Fall 12-20-2013

A Wedge Impact Theory Used to Predict Bow Slamming Forces

Ashok Benjamin Basil Attumaly
aattumal@uno.edu

A Wedge Impact Theory Used to Predict Bow Slamming Forces

A Thesis

Submitted to the graduate faculty of the
University of New Orleans
In partial fulfillment of the
requirements of the degree

Master of Science
in
Engineering
Naval Architecture and Marine Engineering

by

Ashok Benjamin Basil Attumaly

B.Tech, Indian Institute of Technology Delhi, 2010

December, 2013

Acknowledgements

Seafaring vessels and the perils they face at sea have fired my imagination since I was child in my hometown of Kochi, India. In the process of pursuing my masters, during my first internship at Elliott Bay Design Group, I was intrigued by the forces of slamming that act on vessels and the theoretical modeling of the phenomenon of bow wave slamming. Fortunately, the department of Naval Architecture and Marine Engineering at UNO have a long tradition of research in that field.

I would like to thank Dr Brandon Taravella for the guidance he has given me over the course of my project. His knowledge and youthful energy provided me with the backing I need for an endeavor like writing a thesis. I would like to thank Dr Lothar Birk and Dr Chris McKesson for their valuable inputs as part of my evaluation committee. The support extended by my team at North American Shipbuilding has also been incredible.

I'd like to thank my parents and my brother for their continued support over the years. Finally, I'd like to thank my friends in New Orleans who have made graduate school a memorable experience and work, a pleasure.

Table of Contents

List of Figures iii

List of Tables iv

Abstract v

1. Introduction 1

2. Literature Survey 4

3. Impact Investigation Theory 8

3.1. Vorus' Impact Theory 8

3.2. Flow Physics 8

3.3. Velocity Definitions and Orders of Magnitude 10

3.4. Theoretical formulation 11

3.4.1. Pressure continuity 13

Free contour dynamic boundary condition 13

3.4.2. Velocity continuity 14

3.4.3. Displacement continuity 16

3.5. Discretizaton of boundary conditions 17

3.5.1. Pressure continuity 17

3.5.2. Velocity continuity 19

3.5.3. Displacement continuity 20

3.6. Cylinder Pressure and force distribution 22

3.7. Initial Condition 23

3.7.1. Pressure continuity 23

3.7.2. Velocity continuity 23

3.7.3. Displacement continuity 24

3.8. Computation of pressure coefficients over a 3D contour - Linear Strip Theory 25

3.8.1. Time steps and pressure distributions 25

4. Solution methodology and algorithm 27

4.1. Algorithm 29

5. Verification of accuracy of code 31

5.1. Velocity comparison 31

5.2. Keel pressure comparison 34
6. Validation of predicted results against experimental results 39
6.1. Discussion of the comparison 43
7. Time varying impact force variation in a hull form 44
7.1. Characteristics of the hull form 44
7.2. Theoretical background 45
7.3. Problem set-up 45
7.4. Comparison and discussion of results 48
8. Conclusions 53
8.1. Further research suggested 53
9. Works Cited 55
Appendix A: Fortran code for processing of input data 57
Appendix B: Post processing code for data output by code vsheet228.for 86
Appendix C1: Input format (inpf.txt) for offset based section definition 90
Appendix C2: Input format (inpf.txt) for angle based section definition 92
VITA 93

List of Figures

Figure 1(a): Cylinder impact (cuw) (Vorus, 1996).. 9
Figure 1(b) Cylinder impact (cw).. 10
Figure 2: Vortex sheet distribution and velocity components along the wetted portion (Vorus, 1996).. 12
Figure 3 Definition of variables (Vorus, 1996).. 12
Figure 4: Vsii from ζ distribution (Vorus, 1996)... 18
Figure 5: Discretization of 3D hull form into 2D sections (Bajic et al , 2010) 25
Figure 6: Sample pressure contour output after post processing in Excel.................................... 26
Figure 7: Hard chine contours (Vorus, 1996)... 31
Figure 8: Particle velocity, zc, and zb (Vorus, 1996).. 32
Figure 9(a): 20-10 Vs/10 distribution.. 33
Figure 9(b): 20-20 Vs/10 distribution.. 33
Figure 9(c): 20-30 Vs/10 distribution.. 34
Figure 10: Keel pressure coefficient Cp (0,τ) (Vorus, 1996).. 35
Figure 11(a): Keel pressure coefficient on 20-10 contour.. 36
Figure 11(b): Keel pressure coefficient on 20-20 contour.. 36
Figure 11(c): Keel pressure coefficient on 20-30 contour.. 37
Figure 12: Bow section (left) and experimental setup (right) considered by Aarsnes(1996)......... 39
Figure 13: Figure 13: Pressure distribution on the bow section, experimental vs. VOF method (Ghadimi et al, 2013)... 40
Figure 14: Discretized contour of the section used by Aarsnes (1996).. 41
Figure 15: Comparison of code prediction versus Ghadimi et al (2013) & Aarsnes (1996) at τ=0.5351.. 42
Figure 16: Comparison of fluid particle flow in Ghadimi et al (2013) and particle velocity........ 43
Figure 17: Bajic et al (2010) analysis hullform... 44
Figure 18: Discretized hull form contours.. 46
Figure 19: Waterline (YWL) at various time steps... 47
Figure 20(a): Impact force distribution at τ=0.07... 48
Figure 20(b): Impact force distribution at τ=0.10... 48
Figure 20(c): Impact force distribution at τ=0.20... 48
Figure 20(d): Impact force distribution at τ=0.30... 48
Figure 21: Pressure prediction using various class societies' rules (Bajic et al, 2010)................. 49
Figure 22: Comparison of impact force vs. Pressure on Level 1.. 49
Figure 23(a): Cp distribution on frames at τ=0.07.. 50
Figure 23(b): Cp distribution on frames at τ=0.10.. 50
Figure 23(c): Cp distribution on frames at τ=0.20.. 50
Figure 23(d): Cp distribution on frames at τ=0.30.. 50
Figure 24(a): Cp contour distribution on wetted portion of hull at τ=0.07................................... 51
Figure 24(b): Cp contour distribution on wetted portion of hull at τ=0.10................................... 51
Figure 24(c): Cp contour distribution on wetted portion of hull at τ=0.20................................... 52
Figure 24(d): Cp contour distribution on wetted portion of hull at τ=0.30................................... 52

List of Tables

Table 1: Orders of magnitude of cylinder impact parameters (Vorus, 1996)........... 11

Abstract

The pressures and impact forces acting on a hull while experiencing bow wave slamming is analyzed using Vorus' Impact Theory. The theory extends the hydrodynamic analysis of planing hulls from simple wedges to irregular shapes using a Boundary Element Method. A Fortran-based code developed by the Author is used to analyze hullforms. Linear strip theory is used to extend the analysis over a three dimensional hull. Post-processing of output data gives hull pressure distributions at different time steps and is visually presentable.

Impact pressure, Impact force, Planing, Wave slamming, Bow impact, Vorus' theory, Boundary Element Method, Linear strip theory

1. Introduction

The impact problem is one of the present day problems being keenly investigated, especially in the realm of high speed hydrodynamics. There has been a fundamental shift in naval ship design requirements which has seen a greater emphasis on speed and agility of vessels. In the present design environment, vessel designers are striving to design vessels that can achieve greater speed and maneuverability with the available power. This provides the impetus for research in high speed vessels. A better understanding of the hydrodynamics of lift, drag and impact forces on these vessels could provide end users with enhanced operational ability.

The impact problem is also applicable to the realm of normal displacement hulls for estimation of impact forces during wave slamming and similar non-linear phenomena which cannot be easily accounted for using linear hydrodynamics. Recent years have seen a spate of regulations intended at increasing the operational safety of commercial vessels.

A better understanding of impact forces and pressure distribution will allow designers to strengthen hulls sufficiently to withstand these forces and design hullforms which could prevent dangerous excessive pounding of the hull in rough conditions. This would also help class societies develop new regulations to ensure adequate safety of vessels in seas, taking into account phenomena like bow impact, wave slamming, etc. Regulations regarding impact forces can be given more theoretical basis compared to the largely empirical nature of regulations for design readiness of vessels in wave impact situations.

The impact problem has been investigated since the first half of the 20th century. The first investigations into the wave impact problem were by Von Karmen (1929) and Wagner (1932) independently in USA and Germany respectively. Their work has laid the groundwork for much of the theoretical development over the many decades that followed, till present date. Subsequent research works on the subject have devised models to predict the lift and drag of a planing body executing planing motion. Maruo (1967) assumed the vessel's planing surface as a distribution of vortices and tried to solve the problem of pressure distribution using potential theory. Maruo subjected the problem to the linearized free surface boundary condition, including gravity effects. Shen & Ogilvie (1972) approached the problem applying conformal mapping of contours to regular shapes like a line or semi circle and solving the potential of the flow. Taravella & Vorus (2010) have applied the theory of Maruo to a series distribution of offsets and successfully predicted the lift coefficients, coupling the effects of upstream stations on downstream stations on low-aspect ratio hullforms. These theories have, over the decades, come closer and closer to realistic predictions of lift and drag for increasingly complex shapes.

The theories discussed above were initially developed for sea planes and planing hulls. Their applicability to semi-planing, semi-displacement and even displacement hulls, have in recent years been extended. Taravella (2009) developed a hybrid method for predicting lift / drag on semi-planing / semi-displacement hulls.

Many of the theories mentioned above have focused on lift / drag prediction on hull forms. Only a few have focused on the phenomena of impact pressures, Vorus (1996), being one of them. Impact, being a non-linear phenomenon has complexities of flow that are different from merely a lift force. The theory approximates the geometry using linear approximations, and performs the hydrodynamic analysis as a non-linear problem.

More recently Ghadimi et al (2011) have investigated the entry of a wedge onto a horizontal water surface using Schwartz Christoffel mapping. A subsequent work has been published by Ghadimi et al (2013) which computes the pressure distributions and separations across the hull cross section using a VOF (Volume of Fluid) scheme in conjunction with FVM (Finite Volume Method).

The present work develops on the work of Vorus (1996). The work of Vorus has been used to formulate a code in Fortran which can successfully output the pressure distribution on a hullform undergoing impact motion. The theory has been extended to a hullform from a single station using Linear Strip theory. The work attains significance in the context of the fact that the theory developed by Vorus (1996) was a Boundary Element Method (BEM). BEM methods are significantly faster than volume element methods. The present work can help identify regions of significant pressures on hullforms due to wave impact and account for structural strengthening required at these regions or modify hullform designs to reduce these pressures at the conceptual stage of design.

2. Literature Survey

As all studies of theories on impact done before, the literature study for the present theory began with a study of the works of Von Karmen (1929) and Wagner (1932). Von Karmen's work analyzes the impact pressure experienced by a prismatic wedge hull dropped vertically, striking a horizontal water surface. Von Karmen's model approximated this highly non-linear phenomenon into a linear formula by applying the requisite simplifications. There is mass added by virtue of the hydrodynamic effects. The formulation developed for pressure is based on the conservation of momentum. The maximum pressure, located at the middle of the float, is found to be inversely proportional to the angle of deadrise, approaching infinity at zero deadrise according to the formulation. Von Karmen also proposes a limiting value for force at zero deadrise, i.e., for a flat plate. The pressure decreases moving outward along the length of the span of the wetted region of the wedge. Von Karmen suggests that the limiting value suggested by him is an over-approximation as the wedge is not a completely rigid body and there would be deformation in the body by virtue of the applied pressure.

Wagner's theory (1932) was more detailed in its analysis of the horizontal water surface and the effect of the impacting body on the water surface. Wagner's theory introduced the concept of "spray root" and the "wetting factor". Wagner also noted the high pressure gradients near the spray root. Wagner's solution was divided into different zones, the outer domain - the principal region, where the water surface interacts with the surface of the wedge, the splash root, the region between the surface and the spray root, and the splash, the jet region of the flow. Wagner's theory was the first to apply Schwartz Christoffel mapping to compute the pressure distribution on the contour surface.

Most theories proposed till 1967 had confined the analysis to a 2D plane, in a station-wise fashion. Maruo (1967) laid the theoretical foundation for solving the lift and drag force on a three dimensional body by virtue of the incident (forward) velocity. The velocity potential of the flow was solved from the Laplace equation. The zero flux of flow across the hull, and the radiation condition were used by Maruo as boundary conditions for the problem. The analysis was conducted for two limiting cases - the high aspect ratio body, where the behavior is similar to that of an aerofoil, using Bassell functions, and the low aspect ratio, which is a good approximation for elongated bodies (width << length). The mathematical complexity of the equation's formulation led Maruo to assume a high Froude number for the body to simplify the solution. Perhaps the most important contribution of Maruo to the field of impact theory was introducing the effect of gravity to the lift force formulation acting on a body.

Aarsnes (1996) had conducted drop tests of ship sections. The work detailed pressure variation along the span of the section, and also the observed water surface profile. Subsequent researchers have used results published by Aarsnes as experimental data for testing numerical analysis of impact prediction codes.

Vorus (1996) had proposed a boundary element method which takes a unified approach to the flow. Unlike previous theories, the computation for the far and near regions of the flow followed the same formulation. Vorus' theory is time dependent and hence could handle shapes / contours which are dissimilar in time. The theory is geometrically linear in the way it deals with the flat cylinder boundary conditions while simultaneously being hydrodynamically nonlinear by fully retaining the large flow perturbation produced by the impacting flat cylinder in the axis boundary conditions. The present work has been developed based on this theory.

Royce (2001) had extended Vorus' theory to a 2D planing crafts using a hydro-elastic model of impact, comparing pressure distributions for impact of 2D surfaces with experimental observations. The work also introduced the concept of temporal variations of impacting surface during the impact process, referred to as Localflex.

Maruo's work was further expanded by Taravella (2009) and Taravella & Vorus (2010). Their works extended the solution of the pressure distributions for Froude numbers that are not high enough to apply the high Froude number assumption for the low aspect ratio case, as is the case in semi-planing and semi-displacement hulls. This was achieved using Fresnel integrals to solve the flow potential equations. Taravella, in addition to this, applied Michell's (1897) thin ship theory to predict the drag on vessels of moderate Froude numbers. The proposed solutions couple the effects of incident velocities on upstream stations onto the solutions for downstream stations. These works have expanded the applicability of planing hydrodynamics theories from planing hulls to semi-planing and semi-displacement hulls. These works however are not applicable to impact problems where the flow is incident from beneath the hull rather than along the hull.

Ghadimi et al (2011) have taken the approach of using conformal mapping to solve the impact problem. The formulation takes vertical velocity as the input velocity. The authors have used the image method where a Galilean transformation is applied to transform a hull contour to a closed shape. The flow potential is solved for this body which is symmetric about y-axis after the Galilean transformation. For the purpose of solving the potential of the flow, a Schwartz Christoffel transformation is applied to the transformed body. The transformation breaks down the problem of the flow through a rhombus in the physical plane (z-plane) to a uniform vertical flow problem in the mapped plane (w=p+iq). The two free surface boundary conditions that bound the problem are: (1) Kinematic boundary condition on the free surface, and (2)The Dynamic boundary condition on the free surface. The pressure distribution is solved as a function of velocity distribution along a line. The potential of the flow is obtained from the transformed velocities, and application of the dynamic boundary condition to this gives the profile of the free surface of water.

Ghadimi et al (2013) have also done a recent study for comparison of results of Aarsnes work against pressures predicted by a VOF (Volume of Fluid) scheme in conjunction with FVM (Finite Volume Method). The code successfully captures the reattachment of flow after the primary flow separation. The capturing of this detail predicts peak pressures close to the flow separation point. This observation was reported by Aarsnes (1996) in the ship section drop test results.

In the context of classification society rules, Bajic et al (2010) presented a study of major class societies' rules on design impact pressures on a containership. The present regulations in relation to slamming pressures are empirical in nature. The study shows the variation of design pressures along the hull of containership undergoing slamming wave action on the bow section as comparison amongst different class societies. The study also reports the variations in pressure with variation in hullform coefficients, ship speed, bow flare and ship draught. Their study reports a high sensitivity of slamming pressures to the bow flare angle.

3. Impact Investigation Theory

3.1. Vorus' Impact Theory

The impact analysis is performed according to the theory detailed by Vorus (1996). At certain occasions, certain deviations have been used in the code. For example, using the number of segments ni as equal to i, instead of computing this. A correction has been applied to equation (49) in Vorus (1996) (Equation (23) in the present work), using $\left(\frac{\bar{\zeta}_{ij}^2-1}{\bar{\zeta}_{ij}^2}\right)$instead of $\left(\frac{1-\bar{\zeta}_{ij}^2}{\bar{\zeta}_{ij}^2}\right)$ as the first coefficient of multiplication. However, the results have been verified against available data to ascertain the correctness of the code developed for analysis.

Vorus' theory offers a single solution field for the hydrodynamic analysis. The principle complexity of the flat cylinder theory is the increasing transverse flow perturbation and the non-linearity associated with increasing flatness. Vorus' theory has been extended to general contours, with restrictions, from flat cylinders. An advantage of the Vorus theory is that it possible to solve the problem for non-similar, time-dependent flows. The method is a mixed theory - geometrically linear, i.e., the flat cylinder boundary conditions are satisfied on the horizontal axis, and hydrodynamically nonlinear, as in fully retaining the large flow perturbations produced by the impacting flat cylinder in the axis boundary conditions.

3.2. Flow Physics

The hydrodynamic model considered is ideal and incompressible. Gravity is not considered in the problem.

The solution is developed on an impacting flat cylinder model (Figure 1(a) and (b)), Figure 1(b) portrays complete penetration of the cylinder (chine wetted flow) into the water surface. The point where the continuous hull contour terminates, referred to as the chine, is taken as the point where the flow separates provided premature separation doesn't happen. In case of premature separation, there would be no further advance of the point at which the flow contour has zero pressure. The theory is built here on the assumption of symmetry about the y-axis, the vertical plane of the cylinder.

On impact, the free surface is turned back under the contour forming an initially attached jet, as shown in Figure 1(a). The "spray root" advances rapidly along the contour, followed closely by point C. The contour pressure is zero at C and beyond. Point C moves outward till it reaches the chine. Beyond this point, B continues further outward, though C remains fixed on the chine. Point C is the point where the flow detaches itself from the contour. On the upper branch of fluid flow, demarcated by B, the stream velocity is higher than the impact velocity, and on the lower branch, the stream velocity is lower than the impact velocity. Increasing flatness accentuates the difference in velocities between the upper and lower branches. For analysis, the cylinder is collapsed onto the z-axis. An important character of the flow is the drop in tangential velocity in the region $z_c \leq z \leq z_b$ by an order of magnitude on the flow becoming a chine wetted flow.

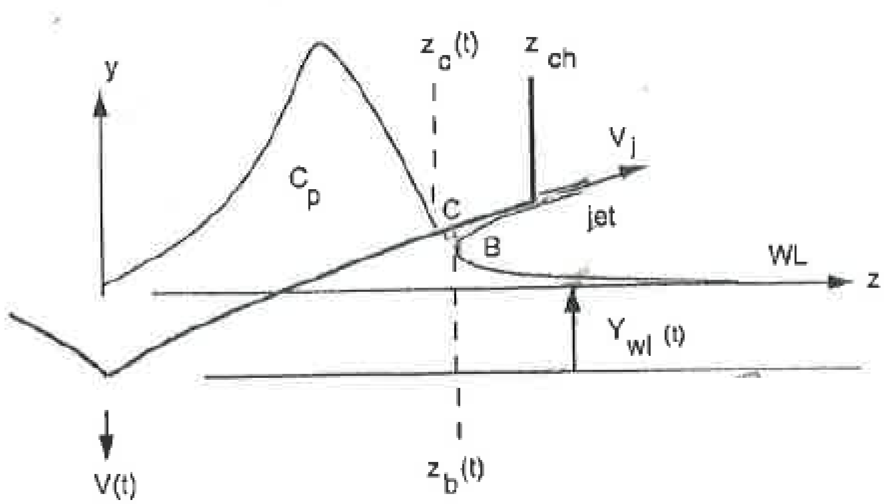


Figure 1(a): Cylinder impact (cuw) (Vorus, 1996)

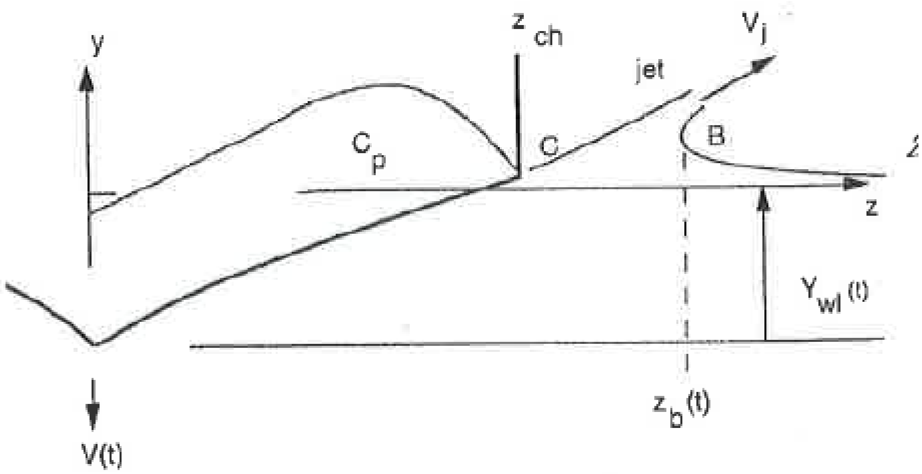


Figure 1(b) Cylinder impact (cw) (Vorus, 1996)

3.3. Velocity Definitions and Orders of Magnitude

Understanding the various velocities - contour velocities and perturbations, is key to understanding the nature of the theory. The velocities on the flat cylinder is split into V_s and V_n, tangential and normal to the surface respectively. The perturbations, v and w, are defined with respect to the original axes of the flat cylinder. For the purpose of simplicity, the flat cylinder is considered symmetric about the y-axis.

The values of contour velocities are as described at different stages of flow:

$V_n = 0,\ z_c \leq z \leq z_b$

$V_n = V,\ z_c \leq z,\ z_c = Z_{ch}$

$V_n = V,\ V_s = 0,\ z_b \leq z,\ \ z_c \leq Z_{ch}$

As described in the previous section, the velocity undergoes significant changes in magnitude depending on the behavior of the flow (CUW / CW). The orders of magnitude of the perturbations and contour velocities are described in Table 1.

Table 1: Orders of magnitude of cylinder impact parameters (Vorus, 1996)

	$0 \leq z \leq z_c$		$z_c \leq z \leq z_b$		$z > z_b$
	(cuw)	(cw)	(cuw)	(cw)	(cuw & cw)
$Z_c(t)/Z_{ch}$	<1	1	<1	1	≤1
v(z,t)	O(1)	O(1)	O(1)	O(β)	O(β)
w(z,t)	O(1/β)	O(1/β)	O(1/β)	O(1)	O(β)
$V_n(z,t)$	0	0	O(β)	V+ O(β)	V+ O(β)
$V_s(z,t)$	O(1/β)	O(1/β)	O(1/β)	O(1)	O(β)
$\frac{\partial}{\partial z}$	O(1)	O(1)	O(1)	O(1)	O(1)
$\frac{\partial}{\partial t}$	O(1/β)	O(1/β)	O(1/β)	O(1)	O(1)

3.4. Theoretical formulation

The impact problem is non-dimensionalized. All impact velocities are non-dimensionalized on a reference velocity V_0, which could be the velocity upon impact at time 0. The offsets of the contour surface are non-dimensionalized on Z_{ch}, the offset of the chine. The time is thus non-dimensionalized with the help of the above defined quantities.

$$\tau = \frac{(V_o\, t)}{Z_{ch}}$$

Zero gravity is assumed for the problem. For the contour outside the zero pressure point, the tangential velocity is assumed to be zero, i.e., $V_s = 0$ for $z \geq z_b$. The remaining boundary conditions are satisfied with a vortex distribution between the axis and the spray root.

The solution is scaled by the zero pressure distribution point offset $z_c(t)$, i.e.,

$\zeta = z/z_c(t)$

The spray-root offset in the ζ-space is then:

$b(\tau) = z_b(\tau)/z_c(\tau)$

The strength of the line vortex distribution in Figure 2 is given by $\gamma(\zeta,\tau) = -2V_s(\zeta,\tau)$

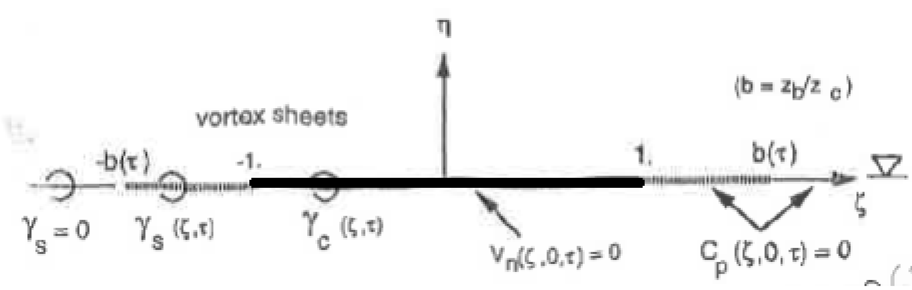


Figure 2: Vortex sheet distribution and velocity components along the wetted portion (Vorus, 1996)

3.4.1. Pressure continuity

Free contour dynamic boundary condition

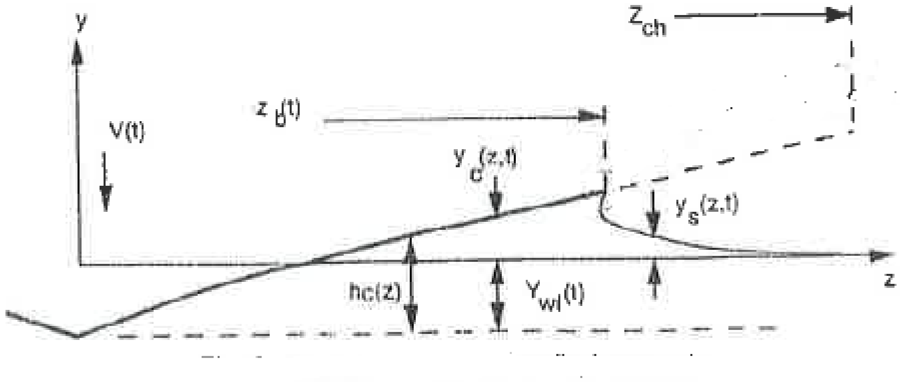


Figure 3: Definition of variables (Vorus, 1996)

Referring to Figure 2, zero pressure is required on the free contour beyond ζ=1:

$Cp(\zeta,\tau) = 0$ on $\zeta \geq 1$

The definition for Cp at $0 \leq \zeta \leq b$, as derived from Bernoulli equation's unsteady form:

$$Cp(\zeta,\tau) = V^2(\tau) - V_n^2(\tau) + 2\, z_{c\tau}\left[\int_{\zeta 0=\zeta}^{b(\tau)} V_s(\zeta,\tau)d\zeta_0 + \zeta V_s(\zeta,\tau)\right] + 2z_c\left[\int_{\zeta 0=\zeta}^{b(\tau)} V_{s\tau}(\zeta,\tau)d\zeta_0 + b_\tau V_s(b,\tau)\right] \qquad 0 \leq \zeta \leq b(\tau) \qquad (1)$$

In (1) $z_c(\tau)$ is the non-dimensional zero pressure point offset, and the subscript τ denotes ∂/∂τ.

Also note that C_p=0 for $1 \leq \zeta \leq b$. This is satisfied when $V_n = 0$ for chine unwetted flow ($z_c < 1$) and $V_n = V(\tau)$ for chine-wetted flow ($z_c = 1$)

$$(V_s - z_{c\tau}\zeta)\frac{\partial V_s}{\partial \zeta} + z_c\frac{\partial V_s}{\partial \zeta} = 0 \qquad 1 \leq \zeta \leq b(\tau) \qquad (2)$$

This is the nonlinear form of Euler's equation, the one-dimensional inviscid Burger's equation on a time and spatially variable stream. Manipulating this equation and applying the condition that Cp = 0 in $1 \le \zeta \le b$, we get

$$z_{b\tau} = \frac{V_s^2(\zeta,\tau)+V_n^2(\tau)-V^2(\tau)}{2V_s(b,\tau)} \qquad (3)$$

Thus a definition of the spray root velocity, $z_{b\tau}$, is obtained from the pressure formulation.

Free vortex sheet distribution. Euler's equation requires that velocity of the particles flowing out from the contour onto the free vortex and out into the jet has a constant velocity at its separation (at $z_c(\tau')$) and for all time $\tau > \tau'$ thereafter. Applying galean transformation to (2) gives the following relations for the position of the particle with velocity $Vs(\zeta,\tau)$:

$$\hat{\zeta}(\zeta',\tau) = \frac{V_s(\zeta',\tau_0)(\tau-\tau_0)+z_c(\tau_0)\zeta'}{z_c(\tau)} \qquad 1 \le \zeta' \le b(\tau_0), \tau \ge \tau_0 \qquad (4)$$

$$\hat{\zeta}(\tau,\tau') = \frac{V_s(1,\tau\prime)(\tau-\tau\prime)+z_c(\tau\prime)}{z_c(\tau)} \qquad \tau \ge \tau \ge \tau_0 \qquad (5)$$

τ_0 is the starting time where $V_s(\zeta,\tau)$ in $1 \le \zeta \le b$ must be known. The uniform $V_s(\tau_0)$ is computed from the wedge similarity solutions. The spray root velocity is always less than the jet root velocity. This implies that the term $\zeta\,\hat{}[b(\tau_0),\tau]$ defined in (4) is always greater than the value of $b(\tau)$. Thus, except at $\zeta = 1$, the free vortex sheet strength is completely defined from equations (4) and (5), given $b(\tau)$.

3.4.2. Velocity continuity

Contour kinematic boundary condition

The kinematic boundary condition is satisfied on the contour segment of the z-axis. In the downward moving coordinate system:

$V_n(\zeta,\tau) = 0$ on $0 \le \zeta \le 1$ (6)

Substituting the definition of V_n in terms of vertical perturbation velocity, vorticity and impact velocity, the definition becomes

$v(\zeta,\tau) + 1/2\, \gamma(\zeta,\tau) \sin \beta(\zeta,\tau) = -V(\tau) \quad 0 \le \zeta \le 1$ (7)

Here β is a function of time as well as the position on the wetted surface. The perturbation velocity, v is eliminated using the Biot Savart law:

$1/2\, \gamma(\zeta,\tau) \sin \beta(\zeta,\tau) + 1/2\pi \int_{-b}^{b} \frac{\gamma(\zeta,\tau)}{(\zeta-\zeta 0))} = -V(\tau)$ (8)

The γ function is split as γ_c and γ_s for the wetted region of the contour and the free sheet region respectively. This conversion, after being solved using the solution for integral equation of the Carleman type and interchange of order and other manipulation processes gives the result:

$$\gamma_c(\zeta,\tau) = -\frac{2\cos\tilde{\beta}\zeta\; \kappa(\zeta,\tau)}{\sqrt{1-\zeta^2}}\left[V(\tau) + \frac{1}{\pi}\int_{s=1}^{b(\tau)} \frac{\gamma_s(s,\tau)ds}{\kappa(s,\tau)\sqrt{s^2-1}} + \frac{\zeta^2-1}{\pi}\int_{s=1}^{b(\tau)} \frac{\gamma_s(s,\tau)ds}{\kappa(s,\tau)\sqrt{s^2-1}\,(s^2-\zeta^2)}\right] \quad (9)$$

s is a dummy variable of ζ-integration. The function κ(ζ,τ) is defined as

$$\kappa(\zeta,\tau) = \prod_{k=1}^{K}\left|\frac{\zeta^2-\zeta_{k+1}^{*2}}{\zeta^2-\zeta_k^{*2}}\right|^{\frac{\tilde{\beta}_k^*(\tau)}{\pi}} \quad (10)$$

The flow should be continuous from ζ = 0 to ζ = 1 and beyond. The laws of physics require the velocity to maintain continuity at all regions of the flow. This requirement serves as one of the key boundary conditions that enable the setting up and solving of the equations. In view of the non-singular character of κ(ζ,τ) at ζ=1, in order that the value of γc remains finite, it is required that:

$$V(\tau) + \frac{1}{\pi}\int_{s=1}^{b} \frac{\gamma_s(s,\tau)ds}{\kappa(s,\tau)\sqrt{s^2-1}} = 0 \qquad (11)$$

This is known as the "Kutta condition". This gives a relationship between the values of γ_s, $z_{b\tau}$ and b.

3.4.3. Displacement continuity

As discussed for velocity continuity, the flow has to maintain continuity at all points of the flow. This serves as another key boundary condition that allows us to set up and solve equations to compute the flow parameters.

The requirement that has to be met is $y_c(z_b,t) = y_s(z_b,t)$. On non-dimensionalizing this problem, we get the equation:

$$v(z,\tau) + 1/2\ \gamma_c(z,\tau)\ \sin\beta(z) = -V(t) \qquad 0 \le z \le zb(t) \qquad (12)$$

The impact velocity is a function of y_c. By rearrangement, application of Biot-Savart law and manipulation of the equation, a relation between y_s and y_c* is obtained.

$$y_s(\xi,\tau) = \frac{1}{2\pi}\int_{\xi_0=-1}^{1} \frac{y_c^*(\xi_0,\tau)}{\xi_0-\xi} d\xi_0 \qquad (13)$$

To maintain continuity of displacement at point C, the requirement $y_s(1,\tau) = -Y_{wl}(\tau) + h_c(1,\tau)$ has to be met.

This is accomplished if:

$$Y_{wl}(\tau) = \frac{2}{\pi}\int_{s=0}^{1} \frac{\cos\tilde{\beta}(s,\tau) + h_c(s,\tau)}{\kappa(s,\tau)\sqrt{1-s^2}} ds \qquad (14)$$

3.5. Discretizaton of boundary conditions

3.5.1. Pressure continuity

The theorem can be applied for computational purposes on contours only after discretization of the equations. The pressure continuity condition presents a jet-head free vortex sheet overlaid on the particle velocity distribution:

$$\hat{\zeta}_{i+1} = \frac{V_{s0}(\tau_i-\tau_0)+z_{c0}b_0}{z_{ci}} \ , \ \hat{\zeta}_i = \frac{V_{s0}(\tau_i-\tau_0)+z_{c0}}{z_{ci}}$$

$$\hat{\zeta}_j = \frac{V_{s\,i-j}(\tau_i-\tau_{i-j})+z_{c\,i-j}}{z_{ci}} \ , \qquad j = 1, i-1 \qquad (15)$$

$$\hat{\zeta}_0 = 1$$

The velocity at the indicated particle positions, can also be transposed in time by the relation:

$$V_s(\hat{\zeta}_{j,}\tau_i) = V_s(1,\tau_{i-j}) = V_{sij} \qquad (16)$$

The strength of the vortex sheet at each segment is given by $\gamma_{sij} \equiv -2\ V_s(\bar{\zeta}_{ij},\tau)$, the length of each segment being Δb_{ij}(j =1 to n_i), evaluated at the ζ_{ij} and averaged at the midpoint to get $\bar{\zeta}_{ij}$. The distribution extends from ζ_{i0} =0 to $\zeta_{ini} = b_i$. A new segment is added at each step, however, provision is made in theory for cases where $n_i < i$ (when the deceleration value is sufficiently high to reverse the advance of z_c).

$$\zeta_{ij} = 1 + \sum_{k=1}^{j} \delta b_{ik} \equiv \zeta_{i\ j-1} + \delta b_{ij} \qquad j = 1, \dots\ , n_i \qquad (17)$$

with

$$\delta b_{ij} = \frac{\delta z_{bn_i-j+1}}{z_{ci}} \ , \quad \delta z_{bi} = (z_{bi\tau} - z_{ci\tau})\Delta\tau \qquad (18)$$

zbiτ used here is the discretized form. Its definition is given by:

$$z_{bi\tau} = \frac{V_{si}^2+V_{ni}^2-V_i^2}{2\,V_{si}(b_i)} \tag{19}$$

The value of $V_{si}(b_i)$ is computed from the velocity distribution shown in Figure 4. The value of velocity is interpolated from the curve as shown by the red line in Figure 4.

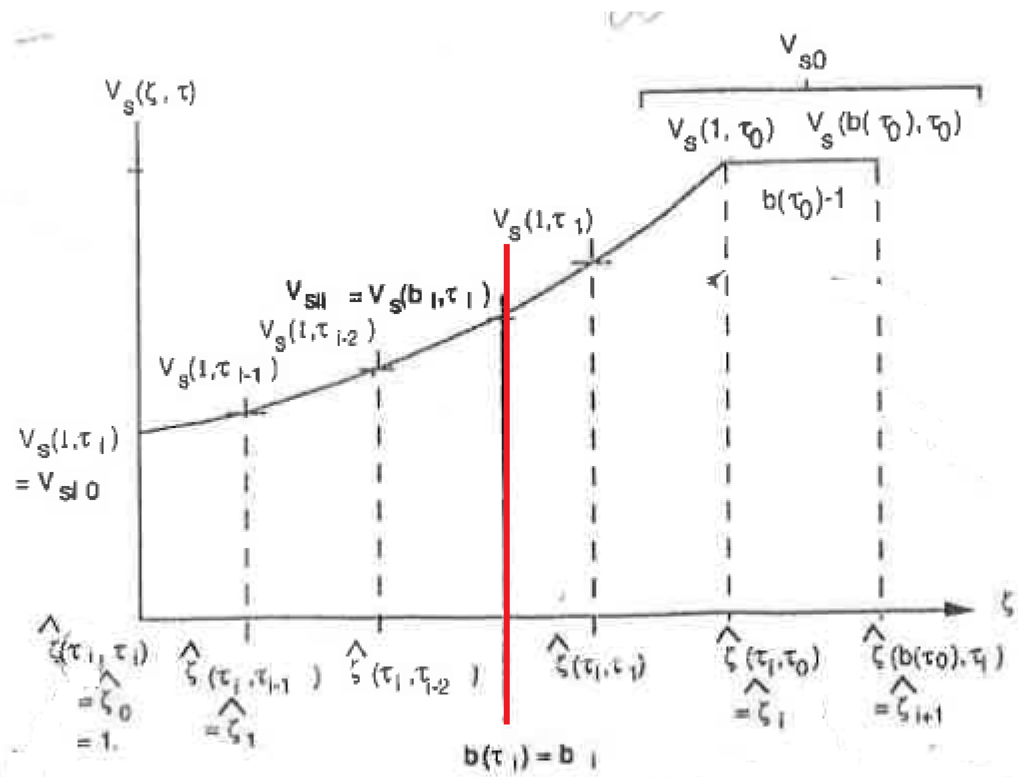


Figure 4: V_{sii} from ζ distribution (Vorus, 1996)

Δb_{i1} is the segment added at time step i at ζ=1. The unknowns in the problem are $V_s(1,\tau)$, Δb_{i1} in addition to Δτ or $z_{ci\tau}$ in case of chine-unwetted flow problems. All other Δb_{ij} s are known from data from previous time steps. All $V_{sij} = V_s(\zeta_{ij},\tau_i)$ for j>0 are also known from previous steps and V_i is externally specified.

For chine wetted case, the z_{ci} is fed as an input to the solution. The value of z_{ci} sets the value for $\Delta\tau_i$.

3.5.2. Velocity continuity

The velocity continuity requirement at ζ=1 as expressed in equation (11) is discretized. The discretized form of this equation is:

$$0 = V_i + \frac{1}{2\pi\bar{\lambda}_i}\sum_{j=1}^{n_i}\frac{\gamma_{sij}}{\kappa_{ij}}(T_{ij} - T_{i\,j-1}) \qquad (20)$$

with

$$T_{ij} = (\zeta_{ij}^2 - 1)^{\bar{\lambda}_i}\, F\left(\bar{\lambda}_i\,, \bar{\lambda}_i, \bar{\lambda}_i + 1, 1 - \zeta_{ij}^2\right) \qquad (21)$$

The function F appearing in the equation is the hypergeometric function of argument $1\text{-}\zeta_{ij}^2$. The other quantities in the expression are:

$$\bar{\lambda}_i \equiv \frac{1}{2} - \frac{\tilde{\beta}_{iK}^*}{\pi} \quad \text{with } \tilde{\beta}_{ik}^* = \tan^{-1}\left(sin\beta(\zeta_k^*, \tau_i)\right), \quad 0 \le \zeta_k^* \le 1 \qquad (22)$$

$$\bar{\kappa}_{ij} \equiv \bar{\kappa}_i\left(\bar{\zeta}_{ij}\right) = \left(\frac{\bar{\zeta}_{ij}^2 - 1}{\bar{\zeta}_{ij}^2}\right)^{\frac{-\tilde{\beta}_{iK}}{\pi}} \prod_{k=1}^{K}\left|\frac{1 - \bar{\zeta}_{ij}^2}{\bar{\zeta}_{ij}^2}\right|^{\frac{\tilde{\beta}_k^*}{\pi}} \qquad (23)$$

$$\bar{\zeta}_{ij} \equiv \frac{1}{2}\left(\zeta_{ij} + \zeta_{i\,j-1}\right) > 1 \qquad (24)$$

The equations for velocity continuity provide a relation for definition of γ_{si1}. This could be viewed as eliminating the unknown V_{si0} (as γ_{si1} is defined in terms of V_{si0} and V_{si1}) in terms of $\Delta b_{i1.}$

3.5.3. Displacement continuity

The displacement condition is employed only in case chine unwetted flows as the value of Y_{wl} is a function of both the height of point C (h_c) on the contour (which could vary in the case of chine unwetted flow) as well as the time step. This is not the case for computation of Y_{wl} in chine unwetted flows as h_c and z_c become constants after chine wetting.

The discretized equation for definition of Y_{wl} is

$$R_i = Y_{wli} = \frac{2}{\pi} \sum_{j=1}^{K} \frac{\cos \tilde{\beta}_{ij}^{*}}{\bar{\kappa}_{ij}} \left[\left(h_{cij}^{*} - S_{ij} z_{bi} \xi_j^{*} \right) \left(P_{1i\,j+1} - P_{1\,ij} \right) + S_{ij} z_{bi} \left(P_{2i\,j+1} - P_{2\,ij} \right) \right] \qquad (25)$$

The definitions of h_c, S_{ij}, κ_{ij}, P_{1ij} and P_{2ij} are defined by:

$$h_{ci}^{*}(\xi) = h_{cij}^{*} + S_{ij} z_{bi} \left(\xi - \xi_j^{*} \right); \quad \xi_j^{*} \le \xi \le \xi_{j+1}^{*}; \quad j = 1, K \qquad (26)$$

$$S_{ij} = \tan \beta_{ij}^{*} \qquad (27)$$

$$\bar{\kappa}_{ij} = \left(\frac{1 - \bar{\xi}_j^2}{\xi_j^2} \right)^{\frac{-\tilde{\beta}_{iK}}{\pi}} \prod_{k=1}^{K} \left| \frac{\bar{\xi}_j^{*\,2} - \xi_{k+1}^{*2}}{\bar{\xi}_j^{*\,2} - \xi_k^{*2}} \right|^{\frac{\tilde{\beta}_k^{*}(\tau)}{\pi}} \qquad (28)$$

$$P_{1ij} = \frac{\xi_j^{*\,2(1-\bar{\lambda}_i)}}{2(1-\bar{\lambda}_i)} \quad F\left(1 - \bar{\lambda}_i, 1 - \bar{\lambda}_i, 2 - \bar{\lambda}_i \; ; \xi_j^2\right) \qquad (29)$$

$$P_{2ij} = \frac{\xi_j^{*\,2(\frac{3}{2}-\bar{\lambda}_i)}}{2(\frac{3}{2}-\bar{\lambda}_i)} \quad F\left(1 - \bar{\lambda}_i, \frac{3}{2} - \bar{\lambda}_i, \frac{5}{2} - \bar{\lambda}_i \; ; \xi_j^2\right) \qquad (30)$$

The relation of R_i to the time step $\Delta\tau_i$ is given by

$$\int_{\tau 0 = \tau i-1}^{\tau i} V(\tau_0) d\tau_0 = -Y_{wl\,i-1} + R_i \qquad (31)$$

Converting this equation to discretized form, we get a definition for $\Delta\tau_i$

$$\Delta\tau = \frac{1}{\dot{V}_{i-1}}\left[-V_{i-1} + \sqrt{V_{i-1}^2 - 2\dot{V}_{i-1}(Y_{wli-1} - R_i)}\right] \qquad (32)$$

For constant velocity cases, the time step value is given by

$$\Delta\tau = (-Y_{wli-1} + R_i)/V_0 \qquad (32a)$$

3.6. Cylinder Pressure and force distribution

The general pressure coefficient is given by (12). Back substituting the burger's equation (13) gives the formulation for C_p:

$$C_{pi}(\zeta) =$$

$$\frac{1}{4}[\gamma_{ci}^2(1) - \gamma_{ci}^2(\zeta)] - z_{ci\tau}\left[\int_{\zeta_0}^{1}\gamma_{ci}(\zeta_0)d\zeta_0 + \zeta\,\gamma_{ci}(\zeta) - \gamma_{ci}(1)\right] - z_{ci}\int_{\zeta 0=\zeta}^{1}\gamma_{ci\tau}(\zeta_0)d\zeta_0 \qquad (33)$$

The value of the vortex element strength in the equation is computed from the formulation given in equation (9), which is result of application of nonsingular contour vortex distribution requirement, combining pressure and velocity continuity. The discretized form is as shown below:

$$\gamma_{ci}(\zeta) = \bar{\kappa}_i(\zeta)\sum_{j=1}^{n_i}\left[S_{ij}(\zeta) - S_{i\,j-1}(\zeta)\right] \qquad 0 \le \zeta \le 1 \qquad (34)$$

Where j-summation is over i elements of the vortex sheet at τ_i. F is the hypergeometric function. S_{ij} and Q_{ij} are defined as:

$$S_{ij}(\zeta) \equiv \frac{cos\tilde{\beta}_i(\zeta)}{\pi\,\bar{\lambda}_i}Q_{ij}(\zeta)^{\bar{\lambda}_i}\;F\left(\bar{\lambda}_i\,,\bar{\lambda}_i,\bar{\lambda}_i + 1\,;Q_{ij}\,(\zeta)\right) \qquad (35)$$

$$Q_{ij}(\zeta) \equiv \frac{\zeta^2(\zeta_{ij}^2-1)}{(\zeta_{ij}^2-\zeta^2)} \qquad (36)$$

The time derivative is given by the relation $\gamma_{c\tau} = (\gamma_{ci} - \gamma_{ci-1})/\Delta\tau$

3.7. Initial Condition

The problem initialization is performed on linear assumptions. The contour of the body is assumed to have constant nonzero deadrise angle β in the immediate vicinity of the keel. The velocity of impact in the small time after the initial impact is considered to be a constant velocity, V_0. The initial flow in this time interval, $0 \leq \tau \leq \tau_0$, is considered to be a wedge similarity flow. The waterline line level, Y_{wl} is given by the simple relation $Y_{wl} = V_0\, \tau$ for $0 \leq \tau \leq \tau_0$.

The three continuity requirements could be applied to this condition to get the starting values for the non-linear solution.

3.7.1. Pressure continuity

For wedge similarity flow, $V_s(\zeta,\tau)=V_s(\zeta)$, i.e., it is independent of time. This constant jet velocity, denoted as V_j, gives the non-dimensional jet head velocity:

$$z_{bi\tau} = \frac{V_j^2-1}{2V_j} \qquad (37)$$

3.7.2. Velocity continuity

Equation (46) reduces to an equation with a single segment at the initial time step τ_0. The vortex strength would be defined by $\gamma_{sij} = \gamma_s = -2V_j$. At i=0; $\zeta_{ij} = \zeta_{01} = b(\tau_0) = b_0 = b$, and $\kappa_{01} = 1$, so (20) reduces to:

$$1 + \frac{\gamma_s}{\pi}\frac{(b^2-1)^\lambda}{2\lambda}\, F\left(\bar{\lambda}_i\,, \bar{\lambda}_i, \bar{\lambda}_i + 1\,; 1 - b^2\right) = 0 \qquad (38)$$

with

$$\lambda \equiv \frac{1}{2} - \frac{\tilde{\beta}}{\pi}\ ;\ \tilde{\beta}\ \equiv\ \tan^{-1}(\sin\beta)$$

3.7.3. Displacement continuity

In (25), K=1, $\beta^*_{01} = \beta$, $h_{c01} = z_{b0} \tan\beta$, κ01 = 1, ξ=0, $P_{201} = 0$, giving

$$V_0\tau \equiv \frac{2}{\pi}\cos\tilde{\beta}\tan\beta\, z_{b0}P_{202} \qquad (39)$$

Taking $z_{b0} = z_{b\tau}\,\tau_0$. This gives a value for zbτ in terms of the values of β as shown in (40). This would serve as the starting point of the solution process.

$$1 = \frac{2}{\sqrt{\pi^3}}\cos\tilde{\beta}\tan\beta\;\, z_{b\tau}\, T(\lambda)\, T\left(\frac{3}{2} - \lambda\right) \qquad (40)$$

3.8. Computation of pressure coefficients over a 3D contour - Linear Strip Theory

Strip theory is applied to get the pressure distribution on the contour. The hull contour is discretized into equi-spaced stations. The 3D hull surface is discretized into a series of stations (2D sections). Vorus' theory is only applicable to 2D sections, so the theory is applied to individual stations, and the results finally combined to get a pressure distribution. Figure 5 shows a sample discretization of a hull.

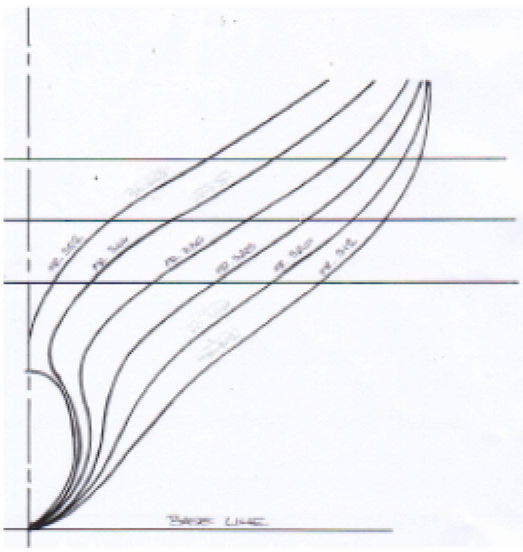


Figure 5: Discretization of 3D hull form into 2D sections (Bajic et al, 2010)

3.8.1. Time steps and pressure distributions

The value of time at time step i differs with the contour chosen for analysis. Station 1 may be at time τ_{i1} at time step i, whereas station 2 may be at time τ_{i2} at time step i. The only case when $\tau_{i2} = \tau_{i1}$ is when the contours are both similar and z_{c0} is same in both cases. In all other cases, for obtaining the distribution of pressure at a time $\tau = \tau_p$, the step corresponding to time $\tau = \tau_p$ is ascertained individually for each station (say $s_1, s_2, s_3, \ldots, s_n$) The pressure distribution at time step s1 is used as the pressure distribution at station 1. Similarly, the pressure distribution at time step s2 is used for station 2, and so on. In this case, $\tau_{s1} = \tau_{s2} = \tau_{s3} = ... = \tau_{sn} = \tau_{p.}$ The pressure distributions at the time steps mentioned above are combined to get the pressure distribution. Figure 6 shows a graphical representation of the non-dimensionalized pressure distribution, with the non-dimensionalized stations collapsed onto a plane:

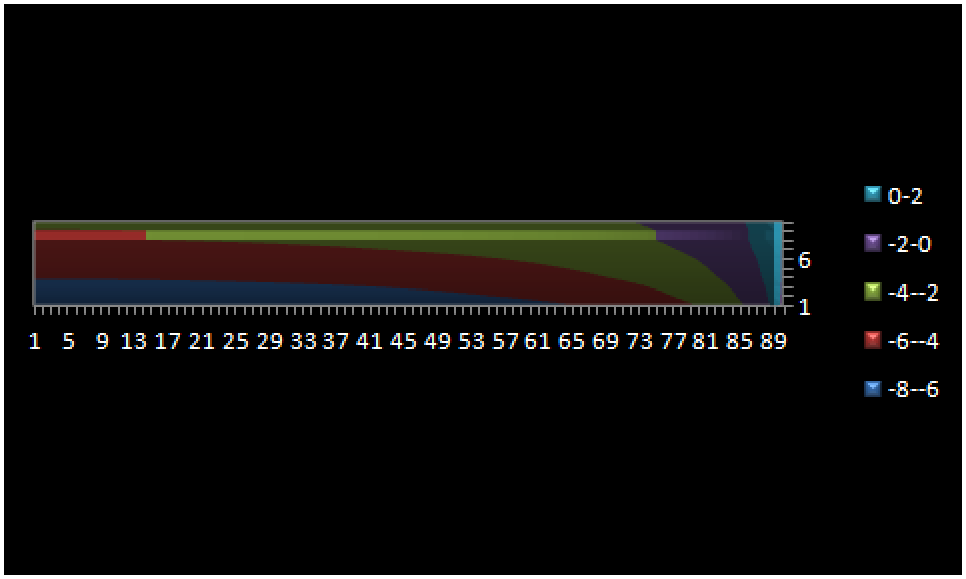


Figure 6: Sample pressure contour output after post processing in Excel

4. Solution methodology and algorithm

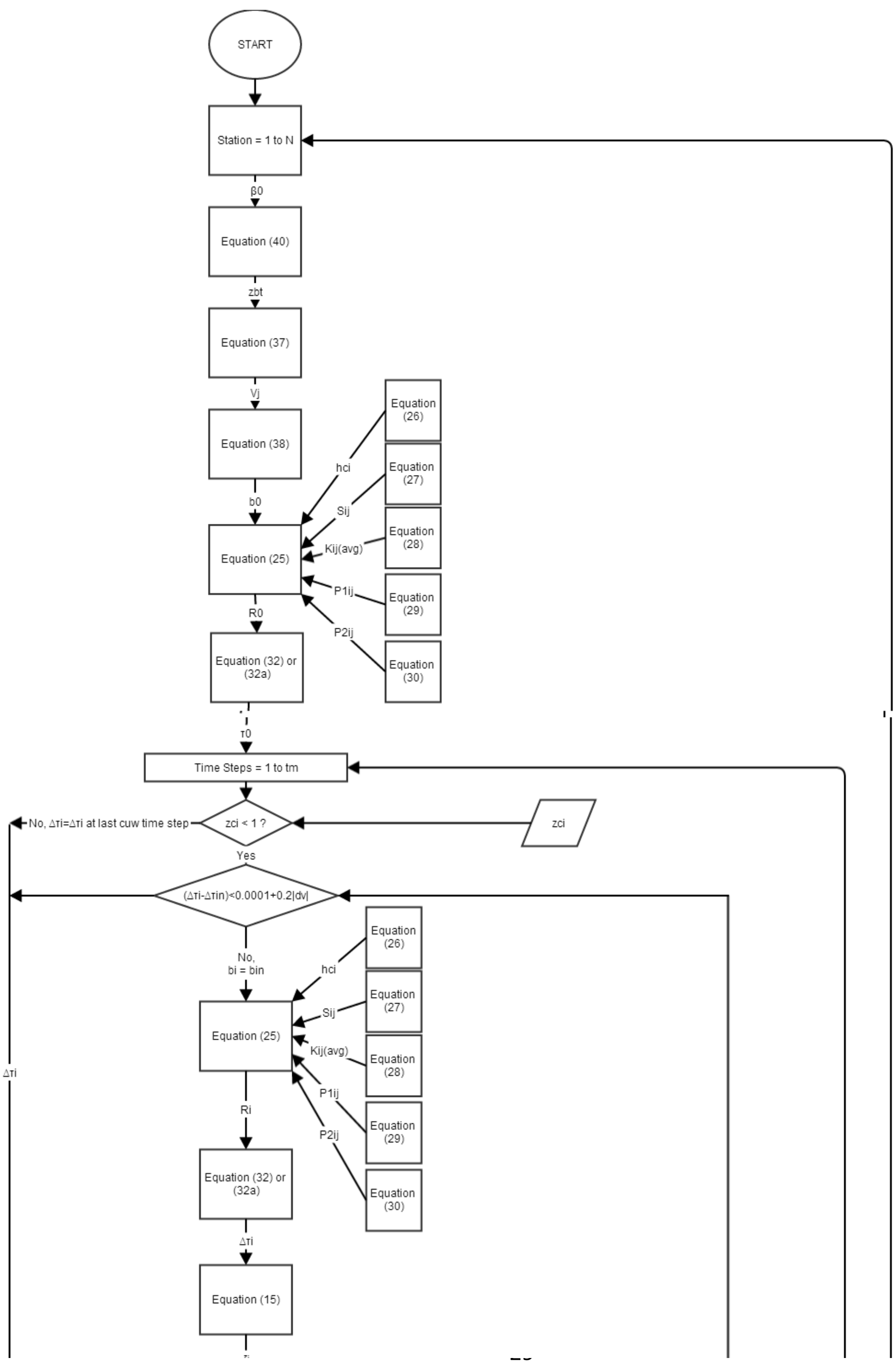
The problem is solved via a nested iteration process of the nonlinear system from the established values at the initial condition.

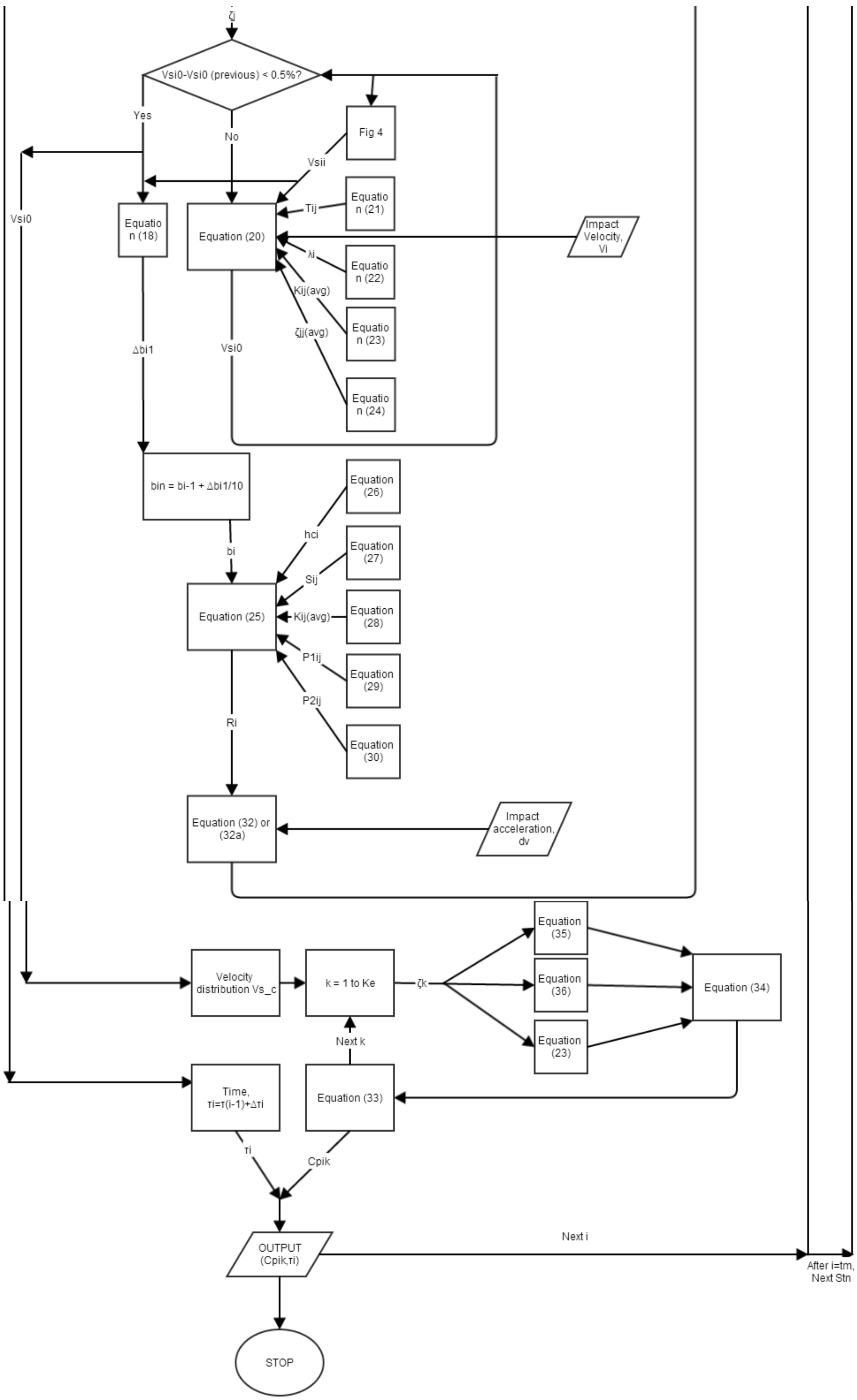
1. The initialization procedure follows the following order
   a. $zb\tau$ is obtained for the specific value of β at the deadrise from (40) using the displacement continuity condition
   b. The value of $z_{b\tau}$ is applied to (37) to obtain V_j form the pressure continuity condition
   c. This value of V_j is used to obtain the value of γ_s (= $2V_j$) This is subsequently applied to (38) to satisfy the velocity continuity. Solving this equation gives the initial value of b.
2. For the chine unwetted step i, z_{ci} is supplied externally as the input
3. b_{i-1} is used as the trial iterate of b_i. This gives the value of zbi (= z_{ci} b_i)
4. The value of R_i is using Equation (25). Solving (25) would require (26), (27), (28), (29) and (30).
5. The value of $\Delta\tau_i$ is computed for the given value of z_{ci} using (32) or (32a) depending the value of acceleration.
6. The computation of $\Delta\tau_i$ gives the time at time step i, $\tau_{i.}$ This quantity is required for computation of the ζ distribution using the relations given in (15).
7. The trial iterate of b_i gives a value for jet velocity at spray root, V_{sii} via interpolation depending on the value of b_i on the ζ_j distribution as shown in Figure 4.
8. This is used to compute the strength of the outermost element on vortex sheet, all other element strengths are available from velocity data from previous time steps.

9. This applied in equation (20) gives the value of velocity at the $\zeta=1$. Solving this requires (21), (22), (23) and (24).
10. The obtained value of Vsii also gives a value for Δb_{i1} from the expression (18)
11. The new value of bi is computed from bi-1 and Δb_{i1}. $b_i = b_{i-1} + \Delta b_{i1}$
12. The value for $\Delta\tau i$ is computed with this value of bi (using Equation (25)) and compared to the $\Delta\tau i$ previously obtained. If the difference is within the range, 0.0001+abs(acceleration) x 0.2, the solution proceeds to the next time step. Else the solution process returns to Step 6 to recompute values of V_{sii}, V_{si0}, Δb_i and $\Delta\tau_i$.
13. If the time step value is sufficiently close, velocity distribution obtained is used to compute the vortex strength on the contour, given by $\gamma_c(\varepsilon,\tau)$ by equation (34).
14. The value of γc in the present time step and the previous time step gives the time derivative by the relation $\gamma_{c\tau} = (\gamma_{ci} - \gamma_{ci-1})/\Delta\tau$
15. This vortex distribution gives the pressure coefficient distribution by equation (33).

4.1. Algorithm

The algorithm of the problem is as described in the following pages:




ζj
Vsi0-Vsi0 (previous) < 0.5%?
Yes
No
Fig 4
Vsii
Vsi0
Equation (18)
Equation (20)
Tij
Equation (21)
Impact Velocity, Vi
λi
Equation (22)
Kij(avg)
Equation (23)
ζj(avg)
Equation (24)
Δbi1
Vsi0
bin = bi-1 + Δbi1/10
Equation (26)
hci
Equation (27)
Sij
bi
Equation (25)
Kij(avg)
Equation (28)
P1ij
Equation (29)
P2ij
Equation (30)
Ri
Equation (32) or (32a)
Impact acceleration, dv
Equation (35)
Velocity distribution Vs_c
k = 1 to Ke
ζk
Equation (36)
Equation (34)
Equation (23)
Next k
Time, τi=τ(i-1)+Δτi
Equation (33)
τi
Cpik
OUTPUT (Cpik,τi)
Next i
After i=tm, Next Stn
STOP

5. Verification of accuracy of code

The code was verified against the curves shown in Dr Vorus' paper (1996). Since Dr Vorus' theory is the basis for the code, it is a requisite that the results predicted match the results described in his paper.

5.1. Velocity comparison

Figure 8 shows the non-dimensionalized velocity distributions on 3 planing hull sections: with a 20-20 contour, a 20-30 contour and a 20-10 contour against time.

Figures 9(a),(b) and (c) show the distribution obtained using the code.

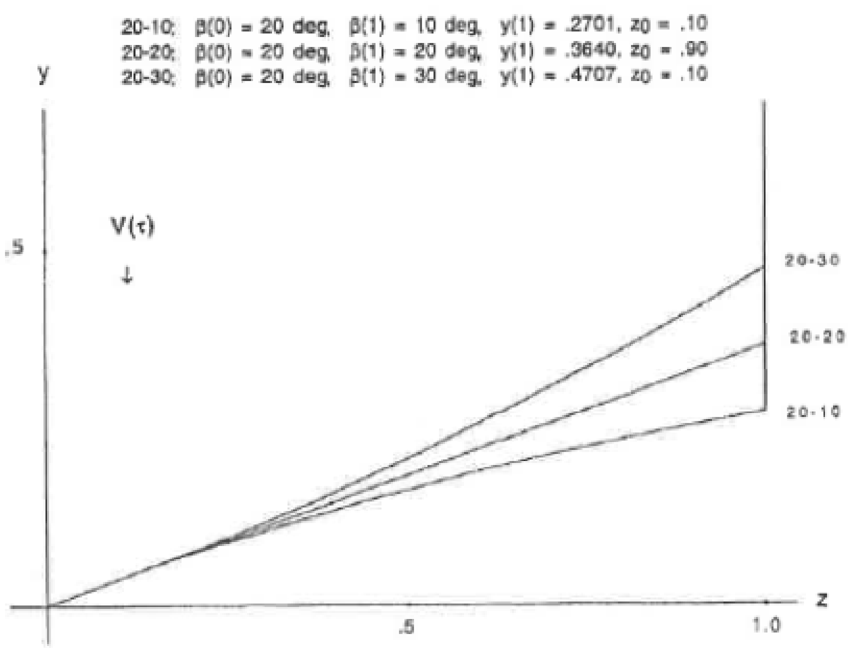


Figure 7: Hard chine contours (Vorus, 1996)

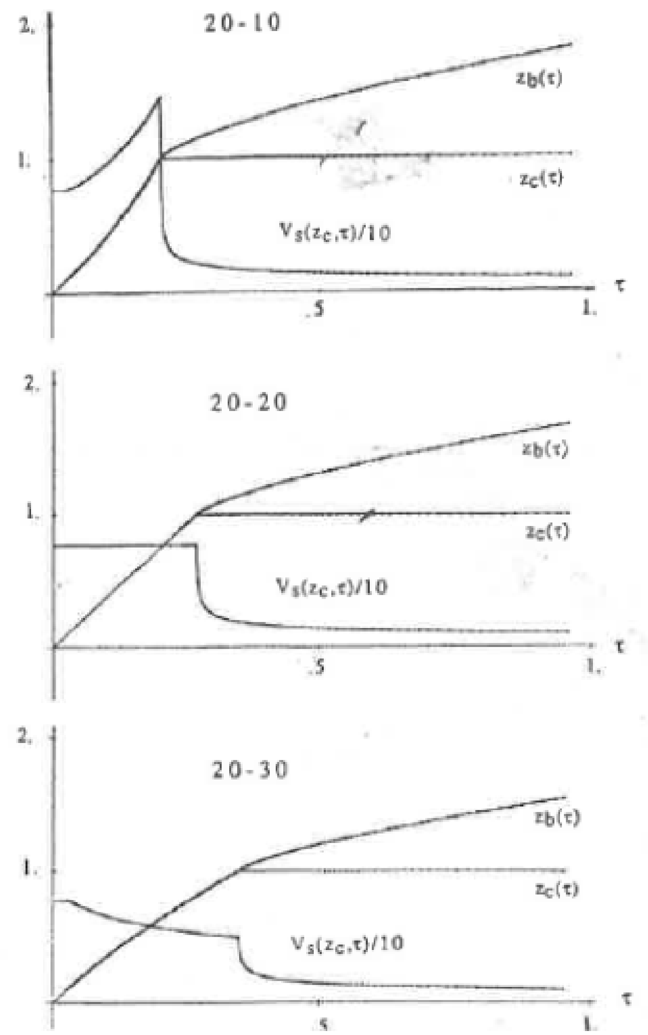


Figure 8: Particle velocity, z_c, and z_b (Vorus, 1996)

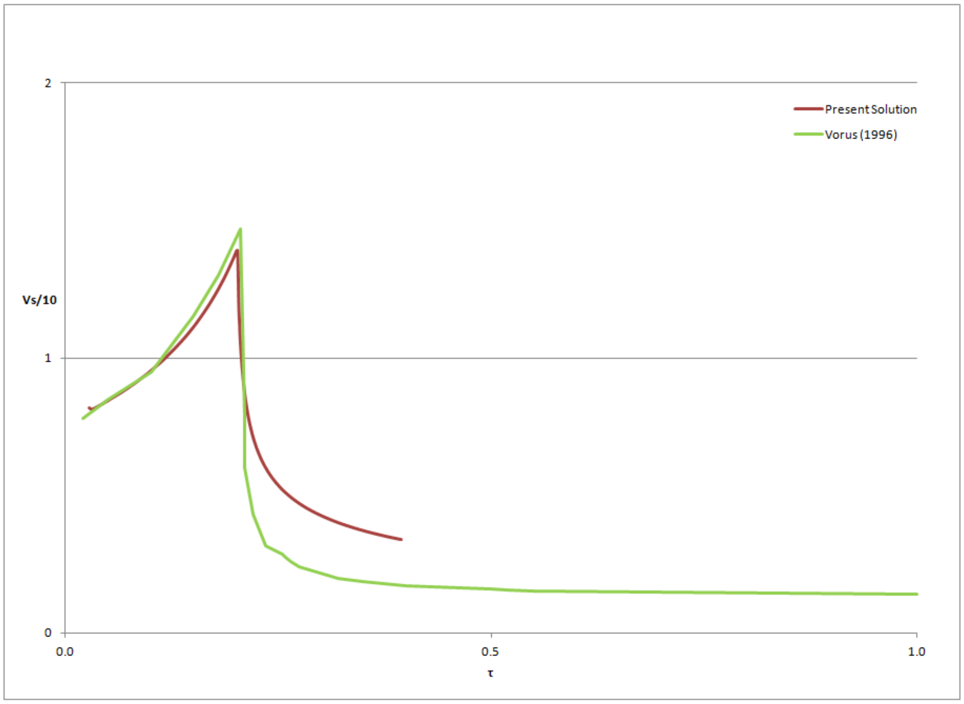


Figure 9(a): 20-10 Vs/10 distribution

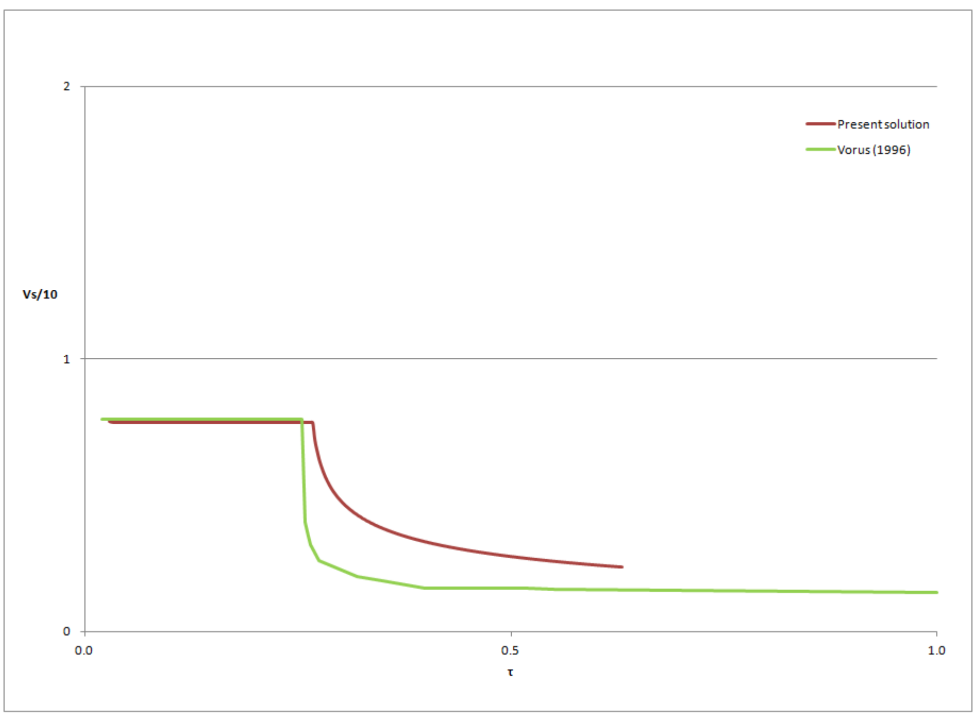


Figure 9(b): 20-20 Vs/10 distribution

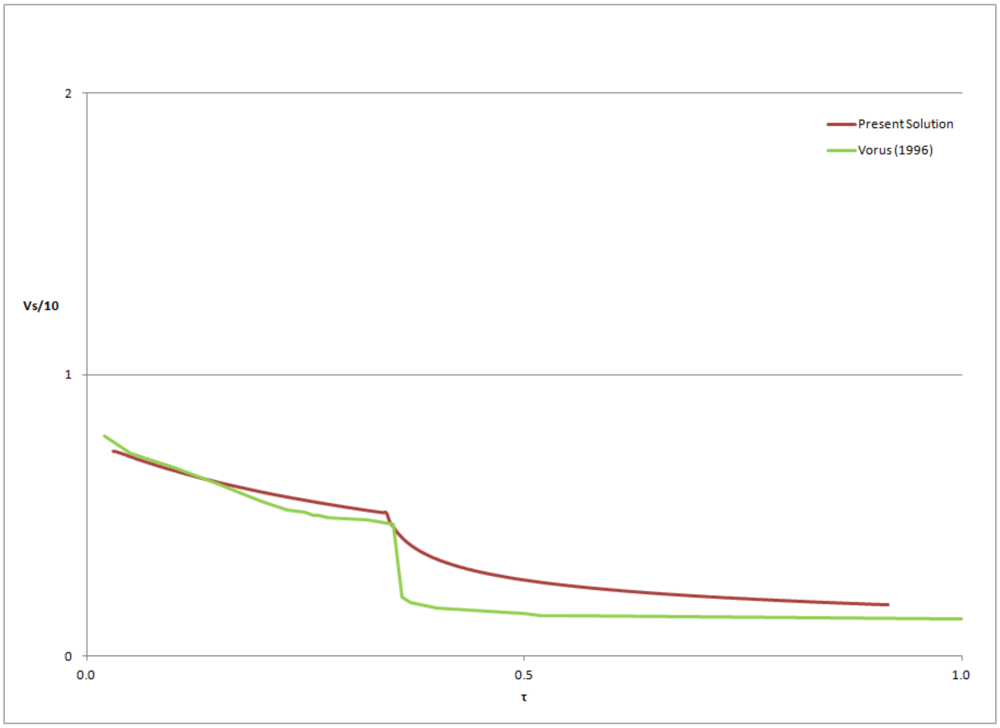


Figure 9(c): 20-30 Vs/10 distribution

A comparison between Figure 8, and Figures 9(a),(b) and (c) shows good agreement between the results obtained from the program and the results described in Vorus (1996).

5.2. Keel pressure comparison

The keel pressure coefficients on the contours as reported in Vorus (1996) are shown in Figure 10. Figures 11(a),(b) and (c) show the keel pressure coefficients on the hull contours, obtained from the code. The values reported by the code are slightly offset from the centerline as there is a pressure discontinuity along the centerline.

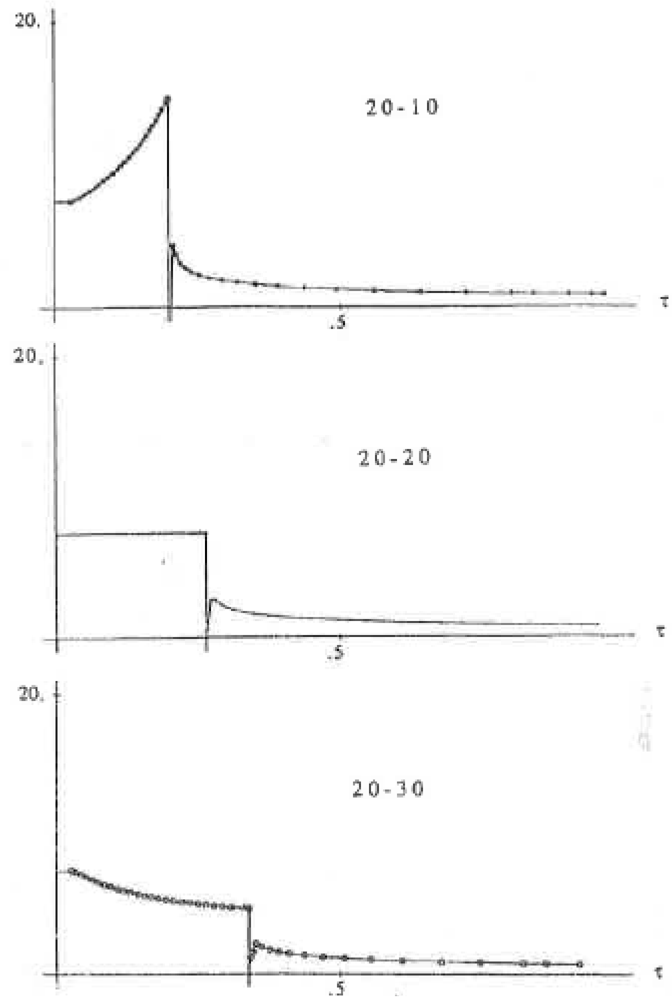


Figure 10: Keel pressure coefficient Cp (0,τ) (Vorus, 1996)

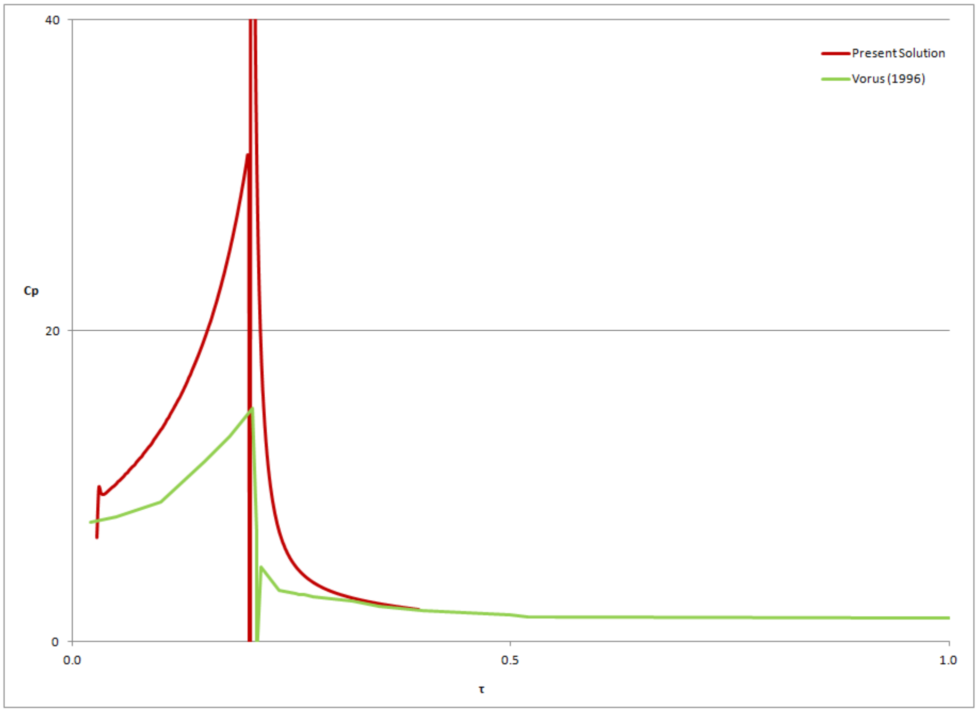


Figure 11(a): Keel pressure coefficient on 20-10 contour

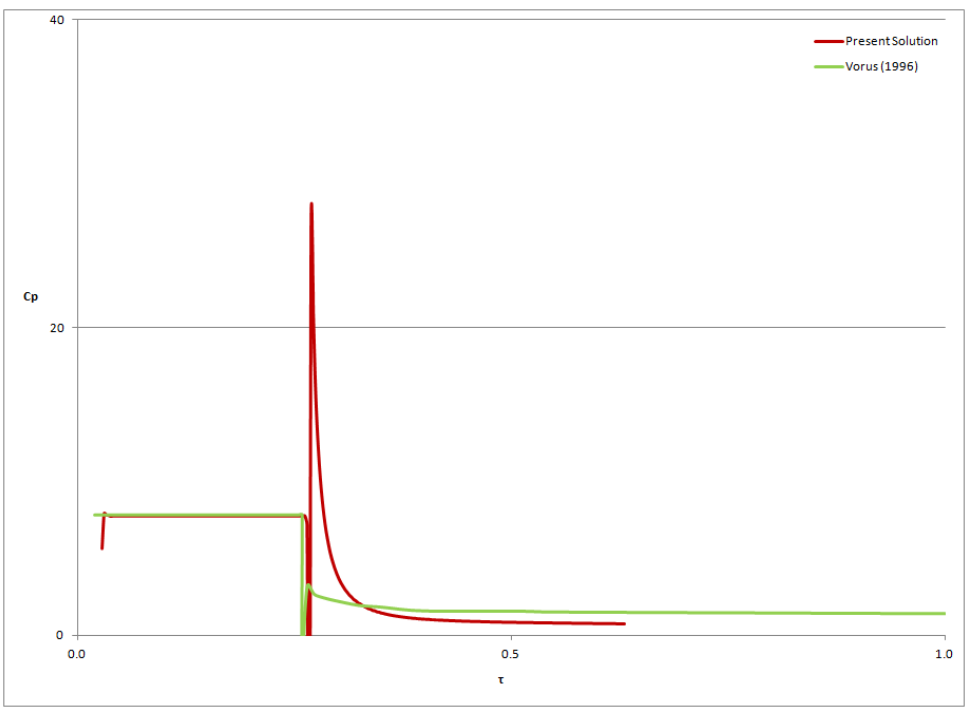


Figure 11(b): Keel pressure coefficient on 20-20 contour

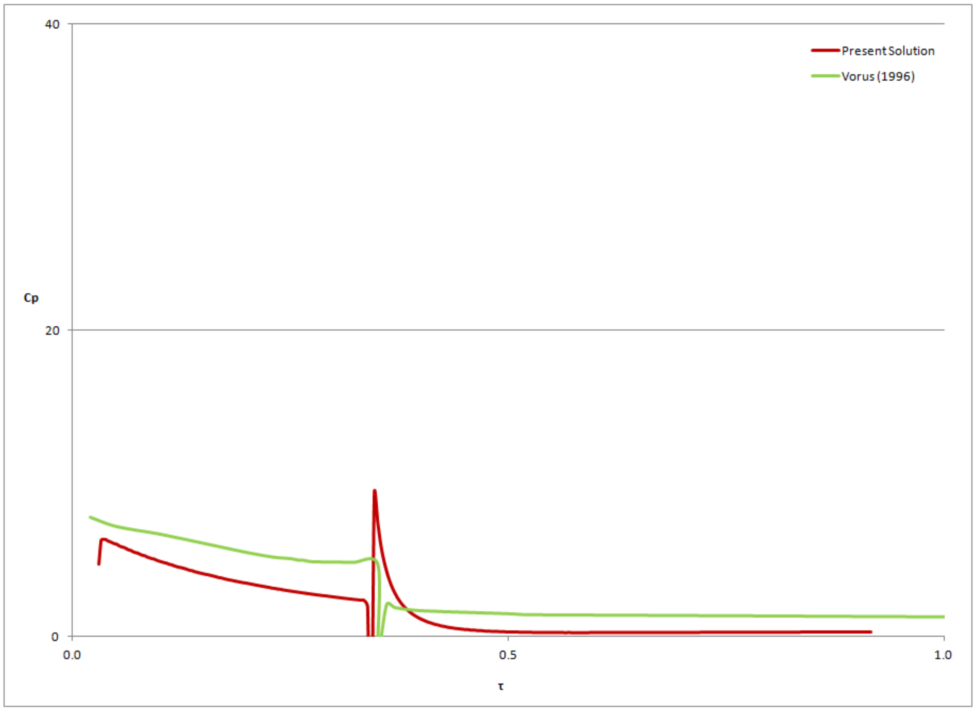


Figure 11(c): Keel pressure coefficient on 20-30 contour

A comparison between Figure 10 and Figures 11(a),(b) and (c) shows good agreement at the lower ends of the curve, i.e., a very short time after the impact for chine unwetted flow. The 20-20 contour shows good agreement with the results reported in Vorus (1996) throughout the time range.

However, the pressure coefficient at the keel is over-reported for the 20-10 contour as the flow approaches chine wetted flow. On the other hand, the pressure coefficient is underreported for the 20-30 contour as the flow approaches chine wetted flow. As the flow approaches separation, the pressure coefficient at the keel is the result of summation of the strength of vortcies distributed on the wetted surface. Underprediction / overprediction of the vortex strengths near the flow separation region of the wetted region would also influence the rate of change of strength of the vortex, which also influences the value of the pressure coefficient.. Thus the differences in the values of pressure coefficients are a result of amplification of errors in the computation of the vortex strengths near the flow separation region of the wetted portion of the hull.

6. Validation of predicted results against experimental results

Ghadimi et al (2013) have done a recent study of on the hullform used by Aarsnes (1996). The work compares the results presented by Aarsnes (1996) against a VOF scheme with FVM formulation. The set up used by Aarsnes used for the experiments is shown in Figure 12.

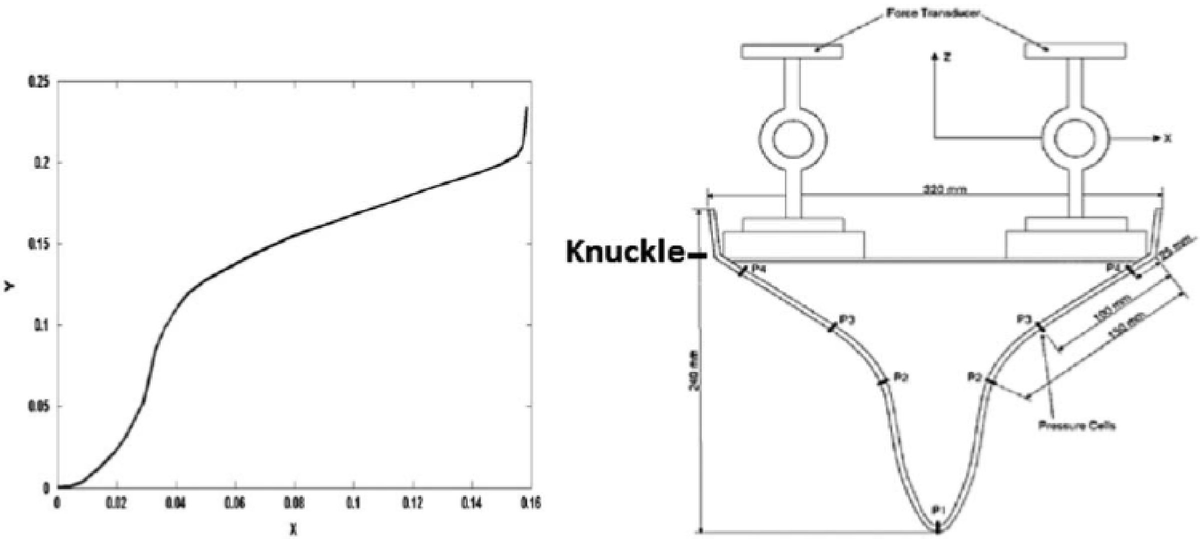


Figure 12: Bow section (left) and experimental setup (right) considered by Aarsnes(1996).

Ghadimi et al have compared the results of Aarsnes experimentation with their numerical simulation of the experiment at a constant velocity of 2.43 m/s. A pressure comparison has been done (Figure 13) against the results from experimentation. Figure 14 presents the pressure variation against the non-dimensionalized wetted portion of the hull as reported by Ghadimi et al (2013).

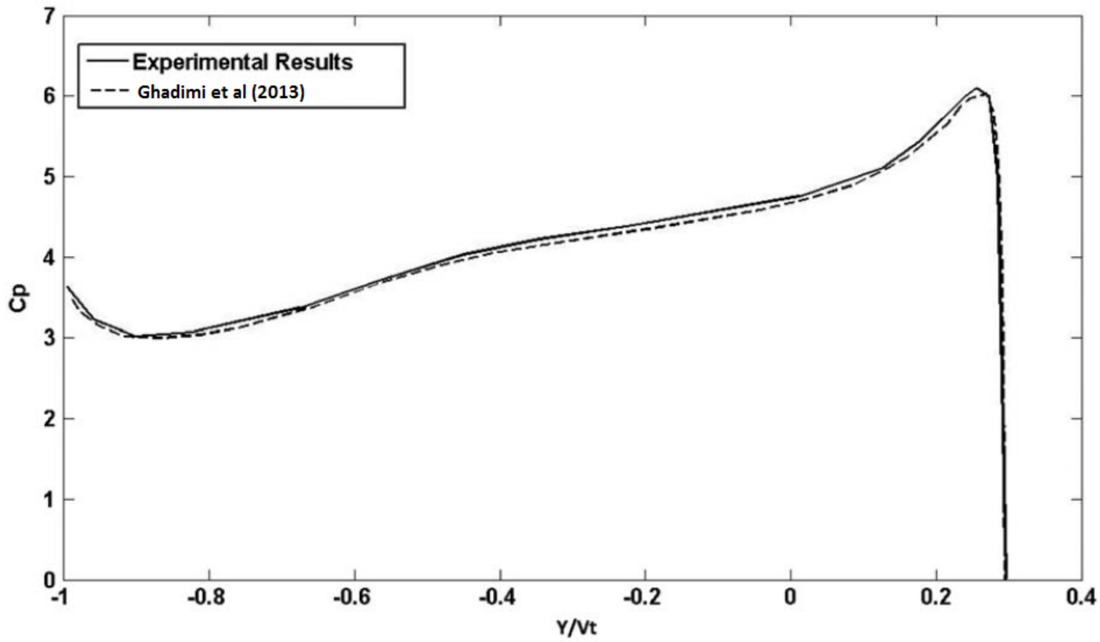


Figure 13: Pressure distribution on the bow section, experimental vs. VOF method (Ghadimi et al, 2013)

For the purpose of comparison, the hull section used by Aarsnes was discretized and the offsets were fed as input to the impakt 1.3 code. Figure 14 shows the discretization scheme employed for the feeding the offsets into the code. The discretized contour has been non-dimensionalized against the half breadth of the hull section.

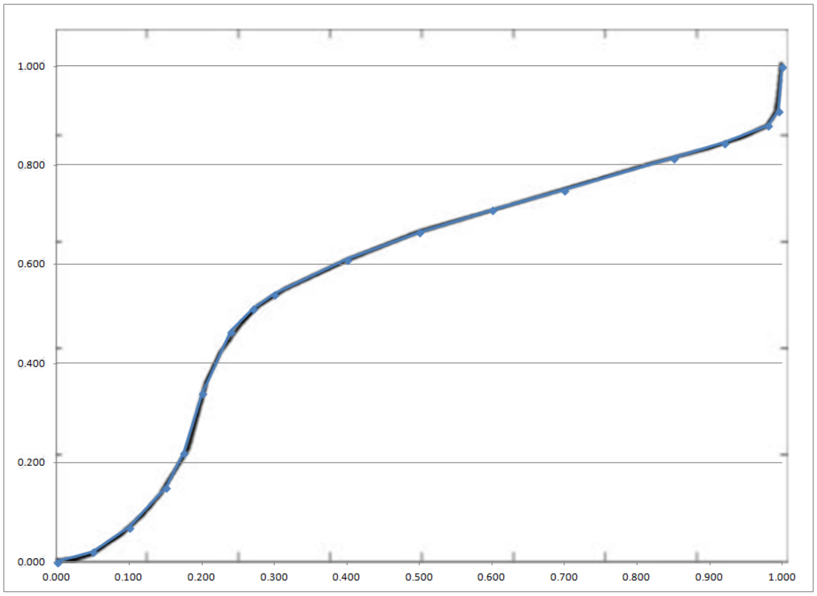


Figure 14: Discretized contour of the section used by Aarsnes

Figure 15 shows a comparison between the pressure predictions of the code and the pressure prediction from Ghadimi et al (2013) and Aarsnes (1996) experiments.

Figure 16 shows a comparison of the flow separations predicted by Ghadimi et al (2013) at t=0.06s and a particle flow history flow till the equivalent non-dimensionalized time $\tau = 0.66$ in the present code.

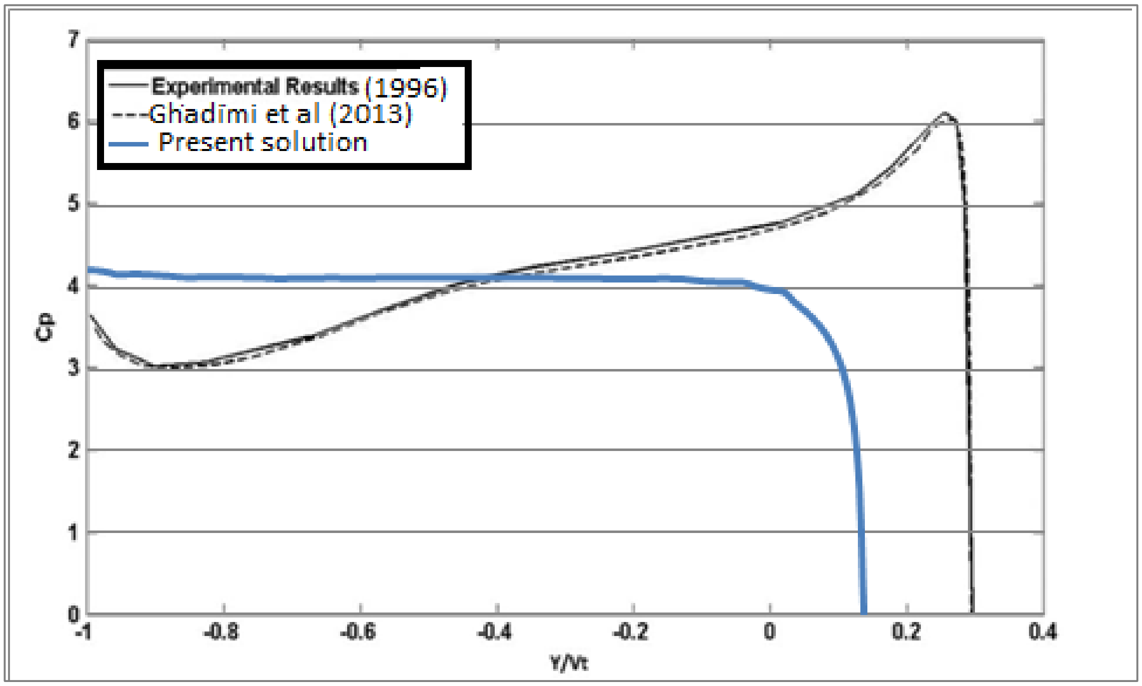


Figure 15: Comparison of code prediction versus Ghadimi et al (2013) & Aarsnes (1996) at τ=0.5351

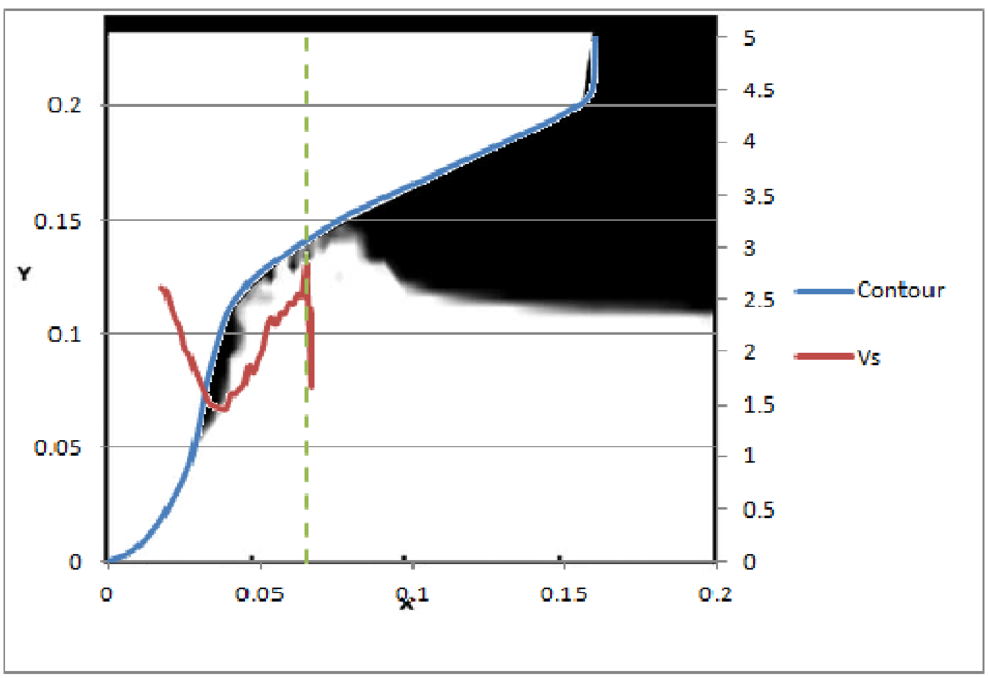


Figure 16: Comparison of fluid particle flow in Ghadimi et al (2013) and particle velocity in the present solution

6.1. Discussion of the comparison

The present code gives a good comparison between Ghadimi et al's model and the present code in terms of flow separation at the time specified. This is shown by the sharp drop in the flow particle velocity (Figure 17) at the green dotted line. This corresponds to the region in the flow simulation where flow separation was observed using the VOF scheme used in conjugation with FVM.

A comparison of the pressures gives a decent correlation at the lower ends of the curve, near the keel. As the flow approaches separation. The first peak in the experimental findings of Aarsnes (1996) and Ghadimi et al (2013) is a result of the primary impact of the body on the free surface. The second peak in the experimental curve is due to the impact of the separated flow on reattachment. Ghadimi et al (2013) have successfully predicted the reattachment and the resulting pressure peak on the hull surface The present code predicts the first peak, however, does not predict the second peak. This is due to the fact that the code does not capture reattachment of the flow and assumes the flow to be separated upon initial separation.

7. Time varying impact force variation in a hull form

Bajic et al(2010) reported results of slamming impact pressures on a container ship at different deck levels. A comparison was done between various class societies' rules regulating design pressures. Though not an exact comparison, the results presented by Bajic et al (2010) can be used as a basis of analysis of impact loads on the hull using a standardized hull form.

7.1. Characteristics of the hull form

The present analysis was performed on the hull used by Bajic et al (2010) for their analysis. Figure 17 shows the hull used for the analysis. A non-dimensionalization has been performed on the hull on the half breadth and depth of the outermost frame, Frame 312.

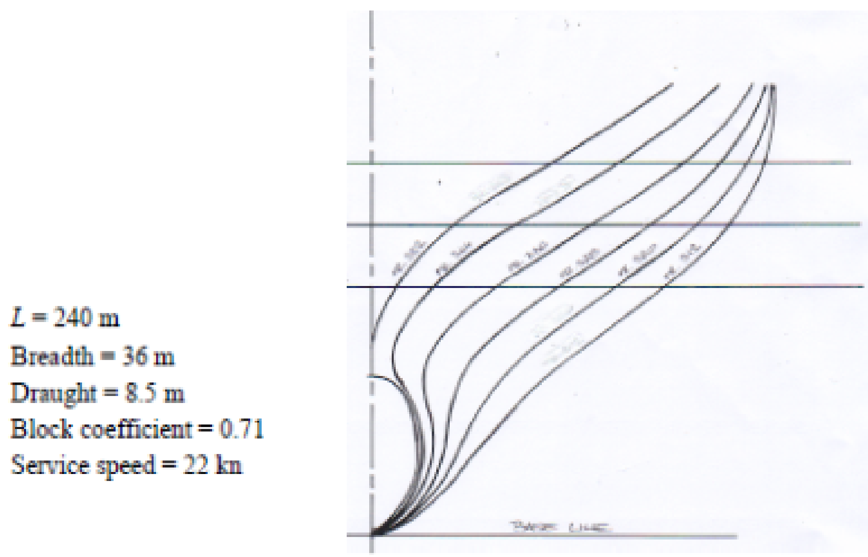


Figure 17: Bajic et al (2010) analysis hullform

The frames used for analysis, Frame 312, 320, 328, 336, 344 and 352, are at 91, 93, 96, 98, 100 and 103% from the aft end respectively.

7.2. Theoretical background

The equations of motion applied to a single station are applied to all the stations individually.

The present solution is not equipped to handle the effects of forward motion on impact forces, so at present the analysis is restricted to cases where the forward motion is zero. The heave velocity of all the stations would be the same. The vertical velocity of each station would be a superimposition of the velocity by virtue of heave as well as the pitching rate.

The time variation of the velocity would be also be a superimposition of the heave accelearation as well as the pitch acceleration.

$$V_i = V_0 + \ddot{z}t + LCF_i \times (\omega + \alpha t) \qquad (41)$$

Where Vo is the heave velocity of the hullform, $\ddot{z}$ is the heave acceleration, ω is the pitching rate and α is the pitch acceleration. LCFi is the distance of Station i from the LCF.

7.3. Problem set-up

The discretized hull form used for analysis is shown in Figure 18.

The impact force distribution on the frames at times τ= 0.07, 0.10, 0.20 and 0.30 are analyzed The waterlines at the analyzed time steps are shown in Figure 19. The discretized contour offsets are fed as input to the vsheet228.exe program. The post processing is performed using the rum1.exe file.

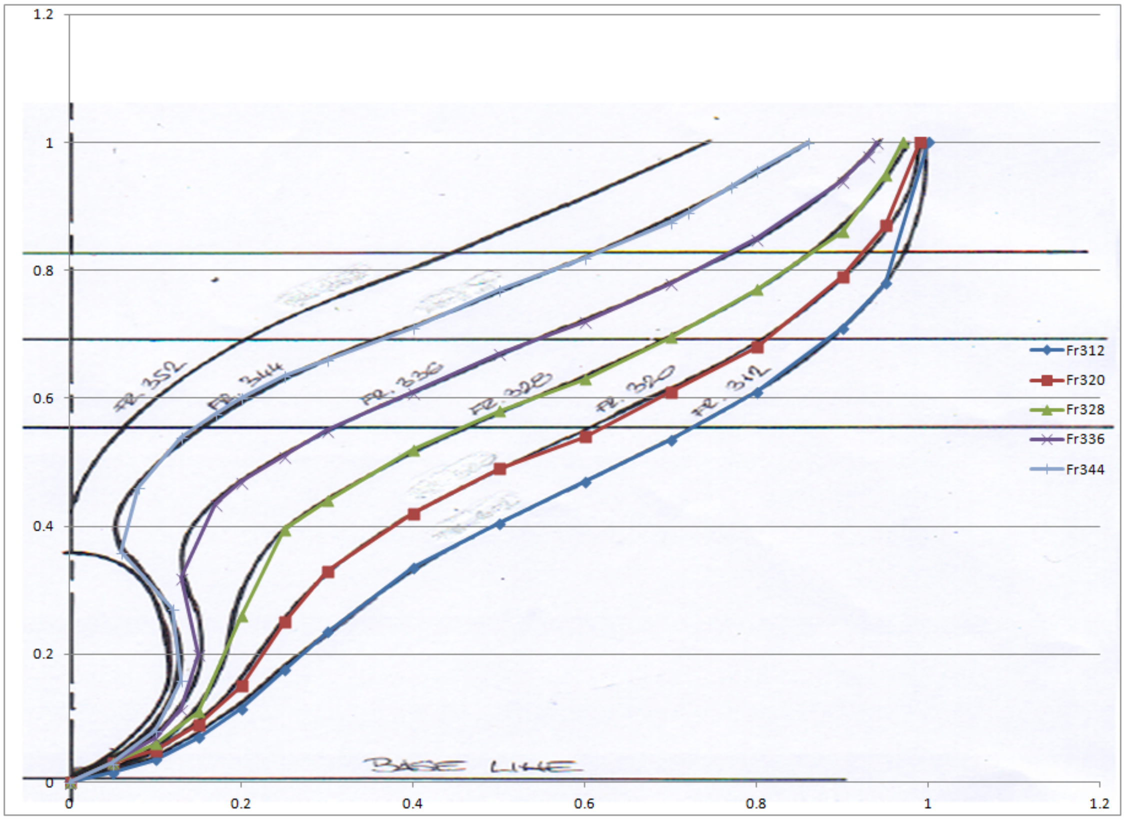


Figure 18: Discretized hull form contours

An important difference in the present analysis is that the initial draught in Bajic et al's analysis is 8.5m, whereas in the present analysis, the initial draught is 0 m, i.e., there is bow emergence. Another difference is that the rules have been applied on the vessel by Bajic et al at a service speed of 22 knots. The present analysis is performed at 0 knots forward speed using the present code. Bajic et al, however, do indicate the impact velocity calculated on the different frames. The assumption used here is that the impact velocity is the only factor that influences the pressure on the hullform, i.e., the coupling effect of forward velocity is ignored.

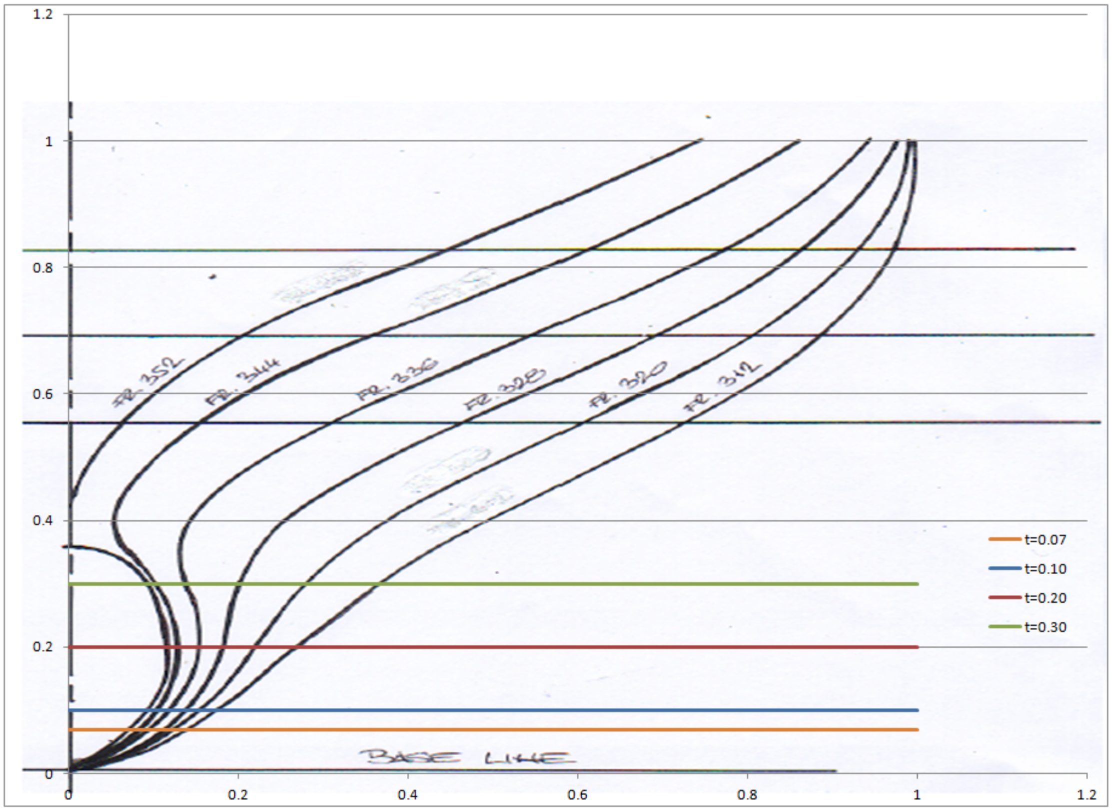
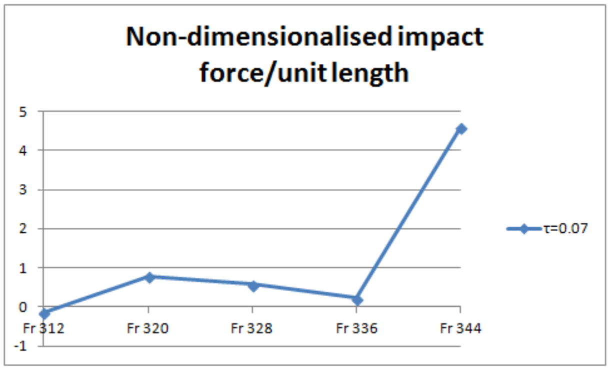


Figure 19: Waterline (YWL) at various time steps

7.4. Comparison and discussion of results

The impact force distribution along the hull at various time steps are shown in Figure 20 (a),(b),(c) and (d).



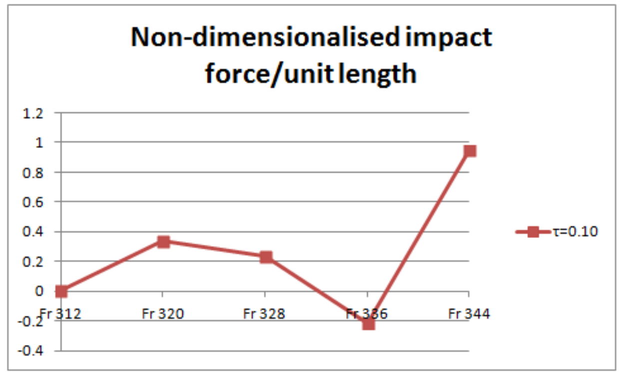
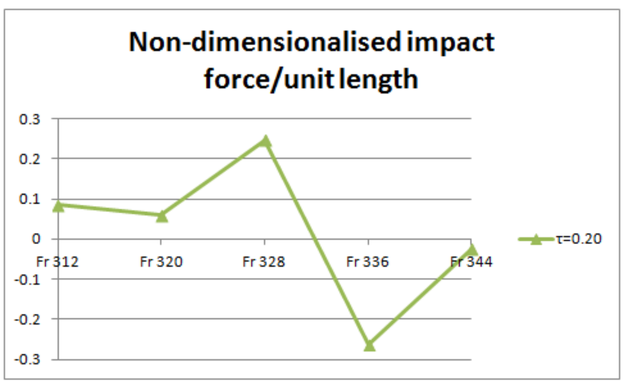


Figure 20(a) Impact force distribution at τ=0.07

Figure 20(b) Impact force distribution at τ=0.10



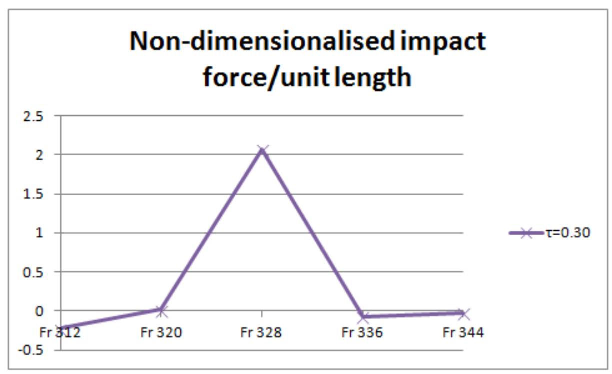


Figure 20(c) Impact force distribution at τ=0.20

Figure 20(d) Impact force distribution at τ=0.30

These results is compared with the results described in Bajic et al's results. The trend of slamming pressures predicted by various class societies' for level 1 should be a good indicator of the force acting on the frame in question. Figure 21 shows the pressure variation at level 1 (Bajic et al, 2010) along the hull. A comparison of the obtained results with those described by Bajic et al shows that the impact force predicted by the present code follows the predicted pressure variation by class societies' rules on slamming pressure.

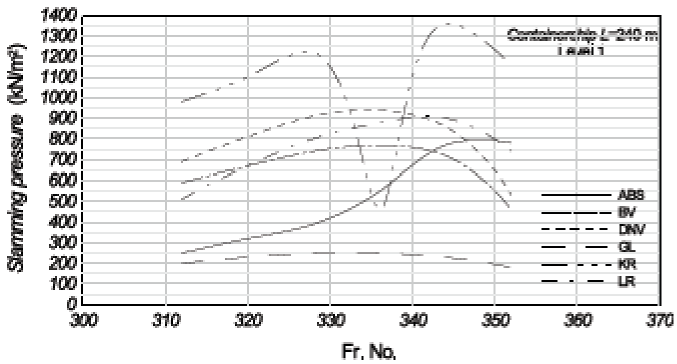


Figure 21: Pressure prediction using various class societies' rules (Bajic et al, 2010)

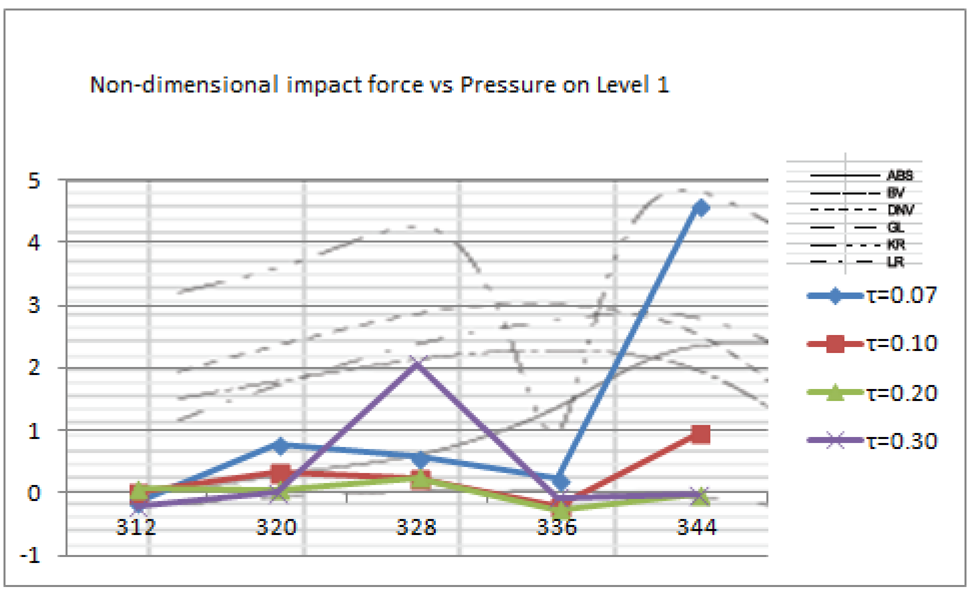
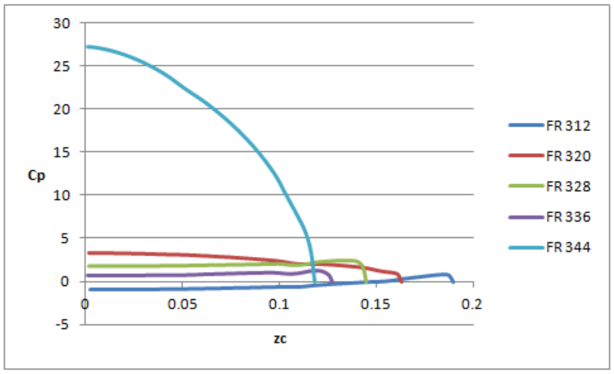


Figure 22: Comparison of impact force vs. Pressure on Level 1

The pressure variation along the span of the wetted portion of the hull at time steps τ = 0.07, 0.10, 0.20 and 0.30 are shown in Figures 23 (a), (b), (c) and (d).



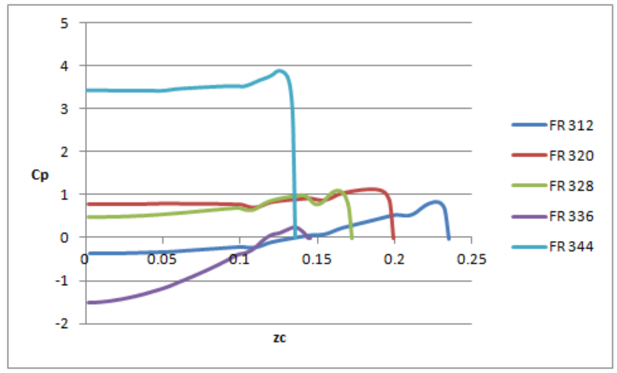
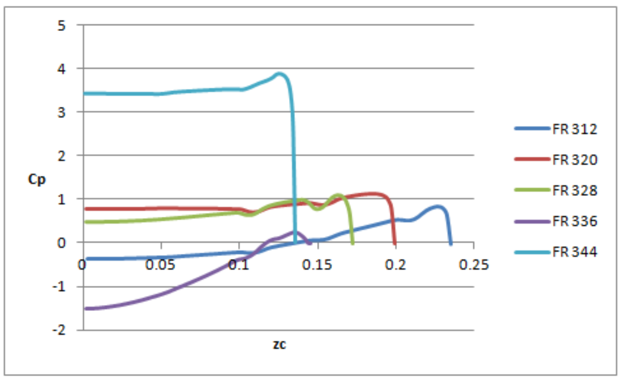


Figure 23(a): Cp distribution on frames at τ=0.07

Figure 23(b): Cp distribution on frames at τ=0.10



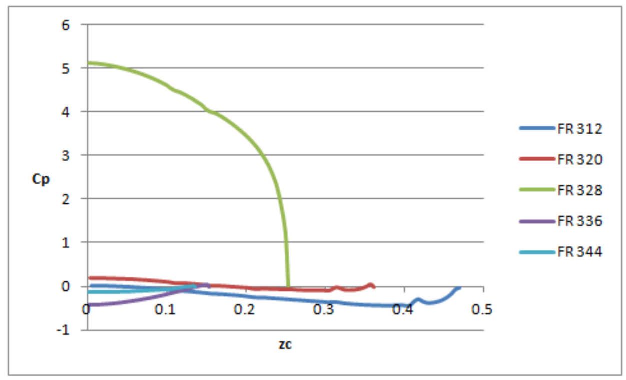


Figure 23(c): Cp distribution on frames at τ=0.20

Figure 23(d): Cp distribution on frames at τ=0.30

Contour pressure distributions have also been developed for the wetted portion of the hull at different time steps, τ=0.07, 0.10, 0.20 and 0.30.

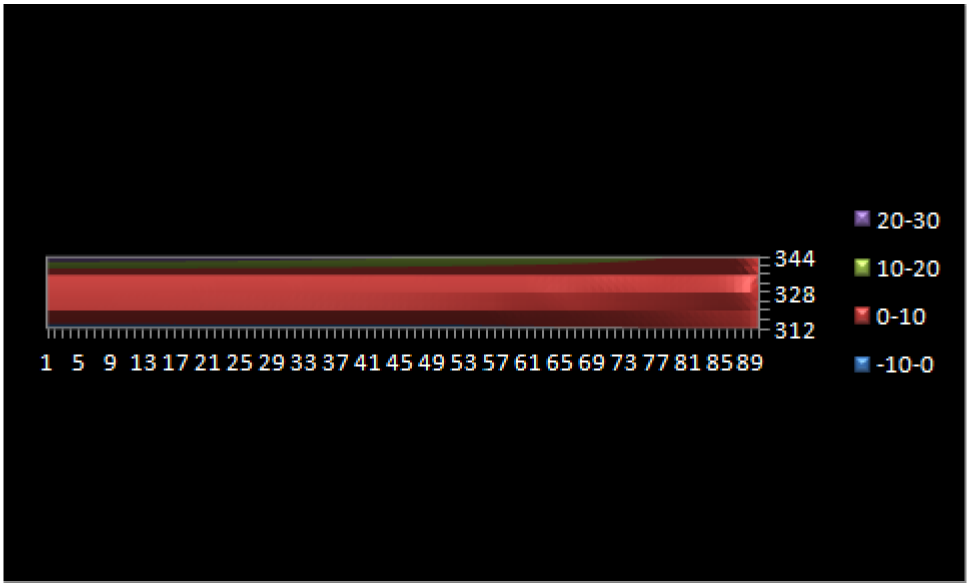


Figure 24(a): Non-dimensionalized Cp distribution on wetted contour of hull at τ=0.07

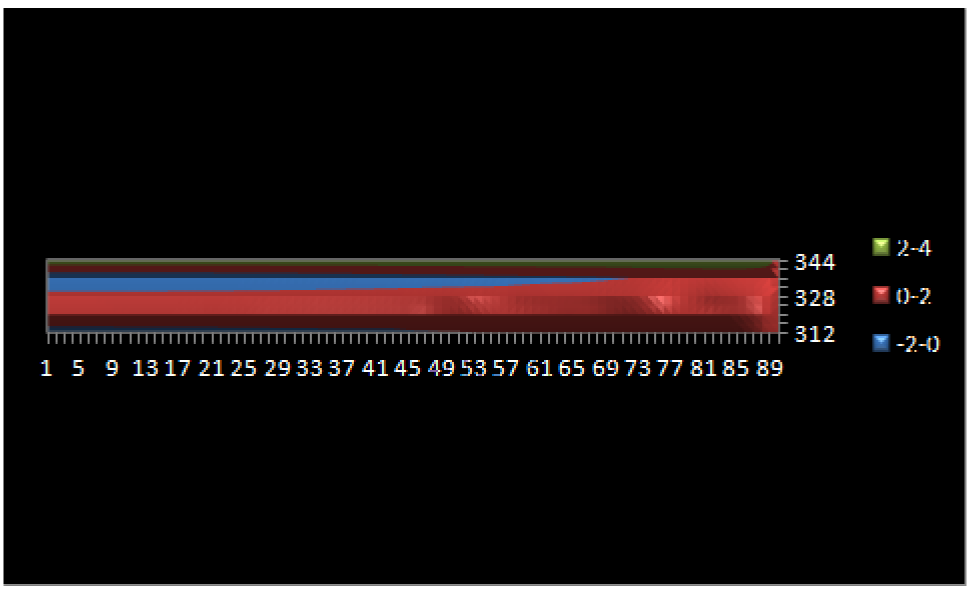


Figure 24(b): Non-dimensionalized Cp distribution on wetted contour of hull at τ=0.10

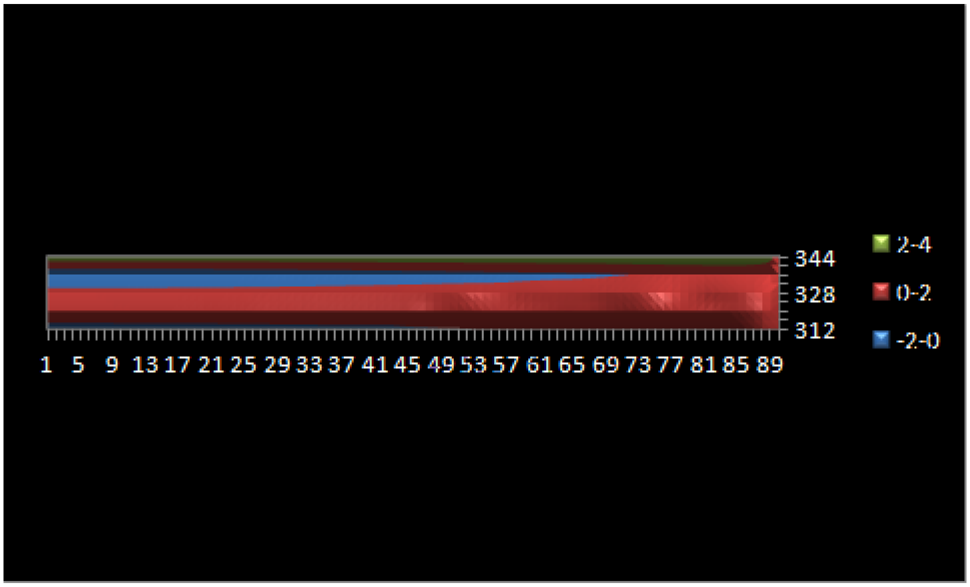


Figure 24(c): Non-dimensionalized Cp distribution on wetted contour of hull at τ=0.20

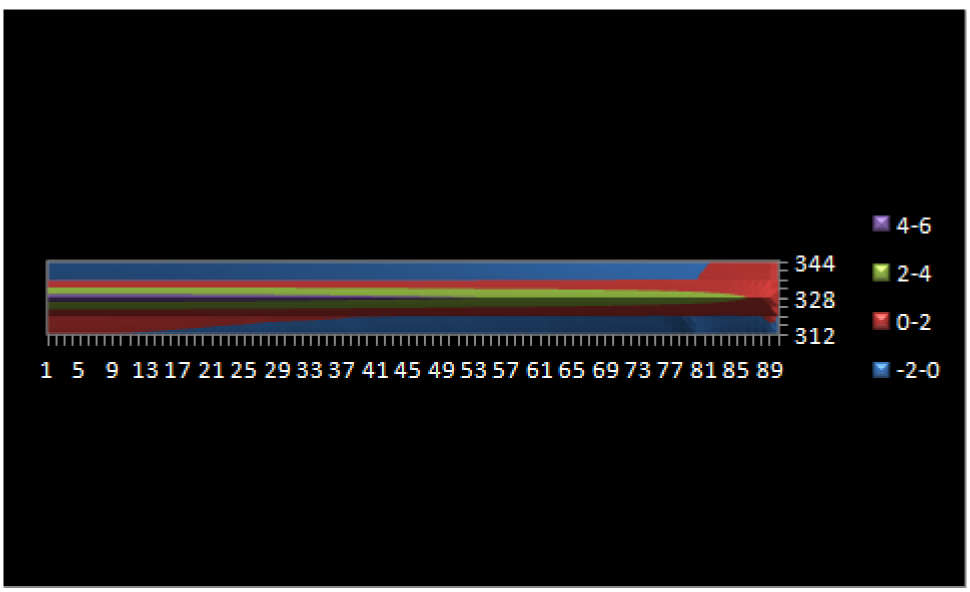


Figure 24(d): Non-dimensionalized Cp distribution on wetted contour of hull at τ=0.30

8. Conclusions

Vorus' theory was successfully implemented using the code developed in the present study. The code compares very well with Vorus' results (1996) and could be applied to shapes more complex than wedges.

The verification of accuracy of the code with regard to irregular shapes (such as a bow section) by comparison gives results that are sufficiently close. Aarsnes' (1996) experiments provided data for this verification. Different schemes are being developed for prediction of bow impact pressures and forces. Ghadimi et al (2013) have detailed a method for pressure and flow contour analysis. A good correlation was observed with the results predicted by Ghadimi et al. The difference in variation of pressure along the span can be traced to the inability of the Vorus' theory to incorporate effects of flow reattachment.

The application of the present theory to a 3D hullform using linear strip theory was also attempted in the study. Comparison of the obtained data with predicted pressures using class societies' rules on wave impact pressures as detailed in Bajic (2010) show that the trend of force variation along the hull follows a similar pattern as the pattern predicted by the societies for wave slamming pressures. Thus the theory confirms to the empirical laws employed by many societies and may be used to improve upon them.

8.1. Further research suggested

Further research suggested for the present theory from the analysis and data presented in this thesis are as follows:

(1) Incorporating effects of flow reattachment to the Vorus' theory. This would be helpful in predicting peak pressures near the point of flow separation.

(2) Investigation of coupling of impact velocities on the stations and the effect of this coupling on the force distribution on the hullform

(3) Incorporation of oblique velocities and gravity into the formulation. This would be helpful in investigation of cases of wave slamming in rough seas where the vessel undergoes forward translation in addition to the vertical slamming motion. A more comprehensive model for lift prediction can be built by combining the theories detailed in the present study and oblique velocity theories such as the theory detailed by Taravella & Vorus (2010).

9. Works Cited

[1] Von Karmen, T., and Wattendorf, F.L. The Impact on Seaplane Floats During Impact. NACA Technical Note No. 321, 1929

[2] Wagner, H., Phenomena Associated with Impact and Sliding on Liquid Surfaces. No 4. Vol 12, 1932.

[3] Maruo, H., High and Low-Aspect Ratio Approximation of Planing Surfaces, Schiffstechnik, 1967.

[4] Ogilvie, T.F., The Waves Generated by a Fine Bow Ship, Ninth Symposium on Naval Hydrodynamics, Paris, August 1972.

[5] Taravella, Brandon M., and Vorus, William S., A General Solution to Low Aspect Ratio Flat Ship Theory, Journal of Engineering Mathematics, Vol 71, Issue 2, October, 2011

[6] Ghadimi, Parviz, Saadatkhah, Amir and Dashtimanesh, Abbas, Analytical Solution of Wedge Water Entry by Using Schwartz-Christoffel Conformal Mapping, International Journal of Modeling, Simulation, and Scientific Computing, Vol 2, No. 3, 2011.

[7] Parviz Ghadimi, Mohammad A. Feizi Chekab, Abbas Dashtimanesh, A Numerical Investigation of the Water Impact of an Arbitrary Bow Section. ISH Journal of Hydraulic Engineering 19:3, 2013.

[8] Vorus, William S., Flat Cylinder Theory for Vessel Impact and Steady Planing Resistance, Journal of Ship Research, Vol 40, No. 2, June, 1996.

[9] Royce, Richard A., 2-D Impact Theory Extended to Planing Craft with Experimental Comparisons, Dissertation, University of New Orleans, 2001

[10] Aarsnes, J.V. "Drop test with ship sections – effect of roll angle, Report 603834.00.01." Norwegian Marine Technology Research Institute, Trondheim, Norway, 1996

[11] Taravella, Brandon M., A Hybrid Method for Predicting Lift and Drag of Semi-Planing / Semi-Displacement Hull forms, Dissertation, University of New Orleans, 2009

[12] Bajić D, Prpić-Oršić J and Turk A. Bow Flare Impact Loads on Containerships, The 19th Symposium on Theory and Practice of Shipbuilding, SORTA 2010

[13] Shanjie Zhang and Jianming Jin, Computation of Special Functions, John Wiley & Sons, Inc., 1996

[14] Michell, J., Wave Resistance of a Ship, Philosophical Magazine, 1897

Appendix A: Fortran code for processing of input data

```
! vsheet228.for
! Working code for 20-20,20-30,20-10
! Includes CW computation
! Convergence criterion is Vsi0
! Shorter Vsi0 comparison,  only recalculates Vsii, not entire dt
! Includes CP comp.
! Maximum number of variables used. Cannot use any more variables, reuse.
! VorC computation achieved.
! dvorC/dt achieved.
! Cp working & GOOD
! Working Cp distribution
! Clean code
! 15 Stations max
! Offset based hullform definition possible
! 30 points max for definition of hull contour
! Using ni=i for Cp computation
! Good Cp comparison with Vorus results
! Acceleration provision introduced into equation, only CONSTANT acc.
! Can take pitch motions in calculation
! Pitch rate provision introduced into equation, only CONSTANT pitch rate.
! Correction to dt-dtn (10% variation) time step to account for acceleration
! Provision for identifying premature separation added.
! Subroutines HYGFX(A,B,C,X,HF), GAMMA(X,GA) & PSI(X,PX) from [13]

      program vortexsheet
        real*8,dimension(450)::Vs,eta,zc,hc,S,R,zeta,
     &    Sij,zeta_c,Vs_c
        real*8::VorC(450,3),beta(101,2),Cp(100,450,15),Fim(450)
     &    ,xy(30,2,15),t(450,15),Cp_time(100,15),betac(2,15)
        real*8::lambda,kij,C,betat,eta_cavg,zeta_cavg,betatKe,Dd,
     & dt,dtn,Vsii,Rn,zbt,zct,kim,dzc,Ft,t_an,time,dv,V,St,betaKe11,
     & Qij,SHF,b,bn,Vj,delb,VorS,intg1,intg2,pit,LCF,prate,betaKep,
     & zccw,Stnspace
        integer::tm,Ke,p,Stn,Stnn,n_sel,np,npp,nppp,sepflag,nps(15)

        double precision::pi
        parameter (pi = 3.14159)

        tm=120
        Stnn=10
        Ke=90
        dv=0.0
        dzc=0.009
        zc0=0.10
```

```
        betaKe11=0.
        zccw=1.0
        sepflag=0

        open (21,file="Vel_data.dat")
        open (22,file="Cp_data.dat")
        open (23,file="Cp_rawdata.dat")
        open (11,file="inpf.txt")
        open (96,file="who.dat")
            read (11,*) Stnn
            read (11,*) Stnspace
            read (11,*) LCF
            read (11,*) n_sel
            do Stn=1,Stnn
               if (n_sel.eq.1) then
                  read (11,*) np
            !         write (6,*) np
                  do j=1,np
                     read (11,*) XY(j,1,Stn), XY(j,2,Stn)
                  enddo
                  nps(Stn)=np
               else
                  read (11,*) betac(1,Stn),betac(2,Stn)
               endif
             enddo

        ! Definition of eta steps
        do j=1,(Ke+1)
           eta(j)=REAL(j-1)/REAL(Ke)
        enddo
        d_eta=eta(2)-eta(1)
        write (6,*) '                    Impakt v1.3'
        write (6,*) '                    ==========='
        write (6,*) '                 Author: A. Benjamin Attumaly'
        write (6,*) ''
        write (6,200,advance='yes')
200     format('Welcome to impakt v1.3. The non-dimensionaled offset ')
        write (6,201,advance='yes')
201     format('data has been read from inpf.txt')
        write (6,*) ''
        write (6,*) Stnn,' Station(s) '
        write (6,*) ''
        write (6,202,advance='no')
202     format('Please input the acceleration (non-dimensionalized)')
        write (6,203,advance='no')
203     format(' of the body: ')
```

```
      read (5,*) dv

      write (6,302,advance='no')
302   format('Please input the pitch rate (non-dimensionalized)')
      write (6,303,advance='no')
303   format(' of the body (+ve Bow down): ')
      read (5,*) pit

      write (6,304,advance='no')
304   format('Please input the pitch rate acceleration ')
      write (6,305,advance='no')
305   format('(non-dimensionalized) of the body (+ve Bow down): ')
      read (5,*) prate


      write (6,*) ''

      write (6,204,advance='yes')
204   format('Please input the number of time steps to compute Cp')
      write (6,205,advance='no')
205   format(' data for ( >100 = CW flow ): ')
      read (5,*) tm

      write (6,*) ''

      write (23,*) Stnn
      write (23,*) tm
      write (23,*) Ke
      write (23,*) dzc
      write (23,*) zc0


      write (6,*) 'Processing input, please wait.'
      write (6,*) 'This may take a few minutes.'

      do Stn=1,Stnn
!        write (6,*) 'Station ',Stn
!        write (6,*) '===================='
!        write (6,*) 'Time Step No.    ','Vsi0       ',
!     &        '         Time'
!        write (6,*) '=======================================',
!     &        '========================='
         write (21,*) ' '
         write (21,*) 'Station ',Stn
         write (21,*) '===================='
         write (21,*) 'TimeStep         ','    Time       ',
```

```
&          '          Vsi0    ','              zc          ',
&                 '       zb      ','       Cp(keel)      '
      write (21,*) '=========================================',
&          '=================================================',
&          '=============================='

      b=1.0001
      dzc=0.009
      v0=1.
      Vj=0.0
      zc0=0.1
      ns=1
      ni=1
      zeta0=1.00
      delb=b-zeta0
      bn=b
      ci=0
      R0=0.
      zccw=1.0
      if(n_sel.eq.1) then
         zccw=XY(nps(Stn),1,Stn)
      endif
      sepflag=0
   ! beta matrix

      p=2
      do k=1,Ke+1
         if (n_sel.eq.1) then
            beta(k,1)=atan((XY(p,2,Stn)-XY(p-1,2,Stn))/
&             (XY(p,1,Stn)-XY(p-1,1,Stn)))
            if(XY(p,1,Stn)<eta(k+1)) then
               p=p+1
            endif
         else
            beta(k,1)=(betac(1,Stn)+eta(k)*
&             (betac(2,Stn)-betac(1,Stn)))*pi/180.
            !write (6,*) beta(k,1)
         endif
      enddo



   ! Initialization Routine
   np=1
   do k=1,Ke+1
      if(eta(k)*zc0>eta(np+1)) then
```

```
        np=np+1
      endif
      beta(k,2)=beta(np,1)+(eta(k)*zc0-eta(np))/
&       (eta(np+1)-eta(np))*(beta(np+1,1)-beta(np,1))
    enddo
    betatKe=atan(sin(beta(Ke,2)))
    lambda=0.5-betatKe/pi

    call GAMMA(lambda,Qij)
    call GAMMA(1.5-lambda,SHF)

    zbt=(pi**1.5)/(2*cos(betatKe)*tan(beta(Ke+1,2))*Qij*SHF)
    !write (6,*) zbt
    Vj=zbt+sqrt(zbt**2+1)

    call hygfx(lambda,lambda,lambda+1,(1-b**2),SHF)

    do while(((-2.0*Vj/pi)*((b**2-1)**lambda/(2*lambda))*
&        SHF+1)>0.001)
      b=b+.0001
      call hygfx(lambda,lambda,lambda+1,(1-b**2),SHF)
    enddo
    !write (6,*) b
    bn=b


    ! END OF INITIALIZATION ROUTINE
    ! vorticity for first time step
    zeta_c(1)=1.00
    zeta_c(2)=b

    do m=1,Ke
      eta_cavg=(eta(m+1)+eta(m))/2.
      kim=((1-eta_cavg**2)/eta_cavg**2)**(-betatKe/pi)
      C=1.0
      do k=1,Ke
        betat=atan(sin(beta(k,2)))
        C=C*abs((eta_cavg**2-(eta(k+1))**2)/
&         (eta_cavg**2-(eta(k))**2))**(betat/pi)
      enddo
      kim=kim*C

      VorC(m,1)=0.

      kij=1.0
      do j=1,ni
```

```
          VorS=-2.0*Vj
          betat=atan(sin(beta(m,2)))
          Qij=(eta(m)**2)*((zeta_c(j+1))**2-1)/((zeta_c(j+1))
&              **2-(eta(m)**2))
          call hygfx(lambda,lambda,lambda+1,Qij,SHF)
          Sij(1)=(cos(betat)/(pi*lambda))*(Qij**lambda)*SHF
          Qij=(eta(m)**2)*((zeta_c(j))**2-1)/((zeta_c(j))**2-
&              (eta(m)**2))
          call hygfx(lambda,lambda,lambda+1,Qij,SHF)
          Sij(2)=(cos(betat)/(pi*lambda))*(Qij**lambda)*SHF
          VorC(m,1)=VorC(m,1)+kim*VorS*(Sij(1)-Sij(2))/kij
       enddo
    enddo

    Vsi0=Vj
    Vs(1)=Vj
    dt=1.0  ! Initializations to enter the do-while loop inside i
    dtn=1.1 ! Initializations to enter the do-while loop inside i

    ! For time step 0, need to compute Ywl0, ie, R0
    zb=b*zc0
    np=1
    do k=1,Ke+1
       if(eta(k)*zb>eta(np+1)) then
          np=np+1
       endif
       beta(k,2)=beta(np,1)+(eta(k)*zb-eta(np))/
&        (eta(np+1)-eta(np))*(beta(np+1,1)-beta(np,1))
    enddo
    betatKe=atan(sin(beta(Ke+1,2)))
    lambda=0.5-betatKe/pi


    do k=1,(Ke+1)
       S(k)=tan(beta(k,2))

       if (k.eq.1) then
          hc(k)=0.0
       else
          hc(k)=hc(k-1)+zb*d_eta*S(k)
       endif
    enddo

    do j=1,Ke
       betat=atan(sin(beta(j,2)))
       ! Computation of kij
```

```
          eta_cavg=(eta(j+1)+eta(j))/2
          kij=((1-eta_cavg**2)/eta_cavg**2)**(-betatKe/pi)
          C=1.0
          do k=1,Ke
             betat=atan(sin(beta(k,2)))
             C=C*abs((eta_cavg**2-(eta(k+1))**2)/
  &                (eta_cavg**2-(eta(k))**2))**(betat/pi)
          enddo
          kij=kij*C

          ! *********** END OF COMPUTATION ****************
          R0=R0+2.0/pi*cos(betat)/kij*((hc(j)-
  &         S(j)*zb*eta(j))*
  &         (Py1(lambda,eta(j+1))-Py1(lambda,eta(j)))+S(j)*zb*
  &         (Py2(lambda,eta(j+1))-Py2(lambda,eta(j))))
       enddo

       Dd=(R0+XY(1,2,Stn))
       if (dv.eq.0) then
          t0=Dd/V0
       else
          t0=(-V0+sqrt(V0**2+2*dV*Dd))/dV
       endif
       ! Definition of initial time from velocity and R0.

       do i=1,tm ! Computations for time step i
          write(22,*) ' '
          write(22,*) '*****************************************'
          write(22,*) 'i= ',i, ', Station ',Stn
!          write(6,*) '*****************************************'
!          write(6,*) 'i= ',i, ', Station ',Stn
          Fim(i)=0.
          ! Updation of Zc
          zc(i)=.10+dzc*i
        if (zc(i).lt.zccw) then

       ! Updation of definition of b
          do j=1,1
             b=b+delb/10
          enddo

          ! Updation of Vs array
          do j=i,1,-1
             Vs(j+1)=Vs(j)
          enddo
          Vs(1)=Vsi0
```

```
        ! *******************

        ! dt=1, and dtn=0, set to enter do-while loop
        ci=0
        cn=0
        dt=1.
        dtn=0.

        do while (abs(dt-dtn)/dt>0.1)   ! exit if within 10%
           b=bn
           ! vsi0=1, and vsi0p=0, set to enter do-while loop
           vsi0=0.
           vsi0p=1.
           ci=0    ! Inner Loop Counter (vsi0 loop)

           ! Water line at time step i
           R(i)=0.

           ! COMPUTATION OF HC (I,J)
           zb=b*zc(i)
           np=1
           do k=1,Ke+1
              if(eta(k)*zb>eta(np+1)) then
                 np=np+1
              endif
              beta(k,2)=beta(np,1)+(eta(k)*zb-eta(np))/
   &            (eta(np+1)-eta(np))*(beta(np+1,1)-beta(np,1))
           enddo
           npp=np
           betaKe11=betatke
           betatKe=atan(sin(beta(Ke+1,2)))
!           write(6,*) 'betatke',betatKe
           lambda=0.5-betatKe/pi

           do k=1,(Ke+1)
              S(k)=tan(beta(k,2))

              if (k.eq.1) then
                 hc(k)=0.0
              else
                 hc(k)=hc(k-1)+zb*d_eta*S(k)
              endif
           enddo

           do j=1,Ke
              betat=atan(sin(beta(j,2)))
```

```
         ! Computation of kij
         eta_cavg=(eta(j+1)+eta(j))/2
         kij=((1-eta_cavg**2)/eta_cavg**2)**
&          (-betatKe/pi)
         C=1.0
         do k=1,Ke
            betat=atan(sin(beta(k,2)))
            C=C*abs((eta_cavg**2-(eta(k+1))**2)/
&          (eta_cavg**2-(eta(k))**2))**(betat/pi)
         enddo
         kij=kij*C
      ! *********** END OF COMPUTATION ****************

         R(i)=R(i)+2.0/pi*cos(betat)/kij*((hc(j)-
&       S(j)*zb*eta(j))*
&       (Py1(lambda,eta(j+1))-Py1(lambda,eta(j)))+S(j)*zb*
&       (Py2(lambda,eta(j+1))-Py2(lambda,eta(j))))
      enddo


      if(i.eq.1) then
         if (dv.eq.0) then
            V=V0
            dt=(R(1)-R0)/V0
            t(i,Stn)=t0+dt
         else
            V=V0+dv*t0+pit*(LCF-Stnspace*(Stn-1))+
&             prate*(LCF-Stnspace*(Stn-1))*t0
            dt=1/dV*(-V+sqrt(V**2+2*dv*(R(1)-R0)))
            t(i,Stn)=t0+dt
         endif
      else
         if (dv.eq.0) then
            V=V0
            if(R(i).gt.R(i-1)) then
               dt=(R(i)-R(i-1))/V0
            else    ! Premature flow separation
               dt=t(i-1,Stn)-t(i-2,Stn)
               zccw=zc(i) ! flow separation point
               sepflag=sepflag+1
            endif

            if(dt.le.((1.0001*zc(i)-zc(i-1))/Vs(1)))then
               dt=t(i-1,Stn)-t(i-2,Stn)
               zccw=zc(i) ! flow separation point
               sepflag=1
```

```
          endif

          t(i,Stn)=t(i-1,Stn)+dt
        else
         V=V0+dv*t(i-1,Stn)+pit*(LCF-Stnspace*(Stn-1))
     &     +prate*(LCF-Stnspace*(Stn-1))*t(i-1,Stn)
          if(R(i).gt.R(i-1)) then
           dt=1/dV*(-V+sqrt(V**2+2*dv*(R(i)-R(i-1))))
          else        ! Premature flow separation
            dt=t(i-1,Stn)-t(i-2,Stn)
            zccw=zc(i)     ! flow separation point
            sepflag=sepflag+1
          endif

          if(dt.le.((1.0001*zc(i)-zc(i-1))/Vs(1)))then
            zccw=zc(i) ! flow separation point
            sepflag=1
            dt=t(i-1,Stn)-t(i-2,Stn)
          endif
          t(i,Stn)=t(i-1,Stn)+dt
        endif
      endif

        ! Updation of zeta matrix
      zeta0=1.
      zeta(i+1)=(Vj*(t(i,Stn)-t0)+zc0*b)/zc(i)
      zeta(i)=(Vj*(t(i,Stn)-t0)+zc0)/zc(i)
      do j=1,i-1
       zeta(j)=(vs(j)*(t(i,Stn)-t(i-j,Stn))+zc(i-j))/zc(i)
      enddo
      ! Computation of ns
      ns=0
      do j=1,i+1
        if (b>zeta(j)) then
          ns=ns+1
        else
          ns=j
          exit
        endif
      enddo
!      write(6,*) 'ns ',ns

      do while ((abs(vsi0-vsi0p)>0.005))
        vsi0p=vsi0

      ! Computation of Vsii
```

```
            if ((ns.eq.1).and.(ci.eq.0)) then
               Vsii=Vs(1)
            else if ((ns.eq.1).and.(ci.ne.0)) then
               Vsii=Vsi0+(b-zeta0)/(zeta(1)-zeta0)*
&             (Vs(1)-Vsi0)
            else
               Vsii=Vs(ns-1)+(b-zeta(ns-1))/(zeta(ns)
&             -zeta(ns-1))*
&             (Vs(ns)-Vs(ns-1))
            endif

        ! Velocity matrix for computation
            do j=1,ns-1
               Vs_c(j)=Vs(j)
            enddo
            Vs_c(ns)=Vsii



          ! Zeta matrix for computation
            do j=1,ns-1
               zeta_c(j)=zeta(j)
            enddo
            zeta_c(ns)=b

          ! Computation of zbt
           if (zc(i).lt.1.0) then
               zbt=(Vsii**2-V**2)/(2*Vsii)
            else
               zbt=Vsii/2
            endif

          ! Computation of zct
            if (i.eq.1) then
               zct=(zc(1)-zc0)/dt
            else
               zct=(zc(i)-zc(i-1))/dt

            endif

          ! Computation of delbij
            do j=1,1
               delb=((zbt-zct)*dt-j+1)/zc(i)
            enddo

          ! Computation of Vsi0
```

```
                St=V
                ! Redefine beta_Ke in terms of zc.
                np=1
                do k=1,Ke+1
                   if(eta(k)*zc(i)>eta(np+1)) then
                      np=np+1
                   endif
                   beta(k,2)=beta(np,1)+(eta(k)*zc(i)-eta(np))/
     &              (eta(np+1)-eta(np))*(beta(np+1,1)-beta(np,1))
                enddo
                betatKe=atan(sin(beta(Ke+1,2)))
                lambda=0.5-betatKe/pi
                !if (i.eq.14) then
                !    write (6,*) np
                !endif

                do j=ns,2,-1
                  ! Computation of kij
                   zeta_cavg=(zeta_c(j)+zeta_c(j-1))/2
                   kij=((zeta_cavg**2-1)/zeta_cavg**2)**
     &               (-betatKe/pi)
                   C=1.0
                   do k=1,Ke
                      betat=atan(sin(beta(k,2)))
                      C=C*((zeta_cavg**2-(eta(k+1))**2)/
     &                  (zeta_cavg**2-(eta(k))**2))**(betat/pi)
                   enddo
                   kij=kij*C
                 ! End of kij computation
                  if (stn.eq.4) then
!                    write (6,*) zeta_c(j)
                  endif
                  St=St-(1/(2.*pi*lambda))*(Vs_c(j)+Vs_c(j-1))/
     &              kij*(Tij(zeta_c(j),lambda)-Tij(zeta_c(j-1),
     &              lambda))
                enddo
                ! kij for j=1
                zeta_cavg=(zeta0+zeta_c(1))/2
                kij=((zeta_cavg**2-1)/zeta_cavg**2)**
     &           (-betatKe/pi)
                C=1.0
                do k=1,Ke
                   betat=atan(sin(beta(k,2)))
                   C=C*((zeta_cavg**2-(eta(k+1))**2)/
     &               (zeta_cavg**2-(eta(k))**2))**(betat/pi)
                enddo
```

```
            kij=kij*C
            ! kij for j=1 computation complete
            Vsi0=St*2.0*pi*lambda*kij/
     &        Tij(zeta_c(j),lambda)-Vs_c(1)
            if (Stn.eq.4) then
            !write(6,*) 'zeta_c(j)',zeta_c(j)
            endif
            ci=ci+1

!             write (6,*) ci
         enddo
         ! End of vsi0 Loop

         bn=b
         do j=1,1
            bn=bn+delb/10
         enddo

         zb=zc(i)*bn
         np=1
         do k=1,Ke+1
            if(eta(k)*zb>eta(np+1)) then
               np=np+1
            endif
            beta(k,2)=beta(np,1)+(eta(k)*zb-eta(np))/
     &        (eta(np+1)-eta(np))*(beta(np+1,1)-beta(np,1))
         enddo
         betatKe=atan(sin(beta(Ke+1,2)))
         lambda=0.5-betatKe/pi

         Rn=0.

         do k=1,(Ke+1)
            S(k)=tan(beta(k,2))
            if (k.eq.1) then
               hc(k)=0.0
            else
               hc(k)=hc(k-1)+zb*d_eta*S(k)
            endif
         enddo

         do j=1,Ke
            betat=atan(sin(beta(j,2)))
            ! Computation of kij
            eta_cavg=(eta(j+1)+eta(j))/2
            kij=((1-eta_cavg**2)/eta_cavg**2)**(-betatKe/pi)
```

```
          C=1.0
          do k=1,Ke
             betat=atan(sin(beta(k,2)))
             C=C*abs((eta_cavg**2-(eta(k+1))**2)/
&               (eta_cavg**2-(eta(k))**2))**(betat/pi)
          enddo
          kij=kij*C
          Rn=Rn+2.0/pi*cos(betat)/kij*((hc(j)-
&           S(j)*zb*eta(j))*
&         (Py1(lambda,eta(j+1))-Py1(lambda,eta(j)))+S(j)*zb*
&           (Py2(lambda,eta(j+1))-Py2(lambda,eta(j))))
       enddo

       if(i.eq.1) then
          if (dv.eq.0) then
             V=V0
             if (Rn.gt.R(i-1))then
                dtn=(Rn-R0)/V0
             else
                dtn=dt
             endif
          else
             V=V0+dv*t0
             if (Rn.gt.R(i-1))then
                dtn=1/dV*(-V+sqrt(V**2+2*dv*(Rn-R0)))
             else
                dtn=dt
             endif
          endif
       else
          if (dv.eq.0) then
             V=V0
             if (Rn.gt.R(i-1))then
                dtn=(Rn-R(i-1))/V0
             else
                dtn=dt
             endif
            if(dtn.le.((1.0001*zc(i)-zc(i-1))/Vs(1)))then
                dtn=dt
                zccw=zc(i) ! flow separation point
                sepflag=1
             endif
          else
             V=V0+dv*t(i-1,Stn)
             if (Rn.gt.R(i-1))then
               dtn=1/dV*(-V+sqrt(V**2+2*dv*(Rn-R(i-1))))
```

```
                else
                    dtn=dt
                endif
               if(dtn.le.((1.0001*zc(i)-zc(i-1))/Vs(1)))then
                    dtn=dt
                    zccw=zc(i) ! flow separation point
                    sepflag=1
                endif
            endif
        endif
        cn=cn+1
    enddo ! End of dt-dtn verification loop
    betaKep=betaKe11
    nppp=npp
!             write (6,*) 'delb/10',delb/10
!             write (6,*) 'bn',bn
   else   ! CW flow
    sepflag=sepflag+1
    zc(i)=zccw

    ! Updation of definition of b
    do j=1,1
        b=b+delb/10
    enddo

    ! Updation of zb
    zb=b*zc(i)

    ! Updation of Vs array
    do j=i,1,-1
        Vs(j+1)=Vs(j)
    enddo
    Vs(1)=Vsi0
    ! *******************

    vsi0=0.  ! Reset for inner loop
    vsi0p=1.  ! Reset for inner loop

    ! Time step obtained from previous computations in CW flow
    dt=t(i-1,Stn)-t(i-2,Stn)
    t(i,Stn)=t(i-1,Stn)+dt
    write (96,*) ' cw dt',dt
    ! Updation of zeta matrix
    zeta0=1.
    zeta(i+1)=(Vj*(t(i,Stn)-t0)+zc0*b)/zc(i)
    zeta(i)=(Vj*(t(i,Stn)-t0)+zc0)/zc(i)
```

```
        do j=1,i-1
           zeta(j)=(vs(j)*(t(i,Stn)-t(i-j,Stn))+zc(i-j))/zc(i)
        enddo

        ! Computation of ns
        ns=0
        do j=1,i+1
           if (b>zeta(j)) then
              ns=ns+1
           else
              ns=j
              exit
           endif
        enddo

        do while ((abs(vsi0-vsi0p)>0.005))
           vsi0p=vsi0

           ! Computation of Vsii
           if ((ns.eq.1).and.(ci.eq.0)) then
              Vsii=Vs(1)
           else if ((ns.eq.1).and.(ci.ne.0)) then
              Vsii=Vsi0+(b-zeta0)/(zeta(1)-zeta0)*
&            (Vs(1)-Vsi0)
           else
              Vsii=Vs(ns-1)+(b-zeta(ns-1))/(zeta(ns)
&            -zeta(ns-1))*
&            (Vs(ns)-Vs(ns-1))
           endif

        ! Velocity matrix for computation
           do j=1,ns-1
              Vs_c(j)=Vs(j)
           enddo
           Vs_c(ns)=Vsii


           ! Zeta matrix for computation
           do j=1,ns-1
              zeta_c(j)=zeta(j)
           enddo
           zeta_c(ns)=b

           ! Computation of zbt
           if(zc(i).lt.1.0) then
              zbt=(Vsii**2-V**2)/(2*Vsii)
```

```
          else
             zbt=Vsii/2
          endif

          ! Computation of zct
          if (i.eq.1) then
             zct=(zc(1)-zc0)/dt
          else
             zct=(zc(i)-zc(i-1))/dt
          endif

          ! Computation of delbij
          do j=1,1
             delb=((zbt-zct)*dt-j+1)/zc(i)
          enddo

          ! Computation of Vsi0
          St=V
          ! Redefine beta_Ke in terms of zc.
          np=1
          do k=1,Ke+1
             if(eta(k)*zc(i)>eta(np+1)) then
                np=np+1
             endif
             beta(k,2)=beta(np,1)+(eta(k)*zc(i)-eta(np))/
&              (eta(np+1)-eta(np))*(beta(np+1,1)-beta(np,1))
          enddo
          betatKe=atan(sin(beta(Ke+1,2)))
          lambda=0.5-betatKe/pi

          do j=ns,2,-1
          ! Computation of kij
             zeta_cavg=(zeta_c(j)+zeta_c(j-1))/2
             kij=((zeta_cavg**2-1)/zeta_cavg**2)**
&              (-betatKe/pi)
             C=1.0
             do k=1,Ke
                betat=atan(sin(beta(k,2)))
                C=C*((zeta_cavg**2-(eta(k+1))**2)/
&              (zeta_cavg**2-(eta(k))**2))**(betat/pi)
             enddo
             kij=kij*C
             ! End of kij computation
             St=St-(1/(2.*pi*lambda))*(Vs_c(j)+Vs_c(j-1))/
&              kij*(Tij(zeta_c(j),lambda)-Tij(zeta_c(j-1),
&              lambda))
```

```
              enddo
              ! kij for j=1
              zeta_cavg=(zeta0+zeta_c(1))/2
              kij=((zeta_cavg**2-1)/zeta_cavg**2)**
     &          (-betatKe/pi)
              C=1.0
              do k=1,Ke
                 betat=atan(sin(beta(k,2)))
                 C=C*((zeta_cavg**2-(eta(k+1))**2)/
     &             (zeta_cavg**2-(eta(k))**2))**(betat/pi)
              enddo
              kij=kij*C
              ! kij for j=1 computation complete
              Vsi0=St*2.0*pi*lambda*kij/
     &             Tij(zeta_c(j),lambda)-Vs_c(1)
              ci=ci+1
           enddo
          endif   ! End of chine wetted / unwetted flow
          write (23,*) t(i,stn)
!          write (6,*) i,vsi0,t(i,Stn)

          write (22,*) 'Time =',t(i,Stn)
!          write (6,*) 'Time =',t(i,Stn)
                 ! *********** COMPUTATION OF CP ***********************
           ! Velocity matrix for computation of Cp for present timestep

           ni=i

           do j=ni,1,-1
              Vs_c(j+1)=Vs_c(j)
           enddo
           Vs_c(1)=Vsi0

           ! Zeta matrix for computation of CP
           do j=ni,1,-1
              zeta_c(j+1)=zeta_c(j)
           enddo
           zeta_c(1)=zeta0

           ! *******************************

           !m= Element no. of target element
           np=1
           do k=1,Ke+1
              if(eta(k)*zc(i)>eta(np+1)) then
                 np=np+1
```

```
          endif
          beta(k,2)=beta(np,1)+(eta(k)*zc(i)-eta(np))/
     &      (eta(np+1)-eta(np))*(beta(np+1,1)-beta(np,1))
        enddo
        betatKe=atan(sin(beta(Ke+1,2)))
        lambda=0.5-betatKe/pi

        ! Defining VorC(*,2) as the value of the element in the
        ! previous timestep

        do m=1,Ke
          Vorc(m,2)=VorC(m,1)
          VorC(m,1)=0.
          betat=atan(sin(beta(m,2)))
          eta_cavg=(eta(m+1)+eta(m))/2
          kim=((1-eta_cavg**2)/eta_cavg**2)**(-betatKe/pi)
          C=1.0
          do k=1,Ke
            betat=atan(sin(beta(k,2)))
            C=C*abs((eta_cavg**2-(eta(k+1))**2)/
     &        (eta_cavg**2-(eta(k))**2))**(betat/pi)
          enddo
          kim=kim*C

          do j=1,ni
            VorS=-(Vs_c(j)+Vs_c(j+1))
            if (m.eq.1) then
              !write (6,*) 'VorS i',j,VorS
            endif
            betat=atan(sin(beta(m,2)))
            !lambda=0.5-betat/pi
            Qij=(eta_cavg**2)*((zeta_c(j+1))**2-1)/
     &        ((zeta_c(j+1))**2-(eta_cavg**2))
            call hygfx(lambda,lambda,lambda+1,Qij,SHF)
            Sij(1)=(cos(betat)/(pi*lambda))*(Qij**lambda)
     &        *SHF
            Qij=(eta_cavg**2)*((zeta_c(j))**2-1)/
     &        ((zeta_c(j))**2-(eta_cavg**2))
            call hygfx(lambda,lambda,lambda+1,Qij,SHF)
            Sij(2)=(cos(betat)/(pi*lambda))*(Qij**lambda)
     &        *SHF

          ! Computation of kij
            zeta_cavg=(zeta_c(j+1)+zeta_c(j))/2
            kij=((zeta_cavg**2-1)/zeta_cavg**2)**
     &        (-betatKe/pi)
```

```
            C=1.0
            do k=1,Ke
               betat=atan(sin(beta(k,2)))
               C=C*((zeta_cavg**2-(eta(k+1))**2)/
     &          (zeta_cavg**2-(eta(k))**2))**(betat/pi)
            enddo
            kij=kij*C
            VorC(m,1)=VorC(m,1)+kim*VorS*(Sij(1)-Sij(2))/kij
         enddo
         vorc(m,3)=(vorc(m,1)-vorc(m,2))/dt
      enddo
      !write (6,*) 'i =',i
      !write (6,*) '================'
!      write (6,*) '          m      ','      Cp(m)         '
!      write (6,*) '=========================================='
      write (22,*) '          m      ','      Cp(m)         '
      write (22,*) '=========================================='

      do m=1,Ke
      ! Integration of VorC from eta(m) to 1.0
         p=Ke-m
         if(m.le.(Ke-2)) then
            if((p/2)*2.eq.p) then ! Even number of segments
               Intg1=0.
               Intg2=0.
               do k=m+2,Ke-2,2
                 Intg1=Intg1+2.0*vorc(k,1)+4.0*vorc(k+1,1)
                 Intg2=Intg2+2.0*vorc(k,3)+4.0*
     &             vorc(k+1,3)
               enddo
               Intg1=(Intg1+vorc(m,1)+vorc(Ke,1)+4.0*
     &          vorc(m+1,1))*d_eta/3.0
               Intg2=(Intg2+vorc(m,3)+vorc(Ke,3)+4.0*
     &          vorc(m+1,3))*d_eta/3.0
            else       ! Odd number of segments
               Intg1=0.
               Intg2=0.
               do k=m+3,Ke-2,2
                 Intg1=Intg1+2.0*vorc(k,1)+4.0*vorc(k+1,1)
                  Intg2=Intg2+2.0*vorc(k,3)+4.0*
     &             vorc(k+1,3)
               enddo
               Intg1=(Intg1+vorc(m+1,1)+vorc(Ke,1)+4.0*
     &          vorc(m+2,1))
     &          *d_eta/3.0+(vorc(m+1,1)+vorc(m,1))/2.*d_eta
               Intg2=(Intg2+vorc(m+1,3)+vorc(Ke,3)+4.0*
```

```
     &                 vorc(m+2,3))*d_eta/3.0+(vorc(m,3)+vorc(m+1,3))
     &                  /2.*d_eta
                      endif
                   elseif (m.eq.(Ke-1)) then
                      Intg1=(vorc(m+1,1)+vorc(m,1))/2.*d_eta
                      Intg2=(vorc(m+1,3)+vorc(m,3))/2.*d_eta
                   else
                      Intg1=0.
                      Intg2=0.
                   endif
                   eta_cavg=(eta(m)+eta(m+1))/2.
                  Cp(m,i,Stn)=0.25*((VorC(Ke,1))**2-(VorC(m,1))**2)-zct*
     &             (Intg1+eta_cavg*VorC(m,1)-VorC(Ke,1))-zc(i)*Intg2
!                  write (6,*) m,Cp(m,i,Stn)
                  write (23,*) Cp(m,i,Stn)
                  write (22,*) m,Cp(m,i,Stn)
                enddo
                if (sepflag.ne.1) then
                   write (21,*) i,t(i,Stn),vsi0,zc(i),zb,Cp(1,i,Stn)
                else
                   write (21,*) i,t(i,Stn),vsi0,zc(i),zb,Cp(1,i,Stn),
     &       'flow separation'
                endif
             enddo ! End of i loop
             write (23,*) zccw
             write (6,*) 'Station ',Stn,'               100% Complete'
             enddo ! End of out Stn loop
             ! Find the minimum time, ie, time that can be analysed
             t_an=t(tm,1)
             do Stn=2,Stnn
                if (t_an>t(tm,Stn)) then
                   t_an=t(tm,Stn)
                endif
             enddo
             write (6,*) ''
             write(6,*) 'Analysis data written to cp_rawdata.dat'

             time=0.100
             do Stn=1,Stnn
!                 write (6,*) 'Stn ',Stn
!                 write (22,*) 'Stn ',Stn
             ! Cp distribution output
             ! Computation of np
                   np=0
                   do j=1,tm
                      if (time>t(j,Stn)) then
```

```
               np=np+1
             else
               np=j
               exit
             endif
           enddo

           do k=1,Ke
             Cp_time(k,Stn)=Cp(k,np,Stn)+
     &         (Cp(k,np+1,Stn)-Cp(k,np,Stn))/(t(np+1,Stn)-
     &          t(np,Stn))*(time-t(np,Stn))
!            write (6,*) 'Cp at time',Cp_time(k,Stn)
!            write (22,*) 'Cp at time',Cp_time(k,Stn)
           enddo
       enddo

       do j=i,1,-1
         Vs(j+1)=Vs(j)
       enddo
       Vs(1)=Vsi0

       endfile(23)
       close(21)
       close(22)
       close(23)
       close(11)
       close(96)


      end program


      SUBROUTINE HYGFX(A,B,C,X,HF)
C
C       ======================================================
C       Purpose: Compute hypergeometric function F(a,b,c,x)
C       Input :  a --- Parameter
C                b --- Parameter
C                c --- Parameter, c <> 0,-1,-2,...
C                x --- Argument   ( x < 1 )
C       Output:  HF --- F(a,b,c,x)
C       Routines called:
C            (1) GAMMA for computing gamma function
C            (2) PSI for computing psi function
C       ======================================================
C
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      LOGICAL L0,L1,L2,L3,L4,L5
      PI=3.141592653589793D0
      EL=.5772156649015329D0
      L0=C.EQ.INT(C).AND.C.LT.0.0
      L1=1.0D0-X.LT.1.0D-15.AND.C-A-B.LE.0.0
      L2=A.EQ.INT(A).AND.A.LT.0.0
      L3=B.EQ.INT(B).AND.B.LT.0.0
      L4=C-A.EQ.INT(C-A).AND.C-A.LE.0.0
      L5=C-B.EQ.INT(C-B).AND.C-B.LE.0.0
      IF (L0.OR.L1) THEN
        WRITE(*,*)'The hypergeometric series is divergent'
        RETURN
      ENDIF
      EPS=1.0D-15
      IF (X.GT.0.95) EPS=1.0D-8
      IF (X.EQ.0.0.OR.A.EQ.0.0.OR.B.EQ.0.0) THEN
        HF=1.0D0
        RETURN
      ELSE IF (1.0D0-X.EQ.EPS.AND.C-A-B.GT.0.0) THEN
        CALL GAMMA(C,GC)
        CALL GAMMA(C-A-B,GCAB)
        CALL GAMMA(C-A,GCA)
        CALL GAMMA(C-B,GCB)
        HF=GC*GCAB/(GCA*GCB)
        RETURN
      ELSE IF (1.0D0+X.LE.EPS.AND.DABS(C-A+B-1.0).LE.EPS) THEN
        G0=DSQRT(PI)*2.0D0**(-A)
        CALL GAMMA(C,G1)
        CALL GAMMA(1.0D0+A/2.0-B,G2)
        CALL GAMMA(0.5D0+0.5*A,G3)
        HF=G0*G1/(G2*G3)
        RETURN
      ELSE IF (L2.OR.L3) THEN
        IF (L2) NM=INT(ABS(A))
        IF (L3) NM=INT(ABS(B))
        HF=1.0D0
        R=1.0D0
        DO 10 K=1,NM
          R=R*(A+K-1.0D0)*(B+K-1.0D0)/(K*(C+K-1.0D0))*X
10         HF=HF+R
        RETURN
      ELSE IF (L4.OR.L5) THEN
        IF (L4) NM=INT(ABS(C-A))
        IF (L5) NM=INT(ABS(C-B))
        HF=1.0D0
```

```
        R=1.0D0
        DO 15 K=1,NM
          R=R*(C-A+K-1.0D0)*(C-B+K-1.0D0)/(K*(C+K-1.0D0))*X
15        HF=HF+R
        HF=(1.0D0-X)**(C-A-B)*HF
        RETURN
      ENDIF
      AA=A
      BB=B
      X1=X
      IF (X.LT.0.0D0) THEN
        X=X/(X-1.0D0)
        IF (C.GT.A.AND.B.LT.A.AND.B.GT.0.0) THEN
          A=BB
          B=AA
        ENDIF
        B=C-B
      ENDIF
      IF (X.GE.0.75D0) THEN
        GM=0.0D0
        IF (DABS(C-A-B-INT(C-A-B)).LT.1.0D-15) THEN
          M=INT(C-A-B)
          CALL GAMMA(A,GA)
          CALL GAMMA(B,GB)
          CALL GAMMA(C,GC)
          CALL GAMMA(A+M,GAM)
          CALL GAMMA(B+M,GBM)
          CALL PSI(A,PA)
          CALL PSI(B,PB)
          IF (M.NE.0) GM=1.0D0
          DO 30 J=1,ABS(M)-1
30          GM=GM*J
          RM=1.0D0
          DO 35 J=1,ABS(M)
35          RM=RM*J
          F0=1.0D0
          R0=1.0D0
          R1=1.0D0
          SP0=0.D0
          SP=0.0D0
          IF (M.GE.0) THEN
            C0=GM*GC/(GAM*GBM)
            C1=-GC*(X-1.0D0)**M/(GA*GB*RM)
            DO 40 K=1,M-1
              R0=R0*(A+K-1.0D0)*(B+K-1.0)/(K*(K-M))*(1.0-X)
40            F0=F0+R0
```

```
            DO 45 K=1,M
45             SP0=SP0+1.0D0/(A+K-1.0)+1.0/(B+K-1.0)-1.0/K
            F1=PA+PB+SP0+2.0D0*EL+DLOG(1.0D0-X)
            DO 55 K=1,250
              SP=SP+(1.0D0-A)/(K*(A+K-1.0))+(1.0-B)/(K*(B+K-1.0))
              SM=0.0D0
              DO 50 J=1,M
50               SM=SM+(1.0D0-A)/((J+K)*(A+J+K-1.0))+1.0/
   &               (B+J+K-1.0)
              RP=PA+PB+2.0D0*EL+SP+SM+DLOG(1.0D0-X)
              R1=R1*(A+M+K-1.0D0)*(B+M+K-1.0)/(K*(M+K))*(1.0-X)
              F1=F1+R1*RP
              IF (DABS(F1-HW).LT.DABS(F1)*EPS) GO TO 60
55             HW=F1
60           HF=F0*C0+F1*C1
          ELSE IF (M.LT.0) THEN
            M=-M
            C0=GM*GC/(GA*GB*(1.0D0-X)**M)
            C1=-(-1)**M*GC/(GAM*GBM*RM)
            DO 65 K=1,M-1
              R0=R0*(A-M+K-1.0D0)*(B-M+K-1.0)/(K*(K-M))*(1.0-X)
65             F0=F0+R0
            DO 70 K=1,M
70             SP0=SP0+1.0D0/K
            F1=PA+PB-SP0+2.0D0*EL+DLOG(1.0D0-X)
            DO 80 K=1,250
              SP=SP+(1.0D0-A)/(K*(A+K-1.0))+(1.0-B)/(K*(B+K-1.0))
              SM=0.0D0
              DO 75 J=1,M
75               SM=SM+1.0D0/(J+K)
              RP=PA+PB+2.0D0*EL+SP-SM+DLOG(1.0D0-X)
              R1=R1*(A+K-1.0D0)*(B+K-1.0)/(K*(M+K))*(1.0-X)
              F1=F1+R1*RP
              IF (DABS(F1-HW).LT.DABS(F1)*EPS) GO TO 85
80             HW=F1
85           HF=F0*C0+F1*C1
          ENDIF
        ELSE
          CALL GAMMA(A,GA)
          CALL GAMMA(B,GB)
          CALL GAMMA(C,GC)
          CALL GAMMA(C-A,GCA)
          CALL GAMMA(C-B,GCB)
          CALL GAMMA(C-A-B,GCAB)
          CALL GAMMA(A+B-C,GABC)
          C0=GC*GCAB/(GCA*GCB)
```

```
          C1=GC*GABC/(GA*GB)*(1.0D0-X)**(C-A-B)
          HF=0.0D0
          R0=C0
          R1=C1
          DO 90 K=1,250
            R0=R0*(A+K-1.0D0)*(B+K-1.0)/(K*(A+B-C+K))*(1.0-X)
            R1=R1*(C-A+K-1.0D0)*(C-B+K-1.0)/(K*(C-A-B+K))
    &           *(1.0-X)
            HF=HF+R0+R1
            IF (DABS(HF-HW).LT.DABS(HF)*EPS) GO TO 95
90           HW=HF
95          HF=HF+C0+C1
        ENDIF
      ELSE
        A0=1.0D0
        IF (C.GT.A.AND.C.LT.2.0D0*A.AND.
    &       C.GT.B.AND.C.LT.2.0D0*B) THEN
          A0=(1.0D0-X)**(C-A-B)
          A=C-A
          B=C-B
        ENDIF
        HF=1.0D0
        R=1.0D0
        DO 100 K=1,250
          R=R*(A+K-1.0D0)*(B+K-1.0D0)/(K*(C+K-1.0D0))*X
          HF=HF+R
          IF (DABS(HF-HW).LE.DABS(HF)*EPS) GO TO 105
100         HW=HF
105       HF=A0*HF
      ENDIF
      IF (X1.LT.0.0D0) THEN
        X=X1
        C0=1.0D0/(1.0D0-X)**AA
        HF=C0*HF
      ENDIF
      A=AA
      B=BB
!       IF (K.GT.120) WRITE(*,115)
!115    FORMAT(1X,'Warning! You should check the accuracy')
      RETURN
      END


      SUBROUTINE GAMMA(X,GA)
C
C       ==================================================
```

```
C       Purpose: Compute gamma function â(x)
C       Input :  x  --- Argument of â(x)
C                    ( x is not equal to 0,-1,-2,úúú)
C       Output:  GA --- â(x)
C       ===================================================
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        DIMENSION G(26)
        PI=3.141592653589793D0
        IF (X.EQ.INT(X)) THEN
          IF (X.GT.0.0D0) THEN
            GA=1.0D0
            M1=X-1
            DO 10 K=2,M1
10             GA=GA*K
          ELSE
            GA=1.0D+300
          ENDIF
        ELSE
          IF (DABS(X).GT.1.0D0) THEN
            Z=DABS(X)
            M=INT(Z)
            R=1.0D0
            DO 15 K=1,M
15             R=R*(Z-K)
            Z=Z-M
          ELSE
            Z=X
          ENDIF
          DATA G/1.0D0,0.5772156649015329D0,
     &          -0.6558780715202538D0, -0.420026350340952D-1,
     &          0.1665386113822915D0,-.421977345555443D-1,
     &          -.9621971527877OD-2, .7218943246663OD-2,
     &          -.11651675918591D-2, -.2152416741149D-3,
     &          .1280502823882D-3, -.201348547807D-4,
     &          -.12504934821D-5, .11330272320D-5,
     &          -.2056338417D-6, .61160950D-8,
     &          .50020075D-8, -.11812746D-8,
     &          .1043427D-9, .77823D-11,
     &          -.36968D-11, .51D-12,
     &          -.206D-13, -.54D-14, .14D-14, .1D-15/
          GR=G(26)
          DO 20 K=25,1,-1
20           GR=GR*Z+G(K)
          GA=1.0D0/(GR*Z)
          IF (DABS(X).GT.1.0D0) THEN
```

```
            GA=GA*R
            IF (X.LT.0.0D0) GA=-PI/(X*GA*DSIN(PI*X))
          ENDIF
        ENDIF
        RETURN
        END


        SUBROUTINE PSI(X,PS)
C
C       ==========================================
C       Purpose: Compute Psi function
C       Input :  x  --- Argument of psi(x)
C       Output:  PS --- psi(x)
C       ==========================================
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        XA=DABS(X)
        PI=3.141592653589793D0
        EL=.5772156649015329D0
        S=0.0D0
        IF (X.EQ.INT(X).AND.X.LE.0.0) THEN
          PS=1.0D+300
          RETURN
        ELSE IF (XA.EQ.INT(XA)) THEN
          N=XA
          DO 10 K=1 ,N-1
10           S=S+1.0D0/K
          PS=-EL+S
        ELSE IF (XA+.5.EQ.INT(XA+.5)) THEN
          N=XA-.5
          DO 20 K=1,N
20           S=S+1.0/(2.0D0*K-1.0D0)
          PS=-EL+2.0D0*S-1.386294361119891D0
        ELSE
          IF (XA.LT.10.0) THEN
            N=10-INT(XA)
            DO 30 K=0,N-1
30             S=S+1.0D0/(XA+K)
            XA=XA+N
          ENDIF
          X2=1.0D0/(XA*XA)
          A1=-.8333333333333D-01
          A2=.83333333333333333D-02
          A3=-.39682539682539683D-02
          A4=.41666666666666667D-02
```

```
    A5=-.7575757575757575758D-02
    A6=.21092796092796093D-01
    A7=-.83333333333333333D-01
    A8=.4432598039215686D0
    PS=DLOG(XA)-.5D0/XA+X2*(((((((A8*X2+A7)*X2+
&       A6)*X2+A5)*X2+A4)*X2+A3)*X2+A2)*X2+A1)
    PS=PS-S
  ENDIF
  IF (X.LT.0.0) PS=PS-PI*DCOS(PI*X)/DSIN(PI*X)-1.0D0/X
  RETURN
  END

  real function Tij(zeeta1,lambda)
     real*8:: lambda,zeeta1,HA
     call hygfx(lambda,lambda,lambda+1.,1-(zeeta1**2),HA)
     Tij=(((zeeta1**2)-1.0)**(lambda))*HA
  return
  end

  real function Py1(lambda1, eta1)
     real*8::lambda1,eta1,HA
     call hygfx(1.0-lambda1,1.0-lambda1,2.0-
&     lambda1,eta1**2,HA)
     Py1=eta1**(2*(1-lambda1))/(2*(1-lambda1))*HA
  return
  end

  real function Py2(lambda1,eta1)
     real*8::lambda1,eta1,HA
     call hygfx(1-lambda1,1.5-lambda1,2.5-lambda1,eta1**2,HA)
     Py2=eta1**(2*(1.5-lambda1))/(2*(1.5-lambda1))*HA
  return
  end
```

Appendix B: Post processing code for data output by code vsheet228.for

```
! Post processing program

      program pprocess
        real*8::t_max,t_min,t(450,15),time,Cp(100,450,15),
   &      Cp_time(100,15),dzc,zci(15),zc0,dx,Fim(15),zccw(15)
        character(len=1024) :: filename,ffname,dat
        character(len=1024) :: format_string
        integer :: ci,np,tm,Stnn,Ke,kim,Stn

        open (11,file="cp_rawdata.dat",action='read',status='old')
        read (11,*) Stnn
        !write (6,*) Stnn
        read (11,*) tm
        !write (6,*) tm
        read (11,*) Ke
        !write (6,*) Ke
        read (11,*) dzc
        !write (6,*) dzc
        read (11,*) zc0
        !write (6,*) zc0

        do Stn=1,Stnn
           do i=1,tm
              read(11,*) t(i,Stn)
              do m=1,Ke
                 read(11,*) Cp(m,i,Stn)
                 !write (6,*) Cp(m,i,Stn)
              enddo
           enddo
           read(11,*) zccw(Stn)
        enddo
        close(11)
        write (6,*) '                Impakt v1.0 Postprocessor'
        write (6,*) '                ========================='
        write (6,*) '               Author: A. Benjamin Attumaly'
        write (6,*) ''
        write (6,200,advance='yes')
200     format('Welcome to impakt v1.0 Postprocessor. ')

        write (6,*) 'Raw data completely loaded from data file'
        write (6,*) ''


        t_max=t(tm,1)
```

```
        do Stn=2,Stnn
           if (t_max>t(tm,Stn)) then
              t_max=t(tm,Stn)
           endif
        enddo

        t_min=t(1,1)
        do Stn=2,Stnn
           if (t_min<t(1,Stn)) then
              t_min=t(1,Stn)
           endif
        enddo

        write(6,100)  t_min,t_max
100       format ('Time impact data avaialble between ',F6.4,' and ',
     &    F6.4,' seconds.')

        write (6,*) ' '
        write (6,101,advance='no')
101       format ('Enter time to analyze: ')
        read (5,*) time

        ci=0

        do while (time>0.)
           ci=ci+1
           do Stn=1,Stnn
              !write (6,*) Stn
              !write (22,*) Stn
!             Cp distribution output
!             Computation of np
              np=0
              do j=1,tm
                 if (time>t(j,Stn)) then
                    np=np+1
                 else
                    np=j
                    exit
                 endif
              enddo
              !write (6,*) 'np ',np

              if ((zc0+dzc*np).lt.zccw(Stn)) then
                 zci(Stn)=zc0+dzc*np
              !   write (6,*) 'zci ',zci(Stn)
              else
```

```
        zci(Stn)=zccw(Stn)
      !    write (6,*) 'zci ',zci(Stn)
      endif

      dx=zci(Stn)/Ke
      !write (6,*) dx
      Fim(Stn)=0.
      do k=1,Ke
        Cp_time(k,Stn)=Cp(k,np,Stn)+
&         (Cp(k,np+1,Stn)-Cp(k,np,Stn))/(t(np+1,Stn)-
&          t(np,Stn))*(time-t(np,Stn))
        !Impact force per unit length
        Fim(Stn)=Cp_time(k,Stn)*dx*2+Fim(Stn)
      enddo
    enddo
    if (ci < 10) then
      format_string = "(A8,I1)"
    else if (ci<100) then
      format_string = "(A8,I2)"
    else
      format_string = "(A5,I3)"
    endif

    write (filename,format_string) "timedata", ci
    dat='.dat'
    ffname=(trim(filename)//dat)

    open (12,file=trim(ffname))
    write (12,109) time
109   format('Pressure distribution at time ',F6.4)
    do Stn=1,Stnn
      write (12,103,advance='no') Stn
103     format('     ',I2,'     ')
    enddo
    write (12,*) ''
    do Stn=1,10
      write (12,104,advance='no')
104     format('============')
    enddo
    write (12,*) ''
    do Stn=1,Stnn
      write (12,120,advance='no') zci(Stn)
120     format('     ',F9.4,'     ')
    enddo
    write (12,*) ''
    do m=1,Ke
```

```
            if (m<10) then
               write (12,106,advance='no') m

106               format (I1,' ')
            else if (m<100) then
               write (12,107,advance='no') m
107               format (I2,' ')
            else
               write (12,108,advance='no') m
108               format (I3,' ')
            endif
            do Stn=1,Stnn
               write (12,105,advance='no') Cp_time(m,Stn)
105               format(F9.4,'   ')
            enddo
            write (12,*) ''
         enddo
         write (6,*) ''
         write (12,*) ''
         do Stn=1,Stnn
            write (6,112) Stn,Fim(Stn)
            write (12,112,advance='no') Stn,Fim(Stn)
112   format('Impact force on station ',I2,' : ',F9.4,' / unit length')
            write (12,*) ''
         enddo
         close(12)
         write (6,*) ''
         write (6,*) 'Cp distribution written to ',trim(ffname)
         write (6,*) ''
         write (6,*) 'Enter a new time point to analyze or ',
   &           'input "0" to quit analysis'
         write (6,*) ''
         write (6,101,advance='no')
102         format ('Enter time to analyze: ')
         read (5,*) time
      enddo
      write (6,*) 'Exiting...'

   endprogram
```

Appendix C1: Input format (inpf.txt) for offset based section definition

5		Number of stations
0.30		Station spacing (Non-dimensionlized on B/2) B = Beam of vessel, H = Depth of vessel hullform
6.66		LCF w.r.t 1st station (Non-dimensionlized on B/2)
1		Input 1 for offset based input
20		Number of points in definition of Stn 1
0	0	
0.05	0.03	
0.1	0.08	
0.13	0.16	
0.12	0.27	
0.06	0.36	
0.08	0.46	
0.13	0.54	Offsets of Stn 1 (Non-dimensionalized on B/2 and H)
0.17	0.575	
0.2	0.6	
0.25	0.635	
0.3	0.66	
0.4	0.71	
0.5	0.77	
0.6	0.82	
0.7	0.875	
0.72	0.89	
0.77	0.93	
0.8	0.955	
0.86	1	
18		Number of points in definition of Stn 2
0	0	
0.05	0.03	
0.1	0.07	
0.13	0.115	
0.15	0.2	
0.13	0.32	
0.17	0.435	
0.2	0.47	Offsets of Stn 2 (Non-dimensionalized on B/2 and H)
0.25	0.51	
0.3	0.55	
0.4	0.61	
0.5	0.67	
0.6	0.72	
0.7	0.78	
0.8	0.85	
0.9	0.94	
0.93	0.98	
0.94	1	
15		Number of points in definition of Stn 3

0	0	
0.05	0.03	
0.1	0.06	
0.15	0.11	
0.2	0.26	
0.25	0.395	
0.3	0.44	
0.4	0.52	Offsets of Stn 3 (Non-dimensionalized on B/2 and H)
0.5	0.58	
0.6	0.63	
0.7	0.695	
0.8	0.77	
0.9	0.86	
0.95	0.95	
0.97	1	

15 → Number of points in definition of Stn 4

0	0	
0.05	0.03	
0.1	0.05	
0.15	0.09	
0.2	0.15	
0.25	0.25	
0.3	0.33	
0.4	0.42	Offsets of Stn 4 (Non-dimensionalized on B/2 and H)
0.5	0.49	
0.6	0.54	
0.7	0.61	
0.8	0.68	
0.9	0.79	
0.95	0.87	
0.99	1	

15 → Number of points in definition of Stn 5

0	0	
0.05	0.015	
0.1	0.035	
0.15	0.07	
0.2	0.115	
0.25	0.175	
0.3	0.235	
0.4	0.335	Offsets of Stn 5 (Non-dimensionalized on B/2 and H)
0.5	0.405	
0.6	0.47	
0.7	0.535	
0.8	0.61	
0.9	0.71	
0.95	0.78	
1	1	

Appendix C2: Input format (inpf.txt) for angle based section definition

3	Number of stations
0.30	Station spacing (Non-dimensionlized on B/2) B = Beam of vessel
6.66	LCF w.r.t 1st station (Non-dimensionlized on B/2)
2	Input 2 for angle based input
20.0 30.0	Definition of Stn 1 (Angle at keel, Angle at hard chine) (deg)
20.0 20.0	Definition of Stn 2
20.0 10.0	Definition of Stn 3

VITA

Ashok Benjamin Attumaly was born in Kochi, India, to Basil Benjamin and Merlin Rosely. He graduated from Kochi Refineries School (Ambalamukal, India) in May 2006 and received his Bachelor of Technology in Mechanical Engineering from the Indian Institute of Technology Delhi in August 2010. He will be receiving his graduate degree in Naval Architecture and Marine Engineering from the University of New Orleans in December 2013.

Benjamin is at present employed as a Naval Architect at North American Shipbuilding in Larose, LA, where he interned during the previous academic year (Fall 2012-Spring 2013). He has interned previously at Elliott Bay Design Group in Seattle, WA (Summer 2012) and Garden Reach Shipbuilders and Engineers Ltd in Kolkata, India (Summer 2008). He has also worked as an Area Manager for Industrial Sales in New Delhi (Petrochemicals) and Mumbai (Bunker fuels) with Bharat Petroleum Corporation Limited (2010-2011).