University of Alberta

FEATURE BASED CONCEPTUAL DESIGN MODELING AND OPTIMIZATION OF VARIATIONAL MECHANISMS

by

Abiy T. Wubneh

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Science

in

Engineering Design

Department of Mechanical Engineering

©Abiy T. Wubneh

Spring 2011

Edmonton, Alberta

Examining Committee

Prof. Chongqing Ru, Chair and examiner
Prof. Ming Lu, external examiner, Civil & Environmental Engineering
Prof. Gary Faulkner, examiner, Mechanical Engineering
Dr. Yongsheng Ma, supervisor, Mechanical Engineering

Abstract

This research investigates and proposes methods to be used for the automation of the conceptual design phases of variational mechanisms. It employs the concept of feature-based modeling approaches. A method is proposed for integrating the dimensional synthesis, mechanical design and CAD generation phases with minimal designer intervention. Extended feature definitions are used in this research to create a smooth data transfer platform between different engineering tools and applications.

This paper also introduces another method by which a set of dimensional data collected from a family of existing products is used to predict possible solutions for a new design. This method, based on artificial neural networks for training and solution generation, is used with optimization algorithms for the dimensional synthesis of mechanisms.

An excavator arm mechanism is used as a case study to demonstrate these methods. The design of this mechanism is carried out based on its digging mode configurations.

Acknowledgement

My sincere gratitude goes to my supervisor, Dr. Yongsheng Ma, for his advises and encouragement during the course of this research.

I also would like to thank a very special and strong person, my Mom, Emodi, for being the best mother one could ever ask for.

Contents

List of Figures

List of Tables

List of Acronyms

2D	2 dimensional
3D	3 dimensional
ANN	Artificial Neural Network
API	Application Programming Interface
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CSYS	Coordinate System
DOF	Degree of Freedom
FBD	Free body diagram
FEA	Finite elements analysis
GUI	Graphical user interface
JTMS	Justification based Truth Maintenance System
KA	Kinematic Analysis
KBE	Knowledge base engineering
MAD	Mass-acceleration diagram
MDA	Minimum deviation area
MOO	Multiple objective optimization
PS	Product Specification
RE	Reverse Engineering
SOO	Single objective optimization
STEP	Standardized exchange for product

List of Symbols

β'	Angular measurement parameter
A	Area
A_{dx}	Area direct stress
A_{tor}	Area torsional
x	Axis x
y	Axis y
σ_{ZX}	Bending stress
β	Boom deflection angle
α_2	Boom lower angular displacement limit
RB_1	Boom rotation matrix #1
RB_2	Boom rotation matrix #2
b	Boom side length
α_1	Boom upper angular displacement limit
α_{bu}	Bucket angle
h	Cross-sectional height
b	Cross-sectional base dimension
t	Cross-sectional plate thickness
dA	Differential area
dig1	Digging configuration angle #1
dig2	Digging configuration angle #2
σ_{dx}	Direct stress
Q	First moment of area about neutral axis
F_x	Force along x-axis
F_y	Force along y-axis
F_z	Force along z-axis
l_1	Hinge to hinge boom length
l_3	Hinge to tip bucket length
A	Homogeneous transformation matrix
H	horizontal excavator arm lengths
J1	Joint 1
J10	Joint 10
J11	Joint 11
J2	Joint 2
J3	Joint 3
J4	Joint 4
J5	Joint 5
J6	Joint 6
J7	Joint 7
J8	Joint 8
J9	Joint 9
h_J10	Joint J10 extension distance

h_J11	Joint J100 extension distance
h_J2	Joint J2 extension distance
h_J8	Joint J8 extension distance
c	Linear measurement parameter
S_3	Maximum Cutting Height
S_7	Maximum Depth Cut at Level Bottom
S_2	Maximum Digging Depth
S_1	Maximum Reach at Ground Level
S_4	Maximum Loading Height
S_6	Maximum Vertical Wall Digging Depth
S_5	Minimum Loading Height
M_x	Moment about x-axis
M_y	Moment about y-axis
M_Z	Moment about z-axis
d_p	pin diameter
l_p	pin length
R	Rotation transformation matrix
SD	Scope Display output
I	Second moment of area
V	Shear force
l_2	Stick length
RS	Stick rotation matrix
$< JS_3$	Stick-Joint J3 interior angle
$< JS_2$	Stick-Joint J2 interior angle
$J2_L$	Stick-Joint J2 left interior angle
$J2_r$	Stick-Joint J2 right interior angle
$J3l$	Stick-Joint J3 lower interior angle
$J2u$	Stick-Joint J3 upper interior angle
$<JS9$	Stick-Joint J9 interior angle
$J9l$	Stick-Joint J9 lower interior angle
$J9u$	Stick-Joint J9 upper interior angle
τ_{tor}	Torsional shear stress
b_2	Transition four-bar coupler link
b_1	Transition four-bar follower link
b_0	Transition four-bar stationary link
b_3	Transition four-bar driver link
T	Translation transformation matrix
τ_b	Transverse shear stress
σ_u	Ultimate stress
V	Vertical excavator arm lengths
σ_y	Yield stress

Chapter 1

Introduction

1.1 Background

The design process of multi-component products which are subject to frequent changes and modification is a very complex process due to the large amount of data involved. Dimensions and parameters defined at the initial stages of the design process are used by latter stages during manufacturing. In the traditional design approach, this set of information is usually lost between the design stages due to the fact that the reusability of knowledge is not given due emphasis.

This complication even gets worse when different parts of the product are designed by different people located in geographically different locations. Changes and modifications evoked by one department take considerable time and other resources before being fully reflected on components being developed by the other departments. Constraint definition and management is also one area affected by the method of data management system adopted in the design process.

Since the traditional CAD files, which are merely the collection of geometric entities, cannot grasp additional information vital to the manufacturing and other aspects of the product development process, they failed to lend themselves for the implementation of an effective knowledge-driven design procedures. This has forced researches to look into possible ways of appending more information to the traditional CAD models.

Features were introduced as a means to address these needs. Features are basically data structures containing a range of information required to fully describe the shape and related aspects of the product. Some of the most commonly used data elements include model geometries, material, manufacturing methods, tolerances, and machining procedures. Recently, more complicated and sophisticated features have been defined to cover previously overlooked but critical aspects.

The research is motivated by the advancement in the areas of feature definitions and their potential applications in the areas of intelligent design automation and integration. It is devoted to extending the use of feature-based modeling concepts to include design intents and constraints.

1.2 Statement of the Problem

Traditional design systems, including CAD and CAE tools, have very limited capability in terms of storing rich-information with data format that can be accessed by the different phases of the product development cycles. This limitation directly affects the choices of the design methodologies and the necessary data communication mechanisms.

The required information by the different stages of the design process has to be transferred in a very effective manner to achieve maximum level of automation and efficiency. The most commonly employed method is storing the data in a computer accessed by these process stages. This method has been working satisfactorily for quite a long time. However, as the trend in the global market gets more competitive and the market span of products gets shorter and shorter, this method had been found to be less reliable and reusable.

Attention is now shifted to developing a method by which the pertinent design information will be stored in a consistent and reusable data structure that is integrated with the product's CAD model. Thus, the previously fragmented design data files and those corresponding product CAD models have to be encapsulated into a single product data model. In fact, modern product models are no longer merely a collection of CAD geometric entities. Customizable and advanced semantic definitions called features have long been introduced enabling the integration of CAD and other engineering data sets, such as manufacturing data.

The conceptual design process of variational mechanism, which is the focus of this research, is well known for its several fragmented modules running on some sets of basic data. Design process and knowledge development is iterative in nature. Each design cycle generates a set of data or a data model to be used in the creation of the next phase model of the product being designed. For example, since the product will have different inertia properties after its 3D embodiment, the next design cycle will have to include the new properties in both physical and geometrical senses, into consideration before commencing the next new cycle. Effective automation of this process requires the systematic creation of the CAD models in a very consistent manner.

Associative relationships between different features of a CAD model have to be constrained systematically to ensure that the final generated model can have comprehensive physical as well as engineering descriptions.

In addition, in latter manufacturing stages the CAD and design information will be needed by the manufacturing system to produce the product. A computer numerical control (CNC) machining operation, for example, requires both the material type (non-geometric property) and the CAD geometric entities for tool selection and tool path generation, respectively.

A design procedure equipped with methods which can satisfy these needs and requirements, most probably, earns itself a place in the advancing industrial application.

Unlike forward kinematics problems, designs of this kind, whose end effector is passing through a prescribed path or set of precision points, inherit the common challenges observed in inverse kinematics problems. The availability of numerous solutions and, in most cases, the lack of solution convergence is some of the most prominent problems making its implementation in automated design environment very difficult.

1.3 Objectives

The top objective of this research is to propose a feature-based conceptual design automation scheme specific to variational mechanism products. This method will attempt to use the capabilities of features in accommodating for both geometrical and non-geometrical types of data into a CAD system. It aims to utilize features to bridge the necessary automation gap in the conceptual design cycle and to further investigate their applicability in terms of embedding conceptual design rules for complex part shapes and structures development.

It is also the objective of this research to propose an additional hybrid dimensional synthesis method based on Artificial Neural Networks (ANN) and optimization techniques. The objective in this area is to overcome the challenges of the dimensional synthesis process in terms of narrowing down the number of available solutions for a given problem.

The applicability of the proposed methods will be demonstrated suing the design procedures of an excavator mechanism as a case study. Other than its typical product development processes, this problem poses some challenges in terms of product configuration, linkage optimization, and its programming implementation in the proposed design method application.

The above two proposed methods will be used together in the same case-study design problem to investigate and measure their performances.

1.4 Scope of the Study

This research is proposing a conceptual product design automation method with the integration of feature-based CAD model generation via APIs C++ and Matlab programming tools.

A hybrid optimization- ANN method is proposed for dimensional synthesis of variational mechanisms. The requirement of defining a high-quality initial solution for optimization search algorithms is addressed by the use of artificial neural networks. The ANN, trained with existing and generated product data, will be used to suggest informed sets of initial solutions for the optimization techniques. Based on these initial solutions, the optimization search algorithms will be used to determine the final solutions for the inverse kinematics problems.

This work also includes the basic optimization design calculations of an excavator arm mechanism for the purpose of demonstrating the proposed methods. Only the digging operational conditions are considered for the design purpose. The design has been carried out only taking the strength requirements (working stresses) into consideration in a numerical approach (design for strength). It will not consider deformation and other dynamic considerations. The Finite Elements Analysis (FEA) will not be included in this work due to the tight time constraint.

1.5 Organization of the Thesis

Relevant concepts and theories in the areas of design automation and feature-based modeling which are previously investigated by other researchers are first reviewed in Chapter 2.

Chapter 3 discusses the proposed approaches. The first section of this chapter covers a topic on the overall design automation and design data communication architectures. The second section, on the other hand, discusses on the details of the hybrid Artificial Neural Network (ANN)-Optimization technique developed for the purpose of dimensional synthesis of variational mechanisms.

The next chapter, Chapter 4, is devoted to the theoretical design calculations of the case study. Starting from user specification inputs, and by assuming known values for joint forces, the cross-sectional parameters of the boom and sticks of an excavator arm mechanism are determined by the use of optimization techniques.

Chapter 5 focuses on the discussion of tools and procedures used to carry out the kinematic and dynamic simulations of the excavator arm mechanism. Modeling procedures and justifications for selecting the digging operation for simulation and design will be discussed in details in this chapter.

Procedures and methods employed for the CAD modeling and programming using the application user interface (API) are discussed in Chapter 6 followed by conclusions and future work recommendations in chapter 7.

Chapter 2

Literature Review

In this chapter, research works and publications carried out by other scholars are reviewed in relevance to the objective of this research. The overall organization of this section is targeted to cover the following major topics:

- Dimensional synthesis of mechanisms and manipulators
- Application of ANN in mechanism dimensional synthesis
- Design automation and integration
- Parametric and feature-based CAD modeling
- Reverse engineering and knowledge fusion in product development

2.1. Dimensional Synthesis

Dimensional synthesis is the first stage in the process of designing mechanisms and manipulators. This process is mainly focused on determining the linear joint-to-joint distances of linkages and members. Laribi et al. [14] discussed an optimization technique developed to determine the linkage dimensions of a DELTA parallel robot for a prescribed workspace. The method uses a Genetic Algorithm to minimize an objective function developed by writing expressions for the end effector location based on a concept called *the power of the point*. In their work, the dimensions of the robots were calculated using an optimization technique which minimizes a volume created by three intersecting surfaces but containing the prescribed cubic workspace. A penalty method is used in their approach to screen out and select feasible solutions from available solution domains.

Using a similar philosophy, but this time with cylindrical prescribed volume, an optimization based dimensional synthesis procedure was suggested by Zhao et al. [43] to determine optimum dimensional parameters for the design of a 2-UPS-PU parallel manipulator. Cylindrical coordinate system was used when formulating the kinematic relationships including the forward and inverse kinematics of the manipulator together with the Jacobian for force and velocity analysis. The prescribed workspace was represented by a cylinder contained inside the minimum workspace volume enclosed by the manipulator movement boundary surfaces.

Analytical and optimizations techniques have been used by several scholars for the purpose of synthesizing manipulator and mechanism dimensions. The multiple numbers of possible solutions was pointed out by several researchers as the primary disadvantage of analytical solution methods. The procedure of synthesizing the linkage dimensions of a four-bar spherical linkage mechanism, proposed by Alizade et al. [1], employed the method of polynomial approximation to transform 5 non-linear equations into 15 linear equations to solve for 5 design parameters.

The objective of their study was to determine the dimensions of a spherical four-bar linkage mechanism by linearization of a set of non-linear equations. The requirement for the synthesized mechanism was that it should be able to trace 5 precision points in space. The minimum deviation area (MDA) was proposed in their work as a constraint criterion to select the most appropriate solution. The result of their investigation was tested by plotting the path of the mechanism against the prescribed precision points using AutoCAD 2000.

2.2. Artificial Neural Networks (ANN)

Artificial neural network procedures were used by Hassan and his colleagues [12] to study the relationship between the joint variables and the position and orientation of the end effector of a six-DOF robot. The study was motivated by the fact that the use of ANN doesn’t require an explicit knowledge of the physics behind the mechanism. The network was trained by the use of real time data collected by sensors mounted on the robot. Designed with an input layer with 6 neurons for three Cartesian location coordinates and three linear velocity components, the network was used to establish a mapping pattern between the inputs and outputs. Their work mainly focused on finding the kinematic Jacobian solutions. Other numerical methods are also discussed by other scholars for solving synthesis and simulation related problems [2].

Problems and shortcoming associated with using ANN were also discussed in their paper. Some of the challenges they discussed include the difficulty of selecting the appropriate network architecture, activation functions, and bias weights. The other problem discussed in the paper is the difficulty and impracticality of collecting a large amount of data for the purpose of training the neural network.

The advantage of using ANN is also highlighted in their work. The fact that this method does not require any detailed knowledge of the mathematic and engineering knowledge involved makes it best suited to a wide range of similar applications. It was pointed out that as long as there is a sufficient number of data for training purposes, the ANN can be used to predict the Jacobian kinematics of other configurations without the need to learn and understand the explicit robotics philosophies. Modifications and changes in existing robot structures can always be addressed by training the ANN with a new set of data reflecting the new modifications.

Another research was carried out by Gao et al. [11] on the areas of application of ANN to dimensional syntheses of mechanisms. Discussed in their work was implementation of generic algorithms and neural networks as a tool to synthesize the dimensions of a six DOF parallel manipulator. They decided to use this method because of the fact that traditional optimizations techniques lack the highly needed convergence property in their solutions when used for handling a larger number of geometric variables. The stiffness and dexterity of the manipulator were taken to be the optimization criteria and they were derived based on kinematic analysis procedures. Levenberg–Marquardt and standard back propagation algorithms were used in the neural network to approximate stiffness and dexterity analytical solutions. Due to the large numbers of variables included in the analysis, they have used two different approaches for the optimizations; Single Objective Optimizations (SOO) and Multiple Objective Optimizations (MOO), namely. With the first approach, the two objectives, stiffness and dexterity, were separately investigated while in the second approach they were investigated together to understand their combined effect. Problems associated with the implementation of their techniques were addressed in their work. Modeling the objective function was one area discussed as a challenge in their work. The other is the convergence difficulty arising due to the involvement of large number of parameters in the formulations of the objective functions specially when using the MOO optimization.

Kinematic synthesis of redundant serial manipulators has been the focus of research. Singla et al. used augmented Lagrangian optimization technique to determine optimum dimensions for a redundant serial manipulator [28]. The algorithm was used for its robustness in identifying feasible solution ranges effectively. The formulation of the problem in their paper was based on minimization of the positional error subjected to the constraints of avoiding manipulator collisions with either external obstacles or its own links.

The work of Jensen and Hansen [13] discusses a method by which dimensional synthesis for both planar and spatial mechanisms are accomplished taking the problem of non-assembly into consideration. The method makes use of a gradient based optimization algorithm. Analytic calculation of sensitivities was done by direct differentiation. The problem was mathematically formulated as a standard optimization problem with inequality to take the non-assembly nature of the problem into account. Newton-Raphson method, due to its rapid convergence property, is used in the minimization of the kinematic constraints. Saddle point and steepest decent methods are used to verify the direction of convergence and stability of the minimization method, respectively.

By representing planar displacements with planar quaternion, Wu et al. [40] formulated the kinematic constraints of closed chain mechanism as a mapping from Cartesian space to a higher dimensional projective space called *image space*. It was pointed out in their work that the use of this method enabled one to reduce the problem of dimensional synthesis into determining algebraic parameters defining the image spaces. Computational simplification was achieved by transforming kinematic equality constraints into geometric constraints. Dealing with geometric parameters of the constraint manifold instead of the mechanism parameters provides ease and flexibility due to the decoupled nature of the relationships.

Procedures and methods to be used to overcome problems arising due to joint clearances have been proposed by Erkaya and Uzmay [9]. A dimensional synthesis of four-bar mechanism was discussed as a case study in their work to demonstrate their proposed method. The clearances were represented by high stiffness and weightless links to make them suitable to be studied under rigid motion considerations but without affecting the overall inertial property of the mechanism. The clearances and the mechanism were characterized and optimized using neural networks and genetic algorithms with the path and transmission angle errors used as the components of the objective function.

To address the problems of convergence uncertainties and limitations on maximum number of precision points of problems solved using optimization and analytical techniques, Vasiliu and Yannou proposed in their work [34] the use of artificial neural networks (ANN).

The ANN designed to be used for the synthesis application takes in the prescribed path and motion as an input and gives out the linkage parameters as an output. The need for large number of data for training purpose is addressed by simulation of the path for a given set of linkage parameters. The ANN was trained using the simulated data in the reverse direction, i. e., for given information on the path prescription; the mechanism parameters were to be determined. It was pointed out in their work that the absence of continuity between different morphologies prohibited and discouraged the use of interpolation techniques.

The other important point discussed in their work is the fact that neural networks perform well only in the data range they were trained with. Normalization of parameters during the utilization phase of the network is needed to bring the input values to within the known range of the training set.

Some researchers were more interested in simulation and analyzing spatial configuration performances of manipulators. These works were motivated by the need to understand the manipulators' performances under some environmental constraints. Frimpong and Li [10] modeled and simulated a hydraulic shovel to investigate its kinematics and spatial configurations when deployed in constrained mining environments. Denavit-Hartenberg homogeneous coordinate transformation techniques were used to translate the relative orientations and configurations of links into other reference frames within the overall assembly. Forward kinematics of the machine was investigated as a five-linkage manipulator. After formulating the kinematic equations, the manipulator was modeled in 3D and was simulated using the MSC ADAMS simulation software for selected time steps.

2.3. Parametric and Feature-based CAD Modeling

Several methods and procedures have been developed and used to automate and increase the efficiency of CAD modeling processes. The depth of data embedded on the CAD modes greatly depends on the specific technique employed to carry out the process [33]. Parametric modeling, among several others, has become one of the most rapidly growing and commonly adopted methods of product modeling in the manufacturing industries. Modifying the *Standardized Exchange of Product* (STEP) format, which contains only geometric information, to accommodate for additional part-specific parameterized information has been the focus of some research [25].

This process takes the traditional CAD geometry modeling method a step further by enforcing permanent geometric constraints among members of CAD objects and features. This system has also its own known limitations in terms of validity in change and modification implementations. Basak and Gulesin, in their study [3], suggested and demonstrated a method in which parametric modeling was used in coordination with feature based and knowledge fusion approaches to increase its robustness in the areas constraint validation. Standard feature libraries were also used in their method to investigate the practicality of proposed part manufacturing techniques by the designers.

Programming through the application programming interfaces (API) of existing commercial CAD packages provides designer with more flexibility to embed the design intent into the CAD models [18]. In their approach, Myung and Han [22] took design unit and functionality features into consideration for the configuration and CAD modeling of selected products. The work of Wang et al. [38] proposed a modeling environment integration method by which interactive parametric design modification was demonstrated using a CAD-linked virtual environment.

The success of parametric modeling greatly depends on the consistency and preservation of topology of CAD features used in the creation of the part being modeled. The parent-child relationships defined have to be validated at all times in order to apply this method in the design process of customizable products. The use of explicit relationship was suggested by Van et al. [33] to increase user control and add sophistication to the modeling process.

Even though parametric modeling techniques are widely used in today’s design and manufacturing industries for facilitating CAD modeling processes, their power had been merely limited to the geometric attributes. Incorporation of additional sets of information such as product material, method of manufacturing, assembly procedures, and design intents to the CAD models have been the focus of several recent research works.

Features, which are basically data structures, have been used to attach the additional information to the CAD models. The type of information ranges from purely geometric to non-geometric parameters. Traditional features used to represent only those attributes related to the geometry of the part. Recently, new types of feature definitions [30] have been introduced to embed other non-geometric aspects of the product/part being designed with the CAD models.

The employment of parametric and feature-based modeling techniques has been proven to contribute a significant role in the implementation of an integrated and automated product design procedure [39]. The interest of manufacturers in reducing the time-to-market and costs associated with the design process had motivated and initiated researches [41] to investigate features in greater depth. The power of feature-based modeling methods was coupled with the concepts of reverse engineering techniques [41] to embed design intents and other constraint into existing product’s retrieved by CAD scanning techniques (reverse engineering). By doing so, manufacturers will be able to reduce the time required to re-fabricate a given existing product with different material and modified design constraints.

The data structures of features can handle more than one type of information. As discussed earlier, information pertinent to product development such as conceptual design intents, geometric constraints, non-geometric parameters, and manufacturing and assembly procedures can be embedded into the CAD model of the product by manipulating its feature (data structure). The extent to which this information can be exploited mainly depends on the feature definition and the level of organization and communication architectures of the network [39]. Ter et al. [30] discussed this issue in their work and proposed a high level abstract unified feature-based approach by categorization and generalization of conceptual data.

The traditional definition of feature, which used to be used to merely describe the shapes and geometries of the CAD models, has been extended to cover assembly design features and other various aspects vital from the point of view of manufacturing and concurrent engineering [5]. Associative relationships, both geometric and non-geometric, between various parameters of two or more members of an assembly were discussed by Ma et al. [18]. This ability opens the door for design automation of frequently updated and modified products. Design customization of products can be benefited from the inclusion of design intents, constraints, and assembly hierarchy data [4] on the CAD files. Incorporating rules and constraints in the CAD files in the form of features requires the definition of a new set of features. By treating a feature more like a data structure than a geometric parameter description, associative relationships between parts that have not been considered before were defined. In addition, the feature definition was extended to cover information pertinent to component mating conditions and interfaces within an assembly.

2.4. Design Automation and Integration

The implementation of collaborative product development process requires a large amount of data to be transferred between applications used by different designers working toward a single product [23]. Change and modifications propagate both ways in these routes. Ma and his colleagues defined a data structure (feature) called *operation* in an effort to address the need to communicate data in feature level [19]. Associative fine-grain product repository mechanism with four-layer information scheme was proposed and demonstrated by the team for this purpose. The method was proposed taking into consideration the possibility of working on different applications and platform due to the multi-discipline nature of product design process.

Features, which have a higher level of semantic organization than the elementary geometric aspect of a product, are currently being used to create the link and bridging the gap in terms of the amount and detail of information needed to be shared by CAD and CAM systems [21,27]. Concurrent and collaborative engineering oriented product development processes require the implementation of an effective change propagation and constraint management mechanism to handle the flow of data between various development stages. In their work, Ma et al. [20] proposed a unified feature approach method for constraint management and change propagation to be used in a collaborative and concurrent environment. The developed algorithm uses the JTMS-based dependency network. The data model was categorized under constraint-based associatively and share entities association. Lourenco et al. [17], in a slightly different approach, investigated a method of interactive manipulation of feature parameters. They proposed a solver-driven algorithm for optimization of geometric constraints using non-application specific constraint solvers.

Excavator arm mechanisms have been investigated from different research point of views. Solazzi discusses [29] with his work the advantages and quantitative analysis of performance improvements achieved by redesigning an existing hydraulic arm mechanism with a different material. Yoon and Manurung [42], on the other hand, investigated the development of a control system by which the operations of an excavator arm mechanism are controlled by the operator's arms movement. Sensors attached at the different joint locations of an operator arm are used to map the arm joint displacements into the mechanism's motion.

The development of new design automation procedures [26,31] together with existing mechanical simulation tools such as SimMechanics of MATLAB® and MSc ADAMS® have given researchers the edge to fully impose explicit constraints when investigating mechanisms and manipulator's kinematic and dynamic responses [35]. The forward and inverse kinematics analyses involved in the design of mechanisms and manipulators are benefitted from the implementation of parametric and feature-based modeling approaches [35]. Work space configurations of manipulators and their dynamic responses require frequent changes and fine-tuning initial parameters which easily can be implemented by the use of appropriate feature definitions.

2.5. Reverse Engineering and Knowledge Fusion

Reverse engineering techniques are used to extract shapes and geometries from exiting products [7,8]. The outputs of RE procedures usually poorly represent the design logic used to create the parts. The gap between reverse engineering (RE) techniques and the requirement of embedding design intents into the CAD files of products retrieved using this method was discussed by Durupt et al. in their work [7,8]. The traditional RE tools allow creating the geometries of existing products but lack embedding the design intents. The method proposed in their work suggested procedures to integrate RE tools with knowledge fusion techniques. Similarly, Li et al. suggested the use of knowledge-driven graphical partitioning approaches to embed design intents to the RE scan results [16, 37].

Topological approaches have recently become more popular for their ability to generated 3D free shape models based on finite element concepts. However, like that of the RE techniques, a lot of effort needs to be done before smoothly extract simple CAD models from this shapes. The work of Larsen and Jensen [15] focuses on the investigation of methodologies to extract parametric models out of topologically optimized 3D shapes.

Chapter 3

The Proposed Approach

3.1 Introduction

Product modeling involves the process of creating part geometries by combining individual basic semantic entities called features. A feature is a data structure with members of geometric elements and associative relations. The ability to create relationships between the data members of different features allows controlling part dimensions parametrically. With this modeling approach constraints can easily be imposed on geometric entities defining the features. The data from the features can easily be accessed and modified making this method robust in managing change propagations and modifications in the design process. In addition to geometric parameters, these features can be designed to store other design entities such as part material specifications and manufacturing methods. The fact that features are basically data structures makes them play an important role in the automation processes of conceptual design cycles.

In this chapter are proposed and discussed two methods to be used in the implementation of feature based CAD modeling techniques in the development and design automation processes of variational mechanism.

In the first section, the general design process modules and data flow architectures will be discussed. The second section focuses on the introduction and testing of a method designed to utilize the powers of artificial neural networks in the mechanism synthesis.

3.2 General Design Automation Method

The design of mechanisms and products that are subject to frequent changes and modifications involves several application-dependent processes utilizing a set common data. The given specifications, standards and design requirements may be changed at any time during the development process. These changes can be evoked by the customers as well as due to newly arising engineering requirements. Without having in place a system by which these activities are handled in a very efficient manner, the costs associated with the changes and modifications could unjustified the product need.

This paper proposes a method by which such changes and design intent modifications are handled in a very cost effective and timely manner using feature based approach to reduce the CAD modeling and the overall design cycle times.

By employing commercially available programming and feature based 3D modeling tools, it is possible to create a reliable automation procedure which accommodates for the inevitable changes and modifications.

3.2.1 The Proposed Design Procedure

The following flowchart summarizes the general automation procedure proposed for this purpose. The area of data communications between different programming and modeling tools will not be fully investigated in this paper due to the time constraints. Instead the intended communication is achieved by the use of neutral text data files written and updated by the program codes developed for this purposes.

In Figure 3-1, design input information in the form of user specification is used to start the process. This input, together with additional engineering rules and intents, is used by the *Kinematic Analysis* algorithm discussed in the next section to synthesize the linear dimensions of the mechanism.

The newly calculated linear dimensions of the mechanism under investigation will be used in upcoming modules to model its skeleton assembly model to be used in an initial kinematic and dynamic analysis. This process results in the identification of forces and moment reactions between contacting joints and bodies. The output of the *Dynamic Analysis and Simulation* module will be used to establish the free body diagrams (FBD) and mass-acceleration diagrams (MAD) to be used during the design and optimization phases.

Results obtained from these stages, together with the initial input specification values, will be used in the design of linkages and members of the mechanism. One or more applicable optimization criteria can be used in order to determine a set of optimum cross-dimensional parameters for the machine elements. In addition to the above mentioned inputs to the *Design and Optimization* phase, design codes, standards, assumption, and factor of safety are some of several additional factors to be considered depending on the type of product being designed.

Based on dimensional data determined by previous processes the 3D models of the mechanism components will be modeled following feature based techniques. Application Programming Interfaces (API) of most commonly used modeling platforms can be used for this purpose. The choice of the programming and modeling tools depends on the compatibility of tools and the familiarity of the personnel using them.

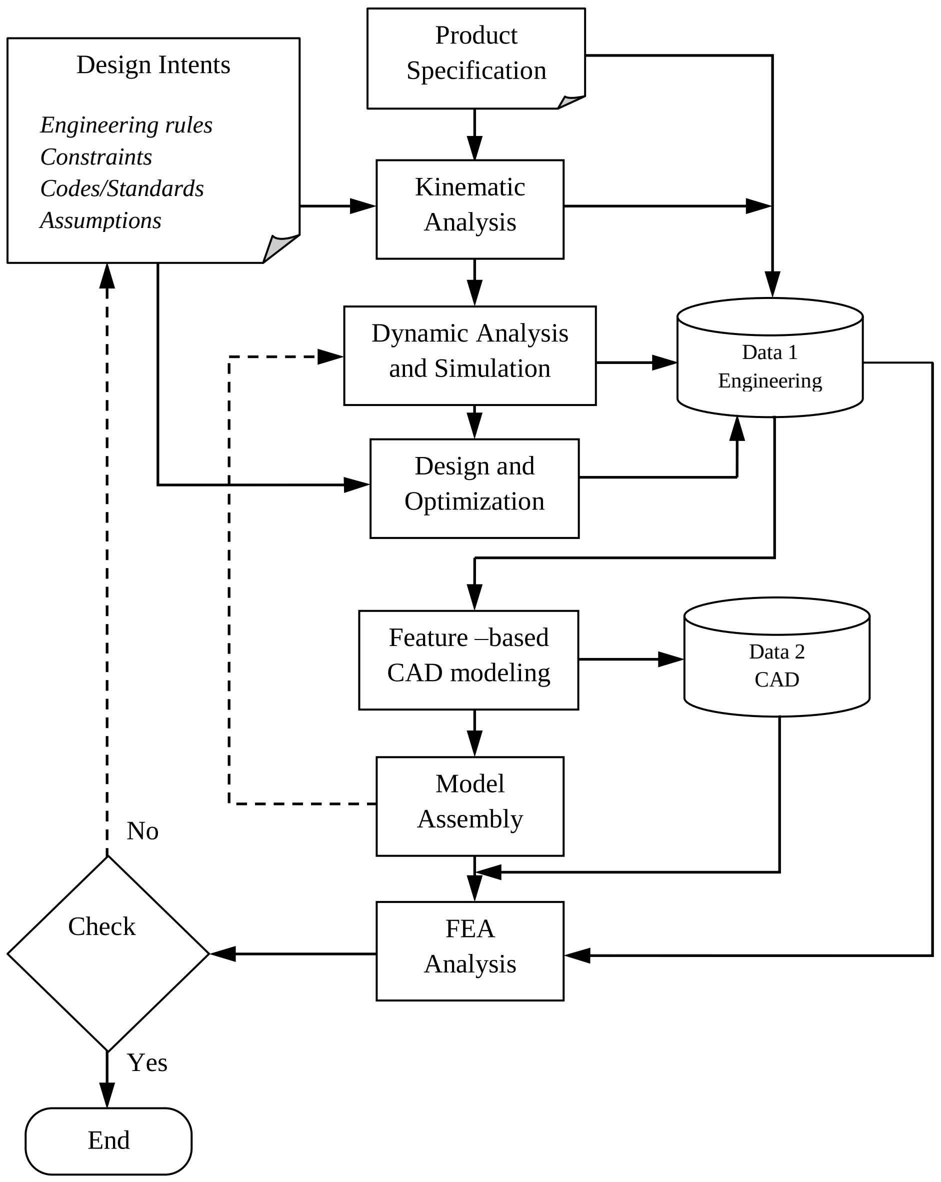


Figure 3-1 Design process modules and data communication routes

For this research, the programming part of the case study was carried out in C++ programming using Visual Studio 2008®. The final 3D models were generated from the codes using the UG NX 7.5 modeling software.

These models, assembled preferably, will be exported back to the *Dynamic Analysis and Simulation* block to take the effects of their newly created 3D dimensions (inertia effects) into consideration. This first stage loop will be repeated until a stopping criterion is met.

The strength and deformation of parts and models passing this screening stage can be further examined using FEA techniques. In the event these components fail to meet the qualification criteria set for the FEA stage, the entire iteration can be restarted with modified input parameters to address the shortcomings.

3.2.2 Features and Data Structures

Concurrent engineering and product development processes involve the participation of personnel with different engineering and technical backgrounds. In most cases these persons work from within different departments requiring an efficient mechanism for smooth information transfer among them.

Any information, whether in the form of initial input or generated data, has a very good chance of being used by more than one function module or application. In addition, a series of design data for a particular product family needs to be stored in a systematic repository database. This has potential for future use in the areas of knowledge fusion and as a training source for artificial neural network applications.

Data structures, implemented by using object oriented programming tools, play the role of addressing these crucial needs. The *Product Specification* input shown in Figure 3-1, needs to be organized in a systematic manner and its scope be defined as "*global*" or "*local*" in order to define its accessibility by individual program modules. This is done by defining a data structure and instantiating its object. The following is an example of a class defined in MATLAB®. The data structure for handling a particular problem is defined by creating an object instance of this class and entering values to its data members.

```
classdef Product_Specification_c
    properties
        Title = 'Specification Parameters'
        T_1 = 'Geometric Spec.'
        G1 = 0;
        G2 = 100;
        Gn = 0;
        T_2 = 'Material Spec'
        Modulus_Elasticity = 210e9;
        Poisson_ratio = 0.3;
    end
end
```

For example, to create a data structure for a new product model called Product_Spec_2010 by instantiating the above definition the following command is used. Note: Neither this particular example data structure nor its values are real values and are used here only for explanation purpose.

```
global Product_model_2010
Product_model_2010 = Product_Specification_c
```

And the values of this data structure are updated using the following object oriented programming syntax:

```
Product_model_2010.G1: new value
Product_model_2010.G2: new value
Product_model_2010.Gn: new value
Product_model_2010.Modulus_Elasticity: new value
Product_model_2010.Poisson_ratio: new value
```

The collection and input methods of the individual entities for the data structure greatly depends on the convenience and applicability to a particular problem. Initial values can be assigned during the definition of the data structure or it can be updated afterwards using both the command line and graphical user interfaces (GUI).

The following is a real example data structure taken from the excavator arm mechanism case study.

```
classdef c_Spec_Data_SI
    properties
       Title = 'Commercial Specifications and Vehicle Dimensions'
        Maximum_Reachout_at_Ground_Level_S1 = 0;
        Maximum_Digging_Depth_S2 = 0;
        Maximum_Cutting_Height_S3 = 0;
        Maximum_Loading_Height_S4 = 0;
        Minimum_Loaidng_Height_S3 = 0;
        Horizontal_Distance_H = 0;
        Vertical_Distance_V = 0;
        Vehicle_Weight = 5000;
    end
end
```

An object of this structure, SpcDat, instantiated and completed with its own values takes the form:

```
SpcDat =
  c_Spec_Data_SI
  Properties:
      Title: 'Commercial Specifications and Vehicle Dimensions'
    Maximum_Reachout_at_Ground_Level_S1: 5.6700
              Maximum_Cutting_Height_S3: 3.7248
              Maximum_Loading_Height_S4: 1.3521
                  Horizontal_Distance_H: 0.9857
                    Vertical_Distance_V: 1.2300
                         Vehicle_Weight: 5000
```

The set of data generated within the *Product Specification* (PS) module is used directly by the Kinematic Analysis (KA) module when calculating the linear dimensions of the mechanism or manipulator. The KA, in turn, generates its own data structure and made it available to be used by downstream functional modules.

The number of programming applications and tools involved in the system dictate the number of databases involved. If there are two or more programming tools running on different platforms involved, it may be required to devise a mechanism by which their respective databases are able to communicate with each other.

In Figure 3-1, it is assumed that the programming environment used for kinematic analysis and dimensional synthesis is different from the one employed by the API of the CAD modeling application, as this is the usual case. This is a very common practice since MATLAB® and Maple are usually used for engineering design calculations and optimizations processes while C#, C++, and VB are used for programming CAD in with the API tools. However, all of these tools are expected to operate based on a common set of data model and parameters produced during the initial phase of the conceptual design cycle. Accordingly, *Data 1* and *Data 2* in Figure 3-1 are communicated by neutral intermediate text data files. Similar data structure is needed to be defined from within the other programming applications involved to read and import the data exported by other applications. These definitions do not have to be the exact copy of the previous one as long as the necessary parameters are imported. But defining all corresponding data structures in a consistently similar manner avoids confusion and helps one with better data management.

The concept of feature has been investigated in greater depth in the last couple of decades to address the emerging product development and manufacturing needs and challenges. At the beginning, the term feature was used to refer only to the geometrical aspects of a model such as slots, holes and chamfers. The fact that products development process includes much more than geometric entities has forced researchers to look into ways of embedding more information into the CAD models. Today’s features have broader meaning in this sense. Both geometric and non-geometric information are able to be imbedded into the model aiding in rapid and reliable product information transfer mechanism.

The following are some of the many features used in this work:

- Coordinate system features: Used in the creation of relative and absolute CSYS.
- Skeleton functional features :Used in the development of skeleton product profiles
- Embodiment features :Features responsible for creation of 3D geometries
- Sheet body features
- Solid body features
- Curve features

3.3 Method for Mechanism Dimensional Synthesis

Existing manipulator mechanism products are frequently redesigned and customized to meet specific operational needs and increased efficiency. Specialized manipulators are needed to perform out-of-the-ordinary tasks under constrained space limitations. Although adding new design features to these existing models is one way of increasing versatility and addressing these needs, the approach might require the development of additional design procedures and incorporating them into the existing knowledge base. In most cases, however, the objective is achieved by adopting a different set of configurations.

The final spatial configurations of the overall assemblies are the bases on which users of these products evaluate the dimensional specifications. The conceptual design of these products usually starts with a set of target configurations or prescribed paths and motions identified by the end users that needed to be achieved by the overall mechanism. The dimensional synthesis phase of the design focuses on determining individual linkage dimensional parameters which when assembled in the mechanisms will meet the configuration requirements.

In the event when a single position and orientation (pose) of the end effectors of the manipulator is defined for known values of its linkage dimensions, the joint variables are calculated using inverse kinematics procedures. Unlike the case of forward kinematics (direct configuration) problems, the solution to inverse kinematics problems usually is not unique. This poses considerable challenge when trying to automate the conceptual design process and implement it using programs. In addition, the fact that the calculated linkage dimensions and joint variables are expected to fully satisfy other set of additional configuration/path parameters makes this approach more difficult to be implemented on multivariable problems.

Formulating a set of parametric geometric relationships for each configuration in terms of similar linkage parameters and searching a solution using optimization techniques is a standard approach. The implementation of this method requires a vector of initial solutions very close to the actual to be defined. Since there is a very strict correlation between the configurations parameters, random values cannot be used as an input when solving this system of equations. Failure to do so may produce mathematically accurate but physically impossible solutions.

In this section is a method proposed by which dimensional synthesis is performed for manipulator mechanisms based on a random configuration parameter inputs. The method will be implemented in MATLAB and tested using the excavator arm mechanism as a case study.

Most optimization techniques usually require a very good initial solution to be defined for them to produce sound solutions. One of the objectives of this paper is to introduce a system by which a set of initial solutions which are reasonably close to the actual solution can be generated. Optimization techniques, when applied to the problems of dimensional synthesis of prescribed precision points, commonly encounter the difficulty of producing reasonable results from the point of view of practicality due to two reasons. The first reason is the closeness requirement of the pre-defined initial solution. The other reason is the incompatibility or feasibility issue of the prescribed precision points. This is to say that prescription of unrealistic and ambitious specifications most likely produce, if the search converges to a solution at all, unrealistic solutions.

The hybrid method proposed in this research can be summarized by the flowchart as shown in Figure 3-2. It is the objective of this paper to introduce a hybrid method in which a well trained artificial neural network (ANN) tool is used to generate a set of high quality initial solution suggestions for target mechanism parameters based on the user specifications while optimization techniques are used to finally synthesize the necessary dimensions. The hybrid method attempts to jointly employ the powers of optimization procedures and neural networks to synthesize the dimensions of mechanisms and manipulators.

The user specifications are quasi-dependent in nature to each other. There is an acceptable range of values for a single configuration parameter, S_i, that can feasibly coexist with a set of the remaining prescribed configuration parameters, $S_{j\neq i}$. In this regard, the given initial specification data set has to be validated before used in the design calculations.

3.3.1 Synthesis and Validation Procedures

The proposed method can be decomposed into the following stages: (1) Artificial neural network training; (2) Input parameter validation; (3) System Testing; (4) Initial solution generation; (5) Mechanism parameter synthesis; (6) Result verification; (7) System test with random existing values.

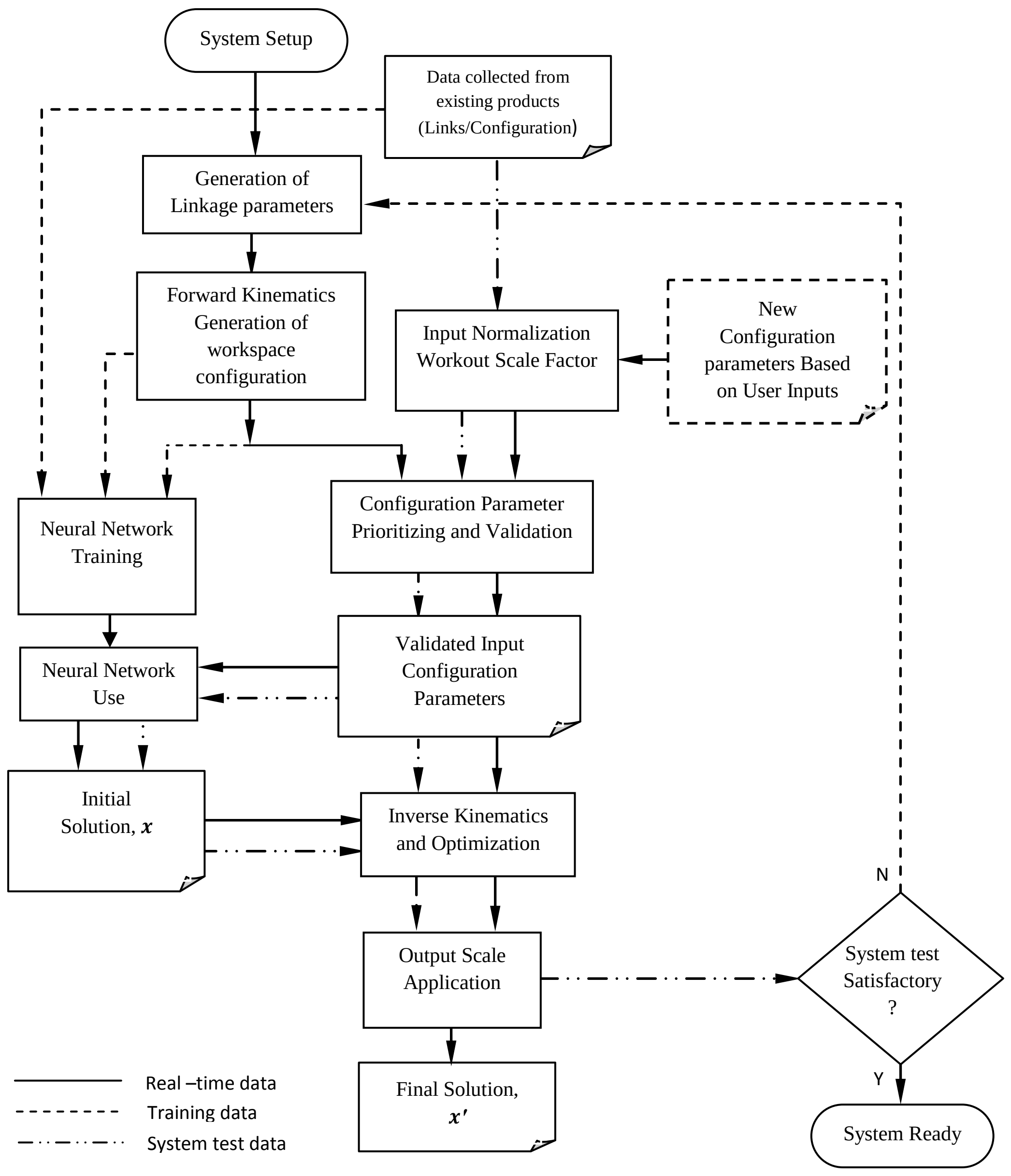

System Setup
Data collected from existing products (Links/Configuration)
Generation of Linkage parameters
Forward Kinematics Generation of workspace configuration
Input Normalization Workout Scale Factor
New Configuration parameters Based on User Inputs
Neural Network Training
Configuration Parameter Prioritizing and Validation
Neural Network Use
Validated Input Configuration Parameters
Initial Solution, $\boldsymbol{x}$
Inverse Kinematics and Optimization
N
Output Scale Application
System test Satisfactory ?
Y
Real –time data
Training data
System test data
Final Solution, $\boldsymbol{x'}$
System Ready

Figure 3-2 Dimensional synthesis methodology based on ANN and optimizations

3.3.2 Artificial Neural Network Training

Essentially, training the ANN is performed to build a database which will be used to generate the feasible suggestions of the initial mechanism parameters according to new configuration specifications. The first step is to collect the training data. Ideally, such training data can be obtained from those existing similar product information catalogues, usually in the form of product families, because the relevant data from that channel is proven workable with both input and output sets. As shown in Figure 3-2, the proposed method makes use of such data as indicated by the top job block. Unfortunately, although these real product data sets are very useful for training the ANN, the number of available data sets is always not enough. To find a solution for the shortage of training data, forward mechanism equations can be utilized to create as many input/output data sets as required [34]. Note that the generation of such simulation data is necessary because the available data is usually insufficient to serve the training purpose and the extra effort of collecting additional real product data is deemed costly.

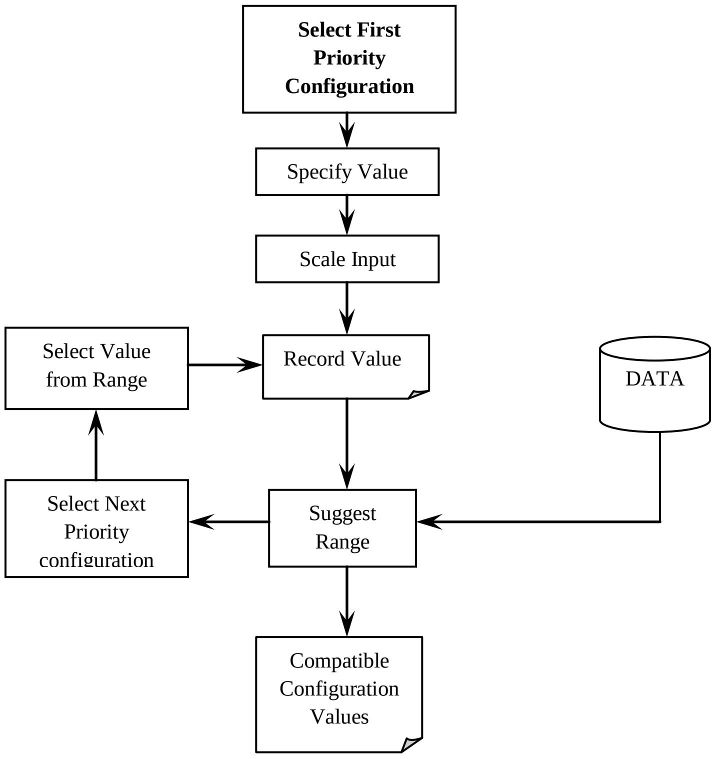
In the case of data generation process, the configuration parameters which define the total workspace of the mechanism assembly will be generated for the given set of linkage dimensions using forward kinematic equations. This is a mapping process in which the **mechanism parameters** (linkage dimensions) are mapped to the envelope **configuration parameters** (work range) of the workspace or the working path in the case of a planar mechanism.

When training the artificial neural network (ANN), both the existing real product data sets and the generated data sets will be used in the reverse sense; that means the configuration parameters are used as the input data for the training while the mechanism parameters are used as the target output data. Note here that the generation and training methods have been used previously [34] and have provided satisfactory outcome while the proposed real product data sets still play an important role to incorporate the industrial optimization factors that are well embedded implicitly in them on top of the mathematical solutions. In fact, it may be potentially useful to increase the weights of such real product data sets in the training of the ANN. Due to the time limitation, this idea of enhancement will be explored in the future.

Since the ANN is expected to be effectively used only for those parameters lying within the ranges of its training data, to make the training data more generically useful, normalizations of the input vector as well as the output vector during the training cycles should be carried out. Similarly, during the application of the trained neural network, the input and output new dimensional parameters have to be scaled or normalized to make sure they lie within the ranges.

3.3.3 Input Configuration Parameter Validation

In addition to the training of ANN, to search for a feasible mechanism parameter solution from a given set of configuration parameters, it is necessary to make sure that the configuration parameter values are compatible with each other and practically feasible to exist. If this condition is not met, the analysis may give results inapplicable to practical cases. Figure 3-3 shows the procedure adopted to validate input configuration parameters. It is worth noting that the term *validation* is used here only to evaluate the given prescribed set of parameters from the point of view of their combined applicability to a particular machine or manipulator configuration. The validation is done by checking if the given multiple input configuration parameters, after being scaled or normalized, lie within the relative permissible ranges established by the collected and generated data. The ranges derived from the collected data are based on the statistic analysis of results of all the generated and real product models data.



Select First Priority Configuration
Specify Value
Scale Input
Select Value from Range
Record Value
DATA
Select Next Priority configuration
Suggest Range
Compatible Configuration Values

Figure 3-3 Input parameter validation and prioritizing scheme

3.3.4 System Testing

To validate the overall procedure, real product data sets are used again for the testing purpose, as shown by line type #3 in Figure 3-2. This time, different from the ANN training process, the work-range configuration parameters are fed into trained ANN module to generate the initial solutions of the targeted mechanism dimensions. Then, together with work-range configuration parameters, the initial dimensions are used as the seeding vector to search for the goal vector of the targeted mechanism dimensions. This set of output goal vectors is compared with the real product mechanism dimension vectors. Theoretically, the system output should be well within the specified tolerance of the system accuracy requirement. Note, relatively, the real product data sets are only a small portion of the overall ANN training data sets. If the system is not up to the accuracy expectation, then more training data sets are required from the both channels as discussed previously.

3.3.5 Initial Solution Generation

In this stage, the validated configuration parameters will be passed to the ANN module to generate initial solutions. This solution is in turn will be used by the appropriate optimization to refine and get the final solutions.

3.3.6 Mechanism dimensions synthesis

The dimension synthesis is carried out with optimization algorithms. This part is specific to the nature of the mechanism in question. In this work, an excavator arm linkage system is studied. The details of the algorithms are introduced in section 4. Finally, the calculated mechanism parameter solutions have to be scaled back to the original ratio before being reported back as a solution to the user.

3.3.7 Results Validation

The optimization results are to be validated before they are adopted in the design and displayed in an appropriate CAD context. The straight forward procedure is to apply forward mechanism simulation to check for the work-range space or path details against the specifications. Theoretically, if the results are not good enough, the troubleshooting procedure must be carried out. So far, with the limited tests, the generated results are quite satisfactory as discussed in the following section, so the troubleshooting method was not explored further.

3.3.8. Random System Validation Check

To measure and validate the performance of the system, again, randomly selected configuration parameter data sets from the existing products' database are selected and used in the generation of mechanism parameters using the proposed method. The results of the procedure are cross-checked against the actual dimensions and the efficiency of the method is measured.

Chapter 4

Design Calculations - Case study

4.1. Excavator Case Representation

In the conceptual design process of an excavator, translating the work-range specification parameters (prescribed points or an envelope path) into linear dimensions of the arm mechanism is the very first stage. To do this, the boom, stick, and buckets of this planar mechanism are represented by linear linkages, and other elements such as hydraulic cylinders and bucket transition four-bar linkages are left out of consideration at this stage. These three links, connected in boom-stick-bucket sequence, are positioned and oriented in different poses such that their final configurations pass through the input specifications. Figures 4-1 and Figure 4-2 show the traditional catalogue specification dimensions $(S_1, S_2, \dots S_5)$ and the representation of the mechanism by linear elements (l_1, l_2, l_3, β), respectively.

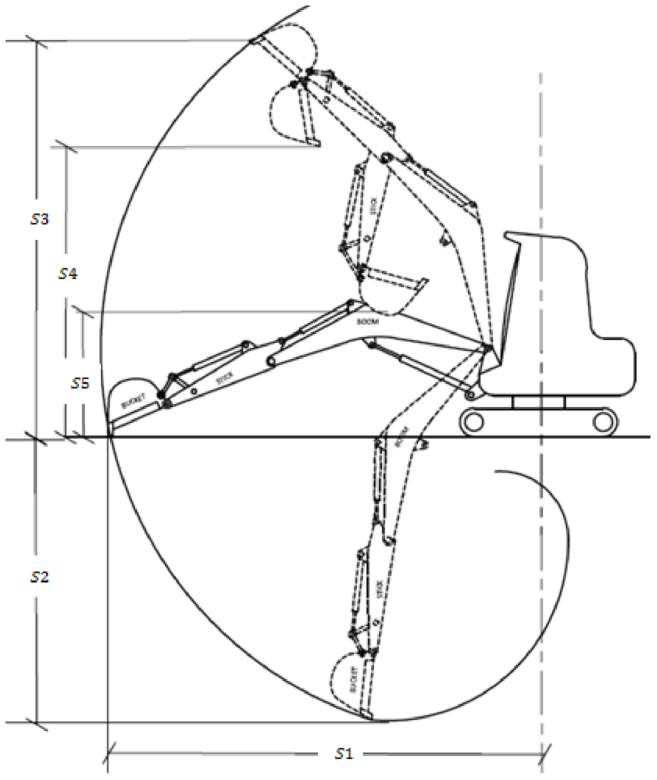


Figure 4-1 Work-range Configuration Parameters

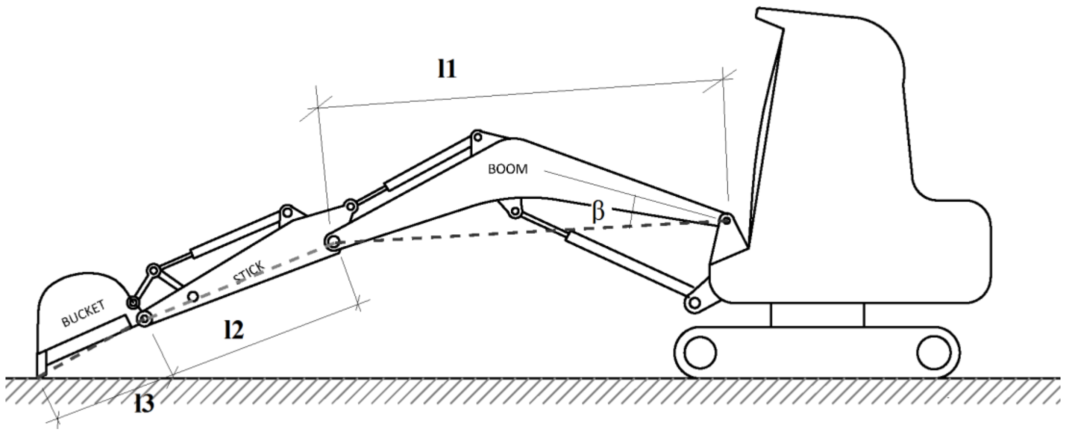


Figure 4-2 Linear representation of excavator arm members

Table 4-1 Hydraulic Excavator Work Space Configuration Parameters

S_1	Maximum Reach at Ground Level
S_2	Maximum Digging Depth
S_3	Maximum Cutting Height
S_4	Maximum Loading Height
S_5	Minimum Loading Height

Table 4-2 Mechanism Linkage Dimensions

$l1$	Hinge to hinge boom length
$l2$	Stick length
$l3$	Hinge to tip bucket length
β	Boom deflection angle β

The design process hence involves determining a set of individual linkage dimensions of the excavator arm mechanism that, when connected to each other and form the overall mechanism, satisfy the working envelope requirements.

Unlike forward kinematic problems in which the location and other properties of the end effector is to be calculated based on different joint variables and linkage dimensional inputs, this problem involves the task of determining the joint variables and linkage dimensions for a given set of end effector configurations; bucket in this case. In forward kinematics or direct configuration analysis, the task is usually to determine the final configuration of the mechanism based on a given set of joint variables and linkage dimensions. This is a relatively simple and straight forward process since the analysis usually leads to a unique solution. The inverse process being investigated in this research, on the other hand, is relatively complex due to the availability of multiple solutions.

4.2. Data Generation for Neural Network Training

The main purpose of this task is to generate configuration and linkage parameter data sets to be used for training the proposed ANN. The ANN will be used in latter stages to narrow down and select a physically viable set of linkage parameters to be used as initial solutions. This is entirely a forward kinematic procedure in which each final vector of configuration parameters**, S**, is determined from a given set of linkage dimensions and joint variables, ***L***.

Where $\boldsymbol{S} = (S_1, S_2, \ldots, S_5)$ *and* $\boldsymbol{L} = (l_1, l_2, l_3, \beta)$

The following sub sections describe the mathematical model derived for working out the envelop path configuration parameters $(S_1, S_2, \ldots, S_5)$ from the mechanism linkage parameters, (l_1, l_2, l_3, β).

4.2.1 Maximum Reach-out at Ground Level (S_1)

The position of the bucket tip is calculated using the forward kinematic methods. To apply this method to this problem, the individual rotational and linear transformation matrices are formulated using the Denavit-Hartenberg convention.

By applying the Law of Cosine to Figure 4-3 shown below the following mathematical relationship is formulated.

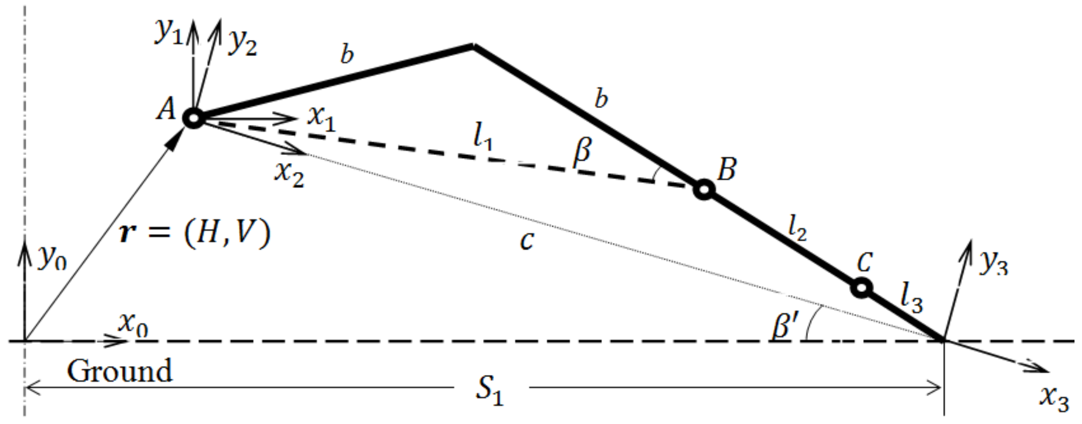


Figure 4-3 Maximum out-reach at ground level

$$c^2 = (l_2 + l_3)^2 + {l_1}^2 - 2l_1(l_2 + l_3)\cos(180 - \beta) \tag{4.1}$$

$$c^2 = (l_2 + l_3)^2 + {l_1}^2 + 2l_1(l_2 + l_3)\cos\beta \tag{4.2}$$

$$c = \sqrt{(l_2 + l_3)^2 + {l_1}^2 + 2l_1(l_2 + l_3)\cos\beta} \tag{4.3}$$

$$\sin\beta' = \frac{V}{c} = \frac{V}{\sqrt{(l_2 + l_3)^2 + {l_1}^2 + 2l_1(l_2 + l_3)\cos\beta}} \tag{4.4}$$

$$\beta' = \sin^{-1}\left(\frac{V}{c}\right) = \sin^{-1}\left(\frac{V}{\sqrt{(l_2 + l_3)^2 + {l_1}^2 + 2l_1(l_2 + l_3)\cos\beta}}\right) \tag{4.5}$$

$$(S_1 - H)^2 = (l_2 + l_3)^2 + {l_1}^2 + 2l_1(l_2 + l_3)\cos\beta - V^2 \tag{4.6}$$

$$(l_2 + l_3)^2 + {l_1}^2 + 2l_1(l_2 + l_3)\cos\beta - V^2 - (S_1 - H)^2 = 0 \tag{4.7}$$

The sequence of frame of reference translation from the origin to a frame located at the tip of the bucket is represented by the homogeneous transformation:

$$A = T_{x\prime H}\, T_{y\prime V}\, R_{z\prime -\beta}\, R_{z\prime -\beta\prime} \tag{4.8}$$

Where

$T_{x\prime H}$	Linear displacement in the positive x direction with H value
$T_{y\prime V}$	Linear displacement in the positive y direction with V value
$R_{z\prime -\beta\prime}$	Rotation about the z axis by angular value of $-\beta'$
$T_{x\prime c}$	Linear displacement in the positive x direction by a value of c

The rotation sequences of Eq. (4.8), when represented by the corresponding matrices, take the form

$$A_{S1} = \begin{bmatrix} 1 & 0 & 0 & H \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & V \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-\beta') & -\sin(-\beta') & 0 & 0 \\ \sin(-\beta') & \cos(-\beta') & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & c \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

The resulting homogenous transformation matrix is then given by:

$$A_{S1} = \begin{bmatrix} \cos\beta' & \sin\beta' & 0 & H + c\cos\beta' \\ -\sin\beta' & \cos\beta' & 0 & V - c\sin\beta' \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

The value of the *maximum out-reach at ground level* is then extracted from the above homogenous transformation matrix. The expression in cell $(1, 4)$ is the value of the x coordinate of the bucket tip from the origin of the fixed reference frame, which in this case is the same as the value of the maximum reach-out at ground level, $S1$.

$$S_1 = |A_{S1}(1{,}4)| = |H + c\cos\beta'| \tag{4.11}$$

4.2.2 Maximum Digging Depth (S_2)

The maximum digging depth requires the definition of angle α_2 measured from the vertical to indicate the lower limit of the boom angular displacement about the base hinge. For a given value of this limiting angle, the maximum digging depth is expressed mathematically using the Denavit-Hartenberg convention.

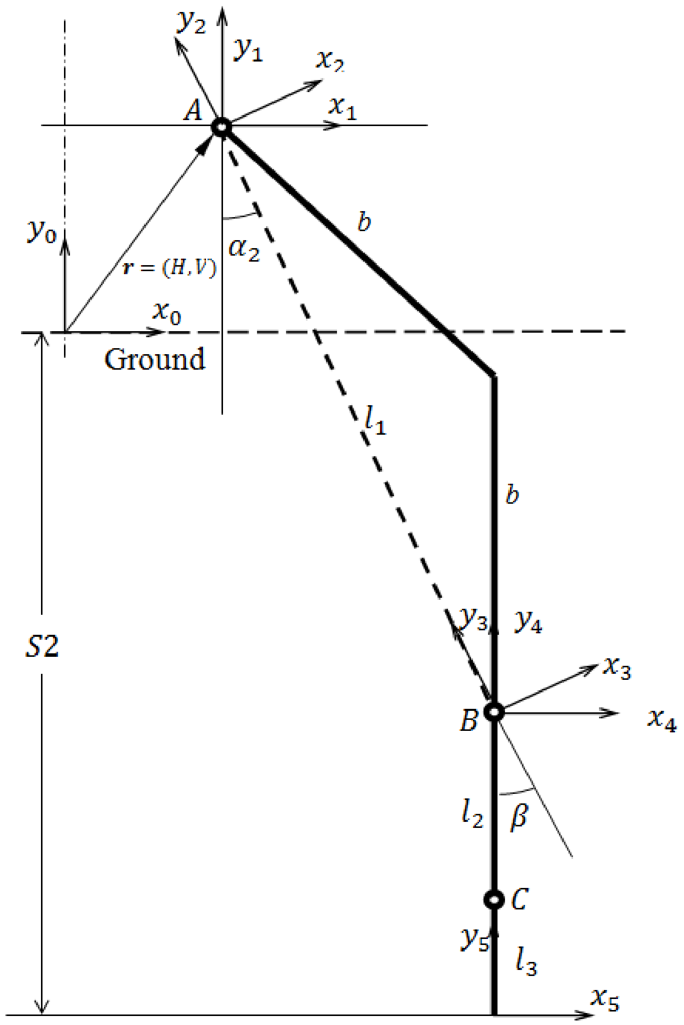


Figure 4-4 Maximum digging depth

Again, by using Law of Cosine,

$$l_1{}^2 = b^2 + b^2 - 2b^2\cos(180 - 2\beta) \tag{4.12}$$

Where b in this equation is the length of each sides of the boom. For the purpose of simplification, they are assumed to be of equal length in this development.

$$l_1 = 2b\cos\beta \tag{4.13}$$

Referring to Figure 4-4,

$$S_2 = l_1\cos\alpha_2 + l_2 + l_3 - V \tag{4.14}$$

$$S_2 = l_1\cos\alpha_2 + l_2 + l_3 - V \tag{4.15}$$

$$l_1\cos\alpha_2 + l_2 + l_3 - V - S_2 = 0 \tag{4.16}$$

The homogeneous transformation sequence in this case is given by

$$A_{S2} = T_{x,H}\ T_{y,V}\ R_{z,(\alpha_2)}\ T_{y,-l1}\ R_{z,-\beta}\ T_{y,-b} \tag{4.17}$$

$$A_{S2} = \begin{bmatrix}1 & 0 & 0 & H\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & V\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos(\alpha_2) & -\sin(\alpha_2) & 0 & 0\\ \sin(\alpha_2) & \cos(\alpha_2) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & -l1\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$\dots\begin{bmatrix}\cos(-\beta) & -\sin(\beta) & 0 & 0\\ \sin(-\beta) & \cos(-\beta) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & -b\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{4.18}$$

The resulting homogeneous transformation matrix takes the form

$$A_{S2} = \begin{bmatrix}\cos(\alpha_2-\beta) & -\sin(\alpha_2-\beta) & 0 & H + l1\sin(\alpha_2) + b\sin(\alpha_2-\beta)\\ \sin(\alpha_2-\beta) & \cos(\alpha_2-\beta) & 0 & V - l1\cos(\alpha_2) - b\cos(\alpha_2-\beta)\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{4.19}$$

The cell in this matrix representing the maximum digging depth is the y displacement is cell$(2,1)$.

$$S_2 = |A_{S2}(2,1)| = |V - l1\cos(\alpha_2) - b\cos(\alpha_2-\beta)\,| \tag{4.20}$$

4.2.3 Maximum Cutting Height (*S*$_3$)

For a given value of the upper angular limit of the boom rotation, α_1, the procedure for maximum cutting height expression formulation follows the similar procedure as that of the maximum digging depth calculation.

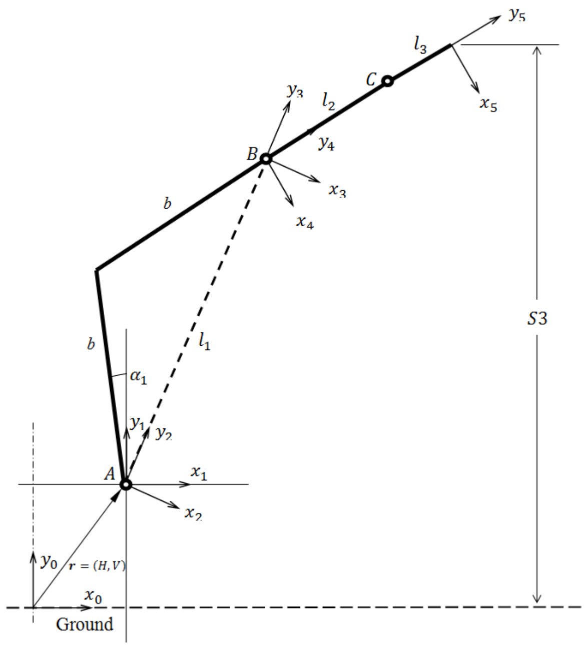


Figure 4-5 Maximum cutting height

$$H_2 = l_1 \cos\theta \tag{4.21}$$

Where θ in this case is given by

$$\theta = (\alpha_1 - \beta) \tag{4.22}$$

$$H_2 = l_1 \cos(\alpha_1 - \beta) \tag{4.23}$$

$$H_3 = l_2 \cos(\theta - \beta) \tag{4.24}$$

$$H_3 = l_2 \cos(\alpha_1 - 2\beta) \tag{4.25}$$

$$H_4 = l_3 \cos(\theta - \beta + \alpha_{bu}) \tag{4.26}$$

$$S_3 = V + H_2 + H_3 + H_4 \tag{4.27}$$

$$S_3 = V + l_1 \cos(\alpha_1 - \beta) + l_2 \cos(\alpha_1 - 2\beta) + l_3 \cos(\alpha_1 - 2\beta + \alpha_{bu}) \tag{4.28}$$

$$l_1 \cos(\alpha_1 - \beta) + l_2 \cos(\alpha_1 - 2\beta) + l_3 \cos(\alpha_1 - 2\beta + \alpha_{bu}) + V - S_3 = 0 \tag{4.29}$$

The homogenous coordinate transformation sequence for this configuration is given by

$$A_{S3} = T_{x,H}\ T_{y,V}\ R_{z,(\alpha_1-\beta)}\ T_{y,l1}\ R_{z,-\beta}\ ,T_{y,b} \tag{4.30}$$

$$A_{S3} = \begin{bmatrix} 1 & 0 & 0 & H \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & V \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\alpha_1 - \beta) & -\sin(\alpha_1 - \beta) & 0 & 0 \\ \sin(\alpha_1 - \beta) & \cos(\alpha_1 - \beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\dots \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-\beta) & -\sin(-\beta) & 0 & 0 \\ \sin(-\beta) & \cos(-\beta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.31}$$

$$A_{S3} = \begin{bmatrix} \cos(\alpha_1 - 2\beta) & -\sin(\alpha_1 - 2\beta) & 0 & H - b\sin(\alpha_1 - 2\beta) - l1\sin(\alpha_1 - \beta) \\ \sin(\alpha_1 - 2\beta) & \cos(\alpha_1 - 2\beta) & 0 & V + b\cos(2\beta - \alpha_1) + l1\cos(\alpha_1 - \beta) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.32}$$

The y displacement component of this matrix represents the maximum cutting height.

$$S_3 = |A_{S3}(2,4)| = |V + b\cos(2\beta - \alpha_1) + l1\cos(\alpha_1 - \beta)| \tag{4.33}$$

4.2.4 Maximum Loading Height (S_4)

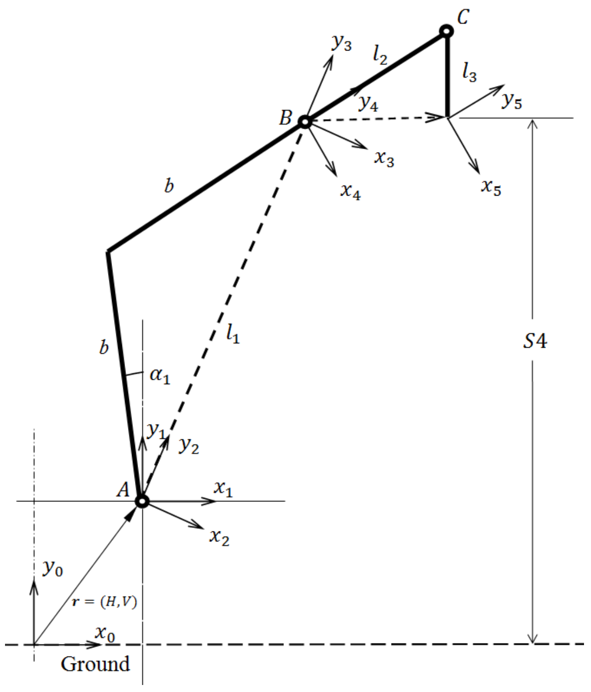


Figure 4-6 Maximum loading height

Referring to Figure 4-6, with slight modification of the expression developed for maximum cutting height and ignoring the orientation angle of the last frame of reference we get:

$$S_4 = V + H_2 + H_3 - l_3 \tag{4.34}$$

$$l_1\cos(\alpha_1 - \beta) + l_2\cos(\alpha_1 - 2\beta) - l_3 + V - S_4 = 0 \tag{4.35}$$

The expression for maximum cutting height is modified with minor changes to make it fit for this configuration. The last linear coordinate translation in this case is limited to $l2$ instead of $(b = l2 + l3)$. The bucket length $l3$ is further deducted from the y displacement component of the matrix.

The final result is given by the following matrix.

$$A_{S3} = \begin{bmatrix} \cos(\alpha_1 - 2\beta) & -\sin(\alpha_1 - 2\beta) & 0 & H - l2\sin(\alpha_1 - 2\beta) - l1\sin(\alpha_1 - \beta) \\ \sin(\alpha_1 - 2\beta) & \cos(\alpha_1 - 2\beta) & 0 & V - l3 + l2\cos(2\beta - \alpha_1) + l1\cos(\alpha_1 - \beta) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.36)$$

$$S_4 = |A_{S3}(2{,}4)| = |V - l3 + l2\cos(2\beta - \alpha_1) + l1\cos(\alpha_1 - \beta)\,| \qquad (4.37)$$

4.2.5 Minimum Loading Height (S_5)

Following similar procedure gives an expression for the homogeneous transformation matrix of the minimum cutting height configuration.

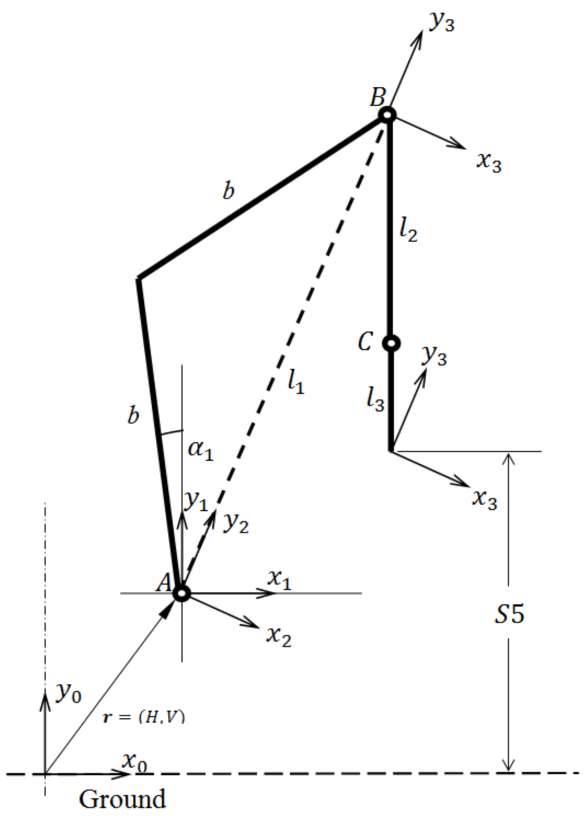


Figure 4-7 Minimum loading height

$$S_5 = V + H_2 - l_2 - l_3 \qquad (4.38)$$

$$S_5 = V + l_1\cos(\alpha_2 - \beta) - l_2 - l_3 \qquad (4.39)$$

$$V + l_1\cos(\alpha_1 - \beta) - l_2 - l_3 - S_5 = 0 \qquad (4.40)$$

$$A_{S3} = \begin{bmatrix} \cos(\alpha_1 - 2\beta) & -\sin(\alpha_1 - 2\beta) & 0 & H - l1\sin(\alpha_1 - \beta) \\ \sin(\alpha_1 - 2\beta) & \cos(\alpha_1 - 2\beta) & 0 & V + l1\cos(\alpha_1 - \beta) - l2 - l3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.41}$$

$$S_5 = |A_{S3}(2,4)| = |V + l1\cos(\alpha_1 - \beta) - l2 - l3| \tag{4.42}$$

4.3. Generation of Training Data

The required training data is generated by mapping the configuration parameter for a set of mechanism dimension parameters. MATLAB is used to implement this task.

$$\begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ \beta \end{bmatrix} \rightarrow \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \end{bmatrix} \tag{4.43}$$

4.3.1 Neural Network Training

Since the ANN is needed to serve the purpose of preliminary inverse kinematic analysis, the output data generated from the forward simulation, $\boldsymbol{S}$, will be used as the input data for its training while the linkage parameters vector, $\boldsymbol{L}$, is the target data.

Since the values of the configuration parameters depend also on the overall dimensions of the vehicle on which they are mounted, constant values for the x and y coordinates of the base hinge, H and V, are used in the analysis.

Accordingly, a two-layer feed forward ANN is designed to map seven input configuration parameters to four target parameters. The ANN has one hidden layer with 20 neurons and one output layer with four neurons. The network is trained using the *Levenberg-Marquardt* back propagation algorithm. Sigmoid activation functions are used for the first layer and a linear one-to-one activation functions for the output layer. The neural network is implemented using the neural network toolbox of MATLAB.

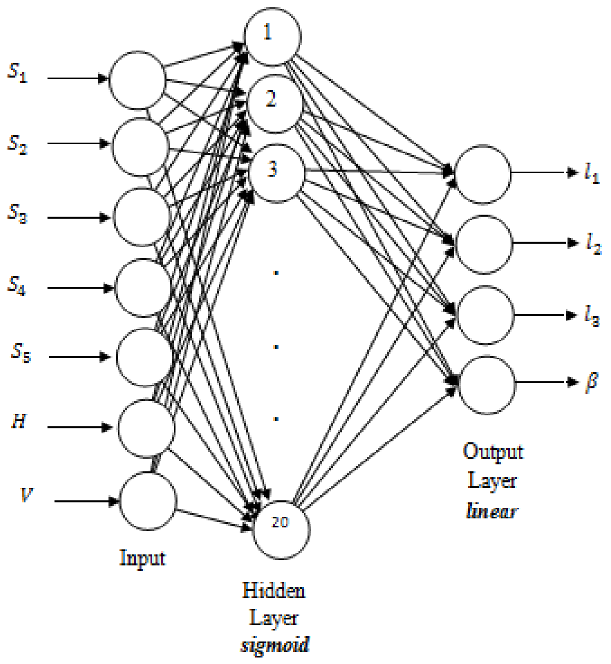


Figure 4-8 Architecture of the Neural Network

Given any one of the configuration parameters, $S_1, S_2, \cdots, S_5$, the developed method identifies possible ranges of the other four configuration parameters based on the data generated in the previous section. Since the data is generated by simulating a specific ranges of the linkage dimensions, this method scales input configuration parameters to make sure they lie within the available data range. Selected output ranges by this method are scaled back to the original before displayed for the user.

The method implemented using a MATLAB program called *f_Parameter_Sorter* provides option to the user to select which configuration parameter to start with and the sequence of upcoming selections. This option gives the flexibility of prioritizing the operational configurations as needed. Once the first item is entered for the first choice of configuration parameter, four different compatible configuration parameter ranges will be suggested for the others. This process will be repeated on the remaining four by selecting which configuration parameter out of the four to prioritize and picking its value from the range provided. The result of this second operation modifies the ranges of compatible values of the remaining three parameters. This process is repeated until all configuration parameters are assigned valid values. Figure 4-9 shows the convergence performance of the ANN training cycles while Figure 4-10 shows the standard ANN algorithm regression chart.

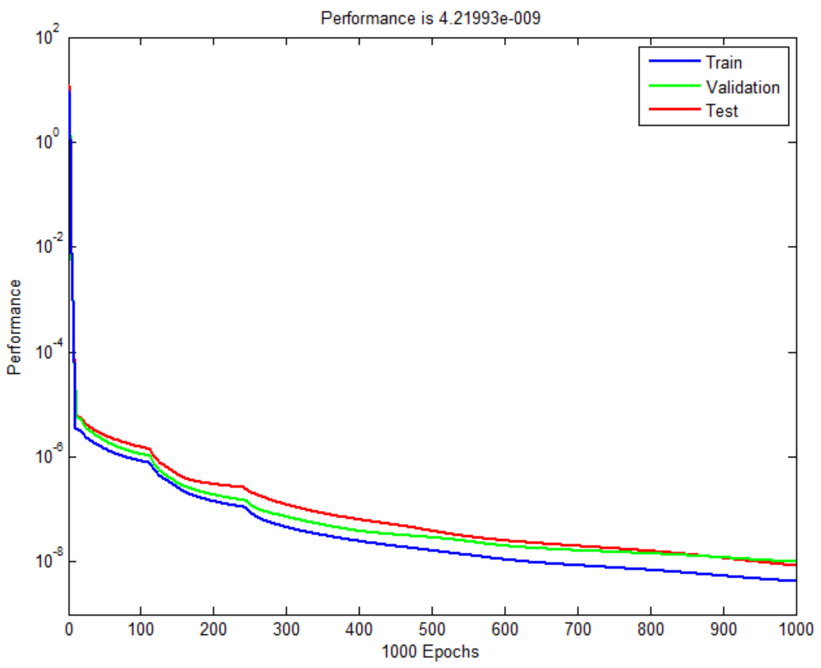


Figure 4-9 Performance of the neural network

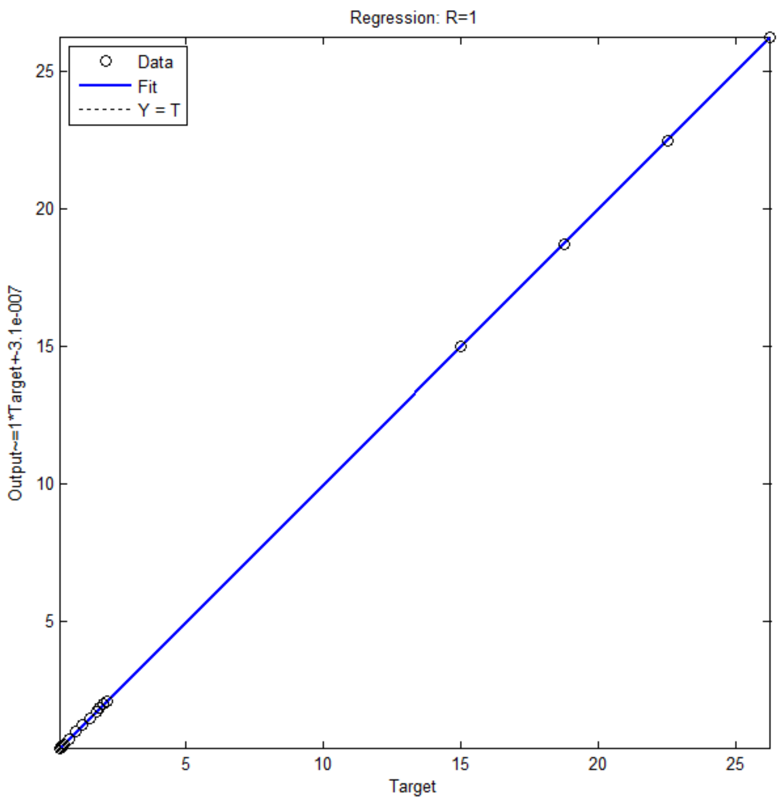


Figure 4-10 Regression result

4.3.2 Solving for Linkage Parameters

The Equations 4-7, 4-16, 4-29, 4-35, and 4-40 relate the specification values $S1$, $S2$, $S3$, $S4$, and $S5$ to the geometric dimensions of the excavator arm mechanism l_1, l_2, l_3, and β. Given the values of the other constant values, these non-linear equations can be solved using optimization techniques to determine the optimum linear and angular dimensions of the arm mechanism.

Since buckets are available as standard parts, the calculation at this algorithm focuses on determining the lengths of the boom and the stick together with the boom deflection angle, i. e. $l_1, l_2, and\ \beta$. The selection of the bucket is done based on the initial solution suggested by the ANN. To determine the above three unknown variables, a combination of three of the above non-linear equations are solved using a MATLAB® function, *fsolve( ),* which employs the power of the *Trust-Region-Reflective* Algorithm.

$$F(\boldsymbol{X}, \boldsymbol{S}) = \boldsymbol{0} \tag{4.44}$$

Where $\boldsymbol{X}$ and $\boldsymbol{S}$ are verctors of unknown mechanism dimension variables and input configuration specification parameters

$$\boldsymbol{X} = \begin{bmatrix} l_1 \\ l_2 \\ \beta \end{bmatrix} \quad \boldsymbol{S} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \end{bmatrix} \tag{4.45}$$

Considering the maximum reach-out at ground level, maximum cutting height, and maximum loading height, the vector of equations will be formulated as follows

$$\begin{bmatrix} {l_1}^2 + (l_2 + l_3)^2 + 2l_1(l_2 + l_3)\cos\beta - V^2 - (S_1 - H)^2 \\ l_1 \cos(\alpha_2 - \beta) + l_2 \cos(\alpha_2 - 2\beta) + l_3 \cos(\theta - \beta + \alpha_{bu}) + V - S_3 \\ l_1 \cos(\alpha_2 - \beta) + l_2 \cos(\alpha_2 - 2\beta) - l_3 + V - S_4 \end{bmatrix} = [\boldsymbol{0}] \tag{4.46}$$

The "*trust-region-reflective"* algorithm used to find the solution requires an initial solution to be defined as a starting point. The accuracy of the output for this particular problem greatly depends on the closeness of the initial solution to the actual solution. This is the stage where the suggested initial solution by the neural network is used. It is also expected that at this stage the viability of the initial input parameters, $S_1, S_2, \cdots, S_5$, is confirmed by the use of the valid ranges developed according to the procedure discussed previously.

4.3.3 Case Study Analysis Results and Discussion

In this research work, 10 existing excavator product configuration data sets are collected. Their contents are given in Table 4-3. A total of 1296 forward simulation data sets were generated and they were used to train the ANN module developed with Matlab. Then to test the system performance, the 10 product working-range parameter values were fed into the ANN, and the output of the ANN, i.e. the initial suggestions for the downstream optimization module, was presented in Table 4-4 (left half). In comparison, the final solutions generated after the optimization processes are also listed in Table 4-4 (right half).

Table 4-3 System testing data collected from the existing products (units: cm/degree)

	Configuration					*Mechanism dimensions*				*Vehicle*	
Product	***S1***	***S2***	***S3***	***S4***	***S5***	***l1***	***l2***	***l3***	***β***	***H***	***V***
1	359	183	344	226	107	174.1	88.2	51.9	24.5	63	75
2	413	252	384	271	109	205.9	102	61.1	25	68	86
3	412	260	359	246	111	201.2	99.4	67.9	28	74	93
4	435	228	422	283	106	203.3	105.2	64.9	30	78	90
5	409	248	385	267	125	201.3	99.5	61.5	25	66	84
6	372	208	371	257	110	171.4	89.1	58.2	24	77	82
7	352	196	331	235	92	159	86.3	49.6	22	77	71
8	345	203	338	238	99	165.1	88.4	49	20	64	73
9	332	184	335	238	104	163.4	83.8	47.5	24	55	71
10	415	254	368	272	110	204.9	102	63.1	25.76	68	81

Table 4-4 Initial and final solutions generated by the proposed method

	ANN initial solution (cm/degree)				**Optimization final solution (cm/degree)**			
Product	***l1***	***l2***	***l3***	***β***	***l1***	***l2***	***l3***	***β***
1	174.7	68.8	63.4	17.28	209	93	63.4	34.97
2	196.97	95.99	65.56	19.4858	200.22	95.47	65.56	30.0233
3	204.53	84.33	70.19	26.524	202.82	85.81	70.19	35.2464
4	181.04	79.14	78.3	10.7072	169.81	83.82	78.3	18.6694
5	177.03	73.81	66.62	15.6772	204.33	83.23	66.62	30.1607
6	168.06	80.5	61.03	13.2475	198.83	91.17	61.03	25.8929
7	153.89	82.59	52.79	15.2362	201.48	99.88	52.79	30.014
8	162.75	94.62	51.38	12.3882	197	98.48	51.38	32.8084
9	161.05	90.9	46.87	14.3593	211.44	97.56	46.87	32.3521
10	195.89	107.73	60.04	24.8409	187.9	92.6	60.04	31.9021

Table 4-5 Accuracy statistics of the system results

Dimensions	Average error (%)	Unbiased standard deviation	Root mean square error (RMSE)
l_1	8.627	0.1569	0.1489
l_2	1.4641	0.1356	0.1286
l_3	7.1778	0.085	0.0806
β	23.858	0.2652	0.2516

Clearly, the ANN module has served the purpose to provide useful initial suggestions that enabled the optimization module to find the feasible solutions for the given mechanism. Further, Table 4-5 shows the comparison results between the final solutions and the original real product data obtained for the 10 existing configurations, the average errors for linear dimensions are pretty close, i.e. within 10%; but the angular β has bigger difference from the original dimension, about 24%. The deviations of these errors are relatively small. Thus, the test results show that the proposed method has a capability to generate a reasonably closer set of initial solutions that can be used by optimization search algorithms employed to find the final solutions. The method can be further improved by fine tuning the optimization algorithms and the boundary conditions as well as using more real product data sets for ANN training.

4.4. Design for Strength

4.4.1 Homogeneous Coordinate Transformation

The output of the SimMechanics simulation provides only joint forces and motions expressed in the global reference frame. These global generalized joint forces have to be transformed into and expressed in separate coordinate systems local to the links or frame members under investigation.

In order to achieve this, three coordinate transformation matrices are developed. The boom, due to its geometrical deflection, has two sides and requires two different matrices to express forces in frames local to these sides. Since the stick has a straight axis, it needs only a single transformation matrix for reference frame manipulation.

The first step in this process is to identify stationary angles which, together with the linear dimensions, help to fully define the geometry of the boom and stick parts. This is followed by defining variable angles responsible for the operational configuration of the arm mechanism - digging operation in this case.

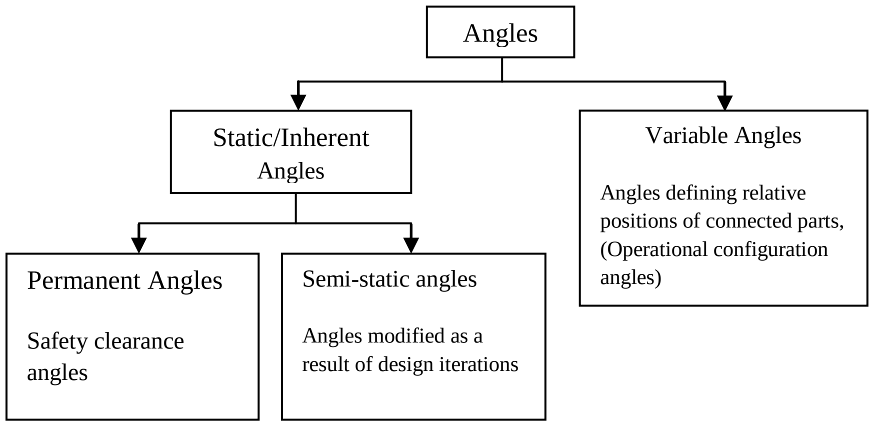

Angles
Static/Inherent Angles
Variable Angles
Angles defining relative positions of connected parts, (Operational configuration angles)
Permanent Angles
Safety clearance angles
Semi-static angles
Angles modified as a result of design iterations

Figure 4-11 Classification of angles

As shown in Figure 4-12 and Figure 4-13, angle $dig1$ and $dig2$ define the orientations of the boom and the stick with respect to the ground, respectively, during a digging operation. Unlike static angles which are always assigned positive values, variable angles are direction sensitive.

4.4.2 Boom Geometries, RB's

Referring to Figure 4-12, expressions for variable angles $dig1$ and $dig2$ can be formulated as follows.

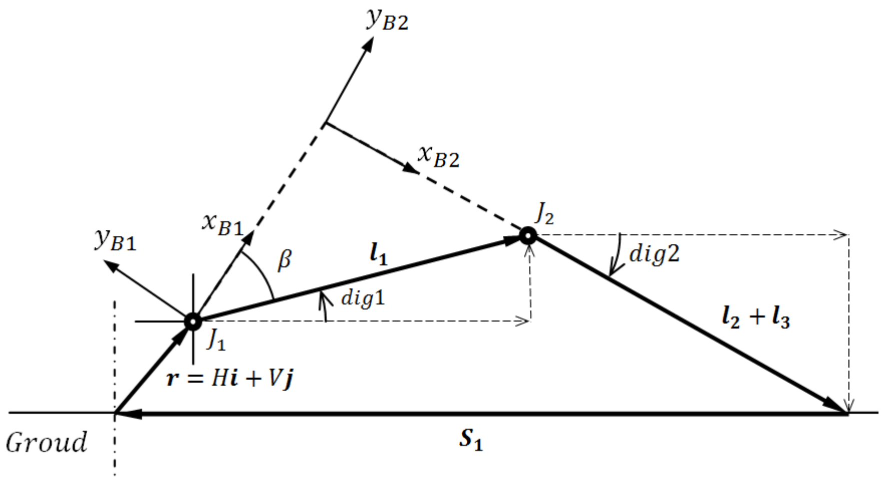


Figure 4-12 Operational configuration angles

Summing the components of vectors along the horizontal direction gives:

$$\sum V_x = S_1 - H \tag{4.47}$$

$$l_1 \cos(dig1) + \ l_2 \cos(dig2) + \ l_3 \cos(dig2) = S_1 - H \tag{4.48}$$

$$l_1 \cos(dig1) + \ (l_2 + l_3) \cos(dig2) - S_1 + H = 0 \tag{4.49}$$

Similarly, summing the components of these vectors along the vertical direction gives another expression.

$$\sum V_y = 0 \tag{4.50}$$

$$V + l_1 \sin(dig1) + (l_2 + l_3) \sin(dig2) = 0 \tag{4.51}$$

Solving Equations (4.49) and (4.51) simultaneously gives the values of $dig1$ and $dig2$.

The homogeneous coordinate transformation matrices for the first and second side of the boom are then derived using these calculated angles.

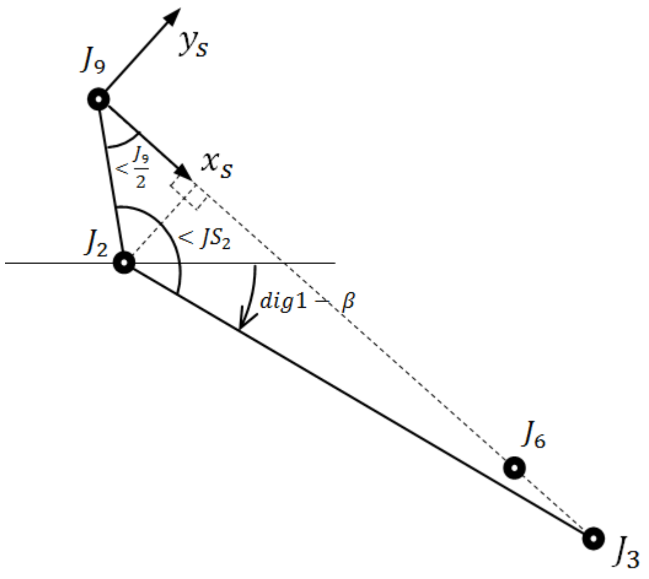


Figure 4-13 Stick structural angles

Boom Rotation Matrix I, RB_1

$$RB_1 = \begin{bmatrix} \cos(dig1 + \beta) & -\sin(dig1 + \beta) & 0 \\ \sin(dig1 + \beta) & \cos(dig1 + \beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.52}$$

Boom Rotation Matrix II, RB_2

$$RB_2 = \begin{bmatrix} \cos(dig1 - \beta) & -\sin(dig1 - \beta) & 0 \\ \sin(dig1 - \beta) & \cos(dig1 - \beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.53}$$

4.4.3 Stick Rotation Matrix, RS

Since the axis of the stick is not parallel to the axis of the second section of the boom, the transformation matrix developed for the second part of the boom cannot be directly used to transform global forces into the frame of reference local to the stick.

The rotation matrix for the stick is formulated by carefully observing the subsequent chain of angular transformations starting from joint $J1$.

$$\tan(J9_L) = \frac{h_{stick}}{h_{tail}} \tag{4.54}$$

$$J9_L = \tan^{-1}\left[\frac{h_{stick}}{h_{tail}}\right] \tag{4.55}$$

$$J9_U = J9L \tag{4.56}$$

$$J9 = J9_U + J9_L = 2 \times J9_L \tag{4.57}$$

$$TS_2 = \sqrt{l_2^2 - h_{stick}^2} \tag{4.58}$$

$$J2_l = \tan^{-1}\left[\frac{h_{tail}}{h_{stick}}\right] \tag{4.59}$$

$$J2_r = \tan^{-1}\left[\frac{TS_2}{h_{stick}}\right] \tag{4.60}$$

$$J2 = J2_l + J2_r \tag{4.61}$$

$$J3_l = tan^{-1}\left[\frac{h_{stick}}{TS_2}\right] \tag{4.62}$$

$$J3_U = J3_L \tag{4.63}$$

$$J3 = J3_L + J3_U = 2 \times J3_L \tag{4.64}$$

$$R_{Z,(\beta+dig1)} \rightarrow R_{Z,-2\beta} \rightarrow R_{Z,J2} \rightarrow R_{Z,(J9_L-180)} \tag{4.65}$$

$$\begin{aligned} net\ angle\ of\ rotation &= \beta + dig1 - 2\beta + J2 + J9_L - 180 \\ &= dig1 - \beta + J2 + J9_L - 180 \end{aligned} \tag{4.66}$$

The stick rotation matrix RS is then given by the expression:

$$RS = \begin{bmatrix} cos(dig1 - \beta + J2 + J9_L - 180) & -sin(dig1 - \beta + J2 + J9_L - 180) & 0 \\ sin(dig1 - \beta + J2 + J9_L - 180) & cos(dig1 - \beta + J2 + J9_L - 180) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.67}$$

4.4.4 Transition Four-bar Dimensional Synthesis

The major linear dimensions defining the working range of the overall mechanism have already been synthesized using the method developed in the previous chapter. For the purpose of making the mechanism complete, the transition four-bar mechanism's dimensions have to be synthesized for a given dimensions of standard buckets and sticks.

For a given dimension of the bucket b_o, the length of the other two linkages of the transition four-bar mechanism can be calculated as follows.

As shown in Figure 4-14, there are two configurations of the four-bar linkage mechanism resulting in a phenomenon called "*mechanism lock*". These two angular positions are considered to be the upper and lower range limits within which the bucket operates.

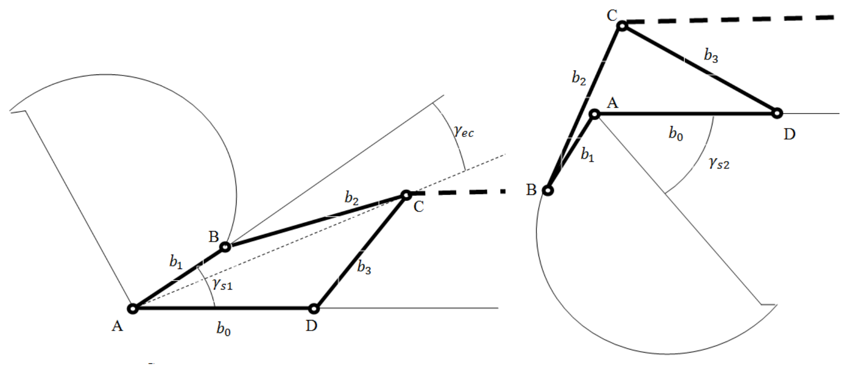


Figure 4-14 Transition four-bar work-ranges

In Figure 4-14 (left) , γ_{ec} is the eccentricity angle necessary to prevent the mechanism from self locking. γ_{s1} in Figure 4-14 (left), is the minimum angle between the back of the bucket and the stick. The value of γ_{s1} depends on the safety clearance angle necessary to avoid physical contact between the links. The value of this angle is subject to change during the life of the design cycle reflecting changes in the dimensions of the contacting parts as a result of cyclic modifications.

γ_{s2} serves the same purpose as γ_{s1} but on the opposite end of the bucket's angular displacement range. Limiting factors in this case include direct contact between mechanical components and volumetric allowance for extra bulk material when loading the bucket. In addition to b_o and b_1, these two angles are assumed to be known to evaluate the lengths of links b_2 and b_3.

The critical values of b_2 and b_3, i. e., those that result in mechanical locking of the mechanism, are determined from the following two simplified geometries corresponding to the two cases as shown by Figure 4-15 and Figure 4-16.

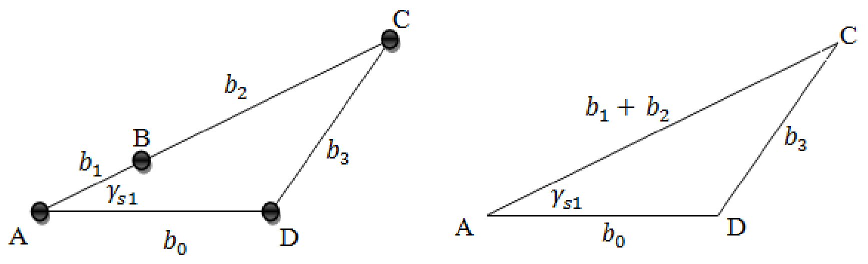


Figure 4-15 Upper mechanism locking configuration

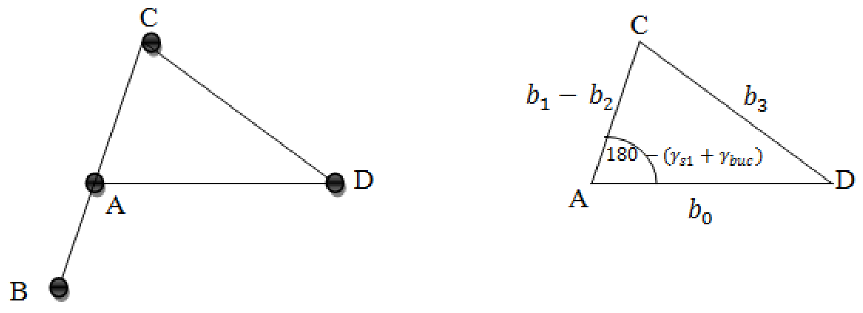


Figure 4-16 Lower mechanism locking configuration

Applying Law of Cosines to the geometry of Figure 4-15 we get

$$b_3{}^2 = b_o{}^2 + (b_1 + b_2)^2 - 2b_0(b_1 + b_2)\cos\gamma_{s1} \tag{4.68}$$

Similarly referring to Figure 4-16

$$b_3{}^2 = b_o{}^2 + (b_2 - b_1)^2 + 2b_0(b_2 - b_1)\cos\ (\gamma_{s2} + \gamma_{bk}) \tag{4.69}$$

Solving Eq. (4.68) and Eq. (4.69) simultaneously gives the values of b_2 and b_3.

This calculation is implemented using the custom MATLAB® function `f_Four-bar_Solver` in the main program.

4.4.5 Stress and Strength Calculations

The expressions for the various stresses considered in this section are developed based on the assumptions and procedures outlined by Shigley and Mischke [44].

(1) Transverse Shear Stress, τ_b

Shear stress, τ, as a result of shear force and bending moment is derived from the relation

$$V = \frac{dM}{dx} \tag{4.70}$$

$$\tau = \frac{VQ}{Ib} \tag{4.71}$$

Where Q is the first moment of area about the neutral axis given by:

$$Q = \int_{y1}^{C} y\, dA \tag{4.72}$$

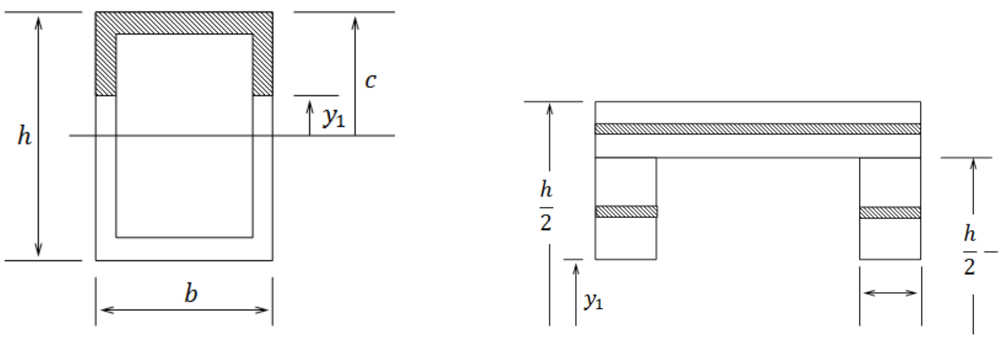


Figure 4-17 Cross-sectional area under transverse shear stress

Q for the given cross-sectional geometry is computed by dividing the area into two sections

Case 1

For $\quad 0 < |y_1| < (\frac{h}{2} - t)$

$$Q = \int_{y_1}^{h/2} y\, dA = \int_{y_1}^{\frac{h}{2}-t} y\, dA_1 + \int_{\frac{h}{2}-t}^{\frac{h}{2}} y\, A_2 \tag{4.73}$$

Where $\mathrm{dA}_1 = 2\mathrm{tdy}$ and $\mathrm{dA}_2 = \mathrm{bdy}$

$$Q_1 = \int_{y_1}^{\frac{h}{2}-t} 2tydy + \int_{(\frac{h}{2}-t)}^{\frac{h}{2}} bydy \tag{4.74}$$

$$Q_1 = t\left(\frac{h}{2} - t\right)^2 - t{y_1}^2 + \frac{bt(h-t)}{2} \tag{4.75}$$

Case 2

For $\quad (\frac{h}{2} - t) \le |y_1| \le \frac{h}{2}$

$$Q_2 = \int_{y_1}^{h/2} ydA_2 = \int_{y_1}^{h/2} bydy \tag{4.76}$$

$$Q_2 = \frac{b}{8}\,(h^2 - 4y_1^2) \tag{4.77}$$

The second moment of area of the entire cross-section, I, is given by

$$Q_{max} = t(\frac{h}{2} - t)^2 + \frac{bt(h-t)}{2} \tag{4.78}$$

$$I = \frac{bh^3}{12} - \frac{(b-2t)(h-2t)^3}{12} \tag{4.79}$$

$$\tau_{max} = \frac{VQ_{max}}{Ib} \tag{4.80}$$

$$b_{max} = 2t \tag{4.81}$$

$$\tau_b = \frac{V\left[t(\frac{h}{2} - t)^2 + \frac{bt(h-t)}{2}\right]}{2t\left[\frac{bh^3}{12} - \frac{(b-2t)(h-2t)^3}{12}\right]} \tag{4.82}$$

(2) Torsional Stress, τ_{tor}

The equations expressing torsional stresses in the cross-sectional areas are developed based on the assumptions and procedures outlined by Shigley and Mischke [44].

Area of torsion, A_{tor}, is given by the following expression

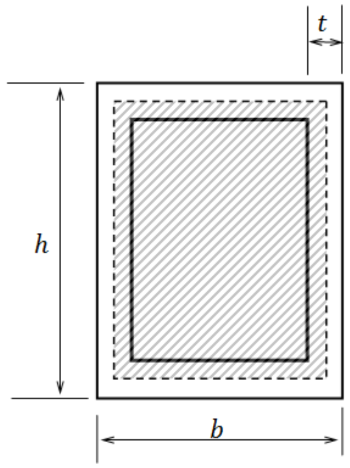


Figure 4-18 Torsional cross-sectional area

$$A_{tor} = \left(b - 2\frac{t}{2}\right)\left(h - 2\frac{t}{2}\right) \tag{4.83}$$

$$A_{tor} = (b-t)(h-t) \tag{4.84}$$

The torsional stress, τ_{tor}, is given by

$$\tau_{tor} = \frac{T}{2(b-t)(h-t)t} \tag{4.85}$$

Where T, the torque creating the torsional stress, is recorded at every joint during the simulation of the mechanism in the SimMechanics environment

(3) Direct Stress, σ_{dx}
Area of direct stress distribution is given by the expression

$$A_{dx} = 2[bt + t(h - 2t)] \tag{4.86}$$

For a given axial longitudinal force, F_{ax}, the direct stress is calculated by using the formula

$$\sigma_{dx} = \frac{F_{ax}}{A_{dx}} \tag{4.87}$$

$$\sigma_{dx} = \frac{F_{ax}}{2[bt + t(h - 2t)]} \tag{4.88}$$

(4) Bending Stresses, σ_{zx}
Bending stress, σ_{zx}, due to bending moment acting about the z-axis, M_z, is given by:

$$\sigma_{zx} = -\frac{M_z y}{I_{zz}} \qquad \left(-\frac{h}{2} \le y \le \frac{h}{2}\right) \tag{4.89}$$

Where I_{zz} is the second moment of area of the cross-section about the centroidal z-axis and it is given by:

$$I_{zz} = \frac{bh^3 - (b - 2t)(h - 2t)^3}{12} \tag{4.90}$$

This stress reaches its maximum value when at the outer most boundaries of the cross section, i. e., when:

$$y = \pm\frac{h}{2} \tag{4.91}$$

$$\sigma_{zx(max)} = -\frac{6\, M_z h}{bh^3 - (b - 2t)(h - 2t)^3} \tag{4.92}$$

In a similar manner, bending stress, σ_{yx}, duet to moment acting about the y-axis is given by:

$$\sigma_{yx} = -\frac{M_y z}{I_{yy}} \qquad \left(-\frac{b}{2} \le z \le \frac{b}{2}\right) \tag{4.93}$$

Where I_{yy} in this case is the second moment of the cross-sectional area about the centroidal y-axis

$$I_{yy} = \frac{hb^3 - (h - 2t)(b - 2t)^3}{12} \tag{4.94}$$

$$\sigma_{yx(max)} = -\frac{6\,M_z b}{hb^3 - (h-2t)(b-2t)^3} \tag{4.95}$$

Superposition of the effects of these two bending stresses and the direct stress gives the maximum value of stress in the x- direction.

$$\boldsymbol{\sigma_{xx(max)} = \sigma_{dx} + \sigma_{yx(max)} + \sigma_{zx(max)}} \tag{4.96}$$

4.4.5 Pin Design

(1) Methods of Pin Failure

1. Localized contact stress
2. Failure of pin due to double shear
3. Failure of pin due to bending moment

(2) Contact Stresses

The interaction between the pin and the casing is modeled by a cylinder-plane contact instead of cylinder-cylinder contact. The resulting magnitudes of Hertz's contact stresses will have relatively higher values than if they were calculated using the cylinder-cylinder assumption because of the reduced contact area.

Where

d_1	Diameter of pin
d_2	Diameter of base hole
ν_1	Poisson's Ratio of the pin material
ν_2	Poisson's Ratio of the base material
E_1	Young's Modulus of Elasticity of the pin material
E_2	Young's Modulus of Elasticity of the base material

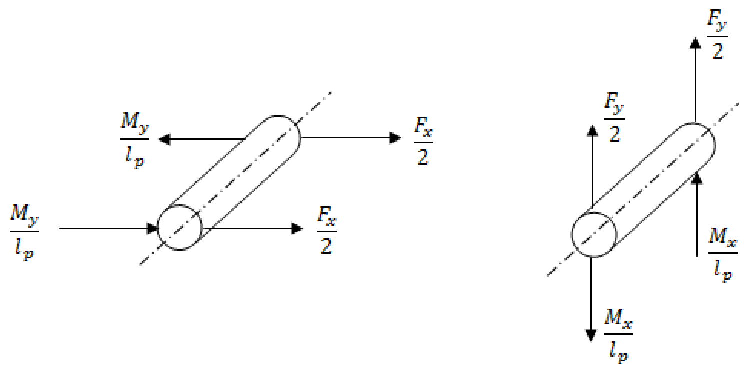


Figure 4-19 Pin loads in global vertical and horizontal planes

The total force exerted on the first end of the pin is given by:

$$\boldsymbol{F}_p = \begin{bmatrix} \frac{F_x}{2} + \frac{M_y}{l_p} \\ \frac{F_y}{2} + \frac{M_x}{l_p} \\ 0 \end{bmatrix} \tag{4.97}$$

Where d_p *and* l_p are the pin diameter and its effective length, respectively

The magnitude of this force, F_{pin}, is:

$$\boldsymbol{F}_p = \sqrt{\left(\frac{F_x}{2} + \frac{M_y}{l_p}\right)^2 + \left(\frac{F_y}{2} + \frac{M_x}{l_p}\right)^2} \tag{4.98}$$

(3) Hertz's Contact Stress

The maxim stress due to contact between the surfaces of the pin and the base is calculated using the Hertz's method. The contact zone between these two surfaces is approximated by a rectangular region. The width of this region, commonly known as "contact half width" is calculated by using the formula:

$$b = k_b \sqrt{F_p l_p} \tag{4.99}$$

$$Where \qquad k_b = \sqrt{\frac{2}{\pi t} \frac{\left(\frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2}\right)}{\frac{1}{d_1} + \frac{1}{d_2}}} \tag{4.100}$$

Since the casing holes are modeled by a plane for the purpose of making the design compatible with higher stresses, the reciprocal term containing d_2 will be approximated by zero. The resulting expression takes the form:

$$k_b = \sqrt{\frac{2d_1\left(\frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2}\right)}{\pi t}} \tag{4.101}$$

The maximum contact pressure is then written as a function of the length (l_p) and diameter (d_p) of the pin as follows:

$$P_{max}(l_p, d_p) = \frac{2F_p l_p}{\pi b t} \tag{4.102}$$

The resulting principal stresses in the pin are given by the relations

$$\sigma_x = -2\nu_1 \frac{2F_p l_p}{\pi b t}\left[\sqrt{1+\left(\frac{z}{b}\right)^2} - \left|\frac{z}{b}\right|\right] \tag{4.103}$$

$$\sigma_y = -\frac{2F_p l_p}{\pi b t}\left[\frac{1+2\left(\frac{z}{b}\right)^2}{\sqrt{1+\left(\frac{z}{b}\right)^2}} - 2\left|\frac{z}{b}\right|\right] \tag{4.104}$$

$$\sigma_z = -\frac{2F_p l_p}{\pi b t}\frac{1}{\sqrt{1+2\left(\frac{z}{b}\right)^2}} \tag{4.105}$$

To calculate the maximum value of stress from one of these three equations, these stress are computed at the critical section $z/b = 0.786$

$$\sigma_x = -1.944\, \nu_1 \frac{F_p l_p}{\pi b t} \tag{4.106}$$

$$\sigma_y = -0.371\, \frac{F_p l_p}{\pi b t} \tag{4.107}$$

$$\sigma_z = -1.572\, \frac{F_p l_p}{\pi b t} \tag{4.108}$$

The allowable contact stress is generally taken as the minimum of $\sigma_y/4$ or $\sigma_u/6$.

(4) Failure of Pin Due to Double Shear

The total shear area of each pin is represented by the equation

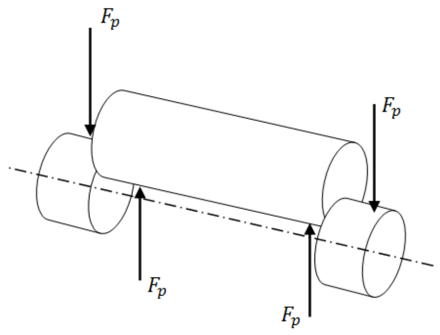


Figure 4-20 Pin under double shear

$$A_{sh} = 2\,\frac{\pi d_p^2}{4} \tag{4.109}$$

Direct shear stress on the pin is calculated by dividing the shearing force by the total area. An equation for this stress is derived in terms of the pin length and the pin diameter so that it can be used in calculating an optimum values for these two pin dimensions.

$$\tau_{sh} = \frac{2|F_p|}{A_{sh}} \tag{4.110}$$

$$\tau_{sh} = \frac{4|F_p|}{\pi d_p^2} \tag{4.111}$$

(5) Failure of Pin Due to Bending Moment

Referring to Figure 4-21 for the loading distribution, the maximum bending moment at the mid-span of the pin is given by the expression:

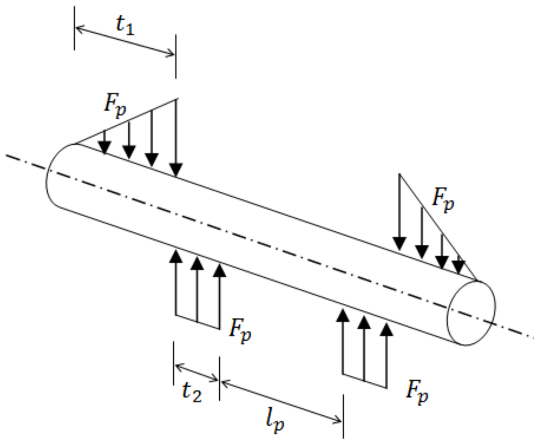


Figure 4-21 Bending load distribution on pins

$$t_1 = t + e_{b1} \tag{4.112}$$

$$t_2 = t + e_{b2} \tag{4.113}$$

$$M_{max} = F_p\left(\frac{t_1}{3} + t_2 + \frac{l_p}{2}\right) - F_p\left(\frac{t_2}{2} + \frac{l_p}{2}\right) \tag{4.114}$$

Where e_{b1} and e_{b2} are base reinforcement extensions.

The maximum bending stress is then given by substituting the above expression into the general formula

$$\sigma_b = \frac{M_{max}}{z} \tag{4.115}$$

$$\sigma_b = \frac{F_p\left[\frac{t_1}{3} + \frac{t_2}{2}\right]}{\frac{\pi}{32} d_p^3} \tag{4.116}$$

Chapter 5

Virtual Mechanism Modeling and Simulation

5.1. Introduction

The design process of linkages and members in the mechanism requires the identification of generalized reaction forces at each joint. These forces will be used in the free body diagrams (FBD) and mass-acceleration diagrams (MAD) during the design stages. This task is implemented by using SimMechanics®, the mechanical simulation and analysis tool available in MATLAB® software.

5.2. Simulation Setup

Now that the major linear dimensions of the mechanism are identified, the next step is determining the reaction forces and moments experienced by each joint as a result of the digging operation. To do this, a skeleton mechanism is constructed using the previously calculated linear dimensions in the SimMchanics modeling environment of MATLAB®. Doing this provides the flexibility of calculating interaction forces and exporting the results into the workspace for further processing. Furthermore, during latter stages of the design cycles, the weightless links can be substituted by their 3D counterparts and the simulation can be re-run to include the inertia effects into considerations.

When using this tool, the mechanism under investigation is constructed by connecting linear linkages of prescribed inertia properties with virtual mechanical joints available in the joint library. The virtual weld joint is used in the event it is required to model bending, splitting, or merging mechanical members.

In order to this approach be applicable in the automation of design processes, two general requirements have to be met. The first requirement is that all linear dimensions need to be defined; either numerically or symbolically. The second one is that all forms of mating constraints between connecting members have to be imposed by the use of joints and limits on joint parameters.

Although inertia properties are possible to be assigned to these bodies at the initial stages as mentioned above, the procedure adopted in this paper uses a different approach suitable for cyclic modifications.

The first round of the simulations starts with weightless and high-stiffness rigid bodies and upcoming simulations will be carried out with updated 3D solid bodies produced as a result of previous design cycles.

The necessary kinetic and kinematic joint variables registered during the simulations are extracted to MATLAB® workspace using the Simulink® scope readers. SimMechanics Link®, another useful tool to import and export CAD solid models into and out of SimMchanics® simulation environment, supports only SolidWorks® and Pro/E® but not UG NX. To overcome this setback SolidWorks is used as an intermediate file transfer tool in exporting the 3D models to SimMechanics.

5.3. Simulink Model Construction

Figure 5-1 shows the major components of an excavator arm mechanism while Figure 5-2 shows the representation of the same mechanism by linear components. The latter model is constructed in SimMechanics environment using standard rigid body and joint library.

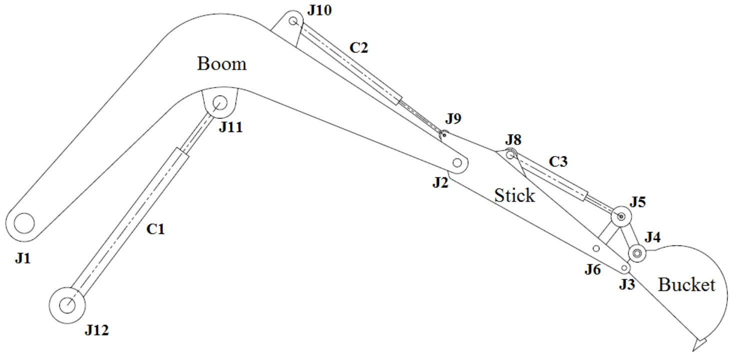


Figure 5-1 Typical excavator arm mechanism

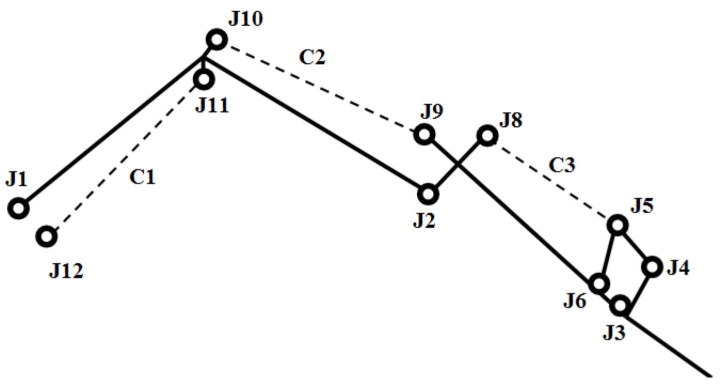


Figure 5-2 Skeleton representation of excavator arm mechanism

5.3.1 Boom Construction

The boom as shown in Figure 5-3, has two sides; $BS_1$and BS_2. It is hinged to the vehicle body with joint J1 and to the stick with joint J2. Joints J10 and J11 are connecting points for hydraulic cylinders C2 and C1, respectively. The deflection of the structure by an angle 2β is modeled by welding the two linear sides.

The two hinges for the hydraulic cylinders, **J10** and **J11**, are located at the end of the extension sticks *h_J10* and *h_J11* to represent for an initial thickness of the boom. The values of these lengths are updated at the end of each conceptual design cycle from the cross-sectional calculation results. This is done because the hinge locations are assumed to be on the surfaces of the boom which are subject to modification at the end of each conceptual design cycle.

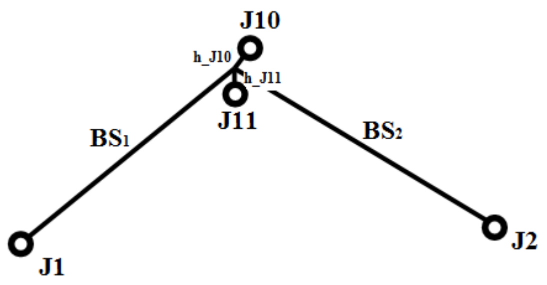


Figure 5-3 Skeleton representation of boom structure

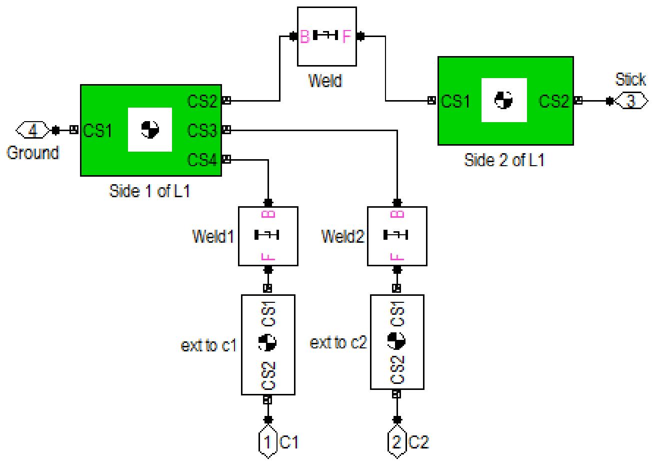


Figure 5-4 SimMechanics model of boom

5.3.2 Stick Construction

The stick has four joints. J2 connects it with the boom while J3 connects it with the bucket. The transition four-bar mechanism is connected with the stick at joint J6 with revolute joint. Joint J8 is the connection point for the second hydraulic cylinder, C2.

Again in this case, the final distances of hinges J2 and J8 from the longitudinal axis of the stick, h_J2 and h_J8, are determined based on the final 3D dimensions of the stick. To start the simulation, however, initial values are assigned for these dimensions. These parameters will be updated at the end of each cycle.

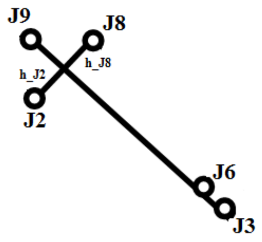


Figure 5-5 Skeleton representation of stick

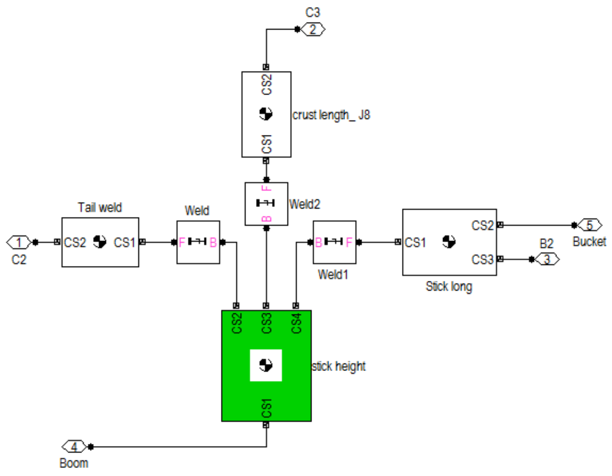


Figure 5-6 SimMechanics model of stick

5.3.4 Bucket Modeling

The bucket, which is the follower link of the transition four-bar mechanism, has only two joints; J3 and J4 to connect it to the stick and the coupler link of the four-bar, respectively.

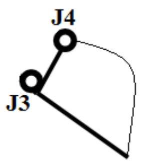


Figure 5-7 Bucket schematics

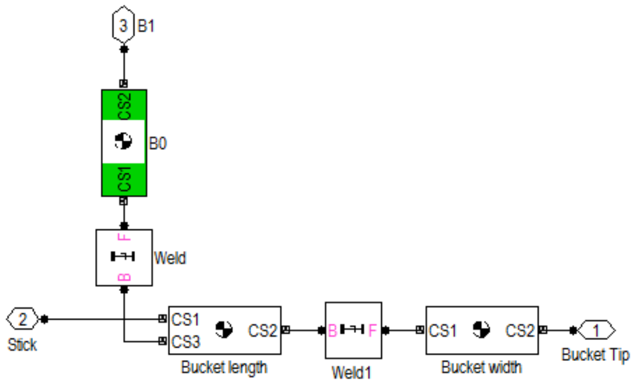


Figure 5-8 Bucket SimMechanics model

The application point of the ground reaction force is selected in such a way the overall mechanism will be subject to severe loading condition. Twisting moment about the x-axis and bending moments about the y and z- axes register maximum readings when the digging force is applied on the bucket at a point furthest from the x-axis. An eccentricity loading distance of half the width of the bucket is introduced for this purpose as shown in Figure 5-9.

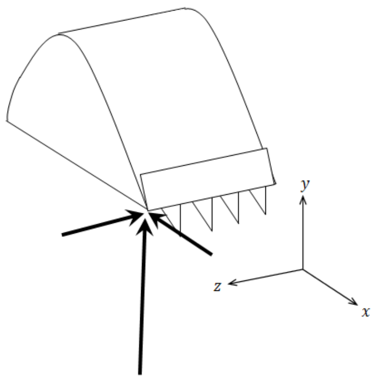


Figure 5-9 Location of application point of digging force

5.3.5 Hydraulic Cylinders

Hydraulic cylinders are represented by simple weightless rigid links only for the purpose of keeping the mechanism rigid. The forces registered at the opposite ends of these links can be used to determine the required hydraulic capacity of the cylinders. However, this task is beyond the scope of this research and will not be discussed here.

5.3.6 Transition Four-bar Linkages

The other two remaining linkages in the transition four-bar mechanism are represented by simple blocks with joints at both ends.

The actual total number of revolute joints in the mechanism is 11. However, in the SimMechanics model there is one more joint needed to be introduced due to the location of the hydraulic force application point on the driving link of the transition four-bar. Two coaxial joints in the real mechanism are required to be modeled by combining them into a single joint. Because this point is chosen to be at the connection point of the driving and coupler links, an additional redundant hinge joint was required for creating a three branch connection. This representation will not have any negative effect on the final outcome of the analysis.

The above SimMechanics sub-models are assembled and simulation environment parameters are defined. Figure 5-10 below shows the final assembled SimMechanics Model of the excavator arm mechanism in the real-time simulation window.

The simulation for this model is run for 2 sec at the digging orientation. Scopes in the model such as $SD1, SD2, \ldots, SD12$ register and export joint variable data to the MATLAB® workspace in vector form.

Because of the selection of the digging mode for the design purpose, it was not necessary to define angular displacement and speed limits.

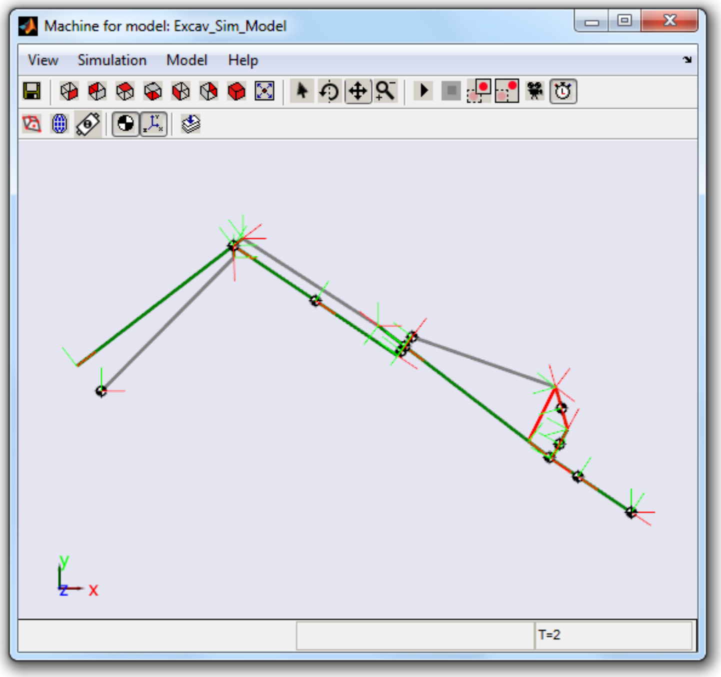


Figure 5-10 Real-time simulation environment

Figure 5-11 shows the assembled SimMechanics diagram of the excavator arm mechanism. The digging force is represented by a constant vector and is applied at the left tip of the bucket.

5.4. Numerical Example

An example problem is investigated in this section to demonstrate the applicability of the proposed methods. The methods and the necessary engineering rules and design intents are programmed and implemented in MATLAB®. This demonstration will be carried out in two stages designed to show the operational procedures of the two proposed methods and their corresponding results. The first stage deals with the application of the hybrid ANN-Optimization technique in the process of dimensional synthesis of mechanisms. In the second section, the calculations and involved and the generated results in the areas of optimizations of cross sectional dimensions of the boom and the stick will be discussed.

The input to the module handling the dimensional synthesis is a set of the required output configurations of the excavator arm mechanism. As discussed before, these values are checked for compatibility to make sure the feasibility of their co-existence.

Input Problem:

```
SpcDat = c_Spec_Data_SI

  Properties:
       Title: 'Commercial Specifications and Vehicle Dimensions'
    Maximum_Reachout_at_Ground_Level_S1: 5.6700
              Maximum_Cutting_Height_S3: 3.7248
              Maximum_Loading_Height_S4: 1.3521
                  Horizontal_Distance_H: 0.9857
                    Vertical_Distance_V: 1.2300
                         Vehicle_Weight: 5000
```

Once the linear dimensions of the overall mechanism are calculated, the next task will be calculating the optimum cross-sectional dimensions of the boom and the stick. This process started by defining necessary geometric and non geometric constraints. In addition to the constraints, initial values for some variables and iteration-dependent dimensions are also initiated at this stage.

Bucket geometric properties:

```
BuckGeo =  c_Bucket_Geo_SI

  Properties:
       Title: 'Bucket Geometries and Dimensions'
```

```
                        Bucket_Length_l3: 0.8112
                         Bucket_Width_BW: 0.4867
                        Bucket_Height_b0: 0.2839
                     Bucket_Pin_Width_bw: 0.2434
                Bucket_Angle_teta_bucket: 95
             Bulk_Volume_Clearance_Angle: 40
    Maximum_Upward_Bucket_Open_Limit_Angle: 35
```

Dimensional Constraints:

```
Dimensional_Constraints =    c_Dimensional_Constraints_SI

  Properties:
       Title: 'Structural Dimensions Constraints'
                 Minimum_Plate_Thickness: 0.0070
                 Maximum_Plate_Thickness: 0.0200
                  Minimum_Base_Dimension: 0.1000
                  Maximum_Base_Dimension: 0.5000
           Minimum_Boom_and_Stick_Height: 0.0100
                     Maximum_Boom_Height: 0.5000
                    Maximum_Stick_Height: 0.5000
     Extension_of_Boom_Pin_Reinforcement: 0.0140
    Extension_of_Stick_Pin_Reinforcement: 0.0140
```

Material Properties:

```
MaterProp = c_Material_Properties_SI

  Properties:
       Title: 'Material Selection and Properties'
                    Prpertiy: 'Poisons E  YS_kpsi TS_kpsi YS_MPa
TS_MPa Elong_2% Area_% BHN'
    Pin_Material: [0.3000 210 234 260 1612 1791 12 43 498]
   Base_Material: [0.3000 210 26 47 179 324 28 50 95]
     Allowable_Stress_in_Pin: 447750000
            Poison_Ratio_Pin: 0.3000
          Youngs_Modulus_Pin: 2.1000e+011
    Allowable_Stress_in_Base: 81000000
           Poison_Ratio_Base: 0.3000
         Youngs_Modulus_Base: 2.1000e+011
           Safety_Factor_Pin: 1.1500
          Safety_Factor_Boom: 1.1500
         Safety_Factor_Stick: 1.1500
      Safety_Factor_Linkages: 1.1500
```

Initial Variable Linkage Imitation:

```
InitialParam = c_Initial_Parameters_SI

  Properties:
       Title: 'Initial Values for Variable Dimensions'
    Distance_to_J2_and_J8_on_Stick: 0.1000
```

```
           Distance_to_J10_on_Boom: 0.1000
           Distance_to_J11_on_Boom: 0.1000
```

Linkage Geometries:

```
LinkDims =  c_LinkDims_SI

  Properties:
        Title: 'Boom and Stick Dimensions'
           Boom_Shortcut_Length_l1: 2.7126
       Boom_Deflection_Angle_betta: 35.6055
             Side_Length_of_Boom_T: 1.6682
                   Stick_Length_l2: 1.5616
                    Stick_Angle_J2: 156.9267
                           J2_left: 70.5982
                          J2_right: 86.3285
                    Stick_Angle_J8: 156.9267
                           J8_left: 70.5982
                          J8_right: 86.3285
                    Stick_Angle_J9: 38.8037
                             J9_up: 19.4018
                          J9_lower: 19.4018
                    Stick_Angle_J3: 7.3429
                             J3_up: 3.6715
                          J3_lower: 3.6715
    Distance_to_J2_and_J8_on_Stick: 0.1000
           Distance_to_J10_on_Boom: 0.1000
           Distance_to_J11_on_Boom: 0.1000
                 Stick_Tail_Length: 0.2839
              Stick_Forward_Length: 1.5584
```

Transition Four-bar Dimensions:

```
FourbarDims = c_Fourbar_Solver_SI

  Properties:
       Title: 'Fourbar Linkage Dimensions'
    Fourbar_Link_b0: 0.2839
    Fourbar_Link_b1: 0.3692
    Fourbar_Link_b2: 0.4461
    Fourbar_Link_b3: 0.2434
```

Operational Configuration Matrices and Variables:

```
OperConfig = c_Operational_Configuration_SI

  Properties:
  Title: 'Configuration Parameters and Rotational Matrices'
         Boom_opertating_angle_dig1: 1.8619
    Boom_Rotational_Matrix_Sec1_RB1: [3x3 double]
    Boom_Rotational_Matrix_Sec2_RB2: [3x3 double]
         Stick_Rotational_Matrix_RS: [3x3 double]
                     Fourbar_teta_1: 72.8748
                     Fourbar_teta_2: -36.7587
```

```
                    Fourbar_teta_3: 278.6715
```

Generalized Joint Forces and Moments:

```
Joint_Forces = c_Joint_Forces_SI

  Properties:
       Title: 'Generalized Forces on Joints'
    JointForces: [12x7 double]
         FORCES: ''
             F1: [3x1 double]
             F2: [3x1 double]
             F3: [3x1 double]
             F4: [3x1 double]
             F5: [3x1 double]
             F6: [3x1 double]
             F7: [3x1 double]
             F8: [3x1 double]
             F9: [3x1 double]
            F10: [3x1 double]
            F11: [3x1 double]
            F12: [3x1 double]
        MOMENTS: ''
             M1: [3x1 double]
             M2: [3x1 double]
             M3: [3x1 double]
             M4: [3x1 double]
             M5: [3x1 double]
             M6: [3x1 double]
             M7: [3x1 double]
             M8: [3x1 double]
             M9: [3x1 double]
            M10: [3x1 double]
            M11: [3x1 double]
            M12: [3x1 double]
```

Force and moment values can be extracted by calling the members of the data structure as follows:

```
Joint_Forces.F5

ans =

  1.0e+004 *

    0.5644
   -1.9908
         0
```

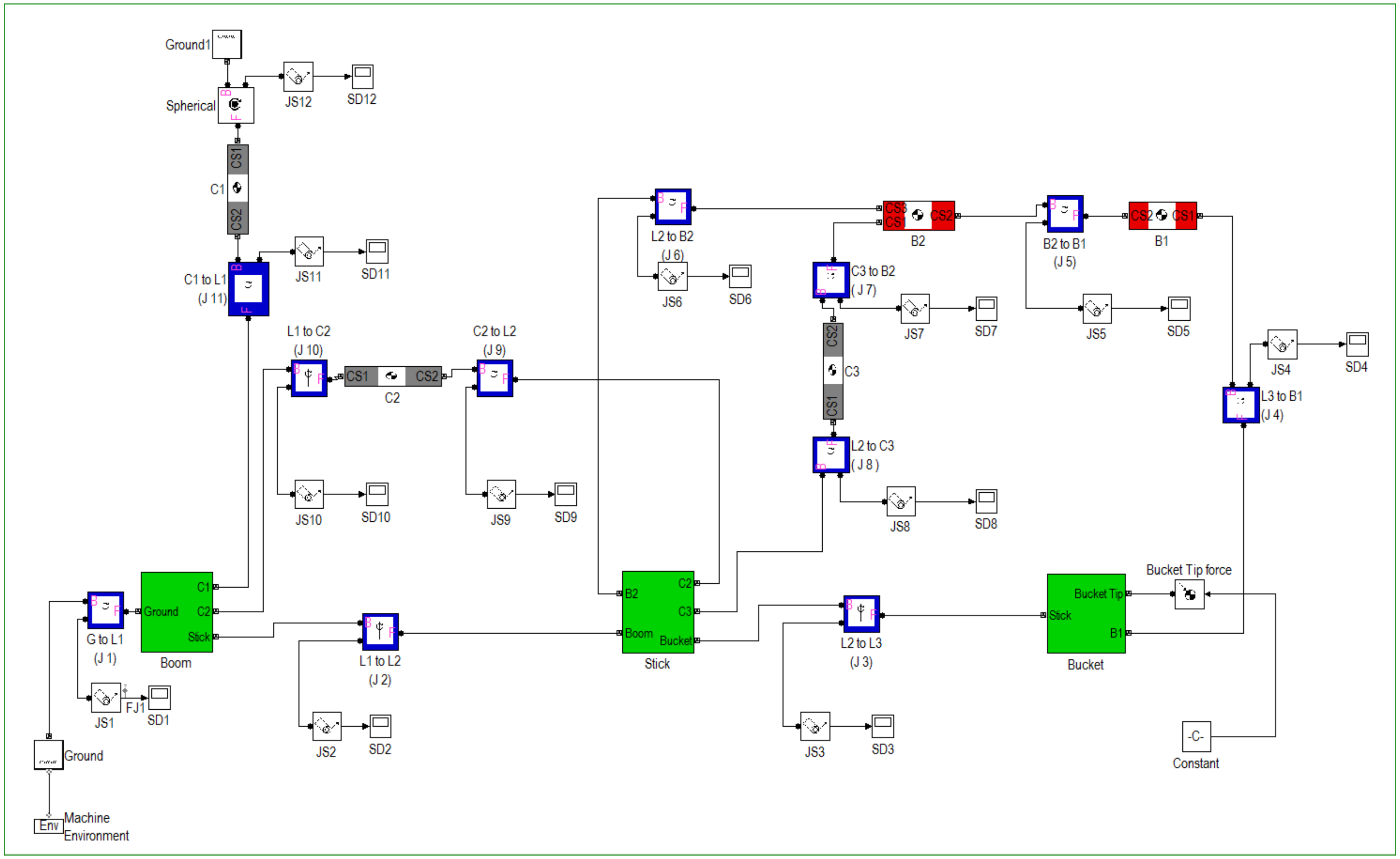


Figure 5-11 SimMechanics diagram for an excavator arm mechanism

Chapter 6

Feature-based CAD Embodiment

6.1. Introduction

The Application Programming Interface (API) open platform is used to write program codes and generate the feature based 3D CAD parts of the boom and stick in NX. The programming part is implemented using Visual Studio 2008® C++.

The results of the engineering design calculations carried out in previous sections using MATLAB® and SimMechanics® are needed to be imported in a systematic manner to be used in the generation of the CAD models. Additionally, the process of importing and exporting data was required to be performed without direct manual involvement. This was accomplished by creating intermediate sets of text data files to bridge the gap.

At the end of the engineering design cycle calculations, a MATLAB® program is used to create or update a set of *.dat** text files containing the necessary input dimensions and parameters data structures. The MATLAB® program writes/updates these files and stores them in specific directories created for this purpose.

These files will automatically be accessed by the API C++ code during the generation of the CAD models. In a similar way done for the exporting command in MATLAB®, a C++ program is developed which is responsible for reading the values of this files and storing them in the internal memory.

The choice of the locations of the shared directories was done taking into consideration the possibility of different sections of the tasks could be performed on different systems. One of the free internet file storage services was used to create a common directory shared by two computers involved in this research. In practical industrial applications, this approach lends itself to the implementation of efficient collaborative project. It provides the flexibility of assigning different tasks to different engineers working in different geographical locations.

Classes and their corresponding objects are instantiated and used to effectively handle the data imported. Most of these data were used in the program more than once and adopting object oriented programming approach proved helpful in managing the data. The following lines show a class for handling custom datum coordinate system (CSYS) creating function parameters.

```
struct DATUM_CSYS_DATA{
      double offset_x;
      double offset_y;
      double offset_z;
      double angle_x;
      double angle_y;
      double angle_z;
      bool transform_sequence;
      int rotation_sequence[2];
      };
```

6.2. Reusability of Functions

All the functions developed for this project are created by the use of the C++ API functions provided in NX open documentations. Direct application the basic functions to this research was found to be very difficult and time consuming because of the need to specifically define most initializing parameters unrelated to the objective of this work.

An effort has been put in place to generalize most of the developed functions and ensure their reusability. Based on the basic C++ API functions, customized functions were developed by incorporating additional procedures to bring user intuitivism while simplifying the definitions of input and output arguments.

More than 40 functions were developed and used in the creation of the boom and the stick CAD files. The details of these functions are provided in Appendix B. The following are lists of some of the tasks these functions are responsible for.

- Reading external data files
- Creating new part model files in specified directories
- Creation of datum CSYS (Absolute and Relative)
- Creation of datum planes
- Extraction of datum planes/axis out of datum CSYS
- Creation of geometric objects such as
    - Points
    - Lines
    - Arcs
    - B-spline Curves from data points

6.3. Boom Modeling

The modeling of the boom part is initialized by creating a blank NX .prt file using the function wub_Create_New_Part_File(char file_path[UF_CFI_MAX_FILE_NAME_SIZE])

The next step after creating a blank CAD modeling environment was to properly position user defined CSYS features for the purpose of simplicity in additional features and object creation. The relative angular orientations and offset distances between consecutive CSYSs were represented by instantiating an object of the class DATUM_CSYS_DATA. In addition to relative linear displacements and angular orientations, these objects also define the coordinate transformation sequences.

In the case study most of the CSYSs were defined and located at the joint locations for the purpose of simplifying creation of joint associative features such as hinges. The custom functions used for this purpose are:

- `wub_CSYS_origin_and_direction(void)`
-
```
wub_CSYS_offset(  tag_t referece_datum_CSYS,
                  const double linear_offset[3],
                  const double angular_offset[3],
                  bool operation_sequence)
```

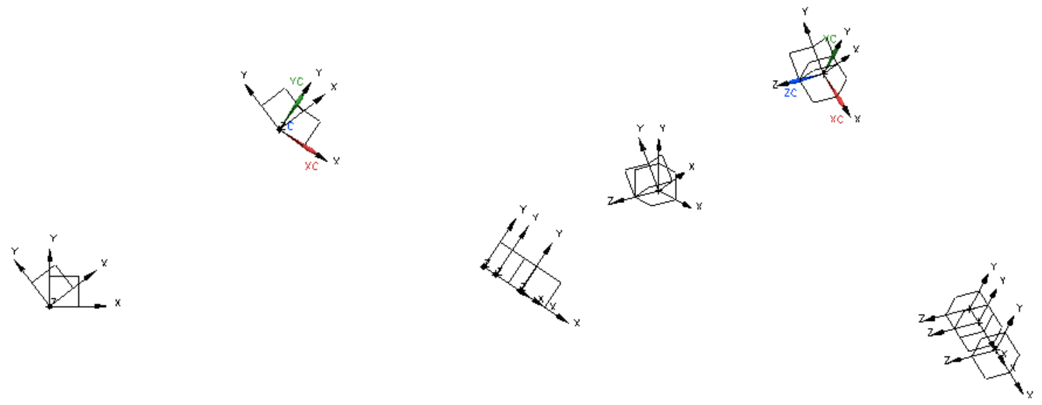


Figure 6-1 Boom coordinate system (CSYS) features

The optimization result vectors for the two sides of the boom exported from MATLAB® were previously saved in the following directory:

```
D:\\My Dropbox\\ForNX\\Delimited Files\\Boom_vec1_Original.dat
D:\\My Dropbox\\ForNX\\Delimited Files\\Boom_vec2_Original.dat
```

These vectors define point coordinates of the top left edge of the boom. B-spline curves representing each side of the boom were created from these data points by importing within their respective CSYS.

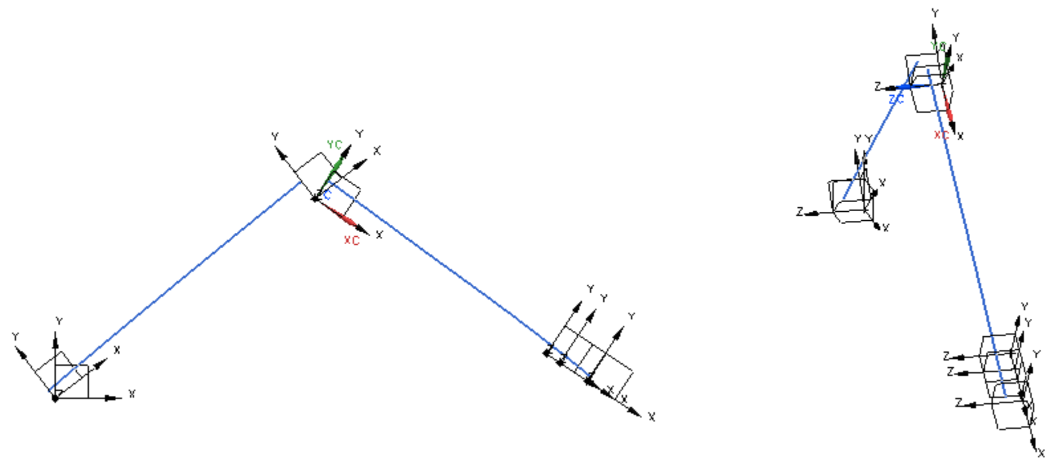


Figure 6-2 Boom B-spine curve features

Since these edges are symmetrical about the local $x - y$ and x-z planes, the curves defining the lower right edges of the boom are created by reflecting the existing curves about their $x - z$ planes within the local CSYS.

The function used for this purpose is:

```
wub_Mirror_a_Curve_through_a_Plane(tag_t Curve_Tag, tag_t Plane_ Tag)
```

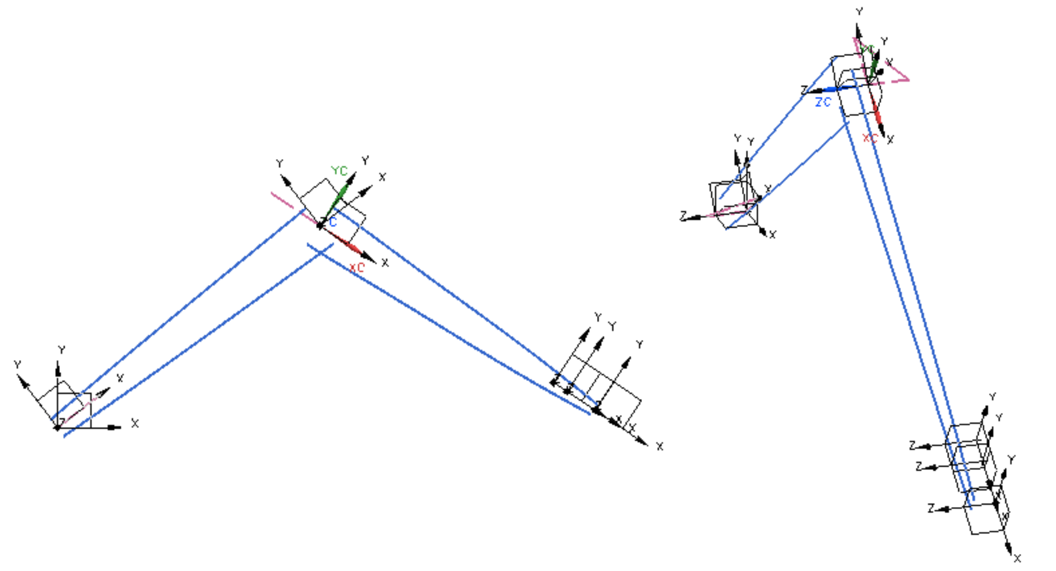


Figure 6-3 Evolvement of features

The curves are modified by trimming and bridging operations to create a joined curve feature. The following functions are used to for these operations.

```
tag_t wub_Trim_Curve_by_Datum_Plane(tag_t Curve_Tag,
                                    tag_t Datum_Plane_Tag,
                                    int which_end);
```

```
tag_t wub_Trim_Curve_by_Curves(tag_t Target_curve_Tag,
                               tag_t Tool_curve1,
                               tag_t Tool_curve2);

tag_t wub_Bridge_Curves(tag_t Curve_1_Tag,
                        tag_t Curve_2_Tag,
                        int Reverse1,
                        int Reverse2,
                        int par1,
                        int par2);
```

The end of the boom at joints J1 and J2 are closed by arcs tangent to the upper and lower edge curves and centered at the origins of the CSYSs.

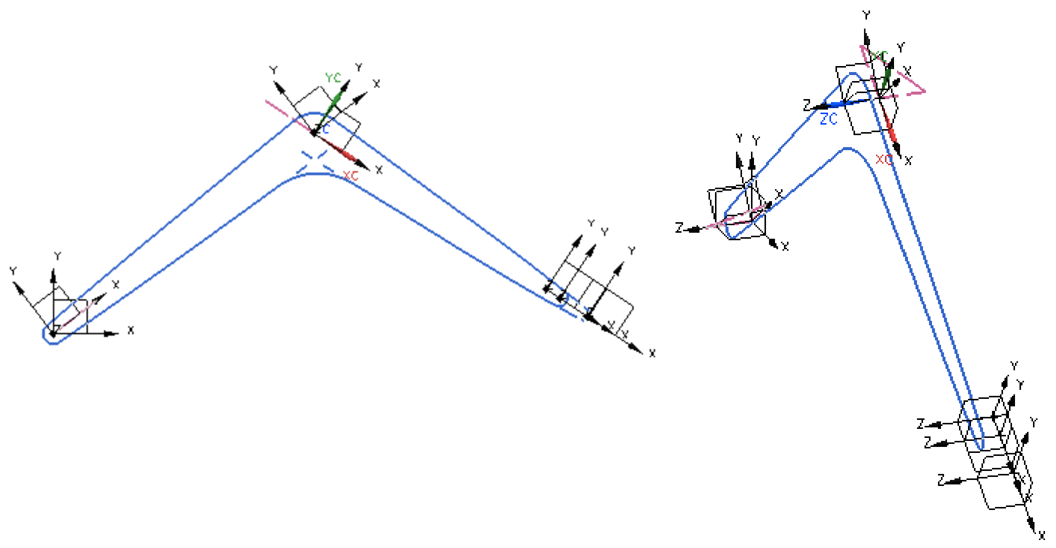


Figure 6-4 Joined curve features

This closed curve feature represents the side wall of the boom. To create the upper and lower floors of the boom it is required to create other sets of curve features defining the boundaries of the surfaces. The above modified closed curve, shown by the green line in Figure 6.5, is projected onto the vertical x-y plane to form a guide line to be used for surface sweeping operation together with the existing one. This projected curve will serve the purpose of defining the right section of the boom as seen from the –x directions.

```
tag_t wub_Create_Projected_Curve(tag_t curve_tag,
                                 tag_t Datum_CSYS_tag,
                                 int Plane_Num)
```

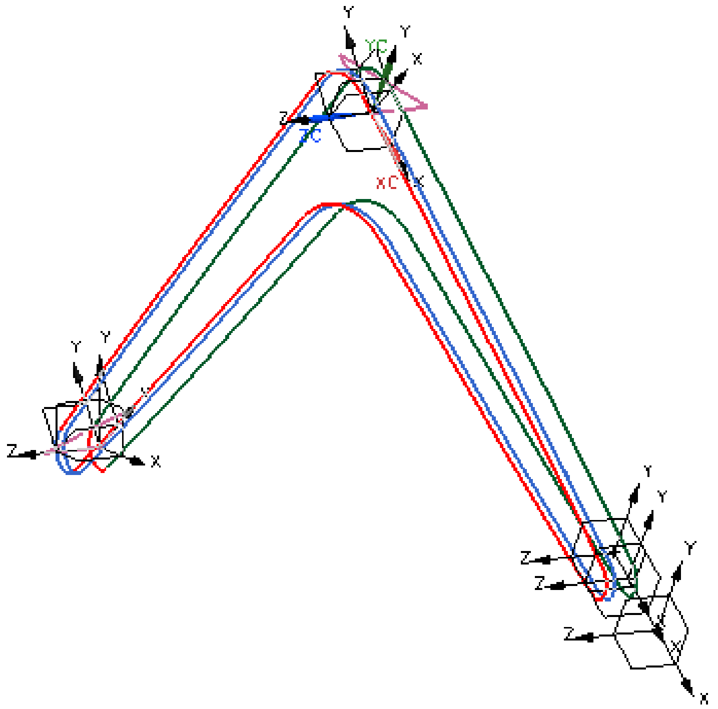


Figure 6-5 Embodiment features

The third closed curve, colored green in Figure 6-5, is created in a very similar procedure followed for the creation of the above curve but with an offset value added in the z direction to accommodate for a welding space.

The top and bottom floor surface features of the boom are generated by sweeping a linear curve guided by the red and the green curves. To avoid potential modeling errors associated with availability of multiple solutions for a given input to the responsible function, this process was carried out in two stages. The side wall surface was created from bounding curves. The functions used for this purpose are:

```
tag_t wub_Join_Curves(tag_t *curves,int n)
tag_t wub_Point_from_Spline(tag_t curve_Tag, int line_end)
tag_t wub_Lines_from_two_points(tag_t point1, tag_t point2)
tag_t wub_SWEEP_2_guides(tag_t Guide_s1, tag_t Guide_s2, tag_t Section)
tag_t wub_BPLANE(tag_t Curve_String[2])
```

Hinge joint features are created by sketching and extruding their profiles. The boom has two types of hinge joints; one that passes thought the plate walls and the other one that is attached externally to the boom structure by welding.

Joint J1 is constructed by introducing a hollow cylindrical feature of appropriate dimensions to the boom structure while joints J2, J10, and J12 are constructed from scratch by sketching and extruding their profile.

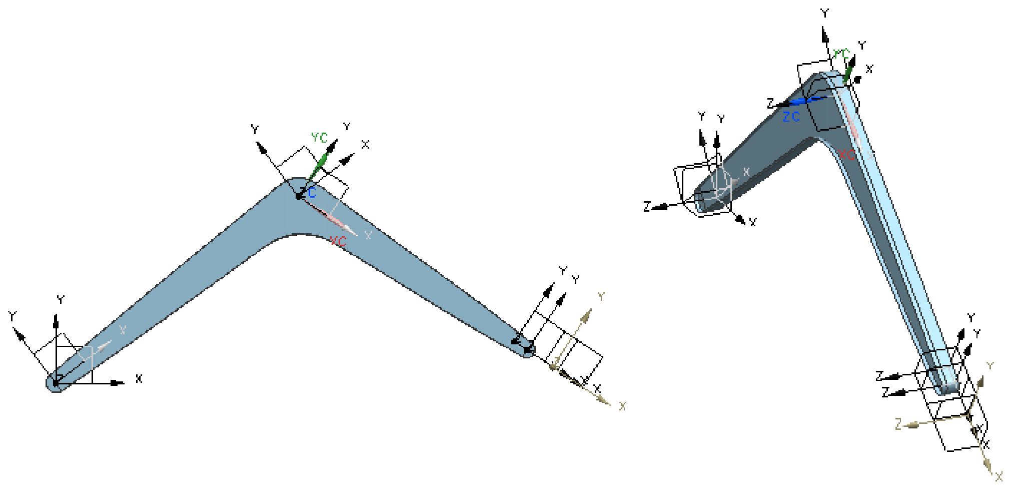


Figure 6-6 Sheet body features

Functions used for this purpose include:

```
tag_t wub_SKETCHES_J2_adopter(char name[30],
                              tag_t Refrence_CSYS,
                              int Plane_num,
                              int Axis_num)

tag_t wub_ARC_on_sketch(tag_t Reference_CSYS)

tag_t wub_ARC_Center_Radius(tag_t Reference_CSYS,
                              int Plane_num,
                              double radius,
                              double arc_center[3],
                              double start_ang,
                              double end_ang)

tag_t wub_ARC_Point_Point_Radius(tag_t Reference_CSYS,
                              int Plane_num,
                              double radius,
                              double arc_center[3],
                              tag_t p1,
                              tag_t p2)

tag_t wub_Extrude(tag_t Connected_Curve, char* limit[2])

tag_t wub_SKETCH_J11(tag_t Reference_CSYS,
                  int Plane_num,
                  tag_t line_Tag,
                  tag_t bridge_tag)

tag_t  wub_SKETCH_J12(tag_t  Reference_CSYS,int  Plane_num,tag_t
Curve_Tag);
```

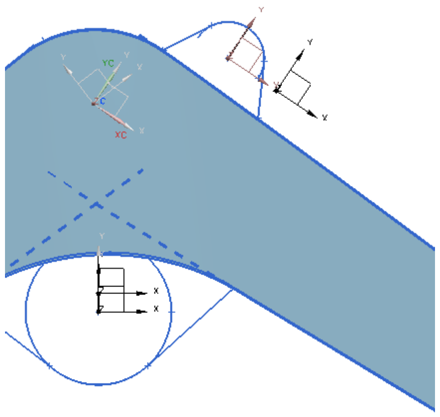


Figure 6-7 Hinge joint profiles construction

The sheet bodies are thickened and the joint sketches are extruded with initial and final limits to create the final solid body features. After the necessary modification on the joint solid features the left side of the solid boom is created by merging the individual solids with each other.

Custom functions used for these operations include:

```
tag_t THICKEN_Sheet(tag_t sheet_body_tag)
tag_t wub_UNITE_SOLIDS(tag_t Target_Solid, tag_t Tool_Solid)
```

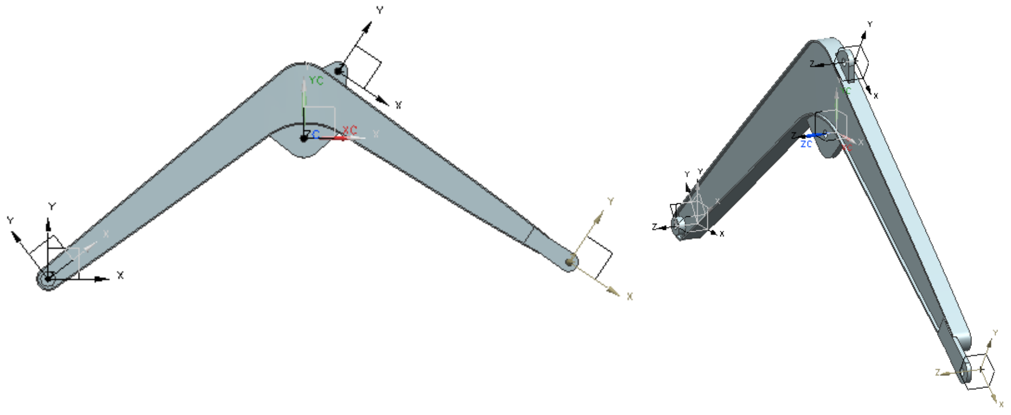


Figure 6-8 Solid body features

This left side of the boom is mirrored about the x-y plane to create the other half of the boom. The original and the newly created halves are then merged together to create the final boom solid body shown by Figure 6-9.

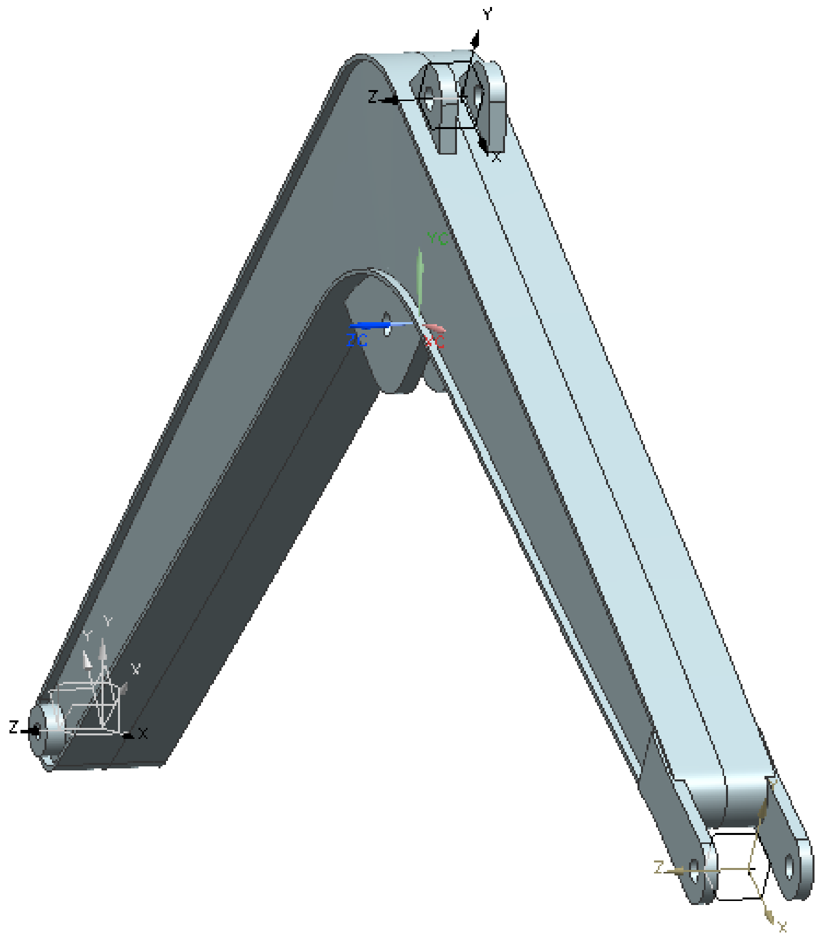


Figure 6-9 Final CAD model of an excavator boom

6.4 Stick CAD Modeling

The programming and part creation procedures followed for the stick are very similar to the one adopted for the boom. Most of the functions developed are reused directly or, in some instances, with minor modifications to address stick-specific modeling needs.

As done for the boom, the modeling process for the stick started by creating a new part file called Stick.prt using the same function.

Data was imported from the intermediate files using similar procedures. User defined CSYS's were created at the joint locations and some critical locations necessary for the creation of sketches.

Generally, the stick construction procedure is relatively easier than that of the boom because of the parallelism of the stick CSYS features (Figure 6-10).

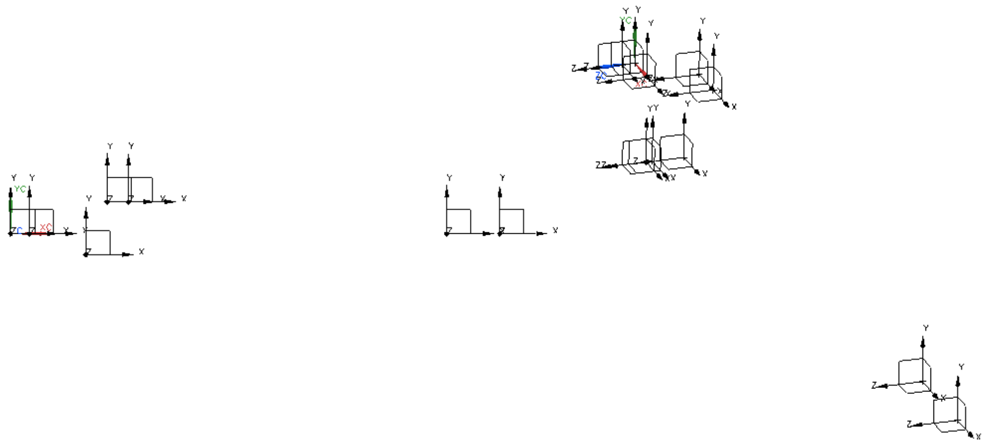


Figure 6-10 Stick coordinate system features

The arcs of the stick at the joints J2, J3, and L9 were constructed first based on the data imported from the neutral text files. Profiles defining the edges of the stick were created by joining these arcs with the maximum middle point straight tangent lines as shown in Figure 6-11.

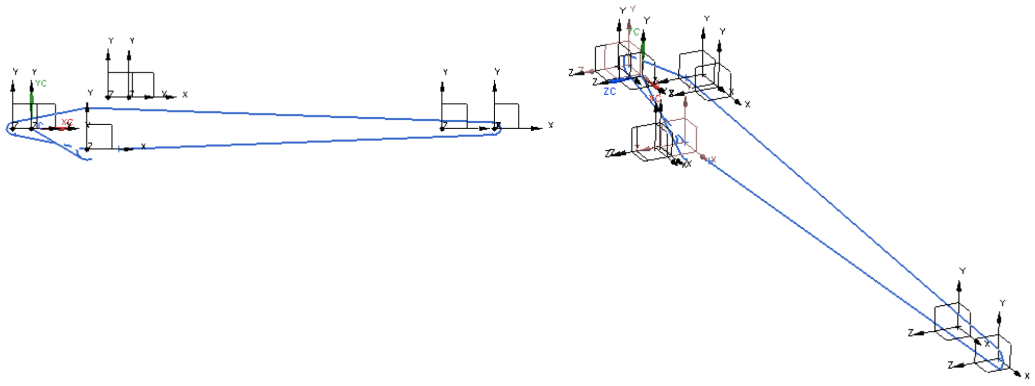


Figure 6-11 Feature evolvement

After performing some line modification operations, such as trimming and joining, the created closed loop curves are projected onto two different planes positioned parallel to the middle x-y plane.

Figure 6-12 shows the cleaned and projected stick profile curves. The green and blue curves will be used as guides to create a sheet body by a sweeping operation while the red curve will be used as a boundary when creating a bounded plane. The pink closed curve will be extruded to create the hinge solid for joint J9.

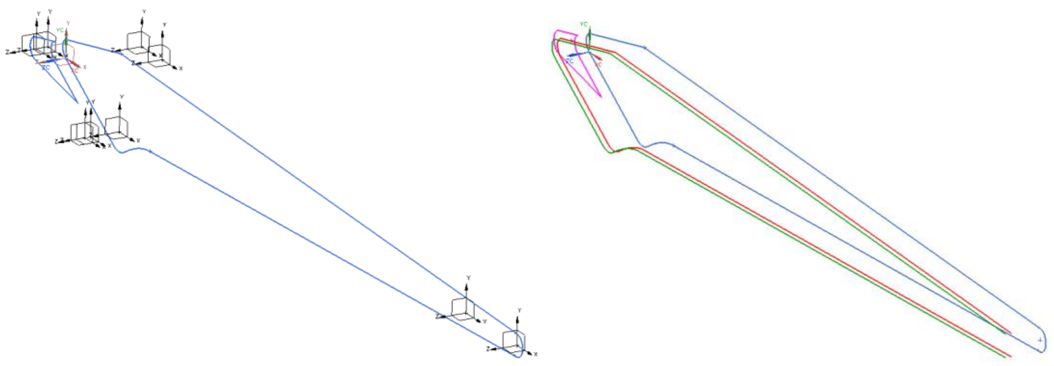


Figure 6-12 Embodiment features

A line element parallel to the z axis was created and used for the sweeping operation. The following figures show the sweeping tool and the resulting sheet body features.

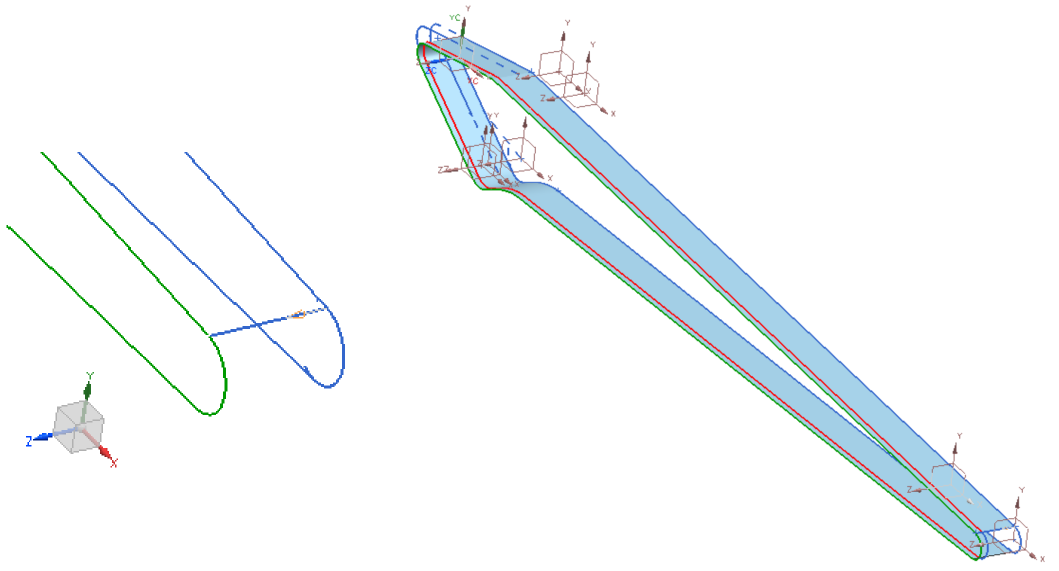


Figure 6-13 Stick sheet body features

The side left side wall is created by using the other projected curve as a boundary in the bounding plane operation.

These planes are thickened with appropriate thickness values and taking necessary inference tolerances into consideration. Joints are created using similar procedure as used in the boom modeling. The final left half of the boom is shown in Figure 6-14.

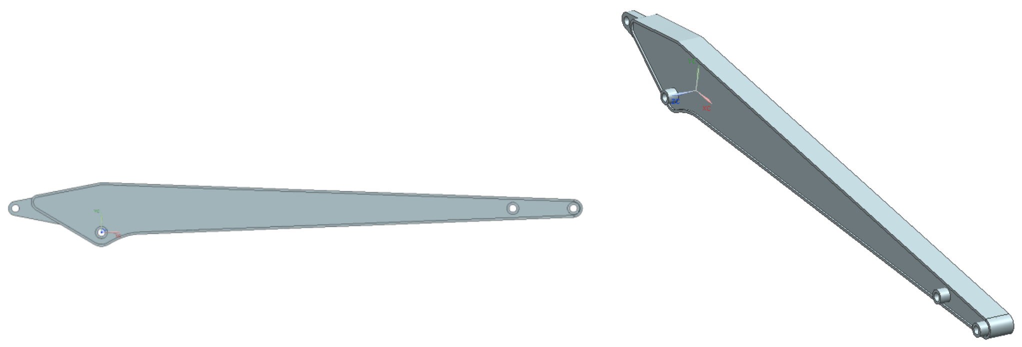


Figure 6-14 Stick solid body feature

The final complete stick solid body feature is created by merging individual extruded and thickened solid features together into a single part and mirroring this part about the x-y plane. The mirrored and its parent solid are converted again into single solid by merging them with similar command to get the final model shown by Figure 6-15.

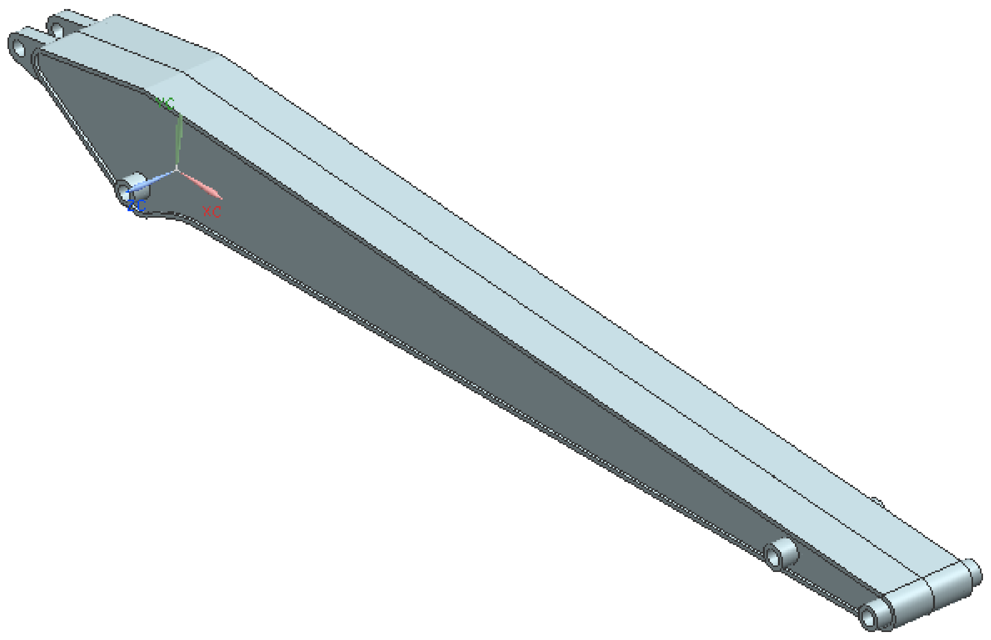


Figure 6-15 Final CAD mode of an excavator arm stick

Chapter 7

Conclusions and Future Works

7.1 Conclusions

7.1.1 Proposed Method

The proposed generative feature-based conceptualization method for product development is promising to upgrading the current information-fragmented and experience-based practice.

The method discussed in this work can effectively capture engineering rules and facilitate the design cycle processes, such as insightful configuration optimization and embodiment development. The method also provides good flexibility in terms of customization and standardization of other products which involve frequent changes.

Reusability of the developed functions has provided evidence that, unlike traditional modeling methods, the knowledge in the design stages can always be embedded, harnessed for a new product, and be reused when developing future generations of similar products.

However, it is worth noting that in the candidate’s opinion, regardless how intelligent a design system is to be in the future, human design expertise is always required while the developed system can only support the decision making more effectively with some productivity tools.

7.1.2 Case Study

The case study proves that the knowledge-driven feature-based conceptual design approach can handle traditionally-known and complex challenges such as machine linkage optimization problems.

The proposed hybrid optimization-ANN dimensional synthesis method has greatly increased the reliability of calculated solutions. Training the ANN with larger size of existing product data is believed to produce solutions reflective of design intents and industrial standards.

A hybrid ANN-Optimization method has been proposed and proved to provide satisfactory result. The optimization procedure, specific to a selected product configuration, was employed to calculate the final linkage dimensions which satisfy work-range configuration requirements.

The ANN has the advantage to generate close initial solutions of the linkage dimensions and, in combination with the optimization techniques, it can assist to generate accurate optimal solutions in an integrated design environment.

7.1.3 Scalability

Although this research work was applied to an excavator case only, the method proposed can be equally applied to other mechanical product development. Product-specific engineering rules, data and procedures have to be replaced.

This method has some novelty in solving similar problems in the product design domain, i.e. a hybrid linkage dimension synthesis method. The general procedure can be followed for the conceptual design of other mechanisms in principle. All the process modules will remain valid. The only exception would be the details of design calculations and optimization criteria since they are usually very specific to the product under discussion. Regardless of the product being designed, the procedure is scalable.

7.2 Future Work

Formal definitions of generic conceptual design features need to be investigated such that embedded engineering rules, constraints, data representations, behaviors can be modeled and managed generically in an object-oriented approach and systematically implemented.

Programming and engineering analysis tools such as Visual C++ and Matlab can be integrated with feature-based tools, e.g. Siemens NX so that the analysis procedure can be part of the integrated conceptual design system. Their input and output as well as the constraints can automatically be managed according to the formal definition of concept problems.

The conceptual design process discussed was based only on the mechanical design aspect of the case-study. The data structures and communication mechanisms can also be equally used in designing other aspects of the product. The conceptual level design of hydraulic circuit subsystem in the case study, for example, can also be modeled and solved under proposed data and information management scheme by implementing its own conceptualization contents.

References

[1] Alizade, R. I., and Kilit, O., "Analytical Synthesis of Function Generating Spherical Four-Bar Mechanism for the Five Precision Points," *Mechanism and Machine Theory*, vol. 40(7) pp. 863-878, 2005.

[2] Barton, M., Shragai, N., and Elber, G., "Kinematic Simulation of Planar and Spatial Mechanisms using a Polynomial Constraints Solver," *Computer-Aided Design and Applications,* vol. 6(1) pp. 115-123, 2009.

[3] Basak, H., and Gulesin, M., "A Feature Based Parametric Design Program and Expert System for Design," *Mathematical and Computational Applications,* vol. 9(3) pp. 359-370, 2004

[4] Bronsvoort, W. F., Bidarra, R., van, d. M., "The Increasing Role of Semantics in Object Modeling," *Computer-Aided Design and Applications,* vol. 7(3) pp. 431-440, 2010.

[5] Danjou, S., Lupa, N., and Koehler, P., "Approach for Automated Product Modeling using Knowledge-Based Design Features," *Computer-Aided Design and Applications,* vol. 5(5) pp. 622-629, 2008.

[6] Dixon, A., and Shah, J. J., "Assembly Feature Tutor and Recognition Algorithms Based on Mating Face Pairs," *Computer-Aided Design and Applications,* vol. 7(3) pp. 319-333, 2010.

[7] Durupt, A., Remy, S., and Ducellier, G., "KBRE: A Knowledge Based Reverse Engineering for Mechanical Components," *Computer-Aided Design and Applications,* vol. 7(2) pp. 279-289, 2010.

[8] Durupt, A., Remy, S., and Ducellier, G., "Knowledge Based Reverse Engineering - an Approach for Reverse Engineering of a Mechanical Part," *Journal of Computing and Information Science in Engineering,* vol. 10(4) 2010.

[9] Erkaya S. and Uzmay I., “A neural–genetic (NN–GA) Approach for Optimizing Mechanisms Having Joints with Clearance,” *Multibody System Dynamics,* vol. 20 pp. 69-83, 2008

[10] Frimpong, S., and Li, Y., "Virtual Prototype Simulation of Hydraulic Shovel Kinematics for Spatial Characterization in Surface Mining Operations," *International Journal of Surface Mining, Reclamation and Environment,* vol. 19(4) pp. 238-250, 2005.

[11] Gao, Z., Zhang, D., and Ge, Y., "Design Optimization of a Spatial Six Degree-of-Freedom Parallel Manipulator Based on Artificial Intelligence Approaches," *Robotics and Computer-Integrated Manufacturing,* vol. 26(2) pp. 180-189, 2010.

[12] Hasan, A. T., Hamouda, A. M. S., Ismail, N., "Trajectory Tracking for a Serial Robot Manipulator Passing through Singular Configurations Based on the Adaptive Kinematics Jacobian Method," *Proceedings of the Institution of Mechanical Engineers.Part I: Journal of Systems and Control Engineering,* vol. 223(3) pp. 393-415, 2009.

[13] Jensen, O. F., and Hansen, J. M., "Dimensional Synthesis of Spatial Mechanisms and the Problem of Non-Assembly," *Multibody System Dynamics,* vol. 15(2) pp. 107-133, 2006.

[14] Laribi, M. A., Romdhane, L., and Zeghloul, S., "Analysis and Dimensional Synthesis of the DELTA Robot for a Prescribed Workspace," *Mechanism and Machine Theory,* vol. 42(7) pp. 859-870, 2007.

[15] Larsen, S., and Jensen, C. G., "Converting Topology Optimization Results into Parametric CAD Models," *Computer-Aided Design and Applications,* vol. 6(3) pp. 407-418, 2009.

[16] Li, M., Zhang, Y. F., and Fuh, J. Y. H., "Retrieving Reusable 3D CAD Models using Knowledge-Driven Dependency Graph Partitioning," *Computer-Aided Design and Applications,* vol. 7(3) pp. 417-430, 2010.

[17] Lourenco, D., Oliveira, P., Noort, A., "Constraint Solving for Direct Manipulation of Features," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM,* vol. 20(4) pp. 369-382, 2006.

[18] Ma, Y., Britton, G. A., Tor, S. B., "Associative Assembly Design Features: Concept, Implementation and Application," *International Journal of Advanced Manufacturing Technology,* vol. 32(5-6) pp. 434-444, 2007.

[19] Ma, Y., Tang, S., Au, C. K., "Collaborative Feature-Based Design Via Operations with a Fine-Grain Product Database," *Computers in Industry,* vol. 60(6) pp. 381-391, 2009.

[20] Ma, Y., Chen, G., and Thimm, G., "Change Propagation Algorithm in a Unified Feature Modeling Scheme," *Computers in Industry,* vol. 59(2-3) pp. 110-118, 2008.

[21] Mantyla, M., Nau, D., and Shah, J., "Challenges in Feature-Based Manufacturing Research," *Communications of the ACM,* vol. 39(2) pp. 77-85, 1996.

[22] Myung, S., and Han, S., "Knowledge-Based Parametric Design of Mechanical Products Based on Configuration Design Method," *Expert Systems with Applications,* vol. 21(2) pp. 99-107, 2001.

[23] Ong, S. K., and Shen, Y., "A Mixed Reality Environment for Collaborative Product Design and Development," *CIRP Annals - Manufacturing Technology,* vol. 58(1) pp. 139-142, 2009.

[24] Prasanna, L., Guhanathan, S., and Agrawal, R., "Automated Approach for Designing Components using Generic and Unitized Parametric Sketches," *Computer-Aided Design and Applications,* vol. 7(1) pp. 109-123, 2010.

[25] Pratt, M. J., and Anderson, B. D., "A Shape Modelling Applications Programming Interface for the STEP Standard," *CAD Computer Aided Design,* vol. 33(7) pp. 531-543, 2001.

[26] Riou, A., and Mascle, C., "Assisting Designer using Feature Modeling for Lifecycle," *CAD Computer Aided Design,* vol. 41(12) pp. 1034-1049, 2009.

[27] Singh, D. K., and Jebaraj, C., "Feature-Based Design for Process Planning of the Forging Process," *International Journal of Production Research,* vol. 46(3) pp. 675-701, 2008.

[28] Singla, E., Tripathi, S., Rakesh, V., "Dimensional Synthesis of Kinematically Redundant Serial Manipulators for Cluttered Environments," *Robotics and Autonomous Systems,* vol. 58(5) pp. 585-595, 2010.

[29] Solazzi, L., "Design of Aluminium Boom and Arm for an Excavator," 2010.

[30] Ter, H.,A.H.M., Lippe, E., and Van, d. W., "Applications of a Categorical Framework for Conceptual Data Modelling," *Acta Informatica,* vol. 34(12) pp. 927-927, 1997.

[31] Thakur, A., Banerjee, A. G., and Gupta, S. K., "A Survey of CAD Model Simplification Techniques for Physics-Based Simulation Applications," *CAD Computer Aided Design,* vol. 41(2) pp. 65-80, 2009.

[32] van, d. M., and Bronsvoort, W. F., "Tracking Topological Changes in Parametric Models," *Computer Aided Geometric Design,* vol. 27(3) pp. 281-293, 2010.

[33] van, d. M., and Bronsvoort, W. F., "Modeling Families of Objects: Review and Research Directions," *Computer-Aided Design and Applications,* vol. 6(3) pp. 291-306, 2009.

[34] Vasiliu, A., and Yannou, B., "Dimensional Synthesis of Planar Mechanisms using Neural Networks: Application to Path Generator Linkages," *Mechanism and Machine Theory,* vol. 36(2) pp. 299-310, 2001.

[35] Verdes, D., Stan, S., Manic, M., "Kinematics analysis, workspace, design and control of 3-RPS and TRIGLIDE medical parallel robots," *2009 2nd Conference on Human System Interactions, HSI '09, May 21, 2009 - May 23,* 2009, pp. 103-108.

[36] Vossoughi, G. R., Abedinnasab, M. H., and Aghababai, O., "Introducing a New 3 Legged 6-DOF Ups Parallel Mechanism for Pole Climbing Applications," *WSEAS Transactions on Systems,* vol. 6(1) pp. 221-228, 2007.

[37] Wang, H., Zhou, X., and Qiu, Y., "Feature-Based Multi-Objective Optimization Algorithm for Model Partitioning," *International Journal of Advanced Manufacturing Technology,* vol. 43(7-8) pp. 830-840, 2009.

[38] Wang, Q., Li, J., Wu, B., "Live Parametric Design Modifications in CAD-Linked Virtual Environment," *International Journal of Advanced Manufacturing Technology,* vol. 50(9-12) pp. 859-869, 2010.

[39] Wong, L. M., and Wang, G. G., "Development of an Automatic Design and Optimization System for Industrial Silencers," *Journal of Manufacturing Systems,* vol. 22(4) pp. 327-339, 2003.

[40] Wu, J., Purwar, A., and Ge, Q. J., "Interactive Dimensional Synthesis and Motion Design of Planar 6R Single-Loop Closed Chains Via Constraint Manifold Modification," *Journal of Mechanisms and Robotics,* vol. 2(3) 2010.

[41] Ye, X., Liu, H., Chen, L., "Reverse Innovative Design - an Integrated Product Design Methodology," *CAD Computer Aided Design,* vol. 40(7) pp. 812-827, 2008.

[42] Yoon, J., and Manurung, A., "Development of an Intuitive User Interface for a Hydraulic Backhoe," *Automation in Construction,* vol. 19(6) pp. 779-790, 2010.

[43] Zhao, Y., Tang, Y., and Zhao, Y., "Dimensional synthesis and analysis of the 2-UPS-PU parallel manipulator," *1st International Conference on Intelligent*

Robotics and Applications, ICIRA 2008, October 15, 2008 - October 17, 2008, pp. 141-151.

[43] Shigley J. E., and Misheke C. R., 2001. Mechanical Engineering Design. McGraw-Hill Higher Education, Inc., New York, N.Y.

Appendix 1 MATLAB CODES

A1.1 General Excavator Design Code (Main Body)

Developed by:

Abiy T Wubneh
Department of Mechanical Engineering
University of Alberta
2010

The following code is the main part of the linkage synthesis computation. It is assembled from individual functions and subroutines to compute different tasks involved in the conceptual design of the excavator arm mechanism. Dimensional synthesis of members and optimization of their cross-sectional area are the main objective of this code. Please refer to the documentation of the respective functions in the next section (Section 2) for details on the individual function components used in this code.

```
clear all;
clc;
clf;
close all;
```

Global Variables

The variables defined here are used by more than one function.

```
global SpcDat BuckGeo Dimensional_Constraints MaterProp InitialParam ...
       LinkDims Four-barDims OperConfig Joint_Forces PinDims
```

Specifications and Constraints

User inputs, material data, and engineering constraints are provided by the following three functions.

```
[SpcDat,BuckGeo] = f_Specification()
[Dimensional_Constraints] = f_Dimensional_Constraints()
[MaterProp] = f_Material_Properties()
```

Dimensional Synthesis

The functions under this category are responsible for calculations of linear dimensions of linkages and members. The function `f_Operational_Configuration()` computes the orientation angles during specified operations: digging in this case.

```
[InitialParam] = f_Initial_Parameters()
[LinkDims] = f_LinkDims()
[Four-barDims] = f_Four-bar_Solver()
[OperConfig] = f_Operational_Configuration()
```

Design and Optimization

Engineering design for strength is carried out by simulation of the model in SimMechanics environment under this category of functions. This division also includes functions responsible for extraction of forces and moments information and computing the FBD and MAD analysis. The function `f_Cross_Sectional_Optimizer()` is used to determine an optimized set of the cross-sectional parameters of the boom and the stick based on **minimum material consumption** objective function.

```
OperForce = f_Operation_Force();
open_system Excav_Sim_Model ;
sim Excav_Sim_Model ;
JoinForce = [SD1; SD2; SD3; SD4; SD5; SD6; SD7; SD8; SD9; SD10; SD11;
SD12];
Joint_Forces = f_Joint_Forces(JoinForce);
[PinDims,Exitter] = f_Pin_Dimensions()
CrossDims = f_Cross_Sectional_Optimizer();
f_exceel_writter()
```

Published with MATLAB® 7.11

A1.2 Functions and Subroutines Details

Contents

2.1 Commercial Specification and Bucket Dimension Inputs:

`f_Specification()`

```
function [SpcDat,BuckGeo] = f_Specification()
S11 = 5670;      % actual value from catalog
S1o = 650;       % screen measurement (random unit)
S3o = 427;
S4o = 155;
Ho = 113;
Vo = 141;
bucket_lengtho = 93;
bucket_widtho = 0.6*bucket_lengtho;
image_ratio = S11/S1o; % image ratio: screen measurement to real values
S1 = image_ratio*S1o/1000; % MAXIMMUM REACHOUT AT GROUND LEVEL
S3 = image_ratio*S3o/1000; % MAXIMUM CUTTING HEIGHT
S4 = image_ratio*S4o/1000; % MAXIMUM LOADING HEIGHT
H = image_ratio*Ho/1000;
V = image_ratio*Vo/1000;

% COMMERCIAL_SPECIFICATION = f_Spec_Data_SI(S1,S3,S4,H,V)
SpcDat = c_Spec_Data_SI;
SpcDat.Maximum_Reachout_at_Ground_Level_S1 = S1;
SpcDat.Maximum_Cutting_Height_S3 = S3;
SpcDat.Maximum_Loading_Height_S4 = S4;
SpcDat.Horizontal_Distance_H = H;
SpcDat.Vertical_Distance_V = V;
bucket_length = image_ratio*bucket_lengtho/1000;
bucket_width = image_ratio*bucket_widtho/1000;
bucket_pin_len = 0.5*bucket_width;
teta_bucket = 95;   % Bucket angles (property of bucket geometry)
b0 = 0.35*bucket_length;    % Initial value assumed for b0
Bucket_Geometry = [bucket_width, bucket_length, bucket_pin_len];
BuckGeo = c_Bucket_Geo_SI;
BuckGeo.Bucket_Length_l3 = bucket_length;
BuckGeo.Bucket_Width_BW = bucket_width;
BuckGeo.Bucket_Height_b0 = b0;
BuckGeo.Bucket_Pin_Width_bw = bucket_pin_len;
BuckGeo.Bucket_Angle_teta_bucket = teta_bucket;
BuckGeo.Bulk_Volume_Clearance_Angle = 40;
BuckGeo.Maximum_Upward_Bucket_Open_Limit_Angle = 35;
```

2.2 Dimensional Constraints and Factor of Safeties:

f_Dimensional_Constraints()

```
function [Dimensional_Constraints] = f_Dimensional_Constraints()
pin_tol = 0.03;
FS_pin = 1.5;
FS_base = 1.25;
thick_min = 7e-3;       % Minimum plate thickness
thick_max = 20e-3;      % maximum plate thickness
base_max = 500e-3;
base_min = 100e-3;
height_min = 10e-3;
height_max = 500e-3;
h_stick_max = 500e-3;
ext_1 = 2*thick_min;     % extension of boom pin reinforcement
ext_2 = 2*thick_min;     % extension of stick pin reinforcement
Dimensional_Constraints = c_Dimensional_Constraints_SI;
Dimensional_Constraints.Minimum_Plate_Thickness = thick_min;
Dimensional_Constraints.Maximum_Plate_Thickness = thick_max;
Dimensional_Constraints.Minimum_Base_Dimension = base_min;
Dimensional_Constraints.Maximum_Base_Dimension = base_max;
Dimensional_Constraints.Minimum_Boom_and_Stick_Height = height_min;
Dimensional_Constraints.Maximum_Boom_Height = height_max;
Dimensional_Constraints.Maximum_Stick_Height = h_stick_max;
Dimensional_Constraints.Extension_of_Boom_Pin_Reinforcement = ext_1;
Dimensional_Constraints.Extension_of_Stick_Pin_Reinforcement = ext_2;
```

2.3 Material Properties Selector:

f_Material_Properties()

```
function [MaterProp] = f_Material_Properties()
Mat_prop = material_data_importer;
Pin_Material = Mat_prop(29,:);
Base_Material = Mat_prop(1,:);

% PIN MATERIAL DATA
SigY_pin = 1e6*Pin_Material(1,5);   % Yield stress in (MPa)
SigU_pin = 1e6*Pin_Material(1,6);   % Ultimate tensile strength in (MPa)
Sigall1_pin = (2/3)*SigY_pin;       % Allowable strength in Mpa
Sigall2_pin = (1/4)*SigU_pin;       % Allowable strength in (MPa)
SSpin = [Sigall1_pin;Sigall2_pin];
SSpin2 = sort(SSpin);
Sig_all_pin =  SSpin2(1,1); % Smallest of the allowable stresses (MPa)
v_pin = Pin_Material(1,1);  % Poisson's Ratio
E_pin = 1e9*Pin_Material(1,2);  %Young's modulus of elasticity in (GPa)

% BASE MATERIAL DATA
SigY_base = 1e6*Base_Material(1,5);
SigU_base = 1e6*Base_Material(1,6);
Sigall1_base = (2/3)*SigY_base;
Sigall2_base = (1/4)*SigU_base;
```

```
SSbase = [Sigall1_base;Sigall2_base];
SSbase2 = sort(SSbase);
Sig_all_base = SSbase2(1,1);
v_base = Base_Material(1,1);
E_base = 1e9*Base_Material(1,2);
MaterProp = c_Material_Properties_SI;
MaterProp.Pin_Material = Pin_Material;
MaterProp.Base_Material = Base_Material;
MaterProp.Allowable_Stress_in_Pin = Sig_all_pin;
MaterProp.Allowable_Stress_in_Base = Sig_all_base;
MaterProp.Poison_Ratio_Pin = v_pin;
MaterProp.Youngs_Modulus_Pin = E_pin;
MaterProp.Poison_Ratio_Base = v_base;
MaterProp.Youngs_Modulus_Base = E_base;
```

2.4 Initial Parameters:

f_Initial_Parameters()

```
function [InitialParam] = f_Initial_Parameters()
InitialParam = c_Initial_Parameters_SI;
```

2.5 Neural Network Training Data Generator:

f_NN_Data_Generator()

```
function [Input_Data_S,Target_Data_L] = f_NN_Data_Generator()
Div_gap = 6;
alp1d = 10;
alp2d = 12.5;
alpbud = 33;
alp1 = alp1d*pi/180;
alp2 = alp2d*pi/180;
alpbu = alpbud*pi/180;

% Boom Limits
L1_lb = 1.5;
L1_ub = 2.5;

% Stick Limits
L2_lb = L1_lb/2;
L2_ub = L1_ub/2;

% Bucket Limits
L3_lb = L1_lb/3;
L3_ub = L1_ub/3;
S_lb = 15;
S_ub = 30;

% Valued Vectors
L1 = L1_lb:(L1_ub-L1_lb)/Div_gap:L1_ub;
L2 = L2_lb:(L2_ub-L2_lb)/Div_gap:L2_ub;
L3 = L3_lb:(L3_ub-L3_lb)/Div_gap:L3_ub;
S = S_lb:(S_ub-S_lb)/Div_gap:S_ub;

L1 = L1';
L2 = L2';
L3 = L3';
H = input('H:     ');
V = input('V:     ');
```

```
S = S';
d = 1;
for i = 1:1:(Div_gap)   % boom length index (L1)
    for j = 1:1:(Div_gap)   % stick length index (L2)
        for k = 1:1:(Div_gap)   % bucket length index (L3)
                 for s = 1:1:(Div_gap)% angle β index (β)
            a = L1(i,1);
            b = L2(j,1) + L3(k,1);
            bl2 = L2(j,1);
            bl3 = L3(k,1);
            betta = S(s,1);
            c = sqrt(a^2 + b^2 + 2*a*b*cosd(betta));

            % 1. MAXIMUM REACH-OUT AT GROUND LEVEL, S1

            betta_s1 = real (rad2deg(asin(V/c)));
            TM_S1_1 = [1 0 0 H;
                0 1 0 V;
                0 0 1 0;
                0 0 0 1];
            TM_S1_2 = [cosd(-betta_s1), -sind(-betta_s1), 0, 0;
                sind(-betta_s1), cosd(-betta_s1), 0, 0;
                0, 0, 1, 0;
                0, 0, 0, 1];
            TM_S1_3 = [1 0 0 c;
                0 1 0 0;
                0 0 1 0;
                0 0 0 1];
            A_S1 = TM_S1_1*TM_S1_2*TM_S1_3;
            S1 = abs(A_S1(1,4));

            % 2. MAXIMUM DIGGING DEPTH, S2

            TM_S2_1 = [1 0 0 H;
                0 1 0 V;
                0 0 1 0;
                0 0 0 1];
            TM_S2_2 = [cosd(alp2),-sind(alp2),0,0;
                sind(alp2),cosd(alp2),0,0;
                0,0,1,0;
                0,0,0,1];
            TM_S2_3 = [1 0 0 0;
                0 1 0 -a;
                0 0 1 0;
                0 0 0 1];
            TM_S2_4 = [cosd(-betta),-sind(-betta),0,0;
                sind(-betta),cosd(-betta),0,0;
                0,0,1,0;
                0,0,0,1];
            TM_S2_5 = [1 0 0 0;
                0 1 0 -b;
                0 0 1 0;
                0 0 0 1];
            A_S2 = TM_S2_1*TM_S2_2*TM_S2_3*TM_S2_4*TM_S2_5;
            S2 = abs(A_S2(2,4));

            % 3. MAXIMUM CUTTING HEIGHT, S3

            TM_S3_1 = [1 0 0 H;
                0 1 0 V;
                0 0 1 0;
                0 0 0 1];
            TM_S3_2 = [cosd(alp1-betta),-sind(alp1-betta),0,0;
                sind(alp1-betta),cosd(alp1-betta),0,0;
                0,0,1,0;
                0,0,0,1];
            TM_S3_3 = [1 0 0 0;
                0 1 0 a;
                0 0 1 0;
                0 0 0 1];
            TM_S3_4 = [cosd(-betta),-sind(-betta),0,0;
                sind(-betta),cosd(-betta),0,0;
                0,0,1,0;
                0,0,0,1];
            TM_S3_5 = [1 0 0 0;
                0 1 0 b;
                0 0 1 0;
                0 0 0 1];
            A_S3 =  TM_S3_1*TM_S3_2*TM_S3_3*TM_S3_4*TM_S3_5;
            S3 = abs(A_S3(2,4));

            % 4. MAXIMUM LOADING HEIGHT, S4

            TM_S4_1 = [1 0 0 H;
                0 1 0 V;
                0 0 1 0;
                0 0 0 1];
            TM_S4_2 = [cosd(alp1-betta),-sind(alp1-betta),0,0;
                sind(alp1-betta),cosd(alp1-betta),0,0;
                0,0,1,0;
                0,0,0,1];
            TM_S4_3 = [1 0 0 0;
                0 1 0 a;
                0 0 1 0;
                0 0 0 1];
            TM_S4_4 = [cosd(-betta),-sind(-betta),0,0;
                sind(-betta),cosd(-betta),0,0;
                0,0,1,0;
                0,0,0,1];
            TM_S4_5 = [1 0 0 0;
                0 1 0 bl2;
                0 0 1 0;
                0 0 0 1];
            A_S4 =  TM_S4_1*TM_S4_2*TM_S4_3*TM_S4_4*TM_S4_5;
            A_S4(2,4) = A_S4(2,4)- bl3;
            S4 = abs(A_S4(2,4));

            % 5. MINIMUM LOADING HEIGHT, S5
            TM_S5_1 = [1 0 0 H;
                0 1 0 V;
                0 0 1 0;
                0 0 0 1];
            TM_S5_2 = [cosd(alp1-betta),-sind(alp1-betta),0,0;
                sind(alp1-betta),cosd(alp1-betta),0,0;
                0,0,1,0;
                0,0,0,1];
            TM_S5_3 = [1 0 0 0;
                0 1 0 a;
                0 0 1 0;
                0 0 0 1];
            A_S5 =  TM_S5_1*TM_S5_2*TM_S5_3;
            A_S5(2,4) = A_S5(2,4)- b;
            S5 = abs(A_S5(2,4));
            Input_Data_S(1:1:7,d) = [S1;S2;S3;S4;S5;H;V];
            Target_Data_L(1:1:4,d) = [a;bl2;bl3;betta];
            d = d+1;
```

```
                 end
        end
    end
end
Input_Data_S;
Target_Data_L;
```

2.6 Creating and Training ANN:

NN_Spec_create_fit_net(inputs,targets)

```
function NN_Spec_net = NN_Spec_create_fit_net(inputs,targets)
% *Create Network*
numHiddenNeurons = 20;  % Adjust as desired
NN_Spec_net = newfit(inputs,targets,numHiddenNeurons);
NN_Spec_net.divideParam.trainRatio = 70/100;       % Adjust as desired
NN_Spec_net.divideParam.valRatio = 15/100;         % Adjust as desired
NN_Spec_net.divideParam.testRatio = 15/100;        % Adjust as desired

% *Train and Apply Network*
[NN_Spec_net,tr] = train(NN_Spec_net,inputs,targets);
outputs = sim(NN_Spec_net,inputs);

% *Plot*
plotperf(tr)
plotfit(NN_Spec_net,inputs,targets)
plotregression(targets,outputs)
```

2.7 Input Specification Parameters Sorter:

f_NN_Parameter_Sorter(Input_Data_S)

```
function                [Spec_Paramters_Final,Norm_Ratio]              =
f_NN_Parameter_Sorter(Input_Data_S)
Tol_spec = 2;
Data_size = size(Input_Data_S);
Data_length = Data_size(1,2);
Vector_spec = ones(1,Data_length);
Occupied = zeros(1,5);
Final_S = zeros(5,1);
RATIO_Mtx = zeros(5,1);
for sc = 1:1:5  % priority counter
    tt_t = 0;
    for occ_counter = 1:1:5
        if Occupied(1,occ_counter)==0
            tt_t = tt_t + 1;
            Disp_vector(1,tt_t) = occ_counter;
        end
    end
    disp('Enter a valied specificaton parameter number to prioritize: ')
    disp(Disp_vector)
    PR = input('   ....')
    Spec_val = input('Value of this parameter:     ');
    Occupied(1,PR) = PR;
    Final_S(PR,1) = Spec_val;
    PR_data = Input_Data_S(PR,:);
    Ratio_N = Spec_val/(mean(PR_data));
    RATIO_Mtx(sc,1) = Ratio_N;
    Ratio_first = RATIO_Mtx(1,1);
    Spec_data_norm = (Spec_val/Ratio_first)*Vector_spec;
    Difference_vec = abs(PR_data - Spec_data_norm);
    rsc = 0 ;   % range size counter
    for vc = 1:1:Data_length % vector size counter
        if 100*((Difference_vec(1,vc))/(Spec_data_norm(1,vc)))<Tol_spec
            rsc = rsc+1;
            for csc = 1:1:5     % configuration type counter
                if Occupied(1,csc) == 0
                    Range_Matrix(csc,rsc) = Input_Data_S(csc,vc);
                end
            end
        end
    end
    for rsc2 = 1:1:5
if Occupied(1,rsc2) == 0
    Lower_Limit_Matrix(rsc2,sc) = Ratio_first*(min(Range_Matrix(rsc2,:)));
    Upper_Limit_Matrix(rsc2,sc) = Ratio_first*(max(Range_Matrix(rsc2,:)));
    Range_Matrix_02(rsc2,1) = max(Lower_Limit_Matrix(rsc2,:));
    Range_Matrix_02(rsc2,2) = min(Upper_Limit_Matrix(rsc2,:));
elseif Occupied(1,rsc2) ~= 0
        Range_Matrix_02(rsc2,:) = [0,0];
        end
    end
    disp(Range_Matrix_02)
    disp(Final_S)
    Norm_Ratio = Ratio_first;
    clearvars Disp_vector Range_Matrix;
end
Spec_Paramters_Final = Final_S;
```

2.8 Linkage Dimension Synthesizer:

f_LinkDims_NN()

```
function [LinkDims_NN] = f_LinkDims_NN()
global SpcDat BuckGeo InitialParam C nn_init_sol
S1 = C(1,1)
S3 = C(4,1);
S4 = C(5,1);
H = C(6,1);
V = C(7,1);
linkdim0 = [ nn_init_sol.nn_boom_len_l1;    % initial solution
            nn_init_sol.nn_sticklen_l2;
            deg2rad(nn_init_sol.nn_deflang_beta)];
options = optimset('Display','iter','MaxIter',300,'MaxFunEvals', 1e8, ...
    'TolX', 1e-5, 'TolFun', 1e-6);
[linkdim,fval,exitflag,jacobian]   =   fsolve(@Links_Eq2_NN,   linkdim0,
options);
ang = linkdim(3,1);
betta = rad2deg(ang);
linkdim(3,1) = betta;
l1 = linkdim(1,1);
l2 = linkdim(2,1);
LinkDims_NN = linkdim;
```

2.9 Four-bar Linkages Solver:

f_Four-bar_Solver()

```
function [Four-barDims] = f_Four-bar_Solver()
global BuckGeo
```

```
% THIS SOLVES THE LINEAR EQUATIONS IN THE VECTOR Fourar_1_Eq.m STARTING
% FROM AN INITIAL SOLUTION DERIVED IN PROPORTION FROM THE LENGTH OF LINK
3.
% global b0 tets11 tets2 tets12 teta_bucket

% Note
%  For  an  excavator  arm  facing  to  the  right  side  of  the  page,  b0
represents
% the linkage on the bucket and b1, b2, and b3 are assigned on counter
% clockwise sense. Hence, b3 will be on link l2.

%
**************************************************************************
% Example: for future reference of calculation results
% b0 = 69.7
% b3 = 65.4
%
% b1 = 90.8
% b2 = 102.5
%
**************************************************************************
b0 = BuckGeo.Bucket_Height_b0;
fb10 = [b0; b0];
options = optimset('Display','iter','MaxIter',300,'MaxFunEvals', 1e8, ...
    'TolX', 1e-5, 'TolFun', 1e-6);
[FB1,fval,exitflag,jacobian] = fsolve(@Four-bar_1_Eq, fb10, options);

b1 = real(FB1(1,1));
b2 = real(FB1(2,1));
Four-barDims = c_Four-bar_Solver_SI;
Four-barDims.Four-bar_Link_b0 = b0;
Four-barDims.Four-bar_Link_b1 = b1;    % Initial value assumed for b3
Four-barDims.Four-bar_Link_b2 = b2;
Four-barDims.Four-bar_Link_b3 = 0.30*BuckGeo.Bucket_Length_l3;
```

2.10 Operational Configurations:

f_Operational_Configuration()

```
function [OperConfig] = f_Operational_Configuration()
global SpcDat BuckGeo LinkDims
V = SpcDat.Vertical_Distance_V;
S1 = SpcDat.Maximum_Reachout_at_Ground_Level_S1;
betta = LinkDims.Boom_Deflection_Angle_betta;
dig0 = [atan(V/S1), (2*pi-atan(V/S1))]';    % Initial solution
options = optimset('Display','iter','MaxIter',300,'MaxFunEvals', 1e8, ...
    'TolX', 1e-5, 'TolFun', 1e-6);
[dig,fval,exitflag,jacobian] = fsolve(@Diggangles_Eq, dig0, options);
dig1 = (180/pi)*(real(dig(1,1)));
dig2 = (180/pi)*(real(dig(2,1)));
stick_angleJ2 = LinkDims.Stick_Angle_J2;    % total angle of J2
TS_1 = LinkDims.Stick_Tail_Length; % length of stick tail section
TS_2 = LinkDims.Stick_Forward_Length; % length of longer stick section
stick_angleJ9 = LinkDims.J9_lower; % lower angle of J9
stick_angleJ3= LinkDims.J3_lower; % lower angle of J3

%  1.1. BOOM ROTATION MATRICES
%   Rotation  matrices  to  transform  forces  from  world  coordinate  system
into
%  local coordinate systems on first and second sides of the boom.
```

```
RB1 = [cosd(dig1 + betta), -sind(dig1 + betta), 0;
    sind(dig1 + betta), cosd(dig1 + betta), 0;
    0, 0, 1]';
RB2 = [cosd(dig1 + betta - 2*betta), -sind(dig1 + betta - 2*betta), 0;
    sind(dig1 + betta - 2*betta), cosd(dig1 + betta - 2*betta), 0;
    0, 0, 1]';

% 1.2. STICK ROTATION MATRIX
% This rotation matrix is used to express vectors defined in the world
% reference frame in axis local to the stick.

RS   =   [cosd(dig1   +   betta   -   2*betta   +   stick_angleJ2   -   (180   -
stick_angleJ9)), ...
    -sind(dig1   +   betta   -   2*betta   +   stick_angleJ2   -   (180   -
stick_angleJ9)), 0;
    sind(dig1 + betta - 2*betta + stick_angleJ2 - (180 - stick_angleJ9)),
...
    cosd(dig1 + betta - 2*betta + stick_angleJ2 - (180 - stick_angleJ9)),
0;
    0, 0, 1]';
global OperConfig
OperConfig = c_Operational_Configuration_SI;

[fb_b2_orient, fb_b1_orient] = Fborient_Solver();
OperConfig.Boom_opertating_angle_dig1 = dig1;
OperConfig.Boom_Rotational_Matrix_Sec1_RB1 = RB1;
OperConfig.Boom_Rotational_Matrix_Sec2_RB2 = RB2;
OperConfig.Stick_Rotational_Matrix_RS = RS;
OperConfig.Four-bar_teta_1 = fb_b2_orient;
OperConfig.Four-bar_teta_2 = fb_b1_orient;
```

2.11 Operation Forces:

f_Operation_Force()

```
function OperForce = f_Operation_Force()
global SpcDat
S1 = SpcDat.Maximum_Reachout_at_Ground_Level_S1;
H = SpcDat.Horizontal_Distance_H;
operating_weight = SpcDat.Vehicle_Weight;
% moment of forces about the rear tip of tracks is given by:
Digging_force_mag = (operating_weight*9.81*H)/(H+S1);
Digging_force_vec_gen            =            Digging_force_mag*[-cosd(90-
15)*cosd(45),cosd(15), ...
    -cosd(90-15)*cosd(45)];
Lifting_force_vec_gen           =           -Digging_force_mag*[-cosd(90-
15)*cosd(45),cosd(15), ...
    -cosd(90-15)*cosd(45)];
OperForce = c_Operation_Force_SI;
    OperForce.Digging_Force = Digging_force_vec_gen;
    OperForce.Lifting_Force = Lifting_force_vec_gen;
```

2.12 Joint Forces:

f_Joint_Forces(JoinForce)

```
function Joint_Forces = f_Joint_Forces(JoinForce)
Joint_Forces = c_Joint_Forces_SI;
    Joint_Forces.JointForces = JoinForce;
% Force Designations
```

```
Joint_Forces.F1 = (JoinForce(1,5:7))';
Joint_Forces.F2 = (JoinForce(2,5:7))';
Joint_Forces.F3 = (JoinForce(3,5:7))';
Joint_Forces.F4 = (JoinForce(4,5:7))';
Joint_Forces.F5 = (JoinForce(5,5:7))';
Joint_Forces.F6 = (JoinForce(6,5:7))';
Joint_Forces.F7 = (JoinForce(7,5:7))';
Joint_Forces.F8 = (JoinForce(8,5:7))';
Joint_Forces.F9 = (JoinForce(9,5:7))';
Joint_Forces.F10 = (JoinForce(10,5:7))';
Joint_Forces.F11 = (JoinForce(11,5:7))';
Joint_Forces.F12 = (JoinForce(12,5:7))';

% Moment Designations

Joint_Forces.M1 = (JoinForce(1,2:4))';
Joint_Forces.M2 = (JoinForce(2,2:4))';
Joint_Forces.M3 = (JoinForce(3,2:4))';
Joint_Forces.M4 = (JoinForce(4,2:4))';
Joint_Forces.M5 = (JoinForce(5,2:4))';
Joint_Forces.M6 = (JoinForce(6,2:4))';
Joint_Forces.M7 = (JoinForce(7,2:4))';
Joint_Forces.M8 = (JoinForce(8,2:4))';
Joint_Forces.M9 = (JoinForce(9,2:4))';
Joint_Forces.M10 = (JoinForce(10,2:4))';
Joint_Forces.M11 = (JoinForce(11,2:4))';
Joint_Forces.M12 = (JoinForce(12,2:4))';
end
```

2.13 Pin Dimensions:

f_Pin_Dimensions()

```
function [PinDims,Exitter] = f_Pin_Dimensions()
% This calculates the diameter and lengths of the pins at each joints

PinDims = c_Pin_Dimensions_SI;
global pin_counter
    AA = [4,-1];
    bb = [0];
    lb = [0.010,0.080]';
    ub = [0.150,0.2]';      % upper bound
    x0 = 0.5*(lb+ub);
for pin_counter = 1:1:3
    options =optimset('Display','iter','Algorithm','active-set','TolX', ...
        1e-9,'TolFun', 1e-6);
    [x,fval,exitflag]  =  fmincon(@pin_objective_function,x0,AA,bb,[],[],
...
        lb,ub,@min_pin_stress_con,options);
    PinDimVec(pin_counter,:) = x;
    Exitter(pin_counter,:) = exitflag;
    lb = [0.010,0.080];
    ub = x;
    x0 = x;
end
PinDims.Pin1 = PinDimVec(1,:);
PinDims.Pin2 = PinDimVec(2,:);
PinDims.Pin3 = PinDimVec(3,:);
clearvars -global pin_counter
```

2.14 Cross-sectional Optimization:

f_Cross_Sectional_Optimizer()

```
function CrossDims = f_Cross_Sectional_Optimizer()
global SpcDat BuckGeo OperConfig Dimensional_Constraints LinkDims ...
    Joint_Forces MaterProp PinDims Four-barDims
% ************************************************************************
% *****************              BOOM          *************************
% ************************************************************************

% Force Designations

F1 = Joint_Forces.F1;
F2 = Joint_Forces.F2;
F3 = Joint_Forces.F3;
F4 = Joint_Forces.F4;
F5 = Joint_Forces.F5;
F6 = Joint_Forces.F6;
F7 = Joint_Forces.F7;
F8 = Joint_Forces.F8;
F9 = Joint_Forces.F9;
F10 = Joint_Forces.F10;
F11 = Joint_Forces.F11;
F12 = Joint_Forces.F12;

% Moment Designations

M1 = Joint_Forces.M1;
M2 = Joint_Forces.M2;
M3 = Joint_Forces.M3;
M4 = Joint_Forces.M4;
M5 = Joint_Forces.M5;
M6 = Joint_Forces.M6;
M7 = Joint_Forces.M7;
M8 = Joint_Forces.M8;
M9 = Joint_Forces.M9;
M10 = Joint_Forces.M10;
M11 = Joint_Forces.M11;
M12 = Joint_Forces.M12;

FS_base = MaterProp.Safety_Factor_Boom;
Sig_working_base = (1/FS_base)*MaterProp.Allowable_Stress_in_Base;
sigall = Sig_working_base;
RB1 = OperConfig.Boom_Rotational_Matrix_Sec1_RB1;
RB2 = OperConfig.Boom_Rotational_Matrix_Sec2_RB2;
RS = OperConfig.Stick_Rotational_Matrix_RS;

% PART ONE OF BOOM ===>  [J1 ----> J10/J11]

% FORCES AND MOMENTS EXPRESSED IN REFERENCE TO A LOCAL AXIS PARALLEL TO
SIDE #1

F1T1 = RB1*F1;
F10T1 = RB1*F10;
F11T1 = RB1*F11;
F2T1 = RB1*F2;
M1T1 = RB1*M1;
M10T1 = RB1*M10;
M11T1 = RB1*M11;
M2T1 = RB1*M2;
```

```
% FORCES EXPRESSED IN REFERENCE TO AN AXIS PARALLEL TO SIDE #2

F1T2 = RB2*F1;           % F1 expressed in T2 axis
F10T2 = RB2*F10;         % F10 expressed in T2 axis
F11T2 = RB2*F11;         % F11 expressed in T2 axis
F2T2 = RB2*F2;
M1T2 = RB2*M1;           % M1 expressed in T2 axis
M10T2 = RB2*M10;         % M10 expressed in T2 axis
M11T2 = RB2*M11;         % M11 expressed in T2 axis
M2T2 = RB2*M2;

% ======================================================================
% =========     MOMENTS ON SECTION #1 AND SECTION #2 OF BOOM  =============
% ======================================================================
T = LinkDims.Side_Length_of_Boom_T;
betta = LinkDims.Boom_Deflection_Angle_betta;
h_j10 = LinkDims.Distance_to_J10_on_Boom;
h_j11 = LinkDims.Distance_to_J11_on_Boom;
dig1 = OperConfig.Boom_opertating_angle_dig1;
thick_min = Dimensional_Constraints.Minimum_Plate_Thickness;
thick_max = Dimensional_Constraints.Maximum_Plate_Thickness;
base_min = PinDims.Pin2(2) + 2*thick_min;
base_max = PinDims.Pin1(2) + 2*thick_min;
height_min = 0;
height_minxx = 2*PinDims.Pin1(1); % ############# (needs revise)
height_max = Dimensional_Constraints.Maximum_Boom_Height;
lb_boomxx = [thick_min,base_min,height_minxx];         % lower bound
lb_boom = [thick_min,base_min,height_min];         % lower bound
ub_boom = [thick_max,base_max,height_max];      % upper bound
x0_boom = 0.5*(lb_boomxx + ub_boom);
A_boom = [2,-1,0;2,0,-1];
b_boom = [0;0];
smplpoint_b = 50;
trial_iter = 0;
iter_limit = 5;
for aa = 0:(T/smplpoint_b):(iter_limit*T/smplpoint_b)
    trial_iter = trial_iter+1;
    anglet1_F1 = 180+dig1+betta;   % Angle of the vector directed to F1 (abs)
    vecb1_F1 =aa*[cosd(anglet1_F1),sind(anglet1_F1),0]'; % The vector to F1
    BM_bFt1 = cross(vecb1_F1,F1);% Bending moment only due to F1 on section #1
    BM_bFDt1 = M1;  % Moment due to the eccentricity of the digging force FD
    BM_boom11 = -(BM_bFt1 + BM_bFDt1); % Total Moment at section aa
    BM_boom1 = RB1*BM_boom11;  % the total bending moment (local)
    BM_bxx = BM_boom1(1,1);    % Torsion about local x on section #1
    BM_byy = BM_boom1(2,1);    % Bending Moment about local y on section #1
    BM_bzz = BM_boom1(3,1);    % Bending Moment about local z on section #1
    BM_bx(trial_iter,1) = BM_bxx;   % Vector of Torsion on section #1
    BM_by(trial_iter,1) = BM_byy;   % Vector of Lateral Bending Moment on #1
    BM_bz(trial_iter,1) = BM_bzz;   % Vector of Bending Moment on #1
    FT1 = -(F1T1);  % Reaction force from the joint (w.r.t. local frame)
    FBx1 = FT1(1,1); % axial forces (tensile or compressive)
    FBy1 = FT1(2,1); % lateral force (local y)
    FBz1 = FT1(3,1); % lateral force (local z)
    FBxx = FBx1; % total axial force
    FBvv = sqrt(FBy1^2 + FBz1^2); % total lateral force
    FB_axial(trial_iter,1) = FBxx; % axial force; total
    FB_shear(trial_iter,1) = FBvv; % total lateral force = total shear force
    bmx = BM_bxx;
    bmy = BM_byy;
    bmz = BM_bzz;
    fx = FBxx;
    fv = FBvv;
    options = optimset('Display','iter','Algorithm','active-set','TolX', ...
        1e-6,'TolFun', 1e-6);
    [x,fval] = fmincon(@min_area,x0_boom,A_boom,b_boom,[],[],lb_boom, ...
        ub_boom,@(x)min_area_con(x,bmx,bmy,bmz,fx,fv,sigall),options);
    TBHB(trial_iter,1:1:3) = x;
    TBHB(trial_iter,4) = fval;
    TBHB(trial_iter,5) = aa;
    x0_boom = x;
    clearvars x bmx bmy bmz fx fv TBHB TBHS BM_bx BM_by BM_bz
end
x0_boom
j = 0;
for t1 = 0:T/smplpoint_b:T
    j = j+1;
    tt(j,1) = t1;
    anglet1_F1 = 180+dig1+betta;
    vecb1_F1 = t1*[cosd(anglet1_F1), sind(anglet1_F1), 0]';
    BM_bFt1 = cross(vecb1_F1,F1);
    BM_bFDt1 = M1;
    BM_boom11 = -(BM_bFt1 + BM_bFDt1);
    BM_boom1 = RB1*BM_boom11;
    BM_bxx = BM_boom1(1,1);
    BM_byy = BM_boom1(2,1);
    BM_bzz = BM_boom1(3,1);
    BM_bx(j,1) = BM_bxx;
    BM_by(j,1) = BM_byy;
    BM_bz(j,1) = BM_bzz;
    FT1 = -(F1T1);
    FBx1 = FT1(1,1);
    FBy1 = FT1(2,1);
    FBz1 = FT1(3,1);
    FBxx = FBx1;
    FBvv = sqrt(FBy1^2 + FBz1^2);
    FB_axial(j,1) = FBxx;
    FB_shear(j,1) = FBvv;
    bmx = BM_bxx;
    bmy = BM_byy;
    bmz = BM_bzz;
    fx = FBxx;
    fv = FBvv;
    options = optimset('Display','iter','Algorithm','active-set','TolX',...
        1e-6,'TolFun', 1e-6);
    [x,fval] = fmincon(@min_area,x0_boom,A_boom,b_boom,[],[],lb_boom,...
        ub_boom,@(x)min_area_con(x,bmx,bmy,bmz,fx,fv,sigall),options);
    TBHB(j,1:1:3) = x;
    TBHB(j,4) = fval;
    TBHB(j,5) = t1;
    x0_boom = x;

    % DEFINING THE COORDINATES OF THE SPLINE IN THE FIRST QUADRANT
    % NOTE: HEIGHT -> Y AXIS IN THE MODELING ENVIRONMENT IN NX
    %       BASE -> Z AXIS
    %       INCREMENTAL VALUE OF "t1" -> X AXIS
    Boom_vec1(j,1) = t1;       % vector of t1   (X-COMPONENT)
    Boom_vec1(j,2) = 0.5*TBHB(j,3);   % half-height measured from the middle
    Boom_vec1(j,3) = 0.5*TBHB(j,2);   % half-base measured from the middle
    NX_Boom_Vec1_orig(j,1) = Boom_vec1(j,1);
    NX_Boom_Vec1_orig(j,2) = ...
       Boom_vec1(j,2) - Dimensional_Constraints.Minimum_Plate_Thickness + 2e-3;
    NX_Boom_Vec1_orig(j,3) = ...
        Boom_vec1(j,3) - Dimensional_Constraints.Minimum_Plate_Thickness;
    NX_Boom_Vec1_offstd(j,1) = Boom_vec1(j,1);
    NX_Boom_Vec1_offstd(j,2) = ...
        Boom_vec1(j,2) - Dimensional_Constraints.Minimum_Plate_Thickness;
    NX_Boom_Vec1_offstd(j,3) = Boom_vec1(j,3) + ...
        Dimensional_Constraints.Extension_of_Boom_Pin_Reinforcement;
    clearvars x bmx bmy bmz fx fy
end

%   PART 2 .. SECOND SECTION OF THE BOOM (FROM JOINT 10/11 TO JOINT 2)

    start_pt = 1;
    t2_starter  =  @(t22)(sqrt(h_j11^2  +  t22^2  -  2*h_j11*t22*cosd(90-
betta)));
    t2_critical  = fminsearch(t2_starter,0);
    i = 1;
    q = 0;
for t2 = 0:T/smplpoint_b:T
    t2_NX = t2;
    if t2<t2_critical
        t2 = t2_critical;
    end
    j = j+1;
    q = q+1;
    tt(j,1) = t1+t2_NX;
    v2_01 = sqrt(T^2 + t2^2 - 2*T*t2*cosd(180-2*betta));
    v2_10 = sqrt(h_j10^2 + t2^2 - 2*h_j10*t2*cosd(2*betta));
    v2_11 = sqrt(h_j11^2 + t2^2 - 2*h_j11*t2*cosd(90 - betta));
    alpp2_01 = (asin((T/v2_01)*sind(180-2*betta)))*180/pi;
    alpp2_10 = (asin((h_j10/v2_10)*sind(2*betta)))*180/pi;
    alpp2_11 = (asin((h_j11/v2_11)*sind(90-betta)))*180/pi;
    anglet2_F1 = 180 + dig1 - betta + alpp2_01;
    anglet2_F10 = 180 + dig1 - betta - alpp2_10;
    anglet2_F11 = 180 + dig1 - betta + alpp2_11;
    vecb2_F1 = v2_01*[cosd(anglet2_F1) sind(anglet2_F1) 0]';
    vecb2_F10 = v2_10*[cosd(anglet2_F10), sind(anglet2_F10), 0]';
    vecb2_F11 = v2_11*[cosd(anglet2_F11), sind(anglet2_F11), 0]';
    BM_bFt2 = cross(vecb2_F1, F1) + cross(vecb2_F10,F10) + ...
        cross(vecb2_F11, F11);
    BM_bFDt2 = M1 + M10 + M11;
    BM_boom22 = -(BM_bFt2 + BM_bFDt2);
    BM_boom2 = RB2*BM_boom22;
    BM_bxx = BM_boom2(1,1);   % Torsion about local x on section #2
    BM_byy = BM_boom2(2,1);   % Lateral Bending Moment about local y on #2
    BM_bzz = BM_boom2(3,1);   % Bending Moment about local z on section #2
    BM_bx(j,1) = BM_bxx;   % Vector of Torsion on section #2
    BM_by(j,1) = BM_byy;   % Vector of Lateral Bending Moment on section #2
    BM_bz(j,1) = BM_bzz;   % Vector of B. Moment in the vertical plane on #2
    FT2 = -(F1T2 + F10T2 + F11T2);  % total reaction force at the cross-sec
    FBx2 = FT2(1,1);
    FBy2 = FT2(2,1);
    FBz2 = FT2(3,1);
    FBxx = FBx2;    % axial force
    FBvv = sqrt(FBy2^2 + FBz2^2);   % resultant transverse(shear) force
    FB_axial(j,1) = FBxx;
    FB_shear(j,1) = FBvv;
    bmx = BM_bxx;
    bmy = BM_byy;
    bmz = BM_bzz;
    fx = FBxx;
    fv = FBvv;
    options       =       optimset('Display','iter','Algorithm','active-
set','TolX',...
        1e-6,'TolFun', 1e-6);
    [x,fval] = fmincon(@min_area,x0_boom,A_boom,b_boom,[],[],lb_boom, ...
        ub_boom,@(x)min_area_con(x,bmx,bmy,bmz,fx,fv,sigall),options);
    TBHB(j,1:1:3) = x;
    TBHB(j,4) = fval;
    TBHB(j,5) = t2_NX;
    step_boom_h = TBHB((j-1),3) - TBHB(j,3);
    if (step_boom_h >= 2e-3) | (step_boom_h <= -2e-3)
        if step_boom_h >= 2e-3
            AAA(i,1) = abs(step_boom_h);
            i = i+1;
            TBHB(j,3) = (TBHB(j,3) + AAA(1,1));
        elseif step_boom_h <= -2e-3
            AAA(i,1) = abs(step_boom_h);
            i = i+1;
            TBHB(j,3) = (TBHB(j,3) - AAA(1,1));
        end
        min_area_onevar_b = @(minbase)((minbase)*(TBHB(j,3))) - ...
            ((minbase-2*TBHB(j,1))*(TBHB(j,3)-2*TBHB(j,1)));
        base_complemenatary                                         =
fminbnd(min_area_onevar_b,base_min,base_max);
        TBHB(j,2) = base_complemenatary;
    end
    Boom_vec2(q,1) = t2_NX;             % vector of t1   (X-COMPONENT)
    Boom_vec2(q,2) = 0.5*TBHB(j,3);     % half-height = (0.5*...)(Y-COMPONENT)
    Boom_vec2(q,3) = 0.5*TBHB(j,2);     % half-base = (Z-COMPONENT)
    NX_Boom_Vec2_orig(q,1) = Boom_vec2(q,1);
    NX_Boom_Vec2_orig(q,2) = ...
    Boom_vec2(q,2) - Dimensional_Constraints.Minimum_Plate_Thickness+ 2e-3;
    NX_Boom_Vec2_orig(q,3) = ...
    Boom_vec2(q,3) - Dimensional_Constraints.Minimum_Plate_Thickness;
    NX_Boom_Vec2_offstd(q,1) = Boom_vec2(q,1);
    NX_Boom_Vec2_offstd(q,2) = Boom_vec2(q,2) -  ...
        Dimensional_Constraints.Minimum_Plate_Thickness;
    NX_Boom_Vec2_offstd(q,3) = Boom_vec2(q,3) + ...
        Dimensional_Constraints.Extension_of_Boom_Pin_Reinforcement;
    x0_boom = x;
    clearvars x bmx bmy bmz fx fy step_boom_h
end
NX_Boom_vec1_Original = 1000*NX_Boom_Vec1_orig;
NX_Boom_vec1_Offsetted = 1000*NX_Boom_Vec1_offstd;
NX_Boom_vec2_Original = 1000*NX_Boom_Vec2_orig;
NX_Boom_vec2_Offsetted = 1000*NX_Boom_Vec2_offstd;
dlmwrite('D:\My Dropbox\ForNX\Delimited Files\Boom_vec1_Original.dat', ...
    NX_Boom_vec1_Original,'delimiter',',','precision',4);
dlmwrite('D:\My Dropbox\ForNX\Delimited Files\Boom_vec1_Offsetted.dat', ...
    NX_Boom_vec1_Offsetted,'delimiter',',','precision',4);
dlmwrite('D:\My Dropbox\ForNX\Delimited Files\Boom_vec2_Original.dat', ...
    NX_Boom_vec2_Original,'delimiter',',','precision',4);
dlmwrite('D:\My Dropbox\ForNX\Delimited Files\Boom_vec2_Offsetted.dat', ...
    NX_Boom_vec2_Offsetted,'delimiter',',','precision',4);
plot(tt,BM_bx,'g')
hold on
plot(tt,BM_by,'b')
plot(tt,BM_bz,'r')
title('Boom - Torsion and Bending Moment Diagrams')
legend('Torsion (local x)','Bending M. (local y)','Bending M. (local z)')
figure
plot(tt,FB_axial,'g')
hold on
plot(tt,FB_shear,'r')
title('Boom Axial and Shear Froces')
legend('Axial Force','Total shear force')
figure
plot(TBHB(:,1:1:4))
title('Dimensions of cross sectional area')
xlabel('Boom span')
legend('Thickness','Base','Height')
```

```
% ==========================================================================
% =============================   END OF BOOM   ============================
% ==========================================================================

% *************************************************************************
% ************************        STICK     ******************************
% *************************************************************************

% FORCES AND MOMENTS EXPRESSED WITH RESPECT TO A LOACAL COORDINATE SYSTEM

F2S = RS*(-F2);
F9S = RS*F9;
F8S = RS*F8;
F6S = RS*F6;
F3S = RS*F3;
M2S = RS*(-M2);
M9S = RS*M9;
M8S = RS*M8;
M6S = RS*M6;
M3S = RS*M3;
angles1_F6 = 180;
angles1_F9 = 180;

% MOMENTS

% PART 1 .. STICK TAIL SECTION

thick_min = Dimensional_Constraints.Minimum_Plate_Thickness;
thick_max = Dimensional_Constraints.Maximum_Plate_Thickness;
h_stick = LinkDims.Distance_to_J2_and_J8_on_Stick;
h_j8 = h_stick;
TS_1 = LinkDims.Stick_Tail_Length;
TS_2 = LinkDims.Stick_Forward_Length;
b3 = Four-barDims.Four-bar_Link_b3;
base_min = PinDims.Pin3(2) + 2*thick_min;
base_max = base_min;
height_min = 2*PinDims.Pin2(1);
height_max = Dimensional_Constraints.Maximum_Stick_Height;
lb_stick = [thick_min,base_min,height_min];     % lower bound
ub_stick = [thick_max,base_max,height_max];     % upper bound
x0_stick = 0.5*(lb_stick + ub_stick);
A_stick = [2 -1 0;2 0 -1];
b_stick = [0;0];
smplpoint_s1 = (TS_1/T)*smplpoint_b;
smplpoint_s2 = ((TS_2-b3)/T)*smplpoint_b;
smplpoint_s3 = (b3/T)* smplpoint_b;
k = 0;
for st1 = 0:TS_1/smplpoint_s1:TS_1
    k = k+1;
    ST(k,1) = st1;
vecs1_F9 = st1*[cosd(angles1_F9); sind(angles1_F9); 0]; % Vector to F9 #1
BM_s1 = cross(vecs1_F9,F9S);    % B. moment only due to F9 (w.r.t. local)
BM_DFs1 = RS*M9;  % Pure bending moment at joint 9 (w.r.t. local frame)
    BM_stick11=-(BM_s1+ BM_DFs1);%total REACTION moment  st1(local frame)
    BM_stick1 = BM_stick11;     % (local frame)
    bmx = BM_stick1(1,1);   % Torsion at location St1
    bmy = BM_stick1(2,1);   % Lateral Bending Moment at location St1
    bmz = BM_stick1(3,1);   % B. Moment in the vertical plane at location St1
    BM_sx(k,1) = bmx;   % formation of Vector of Torsion along the tail sec
    BM_sy(k,1) = bmy;   % '' '''' Lateral B. Moment along the tail ''
    BM_sz(k,1) = bmz;   % '' '' '' B. Moment in the vertical plane
    % calculation of direct shear and axial forces
    FS1 = -(F9S);
    FSx1 = FS1(1,1);
    FSy1 = FS1(2,1);
    FSz1 = FS1(3,1);
    fx = FSx1;  % axial force
    fv = sqrt(FSy1^2 + FSz1^2); %  combined to give resultant value
    FS_axial(k,1) = fx;
    FS_shear(k,1) = fv;
    options = optimset('Display','iter','Algorithm','active-set','TolX', ...
       1e-6,'TolFun', 1e-6);
    [x,fval]                                                          =
fmincon(@min_area,x0_stick,A_stick,b_stick,[],[],lb_stick,...
        ub_stick,@(x)min_area_con(x,bmx,bmy,bmz,fx,fv,sigall),options);
    TBHS(k,1:1:3) = x;
    TBHS(k,4) = fval;
    TBHS(k,5) = st1;
    x0_stick = x;
    Stick_vecc(k,1) = st1; % vector of thickness
    Stick_vecc(k,2) = 0.5*TBHS(k,3);    % vector of half-height
    Stick_vecc(k,3) = 0.5*TBHS(k,2);    % vector of half-base
    NX_Stick_vecc_orig(k,1) = Stick_vecc(k,1);
    NX_Stick_vecc_orig(k,2) = Stick_vecc(k,2) - ...
        Dimensional_Constraints.Minimum_Plate_Thickness + 2e-3;
    NX_Stick_vecc_orig(k,3) = Stick_vecc(k,3) - ...
        Dimensional_Constraints.Minimum_Plate_Thickness;
    NX_Stick_vecc_offstd(k,1) = Stick_vecc(k,1);
    NX_Stick_vecc_offstd(k,2) = Stick_vecc(k,2) - ...
        Dimensional_Constraints.Minimum_Plate_Thickness;
    NX_Stick_vecc_offstd(k,3) = Stick_vecc(k,3) + ...
        Dimensional_Constraints.Extension_of_Boom_Pin_Reinforcement;
    clearvars x bmx bmy bmz fx fy
    end
% PART 2 .. LONGER (INTERMIDIATE) STICK SEECTION
k2 = 0;
for st2 = 0:TS_2/smplpoint_s2:(TS_2 - b3)
    k = k+1;
    k2 = k2+1;
    st22 = TS_1 + st2;
    ST(k,1) = st22;
    vecs2F2 = sqrt((h_stick)^2 + st2^2); % Magnitude of F2 from location st2
    vecs2F8 = sqrt((h_j8)^2 + st2^2); % magnitude of vector from st2 to F8
    alps2F2 = radtodeg(atan(h_stick/st2)); % Intermediate variable
    alps2F8 = radtodeg(atan(h_j8/st2)); % Intermediate variable
    anglest2_F2 = 180 + alps2F2;
    anglest2_F8 = 180 - alps2F8;
    vecs2_F2 = vecs2F2*[cosd(anglest2_F2); sind(anglest2_F2); 0];
    vecs2_F9 = (TS_1 + st2)*[cosd(angles1_F9); sind(angles1_F9); 0];
    vecs2_F8 = vecs2F8*[cosd(anglest2_F8); sind(anglest2_F8); 0];
    BM_s2    =    cross(vecs2_F2,F2S)    +    cross(vecs2_F9,F9S)    +
cross(vecs2_F8,F8S);
    BM_DFs2 = RS*(M9 + M2 + M8);
    BM_stick22 = -(BM_s2 + BM_DFs2);
    BM_stick2 = BM_stick22; % (local frame)
    bmx = BM_stick2(1,1);
    bmy = BM_stick2(2,1);
    bmz = BM_stick2(3,1);
    BM_sx(k,1) = bmx;
    BM_sy(k,1) = bmy;
    BM_sz(k,1) = bmz;

    % axial and direct shear forces
    FS2 = -(F2S + F9S + F8S);
    FSx2 = FS2(1,1);
    FSy2 = FS2(2,1);
    FSz2 = FS2(3,1);
    fx = FSx2;
    fv = sqrt(FSy2^2 + FSz2^2);
    FS_axial(k,1) = fx;
    FS_shear(k,1) = fv;
    options = optimset('Display','iter','Algorithm','active-set','TolX', ...
        1e-6,'TolFun', 1e-6);
    [x,fval] = fmincon(@min_area,x0_stick,A_stick,b_stick,[],[],lb_stick, ...
        ub_stick,@(x)min_area_con(x,bmx,bmy,bmz,fx,fv,sigall),options);
    TBHS(k,1:1:3) = x;
    TBHS(k,4) = fval;
    TBHS(k,5) = st2;
    x0_stick = x;
    Stick_vecc(k,1) = st22; % vector of thickness
    Stick_vecc(k,2) = 0.5*TBHS(k,3);    % vector of half-height
    Stick_vecc(k,3) = 0.5*TBHS(k,2);    % vector of half-base
    Stick_VEC_2nd(k2,1) = st2;
    Stick_VEC_2nd(k2,2) = Stick_vecc(k,2);
    Stick_VEC_2nd(k2,3) = Stick_vecc(k,3);
    NX_Stick_vecc_orig(k,1) = Stick_vecc(k,1);
    NX_Stick_vecc_orig(k,2) = Stick_vecc(k,2) - ...
        Dimensional_Constraints.Minimum_Plate_Thickness + 2e-3;
    NX_Stick_vecc_orig(k,3) = Stick_vecc(k,3) - ...
        Dimensional_Constraints.Minimum_Plate_Thickness;
    NX_Stick_vecc_offstd(k,1) = Stick_vecc(k,1);
    NX_Stick_vecc_offstd(k,2) = Stick_vecc(k,2) - ...
        Dimensional_Constraints.Minimum_Plate_Thickness;
    NX_Stick_vecc_offstd(k,3) = Stick_vecc(k,3) + ...
        Dimensional_Constraints.Extension_of_Boom_Pin_Reinforcement;
    clearvars x bmx bmy bmz fx fy
end

% PART 3 .. SECTION BETWEEN JOINT 6 AND JOINT 3

for st2 = (TS_2 - b3):b3/smplpoint_s3:TS_2
    k = k+1;
    st22 = TS_1 + st2;
    ST(k,1) = st22;
    vecs2F2 = sqrt((h_stick)^2 + st2^2);    % magnitude of vector to F2
    vecs2F8 = sqrt((h_j8)^2 + st2^2);       % magnitude of vector to F8
    alps2F2 = radtodeg(atan(h_stick/st2));  % frame-independent angle
    alps2F8 = radtodeg(atan(h_j8/st2));     % frame-independent angle
    anglest2_F2 = 180 + alps2F2;    % (local frame)
    anglest2_F8 = 180 - alps2F8;    % (local frame)
    vecs2_F2 = vecs2F2*[cosd(anglest2_F2); sind(anglest2_F2); 0];
    vecs2_F9 = (TS_1 + st2)*[cosd(angles1_F9); sind(angles1_F9); 0];
    vecs2_F8 = vecs2F8*[cosd(anglest2_F8); sind(anglest2_F8); 0];
    vecs2_F6  =  (st2-  (TS_2  -  b3))*[cosd(angles1_F6);  sind(angles1_F6);
0];
    BM_s3 = cross(vecs2_F2,F2S) + cross(vecs2_F9,F9S) + ...
        cross(vecs2_F8,F8S) + cross(vecs2_F6,F6S);  % (local frame)
    BM_DFs3 = M9S + M2S + M8S + M6S;    % (local frame)
    BM_stick33 = -(BM_s3 + BM_DFs3);    %(local frame)
    BM_stick3 = BM_stick33; % local frame
    bmx = BM_stick3(1,1);   % Torsion on section #2
    bmy = BM_stick3(2,1);   % Bending Moment on section #2
    bmz = BM_stick3(3,1);% Bending Moment in the vertical plane on sec.#2
    BM_sx(k,1) = bmx;   % Vector of Torsion on section #2
    BM_sy(k,1) = bmy;   % Vector of Lateral Bending Moment on section #2
    BM_sz(k,1)= bmz;% Vector of B. Moment in the vertical plane on sec.#2
    FS3 = -(F2S + F9S + F8S + F6S);     %(local frame)
    FSx3 = FS3(1,1);
    FSy3 = FS3(2,1);
    FSz3 = FS3(3,1);
    fx = FSx3;                      % total axial force  (local frame)
    fv = sqrt(FSy3^2 + FSz3^2);     % total shear force (local frame)
    FS_axial(k,1) = fx;
    FS_shear(k,1) = fv;
    options = optimset('Display','iter','Algorithm','active-set','TolX', ...
       1e-6,'TolFun', 1e-6);
    [x,fval]                                                              =
fmincon(@min_area,x0_stick,A_stick,b_stick,[],[],lb_stick,...
        ub_stick,@(x)min_area_con(x,bmx,bmy,bmz,fx,fv,sigall),options);
    TBHS(k,1:1:3) = x;
    TBHS(k,4) = fval;
    TBHS(k,5) = st2;
    Stick_vecc(k,1) = st22; % vector of thickness
    Stick_vecc(k,3) = 0.5*TBHS(k,2);    % vector of half-base
    Stick_vecc(k,2) = 0.5*TBHS(k,3);    % vector of half-height
    NX_Stick_vecc_orig(k,1) = Stick_vecc(k,1);
    NX_Stick_vecc_orig(k,2) = Stick_vecc(k,2) - ...
        Dimensional_Constraints.Minimum_Plate_Thickness + 2e-3;
    NX_Stick_vecc_orig(k,3) = Stick_vecc(k,3) - ...
        Dimensional_Constraints.Minimum_Plate_Thickness;
    NX_Stick_vecc_offstd(k,1) = Stick_vecc(k,1);
    NX_Stick_vecc_offstd(k,2) = Stick_vecc(k,2) - ...
        Dimensional_Constraints.Minimum_Plate_Thickness;
    NX_Stick_vecc_offstd(k,3) = Stick_vecc(k,3) + ...
        Dimensional_Constraints.Extension_of_Boom_Pin_Reinforcement;
    NX_Stick_vecc = 1000*Stick_vecc;
    x0_stick = x;
end
NX_Stick_vecc = 1000*Stick_vecc;
NX_Stick_vecc_orig = 1000*NX_Stick_vecc_orig;
NX_Stick_vecc_offstd = 1000*NX_Stick_vecc_offstd;
dlmwrite('D:\My Dropbox\ForNX\Delimited Files\NX_Stick_vecc_orig.dat',...
    NX_Stick_vecc_orig,'delimiter','\t','precision',4);
dlmwrite('D:\My Dropbox\ForNX\Delimited Files\NX_Stick_vecc_offstd.dat',...
    NX_Stick_vecc_offstd,'delimiter',',','precision',4);
NX_Stic_vec_Y_val_orig = NX_Stick_vecc_orig(:,2);
Stick_Y_first_orig = NX_Stic_vec_Y_val_orig(1,1);
Stick_Y_middle_orig = max(NX_Stic_vec_Y_val_orig);
Stick_Y_last_orig = NX_Stic_vec_Y_val_orig(k,1);
NX_Stick_Linearized_Data_orig = [Stick_Y_first_orig,
                                  Stick_Y_middle_orig,
                                  Stick_Y_last_orig,
                                  0.5*1000*base_min];
NX_Stic_vec_Y_val_offstd = NX_Stick_vecc_offstd(:,2);
Stick_Y_first_offstd = NX_Stic_vec_Y_val_offstd(1,1);
Stick_Y_middle_offstd = max(NX_Stic_vec_Y_val_offstd);
Stick_Y_last_offstd = NX_Stic_vec_Y_val_offstd(k,1);
NX_Stick_Linearized_Data_offstd = [Stick_Y_first_offstd,
    Stick_Y_middle_offstd,
    Stick_Y_last_offstd,
    0.5*1000*base_min];
Dlmwrite
('D:\My Dropbox\ForNX\Delimited Files\NX_Stick_Linearized_Data_orig.dat',...
    NX_Stick_Linearized_Data_orig,'delimiter',',','precision',4);
Dlmwrite
('D:\My Dropbox\ForNX\Delimited Files\NX_Stick_Linearized_Data_offstd.dat',...
    NX_Stick_Linearized_Data_offstd,'delimiter',',','precision',4);
figure
plot(ST,BM_sx,'g')
hold on
plot(ST,BM_sy,'b')
plot(ST,BM_sz,'r')
title('Stick - Torsion and Bending Moment Diagrams')
legend('Torsion (local x)','Bending M. (local y)','Bending M. (local z)')
figure
```

```
plot(ST,FS_axial,'g')
hold on
plot(ST,FS_shear,'r')
title('Stick Axial and Total Sehar Forces')
legend('Axial Force','Shear Force')
figure
plot(TBHS(:,1:1:4))
title('STICK - Cross-sectional dimensions')
xlabel('Stick span')
legend('Thickness','Base','Height')
% ========================================================================
% =============================   END OF STICK  ===========================
% ========================================================================

CrossDims  = c_Cross_Sectional_Optimizer_SI;
        CrossDims.Boom_Optimized_Crosssectional_Area_TBHB = TBHB;
        CrossDims.Boom_Spline_Vec1 = Boom_vec1;
        CrossDims.Boom_Spline_Vec2 = Boom_vec2;
        CrossDims.Stick_Optimized_Crosssectional_Area_TBHS = TBHS;
        CrossDims.Stick_Spline_Vec = Stick_vecc;
```

2.15 Data Exporter: `f_exceel_writter()`

```
function [] = f_exceel_writter()
global PinDims LinkDims OperConfig Four-barDims

% Pin Dimension Writer
NX_PinDims = 1000*[PinDims.Pin1;
                PinDims.Pin2;
                PinDims.Pin3;
                PinDims.Pin4;
                PinDims.Pin5;
                PinDims.Pin6;
                PinDims.Pin7;
                PinDims.Pin8;
                PinDims.Pin9;
                PinDims.Pin10;
                PinDims.Pin11;
                PinDims.Pin12];

xlswrite('D:\My Dropbox\ForNX\Delimited Files\NX_PinDims',NX_PinDims)
dlmwrite('D:\My Dropbox\ForNX\Delimited Files\NX_PinDims.dat',NX_PinDims,...
    'delimiter','\t','precision',4);

% Link Dimensions Writers

NX_LinkDims = [1000*LinkDims.Boom_Shortcut_Length_l1;
                LinkDims.Boom_Deflection_Angle_betta;
                1000*LinkDims.Side_Length_of_Boom_T;
                1000*LinkDims.Stick_Length_l2;
                LinkDims.Stick_Angle_J2;
                LinkDims.J2_left;
                LinkDims.J2_right;
                LinkDims.Stick_Angle_J8;
                LinkDims.J8_left;
                LinkDims.J8_right;
                LinkDims.Stick_Angle_J9;
                LinkDims.J9_up;
                LinkDims.J9_lower;
                LinkDims.Stick_Angle_J3;
                LinkDims.J3_up;
                LinkDims.J3_lower;
                1000*LinkDims.Distance_to_J2_and_J8_on_Stick;
                1000*LinkDims.Distance_to_J10_on_Boom;
                1000*LinkDims.Distance_to_J11_on_Boom;
                1000*LinkDims.Stick_Tail_Length;
                1000*LinkDims.Stick_Forward_Length];

xlswrite('D:\My Dropbox\ForNX\Delimited Files\NX_LinkDims.xls',NX_LinkDims)
dlmwrite('D:\My Dropbox\ForNX\Delimited Files\NX_LinkDims.dat',NX_LinkDims,...
    'delimiter','\t','precision',4);

% Four-bar Linkage dimensions
NX_Four-barDims = 1000*[Four-barDims.Four-bar_Link_b0;
                    Four-barDims.Four-bar_Link_b1;
                    Four-barDims.Four-bar_Link_b2;
                    Four-barDims.Four-bar_Link_b3];
dlmwrite('D:\My Dropbox\ForNX\Delimited Files\NX_Four-barDims.dat',...
    NX_Four-barDims,'delimiter','\t','precision',4);
% Configuration Parameter Writer
NX_OperConfig = [OperConfig.Boom_opertating_angle_dig1;
                0;
                0;
                0;
                OperConfig.Four-bar_teta_1;
                OperConfig.Four-bar_teta_2;
                OperConfig.Four-bar_teta_3];
xlswrite('D:\My Dropbox\ForNX\Delimited Files\NX_OperConfig.xls',NX_OperConfig)
dlmwrite('D:\My Dropbox\ForNX\Delimited Files\NX_OperConfig.dat',NX_OperConfig,...
    'delimiter','\t','precision',4);
```

Published with MATLAB® 7.11

Appendix 2 API CAD Programming

A2.1 Boom API Programming

A2.1.1 Boom header file:

```
#define UF_CALL(X) (report_error( __FILE__, __LINE__, #X, (X)))


#define xc_axis 1
#define yc_axis 2
#define zc_axis 3
#define xy_plane 1
#define yz_plane 2
#define zx_plane 3

#define NX_BOOM_VEC_1_ORIGINAL
"D:\\My Dropbox\\ForNX\\Delimited Files\\Boom_vec1_Original.dat","r"
#define NX_BOOM_VEC_1_OFFESETTED
"D:\\My Dropbox\\ForNX\\Delimited Files\\Boom_vec1_Offsetted.dat","r"
#define NX_BOOM_VEC_2_ORIGINAL
"D:\\My Dropbox\\ForNX\\Delimited Files\\Boom_vec2_Original.dat","r"
#define NX_BOOM_VEC_2_OFFESETTED
"D:\\My Dropbox\\ForNX\\Delimited Files\\Boom_vec2_Offsetted.dat","r"

typedef struct _iobuf FILE;

struct DATUM_CSYS_DATA{
       double offset_x;
       double offset_y;
       double offset_z;
       double angle_x;
       double angle_y;
       double angle_z;
       bool transform_sequence;
       int rotation_sequence[2];
       };
struct LINK_DIMENSIONS{
       double Boom_shot_len_l1;
       double Boom_defct_ang_betta;
       double Boom_side_len_T;
       double Stick_len_l2;
       double Stick_ang_J2;
       double J2_left;
       double J2_right;
       double Stick_ang_J8;
       double J8_lfet;
       double J8_right;
       double Stick_ang_J9;
       double J9_up;
       double J9_lower;
       double Stick_ang_J3;
       double J3_up;
       double J3_lower;
       double Dist_2_J2andJ8_on_Stick;
       double Dist_2_J10_on_Boom;
       double Dist_2_J11_on_Boom;
       double Stick_tail_len;
       double Stick_forward_len;
       };
```

```
struct PIN_DIMENSIONS{
	double Pin1;
	double Pin2;
	double Pin3;
	double Pin4;
	double Pin5;
	double Pin6;
	double Pin7;
	double Pin8;
	double Pin9;
	double Pin10;
	double Pin11;
	double Pin12;
	};

struct OPERATIONAL_CONFIGURATION{
	double Boom_oper_ang_dig1;
	double Boom_matrix_1_RB1;
	double Boom_matrix_1_RB2;
	double Stick_matrix_RS;
	double Four-bar_teta_1;
	double Four-bar_teta_2;
	double Four-bar_teta_3;
	};

struct FOUR-BAR_DIMENSIONS{
	double link0_bo;
	double link1_b1;
	double link2_b2;
	double link3_b3;
	};


int report_error( char *file,
             int line,
             char *call,
             int irc);
tag_t wub_Create_New_Part_File(char file_path[UF_CFI_MAX_FILE_NAME_SIZE]);
tag_t wub_CSYS_origin_and_direction(void);
tag_t Extract_Smart_tag_of_Datum_CSYS(tag_t tag_DATUM_CSYS);
tag_t Extract_daxis_tag_of_Datum_CSYS(tag_t Datum_CSYS_tag,
                                int Axis_Num);
tag_t Extract_dplane_tag_of_Datum_CSYS(tag_t Datum_CSYS_tag,
                                int Plane_Num);
tag_t wub_set_wcs(tag_t target_DATUM_wcs_CSYS);
tag_t wub_CSYS_offset(tag_t referece_datum_CSYS,
                 const double linear_offset[3],
                 const double angular_offset[3],
                 bool operation_sequence);
tag_t wub_Ceate_Datum_Plane_Offset(     tag_t Referece_DATUM_CSYS,
                                    int Axis_about);
tag_t wub_Create_Projected_Curve(tag_t curve_tag,
                                    tag_t Datum_CSYS_tag,
                                    int Plane_Num);
tag_t wub_Sketch_boom_profile1(tag_t datum);
tag_t wub_Create_Fit_Spline_on_WCS(FILE *csv_file);
tag_t do_ugopen_api(void);
tag_ wub_Create_DATUM_CSYS_Offset_Method(DATUM_CSYS_DATA
                                Transformation_Data,
                                tag_t Reference_CSYS);
int wub_LinkDim_Data_Importer(FILE *dat_filepath,
                                LINK_DIMENSIONS *pLink_Dimensions_obj);
```

```
int wub_OperConfig_Data_Importer(FILE *dat_filepath,
OPERATIONAL_CONFIGURATION *pOperConfig_obj);
tag_t wub_Mirror_a_Curve_through_a_Plane(tag_t Curve_Tag,
                                        tag_t Plane_Tag);
tag_t wub_Trim_Curve_by_Datum_Plane(tag_t Curve_Tag,
                                        tag_t Datum_Plane_Tag,
                                        int which_end);
tag_t wub_Trim_Curve_by_Curves(tag_t Target_curve_Tag,
                                  tag_t Tool_curve1,
                                  tag_t Tool_curve2);
tag_t wub_Bridge_Curves(tag_t Trimmed_Curve_1_Tag,
                           tag_t Trimmed_Curve_2_Tag,
                           int Reverse1,
                           int Reverse2,
                           int par1,
                           int par2);
tag_t wub_Lines_from_two_points(tag_t point1, tag_t point2);
tag_t wub_Lines_Point_Tangent(tag_t point,
                           tag_t tangent,
                           tag_t Reference_CSYS,
                           int Plane_Num);
tag_t wub_SWEEP_2_guides(tag_t Guide_s1, tag_t Guide_s2, tag_t Section);
tag_t wub_Join_Curves(tag_t *curves,int n);
tag_t wub_Point_from_Spline(tag_t curve_Tag, int line_end);
tag_t wub_BPLANE(tag_t Curve_String[2]);
tag_t wub_SKETCHES_J2_adopter(char name[30],
                                  tag_t Refrence_CSYS,
                                  int Plane_num,
                                  int Axis_num);
tag_t THICKEN_Sheet(tag_t sheet_body_tag);
tag_t wub_ARC_on_sketch(tag_t Reference_CSYS);
tag_t wub_ARC_Center_Radius(tag_t Reference_CSYS,
                                  int Plane_num,
                                  double radius,
                                  double arc_center[3],
                                  double start_ang,
                                  double end_ang);
tag_t wub_ARC_Point_Point_Radius(tag_t Reference_CSYS,
                                  int Plane_num,
                                  double radius,
                                  double arc_center[3],
                                  tag_t p1,
                                  tag_t p2);
tag_t wub_Extrude(tag_t Connected_Curve, char* limit[2]);
tag_t wub_SKETCH_J11(tag_t Reference_CSYS,
                     int Plane_num,
                     tag_t line_Tag,
                     tag_t bridge_tag);
tag_t wub_SKETCH_J12(tag_t Reference_CSYS,int Plane_num,tag_t Curve_Tag);
tag_t wub_UNITE_SOLIDS(tag_t Target_Solid, tag_t Tool_Solid);
int wub_Four-barDim_Data_Importer(FILE *dat_filepath,
                                  FOUR-BAR_DIMENSIONS *pFour-
                           bar_dimensions_obj);
```

A2.1.2 Boom Main Codes Assembly:

Main_Body_8.cpp

```
#include <stdio.h>
#include <string.h>
#include <uf.h>
#include <uf_ui.h>
#include <uf_part.h>
#include <uf_csys.h>
#include <uf_modl.h>
#include <uf_modl_primitives.h>
#include <uf_disp.h>
#include <uf_so.h>
#include <uf_curve.h>
#include <uf_obj.h>
#include <uf_object_types.h>
#include <uf_assem.h>
#include <uf_modl_datum_features.h>
#include <uf_defs.h>
#include <stdlib.h>
#include <malloc.h>
#include <uf_sket.h>


#include "Prototype_functions_8.h"

 /*===============================================================*/
 /*====================  MAIN FUNCTION    ======================*/
 /*===============================================================*/
void ufusr(char *param, int *retcode, int paramLen)
       {
       if (UF_CALL(UF_initialize())) return;

       LINK_DIMENSIONS
              Link_Dimensions_obj;
       OPERATIONAL_CONFIGURATION
              OperConfig_obj;
       FOUR-BAR_DIMENSIONS
              Four-bar_dimensions_obj;
       FILE
              *boom_csv_file1_orig,
              *boom_csv_file1_ofstd,
              *boom_csv_file2_orig,
              *boom_csv_file2_ofstd,

              *NX_PinDims_file,
              *NX_LinkDims_file,
              *NX_Four-barDims_file,
              *NX_OperConfig_file;

       NX_LinkDims_file =
       fopen("D:\\My Dropbox\\ForNX\\Delimited Files\\NX_LinkDims.dat","r");
       wub_LinkDim_Data_Importer(NX_LinkDims_file, &Link_Dimensions_obj);
       fclose(NX_LinkDims_file);

       NX_Four-barDims_file =
       fopen("D:\\My Dropbox\\ForNX\\Delimited Files\\NX_Four-barDims.dat","r");
       wub_Four-barDim_Data_Importer(NX_Four-barDims_file, &Four-
       bar_dimensions_obj);
       fclose(NX_Four-barDims_file);
```

```
        NX_OperConfig_file = 
        fopen("D:\\My Dropbox\\ForNX\\Delimited Files\\NX_OperConfig.dat","r");
        wub_OperConfig_Data_Importer(NX_OperConfig_file, &OperConfig_obj);
        fclose(NX_OperConfig_file);

        tag_t
               Boom_part,
               DATUM_CSYS_boom_0,
               DATUM_CSYS_boom_1,
               DATUM_CSYS_boom_2,
               DATUM_CSYS_boom_3,
               DATUM_CSYS_boom_4,
               DATUM_CSYS_boom_5,
               DATUM_CSYS_boom_6,
               DATUM_CSYS_boom_7,
               DATUM_CSYS_boom_8,
               DATUM_CSYS_boom_9,
               DATUM_CSYS_boom_10,
               DATUM_CSYS_boom_11,

               datum_PLANE_1,
               datum_PLANE_2,

               Line_straight,

               bridge_J1,
               bridge_J2,

               Sketch_Boom_profile_1;

        DATUM_CSYS_DATA CSYS_boom_0_data;

        CSYS_boom_0_data.offset_x = 0.0;
        CSYS_boom_0_data.offset_y = 0.0;
        CSYS_boom_0_data.offset_z = 0.0;
        CSYS_boom_0_data.angle_x = 0.0;
        CSYS_boom_0_data.angle_y = 0.0;
        CSYS_boom_0_data.angle_z = 0.0;
        CSYS_boom_0_data.transform_sequence = true;
        CSYS_boom_0_data.rotation_sequence[0] = 1;
        CSYS_boom_0_data.rotation_sequence[1] = 2;

        DATUM_CSYS_DATA CSYS_boom_1_data;

        CSYS_boom_1_data.offset_x = 0.0;
        CSYS_boom_1_data.offset_y = 0.0;
        CSYS_boom_1_data.offset_z = 0.0;
        CSYS_boom_1_data.angle_x = 0.0;
        CSYS_boom_1_data.angle_y = 0.0;
        CSYS_boom_1_data.angle_z = 
Link_Dimensions_obj.Boom_defct_ang_betta + 
OperConfig_obj.Boom_oper_ang_dig1;
        CSYS_boom_1_data.transform_sequence = true;
        CSYS_boom_1_data.rotation_sequence[0] = 1;
        CSYS_boom_1_data.rotation_sequence[1] = 2;

        DATUM_CSYS_DATA CSYS_boom_2_data;

        CSYS_boom_2_data.offset_x = Link_Dimensions_obj.Boom_side_len_T;
        CSYS_boom_2_data.offset_y = 0.0;
        CSYS_boom_2_data.offset_z = 0.0;
        CSYS_boom_2_data.angle_x = 0.0;
```

```
       CSYS_boom_2_data.angle_y = 0.0;
       CSYS_boom_2_data.angle_z = 0.0;
       CSYS_boom_2_data.transform_sequence = true;
       CSYS_boom_2_data.rotation_sequence[0] = 1;
       CSYS_boom_2_data.rotation_sequence[1] = 2;
       DATUM_CSYS_DATA CSYS_boom_3_data;

       CSYS_boom_3_data.offset_x = Link_Dimensions_obj.Boom_side_len_T;
       CSYS_boom_3_data.offset_y = 0.0;
       CSYS_boom_3_data.offset_z = 0.0;
       CSYS_boom_3_data.angle_x = 0.0;
       CSYS_boom_3_data.angle_y = 0.0;
       CSYS_boom_3_data.angle_z = -
2*(Link_Dimensions_obj.Boom_defct_ang_betta);
       CSYS_boom_3_data.transform_sequence = true;
       CSYS_boom_3_data.rotation_sequence[0] = 1;
       CSYS_boom_3_data.rotation_sequence[1] = 2;

       DATUM_CSYS_DATA CSYS_boom_4_data;

       CSYS_boom_4_data.offset_x = Link_Dimensions_obj.Boom_side_len_T;
       CSYS_boom_4_data.offset_y = 0.0;
       CSYS_boom_4_data.offset_z = 0.0;
       CSYS_boom_4_data.angle_x = 0.0;
       CSYS_boom_4_data.angle_y = 0.0;
       CSYS_boom_4_data.angle_z = 0.0;
       CSYS_boom_4_data.transform_sequence = true;
       CSYS_boom_4_data.rotation_sequence[0] = 1;
       CSYS_boom_4_data.rotation_sequence[1] = 2;

       DATUM_CSYS_DATA CSYS_boom_5_data;

       CSYS_boom_5_data.offset_x =
0.9*Link_Dimensions_obj.Boom_side_len_T;
       CSYS_boom_5_data.offset_y = 0.0;
       CSYS_boom_5_data.offset_z = 0.0;
       CSYS_boom_5_data.angle_x = 0.0;
       CSYS_boom_5_data.angle_y = 0.0;
       CSYS_boom_5_data.angle_z = 0.0;
       CSYS_boom_5_data.transform_sequence = true;
       CSYS_boom_5_data.rotation_sequence[0] = 1;
       CSYS_boom_5_data.rotation_sequence[1] = 2;

       DATUM_CSYS_DATA CSYS_boom_6_data;

       CSYS_boom_6_data.offset_x = -
0.05*Link_Dimensions_obj.Boom_side_len_T;
       CSYS_boom_6_data.offset_y = 0.0;
       CSYS_boom_6_data.offset_z = 0.0;
       CSYS_boom_6_data.angle_x = 0.0;
       CSYS_boom_6_data.angle_y = 0.0;
       CSYS_boom_6_data.angle_z = 0.0;
       CSYS_boom_6_data.transform_sequence = true;
       CSYS_boom_6_data.rotation_sequence[0] = 1;
       CSYS_boom_6_data.rotation_sequence[1] = 2;

/*
CREATE PART FILE (NEW)
*/

       char
       Boom_part_path[UF_CFI_MAX_FILE_NAME_SIZE] =
"D:\\NX Files 2010\\Parts\\Boom\\Boom_New.prt";
```

```
       Boom_part = wub_Create_New_Part_File(Boom_part_path);

       DATUM_CSYS_boom_0 =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_boom_0_data,NULL_TAG);
       DATUM_CSYS_boom_1 =
       wub_Create_DATUM_CSYS_Offset_Method(CSYS_boom_1_data,
                                  DATUM_CSYS_boom_0);
       wub_set_wcs(DATUM_CSYS_boom_1);

/*
IMPORT DATA POINTS FROM FILE AND CONSTRUCT THE SPLINE LINES
*/
        tag_t
              Boom_Spline1_Orig,
              Boom_Spline2_Orig,
              Boom_Spline1_Ofstd,
              Boom_Spline2_Ofstd;

boom_csv_file1_orig = fopen(NX_BOOM_VEC_1_ORIGINAL);
Boom_Spline1_Orig = wub_Create_Fit_Spline_on_WCS(boom_csv_file1_orig);
fclose(boom_csv_file1_orig);

boom_csv_file1_ofstd = fopen(NX_BOOM_VEC_1_OFFESETTED);
Boom_Spline1_Ofstd = wub_Create_Fit_Spline_on_WCS(boom_csv_file1_ofstd);
fclose(boom_csv_file1_ofstd);

DATUM_CSYS_boom_2 = wub_Create_DATUM_CSYS_Offset_Method(CSYS_boom_2_data,
                                                    DATUM_CSYS_boom_1);
DATUM_CSYS_boom_3 = wub_Create_DATUM_CSYS_Offset_Method(CSYS_boom_3_data,
                                                    DATUM_CSYS_boom_1);
DATUM_CSYS_boom_4 = wub_Create_DATUM_CSYS_Offset_Method(CSYS_boom_4_data,
                                                    DATUM_CSYS_boom_3);
DATUM_CSYS_boom_5 = wub_Create_DATUM_CSYS_Offset_Method(CSYS_boom_5_data,
                                                    DATUM_CSYS_boom_3);
DATUM_CSYS_boom_6 = wub_Create_DATUM_CSYS_Offset_Method(CSYS_boom_6_data,
                                                    DATUM_CSYS_boom_5);

wub_set_wcs(DATUM_CSYS_boom_3);
boom_csv_file2_orig = fopen(NX_BOOM_VEC_2_ORIGINAL);
Boom_Spline2_Orig = wub_Create_Fit_Spline_on_WCS(boom_csv_file2_orig);
fclose(boom_csv_file2_orig);

boom_csv_file2_ofstd = fopen(NX_BOOM_VEC_2_OFFESETTED);
Boom_Spline2_Ofstd = wub_Create_Fit_Spline_on_WCS(boom_csv_file2_ofstd);
fclose(boom_csv_file2_ofstd);

/*
CREATE MIRROR PLANES
*/
       tag_t
              CSYS1_YZ_datum_plane,
              CSYS1_ZX_datum_plane,
              CSYS2_YZ_datum_plane,
              CSYS2_ZX_datum_plane,
              CSYS3_YZ_datum_plane,
              CSYS3_ZX_datum_plane,
              CSYS5_YZ_datum_plane,
              CSYS6_YZ_datum_plane;

CSYS1_ZX_datum_plane =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_boom_1, zx_plane);
CSYS2_YZ_datum_plane =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_boom_2, yz_plane);
```

```
CSYS2_ZX_datum_plane =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_boom_2, zx_plane);
CSYS3_YZ_datum_plane =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_boom_3, yz_plane);
CSYS3_ZX_datum_plane =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_boom_3, zx_plane);
CSYS5_YZ_datum_plane =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_boom_5, yz_plane);
CSYS6_YZ_datum_plane =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_boom_6, yz_plane);

/*
MIRROR SPLINE LINES ABOUT PLANES
*/
       tag_t
              Boom_Spline1_Orig_mirrored,
              Boom_Spline1_Ofstd_mirrored,
              Boom_Spline2_Orig_mirrored,
              Boom_Spline2_Ofstd_mirrored;

Boom_Spline1_Orig_mirrored =
wub_Mirror_a_Curve_through_a_Plane(Boom_Spline1_Orig,
                                   CSYS1_ZX_datum_plane);
Boom_Spline1_Ofstd_mirrored =
wub_Mirror_a_Curve_through_a_Plane(Boom_Spline1_Ofstd,
                                   CSYS1_ZX_datum_plane);
Boom_Spline2_Orig_mirrored =
wub_Mirror_a_Curve_through_a_Plane(Boom_Spline2_Orig,
                                   CSYS3_ZX_datum_plane);
Boom_Spline2_Ofstd_mirrored =
wub_Mirror_a_Curve_through_a_Plane(Boom_Spline2_Ofstd,
                                   CSYS3_ZX_datum_plane);
/*
TRIM LINES WITH PLANES
*/
       tag_t
              Boom_Spline1_Orig_trmd,
              Boom_Spline1_Ofstd_trmd,
              Boom_Spline2_Orig_trmd,
              Boom_Spline2_Ofstd_trmd,
              Boom_Spline1_Orig_mrrd_trmd,
              Boom_Spline1_Ofstd_mrrd_trmd,
              Boom_Spline2_Orig_mrrd_trmd,
              Boom_Spline2_Ofstd_mrrd_trmd;

Boom_Spline1_Orig_trmd =
wub_Trim_Curve_by_Datum_Plane(Boom_Spline1_Orig,CSYS3_ZX_datum_plane,2);
Boom_Spline1_Ofstd_trmd =
wub_Trim_Curve_by_Datum_Plane(Boom_Spline1_Ofstd,CSYS3_ZX_datum_plane,2);
Boom_Spline2_Orig_trmd = wub_Trim_Curve_by_Datum_Plane(Boom_Spline2_Orig,
                                                 CSYS2_ZX_datum_plane,
                                                 1);
Boom_Spline2_Ofstd_trmd =
wub_Trim_Curve_by_Datum_Plane(Boom_Spline2_Ofstd,CSYS2_ZX_datum_plane,1);
Boom_Spline1_Orig_mrrd_trmd =
wub_Trim_Curve_by_Datum_Plane(Boom_Spline1_Orig_mirrored,
                                   CSYS3_YZ_datum_plane,
                                   2);
Boom_Spline1_Ofstd_mrrd_trmd =
wub_Trim_Curve_by_Datum_Plane(Boom_Spline1_Ofstd_mirrored,
                                   CSYS3_YZ_datum_plane,
                                   2);
```

```
Boom_Spline2_Orig_mrrd_trmd =
wub_Trim_Curve_by_Datum_Plane(Boom_Spline2_Orig_mirrored,
                           CSYS2_YZ_datum_plane,
                           1);
Boom_Spline2_Ofstd_mrrd_trmd =
wub_Trim_Curve_by_Datum_Plane(Boom_Spline2_Ofstd_mirrored,
                           CSYS2_YZ_datum_plane,
                           1);

// FURTHER MODIFICATION FOR JOINT "J2"

Boom_Spline2_Orig_trmd =
wub_Trim_Curve_by_Datum_Plane(Boom_Spline2_Orig_trmd,
                           CSYS5_YZ_datum_plane,
                           2);
Boom_Spline2_Ofstd_trmd =
wub_Trim_Curve_by_Datum_Plane(Boom_Spline2_Ofstd_trmd,
                           CSYS5_YZ_datum_plane,
                           2);
Boom_Spline2_Orig_mrrd_trmd =
wub_Trim_Curve_by_Datum_Plane(Boom_Spline2_Orig_mrrd_trmd,
                           CSYS5_YZ_datum_plane,
                           2);
Boom_Spline2_Ofstd_mrrd_trmd =
wub_Trim_Curve_by_Datum_Plane(Boom_Spline2_Ofstd_mrrd_trmd,
                           CSYS5_YZ_datum_plane,
                           2);

/*
BRIDGING TRIMMED SPLINES
*/
       tag_t
              BRIDGE_Orig_Top,
              BRIDGE_Ofstd_Top,
              BRIDGE_Orig_Down,
              BRIDGE_Ofstd_Down;

BRIDGE_Orig_Top = wub_Bridge_Curves(Boom_Spline1_Orig_trmd,
Boom_Spline2_Orig_trmd,0,0,1,0);
BRIDGE_Ofstd_Top = wub_Bridge_Curves(Boom_Spline1_Ofstd_trmd,
Boom_Spline2_Ofstd_trmd,0,0,1,0);
BRIDGE_Orig_Down = wub_Bridge_Curves(Boom_Spline1_Orig_mrrd_trmd,
Boom_Spline2_Orig_mrrd_trmd,0,0,1,0);
BRIDGE_Ofstd_Down = wub_Bridge_Curves(Boom_Spline1_Ofstd_mrrd_trmd,
Boom_Spline2_Ofstd_mrrd_trmd,0,0,1,0);

       tag_t
              BRIDGE_J1_Orig,
              BRIDGE_J1_Ofstd,
              BRIDGE_J2_Orig,
              BRIDGE_J2_Ofstd;

       BRIDGE_J1_Orig =
wub_Bridge_Curves(Boom_Spline1_Orig_trmd,Boom_Spline1_Orig_mrrd_trmd,1,0,
0,0);
       BRIDGE_J1_Ofstd =
wub_Bridge_Curves(Boom_Spline1_Ofstd_trmd,Boom_Spline1_Ofstd_mrrd_trmd,1,
0,0,0);
       BRIDGE_J2_Orig =
wub_Bridge_Curves(Boom_Spline2_Orig_trmd,Boom_Spline2_Orig_mrrd_trmd,0,1,
1,1);
```

```
	BRIDGE_J2_Ofstd =
wub_Bridge_Curves(Boom_Spline2_Ofstd_trmd,Boom_Spline2_Ofstd_mrrd_trmd,0,
1,1,1);

/*
JOIN LINES TO FORM GUIDES
*/
	tag_t
		GUIDE_Ogig_Top_joined,
		GUIDE_Ogig_Down_joined,
		GUIDE_Ofstd_Total_Joined,

GUIDE_Ofstd_total_curves[7] =	{Boom_Spline1_Ofstd_trmd,
				BRIDGE_Ofstd_Top,
				Boom_Spline2_Ofstd_trmd,
				BRIDGE_J2_Ofstd,
				Boom_Spline2_Ofstd_mrrd_trmd,
				BRIDGE_Ofstd_Down,
				Boom_Spline1_Ofstd_mrrd_trmd};
	tag_t
		GUIDE_Orig_Top[3] = {Boom_Spline1_Orig_trmd,
				BRIDGE_Orig_Top,
				Boom_Spline2_Orig_trmd},
		GUIDE_Orig_Down[3] = {Boom_Spline1_Orig_mrrd_trmd,
				BRIDGE_Orig_Down,
				Boom_Spline2_Orig_mrrd_trmd};

	GUIDE_Ofstd_Total_Joined =
wub_Join_Curves(GUIDE_Ofstd_total_curves,7);
	GUIDE_Ogig_Top_joined = wub_Join_Curves(GUIDE_Orig_Top,3);
	GUIDE_Ogig_Down_joined = wub_Join_Curves(GUIDE_Orig_Down,3);

/*
PROJECTING GUIDE ONTO THE XY PLANE
*/
	tag_t
		PROJECTED_J1_Bridge_Ofstd,
		PROJECTED_GUIDE_Ofstd_Total_Joined,
		PROJECTED_top_Bridge;

PROJECTED_GUIDE_Ofstd_Total_Joined =
wub_Create_Projected_Curve(GUIDE_Ofstd_Total_Joined,
DATUM_CSYS_boom_1,xy_plane);
PROJECTED_J1_Bridge_Ofstd = wub_Create_Projected_Curve(BRIDGE_J1_Ofstd,
DATUM_CSYS_boom_1,xy_plane);
PROJECTED_top_Bridge = wub_Create_Projected_Curve(BRIDGE_Ofstd_Top,
DATUM_CSYS_boom_1,xy_plane);

/*
CREATING POINTS FOR PROFILE GENERATION
*/
	tag_t
		POINT_Top_ofstd,
		PROJECTED_POINT_Top_ofstd,
		POINT_Top_orig,
		POINT_Down_orig,
		POINT_J2_Top_orig,
		POINT_J2_Down_orig;

POINT_Top_ofstd = wub_Point_from_Spline(Boom_Spline1_Ofstd_trmd,0);
PROJECTED_POINT_Top_ofstd = wub_Create_Projected_Curve(POINT_Top_ofstd,
DATUM_CSYS_boom_1,xy_plane);
POINT_Top_orig = wub_Point_from_Spline(Boom_Spline1_Orig_trmd,0);
```

```
POINT_Down_orig = wub_Point_from_Spline(Boom_Spline1_Orig_mrrd_trmd,0);
POINT_J2_Top_orig = wub_Point_from_Spline(Boom_Spline2_Orig_trmd,1);
POINT_J2_Down_orig =
wub_Point_from_Spline(Boom_Spline2_Orig_mrrd_trmd,1);

/*
CREATING PROFILE CURVES (LINES)
*/

       tag_t
              SECTION_Boom_profile_top,
              SECTION_Boom_profile_side,
              SECTION_J2_Boom_profile_side;

SECTION_Boom_profile_top  = wub_Lines_from_two_points(POINT_Top_ofstd,
PROJECTED_POINT_Top_ofstd);
SECTION_Boom_profile_side  = wub_Lines_from_two_points(POINT_Top_orig,
POINT_Down_orig);
SECTION_J2_Boom_profile_side  =
wub_Lines_from_two_points(POINT_J2_Top_orig, POINT_J2_Down_orig);

/*
CREATING SWEEP SHEET SURFACES AND BOUNDED PLANES
*/

       tag_t
              SHEET_Boom_top,
              SHEET_J1,
              SHEET_Boom_side;

SHEET_Boom_top = wub_SWEEP_2_guides(GUIDE_Ofstd_Total_Joined,
PROJECTED_GUIDE_Ofstd_Total_Joined, SECTION_Boom_profile_top);
SHEET_J1 = wub_SWEEP_2_guides(BRIDGE_J1_Ofstd, PROJECTED_J1_Bridge_Ofstd,
SECTION_Boom_profile_top);
SHEET_Boom_side = wub_SWEEP_2_guides(GUIDE_Ogig_Top_joined,
GUIDE_Ogig_Down_joined, SECTION_Boom_profile_side);

tag_t
String_bplnae_J1[2] = {BRIDGE_J1_Orig,SECTION_Boom_profile_side},
String_bplnae_J2[2] = {BRIDGE_J2_Orig,SECTION_J2_Boom_profile_side},
BPLANE_J1,
BPLANE_J2;

BPLANE_J1 = wub_BPLANE(String_bplnae_J1);
BPLANE_J2 = wub_BPLANE(String_bplnae_J2);

/*
CREATE SKETCH
*/
       char
              sketch_name1[30] = {"BOOM_JOINT_#2"};

       tag_t
SKETCH_J2_Adopter = wub_SKETCHES_J2_adopter(sketch_name1,
                                        DATUM_CSYS_boom_4,
                                        xy_plane,
                                        xc_axis);

       tag_t
              arc_J2_outside;
       double J2_center[3] = {0.,0.,0.},
              J2_outer_radius = 50.;
```

```
	arc_J2_outside =
wub_ARC_Center_Radius(DATUM_CSYS_boom_4,xy_plane,J2_outer_radius,J2_cente
r,0.,360.);
	tag_t
		LINE_J2_plate_top,
		LINE_J2_plate_Down;

LINE_J2_plate_top = wub_Create_Projected_Curve(Boom_Spline2_Orig_trmd,
DATUM_CSYS_boom_1,xy_plane);
LINE_J2_plate_Down =
wub_Create_Projected_Curve(Boom_Spline2_Orig_mrrd_trmd,
DATUM_CSYS_boom_1,xy_plane);
//Further modify by trimming
LINE_J2_plate_top = wub_Trim_Curve_by_Datum_Plane(LINE_J2_plate_top,
CSYS6_YZ_datum_plane,1);
LINE_J2_plate_Down = wub_Trim_Curve_by_Datum_Plane(LINE_J2_plate_Down,
CSYS6_YZ_datum_plane,1);

	tag_t
		POINT_J2_top_left_end,
		POINT_J2_down_left_end,
		POINT_J2_top_right_end,
		POINT_J2_down_right_end;

POINT_J2_top_left_end = wub_Point_from_Spline(LINE_J2_plate_top,0);
POINT_J2_down_left_end = wub_Point_from_Spline(LINE_J2_plate_Down,0);
POINT_J2_top_right_end = wub_Point_from_Spline(LINE_J2_plate_top,1);
POINT_J2_down_right_end = wub_Point_from_Spline(LINE_J2_plate_Down,1);

	tag_t
		LINE_J2_left,
		LINE_J2_connecting_top,
		LINE_J2_connecting_Down;

LINE_J2_left = wub_Lines_from_two_points(POINT_J2_top_left_end,
POINT_J2_down_left_end);
LINE_J2_connecting_top = wub_Lines_Point_Tangent(POINT_J2_top_right_end,
arc_J2_outside,DATUM_CSYS_boom_4, xy_plane);
LINE_J2_connecting_Down =
wub_Lines_Point_Tangent(POINT_J2_down_right_end,
arc_J2_outside,DATUM_CSYS_boom_4, xy_plane);

	tag_t
		POINT_tangent1,
		POINT_tangent2;

	POINT_tangent1 = wub_Point_from_Spline(LINE_J2_connecting_top,1);
	POINT_tangent2 = wub_Point_from_Spline(LINE_J2_connecting_Down,1);

	tag_t ARC_J2_secant =
wub_ARC_Point_Point_Radius(DATUM_CSYS_boom_4,xy_plane,J2_outer_radius,J2_
center,POINT_tangent1, POINT_tangent2);

	tag_t
J2_joined_curves[6] = {LINE_J2_plate_top,
			LINE_J2_connecting_top,
			ARC_J2_secant,
			LINE_J2_connecting_Down,
			LINE_J2_plate_Down,
			LINE_J2_left};
	tag_t JOINED_J2_sketch = wub_Join_Curves(J2_joined_curves,6);

	UF_SKET_add_objects(SKETCH_J2_Adopter,1,&LINE_J2_plate_top);
```

```
       UF_SKET_add_objects(SKETCH_J2_Adopter,1,&LINE_J2_plate_Down);
       UF_SKET_add_objects(SKETCH_J2_Adopter,1,&LINE_J2_left);
       UF_SKET_add_objects(SKETCH_J2_Adopter,1,&LINE_J2_connecting_top);
       UF_SKET_add_objects(SKETCH_J2_Adopter,1,&LINE_J2_connecting_Down);
       UF_SKET_add_objects(SKETCH_J2_Adopter,1,&ARC_J2_secant);

       UF_SKET_update_sketch(SKETCH_J2_Adopter);
       UF_SKET_terminate_sketch();
       UF_DISP_set_highlight(SKETCH_J2_Adopter,1);

/*
EXTRUDE SKETCH OF JOINT J2
*/
       char *limit_J2[2] = {"63.","83."};
       tag_t EXTRUDED_J2 = wub_Extrude(JOINED_J2_sketch,limit_J2);

tag_t PROJECTED_csysline =
wub_Create_Projected_Curve(Boom_Spline2_Orig_trmd,
                           DATUM_CSYS_boom_1,
                           xy_plane);

tag_t POINT_csys7 = wub_Point_from_Spline(PROJECTED_csysline,0);

       double
              point_csys7_d[3];
UF_CURVE_ask_point_data(POINT_csys7,point_csys7_d);
UF_CSYS_map_point(UF_CSYS_ROOT_COORDS,point_csys7_d,UF_CSYS_ROOT_WCS_COOR
DS,point_csys7_d);

       DATUM_CSYS_DATA CSYS_boom_7_data;

       CSYS_boom_7_data.offset_x = point_csys7_d[0] + 80.;
       CSYS_boom_7_data.offset_y = point_csys7_d[1] + 40.;
       CSYS_boom_7_data.offset_z = point_csys7_d[2];
       CSYS_boom_7_data.angle_x = 0.0;
       CSYS_boom_7_data.angle_y = 0.0;
       CSYS_boom_7_data.angle_z = 0.0;
       CSYS_boom_7_data.transform_sequence = true;
       CSYS_boom_7_data.rotation_sequence[0] = 1;
       CSYS_boom_7_data.rotation_sequence[1] = 2;

       DATUM_CSYS_boom_7 =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_boom_7_data,DATUM_CSYS_boom_3);

       DATUM_CSYS_DATA CSYS_boom_10_data;

       CSYS_boom_10_data.offset_x = 80.;
       CSYS_boom_10_data.offset_y = 0.0;
       CSYS_boom_10_data.offset_z = 0.0;
       CSYS_boom_10_data.angle_x = 0.0;
       CSYS_boom_10_data.angle_y = 0.0;
       CSYS_boom_10_data.angle_z = 0.0;
       CSYS_boom_10_data.transform_sequence = true;
       CSYS_boom_10_data.rotation_sequence[0] = 1;
       CSYS_boom_10_data.rotation_sequence[1] = 2;

DATUM_CSYS_boom_10 =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_boom_10_data,DATUM_CSYS_boom_7);
tag_t DPLANE_trimmer =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_boom_10, yz_plane);
PROJECTED_csysline = wub_Trim_Curve_by_Datum_Plane(PROJECTED_csysline,
DPLANE_trimmer,2);
```

```
	tag_t
		PROJECTED_LOWER_Bridge =
wub_Create_Projected_Curve(BRIDGE_Ofstd_Down,
DATUM_CSYS_boom_1,xy_plane);
	tag_t
POINT_J12_left_t = wub_Point_from_Spline(PROJECTED_LOWER_Bridge,0),
POINT_J12_right_t = wub_Point_from_Spline(PROJECTED_LOWER_Bridge,1);

	double
		POINT_J12_left_d[3],
		POINT_J12_right_d[3];
	UF_CURVE_ask_point_data(POINT_J12_left_t,POINT_J12_left_d);
	UF_CURVE_ask_point_data(POINT_J12_right_t,POINT_J12_right_d);

	double
POINT_J2_hinge_d[3];
POINT_J2_hinge_d[0] = 0.5*(POINT_J12_left_d[0] + POINT_J12_right_d[0]);
POINT_J2_hinge_d[1] = 0.5*(POINT_J12_left_d[1] + POINT_J12_right_d[1]);
POINT_J2_hinge_d[2] = 0.5*(POINT_J12_left_d[2] + POINT_J12_right_d[2]);

UF_CSYS_map_point(UF_CSYS_ROOT_COORDS,point_csys7_d,UF_CSYS_ROOT_WCS_COOR
DS,point_csys7_d);
UF_CSYS_map_point(UF_CSYS_ROOT_COORDS,point_csys7_d,UF_CSYS_ROOT_WCS_COOR
DS,point_csys7_d);

	DATUM_CSYS_DATA CSYS_boom_8_data;

	CSYS_boom_8_data.offset_x = POINT_J2_hinge_d[0];
	CSYS_boom_8_data.offset_y = POINT_J2_hinge_d[1];
	CSYS_boom_8_data.offset_z = POINT_J2_hinge_d[2];
	CSYS_boom_8_data.angle_x = 0.0;
	CSYS_boom_8_data.angle_y = 0.0;
	CSYS_boom_8_data.angle_z = 0.0;
	CSYS_boom_8_data.transform_sequence = true;
	CSYS_boom_8_data.rotation_sequence[0] = 1;
	CSYS_boom_8_data.rotation_sequence[1] = 2;

DATUM_CSYS_boom_8 =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_boom_8_data,DATUM_CSYS_boom_0);

	DATUM_CSYS_DATA CSYS_boom_9_data;

	CSYS_boom_9_data.offset_x = 0.0;
	CSYS_boom_9_data.offset_y = -25.;
	CSYS_boom_9_data.offset_z = 0.0;
	CSYS_boom_9_data.angle_x = 0.0;
	CSYS_boom_9_data.angle_y = 0.0;
	CSYS_boom_9_data.angle_z = 0.0;
	CSYS_boom_9_data.transform_sequence = true;
	CSYS_boom_9_data.rotation_sequence[0] = 1;
	CSYS_boom_9_data.rotation_sequence[1] = 2;

DATUM_CSYS_boom_9 =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_boom_9_data,DATUM_CSYS_boom_8);

	DATUM_CSYS_DATA CSYS_boom_11_data; // Location of Transition four-
bar

	CSYS_boom_11_data.offset_x = -Four-bar_dimensions_obj.link0_bo;
	CSYS_boom_11_data.offset_y = 0.;
	CSYS_boom_11_data.offset_z = 0.0;
	CSYS_boom_11_data.angle_x = 0.0;
	CSYS_boom_11_data.angle_y = 0.0;
```

```
        CSYS_boom_11_data.angle_z = 0.0;
        CSYS_boom_11_data.transform_sequence = true;
        CSYS_boom_11_data.rotation_sequence[0] = 1;
        CSYS_boom_11_data.rotation_sequence[1] = 2;

DATUM_CSYS_boom_11 =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_boom_11_data,
DATUM_CSYS_boom_4);

tag_t JOINED_J11 =
wub_SKETCH_J11(DATUM_CSYS_boom_7,xy_plane,PROJECTED_csysline,PROJECTED_to
p_Bridge);
        UF_DISP_set_highlight(JOINED_J11,1);
        char* limit_J11[2] = {"25.","50."};
        tag_t EXTRUDE_J11 = wub_Extrude(JOINED_J11, limit_J11);

tag_t JOINED_J12 =
wub_SKETCH_J12(DATUM_CSYS_boom_9,xy_plane,BRIDGE_Ofstd_Down);
        UF_DISP_set_highlight(JOINED_J12,1);
        char* limit_J12[2] = {"25.","50."};
        tag_t EXTRUDE_J12 = wub_Extrude(JOINED_J12, limit_J12);

        tag_t
                THICKEN_top,
                THICKEN_J1_rear,
                THICKEN_side,
                THICKEN_J1_side,
                THICKEN_J2_side;
        tag_t
                UNITED_solid;

        UF_MODL_create_thicken_sheet(SHEET_Boom_top,
                                     "0.",
                                     "7.0",
                                     UF_NULLSIGN,
                                     &THICKEN_top);
        UF_MODL_create_thicken_sheet(SHEET_J1,
                                     "0.",
                                     "-7.0",
                                     UF_NULLSIGN,
                                     &THICKEN_J1_rear);
        UF_MODL_create_thicken_sheet(SHEET_Boom_side,
                                     "0.",
                                     "-7.0",
                                     UF_NULLSIGN,
                                     &THICKEN_side);
        UF_MODL_create_thicken_sheet(BPLANE_J1,
                                     "0.",
                                     "-7.0",
                                     UF_NULLSIGN,
                                     &THICKEN_J1_side);
        UF_MODL_create_thicken_sheet(BPLANE_J2,
                                     "0.",
                                     "7.0",
                                     UF_NULLSIGN,
                                     &THICKEN_J2_side);
        tag_t
                SOLID_TOP,
                SOLID_J1_REAR,
                SOLID_SIDE,
                SOLID_J1_SIDE,
                SOLID_J2_SIDE;
```

```
       UF_CALL(UF_MODL_ask_feat_body(THICKEN_top, &SOLID_TOP));
       UF_CALL(UF_MODL_ask_feat_body(THICKEN_J1_rear, &SOLID_J1_REAR));
       UF_CALL(UF_MODL_ask_feat_body(THICKEN_side, &SOLID_SIDE));
       UF_CALL(UF_MODL_ask_feat_body(THICKEN_J1_side, &SOLID_J1_SIDE));
       UF_CALL(UF_MODL_ask_feat_body(THICKEN_J2_side, &SOLID_J2_SIDE));

       UNITED_solid = wub_UNITE_SOLIDS(SOLID_TOP,SOLID_J1_REAR);
       UNITED_solid = wub_UNITE_SOLIDS(UNITED_solid,SOLID_SIDE);
       UNITED_solid = wub_UNITE_SOLIDS(UNITED_solid,SOLID_J1_SIDE);
       UNITED_solid = wub_UNITE_SOLIDS(UNITED_solid,SOLID_J2_SIDE);

       UNITED_solid = wub_UNITE_SOLIDS(UNITED_solid,EXTRUDED_J2);
       UNITED_solid = wub_UNITE_SOLIDS(UNITED_solid,EXTRUDE_J11);
       UNITED_solid = wub_UNITE_SOLIDS(UNITED_solid,EXTRUDE_J12);

/*
CREATE CYLINDERS
*/

       wub_set_wcs(DATUM_CSYS_boom_1);
       UF_DISP_set_highlight(DATUM_CSYS_boom_1,1);
       double
              origin_bool_J1[3] = {0.,0.,0.},
              dirction_bool_J1[3] = {0.,0.,1};
       tag_t
              bool_J1_cylinder_outside,
              bool_J1_cylinder_inside;
       UF_CSYS_map_point(UF_CSYS_ROOT_WCS_COORDS,
                            origin_bool_J1,
                            UF_CSYS_ROOT_COORDS,
                            origin_bool_J1);
       tag_t
              UNITED_solid_body =
       UF_MODL_create_cylinder(UF_POSITIVE,
                            UNITED_solid,
                            origin_bool_J1,
                            "100.",
                            "75.",
                            dirction_bool_J1,
                            &bool_J1_cylinder_outside);
       UNITED_solid =
wub_UNITE_SOLIDS(UNITED_solid,bool_J1_cylinder_outside);
       UF_MODL_create_cylinder(UF_NEGATIVE,
                            UNITED_solid,
                            origin_bool_J1,
                            "110.",
                            "25.",
                            dirction_bool_J1,
                            &bool_J1_cylinder_inside);
/*
              +++++++++++++++++++++++++++++++++++++++++++++++++
*/

       wub_set_wcs(DATUM_CSYS_boom_4);
       double
              origin_bool_J2[3] = {0.,0.,0.},
              dirction_bool_J2[3] = {0.,0.,1};
       tag_t
              bool_J2_cylinder_outside,
              bool_J2_cylinder_inside;
       UF_CSYS_map_point(UF_CSYS_ROOT_WCS_COORDS
,origin_bool_J2,UF_CSYS_ROOT_COORDS,origin_bool_J2);
```

```
       UF_MODL_create_cylinder(UF_NEGATIVE,
                          UNITED_solid,
                          origin_bool_J2,
                          "110.",
                          "35.",
                          dirction_bool_J2,
                          &bool_J2_cylinder_inside);

/*
       ++++++++++++++++++++++++++++++++++++++++++++++
*/
       wub_set_wcs(DATUM_CSYS_boom_7);
       UF_DISP_set_highlight(DATUM_CSYS_boom_7,1);

       double
              origin_bool_J10[3] = {0.,0.,0.},
              dirction_bool_J10[3] = {0.,0.,1};
       tag_t
              bool_J10_cylinder_outside,
              bool_J10_cylinder_inside;
       UF_CSYS_map_point(UF_CSYS_ROOT_WCS_COORDS
,origin_bool_J10,UF_CSYS_ROOT_COORDS,origin_bool_J10);

       UF_MODL_create_cylinder(UF_NEGATIVE,
                          UNITED_solid,
                          origin_bool_J10,
                          "110.",
                          "35.",
                          dirction_bool_J10,
                          &bool_J10_cylinder_inside);

/*
       ++++++++++++++++++++++++++++++++++++++++++++++
*/
       wub_set_wcs(DATUM_CSYS_boom_9);
       UF_DISP_set_highlight(DATUM_CSYS_boom_9,1);

       double
              origin_bool_J11[3] = {0.,0.,0.},
              dirction_bool_J11[3] = {0.,0.,1};
       tag_t
              bool_J11_cylinder_outside,
              bool_J11_cylinder_inside;
       UF_CSYS_map_point(UF_CSYS_ROOT_WCS_COORDS
,origin_bool_J11,UF_CSYS_ROOT_COORDS,origin_bool_J11);

       UF_MODL_create_cylinder(UF_NEGATIVE,
                          UNITED_solid,
                          origin_bool_J11,
                          "110.",
                          "35.",
                          dirction_bool_J11,
                          &bool_J11_cylinder_inside);
/*
       +++++++++++++++++++++++++++++++++++++++++++
*/
       tag_t
              mirror_plane,
              mirrored_body,
              final_boom_solid;
       mirror_plane =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_boom_1,xy_plane);
       UF_MODL_create_mirror_body(UNITED_solid,
                                       mirror_plane,
                                       &mirrored_body);

       UF_MODL_unite_bodies(UNITED_solid,mirrored_body);

       UF_terminate();
       }

int ufusr_ask_unload(void)
       {
       return(UF_UNLOAD_IMMEDIATELY);
       }
```

A2.2 Stick API CAD Programming

A.2.2.1 Stick Header Files

Prototypeff.h

```
#define UF_CALL(X) (report_error( __FILE__, __LINE__, #X, (X)))
#define RT(X) report_object_type_and_subtype(#X, X)


#define xc_axis 1
#define yc_axis 2
#define zc_axis 3
#define xy_plane 1
#define yz_plane 2
#define zx_plane 3

#define NX_BOOM_VEC_1_ORIGINAL
"D:\\My Dropbox\\ForNX\\Delimited Files\\Boom_vec1_Original.dat","r"
#define NX_BOOM_VEC_1_OFFESETTED
"D:\\My Dropbox\\ForNX\\Delimited Files\\Boom_vec1_Offsetted.dat","r"
#define NX_BOOM_VEC_2_ORIGINAL
"D:\\My Dropbox\\ForNX\\Delimited Files\\Boom_vec2_Original.dat","r"
#define NX_BOOM_VEC_2_OFFESETTED
"D:\\My Dropbox\\ForNX\\Delimited Files\\Boom_vec2_Offsetted.dat","r"


#define NX_STICK_VEC_MEGRGED
"D:\\My Dropbox\\ForNX\\Delimited Files\\NX_Stick_vecc.dat","r"
#define NX_STICK_LINEARIZED
"D:\\My Dropbox\\ForNX\\Delimited
Files\\NX_Stick_Linearized_Data_orig.dat","r"
#define NX_STICK_LINEARIZED_OFFSETTED
"D:\\My Dropbox\\ForNX\\Delimited
Files\\NX_Stick_Linearized_Data_offstd.dat","r"

typedef struct _iobuf FILE;



struct DATUM_CSYS_DATA{
       double offset_x;
       double offset_y;
       double offset_z;
       double angle_x;
       double angle_y;
       double angle_z;
       bool transform_sequence;
       int rotation_sequence[2];
       };
struct LINK_DIMENSIONS{
       double Boom_shot_len_l1;
       double Boom_defct_ang_betta;
       double Boom_side_len_T;
       double Stick_len_l2;
       double Stick_ang_J2;
       double J2_left;
       double J2_right;
       double Stick_ang_J8;
```

```
	double J8_lfet;
	double J8_right;
	double Stick_ang_J9;
	double J9_up;
	double J9_lower;
	double Stick_ang_J3;
	double J3_up;
	double J3_lower;
	double Dist_2_J2andJ8_on_Stick;
	double Dist_2_J10_on_Boom;
	double Dist_2_J11_on_Boom;
	double Stick_tail_len;
	double Stick_forward_len;
	};

struct PIN_DIMENSIONS{
	double Pin1;
	double Pin2;
	double Pin3;
	double Pin4;
	double Pin5;
	double Pin6;
	double Pin7;
	double Pin8;
	double Pin9;
	double Pin10;
	double Pin11;
	double Pin12;
	};

struct OPERATIONAL_CONFIGURATION{
	double Boom_oper_ang_dig1;
	double Boom_matrix_1_RB1;
	double Boom_matrix_1_RB2;
	double Stick_matrix_RS;
	double Four-bar_teta_1;
	double Four-bar_teta_2;
	double Four-bar_teta_3;
	};
struct STICK_DATA{
	double Y_first;
	double Y_middle;
	double Y_last;
	double base_min;
	};

int report_error( char *file, int line, char *call, int irc);
int report_object_type_and_subtype(char *name, tag_t object);
tag_t wub_Create_New_Part_File(char
file_path[UF_CFI_MAX_FILE_NAME_SIZE]);
tag_t wub_CSYS_origin_and_direction(void);
tag_t Extract_Smart_tag_of_Datum_CSYS(tag_t tag_DATUM_CSYS);
tag_t Extract_daxis_tag_of_Datum_CSYS(tag_t Datum_CSYS_tag, int
Axis_Num);
tag_t Extract_dplane_tag_of_Datum_CSYS(tag_t Datum_CSYS_tag, int
Plane_Num);
tag_t wub_set_wcs(tag_t target_DATUM_wcs_CSYS);
tag_t wub_CSYS_offset(tag_t referece_datum_CSYS, const double
linear_offset[3], const double angular_offset[3], bool
operation_sequence);
tag_t wub_Ceate_Datum_Plane_Offset(tag_t Referece_DATUM_CSYS, int
Axis_about);
```

```
tag_t wub_Create_Projected_Curve(tag_t curve_tag, tag_t Datum_CSYS_tag,
int Plane_Num);
tag_t wub_Sketch_boom_profile1(tag_t datum);
tag_t wub_Create_Fit_Spline_on_WCS(FILE *csv_file);
tag_t do_ugopen_api(void);
tag_t wub_Create_DATUM_CSYS_Offset_Method(DATUM_CSYS_DATA
Transformation_Data, tag_t Reference_CSYS);
int wub_LinkDim_Data_Importer(FILE *dat_filepath, LINK_DIMENSIONS
*pLink_Dimensions_obj);
int wub_OperConfig_Data_Importer(FILE *dat_filepath,
OPERATIONAL_CONFIGURATION *pOperConfig_obj);
tag_t wub_Mirror_a_Curve_through_a_Plane(tag_t Curve_Tag, tag_t
Plane_Tag);
tag_t wub_Trim_Curve_by_Datum_Plane(tag_t Curve_Tag, tag_t
Datum_Plane_Tag,int which_end);
tag_t wub_Trim_Curve_by_Curves(tag_t Target_curve_Tag, tag_t Tool_curve1,
tag_t Tool_curve2);
tag_t wub_Bridge_Curves(tag_t Trimmed_Curve_1_Tag, tag_t
Trimmed_Curve_2_Tag, int Reverse1, int Reverse2,int par1, int par2);
tag_t wub_Lines_from_two_points(tag_t point1, tag_t point2);
tag_t wub_Lines_Point_Tangent(tag_t point, tag_t tangent,tag_t
Reference_CSYS, int Plane_Num, int Quadrant);
tag_t wub_SWEEP_2_guides(tag_t Guide_s1, tag_t Guide_s2, tag_t Section);
tag_t wub_Join_Curves(tag_t *curves,int n);
tag_t wub_Point_from_Spline(tag_t curve_Tag, int line_end);
tag_t wub_BPLANE(tag_t Curve_String[2]);
tag_t wub_SKETCHES_J2_adopter(char name[30], tag_t Refrence_CSYS, int
Plane_num, int Axis_num);
tag_t THICKEN_Sheet(tag_t sheet_body_tag);
tag_t wub_ARC_on_sketch(tag_t Reference_CSYS);
tag_t wub_ARC_Center_Radius(tag_t Reference_CSYS,int Plane_num, double
radius, double arc_center[3],double start_ang, double end_ang);
tag_t wub_ARC_Point_Point_Radius(tag_t Reference_CSYS,int Plane_num,
double radius, double arc_center[3],tag_t p1, tag_t p2);
tag_t wub_Extrude(tag_t Connected_Curve, char* limit[2]);
tag_t wub_SKETCH_J11(tag_t Reference_CSYS,int Plane_num,tag_t line_Tag,
tag_t bridge_tag);
tag_t wub_SKETCH_J12(tag_t Reference_CSYS,int Plane_num,tag_t Curve_Tag);
tag_t wub_UNITE_SOLIDS(tag_t Target_Solid, tag_t Tool_Solid);

int wub_Stick_Data_Importer(FILE *dat_filepath, STICK_DATA
*Stick_data_Obj);
tag_t wub_Bridge_Arc_with_Line(tag_t arc, tag_t Line_Tag, int Reverse1,
int Reverse2,int par1, int par2);
tag_t wub_Trim_Curve_by_Curve(tag_t line_to_trim, tag_t boundary_line,int
line_end);
tag_t wub_EXTRACT_Object_out_of_Feature(tag_t item_Tag);
tag_t wub_SKETCH_J8S(tag_t Reference_CSYS,int Plane_num,tag_t Curve_Tag);
```

A2.2.2 Stick Main Codes Assembly:

Stick_Main_12.cpp

```
#include <stdio.h>
#include <string.h>
#include <uf.h>
#include <uf_ui.h>
#include <uf_part.h>
#include <uf_csys.h>
#include <uf_modl.h>
#include <uf_modl_primitives.h>
#include <uf_disp.h>
#include <uf_so.h>
#include <uf_curve.h>
#include <uf_obj.h>
#include <uf_object_types.h>
#include <uf_assem.h>
#include <uf_modl_datum_features.h>
#include <uf_defs.h>
#include <stdlib.h>
#include <malloc.h>
#include <uf_sket.h>


#include "Prototypeff.h"

void ufusr(char *param, int *retcode, int paramLen)
       {
       if (UF_CALL(UF_initialize())) return;

/*
       STICK PART CREATION
*/

tag_t Stick_Part;
char Stick_part_path[UF_CFI_MAX_FILE_NAME_SIZE] =
"D:\\NX Files 2010\\Parts\\StickII.prt";
       Stick_Part = wub_Create_New_Part_File(Stick_part_path);

       STICK_DATA
              Stick_data_Obj;

       FILE
              *stick_data_file;

       stick_data_file = fopen(NX_STICK_LINEARIZED);
       wub_Stick_Data_Importer(stick_data_file,&Stick_data_Obj);
       fclose(stick_data_file);


       LINK_DIMENSIONS
              Link_Dimensions_obj;
       OPERATIONAL_CONFIGURATION
              OperConfig_obj;

       FILE
              *NX_PinDims_file,
              *NX_LinkDims_file,
              *NX_OperConfig_file;
```

```
NX_LinkDims_file =
fopen("D:\\My Dropbox\\ForNX\\Delimited Files\\NX_LinkDims.dat","r");
wub_LinkDim_Data_Importer(NX_LinkDims_file, &Link_Dimensions_obj);
fclose(NX_LinkDims_file);
NX_OperConfig_file =
fopen("D:\\My Dropbox\\ForNX\\Delimited Files\\NX_OperConfig.dat","r");
wub_OperConfig_Data_Importer(NX_OperConfig_file, &OperConfig_obj);
fclose(NX_OperConfig_file);

/*
COORDINATE SYSTEM CREATION
*/

       DATUM_CSYS_DATA CSYS_Stick_SJ9_data;

       CSYS_Stick_SJ9_data.offset_x = 0.0;
       CSYS_Stick_SJ9_data.offset_y = 0.0;
       CSYS_Stick_SJ9_data.offset_z = 0.0;
       CSYS_Stick_SJ9_data.angle_x = 0.0;
       CSYS_Stick_SJ9_data.angle_y = 0.0;
       CSYS_Stick_SJ9_data.angle_z = 0.0;
       CSYS_Stick_SJ9_data.transform_sequence = true;
       CSYS_Stick_SJ9_data.rotation_sequence[0] = 1;
       CSYS_Stick_SJ9_data.rotation_sequence[1] = 2;

       tag_t
DATUM_CSYS_Stick_SJ9 =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_Stick_SJ9_data,NULL_TAG);

DATUM_CSYS_DATA CSYS_Stick_SJ3_data;

CSYS_Stick_SJ3_data.offset_x = Link_Dimensions_obj.Stick_tail_len +
Link_Dimensions_obj.Stick_forward_len;
       CSYS_Stick_SJ3_data.offset_y = 0.0;
       CSYS_Stick_SJ3_data.offset_z = 0.0;
       CSYS_Stick_SJ3_data.angle_x = 0.0;
       CSYS_Stick_SJ3_data.angle_y = 0.0;
       CSYS_Stick_SJ3_data.angle_z = 0.0;
       CSYS_Stick_SJ3_data.transform_sequence = true;
       CSYS_Stick_SJ3_data.rotation_sequence[0] = 1;
       CSYS_Stick_SJ3_data.rotation_sequence[1] = 2;

       tag_t
              DATUM_CSYS_Stick_SJ3 =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_Stick_SJ3_data,DATUM_CSYS_Stick_
SJ9);

       DATUM_CSYS_DATA CSYS_Stick_SJ6_data;

       CSYS_Stick_SJ6_data.offset_x = -200.0;
       CSYS_Stick_SJ6_data.offset_y = 0.0;
       CSYS_Stick_SJ6_data.offset_z = 0.0;
       CSYS_Stick_SJ6_data.angle_x = 0.0;
       CSYS_Stick_SJ6_data.angle_y = 0.0;
       CSYS_Stick_SJ6_data.angle_z = 0.0;
       CSYS_Stick_SJ6_data.transform_sequence = true;
       CSYS_Stick_SJ6_data.rotation_sequence[0] = 1;
       CSYS_Stick_SJ6_data.rotation_sequence[1] = 2;

       tag_t
```

```
             DATUM_CSYS_Stick_SJ6 =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_Stick_SJ6_data,DATUM_CSYS_Stick_
SJ3);

       DATUM_CSYS_DATA CSYS_Stick_SJ2_data;

       CSYS_Stick_SJ2_data.offset_x = Link_Dimensions_obj.Stick_tail_len;
       CSYS_Stick_SJ2_data.offset_y = -Stick_data_Obj.Y_middle;
       CSYS_Stick_SJ2_data.offset_z = 0.0;
       CSYS_Stick_SJ2_data.angle_x = 0.0;
       CSYS_Stick_SJ2_data.angle_y = 0.0;
       CSYS_Stick_SJ2_data.angle_z = 0.0;
       CSYS_Stick_SJ2_data.transform_sequence = true;
       CSYS_Stick_SJ2_data.rotation_sequence[0] = 1;
       CSYS_Stick_SJ2_data.rotation_sequence[1] = 2;

       tag_t
             DATUM_CSYS_Stick_SJ2 =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_Stick_SJ2_data,DATUM_CSYS_Stick_
SJ9);

DATUM_CSYS_DATA CSYS_Stick_SJ8_data;

CSYS_Stick_SJ8_data.offset_x = Link_Dimensions_obj.Stick_tail_len + 80;
       CSYS_Stick_SJ8_data.offset_y = Stick_data_Obj.Y_middle + 40;
       CSYS_Stick_SJ8_data.offset_z = 0.0;
       CSYS_Stick_SJ8_data.angle_x = 0.0;
       CSYS_Stick_SJ8_data.angle_y = 0.0;
       CSYS_Stick_SJ8_data.angle_z = 0.0;
       CSYS_Stick_SJ8_data.transform_sequence = true;
       CSYS_Stick_SJ8_data.rotation_sequence[0] = 1;
       CSYS_Stick_SJ8_data.rotation_sequence[1] = 2;

       tag_t
DATUM_CSYS_Stick_SJ8 =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_Stick_SJ8_data,DATUM_CSYS_Stick_
SJ9);

       DATUM_CSYS_DATA CSYS_Stick_SJ8_2ND_data;

       CSYS_Stick_SJ8_2ND_data.offset_x = 80;
       CSYS_Stick_SJ8_2ND_data.offset_y = 0;
       CSYS_Stick_SJ8_2ND_data.offset_z = 0.0;
       CSYS_Stick_SJ8_2ND_data.angle_x = 0.0;
       CSYS_Stick_SJ8_2ND_data.angle_y = 0.0;
       CSYS_Stick_SJ8_2ND_data.angle_z = 0.0;
       CSYS_Stick_SJ8_2ND_data.transform_sequence = true;
       CSYS_Stick_SJ8_2ND_data.rotation_sequence[0] = 1;
       CSYS_Stick_SJ8_2ND_data.rotation_sequence[1] = 2;

       tag_t
DATUM_CSYS_Stick_SJ8_2ND =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_Stick_SJ8_2ND_data,DATUM_CSYS_St
ick_SJ8);

       DATUM_CSYS_DATA CSYS_Stick_SJ2_CONS_data;

CSYS_Stick_SJ2_CONS_data.offset_x = Link_Dimensions_obj.Stick_tail_len;
       CSYS_Stick_SJ2_CONS_data.offset_y = -Stick_data_Obj.Y_middle;
       CSYS_Stick_SJ2_CONS_data.offset_z = Stick_data_Obj.base_min;
       CSYS_Stick_SJ2_CONS_data.angle_x = 0.0;
       CSYS_Stick_SJ2_CONS_data.angle_y = 0.0;
       CSYS_Stick_SJ2_CONS_data.angle_z = 0.;
```

```
	CSYS_Stick_SJ2_CONS_data.transform_sequence = true;
	CSYS_Stick_SJ2_CONS_data.rotation_sequence[0] = 1;
	CSYS_Stick_SJ2_CONS_data.rotation_sequence[1] = 2;

tag_t DATUM_CSYS_Stick_SJ2_CONS =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_Stick_SJ2_CONS_data,DATUM_CSYS_S
tick_SJ9);

	DATUM_CSYS_DATA CSYS_Stick_SJ2_CONS_OUT_data;

	CSYS_Stick_SJ2_CONS_OUT_data.offset_x = 0.;
	CSYS_Stick_SJ2_CONS_OUT_data.offset_y = 0.;
	CSYS_Stick_SJ2_CONS_OUT_data.offset_z = 12.;
	CSYS_Stick_SJ2_CONS_OUT_data.angle_x = 0.0;
	CSYS_Stick_SJ2_CONS_OUT_data.angle_y = 0.0;
	CSYS_Stick_SJ2_CONS_OUT_data.angle_z = 0.;
	CSYS_Stick_SJ2_CONS_OUT_data.transform_sequence = true;
	CSYS_Stick_SJ2_CONS_OUT_data.rotation_sequence[0] = 1;
	CSYS_Stick_SJ2_CONS_OUT_data.rotation_sequence[1] = 2;

tag_t DATUM_CSYS_Stick_SJ2_CONS_OUT =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_Stick_SJ2_CONS_OUT_data,
                                 DATUM_CSYS_Stick_SJ2_CONS);


	DATUM_CSYS_DATA CSYS_Stick_MIDDLE_data;

CSYS_Stick_MIDDLE_data.offset_x =
0.25*Link_Dimensions_obj.Stick_tail_len;
	CSYS_Stick_MIDDLE_data.offset_y = 0.0;
	CSYS_Stick_MIDDLE_data.offset_z = 0.0;
	CSYS_Stick_MIDDLE_data.angle_x = 0.0;
	CSYS_Stick_MIDDLE_data.angle_y = 0.0;
	CSYS_Stick_MIDDLE_data.angle_z =0.0;
	CSYS_Stick_MIDDLE_data.transform_sequence = true;
	CSYS_Stick_MIDDLE_data.rotation_sequence[0] = 1;
	CSYS_Stick_MIDDLE_data.rotation_sequence[1] = 2;

tag_t DATUM_CSYS_Stick_MIDDLE =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_Stick_MIDDLE_data,DATUM_CSYS_Sti
ck_SJ9);

	DATUM_CSYS_DATA CSYS_Stick_HINGE_data;

	CSYS_Stick_HINGE_data.offset_x = 0.0;
	CSYS_Stick_HINGE_data.offset_y = 0.0;
	CSYS_Stick_HINGE_data.offset_z = 25.0;
	CSYS_Stick_HINGE_data.angle_x = 0.0;
	CSYS_Stick_HINGE_data.angle_y = 0.0;
	CSYS_Stick_HINGE_data.angle_z =0.0;
	CSYS_Stick_HINGE_data.transform_sequence = true;
	CSYS_Stick_HINGE_data.rotation_sequence[0] = 1;
	CSYS_Stick_HINGE_data.rotation_sequence[1] = 2;

tag_t DATUM_CSYS_Stick_HINGE =
wub_Create_DATUM_CSYS_Offset_Method(CSYS_Stick_HINGE_data,DATUM_CSYS_Stic
k_SJ9);

/*
	++++++++++++++++++++++++++++ PLANES  ++++++++++++++++++++++++++++++++
*/
```

```
	tag_t
		PLANE_Horizontal_middle,
		PLANE_Vertical_middle,
		PLANE_Vertical_JS8_SND,
		PLANE_HINGE_xy,
		PLANE_MIDDLE_yz;

PLANE_Horizontal_middle =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_Stick_SJ9, zx_plane);
PLANE_Vertical_middle =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_Stick_SJ9, xy_plane);
PLANE_HINGE_xy = Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_Stick_HINGE,
xy_plane);
PLANE_MIDDLE_yz =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_Stick_MIDDLE, yz_plane);
PLANE_Vertical_JS8_SND =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_Stick_SJ8_2ND, yz_plane);

/*
	++++++++++++++++++++++++++++++ POINTS ++++++++++++++++++++++++++++++
*/

	double
		point_stick_first[3],
		point_stick_middle[3],
		point_stick_last[3];

	point_stick_middle[0] = Link_Dimensions_obj.Stick_tail_len;
	point_stick_middle[1] = Stick_data_Obj.Y_middle;
	point_stick_middle[2] = 0.0;

	tag_t
		POINT_Stick_fist,
		POINT_Stick_middle,
		POINT_Stick_last;

	UF_CURVE_create_point(point_stick_middle,&POINT_Stick_middle);

	/*
	++++++++++++++++++++++++++ ARCS ++++++++++++++++++++++++++++++++++++
	*/

	double
		Rad_J2S_out = 40.0,
		Rad_JS9_out = Stick_data_Obj.Y_first,
		Rad_JS3_out = Stick_data_Obj.Y_last,
		Center_J2S[3] = {0.,0.,0.},
		Center_J3S[3] = {0.,0.,0.},
		Center_J9S[3] = {0.,0.,0.};

	tag_t
		ARC_JS2_out,
		ARC_JS3_out,
		ARC_JS9_out;

	ARC_JS2_out =
wub_ARC_Center_Radius(DATUM_CSYS_Stick_SJ2,xy_plane,Rad_J2S_out,
Center_J2S,180., 300.);
	ARC_JS3_out =
wub_ARC_Center_Radius(DATUM_CSYS_Stick_SJ3,xy_plane,Rad_JS3_out,
Center_J3S,-115., 115.);
	ARC_JS9_out =
wub_ARC_Center_Radius(DATUM_CSYS_Stick_SJ9,xy_plane,Rad_JS9_out,
Center_J9S,85., 300.);
```

```
/*
	++++++++++++++++++++++ ADDITIONAL POINTS +++++++++++++++++++++++++
*/

	double
		pt_J2_RIGHT[3],
		pt_J9_RIGHT[3] = {0.,0.,0.};
	tag_t
		POINT_J2_RIGHT,
		POINT_J9_RIGHT;

	pt_J2_RIGHT[0] = 3*Rad_J2S_out;
	pt_J2_RIGHT[1] = 0;
	pt_J2_RIGHT[2] = 0;

	wub_set_wcs(DATUM_CSYS_Stick_SJ2);
UF_CSYS_map_point(UF_CSYS_ROOT_WCS_COORDS,
			pt_J2_RIGHT,
			UF_CSYS_ROOT_COORDS,
			pt_J2_RIGHT);
UF_CURVE_create_point(pt_J2_RIGHT,&POINT_J2_RIGHT);

wub_set_wcs(DATUM_CSYS_Stick_MIDDLE);
UF_CSYS_map_point(UF_CSYS_ROOT_WCS_COORDS,
			pt_J9_RIGHT,
			UF_CSYS_ROOT_COORDS,
			pt_J9_RIGHT);
UF_CURVE_create_point(pt_J9_RIGHT,&POINT_J9_RIGHT);

/*
	+++++++++++++++++++++++++++ LINES  +++++++++++++++++++++++++++++++++
*/

	tag_t
		LINE_stick_top_left,
		LINE_stick_top_right,
		LINE_stick_top_right_2ND,
		LINE_stick_down_left,
		LINE_stick_down_right,
		LINE_001;

LINE_stick_top_left = wub_Lines_Point_Tangent(POINT_Stick_middle,
ARC_JS9_out,DATUM_CSYS_Stick_SJ9, xy_plane, 2);
LINE_stick_top_right = wub_Lines_Point_Tangent(POINT_Stick_middle,
ARC_JS3_out,DATUM_CSYS_Stick_SJ9, xy_plane, 2);
LINE_stick_top_right_2ND = wub_Lines_Point_Tangent(POINT_Stick_middle,
ARC_JS3_out,DATUM_CSYS_Stick_SJ9, xy_plane, 2);
LINE_stick_down_left =
wub_Mirror_a_Curve_through_a_Plane(LINE_stick_top_left,
PLANE_Horizontal_middle);
LINE_stick_down_right = wub_Lines_Point_Tangent(POINT_J2_RIGHT,
ARC_JS3_out,DATUM_CSYS_Stick_SJ9, xy_plane, 4);
LINE_001 = wub_Lines_Point_Tangent(POINT_J9_RIGHT,
ARC_JS2_out,DATUM_CSYS_Stick_SJ2, xy_plane,4);

	// this line has to be projected before modified
tag_t
	PROJECTED_LINE_stick_top_left;
PROJECTED_LINE_stick_top_left =
wub_Create_Projected_Curve(LINE_stick_top_left, DATUM_CSYS_Stick_HINGE,
xy_plane);

	tag_t
```

```
              LINE_stick_top_left_trmd,
              LINE_stick_down_left_trmd,
              LINE_stick_top_right_2ND_trmd,

              ARC_JS2_out_trmd,
              ARC_JS3_out_trmd,
              ARC_JS9_out_trmd;

LINE_stick_top_left_trmd =
wub_Trim_Curve_by_Datum_Plane(LINE_stick_top_left, PLANE_MIDDLE_yz,2);
LINE_stick_down_left_trmd =
wub_Trim_Curve_by_Curve(LINE_stick_down_left,LINE_001,1);
ARC_JS2_out_trmd = wub_Trim_Curve_by_Curve(ARC_JS2_out,LINE_001,1);
ARC_JS3_out_trmd =
wub_Trim_Curve_by_Curve(ARC_JS3_out,LINE_stick_top_right,0);
ARC_JS3_out_trmd =
wub_Trim_Curve_by_Curve(ARC_JS3_out_trmd,LINE_stick_down_right,1);
ARC_JS9_out_trmd =
wub_Trim_Curve_by_Curve(ARC_JS9_out,LINE_stick_down_left_trmd,0);

LINE_stick_top_right_2ND_trmd =
wub_Trim_Curve_by_Datum_Plane(LINE_stick_top_right_2ND,
PLANE_Vertical_JS8_SND,2);

       UF_DISP_set_highlight(LINE_stick_top_right_2ND_trmd,1);

/*
       +++++++++++++++++++++++ BRIDGES +++++++++++++++++++++++++++++++++++
*/

       tag_t
              BRIDGE_J2_RIGHT,
              BRIDGE_J9_RIGHT;

       BRIDGE_J2_RIGHT =  wub_Bridge_Arc_with_Line(ARC_JS2_out,
                                                  LINE_stick_down_right,
                                                  0,
                                                  0,
                                                  1,
                                                  0);
       BRIDGE_J9_RIGHT = wub_Bridge_Curves( LINE_stick_top_left_trmd,
                                     LINE_001,
                                     0,0,1,0);

/*
       ++++++++++++++++++++  PROJECTION OF CURVES  ++++++++++++++++++++++
*/
       tag_t
              PROJECTED_LINE_stick_down_left_trmd,
              PROJECTED_ARC_JS9_out_trmd,
              PROJECTED_LINE_001,
              PROJECTED_BRIDGE_J9_RIGHT;

PROJECTED_LINE_stick_down_left_trmd =
wub_Create_Projected_Curve(LINE_stick_down_left_trmd,
                           DATUM_CSYS_Stick_HINGE,
                           xy_plane);
PROJECTED_ARC_JS9_out_trmd =
wub_Create_Projected_Curve(ARC_JS9_out_trmd,
                           DATUM_CSYS_Stick_HINGE,
                           xy_plane);
PROJECTED_LINE_001 =
wub_Create_Projected_Curve(LINE_001,
                           DATUM_CSYS_Stick_HINGE,
                          xy_plane);
PROJECTED_BRIDGE_J9_RIGHT =
wub_Create_Projected_Curve(BRIDGE_J9_RIGHT,
                          DATUM_CSYS_Stick_HINGE,
                          xy_plane);
/*
      ------------------ MODIFY PROJECTED CURVES --------------------
*/
      tag_t
            PROJECTED_LINE_stick_top_left_trmd,
            PROJECTED_LINE_001_trmd;

      PROJECTED_ARC_JS9_out_trmd =
wub_Trim_Curve_by_Curve(PROJECTED_ARC_JS9_out_trmd,
                  PROJECTED_LINE_stick_top_left,
                  1);
      PROJECTED_LINE_stick_top_left_trmd =
wub_Trim_Curve_by_Curve(PROJECTED_LINE_stick_top_left,
                  PROJECTED_BRIDGE_J9_RIGHT,
                  1);
      PROJECTED_LINE_001_trmd =
wub_Trim_Curve_by_Curve(PROJECTED_LINE_001,
                  PROJECTED_LINE_stick_down_left_trmd,
                  0);
/*
      ++++++++++++++++++++++  JOINING CURVES  +++++++++++++++++++++++++
*/
      //     HINGE PROFILE (5 ITEMS)
PROJECTED_ARC_JS9_out_trmd =
wub_EXTRACT_Object_out_of_Feature(PROJECTED_ARC_JS9_out_trmd);
PROJECTED_LINE_stick_top_left_trmd =
wub_EXTRACT_Object_out_of_Feature(PROJECTED_LINE_stick_top_left_trmd);
PROJECTED_BRIDGE_J9_RIGHT =
wub_EXTRACT_Object_out_of_Feature(PROJECTED_BRIDGE_J9_RIGHT);
PROJECTED_LINE_001_trmd =
wub_EXTRACT_Object_out_of_Feature(PROJECTED_LINE_001_trmd);
PROJECTED_LINE_stick_down_left_trmd =
wub_EXTRACT_Object_out_of_Feature(PROJECTED_LINE_stick_down_left_trmd);

      //     STICK PROFILE (8 ITEMS)
BRIDGE_J9_RIGHT = wub_EXTRACT_Object_out_of_Feature(BRIDGE_J9_RIGHT);
LINE_stick_top_left_trmd =
wub_EXTRACT_Object_out_of_Feature(LINE_stick_top_left_trmd);
LINE_stick_top_right =
wub_EXTRACT_Object_out_of_Feature(LINE_stick_top_right);
ARC_JS3_out_trmd = wub_EXTRACT_Object_out_of_Feature(ARC_JS3_out_trmd);
LINE_stick_down_right =
wub_EXTRACT_Object_out_of_Feature(LINE_stick_down_right);
BRIDGE_J2_RIGHT = wub_EXTRACT_Object_out_of_Feature(BRIDGE_J2_RIGHT);
ARC_JS2_out_trmd = wub_EXTRACT_Object_out_of_Feature(ARC_JS2_out_trmd);
      LINE_001 = wub_EXTRACT_Object_out_of_Feature(LINE_001);

      tag_t
            joined_data_Hinge[5] = {PROJECTED_ARC_JS9_out_trmd,

      PROJECTED_LINE_stick_top_left_trmd,
                          PROJECTED_BRIDGE_J9_RIGHT,
                          PROJECTED_LINE_001_trmd,
                          PROJECTED_LINE_stick_down_left_trmd};
      tag_t
            joined_data_stick_profile[7] = {LINE_stick_top_right,
                                      LINE_stick_top_left_trmd,
                                      BRIDGE_J9_RIGHT,
                                      LINE_001,
                                      ARC_JS2_out_trmd,
                                      BRIDGE_J2_RIGHT,
                                      LINE_stick_down_right,
                                      };
      tag_t
            JOINED_HINGE_J9,
            JOINED_STICK_PROFILE;
      JOINED_HINGE_J9 = wub_Join_Curves(joined_data_Hinge,5);
      JOINED_STICK_PROFILE =
wub_Join_Curves(joined_data_stick_profile,7);
                                    /*
  ++++++++++++++++ PROJECT PROFILES ON VERTICAL PLANES +++++++++++++++
                                    */

      tag_t
            PROJECTED_Stick_profile,
            PROJECTED_Stick_wall_profile;

      PROJECTED_Stick_profile =
wub_Create_Projected_Curve(JOINED_STICK_PROFILE,
DATUM_CSYS_Stick_SJ2_CONS_OUT, xy_plane);
      PROJECTED_Stick_wall_profile =
wub_Create_Projected_Curve(JOINED_STICK_PROFILE,
DATUM_CSYS_Stick_SJ2_CONS, xy_plane);

/*
++++++++++++++++++++++++  EXTRUDE CURVES +++++++++++++++++++++++++++++
*/

      char* limit_J9S[2] = {"0.","25."};
      tag_t EXTRUDE_J9S = wub_Extrude(JOINED_HINGE_J9, limit_J9S);

/*
++++++++++++++++++++++ SHEET BODIES CONSTRUCTIONS ++++++++++++++++++
*/

tag_t POINT_SECTIONAL  = wub_Point_from_Spline(LINE_stick_top_right, 1);
tag_t POINT_SECTIONAL_prjctd =
wub_Create_Projected_Curve(POINT_SECTIONAL,
DATUM_CSYS_Stick_SJ2_CONS_OUT, xy_plane);
tag_t ARC_JS3_out_trmd_prjcted_out =
wub_Create_Projected_Curve(ARC_JS3_out_trmd,
DATUM_CSYS_Stick_SJ2_CONS_OUT, xy_plane);
tag_t ARC_JS3_out_trmd_prjcted =
wub_Create_Projected_Curve(ARC_JS3_out_trmd, DATUM_CSYS_Stick_SJ2_CONS,
xy_plane);
      tag_t LINE_SECTIONAL =
wub_Lines_from_two_points(POINT_SECTIONAL, POINT_SECTIONAL_prjctd);
      tag_t SWEEP_top =
wub_SWEEP_2_guides(PROJECTED_Stick_profile,
                  JOINED_STICK_PROFILE,
                  LINE_SECTIONAL);

tag_t SWEEP_J3S = wub_SWEEP_2_guides(ARC_JS3_out_trmd_prjcted_out,
                                  ARC_JS3_out_trmd,
                                  LINE_SECTIONAL);

tag_t  Curve_String[2] =
{ARC_JS3_out_trmd_prjcted,PROJECTED_Stick_wall_profile};
      tag_t
            BPLANE_WALL = wub_BPLANE(Curve_String);
/*
      +++++++++++++++++++++++++ THICKEN SURFACES +++++++++++++++++++++++++
*/
        tag_t
                THICKEN_top,
                THICKEN_side,
                THICKEN_J3;
        tag_t
                UNITED_solid;

        UF_MODL_create_thicken_sheet(SWEEP_top,
                                "-1.",
                                "6",
                                UF_NULLSIGN,
                                &THICKEN_top);
        UF_MODL_create_thicken_sheet(SWEEP_J3S,
                                "1.",
                                "-6",
                                UF_NULLSIGN,
                                &THICKEN_J3);
        UF_MODL_create_thicken_sheet(BPLANE_WALL,
                                "0.",
                                "-7.0",
                                UF_NULLSIGN,
                                &THICKEN_side);

/*
        +++++++++++++++++     JOINT STRUCTURES    ++++++++++++++++++++++
*/

        tag_t
                SOLID_THICKEN_top,
                SOLID_THICKEN_side,
                SOLID_THICKEN_J3,
                UNITED_Stick_solid;

        UF_CALL(UF_MODL_ask_feat_body(THICKEN_top, &SOLID_THICKEN_top));
        UF_CALL(UF_MODL_ask_feat_body(THICKEN_side, &SOLID_THICKEN_side));
        UF_CALL(UF_MODL_ask_feat_body(THICKEN_J3, &SOLID_THICKEN_J3));

        UNITED_Stick_solid = wub_UNITE_SOLIDS(SOLID_THICKEN_top,
                                        SOLID_THICKEN_side);
        UNITED_Stick_solid = wub_UNITE_SOLIDS(UNITED_Stick_solid,
                                        SOLID_THICKEN_J3);
        UNITED_Stick_solid =
wub_UNITE_SOLIDS(UNITED_Stick_solid,EXTRUDE_J9S);

        // JOINT 3 HINGE STRUCTRE

        wub_set_wcs(DATUM_CSYS_Stick_SJ3);
                        double
                                origin_bool_SJ3[3] = {0.,0.,0.},
                                dirction_bool_SJ3[3] = {0.,0.,1};
                        tag_t
                                bool_SJ3_cylinder_outside,
                                bool_SJ3_cylinder_inside;
        UF_CSYS_map_point(UF_CSYS_ROOT_WCS_COORDS,
                        origin_bool_SJ3,
                        UF_CSYS_ROOT_COORDS,
                        origin_bool_SJ3);
        UF_MODL_create_cylinder(UF_POSITIVE,
                                        UNITED_Stick_solid,
                                                origin_bool_SJ3,
                                "100.",
                                "40.",
                              dirction_bool_SJ3,
                              &bool_SJ3_cylinder_outside);
       UNITED_Stick_solid =
wub_UNITE_SOLIDS(UNITED_Stick_solid,bool_SJ3_cylinder_outside);
       UF_MODL_create_cylinder(UF_NEGATIVE,
                              UNITED_Stick_solid,
                              origin_bool_SJ3,
                              "110.",
                              "25.",
                              dirction_bool_SJ3,
                              &bool_SJ3_cylinder_inside);

       // JOINT 9 HINGE STRUCTURE

       wub_set_wcs(DATUM_CSYS_Stick_SJ9);
                       double
                              origin_bool_SJ9[3] = {0.,0.,0.},
                              dirction_bool_SJ9[3] = {0.,0.,1};
                       tag_t
                              bool_SJ9_cylinder_outside,
                              bool_SJ9_cylinder_inside;
       UF_CSYS_map_point(UF_CSYS_ROOT_WCS_COORDS,
                       origin_bool_SJ9,
                       UF_CSYS_ROOT_COORDS,
                       origin_bool_SJ9);
       UF_MODL_create_cylinder(UF_NEGATIVE,
                              UNITED_Stick_solid,
                              origin_bool_SJ9,
                              "100.",
                              "25.",
                              dirction_bool_SJ9,
                              &bool_SJ9_cylinder_outside);

       // JOINT 6 (FOUR-BAR MECHANISM CONNECTION POINT) STRUCTURE

       wub_set_wcs(DATUM_CSYS_Stick_SJ6);
                       double
                              origin_bool_SJ6[3] = {0.,0.,0.},
                              dirction_bool_SJ6[3] = {0.,0.,1};
                       tag_t
                              bool_SJ6_cylinder_outside,
                              bool_SJ6_cylinder_inside;
       UF_CSYS_map_point(UF_CSYS_ROOT_WCS_COORDS,
                       origin_bool_SJ6,
                       UF_CSYS_ROOT_COORDS,
                       origin_bool_SJ6);
       UF_MODL_create_cylinder(UF_POSITIVE,
                              UNITED_Stick_solid,
                              origin_bool_SJ6,
                              "100.",
                              "40.",
                              dirction_bool_SJ6,
                              &bool_SJ6_cylinder_outside);
       UNITED_Stick_solid = wub_UNITE_SOLIDS(UNITED_Stick_solid,
                                            bool_SJ6_cylinder_outside)
                                            ;
       UF_MODL_create_cylinder(UF_NEGATIVE,
                              UNITED_Stick_solid,
                              origin_bool_SJ6,
                              "110.",
                              "25.",
                              dirction_bool_SJ6,
                              &bool_SJ6_cylinder_inside);
```

```
        // JOINT 2 STRUCTRE

        wub_set_wcs(DATUM_CSYS_Stick_SJ2);
                            double
                                  origin_bool_SJ2[3] = {0.,0.,0.},
                                  dirction_bool_SJ2[3] = {0.,0.,1};
                            tag_t
                                  bool_SJ2_cylinder_outside,
                                  bool_SJ2_cylinder_inside;
        UF_CSYS_map_point(UF_CSYS_ROOT_WCS_COORDS ,
                      origin_bool_SJ2,
                      UF_CSYS_ROOT_COORDS,
                      origin_bool_SJ2);
        UF_MODL_create_cylinder(UF_POSITIVE,
                                  UNITED_Stick_solid,
                                  origin_bool_SJ2,
                                  "100.",
                                  "40.",
                                  dirction_bool_SJ2,
                                  &bool_SJ2_cylinder_outside);
        UNITED_Stick_solid = wub_UNITE_SOLIDS(UNITED_Stick_solid,
                                          bool_SJ2_cylinder_outside)
                                          ;
        UF_MODL_create_cylinder(UF_NEGATIVE,
                                  UNITED_Stick_solid,
                                  origin_bool_SJ2,
                                  "110.",
                                  "25.",
                                  dirction_bool_SJ2,
                                  &bool_SJ2_cylinder_inside);

        /// MIRROR STICK BODY

        tag_t
              mirror_plane,
              mirrored_body,
              final_sTICK_solid;
        mirror_plane =
Extract_dplane_tag_of_Datum_CSYS(DATUM_CSYS_Stick_SJ9,
                                                      xy_plane);
        UF_MODL_create_mirror_body(UNITED_Stick_solid,
                                  mirror_plane,
                                  &mirrored_body);

    UF_terminate();
        }
int ufusr_ask_unload(void)
        {
        return(UF_UNLOAD_IMMEDIATELY);
        }
```