AD-A063 083 NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF F/G 20/4

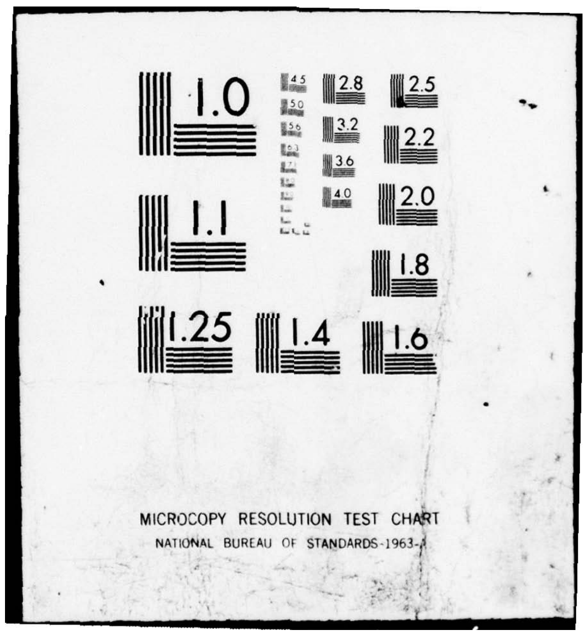
THEORETICAL ANALYSIS OF TRANSONIC FLOW PAST UNSTAGGERED OSCILLA--ETC(U)

SEP 78 P C OLSEN

UNCLASSIFIED NL

1 OF 2

AD A063083


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

(6) THEORETICAL ANALYSIS OF
TRANSONIC FLOW PAST
UNSTAGGERED OSCILLATING CASCADES,

by

(10) Peter Carlton/Olsen

(11) Sep[redacted]978

Thesis Advisor: M.F. Platzer

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Theoretical Analysis of Transonic Flow Past Unstaggered Oscillating Cascades		5. TYPE OF REPORT & PERIOD COVERED Engineer's Thesis September 1978
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Peter Carlton Olsen		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE September 1978
		13. NUMBER OF PAGES 134
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Oscillating Airfoil
Collocation Solution
Potential Flow
Transonic Flow

79 01 09 010

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper presents an independent verification of the collocation method as a technique for calculating the lift on an oscillating airfoil in an unstaggered cascade immersed in transonic flow. This method was → next page

DD FORM 1473 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(20. ABSTRACT Continued)

originally proposed by Gorelov. Results presented here differ somewhat from those presented by him. Two formulations are shown; one is purely numerical, the second employs an analytic expansion for small frequency.

Approved for public release; distribution unlimited.

Theoretical Analysis of Transonic Flow Past Unstaggered Oscillating Cascades

by

Peter Carlton Olsen
Lieutenant, United States Coast Guard
B.S., United States Coast Guard Academy, 1970
M.S., University of West Florida, 1975
M.S.A.E., Naval Postgraduate School, 1977
M.S.O.R., Naval Postgraduate School, 1978

Submitted in partial fulfillment of the requirements for the degree of

AERONAUTICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
September 1978

Author ______________________

Approved by: ______________________ Thesis Advisor

______________________ Second Reader

______________________ Chairman, Department of Aeronautics

______________________ Dean of Science and Engineering

ABSTRACT

This paper presents an independent verification of the collocation method as a technique for calculating the lift on an oscillating airfoil in an unstaggered cascade immersed in transonic flow. This method was originally proposed by Gorelov. Results presented here differ somewhat from those presented by him. Two formulations are shown; one is purely numerical, the second employs an analytic expansion for small frequency.

TABLE OF CONTENTS

I. INTRODUCTION ---------------------------------- 14

II. UNSTEADY TRANSONIC FLOW THEORY -------------- 16

III. SMALL PERTURBATION THEORY OF TRANSONIC FLOW - 23

A. GENERAL CASE ---------------------------- 23

B. BOUNDARY CONDITION ---------------------- 25

C. NONDIMENSIONALIZATION ------------------- 26

D. HARMONIC OSCILLATIONS ------------------- 29

IV. LINEARIZATION OF THE GOVERNING EQUATION ----- 31

A. BASIC SOLUTION -------------------------- 31

B. BOUNDARY CONDITIONS --------------------- 34

C. INITIAL CONDITIONS ---------------------- 34

V. PROBLEM FORMULATION ------------------------- 36

A. CO-ORDINATE SYSTEM ---------------------- 36

B. BOUNDARY CONDITIONS --------------------- 37

1. Upstream Condition ------------------ 37

2. Flow Tangency Condition ------------- 37

C. BASIC SOLUTION TECHNIQUE ---------------- 38

D. COLLOCATION SOLUTION OF THE POTENTIAL EQUATION EXPANDED FOR SMALL k ----------- 51

1. Solution for the Unknown Potential Coefficients ------------------------ 51

2. Calculation of the Potential -------- 57

VI. RESULTS ------------------------------------- 63

VII. RECOMMENDATIONS ----------------------------- 70

APPENDIX A. PROGRAM DESCRIPTION ------------------- 73

LIST OF REFERENCES ------------------------------------ 133
INITIAL DISTRIBUTION LIST ---------------------------- 134

LIST OF SYMBOLS

a	=	local speed of sound	II
a_o	=	speed of sound in the uniform flow	III
c	=	blade semichord	III
f_j	=	elementary function used in collocation solution	V
F	=	specific energy ("head")	II
G	=	function describing the surface of the airfoil as a function of time	III
H	=	function specifying location of blade surface in the vertical axis	III
i	=	$\sqrt{-1}$	II,III,IV,V
k	=	Strouhal number, nondimensional frequency	III,IV,V
m	=	$\sqrt{(\gamma+1)w}$	IV,V
M	=	Mach number = $\frac{\lvert\vec{V}\rvert}{a}$	III
n	=	number of collocation points - 1, order of highest spanning function	V
p	=	pressure	II
	=	nondimensional interblade distance	V
R	=	universal gas constant	II
R.P.	=	"real part of"	III,IV,V
T	=	temperature	II
t	=	time, nondimensional time	II,III,IV,V
U_o	=	uniform velocity from infinity	II,III,VI
u	=	x-component of velocity	II,III
u^o, u^1	=	interference vertical velocities due to reference and adjacent blades respectively, solved so as to satisfy the tangential flow conditions	V,VI

u'	=	small disturbance velocity	III
v	=	y-component of velocity	II,III,IV,V
$\vec{V}$	=	general velocity vector	II
v'	=	small disturbance velocity	III
v^o, v^1	=	vertical velocities due to the reference and adjacent blades respectively, determined from the tangential flow condition	V
w	=	$\tilde{\phi}_x$, a constant used in Gorelov's approximation of the transonic flow potential	IV,V
x	=	horizontal coordinate, may be non-dimensional	
x_*	=	mp (transformed interblade distance in Gorelov's approximation)	V
x_ℓ	=	blade leading edge	IV
x_o	=	center of pitch of the unstaggered cascade	IV,V
y	=	vertical coordinate, may be non-dimensional	
y, y_1	=	vertical coordinates attached to the reference and adjacent blades respectively, may be non-dimensional	IV,V
z, z_1	=	transformed vertical coordinates used in Gorelov's approximation, attached to the reference and adjacent blades respectively. $z = my$, $z_1 = my_1$	IV,V
$\frac{D}{Dt}$	=	substantial derivative w.r.t. time $= \frac{\partial}{\partial t} + \frac{\partial x}{\partial t}\frac{\partial}{\partial x} + \frac{\partial y}{\partial t}\frac{\partial}{\partial y}$ $= \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y}$	II,III
$O(\omega^2)$	=	"of the order of magnitude of ω^2"	V
α	=	angle of attack	II,III,IV,V
α_o	=	maximum amplitude of pitch oscillations	IV

γ	=	ratio of specific heats, c_p/c_v	
δ_{io}	=	Dirac Delta function	V
		= 1 when $i = 0$ 0 when $i \neq 0$	
η	=	$\cos^{-1}(-x)$	V
η_*	=	$\cos^{-1}(1-x_*)$	V
$\bar{\eta}$	=	$\cos^{-1}(-s)$	V
$\hat{\eta}$	=	$\cos^{-1}(x_*-x)$	V
θ_j^o, θ_j^1	=	interference potential coefficients for reference and adjacent blades respectively	V
λ	=	k/m^2	IV,V
μ_j^o, μ_j^1	=	Fourier coefficients describing the motion of the reference and adjacent blades respectively	V
ν	=	angular frequency of oscillation	IV,V
ρ	=	density	II
σ	=	phase angle	V
τ	=	cascade solidity, $\frac{2}{p}$	V
Φ	=	general velocity potential	II,III,VI
Φ_o	=	uniform flow velocity potential	
$\tilde{\phi}$	=	steady flow perturbation potential	III,IV
ϕ^o, ϕ^1	=	perturbation potential in collocation solution due to reference and adjacent blades respectively	V,VI
Φ^o, Φ^1	=	tranformed potentials	V
ψ	=	oscillatory flow potential	III,IV,V
Φ	=	transformed oscillatory potential in Gorelov's coordinates corresponding to ψ	V
ψ^o, ψ^1	=	interference potentials due to reference and adjacent blades respectively	V,VI

Ψ^o, Ψ^1 = transformed potentials in Gorelov's coordinates, coresponding to ψ^o and ψ^1

ω = $\dfrac{k(1-m)^2}{m^4}$

∇ = $\vec{i}\,\dfrac{\partial}{\partial x} + \vec{j}\,\dfrac{\partial}{\partial y}$, Gradient operator, $\vec{i}$ and $\vec{j}$ are unit vectors in the x and y direction respectively

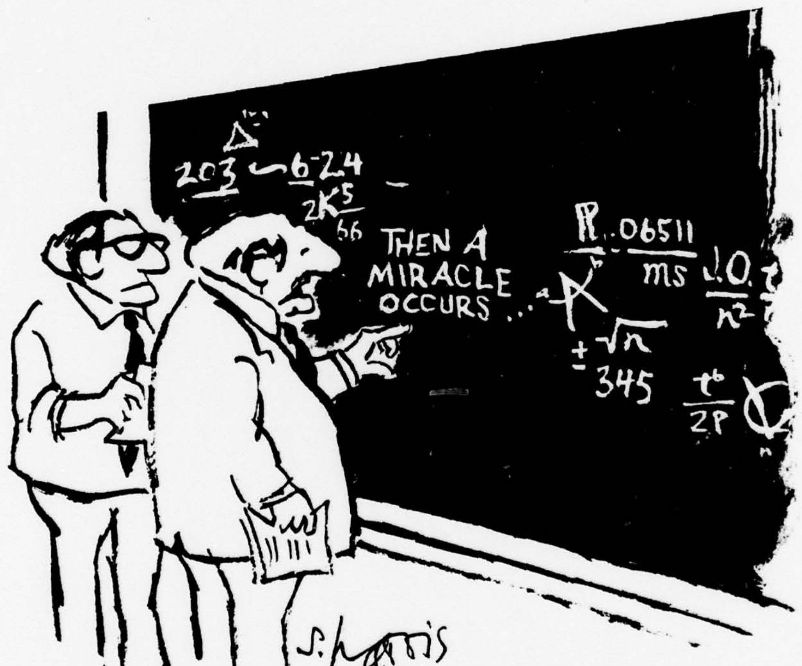
Computer Variables

DK	=	Reduced Frequency, k
DLAMDA	=	λ
DM2	=	m^2
DR	=	mp = x_*
ETA	=	η
ETASTR	=	η_*
IPT	=	Print Parameter
N	=	n
NF	=	not used
OFFSET	=	r
OMEGA	=	ω
QALPHA	=	$0 + i(\lambda - k)$
QCONST	=	$e^{i\sigma}$
QDCL	=	$C\ell_\alpha$
QDCM	=	Cm_α
QDK	=	$0 + ik$
QEXP	=	$0 + i\lambda$
QINTAP,QINTRP	=	Variables used to transmit boundary condition integrals
Q1ABCF,Q1RBCF	=	Interference coefficients for adjacent and reference blades
Q1COF	=	Right hand side vector in collocation solution
Q1INT	=	Known matrix of integrals in collocation solution
Q1RBP,Q1ABP	=	Not used
Q2CP	=	Not used
Q2EXP	=	$e^{-i\lambda x}$

Q2PT	=	Not used
RHO	=	vertical displacement
RHP	=	local input variable for rho
SIGMA	=	σ
TAU	=	τ
XASTN	=	Current x station in adjacent blade coordinates
XSTN	=	Current x station in reference blade coordinates

ACKNOWLEDGEMENT

I would like to thank Professor Max F. Platzer without whose infinite patience this thesis would not have been written.



"I think you should be more explicit here in step two."

I. INTRODUCTION

The analysis of unsteady transonic flows in aircraft turbopropulsion is an area of intense current interest. Rising fuel prices and increasing thrust requirements both point toward the need of turbomachinery capable of performing well with transonic or supersonic internal flow. But, increased flow has increased both the costs and uncertainties of engine designs. Flutter problems have already become a major consideration in engine development. Problems unforeseen in earlier days of turbine engine production have caused long development delays, or forced acceptance of engines producing less than their initial design thrust. These uncertainties cannot be avoided when an attempt is made to extend the state of the art, but they can be reduced by extending the range of analytical modeling.

Such extension must now be done piecemeal. The three-dimensional flows in turbomachinery, including the simultaneous effects of boundary layers, rotation, finite blade thickness, spanwise Mach distributions, and shocks, are well beyond present capability. Perhaps one day complete analysis will be practical, but it is not today. The best that can be done now is to approach the problem from one aspect at a time. Flow through a two dimensional cascade has been a useful tool in this partial analysis.

This thesis was originally to have been an extension of the work of Elder [1] and Schlein [2] to the case of a staggered cascade. Their work, based on Teipel's [3] linearization of the unsteady transonic small perturbation equation, analyzed transonic flow through oscillating unstaggered cascades by use of the collocation method. While the problem was easy to state, it was difficult to solve. Both Elder and Schlein had encountered difficulty in employing the collocation method. Therefore, it was decided that verification of the basic collocation solution presented by Gorelov [4] using a different linearization would be a worthwhile goal in itself.

The following investigation presents a verification of the development in [4], along with numerical results and suggestions for further work.

II. UNSTEADY TRANSONIC FLOW THEORY

Considering inviscid flow only, the following four equations govern the aerodynamic flow problem at hand:

The equation of state

$$p = \rho RT \qquad \text{(II-1)}$$

and the equations for the conservation of

1. Mass: $\operatorname{div}(\rho\vec{v}) + \frac{\partial\rho}{\partial t} = 0$ (II-2)

2. Momentum: $\frac{D\vec{v}}{Dt} + \frac{1}{\rho}\nabla p = 0$ (II-3)

3. Energy: $\frac{DS}{Dt} = 0$ (II-4)

where

$\vec{v}$ = velocity

p = pressure

S = entropy

R = universal gas constant

T = temperature

t = time

ρ = density

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \frac{\partial}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial}{\partial y}\frac{\partial y}{\partial t}$$

The analysis starts with a uniform flow from infinity. This flow has velocity U_o parallel to the x-axis. This formulation can be simplified by working with the total velocity potential, Φ, where

$$u = \frac{\partial \Phi}{\partial x} = \Phi_x = x \text{ component of velocity} = \frac{\partial x}{\partial t} \quad \text{(II-5)}$$

$$v = \frac{\partial \Phi}{\partial y} = \Phi_y = y \text{ component of velocity} = \frac{\partial y}{\partial t} \quad \text{(II-6)}$$

Thus, the initial uniform flow is represented by the uniform flow potential

$$\Phi_o = U_o x \quad \text{(II-7)}$$

This notation may be applied to the conservation equations for mass and momentum. The equation for conservation of mass

$$\text{div}(\rho\vec{v}) + \frac{\partial \rho}{\partial t} = 0 \quad \text{(II-8)}$$

becomes for two-dimensional unsteady flow

$$\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial \rho}{\partial t} = 0$$

$$\left[\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} + v\frac{\partial \rho}{\partial y}\right] + \rho\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0 \quad \text{(II-9)}$$

but

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} + v\frac{\partial \rho}{\partial y} = \frac{D\rho}{Dt}$$

and

$$u = \Phi_x \quad \text{and} \quad v = \Phi_y$$

Thus

$$\frac{D\rho}{Dt} + \rho(\Phi_{xx} + \Phi_{yy}) = 0$$

and

$$\Phi_{xx} + \Phi_{yy} = -\frac{1}{\rho}\frac{D\rho}{Dt} \qquad \text{(II-10)}$$

The speed of sound is given by

$$a^2 = \frac{dp}{d\rho}$$

Thus

$$\frac{D\rho}{Dt} = \frac{d\rho}{dp} \cdot \frac{Dp}{Dt} = \frac{1}{a^2}\frac{Dp}{Dt}$$

Applying this to equation (II-10) yields

$$\Phi_{xx} + \Phi_{yy} = -\frac{1}{\rho a^2}\frac{Dp}{Dt} \qquad \text{(II-11a)}$$

$$\Phi_{xx} + \Phi_{yy} = -\frac{1}{\rho a^2}(up_x + vp_y + p_t) \qquad \text{(II-11b)}$$

$$= -\frac{1}{\rho a^2}[(\nabla\Phi)\cdot(\nabla p) + p_t] \qquad \text{(II-11c)}$$

where ∇ is the gradient operator, $P_x = \frac{\partial P}{\partial x}$, $P_y = \frac{\partial P}{\partial y}$, $P_t = \frac{\partial P}{\partial t}$.

Laying this aside for the moment, consider the momentum equation (II-3)

$$\frac{D\vec{v}}{Dt} + \frac{1}{\rho} \nabla p = 0$$

$$\frac{D\vec{v}}{Dt} = \frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\, \vec{v}$$

Thus

$$\rho[\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\, \vec{v}] = \nabla p \qquad \text{(II-12)}$$

$$\vec{v} = \nabla \Phi$$

so

$$\frac{\partial \vec{v}}{\partial t} = \frac{\partial}{\partial t}(\nabla \Phi) = \nabla \frac{\partial \Phi}{\partial t} \qquad \text{(II-13)}$$

and

$$(\vec{v} \cdot \nabla)\, \vec{v} = \nabla \frac{V^2}{2} - \vec{v} \times (\nabla \times \vec{v})$$

where

$$V^2 = u^2 + v^2$$

$$= \vec{v} \cdot \vec{v}$$

For irrotational flow

$$\nabla \times \vec{v} = 0$$

thus

$$(\vec{v} \cdot \nabla)\, \vec{v} = \frac{\nabla V^2}{2} \qquad \text{(II-14)}$$

Thus

$$\frac{\nabla p}{\rho} + \nabla[\Phi_t + \frac{V^2}{2}] = 0 \qquad \text{(II-15)}$$

which after integration along a streamline becomes

$$\int\frac{dp}{\rho} + \Phi_t + \frac{V^2}{2} = F(t) \qquad \text{(II-16)}$$

For uniform flow from infinity $F(t) = \frac{1}{2} U_o{}^2$ and thus the final result is

$$\int\frac{dp}{\rho} + \Phi_t + \frac{V^2}{2} = \frac{1}{2} U_o{}^2 \qquad \text{(II-17)}$$

Differentiation with respect to t gives

$$p_t = -\rho(\Phi_{tt} + \frac{1}{2}\frac{\partial V^2}{\partial t}) \qquad \text{(II-18)}$$

From (II-3)

$$-\nabla p = \rho \frac{D\vec{v}}{Dt} \qquad \text{(II-19)}$$

Substitute (II-18) and (II-19) into (II-11c) to obtain

$$\Phi_{xx} + \Phi_{yy} = \frac{1}{a^2}[(\nabla\Phi) \cdot \frac{D\vec{v}}{Dt} + \Phi_{tt} + \frac{1}{2}\frac{\partial V^2}{\partial t}] \qquad \text{(II-20)}$$

This may be further simplified

$$\frac{D\vec{v}}{Dt} = \frac{\partial\vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\,\vec{v}$$

$$= \frac{\partial\vec{v}}{\partial t} + \frac{1}{2}\nabla V^2$$

Hence

$$\Phi_{xx} + \Phi_{yy} = \frac{1}{a^2}[(\nabla\Phi)\cdot(\frac{\partial\vec{v}}{\partial t} + \frac{1}{2}\nabla V^2) + \Phi_{tt} + \frac{1}{2}\frac{\partial V^2}{\partial t}] \qquad \text{(II-21)}$$

Expanding terms

$$(\nabla\Phi) \cdot (\frac{\partial\vec{v}}{\partial t}) = \nabla\Phi \cdot (\frac{\partial}{\partial t}\nabla\Phi) = \Phi_x\Phi_{xt} + \Phi_y\Phi_{yt}$$

$$\nabla\Phi \cdot \frac{\nabla V^2}{2} = \frac{\Phi_x{}^2\Phi_{xx}}{2} + \frac{\Phi_y{}^2\Phi_{yy}}{2} + \frac{\Phi_x\Phi_y\Phi_{xy}}{2} + \frac{\Phi_y\Phi_x\Phi_{yx}}{2}$$

$$= \frac{\Phi_x{}^2\Phi_{xx}}{2} + \frac{\Phi_y{}^2\Phi_{yy}}{2} + \frac{2\Phi_x\Phi_y\Phi_{xy}}{2} .$$

$$\frac{1}{2}\frac{\partial V^2}{\partial t} = \frac{1}{2}\frac{\partial}{\partial t}[(\nabla\Phi) \cdot (\nabla\Phi)]$$

$$= \frac{1}{2}[\Phi_x\Phi_{xt} + \Phi_{xt}\Phi_x + \Phi_y\Phi_{yt} + \Phi_{yt}\Phi_y]$$

$$= \Phi_x\Phi_{xt} + \Phi_y\Phi_{yt}$$

The final result obtained by combining terms is

$$\left(1 - \frac{\Phi_x^2}{a^2}\right) {}_{xx} + \left(1 - \frac{\Phi_y^2}{a^2}\right) {}_{yy} - \frac{2\Phi_x\Phi_y\Phi_{xy}}{a^2}$$

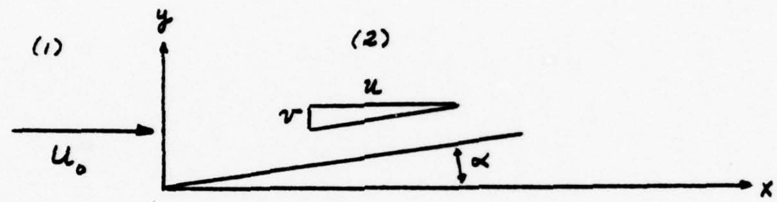
$$- \frac{2\Phi_x}{a^2}\Phi_{xt} - \frac{2\Phi_y}{a^2}\Phi_{yt} - \frac{1}{a^2}\Phi_{tt} = 0 \qquad \text{(II-22)}$$

This result is valid for irrotational, inviscid, two-dimensional, unsteady, compressible flows where gravity has been neglected.

III. SMALL PERTURBATION THEORY OF TRANSONIC FLOW

A. GENERAL CASE

A thin body at a small angle of attack will cause only a slight disturbance in the fluid. A flat plate is an example. Consider flow past a flat plate at angle of attack, α.



The flow at (2) must be parallel to the plate. To achieve this, small disturbance velocities u' and v' must be added to the free stream velocity yielding

$$u = U_o + u'$$

$$v = v'$$

The potential of the disturbed flow may be considered as the sum of the uniform flow potential, $\Phi_o = U_o x$, and the disturbance potential, ϕ

$$\Phi = \Phi_o + \phi \qquad \text{(III-1)}$$

Thus

$$\phi_x = U_o + u' \qquad \text{(III-2a)}$$

$$\phi_y = v \qquad \text{(III-2b)}$$

If ϕ is a function of time, then

$$\Phi_t = \phi_t$$

This result may be substituted into (II-22) leading to

$$[1 - \frac{(U_o + u')^2}{a^2}]\phi_{xx} + [1 - \frac{v^2}{a^2}]\phi_{yy} - 2\frac{(U_o + u')v}{a^2}\phi_{xy}$$

$$- 2\frac{(U_o + u')}{a^2}\phi_{xt} - \frac{2v}{a^2}\phi_{yt} - \frac{1}{a^2}\phi_{tt} = 0 \qquad \text{(III-3)}$$

This expands to yield

$$[1 - \frac{U_o^2 + 2U_o u' + u'^2}{a^2}]\phi_{xx} + [1 - \frac{v^2}{a^2}]\phi_{yy} - 2\frac{(U_o v + u'v)}{a^2}\phi_{xy}$$

$$- 2\frac{(U_o + u')}{a^2}\phi_{xt} - \frac{2v}{a^2}\phi_{yt} - \frac{1}{a^2}\phi_{tt} = 0 \qquad \text{(III-4)}$$

Equation (III-4) may be further simplified as shown by Landahl [3]. All non-linear terms except the $\phi_x\phi_{xx}$ product term can be neglected yielding the following transonic small disturbance equation

$$[(M^2-1) + (\gamma+1)M^2 \frac{\phi_x}{U_o}]\phi_{xx}$$

$$- \phi_{yy} + \frac{2M_o}{a_o}\phi_{xt} + \frac{1}{a_o}\phi_{tt} = 0 \qquad \text{(III-5)}$$

where a_o is the velocity of sound in the free-stream, and γ is the ratio of specific heats, and M = Mach number.

B. BOUNDARY CONDITION

The tangential flow condition requires that the flow be tangent to the airfoil surface at each instant of time. This means that no fluid may flow through the surface of the airfoil and is expressed by the condition

$$\frac{DG}{Dt} = 0 \quad \text{on} \quad G(x,y,t) \qquad \text{(III-6)}$$

where

G(x,y,t) describes the surface of the body as a function of time.

For a thin airfoil restricted to small oscillations, this may be written as

$$G = y - H(x,t) \qquad \text{(III-7)}$$

where

$H(x,t)$ is the function describing the position of the airfoil.

$H(x,t)$ can be written for harmonic pitch oscillations as

$$H(x,t) = R.P.[\alpha_o(x-x_o)\ e^{i\nu t}] \tag{III-8}$$

where the time-varying angle of attack $\alpha(t)$ is given by

$$\alpha(t) = R.P.[\alpha_o e^{i\nu t}]$$

and α_o = maximum amplitude of pitch oscillation

x_o is the pitch axis

ν is the angular frequency of oscillation

$i = \sqrt{-1}$

R.P. = "real part of"

Inserting (III-8) into the flow tangency condition (III-6) gives, after linearization,

$$\phi_y(x,0) = v(x,0) = \alpha_o[U_o + i\nu(x - x_o)]\ e^{i\nu t} \tag{III-9}$$

$$\text{on } y = 0$$

This is a condition for the normal velocity to be prescribed at the airfoil's mean position $y = 0$.

C. NONDIMENSIONALIZATION

The terms in equations (III-5) and (III-9) are dimensional. For the following calculations it is convenient to use non-dimensional quantities. Define non-dimensional time and length to be

$$\bar{x} = \frac{x}{c}$$

$$\bar{y} = \frac{y}{c} \qquad \text{(III-10)}$$

$$\bar{t} = \frac{tU_o}{c}$$

where

U_o = uniform velocity from infinity

c = reference length (blade semichord).

The velocity potential in equation (III-5) may be non-dimensionalized as follows. Let

$$\bar{\phi} = \frac{\phi}{U_o c}$$

Hence:

$$\phi = U_o c\bar{\phi}$$

$$\phi_x = U_o c\bar{\phi}_{\bar{x}}\left(\frac{1}{c}\right)$$

$$= U_o\bar{\phi}_{\bar{x}} \qquad \text{(III-11)}$$

and similarly for the other derivatives in (III-5), yielding

$$[(M^2-1) + (\gamma+1)M^2\bar{\phi}_{\bar{x}}]\phi_{\bar{x}\bar{x}} - \bar{\phi}_{\bar{y}\bar{y}} + 2M^2\bar{\phi}_{\bar{x}\bar{t}} + M^2\bar{\phi}_{\bar{t}\bar{t}} = 0 \quad \text{(III-12)}$$

This equation is non-dimensional.

The boundary condition given in equation (III-9) may be non-dimensionalized in a similar fashion

$$\phi_y(x,0) = \alpha_o[U_o + i\nu(x - x_o)]\, e^{i\nu t} \quad \text{(III-9)}$$

Thus

$$U_o c\bar{\phi}_y = \alpha_o[U_o + ik\frac{U_o}{c}(c\bar{x} - c\bar{x}_o]\, e^{i\nu\bar{t}\frac{c}{U_o}} \quad \text{(III-13)}$$

where

$$k = \frac{\nu c}{U_o} = \text{Strouhal number or reduced frequency}$$

$$U_o c\bar{\phi}_{\bar{y}}\cdot\frac{1}{c} = \alpha_o U_o[1 + ik(\bar{x} - \bar{x}_o)]\, e^{ik\bar{t}} \quad \text{(III-14)}$$

Thus

$$\bar{\phi}_{\bar{y}} = \bar{v}(\bar{x},0) = \alpha_o[1 + ik(\bar{x} - \bar{x}_o)]\, e^{ik\bar{t}} \quad \text{(III-15)}$$

Because the final operations are linear in α_o, set $\alpha_o = 1$, yielding

$$\bar{\phi}_{\bar{y}} = \bar{v}(\bar{x},0) = [1 + ik(\bar{x} - \bar{x}_o)]\, e^{ik\bar{t}} \quad \text{(III-16)}$$

The overbars denoting nondimensional quantities will be dropped from the remainder of the paper. All further quantities shall be assumed appropriately non-dimensional. This yields the following final equations

$$[(M^2-1)+(\gamma+1)M^2\phi_x]\phi_{xx} - \phi_{yy} + 2M^2\phi_{xt} + M^2\phi_{tt} = 0 \tag{III-17}$$

and

$$\phi_y(x,0) = v(x,0) = [1 + ik(x - x_o)]\, e^{ikt} \tag{III-18}$$

where

all quantities are nondimensional and

$\alpha_o = 1$

D. HARMONIC OSCILLATIONS

In the case of harmonic oscillations, equation (III-17) may be simplified still further.

Let

$$\phi = \tilde{\phi} + \text{R.P.}[\psi e^{ikt}]$$

where

$\tilde{\phi}$ = non-dimensional steady flow potential

ψ = non-dimensional oscillatory flow potential

R.P. = "real part of"

Equation (III-17) then becomes

$$(1-M^2)\psi_{xx} + \psi_{yy} - M^2(\gamma+1)[\psi_x\psi_{xx} + \tilde{\phi}_x\psi_{xx} + \tilde{\phi}_{xx}\psi_x]$$
$$+ M^2k^2\psi - 2iMk^2\psi_x = 0 \qquad \text{(III-19)}$$

For M close to 1, this is a nonlinear mixed elliptic-hyperbolic partial differential equation with variable coefficients, the exact type depending on $\tilde{\phi}_x$ and $\tilde{\phi}_{xx}$. However, because flutter analysis is primarily concerned with the stability of small perturbations about a steady flow, the oscillatory component may be assumed small compared to the steady flow potential and therefore the product term $\psi_x\psi_{xx}$ may be neglected, yielding,

$$(1-M^2)\psi_{xx} + \psi_{yy} - M^2(\gamma+1)[\tilde{\phi}_x\psi_{xx} + \tilde{\phi}_{xx}\psi_x]$$
$$= 2iM^2k\psi_x + M^2k^2\psi = 0 \qquad \text{(III-20)}$$

IV. LINEARIZATION OF THE GOVERNING EQUATION

The basic flutter equation, (III-20), is still a non-linear, mixed elliptic-hyperbolic partial differential equation with variable coefficients and difficult to solve. It may yet be further simplified.

A. BASIC SOLUTION

For M = 1, equation (III-20) becomes

$$\psi_{yy} - (\gamma+1)[\tilde{\phi}_x\psi_{xx} + \tilde{\phi}_{xx}\psi_x] - 2ik\psi_x + k^2\psi = 0 \qquad \text{(IV-1)}$$

Now assume

$$\tilde{\phi}_x \approx w = \text{constant} \qquad \text{(IV-2)}$$

$$\tilde{\phi}_{xx} \approx 0$$

throughout the interblade channel. Setting

$$\tilde{\phi}_x(\gamma+1) = w(\gamma+1) = m^2 \qquad \text{(IV-3)}$$

yields

$$m^2\psi_{xx} - \psi_{yy} + 2ik\psi_x - k^2\psi = 0 \qquad \text{(IV-4)}$$

The solution to this equation is found in Garrick and Rubinow [5]

$$\psi(x,y) = -\frac{1}{m} \int_{x_\ell}^{x-my} v(s) J_o[\omega\sqrt{(x-s)^2-(my)^2}] e^{i\lambda(s-x)} ds \quad \text{for} \quad y > 0 \tag{IV-5a}$$

and

$$\psi(x,y) = \frac{1}{m} \int_{x_\ell}^{x+my} v(s) J_o[\omega\sqrt{(x-s)^2-(my)^2}] e^{i\lambda(s-x)} dx \quad \text{for} \quad y < 0 \tag{IV-5b}$$

where

$$v(x) = \lim_{y \to 0} \frac{\partial}{\partial y} \psi(x,y) .$$

$v(x)$ may be obtained directly from the tangential flow boundary condition, and

$$\omega = \frac{k^2(1-m^2)}{m^4}$$

$$\lambda = \frac{k}{m^2}\sqrt{1+m^2} \simeq \frac{k}{m^2}$$ (where this paper employs the approximation used by Gorelov [4])

x_ℓ = blade leading edge,

Gorelov [4], has proposed a further simplification.

Set

$$z = my \tag{IV-6a}$$

$$\Psi(x,z) = \psi(x,y)\, e^{i\lambda x} . \tag{IV-6b}$$

Equation (IV-4) then becomes

$$\Psi_{xx} - \Psi_{zz} + \omega^2\Psi = 0$$

with solution

$$\Psi(x,z) = -\frac{1}{m} \int_{x_\ell}^{x-z} v(s) J_o[\omega\sqrt{(x-s)^2-z^2}]\, e^{i\lambda s} ds \tag{IV-7a}$$

$$z > 0$$

$$\Psi(x,z) = \frac{1}{m} \int_{x_\ell}^{x-z} v(s) J_o[\omega\sqrt{(x-s)^2-z^2}]\, e^{i\lambda s} ds \tag{IV-7b}$$

$$z < 0$$

where

$$v(x) = me^{-i\lambda x} \lim_{z\to 0} \Psi_z(x,z)$$

$v(x)$ is obtained from the tangential flow boundary condition.

For a thin body immersed in the flow, the solutions for $y > 0$, $z > 0$, and $y < 0$, $z < 0$ apply above the body along left-running Mach lines, or below along right running Mach lines respectively.

B. BOUNDARY CONDITIONS

1. Flow Tangency Condition

The boundary condition comes from the tangential flow condition, (III-18)

$$v(x) = [1 + ik(x - x_o)] \qquad \text{(IV-8)}$$

2. Upstream Condition

The final linearlized equation is a hyperbolic differential equation with boundary condition

$$\psi(x,y) = 0 \qquad \text{(IV-9)}$$

when

$$x - x_\ell < |my|$$

for the solution shown in equations (IV-5) or

$$\Psi(x,z) = 0$$

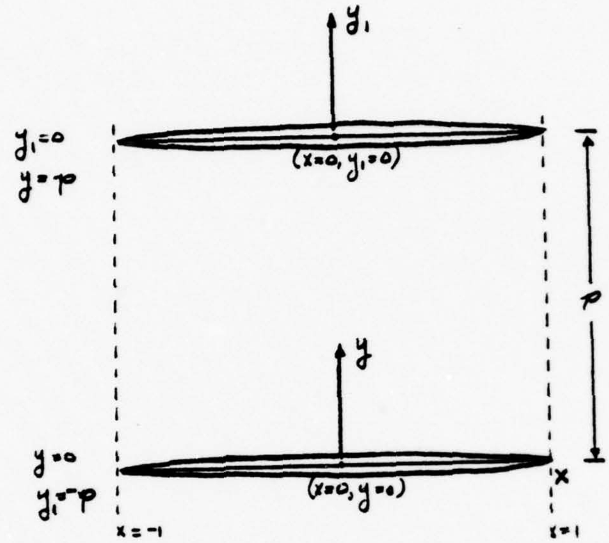
when

$$x - x_\ell < |z|$$

for the solution shown in equations (IV-7).

V. PROBLEM FORMULATION

A. CO-ORDINATE SYSTEM



Assume the geometry shown above. Both blades are thin airfoils of semichord c. All measurements are non-dimensional, normalized to c. The (x,y) co-ordinate system has its origin at the center of the reference (lower) blade. The (x,y_1) system is centered at the middle of the adjacent (upper) blade. The origin of the (x,y_1) system is located at (0,p) in the reference system. Generalizing this convention, the same symbols shall be used for the same quantities on both blades. Where discrimination is required, the quantity associated with the adjacent blade will be marked with superscript ^{1}, the quantity associated with the reference blade will be either unsuperscripted or marked with a superscript ^{o}.

Each blade is assumed to perform a small amplitude harmonic oscillation about its mid-chord point. Both blades are assumed to have identical reduced frequencies, k, and the motion of the adjacent blade lags that of the reference blade by a phase angle σ.

The blades are immersed in a uniform flow from the left at M = 1. The objective is to determine the oscillatory pressure distributions and aerodynamic forces generated by the blades' oscillations. Cascade solidity, $\tau = 2/p$.

B. BOUNDARY CONDITIONS

1. Upstream Condition

$$\psi = 0 \quad \text{whenever} \quad x + 1 < |my| \quad \text{and} \quad x + 1 < |my_1|, \text{ simultaneously} \qquad \text{(V-1)}$$

2. Flow Tangency Condition

Along the reference blade

$$\lim_{y \to 0} \psi_y(x,y) = (1 + ikx) \qquad \text{(V-2a)}$$

Along the adjacent blade

$$\lim_{y_1 \to 0} \psi_{y_1}(x,y_1) = (1 + ikx)e^{i\sigma} \qquad \text{(V-2b)}$$

where

σ is the phase angle between the blades oscillations

C. BASIC SOLUTION TECHNIQUE

Assume that the unsteady potential, ψ, may be written as the sum of four components

$$\psi(x,y) = \phi^o(x,y) + \psi^o(x,y) + \phi^1(x,y_1) + \psi^1(x,y_1) \qquad \text{(V-3)}$$

where:

ϕ^o = potential due to the reference blade alone, known from equation (IV-7)

ϕ^1 = potential due to the adjacent blade alone, known from equation (IV-7)

ψ^o = interference potential required to satisfy tangential flow condition along reference blade, unknown

ψ^1 = interference potential required to satisfy tangential flow condition along adjacent blade, unknown.

This total potential must satisfy the tangential flow condition at the plane of both the reference and adjacent blades. Thus

$$\phi^o_y(x,y=0) + \phi^1_{y_1}(x,y_1=-p) + \psi^o_y(x,y=0) + \psi^1_{y_1}(x,y_1=-p) \qquad \text{(V-4a)}$$

$$= (1 + ikx)$$

at the reference blade, and

$$\phi^o(x,y=p) + \phi^1_{y_1}(x,y_1=0) + \psi^o_y(x,y=p) + \psi^1_{y_1}(x,y_1=0)$$

$$= (1 + ikx)\, e^{i\sigma} \qquad \text{(V-4b)}$$

at the adjacent blade.

But from the unsteady potential solution for a single oscillating blade one has

$$\phi^o_y(x,y=0) = 1 + ikx \qquad \text{(V-5a)}$$

and

$$\phi^1_{y_1}(x,y_1=0) = (1 + ikx)\, e^{i\sigma} \qquad \text{(V-5b)}$$

Thus

$$\phi^1_{y_1}(x,y_1=-p) + \psi^1_{y_1}(x,y_1=-p) + \psi^o_y(x,y=0) = 0 \qquad \text{(V-6a)}$$

along the reference blade, and

$$\phi^o_y(x,y=p) + \psi^o_y(x,y=p) + \psi^1_{y_1}(x,y_1=0) = 0 \qquad \text{(V-6b)}$$

along the adjacent blade.

From equation (IV-7)

$$\phi^{o}(x,y) = -\frac{1}{m}\int_{-1}^{x-my} v^{o}(s)J_{o}[\omega\sqrt{(x-s)^{2}-(my)^{2}}]e^{i\lambda(s-x)}ds$$

$$y > 0 \qquad \text{(V-7a)}$$

$$= \frac{1}{m}\int_{-1}^{x+my} v^{o}(s)J_{o}[\omega\sqrt{(x-s)^{2}-(my)^{2}}]e^{i\lambda(s-x)}ds$$

$$y < 0 \qquad \text{(V-7b)}$$

where

$v^{o}(s) = 1 + iks$

$\lambda = k/m^{2}$

$m = (\gamma+1)w$

$\omega = \dfrac{k^{2}(1-m^{2})}{m^{4}}$

w = mean value of $\tilde{\phi}_{x}$ in the channel

$$\phi^{1}(x,y_{1}) = -\frac{1}{m}\int_{-1}^{x-my_{1}} v^{1}(s)J_{o}[\omega\sqrt{(x-s)^{2}-(my_{1})^{2}}]e^{i\lambda(s-x)}ds$$

$$y_{1} > 0 \qquad \text{(IV-8a)}$$

$$= \frac{1}{m}\int_{-1}^{x+my_{1}} v^{1}(s)J_{o}[\omega\sqrt{(x-s)^{2}-(my_{1})^{2}}]e^{i\lambda(s-x)}ds$$

$$y_{1} < 0 \qquad \text{(IV-8b)}$$

where

$$v^1(s) = (1 + iks)e^{i\sigma}$$

Henceforth attention will be restricted to the flow within the channel, $0 \leq y \leq p$, $-p \leq y_1 \leq 0$ leaving (IV-7a) and (IV-8b) as the governing equations of interest.

The two interference potentials are assumed to have forms identical to (IV-7a) and (IV-8b).

Set

$$\psi^o(x,y) = -\frac{1}{m} \int_{-1}^{x-my} u^o(s)J_o[\omega\sqrt{(x-s)^2-(my)^2}]e^{i\lambda(s-x)}ds \qquad y > 0 \qquad \text{(V-9a)}$$

$$\psi^1(x,y_1) = \frac{1}{m} \int_{-1}^{x+m} u^1(s)J_o[\omega\sqrt{(x-s)^2-(my_1)^2}]e^{i\lambda(s-x)}ds \qquad y_1 < 0 \qquad \text{(V-9b)}$$

where

$u^o(s)$ and $u^1(s)$ are unknown functions to be determined so as to satisfy equations (V-6)

Substitution of (V -7a), (V -8b) and (V -9) into (V -6) yields

$$u^o(x) + \psi^1_{y_1}(x,y_1{=}-p) + \phi^o_{y_1}(x,y_1{=}-p) = 0 \qquad \text{(V-10a)}$$

$$u^1(x) + \psi_y^o(x,y=p) + \phi_y^o(x,y=p) = 0 \qquad \text{(V-10b)}$$

Recalling Gorelov's transformation discussed in [4] and shown in equations (IV-6) above, set

$$\Phi^o(x,z) = \phi^o(x,y) \; e^{i\lambda x}$$

$$\Phi^1(x,z_1) = \phi^1(x,y_1) \; e^{i\lambda x}$$

$$\Psi^o(x,z) = \psi^o(x,y) \; e^{i\lambda x}$$

$$\Psi^1(x,z_1) = \psi^1(x,y_1) \; e^{i\lambda x}$$

where $z = my$

$z_1 = my_1$

Then

$$\frac{e^{i\lambda x}}{m} u^o(x) + \Phi^1_{z_1}(x,z_1=-x_*) + \Psi^1_{z_1}(x,z_1=-x_*) = 0 \qquad \text{(V-11a)}$$

and

$$\frac{e^{i\lambda x}}{m} u^1(x) + \Phi^o_z(x,z = x_*) + \Psi^o_z(x,z=x_*) = 0 \qquad \text{(V-11b)}$$

where

$$x_* = mp \; .$$

To employ the collocation method, assume that $u^1(x)$ and $u^o(x)$ can be approximated as the sum of a set of elementary functions f_j so that

$$u^o(x) \simeq \sum_{j=0}^{n} \theta_j^o f_j(x) \qquad \text{(V-12a)}$$

$$u^1(x) \simeq \sum_{j=0}^{n} \theta_j^1 f_j(x) \qquad \text{(V-12b)}$$

where $f_j(x) = 0$ when $x \leq x_* - 1$

Note that here both u^o and u^1 are expressed in terms of the same elementary functions, f_j.

Because of the slightly supersonic nature of the problem observe that $u^o(x) = 0$ and $u^1(x) = 0$ when $x \leq x_* - 1$.

Equations (V-12) may now be rewritten as

$$e^{i\lambda x}\sum\theta_j^o f_j(x) + \frac{\partial}{\partial z_1}\int_{x_*-1}^{x+z_1} \sum\theta_j^1 f_j(s) J_o[\omega\sqrt{(x-s)^2 - z_1^2}]e^{i\lambda s}ds$$

$$= -e^{i\sigma}\frac{\partial}{\partial z_1}\int_{-1}^{x+z_1} (1+iks) J_o[\omega\sqrt{(x-s)^2 - z_1^2}]e^{i\lambda s}ds$$

$$\text{at } z_1 = -x_* \qquad \text{(V-13a)}$$

$$e^{i\lambda x}\sum\theta_j^1 f_j(x) - \frac{\partial}{\partial z}\int_{x_*-1}^{x-z} \sum\theta_j^o f_j(s) J_o[\omega\sqrt{(x-s)^2-z}]e^{i\lambda s}ds$$

$$= \frac{\partial}{\partial z}\int_{-1}^{x+z} (1+iks) J_o[\omega\sqrt{(x-s)^2 - z^2}]e^{i\lambda s}ds$$

$$\text{at} \quad z = x_* \qquad \text{(V-13b)}$$

where

$$f_j(x) = 0 \quad \text{for} \quad x \leq x_* - 1$$

This simplifies to

$$e^{i\lambda x}\sum\theta_j^o f_j(x) + \sum\theta_j^1\{\frac{\partial}{\partial z_1}\int_{x_*-1}^{x+z_1} f_j(s) J_o[\omega\sqrt{(x-s)^2-z_1^2}]e^{i\lambda s}ds\}$$

$$= -e^{i\sigma}\frac{\partial}{\partial z_1}\int_{-1}^{x+z_1} (1+iks) J_o[\omega\sqrt{(x-s)^2-z_1^2}]e^{i\lambda s}ds$$

$$\text{at} \quad z_1 = -x_* \qquad \text{(V-14a)}$$

and

$$e^{i\lambda x}\sum\theta_j^1 f_j(x) - \sum\theta_j^o\{\frac{\partial}{\partial z}\int_{x_*-1}^{x-z} f_j(s) J_o[\omega\sqrt{(x-s)^2-z^2}]e^{i\lambda s}ds$$

$$= \frac{\partial}{\partial z}\int_{-1}^{x-z} (1+iks) J_o[\omega\sqrt{(x-s)^2-z^2}]e^{i\lambda s}ds \qquad \text{(V-14b)}$$

at $z = x_*$ where

$$f_j(x) = 0 \text{ for } x \leq x_*-1.$$

Performing the indicated differentiation yields

$$e^{i\lambda x}\sum\theta_j^o f_j(x) + \sum\theta_j^1\{\int_{x_*-1}^{x-x_*} f_j(s)\frac{J_1[\omega\sqrt{(x-s)^2-x_*^2}]\omega x_* e^{i\lambda s}ds}{\sqrt{(x-s)^2-x_*^2}}$$

$$+ f_j(x-x_*)e^{i\lambda(x-x_*)}\}$$

$$= e^{i\sigma}\{-\int_{-1}^{x-x_*}(1+iks)\frac{\omega x_* J_1[\omega\sqrt{(x-s)^2-x_*^2}]e^{i\lambda s}ds}{\sqrt{(x-s)^2-x_*^2}}$$

$$- [1+ik(x-x_*)]e^{i\lambda(x-x_*)}\} \qquad \text{(V-15a)}$$

and

$$e^{i\lambda x}\sum\theta_j^1 f_j(x) + \sum\theta_j^o\{\int_{x_*-1}^{x-x_*} f_j(s)\frac{J_1[\omega\sqrt{(x-s)^2-x_*^2}]\omega x_* e^{i\lambda s}ds}{\sqrt{(x-s)^2-x_*^2}}$$

$$+ f_j(x-x_*)e^{i\lambda(x-x_*)}\}$$

$$= -\int_{-1}^{x-x_*}(1+iks)\frac{\omega x_* J_1[\omega\sqrt{(x-s)^2-x_*^2}e^{i\lambda s}ds}{\sqrt{(x-s)^2-x_*^2}} - [1+ik(x-x_*)]e^{i\lambda(x-x_*)}$$

(V-15b)

Gorelov's formulation, equations [2.6, 2.7, 2.8, and 2.9] of [4], can be obtained directly from equations (V-15) by substituting

$$f_j(x) = \cos j\eta - \cos j\eta_*$$

$$v^o = \sum_{j=0}^{n} \mu_j^o \cos j\eta$$

$$v^1 = \sum_{j=0}^{n} \mu_j^1 \cos j\eta$$

where:

$$\eta = \cos(-x)$$

$$\eta_* = \cos(1-x_*)$$

In comparing the two systems care must be taken to note the differing symbols and coordinate systems. The corresponding quantities are:

Here	in[4]
θ_j^o, θ_j^1	$\nu_{o\sigma}$, $\nu_{i\sigma}$
μ_j^o, μ_j^1	$\theta_{o\sigma}$, $\theta_{i\sigma}$
$\eta = \cos^{-1}(-x)$	$\eta = \cos^{-1}(1-x)$
$\eta_* = \cos^{-1}(1-x_*)$	$\eta_* = \cos^{-1}(1-x_*)$
z, z_1	y, y_1
j	σ
σ	ψ

Here $-1 \leq x \leq 1$; in [4] $0 \leq x \leq 2$. This transformation accounts for the differing definitions of η. Making the substitutions results in the system

$$\sum_{j=0}^{n} \{\theta_j^o [\cos j\eta - (1-\delta_{o1}) \cos j\eta_*]$$

$$+ \theta_j^1 \int_{x_*-1}^{x-x_*} \frac{\partial}{\partial z_1} J_o [\omega\sqrt{(x-s)^2 - z_1^2}] [\cos j\hat{\eta} - (1-\delta_{io}) \cos j\eta_*] e^{i\lambda s} ds$$

$$+ \theta_i^1 [\cos j\bar{\eta} - (1-\delta_{io}) \cos j\eta_*] e^{i\lambda(x-x_*)}\}$$

$$= - \sum_{j=0}^{n} \{-\mu_j^1 \int_{-1}^{x-x_*} \frac{\partial}{\partial z_1} J_o [\sqrt{(x-s)^2 - z_1^2}] \cos j\hat{\eta} e^{i\lambda s} ds$$

$$- \mu_j^1 \cos j\bar{\eta} e^{i\lambda(x-x_*)}\} \qquad \text{(V-16a)}$$

at $z_1 = -x_*$

$x > x_* - 1$

and

$$\sum\{\theta_j^1[\cos j\eta - (1-\delta_{oj})\cos j\eta_*]$$

$$+ \theta_j^o \int_{x_*-1}^{x-x_*} \frac{\partial}{\partial z} J_o[\omega\sqrt{(x-s)^2-z^2}][\cos j\eta-(1-\delta_{oj})\cos j\eta_*]e^{i\lambda s}ds$$

$$+ \theta_j^o[\cos j\bar{\eta} - (1-\delta_{oj})\cos j\eta_*]e^{i\lambda(x-x_*)}\}$$

$$= \sum_{j=0}^{n}\{\mu_j^o \int_{-1}^{x-x_*} \frac{\partial}{\partial z} J_o[\omega\sqrt{(x-s)^2-z^2}]\cos j\hat{\eta}e^{i\lambda s}ds$$

$$- \mu_j^o \cos j\bar{\eta}e^{i\lambda(x-x_*)}\} \qquad \text{(V-16b)}$$

where:

$$\eta = \text{arc cos }(-x)$$

$$\eta_* = \text{arc cos }(1-x_*)$$

$$\bar{\eta} = \text{arc cos }(-s)$$

$$\hat{\eta} = \text{arc cos }(-x+x_*)$$

$$\delta_{oj} = \text{Dirac } \delta \text{ function } = \begin{matrix} 1 & \text{for} & j = 0 \\ 0 & \text{for} & j \neq 0 \end{matrix}$$

Given the change in coordinates and notation, this system is equivalent to that shown in [4].

This was the system programmed for computer solution. Because the function, $f_j(x)$, is unaffected by the differentiation with respect to y (or z) the exact form used need not be specified, so that the system shown in (V-15) may be programmed with f undetermined. A subroutine may be written to return the function desired and the remaining program left perfectly general. In the program developed with this thesis both the Gorelov functions shown above and the Legendre polynomials were employed. All the integrals may now be evaluated at n+1 points, x_i, on both blades in $mp-1 < x_i < 1$ and the resulting linear system solved for θ_j^o and θ_j^1, $j=1,2,\ldots,n+1$.

The interference potentials may be constructed by taking

$$\Psi^o(x,z) = \frac{-1}{m}\int_1^{x-z} [\sum \theta_j^o f_j(x)] J_o[\omega\sqrt{(x-s)^2-z^2}]e^{i\lambda s}ds \qquad z > 0 \qquad \text{(V-17a)}$$

$$\Psi^1(x,z_1) = \frac{1}{m}\int_{-1}^{x+z_1} [\sum \theta_j^1 f_j(x)] J_o[\omega\sqrt{(x-s)^2-z_1^2}]e^{i\lambda s}ds \qquad z_1 < 0 \qquad \text{(V-17b)}$$

where

$$f_j(x) = 0, \quad \text{for all} \quad x \leq x_*-1$$

Once the potentials have been calculated as outlined above, the surface pressure may be calculated using the relationship

$$C_p = -2(\psi_x + ik\psi) \quad \text{(V-18a)}$$

$$= -2[\Psi_x + i(k-\lambda)\Psi]e^{-i\lambda x} \quad \text{(V-18b)}$$

Because all the plates are assumed to be in steady oscillation with uniform phase shift, σ, between neighboring plates, then

$$v^1(x) = v^o(x)e^{i\sigma}, \quad u^1(x) = u^o(x)e^{i\sigma}, \quad \Psi(x,y) = -\Psi(x,-y)$$

in this case

$$C_{\ell_\alpha} = 2\int_{-1}^{1}[\Psi_x(x,+0) + i(k-\lambda)\Psi(x,+0)]e^{-i\lambda x}dx \quad \text{(V-19)}$$

where

$$\Psi(x_1,+0) = \frac{-1}{m}\int_{-1}^{x}[(1+iks)+u^o(s)]J_o[\omega(x-s)]e^{i\lambda(s)}ds$$

$$+\frac{e^{i\sigma}}{m}\int_{-1}^{x-x_*}[(1+iks)+u^o(s)]J_o[\omega\sqrt{(x-s)^2-(x_*)^2}]e^{i\lambda(s)}ds$$

where $x_* = mp$. (V-20)

Results from this approach, in the form of values of C_{ℓ_α} for $k = 0.1$, at various values of w are presented in the results for approximations based both on Gorelov's formulation, and on the Legendre polynomials.

D. COLLOCATION SOLUTION OF THE POTENTIAL EQUATION EXPANDED FOR SMALL k,

In order to provide a partially independent check of the results of the main program, the Gorelov function representation of the collocation solution was expanded for small k, and solved at two collocation points, $n = 2$. The resulting potentials, and partial derivatives with respect to x and y were then used to replace the corresponding numerical routines in the main program. The output resulting from the approximations were compared with the purely numerical results obtained from the computer program.

1. Solution For The Unknown Potential Coefficients

The basic system of linear equations used to determine the unknown coefficients is

$$\frac{1}{m} e^{i\lambda x} u^{o} + \Phi^{1}_{z_1} + \Psi^{1}_{z_1} = 0 , \quad z = 0, \quad z_1 = -x_* = -mp \tag{V-21a}$$

$$\frac{1}{m} e^{i\lambda x} u^{1} + \Phi^{o}_{z} + \Psi^{o}_{z} = 0 , \quad z_1 = 0, \quad z = x_* = mp \tag{V-21b}$$

where

$$0 = u^{o}(x) = u^{1}(x) \quad \text{when} \quad x \leq -1 + x_*$$

otherwise

$$u^o = \sum_{j=1}^{n} \theta_j^o(\cos j\eta - \cos j\eta_*) + \theta_o^o$$

$$u^1 = \sum_{j=1}^{n} \theta_j^1(\cos j\eta - \cos j\eta_*) + \theta_o^1$$

where

$$\eta = \text{arc cos } (-x)$$

$$\eta_* = \text{arc cos } (1- x_*) \ .$$

Thus, for n = 2, the system becomes

$$e^{i\lambda x}\{\theta_o^o + \theta_1^o(\cos \eta - \cos \eta_*)\}$$

$$+ \frac{\partial}{\partial z_1} \int_{-1}^{x+z_1} v^1(s)J_o [\omega \sqrt{(x-s)^2 - z_1^2}]e^{i\lambda s}ds$$

$$+ \frac{\partial}{\partial z_1} \int_{x_*-1}^{x+z_1} u^1(s)J_o[\omega\sqrt{(x-s)^2-z_1^2}]e^{i\lambda s}ds = 0$$

$$z_1 = -x_* \qquad \text{(V-22a)}$$

along the reference blade and

$$e^{i\lambda x}\{\theta_o^1 + \theta_1^1(\cos\eta - \cos\eta_*)\} - \frac{\partial}{\partial z}\int_{-1}^{x-z} v^o(s) J_o[\omega\sqrt{(x-s)^2 - z^2}]e^{i\lambda s}ds$$

$$- \frac{\partial}{\partial z}\int_{x_*-1}^{x-z} u^o(s) J_o[\omega\sqrt{(x-s)^2 - z^2}]e^{i\lambda s}ds \Big|_{z = x_*} . \qquad \text{(V-22b)}$$

along the adjacent blade

where $u^o(s) = \theta_o^o + \theta_1^o(\cos\eta - \cos\eta_*)$

$u^1(s) = \theta_o^1 + \theta_1^1(\cos\eta - \cos\eta_*)$

$v^o(s) = 1 + iks$

$v^1(s) = (1 + iks)e^{i\sigma}$

For k sufficiently small, this system may be further simplified by the following approximations

$$J_o[\omega \overline{(x-s)^2 - z^2}] \approx 1 - 0(\omega^2) \approx 1 \qquad \text{(V-23a)}$$

$$J_o[\omega\sqrt{(x-s)^2 - z^2}] \approx 1 - 0(\omega^2) \approx 1 \qquad \text{(V-23b)}$$

$$e^{i\lambda x} \approx 1 + i\lambda x - 0(\lambda^2 x^2) \approx 1 + i\lambda x$$

$$e^{i\lambda s} \approx 1 + i\lambda s - 0(\lambda^2 s^2) \approx 1 + i\lambda s$$

where $0(\omega^2)$ means "of the order of magnitude of ω^2"

$-1 \leq x \leq 1$, $-1 \leq s \leq x - x_*$

$\lambda = k/m^2$, $\omega^2 = \dfrac{k^2(1-m^2)}{m^4}$

The interference source distributions may be replaced by

$$u^o(s) = \theta_o^o + \theta_1^o(-s + x_*-1)$$

$$u^1(s) = \theta_o^1 + \theta_1^1(-s + x_*-1) .$$

If higher order terms are neglected, the result is a system linear in k and λ

$$(1+i\lambda x)[\theta_o^o + \theta_1^o(-x-1+x_*)] + \frac{\partial e^{i\sigma}}{\partial z_1} \int_{-1}^{x+z_1} (1+iks)(1+i\lambda s)ds$$

$$+ \frac{\partial}{\partial z_1} \int_{x_*-1}^{x+z_1} [\theta_o^1 + \theta_1^1(-s-1+x_*)](1+i\lambda s)ds = 0$$

$$z_1 = -x_* \qquad \text{(V-24a)}$$

$$(1+i\lambda x)[\theta_o^1 + \theta_1^1(-x-1+x_*)] - \frac{\partial}{\partial z} \int_{-1}^{x-z} (1-iks)(1+i\lambda s)ds$$

$$- \frac{\partial}{\partial z} \int_{x_*-1}^{x-z} [\theta_o^o + \theta_1^o(-s-1+x_*)](1+i\lambda s)ds = 0$$

$$z = x_* \qquad \text{(V-24b)}$$

Product terms containing $(k\lambda) = \frac{k^2}{m^2}$ may be neglected as of higher order in k, yielding

$$(1+i\lambda x)[\theta_o^o + \theta_1^o(-x-1+x_*)] + \frac{\partial e^{i\sigma}}{\partial z_1} \int_{-1}^{x+z_1} [1+i(\lambda+k)s]ds$$

$$+ \frac{\partial}{\partial z_1} \int_{x_*-1}^{x+z_1} \{[\theta_o^1 + \theta_1^1(-s-1+x_*)]$$

$$+ i\lambda s[\theta_o^1 + \theta_1^1(-s-1+x_*)]\}ds = 0$$

$$z_1 = -x_* \qquad \text{(V-25a)}$$

$$(1+i\lambda x)[\theta_o^1 + \theta_1^1(-x-1+x_*)] - \frac{\partial}{\partial z} \int_{-1}^{x-z} [1+i(\lambda+k)s]ds$$

$$- \frac{\partial}{\partial z} \int_{x_*-1}^{x-z} \{[\theta_o^o + \theta_1^o(-s-1+x_*)] + i\lambda s[\theta_1^o + \theta_1^o(-s-1+x_*)]\}ds = 0$$

$$z = x_* \qquad \text{(V-25b)}$$

Evaluating the indicated derivatives yields

$$(1+i\lambda x)[\theta_o^o + \theta_1^o(-x-1+x_*)] + [1+i(k+\lambda)(x-x_*)]e^{i\sigma}$$

$$+ [\theta_o^1 + \theta_1^1(2x_*-x-1)]+i\lambda(x-x_*)[\theta_o^1+\theta_1^1(2x_*-x-1)]\} = 0$$

(V-26a)

$$(1+i\lambda x)[\theta_o^1+\theta_1^1(-x-1+x_*)] + [1+i(\lambda+k)(x-x_*)]$$

$$+ \{[\theta_o^o+\theta_1^o(2x_*-x-1)] + i\lambda(x-x_*)[\theta_o^o+\theta_1^o(2x_*-x-1)]\} = 0$$

(V-26b)

Thus:

$$\theta_o^o(1+i\lambda x) + \theta_1^o[(-x-1+x_*) + i\lambda x(-x-1+x_*)]$$

$$+ \theta_o^1[1\ i\lambda(x-x_*)] + \theta_1^1[(2x_*-x-1) + i\lambda(x-x_*)(2x_*-x-1)]$$

$$= -e^{i\sigma}[1 + i(k+\lambda)(x-x_*)] \qquad \text{(V-27a)}$$

$$\theta_o^1[1 + i\lambda x] + \theta_1^1[(-x+1-x_*) + i\lambda x(-x+1-x_*)]$$

$$+ \theta_o^o[1+i\lambda x(x-x_*)] + \theta_1^o[(2x_*-x-1)+i\lambda(x-x_*)(2x_*-x-1)]$$

$$= -[1 + i(\lambda+k)(x-x_*)] \qquad \text{(V-27b)}$$

This system may be solved at two points, x_1 and x_2, for θ_o^o, θ_1^o, θ_o^1, and θ_1^1.

2. Calculation Of The Potential

The potential is given by

$$\psi(x,y) = \Psi(x,z)e^{-i\lambda x} \tag{V-28}$$

where

$$\Psi(x,z) = -\frac{1}{m}\int_{-1}^{x}[v^{o}(s)+u^{o}(s)]J_{o}[\omega\sqrt{(x-s)^{2}}]e^{i\lambda s}ds$$

$$+\frac{1}{m}\int_{-1}^{x-x_*}[v^{1}(s)+u^{1}(s)]J_{1}[\omega\sqrt{(x-s)^{2}-x_*^{2}}]e^{i\lambda s}ds.$$

$$u^{o}(s) = u^{1}(s) = 0 \quad \text{for all} \quad s \leq x_*-1 .$$

Thus

$$\Psi(x,z) = -\frac{1}{m}\int_{-1}^{x} v^{o}(s)J_{o}[\omega\sqrt{(x-s)^{2}}]e^{i\lambda s}ds$$

$$-\frac{1}{m}\int_{x_*-1}^{x} u^{o}(s)J_{o}[\omega\sqrt{(x-s)^{2}}]e^{i\lambda s}ds$$

$$+\frac{1}{m}\int_{-1}^{x-x_*} v^{1}(s)J_{o}[\omega\sqrt{(x-s)^{2}-x_*^{2}}]e^{i\lambda s}ds$$

$$+\frac{1}{m}\int_{x_*-1}^{x-x_*} u^{1}(s)J_{o}[\omega\sqrt{(x-s)^{2}-x_*^{2}}]e^{i\lambda s}ds \tag{V-29}$$

Making the same small frequency approximations as in the previous section yields

$$\Psi(x,z) = -\frac{1}{m}\int_{-1}^{x} v^{o}(s)(1+i\lambda s)ds$$

$$-\frac{1}{m}\int_{x_*-1}^{x} u^{o}(s)(1+i\lambda s)ds$$

$$+\frac{1}{m}\int_{-1}^{x-x_*} v^{1}(s)(1+i\lambda s)ds$$

$$+\frac{1}{m}\int_{x_*-1}^{x-x_*} u^{1}(s)(1+i\lambda s)ds \qquad \text{(V-30)}$$

From the general formulation

$$\Psi = \Phi^{1} + \Phi^{o} + \Psi^{1} + \Psi^{o} \qquad \text{(V-31)}$$

Thus, along the reference blade

$$-m\Phi^{o}(x,z=0) = \int_{-1}^{x} v^{o}(s)(1+i\lambda s)ds = \int_{-1}^{x}(1+iks)(1+i\lambda s)ds$$

$$= \int_{-1}^{x}[1+i(k+\lambda)s]ds = \left[s+i(k+\lambda)\frac{s^2}{2}\right]_{-1}^{x} \qquad \text{(V-32)}$$

$$= x + i\frac{(k+\lambda)}{2}x + 1 - i\left(\frac{k+\lambda}{2}\right)$$

$$\Phi^{o}(x,z=0) = -\frac{1}{m}[(1+x) + i(\frac{k+\lambda}{2})(x^2-1)] \qquad (V-33)$$

By inspection

$$\Phi^{1}(x,z_1=-x_*) = \frac{e^{i\sigma}}{m}\{(1+x-x_*) + i(\frac{k+\lambda}{2})[(x-x_*)^2-1]\} \qquad (V-34)$$

$$-m\Psi^{o}(x,z=0) = \int_{x_*-1}^{x} u^{o}(s)(1+i\lambda s)ds \qquad (V-35)$$

$$= \int_{x_*-1}^{x} [\theta_o^o + \theta_1^o(-s+x_*-1)](1+i\lambda s)ds$$

$$= \int_{x_*-1}^{x} \theta_o^o(1+i\lambda s) + \theta_1^o(-s+x_*-1)(1+i\lambda s)ds$$

$$= \theta_o^o[(x+1-x_*)+\frac{i\lambda}{2}(x^2-x_*^2+2x_*-1)]$$

$$+ \int_{x_*-1}^{x} \theta_1^o[-x+x_*-1+i\lambda(-s^2+sx_*-s)]ds$$

$$= \theta_o^o[(x+1-x_*)+\frac{i\lambda}{2}(x^2-x_*^2+2x_*-1)]$$

$$+ \theta_1^o\{(-\frac{s^2}{2}+sx_*-s)+i\lambda(-\frac{s^3}{3}+\frac{s^2x_*}{2}-\frac{s^2}{2})]_{x_*-1}^{x}\} \qquad (V-36)$$

$$= \theta_o^o[(x+1-x_*)+\frac{i\lambda}{2}(x^2-x_*^2+2x_*-1)]$$

$$+ \theta_1^o\{(-\frac{x^2}{2}+x_*-x)-[-\frac{(x_*-1)^2}{2}+x_*(x_*-1)-(x_*-1)]$$

$$+ i\lambda[(-\frac{x^3}{3}+\frac{x^2x_*}{2}-\frac{x^2}{2})+\frac{(x_*-1)^3}{3}-x_*\frac{(x_*-1)^2}{2}+\frac{(x_*-1)^2}{2}]\}$$

$$= \theta_o^o[(x+1-x_*)+\frac{i\lambda}{2}(x^2-x_*^2+2x_*-1)]$$

$$+ \theta_1^o\ -\frac{x^2}{2}+x(x_*-1)+\frac{(x_*-1)^2}{2} - (x_*-1)^2$$

$$+ i\lambda[-\frac{x^3}{3} + x^2\frac{(x_*-1)}{2} + \frac{x_*^3}{3} - x_*^2 + x_* - \frac{1}{3}$$

$$- \frac{x_*^3}{2} + x_*^2 - \frac{x_*}{2} + \frac{x_*^2}{2} - x_* + \frac{1}{2}]\}$$

$$= \theta_o^o[(x+1-x_*) + \frac{i\lambda}{2}(x^2 - x_*^2 + 2x_* - 1)]$$

$$+ \theta_1^o\{-\frac{x^2}{2} + x(x_*-1) - \frac{(x_*-1)^2}{2}$$

$$+ i\lambda[-\frac{x^3}{3} + \frac{x^2(x_*-1)}{2} - \frac{x_*^3}{6} + \frac{x_*^2}{2} - \frac{x_*}{2} + \frac{1}{6}]\}$$

$$\Psi^o = -\frac{1}{m}\{\theta_o^o[(x+1-x_*) + \frac{i\lambda}{2}(x^2 + x_*^2 + 2x_*-1)]$$

$$+ \theta_1^o\{-\frac{x^2}{2} + x(x_*-1) - \frac{(x_*-1)^2}{2}$$

$$+ i\lambda[-\frac{x^3}{3} + \frac{x^2(x_*-1)}{2} - \frac{(x_*-1)^3}{6}]\}\} \quad \text{(V-37)}$$

$m\psi^1(x,z_1=-x_*)$ may be evaluted by substituting $x-x_*$ for x in the expression for $m\Psi^o$ and exchanging θ_o^1 and θ_1^1 for θ_o^o and θ_1^o

$$m\Psi^1(x,z_1=-x_*) =$$
$$\theta_o^1\{[(x-x_*)+1-x_*] + \frac{i\lambda}{2}[(x-x_*)^2 - x_*^2 + 2x_* - 1]\}$$

$$+ \theta_1^1\{\frac{-(x-x_*)^2}{2} + (x-x_*)(x_*-1) - \frac{(x_*-1)^2}{2}$$

$$+ i\lambda[-\frac{(x-x_*)^3}{3} + \frac{x^2(x_*-1)}{2} - \frac{(x_*-1)^3}{6}]\} \quad \text{(V-38)}$$

$$= \theta_o^1 (x+1-2x_*) + \frac{i\lambda}{2}[(x^2-2xx_*+x_2^2) - x_*^2 + 2x_* - 1]\}$$

$$+ \theta_1^1\{-\frac{(x^2-2xx_*+x_*^2)}{2} + (x - \frac{3}{2}x_* + \frac{1}{2})(x_*-1)$$

$$+ i\lambda[-\frac{x^3}{3} + x^2x_* - xx_*^2 + \frac{x_*^3}{3} + (x^2-2xx_*+x_*^2)\frac{(x_*-1)}{2}$$

$$- \frac{(x_*-1)^3}{6}]\}$$

$$= \theta_o^1\{(x+1-2x_*) + \frac{i\lambda}{2}[x^2 - 2xx_* + 2x_* - 1]$$

$$+ \theta_1^1\{- \frac{x^2}{2} + xx_* - \frac{x_*^2}{2} + x(x-1) + \frac{1}{2}(1-3x_*)(x_*-1)$$

$$+ i\lambda[- \frac{x^3}{3} + x^2\frac{(3x_*-1)}{2} + x(- \frac{x_*^2-2xx_*^2-1)}{2}$$

$$+ \frac{x_*^3}{3} + \frac{x_*^3}{2} - \frac{1}{2}]\}$$

$\psi^1(x,z_1=-x_*)$

$$= \frac{1}{m}\{\theta_o^1\{(x+1-2x_*) + \frac{i\lambda}{2}[x^2 - 2xx_* + 2x_* - 1]$$

$$+ \theta_1^1\{[- \frac{x^2}{2} + x(2x_*-1) - xx_*^2 + 2x_* - \frac{1}{2}]$$

$$+ i\lambda[- \frac{x^3}{3} + x^2\frac{(3x_*-1)}{2} + x\frac{(-3x_*^2-1)}{2} + 5\frac{x_*^3}{6} - \frac{1}{2}]\}\}$$

(V-39)

A comparison of the results for the full program and the approximation is given below for $k = 0.01$, $w = 0.05$, $\sigma = \rho$, $n = 2$, yielding $\omega^2 \approx 6.1 \times 10^{-3}$, $\lambda \approx 0.083$

		Full program		Approx
$x = .1285$				
	ϕ	-5.363, .327i		-5.379, .549i
	ϕ_x	-8.634, .361i		-8.66, .379i
$x = .5643$				
	ϕ	-9.754, .809i		-9.804, 1.000i
	ϕ_x	-12.234, .692i		-12.272, .7608i
	C_{ℓ_α}	+31.658,-3.1032i	C_{ℓ_α}	31.7019,-3.2165i

VI. RESULTS

The collocation method was used to solve the partial differential equation resulting from the Gorelov approximation of transonic potential flow in an unstaggered cascade. The system was solved using both the spanning functions proposed by Gorelov in [4], resulting in the equations (V-16); and the Legendre polynomials, resulting in equations (V-15) with f_j replaced by the Legendre polynomial, P_j. The resulting values of C_{ℓ_α} for k = .1, $\tau = 1$, $\sigma = 1$ and seven collocation points on each blade are presented in figures VI-2, VI-3, and VI-4.

Figure VI -1 presents a diagram which is useful in commenting on the other results. This shows the location of the collocation points and first three interference reflections as a function of w expressed as a percentage of that portion of the chord subject to reflection. The collocation points are equally spaced throughout this interval, 12.5% from the leading edge of the interference zone, 12.5% between each pair of points and 12.5% from the blade trailing edge. The independent variable, w, is plotted vertically so that the dependent variable, percent of chord subject to interference, may be more conveniently visualized along the blade. (The curves are not precisely linear, but are very nearly so in the range shown.)

Figure VI -2 shows the C_{ℓ_α} calculated with k = 0.0 in comparison with the results obtained from Ackeret theory.

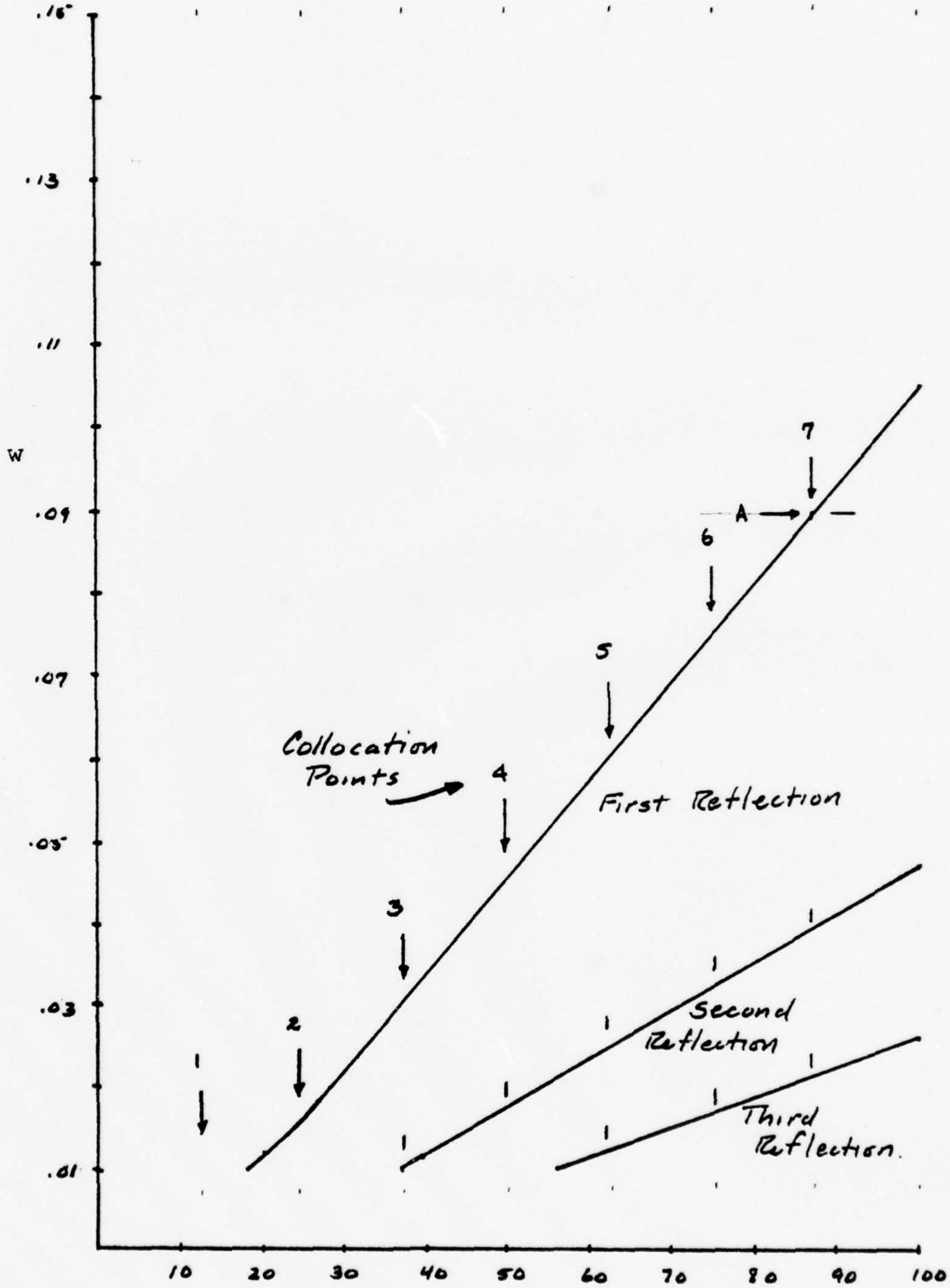
Agreement is good where there is no reflection and the portion of the blade subject to interference is affected by a constant interference potential, $w \geq 0.11$. Throughout the rest of the curve the results calculated here oscillate above and below the theoretical values. This appears to be due to the discrete nature of the approximation used in the collocation method. Rarely is the fraction of the chord subject to interference reflection equal to the fraction of the collocation points which feel it. Where the collocation point fraction lags, as near $w = 0.6$, the collocation results are lower than those due to Ackeret theory. When the collocation point fraction leads, as it does for $w \leq .04$ and briefly for $w \approx .08$, the the collocation results are higher than those due to Ackeret theory. The fault appears to be an intrinsic feature of the small number of points sampled. This results in a set of coefficients similar to those which would be obtained from a generalized Fourier series based on the integration of the Taylor series expansion about each point. This obviously cannot be a good approximation when both the function and its derivative are discontinuous at the reflections.

Figures VI -3 and VI -4 show the results of using Legendre polynomials and Gorelov's functions as spanning functions. The results for both formulations are identical. Gorelov's results are presented for comparison. Agreement is good for $w > 0.05$ except for an anomalous point, marked A .

It is believed that this anomaly is due to the location of the first reflection just ahead of the last collocation point (cf. "A" on Figure VI -1). This will yield a very small contribution from the reflection potential to the linear system from which the collocation points are determined. The resulting system will have a large dynamic range and may be ill-conditioned.

The discrepancy between these results, and those in [3] for $w < 0.5$ is still unexplained, as is the outlying value for $w = 0.5$.

The discontinuities in the imaginary results are believed to be due primarily to the reflection/collocation interaction explained above.



Percent of chord subject to interference

FIGURE VI-1. Location of Reflections and Collocation Points Shown as Percent of Chord Subject to Interference

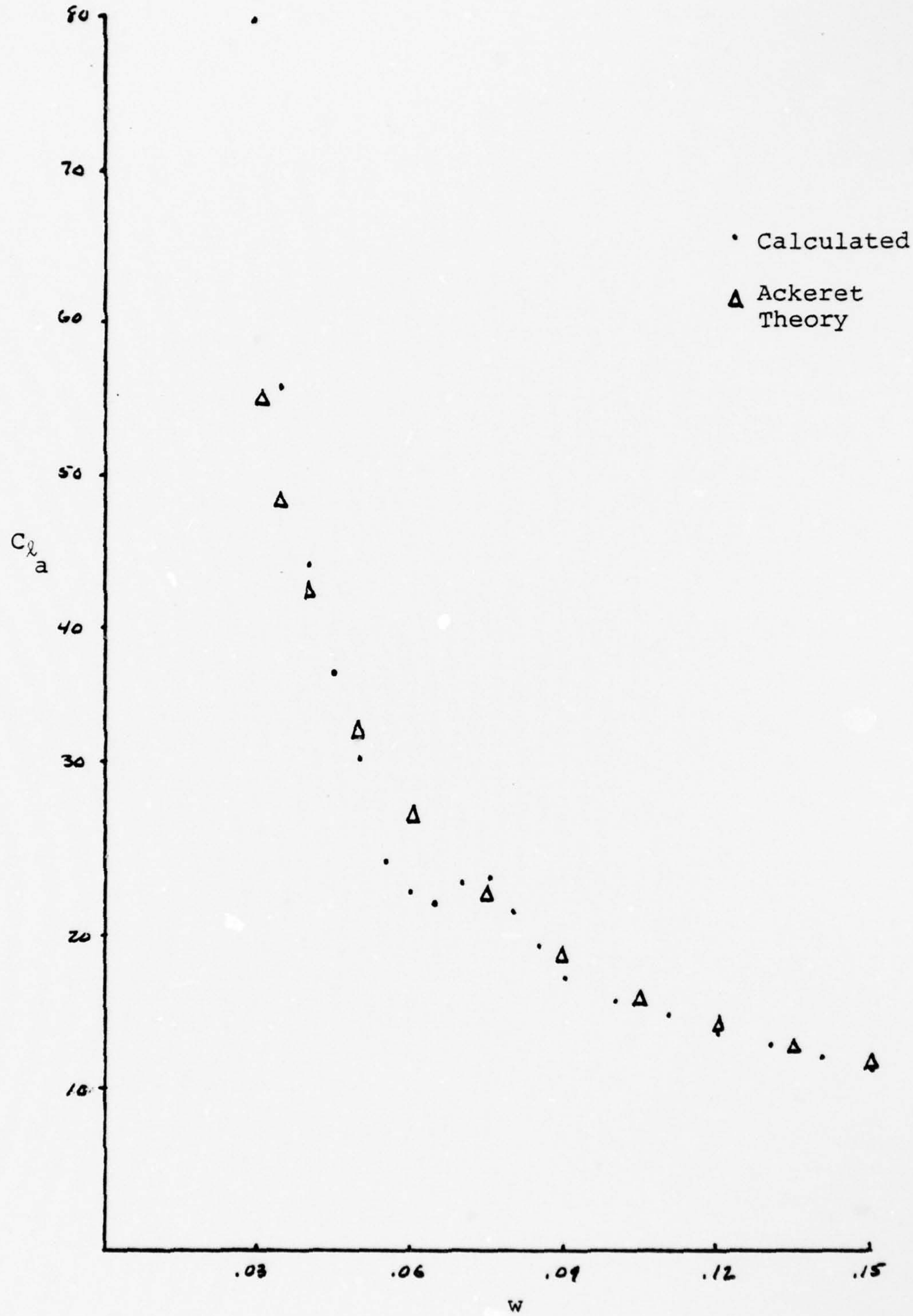


FIGURE VI-2. Comparison of $C\ell_\alpha$-vs-w to that Obtained from Ackeret Theory

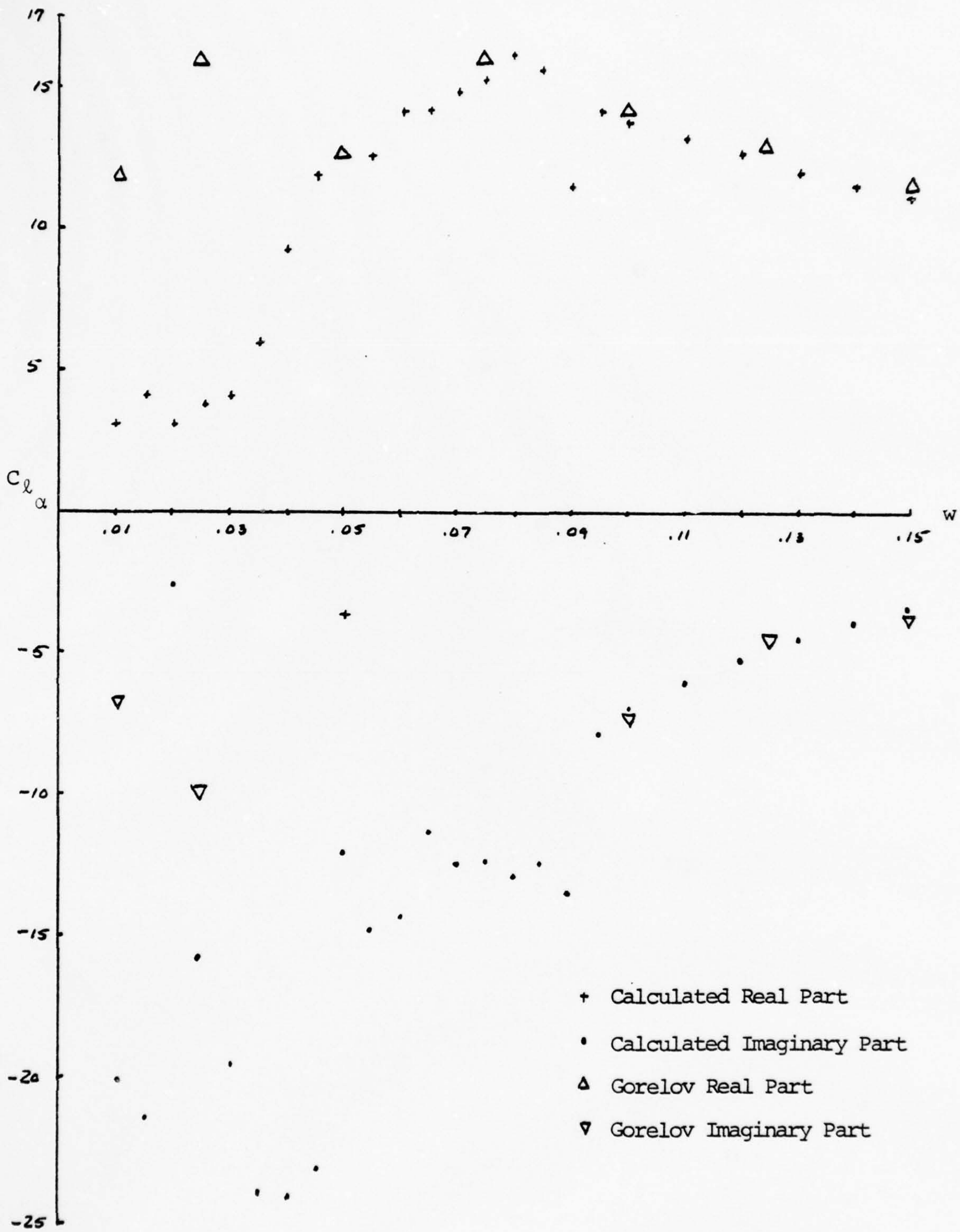


FIGURE VI-3. Plot of $C_{\ell\alpha}$-vs-w, Legendre Polynomials
$k = 0.1$, $\tau = 1.0$, $\sigma = \pi$, $n = 7$
compared with Gorelov's results

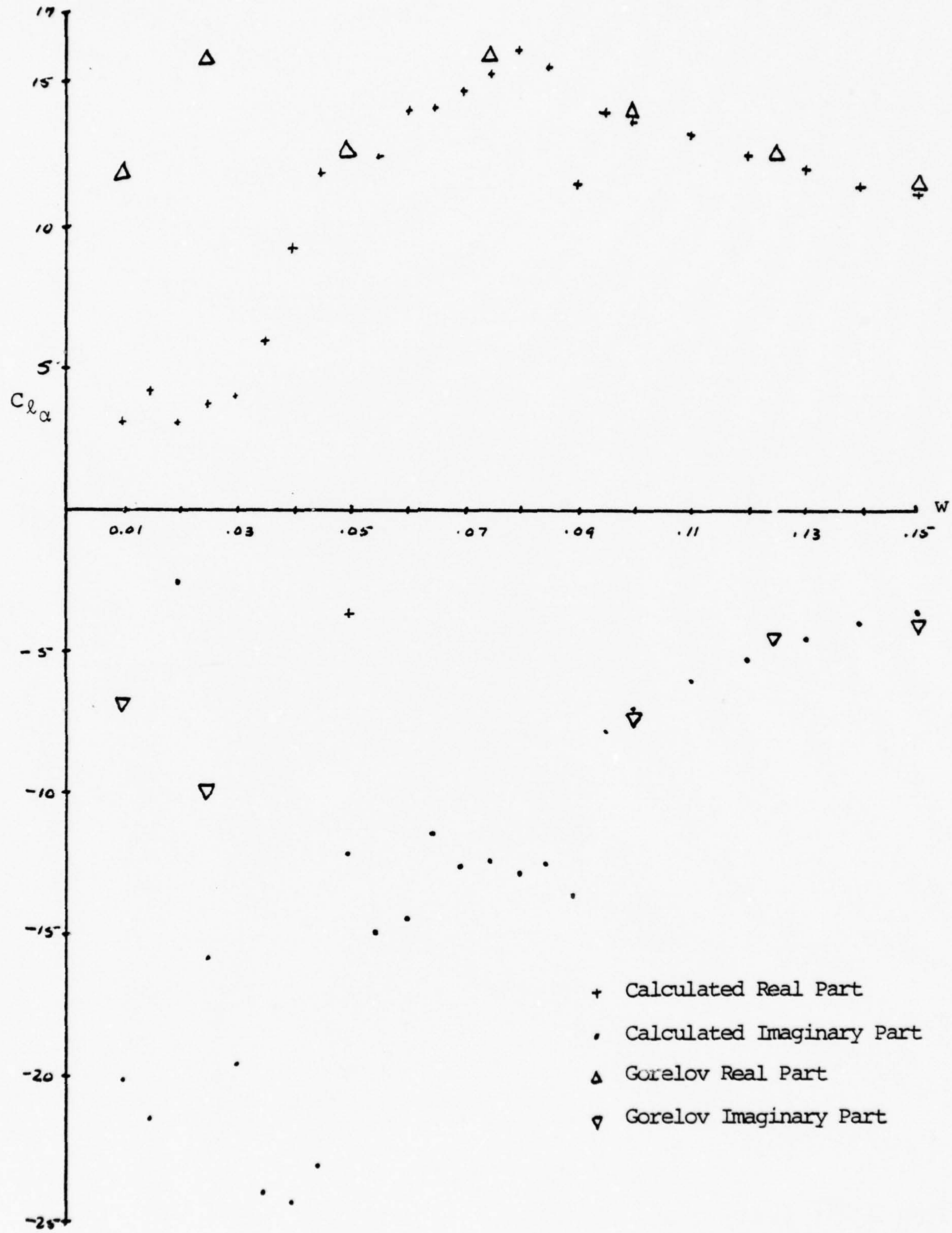


FIGURE VI-4. Plot of C_{ℓ_α}-vs-w, Gorelov Spanning Function $k = 0.1$, $\tau = 1.0$, $\sigma = \pi$, $n = 7$ compared with Gorelov's results

VII . RECOMMENDATIONS

There are two recommendations to be made about the techniques used in the collocation method, and a new area in which it might be employed.

The program developed in the course of writing this thesis employs adaptive Simpson's integration to calculate the elements of a completely determined system. This system is solved to provide the coefficients of the spanning functions. Two improvements may be made:

1. The Simpson's integration scheme may be replaced by a Gaussian integrator. Experience has shown that several thousand function evaluations are required by the Simpson's integration routine when C_{ℓ_α} is to be evaluated for small w. This entails large amounts of computer time and leads to increased accumulations of numerical error. Use of Gaussian integration would probably improve both of these characteristics with little loss of accuracy.

2. The present program treats a completely determined system of dimension 2n+1 by 2n+1, and then solves that system to find the collocation coefficients. This procedure has worked satisfactorily in this thesis, but may not work as well at higher frequencies where the final linear system of equations may be ill-conditioned. As an alternative, it is recommended that the boundary conditions be applied at more than n points, say m points, where m is twice or three times as many points, and that the least squares technique be used to determine the

the collocation coefficients which give the minimum square error over-all. This may be thought of as "sampling more data" in order to get more information about the unknown function. The present program could be easily modified in this regard by replacing the spanning function matrix, Q1ZINT, by a new matrix of the form

$$Q1ZINT' = X^T X$$

where X is the new m by n+1 (m > n+1) matrix, and replacing the present right-hand-side vector, Q1COF with

$$Q1COF' = X^T Y$$

where Y is the new m by 1 right-hand-side vector. An alternative would be to employ a prepackaged statistical linear regression routine after either modifying the routine to accept complex data, or transforming the present system into a larger system of real numbers only.

The new area in which the collocation method might be employed is the calculation of the potential flow about a staggered cascade. The method coud be employed to calculate both the potential in the channel and above the upper blade. The program presented has been designed to enable the calculation of flow within the channel of a staggered cascade. Unfortunately, there was not enough time to extend the study to this case.

APPENDIX A

PROGRAM DESCRIPTION

This section describes the computer program used to calculate the interference solution to the Gorelov linearization for unsteady transonic flow in a channel. The program written in IBM Fortran IV with the basic structure outlined by Stevens [5]. The basic points are:

1. Organization of the program into small subroutines, each of which performs a specific task.
2. Transmission data to and from subroutines via a formal parameter argument list. No common statements are used.

The end objective is code which is both easy to modify and maintain.

Each subroutine is designed with optional diagnostic printing of its input and output. This is controlled by the parameter IPT. The diagnostic output is printed (only) if IPT > 0. Each routine accepts IPT, sets IOT = IPT - 1, and then passes IOT as the print parameter to routines it calls. By this method, diagnostic output can be "cascaded" to any desired level. Large initial values of IPT should be avoided because of the spectacular amount of output which can be generated by the double integrals within Q1DCOF.

1. Main Program; including subroutines READ and ABSA.

   The basic structure is given in Figure A-1. MAIN calls READ to read input data and then ABSA to calculate the

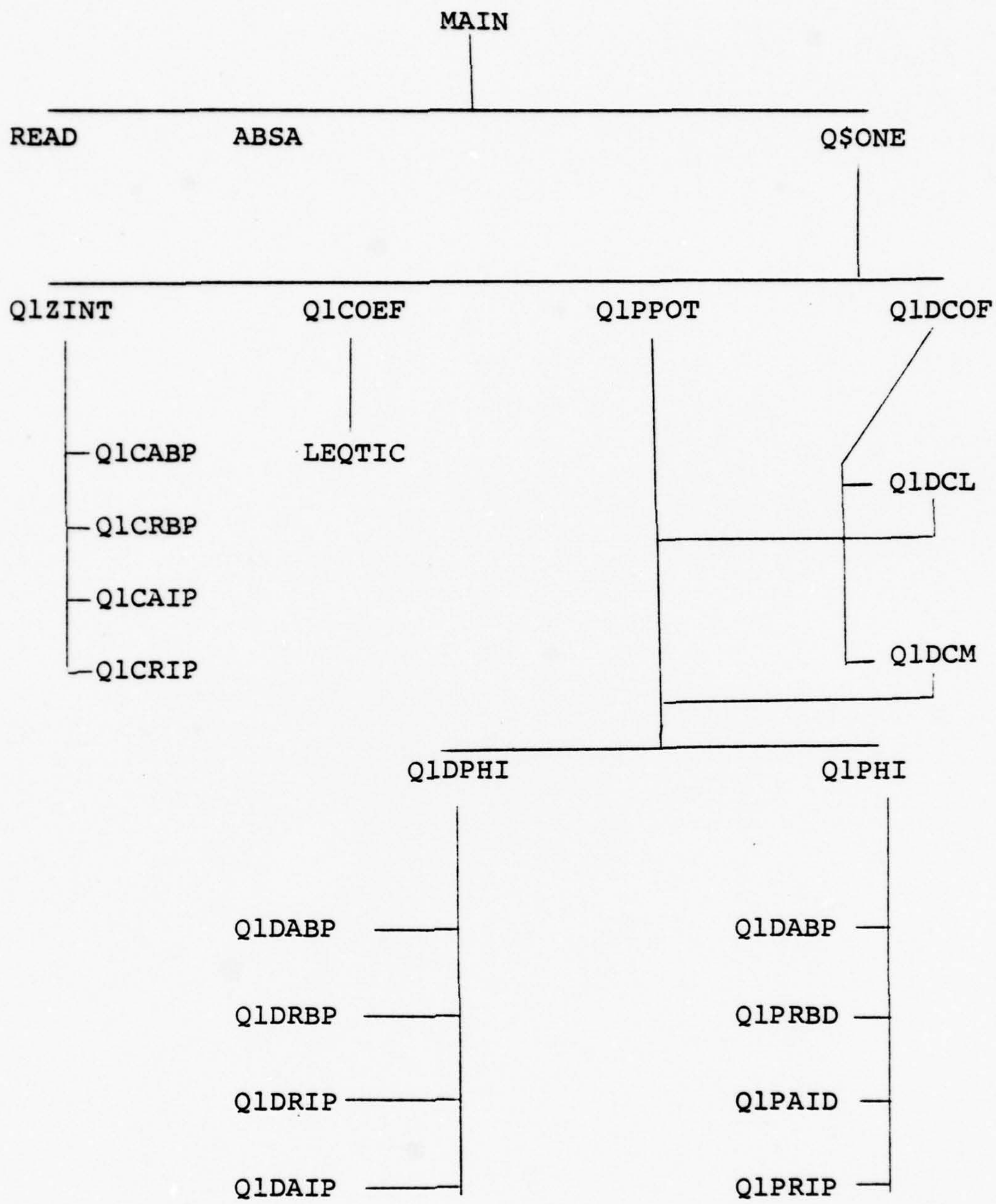


Figure A-1. Program Hierarchy

collocation points. The version of ABSA shown evenly spaces the collocation points across that portion of the blade subject to interference. ABSA may be easily replaced if different point spacing is desired, or if additional points are to be added for an overdetermined system and least squares approximation.

2. Q$ONE This subroutine controls the actual potential calculation. It performs no calculation itself, but calls subordinate subroutines where the calculations are actually performed. The calling hierarchy is shown in Figure A-1.

3. Q1ZINT This subroutine calculates the linear equation system arising from the boundary conditions. Hierarchy is shown in Figure A-2.

The matrix output is carried through Q1INT. Q1ZINT calls the following subprograms

a. Q1CRBP returns the value of ϕ_z^o

b. Q1CABP returns the value of $\phi_{z_1}^1$

c. Q1CRIP returns the values of ψ_z^o

$$\frac{\partial}{\partial z} \int_{-1+x_*}^{x-z} f_j(s) J_o[\omega\sqrt{(x-s)^2-z^2}]ds$$

where f_i is one of the set of i elementary functions, j=1,n

d. Q1CAIP returns the values of $\psi_{z_1}^1$

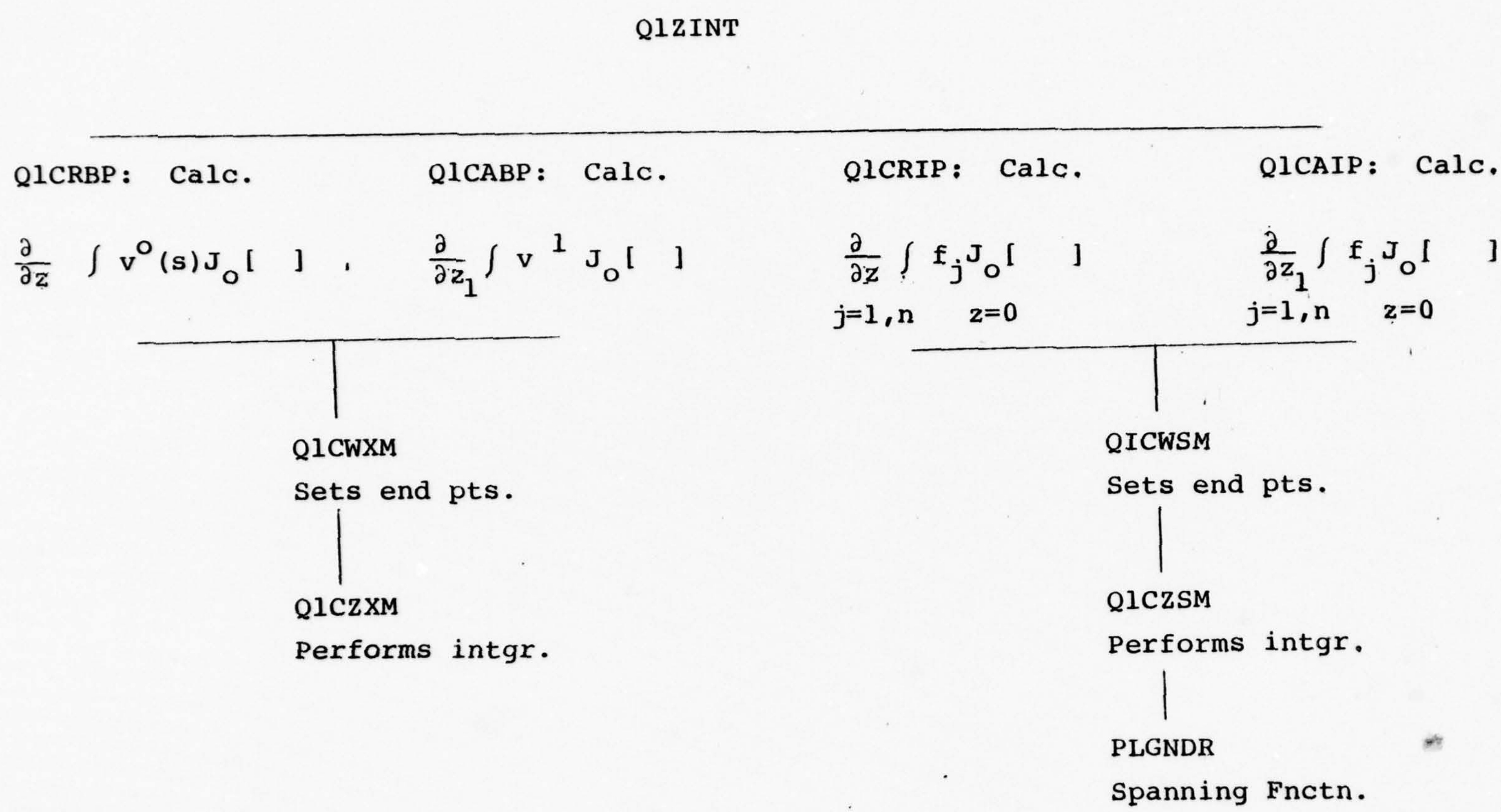

Q1ZINT
Q1CRBP: Calc.
$\frac{\partial}{\partial z} \int v^{o}(s) J_{o}[\ ]$,
Q1CABP: Calc.
$\frac{\partial}{\partial z_1} \int v^{1} J_{o}[\ ]$
Q1CRIP: Calc.
$\frac{\partial}{\partial z} \int f_j J_o[\ ]$
j=1,n z=0
Q1CAIP: Calc.
$\frac{\partial}{\partial z_1} \int f_j J_o[\ ]$
j=1,n z=0
Q1CWXM
Sets end pts.
Q1CZXM
Performs intgr.
QICWSM
Sets end pts.
Q1CZSM
Performs intgr.
PLGNDR
Spanning Fnctn.

Figure A-2

$$\frac{\partial}{\partial z_1} \int_{-1+\text{OFFSET}+x_*}^{x+z_1} f_j(s) J_o[\omega\sqrt{(x-s)^2-y_1^2}]ds$$

where f_j is one of the set of elementary functions. OFFSET is a parameter included to facilitate program conversion to a staggered cascade

e. PLGNDR is the subprogram which returns $f_j(x)$, the elementary spanning function. No other routine contains explicit reference to the spanning function. This facilitates easy replacement of the spanning functions should this be desired.

Q1CRBP and Q1CABP in turn call Q1CWXM and Q1CZXM. Q1CWXM computes end-points and then calls Q1CZXM, a complex integration routine based on SIMP by Shampine and Allen [6]. Q1CAIP and Q1CRIP call Q1CWSM and Q1CZSM to perform the integration. Q1ZINT passes the constant matrix to Q1COEF in the array Q1INT and the right-hand-side vector in the array Q1COF.

4. Q1COEF This subroutine employs the IMSL routine LEQT1C to solve the linear system received from Q1ZINT. The resulting coefficients are Q1ABCF for the adjacent blade and Q1RBCF for the reference blade. LEQT2C, the high precision complex IMSL routine may be directly substituted for LEQT1C. Q1COEF may be rewritten to employ the generalized inverse required for least squares approximation

$$A = (x^T x)^{-1} x^T y \quad \text{where} \quad x = \text{Q1INT}$$

$$y = \text{Q1COF}$$

after first performing the multiplication necessary to replace Q1INT and Q1COF with the proper matrix products in the call to LEQT1C.

5. Q1PPOT This subroutine calculates the potential, ϕ, and ϕ_x, at each collocation point along the reference blade but only if Q1PPOT receives a value of IPT > 0, requiring IPT $\geq$ 2 on input to the main program. If IPT $\leq$ 0, then the subroutine is exited before any calculations are performed. This subroutine is most useful for debugging Q1ZINT and Q1COEF. Q1PPOT calls Q1PABP, Q1PRBP, Q1PRIP, Q1PAIP, Q1DABP, Q1DRBP, Q1DRIP, and Q1DAIP, all of which will be described in the next section.

6. Q1DCOF This subroutine calculates the dimensionless coefficients of lift and moment; C_{ℓ_α}, C_{m_α}. Its internal hierarchy is shown in Figure A-3.

   a. Q1DCL calculates the nondimensional complex coefficient C_{ℓ_α}

   b. Q1DCM calculates the nondimensional complex coefficient C_{m_α}.

   c. Q1PRBP and Q1PABP calculate the potentials due to the reference and adjacent blades respectively. Q1PWXM and Q1PZXM are called to perform the actual integration.

d. Q1PRIP and Q1PAIP calculate the interference potentials along the reference and adjacent blades. Q1PZSM is called to perform the integration.

e. Q1DRBP, Q1DABP, Q1DRIP, and Q1DAIP correspond exactly to subroutines above except that the value returned is the partial derivative of the potential with respect to X. Q1DWXM, Q1DZSM, and Q1DXSM perform the co-reponding integrals.

6. Program Listing The program listing shown below incorporates the Legendre functions as spanning functions. Listings for a subroutine employing Gorelov's spanning function and the linear approximation program follow.

```
      IMPLICIT REAL*8(A-H,O,P,R-Y), COMPLEX*16(Q,Z)                     QLDR0005
      DIMENSION X(13), Q2PT(13), Q2CP(13)                               QLDR0010
C     BLOCK ONE    READ AND EDIT DATA                                   QLDR0015
      CALL ERRSET(208,256,-1,1)                                         QLDR0020
C     WRITE (6,910)                                                     QLDR0025
    1 CALL READ(DK,DR,DW,RHO,OFFSET,SIGMA,N,NF,IPT)                     QLDR0030
C     IF(IPT.GT.0)WRITE(6,990)DK,DR,DW,SIGMA,OFFSET,RHO,N,NF            QLDR0035
  910 FORMAT ('1',5X,'GORELOV CASCADE PROGRAM')                         QLDR0040
  990 FORMAT('0',5X,'INPUT VALUES TRANSMITTED TO MAIN PROGRAM:',/,      QLDR0045
     1' ',10X,'DK',13X,'DR',13X,'DW    ',10X,'SIGMA',10X,'OFFSET',/,    QLDR0050
     1' ',10X,5(E12.5,3X),/,                                            QLDR0055
     2'0',10X,'RHO',12X,'N',5X,'NFCN'                                   QLDR0060
     3,/,' ',10X,1(E12.5,3X), I2,4X,I2)                                 QLDR0065
      CALL ABSA(N,OFFSET,X,RHO,DW    )                                  QLDR0070
C                                                                       QLDR0075
C                                                                       QLDR0080
C     BLOCK TWO     CALLS CALCULATION ROUTINES FOR ZONE ONE AND ZONE TWO QLDR0085
      CALL Q$ONE(DK,DR,DW,RHO,OFFSET,SIGMA,N,NF,IPT,X)                  QLDR0090
      GO TO 1                                                           QLDR0095
      END                                                               QLDR0100
      SUBROUTINE READ (DK,DR,DW,RHO,OFFSET,SIGMA,N,NF,IPT)              QLDR0105
      IMPLICIT REAL*8(A-H,O,P,R-Y)                                      QLDR0110
      READ(5,905) DK,DR,DW,RHP,OFFSET,SIGMA,N,NF,IPT                    QLDR0115
      IF (DK.LT.0.0D0) STOP                                             QLDR0120
      IF (IPT.GT.0) WRITE (6,910)                                       QLDR0125
      GAMMA = 1.4D0                                                     QLDR0130
      DR = RHP * DSQRT((GAMMA + 1.0D0) * DW)                            QLDR0135
      RHO = DR                                                          QLDR0140
      IF(N.LE.13) GO TO  1                                              QLDR0145
      N = 13                                                            QLDR0150
C     WRITE(6,935 )                                                     QLDR0155
    1 IF(OFFSET.LE.1.9D0) GO TO 2                                       QLDR0160
      WRITE(6,940)                                                      QLDR0165
      OFFSET = OFFSET-1.0D0                                             QLDR0170
      GO TO 1                                                           QLDR0175
    2 CONTINUE                                                          QLDR0180
C     IF (IPT.GT.0) WRITE (6,925) DK,DR,DW,SIGMA,OFFSET,RHO,N,NF,IPT    QLDR0185
      RETURN                                                            QLDR0190
  905 FORMAT(6F10.4,3I2)                                                QLDR0195
  910 FORMAT ('1',10X,'STAGGERED SUPERSONIC CASCADE PROGRAM')           QLDR0200
  925 FORMAT ('0',10X,'DK',13X,'DR',13X,'DW    ',10X,'SIGMA',10X,'OFFSET'QLDR0205
     1,/,' ',10X,5(E12.5,3X),/,                                         QLDR0210
     2'0',10X,'RHO',12X,'N',5X,'NFCN',2X,'IPT'                          QLDR0215
     3,/,' ',10X,1(E12.5,3X), I2,4X,I2,4X,I2)                           QLDR0220
  935 FORMAT('0',5X,'ORIGINAL N TOO LARGE (.GT.13) - RESET TO N = 13')  QLDR0225
  940 FORMAT('0',5X,'OFFSET TOO LARGE (.GT.1.9), RESET AS OFFSET =      QLDR0230
     1OFFSET - 1.0D0')                                                  QLDR0235
      END                                                               QLDR0240
```

```
      SUBROUTINE ABSA (N,OFFSET,X,DR,DW    )
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX*16(Q,Z)
      DIMENSION X(13)
      XINT = (2.0D0-DR)/DFLOAT(N+1)
      XL = DR - 1.0D0 + 1.0D-8
      DO 10 I = 1,N
      X(I) = XL + (XINT*DFLOAT(I))
      IF(X(I).EQ.0.0D0) X(I) = .1D-14
   10 CONTINUE
      RETURN
      END
      SUBROUTINE Q$ONE(DK,DR,DW,RHO,OFFSET,SIGMA,N,NF,IPT,X)
      IMPLICIT REAL*8(A-H,O,P,R-Y), COMPLEX*16(Z,Q)
      DIMENSION Q1COF(26), Q1INT(26,26), Q1RBP(13), Q1ABP(13)
      DIMENSION Q1RBCF(13), Q1ABCF(13)
      DIMENSION X(13)
C     IF(IPT.GT.0)WRITE(6,990)DK,DR,RHO,OFFSET,SIGMA,N,NF,IPT
      ICT = IPT - 1
  990 FORMAT('0  SUBROUTINE Q$ONE ENTERED WITH:',/,
     1' ',5X,'DK',13X,'DR',13X,'RHO',12X,'OFFSET',9X,'SIGMA',10X,
     2'N',3X,'NF',2X,'IPT',/,
     3  5X,5(E12.5,3X),I2,2X,I2,2X,I3)
      CALL Q1ZINT(DK,DR,DW,RHO,OFFSET,SIGMA,N,X,ICT,Q1COF,Q1INT)
      CALL Q1COEF(Q1COF,Q1INT,N,IOT,Q1ABCF,Q1RBCF)
      CALL Q1PRCT(DK,DR,DW,RHO,OFFSET,SIGMA,N,X,Q1ABCF,Q1RBCF,ICT)
      I C T = 0
      CALL Q1DCCF(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,ICT)
      RETURN
      END
C
C
C
      SUBROUTINE Q1ZINT(DK,DR,DW,RHO,OFFSET,SIGMA,N,X,
     1      IPT,Q1COF,Q1INT)
      IMPLICIT REAL * 8(A-H,O,P,R-Y), COMPLEX * 16(Z,Q)
      DIMENSION Q1COF(26), Q1INT(26,26), QINTRP(13), QINTAP(13)
      DIMENSION  X(13)
  990 FORMAT ('0',10X,'Q1ZINT ENTERED WITH:',/,
     1 ' ',10X,'DK',13X,'DR',13X,'RHO',12X,'OFFSET',9X, 'SIGMA',10X,
     2  'N',2X,'IPT',/,
     3  ' ',10X,5(E12.5,3X),I2,2X,I3)
      CCONST = CDEXP(DCMPLX(0.0D0,SIGMA))
      IF (IPT.GT.0) WRITE (6,990) DK,DR,RHO,OFFSET,SIGMA,N,IPT
      ICT = IPT - 1
      GAMMA = 1.4D0
      CLAMDA = DK/((GAMMA+1.0D0)*DW)
      DO 90 I = 1,N
      IN = I + N
```

```
      XSTN = X(I)
      XO = XSTN - DR
      CEXP = CDEXP(DCMPLX(0.0D0,DLAMDA*XSTN))
      CALL Q1CAIP(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,QINTAP,N,IOT)
      CALL Q1CRIP(DK,DR,DW,RHO,OFFSET,XSTN,QINTRP,N,ICT)
      DO 20 J = 1,N
       JN = N + J
      J1 = J-1
      Q1INT(I,J) = PLGNDR(XSTN,DR,J1) * CEXP
      Q1INT(IN,J) = QINTRP(J)
      Q1INT(I,JN) =  QINTAP(J)
      Q1INT(IN,JN) = PLGNDR(XSTN-OFFSET,DR,J1) * CEXP
   20 CONTINUE
      Q1COF(IN) = -Q1CRBP(DK,DR,DW,RHO,XSTN,IOT)
      Q1COF(I)  = -Q1CABP(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IOT)
   90 CONTINUE
      RETURN
      END
      COMPLEX FUNCTION Q1CRBP*16(DK,DR,DW,RHO,XSTN,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION QINP(2)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,XSTN,IPT
  990 FORMAT('0',10X,'Q1CRBP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'XSTN',9X,'IPT'
     2  ,/,' ',10X,5(E12.5,','),I3)
      IF(XSTN.LE.RHO-1.0D0) GOTO 20
      IF(XSTN.GT.2.0D0) GOTO 20
      ICT = IPT - 1
      QDK = DCMPLX (0.0D0,DK)
      CALL Q1CWXM(DK,DR,DW,RHO,XSTN,QINP,IOT)
      Q1CRBP = -QDK*QINP(2) - QINP(1)
      IF(IPT.LE.0) RETURN
      GOTO 30
   20 Q1CRBP = DCMPLX(0.0D0,0.0D0)
      IF(IPT.LE.0) RETURN
   30 WRITE (6,995) Q1CRBP
  995 FORMAT('0',10X,'Q1CRBP = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1CABP*16(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION QINP(2)
      IF(IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT
  990 FORMAT('0',10X,'Q1CABP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFFSET',7X,
     2  'SIGMA',8X,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)
      XASTN = XSTN - OFFSET
      IF(XASTN.LE.RHO-1.0D0) GOTO 20
```

```
      IF(XASTN.GT.2.0D0) GOTO 20
      ICT = IPT - 1
      QDK = DCMPLX (0.0D0,DK)
      QCONST = CDEXP( DCMPLX(0.0D0,SIGMA))
      CALL Q1CWXM(DK,DR,DW,RHO,XASTN,QINP,IOT)
      Q1CABP = -(QDK*QINP(2) + QINP(1)) * QCONST
      IF(IPT.LE.0) RETURN
      GO TO 30
   20 Q1CABP = DCMPLX(0.0D0,0.0D0)
      IF (IPT.LE.0) RETURN
   30 WRITE (6,995) Q1CABP
  995 FORMAT('0',10X,'Q1CABP = ',E14.7,',',E14.7)
      RETURN
      END
      SUBROUTINE Q1CWXM(DK,DR,DW,RHO,XSTN,QINP,IPT)
C
C     N IS THE MAXIMUM DEGREE OF THE TERM U IN THE INTEGRALS
C
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Z,Q)
      DIMENSION QINP(2)
      IF(IPT.GT.0) WRITE(6,990) DK,DR,RHO,XSTN,IPT
  990 FORMAT('0',10X,'Q1CWXM ENTERED WITH ARGUMENTS:',/,
     1' ',10X,'DK',16X,'DR',16X,'RHO',15X,'XSTN',14X,'IPT',/,
     2' ',10X,4(E13.6,5X),I2,5X,I3)
      ICT = IPT - 1
      B = XSTN - RHO - 1.0D-10
      A = -1.0D0
      CALL Q1CZXM(DK,DR,DW,RHO,XSTN,A,B,1,QANS,IOT)
      QINP(1) = QANS
      CALL Q1CZXM(DK,DR,DW,RHO,XSTN,A,B,2,QANS,IOT)
      QINP(2) = QANS
      IF(IPT.LE.0) RETURN
      WRITE (6,995)
  995 FORMAT(' ',10X,'Q1CWXM RESULTS:',/,
     1' ',11X,'J',6X,'QINP(I)')
      DO 1001 I = 1,2
  996 FORMAT(' ',10X,I3,5X,E12.6,',',E12.6)
 1001 WRITE (6,996) I, QINP(I)
      RETURN
      END
      SUBROUTINE Q1CZXM(DK,DR,DW,RHO,XSTN,A,B,J,QANS,IPT)
      IMPLICIT REAL*8(A - E,G,H,M,O,P,R - Y), COMPLEX*16(F,Q,Z)
      DIMENSION FV(5),LORR(30),F1T(30),F2T(30),F3T(30),CAT(30),
     1  ARESTT(30),QESTT(30),EPST(30),QPSUM(30)
      F(X) =  (X**J1)*CDEXP(QEXP*(X))
     1      *(OMEGA*RHO/(DSQRT((XSTN-X)*(XSTN-X)-YY))*
     2      MMBSJ1(OMEGA*DSQRT((XSTN-X)*(XSTN-X)-YY),IER))
      GAMMA = 1.4D0
```

```
      DM2 = (GAMMA + 1.0D0) * DW                                        QLDR0965
      OMEGA = DSQRT(DK*DK*(1.0D0-DM2)/(DM2*DM2))                         QLDR0970
      YY = RHO * RHO                                                    QLDR0975
      CLAMDA = DK/DM2                                                   QLDR0980
      CEXP = DCMPLX(0.0D0,CLAMDA)                                       QLDR0985
      J1 = J-1                                                          QLDR0990
      ACC = 1.0D-6                                                      QLDR0995
      U  =  9.0D-13                                                     QLDR1000
      IF (IPT.GT.0) WRITE(6,990)DK,DR,DW,RHO,XSTN,A,B,J,IPT             QLDR1005
  990 FORMAT('0',15X,'Q1CZXM ENTERED WITH ARGUMENTS:',/,                QLDR1010
     1' ',15X,'DK',13X,'DR',13X,'DW    ',10X,'RHO',12X,'XSTN',11X,      QLDR1015
     2'A',14X,'B',14X,'J',2X,'IPT',/,                                   QLDR1020
     3' ',15X,7(E14.7,' '),I2,2X,I3)                                    QLDR1025
      EFOURU = 4.0*U                                                    QLDR1030
      IFLAG = 1                                                         QLDR1035
      EPS = ACC                                                         QLDR1040
      CERROR = DCMPLX(0.0D0,0.0D0)                                      QLDR1045
      LVL = 1                                                           QLDR1050
      LERR(LVL) = 1                                                     QLDR1055
      QPSUM(LVL) = 0.0                                                  QLDR1060
      ALPHA = A                                                         QLDR1065
      DA = B - A                                                        QLDR1070
      AREA = 0.0                                                        QLDR1075
      AREST = 0.0                                                       QLDR1080
      FV(1) = F(ALPHA)                                                  QLDR1085
      FV(3) = F(ALPHA + 0.5*DA)                                         QLDR1090
      FV(5) = F(ALPHA + DA)                                             QLDR1095
      KOUNT = 3                                                         QLDR1100
      WT = DA/6.0                                                       QLDR1105
      QEST = WT*(FV(1) + 4.0*FV(3) + FV(5))                             QLDR1110
    1 DX = 0.5*DA                                                       QLDR1115
      FV(2) = F(ALPHA + 0.5*DX)                                         QLDR1120
      FV(4) = F(ALPHA + 1.5*DX)                                         QLDR1125
      KOUNT = KOUNT + 2                                                 QLDR1130
      WT = DX/6.0                                                       QLDR1135
      QESTL = WT*(FV(1) + 4.0*FV(2) + FV(3))                            QLDR1140
      QESTR = WT*(FV(3) + 4.0*FV(4) + FV(5))                            QLDR1145
      QSUM = QESTL + QESTR                                              QLDR1150
      ARESTL = WT*(CDABS(FV(1)) + CDABS(4.0*FV(2)) + CDABS(FV(3)))      QLDR1155
      ARESTR = WT*(CDABS(FV(3)) + CDABS(4.0*FV(4)) + CDABS(FV(5)))      QLDR1160
      AREA = AREA + ((ARESTL + ARESTR) - AREST)                         QLDR1165
      QDIFF = QEST - QSUM                                               QLDR1170
      IF(CDABS(QDIFF).LE.EPS*DABS(AREA))GO TO 2                         QLDR1175
      IF(DABS(DX).LE.EFOURU*DABS(ALPHA))GO TO 5                         QLDR1180
      IF(LVL.GE.30)GO TO 5                                              QLDR1185
      IF(KOUNT.GE.2000)GO TO 6                                          QLDR1190
      LVL = LVL + 1                                                     QLDR1195
      LERR(LVL) = 0                                                     QLDR1200
```

```
      F1T(LVL) = FV(3)                                                  QLDR1205
      F2T(LVL) = FV(4)                                                  QLDR1210
      F3T(LVL) = FV(5)                                                  QLDR1215
      DA = DX                                                           QLDR1220
      DAT(LVL) = DX                                                     QLDR1225
      AREST = ARESTL                                                    QLDR1230
      ARESTT(LVL) = ARESTR                                              QLDR1235
      QEST = QESTL                                                      QLDR1240
      QESTT(LVL) = QESTR                                                QLDR1245
      EPS = EPS/1.4                                                     QLDR1250
      EPST(LVL) = EPS                                                   QLDR1255
      FV(5) = FV(3)                                                     QLDR1260
      FV(3) = FV(2)                                                     QLDR1265
      GO TO 1                                                           QLDR1270
2     QERROR = QERROR + QDIFF/15.0                                      QLDR1275
3     IF(LORR(LVL).EQ.0)GO TO 4                                         QLDR1280
      QSUM = QPSUM(LVL) + QSUM                                          QLDR1285
      LVL = LVL - 1                                                     QLDR1290
      IF(LVL.GT.1)GO TO 3                                               QLDR1295
      QANS = QSUM - (B**J1) * CDEXP(QEXF*B)                             QLDR1300
      IF(IPT.GT.0) GO TO 11                                             QLDR1305
      IF(IER.EQ.129) GO TO 11                                           QLDR1310
      IF(IFLAG.EQ.1) RETURN                                             QLDR1315
      WRITE(6,990) DK,DR,DW,RHO,XSTN,A,B,J,IPT                          QLDR1320
11    WRITE(6,995) QANS,IFLAG,IER,QERROR                                QLDR1325
995   FORMAT(' ',15X,'RESULTS: QANS = ',E14.7,',',E14.7,/,             QLDR1330
     1' ',15X,'IFLAG',2X,'IER',5X,'QERROR',/,                           QLDR1335
     2' ',15X,I3,2X,I3,5X,E14.7,',',E14.7)                              QLDR1340
      RETURN                                                            QLDR1345
4     QPSUM(LVL) = QSUM                                                 QLDR1350
      LORR(LVL) = 1                                                     QLDR1355
      ALPHA = ALPHA + DA                                                QLDR1360
      DA = DAT(LVL)                                                     QLDR1365
      FV(1) = F1T(LVL)                                                  QLDR1370
      FV(3) = F2T(LVL)                                                  QLDR1375
      FV(5) = F3T(LVL)                                                  QLDR1380
      AREST = ARESTT(LVL)                                               QLDR1385
      QEST = QESTT(LVL)                                                 QLDR1390
      EPS = EPST(LVL)                                                   QLDR1395
      GO TO 1                                                           QLDR1400
5     IFLAG = 2                                                         QLDR1405
      GO TO 2                                                           QLDR1410
6     IFLAG = 3                                                         QLDR1415
      GO TO 2                                                           QLDR1420
      END                                                               QLDR1425
C                                                                       QLDR1430
      SUBROUTINE Q1CAIP(DK,DR,DW,RHO,QFST,SGMA,XSTN,QCAIP,N,IPT)        QLDR1435
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)                 QLDR1440
```

```
      DIMENSION  CINP(13), QCAIP(13)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFST,SGMA,XSTN,IPT
  990 FORMAT('0',10X,'Q1CAIP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFST',9X,'SGMA',9X
     2  ,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)
      XASTN = XSTN - OFST
      IF(XASTN.LE.DR+RHO-1.0D0) GO TO 20
      IF(XASTN.GT.2.0D0) GOTO 20
      ICT = IPT - 1
      QCONST = CDEXP( DCMPLX(0.0D0,SGMA))
      CALL Q1CWSM(DK,DR,DW,RHO,XASTN,N,CINP,IOT)
      DO 10 I = 1,N
      QCAIP(I) = CINP(I)
   10 CONTINUE
      IF(IPT.LE.0) RETURN
      GOTO 30
   20 ZERO = DCMPLX(0.0D0,0.0D0)
      DO 25 I = 1,N
      QCAIP(I) = ZERO
   25 CONTINUE
      IF(IPT.LE.0)RETURN
   30 WRITE (6,995)
  995 FORMAT ('0',10X,'Q1CAIP RESULTS: J      QCAIP(J)')
      DO 40 J = 1,N
      WRITE (6,996) J,QCAIP(J)
  996 FORMAT(' ',26X,I2,3X,E14.7,',',E14.7)
   40 CONTINUE
      RETURN
      END
      SUBROUTINE  Q1CRIP (DK,DR,DW,RHO,OFFSET,XSTN,QCRIP,N,IPT)
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX * 16 (C,Z)
      DIMENSION  QCRIP(13)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,XSTN,IPT
  990 FORMAT('0',10X,'Q1CRIP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,
     2  'XSTN',9X,'IPT'/,' ',10X,5(E12.5,','),I3)
      IF(XSTN.LE.DR+RHO+OFFSET-1.0D0) GOTO 20
      IF(XSTN.GT.2.0D0) GOTO 20
      ICT = IPT - 1
      CALL Q1CWSM(DK,DR,DW,RHO,XSTN,N,QCRIP,IOT)
      IF(IPT.LE.0) RETURN
      GOTO 30
   20 ZERO = DCMPLX(0.0D0,0.0D0)
      DO 25 I = 1,N
   25 QCRIP(I) = ZERO
      IF (IPT.LE.0) RETURN
   30 WRITE (6,995)
  995 FORMAT ('0',10X,'Q1CRIP RESULTS: J      QCRIP(J)')
```

```
      DO 40 J = 1,N
      WRITE (6,996) J,QCRIP(J)
  996 FORMAT(' ',26X,I2,3X,E14.7,',',E14.7)
   40 CONTINUE
      RETURN
      END
      SUBROUTINE Q1CWSM(DK,DR,DW,RHO, XSTN,N,QINF,IPT)
C
C     N IS THE MAXIMUM DEGREE OF THE TERM U IN THE INTEGRALS
C
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Z,Q)
      DIMENSION QINP(13)
      IF(IPT.GT.0) WRITE(6,990) DK,DR,RHO,XSTN,N,IPT
  990 FORMAT('0',10X,'Q1CWSM ENTERED WITH ARGUMENTS:',/,
     1' ',10X,'DK',16X,'DR',16X,'RHO',15X,'XSTN',14X,'N',6X,'IPT',/,
     2' ',10X,4(E13.6,5X),I2,5X,I3)
      IOT = IPT - 1
      B = XSTN - RHO - 1.0D-10
      A = DR-1.0D0
      DO 30 J = 1,N
      CALL Q1CZSM(DK,DR,DW,RHO,XSTN,A,B,J,QANS,IOT)
      QINP(J) = QANS
   30 CONTINUE
      IF(IPT.LE.0) RETURN
      WRITE (6,995)
  995 FORMAT(' ',10X,'Q1CWSM RESULTS:',/,
     1' ',11X,'J',6X,'QINP(I)')
      DO 1001 I = 1,N
  996 FORMAT(' ',10X,I3,5X,E12.6,',',E12.6)
 1001 WRITE (6,996) I, QINP(I)
      RETURN
      END
      SUBROUTINE Q1CZSM(DK,DR,DW,RHO,XSTN,A,B,J,QANS,IPT)
      IMPLICIT REAL*8(A - E,G,H,M,O,P,R - Y), COMPLEX*16(F,Q,Z)
      DIMENSION FV(5),LCRR(30),F1T(30),F2T(30),F3T(30),DAT(30),
     1  ARESTT(30),QESTT(30),EPST(30),QPSUM(30)
      F(X) = PLGNDR(X,DR,J1)*CDEXP(QEXP*(X))
     1      *(OMEGA*RHO/(DSQRT((XSTN-X)*(XSTN-X)-YY))*
     2      MMBSJ1(OMEGA*DSQRT((XSTN-X)*(XSTN-X)-YY),IER))
      GAMMA = 1.4D0
      DM2 = (GAMMA + 1.0D0) * DW
      OMEGA = DSQRT(DK*DK*(1.0D0-DM2)/(DM2*DM2))
      YY = RHO * RHO
      DLAMDA = DK/DM2
      QEXP = DCMPLX(0.0D0,DLAMDA)
      J1 = J-1
      ACC = 1.0D-6
      U  =  9.0D-13
```

```
      IF (IPT.GT.0) WRITE(6,990)DK,DR,DW,RHO,XSTN,A,B,J,IPT
  990 FORMAT('0',15X,'Q1CZSM ENTERED WITH ARGUMENTS:',/,
     1' ',15X,'DK',13X,'DR',13X,'DW      ',10X,'RHO',12X,'XSTN',11X,
     2'A',14X,'B',14X,'J',2X,'IPT',/,
     3' ',15X,7(E14.7,','),I2,2X,I3)
      EFOURU = 4.0*U
      IFLAG = 1
      EPS = ACC
      CERROR = DCMPLX(0.0D0,0.0D0)
      LVL = 1
      LORR(LVL) = 1
      CFSUM(LVL) = 0.0
      ALPHA = A
      DA = B - A
      AREA = 0.0
      AREST = 0.0
      FV(1) = F(ALPHA)
      FV(3) = F(ALPHA + 0.5*DA)
      FV(5) = F(ALPHA + DA)
      KOUNT = 3
      WT = DA/6.0
      QEST = WT*(FV(1) + 4.0*FV(3) + FV(5))
    1 DX = 0.5*DA
      FV(2) = F(ALPHA + 0.5*DX)
      FV(4) = F(ALPHA + 1.5*DX)
      KOUNT = KOUNT + 2
      WT = DX/6.0
      QESTL = WT*(FV(1) + 4.0*FV(2) + FV(3))
      QESTR = WT*(FV(3) + 4.0*FV(4) + FV(5))
      QSUM = QESTL + QESTR
      ARESTL = WT*(CDABS(FV(1)) + CDABS(4.0*FV(2)) + CDABS(FV(3)))
      ARESTR = WT*(CDABS(FV(3)) + CDABS(4.0*FV(4)) + CDABS(FV(5)))
      AREA = AREA + ((ARESTL + ARESTR) - AREST)
      QDIFF = QEST - QSUM
      IF(CDABS(QDIFF).LE.EPS*DABS(AREA))GO TO 2
      IF(DABS(DX).LE.EFOURU*DABS(ALPHA))GO TO 5
      IF(LVL.GE.30)GO TO 5
      IF(KOUNT.GE.2000)GO TO 6
      LVL = LVL + 1
      LORR(LVL) = 0
      F1T(LVL) = FV(3)
      F2T(LVL) = FV(4)
      F3T(LVL) = FV(5)
      DA = DX
      DAT(LVL) = DX
      AREST = ARESTL
      ARESTT(LVL) = ARESTR
      QEST = QESTL
```

```
      QESTT(LVL) = QESTR
      EPS = EPS/1.4
      EPST(LVL) = EPS
      FV(5) = FV(3)
      FV(3) = FV(2)
      GO TO 1
2     QERROR = QERROR + CDIFF/15.0
3     IF(LCRR(LVL).EQ.0)GO TO 4
      QSUM = QPSUM(LVL) + QSUM
      LVL = LVL - 1
      IF(LVL.GT.1)GO TO 3
      QANS = PLGNDR(B,DR,J1) * CDEXP(QEXP*B) - QSUM
      IF(IPT.GT.0) GO TO 11
      IF(IER.EQ.129) GO TO 11
      IF(IFLAG.EQ.1) RETURN
      WRITE(6,990) DK,DR,DW,RHC,XSTN,A,E,J,IPT
 11   WRITE(6,995) QANS,IFLAG,IER,QERRCR
995   FORMAT(' ',15X,'RESULTS: QANS = ',E14.7,',',E14.7,/,
     1' ',15X,'IFLAG',2X,'IER',5X,'CERRCR',/,
     2' ',15X,I3,2X,I3,5X,E14.7,',',E14.7)
      RETURN
4     QPSUM(LVL) = QSUM
      LCRR(LVL) = 1
      ALPHA = ALPHA + DA
      DA = DAT(LVL)
      FV(1) = F1T(LVL)
      FV(3) = F2T(LVL)
      FV(5) = F3T(LVL)
      AREST = ARESTT(LVL)
      QEST = QESTT(LVL)
      EPS = EPST(LVL)
      GO TO 1
5     IFLAG = 2
      GO TO 2
6     IFLAG = 3
      GO TO 2
      END
      REAL FUNCTION PLGNDR*8(X,DR,N)
      IMPLICIT REAL*8(A-H,O-Z)
      IF(N.EQ.0)GOTO 100
      X2 = X*X
      GOTO (101,102,103,104,105,106,107,108,109,110,111,112), N
100   PLGNDR =  1.0D0
      RETURN
101   PLGNDR =  X
      RETURN
102   PLGNDR =  ((3.0D0)*X2-1.0D0)/2.0D0
      RETURN
```

```
103 PLGNDR =   (5.0D0*X2 - 3.0D0)*X/2.0D0
    RETURN
104 PLGNDR =   ((3.5D1*X2 - 3.0D1)*X2 + 3.0D0)/8.0D0
    RETURN
105 PLGNDR =   ((6.3D1*X2 - 7.0D1) *X2 +1.5D1)*X/8.0D0
    RETURN
106 PLGNDR=(((231.0D0*X2-315.0D0)*X2+105.0D0)*X2-5.0D0)/16.0D0
    RETURN
107 PLGNDR=(((429.0D0*X2-693.0D0)*X2+315.0D0)*X2-35.0D0)
   1      *X/16.0D0
    RETURN
108 PLGNDR=((((6435.0D0*X2-12012.0D0)*X2+6930.0D0)*X2
   1      -1260.0D0)*X2+35.0D0)/128.0D0
    RETURN
109 PLGNDR=((((12155.0D0*X2-25740.0D0)*X2+18018.0D0)
   1      *X2-4620.0D0)*X2+315.0D0)/128.0D0
    RETURN
110 PLGNDR=(((((46189.0D0*X2-109395.0D0)*X2+90090.0D0)
   1      *X2-30030.0D0)*X2+3465.0D0)*X2-63.0D0)/256.0D0
    RETURN
111 PLGNDR=(((((88179.0D0*X2-230945.0D0)*X2+218790.0D0)*X2
   1      -90090.0D0)*X2+15015.0D0)*X2-693.0D0)*X2/256.0D0
    RETURN
112 PLGNDR=((((((676039.0D0*X2-1939938.0D0)*X2+2078505)*X2
   1  -1021020.0D0)*X2+225225.0D0)*X2-18018.0D0)*X2-231.0D0)/1024.0D0
    RETURN
    END
    SUBROUTINE Q1COEF( Q1COF,Q1INT,N,IPT,Q1ABCF,Q1RBCF)
    IMPLICIT REAL * 8(A-H,O,P,R-Y), COMPLEX * 16 (Z,C)
    DIMENSION Q1COF(26), Q1INT(26,26), ZWA(300)
    DIMENSION Q1ABCF(13),Q1RBCF(13)
    IF (IPT.GE.0) WRITE (6,90)
    IE = 26
    M = 1
    N2 = 2*N
    IF (IPT.LE.0) GO TO 5
    WRITE (6,98) N,N2
 98 FORMAT('0',5X,'Q1COEF ENTERED WITH ',I2,' DEG  PWR SERIES (',I2,
   1 ' SQUARE MATRIX)')
    DO 2 I = 1,N2
    WRITE(6,92) I, I, Q1COF(I)
 92 FORMAT ('0',10X,'Q1COEF EQUATION SYSTEM, ROW ',I2,/,
   1' ',10X,'Q1COF(',I2,') = ', E14.7,',',E14.7)
    DO 2 J = 1,N
    J2=J+N
    WRITE(6,91) I,J,Q1INT(I,J),I,J2,Q1INT(I,J2)
 91 FORMAT(' ',15X,2('QINT(',I2,',',I2,') = ',E14.7,',',E14.7,10X))
  2 CONTINUE
```

```
    5 IA = 26
      IJOB=0
      CALL LEQT1C(Q1INT,N2,IA,Q1COF,M,IE,IJOB,ZWA,IER)
      IF(IER.EQ.0) GOTO 30
      IF (IER.EQ.129) GO TO 10
      WRITE(6,93)
   93 FORMAT( '0',10X,'Q1COEF - ITERATIVE IMPROVEMENT FAILED, MATRIX TOO
     1 ILL-CONDITIONED. USE RESULTS WITH CAUTION.')
      GO TO 30
   10 WRITE (6,95)
   95 FORMAT('0',10X,'Q1COEF - MATRIX ALGORITHMICALLY SINGULAR.  COEFFIC
     1IENTS SET TO ZERO.')
      ZERO = DCMPLX(0.0D0,0.0D0)
      DO 20 I = 1,N
      I2 = I+N
      Q1COF(I) = ZERO
      Q1COF(I2) = ZERO
   20 CONTINUE
   30 CONTINUE
      DO 35 I = 1,N
      IN = I + N
      Q1ABCF(I) = Q1COF(IN)
      Q1RBCF(I) = Q1COF(I)
   35 CONTINUE
      IF(IPT.LE.0) RETURN
      IF (IPT.GE.0) WRITE (6,94)
      DO 40 I = 1,N
      IM1 = I-1
      WRITE (6,99) I, IM1, Q1RBCF(I), Q1ABCF(I)
   40 CONTINUE
   99 FORMAT('0',5X,I2,10X,I2,10X,2(E14.7,5X,E14.7,10X))
   90 FORMAT( '0',10X,'SUBROUTINE Q1COEF -  COMPLEX POWER SERIES COEFFIC
     1IENTS')
   94 FORMAT( '0',5X,'INDEX',7X,'DEG POLY',4X,'REFERENCE BLADE TERMS Q1C
     3OF(INDEX)',7X,'ADJACENT BLADE TERMS Q1COF(2*INDEX)')
      RETURN
      END
      FUNCTION IQ1IFC(I)
C     ..................................................................
C
C     IQ1IFC IS A FUNCTION SUBROUTINE DESIGNED TO PRODUCE FACTORIALS
C     ..................................................................
      IF(I.EQ.0) GO TO 11
      IF(I.EQ.1) GO TO 11
      IQ1IFC=I
      II=I-1
      DO 10 J=1,II
   10 IQ1IFC=IQ1IFC*(I-J)
```

```
      GO TO 12
   11 ICLIFC=1
   12 RETURN
      END
      SUBROUTINE Q1PPOT(DK,DR,DW,RHO,OFFSET,SIGMA,N,X,Q1ABCF,Q1RBCF,IPT)
      IMPLICIT REAL * 8(A-H,O,P,R-Y), COMPLEX * 16(Z,Q)
      DIMENSION Q1ABCF(13), Q1RBCF(13), QRR(13), QAA(13), QPHI(13)
      DIMENSION QDRR(13),QDAA(13),QDPHI(13)
      DIMENSION X(13)
      IF(IPT.LE.0) RETURN
      WRITE (6,990) DK,DR,DW,RHO,OFFSET,SIGMA,N,IPT
  990 FORMAT('1',10X,'Q1PPOT -CALCULATION OF COMPLEX POTENTIALS FOR ZONE
     1 ONE',/,'0',10X,'THESE CALCULATIONS ARE BASED ON THE SPANNING FUNC
     2TION COEFFICIENTS GIVEN DIRECTLY ABOVE',/
     3 ,' ',10X,'DK',13X,'DR',13X,'DK',13X,'RHO',12X,'OFFSET',9X,'SIGMA'
     4 ,10X,'N',3X,'IPT',/,' ',10X,
     56(E12.5,3X),I2,2X,I3)
   90 FORMAT ('0 X-STATION NUMBER ',I2,', XSTN = ',F6.4,/,
     1'   BL TOTAL POTENTIAL',14X,'REF BL POTENTIAL',8X,
     2 'ADJ BL POTENTIAL',8X,'REF BL INT POTENTIAL',5X,
     3 'ADJ BL INT POTENTIAL')
   91 FORMAT('   R   ',2(E14.7,','),8(E11.4,','))
   92 FORMAT('   A   ',2(E14.7,','),8(E11.4,','))
      ICT = IPT - 1
      RZERO = 0.0D0
      DO 10 I = 1,N
      XSTN = X(I)
      QRO = Q1PRBP(DK,DR,DW,RZERO,XSTN,IOT)
      QRP = Q1PRBP(DK,DR,DW,RHO,XSTN,IOT)
      QAO = Q1PABP(DK,DR,DW,RZERO,OFFSET,SIGMA,XSTN,IOT)
      QAP = Q1PABP(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,ICT)
      QRIPO = Q1PRIP(DK,DR,DW,RZERO,OFFSET,XSTN,Q1RBCF,N,ICT)
      QRIPP = Q1PRIP(DK,DR,DW,RHO,OFFSET,XSTN,Q1RBCF,N,ICT)
      QAIPO = Q1PAIP(DK,DR,DW,RZERO,OFFSET,SIGMA,XSTN,Q1ABCF,N,ICT)
      QAIPP = Q1PAIP(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,Q1ABCF,N,IOT)
      QR = QRO + QAP + QRIPO + QAIPP
      QA = QRP + QAO + QRIPP + QAIPO
      QPHI(I)=Q1PHI(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,XSTN,ICT)
      WRITE (6,90) I,XSTN
      WRITE(6,91) QR,QRO,QAP,QRIPO,QAIPP
      WRITE(6,92) QA,QRP,QAO,QRIPP,QAIPO
      QRR(I) = QR
      QAA(I) = QA
      QDRO = Q1DRBP(DK,DR,DW,RZERO,XSTN,IOT)
      QDRP = Q1DRBP(DK,DR,DW,RHO,XSTN,ICT)
      QDAO = Q1DABP(DK,DR,DW,RZERO,OFFSET,SIGMA,XSTN,IOT)
      QDAP = Q1DABP(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IOT)
      QDRIPO = Q1DRIP(DK,DR,DW,RZERO,OFFSET,XSTN,Q1RBCF,N,IOT)
```

```
      QDRIPP = Q1DRIP(DK,DR,DW,RHO,OFFSET,XSTN,Q1RBCF,N,IOT)
      QDAIPO = Q1DAIP(DK,DR,DW,RZERO,OFFSET,SIGMA,XSTN,Q1ABCF,N,IOT)
      QDAIPP = Q1DAIP(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,Q1ABCF,N,IOT)
      QDR = QDRO + QDAP + QDRIPO + QDAIPP
      QDA = QDRP + QDAO + QDRIPP + QDAIPO
      QDPHI(I)=Q1DPHI(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,
     1      XSTN,IOT)
      WRITE (6,95) I,XSTN
      WRITE(6,91) QDR,QDRO,QDAP,QDRIPO,QDAIPP
      WRITE(6,92) QDA,QDRP,QDAO,QDRIPP,QDAIPO
      QDRR(I) = QDR
      QDAA(I) = QDA
   95 FORMAT ('0 X-STATION NUMBER ',I2,', XSTN = ',F6.4,/,
     1'   BL TOTAL D(POT)/DX',14X,'REF BL D(POT)/DX',8X,
     2 'ADJ BL D(POT)/DX',8X,'REF BL INT D(POT)/DX',5X,
     3 'ADJ BL INT D(POT)/DX')
   10 CONTINUE
      WRITE(6,994)
  994 FORMAT ('1',10X,'SUMMARY LISTING',/,
     1'0',10X,'XSTN',7X,'SINGLE BLADE TOTAL POTENTIAL',6X,
     1  'REF BLADE POTENTIAL',15X,'ADJ BLADE POTENTIAL')
      DO 20 I = 1,N
      XSTN = X(I)
      WRITE(6,94) XSTN,QPHI(I),QRR(I),QAA(I)
   94 FORMAT(' ',10X,F6.4,3(5X,E14.7,',',E14.7))
   20 CONTINUE
      WRITE(6,996)
  996 FORMAT('0',10X,'XSTN',7X,'SINGLE BLADE TOTAL D(POT)/DX',6X,
     1  'REF BLADE D(POT)/DX',15X,'ADJ BLADE D(POT)/DX')
      DO 50 I = 1,N
      XSTN = X(I)
      WRITE(6,94) XSTN,QDPHI(I),QDRR(I),QDAA(I)
   50 CONTINUE
      RETURN
      END
      SUBROUTINE Q1DCOF(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION Q1ABCF(13), Q1RBCF(13)
      IF(IPT.GT.0)WRITE (6,990) DK,DR,DW,RHO,OFFSET,SIGMA,N,IPT
  990 FORMAT('1',10X,'Q1DCOF - CALCULATION OF COMPLEX DIMENSIONLESS AERO
     1DYNAMIC COEFFICIENTS',/,
     3 '0',10X,'DK',13X,'DR',13X,'DW',13X,'RHO',12X,'OFFSET',9X,'SIGMA'
     4 ,10X,'N',3X,'IPT',/,' ',10X,
     56(E12.5,3X),I2,2X,I3)
      IOT = IPT - 3
  999 QDCL = Q1DCL(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,IOT)
      QDCM = Q1DCM(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,IOT)
      GAMMA = 1.4D0
```

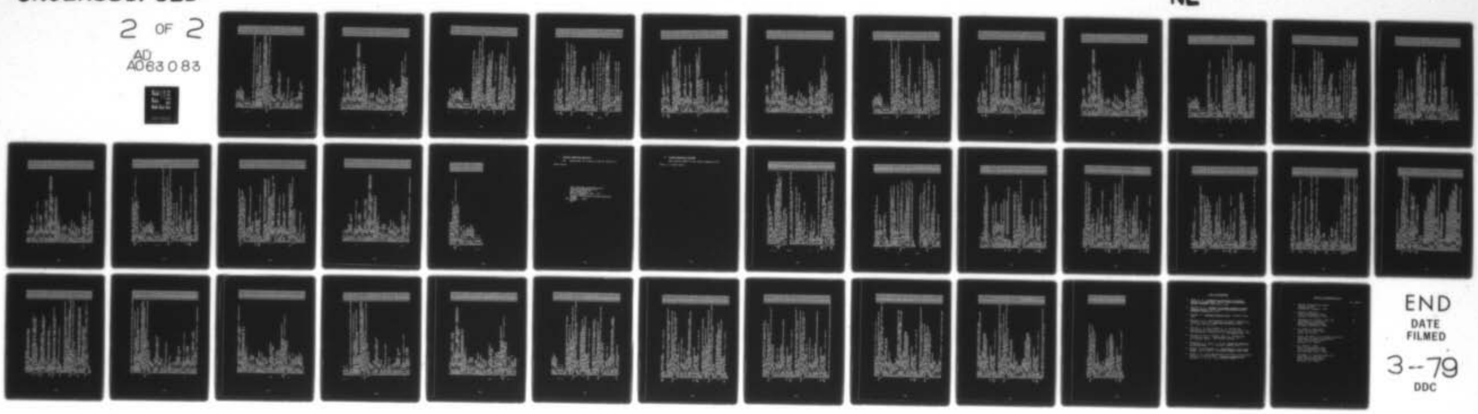
```
      TAU = (2.0D0*DSQRT((GAMMA + 1.0D0)*DW))/DR
      WRITE (6,90) DK,TAU,DW,N,SIGMA,CDCL,CDCM
 C FORMAT('0',5X,'DK = ',F6.3,', TAU = ',F7.4,', DW = ',F6.3,', N = '
 1       ,I2,', SIGMA = ',F6.3,', CL = ',F9.4,',',F9.4,', CM = ',
 2       F9.4,',',F9.4)
      RETURN
      END
      COMPLEX FUNCTION Q1DCL*16(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBC
     1F,IPT)
      IMPLICIT REAL*8(A - E,G,H,O,P,R - Y), COMPLEX*16(F,Q,Z)
      DIMENSION Q1ABCF(13), Q1RBCF(13)
      DIMENSION FV(5),F1T(60),F2T(60),F3T(60),QESTT(60),QPSUM(60)
      DIMENSION DAT(60),  ARESTT(60),EPST(60)
      DIMENSION LORR(60)
      F(X) = (Q1DPHI(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,X,ICT)
     1     -QALPHA*Q1PHI(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,X,ICT))
      IF (IPT.GT.0) WRITE (6,990) DK,DR,DW,RHO,OFFSET,SIGMA,N,IPT
 0 FORMAT('0',10X,'Q1DCL ENTERED WITH:',/,
 3 ' ',10X,'DK',13X,'DR',13X,'DK',13X,'RHO',12X,'OFFSET',9X,'SIGMA'
 4 ,10X,'N',3X,'IPT',/,' ',10X,
 56(E12.5,3X),I2,2X,I3)
      A = - 1.0D0
      B = 1.0D0
      ICT = IPT - 1
      GAMMA = 1.4D0
      DLAMDA = DK/((GAMMA + 1.0D0) * DW)
      QALPHA = DCMPLX(0.0D0,DLAMDA-DK)
      U  =  9.0E - 13
      ACC = 1.0D-5
      EFOURU = 4.0*U
      IFLAG = 1
      EPS = ACC
      QERROR = DCMPLX(0.0D0,0.0D0)
      LVL = 1
      LORR(LVL) = 1
      QPSUM(LVL) = 0.0
      ALPHA = A
      DA = B - A
      AREA = 0.0
      AREST = 0.0
      FV(1) = F(ALPHA)
      FV(3) = F(ALPHA + 0.5*DA)
      FV(5) = F(ALPHA + DA)
      KOUNT = 3
      WT = DA/6.0
      QEST = WT*(FV(1) + 4.0*FV(3) + FV(5))
      DX = 0.5*DA
      FV(2) = F(ALPHA + 0.5*DX)
```

```
      FV(4) = F(ALPHA + 1.5*DX)                                         QLDR3605
      KOUNT = KOUNT + 2                                                 QLDR3610
      WT = DX/6.0                                                       QLDR3615
      QESTL = WT*(FV(1) + 4.0*FV(2) + FV(3))                            QLDR3620
      QESTR = WT*(FV(3) + 4.0*FV(4) + FV(5))                            QLDR3625
      QSUM = QESTL + QESTR                                              QLDR3630
      ARESTL = WT*(CDABS(FV(1)) + CDABS(4.0*FV(2)) + CDABS(FV(3)))      QLDR3635
      ARESTR = WT*(CDABS(FV(3)) + CDABS(4.0*FV(4)) + CDABS(FV(5)))      QLDR3640
      AREA = AREA + ((ARESTL + ARESTR) - AREST)                         QLDR3645
      QDIFF = QEST - QSUM                                               QLDR3650
      IF(CDABS(QDIFF).LE.EPS*DABS(AREA))GO TO 2                         QLDR3655
      IF(DABS(DX).LE.EPCLRU*DABS(ALPHA))GO TO 5                         QLDR3660
      IF(LVL.GE.60)GO TO 5                                              QLDR3665
      IF(KOUNT.GE.4000)GO TO 6                                          QLDR3670
      LVL = LVL + 1                                                     QLDR3675
      LORR(LVL) = 0                                                     QLDR3680
      F1T(LVL) = FV(3)                                                  QLDR3685
      F2T(LVL) = FV(4)                                                  QLDR3690
      F3T(LVL) = FV(5)                                                  QLDR3695
      DA = DX                                                           QLDR3700
      DAT(LVL) = DX                                                     QLDR3705
      AREST = ARESTL                                                    QLDR3710
      ARESTT(LVL) = ARESTR                                              QLDR3715
      QEST = QESTL                                                      QLDR3720
      QESTT(LVL) = QESTR                                                QLDR3725
      EPS = EPS/1.4                                                     QLDR3730
      EPST(LVL) = EPS                                                   QLDR3735
      FV(5) = FV(3)                                                     QLDR3740
      FV(3) = FV(2)                                                     QLDR3745
      GO TO 1                                                           QLDR3750
2     QERROR = QERROR + QDIFF/15.0                                      QLDR3755
3     IF(LORR(LVL).EQ.0)GO TO 4                                         QLDR3760
      QSUM = QPSUM(LVL) + QSUM                                          QLDR3765
      LVL = LVL - 1                                                     QLDR3770
      IF(LVL.GT.1)GO TO 3                                               QLDR3775
      Q1DCL = QSUM * 2.0D0                                              QLDR3780
      IF(IPT.GT.0) GO TO 11                                             QLDR3785
      IF(IFLAG.EQ.1) RETURN                                             QLDR3790
      WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,N,IPT                      QLDR3795
   11 WRITE(6,995) Q1DCL,IFLAG,QERROR                                   QLDR3800
  995 FORMAT(' ',15X,'RESULTS: Q1DCL = ',E14.7,',',E14.7,/,             QLDR3805
     1' ',15X,'IFLAG',2X,'QERROR',/,                                    QLDR3810
     2' ',15X,I3,2X,E14.7,',',E14.7)                                    QLDR3815
      RETURN                                                            QLDR3820
4     QPSUM(LVL) = QSUM                                                 QLDR3825
      LORR(LVL) = 1                                                     QLDR3830
      ALPHA = ALPHA + DA                                                QLDR3835
      DA = DAT(LVL)                                                     QLDR3840
```

AD-A063 083 NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF F/G 20/4
THEORETICAL ANALYSIS OF TRANSONIC FLOW PAST UNSTAGGERED OSCILLA--ETC(U)
SEP 78 P C OLSEN

UNCLASSIFIED NL

2 OF 2
AD
A063083



END
DATE
FILMED
3--79
DDC

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
      FV(1) = F1T(LVL)                                                  QLDR3845
      FV(3) = F2T(LVL)                                                  QLDR3850
      FV(5) = F3T(LVL)                                                  QLDR3855
      AREST = ARESTT(LVL)                                               QLDR3860
      CEST = QESTT(LVL)                                                 QLDR3865
      EPS = EPST(LVL)                                                   QLDR3870
      GC TO 1                                                           QLDR3875
    5 IFLAG = 2                                                         QLDR3880
      GC TO 2                                                           QLDR3885
    6 IFLAG = 3                                                         QLDR3890
      GC TO 2                                                           QLDR3895
      END                                                               QLDR3900
      CCMPLEX FUNCTION C1DCM*16(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCQLDR3905
     1F,IPT)                                                            QLDR3910
      IMPLICIT REAL*8(A - E,G,H,O,P,R - Y), COMPLEX*16(F,C,Z)           QLDR3915
      DIMENSION Q1ABCF(13), Q1RBCF(13)                                  QLDR3920
      DIMENSICN FV(5),F1T(60),F2T(60),F3T(60),QESTT(60),CFSLM(60)       QLDR3925
      DIMENSION DAT(60),  ARESTT(60),EPST(60)                           QLDR3930
      DIMENSICN LCRR(60)                                                QLDR3935
      F(X) = X*(Q1DPHI(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,X,IOT) QLDR3940
     1   -QALPHA*C1PHI(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,X,IOT))QLDR3945
      IF (IPT.GT.0) WRITE (6,990) DK,DR,DW,RHO,OFFSET,SIGMA,N,IPT       QLDR3950
  990 FCRMAT('0',10X,'Q1DCM ENTERED WITH:',/,                           QLDR3955
     3 ' ',10X,'DK',13X,'DR',13X,'DW',13X,'RHO',12X,'OFFSET',9X,'SIGMA' QLDR3960
     4 ,10X,'N',3X,'IPT',/,' ',10X,                                     QLDR3965
     56(E12.5,3X),I2,2X,I3)                                             QLDR3970
      A = - 1.0D0                                                       QLDR3975
      B = 1.0D0                                                         QLDR3980
      ICT = IPT - 1                                                     QLDR3985
      GAMMA = 1.4D0                                                     QLDR3990
      CLAMDA = DK/((GAMMA + 1.0D0) * DW)                                QLDR3995
      QALPHA = DCMPLX(0.0D0,CLAMDA-DK)                                  QLDR4000
      U  =  9.0E - 13                                                   QLDR4005
      ACC = 1.0D-5                                                      QLDR4010
      EFOURU = 4.0*U                                                    QLDR4015
      IFLAG = 1                                                         QLDR4020
      EPS = ACC                                                         QLDR4025
      CERROR = DCMPLX(0.0D0,0.0D0)                                      QLDR4030
      LVL = 1                                                           QLDR4035
      LCRR(LVL) = 1                                                     QLDR4040
      CFSUM(LVL) = 0.0                                                  QLDR4045
      ALPHA = A                                                         QLDR4050
      DA = B - A                                                        QLDR4055
      AREA = 0.0                                                        QLDR4060
      AREST = 0.0                                                       QLDR4065
      FV(1) = F(ALPHA)                                                  QLDR4070
      FV(3) = F(ALPHA + 0.5*DA)                                         QLDR4075
      FV(5) = F(ALPHA + DA)                                             QLDR4080
```

```
      KCUNT = 3                                                         QLDR4085
      WT = DA/6.0                                                       QLDR4090
      CEST = WT*(FV(1) + 4.0*FV(3) + FV(5))                             QLDR4095
    1 DX = 0.5*DA                                                       QLDR4100
      FV(2) = F(ALPHA + C.5*DX)                                         QLDR4105
      FV(4) = F(ALPHA + 1.5*DX)                                         QLDR4110
      KOUNT = KCUNT + 2                                                 QLDR4115
      WT = DX/6.0                                                       QLDR4120
      QESTL = WT*(FV(1) + 4.0*FV(2) + FV(3))                            QLDR4125
      CESTR = WT*(FV(3) + 4.0*FV(4) + FV(5))                            QLDR4130
      CSUM = CESTL + QESTR                                              QLDR4135
      ARESTL = WT*(CDABS(FV(1)) + CDAES(4.0*FV(2)) + CDABS(FV(3)))      QLDR4140
      ARESTR = WT*(CDABS(FV(3)) + CDABS(4.0*FV(4)) + CDAES(FV(5)))      QLDR4145
      AREA = AREA + ((ARESTL + ARESTR) - AREST)                         QLDR4150
      CDIFF = QEST - QSUM                                               QLDR4155
      IF(CDABS(QDIFF).LE.EPS*DABS(AREA))GO TO 2                         QLDR4160
      IF(DABS(DX).LE.EFCURU*DABS(ALPHA))GO TO 5                         QLDR4165
      IF(LVL.GE.60)GO TO 5                                              QLDR4170
      IF(KOUNT.GE.4000)GC TO 6                                          QLDR4175
      LVL = LVL + 1                                                     QLDR4180
      LCRR(LVL) = 0                                                     QLDR4185
      F1T(LVL) = FV(3)                                                  QLDR4190
      F2T(LVL) = FV(4)                                                  QLDR4195
      F3T(LVL) = FV(5)                                                  QLDR4200
      DA = DX                                                           QLDR4205
      DAT(LVL) = CX                                                     QLDR4210
      AREST = ARESTL                                                    QLDR4215
      ARESTT(LVL) = ARESTR                                              QLDR4220
      CEST = QESTL                                                      QLDR4225
      CESTT(LVL) = QESTR                                                QLDR4230
      EPS = EPS/1.4                                                     QLDR4235
      EPST(LVL) = EPS                                                   QLDR4240
      FV(5) = FV(3)                                                     QLDR4245
      FV(3) = FV(2)                                                     QLDR4250
      GC TO 1                                                           QLDR4255
    2 QERRCR = CERROR + QCIFF/15.0                                      QLDR4260
    3 IF(LORR(LVL).EQ.0)GO TO 4                                         QLDR4265
      CSUM = QPSUM(LVL) + QSUM                                          QLDR4270
      LVL = LVL - 1                                                     QLDR4275
      IF(LVL.GT.1)GO TO 3                                               QLDR4280
      Q1DCM = CSUM * 2.0C0                                              QLDR4285
      IF(IPT.GT.0) GO TC 11                                             QLDR4290
      IF(IFLAG.EQ.1) RETURN                                             QLDR4295
      WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,N,IPT                      QLDR4300
   11 WRITE(6,995) Q1DCM,IFLAG,QERRCR                                   QLDR4305
  995 FORMAT(' ',15X,'RESULTS: Q1CCM = ',E14.7,',',E14.7,/,             QLDR4310
     1' ',15X,'IFLAG',2X,'QERROR',/,                                    QLDR4315
     2' ',15X,I3,2X,E14.7,',',E14.7)                                    QLDR4320
```

```
      RETURN
4     QPSUM(LVL) = QSUM
      LCRR(LVL) = 1
      ALPHA = ALPHA + DA
      DA = DAT(LVL)
      FV(1) = F1T(LVL)
      FV(3) = F2T(LVL)
      FV(5) = F3T(LVL)
      AREST = ARESTT(LVL)
      QEST = QESTT(LVL)
      EPS = EPST(LVL)
      GO TO 1
5     IFLAG = 2
      GO TO 2
6     IFLAG = 3
      GO TO 2
      END
      COMPLEX FUNCTION Q1PHI*16 (DK,DR,DW,RHO,OFFSET,SIGMA,
     1     N,Q1ABCF,Q1RBCF,XSTN,IPT)
      IMPLICIT REAL*8(A-H,O,P,R-Y), COMPLEX*16(Q,Z)
      DIMENSION Q1ABCF(13), Q1RBCF(13)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT
990   FORMAT('0',10X,'Q1PHI ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFFSET',7X,'SIGMA'
     2  ,8X,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)
      RZERO = 0.0D0
      IOT = IPT - 1
      Q1PHI = Q1PRBP(DK,DR,DW,RZERO,XSTN,IOT)
     1 + Q1PRIP(DK,DR,DW,RZERO,OFFSET,XSTN,Q1RBCF,N,IOT)
     2 + Q1PABP(DK,DR,DW,DR,OFFSET,SIGMA,XSTN,IOT)
     3 + Q1PAIP(DK,DR,DW,DR,OFFSET,SIGMA,XSTN,Q1ABCF,N,IOT)
      IF(IPT.LE.0) RETURN
      WRITE (6,995) Q1PHI
995   FORMAT('0',10X,'Q1PHI = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1PRBP*16(DK,DR,DW,RHO,XSTN,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION QINP(2)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,XSTN,IPT
990   FORMAT('0',10X,'Q1PRBP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'XSTN',9X,'IPT'
     2  ,/,' ',10X,5(E12.5,','),I3)
      IF(XSTN.LE.RHO-1.0D0) GOTO 20
      IOT = IPT - 1
      QDK = DCMPLX (0.0D0,DK)
      CALL Q1PWXM(DK,DR,DW,RHO,XSTN,QINP,IOT)
      Q1PRBP = -QDK*QINP(2) - QINP(1)
```

```
      IF(IPT.LE.0) RETURN
      GOTO 60
   20 Q1PRBP = DCMPLX(0.0D0,0.0D0)
      IF(IPT.LE.0) RETURN
   60 WRITE (6,995) Q1PRBP
  995 FORMAT('0',10X,'Q1PRBP = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1PABP*16(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION QINP(2)
      IF(IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT
  990 FORMAT('0',10X,'Q1PABP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFFSET',7X,
     2  'SIGMA',8X,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)
      XASTN = XSTN - OFFSET
      IF(XASTN.LE.RHO-1.0D0) GOTO 20
      IOT = IPT - 1
      QCK = DCMPLX (0.0D0,DK)
      QCONST = CDEXP( DCMPLX(0.0D0,SIGMA))
      CALL Q1PWXM(DK,DR,DW,RHO,XASTN,QINP,IOT)
      Q1PABP = (QCK*QINP(2) + QINP(1)) * QCONST
      IF(IPT.LE.0) RETURN
      GO TO 60
   20 Q1PABP = DCMPLX(0.0D0,0.0D0)
      IF (IPT.LE.0) RETURN
   60 WRITE (6,995) Q1PABP
  995 FORMAT ('0', 10X, 'Q1PABP = ', E14.7,',',E14.7)
      RETURN
      END
      SUBROUTINE Q1PWXM(DK,DR,DW,RHO,XSTN,QINP,IPT)
C
C     N IS THE MAXIMUM DEGREE OF THE TERM U IN THE INTEGRALS
C
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Z,Q)
      DIMENSION QINP(2)
      IF(IPT.GT.0) WRITE(6,990) DK,DR,RHO,XSTN,IPT
  990 FORMAT('0',10X,'Q1PWXM ENTERED WITH ARGUMENTS:',/,
     1' ',10X,'DK',16X,'DR',16X,'RHO',15X,'XSTN',14X,'IPT',/,
     2' ',10X,4(E13.6,5X),I2,5X,I3)
      IOT = IPT - 1
      B = XSTN - RHO - 1.0D-10
      A = -1.0D0
      CALL Q1PZXM(DK,DR,DW,RHO,XSTN,A,B,1,QANS,IOT)
      QINP(1) = QANS
      CALL Q1PZXM(DK,DR,DW,RHO,XSTN,A,B,2,QANS,IOT)
      QINP(2) = QANS
      IF(IPT.LE.0) RETURN
```

```
      WRITE (6,995)
  995 FORMAT(' ',10X,'Q1PWXM RESULTS:',/,
     1' ',11X,'J',6X,'QINP(I)')
      DO 1001 I = 1,2
  996 FORMAT(' ',10X,I3,5X,E12.6,',',E12.6)
 1001 WRITE (6,996) I, QINP(I)
      RETURN
      END
      SUBROUTINE Q1PZXM (DK,DR,DW,RHO,XSTN,A,B,J,CANS,IPT)
      IMPLICIT REAL*8 (A - E,G,H,M,O,P,R - Y), COMPLEX*16(F,Q,Z)
      DIMENSION FV(5),LORR(60),F1T(60),F2T(60),F3T(60),DAT(60),
     1  ARESTT(60),QESTT(60),EPST(60),CFSUM(60)
      F(X) = (X**J1)*CDEXP(QEXP*(X))
     1       * MMESJO((OMEGA * DSQRT( (XSTN-X)*(XSTN-X) - YY)), IER)
      GAMMA = 1.4D0
      YY = RHO * RHO
      DM2 = (GAMMA + 1.0D0) * DW
      OMEGA = DSQRT (DK*DK*(1.0D0-DM2)/(DM2*DM2))
      DLAMDA = DK/DM2
      DM = DSQRT(DM2)
      Q2EXP = CDEXP (DCMPLX (0.0D0, -DLAMDA*XSTN))
      QEXP = DCMPLX(0.0D0,DLAMDA)
      J1 = J - 1
      U  =  9.0D-13
      ACC = 1.0D-6
      IF (IPT.GT.0) WRITE(6,990)DK,DR,DW,RHO,XSTN,A,B,J,IPT
  990 FORMAT('0',15X,'Q1PZXM ENTERED WITH ARGUMENTS:',/,
     1' ',15X,'DK',13X,'DR',13X,'DW    ',10X,'RHO',12X,'XSTN',11X,
     2'A',14X,'B',14X,'J',2X,'IPT',/,
     3' ',15X,7(E14.7,','),I2,2X,I3)
      EPCURU = 4.0*U
      IFLAG = 1
      EPS = ACC
      CERROR = DCMPLX(0.0D0,0.0D0)
      LVL = 1
      LORR(LVL) = 1
      CFSUM(LVL) = 0.0
      ALPHA = A
      DA = B - A
      AREA = 0.0
      AREST = 0.0
      FV(1) = F(ALPHA)
      FV(3) = F(ALPHA + 0.5*DA)
      FV(5) = F(ALPHA + DA)
      KOUNT = 3
      WT = DA/6.0
      CEST = WT*(FV(1) + 4.0*FV(3) + FV(5))
    1 DX = 0.5*DA
```

```
      FV(2) = F(ALPHA + 0.5*DX)
      FV(4) = F(ALPHA + 1.5*DX)
      KOUNT = KOUNT + 2
      W1 = DX/6.0
      QESTL = WT*(FV(1) + 4.0*FV(2) + FV(3))
      QESTR = WT*(FV(3) + 4.0*FV(4) + FV(5))
      QSUM = QESTL + QESTR
      ARESTL = WT*(CDABS(FV(1)) + CDABS(4.0*FV(2)) + CDABS(FV(3)))
      ARESTR = WT*(CDABS(FV(3)) + CDABS(4.0*FV(4)) + CDABS(FV(5)))
      AREA = AREA + ((ARESTL + ARESTR) - AREST)
      QDIFF = QEST - QSUM
      IF(CDABS(QDIFF).LE.EPS*DABS(AREA))GO TO 2
      IF(DABS(DX).LE.EFOURU*DABS(ALPHA))GO TO 5
      IF(LVL.GE.60)GO TO 5
      IF(KOUNT.GE.2000)GO TO 6
      LVL = LVL + 1
      LORR(LVL) = 0
      F1T(LVL) = FV(3)
      F2T(LVL) = FV(4)
      F3T(LVL) = FV(5)
      DA = DX
      DAT(LVL) = DX
      AREST = ARESTL
      ARESTT(LVL) = ARESTR
      QEST = QESTL
      QESTT(LVL) = QESTR
      EPS = EPS/1.4
      EPST(LVL) = EPS
      FV(5) = FV(3)
      FV(3) = FV(2)
      GO TO 1
2     QERROR = QERROR + QDIFF/15.0
3     IF(LORR(LVL).EQ.0)GO TO 4
      QSUM = QPSUM(LVL) + QSUM
      LVL = LVL - 1
      IF(LVL.GT.1)GO TO 3
      QANS = QSUM * Q2EXP / DM
      IF(IPT.GT.0) GO TO 11
      IF(IFLAG.EQ.1) RETURN
      WRITE(6,990) DK,DR,DW,RHO,XSTN,A,B,J,IPT
   11 WRITE(6,995) QANS,IFLAG,IER,QERROR
  995 FORMAT(' ',15X,'RESULTS: QANS = ',E14.7,',',E14.7,/,
     1' ',15X,'IFLAG',2X,'IER',5X,'QERROR',/,
     2' ',15X,I3,2X,I3,5X,E14.7,',',E14.7)
      RETURN
4     QPSUM(LVL) = QSUM
      LORR(LVL) = 1
      ALPHA = ALPHA + DA
```

```
      DA = DAT(LVL)                                                     QLDR5285
      FV(1) = F1T(LVL)                                                  QLDR5290
      FV(3) = F2T(LVL)                                                  QLDR5295
      FV(5) = F3T(LVL)                                                  QLDR5300
      AREST = ARESTT(LVL)                                               QLDR5305
      QEST = QESTT(LVL)                                                 QLDR5310
      EPS = EPST(LVL)                                                   QLDR5315
      GO TO 1                                                           QLDR5320
    5 IFLAG = 2                                                         QLDR5325
      GO TO 2                                                           QLDR5330
    6 IFLAG = 3                                                         QLDR5335
      GO TO 2                                                           QLDR5340
      END                                                               QLDR5345
      COMPLEX FUNCTION Q1PAIP*16(DK,DR,DW,RHO,OFST,SGMA,XSTN,Q1CF,N,IPT)QLDR5350
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)                 QLDR5355
      DIMENSION Q1CF(13)                                                QLDR5360
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFST,SGMA,XSTN,IPT        QLDR5365
  990 FORMAT('0',10X,'Q1PAIP ENTERED WITH:',/,                          QLDR5370
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFST',9X,'SGMA',9XQLDR5375
     2  ,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)                      QLDR5380
      XASTN = XSTN - OFST                                               QLDR5385
      IF(XASTN.LE.DR+RHO-1.0D0) GO TO 20                                QLDR5390
      A = DR-1.0D0                                                      QLDR5395
      B = XASTN - RHO - 1.0D-8                                          QLDR5400
      ICT = IPT - 1                                                     QLDR5405
      QCONST = CDEXP( DCMPLX(0.0D0,SGMA))                               QLDR5410
      CALL Q1PZSM(DK,DR,DW,RHO,XASTN,A,B,N,QANS,Q1CF,ICT)               QLDR5415
      Q1PAIP = QANS                                                     QLDR5420
      IF(IPT.LE.0) RETURN                                               QLDR5425
      GOTO 60                                                           QLDR5430
   20 Q1PAIP = DCMPLX(0.0D0,0.0D0)                                      QLDR5435
      IF(IPT.LE.0)RETURN                                                QLDR5440
   60 WRITE (6,995) Q1PAIP                                              QLDR5445
  995 FORMAT('0',10X,'Q1PAIP = ',E14.7,',',E14.7)                       QLDR5450
      RETURN                                                            QLDR5455
      END                                                               QLDR5460
      COMPLEX FUNCTION Q1PRIP*16(DK,DR,DW,RHO,OFFSET,XSTN,Q1CF,N,IPT)   QLDR5465
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)                 QLDR5470
      DIMENSION Q1CF(13)                                                QLDR5475
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,XSTN,IPT                  QLDR5480
  990 FORMAT('0',10X,'Q1PRIP ENTERED WITH:',/,                          QLDR5485
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,                   QLDR5490
     2  'XSTN',9X,'IPT'/,' ',10X,5(E12.5,','),I3)                       QLDR5495
      IF(XSTN.LE.DR+RHO+OFFSET-1.0D0) GOTO 20                           QLDR5500
      A = DR-1.0D0                                                      QLDR5505
      B = XSTN - RHO - 1.0D-8                                           QLDR5510
      ICT = IPT - 1                                                     QLDR5515
      CALL Q1PZSM(DK,DR,DW,RHO,XSTN,A,B,N,QANS,Q1CF,IOT)                QLDR5520
```

```
      Q1FRIP = -QANS                                                    QLDR5525
      IF(IPT.LE.0) RETURN                                               QLDR5530
      GOTO 60                                                           QLDR5535
   20 Q1PRIP = DCMPLX(0.0D0,0.0D0)                                      QLDR5540
      IF (IPT.LE.0) RETURN                                              QLDR5545
   60 WRITE (6,995) Q1PRIP                                              QLDR5550
  995 FORMAT('0',10X,'Q1FRIP = ',E14.7,',',E14.7)                       QLDR5555
      RETURN                                                            QLDR5560
      END                                                               QLDR5565
      SUBROUTINE Q1PZSM (DK,DR,DW,RHO,XSTN,A,B,J,QANS,Q1CF,IPT)         QLDR5570
      IMPLICIT REAL*8 (A - E,G,H,M,O,P,R - Y), COMPLEX*16(F,Q,Z)        QLDR5575
      DIMENSION Q1CF(13)                                                QLDR5580
      DIMENSION FV(5),LORR(60),F1T(60),F2T(60),F3T(60),DAT(60),         QLDR5585
     1  ARESTT(60),QESTT(60),EPST(60),QPSUM(60)                         QLDR5590
      F(X) = QLGNDR(X,DR,J,Q1CF)*CDEXP(QEXP*(X))                        QLDR5595
     1     * MMBSJC((OMEGA * DSQRT( (XSTN-X)*(XSTN-X) - YY)), IER)      QLDR5600
      GAMMA = 1.4D0                                                     QLDR5605
      YY = RHO * RHO                                                    QLDR5610
      DM2 = (GAMMA + 1.0D0) * DW                                        QLDR5615
      OMEGA = DSQRT (DK*DK*(1.0D0-DM2)/(DM2*DM2))                       QLDR5620
      DLAMDA = DK/DM2                                                   QLDR5625
      DM = DSQRT(DM2)                                                   QLDR5630
      Q2EXP = CDEXP (DCMPLX (0.0D0, -DLAMDA*XSTN))                      QLDR5635
      QEXP = DCMPLX(0.0D0,DLAMDA)                                       QLDR5640
      U  =  9.0D-13                                                     QLDR5645
      ACC = 1.0D-6                                                      QLDR5650
      IF (IPT.GT.0) WRITE(6,990)DK,DR,DW,RHO,XSTN,A,B,J,IPT             QLDR5655
  990 FORMAT('0',15X,'Q1PZSM ENTERED WITH ARGUMENTS:',/,                QLDR5660
     1' ',15X,'DK',13X,'DR',13X,'DW   ',10X,'RHO',12X,'XSTN',11X,       QLDR5665
     2'A',14X,'B',14X,'J',2X,'IPT',/,                                   QLDR5670
     3' ',15X,7(E14.7,','),I2,2X,I3)                                    QLDR5675
      EFOURU = 4.0*U                                                    QLDR5680
      IFLAG = 1                                                         QLDR5685
      EPS = ACC                                                         QLDR5690
      QERROR = DCMPLX(0.0D0,0.0D0)                                      QLDR5695
      LVL = 1                                                           QLDR5700
      LORR(LVL) = 1                                                     QLDR5705
      QPSUM(LVL) = 0.0                                                  QLDR5710
      ALPHA = A                                                         QLDR5715
      DA = B - A                                                        QLDR5720
      AREA = 0.0                                                        QLDR5725
      AREST = 0.0                                                       QLDR5730
      FV(1) = F(ALPHA)                                                  QLDR5735
      FV(3) = F(ALPHA + 0.5*DA)                                         QLDR5740
      FV(5) = F(ALPHA + DA)                                             QLDR5745
      KOUNT = 3                                                         QLDR5750
      WT = DA/6.0                                                       QLDR5755
      QEST = WT*(FV(1) + 4.0*FV(3) + FV(5))                             QLDR5760
```

```
1     DX = 0.5*DA                                                        QLDR5765
      FV(2) = F(ALPHA + 0.5*DX)                                          QLDR5770
      FV(4) = F(ALPHA + 1.5*DX)                                          QLDR5775
      KOUNT = KOUNT + 2                                                  QLDR5780
      WT = DX/6.0                                                        QLDR5785
      QESTL = WT*(FV(1) + 4.0*FV(2) + FV(3))                             QLDR5790
      QESTR = WT*(FV(3) + 4.0*FV(4) + FV(5))                             QLDR5795
      QSUM = QESTL + QESTR                                               QLDR5800
      ARESTL = WT*(CDABS(FV(1)) + CDABS(4.0*FV(2)) + CDABS(FV(3)))      QLDR5805
      ARESTR = WT*(CDABS(FV(3)) + CDABS(4.0*FV(4)) + CDABS(FV(5)))      QLDR5810
      AREA = AREA + ((ARESTL + ARESTR) - AREST)                          QLDR5815
      QDIFF = QEST - QSUM                                                QLDR5820
      IF(CDABS(QDIFF).LE.EPS*DABS(AREA))GO TO 2                          QLDR5825
      IF(DABS(DX).LE.EFCURU*DABS(ALPHA))GO TO 5                          QLDR5830
      IF(LVL.GE.60)GO TO 5                                               QLDR5835
      IF(KOUNT.GE.2000)GO TO 6                                           QLDR5840
      LVL = LVL + 1                                                      QLDR5845
      LORR(LVL) = 0                                                      QLDR5850
      F1T(LVL) = FV(3)                                                   QLDR5855
      F2T(LVL) = FV(4)                                                   QLDR5860
      F3T(LVL) = FV(5)                                                   QLDR5865
      DA = DX                                                            QLDR5870
      DAT(LVL) = DX                                                      QLDR5875
      AREST = ARESTL                                                     QLDR5880
      ARESTT(LVL) = ARESTR                                               QLDR5885
      QEST = QESTL                                                       QLDR5890
      QESTT(LVL) = QESTR                                                 QLDR5895
      EPS = EPS/1.4                                                      QLDR5900
      EPST(LVL) = EPS                                                    QLDR5905
      FV(5) = FV(3)                                                      QLDR5910
      FV(3) = FV(2)                                                      QLDR5915
      GO TO 1                                                            QLDR5920
2     QERROR = QERROR + QDIFF/15.0                                       QLDR5925
3     IF(LORR(LVL).EQ.0)GO TO 4                                          QLDR5930
      QSUM = QPSUM(LVL) + QSUM                                           QLDR5935
      LVL = LVL - 1                                                      QLDR5940
      IF(LVL.GT.1)GO TO 3                                                QLDR5945
      QANS = QSUM * Q2EXP / DM                                           QLDR5950
      IF(IPT.GT.0) GO TO 11                                              QLDR5955
      IF(IFLAG.EQ.1) RETURN                                              QLDR5960
      WRITE(6,990) DK,DR,DW,RHO,XSTN,A,E,J,IPT                           QLDR5965
   11 WRITE(6,995) QANS,IFLAG,IER,QERRCR                                 QLDR5970
  995 FORMAT(' ',15X,'RESULTS: QANS = ',E14.7,',',E14.7,/,               QLDR5975
     1' ',15X,'IFLAG',2X,'IER',5X,'QERRCR',/,                            QLDR5980
     2' ',15X,I3,2X,I3,5X,E14.7,',',E14.7)                               QLDR5985
      RETURN                                                             QLDR5990
4     QFSUM(LVL) = QSUM                                                  QLDR5995
      LORR(LVL) = 1                                                      QLDR6000
```

```
      ALPHA = ALPHA + DA
      DA = DAT(LVL)
      FV(1) = F1T(LVL)
      FV(3) = F2T(LVL)
      FV(5) = F3T(LVL)
      AREST = ARESTT(LVL)
      QEST = QESTT(LVL)
      EPS = EPST(LVL)
      GO TO 1
5     IFLAG = 2
      GO TO 2
6     IFLAG = 3
      GO TO 2
      END
      COMPLEX FUNCTION QLGNDR*16(X,DR,N,Q1CF)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y),  COMPLEX * 16(Q,Z)
      DIMENSION Q1CF(13)
      ZSUM = 0.0D0
      DO 10 J1 = 1,N
      J= J1 - 1
      ZSUM = ZSUM + PLGNDR(X,DR,J) * Q1CF(J1)
   10 CONTINUE
      QLGNDR = ZSUM
      RETURN
      END
      COMPLEX FUNCTION Q1DPHI*16 (DK,DR,DW,RHO,OFFSET,SIGMA,
     1    N,Q1ABCF,Q1RBCF,XSTN,IPT)
      IMPLICIT REAL*8(A-H,O,P,R-Y), COMPLEX*16(Q,Z)
      DIMENSION Q1ABCF(13),Q1RBCF(13)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT
  990 FORMAT('0',10X,'Q1DPHI ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFFSET',7X,'SIGMA'
     2  ,8X,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)
      RZERO = 0.0D0
      IOT = IPT - 1
      Q1DPHI = Q1DREP(DK,DR,DW,RZERO,XSTN,IOT)
     1 + Q1DRIP(DK,DR,DW,RZERO,OFFSET,XSTN,Q1RBCF,N,IOT)
     2 + Q1DAEP(DK,DR,DW,DR,OFFSET,SIGMA,XSTN,IOT)
     3 + Q1DAIP(DK,DR,DW,DR,OFFSET,SIGMA,XSTN,Q1ABCF,N,IOT)
      IF(IPT.LE.0) RETURN
      WRITE (6,995) Q1DPHI
  995 FORMAT('0',10X,'Q1DPHI = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1DREP*16(DK,DR,DW,RHO,XSTN,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION QINP(2)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,XSTN,IPT
```

```
990 FORMAT('0',10X,'Q1DRBP ENTERED WITH:',/,
   1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'XSTN',9X,'IPT'
   2  ,/,' ',10X,5(E12.5,','),I3)
      IF(XSTN.LE.RHO-1.0D0) GOTO 20
      ICT = IPT - 1
      QDK = DCMPLX (0.0D0,DK)
      CALL Q1DWXM(DK,DR,DW,RHO,XSTN,QINP,IOT)
      Q1DRBP = -QDK*QINP(2) - QINP(1)
      IF(IPT.LE.0) RETURN
      GOTO 60
   20 Q1DRBP = DCMPLX(0.0D0,0.0D0)
      IF(IPT.LE.0) RETURN
   60 WRITE (6,995) Q1DRBP
  995 FORMAT('0',10X,'Q1DRBP = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1DABP*16(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION QINP(2)
      IF(IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT
  990 FORMAT('0',10X,'Q1DABP ENTERED WITH:',/,
   1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFFSET',7X,
   2  'SIGMA',8X,'XSTN',9X,'IPT',/,' ',10X,7(E12.5,','),I3)
      XASTN = XSTN - OFFSET
      IF(XASTN.LE.RHO-1.0D0) GOTO 20
      ICT = IPT - 1
      QDK = DCMPLX (0.0D0,DK)
      QCONST = CDEXP( DCMPLX(0.0D0,SIGMA))
      CALL Q1DWXM(DK,DR,DW,RHO,XASTN,QINP,IOT)
      Q1DABP = (QDK*QINP(2) + QINP(1)) * QCONST
      IF(IPT.LE.0) RETURN
      GO TO 60
   20 Q1DABP = DCMPLX(0.0D0,0.0D0)
      IF (IPT.LE.0) RETURN
   60 WRITE (6,995) Q1DABP
  995 FORMAT('0',10X,'Q1DABP = ',E14.7,',',E14.7)
      RETURN
      END
      SUBROUTINE Q1DWXM(DK,DR,DW,RHO, XSTN,QINP,IPT)
C
C     N IS THE MAXIMUM DEGREE OF THE TERM U IN THE INTEGRALS
C
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Z,Q)
      DIMENSION QINP(2)
      IF(IPT.GT.0) WRITE(6,990) DK,DR,RHO,XSTN,IPT
  990 FORMAT('0',10X,'Q1DWXM ENTERED WITH ARGUMENTS:',/,
   1' ',10X,'DK',16X,'DR',16X,'RHO',15X,'XSTN',14X,'IPT',/,
   2' ',10X,4(E13.6,5X),I2,5X,I3)
```

```
      ICT = IPT - 1
      B = XSTN - RHO - 1.0D-10
      A = -1.0D0
      CALL Q1DZXM(DK,DR,DW,RHO,XSTN,A,B,1,QANS,IOT)
      QINP(1) = QANS
      CALL Q1DZXM(DK,DR,DW,RHO,XSTN,A,B,2,QANS,IOT)
      QINP(2) = QANS
      IF(IPT.LE.0) RETURN
      WRITE (6,995)
  995 FORMAT(' ',10X,'Q1DWXM RESULTS:',/,
     1' ',11X,'J',6X,'QINP(I)')
      DO 1001 I = 1,2
  996 FORMAT(' ',10X,I3,5X,E12.6,',',E12.6)
 1001 WRITE (6,996) I, QINP(I)
      RETURN
      END
      SUBROUTINE Q1DZXM (DK,DR,DW,RHO,XSTN,A,B,J,QANS,IPT)
      IMPLICIT REAL*8(A - E,G,H,M,O,P,R - Y), COMPLEX*16(F,Q,Z)
      DIMENSION FV(5),LORR(60),F1T(60),F2T(60),F3T(60),DAT(60),
     1  ARESTT(60),QESTT(60),EPST(60),QPSUM(60)
      F(X) = (X**J1)*CDEXP(QEXP*(X))
     1     * MMBSJ1((OMEGA * DSQRT( (XSTN-X)*(XSTN-X) - YY)), IER)
     2     * OMEGA * (XSTN-X)/(DSQRT((XSTN-X)*(XSTN-X)-YY))
      GAMMA = 1.4D0
      YY = RHO * RHO
      DM2 = (GAMMA + 1.0D0) * DW
      OMEGA = DSQRT (DK*DK*(1.0D0-DM2)/(DM2*DM2))
      DLAMDA = DK/DM2
      DM = DSQRT(DM2)
      Q2EXP = CDEXP (DCMPLX (0.0D0, -DLAMDA*XSTN))
      QEXP = DCMPLX(0.0D0,DLAMDA)
      J1 = J - 1
      U  =   9.0D-13
      ACC = 1.0D-6
      IF (IPT.GT.0) WRITE(6,990)DK,DR,DW,RHO,XSTN,A,B,J,IPT
  990 FORMAT('0',15X,'Q1DZXM ENTERED WITH ARGUMENTS:',/,
     1' ',15X,'DK',13X,'DR',13X,'DW     ',10X,'RHO',12X,'XSTN',11X,
     2'A',14X,'B',14X,'J',2X,'IPT',/,
     3' ',15X,7(E14.7,','),I2,2X,I3)
      EPOURU = 4.0*U
      IFLAG = 1
      EPS = ACC
      QERROR = DCMPLX(0.0D0,0.0D0)
      LVL = 1
      LORR(LVL) = 1
      QPSUM(LVL) = 0.0
      ALPHA = A
      DA = B - A
```

```
      AREA = 0.0                                                        QLDR6725
      AREST = 0.0                                                       QLDR6730
      FV(1) = F(ALPHA)                                                  QLDR6735
      FV(3) = F(ALPHA + 0.5*DA)                                         QLDR6740
      FV(5) = F(ALPHA + DA)                                             QLDR6745
      KOUNT = 3                                                         QLDR6750
      WT = DA/6.0                                                       QLDR6755
      QEST = WT*(FV(1) + 4.0*FV(3) + FV(5))                             QLDR6760
    1 DX = 0.5*DA                                                       QLDR6765
      FV(2) = F(ALPHA + 0.5*DX)                                         QLDR6770
      FV(4) = F(ALPHA + 1.5*DX)                                         QLDR6775
      KOUNT = KOUNT + 2                                                 QLDR6780
      WT = DX/6.0                                                       QLDR6785
      QESTL = WT*(FV(1) + 4.0*FV(2) + FV(3))                            QLDR6790
      QESTR = WT*(FV(3) + 4.0*FV(4) + FV(5))                            QLDR6795
      QSUM = QESTL + QESTR                                              QLDR6800
      ARESTL = WT*(CDABS(FV(1)) + CDABS(4.0*FV(2)) + CDABS(FV(3)))      QLDR6805
      ARESTR = WT*(CDABS(FV(3)) + CDABS(4.0*FV(4)) + CDABS(FV(5)))      QLDR6810
      AREA = AREA + ((ARESTL + ARESTR) - AREST)                         QLDR6815
      QDIFF = QEST - QSUM                                               QLDR6820
      IF(CDABS(QDIFF).LE.EPS*DABS(AREA))GO TO 2                         QLDR6825
      IF(DABS(DX).LE.EFCURU*DABS(ALPHA))GO TO 5                         QLDR6830
      IF(LVL.GE.60)GO TO 5                                              QLDR6835
      IF(KOUNT.GE.4000)GO TO 6                                          QLDR6840
      LVL = LVL + 1                                                     QLDR6845
      LORR(LVL) = 0                                                     QLDR6850
      F1T(LVL) = FV(3)                                                  QLDR6855
      F2T(LVL) = FV(4)                                                  QLDR6860
      F3T(LVL) = FV(5)                                                  QLDR6865
      DA = DX                                                           QLDR6870
      DAT(LVL) = DX                                                     QLDR6875
      AREST = ARESTL                                                    QLDR6880
      ARESTT(LVL) = ARESTR                                              QLDR6885
      QEST = QESTL                                                      QLDR6890
      QESTT(LVL) = QESTR                                                QLDR6895
      EPS = EPS/1.4                                                     QLDR6900
      EPST(LVL) = EPS                                                   QLDR6905
      FV(5) = FV(3)                                                     QLDR6910
      FV(3) = FV(2)                                                     QLDR6915
      GO TO 1                                                           QLDR6920
    2 QERROR = QERROR + QDIFF/15.0                                      QLDR6925
    3 IF(LORR(LVL).EQ.0)GO TO 4                                         QLDR6930
      QSUM = QPSUM(LVL) + QSUM                                          QLDR6935
      LVL = LVL - 1                                                     QLDR6940
      IF(LVL.GT.1)GO TO 3                                               QLDR6945
      QANS = ((B**J1)*CDEXP(QEXP*B) - QSUM)*Q2EXP/DM                    QLDR6950
      IF(IPT.GT.0) GO TO 11                                             QLDR6955
      IF(IFLAG.EQ.1) RETURN                                             QLDR6960
```

```
      WRITE(6,990) DK,DF,DW,RHO,XSTN,A,B,J,IPT                          QLDR6965
   11 WRITE(6,995) QANS,IFLAG,IER,QERROR                                 QLDR6970
  995 FORMAT(' ',15X,'Q1DZXM RESULTS: QANS = ',E14.7,',',E14.7,/,        QLDR6975
     1' ',15X,'IFLAG',2X,'IER',5X,'QERROR',/,                            QLDR6980
     2' ',15X,I3,2X,I3,5X,E14.7,',',E14.7)                               QLDR6985
      RETURN                                                             QLDR6990
    4 QFSUM(LVL) = QSUM                                                  QLDR6995
      LORR(LVL) = 1                                                      QLDR7000
      ALPHA = ALPHA + DA                                                 QLDR7005
      DA = DAT(LVL)                                                      QLDR7010
      FV(1) = F1T(LVL)                                                   QLDR7015
      FV(3) = F2T(LVL)                                                   QLDR7020
      FV(5) = F3T(LVL)                                                   QLDR7025
      AREST = ARESTT(LVL)                                                QLDR7030
      QEST = QESTT(LVL)                                                  QLDR7035
      EPS = EPST(LVL)                                                    QLDR7040
      GO TO 1                                                            QLDR7045
    5 IFLAG = 2                                                          QLDR7050
      GO TO 2                                                            QLDR7055
    6 IFLAG = 3                                                          QLDR7060
      GO TO 2                                                            QLDR7065
      END                                                                QLDR7070
      COMPLEX FUNCTION Q1DAIP*16(DK,DR,DW,RHO,OFST,SGMA,XSTN,Q1CF,N,IPT)QLDR7075
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)                  QLDR7080
      DIMENSION Q1CF(13)                                                 QLDR7085
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFST,SGMA,XSTN,IPT         QLDR7090
  990 FORMAT('0',10X,'Q1DAIP ENTERED WITH:',/,                           QLDR7095
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFST',9X,'SGMA',9XQLDR7100
     2  ,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)                       QLDR7105
      XASTN = XSTN - OFST                                                QLDR7110
      IF(XASTN.LE.DR+RHO-1.0D0) GO TO 20                                 QLDR7115
      IOT = IPT - 1                                                      QLDR7120
      QCONST = CDEXP( DCMPLX(0.0D0,SGMA))                                QLDR7125
      A = DR-1.0D0                                                       QLDR7130
      B = XASTN - RHO - 1.0D-8                                           QLDR7135
      CALL Q1DZSM(DK,DR,DW,RHO,XSTN,A,B,N,QANS,Q1CF,IOT)                 QLDR7140
      Q1DAIP = QANS                                                      QLDR7145
      IF(IPT.LE.0) RETURN                                                QLDR7150
      GOTO 60                                                            QLDR7155
   20 Q1DAIP = DCMPLX(0.0D0,0.0D0)                                       QLDR7160
      IF(IPT.LE.0)RETURN                                                 QLDR7165
   60 WRITE (6,995) Q1DAIP                                               QLDR7170
  995 FORMAT('0',10X,'Q1DAIP = ',E14.7,',',E14.7)                        QLDR7175
      RETURN                                                             QLDR7180
      END                                                                QLDR7185
      COMPLEX FUNCTION Q1DRIP*16(DK,DR,DW,RHO,OFFSET,XSTN,Q1CF,N,IPT)    QLDR7190
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)                  QLDR7195
      DIMENSION Q1CF(13)                                                 QLDR7200
```

```
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,XSTN,IPT
  990 FORMAT('0',10X,'Q1DRIF ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,
     2  'XSTN',9X,'IPT'/,' ',10X,5(E12.5,','),I3)
      IF(XSTN.LE.DR+RHO+OFFSET-1.0D0) GOTO 20
      ICT = IPT - 1
      A = DR-1.0D0
      B = XSTN - RHO - 1.0D-8
      CALL Q1DZSM(DK,DR,DW,RHO,XSTN,A,B,N,QANS,Q1CF,IOT)
      Q1DRIP = -QANS
      IF(IPT.LE.0) RETURN
      GOTO 60
   20 Q1DRIP = DCMPLX(0.0D0,0.0D0)
      IF (IPT.LE.0) RETURN
   60 WRITE (6,995) Q1DRIP
  995 FORMAT('0',10X,'Q1DRIP = ',E14.7,',',E14.7)
      RETURN
      END
      SUBROUTINE Q1DZSM (DK,DR,DW,RHO,XSTN,A,B,J,QANS,Q1CF,IPT)
      IMPLICIT REAL*8(A - E,G,H,M,O,P,R - Y), COMPLEX*16(F,Q,Z)
      DIMENSION Q1CF(13)
      DIMENSION FV(5),LORR(60),F1T(60),F2T(60),F3T(60),DAT(60),
     1  ARESTT(60),QESTT(60),EPST(60),QPSUM(60)
      F(X) = QLGNDR(X,DR,J,Q1CF)*CDEXP(CEXP*(X))
     1       * MMBSJ1((OMEGA * DSQRT( (XSTN-X)*(XSTN-X) - YY)), IER)
     2       * OMEGA * (XSTN-X)/(DSQRT((XSTN-X)*(XSTN-X)-YY))
      GAMMA = 1.4D0
      YY = RHO * RHO
      DM2 = (GAMMA + 1.0D0) * DW
      OMEGA = DSQRT (DK*DK*(1.0D0-DM2)/(DM2*DM2))
      DLAMDA = DK/DM2
      DM = DSQRT(DM2)
      Q2EXP = CDEXP (DCMPLX (0.0D0, -DLAMDA*XSTN))
      CEXP = DCMPLX(0.0D0,DLAMDA)
      U  =  9.0D-13
      ACC = 1.0D-6
      IF (IPT.GT.0) WRITE(6,990)DK,DR,DW,RHO,XSTN,A,B,J,IPT
  99? FORMAT('0',15X,'Q1DZSM ENTERED WITH ARGUMENTS:',/,
     1' ',15X,'DK',13X,'DR',13X,'DW   ',10X,'RHO',12X,'XSTN',11X,
     2'A',14X,'B',14X,'J',2X,'IPT',/,
     3' ',15X,7(E14.7,','),I2,2X,I3)
      EFCURU = 4.0*U
      IFLAG = 1
      EPS = ACC
      CERROR = DCMPLX(0.0D0,0.0D0)
      LVL = 1
      LORR(LVL) = 1
      QPSUM(LVL) = 0.0
```

```
      ALPHA = A
      DA = B - A
      AREA = 0.0
      AREST = 0.0
      FV(1) = F(ALPHA)
      FV(3) = F(ALPHA + 0.5*DA)
      FV(5) = F(ALPHA + DA)
      KOUNT = 3
      WT = DA/6.0
      QEST = WT*(FV(1) + 4.0*FV(3) + FV(5))
    1 DX = 0.5*DA
      FV(2) = F(ALPHA + 0.5*DX)
      FV(4) = F(ALPHA + 1.5*DX)
      KOUNT = KOUNT + 2
      WT = DX/6.0
      QESTL = WT*(FV(1) + 4.0*FV(2) + FV(3))
      QESTR = WT*(FV(3) + 4.0*FV(4) + FV(5))
      QSUM = QESTL + QESTR
      ARESTL = WT*(CDABS(FV(1)) + CDABS(4.0*FV(2)) + CDABS(FV(3)))
      ARESTR = WT*(CDABS(FV(3)) + CDABS(4.0*FV(4)) + CDABS(FV(5)))
      AREA = AREA + ((ARESTL + ARESTR) - AREST)
      QDIFF = QEST - QSUM
      IF(CDABS(QDIFF).LE.EPS*DABS(AREA))GO TO 2
      IF(DABS(DX).LE.EPCURU*DABS(ALPHA))GO TO 5
      IF(LVL.GE.60)GO TO 5
      IF(KOUNT.GE.4000)GO TO 6
      LVL = LVL + 1
      LORR(LVL) = 0
      F1T(LVL) = FV(3)
      F2T(LVL) = FV(4)
      F3T(LVL) = FV(5)
      DA = DX
      DAT(LVL) = DX
      AREST = ARESTL
      ARESTT(LVL) = ARESTR
      QEST = QESTL
      QESTT(LVL) = QESTR
      EPS = EPS/1.4
      EPST(LVL) = EPS
      FV(5) = FV(3)
      FV(3) = FV(2)
      GO TO 1
    2 QERROR = QERROR + QDIFF/15.0
    3 IF(LORR(LVL).EQ.0)GO TO 4
      QSUM = QPSUM(LVL) + QSUM
      LVL = LVL - 1
      IF(LVL.GT.1)GO TO 3
      QANS = (QLGNDR(B,DR,J,Q1CF)*CDEXF(CEXP*B) - QSUM)*Q2EXP/DM
```

```
      IF(IPT.GT.0) GO TO 11
      IF(IFLAG.EQ.1) RETURN
      WRITE(6,990) DK,DR,DW,RHO,XSTN,A,E,J,IPT
   11 WRITE(6,995) QANS,IFLAG,IER,GERRCR
  995 FORMAT(' ',15X,'CIDZSN RESULTS: QANS = ',E14.7,',',E14.7,/,
     1' ',15X,'IFLAG',2X,'IER',5X,'GERRCR',/,
     2' ',15X,I3,2X,I3,5X,E14.7,',',E14.7)
      RETURN
    4 CFSUM(LVL) = QSUM
      LCRR(LVL) = 1
      ALPHA = ALPHA + DA
      DA = DAT(LVL)
      FV(1) = F1T(LVL)
      FV(3) = F2T(LVL)
      FV(5) = F3T(LVL)
      AREST = ARESTT(LVL)
      CEST = GESTT(LVL)
      EPS = EPST(LVL)
      GO TO 1
    5 IFLAG = 2
      GO TO 2
    6 IFLAG = 3
      GO TO 2
      END
```

7. Gorelov Spanning Function.

The subprogram for function used by Gorelov is shown below.

```
      REAL FUNCTICN PLGNDR*8(X,DR,N)
      IMPLICIT REAL*8(A-H,O-Z)
      IF(N.EQ.0) GOTO 100
      ETA = DARCOS(-X)
      ETASTR = DARCOS(1.0D0 - DR)
      FN = DFLOAT(N)
      PLGNDR = DCOS(FN*ETA)-DCOS(FN*ETASTR)
      RETURN
100   PLGNDR = 1.0D0
      RETURN
      END
```

8. Linear Expansion Program

The program based on the linear expansion for small k is shown below.

```
      IMPLICIT REAL*8(A-H,O,P,R-Y), COMPLEX*16(Q,Z)
      DIMENSION X(13), Q2PT(13), Q2CP(13)
C     BLOCK ONE   READ AND EDIT DATA
      CALL ERRSET(208,256,-1,1)
C     WRITE (6,910)
    1 CALL READ(DK,DR,DW,RHO,OFFSET,SIGMA,N,NF,IPT)
C     IF(IPT.GT.0)WRITE(6,990)DK,DR,DW,SIGMA,OFFSET,RHO,N,NF
  910 FORMAT ('1',5X,'GORELOV CASCADE PROGRAM LINEARIZED FOR SMALL DK')
  990 FORMAT('0',5X,'INPUT VALUES TRANSMITTED TO MAIN PROGRAM:',/,
     1' ',10X,'DK',13X,'DR',13X,'DW    ',10X,'SIGMA',10X,'OFFSET',/,
     1' ',10X,5(E12.5,3X),/,
     2'0',10X,'RHO',12X,'N',5X,'NFCN'
     3,/,' ',10X,1(E12.5,3X), I2,4X,I2)
      CALL ABSA(N,OFFSET,X,RHO,DW    )
C
C
C     BLOCK TWO    CALLS CALCULATION ROUTINES FOR ZONE ONE AND ZONE TWO
      CALL Q$ONE(DK,DR,DW,RHO,OFFSET,SIGMA,N,NF,IPT,X)
      GO TO 1
      END
      SUBROUTINE READ (DK,DR,DW,RHO,OFFSET,SIGMA,N,NF,IPT)
      IMPLICIT REAL*8(A-H, O, P, R-Y)
      READ(5,905) DK,DR,DW,RHP,OFFSET,SIGMA,N,NF,IPT
      IF (DK.LT.0.0D0) STOP
      IF (IPT.GT.0) WRITE (6,910)
      GAMMA = 1.4D0
      DR = RHP * DSQRT((GAMMA + 1.0D0) * DW)
      RHO = DR
      IF(N.LE.2) GO TO  1
      N = 2
C     WRITE(6,935 )
    1 IF(OFFSET.LE.1.9D0) GO TO 2
      WRITE(6,940)
      OFFSET = OFFSET-1.0D0
      GO TO 1
    2 CONTINUE
C     IF (IPT.GT.0) WRITE (6,925) DK,DR,DW,SIGMA,OFFSET,RHO,N,NF,IPT
      RETURN
  905 FORMAT(6F10.4,3I2)
  910 FORMAT ('1',10X,'GORELOV SLIGHTLY SUPERSONIC CASCADE PROGRAM',/,
     *       ' ',15X,'LINEARIZED FOR SMALL FREQUENCY, DK.')
  925 FORMAT ('0',10X,'DK',13X,'DR',13X,'DW    ',10X,'SIGMA',10X,'OFFSET'
     1,/,' ',10X,5(E12.5,3X),/,
     2'0',10X,'RHO',12X,'N',5X,'NFCN',2X,'IPT'
     3,/,' ',10X,1(E12.5,3X), I2,4X,I2,4X,I2)
  935 FORMAT('0',5X,'ORIGINAL N TOO LARGE (.GT.13) - RESET TO N = 13')
  940 FORMAT('0',5X,'OFFSET TOO LARGE (.GT.1.9), RESET AS OFFSET =
     1OFFSET - 1.0D0')
```

```
      END
      SUBROUTINE ABSA (N,OFFSET,X,DR,DW   )
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX*16(Q,Z)
      DIMENSION X(13)
      XINT = (2.0D0-DR)/DFLOAT(N+1)
      XL = DR - 1.0D0 + 1.0D-8
      DO 10 I = 1,N
      X(I) = XL + (XINT*DFLOAT(I))
      IF(X(I).EQ.0.0D0) X(I) = .1D-14
   10 CONTINUE
      RETURN
      END
      SUBROUTINE Q$ONE(DK,DR,DW,RHO,OFFSET,SIGMA,N,NF,IPT,X)
      IMPLICIT REAL*8(A-H,O,P,R-Y), COMPLEX*16(Z,Q)
      DIMENSION Q1COF(26), Q1INT(26,26), Q1RBP(13), Q1ABP(13)
      DIMENSION Q1RBCF(13), Q1ABCF(13)
      DIMENSION X(13)
C     IF(IPT.GT.0)WRITE(6,990)DK,DR,RHO,OFFSET,SIGMA,N,NF,IPT
      ICT = IPT - 1
  990 FORMAT('0  SUBROUTINE Q$ONE ENTERED WITH:',/,
     1' ',5X,'DK',13X,'DR',13X,'RHO',12X,'OFFSET',9X,'SIGMA',10X,
     2'N',3X,'NF',2X,'IPT',/,
     3  5X,5(E12.5,3X),I2,2X,I2,2X,I3)
      CALL Q1ZINT(DK,DR,DW,RHO,OFFSET,SIGMA,N,X,ICT,Q1COF,Q1INT)
      CALL Q1COEF(Q1COF,Q1INT,N,IOT,Q1ABCF,Q1RBCF)
      CALL Q1PPOT(DK,DR,DW,RHO,OFFSET,SIGMA,N,X,Q1ABCF,Q1RBCF,IOT)
      I O T = 0
      CALL Q1CCOF(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,IOT)
      RETURN
      END
C
C
C
      SUBROUTINE Q1ZINT(DK,DR,DW,RHO,OFFSET,SIGMA,N,X,
     1       IPT,Q1COF,Q1INT)
      IMPLICIT REAL * 8(A-H,O,P,R-Y), COMPLEX * 16(Z,Q)
      DIMENSION Q1COF(26), Q1INT(26,26), QINTRP(13), QINTAP(13)
      DIMENSION  X(13)
  990 FORMAT ('0',10X,'Q1ZINT ENTERED WITH:',/,
     1 ' ',10X,'DK',13X,'DR',13X,'RHO',12X,'OFFSET',9X,'SIGMA',10X,
     2  'N',2X,'IPT',/,
     3  ' ',10X,5(E12.5,3X),I2,2X,I3)
      QCONST = CDEXP(DCMPLX(0.0D0,SIGMA))
      IF (IPT.GT.0) WRITE (6,990) DK,DR,RHO,OFFSET,SIGMA,N,IPT
      ICT = IPT - 1
      GAMMA = 1.4D0
      CLAMDA = DK/((GAMMA+1.0D0)*DW)
      DO 90 I = 1,N
```

```
      IN = I + N
      XSTN = X(I)
      XC = XSTN - DR
      CEXP = CDEXP(DCMPLX(C.ODO,DLAMDA*XSTN))
      CALL Q1CAIP(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,QINTAP,N,IOT)
      CALL Q1CRIP(DK,DR,DW,RHC,OFFSET,XSTN,QINTRP,N,ICT)
      DO 20 J = 1,N
       JN = N + J
      J1 = J-1
      Q1INT(I,1) = DCMPLX(1.0D0,DLAMDA*XSTN)
      TEMP = DR-XSTN-1.0D0
      Q1INT(I,2) = DCMPLX(TEMP,DLAMDA*XSTN*TEMP)
      Q1INT(IN,J) = QINTRP(J)
      Q1INT(I,JN) =  QINTAP(J)
      XASTN = XSTN - OFFSET
      Q1INT(IN,3) = DCMPLX(1.0D0,DLAMDA*XASTN)
      TEMP = DR-XASTN-1.0D0
      Q1INT(IN,4) = DCMPLX(TEMP,DLAMDA*XASTN*TEMP)
   20 CONTINUE
      Q1COF(IN) = -Q1CRBP(DK,DR,DW,RHO,XSTN,IOT)
      Q1COF(I)  = -Q1CABP(DK,DR,DW,RHC,OFFSET,SIGMA,XSTN,IOT)
   90 CONTINUE
      RETURN
      END
      COMPLEX FUNCTION Q1CRBP*16(DK,DR,DW,RHO,XSTN,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION QINP(2)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,XSTN,IPT
  990 FORMAT('0',10X,'Q1CRBP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'XSTN',9X,'IPT'
     2  ,/,' ',10X,5(E12.5,','),I3)
      IF(XSTN.LE.RHO-1.0D0) GOTO 20
      IF(XSTN.GT.2.0D0) GOTO 20
      ICT = IPT - 1
      QDK = DCMPLX (0.0D0,DK)
      GAMMA = 1.4D0
      DLAMDA = DK/((GAMMA+1.0D0)*DW)
      Q1CRBP = DCMPLX(1.0D0,(DK+DLAMDA)*(XSTN-DR))
      IF(IPT.LE.0) RETURN
      GOTO 30
   20 Q1CRBP = DCMPLX(0.0D0,0.0D0)
      IF(IPT.LE.0) RETURN
   30 WRITE (6,995) Q1CRBP
  995 FORMAT('0',10X,'Q1CRBP = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1CABP*16(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
```

```
      DIMENSION QINP(2)
      IF(IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT
  990 FORMAT('0',10X,'Q1CABP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFFSET',7X,
     2  'SIGMA',8X,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)
      XASTN = XSTN - OFFSET
      IF(XASTN.LE.RHO-1.0D0) GOTO 20
      IF(XASTN.GT.2.0D0) GOTO 20
      IOT = IPT - 1
      CDK = DCMPLX (0.0D0,DK)
      CCONST = CDEXP( DCMPLX(0.0D0,SIGMA))
      GAMMA = 1.4D0
      DLAMDA = DK/((GAMMA+1.0D0)*DW)
      Q1CABP = DCMPLX(1.0D0,(DK+DLAMDA)*(XASTN-DR))*QCONST
      IF(IPT.LE.0) RETURN
      GO TO 30
   20 Q1CABP = DCMPLX(0.0D0,0.0D0)
      IF (IPT.LE.0) RETURN
   30 WRITE (6,995) Q1CABP
  995 FORMAT('0',10X,'Q1CABP = ',E14.7,',',E14.7)
      RETURN
      END
      SUBROUTINE Q1CAIP(DK,DR,DW,RHO,OFST,SGMA,XSTN,QCAIP,N,IPT)
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION  QINP(13), QCAIP(13)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFST,SGMA,XSTN,IPT
  990 FORMAT('0',10X,'Q1CAIP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFST',9X,'SGMA',9X
     2  ,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)
      XASTN = XSTN - OFST
      IF(XASTN.LE.DR+RHO-1.0D0) GO TO 20
      IF(XASTN.GT.2.0D0) GOTO 20
      IOT = IPT - 1
      QCONST = CDEXP( DCMPLX(0.0D0,SGMA))
      GAMMA = 1.4D0
      DLAMDA = DK/((GAMMA+1.0D0)*DW)
      QCAIP(1) = DCMPLX(1.0D0,DLAMDA*(XASTN-RHO))
      TEMP = DR+RHO-XASTN-1.0D0
      QCAIP(2)  = DCMPLX(TEMP,DLAMDA*(XASTN-RHO)*TEMP)
      IF(IPT.LE.0) RETURN
      GOTO 30
   20 ZERO = DCMPLX(0.0D0,0.0D0)
      DO 25 I = 1,N
      QCAIP(I) = ZERO
   25 CONTINUE
      IF(IPT.LE.0)RETURN
   30 WRITE (6,995)
  995 FORMAT ('0',10X,'Q1CAIP RESULTS: J     QCAIP(J)')
```

```
      DO 40 J = 1,N
      WRITE (6,996) J,QCRIP(J)
  996 FORMAT(' ',26X,I2,3X,E14.7,',',E14.7)
   40 CONTINUE
      RETURN
      END
      SUBROUTINE  Q1CRIP (DK,DR,DW,RHO,OFFSET,XSTN,QCRIP,N,IPT)
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION  QCRIP(13)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,XSTN,IPT
  990 FORMAT('0',10X,'Q1CRIP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,
     2  'XSTN',9X,'IPT'/,' ',10X,5(E12.5,','),I3)
      IF(XSTN.LE.DR+RHO+OFFSET-1.0D0) GOTO 20
      IF(XSTN.GT.2.0D0) GOTO 20
      IOT = IPT - 1
      GAMMA = 1.4D0
      DLAMDA = DK/((GAMMA+1.0D0)*DW)
      QCRIP(1) = DCMPLX(1.0D0,DLAMDA*(XSTN-RHO))
      TEMP = DR+RHO-XSTN-1.0D0
      QCRIP(2)  = DCMPLX(TEMP,DLAMDA*(XSTN-RHO)*TEMP)
      IF(IPT.LE.0) RETURN
      GOTO 30
   20 ZERO = DCMPLX(0.0D0,0.0D0)
      DO 25 I = 1,N
   25 QCRIP(I) = ZERO
      IF (IPT.LE.0) RETURN
   30 WRITE (6,995)
  995 FORMAT ('0',10X,'Q1CRIP RESULTS: J     QCRIP(J)')
      DO 40 J = 1,N
      WRITE (6,996) J,QCRIP(J)
  996 FORMAT(' ',26X,I2,3X,E14.7,',',E14.7)
   40 CONTINUE
      RETURN
      END
      SUBROUTINE Q1COEF( Q1COF,Q1INT,N,IPT,Q1ABCF,Q1RBCF)
      IMPLICIT REAL * 8(A-H,O,P,R-Y), COMPLEX * 16 (Z,Q)
      DIMENSION Q1COF(26), Q1INT(26,26), ZWA(300)
      DIMENSION Q1ABCF(13),Q1RBCF(13)
      IF (IPT.GE.0) WRITE (6,90)
      IB = 26
      M = 1
      N2 = 2*N
      IF (IPT.LE.0) GO TO 5
      WRITE (6,98) N,N2
   98 FORMAT('0',5X,'Q1COEF ENTERED WITH ',I2,' DEG  PWR SERIES (',I2,
     1 ' SQUARE MATRIX)')
      DO 2 I = 1,N2
```

```
      WRITE(6,92) I, I, Q1COF(I)
   92 FORMAT('0',10X,'Q1COEF EQUATION SYSTEM, ROW ',I2,/,
     1' ',10X,'Q1COF(', I2,') = ', E14.7,',',E14.7)
      DO 2 J = 1,N
      J2=J+N
      WRITE(6,91) I,J,Q1INT(I,J),I,J2,Q1INT(I,J2)
   91 FORMAT(' ',15X,2('QINT(',I2,',',I2,') = ',E14.7,',',E14.7,10X))
    2 CONTINUE
    5 IA = 26
      IJOB=0
      CALL LEQT1C(Q1INT,N2,IA,Q1COF,M,IE,IJOB,ZWA,IER)
      IF(IER.EQ.0) GOTO 30
      IF (IER.EQ.129) GO TO 10
      WRITE(6,93)
   93 FORMAT( '0',10X,'Q1COEF - ITERATIVE IMPROVEMENT FAILED, MATRIX TOO
     1 ILL-CONDITIONED. USE RESULTS WITH CAUTION.')
      GO TO 30
   10 WRITE (6,95)
   95 FORMAT('0',10X,'Q1COEF - MATRIX ALGORITHMICALLY SINGULAR.  COEFFIC
     1IENTS SET TO ZERO.')
      ZERO = DCMPLX(0.0D0,0.0D0)
      DO 20 I = 1,N
      I2 = I+N
      Q1COF(I) = ZERO
      Q1COF(I2) = ZERO
   20 CONTINUE
   30 CONTINUE
      DO 35 I = 1,N
      IN = I + N
      Q1ABCF(I) = Q1COF(IN)
      Q1RBCF(I) = Q1COF(I)
   35 CONTINUE
      IF(IPT.LE.0) RETURN
      IF (IPT.GE.0) WRITE (6,94)
      DO 40 I = 1,N
      IM1 = I-1
      WRITE (6,99) I, IM1, Q1RBCF(I), Q1ABCF(I)
   40 CONTINUE
   99 FORMAT('0',5X,I2,10X,I2,10X,2(E14.7,5X,E14.7,10X))
   90 FORMAT( '0',10X,'SUBROUTINE Q1COEF -  COMPLEX POWER SERIES COEFFIC
     1IENTS')
   94 FORMAT( '0',5X,'INDEX',7X,'DEG POLY',4X,'REFERENCE BLADE TERMS Q1C
     3OF(INDEX)',7X,'ADJACENT BLADE TERMS Q1COF(2*INDEX)')
      RETURN
      END
      SUBROUTINE Q1PPOT(DK,DR,DW,RHO,OFFSET,SIGMA,N,X,Q1ABCF,Q1RBCF,IPT)
      IMPLICIT REAL * 8(A-H,O,P,R-Y), COMPLEX * 16(Z,Q)
      DIMENSION Q1ABCF(13), Q1RBCF(13), CRR(13), CAA(13), CPHI(13)
```

```
      DIMENSION QDRR(13),QDAA(13),QDPHI(13)
      DIMENSION X(13)
      IF(IPT.LE.0) RETURN
      WRITE (6,990) DK,DR,DW,RHO,OFFSET,SIGMA,N,IPT
  990 FORMAT('1',10X,'Q1PPOT -CALCULATION OF COMPLEX POTENTIALS FOR ZONE
     1 ONE',/,'0',10X,'THESE CALCULATIONS ARE BASED ON THE SPANNING FUNC
     2TION COEFFICIENTS GIVEN DIRECTLY ABOVE',/
     3 ,' ',10X,'DK',13X,'DR',13X,'DW',13X,'RHO',12X,'OFFSET',9X,'SIGMA'
     4 ,10X,'N',3X,'IPT',/,' ',10X,
     56(E12.5,3X),I2,2X,I3)
   90 FORMAT ('0 X-STATION NUMBER ',I2,', XSTN = ',F6.4,/,
     1'   BL TOTAL POTENTIAL',14X,'REF BL POTENTIAL',8X,
     2 'ADJ BL POTENTIAL',8X,'REF BL INT POTENTIAL',5X,
     3 'ADJ BL INT POTENTIAL')
   91 FORMAT('  R  ',2(E14.7,',',),8(E11.4,',',))
   92 FORMAT('  A  ',2(E14.7,',',),8(E11.4,',',))
      IOT = IPT - 1
      RZERO = 0.0D0
      DO 10 I = 1,N
      XSTN = X(I)
      QRO = Q1PRBP(DK,DR,DW,RZERO,XSTN,IOT)
      QRP = Q1PRBP(DK,DR,DW,RHO,XSTN,IOT)
      QAO = Q1PABP(DK,DR,DW,RZERO,OFFSET,SIGMA,XSTN,IOT)
      QAP = Q1PABP(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IOT)
      QRIPO = Q1PRIP(DK,DR,DW,RZERO,OFFSET,XSTN,Q1RBCF,N,IOT)
      QRIPP = Q1PRIP(DK,DR,DW,RHO,OFFSET,XSTN,Q1RBCF,N,IOT)
      QAIPO = Q1PAIP(DK,DR,DW,RZERO,OFFSET,SIGMA,XSTN,Q1ABCF,N,IOT)
      QAIPP = Q1PAIP(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,Q1ABCF,N,IOT)
      QR = QRO + QAP + QRIPO + QAIPP
      QA = QRP + QAO + QRIPP + QAIPO
      QPHI(I)=Q1PPHI(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,XSTN,IOT)
      WRITE (6,90) I,XSTN
      WRITE(6,91) QR,QRO,QAP,QRIPO,QAIPP
      WRITE(6,92) QA,QRP,QAO,QRIPP,QAIPO
      QRR(I) = QR
      QAA(I) = QA
      QDRO = Q1DRBP(DK,DR,DW,RZERO,XSTN,IOT)
      QDRP = Q1DRBP(DK,DR,DW,RHO,XSTN,IOT)
      QDAO = Q1DABP(DK,DR,DW,RZERO,OFFSET,SIGMA,XSTN,IOT)
      QDAP = Q1DABP(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IOT)
      QDRIPO = Q1DRIP(DK,DR,DW,RZERO,OFFSET,XSTN,Q1RBCF,N,IOT)
      QDRIPP = Q1DRIP(DK,DR,DW,RHO,OFFSET,XSTN,Q1RBCF,N,IOT)
      QDAIPO = Q1DAIP(DK,DR,DW,RZERO,OFFSET,SIGMA,XSTN,Q1ABCF,N,IOT)
      QDAIPP = Q1DAIP(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,Q1ABCF,N,IOT)
      QDR = QDRO + QDAP + QDRIPO + QDAIPP
      QDA = QDRP + QDAO + QDRIPP + QDAIPO
      QDPHI(I)=Q1DPHI(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,
     1       XSTN,IOT)
```

```
      WRITE (6,95) I,XSTN
      WRITE(6,91) QDR,QDRO,QDAP,QDRIPO,QDAIPP
      WRITE(6,92) QDA,QDRP,QDAO,QDRIPP,QDAIPO
      QDRR(I) = QDR
      QDAA(I) = QDA
   95 FORMAT ('0 X-STATION NUMBER ',I2,', XSTN = ',F6.4,/,
     1'   BL TOTAL D(POT)/DX',14X,'REF BL D(POT)/DX',8X,
     2 'ADJ BL D(POT)/DX',8X,'REF BL INT D(POT)/DX',5X,
     3 'ADJ BL INT D(POT)/DX')
   10 CONTINUE
      WRITE(6,994)
  994 FORMAT ('1',10X,'SUMMARY LISTING',/,
     1'0',10X,'XSTN',7X,'SINGLE BLADE TOTAL POTENTIAL',6X,
     1  'REF BLADE POTENTIAL',15X,'ADJ BLADE POTENTIAL')
      DO 20 I = 1,N
      XSTN = X(I)
      WRITE(6,94) XSTN,CPHI(I),QRR(I),QAA(I)
   94 FORMAT(' ',10X,F6.4,3(5X,E14.7,',',E14.7))
   20 CONTINUE
      WRITE(6,996)
  996 FORMAT('0',10X,'XSTN',7X,'SINGLE BLADE TOTAL D(POT)/DX',6X,
     1  'REF BLADE D(POT)/DX',15X,'ADJ BLADE D(POT)/DX')
      DO 50 I = 1,N
      XSTN = X(I)
      WRITE(6,94) XSTN,CDPHI(I),QDRR(I),QDAA(I)
   50 CONTINUE
      RETURN
      END
      SUBROUTINE Q1DCOF(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION Q1ABCF(13), Q1RBCF(13)
      IF(IPT.GT.0)WRITE (6,990) DK,DR,DW,RHO,OFFSET,SIGMA,N,IPT
  990 FORMAT('1',10X,'Q1DCOF - CALCULATION OF COMPLEX DIMENSIONLESS AERO
     1DYNAMIC COEFFICIENTS',/,
     3 '0',10X,'DK',13X,'DR',13X,'DW',13X,'RHO',12X,'OFFSET',9X,'SIGMA'
     4 ,10X,'N',3X,'IPT',/,' ',10X,
     56(E12.5,3X),I2,2X,I3)
      IOT = IPT - 3
  999 QDCL = Q1DCL(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,IOT)
      QDCM = Q1DCM(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,IOT)
      GAMMA = 1.4D0
      TAU = (2.0D0*DSQRT((GAMMA + 1.0D0)*DW))/DR
      WRITE (6,90) DK,TAU,DW,N,SIGMA,QDCL,QDCM
   90 FORMAT('0',5X,'DK = ',F6.3,', TAU = ',F7.4,', DW = ',F6.3,', N = '
     1     ,I2,', SIGMA = ',F6.3,', CL = ',F9.4,',',F9.4,', CM = ',
     2      F9.4,',',F9.4)
      RETURN
      END
```

```
      COMPLEX FUNCTION Q1DCL*16(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBC
     1F,IPT)
      IMPLICIT REAL*8(A - E,G,H,O,P,R - Y), COMPLEX*16(F,Q,Z)
      DIMENSION Q1ABCF(13), Q1RBCF(13)
      DIMENSION FV(5),F1T(60),F2T(60),F3T(60),QESTT(60),CPSUM(60)
      DIMENSION DAT(60),   ARESTT(60),EPST(60)
      DIMENSION LORR(60)
      F(X) = (Q1DPHI(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,X,ICT)
     1   -QALPHA*Q1PHI(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,X,ICT))
      IF (IPT.GT.0) WRITE (6,990) DK,DR,DW,RHO,OFFSET,SIGMA,N,IPT
  990 FORMAT('0',10X,'Q1DCL ENTERED WITH:',/,
     3 ' ',10X,'DK',13X,'DR',13X,'DK',13X,'RHO',12X,'OFFSET',9X,'SIGMA'
     4 ,10X,'N',3X,'IPT',/,' ',10X,
     56(E12.5,3X),I2,2X,I3)
      A = - 1.0D0
      B = 1.0D0
      ICT = IPT - 1
      GAMMA = 1.4D0
      DLAMDA = DK/((GAMMA + 1.0D0) * DW)
      QALPHA = DCMPLX(0.0D0,DLAMDA-DK)
      U  =   9.0E - 13
      ACC = 1.0D-5
      EFOURU = 4.0*U
      IFLAG = 1
      EPS = ACC
      QERROR = DCMPLX(0.0D0,0.0D0)
      LVL = 1
      LORR(LVL) = 1
      CPSUM(LVL) = 0.0
      ALPHA = A
      DA = B - A
      AREA = 0.0
      AREST = 0.0
      FV(1) = F(ALPHA)
      FV(3) = F(ALPHA + 0.5*DA)
      FV(5) = F(ALPHA + DA)
      KOUNT = 3
      WT = DA/6.0
      QEST = WT*(FV(1) + 4.0*FV(3) + FV(5))
    1 DX = 0.5*DA
      FV(2) = F(ALPHA + 0.5*DX)
      FV(4) = F(ALPHA + 1.5*DX)
      KOUNT = KOUNT + 2
      WT = DX/6.0
      QESTL = WT*(FV(1) + 4.0*FV(2) + FV(3))
      QESTR = WT*(FV(3) + 4.0*FV(4) + FV(5))
      QSUM = QESTL + QESTR
      ARESTL = WT*(CDABS(FV(1)) + CDABS(4.0*FV(2)) + CDABS(FV(3)))
```

```
      ARESTR = WT*(CDABS(FV(3)) + CCABS(4.0*FV(4)) + CDABS(FV(5)))
      AFEA = AREA + ((ARESTL + ARESTR) - AREST)
      QCIFF = QEST - QSUM
      IF(CCABS(QCIFF).LE.EPS*DABS(AREA))GO TO 2
      IF(DABS(DX).LE.EFCURU*DABS(ALPHA))GO TO 5
      IF(LVL.GE.60)GO TO 5
      IF(KCUNT.GE.4000)GO TO 6
      LVL = LVL + 1
      LCRR(LVL) = 0
      F1T(LVL) = FV(3)
      F2T(LVL) = FV(4)
      F3T(LVL) = FV(5)
      DA = DX
      DAT(LVL) = DX
      AREST = ARESTL
      ARESTT(LVL) = ARESTR
      QEST = QESTL
      QESTT(LVL) = QESTR
      EPS = EPS/1.4
      EPST(LVL) = EPS
      FV(5) = FV(3)
      FV(3) = FV(2)
      GC TO 1
2     CERROR = QERROR + QDIFF/15.0
3     IF(LORR(LVL).EQ.0)GO TO 4
      QSUM = QPSUM(LVL) + CSUM
      LVL = LVL - 1
      IF(LVL.GT.1)GO TO 3
      Q1DCL = CSUM * 2.0D0
      IF(IPT.GT.0) GO TC 11
      IF(IFLAG.EQ.1) RETURN
      WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,N,IPT
   11 WFITE(6,995) Q1DCL,IFLAG,CERROR
995   FCRMAT(' ',15X,'RESULTS: Q1DCL = ',E14.7,',',E14.7,/,
     1' ',15X,'IFLAG',2X,'QERROR',/,
     2' ',15X,I3,2X,E14.7,',',E14.7)
      RETURN
4     QFSUM(LVL) = QSUM
      LCRR(LVL) = 1
      ALPHA = ALPHA + DA
      DA = DAT(LVL)
      FV(1) = F1T(LVL)
      FV(3) = F2T(LVL)
      FV(5) = F3T(LVL)
      AREST = ARESTT(LVL)
      CEST = QESTT(LVL)
      EPS = EPST(LVL)
      GC TO 1
```

```
5     IFLAG = 2
      GO TO 2
6     IFLAG = 3
      GO TO 2
      END
      COMPLEX FUNCTION Q1DCM*16(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBC
     1F,IPT)
      IMPLICIT REAL*8(A - E,G,H,O,P,R - Y), COMPLEX*16(F,Q,Z)
      DIMENSION Q1ABCF(13), Q1RBCF(13)
      DIMENSION FV(5),F1T(60),F2T(60),F3T(60),QESTT(60),QPSUM(60)
      DIMENSION DAT(60),  ARESTT(60),EPST(60)
      DIMENSION LORR(60)
      F(X) = X*(Q1DPHI(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,X,ICT)
     1    -QALPHA*Q1PHI(DK,DR,DW,RHO,OFFSET,SIGMA,N,Q1ABCF,Q1RBCF,X,ICT))
      IF (IPT.GT.0) WRITE (6,990) DK,DR,DW,RHO,OFFSET,SIGMA,N,IPT
990   FORMAT('0',10X,'Q1DCM ENTERED WITH:',/,
     3 ' ',10X,'DK',13X,'DR',13X,'DK',13X,'RHO',12X,'OFFSET',9X,'SIGMA'
     4 ,10X,'N',3X,'IPT',/,' ',10X,
     56(E12.5,3X),I2,2X,I3)
      A = - 1.0D0
      B = 1.0D0
      ICT = IPT - 1
      GAMMA = 1.4D0
      DLAMDA = DK/((GAMMA + 1.0D0) * DW)
      QALPHA = DCMPLX(0.0D0,DLAMDA-DK)
      U  =  9.0E - 13
      ACC = 1.0D-5
      EPSURU = 4.0*U
      IFLAG = 1
      EPS = ACC
      QERROR = DCMPLX(0.0D0,0.0D0)
      LVL = 1
      LORR(LVL) = 1
      QPSUM(LVL) = 0.0
      ALPHA = A
      DA = B - A
      AREA = 0.0
      AREST = 0.0
      FV(1) = F(ALPHA)
      FV(3) = F(ALPHA + 0.5*DA)
      FV(5) = F(ALPHA + DA)
      KOUNT = 3
      WT = DA/6.0
      QEST = WT*(FV(1) + 4.0*FV(3) + FV(5))
1     DX = 0.5*DA
      FV(2) = F(ALPHA + 0.5*DX)
      FV(4) = F(ALPHA + 1.5*DX)
      KOUNT = KOUNT + 2
```

```
      WT = DX/6.0
      QESTL = WT*(FV(1) + 4.0*FV(2) + FV(3))
      QESTR = WT*(FV(3) + 4.0*FV(4) + FV(5))
      QSUM = QESTL + QESTR
      ARESTL = WT*(CDABS(FV(1)) + CDABS(4.0*FV(2)) + CDABS(FV(3)))
      ARESTR = WT*(CDABS(FV(3)) + CDABS(4.0*FV(4)) + CDABS(FV(5)))
      AREA = AREA + ((ARESTL + ARESTR) - AREST)
      QDIFF = QEST - QSUM
      IF(CDABS(QDIFF).LE.EPS*DABS(AREA))GO TO 2
      IF(DABS(DX).LE.EPOLRU*DABS(ALPHA))GO TO 5
      IF(LVL.GE.60)GO TO 5
      IF(KOUNT.GE.4000)GO TO 6
      LVL = LVL + 1
      LORR(LVL) = 0
      F1T(LVL) = FV(3)
      F2T(LVL) = FV(4)
      F3T(LVL) = FV(5)
      DA = DX
      DAT(LVL) = DX
      AREST = ARESTL
      ARESTT(LVL) = ARESTR
      QEST = QESTL
      QESTT(LVL) = QESTR
      EPS = EPS/1.4
      EPST(LVL) = EPS
      FV(5) = FV(3)
      FV(3) = FV(2)
      GO TO 1
2     QERROR = QERROR + QDIFF/15.0
3     IF(LORR(LVL).EQ.0)GO TO 4
      QSUM = QPSUM(LVL) + QSUM
      LVL = LVL - 1
      IF(LVL.GT.1)GO TO 3
      Q1DCM = QSUM * 2.0D0
      IF(IPT.GT.0) GO TO 11
      IF(IFLAG.EQ.1) RETURN
      WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,N,IPT
   11 WRITE(6,995) Q1DCM,IFLAG,QERROR
  995 FORMAT(' ',15X,'RESULTS: Q1DCM = ',E14.7,',',E14.7,/,
     1' ',15X,'IFLAG',2X,'QERROR',/,
     2' ',15X,I3,2X,E14.7,',',E14.7)
      RETURN
4     QPSUM(LVL) = QSUM
      LORR(LVL) = 1
      ALPHA = ALPHA + DA
      DA = DAT(LVL)
      FV(1) = F1T(LVL)
      FV(3) = F2T(LVL)
```

```
      FV(5) = F3T(LVL)
      AREST = ARESTT(LVL)
      QEST = QESTT(LVL)
      EPS = EPST(LVL)
      GO TO 1
5     IFLAG = 2
      GO TO 2
6     IFLAG = 3
      GO TO 2
      END
      COMPLEX FUNCTION Q1PHI*16 (DK,DR,DW,RHO,OFFSET,SIGMA,
     1      N,Q1ABCF,Q1RBCF,XSTN,IPT)
      IMPLICIT REAL*8(A-H,O,P,R-Y), COMPLEX*16(Q,Z)
      DIMENSION Q1ABCF(13), Q1RBCF(13)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT
990   FORMAT('0',10X,'Q1PHI ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFFSET',7X,'SIGMA'
     2  ,8X,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)
      RZERO = 0.0D0
      IOT = IPT - 1
      Q1PHI = Q1PRBP(DK,DR,DW,RZERO,XSTN,IOT)
     1 + Q1PRIP(DK,DR,DW,RZERO,OFFSET,XSTN,Q1RBCF,N,IOT)
     2 + Q1PABP(DK,DR,DW,DR,OFFSET,SIGMA,XSTN,IOT)
     3 + Q1PAIP(DK,DR,DW,DR,OFFSET,SIGMA,XSTN,Q1ABCF,N,IOT)
      IF(IPT.LE.0) RETURN
      WRITE (6,995) Q1PHI
995   FORMAT('0',10X,'Q1PHI = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1PRBP*16(DK,DR,DW,RHO,XSTN,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION QINP(2)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,XSTN,IPT
990   FORMAT('0',10X,'Q1PRBP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'XSTN',9X,'IPT'
     2  ,/,' ',10X,5(E12.5,','),I3)
      IF(XSTN.LE.RHO-1.0D0) GOTO 20
      IOT = IPT - 1
      GAMMA = 1.4D0
      DM2 = (GAMMA + 1.0D0) * DW
      DLAMDA = DK/DM2
      PR = 1.0+XSTN-RHO
      PIMAG = (DK+DLAMDA)*((XSTN-RHO)*(XSTN-RHO) - 1.0D0)
      Q1PRBP = - DCMPLX(PR,PIMAG-DLAMDA*XSTN*PR)/DSQRT(DM2)
      IF(IPT.LE.0) RETURN
      GOTO 60
20    Q1PRBP = DCMPLX(0.0D0,0.0D0)
      IF(IPT.LE.0) RETURN
```

```
   60 WRITE (6,995) Q1PRBP
  995 FORMAT('0',10X,'Q1PRBP = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1PABP*16(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION QINP(2)
      IF(IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT
  990 FORMAT('0',10X,'Q1PABP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFFSET',7X,
     2  'SIGMA',8X,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)
      XASTN = XSTN - OFFSET
      IF(XASTN.LE.RHO-1.0D0) GOTO 20
      ICT = IPT - 1
      CCONST = CDEXP( DCMPLX(0.0D0,SIGMA))
      GAMMA = 1.4D0
      DM2 = (GAMMA + 1.0D0) * DW
      DLAMDA = DK/DM2
      PR = 1.0D0 + XASTN - RHO
      PIMAG = (DK+DLAMDA)*((XASTN-RHO)*(XASTN-RHO) - 1.0D0)
      Q1PABP = DCMPLX(PR,PIMAG-DLAMDA*XASTN*PR)*(CCONST/DSQRT(DM2))
      IF(IPT.LE.0) RETURN
      GO TO 60
   20 Q1PABP = DCMPLX(0.0D0,0.0D0)
      IF (IPT.LE.0) RETURN
   60 WRITE (6,995) Q1PABP
  995 FORMAT('0',10X,'Q1PABP = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1DPHI*16 (DK,DR,DW,RHO,OFFSET,SIGMA,
     1    N,Q1ABCF,Q1RBCF,XSTN,IPT)
      IMPLICIT REAL*8(A-H,O,P,R-Y), COMPLEX*16(Q,Z)
      DIMENSION Q1ABCF(13),Q1RBCF(13)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT
  990 FORMAT('0',10X,'Q1DPHI ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFFSET',7X,'SIGMA'
     2  ,8X,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)
      RZERO = 0.0D0
      ICT = IPT - 1
      Q1DPHI = Q1CRBP(DK,DR,DW,RZERO,XSTN,IOT)
     1 + Q1DRIP(DK,DR,DW,RZERO,OFFSET,XSTN,Q1RBCF,N,IOT)
     2 + Q1DABP(DK,DR,DW,DR,OFFSET,SIGMA,XSTN,IOT)
     3 + Q1DAIP(DK,DR,DW,DR,OFFSET,SIGMA,XSTN,Q1ABCF,N,IOT)
      IF(IPT.LE.0) RETURN
      WRITE (6,995) Q1DPHI
  995 FORMAT('0',10X,'Q1DPHI = ',E14.7,',',E14.7)
      RETURN
      END
```

```
      COMPLEX FUNCTION Q1DRBP*16(DK,DR,DW,RHO,XSTN,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION QINP(2)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,XSTN,IPT
  990 FORMAT('0',10X,'Q1DRBP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'XSTN',9X,'IPT'
     2  ,/,' ',10X,5(E12.5,','),I3)
      IF(XSTN.LE.RHO-1.0D0) GOTO 20
      ICT = IPT - 1
      GAMMA = 1.4D0
      DM2 = (GAMMA + 1.0D0) * DW
      DLAMDA = DK/DM2
      DM2 = (GAMMA + 1.0D0) * DW
      PIMAG = (DK+DLAMDA)*(XSTN-RHO)
      Q1DRBP = -DCMPLX(1.0D0,PIMAG-DLAMDA*XSTN)/DSQRT(DM2)
      IF(IPT.LE.0) RETURN
      GOTO 60
   20 Q1DRBP = DCMPLX(0.0D0,0.0D0)
      IF(IPT.LE.0) RETURN
   60 WRITE (6,995) Q1DRBP
  995 FORMAT('0',10X,'Q1DRBP = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1DABP*16(DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT)
      IMPLICIT REAL * 8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION QINP(2)
      IF(IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFFSET,SIGMA,XSTN,IPT
  990 FORMAT('0',10X,'Q1DABP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFFSET',7X,
     2  'SIGMA',8X,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)
      XASTN = XSTN - OFFSET
      IF(XASTN.LE.RHO-1.0D0) GOTO 20
      ICT = IPT - 1
      QCONST = CDEXP( DCMPLX(0.0D0,SIGMA))
      GAMMA = 1.4D0
      DM2 = (GAMMA + 1.0D0) * DW
      DLAMDA = DK/DM2
      PIMAG = (DK+DLAMDA)*(XASTN-RHO)
      Q1DABP =  DCMPLX(1.0D0,PIMAG-DLAMDA*XASTN)*QCONST/DSQRT(DM2)
      IF(IPT.LE.0) RETURN
      GO TO 60
   20 Q1DABP = DCMPLX(0.0D0,0.0D0)
      IF (IPT.LE.0) RETURN
   60 WRITE (6,995) Q1DABP
  995 FORMAT('0',10X,'Q1DABP = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1PAIP*16(DK,DR,DW,RHO,OFST,SGMA,XSTN,Q1DF,N,IPT)
```

```
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX * 16 (C,Z)
      DIMENSION Q1CF(13)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFST,SGMA,XSTN,IPT
  990 FORMAT('0',10X,'Q1PAIP ENTERED WITH:',/,
     1 ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFST',9X,'SGMA',5X
     2 ,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)
      XASTN = XSTN - OFST
      IF(XASTN.LE.DR+RHO-1.0D0) GO TO 20
      CCONST = CDEXP( DCMPLX(0.0D0,SGMA))
      GAMMA = 1.4D0
      DM2 = (GAMMA+1.0D0)*DW
      DLAMDA = DK/DM2
      XX=XASTN-RHO
      Q1 = Q1CF(1)
      Q2 = Q1CF(2)
      PR1 = XX+1.0D0-DR
      PR2 = (DR-1.0D0 - XX/2.0D0)*XX - (DR-1.0D0)*(DR-1.0D0)/2.0D0
      PIMAG1 = (XX*XX+(2.0D0-DR)*DR-1.0D0)*DLAMDA/2.0D0
      PIMAG2 =  XX*XX*((DR-1.0D0)/2.0D0-XX)-((DR-1.0D0)**3)/6.0D0
      PIMAG1=PIMAG1*DLAMDA
      PIMAG2 = PIMAG2 * DLAMDA
      QP1 = DCMPLX(PR1,PIMAG1-DLAMDA*PR1*XASTN)
      QP2 = DCMPLX(PR2,PIMAG2-DLAMDA*PR2*XASTN)
      Q1PAIP = (Q1*QP1 + Q2*QP2)/CDSQRT(DM2)
      IF(IPT.LE.0) RETURN
      GOTO 30
   20 Q1PAIP = DCMPLX(0.0D0,0.0D0)
      IF(IPT.LE.0)RETURN
   30 WRITE (6,995) Q1PAIP
  995 FORMAT('0',10X,'Q1PAIP = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1PRIP*16(DK,DR,DW,RHO,OFFSET,XSTN,Q1CF,N,IPT)
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION Q1CF(13)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,XSTN,IPT
  990 FORMAT('0',10X,'Q1PRIP ENTERED WITH:',/,
     1 ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,
     2 'XSTN',9X,'IPT'/,' ',10X,5(E12.5,','),I3)
      IF(XSTN.LE.DR+RHO+OFFSET-1.0D0) GOTO 20
      GAMMA = 1.4D0
      DM2 = (GAMMA+1.0D0)*DW
      DLAMDA = DK/DM2
      XX = XSTN - RHO
      Q2 = Q1CF(2)
      Q1 = Q1CF(1)
      PR1 = XX+1.0D0-DR
      PR2 = (DR-1.0D0 - XX/2.0D0)*XX - (DR-1.0D0)*(DR-1.0D0)/2.0D0
```

```
      PIMAG1 = (XX*XX+(2.0D0-DR)*DR-1.0D0)*DLAMDA/2.0D0
      PIMAG2 =   XX*XX*((DR-1.0D0)/2.0D0-XX)-((DR-1.0D0)**3)/6.0D0
      PIMAG1=PIMAG1*DLAMDA
      PIMAG2 = PIMAG2 * DLAMDA
      QP1 = DCMPLX(PR1,PIMAG1-DLAMDA*PR1*XSTN)
      QP2 = DCMPLX(PR2,PIMAG2-DLAMDA*PR2*XSTN)
      Q1PRIP =-(Q1*QP1+Q2*QP2)/DSQRT(DM2)
      IF(IPT.LE.0) RETURN
      GOTO 30
   20 Q1PRIP = DCMPLX(0.0D0,0.0D0)
      IF (IPT.LE.0) RETURN
   30 WRITE (6,995) Q1PRIP
  995 FORMAT('0',10X,'Q1PRIP = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1DAIP*16(DK,DR,DW,RHO,OFST,SGMA,XSTN,Q1CF,N,IPT)
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
      DIMENSION Q1CF(13)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,OFST,SGMA,XSTN,IPT
  990 FORMAT('0',10X,'Q1DAIP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,'OFST',9X,'SGMA',9X
     2  ,'XSTN',9X,'IPT'/,' ',10X,7(E12.5,','),I3)
      XASTN = XSTN - OFST
      IF(XASTN.LE.DR+RHO-1.0D0) GO TO 20
      QCNST = CDEXP( DCMPLX(0.0D0,SGMA))
      GAMMA = 1.4D0
      DM2 = (GAMMA+1.0D0)*DW
      DLAMDA = DK/DM2
      XX = XASTN - RHO
      Q2 = Q1CF(2)
      Q1 = Q1CF(1)
      PR1 = 1.0D0
      PR2 = DR-1.0D0-XX
      PIMAG1 = DLAMDA*XX
      PIMAG2 =DLAMDA*(DR-1.0D0-XX)*XX*XX
      QP1 = DCMPLX(PR1,PIMAG1-DLAMDA*PR1*XASTN)
      QP2 = DCMPLX(PR2,PIMAG2-DLAMDA*PR2*XASTN)
      Q1DAIP = (Q1*QP1 + Q2*QP2)/DSQRT(DM2)
      IF(IPT.LE.0) RETURN
      GOTO 30
   20 Q1DAIP = DCMPLX(0.0D0,0.0D0)
      IF(IPT.LE.0)RETURN
   30 WRITE (6,995) Q1DAIP
  995 FORMAT('0',10X,'Q1DAIP = ',E14.7,',',E14.7)
      RETURN
      END
      COMPLEX FUNCTION Q1DRIP*16(DK,DR,DW,RHO,OFFSET,XSTN,Q1CF,N,IPT)
      IMPLICIT REAL*8 (A-H,O,P,R-Y), COMPLEX * 16 (Q,Z)
```

```
      DIMENSION Q1CF(13)
      IF (IPT.GT.0) WRITE(6,990) DK,DR,DW,RHO,XSTN,IPT
  990 FORMAT('0',10X,'Q1DRIP ENTERED WITH:',/,
     1  ' ',10X,'DK',11X,'DR',11X,'DW',11X,'RHO',10X,
     2  'XSTN',9X,'IPT'/,' ',10X,5(E12.5,','),I3)
      IF(XSTN.LE.DR+RHO+OFFSET-1.0D0) GOTO 20
      GAMMA = 1.4D0
      DM2 = (GAMMA+1.0D0)*DW
      DLAMDA = DK/DM2
      Q1 = Q1CF(1)
      Q2 = Q1CF(2)
      XX = XSTN - RHO
      PR1 = 1.0D0
      PR2 = DR-1.0D0-XX
      PIMAG2 =DLAMDA*(DR-1.0D0-XX)*XX*XX
      PIMAG1 = DLAMDA*XX
      QP1=DCMPLX(PR1,PIMAG1-DLAMDA*PR1*XSTN)
      QP2 = DCMPLX(PR2,PIMAG2-DLAMDA*PR2*XSTN)
      Q1DRIP =-(Q1*QP1+Q2*QP2)/CSQRT(DM2)
      IF(IPT.LE.0) RETURN
      GOTO 30
   20 Q1DRIP = DCMPLX(0.0D0,0.0D0)
      IF (IPT.LE.0) RETURN
   30 WRITE (6,995) Q1DRIP
  995 FORMAT('0',10X,'Q1DRIP = ',E14.7,',',E14.7)
      RETURN
      END
```

LIST OF REFERENCES

1. Elder, P. R., A Theoretical Analysis of Unsteady Transonic Cascade Flow, Thesis, Naval Postgraduate School, Monterey, California, 1972.

2. Schlein, P. B., A Study of Unsteady Transonic Interference Effects, Thesis, Naval Postgraduate School, Monterey, California, 1975.

3. Landahl, M., Unsteady Transonic Flow, Pergamon Press, 1961.

4. Gorelov, D. N., "Oscillations of a Plate Cascade in a Transonic Gas Flow," Mekhanika Zhidkosti i Gaza, v. 1, no. 1, pp. 69-74, 1966.

5. Garrick, I. E. and Rubinow, S. J., Flutter and Oscillating Air-Force Calculations for an Airfoil in Two-Dimensional Supersonic Flow, NACA Report 846, 1946.

6. Stevens, W. P.G. J. Meyers, and L. L. Constantine, "Structured Design," IBM Systems Journal, No. 2, pp. 115-139, 1974.

7. Shampine, L. F., and R. C. Allen, Numerical Computing: An Introduction, W. B. Saunders Company, 1973.

8. Ashley, H. and Landahl, M., Aerodynamics of Wings and Bodies, Addison-Wesley Publishing Company, Inc., 1965.

9. Miles, J. W., "The Compressible Flow Past an Oscillating Airfoil in a Wind Tunnel," Journal of Aeronautical Sciences, v. 23, no. 7, pp. 671-678, 1956.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Professor M.F. Platzer, Code 57Pl Department of Aeronautics Naval Postgraduate School Monterey, California 93940	10
4.	Lt. Peter C. Olsen USCG 16207 Pointer Ridge Drive Bowie, Md. 20716	4
5.	Assoc. Professor F.R. Richards, Code 55rh Department of Operations Research Naval Postgraduate School Monterey, Ca. 93940	1
6.	Mr. Robert C. Olsen Quality Engineering, WSOB Pratt and Whitney Aircraft East Hartford, Conn. 08106	1
7.	Professor D.J. Collins, Code 67Co Department of Aeronautics Naval Postgraduate School Monterey, Ca. 93940	1