SELINUS UNIVERSITY
OF SCIENCES AND LITERATURE

REEVALUATION OF LINEAR PROGRAMMING METHODS FOR THE MATHEMATICAL REPRODUCTION AND FAIRING OF LINES OF DAMAGED REGIONS OF SHIPS AND GENERATION OF CUTTING AND NESTING CODES ACCEPTABLE BY AUTOMATIC MACHINES AVAILABLE AT SHIPYARDS

By Ioannis Kolliniatis
Msc in Mechanical Engineering(MIT)
Naval Architect's Degree (MIT)

A DISSERTATION

Presented to the Department of
Mechanical Engineering at Selinus University

Faculty of Engineering and Technology
in fulfillment of the requirements
for the degree of Doctor of Philosophy
in Mechanical Engineering

2023

ABSTRACT

The present thesis is based on the basics of previous work carried out in the field of ship lines fairing by using Linear Programming, a well-known method of Operations Research.
This method has been developed and tested theoretically some years ago but as the capacity of electronic computers at that period was minimal, its accuracy, effectiveness and capacity for ship lines fairing had not been fully investigated.
Initial target of the present work is to develop new, user friendly soft wear to be used as a better tool to overcome previous limitations and by its use to proceed to a more thorough investigation of its potential.
Compared to other well-known and well-established methods of ships lines E to a desired waterline, levels of required ship stability and others.
Parallel to the tremendous and continuing development of the capacity of computers, almost all yards are more and more including in their production procedures numerically controlled processes, useful for passing automatically information to their cutting plates and bending frames machines. This transfer of data is very useful not only for the definition of plates cutting contours but also for nesting and marking.
Modern yards in their programs for construction of new ships are nowadays using well developed packages, such as Tribon and others that not only supply faired and detailed data ready to be used in production but also are capable of generating cutting and nesting tapes ready to be used as inputs to their cutting and marking automatic machines.
This ability minimizes considerably time and man-hours consumed compared to the use of old fashion lofting methods.
In cases of extended hull repairs over or under the waterline as those due to hull damages due to collision, the situation is not so simple. More difficulties are experienced with ships of some age, were detailed construction information is either lacking or it is not available with the required completeness and accuracy.
In addition to the main scope of this work in investigating the pros and cons of using Linear Programming for lines fairing, soft wear is developed that could result to serious improvements in handling more efficiently and thus make cheaper and faster the restoration of extended structural damages to the external hull especially of ships of some age.
The above mentioned soft wear could be used as a low cost alternative as compared to other existing very expensive packages available in the market to improve efficiency and decrease costs of yards in developing countries especially in the field of steel repairs.

PART ONE
HULL GEOMETRICAL RECONSTRUCTION AFTER DAMAGE

CHAPTER 1
INTRODUCTION

Basic motives for undertaking this research effort were the observations and experiences of the researcher during his long employment in shipyards in Greece and abroad.

The fact that significant programs, mainly for construction of ships of naval type, have been undertaken by the two leading shipyards in Greece, resulted, to some extent, to the familiarization of their personnel with modern processes and procedures.

In spite of that, the above familiarization, did not, as one should expect, resulted to significant improvements in their production facilities and procedures.

The reasons why this did not happen are not falling within the scope of this work.

In spite of its positive effects one of those reasons had at the same time considerable drawbacks.

With the exception of certain constructions of smaller complexity vessels such as landing ships, small oilers and patrol craft, in all other cases (frigates, submarines, fast patrol missile boats), the primary effort being to minimize potential problems in the integration of their construction programs led to the following two choices:

First, the construction had to be based almost exclusively to purchasing design, know how, row materials and equipment from previous designers/constructors of the prototype sister ships (Vosper, Blohm &Voss, Hdw, Fincantieri).

Second, all efforts had to be directed towards the timely delivery of the ships, a choice that resulted in placing small if no interest at all in adopting as a secondary target, most modern processes and procedures to the yards' infrastructures.

Especially in the sector of steel works, parallel to the use of most modern production equipment, older and very time-consuming processes, such as lines fairing and frame and plates forming by use of wooden models etc. were kept in use.

Needless to say, that shipyards in Greece, are not a unique exception of delayed modernization.

More specifically in steel repair works and mainly when, due to damage, extended work has to be carried out to the ships cell, production methods in small and medium size yards, globally, remain using logic, methods and processes of previous decades .In spite of the rapid development of digital technology methods, this continuous to remain unchanged.

The present effort attempts to be a further development of previous work in ship lines fairing by the use of Linear Programming, a well-known and broadly used method of optimization.

This method has been developed and theoretically tested in the past but due to the small size and speed of computers at that period, its effectiveness, capabilities and limitations were not fully explored.

The primary target of this work is to develop user friendly soft wear to be used as a tool in overcoming previous limitations, so that its potential and possible drawbacks could be investigated in depth.

Compared to other well-established methods the use of Linear Programming for fairing ship lines, in addition to minimizing deviations between given offsets and those of faired lines, provides some additional, very useful, capabilities.

More specifically in formulating the appropriate Linear Programming problem we can imply very useful constrains such as limiting ship displacement or its cargo deadweight capacity, limits to the breadth or even having a specified level of stability.
Needless to say, that simultaneous satisfaction of a number of such limitations is not always feasible. However, application of Linear Programming will show this inability.
It is very well known that nowadays a number of yards are supplied and using equipment and capabilities to automatically supply cutting machines with appropriate faired data. At the same time in addition to cutting information nesting and marking information can be supplied.
A number of the available today in the Market packages have the above capabilities that compared to the older classical methods of fairing and transmission of information to production equipment, result to very considerable decrease of labor hours and cost.
The high cost of such packages prohibits their purchasing by small yards in developing countries, especially those specializing in ship steel repairs and not in new constructions so much.
This is why In addition to the investigation of the potential of using Linear Programming for fairing ships lines, a parallel very essential target of the soft wear to be developed within the frame of the present work, will be to include to the maximum possible extent a considerable number of the above capabilities.

CHAPTER 2
BASIC TARGETS

As basic targets of the present research the following have been set:

2.1. To develop flexible soft wear to be used in a user friendly way capable to be used even by non-specialized to the core of the method users.

2.2. To provide the ability to easily modify initial data in such a way that alternative cases could be examined (fairing of lines only or fairing plus generation of production data etc.).

2.3. To provide the ability to check in advance the quality of supplied data describing the geometry of the ships cell and either to exclude or to correct the so called bad or wrong points before proceeding to the mathematical fairing.

2.4. To generate using Linear Programming analytic splines for the mathematical description of the ships cell minimizing deviations between given points and what is considered by Naval Architects as "ships fair lines".

2.5. To be able to fair in two dimensions (2D) each station, first with input data in every one of them and then with the produced faired results to curry out subsequent fairing of every waterline.

2.6. To provide the ability to continue with 3-dimensional fairing of the surface (3D).

2.7. To provide the ability to compare supplied offsets with those produced after 2D or 3D fairing.

2.8. To have the ability to check the quality of the faired output in relation with the available alternative choices of the existing today in the Market ready packages for solving large Linear Programming problems.

2.9. Based on the produced faired information to have the ability to produce and show developed areas corresponding to predefined view areas of the ships cell including views of transverse frames and longitudinal stringers. Moreover, to provide the ability to produce acceptable by production machines codes for cutting and marking plates as well as for cutting models of transverse frames from thin plates and to some extend nesting of them.

2.10. For the case of ship repairs, especially for damages above the waterline, to provide a tool for investigating the ability of using a cloud of points taken by laser measurements from the undamaged ship side as a source of data to be used for the applications referred to in the previous paragraphs.

2**.11.** To check the accuracy and quality of produced information

2.12. To provide tools for graphical presentations of all intermediately produced information.

2.13. To proceed to testing the method of fairing and the quality of the relevant soft wear in real cases, so that comparison with known results can be made.

CHAPTER 3
SHIP HULL STEEL REPAIRS AS PRIMARY INITIATIVE OF THIS WORK

Provided that appropriate clauses have been included in the construction contracts, it is possible that all information describing detailed hull geometry, including cutting, marking and nesting data are available to the Owner at electronic form.
In spite of the fact that availability of such information could prove useful in future repairs, their reference to a new ship construction and not to a specific repair should not be disregarded. Moreover, the fact that the above information is related to the specific yard's cutting machines makes difficult its adaptation for repairs in any other yard.
Since it is almost sure that the necessary repairs to a ship hull will not take place at its construction yard, the above adaptation to a specific hull region and to the available to the repair yard automatic cutting machines will be tedious and very time consuming.
Much more difficult is the situation of older ships where the above information is not available and where, in most cases, only drawings of scale 1 to 100 are available.
It has to be reminded that in such cases when repairs on the ship's hull have to be carried out over or under its waterline the following steps will be necessary
a. After ship's arrival to the repairs yard, measurements must be taken from the ship itself and/or available plans and manually faired at 1:1 scale on the ground, a very tedious and time-consuming process.
b. After having the lines faired information of expanded areas, models of shapes of transverse frames in wood, marking and cutting tapes applicable to the specific yard's machines must be produced.
The above steps, although different among yards, have something in common. They all increase cost and they all put the ship 'off hire' for a considerably increased period of time.
The present work is targeting to automate to the maximum possible degree the above actions along the following lines:
a. For repairs on regions of the hull below the waterline measurements are taken from the best scale available drawings.
b. For regions above the waterline a cloud of laser measurements is used, taken from the symmetrical region of the hull of the undamaged side. Alternatively, measurements can be taken from the best scale available drawings.
c. In any of the above cases the ship is not necessary to be at the repair yard for the above information to be collected. It can be collected at any place of the world with the ship continuing its mission and be forwarded to the yard many days before her arrival to the site where actual work has to take place.
d. Having the above information and using the soft wear developed in this work all preliminary preparation, fairing, making models, cutting, marking, nesting tapes and the new pre constructed steel piece to replace the damaged one could wait ready at the pier before ship obligations allow her arrival at the yard.
e. The ship will arrive at the yard when all preconstructions are ready and her obligations allow. Damaged piece will then be removed and the new constructed one will be placed and welded in place.
In conclusion it is more than obvious that there will be a very considerable decrease of the repair duration and thus of the ship remaining out of hire. Moreover, the charge to the Owner by the Yard will be considerably lower, since the ships stay at the yard and/ or on dock will be minimal.

At the same time benefits to the yard will also be considerable. Directly from diminishing man hours for carrying out the work and indirectly by attracting more customers by considerably shortening repairs duration.

PART TWO- MATHEMATICAL BACKGROUND

CHAPTER 4
FAIRING AS PREREQUISITE FOR THE TARGETS OF THIS WORK

According to a specialized USN Naval Surface Warfare Center report [4], a significant number of shipyards that have responded to a relevant questionnaire, expressed their dissatisfaction with the existing, at that time, Computer Aided Lofting systems (CAL).

As they had explained in their answers, this lack of satisfaction was mainly due to the usual practice of CAL software developers to provide excess plate material at seams and butts of plate developed areas. This policy was a result of the necessity to cover inaccuracies originating from the cutting and/or welding automatic machines capabilities as well as from some, inevitable in some cases, errors of the steel fitters during erection.

On the other hand, in order to minimize erection labor hours, shipyard personnel are always in favor of the exact matching of the developed plate contours with those of seams and butts at the locations where they are going to be welded. Such exact matching will minimize cuts and edge preparations work at situ.

Efforts to bridge the above different approaches were not generally successful since the two sides examine the subject from different points of view. **An additional extra difficulty to bridge this gap is due to the fact that in cases of plates with double curvature their developed areas are only approximations** .Of course this is not true for the case of ships or better for some specific areas of their shell, designed and constructed with mathematically produced developable surfaces such as those described by L.W. Ferris in [5].

Very critical for further investigation of the essence of the above different approaches, is to examine if deviations between mathematically produced plate developments and reality could, with the criteria of the usual shipyard practices, be considered acceptable. Moreover, it is necessary to examine whether with the existing mathematical tools, is practically possible, using appropriate soft wear, to locate shell plate areas where inaccuracies of plate developments are critical and therefore inacceptable.

Answers to the above questions are of course depending on the methodology and practices that every yard is using for fairing ship lines.

As it will become apparent in the following paragraphs of this work (as well as from other different approaches of the same problem), analytic splines are often used for the mathematical representation of areas of the ships shell.

Suitability and feasibility of using analytic splines in connection with linear programming as a tool to minimize deviations during fairing of ship lines, must be investigated not only in relation with the above mentioned problematic but also taking into consideration the specific targets of the present work.

In conclusion and taking all the above into consideration, it is obvious that the development of appropriate and reliable soft wear for fairing ship lines by using linear programming as a tool of optimization, must be the first step for achieving the research targets of this work.

CHAPTER 5
BRIEF HISTORICAL BACKGROUND AND EXISTING KNOW HOW

Knowhow and related soft wear in the area of fairing ship lines were developed parallel and as rapidly as the capabilities of electronic computers.
Given that a comparative evaluation of different approaches to this very interesting problem is not part of this work, the following, very briefly, can be said on their development:
a. During the period between 1950 and 1975 polynomials and cyclic arcs have been used.
b. In 1965, following developments in the automotive industry, the idea of using polynomials has been further developed by making use of properties and capabilities of Bezier curves.
c. In 1974 the use of Bezier curves has been further developed in Europe by using B-splines (Basis splines) witch in their parametric use are distinguished to Uniform B-splines (UBS) and Non-Uniform B-splines (NUBS).
d. The use of analytic splines in the US appears at the beginning of the sixties.
e. In 1980 using ideas of the sixties, a denominator has been added in the formulation of B-splines as an effort to affect the appearance of the curves they were representing. The names used for this form of B splines were Rational B-splines (URBS) and Non-Uniform parameterized B-splines (NURBS).
f. After 1990 the use of Bezier, B-splines και NURBS were adopted as standard methods to be used in CAD systems.
All of the processes referred to above can and have been used both in two dimensions (for ships 2D fairing of stations and waterlines) as well as in three-dimension applications (3D direct surface fairing).
Especially in 3d applications, the targeted smooth area is generated using information taken from a three-dimensional truss which is defined by the given points (see Fig. 5.1).

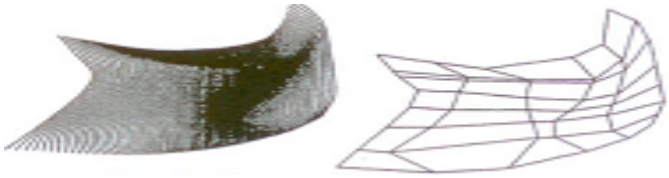


Fig. 5.1 Truss connecting points and faired area

The scope of this work does not include comparisons among many existing and continually improving approaches for fairing ship lines, since such comparisons have been carried out in the past by many researchers and in considerable depth.
The interesting question to which this work is attempting to give an answer is if a combined use of analytic splines in the form proposed by Theilheimer and Starkweather (1), with today's ability to solve problems of optimization with linear programming with a very large number of variables, could be used to improve applicable in many yards, very time (and man-hour) consuming procedures especially for carrying out extended repairs after damage to the ship's hull.

The exact method to be used, following the ideas of S. Berger, A. Webster, R. Tapia, and D. Atkins (2) of using analytic splines together with linear programming for fairing ship lines, will be explained in chapter 8.

CHAPTER 6
JUSTIFICATION FOR THE USE OF LINEAR PROGRAMMING

It is well known that the following basic requirements should be satisfied in fairing ship lines:

a**.** The curves and their first and second derivatives (tangents and curvatures) should be continuous along their lengths.

b**.** The curves should not have unusual to the naval practice inflection points.

c**.** Deviations between the supplied points and the corresponding ones of the produced mathematical curves should be minimal.

In most classical applications of fairing ship lines, the method of mean squares has been used for minimizing the errors between the supplied points and the corresponding ones of the produced mathematically curves. This method has the following two negative characteristics:

a**.** Since it is based on minimizing the sum of the squares of the deviations it is possible to have some very small deviations at some points but at the same time it is also possible that in some other points deviations can be considerably large.

b**.** In spite of the fact that restrictions can be imposed, those restrictions can only be placed in the form of equalities.

On the other hand, with linear programming it is possible to minimize deviations at every single point.

Moreover as we are going to make clear later, exclusion of unusual and unwanted inflection points can be guaranteed if the second derivative of the curve at any one of the given points has the same sign as the corresponding second difference of the submitted for fairing distinct points (by making their product positive). This method of avoiding unwanted inflection points can be applied in linear programming since the method can accept restrictions in the form of inequalities.

Finally, the ability to include in the formulation of the linear programming problem restrictions in the form of inequalities makes possible to cope with other naval architectural requirements such as limitation to displacement, guarantee of a minimum required carrying capacity at a specified waterline, not exceeding a desired beam and others.

CHAPTER 7
EXISTENCE LOCATION AND HANDLING OF SUSPITIOUS POINTS

As suspicious points in the data set are considered those that if remain unchanged in the formulation of the fairing problem could result to inacceptable shape of curves.
The strategy for discovering such suspicious points in the data set is described in detail in [1].
In order that the soft wear to be developed herein is more easily understood some very elementary information will be given below for locating inacceptable points in the data set as well as on the method by which those points can be corrected (if they are curable). Alternatively, those points can be considered as completely wrong and they must immediately be dropped out.
The second difference between any three points on a plane is defined by (7.1)

$$r_1 = \frac{2}{X_2 - X_1} * \left[\frac{Y_3 - Y_2}{X_3 - X_2} - \frac{Y_3 - Y_1}{X_3 - X_1} \right] \qquad (7.1)$$

In figure 7.1 the second difference corresponding to point 2 is negative which means that point 2 lies above the line connecting points 1 and 3.
This observation leads to the conclusion that for every curve fulfilling the above-mentioned requirements of fairness there is a domain where its curvature must have the same sign as the second difference of the corresponding supplied discrete points of the same region.
That means that if we consider the case of 4 points the second differences of the two center ones of which are of different signs then an inflection point is existing between these two points.
Therefore, in the case of 5 points for example if second differences change sign twice, we have two inflection points. Such a case is indicated in figure 7.2.
The possibility that such a case is a conscious decision of the lines designer is almost zero and therefore one should conclude that one or more of the supplied points is wrong.
In other words, if data in stations and waterlines are systematically examined, in regions where in any group of 5 successive points we have two changes in the corresponding second differences inacceptable (suspicious) points are existing.

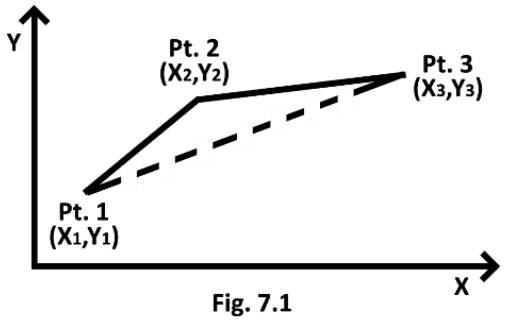


Fig. 7.1
Change of sign of 2nd Difference

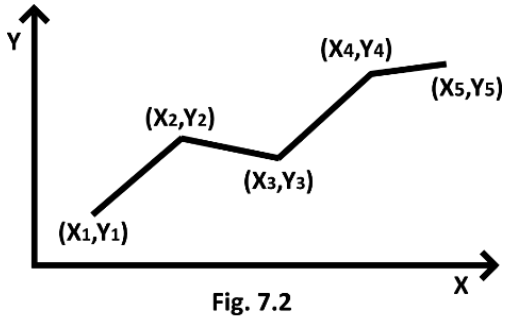


Fig. 7.2
Two concecutive sign changes

It seems logical to assume that in any group of five consecutive points it is very improbable to have two intended wrong points. In case this assumption is true we can distinguish, as indicated in figures 7.3, 7.4 and 7.5 three possibilities.

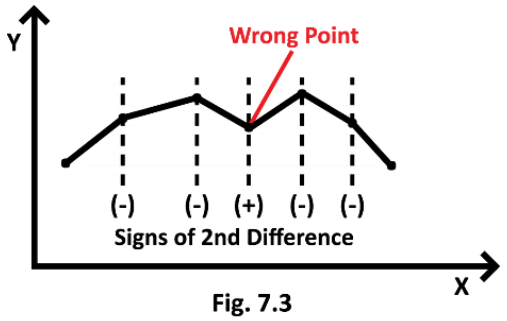


Fig. 7.3
Change of Sign of 2nd difference twice

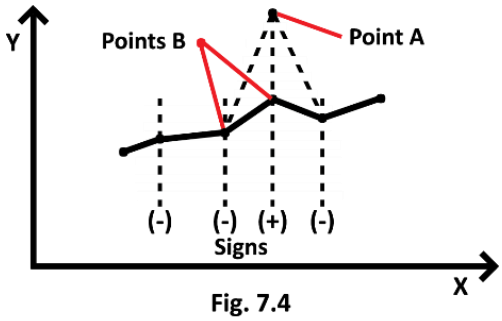


Fig. 7.4
Change of Sign of 2nd difference 3 times

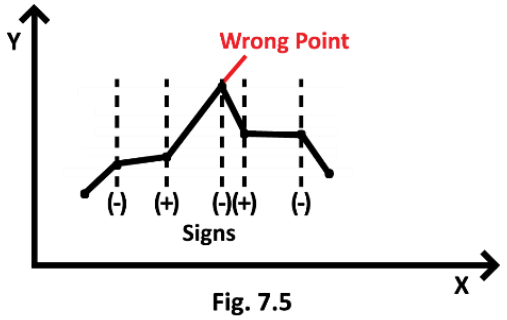


Fig. 7.5
Change of Sign of 2nd difference 4 times

In the case of figure 7.3 we have two consecutive changes of the second derivative sign. It is obvious from this figure that the middle point is the inacceptable one.
In the case of figure 7.4 where we have three consecutive changes of the second derivative, we recognize two possibilities. Inacceptable point to be either of the points marked with A in the figure (determinate case) or one of those marked with B (indeterminate case).
Indeterminate case (points B) is that where omission of either of the two suspicious points results to compatibility of the remaining points. This means that we cannot say which one of the two points is responsible for the initial incompatibility.
Determinate case (point A). If this is the case, we have two possibilities for further investigation. If omission of the first suspicious point cures the problem while omission of the second does not, then the first point is the one that has to be omitted.
On the other hand, if none of the two suspicion points alone is generating the inconsistency then the whole sequence of points must be reexamined for errors.

A last case is that of fig. 7.5 where it is obvious that the middle point is that creating the trouble.
The answer to the question what one should do for the wrong points the following two choices are existing:
a. Drop the wrong points
b. Correct (arbitrarily but logically) the wrong points by moving them within the limits of an acceptable region
In figure 7.6 we can see how this acceptable region is defined in the case that the extensions of lines ab and de lie on the same side of the line connecting the points surrounding the wrong point.
Figure 7.7 shows the definition of the acceptable region in the opposite to the above case.

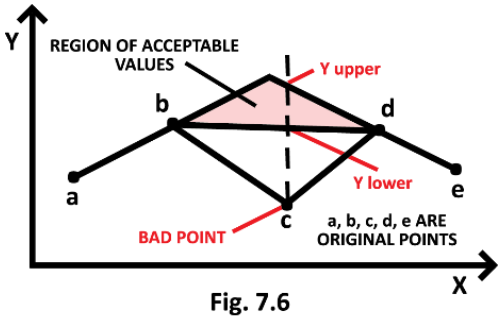


Fig. 7.6
Acceptable region for correction of point

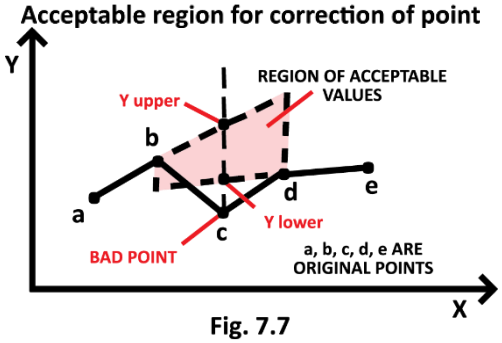


Fig. 7.7
Acceptable region for correction of point

The necessary search for suspicious points and for their necessary correction is part of the soft wear developed as part of the present work.

CHAPTER 8
FORMATION OF THE LINEAR PROGRAMMING PROBLEM

The general description of a linear programming problem is as follows:
Given a system of M linear equations with N unknowns find a set of non-negative values of the variables that they will satisfy given restrictions (constraints) and at the same time will make minimum (or maximum) the value of a given linear function of the variables which is called the objective function.
Theilheimer[1] proposed relation **(8.1)** for the description of the analytic spline. This form is very useful to be used for fairing a curve such that of a ship's station or water-line.

$$Y(X)=A_0+A_1X+A_2X^2+A_3X^3+A_4(X-a_2)^3_{\#}+\ldots..+A_{N+1}(X-a_{N-1})^3_{\#}$$

$$\text{Where: } (X-a)^3_{\#}$$
$$\text{Has value of } (X-a)^3 \text{ when } X \geq a \text{ and } 0 \text{ when } X \leq a$$

(8.1)

8.1. Formulation of the problem of fairing at two dimensions
The requirement that at every given point the second difference should be of the same sign with the corresponding second derivative of the curve we are searching for, together with our aim that the difference between given points and corresponding ones of the curve to be smaller than a small number λ (The objective function) imply that relations **(8.2)** and **(8.3)** below must be satisfied.

$$r_i \cdot Y''(X_i) \geq 0 \quad i=1, \ldots, N$$
(8.2)

$$|Y(X_i)-Y_i| \leq \lambda$$
(8.3)

Relation (8.3) can be transformed as follows:

$$\lambda-Y(X_i) \geq -Y_i$$
(8.4)

$$\lambda+Y(X_i) \geq Y_i$$
(8.5)

$$\text{Where } i=1, 2\ldots., N$$

Linear inequalities (8.2), (8.4) and (8.5) form the constraints to be imposed, so that with the help of linear programming we can find the coefficients $\mathbf{A_0}$ to $\mathbf{A_{N+1}}$ of (8.1) as well as the value of λ.
A number of the ready packages for solving linear programming problems accept only positive values of the variables.
To overcome this difficulty the coefficients $\mathbf{A_i}$ can be expressed in the form:

$$A_i = A_i' - A_i'' \qquad where\ A_i'\ and\ A_i''\ positive$$

After this replacement the linear programming problem for fairing in two dimensions can be stated as follows:

Minimize λ subject to the following constraints:

$$-\lambda + (A_0' - A_0'') + (A_1' - A_1'')X_i + (A_2' - A_3'')X_i^2 + \ldots (A_{N+1}') - (A_{N+1}'')(X_i - a_{N-1})_{\#}^3 \le Y_i$$

$$-\lambda - (A_0' - A_0'') - (A_1' - A_1'')X_i - (A_2' - A_3'')X_i^2 - \ldots (A_{N+1}') - (A_{N+1}'')(X_i - a_{N-1})_{\#}^3 \le -Y_i$$

$$-r_i[2(A_2' - A_2'') + 6(A_3' - A_3'')X_i + \ldots + 6(A_{N+1}' - A_{N+1}'')(X_i - a_{N-1})_{\#}] \le 0$$

Where i=1, 2…, N

The solution of this linear programming problem will give us the coefficients of the mathematical expression (analytic spline) describing as accurately as possible the supplied discrete points as well as λ which expresses the value of the maximum deviation between ordinates of the supplied points and the corresponding ones of the curve.

8.2. Formulation of the problem of fairing at three dimensions

In an analogous way the analytic spline in three dimensions can take the form of **(8.6)**:

$$Y(X,Z) = \sum_{i,j=0}^{3} A_{ij} X^i Z^j + \sum_{i=2}^{N-1} B_i (X - a_i)_{\#}^3 + \sum_{i=2}^{M-1} C_i (Z - b_i)_{\#} + \sum_{i=2}^{N-1} \sum_{j=2}^{M-1} D_{ij} (X - a_i)_{\#}^3 (Z - b_j)_{\#}^3$$

(8.6)

The above expression describes the faired curve in the domain

$$a_1 \le X \le a_N$$
$$b_1 \le Z \le b_M$$

Where α_i = X - distance of p station from the origin and b_j=Z - distance of p waterline from the origin.

$$Y(X_i,Z_j) - Y_{ij} \leq \lambda, \quad \begin{matrix} i = 1, 2,...., N \\ j = 1, 2,...., M \end{matrix}$$

(8.7)

Finally linear programming formulation now takes the form:

$$\left.\begin{matrix} \lambda - Y(X_i, Z_j) \geq -Y_{ij} \\ -\lambda + Y(X_i, Z_j) \geq Y_{ij} \\ r_{ij} \cdot \left(\frac{\partial^2 Y}{\partial X^2}\right)_{X_i, Z_j} \geq 0 \\ s_{ij} \cdot \left(\frac{\partial^2 Y}{\partial Z^2}\right)_{X_i, Z_j} \geq 0 \end{matrix}\right\} \begin{matrix} i = 1, 2,...,N \\ j = 1, 2,...,M \end{matrix}$$

(8.8)

Interested readers can be informed on the formulation of the problem in this case in reference (1).

The approach presented in paragraphs 8.1 and 8.2 above, entirely developed by the authors of (1), is part of the basis of the soft wear developed in this work.

CHAPTER 9
SOLVING OPTIMIZATION PROBLEMS BY LINEAR PROGRAMMING - AVAILABLE SOFT WEAR

9.1 Introduction

The use of Linear Programming as a tool of mathematical optimization received great attention during the fifties.

At the beginning the use of the method was limited to applications with a limited (small) number of variables. The relevant problems were solved either geometrically or by using the method of Simplex Tableau.

From the first steps of its application, in an effort to shorten time required to find solutions, it was recognized that for every problem called **'the primal problem**' for maximizing an objective function depending linearly on some independent variables and subject to linear constraints there existed a formulation of another problem, called '**the dual problem**' of minimizing some other variable.

Provided that the solution of either of the two problems was feasible the objective function would receive the same value no matter which of the two problems was solved.

This property proved to be very useful since by transforming the formulation of the problem from its primal version to its dual and vice versa, reduction of the difficulty and shortening of the duration and effort for its solution could be achieved.

Historically it is useful to state that for one of the first attempts to fair 80 half breadths of a ship at 10 stations and 8 waterlines with an IBM 7090 computer it was necessary to spend approximately 7 hours.

With the formulation of the dual problem this time was shorten to 17 minutes.

With today's capacities of computers this problem is practically non existing but the above example is an indication of the importance of transforming a primal problem to its dual.

Interested readers may look for further details in books of Operations Research, such as (6) and (7).

It should here be repeated that both the objective function and the constraints should depend linearly from the independent variables.

If this is not the case application of nonlinear optimization methods should be applied.

Ebru Sarioz (13) developed a very interesting approach for fairing of ship lines by the use of a nonlinear optimization method.

Studying hers as well as other similar approaches for ship lines fairing (15-20) it **becomes evident that almost all of them are targeting to new ships preliminary designs.**

This is not the case of the present work since our orientation is towards ship steel repairs and not to the design of new ships. Therefore what we are trying to do is to imitate the damaged region of a ship with the undamaged one to the maximum possible degree. Therefore using data in the form of a cloud of random measurements taken by laser from the undamaged side of a ship in connection with fairing them by linear programming optimization methods seems to serve properly our target.

Finally one cannot disregard Kollman's position (14) **"mathematicians have invented powerful methods based on regular networks but practically all networks in shipbuilding are irregular".**

9.2 Software

The rapid development of computer capabilities made necessary the invention of appropriate algorithms and soft wear in the area of optimization applications including solution of Linear Programming problems with an extremely large number of variables and constrains.

In connection with the methods and the targets of this work it will be useful to refer briefly to the capabilities of the processes named Linprog of Matlab R12 [(9) και (10)].

Although the above version of Matlab is an old one it is the one used by the Author for many years during his involvement with this and other works. The same is true for the use of DIGITAL Visual Fortran Version 5.0.A Professional, used also partly for writing some of the problems. Interconnection of programs written in the above two, completely different languages was satisfactory. Of course, since then both languages were further developed and newer versions are available in the Market. However, adaptation of the programs contained in this work to newer versions of both languages is relatively easy, since listings and block diagrams have been included.

Coming back to Linprog, the method of solving Large or Medium size Linear Programming problems is based the on theoretical work of Mehrotra [11] and the algorithm LIPSOL [12] is used extensively for the present research project.

The appropriate formulation of the problem is as follows:

Minimize f(x) subject to the following constraints:

$\mathbf{A_{eq} = b_{eq}}$ $\mathbf{A_{ineq} \leq b_{ineq}}$ and $\mathbf{l \leq x \leq u}$

Where:

$\mathbf{A_{eq}}$ and $\mathbf{A_{ineq}}$ matrices and $\mathbf{b_{eq}}$, $\mathbf{b_{ineq}}$, **l** and **u** vectors.

The Matlab Command for initiating solution of an optimization problem subject to constraints in the form of equalities and inequalities has the following form:

[x,fval,exitflag,output,lambda]=linprog(f,A,b,Aeq,beq,lb,[],[],optimset('Display','iter',
'maxiter',500,'LargeScale','on','TolX', 0.1E+01,'Tolfun', 0.1E-02))

In the right-hand side of this command

f is the vector with the coefficients of the linear objective function to be minimized.

A is the matrix of the coefficients of constraints having the form of inequalities.

Aeq is the matrix of the coefficients of constraints having the form of equalities.

b Vector with values of the right-hand side of inequalities.

beq Vector with values of the right-hand side of equalities.

lb Vector of lower values allowed for the variables.

To the right of the key word **maxiter** we specify the maximum desired number of iteration cycles for reaching the optimum.

To the right of the key word **Largescale** we can specify **on** in case we desire the use of a large-scale optimization algorithm or **off** in case we prefer use of a **Medium scale** algorithm.

To the right of the key word **TolX** we can specify the desired precision of the optimum values of the independent variables we are looking for (**this precision affects time and convergence**).

To the right of the key word **TolFun** we can specify the desired precision of the objective function f(x) to be minimized. **(This precision affects time and convergence).**

To the left side of the above command:
x Is the name of the Matlab vector of the values of the variables that minimize f(x).
fval The name of the Matlab variable containing the minimized value of f(x).
Exitflag Matlab produced Index with values **-1**, **0** or **1** as follows:
-1 Solution not converging.
1 Solution converges to the values of x and f(x).
0 Insufficient for convergence number of cycles (**maxiter**).
Output Name of a Matlab variable with information regarding convergence or non-convergence.
Lambda Name of a Matlab variable for further investigation of non-convergence.

9.3 Sizes of Arrays

As it will be discussed later in using MATLAB for optimization the sizes of the relevant arrays are of importance since, they seriously affect time to reach the solution and ability to converge to a solution.
If for example we have only inequality constraints at N stations and M waterlines those sizes will be:

Size(A)=[2*M*N+(N-2)*M+(M-2)*N ,33+2*(N-2)+2*(M-2)+2*(N-2)*(M-2)]
Size(b)=[2*M*N+(N-2)*M+(M-2)*N
SIze(f)=[33+2*(N-2)+2*(M-2)+2*(N-2)*(M-2)]
Size(lb)=[33+2*(N-2)+2*(M-2)+2*(N-2)*(M-2)]

For N=8 and M=10 we will have
Size(A)=[284,157] size(b)=[284] size(f)=[157] and size(lb)=[157]

While
if we double N,M (N=16,M=20)
Size(A)=[1208,601] size(b)=[1208] size(f)=[601] and size(lb)=[601]

That indicates that in such a case it will be necessary to solve a system of 1208 equations with 601 unknowns.
The specific version of MATLAB used in this work presents inability in solving problems with A having more than 500 rows.
In chapter 10 it will be explained how we can overcome this difficulty.

PART THREE- DEVELOPED SOFTWARE

CHAPTER 10
DESCRIPTION FUNCTIONS AND FLOW CHARTS

10.1 Generalities

Programs developed in this work were written either in FORTRAN or in MATLAB, the choice based on the character and capabilities of each language.
Transfer of data and interconnection of information between programs written in two different languages, although tedious, proved very useful for various reasons such as storing of files, plotting curves and using ready packages for linear programming, available in MATLAB. Especially for the initial supply of input data in a more user-friendly way an XL procedure with appropriate macros has been developed.
Interconnection of above programs and procedures will become clear from their description and from the relevant block diagrams.

10.2. Manual or XL choice for preparation of initial data

All necessary input information is supplied to the control program by a FORTRAN input file (C:\naval\fair.bak8\mat3d.dat), containing the following information:
a. Geometrical data taken from the ships plans or measured by laser (modified by triangulation) as it will be explained later.
b. Control numbers indicating actions to be taken
The above file can be easily created by an XL process named **"PREPARE DATA IN XL_test1"** stored in C:\NAVAL|fair.bak8.
By pressing the button **"DOKO**" at the left bottom corner of diagram 10.2.1 the table shown in fig 10.2.1 will appear. By pressing the button named "**bottom"** of the upper right area (white area) all tables possibly filled with previous data will be emptied.

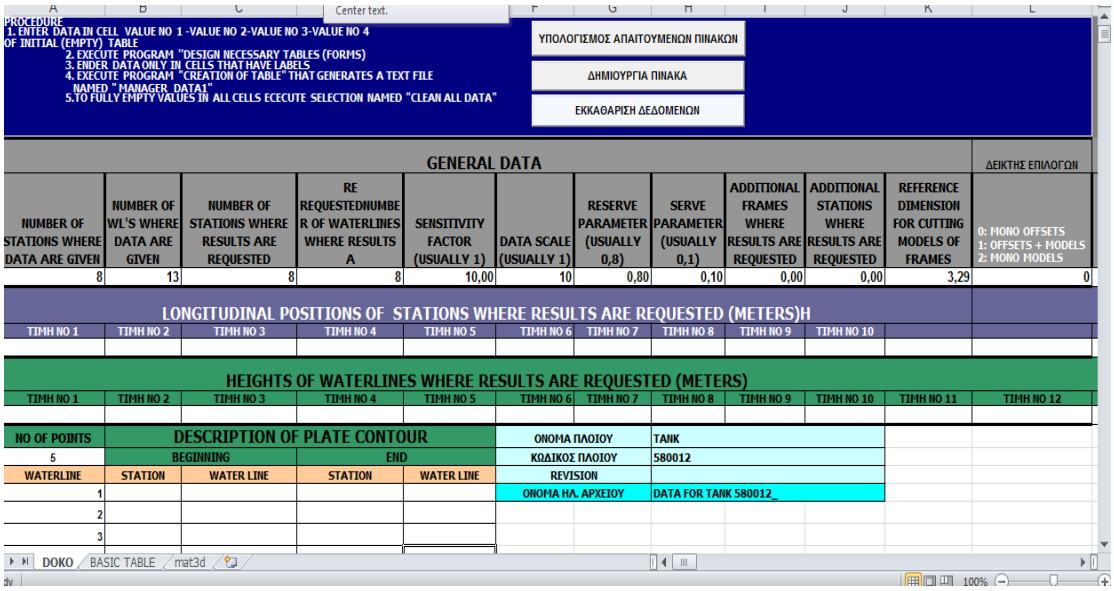


Fig. 10.2.1
Empty XL form for supplying initial data

New data in cells of line 4 and cell 12A must then be entered and by pressing **the** button named **"top"** of the upper right area (white area) a new empty table corresponding to the new size of problem will be created (fig. 10.2.2).

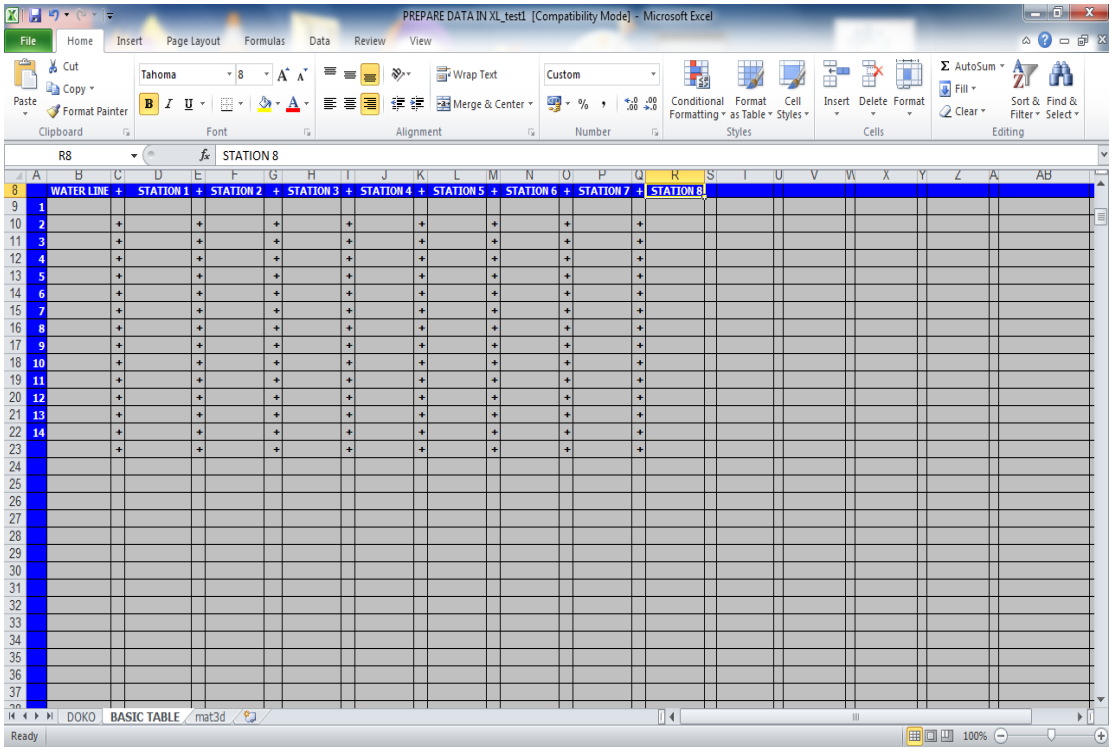


Fig. 10.2.2
Empty XL form to supply offsets

After entering offset values in table 10.2.2 one should press the **"middle"** button of the upper right area (white area) in order to export the necessary data to FORTRAN input file (C:\naval\fair.bak8\mat3d.dat) containing the supplied new input values.
The empty tables 10.2.1 and 10.2.2 will now look as shown in figures 10.2.3 and 10.2.4 below.

3. ENTER DATA ONLY IN CELLS THAT HAVE LABELS
4. EXECUTE PROGRAM "CREATION OF TABLE" THAT GENERATES A TEXT FILE NAMED "MANAGER_DATA1"
5.TO FULLY EMPTY VALUES IN ALL CELLS ECECUTE SELECTION NAMED "CLEAN ALL DATA"

ΔΗΜΙΟΥΡΓΙΑ ΠΙΝΑΚΑ

ΕΚΚΑΘΑΡΙΣΗ ΔΕΔΟΜΕΝΩΝ

GENERAL DATA

ΔΕΙΚΤΗΣ ΕΠΙΛΟΓΩΝ

NUMBER OF STATIONS WHERE DATA ARE GIVEN	NUMBER OF WL'S WHERE DATA ARE GIVEN	NUMBER OF STATIONS WHERE RESULTS ARE REQUESTED	RE REQUESTEDNUMBER OF WATERLINES WHERE RESULTS A	SENSITIVITY FACTOR (USUALLY 1)	DATA SCALE (USUALLY 1)	RESERVE PARAMETER (USUALLY 0,8)	SERVE PARAMETER (USUALLY 0,1)	ADDITIONAL FRAMES WHERE RESULTS ARE REQUESTED	ADDITIONAL STATIONS WHERE RESULTS ARE REQUESTED	REFERENCE DIMENSION FOR CUTTING MODELS OF FRAMES	0: MONO OFFSETS 1: OFFSETS + MODELS 2: MONO MODELS
8	13	8	8	10,00	10	0,80	0,10	0,00	0,00	3,29	0

LONGITUDINAL POSITIONS OF STATIONS WHERE RESULTS ARE REQUESTED (METERS)H

ΤΙΜΗ ΝΟ 1	ΤΙΜΗ ΝΟ 2	ΤΙΜΗ ΝΟ 3	ΤΙΜΗ ΝΟ 4	ΤΙΜΗ ΝΟ 5	ΤΙΜΗ ΝΟ 6	ΤΙΜΗ ΝΟ 7	ΤΙΜΗ ΝΟ 8	ΤΙΜΗ ΝΟ 9	ΤΙΜΗ ΝΟ 10
0,07683	0,15677	0,23652	0,3162	0,3974	0,476	0,5565	0,638		

HEIGHTS OF WATERLINES WHERE RESULTS ARE REQUESTED (METERS)

ΤΙΜΗ ΝΟ 1	ΤΙΜΗ ΝΟ 2	ΤΙΜΗ ΝΟ 3	ΤΙΜΗ ΝΟ 4	ΤΙΜΗ ΝΟ 5	ΤΙΜΗ ΝΟ 6	ΤΙΜΗ ΝΟ 7	ΤΙΜΗ ΝΟ 8	ΤΙΜΗ ΝΟ 9	ΤΙΜΗ ΝΟ 10	ΤΙΜΗ ΝΟ 11	ΤΙΜΗ ΝΟ 12
0,12	0,13	0,14	0,15	0,16	0,18	0,2	2,4				

NO OF POINTS	DESCRIPTION OF PLATE CONTOUR			
5	BEGINNING		END	
WATERLINE	STATION	WATER LINE	STATION	WATER LINE
1	0,1568	0,14	0,1568	0,18
2	0,1568	0,18	0,3974	0,18
3	0,3974	0,18	0,3974	0,14
4	0,3974	0,14	0,1568	0,14

ΟΝΟΜΑ ΠΛΟΙΟΥ	TANK
ΚΩΔΙΚΟΣ ΠΛΟΙΟΥ	580012
REVISION	
ΟΝΟΜΑ ΗΛ. ΑΡΧΕΙΟΥ	DATA FOR TANK 580012_

Fig. 10.2.3
Filled XL form with initial data

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
		WATER LINE	+	TOMH NO 1	+	TOMH NO 2	+	TOMH NO 3	+	TOMH NO 4	+	TOMH NO 5	+	TOMH NO 6	+	TOMH NO 7	+	TOMH NO 8	
1				0,07683		0,15677		0,23652		0,3162		0,3974		0,476		0,5565		0,638	
2		0,12	+	0,7886	+	0,7733	+	0,7523	+	0,7359	+	0,7172	+	0,7025	+	0,6852	+	0,6667	
3		0,13	+	0,7977	+	0,7821	+	0,7618	+	0,7458	+	0,7273	+	0,7125	+	0,6951	+	0,6765	
4		0,14	+	0,8059	+	0,7902	+	0,7708	+	0,7549	+	0,7369	+	0,7218	+	0,7044	+	0,6859	
5		0,15	+	0,8135	+	0,7974	+	0,7792	+	0,7634	+	0,7459	+	0,7304	+	0,7132	+	0,695	
6		0,16	+	0,8203	+	0,8041	+	0,7872	+	0,7713	+	0,7542	+	0,7385	+	0,7215	+	0,7036	
7		0,17	+	0,8266	+	0,8101	+	0,7946	+	0,7786	+	0,7619	+	0,746	+	0,7293	+	0,7117	
8		0,18	+	0,8323	+	0,8157	+	0,8016	+	0,7855	+	0,7691	+	0,753	+	0,7367	+	0,7194	
9		0,19	+	0,8376	+	0,8209	+	0,8082	+	0,7919	+	0,7758	+	0,7569	+	0,7437	+	0,7266	
10		0,2	+	0,8426	+	0,8258	+	0,8144	+	0,7979	+	0,7821	+	0,7659	+	0,7503	+	0,7335	
11		0,21	+	0,8473	+	0,8304	+	0,8203	+	0,8036	+	0,7881	+	0,7719	+	0,7566	+	0,7401	
12		0,22	+	0,8518	+	0,8349	+	0,8259	+	0,8091	+	0,7937	+	0,7776	+	0,7626	+	0,7464	
13		0,23	+	0,8562	+	0,8394	+	0,8311	+	0,8143	+	0,7991	+	0,7832	+	0,7684	+	0,7525	
14		0,24	+	0,8606	+	0,8438	+	0,8362	+	0,8195	+	0,8044	+	0,7887	+	0,774	+	0,7585	
			+		+		+		+		+		+		+		+		

Fig. 10.2.4
Filled XL form with offsets

The basic input file to FORTRAN -MATLAB fairing programs is shown in Fig. 10.2.5.
In supplying input data in this way, offsets in every station will correspond to the same group of waterlines.

```
07.000   10.0000  03.0000  4.0000   1.0000   01.0000  00.8000 0.1000  0 0  03.2900  1 0.6
            00.0000     01.0000     02.0000     03.0000     04.0000     05.0000     06.0000
09.2750  +  06.8785  +  06.8054  +  06.5813  +  06.1898  +  05.5959  +  04.7237  +  03.3636
11.2750  +  08.3617  +  08.3017  +  08.1190  +  07.8050  +  07.3429  +  06.7021  +  05.8240
13.2750  +  09.8450  +  09.7940  +  09.6397  +  09.3767  +  08.9957  +  08.4808  +  07.8053
15.2750  +  11.3282  +  11.2840  +  11.1503  +  10.9237  +  10.5985  +  10.1651  +  09.6088
17.2750  +  12.8114  +  12.7723  +  12.6544  +  12.4552  +  12.1710  +  11.7955  +  11.3196
19.2750  +  14.2947  +  14.2597  +  14.1541  +  13.9763  +  13.7236  +  13.3917  +  12.9745
21.2750  +  15.7779  +  15.7462  +  15.6506  +  15.4901  +  15.2625  +  14.9647  +  14.5926
23.2750  +  17.2612  +  17.2322  +  17.1449  +  16.9985  +  16.7913  +  16.5211  +  16.1848
23.3750  +  17.3353  +  17.3064  +  17.2196  +  17.0738  +  16.8675  +  16.5986  +  16.2639
24.2750  +  18.0028  +  17.9750  +  17.8913  +  17.7511  +  17.5528  +  17.2945  +  16.9735
            01.0000     02.0000     03.0000     04.0000  5.
   11.275  13.275  15.275   17.275 19.275 21.275   23.275
   4
01.0000      13.2750    02.0000  17.2750
02.0000  17.2750  03.0000  17.2750
03.0000  17.2750     03.0000  13.2750
```

Fig. 10.2.5
Basic input files to FORTRAN (mat3d.dat)

If it is desired to give offset values at different waterlines for each station, manual preparation of file mat3d_old.dat (**Fig 10.2.6**) will be necessary, indicating the above correspondence (see for example station 1 where offsets are given at waterlines .25 .75 1. 1.5 2. 3. 3.5 while in station 7 offsets are given at waterlines .25 .5 1. 1.5 2. 2.5 3.5.In such a case input data are supplied to the programs through file mat3d_old.dat.
In files mat3d.dat and mat3d_old.dat the + sign in front of offsets specifies that their values are allowed to change during the 2d fairing process. If * is placed instead of + the corresponding offset(s) should remain unchanged during the 2d fairing. During the subsequent 3d fairing values of all offsets are allowed to change.

```
07.000  07.0000         10.00   09.00   1.0000  01.0000  00.8000 00.1000  0 0  03.2900 1 0.6
         0.8000      01.6000     02.4000     03.2000     04.0000     04.8000     05.6000
00.2500  +  08.5730  +  08.4590  +  08.3380  +  08.2090  +  08.0740  +  07.9320  +  07.7830
00.5000  +  09.2720  +  08.8720  +  09.0480  +  08.6280  +  08.4960  +  08.6620  +  08.2130
00.7500  +  09.4940  +  09.3880  +  09.2770  +  08.9250  +  09.0330  +  08.9020  +  08.7650
01.0000  +  09.8080  +  09.7110  +  09.6080  +  09.4990  +  09.6310  +  09.2620  +  09.1360
01.2500  +  10.0190  +  09.9310  +  09.8370  +  09.7370  +  09.8110  +  09.5200  +  09.4030
02.0000  +  10.2640  +  10.0840  +  10.1170  +  09.9000  +  09.9450  +  09.7090  +  09.6020
02.5000  +  10.3340  +  10.2720  +  10.2030  +  10.1280  +  10.0400  +  09.9630  +  09.8730
.25 .25 .25  .25 .25 .25 .25
.75 .5  .75  .5  .5  .75 .5
1.   1.   1. .75  1.  1.  1.
1.5  1.5 1.5  1.5  2. 1.5  1.5
2.   2.  2.   2. 2.5  2. 2.
3.   2.5 3.  2.5 3. 2.5  2.5
3.5  3.5 3.5 3.5 3.5 3.5 3.5
0.8 1.2 1.25 1.399 1.6 1.7 1.733 1.866  2. 2.133        2.799  3.049 3.5    3.2 3.399 3.5
0.25 .375 .5  .625 .75 .972 1.  1.194 1.25        1.416  1.861  2.25   3. 3.5       2.083 2.25
05.0000
01.2000  00.3750  00.8000  00.5000
00.8000  00.5000  02.1330  01.2500
02.1330  01.2500  01.7330  00.7500
01.7330  00.7500  01.6000  00.3750
```

Fig. 10.2.6
Modified input file to FORTRAN (mat3d_old.dat)

10.3 Programs documentation

10.3.1 Main program (checkpts_no_debug_mad.for)

1. INTERNAL NAME - TEST1
2. AUTHOR - Ioannis Kolliniatis
3. TYPE – Specific
4. SHORT DESCRIPTION - Reads input data and control numbers related to specific tasks
5. LANGUAGE – Fortran
6. FORTRAN LISTING - See ANNEX (1) of Appendix A
7. FLOW CHART - As shown in PLATE (1) of Appendix A

8 CALLS-Subroutine "chkpts_fmatr_no_debug.for" and MATLAB module "FAIR8.m"

9 INPUT VALUES

RESPONCES TO QUESTIONS THAT APPEAR ON SCREEN

nio Value 2(direct 3D fairing), Value 3(2D+3D fairing)

IDEB Value 0 (No Debug), Value 1(Debug)

N_PO No of cycles for points correction

OFF_KIND

- 0 input offsets must be given at same waterlines for all stations (input is taken from file "mat3d.dat").
- 1 input offsets can be given at different waterlines at different stations (input is taken from file "mat3d_old.dat").

LINE 1 of file "mat3d.dat or mat3d_old.dat"

N: Number of **stations** where data are given (for example theoretical stations)

M: Number of **waterlines** where data are given (for example theoretical waterlines)

N1: Number of **stations where results are requested** (for example actual real frames)

M2:Number of **waterlines where results are requested** (for example actual water-lines)

fac: Dummy

scale: Scale factor on data length readings

spax: IRRELEVANT

spaz: IRRELEVANT

NOPOX: No of **additional stations** where results are desired (for example 5)

NOPOZ: No of **additional WL's** where results are desired (for example 10)

off: Maximum allowed deviation between supplied offsets and those after their correction and fairing , at same waterlines and stations.

In case the actual deviation exceeds "off" the process stops at this point and can be repeated after the conflict is resolved.

ISENS: Index with value as follows:

- 0 Produce faired lines plan and shell developments
- 1 Produce faired lines plan, shell developments, cut and mark files
- 2 Produce faired lines plan only

Tolf: Acceptable max deviation between data and results after fairing

BLOCK 2
line 2 Positions of stations (0.8 1.6 …., N in number)

BLOCK 3
Input offsets (M lines) as follows
[WL, offsets (N in number)]

BLOCK 4
Stations where results are requested (one line) - N1 in number

BLOCK 5
Waterlines where results are requested (one line) - M1 in number

DATA IN FOLLOWING BLOCKS (6 and 7) NECESSARY only if ISENS=0 or 1
BLOCK 6
No of view sides (straight lines) of polygon plate to be developed (KKO in number)

BLOCK 7
Coordinates of start and end points of view sides of the polygon area (plate) to be developed (KKO lines) as follows:
[beg x, beg z, end x, end z]

IMPORTANT NOTE: [beg x, beg z, end x, end z] must exist in Blocks 4 and 5)

10._LONG DESCRIPTION
After reading input data this program calls subroutine "chkpts_fmatr_no_debug_mad.for" **twice"** (one time for stations and one for waterlines) which prepares all necessary information for calling other subprograms for correcting wrong points (those not acceptable in ship design practice) and then for carrying out 2d fairing of sections and waterlines if it has been requested.
During the first call of "chkpts_fmatr_no_debug_mad.for" stations are faired and during the second waterlines
Important Note: In case only direct 3d fairing has been requested no correction of wrong points and no 2d fairing will take place. Therefore, the possibility the faired lines not to be acceptable according to ship design practice, is very high.
In case more than one cycles of fairing are required for testing reasons the required number of cycles must be specified (parameter N_PO to questions posed on screen)
Before exit from this program a call of MATLAB module "FAIR8.m" will take place, which as it is going to be explained at its documentation, will do the 3d fairing job by the use of Linear Programming.

10.3.2 Subroutine “chkpts_fmatr_no_debug_mad.for”

1. INTERNAL NAME - chkpts_fmatr
2. AUTHOR - Ioannis Kolliniatis
3. TYPE – Specific
4. SHORT DESCRIPTION - Supplied with data and control numbers from the main input program this subroutine calls two others first for correction of offsets, if necessary and subsequently for 2D fairing of stations. **Fairing of waterlines using as input the already faired offsets follows.**
5. LANGUAGE – Fortran
6. FLOW CHART - As shown in PLATE 2 of Appendix A
7. FORTRAN LISTING - See ANNEX 2 of Appendix A
8. INPUT VARIABLES Those contained at its call
9. LONG DESCRIPTION – First this routine, calling subroutine “chkpts_no_debug_mad.for“ checks the quality of offsets supplied as input to the stations (M points) and makes correction of unacceptable offsets .Then using subroutine “ffmatr1_no_debug.f” fairs the stations using the corrected (if necessary) offsets and produces new ones at the requested stations (M1).Using these faired offsets every waterline is then faired (without any further search for bad points) and new offsets are produced at the requested stations and waterlines to be used in the next phase of 3D fairing.
10. CALLSubroutines “chkpts_no_debug_mad.for“ and “ffmatr1_no_debug.f”

10.3.3 Subroutine “chkpts_no_debug_mad.for”

1. INTERNAL NAME - chkpts
2. AUTHOR - Ioannis Kolliniatis
3. TYPE - Specific
4. SHORT DESCRIPTION - Supplied with data and control numbers from the calling routine **“chkpts_fmatr_no_debug_mad.for”** detects and corrects ‘Bad points’ and returns corrected offsets.
5. LANGUAGE – Fortran
6. FLOW CHART - As shown in PLATE (3) of Appendix A
7. FORTRAN LISTING - See ANNEX (3) of Appendix A
8. INPUT VARIABLES - Those contained at its call
9. LONG DESCRIPTION - First this routine, examines the sequence of changes of 2^{nd} differences of successive overlapping groups of seven offsets of each station and following the principles referred to in chapter 7 locates possible ‘bad points’ (for instance group 1 2 3 4 5 6 7, then group 2 3 4 5 6 7 8 and so on). If wrong points are found and the situation is curable, appropriate steps of correction are taken following the principles referred to in chapter 7.
10. CALLS - None

10.3.4 Subroutine “ffmatr1_no_debug_mad.f”

1. INTERNAL NAME -“ffmatr1”
2. AUTHOR - Ioannis Kolliniatis
3. TYPE - Specific
4. SHORT DESCRIPTION - Supplied with data and control numbers from the calling routine **“chkpts_fmatr_no_debug_mad.for”** and corrected as referred at 10.3.3 above offsets, this routine proceeds to 2d fairing first of stations and then of waterlines.
5. LANGUAGE – Fortran
6. FLOW CHART - As shown in PLATE (4) of Appendix A.
7. FORTRAN LISTING - See ANNEX (4) of Appendix A.
8. INPUT VARIABLES - Those contained at its call.
9. LONG DESCRIPTION - First this routine prepares MATLAB custom module mat8.m necessary for **matlab/linprog** to prepare coefficients for carrying out fairing based on Linear Programming as descried in (8.10). After mat8 has run, a MATLAB custom module named mplot8.m is automatically generated which contains the calculated offsets and is useful for producing plots of each corrected and faired station and each waterline, so that the user can inspect their shape.
10. CALLS - None

10.3.5 MATLAB Process “mynew128.m ”

1. AUTHOR - Ioannis Kolliniatis
2. TYPE – Specific
3. SHORT DESCRIPTION - This module is called by “check-pts_no_debug_mad.for” only when ISENS has values 0 or 1 i.e. if shapes of developed areas of view polygons and/or shapes of frame models are required.
4. LANGUAGE – MATLAB
5. FLOW CHART - As shown in PLATE (5) of Appendix A.
6. MATLAB LISTING - See ANNEX (5) of Appendix A.
7. INPUT VARIABLES - Those loaded.
8. LONG DESCRIPTION – First module “mynew12_previous8”, is called which in turn calls “mynew42_previous8” and “mynew52_previous8” in sequence. Those modules locate perimeter points lying on the view sides of the polygon the development of which has been requested. Then modules ”mynew428” and ”mynew528” are automatically called to prepare and export the following Fortran files, useful for the next steps.
9. IMPORTANT Fortran files exported for further use:

LLENGTHS.dat
HHEIGHTS.dat
RIO.dat
RIO1.dat
RIO2.dat

10. CALLS Matlab modules “mynew12_previous8”, ”mynew428”, ”mynew528”

Listings of the above modules are contained in ANNEX (5) of Appendix A. However no block diagrams are included since they are self-explanatory

10.3.6 MATLAB Process "FAIR8.m "

1. AUTHOR - Ioannis Kolliniatis
2. TYPE – Specific
3. SHORT DESCRIPTION -This module is called by "check-pts_no_debug_mad.dat" Fortran Program (description of which was given in 10.3.1. It loads data produced by it and prepares a 2D array, named AA which contains the multipliers of the inequality constraints of the linear problem to be solved (see equation 8.6 of chapter 8 and array A of the MATLAB formulation of the problem of chapter 9).
4. LANGUAGE – MATLAB
5. FLOW CHART : As shown in PLATE (6) of Appendix A.
6. MATLAB LISTING - See ANNEX (6) of Appendix A.
7. INPUT VARIABLES - Those loaded.
8. LONG DESCRIPTION –The multipliers referred to above are computed as follows:

A_{ij} multipliers for offset constraints: computed by the main part of module FAIR8.
B_I multipliers for offset constraints: computed by module COMPUTE_B
C_J multipliers for offset constraints: computed by module COMPUTE_C
D_{ij} multipliers for offset constraints: computed by module COMPUTE_D

A_{ij} multipliers for 2^{nd} differences relative to x constraints computed by module COMPUTE_Y_X
B_I multipliers for 2^{nd} differences relative to x constraints: computed by module COMPUTE_Y_X_B
C_J multipliers for 2^{nd} differences relative to x constraints: computed by module COMPUTE_Y_X_B
D_{ij} multipliers for 2^{nd} differences relative to x constraints: computed by module COMPUTE_Y_X_D

A_{ij} multipliers for 2^{nd} differences relative to z constraints: by module "COMPUTE_Y_Z"
B_I multipliers for 2^{nd} differences relative to z constraints: by module "COMPUTE_Y_Z_C1"
C_J multipliers for 2^{nd} differences relative to z constraints: by module "COMPUTE_Y_Z_C1"
D_{ij} multipliers for 2^{nd} differences relative to z constraints: by module "COMPUTE_Y_Z_D1"

Details of above sub arrays are shown the following table

MODUL	**Sub array of AA multi-pliers (see equation (8.6)**	**Name and size of param-eter**
COMPUTE_B	Sub array of B multipliers of Equation (8.6) for offsets	BBB_FIN Size [2*N*M,2*(N-2)]
COMPUTE_ C	Sub array of C multipliers of Equation (8.6) for offsets	CCC_FIN Size [2*N*M,2*(M-2)]
COMPUTE_ D	Sub array of D multipliers of Equation (8.6) for offsets	DDD_FIN Size {2*N*M,2*(N-2)-*(M-2)]
COMPUTE_Y_X	Sub array of A multipliers of Equation (8.6) for SEC-OND DIFFERENCES	Y_X Size [(N-2)*M, 33]
COMPUTE_Y_X_B	Sub array of B multipliers of Equation (8.6) for SEC-OND DIFFERENCES	BBBB_DER Size[(N-2)*M, 2*(N-2)
COMPUTE_Y_X_D1	Sub array of D multipliers of Equation (8.6) for SEC-OND DIFFERENCES	DIO_X_DER2 Size[(N-2)*M,2(*M-2)]
COMPUTE_Y_Z	Sub array of A multipliers of Equation (8.6) for SEC-OND DIFFERENCES	Y_Z Size[(M-2)*N ,33]
COMPUTE_Y_Z_C1	Sub array of C multipliers of Equation (8.6) for SEC-OND DIFFERENCES	CCCC_Z_DER1 Size [(M-2)*N,2*(M-2)]
COMPUTE_Y_Z_D1	Sub array of D multipliers of Equation (8.6) for SEC-OND DIFFERENCES	DIO_Z_DER2 Size [(M-2)*N,2*(N-2)*(M-2)]

FAIR8	Sub array of A multipliers of Equation (8.6) for offsets	AA(1:2nm,1:33) where nm2=2*N*M Size(2*N*M,33]
COMPUTE_Y_X_B	Sub array of C multipliers of Equation (8.6) for SEC-OND DIFFERENCES	Zero array CCC_DER Size (N-2)*M,2*(N-2)]
COMPUTE_Y_Z_B	Sub array of B multipliers	Zero array BBB_DER

	of Equation (8.6) for SECOND DIFFERENCES	Size [(M-2)*N ,2*(M-2)]

The above 12 sub arrays are combined to form array AA which is the output of this module.

AA (1:2*N*M)	BBB_FIN	CCC_FIN	DDD_FIN
Y_X	BBBB_DER	CCC_DER	DIO_X_DER
Y_Z	BBB_DER	CCCC_Z_DER	DIO_Z_DER3

Block diagrams for modules generating above sub arrays (except for FAIR8) are not included since, they are self-explanatory from their Matlab listings

9.-CALLS

Depending on the number of lines of AA (number of restrictions) FAIR8 calls module "breadths8_fair8_correct_1.m" if this number does not exceeds 500 , or breadths8_fair8_corrcet8ggg elsewhere.

10.3.7 MATLAB Process "breadths8_fair8_correct_1.m "

1. AUTHOR - Ioannis Kolliniatis
2. TYPE – Specific
3. SHORT DESCRIPTION -This module is called by Matlab Process FAIR8.m (the description of which is given in paragraph 10.3.6) if the number of restrictions of the corresponding Linear Programming problem does not exceed 500.
4. LANGUAGE – MATLAB
5. FLOW CHART - As shown in PLATE (7) of Appendix A.
6. MATLAB LISTING - See ANNEX (7) of Appendix A.
7. INPUT VARIABLES - Those loaded and array AA as produced by module "FAIR8.m".
8. LONG DESCRIPTION –After loading the necessary information this module proceeds to optimization using Matlab "linprog" function for Large size problems. If desired the user can change parameters such as using Medium size algorithm and/or tolerances (Tolfun ,tolX etc) .**In some cases that may be found useful to achieve convergence.**

In case a plate development is desired by another custom module named "test_test_test8_mad.m" (initially specified ISENS value is 0 or 1) subroutine "arc_lengths" is called to produce piecewise expansions of station and waterline contours .

Plots of portions of body plan and waterlines plan appear for inspection and other necessary for further use information is saved in directory c:\NAVAL\fair.bak8\TEMP

9.- CALLS Matlab module "linprog" and custom module "arc_lengths".

10.3.8 MATLAB Process "breadths8_fair8_correct8ggg.m "

1.-AUTHOR - Ioannis Kolliniatis
2.-TYPE – Specific
3.-SHORT DESCRIPTION -This module is called by Matlab Process FAIR8.m (description of which is given in paragraph 10.3.6) only if the number of restrictions of the corresponding Linear Programming problem exceeds 500.
4.-LANGUAGE – MATLAB
5.-FLOW CHART - As shown in PLATE (8) of Appendix A.
6.-MATLAB LISTING - See ANNEX (8) of Appendix A.
7.-INPUT VARIABLES - Those loaded and array AA as produced by module "FAIR8.m".
8.-LONG DESCRIPTION –After loading the necessary information this module proceeds to the partitioning of AA. From paragra ph (9.3) of chapter 9 we know that the number of rows of AA is

NO_LI=2*M*N+(N-2)*M+(M-2)*N [10.3.8a]

while the number of columns is

NO_CO= 33+2*(N-2)+2*(M-2)+2*(N-2)*(M-2 [10.3.8b]

AA is partitioned to sub arrays (denoted as aa_k) in such a way that each aa_k has the same number of columns (NO_CO) with AA, while we want that its number of rows not to exceed 500.
Parameter FA in this module defines the number of rows of sub arrays aa_k with the exception of the last one, which (depending on the size of AA) can have an equal or smaller number of rows

We can denote with PS the **basic partition** step defined as:

PS=Value of [10.3.8a] for N=FA
Then
PS=2*M*FA+(FA-2)*M+(M-2)*FA [10.3.8c]

For FA=3 PS=10*M-6

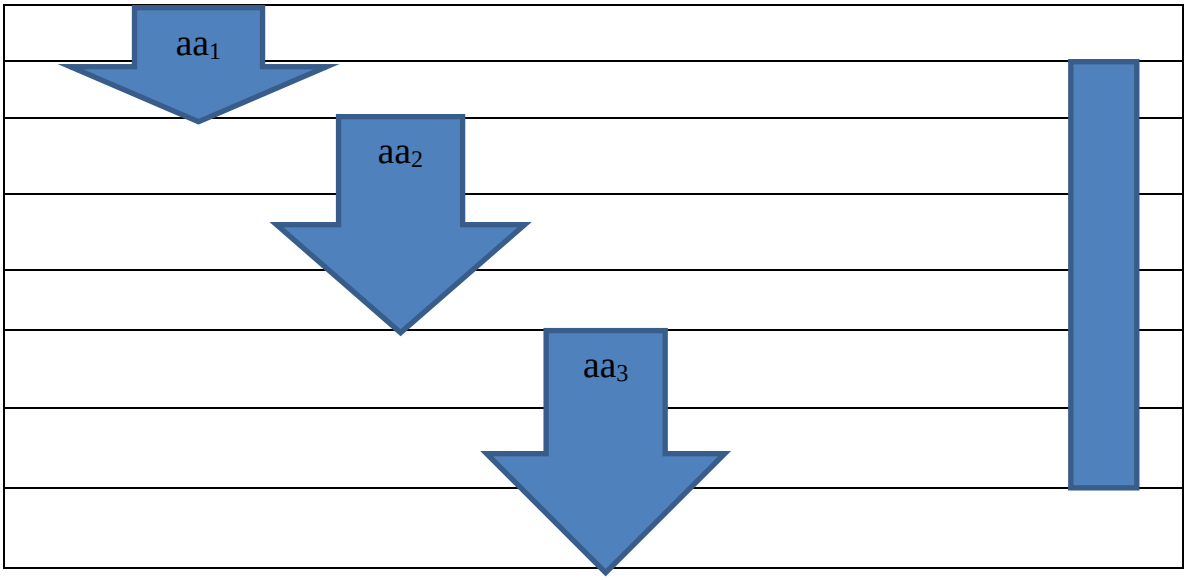
Fairing takes place progressively in longitudinal groups (segments) the first of which consists of the following rows:

For offsets constrains	2*FA*M
For 2nd difference constrains along x-axis	(FA-1)*M
For 2nd difference constrains along z-axis	(M-2)*FA
Total rows of first segment	4*FA*M-2*FA-M

For better fairness it looks reasonable that subsequent segments should overlap their previous ones (on one station).
In those cases each of the intermediate segments, except the last one, will consist of the following rows:
For offsets constrains 2*(FA+1)*M

For 2nd difference constrains	along	x-axis	(FA+1)*M
For 2nd difference constrains	along	z-axis	(M-2)*(FA+1)
Total rows of intermediate segment			4*(FA*M+M)-2*(FA+1)

Depending on the number of rows of AA the number of rows of the last sub array will range from a value equal to that of the intermediate ones or smaller.
Therefore the number of rows of the intermediate sub arrays is a direct indication whether the limit of 500 restrictions imposed by Matlab is exceeded or not
For every unit increase of the value of FA the maximum number of rows of aa_k is increasing by 4*M-2. Change of the value of FA can be made by the user inside this module
Figure 10.3.8 indicates the shape of each of the sub arrays aa_k entering "linprog" in attempting progressive, longitudinally, 3d fairing of each segment.
The arrows indicate the span of blocks of rows of each sub array but every one of them has NO_CO columns.
The bar at the right hand side of this figure shows the region of stations where second differences exist.



F ig 10.3.8
Shape of aa_k sub arrays for FA=3

After loading the necessary information this module proceeds to optimization using ,**repeatedly,** Matlab "linprog" custom function for Large size problems. If desired, the user can change parameters such as using Medium size algorithm and/or different tolerances (TolFun,TolX). **In some cases that may be found useful to achieve**
convergence.
In case a plate development is desired by another module named "test_test_test8_mad1.m" (initially specified ISENS value is 0 or 1) custom

module “arc_lengths.m” is called to produce piecewise expansions of stations and waterlines .
Plots of portions of body plan and waterlines plan appear for inspection and other necessary for further use information, is saved in directory c:\NAVAL\fair.bak8\TEMP.

9.- CALLS Matlab module “linprog” and custom module “arc_lengths”.

10.3.9 MATLAB Process “arc_lengths.m ”

1.-AUTHOR - Ioannis Kolliniatis
2.-TYPE – Specific
3.-SHORT DESCRIPTION -This module is called by Matlab Processes “breadths8_fair8_correct_1” and “breadths8_fair8_correct8ggg” (descriptions are given in paragraphs 10.3.7 and 10.3.8) depending on the number of restrictions of the linear problem (500 or grader than 500).
4.-LANGUAGE – MATLAB
5.-FLOW CHART - As shown in PLATE (9) of Appendix A.
6.-MATLAB LISTING - See ANNEX (9) of Appendix A.
7.-INPUT VARIABLES - Those loaded.
8.-LONG DESCRIPTION –After loading the necessary information this module proceeds to the step by step computation of the first derivatives y_x and y_z and the integration of functions $[1+(y_x)^2]^{0.5}$ and $[1+(y_z)^2]^{0.5}$ to find arc lengths of lines lying between stations and between waterlines.
In addition expanded angles “RAM_ARC1” are saved as a mat file in c:\matlabr12\bin\win32 and exported also to FORTRAN in file “c:\naval\fair.bak8\dddxz.dat”
9.- CALLS custom module “rs_compute.m”.

10.3.10 MATLAB Process "test_test_test8_mad1.m "

1.-AUTHOR - Ioannis Kolliniatis
2.-TYPE – Specific
3.-SHORT DESCRIPTION -This module can be called independently, provided that custom Matlab module "FAIR8"(see 10.3.6) has already run with ISENS value of 1 or 0. Given a region of a view side enclosed by straight lines **"limiting view straight lines"**, the module will prepare and show its developed view and it will generate acceptable to an automatic cutting machine information (in the examples of this work for ESAB BU/DB P1401/1992) for cutting the developed plate and for marking on it developed contours of frames.
4.-LANGUAGE – MATLAB
5.-FLOW CHART - As shown in PLATE (10) of Appendix A.
6.-MATLAB LISTING - See ANNEX (10) of Appendix A.
7.-INPUT VARIABLES - Those loaded.
8.-LONG DESCRIPTION –After loading the necessary information this module proceeds to the step by step computation of the first derivatives y_x and y_z and the integration of functions $[1+(y_x)^2]^{0.5}$ and $[1+(y_z)^2]^{0.5}$ to find expanded arc lengths of successive projected rectangular elements (**view elementary rectangles)** contained between frames and waterlines of the view side area
First custom module "load_data8.m" is called to produce the above mentioned expanded lengths of view waterlines and view stations parts lying between successive stations and successive waterlines (**grid view points)** of view elementary rectangles. These elementary lengths are denoted as "xloc1" and "zloc1" respectively.
Corner developed angles of view elementary rectangles, denoted as "dddxz", produced by "mat3d400f_1_new_mad" (for description see paragraph 10.3.12) are imported.
The view elementary rectangles are combined vertically to form tiers and the tiers are combined horizontally to produce information for the entire developed area.
The perimeter points of the developed plate are collected and arranged in a counter wise arrangement to allow progressive simulation of the movement of the cutting burner.
Information for producing cutting and marking codes (in this work for ESAB BU/DB P1401/1992) by Fortran program "mat3d400f_1_new_mad" is exported to the following Fortran files:
'RE_TAB_X.dat'
'IM_TAB_X.dat'
'RE_TAB_X_MO.dat'
'IM_TAB_X_MO.dat'
'SI_I_SS.dat'
'SI_J_SS.dat'
'SI_I_MO.dat'
'SI_J_MO.dat'
'ID_FOR.dat'
'TOT_X_PLOT.dat'
'TOT_Z_PLOT.dat'
'SI_PLOT.dat'

Generation of such files for other cutting machines is easy by modifying slightly this module, so that the produced code files will comply with their requirements.9-CALLS custom module “load_data8.m”.

10.3.11 MATLAB Process “load_data8.m ”

1.-AUTHOR - Ioannis Kolliniatis
2.-TYPE – Specific
3.-SHORT DESCRIPTION -This module is called by “test_test_test8_mad1” (description was given in paragraph 10.3.10) . For the specified view sight enclosed by straight lines (denoted as “**limiting view straight lines**”) . It will import parameters “xloc1” and “zloc1” (their physical meaning has been explained in paragraph 10.3.10).
4.-LANGUAGE – MATLAB
5.-FLOW CHART - As shown in PLATE (11) of Appendix A.
6.-MATLAB LISTING - See ANNEX (11) of Appendix A.
7.-INPUT VARIABLES - Those loaded.
8.-LONG DESCRIPTION –After loading the necessary information this module will import expanded lengths of view waterlines and view stations parts lying between successive stations and successive waterlines (**grid view points** of **view elementary rectangles**).
Two basic arrays namely “all_pts” and “per_pts” are formed containing information as follows:
“all_pts” of size equal to [size(LLENGTHS)*size(HHEIGHTS),5] containing serial no of station, serial no of waterline ,position of station, position of waterline, and expanded angles (dddxz).
“per_pts ” of size [no of perimeter points ,7] containing serial no of station, serial no of waterline ,position of station, position of waterline, serial no of side of the view polygon , expanded angle at position and a column with null values.
9- CALLS None

10.3.12 FORTRAN program "mat3d400f_1_new_mad.for"

1. AUTHOR - Ioannis Kolliniatis
2. TYPE – Specific
3. SHORT DESCRIPTION– This program can be called independently provided that program "checkpts_no_debug_mad" has run. Supplied with data produced by the above FORTRAN program it prepares files useful for several plots and for use by other Matab modules.
4. LANGUAGE – Fortran
5. FLOW CHART - As shown in PLATE 12 of Appendix A
6. FORTRAN LISTING - See ANNEX 12 of Appendix A
7. INPUT VARIABLES Those contained in file "mat3d1.dat".
8. LONG DESCRIPTION

This program reads data from the files listed in the following table containing their names and their numbers. The majority of them are produced by Fortran program " checkpts_no_debug_mad", while few of them have been exported by custom Matlab modules.

ISENS.dat File NO (552)	HHEIGHTS.dat File NO (1023)	LENGTHS.dat File NO (801)	NO_LE_VERT.dat File NO (1012)
Error.dat File NO (905)	SI_XX_ALL.dat File NO (773)	HEIGHTS.dat File NO (802)	NNOKKO.dat File NO (1024)
Mat3d1.dat File NO (1)	XX_ALL.dat File NO (771)	Ddxx.dat File NO (9502)	
Mat3d421.dat File NO (421) Submitted.dat File NO (300)	N_SI_LE.dat File File NO (701) M_SI_HE.dat File File NO (702)	Ddzz.dat File NO (9602) Offsets1.dat File NO (8003)	
Submitted1.dat File NO (3300)	KEEP.dat File NO (853)	SI_ROWS_VERT.dat File NO 1013)	
NN_SI_LE.dat File NO (1021)	Points.dat File NO (601)	SI_COLS_VERT.dat File NO (1014)	
MM_SI_HE.dat File NO (1020)	LE_KEEP.dat File NO (851)	LE_NO_VERT.dat File NO (1010)	
LLENGTHS.dat File File NO (1022)	HE_KEEP.dat File File NO (652)	HE_NO_VERT.dat File NO(1011)	

Using the information mentioned above the program recalculates offsets, x and z 1st derivatives with a different method for comparison reasons.
Then Matlab files "myplot.m", "myplot1.m" and "myplot3d.m" useful for plotting body plan, waterlines plan and a 3d plot lines respectively, are prepared.
Finally Fortran file "vert.dat" is prepared necessary as basic input to Matlab process "test_test_test8_mad1".

10.-CALLS. Fortran Subroutine "int.for"

10.3.13 MATLAB Process "models18_7.m "

1. AUTHOR - Ioannis Kolliniatis
2. TYPE – Specific
3. SHORT DESCRIPTION - This program can be called independently provided that FORTRAN program " checkpts_no_debug_mad" has run. Supplied with data produced by the above program prepares information useful for making codes acceptable by any automatic cutting machine (in this work specifically for ESAB) to cut models of frames from thin plates.
4. LANGUAGE – Matlab
5. FLOW CHART - As shown in PLATE 13 of Appendix A
6. MATLAB LISTING - See ANNEX 13 of Appendix A
7. INPUT VARIABLES Those loaded
8. LONG DESCRIPTION

For material economy (minimum scrap) and for other practical reasons (minimum weight) the models of frames should be hollow by removing a triangular part from their core .In addition they should not have full breadth as that corresponding to offsets measured from centerline. Parameter BRA specifies the minimum allowed bottom breadth of the most narrow frame model .Parameter DELTA specifies the distance between nested frame models in x, y directions while parameter OFF defines the beginning of coordinates for the nesting process.

First the dimensions of the plate on the surface of which nesting will take place are examined. In case either the length of the longest model exceeds the length of the plate on which nesting will take place (PLATE_X) or the breadth of the plate (PLATE_Y) is not sufficient for the requirements of the nesting scheme the program stops. A diagnostic appears suggesting a change to the nesting inputs (change dimensions of plate or of number of frame models to be nested and cut from the available plate).

After counting the number of frame models that can during nesting be placed on every zone along the x axis, trajectories of successive movements of the marking tool and of the cutting burner are computed.

All necessary information for generating appropriate code acceptable by a specific cutting machine is exported. Fortran Program **"mat3d400f_1_tape.f**" will do this job for ESAB BU/DB P1201 (see paragraph 10.3.14).

9.- CALLS None

10.3.14 FORTRAN program "mat3d400f_1_tape.f"

1. AUTHOR - Ioannis Kolliniatis
2. TYPE – Specific
3. SHORT DESCRIPTION - Supplied with data produced by Matlab custom module "**models18_7**" this module prepares the necessary code ,approved by cutting machine ESAB BU/DB P1201 , for nesting and cutting models of frames.
4. LANGUAGE – Matlab
5. FLOW CHART – Not necessary
6. MATLAB LISTING - See ANNEX 13 of Appendix A
7. INPUT VARIABLES Those loaded
8. LONG DESCRIPTION File "tape.mpg" is prepared (with information produced and exported by Matlab custom module "models18_7*")* This file is an example of code for cutting models of frames corresponding to the development of a view sight area of the ship hull from a flat thin plate. The example refers to a given automatic cutting machine, in this case ESAB.

The following table relates Fortran variable names with corresponding ESAB codes and their action and is helpful in understanding the contents of file 'tape.mpg'.

VARIABLE	CODE	MEANING
necbeg	5	fast mode on
necend	6	fast mode off
markbeg	9	compressed air marking on
markend	10	compressed air marking off
marksta1	38	cutting groove compensation off
marksta2	45	automatic height adjustment on
marksta3	115	
markstop	12	marking tool offset off
istart	30	cutting groove offset right on
istart1	29	cutting groove offset left on
ibegcut	53	cutting cycle plasma burner on
iendcut	54	cutting cycle plasma burner off
iendflame	38	cutting groove
istop	63	end of program

STUDY CASE CONE-1
1.-DESCRIPTION

In order to examine the reliability of the methods and of the soft wear developed in this work a simple case of an inversed cone was selected as a first study case. An additional reason for this selection is that the lateral area of a cone is a developable surface and therefore comparisons with analytically computed information are possible.
Table 1 below contains the basic input file ("mat3d.dat"). Data have been given at 7 stations and at 10 waterlines while results have been specified for the same stations and waterlines. Developed area is requested for a view rectangular plate spanning 4 frames and 4 waterlines. An ISENS value of 1 has been given indicating that a plate development as well as frames models information should be produced. This implies also that a 2d plus a 3d fairing will be necessary.
Input offsets have been calculated analytically

```
07.000  10.0000   07.0000   10.0000  1.0000  01.0000   00.8000 0.1000  0 0 | 03.2900     1   0.6
           00.0000     01.0000     02.0000     03.0000     04.0000     05.0000     06.0000
09.2750  +  06.8785  +  06.8054  +  06.5813  +  06.1898  +  05.5959  +  04.7237  +  03.3636
11.2750  +  08.3617  +  08.3017  +  08.1190  +  07.8050  +  07.3429  +  06.7021  +  05.8240
13.2750  +  09.8450  +  09.7940  +  09.6397  +  09.3767  +  08.9957  +  08.4808  +  07.8053
15.2750  +  11.3282  +  11.2840  +  11.1503  +  10.9237  +  10.5985  +  10.1651  +  09.6088
17.2750  +  12.8114  +  12.7723  +  12.6544  +  12.4552  +  12.1710  +  11.7955  +  11.3196
19.2750  +  14.2947  +  14.2597  +  14.1541  +  13.9763  +  13.7236  +  13.3917  +  12.9745
21.2750  +  15.7779  +  15.7462  +  15.6506  +  15.4901  +  15.2625  +  14.9647  +  14.5926
23.2750  +  17.2612  +  17.2322  +  17.1449  +  16.9985  +  16.7913  +  16.5211  +  16.1848
23.3750  +  17.3353  +  17.3064  +  17.2196  +  17.0738  +  16.8675  +  16.5986  +  16.2639
24.2750  +  18.0028  +  17.9750  +  17.8913  +  17.7511  +  17.5528  +  17.2945  +  16.9735
   0.0  01.0000      02.0000                 03.0000          04.0000   5.    6.
      9.275  11.275  13.275  15.275   17.275 19.275 21.275   23.275 23.375 24.275
4
01.000          13.2750         01.0000         19.2750
01.0000         19.2750         04.0000         19.2750
04.0000         19.2750         04.0000         13.2750
04.0000         13.2750         01.0000         13.2750
```

Table 1
Basic input file "mat3d.dat"

Figure 1 shows the corrected and faired 2d and 3d body plan together with the submitted data.

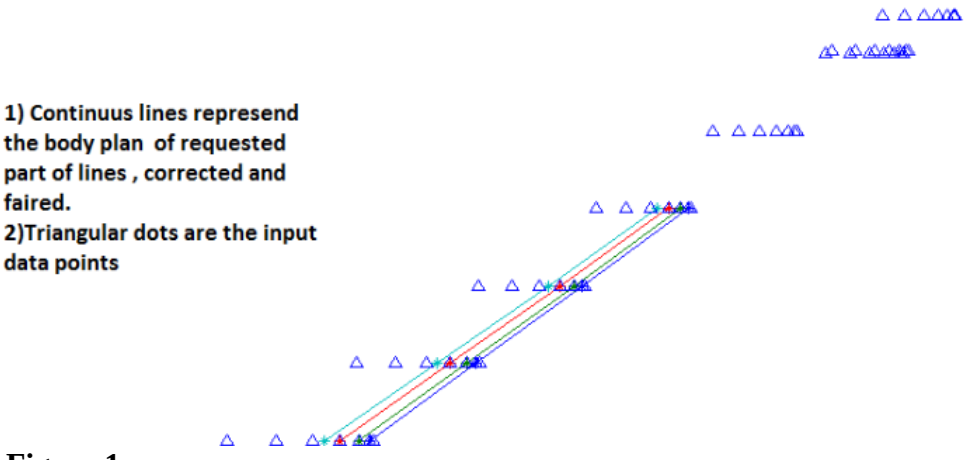


Figure 1
Corrected and faired body plan

Figure 2 shows the corrected and faired 2d and 3d waterlines plan together with the submitted data.

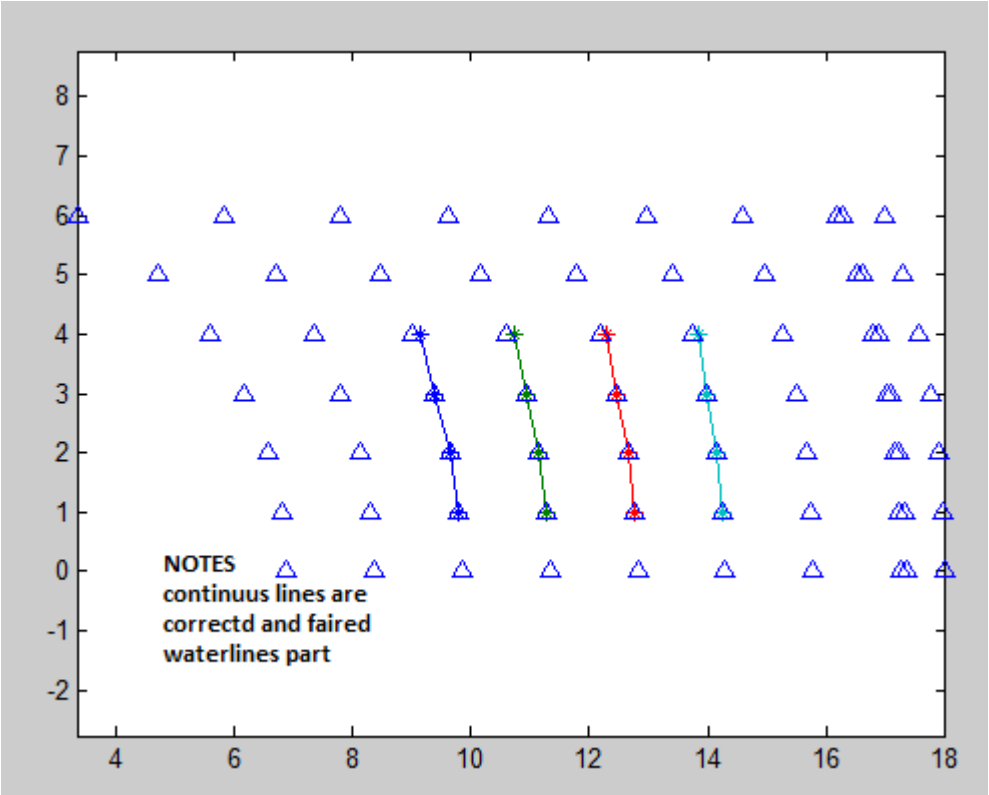


Fig 2 Corrected and faired waterlines plan

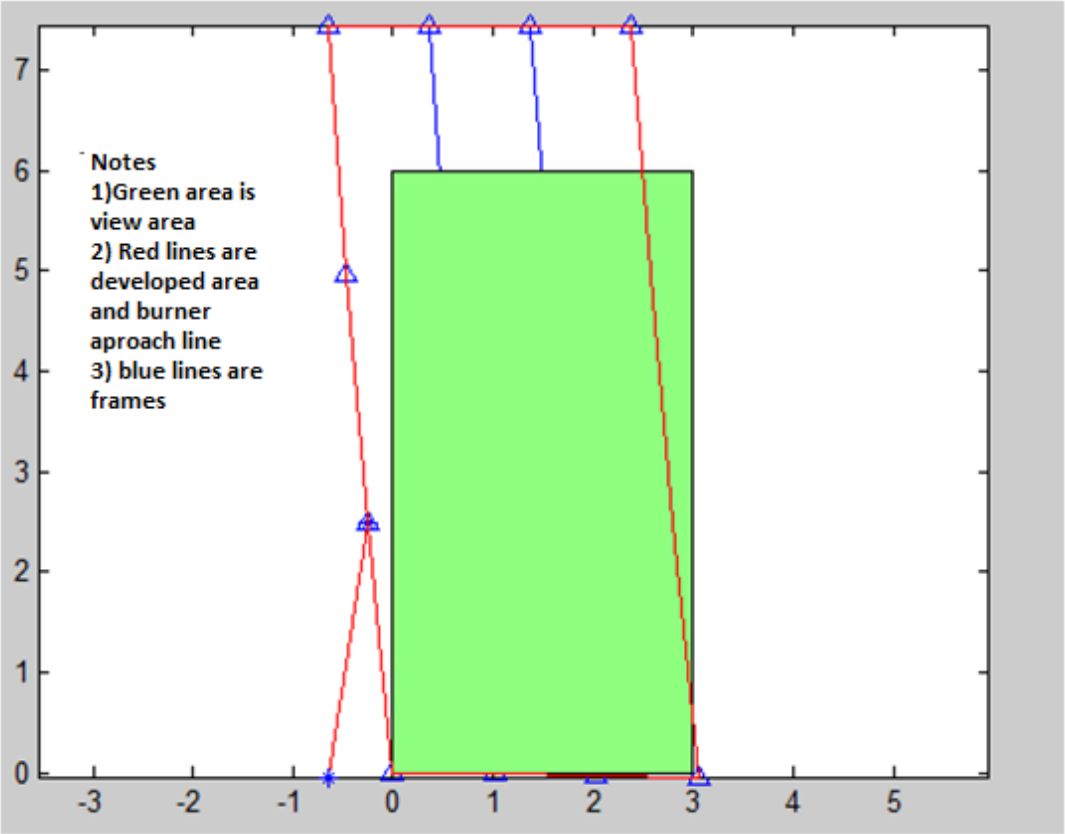


Fig 3 View area and its development

CONCLUSIONS

This simple artificial case supplied by input data already correct and faired, in the naval architects sense, is a first proof of the ability of developed soft wear to handle fairing of ships lines by linear programming.
More conclusions will be apparent by the examination of following more realistic study cases.

STUDY CASE CONE-2

1.-DESCRIPTION

In this study case intentional errors have been inserted in the data of station 1 in such a way that unintentional inflection points would appear. Data in all other stations have remained unchanged as given in study case CONE-1.
Table 1 below contains the basic input file ("mat3d.dat"). Data have been given at 7 stations and at 9 waterlines while results have been specified for the same stations and waterlines. No developed area has been requested (ISENS value equal to 2). A 2d plus a 3d fairing was specified.

```
07.000  9.0000   07.0000   9.0000  1.0000  01.0000   00.8000 0.1000  0 0   03.2900  2   0.6
          00.0000     01.0000     02.0000     03.0000     04.0000     05.0000     06.0000
09.2750  +  06.8785  +  06.8054  +  06.5813  +  06.1898  +  05.5959  +  04.7237  +  03.3636
11.2750  +  08.2617  +  08.3017  +  08.1190  +  07.8050  +  07.3429  +  06.7021  +  05.8240
13.2750  +  09.9450  +  09.7940  +  09.6397  +  09.3767  +  08.9957  +  08.4808  +  07.8053
15.2750  +  11.2282  +  11.2840  +  11.1503  +  10.9237  +  10.5985  +  10.1651  +  09.6088
17.2750  +  12.9114  +  12.7723  +  12.6544  +  12.4552  +  12.1710  +  11.7955  +  11.3196
19.2750  +  14.1947  +  14.2597  +  14.1541  +  13.9763  +  13.7236  +  13.3917  +  12.9745
21.2750  +  15.8779  +  15.7462  +  15.6506  +  15.4901  +  15.2625  +  14.9647  +  14.5926
23.2750  +  17.1612  +  17.2322  +  17.1449  +  16.9985  +  16.7913  +  16.5211  +  16.1848
24.2750  +  18.0028  +  17.9750  +  17.8913  +  17.7511  +  17.5528  +  17.2945  +  16.9735
   0.  01.0000      02.0000              03.0000         04.0000   5.   6.
      9.275  11.275  13.275  15.275   17.275 19.275 21.275   23.275  24.275
4
01.000          13.2750         01.0000         19.2750
01.0000         19.2750         04.0000         19.2750
04.0000         19.2750         04.0000         13.2750
04.0000         13.2750         01.0000         13.2750
STUDY CASE CONE_2 WRONG DATA IN STATION 1 ISENS=2
```

Table 1
Basic input file "mat3d.dat"

Figure 1 shows the supplied data as well as the corrected and faired results of station 1.

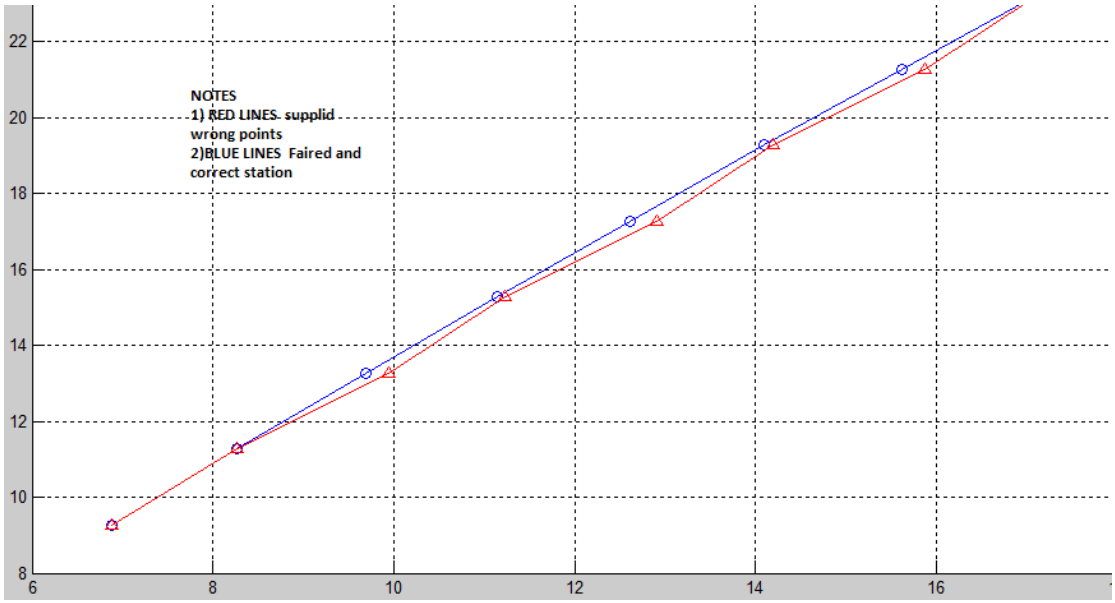


Figure 1
Supplied and corrected data of station 1

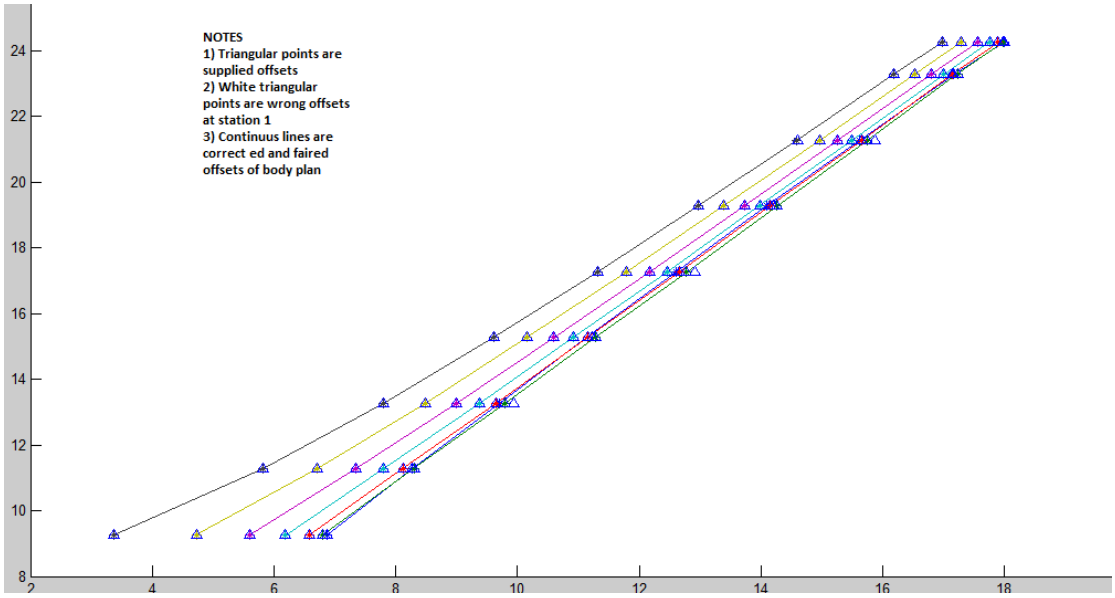


Figure 2
Corrected and faired body plan

Figure 2 shows the corrected and faired 2d and 3d waterlines plan together with the submitted data while figure 3 shows the corrected and faired 2d and 3d body plan.

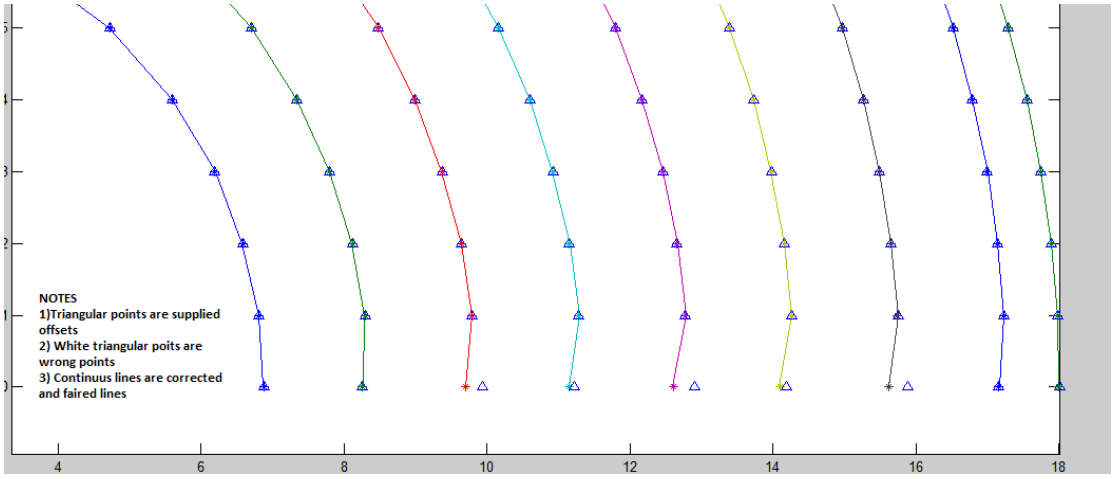


Fig 3.- Corrected and faired waterlines plan

CONCLUSIONS

Total curvatures have been computed by Matlab custom module "curv.m". **Their values are merely zeros since the cone lateral area is developable**
All intentional errors have been corrected and the lines have been faired.

STUDY CASE 1

1._DESCRIPTION

Table 1 below contains random view measurements received by laser for a region located at the starboard side of the bow of Hellenic Navy ship PROMITHEUS ,approximately between frames 118 and 126 with the ship lying, upright on dock.

Long distance (mm) from reference plane	Offsets (mms)	Heights (mm) from reference plane
-2.4500000000000000e+001	6.3428999999999996e+003	1.1950000000000000e+002
-1.7100000000000001e+001	7.2488000000000002e+003	5.3570000000000005e+002
-1.8899999999999999e+001	8.3245000000000000e+003	1.6272000000000000e+003
8.8000000000000007e+000	9.0182000000000007e+003	3.0031999999999998e+003
7.5500000000000000e+002	6.6063999999999996e+003	2.6950000000000000e+002
7.8139999999999998e+002	7.6613999999999996e+003	9.8829999999999995e+002
7.7580000000000007e+002	8.3720000000000000e+003	1.8920999999999999e+003
7.7280000000000007e+002	8.7422999999999993e+003	2.5689000000000001e+003
1.5597000000000000e+003	6.8240000000000000e+003	4.8280000000000001e+002
1.5759000000000001e+003	7.2991000000000004e+003	8.1339999999999998e+002
1.5875000000000000e+003	8.1108000000000002e+003	1.7173000000000000e+003
1.5783000000000000e+003	8.6105000000000000e+003	2.6075000000000000e+003
2.3557999999999997e+003	6.7980000000000000e+003	5.7920000000000005e+002
2.3645999999999999e+003	7.0418999999999996e+003	7.5000000000000000e+002
2.3733000000000002e+003	7.7133999999999996e+003	1.4065000000000000e+003
2.3618000000000002e+003	8.4837999999999993e+003	2.6389000000000001e+003
3.1599000000000001e+003	6.7660000000000000e+003	6.8500000000000000e+002
3.1728999999999996e+003	7.4128000000000002e+003	1.2537000000000000e+003
3.1665999999999999e+003	8.0241999999999998e+003	2.0775000000000000e+003
3.1577000000000003e+003	8.3560000000000000e+003	2.6741999999999998e+003
3.9716999999999998e+003	6.7320000000000000e+003	7.9710000000000002e+002
3.9715000000000000e+003	7.4398000000000002e+003	1.4784000000000001e+003
3.9690999999999999e+003	7.9168000000000002e+003	2.1641999999999998e+003
3.9798000000000002e+003	8.2150000000000000e+003	2.7080999999999999e+003
4.7521000000000004e+003	6.7006000000000004e+003	8.9889999999999998e+002
4.7644000000000005e+003	7.0676000000000004e+003	1.2415000000000000e+003
4.7721000000000004e+003	7.6535000000000000e+003	1.9924000000000001e+003
4.7763000000000002e+003	8.0821999999999998e+003	2.7454000000000001e+003
5.5652999999999993e+003	6.6621999999999998e+003	1.0180000000000000e+003
5.5710000000000000e+003	6.9123999999999996e+003	1.2595999999999999e+003
5.5685000000000000e+003	7.6126000000000004e+003	2.1761999999999998e+003
5.5690000000000000e+003	7.9486000000000004e+003	2.7811999999999998e+003
6.3868000000000002e+003	6.6095000000000000e+003	1.1498000000000000e+003
6.3883999999999996e+003	7.1136999999999998e+003	1.6953000000000000e+003
6.3768999999999996e+003	7.5488999999999996e+003	2.3401999999999998e+003
6.3741000000000004e+003	7.8083999999999996e+003	2.8173000000000002e+003

TABLE 1
Measurements of viewpoints taken by laser

Figure 1 is a plot of the those random viewpoints measured by laser ,while figure 2 is a transverse frames plot at predefined longitudinal places corresponding to the actual positions of the ship transverse frames. Data for this plot (covering the region where output has been requested) were produced by using Matlab process mynew.m using triangulation .

Offsets used for this plot have neither been examined nor corrected yet for inconsistencies or mistakes. They also have not been faired.

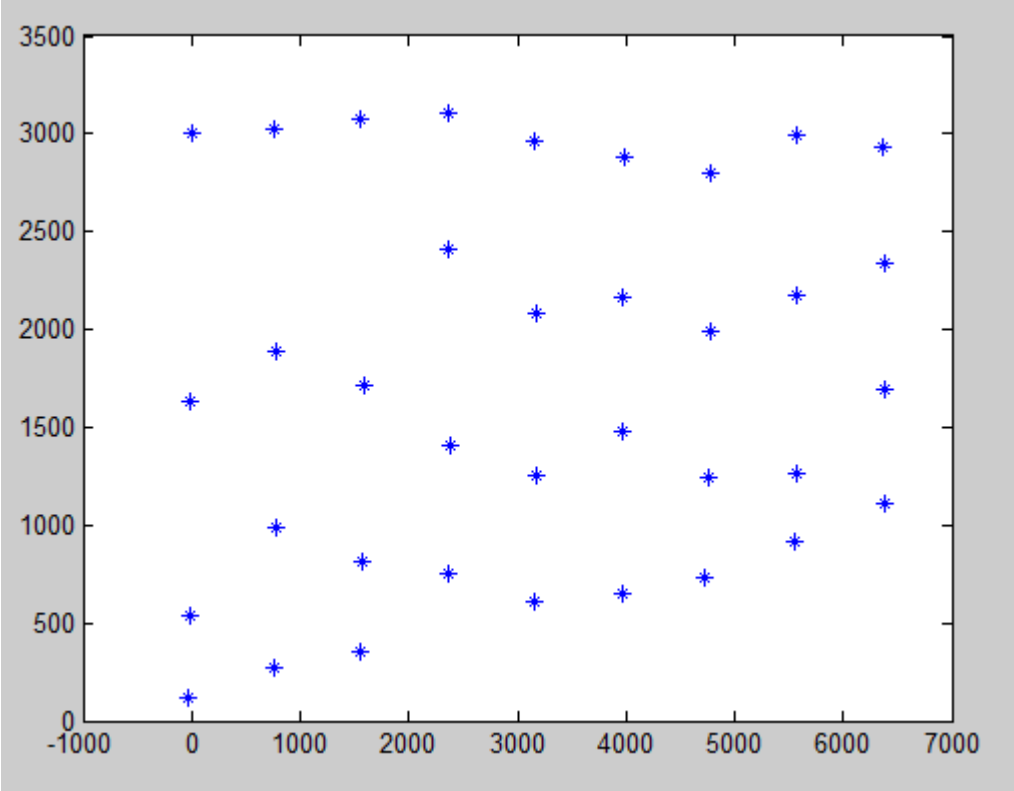


Fig 1
Plot of cloud of view random points at ship's bow

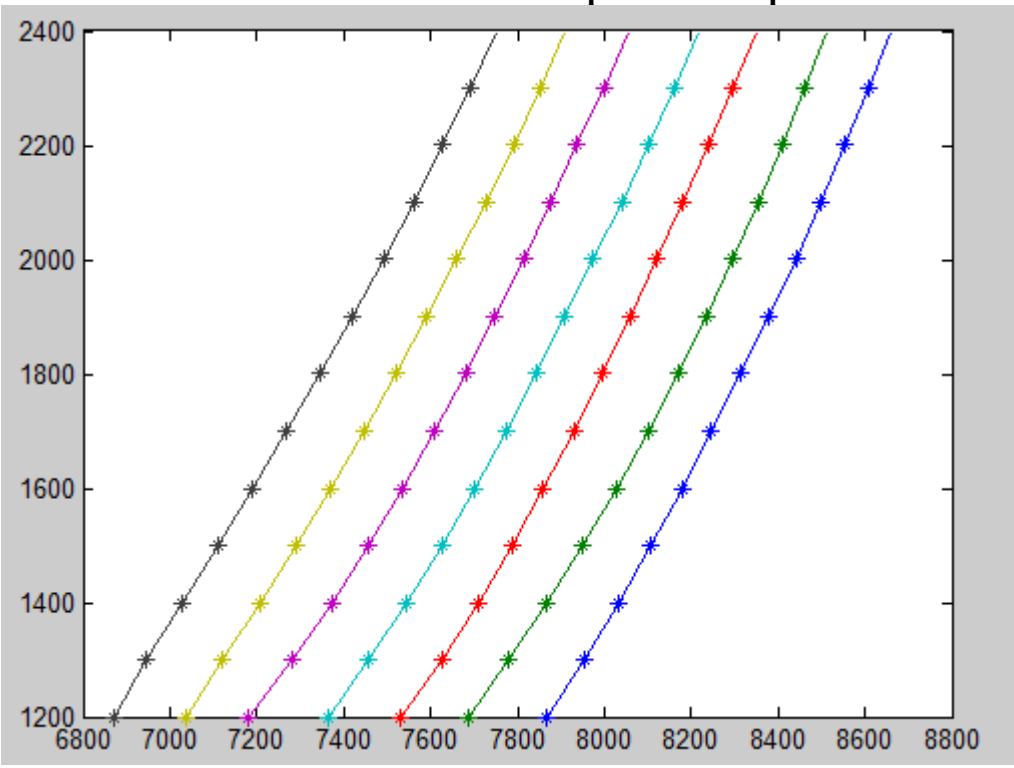


Fig 2

Plot of sections with points received by triangulation

```
07.000   13.0000  07.0000  13.0000  1.0000   10.000   00.8000  00.1000  0 0      03.2900   1   0.6
      00.0768     00.1568     00.2365     00.3162     00.3974     00.4760     00.5565
00.1200  +  00.7868  +  00.7686  +  00.7531  +  00.7363  +  00.7181  +  00.7036  +  00.6873  +  00.6667
00.1300  +  00.7954  +  00.7778  +  00.7625  +  00.7458  +  00.7279  +  00.7121  +  00.6947  +  00.6765
00.1400  +  00.8033  +  00.7865  +  00.7709  +  00.7545  +  00.7372  +  00.7208  +  00.7029  +  00.6859
00.1500  +  00.8108  +  00.7948  +  00.7786  +  00.7626  +  00.7457  +  00.7291  +  00.7110  +  00.6950
00.1600  +  00.8178  +  00.8027  +  00.7859  +  00.7702  +  00.7536  +  00.7370  +  00.7190  +  00.7036
00.1700  +  00.8246  +  00.8102  +  00.7929  +  00.7774  +  00.7610  +  00.7446  +  00.7268  +  00.7117
00.1800  +  00.8312  +  00.8172  +  00.7995  +  00.7842  +  00.7681  +  00.7519  +  00.7345  +  00.7194
00.1900  +  00.8378  +  00.8237  +  00.8059  +  00.7908  +  00.7749  +  00.7591  +  00.7420  +  00.7266
00.2000  +  00.8441  +  00.8297  +  00.8121  +  00.7974  +  00.7814  +  00.7661  +  00.7492  +  00.7335
00.2100  +  00.8499  +  00.8354  +  00.8181  +  00.8039  +  00.7876  +  00.7728  +  00.7562  +  00.7401
00.2200  +  00.8555  +  00.8408  +  00.8239  +  00.8102  +  00.7937  +  00.7791  +  00.7628  +  00.7464
00.2300  +  00.8608  +  00.8460  +  00.8296  +  00.8161  +  00.7998  +  00.7850  +  00.7692  +  00.7525
00.2400  +  00.8658  +  00.8510  +  00.8351  +  00.8217  +  00.8057  +  00.7907  +  00.7752  +  00.7585
00.0768  00.1568  00.2365  00.3162     00.3974  00.4760  00.5565  00.6380
00.1200  00.1300  00.1400  00.1500  00.1600  00.17000 00.1800  00.1900  00.2000  .21   .2200 .23   00.24000
04.0000
00.1568  00.1400  00.1568  00.2200
00.1568  00.2200  00.4760  00.2200
00.4760  00.2200  00.4760  00.1400
00.4760  00.1400  00.1568  00.1400
```

Table 2

Fortran input file to " checkpts_no_debug_mad

Table 2 is a copy of the input file to program "checkpts_no_debug_mad" with IS-ENS=1 implying that development of a view plate as well as views of frames models are required. The view plane specified lies between waterlines 1.40 and 2.20 meters and between stations at 1.568 and 3.476 meters.

This file has been created by an XL process

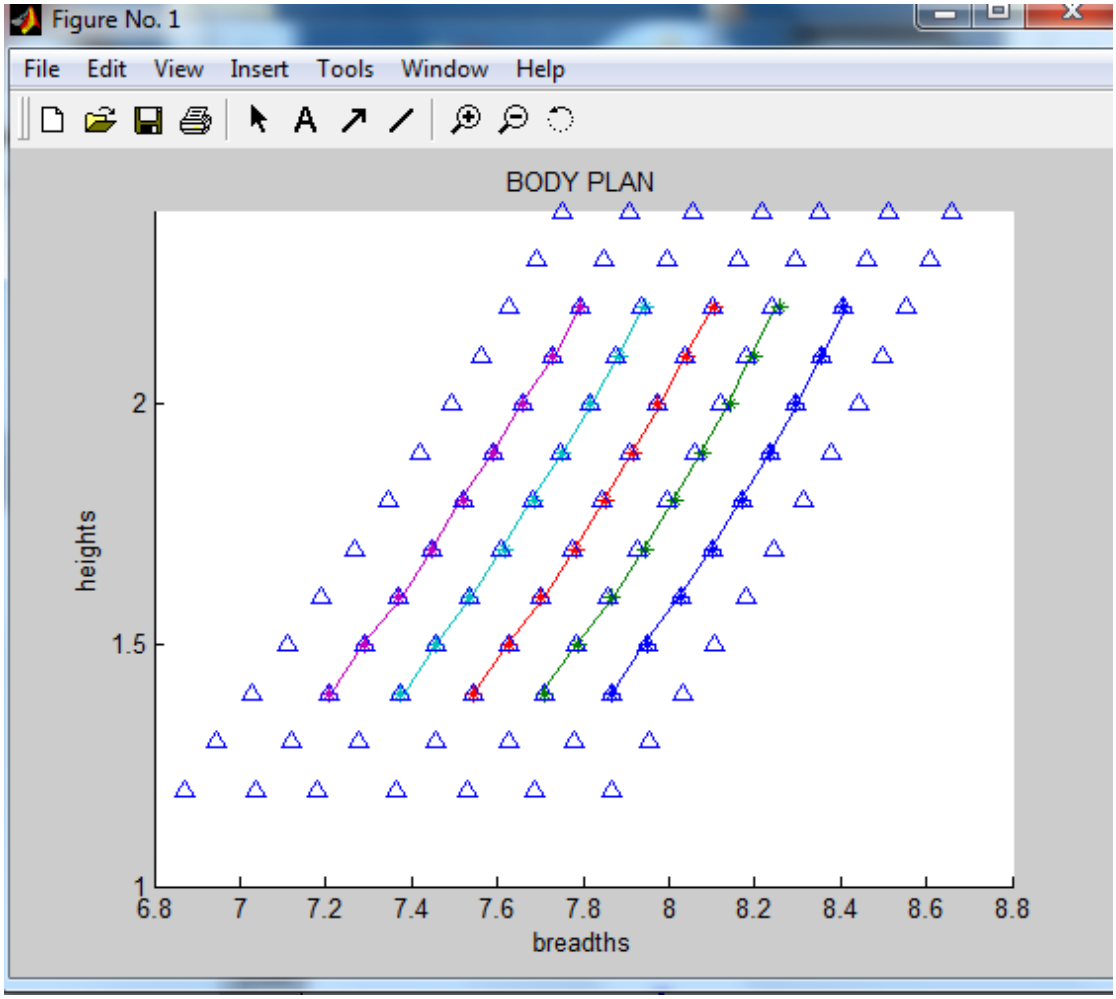


Fig 3
Body plan after fairing

In fig 4 we see a partial body plan based on data as supplied (**red lines**) superimposed by another one drawn with data corrected and faired with 2d+3d fairing process, based on linear programming (**blue lines**) The maximum deviation between points of the two sets of curves is 2.1 cm and it was generated mainly during the 2d points correction and fairing of waterlines (blue lines).

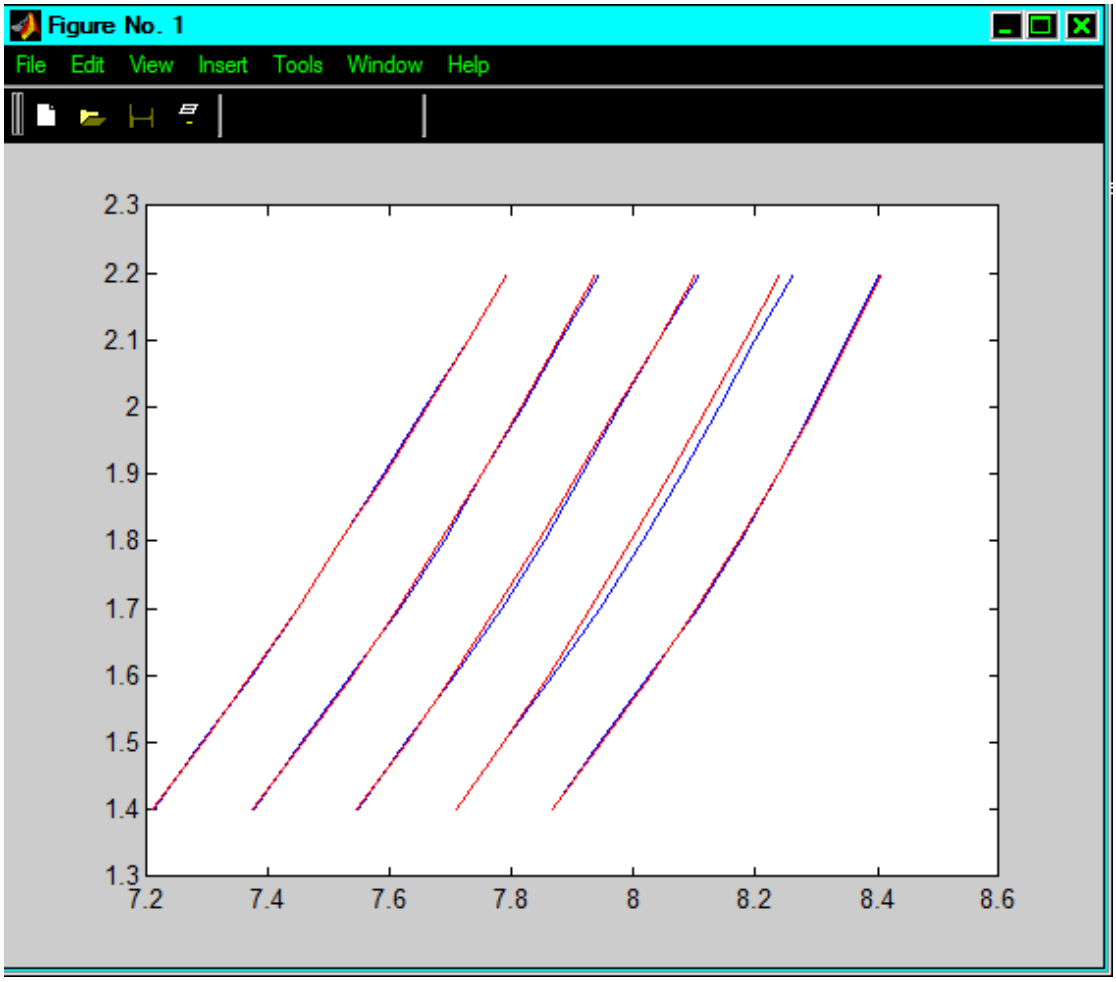


Fig 4
Comparative body plan of view side plate

Fig 5 shows the corresponding partial waterline plans

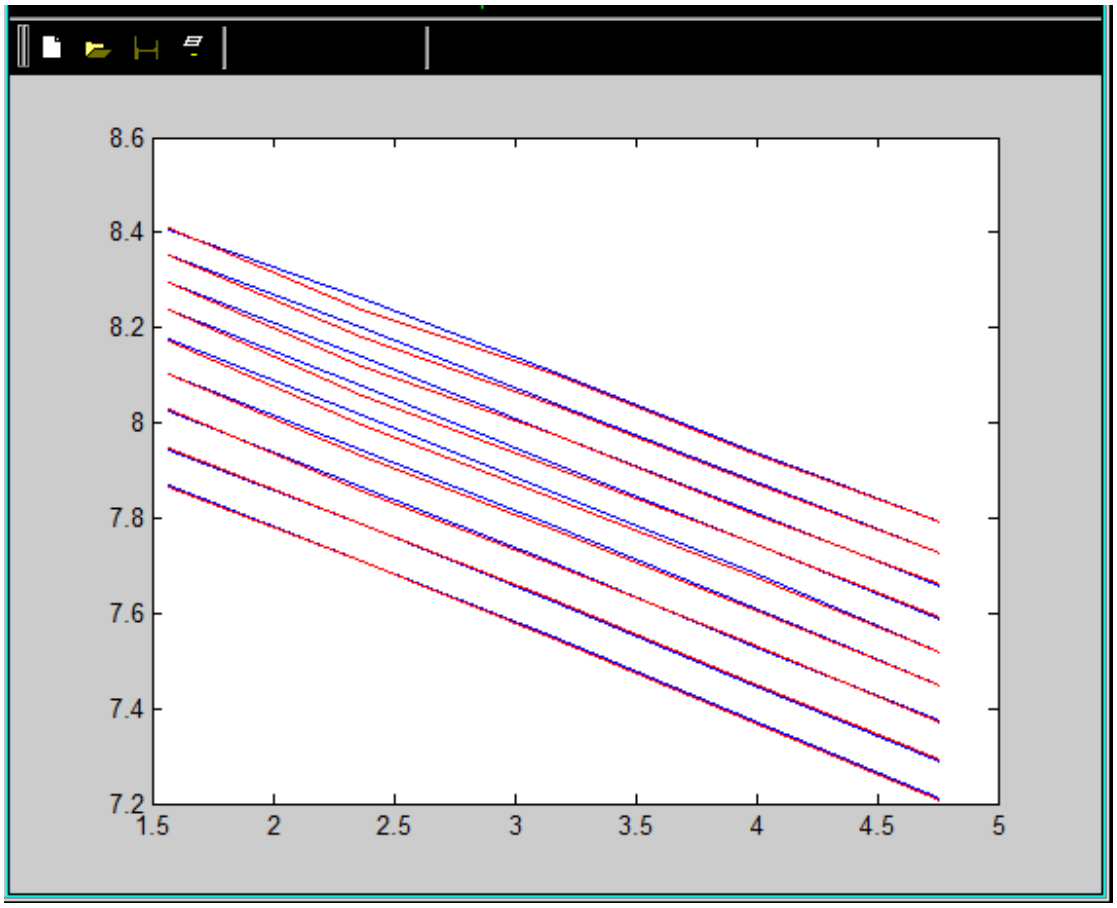


Fig 5
Comparative waterlines plan of view side plate

Figure 6 shows the process of nesting and cutting, models of frames to be used in production. By using appropriate code acceptable by an automatic cutting machine such models could be cut from a thin plate. Big holes are provided in the models to reduce their weight.
Red lines indicate successive movements of the cutting burner while blue lines indicate successive real "plasma" burning.
In this study case nesting is horizontally limited by a specified width of the thin plate of 5 meters.

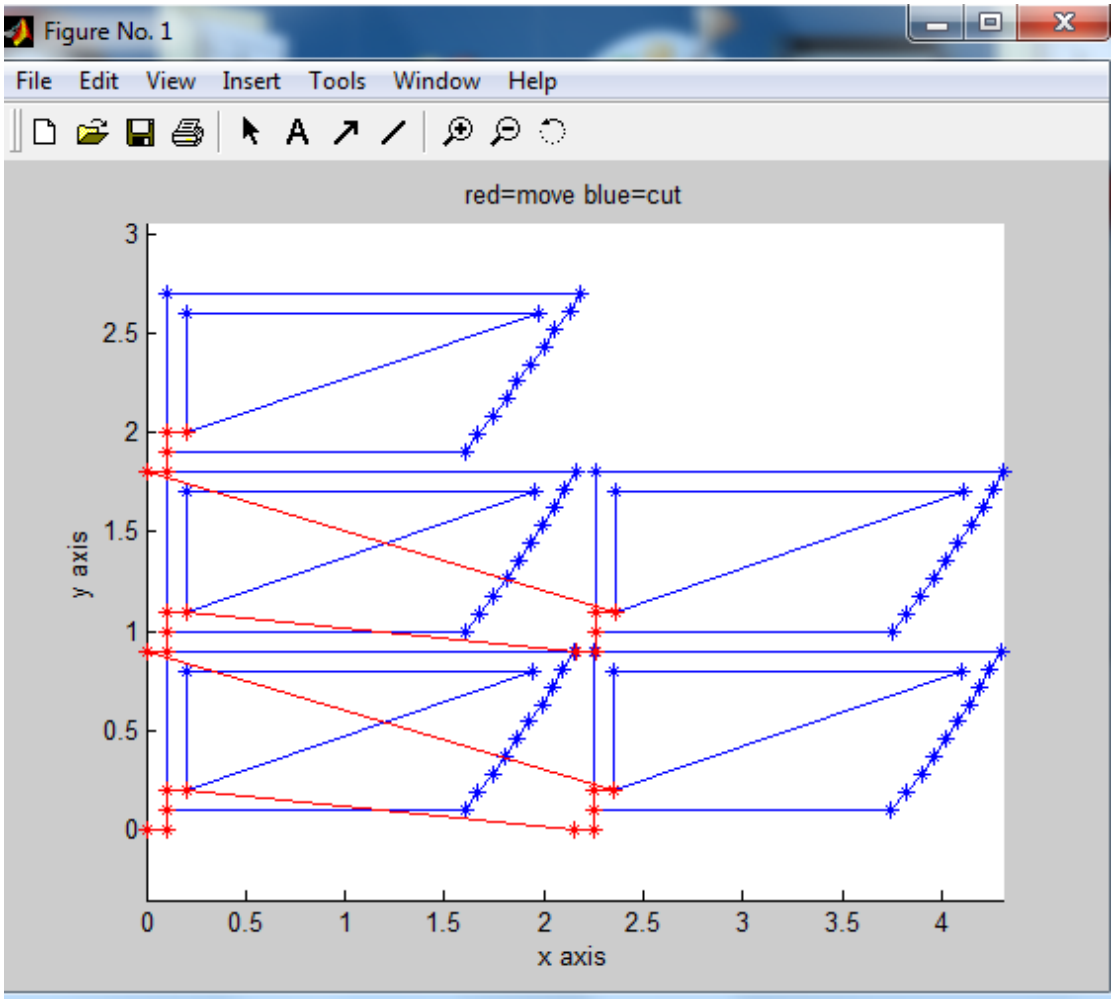


Fig 6
Process of cutting nested frame models

Figure 7 shows the shape of the plate development corresponding to a specified view area.
The green shaded area in figure 6 shows the view area while the blue external lines describe its development .
Needless to say that the developed shape is absolutely correct only and only if the view side region is developable

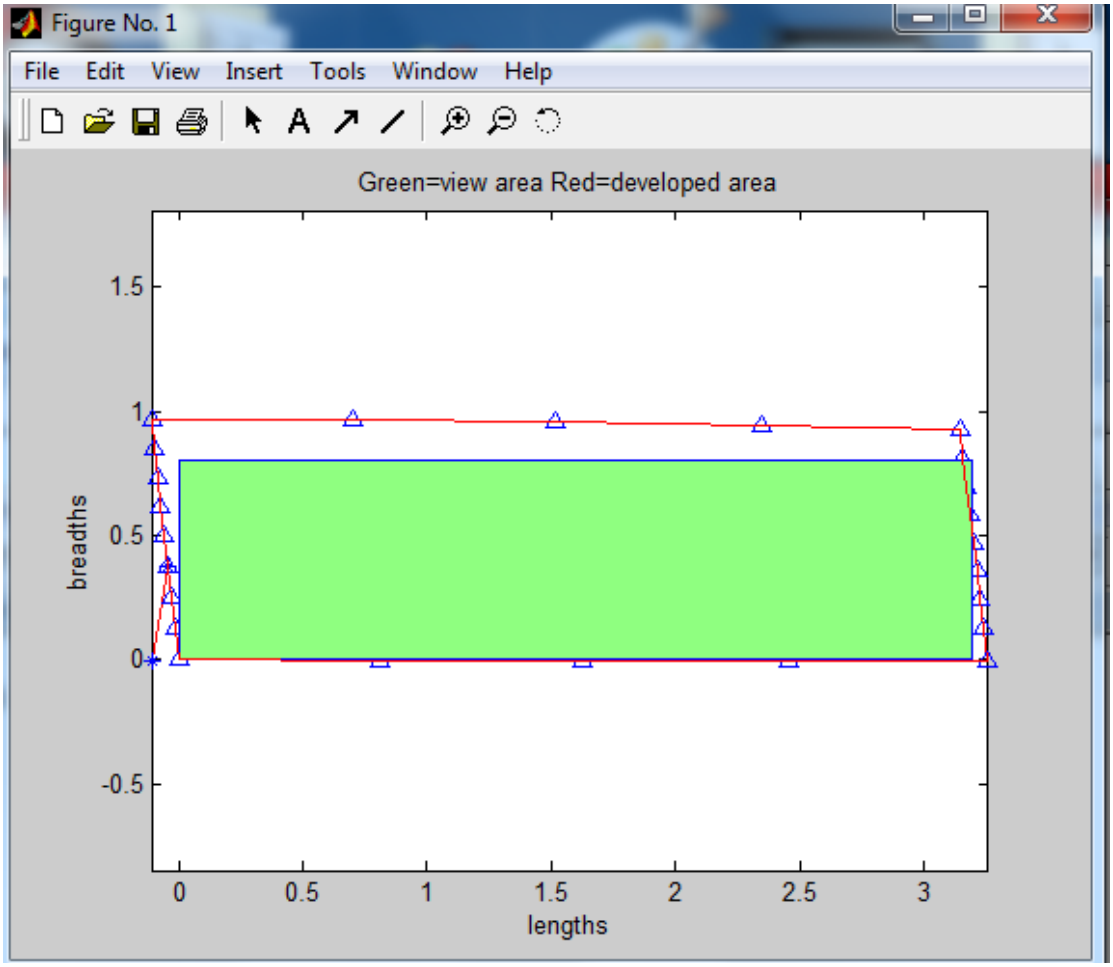


Fig 7
View plate and its development

Fig 8 shows the process of marking the frame contours on the plate. Green lines show marking on the plate of the frame contours by the marking tool of the cutting machine while red lines show the movements of the marking tool from frame to frame

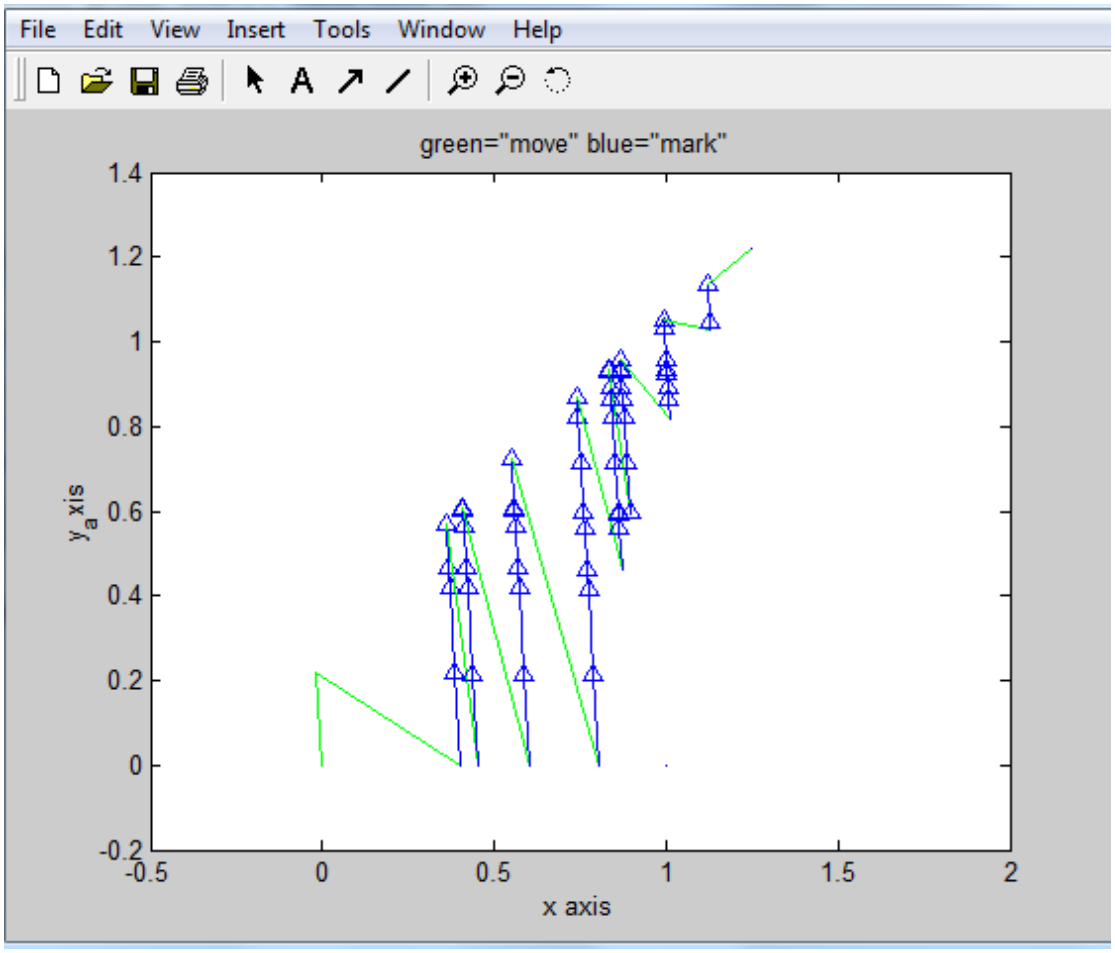


Fig 8
Marking of frames

STUDY CASE 2

1._DESCRIPTION

In this case input data to be used have been taken from ships preliminary lines plan of scale 1:100 (table 1).

For testing reasons offsets have been used at different waterlines for every station . For example at section 1 offsets specified belong to waterlines 0.25 .75 1. 1.5 2. 3. 3.5 while at section 2 at waterlines .25 .5 1. 1.5 2. 2.5 3.5

(see table 1).

Table 1 is a copy of the input file to program "checkpts_no_debug_mad" with ISENS=1 implying that development (expansion) of a view plate as well as views of frame models are required. Figure 1 shows the corresponding body plan. **By just looking at this figure it becomes obvious that inaccuracies and possibly reading errors make those lines non acceptable**

```
07.000  07.0000         10.00   09.00   1.0000  01.0000  00.8000 00.1000   0 0  03.2900 1 0.6
         0.8000      01.6000     02.4000     03.2000     04.0000     04.8000     05.6000
00.2500  +  08.5730  +  08.4590  +  08.3380  +  08.2090  +  08.0740  +  07.9320  +  07.7830
00.5000  +  09.2720  +  08.8720  +  09.0480  +  08.6280  +  08.4960  +  08.6620  +  08.2130
00.7500  +  09.4940  +  09.3880  +  09.2770  +  08.9250  +  09.0330  +  08.9020  +  08.7650
01.0000  +  09.8080  +  09.7110  +  09.6080  +  09.4990  +  09.6310  +  09.2620  +  09.1360
01.2500  +  10.0190  +  09.9310  +  09.8370  +  09.7370  +  09.8110  +  09.5200  +  09.4030
02.0000  +  10.2640  +  10.0840  +  10.1170  +  09.9000  +  09.9450  +  09.7090  +  09.6020
02.5000  +  10.3340  +  10.2720  +  10.2030  +  10.1280  +  10.0400  +  09.9630  +  09.8730
.25 .25 .25  .25 .25 .25 .25
.75 .5  .75  .5  .5  .75 .5
1.   1.   1. .75  1.  1.  1.
1.5  1.5 1.5  1.5  2. 1.5  1.5
2.   2.  2.   2. 2.5  2. 2.
3.   2.5 3.  2.5 3. 2.5  2.5
3.5  3.5 3.5 3.5 3.5 3.5 3.5
0.8 1.2 1.25 1.399 1.6 1.7 1.733 1.866  2. 2.133        2.799  3.049 3.5    3.2 3.399 3.5
0.25 .375 .5  .625 .75 .972 1.  1.194 1.25        1.416  1.861  2.25   3. 3.5       2.083 2.25
05.0000
01.2000  00.3750  00.8000  00.5000
00.8000  00.5000  02.1330  01.2500
02.1330  01.2500  01.7330  00.7500
01.7330  00.7500  01.6000  00.3750
01.6000  00.3750     01.2000  00.3750
```

Table 1

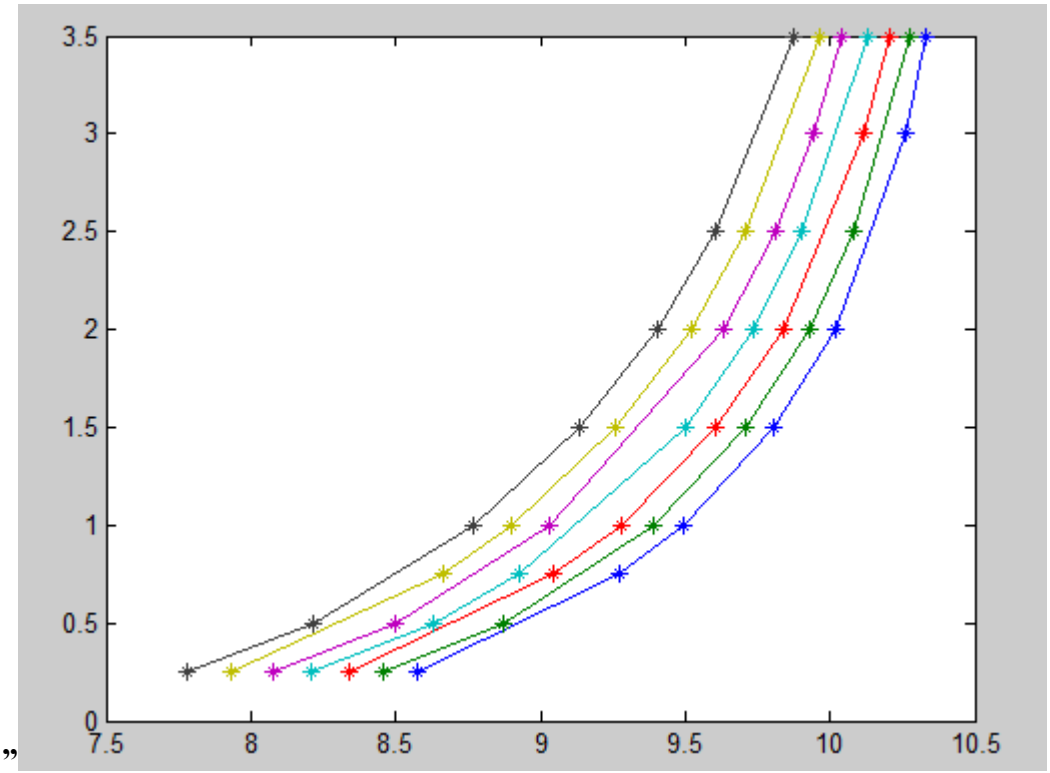


Fig 1

Plot of sections with data received from a 1:100 lines plan

In fig 2 we see a partial body plan based on the supplied data (table 1 above) and covering the region where a plate development of a peculiar view scheme has been requested (stations lying in the region between .375 and 1.25 meters and between waterlines 0.8 and 2.133 meters).

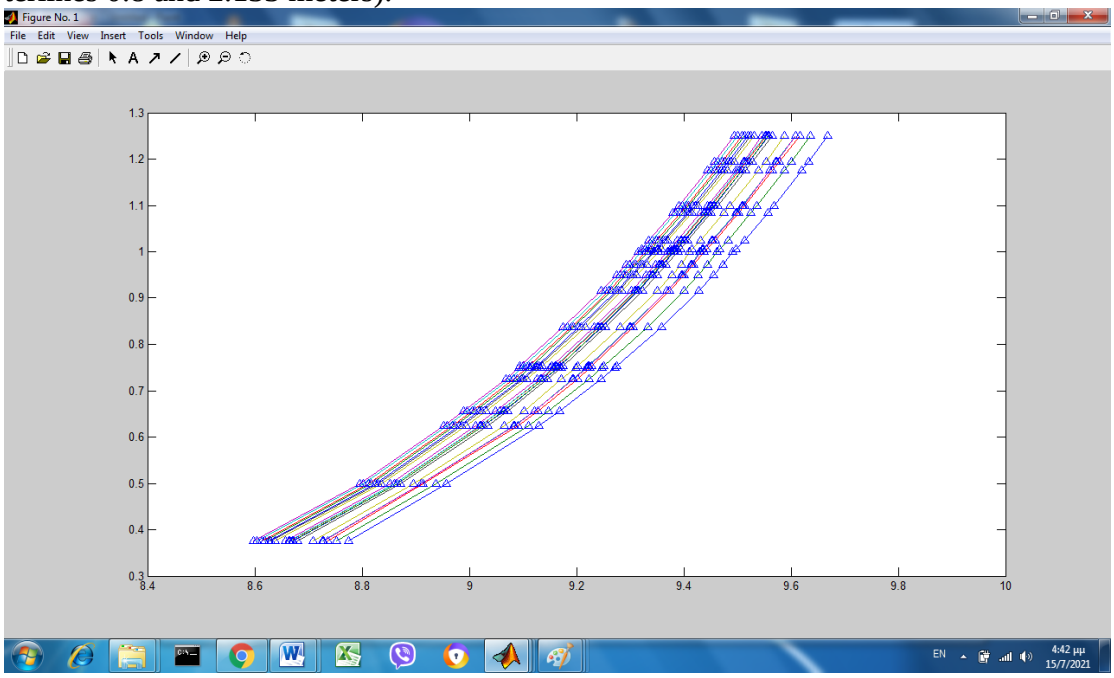


Fig 2

Body plan at region where a plate development was requested

Fig 3 shows the corresponding partial waterline plans of the same region.

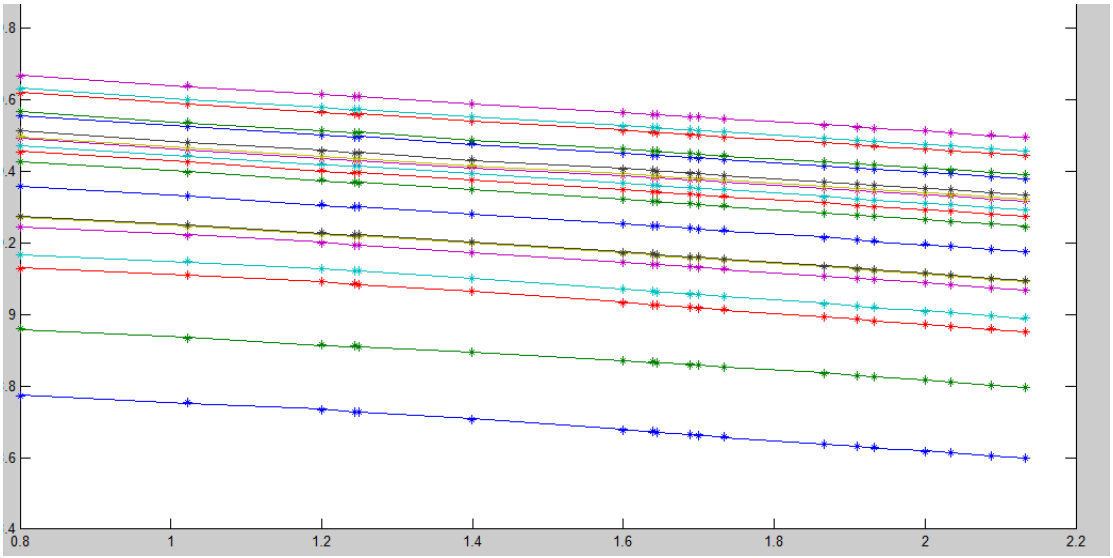


Fig 3

Waterlines plan at region where a plate development was requested

Figure 4 shows the process of nesting and cutting, models of frames to be used in production. By using appropriate code such models could be cut by an ESAP cutting machine from a thin plate. Big holes are provided in the models to reduce their weight. Note that the last two frame models have no holes cuts since their heights are two small.

Red lines indicate successive movements of the cutting burner while blue lines indicate successive real "plasma" burning.

In this study case nesting is lengthwise limited by a specified width of a thin plate of 5 meters.

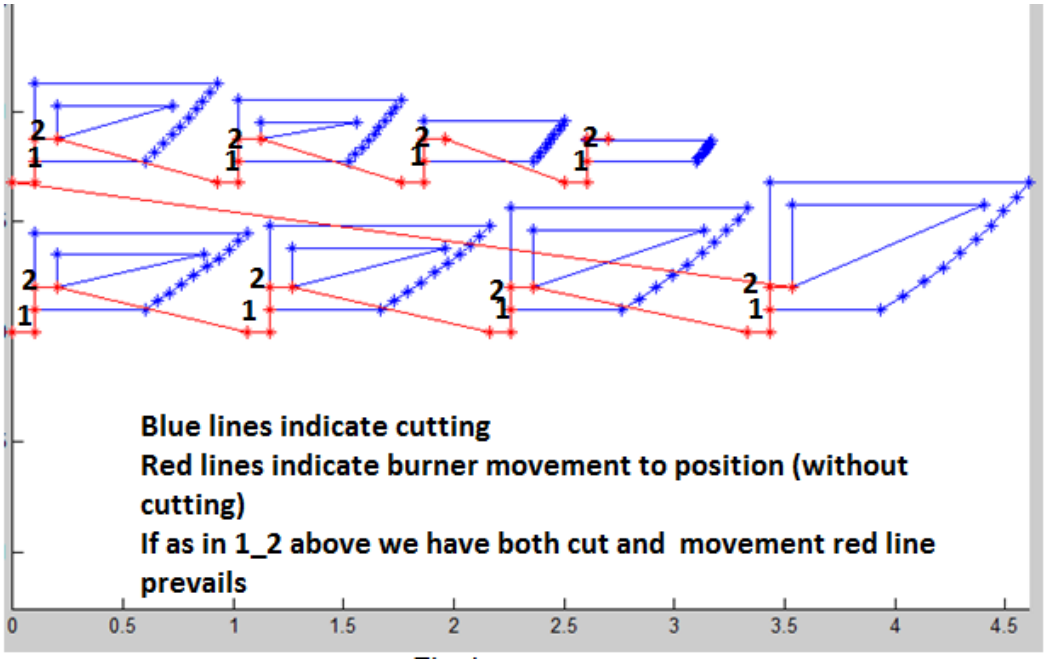


Fig 4

Process of cutting nested frame models

Figure 5 shows the development of a rather peculiar view area

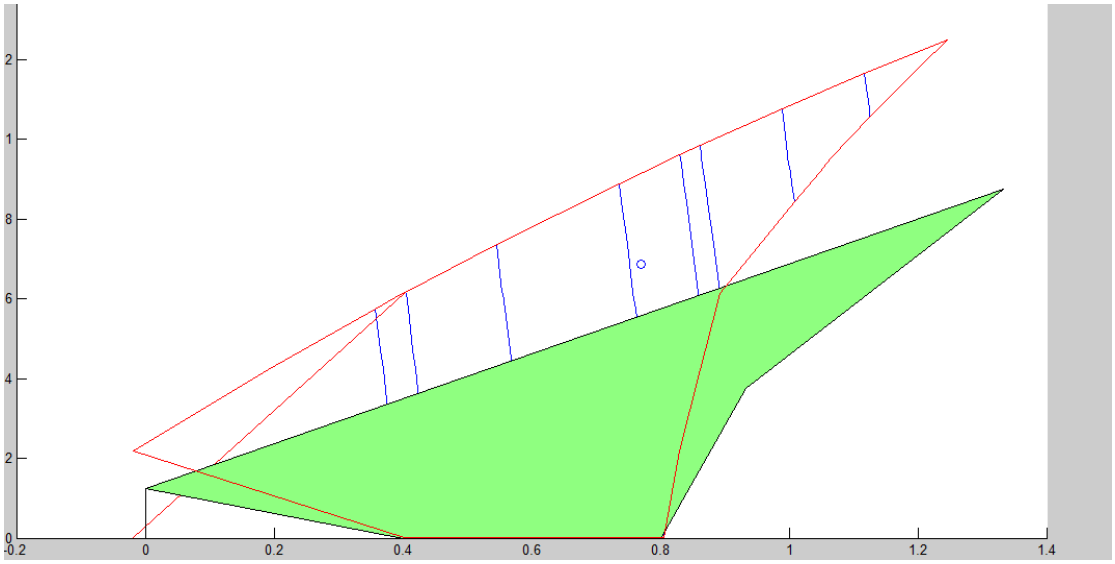


Fig 5
View plate (green) and its development

The green shaded area in figure 5 shows the view plate sight while the red perimeter lines describe the movement of the burner and the following cutting path (counter clockwise). Blue lines are expansions of artificially located transverse frames for testing reasons.
Needless to say that the developed shape is correct only to the degree the view sight region is developable

Fig 6 shows the process of marking the frame contours on the plate. Green lines show marking on the plate of the frame contours by the marking tool of the cutting machine while red lines show the movements of the marking tool from frame to frame

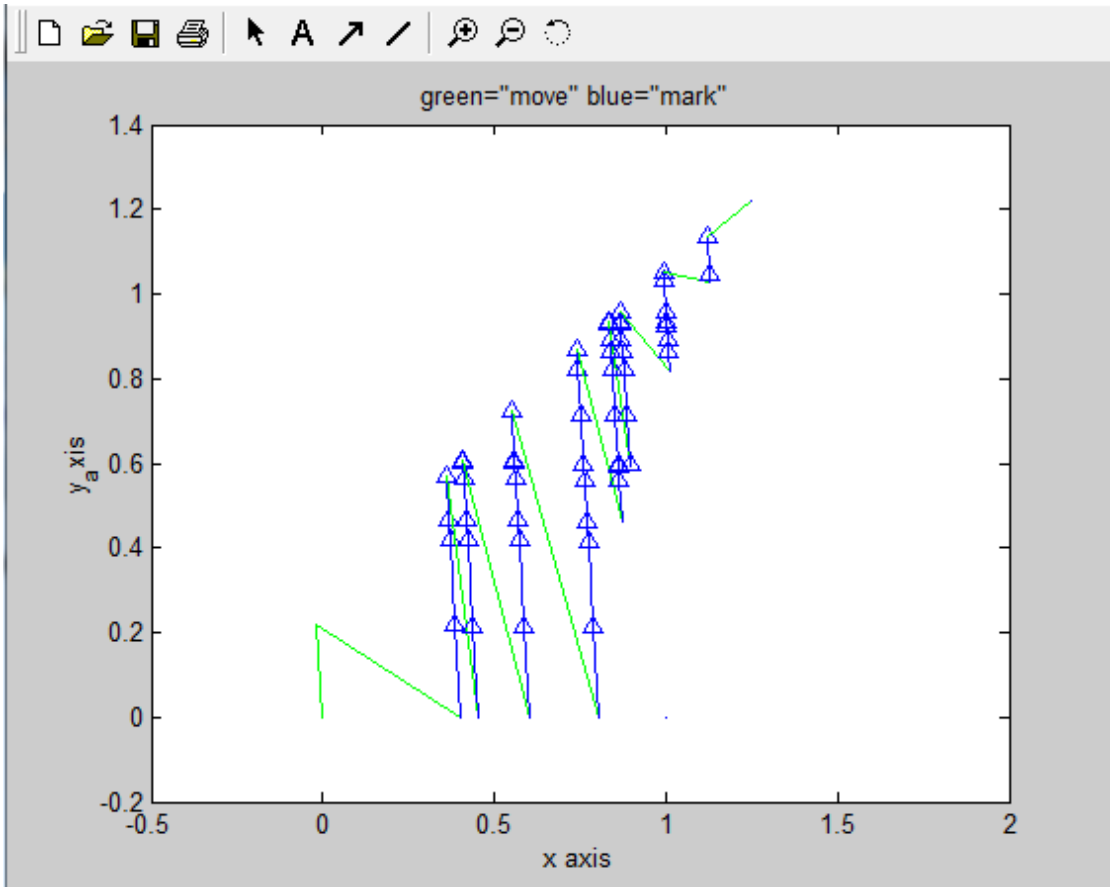


Fig 6
Marking of frames

STUDY CASE 3

1._DESCRIPTION

In this case input data to be used have been taken from a DELTA MARINE ships stern preliminary lines plan of scale 1:100 (table 1).

Table 1 is a copy of the input file to program "checkpts_no_debug_mad" with IS-ENS=1 implying that development (expansion) of a peculiar shape view plate as well as views of frame models are required. Figure 1 shows the corresponding body plan. **By just looking at this figure it becomes obvious that inaccuracies and possibly reading errors make those lines non acceptable**

```
07.0000          08.0000          7.0000   7.0000   1.0000   1.0000   00.8000          00.
      005.110      010.210      015.320      020.420      025.520      030.630      035.7
00.0000  +  00.0000  +  00.5130  +  01.4600  +  02.8470  +  04.6690  +  06.8420  +
00.5000  +  00.6660  +  01.7090  +  02.7560  +  04.6380  +  06.9320  +  08.3730  +
01.0000  +  01.0870  +  02.0470  +  03.1230  +  05.3930  +  07.6580  +  08.7560  +
01.5000  +  01.2570  +  02.1860  +  03.4440  +  06.1310  +  08.1550  +  08.9640  +
02.0000  +  01.2700  +  02.2270  +  03.8060  +  06.8240  +  08.5250  +  09.0710  +
02.5000  +  01.1360  +  02.2450  +  04.3540  +  07.4460  +  08.7940  +  09.1000  +
03.0000  +  00.8980  +  02.3080  +  05.1710  +  08.0090  +  08.9800  +  09.1000  +
03.5000  +  00.6540  +  02.4990  +  06.1960  +  08.5060  +  09.0800  +  09.1000  +
 10.21 11. 1.5  12. 12.5  13.5  14.                                              .511
.5 .6 .75 1. 1.25 1.5 1.75
5
11.00 .5  10.21  1.5
10.21  1.5 14.00  1.75
14.00  1.75 13.5 1.25
13.5  1.25 12.5  .5
12.5 .5  11.00 .5
```

Table 1
Fortran input file to " checkpts_no_debug_mad"

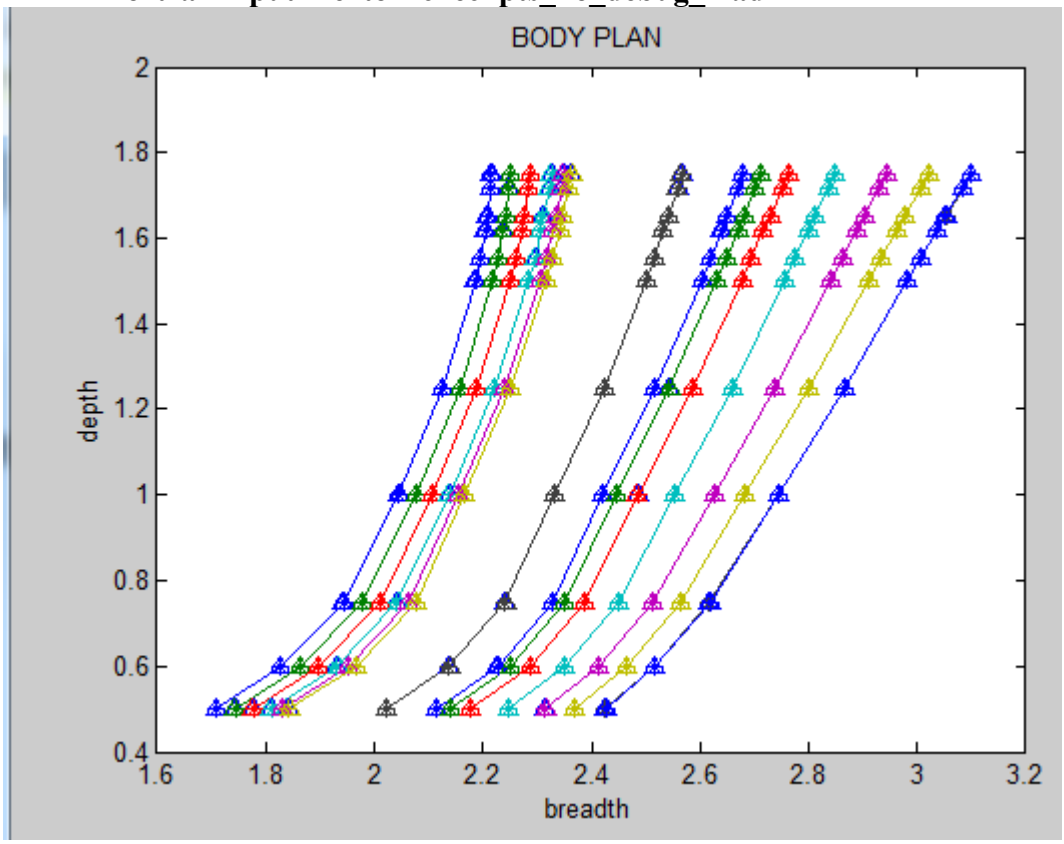


Fig 1
Plot of sections with data received from a 1:100 lines plan

Fig 3 shows the corresponding partial waterline plans of the same region.

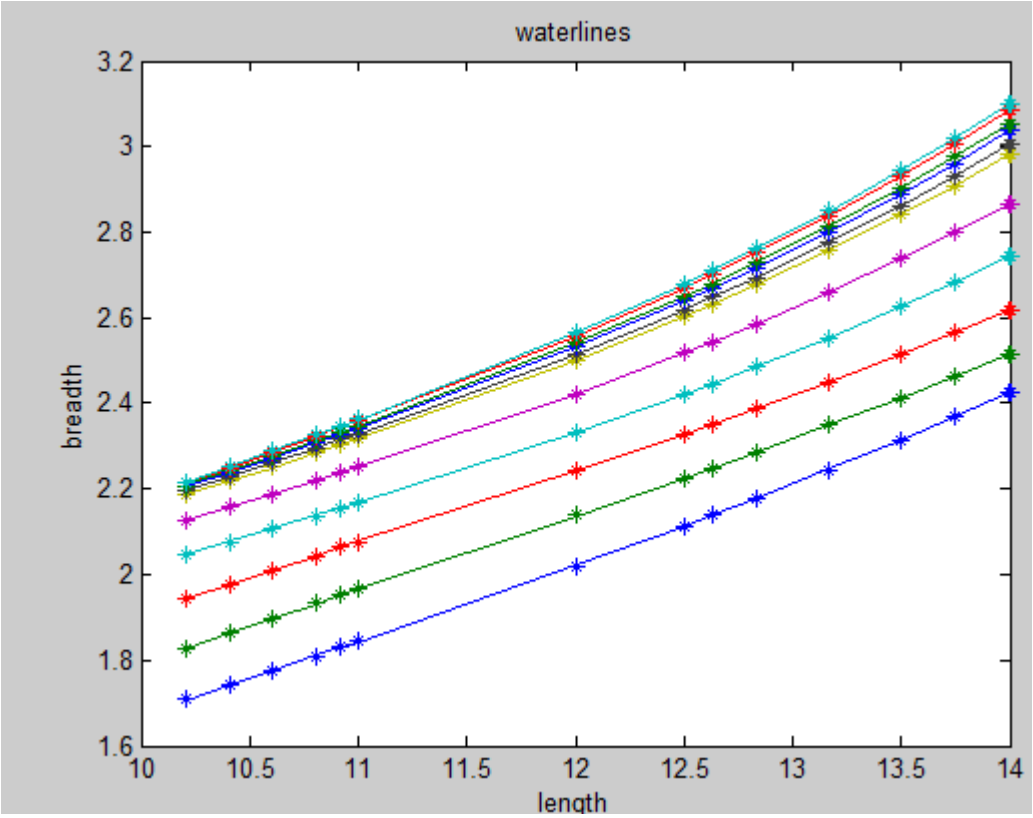


Fig 3
Waterlines plan at region where a plate development was requested

Figure 4 shows the development of a rather peculiar view area
The green shaded area in figure 4 shows the view plate sight while the red perimeter lines describe the movement of the burner and the following cutting path (counter clockwise). Blue lines are expansions of artificially located transverse frames for testing reasons.
Needless to say that the developed shape is correct only to the degree the view sight region is developable

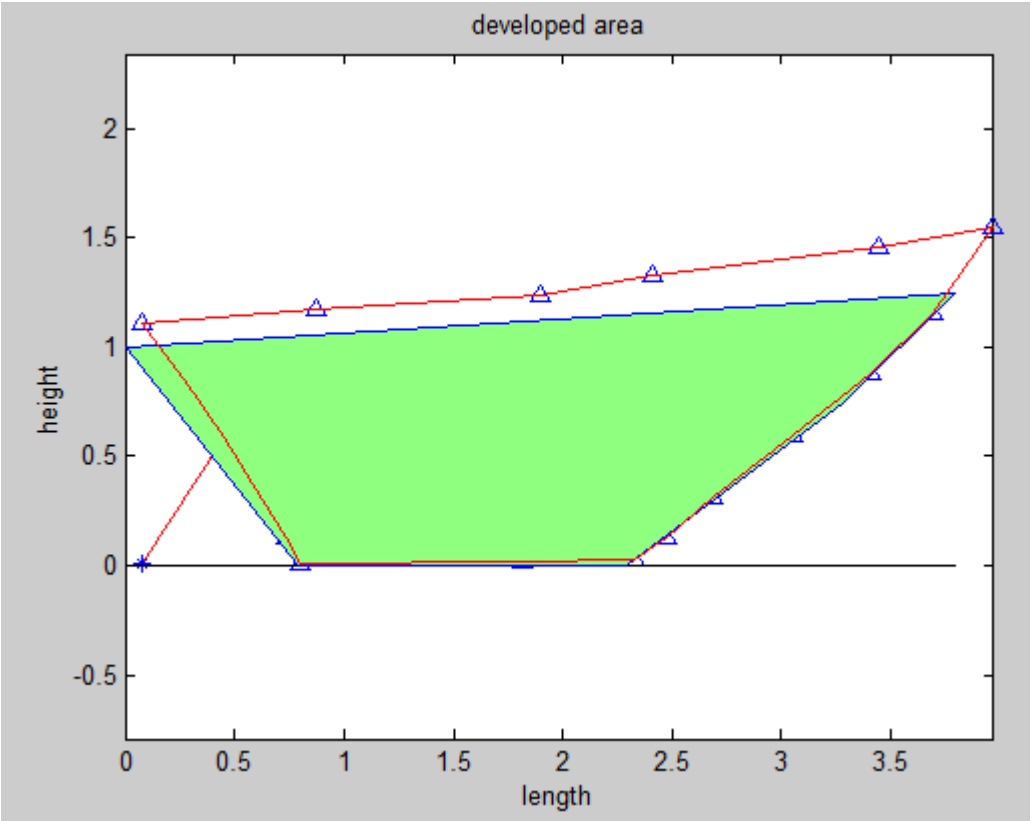


Fig 4
View plate (green) and its development

Figure 5 shows the process of nesting and cutting, models of frames to be used in production. By using appropriate code such models could be cut by an ESAP cutting machine from a thin plate. Big holes are provided in the models to reduce their weight. Note that the last two frame models have no holes cuts since their heights are two small.
Red lines indicate successive movements of the cutting burner while blue lines indicate successive real "plasma" burning.
In this study case nesting is lengthwise limited by a specified width of a thin plate of 5 meters.

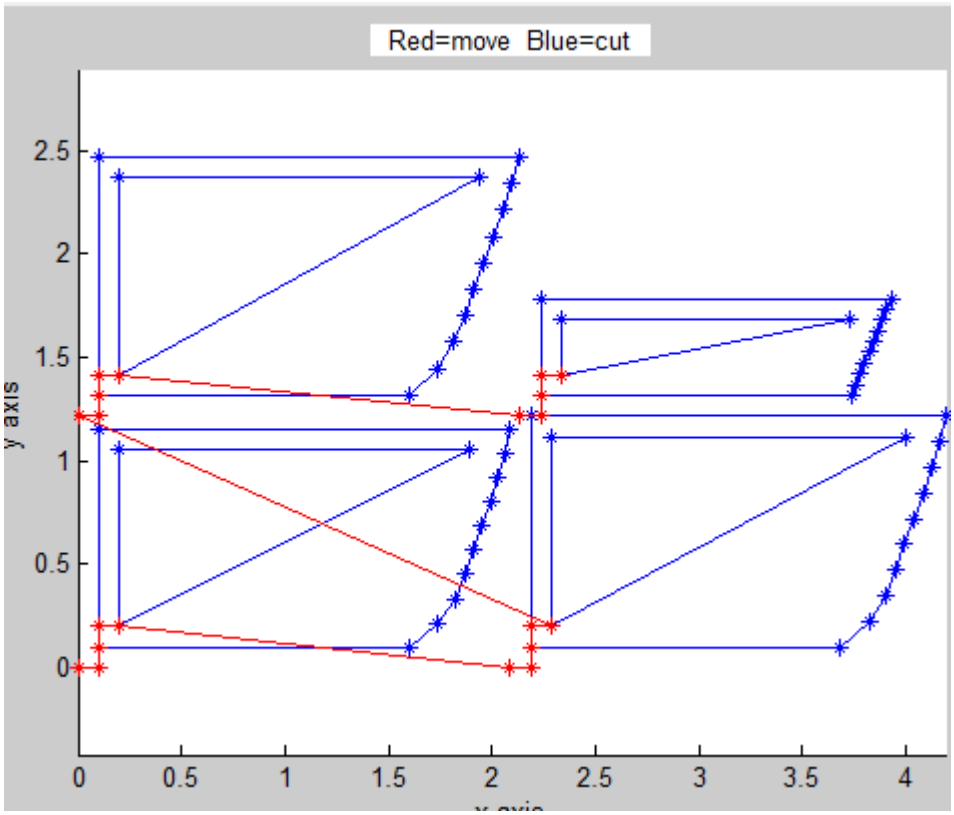


Fig 5
Process of cutting nested frame models

Fig 6 shows the process of marking the frame contours on the plate. Green lines show marking on the plate of the frame contours by the marking tool of the cutting machine while red lines show the movements of the marking tool from frame to frame

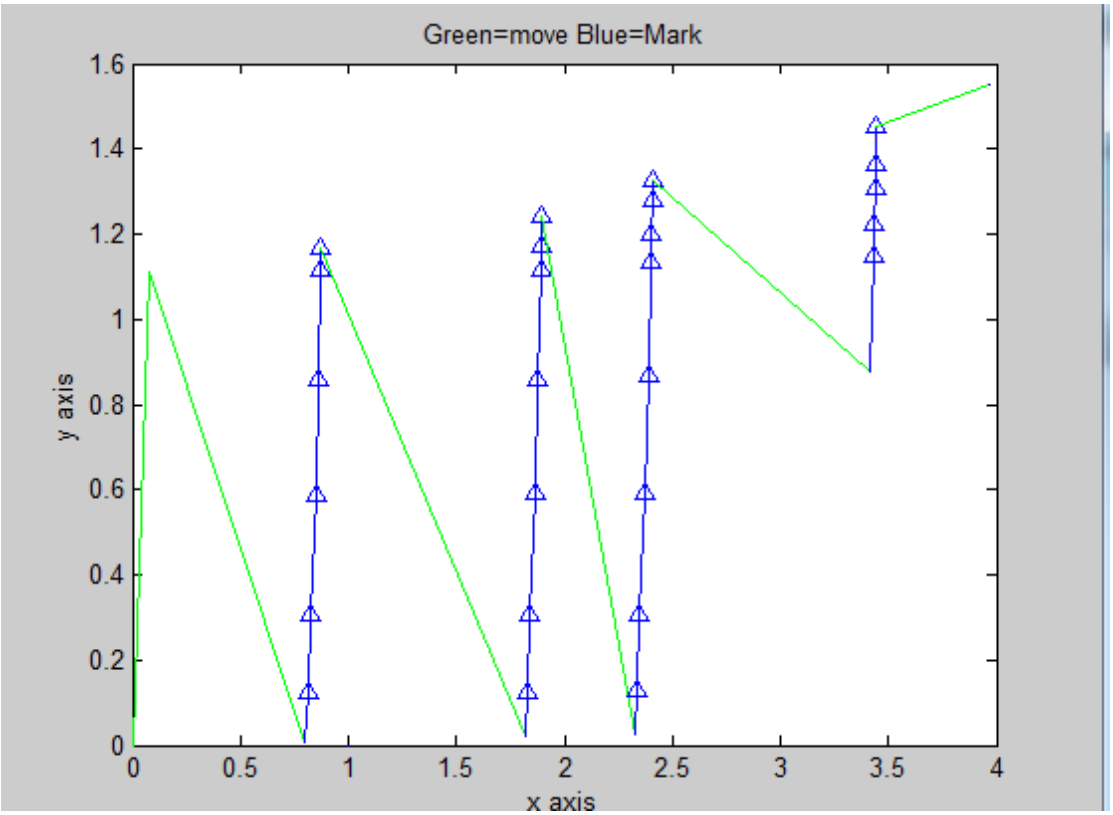


Fig 6
Marking of frames

PART FOUR –CONCLUSIONS

CHAPTERD 11
DISCUSSION OF RESULTS

Although the initial target of this work was to develop a method and soft wear for the mathematical representation of the geometry of the ships cell as a tool to make easier its restoration in case of damage, its potential is much broader.
The soft wear developed not only provides a tool for the hull mathematical representation but having also a production orientation is capable of generating necessary codes acceptable by automatic cutting machines for cutting ,marking and nesting without the need of human intervention.
The use of "linprog" Matlab module for solving large scale linear programming problems presented the following difficulties:
a.- Although the application of linear programming guarantees the satisfaction of constraints at places where offsets are supplied as inputs, it is not capable of providing correct information in between.
b.- Correction and fairing **of supplied input data at stations** followed by a similar process with the so **produced already corrected results at the waterlines where results were requested** and then a 3d fairing of the so faired offsets **overcomes this difficulty.**
c.-The older Matlab version used by the author presented serious difficulties in solving problems where the number of restrictions exceeded five hundred.
To overcome this difficulty it was necessary to split the problem to a number of smaller ones, describing ship geometry at groups of sections, and then combining them in such a way that transition was smooth from one group to its next one.
A positive side of this approach is that excluding the bow and stern regions the remaining part of lines of a normal shape ship (for example a ship without chine) can be corrected and faired in one pass.

A Program for expanding view (projected) areas has been developed. **However this program can yield absolutely correct results only if the relevant regions are developable as in the case of ships designed with developable surfaces and/or for the region of the parallel middle body of any ship.**
The following study cases were examined:
a.- A developable surface as that of a cone (study cases CONE_1 and CONE_2).
b.-A case of a real ship surface with input data already correct and faired for checking the performance of the programs (Study case 2)
c.-A case of a docked ship using data randomly taken from its undamaged side by laser (random cloud points, study case 1).
d.- A case from the stern of another ship (where changes of shape are rapid) with input data taken from a plan of scale 1:250
Basic observations, briefly saying, were:
a.- Results in the case of cone have shown absolute coincidence with reality if correct input data have been supplied.
b.- Results in the case of cone with arbitrary included errors were corrected and faired in a satisfactory way.
c.- Results with data taken from a real ship surface already faired were satisfactory. Negligible deviations between supplied and produced results appeared.
d.- In the last case where a cloud of laser measurements were used a complete production process took place. A mathematically faired plate was marked and cut by

an ESAB automatic cutting machine using code produced by the soft wear of the present work.
This plate has been formed using thin plate hollow frame models cut by using ESAB acceptable codes produced by soft wear of this work .It was then welded in place without any correction. Fairing at the loft shop was not necessary.
Serious difficulties experienced in using simultaneously FORTRAN programs and Matlab modules since transfer of information between each other should be made with great attention. This difficulty was balanced by the ability to take advantage of the special characteristics of the two languages.

CHAPTER 12.
RECOMMENTATIONS FOR FUTURE EXTENSIONS OF THIS WORK

The prevailing during the last two years conditions with Covit19, among which the limited mobility together with restrictions to visit and work in public places, imposed to the author serious restrictions to investigate further some aspects that could possibly result, to improvements of his approach.
As a consequence the following aspects should be further investigated:
a.-Examine the possibility to approach the subject using Dynamic instead of Linear programming.
b.-Examine the possibility to improve the approach by taking advantage of the increased capabilities of newer Matlab versions.
c.-Investigate more study cases especially with more dense cloud points collected by laser.
Adaptation of the present approach to handle ships with chine will be of much interest.
Cloud measurements taken from land can be very accurate when the ship is also on land. Obviously this is not the case with the ship afloat. Therefore the design of a simple gadget that could temporarily be steadily connected on any ship, having the laser equipment steadily connected on it , will be necessary for practical application and testing.
It must be noted that in this work passing from an input of cloud random measurements to measurements at stations and waterlines was done through triangulation. However correction of possible wrong points and fairing in the old sense could possibly be achievable directly without passing through the above transformation. That would be a worthwhile effort.
It is more than obvious that as the time passes all ships will be supplied with detailed information useful all over their useful life. This development will obviously eliminate progressively the need of measurements with laser or by any other methods. However a considerable part of this work will, after detailed testing, be useful especially at smaller and older yards. This implies the necessity of more testing in situ.
Even if that will be the case production methods will still need improvements to reach a stage that could be considered as fully automated.
Automation in marking, cutting and nesting is already well developed. This is not the case for plate forming which is still depending on ship loft and models of frames.
Multi piston presses fed with digital information such as that produced herein, could be considered as an interesting vision, that could eliminate time and man hours consuming trial and error methods used in our days.

ACKNOWLEDGMENTS

The author would like to express his deep appreciation to Professor (Emeritus) of National Technical University of Athens (NTUA) Theodore Loukakis, not only for his huge work in ship hydrodynamics ,but mainly for his successful efforts to found the Department of Naval Architecture and Marine Engineering at the above Institution.

Gratitude and thanks are due to Professor(Emeritus**)** George Tsabiras and Professor George Zarafonitis of NTUA for their inspiration in directing me to get involved with such an interesting subject.

Finally the author would express his gratitude and thanks to one of his old and beloved studenrs , namely mr Athanasios Malliris , Naval Architect (MIT) , who in addition to his overall support of social character, has devoted much of his time in transferring to me much of his long experience on the construction methods of merchant ships in China and elsewhere.

LIST OF REFFERENCES

1. 136.F. Theilheimer and W. Starkweather. The fairing of ship lines on a high speed computer. *Numerical Tables Aids Computation*, 15:338{355, 1961

2. 10. S. Berger, A. Webster, R. Tapia, and D. Atkins. Mathematical ship lofting. *J Ship Research*, 10:203{222, 1966.

3. 115.D. Rogers and S. Satter_eld. B-spline surfaces for ship hull design. *Computer Graphic s(Proc. Siggraph 80)*, 14(3):211{217, 1980.

4. Limitations of Computerized Lofting for Shell Plate Development, Naval Surface Warfare Center CD Code 2230,1993

5. L.W. Ferris,,A standard Series of Developable Surfaces, Marine Technology, January 1968

6.-S.Vaida, An introduction to Linear Programming and the Theory of Games, 1960,New York John Wiley & Sins Inc

7.-Charles D.Flagle,William H.Huggins, Robert H. Roy Operations Research and Systems Engineering ,1960, The Johns Hopkins Press,Baltimore, Chapter XIII written by Vincent V.McRAE

8.- G.Hadley Nonlinear and Dynamic Programming, 1964, Addison-Wesley Publishing Company,Inc London

9.- Optimization Toolbox for Use with MATLAB, User's Guide, Version 2, the Math Works Inc.

10.-Duan Hanselman , Bruce Littlefield , Mastering Matlab 6, A Comprehensive Tutorial and Reference ,Prentice Hall Inc,2001

11.- Mehrotra S. On the Implementation of a Primal-Dual Interior Point Method SIAM Journal on Optimization ,Vol 2,pp.575-601,1992

12.-Zhang Y, Solving Large scale Linear Programs by Interior –Point Methods Under the MATLAB Environment, Department of Mathematics and Statistics ,University of Maryland, Baltimore, Technical Report TR96-01,July,1995

13.-Ebru Sarioz An optimization Approach for Fairing of Ship Hull Forms-Faucalty of Naval Architecture and Ocean Enginnering ,Instabul Technical University,Turkey-Science Direct Ocean Enginnering 33 (2006) 2105-2118

14.-H. J. Kollman- Hull Form Design and Fairing , Tradition Restord-Scheepsbouwkundig Advies en Reken Cendrum SARC-Bussum,the Netherlands,1997

15.-Fog, N.G., 1985. B-spline surface system for ship hull design. ICCAS Computer Applications in theAutomation of Shipyard Operation and Ship Design V, 359–366.

16.-Kantorowitz, E., Drabkin, R., Krits, A., 2000. Fitting correctly shaped splines to ship lines given by inaccurate points. Ship Technology Research 47 (2), 63–66.

17.-Liu, J.P., Koyama, T., Yamato, H., 1991. Constrained Smoothing B-spline Curve Fitting for Ship Hull Generation and Fairing. Proceedings of Computer Applications in the Automation of Shipyard Operation and Ship Design VI. Elsevier Science Publishers, Amsterdam, pp. 221–232.

18.-McCallum, K.J., Zhang, J.M., 1986. Curve-smoothing techniques using B-splines. The Computer Journa l29 (6), 564–569.

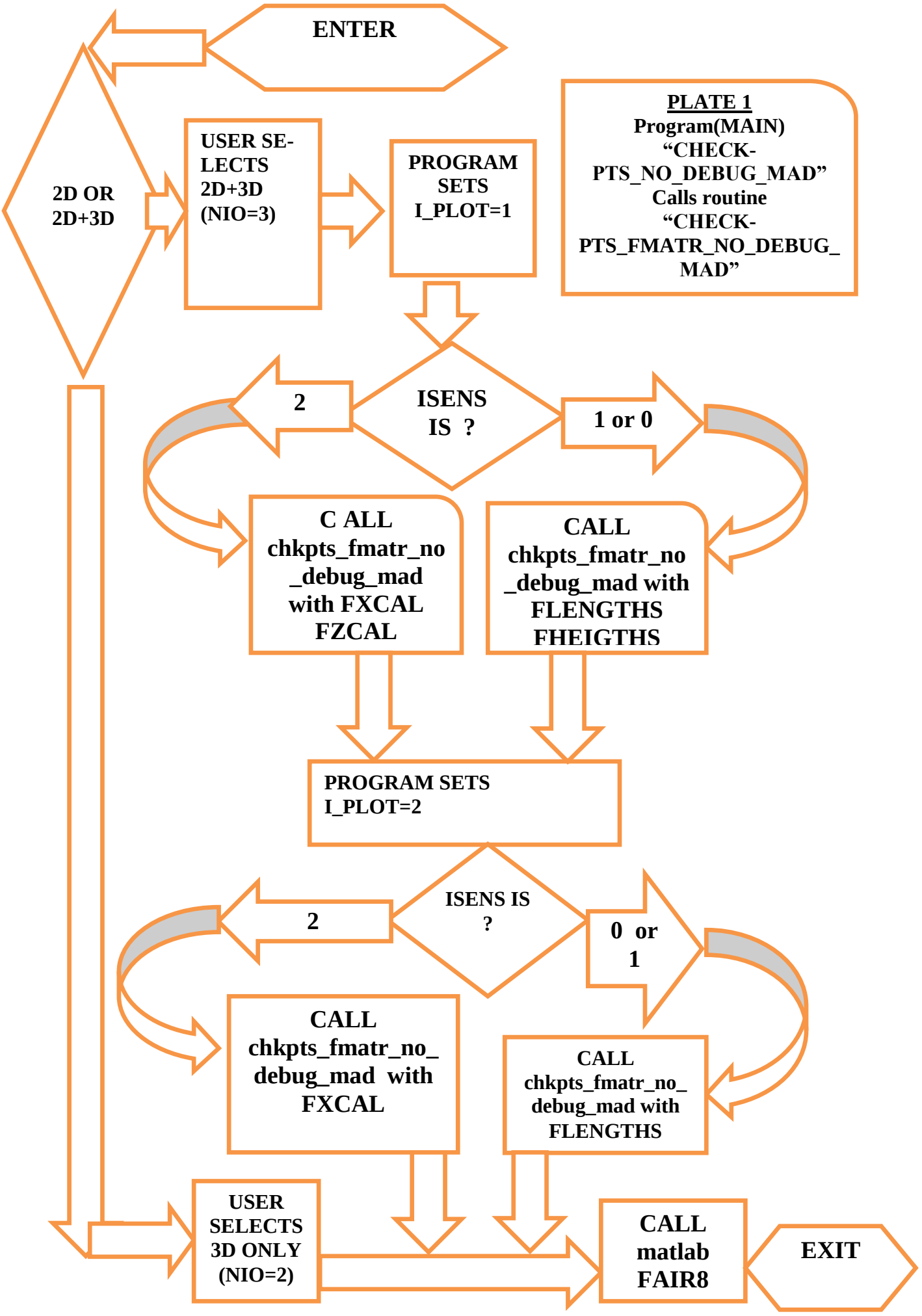
19.-Moreton, H.P., Sequin, C.H., 1992. Functional optimisation for fair surface design. Computer Graphics 26 (2), 167–176.
20.-Nowacki, H., Lu¨, X., 1994. Fairing composite polynomial curves with constraints. Computer Aided Geometric Design 11 (1), 1–15.
Nowacki, H., Reese, D., 1983. Design and fairin

APPENDIX A

PROGRAMS DOCUMENTATION

ANNEXES AND PLATES

ANNEX(1) + PLATE(1)
DOCUMENTATION OF FORTRAN PROGRAM
“checkpts_no_debug_mad.for”


ENTER
2D OR 2D+3D
USER SE-LECTS 2D+3D (NIO=3)
PROGRAM SETS I_PLOT=1
PLATE 1
Program(MAIN) "CHECK-PTS_NO_DEBUG_MAD"
Calls routine "CHECK-PTS_FMATR_NO_DEBUG_MAD"
ISENS IS ?
2
1 or 0
C ALL chkpts_fmatr_no_debug_mad with FXCAL FZCAL
CALL chkpts_fmatr_no_debug_mad with FLENGTHS FHEIGTHS
PROGRAM SETS I_PLOT=2
ISENS IS ?
2
0 or 1
CALL chkpts_fmatr_no_debug_mad with FXCAL
CALL chkpts_fmatr_no_debug_mad with FLENGTHS
USER SELECTS 3D ONLY (NIO=2)
CALL matlab FAIR8
EXIT

FORTRAN PROGRAM "checkpts_no_debug_mad"

```
---------------------------------------
PROGRAM TEST1
C     PREPARES DATA FOR 3D FAIRING PROBLEM
C     N= NUMBER OF STATIONS  WHERE DATA ARE GIVEN
C     M= NUMBER OF WATERLINES WHERE DATA ARE GIVEN
C     X(I) I=1,N X COORDINATE OF GIVEN STATION
C     Z(J) J=1,M Z COORDINATE OF GIVEN WATERLINE
C     Y(I,J) I=1,N _J=1,M GIVEN HALF BREADTHS
C     ZCAL(J1) J1=1,M1 COORDINATES OF WL's where output is required
C     XCAL(I1) I1=1,N1 COORDINATES OF STATIONS where output is required
c     YCAL(I1,J1) I1*J1 CALCULATED HALF BREADTHS

IMPLICIT REAL*8(A-H,O-Z)
integer engopen,enggetmatrix,mxcreatefull,mxgetpr
integer ep,T,T1,T2      ,T3      ,T22     ,T11     ,t21,ti1400,ti1401,t20
integer engputmatrix,engevalstring,engclose
integer status
double precision aa(6000,6000),aaf(2250000),aaq(2250000)
CHARACTER*1 TI(50,50),TI1(1500),TIF(50,50),TIVAL
CHARACTER*100 mydir
CHARACTER*100 TITLE
DIMENSION X(500),Y(500,500),XXCAL(800),YCAL(800,800)
*,CC(800,800),S(500,500),DD(6000,6000)
*,Z(50),XCAL(80),ZCAL(80),Y1(1500),xcalfac(80),zcalfac(80)
*,AAA(800),CCC(1500),R(500,500),RR(800,800),RRC(800,800),yy(1500)
*,xlim(10,4),ylim(10,4),zlim(10,4),xxlim(40),zzlim(40)
*,xelim(40),zelim(40),y1eq(1500),y1ineq(1500),nti(1500),ntj(1500)
*,ntii(1500),ntjj(1500),ntiii(1500),ntjjj(1500),nsta(40),nend(40)
DIMENSION FX(500),FY(500,500),FZ(500),dio(500),dddd1(500,500),
*FXCAL(80),FZCAL(80),DDDD2(500,500),DDDD3(500,500)
DIMENSION XF(800),XXCALF(800),yyF(800),iniF(800),dddd(800),xfb(800)
DIMENSION FXF(7),F(7),H(7),IFF(1500),FZO(500,500),FXO(500,500)
*,npoints(1400,1400)
dimension x_left_down(100),z_left_down(100),slope(100)
dimension y_point(100),FLENGTHS(1500),FHEIGHTS(1500),H_2dif(5),
* yyf_2dif(80),finit(7),ddc(5),finit_2dif(80),yyf1(800),finit1(7),
*yyf2(800),FY_IN(500),H_stat(500)

dimension che(80),che1(80,80),val(80),val1(80,70)

OPEN(1,FILE='mat3d.DAT')
OPEN(22,FILE='mat3d22.OUT')
OPEN(220,FILE='rs.OUT')
OPEN(221,FILE='r_IN_2d')
OPEN(222,FILE='s_IN_2d')
OPEN(321,FILE='r_OUT_2d')
OPEN(322,FILE='s_OUT_2d')
open(3030,file='mat3d.m')
```

```
OPEN (88,FILE='MAT3D1.DAT')
close (88,status='DELETE')
OPEN (88,FILE='MAT3D1.DAT')

close (8808,status='DELETE')
OPEN (8888,FILE='CONSTANTS.DAT')

close (88888,status='DELETE')
OPEN (88888,FILE='CONSTANTS1.DAT')

open (881,file='mat3d2.dat')
open (100,file='A.dat')
open (101,file='b.dat')
open (2001,file='DD.dat')
open (102,file='xzcoord.dat')
open (103,file='beq.dat')
open(200,file='skip')
OPEN (300,FILE='submitted.dat')
open (421,file='mat3d421.out')
open (301,file='submitted1.dat')
open (302,file='ndf.dat')
open (4000,file='nozero.dat')
open (5000,file='AEQ.dat')
open (110,file='AB.dat')
open (501,file='ddgiven400.dat')

open (502,file='dddd1400.dat')
open (500,file='zzgiven400.dat')
open (600,file='IFF.dat')

open (700,file='M_WL.dat')
open (701,file='N_STA.dat')

open (402, file='gendat.dat')

close (404,status='DELETE')
open (404, file='gendat1.dat')

open (7001, file='FX.dat')
open (7002, file='FZ.dat')

open (601, file='x1cal.dat')
open (602, file='z1cal.dat')

close (903,status='DELETE')
close (904,status='DELETE')
open (903, file='xxlim.dat')
open (904, file='zzlim.dat')

close (905,status='DELETE')
```

```
close (906,status='DELETE')
open (905, file='xelim.dat')
open (906, file='zelim.dat')

close (1000,status='DELETE')
open (1000, file='DET.dat')

open (3, file='A_TEST.dat')

open (907, file='new_fy.dat')
close (907,status='DELETE')
open (907, file='new_fy.dat')

open (908, file='DDDD2_MAIN.dat')
open (909, file='DDDD1_MAIN.dat')
open (2002, file='fval_all.dat')
open (2004, file='FY_GIVEN.dat')
open (3002, file='FXCAL.dat')
open (3003, file='FZCAL.dat')
open (4001, file='x.dat')
open (4002, file='z.dat')

open (550, file='FZO.dat')
open (551, file='I_EXIT.dat')

open (552, file='ISENS.dat')

open (901, file='N_SI_LE.dat')
open (902, file='M_SI_HE.dat')
open (801, file='LENGTHS.dat')
open (802, file='HEIGHTS.dat')

open (9011, file='NN_SI_LE.dat')
open (9021, file='MM_SI_HE.dat')

close (8011,status='DELETE')
close (8021,status='DELETE')
open (8011, file='LLENGTHS.dat')
open (8021, file='HHEIGHTS.dat')

open(702,file='val.dat')

open(704,file='val_r.dat')
open(705,file='NO_EQUAT.dat')

close (706,status='DELETE')
OPEN(706,FILE='title.dat')

close (901,status='DELETE')
close (902,status='DELETE')
```

```
close (801,status='DELETE')
close (802,status='DELETE')
close (9011,status='DELETE')
close (9021,status='DELETE')
close (8011,status='DELETE')
close (8021,status='DELETE')

OPEN(707,FILE='nio.dat')
TIVAL='*'

write(*,*) ('Give 3 (nio=3) for 2d and then 3d fairing')
write(*,*) ('Give 2 (nio=2) for DIRECT 3d fairing')
Read (*,*) nio
write (707,*) nio

WRITE  (*,*) ('WRITE A NUMBER FOR THE  PROJECT AND PRESS "ENTER"')
READ(*,*) title

mydir=title
write (*,*) mydir,title,kol

result=makedirqq(mydir)

write (706,*) title

write(*,*) ('Give 0 (IDEB=0) for no_debug OR IDEB=1 FOR DEBUG')
read (*,*) IDEB
write(*,*) ('Give 0 for input offsets at SAME WL_s on
* every station (OFF_KIND=0)')
write(*,*) ('Give 1 for input offsets at DIFFERENT WL_s on
* every station (OFF_KIND=1)')
read (*,*) OFF_KIND
write (*,*) ('SPECIFY NO OF CYCLES FOR POINT CORRECTION ,N_PO')
read (*,*) N_PO
1121  format(a100)

write (*,*) title
pause 1121
if (OFF_KIND.eq.0) then
I_FIL=1
open (1, file='mat3d.dat')

else
I_FIL=11111
open (11111, file='mat3d_old.dat')
endif
READ (I_FIL,*) N,M,N1,M1,fac,scale,spax,spaz,NOPOX,NOPOZ,off,
*isens,tolf
write (*,*) nio,OFF_KIND,ISENS
```

```
if( (nio.eq.2).and.(OFF_KIND.eq.1.0).and.(ISENS.eq.2) ) then
pause
write (*,*) 'ATTENTION !!!!! direct 3d correction of wrong '
write (*,*) 'points and fairing canot take place '
write (*,*)'Rerun the  program with nio=3'
pause
go to 46002
3232    pause 3232
else
endif

write (300,*) N,M,N1,M1,fac,scale,spax,spaz,NOPOX,NOPOZ,
* off,isens
write (*,*) N,M,N1,M1,fac,scale,spax,spaz,NOPOX,NOPOZ,
* off,isens
write (552,*) ISENS
I_EXIT=0
write (1000,*)  '       INITIAL DATA              '
write (1000,*)  '       ------------              '
write (1000,*)  '                                 '

write (1000,*)  ' NUMBER OF STATIONS              ',N
write (1000,*)  ' NUMBER OF WATERLINES            ',M
write (1000,*)  ' NUMBER OF STATIONS FOR RESULTS  ',N1
write (1000,*)  ' NUMBER OF WATERLINES FOR RESULTS  ',M1
write (1000,*)  ' ARBITRARY FACTOR                ',fac
write (1000,*)  ' SCALE                           ',scale
write (1000,*)  ' SPACING OF ADDITIONAL STATIONS  ',spax
write (1000,*)  ' SPACING OF ADDITIONAL WATERLINES  ',spaz
write (1000,*)  ' NO OF ADDITIONAL STATIONS       ',NOPOX
write (1000,*)  ' NO OF ADDITIONAL WATERLINES     ',NOPOZ
write (1000,*)  ' OFF CENTER DISTANCE  ,metres    ',off

write (1000,*)  ' ALLOWED POINTS TOLERANCE   ,metres',tolf
write (1000,*)  '                                 '
write (1000,*)  ' SELECTION CODE                  ',isens

write (1000,*)  '    for 0 SELECTION CODE-LINES+PLATE     '
write (1000,*)  '    for 1 SELECTION CODE-LINES+PLATE +CUT TAPES'
write (1000,*)  '    for 2 SELECTION CODE-LINES ONLY          '

write (1000,*)  '                                 '
write (1000,*)  ' SELECTION CODE  FOR TYPE OF FAIRING',nio
write (1000,*)  '    for 2 3D FAIRING      '
write (1000,*)  '    for 3 2D AND THEN 3D FAIRING'

write (1000,*)  ''
write (1000,*)  ''

OFF=OFF/(SCALE*FAC)
```

```
READ (I_FIL,*) (FX(I),I=1,N)
write (300,*) (FX(I),I=1,N)

write (7001,1001) (SCALE*FX(I),I=1,N)

101 FORMAT (F8.3,tr1,A1,15(TR1,F8.3,TR1,A1))
1001 FORMAT (F8.3)
10011 format(100F8.3)
DO 10 J=1,m
11101 READ (I_FIL,101) FZ(j),(TIF(i,j),FY(I,j),I=1,n)
write (300,101) FZ(J),(TIF(I,J),FY(I,J),I=1,N)
write (2004,10011) (SCALE*FY(I,J),I=1,N)
write (7002,101) SCALE*FZ(J)
10 WRITE (200,101) FZ(J),(TIF(I,J),FY(I,J),I=1,N)
If(IDEB-1) 5502,5501,5502
5501 pause 5501
5502 continue

write (*,*) M,N
If(IDEB-1) 5504,5503,5504
5503 pause 5503
5504 continue
n_sav=N
m_sav=M

C     CHOSE WAY OF READING FZO
if (OFF_KIND.eq.0) then
c     NEW kind of data
DO 91112 I=1,N
DO 91112 J=1,M
FZO(I,J)=FZ(J)

550 FORMAT (100F10.3)
If(IDEB-1) 5506,5505,5506
5505 pause 5505
5506 continue
91112 continue
DO 91115 J=1,M
write (550,550) (SCALE*FZO(I,J),I=1,N)
write (300,*) (FZO(I,J),I=1,N)
write (*,*) (FZO(I,J),I=1,N)
91115 continue

else
c     OLD kind of data
DO 91122 J=1,M
read (I_FIL,*) (FZO(I,J),I=1,N)
write (*,*) (FZO(I,J),I=1,N)
write (550,550) (SCALE*FZO(I,J),I=1,N)
```

```
If(IDEB-1) 5516,5515,5516
5515 pause 5505
5516 continue

write (300,*) (FZO(I,J),I=1,N)
91122 continue
endif
C     end CHOSE WAY OF READING FZO

DO 91113 I=1,N
DO 91113 J=1,M
FXO(I,J)=FX(I)*SCALE
write (*,*) FXO(I,J)
91113 continue

DO 91114 J=1,M
write (*,*) (FXO(I,J),I=1,N)
91114 continue

write (*,*) N1,M1

READ (I_FIL,*) (FXCAL(I),I=1,N1)

WRITE (3002,5002)(SCALE*FXCAL(I),I=1,N1)
READ (I_FIL,*) (FZCAL(J),J=1,M1)

WRITE (3003,5002) (SCALE*FZCAL(J),J=1,M1)

c         MULTIPLY DATA BY factor SCALE
DO 102 I=1,N
102 FX(I)=SCALE*FX(I)

DO  103 J=1,M
103 FZ(J)=SCALE*FZ(J)

DO  104 J=1,M
DO  104 I=1,N
104 FY(I,J)=SCALE*FY(I,J)

DO 105 I=1,N1
105 FXCAL(I)=SCALE*FXCAL(I)

DO  106 J=1,M1
106 FZCAL(J)=SCALE*FZCAL(J)

DO  107 J=1,M
DO  107 I=1,N
107 FZO(I,J)=SCALE*FZO(I,J)

c         end MULTIPLY DATA BY factor SCALE
```

```
c     PREPARE ADDITIONAL FXCAL, FZCAL
c     FIRST FIND ADDITIONAL X POINTS
XCONST=FXCAL(1)

do 1101 INEW=N1+1,N1+NOPOX
STEPX=(FXCAL(N1)-FXCAL(1))/(NOPOX+1)


FXCAL(INEW)=XCONST+(INEW-N1)*STEPX
write (*,*) XCONST,I_NEW,N1,STEPX,NOPOX,FXCAL(INEW)
pause 1101
1101 continue

N_ALL=N1+NOPOX
M_ALL=M1+NOPOZ
write (*,*) 'N_ALL',N_ALL,'M_ALL',M_ALL,N1,M1,NOPOX,NOPOZ
c     END COUNT TOTAL X-OUTPUT POINTS

c     PUT X-POINS IN ASCENDING ORDER
do 11011 I=1,N1+NOPOX
do 11011 J=2,N1+NOPOX
if(FXCAL(J)-FXCAL(J-1).LT.0) then
small=FXCAL(J)
big=FXCAL(J-1)
FXCAL(j)=BIG
FXCAL(J-1)=SMALL
else
continue
endif
11011 continue
c     END PUT X-POINS IN ASCENDING ORDER

c     COUNT TOTAL Z-OUTPUT POINTS
ZCONST=FZCAL(1)

do 1102 INEW=M1+1,M1+NOPOZ
STEPZ=(FZCAL(M1)-FZCAL(1))/(NOPOZ+1)

write (*,*) 'STEPZ=', stepz,FzCAL(m1),FzCAL(1)
FZCAL(INEW)=ZCONST+(INEW-M1)*STEPZ
write (*,*)FZCAL(INEW)
pause 1102
1102 continue
c     END COUNT TOTAL X-OUTPUT POINTS


M_ALL=M1+NOPOZ
write (*,*) 'M_ALL',M_ALL
```

```
c     PUT Z-POINS IN ASCENDING ORDER
do 11012 I=1,M1+NOPOZ
do 11012 J=2,M1+NOPOZ
if(FZCAL(J)-FZCAL(J-1).LT.0) then
small=FZCAL(J)
big=FZCAL(J-1)
FZCAL(j)=BIG
FZCAL(J-1)=SMALL
else
continue
endif
11012 continue

c     END PUT Z-POINS IN ASCENDING ORDER

N1TOT=N_ALL
M1TOT=M_ALL
c     END PREPARE ADDITIONAL FXCAL, FZCAL


if(  (NOPOX.eq.0). and. (NOPOZ.eq.0) ) then

N1TOT=N1
M1TOT=M1
write(*,*) N1,M1
write(*,*) N!TOT
write (*,*) M1TOT
pause 321
do 321 I=1,N1TOT
write (601,*) fxcal(I)
321 continue


do 322 I=1,M1TOT
write (602,*)  fzcal(I)
write (*,*) I,FZCAL(I)

322 continue
write (*,*) N1TOT,M1TOT
write (300,*) (FXCAL(I),I=1,N1TOT),nio
write (300,*) (FZCAL(J),J=1,M1TOT)
write (*,*) (FXCAL(I),I=1,N1TOT),nio
write (*,*) (FZCAL(J),J=1,M1TOT)

pause 1102
else

write (*,*) (FXCAL(I),I=1,N1TOT),nio
write (*,*) (FZCAL(I),I=1,N1TOT)
rewind 3002
```

```
rewind 3003
pause 1103
write (3002,5002) (SCALE*FXCAL(I),I=1,N1TOT)
write (3003,5002) (SCALE*FZCAL(I),I=1,N1TOT)
pause 1101
endif

C     DATA READ

if (ISENS-1) 30140,30140,1021
30140 read (I_FIL,*) kko

write (88,*) kko
do 11112 i=1,kko
read (I_FIL,*) xxlim(i),zzlim(i),xelim(i),zelim(i)
write(903,65516) SCALE*xxlim(i)
write(904,65516) SCALE*zzlim(i)
write(905,65516) SCALE*xelim(i)
write(906,65516) SCALE*zelim(i)
write (300,*) xxlim(i),zzlim(i),xelim(i),zelim(i)
11112  write (88,3013) xxlim(i),zzlim(i),xelim(i),zelim(i)

write(903,65516) SCALE*xxlim(1)
write(904,65516) SCALE*zzlim(1)

write(905,65516) SCALE*xelim(1)
write(906,65516) SSCALE*zelim(1)
65516 format(f10.4)
65517 format(100f10.4)

go to 30000
30150 read (I_FIL,*) kko

write (88,*) kko
do 11113 i=1,kko
read (I_FIL,*) xxlim(i),zzlim(i),xelim(i),zelim(i)
write (88,*) xxlim(i),zzlim(i),xelim(i),zelim(i)
11113  continue

c     END SPECIFY POINTS WHERE RESULTS ARE REQUIRED

30000 continue
write (*,*) kko ,xxlim(5), xelim(5), zzlim(5), zelim(5)
write (404,*) N_ALL,M_ALL,kko,nn1,mm1

ep=engopen('matlab')
pause 30002
t20=engevalstring(ep,'mynew128')
status= engClose(ep)
C     END PRODUCE IN MATLAB NET OF POINTS OF ANY PLATE and grid
```

```
C     IMPORT AND READ DATA PRODUCED BY MATLAB
open (6011, file='points.dat')

open (901, file='N_SI_LE.dat')
open (902, file='M_SI_HE.dat')

open (801, file='LENGTHS.dat')
open (802, file='HEIGHTS.dat')

open (9011, file='NN_SI_LE.dat')
open (9021, file='MM_SI_HE.dat')

open (8011, file='LLENGTHS.dat')
open (8021, file='HHEIGHTS.dat')

rewind 901
rewind 902
rewind 9011
rewind 9021

READ(901,*) N_LE
READ(902,*) M_HE
WRITE(88,*) N_LE,M_HE

READ(9011,*) NN_LE
READ(9021,*) MM_HE
nn1=N_LE
mm1=M_HE
rewind 6011

do 202 i=1,nn1
read (6011,*) (npoints(j,i),j=1,mm1)

202 continue

write (*,* ) nn1,mm1

C     PROCEED AND CHECK TWO DIMENSIONAL FAIRING

1021 continue

Kn=1
NF=M
MF=M

DO 81820 LIO=1,N_PO
C        MODIFY THIS DO IF YOU WANT MORE CYCLES OF OF COMPUTA-
TIONS
rewind 4001
```

```
rewind 4002
DO 81822 J_OUT=1,N
81822  WRITE (*,*) 'FY',( FY(J_OUT,I_OUT),I_OUT=1,M)


111      do 11 i=1,n1
x(i)=fxcal(i)/fac
11 write (4001,*) x(i)

do 12 j=1,m1
z(j)=fzcal(j)/fac
12 write (4002,*) z(j)

if (LIO.eq.1) then
do 1313 i=1,n
do 1313 j=1,m
1313 y(i,j)=fy(i,j)/fac

else
close (907,status='DELETE')
OPEN(907,FILE='new_fy.dat')
rewind 907

do 13131 i=1,n
do 13131 j=1,m
13131 y(i,j)=DDDD3(i,j)
endif

do 1314 i=1,n
do 1314 j=1,m
1314 FZO(i,j)=FZO(i,j)/fac

81819 continue
i_plot=1

if(ISENS-1) 2022, 2022,2021
2021 continue
call chkpts_fmatr(m,n,xf,f,fy,FZO,jn,dddd1,nio,I_EXIT,i_plot
*,FXCAL,FZCAL,N1,M1,FX,TIF,n_sav,m_sav,ISENS,N_LE,M_HE,IDEB)
go to 2023
2022    write (*,*) M_HE
write (*,*) N_LE
DO 3001 K=1,N_LE
READ (8011,*) FLENGTHS(K)
3001 write (*,*) FLENGTHS(K)
DO 3002 L=1,M_HE
READ (8021,*) FHEIGHTS(L)
3002 write (*,*) FHEIGHTS(L)

rewind 4001
```

```
rewind 4002
do  5021 i=1,N_LE
x(i)=FLENGTHS(i)
5021    write (4001,*) x(i)

do  5022 j=1,M_HE
z(j)=FHEIGHTS(j)
5022    write (4002,*) z(j)

call chkpts_fmatr(m,n,xf,f,fy,FZO,jn,dddd1,nio,I_EXIT,i_plot
*,FLENGTHS,FHEIGHTS,N_LE,M_HE,FX,TIF,n_sav,m_sav,ISENS,N_LE,M_HE
*,IDEB)
2023 continue

777   continue
ic=ic+1
write (*,*) 'IC VALUE',ic
if(ISENS-1) 4022, 4022,4021
4021    write (*,*) 'M1 is',M1,'N1 is',N1,'M is',M,'N is',N
DO 81825 J_OUT=1,m1
DO 81825 i_OUT=1,N

write (909,*)  DDDD1(i_OUT,j_OUT)
81825  WRITE (*,*) 'DDDD1in main vertically', DDDD1(i_OUT,j_OUT)
*,I_OUT,J_out
go to 4023
4022 write (*,*) 'M_HE is',M_HE,'N_LE is',N_LE,'M is',M,'N is',N
DO 81635 J_OUT=1,M_HE
DO 81635 i_OUT=1,N

write (909,*)  DDDD1(i_OUT,j_OUT)
81635  WRITE (*,*) 'DDDD1in main vertically', DDDD1(i_OUT,j_OUT)
*,I_OUT,J_out

4023 continue

if (ISENS-1)7002,7002,7001

7001 DO 81823 J_OUT=1,m1
DO 81823 I_OUT=1,n
81823 DDDD2(J_OUT,I_OUT)=DDDD1(I_OUT,J_OUT)
DO 81829 J_OUT=1,m1
DO 81829 i_OUT=1,n
write (908,*) DDDD2(J_OUT,i_OUT),I_OUT,J_OUT
81829 WRITE (*,*) 'DDDD2in main ', DDDD2(J_OUT,i_OUT),I_OUT,J_OUT
go to 7010

7002 DO 7003 J_OUT=1,M_HE
DO 7003 i_OUT=1,n
7003 DDDD2(J_OUT,I_OUT)=DDDD1(I_OUT,J_OUT)
```

```
DO 7004 J_OUT=1,M_HE
DO 7004 i_OUT=1,n
7004  write (908,*) DDDD2(J_OUT,i_OUT),I_OUT,J_OUT
7010 continue

C     RECHECK CORECTNESS OF POINTS LONGITUDINALLY
write (*,*) nio

do 13 i=1,n
do 13 j=1,m
13 y(i,j)=dddd1(i,j)/fac
go to 81809

81809 continue

i_plot=2
write (*,*) 'I_PLOT=',I_PLOT,'N    ', N, 'M1   ',M1
write (*,*) N,N1,M1,n_sav,m_sav,N_LE,M_HE

IF (ISENS-1) 6022, 6022, 6021
6021 call chkpts_fmatr(N,M1,xf,f,DDDD2,FXO,jn,DDDD3,nio,
*I_EXIT,i_plot,FXCAL,FZCAL,N1,M1,FX ,TIF, n_sav,m_sav,
*ISENs,N_LE,M_HE,IDEB)

GO TO 6025
6022 call chkpts_fmatr(N,M1,xf,f,DDDD2,FXO,jn,DDDD3,nio,
*I_EXIT,i_plot, FLENGTHS,FHEIGHTS,N_LE,M_HE,FX
*,TIF, n_sav,m_sav,ISENs,N_LE,M_HE,IDEB)
6025 CONTINUE

rewind 5001
open (5001,file='DDDD3.dat')
M1_KP=M1
N1_KP=N1
IF (ISENS-1) 7022, 7022, 7025
7022 M1=M_HE
N1=N_LE
7025 DO 411 J=1,M1
5002 format (100F15.3)
WRITE (5001,5002) (DDDD3(J,JIO),JIO=1,N1)
411 WRITE (*,*) 'DDDD3=',(DDDD3(J,JIO),JIO=1,N1) ,J,JIO

DO 82835 J_OUT=1,m1
DO 82835 i_OUT=1,N_SAV
82835  WRITE (*,*) 'DDDD3in main horizonally', DDDD3(J_OUT,i_OUT)
*,j_OUT,i_out

write (*,*) N1_SAV,m1_SAV,N1,M1

DO 81835 J=1,N1
```

```
DO 81835 Jio=1,M1
y(j,jio)=dddd3(jio,j)
write (*,*) j,jio, y(j,jio)
81835  fy(j,jio)=dddd3(jio,j)

DO 81834 J=1,M1
write (907,8089) (y(JIO,j),JIO=1,N1)
81834 WRITE (*,*) 'new_y=',(y(JIO,j),JIO=1,N1)
pause 81834

8089    format (100f25.13)
6000 continue
C     END RECHECK CORECTNESS OF POINTS LONGITUDINALLY

81818 continue

81820   continue
c     NOW START 3D PROCESSING
ep=engopen('matlab')

c     SCIP FURTHER PROCESS IF DATA INCOMPATIBLE
IF (I_EXIT.GT.1) THEN
GO TO 45000
ELSE
CONTINUE
ENDIF
c     end SCIP FURTHER PROCESS IF DATA INCOMPATIBLE

TIYES=0
TINO=0
DO 9 I=1,N
DO 9 J=1,M
IF (TIf(I,J).EQ.TIVAL) THEN
TIYES=TIYES+1
GO TO 9
else
ENDIF
TINO=TINO+1
9 CONTINUE
WRITE (200,*) TIYES,TINO,'TEST'

write (301,*) (X(I),I=1,N)
DO 1010 J=1,M1
1010 write (301,*) Z(J),(Y(I,J),I=1,N)
write (301,*) (XCAL(I),I=1,N1)
write (301,*) (ZCAL(J),J=1,M1)
```

```
C    CALCULATE BASIC VARIABLES
C    NM=N*M NO OF BASIC EQUATIONS NO WITH - LAMDA
C    NM2=2*N*M=NO OF BASIC EQUATIONS WITH - AND + LAMDA
C    NSTAv=NO OF STANDARD VARIABLES NO,A's =33
C    NB=NO OF COEFFICIENTS B's=2*(n-2)
C    NC=NO OF COEFFICIENTS C's=2*(m-2)
C    ND=NO OF COEFFICIENTS D's=2*(n-2)*(m-2)
C    NEQ=NO OF EQUATIONS=N*M*2+N(M-2)+M(N-2)
C    IBEG=NO OF BEGIN OF ARTIFICIAL VARIABLES=
C    IEND=NO OF END OF ARTIFICIAL VARIABLES=
C    NP =NO OF P DUMMY VARIABLES=N*M
C    NS= NO OF S DUMMY VARIABLES=N*M
C    NX= NO OF X ARTIFICIAL DUMMY VARIABLES=N*M
C    NQ= NO OF Q DUMMY VARIABLES=M*(N-2)
C    NR= NO OF R DUMMY VARIABLES=N*(M-2)
C              MAX NO OF VARIABLES=33+2*(N-2)+2*(M-2)+2*(N-2)*(M-
2)+N*M+N*M+N*M+        M*(N-2)+N*(M-2)=

N=N1
M=M1
WRITE (*,*) N,M
If(IDEB-1) 5508,5507,5508
5507 pause 5507
5508 continue

If(IDEB-1) 55081,55071,55081
55071 pause 55071
55081 continue
nm=n*m
nm2=2*nm
nstav=33
nb=2*(n-2)
nc=2*(m-2)
nd=2*(n-2)*(m-2)
neq=2*n*m+n*(m-2)+m*(n-2)
np=nm
ns=nm
nx=nm
nq=m*(n-2)
nr=n*(m-2)
nstavf=nstav
nbf=nstav+nb
ncf=nbf+nc
ndf=ncf+nd
npf=ndf+np
nsf=npf+ns
nxf=nsf+nx
nqf=nxf+nq
```

```
nrf=nqf+nr
NM2=2*NM
nqfe=nm2+nq
nrfe=nqfe+nr
NATO=NDF+NP+NS+NX+NQ+NR
write (*,*)'nm',nm,'nm2',nm2,'nstav',nstav,'nb',nb,'nc',nc,'nd',nd
write (*,*) 'neq',neq,'np',np,'ns',ns,'nx',nx,'nq',nq,'nv',nv
write (*,*) 'nstav',nstav,'nbf',nff,'ncf',ncf,'ndf',ndf,'npf',npf
write (*,*) 'nsf',nsf,'nxf',nxf,'nqf',nqf,'nrf',nrf,'nm2',nm2
write (*,*) 'nqfe', nqfe, 'nrfe',nrfe,'NATO',nato

write (8888,*)'nm',nm,'nm2',nm2,'nstav',nstav,'nb',nb,'nc',nc
write (8888,*) 'nd',nd,'neq',neq,'np',np,'ns',ns,'nx',nx,'nq',nq
write (8888,*) 'nv',nv,'nstav',nstav,'nbf',nbf,'ncf',ncf,'ndf',ndf
write (8888,*)'npf',npf ,'nsf',nsf,'nxf',nxf,'nqf',nqf,'nrf',nrf
write (8888,*) 'nm2',nm2,'nqfe', nqfe, 'nrfe',nrfe,'NATO',nato

If(IDEB-1) 9902,9901,9902
9901   pause 9901
9902 continue

8 format(30I5)
write (88888,8)nm,nm2,nstav,nb,nc,nd,neq,np,ns,nx,nq, nv,nstav,nbf
*,ncf,ndf,npf ,nsf,nxf,nqf,nrf,nm2, nqfe, nrfe,nato,n,m
888 FORMAT (8I8)
do 301 j=1,nrf
301 ccc(j)=0.
write(*,*)  'tolf'
write (302,*) ndf
write (88,410) NDF,NEQ,N1TOT,M1TOT,NBF,NCF,NDF,N,M,isens
*,nopox,nopoz

C     PREPARE SIZE OF ARRAY DD (NDF rows NEQ COLUMNS
do 50111 ID=1,NEQ
do 50111 JD=1,NDF
50111 DD(ID,JD)=0
C     end PREPARE SIZE OF ARRAY DD (NDF rows NEQ COLUMNS

C     CALCULATE FF.DAT
IFF(1)=1
do 30111 IFF1=2,ndf
30111 IFF(IFF1)=0.
WRITE (600,30112) (IFF(IFF1),IFF1=1,ndf)
30112 FORMAT(2600I2)
write (88,3013) fac,scale,spax,spaz,SPAX1,SPAZ1,off

DO 3011 oo=2,N-1
xfac=x(oo)
3011 WRITE (88,3013) Xfac
DO 3012 oo=2,M-1
```

```
zfac=z(oo)
3012 WRITE (88,3013) Zfac
do 4012 oo=1,n1tot
4012 xcalfac(oo)=fxcal(oo)/fac
WRITE (88,3013) (XCALfac(oo),oo=1,N1TOT)
write (88,3013) (fxcal(oo),oo=1,n1TOT)
do 4013 oo=1,m1tot
4013 zcalfac(oo)=fzcal(oo)/fac
WRITE (88,3013) (ZCALfac(oo),oo=1,M1TOT)
write (88,3013) (fzcal(oo),oo=1,m1TOT)

3013 FORMAT (200(F12.6))
3014 FORMAT('\')

do 12121 i=1,n1tot
12121 write(601,65527) fxcal(i)

do 12122 i=1,m1tot
12122  write (602,65527)  fzcal(i)

65527 format (100f12.5)

4041 format('f=[')
4042 format ('];')
3021 format (8(e15.6,';'))
write (3,4041)
do 30210 k=1,ndf
30210 ccc(k)=ccc(k)
write (3,3021) (ccc(k),k=1,ndf)
write (3,4042)
do 3025 k=1,nrf
3025 y1(k)=0.
DO 303 J=1,M
DO 303 I=1,N

TI1((I-1)*M+J)=TIf(I,J)
nti((i-1)*m+j)=i
ntj((i-1)*m+j)=j
303 Y1((i-1)*m+j)=Y(I,J)
do 31 j=nm+1,nm2
31 y1(j)=-y1(j-nm)
do 32 j=nm2+1,nrfe
32 y1(j)=0
3022 format (1(e20.11,';'))

C     CHECK IF THERE ARE EQUALITY RESTRICTIONS AND HANDLE AC-
CORDINGLY
IF(TIYES.EQ.0) then
go to 50431
else
```

```
endif
ieq=0
iineq=0
DO 3310 IJ=1,NM
IF(ti1(ij).eq.tival) then
y1eq(ieq+1)=y1(ij)
ieq=ieq+1
ntii(ieq)=nti(ij)
ntjj(ieq)=ntj(ij)
go to 3310
else
endif

y1ineq(iineq+1)=y1(ij)
iineq=iineq+1
ntiii(iineq)=nti(ij)
ntjjj(iineq)=ntj(ij)
3310 continue
nieq=2*ieq
write (200,*) nieq ,'equalities'

do 3410 i=ieq+1,2*ieq
3410 y1eq(i)=-y1eq(i-ieq)

do 341 i=1,2*ieq
341      write (200,*) y1eq(i),ntii(i),ntjj(i)


do 3510 i=iineq+1,2*iineq
3510 y1ineq(i)=-y1ineq(i-iineq)
ib=2*iineq
nineq=ib
ie=ib+n*(m-2)+m*(n-2)
do 3520 i=ib+1,ie
nineq=nineq+1
3520 y1ineq(i)=0.
write (200,*) nineq,'inequalities'


do 351 i=1,ie
351 write (200,*) y1ineq(i)

5043 format('beq=[')
write (3,5043)
write (3,3022) (y1eq(i),i=1,nieq)
write (3,4044)
open (103,file='beq.dat')
write (103,3022) (y1eq(i),i=1,nieq)
write (*,*) (y1eq(i),i=1,nieq)
```

```
40431 continue
4043 format('b=[')
4044 format ('];')
write (3,4043)
write (3,3022) (y1ineq(i),i=1,ie)
write (101,3022) (y1ineq(i),i=1,ie)

If(IDEB-1) 9906,9905,9906
9905   pause 9905
9906 continue

write(*,*) ie,nineq ,nieq
write (3,4044)
go to 50432


50431 continue

write (3,4043)
write (3,3022) (y1(i),i=1,neq)
write (101,3022) (y1(i),i=1,neq)
write (3,4044)
50432 continue

N_KP1=n
M_KP1=m
if (ISENS-1) 33201 ,33201, 33101
33101 continue
go to 33401
33201 n=N_LE
m=M_HE
33401 continue

C     CALCULATION OF R's AND S's
do 33 i=2,n-1
do 33 j=1,m
ADEN=X(I+1)-X(I-1)
A1=(Y(I+1,J)-Y(I,J)) / ( X(I+1)-X(I))
A2= (Y(I,J)-Y(I-1,J)) / ( X(I)-X(I-1))
R(I,J)=2*(A1-A2)/ADEN
write (*,*) Y(I+1,J),Y(I,J),Y(I-1,J), X(I+1),X(I),X(I-1)
write (*,*) i,a1,a2,x(I+1),x(I),ADEN,R(I,J)

33 R(I,J)=2*(A1-A2)/ADEN

do 34 i=1,n
do 34 j=2,m-1
BDEN =Z(J+1)-Z(J-1)
B1= (Y(I,J+1)-Y(I,J)) /  (Z(J+1)-Z(J))
B2= (Y(I,J)-Y(I,J-1)) /    (Z(J)-Z(J-1))
```

```
write (*,*) Y(I,J+1),Y(I,J),Y(I,J-1), Z(J+1),Z(J),Z(J-1)
write (*,*) i,b1,b2,z(j+1),z(j),BDEN
34 S(I,J)=2*(B1-B2)/BDEN

if (isens-1) 3302 ,3302,3301
3301 m=M_KP1
n=N_KP1
go to 3304
3302 continue
m=M_HE
n=N_LE
3304 continue

close (440,status='DELETE')
close (450,status='DELETE')

open (440, file='r.dat')
open (450, file='s.dat')

do 330 i=2,n-1
do 330 j=1,m
330 write (421,331) i,j,r(i,j)


do 440 i=2,n-1
write (440,441) i, (r(i,j),j=1,M)
440 continue
441 format(I5,100E45.20)

do 33011 i=1,n
do 33011 j=2,m-1
33011 write (421,332) i, j,s(i,j)

do 450 i=1,n
write (450,441) i, (s(i,j),j=2,m-1)
450 continue

331 format ('   r(',2i2,')=',E15.5)
332 format ('   s(',2i2,')=',E15.5)

n=N_KP1
m=M_KP1


do 35 i=1,nrfe
do 35 j=1,nrf
DD(i,j)=0.
35 aa(i,j)=0.
```

```
do 40 i=1,nm
aa(i,1)=-1.
aa(i,2)=1.
40 aa(i,3)=-1.
do 50 i=nm+1,nm2
aa(i,1)=-1
aa(i,2)=-1
50 aa(i,3)=1
do 55 i=nm2+1,nrfe
do 55 j=1,5
55 aa(i,j)=0.
DO 70 I=1,N
do 70 J=1,M
aa((I-1)*M+J,5)=-z(J)
70 aa((I-1)*M+J,4)=z(J)
do 80 i=nm+1,nm2
aa(i,4)=-aa(i-nm,4)
80 aa(i,5)=-aa(i-nm,5)
do 90 I=1,N
DO 90 J=1,M
aa((i-1)*M+J,7)=-z(J)**2
90 aa((i-1)*M+J,6)=Z(J)**2

do 100 i=nm+1,nm2
aa(i,6)=-aa(i-nm,6)
100 aa(i,7)=-aa(i-nm,7)

do 110 i=1,N
DO 110 J=1,M
aa((i-1)*M+J,9)=-z(J)**3
110 aa((i-1)*M+J,8)=Z(J)**3

do 120 i=nm+1,nm2
aa(i,8)=-aa(i-nm,8)
120 aa(i,9)=-aa(i-nm,9)

do 121 i=1,N
DO 121 J=1,M
DD((i-1)*M +J,11)=-1
DD((i-1)*M+J,10)=1
aa((i-1)*M +J,11)=-x(i)
121 aa((i-1)*M+J,10)=X(I)

do 122 i=nm+1,nm2
DD(i,11)=-DD(i-nm,11)
DD(i,10)=-DD(i-nm,10)
aa(i,11)=-aa(i-nm,11)
122 aa(i,10)=-aa(i-nm,10)

do 123 i=1,N
```

```
DO 123 J=1,M
DD((i-1)*M+J,13)=-Z(J)
DD((i-1)*M+J,12)=Z(J)
aa((i-1)*M+J,13)=-X(I)*Z(J)
123 aa((i-1)*M+J,12)=X(I)*Z(J)

do 124 i=nm+1,nm2
DD(i,13)=-DD(i-nm,13)
DD(i,12)=-DD(i-nm,12)
aa(i,13)=-aa(i-nm,13)
124 aa(i,12)=-aa(i-nm,12)

do 125 i=1,N
DO 125 J=1,M
DD((i-1)*M+J,15)=-(z(J)**2)
DD((i-1)*M+J,14)=(Z(J)**2)
aa((i-1)*M+J,15)=-(z(J)**2)*x(i)
125 aa((i-1)*M+J,14)=(Z(J)**2)*X(I)

do 126 i=nm+1,nm2
DD(i,15)=-DD(i-nm,15)
DD(i,14)=-DD(i-nm,14)
aa(i,15)=-aa(i-nm,15)
126 aa(i,14)=-aa(i-nm,14)

do 127 i=1,N
DO 127 J=1,M
DD((i-1)*M+J,17)=-(Z(J)**3)
DD((i-1)*M+J,16)=(Z(J)**3)
aa((i-1)*M+J,17)=-(Z(J)**3)*X(i)
127 aa((i-1)*M+J,16)=(Z(J)**3)*X(I)

do 128 i=nm+1,nm2
DD(i,17)=-DD(i-nm,17)
DD(i,16)=-DD(i-nm,16)
aa(i,17)=-aa(i-nm,17)
128 aa(i,16)=-aa(i-nm,16)

do 129 i=1,N
DO 129 J=1,M
DD((i-1)*M+J,19)=-2.*(x(i))
DD((i-1)*M+J,18)= 2.*X(I)
aa((i-1)*M+J,19)=-(x(i)**2)
129 aa((i-1)*M+J,18)=X(I)**2

do 1291 i=nm+1,nm2
DD(i,19)=-DD(i-nm,19)
DD(i,18)=-DD(i-nm,18)
aa(i,19)=-aa(i-nm,19)
1291 aa(i,18)=-aa(i-nm,18)
```

```
do 1292 i=1,N
DO 1292 J=1,M
DD((i-1)*M+J,21)=-2.*(x(i))*z(j)
DD((i-1)*M+J,20)=2.*(X(I))*Z(j)
aa((i-1)*M+J,21)=-(x(i)**2)*z(j)
1292 aa((i-1)*M+J,20)=(X(I)**2)*Z(j)

do 1293 i=nm+1,nm2
DD(i,21)=-DD(i-nm,21)
DD(i,20)=-DD(i-nm,20)
aa(i,21)=-aa(i-nm,21)
1293 aa(i,20)=-aa(i-nm,20)

DO 1294 I=1,N
DO 1294 J=1,M
DD((I-1)*M+J,23)=-2.*(X(I))*(Z(J)**2)
DD((I-1)*M+J,22)=2.*(X(I))*(Z(J)**2)
AA((I-1)*M+J,23)=-(X(I)**2)*(Z(J)**2)
1294 AA((I-1)*M+J,22)=(X(I)**2)*(Z(J)**2)

DO 1295 I=NM+1,NM2
DD(I,23)=-DD(I-NM,23)
DD(I,22)=-DD(I-NM,22)
AA(I,23)=-AA(I-NM,23)
1295 AA(I,22)=-AA(I-NM,22)

DO 1296 I=1,N
DO 1296 J=1,M
DD((I-1)*M+J,25)=-2*(X(I))*(Z(J)**3)
DD((I-1)*M+J,24)=2.*(X(I))*(Z(J)**3)
AA((I-1)*M+J,25)=-(X(I)**2)*(Z(J)**3)
1296 AA((I-1)*M+J,24)=(X(I)**2)*(Z(J)**3)

DO 1297 I=NM+1,NM2
DD(I,25)=-DD(I-NM,25)
DD(I,24)=-DD(I-NM,24)
AA(I,25)=-AA(I-NM,25)
1297 AA(I,24)=-AA(I-NM,24)

DO 1298 I=1,N
DO 1298 J=1,M
DD((I-1)*M+J,27)=-3.*X(I)**2
DD((I-1)*M+J,26)=3.*X(I)**2
AA((I-1)*M+J,27)=-X(I)**3
1298 AA((I-1)*M+J,26)=X(I)**3

DO 1299 I=NM+1,NM2
DD(I,27)=-DD(I-NM,27)
DD(I,26)=-DD(I-NM,26)
```

```
AA(I,27)=-AA(I-NM,27)
1299 AA(I,26)=-AA(I-NM,26)

DO 1301 I=1,N
DO 1301 J=1,M
DD((I-1)*M+J,29)=-3.*(X(I)**2)*Z(J)
DD((I-1)*M+J,28)=3.*(X(I)**2)*Z(J)
AA((I-1)*M+J,29)=-(X(I)**3)*Z(J)
1301 AA((I-1)*M+J,28)=(X(I)**3)*Z(J)

DO 1302 I=NM+1,NM2
DD(I,29)=-DD(I-NM,29)
DD(I,28)=-DD(I-NM,28)
AA(I,29)=-AA(I-NM,29)
1302 AA(I,28)=-AA(I-NM,28)

DO 1303 I=1,N
DO 1303 J=1,M
DD((I-1)*M+J,31)=-3.*(X(I)**2)*(Z(J)**2)
DD((I-1)*M+J,30)=3.*(X(I)**2)*(Z(J)**2)
AA((I-1)*M+J,31)=-(X(I)**3)*(Z(J)**2)
1303 AA((I-1)*M+J,30)=(X(I)**3)*(Z(J)**2)

DO 1304 I=NM+1,NM2
DO 1304 J=1,M
DD(I,31)=-DD(I-NM,31)
DD(I,30)=-DD(I-NM,30)
AA(I,31)=-AA(I-NM,31)
1304 AA(I,30)=-AA(I-NM,30)

DO 1305 I=1,N
DO 1305 J=1,M
DD((I-1)*M+J,33)=-3.*(X(I)**2)*(Z(J)**3)
DD((I-1)*M+J,32)=3.*(X(I)**2)*(Z(J)**3)
AA((I-1)*M+J,33)=-(X(I)**3)*(Z(J)**3)
1305 AA((I-1)*M+J,32)=(X(I)**3)*(Z(J)**3)

DO 1306 I=NM+1,NM2
do 1306 j=1,m
DD(I,33)=-DD(I-NM,33)
DD(I,32)=-DD(I-NM,32)
AA(I,33)=-AA(I-NM,33)
1306 AA(I,32)=-AA(I-NM,32)
continue

C     CALCULATION OF B's

do 1250 i=1,n-2
io=i+2
```

```
do 1250 k=1,m
DO 1250 J=34,NBF,2
NUP=2*io-4
NU=(J/2)-15
IF ((J-33).GT.NUP) GO TO 1250
DD((i+1)*m+K,J)=3.*(x(io)-x(nu))**2.
DD((i+1)*m+k,j+1)=-3.*(x(io)-x(nu))**2.
DD((i+1)*m+k+nm,j)=-3.*(x(io)-x(nu))**2.
DD((i+1)*m+k+nm,j+1)=3.*(x(io)-x(nu))**2.
AA((i+1)*m+K,J)=(x(io)-x(nu))**3.
aa((i+1)*m+k,j+1)=-(x(io)-x(nu))**3.
aa((i+1)*m+k+nm,j)=-(x(io)-x(nu))**3.
aa((i+1)*m+k+nm,j+1)=(x(io)-x(nu))**3.
write(*,*) io,nu,X(io),x(NU)

1250 CONTINUE

go to 1261
do 1260 i=1,n-2
io=i+2

do 1260 k=1,m
DO 1260 J=35,NBF,2
NUP=2*io-4
NU=(J/2)-15
IF ((J-33).GT.NUP) GO TO 1260
DD((i+1)*m+k,J)=-3.*(X(Io)-X(NU))**2.
DD((i+1)*m+k,j+1)=-DD((i+1)*m+k,j)
DD((i+1)*m+k+nm,j)=3.*(x(io)-x(nu))**2.
DD((i+1)*m+k+nm,j+1)=-3.*(x(io)-x(nu))**2.
AA((i+1)*m+k,J)=-(X(Io)-X(NU))**3.
aa((i+1)*m+k,j+1)=-aa((i+1)*m+k,j)
aa((i+1)*m+k+nm,j)=(x(io)-x(nu))**3.
aa((i+1)*m+k+nm,j+1)=-(x(io)-x(nu))**3.

1260 CONTINUE
1261 continue

C     CALCULATIONS OF C's
do 1270 i=1,n
do 1270 j=1,m
io=(i-1)*m+j
if (j.lt.3) go to 1268
continue
do 1269 k=1,j-2
DD((i-1)*m+j,nbf+2*(k-1)+1)=0.
DD((i-1)*m+j+nm,nbf+2*(k-1)+1)=0.
DD((i-1)*m+j,nbf+2*(k-1)+2)=0.
DD((i-1)*m+j+nm,nbf+2*(k-1)+2)=0.
aa((i-1)*m+j,nbf+2*(k-1)+1)=((z(j)-z(k+1))**3)
```

```
aa((i-1)*m+j+nm,nbf+2*(k-1)+1)=-((z(j)-z(k+1))**3)
aa((i-1)*m+j,nbf+2*(k-1)+2)=-((z(j)-z(k+1))**3)
aa((i-1)*m+j+nm,nbf+2*(k-1)+2)=((z(j)-z(k+1))**3)
1269 continue
1268 continue
1270 continue


C     CALCULATION OF C's FOR SECOND DIFFERENCES z direction
do 1275 i=1,n
do 1275 j=3,m-1
in=(i-1)*(m-2)+j-1+nqfe
do 1273 k=1,j-2
inj=nbf+2*(k-1)+1
write (22,8877) z(j),z(k+1),s(i,j),i,j,k
8877 format (3f10.3,3i5)
aa(in,inj)=-6.*(z(j)-z(k+1))*s(i,j)
aa(in,inj+1)=-aa(in,inj)
1273 continue
1274 continue
1275 continue

C     CALCULATION OF B's FOR SECOND DIFFERENCES X direction
DO 1289 I=3,N
DO 1289 J=1,M
IN=NM2+(I-2)*M+J
DO 1287 K=1,I-2
INJ=33+2*(K-1)+1
AA(IN,INJ)=-6.*(X(i)-X(K+1))*R(I,J)
AA(IN,INJ+1)=-AA(IN,INJ)
1287 CONTINUE
1288 CONTINUE
1289 CONTINUE
1453 continue


C     CALCULATIONS   OF D's
do 1280 i=3,n
do 1280 j=3,m
do 1280 kk=1,i-2
do 1279 k=1,j-2
ini=(i-1)*m+j
inj=ncf+1+2*((kk-1)*(m-2)+(k-1))
innm=ini+nm
DD(ini,inj)= 3.*(((x(i)-x(kk+1))**2)*((z(j)-z(k+1))**3))
DD(innm,inj)=-3.* (((x(i)-x(kk+1))**2)*((z(j)-z(k+1))**3))
DD(ini,inj+1)=-DD(ini,inj)
DD(innm,inj+1)=-DD(innm,inj)

aa(ini,inj)= (((x(i)-x(kk+1))**3)*((z(j)-z(k+1))**3))
```

```
aa(innm,inj)=- (((x(i)-x(kk+1))**3)*((z(j)-z(k+1))**3))
aa(ini,inj+1)=-aa(ini,inj)
aa(innm,inj+1)=-aa(innm,inj)

1279 continue
1278 continue
1280 continue


C     COEFFICIENTS ROR D's FOR SECOND DIFERRENCES-FIRST PART IN x
DIRECTION
DO 1284 I=3,n
DO 1284 J=3,M
DO 1284 KK=1,I-2
DO 1281 K=1,J-2
ini=(i-2)*m+j+nm2
inj=(ncf+1)+2*(kk-1)*(m-2)+2*(k-1)
11111 format(7i5,7e10.3)

AA(INI,INJ)=-6.*(((X(I)-X(KK+1)))*((Z(J)-Z(K+1))**3))*R(I,J)
AA(INI,INJ+1)=-AA(INI,INJ)
write(200,11111) ncf+1,i,j,kk,k,ini,inj,x(i),x(kk+1),z(j),z(k+1)
*,aa(ini,inj),aa(ini,inj+1),r(i,j)

1281 CONTINUE
1284 CONTINUE
write (*,*) 'D;s in X direction' , INI,INJ,INJ+1

If(IDEB-1) 9908,9907,9908
9907   pause 9907
9908 continue


C     COEFFICIENTS FOR D's FOR SECOND DIFFERENCES-SECOND PART in
Z DIRECTION
DO 1384 I=3,N
DO 1384 J=3,M
DO 1384 KK=1,I-2
DO 1384 K=1,J-2

ini=(i-1)*(m-2)+j-1 +nqfe

inj=ncf+1+2*((kk-1)*(m-2)+(k-1))

AA(INI,INJ)=-6.*(((Z(J)-Z(K+1)))*((X(I)-X(KK+1))**3))*S(I,J)
1384 AA(INI,INJ+1)=-AA(INI,INJ)
write (*,*) 'D;s in Z direction' , INI,INJ,INJ+1
If(IDEB-1) 9910,9909,9910
9909   pause 9909
9910 continue
```

```
C     calculations of remaining coefficients for second differeces equations

C     A COEFFICIENTS FOR X SECOND DERRIVATIVE
do 1285 i=2,n-1
do 1285 j=1,m
in=nm2+(i-2)*m+j
aa(in,18)=-2.*r(i,j)
1285 aa(in,19)=-aa(in,18)
do 1290 i=2,n-1
do 1290 j=1,m
aa(nm2+(i-2)*m+j,20)=-z(j)*r(i,j)*2.
1290 aa(nm2+(i-2)*m+j,21)=z(j)*r(i,j)*2.
do 1310 i=2,n-1
do 1310 j=1,m
aa(nm2+(i-2)*m+j,22)=-(z(j)**2)*r(i,j)*2.
1310 aa(nm2+(i-2)*m+j,23)=(z(j)**2)*r(i,j)*2.
do 1320 i=2,n-1
do 1320 j=1,m
in=nm2+(i-2)*m+j
aa(in,24)=-(z(j)**3)*r(i,j)*2.
1320 aa(in,25)=(z(j)**3)*r(i,j)*2.
do 1330 i=2,n-1
do 1330 j=1,m
in=nm2+(i-2)*m+j
aa(in,26)=-6.*x(i)*r(i,j)
1330 aa(in,27)=6.*x(i)*r(i,j)
do 1340 i=2,n-1
do 1340 j=1,m
in=nm2+(i-2)*m+j
aa(in,28)=-6.*x(i)*z(j)*r(i,j)
1340 aa(in,29)=-aa(in,28)
do 1350 i=2,n-1
do 1350 j=1,m
in=nm2+(i-2)*m+j
aa(in,30)=-6.*x(i)*(z(j)**2)*r(i,j)
1350 aa(in,31)=-aa(in,30)
do 1360 i=2,n-1
do 1360 j=1,m
in=nm2+(i-2)*m+j
aa(in,32)=-6.*x(i)*(z(j)**3)*r(i,j)
1360 aa(in,33)=-aa(in,32)
C     end A COEFFICIENTS FOR X SECOND DERRIVATIVE

C     A COEFFICIENTS FOR Z SECOND DERRIVATIVE
do 1370 i=1,n
do 1370 j=2,m-1
in =nqfe+(i-1)*(m-2)+j-1
aa(in,6)=-2.*s(i,j)
1370 aa(in,7)=-aa(in,6)
```

```
do 1380 i=1,n
do 1380 j=2,m-1
in=nqfe+(i-1)*(m-2)+j-1
aa(in,8)=-6.*z(j)*s(i,j)
1380 aa(in,9)=-aa(in,8)
do 1390 i=1,n
do 1390 j=2,m-1
in=nqfe+(i-1)*(m-2)+j-1
aa(in,14)=-2.*x(i)*s(i,j)
1390 aa(in,15)=-aa(in,14)
do 1400 i=1,n
do 1400 j=2,m-1
in=nqfe+(i-1)*(m-2)+j-1
aa(in,16)=-6.*x(i)*z(j)*s(i,j)
1400 aa(in,17)=-aa(in,16)
do 1410 i=1,n
do 1410 j=2,m-1
in=nqfe+(i-1)*(m-2)+j-1
aa(in,22)=-2.*(x(i)**2)*s(i,j)
1410 aa(in,23)=-aa(in,22)
do 1420 i=1,n
do 1420 j=2,m-1
in=nqfe+(i-1)*(m-2)+j-1
aa(in,24)=-6.*(x(i)**2)*z(j)*s(i,j)
1420 aa(in,25)=-aa(in,24)
do 1430 i=1,n
do 1430 j=2,m-1
in =nqfe+(i-1)*(m-2)+j-1
aa(in,30)=-2.*(x(i)**3)*s(i,j)
1430 aa(in,31)=-aa(in,30)
do 1440 i=1,n
do 1440 j=2,m-1
in=nqfe+(i-1)*(m-2)+j-1
aa(in,32)=-6.*(x(i)**3)*(z(j))*s(i,j)
1440 aa(in,33)=-aa(in,32)
C      end A  COEFFICIENTS FOR Z SECOND DERRIVATIVE

go to 1461

C     COEFFICIENS OF P
DO 1450 I=1,NM
DO 1450 J=NDF+1,NPF
IF (I.EQ.(J-NDF)) AA(I,J)=-1.
1450 CONTINUE

C     COEFFICIENS OF UNIT MATRIX
DO 1460 I=1,NRFE
DO 1460 J=NPF+1,NRF
IF (I.EQ.(J-NPF)) AA(I,J)=1.
1460 CONTINUE
```

```
1461 continue

IF (TIYES.EQ.0) THEN
C     PREPARE A IF THERE ARE ONLY INEQUALITIES
1133 format ('A=[')
write (3,1133)
1131 format (8(f10.3,';'))
2231 format (25000(e35.27 )/';')

do 1130 i=1,neq
write (3,2231) (aa(i,j),j=1,NdF)
write (100,2231) (aa(i,j),j=1,ndf)
write (2001,2231) (DD(i,j),j=1,ndf)
1130 write (3,2232)


fo=1e-04
write (3,1132) ndf,fo
2232 format (';')

1132 format (2h];/'lb=zeros(',i3,',1);'/7hAeq=[];/7hbeq=[];/
*'format long'/
*'[x,fval,exitflag,output,lambda]='
*28hlinprog(f,A,b,Aeq,beq,lb,[],
*40hoptimset('Display','iter','maxiter',999,
*26h'LargeScale','off','TolX',,e10.1,'))'/
*'save mytest.dat x -ASCII -double')
WRITE (22,499) (XCAL(I),I=1,N1)
WRITE (22,499) (ZCAL(J),J=1,M1)

write (22,410) N,M,N1,M1,NSTAVF,NBF,NCF,NDF,NPF,NSF,NXF,NQF,NRF
write (22,410) NM,NM2,NQFE,NRFE
400 FORMAT (200f10.3)
499 format (200E12.5)
498 format (200f8.3)
410 FORMAT (20I8)
500 FORMAT (15H END OF INPUT'E)
6060 continue
write(102,*) n,m
write(102,400) (x(i),i=1,n)
write(102,400) (z(j),j=1,m)
GO TO 8000
ENDIF
C     PREPARE A IF THERE ARE TIYES EQUALITIES AND TINO INEQUALI-
TIES
C     FIRST HANDLE EQUALITIES
11333 format ('Aeq=[')
write (3,11333)
zer=0.
noii=0
```

```
do 6070 i=1,n
do 6070 j=1,m
do 6070 ij=1,ieq
if(  (ntii(ij).eq.i). and. (ntjj(ij).eq.j) ) then
continue
write (200,*) i,j,ij
ii=(i-1)*m+j
write (3,2231) zer,(aa(ii,jj),jj=2,ndf)
write (110,2231 ) zer,(aa(ii,jj),jj=2,ndf)

write (3,2232)

60701 continue

go to 6070
endif
continue
6070 continue

do 6080 i=1,n
do 6080 j=1,m
do 6080 ij=1,ieq
if(  (ntii(ij).eq.i). and. (ntjj(ij).eq.j) ) then
continue
write (200,*) i,j,ij
ii=(i-1)*m+j

write (3,2231) zer,(aa(ii+nm,jj),jj=2,ndf)
write (3,2232)
write (110,2231) zer,(aa(ii+nm,jj),jj=2,ndf)
go to 6080
endif
continue
6080 continue
6081 FORMAT (';')
fo=1e-04
WRITE (3,6081)

C     THEN HANDLE INEQUALITIES
write (3,1133)
do 6090 i=1,n
do 6090 j=1,m
do 6090 ij=1,iineq
if(  (ntiii(ij).eq.i).and. (ntjjj(ij).eq.j) ) then
continue
write (200,*) i,j,ij
ii=(i-1)*m+j
write (3,2231) (aa(ii,jj),jj=1,ndf)
write (100,2231) (aa(ii,jj),jj=1,ndf)
```

```
write (2001,2231) (DD(ii,jj),jj=1,ndf)
write (3,2232)

go to 6090
endif
continue
6090 continue

do 6095 i=1,n
do 6095 j=1,m
do 6095 ij=1,iineq
if(  (ntiii(ij).eq.i).and. (ntjjj(ij).eq.j) ) then
continue
else
write (200,*) i,j,ij
ii=(i-1)*m+j

write (3,2231) (aa(ii+nm,jj),jj=1,ndf)
write (100,2231) (aa(ii+nm,jj),jj=1,ndf)
write (2001,2231) (DD(ii+nm,jj),jj=1,ndf)

write (3,2232)
go to 6095
endif
continue
6095 continue

C         NOW HANDLE SECOND DERRIVATIVE INEQUALITIES
do  6100 i=2*N*M+1,NRFE
write (3,2231) (aa(i,j),j=1,NdF)
write (100,2231) (aa(i,j),j=1,ndf)
write (2001,2231) (DD(i,j),j=1,ndf)
6100 write (3,2232)
WRITE(3,6081)


C     GIVE SPECIAL MATLAB ORDERS
6300 format ('lb=zeros(',i3,',1);'/
*'format long'/
*'[x,fval,exitflag,output,lambda]='
*28hlinprog(f,A,b,Aeq,beq,lb,[],


*40hoptimset('Display','iter','maxiter',999,
*26h'LargeScale','off','TolX',,e10.1,'))'/
*'save mytest.dat x -ASCII -double')
WRITE (3,6300) ndf,fo
8000 continue
write (*,*) ' please wait for matlab to prepare file MYTEST.DAT'
ep=engopen('matlabr')
```

```
T = mxcreatefull(1,ndf,0)
call mxsetname(T,'f')
call mxcopyreal8toptr(ccc,mxgetpr(T),ndf)
status = engputmatrix(ep,T)


nio=400
iibb=0
iiaa=0
do 11131 i=1,nrfe
do 11131 j=1,ndf
iaaa=(i-1)*n+j
aaf(iaaa)=aa(i,j)
if (aaf(iaaa).ne.0) then
iiaa=iiaa+1
write (4000,*) aaf(iaaa)
else
iibb=iibb+1
endif
continue
11131 continue

T2 = mxcreatefull(1,400,0)
call mxsetname(T2,'A')
call mxcopyreal8toptr(aaf,mxgetpr(T2),400)
status = engputmatrix(ep,T2)

if (status.ne.0) then
write (6,*) 'engputmatrix failed'
stop
endif

nio=400
iibb=0
iiaa=0
do 12131 i=1,nrfe
do 12131 j=1,ndf
iaaa=(i-1)*n+j
aaq(iaaa)=aa(i,j)
if (aaq(iaaa).ne.0) then
iiaa=iiaa+1
write (5000,*) aaq(iaaa)
else
iibb=iibb+1
endif
continue
12131 continue


T22 = mxcreatefull(1,200,0)
```

```
call mxsetname(T22,'AEQ')
call mxcopyreal8toptr(aaq,mxgetpr(T22),200)
status = engputmatrix(ep,T22)
if (status.ne.0) then
write (6,*) 'engputmatrix failed'
stop
endif

t22=engevalstring(ep,'load c:\naval\fair.bak8\ndf.dat')

if (TIYES.EQ.0) then
C     HANDLE INEQUALITY CONSTRAINTS ONLY
write (*,*) 'INEQUALITY CONTRAINTS ONLY'
t21=engevalstring(ep,'load c:\naval\fair.bak8\A.dat;')

T1 = mxcreatefull(1,neq,0)
call mxsetname(T1,'b')
call mxcopyreal8toptr(y1,mxgetpr(T1),neq)
status = engputmatrix(ep,T1)
ti1501=engevalstring(ep,'AB=[]')
ti1502=engevalstring(ep,'beq=[]')



write (*,*) 'EVALUATE fair8 CALLING  BREADTHS8_FAIR8_CORRECT'
write (*,*) 'or BREADTHS8_FAIR8_CORREC8GGG if NO_EQUAT>500'

NO_EQUAT=2*M*N+N*(M-2)+M*(N-2)
WRITE (*,*) 'TOTAL NO OF EQUATIONS=',  NO_EQUAT
write (705,*) NO_EQUAT


t20=engevalstring(ep,'fair8')

c     END EVALUATE breadths8_fair8

endif
C     HANDLE BOTH INEQUALITIES AND QUALITIES
C     FIRST EQUALITIES
write (*,*) 'BOTH INEQUALITY AND QUALITY CONTRAINTS '
t33=engevalstring(ep,'load c:\naval\fair.bak8\AB.dat;')

T11 = mxcreatefull(1,nieq,0)
call mxsetname(T11,'beq')
call mxcopyreal8toptr(y1eq,mxgetpr(T11),nieq)
status = engputmatrix(ep,T11)
ti1400=engevalstring(ep,'load c:\naval\fair.bak8\b.dat;')
ti1401=engevalstring(ep,'load c:\naval\fair.bak8\beq.dat;')
```

```
If(IDEB-1) 9912,9911,9912
9911   pause 9911
9912 continue
C         THEN INEQUALITIES
t21=engevalstring(ep,'load c:\naval\fair.bak8\A.dat;')

T1 = mxcreatefull(1,nineq,0)
call mxsetname(T1,'b')
call mxcopyreal8toptr(y1ineq,mxgetpr(T1),nineq)
status = engputmatrix(ep,T1)
ti1400=engevalstring(ep,'load c:\naval\fair.bak8\b.dat;')
ti1401=engevalstring(ep,'load c:\naval\fair.bak8\beq.dat;')
continue
77777 continue
If(IDEB-1) 9914,9913,9914
9913   pause 9913
9914 continue

77778 continue
If(IDEB-1) 99161,99151,99161
99151   pause 99151
99161 continue

open (7777,file='fval.dat')
read (7777,*) fva

fva1=fva*fac
write (*,*) fva*fac
write (*,*) 'MAX POINTS DEVIATION  IS   ',fva1,'   METERS'
write (1000,*) 'MAX POINTS DEVIATION  IS   ',fva1,'   METERS'
write (1000,*) 'MAX ALLOWED DEVIATION  IS   ',TOLF,'   METERS'

error=-fva*fac+tolf
write (*,*) error,fva*fac,tolf
open (77781,file='error.dat')
write (77781,*) error,fva*fac,tolf
rewind 77781

open (7778,file='exitflag.dat')
read (7778,*) ex
write (*,*) 'EXITFLAG  IS   ',ex
write (1000,*) 'EXITFLAG  IS   ',ex

if (ex) 77781,77782,77783

77781 write (*,*) 'OPTIMIZATION NOT CONVERGING'
write (1000,*) 'OPTIMIZATION NOT CONVERGING'
go to 77784
77782 write (*,*) 'OPTIMIZATION CYCLES NOT SUFFICIENT'
write (1000,*) 'OPTIMIZATION CYCLES NOT SUFFICIENT'
```

```
go to 77784
77783  write (*,*) 'OPTIMIZATION CONVERGES'
write (1000,*) 'OPTIMIZATION CONVERGES'
77784 write (*,*) '!!!BE CAREFULL IF EXITFLAG IS 0 OR -1'

status= engClose(ep)

do 99998 j=1,m
write (501,45016) (dddd1(jo,j)*scale,jo=1,n)
write (502,45016) (fy(jo,j)*scale,jo=1,n)
99998 continue
do 99997 j=1,m
99997 write (500,45016) fz(j)*scale
91917 continue
ep=engopen('matlab')
ti1400=engevalstring(ep,'load c:\naval\fair.bak8\ddgiven400.dat;')
ti1401=engevalstring(ep,'load c:\naval\fair.bak8\zzgiven400.dat;')
ti1402=engevalstring(ep,'load c:\naval\fair.bak8\dddd1400.dat;')

45016 format(50f12.2)
45000 CONTINUE
IF (I_EXIT.GT.1) THEN
WRITE (*,*)'!!!FINAL PROCESS WILL BE STOPED DUE TO INCMPATIBLITY'
WRITE (*,*)'!!!PLEASE SEE INCOMPATIBILITIES IN  FILE detail.dat '

ELSE
CONTINUE
ENDIF
46000  if (I_PLOT-1) 46002, 46001, 46002
46001 WRITE(*,*) 'LENGTH OF PLATE NOT ENOUGH TO CUT LARGER
FRAME MODEL'
WRITE(*,*) 'MIN PLATE LENGTH SHOULD BE LARGER THAN', '    METERS'
46002 continue
close (1000,status='KEEP')

stop
END
```

ANNEX(2) + PLATE(2)
DOCUMENTATION OF FORTRAN SUBROUTINE
“chckpts_fmatr_no_debug_mad.for”

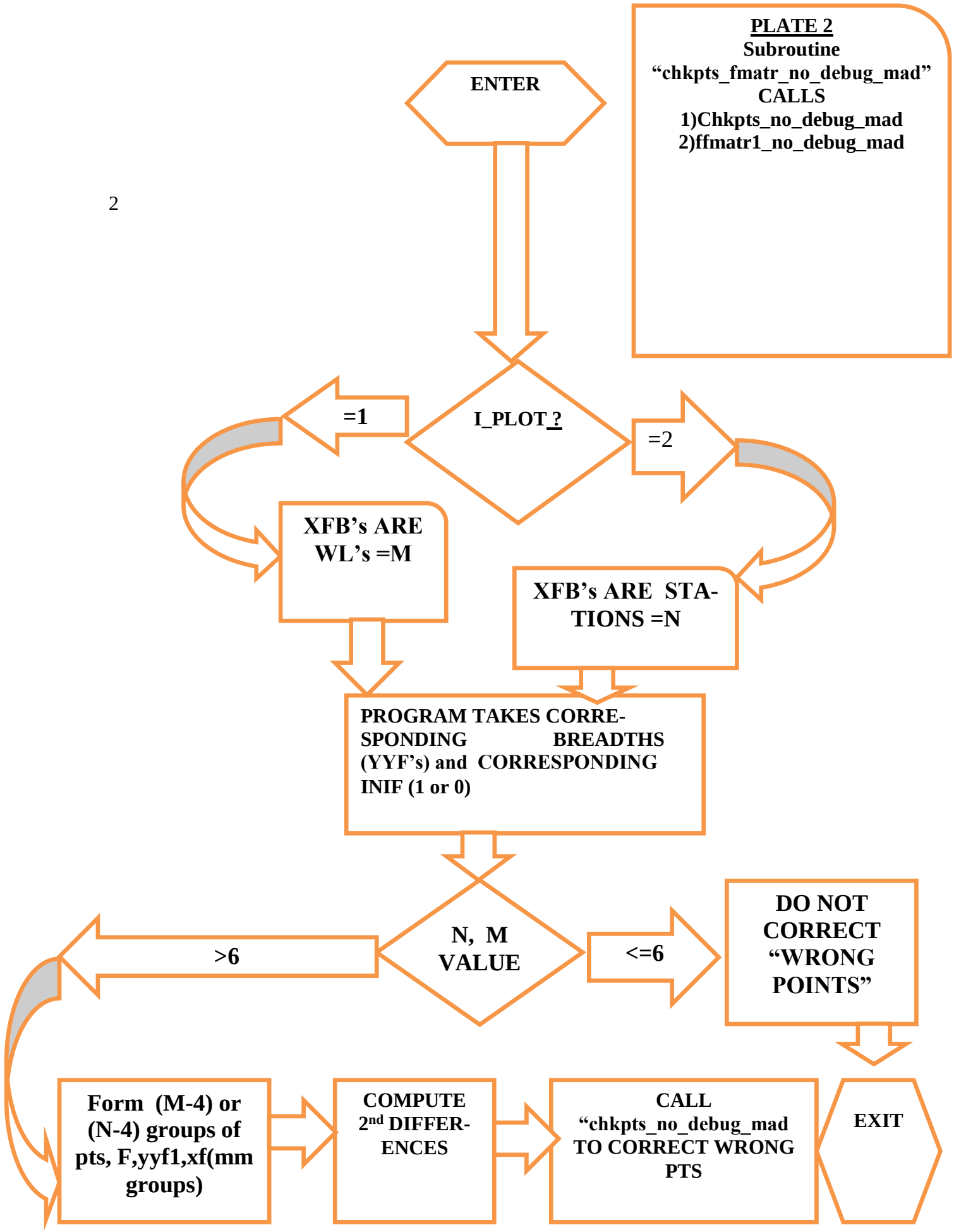


PLATE 2
Subroutine
"chkpts_fmatr_no_debug_mad"
CALLS
1)Chkpts_no_debug_mad
2)ffmatr1_no_debug_mad
ENTER
I_PLOT ?
=1
=2
XFB's ARE WL's =M
XFB's ARE STA-TIONS =N
PROGRAM TAKES CORRE-SPONDING BREADTHS (YYF's) and CORRESPONDING INIF (1 or 0)
N, M VALUE
>6
<=6
DO NOT CORRECT "WRONG POINTS"
Form (M-4) or (N-4) groups of pts, F,yyf1,xf(mm groups)
COMPUTE 2nd DIFFER-ENCES
CALL "chkpts_no_debug_mad TO CORRECT WRONG PTS
EXIT

FORTRAN SUBROUTINE "chkpts_fmatr_no_debug.for"

```
      SUBROUTINE chkpts_fmatr(m,n,xf,f,fy,FZO,jn,dddd1,nio,
     *I_EXIT,i_plot, FXCAL,FZCAL,N1,M1,FX,TIF,n_sav,m_sav,ISENS,
     *N_LE,M_HE,IDEB,LIO,H)

     IMPLICIT REAL*8(A-H,O-Z)
          CHARACTER*1 TIV,TIVAL,TIF(50,50)
C     PROCEED AND CHECK TWO DIMENSIONAL FAIRING
c      if(nio.eq.2) then  go to 81818
c          else
c          endif
      dimension dio(500),H(7),yyF(800),XF(800),
     *xXCALF(800),yyf_2dif(80),H_2dif(500),finit_2dif(80),H_stat(500)
     *,finit1(7),yyf1(800),F(7),iniF(800),yyf2(800),xfb(800),FY_IN(500)
     *,FZ(500),FY(500,500),x(80),yy(80),XXCAL(80),DDDD(80),
     *dddd1(500,500),dddd2(500,500),dddd3(500,500),I_WRO(100),finit(500)
      dimension ddc(500),FZO(500,500),FX(500),FXO(500,500),FXCAL(80),
     * FZCAL(80),FX_IN(500),che(80),che1(80,80),val(80),val1(80,80)

      dimension val_r(80),val1_r(80,80),che_r(80),che_r1(80,80)

          IF(LIO.ne.1) then
          n=n1
      else
          endif

      if(nio.eq.2) then
          go to 81818
          else
          endif

          write (*,*) 'entered CHKPTS_FMATR'
      write (*,*) 'entered CHKPTS_FMATR valuew N1 M1',N1,M1
      write (*,*) m,n,i_plot,N_sav,M_sav

          If(IDEB-1) 7702,7701,7702
 7701 pause 7701
 7702 continue

      TIVAL='*'

      Do  25 II=1,n
   25 write (*,*) (fy(II,JJ),JJ=1,m)

          If(IDEB-1) 7704,7703,7704
 7703 pause 7703
 7704 continue
      Do  26 II=1,n
   26 write (*,*) (FZO(II,JJ),JJ=1,m)
```

```
          If(IDEB-1) 7706,7705,7706
 7705 pause 7705
 7706 continue

     NF=m

          MF=M
           TIV='*'

     IF (ISENS-1) 3012,3012,3015

 3012 continue
     if (I_PLOT-1) 3013,3014,3013
 3014   N_KP=N
     go to 3015

 3013 continue
     N=M_HE
 3015 continue
     write(*,*) 'I_PLOT Is ' ,I_PLOT, 'ISENS IS ',ISENS
     write(*,*)  'N is    ',N
          If(IDEB-1) 7708,7707,7708
 7707 pause 7707
 7708 continue


     DO 30 J=1,N
          J_NO_SAV=J
     WRITE (1000,*) '    '
     WRITE (*,*) 'POSITION -STATION OR WL NO    ', J

          If(IDEB-1) 7710,7709,7710
 7709 pause 7709
 7710 continue

C     PREPARE ORIGINAL OFFSETS FOR MPLOT
     DO 22 OO=1,M
          FY_IN(OO)=FY(J,OO)
   22 CONTINUE
C     END PREPARE ORIGINAL OFFSETS FOR MPLOT

C     PREPARE ORIGINAL HEIGHTS FOR MPLOT
     DO 221 OO=1,M

          FX_IN(OO)=FZO(J,OO)
  221 CONTINUE
C     END PREPARE ORIGINAL HEIGHTS FOR MPLOT
          JN=J
     itess=0
```

```
          IF(LIO.ne.1) then
          M=M1
          else
          endif

          DO 1 JIO=1,M
          write (*,*) I_PLOT
          if (i_plot-1) 12,11,12

   11     XFb(JIO)=FZO(J,JIO)

       go to 13

   12 CONTINUE

       do 121 JIO1=1,N
       XFb(JIO1)=FZO(jio1,1)
  121 continue
   13     write (*,*) 'XFB(jio)',XFB(jio),N

          If(IDEB-1) 7712,7711,7712

 7711 pause 7711
 7712 continue

       if ((LIO.ne.1).and.(I_PLOT.eq.2)) then

          YYF(JIO)=fy(JIO,J)
C      YYF(JIO)=fy(J,JIO)
       yyf2(jio)=FY(JIO,j)
       write (*,*) JIO,J,yyf2(jio)
c         pause 81
          else

          YYF(JIO)=FY(J,JIO)
          yyf2(jio)=FY(J,JIO)
          write (*,*) J,JIO,yyf2(jio)
c         pause 80
          endif


       if (i_plot-1) 81,80,81

   80 XXCALF(JIO)=FZO(J,JIO)
       go to 87
   81     XXCALF(JIO)=FZO(JIO,J)
   87 continue

       write (*,*) 'XXCALF(JIO)', XXCALF(JIO)
```

```
          WRITE (*,*) j,jio,TIF(J,JIO),'TIVAL is',tival

          If(IDEB-1) 7714,7713,7714
 7713 pause 7713
 7714 continue

          IF (TIF(J,JIO).EQ.TIVAL) THEN

          INIF(JIO)=1
          GO TO 96
          ENDIF
          INIF(JIO)=0
  96 CONTINUE

 3031 format (2hno,i5,7h  round)
   1      CONTINUE
     write (*,*) M

      If(IDEB-1) 7716,7715,7716
 7715 pause 7715
 7716 continue

 c         CHECK QUALITY OF DATA-CORRECT IF NECESSARY
 81817 continue

      ICON=1
 C     THIS DO (DO 1112 WORKS ONLY IF M>6
 C     IF M<6 CORRECTION IN DATA QUALITY IS NOT CARRIED OUT


      if ((LIO.ne.1).and.(I_PLOT.ne.1 )) then
          M=M1
          else
          endif

      DO 1112 mm=1,M-6
          no=mm
       write (*,*)  'CHECK AND CORRECTION OF  DATA QUALITY '
          write (*,*)  'no of group in 5 points set in CHKPTS_FMATR is   ',no

          If(IDEB-1) 7718,7717,7718
 7717 pause 7717
 7718 continue

          write(*,*) 'H',H(1),H(2),H(3),H(4),H(5),H(6),H(7)
      write (*,*) 'mm is',mm

      if (mm.GT.1)        THEN
          DO 1114         KK=1,6
      XF(MM+KK-1)=xxcalf(KK+1)
```

```
1114 yyf(MM+KK-1)=H(KK+1)

        ELSE
     CONTINUE
     ENDIF

        DO 1111 nn=1,7

        F(NN)=yyf(mm+nn-1)

        yyf1(nn)=yyf2(mm+nn-1)

1111 XF(NN)=xxcalf(mm+nn-1);

     write (*,*) 'XF=',XF(1),XF(2),XF(3),XF(4),XF(5),XF(6),XF(7)

     DO 700 I=2,6

        A=(yyF1(I+1)-yyF1(I))/(XF(I+1)-XF(I))
        A1=(yyF1(I)-yyF1(I-1))/(XF(I)-XF(I-1))
        C=2./(XF(I+1)-XF(I-1))

     DD=C*(A-A1)

 700 finit1(i)=DD

        If(IDEB-1) 7720,7719,7720
7719 pause 7719
7720 continue

          If(IDEB-1) 77201,77191,77201
77191 pause 77191
77201 continue

     write (*,*) 'f=',F(1),F(2),F(3),F(4),F(5),F(6),F(7)
     write (*,*) 'XF=',XF(1),XF(2),XF(3),XF(4),XF(5),XF(6),XF(7)
     I_WRONG=0

     do 7011 njio=1,3
c       do 7011 njio=1,5

        write(*,*) 'njio   ',njio
        write(*,*) 'F=',(F(KII),KII=1,22)
     write(*,*) 'XF=',(XF(KII),KII=1,22)
c       pause 7720
     CALL CHKPTS(XF,F,7,H,iTEST,H_2DIF,finit,ddc,no,JN,I_WRONG
    *     ,i_plot,N_LE,IDEB,LIO)
        write (*,*) 'RETURN TO CHKPTS_FMATR FROM CHKPTS'
        write(*,*) (H(KII),KII=1,10)
c       pause 7011
```

```
        If(IDEB-1) 7722,7721,7722
 7721 pause 7721
 7722 continue

        do 7012 mjio=1,7
 7012   f(mjio)=h(mjio)
 7011 continue

        I_WRO(mm)=I_WRONG
        If(IDEB-1) 7724,7723,7724
 7723 pause 7723
        pause 55555
        pause 55555
        pause 55555
 7724 continue


    I_WRO(mm)=I_WRONG

    write (*,*) 'I_WRO(mm)=    ',((I_WRO(Imm)),imm=1,NO),'NO',NO


C     COLLECT DATA FOR CORRECTED OFFSETS OF STATION
    DO 701 NNH=1,7
    H_STAT(mm+NNH-1)= H(NNH)
        WRITE (*,*) H(NNH),H_STAT(mm+NNH-1),mm+NNH-1
  701 continue
C      end COLLECT DATA FOR CORRECTED OFFSETS OF STATION

        itess=itess+itest

                If(IDEB-1) 7726,7725,7726
 7725 pause 7725
 7726 continue


        write (*,*) 'itest=', itest, 'itess=', itess
        do 11121 nnn=1,5


    finit_2dif(icon+NNn)=finit1(nnn+1)


    yyf_2dif(ICON+NNn)=H_2DIF(nnn)

11121 write (*,*) icon+nnn, nnn,finit_2dif(ICON+NNn),
   *yyf_2dif(ICON+NNn)

    ICON=ICON+1
```

```
      DO 1115 LL=1,7
          write(*,*) LL, 'LL IS'

      XF(MM+LL-1)=xxcalf(LL)
      yyf(MM+LL-1)=H(LL)

          write (*,*) XF(MM+LL-1), yyf(MM+LL-1),MM+LL-1,MM,LL,M


          write (*,*) (H(RR),RR=1,7)

 1115 continue

 1112 continue

          If(IDEB-1) 7728,7727,7728
 7727 pause 7727
 7728 continue
                   If(IDEB-1) 7730,7729,7730
 7729 pause 7729
 7730 continue

      write (1000,*)  '2ND DIFF AS GIVEN    2ND DIFF AS CORRECTED'
     *, '      INDEX'

          write (1000,*)  '-----------------    -------------------- '
     *, '   --------'
      write (*,*) 'yyf_2dif-INITIAL 2ND DIF   finit_2dif-FINAL 2ND DIF'

          do 11151 N_DIF=2,m-1
          IF ((finit_2dif(N_DIF)-yyf_2dif(N_DIF)).NE.0) THEN
      write (1000,*) finit_2dif(N_DIF) ,yyf_2dif(N_DIF),'   *  '
          ELSE
      write (1000,*) finit_2dif(N_DIF) ,yyf_2dif(N_DIF)
          ENDIF
 11151  write (*,*) finit_2dif(N_DIF),yyf_2dif(N_DIF)

      GO TO 11182

 C        CHECK FOR INCOMPATIBILITY
      DO 11181 I_WR=1,no
          IF (I_WRO(I_WR).EQ.1) THEN
  1002 FORMAT (A26)
      WRITE (1000,1002) '!!!!!INCOMBATIBLE DATA!!!!!'
  1003 FORMAT (A47)
      WRITE  (1000,1003)  '!!!CHECK  ICOMPATIBLE  STATIONS  AND  REPEAT
 !!!!!'
  1004 FORMAT (A47)
      WRITE (1000,1004) '!!!SEE FILE details.dat !!!!!'
          I_EXIT =I_EXIT+1
```

```
          GO TO 81819
          ELSE
          CONTINUE
          ENDIF
11181 continue
c      end CHECK FOR INCOMPATIBILITY

11182 continue
          If(IDEB-1) 7732,7731,7732
 7731 pause 7731
 7732 continue

C     bY PASS 2D FAIRING IF NIO IS 2
        if(nio.eq.2) then
C     WRITE RESULTS OF DATA CORRECTION IN DETAILS.DAT
      if(iTESs.gt.0) then
      write (1000,*) '   '
      write (1000,*) ('DATA CORRECTED-SEE BELOW')
      WRITE (1000,*) ('------------------------')
      WRITE (1000,*) ('WATERLINE  SUBM-HBRS
     * CORR-HBRS CORR+FAIRD HBRS')
          WRITE          (1000,*) ('-------  ---------
     * --------- ---------------')

          DO 11161 OO=1,M
      IF( (FY(J,OO)-H_STAT(oo)).NE.0) THEN
      WRITE (*,*) FZO(J,OO), FY(J,OO),H_STAT(oo),DIO(OO),'   *'
      WRITE (1000,1001) FZO(J,OO), FY(J,OO),H_STAT(oo),DIO(OO),TIV
      ELSE
      WRITE (1000,1001) FZO(J,OO), FY(J,OO),H_STAT(oo),DIO(OO)
          ENDIF
11161 CONTINUE

      ELSE

          write (1000,*) '   '
      WRITE (*,*) ('DATA CORRECT AS SUBMITTED')
          WRITE (1000,*) ('DATA CORRECT AS SUBMITTED')
      WRITE (*,*) ('------------------------')
       WRITE (1000,*) ('WATERLINE  SUBM-HBRS
     * CORR-HBRS CORR+FAIRD HBRS')
          WRITE          (1000,*) ('-------  ---------
     * --------- ---------------')



          DO 11171 OO=1,M
          WRITE (1000,1001) FZO(J,OO), FY(J,OO),H_STAT(oo),DIO(OO)
11171 WRITE (*,*) FZO(J,OO), FY(J,OO),H_STAT(oo),DIO(OO)
```

```
c         pause
          ENDIF
C         END WRITE RESULTS OF DATA CORRECTION IN DETAILS.DAT

C         END PROCEED AND CHECK TWO DIMENSIONAL FAIRING


      do 3033 jio=1,m
          WRITE (*,*) 'J=', J, 'JIO=', JIO, 'DIO=', h(JIO)
c         pause 3033
      dddd1(j,jio)=yyf(jio)
 3033 continue
          go to 30
          endif


      J_NO=j


      close (708,status='DELETE')
          call
FFMATR1(NF,MF,XFb,YYF,INIF,XXCALF,dio,YYF_2DIF,FY_IN,J_NO_SAV
     *,i_plot,FXCAL,FZCAL,N1,M1,FX,FX_IN,TIF,ISENS,N_LE,IDEB,LIO)
      write (       *,*) i_plot, 'i_plot',j
c         pause 1000
      if(i_plot-1) 882,881,882
c     MODIFY 1d ARRAY (che) to 2d AEEAY (che1)
  881 open(701,file='derr_2.dat')
      rewind 701
          do 3333 k=1,5
          read(701,*) che(k)
          write(*,*)  che(k)

 3333 continue

      do 3334 k=1,5
      CHE1(K,J)=che(k)
 3334 write (*,*) k,J,CHE1(K,J)

c     end MODIFY 1d ARRAY (che) to 2d AEEAY (che1)

c     MODIFY 1d ARRAY (VAL) to 2d AEEAY (VAL1)
      open(702,file='VAL.dat')
      rewind 702
          do 33331 k=1,5
          read(702,*) VAL(k)
          write(*,*)  VAL(k)

33331 continue
```

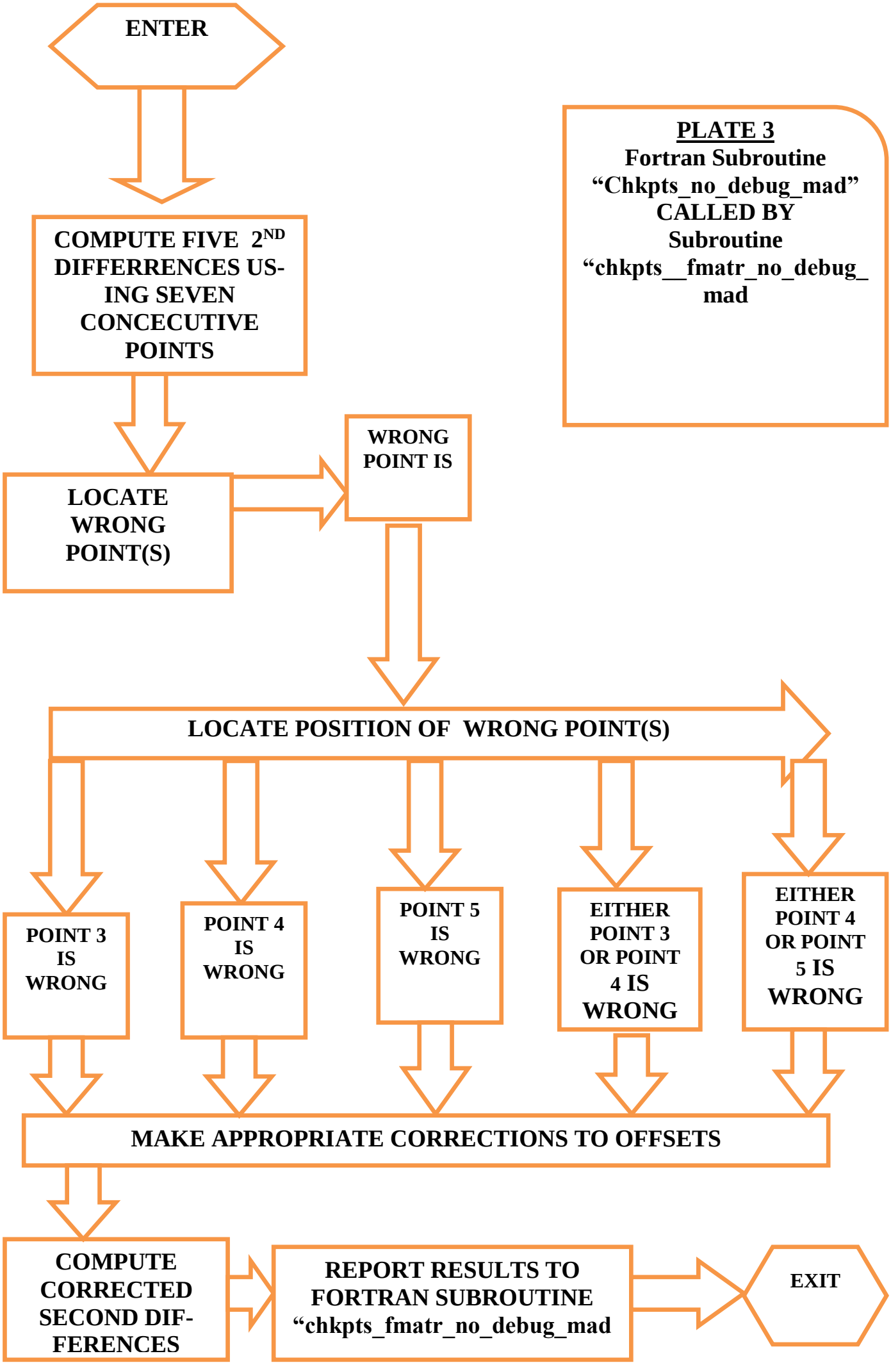
```
      do 33341 k=1,5
      VAL1(K,J)=VAL(k)
33341 write (*,*) k,J,VAL1(K,J)

c     end MODIFY 1d ARRAY (VAL) to 2d AEEAY (VAL1)

      go to 883
  882  CONTINUE

c      MODIFY 1d ARRAY (che_r) to 2d ARRAY (che_r1)
      open(708,file='derr_2r.dat')
      rewind 708
           do 33371 k=1,5
           read(708,*) che_r(k)
           WRITE(*,*) K
           write(*,*)  che_r(k)
c          pause 33371
33371 continue

      do 33381 k=1,5
      che_r1(K,J)=che_r(k)
33381 write (*,*) k,J,che_r1(K,J)
c      pause 33381
c     end MODIFY 1d ARRAY (che_r) to 2d AEEAY (che_r1)

  883 continue

      if(iTESs.gt.0) then
      write (1000,*) '   '
      write (1000,*) ('DATA CORRECTED-SEE BELOW')
      WRITE (1000,*) ('------------------------')
      WRITE (1000,*) ('WATERLINE  SUBM-HBRS
     * CORR-HBRS CORR+FAIRD HBRS')
           WRITE          (1000,*) ('-------  ---------
     * --------- ---------------')

c          DO 1116 OO=1,M
      DO 1116 OO=1,M1
      IF( (FY(J,OO)-H_STAT(oo)).NE.0) THEN
      WRITE (*,*) FZO(J,OO), FY(J,OO),H_STAT(oo),DIO(OO),'   *'
      WRITE (1000,1001) FZO(J,OO), FY(J,OO),H_STAT(oo),DIO(OO),TIV
      ELSE
      WRITE (1000,1001) FZO(J,OO), FY(J,OO),H_STAT(oo),DIO(OO)
           ENDIF
 1116 CONTINUE

      ELSE

           write (1000,*) '   '
      WRITE (*,*) ('DATA CORRECT AS SUBMITTED')
```

```
          WRITE (1000,*) ('DATA CORRECT AS SUBMITTED')
       WRITE (*,*) ('-------------------------')
        WRITE (1000,*) ('WATERLINE  SUBM-HBRS
      * CORR-HBRS CORR+FAIRD HBRS')
          WRITE          (1000,*) ('-------  ---------
      * ---------- ---------------')


c         DO 1117 OO=1,M
       DO 1117 OO=1,M1
          WRITE (1000,1001) FZO(J,OO), FY(J,OO),H_STAT(oo),DIO(OO)
 1117 WRITE (*,*) FZO(J,OO), FY(J,OO),H_STAT(oo),DIO(OO)
 1001 FORMAT (4(F10.3),A1    )

          ENDIF
       write (*,3031) j

       if (I_plot-1) 302,301,302

  301 do 3032 jio=1,M1
          WRITE (*,*) 'J=', J, 'JIO=', JIO, 'DIO=', DIO(JIO)
       dddd1(j,jio)=dio(jio)
 3032 WRITE (*,*) 'J=', J, 'JIO=', JIO, 'DDDD1=', DDDD1(J,JIO)

       go to 81819

  302 do 4031 jio=1,N1
          WRITE (*,*) 'J=', J, 'JIO=', JIO, 'DIO=', DIO(JIO)
       dddd1(j,jio)=dio(jio)
 4031 WRITE (*,*) 'J=', J, 'JIO=', JIO, 'DDDD1=', DDDD1(J,JIO)

81819 CONTINUE

c     Write 2d ARRAY che1 in file 322 (s_out_2d)
       rewind 322
 3221 format(80E14.3)
       do 31 i1=1,M-2
          write(*,*) i1, (che1(i1,j1),j1=1,N_SAV)
          write(322,3221) (che1(i1,j1),j1=1,N_SAV)
   31     continue
c     end Write 2d ARRAY che1 in file 322 (s_out_2d)

c     Write 2d ARRAY val1 in file 222 (s_in_2d)
       rewind 222
       do 311 i1=1,M-2
          write(*,*) i1, (val1(i1,j1),j1=1,N_SAV)

          write(222,3221) (val1(i1,j1),j1=1,N_SAV)
   311    continue
c     end Write 2d ARRAY val1 in file 222 (s_in_2d)
```

```
c     Write 2d ARRAY val1_r in file 221 (r_in_2d)
      rewind 221
      do 3111 i1=1,N_SAV-2
           write(*,*) i1, (val1_r(i1,j1),j1=1,11)
           write(221,3221) (val1_r(i1,j1),j1=1,11)
 3111   continue
c     end Write 2d ARRAY val1_r in file 221 (r_in_2d)

c     Write 2d ARRAY val1_r in file 221 (r_in_2d)
      rewind 321
      do 3112 i1=1,M-2
           write(*,*) i1, (che_r1(i1,j1),j1=1,11)
           write(321,3221) (che_r1(i1,j1),j1=1,11)
c          pause 3112
 3112   continue
c     end Write 2d ARRAY val1_r in file 221 (r_in_2d)
   30 continue
      pause 30

       if (nio.eq.2) then
      go to 81818
           else
           go to 81820
      endif

81818 continue

      write (*,*)
           write (*,*) 'exit CHKPTS_FMATR'
           write (*,*) 'BEFORE EXITING CHKPTS_FMATR valuew N1 M1',N1,M1
           GO TO 81820

81820 CONTINUE

      IF (I_PLOT-1) 411, 412, 411

  412 rewind 6001
           open (6001,file='DDDD1.dat')

           IF (LIO.ne.1) then
           N_SAV=n1
           else
           endif

           DO 41 J=1,N_SAV

 6002 format (100F15.3)

      WRITE (6001,6002) (DDDD1(J,JIO),JIO=1,M1)
```

```
   41 WRITE (*,*) 'DDDD1=',(DDDD1(J,JIO),JIO=1,M1) ,J,JIO
c      pause 411
  411 CONTINUE

      return
           end
```

ANNEX(3) + PLATE(3)
DOCUMENTATION OF FORTRAN
SUBROUTINE
“chkpts_no_debug_mad.for”


ENTER
PLATE 3
Fortran Subroutine
"Chkpts_no_debug_mad"
CALLED BY
Subroutine
"chkpts__fmatr_no_debug_
mad
COMPUTE FIVE 2ND DIFFERRENCES US-ING SEVEN CONCECUTIVE POINTS
LOCATE WRONG POINT(S)
WRONG POINT IS
LOCATE POSITION OF WRONG POINT(S)
POINT 3 IS WRONG
POINT 4 IS WRONG
POINT 5 IS WRONG
EITHER POINT 3 OR POINT 4 IS WRONG
EITHER POINT 4 OR POINT 5 IS WRONG
MAKE APPROPRIATE CORRECTIONS TO OFFSETS
COMPUTE CORRECTED SECOND DIF-FERENCES
REPORT RESULTS TO FORTRAN SUBROUTINE "chkpts_fmatr_no_debug_mad
EXIT

FORTRAN SUBROUTINE "chkpts_no_debug.for"

```
            Subroutine
CHKPTS(XF,F,M,H,iTEST,H_2DIF,finit,dd,no,JN,I_WRONG
    *,i_plot,N_LE,IDEB,LIO)

C     CHEKING OF POINTS COMPATIBILITY THRU SECOND DIFFERENCES
C     FOR THE HALF BREADTHS OF A SINGLE STATION
C     M= NUMBER OF WATERLINES
C     XF(I) I=1,7 HEIGHTS OF WATERLINES
C     F(I) I=1,7 GIVEN HALF BREADTHS
C     B(I) I=1,7 ARRAY OF CORRECTED HALF BREADTHS
C     THIS ROUTINE WORKS ONLY FOR A SET OF 7 POINTS
C     D1 D2 D3 D4 D5  ARE THE 5 CENTER SECOND DIFFERENCES OF THE  7
POINTS


      IMPLICIT REAL*8(A-H,O-Z)


c     DECLARATIONS
      DIMENSION XF(7),F(7),B(7),DD(500),H(7),DDC(500),H_2dif(500)
     *,finit(500),XX_LIM(3),DDIF_L(3),DDIF_R(3),R45_TRA(3),R12_TRA(3)
     *,R12_45(3)
      write (*,*) 'ENTERED CHECKPTS '
          If(IDEB-1) 7702,7701,7702
 7701   pause 7701
 7702 continue

      do 15 ii=1,7
      write (*,*) XF(ii),F(ii),i_plot
                  If(IDEB-1) 7704,7703,7704
 7703   pause 7703
 7704 continue
  15 continue

          ITEST=0
          DO 20 I=1,m
  20      H(I)=F(I)
      write (*,*) 'group no of  5 points cycle  in CHECKPTS ',no
      write (*,*) 'No of station (JN) in routine',JN




  10 icheck=0

C     CALCULATIONS OF SECOND DIFFERENCES

      DO 30 I=2,6
          A=(F(I+1)-F(I))/(XF(I+1)-XF(I))
```

```
          A1=(F(I)-F(I-1))/(XF(I)-XF(I-1))
          C=2./(XF(I+1)-XF(I-1))

      DD(I)=C*(A-A1)
   30 finit(i)=DD(I)

      D1=DD(2)
      D2=DD(3)
      D3=DD(4)
          D4=DD(5)
      D5=DD(6)
      write (*,*) D1,D2,D3,D4,D5
      If(IDEB-1) 7706,7705,7706
 7705   pause 7705
 7706 continue

      IF (((D1.Lt.0).AND.(D2.Gt.0).AND.(D3.Lt.0).AND.(D4.Gt.0
     *).AND.(D5.Lt.0)).OR.
     *((D1.Gt.0).AND.(D2.Lt.0).AND.(D3.Gt.0).AND.(D4.Lt.0
     *).AND.(D5.Gt.0)))

     *THEN
 C     VERY UNLIKELY CASE
       If(IDEB-1) 7708,7707,7708
 7707   pause 7707
 7708 continue

      I_WRONG=1
      write (*,*) 'very unlikely case-GO TO 1000 I_WRONG=',I_WRONG

      GO TO 5

      ELSE
      continue
      ENDIF




          IF (((D1.Lt.0).AND.(D2.Gt.0).AND.(D3.Lt.0).AND.(D4.Lt.0
     *).AND.(D5.LE.0)).OR.
     *((D1.Lt.0).AND.(D2.Gt.0).AND.(D3.Lt.0).AND.(D4.Lt.0
     *).AND.(D5.GE.0)).OR.
     *((D1.Gt.0).AND.(D2.Lt.0).AND.(D3.Gt.0).AND.(D4.Gt.0
     *).AND.(D5.Gt.0)).OR.
     *((D1.Gt.0).AND.(D2.Lt.0).AND.(D3.Gt.0).AND.(D4.Gt.0
     *).AND.(D5.Lt.0)).or.
     *((D1.lt.0).AND.(D2.eq.0).AND.(D3.Gt.0).AND.(D4.lt.0
     *).AND.(D5.Lt.0)).or.
     *((D1.eq.0).AND.(D2.lt.0).AND.(D3.Gt.0).AND.(D4.lt.0
```

```
     *).AND.(D5.Lt.0)).or.
     *((D1.gt.0).AND.(D2.lt.0).AND.(D3.Gt.0).AND.(D4.eq.0
     *).AND.(D5.gt.0)).or.
     *((D1.lt.0).AND.(D2.gt.0).AND.(D3.eq.0).AND.(D4.lt.0
     *).AND.(D5.lt.0)).or.
     *((D1.lt.0).AND.(D2.gt.0).AND.(D3.lt.0).AND.(D4.lt.0
     *).AND.(D5.eq.0)).or.
     *((D1.gt.0).AND.(D2.lt.0).AND.(D3.gt.0).AND.(D4.gt.0
     *).AND.(D5.eq.0)).or.
     *((D1.lt.0).AND.(D2.gt.0).AND.(D3.lt.0).AND.(D4.eq.0
     *).AND.(D5.lt.0)).or.
     *((D1.lt.0).AND.(D2.gt.0).AND.(D3.lt.0).AND.(D4.eq.0
     *).AND.(D5.lt.0)).or.
     *((D1.gt.0).AND.(D2.lt.0).AND.(D3.gt.0).AND.(D4.eq.0
     *).AND.(D5.eq.0)).or.
     *((D1.lt.0).AND.(D2.gt.0).AND.(D3.lt.0).AND.(D4.eq.0
     *).AND.(D5.eq.0)))
     *THEN
C    CASE I

      icheck=1
      ITEST=1
      write (*,*) 'no 3/7(point)',(no-1)+3, ' OF STATION',JN,
     *    '   must be corrected-GO TO 5'

      GO TO 5


      ELSE

      ENDIF

                  IF (((D1.Lt.0).AND.(D2.Lt.0).AND.(D3.Gt.0).AND.(D4.Lt.0
     *).AND.(D5.LE.0)).OR.

     *((D1.Gt.0).AND.(D2.Gt.0).AND.(D3.Lt.0).AND.(D4.Gt.0
     *).AND.(D5.GE.0)).or.
     *((D1.eq.0).AND.(D2.Gt.0).AND.(D3.Lt.0).AND.(D4.Gt.0
     *).AND.(D5.Gt.0)).or.
     *((D1.eq.0).AND.(D2.Gt.0).AND.(D3.Lt.0).AND.(D4.Gt.0
     *).AND.(D5.eq.0)).or.
     *((D1.eq.0).AND.(D2.lt.0).AND.(D3.gt.0).AND.(D4.lt.0
     *).AND.(D5.eq.0)))

     *THEN


      write (*,*) 'no 4/7(point)',(no-1)+4, ' OF STATION',JN,
     *    '   must be corrected-GO TO 6'
      ITEST=1
```

```
      GO TO 6

          else


      ENDIF

                          IF (((D1.Lt.0).AND.(D2.Lt.0).AND.(D3.Lt.0).AND.(D4.Gt.0
     *).AND.(D5.LE.0)).OR.
     *((D1.Lt.0).AND.(D2.Gt.0).AND.(D3.Gt.0).AND.(D4.Lt.0
     *).AND.(D5.Gt.0)).OR.
     *((D1.Gt.0).AND.(D2.Gt.0).AND.(D3.Gt.0).AND.(D4.Lt.0
     *).AND.(D5.Gt.0)).OR.
     *((D1.Gt.0).AND.(D2.Lt.0).AND.(D3.Lt.0).AND.(D4.Gt.0
     *).AND.(D5.Lt.0)).or.
     *((D1.lt.0).AND.(D2.eq.0).AND.(D3.Lt.0).AND.(D4.Gt.0
     *).AND.(D5.Lt.0)).or.
     *((D1.eq.0).AND.(D2.lt.0).AND.(D3.Lt.0).AND.(D4.Gt.0
     *).AND.(D5.Lt.0)).or.
     *((D1.gt.0).AND.(D2.eq.0).AND.(D3.gt.0).AND.(D4.lt.0
     *).AND.(D5.gt.0)).or.
     *((D1.gt.0).AND.(D2.eq.0).AND.(D3.Lt.0).AND.(D4.Gt.0
     *).AND.(D5.Lt.0)).or.
     *((D1.eq.0).AND.(D2.gt.0).AND.(D3.gt.0).AND.(D4.lt.0
     *).AND.(D5.gt.0)).or.
     *((D1.eq.0).AND.(D2.eq.0).AND.(D3.gt.0).AND.(D4.lt.0
     *).AND.(D5.gt.0)).or.
     *((D1.eq.0).AND.(D2.eq.0).AND.(D3.lt.0).AND.(D4.gt.0
     *).AND.(D5.lt.0)))


     *THEN
C    CASE III

      icheck=1
      ITEST=1
          write (*,*)'no 5/7(point)',(no-1)+5, ' OF STATION',JN,
     *    '   must be corrected-GO TO 7'

      GO TO 7

      else

          ENDIF
                          IF (((D1.Lt.0).AND.(D2.Gt.0).AND.(D3.Lt.0).AND.(D4.Gt.0
     *).AND.(D5.Gt.0)).OR.
     *((D1.Gt.0).AND.(D2.Lt.0).AND.(D3.Gt.0).AND.(D4.Lt.0
     *).AND.(D5.Lt.0)).or.
     *((D1.Gt.0).AND.(D2.Lt.0).AND.(D3.Gt.0).AND.(D4.eq.0
     *).AND.(D5.Lt.0)).or.
```

```
     *((D1.lt.0).AND.(D2.gt.0).AND.(D3.lt.0).AND.(D4.gt.0
     *).AND.(D5.eq.0)).or.
     *((D1.Gt.0).AND.(D2.Lt.0).AND.(D3.Gt.0).AND.(D4.lt.0
     *).AND.(D5.eq.0)))

     *THEN
C     CASE IV

      ITEST=1
           write (*,*)'no 3 or 4 /7(pt)',(no-1)+3,(no-1)+4, ' OF STATION',JN,
     *     '   must be corrected-GO TO 8'

      GO TO 8

      else

           ENDIF
                                      IF
(((D1.Lt.0).AND.(D2.Lt.0).AND.(D3.Gt.0).AND.(D4.Lt.0
     *).AND.(D5.Gt.0)).OR.
     *((D1.Gt.0).AND.(D2.Gt.0).AND.(D3.Lt.0).AND.(D4.Gt.0
     *).AND.(D5.Lt.0)).or.
     *((D1.lt.0).AND.(D2.eq.0).AND.(D3.gt.0).AND.(D4.lt.0
     *).AND.(D5.gt.0)).or.
     *((D1.eq.0).AND.(D2.lt.0).AND.(D3.gt.0).AND.(D4.lt.0
     *).AND.(D5.gt.0)) .or.
     *((D1.eq.0).AND.(D2.gt.0).AND.(D3.lt.0).AND.(D4.gt.0
     *).AND.(D5.lt.0)) )

     *THEN
C     CASE V

      ITEST=1
      write (*,*)'no 4 or 5 /7(pt)',(no-1)+4,(no-1)+5, ' OF STATION',JN,
     *     '   must be corrected-GO TO 9'

           GO TO 9

      else

      go to 300
           write (*,*) 'CORRECTIONS FINISHED-GO TO 300'

           ENDIF


    5 CONTINUE
C     CASE I
      A1=F(2)*(XF(3)-XF(4))
           A2=F(4)*(XF(3)-XF(2))
```

```
      A3=XF(2)-XF(4)
      YC=(A1-A2)/A3
      B1=F(4)-F(1)
   B2=XF(1)*(F(2)-F(1))/(XF(2)-XF(1))
   B3=XF(4)*(F(5)-F(4))/(XF(5)-XF(4))
      C1=(F(2)-F(1))/(XF(2)-XF(1))
      C2=(F(5)-F(4))/(XF(5)-XF(4))
   XLIM=(B1+B2-B3)/(C1-C2);
   D1=F(2)*(XF(3)-XF(1))
      D2=F(1)*(XF(3)-XF(2))
      D3=XF(2)-XF(1)
      Y12=(D1-D2)/D3
      D1=F(5)*(XF(3)-XF(4))
      D2=F(4)*(XF(3)-XF(5))
      D3=XF(5)-XF(4)
      Y45=(D1-D2)/D3
   IF  ((XLIM.LE.XF(4)).AND.(XLIM.GE.XF(2)))          THEN
      CONTINUE
      write (*,*) 'CORRECT AS TRIANGLE-GO TO 51'
   write (*,*) 'XLIM=       ',XLIM

      GO TO 51

      ELSE
   R1=Y45
      R2=Y12
      H(3)=(R1+R2)/2.

   write (*,*) 'CORRECTION AS TRAPEZIUM FINISHED-GO TO 300'
      write (*,*) 'XLIM=       ',XLIM,R1,R2
      WRITE(*,*) (H(NIO),NIO=1,7)

   DO 301 I=2,6
      A=(H(I+1)-H(I))/(XF(I+1)-XF(I))
      A1=(H(I)-H(I-1))/(XF(I)-XF(I-1))
      C=2./(XF(I+1)-XF(I-1))

301 DD(I)=C*(A-A1)
   WRITE(*,*) (DD(NIO),NIO=2,6)
   GO TO 300

   ENDIF
 51 continue
   IF (XF(3).GE.XLIM)      THEN
      write (*,*) 'GO TO 52'
      GO TO 52

      ELSE
      R1=YC
      R2=Y12
```

```
        H(3)=(R1+R2)/2.
        If(IDEB-1) 7710,7709,7710
7709    pause 7709
7710 continue


        ENDIF
        write (*,*) 'CORRECTION FINISHED- GO TO 300'
        GO TO 300

  52    R1=YC
        R2=Y45
        H(3)=(R1+R2)/2.
        If(IDEB-1) 7712,7711,7712
7711    pause 7711
7712 continue


        write (*,*) 'CORRECTION FINISHED GO TO 300'
        GO TO 300

   6 CONTINUE

C     CASE II
        A1=F(3)*(XF(4)-XF(5))
        A2=F(5)*(XF(4)-XF(3))
        A3=XF(3)-XF(5)
        YC=(A1-A2)/A3
        B1=F(5)-F(2)
     B2=XF(2)*(F(3)-F(2))/(XF(3)-XF(2))
     B3=XF(5)*(F(6)-F(5))/(XF(6)-XF(5))
        C1=(F(3)-F(2))/(XF(3)-XF(2))
        C2=(F(6)-F(5))/(XF(6)-XF(5))
        XLIM=(B1+B2-B3)/(C1-C2)
        D1=F(3)*(XF(4)-XF(2))
        D2=F(2)*(XF(4)-XF(3))
        D3=XF(3)-XF(2)
        Y12=(D1-D2)/D3
        D1=F(6)*(XF(4)-XF(5))
        D2=F(5)*(XF(4)-XF(6))
        D3=XF(6)-XF(5)
        Y45=(D1-D2)/D3
        IF  ((XLIM.LE.XF(5)).AND.(XLIM.GE.XF(3)))     THEN
        CONTINUE
        write (*,*) 'CORRECT AS TRIANGLE- GO TO 61'
        GO TO 61

        ELSE
        R1=Y45
        R2=Y12
```

```
        H(4)=(R1+R2)/2.
        write (*,*) 'CORRECTION AS TRAPEZIUM FINISHED-GO TO 300'
    GO TO 300

        ENDIF
  61 CONTINUE
        IF (XF(4).GE.XLIM) THEN
        write (*,*) 'GO TO 62'
        GO TO 62

        ELSE
        R1=YC
        R2=Y12
        H(4)=(R1+R2)/2.
        write (*,*) 'CORRECTION FINISHED- GO TO 300'
    go to 100

        ENDIF
  62    R1=YC
        R2=Y45
        H(4)=(R1+R2)/2.
        If(IDEB-1) 7714,7713,7714
 7713   pause 7713
 7714 continue


        write (*,*) 'CORRECTION FINISHED-GO TO 100'
        GO TO 300

    7 CONTINUE
C     CASE III
        A1=F(4)*(XF(5)-XF(6))
        A2=F(6)*(XF(5)-XF(4))
        A3=XF(4)-XF(6)
        YC=(A1-A2)/A3

        B1=F(6)-F(3)
    B2=XF(3)*(F(4)-F(3))/(XF(4)-XF(3))
    B3=XF(6)*(F(7)-F(6))/(XF(7)-XF(6))
        C1=(F(4)-F(3))/(XF(4)-XF(3))
        C2=(F(7)-F(6))/(XF(7)-XF(6))
    XLIM=(B1+B2-B3)/(C1-C2)

    D1=F(4)*(XF(5)-XF(3))
        D2=F(3)*(XF(5)-XF(4))
        D3=XF(4)-XF(3)
        Y12=(D1-D2)/D3

    D1=F(7)*(XF(5)-XF(6))
        D2=F(6)*(XF(5)-XF(7))
```

```
        D3=XF(7)-XF(6)
        Y45=(D1-D2)/D3
        IF  ((XLIM.LE.XF(6)).AND.(XLIM.GE.XF(4)))      THEN
        CONTINUE
        write (*,*) 'CORRECT AS TRIANGLE -GO TO 71'
        GO TO 71

        ELSE
        R1=Y45
        R2=Y12
        H(5)=(R1+R2)/2.
   write (*,*) 'CORRECTION AS TRAPEZIUM FINISHED-GO TO 300'
        write (*,*) 'XLIM=        ',XLIM,R1,R2
        WRITE(*,*) (H(NIO),NIO=1,7)


        DO 701 I=2,6
        A=(H(I+1)-H(I))/(XF(I+1)-XF(I))
        A1=(H(I)-H(I-1))/(XF(I)-XF(I-1))
        C=2./(XF(I+1)-XF(I-1))

 701 DD(I)=C*(A-A1)
   WRITE(*,*) (DD(NIO),NIO=2,6)
        GO TO 300



        ENDIF
  71 CONTINUE
        IF (XF(5).GE.XLIM)  THEN
        write (*,*) 'GO TO 72'
        GO TO 72

        ELSE
        R1=YC
        R2=Y12
        H(5)=(R1+R2)/2.
        If(IDEB-1) 7716,7715,7716
7715   pause 7715
7716 continue

        write (*,*) 'CORRECTION FINISHED-GO TO 300'
        GO TO 300

    endif
  72    R1=YC
        R2=Y45
        H(5)=(R1+R2)/2.
                If(IDEB-1) 7718,7717,7718
7717   pause 7717
```

```
 7718 continue

          write (*,*) 'CORRCTIONS FINISHED -GO TO 300'
          GO TO 300


    8 CONTINUE
C     CASE IV
          in1=0
          in2=0
      H1=(2./(XF(4)-XF(1)))*
     *(((F(4)-F(3))/(XF(4)-XF(3)))-
     *((F(3)-F(2))/(XF(3)-XF(2))))

      H3=(2./(XF(5)-XF(3)))*
     *(((F(5)-F(4))/(XF(5)-XF(4)))-
     *((F(4)-F(3))/(XF(4)-XF(3))))


          H4= (2./(XF(6)-XF(4)))*
     *(((F(6)-F(5))/(XF(6)-XF(5)))-
     *((F(5)-F(4))/(XF(5)-XF(4))))

      H5=(2./(XF(7)-XF(5)))*
     *(((F(7)-F(6))/(XF(7)-XF(6)))-
     *((F(6)-F(5))/(XF(6)-XF(5))))
C         DIFF D2 REMOVED CHECK NOW
                          IF (((H1.Lt.0).AND.(H3.Lt.0).AND.(H4.Gt.0).AND.(H5.Lt.0
     *)).OR.
     *((H1.Lt.0).AND.(H3.Gt.0).AND.(H4.Lt.0).AND.(H5.Lt.0
     *)).OR.
     *((H1.Lt.0).AND.(H3.Gt.0).AND.(H4.Lt.0).AND.(H5.Gt.0
     *)).OR.
     *((H1.Gt.0).AND.(H3.Gt.0).AND.(H4.Lt.0).AND.(H5.Gt.0
     *)).OR.
     *((H1.Gt.0).AND.(H3.Lt.0).AND.(H4.Gt.0).AND.(H5.Gt.0
     *)).OR.
     *((H1.Gt.0).AND.(H3.Lt.0).AND.(H4.Gt.0).AND.(H5.Lt.0
     *)))
     *THEN
      in1=30
      H1=(2./(XF(3)-XF(1)))*
     *(((F(3)-F(2))/(XF(3)-XF(2)))-
     *((F(2)-F(1))/(XF(2)-XF(1))))

          H4= (2./(XF(6)-XF(3)))*
     *(((F(6)-F(5))/(XF(6)-XF(5)))-
     *((F(5)-F(3))/(XF(5)-XF(3))))
c         SIMILAR TO CASE I (D2 PROBLEMATIC)
c      PAUSE 41
```

```
      '
       Write(*,*) in1


            ELSE
            in1=31
            H1=(2./(XF(3)-XF(1)))*
     *(((F(3)-F(2))/(XF(3)-XF(2)))-
     *((F(2)-F(1))/(XF(2)-XF(1))))

            H4= (2./(XF(6)-XF(3)))*
     *(((F(6)-F(5))/(XF(6)-XF(5)))-
     *((F(5)-F(3))/(XF(5)-XF(3))))
            in2=31
C           SIMILAR TO CASE II(D3 no PROBLEMATIC)
            If(IDEB-1) 7720,7719,7720
 7719   pause 7719
 7720 continue

c           GO TO          6
            ENDIF
            IF (((H1.Lt.0).AND.(H3.Lt.0).AND.(H4.Gt.0).AND.(H5.Lt.0
     *)).OR.
     *((H1.Lt.0).AND.(H3.Gt.0).AND.(H4.Lt.0).AND.(H5.Lt.0
     *)).OR.
     *((H1.Lt.0).AND.(H3.Gt.0).AND.(H4.Lt.0).AND.(H5.Gt.0
     *)).OR.
     *((H1.Gt.0).AND.(H3.Gt.0).AND.(H4.Lt.0).AND.(H5.Gt.0
     *)).OR.
     *((H1.Gt.0).AND.(H3.Lt.0).AND.(H4.Gt.0).AND.(H5.Gt.0
     *)).OR.
     *((H1.Gt.0).AND.(H3.Lt.0).AND.(H4.Gt.0).AND.(H5.Lt.0
     *)))
     *THEN
            in2=40
             else
            in2=41
            endif
       Write(*,*) in1,in2




            IF ((in1.eq.30).AND.(in2.eq.40)) THEN
            write (*,*) 'GO TO 1001'
            go to 1001

c           'Neither point #3 or  #4 corrction solves problem
            else
            continue
```

```
      endif

      IF ((in1.eq.31).AND.(in2.eq.41)) THEN
c     Both  points #3 AND #4 corrction solves problem
           write (*,*) 'Both  points #3 AND #4 if corrected solve
     * the problem INDETERMINATE- RECHECK DATA ON STATION NO.JN'
           write (*,*) 'GO TO 1002'
           go to 1002

           else
           continue
      endif

      IF ((in1.eq.30).AND.(in2.eq.41)) THEN
           write (*,*) 'Point #3 (DIFFERENCE #2)
     * of station NO.JN WILL BE CORRECTED...'
           write (*,*) 'GO TO 6'
           go to 6

           else
           continue
      endif

      IF ((in1.eq.31).AND.(in2.eq.40)) THEN
           write (*,*) 'Point #4 (DIFFERENCE #3)
     * of station NO JN.WILL BE CORRECTED...'
           write (*,*) 'GO TO 7'
           go to 7

           else
           continue
      endif


C     CASE V

    9  in1=0
            in2=0
           H1=(2./(XF(3)-XF(1)))*
     *(((F(3)-F(2))/(XF(3)-XF(2)))-
     *((F(2)-F(1))/(XF(2)-XF(1))))

      H2=(2./(XF(2)-XF(5)))*
     *(((F(5)-F(3))/(XF(5)-XF(3)))-
     *((F(3)-F(2))/(XF(3)-XF(2))))


           H4= (2./(XF(6)-XF(3)))*
     *(((F(6)-F(5))/(XF(6)-XF(5)))-
     *((F(5)-F(3))/(XF(5)-XF(3))))
```

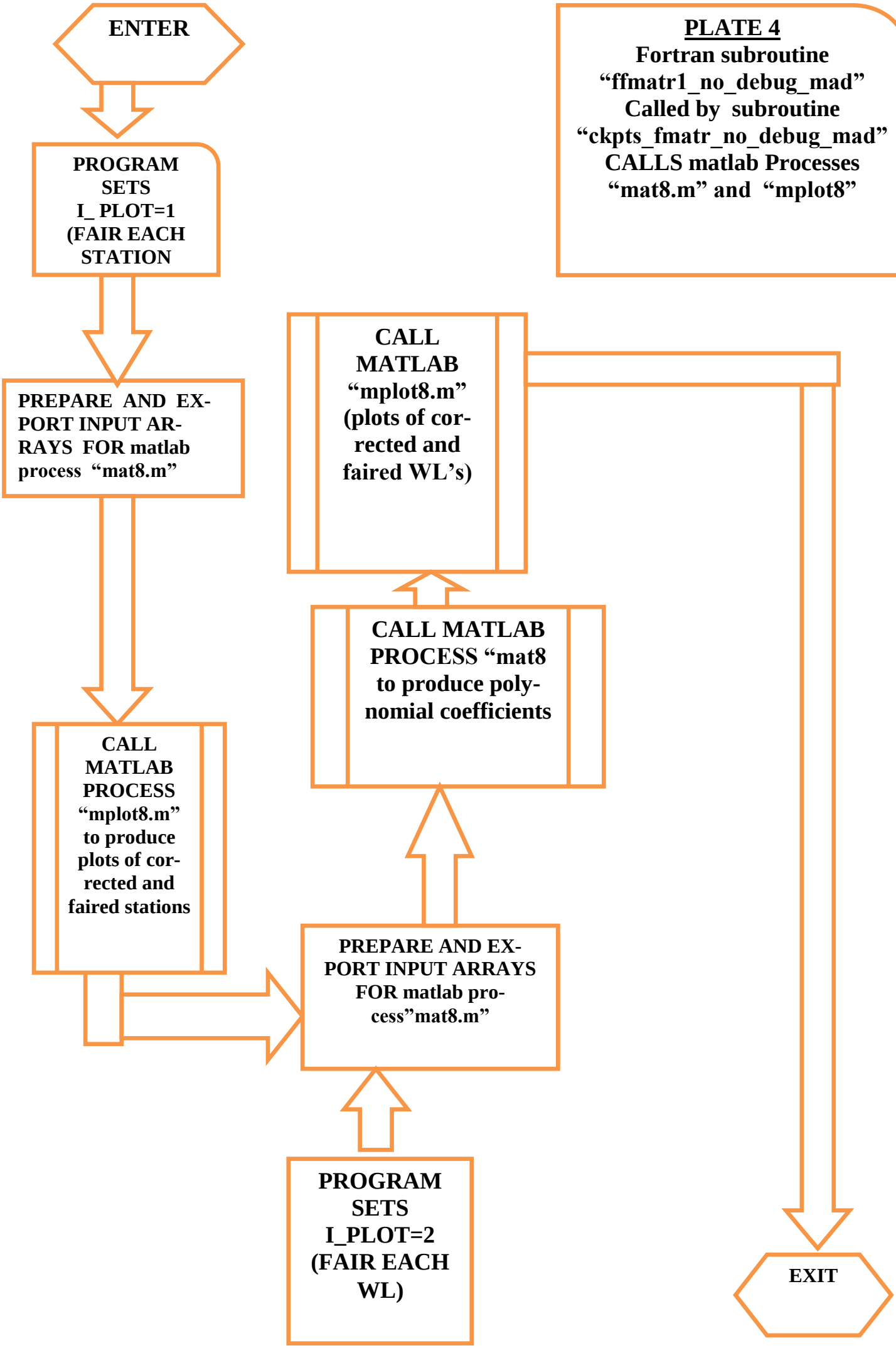
```
     H5=(2./(XF(7)-XF(5)))*
    *(((F(7)-F(6))/(XF(7)-XF(6)))-
    *((F(6)-F(5))/(XF(6)-XF(5))))
C        DIFF D3 REMOVED CHECK NOW
                                  IF
(((H1.Lt.0).AND.(H2.Lt.0).AND.(H4.Gt.0).AND.(H5.Lt.0
    *)).OR.
    *((H1.Lt.0).AND.(H2.Gt.0).AND.(H4.Lt.0).AND.(H5.Lt.0
    *)).OR.
    *((H1.Lt.0).AND.(H2.Gt.0).AND.(H4.Lt.0).AND.(H5.Gt.0
    *)).OR.
    *((H1.Gt.0).AND.(H2.Gt.0).AND.(H4.Lt.0).AND.(H5.Gt.0
    *)).OR.
    *((H1.Gt.0).AND.(H2.Lt.0).AND.(H4.Gt.0).AND.(H5.Gt.0
    *)).OR.
    *((H1.Gt.0).AND.(H2.Lt.0).AND.(H4.Gt.0).AND.(H5.Lt.0
    *))  )
    *THEN

     in1=30
      H2=(2./(XF(4)-XF(2)))*
    *(((F(4)-F(3))/(XF(4)-XF(3)))-
    *((F(3)-F(2))/(XF(3)-XF(2))))

          H5=(2./(XF(7)-XF(4)))*
    *(((F(7)-F(6))/(XF(7)-XF(6)))-
    *((F(6)-F(4))/(XF(6)-XF(4))))

c        SIMILAR TO CASE I (D2 PROBLEMATIC)
     If(IDEB-1) 7722,7721,7722
 7721   pause 7721
 7722 continue

c        GO TO          5
     Write(*,*) in1


          ELSE
          in1=31




     H2=(2./(XF(4)-XF(2)))*
    *(((F(4)-F(3))/(XF(4)-XF(3)))-
    *((F(3)-F(2))/(XF(3)-XF(2))))

          H5=(2./(XF(7)-XF(4)))*
    *(((F(7)-F(6))/(XF(7)-XF(6)))-
```

```
     *((F(6)-F(4))/(XF(6)-XF(4))))
C         SIMILAR TO CASE II(D3 no PROBLEMATIC)

          If(IDEB-1) 7724,7723,7724
 7723   pause 7723
 7724 continue

c         GO TO          6
          ENDIF
          IF (((H1.Lt.0).AND.(H3.Lt.0).AND.(H4.Gt.0).AND.(H5.Lt.0
     *)).OR.
     *((H1.Lt.0).AND.(H3.Gt.0).AND.(H4.Lt.0).AND.(H5.Lt.0
     *)).OR.
     *((H1.Lt.0).AND.(H3.Gt.0).AND.(H4.Lt.0).AND.(H5.Gt.0
     *)).OR.
     *((H1.Gt.0).AND.(H3.Gt.0).AND.(H4.Lt.0).AND.(H5.Gt.0
     *)).OR.
     *((H1.Gt.0).AND.(H3.Lt.0).AND.(H4.Gt.0).AND.(H5.Gt.0
     *)).OR.
     *((H1.Gt.0).AND.(H3.Lt.0).AND.(H4.Gt.0).AND.(H5.Lt.0
     *)))
     *THEN
          in2=40
           else
          in2=41
          endif

          IF ((in1.eq.30).AND.(in2.eq.40)) THEN
          write (*,*) 'GO TO 10011'
          go to 10011

c         Neither point #4 or  #5 corrction solves problem
          else
          continue
      endif




      IF ((in1.eq.31).AND.(in2.eq.41)) THEN
c     Both  points #4 AND #5 corrction solves problem
      write (*,*) 'GO TO 10021'
      go to 10021

      else
      continue
          endif

      IF ((in1.eq.30).AND.(in2.eq.41)) THEN
```

```
          write (*,*) 'Point #4 (DIFFERENCE #3)
     * of station NO JN.WILL BE CORRECTED...'
          write (*,*) 'GO TO 6'
          go to 6

          else
          continue
       endif

      IF ((in1.eq.31).AND.(in2.eq.40)) THEN
          write (*,*) 'Point #5 (DIFFERENCE #4)
     * of station NO.JN.WILL BE CORRECTED...'
          write (*,*) 'GO TO 7'
          go to 7

          else
          continue
       endif

  100    CONTINUE

C     CALCULATE NEW SECOND DIFFERENCES
          DO 101 I=2,6
          A=(H(I+1)-H(I))/(XF(I+1)-XF(I))
          A1=(H(I)-H(I-1))/(XF(I)-XF(I-1))
          C=2./(XF(I+1)-XF(I-1))
       DDC(I)=C*(A-A1)

  101 continue

          if (icheck.eq.0) then
          write (*,*) 'NO NECESSITY FOR CORRECTIONS -GO TO 300'
       go to 300

          else
       do 250 i=1,7
  250 f(i)=h(i)
          go to 10
          write (*,*) 'GO TO 10'

          endif
 1000 WRITE (*,*) 'VERY UNLIKELY CASE'
       write (*,*) 'GO TO 300'
       go to 300

  1001 continue

       write (*,*) 'Neither point #3 or  #4 if corrected solves
      * the problem RECHECK DATA ON STATION NO.JN.'
       I_WRONG=1
```

```
      write (*,*) 'Neither point #3 or  #4 if corrected solves
     *the problem I_WRONG=',I_WRONG
           write (*,*) 'GO TO 300'
           go to 300

 1002 continue
           write (*,*) 'Both  points #3 AND #4 if corrected solve
     *the problem INDETERMINATE- RECHECK DATA ON STATION NO.JN'
      I_WRONG=1
      write (*,*) 'Both  points #3 AND #4 if corrected solve
     *the problem I_WRONG=',I_WRONG
      write (*,*) 'GO TO 300'
      go to 300

10011 continue
      write (*,*) 'Neither point #4 or  #5 if corrected solves
     *the problem RECHECK DATA ON STATION NO.JN.'
      write (*,*) 'Neither point #4 or  #5 if corrected solves
     *the problem I_WRONG=',I_WRONG
           write (*,*) 'GO TO 300'
           go to 300

10021 continue
           write (*,*) 'Both  points #4 AND #5 if corrected solve
     *the problem INDETERMINATE- RECHECK DATA ON STATION NO.JN.'
           I_WRONG=1
      write (*,*) 'very unlikely case-Both pts #4 AND #5 solve the problem
     * I_WRONG=',I_WRONG
  300 continue
c     recalculate FINAL second differences
      DO 3000 I=2,6
           A=(H(I+1)-H(I))/(XF(I+1)-XF(I))
           A1=(H(I)-H(I-1))/(XF(I)-XF(I-1))
           C=2./(XF(I+1)-XF(I-1))

 3000 H_2DIF(I-1)=C*(A-A1)

      DD1=H_2DIF(1)
      DD2=H_2DIF(2)
      DD3=H_2DIF(3)
           DD4=H_2DIF(4)
      DD5=H_2DIF(5)




      RETURN
      END
```

ANNEX(4) + PLATE(4)
DOCUMENTATION OF FORTRAN SUBROUTINE
“ffmatr1_no_debug_mad.f”


ENTER
PLATE 4
Fortran subroutine
"ffmatr1_no_debug_mad"
Called by subroutine
"ckpts_fmatr_no_debug_mad"
CALLS matlab Processes
"mat8.m" and "mplot8"
PROGRAM SETS I_ PLOT=1 (FAIR EACH STATION
PREPARE AND EX-PORT INPUT AR-RAYS FOR matlab process "mat8.m"
CALL MATLAB PROCESS "mplot8.m" to produce plots of cor-rected and faired stations
PREPARE AND EX-PORT INPUT ARRAYS FOR matlab pro-cess"mat8.m"
CALL MATLAB PROCESS "mat8 to produce poly-nomial coefficients
CALL MATLAB "mplot8.m" (plots of cor-rected and faired WL's)
PROGRAM SETS I_PLOT=2 (FAIR EACH WL)
EXIT

SUBROUTINE "ffmatr1_no_debug_mad.f"

```
      SUBROUTINE FFMATR1(N,M,X,YY,INI,XXCAL,dddd,yyf_2dif,FY_IN,j_no
     *,i_plot,FXCAL,FZCAL,N11,M11,FX,FX_IN,TIF,ISENS,N_LE,IDEB,LIO)

C     TWO DIMENSIONAL FAIRING BY LINEAR PROGRAMMING

C     N= NUMBER OF GIVEN POINTS
C     X(I) I=1,N HEIGHTS OF COORDINATES
C     Y(I) I=1,N GIVEN HALF BREADTHS
C     INI(I) INDEX 0=NO FIXED POINT,1=FIXED POINT
C     YCAL(J) J=1,M CALCULATED HALF BREADTHS
C      XCAL(J) J=1,M HEIGHTS WHERE FAIRED HALF BREADTHS ARE RE-
QUIRED
C     AA(N,N) COEFFICIENS FOR + EQUATIONS OF LINEAR PROGAMMING
PROBLEM
C     CC(N,N) AS ABOVE FOR - EQUATIONS=-AA(N,N)

      IMPLICIT REAL*8(A-H,O-Z)

c     DECLARATIONS FIRST PART
      DIMENSION X(80),Y(80),XXCAL(80),YCAL(80),AA(80,80),CC(80,80)
     *,AAA(80),CCC(80),R(80),RR(80,80),RRC(80,80),yy(800) ,d(80)
     *,ini(80),yeq(80),yineq(80),iq(80),inq(80)

C     DECLARATIONS SECOND PART
          DIMENSION A(80,80),B(80,80),BMIN(80,80),CJ(80),XB(80,1),pppp(80)
      DIMENSION BB(80),B1(80,1),T(80),C(80),XB1(80),XCAL(80),DDDD(80)
      dimension ssss(80),yyf_2dif(80),FY_IN(500),FX_IN(500)
      dimension FXCAL(80),FZCAL(80),FX(500)
      DIMENSION FLE_CH(1500),FHE_CH (1500),VAL(80)
C     DECLARATIONS FOR CALLING MATLAB
          integer engopen,enggetmatrix,mxcreatefull,mxgetpr

          integer ep,T,T1,T2      ,T3      ,T22     ,T11     ,t21,ti1400,ti1401,t20
      CHARACTER*1 TIV,TIVAL,TIF(50,50)
          integer engputmatrix,engevalstring,engclose
          integer status
      OPEN(1011,FILE='i_plot.DAT')

      OPEN(1089,FILE='circ_s.DAT')
      OPEN(1090,FILE='circ_r.DAT')
          rewind 1011
          write (1011,*) i_plot
      if (ISENS-1) 12 ,12, 11
   11 continue
      open (8023, file='J_NO.dat')
          rewind 8023

      write (8023,*) J_NO
```

```
          go to 13
   12 open (8011, file='LLENGTHS.dat')
          open (8021, file='HHEIGHTS.dat')
      open (8023, file='J_NO.dat')
          rewind 8023

      write (8023,*) J_NO
      rewind 8011
          rewind 8023
          rewind  8021


          do 203 i=1,N11
      READ(8011,*) FLE_CH(i)
   203 write(*,*) FlE_CH(i)
          do 204 i=1,M11
      READ(8021,*) FHE_CH(i)
   204 write(*,*) FHE_CH(i)
    13 continue
      pause 13

          If(IDEB-1) 6602,6601,6602
  6601 pause 6601
  6602 continue

C     PLACE ZEROS TO VARIABLE aa (according to each dimensin)
      do 2000 I=1,80
      do 2000 J=1,80
  2000 aa(I,J)=0.
C     end PLACE ZEROS TO VARIABLE aa (according to each dimension)

C     USEFUL FOR TESTING FOR UP TO 30 WATERLINES
      open (8025, file='N1_M1.dat')
          CLOSE (8025, STATUS = 'DELETE')
          open (8025, file='N1_M1.dat')
   8025 FORMAT (4I3)
      WRITE (8025,8025) n,m,n11,m11
      WRITE (*,*) n,m,n11,m11
          WRITE (*,*) 'X',(x(i),I=1,30)
      WRITE (*,*) 'yy',(YY(i),I=1,30)
      WRITE (*,*) 'XXCAL',(XXCAL(i),I=1,30)
      WRITE (*,*) 'XXCAL',(XXCAL(i),I=1,30)

          open (8024, file='yy.dat')
          CLOSE (8024, STATUS = 'DELETE')
          open (8024, file='yy.dat')
   8024 FORMAT (100f10.3)
      if (i_plot-1) 80242 ,80242 ,80241
  80241      WRITE (8024,8024) (YY(i),I=1,n11)
      WRITE (*,*) 'n11',n11,(YY(i),I=1,n11)
```

```
          go to 80243

80242     WRITE (8024,8024) (YY(i),I=1,n)
      WRITE (*,*) 'n',n,(YY(i),I=1,n)
80243 continue
      PAUSE 2000
C     end USEFUL FOR TESTING FOR UP TO 30 WATERLINES

C     OPENING FILES FIRST PART
      OPEN(22,FILE='mat22.OUT')

          rewind 3030
      open(3030,file='mat8.m')
      open (701,file='f.dat')

C     OPENING FILES SECOND PART
      OPEN(102,FILE='mTEST.out')
      open(1033,file='mtest33.out')
          open (1088,file='mplot8.m')
          open (10881,file='mplot1.m')

          write (*,*) '   '
      write (*,*) 'ENTERED FFMATR,   i_plot=   ',i_plot
      write (*,*) N,M,N11,M11
 6603 pause 6603
          PAUSE 100
          PAUSE 100
          PAUSE 100
          N1_SAV=N11
          M1_SAV=M11

          if(i_plot-1) 502,501,502

  501     do 5 i=1,m
    5 write (*,*) X(I),Yy(I),i_plot
          If(IDEB-1) 6606,6605,6606
 6605 pause 6605
 6606 continue

      write (*,*) m,m11,n,n11
          do 6 i=1,M11
    6 write (*,*) FZCAL(i)

          If(IDEB-1) 6608,6607,6608
 6607 pause 6607
 6608 continue

      go to 503
  502 continue
```

```
      do 66 i=1,N
      write (*,*) X(I),Yy(I),i_plot
   66 continue
      DO 67  i=1,N11
   67 write (*,*) FXCAL(i)

           If(IDEB-1) 6610,6609,6610
 6609 pause 6609
 6610 continue

  503    continue

C     CALCULATIONS FIRST PART
C     PREPARE COEFFICIENTS-CALL'MATLAB'-RETURN TO FORTRAN
       WRITE (*,*) 'J_NO',J_NO
      DO 30 I=1,N
           rewind 702
      rewind 704
   30 AAA(I)=1.

      ieq=0
           iineq=0
      do 312 i=1,n
           WRITE (*,*) 'CHECK',I,n,INI(I)
                    If(IDEB-1) 6612,6611,6612
 6611 pause 6611
 6612 continue

           if (ini(i).eq.1) then
           ieq=ieq+1
           iq(ieq)=i
           yeq(ieq)=yy(i)
           go to 311
           endif

           iineq=iineq+1
           inq(iineq)=i
           yineq(iineq)=yy(i)
  311 continue
  312 continue

      M1=3*n-2
      N1=6*n+3
      kk1=9+2*(n-2)

      do 301 j=1,kk1
      ccc(j)=0.
  301 y(j)=0.

      ccc(1)=1
```

```
4041  format ('f=[')
4042  format ('];')
3021  format (8(e15.6,';'))
      write (3030,4041)
      write (3030,3021) (ccc(k),k=1,kk1)
      write (701,400) (ccc(k),k=1,kk1)
           write (3030,4042)

      do 31 j=n+1,2*n
      y(j-n)=yy(j-n)
   31 y(j)=-yy(j-n)
      do 32 j=2*n+1,3*n-2
   32 y(j)=0.

      kk1=3*n-2
 3022 format (8(e15.6,';'))
 4043 format ('b=[')
40431 format ('beq=[')
 4044 format ('];')

      if (iineq.ne.0) then
           go to 4045
           endif

           write (3030,4043)
           zero=0.
           write (3030,3022) (zero,i=1,n-2)
           write (3030,4044)
40471 continue

           if (ieq.ne.0) then
           go to 4046
           endif
           go to 40461

 4045 write (3030,4043)
      fcon=0.
           write (3030,3022) ((yineq(i),-yineq(i)),i=1,iineq) ,
     *(fcon,i=1,n-2)
           write (3030,4044)
      go to 40471
 4046 write (3030,40431)
      write (3030,3022) ((yeq(i)),i=1,ieq)
           write (3030,4044)
           go to 40462
40461 continue
      write (3030,40463)
40462 continue
40463 format ('beq=[ ]')
 4047   continue
```

```
      do 40 i=1,n
      i1=i
      aa(i,1)=-1.
      aa(i,2)=1.
   40 aa(i,3)=-1.
      i2=i1+1
      i4=2*i1
           WRITE (*,*) 'I2',I2,'I4',I4
           do 50 i=i2,i4
      i5=i
      aa(i,1)=-1
      aa(i,2)=-1
   50 aa(i,3)=1
      i51=i5+1
      i5n2=i5+n-2
      WRITE (*,*) 'i51', i51, 'i5n2', i5n2
      do 55 i=i51,i5n2

                    If(IDEB-1) 6614,6613,6614
 6613 pause 6613
 6614 continue

      do 55 j=1,5
   55 aa(i,j)=0.
      do 70 i=1,n
      aa(i,5)=-x(i)
   70 aa(i,4)=x(i)
      do 80 i=i2,i4
      aa(i,4)=-x(i-n)
   80 aa(i,5)=x(i-n)
      do 90 i=1,n
      aa(i,7)=-x(i)**2
   90 aa(i,6)=x(i)**2
      do 100 i=i2,i4
      aa(i,6)=-x(i-n)**2
  100 aa(i,7)=x(i-n)**2
      do 110 i=1,n
      aa(i,9)=-x(i)**3
  110 aa(i,8)=x(i)**3
      do 120 i=i2,i4
      aa(i,8)=-x(i-n)**3
  120 aa(i,9)=x(i-n)**3
      N1M1=3*N+5
      NM=2*N+5
 1000 continue
      OPEN(4444,FILE='MAT3D4444.DAT')

C     PASS DATA TO 'MAT1.DAT'
      WRITE (4444,*) N ,M,NM
```

```
    WRITE (4444,400) (XXCAL(J),J=1,M)
        DO 121 I=1,N
 121 WRITE (4444,400) X(I)

    DO 125 I=3,N
    DO 125 J=10,NM,2
    NUP=2*i-4
    NU=(J/2)-3
    IF ((J-9).GT.NUP) GO TO 125
    AA(I,J)=(X(I)-X(NU))**3.
        write (*,*) I,J,X(I),X(NU),(X(I)-X(NU))**3.,AA(I,J)
                If(IDEB-1) 6616,6615,6616
6615 pause 6615
6616 continue

 125 CONTINUE
    DO 126 I=3,N
    DO 126 J=11,NM,2
    NUP=2*i-4
    NU=(J/2)-3
    IF ((J-9).GT.NUP) GO TO 126
    AA(I,J)=-(X(I)-X(NU))**3.
 126 CONTINUE
    do 127 i=i2+2,i4
    do 127 j=10,nm,2
    nup=2*(i-n)-4
    nu=(j/2)-3
    if ((j-9).gt.nup) go to 127
    aa(i,j)=-(x(i-n)-x(nu))**3.
 127 continue
    do 128 i=i2+2,i4
    do 128 j=11,nm,2
    nup=2*(i-n)-4
    nu=(j/2)-3
    if ((j-9).gt.nup) go to 128
    aa(i,j)=(x(i-n)-x(nu))**3.
 128 continue

C     CALCULATE CONSTANTS FOR SECOND DIFFERENCE
    if (i_plot-1)  66171,66172,66171
66171 n_all=n11
    go to 66173
66172 n_all=n
66173 write (*,*) i_plot,n,n_all

c 6617 pause 6617
 6618 continue

    do 135 i=2,n_all-1
        write (22,*) x(I+1),x(i-1)
```

```
      f1=2./(x(i+1)-x(i-1))
      f2=(y(i+1)-y(i))/(x(i+1)-x(i))
      f3=(y(i)-y(i-1))/(x(i)-x(i-1))
      write (220,*) x(I+1),x(i-1),f1*(f2-f3),i_plot
      IF(i_plot-1) 1351,1351,1352
 1351   CONTINUE
      VAL(i-1)=f1*(f2-f3)
      write (222,*) VAL(i-1),x(I+1),x(i-1),I_PLOT
      write (702,*) VAL(i-1)
      GO TO 1353
 1352   continue
      VAL(i-1)=f1*(f2-f3)
           write (221,*) VAL(i-1),x(I+1),x(i-1),I_PLOt
           write (704,*) VAL(i-1)
 1353 CONTINUE

  135 r(2*n+i-1)=f1*(f2-f3)
      do 136 i=i51,i5n2
  136 write (22,400) r(i)

C     CALCULATE B'S FOR SECOND DIFERRENCES
      do 137 i=i51,i5n2

      aa(i,6)=-2.*r(i)
      aa(i,7)=2.*r(i)
      aa(i,8)=-6.*r(i)*x(i-2*n+1)
           write (*,*) i,r(i),x(i-2*n+1)

                    If(IDEB-1) 6620,6619,6620

 6619 pause 6619
 6620 continue

  137 aa(i,9)=6.*r(i)*x(i-2*n+1)
      do 138 i=2*n+1,3*n-2
      do 138 j=10,nm-2,2
      nup=2*(i-n)-1
      nu=(j/2)-3
      if ((j).ge.nup) go to 138
      aa(i,j)=-6.*r(i)*(x(i-2*n+1)-x(nu))
           aa(i,j+1)=-aa(i,j)
  138 continue

      do 139 i=2*n+1,3*n-2
           max=9+2*(i-2*n)
           do 139 j=10,nm-2
           if(j.gt.max) then
           aa(i,j)=0.
           endif
           continue
```

```
 139 continue

     WRITE (22,400) (CCC(I),I=1,N)
     DO 178 I=1,N
 178 WRITE(22,400) (CC(J,I),J=1,N)
     DO 180 I=1,N
 180 WRITE (22,400) X(I),Y(I)
     WRITE (22,400) (R(I),I=2,N-1)
     DO 190 I=1,N
     DO 190 J=1,N
     RR(I,J)=0.
 190  RRC(I,J)=0.
     DO 1100 I=2,N-1
     RR(I,2)=2.*R(I)
 1100 Rr(I,3)=6.*R(I)*X(I)
     DO 1110 I=3,N-1
     N11=I-2
     DO 1110 J1=1,N11
 1110 RR(I,J1+3)=6.*R(I)*(X(I)-X(J1+1))
     DO 1120 I=1,N
 1120 WRITE (22,400) (RR(I,J),J=1,N)
     CONTINUE
     write (22,500)
     CONTINUE
 1133 format ('A=[')
11133 format ('Aeq=[')
     write (3030,1133)
 1131 format (8(f10.3,';'))
 2231 format (100(e15.6)/';')
     kk1=9+2*(n-2)

          do 11302 ii=1,iineq
     do 1130 i=1,m1
     if (i.eq.inq(ii)) then
          write (3030,2231) (aa(i,j),j=1,kk1)
          write (3030,2232)
          write (3030,2231) (aa(i+n,j),j=1,kk1)
     write (3030,2232)
          endif
 1130   continue
11302 continue

     do 11303 i=2*n+1,m1
     write (3030,2231) (aa(i,j),j=1,kk1)
          write (3030,2232)
11303 continue
     write (3030,21303)

     write (3030,11133)
          do 21302 ii=1,ieq
```

```
      do 2130 i=1,m1
      if (i.eq.iq(ii)) then
      afirst=0.
           write (3030,2231) afirst,(aa(i,j),j=2,kk1)
           write (3030,2232)
           endif
 2130   continue
21302 continue
21303 format (']')

11331 format ('save mtest.dat x -ASCII -double')
11332 format ('save c:\naval\fair.bak8\mtest.dat x -ASCII -double')
11333 format ('save c:\naval\fair.bak8\fval_2d.dat fval -ASCII -double')
11334 format('SI_A=size(A);SI_A=SI_A(1,1);beg=2*((SI_A+2)/3)+1;all=A*x;
     *derr_2=all(1:SI_A);')
      fo=1e-02
11336 format ('load c:\naval\fair.bak8\i_plot.dat')
11337  format ('save derr_2.dat derr_2 -ASCII -double')

11335 FORMAT('if (i_plot==1)')

11338 format('else')
11339 format('save derr_2r.dat derr_2 -ASCII -double')
11340 format('end')
      write (3030,1132) kk1,fo

           write (3030,11331)
      write (3030,11332)
      write (3030,11333)

      write (3030,11334)
      write (3030,11336)
      write (3030,11335)
           write (3030,11337)
           write (3030,11338)
           write (3030,11339)

           write (3030,11340)
c          write (3030,5008)
c 5008 format ('mat8_1')


11351  FORMAT ('COMPUTE DATA FOR PLOT')
11352 FORMAT      ('load c:\naval\fair.bak8\FXCAL.dat')
11353 FORMAT      ('load c:\naval\fair.bak8\FZ.dat')
11354 FORMAT     ('load c:\naval\fair.bak8\FX.dat')
11355 FORMAT     ('load c:\naval\fair.bak8\N1_M1.dat')
11375 FORMAT     (' LIO=[]; DIM=[]; KSI=[]; idd=[]; id_fin=[]')
11356 FORMAT     ('if ((i_plot==2)&(ISENS==2));  LIO=FXCAL;')
11357 FORMAT('id_ref=size(FXCAL);id_ref=id_ref(1,1);DIM=FXCAL;')
```

```
11381 FORMAT('else;end')

11358 FORMAT     ('if ((i_plot==2)&(ISENS~=2)); load c:\naval\fair.bak8\
     *LLENGTHS.dat; LIO=LLENGTHS;')
11359 FORMAT     ('id_ref=size(LLENGTHS)')
11360 FORMAT     ('DIM=FXCAL;else;end')
11361 FORMAT     ('id_ref=id_ref(1,1)')
11362 FORMAT     ('N_DO=N1_M1(1,2)')

11363 FORMAT     ('ALL=A*x')
11364          FORMAT                                   ('for          I=1:N_DO;id
=find(ALL>0);ALL_PLU=ALL(id),ALL_ALL=
     *[DIM ALL_PLU]')
11365 FORMAT     ('SI_ALL=size(ALL_ALL); SI_ALL=SI_ALL(1,1); end')

11366 FORMAT     ('IC=0')
11367 FORMAT     ('for I=1:id_ref; for J=1:SI_ALL')
11368 FORMAT     ('if ISENS~=2;LIO=LLENGTHS(I);else;LIO=FXCAL(I); end')
11369 FORMAT     ('if (ALL_ALL(J,1)==LIO); IC=IC+1; id_fin(IC)=J; else;
     *end; end; end;')

11370 FORMAT     ('idd=id_fin')
11371 FORMAT     ('KSI=ALL_ALL(idd(1:end),2)')

11372 FORMAT     ('if ISENS~=2; plot(KSI,LLENGTHS); hold on;
     * plot(KSI,LLENGTHS);axis equal')
11373 FORMAT    ('else; plot(KSI,FXCAL);hold on; plot(KSI,FXCAL);
     * ;axis equal;end')

 2232 format (';')

 1132 format (2h];/'lb=zeros(',i3,',1);'/
     *'[x,fval,exitflag,output,lambda]='
     *31hlinprog(f,A,b,Aeq,beq,lb,[],[],
     *40hoptimset('Display','iter','maxiter',200,  ,
     *26h'LargeScale','off','TolX',
     ,e10.1,'))')

      open (1031,file='mtest.dat')

          write (22,410) ibeg,iend
  400 FORMAT (80f10.3)
  499 format (80e12.5)
  410 FORMAT (8I8)
  500 FORMAT (15H END OF INPUT'E)

C     GIVE SPECIAL MATLAB ORDERS

      ep=engopen('matlab')
          ti1401=engevalstring(ep,'cd c:\naval\fair.bak8')
```

```
      close (1031)
           rewind 2000

           pause 2800
           ti1400=engevalstring(ep,'mat8')
      pause 2801

      open (1031,file='mtest.dat')
      open (2000,file='fval_2d.dat')
      read(2000,*) fval
      close (2000)
      write (*,*) 'value of fval is     ',fval
      write (2002,*) 'value of fval is     ',fval,I_plot
                  If(IDEB-1) 6622,6621,6622
 6621 pause 6621
           pause 222
      pause 222
           pause 222
           pause 222
 6622 continue

c     END OF FIRST PART

C     BEGINNING OF SECOND PART
      rewind 10881
      open (10881,file='mplot1.m')

           rewind 1033
      open(1033,file='mtest33.out')
      idebug=m
      ncount=nm

      if (i_plot-1) 12192, 12191,12192
12191 idebug=M11
      do 12195 i=1,idebug
      write (*,*) ISENS

           If(IDEB-1) 6624,6623,6624
 6623 pause 6623
 6624 continue

      IF (ISENS-1)13192 ,13192,13191
13191 xcal(i)=FZcal(i)

      If(IDEB-1) 6626,6625,6626
 6625 pause 6625
 6626 continue
```

```
      GO TO 13195

13192 write (*,*) 'STOPS HERE'

      xcal(i)=FHE_CH(i)
           If(IDEB-1) 6628,6627,6628
 6627 pause 6627
 6628 continue
13195 continue
      Write (*,*) xcal(i)
12195 continue
      do 1022 i=1,m
      a(i,4)=XXCAL(i)
 1022 write(*,*) m,a(i,4)

           If(IDEB-1) 6630,6629,6630
 6629 pause 6629
 6630 continue

      go to 1219

12192 idebug=N11
      do 12193 io=1,N1_SAV

      IF (ISENS-1)15192 ,15192,15191

15191   xcal(io)=FXcal(io)
      write (*,*) io,xcal(io),FXCAL(IO)
           If(IDEB-1) 6632,6631,6632
 6631 pause 6631
 6632 continue

      GO TO 15195

15192 xcal(io)=FLE_CH(io)
      write (*,*) io,xcal(io),FLE_CH(io)

           If(IDEB-1) 6634,6633,6634
 6633 pause 6633
 6634 continue

15195  Write (*,*) xcal(i)
12193 write (*,*) XCAL(io),IO,n,n1_sav

           If(IDEB-1) 6636,6635,6636
 6635 pause 6635
 6636 continue

      idebug= N1_SAV
      do 1021 i=2,N-1
```

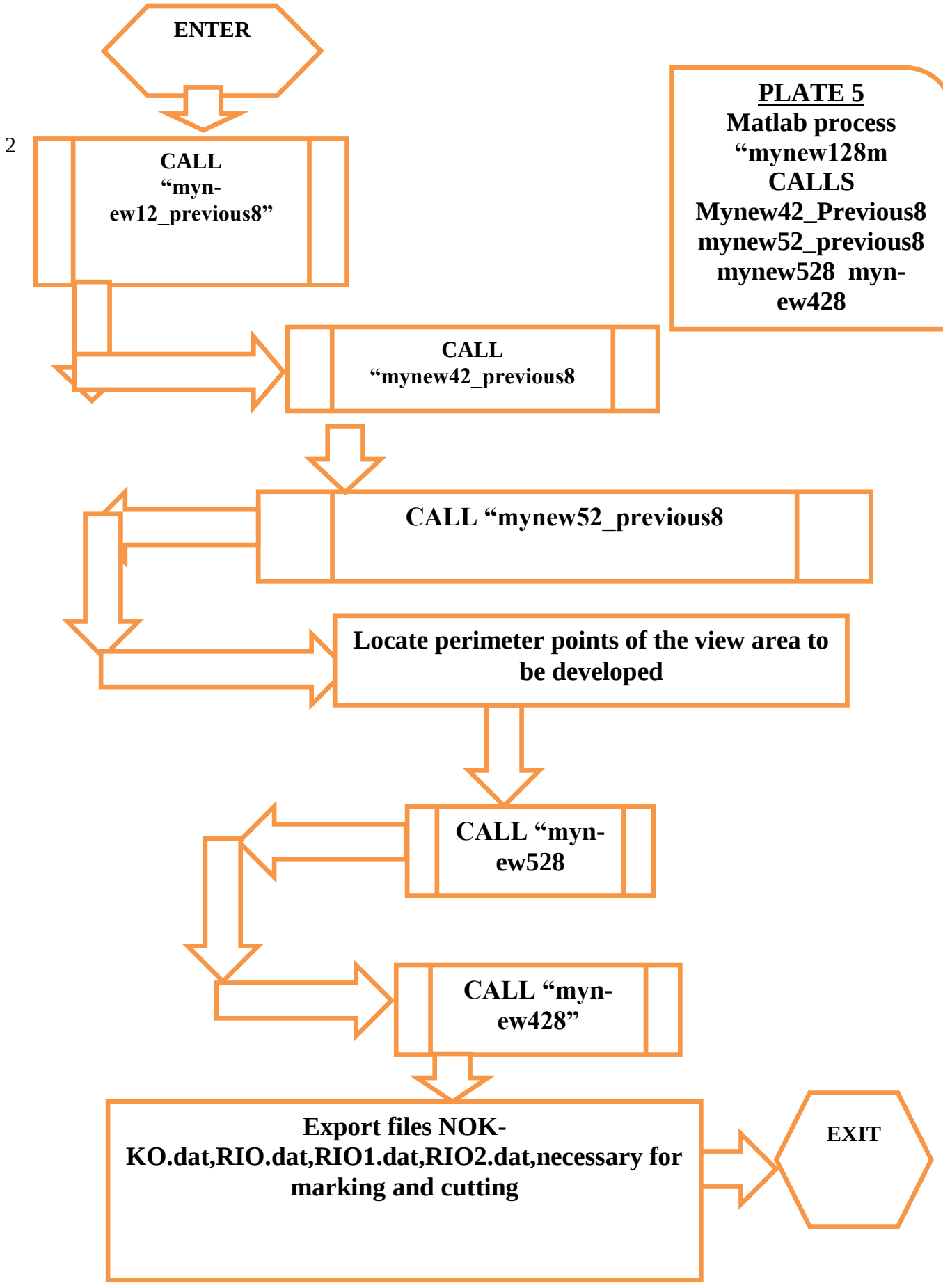
```
    a(i,4)=X(i)
1021 write(*,*) i,N,a(i,4)

        If(IDEB-1) 6638,6637,6638
6637 pause 6637
6638 continue

1219 write (*,*) XCAL(io),i_plot, idebug

        write (102,*) m,idebug,ncount
    WRITE (102,400) (XCAL(I),I=1,IDEBUG)
    WRITE (*,*) (XCAL(I),I=1,IDEBUG)
    pause 6639
6639 pause 6639
6640 continue

        WRITE (102,400) (A(I,4),I=1,M)
    nlim=5+2*m
    do 9 i=1,ncount
    READ (1031,*) XB1(I)
  9 WRITE (102,400) (XB1(I))
        ppp=0.

    do 1001 i=1,idebug
    ddd=xb1(2)-xb1(3)+(xb1(4)-xb1(5))*xcal(i)+
   1((xb1(6)-xb1(7))*xcal(i)**2)+(xb1(8)-xb1(9))*
   2(xcal(i)**3)
    ppp=2.*(xb1(6)-xb1(7))+6*(xb1(8)-xb1(9))*xcal(i)
        sss=(xb1(4)-xb1(5))+2.*(xb1(6)-xb1(7))*xcal(i)+3.*(xb1(8)-xb1(9))
   1*(xcal(i)**2.)
    dd1=0.
        pp1=0.
        ss1=0.
        if(I_PLOT.eq.2)         nlim=5+2*N

    do 801 io=10,nlim,2
    iii=(io-6)/2
        write (*,*)io,iii,nlim,a(iii,4)
    dd1=dd1+(xb1(io)-xb1(io+1))*(.5*((xcal(i)-a(iii,4))**3)
   1+.5*dabs((xcal(i)-a(iii,4))**3))

    pp1=pp1+6*(xb1(io)-xb1(io+1))*(.5*((xcal(i)-a(iii,4)))
   1+.5*dabs((xcal(i)-a(iii,4))))
    ss1=ss1+3.*(xb1(io)-xb1(io+1))*(.5*((xcal(i)-a(iii,4))**2)
   1+.5*dabs((xcal(i)-a(iii,4))**2))
 801 continue

    write (*,*) ppp,pp1,ppp+pp1,xcal(i)-a(iii,4),xcal(i), a(iii,4)

        write (*,*) xcal(i),(xb1(ki),ki=1,20)
```

```
          If(IDEB-1) 6644,6643,6644
 6643 pause 6643
 6644 continue

      dio=ddd+dd1
           p=ppp+pp1
           s=sss+ss1
           ssss(i)=s
C      DDDD(I),PPPP(I),SSSS(I) ARE THE CALCULATED ORDINATE,ITS SEC-
OND
C     AND FIRST DIFFERENCES RESPECTIVELY
      pppp(i)=p
      DDDD(I)=Dio

 1001 write (102,4001) xcal(i),ddd,dd1 ,dio

      write (102,401) (xb1(ii),ii=1,ncount)
      Write (1033,402)
  402 format ('CALCULATED RESULTS'/'----------------'/    W.L.   HALF
     1BREATH  1ST DERRIV  2ND DERRIV'/'-----------------------------')
      DO 901 I=1,IDEBUG
  901 WRITE (1033,400) XCAL(I),DDDD(I),ssss(i),pppp(i)
      WRITE (1033,401) XB1(1)
  401 format (//'    MAX DEVIATION=',F20.10)
 4001 FORMAT (8F10.3)
 5001 FORMAT (15H END OF INPUT'E)

C     PREPARE DATA FOR PLOT in 'mplot8.m'
      WRITE (1088,5021)
      WRITE (1088,5022)
      WRITE (1088,5023)
      WRITE (1088,5024)

      WRITE (1088,5012)
 5012 FORMAT('y=[')
 5021 format ('load c:\naval\fair.bak8\FZO.dat')
 5022 format ('load c:\naval\fair.bak8\FY_GIVEN.dat')
 5023 format ('load c:\naval\fair.bak8\FX.dat')
 5024 format ('load c:\naval\fair.bak8\J_NO.dat')

      write (1088,5016) (dddd(i),i=1,idebug)
           write (1088,5013)
 5013 format(']')

      write (1088,5014)
 5014 format('x=[')
      write (1088,5016) (xcal(i),i=1,idebug)
           write (1088,5013)
      write (1088,6014)
 6014 format('y_given=[')
```

```
      write (1088,5016) (FY_IN(i),i=1,idebug)
           write (1088,5013)

 6015 format('x_given=[')
      write (1088,6015)
      write (1088,5016) (FX_IN(i),i=1,idebug)
           write (1088,5013)

 5016 format(50f12.4)

      if (i_plot.eq.1) then
      write (1088,50171)
           write (1088,6016)
      write (1088,70171)
      write (1088,60171)
      write (1088,60172)
           write (1088,5118)

 1091 format (80e14.3)
           write (1089,1091) (pppp(i),i=2,idebug-1)
           write (*,*) (pppp(i),i=2,idebug-1)

                    If(IDEB-1) 6646,6645,6646
 6645 pause 6645
       pause 5016
 6646 continue
           go to 51172

           else
      write (1088,5017)
      write (1088,6016)
      write (1088,9017)
      write (1088,6017)
      write (1088,5118)
      write (1090,1091) (pppp(i),i=2,idebug-1)
           write (*,*) (pppp(i),i=2,idebug-1)
                    If(IDEB-1) 6648,6647,6648
 6647 pause 6647
 6648 continue

           continue
           endif
51172 continue

50171 format  ('plot(y,x)')
70171 format  (13hplot(y,x,'O')  )
 5017 format ('plot(x,y)')
```

```
 6016 format ('hold on')
60171 format (47hplot(FY_GIVEN(1:end,J_NO),FZO(1:end,J_NO),'r' ))
60172 format (48hplot(FY_GIVEN(1:end,J_NO),FZO(1:end,J_NO),'r^' ))

 9017 format (12hSIX=size(FX))
 6017 format (28hplot(FX,y_given(1:SIX),'r^'))

c      6017 format (20hplot(FX,y_given,'*'))
 5118 format('grid on')

 5119 format ('axis equal')
10000 continue
51201 format('hold off')

      PAUSE
          WRITE (*,*)'     SEE GRAPH FOR STATION NO....  ',J_NO

          ti1401=engevalstring(ep,'cd c:\naval\fair.bak8')
          ti1400=engevalstring(ep,'mplot8')


          pause

c         CLOSE (103, STATUS = 'DELETE')
      pause 100001
          pause 100001
          pause 100001
c     CLOSE (3030, STATUS = 'DELETE')
      CLOSE (3030)
          rewind (1088)
10001 continue
c     CLOSE (1031, STATUS = 'DELETE')
      status= engClose(ep)
          write (*,*) '   '
      M11=M1_SAV
          N11=n1_SAV
      write (*,*) 'EXIT FFMATR'
c     ti1408=engevalstring(ep,'mat8_1')
          RETURN
      END
```

ANNEX(5) + PLATE(5)
DOCUMENTATION OF MATLAB PROCESSES
“mynew128.m”
“mynew12_previous8.m”
“mynew42_previous8.m”
“mynew52_previous8.m”
“mynew528.m”
“mynew428.m”


ENTER
2
CALL "myn-ew12_previous8"
PLATE 5
Matlab process "mynew128m CALLS Mynew42_Previous8 mynew52_previous8 mynew528 myn-ew428
CALL "mynew42_previous8
CALL "mynew52_previous8
Locate perimeter points of the view area to be developed
CALL "myn-ew528
CALL "myn-ew428"
Export files NOK-KO.dat,RIO.dat,RIO1.dat,RIO2.dat,necessary for marking and cutting
EXIT

MATLAB MODULE “mynew128.m”

```
%mynew42
mynew12_previous
mynew52
mynew42
format long
LENGTHS1_fin=[]
HEIGHTS1_fin=[]
LENGTHS_tem=[]
HEIGHTS_tem=[]
%JOIN HEIGHTS PRODUCED BY mynew4 and mynews5 AND SORT BY WA-
TERLINES
SI_HE5=size(HEIGHTS5)
SI_HE5=SI_HE5(1,2)

if SI_HE5==1
HEIGHTS=[HEIGHTS4' HEIGHTS5']
'Channel 1'
else
   ' chanel 2'
 HEIGHTS5=HEIGHTS5'
 HEIGHTS=[HEIGHTS4' HEIGHTS5']
end %end if

%LENGTHS=[LENGTHS4'
    %LENGTHS5']
LENGTHS=[LENGTHS4
    LENGTHS5]
HEIGHTS_KEEP=HEIGHTS
HEIGHTS=HEIGHTS'
HEI=HEIGHTS
[HEIGHTS,id]=sort(HEIGHTS)
HEI1=HEIGHTS
%END JOIN HEIGHTS PRODUCED BY mynew4 and mynews5 AND SORT BY
WATERLINES

%JOIN LENGTHS PRODUCED BY mynew4 and mynews5 AND SORT BY WA-
TERLINES
LENGTHS=[LENGTHS4' LENGTHS5']
LENGTHS_KEEP=LENGTHS
LENGTHS=LENGTHS'
LENGTHS=LENGTHS(id)
%END LENGTHS PRODUCED BY mynew4 and mynews5 AND SORT BY WA-
TERLINES

%JOIN  KKO INDEXES   PRODUCED BY mynew42 and mynews52 AND SORT
BY WATERLINES
NOKKO=[NOKKO4' NOKKO5']
```

```
NOKKO_KEEP=NOKKO
NOKKO=NOKKO'
NOKKO=NOKKO(id)
%END JOIN KKO INDEXES  PRODUCED BY mynew42 and mynews52 AND SORT BY WATERLINES



%PASS COORDINATES OF POLYGON PERIMETER(HEIGHTS_KEEP,LENGTHS_KEEP) TO FORTRAN

LE_KEEP=LENGTHS_KEEP'
HE_KEEP=HEIGHTS_KEEP'
NOKKO_KEEP=NOKKO_KEEP'
KEEP=size(LE_KEEP)
KEEP=KEEP(1,1)
save c:\naval\fair.bak6\LE_KEEP.dat LE_KEEP -ASCII -double
save c:\naval\fair.bak6\HE_KEEP.dat HE_KEEP -ASCII -double
save c:\naval\fair.bak6\NOKKO_KEEP.dat NOKKO_KEEP -ASCII -double
save c:\naval\fair.bak6\KEEP.dat KEEP -ASCII -double
%END PASS COORDINATES OF POLYGON PERIMETER(HEIGHTS_KEEP,LENGTHS_KEEP) TO FORTRAN

SI_HE=size(HEIGHTS)
SI_HE=SI_HE(1,1)

SI_LE=size(LENGTHS)
SI_LE=SI_LE(1,1)

SI_NOKKO=size(NOKKO)
SI_NOKKO=SI_NOKKO(1,1)

%ELIMINATE DOUBLE (EQUAL) VALUES FROM HEIGHTS
HEIGHTS_tem=HEIGHTS

if( HEIGHTS(1)==HEIGHTS(2))
     HEIGHTS(1)= -1

   else

end



for i=2:SI_HE
   if( HEIGHTS(i)==HEIGHTS(i-1))
      continue

   else
```

```
    HEIGHTS_tem(i)= -1


end
end
a=find(HEIGHTS_tem<0)
HEIGHTS1_fin=HEIGHTS(a)
NOKKO1_fin=NOKKO(a)
%END ELIMINATE DOUBLE (EQUAL) VALUES FROM HEIGHTS

SI_HE=size(HEIGHTS1_fin)
SI_HE=SI_HE(1,1)


SI_NOKKO1=size(NOKKO1_fin)
SI_HE=SI_NOKKO1(1,1)


%SORT ACOORDING TO LENGTHS
LENGTHS=sort(LENGTHS)

%ELIMINATE DOUBLE (EQUAL) VALUES FROM LENGHTS
LENGTHS_tem=LENGTHS

if( LENGTHS(1)==LENGTHS(2))
     LENGTHS(1)= -1

   else

end

for i=2:SI_LE
   if( LENGTHS(i)==LENGTHS(i-1))
         continue
      else

    LENGTHS_tem(i)= -1

end
end
a=find(LENGTHS_tem<0)
LENGTHS1_fin=LENGTHS(a)
NOKKO2_fin=NOKKO(a)

%END ELIMINATE DOUBLE (EQUAL) VALUES FROM LENGHTS
```

```
%ELIMINATE DOUBLE (EQUAL) VALUES FROM HEIGHTS
icount_yes=0
icount_no=0
HEIGHTS_tem=HEIGHTS
LENGTHS_tem=LENGTHS
NOKKO_tem=NOKKO
for i=1:SI_HE
   for k=i+1:SI_HE
      [i k HEIGHTS_tem(i) HEIGHTS_tem(k) LENGTHS_tem(i) LENGTHS_tem(k)
NOKKO_tem(i) NOKKO_tem(k) ]
       %pause
   if(                                          (HEIGHTS_tem(i)==HEIGHTS_tem(k))&
(LENGTHS_tem(i)==LENGTHS_tem(k))&(NOKKO_tem(i)==NOKKO_tem(k))     )
     HEIGHTS_tem(i)= -1
       icount_yes=  icount_yes +1
     [i k HEIGHTS_tem(i) HEIGHTS_tem(k) LENGTHS_tem(i) LENGTHS_tem(k)
NOKKO_tem(i) NOKKO_tem(k) ]
       %pause
   else
    icount_no=  icount_no +1
  [i  k  HEIGHTS_tem(i)  HEIGHTS_tem(k)  LENGTHS_tem(i)  LENGTHS_tem(k)
NOKKO_tem(i) NOKKO_tem(k) ]
   %pause
end
end
end
a=find(HEIGHTS_tem>=0)
HEIGHTS1_fin=HEIGHTS(a)
LENGTHS1_fin=LENGTHS(a)
NOKKO1_fin=NOKKO(a)
%END ELIMINATE DOUBLE (EQUAL) VALUES FROM HEIGHTS

SI_HE=size(HEIGHTS1_fin)
SI_HE=SI_HE(1,1)
SI_LE=SI_HE


SI_NOKKO1=size(NOKKO1_fin)
SI_HE=SI_NOKKO1(1,1)


%SORT ACOORDING TO LENGTHS
LENGTHS=sort(LENGTHS)

%ELIMINATE DOUBLE (EQUAL) VALUES FROM LENGHTS

%END ELIMINATE DOUBLE (EQUAL) VALUES FROM LENGHTS
```

```
%save c:\naval\fair.bak6\LLENGTHS.dat LENGTHS1_fin -ASCII -double
%save c:\naval\fair.bak6\HHEIGHTS.dat HEIGHTS1_fin -ASCII -double
%save c:\naval\fair.bak6\NNOKKO.dat NOKKO1_fin -ASCII -double


%save c:\naval\fair.bak6\NN_SI_LE.dat SI_LE -ASCII -double
%save c:\naval\fair.bak6\MM_SI_HE.dat SI_HE -ASCII -double
%save c:\naval\fair.bak6\LENGTHS4.dat LENGTHS4 -ASCII -double
save c:\naval\fair.bak6\HEIGHTS4.dat HEIGHTS4 -ASCII -double
save c:\naval\fair.bak6\LENGTHS5.dat LENGTHS5 -ASCII -double
save c:\naval\fair.bak6\HEIGHTS5.dat HEIGHTS5 -ASCII -double

% PART OF OLD mynew1.m

%format bank
%load c:\naval\fair.bak6\x1cal.dat % tolal long data
%load c:\naval\fair.bak6\z1cal.dat % total vertical data
load c:\naval\fair.bak6\xxlim.dat % lmits of plate of interest
load c:\naval\fair.bak6\zzlim.dat % lmits of plate of interest
load c:\naval\fair.bak6\gendat1.dat % load no of stations (nn2) and waterlines (mm2)
x1cal=LENGTHS1_fin
z1cal=HEIGHTS1_fin
%mm2=gendat1(2)
%nn2=SI_LE
nn2=SI_HE
mm2=SI_HE
%nn2=gendat1(1)nn2
mm2

kko=gendat1(3)
n=kko
z=zeros(5,1)
plot(xxlim,zzlim,'r')
hold on
axis equal
for i=1:nn2
plot(x1cal(i),z1cal,'*')

hold on
end
z1cal=z1cal'
%END PLOT ORIGINAL DATA AND AREA OF INEREST



dub_x1=[]
```

```
% MAKING ARRAY XICAL DOUBLE
for jj=1:mm2

    dub_x1(:,jj)=x1cal
 end
%END MAKING ARRAY XICAL DOUBLE

%MAKING ARRAY ZICAL DOUBLE
%for jjj=1:nn2
dub_z1=[]

for jjj=1:nn2
    dub_z1(1:mm2,jjj)=z1cal

 end

%END MAKING ARRAY ZICAL DOUBLE

%dub_z1=dub_z1'
%du_z1=dub_z1(1:nn2,1:mm2)
%size_x=size(x1cal)
%size_z=size(z1cal)
%size_dif=size_z-size_x

tri=delaunay(xxlim,zzlim)


 out1=tsearch(xxlim,zzlim,tri,[dub_x1],[dub_z1']) % find points within bounds

 point1=isnan(out1)

point1
%save c:\naval\fair.bak6\points.dat point1 -ASCII -double




%PREPARE AND EXPORT SPECIAL DATA ARRAYS FOR NON VERTICAL
VIEW SIDES
%ELIMINATE DOUBLE (EQUAL) VALUES FROM HEIGHTS

icount_yes=0
icount_no=0
HEIGHTS_tem=[HEIGHTS5
   HEIGHTS4]

LENGTHS_tem=[LENGTHS5
   LENGTHS4]
```

```
NOKKO_tem=[NOKKO5
   NOKKO4]

for i=1:SI_HE
   for k=i+1:SI_HE
      [i k HEIGHTS_tem(i) HEIGHTS_tem(k) LENGTHS_tem(i) LENGTHS_tem(k)
NOKKO_tem(i) NOKKO_tem(k) ]
       %pause
   if(                                    (HEIGHTS_tem(i)==HEIGHTS_tem(k))&
(LENGTHS_tem(i)==LENGTHS_tem(k)))&(NOKKO_tem(i)==NOKKO_tem(k)    )
     NOKKO_tem(i)= -1
       icount_yes=  icount_yes +1
     [i k HEIGHTS_tem(i) HEIGHTS_tem(k) LENGTHS_tem(i) LENGTHS_tem(k)
NOKKO_tem(i) NOKKO_tem(k) ]
       %pause
   else
    icount_no=  icount_no +1
  [i k HEIGHTS_tem(i) HEIGHTS_tem(k) LENGTHS_tem(i) LENGTHS_tem(k)
NOKKO_tem(i) NOKKO_tem(k) ]
   %pause
end
end
end
[a NOKKO_fin ]=find(NOKKO_tem>=0)
NOKKO1_fin=NOKKO_tem(a)
HEIGHTS1_fin=HEIGHTS_tem(a)
LENGTHS1_fin=LENGTHS_tem(a)
[ NO_LE_VERT id]=sort(NOKKO1_fin)
 LE_NO_VERT=LENGTHS1_fin(id)
 HE_NO_VERT=HEIGHTS1_fin(id)

SI_ROWS_VERT=size(LE_NO_VERT)
SI_ROWS_VERT=SI_ROWS_VERT(1,1)
SI_COLS_VERT=size(HE_NO_VERT)
SI_COLS_VERT=SI_COLS_VERT(1,1)

% INCLUDE DATA OF FIRST POINT OF FIRST SEGMENT  AS LAST POINT
OF LAST SEGMENT
%LENGTHS1_fin(SI_ROWS_VERT+1)=LENGTHS1_fin(1)
%HEIGHTS1_fin(SI_ROWS_VERT+1)=HEIGHTS1_fin(1)
%NOKKO1_fin(SI_ROWS_VERT+1)=kko
%SI_ROWS_VERT=size(LENGTHS1_fin)
%SI_ROWS_VERT=SI_ROWS_VERT(1,1)
%SI_COLS_VERT=SI_ROWS_VERT
% end INCLUDE DATA OF FIRST POINT OF FIRST SEGMENT  AS LAST
POINT OF LAST SEGMENT

%NEW PART
%SI_COLS_VERT=SI_COLS_VERT+1
%SI_ROWS_VERT=SI_ROWS_VERT+1
```

```
%LE_NO_VERT(SI_ROWS_VERT)=LE_NO_VERT(1)
%HE_NO_VERT(SI_ROWS_VERT)=HE_NO_VERT(1)
%NO_LE_VERT(SI_ROWS_VERT)=NO_LE_VERT(SI_ROWS_VERT-1)
%end NEW PART



%save c:\naval\fair.bak6\N_SI_LE.dat SI_LE -ASCII -double
%save c:\naval\fair.bak6\M_SI_HE.dat SI_HE -ASCII -double
save c:\naval\fair.bak6\SI_ROWS_VERT.dat SI_ROWS_VERT -ASCII -double % FILE NO 1013
 save c:\naval\fair.bak6\SI_COLS_VERT.dat SI_COLS_VERT -ASCII -double % FILE NO 1014
 LE_NO_VERT=LENGTHS1_fin
 HE_NO_VERT=HEIGHTS1_fin
 NO_LE_VERT=NOKKO1_fin
[LENGTHS1_fin id]=sort(LE_NO_VERT)
save c:\naval\fair.bak6\LE_NO_VERT.dat LENGTHS1_fin -ASCII -double
%HEIGHTS1_fin=sort(HE_NO_VERT)
HEIGHTS1_fin=HE_NO_VERT(id)
NO_LE_VERT1_fin=NO_LE_VERT(id)
save c:\naval\fair.bak6\HE_NO_VERT.dat HEIGHTS1_fin -ASCII -double
save c:\naval\fair.bak6\NO_LE_VERT.dat NO_LE_VERT1_fin -ASCII -double
%save c:\naval\fair.bak6\NOKKO.dat NOKKO1_fin -ASCII -double



 %END PREPARE AND EXPORT SPECIAL DATA ARRAYS FOR NON VERTICAL VIEW SIDES



% SELECT DATA FOR MARGINS OF SIGHT VIEW OF ANYPLATE TO BE USED IN FORTRAN
load c:\naval\fair.bak6\xxtop_area.dat % limits of plate of interest
load c:\naval\fair.bak6\yytop_area.dat

lmax=max(xxlim)
lmin=min(xxlim)
hmax=max(zzlim)
hmin=min(zzlim)

%NEW PART
%NOKKO1_fin=NO_LE_VERT
%HEIGHTS1_fin=HE_NO_VERT
%LENGTHS1_fin=LE_NO_VERT
%END NEW PART

si_le_fin=size(LENGTHS1_fin)
```

```
si_le_fin=si_le_fin(1,1)
io=0
ix=0
FL_FIN=[]
LE_NO=[]



for i=1:si_le_fin

   if((LENGTHS1_fin(i)<=lmax)&(LENGTHS1_fin(i)>=lmin))

      io=io+1
       FL_FIN(io)=LENGTHS1_fin(i)
       %LE_NO(io)=NOKKO2_fin(i)
       LE_NO(io)=NOKKO1_fin(i)
   else
     ix=ix+1
  end
end
NOKKO_FL=io
LE_NO1=[FL_FIN
   LE_NO]

cmax=max(zzlim)
cmin=min(zzlim)
si_he_fin=size(HEIGHTS1_fin)
si_he_fin=si_he_fin(1,1)
io=0
ix=0
FH_FIN=[]
HE_NO=[]
for i=1:si_he_fin

   if((HEIGHTS1_fin(i)<=hmax)&(HEIGHTS1_fin(i)>=hmin))
      io=io+1
       FH_FIN(io)=HEIGHTS1_fin(i)
       HE_NO(io)=NOKKO1_fin(i)
   else
     ix=ix+1
  end
end
NOKKO_FH=io
HE_NO1=[FH_FIN
   HE_NO]


save c:\naval\fair.bak6\FL_FIN.dat FL_FIN -ASCII -double %FILE NO 921
save c:\naval\fair.bak6\FH_FIN.dat FH_FIN -ASCII -double  %FILE NO 922
```

```
save c:\naval\fair.bak6\NOKKO_FL.dat NOKKO_FL -ASCII -double  %FILE NO 923
%save c:\naval\fair.bak6\NOKKO_FH.dat NOKKO_FH -ASCII -double  %FILE NO 924
save c:\naval\fair.bak6\LE_NO1.dat LE_NO1 -ASCII -double % NOT USEFUL IN FORTRAN
save c:\naval\fair.bak6\HE_NO1.dat HE_NO1 -ASCII -double  % NOT USEFUL IN FORTRAN
% end SELECT DATA FOR MARGINS OF SIGHT VIEW OF ANYPLATE TO BE USED IN FORTRAN

% RETRIEVE DATA Of LE_NO_VERT , HE_NO_VERT NOKO_LE_VERT FROM FORTRAN AND PLOT
%load c:\naval\fair.bak6\LE_NO_VERT.dat
%load c:\naval\fair.bak6\HE_NO_VERT.dat
%load c:\naval\fair.bak6\NOKO_LE_VERT.dat
plot(LE_NO_VERT,HE_NO_VERT,'o')
% end RETRIEVE DATA Of LE_NO_VERT , HE_NO_VERT NOKO_LE_VERT FROM FORTRAN AND PLOT
load c:\naval\fair.bak6\LE_NO_VERT.dat
load c:\naval\fair.bak6\HE_NO_VERT.dat
load c:\naval\fair.bak6\LLENGTHS.dat
load c:\naval\fair.bak6\HHEIGHTS.dat
```

MATLAB module "mynew12_previous8.m"

```
mynew42_previous8
mynew52_previous8

load c:\naval\fair.bak8\xxlim.dat % lmits of plate of interest
load c:\naval\fair.bak8\zzlim.dat % lmits of plate of interest
load c:\naval\fair.bak8\gendat1.dat % load no of stations (nn2) and waterlines (mm2)
load c:\naval\fair.bak8\x1cal.dat % IMPORT DATA for ALL FRAMES and LONGI-
TUDINALS
load c:\naval\fair.bak8\z1cal.dat
xcal=x1cal
zcal=z1cal
x1cal=[]
z1cal=[]% end IMPORT DATA for ALL FRAMES and LONGITUDINALS

X_MAX=max(xxlim) % KEEP ONLY xcal zcal LYING WITHIN PLATE LIMITS
(xxlim,zzlim)
X_MIN=min(xxlim)
Z_MAX=max(zzlim)
Z_MIN=min(zzlim)
idx=find(xcal<=X_MAX&xcal>=X_MIN)
idz=find(zcal<=Z_MAX&zcal>=Z_MIN)
xcal=xcal(idx)
zcal=zcal(idz)
si_xcal=size(xcal)
si_xcal=si_xcal(1,1)
si_zcal=size(zcal)
si_zcal=si_zcal(1,1)% end KEEP ONLY xcal zcal LYING WITHIN PLATE LIMITS
(xxlim,zzlim)

format long
LENGTHS1_fin=[]
HEIGHTS1_fin=[]
LENGTHS_tem=[]
HEIGHTS_tem=[]
%JOIN HEIGHTS PRODUCED BY mynew4 and mynews5 AND SORT BY WA-
TERLINES
HEIGHTS=[HEIGHTS4' HEIGHTS5']
LENGTHS=[LENGTHS4' LENGTHS5']
HEIGHTS_KEEP=HEIGHTS
HEIGHTS=HEIGHTS'
HEI=HEIGHTS
[HEIGHTS,id]=sort(HEIGHTS)
HEI1=HEIGHTS
HEIGHTS_PREV=HEIGHTS
%END JOIN HEIGHTS PRODUCED BY mynew4 and mynews5 AND SORT BY
WATERLINES
```

```
%JOIN LENGTHS PRODUCED BY mynew4 and mynews5 AND SORT BY WA-
TERLINES
LENGTHS=[LENGTHS4' LENGTHS5']
LENGTHS_KEEP=LENGTHS
LENGTHS=LENGTHS'
LENGTHS=LENGTHS(id)
LENGTHS_PREV=LENGTHS
%END LENGTHS PRODUCED BY mynew4 and mynews5 AND SORT BY WA-
TERLINES

%JOIN  KKO INDEXES  PRODUCED BY mynew42 and mynews52 AND SORT
BY WATERLINES
NOKKO=[NOKKO4' NOKKO5']
NOKKO_KEEP=NOKKO
NOKKO=NOKKO'
NOKKO=NOKKO(id)
NOKKO_PREV=NOKKO
%END JOIN KKO INDEXES  PRODUCED BY mynew42 and mynews52 AND
SORT BY WATERLINES


%PASS COORDINATES OF POLYGON PERIME-
TER(HEIGHTS_KEEP,LENGTHS_KEEP) TO FORTRAN
LE_KEEP=LENGTHS_KEEP'
HE_KEEP=HEIGHTS_KEEP'
NOKKO_KEEP=NOKKO_KEEP'
KEEP=size(LE_KEEP)
KEEP=KEEP(1,1)
save c:\naval\fair.bak8\LE_KEEP.dat LE_KEEP -ASCII -double
save c:\naval\fair.bak8\HE_KEEP.dat HE_KEEP -ASCII -double
save c:\naval\fair.bak8\NOKKO_KEEP.dat NOKKO_KEEP -ASCII -double
save c:\naval\fair.bak8\KEEP.dat KEEP -ASCII -double
%END PASS COORDINATES OF POLYGON PERIME-
TER(HEIGHTS_KEEP,LENGTHS_KEEP) TO FORTRAN

SI_HE1=size(HEIGHTS)
SI_HE1=SI_HE1(1,1)

SI_LE1=size(LENGTHS)
SI_LE1=SI_LE1(1,1)

SI_NOKKO=size(NOKKO)
SI_NOKKO=SI_NOKKO(1,1)

%ELIMINATE DOUBLE (EQUAL) VALUES FROM HEIGHTS
HEIGHTS_tem=HEIGHTS

if( HEIGHTS(1)==HEIGHTS(2))
    HEIGHTS(1)= -1
```

```
   else

end

for i=2:SI_HE1
   if( HEIGHTS(i)==HEIGHTS(i-1))
      continue

   else

    HEIGHTS_tem(i)= -1


end
end
a=find(HEIGHTS_tem<0)
HEIGHTS1_fin=HEIGHTS(a)
NOKKO1_fin=NOKKO(a)
%END ELIMINATE DOUBLE (EQUAL) VALUES FROM HEIGHTS

SI_HE1=size(HEIGHTS1_fin)
SI_HE1=SI_HE1(1,1)

SI_NOKKO1=size(NOKKO1_fin)
SI_HE1=SI_NOKKO1(1,1)

%SORT ACOORDING TO LENGTHS
LENGTHS=sort(LENGTHS)

%ELIMINATE DOUBLE (EQUAL) VALUES FROM LENGHTS
LENGTHS_tem=LENGTHS

if( LENGTHS(1)==LENGTHS(2))
     LENGTHS(1)= -1

   else

end

for i=2:SI_LE1
   if( LENGTHS(i)==LENGTHS(i-1))
         continue
      else

    LENGTHS_tem(i)= -1

end
end
a=find(LENGTHS_tem<0)
LENGTHS1_fin=LENGTHS(a)
```

```
NOKKO2_fin=NOKKO(a)
%END ELIMINATE DOUBLE (EQUAL) VALUES FROM LENGHTS

SI_HE1=size(HEIGHTS1_fin)
SI_HE1=SI_HE1(1,1)

SI_LE1=size(LENGTHS1_fin)
SI_LE1=SI_LE1(1,1)

SI_NOKKO1=size(NOKKO1_fin)
SI_NOKKO1=SI_NOKKO1(1,1)

SI_NOKKO2=size(NOKKO2_fin)
SI_NOKKO2=SI_NOKKO2(1,1)

save c:\naval\fair.bak8\LLENGTHS.dat LENGTHS1_fin -ASCII -double
save c:\naval\fair.bak8\HHEIGHTS.dat HEIGHTS1_fin -ASCII -double
save c:\naval\fair.bak8\NNOKKO.dat NOKKO1_fin -ASCII -double
save c:\naval\fair.bak8\LENGTHS.dat LENGTHS1_fin -ASCII -double
save c:\naval\fair.bak8\HEIGHTS.dat HEIGHTS1_fin -ASCII -double
save c:\naval\fair.bak8\NOKKO.dat NOKKO1_fin -ASCII -double
save c:\naval\fair.bak8\NOKKO.dat NOKKO2_fin -ASCII -double
save c:\naval\fair.bak8\NN_SI_LE.dat SI_LE1 -ASCII -double
save c:\naval\fair.bak8\N_SI_LE.dat SI_LE1 -ASCII -double

save c:\naval\fair.bak8\MM_SI_HE.dat SI_HE1 -ASCII -double
save c:\naval\fair.bak8\M_SI_HE.dat SI_HE1 -ASCII -double
save c:\naval\fair.bak8\LENGTHS4.dat LENGTHS4 -ASCII -double
save c:\naval\fair.bak8\HEIGHTS4.dat HEIGHTS4 -ASCII -double
save c:\naval\fair.bak8\LENGTHS5.dat LENGTHS5 -ASCII -double
save c:\naval\fair.bak8\HEIGHTS5.dat HEIGHTS5 -ASCII -double

x1cal=LENGTHS1_fin
z1cal=HEIGHTS1_fin
nn2=SI_LE1
mm2=SI_HE1
mm2

kko=gendat1(3)
n=kko
z=zeros(5,1)
plot(xxlim,zzlim,'r')
hold on
axis equal
for i=1:nn2
plot(x1cal(i),z1cal,'*')

hold on
end
z1cal=z1cal'
```

```
%END PLOT ORIGINAL DATA AND AREA OF INEREST

% MAKING ARRAY XICAL DOUBLE
for jj=1:mm2

    dub_x1(:,jj)=x1cal
 end
%END MAKING ARRAY XICAL DOUBLE

%MAKING ARRAY ZICAL DOUBLE
for jjj=1:nn2
    dub_z1(1:mm2,jjj)=z1cal

 end
%END MAKING ARRAY ZICAL DOUBLE

tri=delaunay(xxlim,zzlim)

 out1=tsearch(xxlim,zzlim,tri,[dub_x1],[dub_z1]') % find points within bounds

 point1=isnan(out1)

save c:\naval\fair.bak8\points.dat point1 -ASCII -double

LI=[] % PREPARE AND SAVE DATA IN RIO_NEW.dat FOR PLATE PERIME-
TER (TO BE MADE BY 'test.test.test'
HJ=[]
IN=[]
LE45=[LENGTHS4
   LENGTHS5]
HE45=[HEIGHTS4
   HEIGHTS5]
ic=0
six1=size(x1cal)
siz1=size(z1cal)
SI45=size(LE45)
SI45=SI45(1,1)
six1=six1(1,1)
siz1=siz1(1,1)

IN=[]
IN(1:six1,1:siz1)=0
XIN=[]
XIN(1:six1,1:siz1)=0
ZIN=[]
ZIN(1:six1,1:siz1)=0
for i=1:six1
   for j=1:siz1
      for I=1:SI45
         if(LE45(I)==x1cal(i)&HE45(I)==z1cal(j))
```

```
            [ I i j LE45(I) x1cal(i) HE45(I) z1cal(j)]
            %pause
            ic=ic+1
            IN(i,j)=1
            LI(ic)=1
            HJ(ic)=1
          else
       end
   end
end
end
SI_IN=size(IN)
SI_IN1=SI_IN(1,1)
SI_IN2=SI_IN(1,2)
INN=IN(1:SI_IN1,1:SI_IN2)
id=find(INN(end,1:end)==1)
si_id=size(id)
si_id=si_id(1,2)

if(si_id>1)
MIN=id(1)
MAX=id(2)
INN(end,MIN:MAX)=1
else
end
save c:\naval\fair.bak8\RIO_NEW.dat INN -ASCII -double
% end PREPARE AND SAVE DATA IN RIO_NEW.dat FOR PLATE PERIMETER
(TO BE MADE BY 'test.test.test'

HJ=[] % PREPARE AND SAVE DATA IN RIO1_NEW.dat FOR PLATE FRAMES
(TO BE MADE BY 'test.test.test'
HJ=[]
ZE4(1:size(LENGTHS5))=0
LLE4=[LENGTHS4' ZE4]
HHE4=[HEIGHTS4'ZE4]
XIN=[]
XIN(1:six1,1:siz1)=0

for i=1:six1     %size x1cal
   for j=1:siz1  % size z1cal

            for I=1:SI45
          if(LLE4(I)==x1cal(i)&HE45(I)==z1cal(j))

             XIN(i,j)=1

          else

       end
   end
```

```
end
end
end
for j=1:six1

   id=find (XIN(j,1:end )==1)
   MAX=max(id)
   MIN=min(id)
   si_id=size(id)
   si=si_id(1,1)
   for si=2
      XIN(j,MIN:MAX)=1
   %pause
end
end
save c:\naval\fair.bak8\RIO1_NEW.dat XIN -ASCII -double
% end PREPARE AND SAVE DATA IN RIO1_NEW.dat FOR PLATE FRAMES
(TO BE MADE BY 'test.test.test'

HJ=[] % PREPARE AND SAVE DATA IN RIO2_NEW.dat FOR PLATE LONGI-
TUDINALS (TO BE MADE BY 'test.test.test'
HJ=[]
ZE5(1:size(LENGTHS4))=0
LLE5=[ZE5 LENGTHS5' ]
HHE5=[ ZE5 HEIGHTS5']

ZIN=[]
ZIN(1:six1,1:siz1)=0

for i=1:six1     %size x1cal
   for j=1:siz1  % size z1cal

            for I=1:SI45

                  if(HHE5(I)==z1cal(j)&LE45(I)==x1cal(i))
                     ZIN(i,j)=1

                [HHE5(I) z1cal(j) LE45(I) x1cal(i) I i j]
              else

      end
   end
end
end

for j=1:siz1

   id=find (ZIN(1:end,j )==1)

   MAX=max(id)
```

```
   MIN=min(id)
   si_id=size(id)
   si=si_id(1,1)
   for si=2

end
end
save c:\naval\fair.bak8\RIO2_NEW.dat ZIN -ASCII -double
```

MATLAB MODULE ”mynew42_previous8.m”

```
---------------------------------
format long
load c:\naval\fair.bak8\x1cal.dat % tolal long data
load c:\naval\fair.bak8\FXCAL.dat
load c:\naval\fair.bak8\z1cal.dat % total vertical data
load c:\naval\fair.bak8\xxlim.dat % lmits of plate of interest
load c:\naval\fair.bak8\zzlim.dat % lmits of plate of interest
load c:\naval\fair.bak8\gendat1.dat % load no of stations (nn2) and waterlines (mm2)
load c:\naval\fair.bak8\ISENS.dat
mm2=gendat1(2)
nn2=gendat1(1)
kko=gendat1(3)
SI_x1cal=size(x1cal);   SI_x1cal=SI_x1cal(1,1)
SI_z1cal=size(z1cal);   SI_z1cal=SI_z1cal(1,1)

n=kko
% END IMPORT DATA

xxlim(n+1)=xxlim(1)
zzlim(n+1)=zzlim(1)
% FIND LIMITS (X<Z) OF EDGES OF PLATE OF INTEREST
for i=1:n
   x_beg(i)=xxlim(i)
   z_beg(i)=zzlim(i)
   %x_end(i)=xxlim(i+1)+.0001
   %z_end(i)=zzlim(i+1)+.0001
   x_end(i)=xxlim(i+1)
   z_end(i)=zzlim(i+1)
end
% END  FIND LIMITS (X,Z) OF EDGES OF PLATE OF INTEREST

% FIND EXTREME  POINT LIMITS  OF PLATE OF INTEREST PROJECTION
for i=1:n
most_max_z(i)=max(z_beg(i),z_end(i))

end
for i=1:n
most_min_z(i)=min(z_beg(i),z_end(i))

end
for i=1:n
most_max_x(i)=max(x_beg(i),x_end(i))

end
for i=1:n
most_min_x(i)=min(x_beg(i),x_end(i))

end
```

```
% END FIND EXTREME  POINT LIMITS  OF PLATE OF INTEREST PROJEC-
TION

% FIND TANGENTS OF STRAIGHT LINES ENCLOSING POLYGON
  for i=1:n
   theta(i)=(z_end(i)-z_beg(i))/ (x_end(i)-x_beg(i))
end
% END FIND TANGENTS OF STRAIGHT LINES ENCLOSING POLYGON

%FIND Z COORDINATES CORRESPONDING TO X COORD OF POINTS OF
INEREST
%LYING ON PERIMETER OF POLYGON
NX=nn2

SI_FX=size(FXCAL); SI_FX=SI_FX(1,2);

X=[]
if(SI_FX<=SI_x1cal)
X=x1cal'
else
X=FXCAL'
end
X_SAV=X    % end NEW MODIFIED PART 11/3/2011AR

for m=1:kko
for n=1:nn2-1
XX(n,m)=0
end
end

index=0
Z=[]
for kk=1:NX
    for i=1:kko


Z_TEM= z_beg(i)+theta(i)*(X(kk)-x_beg(i))

if Z_TEM<=most_max_z(i)

  if (Z_TEM>=most_min_z(i))&(theta(i)~=NaN)
        if (X(kk)<=most_max_x(i))&(X(kk)>=most_min_x(i))

    XX(kk,i)=X(kk)
    Z(kk,i)=Z_TEM

    N_KK(kk,i)=kk
    N_I(kk,i)=i

end
```

```
end

end

end

end
%END FIND Z COORDINATES CORRESPONDING TO X COORD OF POINTS
OF INEREST
%LYING ON PERIMETER OF POLYGON

%PLACE nan IN WHOLE ZZ ARRAY
for m=1:kko
for n=1:nn2
ZZ(n,m)=nan
end
end
% END PLACE nan IN WHOLE ZZ ARRAY

for m=1:kko
   kk=find(Z(1:end,m)>0)
   si_kk=size(kk)
      ZZ(1:si_kk,m)=Z(kk,m)
   end

ZZZ=[z_beg
z_end
ZZ]
% EQUALIZE SIZE OF ARRAYS XX AND N_I WITH ARRAY Z
% N_KK and N_I already equalized
cow=size(Z)
XX=XX(1:cow,1:end)
N_I=N_I(1:cow,1:end)
% END EQUALIZE SIZE OF ARRAYS XX AND N_I WITH ARRAY Z

%TRANSFORM Z , XX  AND N_I ARRRAYS TO VECTORS
Z_VEC=Z(:)
XX_VEC=XX(:)
N_KK_VEC=N_KK(:)
N_I_VEC=N_I(:)
% END TRANSFORM Z , XX  AND N_I ARRRAYS TO VECTORS

k=size(Z_VEC)

%TRANSFORM ARRAYS X_VEC and ZZ_VEC to  VECTORS , ELIMINATE
ZEROS
%AND PUT THEM  IN ASCENDING ORDER of STATIONS
y=find(Z_VEC>0)
x=Z_VEC(y)
```

```
IN_N_KK_VEC=N_KK_VEC(y)
n=size(x)

y1=find(XX_VEC>0)
x1=XX_VEC(y1)
IN_N_I_VEC=N_I_VEC(y1)

[xx,id]=sort(x)
xx1=x1(id)
NKK=IN_N_KK_VEC(id)
NI=IN_N_I_VEC(id)
xxx=xx
xxx1=xx1
[xxx NI xxx1 NKK]
% END TRANSFORM ARRAYS X_VEC and ZZ_VEC to  VECTORS , ELIMI-
NATE ZEROS
%AND PUT THEM  IN ASCENDING ORDER of STATIONS

LENGTHS4=xxx1
HEIGHTS4=xxx
NOKKO4=NI

LE=LENGTHS4*10000
LENGTHS4=round(LE)/10000

HE=HEIGHTS4*10000
HEIGHTS4=round(HE)/10000

plot(LENGTHS4, HEIGHTS4,'*')

LLL_KP=LENGTHS4
HHH_KP=HEIGHTS4
```

MATLAB_MODULE "mynew52_previous8.m"

```
format long
load c:\naval\fair.bak8\x1cal.dat % tolal long data
load c:\naval\fair.bak8\z1cal.dat % total vertical data
load c:\naval\fair.bak8\FXCAL.dat % tolal requested long data
load c:\naval\fair.bak8\FZCAL.dat% tolal requested vert data
load c:\naval\fair.bak8\xxlim.dat % lmits of plate of interest
load c:\naval\fair.bak8\zzlim.dat % lmits of plate of interest
load c:\naval\fair.bak8\gendat1.dat % load no of stations (nn2) and waterlines (mm2)
mm2=gendat1(2)
nn2=gendat1(1)
kko=gendat1(3)
n=kko
% END IMPORT DATA
xxlim(n+1)=xxlim(1)
zzlim(n+1)=zzlim(1)
% FIND LIMITS (X,Z) OF EDGES OF PLATE OF INTEREST
x_beg=[]
   z_beg=[]
   x_end=[]
   z_end=[]
for i=1:n
   x_beg(i)=xxlim(i)
   z_beg(i)=zzlim(i)
   %x_end(i)=xxlim(i+1)+.0001
   %z_end(i)=zzlim(i+1)+.0001
   x_end(i)=xxlim(i+1)
   z_end(i)=zzlim(i+1)
end
% END  FIND LIMITS (X,Z) OF EDGES OF PLATE OF INTEREST

% FIND EXTREME  POINT LIMITS  OF PLATE OF INTEREST PROJECTION
for i=1:n
most_max_z(i)=max(z_beg(i),z_end(i))
end
for i=1:n
most_min_z(i)=min(z_beg(i),z_end(i))
end
for i=1:n
most_max_x(i)=max(x_beg(i),x_end(i))
end
for i=1:n
most_min_x(i)=min(x_beg(i),x_end(i))
end
% END FIND EXTREME  POINT LIMITS  OF PLATE OF INTEREST PROJEC-
TION
```

```
% FIND TANGENTS OF STRAIGHT LINES ENCLOSING POLYGON
  for i=1:n
   theta(i)=(z_end(i)-z_beg(i))/ (x_end(i)-x_beg(i)+0.00001)
end
% END FIND TANGENTS OF STRAIGHT LINES ENCLOSING POLYGON

%FIND X COORDINATES CORRESPONDING TO Z COORD OF POINTS OF
INEREST
%LYING ON PERIMETER OF POLYGON
SI_FZ=size(FZCAL); SI_FZ=SI_FZ(1,2)

MZ=SI_FZ
    if(SI_FZ<=SI_z1cal)
     Z=z1cal
 else
      Z=FZCAL'
 end
 Z_SAV=Z    % end  NEW MODIFIED PART 11/3/2011

for m=1:kko
for n=1:mm2
ZZ(n,m)=0
end
end
index=0
X=[]
absmin=min(most_min_x)
absmax= max(most_max_x)

for kk=1:MZ
    for i=1:kko

if(x_beg(i)==x_end(i))
     X_TEM=x_beg(i)
 else
X_TEM= x_beg(i)+1/(theta(i)+.00001)*(Z(kk)-z_beg(i))

end

if
X_TEM>=most_min_x(i)&X_TEM<=most_max_x(i)&Z(kk)>=most_min_z(i)&Z(kk
)<=most_max_z(i)

   [i kk X_TEM most_min_x(i)most_max_x(i)Z(kk) most_min_z(i) most_max_z(i)]

    ZZ(kk,i)=Z(kk)
    X(kk,i)=X_TEM
    M_KK(kk,i)=kk
    M_J(kk,i)=i
    'CORRCT POINT'
```

```
   [i kk X_TEM ]

   else
   'WRONG POINT'
   [i kk X_TEM absmin absmax]

end

end

end

end
%END FIND X COORDINATES CORRESPONDING TO Z COORD OF POINTS
OF INEREST
%LYING ON PERIMETER OF POLYGON

%PLACE nan IN WHOLE XX ARRAY
for m=1:kko
for n=1:nn2
XX(n,m)=nan
end
end
% END PLACE nan IN WHOLE XX ARRAY

for m=1:kko
   kk=find(X(1:end,m)>0)
   m
   si_kk=size(kk)

   XX(1:si_kk,m)=X(kk,m)

end
XX(5,4)=nan

% EQUALIZE SIZE OF ARRAY ZZ WITH ARRAY X
cow=size(X)
ZZ=ZZ(1:cow,1:end)
M_J=M_J(1:cow,1:end)
% END EQUALIZE SIZE OF ARRAY ZZ WITH ARRAY X

%TRANSFORM X and ZZ ARRRAYS TO VECTORS
%X_VEC=[]
X_VEC=X(:)
ZZ_VEC=ZZ(:)
M_KK_VEC=M_KK(:)
M_J_VEC=M_J(:)
% END TRANSFORM X and ZZ ARRRAYS TO VECTORS

k=size(X_VEC)
```

```
%TRANSFORM ARRAYS X_VEC and ZZ_VEC to  VECTORS , ELIMINATE ZEROS
%AND PUT THEM  IN ASCENDING ORDER of STATIONS

y=find(X_VEC>0)
x=X_VEC(y)
IN_M_KK_VEC=M_KK_VEC(y)
n=size(x)

y1=find(ZZ_VEC>0)
x1=ZZ_VEC(y1)
IN_M_J_VEC=M_J_VEC(y1)

[xx,id]=sort(x)
xx1=x1(id)
MJ=IN_M_J_VEC(id)
xxx=xx
xxx1=xx1
{xxx xxx1}
% END TRANSFORM ARRAYS X_VEC and ZZ_VEC to  VECTORS , ELIMI-NATE ZEROS
%AND PUT THEM  IN ASCENDING ORDER of STATIONS

LENGTHS5=xxx
HEIGHTS5=xxx1
NOKKO5=MJ


LE=LENGTHS5*10000
LENGTHS5=round(LE)/10000

HE=HEIGHTS5*10000
HEIGHTS5=round(HE)/10000
plot(LENGTHS5, HEIGHTS5,'*')

%LENGTHS AND HEIGHTS ARE COUPLES OF COORDINATES OF PLATE PERIMETER POINTS
%SORTED WITH STATIONS
```

MODULE "mynew528"

```
----------------
format long
load c:\naval\fair.bak8\x1cal.dat % tolal long data
load c:\naval\fair.bak8\z1cal.dat % total vertical data
load c:\naval\fair.bak8\FXCAL.dat % tolal reqqueted long data
load c:\naval\fair.bak8\FZCAL.dat % total requested vertical data

load c:\naval\fair.bak8\xxlim.dat % lmits of plate of interest
load c:\naval\fair.bak8\zzlim.dat % lmits of plate of interest
load c:\naval\fair.bak8\xelim.dat % lmits of plate of interest
load c:\naval\fair.bak8\zelim.dat % lmits of plate of interest

load c:\naval\fair.bak8\gendat1.dat % load no of stations (nn2) and waterlines (mm2)
mm2=gendat1(2)
nn2=gendat1(1)
kko=gendat1(3)



% END IMPORT DATA
xxlim(n+1)=xxlim(1)
zzlim(n+1)=zzlim(1)

% FIND LIMITS (X,Z) OF EDGES OF PLATE OF INTEREST
x_beg=[]
   z_beg=[]
   x_end=[]
   z_end=[]
for i=1:kko
   x_beg(i)=xxlim(i)
   z_beg(i)=zzlim(i)
   %x_end(i)=xxlim(i+1)+.0001
   %z_end(i)=zzlim(i+1)+.0001
    x_end(i)=xxlim(i+1)
   z_end(i)=zzlim(i+1)
end
% END  FIND LIMITS (X,Z) OF EDGES OF PLATE OF INTEREST

% FIND EXTREME  POINT LIMITS  OF PLATE OF INTEREST PROJECTION
for i=1:kko
most_max_z(i)=max(z_beg(i),z_end(i))
end
for i=1:kko
most_min_z(i)=min(z_beg(i),z_end(i))
end
for i=1:kko
most_max_x(i)=max(x_beg(i),x_end(i))
end
```

```
for i=1:kko
most_min_x(i)=min(x_beg(i),x_end(i))
end
% END FIND EXTREME  POINT LIMITS  OF PLATE OF INTEREST PROJEC-
TION

% FIND TANGENTS OF STRAIGHT LINES ENCLOSING POLYGON
theta(1:kko)=nan
  for i=1:kko
      %if(z_beg(i)~=z_end(i))&(x_beg(i)~=x_end(i))
       if(x_beg(i)~=x_end(i))
   %theta(i)=(z_end(i)-z_beg(i))/ (x_end(i)-x_beg(i)+0.00001)
   theta(i)=(z_end(i)-z_beg(i))/ (x_end(i)-x_beg(i))
else
end
end
% END FIND TANGENTS OF STRAIGHT LINES ENCLOSING POLYGON

%FIND X COORDINATES CORRESPONDING TO Z COORD OF POINTS OF
INEREST
%LYING ON PERIMETER OF POLYGON
%MZ=mm2
if (SI_FZ<=SI_z1cal)
MZ=SI_z1cal
Z=z1cal'
else
Z=FZCAL
MZ=SI_FZ
end

for m=1:kko

for n=1:mm2
%XX(n,m)=0
ZZ(n,m)=0
end
end

index=0
X=[]

absmin=min(most_min_x)
absmax= max(most_max_x)

    for i=1:kko

     for kk=1:MZ
        if(theta(i)~=inf)
%X_TEM(kk,i)= x_beg(i)+1/(theta(i)+.001)*(Z(kk)-z_beg(i))
X_TEM(kk,i)= x_beg(i)+1/(theta(i))*(Z_SAV(kk)-z_beg(i))
```

```
M_J(kk,i)=i
else

    'CHECK RESULT'


%else
 %X_TEM(1,i)= x_beg(i)
 %X_TEM(2,i)= x_beg(i)
 %end
%[X_TEM max_x]
%if X_TEM<=most_max_x(i)
    %if X_TEM>=most_min_x(i)
  %if X_TEM<= absmax
    %if X_TEM>= absmin
    %if ((X_TEM<= absmax)&(X_TEM>= absmin))

   ZZ(kk,i)=Z(kk)
   X(kk,i)=X_TEM

   M_J(kk,i)=i

end

end

end
HEIGHTS5=[]
LENGTHS5=[]
NOKKO5=[]

for i=1:kko
x_max=max(xxlim(i), xelim(i))
    x_min=min(xxlim(i), xelim(i))
    z_max=max(zzlim(i), zelim(i))
    z_min=min(zzlim(i), zelim(i))
  xe_max=max(xxlim(i), xelim(i))
    xe_min=min(xxlim(i), xelim(i))
    ze_max=max(zzlim(i), zelim(i))
    ze_min=min(zzlim(i), zelim(i))
X_TEM1=(10^4*X_TEM)
X_TEM2=round(X_TEM1)
X_TEM3=X_TEM2/10^4
%end
%id=find(X_TEM3(1:end,i)<=x_max&X_TEM3(1:end,i)>=x_min&Z(i)<=z_max&Z
(i)>=z_min)
id=find(X_TEM3(1:end,i)<=x_max&X_TEM3(1:end,i)>=x_min)
X_TEM4=X_TEM3(id,i)
%LENGTHS5=[LENGTHS5  X_TEM4]
LENGTHS5=[LENGTHS5
```

```
   X_TEM4]
Z_TEM4=Z(id)
%HE=[HEIGHTS5  Z_TEM4]
HE=[HEIGHTS5'  Z_TEM4]
HEIGHTS5=HE'
M_J4=M_J(id,i)
%NOKKO5=[NOKKO5  M_J4]
NOKKO5=[NOKKO5
   M_J4]
[X_TEM4 Z_TEM4' ]
plot (X_TEM4,Z_TEM4,'*')
hold on
%pause
end
HEIGHTS5=HE
%END FIND X COORDINATES CORRESPONDING TO Z COORD OF POINTS
OF INEREST
%LYING ON PERIMETER OF POLYGON
```

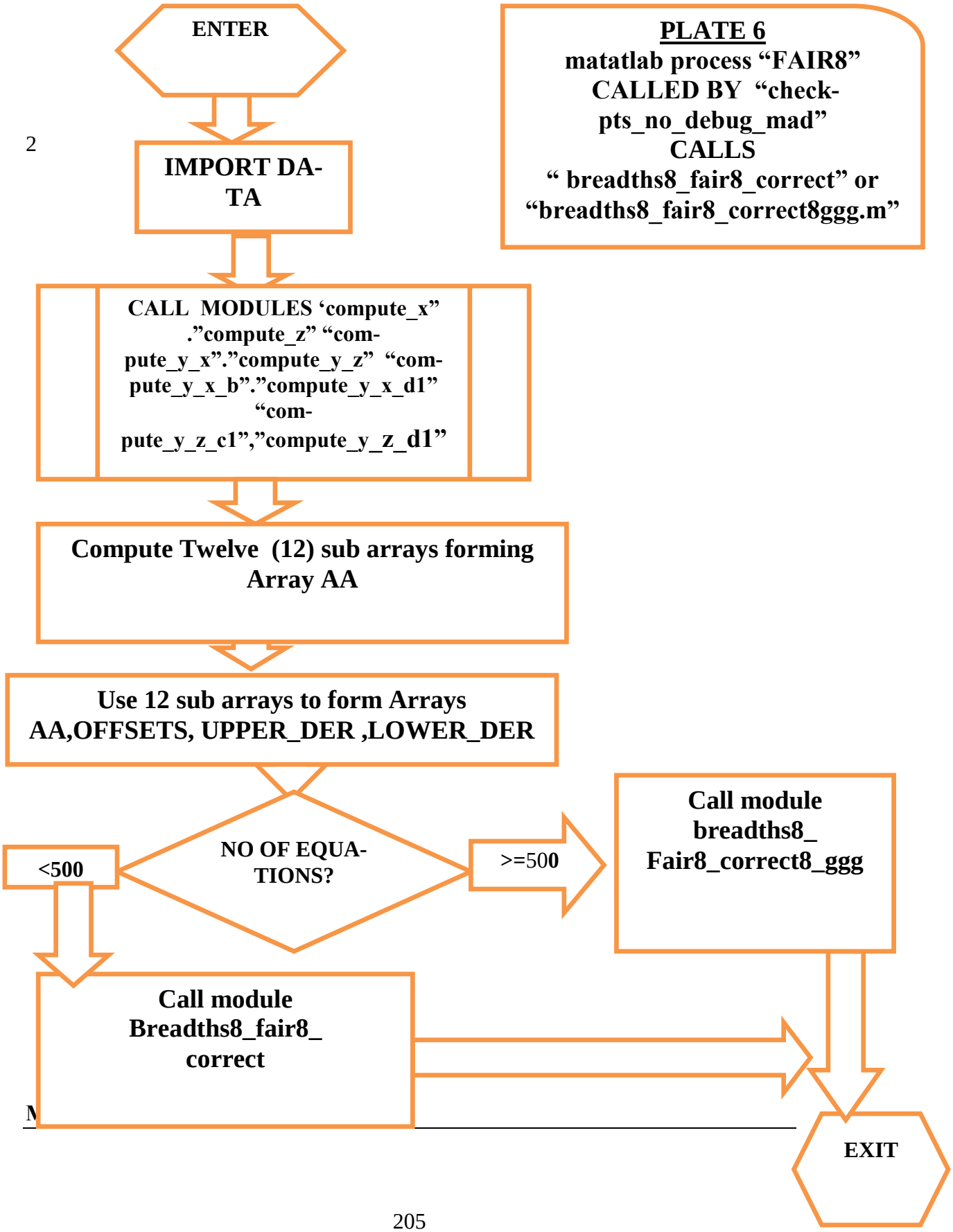
MATLAB MODULE "mynew428"

```
-----------------------
format long
load c:\naval\fair.bak8\x1cal.dat % tolal long data
load c:\naval\fair.bak8\z1cal.dat % total vertical data
load c:\naval\fair.bak8\FXCAL.dat % tolal requested long data
load c:\naval\fair.bak8\FZCAL.dat % total requested vertical data

load c:\naval\fair.bak8\xxlim.dat % lmits of plate of interest
load c:\naval\fair.bak8\zzlim.dat % lmits of plate of interest
load c:\naval\fair.bak8\gendat1.dat % load no of stations (nn2) and waterlines (mm2)
mm2=gendat1(2)
nn2=gendat1(1)
kko=gendat1(3)

load c:\naval\fair.bak8\FXCAL.dat
load c:\naval\fair.bak8\FZCAL.dat
%x1cal=FXCAL
%z1cal=FZCAL
x1cal=x1cal
z1cal=z1cal

n=kko
% END IMPORT DATA
xxlim(n+1)=xxlim(1)
zzlim(n+1)=zzlim(1)

% FIND LIMITS (X<Z) OF EDGES OF PLATE OF INTEREST
for i=1:n
   x_beg(i)=xxlim(i)
   z_beg(i)=zzlim(i)
   %x_end(i)=xxlim(i+1)+.0001
   %z_end(i)=zzlim(i+1)+.0001
   x_end(i)=xxlim(i+1)
   z_end(i)=zzlim(i+1)
end
% END  FIND LIMITS (X,Z) OF EDGES OF PLATE OF INTEREST

% FIND EXTREME  POINT LIMITS  OF PLATE OF INTEREST PROJECTION
for i=1:n
%most_max_z(i)=max(z_beg(i),z_end(i))-0.0001
most_max_z(i)=max(z_beg(i),z_end(i))

end
for i=1:n
%most_min_z(i)=min(z_beg(i),z_end(i))+0.0001
most_min_z(i)=min(z_beg(i),z_end(i))

end
```

```
for i=1:n
%most_max_x(i)=max(x_beg(i),x_end(i))-0.0001
most_max_x(i)=max(x_beg(i),x_end(i))

end
for i=1:n
%most_min_x(i)=min(x_beg(i),x_end(i))+0.0001
most_min_x(i)=min(x_beg(i),x_end(i))

end
% END FIND EXTREME  POINT LIMITS  OF PLATE OF INTEREST PROJEC-
TION

% FIND TANGENTS OF STRAIGHT LINES ENCLOSING POLYGON
theta(1:kko)=nan
  for i=1:kko
      %if(z_beg(i)~=z_end(i))&(x_beg(i)~=x_end(i))
       if(x_beg(i)~=x_end(i))
   %theta(i)=(z_end(i)-z_beg(i))/ (x_end(i)-x_beg(i)+0.00001)
   theta(i)=(z_end(i)-z_beg(i))/ (x_end(i)-x_beg(i))
else
end
end

% END FIND TANGENTS OF STRAIGHT LINES ENCLOSING POLYGON

%FIND Z COORDINATES CORRESPONDING TO X COORD OF POINTS OF
INEREST
%LYING ON PERIMETER OF POLYGON
NX=nn2
%X=x1cal'
X=FXCAL
for m=1:kko
for n=1:nn2-1
%XX(n,m)=nan
XX=[]
end
end
N_I=[]
index=0
Z=[]

for kk=1:NX
    for i=1:kko

     Z_TEM= z_beg(i)+theta(i)*(X_SAV(kk)-x_beg(i))

if Z_TEM<=most_max_z(i)
     %if (Z_TEM>=most_min_z(i))&(theta(i)==0)
  if (Z_TEM>=most_min_z(i))&(theta(i)~=NaN)
```

```
        if (X_SAV(kk)<=most_max_x(i))&(X_SAV(kk)>=most_min_x(i))

    XX(kk,i)=X_SAV(kk)
    Z(kk,i)=Z_TEM

    N_KK(kk,i)=kk
    N_I(kk,i)=i
    [ i]
    %pause
end
end
end
end
end
%END FIND Z COORDINATES CORRESPONDING TO X COORD OF POINTS
OF INEREST
%LYING ON PERIMETER OF POLYGON

%PLACE nan IN WHOLE ZZ ARRAY
for m=1:kko
for n=1:nn2

ZZ(n,m)=nan
end
end
% END PLACE nan IN WHOLE ZZ ARRAY
ll=find(Z>0)
ZZ=Z(ll)
NNI=N_I(ll)
XX=XX(ll)
LENGTHS4=XX
HEIGHTS4=ZZ
NOKKO4=NNI
plot (LENGTHS4,HEIGHTS4,'*')
```

ANNEX(6) + PLATE(6)
DO CUMENTATION OF MATLAB PROCESS
“FAIR8.m”


ENTER
PLATE 6
matatlab process "FAIR8"
CALLED BY "check-
pts_no_debug_mad"
CALLS
" breadths8_fair8_correct" or
"breadths8_fair8_correct8ggg.m"
IMPORT DA-
TA
CALL MODULES 'compute_x"
."compute_z" "com-
pute_y_x"."compute_y_z" "com-
pute_y_x_b"."compute_y_x_d1"
"com-
pute_y_z_c1","compute_y_z_d1"
Compute Twelve (12) sub arrays forming
Array AA
Use 12 sub arrays to form Arrays
AA,OFFSETS, UPPER_DER ,LOWER_DER
NO OF EQUA-
TIONS?
<500
>=500
Call module
breadths8_
Fair8_correct8_ggg
Call module
Breadths8_fair8_
correct
EXIT

```
load c:\naval\fair.bak8\ndf.dat
load c:\naval\fair.bak8\IFF.dat
load c:\naval\fair.bak8\new_fy.dat
load c:\naval\fair.bak8\nio.dat
load c:\naval\fair.bak8\A.dat
SI_A=size(A)
I_PRO=SI_A(1,1)

save c:\naval\fair.bak8\I_PRO.dat I_PRO -ASCII -double
save c:\matlabr12\bin\win32\I_PRO.dat I_PRO -ASCII -double

lb=zeros(ndf,1);

load c:\naval\fair.bak8\FY_GIVEN.dat
load c:\naval\fair.bak8\FX.dat
load c:\naval\fair.bak8\FZ.dat
load c:\naval\fair.bak8\constants1.dat
load c:\naval\fair.bak8\z.dat
load c:\naval\fair.bak8\x.dat
load c:\naval\fair.bak8\r.dat
load c:\naval\fair.bak8\s.dat
load c:\naval\fair.bak8\ISENS.dat
co=constants1
nm=co(1); nm2=co(2); nstav=co(3); nb=co(4); nc=co(5); nd=co(6); neq=co(7);
np=co(8); ns=co(9);
nx=co(10);nq=co(11); nv=co(12); nstav=co(13); nbf=co(14); ncf=co(15); ndf=co(16);
npf=co(17); nsf=co(18);
nxf=co(19); nqf=co(20); nrf=co(21); nm2=co(22); nqfe=co(23); nrfe=co(24);
NATO=co(25);
N=(nb/2)+2;
M=(nc/2)+2;

%NEW_PART SPECIAL CASE if nio=2
if (nio==2)
load c:\naval\fair.bak8\FY_GIVEN.dat
new_fy=[]
new_fy=FY_GIVEN(1:end,1:end)
SI_FY=size(new_fy)
N=SI_FY(1,2); M=SI_FY(1,1);

x=FX
z=FZ
nm=N*M
nm2=2*nm
else
end
%end NEW_PART SPECIAL CASE if nio=2

b=[]
b(1:2*M*N+(N-2)*M+ (M-2)*N,1)=0
```

```
b(1:M*N,1)=new_fy(:)
b(M*N+1:2*M*N,1)=-new_fy(:)
AB=[]
beq=[]
RS_COMPUTE % COMPUTE parameters r and s
RS_S=s
RS_R=r

% PREPARE 'r_all'
rr1=[];
%rr1=r(1:(N-2),2:end); rrr1=rr1(:);
rr1=r(1:(N-2),2:end);  rr1=rr1',rrr1=rr1(:);

si_rrr1=size(rrr1); si_rrr1=si_rrr1(1,1)
r_all=[];

for I1=1:33
   r_all(1:si_rrr1,I1)=rrr1
end
% end PREPARE 'r_all'

% PREPARE 's_all'
ss1=[]

%ss1=s(1:end,2:end); sss1=ss1(:);
ss1=s(1:end,2:end); ss1=ss1'; sss1=ss1(:);

si_sss1=size(sss1); si_sss1=si_sss1(1,1)
s_all=[];

for I1=1:33
   s_all(1:si_sss1,I1)=sss1
end
% end PREPARE 's_all'

Y_X(1:(N-2)*M,1:ndf)=0;
Y_Z(1:(M-2)*N,1:ndf)=0;

AA=[];

AA(1:nm,1)=-1; AA(1:nm,2)=1; AA(1:nm,3)=-1;
AA(nm+1:nm2,1)=-1; AA(nm+1:nm2,2)=-1; AA(nm+1:nm2,3)=1;
AA(nm2+1:nrfe,1:5)=0;

for I=1:N
    for J=1:M
    K=  (I-1)*M+J;
    AA(K,5)=-z(J);
    AA(K,4)=z(J);
end
```

```
end

for I=nm+1:nm2
    AA(I,4)=-AA(I-nm,4);
    AA(I,5)=-AA(I-nm,5);
end %AA's of nm2 raws and of 5 columns

for I=1:N
    for J=1:M
K=(I-1)*M+J;

    AA(K,7)=-z(J)^2;
    AA(K,6)=z(J)^2;
end
end %AA's OF FIRST  nm raws and of 7 columns

 for I=nm+1:nm2
    AA(I,6)=-AA(I-nm,6);
    AA(I,7)=-AA(I-nm,7);
end %AA's of FROM NM TO NM2 raws and of 7 columns

 for I=1:N
    for J=1:M
    K=(I-1)*M+J;
    AA(K,9)=-z(J)^3;
    AA(K,8)=z(J)^3;
end
end%AA's ofOF FIRST  nm raws and of 9 columns

   for I=nm+1:nm2
       AA(I,8)=-AA(I-nm,8);
       AA(I,9)=-AA(I-nm,9);
   end %AA's of FROM NM TO NM2 raws and of 9 columns

    for I=1:N
      for J=1:M
       AA((I-1)*M +J,11)=-x(I);  AA((I-1)*M+J,10)=x(I);
  end
end

     for I=nm+1:nm2
       AA(I,11)=-AA(I-nm,11);  AA(I,10)=-AA(I-nm,10);
  end

   for I=1:N
      for J=1:M
       AA((I-1)*M+J,13)=-x(I)*z(J); AA((I-1)*M+J,12)=x(I)*z(J);
  end
end
```

```
 for I=nm+1:nm2
      AA(I,13)=-AA(I-nm,13); AA(I,12)=-AA(I-nm,12);
  end

for I=1:N
     for J=1:M
          AA((I-1)*M+J,15)=-(z(J)^2)*x(I); AA((I-1)*M+J,14)=(z(J)^2)*x(I);
end
end
     for I=nm+1:nm2
          AA(I,15)=-AA(I-nm,15);AA(I,14)=-AA(I-nm,14);
end

  for I=1:N
     for J=1:M
          DD((I-1)*M+J,17)=-(z(J)^3);          DD((I-1)*M+J,16)=(z(J)^3);          AA((I-
1)*M+J,17)=-(z(J)^3)*x(I); AA((I-1)*M+J,16)=(z(J)^3)*x(I);
end
end
     for I=nm+1:nm2
          AA(I,17)=-AA(I-nm,17); AA(I,16)=-AA(I-nm,16);
end

for I=1:N
     for J=1:M
          AA((I-1)*M+J,19)=-(x(I)^2); AA((I-1)*M+J,18)=x(I)^2;
    %[I M J  x(I)  x(I)^2  (I-1)*M+J  AA((I-1)*M+J,18)]
    %pause
end
end
     for I=nm+1:nm2
          AA(I,19)=-AA(I-nm,19);AA(I,18)=-AA(I-nm,18);
end

   for I=1:N
     for J=1:M
            AA((I-1)*M+J,21)=-(x(I)^2)*z(J); AA((I-1)*M+J,20)=(x(I)^2)*z(J);
  end
end
       for I=nm+1:nm2
          AA(I,21)=-AA(I-nm,21);  AA(I,20)=-AA(I-nm,20);
end

    for I=1:N
     for J=1:M

     AA((I-1)*M+J,23)=-(x(I)^2)*(z(J)^2);
     AA((I-1)*M+J,22)= (x(I)^2)*(z(J)^2);
  end
end
```

```
 for  I=nm+1:nm2
     AA(I,23)=-AA(I-nm,23); AA(I,22)=-AA(I-nm,22);
  end

 for I=1:N
    for J=1:M

     AA((I-1)*M+J,25)=-(x(I)^2)*(z(J)^3);
  AA((I-1)*M+J,24)=(x(I)^2)*(z(J)^3);
end
end

     for I=nm+1:nm2
          AA(I,25)=-AA(I-nm,25);AA(I,24)=-AA(I-nm,24);
end

 for I=1:N
     for J=1:M

     AA((I-1)*M+J,27)=-x(I)^3;
     AA((I-1)*M+J,26)=x(I)^3;
end
end

 for I=nm+1:nm2
          AA(I,27)=-AA(I-nm,27);AA(I,26)=-AA(I-nm,26);
end


for I=1:N
     for J=1:M

     AA((I-1)*M+J,29)=-(x(I)^3)*z(J);
     AA((I-1)*M+J,28)=(x(I)^3)*z(J);
  end
end

     for I=nm+1:nm2
           AA(I,29)=-AA(I-nm,29); AA(I,28)=-AA(I-nm,28);
  end

    for I=1:N
     for J=1:M

     AA((I-1)*M+J,31)=-(x(I)^3)*(z(J)^2);
     AA((I-1)*M+J,30)=(x(I)^3)*(z(J)^2);
     [x(I) ]
        [z(J)]
```

```
        TEST=  (x(I)^3)*(z(J)^2) -(z(J)^2)* (x(I)^3)
       [ (I-1)*M+J] ;
    end
end

    for I=nm+1:nm2
          AA(I,31)=-AA(I-nm,31); AA(I,30)=-AA(I-nm,30);
  end

for  I=1:N
    for J=1:M

    AA((I-1)*M+J,33)=-(x(I)^3)*(z(J)^3);
    AA((I-1)*M+J,32)=(x(I)^3)*(z(J)^3);
end
end
    for I=nm+1:nm2
                AA(I,33)=-AA(I-nm,33); AA(I,32)=-AA(I-nm,32);
end
% end FIRST PART

COMPUTE_B    %OUTPUT IN  BBB_FIN

COMPUTE_C   %OUTPUT IN  CCC_FIN

COMPUTE_D   %OUTPUT IN  DDD_FIN
% FIRST PART+ COMPUTE_B+COMPUTE_C+COMPUTE_ D CAN FORM THE
OFFSETS PART OF A
%OFFSETS=[AA(1:2*N*M,1:33) BBB_FIN CCC_FIN DDD_FIN];
OFFSETS=[AA(1:nm2,1:33) BBB_FIN CCC_FIN DDD_FIN];
COMPUTE_Y_X % OUTPUT IN  Y_X
COMPUTE_Y_Z % OUTPUT IN  Y_Z
COMPUTE_Y_X_B % OUTPUT BBBB_DER  & CCC_DER(all zeros)
COMPUTE_Y_Z_C1 % OUTPUT CCCC_Z_DER1 & BBB_DER_Z(all zeros)
COMPUTE_Y_X_D1 % OUTPUT DIO_X_DER1
COMPUTE_Y_Z_D1  % OUTPUT DIO_Z_DER1

UPPER_DER=[Y_X  BBBB_DER CCC_DER  DIO_X_DER2]; %UPPER PART OF
SECOND (X) DER
LOWER_DER=[Y_Z     BBB_DER_Z     CCCC_Z_DER1              DIO_Z_DER2]
;%LOWER PART OF SECOND (Z) DER

AA=[OFFSETS
   UPPER_DER
   LOWER_DER];
save c:\naval\fair.bak8\AA.dat AA -ASCII -double

load c:\naval\fair.bak8\A.dat

% SELECT WICH PROGRAM TO RUN
```

```
S_AA=size(AA)
S_AA=S_AA(1,1)
I_PRO=[]
if(S_AA<500)
   'NO OF EQUATIONS LESS THAN 500'
   'FURTHER ACTION WITH breadths8_fair8_correct'
   'PRESS RETURN TO CONTINUE'
   %pause
   I_PRO=1

   save c:\naval\fair.bak8\I_PRO.dat I_PRO -ASCII -double
   breadths8_fair8_correct_1

else
   'NO OF EQUATIONS MORE THAN 500'
   'FURTHER ACTION WITH breadths8_fair8_correct8ggg'
   'PRESS RETURN TO CONTINUE'
  % pause
   I_PRO=2
   save c:\naval\fair.bak8\I_PRO.dat I_PRO -ASCII -double
   breadths8_fair8_correct8ggg

end
SI_A=size(A)
I_PRO=SI_A(1,1)

save c:\naval\fair.bak8\I_PRO.dat I_PRO -ASCII -double
save c:\matlabr12\bin\win32\I_PRO.dat I_PRO -ASCII -double

lb=zeros(ndf,1);

load c:\naval\fair.bak8\FY_GIVEN.dat
load c:\naval\fair.bak8\FX.dat
load c:\naval\fair.bak8\FZ.dat
load c:\naval\fair.bak8\constants1.dat
load c:\naval\fair.bak8\z.dat
load c:\naval\fair.bak8\x.dat
load c:\naval\fair.bak8\r.dat
load c:\naval\fair.bak8\s.dat
load c:\naval\fair.bak8\ISENS.dat
co=constants1
nm=co(1); nm2=co(2); nstav=co(3); nb=co(4); nc=co(5); nd=co(6); neq=co(7);
np=co(8); ns=co(9);
nx=co(10);nq=co(11); nv=co(12); nstav=co(13); nbf=co(14); ncf=co(15); ndf=co(16);
npf=co(17); nsf=co(18);
nxf=co(19); nqf=co(20); nrf=co(21); nm2=co(22); nqfe=co(23); nrfe=co(24);
NATO=co(25);
N=(nb/2)+2;
M=(nc/2)+2;
```

```
%NEW_PART SPECIAL CASE if nio=2
if (nio==2)
load c:\naval\fair.bak8\FY_GIVEN.dat
new_fy=[]
new_fy=FY_GIVEN(1:end,1:end)
SI_FY=size(new_fy)
N=SI_FY(1,2); M=SI_FY(1,1);

x=FX
z=FZ
nm=N*M
nm2=2*nm
else
end
%end NEW_PART SPECIAL CASE if nio=2

b=[]
b(1:2*M*N+(N-2)*M+ (M-2)*N,1)=0
b(1:M*N,1)=new_fy(:)
b(M*N+1:2*M*N,1)=-new_fy(:)
AB=[]
beq=[]
RS_COMPUTE % COMPUTE parameters r and s
RS_S=s
RS_R=r

% PREPARE 'r_all'
rr1=[];
rr1=r(1:(N-2),2:end); rrr1=rr1(:);
si_rrr1=size(rrr1); si_rrr1=si_rrr1(1,1)
r_all=[];

for I1=1:33
r_all(1:si_rrr1,I1)=rrr1
end
% end PREPARE 'r_all'

% PREPARE 's_all'
ss1=[]
ss1=s(1:end,2:end); sss1=ss1(:);
si_sss1=size(sss1); si_sss1=si_sss1(1,1)
s_all=[];

for I1=1:33
s_all(1:si_sss1,I1)=sss1
end
% end PREPARE 's_all'

Y_X(1:(N-2)*M,1:ndf)=0;
Y_Z(1:(M-2)*N,1:ndf)=0;
```

```
AA=[];

AA(1:nm,1)=-1; AA(1:nm,2)=1; AA(1:nm,3)=-1;
AA(nm+1:nm2,1)=-1; AA(nm+1:nm2,2)=-1; AA(nm+1:nm2,3)=1;
AA(nm2+1:nrfe,1:5)=0;

for I=1:N
for J=1:M
K=  (I-1)*M+J;
AA(K,5)=-z(J);
AA(K,4)=z(J);
end
end

for I=nm+1:nm2
AA(I,4)=-AA(I-nm,4);
AA(I,5)=-AA(I-nm,5);
end %AA's of nm2 raws and of 5 columns

for I=1:N
for J=1:M
K=(I-1)*M+J;

AA(K,7)=-z(J)^2;
AA(K,6)=z(J)^2;
end
end %AA's OF FIRST  nm raws and of 7 columns

for I=nm+1:nm2
AA(I,6)=-AA(I-nm,6);
AA(I,7)=-AA(I-nm,7);
end %AA's of FROM NM TO NM2 raws and of 7 columns

for I=1:N
for J=1:M
K=(I-1)*M+J;
AA(K,9)=-z(J)^3;
AA(K,8)=z(J)^3;
end
end%AA's ofOF FIRST  nm raws and of 9 columns

for I=nm+1:nm2
AA(I,8)=-AA(I-nm,8);
AA(I,9)=-AA(I-nm,9);
end %AA's of FROM NM TO NM2 raws and of 9 columns

for I=1:N
for J=1:M
AA((I-1)*M +J,11)=-x(I);  AA((I-1)*M+J,10)=x(I);
```

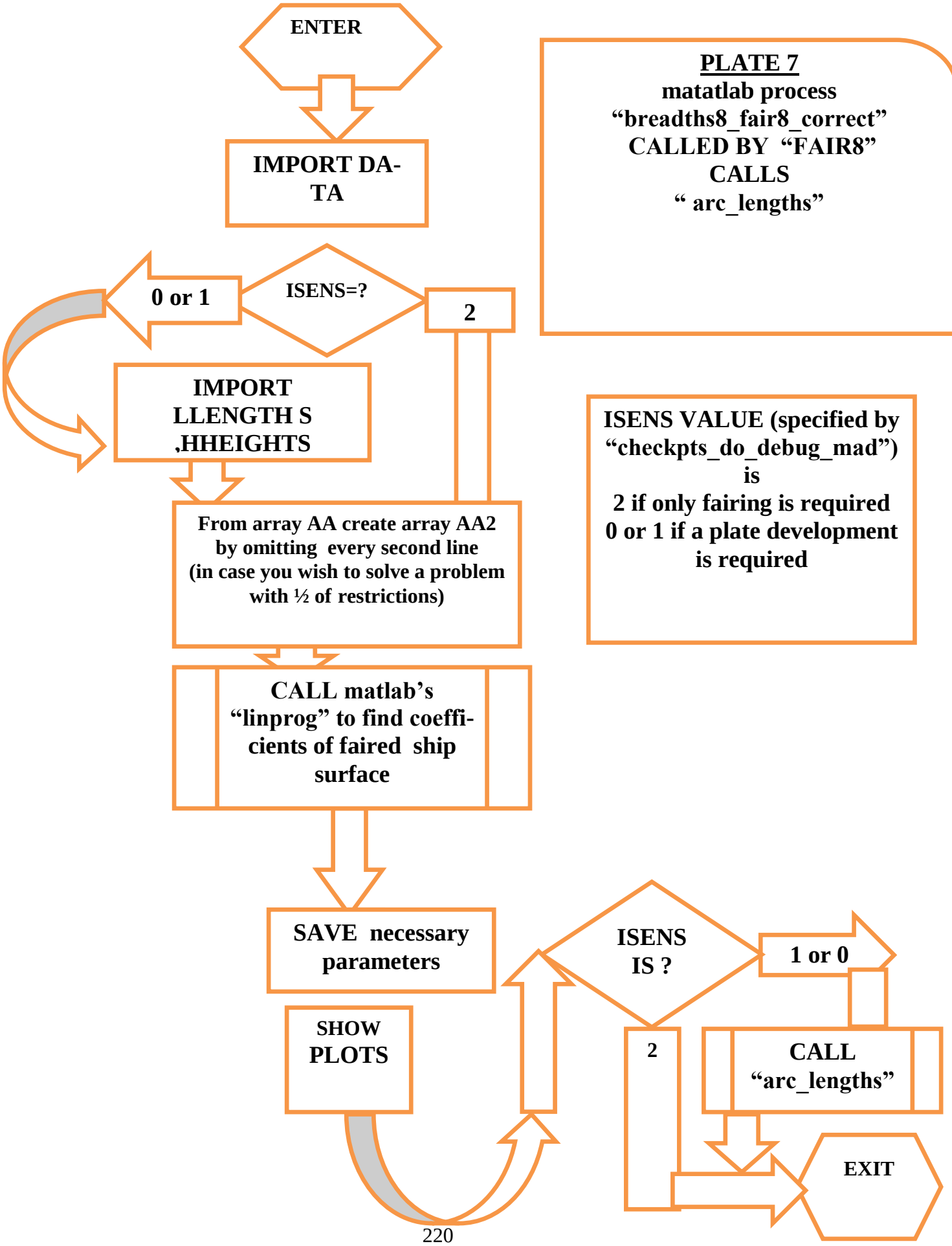
```
end
end

for I=nm+1:nm2
AA(I,11)=-AA(I-nm,11);  AA(I,10)=-AA(I-nm,10);
end

for I=1:N
for J=1:M
AA((I-1)*M+J,13)=-x(I)*z(J); AA((I-1)*M+J,12)=x(I)*z(J);
end
end

for I=nm+1:nm2
AA(I,13)=-AA(I-nm,13); AA(I,12)=-AA(I-nm,12);
end

for I=1:N
for J=1:M
AA((I-1)*M+J,15)=-(z(J)^2)*x(I); AA((I-1)*M+J,14)=(z(J)^2)*x(I);
end
end
for I=nm+1:nm2
AA(I,15)=-AA(I-nm,15);AA(I,14)=-AA(I-nm,14);
end

for I=1:N
for J=1:M
DD((I-1)*M+J,17)=-(z(J)^3);     DD((I-1)*M+J,16)=(z(J)^3);     AA((I-1)*M+J,17)=-
(z(J)^3)*x(I); AA((I-1)*M+J,16)=(z(J)^3)*x(I);
end
end
for I=nm+1:nm2
AA(I,17)=-AA(I-nm,17); AA(I,16)=-AA(I-nm,16);
end

for I=1:N
for J=1:M
AA((I-1)*M+J,19)=-(x(I)^2); AA((I-1)*M+J,18)=x(I)^2;
%[I M J  x(I)  x(I)^2  (I-1)*M+J  AA((I-1)*M+J,18)]
%pause
end
end
for I=nm+1:nm2
AA(I,19)=-AA(I-nm,19);AA(I,18)=-AA(I-nm,18);
end

for I=1:N
for J=1:M
AA((I-1)*M+J,21)=-(x(I)^2)*z(J); AA((I-1)*M+J,20)=(x(I)^2)*z(J);
```

```
end
end
for I=nm+1:nm2
AA(I,21)=-AA(I-nm,21);  AA(I,20)=-AA(I-nm,20);
end

for I=1:N
for J=1:M

AA((I-1)*M+J,23)=-(x(I)^2)*(z(J)^2);
AA((I-1)*M+J,22)= (x(I)^2)*(z(J)^2);
end
end


for  I=nm+1:nm2
AA(I,23)=-AA(I-nm,23); AA(I,22)=-AA(I-nm,22);
end

for I=1:N
for J=1:M

AA((I-1)*M+J,25)=-(x(I)^2)*(z(J)^3);
AA((I-1)*M+J,24)=(x(I)^2)*(z(J)^3);
end
end

for I=nm+1:nm2
AA(I,25)=-AA(I-nm,25);AA(I,24)=-AA(I-nm,24);
end

for I=1:N
for J=1:M

AA((I-1)*M+J,27)=-x(I)^3;
AA((I-1)*M+J,26)=x(I)^3;
end
end

for I=nm+1:nm2
AA(I,27)=-AA(I-nm,27);AA(I,26)=-AA(I-nm,26);
end


for I=1:N
for J=1:M

AA((I-1)*M+J,29)=-(x(I)^3)*z(J);
AA((I-1)*M+J,28)=(x(I)^3)*z(J);
end
```

```
end

for I=nm+1:nm2
AA(I,29)=-AA(I-nm,29); AA(I,28)=-AA(I-nm,28);
end

for I=1:N
for J=1:M

AA((I-1)*M+J,31)=-(x(I)^3)*(z(J)^2);
AA((I-1)*M+J,30)=(x(I)^3)*(z(J)^2);
[x(I) ]
[z(J)]
TEST=  (x(I)^3)*(z(J)^2) -(z(J)^2)* (x(I)^3)
[ (I-1)*M+J] ;
end
end

for I=nm+1:nm2
AA(I,31)=-AA(I-nm,31); AA(I,30)=-AA(I-nm,30);
end

for  I=1:N
for J=1:M

AA((I-1)*M+J,33)=-(x(I)^3)*(z(J)^3);
AA((I-1)*M+J,32)=(x(I)^3)*(z(J)^3);
end
end
for I=nm+1:nm2
AA(I,33)=-AA(I-nm,33); AA(I,32)=-AA(I-nm,32);
end
% end FIRST PART

COMPUTE_B    %OUTPUT IN  BBB_FIN

COMPUTE_C   %OUTPUT IN  CCC_FIN

COMPUTE_D   %OUTPUT IN  DDD_FIN
% FIRST PART+ COMPUTE_B+COMPUTE_C+COMPUTE_ D CAN FORM THE
OFFSETS PART OF A
%OFFSETS=[AA(1:2*N*M,1:33) BBB_FIN CCC_FIN DDD_FIN];
OFFSETS=[AA(1:nm2,1:33) BBB_FIN CCC_FIN DDD_FIN];
COMPUTE_Y_X % OUTPUT IN  Y_X
COMPUTE_Y_Z % OUTPUT IN  Y_Z
COMPUTE_Y_X_B % OUTPUT BBBB_DER  & CCC_DER(all zeros)
COMPUTE_Y_Z_C1 % OUTPUT CCCC_Z_DER1 & BBB_DER_Z(all zeros)
COMPUTE_Y_X_D1 % OUTPUT DIO_X_DER1
COMPUTE_Y_Z_D1  % OUTPUT DIO_Z_DER1
```

```
UPPER_DER=[Y_X  BBBB_DER CCC_DER  DIO_X_DER2]; %UPPER PART OF SECOND (X) DER
LOWER_DER=[Y_Z     BBB_DER_Z     CCCC_Z_DER1          DIO_Z_DER2] ;%LOWER PART OF SECOND (Z) DER

AA=[OFFSETS
UPPER_DER
LOWER_DER];
save c:\naval\fair.bak8\AA.dat AA -ASCII -double

load c:\naval\fair.bak8\A.dat

% SELECT WICH PROGRAM TO RUN
S_AA=size(AA)
S_AA=S_AA(1,1)
I_PRO=[]
if(S_AA<500)
'NO OF EQUATIONS LESS THAN 500'
'FURTHER ACTION WITH breadths8_fair8_correct'
'PRESS RETURN TO CONTINUE'
%pause
I_PRO=1
save c:\naval\fair.bak8\I_PRO.dat I_PRO -ASCII -double
breadths8_fair8_correct
else
'NO OF EQUATIONS MORE THAN 500'
'FURTHER ACTION WITH breadths8_fair8_correct8ggg'
'PRESS RETURN TO CONTINUE'
% pause
I_PRO=2
save c:\naval\fair.bak8\I_PRO.dat I_PRO -ASCII -double
breadths8_fair8_correct8ggg

end
```

ANNEX(7) + PLATE(7)
DOCUMENTATION OF MATLAB PROCESS
“breadths8_fair8_correct.m”


ENTER
IMPORT DA-
TA
ISENS=?
0 or 1
2
IMPORT
LLENGTH S
,HHEIGHTS
From array AA create array AA2
by omitting every second line
(in case you wish to solve a problem
with ½ of restrictions)
CALL matlab's
"linprog" to find coeffi-
cients of faired ship
surface
SAVE necessary
parameters
SHOW
PLOTS
ISENS
IS ?
1 or 0
2
CALL
"arc_lengths"
EXIT
PLATE 7
matatlab process
"breadths8_fair8_correct"
CALLED BY "FAIR8"
CALLS
" arc_lengths"
ISENS VALUE (specified by
"checkpts_do_debug_mad")
is
2 if only fairing is required
0 or 1 if a plate development
is required

MATLAB MODULE "breadths8_fair8_correct1.m"

```
----------------------------------------
format long
load c:\naval\fair.bak8\ndf.dat
load c:\naval\fair.bak8\A.dat
load c:\naval\fair.bak8\nm.dat
load c:\naval\fair.bak8\zzgiven400.dat
load c:\naval\fair.bak8\GENDAT1.dat
load c:\naval\fair.bak8\CONSTANTS1.dat
load c:\naval\fair.bak8\FZCAL.dat
load c:\naval\fair.bak8\FXCAL.dat
load c:\naval\fair.bak8\FY_GIVEN.dat
load c:\naval\fair.bak8\FX.dat
load c:\naval\fair.bak8\FZO.dat
load c:\naval\fair.bak8\new_fy.dat
load c:\naval\fair.bak8\ISENS.dat
    if (ISENS<2)

load c:\naval\fair.bak8\HHEIGHTS.dat
load c:\naval\fair.bak8\LLENGTHS.dat
LLENGTHS1=LLENGTHS
HHEIGHTS1=HHEIGHTS
else
end

n=CONSTANTS1(26)
m=CONSTANTS1(27)

AA_KEEP1=AA;

lb=zeros(ndf,1);
x0=zeros(ndf,1);

   AA1=[]
   AA1=[AA(1:end,1:end)];
   b1=[b(1:end,1:end)];

   % ATEMPT TO DECREASE NO OF EQUATIONS (option)
   SI_AA1=size(AA1)
   SI_A1_LI= SI_AA1(1,1)
   SI_A1_CO= SI_AA1(1,2)
   SI_b1=size(b1)
   SI_b1=SI_b1(1,1)
   L=0
   b2=[]
   AA2=[]
   for I=1:2:SI_A1_LI
      L=L+1
      AA2(L,1:SI_A1_CO)=AA1(I,1:SI_A1_CO);
      b2(L)=b1(I);
```

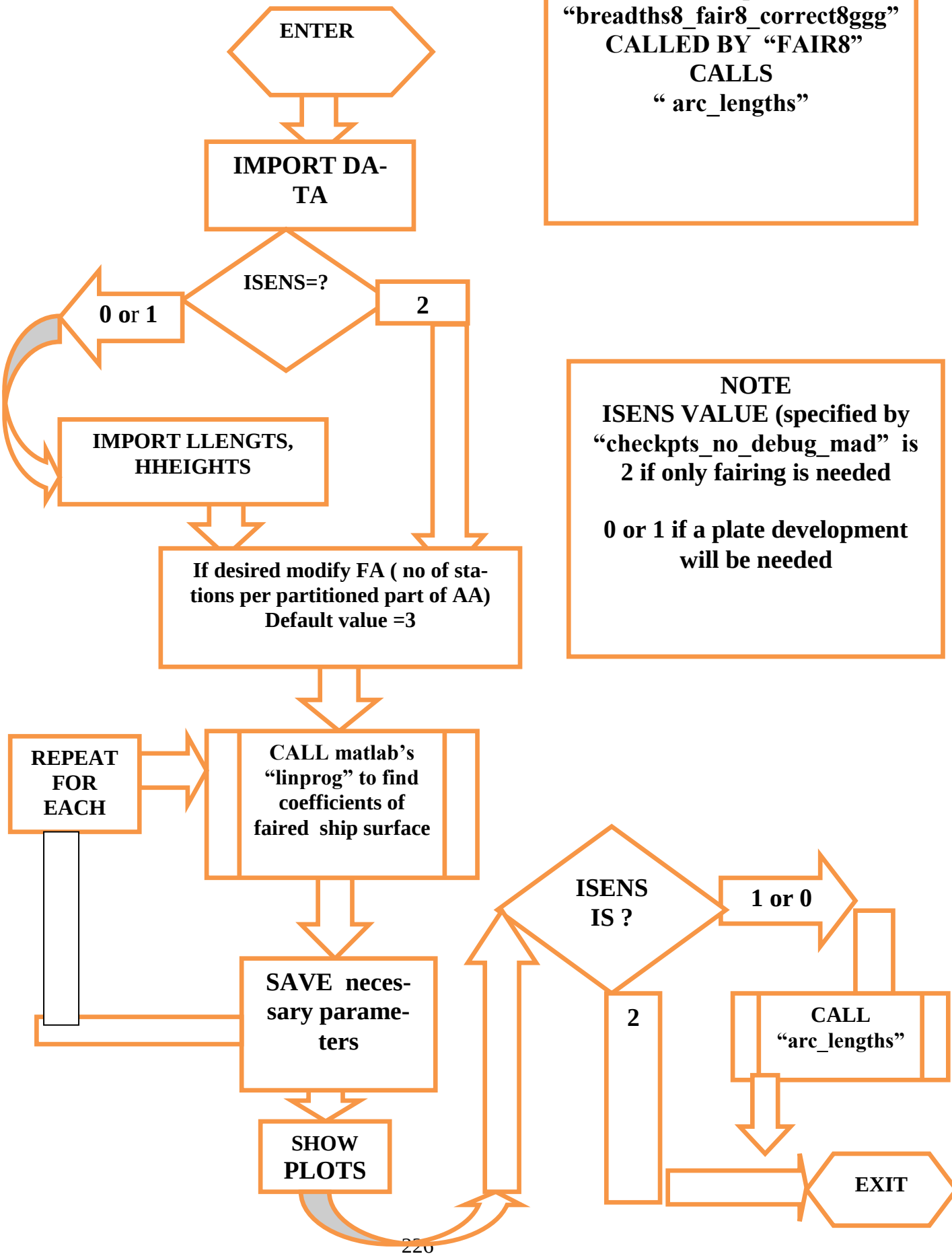
```
 end
% end ATEMPT TO DECREASE NO OF EQUATIONS (option)

AB=[]
beq=[]

[xx,fval,exitflag,output,lambda]=linprog(IFF,AA,b,AB,beq,[lb],[],[],optimset('Displa
y','iter','maxiter',3000,'LargeScale','on','TolX',   1E-05,'Tolfun',   1E-04))

'Press ENTER and wait until fairing by LARGE SCALE ALGORITHM is finished (
max 3000 cycles'
XX_ALL=xx;
SI_XX_ALL=size(XX_ALL)
save  c:\naval\fair.bak8\TEMP\XX_ALL.dat XX_ALL -ASCII -double
save  c:\naval\fair.bak8\TEMP\SI_XX_ALL.dat SI_XX_ALL -ASCII -double
save  c:\naval\fair.bak8\\XX_ALL.dat XX_ALL -ASCII -double
save  c:\naval\fair.bak8\SI_XX_ALL.dat SI_XX_ALL -ASCII -double

mytest1=xx
xx1=xx(2:end)
save  c:\naval\fair.bak8\TEMP\mytest.dat xx1 -ASCII -double
save  c:\naval\fair.bak8\TEMP\mytest1.dat xx1 -ASCII -double
save  c:\naval\fair.bak8\TEMP\exitflag.dat exitflag -ASCII -double
save  c:\naval\fair.bak8\TEMP\fval.dat fval -ASCII -double

load c:\naval\fair.bak8\x.dat
load c:\naval\fair.bak8\z.dat
offsets=[]

off=AA(1:end,2:end)*mytest1(2:end)
%offsets=off(1:n*m)
offsets=off(1:N*M)

%r_beg=2*m*n+1 %INFORMATION TO CHECK R_DERRIVATIVES
r_beg=2*m*N+1 %INFORMATION TO CHECK R_DERRIVATIVES

%r_end=r_beg-1+(n-2)*m
r_end=r_beg-1+(N-2)*m

r_derr=off(r_beg:r_end) % R_DERRIVATIVES AS COMPUTED

abs_r=abs(r_derr)
err_intex=abs_r
id=find(err_intex<.00000001)
err_intex(id)=0

s_beg=r_end+1 %INFORMATION TO CHECK S_DERRIVATIVES
%s_end=r_end+(m-2)*n
s_end=r_end+(m-2)*N
s_derr=off(s_beg:s_end) % S_DERRIVATIVES AS COMPUTED
```

```
abs_s=abs(s_derr)
err_intex_s=abs_s
id_s=find(err_intex_s<.00000001)
err_intex_s(id_s)=0

r_give=RS_R(1:end,2:end)
r_given=r_give(:)
TEST_R=[r_derr r_given  r_derr.* r_given err_intex ]

%find MAX_DERRIVED_R_CURVATURE AT POINTS WHERE TEST_R AP-
PEARS NEGATIVE
id=find(TEST_R(1:end,3)<0)
MAX_DERRIVED_R_CURVATURE=max(abs(r_derr(id)))

s_give=RS_S(1:end,2:end)
s_give=s_give'
s_given=s_give(:)
TEST_S=[s_derr s_given  s_derr.* s_given err_intex_s ]

%find MAX_DERRIVED_S_CURVATURE AT POINTS WHERE TEST_S AP-
PEARS NEGATIVE
id=find(TEST_S(1:end,3)<0)
MAX_DERRIVED_S_CURVATURE=max(abs(s_derr(id)))

'Press ENTER to see FAIRED BODY PLAN and to save offsets  in
c:\naval\fair.bak8\myplot.dat'
%pause
offsets1=[]
for I=1:n;
  for J=1:m;
  offsets1(I,J)=offsets( (I-1)*m+J)
end
end
hold on

if (ISENS==2)
   HHEIGHTS=FZCAL
   LLENGTHS=FXCAL
else
end
plot(offsets1(1:end,1:end)',HHEIGHTS)
hold on
plot (new_fy,HHEIGHTS,'*')
plot (FY_GIVEN,FZ,'b^')
'SAVE plot in c:\naval\fair8\TEMP\body_plan.fig AND PRESS RETURN TO CON-
TINUE(MANUALLY!!)'
pause
hold off
'Press ENTER to see WATER LINES  PLAN and to save offsets  in
c:\naval\fair.bak8\myplot1.dat'
```

```
offsets2=offsets1'
plot(offsets2',LLENGTHS)
hold on
axis equal
plot(new_fy',LLENGTHS,'*')
plot(FY_GIVEN',FX,'b^')
'SAVE plot in c:\naval\fair8\TEMP\Waterlines_plan.fig AND PRESS RETURN TO
CONTINUE (MANUALLY)'
pause
save  c:\naval\fair.bak8\TEMP\myplot1.dat offsets2 -ASCII -double
save  c:\naval\fair.bak8\TEMP\myplot.dat offsets1 -ASCII -double
save  c:\naval\fair.bak8\offsets2.dat offsets2 -ASCII -double
save  c:\naval\fair.bak8\offsets1.dat offsets1 -ASCII -double
save  c:\naval\fair.bak8\TEMP\r_derr1.dat r_derr -ASCII -double
save  c:\naval\fair.bak8\TEMP\s_derr1.dat s_derr -ASCII -double

if(ISENS~=2)
   arc_lengths % CALL PROGRAM "arc_lengths"
else
end

hold off
if((nio==3)&(ISENS~=2))

   cla
   test_test_test8_mad1
else
end
```

ANNEX(8) + PLATE(8)
DOCUMENTATION OF MATLAB PROCESS
“breadths8_fair8_correct8ggg.m”

PLATE 8
matatlab process
"breadths8_fair8_correct8ggg"
CALLED BY "FAIR8"
CALLS
" arc_lengths"



NOTE
ISENS VALUE (specified by "checkpts_no_debug_mad" is 2 if only fairing is needed

0 or 1 if a plate development will be needed

MATLAB MODULE "breadths8_fair8_correct8ggg"

```
----------------------------------------
format long

load c:\naval\fair.bak8\ndf.dat
load c:\naval\fair.bak8\IFF.dat
load c:\naval\fair.bak8\A.dat
%load c:\naval\fair.bak8\b.dat
load c:\naval\fair.bak8\AB.dat
load c:\naval\fair.bak8\beq.dat

beq=[]

load c:\naval\fair.bak8\nm.dat
%load c:\naval\fair.bak8\DD.dat
%load c:\naval\fair.bak8\DD.dat
load c:\naval\fair.bak8\zzgiven400.dat
load c:\naval\fair.bak8\GENDAT1.dat
load c:\naval\fair.bak8\CONSTANTS1.dat
load c:\naval\fair.bak8\FZCAL.dat
load c:\naval\fair.bak8\FXCAL.dat
load c:\naval\fair.bak8\FY_GIVEN.dat
load c:\naval\fair.bak8\FX.dat
load c:\naval\fair.bak8\FZO.dat
%load c:\naval\fair.bak8\new_fy.dat
load c:\naval\fair.bak8\ISENS.dat
load c:\naval\fair.bak8\NO_EQUAT.dat
    if (ISENS<2)

load c:\naval\fair.bak8\HHEIGHTS.dat
load c:\naval\fair.bak8\LLENGTHS.dat


else
end

n=CONSTANTS1(26);
m=CONSTANTS1(27);
AA_KEEP=AA;
lb=zeros(528,1);

AA=AA_KEEP;

%MODIFY AA TO AA1 IF NECESSARY (ONLY FOR TESTING)
   AA1=[];
   AA1=[AA(1:end,1:end)];
   b1=[b(1:end,1:end)];

   %PRELIMINARY STEPS FOR PARTITONING ARRAY "AA"
   FA=4
```

```
   G5=N/FA
   G5_ALL=fix(G5) % NO OF GROUPS OF 3
   G5_FIN=rem(N,FA) % ITEMS IN LAST GROUP
   for I=1:G5_ALL
   G55(I,1:2)=[I,FA]
end
G551=G55(1:end,1);
G552=G55(1:end,2);
if (G5_FIN~=0)
G55(G5_ALL+1,1:2)=[G5_ALL+1 G5_FIN];
else
end
SI_G=size(G55);
SI_G=SI_G(1,1);
'pause 55'
%pause;
%end PRELIMINARY STEPS FOR PARTITONING ARRAY "AA"

%START PARTITIONING
A_NEW1=[]; A_NEW2=[]; A_NEW3=[]; A_NEW4=[]; %SET ARRAYS
A_NEW1(1:G55(1,2)*M,1:ndf,1:SI_G)=0;
A_NEW2(1:G55(1,2)*M,1:ndf,1:SI_G)=0;
A_NEW3(1:(G55(1,2)-2)*M,1:ndf,1:SI_G)=0;
A_NEW4(1:G55(1,2)*(M-2),1:ndf,1:SI_G)=0; %end SET ARRAYS


IC=0; IC1=0; A_NEW1=[]; A_NEW2=[]; OFF1=OFFSETS(1:N*M,1:end);
OFF2=OFFSETS(N*M+1:2*N*M,1:end);
OFF3=UPPER_DER ; OFF4=LOWER_DER; SI_OFF3=size(OFF3);
SI_OFF3=SI_OFF3(1,1);
SI_OFF4=size(OFF4); SI_OFF4=SI_OFF4(1,1); OFF1_BEG=0; OFF1_END=0;
A_NEW3=[]; A_NEW4=[];OFF1_BEG_DER=0; OFF1_END_DER=0;
A_NEW1=[]; A_NEW2=[]; b_new1=[]; b_new2=[]; A_NEW11=[];b_new3=[];
b_new4=[]
offsets1_all=[]; A_KP=[]; A_KEEP=[]; B_KP=[]; B_KEEP=[]; b_new1_fin=[];
b_new2_fin=[]; b_new3_fin=[]; b_new4_fin=[];OFF3_BEG_DER=0;
OFF3_END_DER=0;
OFF4_BEG_DER=0; OFF4_END_DER=0; L=0;
A_NEW1_END=[]; A_NEW2_END=[]; A_NEW3_END=[]; A_NEW4_END=[];
XX_ALL=[];
A_NEW1=[], A_NEW2=[]; A_NEW3=[]; A_NEW4=[];
A_NEW1_END=[]; A_NEW2_END=[]; A_NEW3_END=[]; A_NEW4_END=[];
XX_ALL=[];

OFF1_BEG=0; OFF1_END=0
  offsets=[]; offsets1=[];
 IC2=0; IC1=0;
for L=1:SI_G  %  HANDLE OFFSETS
%for L=1:5
```

```
   G=0
   AEQ=[]; ANE=[]; SI_KP_L=[];
       IC1=0;IC=IC+1;
      OFF1_BEG= 1+ OFF1_END; OFF1_END=OFF1_END+G55(L,2)*M;
      A_NEW1(1:OFF1_END+1-
OFF1_BEG,:,L)=OFF1(OFF1_BEG:OFF1_END,1:ndf);
      A_NEW1_END(:,:,L)=A_NEW1(end-1*M+1:end,1:end,L);
      spy(A_NEW1(:,:,L))

      b_new1=b(OFF1_BEG:OFF1_END);
      b_new1_end=b_new1(end-1*M+1:end);
      b_new1=(b_new1)';
      b_new1_en=b_new1_end';
      b_new1_fin=[b_new1_fin
         b_new1_en];
      'pause 102'
      %pause
     [OFF1_BEG OFF1_END];
'pause down 104'
%pause
% END CALCULATE A_NEW1 and A_NEW1_END



 A_NEW2(1:OFF1_END+1-
OFF1_BEG,:,L)=OFF2(OFF1_BEG:OFF1_END,1:ndf);
 A_NEW2_END(:,:,L)=A_NEW2(end-1*M+1:end,1:end,L);
      b_new2=b(OFF1_BEG+N*M:OFF1_END+N*M);
      b_new2_end=b_new2(end-1*M+1:end);
      b_new2=(b_new2)';
      b_new2_en=b_new2_end';
      b_new2_fin=[b_new2_fin
         b_new2_en];
      spy([A_NEW1
         A_NEW2])
    % END CALCULATE A_NEW2 and A_NEW2_END


PL=0
  % HANDLE  UPPER (A_NEW3)  AND LOWER  (A_NEW4) DERRIVATIVES

  MI=0

  if (G55(L,2)==2); MI1=1; else; MI1=0; end;
  if (G55(L,2)==1); MI2=2; else; MI2=0; end;
  MI=MI1+MI2
  'pause 125'
  %pause
```

```
  if (L>1)&(L~=SI_G); PL=1; else; PL=0; end;

  OFF3_BEG_DER=                1+                OFF3_END_DER;
OFF3_END_DER=OFF3_END_DER+(G55(L,2)-1+PL-MI )*M;

  A_NEW3(1:OFF3_END_DER+1-
OFF3_BEG_DER,1:ndf,L)=OFF3(OFF3_BEG_DER:OFF3_END_DER,1:ndf);
  [L      MI      PL      OFF3_END_DER+1-OFF3_BEG_DER      OFF3_BEG_DER
OFF3_END_DER]
  'pause 131'
   spy(A_NEW3(1:OFF3_END_DER+1-OFF3_BEG_DER,:,L))
  %pause
  if (L>1)&(L~=SI_G);% NO INFORM FOR PREVIOUS NECESSARY FOR
FIRTS & LAST GROUP
  A_NEW3_END=A_NEW3(end-M+1:end,1:ndf,L-1);


end
[OFF3_BEG_DER OFF3_END_DER]
'pause 130'
%pause;


  SI3=size(A_NEW3_END)
 %pause
SI_B3=size(A_NEW3_END); SI_B3=SI_B3(1,1);
     b_new3_fi(1,1:SI_B3)=0;
     b_new3_fin=[b_new3_fin
       b_new3_fi];



    OFF4_BEG_DER=                1+                OFF4_END_DER;
OFF4_END_DER=OFF4_END_DER+G55(L,2)*(M-2);
    A_NEW4(1:OFF4_END_DER+1-
OFF4_BEG_DER,1:ndf,L)=OFF4(OFF4_BEG_DER:OFF4_END_DER,1:ndf);
    if (L>1)&(L~=SI_G);
   A_NEW4_END=A_NEW4(end-(M-2)+1:end,1:ndf,L-1);
end
    SI4=size(A_NEW4_END)


    SI_B4=size(A_NEW4_END); SI_B4=SI_B4(1,1);

     b_new4_fi(1,1:SI_B4)=0;
     b_new4_fin=[b_new4_fin
       b_new4_fi];
      [L]

     [OFF4_BEG_DER OFF4_END_DER]
```

```
       'pause 155'
       %pause;

  [size(A_NEW1) size(A_NEW2) size(A_NEW3) size(A_NEW4) ]
  spy([A_NEW1(:,:,L)
      A_NEW2(:,:,L)
      A_NEW3(:,:,L)
      A_NEW4(:,:,L)])
     [PL    MI    size(A_NEW1_END)    size(A_NEW2_END)    size(A_NEW3_END)
size(A_NEW4_END) ]
     'pause 169'
    %pause


   %if L~=SI_G;
    %spy([A_NEW1_END(:,:,L)
     %A_NEW2_END(:,:,L)
     %A_NEW3_END(:,:,L)
     %A_NEW4_END(:,:,L)])
     %pause
     %else
     %end
  'pause 176'
  %pause
  SI_B4=size(A_NEW4_END); SI_B4=SI_B4(1,1);


     SI_3 =size(A_NEW3); SI_3=SI_3(1,1);
     SI_4 =size(A_NEW4); SI_4=SI_4(1,1);
     si_b=[];

       si_b3=SI_3
     si_b4=SI_4
       b_new3=[]; b_new4=[];
       b_new3(1:si_b3)=0;
       b_new4(1:si_b4)=0;


      b_all=[]
      b_all=[b_new1'
         b_new2'
         b_new3'
         b_new4'];

      b_all_end=[(b_new1_fin(L,:))'
      (b_new2_fin(L,:))'
      %(b_new3_fin(L,:))'
      %(b_new4_fin(L,:))'];
      (b_new3_fin(1,1:end))'
      (b_new4_fin(1,1:end))'];
```

```
      [size(b_all,1) L]
      SI_B=[]
      SI_B(1:size(b_all,1),1)=L;


  [OFF1_BEG_DER OFF1_END_DER];
   %pause
  % end HANDLE  UPPER (A_NEW3)  AND LOWER  (A_NEW4) DER-
RIVATIVES
  AEQ=[]; beq=[]; ANE=[]; bne=[];
  A_ALL1=[ A_NEW1(:,:,L)

         A_NEW2(:,:,L)

         A_NEW3(:,:,L)

         A_NEW4(:,:,L)];

  SI=size(A_ALL1,1)
  [L]
  'pause 254'
  %pause


   if L==1; AEQ=[]; beq=[];
      ANE=A_ALL1(1:SI,1:end)
      bne=b_all(1:SI)
 'pause `261 CHANNEL 1'
 %pause

  size(AEQ)
  [L size(AEQ)]
  'pause 266'
  %pause
  size(ANE)
   SI_KP_L(L)=FA*M;
   [L size(ANE) size(bne)]
'pause 271 END OF FIRST GROUP '
%pause

else; end;  % END FIX  1ST STATION OF FIRST GROUP

       %if ((L~=1)&(L~=SI_G));  AEQ=[];
       if (((L~=1)&(L~=SI_G))| (L>1)& (G55(end,2))==(G55(end-1,2)) );  AEQ=[];
          'PAUSE 246 CHANNEL2'
          IC2=IC2+1
          %pause
          spy(A_NEW1_END(:,:,L-1))
          %pause
         SI_KP=size ([   A_NEW1_END(:,:,L-1)
```

```
          A_NEW1(:,:,L)]);
       SI_KP_L(L)=SI_KP(1,1);

         A_ALL=[   A_NEW1_END(:,:,L-1)
          A_NEW1(:,:,L)
           A_NEW2_END(:,:,L-1)
         A_NEW2(:,:,L)
          %A_NEW3_END(:,:,L-1)
          A_NEW3_END
         A_NEW3(:,:,L)
        %A_NEW4_END(:,:,L-1)
        A_NEW4_END
         A_NEW4(:,:,L)];


    b_all=[ (b_new1_fin(L-1,:))'
        b_new1'
          (b_new2_fin(L-1,:))'
          b_new2'
         (b_new3_fin(L-1,:))'
           b_new3'
          (b_new4_fin(L-1,:))'
           b_new4'];


       ANE=A_ALL;
       bne=b_all;
       [L size(ANE) size(bne)]
'pause 310 END OF INDERMDIATE GROUPS '
%pause
else

end

if ((L==SI_G))&(G55(end,2))~=(G55(end-1,2)) ;  AEQ=[];
           'PAUSE 292 CHANNEL3'
           %pause
           IC1=IC1+1
          MI=1
        A_ALL=[   A_NEW1_END(:,:,L-1)
              A_NEW1(1:end-(FA-G55(end,2))*MI*M,:,L)

           A_NEW2_END(:,:,L-1)
         A_NEW2(1:end-(FA-G55(end,2))*M*MI,:,L)


         %A_NEW3_END
         A_NEW3(1:end-(FA-G55(end,2))*MI*M,:,L)

        %A_NEW4_END(:,:,L-1)
```

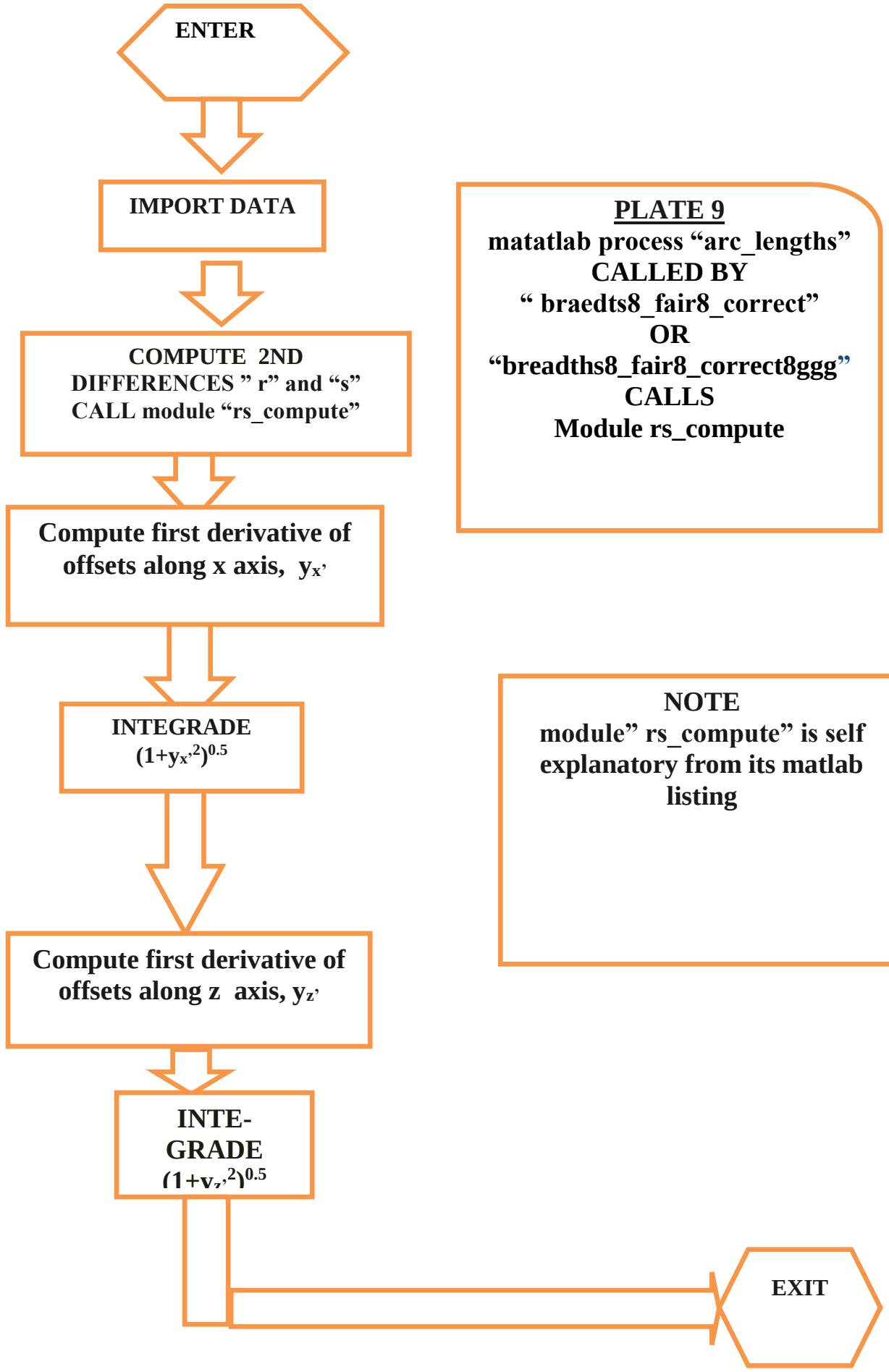
```
        A_NEW4_END
         A_NEW4(1:end-(FA-G55(end,2))*(MI)*(M-2),:,L)];

      si_b4=size(A_NEW4(1:end-(MI)*(M-2),:,L)); si_b4=si_b4(1,1)
      b_new4=[]; b_new4(1,1:si_b4)=0
        si_b=size(A_NEW3(1:end-MI*M,:,L));                    si_b=si_b(1,1);
b_ne3=b_new3(1,1:si_b);
       b_new3=b_ne3;

       F=[]; F1=[]; b_new3=[];b_new4=[]

       %F(1:G55(end,2)*(N-2))=0; b_new3=F'
       F(1:G55(end,2)*M)=0; b_new3=F'
       F1(1:(G55(end,2)+1)*(M-2))=0; b_new4=F1'
    b_all=[ (b_new1_fin(L-1,:))'
       b_new1'
         (b_new2_fin(L-1,:))'
         b_new2'
        %(b_new3_fin(L-1,:))'
        %b_new3'
        b_new3
         %(b_new4_fin(L-1,:))'
          %b_new4'];
          b_new4]
      ANE=A_ALL;
      bne=b_all;
   else
   end
   %lb(1:591)=0
   SI_IFF=size(IFF); SI_IFF=SI_IFF(1,2); lb(1:SI_IFF)=0;
   [L size(IFF) size(ANE) size(bne) size(AEQ) size(beq) size(lb)]
   'pause 339'
   'PRESS ENDER AND WAIT'
   size(ANE)
   size(bne)
   [L G55(1,2)]
  'pause 356'
  %pause
  SI_AN=size(ANE); SI_AN=SI_AN(1,1);SI_AE=size(AEQ); SI_AE=SI_AE(1,1);
if SI_AN+SI_AE>500 %NO OF EUATIONS MORE THAN 500. STOP PROCESS
 'SPLIT PARAMETER (FA) RESULTS TO NO OF EQUATIONS >500'
 'SELECT A SMALLER VALUE OF FA AND REPEAT'
 'NO EQUATIONS '
 [SI_AN+SI_AE]

 'FA VALUE'
 [FA]
 'PRESS RETURN TO GO TO END'
 %pause
 return
```

```
else
end
 'NO EQUATIONS '
 [SI_AN+SI_AE]
 'pause 384'
   xx=[];fval=[];exitflag=[];output=[];lambda=[]; result=[];

[xx,fval,exitflag,output,lambda]=linprog(IFF,ANE,bne,AEQ,beq',[lb],[],[],optimset('
Display','iter','maxiter',300,'LargeScale','on','TolX',   1E-03,'Tolfun',   1E-03))
[size(xx) size(ANE) size(AEQ)]
%                    ATTENTION'"Do                    not                    change
'TolX[xx,fval,exitflag,output,lambda]=linprog(IFF,ANE,bne,AEQ,beq',[lb],[],[],optim
set('Display','iter','maxiter',300,'LargeScale','on','TolX',   1E-02,'Tolfun',   1E-02))
[size(xx) size(ANE) size(AEQ)]'
  'PAUSE EXIT FAIRING 296'
  'L is'
  L
%  pause
  si_xx=size(xx); si_xx=si_xx(1,1); s_xx(L)=si_xx

  XX_ALL(1:si_xx,L)=xx;
  EXITFLAG_ALL(L)=exitflag
  FVAL_ALL(L)=fval

  if L==1
      A_TEM=[AEQ
          ANE];
  A_FIN=A_TEM;
  size(A_FIN)
'pause 333'
%pause
else
A_FIN=ANE;
size(A_FIN)
'pause 338'
%pause
end
 A_END=[AEQ
     ANE];
 if (L==1); OFF=[]; OFF=A_END(1:FA*M,2:end)*XX_ALL(2:end,L);
else; end

   if                    (L>1&L~=SI_G);                                        OFF=[];
OFF=A_END(M+1:(FA+1)*M,2:end)*XX_ALL(2:end,L);
else; end;

  if ((L==SI_G)& ( G5_FIN~=0))
  OFF=[]; OFF=A_END(M+1:G55(L-1,2)*M,2:end  )*XX_ALL(2:end,L);
else; end
```

```
if ((L==SI_G)& ( G5_FIN==0))
   OFF=[]; OFF=A_END(M+1:(FA+1)*M,2:end)*XX_ALL(2:end,L);
  %OFF=[]; OFF=A_END(1:FA*M,2:end  )*XX_ALL(2:end,L);
  'pause 346'
  %pause
else;end

offsets=[offsets
   OFF]
'pause 350'
%pause
end % BASIC LOOP

SI_OF=size(offsets)
SI_XX_ALL=size(XX_ALL)
save  c:\naval\fair.bak8\XX_ALL.dat XX_ALL -ASCII -double
save  c:\naval\fair.bak8\EXITFLAG_ALL.dat EXITFLAG_ALL -ASCII -double
save  c:\naval\fair.bak8\FVAL1_ALL.dat FVAL_ALL -ASCII -double
save  c:\naval\fair.bak8\SI_OF.dat SI_OF -ASCII -double
save  c:\naval\fair.bak8\SI_XX_ALL.dat SI_XX_ALL -ASCII -double

save  c:\naval\fair.bak8\TEMP\XX_ALL.dat XX_ALL -ASCII -double
save    c:\naval\fair.bak8\TEMP\EXITFLAG_ALL.dat  EXITFLAG_ALL  -ASCII  -
double
save  c:\naval\fair.bak8\TEMP\FVAL1_ALL.dat FVAL_ALL -ASCII -double
save  c:\naval\fair.bak8\TEMP\SI_OF.dat SI_OF -ASCII -double
save  c:\naval\fair.bak8\TEMP\SI_XX_ALL.dat SI_XX_ALL -ASCII -double

  if ISENS==2; HHEIGHTS=FZCAL; LLENGTHS=FXCAL;
  else;

  end;
  for II=1:N
for JJ=1:M
    offsets1(JJ,II)=offsets((II-1)*M+JJ);
    %[II,JJ (II-1)*M+JJ offsets((II-1)*M+JJ)]
       %pause
end
end
hold off
save c:\MATLABR12\BIN\WIN32\offsets1.dat offsets1 -ASCII -double
plot(offsets1,HHEIGHTS')
hold on
plot(offsets1,HHEIGHTS','^b')
plot(offsets1,HHEIGHTS','*')
plot(new_fy,HHEIGHTS','^')
'SAVE plot in c:\naval\fair8\TEMP\body_plan.fig AND PRESS RETURN TO CON-
TINUE(MANUALLY!!)'
pause
hold off
```

```
'Press ENTER to see WATER LINES  PLAN and to save offsets  in
c:\naval\fair.bak8\myplot1.dat'
hold off
offsets2=[]
offsets2=offsets1';
plot(LLENGTHS,offsets2)
hold on
plot(LLENGTHS,new_fy','*')
%save offsets1
%save offsets2
save c:\MATLABR12\BIN\WIN32\offsets2.dat offsets2 -ASCII -double
if ISENS~=2
arc_lengths % CALL "arc_lengths.m"
else
end
%xloc1=xloc1'; save  c:\naval\fair.bak8\xloc1.dat xloc1 -ASCII -double
%zloc1=zloc1'; save  c:\naval\fair.bak8\zloc1.dat zloc1 -ASCII -double
%save  c:\naval\fair.bak8\FUN_X1.dat FUN_X1 -ASCII -double
%save  c:\naval\fair.bak8\FUN_Z1.dat FUN_Z1 -ASCII -double

off_1=offsets1'
off_2=offsets2'
save  c:\naval\fair.bak8\offsets1.dat off_1 -ASCII -double
save  c:\naval\fair.bak8\offsets2.dat off_2 -ASCII -double
save  c:\naval\fair.bak8\TEMP\offsets1.dat off_1 -ASCII -double
save  c:\naval\fair.bak8\TEMP\offsets2.dat off_2 -ASCII -double

hold off
if((nio==3)&(ISENS~=2))
   %models18_7
   %pause

   hold off
   %test_test_test8_mad1 % CALL"test_test_test8_mad1"
   arc_lengths % CALL "arc_lengths"
else
end
```

ANNEX(9) + PLATE(9)
DOCUMENTATION OF MATLAB PROCESS
“arc_lengths.m”


ENTER
IMPORT DATA
COMPUTE 2ND DIFFERENCES " r" and "s" CALL module "rs_compute"
Compute first derivative of offsets along x axis, $y_{x'}$
INTEGRADE $(1+y_{x'}^2)^{0.5}$
Compute first derivative of offsets along z axis, $y_{z'}$
INTE-GRADE $(1+y_{z'}^2)^{0.5}$
EXIT
PLATE 9
matatlab process "arc_lengths"
CALLED BY
" braedts8_fair8_correct"
OR
"breadths8_fair8_correct8ggg"
CALLS
Module rs_compute
NOTE
module" rs_compute" is self explanatory from its matlab listing

MATLAB MODULE "arc_lengths"

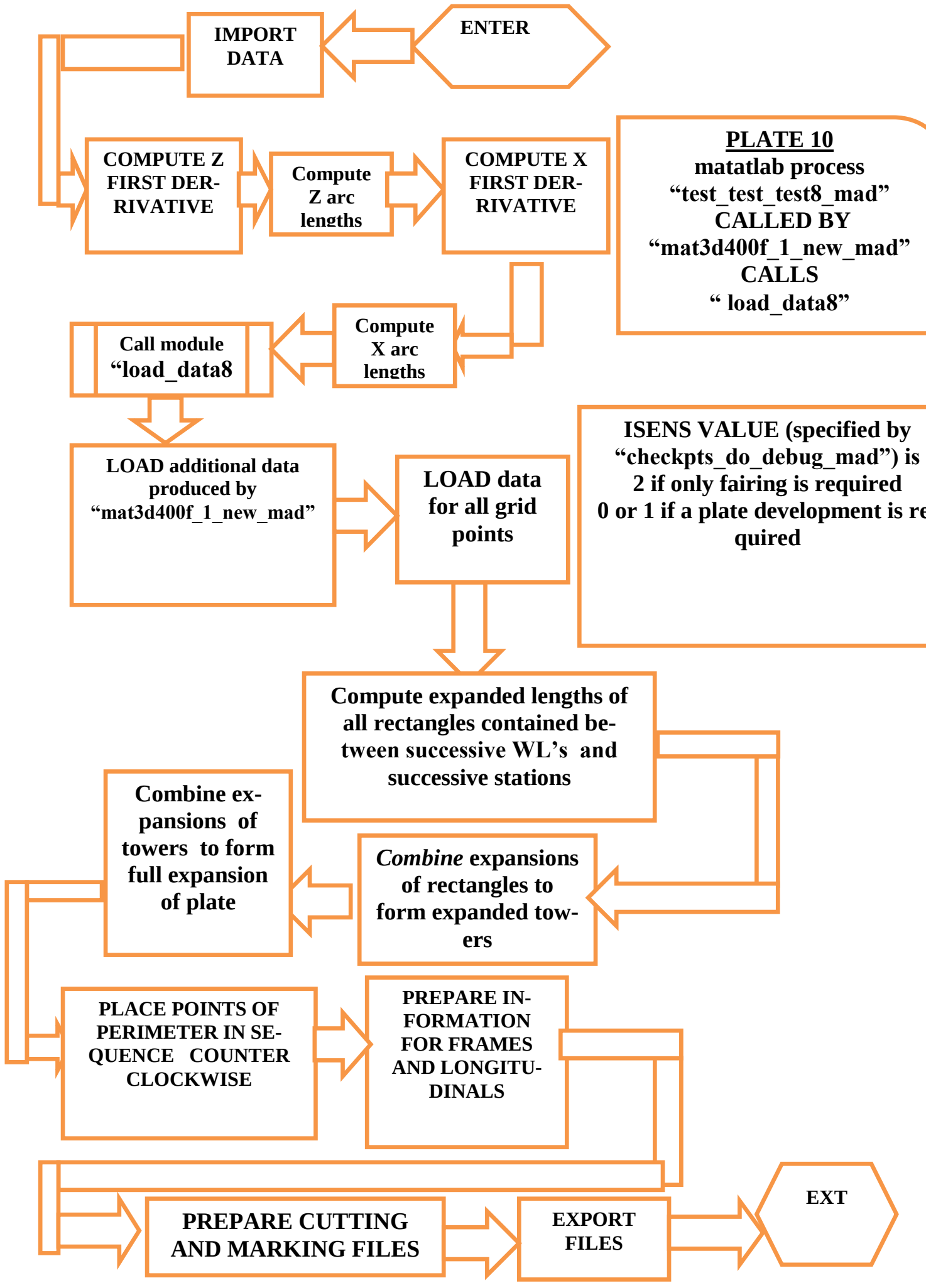
```
load c:\naval\fair.bak8\ndf.dat
load c:\naval\fair.bak8\IFF.dat
load c:\naval\fair.bak8\b.dat
load c:\naval\fair.bak8\AB.dat
load c:\naval\fair.bak8\beq.dat
load c:\naval\fair.bak8\new_fy.dat
load c:\naval\fair.bak8\I_PRO.dat
load c:\naval\fair.bak8\constants1.dat
load c:\naval\fair.bak8\z.dat
load c:\naval\fair.bak8\x.dat

co=constants1
nm=co(1); nm2=co(2); nstav=co(3); nb=co(4); nc=co(5); nd=co(6); neq=co(7);
np=co(8); ns=co(9);
nx=co(10);nq=co(11); nv=co(12); nstav=co(13); nbf=co(14); ncf=co(15); ndf=co(16);
npf=co(17); nsf=co(18);
nxf=co(19); nqf=co(20); nrf=co(21); nm2=co(22); nqfe=co(23); nrfe=co(24);
NATO=co(25);
N=(nb/2)+2;
M=(nc/2)+2;
RS_COMPUTE % COMPUTE parameters r and s
RS_S=s
RS_R=r

rr1=[];
rr1=r(1:(N-2),2:end); rrr1=rr1(:);

r_all=[];
SI_D_X_CO=constants1(6);
for I1=1:SI_D_X_CO
  r_all= [r_all rrr1];
end
% end PREPARE 'r_all'

% PREPARE 's_all'
ss1=[]
ss1=s(1:end,2:end); sss1=ss1(:);
s_all=[];
SI_D_Z_CO=constants1(6);
for I1=1:SI_D_Z_CO
  s_all= [s_all sss1];end

% COMPUTE offsets1 FIRST DERRIVATIVES RELATIVE TO X by 'polyfit'
DERR_X=[]; DERR_X1=[];DERR_Z=[]; DERR_Z1=[];
FUN_X=[]; FUN_Z=[]; FUN_X1=[]; FUN_Z1=[];
syms X DERR_X FUN_X

if I_PRO==1
```

```
SI_OF2=size(offsets1);  SI_OF2_LI=SI_OF2(1,1);  SI_OF2_CO=SI_OF2(1,2)
else
 SI_OF2=size(offsets2);  SI_OF2_LI=SI_OF2(1,1);  SI_OF2_CO=SI_OF2(1,2)
end

p=[];
for Q=1: SI_OF2_CO %
for V=1:SI_OF2_LI %for every STATION

 if I_PRO==1
y=offsets2(Q,1:SI_OF2_LI); % offsets on each wareline
else
  y=offsets1(Q,1:SI_OF2_LI);
end

p(Q,1:3,V)=polyfit(LLENGTHS',y,2);
DERR_X(Q,1)=2*p(Q,1,V)*X+p(Q,2,V); % Symbolic derrivative relative to X
FUN_X(Q,1)=(1+DERR_X(Q,1).^2).^0.5; % X SYMBOLIC INTEGRAD
end
end

FUN_X1=[];
for Q=1: SI_OF2_CO
for V=1:SI_OF2_LI
X=LLENGTHS(V);
  FUN_X1(Q,V)=subs( FUN_X(Q,1),X);  %X_INERGAD
  DERR_X1(Q,V)=subs( DERR_X(Q,1),X);
end
end
% end COMPUTE offsets1 FIRST DERRIVATIVS RELATIVE TO X by 'polyfit'

% INTEGRADE
xloc1=[];
LL=diff(LLENGTHS); SI_LL=size(LL); SI_LL=SI_LL(1,1)
for J=1:Q
for I=1:SI_LL; MEAN=(FUN_X1(J,I)+FUN_X1(J,I+1))/2; PROD=MEAN*LL(I);
 xloc1(J,I)=PROD;
end
end
xloc1(1:end,I+1)=xloc1(1:end,I);
% end INTEGRADE

 % COMPUTE offsets1 FIRST DERRIVATIVES RELATIVE TO Z by 'polyfit'
syms Z DERR_Z FUN_Z
if I_PRO==1
SI_OF2=size(offsets1);  SI_OF2_LI=SI_OF2(1,1);  SI_OF2_CO=SI_OF2(1,2)
else
 offsets1=offsets1' %TEMPORARY REVERSE
end
pz=[];
```

```
for Q=1: SI_OF2_LI % % for 25  WL's
for V=1:SI_OF2_LI %for every STATIONS
   yz=offsets1(Q,1:SI_OF2_CO); % offsets on each wareline 23

pz(Q,1:3,V)=polyfit(HHEIGHTS',yz,2);
DERR_Z(Q,1)=2*pz(Q,1,V)*Z+pz(Q,2,V); % Symbolic derrivative relative to X
FUN_Z(Q,1)=(1+DERR_Z(Q,1).^2).^0.5; % X SYMBOLIC INTEGRAD
end
end

FUN_Z1=[];

for Q=1: SI_OF2_LI
for V=1:SI_OF2_CO
Z=HHEIGHTS(V);
  FUN_Z1(Q,V)=subs( FUN_Z(Q,1),Z);  %X_INERGAD
  DERR_Z1(Q,V)=subs( DERR_Z(Q,1),Z);
end
end
% end COMPUTE offsets1 FIRST DIDDERENCES RELATIVE TO X by 'polyfit'

% INTEGRADE
zloc1=[];
HH=diff(HHEIGHTS); SI_HH=size(HH); SI_HH=SI_HH(1,1)
for J=1:Q
for I=1:SI_HH; MEAN=(FUN_Z1(J,I)+FUN_Z1(J,I+1))/2; PROD=MEAN*HH(I);
 zloc1(J,I)=PROD;
end
end
zloc1(1:end,I+1)=zloc1(1:end,I);
% end INTEGRADE

xloc1=xloc1'; save  c:\naval\fair.bak8\xloc1.dat xloc1 -ASCII -double
save  c:\naval\fair.bak8\zloc1.dat zloc1 -ASCII -double
save  c:\naval\fair.bak8\FUN_X1.dat FUN_X1 -ASCII -double
save  c:\naval\fair.bak8\FUN_Z1.dat FUN_Z1 -ASCII -double

DERR_X1=DERR_X1';
save  c:\naval\fair.bak8\DDXX.dat DERR_X1 -ASCII -double
DERR_Z1=DERR_Z1;
save  c:\naval\fair.bak8\DDZZ.dat DERR_Z1 -ASCII -double

if I_PRO==2 %RESTORE offsets1 if I_PRO=2
offsets1=offsets1'
else
end%end RESTORE offsets1 if I_PRO=2

%COMPUTE EXPANDED ANGLES
NUM=DERR_X1.*DERR_Z1
QX=1+DERR_X1.^2; QZ=1+DERR_Z1.^2;; QXZ=QX.*QZ; DEN=sqrt(QXZ)
```

```
delete RAM_ARC1.mat

RAM=NUM./DEN; RAM_ARC=acos(RAM)
RAM_ARC1=57.3*RAM_ARC
RAM_ARC1(1:end,end)=90
RAM_ARC1(end,1:end)=90
save  c:\naval\fair.bak8\dddxz.dat RAM_ARC1 -ASCII -double
%save RAM_ARC1
save c:\matlabr12\bin\win32\RAM_ARC1.dat RAM_ARC1 -ASCII -double
% RAM_ARC1 are exactly the same as dddxz of mat3d400f_1_new_mad
%end COMPUTE EXPANDED ANGLE
load c:\naval\fair.bak8\dddxz.dat
C=dddxz-RAM_ARC1
max(max(abs(C)))
```

ANNEX(10) + PLATE(10)
DOCUMENTATION OF MATLAB PROCESS
“test_test_test_8_mad1”


ENTER
IMPORT DATA
COMPUTE Z FIRST DER-RIVATIVE
Compute Z arc lengths
COMPUTE X FIRST DER-RIVATIVE
PLATE 10
matatlab process
“test_test_test8_mad”
CALLED BY
“mat3d400f_1_new_mad”
CALLS
“ load_data8”
Compute X arc lengths
Call module “load_data8
LOAD additional data produced by “mat3d400f_1_new_mad”
LOAD data for all grid points
ISENS VALUE (specified by “checkpts_do_debug_mad”) is
2 if only fairing is required
0 or 1 if a plate development is re
quired
Compute expanded lengths of all rectangles contained be-tween successive WL’s and successive stations
Combine ex-pansions of towers to form full expansion of plate
Combine expansions of rectangles to form expanded tow-ers
PLACE POINTS OF PERIMETER IN SE-QUENCE COUNTER CLOCKWISE
PREPARE IN-FORMATION FOR FRAMES AND LONGITU-DINALS
PREPARE CUTTING AND MARKING FILES
EXPORT FILES
EXT

```
% THIS PROGRAM PREPARES AND REARRANGES DATA  OF THE WHOLE GRID OF DEVELOPED AREA
% OF ANY PLATE USING AS INPUT DATA  PRODUCED IN FORTRAN BY 'mat3d400f_1_new_mad1"
% AND PLOTS POINTS OF THE DEVELOPED AREA DIAGRAM PROGRES-SIVELY SIMULATING ACTION
% OF AN AUTOMATIC CUTTING MACHINE
% IT IMPORTS DATA by calling "load_data8"
load c:\naval\fair.bak8\gendat8.dat
load c:\naval\fair.bak8\offsets1.dat
myplot=offsets1
OFF=gendat8(1)
save  c:\naval\fair.bak8\myplot.dat myplot -ASCII -double

load c:\naval\fair.bak8\LLENGTHS.dat
load c:\naval\fair.bak8\HHEIGHTS.dat
SI_LL=size(LLENGTHS)
SI_LL=SI_LL(1,1)
SI_HH=size(HHEIGHTS)
SI_HH=SI_HH(1,1)
SI_MO=size(myplot)
% COMPUTE  Z FIRST DIFFERENCES
ARC_Z=[]
for I=1:SI_LL-1
   for J=2:SI_HH
    DIF_MO=myplot(I,J)-myplot(I,J-1)
    DIF_HH=HHEIGHTS(J)-HHEIGHTS(J-1)
    AR_Z=(DIF_MO^2+DIF_HH^2)^.5
    ARC_Z(I,J-1)=AR_Z
    %pause
 end
end

Y_Z=[];
HH_Z=[];
Y1=[];
P=[];
RES_Z=[];
for I=1:SI_LL-1
   for J=1:SI_HH-1
      if((J+2)<=SI_HH)
       Y1=myplot(I,J:J+2)-myplot(I,1);
     Y_Z=Y1';
     HH_Z=HHEIGHTS(J:J+2)-HHEIGHTS(1);

     CO=polyfit(HH_Z,Y_Z,2);
  else
  end
[CO(1) CO(2) CO(3)];
```

```
CO1=CO(1); CO2=CO(2); CO3=CO(3); % COEFFICIENS OF SECOND DEGREE
FIT
syms Z
S=2*CO1*Z+CO2;  % X RERRIVATIVE
INTEGR=(1+S^2)^.5;
Z_BEG=subs(Z,HHEIGHTS(J)-HHEIGHTS(1));
Z_END=subs(Z,HHEIGHTS(J+1)-HHEIGHTS(1));
[Z_BEG Z_END];
INTEGRAL=int(INTEGR,Z_BEG,Z_END);
Z1=HHEIGHTS(J);
 RES_Z(I,J)=subs(INTEGRAL,Z,Z1);
end
end
% end COMPUTE  Z FIRST DIFFERENCES

% COMPUTE X FIRST DERRIVATIVE
ARC_X=[]
for I=1:SI_HH-1
   for J=2:SI_LL
    DIF_MO=myplot(J,I)-myplot(J-1,I)
    DIF_LL=LLENGTHS(J)-LLENGTHS(J-1)
    AR_X=(DIF_MO^2+DIF_LL^2)^.5
    ARC_X(I,J-1)=AR_X
 end
end
% end COMPUTE X FIRST DERRIVATIVE


load_data8 % EXTERNAL CALL of module "load_data8"


% LOAD ADDITIONAL DATA for xloc1 zloc1 Produced by " arc_lengths" called
% either by "breadths8_fair8_correct" or by "breadths8_fair8_correct8ggg"
dat1=[];
dat2=[];
dat3=[];
dat6=[];
count=0;


 dat1=[1:(si_LL-1)*(si_HH-1)];
        dat1=dat1';

        xxloc1=xloc1(1:end-1,1:end);
        xxloc2=xxloc1(1:end,1:end-1);
        dat3=xxloc2(:);

        zzloc1=zloc1(1:end-1,1:end);
        zzloc2=zzloc1(1:end,1:end-1);
        dat6=zzloc2(:);
```

```
        dat2=[];
        dat22=[];
        for i=1:si_LL-1;
        dat2=[1:si_HH-1];
        dat2=[dat2 dat22];
       dat22=dat2;
   end
   dat2=dat22';

 all_pts1=[dat1
    dat2
    dat3
    dat6];
    all_pts1=all_pts1';
 % end LOAD ADDITIONAL DATA for xloc1 zloc1

 % LOAD ADDITIONAL DATA for ef iko  produced by "mat3d400f_1_new_mad"
si_all=size(all_pts);
si_per=size(per_pts);
all_pts(1:end,6)=888888;
all_pts(1:end,7)=888888;
for i=1:si_all(1,1)
   for io=1:si_per(1,1)
      if (all_pts(i,1)==per_pts(io,1))&(all_pts(i,2)==per_pts(io,2))
      all_pts(i,6)=per_pts(io,5);
      all_pts(i,7)=per_pts(io,6);
   else
   end
   end
end
% end LOAD ADDITIONAL DATA for ef iko produced by
"mat3d400f_1_new_mad"

 all_pts(si_all(1,1)+1,1:si_all(1,2)+2)=all_pts(1,1:end); % FILL LAST LINE OF all
GRID POINTS
 all_pts(end,end)=per_pts(end,end);
 all_pts(end,6)=per_pts(1,1);
 all_pts(end,7)=per_pts(1,6); %  end FILL LAST LINE OF all GRID POINTS

 si_all=size(all_pts);
 si_per=size(per_pts);
 all_lins=si_all(1,1); % LINES OF all GRID POINTS
 all_cols=si_all(1,2); % COLUMNS OF all GRID POINTS
 per_lins=si_per(1,1); % LINES OF perimeter GRID POINTS
 per_cols=si_per(1,2); % COLUMNS OF perimeter GRID POINTS

 % ADD ADDITIONAL INFORMATION FOR all GRID POINTS
 hold on
```

```
 'BEGIN OF LOOP';
id=[];
 count=0;
 si_per=size(per_pts);
 si_per=si_per(1,1);
for i=1:si_per
 if ( per_pts(i,6)~=0 )
    if ( per_pts(i,6)~=999999 )
  count=count+1;
  id(count)=i;
else
 end
else
end
end
si_id=size(id);
si_id=si_id(1,2);

id1=[];
for j=1:si_id
   I_FOUNT(j)=per_pts(id(j),1);
   IF=I_FOUNT(j);
   J_FOUNT(j)=per_pts(id(j),2);
   zloc_fount(j)=per_pts(id(j),7);
    iko_fount(j)=per_pts(id(j),5);
    id1=find(                                    (all_pts(1:1:end,1)==I_FOUNT(j))&
(all_pts(1:1:end,2)==J_FOUNT(j))&(all_pts(1:1:end,6)==iko_fount(j)) );

    all_pts(id1,8)=zloc_fount(j);
end
 % end ADD ADDITIONAL INFORMATION FOR all GRID POINTS

% CALCULATE DATA FOR ALL RECTANGLES
si_pts=size(all_pts);
all_cols=si_pts(1,2); %NO OF TOTAL COLUMNS OF all_pts
all_lins=si_pts(1,1); %NO OF TOTAL LINES OF all_pts

si_i=si_LL; %NO OF RECTANGLES COUNTED HORIZONTALLY
si_j=si_HH; %NO OF RECTANGLES COUNTED VERTICALLY

load c:\naval\fair.bak8\dddxz.dat % PRODUCED by "mat3d400f_1_new_mad"
%%load RAM_ARC1
%load       c:\matlabr12\bin\win32\RAM_ARC1.dat       %       PRODUCED       by
"mat3d400f_1_new_mad"
%load
dddxz=RAM_ARC1
ddd=dddxz(1:si_i,1:si_j);
load c:\naval\fair.bak8\zloc1.dat
load c:\naval\fair.bak8\xloc1.dat
load c:\naval\fair.bak8\points.dat
```

```
xlo1=xloc1(1:si_i,1:si_j);
zlo1=zloc1(1:si_i,1:si_j);

% CALCULATE BOTTOM TRIANGLES ANGLES
D=[];
      SIN_KSI=[];
      SIN_PHI= [];
      ASIN_KSI=[];
      ASIN_PHI=[];
      COS_DDD_DOT=[];
FAC=2*pi/360;

xxlo1=xlo1(1:end-1,1:end);
zzlo1=zlo1(1:end-1,1:end);
ddd1=ddd(1:end-1,1:end);

  D=(xxlo1.^2+zzlo1.^2-2*xxlo1.*zzlo1.*cos(ddd1*FAC)).^.5;

      %FINDING THE TWO ANGLES OF BOTTOM TRIANGLES by use of cosine
law
      XI=xxlo1
      ZI=zzlo1
      DI=D
      ACOS_KI=(ZI.^2+DI.^2-XI.^2)./(2*ZI.*DI)
      ACOS_PI=(XI.^2+DI.^2-ZI.^2)./(2*XI.*DI)

      ASIN_KSI=acos(ACOS_KI)/FAC
      ASIN_PHI=acos(ACOS_PI)/FAC
      %end FINDING THE TWO ANGLES OF BOTTOM TRIANGLES by use of
cosine law

   %USEFUL FOR NEXT STEP
     xxlo1=xlo1(1:end,1:end);
zzlo1=zlo1(1:end,1:end);
ddd1=ddd(1:end,1:end);

 % CALCULATE TOP TRIANGLES ANGLES
  DO=(xxlo1.^2+zzlo1.^2-2*xxlo1.*zzlo1.*cos(ddd1*FAC)).^.5;
     COS_DDD_DOT=[];
     ACOS_DDD_DOT=[];
      SIN_DDD_DOT=[];
      SIN_PHI_DOT=[];
     SIN_KSI_DOT=[];
     ASIN_KSI_DOT=[];
     ASIN_PHI_DOT=[];

     %FINDING THE TWO ANGLES OF TOP TRIANGLES by use of cosine law
      XI_DOT=xxlo1
      ZI_DOT=zzlo1
```

```
      DI_DOT=DO
      ACOS_PI_DOT=(ZI_DOT.^2+DI_DOT.^2-
XI_DOT.^2)./(2*ZI_DOT.*DI_DOT)
      ACOS_KI_DOT=(XI_DOT.^2+DI_DOT.^2-
ZI_DOT.^2)./(2*XI_DOT.*DI_DOT)
      ACOS_KI_DOT=ACOS_KI_DOT(1:end-1,1:end-1)
      ACOS_PI_DOT=ACOS_PI_DOT(1:end-1,1:end-1)

      ASIN_KSI_DOT=acos(ACOS_KI_DOT)/FAC
      ASIN_PHI_DOT=acos(ACOS_PI_DOT)/FAC
      %end FINDING THE TWO ANGLES OF TOP TRIANGLES by use of cosine
law


      ddd2(1:si_i,1:si_j)=0

      %FINDING THE THIRD ANGLE OF TOP TRIANGLES by use of cosine law
       xxloc1=xloc1
      zzloc1=zloc1
      COS_DDD_DOT=(xxloc1.^2+zzloc1.^2-DO.^2)./(2*xxloc1.*zzloc1);
      COS_DDD_DOT=COS_DDD_DOT(1:end-1,1:end-1);
      ACOS_DDD_DOT=acos(COS_DDD_DOT)/FAC;

       ONES=ones(si_i-1,si_j-1);
     SIN_DDD_DOT=(ONES-COS_DDD_DOT.^2).^.5;


% CHECK SUMS
SUM=ACOS_DDD_DOT+ ASIN_KSI_DOT+ASIN_PHI_DOT;
% end CHECK SUMS

% end CALCULATE DATA FOR ALL RECTANGLES

% COMBINE  TO FIND ALL FOUR INTERNAL ANGLES OF TETRAEDRA
LE_BOT_COR=ddd(1:si_i-1,1:si_j-1);
LE_TOP_COR=ASIN_KSI(1:si_i-1,1:si_j-1)+ASIN_KSI_DOT;
RI_BOT_COR=ASIN_PHI(1:si_i-1,1:si_j-1)+ASIN_PHI_DOT;
RI_TOP_COR=ACOS_DDD_DOT;
SUM=LE_BOT_COR+LE_TOP_COR+RI_BOT_COR+RI_TOP_COR;
[LE_BOT_COR LE_TOP_COR RI_BOT_COR RI_TOP_COR SUM];
% end COMBINE  TO FIND ALL FOUR INTERNAL ANGLES OF TETRAEDRA

% FIND ASSOCIATED EXPANDED LENGTHS (all GRID POINTS)
BOTTOM=xlo1(1:si_i-1,1:si_j-1);
TOP=xlo1(1:si_i-1,2:si_j);
LEFT=zlo1(1:si_i-1,1:si_j-1);
RIGHT=zlo1(2:si_i,1:si_j-1);

[BOTTOM LEFT TOP RIGHT ];
% end FIND ASSOCIATED EXPANDED LENGTHS (all GRID POINTS)
```

```
% 1.-COMBINE POLYGONS VERICALLY
%1.1.-Select data for lengths and angles
SI=si_i-1;
SJ=si_j-1;
REALA=[];
IMAGA=[];
REALB=[];
IMAGB=[];
REALC=[];
IMAGC=[];
REALD=[];
IMAGD=[];
BOT=[];
TO=[];
LE=[];
RI=[];
RB=[];
RC=[];
RD=[];

RA_ST=0
   RC_ST=0;
   RD_ST=0;
   RB_ST=0;
for I=1:SI

   b=0
   a=0;
   for J=1:SJ


      BOT=BOTTOM(I,J);
      TO=TOP(I,J);
      LE=LEFT(I,J);
      RI=RIGHT(I,J);

      LE_BOT=LE_BOT_COR(I,J);
      LE_TOP=LE_TOP_COR(I,J);
      RI_BOT=RI_BOT_COR(I,J);
      RI_TOP=RI_TOP_COR(I,J);
      REALA(I,J)=0;
      IMAGA(I,J)=0;

     [BOT TO LE RI LE_BOT LE_TOP RI_BOT RI_TOP] ;
     [LE_BOT ]

     a=LE_BOT-90-b;

      REALB(I,J)=-LE*sin(a*FAC);
```

```
       IMAGB(I,J)=LE*cos(a*FAC);
      ' I J';
      [I J a  b LE REALB(I,J) IMAGB(I,J) ];


       [LE LE_BOT];

       b=90-LE_TOP-a;

       REALC(I,J)=TO*cos(b*FAC);
       IMAGC(I,J)=-TO*sin(b*FAC);

       c=RI_TOP-90-b;

       REALD(I,J)=RI*sin(c*FAC);
       IMAGD(I,J)=-RI*cos(c*FAC);
       RA(I,J)=0+RA_ST;
       [I J RB_ST];


RB(I,J)=RB_ST+(REALA(I,J)+REALB(I,J))+(IMAGA(I,J)+IMAGB(I,J))*sqrt(-
1)+RA_ST;
       [ I J RB_ST REALA(I,J) REALB(I,J) IMAGA(I,J) IMAGB(I,J) RA_ST];
       'CHECK RB';
        %pause
       RC(I,J)=RB(I,J)+(REALC(I,J))+(IMAGC(I,J))*sqrt(-1);

       RD(I,J)=RC(I,J)+(REALD(I,J))+(IMAGD(I,J))*sqrt(-1);

       RB_ST=RB(I,J);
      [I J RB_ST RB(I,J) RC(I,J) RD(I,J)];


   end %end DO J
   RB_ST= real(RB(I,J));
  RA_ST=RD(1,1);
end  %end DO I\
hold on

% SAVE DATA FOR FIRST TOWER AND GO TO END IF THERE IS ONLY
ONE TOWER
RR_2=[RA(1,1:end);
   RB(1,1:end);
   RC(1,1:end);
   RD(1,1:end);
RA(1,1:end)];

hold on
   %HANDLE CASES WHERE I>1
   if(SI>1)
```

```
   RRC(1,1:SJ)=RC (1,1:SJ);
         G_AL=[]
   for I=2:SI
      for J=1:SJ

         g=180-(RI_BOT_COR(I-1,J)+ddd(I,J));
         [I J RI_BOT_COR(I-1,J) ddd(I,J) g];

         hold on

         gg(I,J)=BOTTOM(I,J)*sin(g*FAC)*sqrt(-1)
         gg1(I,J)=sum(gg(I,1:J))
         G_AL=G_AL+gg(I,J)
         %G_ALL(I,J)=G_AL

         RRC(I,J)=RRC(I-1,J)+TOP(I,J)*cos(b*FAC)+gg1(I,J)
         [I J b g G_AL RRC(I-1,J) TOP(I,J)*cos(b*FAC) BOT-
TOM(I,J)*sin(b*FAC)*sqrt(-1)]
          'pause 392'

end  % end of DO J
end % end of DO I


  SI_RRC=size(RRC);
RRD(1:SI_RRC(1,1),1:SI_RRC(1,2))=0;
RRD(1,1)=RA_ST;

 for II=2:SI
    JJ=1;

      RRD(II,JJ)=RRD(II-
1,JJ)+BOTTOM(II,JJ)*cos(g*FAC)+BOTTOM(II,JJ)*sin(g*FAC)*sqrt(-1); %NEW
      [RRD(II,JJ) RRD(II-1,JJ) BOTTOM(II,JJ)];
   end

else  %CASE OF ONLY ONE TOWER
   RR=[];
  RR=RR_2;
  plot(RR,'*')
  plot(RR)
end


RR_FIN=[];
TEST=[];
%%  KEEP ONLY GRID POINTS OF PLATE GRID (ALL)
if ef(1,1)<0 % DISTINQUISH BETEWEEN + OR - INCLINATION
RZ=NaN;
else
```

```
   RZ=0+0*sqrt(-1);
end   %  end DISTINQUISH BETEWEEN + OR - INCLINATION

RR_FIN(1,1:SJ+1)=[RZ  RB(1,1:SJ)];
for k=2:SI+1
   RR_FIN(k,1:SJ+1)=[RRD(k-1,1) RRC(k-1,1:SJ)];
end % RR_FIN ARE POINTS OF RECTANGULAR

RESU=[];
RESU1=[];
SI_RR=size(RR_FIN);
SI_RR1=SI_RR(1,1);
SI_RR2=SI_RR(1,2);
load c:\naval\fair.bak8\RIO_NEW.dat
load c:\naval\fair.bak8\RIO1_NEW.dat
load c:\naval\fair.bak8\RIO2_NEW.dat

RIO1=RIO_NEW(1:end,1:end);
RESU1=RIO1.*RR_FIN;

idd=find (RESU1>0|RESU1<0);
RESU=RESU1(idd);
if(ef(1)>=0)
   RZ=0
   RESU=[RZ
      RESU];
else
end

hold on
id=find(RESU>=0|RESU<0);

RES_PER1=RESU(id);

RES_PER2=mean(mean(RES_PER1));
plot(RES_PER2,'O')
REAL_X=real(RES_PER2);
IMAG_Z=imag(RES_PER2);

ss=size(RESU)
ss=ss(1,1)
for i=1:ss %PLACE POINTS OF PERIMETER IN SEQUENCE COUNTER CLOCK
WISE
 VECTOR(i)= (real(RESU(i))-REAL_X)     +(imag(RESU(i))-IMAG_Z)*sqrt(-1);
 VE_LE(i)=(real(VECTOR(i))^2+imag(VECTOR(i))^2)^2;
 RES(i)=angle(VECTOR(i))/FAC;
end

KEEP=RES;
[OUT id]=sort(RES);
```

```
TOT_X_ALL=real(RES_PER1(id));
TOT_Z_ALL=imag(RES_PER1(id));
ss=size(TOT_X_ALL);
ss=ss(1,1);
TOT_X_ALL(ss+1)=TOT_X_ALL(1);
TOT_Z_ALL(ss+1)=TOT_Z_ALL(1);% end PLACE POINTS OF PERIMETER IN
SEQUENCE COUNTER CLOCK WISE

plot(TOT_X_ALL,TOT_Z_ALL)

%% START PROCESS FOR FRAMES
RESU_FR=RIO1_NEW.*RR_FIN;
RIO1_FR=RIO1_NEW.*RR_FIN;
%% end START PROCESS FOR FRAMES

%% START PROCESS FOR LONGITUDINALS
RIO2_LG=RIO2_NEW .*RR_FIN;
%% end START PROCESS FOR LONGITUDINALS

NO=size(RR_FIN);
NO_FRS=NO(1,1); % NUMBER OF FRAMES
NO_LGS=NO(1,2); % NO OF LONGITUDINALS

% HANDLE FRAMES PREPARE FILES FOR MARKING FRAMES
ic=0; id=0; FR=[]; A=[]; TAB_X=[]; TAB_X_MOVE=[]; id=[]; F=[];ID_FOR=[];
for I=1:NO_FRS
   id=find (RIO1_NEW(I,1:end)~=0);
   si_id=size(id);
   si_id=si_id(1,2)

   if (si_id~=0)
      ic=ic+1;
     'YES';
      F=RIO1_NEW(I,id).*RR_FIN(I,id)
      TAB_X(ic,1:si_id)=F
      %pause
      SI_F=size(F); SI_F=SI_F(1,2);
    SI_MAX(I)=SI_F
    ID_FOR(ic)=si_id
 else
      'NO';
   end

end %  TAB_X CONTAINS DATA FOR MARKING

MAX_SI_F=max(SI_MAX)
TAB_X_MOVE(1:ic,1:2)=0.

ic=0; id=0; FR=[]; A=[];
for I=1:NO_FRS
```

```
   id=find (RIO1_NEW(I,1:end)~=0);
   si_id=size(id); si_id=si_id(1,1);
   if (si_id~=0)
      ic=ic+1;
      'YES';
      F=RIO1_NEW(I,id).*RR_FIN(I,id);
      SI_F=size(F);
      SI_F=SI_F(1,2);

     FR(ic,1:SI_F )= F;
    [ic  SI_F si_id]
    %pause
    FRR=[A  NaN F];
    A=FRR;
   else
      'NO';
   end

   if (I<NO_FRS)
   TAB_X_MOVE(ic,1:2)=[TAB_X(ic,SI_F) TAB_X(ic+1,1)]
else
end
   TAB_X_MOVE(ic,1:2)
   pause
end %TAB_X_MOVE CONTAINS DATA FOR MOVING BURNER


%id=find(min(min(real(TAB_X))))
% PLACE [0.] AS BEGINNING of BURNER MOVING FOR MARKING
id=min(min(real(TAB_X)))
BEG1=[0 ]
BEG2=id
BEG=[BEG1 BEG2]
TAB_X_MOVE=[BEG
   TAB_X_MOVE]% end PLACE [0.]  AS BEGINNING of BURNER MOVING
FOR MARKING

TAB_X_MOVE=TAB_X_MOVE(1:end-1,1:end)% KEEP "TAB_X_MOVE" SIZE
EQUAL TO no OF FRAMES
RE_TAB_X=real(TAB_X); IM_TAB_X=imag(TAB_X);
RE_TAB_X_MO=real(TAB_X_MOVE); IM_TAB_X_MO=imag(TAB_X_MOVE);

  save c:\naval\fair.bak8\RE_TAB_X.dat RE_TAB_X -ASCII -double
  save c:\naval\fair.bak8\IM_TAB_X.dat IM_TAB_X -ASCII -double
  save c:\naval\fair.bak8\RE_TAB_X_MO.dat RE_TAB_X_MO -ASCII -double
  save c:\naval\fair.bak8\IM_TAB_X_MO.dat IM_TAB_X_MO -ASCII -double
  save c:\naval\fair.bak8\ID_FOR.dat ID_FOR -ASCII -double

  SI_MO=size(TAB_X_MOVE); SI_I_MO=SI_MO(1,1); SI_J_MO=SI_MO(1,2);
  SI_SS=size(TAB_X); SI_I_SS=SI_SS(1,1); SI_J_SS=SI_SS(1,2);
```

```
  save c:\naval\fair.bak8\SI_I_MO.dat SI_I_MO -ASCII -double
  save c:\naval\fair.bak8\SI_J_MO.dat SI_J_MO -ASCII -double
  save c:\naval\fair.bak8\SI_I_SS.dat SI_I_SS -ASCII -double
  save c:\naval\fair.bak8\SI_J_SS.dat SI_J_SS -ASCII -double

id=find (TAB_X==0) % replace zeros with NaN'S
  TAB_X(id)=NaN%

  %PLOT DEVELOPED FRAMES PROGRESSIVELY
  SI_MO=size(TAB_X_MOVE); SI_I_MO=SI_MO(1,1); SI_J_MO=SI_MO(1,2);
  SI_SS=size(TAB_X); SI_I_SS=SI_SS(1,1); SI_J_SS=SI_SS(1,2);

    J=0
   TAB_X_MOVE(1,1)=0;                    TAB_X_MOVE(1,2)=TAB_X_MOVE(2,1);
TAB_X(1,1)=0;
  for I=1:SI_I_SS
        J=J+1

        plot(TAB_X_MOVE(J,1:SI_J_MO),'g')

        pause

        hold on

        for II=2:SI_J_SS
       plot(TAB_X(I,II-1:II),'b')
       if II==2
       else
        plot(TAB_X(I,II-1:II),'b^')
    end

         pause
   end
end
 %SAVE DATA for reproduction of "FRAMES" By  "test_test_test8_mad1_fr.m"
 TAB_X_RE=real(TAB_X);TAB_X_IM=imag(TAB_X);

TAB_X_MOVE_RE=real(TAB_X_MOVE);TAB_X_MOVE_IM=imag(TAB_X_M
OVE);

 save c:\naval\fair.bak8\TEMP\TAB_X_RE.dat TAB_X_RE -ASCII -double
 save c:\naval\fair.bak8\TEMP\TAB_X_IM.dat TAB_X_IM -ASCII -double
 save c:\naval\fair.bak8\TEMP\TAB_X_MOVE_RE.dat TAB_X_MOVE_RE -ASCII
-double
 save c:\naval\fair.bak8\TEMP\TAB_X_MOVE_IM.dat TAB_X_MOVE_IM -ASCII
-double
 %SAVE DATA for reproduction of "FRAMES" By  "test_test_test8_mad1_fr.m"

    delete SI_I_SS.mat
```

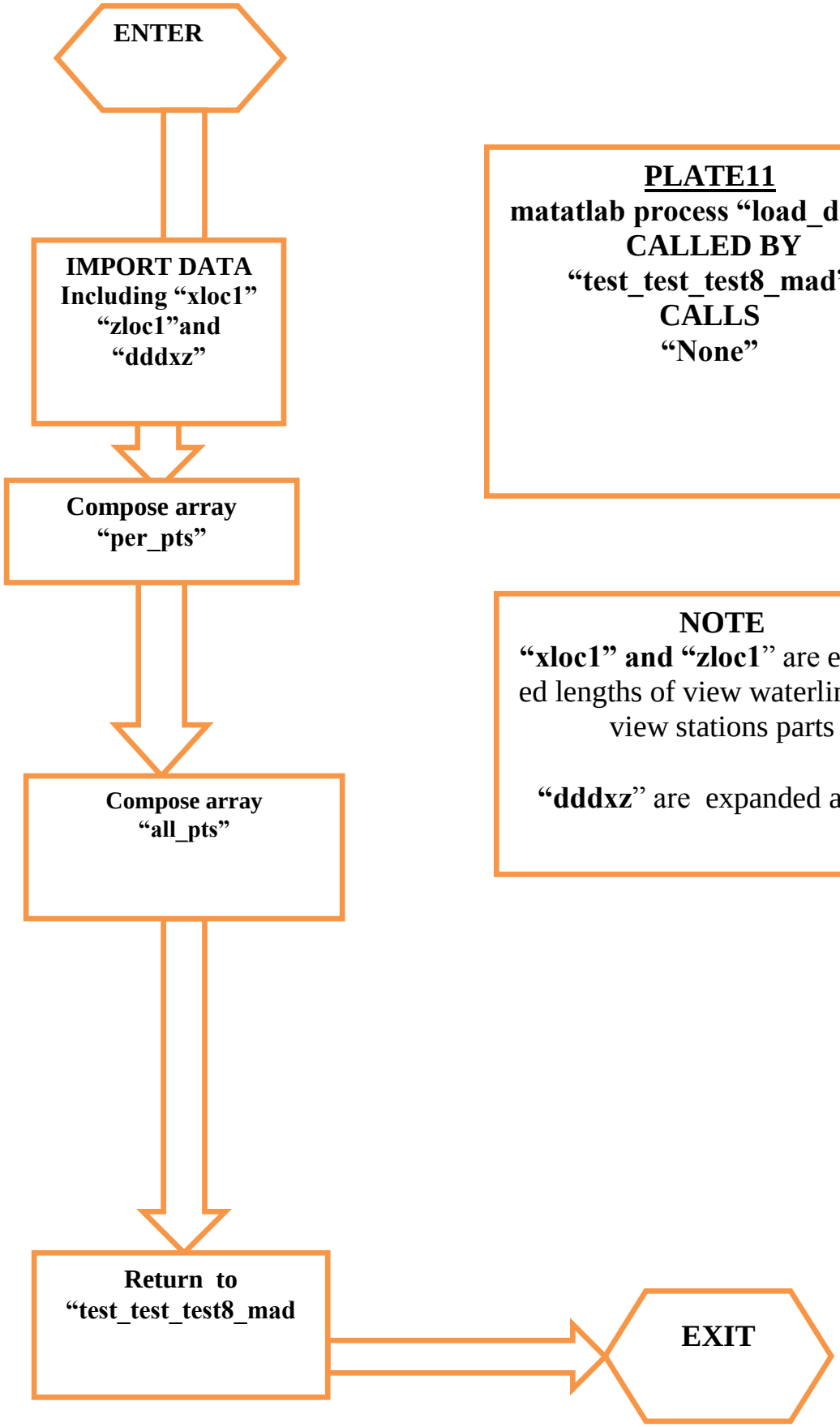
```
   delete SI_J_MO.mat
   delete TAB_X_MOVE.mat
   delete TAB_X.mat
   delete SI_J_SS.mat

   save SI_I_SS; save SI_J_MO; save TAB_X_MOVE; save TAB_X; save SI_J_SS;

   hold off

% PREPARE MARK CODE FOR FRAMES (1 FOR MARK, 0 for JUMP)
SI_A=size(A);
SI_A=SI_A(1,2);
FR_CODE(1:SI_A)=0;
id=find(A>=0|A<=0);
FR_CODE(id)=1;
[FR_CODE' A'];
% end PREPARE MARK CODE FOR FRAMES (1 FOR MARK, 0 for JUMP)

plot(A.*FR_CODE);
'pause 616  PRESS_RETURN REPEATDLY TO CONTINUE'
pause
% HANDLE LONGITUDINALS
ic=0;
LG=[];
B=[];
for I=1:NO_LGS
   id=find (RIO2_NEW(1:end,I)~=0);
   si_id=size(id);
   si_id=si_id(1,1);
    if (si_id~=0)
      ic=ic+1;
      'YES';

      L=RIO2_NEW(id,I).*RR_FIN(id,I);

      SI_L=size(L);
      SI_L=SI_L(1,2);
      REAL_L=real(L);
      RREAL_L=REAL_L';

      IMAG_L=imag(L);
      IIMAG_L=IMAG_L';
      LL=RREAL_L+IIMAG_L*sqrt(-1);
    LGG=[B  NaN LL];
    B=LGG;
 else
      'NO';
   end
end
% end HANDLE LONGITUDINALS
```

```
% PREPARE MARK CODE FOR FRAMES (1 FOR MARK, 0 for JUMP)
SI_B=size(B);
SI_B=SI_B(1,2);
LG_CODE(1:SI_B)=0;
id=find(B>=0|B<=0);
LG_CODE(id)=1;
[LG_CODE' B'];
% end PREPARE MARK CODEFOR FRAMES (1 FOR MARK, 0 for JUMP)

x1loc1=xloc1(1:end,1:end-1);
x2loc1=x1loc1(1:end-1,1:end);
x3loc1=x2loc1(:);
size(x3loc1);
size(dat3);
size(A);

A=dat3-x3loc1;
x1loc1=[];
x2lox1=[];
x3loc1=[];
A=[];
x1loc1=xloc1(1:end-1,1:end);
x2loc1=x1loc1(1:end,1:end-1);
x3loc1=x2loc1(:);
size(x3loc1);
size(dat3);

A=dat3-x3loc1;
d=dat3;
count=0;
p=0;
dor=[];
count=0;
for i=1:si_LL-1
    for j=1:si_HH-1
      count=count+1;
       p=xloc1(i,j);
       dor(count)=p;
       %pause
    end
 end

%SAVE FILE dddxz
dddxz=dddxz_all
%end SAVE FILE dddxz
dor;

A=min(TOT_X_ALL)
B=min(TOT_Z_ALL)
```

```
TOT_X_PLOT=[A
   TOT_X_ALL(1)
     TOT_X_ALL]
  TOT_Z_PLOT=[B
   TOT_Z_ALL(1)
     TOT_Z_ALL]

SI_PLOT_X=size(TOT_X_PLOT); SI_PLOT=SI_PLOT_X(1,1);
save c:\naval\fair.bak8\SI_PLOT.dat SI_PLOT -ASCII -double

  % USEFUL FOR PTOGRESSIVE TESTING ONLY
  plot(  TOT_X_PLOT(1:2),TOT_Z_PLOT(1:2),'r')
  hold on
  plot(  TOT_X_PLOT(1:2),TOT_Z_PLOT(1:2),'*')

for II=3:ss+2 % PLOT PLATE PERIMETER PROGRESSIVELY

 plot(  TOT_X_PLOT(II:II+1),TOT_Z_PLOT(II:II+1),'b'  )
 plot(  TOT_X_PLOT(II:II+1),TOT_Z_PLOT(II:II+1),'^'  )
 axis equal
   pause
end% end PLOT PLATE PERIMETER PROGRESSIVELY

delete TOT_X_PLOT.mat
delete ss.mat
save TOT_X_PLOT; save ss;
save c:\naval\fair.bak8\TOT_X_PLOT.dat TOT_X_PLOT -ASCII -double
save c:\naval\fair.bak8\TOT_Z_PLOT.dat TOT_Z_PLOT -ASCII -double
save c:\naval\fair.bak8\ss.dat ss -ASCII -double

     hold on
    load c:\naval\fair.bak8\xelim.dat
load c:\naval\fair.bak8\zelim.dat

xelim(end)=xelim(1)
zelim(end)=zelim(1)
id_x=find(min(xelim))
xxelim=xelim(id_x); zzelim=zelim(id_x);
si_xe=size(xxelim); si_xe=si_xe(1,1);
if(si_xe>1);
   id_z=find(min(zzelim))
   x_sub=xxelim(id_z)

else
   x_sub=xxelim(id_x)

end
xelim1=xelim-x_sub
zelim1=zelim
```

```
z_sub=min(zelim1)
zelim1=zelim1-z_sub
area(xelim1,zelim1)
plot(xelim1,zelim1)
plot( TOT_X_PLOT(3:ss+3),TOT_Z_PLOT(3:ss+3),'r')

save c:\naval\fair.bak8\TEMP\TOT_X_PLOT.dat TOT_X_PLOT -ASCII -double
save c:\naval\fair.bak8\TEMP\TOT_Z_PLOT.dat TOT_Z_PLOT -ASCII -double
save c:\naval\fair.bak8\TEMP\ss.dat ss -ASCII -double

save c:\naval\fair.bak8\TEMP\xelim.dat xelim -ASCII -double
save c:\naval\fair.bak8\TEMP\zelim.dat zelim -ASCII -double
```

ANNEX(11) + PLATE(11)
DOCUMENTATION OF MATLAB PROCESS
“load_data 8”


ENTER
IMPORT DATA
Including "xloc1"
"zloc1"and
"dddxz"
Compose array
"per_pts"
Compose array
"all_pts"
Return to
"test_test_test8_mad
EXIT
PLATE11
matatlab process "load_data8"
CALLED BY
"test_test_test8_mad"
CALLS
"None"
NOTE
"xloc1" and "zloc1" are expand-
ed lengths of view waterlines and
view stations parts
"dddxz" are expanded angles

MATLAB_MODULE “models18_7”

```
% 1)THIS PROGRAM APPLIES (ONLY!!!!)  TO PRODUCE DATA TO CUT MODELS OF TRANSVERSE
%FRAMES FOR DEVELOPED PART ONLY AS PRODUCED BY Matlab module "test_test_test8_mad1"

%LOAD OR MODIFY PLATE DATA
load c:\naval\fair.bak8\gendat8.dat
OFF=gendat8(1);  PLATE_X=gendat8(2);  PLATE_Y=gendat8(3);  DEL-TA=gendat8(4); BRA=gendat8(5);
['  OFF  ''  PLATE_X  ''  PLATE_Y  ''DELTA  '  'BRA']
[OFF PLATE_X PLATE_Y DELTA BRA]
%end  %LOAD OR MODIFY PLATE DATA

%SAVE  PLATE DATA
gendata8=[OFF PLATE_X PLATE_Y DELTA]
save c:\naval\fair.bak8\gendata8.dat gendata8 -ASCII -double
%end SAVE  PLATE DATA

%LOAD ADDITIONAL DATA
load c:\naval\fair.bak8\LLENGTHS.dat
load c:\naval\fair.bak8\HHEIGHTS.dat
load c:\naval\fair.bak8\myplot.dat
load c:\naval\fair.bak8\RIO1_NEW.dat
load c:\naval\fair.bak8\offsets1.dat
%end LOAD ADDITIONAL DATA

m_xcal=LLENGTHS
m_zcal=HHEIGHTS
OFF=gendat8(1)

SI_LL=size(LLENGTHS)
SI_LL=SI_LL(1,1)
SI_HH=size(HHEIGHTS)
SI_HH=SI_HH(1,1)

myplo=offsets1

IC=0; MY=[]; HH=[]; ICC=[]; HHH=[];

for I=1:SI_LL %FIND WIDTHS (MY) and HEIGHTS (HHH) CORRESPONDING TO grid of developed frames
   id=find(RIO1_NEW(I,1:end)==1)
   [I id RIO1_NEW(I,1:end)]
    SI_ID=size(id)
   SI_ID=SI_ID(1,2)
   if( SI_ID>1)  % FIND POINTS OF FRAMES OF DEVELOPED PLATE (RIO_NEW1 HAS VALUE 1)
   IC=IC+1
```

```
   MY(IC,1:SI_ID)=myplo(I,id) % CORRESPONDING BREADTHS
   [MY(IC,1:SI_ID)]

   'CHECK'
   [SI_ID id]
   HH=[]
   HH(1:SI_ID)=HHEIGHTS(id)
   HHH(IC,1:SI_ID)=  HH(1:SI_ID)      %HHH(IC,1:SI_ID)=HH(I,id)  %  CORRE-
SPONDING WATERLINES
else
end
end %end FIND WIDTHS (MY) and HEIGHTS (HHH)  CORRESPONDING TO
grid of developed frames

SI_MY=size(MY)
for J=1:SI_MY(1,2) %  PLACE NaN  IN ALL  (0,0) POINTS
for I=1:IC
   if (  (MY(I,J)==0) & (HHH(I,J)==0) )
      MY(I,J)=NaN
      HHH(I,J)=NaN
   end
end
end  %  end PLACE NaN  IN ALL  (0,0) POINTS

HHHH=HHH
MYY=MY

% fill with NaN's COLUMNS HAVING ZEROS IN ALL LINES
 IC=0
 MYYY=[]
 HHHHH=[]
 for J=1:SI_MY(1,2)
 if(  MYY(1:SI_MY,J)==0)
 else
    IC=IC+1
MYYY(1:SI_MY,J )=MYY(1:SI_MY,J)
HHHHH(1:SI_MY,J )=HHHH(1:SI_MY,J)
end
end %end fill with NaN's COLUMNS HAVING ZEROS IN ALL LINES

NEW_SI=size(MYYY) % REPLACE ZEROS WITH NaN's AT ANY PLACE

 for J=1:NEW_SI(1,2)
  for I=1:NEW_SI(1,1)
     if (MYYY(I,J)==0    )
        MYYY(I,J)=NaN
        HHHHH(I,J)=NaN
     else
     end
  end
```

```
end % end REPLACE ZEROS WITH NaN's AT ANY PLACE

HHHHH=HHHHH' % REVERSE ARRAYS
SI=size(HHHHH)
SI_LL=SI(1,1)
SI_HH=SI(1,2)

H_BOT=[]
A={}

for J=1:SI_HH
A=HHHHH(1:end,J)
id=find(A>=0)
HH_BOT_MIN(J)=min(A(id))+DELTA
HH_BOT_MAX(J)=max(A(id))-DELTA
end
MYYY=MYYY'   % end REVERSE ARRAYS

NEW_SI_CO=NEW_SI(1,1)
NEW_SI_LE=NEW_SI(1,2)

 for I=1:NEW_SI_LE
    for J=1:NEW_SI_CO
       HHHHH1(I,J)= HHHHH(I,J)- HHHHH(1,J)
    end
 end

 axis equal

 for I=1:NEW_SI_CO % SEE RESULTS IN REAL 2D SHAPE FOR EVERY STA-
TION
   %hold on
  plot( MYYY(1:end,I),HHHHH1(1:end,I))
  MYY=MYYY(1:end,I)
  HHH=HHHHH1(1:end,I)
  save c:\naval\fair.bak8\MYY.dat MYY -ASCII -double
  save c:\naval\fair.bak8\HHH.dat HHH -ASCII -double
  pause
end  % SEE RESULTS IN REAL 2D SHAPE FOR EVERY STATION
 m_dddd=MYYY
SI_D=size(m_dddd); SI_D_LE=SI_D(1,1);  SI_D_CO=SI_D(1,2)

COUNT=0

BRA=gendat8(5) % WIDTH OF   FRAME MODELS AT BOTTOM

PLX1=0;  PLX2=PLATE_X;  PLX3=PLATE_X;   PLX4=0;  PLX5=0   % PLOT PE-
RIMETER OF PLATE
PLY1=0; PLY2=0; PLY3=PLATE_Y;  PLY4=PLATE_Y; PLY5=0
```

```
plot([PLX1,PLY1], [PLX2,PLY2], [PLX3,PLY3], [PLX4,PLY4],'*')% end PLOT
PERIMETER OF PLATE

hold on
PL(1)=0
PL(2)=PLATE_X
PL(3)=PLATE_X+PLATE_Y*sqrt(-1)
PL(4)=PLATE_Y*sqrt(-1)
PL(5)=0
plot(PL)
hold on % end PLOT PERIMETER OF PLATE AT CUT MACHINE


 MYYY_SAV=MYYY % KEEP BREATHS
 HHHHH_SAV=HHHHH % KEEP WATERLINES

 MIND=0
 CORRD=[]

 SI_ALL=size(MYYY)
 IC=SI_ALL(1,1)
 SI_HH=SI_ALL(1,2)

for I=1:SI_HH
      MIND(I)= min(MYYY(1:end,I))
 CORRD(1:IC,I)=MYYY(1:IC,I)-MIND(I)+BRA %CORRECTED OVERALL OFF-
SETS OF MODELS
end

MA=max(max(CORRD))
MAX_X=MA+OFF
    if(PLATE_X<MAX_X)
       'FURTHER PROCESS STOPS SINCE PLATE LENGTH (PLATE_X) '
       'IS SMALLER THAN MAXIMUM FRAME OFFSET'
       'THE REQUIRED MIN PLATE LENGTH IS'
       [MAX_X]
       pause
       A='ERROR'
      save c:\naval\fair.bak8\ERROR_MAD.dat A -ASCII -double
       return
    else
    end
% PREPARATION TO FIND NO OF FRAMES THAT CAN FIT IN EVERY HOR-
IZONTAL ZONE
STEPX=[]
STEPX(1)=0
  for I=2: SI_HH
STEPX(I)=OFF+max(CORRD(1:SI_LL,I))
end % STEPX(I) IS TOTAL WIDTH OF EVERY FRAME MODEL
```

```
STEPXX=[]
  for I=1: SI_HH
STEPXX(I)=OFF+max(CORRD(1:SI_LL,I))
end % STEPXX(I) IS TOTAL WIDTH OF EVERY FRAME MODEL

A=0
ACC_STEPX=[]
for I=1: SI_HH
A=A+STEPX(I)
ACC_STEPX(I)=A
end % ACC_STEPX IS THE PROGRSSIVE ACCUMULATED LENGTH OF
FRAME MODELS

C=0
ACC_STEPXX=[]
for I=1: SI_HH
C=C+STEPXX(I)
ACC_STEPXX(I)=C
end % ACC_STEPXX X IS THE PROGRSSIVE ACCUMULATED LENGTH OF
FRAME MODELS

STEPZ=[]
DIFZ=[]
STEPZ(1)=OFF
for I=1: SI_HH
   MAXZ(I)=max(HHHHH1(1:SI_LL,I))
   MINZ(I)=min(HHHHH1(1:SI_LL,I))
   DIFZ(I)=MAXZ(I)-MINZ(I)
  STEPZ(I)=OFF+ DIFZ(I) %STEPZ(I) IS TOTAL HEIGHT OF EVERY FRAME
MODEL
end


STEPZZ=[]
DIFZ=[]
for I=1: SI_HH
   MAXZ(I)=max(HHHHH1(1:SI_LL,I))
   MINZ(I)=min(HHHHH1(1:SI_LL,I))
   DIFZ(I)=MAXZ(I)-MINZ(I)
STEPZZ(I)=OFF+ DIFZ(I) %STEPZZ(I) IS TOTAL HEIGHT OF EVERY FRAME
MODEL
end

B=0
ACC_STEPZ=[]
for I=1: SI_HH
B=B+STEPZ(I)
ACC_STEPZ(I)=B
end % ACC_STEPZ IS THE PROGRSSIVE ACCUMULATED HEIGHT OF
FRAME MODELS
```

```
D=0
ACC_STEPZZ=[]
for I=1: SI_HH
D=D+STEPZZ(I)
ACC_STEPZZ(I)=D
end % ACC_STEPZZ IS THE PROGRSSIVE ACCUMULATED HEIGHT OF FRAME MODELS
% end PREPARATION TO FIND NO OF FRAMES THAT CAN FIN IN EVERY HORIZONTAL ZONE

% CHECK NUMBER OF FRAMES THAT CAN BE FITTED IN THE WIDTH OF PLATE
IM=sqrt(-1)
CUT=[]
axis equal
CUT=[]
CUT1=[]
CUT2=[]
DIF=[]
CD=[]
STE(I)=0

CUT_SAV=[]
SI_HHHHH1=size(HHHHH1)
SI_LL_SAV=SI_HHHHH1(1,2)

IC(1:SI_HH)=0
IC1=[]
IC1(1)=0
AR=[]
AR1=[]
AR2=[]

I=1
for I=1+IC1(I):SI_HH
   A=0
   I_COUNT(1)=0
   SI_IC1=size(IC1)
   SI_IC1=SI_IC1(1,2)
   if(I<=SI_IC1)

  for J=1+IC1(I):SI_HH
  A=A+STEPXX(J)
   if(A<=PLATE_X)

      I_COUNT=I_COUNT+1
      IC(I)=I_COUNT

   AR(I,J)=A
```

```
    IC1(I+1)=IC1(I)+IC(I)
 else
 end

end
else
end
end% end FIND SUCCESIVE SUMS OF STEPXX (1:SI_HH  ,2:SI_HH etc)

SI_AR=size(AR)
SI_AR_LI=SI_AR(1,1)
SI_AR_CO=SI_AR(1,2)

AR1(1:SI_AR_LI,1:SI_AR_CO)=0
for I=1:SI_AR_LI
id=find(AR(I,1:end)>0)
si_id=size(id)
si_id=si_id(1,2)
AR1(I,1:si_id)=AR(I,id)
end

ICO=0
AR3=[]
 for J=1:SI_AR_CO
if(sum(AR1(1:SI_AR_LI,J))~=0)
ICO=ICO+1
AR3(1:SI_AR_LI,ICO)=AR1(1:SI_AR_LI,J)
else
end
end

ZE=[]
SI_AR3=size(AR3)
SI_AR3_LI=SI_AR3(1,1)
SI_AR3_CO=SI_AR3(1,2)
ZE(1:SI_AR3_LI,1)=0
AR4=[ZE AR3]
 SI_AR4=size(AR4)
 SI_AR4_LI=SI_AR4(1,1) %NO OF HORIZONTAL ZONES
 SI_AR4_CO=SI_AR4(1,2) % TOTAL NO OF FRAMES (SI_HH)

 if(SI_AR4_LI>1)

 id=[]
  for I=1: SI_AR4_LI

     ido=find(AR4(I,1:end)>0)
     si=size(ido); si=si(1,2)
     id(I,1:si)=find(AR4(I,1:end)>0)
  end
```

```
 idd=id(1:end,1)-1 %idd is NO OF LONG START POINTS IN EVERY HORIZON-
TAL ZONE
else
   idd=[1:SI_HH]
end

 % START PROCESS OF MARKING AND CUTTING FRAMES
 for L=1:SI_LL_SAV
     for J=2:SI_D_LE

   DIF_Z(J-1,L)=HHHHH1(J,L)-HHHHH1(J-1,L)
   if( DIF_Z(J-1,L)>0|DIF_Z(J-1,L)<0)

  else
   DIF_Z(J-1,L)=0
  end
end
end
DIF_Z=DIF_Z'

   for I=1:SI_AR4_LI % FIND NO OF BEGINNING POINTS IN EVERY ZONE
      id=find(AR4(I,2:SI_AR4_CO)>0)
      si_id=size(id)
      si_id1=si_id(1,2)
      SI_J(I)=si_id1
   end % SI_J(I) IS NO OF BEGINNING POINTS IN EVERY ZONE (I)

   I=0
    for IK=1:SI_AR4_LI

        VERT=[]
          if (IK==1)
             VERT=0
          else

             VERT=max(imag(CUT(1:end,I)))
          end
          NO_FRS=sum(SI_J)
       for J=1:SI_J(IK)

         I=I+1
         CUT(1,I)=AR4(IK,J)+VERT*sqrt(-1);  CD(1,I)=0;

          CUT(2,I)=CUT(1,I)+OFF+.0000001i;  CD(2,I)=0;

   CUT(3,I)=CUT(2,I)+(OFF)*sqrt(-1);                    CD(3,I)=0;
   CUT(4,I)=CUT(2,I)+(OFF)*sqrt(-1);                    CD(4,I)=1;
   CUT(5,I)=CUT(3,I)+BRA;                             CD(5,I)=1

     % GENERATE MORE POINTS to the offsets side of frame
```

```
        for II=6:6+SI_D_LE-2

        II_II(I)=6+SI_D_LE-2 % UPPER CORNERS

         DIF=MYYY(II-4,I)-MYYY(II-5,I)

    CUT(II,I)=CUT(II-1,I)+DIF + DIF_Z(I,II-5)*sqrt(-1);        CD(II,I)=1;

      X_VAL=(real(CUT(II,I)))^2 % PUT ZEROS IN PERIMETER POINTS AT
Z_CAL'S HAVING ZERO VALUES
      Z_VAL=(imag(CUT(II,I)))^2
      XZ_VAL=X_VAL+Z_VAL
     if(  (XZ_VAL>=0)     )
      CUT(II,I)=CUT(II,I); CD(II,I)=1; % UPPER RIGHT CORNER OF ALL
FRAMES

      else
        CUT(II,I)=CUT(II-1,I) % UPPER CORNER OF FRAME
      end
   end

   CUT_SAV(I)= CUT(II,I)

   CUT(II+1,I)=CUT(II,I)-real(CUT(II,I))+real(CUT(3,I)); CD(II+1,I)=1;
   CUT(II+2,I)=CUT(II+1,I)-(imag(CUT(II+1,I))-imag(CUT(3,I)))*sqrt(-1);
CD(II+2,I)=1;% END DATA FOR CUTTING PERIMETER
   IO=II+2

   % START CUTTING OF HOLE
   CUT(II+3,I)=CUT(II+2,I);  CD(II+3,I)=0;

   CUT(II+4,I)=CUT(II+3,I)+OFF*sqrt(-1);    CD(II+4,I)=0;

   CUT(II+5,I)=CUT(II+4,I)+OFF;  CD(II+5,I)=0;

   III=II+9

  JJ=2

    CUT(III+JJ-5,I)=CUT(II,I)-2*OFF-OFF*sqrt(-1); CD(III+JJ-5,I)=0;
    CUT(III+JJ-4,I)=CUT(II+1,I)+OFF-OFF*sqrt(-1); CD(III+JJ-4,I)=1;
    CUT(III+JJ-3,I)=  real(CUT(III+JJ-4,I))  +imag(CUT(II+5,I))*sqrt(-1);  CD(III+JJ-
3,I)=1;

   CUT(III+JJ-2,I)=CUT(III+JJ-6+3,I);     CD(III+JJ-5+4,I)=0;
end
end

    [III+JJ-5 III+JJ-4 III+JJ-3 III+JJ-2]
```

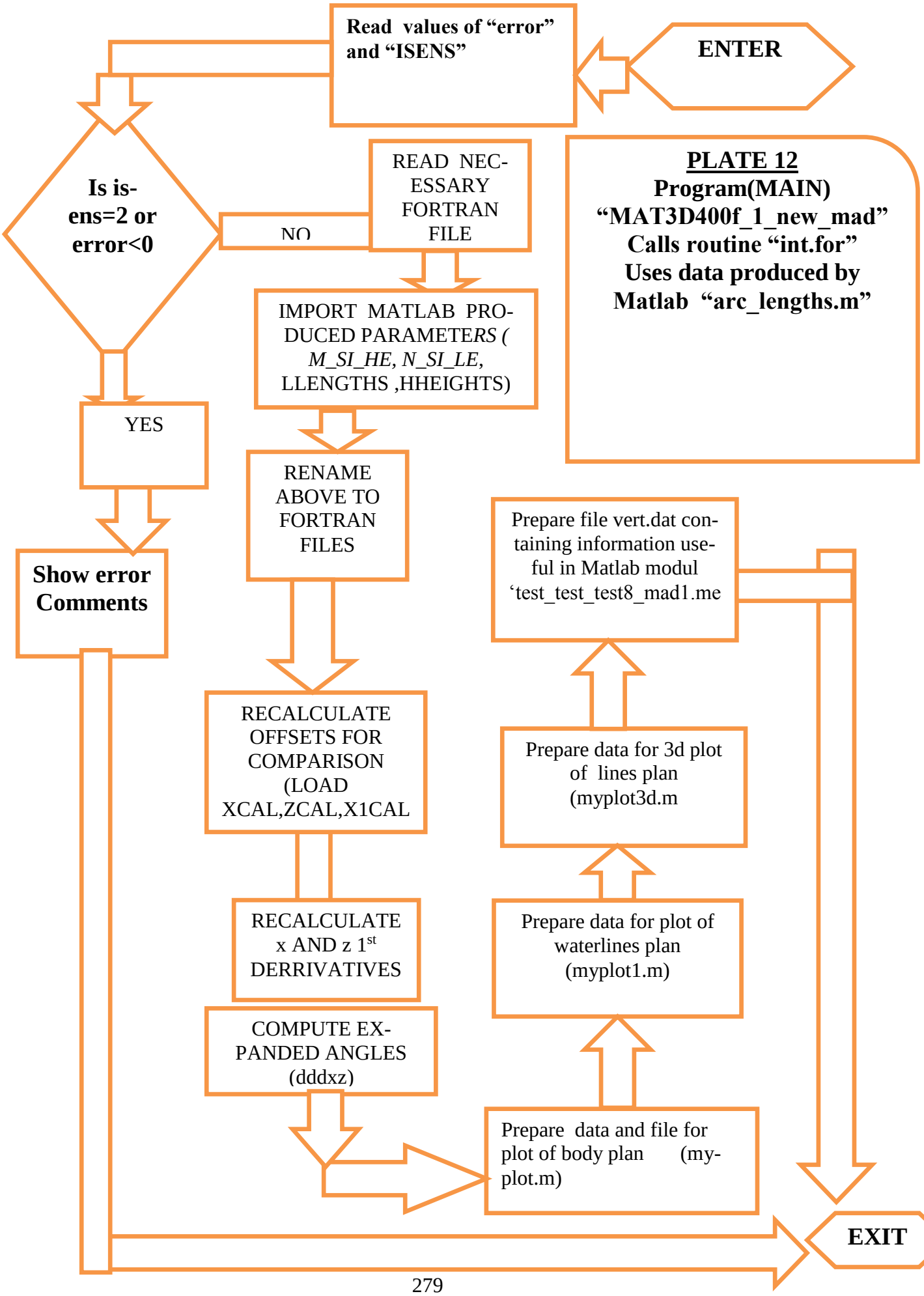
```
    % GENERATING MORE OFFSETS OF FAIRED FRAMES
    CU1=[]
    CU2=[]
    CU3=[]
SI_CUT=size(CUT)
SI_CUT=SI_CUT(1,1)
for I=1:SI_HH
CU1(1:5,I)=CUT(1:5,I)
CU2(1:4,I)=CUT(5:8,I)
SI_CU2=size(CU2)
SI_CU2=SI_CU2(1,1)

for J=2:SI_CU2
   if CU2(J,I)-CU2(J-1,I)==0

      CU2(J,I)=CU1(end,I)
else

end
end
NO_STEPS=9
Z_POS=[]
COR_POS=[]
id=[]
MIN_Z=min(HHHHH1(1:end,I))+OFF
[I]
[HHHHH1(1:end,I)]
id=find(HHHHH1(1:end,I)>=0)

'PAUSE 452'
Z_POS=OFF+HHHHH1(id,I)
COR_POS=CORRD(id,I)
id=[]

MAX_Z=max(HHHHH1(1:end,I))+OFF

SPAN=MAX_Z-MIN_Z
STEP=SPAN/NO_STEPS
Z=[MIN_Z:STEP:MAX_Z MAX_Z]
[MIN_Z MAX_Z SPAN STEP]
[Z]
'pause 464'

SI_PO=size(Z)
SI_PO=SI_PO(1,2)
XX=[]

XX=interp1(Z_POS,COR_POS,Z,'cubic')
[Z_POS COR_POS]
[Z]
```

```
'pause 473'
[I]
   [imag(CU2(1:end,I))]
   [real(CU2(1:end,I))]
pause
plot(XX,Z)
plot(XX,Z,'^')
'pause 480'

CU22(1:SI_PO,I)=XX'+Z'*sqrt(-1)+real(CUT(5,I))-
real(CUT(5,1))+imag(CU1(end,I))*sqrt(-1)+OFF-OFF*sqrt(-1)
[CU22]
'pause 484'
%pause
CU3(1:SI_CUT-IO+2,I)=CUT(IO-1:SI_CUT,I)
end % end GENERATING MORE OFFSETS OF FAIRED FRAMES

  CUT_SMOOTH=[CU1
   CU22
   CU3]
CD3_CU=[]
SI_CU3=size(CU3)
SI_CU3=SI_CU3(1,1)

CD3_CU(1:SI_CU3,1:SI_HH)=CD(end-(SI_CU3-1):end,1:SI_HH)

SI_CU22=size(CU22)
SI_CU22=SI_CU22(1,1)
CD22_CU(1:SI_CU22,1:SI_HH)=1

CD1_CU=CD(1:5,1:SI_HH)
CD_CUT=[CD1_CU
   CD22_CU
   CD3_CU]

CUT_CUT=CUT_SMOOTH(:)

CD1_CUT=[]

  for I=1:SI_HH
   CD1_CUT=[CD1_CUT
    CD_CUT(1:end,1)]
end
CUT=[]
CUT=[CUT_CUT CD1_CUT(1:end,1)]

CUT=[real(CUT_CUT) imag(CUT_CUT) CD1_CUT(1:end,1)]

% MODIFY CUT_CUT TO ELIMINATE HOLES ON MODELS IF THEIR
BREADTHS IS TOO SMALL
```

```
hold off
I_TRUE=0
I_FALSE=0
for I_FR=1:NO_FRS
   I_FR
   pause
   if imag( (  CUT_CUT((I_FR-1)*25+17)-CUT_CUT((I_FR-1)*25+5) ))<2*OFF
       CUT_CUT( (I_FR-1)*25+22:(I_FR-1)*25+25)=CUT_CUT( (I_FR-1)*25+21);
      I_TRUE=I_TRUE+1
   else
    I_FALSE=I_FALSE+1
   end
end
%end MODIFY CUT_CUT TO ELIMINATE HOLES ON MODELS IF THEIR
BREADTHS IS TOO SMALL  hold off

 % PLOT FRAME MODELS PROGRESSIVELY
SI=size(CUT_CUT)
 for I=2:SI

    hold on
    axis equal

     if (CD_CUT(I)==0&CD_CUT(I-1)==0)

        plot(CUT_CUT(I-1:I),'r')
         plot(CUT_CUT(I-1:I),'r*')
         [I CD_CUT(I)]
        pause

    else
        hold on
        plot(CUT_CUT(I-1:I),'b')
        plot(CUT_CUT(I-1:I),'b*')
        [I CD_CUT(I)]
        pause

    end

 end
 % end PLOT FRAME MODELS PROGRESSIVELY

 % SAVE RESULTS
 RE_CUT_CUT=real(CUT_CUT)
 IM_CUT_CUT=imag(CUT_CUT)

save c:\naval\fair.bak8\TEMP\RE_CUT_CUT.dat RE_CUT_CUT -ASCII -double
save c:\naval\fair.bak8\TEMP\IM_CUT_CUT.dat IM_CUT_CUT -ASCII -double
save c:\naval\fair.bak8\TEMP\CD_CUT.dat CD_CUT -ASCII -double
```

```
save c:\naval\fair.bak8\RE_CUT_CUT.dat RE_CUT_CUT -ASCII -double
save c:\naval\fair.bak8\IM_CUT_CUT.dat IM_CUT_CUT -ASCII -double
save c:\naval\fair.bak8\CD_CUT.dat CD_CUT -ASCII -double
% end SAVE RESULTS
```

ANNEX(12) + PLATE(12)
DOCUMENTATION OF FORTRAN PROGRAM
“mat3d400f_1_new_mad.for”


Read values of "error" and "ISENS"
ENTER
Is is-ens=2 or error<0
NO
READ NEC-ESSARY FORTRAN FILE
PLATE 12
Program(MAIN)
"MAT3D400f_1_new_mad"
Calls routine "int.for"
Uses data produced by Matlab "arc_lengths.m"
IMPORT MATLAB PRO-DUCED PARAMETERS (M_SI_HE, N_SI_LE, LLENGTHS ,HHEIGHTS)
YES
RENAME ABOVE TO FORTRAN FILES
Show error Comments
Prepare file vert.dat con-taining information use-ful in Matlab modul 'test_test_test8_mad1.me
RECALCULATE OFFSETS FOR COMPARISON (LOAD XCAL,ZCAL,X1CAL
Prepare data for 3d plot of lines plan (myplot3d.m
RECALCULATE x AND z 1st DERRIVATIVES
Prepare data for plot of waterlines plan (myplot1.m)
COMPUTE EX-PANDED ANGLES (dddxz)
Prepare data and file for plot of body plan (my-plot.m)
EXIT

FORTRAN PROGRAM "mat3d400f_1_new_mad.f

```
------------------------------------
      PROGRAM DEVELOP
C       SECOND PHASE OF PRODUCING RESULTS  BY LINEAR PROGRAM-
MING METHOD

C     LINEAR PROGRAMMING PROBLEM
C     MAXIMIZE Z SUBJECT TO CONSTRAINTS A*X<=B1

C     INPUT DATA
C        N=NUMBER OF PARAMETERS COMING FROM MATLAB=33+2*(n-
2)*m+2*(m-2)*n
c     +2*(m-2)*(n-2)
C     M=NUMBER OF EQUATIONS=2*n*m+(n-2)*m+n*(m-2)
C     nn1=NO OF STATIONS WHERE RESULTS ARE REQUIRED
C     mm1=NO OF WATERLINES WHERE RESULTS ARE REQUIRED
C     nbf=33+2*n*(m-2)
C     ncf=33+2*n*(m-2)+2*(n-2)*m
c     ndf=33+2*n*(m-2)+2*(n-2)*m+2*(n-2)*(m-2)
C     n2=NUMBER OF STATIONS WHERE DATA GIVEN
C     m2=no OF WATERLINES WHERE DATA GIVEN
c     fac=FACTOR OF AMPLIFICATION
C     scale=SCALE FACTOR
C     SPAZ=VERTICAL SPACING OF REQUIRED OUTPUT
C     SPAX=LONG SPACING OF REQUIRED OUTPUT

C     isens:
C       0 PRODUCE OFFSETS AND PLATE INFORMATION
C       1 PRODUCE OFFSETS, PLATE AND CUTTING TAPES
C       2 PRODUCE ONLY OFFSETS

C     ALL ABOVE DATA ARE read FROM  FILE 'mat3d1.dat'
C     ADDITIONAL INFORMATION IS TRANDFERED FROM MATLAB

      IMPLICIT REAL*8(A-H,O-Z)

      integer engopen,enggetmatrix,mxcreatefull,mxgetpr
          integer ep,T,T1,T2      ,T3
          integer engputmatrix,engevalstring,engclose
          integer status

          real*8 : :xxx(1600)
      character*5 aaa1
      character*2 bbb1
          DIMENSION A(1400,1400),B(1,1),BMIN(1,1),CJ(1400),XB(1600,1),
     *BB(1400),B1(1400,1),T(1400),C(1400),XCAL(1600),
     *ZCAL(1600),dddd(1600,1600),DDDX(1600,1600),DDDZ(1600,1600)
     *,xxcal(1600,
     *1600),zzcal(1600,1600),DDDXZ(1600,1600),DDXX(1600,1600)
     *,DDZZ(1600,1600),dddz_exp(1600,1600),
```

```
*ZLOC(1600,1600),XLOC(1600,1600),XBOTLE(1600,1600),
*YBOTLE(1600,1600),
*XBOTRI(1600,1600),YBOTRI(1600,1600),XTOPLE(1600,1600),
*YTOPLE(1600,1600),
*XTOPRI(1600,1600),YTOPRI(1600,1600),XXTOPLE(4000,4000)
*,YYTOPLE(3000,3000)
*,THETAR1(1600,1600),xixidot(1600,1600),phiphi(1600,1600),
*yytop(1600),
*yytople1(1600,1600),xxtople1(1600,1600),angle1(1600,1600),
*r1r1(1600,1600)
*,xxlim(1600),zzlim(1600),ISEL(1600),JSEL(1600),x1cal(1600),
*z1cal(1600)
*,xelim(1600),zelim(1600),IESEL(1600),JESEL(1600),jfin(1600),
*ifin(1600)
*,xtape(1600),ytape(1600),xio(1600)
*,ISEL1(1600),JSEL1(1600),SSSX(1600,1600),SSSZ(1600,1600)
*,rio(1600,1600)
*,sio(1600,1600),x400(1600),z400(1600),y400(1600,1600)
*,za400(1600),da400(1600,1600),gendat(100),ISEL2(1600),
*JSEL2(1600),JSEL5(1600),ISEL5(1600),nq_eq(100)


*,npoints(1600,1600),FLE_KP(1600),FHE_KP(1600),inv_rev(100)
  dimension x_left_down(1600),z_left_down(1600),slope(1600)
 dimension y_point(1600),FLENGTHS(1600),FHEIGHTS(1600)
*,xxtop_area(1600),yytop_area(1600),ddexx_tot(1600,100),ffk1(100),
*DDEXX(1600,100), DDEXX1(1600,100),DDEXX2(1600,100),
*DDEXX3(1600,100),ef(100),eff(1),xbe_in(100),zbe_in(100),
*X_START(100),Z_START(100),dddxz1(1600,1600),
*FL_FIN(100),FH_FIN(100)
  dimension xxlim1(1600),zzlim1(1600),xelim1(1600),zelim1(1600)
      dimension x_ori(1600),z_ori(1600),f_integr(100,100)
      dimension i_save(100),j_save(100)
 dimension FLE_NO_VERT(100),FHE_NO_VERT(100),
*NOKO_LE_VERT(100)
 dimension xcal_v(1600),zcal_v(1600),ttest(5,5,5)
      dimension i_outf(1600,100), j_outf(1600,100), dddxz_outf(1600,100)
      dimension FLLENGTHS(1600),FHHEIGHTS(1600),NNOKKO(1600)
      dIMENSION ino_i(100),xcal_vf(1600,100),zcal_vf(1600,100)
 dIMENSION fmax_xx(100),fmin_xx(100)
 dIMENSION zloc1(1600,1600),xloc1(1600,1600)
 dIMENSION offsets1(200,200)

 dIMENSION xb1_mad(2000,40),SI_XX_ALL(10,10)

 dIMENSION TOT_X_PLOT(100),TOT_Z_PLOT(100)


 OPEN(1,FILE='mat3d1.DAT')
 OPEN(2,FILE='mat3d.OUT')
```

```
OPEN(11,FILE='constants.dat')
open (421,file='mat3d421.out')
open (422,file='mat3d422.out')
open(44,file='mat3d44.out')
open(33,file='mat3d33.out')
open(55,file='mat3d55.out')
     open (66,file='mat3d66.out')

     open (77,file='mytest.dat')
     open (771,file='XX_ALL.dat')
open (772,file='XX_ALL_IN.dat')
open (773,file='SI_XX_ALL.dat')
open (774,file='SI_OF.dat')

     OPEN (88,FILE='TESTX.OUT')
     OPEN (99,FILE='TESTZ.OUT')
OPEN (991,FILE='TESTZX.OUT')
     OPEN (85,FILE='myplotxz.m')
     OPEN (95,FILE='myplotz.m')
     open (961,file='spatial_x.out')
open (962,file='spatial_z.out')
open (963,file='spatial_xtan.out')
open (964,file='spatial_ztan.out')
     open (100,file='equal.out')
     OPEN (101,FILE='EQUAL1.OUT')
open (22,file='MARK.DAT')
     open (190,file='mymodel.m')
open (193,file='cut_new.dat')
open (7001,file='FX.dat')
open (7002,file='FZ.dat')
     open (192,file='model.dat')
     open (201,file='local.dat')
     open (202,file='myplot.m')
open (2021,file='myplot.dat')
     open (203,file='myplot1.m')
     OPEN (204,FILE='myplot3d.m')

     OPEN (205,FILE='TAPE.MPG')
     close (205,status='DELETE')
OPEN (205,FILE='TAPE.MPG')

     open (191,file='cut.MPG')
close (191,status='DELETE')
     open (191,file='cut.MPG')

open (300,file='submitted.dat')
open (301,file='given.dat')
     open (3300,file='submitted1.dat')
open (400, file='zz400.dat')
open (401, file='dd400.dat')
```

```
      open (402, file='gendat.dat')
open (403, file='control.dat')
open (404, file='gendat1.dat')


      open (501, file='x1cal.dat')
open (502, file='z1cal.dat')
open (503, file='xxlim.dat')
open (504, file='zzlim.dat')

      open (511, file='xxtop_area.dat')
open (512, file='zztop_area.dat')

open (513, file='ik.dat')
open (514, file='jl.dat')
      open (552, file='ISENS.dat')

open (905, file='error.dat')
open (906, file='int_input.dat')

open (907, file='xxcal.dat')
open (908, file='zzcal.dat')
open (909, file='xcal_ref.dat')
open (910, file='zcal_ref.dat')
open (69, file='aa10.dat')
      open (70, file='aa4.dat')
open (995, file='DEZ2.dat')
open (996, file='DEZ3.dat')

open (1001, file='vert.dat')

open (1002, file='I_SAVE.dat')

open (911, file='xloc.dat')
open (912, file='zloc.dat')
      open (913, file='dddxz.dat')

open (9111, file='xloc1.dat')

open (9121, file='zloc1.dat')

open (921, file='FL_FIN.dat')
open (922, file='FH_FIN.dat')
      open (923, file='NOKKO_FL.dat')
      open (924, file='NOKKO_FH.dat')

open (1010, file='LE_NO_VERT.dat')
open (1011, file='HE_NO_VERT.dat')
      open (1012, file='NO_LE_VERT.dat')
      open (1013, file='SI_ROWS_VERT.dat')
```

```
      open (1014, file='SI_COLS_VERT.dat')

      open (1020, file='MM_SI_HE.dat')
      open (1021, file='NN_SI_LE.dat')
          open (1022, file='LLENGTHS.dat')
          open (1023, file='HHEIGHTS.dat')
      open (1024, file='NNOKKO.dat')

      open (6600,file='NOS.dat')
          open (664, file='A4.dat')
      open (6610,file='A10.dat')
      open (7000,file='X_DERR.dat')
      open (7001,file='Z_DERR.dat')

      open (8000,file='ef.dat')
          open (8001,file='X_START.dat')
          open (8002,file='Z_START.dat')
      open (8003,file='offsets1.dat')

      open (9501,file='dddx.dat')
          open (9502,file='ddxx.dat')
          open (9601,file='dddz.dat')
          open (9602,file='ddzz.dat')


      open (9506,file='DX_TEM.dat')

      write(*,*) ('NEW TEST')
          write(*,*) ('Give 0 for no_debug OR 1 FOR DEBUG')
      read (*,*) IDEB
      rewind 552
      read (552,*) ISENS
          write (*,*) ISENS

          If(IDEB-1) 5506,5505,5506
 5505 pause 5506
 5506 continue

C     CHECK IF GIVEN POINTS DEVIATION EXCEEDS ACCEPTABLE LEVEL
      read (905,*) error,fval,tolf
      write (*,*) '                              '
      write (*,*)  '     !!!!! ATENTION!!!!'
          write (*,*)  '                              '
          write (*,*) 'error=           ',error,'meters'
      write (*,*) 'fval=            ',fval,'meters'
      write (*,*) 'Tolerance Allowed=',tolf,'meters'
          write (*,*) 'ISENS VALUE IS    ', ISENS

C     CHECK IF DEVIATIONS AFTER FAIRING EXHEEDS SET LIMIT
      if (error) 31,31,32
```

```
  31 continue
     write (*,*) '!!!! ERROR Tolerance on deviation exceeded'
     write (*,*) '!!!! Program "mat3d400f-1_new_mad" WILL NOT RUN'
          go to 99992

  32 continue
     if(isens-2) 321, 322,321
 322 write (*,*) '!!!! ERROR ISENS=2 Only lines fairing is requested'
     write (*,*) '!!!! Program "mat3d400f-1_new_mad" WILL NOT RUN'
     go to 99992

 321   WRITE(6,*) 'OFFSETS-CUTTING FILES AND GRAPHS DATA PREPA-
RATION'
C     end ABOVE CHECKS

C     READ DATA FROM FILE 'mat3d1.dat' for ISENS=2 OR if ISENS=0 or 1
     IF (ISENS-1) 22122,22122,22123
22122  read (1,*) kko
     kko_sav=kko
          write (100,*) kko
          do 20113 iko=1,kko
          read (1,*) xxlim(iko),zzlim(iko),xelim(iko),zelim(iko)
          write(503,65516) xxlim(iko)
     write(504,65516) zzlim(iko)
65516 format(f10.4)
     xxlim1(iko)=xxlim(iko)
     xelim1(iko)=xelim(iko)
     zzlim1(iko)=zzlim(iko)
     zelim1(iko)=zelim(iko)
     write (*,*) xxlim1(iko),zzlim1(iko),xelim1(iko),zelim1(iko)
20113 write (100,*) xxlim(iko),zzlim(iko),xelim(iko),zelim(iko)
     write(503,65516) xxlim(1)
     write(504,65516) zzlim(1)
C      end READ ADDITIONAL DATA IF A PLATE DEVELOPMENT IS WANT-
ED

     READ (1,*) n2,m2
     WRITE (*,*) N2,M2
22123      READ (1,*) N,M,nn1,mm1,nbf,ncf,ndf,n2,m2,ISENS,nopox,nopoz
     WRITE (*,*) N,M,nn1,mm1,nbf,ncf,ndf,n2,m2,ISENS,nopox,nopoz
     N_XB=N

          If(IDEB-1) 5508,5507,5508
 5507 pause 5507
 5508 continue

          WRITE (6600,*) NBF,NCF,NDF

          read (1,*) fac,scale,spax,spaz,SPAX1,SPAZ1
          write (*,*)fac,scale,spax,spaz,SPAX1,SPAZ1
```

```
    MBEG=M
 331 format (a5,2i2,a2,E15.5)
 332 format (a5,2i2,a2,e15.5)
 333 format ('                                        ')
         ij=(n-1-2+1)*(m-1+1)+(n-1+1)*(m-1-2+1)
    write (422,*) ij

    do 1101 i=2,n2-1
         do 1101 j=1,m2
    read (421,331) aaa1,k,l,bbb1,rio(i,j)
         write(422,332) aaa1,k,l,bbb1,rio(i,j)
    write (422,*) i,j
1101 continue

    do 1102 i=1,n2
    write (422,333)
         do 1102 j=2,m2-1
    read (421,331) aaa1,k,l,bbb1,sio(i,j)
         write(422,332) aaa1,k,l,bbb1,sio(i,j)
    write  (422,*) i,j
1102 continue

3301 format(4i5,7f10.4,i5)
         read (300,*) N4,M4,N14,M14,fac4,scale4,spax4,spaz4,SPAX14,SPAZ14
   *,off4,isens4
    WRITE(301,3301) N4,M4,N14,M14,fac4,scale4,spax4,spaz4,SPAX14,SPAZ14
   *,off4,isens4
    write(*,*) M2,N2,N4,M4

         If(IDEB-1) 5510,5509,5510
5509 pause 5509
5510 continue

3302 format(20f12.6)

    READ (3300,*) Z400(J),(Y400(I,J),I=1,N4)
  10 write (301,3302) Z400(J),(Y400(I,J),I=1,N4)

         do 1010 io=34,nbf,2
         iii=1+((io-32)/2)*m2
    read (1,*) a(iii,10)
         write(*,*) 'a(iii,10)', a(iii,10)
    write(6610,*)  a(iii,10)
1010 write (66,1020) iii, a(iii,10)

    rewind 773
    REWIND 771
         read (773,*) (SI_XX_ALL(1,LL),LL=1,2)
    ILIN=SI_XX_ALL(1,1)
         ICOL=SI_XX_ALL(1,2)
```

```
      write (*,*) ILIN,ICOL

      If(IDEB-1) 5522,5521,5522
 5521 pause 5508
 5522 continue

           do 8 LLL=1,ILIN
      read (771,*) (xb1_mad(LLL,L),L=1,ICOL)
      write (772,*) (xb1_mad(LLL,L),L=1,ICOL)
    8 continue
 1020 format (i3,4f10.3)

           do 1030 io=nbf+1,ncf,2
           iii=(io-(nbf+1-4))/2
           read (1,*) a(iii,4)
      write(*,*) 'a(iii,4)', a(iii,4)
      write(664,*)  a(iii,4)
 1030 write (66,1020)  iii,a(iii,4)
 1111 format (200f10.3)


      READ (1,*) (XCAL(I),I=1,nn1)
           read (1,*) (x1cal(i),i=1,nn1)
      read (1,*) (zcal(j),j=1,mm1)
           read (1,*) (z1cal(j),j=1,mm1)
C     end READ DATA FROM FILE mat3d1.dat

      do 12121 i=1,nn1
           x_ori(i)=x1cal(i)
 12121 write(501,65527) x1cal(i)

      do 12122 i=1,mm1
           z_ori(i)=z1cal(i)
 12122  write (502,65527)  z1cal(i)

      WRITE (2,400) (XCAL(I),I=1,nn1)

      write (2,500)

C     READ ADDITIONAL DATA IF A PLATE DEVELOPMENT IS WANTED
      WRITE (*,*) isens

      If(IDEB-1) 5524,5523,5524
 5523 pause 5508
 5524 continue


      IF (ISENS.EQ.2.0) GO TO 20021

           do 11112 iko=1,kko
```

```
          write(503,65516) xxlim(iko)
      write(504,65516) zzlim(iko)
      xxlim1(iko)=xxlim(iko)
      xelim1(iko)=xelim(iko)
      zzlim1(iko)=zzlim(iko)
      zelim1(iko)=zelim(iko)
      write (*,*) xxlim1(iko),zzlim1(iko),xelim1(iko),zelim1(iko)
11112 write (100,*) xxlim(iko),zzlim(iko),xelim(iko),zelim(iko)
      write(503,65516) xxlim(1)
      write(504,65516) zzlim(1)
65527 format (100f12.5)
C      end READ ADDITIONAL DATA IF A PLATE DEVELOPMENT IS WANT-
ED


C     IMPORT AND READ DATA PRODUCED BY MATLAB
          do 201 i=1,n-1
          REWIND 77

  201 WRITE (2,*) xb1_mad(i,1)

      open (601, file='points.dat')

      open (701, file='N_SI_LE.dat')
      open (702, file='M_SI_HE.dat')

          open (801, file='LENGTHS.dat')
          open (802, file='HEIGHTS.dat')

      open (851, file='LE_KEEP.dat')
          open (852, file='HE_KEEP.dat')
      open (853, file='KEEP.dat')

      READ(701,*) N_LE
      READ(702,*) M_HE
      write (*,*) 'N_LE IN DO 202 is', N_LE
          write (*,*) 'M_HE IN DO 202 is', M_HE

          rewind 853
      READ(853,*) NM_KEEP
      do 202 i=1,N_LE

      read (601,*) (npoints(j,i),j=1,M_HE)
          write (*,*) (npoints(j,i),j=1,M_HE)
  202 continue
      write (*,*) 'NM_KEEP', NM_KEEP

          rewind 851
          rewind 852
      do 2021 ikik=1,NM_KEEP
```

```
 2021 read(851,*) FLE_KP(ikik)
          read(852,*) (FHE_KP(ik),ik=1,NM_KEEP)

      Write (*,*) nn1, mm1
      READ (1021,*) NN_LE
      READ (1020,*) MM_HE

      WRITE(*,*) 'N_LE is', N_LE,'M_HE is',M_HE
          WRITE(*,*) 'NN_LE is', NN_LE,'MM_HE is',MM_HE

      If(IDEB-1) 5526,5525,5526
 5525 pause 5525
 5526 continue

      REWIND 1022
      REWIND 1023

      DO 10281 I_vert=1,NN_LE
      read(1022,*) FLLENGTHS(I_VERT)
10281 write(*,*) FLLENGTHS(I_VERT),'I_VERT IS',i_vert
      write(*,*) 'PAUSE 10281'

      If(IDEB-1) 5528,5527,5528
 5527 pause 5528
 5528 continue

      DO 10291 I_vert=1,MM_HE
      read (1023,*) FHHEIGHTS(I_VERT)
      write(*,*)  FHHEIGHTS(I_VERT),'I_VERT IS',i_vert
10291 continue
      write(*,*) 'PAUSE 10291'

      If(IDEB-1) 5530,5529,5530
 5529 pause 5529
 5530 continue

      do 203 i=1,N_LE
  203 READ(801,*) FLENGTHS(i)

      do 204 i=1,M_HE
  204 READ(802,*) FHEIGHTS(i)

C     end IMPORT AND READ DATA PRODUCED BY MATLAB

C      DATA READ-FILE npoints.dat CONTAINS ZEROES FOR POINTS OF  IN-
TEREST
C     AND POSITIVE INTEGERS FOR POINTS OF NO INTEREST
C     FLENGTHS ARE  NEW STATIONS OF INTEREST (n_le IN NUMBER)
C     FHEIGHTS ARE  NEW WATERLINES OF INTEREST (m_he IN NUMBER)
```

```
C     FIND POINTS OF PLATE PERIMETER
      write (*,*) 'NM_KEEP',NM_KEEP ,'N_LE', N_LE,'M_HE', M_HE

           KIS5=0

      do 20000 i=1,NM_KEEP
      do 20000 ik=1,N_LE
           do 20000 jl=1,M_HE

      if ( ( (FLENGTHS(ik).eq.FLE_KP(i) ).and.
     *     (FHEIGHTS(jl).eq.FHE_KP(i) ) ) )then
       GO TO 20001
      else
            go to 20000
           endif
20001 continue
      KIS5=KIS5+1
      write (*,*) 'POINTS FOUND',ik,jl,kis5

           JSEL5(KIS5)=jl
           ISEL5(KIS5)=ik
      write(513,*) ik
      write(514,*) jl

      go to 20002

20002 continue
20000 continue

      write (*,*) 'KIS5', KIS5

           If(IDEB-1) 5532,5531,5532
 5531 pause 5531
 5532 continue
C     end FIND POINTS OF PLATE PERIMETER

c        RELOAD XCAL,ZCAL,X1CAL Z1CAL (IN ORDER TO BE USED FOR
PLATE DEVELOPMENT)
      do 205 LE=1,N_LE
           X1cal(LE)=FLENGTHS(LE)
  205 continue
      WRITE (*,*) (X1cal(LE),LE=1,N_LE)

      do 206 HE=1,M_HE
           z1cal(HE)=FHEIGHTS(HE)
  206 continue
      WRITE (*,*) (Z1cal(HE),HE=1,M_HE)

      do 207 LE=1,N_LE
           Xcal(LE)=x1cal(LE)/fac
```

```
 207 continue
     WRITE (*,*) (Xcal(LE),LE=1,N_LE)
c      pause 207
           do 208 HE=1,M_HE
           zcal(HE)=z1cal(HE)/fac

 208 continue
     WRITE (*,*) (zcal(HE),HE=1,M_HE)
c      pause 208
           nn1=N_LE
     mm1=M_HE
           write (*,*) N_LE,M_HE
c          pause 209
c      end  RELOAD XCAL,ZCAL,X1CAL Z1CAL (IN ORDER TO BE USED FOR
PLATE DEVELOPMENT)

     ise=1

           do  38101 i=1,N_LE
     do  38101 j=1,M_HE
           if (npoints(j,i)) 38101,38102,38101

38102 KIS1=ISE

     JSEL1(KIS1)=J

     isel1 (kis1)=i
           ISE=ISE+1

38101 continue
     write(*,*) 'ise', ise

           if (ise.gt.1599) go to 99991

C     CALCULATION OF OFFSETS
     do 21001 ikik=1,10
           write   (*,*) A(IKIK,10)
     write (*,*) A(IKIK,4)
21001  continue

           If(IDEB-1) 5534,5533,5534
 5533 pause 5533
 5534 continue

20021  CONTINUE
     write(*,*) ISENS,'ISENS'

      If(IDEB-1) 5536,5535,5536
 5535 pause 5535
```

```
 5536 continue

     if(ISENS-1)  30121,30121,30122

30121 DO 3011 I=1,NN1
     XCAL(I)=FLLENGTHS(I)
 3011 write (*,*) xcal(I)

          DO 3012 J=1,MM1
          ZCAL(J)=FHHEIGHTS(J)
 3012   write (*,*) zcal(J)
30122 continue

     If(IDEB-1) 5538,5537,5538
 5537 pause 5537
 5538 continue

      If(IDEB-1) 5540,5539,5540
 5539 pause 5539
 5540 continue

     WRITE (2,400) (xb1_mad(I,1), I=1,N)
20023 continue
     nlim=(19+2*m)/3

     do 100 i=1,nn1
           fio=i/3.01

     L1=fio+1
     WRITE (*,*) 'L-VALUE IS  ',L1,i ,fio,nn1,mm1

     do 100 j=1,mm1
     write (55,400) ((xb1_mad(ik,L1)-xb1_mad(ik+1,L1)),ik=2,49,2)
     A23=xb1_mad(2,L1)-xb1_mad(3,L1)
     A45=(xb1_mad(4,L1)-xb1_mad(5,L1))*zcal(j)
     A67=((xb1_mad(6,L1)-xb1_mad(7,L1))*zcal(j)**2)
     A89=(xb1_mad(8,L1)-xb1_mad(9,L1))*(zcal(j)**3)
     A1011=(xb1_mad(10,L1)-xb1_mad(11,L1))*xcal(i)
     A1213=(xb1_mad(12,L1)-xb1_mad(13,L1))*xcal(i)*zcal(j)
     A1415=(xb1_mad(14,L1)-xb1_mad(15,L1))*xcal(i)*zcal(j)**2
     A1617=(xb1_mad(16,L1)-xb1_mad(17,L1))*xcal(i)*zcal(j)**3
     A1819=(xb1_mad(18,L1)-xb1_mad(19,L1))*xcal(i)**2
     A2021=(xb1_mad(20,L1)-xb1_mad(21,L1))*(xcal(i)**2)*zcal(j)
     A2223=(xb1_mad(22,L1)-xb1_mad(23,L1))*(xcal(i)**2)*(zcal(j)**2)
     A2425=(xb1_mad(24,L1)-xb1_mad(25,L1))*(xcal(i)**2)*(zcal(j)**3)
     A2627=(xb1_mad(26,L1)-xb1_mad(27,L1))*(xcal(i)**3)
     A2829=(xb1_mad(28,L1)-xb1_mad(29,L1))*(xcal(i)**3)*(zcal(j))
     A3031=(xb1_mad(30,L1)-xb1_mad(31,L1))*(xcal(i)**3)*(zcal(j)**2)
     A3233=(xb1_mad(32,L1)-xb1_mad(33,L1))*(xcal(i)**3)*(zcal(j)**3)
```

```
   ddd_test=A23+A45+A67+A89+A1011+A1213+A1415+A1617+A1819+
  *A2021+A2223+A2425+A2627+A2829+A3031+A3233

   ddd=xb1_mad(2,L1)-xb1_mad(3,L1)+
  *(xb1_mad(4,L1)-xb1_mad(5,L1))*zcal(j)+
  *((xb1_mad(6,L1)-xb1_mad(7,L1))*zcal(j)**2)+
  *(xb1_mad(8,L1)-xb1_mad(9,L1))*(zcal(j)**3)+
  *(xb1_mad(10,L1)-xb1_mad(11,L1))*xcal(i)+
  *(xb1_mad(12,L1)-xb1_mad(13,L1))*xcal(i)*zcal(j)+
  *(xb1_mad(14,L1)-xb1_mad(15,L1))*xcal(i)*zcal(j)**2+
  *(xb1_mad(16,L1)-xb1_mad(17,L1))*xcal(i)*zcal(j)**3+
  *(xb1_mad(18,L1)-xb1_mad(19,L1))*xcal(i)**2+
  *(xb1_mad(20,L1)-xb1_mad(21,L1))*(xcal(i)**2)*zcal(j)+
  *(xb1_mad(22,L1)-xb1_mad(23,L1))*(xcal(i)**2)*(zcal(j)**2)+
  *(xb1_mad(24,L1)-xb1_mad(25,L1))*(xcal(i)**2)*(zcal(j)**3)+
  *(xb1_mad(26,L1)-xb1_mad(27,L1))*(xcal(i)**3)+
  *(xb1_mad(28,L1)-xb1_mad(29,L1))*(xcal(i)**3)*(zcal(j))+
  *(xb1_mad(30,L1)-xb1_mad(31,L1))*(xcal(i)**3)*(zcal(j)**2)+
  *(xb1_mad(32,L1)-xb1_mad(33,L1))*(xcal(i)**3)*(zcal(j)**3)

       write (*,*) xcal(i),zcal(j), ddd_test,ddd
   dd1=0.

   do 80 io=34,nbf,2
   iii=1+((io-32)/2)*m2
   dtem1=xb1_mad(io,L1)-xb1_mad(io+1,L1)
   dtem2=(xcal(i)-a(iii,10))

802    format (6f10.3,2i4)

   d_plus=dtem1*(.5*(dtem2**3)+.5*dabs(dtem2**3))
   dd1=dd1+dtem1*(.5*(dtem2**3)+.5*dabs(dtem2**3))

801 format (6i4,6f8.3,2E15.5)
 80 continue

   dd2=0.
   do 81 io=nbf+1,ncf,2
   iii=(io-(nbf+1-4))/2
   dd2=dd2+(xb1_mad(io,L1)-xb1_mad(io+1,L1))*(.5*((zcal(j)-a(iii,4))
  ***3)+.5*dabs((zcal(j)-a(iii,4))**3))
 81 continue
   dd3=0.
   do 82 kk=2,n2-1
   do 82 k=2,m2-1
   io=ncf+2*((kk-2)*(m2-2)+(k-2))+1
   inx=(kk-1)*m2+1
   inz=k

   dd3=dd3+(xb1_mad(io,L1)-xb1_mad(io+1,L1))*
```

```
     *((.5*(zcal(j)-a(inz,4))**3)+dabs(.5*(zcal(j)-a(inz,4))**3))*
     *((.5*(xcal(i)-a(inx,10))**3)+dabs(.5*(xcal(i)-a(inx,10))**3))

      d_tem_1=(xb1_mad(io,L1)-xb1_mad(io+1,L1))

           d_tem=((.5*(zcal(j)-a(inz,4))**3)+dabs(.5*(zcal(j)-a(inz,4))**3))*
     *((.5*(xcal(i)-a(inx,10))**3)+dabs(.5*(xcal(i)-a(inx,10))**3))


   82 continue
      d=ddd+dd1+dd2+dd3


      DDDD(I,j)=D
  100 write (2,400) xcal(i),zcal(j),ddd,dd1,dd2,dd3 ,d

      If(IDEB-1) 5542,5541,5542
 5541 pause 5541
 5542 continue

      write (2,401) xb1_mad(1,1)
      Write (33,402)
  402 format ('CALCULATED RESULTS'/'----------------'/  W.L.    HALF
     1BREATH')
      do 900 i=1,nn1
  900    xcal(i)=xcal(i)*fac*scale
C
           do 901 j=1,mm1
  901    zcal(j)=zcal(j)*fac*scale
C
      do 902 i=1,nn1
           do 902 j=1,mm1
  902 dddd(i,j)=dddd(i,j)*fac*scale

      DO 90 I=1,nn1
      write (33,400) xcal(i)
      WRITE (33,400) (zcal(j),j=1,mm1)
      write (33,400) (dddd(i,j),j=1,mm1)

      write (301,400) xcal(i)
      WRITE (301,400) (zcal(j),j=1,mm1)
      write (301,400) (dddd(i,j),j=1,mm1)

           write (301,400) ((DDDD(I,J)-SCALE*Y400(I,J)),j=1,mm1)

   90 continue

      do 91 j=1,mm1
   91 write (301,400) zcal(j),(dddd(i,j),i=1,nn1)
```

```
      xb1_mad(1,1)=xb1_mad(1,1)*fac*scale
      WRITE (33,401) XB1_mad(1,1)
  401 format (//'    MAX DEVIATION=',F20.10)
  400 FORMAT (9F12.5)
  409 FORMAT (100F12.5)
  411 FORMAT (100F20.10)
  500 FORMAT (15H END OF INPUT'E)
      write (2,501) xb1_mad(1,1)
      do 150 i=2,n
  150 write (2,5011) xb1_mad(i,1)

      zzz1=xb1_mad(2,1)-xb1_mad(3,1)+.7*(xb1_mad(10,1)-xb1_mad(11,1))
     *+.7**2*(xb1_mad(18,1)-xb1_mad(19,1))+
     *.7**3*(xb1_mad(26,1)-xb1_mad(27,1))
      write (2,501) zzz1
  501 format (e18.10)
 5011 format (2e10.5)
      WRITE (*,*) isens
c     end CALCULATION OF OFFSETS

c       CALCULATION OF X-FIRST DERRIVATIVES

      if(ISENS-1)  40121,40121,20022

           If(IDEB-1) 5544,5543,5544
 5543 pause 5543
 5544 continue

40121 DO 4011 I=1,NN1
      XCAL(I)=FLLENGTHS(I)

 4011 CONTINUE
          DO 4012 J=1,MM1
          ZCAL(J)=FHHEIGHTS(J)

 4012 CONTINUE

      DO 1011 I=1,NN_LE
          XCAL(I)=FLLENGTHS(I)/(FAC*SCALE)
 1011 write (*,*) XCAL(I)


      If(IDEB-1) 5546,5545,5546
 5545 pause 5545
 5546 continue

          DO 1012 J=1,MM_HE
      ZCAL(J)=FHHEIGHTS(J)/(FAC*SCALE)
 1012    write (*,*) ZCAL(J)
```

```
7005 format (4F20.10,2I3,2F20.10)

4022 format ('CALCULATED RESULTS'/'----------------'/  W.L.    X-DER
   1RIVATIVE')

    do 9000 iX=1,NN_LE
9000   xcal(iX)=xcal(iX)*FAC*SCALE
       do 9010 jX=1,MM_HE
9010   zcal(jX)=zcal(jX)*FAC*SCALE

9021 FORMAT (80E20.8)
9022 format( 2i5,80E20.8)

    DO 6000 IX=1,nn1

       write (961,9021) zcal(IX), xcal(IX), (dddX(iX,jX),jX=1,mm1)
       write (963,9021) zcal(IX),xcal(ix), (ddXX(iX,jX),jX=1,mm1)
6000 continue

c     IMPORT DATA PRODUCED BY matlab "arc_lengths.m"
    rewind 9502
       do 6001 I_LI=1,NN_LE-1
    write (*,*) I_LI,NN_LE-1,MM_HE
    read(9502,*) ( ddXX(I_LI,I_CO),I_CO=1,MM_HE)

       write(*,*) ( ddXX(I_LI,I_CO),I_CO=1,MM_HE)
6001 continue

    If(IDEB-1) 5548,5547,5548
5547 pause 5547
5548 continue

c     endIMPORT DATA PRODUCED BY matlab "arc_lengths.m"
C     end CALCULATION OF FIRST X-DERRIVATIVES

c     CALCULATION OF FIRST Z-DERRIVATIVE

4024 format ('CALCULATED RESULTS'/'----------------'/  W.L.    Z-DER
   1RIVATIVE')

    do 3000 iZ=1,NN_LE
3000   xcal(iZ)=xcal(iZ)*fac*scale
       do 3001 jZ=1,MM_HE
3001   zcal(jZ)=zcal(jZ)*fac*scale

    DO 2000 IZ=1,nn1

    write (962,9021) zcal(IZ),xcal(IZ),(dddZ(iZ,jx),jx=1,mm1)
    write (964,9021) zcal(IZ),xcal(IZ),(ddZZ(iZ,jX),jX=1,mm1)
2000 continue
```

```
c     IMPORT DATA PRODUCED BY matlab "arc_lengths.m"
      rewind 9602

      do 6002 I_LI=1,NN_LE

      write (*,*) I_LI,MM_HE,NN_LE
           read(9602,*) ( ddzz(I_LI,I_CO),I_CO=1,MM_HE-1)
           write(*,*) ( ddzz(I_LI,I_CO),I_CO=1,MM_HE-1)

 6002 continue

      If(IDEB-1) 5552,5551,5552
 5551 pause 5551
 5552 continue

c     end IMPORT DATA PRODUCED BY matlab "arc_lengths.m"
C     end CALCULATION OF Z-DERRIVATIVE   [DDDZ(I,J)]   AND   Z-SECOND
DERRIVATIVE [SSSZ(I,J)]

 5005 CONTINUE
      write (*,*) NN_LE,MM_HE

      If(IDEB-1) 5554,5553,5554
 5553 pause 5553
 5554 continue

C      CALCULATE COS OF ANGLE BETWEEN COORDINATE PLANES FOR
EVERY POINT
c     CALCULATED and IMPORTED  FROM  "arc_lengths" WITH
c     PARAMETER  NAME "RAM_ARC1" exported as "dddxz.dat"
      rewind 913
      DO 40051 IX=1,NN_LE
           read (913,*) (dddxz(ix,IZ),IZ=1,MM_HE)
40051 continue

      rewind 913
           DO 50051 IX=1,NN_LE
           write (913,411) (dddxz(ix,IZ),IZ=1,MM_HE)
50051 continue
C     end CALCULATE COS OF ANGLE

20022 CONTINUE
C      PREPARE PRELIMINARY DATA FOR PLOT OF BODY PLAN AND WL
PLAN
C     IN 'MYPLOT.M' AND 'MYPLOT1.M'
      write (202,35012)
           write (203,35112)
35012 format('y=[')
35112 format('y1=[')
      write (202,35016) (zcal(j),j=1,mm1)
```

```
          write (203,35016) (xcal(i),i=1,nn1)
          write (202,35013)
          write (203,35013)
35013 format(']')
     write(202,35014)
          write (203,35114)
35014 format('x=[')
35114 format('x1=[')
C     end PREPARE DATA FOR PLOT OF BODY PLAN AND WL PLAN
C     IN 'MYPLOT.M' AND 'MYPLOT1.M'

C               REPLACE  dddd's  with  offsets1's  COMONG  FROM  matlab
(breadths8_fair8_correct)
     rewind 8003
          Do 340 IR=1,nn1
          read(8003,*) (offsets1(IR,JR),JR=1,mm1)
          write(*,*)  (offsets1(IR,JR),JR=1,mm1)

  340 continue
     DO 341 IR=1,nn1
     DO 341 JR=1,mm1
          dddd(IR,JR)=offsets1(IR,JR)
     write(*,*)  dddd(IR,JR)
  341 continue

     If(IDEB-1) 5556,5555,5556
 5555 pause 5555
 5556 continue
C             end  REPLACE  dddd's  with  offsets1's  COMONG  FROM  matlab
(breadths8_fair8_correct)

C     COMPLETE DATA IN FILES "mplot" AND mplot1"
     do 35015 IR=1,nn1
          write(202,35016) (dddd(IR,JR),JR=1,mm1)
     write(2021,35016) (dddd(IR,JR),JR=1,nn1)
35015 write (202,35018)

     do 35025 j=1,nn1
          write (203,35016) (dddd(j,jo),jo=1,mm1)
35025  write (203,35018)

35016 format(1200f25.17)
     write(202,35013)
          write (202,35017)
35017 format ('plot(x,y)'/'grid on'/'axis equal')
     write (202,35118)
35018 format(';')

     write (203,35013)
          write (203,35117)
```

```
          write (203,35118)
35117 format('plot(y1,x1)')
35118 format('grid on'/'axis equal')

35031 format ('hold on'/'load c:\naval\fair.bak8\dddd1400.dat')
35032 format ('load c:\naval\fair.bak8\zzgiven400.dat')
35033 format (29Hplot(dddd1400,zzgiven400,'*')/'axis equal'/'hold on')
35034 format('mmyplot')

     write (202,35031)
     write (202,35032)
          write (202,35033)
     write (202,35034)
C     end COMPLETE DATA IN FILES "mplot" AND mplot1"

C     PREPARE DATA FOR 3D PLOT IN 'MYPLOT3D.M'
     write (204,45012)
45012 format('z3d=[')
45112 format('x3d=[')

     write (204,45016) (zcal(j),j=1,mm1)
          write (204,45013)
          write (204,45112)

          write (204,45016) (xcal(i),i=1,nn1)
          write (204,45013)

45013 format(']')
     write(204,45014)

45014 format('y3d=[')

     do 45015 i=1,mm1
          write(204,45016) (dddd(j,i),j=1,nn1)
45015 write (204,45018)

45016 format(150f12.2)
     write(204,45013)
     write (204,45019)
          write (204,45017)
          write (204,45020)
          write (204,45021)

45017 format ('mesh(y3d,X,Z)'/'hold on'/'mesh(-y3d,X,Z)')
45019 format ('[X,Z]=meshgrid(x3d,z3d)')
45018 format(';')
45020 format (53Hxlabel('Breadth'),ylabel('Frame'),zlabel('Waterline'))
45021 format(27Htext(1.,24.,'3D BODY PLAN'))
C     end PREPARE DATA FOR 3D PLOT IN 'MYPLOT3D.M'
```

```
          do 99998 j=1,nn1

99998 write (401,45016) (dddd(j,jo),jo=1,mm1)
          do 99997 j=1,mm1
99997 write (400,45016) zcal(j)
          ep=engopen('matlab')
          go to 99992

c
99991 continue
      pause
      write (*,*) 29hIMPOSSIBLE TO PROCEED FURTHER
      write (*,*) 34hARRAYS EXCEED MAXIMUM SIZE OF 1600
      write (*,*) 51hPLEASE USE LOWER NOPOZ, NOPOX NUMBER LIMITING
VALUE
          pause
99992 continue


      DO 11011 I=1,nni

11011  XCAL(I)=XCAL(I)/(FAC*SCALE)

      write (*,*) nn1,mm1, nbf,ncf,n2,m2

      If(IDEB-1) 5558,5557,5558
 5557 pause 5557
 5558 continue

c     DEFINE INDEXES FOR REVERSING  (INV_REV(IKO)
      do  59 iko=1,kko_sav
          inv_rev(iko)=-1
   59 continue
      do  60 iko=1,kko_sav
          xel=xelim1(iko)-xxlim1(iko)
          zel=zelim1(iko)-zzlim1(iko)
          if (( xel.lt.0).and.(zel.ge.0)) inv_rev(iko)=2
          continue
      if (( xel.lt.0).and.(zel.lt.0)) inv_rev(iko)=3
      continue
      if (( xel.ge.0).and.(zel.lt.0)) inv_rev(iko)=1
   60 continue
      write (*,*) 'INV_REV', (inv_rev(iko),iko=1,kko)
c      end DEFINE INDEXES FOR REVERSING  (INV_REV(IKO)

c     FIND TANGENT AND STARTING COORDINATES OF SIGHT EDGES OF
PLATE
c     TANGENT VALUES  nq_eq(iko) 1=VERTICAL 2=HORZONTAL
c      3=INCLNED  4=ZERO LENGTH
      do  79 iko=1,kko_sav
```

```
          xb=xxlim(iko)
      zb=zzlim(iko)
          xe=xelim(iko)
      ze=zelim(iko)
c      nq_eq(iko)=0

          xbe_in(iko)=xxlim(iko)-xelim(iko)
      zbe_in(iko)=zzlim(iko)-zelim(iko)

          write (*,*) 'xbe is', xbe_in(iko),'zbe is', zbe_in(iko)

               if ((xbe_in(iko).eq.0.).and.(zbe_in(iko).ne.0.)) go to 73
          continue
      if ((xbe_in(iko).ne.0.).and.(zbe_in(iko).eq.0.)) go to 74
          continue
      if ((xbe_in(iko).ne.0.).and.(zbe_in(iko).ne.0.)) go to 76
          continue
      if ((xbe_in(iko).eq.0.).and.(zbe_in(iko).eq.0.)) go to 77


    73 continue
c     SIDE VIEW IS VERTICAL
      nq_eq(iko)=1
          ef(iko)=999999
          go to 75


    74 continue
c     SIDE VIEW IS HORIZONTAL
      nq_eq(iko)=2
          ef(iko)=0
      go to 75

    76 continue
c     SIDE VIEW IS INCLINED
      nq_eq(iko)=3
c         ef(iko)=zbe_in(iko)/xbe_in(iko)
      ef(iko)=xbe_in(iko)/zbe_in(iko)
      go to 75

    77 continue
c     SIDE VIEW IS OF ZERO LENGTH
      nq_eq(iko)=4
          ef(iko)=Nan

    75    continue
          X_START(iko)=XXLIM(iko)
          Z_START(iko)= ZZLIM(iko)
          write (*,*) 'nq_eq(iko) is ', nq_eq(iko),'X_START(iko) is ',
     *X_START(iko),'Z_START(iko) is ',Z_START(iko)
```

```
          write (*,*)'VALUE OF TANGENT IS' ,ef(iko), 'xbe is' ,xbe_in(iko)
      write (*,*)  '       '

   79 continue
c     end   FIND TANGENT AND STARTING COORDINATES OF SIGHT EDGES
OF PLATE

C         SAVE ef X_START Z_START in ef.dat( file 8000). X_START(file800!)
Z_START(file(8002)
      do 791 iko=1,kko_sav
          write(8000,*) ef(iko)
      write(8001,*) X_START(iko)
      write(8002,*) Z_START(iko)
  791 continue
C     end SAVE ef X_START Z_START

       If(IDEB-1) 5562,5561,5562
 5561 pause 5561
 5562 continue

c     START CALCULATION OF DEVELOPED PIECES
          icount=0
      INDEX_SIDE=0
      IX=0
          JZ=0


      write (*,*) kko_sav
          kko=kko_sav

      If(IDEB-1) 5564,5563,5564
 5563 pause 5563
 5564 continue

cccc    DESIDE IF SIDE EDGE IS VERTICAL OR NOT AND HANDLE
ACOORTINGLY
      do 8111 iko=1,kko
          if (xbe_in(iko).eq.0)  go to 811

            If(IDEB-1) 5566,5565,5566
 5565 pause 5565
 5566 continue
      continue

          go to 812
  811 continue

cckk     SIDE EDGE IS VERTICAL_ HANDLE
      write (*,*) 'SIDE EDGE IS VERTICAL_ HANDLE'
```

```
      If(IDEB-1) 5568,5567,5568
 5567 pause 5567
 5568 continue

     iNDEX_SIDE= INDEX_SIDE+1

ccc   DO 25006 Io=1,N2
          DO 25006 Io=1,Nn1
     JZ=0

cc        DO 25015 jo=1,m2-2
          DO 25015 Jo=1,MM1

     xc=xcal(io)
          zc=zcal(jo)

     fmax_xxlim=max(xelim1(iko),xxlim1(iko))
          fmin_xxlim=min(xxlim1(iko),xelim1(iko))
          fmax_zzlim=max(zelim1(iko),zzlim1(iko))
          fmin_zzlim=min(zzlim1(iko),zelim1(iko))


     x_max_dif=xc-fmax_xxlim
          x_min_dif=xc-fmin_xxlim
          z_max_dif=zc-fmax_zzlim
          z_min_dif=zc-fmin_zzlim

          WRITE (*,*) 'DIF',io,jo,zcal(jo),x_max_dif,z_max_dif,z_min_dif,zc
     if(    (x_max_dif.eq.0).and.
    *(z_max_dif.lt.0).and.(z_min_dif.ge.0)      ) go to 25013

     go to 25015
25013 continue

     icount=icount+1
     JZ=JZ+1
     IX=IX+1

          write (*,*) 'IX=', IX, 'JZ=', JZ,'icount is',icount
     WRITE (*,*) io,jo,iko,zloc(io,jo),xc,zc
    *,dddxz(io,jo),inv_rev(iko),ix,jz

      WRITE (1001,1001) io,jo,iko,zloc(io,jo),xcal(io),zcal(jo)
    *,dddxz(io,jo),ef(iko),inv_rev(iko),ix,Jz,nq_eq(iko)

 1001 format(3I5,f15.8,2f10.3,f15.8,f15.3,4I5,2f15.3)

     IKO_CUR=iko
```

```
          JO_SAVE=JO
          IO_SAVE=io

25015 CONTINUE
cc    END DO 25015 OF INTERNAL GROUP (JO)

25006 CONTINUE
      write (*,*) IO_SAVE,JO_SAVE
ccc       DO 25006  END OF EXTERNAL GROUP (IO)

cckk      end HANDLING VERTICAL SIDE EDGE
      go to 81111
  812 continue

81111 CONTINUE
c         end DESIDE IF SIDE EDGE IS VERTICAL OR NOT AND HANDLE
ACOORTINGLY


c     SIDE EDGE IS NOT VERTICAL_HANDLE
c     Import data from MATLAB useful  for  NON VERTICAL vew sides
      fmax_xxlim=max(xelim1(iko),xxlim1(iko))
          fmin_xxlim=min(xxlim1(iko),xelim1(iko))
          fmax_zzlim=max(zelim1(iko),zzlim1(iko))
          fmin_zzlim=min(zzlim1(iko),zelim1(iko))
          fmax_xx(iko)=fmax_xxlim
          fmin_xx(iko)=fmin_xxlim
1013  FORMAT (100I4)
10013 FORMAT (f4.2)
      rewind 1010
          rewind 1011
          rewind 1012
          rewind 1013
          rewind 1014

      rewind 1020
          rewind 1021
          rewind 1022
          rewind 1023
          rewind 1024

      read (1013,*) si_rows_VERT
      read (1014,*) si_cols_VERT
      write (*,*) 'si_cols_VERT',  si_cols_VERT

10023 format(f14.3)
      read (1021,*) NN_SI_LE
      read (1020,*) MM_SI_HE

      isi_rows_VERT=si_rows_VERT
```

```
      isi_cols_VERT=si_cols_VERT
      IN_SI_LE=NN_SI_LE
      IM_SI_HE=MM_SI_HE

C     DO 1018 I_vert=1,isi_rows_VERT
      DO 1018 I_vert=1,isi_COLS_VERT
      read(1010,*) Fle_no_vert(I_VERT)
 1018 write(*,*) fle_no_vert(I_VERT)
      write(*,*) 'PAUSE 1018'
c          end DO 1018

C          DO 1019 I_vert=1,isi_rows_VERT
      DO 1019 I_vert=1,isi_COLS_VERT
      read (1011,*) fhe_no_vert(I_VERT)
      write(*,*) fhe_no_vert(I_VERT)
 1019 continue
c          end DO 1019
      write(*,*) 'PAUSE 1019'

           If(IDEB-1) 5570,5569,5570
 5569 pause 5569
 5570 continue

C          DO 1023 I_vert=1,isi_rows_VERT
      DO 1023 I_vert=1,isi_COLS_VERT
      read (1012,*) NOKO_LE_VERT(I_VERT)
      write (*,*) NOKO_LE_VERT(I_VERT)
 1023 continue
c          end DO 1023

      write(*,*) 'PAUSE 1023'
c          pause 1023
      write (*,*) IN_SI_LE,IM_SI_HE,NN_SI_LE,MM_SI_HE

           DO 1028 I_vert=1,IN_SI_LE
      read(1022,*) FLLENGTHS(I_VERT)
 1028 write(*,*) FLLENGTHS(I_VERT)
      write(*,*) 'PAUSE 1028'
c          end DO 1028

           DO 1029 I_vert=1,IM_SI_HE
      read (1023,*) FHHEIGHTS(I_VERT)
      write(*,*)  FHHEIGHTS(I_VERT)
 1029 continue
c          end DO 1029
      write(*,*) 'PAUSE 1029'

           DO 1033 I_vert=1,IM_SI_HE
      read (1024,*) NNOKKO(I_VERT)
      write (*,*) NNOKKO(I_VERT)
```

```
 1033 continue
      write(*,*) 'PAUSE 1033'
c          end DO 1033
c     End import data from MATLAB  useful for  NON VERTICAL vew sides

      IX=IX+1
           fk1_dot=ef(iko)

                     ffk2=Z_START(iko)
      i_co=0
      write (*,*) 'isi_rows_vert',isi_rows_vert
           j=0
           NO_I=0

           do 11101 i=1,isi_rows_vert
           write (*,*) 'NOKO_LE_VERT(i) is ', NOKO_LE_VERT(i),'iko is',iko
      write (*,*) 'nn1 is',nn1
           if (NOKO_LE_VERT(i).eq.iko) GO TO 1024
      GO TO 11101
 1024  j=j+1
      xcal_v(i)=FLE_NO_VERT(i)
      zcal_v(i)=FHE_NO_VERT(i)
      if ( j.eq.1) xcal_v(1)=XCAL_v(i)

C          DO 1999 i_deg=1,nn1
C     DO 1999 j_deg=1,mm1

           DO 1999 i_deg=1,NN_SI_LE
      DO 1999 j_deg=1,MM_SI_HE
           if=0
           write (*,*) FLLENGTHS(i_deg),xcal_v(i),FHHEIGHTS(j_deg),
     * zcal_v(i),i,i_deg,j_deg,fmin_xx(iko)


      if( (FLLENGTHS(i_deg).eq.xcal_v(i)).and.
     *(FHHEIGHTS(j_deg).eq.zcal_v(i)).and.
     * (xcal_v(i).ge.fmin_xx(iko))    ) go to 1998
           go to 2001
           continue

 1998 continue
      NO_I=NO_I+1
           INO_I(NO_I)=i
      write (*,*) 'i_deg is', i_deg, 'j_deg is', j_deg,
     * ' i is', i, xcal_v(i),
     *zcal_v(i),dddxz(i_deg,j_deg),'NO_I IS', NO_I ,
     *'INO_I(NO_I)',INO_I(NO_I)
           i_out=i_deg
           J_out=j_deg
      dddxz_out=dddxz(i_deg,j_deg)
```

```
          i_co=i_co+1
c
 2001 continue

 1999 continue

      write (*,*) 'iko=', iko,' i=', i,' j=', j,xcal_v(i),zcal_v(i),fk2

      I_SAVE(i)=(i-1)+io_save
      J_SAVE(i)=(i-1)+jo_save
      write (1002,*) 'i', i , 'I_SAVE(i)', I_SAVE(i),J_SAVE(i)
          write (*,*) 'i', i , 'I_SAVE(i)', I_SAVE(i),J_SAVE(i)

      z1=fkk2
          write (*,*) ' z value is',z1,fk1_dot, fk1,xcal_v(i),'i=   ',i
      write (*,*) fk1_dot,xcal_v(1), xcal_v(i)

                    If(IDEB-1) 5572,5571,5572
 5571 pause 5571
 5572 continue

      if( ef(iko)) 21111, 21112,21111

21111  fk1=1/ef(iko)
      go to 21113
21112 fk1=0
      fk2=z1
          eff(iko)=9999999
      fk2=ffk2-(xcal_v(1)/eff(iko))

          write (*,*) 'eff(iko)',eff(iko),'fk1', fk1, 'fk2', fk2
          go to 21114

21113      fk2=ffk2-(xcal_v(1)/ef(iko))
      eff(iko)=ef(iko)

21114     z=fk1*xcal_v(i)+fk2
          write (*,*) 'xcal_v(i)',xcal_v(i), xcal_v(1),  z,FK1,FK2

                    If(IDEB-1) 5574,5573,5574
 5573 pause 5573
 5574 continue


c     FIND  GROUP OF STATIONS AND APPLICABLE xb's (xb1_mad)
      DO 2014 LIO=1,N_LE
      if ( FLENGTHS(LIO)-xcal_v(i) ) 2011,2010,2011
 2010  fio=LIO/3
      L=fio+1
          write  (*,*) i,LIO,FLENGTHS(LIO) ,xcal_v(i) ,fio,L
```

```
          go to 2014
 2011 continue
 2014 continue
c     end FIND  GROUP OF STATIONS AND APPLICABLE xb's (xb1_mad)


cccc  RECALCULATE DERRIVATIVES
          write (*,*) 'fk1 is',fk1,'fk1_dot is',fk1_dot,ef(iko),fk2
          A45=(xb1_mad(4,L)-xb1_mad(5,L))*fk1

      A67= (xb1_mad(6,L)-xb1_mad(7,L))*( 2*fk1*(fk1*xcal_v(i)+fk2)  )

      A89=(xb1_mad(8,L)-xb1_mad(9,L))*( 3*fk1*(fk1*xcal_v(i)+fk2)**2  )

      A1011=(xb1_mad(10,L)-xb1_mad(11,L))

      A1213=(xb1_mad(12,L)-xb1_mad(13,L))*((fk1*xcal_v(i)
     *     +fk2)+xcal_v(i)*fk1 )

      A1415=(xb1_mad(14,L)-xb1_mad(15,L))*
     * ( (fk1*xcal_v(i)+fk2)**2+ 2*fk1*xcal_v(i)*
     *( fk1*xcal_v(i)+fk2  ))


      A1617=(xb1_mad(16,L)-xb1_mad(17,L))*
     * ( (fk1*xcal_v(i)+fk2)**3+ 3*fk1*xcal_v(i)*
     *( fk1*xcal_v(i)+fk2  )**2)

      A1819=(xb1_mad(18,L)-xb1_mad(19,L))*2*xcal_v(i)


      A2021=(xb1_mad(20,L)-xb1_mad(21,L))*( 2*xcal_v(i)*(fk1*xcal_v(i)
     *+fk2)
     *+fk1*xcal_v(i)**2 )

      A2223=(xb1_mad(22,L)-xb1_mad(23,L))*
     *( 2*xcal_v(i)*(fk1*xcal_v(i)+fk2)**2+
     * 2*fk1*xcal_v(i)**2*(fk1*xcal_v(i)+fk2) )

      A2425=(xb1_mad(24,L)-xb1_mad(25,L))*(2*xcal_v(i)*(fk1*xcal_v(i)+
     *fk2)**3
     *+3*fk1*xcal_v(i)**2*(fk1*xcal_v(i)+fk2)**2 )

      A2627=(xb1_mad(26,L)-xb1_mad(27,L))*3*xcal_v(i)**2

      A2829=(xb1_mad(28,L)-xb1_mad(29,L))*(3*xcal_v(i)**2*(fk1*xcal_v(i
     *)+fk2)
     *+xcal_v(i)**3*fk1 )

      A3031=(xb1_mad(30,L)-xb1_mad(31,L))*(3*xcal_v(i)**2*(fk1*xcal_v(i)
```

```
     *+fk2)**2
     *+2*fk1*xcal_v(i)**3*(fk1*xcal_v(i)+fk2) )


      A3233=(xb1_mad(32,L)-xb1_mad(33,L))*(3*xcal_v(i)**2*(fk1*xcal_v(i)
     *+fk2)**3
     *+3*fk1*xcal_v(i)**3*(fk1*xcal_v(i)+fk2)**2 )

          write (*,*) 'fk1 is',fk1
      write (*,*) A45, A67, A89, A1011, A1213, A1415,
     * A1617, A1819, A2021
      write (*,*) A2223, A2425, A2627, A2829, A3031, A3233

       dEXX=A45+A67+A89+A1011+A1213+A1415+A1617+A1819+A2021+
     *A2223+A2425+A2627+A2829+A3031+A3233
      write(*,*) 'DEXX IS',dexx

c     CALCULATION OF B;s
      dEXX1=0
      do 180 io=34,nbf,2
      iii=1+((io-32)/2)*m2
      dtem1=xb1_mad(io,L)-xb1_mad(io+1,L)
      dtem2=(xcal_v(i)-a(iii,10))

          IF (DTEM2.LE.0.) GO TO 79991

       dEXX1=dEXX1+3*dtem1*(.5*(dtem2**2)+.5*dabs(dtem2**2))


          IF (DTEM2.LE.0.) GO TO 79991

79991 continue
  180 continue
      write(*,*) 'DEXX1 IS',dexx1
c     end CALCULATION OF B;s


c     CALCULATION OF C;s
      dEXX2=0
      do 15000 io=nbf+1,ncf,2
      iii=(io-(nbf+1-4))/2

      write(*,*) fk1,xcal_v(i),fk2,A(III,4)
      f_check=(fk1*xcal_v(i)+fk2)-A(III,4)
          if (f_check) 1801,1801,1802
c         IF (((fk1*xcal_v(i)+fk2)-A(III,4)).LE.0.) GO TO 15000
 1802   tem1=.5*3*fk1*(fk1*xcal_v(i)+fk2-a(iii,4)    )**2
      tem2=.5*3*dabs(fk1*(fk1*xcal_v(i)+fk2-a(iii,4)    )**2)
          go to 1803
 1801   TEM1=0
```

```
      TEM2=0

 1803 dEXX2=dEXX2+(xb1_mad(io,L)-xb1_mad(io+1,L))*(tem1+tem2)
15000 continue
      write(*,*) 'DEXX2 IS',dexx2,tem1,tem2,xb1_mad(io,L),
     *xb1_mad(io+1,L),io
c     end CALCULATION OF C;s


c     CALCULATION OF D;s
      icount=0
           zzca=fk1*(xcal_v(i)-xcal_v(1))+ffk2
      write (*,*) 'fk1=',fk1,'xcal_v(i)',xcal_v(i),'ffk2', ffk2

           dEXX3=0
           do 18002 kk=2,n2-1

      do 18002 k=2,m2-1
      io=ncf+2*((kk-2)*(m2-2)+(k-2))+1
      inx=(kk-1)*m2+1
      inz=k

      IF (((XCAL_v(I)-A(INX,10)).LE.0.).OR.((zzca-A(INZ,4)).LE.0)) THEN
      GO TO 17998
           ELSE
      DEXx3=DEXx3+(xb1_mad(io,L)-xb1_mad(io+1,L))*
     *((.5*(zzca-a(inz,4))**3)+dabs(.5*(zzca-a(inz,4))**3))*
     *((.5*3*(xcal_v(i)-a(inx,10))**2)+dabs(.5*3*
     *(xcal_v(i)-a(inx,10))**2))

      TIM1=((.5*(zzca-a(inz,4))**3)+dabs(.5*(zzca-a(inz,4))**3))
      write(*,*) 'zzca=',zzca,'a(inz,4)',a(inz,4),a(inx,10),XCAL_V(I)
      TIM2=((.5*3*(xcal_v(i)-a(inx,10))**2)
     *+dabs(.5*3*(xcal_v(i)-a(inx,10))**2))

      FTIMIO=(xb1_mad(io,L)-xb1_mad(io+1,L))
           write (*,*) 'io=', io, 'xb1_mad(io,L)=', xb1_mad(io,L),
     * 'xb1_mad(io+1,L)=', xb1_mad(io+1,L),'FTIMIO=',FTIMIO
      dexx33=dexx33+TIM1*TIM2*FTIMIO
           write(*,*) dexx3, dexx33,TIM1,TIM2,FTIMIO

      GO TO 17998

      ENDIF
17998 continue
       ICOUNT1=ICOUNT1+1
      write (44,801) i,j,kk,k,io,inz,xb1_mad(io,L),xb1_mad(io+1,L),
     *a(k1,10)
     *,a(k+1,4),xcal_v(i),zcal_v(j)
18002 continue
```

```
      write(*,*) 'DEXX3 IS',dexx3
c     end DO 18002  end CALCULATION OF D's

      write (*,*) dexx,dexx1,dexx2,dexx3,dexx+dexx1+dexx2+dexx3

c     CALCULATE DERRIVATIVES BY SUMMING ABOVE

      DDEXX(i,iko)=dEXX
      DDEXX1(i,iko)=dEXX1
      DDEXX2(i,iko)=dEXX2
          DDEXX3(i,iko)=dEXX3
          ddexx_tot(i_co,iko)=dEXX+dEXX1+dEXX2+dexx3
      i_outf(i_co,iko)=i_out
      j_outf(i_co,iko)=j_out
          dddxz_outf(i_co,iko)=dddxz_out
      xcal_vf(i_co,iko)=xcal_v(i)
      zcal_vf(i_co,iko)=zcal_v(i)


          write(*,*) 'ddexx_tot(i_co,iko)',ddexx_tot(i_co,iko),i_co,iko
          write (*,*) i_outf(i_co,iko),dddxz_outf(i_co,iko)
      write(*,*) i_co

11101 continue
C     end DO  11101 RECALCULATE DERRIVATIVES

          do 25001 i=1,i_co
          write (*,*) DDEXX(i,iko),DDEXX1(i,iko),DDEXX2(i,iko),DDEXX3(i,iko)
25001 continue

      do 25002 i=1,i_co
          write (*,*) XCAL_v(I), fk1_dot,ddexx_tot(i,iko)
25002 continue

c     CALCULATION OF INTEGRAD
      do 25010 ix_x=1,i_co

 2803 f_integr(ix_x,iko)=(1+ddexx_tot(ix_x,iko)**2+(1/eff(iko))**2)**0.5
          write (*,*) 'f_integrad is',f_integr(ix_x,iko),ix_x,iko,xcal(ix_x)
          write (*,*) 'ddexx_tot(ix_x,iko)',ddexx_tot(ix_x,iko)
          write (*,*) 'ddexx_tot(ix_x,iko)**2',ddexx_tot(ix_x,iko)**2
      write (*,*) '1/eff(iko)',1/eff(iko),'1/eff(iko)*2',(1/eff(iko))**2
          write (*,*) 'eff(iko)**2',eff(iko)**2,'ix_x',ix_x,'iko',iko
25010 continue
c     end CALCULATION OF INTEGRAD
c     end  CALCULATE DERRIVATIVES BY SUMMING ABO

c     INDEGRADE
      write (*,*) 'INTEGRADE    ','nn1=  ',nn1,'i_co',i_co,'ix_x', ix_x
```

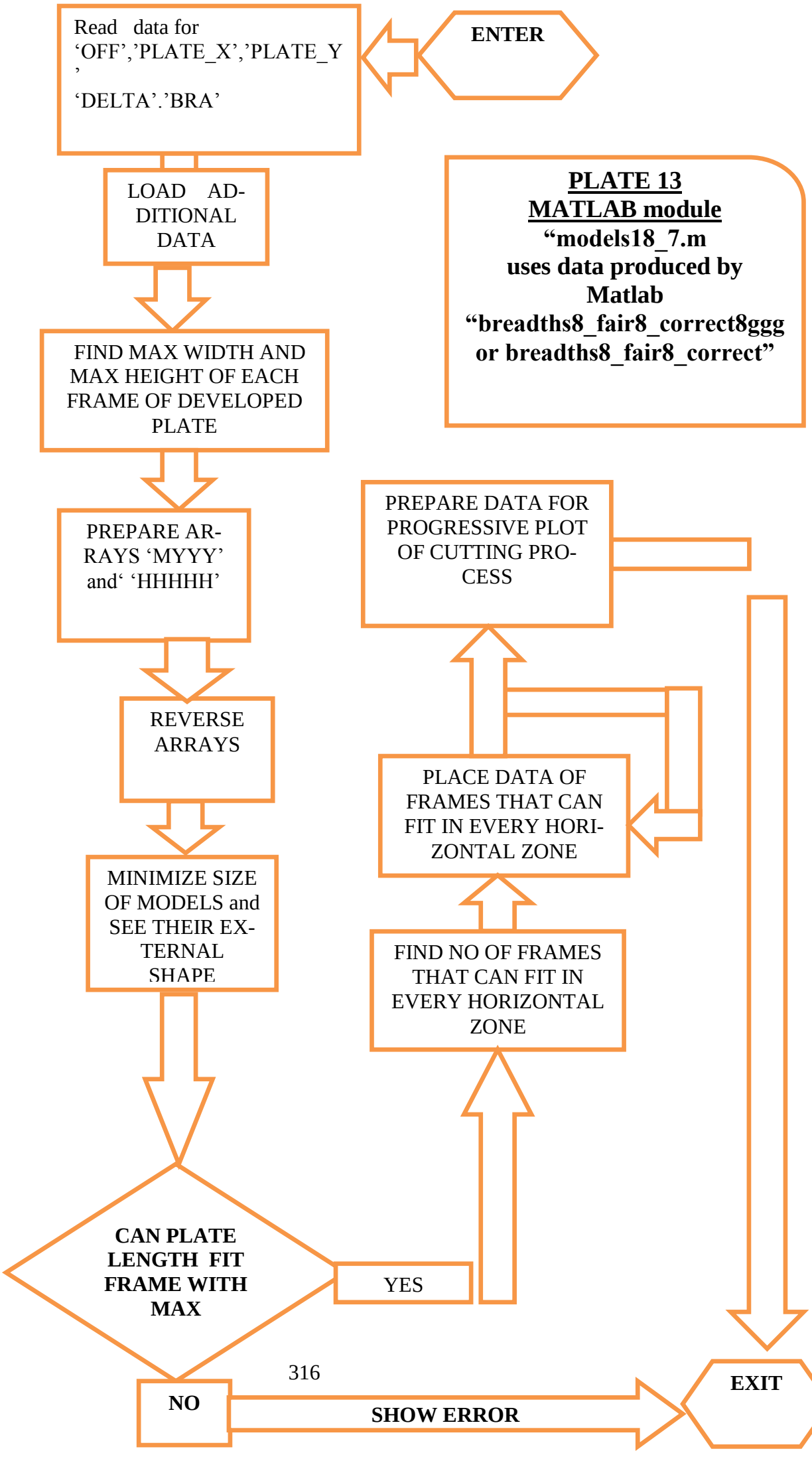
```
      IN1=0
           IN2=0
           IN3=0
      do 25020 ix_x=1,i_CO-1
      IN1=INO_I(IX_X)
      IN2=INO_I(IX_X+1)
      IN3=INO_I(IX_X+2)
      fmin_x=xcal_v(IN1)
           fmax_x=xcal_v(IN2)
           fmid_x=xcal_v(IN3)
      write (*,*) IN1,IN2,IN3
      write (*,*) fmin_x,fmax_x,fmid_x

      y0=f_integr(ix_x,iko)
           y1=f_integr(ix_x+1,iko)
           y2=f_integr(ix_x+2,iko)

           IN1=INO_I(IX_X)
      IN2=INO_I(IX_X+1)
      IN3=INO_I(IX_X+2)

      x0=xcal_v(IN1)
           x1=xcal_v(IN2)
           x2=xcal_v(IN3)

           write (*,*) 'ix_x', ix_x ,'x0=', x0, 'x1=', x1, 'x2=', x2,
     *'IN1=',IN1,'IN2=',IN2 , 'IN3=',IN3,'i_co=',i_co

           if (i_co-2) 24002, 24001,24002



24001 dif_x=abs(fmax_x-fmin_x)
           area1=dif_x*(f_integr(1,iko)+f_integr(2,iko))/2
           AR=area1
      write (*,*) dif_x,f_integr(1,iko),f_integr(2,iko),AR
           go to 24003
24002 continue

           if(ix_x-(i_co-2)) 25021,25021,25022

25021 continue


C     CUT
      write (*,*) y0,y1,y2,x0,x1,x2
      write (*,*) ix_x,IX_x+1,IX_X+2
           abeg=x0
      z0=(abeg-xcal_v(1))*fk1_dot+fk2
           do  25031 jz_z=1,mm1
```

```
          if (abs(z0-zcal(jz_z)).lt.0.0001) jz_jj=jz_z
          write (*,*) z0,zcal(jz_z),jz_jj
25031  continue

          aend=x1
          IX_XX=ix_x
          write (*,*) x0,x1,x2,y0,y1,y2,abeg,aend,z0,z1,z2
          call int(x0,x1,x2,y0,y1,y2,abeg,aend,area)


29001 AR=area
24003  continue
     write (*,*) 'area', area ,xloc(IX_xx,iko)
c         SELECT DATA FOR vert.dat ONLY FOR ANYPLATE SIGHT VIEW
BOUNDARIES

          write (*,*) 'area=', area,xloc(IX_xx,iko)
     write (*,*)  'write now FILe 1001'

          If(IDEB-1) 5576,5575,5576
 5575 pause 5575
 5576 continue

     ix_XX=ix_x

     FAC=(2*3.14159)/360
          tem=datan(ddexx_tot(ix_x,iko))/FAC

     AR=0
     WRITE (1001,1001) i_outf(IX_xx,iko),j_outf(IX_xx,iko)
    *,iko,AR,xcal_vf(ix_xx,iko),zcal_vf(ix_xx,iko)
    *,tem,ef(iko),inv_rev(iko),ix_xx,ix_xx,nq_eq(iko)


26001 continue

          go to 25020

25022 continue

     y0=f_integr(i_co-2,iko)
          y1=f_integr(i_co-1,iko)
          y2=f_integr(i_co,iko)

     IN1=INO_I(i_co-2)
     IN2=INO_I(i_co-1)
          IN3=INO_I(i_co)
     x0=xcal_v(IN1)
          x1=xcal_v(IN2)
          x2=xcal_v(IN3)
```

```
      write (*,*) 'ix_x', ix_x ,'x0=', x0, 'x1=', x1, 'x2=', x2,
     *'IN1=',IN1,'IN2=',IN2 , 'IN3=',IN3

           if (i_co-2) 23002, 23001,23002
23001  fmax_x=max(x0,x1,x2)
           fmin_x=min(x0,x1,x2)
           dif_x=fmax_x-fmin_x
           area1=dif_x*(f_integr(1,iko)+f_integr(2,iko))/2
           go to 23003
           write (*,*) 'dif_x', dif_x
23002 write (*,*)  y0 ,y1,y2,x0,x1,x2
           write (*,*) ix_x,IX_x+1,iX_X+2

      abeg=x1
           z0=(abeg-xcal_v(1))*fk1_dot+fk2
                    do  25041 jz_z=1,mm1
           if (abs(z0-zcal(jz_z)).lt.0.0001) jz_jj=jz_z
           write (*,*) z0,zcal(jz_z),jz_jj
25041  continue
           aend=x2
           IX_XX=ix_x
           write (*,*) x0,x1,x2,y0,y1,y2,abeg,aend,z0,z1,z2
           call int(x0,x1,x2,y0,y1,y2,abeg,aend,area)

C     xloc(IX_xx,iko)=area HAS NO MEANING ANY MORE
      xloc(IX_xx,iko)=0
           go to 23004

C23003 xloc(IX_xx,iko)=area1 HAS NO MEANING ANY MORE
23003 xloc(IX_xx,iko)=0
23004 continue
      write (*,*) 'area', area ,xloc(IX_xx,iko)
c         SELECT DATA FOR vert.dat ONLY FOR ANYPLATE SIGHT VIEW
BOUNDARIES

           write (*,*) 'area=', area,xloc(IX_xx,iko)
           write (*,*)  'write now FILe 1001'

      tem=atan(ddexx_tot(ix_x,iko))/FAC

      WRITE (1001,1001) i_outf(IX_xx,iko),j_outf(IX_xx,iko),
     *iko,xloc(IX_xx,iko),xcal_vf(ix_xx,iko),zcal_vf(ix_xx,iko)
     *,tem,ef(iko),inv_rev(iko),ix_xx,ix_xx,nq_eq(iko)
27001 continue
c       end SELECT DATA FOR vert.dat ONLY FOR ANYPLATE SIGHT VIEW
BOUNDARIES

25020 continue
 8111    continue
```

```
c     end INDEGRADE
c     end START CALCULATION OF DEVELOPED PIECES

  173 continue

25004 CONTINUE

      PAUSE
      PAUSE
          IF (ISENS.eq.2.) GO TO 29999
c      t20=engevalstring(ep,'test_test_test8')
29999 continue
      stop
      END
```

ANNEX(13) + PLATE(13)
DOCUMENTATION OF MATLAB PROCESS
“models18_7.m”


Read data for 'OFF','PLATE_X','PLATE_Y' 'DELTA'.'BRA'
ENTER
LOAD AD-DITIONAL DATA
PLATE 13 MATLAB module "models18_7.m uses data produced by Matlab "breadths8_fair8_correct8ggg or breadths8_fair8_correct"
FIND MAX WIDTH AND MAX HEIGHT OF EACH FRAME OF DEVELOPED PLATE
PREPARE AR-RAYS 'MYYY' and' 'HHHHH'
PREPARE DATA FOR PROGRESSIVE PLOT OF CUTTING PRO-CESS
REVERSE ARRAYS
PLACE DATA OF FRAMES THAT CAN FIT IN EVERY HORI-ZONTAL ZONE
MINIMIZE SIZE OF MODELS and SEE THEIR EX-TERNAL SHAPE
FIND NO OF FRAMES THAT CAN FIT IN EVERY HORIZONTAL ZONE
CAN PLATE LENGTH FIT FRAME WITH MAX
YES
NO
SHOW ERROR
EXIT

MATLAB MODULE “models18_7.m

```
    % 1)THIS PROGRAM APPLIES (ONLY!!!!)  TO PRODUCE DATA TO CUT MODELS OF TRANSVERSE
%FRAMES FOR DEVELOPED PART ONLY AS PRODUCED BY Matlab module "test_test_test8_mad1"

%LOAD OR MODIFY PLATE DATA
load c:\naval\fair.bak8\gendat8.dat
OFF=gendat8(1);      PLATE_X=gendat8(2);      PLATE_Y=gendat8(3);      DEL-TA=gendat8(4); BRA=gendat8(5);
['      OFF      ' ' PLATE_X      ' ' PLATE_Y      ' 'DELTA  '    'BRA']
[OFF PLATE_X PLATE_Y DELTA BRA]
%end  %LOAD OR MODIFY PLATE DATA

%SAVE  PLATE DATA
gendata8=[OFF PLATE_X PLATE_Y DELTA]
save c:\naval\fair.bak8\gendata8.dat gendata8 -ASCII -double
%end SAVE  PLATE DATA

%LOAD ADDITIONAL DATA
load c:\naval\fair.bak8\LLENGTHS.dat
load c:\naval\fair.bak8\HHEIGHTS.dat
load c:\naval\fair.bak8\myplot.dat
load c:\naval\fair.bak8\RIO1_NEW.dat
load c:\naval\fair.bak8\offsets1.dat
%end LOAD ADDITIONAL DATA

m_xcal=LLENGTHS
m_zcal=HHEIGHTS
OFF=gendat8(1)

SI_LL=size(LLENGTHS)
SI_LL=SI_LL(1,1)
SI_HH=size(HHEIGHTS)
SI_HH=SI_HH(1,1)

myplo=offsets1

IC=0; MY=[]; HH=[]; ICC=[]; HHH=[];

for I=1:SI_LL %FIND WIDTHS (MY) and HEIGHTS (HHH) CORRESPONDING TO grid of developed frames
   id=find(RIO1_NEW(I,1:end)==1)
   [I id RIO1_NEW(I,1:end)]
    SI_ID=size(id)
   SI_ID=SI_ID(1,2)
   if( SI_ID>1)  % FIND POINTS OF FRAMES OF DEVELOPED PLATE (RIO_NEW1 HAS VALUE 1)
   IC=IC+1
```

```
   MY(IC,1:SI_ID)=myplo(I,id) % CORRESPONDING BREADTHS
   [MY(IC,1:SI_ID)]

   'CHECK'
   [SI_ID id]
   HH=[]
   HH(1:SI_ID)=HHEIGHTS(id)
   HHH(IC,1:SI_ID)=  HH(1:SI_ID)      %HHH(IC,1:SI_ID)=HH(I,id)  %  CORRE-
SPONDING WATERLINES
else
end
end %end FIND WIDTHS (MY) and HEIGHTS (HHH)  CORRESPONDING TO
grid of developed frames

SI_MY=size(MY)
for J=1:SI_MY(1,2) %  PLACE NaN  IN ALL  (0,0) POINTS
for I=1:IC
   if (  (MY(I,J)==0) & (HHH(I,J)==0) )
      MY(I,J)=NaN
      HHH(I,J)=NaN
   end
end
end  %  end PLACE NaN  IN ALL  (0,0) POINTS

HHHH=HHH
MYY=MY

% fill with NaN's COLUMNS HAVING ZEROS IN ALL LINES
 IC=0
 MYYY=[]
 HHHHH=[]
 for J=1:SI_MY(1,2)
 if(  MYY(1:SI_MY,J)==0)
 else
    IC=IC+1
MYYY(1:SI_MY,J )=MYY(1:SI_MY,J)
HHHHH(1:SI_MY,J )=HHHH(1:SI_MY,J)
end
end %end fill with NaN's COLUMNS HAVING ZEROS IN ALL LINES

NEW_SI=size(MYYY) % REPLACE ZEROS WITH NaN's AT ANY PLACE

 for J=1:NEW_SI(1,2)
  for I=1:NEW_SI(1,1)
     if (MYYY(I,J)==0    )
        MYYY(I,J)=NaN
        HHHHH(I,J)=NaN
     else
     end
  end
```

```
end % end REPLACE ZEROS WITH NaN's AT ANY PLACE

HHHHH=HHHHH' % REVERSE ARRAYS
SI=size(HHHHH)
SI_LL=SI(1,1)
SI_HH=SI(1,2)

H_BOT=[]
A={}

for J=1:SI_HH
A=HHHHH(1:end,J)
id=find(A>=0)
HH_BOT_MIN(J)=min(A(id))+DELTA
HH_BOT_MAX(J)=max(A(id))-DELTA
end
MYYY=MYYY'   % end REVERSE ARRAYS

NEW_SI_CO=NEW_SI(1,1)
NEW_SI_LE=NEW_SI(1,2)

 for I=1:NEW_SI_LE
    for J=1:NEW_SI_CO
       HHHHH1(I,J)= HHHHH(I,J)- HHHHH(1,J)
    end
 end

 axis equal

 for I=1:NEW_SI_CO % SEE RESULTS IN REAL 2D SHAPE FOR EVERY STA-
TION
   %hold on
  plot( MYYY(1:end,I),HHHHH1(1:end,I))
  MYY=MYYY(1:end,I)
  HHH=HHHHH1(1:end,I)
  save c:\naval\fair.bak8\MYY.dat MYY -ASCII -double
  save c:\naval\fair.bak8\HHH.dat HHH -ASCII -double
  pause
end  % SEE RESULTS IN REAL 2D SHAPE FOR EVERY STATION
 m_dddd=MYYY
SI_D=size(m_dddd); SI_D_LE=SI_D(1,1);  SI_D_CO=SI_D(1,2)

COUNT=0

BRA=gendat8(5) % WIDTH OF   FRAME MODELS AT BOTTOM

PLX1=0; PLX2=PLATE_X; PLX3=PLATE_X;   PLX4=0; PLX5=0   % PLOT PE-
RIMETER OF PLATE
PLY1=0; PLY2=0; PLY3=PLATE_Y;  PLY4=PLATE_Y; PLY5=0
```

```
plot([PLX1,PLY1], [PLX2,PLY2], [PLX3,PLY3], [PLX4,PLY4],'*')% end PLOT
PERIMETER OF PLATE

hold on
PL(1)=0
PL(2)=PLATE_X
PL(3)=PLATE_X+PLATE_Y*sqrt(-1)
PL(4)=PLATE_Y*sqrt(-1)
PL(5)=0
plot(PL)
hold on % end PLOT PERIMETER OF PLATE AT CUT MACHINE


 MYYY_SAV=MYYY % KEEP BREATHS
 HHHHH_SAV=HHHHH % KEEP WATERLINES

 MIND=0
 CORRD=[]

 SI_ALL=size(MYYY)
 IC=SI_ALL(1,1)
 SI_HH=SI_ALL(1,2)

for I=1:SI_HH
      MIND(I)= min(MYYY(1:end,I))
 CORRD(1:IC,I)=MYYY(1:IC,I)-MIND(I)+BRA %CORRECTED OVERALL OFF-
SETS OF MODELS
end

MA=max(max(CORRD))
MAX_X=MA+OFF
    if(PLATE_X<MAX_X)
       'FURTHER PROCESS STOPS SINCE PLATE LENGTH (PLATE_X) '
       'IS SMALLER THAN MAXIMUM FRAME OFFSET'
       'THE REQUIRED MIN PLATE LENGTH IS'
       [MAX_X]
       pause
       A='ERROR'
      save c:\naval\fair.bak8\ERROR_MAD.dat A -ASCII -double
       return
    else
    end
% PREPARATION TO FIND NO OF FRAMES THAT CAN FIT IN EVERY HOR-
IZONTAL ZONE
STEPX=[]
STEPX(1)=0
  for I=2: SI_HH
STEPX(I)=OFF+max(CORRD(1:SI_LL,I))
end % STEPX(I) IS TOTAL WIDTH OF EVERY FRAME MODEL
```

```
STEPXX=[]
  for I=1: SI_HH
STEPXX(I)=OFF+max(CORRD(1:SI_LL,I))
end % STEPXX(I) IS TOTAL WIDTH OF EVERY FRAME MODEL

A=0
ACC_STEPX=[]
for I=1: SI_HH
A=A+STEPX(I)
ACC_STEPX(I)=A
end % ACC_STEPX IS THE PROGRSSIVE ACCUMULATED LENGTH OF
FRAME MODELS

C=0
ACC_STEPXX=[]
for I=1: SI_HH
C=C+STEPXX(I)
ACC_STEPXX(I)=C
end % ACC_STEPXX X IS THE PROGRSSIVE ACCUMULATED LENGTH OF
FRAME MODELS

STEPZ=[]
DIFZ=[]
STEPZ(1)=OFF
for I=1: SI_HH
   MAXZ(I)=max(HHHHH1(1:SI_LL,I))
   MINZ(I)=min(HHHHH1(1:SI_LL,I))
   DIFZ(I)=MAXZ(I)-MINZ(I)
  STEPZ(I)=OFF+ DIFZ(I) %STEPZ(I) IS TOTAL HEIGHT OF EVERY FRAME
MODEL
end


STEPZZ=[]
DIFZ=[]
for I=1: SI_HH
   MAXZ(I)=max(HHHHH1(1:SI_LL,I))
   MINZ(I)=min(HHHHH1(1:SI_LL,I))
   DIFZ(I)=MAXZ(I)-MINZ(I)
STEPZZ(I)=OFF+ DIFZ(I) %STEPZZ(I) IS TOTAL HEIGHT OF EVERY FRAME
MODEL
end

B=0
ACC_STEPZ=[]
for I=1: SI_HH
B=B+STEPZ(I)
ACC_STEPZ(I)=B
end % ACC_STEPZ IS THE PROGRSSIVE ACCUMULATED HEIGHT OF
FRAME MODELS
```

```
D=0
ACC_STEPZZ=[]
for I=1: SI_HH
D=D+STEPZZ(I)
ACC_STEPZZ(I)=D
end % ACC_STEPZZ IS THE PROGRSSIVE ACCUMULATED HEIGHT OF FRAME MODELS
% end PREPARATION TO FIND NO OF FRAMES THAT CAN FIN IN EVERY HORIZONTAL ZONE

% CHECK NUMBER OF FRAMES THAT CAN BE FITTED IN THE WIDTH OF PLATE
IM=sqrt(-1)
CUT=[]
axis equal
CUT=[]
CUT1=[]
CUT2=[]
DIF=[]
CD=[]
STE(I)=0

CUT_SAV=[]
SI_HHHHH1=size(HHHHH1)
SI_LL_SAV=SI_HHHHH1(1,2)

IC(1:SI_HH)=0
IC1=[]
IC1(1)=0
AR=[]
AR1=[]
AR2=[]

I=1
for I=1+IC1(I):SI_HH
   A=0
   I_COUNT(1)=0
   SI_IC1=size(IC1)
   SI_IC1=SI_IC1(1,2)
   if(I<=SI_IC1)

  for J=1+IC1(I):SI_HH
  A=A+STEPXX(J)
   if(A<=PLATE_X)

      I_COUNT=I_COUNT+1
      IC(I)=I_COUNT

   AR(I,J)=A
```

```
    IC1(I+1)=IC1(I)+IC(I)
 else
 end

end
else
end
end% end FIND SUCCESIVE SUMS OF STEPXX (1:SI_HH  ,2:SI_HH etc)

SI_AR=size(AR)
SI_AR_LI=SI_AR(1,1)
SI_AR_CO=SI_AR(1,2)

AR1(1:SI_AR_LI,1:SI_AR_CO)=0
for I=1:SI_AR_LI
id=find(AR(I,1:end)>0)
si_id=size(id)
si_id=si_id(1,2)
AR1(I,1:si_id)=AR(I,id)
end

ICO=0
AR3=[]
 for J=1:SI_AR_CO
if(sum(AR1(1:SI_AR_LI,J))~=0)
ICO=ICO+1
AR3(1:SI_AR_LI,ICO)=AR1(1:SI_AR_LI,J)
else
end
end

ZE=[]
SI_AR3=size(AR3)
SI_AR3_LI=SI_AR3(1,1)
SI_AR3_CO=SI_AR3(1,2)
ZE(1:SI_AR3_LI,1)=0
AR4=[ZE AR3]
 SI_AR4=size(AR4)
 SI_AR4_LI=SI_AR4(1,1) %NO OF HORIZONTAL ZONES
 SI_AR4_CO=SI_AR4(1,2) % TOTAL NO OF FRAMES (SI_HH)

 if(SI_AR4_LI>1)

 id=[]
  for I=1: SI_AR4_LI

     ido=find(AR4(I,1:end)>0)
     si=size(ido); si=si(1,2)
     id(I,1:si)=find(AR4(I,1:end)>0)
  end
```

```
 idd=id(1:end,1)-1 %idd is NO OF LONG START POINTS IN EVERY HORIZON-
TAL ZONE
else
   idd=[1:SI_HH]
end

 % START PROCESS OF MARKING AND CUTTING FRAMES
 for L=1:SI_LL_SAV
     for J=2:SI_D_LE

   DIF_Z(J-1,L)=HHHHH1(J,L)-HHHHH1(J-1,L)
   if( DIF_Z(J-1,L)>0|DIF_Z(J-1,L)<0)

  else
   DIF_Z(J-1,L)=0
  end
end
end
DIF_Z=DIF_Z'

   for I=1:SI_AR4_LI % FIND NO OF BEGINNING POINTS IN EVERY ZONE
      id=find(AR4(I,2:SI_AR4_CO)>0)
      si_id=size(id)
      si_id1=si_id(1,2)
      SI_J(I)=si_id1
   end % SI_J(I) IS NO OF BEGINNING POINTS IN EVERY ZONE (I)

   I=0
    for IK=1:SI_AR4_LI

        VERT=[]
          if (IK==1)
             VERT=0
          else

             VERT=max(imag(CUT(1:end,I)))
          end
          NO_FRS=sum(SI_J)
       for J=1:SI_J(IK)

         I=I+1
         CUT(1,I)=AR4(IK,J)+VERT*sqrt(-1);  CD(1,I)=0;

          CUT(2,I)=CUT(1,I)+OFF+.0000001i;  CD(2,I)=0;

   CUT(3,I)=CUT(2,I)+(OFF)*sqrt(-1);                    CD(3,I)=0;
   CUT(4,I)=CUT(2,I)+(OFF)*sqrt(-1);                    CD(4,I)=1;
   CUT(5,I)=CUT(3,I)+BRA;                             CD(5,I)=1

     % GENERATE MORE POINTS to the offsets side of frame
```

```
        for II=6:6+SI_D_LE-2

        II_II(I)=6+SI_D_LE-2 % UPPER CORNERS

         DIF=MYYY(II-4,I)-MYYY(II-5,I)

    CUT(II,I)=CUT(II-1,I)+DIF + DIF_Z(I,II-5)*sqrt(-1);        CD(II,I)=1;

      X_VAL=(real(CUT(II,I)))^2 % PUT ZEROS IN PERIMETER POINTS AT
Z_CAL'S HAVING ZERO VALUES
      Z_VAL=(imag(CUT(II,I)))^2
      XZ_VAL=X_VAL+Z_VAL
     if(  (XZ_VAL>=0)     )
      CUT(II,I)=CUT(II,I); CD(II,I)=1; % UPPER RIGHT CORNER OF ALL
FRAMES

      else
        CUT(II,I)=CUT(II-1,I) % UPPER CORNER OF FRAME
      end
   end

   CUT_SAV(I)= CUT(II,I)

   CUT(II+1,I)=CUT(II,I)-real(CUT(II,I))+real(CUT(3,I)); CD(II+1,I)=1;
   CUT(II+2,I)=CUT(II+1,I)-(imag(CUT(II+1,I))-imag(CUT(3,I)))*sqrt(-1);
CD(II+2,I)=1;% END DATA FOR CUTTING PERIMETER
   IO=II+2

   % START CUTTING OF HOLE
   CUT(II+3,I)=CUT(II+2,I);  CD(II+3,I)=0;

   CUT(II+4,I)=CUT(II+3,I)+OFF*sqrt(-1);    CD(II+4,I)=0;

   CUT(II+5,I)=CUT(II+4,I)+OFF;  CD(II+5,I)=0;

   III=II+9

  JJ=2

    CUT(III+JJ-5,I)=CUT(II,I)-2*OFF-OFF*sqrt(-1); CD(III+JJ-5,I)=0;
    CUT(III+JJ-4,I)=CUT(II+1,I)+OFF-OFF*sqrt(-1); CD(III+JJ-4,I)=1;
    CUT(III+JJ-3,I)=  real(CUT(III+JJ-4,I))  +imag(CUT(II+5,I))*sqrt(-1);  CD(III+JJ-
3,I)=1;

   CUT(III+JJ-2,I)=CUT(III+JJ-6+3,I);     CD(III+JJ-5+4,I)=0;
end
end

    [III+JJ-5 III+JJ-4 III+JJ-3 III+JJ-2]
```

```
    % GENERATING MORE OFFSETS OF FAIRED FRAMES
    CU1=[]
    CU2=[]
    CU3=[]
SI_CUT=size(CUT)
SI_CUT=SI_CUT(1,1)
for I=1:SI_HH
CU1(1:5,I)=CUT(1:5,I)
CU2(1:4,I)=CUT(5:8,I)
SI_CU2=size(CU2)
SI_CU2=SI_CU2(1,1)

for J=2:SI_CU2
   if CU2(J,I)-CU2(J-1,I)==0

      CU2(J,I)=CU1(end,I)
else

end
end
NO_STEPS=9
Z_POS=[]
COR_POS=[]
id=[]
MIN_Z=min(HHHHH1(1:end,I))+OFF
[I]
[HHHHH1(1:end,I)]
id=find(HHHHH1(1:end,I)>=0)

'PAUSE 452'
Z_POS=OFF+HHHHH1(id,I)
COR_POS=CORRD(id,I)
id=[]

MAX_Z=max(HHHHH1(1:end,I))+OFF

SPAN=MAX_Z-MIN_Z
STEP=SPAN/NO_STEPS
Z=[MIN_Z:STEP:MAX_Z MAX_Z]
[MIN_Z MAX_Z SPAN STEP]
[Z]
'pause 464'

SI_PO=size(Z)
SI_PO=SI_PO(1,2)
XX=[]

XX=interp1(Z_POS,COR_POS,Z,'cubic')
[Z_POS COR_POS]
[Z]
```

```
'pause 473'
[I]
   [imag(CU2(1:end,I))]
   [real(CU2(1:end,I))]
pause
plot(XX,Z)
plot(XX,Z,'^')
'pause 480'

CU22(1:SI_PO,I)=XX'+Z'*sqrt(-1)+real(CUT(5,I))-
real(CUT(5,1))+imag(CU1(end,I))*sqrt(-1)+OFF-OFF*sqrt(-1)
[CU22]
'pause 484'
%pause
CU3(1:SI_CUT-IO+2,I)=CUT(IO-1:SI_CUT,I)
end % end GENERATING MORE OFFSETS OF FAIRED FRAMES

  CUT_SMOOTH=[CU1
   CU22
   CU3]
CD3_CU=[]
SI_CU3=size(CU3)
SI_CU3=SI_CU3(1,1)

CD3_CU(1:SI_CU3,1:SI_HH)=CD(end-(SI_CU3-1):end,1:SI_HH)

SI_CU22=size(CU22)
SI_CU22=SI_CU22(1,1)
CD22_CU(1:SI_CU22,1:SI_HH)=1

CD1_CU=CD(1:5,1:SI_HH)
CD_CUT=[CD1_CU
   CD22_CU
   CD3_CU]

CUT_CUT=CUT_SMOOTH(:)

CD1_CUT=[]

  for I=1:SI_HH
   CD1_CUT=[CD1_CUT
    CD_CUT(1:end,1)]
end
CUT=[]
CUT=[CUT_CUT CD1_CUT(1:end,1)]

CUT=[real(CUT_CUT) imag(CUT_CUT) CD1_CUT(1:end,1)]

% MODIFY CUT_CUT TO ELIMINATE HOLES ON MODELS IF THEIR
BREADTHS IS TOO SMALL
```

```
hold off
I_TRUE=0
I_FALSE=0
for I_FR=1:NO_FRS
   I_FR
   pause
   if imag( (  CUT_CUT((I_FR-1)*25+17)-CUT_CUT((I_FR-1)*25+5) ))<2*OFF
       CUT_CUT( (I_FR-1)*25+22:(I_FR-1)*25+25)=CUT_CUT( (I_FR-1)*25+21);
      I_TRUE=I_TRUE+1
   else
    I_FALSE=I_FALSE+1
   end
end
%end MODIFY CUT_CUT TO ELIMINATE HOLES ON MODELS IF THEIR
BREADTHS IS TOO SMALL  hold off

 % PLOT FRAME MODELS PROGRESSIVELY
SI=size(CUT_CUT)
 for I=2:SI

    hold on
    axis equal

     if (CD_CUT(I)==0&CD_CUT(I-1)==0)

        plot(CUT_CUT(I-1:I),'r')
         plot(CUT_CUT(I-1:I),'r*')
         [I CD_CUT(I)]
        pause

    else
        hold on
        plot(CUT_CUT(I-1:I),'b')
        plot(CUT_CUT(I-1:I),'b*')
        [I CD_CUT(I)]
        pause

    end

 end
 % end PLOT FRAME MODELS PROGRESSIVELY

 % SAVE RESULTS
 RE_CUT_CUT=real(CUT_CUT)
 IM_CUT_CUT=imag(CUT_CUT)

save c:\naval\fair.bak8\TEMP\RE_CUT_CUT.dat RE_CUT_CUT -ASCII -double
save c:\naval\fair.bak8\TEMP\IM_CUT_CUT.dat IM_CUT_CUT -ASCII -double
save c:\naval\fair.bak8\TEMP\CD_CUT.dat CD_CUT -ASCII -double
```

```
save c:\naval\fair.bak8\RE_CUT_CUT.dat RE_CUT_CUT -ASCII -double
save c:\naval\fair.bak8\IM_CUT_CUT.dat IM_CUT_CUT -ASCII -double
save c:\naval\fair.bak8\CD_CUT.dat CD_CUT -ASCII -double
% end SAVE RESULTS
```

ANNEX(14)
DOCUMENTATION OF FORTRAN PROGRAM
“mat3d400f_1_tape.f”

FORTRAN PROGRAM "mat3d400f_1_tape.for"

```
      PROGRAM DEVELOP
      IMPLICIT REAL*8(A-H,O-Z)

      dIMENSION xb1_mad(2000,40),SI_XX_ALL(10,10)
          dIMENSION RE_TAB_X(50,50),RIM_TAB_X(50,50)
      dIMENSION RE_TAB_X_MO(50,50),RIM_TAB_X_MO(50,50),ID_FOR(50)
      dIMENSION TOT_X_PLOT(100),TOT_Z_PLOT(100)


      open (9601, file='RE_TAB_X.dat')
      open (9602, file='IM_TAB_X.dat')
          open (9603, file='RE_TAB_X_MO.dat')
          open (9604, file='IM_TAB_X_MO.dat')

          open (9605, file='SI_I_SS.dat')
          open (9606, file='SI_J_SS.dat')
          open (9607, file='SI_I_MO.dat')
          open (9608, file='SI_J_MO.dat')
          open (9609, file='ID_FOR.dat')

          open (9610, file='TOT_X_PLOT.dat')
          open (9611, file='TOT_Z_PLOT.dat')
      open (9612, file='SI_PLOT.dat')

      open (205, file='TAPE.mpg')

      necbeg=5
          necend=6
          markbeg=9
          markend=10
          marksta1=38
          marksta2=45
          marksta4=46
          marksta3=115
          markstop=12
          istart=30
          ISTART1=29
      ibegcut=53
      iendcut=54
      iendflame=38
      istop=63
          markfin=46
          markcheck=0
c     end LIST OF CODES FOR "esab"
          rewind 9601
          rewind 9602
          rewind 9603
```

```
          rewind 9604
          rewind 9605
          rewind 9606
          rewind 9607
          rewind 9608
          rewind 9609
          rewind 9610
          rewind 9611
      rewind 9612

c     end OPENINING OF FILES

      read(9605,*) I_I_SS
          read(9606,*) I_J_SS
          read(9607,*) I_I_MO
          read(9608,*) I_J_MO
          read(9609,*) (ID_FOR(I),I=1,I_I_SS)
          write(*,*) (ID_FOR(I),I=1,I_I_SS)

          Do 9601 IR=1,I_I_SS
          read(9601,*) (RE_TAB_X(IR,JR),JR=1,I_J_SS)
          read (9602,*) (RIM_TAB_X(IR,JR),JR=1,I_J_SS)
 9601 continue

      DO 9602 I=1,I_I_MO
          read (9603,*) (RE_TAB_X_MO(I,J),J=1,I_J_MO)
          read (9604,*) (RIM_TAB_X_MO(I,J),J=1,I_J_MO)
 9602 write(*,*) (RIM_TAB_X_MO(I,J),J=1,I_J_MO)
c     end READING OF FILES PRODUCED BY Matlab
c     MODULE  "test_test_test8_mad1'

95510 format(4i3)
95505 format (50f15.5)
  708 format (i3)
99999 format (i2)
      write(205,708) marksta1
          write(205,708) marksta2
          write(205,708)
          write(205,708) markbeg
          write(205,708) markend
c     end SUPPLY OF initial "esab"  CODES

          do 101 K=1,I_I_SS

          write(205,708)necbeg
      Write (205,95505) 1000*RE_TAB_X_MO(K,1),1000*RIM_TAB_X_MO(k,1)
      write(205,708)necend
c     end  write code and  MOVEMENT BLOCK DATA

      write(205,708)markbeg
```

```
          L_END=ID_FOR(K)
       DO 101 L=1,L_END
       Write (205,95505) 1000*RE_TAB_X(K,L), 1000*RIM_TAB_X(k,L)
       write(205,708)markend
c      end  write code and MARKING BLOCK DATA
  101    continue
c      end FIRST MAKE THE PART OF FILE FOR MARKING

c      NOW START PREPARING "esab" CODE FOR CUTTING
c      THE DEVELOPED PLATE
       write (205,708) necbeg
            write (205,708) necend
            write (205,708) markstop
            write (205,708) marksta4
            write (205,708) markcheck
            write (205,708) istart
            write (205,708) ibegcut
            rewind 9612

            read (9612,*) SI_PLOT


            DO 110 I=1,SI_PLOT
            read(9610,*) TOT_X_PLOT(I)
       READ(9611,*) TOT_Z_PLOT(I)
       write (205,95505) 1000*TOT_X_PLOT(I),1000*TOT_Z_PLOT(I)
  110 continue

       write (205,708) iendcut
            write (205,708) marksta1
            write (205,708) istop

C      end prepare file for marking and cutting "TAPE .mpg"

       stop
            end
```

ANNEX(15)
DOCUMENTATION OF MATLAB MODULE
“mynew.m”

matlab module “mynew.m”

```
%CHECK OR MODIFY ORIGINAL DATA
load x_rand_theo
load y_rand_theo
load z_rand_theo
%load x_given
%load y_given
%load z_given
xi_given=x_rand_theo
yi_given=y_rand_theo
zi_given=z_rand_theo
load Z
load X

GIVEN=[xi_given,yi_given,zi_given]
pause

%SEE PLOT OF GIVEN RANDOM DATA
s=size(xi_given)
zero=zeros(s)
plot3(x_given,z_given,zero,'o')
axis equal
hold on
pause
stem3(xi_given,zi_given,y_given,'^','fi11')

grid on

pause

%IDENTIFY TRIANGLE VERTICE
tri=delaunay3(xi_given,yi_given,zi_given)


trimesh(tri,xi_given,yi_given,zero,'EdgeColor','r','FaceColor','none')
defaultFaceColor= [0.6875 0.8750 0.8984]
hold on
trisurf(tri,xi_given,yi_given,zero,'FaceColor',defaultFaceColor,'FaceAlpha',0.9)
hold off
Y=griddata(xi_given,zi_given,yi_given,X,Z','cubic')
%tsearch(x_given,y_given,tri,X(3),Z(3))
%interp1(x_given,y_given,X,Z,'cubic')

%SEE ACCEPTABLE LIMITS
%trimesh(tri,x_given,y_given)
'SPECIFY LIMITS OF REQUIRED GRID AREA '
'FORM VECTOR OF REQUIRED LONGITUDINAL POSITIONS-VECTOR X'
%pause
```

```
'FORM VECTOR OF REQUIRED VERTICAL POSITIONS-VECTOR Z'


%MAKE CUBIC GRID WITH NAME Y
'MAKE CUBIC GRID WITH NAME Y'

'to see results with waterlines,press enter'
%pause
RESULTS=[Z', Y]
'to save results  in file RESULTS press enter'
save RESULTS
save c:\naval\fair.bak8\RESULTS.dat RESULTS -ASCII -double
save c:\naval\fair.bak8\GIVEN.dat GIVEN -ASCII -double
X=X'
save c:\naval\fair.bak8\STATIONS.dat X -ASCII -double
plot(Y,Z)
hold on
plot(Y,Z,'*')
```

ANNEX(16)
DOCUMENTATION OF MATLAB MODULE
“rs_compute.m”

Matlab module "rs_compute.m"

```
load c:\naval\fair.bak8\new_fy.dat
load c:\naval\fair.bak8\DDDD1_MAIN.dat
load c:\naval\fair.bak8\DDDD3.dat
load c:\naval\fair.bak8\FY_GIVEN.dat

load c:\naval\fair.bak8\r.dat
load c:\naval\fair.bak8\s.dat
%return
y_init=new_fy
y=new_fy;
si_y=size(y)

si_x=size(x);
si_x=si_x(1,1);
si_z=size(z);
si_z=si_z(1,1);
si_y=size(y);
si_y_li=si_y(1,1);
si_y_co=si_y(1,2);
%CALCULATION OF R's AND S's (R LONGITUDINALLy S
% VERTICALLY)
format long
r=[]
for j=1:si_z
for i=2:si_x-1

ADEN=x(i+1)-x(i-1);

A1=(y(j,i+1)-y(j,i)) / ( x(i+1)-x(i));
A2= (y(j,i)-y(j,i-1)) / ( x(i)-x(i-1));
r(i-1,j)=2*(A1-A2)/ADEN;
II=2:si_x-1

end
end
r=[ II' r]

s=[]
s_init=[]
for j=2:si_z-1
for i=1:si_x
%for j=2:si_z-1
BDEN =z(j+1)-z(j-1);
B1= (y(j+1,i)-y(j,i)) /  (z(j+1)-z(j));
B2= (y(j,i)-y(j-1,i)) /    (z(j)-z(j-1));

B1_init= (y_init(j+1,i)-y_init(j,i)) /  (z(j+1)-z(j));
```

```
B2_init= (y_init(j,i)-y_init(j-1,i)) /    (z(j)-z(j-1));

s(j-1,i)=2*(B1-B2)/BDEN;
%[i j B1 B2 B1-B2 s(i,j-1) ];
s_init(j-1,i)=2*(B1_init-B2_init)/BDEN;

end
end
s=s'

II=[1:si_x]

s=[II' s]
```