EURYALE JORGE GODOY DE JESUS ZERBINI

SIMULAÇÃO NUMÉRICA DO ESCOAMENTO EM CÂMARAS CICLÔNICAS COM ELEVADA RECIRCULAÇÃO

TESE APRESENTADA À ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO PARA A OBTENÇÃO DO TÍTULO DE DOUTOR EM ENGENHARIA

ORIENTADOR : PROF. DR. CLEMENTE GRECO
PROFESSOR ASSISTENTE DOUTOR DO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA EPUSP

SÃO PAULO
1992

AGRADECIMENTOS

Ao meu orientador Prof. Clemente Greco que me incentivou, desde o meu Curso de Graduação, a procurar entender a Natureza e que realizou um trabalho muito mais amplo e coerente do que apenas orientar meu Programa de Doutorado. Além disso foi um amigo (o Humberto Eco tem razão) nestes anos duros e turbulentos.

Ao meu amigo e orientador de Mestrado Prof. Otávio de Mattos Silvares pelo constante interesse e apoio desde o meu Curso de Graduação e por ter me mostrado que a pesquisa em engenharia pode ser uma experiência muito interessante.

Ao Prof. Marcos de M. Pimenta pelas frutuosas conversas e sugestões apresentadas no Exame de Qualificação.

Aos Profs. Hernani L. Brinati e Marco A. Brinati pela amizade e pelo incentivo fornecido durante toda a minha permanência na Universidade de São Paulo.

A Olga e Eunice pela compreensão e finalmente ao Gastão pela companhia nas noites escuras e divergentes.

RESUMO

Este trabalho apresenta modelos matemáticos para a simulação do escoamento turbulento em câmaras ciclônicas , admitindo que este seja axissimétrico , e sua implementação em microcomputadores. As condições experimentais de Ustimenko e Bukhman [98] foram utilizadas como referência para o estudo do comportamento dos modelos matemáticos utilizados. As tensões aparentes de Reynolds são obtidas a partir de duas modelagens distintas: a primeira utilizando o conceito de viscosidade efetiva e baseado nas equações de transporte da energia cinética da turbulência e de sua taxa de dissipação, e a segunda baseada num modelo proposto que avalia as tensões de Reynolds na forma algébrica.

É mostrado que o modelo baseado no conceito de viscosidade efetiva falha na descrição de muitos aspectos importantes do escoamento que ocorre em câmaras ciclônicas e que os resultados obtidos com a utilização do modelo algébrico das tensões de Reynolds apresentam melhor aderência àqueles obtidos por via experimental.

ABSTRACT

The problem of confined axisymmetric turbulent flows with strong swirl encountered in cyclone chambers is formulated in terms of the momentum and mass conservation equations and solved on a micro - computer using the finite volume approach. The computer program is applied to simulate the experimental conditions described by Ustimenko and Bukhman [98] for this kind of flow. The Reynolds stresses are obtained first from the usual and modified two - equation turbulence models , based on the solution of the two transport equations of turbulent kinetic energy and its dissipation rate , and latter from two proposed algebraic Reynolds stress models. It is show that the first closure fails to reproduce many important features of the cyclone chamber flow and the results obtained with the algebraic Reynolds stress models display better agreement between the predicted and experimental profiles of axial and tangencial velocity components in the cyclone chamber.

ÍNDICE GERAL

NOMENCLATURA

a	termo de equação algébrica
A	área
C_D	constante empírica
C_{in}	constante que representa a turbulência convectada para o equipamento
C_{gs}	constante empírica
C_k	constante empírica
C_1	constante empírica
C_2	constante empírica
$C_{\varepsilon 1}$	constante empírica
$C_{\varepsilon 2}$	constante empírica
$C_{\varepsilon 3}$	constante empírica
C_μ	constante empírica
D_c	diâmetro da câmara
D_{ij}	tensor de difusão de $< u_i u_j >$
E	constante empírica
f_i	força de campo por unidade de massa na direção i
gen	geração de energia cinética de turbulência por unidade de volume
k	energia cinética turbulenta
ℓ	escala de turbulência
n	número da iteração temporal
p	pressão

p'_P	correção de pressão
p'	flutuação da pressão
P	geração de energia cinética de turbulência por unidade de massa
P_{ij}	tensor geração de energia cinética de turbulência
Pe	número de Peclet
Pr	número de Prandtl
r	coordenada radial
$R_{\Phi,P}$	erro residual para a variável Φ no volume finito referente ao nó P
Resol	somatória normalizada dos erros residuais absolutos
Ri	número de Richardson
S_c	parte da linearização de S_Φ que não é multiplicada por Φ
S_p	parte da linearização de S_Φ que é multiplicada por Φ
S_Φ	termo fonte de Φ
u	flutuação de U
U	velocidade média axial
u_i	flutuação de U_i
U_i	velocidade na direção i
v	flutuação de V
V	velocidade média na direção radial
w	flutuação de W
W	velocidade média tangencial
x	coordenada axial
y	distância normal à parede

α	coeficiente de sub-relaxação
β	constante empírica
γ_{21}	parâmetro na Eq. 3Ø
γ_{31}	parâmetro na Eq. 32
γ_{32}	parâmetro na Eq. 32
Γ	coeficiente de difusão
ε	taxa de dissipação de energia cinética turbulenta
ε_{ij}	tensor taxa de dissipação de energia cinética turbulenta
δ_{xw}	distância entre os nós W e P
δ_{ij}	delta de Kronecker
Δt	intervalo de tempo
ΔV	volume do elemento
$\varkappa$	constante de Von Karman
λ	comprimento de mistura ou função de P e ε
ν	viscosidade cinemática
μ	viscosidade dinâmica
Π_{ij}	tensor de redistribuição pressão-tensão
ρ	massa específica
σ_{rx}	tensão de cisalhamento na direção de x
$\sigma_{r\theta}$	tensão de cisalhamento na direção de θ
σ_{tot}	tensão de cisalhamento total
σ_{w}	tensão de cisalhamento na parede
σ_{ij}	tensor das tensões
Φ	variável geral

os sub-escritos adicionais se referem as seguintes condições :

e	superfície direita
E	nó a direita de P

eff	efetivo
ε	taxa de dissipação de k
in	secção de alimentação
k	energia cinética turbulenta
max	valor máximo
n	superfície superior
N	nó superior a P
P	nó P
s	superfície inferior
S	nó inferior a P
tot	total
turb	turbulento
viz	vizinhos a P
w	superfície esquerda
W	nó a esquerda de P

e os super-escritos adicionais se referem as sequintes condições:

,,,	unidade de volume
~	valor instantâneo
+	adimensional
*	total, ou valor de iteração intermediária ou referente a velocidade de atrito
m	média de Favre
n	número da iteração temporal
novo	valor recém - obtido
P	nó P
velho	valor de iteração já realizada
Φ	a variável geral Φ

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

I INTRODUÇÃO e OBJETIVOS

O projeto e desenvolvimento criterioso de sistemas de combustão envolve o tratamento de um número bastante grande de fenômenos inter-relacionados, os quais mesmo ocorrendo de forma isolada, já se mostram complexos. Por este motivo, o projeto tradicional destes equipamentos baseia-se na utilização intensiva de dados experimentais e correlações empíricas globais, obtidas em equipamentos semelhantes e em condições de operação próximas daquela de projeto, e sobretudo são fundamentadas na experiência anterior do projetista. Mesmo o processo clássico de mudança de escala entre uma planta piloto e uma unidade industrial pode levar a resultados não satisfatórios, pois as regras de mudança de escala, normalmente utilizadas, não são exatas devido a inter-relação e não linearidade dos fenômenos relevantes.

Nos últimos 30 anos foi desenvolvido um conjunto significativo de procedimentos matemáticos que permitem a simulação numérica de escoamentos monocomponente, mono e bifásicos, e multicomponentes com transferência de calor e presença de reações químicas. Neste mesmo período ocorreu tambem um aumento significativo da compreensão e modelagem dos fenômenos superficiais, das reações químicas e principalmente da turbulência. Isto permitiu a criação de novos procedimentos dedicados a simular numericamente os processos físico-químicos que ocorrem nos sistemas de combustão.

Ainda existem diferenças, em muitos casos até significativas, entre os resultados obtidos por simulação numérica e os obtidos experimentalmente, mas estas diferenças diminuem com o refinamento constante dos modelos matemáticos dos fenômenos em questão. Mesmo assim a "Mecânica dos Meios Contínuos Computacional" já pode ser considerada uma ferramenta auxiliar no projeto de equipamentos industriais.

O objetivo desta Tese é a simulação numérica do escoamento de fluído incompressível e monofásico em câmaras ciclônicas admitindo a hipótese de axissimetria. Para este fim serão propostos um modelo de turbulência que avalia as tensões de Reynolds na forma algébrica, obtido a partir do rearranjo de outros modelos, e de uma formulação alternativa para as funções de parede. Este escoamento é muito interessante pois existem condições para a ocorrência de gradientes adversos de pressão que provocam uma forte recirculação na região central do escoamento. O comportamento , formação e o desenvolvimento da zona de recirculação ainda é objeto de especulações teóricas , Escudier e Keller [23], e o tratamento dado por estes ao problema lembra muito o dedicado aos escoamentos compressíveis com velocidades próximas a do som (regime trans-sônico). Esta recirculação é responsável pela alta eficiência na conversão de combustível sólido em combustores ciclônicos.

O procedimento numérico dedicado a simular o escoamento em câmaras ciclônicas é o passo inicial, e o mais importante, para o desenvolvimento de um procedimento numérico confiável dedicado a simulação de combustores ciclônicos. Estes, sem dúvida, serão utilizados no futuro próximo na combustão de sólidos de baixa qualidade.

II FORMULAÇÃO GERAL DO PROBLEMA E ESCOLHA DO CASO PADRÃO

As hipóteses iniciais para a simplificação do tratamento dado a modelagem do escoamento em câmaras ciclônicas são que o fluído é incompressível e Stokesiano, que o escoamento é turbulento e axissimétrico e que o regime é estacionário.

O sistema de coordenadas que melhor se adpta a este tipo de escoamento é o cilíndrico (r , x) onde r é a coordenada radial e x é a axial. Além do uso deste sistema será utilizado o cartesiano, devido a facilidade de operação neste sistema utilizando-se a notação tensorial compacta.

Define-se estado estacionário como aquele que apresenta médias temporais constantes, ou seja independentes da escolha da origem do tempo. Nestas condições, a equação instantânea de Navier, que representa a lei, no sentido empírico, da conservação da quantidade de movimento para um fluído submetido a qualquer tipo de forças moleculares é :

$$\frac{\partial}{\partial x_j} (\tilde{\rho} \tilde{U}_i \tilde{U}_j) = \frac{\partial}{\partial x_j} \tilde{\sigma}^*_{ij} + \tilde{\rho} \tilde{f}_i \qquad (1)$$

onde.

$\tilde{U}_i$ velocidade instantânea na direção i

$\tilde{\sigma}^*_{ij}$ tensor total instantâneo das tensões

$\tilde{f}_i$ força de campo, por unidade de massa, na direção i

$\tilde{\rho}$ massa específica instantânea

O tensor total instantâneo das tensões pode ser desmembrado, utilizando-se as hipóteses de Stokes, numa pressão escalar instantânea , $\tilde{p}$, definida como um terço da soma das tensões normais e num tensor instantâneo , $\tilde{\sigma}_{il}$, referente as deformações e dilatações volumétricas (" bulk "). De acordo com este desmembramento, para um fluído incompressível tem-se:

$$\tilde{\sigma}^{*}_{il} = - \tilde{p}\, \delta_{il} + \tilde{\sigma}_{il} \qquad (2)$$

onde

δ_{il} delta de Kronecker

Para um fluído Stokesiano incompressível, o tensor das tensões instantâneo é definido, por exemplo segundo Bradshaw [14], a partir do tensor das taxas instantâneas de deformação do sequinte modo :

$$\tilde{\sigma}_{il} = \mu \left(\frac{\partial \tilde{U}_i}{\partial x_l} + \frac{\partial \tilde{U}_l}{\partial x_i} \right) \qquad (3)$$

onde

μ viscosidade dinâmica

Aplicando as equações 2 e 3 em 1 , obtem-se uma das formas da equação de Navier - Stokes .

$$\frac{\partial}{\partial x_j}\left(\rho \tilde{U}_i \tilde{U}_j\right) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial \tilde{U}_i}{\partial x_j} + \frac{\partial \tilde{U}_j}{\partial x_i}\right)\right) + \rho \tilde{f}_i \qquad (4)$$

onde

ρ massa específica

No regime estacionário a equação instantânea da conservação da massa é :

$$\frac{\partial \tilde{U}_i}{\partial x_i} = \emptyset \qquad (5)$$

e todas as grandezas físicas podem ser decompostas num valor médio (independente do tempo) e de uma flutuação que apresenta média nula, por exemplo , tomando-se a decomposição de Reynolds, tem-se :

$$\tilde{U}_i = U_i + u_i \qquad (6)$$

com

$$\langle \tilde{U}_i \rangle = U_i$$

e

$$\langle u_i \rangle = \emptyset$$

onde

U_i valor médio de $\tilde{U}_i$

u_i flutuação de $\tilde{U}_i$

$< >$ operador média do tipo "ensemble" definido por :

$$< \tilde{U}_i > = \lim_{N \to \infty} \frac{1}{N} \sum_{1}^{N} \tilde{U}_i$$

Um outro modo de se efetuar a decomposição, útil nos casos onde ocorre variações de massa específica (como nos escoamentos reativos) é o de Favre [6], onde a velocidade mássica média é definida por :

$$U_i^m = \frac{< \tilde{\rho} \tilde{U}_i >}{< \tilde{\rho} >}$$

Comparando as duas decomposições anteriores, nota-se que a adoção da hipótese de incompressibilidade torna as duas definições equivalentes.

Aplicando a decomposição de Reynolds nas equações 4 e 5 , admitindo-se que o efeito das forças de campo sobre o fluído seja desprezível e tomando-se a média da equação, obtem-se :

$$\frac{\partial}{\partial x_j} (\rho U_i U_j) = - \frac{\partial p}{\partial x_i} - \frac{\partial}{\partial x_j} (\rho < u_i u_j >) + \frac{\partial}{\partial x_j} (\mu (\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i})) \quad (7)$$

e

$$\frac{\partial U_j}{\partial x_j} = \emptyset \qquad (8)$$

Para a solução deste sistema de equações, além da obrigatoriedade do estabelecimento das condições de contôrno adequadas, é necessária a determinação dos valores de $\langle \rho u_i u_j \rangle$ no campo de escoamento. É possível construir uma equação para a distribuição das tensões aparentes de Reynolds no campo de escoamento, por exemplo Bradshaw [14], mas esta envolve correlações triplas. Se derivarmos uma equação para as correlações triplas, estas conterão as quádruplas e assim por diante, ou seja, em algum momento no processo de avaliação do termo $\langle \rho u_i u_j \rangle$ será necessária a utilização de hipóteses " ad hoc " para tornar possível a solução do conjunto de equações 7 e 8 . Este tipo de problema é conhecido como o de fechamento (" closure "), e neste caso, o fechamento é realizado pelo modelo de turbulência.

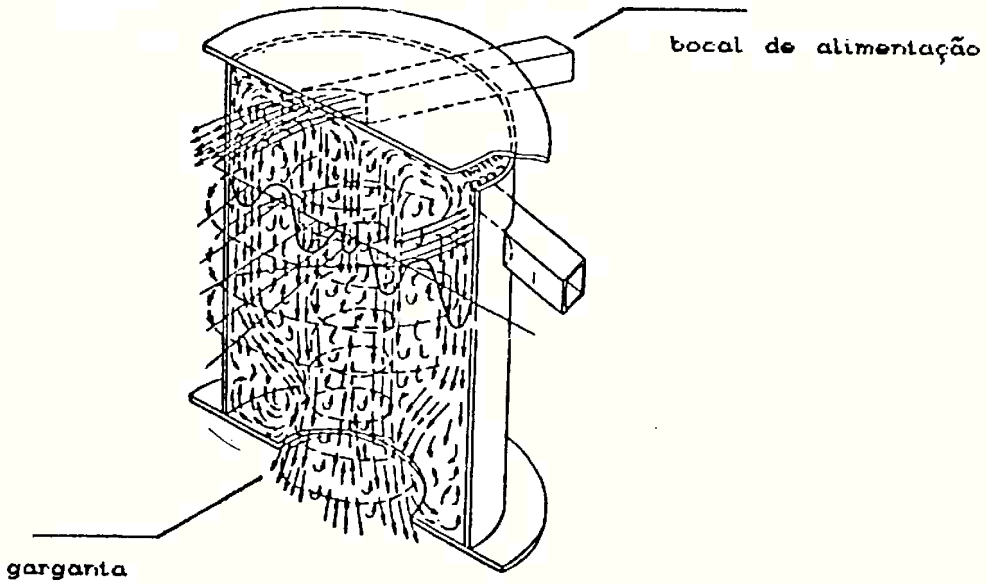


Fig. 1 Vista Geral da Câmara Ciclônica

O escoamento escolhido como padrão e utilizado em todos os testes dos procedimentos numéricos desenvolvidos nesta Tese é o que ocorre em câmaras ciclônicas com quatro bocais de alimentação que foram ensaiados experimentalmente por Ustimenko e e Bukhman [98]. O arranjo utilizado por eles e uma descrição pictórica do escoamento pode se vista na Fig.1

A formulação geral do problema se resume , então, em resolver o sistema de equações 7 e 8 , com condições de contôrno adequadas a geometria e condições de operação das câmaras ciclônicas e utilizando modelos de turbulência para o fechamento do problema.

III REVISÃO DA LITERATURA

Devido a natureza complexa dos escoamentos confinados com componente tangencial intensa, exitem poucos trabalhos publicados que utilizam exclusivamente ferramentas analíticas para a solução do escoamento. Entre eles destacam-se o de Smith [91] e o de Bloor e Inghan [7]. No primeiro destes é analisado o escoamento uni-dimensional de um fluído incompressível em regime estacionário num ciclone cilíndrico. O campo de escoamento é dividido em duas regiões : a primeira, central, inviscida onde são admitidos os perfís de velocidade axial e tangencial e a segunda, adjacente a parede lateral, onde estão contidos todos os efeitos viscosos. Assim o problema foi reduzido a solução de equações diferenciais totais. Este modelo é arbitrário pois adotou-se " a priori " os perfís de velocidade na região central e a comparação dos resultados previstos pelo modelo com os apresentados por Ustimenko e Bukhman [98] mostra sérias discrepâncias. Já o segundo trabalho visa a determinação do escoamento incompressível, inviscido e em regime estácionario num ciclone de corpo tronco-cônico. O método utilizado na solução é o das linhas de corrente e o problema apresenta solução exata. A aderência dos resultados obtidos por este modelo simulando as condições experimentais de Kelsall [45] é boa na região central do escoamento mas torna-se débil na próxima a parede.

A dificuldade de determinar ,por via analítica, os perfís de velocidade e pressão em câmaras ciclônicas incentivou Baluev e Troyankin [2] e [3] a utilizar a via experimental para este fim.Foram realizadas várias séries de testes,utilizando-se ar como fluído de trabalho,em 23 configurações diferentes de câmaras ciclônicas cilíndricas obtidas com a variação do número e posição dos bocais de alimentação e com a variação do diâmetro da garganta.Em todas as configurações o diâmetro da câmara é o mesmo e igual a Ø.18 m .As conclusões principais são que pequenas alterações no comprimento,número,forma e posição dos bocais de alimentação e no diâmetro da garganta afetam de forma significativa o escoamento na câmara e que os escoamentos turbulentos são similares numa mesma configuração de câmara.A partir destas conclusões são apresentadas várias correlações empíricas que foram testadas com relativo sucesso em ciclones com diâmetros que variam de Ø.44 a Ø.65m .O problema na utilização destes resultados é o excesso de constantes empíricas que foram ajustadas para as condições particulares dos casos estudados.Assim a validade das mesmas constantes para casos um pouco diferentes dos ensaidos é questionável.

Já Ustimenko e Bukhman [98] levantaram experimentalmente os perfís de velocidade,pressão,intensidade de turbulência (energia cinética da turbulência) e das correlalações turbulentas duplas para o escoamento de ar numa câmara ciclônica cilíndrica.O diâmetro desta é igual a Ø.25 m,comprimento de Ø.386 m e diâmetros de garganta iguais a Ø.1Ø e Ø.25 m.O ar é admitido por um bocal tangencial de área igual a $1.57 \times 1Ø^{-3}$ m^2

ou por quatro bocais idênticos , equidistantes e que apresentam a mesma área total daquela do bocal único. As principais conclusões que podem ser obtidas são :

-para o caso em que a alimentação é realizada por quatro bocais, as distribuições de velocidades, pressão e correlações turbulentas são axissimétricas,

-os escoamentos são similares nas varias condições ensaiadas com o número de Reynolds , baseado no diâmetro da câmara e velocidade axial média, variando de 5400 a 9900 ,

-os efeitos provocados pela presença da garganta , localização e número de bocais sobre o escoamento são marcantes.

Todas as medidas realizadas neste trabalho foram realizadas com anemômetros de fio quente e então contém o erro inerente ao processo de medida que , nas condições ensaiadas, não é desprezível segundo Holman e Moore [38] e Buchhave [15]. Para evitar o erro induzido pela presença da ponta de prova sobre o escoamento, Escudier et al. [22] utilizaram um anemômetro laser para a determinação dos perfís de velocidade numa câmara ciclônica operando com água como fluído de trabalho. Os resultados apresentados tem, sem dúvida , uma qualidade melhor do que a dos obtidos por Ustimenko e Bukhman mas se referem a poucas secções transversais da câmara, o que torna difícil utilizá-los como o caso padrão desta Tese.

Aparentemente o primeiro trabalho de simulação numérica de escoamento confinado com componente tangencial intensa, realizado a partir da integração das equações fundamentais é o de Crowe e Pratt [18].O método utilizado na modelagem e solução do problema é o de Gosman et al. [29], ou seja, as variáveis primitivas são a vorticidade e a linha de corrente e a discretização utilizada é a dos volumes finitos (na época era conhecida como tanque e tubo).O escoamento simulado é aquele que ocorre em ciclones com alimentação axial e " swirl " imposto por pás direcionadoras e com saída de gás "limpo" tambem axial.O campo de escoamento foi dividido em 121 volumes elementares e o modelo de turbulência utilizado é o de comprimento de mistura.No trabalho não é especificada a expressão para o comprimento de mistura e não são apresentados os perfís de velocidade e pressão obtidos na simulação.Os resultados apresentados se referem as trajetórias médias das partículas no escoamento e o comportamento da eficiência de coleta em função da concentração de particulado na secção de alimentação do equipamento.

Um outro trabalho na mesma linha é o de Kubo e Gouldin [5Ø] que apresenta a simulação do escoamento incompressível, isotérmico com componente tangencial moderada num tubo cilíndrico.O método utilizado tambem é o de Gosman et al. [29] ,mas utiliza-se para a avaliação da viscosidade efetiva o modelo de turbulência baseado nas equações de conservação de k e ε (energia cinética turbulenta e taxa de dissipação de energia cinética turbulenta) proposto por Jones e Launder [4Ø].Não é especificado o número de volumes elementares utilizados na dis-

cretização mas é reportado com muita cautela a escolha das condições de contorno, principalmente as de alimentação, que são críticas para o sucesso da simulação de escoamentos elípticos com com componente tangencial de velocidade. Neste trabalho tambem não consta uma comparação entre resultados experimentais e os dados obtidos por simulação para as mesmas condições de operação.

Busnaina e Lilley [16] utilizaram o método MAC ("Marker and Cell"), Harlow e Welch [34], para simular o escoamento em camâras ciclônicas. Uma descrição mais detalhada do método pode ser encontrada em Hirt [36] , uma implementação para casos axissimétricos, laminares e sem componente tangencial em Hirt et al. [35] e para casos tri-dimensionais turbulentos em Vastistas et al. [99] . Este método foi desenvolvido para resolver o conjunto de equações da continuidade e Navier-Stokes na forma transiente. As variáveis primitivas são as velocidades e a pressão, a discretização utilizada é a das diferenças finitas e a malha empregada é deslocada, ou seja, as velocidades são calculadas em pontos localizados entre aqueles onde são avaliados a pressão e outras variáveis como a velocidade tangencial, k e ε . Como as equações de transporte são parabólicas no tempo é possivel utilizar um procedimento que caminhe no tempo, começando de um estado arbitrário até atingir o regime estacionário. O grande problema do método, implementado por Hirt et al. [35], mas que garante sua simplicidade é a utilização da aproximação explícita para a derivada temporal, assim para qualquer variável Φ (u, v, w, k, ε) pode-se escrever :

$$\Phi^{n+1} = \Phi^{n} + \Delta t \times [---------]^{n}$$

onde

Δt intervalo de tempo

n número da iteração temporal

Todos os efeitos convectivos e difusivos estão contidos no termo entre colchetes no lado direito da equação. Como a discretização é explícita, deve-se tomar cuidado na escolha do intervalo de tempo para se garantir a estabilidade do processo de solução do problema. Não se conhece um método exato que determine o intervalo de tempo máximo que pode ser utilizado a partir do conjunto de equações não lineares e da escolha da malha, então sua escolha é baseada em critérios lineares e em experiências numéricas. Uma análise deste método revela similaridades muito grandes com o método SIMPLE desenvolvido por Patankar e Spalding [73] e o artifício de se conseguir uma solução em regime estacionário a partir de uma análise temporal pode ser visto, Patankar [74], como um procedimento de sub-relaxação. Busnaina e Lilley simularam as condições experimentais de Ustimenko e Bukhman [98], utilizando uma malha uniforme de 12×12, gradientes nulos para as velocidades tangenciais as paredes como condição de contorno, uma viscosidade isotrópica efetiva de valor não declarado e tambem mostram resultados de simulações obtidas em situações irreais, como por exemplo, o comportamento do escoamento em função do número de swirl. Para ciclones com entrada tangencial este número é fixo pois é definido como a relação entre o momento da quantidade de movimento angular do fluído que cruza a secção de entrada e a quantidade de movimento média do

fluído na secção da garganta multiplicada pelo raio da garganta. Nestas condições o número de swirl é constante, Syred e Beér [95], e função única da geometria do equipamento. Analisando os resultados relatados neste artigo nota-se que a recirculação existente no ciclone é muito mal descrita pelo procedimento de simulação adotado.

Khalil [46] tambem simulou a experiência de Ustimenko e Bukhman utilizando o método SIMPLE e o modelo de turbulência k-ε de Launder e Spalding [53] levemente modificado com as proposições de Pope [78]. No modelo k-ε original a viscosidade turbulenta é definida por:

$$\mu_{turb} = \rho C_{\mu} \frac{k^2}{\varepsilon}$$

onde

C_{μ}	constante e igual a Ø.Ø9 (para escoamentos com alto número de Reynolds)
k	energia cinética turbulenta
ε	taxa de dissipação da energia cinética turbulenta

No trabalho de Pope propõe-se que C_{μ} seja uma função da taxas locais de deformação e rotação o que altera substancialmente os perfís da viscosidade turbulenta no escoamento. É bom notar que estas alterações não são efetuadas unicamente pela alteração da definição da viscosidade turbulenta mas tambem pela presença de C_{μ} na equação de transporte da taxa de dissipação de energia cinética turbulenta. Na simulação de Khalil foi utilizada uma malha de 2Ø×2Ø e reportou-se que o aumento desta para 24×24 resultou numa varia-

ção máxima ,entre as duas soluções,menor que 3% . Os resultados apresentados são muito próximos dos obtidos por via experimental.

O primeiro trabalho do grupo de simulação de escoamentos reativos da Universidade de Sheffield (Boysan,Swithenbank, Ayers e Weber) em escoamentos turbulentos,confinados e com componente tangencial intensa (1980) está comentado e referenciado em Gupta [31] e trata da simulação do experimento de Ustimenko e Bukhman [98] .O método utilizado para a solução do escoamento é o SIMPLE,foi utilizada uma malha de 32x21 e o domínio de solução foi estentido para fora da garganta da câmara.Isto foi feito para diminuir a influência das condições de contôrno na descarga sobre o escoamento interno.São apresentados resultados para dois tipos de modelos de turbulência:

-k-ε usual de Launder e Spalding [53]

-k-ε com correção da equação de transporte da taxa de dissipação de energia cinética turbulenta baseano número de Richardson, definido por :

$$Ri = \frac{k^2}{\varepsilon^2} \frac{W}{r^2} \frac{\partial}{\partial r} (r W)$$

onde

Ri	número de Richardson
k	energia cinética turbulenta
ε	taxa de dissipação da energia cinética turbulenta
W	velocidade média turbulenta
r	coordenada radial

e utilizando uma viscosidade efetiva para a equação da velocidade tangencial menor do que aquela das outras equações.

Os resultados apresentados para o primeiro modelo são muito próximos daqueles que serão apresentados nesta Tese, ou seja, não é obtida a zona de recirculação central na câmara. Já o segundo modelo propicia a formação de uma pequena recirculação, mas a simulação continua grosseira. As mudanças realizadas sobre o modelo k-ε original são arbitrárias e feitas de modo a simular a anisotropia da viscosidade efetiva que já havia sido detectada por Lilley e Chigier [63].

Para melhorar a descrição do escoamento, Boysan e Swithenbank [8], simularam o mesmo caso mas utilizaram um modelo de turbulência que avalia as tensões de Reynolds de forma algébrica baseado nas hipóteses de Rodi [84]. A malha utilizada na simulação era não uniforme com 40×21 nós. O resultado da alteração do modelo de turbulência é significativo pois descreve-se razoavelmente bem a zona de recirculação central do escoamento. A principal crítica a este trabalho é que nos cálculos das tensões de Reynolds só foram levados em conta os termos que contém a derivada radial da velocidade tangencial média e a relação entre a velocidade média tangencial e o raio ($\partial W/\partial r$ e W/r). Alegou-se que estes termos apresentavam ordem de grandeza superior aos outros mas não se mostra o resultado de uma simulação com todos os termos decorrentes do modelo de turbulência. Aparentemente isto foi feito para diminuir o tempo de processamento e para garantir a estabilidade do processo de simulação. O mesmo algorítmo

computacional foi utilizado para simular o comportamento de um ciclone de alta eficiência, Boysan et al.[9] ,e a aderência dos resultados obtidos para o diâmetro de corte e curva de eficiência é razoável(erro ≃ 20%).Neste caso e em todos os tratados pelo grupo de Sheffield o escoamento bi-fásico (sólido-gás) é tratado como disperso.

O trabalho posterior de Boysan et al. [10] utilizou outro modelo de turbulência baseado no trabalho de Launder et al.[54] com as mesmas hipóteses de Rodi para torná-lo algébrico e incluiu um modelo bastante simples para a combustão de carvão para que fosse possível simular a operação de combustores ciclônicos. Não é mostrada nenhuma comparação com dados experimentais que só são feitas no trabalho posterior [11] onde são simuladas as condições experimentais de Barnhart e Laurendau [5].Devido a dificuldades experimentais ,esta última referência não apresenta os perfís de velocidades e concentrações no interior do gaseificador ciclônico,mas só resultados globais.Deste modo foram comparados sómente as concentrações de O_2 , CO_2 , CO e H_2O ,a temperatura da mistura na garganta do ciclone e a taxa de conversão de carbono e nada pode se concluir sobre o comportamento do modelo de turbulência e sua eficiência na simulação do escoamento.O erro erro relatado para a temperatura é de 25% e para a taxa de conversão é 10%.O último trabalho de Boysan et al.[12] mostra um refinamento na modelagem das reações de combustão e a simulação de uma das experiências de Hoy [39].Novamente são apresentados apenas comparações de macro valores que no global são melhores do que as do caso anterior.Analisando o conjunto de trabalhos,resta a dúvida

sobre o comportamento do modelo de turbulência de Launder et al.[54] ,acrescido das hipóteses de Rodi [84],na simulação de escoamento em câmaras ciclônicas.

Os trabalhos de Periclous e Rhodes [76],Davidson [2Ø] e de Karniven e Ahlstedt [44] mostram grande similaridade pois utilizam o mesmo modelo de comprimento de mistura para a avaliação da viscosidade turbulenta.O primeiro e o último são aplicações do programa de simulação de escoamentos PHOENIX,descrito em Markatos et al. [69],e utilizados,respectivamente, na obtenção do comportamento de hidro-ciclones e ciclones.O segundo mostra a simulação do escoamento em hidro-ciclones e utiliza o algorítimo SIMPLE.Os comprimentos de mistura adotados são proporcionais aos diâmetros da secção no corpo do ciclone e ao diâmetro do pescoço.A viscosidade turbulenta sugerida é dada por :

$$\mu_{turb} = \rho \lambda^2 \left| \frac{\partial W}{\partial r} - \frac{W}{r} \right| + C_{in}$$

e

$$\mu_{eff} = \mu + \mu_{turb}$$

onde

μ	viscosidade
μ_{turb}	viscosidade turbulenta
μ_{eff}	viscosidade efetiva
ρ	massa específica
λ	comprimento de mistura

W	velocidade tangencial média
r	coordenada radial
C_{in}	constante que representa a turbulência convectada para o equipamento

Mas as hipóteses " ad hoc " não se restringem a esta definição de viscosidade turbulenta, pois a equação de conservação de quantidade de movimento tangencial, na forma conservativa, utilizada é:

$$\frac{1}{r}\left[\frac{\partial}{\partial x}(\rho UrW) + \frac{\partial}{\partial r}(\rho VrW) - \frac{\partial}{\partial x}\left(r\mu_{eff}\frac{\partial W}{\partial x}\right) + \right.$$

$$\left. - \frac{\partial}{\partial r}\left(r\,\mu_{eff}\frac{\partial W}{\partial r}\right)\right] = \emptyset$$

Na forma correta, por exemplo Schilichting [85], o lado direito da equação deve ser igual a:

$$-\frac{\rho V W}{r} - \frac{W^2}{r}\frac{\partial}{\partial r}\left(r\,\mu_{eff}\right)$$

onde

x	coordenada axial
r	coordenada radial
U	velocidade média axial
V	velocidade média radial
W	velocidade média tangencial
μ_{eff}	viscosidade efetiva

lteração da forma desta equação de conservação de quantidade de imento é injustificada mas produz resultados satisfatórios nas ulações apresentadas, por exemplo: os resultados obtidos por Peleous e Rhodes são bons e muito mais próximos dos experimentais Kelsall [45] do que os de Bloor e Inghan [7]. A boa lidade dos resultados apresentados nestes trabalhos é conseguida que se introduziu, de maneira deselegante, uma anisotropia viscosidade efetiva que poderia ter sido feita utilizando-se a reção de viscosidade turbulenta , para a equação de conservação quantidade de movimento tangencial , proposta por Bradshaw e que descrita em Lilley [64].

Fu et al. [25] simularam três jatos axissimétri-, dois deles com "swirl" e apresentando zona central de recircu-ão. Um dos objetivos do trabalho é quantificar a influência da tese de Rodi, que torna o modelo diferencial de transporte das ões de Reynolds em um modelo algébrico, nos resultados das ulações. Para isto foi utilizado o mesmo programa de computador ado em volumes finitos e o modelo de turbulência de Launder et 54]. Reporta-se que para escoamentos internos, onde os perfis nergia cinética turbulenta são praticamente impostos pela ite-o fluído - parede, o comportamento dos resultados dos dois mo-s é muito próximo o que justifica a adoção da hipótese de .O mesmo não se pode dizer para os escoamentos livres onde os s, gerados pela adoção da hipótese, podem ser relativamente des.

O escoamento de um jato central envolto por um jato ar com componente tangencial de velocidade e confinados num

tubo cilíndrico foi simulado por Jones e Pascau [42].Utilizou-se para este fim um programa de computador baseado em volumes finitos e dois modelos de turbulência para o fechamento do problema.O primeiro foi um k-ε clássico, Launder e Spalding [53],e o segundo foi um modelo diferencial de transporte das tensões de Reynolds calcado no trabalho de Jones e Musonge [41] que é próximo àquele derivado em Launder et al.[54]. Os resultados apresentados mostram novamente a inabilidade do primeiro modelo de turbulência na simulação de escoamentos complexos e que os resultados propiciados pelo segundo são bastante próximos dos obtidos por via experimental.No artigo tambem é mostrado o quanto é infrutífera a alteração da equação clássica de transporte de ε ,por exemplo [54],para a de Bardina et al. [4] que inclui um termo adicional para corrigir a influência da rotação sobre a transferência de energia das grandes escalas para as pequenas.

IV MÉTODO UTILIZADO PARA A PREDIÇÃO DO ESCOAMENTO E MODELOS DE TURBULÊNCIA

4.1 Preâmbulo

Atualmente existem , para as equações diferenciais relevantes à mecânica dos fluídos, vários métodos de discretização e solução das equações algébricas geradas neste processo. O método de discretização adotado neste trabalho é o dos volumes finitos e o utilizado na solução do escoamento é o implícito da família SIMPLE . Uma descrição detalhada dos dois métodos pode ser encontrada em Patankar [74], e esta escolha foi determinada pelos seguintes aspectos:

- a discretização é conservativa;
- o algorítmo explícito de Hirt et [35] é ineficiente em computadores com arquitetura não paralela devido a problemas de estabilidade numérica que podem ser vistos em Nogotov [72] e no tralho de Sini e Dekeyser [89];
- sua implementação é mais simples, exige menos memória de computador e apresenta maior velocidade de convergência , Chabard e Violet [17], que aquela referente a discretização por elementos finitos.

Um outro método para a predição do escoamento é a simulação direta, discutida extensamente por Givi [28], mas esta abordagem foi descartada por exigir um esforço computacional incompatível com o equipamento disponível que é um micro-computador do tipo IBM -PC.

4.2 Discretização das Equações

A discretização das equações relevantes ao problema será realizada, sem perda de generalidade, admitindo-se que os efeitos da turbulência sobre o escoamento podem ser modelados de uma maneira difusiva, ou seja, todo o transporte devido a turbulência é igual ao produto de um coeficiente de difusão pelo gradiente do potencial relativo ao transporte. Para exemplificar, pode-se aplicar o modelo de Boussinesq, Launder e Spalding [52], que define a viscosidade turbulenta a partir dos tensores de Reynolds e das taxas de deformação como

$$\rho \langle u_i \, u_j \rangle = - \mu_{turb} \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right)$$

na equação de conservação da quantidade de movimento(Eq. 7). Assim pode-se obter:

$$\frac{\partial}{\partial x_j} (\rho U_i U_j) = - \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \left(\mu_{eff} \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) \right)$$

Utilizando o mesmo argumento para todas as equações relevantes, pode-se mostrar que estas apresentam uma forma comum,

dada por:

$$\frac{\partial}{\partial x_j}\left(\rho U_j \Phi - \Gamma_\Phi \frac{\partial \Phi}{\partial x_j}\right) = S_\Phi \qquad (9)$$

onde

Φ	propriedade intensiva
Γ_Φ	coeficiente de difusão de Φ
S_Φ	termo fonte de Φ

Convertendo a Eq. 9 para o sistema de coordenadas cilíndrico e admitindo que o termo fonte S_Φ possa ser linearizado na forma $S_P \Phi + S_C$, tem-se:

$$\frac{1}{r}\left[\frac{\partial}{\partial x}\left(r\rho U\Phi - r\Gamma_\Phi \frac{\partial \Phi}{\partial x}\right) + \frac{\partial}{\partial r}\left(r\rho V\Phi - r\Gamma_\Phi \frac{\partial \Phi}{\partial r}\right)\right] = S_p \Phi + S_c \qquad (10)$$

onde

x	coordenada axial
r	coordenada radial
U	velocidade média axial
V	velocidade média radial

Integrando esta última equação para um volume de controle típico ,como o mostrado na Fig.2 ,utilizando o Teorema de Gauss e admitindo que o valor de Φ que prevalece na região em questão é Φ_P , obtem-se :

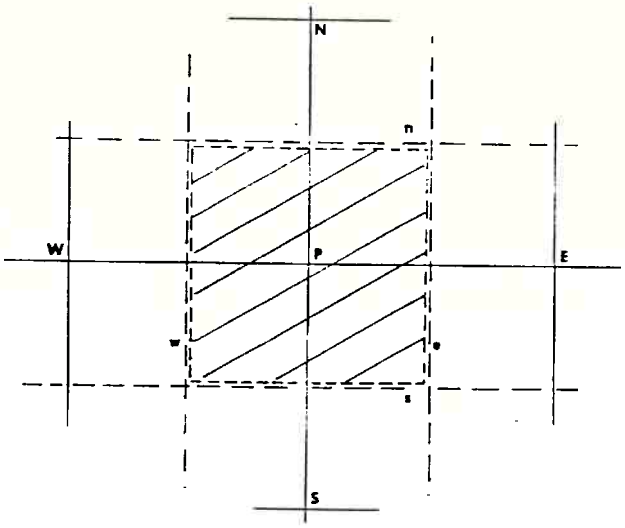


Fig.2 Volume de Controle Típico

$$\left[\rho V \Phi - \Gamma_\Phi \frac{\partial \Phi}{\partial r} \right]_n A_n - \left[\rho V \Phi - \Gamma_\Phi \frac{\partial \Phi}{\partial r} \right]_s A_s +$$

$$\left[\rho U \Phi - \Gamma_\Phi \frac{\partial \Phi}{\partial x} \right]_e A_e - \left[\rho U \Phi - \Gamma_\Phi \frac{\partial \Phi}{\partial x} \right]_w A_w =$$

$$(S_p \Phi_p + S_c) \Delta V \qquad (11)$$

onde

A_n área superior do volume de controle

A_s área inferior do volume de controle

A_w e A_e áreas laterais do volume de controle

ΔV volume do elemento

$[\ ]_i$ indicam que o valor da função deve ser avaliado na superfície i

A equação da continuidade pode ser obtida fazendo-se $\Phi = 1$ e $S_\Phi = \emptyset$ na Eq. 9 . Utilizando os mesmos argumentos para a obtenção da Eq.11, tem-se :

$$\left[\rho V A\right]_n - \left[\rho V A\right]_s + \left[\rho U A\right]_e - \left[\rho U A\right]_w = \emptyset \quad (12)$$

Para transformar a Eq. 11 numa equação algébrica é necessária a utilização de uma função de interpolação para a variável dependente. A escolha desta função é fundamental para a eficiência e fidedignidade do processo de solução. Uma das formulações mais utilizadas para a interpolação da variável dependente na fronteira do volume de controle é a híbrida, Spalding [92], que é constituida pela combinação da aproximação a montante (" upwind ") para o termo convectivo e de diferença central para a termo difusivo.

Por exemplo, a aproximação híbrida para o colchete referente a superfície esquerda do volume de controle , admitindo que as distâncias entre a superfície e os pontos onde se avaliam Φ_P e Φ_W sejam iguais é:

$$(\rho U)_w \left(\Phi_W + \Phi_P \right)/2 - \Gamma_{\Phi w} \left(\Phi_P - \Phi_W \right)/\delta x_w \quad \text{para} \quad | Pe_w | < 2$$

$$(\rho U)_w \Phi_W \quad \text{para} \quad Pe_w \geq 2$$

$$(\rho U)_w \Phi_P \quad \text{para} \quad Pe_w \leq -2$$

onde

Pe_w número de Peclet para a superfície w e definido por $(\rho U_w)\ \delta x_w \ / \ \Gamma_{\Phi\ w}$

δx_w distância entre os nós W e P

Deste modo a Eq. 11 pode ser transformada numa equação algébrica que envolve as variáveis dependentes representativas dos volumes

de controles vizínhos,ou seja :

$$a_N \Phi_N + a_S \Phi_S + a_E \Phi_E + a_W \Phi_W + S_c \Delta V = a_P \Phi_P \qquad (13)$$

$$\text{com} \quad a_P = \sum_{viz} a_{viz} - S_P \Delta V$$

4.3 Método Utilizado na Predição do Escoamento

Com as equações pertinentes na forma algébrica é possível utilizar um procedimento iterativo,pois o problema é não linear,para a solução do escoamento. Neste trabalho utilizaram-se dois procedimentos para este fim;o primeiro é o SIMPLE,descrito por Patankar e Spalding [73] e mais extensamente por Patankar [74], e o segundo conhecido como SIMPLEC ,que é uma pequena varição do primeiro, e foi proposto por Van Doormaal e Raithby [21].

De uma maneira suscinta,o procedimento SIMPLE consiste em utilizar malhas defasadas onde as velocidades ,axial e radial são avaliadas em pontos localizados entre aqueles onde são calculadas a pressão ,velocidade tangencial,energia cinética turbulenta e sua taxa de dissipação,a localização escolhida está mostrada na Fig. 3, e a seguinte sequência de cálculos:

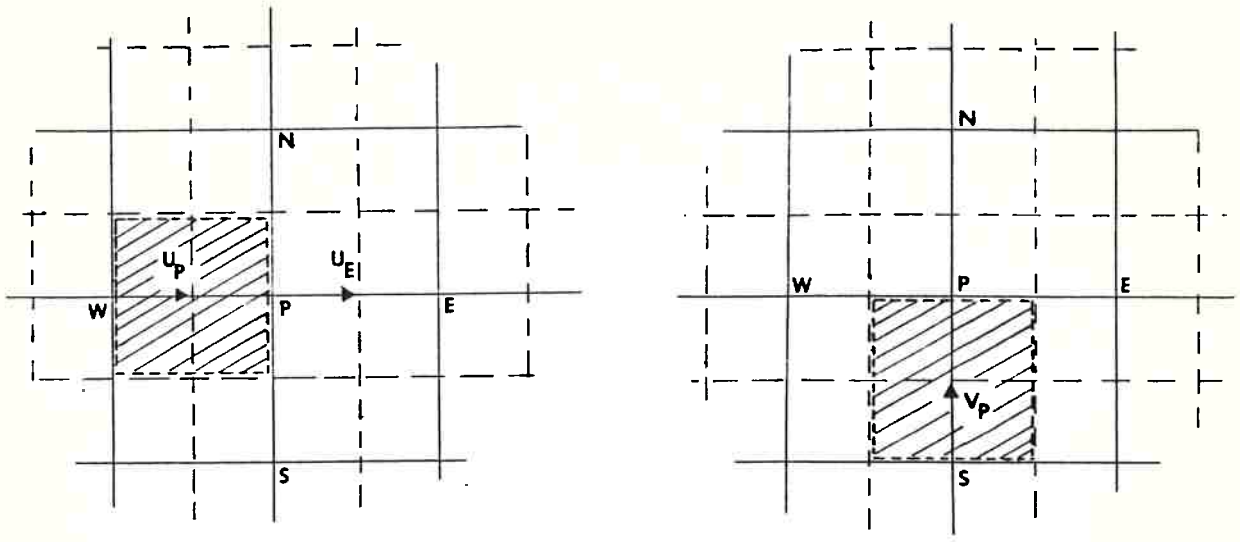


Fig.3 Volumes de Controle Para U e V

- admitir o campo de pressão p^*,
- resolver os sistemas algébricos obtidos a partir das equações de conservação da quantidade de movimento obtendo os campos de velocidade intermediários U^*, V^* e W^*,
- Calcular a correção de pressão p' utilizando a equação conhecida como a de Poisson [65],
- Calcular o novo campo de pressão somando p' a p^*,
- Calcular as novas velocidades U,V,W a partir das antigas e da correção do campo de pressão,
- Resolver os sistemas algébricos relativos aos outros Φ 's, como por exemplo, para a energia ci-

nética turbulenta e para sua taxa de dissipação,

- Admitir que o novo campo de pressão seja o campo de pressão p^* e voltar ao segundo passo do procedimento até que a convergência seja atingida.

A única diferença entre os dois procedimentos é o tratamento dado ao acoplamento pressão- velocidade. A correção do campo de velocidades (quinto passo do procedimento) é realizada de maneira ' unidimensional ', por exemplo : a correção da velocidade U_e^*, mostrada na Fig.3 , é realizada pela relação :

$$U_e = U_e^* + d_e \left(p_P' - p_E' \right)$$

onde

d_e coeficiente relativo ao ajuste de pressão

No procedimento SIMPLE d_e é obtido a partir de uma aproximação rudimentar das equações de conservação da quantidade de movimento e é igual a relação entre a área da face direita do volume de controle e o coeficiente a_E da equação algébrica referente a esta equação de conservação. No procedimento SIMPLEC a aproximação realizada é mais consistente e a expressão de d_e torna-se igual a :

$$d_e = \frac{A_e}{a_e - \sum a_{viz}}$$

onde

A_e Área lateral direita do volume de controle

a_e coeficiente da equação algébrica para a quantidade de movimento na direção x

$\sum a_{viz}$ somátoria dos coeficientes da equação algébrica para a quantidade de movimento na direção x, excluindo-se a_e

Como o procedimento resolve um problema não linear a partir de uma sequência de problemas lineares torna-se necessária a utilização da sub-relaxação, ou seja, o campo de qualquer variável Φ , obtido por solução do seu sistema de equações algébrico, deve ser substituído por :

$$\alpha \Phi^{novo} + (1 - \alpha) \Phi^{velho}$$

onde

α coeficiente de sub-relaxação ($\emptyset < \alpha < 1$)

Φ^{novo} valores de Φ recém-obtidos

Φ^{velho} valores de Φ no ciclo anterior

Não existe um modo de se determinar ," a priori ", os valores de α de modo que o procedimento apresente convergência e o único método para a sua determinação é o experimento numérico.

4.4 Modelo de Turbulência k - ε Básico

O modelo de turbulência k - ε é um dos membros da família dos modelos baseados em duas equações diferenciais, Launder e Spalding [52], que tenta descrever o fenômeno em função de uma única velocidade e de uma única escala turbulentas. Neste modelo

admite-se que a velocidade turbulenta pode ser caracterizada pela raíz quadrada da energia cinética turbulenta ($k = < u_i u_i >/2$) e a escala da turbulência por :

$$\varepsilon = \frac{k^{3/2}}{\ell}$$

onde

k	energia cinética tubulenta
ε	taxa de dissipação de k
ℓ	escala da turbulência

que foi obtida para turbulência isotrópica que apresenta espectro de energia no equilibrio.

A equação de conservação da energia cinética turbulen foi obtida através da contração da equação de transporte das tens de Reynolds, utilizando-se a definição ampliada de Boussinesq para a viscosidade turbulenta ,

$$\rho < u_i\ u_j > = - \mu_{turb} \left[\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right] + \frac{2}{3} k \delta_{ij} \quad (14$$

e admitindo que a difusão de energia turbulenta pode ser modelada como difusiva, ou seja :

$$U_i \frac{\partial k}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\frac{\Gamma_k}{\rho} \frac{\partial k}{\partial x_i} \right) + P - \varepsilon \qquad (15)$$

onde

Γ_k coeficiente para difusão de k

P geração de energia cinética turbulenta, dada por :

$$P = - \frac{\mu_{eff}}{\rho} \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) \frac{\partial U_j}{\partial x_i}$$

e $\mu_{eff} = \mu + \mu_{turb}$

A taxa de dissipação da energia cinética turbulenta, tomada como o traço do tensor de dissipação, que é definido por:

$$\varepsilon_{ij} = - \frac{2\mu}{\rho} < \left(\frac{\partial u_i}{\partial x_l} \frac{\partial u_j}{\partial x_l} \right) >$$

apresenta uma equação de transporte obtida a partir das equações de conservação da quantidade de movimento e de transporte das tensões de Reynolds, por exemplo a versão de Daly e Harlow [19], mas esta envolve termos de difícil interpretação e modelagem. A versão da equação normalmente utilizada é a de Hanjalic e Launder [33] e foi desenvolvida para escoamentos que apresentam alto número de Reynolds. Os dois utilizaram para a simplificação da equação a definição de viscosidade turbulenta de Boussinesq, Eq. 14 , a análise dimensional , admitiram que a dissipação se dá de maneira isotrópica e que a difusão de ε pode ser modelada como difusiva. O próprio Launder [58] reconhece que sem uma certa dose de criatividade é impossível obter a equação normalmente utilizada e que todas as hipóteses simplificadoras são passíveis de

questionamento. Aparentemente o maior problema na derivação da equação é que ela utiliza apenas uma escala para descrever o processo local de dissipação de energia o que contraria vários resultados experimentais, por exemplo os de Browne referenciados por Taulbee [96]. A forma da equação de transporte de ε é :

$$U_i \frac{\partial \varepsilon}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\frac{\Gamma_\varepsilon}{\rho} \frac{\partial \varepsilon}{\partial x_i} \right) + C_{\varepsilon 1} P \frac{\varepsilon}{k} + C_{\varepsilon 2} \frac{\varepsilon^2}{k} \qquad (16)$$

onde

Γ_ε coeficiente de difusão para ε

$C_{\varepsilon 1}$ e $C_{\varepsilon 2}$ constantes empíricas

Neste modelo a viscosidade turbulenta é isotrópica e definida do sequinte modo:

$$\mu_{turb} = C_\mu \rho \frac{k^2}{\varepsilon} \qquad (17)$$

onde

C_μ constante

e os coeficientes de difusão, Γ_k e Γ_ε , são relacionados com a viscosidade através de equivalentes ao número de Prandt, ou seja :

$$\Gamma_k = \frac{\mu}{Pr_k} \qquad e \qquad \Gamma_\varepsilon = \frac{\mu}{Pr_\varepsilon} \qquad (18)$$

A utilização das equações 15,16,17 e 18 torna possível a solução do problema composto pelas equações de conservação da quantidade de movimento e da continuidade, Eqs.7 e 8, mas ainda é necessário estabelecer valores para as constantes presentes no modelo de turbulência. Estas foram ajustadas, utilizando-se experimentos numéricos, para escoamentos simples, no sentido de Bradshaw [13] e Lakshminarayana [51], e os valores clássicos , Markatos [7Ø], são :

$C_\mu = Ø.Ø9$

$C_{\varepsilon 1} = 1.44$

$C_{\varepsilon 2} = 1.92$

$Pr_k = 1.Ø$

$Pr_\varepsilon = 1.3$

4.5 Modelos de Turbulência k - ε Modificados

O modelo k - ε básico proporciona resultados muito bons em escoamentos simples como pode ser visto em Singhal e Spalding [88] e em escoamentos mais complexos como o jato axis-

simétrico livre , com velocidade tangencial pequena, reportado por Leschziner e Rodi [62]. Já as aplicações do mesmo modelo a jatos confinados com velocidades tangenciais moderadas, descritas por Srinivasan e Mongia [93] , levaram a resultados muito diferentes daqueles obtidos experimentalmente nas mesmas condições. As principais deficiências reportadas foram a má descrição da zona central de recirculação e nos perfís de energia cinética turbulenta.

Como existem incertezas na equação de transporte da taxa de dissipação de energia cinética turbulenta e do valor de C_μ várias tentativas foram feitas para melhorar o desempenho do modelo básico em situações onde a velocidade tangencial apresenta a mesma ordem de grandeza da axial ou radial. As modificações se resumem a alterar a forma da equação de transporte de ε e na substituição da constante C_μ por uma função que é dependente das condições locais do escoamento.

Uma das modificações mais utilizadas é a proposta por Launder, Pridden e Sharma [55] que tornaram a equação de ε sensível ao perfil da velocidade tangencial através do número de Richardson, definido por :

$$Ri = \frac{2\ W/r\ (\partial W/\partial r + W/r\ )}{(\partial U/\partial r)^2 + [\ r(\partial/\partial r)(W/r)]^2}$$

e alterando o lado direito da Eq. 16 para :

$$C_{\varepsilon 1} P \frac{\varepsilon}{k} - C_{\varepsilon 2} (1 - C_{gs} Ri) \frac{\varepsilon^2}{k}$$

onde

C_{gs} constante (aproximadamente Ø.ØØ2)

Esta alteração " ad hoc ",para um perfil de momento da quantidade de movimento com gradiente positivo, tenderá a produzir um número de Richardson positivo que aumentará a taxa de dissipação ,provocando então ,o decréscimo da viscosidade turbulenta.O efeito do número de Richarson sobre a viscosidade turbulenta é coerente com o critério de estabilidade de Von Karman citado por Sloan et al. [9Ø].

O outro modo de se modificar o comportamento do modelo é o utilizado por Pourahmadi e Humphrey [8Ø] que trocaram o valor da constante empírica C_μ por uma função que é obtida a partir de um modelo algébrico das tensões de Reynolds. Este procedimento é muito particular pois para a determinação da função que substitui C_μ são necessárias hipóteses simplificadoras que são dependentes do caso em questão.No caso de simulação do escoamento em câmaras ciclônicas a modificação de C_μ proposta por Boysan et al. [8] é:

$$C_\mu = \frac{[\frac{2}{3} (1 - \alpha - \beta) (\alpha + \beta) - \gamma] \frac{\varepsilon}{P}}{1 - \frac{k^2}{P^2} (\alpha \frac{\partial W}{\partial r} - \beta \frac{W}{r}) (\beta \frac{\partial W}{\partial r} - \alpha \frac{W}{r})}$$

onde α, β e γ são funções da geração de energia cinética turbulenta (P) e da taxa de dissipação de energia cinética turbulenta (ε).

Por mais que se modifique o modelo k - ε este ainda continua se baseando na hipótese de Boussinesq o que implica em viscosidades isotrópicas, assim a anisotropia das tensões de Reynolds é provocada pela falta de isotropia das deformações principais. Sabe-se que isto não é verdade para escoamentos complexos, por exemplo os resultados de Lilley e Chigier [63] e os de Scott e Rask [86]. Desta maneira não se pode esperar que o modelo se comporte, em escoamentos complexos, do mesmo modo que nos simples.

4.6 Modelo Algébrico das Tensões de Reynolds

Um dos modos de se abandonar a hipótese de Boussinesq , mantendo o tempo de processamento da simulação do escoamento em níveis compatíveis com um micro-computador é o modelo algébrico das tensões de Reynolds. Este modelo é baseado no truncamento da equação diferencial de transporte das tensões de Reynolds e nas equações de transporte de k e ε.

A equação de transporte das tensões de Reynolds pode ser determinada, por exemplo em Bradshaw [14], diretamente da equação de conservação da quantidade de movimento. Esta pode ser escrita, no regime estacionário para um fluído incompressível , da sequinte forma:

$$U_l \frac{\partial \langle u_i u_j \rangle}{\partial x_l} = P_{ij} + \Pi_{ij} + D_{ij} - \varepsilon_{ij} \qquad (19)$$

onde

P_{ij} tensor geração de turbulência

$$P_{ij} = - \left(\langle u_i u_l \rangle \frac{\partial U_j}{\partial x_l} + \langle u_j u_l \rangle \frac{\partial U_i}{\partial x_l} \right)$$

Π_{ij} tensor de redistribuição pressão - tensão

$$\Pi_{ij} = \left\langle \frac{p'}{\rho} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right\rangle$$

p' é a flutuação da pressão

D_{ij} tensor de difusão

$$D_{ij} = - \frac{\partial}{\partial x_l} \left(\langle u_i u_j u_l \rangle \right) - \frac{1}{\rho} \left(\frac{\partial}{\partial x_i} \left(\langle p' u_j \rangle \right) + \right.$$

$$\left. + \frac{\partial}{\partial x_j} \left(\langle p' u_i \rangle \right) \right) + \frac{\mu}{\rho} \frac{\partial \langle u_i u_j \rangle}{\partial x_l}$$

ε_{ij} tensor dissipação

$$\varepsilon_{ij} = \frac{2\,\mu}{\rho} \left(< \frac{\partial u_i}{\partial x_l} \quad \frac{\partial u_j}{\partial x_l} > \right)$$

Propõe-se para a simplificação da Eq. 19 as sequintes hipóteses:

- O termo difusivo, D_{ij} é modelado como no trabalho de Daly e Harlow [19] ;

$$D_{ij} = C_{\varepsilon 3} \frac{\partial}{\partial x_k} \left(\frac{k}{\varepsilon} < u_k\, u_l > \frac{\partial < u_i\, u_j >}{\partial \quad x_l} \right)$$

onde $C_{\varepsilon 3}$ é uma constante empírica.

- A dissipação é admitida isotrópica para escoamentos com alto número de Reynolds;

$$\varepsilon_{ij} = \frac{2}{3}\, \delta_{ij}\, \varepsilon$$

- O tensor de redistribuição é simplificado da mesma maneira daquela apresentada no modelo II de Launder et al.[54],para regiões distantes de paredes , e modificado pelos resultados de Gibson e Launder [26] e [27] ;

$$\Pi_{ij} = - C_1 \frac{\varepsilon}{k} \left(< u_i u_j > - \frac{2}{3} \delta_{ij} k \right) +$$

$$- C_2 \left(P_{ij} - \frac{1}{3} \delta_{ij} P_{kk} \right)$$

onde C_1 e C_2 são constantes empíricas

- As equações de transporte de k (redundante se for utilizado o modelo diferencial de transporte das tensões de Reynolds) e de ε mantém a mesma forma das utilizadas no modelo k - ε mas o termo difusivo é alterado para a versão aconselhada por Daly e Harlow [19].

  A equação de transporte de k , para as mesmas condições da Eq. 19 é :

$$U_l \frac{\partial k}{\partial x_l} = C_k \frac{\partial}{\partial x_l} \left(\frac{k}{\varepsilon} < u_l u_m > \frac{\partial k}{\partial x_m} \right) + P - \varepsilon \qquad (2\emptyset)$$

onde

$$P = - < u_i u_l > \frac{\partial U_i}{\partial x_l}$$

e C_k é uma constante empírica.

A equação de transporte de ε nas mesmas condições é:

$$U_l \frac{\partial \varepsilon}{\partial x_l} = C_{\varepsilon 3} \frac{\partial}{\partial x_l} \left(\frac{k}{\varepsilon} < u_l u_m > \frac{\partial \varepsilon}{\partial x_m} \right) + C_{\varepsilon 1} P \frac{k}{\varepsilon} +$$

$$- C_{\varepsilon 2} \frac{\varepsilon^2}{k} \qquad (21)$$

onde

$C_{\varepsilon 1}$, $C_{\varepsilon 2}$ e $C_{\varepsilon 3}$ são constantes empíricas.

Existem dois modos de transformar os modelos diferenciais de transporte das tensões de Reynolds em modelos algébricos, um devido a Rodi [84] e outro devido a Launder [57]. Após vários experimentos numéricos em simulações de escoamentos confinados com componente tangencial de velocidade verificou-se que os resultados das duas transformações são muito próximos e o próprio Launder reconhece o fato em Fu et al. [25]. Escolheu-se então a aproximação de Rodi que é baseada na suposição que $< u_i\ u_j > / k$ é constante no escoamento. Utilizando esta proposição na Eq. 19. , modificada pelas proposições anteriores, obtem-se :

$$< u_i\ u_j > = k \left[\frac{2}{3} k + \frac{1 - C_2}{C_1} \frac{\frac{P_{ij}}{\varepsilon} - \frac{2}{3} \delta_{ij} \frac{P}{\varepsilon}}{1 + \frac{1}{C_1} \left(\frac{P}{\varepsilon} - 1 \right)} \right]$$

onde C_1 e C_2 são constantes empíricas

A aproximação de Rodi prejudica a representação do transporte das tensões de Reynolds , Lumley et al. [68], então seus resultados serão mais precisos em escoamentos onde prevaleça o equilíbrio. No sistema de coordenadas cilíndrico o conjunto de Eqs. 19, 20 e 21

submetidas a aproximacão de Rodi tem a sequinte forma :

Equação para < uu >

$$\left[1 + \frac{4}{3} \frac{\lambda k}{\varepsilon} \frac{\partial U}{\partial x} \right] \langle uu \rangle = \frac{2k}{3} \left[1 + \frac{\lambda}{\varepsilon} \left[\frac{\partial V}{\partial r} \langle vv \rangle + \right. \right.$$

$$+ \frac{v}{r} \langle ww \rangle + \left(\frac{\partial V}{\partial x} - 2 \frac{\partial U}{\partial r} \right) \langle uv \rangle + \frac{\partial W}{\partial x} \langle uw \rangle +$$

$$\left. \left. + \left(\frac{\partial W}{\partial r} - \frac{W}{r} \right) \langle vw \rangle \right] \right] \qquad (22)$$

onde

$$\lambda = \frac{1 - C_2}{C_1 - 1 + \frac{P}{\varepsilon}}$$

Equação para < vv >

$$\left[1 + \frac{4}{3} \frac{\lambda k}{\varepsilon} \frac{\partial V}{\partial r} \right] \langle vv \rangle = \frac{2k}{3} \left[1 + \frac{\lambda}{\varepsilon} \left[\frac{\partial U}{\partial x} \langle uu \rangle + \right. \right.$$

$$+ \frac{V}{r} \langle ww \rangle + \left(\frac{\partial U}{\partial r} - 2 \frac{\partial V}{\partial x} \right) \langle uv \rangle + \frac{\partial W}{\partial x} \langle uw \rangle +$$

$$\left. \left. + \left(\frac{\partial W}{\partial r} + 2 \frac{W}{r} \right) \langle vw \rangle \right] \right] \qquad (23)$$

Equação para < ww >

$$\left[1 + \frac{4}{3} \frac{\lambda k}{\varepsilon} \frac{V}{r} \right] \langle ww \rangle = \frac{2k}{3} \left[1 + \frac{\lambda}{\varepsilon} \left[\frac{\partial U}{\partial x} \langle uu \rangle + \right. \right.$$

$$+ \frac{\partial V}{\partial r} \langle vv \rangle + \left(\frac{\partial V}{\partial x} + \frac{\partial U}{\partial r} \right) \langle uv \rangle - \left(2 \frac{\partial W}{\partial r} + \frac{W}{r} \right) \langle vw \rangle +$$

$$\left. \left. - 2 \frac{\partial W}{\partial x} \langle uw \rangle \right] \right] \qquad (24)$$

Equação para < uv >

$$\left[1 - \frac{\lambda k}{\varepsilon} \frac{V}{r} \right] \langle uv \rangle = \frac{\lambda k}{\varepsilon} \left[- \frac{\partial V}{\partial x} \langle uu \rangle - \frac{\partial U}{\partial r} \langle vv \rangle + \right.$$

$$\left. + \frac{W}{r} \langle uw \rangle \right] \qquad (25)$$

Equação para < uw >

$$\left[1 - \frac{\lambda k}{\varepsilon} \frac{\partial V}{\partial r} \right] \langle uw \rangle = \frac{\lambda k}{\varepsilon} \left[- \frac{\partial W}{\partial x} \langle uu \rangle - \frac{\partial W}{\partial r} \langle uv \rangle \right.$$

$$\left. - \frac{\partial U}{\partial r} \langle vw \rangle \right] \qquad (26)$$

Equação para < vw >

$$\left[1 - \frac{\lambda\, k}{\varepsilon} \frac{\partial U}{\partial x} \right] \langle vw \rangle = \frac{\lambda\, k}{\varepsilon} \left[- \frac{\partial W}{\partial r} \langle vv \rangle + \frac{W}{r} \langle ww \rangle + \right.$$

$$\left. - \frac{\partial W}{\partial x} \langle uv \rangle - \frac{\partial V}{\partial x} \langle uw \rangle \right] \qquad (27)$$

Equação para a energia cinética turbulenta (k)

$$\frac{1}{r} \left[\frac{\partial}{\partial x} \left[\rho r U k - \frac{\partial}{\partial x} \left(C_k \rho \frac{k}{\varepsilon} \langle uu \rangle + \frac{\mu}{Pr_k} \right) r \frac{\partial k}{\partial x} \right] + \right.$$

$$\left. + \frac{\partial}{\partial r} \left[\rho r V k - \frac{\partial}{\partial r} \left(C_k \rho \frac{k}{\varepsilon} \langle vv \rangle + \frac{\mu}{Pr_k} \right) r \frac{\partial k}{\partial r} \right] \right] =$$

$$= \rho P - \rho \varepsilon + \frac{\partial}{\partial x} \left[\rho C_k \frac{k}{\varepsilon} \langle uv \rangle \frac{\partial k}{\partial r} \right] +$$

$$+ \frac{1}{r} \frac{\partial}{\partial r} \left[r \rho C_k \frac{k}{\varepsilon} \langle uv \rangle \frac{\partial k}{\partial x} \right]$$

onde

$$P = - \left[\langle uu \rangle \frac{\partial U}{\partial x} + \langle vv \rangle \frac{\partial V}{\partial r} + \langle ww \rangle \frac{V}{r} + \langle uv \rangle \left(\frac{\partial U}{\partial r} + \right. \right.$$

$$\left. \left. + \frac{\partial V}{\partial x} \right) + \langle uw \rangle \frac{\partial W}{\partial x} + \langle vw \rangle \left(r \frac{\partial}{\partial r} \left(\frac{W}{r} \right) \right) \right] \qquad (28)$$

Equação para a taxa de dissipação (ε)

$$\frac{1}{r}\left[\frac{\partial}{\partial x}\left[\rho r U\varepsilon - \frac{\partial}{\partial x}\left(C_{\varepsilon 3}\,\rho\,\frac{k}{\varepsilon}\,\langle uu\rangle + \frac{\mu}{Pr_{\varepsilon}}\right) r\,\frac{\partial \varepsilon}{\partial x}\right] + \right.$$

$$\left. + \frac{\partial}{\partial r}\left[\rho r V\varepsilon - \frac{\partial}{\partial r}\left(C_{\varepsilon 3}\,\rho\,\frac{k}{\varepsilon}\,\langle vv\rangle + \frac{\mu}{Pr_{\varepsilon}}\right) r\,\frac{\partial \varepsilon}{\partial r}\right]\right] =$$

$$= \left(C_{\varepsilon 1}P - C_{\varepsilon 2}\varepsilon\right)\frac{\rho\,\varepsilon}{k} + \frac{\partial}{\partial x}\left[\rho\,C_{\varepsilon 3}\,\frac{k}{\varepsilon}\,\langle uv\rangle\,\frac{\partial \varepsilon}{\partial r}\right] +$$

$$+ \frac{1}{r}\,\frac{\partial}{\partial r}\left[r\rho\,C_{\varepsilon 3}\,\frac{k}{\varepsilon}\,\langle uv\rangle\,\frac{\partial \varepsilon}{\partial x}\right] \qquad (29)$$

Para que a Eq. 7 ,conservação da quantidade de movimento, fique coerente com o modelo de turbulência, basta substituir os termos que envolvem $\langle u_i u_j \rangle$ pelas expressões algébricas correspondentes. Além disto é preciso rearranjar os termos das equações de conservação da quantidade de movimento para que estas apresentem a forma "canônica" mostrada na Eq. 9.A sequir estão escritas as equações de conservação que são muito próximas ,e mais coerentes, do que aquelas apresentadas por Sloan et al.[90].

Equação de conservação da quantidade de movimento na direção x (componente axial , Eq. 30)

$$\frac{1}{r}\left[\frac{\partial}{\partial x}\left[\rho r U U - \frac{\partial}{\partial x}\left(2\,\lambda\,\rho\,\frac{k}{\varepsilon}\,\langle uu\rangle + \mu\right) r\,\frac{\partial U}{\partial x}\right] +\right.$$

$$+ \frac{\partial}{\partial r}\left[\rho r V U - \frac{\partial}{\partial r}\left(\frac{\lambda \rho}{\gamma_{21}}\frac{k}{\varepsilon}\langle vv\rangle + \mu\right) r \frac{\partial U}{\partial r}\right] =$$

$$= -\frac{\partial p}{\partial x} + \mu \frac{\partial}{\partial x}\left(\frac{\partial U}{\partial x}\right) + \frac{\mu}{r}\frac{\partial}{\partial r}\left(\frac{\partial V}{\partial x}\right) + S_{11}^{U} + S_{21}^{U}$$

onde

$$\lambda = \frac{1 - C_2}{C_1 - 1 + \frac{P}{\varepsilon}}$$

$$S_{11}^{U} = \frac{\partial}{\partial x}\left[2\rho\lambda \frac{k}{\varepsilon}\langle uv\rangle \frac{\partial U}{\partial r} - \frac{2}{3}\rho k\left(1 - \frac{\lambda P}{\varepsilon}\right)\right]$$

$$S_{21}^{U} = \frac{1}{r}\frac{\partial}{\partial r}\left[r \rho \frac{\lambda}{\gamma_{21}}\frac{k}{\varepsilon}\left(\langle uu\rangle \frac{\partial V}{\partial x} - \langle uw\rangle \frac{W}{r}\right)\right]$$

$$\gamma_{21} = 1 - \frac{\lambda k}{\varepsilon}\frac{V}{r}$$

Equação de conservação da quantidade de movimento na direção r (componente radial , Eq. 31)

$$\frac{1}{r}\left[\frac{\partial}{\partial x}\left[\rho r U V - \frac{\partial}{\partial x}\left(\frac{\rho \lambda}{\gamma_{21}}\frac{k}{\varepsilon}\langle uu\rangle + \mu\right) r \frac{\partial V}{\partial x}\right] +\right.$$

$$+ \frac{\partial}{\partial r}\left[\rho r VV - \frac{\partial}{\partial r}\left(2\rho\lambda \frac{k}{\varepsilon}\langle vv\rangle + \mu\right) r \frac{\partial V}{\partial r}\right] =$$

$$= -\frac{\partial p}{\partial r} + \mu \frac{\partial}{\partial x}\left(\frac{\partial U}{\partial r}\right) + \frac{\mu}{r}\frac{\partial}{\partial r}\left(\frac{r\,\partial V}{\partial r}\right) - 2\mu \frac{V}{r^2} +$$

$$+ \frac{\rho}{r}\left(W^2 + \langle ww\rangle\right) + S_{21}^V + S_{22}^V$$

onde

$$S_{22}^V = \frac{1}{r}\frac{\partial}{\partial r}\left[r\rho\left[\frac{2\lambda k}{\varepsilon}\left(\langle uv\rangle \frac{\partial V}{\partial x} - \langle vw\rangle \frac{V}{r}\right) + \right.\right.$$

$$\left.\left. - \frac{2k}{3}\left(1 - \frac{\lambda P}{\varepsilon}\right)\right]\right]$$

$$S_{21}^V = \frac{1}{r}\frac{\partial}{\partial x}\left[r\rho\frac{\lambda}{\gamma_{21}}\frac{k}{\varepsilon}\left(\langle vv\rangle \frac{\partial U}{\partial r} - \langle uw\rangle \frac{W}{r}\right)\right]$$

Equação de conservação da quantidade de movimento para a componente tangencial , Eq. 32 .

$$\frac{1}{r}\left[\frac{\partial}{\partial x}\left[\rho r UW - \frac{\partial}{\partial x}\left(\frac{\rho\lambda}{\gamma_{31}}\frac{k}{\varepsilon}\langle uu\rangle + \mu\right) r \frac{\partial W}{\partial x}\right] +\right.$$

$$\left. + \frac{\partial}{\partial r}\left[\rho r VW - \frac{\partial}{\partial r}\left(\frac{\rho\lambda}{\gamma_{23}}\frac{k}{\varepsilon}\langle vv\rangle + \mu\right) r \frac{\partial W}{\partial r}\right] =\right.$$

$$= - \frac{\rho}{r} (VW + \langle vw \rangle) - \frac{\mu W}{r^2} + S_{31}^W + S_{23}^W$$

onde

$$S_{31}^W = \frac{\partial}{\partial x} \left[\frac{\rho \lambda}{\gamma_{31}} \frac{k}{\varepsilon} (\langle uv \rangle \frac{\partial W}{\partial r} + \langle vw \rangle \frac{\partial U}{\partial r}) \right]$$

$$S_{23}^W = \frac{1}{r} \frac{\partial}{\partial r} \left[r \rho \frac{\lambda}{\gamma_{23}} \frac{k}{\varepsilon} (\langle uv \rangle \frac{\partial W}{\partial x} - \langle ww \rangle \frac{W}{r} + \langle uw \rangle \frac{\partial V}{\partial x}) \right]$$

$$\gamma_{31} = 1 - \frac{\lambda k}{\varepsilon} \frac{\partial V}{\partial r}$$

$$\gamma_{23} = 1 - \frac{\lambda k}{\varepsilon} \frac{\partial U}{\partial x}$$

O procedimento para a simulação de escoamentos utilizando o modelo de turbulência algébrico é semelhante ao empregado nos que utilizam a hipótese de Boussinesq, pois para uma ponto do escoamento é possível determinar todas as tensões de Reynolds a partir da energia cinética turbulenta, de sua taxa de dissipação e dos valores das velocidades e de suas derivadas. O que se perde neste tipo de modelo é a relação forte entre as taxas de deformação e as tensões que existe nos modelos de turbulência

que utilizam alguma definição de viscosidade turbulenta. Por este motivo o número de iterações necessário para atingir a convergência é superior ao referente a modelos baseados em viscosidade efetiva.

4.7 Modelo Algébrico Modificado das Tensões de Reynolds

Para amenizar os efeitos da hipótese de Rodi [84] sobre o tratamento dado a convecção e difusão das tensões de Reynolds no modelo diferencial de transporte ,Boysan ,referenciado por Sloan et al.[90], propos a introdução de termos que simulam estes mecanismos no modelo algébrico.Isto é realizado de uma maneira arbitrária e as modificações propostas são :

- correlação < vv > (Eq. 23)

$$2\,\beta\,\frac{\lambda\,k}{\varepsilon}\,\frac{W}{r}\,\langle vw\rangle$$

- correlação < ww > (Eq. 24)

$$-2\,\beta\,\frac{\lambda\,k}{\varepsilon}\,\frac{W}{r}\,\langle vw\rangle$$

- correlação < uv > (Eq. 25)

$$\beta\,\frac{\lambda\,k}{\varepsilon}\,\frac{W}{r}\,\langle uw\rangle$$

- correlação $< uw >$ (Eq. 26)

$$- \beta \frac{\lambda \, k}{\varepsilon} \frac{W}{r} < uv >$$

- correlação $< vw >$ (Eq. 27)

$$\beta \frac{\lambda \, k}{\varepsilon} \frac{W}{r} (< ww > - < vv >)$$

onde

β é uma constante empírica com valor próximo a Ø.3

É interessante notar que as modificações propostas não alteram o valor da energia cinética turbulenta ,pois $< u_i \, u_i >$ continua sendo igual a 2k .

Estes modelos algébricos de turbulência e o k -ε utilizam basicamente a mesma equação de transporte para ε que descreve o processo de dissipação a partir de uma única escala. Provavelmente este é o aspecto mais negativo destes modelos locais de turbulência.

V IMPLEMENTAÇÃO DO PROGRAMA DE SIMULAÇÃO

5.1 Função Interpolação e Método Utilizado na Solução dos Sistemas de Equações Algébricos

Para avaliar os fluxos das variáveis dependentes nas fronteiras dos volumes finitos,de maneira algébrica,é necessária a utilização de funções de interpolação para as variáveis dependentes.Para este fim foram utilizados os sequintes esquemas de interpolação :

- híbrido, Spalding [92] ,que é a combinação da aproximação a montante ("upwind") para o termo convectivo e diferença central para o termo difusivo.Neste esquema são utilizados os quatro nós vizinhos àquele correspondente ao volume de controle finito em questão ;

- SUDS, Raithby [82], que é a combinação da aproximação a montante inclinada (" skew upwind ") para o termo convectivo e diferença central para o termo difusivo.Neste esquema são utilizados os oito nós mais próximos àquele referente ao volume finito em questão ;

- QUICK e seu derivado QUICKER , Leonard [59] e Pollard e Siu [77] respectivamente , que

aproximam o termo convectivo nas superfícies dos volumes finitos a partir de um polinômio do terceiro grau definido pelos valores das variáveis dependentes dos três nós situados na linha perpendicular a superfície em questão. Neste esquema tambem são utilizados oito nós para a obtenção da forma discretizada da equação diferencial de transporte.

Neste trabalho não se testou o esquema que avalia o termo convectivo a partir de polinômios " spline " ajustados com os valores das variáveis dependentes relativas aos nós vizinhos a superfície em que se deseja calcular os fluxos destas variáveis. Segundo Karki et al. [43] o número de iterações necessárias para a obtenção da convergência é menor que aquele relativo ao mesmo programa utilizando o esquema híbrido mas o esforço computacional para isto é bastante elevado. Para as condições relatadas no artigo não é citada nenhuma referência a instabilidades numéricas. A análise deste esquema e do QUICK revela uma similaridade acentuada mas a utilização intensiva de memórias inviabilizou sua implantação.

O esquema híbrido é uma aproximação que garante estabilidade incondicional para o sistema de equações algébricos gerado no processo de discretização, mas induz erros significativos quando a velocidade total não é perpendicular as superfícies dos volumes finitos. Este aspecto, conhecido como difusão numérica devido a inclinação da linha de corrente em relação à malha está muito discutido em Roache [83] e em Raithby [81]. O único modo de tornar este erro tolerável , utilizando a aproximação híbrida

para o termo convectivo ,é o refinamento da malha mas isto necessáriamente implica no aumento do esforço computacional para a solução do mesmo problema.

Um outro modo de se diminuir os efeitos da difusão numérica é a utilização de esquemas mais sofisticados como o SUDS e os da família QUICK.O primeiro foi utilizado por Militzer [71] para a melhor descrição da zona de recirculação existente no escoamento de jatos paralelos não confinados e os da segunda família em várias condições,sempre em escoamentos laminares,por Pollard e Siu [77].Apesar dos esquemas da família QUICK serem considerados de segunda-ordem estes apresentam instabilidades.Isto pode ser visto em Han et al. [32] e em Patel et al. [75] e no caso particular deste trabalho não foi conseguido um resultado final utilizando estes esquemas. O SUDS também não induz estabilidade incondicional como a do esquema híbrido,mas neste trabalho não se notou problemas desta ordem . Um outro motivo para a escolha do SUDS como esquema para a interpolação da variável dependente é que de acordo com Leschziner e Rodi [60] e [61] os resultados provenientes de simulações que usam este esquema são muito próximos daqueles obtidos quando se utiliza o QUICK .

Os métodos utilizados para a solução dos sistemas de equações algébricas gerados no processo de discretização são o método de Thomas aplicado sucessivamente as linhas e colunas de volumes finitos,descrito em Patankar [74],e o método implícito de Stone [94] tambem sucessivamente aplicado as linhas e colunas.Usualmente a utilização do método de Stone reduz de forma moderada o número de iterações necessário para a obtenção da convergência .

5.2 Funções de parede

Para evitar a utilização de um número excessivo de volumes finitos nas regiões próximas as paredes e de modelos de turbulência mais complexos que levem em conta as bruscas variações que ocorrem no escoamento nestas regiões, foram utilizadas as " funções de parede " que relacionam as velocidades, energia cinética turbulenta e sua taxa de dissipação com àquelas da região logarítmica. Estas funções são obtidas a partir da análise uni-dimensional do escoamento admitindo-se que as difusões perpendiculares a parede são os fenômenos dominantes e que a turbulência está , localmente , em equilíbrio.

Investigações experimentais do perfil de velocidade na região vizinha a fronteiras sólidas; no caso de escoamentos simples com pequeno gradiente de pressão, como por exemplo, sobre placas planas e tubos; mostraram que é possível correlacionar, nesta região, a velocidade paralela a superfície com a distância normal a superfície do sequinte modo :

$$U^+ = U^+ (y^+)$$

onde

$$U^+ = \frac{U}{U^*} \qquad \text{velocidade adimensional}$$

$$y^+ = \frac{y\, U^*}{\nu} \qquad \text{coordenada adimensional}$$

$$U^* = \left[\frac{\sigma_w}{\rho} \right]^{\frac{1}{2}}$$ velocidade de atrito

σ_w tensão de cizalhamento na parede

y distância normal a parede

Admitindo ainda que o comprimento de mistura seja proporcional a distância normal a parede, ou seja, a análise não é válida para a sub-camada viscosa, tem-se:

para $y^+ > 11.5$

$$U^+ = \frac{1}{\varkappa} \ln (E\ y^+)$$

onde

$\varkappa$ constante de Von Karman ($\varkappa = Ø.41$)

E constante empírica ($E = 9.8$)

A tensão de cizalhamento na parede pode ser determinada, Launder e Spalding [52], a partir da equação de transporte da energia cinética turbulenta, admitindo-se que a difusão e a convecção de k é desprezível, ou seja , prevalece o equilibrio local e que a viscosidade efetiva é isotrópica. Assim:

$$\sigma_w = (C_D\ C_\mu)^{\frac{1}{2}}\ \rho\ k \qquad (33)$$

onde

C_D e C_μ constantes empíricas

No caso de escoamentos complexos, como o que ocorre na parede lateral da câmara ciclônica, torna-se necessária a alteração da formulação das funções de parede. Primeiramente propõe-se que o perfil logarítmico de velocidade seja válido, mas que as velocidades sejam avaliadas da sequinte forma :

$$U^+_{tot} = \frac{1}{\varkappa} \ln (E\, y^+)$$

com

$U_{tot} = (U^2 + W^2)^{1/2}$ velocidade total

$U^+_{tot} = \dfrac{U_{tot}}{U^*}$ velocidade adimensional

$U^* = \left[\dfrac{\sigma_{tot}}{\rho} \right]^{1/2}$ velocidade total de atrito

$\sigma_{tot} = (\sigma^2_{rx} + \sigma^2_{r\theta})^{1/2}$ tensão de cizalhamento total

σ_{rx} tensão de cisalhamento na direção de x

$\sigma_{r\theta}$ tensão de cizalhamento na direção de θ

$y^+ = \dfrac{y\, U^*}{\nu}$ coordenada adimensional

y distância normal a parede

Como a anisotropia das tensões de Reynolds torna-se pequena na região próxima a parede , como pode ser visto em Kitoh [49], pode-se avaliar a tensão de cizalhamento total na parede do mesmo modo daquele adequado a escoamentos simples, como por exemplo, pela Eq. 33 .

Para a determinação da taxa de dissipação da energia cinética turbulenta, admite-se as mesmas hipóteses daquelas utilizadas na avaliação da energia cinética turbulenta e seu valor pode ser obtido a partir de sua equação de transporte. Assim, para a região considerada tem-se :

$$\varepsilon = \frac{(C_D \, C_\mu)^{3/4}}{C_D \, \varkappa} \; \frac{k}{y}$$

onde

ε	taxa de dissipação de energia cinética turbulenta
C_D e C_μ	constantes empíricas
$\varkappa$	constante de Von Karman
k	energia cinética turbulenta
y	distância normal a parede

Os resultados obtidos para as funções de parede são próximos dos apresentados por Lilley e Rhode [66] e para verificar sua funcionalidade foram simuladas as condições experimentais de Backshall e Landis [1] , de Scott e Barlet

[87] e King et al. [48]. Nos três casos o comportamento das funções se mostrou adequado para valores de y^+ variando de 30 a 350.

A Tabela 1 mostra a relação das funções de parede utilizadas na simulação do escoamento para os nós adjacentes a parede lateral da câmara ciclônica. Para as outras superfícies o desenvolvimento das funções é análogo.

Tabela 1 Funções de Parede
(superfície lateral)

Os valores de k e ε se referem aos nós vizinhos a superfície lateral da câmara e y é a distância entre esta superfície e os nós adjacentes a esta.

$$\sigma_{rx} = \rho \langle uv \rangle = - \mu_{eff} \left(\frac{\partial U}{\partial r} + \frac{\partial V}{\partial x} \right) =$$

$$= \frac{\rho \left(C_D C_\mu \right)^{1/4} k^{1/2} \varkappa U}{\ln \left(E y^+ \right)}$$

$$\sigma_{r\theta} = \rho \langle vw \rangle = - \mu_{eff} \left(\frac{\partial W}{\partial r} - \frac{W}{r} \right) =$$

$$= \frac{\rho \left(C_D C_\mu \right)^{1/4} k^{1/2} \varkappa W}{\ln \left(E y^+ \right)}$$

(continuação da Tab. 1)

$$gen = \rho P = - \left[\rho \langle uu \rangle \frac{\partial U}{\partial x} + \rho \langle vv \rangle \frac{\partial V}{\partial r} + \rho \langle ww \rangle \frac{V}{r} + \right.$$

$$+ \rho \langle uw \rangle \frac{\partial W}{\partial x} + \sigma_{rx} \left(\frac{\partial U}{\partial r} + \frac{\partial V}{\partial x} \right) +$$

$$\left. + \sigma_{r\theta} \left(r \frac{\partial}{\partial r} \left(\frac{W}{r} \right) \right) \right]$$

nos casos em que se utiliza uma definição de viscosidade efetiva isotrópica ,a expressão para a geração de energia cinética turbulenta por unidade de volume ,gen , é :

$$gen = \rho P = 2 \mu_{eff} \left[\left(\frac{\partial U}{\partial x} \right)^2 + \left(\frac{\partial V}{\partial r} \right)^2 + \left(\frac{V}{r} \right)^2 \right] +$$

$$+ \mu_{eff} \left(\frac{\partial W}{\partial x} \right)^2 + \frac{\sigma_{rx}^2}{\mu_{eff}} + \frac{\sigma_{r\theta}^2}{\mu_{eff}}$$

com

$$U_{tot} = \left(U^2 + W^2 \right)^{1/2}$$

$$\sigma_{tot} = \left(\sigma_{rx}^2 + \sigma_{r\theta}^2 \right)^{1/2}$$

$$U_{tot}^+ = \frac{U_{tot}}{U^*} = \frac{U_{tot}}{(\sigma_{tot}/\rho)^{1/2}} = \frac{\ln (E y^+)}{\varkappa}$$

$$y^+ = \frac{y U^*}{\nu} = \frac{y \rho (\sigma_{tot}/\rho)^{1/2}}{\mu} = \frac{y \rho (C_D C_\mu)^{1/4} k^{1/2}}{\mu}$$

(continuação da Tab. 1)

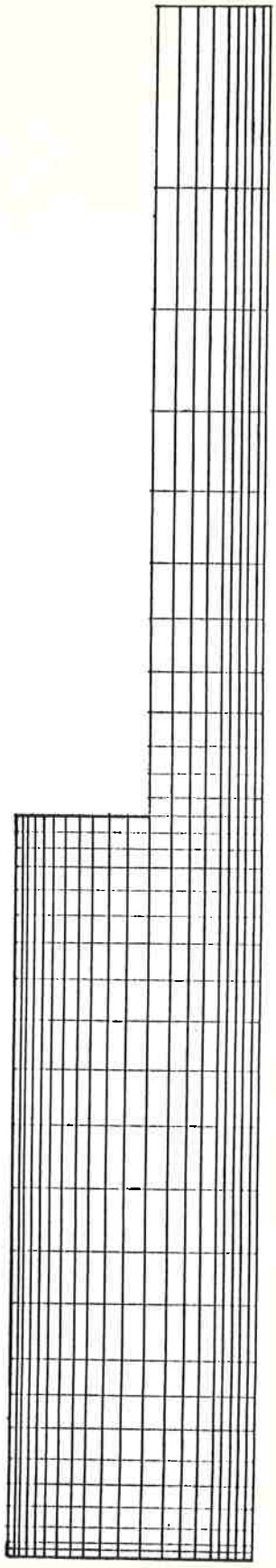
$$k = \frac{1}{\rho} \left(C_\mu \, C_D \right)^{-1/2} \left| \sigma_{tot} \right|$$

$$\varepsilon = \frac{\left(C_D \, C_\mu \right)^{3/4}}{C_D \, \varkappa} \; \frac{k^{3/2}}{y}$$

5.3 Malha

Foram utilizadas duas malhas não uniformes para a delimitação dos volumes finitos. A primeira, denominada grossa e mostrada na Fig. 4, apresenta 36 linhas verticais por 21 horizontais e a segunda, fina, com 43 linhas verticais por 21 horizontais. A malha fina só foi utilizada para verificar se os resultados obtidos nas simulações, utilizando a grossa, eram independentes da malha utilizada. Usualmente a malha fina utilizada para esta verificação de independência apresenta um número bem maior de linhas tanto na horizontal quanto na vertical , mas com o computador disponível só foi possível utilizar a malha fina descrita anteriormente. O fator máximo de expansão ou compressão , definido como a relação entre os comprimentos entre duas linhas horizontais ou verticais sucessivas, utilizado é 1.2 o que não prejudica, segundo Leschziner e Rodi [61], a metodologia utilizada na discretização das equações diferenciais de transporte. É interessante notar que a malha se estende para além da garganta da câmara e isto foi feito para tornar mínima a influência das condições de

linha vertical



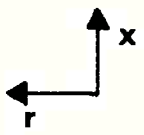


Fig. 4 Malha Grossa

contorno na descarga do equipamento sobre o escoamento interno à câmara. Uma discussão interessante sobre este ponto pode ser encontrada em Escudier [24] que relaciona alguns casos onde este cuidado não foi tomado ,por exemplo Lilley e Rhode [66], e o efeito disto sobre os resultados das simulações.

5.4 Descrição do Caso Padrão e Condições de Contorno Utilizadas nas Simulações.

Como caso padrão foi escolhido o escoamento na câmara ciclônica descrito em Ustimenko e Bukhman [98] para a situação onde o diâmetro de garganta é igual a Ø.1Ø m ,a alimentação é realizada por quatro bocais idênticos e equidistantes e o número de Reynolds igual a 5425 .Este é baseado no diâmetro da câmara (Ø.25 m) e na velocidade axial média no corpo do equipamento.Esta condição experimental foi simulada admitindo-se que a admissão de ar no equipamento seja realizada em toda a periferia ,para que seja possível a análise axissimétrica,com uma largura igual a dos bocais utilizados no experimento.Isto impõe que a velocidade radial de alimentação seja igual a Ø.36 m/s e que a velocidade tangencial ,no mesmo local, seja 2.16 m/s.

No eixo de simetria foram admitidas as sequintes condições de contorno :

$$\frac{\partial U}{\partial r} = Ø$$

$$V = Ø$$

$$W = Ø$$

$$\frac{\partial k}{\partial r} = \emptyset$$

$$\frac{\partial \varepsilon}{\partial r} = \emptyset$$

$$\langle uv \rangle = \langle vw \rangle = \langle uw \rangle = \emptyset$$

onde

U	velocidade média axial
V	velocidade média radial
W	velocidade média tangencial
k	energia cinética turbulenta
ε	taxa de dissipação de k
x	coordenada axial
r	coordenada radial

Nas paredes da câmara foi admitido que as velocidades perpendiculares as paredes são nulas e que os valores de k e ε são calculados utilizando as "funções parede ".Já para a parede da câmara de extensão foi considerado que a velocidade perpendicular a ela é nula mas a derivadas radiais das velocidades axial e tangencial,energia cinética turbulenta e sua taxa de dissipação são nulas.

Na secção de descarga foi admitido que a velocidade radial apresenta valor nulo e que as derivadas axiais de todas as outras variáveis dependentes tambem são nulas. A adoção de valor nulo para a velocidade radial ,nesta secção,reduz um pouco o número de iterações necessário para a obtenção da convergência e não altera os resultados da simulação do escoamento no interior da

câmara. Para a secção de alimentação as condições de contorno adotadas são :

$$U_{in} = \emptyset \qquad m/s$$

$$V_{in} = -\ \emptyset.36 \qquad m/s$$

$$W_{in} = 2.16 \qquad m/s$$

$$k_{in} = \emptyset.\emptyset3\ V_{in}^2 \qquad m^2/s^2$$

$$\varepsilon_{in} = \frac{k_{in}^{3/2}}{\emptyset.\emptyset\emptyset5\ D_c} \qquad m^2/s^3$$

onde

D_c diâmetro da câmara

Os valores utilizados para a energia cinética turbulenta e para sua taxa de dissipação na secção de alimentação foram obtidos no trabalho de Gosman et al. [3Ø] e estes foram sugeridos para o caso de simulação de escoamentos em ciclones para despoeiramento.

5.5 Critério de Convergência e Medidas Tomadas Para o Aumento de Estabilidade do Processo de Simulação

O critério de convergência utilizado é o da somatoria normalizada dos erros residuais absolutos.O erro

residual é definido ,para cada variável dependente,em cada volume finito como :

$$R_{\Phi,P} = a_P \Phi_P - \sum_{viz} a_{viz} \Phi_{viz} - S_c^{\Phi,P}$$

onde

$R_{\Phi,P}$ erro residual para a variável Φ no volume finito referente ao nó P

a_i coeficientes das equações algébricas relativas a Φ

$S_c^{\Phi,P}$ termo da fonte de Φ que não é multiplicado por Φ_P

A somatória normalizada dos erros residuais absolutos é definida por :

$$Resol = \sum_{P} \frac{| R_{\Phi,P} |}{norma_{\Phi}}$$

onde a somatória se extende a todos os volumes finitos e as normas utilizadas foram :

$norma_U$ fluxo em massa no equipamento multiplicado pela velocidade axial média na câmara

$norma_V = norma_U$

$norma_W$ fluxo em massa no equipamento multiplicado pela velocidade tangencial na secção de alimentação

norma k	fluxo em massa no equipamento multiplicado pelo quadrado da velocidade radial na secção de alimentação e dividido por 2
norma continuidade	fluxo em massa no equipamento

Admite-se que a convergência é atingida quando todas as somatórias normalizadas dos erros absolutos sejam menores que Ø.ØØ4. Testes numéricos demostraram que as diferenças nos resultados das simulações são pequenos quando se adota o valor de Ø.ØØ2 ,em vez de Ø.ØØ4 ,para o valor padrão de convergência, o que demonstra a adequação do valor escolhido .

Alem da utilização de sub-relaxação intensa,foram utilizados as sequintes medidas para aumentar a estabilidade do processo dedicado a simular o escoamento na câmara ciclônica:

- O escoamento não foi simulado em uma vez e por questões de estabilidade a velocidade tangencial foi aumentada gradativamente até o seu valor final. Quando utilizou-se o modelo de turbulência k - ε , foram adotadas 4Ø simulações completas para atingir o valor referente a condição experimental e para o modelo algébrico das tensões de Reynolds foram necessárias 8Ø simulações.

- Utilizou-se o artifício da fonte falsa descrito por Lilley [63] que consiste em se adicionar aos termos fontes das equações discretizadas,quando o resíduo da conservação de massa no volume finito é

positivo, o termo :

$$\Delta m \left(\phi_P^{velho} - \phi_P^{novo} \right)$$

onde

Δm — resíduo da conservação de massa

ϕ_P^{velho} — valor de ϕ no nó P obtido na iteração anterior

ϕ_P^{novo} — valor de ϕ no nó P na iteração corrente

Esta medida é interessante pois aumenta o valor do termo da diagonal principal e como decorrência o aumento de estabilidade do sistema de equações.

- Foi utilizado, como no trabalho de Leschizer e Rodi [62] , o artifício da aceleração centrífuga de Gosman que consiste em substituir o termo W^2/r na equação de conservação da quantidade de movimento radial por :

$$\frac{W_P^2}{r} \left[1 + \frac{\alpha}{W_P} \left(V_P^{velho} - V_P^{novo} \right) \right] \qquad (34)$$

onde

α é igual a 2 e foi definido por experiências numéricas

Na convergência o termo entre parênteses é nulo e a equação volta a ter sua forma original. Esta alteração força o acoplamento entre as equações de conservação de quantidade de movimento radial e tangencial e aumenta , como na alteração anterior, os termos da diagonal do sistema de equações algébrico gerado pela discretização.

- Quando o modelo algébrico das tensões de Reynolds é utilizado, as tensões < uv > , < uw > e < vw > não são avaliadas nos nós dedicados às variáveis gerais (todas as dependentes menos U e V) mas de maneira estagiada como mostra a Fig. 5 . Este arranjo aumenta consideravelmente a estabilidade do método de simulação e foi proposto inicialmente por Pope e Whitelaw [79] e estendido por Hogg e Leschziner [37] para o caso de escoamentos axissimétricos .

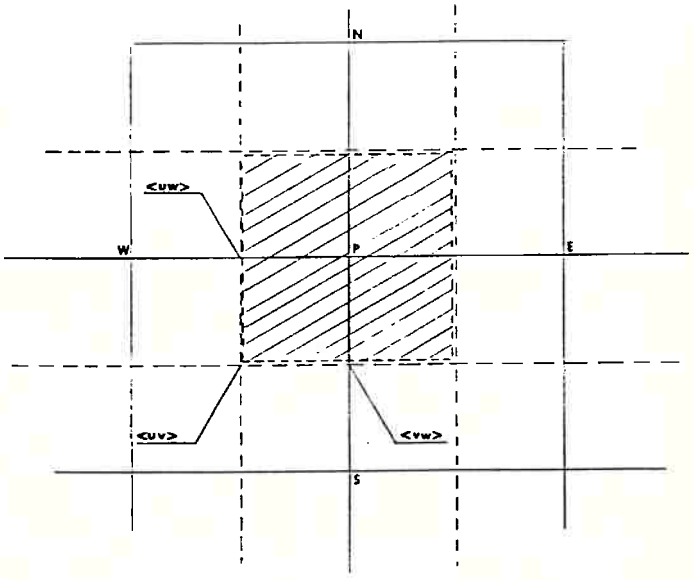


Fig. 5 Pontos de Avaliação das Tensões de Reynolds

- Quando se utiliza o modelo algébrico das tensões de Reynolds é necessário um certo cuidado na linearização do termo

$$(C_{\varepsilon 1} P - C_{\varepsilon 2} \varepsilon) \frac{\rho \varepsilon}{k}$$

presente na equação de transporte da taxa de dissipação da energia cinética turbulenta, Eq. 29 . Neste trabalho a linearização foi realizada de modo análogo ao do artigo de Launder e Morse [56] e consiste em avaliar o termo do sequinte modo :

$$C_{\varepsilon 1} \frac{\rho P \varepsilon^{velho}}{k} - \frac{1}{2} C_{\varepsilon 2} \frac{\rho \varepsilon^{velho}}{k} (\varepsilon^{velho} + \varepsilon^{novo})$$

Para todas as simulações realizadas esta linearização se mostrou eficiente e garantiu a estabilidade do processo de simulação dos escoamentos.

5.6 Arranjo Final das Equações de Transporte e Resumo dos Valores das Constantes Empíricas dos Modelos de Turbulência

O método utilizado, nesta Tese, para a simulação de escoamentos exige que todas as equações de transporte apresentem a mesma forma, que é aquela já mostrada na Eq. 1Ø, e repetida aqui por conveniência.

$$\frac{1}{r}\left[\frac{\partial}{\partial x}\left(r\rho U\Phi - r\Gamma_{\Phi}\frac{\partial \Phi}{\partial x}\right) + \frac{\partial}{\partial r}\left(r\rho V\Phi + \right.\right.$$

$$\left.\left. - r\Gamma_{\Phi}\frac{\partial \Phi}{\partial r}\right)\right] = S_P\Phi + S_C \qquad (10)$$

Quando se utiliza um modelo de turbulência baseado numa viscosidade efetiva isotrópica as equações de transporte, para escoamentos axissimétricos , na forma da Eq. 10 são :

Equação de conservação da quantidade de movimento axial

$$\phi = U$$

$$\Gamma_{\phi} = \mu_{eff}$$

$$S'''_{C,U} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left(\mu_{eff}\frac{\partial U}{\partial x}\right) + \frac{1}{r}\frac{\partial}{\partial r}\left(r\mu_{eff}\frac{\partial V}{\partial x}\right) + $$

$$- \frac{2}{3}\frac{\partial}{\partial x}(\rho k)$$

$$S'''_{P,U} = 0$$

Equação de conservação da quantidade de movimento radial

$$\phi = V$$

$$\Gamma_{\phi} = \mu_{eff}$$

$$S'''_{C,V} = -\frac{\partial p}{\partial r} + \frac{\rho W^2}{r} + \frac{\partial}{\partial x}\left(\mu_{eff}\frac{\partial U}{\partial r}\right) +$$

$$+ \frac{1}{r} \frac{\partial}{\partial r} \left(r \mu_{eff} \frac{\partial V}{\partial r} \right) - \frac{2}{3} \frac{\partial}{\partial r} (\rho k)$$

$$S'''_{P,V} = - \frac{2 \mu_{eff}}{r^2}$$

Quando se utiliza o artifício de Gosman, Eq. 34 , as fontes se modificam para :

$$S'''_{C,V} = - \frac{\partial p}{\partial r} + \frac{\rho W^2}{r} + \frac{\alpha W}{r} V + \frac{\partial}{\partial x} \left(\mu_{eff} \frac{\partial U}{\partial r} \right) +$$

$$+ \frac{1}{r} \frac{\partial}{\partial r} \left(r \mu_{eff} \frac{\partial V}{\partial r} \right)$$

$$S'''_{P,V} = - \frac{2 \mu_{eff}}{r^2} - \frac{\alpha W}{r}$$

Equação de conservação da quantidade de movimento tangencial

$$\phi = W$$

$$\Gamma_{\phi} = \mu_{eff}$$

$$S'''_{C,W} = - \frac{\rho V W}{r} - \frac{W}{r^2} \frac{\partial}{\partial r} (r \mu_{eff})$$

$$S'''_{P,W} = \emptyset$$

Equação de transporte da energia cinética turbulenta

$$\phi = k$$

$$\Gamma_{\phi} = \frac{\mu_{eff}}{Pr_k}$$

$$S'''_{C,k} = gen \qquad \text{(geração de energia cinética turbulenta por unidade de volume)}$$

$$gen = \mu_{eff} \left[2 \left[\left(\frac{\partial U}{\partial x} \right)^2 + \left(\frac{\partial V}{\partial r} \right)^2 + \left(\frac{V}{r} \right)^2 \right] + \left(\frac{\partial U}{\partial r} + \frac{\partial V}{\partial x} \right)^2 + \left(r \frac{\partial}{\partial r} \left(\frac{W}{r} \right) \right)^2 + \left(\frac{\partial W}{\partial x} \right)^2 \right]$$

$$S'''_{P,k} = - C_{\mu} C_D \rho^2 k \mu_{eff}^{-1}$$

Equação para o transporte da taxa de dissipação de energia cinética turbulenta

$$\phi = \varepsilon$$

$$\Gamma_{\phi} = \frac{\mu_{eff}}{Pr_{\varepsilon}}$$

$$S'''_{C,\varepsilon} = C_{\varepsilon 1} C_{\mu} gen \rho k \mu_{eff}^{-1}$$

$$S'''_{P,\varepsilon} = - C_{\varepsilon 2} \rho \varepsilon k^{-1}$$

A notação $S'''_{C,\phi}$ e $S'''_{P,\phi}$ se referem aos valores das fontes por unidade de volume.

Os valores adotados para as constantes empíricas dos modelo de turbulência k - ε são os clássicos e valem :

$C_{\mu} = 0.09$

$C_{\varepsilon 1} = 1.44$

$C_{\varepsilon 2} = 1.92$

$Pr_{k} = 1.0$

$Pr_{\varepsilon} = 1.3$

Quando se utiliza o modelo algébrico das tensões de Reynolds, para escoamentos axissimétricos, a forma da Eq. 10 deve ser alterada para :

$$\frac{1}{r}\left[\frac{\partial}{\partial x}\left(r\rho U\Phi - r\Gamma_{\Phi x}\frac{\partial\Phi}{\partial x}\right) + \frac{\partial}{\partial r}\left(r\rho V\Phi + \right.\right.$$

$$\left.\left. - r\Gamma_{\Phi r}\frac{\partial\Phi}{\partial r}\right)\right] = S_P\Phi + S_C \qquad (35)$$

pois a hipótese de isotropia da viscosidade turbulenta deve ser relaxada. Assim as equações de transporte, na forma adequada

são :

Equação de conservação da quantidade de movimento axial

$$\phi = U$$

$$\Gamma_{\phi x} = 2 \rho \lambda \frac{k}{\varepsilon} \langle uu \rangle + \mu$$

$$\Gamma_{\phi r} = \frac{\rho \lambda}{\gamma_{21}} \frac{k}{\varepsilon} \langle vv \rangle + \mu$$

$$S'''_{C,U} = - \frac{\partial p}{\partial x} + \mu \frac{\partial}{\partial x} \left(\frac{\partial U}{\partial x} \right) + \frac{\mu}{r} \frac{\partial}{\partial r} \left(\frac{\partial V}{\partial x} \right) +$$

$$+ S^U_{11} + S^U_{21}$$

$$S'''_{P,U} = \emptyset$$

onde

γ_{21} , S^U_{11} , S^U_{21} e λ definidos na Eq. 3Ø

Equação de conservação da quantidade de movimento radial

$$\phi = V$$

$$\Gamma_{\phi x} = \frac{\rho \lambda}{\gamma_{21}} \frac{k}{\varepsilon} \langle uu \rangle + \mu$$

$$\Gamma_{\phi r} = 2 \rho \lambda \frac{k}{\varepsilon} \langle vv \rangle + \mu$$

$$S'''_{C,V} = -\frac{\partial p}{\partial r} + \mu \frac{\partial}{\partial x}\left(\frac{\partial U}{\partial r}\right) + \frac{\mu}{r}\frac{\partial}{\partial r}\left(r\frac{\partial V}{\partial r}\right) + \frac{\rho}{r}\left(W^2 + \langle ww \rangle\right) + \frac{\alpha W}{r} V + S^V_{21} + S^V_{22}$$

$$S'''_{P,V} = -\frac{2\mu}{r^2} - \frac{\alpha W}{r}$$

onde

γ_{21} , S^V_{21} e S^V_{22} definidos ne Eq. 31

Equação de conservação da quantidade de movimento tangencial

$$\phi = W$$

$$\Gamma_{\phi x} = \frac{\rho \lambda}{\gamma_{31}} \frac{k}{\varepsilon} \langle uu \rangle + \mu$$

$$\Gamma_{\phi r} = \frac{\rho \lambda}{\gamma_{23}} \frac{k}{\varepsilon} \langle vv \rangle + \mu$$

$$S'''_{C,W} = -\frac{\rho}{r}\left(VW + \langle vw \rangle\right) - \frac{\mu W}{r^2} + S^W_{31} + S^W_{23}$$

$$S'''_{P,W} = \emptyset$$

onde

γ_{31} , γ_{23} , S^W_{31} e S^W_{23} definidos na Eq. 32

Equação de transporte para a energia cinética turbulenta

$$\phi = k$$

$$\Gamma_{\phi x} = C_k \rho \frac{k}{\varepsilon} \langle uu \rangle + \frac{\mu}{Pr_k}$$

$$\Gamma_{\phi r} = C_k \rho \frac{k}{\varepsilon} \langle vv \rangle + \frac{\mu}{Pr_k}$$

$$S'''_{C,k} = \rho P - \rho \varepsilon + \frac{\partial}{\partial x}\left[\rho C_k \frac{k}{\varepsilon} \langle uv \rangle \frac{\partial k}{\partial r} \right] +$$

$$+ \frac{1}{r} \frac{\partial}{\partial r}\left[\rho C_k \frac{k}{\varepsilon} \langle uv \rangle \frac{\partial k}{\partial x} \right]$$

$$S'''_{P,k} = \emptyset$$

onde

P é a geração de energia cinética de turbulência por unidade de massa, definida na Eq. 28 , C_k e Pr_k são constantes empíricas

Equação de transporte da taxa de dissipação da energia cinética turbulenta

$$\phi = \varepsilon$$

$$\Gamma_{\phi x} = C_{\varepsilon 3} \rho \frac{k}{\varepsilon} \langle uu \rangle + \frac{\mu}{Pr_\varepsilon}$$

$$\Gamma_{\phi r} = C_{\varepsilon 3}\,\rho\,\frac{k}{\varepsilon}\,\langle vv\rangle + \frac{\mu}{Pr_{\varepsilon}}$$

$$S'''_{C,\varepsilon} = C_{\varepsilon 1}\,\rho\,P\,\frac{\varepsilon}{k} - \frac{1}{2}\,C_{\varepsilon 2}\,\rho\,\frac{\varepsilon^2}{k} + \frac{\partial}{\partial x}\left[\rho C_{\varepsilon 3}\,\frac{k}{\varepsilon}\langle uv\rangle\,\frac{\partial \varepsilon}{\partial r}\right] + \frac{1}{r}\,\frac{\partial}{\partial r}\left[r\rho C_{\varepsilon 3}\,\frac{k}{\varepsilon}\langle uv\rangle\,\frac{\partial \varepsilon}{\partial x}\right]$$

$$S'''_{P,\varepsilon} = -\frac{1}{2}\,C_{\varepsilon 2}\,\rho\,\frac{\varepsilon}{k}$$

onde

$C_{\varepsilon 1}$, $C_{\varepsilon 2}$, $C_{\varepsilon 3}$ e Pr_{ε} são constantes empíricas

Os valores das constantes empíricas utlilizados neste modelo algébrico de tensões de Reynolds foram escolhidos a partir dos disponíveis na bibliografia e não foi feita a otimização destes objetivando a melhor aderência dos valores obtidos por simulação com os experimentais.Os valores adotados são :

$C_1 = 1.8$ $C_2 = Ø.6$	Gibson e Launder	[27]
$C_k = Ø.22$	Launder e Morse	[56]
$C_{\varepsilon 3} = Ø.18$	Fu et al.	[25]
$Pr_k = Ø.9$ $Pr_{\varepsilon} = 1.22$	Sloan et al.	[9Ø]
$C_{\varepsilon 1} = 1.44$ $C_{\varepsilon 2} = 1.92$	Fu et al.	[25]

Quando se utiliza a modificação de Boysan no modelo algébrico adota-se que β é igual a Ø.3 como descrito em Sloan et al. [9Ø].

5.7 Resumo da Abordagem Proposta

- Utiliza-se a equação de transporte das tensões de Reynolds truncada para a forma algébrica admitindo-se que a relação entre $< u_i \; u_j >$ e a energia cinética turbulenta é constante no escoamento. Admite-se que a dissipação é isotrópica, que o termo difusivo das tensões de Reynolds é bem representado pelo tratamento de Daly e Harlow [19] e que o acoplamento entre flutuação de pressão e as tensões de Reynolds seja função das tensões de Reynolds, da energia cinética turbulenta e de sua taxa de dissipação e do tensor de geração de energia cinética turbulenta conforme o tratamento de Gibson e Launder [26] e [27]. Propõe-se tambem a utilização de valores clássicos para as constantes empíricas do modelo de turbulência completo, ou seja, não será feita otimização das constantes com o objetivo de obter a melhor descrição possível do escoamento.

-Propos - se novas escalas para a função de parede adequadas para a região logarítmica de escoamentos confinados com componente tangencial de velocidade que foram testadas, com sucesso, em três condições experimentais distintas.

VI RESULTADOS DAS SIMULAÇÕES

Os resultados obtidos por simulação ,utilizando-se malha grossa e a função de interpolação SUDS , serão mostrados do mesmo modo que o utilizado por Ustimenko e Bukhman [98] ,ou seja,serão apresentados os perfís de velocidade axial e tangencial e o de energia cinética turbulenta em quatro secções transversais .As distâncias entre estas secções e a base da câmara ciclônica são Ø.159 m, Ø.223 m, Ø.286 m e Ø.35Ø m e estão mostradas na Fig. 6.

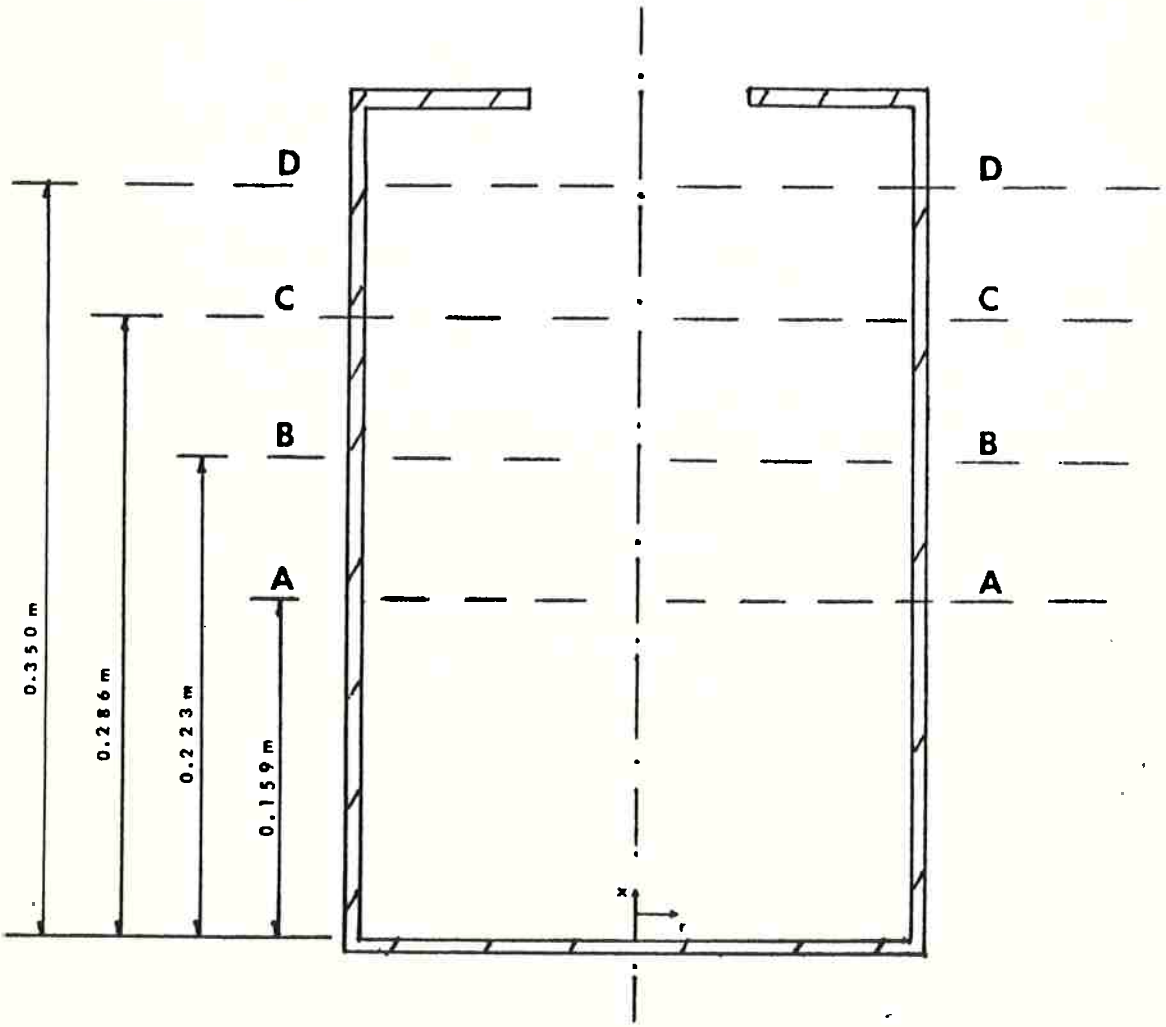


Fig. 6 Secções da Câmara

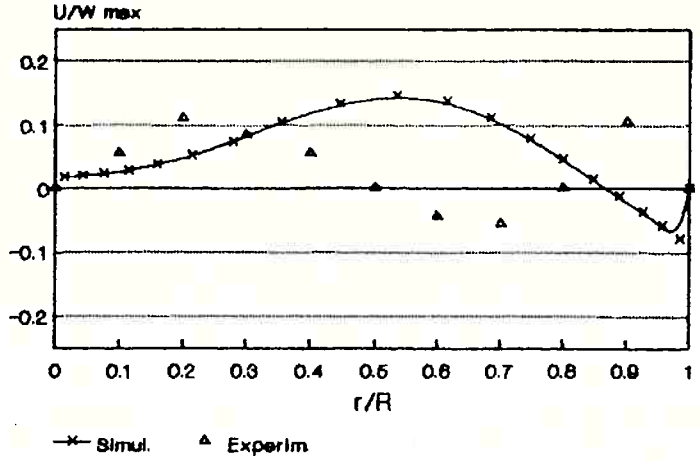
As Figs. 7, 8, 9 e 10 mostram os perfis de velocidade axial e tangencial e o de energia cinética turbulenta obtidos utilizando-se o modelo de turbulência k - ε básico com as constantes empíricas clássicas definidas no capítulo anterior. Não serão apresentados os resultados de simulações que utilizaram os sequintes modelos de turbulência k - ε modificados :

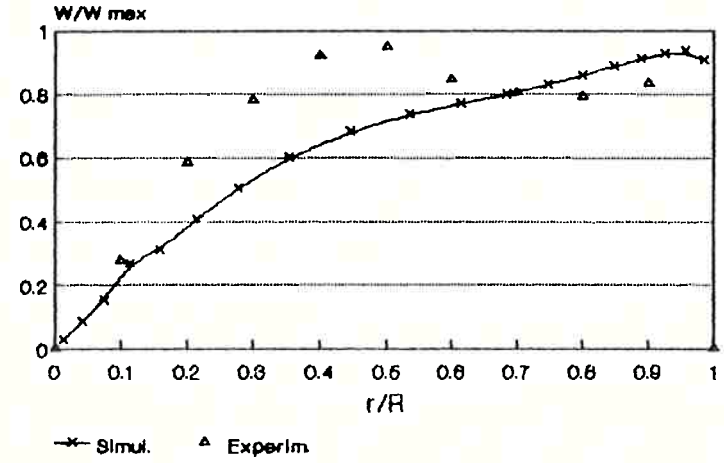
- o de Lilley [67] que é baseado na simples alteração das constantes empíricas do modelo básico;
- o de Launder et al. [55] que utiliza uma forma modificada, pela introdução do número de Richardson , para a equação de transporte de ε ;
- o de Pope [78] que substituiu a constante C_μ por uma função baseada nas taxas locais de deformação e rotação.

Isto foi feito porque não se verificou uma alteração substancial na descrição do escoamento em relação àquela obtida com a utilização do modelo k - ε básico.

As Figs. 11, 12, 13 e 14 mostram os mesmos perfis obtidos utilizando-se o modelo algébrico das tensões de Reynolds descrito no ítem 4.4. Já para a criação das Figs. 15 , 16, 17 e 18 foi utilizado o modelo algébrico de tensões modificado pelas adições de Boysan descritas no ítem 4.5 .

Todos os resultados apresentados estão na forma adimensional. A coordenada radial foi adimensionalizada pelo raio da câmara, as velocidades axial e radial pela máxima velocidade média tangencial encontrada no equipamento, W_{max} , e a energia cinética turbulenta pelo valor de $W_{max}^2 / 2$.





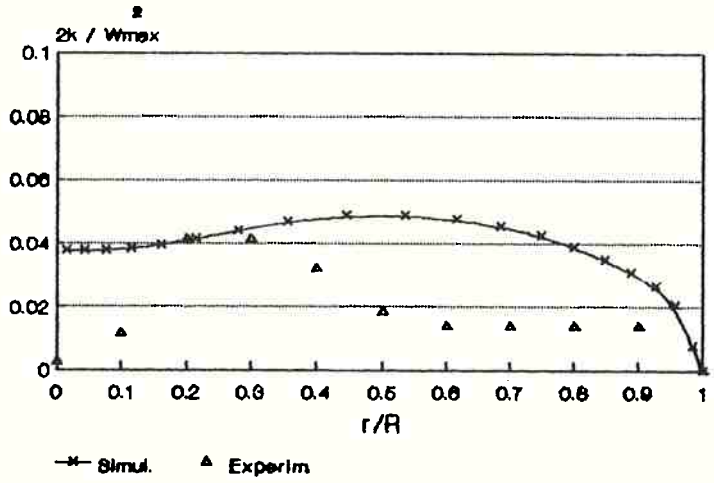
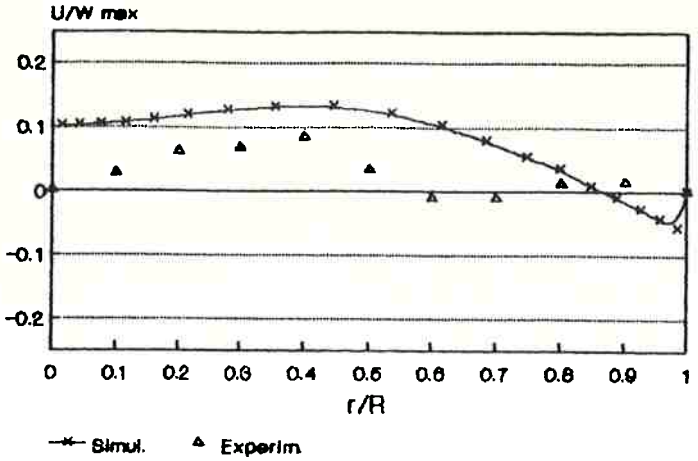
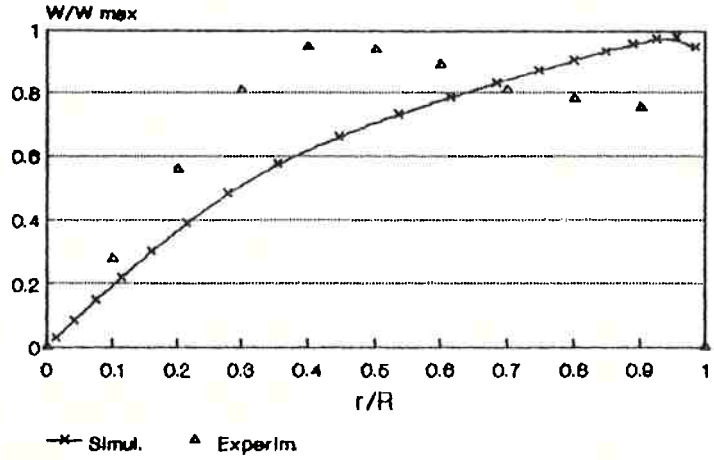


Fig. 7 Perfis de Velocidade e de k para a Secção A-A modelo de turbulência k - ε





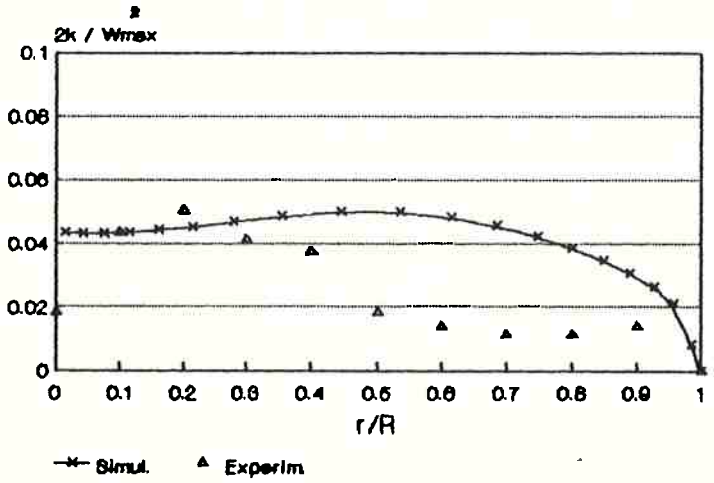
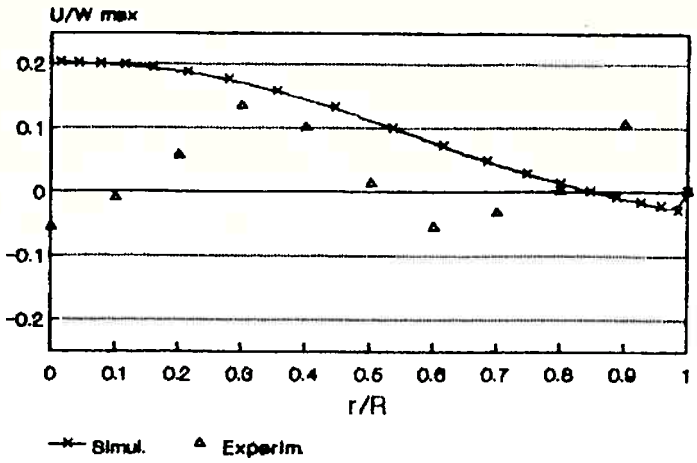
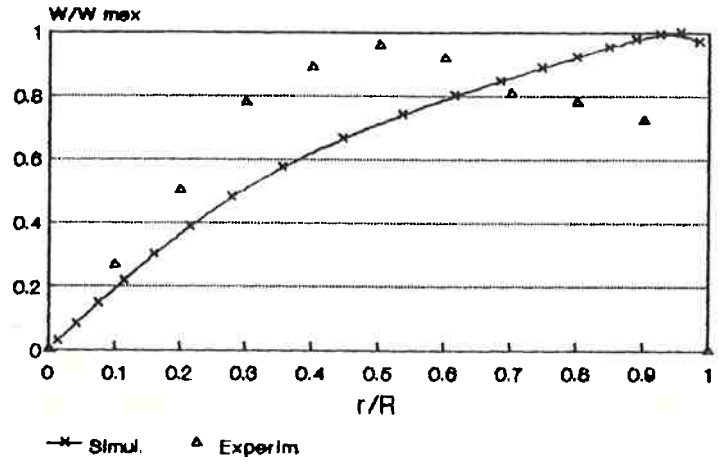


Fig. 8 Perfís de Velocidade e de k para a Secção B-B modelo de turbulência k - ε





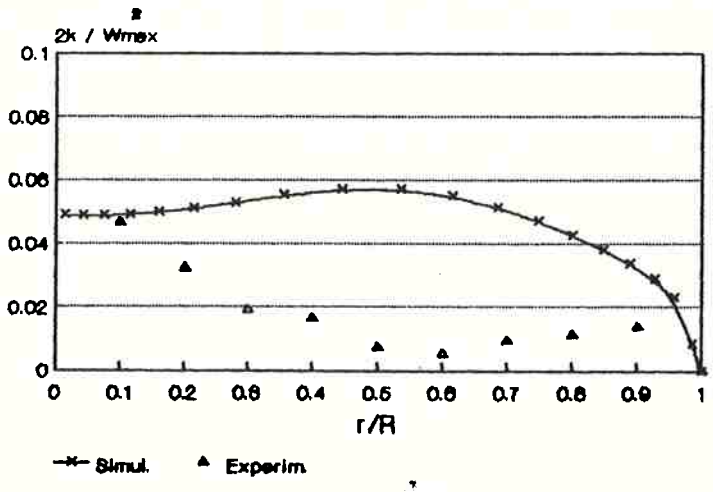
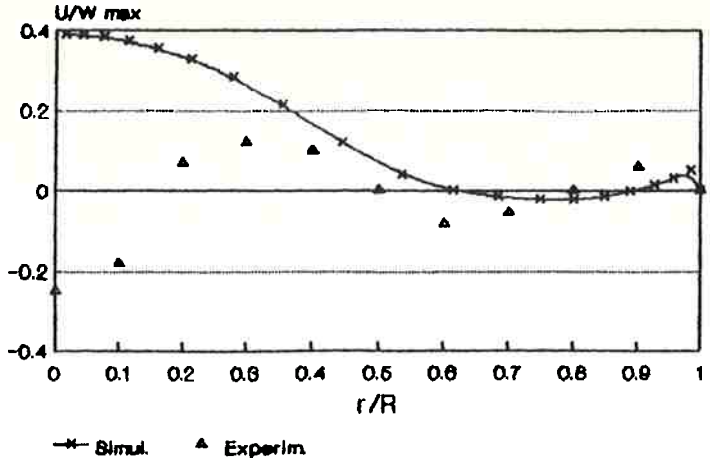
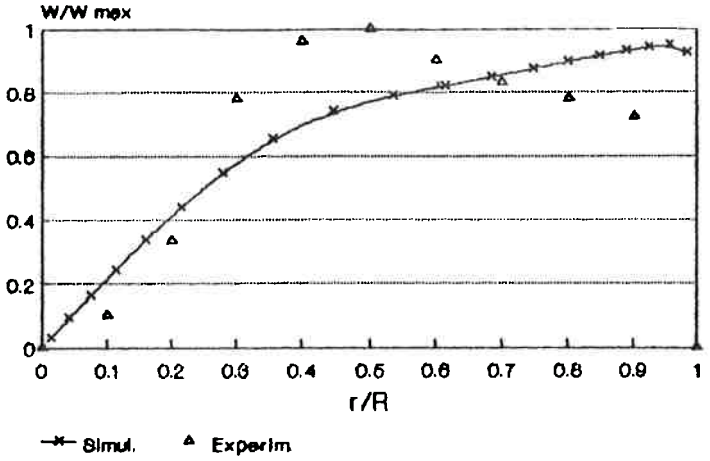


Fig. 9 Perfís de Velocidade e de k para a Secção C-C modelo de turbulência k - ε





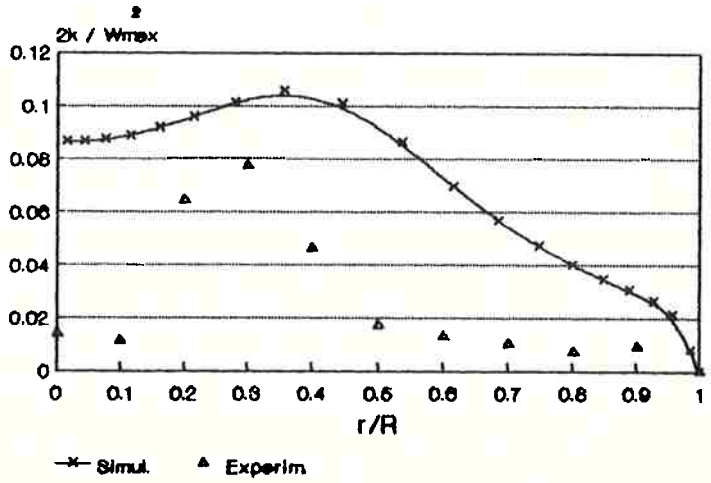
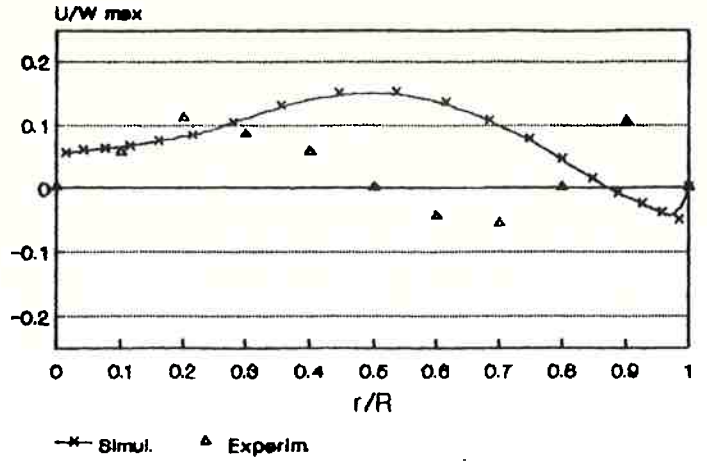
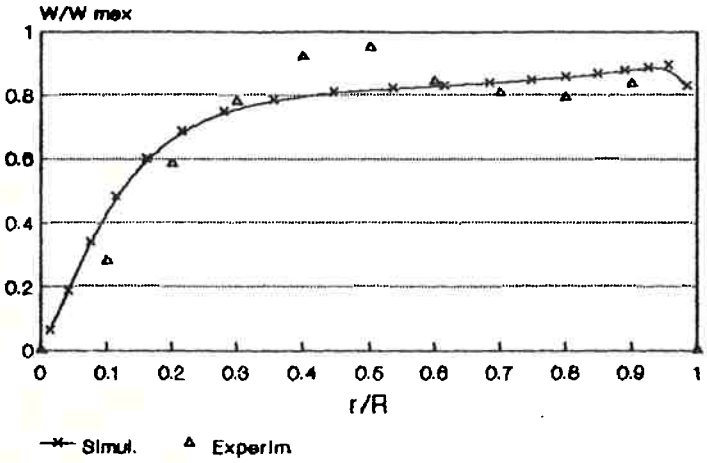


Fig. 1Ø Perfis de Velocidade e de k para a Secção D-D modelo de turbulência k - ε





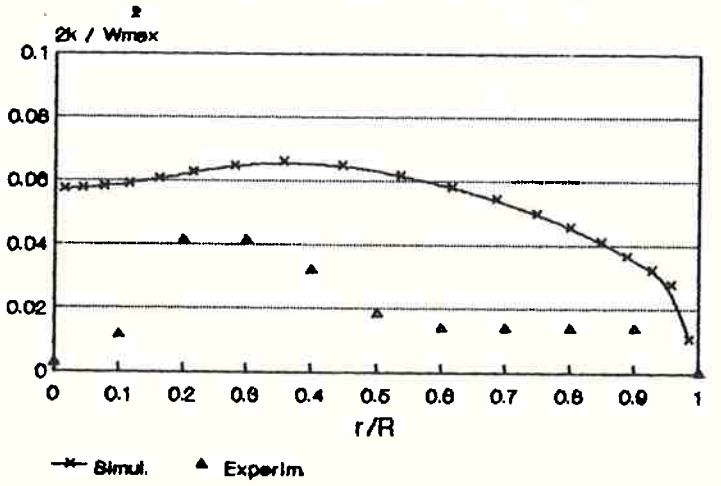
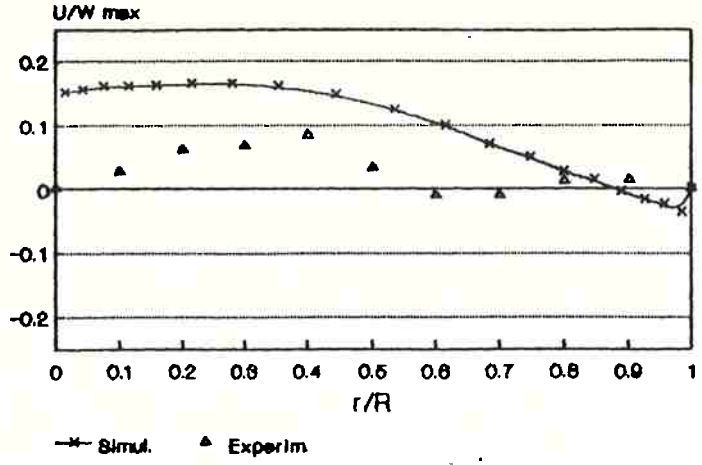
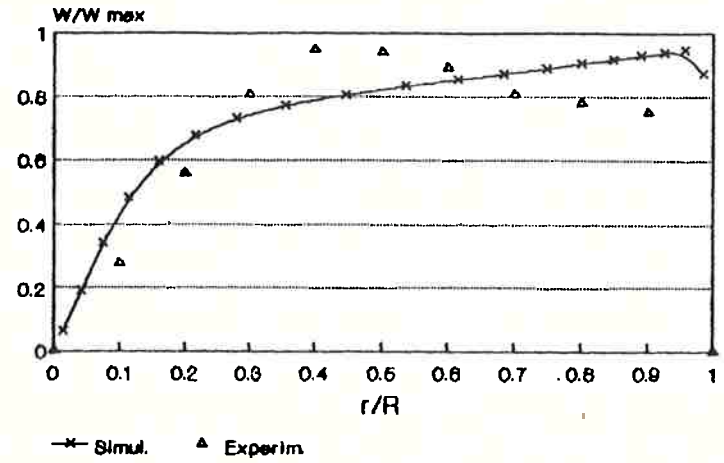


Fig. 11 Perfis de Velocidade e de k para a Secção A-A modelo algébrico das tensões de Reynolds





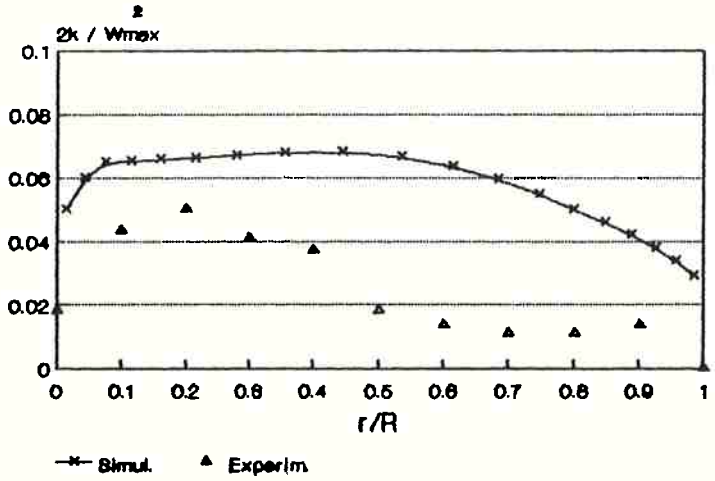
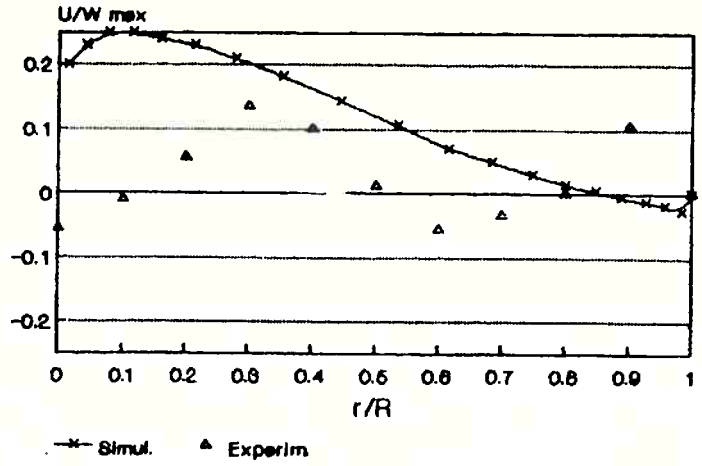
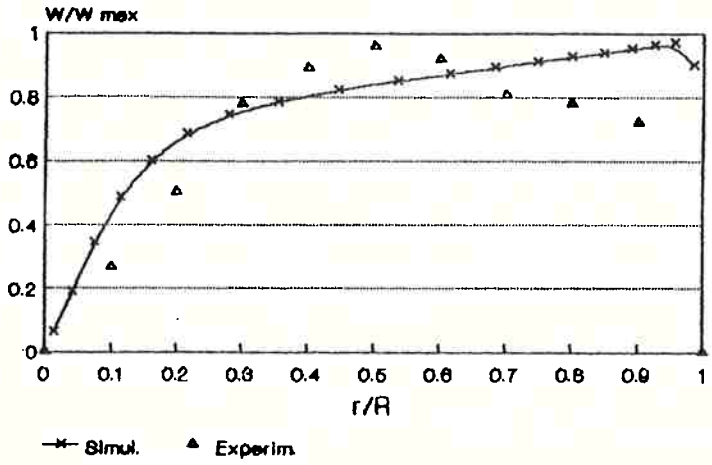


Fig. 12 Perfis de Velocidade e de k para a Secção B-B modelo algébrico das tensões de Reynolds





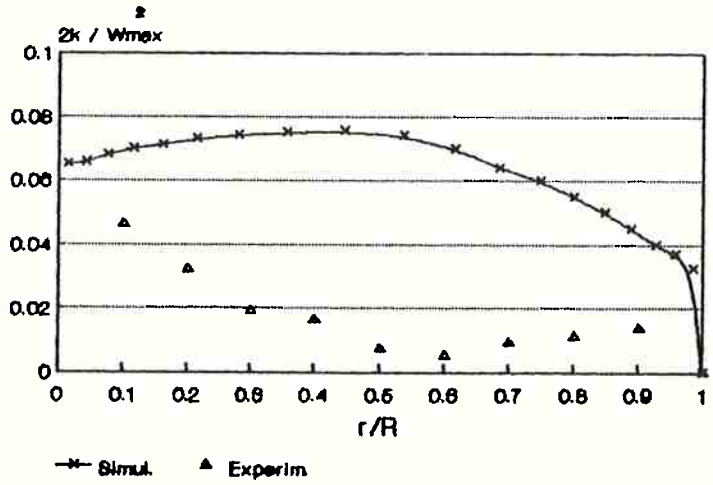
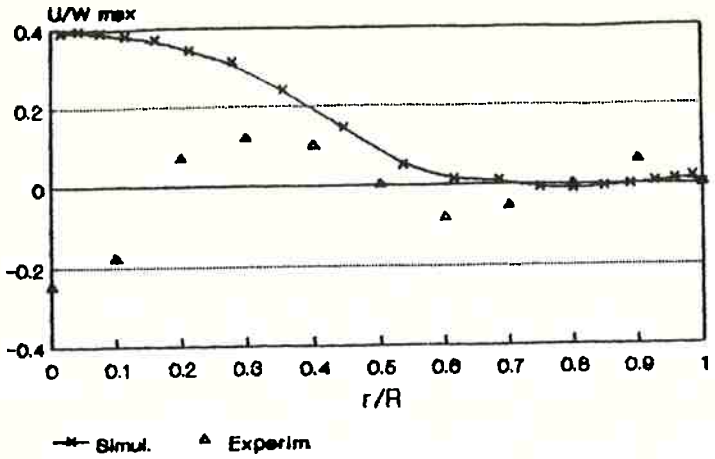
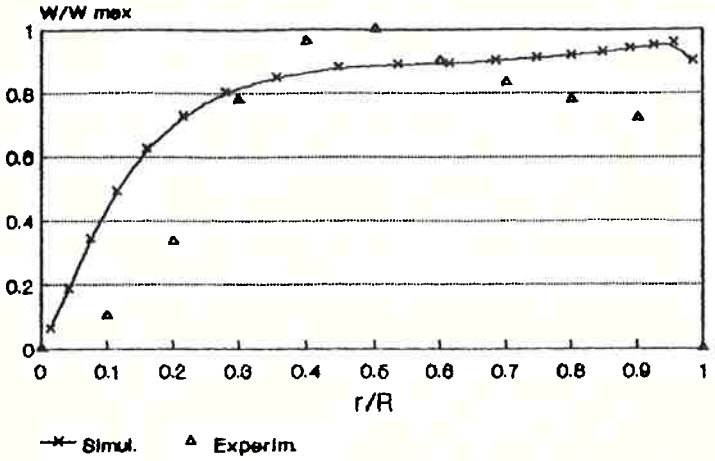


Fig. 13 Perfis de Velocidade e de k para a Secção C-C modelo algébrico das tensões de Reynolds





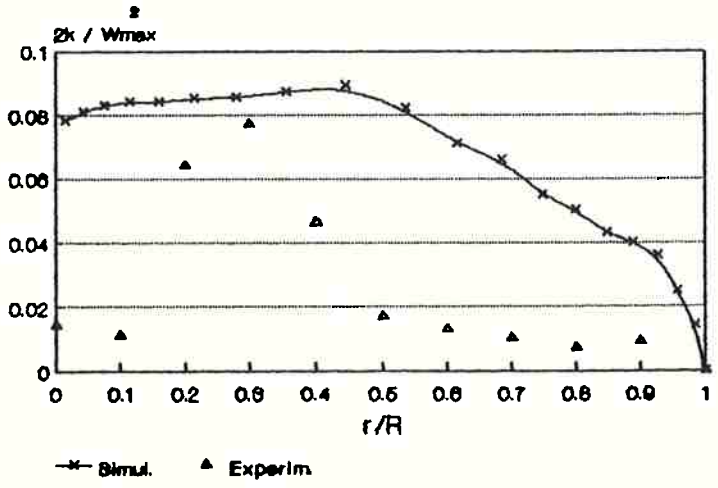
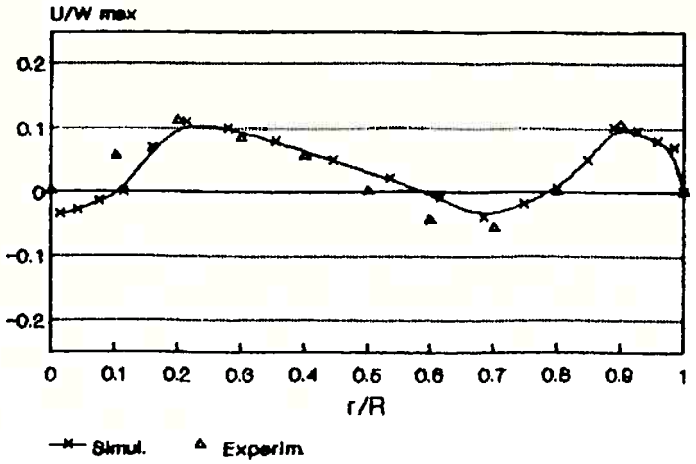
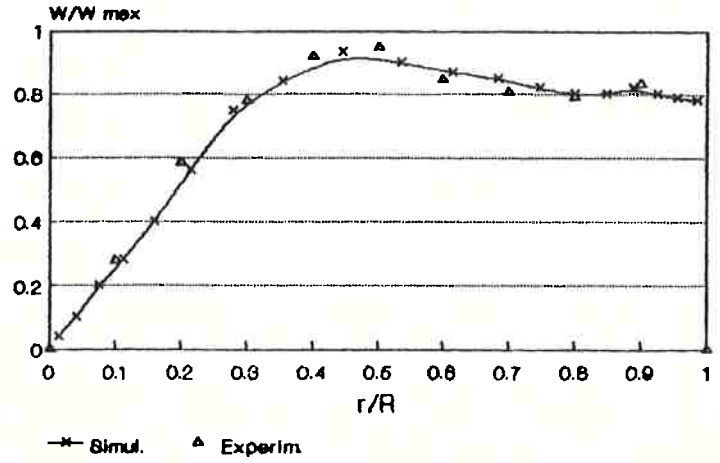


Fig. 14 Perfis de Velocidade e de k para a Secção D-D modelo algébrico das tensões de Reynolds





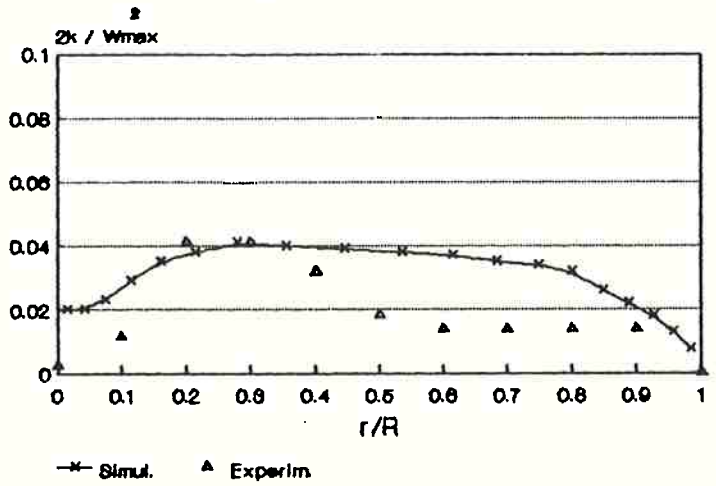
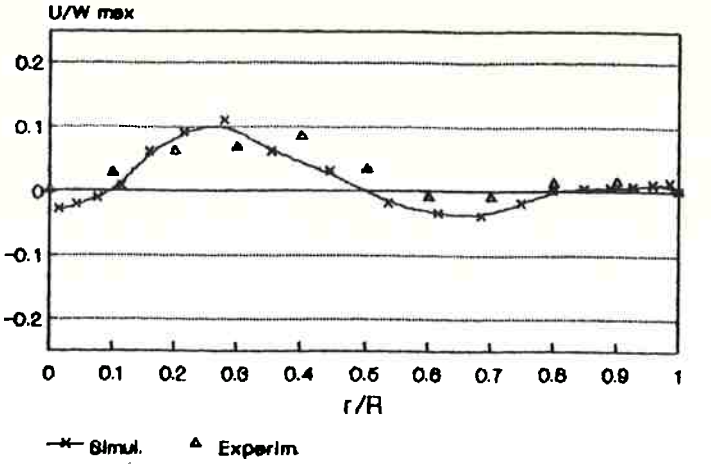
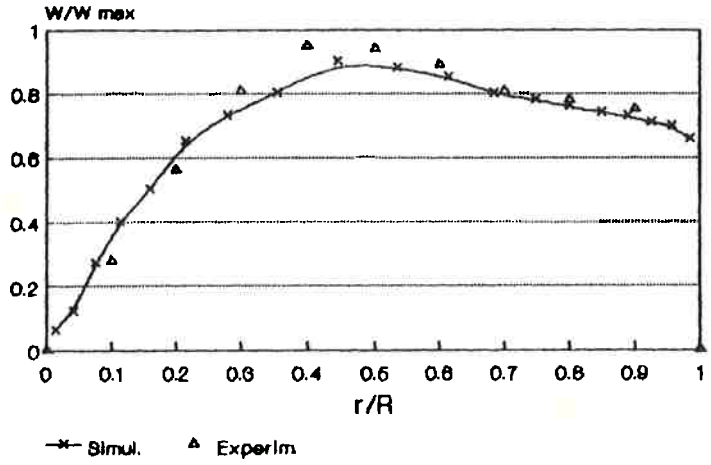


Fig. 15 Perfís de Velocidade e de k para a Secção A-A modelo algébrico das tensões modificado





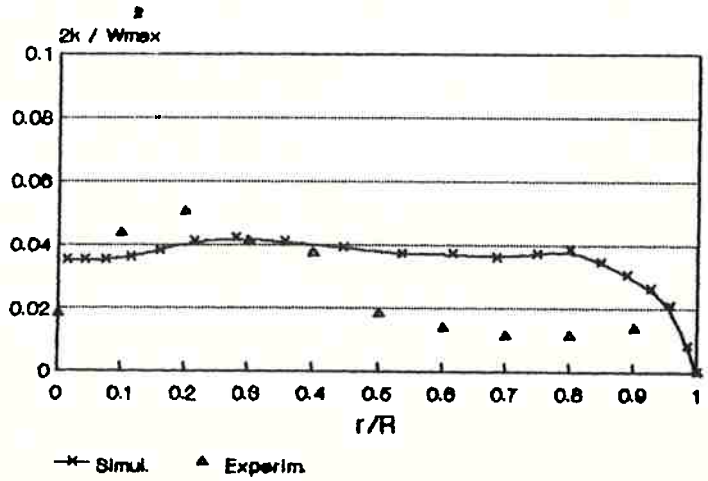
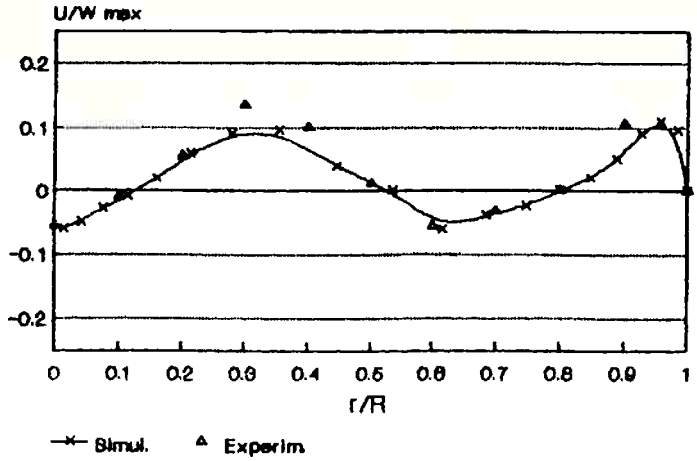
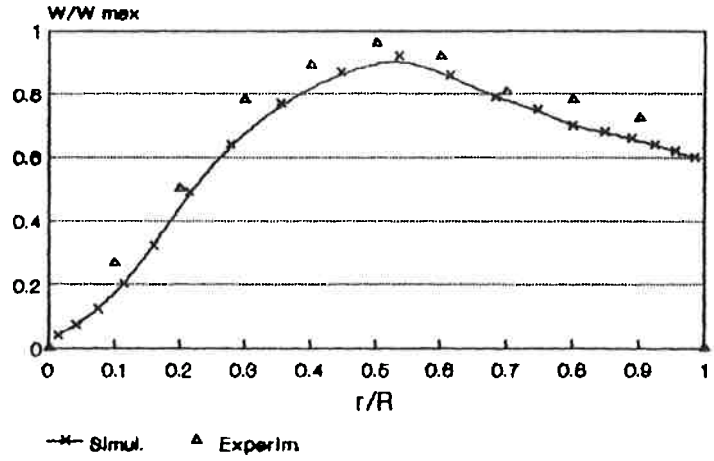


Fig. 16 Perfís de Velocidade e de k para a Secção B-B modelo algébrico de tensões modificado





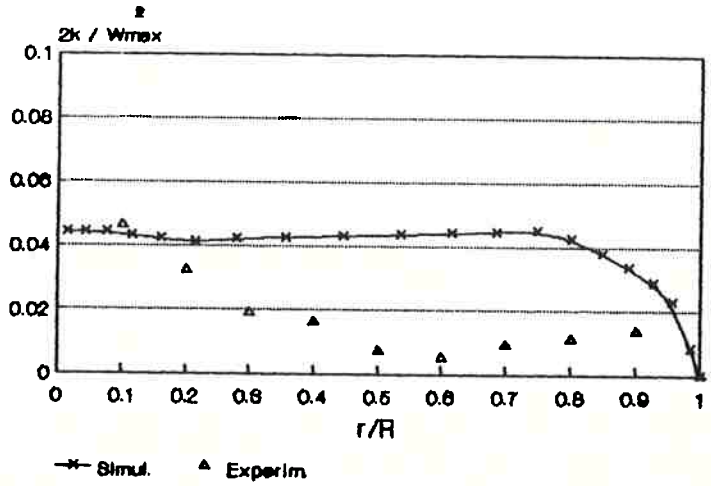
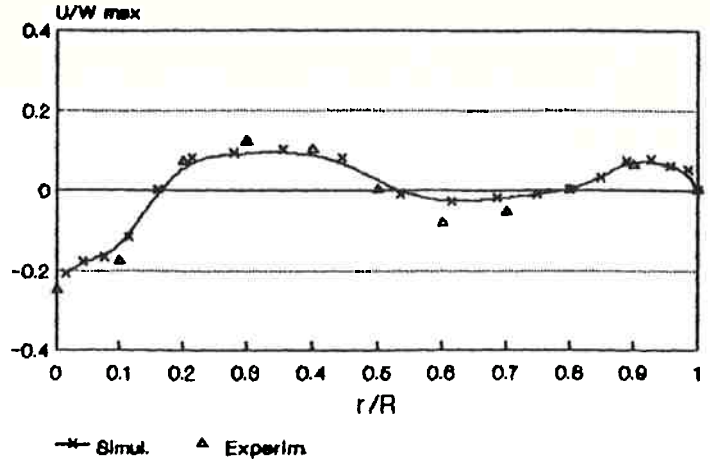
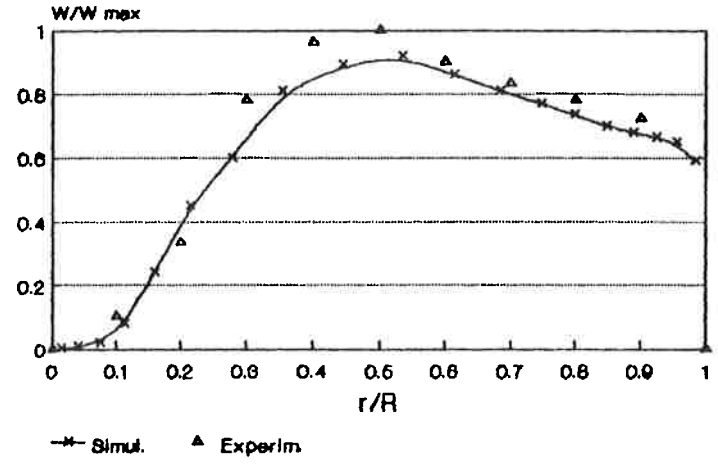


Fig. 17 Perfis de Velocidade e de k para a Secção C-C modelo algébrico de tensões modificado





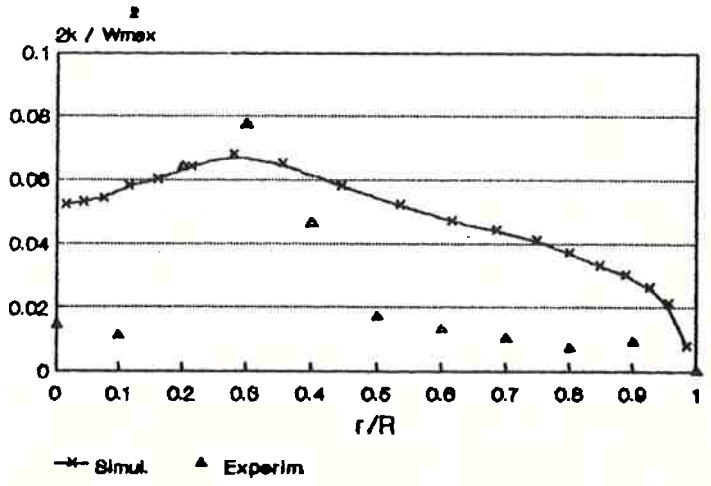


Fig. 18 Perfis de Velocidade e de k para a Secção D-D modelo algébrico de tensões modificado

VII CONCLUSÕES E CONSIDERAÇÕES FINAIS

Os resultados das simulações apresentados no Capítulo anterior foram obtidos utilizando-se malha grossa e a função de interpolação SUDS. Quando a malha grossa é substituida pela fina, no processo de simulação, a diferença máxima entre os campos das mesmas variáveis dependentes é inferior a 3% e ocorre na simulação que utiliza o modelo algébrico das tensões de Reynolds. Nestas condições pode-se afirmar que a solução é independente da malha utilizada.

A substituição da função de interpolação SUDS pela função híbrida, quando se utiliza malha grossa e qualquer um dos modelos de turbulência utilizados nesta Tese, altera de forma mínima os campos das variáveis dependentes. Isto indica que a malha está bem alocada no campo de escoamento pois os efeitos da difusão falsa não foram significativos.

As Figs. 7,8,9 e 10 , relativas a utilização do modelo de turbulência k - ε básico , mostram que não foi obtida uma recirculação central no escoamento e que os perfís de velocidade média tangencial se aproximam daquele referente aos vórtices forçados. Já os perfís de energia cinética turbulenta apresentam valores superiores aos medidos experimentalmente, mas a ordem de grandeza é a mesma. Este conjunto de evidências mostra que o modelo de turbulência k - ε com sua definição de viscosidade efetiva isotrópica é inadequado para a simulação de escoamentos em câmaras

ciclônicas.Os perfís de velocidade média axial e tangencial obtidos com as três alterações do modelo k - ε básico citadas no Capítulo anterior não apresentaram mudanças radicais em relação aos perfís mostrados nas Figs.7 a 1Ø .A Fig. 19 mostra o perfil de velocidade média axial referente a secção C - C da Fig.6 obtido utilizando-se a formulação de Pope [78] para C_μ . Nota-se que a simulação continua grosseira e que não se obteve um valor negativo para a velocidade média axial no eixo de simetria da câmara.

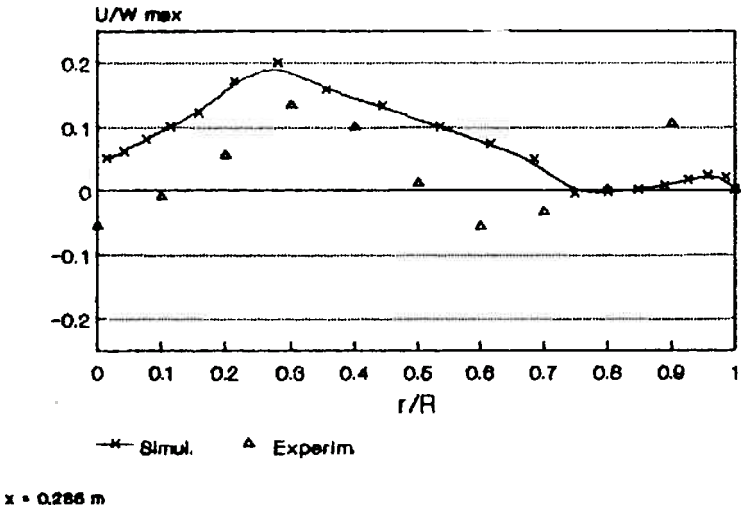


Fig. 19 Perfil de Velocidade Média Axial para a Secção C-C modelo de turbulência k - ε modificado por Pope

As Figs. 11,12,13 e 14 ,relativas a utilização do modelo algébrico das tensões de Reynolds,mostram que tambem não foi obtida uma recirculação central no escoamento e que os perfís de velocidade tangencial são mais próximos dos valores experimentais do que os referentes ao modelo k - ε .A tendência do perfil de velocidade tangencial continua sendo a do vórtice forçado e esta manifestação mostra que a redistribuição de energia cinética turbulenta entre as tensões principais de Reynolds está errada.

As Figs. 15 , 16 ,17 e 18 ,relativas a utilização do modelo algébrico das tensões de Reynolds com as modificações de Boysan ,mostram uma melhora significativa na descrição global do escoamento .Nota-se que a recirculação está presente na simulação mas se apresenta mais intensa do que a observada experimentalmente e chegou a alcançar a Secção A - A.Os perfís de velocidade média axial e tangencial nas regiões próximas as paredes laterais são muito próximos dos reportados por Ustimenko e Bukhman [98] e os maiores afastamentos ocorrem nas regiões que apresentam o raio adimensional próximo a Ø.5. Já os perfís da energia cinética turbulenta apresentam comportamento diverso dos relativos as velocidades e sempre apresentam valores maiores que os medidos experimentalmente.É muito difícil discutir esta disparidade pois o erro pode ser causado pelo modelamento da turbulência ou pode residir parcialmente no processo de medida utilizado no experimento.Para verificar o macro-comportamento da simulação comparou-se o valor da perda de carga obtida nestas condições com o valor determinado por equações empíricas

correlacionadas por Tager [97] para o escoamento em câmaras ciclônicas de mesmo aspecto geométrico.O resultado da comparação é que a perda de carga obtida por simulação é 5 % superior àquela obtida pelo procedimento empírico, o que é bastante satisfatório.

Tendo em vista a qualidade dos resultados obtidos por simulação e a complexidade do escoamento em câmaras ciclônicas ,pode-se afirmar que as proposições feitas nesta Tese , acrescidas das modificações de Boysan , são plausíveis e merecem ser testadas em outras geometrias de câmaras ciclônicas que apresentem elevada recirculação.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Backshall,R. G.; Landis, F. , The Boundary Layer Velocity Distribution in Turbulent Swirling Pipe Flow,J. Basic Eng.,ASME,Vol.91,Dec. 1969,pp 728-733

[2] Baluev,E. D.;Troyakin, Yu. V., Study of the Aerodynamic Structure of Gas Flow in Cyclone Chambers,Thermal Eng., Vol. 14,No 1,1967,pp 84-87

[3] Baluev,E. D.;Troyakin, Yu. V.,The Effect of the Design Parameters on the Aerodynamics of Cyclone Chambers, Thermal Eng.,Vol. 13,No 1,1967,pp 99-1Ø5

[4] Bardina, J.;Ferziger, J.H.;Rogallo, R.S.,Effect of Rotation on Isotropic Turbulence Computation and Modelling,J. Fluid. Mech.,Vol. 154,1985,pp 321-336

[5] Barnhart,J. S.;Laurendeau,N. M.,Pulverized Coal Combustion and Gasification in a Cyclone Reactor Experiment,Parte 1 e 2 ,Ind Eng Chem Process Des Dev, Vol.21,1982,pp 671-689

[6] Bilger, R. W.,A Note on Favre Averaging in Variable Density Flows, Comb. Sci. Technol.,Vol.11,No. 2,1975, pp 215-23Ø

[7] Bloor, M. I. G.;Ingham,D. B.,The Flow in Industrial Cyclones,J. Fluid Mech.,Vol. 178,1987,pp 5Ø7-519

[8] Boysan F.;Swithenbank,J.,Numerical Prediction of Confined Vortex Flows,em Turbulent Shear Flows II, Eds. Durst, F.;Launder, B. E.;Schmidt,F. W.;Whitelaw, J. H.,Springer Verlag,New York,1981

[9] Boysan,F.;Ayers, W. H.;Swithenbank, J., A Fundamental Mathematical Modelling Approach to Cyclone Design,Tran IChemE,Vol. 6Ø,1982,pp 222-23Ø

[1Ø] Boysan, F.;Weber, R.;Swithenbank, J. A.;Ayers, W. H., Mathematical Modelling of Heterogeneous Combustion in Strongly Swirling Flows,Int. Symp. on Refined Modelling of Flows,Paris,1982

[11] Boysan, F.;Weber, R.;Swithenbank, J.A.,Mathematical Modelling of an Entrained Coal Gasifier,Int. Gas Res. Conference,1983

[12] Boysan, F.;Weber, R.;Swithenbank, J.A.,Modelling Coal Fired Cyclone Combustors,Comb. Flame,Vol..63,1986,pp 73-86

[13] Bradshaw, P.,Review - Complex Turbulent Flows,J.Fluids Eng.,ASME,Vol. 97,1975,pp 146-154

[14] Bradshaw, P., Introduction, Cap. 1 em Turbulence,Ed. Bradshaw P.,Topics in Applied Physics,Vol.12,Springer Verlag,New York,1976,pp 1-43

[15] Buchhave, P. , Advances in Turbulence Measurements Techniques,em Advances in Turbulence,Eds. George,W.

K. e Arndt , R.,Hemisphere,New York,1989,pp 159-194

[16] Busnaina, A. A.;Lilley, D.G.;Numerical Simulation of Swirling Flow in a Cyclone Chamber, em Vortex Flows, ASME Symp. on Fluid Mechanics of Combustion Systems, Boulder,1981,pp 169-178

[17] Chabard, J. P.;Viollet, P.L.,Les Défis des Codes de Mécanique de Fluides pour les Années à Venir.Exemples Divers d'Application,R. Générale de Thermique,Vol.XXX, 1991,pp 552-562

[18] Crowe, C. T.;Pratt, D. T.,Analysis of the Flow Field in Cyclone Separators, Computers and Fluids,Vol. 2, 1974,pp 249-26Ø

[19] Daly, B. J.;Harlow, F.H., Transport Equations in Turbulence,Phys. Fluids,Vol. 13,197Ø,pp 2634-2649

[2Ø] Davidson, M.,Numerical Calculation of Flow in a Hydrocyclone Operating Without an Air Core,Appl. Math. Modelling,Vol. 12,April 1988, pp 119-128

[21] Doormaal,J.P.;Raithby G.D., Enhancements of the SIMPLE Method for Predicting Incompressible Fluid Flows,Num. Heat Transfer,Vol.7,1984,pp 147-163

[22] Escudier,M. P.;Bornstein, J.;Zehnder, N.,Observations and LDA Measurements of Confined Turbulent Vortex Flow,J. Fluid Mech.,Vol. 98,198Ø,pp 49-63

[23] Escudier,M. P.; Keller,J. J.,Recirculation in Swirling Flow :A Manifestation of Vortex Breakdown ,AIAA J.,Vol. 23 ,jan. 1985,pp 111-116

[24] Escudier,M. P.,Confined Vortices in Flow Machinery, Ann. Rev. Fluid Mech.,Vol 19 ,1987,pp 27-52

[25] Fu,S.;Huang,P.G.;Launder,B. E.;Leschziner,M. A., A Comparison of Algebraic and Differential Second-Moment Closures for Axisymmetric Turbulent Shear Flows With and Without Swirl,J. Fluids Eng.,ASME,Vol.11Ø, 1988,pp 216-221

[26] Gibson,M. M.;Launder, B. E.,On the Calculation of Horizontal,Turbulent,Free Shear Flows Under Gravitational Influence,J. Heat Transfer,Vol 98,Feb. 1976, pp 81-87

[27] Gibson,M. M.;Launder, B. E.,Ground Effects on Pressure Fluctuations in the Atmospheric Boundary Layers,J. Fluid Mech.,Vol. 8Ø,1978,pp 491-511

[28] Givi, P.,Model-Free Simulations of Turbulent Reactive Flows,Prog. Energy Combust. Sci.,Vol. 15,1989,pp 1-1Ø7

[29] Gosman,A. D.;Pun,W. M.;Runchal, A. K.,Spalding D. B., Wolfstein,M.,Heat and Mass Transfer in Recirculating Flows,Academic Press,Great Britain,1969

[3Ø] Gosman, A. D.;Khalil, E. E.;Whitelaw, J. H. , The Calculation of Two-Dimensional Turbulent Recirculating

Flows, Turbulent Shear Flows, Vol. I. Eds. Durst, F. ; Launder, B. E. ; Whitelaw, J. H. , Springer Verlag, New York, 1979

[31] Gupta, A. K. ; Lilley, D.G. ; Syred, N. , Swirl Flows, Abacus Press, Great Britain, 1984

[32] Han, T. ; Humphrey, J. A. C. ; Launder, B.E. , A Comparison of Hybrid and Quadratic-Upstream Differencing in High Reynolds Number Elliptic Flows, Comp. Meths. Appl. Mech. Eng. , Vol. 29, 1981, pp 81-95

[33] Hanjalic, K. ; Launder, B. E. , A Reynolds Stress Model of Turbulence and its Application to Thin Shear Flows, J. Fluid Mech. , Vol 61, 1972, pp 33-62

[34] Harlow, F. W. ; Welch, J. E. , Numerical Calculation of Time-Dependent Viscous Incompressible Flow, Phys. Fluids, Vol. 8, 1965, pp 2182

[35] Hirt, C. W. ; Nichols, B. D. ; Romero, N. C. , SOLA - A Numerical Solution Algorithm for Transient Fluid Flows, Los Alamos Scientific Laboratory , Report LA-5852, 1976

[36] Hirt, C. W. , Simplified Solution Algorithms for Fluid Flow Problems, em Numerical Methods for Partial Differential Equations, Ed. Parter S.V. , Academic Press, New York, 1979

[37] Hogg, S. ; Leschziner, M. A. , Computation of Highly Swirling Confined Flow with a Reynolds Stress

Turbulence Model, AIAA J., Vol. 27, jan 1989 , pp 57-63

[38] Holman, J. P.; Moore, G. D., An Experimental Study of Vortex Chamber Flow, J. Basic. Eng., ASME, Vol. 83, 1961, pp 632-641

[39] Hoy, H. R.; Roberts, A. G.; Wilkins, D. M., The Development of a Small Slagging Cyclone Combustor, BCURA, Document C/52Ø4, 1958

[4Ø] Jones, W. P.; Launder, B. G., The Prediction of Laminarization with a Two Equation Model of Turbulence, Int. J. Heat Mass Transfer, Vol. 15, 1972, pp 3Ø1-314

[41] Jones, W. P.; Musonge, P., Closure of the Reynolds Stress and Scalar Flux Equations, Phys. Fluids, Vol. 31, 1988, pp 3589-36Ø4

[42] Jones, W. P.; Pascau, A. , Calculation of Confined Swirling Flows With a Second Moment Closure, J. Fluids Eng., ASME, Vol. 111 , 1989, pp 248-255

[43] Karki, H. C.; Patankar, S. V.; Mongia, H. C., Three-dimensional Fluid Flow Calculation Using a Flux-Spline Method, AIAA J., Vol. 28, April 199Ø, pp 631-634

[44] Karniven, R.; Ahlstedt, H., Use of Phoenics with Modifications in Some Process Problems, em Numerical Simulation of Fluid Flow and Heat/Mass Transfer Processes, Eds. Markatos N. C. e Rhodes N., Springer-Verlag, Berlin, 1986

[45] Kelsall,D. F.,A Study of the Motion of Solid Particles in a Hydraulic Cyclone,Trans. Inst. Chem. Eng.,Vol.3Ø, 1952,pp 87-1Ø4

[46] Khalil, E.E.,Numerical Computational of Turbulent Flow Structure in a Cyclone Chamber,Joint ASME/AIChE 18th National Heat Transfer Conference,San Diego,1979,paper 79-HT-31

[47] Khodadadi, J.M.; Vlachos, N. S.,Effects of Turbulence Model Constants on Computation of Confined Swirling Flows,AIAA J.,Vol. 28,April 199Ø,pp 75Ø-752

[48] Kind, R. J.; Yowakim,F. W.; Sjolander,S. A., The Law of Wall for Swirling Flow in Annular Ducts,J. Fluids Eng.,ASME,Vol 111,1989,pp 16Ø-164

[49] Kitoh,O.,Experimental Study of Turbulent Swirling Flow in a Straigth Pipe,J. Fluid Mech.,Vol.225,1991, pp 445-479

[5Ø] Kubo,I.;Gouldin,F. C.,Numerical Calculation of Turbulent Swirling Flow,J. Fluids Eng.,ASME,Vol. 97,1975, pp 31Ø-315

[51] Lakshminarayana,B.,Turbulence Modeling for Complex Shear Flows,AIAA J.,Vol. 24 , dec. 1986,pp 19ØØ-1917

[52] Launder, B. E.;Spalding, D. B.,Mathematical Models of Turbulence,Academic Press,London,1972

[53] Launder, B. E.;Spalding, D. B.,The Numerical Computation of Turbulent Flows,Comp. Meths. Appl. Mech. Eng., Vol. 3,1974,pp 269-289

[54] Launder, B.E.;Reece, G.J.;Rodi, W.,Progress in the Development of a Reynolds-Stress Turbulence Closure,J. Fluid Mech.,Vol. 68,1975,pp 537-566

[55] Launder, B. E.;Priddin ,C. H.;Sharma, B. I.,The Calculation of Turbulent Boundary Layers on Spinning and Curved Surfaces,J. Fluids Eng.,ASME,Vol.99,1977, pp 231-239

[56] Launder, B. E.;Morse A.,Numerical Prediction of Axisymmetric Free Shear Flows with a Reynolds Stress Closure,Turbulent Shear Flows I,Eds. Schmidt, F.W. ; Whitelaw, J. H.; Durst, F. ; Launder, B. E.,Springer Verlag,1979

[57] Launder, B. E.,A Generalized Algebraic Stress Transport Hypothesis,AIAA J.,Vol. 2Ø,March 1982,pp 436-437

[58] Launder, B. E.,Progress and Prospects in Phenomenological Turbulence Models,em Theoretical Approaches to Turbulence,Eds. Dwoyer ,D.L.;Hussani ,M.Y.;Voight,R., Springer Verlag,New York,1984,pp 155-186

[59] Leonard, B. P., A Stable and Accurate Convective Modelling Procedure Based on Quadratic Upstream Interpolation,Comp. Meths. Appl. Mech. Eng.,Vol. 19, 1979,pp 58-98

[6Ø] Leschziner, M. A., Practical Evaluation of Three Finite Difference Schemes for the Computation of Steady - State Recirculating Flows, Comp. Meths. Appl. Mech. Eng., Vol. 23, 198Ø, pp 293-312

[61] Leschziner, M. A.; Rodi, W., Calculation of Annular and Twin Parallel Jets Using Various Discretization Schemes and Turbulence Model Variations, J. Fluids Eng., ASME, Vol. 1Ø3, 1981, pp 352-36Ø

[62] Leschziner, M. A.; Rodi, W., Computation of Strongly Swirling Axisymmetric Free Jets, AIAA J., Vol. 22, December 1984, pp 1742-1747

[63] Lilley, D. G.; Chigier, N.A., Nonisotropic Turbulent Stress Distribution in Swirling Flows From Mean Value Distributions, Int. J. Heat Mass Transfer, Vol. 14, 1971, pp 573-585

[64] Lilley, D. G., Nonisotropic Turbulence in Swirling Flows, Acta Astronautica, Vol. 3, 1976, pp 919-933

[65] Lilley, D. G., Primitive Pressure-Velocity Code for the Computation of Strongly Swirling Flows, AIAA J., Vol. 14, June 1976, pp 749-756

[66] Lilley, D. G.; Rhode, D.L., A Computer Code for Swirling Turbulent Axisymmetric Recirculating Flows in Practical Combustor Geometries, NASA, CR-3442, 1982

[67] Lilley, D.G., Investigations of Flowfields Found in Typical Combustor Geometries, NASA, CR-3869, 1985

[68] Lumley, J. L.; Hestad, D. E.; Morel; P., Modeling the Effect of Buoyancy and Rotation on Turbulence, Int.

Symp. on Refined Flow Modeling and Turbulence Measurements, Iowa, 1985, pp 261-270

[69] Markatos, N. C.; Rhodes, N.; Tatchell, D. G., A General Purpose Program for the Analysis of Fluid Flow Problems, em Numerical Methods for Fluid Dynamics, Eds. Morton, K.W.; Baines, M.J., Academic Press, London, 1982

[70] Markatos, N. C., The Mathematical Modelling of Turbulent Flows, Appl. Math. Modelling, Vol. 10, June 1986, pp 190-220

[71] Militzer, J.; Nicoll, W.B.; Alpay, S.A. , Some Observations on the Numerical Calculation of the Recirculation Region of Twin Parallel Symmetric Jet Flow, Symp. on Turbulent Shear Flow, Penn. State University, 1977

[72] Nogotov, E.F., Applications of Numerical Heat Transfer, McGraw-Hill, New York, 1978

[73] Patankar, S. V.; Spalding, D. B., A Calculation Procedure for Heat, Mass and Momentum Transfer in Three-Dimensional Parabolic Flows, Int. J. Heat Mass Transfer, Vol. 15, 1972, pp 1787-1806

[74] Patankar, S. V. , Numerical Heat Transfer and Fluid Flow, Hemisphere, New York, 1980

[75] Patel, M.; Cross M.; Markatos, N. C.; Mace, A. C. H. , An Evaluation of Eleven Discretization Schemes for Predicting Elliptic Flow and Heat Transfer in Supersonic Jets, Int. J. Heat Mass Transfer, Vol. 30, 1987, pp 1907-1925

[76] Pericleous, K. A., Rhodes N., The Hydrocyclone Classifier -A Numerical Approach, Int. J. Miner. Process. Vol. 17,

1986, pp 23-43

[77] Pollard A. ; Siu A. L., Laminar Flow Calculations Using a Modified Second Order Discretisation Scheme, Int. Symp. on Refined Modelling of Flows, Paris, 1982

[78] Pope, S. B. , A More General Effective Viscosity Hypothesis, J. Fluid Mech., Vol. 72, 1975, pp 331-340

[79] Pope, S. B. ; Whitelaw, J. H. , The Calculation of Near-Wake Flows, J. Fluid Mech., Vol. 73, 1976, pp 9-32

[80] Pourahmadi, F. ; Humphrey, J. A., Prediction of Curved Channel Flow with an Extended k-ε Model of Turbulence, AIAA J., Vol. 21, Oct. 1983, pp 1365-1373

[81] Raithby, G. D., A Critical Evaluation of Upstream Differencing Applied to Problems Involving Fluid Flow, Comp. Meths. Appl. Mech. Eng., Vol. 9, 1976, pp 75-103

[82] Raithby, G. D., Skew Upstream Differencing Schemes for Problems Involving Fluid Flow, Comp. Meths. Appl. Mech. Eng. , Vol. 9, 1976, pp 153-164

[83] Roache, P. J., Computational Fluid Dynamics, Hermosa, New Mexico, 1972

[84] Rodi, W., A New Algebraic Relation for Calculating the Reynolds Stresses, ZAMM, Vol. 56, 1976, pp 219-220

[85] Schlichting, H., Boundary Layer Theory , McGraw-Hill, New York, 1968

[86] Scott, C. J. ; Rask, D. R., Turbulent Viscosities for Swirling Flow in a Stationary Annulus, J. Fluids Eng.,

ASME, Vol. 95, pp 557-565

[87] Scott, C. J.; Bartelt, K. W., Decaying Annular Swirl Flow With Inlet Solid Body Rotation, J. Fluids Eng., ASME, Vol. 98 , pp 33-40

[88] Singhal, A. K.; Spalding, D.B., Predictions of Two-dimensional Boundary Layers with the Aid of the k-ε model of Turbulence, Comp. Meths. Appl. Mech. Eng., Vol 25, 1981, pp 365-383

[89] Sini, J.F.; Dekeyser, I, Numerical Prediction of Turbulent Plane Jets and Forced Plumes by Use of the k -ε Model of Turbulence, Int. J. Heat Mass Transfer, Vol. 30, 1987, pp 1787-1801

[90] Sloan, D.G.; Smith, P.J.; Smoot, L.D., Modeling of Swirl in Turbulent Flow Systems, Prog. Energy Combust. Sci., Vol. 12, 1986, pp 163-250

[91] Smith , J. L. J. , An Analysis of the Vortex Flow in the Cyclone Separator, J. Basic Eng., ASME, dez. 1962, pp 609-618

[92] Spalding, D. B., A Novel Finite Difference Formulation for Differential Expressions Involving Both First and Second Derivatives, Int. J. Num. Meths. Eng., Vol. 4, 1972, pp 551-559

[93] Srinivasan, R.; Mongia, H.C., Numerical Computations of Swirling Recirculating Flows, NASA, CR-165197, 1980

[94] Stone, H. L., Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations, SIAM J. Numer. Anal., Vol. 5, 1968,

pp 530-558

[95] Syred, N.;Beér, J.M.,Combustion in Swirling Flows: A Review,Comb. Flame,Vol. 23,1974,143,201

[96] Taulbee, D. B.,Engineering Turbulence Models, em Advances in Turbulence,Eds. George,W.K. ;Arndt,R., Hemisphere,New York,1989,pp 75-126

[97] Tager, S.A. ,Calculating the Aerodynamic Resistance of Cyclone Combustion Chambers,Teploenergetika,Vol. 18, 1971,pp 88-91

[98] Ustimenko,B. P.;Bukhman,M. A.,Turbulent Flow Structure in a Cyclone Chamber, Thermal Eng. ,Vol. 15 ,No.2,1968, pp 90-94

[99] Vastidas,G.H.;Busnaina,A.A.;Lilley,D. G.,A Simple Code for Laminar or Turbulent Flowfield Predictions with Applications,Proc.Computational Methods Conference,1986, pp 177-188

```
Line#  Source Line      Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

  1
  2  $ large
  3              program pandora
  4
  5  cØ*****************     declarations
  6
  7
  8         logical iread,ifine,irelax,iconve,iomega,ifirst,
  9        1        istone,isimplec,iupdate,iurfp,iorder,icorrp,
 1Ø        2        iold,iupwind
 11
 12         common
 13        1/uvel/resolu,nswpu,urfu,dxepu(48),dxpwu(48),sewu(48)
 14        2/vvel/resolv,nswpv,urfv,dynpv(24),dypsv(24),snsv(24)
 15        3/wvel/resolw,nswpw,urfw,iomega
 16        4/corrp/resolm,nswpp,urfp,du(48,24),dv(48,24),ipref,jpref
 17        5/tenerg/resolk,nswpk,urfk,iold
 18        6/tdissp/resole,nswpd,urfe
 19        7/var/u(48,24), v(48,24), w(48,24), p(48,24), pp(48,24), te(48,24)
 2Ø        8     ed(48,24)
 21        9/geral/it,jt,ni,nj,nim1,njm1,great,jmax(48),jmaxp1(48)
 22        1/geom/x(48),y(24),dxep(48),dxpw(48),dynp(24),dyps(24),
 23        2       sns(24),sew(48),xu(48),yv(24),r(24),rv(24),
 24        3       wfn(24),wfs(24),wfe(48),wfw(48),rcv(24),ifine
 25
 26          common
 27        1/flupr/urften,viscos,densit,prandt,den(48,24)
 28        2/causo/uin,tein,edin,flowin,alamda,
 29        3        rsmall,rlarge,xtota,jstep,istep,jstp1,jstm1,istp1,istm1,
 3Ø        4        jexit,jexitp1,inleti,inletf,jinlet,rgarg,ngarg,ngarm1,
 31        5        ngarp1
 32        6/turbu/gen(48,24),cd,cmu,c1e,c2e,betarodi,cappa,elog,
 33        7       pred,prte,c1,c2,umc2,c1m1,ck,ce3
 34        8/wallf/yplusn(48),xplusw(24),xpluse(24),taun(48),tauw(24),
 35        9        taue(24)
 36        1/coef/ap(48,24),an(48,24),as(48,24),ae(48,24),aw(48,24),su(48,24)
 37        2       sp(48,24)
 38        3/stone/tetastone,source,istone,niter,isimplec
 39
 4Ø         common
 41        1/stress/uu(48,24),vv(48,24),ww(48,24),uv(48,24),
 42        2         uw(48,24),vw(48,24)
 43        3/localstress/uvc(48,24),uwc(48,24),vwc(48,24)
 44
 45         common
 46        1/old/genold(48,24)
 47        2/wind/iupwind
 48
 49  c1*****************     parameters and control variables
 5Ø
 51         iread=.true.
 52         ifine=.false.
 53         irelax=.false.
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.Ø

```
      iomega=.true.
      ifirst=.false.
      iupdate=.true.
      icorrp=.true.

c-----------------------------------
                      iconve=.false.
c-----------------------------------

                      isimplec=.true.
                      iupwind=.false.
                      iurfp=.false.
                      istone=.true.
                      iorder=.false.
                      iold=.true.


      niter=Ø
      maxit=5ØØØ

      it=48
      jt=24
      great=1.e3Ø
      pi=3.1415927

      nswpu=2
      nswpv=2
      nswpw=2
      nswpp=4
      nswpk=2
      nswpd=2

c-------------------          domain

                    call grid

                       call geo


c-------------------          fluid properties

      densit=1.211
      viscos=1.8e-5

c-------------------          turbulent constants

      cmu=Ø.Ø9
      cd=1.ØØ
      c1e=1.44
      c2e=1.92
      cappa=Ø.4187
      elog=9.793
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
107         pred=cappa*cappa/(c2e-c1e)/(cmu**Ø.5)
108         prte=Ø.9
109         ck=Ø.22
110         ce3=Ø.15
111         c1=1.8
112         c2=Ø.6
113         umc2=1.Ø-c2
114         c1m1=c1-1.Ø
115         firodi=Ø.3
116         betarodi=firodi/umc2
117
118   c-------------------          boundary values
119
120         uin=Ø.Ø
121         vin=-Ø.36
122         win=2.16
123         turbin=Ø.Ø3
124
125   c-------------------         calculated values
126
127         tein=turbin*vin**2
128
129   c--------------------gosman
130
131         alamda=Ø.Ø1
132
133   c-------------------
134
135         edin=tein**1.5/(alamda*rlarge)
136
137   c-------------------          pressure reference
138
139         ipref=inleti
140         jpref=jmax(inleti)
141
142   c-------------------          program control
143
144        sormax=Ø.ØØ4
145
146
147   c2*****************     INITIAL OPERATIONS
148
149
150   c-------------------          set variables to initial value
151
152          do 2ØØ i=1,ni
153          taun(i)=1.Ø
154          yplusn(i)=Ø.Ø
155
156            do 2ØØ j=1,jmaxp1(i)
157
158            tauw(j)=1.Ø
159            taue(j)=1.Ø
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
 160            xplusw(j)=Ø.Ø
 161            xpluse(j)=Ø.Ø
 162
 163
 164            u(i,j)=Ø.Ø
 165            v(i,j)=Ø.Ø
 166            w(i,j)=Ø.Ø
 167            p(i,j)=Ø.Ø
 168            pp(i,j)=Ø.Ø
 169            te(i,j)=Ø.Ø
 170            ed(i,j)=Ø.Ø
 171            den(i,j)=densit
 172            du(i,j)=Ø.Ø
 173            dv(i,j)=Ø.Ø
 174            su(i,j)=Ø.Ø
 175            sp(i,j)=Ø.Ø
 176            uu(i,j)=Ø.Ø
 177            vv(i,j)=Ø.Ø
 178            ww(i,j)=Ø.Ø
 179            uv(i,j)=Ø.Ø
 180            uw(i,j)=Ø.Ø
 181            vw(i,j)=Ø.Ø
 182
 183  2ØØ       continue
 184
 185           flowin=Ø.Ø
 186           arden=Ø.Ø
 187           ardens=Ø.Ø
 188           xmonin=Ø.Ø
 189           ymonin=Ø.Ø
 190           wmonin=Ø.Ø
 191           angmom=Ø.Ø
 192           camonin=Ø.Ø
 193
 194
 195
 196  c--------------------          initialize
 197
 198           do 2Ø5 iin=inleti,inletf
 199
 2ØØ              u(iin,nj)=uin
 2Ø1              v(iin,nj)=vin
 2Ø2              w(iin,nj)=win
 2Ø3              te(iin,nj)=tein
 2Ø4              ed(iin,nj)=edin
 2Ø5              uu(iin,nj)=2.*tein/3.
 2Ø6              vv(iin,nj)=2.*tein/3.
 2Ø7              ww(iin,nj)=2.*tein/3.
 2Ø8
 2Ø9
 21Ø         arden=Ø.5*(den(iin,njm1)+den(iin,nj))*rv(jinlet+1)*sew(iin)
 211         ardens=ardens+arden
 212         flowin=flowin+arden*abs(v(iin,nj))
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
213           ymonin=ymonin+arden*v(iin,nj)*v(iin,nj)
214           wmonin=wmonin+arden*abs(v(iin,nj))*w(iin,nj)
215           angmom=angmom+arden*abs(v(iin,nj))*w(iin,nj)*rv(jinlet)
216
217   2Ø5       continue
218
219              if (win .lt. 1.e-3Ø ) wmonin=1.Ø
22Ø
221           umeancore=2*flowin/(den(inleti,nj)*rv(nj)*rv(nj))
222           umeanthroat=2*flowin/(den(inleti,nj)*rv(jexitp1)*rv(jexitp1))
223           xmonin=den(inleti,nj)*rv(nj)**2/2*umeancore*umeancore
224           xmonthr=den(inleti,nj)*rgarg**2/2*umeanthroat*umeanthroat
225           camonin=Ø.5*flowin*vin*vin
226
227           re=umeancore*rlarge*2.Ø*densit/viscos
228
229           swirlind=angmom/(xmonthr*rgarg)
23Ø
231
232                 if ( ifirst ) then
233
234                       sai1=2Ø.*tein/3.
235                       sai2=edin/1Ø.
236                       do 22Ø i=2,nim1
237                       do 22Ø j=2,jmax(i)
238                       te(i,j)=tein*1Ø
239                       gen(i,j)=1.e-4
24Ø                       ed(i,j)=sai2
241                       uu(i,j)=sai1
242                       uvc(i,j)=Ø.
243                       uwc(i,j)=Ø.
244                       vv(i,j)=sai1
245                       vwc(i,j)=Ø.
246                       ww(i,j)=sai1
247   22Ø                 continue
248
249                     do 221 i=3,ngarm1
25Ø                     do 221 j=2,jexit
251                     u(i,j)=umeancore+(umeanthroat-umeancore)*
252         1           float(i-3)/float(ngarm1-3)
253   221               continue
254
255                     do 222 i=ngarg,nim1
256                     do 222 j=2,jexit
257                     u(i,j)=umeanthroat
258   222               continue
259
26Ø                     do 223 i=3,ngarm1
261                     jj=jmax(i)
262                     do 223 j=jexitp1,jj
263                     u(i,j)=umeancore
264   223               continue
265
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
                        dumpet=Ø.15

                  do 224 i=2,nim1
                  do 224 j=2,jexit-2
                   w(i,j)=dumpet*win*r(j)/rgarg*float(i)/float(nim1)
  224             continue

                  do 225 i=2,nim1
                  jj=jmax(i)
                  do 225 j=jexit-1,jj
                   w(i,j)=dumpet*win*(1-r(j)/rlarge)*float(i)/float(nim1)
  225             continue

                  do 226 i=2,nim1
                  do 226 j=2,jmax(i)
                    p(i,j)=p(inleti,jinlet)-densit*u(i,j)*u(i,j)/2.
  226             continue


                   else

                       continue

                       end if

  c---------------- wall adimensional functions-initial value

                 do 23Ø i=2,nim1
                  yplusn(i)=11.Ø
  23Ø            continue

                         do 234 j=jstep,nj
                         xplusw(j)=11.Ø
                            if(j .eq. jstep) xplusw(j)=Ø.Ø
  234                    continue

                               do 235 j=jexit,nj
                               xpluse(j)=11.Ø
                                 if (j  .eq.  jexit) xpluse(j)=Ø.Ø
  235                          continue


  c------------------           initial urf's


                 if ( iurfp ) then
                 urfp=Ø.7
                 else
                 urfp=1.Ø
                 end if


  c------------------           initial output
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
319
320         ik=jmaxp1(istep)
321         yslope=yv(ik)-yv(jstp1)
322         if(jmax(istep) .lt. njm1) alpha=atan(yslope/sewu(2))*180./pi
323         if((jstep .lt. njm1) .and. (jmax(istep) .eq. njm1)) alpha=90.
324
325          write(*,*)'  alpha         ',alpha
326          write(*,*)'  rsmall        ',rsmall
327          write(*,*)'  rgarg         ',rgarg
328          write(*,*)'  rlarge        ',rlarge
329          write(*,*)'  xtota         ',xtota
330          write(*,*)'  re            ',re
331          write(*,*)'  umeancore     ',umeancore
332          write(*,*)'  umeanthroat   ',umeanthroat
333          write(*,*)'  swirlind      ',swirlind
334
335
336           if( iread ) then
337                 open(unit=8,file='partida.doc',form='formatted')
338
339                 do 240 i=2,ni
340                 do 240 j=2,jt
341                 read(8,*) itu,jtu,u(i,j),v(i,j),w(i,j),p(i,j)
342   240           continue
343
344
345                 do 241 i=2,ni
346                 do 241 j=2,jt
347                 read(8,*) itu,jtu,te(i,j),ed(i,j),gen(i,j),uu(i,j)
348
349   241           continue
350
351                 do 242 i=2,ni
352                 do 242 j=2,jt
353                 read(8,*) itu,jtu,vv(i,j),ww(i,j),uv(i,j),uw(i,j)
354   242           continue
355
356
357               do 243 i=2,ni
358               do 243 j=2,jt
359               read(8,*)itu,jtu,vw(i,j),uvc(i,j),uwc(i,j),vwc(i,j)
360   243         continue
361
362                do 244 i=2,ni
363                read(8,*) taun(i),yplusn(i)
364   244          continue
365
366                do 245 j=2,jt
367                read(8,*) tauw(j),xplusw(j),taue(j),xpluse(j)
368   245          continue
369
370
371                close (unit=8)
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
372
373
374                winold=w(inleti,nj)
375                vinold=v(inleti,nj)
376
377                do 249 iin=inleti,inletf
378
379                u(iin,nj)=uin
380                v(iin,nj)=vin
381                w(iin,nj)=win
382                te(iin,nj)=tein
383                ed(iin,nj)=edin
384                uu(iin,nj)=2.*tein/3.
385                vv(iin,nj)=2.*tein/3.
386                ww(iin,nj)=2.*tein/3.
387
388  249           continue
389
390
391                   if ( ifirst ) then
392
393                   dumpet=Ø.15
394                   do 25Ø i=2,nim1
395                   do 25Ø j=2,jexit-1
396                   w(i,j)=dumpet*win*r(j)/rgarg*float(i)/float(nim1)
397  25Ø              continue
398
399                   do 255 i=2,ngarm1
4ØØ                   jj=jmax(i)
4Ø1                   do 255 j=jexit,jj
402                   w(i,j)=dumpet*win*(1-r(j)/rlarge)*float(i)/float(nim1)
403  255              continue
4Ø4
4Ø5                   end if
4Ø6
4Ø7                         if ( iupdate ) then
4Ø8                         if (winold  .lt. 1e-5) winold=1.Ø
4Ø9                         relac=win/winold
41Ø
411                                 totp=Ø.Ø
412                                 do 257 j=2,9
413                                 totp=totp+p(23,j)*rcv(j)*sns(j)*2.*pi
414  257                            continue
415                                 totp=totp/(pi*rgarg*rgarg)
416
417                           write(*,*)'     pressao media saida',totp
418
419
42Ø                    if (   .not. ifirst ) then
421
422
423                                 do 258 i=2,nim1
424                                 do 258 j=2,jmax(i)
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
                                        w(i,j)=w(i,j)*relac**Ø.8
258                                     continue


                    end if

                    end if
                          if ( icorrp ) then

c-----------------------------------------pressure correction

             corrp=-32.

c-------------------------------------------------------------
                                        corrp=(corrp-totp)

                                        do 26Ø i=2,nim1
                                        do 26Ø j=2,jexit

                       if ( i .le. ngarg ) then
                    p(i,j)=p(i,j)+corrp*(2.-float(i)/float(ngarg))
                       else
                    p(i,j)=p(i,j)-corrp*(float(i)/float(ngarg))
                    end if


26Ø                                     continue


                    do 29Ø j=2,jexit
                    p(ni,j)=p(nim1,j)
29Ø                 continue


                                  end if
        end if

          do 299 i=2,ni
          do 299 j=2,jmax(i)
          genold(i,j)=gen(i,j)
299       continue


c3*****************    ITERATION LOOP


             write(*,*)'       ITERATION LOOP'


3ØØ   continue
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
478         niter=niter+1
479
480         write(*,*)'     step',niter
481
482               do 31Ø i=2,nim1
483               do 31Ø j=2,jmax(i)
484               pp(i,j)=Ø.
485  31Ø          continue
486
487                  if( irelax ) then
488
489  c-------------------     INCREASE URF'S AS CONVERGENCE NEARTS
490Ø
491            urfu=.Ø5+(float(niter))*((.15-.Ø5)/1ØØ.)
492            if (urfu .gt. .15) urfu=.15
493
494            urfv=.Ø2+(float(niter))*((.12-.Ø2)/1ØØ.)
495            if (urfv .gt. .12) urfv=.12
496
497            urfw=.Ø5+(float(niter))*((.15-.Ø5)/1ØØ.)
498            if (urfw .gt. .15) urfw=.15
499
5ØØ            urfk=.Ø5+(float(niter))*((.2Ø-.Ø5)/1ØØ.)
5Ø1            if (urfk .gt. .2Ø) urfk=.2Ø
5Ø2
5Ø3            urfe=.Ø5+(float(niter))*((.2Ø-.Ø5)/1ØØ.)
5Ø4            if (urfe .gt. .2Ø ) urfe=.2Ø
5Ø5
5Ø6            urften=1.
5Ø7
5Ø8
5Ø9                 else
51Ø
511  c-------------------    constant urf's
512  c--------------------ok para ciclonØØ-----
513
514
515
516             urfu=Ø.1Ø
517             urfv=Ø.Ø8
518             urfw=Ø.1Ø
519             urfk=Ø.15
52Ø             urfe=Ø.15
521
522             urften=Ø.75
523
524
525                 end if
526
527
528
529  c-------------------          update main dependent variables
530
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
531
532                     if ( iorder ) then
533
534
535                  call calcu
536                     call calcv
537                        call calcp
538                           call calcw
539                              call calcte
540                                 call calced
541                                    call props
542
543                     else
544
545
546                  call calcu
547                     call calcw
548                        call calcv
549                           call calcp
550                              call calcte
551                                 call calced
552                                    call props
553
554
555                     end if
556
557
558
559   c-------------------          intermediate output
560
561         resolm=resolm/flowin
562         resolu=resolu/xmonin
563         resolv=resolv/xmonin
564         resolw=resolw/wmonin
565         resolk=resolk/camonin
566
567   c-------------------          termination tests
568
569         source=amax1(resolm,resolu,resolv,resolw,resolk)
570
571             write(*,*)'  source ',source
572             write(*,*)'  resolm',resolm
573             write(*,*)'  resolu',resolu
574             write(*,*)'  resolv',resolv
575             write(*,*)'  resolw',resolw
576             write(*,*)'  resolk',resolk
577
578         if( niter .lt. maxit) then
579
580               if( niter .ge. 15Ø    .and. source .ge. 4.Ø) then
581               write(*,*) '    THE SOLUTION IS NOT CONVERGING - source val.
582               stop
583               end if
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
584
585               if( niter .lt. 1Ø .or. source .gt. sormax) then
586               go to 3ØØ
587               end if
588
589               if( niter .ge. 1Ø .and. source .le. sormax) then
590               write(*,*) '    THE SOLUTION CONVERGED '
591               iconve=.true.
592               end if
593
594
595
596         else
597
598               write(*,*) '    THE SOLUTION IS NOT CONVERGING - maxit '
599
6ØØ         end if
6Ø1
6Ø2
6Ø3
6Ø4   c4*****************    OUTPUT
6Ø5
6Ø6               if (.not. iconve ) then
6Ø7               stop
6Ø8               end if
6Ø9
61Ø         write(*,*)'      WRITING IN THE RESULT.DOC'
611
612   4Ø5       open(unit=8,file='result.doc',form='formatted')
613         write(8,4ØØØ)
614         write(8,4Ø1Ø)niter,source
615         write(8,4Ø2Ø)
616               do 41Ø i=2,ni
617               do 41Ø j=2,jt
618               write(8,4Ø3Ø)i,j,u(i,j),v(i,j),w(i,j),p(i,j)
619   41Ø         continue
62Ø
621         write(8,4Ø4Ø)
622                do 42Ø i=2,ni
623                do 42Ø j=2,jt
624                write(8,4Ø5Ø)i,j,te(i,j),ed(i,j),gen(i,j),uu(i,j)
625   42Ø          continue
626
627                do 425 i=2,ni
628                do 425 j=2,jt
629                write(8,4Ø5Ø)i,j,vv(i,j),ww(i,j),uv(i,j),uw(i,j)
63Ø   425          continue
631
632                do 427 i=2,ni
633                do 427 j=2,jt
634                write(8,4Ø55)i,j,vw(i,j),uvc(i,j),uwc(i,j),vwc(i,j)
635   427          continue
636
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
637                 do 43Ø i=2,ni
638                 write(8,4Ø6Ø)taun(i),yplusn(i)
639   43Ø           continue
64Ø
641                 do 44Ø j=2,jt
642                 write(8,4Ø7Ø)tauw(j),xplusw(j),taue(j),xpluse(j)
643   44Ø           continue
644
645           close(unit=8)
646
647           stop
648
649   cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
650
651   4ØØØ    format(4x1Øh1TERATIONS,6x6hSOURCE)
652   4Ø1Ø    format(7x,i4,1Øx,e16.6)
653   4Ø2Ø    format(5x1hI,5x1hJ,1Øx1hU,15x1hV,15x1hW,15x1hP)
654   4Ø3Ø    format(2xi4,2xi4,1p4e16.6)
655   4Ø4Ø    format(4x21hTURBULENCE QUANTITIES)
656   4Ø5Ø    format(2xi4,2xi4,1p4e16.6)
657   4Ø55    format(2xi4,2xi4,1p4e16.6)
658   4Ø6Ø    format(1p2e16.6)
659   4Ø7Ø    format(1p4e16.6)
66Ø           end
```

main Local Symbols

Name	Class	Type	Size	Offset
MAXIT	local	INTEGER*4	4	ØØØ2
ICONVE.	local	LOGICAL*4	4	ØØØ6
CAMONIN	local	REAL*4	4	ØØØa
IORDER.	local	LOGICAL*4	4	ØØØe
IRELAX.	local	LOGICAL*4	4	ØØ12
CORRP	local	REAL*4	4	ØØ16
IURFP	local	LOGICAL*4	4	ØØ1a
TOTP.	local	REAL*4	4	ØØ1e
UMEANTHROAT	local	REAL*4	4	ØØ22
I	local	INTEGER*4	4	ØØ26
J	local	INTEGER*4	4	ØØ2a
VINOLD.	local	REAL*4	4	ØØ2e
IUPDATE	local	LOGICAL*4	4	ØØ32
WINOLD.	local	REAL*4	4	ØØ36
SAI1.	local	REAL*4	4	ØØ3a
DUMPET.	local	REAL*4	4	ØØ3e
SAI2.	local	REAL*4	4	ØØ42
ICORRP.	local	LOGICAL*4	4	ØØ46
IFIRST.	local	LOGICAL*4	4	ØØ4a
IK.	local	INTEGER*4	4	ØØ4e
JJ.	local	INTEGER*4	4	ØØ52
TURBIN.	local	REAL*4	4	ØØ56
RE.	local	REAL*4	4	ØØ5a

main Local Symbols

Name	Class	Type	Size	Offset
WMONIN.	local	REAL*4	4	Ø05e
XMONIN.	local	REAL*4	4	Ø062
PI.	local	REAL*4	4	Ø066
YMONIN.	local	REAL*4	4	Ø06a
SORMAX.	local	REAL*4	4	Ø06e
YSLOPE.	local	REAL*4	4	Ø072
UMEANCORE	local	REAL*4	4	Ø076
IIN	local	INTEGER*4	4	Ø07a
IREAD	local	LOGICAL*4	4	Ø07e
ALPHA	local	REAL*4	4	Ø082
RELAC	local	REAL*4	4	Ø086
ARDEN	local	REAL*4	4	Ø08a
SWIRLIND.	local	REAL*4	4	Ø08e
VIN	local	REAL*4	4	Ø092
WIN	local	REAL*4	4	Ø096
XMONTHR	local	REAL*4	4	Ø09a
ITU	local	INTEGER*4	4	Ø09e
JTU	local	INTEGER*4	4	ØØa2
ARDENS.	local	REAL*4	4	ØØa6
FIRODI.	local	REAL*4	4	ØØaa
ANGMOM.	local	REAL*4	4	ØØae
NGARP1.	CAUSO	INTEGER*4	4	Ø058
U	VAR	REAL*4	46Ø8	ØØØØ
V	VAR	REAL*4	46Ø8	12ØØ
GEN	TURBU	REAL*4	46Ø8	ØØØØ
W	VAR	REAL*4	46Ø8	24ØØ
P	VAR	REAL*4	46Ø8	36ØØ
CD.	TURBU	REAL*4	4	12ØØ
CMU	TURBU	REAL*4	4	1204
PP.	VAR	REAL*4	46Ø8	48ØØ
C1E	TURBU	REAL*4	4	12Ø8
TE.	VAR	REAL*4	46Ø8	5aØØ
C2E	TURBU	REAL*4	4	12Øc
ED.	VAR	REAL*4	46Ø8	6cØØ
BETARODI.	TURBU	REAL*4	4	121Ø
CAPPA	TURBU	REAL*4	4	1214
IT.	GERAL	INTEGER*4	4	ØØØØ
ELOG.	TURBU	REAL*4	4	1218
JT.	GERAL	INTEGER*4	4	ØØØ4
PRED.	TURBU	REAL*4	4	121c
NI.	GERAL	INTEGER*4	4	ØØØ8
PRTE.	TURBU	REAL*4	4	122Ø
NJ.	GERAL	INTEGER*4	4	ØØØc
C1.	TURBU	REAL*4	4	1224
NIM1.	GERAL	INTEGER*4	4	ØØ1Ø
C2.	TURBU	REAL*4	4	1228
NJM1.	GERAL	INTEGER*4	4	ØØ14
GREAT	GERAL	REAL*4	4	ØØ18

Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

main Local Symbols

Name	Class	Type	Size	Offset
UMC2	TURBU	REAL*4	4	122c
C1M1	TURBU	REAL*4	4	123Ø
JMAX	GERAL	INTEGER*4	192	ØØ1c
JMAXP1	GERAL	INTEGER*4	192	ØØdc
IFINE	GEOM	LOGICAL*4	4	Ø9ØØ
CK	TURBU	REAL*4	4	1234
CE3	TURBU	REAL*4	4	1238
X	GEOM	REAL*4	192	ØØØØ
IOMEGA	WVEL	LOGICAL*4	4	ØØØc
YPLUSN	WALLF	REAL*4	192	ØØØØ
Y	GEOM	REAL*4	96	ØØc Ø
XPLUSW	WALLF	REAL*4	96	ØØc Ø
DXEP	GEOM	REAL*4	192	Ø12Ø
XPLUSE	WALLF	REAL*4	96	Ø12Ø
ISTONE	STONE	LOGICAL*4	4	ØØØ8
DXPW	GEOM	REAL*4	192	Ø1eØ
DYNP	GEOM	REAL*4	96	Ø2aØ
TAUN	WALLF	REAL*4	192	Ø18Ø
ISIMPLEC	STONE	LOGICAL*4	4	ØØ1Ø
TAUW	WALLF	REAL*4	96	Ø24Ø
DYPS	GEOM	REAL*4	96	Ø3ØØ
SNS	GEOM	REAL*4	96	Ø36Ø
TAUE	WALLF	REAL*4	96	Ø2aØ
SEW	GEOM	REAL*4	192	Ø3cØ
XU	GEOM	REAL*4	192	Ø48Ø
AP	COEF	REAL*4	46Ø8	ØØØØ
YV	GEOM	REAL*4	96	Ø54Ø
IOLD	TENERG	LOGICAL*4	4	ØØØc
AN	COEF	REAL*4	46Ø8	12ØØ
IUPWIND	WIND	LOGICAL*4	4	ØØØØ
AS	COEF	REAL*4	46Ø8	24ØØ
R	GEOM	REAL*4	96	Ø5aØ
RV	GEOM	REAL*4	96	Ø6ØØ
AE	COEF	REAL*4	46Ø8	36ØØ
WFN	GEOM	REAL*4	96	Ø66Ø
RESOLU	UVEL	REAL*4	4	ØØØØ
AW	COEF	REAL*4	46Ø8	48ØØ
WFS	GEOM	REAL*4	96	Ø6cØ
SU	COEF	REAL*4	46Ø8	5aØØ
NSWPU	UVEL	INTEGER*4	4	ØØØ4
SP	COEF	REAL*4	46Ø8	6cØØ
WFE	GEOM	REAL*4	192	Ø72Ø
URFU	UVEL	REAL*4	4	ØØØ8
WFW	GEOM	REAL*4	192	Ø7eØ
DXEPU	UVEL	REAL*4	192	ØØØc
TETASTONE	STONE	REAL*4	4	ØØØØ
RCV	GEOM	REAL*4	96	Ø8aØ
DXPWU	UVEL	REAL*4	192	ØØcc

main Local Symbols

Name	Class	Type	Size	Offset
SOURCE.	STONE	REAL*4	4	ØØØ4
SEWU.	UVEL	REAL*4	192	Ø18c
URFTEN.	FLUPR	REAL*4	4	ØØØØ
NITER	STONE	INTEGER*4	4	ØØØc
RESOLV.	VVEL	REAL*4	4	ØØØØ
VISCOS.	FLUPR	REAL*4	4	ØØØ4
UU.	STRESS	REAL*4	46Ø8	ØØØØ
NSWPV	VVEL	INTEGER*4	4	ØØØ4
DENSIT.	FLUPR	REAL*4	4	ØØØ8
VV.	STRESS	REAL*4	46Ø8	12ØØ
PRANDT.	FLUPR	REAL*4	4	ØØØc
URFV.	VVEL	REAL*4	4	ØØØ8
WW.	STRESS	REAL*4	46Ø8	24ØØ
DEN	FLUPR	REAL*4	46Ø8	ØØ1Ø
DYNPV	VVEL	REAL*4	96	ØØØc
UV.	STRESS	REAL*4	46Ø8	36ØØ
DYPSV	VVEL	REAL*4	96	ØØ6c
UIN	CAUSO	REAL*4	4	ØØØØ
UW.	STRESS	REAL*4	46Ø8	48ØØ
SNSV.	VVEL	REAL*4	96	ØØcc
TEIN.	CAUSO	REAL*4	4	ØØØ4
VW.	STRESS	REAL*4	46Ø8	5aØØ
EDIN.	CAUSO	REAL*4	4	ØØØ8
RESOLW.	WVEL	REAL*4	4	ØØØØ
UVC	LOCALST	REAL*4	46Ø8	ØØØØ
NSWPW	WVEL	INTEGER*4	4	ØØØ4
FLOWIN.	CAUSO	REAL*4	4	ØØØc
UWC	LOCALST	REAL*4	46Ø8	12ØØ
ALAMDA.	CAUSO	REAL*4	4	ØØ1Ø
URFW.	WVEL	REAL*4	4	ØØØ8
VWC	LOCALST	REAL*4	46Ø8	24ØØ
RSMALL.	CAUSO	REAL*4	4	ØØ14
RLARGE.	CAUSO	REAL*4	4	ØØ18
RESOLM.	CORRP	REAL*4	4	ØØØØ
GENOLD.	OLD	REAL*4	46Ø8	ØØØØ
NSWPP	CORRP	INTEGER*4	4	ØØØ4
XTOTA	CAUSO	REAL*4	4	ØØ1c
URFP.	CORRP	REAL*4	4	ØØØ8
JSTEP	CAUSO	INTEGER*4	4	ØØ2Ø
DU.	CORRP	REAL*4	46Ø8	ØØØc
ISTEP	CAUSO	INTEGER*4	4	ØØ24
JSTP1	CAUSO	INTEGER*4	4	ØØ28
DV.	CORRP	REAL*4	46Ø8	12Øc
IPREF	CORRP	INTEGER*4	4	24Øc
JSTM1	CAUSO	INTEGER*4	4	ØØ2c
JPREF	CORRP	INTEGER*4	4	241Ø
ISTP1	CAUSO	INTEGER*4	4	ØØ3Ø
ISTM1	CAUSO	INTEGER*4	4	ØØ34

main Local Symbols

Name	Class	Type	Size	Offset
RESOLK.	TENERG	REAL*4	4	ØØØØ
JEXIT	CAUSO	INTEGER*4	4	ØØ38
NSWPK	TENERG	INTEGER*4	4	ØØØ4
JEXITP1	CAUSO	INTEGER*4	4	ØØ3c
URFK.	TENERG	REAL*4	4	ØØØ8
INLETI.	CAUSO	INTEGER*4	4	ØØ4Ø
INLETF.	CAUSO	INTEGER*4	4	ØØ44
RESOLE.	TDISSP	REAL*4	4	ØØØØ
JINLET.	CAUSO	INTEGER*4	4	ØØ48
RGARG	CAUSO	REAL*4	4	ØØ4c
NSWPD	TDISSP	INTEGER*4	4	ØØØ4
URFE.	TDISSP	REAL*4	4	ØØØ8
NGARG	CAUSO	INTEGER*4	4	ØØ5Ø
NGARM1.	CAUSO	INTEGER*4	4	ØØ54

Global Symbols

Name	Class	Type	Size	Offset
CALCED.	extern	***	***	***
CALCP	extern	***	***	***
CALCTE.	extern	***	***	***
CALCU	extern	***	***	***
CALCV	extern	***	***	***
CALCW	extern	***	***	***
CAUSO	common	***	92	ØØØØ
COEF.	common	***	32256	ØØØØ
CORRP	common	***	9236	ØØØØ
FLUPR	common	***	4624	ØØØØ
GEO	extern	***	***	***
GEOM.	common	***	2308	ØØØØ
GERAL	common	***	412	ØØØØ
GRID.	extern	***	***	***
LOCALSTRESS	common	***	13824	ØØØØ
OLD	common	***	4608	ØØØØ
PROPS	extern	***	***	***
STONE	common	***	2Ø	ØØØØ
STRESS.	common	***	27648	ØØØØ
TDISSP.	common	***	12	ØØØØ
TENERG.	common	***	16	ØØØØ
TURBU	common	***	4668	ØØØØ
UVEL.	common	***	588	ØØØØ
VAR	common	***	32256	ØØØØ
VVEL.	common	***	3ØØ	ØØØØ
WALLF	common	***	768	ØØØØ
WIND.	common	***	4	ØØØØ
WVEL.	common	***	16	ØØØØ

Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

Global Symbols

Name	Class	Type	Size	Offset
main.	FSUBRT	***	***	ØØØØ

Code size = 2e5d (11869)
Data size = Ø41b (1Ø51)
Bss size = ØØb2 (178)

No errors detected

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
 1
 2             subroutine grid
 3
 4  cØ*******************      preliminaires
 5
 6        logical ifine
 7
 8        common
 9       1/geral/it,jt,ni,nj,nim1,njm1,great,jmax(48),jmaxp1(48)
10       2/geom/x(48),y(24),dxep(48),dxpw(48),dynp(24),dyps(24),
11       3      sns(24),sew(48),xu(48),yv(24),r(24),rv(24),
12       4      wfn(24),wfs(24),wfe(48),wfw(48),rcv(24),ifine
13
14       5/causo/uin,tein,edin,flowin,alamda,
15       6       rsmall,rlarge,xtota,jstep,istep,jstp1,jstm1,istp1,istm1,
16       7       jexit,jexitp1,inleti,inletf,jinlet,rgarg,ngarg,ngarm1,
17       8       ngarp1
18
19  c1*******************     definition
20
21             istep=2
22             jstep=1
23
24                   nj=21
25                   jexit=9
26                   jinlet=2Ø
27
28                istp1=istep+1
29                istm1=istep-1
30                jstp1=jstep+1
31                jstm1=jstep-1
32                njm1=nj-1
33                jexitp1=jexit+1
34
35
36             if (ifine) then
37
38  c----------------------     fine mesh grid lines in x-direction
39
40             inleti=2
41             inletf=7
42
43             ni=3Ø
44             nim1=ni-1
45             epsx=1.Ø9Ø122
46             dx=7.71785e-3
47
48             x(1)=-Ø.5*dx
49             x(2)=Ø.5*dx
50                  do 1Ø5 i=3,16
51                  x(i)=x(i-1)+dx
52                  dx=epsx*dx
53  1Ø5             continue
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
 54              dx=dx/epsx
 55
 56              do 11Ø i=17,nim1
 57              dx=dx/epsx
 58              x(i)=x(i-1)+dx
 59  11Ø         continue
 6Ø
 61              x(ni)=x(nim1)+(x(nim1)-x(nim1-1))
 62              xtota=(x(nim1)+x(ni))/2.
 63
 64              ngarg=ni
 65              ngarm1=ngarg-1
 66              ngarp1=ngarg+1
 67              ni=43
 68              nim1=ni-1
 69              dx=x(ngarg)-x(ngarm1)
 7Ø              epsx=1.2
 71
 72                    do 115 i=ngarp1,ni
 73                    dx=dx*epsx
 74                    x(i)=x(i-1)+dx
 75  115               continue
 76
 77
 78
 79              else
 8Ø
 81  c---------------------    coarse mesh grid lines in x-direction
 82
 83              inleti=2
 84              inletf=6
 85
 86              ni=23
 87              nim1=ni-1
 88              epsx=1.16693
 89              dx=8.55567e-3
 9Ø
 91              x(1)=-Ø.5*dx
 92              x(2)=Ø.5*dx
 93
 94                   do 12Ø i=3,12
 95                   x(i)=x(i-1)+dx
 96                   dx=epsx*dx
 97  12Ø              continue
 98
 99                   do 125 i=13,nim1
1ØØ                  dx=dx/epsx
 1Ø1                 x(i)=x(i-1)+dx
 1Ø2 125             continue
 1Ø3
 1Ø4             x(ni)=x(nim1)+(x(nim1)-x(nim1-1))
 1Ø5             xtota=(x(nim1)+x(ni))/2.
 1Ø6
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
 1Ø7              ngarg=ni
 1Ø8              ngarm1=ngarg-1
 1Ø9              ngarp1=ngarg+1
 11Ø              ni=36
 111              nim1=ni-1
 112              dx=x(ngarg)-x(ngarm1)
 113              epsx=1.2
 114
 115                    do 13Ø i=ngarp1,ni
 116                    dx=dx*epsx
 117                    x(i)=x(i-1)+dx
 118  13Ø               continue
 119
 12Ø             end if
 121
 122
 123  c---------------------    no. of j-cells of wall
 124  c---------------------    for each i grid line
 125
 126        jmax(1)=jstep
 127        jmaxp1(1)=jmax(1)+1
 128
 129           do 15Ø i=2,ngarm1
 13Ø
 131  c---------------------    provision for expansion
 132
 133           jmax(i)=jmax(i-1)+19
 134           if(jmax(i-1) .eq. njm1)    jmax(i)=jmax(i-1)
 135           jmaxp1(i)=jmax(i)+1
 136  15Ø      continue
 137
 138
 139           do 16Ø i=ngarg,ni
 14Ø           jmax(i)=jexit
 141           jmaxp1(i)=jmax(i)+1
 142  16Ø      continue
 143
 144  c---------------------    grid lines in y direction
 145
 146            epsy=1.18109
 147            dy=3.5e-3
 148
 149            y(1)=-Ø.5*dy
 15Ø            y(2)=Ø.5*dy
 151
 152            do 17Ø j=3,1Ø
 153            y(j)=y(j-1)+dy
 154            dy=dy*epsy
 155  17Ø       continue
 56
 157                 y(11)=y(1Ø)+(y(1Ø)-y(9))
 158
 159            epsy=1.14Ø5Ø12
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
16Ø               dy=3.5e-3
161
162               do 18Ø j=12,njm1
163               y(j)=y(j-1)+epsy**(njm1-j)*dy
164  18Ø          continue
165
166               y(nj)=y(njm1)+(y(njm1)-y(njm1-1))
167
168               rsmall=Ø.5*(y(jstep)+y(jstp1))
169               rlarge=Ø.5*(y(nj)+y(njm1))
17Ø               rgarg=Ø.5*(y(jexit)+y(jexitp1))
171
172                     return
173
174                        end
```

GRID Local Symbols

Name	Class	Type	Size	Offset
EPSX	local	REAL*4	4	ØØØØ
EPSY	local	REAL*4	4	ØØØ4
I	local	INTEGER*4	4	ØØØ8
J	local	INTEGER*4	4	ØØØc
DX	local	REAL*4	4	ØØ1Ø
DY	local	REAL*4	4	ØØ14
NGARG	CAUSO	INTEGER*4	4	ØØ5Ø
NGARM1	CAUSO	INTEGER*4	4	ØØ54
NGARP1	CAUSO	INTEGER*4	4	ØØ58
IFINE	GEOM	LOGICAL*4	4	Ø9ØØ
IT	GERAL	INTEGER*4	4	ØØØØ
JT	GERAL	INTEGER*4	4	ØØØ4
NI	GERAL	INTEGER*4	4	ØØØ8
NJ	GERAL	INTEGER*4	4	ØØØc
NIM1	GERAL	INTEGER*4	4	ØØ1Ø
NJM1	GERAL	INTEGER*4	4	ØØ14
GREAT	GERAL	REAL*4	4	ØØ18
JMAX	GERAL	INTEGER*4	192	ØØ1c
JMAXP1	GERAL	INTEGER*4	192	ØØdc
X	GEOM	REAL*4	192	ØØØØ
Y	GEOM	REAL*4	96	ØØc Ø
DXEP	GEOM	REAL*4	192	Ø12Ø
DXPW	GEOM	REAL*4	192	Ø1eØ
DYNP	GEOM	REAL*4	96	Ø2aØ
DYPS	GEOM	REAL*4	96	Ø3ØØ
SNS	GEOM	REAL*4	96	Ø36Ø
SEW	GEOM	REAL*4	192	Ø3c Ø
XU	GEOM	REAL*4	192	Ø48Ø
YV	GEOM	REAL*4	96	Ø54Ø
R	GEOM	REAL*4	96	Ø5aØ
RV	GEOM	REAL*4	96	Ø6ØØ
WFN	GEOM	REAL*4	96	Ø66Ø
WFS	GEOM	REAL*4	96	Ø6c Ø

Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

GRID Local Symbols

Name	Class	Type	Size	Offset
WFE	GEOM	REAL*4	192	Ø72Ø
WFW	GEOM	REAL*4	192	Ø7eØ
RCV	GEOM	REAL*4	96	Ø8aØ
UIN	CAUSO	REAL*4	4	ØØØØ
TEIN	CAUSO	REAL*4	4	ØØØ4
EDIN	CAUSO	REAL*4	4	ØØØ8
FLOWIN	CAUSO	REAL*4	4	ØØØc
ALAMDA	CAUSO	REAL*4	4	ØØ1Ø
RSMALL	CAUSO	REAL*4	4	ØØ14
RLARGE	CAUSO	REAL*4	4	ØØ18
XTOTA	CAUSO	REAL*4	4	ØØ1c
JSTEP	CAUSO	INTEGER*4	4	ØØ2Ø
ISTEP	CAUSO	INTEGER*4	4	ØØ24
JSTP1	CAUSO	INTEGER*4	4	ØØ28
JSTM1	CAUSO	INTEGER*4	4	ØØ2c
ISTP1	CAUSO	INTEGER*4	4	ØØ3Ø
ISTM1	CAUSO	INTEGER*4	4	ØØ34
JEXIT	CAUSO	INTEGER*4	4	ØØ38
JEXITP1	CAUSO	INTEGER*4	4	ØØ3c
INLETI	CAUSO	INTEGER*4	4	ØØ4Ø
INLETF	CAUSO	INTEGER*4	4	ØØ44
JINLET	CAUSO	INTEGER*4	4	ØØ48
RGARG	CAUSO	REAL*4	4	ØØ4c

Global Symbols

Name	Class	Type	Size	Offset
CAUSO	common	***	92	ØØØØ
GEOM	common	***	23Ø8	ØØØØ
GERAL	common	***	412	ØØØØ
GRID	FSUBRT	***	***	ØØØØ

Code size = Øb78 (2936)
Data size = ØØ3a (58)
Bss size = ØØ18 (24)

No errors detected

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
 1
 2         subroutine geo
 3
 4   cØ*****************       preliminaries
 5
 6
 7          common
 8        1/uvel/resolu,nswpu,urfu,dxepu(48),dxpwu(48),sewu(48)
 9        2/vvel/resolv,nswpv,urfv,dynpv(24),dypsv(24),snsv(24)
10Ø      3/wvel/resolw,nswpw,urfw,iomega
11        4/corrp/resolm,nswpp,urfp,du(48,24),dv(48,24),ipref,jpref
12        5/var/u(48,24), v(48,24), w(48,24), p(48,24), pp(48,24), te(48,24)
13        6      ed(48,24)
14
15         common
16        1/geral/it,jt,ni,nj,nim1,njm1,great,jmax(48),jmaxp1(48)
17        2/geom/x(48),y(24),dxep(48),dxpw(48),dynp(24),dyps(24),
18        3       sns(24),sew(48),xu(48),yv(24),r(24),rv(24),
19        4       wfn(24),wfs(24),wfe(48),wfw(48),rcv(24),ifine
20Ø       5/causo/uin,tein,edin,flowin,alamda,
21        6        rsmall,rlarge,xtota,jstep,istep,jstp1,jstm1,istp1,istm1,
22        7        jexit,jexitp1,inleti,inletf,jinlet,rgarg,ngarg,ngarm1,
23        8        ngarp1
24
25
26   c1*****************       calculate geometric quantities
27
28            do 1ØØ j=1,nj
29            r(j)=y(j)
30   1ØØ      continue
31
32   c---------------------     general control volume
33
34         dxpw(1)=Ø.Ø
35         dxep(ni)=Ø.Ø
36
37            do 1Ø1 i=1,nim1
38            dxep(i)=x(i+1)-x(i)
39            dxpw(i+1)=dxep(i)
40   1Ø1      continue
41
42                dyps(1)=Ø.Ø
43                dynp(nj)=Ø.Ø
44
45            do 1Ø2 j=1,njm1
46            dynp(j)=y(j+1)-y(j)
47            dyps(j+1)=dynp(j)
48   1Ø2      continue
49
50              sew(1)=Ø.Ø
51              sew(ni)=Ø.Ø
52
53                      do 1Ø3 i=2,nim1
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
                    sew(i)=Ø.5*(dxep(i)+dxpw(i))
103                 continue

          sns(1)=Ø.Ø
          sns(nj)=Ø.Ø

               do 1Ø4 j=2,njm1
               sns(j)=Ø.5*(dynp(j)+dyps(j))
1Ø4            continue


c---------------------    u control volume

       xu(1)=Ø.Ø

          do 1Ø5 i=2,ni
          xu(i)=Ø.5*(x(i)+x(i-1))
1Ø5       continue

          dxpwu(1)=Ø.Ø
          dxpwu(2)=Ø.Ø
          dxepu(1)=Ø.Ø
          dxepu(ni)=Ø.Ø

               do 1Ø6 i=2,nim1
               dxepu(i)=xu(i+1)-xu(i)
               dxpwu(i+1)=dxepu(i)
1Ø6            continue

        sewu(1)=Ø.Ø

                  do 1Ø7 i=2,ni
                  sewu(i)=x(i)-x(i-1)
1Ø7               continue

c---------------------    v control volume

         yv(1)=Ø.Ø
         rv(1)=Ø.Ø

           do 112 j=2,nj
           rv(j)=Ø.5*(r(j)+r(j-1))
           yv(j)=Ø.5*(y(j)+y(j-1))
112        continue

         rcv(1)=r(1)
         rcv(nj)=r(nj)

            do 113 j=2,njm1
            rcv(j)=Ø.5*(rv(j+1)+rv(j))
113         continue

         dypsv(1)=Ø.Ø
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
107           dypsv(2)=Ø.Ø
108           dynpv(nj)=Ø.Ø
109
110             do 114 j=2,njm1
111             dynpv(j)=yv(j+1)-yv(j)
112             dypsv(j+1)=dynpv(j)
113   114       continue
114
115                 snsv(1)=Ø.Ø
116
117                       do 115 j=2,nj
118                       snsv(j)=y(j)-y(j-1)
119   115                 continue
120
121   c---------------------    velocities weighting factors
122
123                 do 12Ø j=3,njm1
124                 wfn(j)=snsv(j+1)/(snsv(j+1)+snsv(j))
125                 wfs(j)=snsv(j-1)/(snsv(j-1)+snsv(j))
126   12Ø           continue
127
128                 do 121 i=2,nim1
129                 wfe(i)=sewu(i+1)/(sewu(i+1)+sewu(i))
130                      if(i .le. 2)  then
131                      continue
132                      else
133                      wfw(i)=sewu(i-1)/(sewu(i-1)+sewu(i))
134                      end if
135   121            continue
136
137             return
138
139             end
```

GEO Local Symbols

Name	Class	Type	Size	Offset
I	local	INTEGER*4	4	ØØØØ
J	local	INTEGER*4	4	ØØØ4
DYNP	GEOM	REAL*4	96	Ø2aØ
DYPS	GEOM	REAL*4	96	Ø3ØØ
SNS	GEOM	REAL*4	96	Ø36Ø
SEW	GEOM	REAL*4	192	Ø3cØ
XU	GEOM	REAL*4	192	Ø48Ø
YV	GEOM	REAL*4	96	Ø54Ø
R	GEOM	REAL*4	96	Ø5aØ
RV	GEOM	REAL*4	96	Ø6ØØ
WFN	GEOM	REAL*4	96	Ø66Ø
WFS	GEOM	REAL*4	96	Ø6cØ
WFE	GEOM	REAL*4	192	Ø72Ø
WFW	GEOM	REAL*4	192	Ø7eØ
RCV	GEOM	REAL*4	96	Ø8aØ

Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

GEO Local Symbols

Name	Class	Type	Size	Offset
IFINE	GEOM	INTEGER*4	4	Ø9ØØ
UIN	CAUSO	REAL*4	4	ØØØØ
RESOLU	UVEL	REAL*4	4	ØØØØ
TEIN	CAUSO	REAL*4	4	ØØØ4
NSWPU	UVEL	INTEGER*4	4	ØØØ4
EDIN	CAUSO	REAL*4	4	ØØØ8
URFU	UVEL	REAL*4	4	ØØØ8
FLOWIN	CAUSO	REAL*4	4	ØØØc
DXEPU	UVEL	REAL*4	192	ØØØc
ALAMDA	CAUSO	REAL*4	4	ØØ1Ø
DXPWU	UVEL	REAL*4	192	ØØcc
RSMALL	CAUSO	REAL*4	4	ØØ14
SEWU	UVEL	REAL*4	192	Ø18c
RLARGE	CAUSO	REAL*4	4	ØØ18
RESOLV	VVEL	REAL*4	4	ØØØØ
XTOTA	CAUSO	REAL*4	4	ØØ1c
NSWPV	VVEL	INTEGER*4	4	ØØØ4
JSTEP	CAUSO	INTEGER*4	4	ØØ2Ø
ISTEP	CAUSO	INTEGER*4	4	ØØ24
URFV	VVEL	REAL*4	4	ØØØ8
JSTP1	CAUSO	INTEGER*4	4	ØØ28
DYNPV	VVEL	REAL*4	96	ØØØc
JSTM1	CAUSO	INTEGER*4	4	ØØ2c
DYPSV	VVEL	REAL*4	96	ØØ6c
ISTP1	CAUSO	INTEGER*4	4	ØØ3Ø
SNSV	VVEL	REAL*4	96	ØØcc
ISTM1	CAUSO	INTEGER*4	4	ØØ34
RESOLW	WVEL	REAL*4	4	ØØØØ
JEXIT	CAUSO	INTEGER*4	4	ØØ38
NSWPW	WVEL	INTEGER*4	4	ØØØ4
JEXITP1	CAUSO	INTEGER*4	4	ØØ3c
INLETI	CAUSO	INTEGER*4	4	ØØ4Ø
URFW	WVEL	REAL*4	4	ØØØ8
IOMEGA	WVEL	INTEGER*4	4	ØØØc
INLETF	CAUSO	INTEGER*4	4	ØØ44
JINLET	CAUSO	INTEGER*4	4	ØØ48
RGARG	CAUSO	REAL*4	4	ØØ4c
RESOLM	CORRP	REAL*4	4	ØØØØ
NGARG	CAUSO	INTEGER*4	4	ØØ5Ø
NSWPP	CORRP	INTEGER*4	4	ØØØ4
URFP	CORRP	REAL*4	4	ØØØ8
NGARM1	CAUSO	INTEGER*4	4	ØØ54
NGARP1	CAUSO	INTEGER*4	4	ØØ58
DU	CORRP	REAL*4	46Ø8	ØØØc
DV	CORRP	REAL*4	46Ø8	12Øc
IPREF	CORRP	INTEGER*4	4	24Øc
JPREF	CORRP	INTEGER*4	4	241Ø
U	VAR	REAL*4	46Ø8	ØØØØ

Microsoft FORTRAN Optimizing Compiler Version 5.00

GEO Local Symbols

Name	Class	Type	Size	Offset
V	VAR	REAL*4	4608	1200
W	VAR	REAL*4	4608	2400
P	VAR	REAL*4	4608	3600
PP	VAR	REAL*4	4608	4800
TE	VAR	REAL*4	4608	5a00
ED	VAR	REAL*4	4608	6c00
IT	GERAL	INTEGER*4	4	0000
JT	GERAL	INTEGER*4	4	0004
NI	GERAL	INTEGER*4	4	0008
NJ	GERAL	INTEGER*4	4	000c
NIM1	GERAL	INTEGER*4	4	0010
NJM1	GERAL	INTEGER*4	4	0014
GREAT	GERAL	REAL*4	4	0018
JMAX	GERAL	INTEGER*4	192	001c
JMAXP1	GERAL	INTEGER*4	192	00dc
X	GEOM	REAL*4	192	0000
Y	GEOM	REAL*4	96	00c0
DXEP	GEOM	REAL*4	192	0120
DXPW	GEOM	REAL*4	192	01e0

Global Symbols

Name	Class	Type	Size	Offset
CAUSO	common	***	92	0000
CORRP	common	***	9236	0000
GEO	FSUBRT	***	***	0000
GEOM	common	***	2308	0000
GERAL	common	***	412	0000
UVEL	common	***	588	0000
VAR	common	***	32256	0000
VVEL	common	***	300	0000
WVEL	common	***	16	0000

Code size = 0c0d (3085)
Data size = 000c (12)
Bss size = 0008 (8)

No errors detected

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
      subroutine calcu

cØ******************      preliminaries

      logical istone,isimplec,iupwind

      common
     1/uvel/resolu,nswpu,urfu,dxepu(48),dxpwu(48),sewu(48)
     2/vvel/resolv,nswpv,urfv,dynpv(24),dypsv(24),snsv(24)
     3/corrp/resolm,nswpp,urfp,du(48,24),dv(48,24),ipref,jpref
     4/var/u(48,24), v(48,24), w(48,24), p(48,24), pp(48,24), te(48,24)
     5     ed(48,24)
     6/geral/it,jt,ni,nj,nim1,njm1,great,jmax(48),jmaxp1(48)
     7/geom/x(48),y(24),dxep(48),dxpw(48),dynp(24),dyps(24),
     8       sns(24),sew(48),xu(48),yv(24),r(24),rv(24),
     9       wfn(24),wfs(24),wfe(48),wfw(48),rcv(24),ifine

      common
     1/flupr/urften,viscos,densit,prandt,den(48,24)
     2/turbu/gen(48,24),cd,cmu,c1e,c2e,betarodi,cappa,elog,
     3       pred,prte,c1,c2,umc2,c1m1,ck,ce3
     4/causo/uin,tein,edin,flowin,alamda,
     5        rsmall,rlarge,xtota,jstep,istep,jstp1,jstm1,istp1,istm1,
     6        jexit,jexitp1,inleti,inletf,jinlet,rgarg,ngarg,ngarm1,
     7        ngarp1
     8/coef/ap(48,24),an(48,24),as(48,24),ae(48,24),aw(48,24),su(48,24)
     9       sp(48,24)
     1/stone/tetastone,source,istone,niter,isimplec

      common
     1/stress/uu(48,24),vv(48,24),ww(48,24),uv(48,24),
     2         uw(48,24),vw(48,24)
     3/wind/iupwind

      dimension rait(48,24),akw(48,24),ake(48,24),aks(48,24),
     1          akn(48,24)



c1******************      coefficients

      do 1ØØ i=3,nim1
      do 1ØØ j=2,njm1

c---------------------     areas and volume

      arean=rv(j+1)*sewu(i)
      areas=rv(j)*sewu(i)
      areaew=rcv(j)*sns(j)
      vol=rcv(j)*sewu(i)*sns(j)

c---------------------     diffusion coefficients
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
107
108               end if
109
110
111             visn=Ø.25*(vis1+vis2+vis3+vis4)+viscos
112             viss=Ø.25*(vis1+vis4+vis5+vis6)+viscos
113
114
115           vise=2.*den(i,j)*alamb1*te(i,j)/ed(i,j)*uu(i,j)+viscos
116           visw=2.*den(i-1,j)*alamb2*te(i-1,j)/ed(i-1,j)*uu(i-1,j)+viscos
117
118                else
119
120                visn=Ø.
121                viss=Ø.
122                vise=Ø.
123                visw=Ø.
124
125                end if
126
127
128          dn=visn*arean/dynp(j)
129          ds=viss*areas/dyps(j)
130          de=vise*areaew/dxepu(i)
131          dw=visw*areaew/dxpwu(i)
132
133
134           if ( iupwind ) then
135
136
137   c---------------------    convection coefficients
138
139          qn=Ø.5*(den(i,j+1)+den(i,j))*v(i,j+1)
140          qnw=Ø.5*(den(i-1,j)+den(i-1,j+1))*v(i-1,j+1)
141          qs=Ø.5*(den(i,j-1)+den(i,j))*v(i,j)
142          qsw=Ø.5*(den(i-1,j)+den(i-1,j-1))*v(i-1,j)
143          qe=den(i,j)*(u(i,j)*wfe(i)+(1.Ø-wfe(i))*u(i+1,j))
144          qw=den(i-1,j)*(u(i,j)*wfw(i)+(1.Ø-wfw(i))*u(i-1,j))
145
146          fn=Ø.5*(qn+qnw)*arean
147          fs=Ø.5*(qs+qsw)*areas
148          fe=qe*areaew
149          fw=qw*areaew
150
151
152   c---------------------    main coefficients-HYBRID
153
154          an(i,j)=amax1(abs(Ø.5*fn),dn)-Ø.5*fn
155          as(i,j)=amax1(abs(Ø.5*fs),ds)+Ø.5*fs
156          ae(i,j)=amax1(de,-wfe(i)*fe,(1.Ø-wfe(i))*fe)-(1.Ø-wfe(i))*fe
157          aw(i,j)=amax1(dw,wfw(i)*fw,-(1.Ø-wfw(i))*fw)+(1.Ø-wfw(i))*fw
158          akn(i,j)=Ø.Ø
159          aks(i,j)=Ø.Ø
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
160         akw(i,j)=Ø.Ø
161         ake(i,j)=Ø.Ø
162
163         smp=fn-fs+fe-fw
164
165         else
166
167
168   c---------------------    main coefficients-SUDS-RAITHBY
169
170   c---------------------    west
171
172            uwest=u(i,j)*wfw(i)+(1.Ø-wfw(i))*u(i-1,j)
173            vwest=v(i-1,j)*wfn(j)+(1.Ø-wfn(j))*v(i-1,j+1)
174
175            alw=Ø.5*den(i-1,j)*uwest*areaew
176
177                 if ( uwest .ge. Ø.Ø ) then
178                 isuw=1
179                 else
180                 isuw=-1
181                 end if
182
183                 if ( vwest .ge. Ø.Ø ) then
184                 isvw=1
185                 else
186                 isvw=-1
187                 end if
188
189            lwi=i-(1+isuw)/2
190            mwi=j-isvw
191            kwi=j+(1-isvw)/2
192
193                  dob=den(i-1,j)*areaew/dyps(kwi)*dxpwu(i)/4.*abs(vwest)
194                  akw(i,j)=float(isuw)*amin1(abs(alw),dob)
195
196                  aw(i,j)=dw+(alw-akw(i,j))*float(1+isuw)
197
198   c---------------------    east
199
200            ueast=u(i,j)*wfe(i)+(1.Ø-wfe(i))*u(i+1,j)
201            veast=v(i,j)*wfn(j)+(1-wfn(j))*v(i,j+1)
202
203            ale=Ø.5*den(i,j)*ueast*areaew
204
205                 if ( ueast .ge. Ø.Ø ) then
206                 isue=1
207                 else
208                 isue=-1
209                 end if
210
211                 if ( veast .ge. Ø.Ø ) then
212                 isve=1
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
213                else
214                isve=-1
215                end if
216
217           lei=i+(1-isue)/2
218           mei=j-isve
219           kei=j+(1-isve)/2
220
221                dob=den(i,j)*areaew/dyps(kei)*dxepu(i)/4*abs(veast)
222                ake(i,j)=float(isue)*amin1(abs(ale),dob)
223
224                ae(i,j)=de-(ale-ake(i,j))*float(1-isue)
225
226   c---------------------    north
227
228           unort=Ø.5*(u(i,j)+u(i,j+1))
229           vnort=Ø.5*(v(i,j+1)+v(i-1,j+1))
230           rotemp=Ø.25*(den(i,j)+den(i,j+1)+den(i-1,j)+den(i-1,j+1))
231
232           aln=Ø.5*rotemp*vnort*arean
233
234                if ( vnort .ge. Ø.Ø ) then
235                isvn=1
236                else
237                isvn=-1
238                end if
239
240                if ( unort .ge. Ø.Ø ) then
241                isun=1
242                else
243                isun=-1
244                end if
245
246           lni=i-isun
247           mni=j+(1-isvn)/2
248           kni=i+(1-isun)/2
249
250                dob=rotemp*arean/dxpwu(kni)*dynp(j)/4.*abs(unort)
251                akn(i,j)=float(isvn)*amin1(abs(aln),dob)
252
253                an(i,j)=dn-(aln-akn(i,j))*float(1-isvn)
254
255   c---------------------    south
256
257           usout=Ø.5*(u(i,j)+u(i,j-1))
258           vsout=Ø.5*(v(i,j)+v(i-1,j))
259           rotemp=Ø.25*(den(i,j)+den(i,j-1)+den(i-1,j)+den(i-1,j-1))
260
261           als=Ø.5*rotemp*vsout*areas
262
263                if ( vsout .ge. Ø.Ø ) then
264                isvs=1
265                else
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
266                isvs=-1
267                end if
268
269                if ( usout .ge. Ø.Ø ) then
270Ø                isus=1
271                else
272                isus=-1
273                end if
274
275            lsi=i-isus
276            msi=j-(1+isvs)/2
277            ksi=i+(1-isus)/2
278
279                 dob= rotemp*areas/dxpwu(ksi)*dyps(j)/4.*abs(usout)
280Ø                aks(i,j)=float(isvs)*amin1(abs(als),dob)
281
282                as(i,j)=ds+(als-aks(i,j))*float(1+isvs)
283
284        smp=2.*(aln-als+ale-alw)
285
286        end if
287
288  c---------------------    coefficients of source terms
289  c---------------------    Rodi ASM - Gibson & Launder
290Ø
291        cp=amax1(Ø.Ø,smp)
292
293        du(i,j)=areaew
294
295              if ( j .gt. jmax(i) ) then
296
297              sxxe=Ø.
298              sxxw=Ø.
299              srxn=Ø.
300Ø              srxs=Ø.
301
302              else
303
304        dudxe=(u(i+1,j)-u(i,j))/dxepu(i)
305        dudxw=(u(i,j)-u(i-1,j))/dxpwu(i)
306
307        dvdxn=(v(i,j+1)-v(i-1,j+1))/sewu(i)
308        dvdxs=(v(i,j)-v(i-1,j))/sewu(i)
309
310            if ( j .eq. 2) then
311
312            utmp1=u(i,3)*wfe(i)+(1.Ø-wfe(i))*u(i+1,3)
313            utmp2=u(i,2)*wfe(i)+(1.Ø-wfe(i))*u(i+1,2)
314            dudye=(utmp1-utmp2)/r(3)
315            utmp1=u(i,3)*wfw(i)+(1.Ø-wfw(i))*u(i-1,3)
316            utmp2=u(i,2)*wfw(i)+(1.Ø-wfw(i))*u(i-1,2)
317            dudyw=(utmp1-utmp2)/r(3)
318
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
319             else
320Ø
321             utmp1=u(i,j+1)*wfe(i)+(1.Ø-wfe(i))*u(i+1,j+1)
322             utmp2=u(i,j-1)*wfe(i)+(1.Ø-wfe(i))*u(i+1,j-1)
323             dudye=(utmp1-utmp2)/(r(j+1)-r(j-1))
324             utmp1=u(i,j+1)*wfw(i)+(1.Ø-wfw(i))*u(i-1,j+1)
325             utmp2=u(i,j-1)*wfw(i)+(1.Ø-wfw(i))*u(i-1,j-1)
326             dudyw=(utmp1-utmp2)/(r(j+1)-r(j-1))
327
328             end if
329
330Ø             uvØ=uv(i,j)*wfn(j)+(1.-wfn(j))*uv(i,j+1)
331             uv1=uv(i+1,j)*wfn(j)+(1.-wfn(j))*uv(i+1,j+1)
332             uvtmp1=uvØ*wfe(i)+(1.-wfe(i))*uv1
333
334             uv1=uv(i-1,j)*wfn(j)+(1.-wfn(j))*uv(i-1,j+1)
335             uvtmp2=uvØ*wfw(i)+(1.-wfw(i))*uv1
336
337           sxxe=2.*den(i,j)*(alamb1*te(i,j)/ed(i,j)*
338      1         (uvtmp1*dudye+gen(i,j)/3.)-te(i,j)/3.)
339
340Ø           sxxw=2.*den(i-1,j)*(alamb2*te(i-1,j)/ed(i-1,j)*
341      1         (uvtmp2*dudyw+gen(i-1,j)/3.)-te(i-1,j)/3.)
342
343
344
345         spock1=te(i,j)+te(i-1,j)
346         spock2=ed(i,j)+ed(i-1,j)
347         spock3=gen(i,j)+gen(i-1,j)
348         spock4=den(i,j)+den(i-1,j)
349         spock5=uu(i,j)+uu(i-1,j)
350Ø         spock6=w(i,j)+w(i-1,j)
351
352
353         tetmp=Ø.25*(spock1+te(i,j+1)+te(i-1,j+1))
354         edtmp=Ø.25*(spock2+ed(i,j+1)+ed(i-1,j+1))
355         getmp=Ø.25*(spock3+gen(i,j+1)+gen(i-1,j+1))
356         detmp=Ø.25*(spock4+den(i,j+1)+den(i-1,j+1))
357         uutmp=Ø.25*(spock5+uu(i,j+1)+uu(i-1,j+1))
358         uwtmp=Ø.5*(uw(i,j)+uw(i,j+1))
359         wtmp=Ø.25*(spock6+w(i,j+1)+w(i-1,j+1))
360Ø
361         alambt=umc2/(c1m1+getmp/edtmp)
362         gamarx=1.-alambt*tetmp/edtmp*Ø.5*(v(i,j+1)+v(i-1,j+1))/rv(j+1)
363
364          if ( ( i .gt. inletf ) .and. ( j .eq. jmax(i) ) ) then
365          srxn=Ø.
366          else
367          srxn=rv(j+1)*detmp*alambt/gamarx*tetmp/edtmp*(uutmp*dvdxn-
368      1       uwtmp*wtmp/rv(j+1)*(1+betarodi))
369          end if
370Ø
371
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
372           tetmp=Ø.25*(spock1+te(i,j-1)+te(i-1,j-1))
373           edtmp=Ø.25*(spock2+ed(i,j-1)+ed(i-1,j-1))
374           getmp=Ø.25*(spock3+gen(i,j-1)+gen(i-1,j-1))
375           detmp=Ø.25*(spock4+den(i,j-1)+den(i-1,j-1))
376           uutmp=Ø.25*(spock5+uu(i,j-1)+uu(i-1,j-1))
377           uwtmp=Ø.5*(uw(i,j)+uw(i,j-1))
378           wtmp=Ø.25*(spock6+w(i,j-1)+w(i-1,j-1))
379
380            if ( j .eq. 2 ) then
381           srxs=Ø.
382            else
383           alambt=umc2/(c1m1+getmp/edtmp)
384           gamarx=1.-alambt*tetmp/edtmp*Ø.5*(v(i,j)+v(i-1,j))/rv(j)
385           srxs=detmp*alambt/gamarx*tetmp/edtmp*(rv(j)*uutmp*dvdxs-
386         1       uwtmp*wtmp*(1+betarodi))
387            end if
388
389            end if
390
391          sorce1=viscos*(dudxe-dudxw)/sewu(i)
392          sorce2=viscos*(rv(j+1)*dvdxn-rv(j)*dvdxs)/(rcv(j)*dynpv(j))
393          sorce3=(sxxe-sxxw)/sewu(i)
394          sorce4=(srxn-srxs)/(rcv(j)*dynpv(j))
395
396          su(i,j)=cp*u(i,j)+du(i,j)*(p(i-1,j)-p(i,j))
397          su(i,j)=su(i,j)+(sorce1+sorce2+sorce3+sorce4)*vol
398
399
400   c----------------------     source modification- suds
401
402          su(i,j)=su(i,j)+2.*(akw(i,j)*u(lwi,mwi)+aks(i,j)*u(lsi,msi)
403         1         -ake(i,j)*u(lei,mei)-akn(i,j)*u(lni,mni))
404
405          sp(i,j)=-cp
406
407   1ØØ    continue
408
409
410   c2*******************      boundary
411
412          call boundary (2)
413
414   c3*******************      residual source calculation
415
416          resolu=Ø.Ø
417          do 3ØØ i=3,nim1
418          do 3ØØ j=2,njm1
419          rait(i,j)=2.*(akw(i,j)+aks(i,j)-ake(i,j)-akn(i,j))
420
421          ap(i,j)=an(i,j)+as(i,j)+ae(i,j)+aw(i,j)-sp(i,j)+rait(i,j)
422          du(i,j)=du(i,j)/ap(i,j)
423
424          resol=an(i,j)*u(i,j+1)+as(i,j)*u(i,j-1)+ae(i,j)*u(i+1,j)
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
425        1     +aw(i,j)*u(i-1,j)-ap(i,j)*u(i,j)+su(i,j)
426
427         vol=rcv(j)*sewu(i)*sns(j)
428         sorvol=great*vol
429         if( -sp(i,j) .gt. Ø.5*sorvol )  resol=resol/sorvol
430         resolu=resolu+abs(resol)
431
432   c---------------------    under-relaxation
433
434         ap(i,j)=ap(i,j)/urfu
435         su(i,j)=su(i,j)+(1.-urfu)*ap(i,j)*u(i,j)
436         du(i,j)=du(i,j)*urfu
437   3ØØ   continue
438
439   c4******************      solution
440
441                 if (istone) then
442
443                   tetastone=1.85
444                 else
445                   tetastone=1.Ø
446
447                 end if
448
449         do 4ØØ n=1,nswpu
450
451                 call stone1 (3,2,ni,jmax,it,jt,u,2)
452
453   4ØØ   continue
454
455
456   c5******************      simplec approximation
457
458                     if ( isimplec ) then
459
460             do 5ØØ i=3,nim1
461             do 5ØØ j=2,njm1
462
463             gundi=an(i,j)+as(i,j)+aw(i,j)+ae(i,j)
464             du(i,j)=du(i,j)*ap(i,j)/(ap(i,j)-gundi)
465
466   5ØØ       continue
467
468                   end if
469
470                      return
471
472                         end
```

Microsoft FORTRAN Optimizing Compiler Version 5.00

CALCU Local Symbols

Name	Class	Type	Size	Offset
ISVN.	local	INTEGER*4	4	0000
VISN.	local	REAL*4	4	0004
GAMARX.	local	REAL*4	4	0008
UEAST	local	REAL*4	4	000c
VEAST	local	REAL*4	4	0010
DVDXN	local	REAL*4	4	0014
ISUS.	local	INTEGER*4	4	0018
RESOL	local	REAL*4	4	001c
ISVS.	local	INTEGER*4	4	0020
VISS.	local	REAL*4	4	0024
WTMP.	local	REAL*4	4	0028
SXXE.	local	REAL*4	4	002c
ISUW.	local	INTEGER*4	4	0030
DVDXS	local	REAL*4	4	0034
ISVW.	local	INTEGER*4	4	0038
DE.	local	REAL*4	4	003c
VISW.	local	REAL*4	4	0040
I	local	INTEGER*4	4	0044
TETMP	local	REAL*4	4	0048
J	local	INTEGER*4	4	004c
SRXN.	local	REAL*4	4	0050
FE.	local	REAL*4	4	0054
DUDXW	local	REAL*4	4	0058
VTEMP	local	REAL*4	4	005c
UVTMP1.	local	REAL*4	4	0060
DUDYW	local	REAL*4	4	0064
N	local	INTEGER*4	4	0068
UVTMP2.	local	REAL*4	4	006c
ALAMB1.	local	REAL*4	4	0070
ALAMB2.	local	REAL*4	4	0074
SRXS.	local	REAL*4	4	0078
AKE	local	REAL*4	4608	007c
ALE	local	REAL*4	4	127c
DN.	local	REAL*4	4	1280
CP.	local	REAL*4	4	1284
FN.	local	REAL*4	4	1288
DOB	local	REAL*4	4	128c
QE.	local	REAL*4	4	1290
ROTEMP.	local	REAL*4	4	1294
DS.	local	REAL*4	4	1298
UNORT	local	REAL*4	4	129c
UWEST	local	REAL*4	4	12a0
VNORT	local	REAL*4	4	12a4
KEI	local	INTEGER*4	4	12a8
VWEST	local	REAL*4	4	12ac
FS.	local	REAL*4	4	12b0
SXXW.	local	REAL*4	4	12b4
LEI	local	INTEGER*4	4	12b8
AKN	local	REAL*4	4608	12bc

CALCU Local Symbols

Name	Class	Type	Size	Offset
UUTMP	local	REAL*4	4	24bc
UVØ	local	REAL*4	4	24cØ
ALN	local	REAL*4	4	24c4
MEI	local	INTEGER*4	4	24c8
DW.	local	REAL*4	4	24cc
UV1	local	REAL*4	4	24dØ
UWTMP	local	REAL*4	4	24d4
FW.	local	REAL*4	4	24d8
QN.	local	REAL*4	4	24dc
AKS	local	REAL*4	46Ø8	24eØ
ALS	local	REAL*4	4	36eØ
USOUT	local	REAL*4	4	36e4
VSOUT	local	REAL*4	4	36e8
KNI	local	INTEGER*4	4	36ec
LNI	local	INTEGER*4	4	36fØ
VIS1.	local	REAL*4	4	36f4
AKW	local	REAL*4	46Ø8	36f8
MNI	local	INTEGER*4	4	48f8
ALW	local	REAL*4	4	48fc
QS.	local	REAL*4	4	49ØØ
VIS2.	local	REAL*4	4	49Ø4
SORVOL.	local	REAL*4	4	49Ø8
VIS3.	local	REAL*4	4	49Øc
VIS4.	local	REAL*4	4	491Ø
KSI	local	INTEGER*4	4	4914
VIS5.	local	REAL*4	4	4918
AREAN	local	REAL*4	4	491c
LSI	local	INTEGER*4	4	492Ø
QW.	local	REAL*4	4	4924
VIS6.	local	REAL*4	4	4928
MSI	local	INTEGER*4	4	492c
KWI	local	INTEGER*4	4	493Ø
LWI	local	INTEGER*4	4	4934
AREAS	local	REAL*4	4	4938
SORCE1.	local	REAL*4	4	493c
MWI	local	INTEGER*4	4	494Ø
SORCE2.	local	REAL*4	4	4944
SORCE3.	local	REAL*4	4	4948
SORCE4.	local	REAL*4	4	494c
SMP	local	REAL*4	4	495Ø
RAIT.	local	REAL*4	46Ø8	4954
SPOCK1.	local	REAL*4	4	5b54
ALAMBT.	local	REAL*4	4	5b58
VOL	local	REAL*4	4	5b5c
SPOCK2.	local	REAL*4	4	5b6Ø
SPOCK3.	local	REAL*4	4	5b64
SPOCK4.	local	REAL*4	4	5b68
SPOCK5.	local	REAL*4	4	5b6c

CALCU Local Symbols

Name	Class	Type	Size	Offset
AREAEW.	local	REAL*4	4	5b7Ø
SPOCK6.	local	REAL*4	4	5b74
ISUE.	local	INTEGER*4	4	5b78
QNW	local	REAL*4	4	5b7c
GUNDI	local	REAL*4	4	5b8Ø
UTMP1	local	REAL*4	4	5b84
ISVE.	local	INTEGER*4	4	5b88
VISE.	local	REAL*4	4	5b8c
UTMP2	local	REAL*4	4	5b9Ø
DETMP	local	REAL*4	4	5b94
EDTMP	local	REAL*4	4	5b98
DUDXE	local	REAL*4	4	5b9c
DUDYE	local	REAL*4	4	5ba0
QSW	local	REAL*4	4	5ba4
GETMP	local	REAL*4	4	5ba8
ISUN.	local	INTEGER*4	4	5bac
SNS	GEOM	REAL*4	96	Ø36Ø
AW.	COEF	REAL*4	46Ø8	48ØØ
SEW	GEOM	REAL*4	192	Ø3cØ
SU.	COEF	REAL*4	46Ø8	5aØØ
XU.	GEOM	REAL*4	192	Ø48Ø
SP.	COEF	REAL*4	46Ø8	6cØØ
YV.	GEOM	REAL*4	96	Ø54Ø
TETASTONE	STONE	REAL*4	4	ØØØØ
R	GEOM	REAL*4	96	Ø5aØ
RV.	GEOM	REAL*4	96	Ø6ØØ
SOURCE.	STONE	REAL*4	4	ØØØ4
WFN	GEOM	REAL*4	96	Ø66Ø
NITER	STONE	INTEGER*4	4	ØØØc
WFS	GEOM	REAL*4	96	Ø6cØ
UU.	STRESS	REAL*4	46Ø8	ØØØØ
WFE	GEOM	REAL*4	192	Ø72Ø
WFW	GEOM	REAL*4	192	Ø7eØ
VV.	STRESS	REAL*4	46Ø8	12ØØ
WW.	STRESS	REAL*4	46Ø8	24ØØ
RCV	GEOM	REAL*4	96	Ø8aØ
IFINE	GEOM	INTEGER*4	4	Ø9ØØ
UV.	STRESS	REAL*4	46Ø8	36ØØ
UW.	STRESS	REAL*4	46Ø8	48ØØ
VW.	STRESS	REAL*4	46Ø8	5aØØ
URFTEN.	FLUPR	REAL*4	4	ØØØØ
VISCOS.	FLUPR	REAL*4	4	ØØØ4
ISTONE.	STONE	LOGICAL*4	4	ØØØ8
ISIMPLEC.	STONE	LOGICAL*4	4	ØØ1Ø
DENSIT.	FLUPR	REAL*4	4	ØØØ8
IUPWIND	WIND	LOGICAL*4	4	ØØØØ
PRANDT.	FLUPR	REAL*4	4	ØØØc
DEN	FLUPR	REAL*4	46Ø8	ØØ1Ø

CALCU Local Symbols

Name	Class	Type	Size	Offset
RESOLU.	UVEL	REAL*4	4	0000
NSWPU	UVEL	INTEGER*4	4	0004
GEN	TURBU	REAL*4	4608	0000
CD.	TURBU	REAL*4	4	1200
URFU.	UVEL	REAL*4	4	0008
CMU	TURBU	REAL*4	4	1204
DXEPU	UVEL	REAL*4	192	000c
C1E	TURBU	REAL*4	4	1208
DXPWU	UVEL	REAL*4	192	00cc
C2E	TURBU	REAL*4	4	120c
SEWU.	UVEL	REAL*4	192	018c
BETARODI.	TURBU	REAL*4	4	1210
CAPPA	TURBU	REAL*4	4	1214
RESOLV.	VVEL	REAL*4	4	0000
ELOG.	TURBU	REAL*4	4	1218
NSWPV	VVEL	INTEGER*4	4	0004
PRED.	TURBU	REAL*4	4	121c
URFV.	VVEL	REAL*4	4	0008
PRTE.	TURBU	REAL*4	4	1220
DYNPV	VVEL	REAL*4	96	000c
C1.	TURBU	REAL*4	4	1224
DYPSV	VVEL	REAL*4	96	006c
C2.	TURBU	REAL*4	4	1228
SNSV.	VVEL	REAL*4	96	00cc
UMC2.	TURBU	REAL*4	4	122c
C1M1.	TURBU	REAL*4	4	1230
RESOLM.	CORRP	REAL*4	4	0000
NSWPP	CORRP	INTEGER*4	4	0004
CK.	TURBU	REAL*4	4	1234
URFP.	CORRP	REAL*4	4	0008
CE3	TURBU	REAL*4	4	1238
DU.	CORRP	REAL*4	4608	000c
UIN	CAUSO	REAL*4	4	0000
DV.	CORRP	REAL*4	4608	120c
IPREF	CORRP	INTEGER*4	4	240c
TEIN.	CAUSO	REAL*4	4	0004
JPREF	CORRP	INTEGER*4	4	2410
EDIN.	CAUSO	REAL*4	4	0008
FLOWIN.	CAUSO	REAL*4	4	000c
ALAMDA.	CAUSO	REAL*4	4	0010
U	VAR	REAL*4	4608	0000
V	VAR	REAL*4	4608	1200
RSMALL.	CAUSO	REAL*4	4	0014
RLARGE.	CAUSO	REAL*4	4	0018
W	VAR	REAL*4	4608	2400
P	VAR	REAL*4	4608	3600
XTOTA	CAUSO	REAL*4	4	001c
PP.	VAR	REAL*4	4608	4800

CALCU Local Symbols

Name	Class	Type	Size	Offset
JSTEP	CAUSO	INTEGER*4	4	ØØ2Ø
TE.	VAR	REAL*4	46Ø8	5aØØ
ISTEP	CAUSO	INTEGER*4	4	ØØ24
JSTP1	CAUSO	INTEGER*4	4	ØØ28
ED.	VAR	REAL*4	46Ø8	6cØØ
JSTM1	CAUSO	INTEGER*4	4	ØØ2c
ISTP1	CAUSO	INTEGER*4	4	ØØ3Ø
IT.	GERAL	INTEGER*4	4	ØØØØ
ISTM1	CAUSO	INTEGER*4	4	ØØ34
JT.	GERAL	INTEGER*4	4	ØØØ4
NI.	GERAL	INTEGER*4	4	ØØØ8
JEXIT	CAUSO	INTEGER*4	4	ØØ38
NJ.	GERAL	INTEGER*4	4	ØØØc
JEXITP1	CAUSO	INTEGER*4	4	ØØ3c
NIM1.	GERAL	INTEGER*4	4	ØØ1Ø
INLETI.	CAUSO	INTEGER*4	4	ØØ4Ø
NJM1.	GERAL	INTEGER*4	4	ØØ14
INLETF.	CAUSO	INTEGER*4	4	ØØ44
GREAT	GERAL	REAL*4	4	ØØ18
JINLET.	CAUSO	INTEGER*4	4	ØØ48
RGARG	CAUSO	REAL*4	4	ØØ4c
JMAX.	GERAL	INTEGER*4	192	Øø1c
JMAXP1.	GERAL	INTEGER*4	192	ØØdc
NGARG	CAUSO	INTEGER*4	4	ØØ5Ø
NGARM1.	CAUSO	INTEGER*4	4	ØØ54
NGARP1.	CAUSO	INTEGER*4	4	ØØ58
X	GEOM	REAL*4	192	ØØØØ
Y	GEOM	REAL*4	96	ØØc0
DXEP.	GEOM	REAL*4	192	Ø12Ø
AP.	COEF	REAL*4	46Ø8	ØØØØ
AN.	COEF	REAL*4	46Ø8	12ØØ
DXPW.	GEOM	REAL*4	192	Ø1eØ
DYNP.	GEOM	REAL*4	96	Ø2aØ
AS.	COEF	REAL*4	46Ø8	24ØØ
AE.	COEF	REAL*4	46Ø8	36ØØ
DYPS.	GEOM	REAL*4	96	Ø3ØØ

Global Symbols

Name	Class	Type	Size	Offset
BOUNDARY.	extern	***	***	***
CALCU	FSUBRT	***	***	ØØØØ
CAUSO	common	***	92	ØØØØ
COEF.	common	***	32256	ØØØØ
CORRP	common	***	9236	ØØØØ
FLUPR	common	***	4624	ØØØØ

Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

Global Symbols

Name	Class	Type	Size	Offset
GEOM.	common	***	23Ø8	ØØØØ
GERAL	common	***	412	ØØØØ
STONE	common	***	2Ø	ØØØØ
STONE1.	extern	***	***	***
STRESS.	common	***	27648	ØØØØ
TURBU	common	***	4668	ØØØØ
UVEL.	common	***	588	ØØØØ
VAR	common	***	32256	ØØØØ
VVEL.	common	***	3ØØ	ØØØØ
WIND.	common	***	4	ØØØØ

Code size = 2fa3 (12195)
Data size = ØØ42 (66)
Bss size = 5bbØ (23472)

No errors detected

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
 1
 2
 3         subroutine calcv
 4
 5   cØ******************      preliminaries
 6
 7         logical istone,isimplec,iomega,iupwind
 8
 9         common
10        1/uvel/resolu,nswpu,urfu,dxepu(48),dxpwu(48),sewu(48)
11        2/vvel/resolv,nswpv,urfv,dynpv(24),dypsv(24),snsv(24)
12        3/wvel/resolw,nswpw,urfw,iomega
13        4/corrp/resolm,nswpp,urfp,du(48,24),dv(48,24),ipref,jpref
14        5/var/u(48,24), v(48,24), w(48,24), p(48,24), pp(48,24), te(48,24)
15        6      ed(48,24)
16        7/geral/it,jt,ni,nj,nim1,njm1,great,jmax(48),jmaxp1(48)
17        8/geom/x(48),y(24),dxep(48),dxpw(48),dynp(24),dyps(24),
18        9       sns(24),sew(48),xu(48),yv(24),r(24),rv(24),
19        1       wfn(24),wfs(24),wfe(48),wfw(48),rcv(24),ifine
20
21          common
22        1/flupr/urften,viscos,densit,prandt,den(48,24)
23        2/turbu/gen(48,24),cd,cmu,c1e,c2e,betarodi,cappa,elog,
24        3       pred,prte,c1,c2,umc2,c1m1,ck,ce3
25        4/causo/uin,tein,edin,flowin,alamda,
26        5        rsmall,rlarge,xtota,jstep,istep,jstp1,jstm1,istp1,istm1,
27        6        jexit,jexitp1,inleti,inletf,jinlet,rgarg,ngarg,ngarm1,
28        7        ngarp1
29        8/coef/ap(48,24),an(48,24),as(48,24),ae(48,24),aw(48,24),su(48,24)
30        9       sp(48,24)
31        1/stone/tetastone,source,istone,niter,isimplec
32
33         common
34        1/stress/uu(48,24),vv(48,24),ww(48,24),uv(48,24),
35        2         uw(48,24),vw(48,24)
36        3/wind/ iupwind
37
38         dimension rait(48,24),akw(48,24),ake(48,24),aks(48,24),
39        1           akn(48,24)
40
41   c---------------------    Gosman's  correction
42
43            alfag=2.Ø
44
45   c1******************      coefficients
46
47         do 1ØØ i=2,nim1
48         do 1ØØ j=3,njm1
49
50   c----------------------   areas and volume
51
52         arean=r(j)*sew(i)
53         areas=r(j-1)*sew(i)
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
54          areaew=rv(j)*snsv(j)
55          vol=rv(j)*sew(i)*snsv(j)
56
57    c---------------------    diffusion coefficients
58
59                  if ( j .le. jmax(i) ) then
60
61          alambt=umc2/(c1m1+gen(i,j)/ed(i,j))
62          alamb1=alambt
63          vtmp=v(i,j)*wfn(j)+(1.Ø-wfn(j))*v(i,j+1)
64          gamarx=1.-alambt*te(i,j)/ed(i,j)*vtmp/r(j)
65          vis1=den(i,j)*alambt/gamarx*te(i,j)/ed(i,j)*uu(i,j)
66
67          if ( ((i .eq. ngarm1) .and. (j .ge. jexitp1)) .or.
68         1     (i .eq. nim1) ) then
69          vis2=Ø.
70          vis3=Ø.
71
72          else
73
74          alambt=umc2/(c1m1+gen(i+1,j)/ed(i+1,j))
75          vtmp=v(i+1,j)*wfn(j)+(1.Ø-wfn(j))*v(i+1,j+1)
76          gamarx=1.-alambt*te(i+1,j)/ed(i+1,j)*vtmp/r(j)
77          vis2=den(i+1,j)*alambt/gamarx*te(i+1,j)/ed(i+1,j)*uu(i+1,j)
78
79          alambt=umc2/(c1m1+gen(i+1,j-1)/ed(i+1,j-1))
80          vtmp=v(i+1,j)*wfs(j)+(1.Ø-wfs(j))*v(i+1,j-1)
81          gamarx=1.-alambt*te(i+1,j-1)/ed(i+1,j-1)*vtmp/r(j-1)
82          vis3=den(i+1,j-1)*alambt/gamarx*te(i+1,j-1)/ed(i+1,j-1)
83         1     *uu(i+1,j-1)
84
85          end if
86
87          alambt=umc2/(c1m1+gen(i,j-1)/ed(i,j-1))
88          alamb2=alambt
89          vtmp=v(i,j)*wfs(j)+(1.Ø-wfs(j))*v(i,j-1)
90          gamarx=1.-alambt*te(i,j-1)/ed(i,j-1)*vtmp/r(j-1)
91          vis4=den(i,j-1)*alambt/gamarx*te(i,j-1)/ed(i,j-1)*uu(i,j-1)
92
93          if ( i.eq. 2 ) then
94          vis5=Ø.
95          vis6=Ø.
96          else
97          alambt=umc2/(c1m1+gen(i-1,j-1)/ed(i-1,j-1))
98          vtmp=v(i-1,j)*wfs(j)+(1.Ø-wfs(j))*v(i-1,j-1)
99          gamarx=1.-alambt*te(i-1,j-1)/ed(i-1,j-1)*vtmp/r(j-1)
100         vis5=den(i-1,j-1)*alambt/gamarx*te(i-1,j-1)/ed(i-1,j-1)
101        1     *uu(i-1,j-1)
102
103         alambt=umc2/(c1m1+gen(i-1,j)/ed(i-1,j))
104         vtmp=v(i-1,j)*wfn(j)+(1.Ø-wfn(j))*v(i-1,j+1)
105         gamarx=1.-alambt*te(i-1,j)/ed(i-1,j)*vtmp/r(j)
106         vis6=den(i-1,j)*alambt/gamarx*te(i-1,j)/ed(i-1,j)*uu(i-1,j)
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
107
108
109         end if
110
111            vise=Ø.25*(vis1+vis2+vis3+vis4)+viscos
112            visw=Ø.25*(vis1+vis4+vis5+vis6)+viscos
113
114          visn=2.*den(i,j)*alamb1*te(i,j)/ed(i,j)*vv(i,j)+viscos
115          viss=2.*den(i,j-1)*alamb2*te(i,j-1)/ed(i,j-1)*vv(i,j-1)+viscos
116
117                else
118
119                visn=Ø.
120                viss=Ø.
121                vise=Ø.
122                visw=Ø.
123
124                end if
125
126         dn=visn*arean/dynpv(j)
127         ds=viss*areas/dypsv(j)
128         de=vise*areaew/dxep(i)
129         dw=visw*areaew/dxpw(i)
130
131
132          if ( iupwind  ) then
133
134
135   c---------------------    convection coefficients
136
137         qn=den(i,j)*(v(i,j)*wfn(j)+(1.Ø-wfn(j))*v(i,j+1))
138         qs=den(i,j-1)*(v(i,j)*wfs(j)+(1.Ø-wfs(j))*v(i,j-1))
139         qe=Ø.5*(den(i+1,j)+den(i,j))*u(i+1,j)
140         qse=Ø.5*(den(i,j-1)+den(i+1,j-1))*u(i+1,j-1)
141         qw=Ø.5*(den(i,j)+den(i-1,j))*u(i,j)
142         qsw=Ø.5*(den(i,j-1)+den(i-1,j-1))*u(i,j-1)
143
144         fn=qn*arean
145         fs=qs*areas
146         fe=Ø.5*(qe+qse)*areaew
147         fw=Ø.5*(qw+qsw)*areaew
148
149
150
151
152   c---------------------    main coefficients-HYBRID
153
154         an(i,j)=amax1(dn,-wfn(j)*fn,(1.Ø-wfn(j))*fn)-(1.Ø-wfn(j))*fn
155         as(i,j)=amax1(ds,wfs(j)*fs,-(1.Ø-wfs(j))*fs)+(1.Ø-wfs(j))*fs
156         ae(i,j)=amax1(abs(Ø.5*fe),de)-Ø.5*fe
157         aw(i,j)=amax1(abs(Ø.5*fw),dw)+Ø.5*fw
158
159           smp=(fn-fs+fe-fw)
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
160Ø
161
162           else
163
164   c---------------------    main coefficients-SUDS-RAITHBY
165
166   c---------------------    west
167
168           uwest=Ø.5*(u(i,j)+u(i,j-1))
169           vwest=Ø.5*(v(i,j)+v(i-1,j))
17Ø           rotemp=Ø.25*(den(i,j)+den(i,j-1)+den(i-1,j)+den(i-1,j-1))
171           alw=Ø.5*rotemp*uwest*areaew
172
173                if ( uwest .ge. Ø.Ø ) then
174                isuw=1
175                else
176                isuw=-1
177                end if
178
179                if ( vwest .ge. Ø.Ø ) then
18Ø                isvw=1
181                else
182                isvw=-1
183                end if
184
185           lwi=i-(1+isuw)/2
186           mwi=j-isvw
187           kwi=j+(1-isvw)/2
188
189                dob=rotemp*areaew/dypsv(kwi)*dxpw(i)/4.*abs(vwest)
19Ø                akw(i,j)=float(isuw)*amin1(abs(alw),dob)
191
192                aw(i,j)=dw+(alw-akw(i,j))*float(1+isuw)
193
194   c---------------------    east
195
196           ueast=Ø.5*(u(i+1,j)+u(i+1,j-1))
197           veast=Ø.5*(v(i,j)+v(i+1,j))
198           rotemp=Ø.25*(den(i,j)+den(i,j-1)+den(i+1,j)+den(i+1,j-1))
199           ale=Ø.5*rotemp*ueast*areaew
2ØØ
2Ø1                if ( ueast .ge. Ø.Ø ) then
2Ø2                isue=1
2Ø3                else
2Ø4                isue=-1
2Ø5                end if
2Ø6
2Ø7                if ( veast .ge. Ø.Ø ) then
2Ø8                isve=1
2Ø9                else
21Ø                isve=-1
211                end if
212
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
213           lei=i+(1-isue)/2
214           mei=j-isve
215           kei=j+(1-isve)/2
216
217                 dob=rotemp*areaew/dypsv(kei)*dxep(i)/4*abs(veast)
218                 ake(i,j)=float(isue)*amin1(abs(ale),dob)
219
220                 ae(i,j)=de-(ale-ake(i,j))*float(1-isue)
221
222   c---------------------     north
223
224           unort=wfe(i)*u(i,j)+(1.Ø-wfe(i))*u(i+1,j)
225           vnort=wfn(j)*v(i,j)+(1.Ø-wfn(j))*v(i,j+1)
226           aln=Ø.5*den(i,j)*vnort*arean
227
228                 if ( vnort .ge. Ø.Ø ) then
229                 isvn=1
230                 else
231                 isvn=-1
232                 end if
233
234                 if ( unort .ge. Ø.Ø ) then
235                 isun=1
236                 else
237                 isun=-1
238                 end if
239
240           lni=i-isun
241           mni=j+(1-isvn)/2
242           kni=i+(1-isun)/2
243
244                 dob=den(i,j)*arean/dxpw(kni)*dynpv(j)/4.*abs(unort)
245                 akn(i,j)=float(isvn)*amin1(abs(aln),dob)
246
247                 an(i,j)=dn-(aln-akn(i,j))*float(1-isvn)
248
249   c---------------------     south
250
251           usout=wfe(i)*u(i,j-1)+(1.Ø-wfe(i))*u(i+1,j-1)
252           vsout=wfs(j)*v(i,j)+(1.Ø-wfs(j))*v(i,j-1)
253           als=Ø.5*den(i,j-1)*vsout*areas
254
255                if ( vsout .ge. Ø.Ø ) then
256                isvs=1
257                else
258                isvs=-1
259                end if
260
261                if ( usout .ge. Ø.Ø ) then
262                isus=1
263                else
264                isus=-1
265                end if
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
266
267           lsi=i-isus
268           msi=j-(1+isvs)/2
269           ksi=i+(1-isus)/2
270Ø
271                dob= den(i,j-1)*areas/dxpw(ksi)*dypsv(j)/4.*abs(usout)
272                aks(i,j)=float(isvs)*amin1(abs(als),dob)
273
274                as(i,j)=ds+(als-aks(i,j))*float(1+isvs)
275
276           smp=2.*(aln-als+ale-alw)
277
278
279             end if
280Ø
281
282   c----------------------    source terms
283
284
285           cp=amax1(Ø.Ø,smp)
286
287           dv(i,j)=Ø.5*(arean+areas)
288           dudye=(u(i+1,j)-u(i+1,j-1))/snsv(j)
289           dudyw=(u(i,j)-u(i,j-1))/snsv(j)
290Ø
291                if ( j .gt. jmax(i) ) then
292
293                srrn=Ø.
294                srrs=Ø.
295                srxn=Ø.
296                srxs=Ø.
297
298                else
299
300              utmp1=u(i,j-1)*wfe(i)+(1.Ø-wfe(i))*u(i+1,j-1)
3Ø1              utmp2=u(i,j+1)*wfe(i)+(1.Ø-wfe(i))*u(i+1,j+1)
3Ø2              dudyn=(utmp2-utmp1)/(r(j+1)-r(j-1))
3Ø3
3Ø4              if ( j .eq. 3 ) then
3Ø5              dudys=Ø.
3Ø6              else
3Ø7              utmp2=u(i,j)*wfe(i)+(1.Ø-wfe(i))*u(i+1,j)
3Ø8              utmp1=u(i,j-2)*wfe(i)+(1.Ø-wfe(i))*u(i+1,j-2)
3Ø9              dudys=(utmp2-utmp1)/(r(j)-r(j-2))
310              end if
311
312              dvdyn=(v(i,j+1)-v(i,j))/dynpv(j)
313              dvdys=(v(i,j)-v(i,j-1))/dypsv(j)
314
315
316                  vtmp1=v(i+1,j)*wfn(j)+(1.Ø-wfn(j))*v(i+1,j+1)
317                  vtmp2=v(i-1,j)*wfn(j)+(1.Ø-wfn(j))*v(i-1,j+1)
318                  dvdxn=(vtmp1-vtmp2)/(x(i+1)-x(i-1))
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
319                 vtmp1=v(i+1,j)*wfs(j)+(1.Ø-wfs(j))*v(i+1,j-1)
320Ø                vtmp2=v(i-1,j)*wfs(j)+(1.Ø-wfs(j))*v(i-1,j-1)
321                 dvdxs=(vtmp1-vtmp2)/(x(i+1)-x(i-1))
322                 uvtmp1=uv(i,j)*wfe(i)+(1.-wfe(i))*uv(i+1,j)
323                 vwtmp1=vw(i,j)*wfn(j)+(1.-wfn(j))*vw(i,j+1)
324                 vwtmp2=vw(i,j)*wfs(j)+(1.-wfs(j))*vw(i,j-1)
325
326                 uvtmp2=uv(i,j+1)*wfe(i)+(1.-wfe(i))*uv(i+1,j+1)
327                 uvtmp=uvtmp1*wfn(j)+(1.-wfn(j))*uvtmp2
328
329          srrn=r(j)*den(i,j)*(2.*alamb1*te(i,j)/ed(i,j)*(uvtmp*dvdxn-
330Ø       1     vwtmp1*w(i,j)/r(j)*(1+betarodi)+gen(i,j)/3.)-2.*te(i,j)/3.)
331
332                 uvtmp2=uv(i,j-1)*wfe(i)+(1.-wfe(i))*uv(i+1,j-1)
333                 uvtmp=uvtmp1*wfs(j)+(1.-wfs(j))*uvtmp2
334
335          srrs=r(j-1)*den(i,j-1)*(2.*alamb2*te(i,j-1)/ed(i,j-1)*(uvtmp
336       1     *dvdxs-vwtmp2*w(i,j-1)/r(j-1)*(1+betarodi)+gen(i,j-1)/3.)
337       2     -2.*te(i,j-1)/3.)
338
339
340Ø
341          vtmp=Ø.5*(v(i,j)+v(i+1,j))
342          tetmp=Ø.25*(te(i,j)+te(i+1,j)+te(i,j-1)+te(i+1,j-1))
343          edtmp=Ø.25*(ed(i,j)+ed(i+1,j)+ed(i,j-1)+ed(i+1,j-1))
344          gentmp=Ø.25*(gen(i,j)+gen(i+1,j)+gen(i,j-1)+gen(i+1,j-1))
345          dentmp=Ø.25*(den(i,j)+den(i+1,j)+den(i,j-1)+den(i+1,j-1))
346          wtmp=Ø.25*(w(i,j)+w(i+1,j)+w(i,j-1)+w(i+1,j-1))
347          vvtmp=Ø.25*(vv(i,j)+vv(i+1,j)+vv(i,j-1)+vv(i+1,j-1))
348          alambt=umc2/(c1m1+gentmp/edtmp)
349          gamarx=1.-alambt*tetmp/edtmp*vtmp/rv(j)
350Ø         uwtmp=Ø.5*(uw(i+1,j)+uw(i+1,j-1))
351
352          if ( ( i .eq. ngarm1 ) .and. ( j  .gt. jexit ) ) then
353          srxe=Ø.
354          else
355          srxe=dentmp*alambt/gamarx*tetmp/edtmp*(vvtmp*dudye
356       1      -uwtmp*wtmp/rv(j)*(1+betarodi))
357          end if
358
359          vtmp=Ø.5*(v(i,j)+v(i-1,j))
360Ø         tetmp=Ø.25*(te(i,j)+te(i-1,j)+te(i,j-1)+te(i-1,j-1))
361          edtmp=Ø.25*(ed(i,j)+ed(i-1,j)+ed(i,j-1)+ed(i-1,j-1))
362          gentmp=Ø.25*(gen(i,j)+gen(i-1,j)+gen(i,j-1)+gen(i-1,j-1))
363          dentmp=Ø.25*(den(i,j)+den(i-1,j)+den(i,j-1)+den(i-1,j-1))
364          wtmp=Ø.25*(w(i,j)+w(i-1,j)+w(i,j-1)+w(i-1,j-1))
365          vvtmp=Ø.25*(vv(i,j)+vv(i-1,j)+vv(i,j-1)+vv(i-1,j-1))
366          alambt=umc2/(c1m1+gentmp/edtmp)
367          gamarx=1.-alambt*tetmp/edtmp*vtmp/rv(j)
368          uwtmp=Ø.5*(uw(i,j)+uw(i,j-1))
369
370          if ( i .eq. 2  ) then
371          srxw=Ø.
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
372          else
373          srxw=dentmp*alambt/gamarx*tetmp/edtmp*(vvtmp*dudyw
374         1      -uwtmp*wtmp/rv(j)*(1+betarodi))
375          end if
376
377
378             end if
379
380           sorce1=viscos*(dudye-dudyw)/dxepu(i)
381           sorce2=viscos*(rcv(j)*dvdyn-rcv(j-1)*dvdys)/(rv(j)*snsv(j))
382           sorce3=(srrn-srrs)/(rv(j)*snsv(j))
383           sorce4=(srxe-srxw)/dxepu(i)
384
385              wgos=(w(i,j)+w(i,j-1))*(w(i,j)+w(i,j-1))
386         1          +(ww(i,j)+ww(i,j-1))*(ww(i,j)+ww(i,j-1))
387           restatum=(den(i,j)+den(i,j-1))*wgos/(8.*rv(j))
388
389                if ( iomega .and. j .le. jmax(i) ) then
390           restatum=restatum*(1.+2.*alfag*v(i,j)/sqrt(wgos))
391                end if
392
393           su(i,j)=cp*v(i,j)+dv(i,j)*(p(i,j-1)-p(i,j))
394           su(i,j)=su(i,j)+(sorce1+sorce2+sorce3+sorce4+restatum)*vol
395
396   c---------------------     source modification- suds
397
398          su(i,j)=su(i,j)+2*(akw(i,j)*v(lwi,mwi)+aks(i,j)*v(lsi,msi)
399         1         -ake(i,j)*v(lei,mei)-akn(i,j)*v(lni,mni))
400
401   c---------------------
402
403           sp(i,j)=-cp-2.*viscos*vol/(rv(j)*rv(j))
404
405   1ØØ    continue
406
407
408
409   c2******************      boundary
410
411          call boundary (3)
412
413
414   c3******************      residual source calculation
415
416          resolv=Ø.
417
418          do 3ØØ i=2,nim1
419          do 3ØØ j=3,njm1
420          vol=rv(j)*sew(i)*snsv(j)
421          rait(i,j)=2*(akw(i,j)+aks(i,j)-ake(i,j)-akn(i,j))
422          wgos=(w(i,j)+w(i,j-1))*(w(i,j)+w(i,j-1))
423         1          +(ww(i,j)+ww(i,j-1))*(ww(i,j)+ww(i,j-1))
424          gosman=alfag*sqrt(wgos)/rv(j)*vol*(den(i,j)+den(i,j-1))/4.
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
425
426         ap(i,j)=an(i,j)+as(i,j)+ae(i,j)+aw(i,j)-sp(i,j)+rait(i,j)+gosman
427         dv(i,j)=dv(i,j)/ap(i,j)
428
429         resol=an(i,j)*v(i,j+1)+as(i,j)*v(i,j-1)+ae(i,j)*v(i+1,j)
430        1      +aw(i,j)*v(i-1,j)-ap(i,j)*v(i,j)+su(i,j)
431
432         sorvol=great*vol
433         if(-sp(i,j) .gt. Ø.5*sorvol) resol=resol/sorvol
434         resolv=resolv+abs(resol)
435
436   c---------------------    under-relaxation
437
438         ap(i,j)=ap(i,j)/urfv
439         su(i,j)=su(i,j)+(1.-urfv)*ap(i,j)*v(i,j)
440         dv(i,j)=dv(i,j)*urfv
441   3ØØ   continue
442
443   c4******************      solution
444
445                 if ( istone ) then
446
447                    tetastone=1.85
448                 else
449                    tetastone=1.Ø
450
451                  end if
452
453         do 4ØØ n=1,nswpv
454
455         call stone1 (2,3,ni,jmax,it,jt,v,3)
456
457   4ØØ   continue
458
459   c5******************      simplec approximation
460
461                    if ( isimplec ) then
462
463           do 5ØØ i=2,nim1
464           do 5ØØ j=3,njm1
465
466           gundi=an(i,j)+as(i,j)+aw(i,j)+ae(i,j)
467           dv(i,j)=dv(i,j)*ap(i,j)/(ap(i,j)-gundi)
468
469   5ØØ     continue
470
471                   end if
472
473                      return
474
475                         end
```

Microsoft FORTRAN Optimizing Compiler Version 5.00

CALCV Local Symbols

Name	Class	Type	Size	Offset
WGOS	local	REAL*4	4	0000
ISVN	local	INTEGER*4	4	0004
VISN	local	REAL*4	4	0008
GAMARX	local	REAL*4	4	000c
SRXE	local	REAL*4	4	0010
UEAST	local	REAL*4	4	0014
VEAST	local	REAL*4	4	0018
DVDXN	local	REAL*4	4	001c
DUDYN	local	REAL*4	4	0020
ISUS	local	INTEGER*4	4	0024
RESOL	local	REAL*4	4	0028
GOSMAN	local	REAL*4	4	002c
DVDYN	local	REAL*4	4	0030
SRRN	local	REAL*4	4	0034
ISVS	local	INTEGER*4	4	0038
VISS	local	REAL*4	4	003c
VTMP	local	REAL*4	4	0040
WTMP	local	REAL*4	4	0044
DENTMP	local	REAL*4	4	0048
ISUW	local	INTEGER*4	4	004c
DVDXS	local	REAL*4	4	0050
DUDYS	local	REAL*4	4	0054
ISVW	local	INTEGER*4	4	0058
DE	local	REAL*4	4	005c
VISW	local	REAL*4	4	0060
I	local	INTEGER*4	4	0064
TETMP	local	REAL*4	4	0068
DVDYS	local	REAL*4	4	006c
SRRS	local	REAL*4	4	0070
J	local	INTEGER*4	4	0074
GENTMP	local	REAL*4	4	0078
SRXN	local	REAL*4	4	007c
FE	local	REAL*4	4	0080
UVTMP1	local	REAL*4	4	0084
DUDYW	local	REAL*4	4	0088
N	local	INTEGER*4	4	008c
UVTMP2	local	REAL*4	4	0090
ALAMB1	local	REAL*4	4	0094
VWTMP1	local	REAL*4	4	0098
ALAMB2	local	REAL*4	4	009c
VWTMP2	local	REAL*4	4	00a0
SRXS	local	REAL*4	4	00a4
AKE	local	REAL*4	4608	00a8
ALE	local	REAL*4	4	12a8
DN	local	REAL*4	4	12ac
CP	local	REAL*4	4	12b0
SRXW	local	REAL*4	4	12b4
FN	local	REAL*4	4	12b8
DOB	local	REAL*4	4	12bc

Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

CALCV Local Symbols

Name	Class	Type	Size	Offset
QE.	local	REAL*4	4	12cØ
ROTEMP.	local	REAL*4	4	12c4
DS.	local	REAL*4	4	12c8
UNORT	local	REAL*4	4	12cc
UWEST	local	REAL*4	4	12dØ
VNORT	local	REAL*4	4	12d4
KEI	local	INTEGER*4	4	12d8
VWEST	local	REAL*4	4	12dc
FS.	local	REAL*4	4	12eØ
LEI	local	INTEGER*4	4	12e4
AKN	local	REAL*4	46Ø8	12e8
ALN	local	REAL*4	4	24e8
MEI	local	INTEGER*4	4	24ec
DW.	local	REAL*4	4	24fØ
ALFAG	local	REAL*4	4	24f4
UVTMP	local	REAL*4	4	24f8
UWTMP	local	REAL*4	4	24fc
VVTMP	local	REAL*4	4	25ØØ
FW.	local	REAL*4	4	25Ø4
QN.	local	REAL*4	4	25Ø8
AKS	local	REAL*4	46Ø8	25Øc
ALS	local	REAL*4	4	37Øc
USOUT	local	REAL*4	4	371Ø
VSOUT	local	REAL*4	4	3714
KNI	local	INTEGER*4	4	3718
LNI	local	INTEGER*4	4	371c
VIS1.	local	REAL*4	4	372Ø
AKW	local	REAL*4	46Ø8	3724
MNI	local	INTEGER*4	4	4924
ALW	local	REAL*4	4	4928
QS.	local	REAL*4	4	492c
VIS2.	local	REAL*4	4	493Ø
SORVOL.	local	REAL*4	4	4934
VIS3.	local	REAL*4	4	4938
VIS4.	local	REAL*4	4	493c
KSI	local	INTEGER*4	4	494Ø
VIS5.	local	REAL*4	4	4944
AREAN	local	REAL*4	4	4948
LSI	local	INTEGER*4	4	494c
QW.	local	REAL*4	4	495Ø
VIS6.	local	REAL*4	4	4954
MSI	local	INTEGER*4	4	4958
QSE	local	REAL*4	4	495c
KWI	local	INTEGER*4	4	496Ø
LWI	local	INTEGER*4	4	4964
AREAS	local	REAL*4	4	4968
SORCE1.	local	REAL*4	4	496c
MWI	local	INTEGER*4	4	497Ø

CALCV Local Symbols

Name	Class	Type	Size	Offset
SORCE2	local	REAL*4	4	4974
SORCE3	local	REAL*4	4	4978
SORCE4	local	REAL*4	4	497c
SMP	local	REAL*4	4	498Ø
RAIT	local	REAL*4	46Ø8	4984
ALAMBT	local	REAL*4	4	5b84
VOL	local	REAL*4	4	5b88
RESTATUM	local	REAL*4	4	5b8c
AREAEW	local	REAL*4	4	5b9Ø
ISUE	local	INTEGER*4	4	5b94
GUNDI	local	REAL*4	4	5b98
UTMP1	local	REAL*4	4	5b9c
ISVE	local	INTEGER*4	4	5baØ
VISE	local	REAL*4	4	5ba4
VTMP1	local	REAL*4	4	5ba8
UTMP2	local	REAL*4	4	5bac
VTMP2	local	REAL*4	4	5bbØ
EDTMP	local	REAL*4	4	5bb4
DUDYE	local	REAL*4	4	5bb8
QSW	local	REAL*4	4	5bbc
ISUN	local	INTEGER*4	4	5bcØ
Y	GEOM	REAL*4	96	ØØcØ
DXEP	GEOM	REAL*4	192	Ø12Ø
AP	COEF	REAL*4	46Ø8	ØØØØ
AN	COEF	REAL*4	46Ø8	12ØØ
DXPW	GEOM	REAL*4	192	Ø1eØ
DYNP	GEOM	REAL*4	96	Ø2aØ
AS	COEF	REAL*4	46Ø8	24ØØ
AE	COEF	REAL*4	46Ø8	36ØØ
DYPS	GEOM	REAL*4	96	Ø3ØØ
SNS	GEOM	REAL*4	96	Ø36Ø
AW	COEF	REAL*4	46Ø8	48ØØ
SEW	GEOM	REAL*4	192	Ø3cØ
SU	COEF	REAL*4	46Ø8	5aØØ
XU	GEOM	REAL*4	192	Ø48Ø
SP	COEF	REAL*4	46Ø8	6cØØ
YV	GEOM	REAL*4	96	Ø54Ø
TETASTONE	STONE	REAL*4	4	ØØØØ
R	GEOM	REAL*4	96	Ø5aØ
RV	GEOM	REAL*4	96	Ø6ØØ
SOURCE	STONE	REAL*4	4	ØØØ4
WFN	GEOM	REAL*4	96	Ø66Ø
NITER	STONE	INTEGER*4	4	ØØØc
WFS	GEOM	REAL*4	96	Ø6cØ
UU	STRESS	REAL*4	46Ø8	ØØØØ
WFE	GEOM	REAL*4	192	Ø72Ø
WFW	GEOM	REAL*4	192	Ø7eØ
VV	STRESS	REAL*4	46Ø8	12ØØ

CALCV Local Symbols

Name	Class	Type	Size	Offset
ISTONE.	STONE	LOGICAL*4	4	ØØØ8
WW.	STRESS	REAL*4	46Ø8	24ØØ
RCV	GEOM	REAL*4	96	Ø8aØ
ISIMPLEC.	STONE	LOGICAL*4	4	ØØ1Ø
IOMEGA.	WVEL	LOGICAL*4	4	ØØØc
IFINE	GEOM	INTEGER*4	4	Ø9ØØ
UV.	STRESS	REAL*4	46Ø8	36ØØ
UW.	STRESS	REAL*4	46Ø8	48ØØ
IUPWIND	WIND	LOGICAL*4	4	ØØØØ
VW.	STRESS	REAL*4	46Ø8	5aØØ
URFTEN.	FLUPR	REAL*4	4	ØØØØ
RESOLU.	UVEL	REAL*4	4	ØØØØ
VISCOS.	FLUPR	REAL*4	4	ØØØ4
NSWPU	UVEL	INTEGER*4	4	ØØØ4
DENSIT.	FLUPR	REAL*4	4	ØØØ8
PRANDT.	FLUPR	REAL*4	4	ØØØc
URFU.	UVEL	REAL*4	4	ØØØ8
DEN	FLUPR	REAL*4	46Ø8	ØØ1Ø
DXEPU	UVEL	REAL*4	192	ØØØc
DXPWU	UVEL	REAL*4	192	ØØcc
GEN	TURBU	REAL*4	46Ø8	ØØØØ
SEWU.	UVEL	REAL*4	192	Ø18c
CD.	TURBU	REAL*4	4	12ØØ
CMU	TURBU	REAL*4	4	12Ø4
RESOLV.	VVEL	REAL*4	4	ØØØØ
C1E	TURBU	REAL*4	4	12Ø8
NSWPV	VVEL	INTEGER*4	4	ØØØ4
C2E	TURBU	REAL*4	4	12Øc
URFV.	VVEL	REAL*4	4	ØØØ8
DYNPV	VVEL	REAL*4	96	ØØØc
BETARODI.	TURBU	REAL*4	4	121Ø
CAPPA	TURBU	REAL*4	4	1214
DYPSV	VVEL	REAL*4	96	ØØ6c
ELOG.	TURBU	REAL*4	4	1218
SNSV.	VVEL	REAL*4	96	ØØcc
PRED.	TURBU	REAL*4	4	121c
PRTE.	TURBU	REAL*4	4	122Ø
RESOLW.	WVEL	REAL*4	4	ØØØØ
C1.	TURBU	REAL*4	4	1224
NSWPW	WVEL	INTEGER*4	4	ØØØ4
C2.	TURBU	REAL*4	4	1228
URFW.	WVEL	REAL*4	4	ØØØ8
UMC2.	TURBU	REAL*4	4	122c
C1M1.	TURBU	REAL*4	4	123Ø
RESOLM.	CORRP	REAL*4	4	ØØØØ
NSWPP	CORRP	INTEGER*4	4	ØØØ4
CK.	TURBU	REAL*4	4	1234
URFP.	CORRP	REAL*4	4	ØØØ8

CALCV Local Symbols

Name	Class	Type	Size	Offset
CE3	TURBU	REAL*4	4	1238
DU.	CORRP	REAL*4	46Ø8	ØØØc
UIN	CAUSO	REAL*4	4	ØØØØ
DV.	CORRP	REAL*4	46Ø8	12Øc
IPREF	CORRP	INTEGER*4	4	24Øc
TEIN.	CAUSO	REAL*4	4	ØØØ4
JPREF	CORRP	INTEGER*4	4	241Ø
EDIN.	CAUSO	REAL*4	4	ØØØ8
FLOWIN.	CAUSO	REAL*4	4	ØØØc
ALAMDA.	CAUSO	REAL*4	4	ØØ1Ø
U	VAR	REAL*4	46Ø8	ØØØØ
V	VAR	REAL*4	46Ø8	12ØØ
RSMALL.	CAUSO	REAL*4	4	ØØ14
RLARGE.	CAUSO	REAL*4	4	ØØ18
W	VAR	REAL*4	46Ø8	24ØØ
P	VAR	REAL*4	46Ø8	36ØØ
XTOTA	CAUSO	REAL*4	4	ØØ1c
PP.	VAR	REAL*4	46Ø8	48ØØ
JSTEP	CAUSO	INTEGER*4	4	ØØ2Ø
TE.	VAR	REAL*4	46Ø8	5aØØ
ISTEP	CAUSO	INTEGER*4	4	ØØ24
JSTP1	CAUSO	INTEGER*4	4	ØØ28
ED.	VAR	REAL*4	46Ø8	6cØØ
JSTM1	CAUSO	INTEGER*4	4	ØØ2c
ISTP1	CAUSO	INTEGER*4	4	ØØ3Ø
IT.	GERAL	INTEGER*4	4	ØØØØ
ISTM1	CAUSO	INTEGER*4	4	ØØ34
JT.	GERAL	INTEGER*4	4	ØØØ4
NI.	GERAL	INTEGER*4	4	ØØØ8
JEXIT	CAUSO	INTEGER*4	4	ØØ38
NJ.	GERAL	INTEGER*4	4	ØØØc
JEXITP1	CAUSO	INTEGER*4	4	ØØ3c
NIM1.	GERAL	INTEGER*4	4	ØØ1Ø
INLETI.	CAUSO	INTEGER*4	4	ØØ4Ø
NJM1.	GERAL	INTEGER*4	4	ØØ14
INLETF.	CAUSO	INTEGER*4	4	ØØ44
GREAT	GERAL	REAL*4	4	ØØ18
JINLET.	CAUSO	INTEGER*4	4	ØØ48
RGARG	CAUSO	REAL*4	4	ØØ4c
JMAX.	GERAL	INTEGER*4	192	ØØ1c
JMAXP1.	GERAL	INTEGER*4	192	ØØdc
NGARG	CAUSO	INTEGER*4	4	ØØ5Ø
NGARM1.	CAUSO	INTEGER*4	4	ØØ54
NGARP1.	CAUSO	INTEGER*4	4	ØØ58
X	GEOM	REAL*4	192	ØØØØ

Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

Global Symbols

Name	Class	Type	Size	Offset
BOUNDARY.	extern	***	***	***
CALCV	FSUBRT	***	***	ØØØØ
CAUSO	common	***	92	ØØØØ
COEF.	common	***	32256	ØØØØ
CORRP	common	***	9236	ØØØØ
FLUPR	common	***	4624	ØØØØ
GEOM.	common	***	23Ø8	ØØØØ
GERAL	common	***	412	ØØØØ
STONE	common	***	2Ø	ØØØØ
STONE1.	extern	***	***	***
STRESS.	common	***	27648	ØØØØ
TURBU	common	***	4668	ØØØØ
UVEL.	common	***	588	ØØØØ
VAR	common	***	32256	ØØØØ
VVEL.	common	***	3ØØ	ØØØØ
WIND.	common	***	4	ØØØØ
WVEL.	common	***	16	ØØØØ

Code size = 3416 (13334)
Data size = ØØ48 (72)
Bss size = 5bc4 (23492)

No errors detected

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.00

```
      subroutine  calcw

c********************    preliminaires

      logical iomega,ifine,istone,isimplec

      common
     1/uvel/resolu,nswpu,urfu,dxepu(48),dxpwu(48),sewu(48)
     2/vvel/resolv,nswpv,urfv,dynpv(24),dypsv(24),snsv(24)
     3/wvel/resolw,nswpw,urfw,iomega
     4/var/u(48,24), v(48,24), w(48,24), p(48,24), pp(48,24), te(48,24)
     5     ed(48,24)
     6/geral/it,jt,ni,nj,nim1,njm1,great,jmax(48),jmaxp1(48)
     7/geom/x(48),y(24),dxep(48),dxpw(48),dynp(24),dyps(24),
     8      sns(24),sew(48),xu(48),yv(24),r(24),rv(24),
     9      wfn(24),wfs(24),wfe(48),wfw(48),rcv(24),ifine

      common
     1/fluprr/urften,viscos,densit,prandt,den(48,24)
     2/turbu/gen(48,24),cd,cmu,c1e,c2e,betarodi,cappa,elog,
     3      pred,prte,c1,c2,umc2,c1m1,ck,ce3
     4/causo/uin,tein,edin,flowin,alamda,
     5       rsmall,rlarge,xtota,jstep,istep,jstp1,jstm1,istp1,istm1,
     6       jexit,jexitp1,inleti,inletf,jinlet,rgarg,ngarg,ngarm1,
     7       ngarp1
     8/coef/ap(48,24),an(48,24),as(48,24),ae(48,24),aw(48,24),su(48,24)
     9      sp(48,24)
     1/stone/tetastone,source,istone,niter,isimplec

      common
     1/stress/uu(48,24),vv(48,24),ww(48,24),uv(48,24),
     2        uw(48,24),vw(48,24)


c---------------------    if no swirl return to main program

            if( iomega ) then
               continue
            else
               return
            end if

c1*******************      coefficients

      do 100 i=2,nim1
      do 100 j=2,njm1


c---------------------    areas and volume

      arean=rv(j+1)*sew(i)
      areas=rv(j)*sew(i)
      areaew=rcv(j)*sns(j)
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
54          vol=rcv(j)*sns(j)*sew(i)
55
56
57    c---------------------    diffusion coefficients
58
59                        if ( j  .le. jmax(i) ) then
60Ø
61          vtmp1=v(i+1,j+1)+v(i,j+1)
62          vtmp2=v(i+1,j)+v(i,j)
63          dvdye=Ø.5*(vtmp1-vtmp2)/dynpv(j)
64          getme=Ø.5*(gen(i+1,j)+gen(i,j))
65          tetme=Ø.5*(te(i+1,j)+te(i,j))
66          edtme=Ø.5*(ed(i+1,j)+ed(i,j))
67          alambe=umc2/(c1m1+getme/edtme)
68          gamtxe=1.-alambe*tetme/edtme*dvdye
69
70Ø         if ( ( i .eq. ngarm1) .and. ( j .ge. jexitp1 ) ) then
71          vise=Ø.
72          else
73          vise=Ø.25*(den(i,j)+den(i+1,j))*alambe/gamtxe*tetme/edtme*
74        1      (uu(i,j)+uu(i+1,j))+viscos
75          end if
76
77
78          vtmp1=v(i-1,j+1)+v(i,j+1)
79          vtmp2=v(i-1,j)+v(i,j)
80Ø         dvdyw=Ø.5*(vtmp1-vtmp2)/dynpv(j)
81          getmw=Ø.5*(gen(i-1,j)+gen(i,j))
82          tetmw=Ø.5*(te(i-1,j)+te(i,j))
83          edtmw=Ø.5*(ed(i-1,j)+ed(i,j))
84          alambw=umc2/(c1m1+getmw/edtmw)
85          gamtxw=1.-alambw*tetmw/edtmw*dvdyw
86
87          if ( i .eq. 2 ) then
88          visw=Ø.
89          else
90          visw=Ø.25*(den(i,j)+den(i-1,j))*alambw/gamtxw*tetmw/edtmw*
91        1      (uu(i,j)+uu(i-1,j))+viscos
92          end if
93
94          utmp1=u(i+1,j+1)+u(i+1,j)
95          utmp2=u(i,j+1)+u(i,j)
96          dudxn=Ø.5*(utmp1-utmp2)/dxepu(i)
97          getmn=Ø.5*(gen(i,j+1)+gen(i,j))
98          tetmn=Ø.5*(te(i,j+1)+te(i,j))
99          edtmn=Ø.5*(ed(i,j+1)+ed(i,j))
100         alambn=umc2/(c1m1+getmn/edtmn)
101         gamrtn=1.-alambn*tetmn/edtmn*dudxn
102         visn=Ø.25*(den(i,j)+den(i,j+1))*alambn/gamrtn*tetmn/edtmn*
103       1      (vv(i,j)+vv(i,j+1))+viscos
104
105
106         utmp1=u(i+1,j-1)+u(i+1,j)
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
107         utmp2=u(i,j-1)+u(i,j)
108         dudxs=Ø.5*(utmp1-utmp2)/dxepu(i)
109         getms=Ø.5*(gen(i,j-1)+gen(i,j))
110         tetms=Ø.5*(te(i,j-1)+te(i,j))
111         edtms=Ø.5*(ed(i,j-1)+ed(i,j))
112         alambs=umc2/(c1m1+getms/edtms)
113         gamrts=1.-alambs*tetms/edtms*dudxs
114
115         if ( j .eq. 2) then
116         viss=Ø.
117         else
118         viss=Ø.25*(den(i,j)+den(i,j-1))*alambs/gamrts*tetms/edtms*
119        1      (vv(i,j)+vv(i,j-1))+viscos
120         end if
121
122
123                   dn=visn*arean/dynp(j)
124                   ds=viss*areas/dyps(j)
125                   de=vise*areaew/dxep(i)
126                   dw=visw*areaew/dxpw(i)
127
128
129                   end if
130
131   c---------------------     convection coefficients
132
133         qn=Ø.5*(den(i,j)+den(i,j+1))*v(i,j+1)
134         qs=Ø.5*(den(i,j)+den(i,j-1))*v(i,j)
135         qe=Ø.5*(den(i,j)+den(i+1,j))*u(i+1,j)
136         qw=Ø.5*(den(i,j)+den(i-1,j))*u(i,j)
137
138         fn=qn*arean
139         fs=qs*areas
140         fe=qe*areaew
141         fw=qw*areaew
142
143
144   c---------------------     main coefficients - HYBRID
145
146         an(i,j)=amax1(abs(Ø.5*fn),dn)-Ø.5*fn
147         as(i,j)=amax1(abs(Ø.5*fs),ds)+Ø.5*fs
148         ae(i,j)=amax1(abs(Ø.5*fe),de)-Ø.5*fe
149         aw(i,j)=amax1(abs(Ø.5*fw),dw)+Ø.5*fw
150
151
152   c---------------------     source terms
153
154         smp=fn-fs+fe-fw
155         cp=amax1(Ø.,smp)
156
157                   if ( j .le. jmax(i) ) then
158
159            vavg=Ø.5*(v(i,j+1)+v(i,j))
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
160            vwavg=Ø.5*(vw(i,j+1)+vw(i,j))
161
162
163                  if ( j .eq. 2 ) then
164                  dwdye=(w(i,2)+w(i,3)+w(i+1,2)+w(i+1,3))/(4.*rv(3))
165                  dudye=(u(i+1,3)-u(i+1,2))/(r(3)-r(1))
166                  else
167                  wtmp1=w(i,j+1)+w(i+1,j+1)
168                  wtmp2=w(i,j-1)+w(i+1,j-1)
169                  dwdye=Ø.5*(wtmp1-wtmp2)/(r(j+1)-r(j-1))
170                  dudye=(u(i+1,j+1)-u(i+1,j-1))/(r(j+1)-r(j-1))
171                  end if
172
173            wtmp=Ø.5*(w(i,j)+w(i+1,j))
174            vwtmp=Ø.25*(vw(i,j)+vw(i+1,j)+vw(i,j+1)+vw(i+1,j+1))
175
176            if  ( ( ( i .eq. ngarm1 ) .and. ( j  .gt. jexit ) ) ) then
177            stxe=Ø.
178            else
179            stxe=Ø.5*(den(i,j)+den(i+1,j))*alambe/gamtxe*tetme/edtme*
180          1       (Ø.5*(uv(i+1,j)+uv(i+1,j+1))*(dwdye+betarodi*wtmp/r(j))
181          2       +vwtmp*dudye)
182            end if
183
184                  if ( j .eq. 2 ) then
185                  dwdyw=(w(i,2)+w(i,3)+w(i-1,2)+w(i-1,3))/(4.*rv(3))
186                  dudyw=(u(i,3)-u(i,2))/(r(3)-r(1))
187                  else
188                  wtmp1=w(i,j+1)+w(i-1,j+1)
189                  wtmp2=w(i,j-1)+w(i-1,j-1)
190                  dwdyw=Ø.5*(wtmp1-wtmp2)/(r(j+1)-r(j-1))
191                  dudyw=(u(i,j+1)-u(i,j-1))/(r(j+1)-r(j-1))
192                  end if
193
194            wtmp=Ø.5*(w(i,j)+w(i-1,j))
195            vwtmp=Ø.25*(vw(i,j)+vw(i-1,j)+vw(i,j+1)+vw(i-1,j+1))
196
197            if ( i .eq. 2 ) then
198            stxw=Ø.
199            else
200            stxw=Ø.5*(den(i,j)+den(i-1,j))*alambw/gamtxw*tetmw/edtmw*
201          1       (Ø.5*(uv(i,j)+uv(i,j+1))*(dwdyw+betarodi*wtmp/r(j))
202          2       +vwtmp*dudyw)
203             end if
204
205
206                  wtmp1=w(i+1,j+1)+w(i+1,j)
207                  wtmp2=w(i-1,j+1)+w(i-1,j)
208                  dwdxn=Ø.5*(wtmp1-wtmp2)/(x(i+1)-x(i-1))
209                  dvdxn=(v(i+1,j+1)-v(i-1,j+1))/(x(i+1)-x(i-1))
210                  uvtmp=uv(i,j+1)*wfe(i)+(1.-wfe(i))*uv(i+1,j+1)
211                  uwtmp=Ø.25*(uw(i,j)+uw(i+1,j)+uw(i,j+1)+uw(i+1,j+1))
212                  wtmp=Ø.5*(w(i,j)+w(i,j+1))
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
213
214           if ( (j .eq. jmax(i)) .and. ( i .gt. inletf ) ) then
215           srtn=Ø.
216           else
217           srtn=rv(j+1)*Ø.5*(den(i,j)+den(i,j+1))*alambn/gamrtn*tetmn
218          1      /edtmn*(Ø.5*(vv(i,j+1)+vv(i,j))*betarodi*wtmp/rv(j+1)+
219          2      uvtmp*dwdxn-Ø.5*(ww(i,j+1)+ww(i,j))
220Ø         3      *wtmp/rv(j+1)*(1.+betarodi)+uwtmp*dvdxn)
221            end if
222
223                    wtmp1=w(i+1,j)+w(i+1,j-1)
224                    wtmp2=w(i-1,j)+w(i-1,j-1)
225                    dwdxs=Ø.5*(wtmp1-wtmp2)/(x(i+1)-x(i-1))
226                    dvdxs=(v(i+1,j)-v(i-1,j))/(x(i+1)-x(i-1))
227                    uvtmp=uv(i,j)*wfe(i)+(1.-wfe(i))*uv(i+1,j)
228                    uwtmp=Ø.25*(uw(i,j)+uw(i,j-1)+uw(i+1,j)+uw(i+1,j-1))
229                    wtmp=Ø.5*(w(i,j)+w(i,j-1))
230Ø
231           if (j .eq. 2) then
232           srts=Ø.
233           else
234           srts=Ø.5*(den(i,j)+den(i,j-1))*alambs/gamrts*tetms
235          1      /edtms*(Ø.5*(vv(i,j-1)+vv(i,j))*betarodi*wtmp+
236          2      uvtmp*dwdxs*rv(j)-Ø.5*(ww(i,j-1)
237          3      +ww(i,j))*wtmp*(1.+betarodi)+uwtmp*dvdxs*rv(j))
238            end if
239
240                        end if
241
242           sorce1=-den(i,j)*(vavg*w(i,j)+vwavg)/rcv(j)
243           sorce2=-viscos*w(i,j)/(rcv(j)*rcv(j))
244           sorce3=(stxe-stxw)/sew(i)
245           sorce4=(srtn-srts)/(rcv(j)*dynpv(j))
246
247                su(i,j)=cp*w(i,j)
248                su(i,j)=su(i,j)+(sorce1+sorce2+sorce3+sorce4)*vol
249                sp(i,j)=-cp
250
251
252  1ØØ    continue
253
254
255  c2******************      boundary
256
257          call boundary (5)
258
259  c3******************      residual source calc.
260
261          resolw=Ø.Ø
262
263          do 3ØØ i=2,nim1
264          do 3ØØ j=2,njm1
265
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
266         ap(i,j)=an(i,j)+as(i,j)+ae(i,j)+aw(i,j)-sp(i,j)
267
268         resol=an(i,j)*w(i,j+1)+as(i,j)*w(i,j-1)+ae(i,j)*w(i+1,j)
269        1      +aw(i,j)*w(i-1,j)-ap(i,j)*w(i,j)+su(i,j)
270         vol=rcv(j)*sns(j)*sew(i)
271         sorvol=great*vol
272         if(-sp(i,j) .gt. Ø.5*sorvol) resol=resol/sorvol
273         if(j .le. 2) resol=Ø.Ø
274         resolw=resolw+abs(resol)
275
276
277   c---------------------    under relaxation
278
279         ap(i,j)=ap(i,j)/urfw
280         su(i,j)=su(i,j)+(1.-urfw)*ap(i,j)*w(i,j)
281   3ØØ   continue
282
283
284   c4******************      solution
285
286                 if ( istone ) then
287
288                    tetastone=1.85
289                  else
290                    tetastone=1.Ø
291
292                  end if
293
294         do 4ØØ n=1,nswpw
295
296             call stone1 (2,2,ni,jmax,it,jt,w,5)
297
298   4ØØ   continue
299
3ØØ                   do 5ØØ j=2,jexit
3Ø1                   w(ni,j)=w(nim1,j)
3Ø2   5ØØ             continue
3Ø3
3Ø4                     return
3Ø5
3Ø6                        end
```

CALCW Local Symbols

Name	Class	Type	Size	Offset
GETMS	local	REAL*4	4	ØØØØ
VISN.	local	REAL*4	4	ØØØ4
EDTMW	local	REAL*4	4	ØØØ8
DUDXN	local	REAL*4	4	ØØØc
DVDXN	local	REAL*4	4	ØØ1Ø
STXE.	local	REAL*4	4	ØØ14

CALCW Local Symbols

Name	Class	Type	Size	Offset
GETMW	local	REAL*4	4	ØØ18
RESOL	local	REAL*4	4	ØØ1c
DWDXN	local	REAL*4	4	ØØ2Ø
VISS	local	REAL*4	4	ØØ24
GAMTXE	local	REAL*4	4	ØØ28
SRTN	local	REAL*4	4	ØØ2c
WTMP	local	REAL*4	4	ØØ3Ø
DUDXS	local	REAL*4	4	ØØ34
TETMN	local	REAL*4	4	ØØ38
DVDXS	local	REAL*4	4	ØØ3c
DE	local	REAL*4	4	ØØ4Ø
GAMRTN	local	REAL*4	4	ØØ44
VISW	local	REAL*4	4	ØØ48
I	local	INTEGER*4	4	ØØ4c
DWDXS	local	REAL*4	4	ØØ5Ø
J	local	INTEGER*4	4	ØØ54
VWAVG	local	REAL*4	4	ØØ58
FE	local	REAL*4	4	ØØ5c
SRTS	local	REAL*4	4	ØØ6Ø
DUDYW	local	REAL*4	4	ØØ64
TETMS	local	REAL*4	4	ØØ68
N	local	INTEGER*4	4	ØØ6c
GAMRTS	local	REAL*4	4	ØØ7Ø
DVDYW	local	REAL*4	4	ØØ74
DWDYW	local	REAL*4	4	ØØ78
TETMW	local	REAL*4	4	ØØ7c
DN	local	REAL*4	4	ØØ8Ø
CP	local	REAL*4	4	ØØ84
FN	local	REAL*4	4	ØØ88
STXW	local	REAL*4	4	ØØ8c
QE	local	REAL*4	4	ØØ9Ø
DS	local	REAL*4	4	ØØ94
GAMTXW	local	REAL*4	4	ØØ98
FS	local	REAL*4	4	ØØ9c
DW	local	REAL*4	4	ØØaØ
UVTMP	local	REAL*4	4	ØØa4
UWTMP	local	REAL*4	4	ØØa8
FW	local	REAL*4	4	ØØac
VWTMP	local	REAL*4	4	ØØbØ
QN	local	REAL*4	4	ØØb4
ALAMBE	local	REAL*4	4	ØØb8
QS	local	REAL*4	4	ØØbc
SORVOL	local	REAL*4	4	ØØcØ
AREAN	local	REAL*4	4	ØØc4
QW	local	REAL*4	4	ØØc8
ALAMBN	local	REAL*4	4	ØØcc
AREAS	local	REAL*4	4	ØØdØ
SORCE1	local	REAL*4	4	ØØd4

CALCW Local Symbols

Name	Class	Type	Size	Offset
SORCE2.	local	REAL*4	4	00d8
SORCE3.	local	REAL*4	4	00dc
EDTME	local	REAL*4	4	00e0
SORCE4.	local	REAL*4	4	00e4
SMP	local	REAL*4	4	00e8
ALAMBS.	local	REAL*4	4	00ec
VOL	local	REAL*4	4	00f0
GETME	local	REAL*4	4	00f4
VAVG.	local	REAL*4	4	00f8
ALAMBW.	local	REAL*4	4	00fc
AREAEW.	local	REAL*4	4	0100
UTMP1	local	REAL*4	4	0104
VISE.	local	REAL*4	4	0108
EDTMN	local	REAL*4	4	010c
UTMP2	local	REAL*4	4	0110
VTMP1	local	REAL*4	4	0114
WTMP1	local	REAL*4	4	0118
VTMP2	local	REAL*4	4	011c
WTMP2	local	REAL*4	4	0120
DUDYE	local	REAL*4	4	0124
GETMN	local	REAL*4	4	0128
DVDYE	local	REAL*4	4	012c
DWDYE	local	REAL*4	4	0130
EDTMS	local	REAL*4	4	0134
TETME	local	REAL*4	4	0138
TETASTONE	STONE	REAL*4	4	0000
YV.	GEOM	REAL*4	96	0540
R	GEOM	REAL*4	96	05a0
SOURCE.	STONE	REAL*4	4	0004
RV.	GEOM	REAL*4	96	0600
NITER	STONE	INTEGER*4	4	000c
WFN	GEOM	REAL*4	96	0660
WFS	GEOM	REAL*4	96	06c0
UU.	STRESS	REAL*4	4608	0000
VV.	STRESS	REAL*4	4608	1200
WFE	GEOM	REAL*4	192	0720
WFW	GEOM	REAL*4	192	07e0
WW.	STRESS	REAL*4	4608	2400
RCV	GEOM	REAL*4	96	08a0
UV.	STRESS	REAL*4	4608	3600
UW.	STRESS	REAL*4	4608	4800
VW.	STRESS	REAL*4	4608	5a00
URFTEN.	FLUPR	REAL*4	4	0000
VISCOS.	FLUPR	REAL*4	4	0004
DENSIT.	FLUPR	REAL*4	4	0008
PRANDT.	FLUPR	REAL*4	4	000c
DEN	FLUPR	REAL*4	4608	0010
IOMEGA.	WVEL	LOGICAL*4	4	000c

CALCW Local Symbols

Name	Class	Type	Size	Offset
IFINE	GEOM	LOGICAL*4	4	0900
GEN	TURBU	REAL*4	4608	0000
ISTONE	STONE	LOGICAL*4	4	0008
CD	TURBU	REAL*4	4	1200
CMU	TURBU	REAL*4	4	1204
ISIMPLEC	STONE	LOGICAL*4	4	0010
C1E	TURBU	REAL*4	4	1208
C2E	TURBU	REAL*4	4	120c
RESOLU	UVEL	REAL*4	4	0000
NSWPU	UVEL	INTEGER*4	4	0004
BETARODI	TURBU	REAL*4	4	1210
CAPPA	TURBU	REAL*4	4	1214
URFU	UVEL	REAL*4	4	0008
ELOG	TURBU	REAL*4	4	1218
DXEPU	UVEL	REAL*4	192	000c
PRED	TURBU	REAL*4	4	121c
DXPWU	UVEL	REAL*4	192	00cc
PRTE	TURBU	REAL*4	4	1220
SEWU	UVEL	REAL*4	192	018c
C1	TURBU	REAL*4	4	1224
C2	TURBU	REAL*4	4	1228
RESOLV	VVEL	REAL*4	4	0000
NSWPV	VVEL	INTEGER*4	4	0004
UMC2	TURBU	REAL*4	4	122c
C1M1	TURBU	REAL*4	4	1230
URFV	VVEL	REAL*4	4	0008
DYNPV	VVEL	REAL*4	96	000c
CK	TURBU	REAL*4	4	1234
CE3	TURBU	REAL*4	4	1238
DYPSV	VVEL	REAL*4	96	006c
SNSV	VVEL	REAL*4	96	00cc
UIN	CAUSO	REAL*4	4	0000
TEIN	CAUSO	REAL*4	4	0004
RESOLW	WVEL	REAL*4	4	0000
EDIN	CAUSO	REAL*4	4	0008
NSWPW	WVEL	INTEGER*4	4	0004
FLOWIN	CAUSO	REAL*4	4	000c
URFW	WVEL	REAL*4	4	0008
ALAMDA	CAUSO	REAL*4	4	0010
U	VAR	REAL*4	4608	0000
RSMALL	CAUSO	REAL*4	4	0014
RLARGE	CAUSO	REAL*4	4	0018
V	VAR	REAL*4	4608	1200
W	VAR	REAL*4	4608	2400
XTOTA	CAUSO	REAL*4	4	001c
P	VAR	REAL*4	4608	3600
JSTEP	CAUSO	INTEGER*4	4	0020
PP	VAR	REAL*4	4608	4800

Microsoft FORTRAN Optimizing Compiler Version 5.00

CALCW Local Symbols

Name	Class	Type	Size	Offset
ISTEP	CAUSO	INTEGER*4	4	0024
JSTP1	CAUSO	INTEGER*4	4	0028
TE	VAR	REAL*4	4608	5a00
JSTM1	CAUSO	INTEGER*4	4	002c
ED	VAR	REAL*4	4608	6c00
ISTP1	CAUSO	INTEGER*4	4	0030
ISTM1	CAUSO	INTEGER*4	4	0034
IT	GERAL	INTEGER*4	4	0000
JT	GERAL	INTEGER*4	4	0004
JEXIT	CAUSO	INTEGER*4	4	0038
NI	GERAL	INTEGER*4	4	0008
JEXITP1	CAUSO	INTEGER*4	4	003c
NJ	GERAL	INTEGER*4	4	000c
INLETI	CAUSO	INTEGER*4	4	0040
NIM1	GERAL	INTEGER*4	4	0010
INLETF	CAUSO	INTEGER*4	4	0044
NJM1	GERAL	INTEGER*4	4	0014
JINLET	CAUSO	INTEGER*4	4	0048
GREAT	GERAL	REAL*4	4	0018
RGARG	CAUSO	REAL*4	4	004c
JMAX	GERAL	INTEGER*4	192	001c
NGARG	CAUSO	INTEGER*4	4	0050
JMAXP1	GERAL	INTEGER*4	192	00dc
NGARM1	CAUSO	INTEGER*4	4	0054
NGARP1	CAUSO	INTEGER*4	4	0058
X	GEOM	REAL*4	192	0000
Y	GEOM	REAL*4	96	00c0
AP	COEF	REAL*4	4608	0000
DXEP	GEOM	REAL*4	192	0120
AN	COEF	REAL*4	4608	1200
AS	COEF	REAL*4	4608	2400
DXPW	GEOM	REAL*4	192	01e0
DYNP	GEOM	REAL*4	96	02a0
AE	COEF	REAL*4	4608	3600
AW	COEF	REAL*4	4608	4800
DYPS	GEOM	REAL*4	96	0300
SNS	GEOM	REAL*4	96	0360
SU	COEF	REAL*4	4608	5a00
SEW	GEOM	REAL*4	192	03c0
SP	COEF	REAL*4	4608	6c00
XU	GEOM	REAL*4	192	0480

Global Symbols

Name	Class	Type	Size	Offset
BOUNDARY	extern	***	***	***

Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

Global Symbols

Name	Class	Type	Size	Offset
CALCW	FSUBRT	***	***	ØØØØ
CAUSO	common	***	92	ØØØØ
COEF.	common	***	32256	ØØØØ
FLUPR	common	***	4624	ØØØØ
GEOM.	common	***	23Ø8	ØØØØ
GERAL	common	***	412	ØØØØ
STONE	common	***	2Ø	ØØØØ
STONE1.	extern	***	***	***
STRESS.	common	***	27648	ØØØØ
TURBU	common	***	4668	ØØØØ
UVEL.	common	***	588	ØØØØ
VAR	common	***	32256	ØØØØ
VVEL.	common	***	3ØØ	ØØØØ
WVEL.	common	***	16	ØØØØ

Code size = 1ec8 (7880)
Data size = ØØ34 (52)
Bss size = Ø13c (316)

No errors detected

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
 1
 2        subroutine calcp
 3
 4  cØ******************      preliminaires
 5
 6        logical ifine,istone,isimplec
 7
 8        common
 9       1/corrp/resolm,nswpp,urfp,du(48,24),dv(48,24),ipref,jpref
10       2/var/u(48,24), v(48,24), w(48,24), p(48,24), pp(48,24), te(48,24)
11       3     ed(48,24)
12       4/geral/it,jt,ni,nj,nim1,njm1,great,jmax(48),jmaxp1(48)
13       5/geom/x(48),y(24),dxep(48),dxpw(48),dynp(24),dyps(24),
14       6       sns(24),sew(48),xu(48),yv(24),r(24),rv(24),
15       7       wfn(24),wfs(24),wfe(48),wfw(48),rcv(24),ifine
16
17         common
18       1/flupr/urften,viscos,densit,prandt,den(48,24)
19       2/causo/uin,tein,edin,flowin,alamda,
20       3        rsmall,rlarge,xtota,jstep,istep,jstp1,jstm1,istp1,istm1,
21       4        jexit,jexitp1,inleti,inletf,jinlet,rgarg,ngarg,ngarm1,
22       5        ngarp1
23       6/coef/ap(48,24),an(48,24),as(48,24),ae(48,24),aw(48,24),su(48,24)
24       7       sp(48,24)
25       8/stone/tetastone,source,istone,niter,isimplec
26
27        resolm=Ø.Ø
28
29  c1******************      coefficients
30
31        do 1ØØ i=2,nim1
32        do 1ØØ j=2,jmax(i)
33
34  c---------------------     areas and volumes
35
36        arean=rv(j+1)*sew(i)
37        areas=rv(j)*sew(i)
38        areaew=rcv(j)*sns(j)
39        vol=rcv(j)*sew(i)*sns(j)
40
41  c---------------------     coefficients
42
43        denn=Ø.5*(den(i,j)+den(i,j+1))
44        dens=Ø.5*(den(i,j)+den(i,j-1))
45        dene=Ø.5*(den(i,j)+den(i+1,j))
46        denw=Ø.5*(den(i,j)+den(i-1,j))
47
48           an(i,j)=denn*arean*dv(i,j+1)
49           as(i,j)=dens*areas*dv(i,j)
5Ø           ae(i,j)=dene*areaew*du(i+1,j)
51           aw(i,j)=denw*areaew*du(i,j)
52
53  c---------------------     source terms
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
54
55          fn=denn*v(i,j+1)*arean
56          fs=dens*v(i,j)*areas
57          fe=dene*u(i+1,j)*areaew
58          fw=denw*u(i,j)*areaew
59          smp=fn-fs+fe-fw
60          sp(i,j)=Ø.Ø
61          su(i,j)=-smp
62
63    c---------------------     sum of absolute mass sources
64
65          resolm=resolm+abs(smp)
66
67    1ØØ   continue
68
69    c2******************       boundary
70
71          call boundary (4)
72
73    c3******************       final  assembly
74
75                do 3ØØ i=2,nim1
76                do 3ØØ j=2,njm1
77                ap(i,j)=an(i,j)+as(i,j)+ae(i,j)+aw(i,j)-sp(i,j)
78    3ØØ         continue
79
80    c4******************       solution
81
82                if ( istone ) then
83
84                   tetastone=1.85
85                 else
86                   tetastone=1.Ø
87
88                 end if
89
90          do 4ØØ n=1,nswpp
91
92          call stone1 (2,2,ni,jmax,it,jt,pp,4)
93
94    4ØØ   continue
95
96                      call adjust
97
98             return
99
1ØØ              end
```

CALCP Local Symbols

Name	Class	Type	Size	Offset
I	local	INTEGER*4	4	0000
J	local	INTEGER*4	4	0004
FE	local	REAL*4	4	0008
N	local	INTEGER*4	4	000c
FN	local	REAL*4	4	0010
FS	local	REAL*4	4	0014
DENE	local	REAL*4	4	0018
FW	local	REAL*4	4	001c
DENN	local	REAL*4	4	0020
AREAN	local	REAL*4	4	0024
DENS	local	REAL*4	4	0028
AREAS	local	REAL*4	4	002c
DENW	local	REAL*4	4	0030
SMP	local	REAL*4	4	0034
VOL	local	REAL*4	4	0038
AREAEW	local	REAL*4	4	003c
DENSIT	FLUPR	REAL*4	4	0008
PRANDT	FLUPR	REAL*4	4	000c
DEN	FLUPR	REAL*4	4608	0010
UIN	CAUSO	REAL*4	4	0000
TEIN	CAUSO	REAL*4	4	0004
EDIN	CAUSO	REAL*4	4	0008
FLOWIN	CAUSO	REAL*4	4	000c
ALAMDA	CAUSO	REAL*4	4	0010
RSMALL	CAUSO	REAL*4	4	0014
RLARGE	CAUSO	REAL*4	4	0018
XTOTA	CAUSO	REAL*4	4	001c
JSTEP	CAUSO	INTEGER*4	4	0020
ISTEP	CAUSO	INTEGER*4	4	0024
JSTP1	CAUSO	INTEGER*4	4	0028
IFINE	GEOM	LOGICAL*4	4	0900
JSTM1	CAUSO	INTEGER*4	4	002c
ISTONE	STONE	LOGICAL*4	4	0008
ISTP1	CAUSO	INTEGER*4	4	0030
ISIMPLEC	STONE	LOGICAL*4	4	0010
ISTM1	CAUSO	INTEGER*4	4	0034
RESOLM	CORRP	REAL*4	4	0000
JEXIT	CAUSO	INTEGER*4	4	0038
NSWPP	CORRP	INTEGER*4	4	0004
JEXITP1	CAUSO	INTEGER*4	4	003c
URFP	CORRP	REAL*4	4	0008
INLETI	CAUSO	INTEGER*4	4	0040
DU	CORRP	REAL*4	4608	000c
INLETF	CAUSO	INTEGER*4	4	0044
DV	CORRP	REAL*4	4608	120c
JINLET	CAUSO	INTEGER*4	4	0048
IPREF	CORRP	INTEGER*4	4	240c
RGARG	CAUSO	REAL*4	4	004c
JPREF	CORRP	INTEGER*4	4	2410

CALCP Local Symbols

Name	Class	Type	Size	Offset
NGARG	CAUSO	INTEGER*4	4	Ø05Ø
NGARM1	CAUSO	INTEGER*4	4	Ø054
NGARP1	CAUSO	INTEGER*4	4	Ø058
U	VAR	REAL*4	46Ø8	ØØØØ
V	VAR	REAL*4	46Ø8	12ØØ
W	VAR	REAL*4	46Ø8	24ØØ
AP	COEF	REAL*4	46Ø8	ØØØØ
P	VAR	REAL*4	46Ø8	36ØØ
AN	COEF	REAL*4	46Ø8	12ØØ
PP	VAR	REAL*4	46Ø8	48ØØ
AS	COEF	REAL*4	46Ø8	24ØØ
TE	VAR	REAL*4	46Ø8	5aØØ
AE	COEF	REAL*4	46Ø8	36ØØ
AW	COEF	REAL*4	46Ø8	48ØØ
ED	VAR	REAL*4	46Ø8	6cØØ
SU	COEF	REAL*4	46Ø8	5aØØ
SP	COEF	REAL*4	46Ø8	6cØØ
IT	GERAL	INTEGER*4	4	ØØØØ
JT	GERAL	INTEGER*4	4	ØØØ4
TETASTONE	STONE	REAL*4	4	ØØØØ
NI	GERAL	INTEGER*4	4	ØØØ8
NJ	GERAL	INTEGER*4	4	ØØØc
SOURCE	STONE	REAL*4	4	ØØØ4
NIM1	GERAL	INTEGER*4	4	ØØ1Ø
NITER	STONE	INTEGER*4	4	ØØØc
NJM1	GERAL	INTEGER*4	4	ØØ14
GREAT	GERAL	REAL*4	4	ØØ18
JMAX	GERAL	INTEGER*4	192	ØØ1c
JMAXP1	GERAL	INTEGER*4	192	ØØdc
X	GEOM	REAL*4	192	ØØØØ
Y	GEOM	REAL*4	96	ØØcØ
DXEP	GEOM	REAL*4	192	Ø12Ø
DXPW	GEOM	REAL*4	192	Ø1eØ
DYNP	GEOM	REAL*4	96	Ø2aØ
DYPS	GEOM	REAL*4	96	Ø3ØØ
SNS	GEOM	REAL*4	96	Ø36Ø
SEW	GEOM	REAL*4	192	Ø3cØ
XU	GEOM	REAL*4	192	Ø48Ø
YV	GEOM	REAL*4	96	Ø54Ø
R	GEOM	REAL*4	96	Ø5aØ
RV	GEOM	REAL*4	96	Ø6ØØ
WFN	GEOM	REAL*4	96	Ø66Ø
WFS	GEOM	REAL*4	96	Ø6cØ
WFE	GEOM	REAL*4	192	Ø72Ø
WFW	GEOM	REAL*4	192	Ø7eØ
RCV	GEOM	REAL*4	96	Ø8aØ
URFTEN	FLUPR	REAL*4	4	ØØØØ
VISCOS	FLUPR	REAL*4	4	ØØØ4

Global Symbols

Name	Class	Type	Size	Offset
ADJUST.	extern	***	***	***
BOUNDARY.	extern	***	***	***
CALCP	FSUBRT	***	***	ØØØØ
CAUSO	common	***	92	ØØØØ
COEF.	common	***	32256	ØØØØ
CORRP	common	***	9236	ØØØØ
FLUPR	common	***	4624	ØØØØ
GEOM.	common	***	23Ø8	ØØØØ
GERAL	common	***	412	ØØØØ
STONE	common	***	2Ø	ØØØØ
STONE1.	extern	***	***	***
VAR	common	***	32256	ØØØØ

Code size = Ø5Ø5 (1285)
Data size = ØØ1a (26)
Bss size = ØØ4Ø (64)

No errors detected

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.00

```
 1
 2          subroutine adjust
 3
 4
 5  c0******************       preliminaires
 6
 7          common
 8         1/corrp/resolm,nswpp,urfp,du(48,24),dv(48,24),ipref,jpref
 9         2/var/u(48,24), v(48,24), w(48,24), p(48,24), pp(48,24), te(48,24)
10         3      ed(48,24)
11         4/geral/it,jt,ni,nj,nim1,njm1,great,jmax(48),jmaxp1(48)
12
13           common
14         1/causo/uin,tein,edin,flowin,alamda,
15         2        rsmall,rlarge,xtota,jstep,istep,jstp1,jstm1,istp1,istm1,
16         3        jexit,jexitp1,inleti,inletf,jinlet,rgarg,ngarg,ngarm1,
17         4        ngarp1
18
19
20  c1******************       correct velocities and pressure
21
22  c---------------------      velocities
23
24           do 101 i=2,nim1
25           jj=jmax(i)
26
27             do 100 j=3,jj
28             v(i,j)=v(i,j)+dv(i,j)*(pp(i,j-1)-pp(i,j))
29  100        continue
30
31                  jj=jmax(i-1)
32
33                  do 101 j=2,jj
34                       if(i .ne. 2) then
35                       u(i,j)=u(i,j)+du(i,j)*(pp(i-1,j)-pp(i,j))
36                       else
37                       continue
38                       end if
39  101             continue
40
41  c---------------------      pressures
42
43           ppref=pp(ipref,jpref)
44
45           do 110 i=2,nim1
46             jj=jmax(i)
47
48               do 110 j=2,jj
49
50                p(i,j)=p(i,j)+urfp*(pp(i,j)-ppref)
51
52  110           continue
53
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
54                if( j .le. jmax(i) ) then
55
56            alambt=umc2/(c1m1+gen(i,j)/ed(i,j))
57            alamb1=alambt
58            vtemp=v(i,j)*wfn(j)+(1.-wfn(j))*v(i,j+1)
59            gamarx=1.-alambt*te(i,j)/ed(i,j)*vtemp/r(j)
6Ø            vis1=den(i,j)*alambt/gamarx*te(i,j)/ed(i,j)*vv(i,j)
61
62            if (( i .gt. inletf ) .and. ( j .eq. jmax(i))) then
63
64            vis2=Ø.
65            vis3=Ø.
66
67            else
68
69            alambt=umc2/(c1m1+gen(i,j+1)/ed(i,j+1))
7Ø            vtemp=v(i,j+1)*wfn(j+1)+(1.-wfn(j+1))*v(i,j+2)
71            gamarx=1.-alambt*te(i,j+1)/ed(i,j+1)*vtemp/r(j+1)
72            vis2=den(i,j+1)*alambt/gamarx*te(i,j+1)/ed(i,j+1)*vv(i,j+1)
73
74            alambt=umc2/(c1m1+gen(i-1,j+1)/ed(i-1,j+1))
75            vtemp=v(i-1,j+1)*wfn(j+1)+(1.-wfn(j+1))*v(i-1,j+2)
76            gamarx=1.-alambt*te(i-1,j+1)/ed(i-1,j+1)*vtemp/r(j+1)
77            vis3=den(i-1,j+1)*alambt/gamarx*te(i-1,j+1)/ed(i-1,j+1)
78          1       *vv(i-1,j+1)
79
8Ø            end if
81
82            alambt=umc2/(c1m1+gen(i-1,j)/ed(i-1,j))
83            alamb2=alambt
84            vtemp=v(i-1,j)*wfn(j)+(1.-wfn(j))*v(i-1,j+1)
85            gamarx=1.-alambt*te(i-1,j)/ed(i-1,j)*vtemp/r(j)
86            vis4=den(i-1,j)*alambt/gamarx*te(i-1,j)/ed(i-1,j)*vv(i-1,j)
87
88            if ( j  .eq. 2 ) then
89
9Ø            vis5=Ø.
91            vis6=Ø.
92
93            else
94
95            alambt=umc2/(c1m1+gen(i-1,j-1)/ed(i-1,j-1))
96            vtemp=v(i-1,j-1)*wfn(j-1)+(1.-wfn(j-1))*v(i-1,j)
97            gamarx=1.-alambt*te(i-1,j-1)/ed(i-1,j-1)*vtemp/r(j-1)
98            vis5=den(i-1,j-1)*alambt/gamarx*te(i-1,j-1)/ed(i-1,j-1)
99          1       *vv(i-1,j-1)
1ØØ
101
1Ø2           alambt=umc2/(c1m1+gen(i,j-1)/ed(i,j-1))
1Ø3           vtemp=v(i,j-1)*wfn(j-1)+(1.-wfn(j-1))*v(i,j)
1Ø4           gamarx=1.-alambt*te(i,j-1)/ed(i,j-1)*vtemp/r(j-1)
1Ø5           vis6=den(i,j-1)*alambt/gamarx*te(i,j-1)/ed(i,j-1)*vv(i,j-1)
1Ø6
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.00

```
54                  return
55
56                  end
```

ADJUST Local Symbols

Name	Class	Type	Size	Offset
I	local	INTEGER*4	4	0000
J	local	INTEGER*4	4	0004
JJ	local	INTEGER*4	4	0008
PPREF	local	REAL*4	4	000c
RESOLM	CORRP	REAL*4	4	0000
NSWPP	CORRP	INTEGER*4	4	0004
URFP	CORRP	REAL*4	4	0008
DU	CORRP	REAL*4	4608	000c
DV	CORRP	REAL*4	4608	120c
IPREF	CORRP	INTEGER*4	4	240c
JPREF	CORRP	INTEGER*4	4	2410
U	VAR	REAL*4	4608	0000
V	VAR	REAL*4	4608	1200
W	VAR	REAL*4	4608	2400
P	VAR	REAL*4	4608	3600
PP	VAR	REAL*4	4608	4800
TE	VAR	REAL*4	4608	5a00
ED	VAR	REAL*4	4608	6c00
IT	GERAL	INTEGER*4	4	0000
JT	GERAL	INTEGER*4	4	0004
NI	GERAL	INTEGER*4	4	0008
NJ	GERAL	INTEGER*4	4	000c
NIM1	GERAL	INTEGER*4	4	0010
NJM1	GERAL	INTEGER*4	4	0014
GREAT	GERAL	REAL*4	4	0018
JMAX	GERAL	INTEGER*4	192	001c
JMAXP1	GERAL	INTEGER*4	192	00dc
UIN	CAUSO	REAL*4	4	0000
TEIN	CAUSO	REAL*4	4	0004
EDIN	CAUSO	REAL*4	4	0008
FLOWIN	CAUSO	REAL*4	4	000c
ALAMDA	CAUSO	REAL*4	4	0010
RSMALL	CAUSO	REAL*4	4	0014
RLARGE	CAUSO	REAL*4	4	0018
XTOTA	CAUSO	REAL*4	4	001c
JSTEP	CAUSO	INTEGER*4	4	0020
ISTEP	CAUSO	INTEGER*4	4	0024
JSTP1	CAUSO	INTEGER*4	4	0028
JSTM1	CAUSO	INTEGER*4	4	002c
ISTP1	CAUSO	INTEGER*4	4	0030
ISTM1	CAUSO	INTEGER*4	4	0034
JEXIT	CAUSO	INTEGER*4	4	0038
JEXITP1	CAUSO	INTEGER*4	4	003c
INLETI	CAUSO	INTEGER*4	4	0040

Microsoft FORTRAN Optimizing Compiler Version 5.Ø0

ADJUST Local Symbols

Name	Class	Type	Size	Offset
INLETF.	CAUSO	INTEGER*4	4	Ø044
JINLET.	CAUSO	INTEGER*4	4	Ø048
RGARG	CAUSO	REAL*4	4	Ø04c
NGARG	CAUSO	INTEGER*4	4	Ø050
NGARM1.	CAUSO	INTEGER*4	4	Ø054
NGARP1.	CAUSO	INTEGER*4	4	Ø058

Global Symbols

Name	Class	Type	Size	Offset
ADJUST.	FSUBRT	***	***	ØØØØ
CAUSO	common	***	92	ØØØØ
CORRP	common	***	9236	ØØØØ
GERAL	common	***	412	ØØØØ
VAR	common	***	32256	ØØØØ

Code size = Ø365 (869)
Data size = ØØØ6 (6)
Bss size = ØØ1Ø (16)

No errors detected

```
Line#  Source Line      Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

 1
 2        subroutine calcte
 3
 4  cØ*******************      preliminaries
 5
 6        logical ifine,istone,isimplec,iold
 7
 8        common
 9       1/tenerg/resolk,nswpk,urfk,iold
10       2/var/u(48,24), v(48,24), w(48,24), p(48,24), pp(48,24), te(48,24)
11       3     ed(48,24)
12       4/geral/it,jt,ni,nj,nim1,njm1,great,jmax(48),jmaxp1(48)
13       5/geom/x(48),y(24),dxep(48),dxpw(48),dynp(24),dyps(24),
14       6       sns(24),sew(48),xu(48),yv(24),r(24),rv(24),
15       7       wfn(24),wfs(24),wfe(48),wfw(48),rcv(24),ifine
16       8/stone/tetastone,source,istone,niter,isimplec
17
18         common
19       1/flupr/urften,viscos,densit,prandt,den(48,24)
20       2/turbu/gen(48,24),cd,cmu,c1e,c2e,betarodi,cappa,elog,
21       3       pred,prte,c1,c2,umc2,c1m1,ck,ce3
22       4/causo/uin,tein,edin,flowin,alamda,
23       5        rsmall,rlarge,xtota,jstep,istep,jstp1,jstm1,istp1,istm1,
24       6        jexit,jexitp1,inleti,inletf,jinlet,rgarg,ngarg,ngarm1,
25       7        ngarp1
26       8/coef/ap(48,24),an(48,24),as(48,24),ae(48,24),aw(48,24),au(48,24)
27       9       sp(48,24)
28       1/susptmp/sukd(48,24),spkd(48,24)
29       2/old/genold(48,24)
30
31        common
32       1/stress/uu(48,24),vv(48,24),ww(48,24),uv(48,24),
33       2         uw(48,24),vw(48,24)
34       3/boundst/genuv(48,24),genuv1(48,24),genuv2(48,24),genuv3(48,24)
35
36
37  c---------------------
38
39                  if ( iold ) then
40
41                      do 1Ø j=2,jexit
42                      genold(ni,j)=gen(nim1,j)
43  1Ø                  continue
44
45                       do 2Ø i=2,nim1
46                       do 2Ø j=2,jmax(i)
47
48                       genold(i,j)=gen(i,j)
49
50  2Ø                   continue
51
52                  end if
53
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
54  c---------------------
55
56
57  c1******************      coefficients
58
59
60
61        do 1ØØ i=2,nim1
62        do 1ØØ j=2,njm1
63
64
65  c---------------------     areas and volume
66
67        arean=rv(j+1)*sew(i)
68        areas=rv(j)*sew(i)
69        areaew=rcv(j)*sns(j)
7Ø        vol=rcv(j)*sns(j)*sew(i)
71
72  c---------------------     diffusion coefficients
73
74                  if (  j  .le. jmax(i) ) then
75
76         dentmp=Ø.5*(den(i,j)+den(i+1,j))
77         tetmp=Ø.5*(te(i,j)+te(i+1,j))
78         edtmp=Ø.5*(ed(i,j)+ed(i+1,j))
79         uutmp=Ø.5*(uu(i,j)+uu(i+1,j))
80               gamae=ck*dentmp*tetmp/edtmp*uutmp+viscos/prte
81
82         dentmp=Ø.5*(den(i,j)+den(i-1,j))
83         tetmp=Ø.5*(te(i,j)+te(i-1,j))
84         edtmp=Ø.5*(ed(i,j)+ed(i-1,j))
85         uutmp=Ø.5*(uu(i,j)+uu(i-1,j))
86               gamaw=ck*dentmp*tetmp/edtmp*uutmp+viscos/prte
87
88         dentmp=Ø.5*(den(i,j)+den(i,j+1))
89         tetmp=Ø.5*(te(i,j)+te(i,j+1))
9Ø         edtmp=Ø.5*(ed(i,j)+ed(i,j+1))
91         vvtmp=Ø.5*(vv(i,j)+vv(i,j+1))
92               gaman=ck*dentmp*tetmp/edtmp*vvtmp+viscos/prte
93
94         dentmp=Ø.5*(den(i,j)+den(i,j-1))
95         tetmp=Ø.5*(te(i,j)+te(i,j-1))
96         edtmp=Ø.5*(ed(i,j)+ed(i,j-1))
97         vvtmp=Ø.5*(vv(i,j)+vv(i,j-1))
98               gamas=ck*dentmp*tetmp/edtmp*vvtmp+viscos/prte
99
1ØØ
1Ø1                        dn=gaman*arean/dynp(j)
1Ø2                        ds=gamas*areas/dyps(j)
1Ø3                        de=gamae*areaew/dxep(i)
1Ø4                        dw=gamaw*areaew/dxpw(i)
1Ø5
1Ø6               end if
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
107
108  c---------------------    convection coefficients
109
110         qn=Ø.5*(den(i,j)+den(i,j+1))*v(i,j+1)
111         qs=Ø.5*(den(i,j)+den(i,j-1))*v(i,j)
112         qe=Ø.5*(den(i,j)+den(i+1,j))*u(i+1,j)
113         qw=Ø.5*(den(i,j)+den(i-1,j))*u(i,j)
114
115         fn=qn*arean
116         fs=qs*areas
117         fe=qe*areaew
118         fw=qw*areaew
119
120
121  c---------------------    main coefficients-HYBRID
122
123
124               an(i,j)=amax1(abs(Ø.5*fn),dn)-Ø.5*fn
125               as(i,j)=amax1(abs(Ø.5*fs),ds)+Ø.5*fs
126               ae(i,j)=amax1(abs(Ø.5*fe),de)-Ø.5*fe
127               aw(i,j)=amax1(abs(Ø.5*fw),dw)+Ø.5*fw
128
129
130
131  c---------------------     source terms
132
133         smp=fn-fs+fe-fw
134         cp=amax1(Ø.Ø,smp)
135
136                   if ( j .le. jmax(i) ) then
137
138         dudx=(u(i+1,j)-u(i,j))/sew(i)
139         dvdy=(v(i,j+1)-v(i,j))/sns(j)
140         if ( j .eq. 2 ) then
141         dudy=Ø.
142         else
143         dudy=(u(i,j+1)+u(i+1,j+1)-u(i,j-1)-u(i+1,j-1))/(4*sns(j))
144         end if
145         dvdx=(v(i+1,j)+v(i+1,j+1)-v(i-1,j)-v(i-1,j+1))/(4*sew(i))
146         dwdy=(w(i,j+1)-w(i,j-1))/(dynp(j)+dyps(j))
147         dwdx=(w(i+1,j)-w(i-1,j))/(dxpw(i)+dxep(i))
148
149               if ( rv(j) .eq. Ø ) then
150               vtmpdr=Ø.5*v(i,j+1)/rv(j+1)
151               else
152               vtmpdr=Ø.5*(v(i,j)/rv(j)+v(i,j+1)/rv(j+1))
153               end if
154
155          gast1=uv(i,j)*(u(i,j)-u(i,j-1))/(y(j)-y(j-1))
156          gast2=uv(i+1,j)*(u(i+1,j)-u(i+1,j-1))/(y(j)-y(j-1))
157          gast3=uv(i,j+1)*(u(i,j+1)-u(i,j))/(y(j+1)-y(j))
158          gast4=uv(i+1,j+1)*(u(i+1,j+1)-u(i+1,j))/(y(j+1)-y(j))
159
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
160Ø
161          gast11=uv(i,j)*(v(i,j)-v(i-1,j))/(x(i)-x(i-1))
162          gast21=uv(i+1,j)*(v(i+1,j)-v(i,j))/(x(i+1)-x(i))
163          gast31=uv(i,j+1)*(v(i,j+1)-v(i-1,j+1))/(x(i)-x(i-1))
164          gast41=uv(i+1,j+1)*(v(i+1,j+1)-v(i,j+1))/(x(i+1)-x(i))
165
166
167
168                  genuv1(i,j)=Ø.25*((gast1+gast11)+(gast3+gast31)+
169        1                           (gast2+gast21)+(gast4+gast41))
170
171
172               wdr1=w(i,j)/r(j)
173               wdr2=w(i,j-1)/r(j-1)
174               wdr3=rv(j)*(wdr1-wdr2)/(r(j)-r(j-1))
175               gast1=vw(i,j)*wdr3
176
177               wdr1=w(i,j+1)/r(j+1)
178               wdr2=w(i,j)/r(j)
179               wdr3=rv(j+1)*(wdr1-wdr2)/(r(j+1)-r(j))
180               gast2=vw(i,j+1)*wdr3
181
182
183                   genuv2(i,j)=Ø.5*((gast1)+(gast2))
184
185
186               gast1=uw(i,j)*(w(i,j)-w(i-1,j))/(x(i)-x(i-1))
187               gast2=uw(i+1,j)*(w(i+1,j)-w(i,j))/(x(i+1)-x(i))
188
189
190                   genuv3(i,j)=Ø.5*((gast1)+(gast2))
191
192
193
194          gen(i,j)=-((vv(i,j)*dvdy)+(uu(i,j)*dudx)+genuv(i,j)
195        1            +genuv1(i,j)+(ww(i,j)*vtmpdr)+genuv2(i,j)
196        2            +genuv3(i,j))
197
198
199
200                 if ( j  .eq. 2 ) then
201
202             dkdye=(te(i,3)+te(i+1,3)-te(i,2)-te(i+1,2))/(4.*rv(3))
203             dkdyw=(te(i,3)+te(i-1,3)-te(i,2)-te(i-1,2))/(4.*rv(3))
204
205             ttmp1=te(i+1,3)+te(i+1,2)
206             ttmp2=te(i-1,3)+te(i-1,2)
207             dkdxn=Ø.5*(ttmp1-ttmp2)/(x(i+1)-x(i-1))
208
209             dkdxs=(te(i+1,2)-te(i-1,2))/(x(i+1)-x(i-1))
210
211
212                 else
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
213
214
215                ttmp1=te(i+1,j+1)+te(i,j+1)
216                ttmp2=te(i+1,j-1)+te(i,j-1)
217                dkdye=Ø.5*(ttmp1-ttmp2)/(r(j+1)-r(j-1))
218
219                ttmp1=te(i-1,j+1)+te(i,j+1)
220                ttmp2=te(i-1,j-1)+te(i,j-1)
221                dkdyw=Ø.5*(ttmp1-ttmp2)/(r(j+1)-r(j-1))
222
223
224                ttmp1=te(i+1,j+1)+te(i+1,j)
225                ttmp2=te(i-1,j+1)+te(i-1,j)
226                dkdxn=Ø.5*(ttmp1-ttmp2)/(x(i+1)-x(i-1))
227
228                ttmp1=te(i+1,j-1)+te(i+1,j)
229                ttmp2=te(i-1,j-1)+te(i-1,j)
230                dkdxs=Ø.5*(ttmp1-ttmp2)/(x(i+1)-x(i-1))
231
232
233                end if
234
235             if ( j .eq. 2 ) then
236            skss=Ø.
237             else
238            skss=Ø.25*rv(j)*ck*(den(i,j)+den(i,j-1))*(te(i,j)+te(i,j-1))/
239          1     (ed(i,j)+ed(i,j-1))*(uv(i,j)+uv(i+1,j))*dkdxs
240             end if
241
242             if ( i .eq. 2 ) then
243            skxw=Ø.
244             else
245            skxw=Ø.25*ck*(den(i,j)+den(i-1,j))*(te(i,j)+te(i-1,j))/(ed(i,j)+
246          1     ed(i-1,j))*(uv(i,j)+uv(i,j+1))*dkdyw
247             end if
248
249             if ( ( i .eq. ngarm1 ) .and. ( j .gt. jexit ) ) then
250            skxe=Ø.
251             else
252            skxe=Ø.25*ck*(den(i,j)+den(i+1,j))*(te(i,j)+te(i+1,j))/(ed(i,j)+
253          1     ed(i+1,j))*(uv(i+1,j)+uv(i+1,j+1))*dkdye
254             end if
255
256             if ( ( j .eq. jmax(i) ) .and. ( i .gt. inletf ) ) then
257            sksn=Ø.
258             else
259            sksn=Ø.25*rv(j+1)*ck*(den(i,j)+den(i,j+1))*(te(i,j)+te(i,j+1))/
260          1     (ed(i,j)+ed(i,j+1))*(uv(i,j+1)+uv(i+1,j+1))*dkdxn
261             end if
262
263
264             sorce1=(skxe-skxw)/sew(i)
265             sorce2=(sksn-skss)/(sns(j)*rcv(j))
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
266
267
268           if ( ( i .eq. 2 ) .or. ( (i .eq. ngarm1) .and.
269          1        ( j .gt. jexit) ) .or. ( ( j .eq. jmax(i) ) .and.
270Ø       2        ( i .gt. inletf) ) ) then
271
272             sukd(i,j)=cp*te(i,j)+(sorce1+sorce2)*vol
273           else
274             sukd(i,j)=cp*te(i,j)+(sorce1+sorce2-den(i,j)*ed(i,j))*vol
275             su(i,j)=sukd(i,j)+den(i,j)*gen(i,j)*vol
276           end if
277
278           spkd(i,j)=-cp
279           sp(i,j)=spkd(i,j)
280
281
282                end if
283
284   1ØØ   continue
285
286   c2******************      boundary
287
288         call boundary (6)
289
290   c3******************      residual source calc.
291
292         resolk=Ø.Ø
293
294         do 3ØØ i=2,nim1
295         do 3ØØ j=2,njm1
296         ap(i,j)=an(i,j)+as(i,j)+ae(i,j)+aw(i,j)-sp(i,j)
297         resol=an(i,j)*te(i,j+1)+as(i,j)*te(i,j-1)+ae(i,j)*te(i+1,j)
298        1     +aw(i,j)*te(i-1,j)-ap(i,j)*te(i,j)+su(i,j)
299
300         vol=rcv(j)*sns(j)*sew(i)
301         sorvol=great*vol
302
303
304         if(-sp(i,j) .gt. Ø.5*sorvol) resol=resol/sorvol
305         resolk=resolk+abs(resol)
306
307   c--------------------     under relaxation
308
309         ap(i,j)=ap(i,j)/urfk
310         su(i,j)=su(i,j)+(1.-urfk)*ap(i,j)*te(i,j)
311   3ØØ   continue
312
313   c4******************      solution
314
315                if (istone) then
316
317                  tetastone=1.85
318                else
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.00

```
319                    tetastone=1.Ø
320Ø
321                  end if
322
323          do 4ØØ n=1,nswpk
324
325          call stone1 (2,2,ni,jmax,it,jt,te,6)
326
327   4ØØ    continue
328
329   c---------------------     exit update
330Ø
331          do 42Ø j=2,jexit
332          gen(ni,j)=gen(nim1,j)
333          te(ni,j)=te(nim1,j)
334   42Ø    continue
335
336   c---------------------
337
338                  if ( .not. iold ) then
339
340Ø                    do 5ØØ i=2,ni
341                    do 5ØØ j=2,jmax(i)
342
343                    genold(i,j)=gen(i,j)
344
345   5ØØ              continue
346
347                   end if
348
349   c---------------------
350Ø
351                 return
352
353                     end
```

CALCTE Local Symbols

Name	Class	Type	Size	Offset
DKDYW	local	REAL*4	4	ØØØØ
SKSS	local	REAL*4	4	ØØØ4
RESOL	local	REAL*4	4	ØØØ8
DENTMP	local	REAL*4	4	ØØØc
DE	local	REAL*4	4	ØØ1Ø
I	local	INTEGER*4	4	ØØ14
TETMP	local	REAL*4	4	ØØ18
J	local	INTEGER*4	4	ØØ1c
FE	local	REAL*4	4	ØØ2Ø
SKXW	local	REAL*4	4	ØØ24
N	local	INTEGER*4	4	ØØ28
GAST11	local	REAL*4	4	ØØ2c

Microsoft FORTRAN Optimizing Compiler Version 5.Ø0

CALCTE Local Symbols

Name	Class	Type	Size	Offset
GAST21	local	REAL*4	4	Ø03Ø
DN	local	REAL*4	4	Ø034
GAST31	local	REAL*4	4	Ø038
CP	local	REAL*4	4	Ø03c
GAST41	local	REAL*4	4	Ø04Ø
FN	local	REAL*4	4	Ø044
QE	local	REAL*4	4	Ø048
DS	local	REAL*4	4	Ø04c
FS	local	REAL*4	4	Ø05Ø
DW	local	REAL*4	4	Ø054
GAMAE	local	REAL*4	4	Ø058
UUTMP	local	REAL*4	4	Ø05c
VTMPDR	local	REAL*4	4	Ø06Ø
FW	local	REAL*4	4	Ø064
VVTMP	local	REAL*4	4	Ø068
WDR1	local	REAL*4	4	Ø06c
WDR2	local	REAL*4	4	Ø07Ø
QN	local	REAL*4	4	Ø074
WDR3	local	REAL*4	4	Ø078
GAST1	local	REAL*4	4	Ø07c
GAST2	local	REAL*4	4	Ø08Ø
GAST3	local	REAL*4	4	Ø084
GAST4	local	REAL*4	4	Ø088
QS	local	REAL*4	4	Ø08c
GAMAN	local	REAL*4	4	Ø09Ø
SORVOL	local	REAL*4	4	Ø094
AREAN	local	REAL*4	4	Ø098
QW	local	REAL*4	4	Ø09c
GAMAS	local	REAL*4	4	Ø0aØ
AREAS	local	REAL*4	4	Ø0a4
SORCE1	local	REAL*4	4	Ø0a8
GAMAW	local	REAL*4	4	Ø0ac
SORCE2	local	REAL*4	4	Ø0bØ
SMP	local	REAL*4	4	Ø0b4
DKDYE	local	REAL*4	4	Ø0b8
VOL	local	REAL*4	4	Ø0bc
DUDX	local	REAL*4	4	Ø0cØ
AREAEW	local	REAL*4	4	Ø0c4
TTMP1	local	REAL*4	4	Ø0c8
DVDX	local	REAL*4	4	Ø0cc
DUDY	local	REAL*4	4	Ø0dØ
TTMP2	local	REAL*4	4	Ø0d4
DWDX	local	REAL*4	4	Ø0d8
DVDY	local	REAL*4	4	Ø0dc
DWDY	local	REAL*4	4	Ø0eØ
DKDXN	local	REAL*4	4	Ø0e4
EDTMP	local	REAL*4	4	Ø0e8
SKXE	local	REAL*4	4	Ø0ec

CALCTE Local Symbols

Name	Class	Type	Size	Offset
DKDXS	local	REAL*4	4	ØØfØ
SKSN	local	REAL*4	4	ØØf4
VW	STRESS	REAL*4	46Ø8	5aØØ
VISCOS	FLUPR	REAL*4	4	ØØØ4
DENSIT	FLUPR	REAL*4	4	ØØØ8
PRANDT	FLUPR	REAL*4	4	ØØØc
GENUV	BOUNDST	REAL*4	46Ø8	ØØØØ
GENUV1	BOUNDST	REAL*4	46Ø8	12ØØ
DEN	FLUPR	REAL*4	46Ø8	ØØ1Ø
GENUV2	BOUNDST	REAL*4	46Ø8	24ØØ
GENUV3	BOUNDST	REAL*4	46Ø8	36ØØ
GEN	TURBU	REAL*4	46Ø8	ØØØØ
CD	TURBU	REAL*4	4	12ØØ
CMU	TURBU	REAL*4	4	12Ø4
C1E	TURBU	REAL*4	4	12Ø8
C2E	TURBU	REAL*4	4	12Øc
BETARODI	TURBU	REAL*4	4	121Ø
CAPPA	TURBU	REAL*4	4	1214
ELOG	TURBU	REAL*4	4	1218
PRED	TURBU	REAL*4	4	121c
PRTE	TURBU	REAL*4	4	122Ø
IFINE	GEOM	LOGICAL*4	4	Ø9ØØ
C1	TURBU	REAL*4	4	1224
ISTONE	STONE	LOGICAL*4	4	ØØØ8
C2	TURBU	REAL*4	4	1228
ISIMPLEC	STONE	LOGICAL*4	4	ØØ1Ø
IOLD	TENERG	LOGICAL*4	4	ØØØc
UMC2	TURBU	REAL*4	4	122c
C1M1	TURBU	REAL*4	4	123Ø
RESOLK	TENERG	REAL*4	4	ØØØØ
CK	TURBU	REAL*4	4	1234
CE3	TURBU	REAL*4	4	1238
NSWPK	TENERG	INTEGER*4	4	ØØØ4
URFK	TENERG	REAL*4	4	ØØØ8
UIN	CAUSO	REAL*4	4	ØØØØ
TEIN	CAUSO	REAL*4	4	ØØØ4
U	VAR	REAL*4	46Ø8	ØØØØ
EDIN	CAUSO	REAL*4	4	ØØØ8
V	VAR	REAL*4	46Ø8	12ØØ
W	VAR	REAL*4	46Ø8	24ØØ
FLOWIN	CAUSO	REAL*4	4	ØØØc
ALAMDA	CAUSO	REAL*4	4	ØØ1Ø
P	VAR	REAL*4	46Ø8	36ØØ
PP	VAR	REAL*4	46Ø8	48ØØ
RSMALL	CAUSO	REAL*4	4	ØØ14
RLARGE	CAUSO	REAL*4	4	ØØ18
TE	VAR	REAL*4	46Ø8	5aØØ
XTOTA	CAUSO	REAL*4	4	ØØ1c

Microsoft FORTRAN Optimizing Compiler Version 5.00

CALCTE Local Symbols

Name	Class	Type	Size	Offset
ED.	VAR	REAL*4	4608	6c00
JSTEP	CAUSO	INTEGER*4	4	0020
IT.	GERAL	INTEGER*4	4	0000
ISTEP	CAUSO	INTEGER*4	4	0024
JSTP1	CAUSO	INTEGER*4	4	0028
JT.	GERAL	INTEGER*4	4	0004
JSTM1	CAUSO	INTEGER*4	4	002c
NI.	GERAL	INTEGER*4	4	0008
ISTP1	CAUSO	INTEGER*4	4	0030
NJ.	GERAL	INTEGER*4	4	000c
ISTM1	CAUSO	INTEGER*4	4	0034
NIM1.	GERAL	INTEGER*4	4	0010
NJM1.	GERAL	INTEGER*4	4	0014
JEXIT	CAUSO	INTEGER*4	4	0038
GREAT	GERAL	REAL*4	4	0018
JEXITP1	CAUSO	INTEGER*4	4	003c
JMAX.	GERAL	INTEGER*4	192	001c
INLETI.	CAUSO	INTEGER*4	4	0040
JMAXP1.	GERAL	INTEGER*4	192	00dc
INLETF.	CAUSO	INTEGER*4	4	0044
JINLET.	CAUSO	INTEGER*4	4	0048
RGARG	CAUSO	REAL*4	4	004c
X	GEOM	REAL*4	192	0000
NGARG	CAUSO	INTEGER*4	4	0050
Y	GEOM	REAL*4	96	00c0
DXEP.	GEOM	REAL*4	192	0120
NGARM1.	CAUSO	INTEGER*4	4	0054
NGARP1.	CAUSO	INTEGER*4	4	0058
DXPW.	GEOM	REAL*4	192	01e0
DYNP.	GEOM	REAL*4	96	02a0
AP.	COEF	REAL*4	4608	0000
DYPS.	GEOM	REAL*4	96	0300
SNS	GEOM	REAL*4	96	0360
AN.	COEF	REAL*4	4608	1200
SEW	GEOM	REAL*4	192	03c0
AS.	COEF	REAL*4	4608	2400
XU.	GEOM	REAL*4	192	0480
AE.	COEF	REAL*4	4608	3600
YV.	GEOM	REAL*4	96	0540
AW.	COEF	REAL*4	4608	4800
SU.	COEF	REAL*4	4608	5a00
R	GEOM	REAL*4	96	05a0
RV.	GEOM	REAL*4	96	0600
SP.	COEF	REAL*4	4608	6c00
WFN	GEOM	REAL*4	96	0660
SUKD.	SUSPTMP	REAL*4	4608	0000
WFS	GEOM	REAL*4	96	06c0
SPKD.	SUSPTMP	REAL*4	4608	1200

Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

CALCTE Local Symbols

Name	Class	Type	Size	Offset
WFE	GEOM	REAL*4	192	Ø72Ø
WFW	GEOM	REAL*4	192	Ø7eØ
GENOLD	OLD	REAL*4	46Ø8	ØØØØ
RCV	GEOM	REAL*4	96	Ø8aØ
TETASTONE	STONE	REAL*4	4	ØØØØ
UU	STRESS	REAL*4	46Ø8	ØØØØ
VV	STRESS	REAL*4	46Ø8	12ØØ
SOURCE	STONE	REAL*4	4	ØØØ4
WW	STRESS	REAL*4	46Ø8	24ØØ
NITER	STONE	INTEGER*4	4	ØØØc
UV	STRESS	REAL*4	46Ø8	36ØØ
UW	STRESS	REAL*4	46Ø8	48ØØ
URFTEN	FLUPR	REAL*4	4	ØØØØ

Global Symbols

Name	Class	Type	Size	Offset
BOUNDARY	extern	***	***	***
BOUNDST	common	***	18432	ØØØØ
CALCTE	FSUBRT	***	***	ØØØØ
CAUSO	common	***	92	ØØØØ
COEF	common	***	32256	ØØØØ
FLUPR	common	***	4624	ØØØØ
GEOM	common	***	23Ø8	ØØØØ
GERAL	common	***	412	ØØØØ
OLD	common	***	46Ø8	ØØØØ
STONE	common	***	2Ø	ØØØØ
STONE1	extern	***	***	***
STRESS	common	***	27648	ØØØØ
SUSPTMP	common	***	9216	ØØØØ
TENERG	common	***	16	ØØØØ
TURBU	common	***	4668	ØØØØ
VAR	common	***	32256	ØØØØ

Code size = 22c5 (89Ø1)
Data size = ØØ32 (5Ø)
Bss size = ØØf8 (248)

No errors detected

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
      subroutine calced

cØ******************      preliminaries

      logical ifine,istone,isimplec

      common
     1/tdissp/resole,nswpd,urfe
     2/geral/it,jt,ni,nj,nim1,njm1,great,jmax(48),jmaxp1(48)
     3/geom/x(48),y(24),dxep(48),dxpw(48),dynp(24),dyps(24),
     4      sns(24),sew(48),xu(48),yv(24),r(24),rv(24),
     5      wfn(24),wfs(24),wfe(48),wfw(48),rcv(24),ifine
     6/stone/tetastone,source,istone,niter,isimplec

      common
     1/fluprl/urften,viscos,densit,prandt,den(48,24)
     2/turbu/gen(48,24),cd,cmu,c1e,c2e,betarodi,cappa,elog,
     3      pred,prte,c1,c2,umc2,c1m1,ck,ce3
     4/causo/uin,tein,edin,flowin,alamda,
     5       rsmall,rlarge,xtota,jstep,istep,jstp1,jstm1,istp1,istm1,
     6       jexit,jexitp1,inleti,inletf,jinlet,rgarg,ngarg,ngarm1,
     7       ngarp1
     8/coef/ap(48,24),an(48,24),as(48,24),ae(48,24),aw(48,24),su(48,24)
     9      sp(48,24)
     1/var/u(48,24), v(48,24), w(48,24), p(48,24), pp(48,24), te(48,24)
     2     ed(48,24)
     3/susptmp/sukd(48,24),spkd(48,24)

      common
     1/stress/uu(48,24),vv(48,24),ww(48,24),uv(48,24),
     2        uw(48,24),vw(48,24)


c1******************      coefficients


      do 1ØØ i=2,nim1
      do 1ØØ j=2,njm1


c---------------------     areas and volume

      arean=rv(j+1)*sew(i)
      areas=rv(j)*sew(i)
      areaew=rcv(j)*sns(j)
      vol=rcv(j)*sns(j)*sew(i)

c---------------------     diffusion coefficients

                if ( j  .le. jmax(i) ) then

       dentmp=Ø.5*(den(i,j)+den(i+1,j))
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
54          tetmp=Ø.5*(te(i,j)+te(i+1,j))
55          edtmp=Ø.5*(ed(i,j)+ed(i+1,j))
56          uutmp=Ø.5*(uu(i,j)+uu(i+1,j))
57                gamae=ce3*dentmp*tetmp/edtmp*uutmp+viscos/pred
58
59          dentmp=Ø.5*(den(i,j)+den(i-1,j))
6Ø          tetmp=Ø.5*(te(i,j)+te(i-1,j))
61          edtmp=Ø.5*(ed(i,j)+ed(i-1,j))
62          uutmp=Ø.5*(uu(i,j)+uu(i-1,j))
63                gamaw=ce3*dentmp*tetmp/edtmp*uutmp+viscos/pred
64
65          dentmp=Ø.5*(den(i,j)+den(i,j+1))
66          tetmp=Ø.5*(te(i,j)+te(i,j+1))
67          edtmp=Ø.5*(ed(i,j)+ed(i,j+1))
68          vvtmp=Ø.5*(vv(i,j)+vv(i,j+1))
69                gaman=ce3*dentmp*tetmp/edtmp*vvtmp+viscos/pred
7Ø
71          dentmp=Ø.5*(den(i,j)+den(i,j-1))
72          tetmp=Ø.5*(te(i,j)+te(i,j-1))
73          edtmp=Ø.5*(ed(i,j)+ed(i,j-1))
74          vvtmp=Ø.5*(vv(i,j)+vv(i,j-1))
75                gamas=ce3*dentmp*tetmp/edtmp*vvtmp+viscos/pred
76
77
78                               dn=gaman*arean/dynp(j)
79                               ds=gamas*areas/dyps(j)
8Ø                               de=gamae*areaew/dxep(i)
81                               dw=gamaw*areaew/dxpw(i)
82
83                  end if
84
85    c----------------------    convection coefficients
86
87          qn=Ø.5*(den(i,j)+den(i,j+1))*v(i,j+1)
88          qs=Ø.5*(den(i,j)+den(i,j-1))*v(i,j)
89          qe=Ø.5*(den(i,j)+den(i+1,j))*u(i+1,j)
9Ø          qw=Ø.5*(den(i,j)+den(i-1,j))*u(i,j)
91
92          fn=qn*arean
93          fs=qs*areas
94          fe=qe*areaew
95          fw=qw*areaew
96
97
98    c----------------------    main coefficients-HYBRID
99
1ØØ                an(i,j)=amax1(abs(Ø.5*fn),dn)-Ø.5*fn
1Ø1                as(i,j)=amax1(abs(Ø.5*fs),ds)+Ø.5*fs
1Ø2                ae(i,j)=amax1(abs(Ø.5*fe),de)-Ø.5*fe
1Ø3                aw(i,j)=amax1(abs(Ø.5*fw),dw)+Ø.5*fw
1Ø4
1Ø5   c----------------------    source terms
1Ø6
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
107            smp=fn-fs+fe-fw
108            cp=amax1(Ø.Ø,smp)
109
110                  if ( j .le. jmax(i) ) then
111
112                  if ( j  .eq. 2 ) then
113
114              dedye=(ed(i,3)+ed(i+1,3)-ed(i,2)-ed(i+1,2))/(4.*rv(3))
115              dedyw=(ed(i,3)+ed(i-1,3)-ed(i,2)-ed(i-1,2))/(4.*rv(3))
116
117              etmp1=ed(i+1,3)+ed(i+1,2)
118              etmp2=ed(i-1,3)+ed(i-1,2)
119              dedxn=Ø.5*(etmp1-etmp2)/(x(i+1)-x(i-1))
120
121              dedxs=(ed(i+1,2)-ed(i-1,2))/(x(i+1)-x(i-1))
122
123
124                  else
125
126
127                  etmp1=ed(i+1,j+1)+ed(i,j+1)
128                  etmp2=ed(i+1,j-1)+ed(i,j-1)
129                  dedye=Ø.5*(etmp1-etmp2)/(r(j+1)-r(j-1))
130
131                  etmp1=ed(i-1,j+1)+ed(i,j+1)
132                  etmp2=ed(i-1,j-1)+ed(i,j-1)
133                  dedyw=Ø.5*(etmp1-etmp2)/(r(j+1)-r(j-1))
134
135                  etmp1=ed(i+1,j+1)+ed(i+1,j)
136                  etmp2=ed(i-1,j+1)+ed(i-1,j)
137                  dedxn=Ø.5*(etmp1-etmp2)/(x(i+1)-x(i-1))
138
139                  etmp1=ed(i+1,j-1)+ed(i+1,j)
140                  etmp2=ed(i-1,j-1)+ed(i-1,j)
141                  dedxs=Ø.5*(etmp1-etmp2)/(x(i+1)-x(i-1))
142
143                  end if
144
145
146             if ( j .eq. 2 ) then
147            edss=Ø.
148             else
149            edss=Ø.5*rv(j)*ce3*(den(i,j)+den(i,j-1))*(te(i,j)+te(i,j-1))/
150           1      (ed(i,j)+ed(i,j-1))*(uv(i,j)+uv(i+1,j))*dedxs
151             end if
152
153             if ( i .eq. 2 ) then
154            edxw=Ø.
155             else
156            edxw=Ø.25*ce3*(den(i,j)+den(i-1,j))*(te(i,j)+te(i-1,j))/
157           1       (ed(i,j)+ed(i-1,j))*(uv(i,j)+uv(i,j+1))*dedyw
158             end if
159
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
160           if ( ( i .eq. ngarm1 ) .and. ( j .gt. jexit ) ) then
161         edxe=Ø.
162           else
163         edxe=Ø.25*ce3*(den(i,j)+den(i+1,j))*(te(i,j)+te(i+1,j))/
164        1       (ed(i,j)+ed(i+1,j))*(uv(i+1,j)+uv(i+1,j+1))*dedye
165           end if
166
167           if ( ( j .eq. jmax(i) ) .and. ( i .gt. inletf ) ) then
168         edsn=Ø.
169           else
170         edsn=Ø.25*rv(j+1)*ce3*(den(i,j)+den(i,j+1))*(te(i,j)+te(i,j+1))/
171        1       (ed(i,j)+ed(i,j+1))*(uv(i,j+1)+uv(i+1,j+1))*dedxn
172           end if
173
174
175         sukd(i,j)=cp*ed(i,j)
176         sorce1=(edxe-edxw)/sew(i)
177         sorce2=(edsn-edss)/(sns(j)*rcv(j))
178         sorce3=den(i,j)*(c1e*gen(i,j)*ed(i,j)/te(i,j)-
179        1          Ø.5*c2e*ed(i,j)*ed(i,j)/te(i,j))
180
181         su(i,j)=sukd(i,j)+(sorce1+sorce2+sorce3)*vol
182
183         spkd(i,j)=-cp
184         sp(i,j)=spkd(i,j)-Ø.5*c2e*ed(i,j)*den(i,j)*vol/te(i,j)
185
186               end if
187
188   1ØØ   continue
189
190
191   c2*****************       boundary
192
193         call boundary (7)
194
195
196   c3*****************     residual source calc.
197
198         resole=Ø.Ø
199
200         do 3ØØ i=2,nim1
201         do 3ØØ j=2,njm1
202         ap(i,j)=an(i,j)+as(i,j)+ae(i,j)+aw(i,j)-sp(i,j)
203         resol=an(i,j)*ed(i,j+1)+as(i,j)*ed(i,j-1)+ae(i,j)*ed(i+1,j)
204        1      +aw(i,j)*ed(i-1,j)-ap(i,j)*ed(i,j)+su(i,j)
205
206         vol=rcv(j)*sns(j)*sew(i)
207         sorvol=great*vol
208         if(-sp(i,j) .gt. Ø.5*sorvol) resol=resol/sorvol
209         resole=resole+abs(resol)
210
211   c---------------------     under relaxation
212
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
213         ap(i,j)=ap(i,j)/urfe
214         su(i,j)=su(i,j)+(1.-urfe)*ap(i,j)*ed(i,j)
215
216   3ØØ   continue
217
218   c4*****************       solution
219
22Ø                 if (istone) then
221
222                   tetastone=1.85
223                 else
224                   tetastone=1.Ø
225
226                 end if
227
228         do 4ØØ n=1,nswpd
229
23Ø         call stone1 (2,2,ni,jmax,it,jt,ed,7)
231
232   4ØØ   continue
233
234
235   c---------------------     exit update
236
237
238             do 42Ø j=2,jexit
239             ed(ni,j)=ed(nim1,j)
240   42Ø       continue
241
242
243                   return
244
245                         end
```

CALCED Local Symbols

Name	Class	Type	Size	Offset
RESOL	local	REAL*4	4	ØØØØ
DENTMP	local	REAL*4	4	ØØØ4
DE	local	REAL*4	4	ØØØ8
I	local	INTEGER*4	4	ØØØc
TETMP	local	REAL*4	4	ØØ1Ø
J	local	INTEGER*4	4	ØØ14
FE	local	REAL*4	4	ØØ18
N	local	INTEGER*4	4	ØØ1c
DN	local	REAL*4	4	ØØ2Ø
CP	local	REAL*4	4	ØØ24
FN	local	REAL*4	4	ØØ28
QE	local	REAL*4	4	ØØ2c
DS	local	REAL*4	4	ØØ3Ø
FS	local	REAL*4	4	ØØ34

CALCED Local Symbols

Name	Class	Type	Size	Offset
DW.	local	REAL*4	4	ØØ38
GAMAE	local	REAL*4	4	ØØ3c
UUTMP	local	REAL*4	4	ØØ4Ø
FW.	local	REAL*4	4	ØØ44
VVTMP	local	REAL*4	4	ØØ48
QN.	local	REAL*4	4	ØØ4c
QS.	local	REAL*4	4	ØØ5Ø
GAMAN	local	REAL*4	4	ØØ54
SORVOL.	local	REAL*4	4	ØØ58
EDXE.	local	REAL*4	4	ØØ5c
ETMP1	local	REAL*4	4	ØØ6Ø
AREAN	local	REAL*4	4	ØØ64
ETMP2	local	REAL*4	4	ØØ68
QW.	local	REAL*4	4	ØØ6c
GAMAS	local	REAL*4	4	ØØ7Ø
EDSN.	local	REAL*4	4	ØØ74
DEDYE	local	REAL*4	4	ØØ78
AREAS	local	REAL*4	4	ØØ7c
SORCE1.	local	REAL*4	4	ØØ8Ø
GAMAW	local	REAL*4	4	ØØ84
SORCE2.	local	REAL*4	4	ØØ88
SORCE3.	local	REAL*4	4	ØØ8c
EDSS.	local	REAL*4	4	ØØ9Ø
SMP	local	REAL*4	4	ØØ94
VOL	local	REAL*4	4	ØØ98
DEDXN	local	REAL*4	4	ØØ9c
AREAEW.	local	REAL*4	4	ØØaØ
EDXW.	local	REAL*4	4	ØØa4
DEDXS	local	REAL*4	4	ØØa8
EDTMP	local	REAL*4	4	ØØac
DEDYW	local	REAL*4	4	ØØbØ
C2E	TURBU	REAL*4	4	12Øc
BETARODI.	TURBU	REAL*4	4	121Ø
CAPPA	TURBU	REAL*4	4	1214
ELOG.	TURBU	REAL*4	4	1218
PRED.	TURBU	REAL*4	4	121c
PRTE.	TURBU	REAL*4	4	122Ø
C1.	TURBU	REAL*4	4	1224
C2.	TURBU	REAL*4	4	1228
UMC2.	TURBU	REAL*4	4	122c
C1M1.	TURBU	REAL*4	4	123Ø
CK.	TURBU	REAL*4	4	1234
CE3	TURBU	REAL*4	4	1238
UIN	CAUSO	REAL*4	4	ØØØØ
TEIN.	CAUSO	REAL*4	4	ØØØ4
IFINE	GEOM	LOGICAL*4	4	Ø9ØØ
EDIN.	CAUSO	REAL*4	4	ØØØ8
ISTONE.	STONE	LOGICAL*4	4	ØØØ8

CALCED Local Symbols

Name	Class	Type	Size	Offset
ISIMPLEC.	STONE	LOGICAL*4	4	ØØ1Ø
FLOWIN.	CAUSO	REAL*4	4	ØØØc
ALAMDA.	CAUSO	REAL*4	4	ØØ1Ø
RSMALL.	CAUSO	REAL*4	4	ØØ14
RESOLE.	TDISSP	REAL*4	4	ØØØØ
RLARGE.	CAUSO	REAL*4	4	ØØ18
NSWPD	TDISSP	INTEGER*4	4	ØØØ4
URFE.	TDISSP	REAL*4	4	ØØØ8
XTOTA	CAUSO	REAL*4	4	ØØ1c
JSTEP	CAUSO	INTEGER*4	4	ØØ2Ø
IT.	GERAL	INTEGER*4	4	ØØØØ
ISTEP	CAUSO	INTEGER*4	4	ØØ24
JSTP1	CAUSO	INTEGER*4	4	ØØ28
JT.	GERAL	INTEGER*4	4	ØØØ4
JSTM1	CAUSO	INTEGER*4	4	ØØ2c
NI.	GERAL	INTEGER*4	4	ØØØ8
ISTP1	CAUSO	INTEGER*4	4	ØØ3Ø
NJ.	GERAL	INTEGER*4	4	ØØØc
ISTM1	CAUSO	INTEGER*4	4	ØØ34
NIM1.	GERAL	INTEGER*4	4	ØØ1Ø
NJM1.	GERAL	INTEGER*4	4	ØØ14
JEXIT	CAUSO	INTEGER*4	4	ØØ38
GREAT	GERAL	REAL*4	4	ØØ18
JEXITP1	CAUSO	INTEGER*4	4	ØØ3c
JMAX.	GERAL	INTEGER*4	192	ØØ1c
INLETI.	CAUSO	INTEGER*4	4	ØØ4Ø
JMAXP1.	GERAL	INTEGER*4	192	ØØdc
INLETF.	CAUSO	INTEGER*4	4	ØØ44
JINLET.	CAUSO	INTEGER*4	4	ØØ48
RGARG	CAUSO	REAL*4	4	ØØ4c
X	GEOM	REAL*4	192	ØØØØ
NGARG	CAUSO	INTEGER*4	4	ØØ5Ø
Y	GEOM	REAL*4	96	ØØc0
DXEP.	GEOM	REAL*4	192	Ø12Ø
NGARM1.	CAUSO	INTEGER*4	4	ØØ54
NGARP1.	CAUSO	INTEGER*4	4	ØØ58
DXPW.	GEOM	REAL*4	192	Ø1eØ
DYNP.	GEOM	REAL*4	96	Ø2aØ
AP.	COEF	REAL*4	46Ø8	ØØØØ
DYPS.	GEOM	REAL*4	96	Ø3ØØ
SNS	GEOM	REAL*4	96	Ø36Ø
AN.	COEF	REAL*4	46Ø8	12ØØ
SEW	GEOM	REAL*4	192	Ø3cØ
AS.	COEF	REAL*4	46Ø8	24ØØ
XU.	GEOM	REAL*4	192	Ø48Ø
AE.	COEF	REAL*4	46Ø8	36ØØ
YV.	GEOM	REAL*4	96	Ø54Ø
AW.	COEF	REAL*4	46Ø8	48ØØ

CALCED Local Symbols

Name	Class	Type	Size	Offset
SU.	COEF	REAL*4	4608	5a00
R	GEOM	REAL*4	96	05a0
RV.	GEOM	REAL*4	96	0600
SP.	COEF	REAL*4	4608	6c00
WFN	GEOM	REAL*4	96	0660
WFS	GEOM	REAL*4	96	06c0
U	VAR	REAL*4	4608	0000
WFE	GEOM	REAL*4	192	0720
V	VAR	REAL*4	4608	1200
WFW	GEOM	REAL*4	192	07e0
W	VAR	REAL*4	4608	2400
RCV	GEOM	REAL*4	96	08a0
P	VAR	REAL*4	4608	3600
PP.	VAR	REAL*4	4608	4800
TETASTONE	STONE	REAL*4	4	0000
TE.	VAR	REAL*4	4608	5a00
SOURCE.	STONE	REAL*4	4	0004
ED.	VAR	REAL*4	4608	6c00
NITER	STONE	INTEGER*4	4	000c
SUKD.	SUSPTMP	REAL*4	4608	0000
SPKD.	SUSPTMP	REAL*4	4608	1200
URFTEN.	FLUPR	REAL*4	4	0000
VISCOS.	FLUPR	REAL*4	4	0004
UU.	STRESS	REAL*4	4608	0000
DENSIT.	FLUPR	REAL*4	4	0008
VV.	STRESS	REAL*4	4608	1200
PRANDT.	FLUPR	REAL*4	4	000c
WW.	STRESS	REAL*4	4608	2400
DEN	FLUPR	REAL*4	4608	0010
UV.	STRESS	REAL*4	4608	3600
UW.	STRESS	REAL*4	4608	4800
GEN	TURBU	REAL*4	4608	0000
VW.	STRESS	REAL*4	4608	5a00
CD.	TURBU	REAL*4	4	1200
CMU	TURBU	REAL*4	4	1204
C1E	TURBU	REAL*4	4	1208

Global Symbols

Name	Class	Type	Size	Offset
BOUNDARY.	extern	***	***	***
CALCED.	FSUBRT	***	***	0000
CAUSO	common	***	92	0000
COEF.	common	***	32256	0000
FLUPR	common	***	4624	0000
GEOM.	common	***	2308	0000

Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

Global Symbols

Name	Class	Type	Size	Offset
GERAL	common	***	412	ØØØØ
STONE	common	***	2Ø	ØØØØ
STONE1	extern	***	***	***
STRESS	common	***	27648	ØØØØ
SUSPTMP	common	***	9216	ØØØØ
TDISSP	common	***	12	ØØØØ
TURBU	common	***	4668	ØØØØ
VAR	common	***	32256	ØØØØ

Code size = 1568 (548Ø)
Data size = ØØ2a (42)
Bss size = ØØb4 (18Ø)

No errors detected

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
      subroutine props


c------------------ algebraic Reynolds Stress - Rodi Gibson Launder


       common
     1/var/u(48,24), v(48,24), w(48,24), p(48,24), pp(48,24), te(48,24)
     2      ed(48,24)
     3/geral/it,jt,ni,nj,nim1,njm1,great,jmax(48),jmaxp1(48)
     4/geom/x(48),y(24),dxep(48),dxpw(48),dynp(24),dyps(24),
     5       sns(24),sew(48),xu(48),yv(24),r(24),rv(24),
     6       wfn(24),wfs(24),wfe(48),wfw(48),rcv(24),ifine
     7/flupr/urften,viscos,densit,prandt,den(48,24)
     8/turbu/gen(48,24),cd,cmu,c1e,c2e,betarodi,cappa,elog,
     9       pred,prte,c1,c2,umc2,c1m1,ck,ce3

       common
     1/stress/uu(48,24),vv(48,24),ww(48,24),uv(48,24),
     2         uw(48,24),vw(48,24)
     3/old/genold(48,24)
     4/localstress/uvc(48,24),uwc(48,24),vwc(48,24)

c------------------

           dimension agau(7,7)
           tasort=1.-urften
           ngauss=6
           ngp1=ngauss+1
           epsgauss=1e-1Ø

c------------------

           do 1ØØ i=2,nim1
           do 1ØØ j=2,jmax(i)


           dudx=(u(i+1,j)-u(i,j))/sew(i)
           dvdy=(v(i,j+1)-v(i,j))/sns(j)

              if ( j .eq. 2 ) then
       dudy=Ø.
              else
       dudy=(u(i,j+1)+u(i+1,j+1)-u(i,j-1)-u(i+1,j-1))/(4*sns(j))
              end if

       dvdx=(v(i+1,j)+v(i+1,j+1)-v(i-1,j)-v(i-1,j+1))/(4*sew(i))


           dwdy=(w(i,j+1)-w(i,j-1))/(dynp(j)+dyps(j))
           dwdx=(w(i+1,j)-w(i-1,j))/(dxpw(i)+dxep(i))
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
54              if ( rv(j) .eq. Ø. ) then
55              vtmpdr=Ø.5*v(i,j)/rv(j+1)
56              else
57              vtmpdr=Ø.5*(v(i,j)/rv(j)+v(i,j+1)/rv(j+1))
58              end if
59
6Ø              alambt=umc2/(c1m1+genold(i,j)/ed(i,j))
61              foin=alambt*te(i,j)/ed(i,j)
62              foin1=2.*foin/3.
63
64
65  c-------------       uu
66
67              agau(1,1)=1.+2.*foin1*dudx
68              agau(1,2)=-foin1*dvdy
69              agau(1,3)=-foin1*vtmpdr
7Ø              agau(1,4)=-(dvdx-2.*dudy)*foin1
71              agau(1,5)=-foin1*dwdx
72              agau(1,6)=-foin1*(dwdy-w(i,j)/r(j))
73              agau(1,7)=2.*te(i,j)/3.
74
75
76  c-------------       vv
77
78              agau(2,1)=-foin1*dudx
79              agau(2,2)=1.+2*foin1*dvdy
8Ø              agau(2,3)=-foin1*vtmpdr
81              agau(2,4)=-(dudy-2.*dvdx)*foin1
82              agau(2,5)=-foin1*dwdx
83              agau(2,6)=-foin1*(dwdy+w(i,j)/r(j)*(2.+3.*betarodi))
84              agau(2,7)=2*te(i,j)/3.
85
86
87  c-------------       ww
88
89               agau(3,1)=-foin1*dudx
9Ø               agau(3,2)=-foin1*dvdy
91               agau(3,3)=1.+2.*foin1*vtmpdr
92               agau(3,4)=-(dvdx+dudy)*foin1
93               agau(3,5)=2.*foin1*dwdx
94               agau(3,6)=foin1*(2.*dwdy+(1.+3.*betarodi)*w(i,j)/r(j))
95               agau(3,7)=2*te(i,j)/3.
96
97
98  c-------------       uvc
99
1ØØ              agau(4,1)=foin*dvdx
1Ø1              agau(4,2)=foin*dudy
1Ø2              agau(4,3)=Ø.
1Ø3              agau(4,4)=1.-foin*vtmpdr
1Ø4              agau(4,5)=-foin*w(i,j)/r(j)*(1.+betarodi)
1Ø5              agau(4,6)=Ø.
1Ø6              agau(4,7)=Ø.
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
107
108  c-------------      uwc
109
110               agau(5,1)=foin*dwdx
111               agau(5,2)=Ø.
112               agau(5,3)=Ø.
113               agau(5,4)=foin*(dwdy+betarodi*w(i,j)/r(j))
114               agau(5,5)=1.-foin*dvdy
115               agau(5,6)=foin*dvdy
116               agau(5,7)=Ø.
117
118
119  c-------------      vwc
120
121               agau(6,1)=Ø.
122               agau(6,2)=foin*(dwdy+betarodi*w(i,j)/r(j))
123               agau(6,3)=-foin*(1.+betarodi)*w(i,j)/r(j)
124               agau(6,4)=foin*dwdx
125               agau(6,5)=foin*dvdx
126               agau(6,6)=1.-foin*dudx
127               agau(6,7)=Ø.
128
129
130  c-------------     solution
131
132             deter=1.
133             do 9 k=1,ngauss
134             deter=deter*agau(k,k)
135             if( dabs(agau(k,k)) .gt. epsgauss ) go to 5
136             write(*,*)'matrix may be singular - props subroutine '
137             write(*,*)'node ',i,j
138             stop
139  5          kp1=k+1
140             do 6 jga=kp1,ngp1
141             agau(k,jga)=agau(k,jga)/agau(k,k)
142  6          continue
143             agau(k,k)=1
144             do 9 iga=1,ngauss
145             if ( (iga .eq. k ) .or. ( agau(iga,k) .eq. Ø) ) go to 9
146             do 8 jga=kp1,ngp1
147             agau(iga,jga)=agau(iga,jga)-agau(iga,k)*agau(k,jga)
148  8          continue
149             agau(iga,k)=Ø.
150  9          continue
151
152
153  c------------------- under-relax
154
155             uu(i,j)=urften*agau(1,ngp1)+tasort*uu(i,j)
156             vv(i,j)=urften*agau(2,ngp1)+tasort*vv(i,j)
157             ww(i,j)=urften*agau(3,ngp1)+tasort*ww(i,j)
158             uvc(i,j)=urften*agau(4,ngp1)+tasort*uvc(i,j)
159             uwc(i,j)=urften*agau(5,ngp1)+tasort*uwc(i,j)
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.00

```
160                vwc(i,j)=urften*agau(6,ngp1)+tasort*vwc(i,j)
161
162   100    continue
163
164                  call boundary(1)
165
166                  do 200 i=2,nim1
167                  do 200 j=2,jmax(i)
168
169                  if ( uu(i,j) .lt. 0. ) uu(i,j)=0.
170                  if ( vv(i,j) .lt. 0. ) vv(i,j)=0.
171                  if ( ww(i,j) .lt. 0. ) ww(i,j)=0.
172
173                  if  ( i .eq. 2 )  then
174                  uw(i,j)=0.
175                  else
176                  uw(i,j)=0.5*(uwc(i,j)+uw(i-1,j))
177                  end if
178
179                  if ( j .eq. 2 ) then
180                  vw(i,j)=0.
181                  else
182                  vw(i,j)=0.5*(vwc(i,j)+vwc(i,j-1))
183                  end if
184
185   200            continue
186
187   c-------------- uv node
188
189                do 300 i=3,nim1
190                do 300 j=3,jmax(i)
191
192                ip1=i+1
193                im1=i-1
194                jp1=j+1
195                jm1=j-1
196
197                dudx=(u(ip1,j)+u(ip1,jm1)-u(im1,j)-u(im1,jm1))/
198         1           (2.*(xu(ip1)-xu(im1)))
199                dudy=(u(i,j)-u(i,jm1))/(y(j)-y(jm1))
200                dvdx=(v(i,j)-v(im1,j))/(x(i)-x(im1))
201                dvdy=(v(i,jp1)+v(im1,jp1)-v(i,jm1)-v(im1,jm1))/
202         1           (2.*(rv(jp1)-rv(jm1)))
203                vtmpdr=(v(i,j)+v(im1,j))/(2.*rv(j))
204                wtmpdr=((w(i,j)+w(i-1,j))/r(j)+(w(i,j-1)+
205         1                w(i-1,j-1))/r(j-1))/4.
206                dwdx=(w(i,j)+w(i,j-1)-w(i-1,j)-w(i-1,j-1))
207         1           /(2.*(x(i)-x(i-1)))
208                dwdy=(w(i,j)+w(i-1,j)-w(i,j-1)-w(i-1,j-1))
209         1           /(2.*(r(j)-r(j-1)))
210                geno1=(genold(i,j)+genold(im1,j)+genold(im1,jm1)
211         1            +genold(i,jm1))/4.
212                edno1=(ed(i,j)+ed(im1,j)+ed(im1,jm1)+ed(i,jm1))/4.
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
213                teno1=(te(i,j)+te(im1,j)+te(im1,jm1)+te(i,jm1))/4.
214
215                alambt=umc2/(c1m1+geno1/edno1)
216                foin=alambt*teno1/edno1
217                foin1=2.*foin/3.
218
219   c------------        uuno1
220
221               agau(1,1)=1.+2.*foin1*dudx
222               agau(1,2)=-foin1*dvdy
223               agau(1,3)=-foin1*vtmpdr
224               agau(1,4)=-(dvdx-2.*dudy)*foin1
225               agau(1,5)=-foin1*dwdx
226               agau(1,6)=-foin1*(dwdy-wtmpdr)
227               agau(1,7)=2.*teno1/3.
228
229   c------------        vvno1
230
231               agau(2,1)=-foin1*dudx
232               agau(2,2)=1.+2*foin1*dvdy
233               agau(2,3)=-foin1*vtmpdr
234               agau(2,4)=-(dudy-2.*dvdx)*foin1
235               agau(2,5)=-foin1*dwdx
236               agau(2,6)=-foin1*(dwdy+wtmpdr*(2.+3.*betarodi))
237               agau(2,7)=2*teno1/3.
238
239   c-------------       wwno1
240
241                agau(3,1)=-foin1*dudx
242                agau(3,2)=-foin1*dvdy
243                agau(3,3)=1.+2.*foin1*vtmpdr
244                agau(3,4)=-(dvdx+dudy)*foin1
245                agau(3,5)=2.*foin1*dwdx
246                agau(3,6)=foin1*(2.*dwdy+(1.+3.*betarodi)*wtmpdr)
247                agau(3,7)=2*teno1/3.
248
249
250
251   c------------        uvno1
252
253               agau(4,1)=foin*dvdx
254               agau(4,2)=foin*dudy
255               agau(4,3)=Ø.
256               agau(4,4)=1.-foin*vtmpdr
257               agau(4,5)=-foin*wtmpdr*(1.+betarodi)
258               agau(4,6)=Ø.
259               agau(4,7)=Ø.
260
261   c------------        uwno1
262
263               agau(5,1)=foin*dwdx
264               agau(5,2)=Ø.
265               agau(5,3)=Ø.
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.00

```
266                agau(5,4)=foin*(dwdy+betarodi*wtmpdr)
267                agau(5,5)=1.-foin*dvdy
268                agau(5,6)=foin*dvdy
269                agau(5,7)=0.
270
271
272   c-------------     vwno1
273
274                agau(6,1)=0.
275                agau(6,2)=foin*(dwdy+betarodi*wtmpdr)
276                agau(6,3)=-foin*(1.+betarodi)*wtmpdr
277                agau(6,4)=foin*dwdx
278                agau(6,5)=foin*dvdx
279                agau(6,6)=1.-foin*dudx
280                agau(6,7)=0.
281
282
283   c-------------    solution
284
285              deter=1.
286              do 309 k=1,ngauss
287              deter=deter*agau(k,k)
288              if( dabs(agau(k,k)) .gt. epsgauss ) go to 305
289              write(*,*)'matrix may be singular - props subroutine '
290              write(*,*)'uv node ',i,j
291              stop
292   305        kp1=k+1
293              do 306 jga=kp1,ngp1
294              agau(k,jga)=agau(k,jga)/agau(k,k)
295   306        continue
296              agau(k,k)=1
297              do 309 iga=1,ngauss
298              if ( (iga .eq. k ) .or. ( agau(iga,k) .eq. 0) ) go to 309
299              do 308 jga=kp1,ngp1
300              agau(iga,jga)=agau(iga,jga)-agau(iga,k)*agau(k,jga)
301   308        continue
302              agau(iga,k)=0.
303   309        continue
304
305
306   c-------------------- under-relax
307
308              uv(i,j)=urften*agau(4,ngp1)+tasort*uv(i,j)
309
310   c--------------------
311
312   300    continue
313
314
315          return
316
317          end
```

Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

PROPS Local Symbols

Name	Class	Type	Size	Offset
IM1	local	INTEGER*4	4	ØØØ2
JM1	local	INTEGER*4	4	ØØØ6
I	local	INTEGER*4	4	ØØØa
IP1	local	INTEGER*4	4	ØØØe
J	local	INTEGER*4	4	ØØ12
JP1	local	INTEGER*4	4	ØØ16
K	local	INTEGER*4	4	ØØ1a
KP1	local	INTEGER*4	4	ØØ1e
IGA	local	INTEGER*4	4	ØØ22
NGAUSS	local	INTEGER*4	4	ØØ26
JGA	local	INTEGER*4	4	ØØ2a
NGP1	local	INTEGER*4	4	ØØ2e
EDNO1	local	REAL*4	4	ØØ32
GENO1	local	REAL*4	4	ØØ36
FOIN1	local	REAL*4	4	ØØ3a
VTMPDR	local	REAL*4	4	ØØ3e
TASORT	local	REAL*4	4	ØØ42
WTMPDR	local	REAL*4	4	ØØ46
AGAU	local	REAL*4	196	ØØ4a
TENO1	local	REAL*4	4	Ø10e
EPSGAUSS	local	REAL*4	4	Ø112
FOIN	local	REAL*4	4	Ø116
ALAMBT	local	REAL*4	4	Ø11a
DETER	local	REAL*4	4	Ø11e
DUDX	local	REAL*4	4	Ø122
DVDX	local	REAL*4	4	Ø126
DUDY	local	REAL*4	4	Ø12a
DWDX	local	REAL*4	4	Ø12e
DVDY	local	REAL*4	4	Ø132
DWDY	local	REAL*4	4	Ø136
CAPPA	TURBU	REAL*4	4	1214
ELOG	TURBU	REAL*4	4	1218
PRED	TURBU	REAL*4	4	121c
PRTE	TURBU	REAL*4	4	1220
C1	TURBU	REAL*4	4	1224
C2	TURBU	REAL*4	4	1228
UMC2	TURBU	REAL*4	4	122c
C1M1	TURBU	REAL*4	4	1230
CK	TURBU	REAL*4	4	1234
CE3	TURBU	REAL*4	4	1238
UU	STRESS	REAL*4	4608	ØØØØ
VV	STRESS	REAL*4	4608	1200
WW	STRESS	REAL*4	4608	2400
UV	STRESS	REAL*4	4608	3600
UW	STRESS	REAL*4	4608	4800
U	VAR	REAL*4	4608	ØØØØ
VW	STRESS	REAL*4	4608	5a00
V	VAR	REAL*4	4608	1200
W	VAR	REAL*4	4608	2400

PROPS Local Symbols

Name	Class	Type	Size	Offset
GENOLD.	OLD	REAL*4	4608	0000
P	VAR	REAL*4	4608	3600
PP.	VAR	REAL*4	4608	4800
UVC	LOCALST	REAL*4	4608	0000
TE.	VAR	REAL*4	4608	5a00
UWC	LOCALST	REAL*4	4608	1200
ED.	VAR	REAL*4	4608	6c00
VWC	LOCALST	REAL*4	4608	2400
IT.	GERAL	INTEGER*4	4	0000
JT.	GERAL	INTEGER*4	4	0004
NI.	GERAL	INTEGER*4	4	0008
NJ.	GERAL	INTEGER*4	4	000c
NIM1.	GERAL	INTEGER*4	4	0010
NJM1.	GERAL	INTEGER*4	4	0014
GREAT	GERAL	REAL*4	4	0018
JMAX.	GERAL	INTEGER*4	192	001c
JMAXP1.	GERAL	INTEGER*4	192	00dc
X	GEOM	REAL*4	192	0000
Y	GEOM	REAL*4	96	00c0
DXEP.	GEOM	REAL*4	192	0120
DXPW.	GEOM	REAL*4	192	01e0
DYNP.	GEOM	REAL*4	96	02a0
DYPS.	GEOM	REAL*4	96	0300
SNS	GEOM	REAL*4	96	0360
SEW	GEOM	REAL*4	192	03c0
XU.	GEOM	REAL*4	192	0480
YV.	GEOM	REAL*4	96	0540
R	GEOM	REAL*4	96	05a0
RV.	GEOM	REAL*4	96	0600
WFN	GEOM	REAL*4	96	0660
WFS	GEOM	REAL*4	96	06c0
WFE	GEOM	REAL*4	192	0720
WFW	GEOM	REAL*4	192	07e0
RCV	GEOM	REAL*4	96	08a0
IFINE	GEOM	INTEGER*4	4	0900
URFTEN.	FLUPR	REAL*4	4	0000
VISCOS.	FLUPR	REAL*4	4	0004
DENSIT.	FLUPR	REAL*4	4	0008
PRANDT.	FLUPR	REAL*4	4	000c
DEN	FLUPR	REAL*4	4608	0010
GEN	TURBU	REAL*4	4608	0000
CD.	TURBU	REAL*4	4	1200
CMU	TURBU	REAL*4	4	1204
C1E	TURBU	REAL*4	4	1208
C2E	TURBU	REAL*4	4	120c
BETARODI.	TURBU	REAL*4	4	1210

Microsoft FORTRAN Optimizing Compiler Version 5.Ø0

Global Symbols

Name	Class	Type	Size	Offset
BOUNDARY.	extern	***	***	***
FLUPR	common	***	4624	ØØØØ
GEOM.	common	***	23Ø8	ØØØØ
GERAL	common	***	412	ØØØØ
LOCALSTRESS	common	***	13824	ØØØØ
OLD	common	***	46Ø8	ØØØØ
PROPS	FSUBRT	***	***	ØØØØ
STRESS.	common	***	27648	ØØØØ
TURBU	common	***	4668	ØØØØ
VAR	common	***	32256	ØØØØ

Code size = 1a1e (6686)
Data size = ØØa9 (169)
Bss size = Ø13a (314)

No errors detected

```
Line#  Source Line      Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

 1
 2          subroutine boundary (nchap)
 3
 4   cØ*******************      preliminaires
 5
 6          logical iomega,ifine
 7
 8          common
 9         1/uvel/resolu,nswpu,urfu,dxepu(48),dxpwu(48),sewu(48)
10         2/vvel/resolv,nswpv,urfv,dynpv(24),dypsv(24),snsv(24)
11         3/wvel/resolw,nswpw,urfw,iomega
12         4/corrp/resolm,nswpp,urfp,du(48,24),dv(48,24),ipref,jpref
13         5/var/u(48,24), v(48,24), w(48,24), p(48,24), pp(48,24), te(48,24)
14         6      ed(48,24)
15         7/geral/it,jt,ni,nj,nim1,njm1,great,jmax(48),jmaxp1(48)
16         8/geom/x(48),y(24),dxep(48),dxpw(48),dynp(24),dyps(24),
17         9       sns(24),sew(48),xu(48),yv(24),r(24),rv(24),
18         1       wfn(24),wfs(24),wfe(48),wfw(48),rcv(24),ifine
19
20           common
21         1/flupr/urften,viscos,densit,prandt,den(48,24)
22         2/causo/uin,tein,edin,flowin,alamda,
23         3        rsmall,rlarge,xtota,jstep,istep,jstp1,jstm1,istp1,istm1,
24         4        jexit,jexitp1,inleti,inletf,jinlet,rgarg,ngarg,ngarm1,
25         5        ngarp1
26         6/susptmp/sukd(48,24),spkd(48,24)
27         7/turbu/gen(48,24),cd,cmu,c1e,c2e,betarodi,cappa,elog,
28         8       pred,prte,c1,c2,umc2,c1m1,ck,ce3
29         9/wallf/yplusn(48),xplusw(24),xpluse(24),taun(48),tauw(24),
30         1        taue(24)
31         2/coef/ap(48,24),an(48,24),as(48,24),ae(48,24),aw(48,24),su(48,24)
32         3       sp(48,24)
33
34          common
35         1/stress/uu(48,24),vv(48,24),ww(48,24),uv(48,24),
36         2         uw(48,24),vw(48,24)
37         3/boundst/genuv(48,24),genuv1(48,24),genuv2(48,24),genuv3(48,24)
38         4/localstress/uvc(48,24),uwc(48,24),vwc(48,24)
39
40
41   c---------------------    selection
42
43          if (nchap .eq. 1) go to 1
44
45          if (nchap .eq. 2) go to 2
46
47          if (jstep .eq. njm1) go to 11ØØ
48
49   c---------------------    out of range values
50
51                do 1ØØØ i=2,ngarm1
52
53                if(jmax(i) .eq. njm1)  go to 11ØØ
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
54
55                jini=jmaxp1(i)
56                do 1ØØØ j=jini,njm1
57                sp(i,j)=-great
58                su(i,j)=Ø.
59  1ØØØ          continue
60
61  11ØØ          do 12ØØ i=ngarg,ni
62                 do 12ØØ j=jexitp1,njm1
63                 sp(i,j)=-great
64                 su(i,j)=Ø.
65  12ØØ          continue
66
67  13ØØ           go to ( 1,2,3,4,5,6,7 ) , nchap
68
69
70  c1******************      properties
71
72  1      continue
73
74         do 1ØØ j=2,jexit
75
76              uu(ni,j)=uu(nim1,j)
77              vv(ni,j)=vv(nim1,j)
78              ww(ni,j)=ww(nim1,j)
79              uvc(ni,j)=uvc(nim1,j)
8Ø              uwc(ni,j)=uwc(nim1,j)
81              vwc(ni,j)=vwc(nim1,j)
82
83  1ØØ    continue
84
85         return
86
87
88  c2******************      u momentum
89
9Ø  2      continue
91
92  c---------------------    out of range values
93
94                if(jstep .eq. njm1) then
95
96  c---------------------    constant section
97
98                 continue
99
1ØØ               else
101
102                 do 2Ø2 i=3,ngarm1
1Ø3                        if(jmax(i-1) .eq. njm1) then
1Ø4
105 c---------------------    end of enlargement
1Ø6
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
107                           go to 2Ø4
108                           else
109                           jini=jmaxp1(i-1)
110                           do 2Ø3 j=jini,njm1
111                           sp(i,j)=-great
112                           su(i,j)=Ø.
113  2Ø3                      continue
114                           end if
115
116  2Ø2                 continue
117
118                   end if
119
120  c---------------------    east wall
121
122  2Ø4                      do 2Ø5 j=jexitp1,njm1
123                           sp(ngarg,j)=-great
124                           su(ngarg,j)=Ø.
125  2Ø5                      continue
126
127  c---------------------    extension
128
129                       do 2Ø6 i=ngarp1,ni
130                       jini=jmaxp1(i-1)
131                       do 2Ø6 j=jini,njm1
132                       sp(i,j)=-great
133                       su(i,j)=Ø.
134  2Ø6                  continue
135
136  c---------------------    north side wall
137
138              bim=cmu**Ø.25
139
140              do 2Ø8 i=3,inleti
141              j=jmax(i-1)
142              an(i,j)=Ø.Ø
143              yp=yv(j+1)-y(j)
144
145              romed=Ø.5*(den(i,j)+den(i-1,j))
146              sqrtk=sqrt(Ø.5*(te(i,j)+te(i-1,j)))
147              yplusm=Ø.5*(yplusn(i)+yplusn(i-1))
148
149              if(yplusm .le. 11.5Ø) then
150              turact=viscos/yp
151              else
152              turact=romed*bim*sqrtk*cappa/alog(elog*yplusm)
153              end if
154
155              sp(i,j)=sp(i,j)-turact*sewu(i)*rv(j+1)
156
157                 if( (jmax(i-1) .ne. jmax(i)) .or. i .eq. inleti ) then
158
159  c---------------------    half control volume
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.00

```
160
161                  sp(i,j)=sp(i,j)/2.
162                  else
163                  continue
164                  end if
165
166  208          continue
167
168               do 210 i=inletf+1,ngarm1
169               j=jmax(i-1)
170               an(i,j)=0.0
171               yp=yv(j+1)-y(j)
172
173               romed=0.5*(den(i,j)+den(i-1,j))
174               sqrtk=sqrt(0.5*(te(i,j)+te(i-1,j)))
175               yplusm=0.5*(yplusn(i)+yplusn(i-1))
176
177               if(yplusm .le. 11.50) then
178               turact=viscos/yp
179               else
180               turact=romed*bim*sqrtk*cappa/alog(elog*yplusm)
181               end if
182
183               sp(i,j)=sp(i,j)-turact*sewu(i)*rv(j+1)
184
185                  if((jmax(i-1) .ne. jmax(i)) .or. i .eq. inletf+1 ) then
186
187  c----------------------    half control volume
188
189                  sp(i,j)=sp(i,j)/2.
190                  else
191                  continue
192                  end if
193
194  210          continue
195
196  c----------------------    extension
197
198               do 215 i=ngarg,nim1
199
200               j=jmax(i)
201               an(i,j)=0.0
202
203  215          continue
204
205  c----------------------    symmetry axis
206
207            do 220 i=1,ni
208            as(i,2)=0.0
209  220       continue
210
211  c----------------------    west side wall
212
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
213  c--------     no modif
214
215  c---------------------    outlet
216
217  235       ardens=Ø.Ø
218            flowout=Ø.Ø
219
220              do 24Ø j=2,jexit
221              arden=Ø.5*(den(nim1,j)+den(nim1-1,j))*rcv(j)*sns(j)
222              ardens=ardens+arden
223              flowout=flowout+arden*u(nim1,j)
224  24Ø         continue
225
226              bumco=(flowin-flowout)/ardens
227
228             do 25Ø j=2,jexit
229
23Ø             u(ni,j)=u(nim1,j)+bumco
231
232  25Ø       continue
233
234  c---------------------    east side wall
235
236  c--------     no modif
237
238           return
239
240  c3******************      v momentum
241
242  3         continue
243
244  c---------------------    west side wall
245
246            bim=cmu**Ø.25
247
248            if(jstep .eq. njm1) then
249              continue
25Ø            else
251              do 32Ø i=2,ngarm1
252              if (jmax(i-1) .ge. jmax(i)) go to 33Ø
253              jini=jmaxp1(i-1)
254              jfim=jmax(i)
255               do 31Ø j=jini,jfim
256               aw(i,j)=Ø.Ø
257               xp=x(i)-xu(i)
258
259               romed=Ø.5*(den(i,j)+den(i,j-1))
26Ø               sqrtk=sqrt(Ø.5*(te(i,j)+te(i,j-1)))
261               xplusm=Ø.5*(xplusw(j)+xplusw(j-1))
262                   if (xplusm .le. 11.5Ø) then
263                   turact=viscos/xp
264                   else
265                   turact=romed*bim*sqrtk*cappa/alog(elog*xplusm)
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
266                     end if
267
268                  sp(i,j)=sp(i,j)-turact*snsv(j)*rv(j)
269                  if( j .eq. jmaxp1(i-1)) sp(i,j)=sp(i,j)/2.
270
271  31Ø             continue
272
273  32Ø             continue
274
275               end if
276
277
278  c----------------------    east side wall
279
280  33Ø         continue
281
282            if (jstep .eq. njm1 ) then
283                     continue
284            else
285                     do 35Ø j=jexitp1,njm1
286                     ae(ngarm1,j)=Ø.Ø
287                     xp=xu(ngarg)-x(ngarm1)
288
289              romed=Ø.5*(den(ngarm1,j)+den(ngarm1,j-1))
29Ø              sqrtk=sqrt(Ø.5*(te(ngarm1,j)+te(ngarm1,j-1)))
291                  if ( j .eq. jexitp1) then
292                  xplusm=xpluse(j)
293                  else
294                  xplusm=Ø.5*(xpluse(j)+xpluse(j-1))
295                  end if
296
297                   if (xplusm .le. 11.5Ø) then
298                   turact=viscos/xp
299                   else
3ØØ                  turact=romed*bim*sqrtk*cappa/alog(elog*xplusm)
3Ø1                  end if
3Ø2             sp(ngarm1,j)=sp(ngarm1,j)-turact*snsv(j)*rv(j)
3Ø3
3Ø4             if ( j .eq. jexitp1) sp(ngarm1,j)=sp(ngarm1,j)/2.
3Ø5  35Ø        continue
3Ø6
3Ø7            end if
3Ø8
3Ø9  c----------------------    north wall
31Ø
311  c---------     no modif
312
313          return
314
315  c4*****************      pressure correction
316
317  4       continue
318
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.00

```
319
320  c---------------------    west side wall
321
322                if(jstep .eq. njm1) then
323                   continue
324                else
325                   do 420 i=2,ngarm1
326                   if(jmax(i-1) .ge. jmax(i)) go to 425
327                   jini=jmaxp1(i-1)
328                   jfim=jmax(i)
329                       do 410 j=jini,jfim
330                       aw(i,j)=0.
331  410                  continue
332  420               continue
333                end if
334
335  c---------------------    east side wall
336
337  425                do 430 j=jexitp1,njm1
338                     ae(ngarm1,j)=0.0
339  430                continue
340
341
342  c---------------------    north wall
343
344             do 440 i=2,inleti-1
345             an(i,jmax(i))=0.0
346  440        continue
347
348             do 445 i=inletf+1,nim1
349             an(i,jmax(i))=0.0
350  445        continue
351
352  c---------------------    symmetry axis
353
354             do 450 i=2,nim1
355             as(i,2)=0.0
356  450        continue
357
358  c---------------------    outlet
359
360             do 460 j=2,jexit
361             ae(nim1,j)=0.
362  460        continue
363
364
365        return
366
367
368  c5******************      SWIRL VELOCITY
369
370  5     continue
371
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
372  c---------------------    north side wall
373
374        bim=cmu**Ø.25
375
376        do 51Ø i=2,inleti-1
377        j=jmax(i)
378        an(i,j)=Ø.Ø
379        yp=yv(j+1)-y(j)
38Ø        sqrtk=sqrt(te(i,j))
381        yplusm=yplusn(i)
382
383            if(yplusm .le. 11.5Ø) then
384              turact=viscos/yp
385            else
386              turact=den(i,j)*bim*sqrtk*cappa/alog(elog*yplusm)
387
388            end if
389
39Ø             sp(i,j)=sp(i,j)-turact*sew(i)*rv(j+1)
391
392  51Ø   continue
393
394        do 515 i=inletf+1,nim1
395        j=jmax(i)
396        an(i,j)=Ø.Ø
397        yp=yv(j+1)-y(j)
398        sqrtk=sqrt(te(i,j))
399        yplusm=yplusn(i)
4ØØ
4Ø1            if(yplusm .le. 11.5Ø) then
4Ø2              turact=viscos/yp
4Ø3            else
4Ø4              turact=den(i,j)*bim*sqrtk*cappa/alog(elog*yplusm)
4Ø5
4Ø6            end if
4Ø7
4Ø8             sp(i,j)=sp(i,j)-turact*sew(i)*rv(j+1)
4Ø9  515   continue
41Ø
411  c---------------------    extension
412
413        do 517 i=ngarg,nim1
414        j=jmax(i)
415        an(i,j)=Ø.Ø
416  517   continue
417
418  c---------------------    west side wall
419
42Ø           if(jstep .eq. njm1) then
421              continue
422           else
423              do 53Ø i=2,ngarm1
424
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
425                if(jmax(i-1) .ge. jmax(i)) go to 535
426
427                jini=jmaxp1(i-1)
428                jfim=jmax(i)
429                  do 52Ø j=jini,jfim
430                  aw(i,j)=Ø.Ø
431                  xp=x(i)-xu(i)
432                  sqrtk=sqrt(te(i,j))
433                  xplusm=xplusw(j)
434
435                     if(xplusm .le. 11.5Ø) then
436                     turact=viscos/xp
437                     else
438                     turact=den(i,j)*bim*sqrtk*cappa/alog(elog*xplusm)
439                     end if
440
441                sp(i,j)=sp(i,j)-turact*sns(j)*rcv(j)
442
443
444  52Ø       continue
445  53Ø       continue
446
447             end if
448
449  c----------------------    east side wall
450
451  535          if (jexit .eq. njm1) then
452                   continue
453               else
454                   do 54Ø j=jexitp1,jmax(ngarm1)
455                   ae(ngarm1,j)=Ø.Ø
456                   xp=xu(ngarg)-x(ngarm1)
457                   sqrtk=sqrt(te(ngarm1,j))
458                   xplusm=xpluse(j)
459
460                    if ( xplusm .le. 11.5Ø ) then
461                      turact=viscos/xp
462                    else
463                      turact=den(ngarm1,j)*bim*sqrtk*cappa/alog(elog*xplusm
464                    end if
465
466                   sp(ngarm1,j)=sp(ngarm1,j)-turact*sns(j)*rcv(j)
467  54Ø              continue
468
469               end if
470
471  c----------------------    symmetry axis
472
473  c----------------------    fix w for solid rotation
474
475             do 55Ø i=2,nim1
476             gum=w(i,3)*r(2)/r(3)
477              su(i,2)=great*gum
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
478              sp(i,2)=-great
479   55Ø        continue
48Ø
481   c---------------------    outlet
482
483                do 56Ø j=2,jexit
484                ae(nim1,j)=Ø.Ø
485   56Ø          continue
486
487         return
488
489   c6******************      turbulent kinetic energy
49Ø
491   6     continue
492
493         bim=cmu**Ø.25
494         tim=cmu**Ø.75
495
496   c---------------------    west side wall
497
498           if(jstep .eq. njm1) then
499              continue
5ØØ           else
5Ø1              do 62Ø i=2,ngarm1
5Ø2              if(jmax(i-1) .ge. jmax(i)) go to 625
5Ø3              jini=jmaxp1(i-1)
5Ø4              jfim=jmax(i)
5Ø5               do 61Ø j=jini,jfim
5Ø6               aw(i,j)=Ø.Ø
5Ø7               dwdx=(w(i+1,j)-w(i-1,j))/(dxpw(i)+dxep(i))
5Ø8               dvdx=(v(i+1,j)+v(i+1,j+1)-v(i-1,j)-v(i-1,j+1))/(4*sew(i))
5Ø9               if (j .eq. 2 ) then
51Ø               dudy=Ø.
511               else
512               dudy=(u(i,j+1)+u(i+1,j+1)-u(i,j-1)-u(i+1,j-1))/(4*sns(j))
513               end if
514               vavg=v(i,j)*wfn(j)+(1.-wfn(j))*v(i,j+1)
515               veff=sqrt(vavg*vavg+w(i,j)*w(i,j))
516
517               sqrtk=sqrt(te(i,j))
518               xp=x(i)-xu(i)
519               vol=rcv(j)*sns(j)*sew(i)
52Ø               xplusw(j)=den(i,j)*sqrtk*bim*xp/viscos
521
522               if(xplusw(j) .le. 11.5Ø) then
523
524               tauxr=-viscos*vavg/xp
525               tauxw=-viscos*w(i,j)/xp
526               tauw(j)=sqrt(tauxr*tauxr+tauxw*tauxw)
527               diterm=tim*te(i,j)**Ø.5*xplusw(j)/xp
528
529               else
53Ø
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.00

```
                turact=den(i,j)*bim*sqrtk*cappa/alog(elog*xplusw(j))
                tauxr=-turact*vavg
                tauxw=-turact*w(i,j)
                tauw(j)=turact*veff
                diterm=tim*te(i,j)**0.5*alog(elog*xplusw(j))/
     1                 (cappa*xp)

              end if

             gen1=(abs(tauxr*(dudy+dvdx))+abs(tauxw*dwdx))/den(i,j)
             gen2=gen(i,j)+(genuv1(i,j))+(genuv3(i,j))
             gen(i,j)=gen2+gen1
             su(i,j)=den(i,j)*gen(i,j)*vol+sukd(i,j)
             sp(i,j)=spkd(i,j)-den(i,j)*diterm*vol

610       continue
620       continue
625                     tauw(nj)=tauw(njm1)
          end if

c---------------------     east side wall

           if (jexit .eq. njm1) then

           continue

           else

              i=ngarm1
              do 630 j=jexitp1,njm1
              ae(i,j)=0.0
              dwdx=(w(i+1,j)-w(i-1,j))/(dxpw(i)+dxep(i))
              dvdx=(v(i+1,j)+v(i+1,j+1)-v(i-1,j)-v(i-1,j+1))/(4*sew(j))
              dudy=(u(i,j+1)+u(i+1,j+1)-u(i,j-1)-u(i+1,j-1))/(4*sns(j))
              vavg=v(i,j)*wfn(j)+(1.-wfn(j))*v(i,j+1)
              veff=sqrt(vavg*vavg+w(i,j)*w(i,j))

              sqrtk=sqrt(te(i,j))
              xp=xu(ngarg)-x(i)
              vol=rcv(j)*sns(j)*sew(i)
              xpluse(j)=den(i,j)*sqrtk*bim*xp/viscos

              if(xpluse(j) .le. 11.50) then

              tauxr=viscos*vavg/xp
              tauxw=viscos*w(i,j)/xp
              taue(j)=sqrt(tauxr*tauxr+tauxw*tauxw)
              diterm=tim*te(i,j)**0.5*xpluse(j)/xp

              else

               turact=den(i,j)*bim*cappa/alog(elog*xpluse(j))
               tauxr=turact*vavg
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
584                 tauxw=turact*w(i,j)
585                 taue(j)=turact*veff
586                 diterm=tim*te(i,j)**Ø.5*alog(elog*xpluse(j))/
587       1                   (cappa*xp)
588
589                end if
590
591               gen1=(abs(tauxr*(dudy+dvdx))+abs(tauxw*dwdx))/den(i,j)
592               gen2=gen(i,j)+(genuv1(i,j))+(genuv3(i,j))
593               gen(i,j)=gen2+gen1
594               su(i,j)=den(i,j)*gen(i,j)*vol+sukd(i,j)
595               sp(i,j)=spkd(i,j)-den(i,j)*diterm*vol
596
597  63Ø       continue
598                     taue(nj)=taue(njm1)
599           end if
6ØØ
6Ø1  c---------------------    outlet
6Ø2
6Ø3                 do 64Ø j=2,jexit
6Ø4                 ae(nim1,j)=Ø.Ø
6Ø5  64Ø            continue
6Ø6
6Ø7  c---------------------    symmetry axis
6Ø8
6Ø9
61Ø       do 65Ø i=2,nim1
611
612       as(i,2)=Ø.Ø
613
614  65Ø  continue
615
616
617  c---------------------    north side wall
618
619       do 66Ø i=2,inleti-1
62Ø       j=jmax(i)
621       an(i,j)=Ø.Ø
622
623       dwdy=(w(i,j+1)-w(i,j-1))/(dynp(j)+dyps(j))
624       dudy=(u(i,j+1)+u(i+1,j+1)-u(i,j-1)-u(i+1,j-1))/(4*sns(j))
625       dvdx=(v(i+1,j)+v(i+1,j+1)-v(i-1,j)-v(i-1,j+1))/(4*sew(i))
626       uavg=u(i,j)*wfe(i)+(1.Ø-wfe(i))*u(i+1,j)
627       ueff=sqrt(uavg*uavg+w(i,j)*w(i,j))
628
629       yp=yv(j+1)-y(j)
63Ø       sqrtk=sqrt(te(i,j))
631       vol=rcv(j)*sns(j)*sew(i)
632       yplusn(i)=den(i,j)*sqrtk*bim*yp/viscos
633
634           if(yplusn(i) .le. 11.5Ø) then
635
636           taurx=viscos*uavg/yp
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.00

```
637             taurw=viscos*(w(i,j)/yp+w(i,j)/y(j))
638             taun(i)=sqrt(taurx*taurx+taurw*taurw)
639             diterm=tim*te(i,j)**0.5*yplusn(i)/yp
640
641             else
642
643             turact=den(i,j)*bim*sqrtk*cappa/alog(elog*yplusn(i))
644             taurx=turact*uavg
645             taurw=turact*w(i,j)
646             taun(i)=turact*ueff
647             diterm=tim*te(i,j)**0.5*alog(elog*yplusn(i))/(cappa*yp)
648
649             end if
650
651          gen1=(abs(taurx*(dudy+dvdx))+abs(taurw*(dwdy-w(i,j)/y(j))))
652         1       /den(i,j)
653
654          gen2=gen(i,j)+(genuv1(i,j))+(genuv2(i,j))
655
656          gen(i,j)=gen2+gen1
657          su(i,j)=den(i,j)*gen(i,j)*vol+sukd(i,j)
658          sp(i,j)=spkd(i,j)-den(i,j)*diterm*vol
659
660   660       continue
661
662
663          do 670 i=inletf+1,nim1
664          j=jmax(i)
665          an(i,j)=0.0
666
667          dwdy=(w(i,j+1)-w(i,j-1))/(dynp(j)+dyps(j))
668          dudy=(u(i,j+1)+u(i+1,j+1)-u(i,j-1)-u(i+1,j-1))/(4*sns(j))
669          dvdx=(v(i+1,j)+v(i+1,j+1)-v(i-1,j)-v(i-1,j+1))/(4*sew(i))
670          uavg=u(i,j)*wfe(i)+(1.0-wfe(i))*u(i+1,j)
671          ueff=sqrt(uavg*uavg+w(i,j)*w(i,j))
672
673          yp=yv(j+1)-y(j)
674          sqrtk=sqrt(te(i,j))
675          vol=rcv(j)*sns(j)*sew(i)
676          yplusn(i)=den(i,j)*sqrtk*bim*yp/viscos
677
678              if(yplusn(i) .le. 11.50) then
679
680              taurx=viscos*uavg/yp
681              taurw=viscos*(w(i,j)/yp+w(i,j)/y(j))
682              taun(i)=sqrt(taurx*taurx+taurw*taurw)
683              diterm=tim*te(i,j)**0.5*yplusn(i)/yp
684
685              else
686              turact=den(i,j)*bim*sqrtk*cappa/alog(elog*yplusn(i))
687              taurx=turact*uavg
688              taurw=turact*w(i,j)
689              taun(i)=turact*ueff
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
690             diterm=tim*te(i,j)**Ø.5*alog(elog*yplusn(i))/(cappa*yp)
691             end if
692
693          gen1=(abs(taurx*(dudy+dvdx))+abs(taurw*(dwdy-w(i,j)/y(j))))
694         1      /den(i,j)
695          gen2=gen(i,j)+(genuv1(i,j))+(genuv2(i,j))
696
697          gen(i,j)=gen2+gen1
698          su(i,j)=den(i,j)*gen(i,j)*vol+sukd(i,j)
699          sp(i,j)=spkd(i,j)-den(i,j)*diterm*vol
700
701  67Ø       continue
702
703              taun(ni)=taun(nim1)
704
705
706          return
707
708  c7*****************       dissipation
709
710  7       continue
711
712  c---------------------     west side wall
713
714            tim=cmu**Ø.75
715
716             if(jstep .eq. njm1) then
717               continue
718             else
719               do 72Ø i=2,ngarm1
720
721               if(jmax(i-1) .ge. jmax(i)) go to 725
722
723                  jini=jmaxp1(i-1)
724                  jfim=jmax(i)
725
726                   do 71Ø j=jini,jfim
727                   xp=x(i)-xu(i)
728                   su(i,j)=great*tim/(cappa*xp)*te(i,j)**1.5
729                   sp(i,j)=-great
730  71Ø            continue
731
732  72Ø            continue
733
734             end if
735
736  c---------------------     east side wall
737
738  725              if ( jexit .eq. njm1 ) then
739                  continue
740              else
741
742                  do 73Ø j=jexitp1,njm1
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.00

```
                xp=xu(ngarg)-x(ngarm1)
                su(ngarm1,j)=great*tim/(cappa*xp)*te(ngarm1,j)**1.5
                sp(ngarm1,j)=-great
730             continue

              end if

c---------------------    outlet

              do 740 j=2,jexit
              ae(nim1,j)=0.0
740           continue

c---------------------    north side wall

              do 750 i=2,inleti-1
              j=jmax(i)
              yp=yv(j+1)-y(j)
              su(i,j)=great*tim/(cappa*yp)*te(i,j)**1.5
              sp(i,j)=-great
750           continue

              do 755 i=inletf+1,nim1
              j=jmax(i)
              yp=yv(j+1)-y(j)
              su(i,j)=great*tim/(cappa*yp)*te(i,j)**1.5
              sp(i,j)=-great
755           continue



c---------------------    symmetry axis

                do 760 i=2,nim1
                as(i,2)=0.0
760             continue

        return

            end
```

BOUNDARY Local Symbols

Name	Class	Type	Size	Offset
NCHAP	param			0006
DITERM	local	REAL*4	4	0000
I	local	INTEGER*4	4	0004
J	local	INTEGER*4	4	0008
GEN1	local	REAL*4	4	000c
GEN2	local	REAL*4	4	0010

BOUNDARY Local Symbols

Name	Class	Type	Size	Offset
TAURW	local	REAL*4	4	Ø014
TURACT	local	REAL*4	4	Ø018
TAURX	local	REAL*4	4	Ø01c
TAUXR	local	REAL*4	4	Ø02Ø
SQRTK	local	REAL*4	4	Ø024
BIM	local	REAL*4	4	Ø028
TAUXW	local	REAL*4	4	Ø02c
UEFF	local	REAL*4	4	Ø03Ø
JFIM	local	INTEGER*4	4	Ø034
VEFF	local	REAL*4	4	Ø038
XP	local	REAL*4	4	Ø03c
GUM	local	REAL*4	4	Ø04Ø
XPLUSM	local	REAL*4	4	Ø044
YP	local	REAL*4	4	Ø048
TIM	local	REAL*4	4	Ø04c
ARDEN	local	REAL*4	4	Ø05Ø
YPLUSM	local	REAL*4	4	Ø054
JINI	local	INTEGER*4	4	Ø058
FLOWOUT	local	REAL*4	4	Ø05c
VOL	local	REAL*4	4	Ø06Ø
UAVG	local	REAL*4	4	Ø064
VAVG	local	REAL*4	4	Ø068
DUDY	local	REAL*4	4	Ø06c
DVDX	local	REAL*4	4	Ø07Ø
BUMCO	local	REAL*4	4	Ø074
DWDX	local	REAL*4	4	Ø078
ROMED	local	REAL*4	4	Ø07c
DWDY	local	REAL*4	4	Ø08Ø
ARDENS	local	REAL*4	4	Ø084
CE3	TURBU	REAL*4	4	1238
DXEP	GEOM	REAL*4	192	Ø12Ø
DXPW	GEOM	REAL*4	192	Ø1eØ
DYNP	GEOM	REAL*4	96	Ø2aØ
YPLUSN	WALLF	REAL*4	192	ØØØØ
XPLUSW	WALLF	REAL*4	96	ØØcØ
DYPS	GEOM	REAL*4	96	Ø3ØØ
SNS	GEOM	REAL*4	96	Ø36Ø
XPLUSE	WALLF	REAL*4	96	Ø12Ø
TAUN	WALLF	REAL*4	192	Ø18Ø
SEW	GEOM	REAL*4	192	Ø3cØ
XU	GEOM	REAL*4	192	Ø48Ø
TAUW	WALLF	REAL*4	96	Ø24Ø
YV	GEOM	REAL*4	96	Ø54Ø
TAUE	WALLF	REAL*4	96	Ø2aØ
R	GEOM	REAL*4	96	Ø5aØ
RV	GEOM	REAL*4	96	Ø6ØØ
AP	COEF	REAL*4	46Ø8	ØØØØ
WFN	GEOM	REAL*4	96	Ø66Ø

BOUNDARY Local Symbols

Name	Class	Type	Size	Offset
AN.	COEF	REAL*4	46Ø8	12ØØ
WFS	GEOM	REAL*4	96	Ø6cØ
AS.	COEF	REAL*4	46Ø8	24ØØ
WFE	GEOM	REAL*4	192	Ø72Ø
AE.	COEF	REAL*4	46Ø8	36ØØ
WFW	GEOM	REAL*4	192	Ø7eØ
AW.	COEF	REAL*4	46Ø8	48ØØ
RCV	GEOM	REAL*4	96	Ø8aØ
SU.	COEF	REAL*4	46Ø8	5aØØ
IOMEGA.	WVEL	LOGICAL*4	4	ØØØc
SP.	COEF	REAL*4	46Ø8	6cØØ
IFINE	GEOM	LOGICAL*4	4	Ø9ØØ
URFTEN.	FLUPR	REAL*4	4	ØØØØ
UU.	STRESS	REAL*4	46Ø8	ØØØØ
VISCOS.	FLUPR	REAL*4	4	ØØØ4
VV.	STRESS	REAL*4	46Ø8	12ØØ
RESOLU.	UVEL	REAL*4	4	ØØØØ
DENSIT.	FLUPR	REAL*4	4	ØØØ8
WW.	STRESS	REAL*4	46Ø8	24ØØ
NSWPU	UVEL	INTEGER*4	4	ØØØ4
PRANDT.	FLUPR	REAL*4	4	ØØØc
UV.	STRESS	REAL*4	46Ø8	36ØØ
DEN	FLUPR	REAL*4	46Ø8	ØØ1Ø
URFU.	UVEL	REAL*4	4	ØØØ8
UW.	STRESS	REAL*4	46Ø8	48ØØ
DXEPU	UVEL	REAL*4	192	ØØØc
VW.	STRESS	REAL*4	46Ø8	5aØØ
UIN	CAUSO	REAL*4	4	ØØØØ
DXPWU	UVEL	REAL*4	192	ØØcc
TEIN.	CAUSO	REAL*4	4	ØØØ4
SEWU.	UVEL	REAL*4	192	Ø18c
EDIN.	CAUSO	REAL*4	4	ØØØ8
GENUV	BOUNDST	REAL*4	46Ø8	ØØØØ
GENUV1.	BOUNDST	REAL*4	46Ø8	12ØØ
RESOLV.	VVEL	REAL*4	4	ØØØØ
FLOWIN.	CAUSO	REAL*4	4	ØØØc
GENUV2.	BOUNDST	REAL*4	46Ø8	24ØØ
ALAMDA.	CAUSO	REAL*4	4	ØØ1Ø
NSWPV	VVEL	INTEGER*4	4	ØØØ4
GENUV3.	BOUNDST	REAL*4	46Ø8	36ØØ
RSMALL.	CAUSO	REAL*4	4	ØØ14
URFV.	VVEL	REAL*4	4	ØØØ8
RLARGE.	CAUSO	REAL*4	4	ØØ18
DYNPV	VVEL	REAL*4	96	ØØØc
UVC	LOCALST	REAL*4	46Ø8	ØØØØ
DYPSV	VVEL	REAL*4	96	Ø06c
XTOTA	CAUSO	REAL*4	4	ØØ1c
UWC	LOCALST	REAL*4	46Ø8	12ØØ

Microsoft FORTRAN Optimizing Compiler Version 5.00

BOUNDARY Local Symbols

Name	Class	Type	Size	Offset
SNSV.	VVEL	REAL*4	96	00cc
JSTEP	CAUSO	INTEGER*4	4	0020
VWC	LOCALST	REAL*4	4608	2400
ISTEP	CAUSO	INTEGER*4	4	0024
JSTP1	CAUSO	INTEGER*4	4	0028
RESOLW.	WVEL	REAL*4	4	0000
JSTM1	CAUSO	INTEGER*4	4	002c
NSWPW	WVEL	INTEGER*4	4	0004
ISTP1	CAUSO	INTEGER*4	4	0030
URFW.	WVEL	REAL*4	4	0008
ISTM1	CAUSO	INTEGER*4	4	0034
RESOLM.	CORRP	REAL*4	4	0000
JEXIT	CAUSO	INTEGER*4	4	0038
NSWPP	CORRP	INTEGER*4	4	0004
JEXITP1	CAUSO	INTEGER*4	4	003c
URFP.	CORRP	REAL*4	4	0008
INLETI.	CAUSO	INTEGER*4	4	0040
DU.	CORRP	REAL*4	4608	000c
INLETF.	CAUSO	INTEGER*4	4	0044
DV.	CORRP	REAL*4	4608	120c
JINLET.	CAUSO	INTEGER*4	4	0048
IPREF	CORRP	INTEGER*4	4	240c
RGARG	CAUSO	REAL*4	4	004c
JPREF	CORRP	INTEGER*4	4	2410
NGARG	CAUSO	INTEGER*4	4	0050
NGARM1.	CAUSO	INTEGER*4	4	0054
NGARP1.	CAUSO	INTEGER*4	4	0058
U	VAR	REAL*4	4608	0000
V	VAR	REAL*4	4608	1200
SUKD.	SUSPTMP	REAL*4	4608	0000
W	VAR	REAL*4	4608	2400
SPKD.	SUSPTMP	REAL*4	4608	1200
P	VAR	REAL*4	4608	3600
PP.	VAR	REAL*4	4608	4800
GEN	TURBU	REAL*4	4608	0000
TE.	VAR	REAL*4	4608	5a00
ED.	VAR	REAL*4	4608	6c00
CD.	TURBU	REAL*4	4	1200
CMU	TURBU	REAL*4	4	1204
C1E	TURBU	REAL*4	4	1208
IT.	GERAL	INTEGER*4	4	0000
C2E	TURBU	REAL*4	4	120c
JT.	GERAL	INTEGER*4	4	0004
NI.	GERAL	INTEGER*4	4	0008
BETARODI.	TURBU	REAL*4	4	1210
CAPPA	TURBU	REAL*4	4	1214
NJ.	GERAL	INTEGER*4	4	000c
ELOG.	TURBU	REAL*4	4	1218

BOUNDARY Local Symbols

Name	Class	Type	Size	Offset
NIM1	GERAL	INTEGER*4	4	ØØ1Ø
PRED	TURBU	REAL*4	4	121c
NJM1	GERAL	INTEGER*4	4	ØØ14
PRTE	TURBU	REAL*4	4	122Ø
GREAT	GERAL	REAL*4	4	ØØ18
C1	TURBU	REAL*4	4	1224
JMAX	GERAL	INTEGER*4	192	ØØ1c
C2	TURBU	REAL*4	4	1228
JMAXP1	GERAL	INTEGER*4	192	ØØdc
UMC2	TURBU	REAL*4	4	122c
C1M1	TURBU	REAL*4	4	123Ø
X	GEOM	REAL*4	192	ØØØØ
Y	GEOM	REAL*4	96	ØØcØ
CK	TURBU	REAL*4	4	1234

Global Symbols

Name	Class	Type	Size	Offset
BOUNDARY	FSUBRT	***	***	ØØØØ
BOUNDST	common	***	18432	ØØØØ
CAUSO	common	***	92	ØØØØ
COEF	common	***	32256	ØØØØ
CORRP	common	***	9236	ØØØØ
FLUPR	common	***	4624	ØØØØ
GEOM	common	***	23Ø8	ØØØØ
GERAL	common	***	412	ØØØØ
LOCALSTRESS	common	***	13824	ØØØØ
STRESS	common	***	27648	ØØØØ
SUSPTMP	common	***	9216	ØØØØ
TURBU	common	***	4668	ØØØØ
UVEL	common	***	588	ØØØØ
VAR	common	***	32256	ØØØØ
VVEL	common	***	3ØØ	ØØØØ
WALLF	common	***	768	ØØØØ
WVEL	common	***	16	ØØØØ

Code size = 45cd (17869)
Data size = ØØ3c (6Ø)
Bss size = ØØ88 (136)

No errors detected

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

```
 1
 2
 3          subroutine stone1 (istart,jstart,ni,jmax,it,jt,phi,nchap)
 4
 5   cØ******************      preliminaires
 6
 7          logical istone,isimplec
 8
 9          dimension phi(it,jt),a(48),b(48),c(48),d(48),jmax(it)
10          common
11         1/causo/uin,tein,edin,flowin,alamda,
12         2        rsmall,rlarge,xtota,jstep,istep,jstp1,jstm1,istp1,istm1,
13         3        jexit,jexitp1,inleti,inletf,jinlet,rgarg,ngarg,ngarm1,
14         4        ngarp1
15         5/coef/ap(48,24),an(48,24),as(48,24),ae(48,24),aw(48,24),su(48,24)
16         6       sp(48,24)
17         7/stone/tetastone,source,istone,niter,isimplec
18
19   c---------------------     vertical sweep and horizontal sweep
20
21            alg=tetastone-1.
22            jsm1=jstart-1
23            nim1=ni-1
24            a(jsm1)=Ø
25
26   c1******************      w-e sweep
27
28                      do 1ØØ i=istart,nim1
29
30                      c(jsm1)=phi(i,jsm1)
31
32   c---------------------     s-n traverse
33
34          ji=jmax(i)
35          if (nchap .eq. 2) ji=jmax(i-1)
36          do 1Ø1 j=jstart,ji
37
38   c---------------------     tdma coefficients
39
40          a(j)=an(i,j)
41          b(j)=as(i,j)
42          c(j)=ae(i,j)*(phi(i+1,j)-alg*phi(i,j))+aw(i,j)*phi(i-1,j)+su(i,j)
43          d(j)=ap(i,j)-ae(i,j)*alg
44
45   c---------------------     recurrence formula
46
47          term=1./(d(j)-b(j)*a(j-1))
48          a(j)=a(j)*term
49          c(j)=(c(j)+b(j)*c(j-1))*term
50   1Ø1    continue
51
52   c---------------------     new phi s
53
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 5.00

```
54          do 102 jj=jstart,ji
55          j=ji+1+jsm1-jj
56    102   phi(i,j)=a(j)*phi(i,j+1)+c(j)
57
58
59    100                 continue
60
61
62    c2********************     s-n sweep
63
64              ism1=istart-1
65              nk1=jmax(istart)
66              a(ism1)=0
67
68
69                        do 200 j=jstart,nk1
70
71                        c(ism1)=phi(ism1,j)
72
73    c-----------------------    w-e traverse
74
75                         do 202 i=istart,nim1
76
77
78    c-----------------------    tdma coefficients
79
80          a(i)=ae(i,j)
81          b(i)=aw(i,j)
82          c(i)=an(i,j)*(phi(i,j+1)-alg*phi(i,j))+as(i,j)*phi(i,j-1)+su(i,j)
83          d(i)=ap(i,j)-an(i,j)*alg
84
85    c-----------------------    recurrence formula
86
87          term=1./(d(i)-b(i)*a(i-1))
88          a(i)=a(i)*term
89          c(i)=(c(i)+b(i)*c(i-1))*term
90    201   continue
91
92    202   continue
93
94    c-----------------------    new phi s
95
96          do 203 icone=istart,nim1
97          i=nim1+istart-icone
98    203   phi(i,j)=a(i)*phi(i+1,j)+c(i)
99
100
101   200                 continue
102
103             return
104
105                end
```

Microsoft FORTRAN Optimizing Compiler Version 5.Ø0

STONE1 Local Symbols

Name	Class	Type	Size	Offset
NCHAP	param			ØØØ6
PHI	param			Ø00a
JT	param			Ø00e
IT	param			Ø012
JMAX	param			Ø016
NI	param			Ø01a
JSTART	param			Ø01e
ISTART	param			Ø022
__V64	param			fff8
__V65	param			fffa
__V66	param			fffc
__V67	param			fffe
A	local	REAL*4	192	ØØØØ
B	local	REAL*4	192	Ø0c0
C	local	REAL*4	192	Ø180
D	local	REAL*4	192	Ø240
I	local	INTEGER*4	4	Ø300
NK1	local	INTEGER*4	4	Ø304
J	local	INTEGER*4	4	Ø308
JI	local	INTEGER*4	4	Ø30c
JJ	local	INTEGER*4	4	Ø310
ALG	local	REAL*4	4	Ø314
NIM1	local	INTEGER*4	4	Ø318
ISM1	local	INTEGER*4	4	Ø31c
JSM1	local	INTEGER*4	4	Ø320
ICONE	local	INTEGER*4	4	Ø324
TERM	local	REAL*4	4	Ø328
ISTONE	STONE	LOGICAL*4	4	ØØØ8
ISIMPLEC	STONE	LOGICAL*4	4	Ø010
UIN	CAUSO	REAL*4	4	ØØØØ
TEIN	CAUSO	REAL*4	4	ØØØ4
EDIN	CAUSO	REAL*4	4	ØØØ8
FLOWIN	CAUSO	REAL*4	4	ØØØc
ALAMDA	CAUSO	REAL*4	4	Ø010
RSMALL	CAUSO	REAL*4	4	Ø014
RLARGE	CAUSO	REAL*4	4	Ø018
XTOTA	CAUSO	REAL*4	4	Ø01c
JSTEP	CAUSO	INTEGER*4	4	Ø020
ISTEP	CAUSO	INTEGER*4	4	Ø024
JSTP1	CAUSO	INTEGER*4	4	Ø028
JSTM1	CAUSO	INTEGER*4	4	Ø02c
ISTP1	CAUSO	INTEGER*4	4	Ø030
ISTM1	CAUSO	INTEGER*4	4	Ø034
JEXIT	CAUSO	INTEGER*4	4	Ø038
JEXITP1	CAUSO	INTEGER*4	4	Ø03c
INLETI	CAUSO	INTEGER*4	4	Ø040
INLETF	CAUSO	INTEGER*4	4	Ø044
JINLET	CAUSO	INTEGER*4	4	Ø048
RGARG	CAUSO	REAL*4	4	Ø04c

Microsoft FORTRAN Optimizing Compiler Version 5.ØØ

STONE1 Local Symbols

Name	Class	Type	Size	Offset
NGARG	CAUSO	INTEGER*4	4	ØØ5Ø
NGARM1	CAUSO	INTEGER*4	4	ØØ54
NGARP1	CAUSO	INTEGER*4	4	ØØ58
AP	COEF	REAL*4	46Ø8	ØØØØ
AN	COEF	REAL*4	46Ø8	12ØØ
AS	COEF	REAL*4	46Ø8	24ØØ
AE	COEF	REAL*4	46Ø8	36ØØ
AW	COEF	REAL*4	46Ø8	48ØØ
SU	COEF	REAL*4	46Ø8	5aØØ
SP	COEF	REAL*4	46Ø8	6cØØ
TETASTONE	STONE	REAL*4	4	ØØØØ
SOURCE	STONE	REAL*4	4	ØØØ4
NITER	STONE	INTEGER*4	4	ØØØc

Global Symbols

Name	Class	Type	Size	Offset
CAUSO	common	***	92	ØØØØ
COEF	common	***	32256	ØØØØ
STONE	common	***	2Ø	ØØØØ
STONE1	FSUBRT	***	***	ØØØØ

Code size = Ø64f (1615)
Data size = ØØØ8 (8)
Bss size = Ø32c (812)

No errors detected