S32 * X3)) / (S32 - S12);
561:        Y2 := Y3 - (S32 * X3) + (S32 * X2);
562:        aGPN.XX := X2;
563:        aGPN.YY := Y2;
564:        posi:= round(pnt2);    {PUT #4, PT2, GPN }
565:        seek(FGP,posi);
566:        write(FGP,aGPN);
567:        aGR.XX2 := X2;
568:        aGR.YY2 := Y2;
569:        seek(FGR,R);           {PUT #5, R, GR}
570:        write(FGR,aGR);
571:        end;
572:
573:     if r>icl then
574:        begin
575:        IF (aGR.POINT4 <> 0) AND (aGR.KI4 <> 0) THEN
576:           begin
577:           IF aGR.FLAG= 10 THEN
578:              begin
579:              TH2 := aGR.TH2 * CON;
580:              TH4 := aGR.TH4 * CON;
581:              S24 := TAN(0.5 * (TH2 + TH4));
582:              S34 := TAN(A3II + A4II);
583:              posi:=round(pnt1);  {GET #4, PT1, GPN}
584:              seek(FGP,posi);
585:              read(FGP,aGPN);
586:              aGR.XX1 := aGPN.XX;
587:              aGR.YY1 := aGPN.YY;
588:              posi:=round(pnt2);    {GET #4, PT2, GPN}
589:              seek(FGP,posi);
590:              read(FGP,aGPN);
591:              X2 := aGPN.XX;
592:              Y2 := aGPN.YY;
593:              aGR.XX2 := X2;
594:              aGR.YY2 := Y2;
```

```
595:              posi:=round(pnt3);     {GET #4, PT3, GPN }
596:              seek(FGP,posi);
597:              read(FGP,aGPN);
598:              X3 := aGPN.XX;
599:              Y3 := aGPN.YY;
600:              aGR.XX3 := X3;
601:              aGR.YY3 := Y3;
602:              X4 := (Y2 - Y3 + (S34 * X3) - (S24 * X2)) / (S34 - S24);
603:              Y4 := Y2 - (S24 * X2) + (S24 * X4);
604:              aGPN.XX := X4;
605:              aGPN.YY := Y4;
606:              posi:=round(pnt4);      {PUT #4, PT4, GPN}
607:              seek(FGP,posi);
608:              write(FGP,aGPN);
609:              aGR.XX4 := X4;
610:              aGR.YY4 := Y4;
611:              seek(FGR,r);        {PUT #5, R, GR}
612:              write(FGR,aGR);
613:              end
614:           ELSE
615:              begin
616:              S24 := TAN(A2I + A4I);
617:              S34 := TAN(A3II + A4II);
618:              posi:= round(pnt1);  {GET #4, PT1, GPN}
619:              seek(FGP,posi);
620:              read(FGP,aGPN);
621:              aGR.XX1 := aGPN.XX;
622:              aGR.YY1 := aGPN.YY;
623:              posi:= round(pnt2);     {GET #4, PT2, GPN    }
624:              seek(FGP,posi);
625:              read(FGP,aGPN);
626:              X2 := aGPN.XX;
627:              Y2 := aGPN.YY;
628:              aGR.XX2 := X2;
629:              aGR.YY2 := Y2;
630:              posi:= round(pnt3);      {GET #4, PT3, GPN }
631:              seek(FGP,posi);
632:              read(FGP,aGPN);
633:              X3 := aGPN.XX;
634:              Y3 := aGPN.YY;
635:              aGR.XX3 := X3;
636:              aGR.YY3 := Y3;
637:              X4 := (Y2 - Y3 + (S34 * X3) - (S24 * X2)) / (S34 - S24);
638:              Y4 := Y2 - (S24 * X2) + (S24 * X4);
639:              aGPN.XX := X4;
640:              aGPN.YY := Y4;
641:              posi:= round(pnt4);    { PUT #4, PT4, GPN }
642:              seek(FGP,posi);
643:              write(FGP,aGPN);
644:              aGR.XX4 := X4;
645:              aGR.YY4 := Y4;
646:              seek(FGR,r);
647:              write(FGR,aGR);            {PUT #5, R, GR }
648:              end; {ELSE}
649:           end {THEN}
650:        ELSE  Begin
651:              S23 := TAN(A2I + A3I);
652:              posi:= round(pnt1);  {GET #4, PT1, GPN }
653:              seek(FGP,posi);
654:              read(FGP,aGPN);
655:              aGR.XX1 := aGPN.XX;
656:              aGR.YY1 := aGPN.YY;
657:              posi:= round(pnt2);      {GET #4, PT2, GPN }
658:              seek(FGP,posi);
659:              read(FGP,aGPN);
660:              X2 := aGPN.XX;
```

```
661:              Y2 := aGPN.YY;
662:              aGR.XX2 := X2;
663:              aGR.YY2 := Y2;
664:              X3 := ((S23 * X2) - Y2) / S23;
665:              Y3 := 0;
666:              aGPN.XX := X3;
667:              aGPN.YY := Y3;
668:              posi:= round(pnt3);      {PUT #4, PT3, GPN}
669:              seek(FGP,posi);
670:              write(FGP,aGPN);
671:              aGR.XX3 := X3;
672:              aGR.YY3 := Y3;
673:              seek(FGR,R);          {PUT #5, R, GR}
674:              write(FGR,aGR);
675:              end; {ELSE}
676:         end;   {if}
677:      end; {For}
678:
679: End;
680:
681: Procedure Calcula3c(k,count3c,FLA3c:integer;pnt1,pnt2,pnt3:SmallVetor;
682:                      GPN3c:GPNO; var GR3c:gridpoint);
683: var
684: posi:longint;
685: begin
686: GR3c.CELLNO := COUNT3c;
687: GR3c.FLAG := FLA3c;
688:
689: GR3c.POINT1 := Pnt1[k];   {GET #4, POINT1[J], GPN}
690: posi:=round(pnt1[k]);
691: seek(FGP,posi);
692: read(FGP,GPN3c);
693: GR3c.KI1 := GPN3c.KI;
694: GR3c.KII1 := GPN3c.KII;
695: GR3c.TH1 := GPN3c.THT;
696: GR3c.NU1 := GPN3c.NU;
697: GR3c.XX1 := GPN3c.XX;
698: GR3c.YY1 := GPN3c.YY;
699:
700: GR3c.POINT2 := Pnt2[k];    {GET #4, POINT2[J], GPN }
701: posi:=round(pnt2[k]);
702: seek(FGP,posi);
703: read(FGP,GPN3c);
704: GR3c.KI2 := GPN3c.KI;
705: GR3c.KII2 := GPN3c.KII;
706: GR3c.TH2 := GPN3c.THT;
707: GR3c.NU2 := GPN3c.NU;
708: GR3c.XX2 := GPN3c.XX;
709: GR3c.YY2 := GPN3c.YY;
710:
711: GR3c.POINT3 := Pnt3[k];        {GET #4, POINT3[J], GPN}
712: posi:=round(pnt3[k]);
713: seek(FGP,posi);
714: read(FGP,GPN3c);
715: GR3c.KI3 := GPN3c.KI;
716: GR3c.KII3 := GPN3c.KII;
717: GR3c.TH3 := GPN3c.THT;
718: GR3c.NU3 := GPN3c.NU;
719: GR3c.XX3 := GPN3c.XX;
720: GR3c.YY3 := GPN3c.YY;
721:
722: GR3c.POINT4 := 0;
723: GR3c.KI4 := 0;
724: GR3c.KII4 := 0;
725: GR3c.TH4 := 0;
726: GR3c.NU4 := 0;
```

```
GR3c.XX4 := 0;
GR3c.YY4 := 0;
                {PUT #5, COUNT, GR}
seek(FGR,count3c);
write(FGR,GR3c);
exit;
end;

Procedure Calcula4c(k,count4c,FLA4c:integer;pnt1,pnt2,pnt3,pnt4:SmallVetor;GPN4c:GP
                    var GR4c:gridpoint);
var
posi:longint;
begin
GR4c.CELLNO := count4c;
GR4c.FLAG := FLA4c;

GR4c.POINT1 := Pnt1[k];        {GET #4, POINT1[J], GPN }
posi:=round(pnt1[k]);
seek(FGP,posi);
read(FGP,GPN4c);
GR4c.KI1 := GPN4c.KI;
GR4c.KII1 := GPN4c.KII;
GR4c.TH1 := GPN4c.THT;
GR4c.NU1 := GPN4c.NU;
GR4c.XX1 := GPN4c.XX;
GR4c.YY1 := GPN4c.YY;

GR4c.POINT2 := Pnt2[k];         {GET #4, POINT2[J], GPN }
posi:=round(pnt2[k]);
seek(FGP,posi);
read(FGP,GPN4c);
GR4c.KI2 := GPN4c.KI;
GR4c.KII2 := GPN4c.KII;
GR4c.TH2 := GPN4c.THT;
GR4c.NU2 := GPN4c.NU;
GR4c.XX2 := GPN4c.XX;
GR4c.YY2 := GPN4c.YY;

GR4c.POINT3 := Pnt3[k];           {GET #4, POINT3[J], GPN }
posi:=round(pnt3[k]);
seek(FGP,posi);
read(FGP,GPN4c);
GR4c.KI3 := GPN4c.KI;
GR4c.KII3 := GPN4c.KII;
GR4c.TH3 := GPN4c.THT;
GR4c.NU3 := GPN4c.NU;
GR4c.XX3 := GPN4c.XX;
GR4c.YY3 := GPN4c.YY;

GR4c.POINT4 := Pnt4[k];    { GET #4, POINT4[J], GPN}
posi:=round(pnt4[k]);
seek(FGP,posi);
read(FGP,GPN4c);
GR4c.KI4 := GPN4c.KI;
GR4c.KII4 := GPN4c.KII;
GR4c.TH4 := GPN4c.THT;
GR4c.NU4 := GPN4c.NU;
GR4c.XX4 := GPN4c.XX;
GR4c.YY3 := GPN4c.YY;

seek(FGR,count4c);
write(FGR,GR4c);
        {PUT #5, COUNT, GR}

end;
{Procedure TForm1.IRANP;  }
```

```
793: Procedure IRANP(ROW,icl:integer;var aGPN:GPNO; aGR:gridpoint);
794: var posi:longint;
795:     i,j,FLA,count,cnt : integer;
796:     K2I,K2II,THT2,NU2,K3I,K3II,THT3,NU3,K4I,K4II,THT4,NU4:single;
797:     Point1,Point2,Point3,Point4 ,CELLR : SmallVetor;
798: Begin
799: count:=0;
800: write(FGR,aGR);
801: FOR I:= 1 TO ROW do
802:     begin
803:     IF I = 1 THEN CELLR[I]:= ICL ELSE CELLR[I]:= ICL + 2 - I;
804:
805:     FOR J:= 1 TO CELLR[I] do
806:         begin
807:         count:= count + 1;
808:         cnt:=count;
809:         IF (I = 1) AND (J < ICL) THEN
810:            begin
811:            FLA:=0;
812:            POINT1[J] := CNT;
813:            POINT2[J]:= CNT + 1;
814:            POINT3[J]:= CNT + ICL;
815:            POINT4[J]:= CNT + ICL + 1;
816:            CALCULA4c(j,count,FLA,Point1,Point2,Point3,Point4,aGPN,aGR);
817:            end;
818:         IF (I = 1) AND (J = ICL) THEN
819:            begin
820:            POINT1[J]:= CNT;
821:            POINT2[J] := CNT + ICL + 1;
822:            POINT3[J] := CNT + ICL;
823:                      {GET #4, POINT3(J), GPN}
824:            posi:=round(point3[j]);
825:            seek(FGP,posi);
826:            read(FGP,aGPN);
827:            FLA := 10;
828:            K2I:= aGPN.KI;
829:            K2II:= aGPN.KII;
830:            THT2:= 0.5 * (K2I + K2II);
831:            NU2:= 0.5 * (K2I - K2II);
832:            aGPN.KI := K2I;
833:            aGPN.KII := K2II;
834:            aGPN.THT := THT2;
835:            aGPN.NU := NU2;
836:                  {PUT #4, POINT2(J), GPN}
837:            posi:=round(point2[j]);
838:            seek(FGP,posi);
839:            write(FGP,aGPN);
840:            CALCULA3c(j,count,FLA,point1,point2,point3,aGPN,aGR);
841:            end;     {I=1; J=ICL}
842:         IF (I > 1) AND (J = 1) THEN
843:            begin
844:            POINT1[J] := CNT + I - 2;
845:            POINT2[J] := CNT + I - 1;
846:            POINT3[J] := CNT + ICL + 1;
847:                     { GET #4, POINT2(J), GPN}
848:            posi:=round(point2[j]);
849:            seek(FGP,posi);
850:            read(FGP,aGPN);
851:            FLA := 0;
852:            K3I := aGPN.KI;
853:            K3II := -K3I;
854:            THT3 := 0.5 * (K3I + K3II);
855:            NU3 := 0.5 * (K3I - K3II);
856:            aGPN.KI := K3I;
857:            aGPN.KII := K3II;
858:            aGPN.THT := THT3;
```

```
859:             aGPN.NU := NU3;
860:               { PUT #4, POINT3(J), GPN    }
861:             posi:=round(point3[j]);
862:             seek(FGP,posi);
863:             write(FGP,aGPN);
864:             CALCULA3c(j,count,FLA,point1,point2,point3,aGPN,aGR);
865:             end;      { I>1; J=1 }
866:          IF (I > 1) AND (J > 1) THEN
867:              begin     { del 990} {1}
868:              POINT1[J] := CNT + I - 2;
869:              POINT2[J] := CNT + I - 1;
870:              POINT3[J] := CNT + ICL;
871:              POINT4[J] := CNT + ICL + 1;
872:              IF J = CELLR[I] THEN
873:                 begin                                    {2}
874:                        {GET #4, POINT3(J), GPN}
875:                 posi:=round(Point3[j]);
876:                 seek(FGP,posi);
877:                 read(FGP,aGPN);
878:                 FLA := 10;
879:                 K4I := aGPN.KI;
880:                 K4II := aGPN.KII;
881:                 THT4 := 0.5 * (K4I + K4II);
882:                 NU4 := 0.5 * (K4I - K4II);
883:                 aGPN.KI := K4I;
884:                 aGPN.KII := K4II;
885:                 aGPN.THT := THT4;
886:                 aGPN.NU := NU4;
887:                      {PUT #4, POINT4(J), GPN}
888:                 posi:=round(point4[j]);
889:                 seek(FGP,posi);
890:                 write(FGP,aGPN);
891:
892:                 end          {2}
893:              else
894:                 begin                                    {3}
895:                 posi:=round(point2[j]);
896:                 seek(FGP,posi);
897:                 read(FGP,aGPN);           {GET #4, POINT2(J), GPN }
898:                 K4I := aGPN.KI;
899:                 posi:=round(point3[j]);
900:                 seek(FGP,posi);
901:                 read(FGP,aGPN);      {GET #4, POINT3(J), GPN}
902:                 K4II := aGPN.KII;
903:                 FLA := 0;
904:                 THT4 := 0.5 * (K4I + K4II);
905:                 NU4 := 0.5 * (K4I - K4II);
906:                 aGPN.KI := K4I;
907:                 aGPN.KII := K4II;
908:                 aGPN.THT := THT4;
909:                 aGPN.NU := NU4;
910:                       {PUT #4, POINT4(J), GPN}
911:                 posi:=round(point4[j]);
912:                 seek(FGP,posi);
913:                 write(FGP,aGPN);
914:                 end;      {3}
915:              CALCULA4c(j,count,FLA,Point1,Point2,Point3,Point4,aGPN,aGR);
916:              end;  { I>1 ; J>1 }
917:          end;{ de j}
918:      end;     { de i}
919: end;   {prc}
920:
921: {Procedure TForm1.CIPAKC(N:integer;var agpn:gpno); }
922: Procedure CIPAKC(N: Integer;Thetamax,Defang:single; var aGPN:gpno; NU:BigVetor);
923: var I:integer ;
924:      MC,MU,THT,KI,KII,XX,YY: BigVetor;
```

```
925: Begin
926: FOR I:= 1 TO (2*N) do   { looping para zerar os vetores }
927:     begin
928:     XX[I]:= 0; YY[I]:= 0; NU[I]:= 0; MU[I]:= 0; MC[I]:= 0; THT[I]:= 0;
929:     KI[I]:= 0; KII[I]:= 0;
930:     end;
931: write(fgp,aGPN);
932: FOR I:= 1 TO N do
933:     begin
934:     XX[I] := 0;
935:     YY[I] := 1;
936:     IF I = 1 THEN THT[I] := Thetamax - ((N - 1) * DEFANG)
937:        ELSE THT[I] := THT[I - 1] + DEFANG;
938:     NU[I] := THT[I];
939:     KI[I] := THT[I] + NU[I];
940:     KII[I] := THT[I] - NU[I];
941:     aGPN.KI := KI[I];
942:     aGPN.KII := KII[I];
943:     aGPN.THT := THT[I];
944:     aGPN.NU := NU[I];
945:     aGPN.MC := MC[I];
946:     aGPN.MU := MU[I];
947:     aGPN.XX := XX[I];
948:     aGPN.YY := YY[I];
949:              { PUT #4, N, GPN}
950:     write(FGP,aGPN);
951:     end;
952: FOR I := (N + 1) TO (2 * N) do
953:     begin
954:     IF I = (N + 1) THEN THT[I] := Thetamax - ((N - 1) * DEFANG)
955:      ELSE THT[I] := THT[I - 1] + DEFANG;
956:     NU[I] := THT[I];
957:     KI[I] := THT[I] + NU[I];
958:     KII[I] := THT[I] - NU[I];
959:     aGPN.KI := KI[I];
960:     aGPN.KII := KII[I];
961:     aGPN.THT := THT[I];
962:     aGPN.NU := NU[I];
963:             { PUT #4, NN, GPN}
964:     write(FGP,aGPN);
965:     end;
966:
967: end;
968:
969: procedure TForm1.BitBtn2Click(Sender: TObject);
970: begin
971: Edit1.clear;
972: Edit2.Clear;
973: Edit3.Clear;
974:
975: end;
976:
977: PROCEDURE TForm1.BitBtn1Click(Sender: TObject);
978: {Rotina principal do programa; interface gráfica inicial}
979:
980: var
981:    RF : ^Keeper;
982:    ptGr : ^gridpoint;
983:    ptGPN : ^GPNO;
984:    aNU : BigVetor;
985:    cel : SmallVetor;
986:
987:    a,b,c,numdr,numd,thetam,DefAng:single;
988:    code,cell,row,gp,ct,cont:integer;
989:    DC : TCoord;
990:
```

```
 991: begin
 992: rewrite(Fgr);
 993: rewrite(Fgp);
 994: new(ptGr);
 995: new(ptGPN);
 996: new(RF);
 997:   if controle=1 then  begin
 998:   form3.table1.open;
 999:   form3.table1.edit;
1000:   form3.table1.first;
1001:  while not form3.table1.EOF do
1002:            begin
1003:            form3.Table1.Edit;
1004:            form3.table1.delete;
1005:            end;
1006:                end;
1007: form3.table1.close;
1008: Val(Edit1.Text,ed1, Code);       {Leitura de dados pela interface      }
1009: Val(Edit2.Text,ed2, Code);       {  com usuário                        }
1010: Val(Edit5.Text,t0,code);
1011: Val(Edit6.Text,p0,code);
1012: Val(Edit7.Text,R,code);
1013: P0:=P0*_105;
1014: icl:=strtoInt(Edit4.text);
1015: b:=(ed2+1)/(ed2-1);
1016: a:=sqrt((1/b)*(sqr(ed1)-1));
1017: c:=sqrt((sqr(ed1)-1));
1018: numdr:=(sqrt(b)*ArcTan(A))-ArcTan(C);
1019: numd:=numdr*(180/pi);
1020: thetam:=numd/2;
1021: Val(edit3.text,fm,Code);
1022: { fm : é o diâmetro em mm, que serve como fator de multiplicação}
1023: fm:=fm/1000;
1024: cel[1]:=0;
1025: DefAng:=Thetam/ICL;
1026: for cont:=2 to ICL do cel[cont]:=cel[cont-1]+cont;
1027: cell:=cel[icl]+icl;
1028: gp:=cell+icl+1;
1029: row:=icl;
1030:                          {Cálculos isoentrópicos iniciais}
1031: p:=P0*power((2/(ed2+1)),(ed2/(ed2-1)));
1032: t:=T0*(2/(ed2+1));
1033: ro:=p/(t*R);
1034: a0:=sqrt(ed2*R*T0);
1035: aa:=sqrt(ed2*R*t);
1036: aaa:=(pi*sqr(fm))/4;
1037: w:=ro*aaa*aa;
1038: {Calcula as propriedades dos pontos iniciais:}
1039: CIPAKC(icl,Thetam,DefAng,ptGPN^,aNU);
1040: {Monta as células de todo o perfil:}
1041: IRANP(row,icl,ptGPN^,ptGR^);
1042: {Calcula Mach e ângulo de Mach(μ) para todos os pontos do grid:}
1043: for cont:=1 to gp do Secant(B,cont,ptGPN^,aNu);
1044: {Transfere o resultado anterior para a matriz de células}
1045: for cont:=1 to cell do MyStore(ptGPN^,ptGR^,cont);
1046: {Calcula as cordenadas x,y de todos os pontos das células. Cada célula
1047:   é um pedaço do perfil com 4 pontos(interior) ou 3 pontos(nas bordas)}
1048: CCXY(cell,icl,thetam,DefAng,ptGPN^,ptGR^);
1049: {Tranfere o resultado para a matriz de pontos do grid}
1050: for cont:=1 to cell do FinalStore(ptGPN^,ptGR^,cont);
1051: reset(FGR);
1052:
1053: AssignFile(OutPut,'c:\saida.txt');
1054: writeln('no ','    TH   ','   NU   ','  Mach    ',
1055:         ' MU   ','  XX (m) ',' YY (m)     ');
1056: for cont:=1 to (cell+icl+1) do
```

```
1057:    begin                                    {Saida simples para impressora}
1058:    seek(FGp,cont);
1059:    read(FGp,ptGpn^);
1060:    writeln(cont:3,ptGPN^.THT:8:3,ptGPN^.NU:8:3,ptGPN^.MC:8:3,
1061:            ptGPN^.MU:8:3,ptGPN^.XX:8:3,ptGPN^.YY:8:3);
1062:    end;
1063: reset(OutPut);
1064:
1065: ct:=0;
1066: form3.table1.open;
1067: for cont:=1 to cell do
1068:   begin
1069:   seek(FGR,cont);              {Monta uma janela de saída de dados}
1070:   read(FGR,ptGR^);
1071:
1072:   if ptGr^.flag<>0 then
1073:      begin
1074:      if (ptGr^.xx4=0) and (ptGr^.yy4=0) then
1075:         begin
1076:         ct:=ct+2;
1077:         form2.stringgrid1.Cells[0,0]:='Perfil';
1078:         form2.stringgrid1.Cells[1,0]:=' X (m)';
1079:         form2.stringgrid1.Cells[2,0]:=' Y (m)';
1080:
1081:         form2.stringgrid1.cells[0,ct-1]:=inttostr(ct-1);
1082:         form2.stringgrid1.cells[1,ct-1]:=floattostr(ptGr^.XX1);
1083:         form2.stringgrid1.cells[2,ct-1]:=floattostr(ptGr^.YY1);
1084:
1085:         RF[ct-1].X:=ptGr^.xx1;
1086:         RF[ct-1].Y:=ptGr^.yy1;
1087:
1088:         form2.stringgrid1.cells[0,ct]:=inttostr(ct);
1089:         form2.stringgrid1.cells[1,ct]:=floattostr(ptGr^.XX2);
1090:         form2.stringgrid1.cells[2,ct]:=floattostr(ptGr^.YY2);
1091:
1092:         RF[ct].X:=ptGr^.xx2;
1093:         RF[ct].Y:=ptGr^.yy2;
1094:         end;
1095:
1096:      if (ptGr^.xx4<>0) and (ptGr^.yy4<>0)   then
1097:         begin
1098:         ct:=ct+1;
1099:         form2.stringgrid1.cells[0,ct]:=inttostr(ct);
1100:         form2.stringgrid1.cells[1,ct]:=floattostr(ptGr^.XX4);
1101:         form2.stringgrid1.cells[2,ct]:=floattostr(ptGr^.YY4);
1102:
1103:         RF[ct].X:=ptGr^.xx4;
1104:         RF[ct].Y:=ptGr^.yy4;
1105:         end;
1106:      end;
1107:
1108:   form2.show;
1109:   end;
1110: form3.table1.open;
1111: form3.table1.first;
1112:
1113:
1114:
1115: case controle of          {Monta gráficos de até 5 perfis diferentes}
1116:
1117: 1: begin
1118: for cont:=1 to icl do
1119:     begin
1120:     form3.table1.edit;
1121:     form3.table1.FieldByName('XX').asfloat:=RF[cont].X*fm;
1122:     form3.table1.FieldByName('YY').asfloat:=RF[cont].Y*fm;
```

```
1123:      form3.table1.Post;
1124:      form3.table1.Append;
1125:      end;
1126:      end;
1127:
1128: 2: begin
1129:      form3.table1.First;
1130: for cont:=1 to icl do
1131:      begin
1132:      form3.table1.edit;
1133:      form3.table1.FieldByName('XX1').asfloat:=RF[cont].X*fm;
1134:      form3.table1.FieldByName('YY1').asfloat:=RF[cont].Y*fm;
1135:      form3.table1.Post;
1136:      {form3.table1.Append;  }
1137:      form3.table1.next;
1138:      end;
1139:      end;
1140:
1141: 3: begin
1142:         form3.table1.First;
1143: for cont:=1 to icl do
1144:      begin
1145:      form3.table1.edit;
1146:      form3.table1.FieldByName('XX2').asfloat:=RF[cont].X*fm;
1147:      form3.table1.FieldByName('YY2').asfloat:=RF[cont].Y*fm;
1148:      form3.table1.Post;
1149:     { form3.table1.Append;}
1150:      form3.table1.next;
1151:      end;
1152:      end;
1153:
1154: 4: begin
1155:      form3.table1.First;
1156: for cont:=1 to icl do
1157:      begin
1158:      form3.table1.edit;
1159:      form3.table1.FieldByName('XX3').asfloat:=RF[cont].X*fm;
1160:      form3.table1.FieldByName('YY3').asfloat:=RF[cont].Y*fm;
1161:      form3.table1.Post;
1162:      {form3.table1.Append;}
1163:      form3.table1.next;
1164:      end;
1165:      end;
1166:
1167: 5: begin
1168:     form3.table1.First;
1169: for cont:=1 to icl do
1170:      begin
1171:      form3.table1.edit;
1172:      form3.table1.FieldByName('XX4').asfloat:=RF[cont].X*fm;
1173:      form3.table1.FieldByName('YY4').asfloat:=RF[cont].Y*fm;
1174:      form3.table1.Post;
1175:     { form3.table1.Append;  }
1176:     form3.table1.next;
1177:      end;
1178:      end;
1179:
1180:      end; {case }
1181: {}
1182: DC:=CX2(strtoint(edit4.text),icl);
1183:    form2.StringGrid1.cells[3,0]:='Ro';
1184:    {for ct:=1 to (strtoint(edit4.text) div 2) do }
1185:
1186: Forza:=CalForza(strtoint(edit4.text),DC);
1187:   form2.StringGrid1.cells[4,0]:='Força';
1188:   form2.StringGrid1.cells[5,0]:='Theta';
```

```
1189:  form2.StringGrid1.cells[6,0]:='ni';
1190:  form2.StringGrid1.cells[7,0]:='Mach';
1191:  form2.StringGrid1.cells[8,0]:='μ';
1192:  form2.StringGrid1.cells[9,0]:='Temp(K)';
1193:  form2.StringGrid1.cells[10,0]:='Vel(m/s)';
1194:  form2.StringGrid1.cells[11,0]:='Pressão(Pa)';
1195:
1196:  { for ct:=1 to strtoint(edit4.text)-1 do   }
1197:    for ct:=1 to icl do
1198:        begin
1199:        form2.stringgrid1.cells[3,ct+1]:=floattostr(DC[ct]);
1200:        form2.stringgrid1.cells[4,ct+1]:=floattostr(Forza[ct]);
1201:
1202:        end;
1203:
1204:     cont:=1;
1205:     reset(fgp);
1206:     seek(fgp,(strtoint(edit4.text)+1));
1207:     while not eof(fgp) do
1208:      begin
1209:      read(fgp,agpno[cont]);
1210:      inc(cont);
1211:      end; {end del while}
1212:
1213: {for ct:=1 to filesize(fgp)-(strtoint(edit4.text)+1) do  }
1214:  for ct:=1 to (filesize(fgp)-icl) do
1215:     begin
1216:     form2.stringgrid1.cells[5,ct+1]:=floattostr(agpno[ct].tht);
1217:     form2.stringgrid1.cells[6,ct+1]:=floattostr(agpno[ct].nu);
1218:     form2.stringgrid1.cells[7,ct+1]:=floattostr(agpno[ct].mc);
1219:     form2.stringgrid1.cells[8,ct+1]:=floattostr(agpno[ct].mu);
1220:     form2.stringgrid1.cells[9,ct+1]:=floattostr(Tc[ct]);
1221:     form2.stringgrid1.cells[10,ct+1]:=floattostr(Vc[ct]);
1222:     form2.stringgrid1.cells[11,ct+1]:=floattostr(Pp[ct]);
1223:
1224:     end;
1225: dispose(ptGPN);
1226: dispose(ptGR);
1227: dispose(RF);
1228: End;
1229:
1230:  procedure TForm1.Exit1Click(Sender: TObject);
1231: begin
1232: if MessageDlg('Saindo do Sistema',MtConfirmation,MbOkCancel,0)=mrOk then
1233: Form1.Close;
1234: end;
1235:
1236: procedure TForm1.Save1Click(Sender: TObject);
1237: begin
1238:   SaveDialog1.Execute;
1239: end;
1240:
1241: procedure TForm1.Load1Click(Sender: TObject);
1242: begin
1243:  OpenDialog1.Execute;
1244: end;
1245:
1246: procedure TForm1.SpeedButton1Click(Sender: TObject);
1247: begin
1248:  OpenDialog1.Execute;
1249: end;
1250:
1251: procedure TForm1.SpeedButton4Click(Sender: TObject);
1252: begin
1253:  if MessageDlg('Saindo do Sistema',MtConfirmation,MbOkCancel,0)=mrOk then
1254: Form1.Close;
```

```
1255: end;
1256:
1257: procedure TForm1.FormCreate(Sender: TObject);
1258: begin
1259: AssignFile(Fgr,'Gr.dat');
1260:  rewrite(Fgr);
1261:  reset(fgr);
1262:  reset(fgr);
1263: AssignFile(Fgp,'Gp.dat');
1264:  rewrite(Fgp);
1265:  reset(fgp);
1266: {ResetTable(table1); }
1267: controle:=1;
1268:
1269: end;
1270:
1271: procedure TForm1.ResetTable(Sender: TObject);
1272: begin
1273: {table1.open;
1274: table1.first;
1275: while not table1.EOF do
1276:     table1.delete;
1277:     {table1.post; }
1278:
1279: {table1.close;
1280:  }
1281:
1282: end;
1283:
1284: end.
```

```
unit tresult;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids, StdCtrls, Buttons;

type
  TForm2 = class(TForm)
    StringGrid1: TStringGrid;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
    Procedure EscolheLinha;

  end;

var
  Form2: TForm2;
  controle:integer;
  serie :shortstring;
implementation

uses UG, U1U, U4, U5;

{$R *.DFM}

Procedure Tform2.EscolheLinha;
Begin
case controle of                    {Apresenta os gráficos de perfil}
 1: begin
    Form3.Series1.Clear;
    with Form1 do
         while not form3.Table1.EOF do
         begin
         Form3.Series1.AddXY(form3.Table1.FieldbyName('XX').AsFloat,
                             form3.Table1.FieldbyName('YY').AsFloat,
                             floattostr(form3.table1.FieldbyName('XX').AsFloat),
                             Form3.Series1.ValueColor[10]);

         form3.Table1.next;
         end;
    end;
 2: begin
    Form3.Series2.Clear;
    with Form1 do
         while not form3.Table1.EOF do
         begin
         Form3.Series2.AddXY(form3.Table1.FieldbyName('XX').AsFloat,
                             form3.Table1.FieldbyName('YY').AsFloat,
                             floattostr(form3.table1.FieldbyName('XX').AsFloat),
                             Form3.Series2.ValueColor[20]);

         form3.Table1.next;
         end;
    end;
 3: begin
    Form3.Series3.Clear;
    with Form1 do
```

```
67:          while not form3.Table1.EOF do
68:          begin
69:          Form3.Series3.AddXY(form3.Table1.FieldbyName('XX').AsFloat,
70:                              form3.Table1.FieldbyName('YY').AsFloat,
71:                              floattostr(form3.table1.FieldbyName('XX').AsFloat),
72:                              Form3.Series3.ValueColor[30]);
73:
74:          form3.Table1.next;
75:          end;
76:     end;
77:  4: begin
78:     Form3.Series4.Clear;
79:     with Form1 do
80:          while not form3.Table1.EOF do
81:          begin
82:          Form3.Series4.AddXY(form3.Table1.FieldbyName('XX').AsFloat,
83:                              form3.Table1.FieldbyName('YY').AsFloat,
84:                              floattostr(form3.table1.FieldbyName('XX').AsFloat),
85:                              Form3.Series4.ValueColor[40]);
86:
87:          form3.Table1.next;
88:          end;
89:     end;
90:  5: begin
91:     Form3.Series5.Clear;
92:     with Form1 do
93:          while not form3.Table1.EOF do
94:          begin
95:          Form3.Series5.AddXY(form3.Table1.FieldbyName('XX').AsFloat,
96:                              form3.Table1.FieldbyName('YY').AsFloat,
97:                              floattostr(form3.table1.FieldbyName('XX').AsFloat),
98:                              Form3.Series5.ValueColor[50]);
99:
100:         form3.Table1.next;
101:         end;
102:    end;
103:
104: end; {case}
105: End;
106:
107: Procedure TForm2.BitBtn2Click(Sender: TObject);
108:
109: Begin
110:
111: form3.table1.open;
112: form3.table1.active;
113: form3.table1.First;
114: {form1.Table1.Edit;}
115:
116:
117: if controle > 5 then controle:=1;
118: {EscolheLinha; }
119: form3.DBChart1.Series[controle-1].Title:='Mac '+form1.Edit1.text;
120: Form3.show;
121: {Form1.ResetTable(form.table1); }
122: controle:=controle+1;
123: end;
124:
125: procedure TForm2.BitBtn1Click(Sender: TObject);
126: begin
127:
128:
129: form4.quickrep1.print;
130: form3.table1.IndexName:='';
131:
132: end;
```

```
133:
134: END.
```

```
unit UG;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Series, DBChart, TeEngine, ExtCtrls, TeeProcs, Chart, Db, DBTables,
  OleCtrls, chartfx3, StdCtrls, Buttons;

type
  TForm3 = class(TForm)
    Table1: TTable;
    DBChart1: TDBChart;
    Series1: TLineSeries;
    Series2: TLineSeries;
    Series3: TLineSeries;
    Series4: TLineSeries;
    Series5: TLineSeries;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    procedure BitBtn2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

{$R *.DFM}

procedure TForm3.BitBtn2Click(Sender: TObject);
begin

dbchart1.PrintLandscape;
end;

end.
```

```
 1: unit U4;
 2:
 3: interface
 4:
 5: uses
 6:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 7:   Qrctrls, quickrpt, ExtCtrls, Db, DBTables;
 8:
 9: type
10:   TForm4 = class(TForm)
11:     QuickRep1: TQuickRep;
12:     QRLabel1: TQRLabel;
13:     QRLabel2: TQRLabel;
14:     QRLabel3: TQRLabel;
15:     QRSysData1: TQRSysData;
16:     QRDBText1: TQRDBText;
17:     QRDBText2: TQRDBText;
18:   private
19:     { Private declarations }
20:   public
21:     { Public declarations }
22:   end;
23:
24: var
25:   Form4: TForm4;
26:
27: implementation
28:
29: uses UG;
30:
31: {$R *.DFM}
32:
33:
34:
35: end.
```

X) Conclusões

Fica claro através dos testes com o programa Bocal que o perfil é extremamente sensível, em forma e tamanho, ao número de Mach e à constante γ do gás. Por outro lado a força de empuxo do bocal varia muito pouco em função do Mach, se nos basearmos pelo programa, e varia consideravelmente com a pressão de entrada no bocal e mais ainda com o diâmetro da garganta. Estas variáveis são fundamentais para o tempo de vazão dos gases de queima e, no caso do transitório na câmara de combustão, para o controle da própria pressão de combustão.

Para os dados de caso dado, com pressão de entrada de 150atm, temperatura de entrada de 2500K, γ=1.2, R=207.85 e diâmetro de garganta igual a 8mm, o valor de Mach ideal parece ser de 2.1, onde a pressão de descarga é bem próxima e superior à pressão atmosférica e o empuxo máximo é de cerca de 930N. Comparando com um cálculo isoentrópico unidimensional, chegamos a um valor 86% na relação caso bidimensional por caso ideal.

É claro que muitos fatores estruturais, materiais e de aerodinâmica externa devem ser levados em conta antes da construção de um protótipo de foguete. Entretanto, os objetivos deste estudo foram alcançados, revelando uma ferramenta poderosa para este tipo de projeto.

Futuros desenvolvimentos podem ser feitos a partir da etapa alcançada, como por exemplo o uso do perfil obtido em um método bidimensional mais completo como o das diferenças finitas, obtendo-se valores mais precisos no interior do bocal para grandezas como pressão e temperatura. O perfil tambem pode ser projetado com mais rigor, analisando-se o transitório de pressão na entrada e na saída do bocal, de forma a garantir que o bocal fique blocado a maior parte do tempo, sem no entanto comprometer a câmara de combustão com pressões acima da resistência do material.

Não se descarta tambem o uso do programa no projeto de túneis de vento supersônicos, embora talvez fossem necessárias algumas correções para obter linhas de escoamento paralelas.

XI) Bibliografia

Anderson, J.D.,Jr. (1990). *Modern Compressible Flow with Historical Perspective*, 2nd ed. McGraw-Hill, New York.

Hodge, B.K. (1995). *Compressible Fluid Dynamics*. Prentice-Hall, New Jersey

Owczarek, K. (1964). *Fundamentals of Gas Dyanamics*, International, Scranton, Pa.

Shapiro, A.H. (1953). *The Dynamics and Thermodynamics of Compressible Fluid Flow*. Ronald Press, New York.

Sutton, G.P. (1992). *Rocket Propulsion Elements*, 6th ed. John Wiley, New York.

Zucrow, M.J.,and Hoffman, J.D. (1977). *Gas Dynamics*. John Wiley, New York; Vol. 1,2.