Feature-Based Mechanism Design

Abiy Wubneh and Y.-S. Ma

1 Introduction

This chapter presents a feature-based concept design method for variational mechanisms. The proposed method integrates dimensional synthesis, mechanical design, and CAD generation with advanced feature technology with minimal designer intervention. Extended feature definitions are used in this research to create a smooth data flow connection between different engineering tools and applications. An excavator arm mechanism is used as a case study to demonstrate the proposed design method. The design of the given mechanism is carried out based on its digging mode configurations as introduced in the previous chapter.

The design process of multi-component products which are subject to frequent changes and modification is very complex due to the large amount of data involved. Dimensions and parameters defined by customers at the initial stages of the design process are used by latter design stages during a product lifecycle. This situation is further complicated when different parts of the product are designed by people located in various geographic locations. Changes and modifications evoked by one department are not reflected on components being developed by other departments without considerable time and resources. Constraint definition and management is just one area affected by the chosen data management system adopted in the design process.

Traditional CAD systems lack effective means for the implementation of effective knowledge-driven design procedures, especially for embedding engineering semantic patterns and the subsequent evolvement in CAD-supported design processes. This has forced researchers to look into possible ways of enriching information in the traditional CAD models.

A. Wubneh · Y.-S. Ma (✉)
Department of Mechanical Engineering, University of Alberta,
Edmonton, AB T6G 2G8, Canada
e-mail: yongsheng.ma@ualberta.ca

Y.-S. Ma (ed.), *Semantic Modeling and Interoperability in Product and Process Engineering*, Springer Series in Advanced Manufacturing,
DOI: 10.1007/978-1-4471-5073-2_10,

Features were introduced as a means to address the engineering semantic pattern modeling need. Features are basically object data structures containing a range of information and service functions required to fully describe and manipulate the shape and related semantic aspects of the engineering pattern. They have been used to deal with the most commonly used data elements including model geometries, materials, manufacturing methods, tolerances, and machining procedures. Recently, more complicated and sophisticated features have been defined to cover previously overlooked but critical aspects of the design process.

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

2 Statement of 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 different phases of the product development cycles. This limitation directly affects the choices of the design methodologies and the necessary data communication mechanisms. The information required by different stages of product design process has to be transferred in a effective manner to achieve maximum automation and efficiency. 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 models. Thus, the previously fragmented design data files and those corresponding product CAD models have to be combined 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 used to integrate different engineering data structures with strong associativity to engineering thinking patterns and semantics.

The conceptual design process of variational mechanism, which is the focus of this research, is known for its various fragmented modules running on some sets of common 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 consider the new properties (both physical and geometric) before commencing the next cycle. Effective automation of this process requires the systematic creation of CAD models in a very consistent manner.

Associative relationships between different features of a mechanism design model have to be systematically constrained to ensure the final generated model has comprehensive physical and engineering descriptions. A design procedure

equipped with methods which can satisfy these requirements will earn itself an important place in advancing the industrial application.

3 Objectives

The objective of this research is to propose a feature-based conceptual design automation scheme specific to variational mechanism products. This scheme 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 current automation gap in the conceptual design cycle and to further investigate features' applicability in terms of embedding conceptual design rules for complex part shapes and structures development.

The applicability of the proposed methods will be demonstrated using the design procedure of an excavator mechanism as a case study. This chapter addresses feature-based product design aspects of product configuration, linkage optimization, and their programming implementation.

4 Scope of 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. The case study mentioned includes the basic generation and optimization design calculations of an excavator arm mechanism. Only the digging operational conditions are considered for the design purpose. The design has been carried out mainly taking the strength requirements (working stresses) into consideration in a numerical approach (design for strength). Other design aspects, such as dynamic behavior and finite element analysis (FEA), are beyond the scope of this work due to the resource constraints.

5 Literature Review

This chapter reviews research works and publications of other scholars which are relevant to the objective of this research. The overall organization of this section is targeted to cover the following major topics:

- Design automation and integration
- Parametric and feature-based CAD modeling
- Reverse engineering and knowledge fusion in product development.

5.1 Parametric and Feature-Based CAD Modeling

Several methods and procedures have been developed to automate and increase the efficiency of CAD modeling processes. The depth of data embedded in the CAD modes greatly depends on the techniques employed to carry out the process [22]. Parametric modeling, among others, has become one of the most rapidly growing and commonly adopted methods of product modeling in the manufacturing industries. Much research effort has been focused on modifying the *Standardized Exchange of Product* (STEP) format, which contains only geometric information, to accommodate for additional part-specific parameterized information [15].

This approach 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 known limitations in terms of validity in change and modification implementations. Basak and Gulesin's study [1] 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. This method also used standard feature libraries to investigate the practicality of proposed part manufacturing techniques by the designers.

Programming through the application programming interfaces (APIs) of existing commercial CAD packages provides designers with more flexibility to embed the design intent into the CAD models [9]. In their approach, Myung and Han [13] took design unit and functionality features into consideration for the configuration and CAD modeling of selected products. The work of Wang et al. [25] 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 defined parent–child relationships have to be consistently validated in order to apply this method in the design process of customizable products. The use of explicit relationship was suggested by Van et al. [22] to increase user control and add sophistication to the modeling process.

Although parametric modeling techniques are widely used in today's design and manufacturing industries for facilitating CAD modeling processes, their use has been limited to 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 additional information to the CAD models. The type of information ranges from purely geometric to non-geometric parameters. Traditional features represented only those attributes related to the geometry of the part. Recently, new types of feature definitions [20] 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 significantly the implementation of an integrated and automated product design procedure [26]. The interest of manufacturers in reducing the time to market and costs associated with the design process has motivated researchers [27] to investigate features in greater depth. The power of feature-based modeling methods was coupled with the concepts of reverse engineering techniques [27] to embed design intents and other constraints into existing products retrieved by CAD scanning techniques (reverse engineering). By employing this method, manufacturers will reduce the time required to re-fabricate a given existing product with different materials 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 [26]. Ter et al. [20] 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 a feature, which used to be merely a description of the shapes and geometries of the CAD models, has been extended to cover assembly design features and other vital aspects in regard to manufacturing and concurrent engineering [3]. Associative relationships, both geometric and non-geometric, between various parameters of two or more members of an assembly were discussed by Ma et al. [9]. This ability opens the door for design automation of frequently updated and modified products. Design customization of products can benefit from the inclusion of design intents, constraints, and assembly hierarchy data [2] 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 were not previously considered can be defined. Additionally, the feature definition is extended to cover information pertinent to component mating conditions and interfaces within an assembly.

5.2 Design Automation and Integration

The implementation of a collaborative product development process requires a large amount of data to be transferred between applications used by different designers working toward a single product [14]. Change and modifications are part of a two-way process in this approach. Ma and his colleagues defined a data structure (feature) called *operation* in an effort to address the need to communicate data at the feature level [11]. An associative fine-grain product repository mechanism with a four-layer information scheme was proposed and demonstrated by

the team for this purpose. The method was proposed with consideration for the probability of using different applications and platforms due to the interdisciplinary nature of the product design process.

Features, which have a higher level of semantic organization than the simple geometric aspects of a product, are currently being used to create the link and bridge the gap in terms of the amount and detail of information needed to be shared by CAD and CAM systems [12, 18]. Concurrent and collaborative engineering 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. [10] proposed a unified feature approach method for constraint management and change propagation to be used in a collaborative and concurrent environment. The algorithm developed uses the JTMS-based dependency network. The data model was categorized under constraint-based associativity and shares entity references. Lourenco et al. [8], 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 perspectives. Solazzi discusses [19] the advantages and quantitative analysis of performance improvements achieved by redesigning an existing hydraulic arm mechanism with a different material. Yoon and Manurung [28] investigated the development of a control system by which the operations of an excavator arm mechanism are controlled by the operator's own arm movements. Sensors attached at different joint locations of an operator's arm are used to map the arm joint displacements in the mechanism's motion.

The development of new design automation procedures [16, 21] together with existing mechanical simulation tools such as SimMechanics of MATLAB® and MSc ADAMS® have given researchers the opportunity to fully impose explicit constraints when investigating mechanisms and manipulators' kinematic and dynamic responses [23]. The forward and inverse kinematics analyses involved in the design of mechanisms and manipulators benefit from the implementation of parametric and feature-based modeling approaches [23]. Work space configurations of manipulators and their dynamic responses require frequent changes and fine-tuning initial parameters which can be easily implemented with the use of appropriate feature definitions.

5.3 Reverse Engineering and Knowledge Fusion

Reverse engineering (RE) techniques had been used to extract shapes and geometries from existing products [4, 5]. The results of RE procedures usually poorly represent the design logic used to create the parts. The gap between 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. [4, 5].

The traditional RE tools allow creating the geometries of existing products but lack the capability of embedding the design intents. The method proposed in Durupt's 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 include design intents in the RE scan results [7, 24].

Topological approaches have recently become more popular for their ability to generate 3D free shape models based on finite element concepts. However, like that of the RE techniques, there is much research to be done before it will be possible to smoothly extract simple CAD models from these shapes. The work of Larsen and Jensen [6] focuses on the investigation of methodologies to extract parametric models out of topologically optimized 3D shapes.

6 The Proposed Approach

Product modeling involves creating and combining individual basic semantic entities called features and further generating part geometries. 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 be imposed on the geometric entities defining the features. The data from the features can be easily accessed and modified, making this method ideal for 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 allows them to play an important role in the automation processes of conceptual design cycles.

In this chapter, we will propose and discuss two methods to be used in the implementation of feature-based CAD modeling techniques in the development and design of automation processes of variational mechanisms.

6.1 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 of 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 customers as well as due to newly arising engineering requirements. Without a system to address these changes efficiently, the costs associated with the modifications could undermine the product need.

This section proposes a method by which such changes and design intent modifications are handled in a very cost-effective and timely manner using a 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 inevitable changes and modifications in the design process.

6.2 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 are beyond the scope of this chapter. The authors will instead focus on the intended communication, which is achieved through the use of neutral text data files written and updated by program codes developed for this purpose.

In Fig. 1, the design process begins with input information in the form of user specifications. 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. These dimensions will then be used as the skeleton assembly model in an initial kinematic and dynamic analysis. This analysis 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 (FBDs) and mass-acceleration diagrams (MADs) which will be used during the design and optimization phases.

Results obtained in 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 abovementioned considerations in the design and optimization phases, several additional factors may be considered (depending on the type of product) such as design codes, standards, assumption, and safety factors.

Based on dimensional data determined by previous processes, the 3D models of the mechanism components will be modeled using 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.

For this case study, the programming component 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, preferably assembled, will be exported back to the *Dynamic Analysis and Simulation* block to take the effects of their newly created 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.

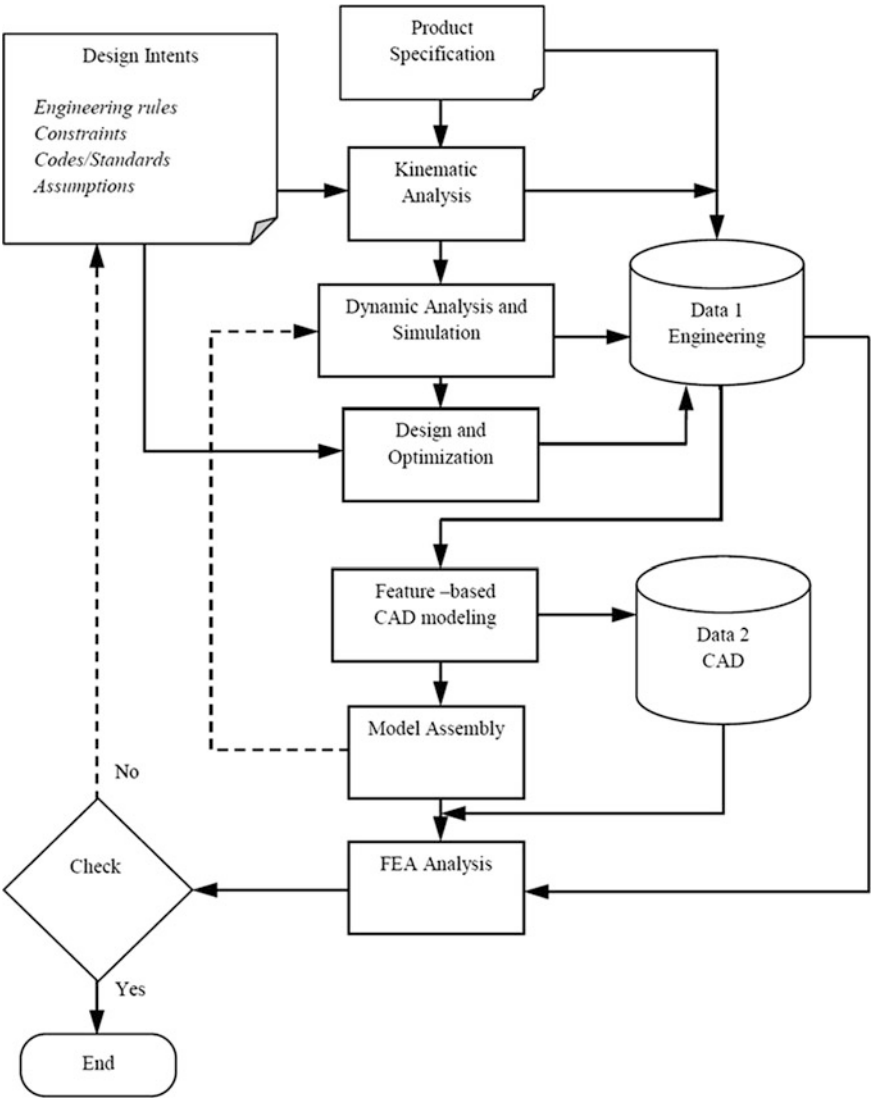


Fig. 1 Design process modules and data communication routes

7 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 individuals 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 due to its potential for future use in the areas of knowledge fusion, as well as a training source for artificial neural network applications.

Data structures, implemented by using object-oriented programming tools, address these needs. The *Product Specification* input shown in Fig. 1 needs to be organized systematically and its scope needs to be defined as "global" or "local" in order to establish 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, a data structure for a new product model called Product_Spec_2010 is created by instantiating the above definition and using the following command. (Note: Neither this particular example data structure nor its values are real; they are used here only for explanatory purposes.)

```
global Product_model_2010
Product_model_2010 = Product_Specification_c
```

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 depend on the convenience and applicability to a particular problem. Initial

values can be assigned during the definition of the data structure, or they can be updated afterward 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.

```
classdefc_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 makes it available for use 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 Fig. 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 with the API tools. However, all of these tools are expected to

operate based on a common set of data models and parameters produced during the initial phase of the conceptual design cycle. Accordingly, *Data 1* and *Data 2* in Fig. 1 are communicated by neutral intermediate text data files. Similar data structures need 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 an exact copy of the previous one as long as the necessary parameters are imported. Defining all corresponding data structures consistently avoids confusion and facilitates better data management.

The concept of feature has been investigated in great depth in recent decades to address emerging product development and manufacturing needs and challenges. Originally, the term feature was used to refer only to the geometrical aspects of a model such as slots, holes, and chamfers. Since the product development process involves much more than geometric entities, researchers have sought ways of embedding more information into the CAD models. Consequently, today's features have a much broader definition; both geometric and non-geometric information are embedded in the model, aiding in rapid and reliable product information transfer. Some of the many features developed in the work reported here include:

- Coordinate system (CSYS) 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.

8 Linkage Geometry Design

8.1 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—in this case, the digging operation (Fig. 2).

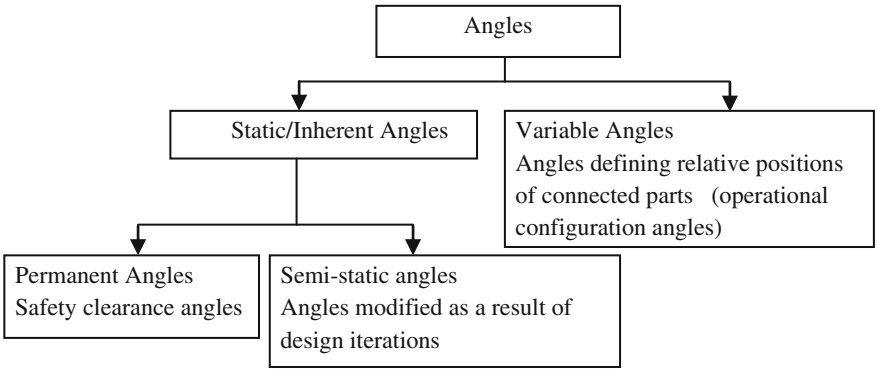


Fig. 2 Classification of angles

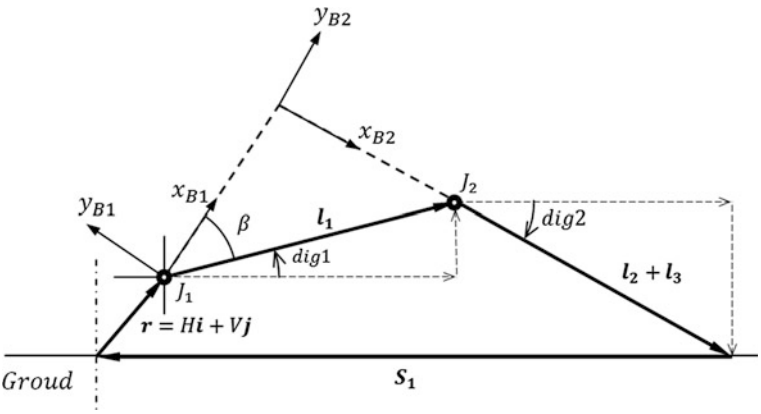


Fig. 3 Operational configuration angles

As shown in Figs. 3 and 4, angle dig1 and dig2 define the orientations of the boom and the stick with respect to the ground during a digging operation. Unlike static angles which are always assigned positive values, variable angles are direction-sensitive.

8.2 Boom Geometries

Referring to Fig. 3, expressions for variable angles dig1 and dig2 can be formulated as follows.

Summing the components of vectors along the horizontal direction gives:

$$\sum V_x = S_1 - H \tag{1}$$

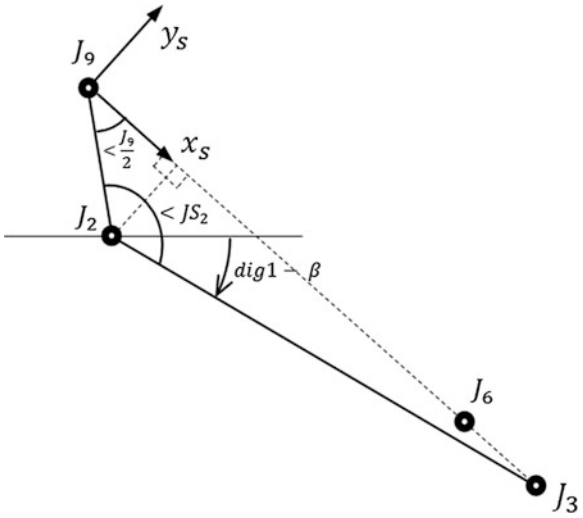


Fig. 4 Stick structural angles

$$l_1 \cos(\text{dig1}) + l_2 \cos(\text{dig2}) + l_3 \cos(\text{dig2}) = S_1 - H \tag{2}$$

$$l_1 \cos(\text{dig1}) + (l_2 + l_3) \cos(\text{dig2}) - S_1 + H = 0 \tag{3}$$

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

$$\sum V_y = 0 \tag{4}$$

$$V + l_1 \sin(\text{dig1}) + (l_2 + l_3) \sin(\text{dig2}) = 0 \tag{5}$$

Solving Eqs. (3) and (5) 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.

Boom Rotation Matrix I, RB_1

$$RB_1 = \begin{bmatrix} \cos(\text{dig1} + \beta) & -\sin(\text{dig1} + \beta) & 0 \\ \sin(\text{dig1} + \beta) & \cos(\text{dig1} + \beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Boom Rotation Matrix II, RB_2

$$RB_2 = \begin{bmatrix} \cos(\text{dig1} - \beta) & -\sin(\text{dig1} - \beta) & 0 \\ \sin(\text{dig1} - \beta) & \cos(\text{dig1} - \beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

8.3 Stick Rotation Matrix

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 $J1$.

$$\tan(J9_L) = \frac{h_{\text{stick}}}{h_{\text{tail}}} \tag{8}$$

$$J9_L = \tan^{-1}\left[\frac{h_{\text{stick}}}{h_{\text{tail}}}\right] \tag{9}$$

$$J9_U = J9L \tag{10}$$

$$J9 = J9_U + J9_L = 2 \times J9_L \tag{11}$$

$$TS_2 = \sqrt{l_2^2 - h_{\text{stick}}^2} \tag{12}$$

$$J2_l = \tan^{-1}\left[\frac{h_{\text{tail}}}{h_{\text{stick}}}\right] \tag{13}$$

$$J2_r = \tan^{-1}\left[\frac{\text{TS}_2}{h_{\text{stick}}}\right] \tag{14}$$

$$J2 = J2_l + J2_r \tag{15}$$

$$J3_l = \tan^{-1}\left[\frac{h_{\text{stick}}}{\text{TS}_2}\right] \tag{16}$$

$$J3_U = J3_L \tag{17}$$

$$J3 = J3_L + J3_U = 2 \times J3_L \tag{18}$$

$$R_{Z,(\beta+\text{dig1})} \rightarrow R_{Z,-2\beta} \rightarrow R_{Z,J2} \rightarrow R_{Z,(J9_L-180)} \tag{19}$$

$$\begin{aligned}\text{net angle of rotation} &= \beta + \text{dig1} - 2\beta + J2 + J9_L - 180 \\ &= \text{dig1} - \beta + J2 + J9_L - 180\end{aligned} \tag{20}$$

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

$$\text{RS} = \begin{bmatrix} \cos(\text{dig1} - \beta + J2 + J9_L - 180) & -\sin(\text{dig1} - \beta + J2 + J9_L - 180) & 0 \\ \sin(\text{dig1} - \beta + J2 + J9_L - 180) & \cos(\text{dig1} - \beta + J2 + J9_L - 180) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{21}$$

8.4 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 the 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 Fig. 5, 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.

In Fig. 5, γ_{ec} is the eccentricity angle necessary to prevent the mechanism from self-locking. γ_{s1} 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 in Figs. 6 and 7.

Applying Law of Cosines to the geometry of Fig. 6 we get

$$b_3^2 = b_o^2 + (b_1 + b_2)^2 - 2b_0(b_1 + b_2)\cos\gamma_{s1} \tag{22}$$

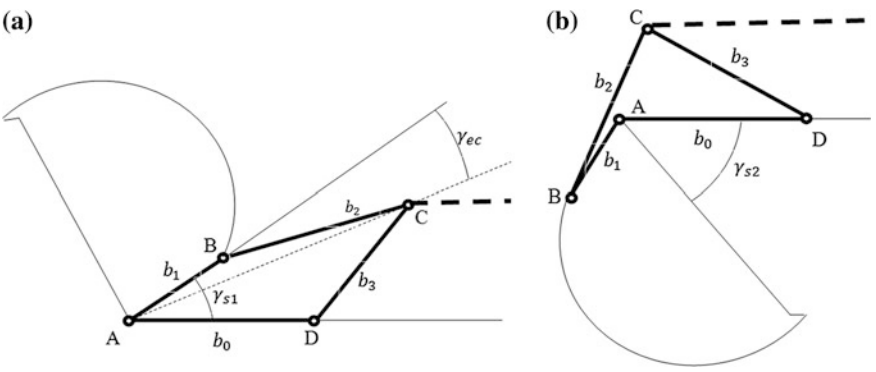


Fig. 5 Transition four-bar work ranges

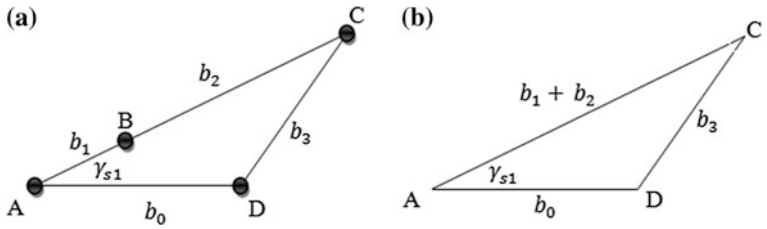


Fig. 6 Upper mechanism locking configuration

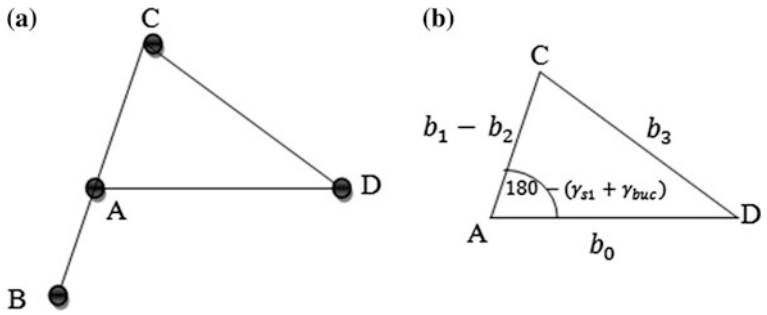


Fig. 7 Lower mechanism locking configuration

Similarly referring to Fig. 7

$$b_3^2 = b_o^2 + (b_2 - b_1)^2 + 2b_0(b_2 - b_1)\cos(\gamma_{s2} + \gamma_{\text{bk}}) \tag{23}$$

Solving Eqs. (22) and (23) 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.

9 Stress and Strength Calculations

9.1 Basic Stresses Involved

The expressions for the various stresses considered in this section are developed based on the assumptions and procedures outlined by Shigley and Mischke [17].

9.1.1 Transverse Shear Stress, τ_b

Shear stress, τ, as a result of shear force and bending moment is derived from the relation (Fig. 8)

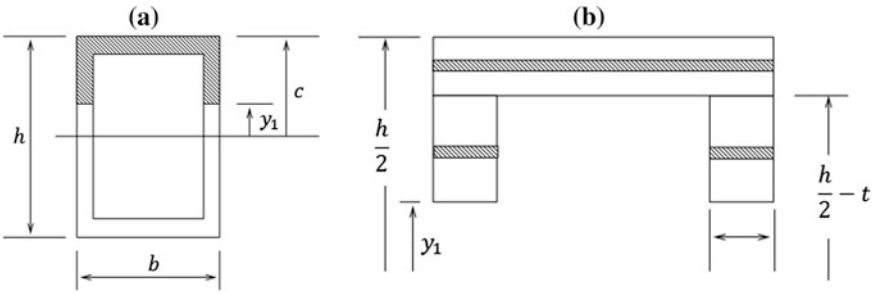


Fig. 8 Cross-sectional area under transverse shear stress

$$V = \frac{\mathrm{d}M}{\mathrm{d}x} \tag{24}$$

$$\tau = \frac{VQ}{Ib} \tag{25}$$

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

$$Q = \int_{y1}^{C} y \, \mathrm{d}A \tag{26}$$

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

Case 1

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

$$Q = \int_{y_1}^{h/2} y \, \mathrm{d}A = \int_{y_1}^{\frac{h}{2}-t} y \, \mathrm{d}A_1 + \int_{\frac{h}{2}-t}^{\frac{h}{2}} y \, \mathrm{d}A_2 \tag{27}$$

where $\mathrm{d}A_1 = 2t \, \mathrm{d}y$ and $\mathrm{d}A_2 = b \, \mathrm{d}y$

$$Q_1 = \int_{y_1}^{\frac{h}{2}-t} 2ty \, \mathrm{d}y + \int_{\left(\frac{h}{2}-t\right)}^{\frac{h}{2}} by \, \mathrm{d}y \tag{28}$$

$$Q_1 = t\left(\frac{h}{2} - t\right)^2 - ty_1^2 + \frac{bt(h-t)}{2} \tag{29}$$

Case 2

For $\left(\frac{h}{2} - t\right) \leq |y_1| \leq \frac{h}{2}$

$$Q_2 = \int_{y_1}^{h/2} y \, \mathrm{d}A_2 = \int_{y_1}^{h/2} by \, \mathrm{d}y \tag{30}$$

$$Q_2 = \frac{b}{8}\left(h^2 - 4y_1^2\right) \tag{31}$$

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

$$Q_{\max} = t\left(\frac{h}{2} - t\right)^2 + \frac{bt(h-t)}{2} \tag{32}$$

$$I = \frac{bh^3}{12} - \frac{(b-2t)(h-2t)^3}{12} \tag{33}$$

$$\tau_{\max} = \frac{VQ_{\max}}{Ib} \tag{34}$$

$$b_{\max} = 2t \tag{35}$$

$$\tau_b = \frac{V\left[t\left(\frac{h}{2} - t\right)^2 + \frac{bt(h-t)}{2}\right]}{2t\left[\frac{bh^3}{12} - \frac{(b-2t)(h-2t)^3}{12}\right]} \tag{36}$$

9.1.2 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 [17].

Area of torsion, A_{tor}, is given by the following expression (Fig. 9).

$$A_{\text{tor}} = \left(b - 2\frac{t}{2}\right)\left(h - 2\frac{t}{2}\right) \tag{37}$$

$$A_{\text{tor}} = (b-t)(h-t) \tag{38}$$

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

$$\tau_{\text{tor}} = \frac{T}{2(b-t)(h-t)t} \tag{39}$$

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

9.1.3 Direct Stress, σ_{dx}

Area of direct stress distribution is given by the expression

$$A_{\text{dx}} = 2[bt + t(h-2t)] \tag{40}$$

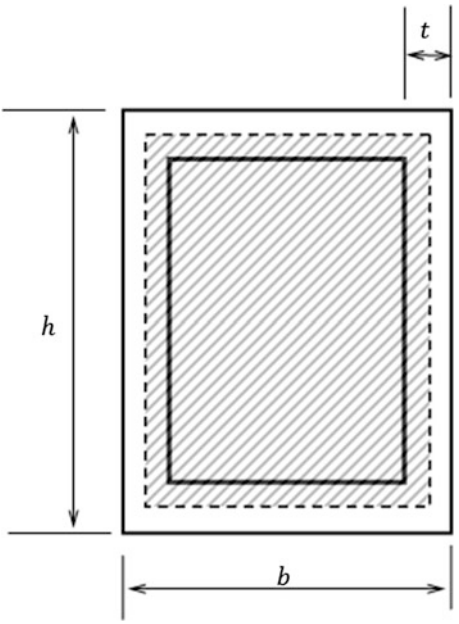


Fig. 9 Torsional cross-sectional area

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

$$\sigma_{dx} = \frac{F_{ax}}{A_{dx}} \tag{41}$$

$$\sigma_{dx} = \frac{F_{ax}}{2[bt + t(h-2t)]} \tag{42}$$

9.1.4 Bending Stresses, σ_{zx}

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

$$\sigma_{zx} = -\frac{M_z y}{I_{zz}} \left(-\frac{h}{2} \leq y \leq \frac{h}{2} \right) \tag{43}$$

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

$$I_{zz} = \frac{bh^3 - (b-2t)(h-2t)^3}{12} \tag{44}$$

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

$$y = \pm\frac{h}{2} \tag{45}$$

$$\sigma_{zx(\max)} = -\frac{6M_z h}{bh^3 - (b-2t)(h-2t)^3} \tag{46}$$

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

$$\sigma_{yx} = -\frac{M_y z}{I_{yy}}\left(-\frac{b}{2} \le z \le \frac{b}{2}\right) \tag{47}$$

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

$$I_{yy} = \frac{hb^3 - (h-2t)(b-2t)^3}{12} \tag{48}$$

$$\sigma_{yx(\max)} = -\frac{6M_z b}{hb^3 - (h-2t)(b-2t)^3} \tag{49}$$

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

$$\sigma_{xx(\max)} = \sigma_{\mathrm{dx}} + \sigma_{yx(\max)} + \sigma_{zx(\max)} \tag{50}$$

9.2 Pin Design

9.2.1 Methods of Pin Failure

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

9.2.2 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 (Fig. 10).

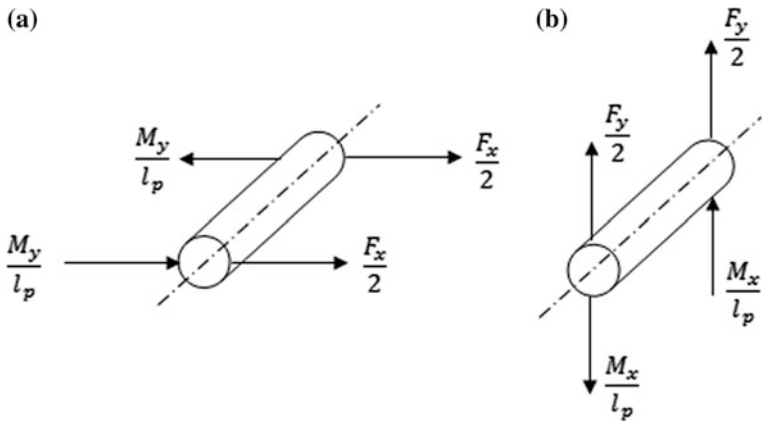


Fig. 10 Pin loads in global vertical and horizontal planes

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.

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

$$F_p = \begin{bmatrix} \frac{F_x}{2} + \frac{M_y}{l_p} \\ \frac{F_y}{2} + \frac{M_x}{l_p} \\ 0 \end{bmatrix} \tag{51}$$

where d_p and l_p are the pin diameter and its effective length, respectively.

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

$$F_p = \sqrt{\left(\frac{F_x}{2} + \frac{M_y}{l_p}\right)^2 + \left(\frac{F_y}{2} + \frac{M_x}{l_p}\right)^2} \tag{52}$$

9.2.3 Hertz's Contact Stress

The maximum 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:

$$b = k_b\sqrt{F_p l_p} \tag{53}$$

$$\text{where } k_b = \sqrt{\frac{2}{\pi t}\frac{\left(\frac{1-v_1^2}{E_1}+\frac{1-v_2^2}{E_2}\right)}{\frac{1}{\mathrm{d}_1}+\frac{1}{\mathrm{d}_2}}} \tag{54}$$

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:

$$k_b = \sqrt{\frac{2d_1\left(\frac{1-v_1^2}{E_1}+\frac{1-v_2^2}{E_2}\right)}{\pi t}} \tag{55}$$

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

$$P_{\max}(l_p, d_p) = \frac{2F_p l_p}{\pi b t} \tag{56}$$

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

$$\sigma_x = -2v_1\frac{2F_p l_p}{\pi b t}\left[\sqrt{1+\left(\frac{z}{b}\right)^2} - \left|\frac{z}{b}\right|\right] \tag{57}$$

$$\sigma_y = -\frac{2F_p l_p}{\pi b t}\left[\frac{1+2\left(\frac{z}{b}\right)^2}{\sqrt{1+\left(\frac{z}{b}\right)^2}} - 2\left|\frac{z}{b}\right|\right] \tag{58}$$

$$\sigma_z = -\frac{2F_p l_p}{\pi b t}\frac{1}{\sqrt{1+2\left(\frac{z}{b}\right)^2}} \tag{59}$$

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

$$\sigma_x = -1.944v_1\frac{F_p l_p}{\pi b t} \tag{60}$$

$$\sigma_y = -0.371\frac{F_p l_p}{\pi b t} \tag{61}$$

$$\sigma_z = -1.572\frac{F_p l_p}{\pi b t} \tag{62}$$

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

9.2.4 Failure of Pin Due to Double Shear

The total shear area of each pin is represented by the equation (Fig. 11)

$$A_{\text{sh}} = 2\frac{\pi d_p^2}{4} \tag{63}$$

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.

$$\tau_{\text{sh}} = \frac{2|F_p|}{A_{\text{sh}}} \tag{64}$$

$$\tau_{\text{sh}} = \frac{4|F_p|}{\pi d_p^2} \tag{65}$$

9.2.5 Failure of Pin Due to Bending Moment

Referring to Fig. 12 for the loading distribution, the maximum bending moment at the mid-span of the pin is given by the expression:

$$t_1 = t + e_{b1} \tag{66}$$

$$t_2 = t + e_{b2} \tag{67}$$

$$M_{\max} = F_p\left(\frac{t_1}{3} + t_2 + \frac{l_p}{2}\right) - F_p\left(\frac{t_2}{2} + \frac{l_p}{2}\right) \tag{68}$$

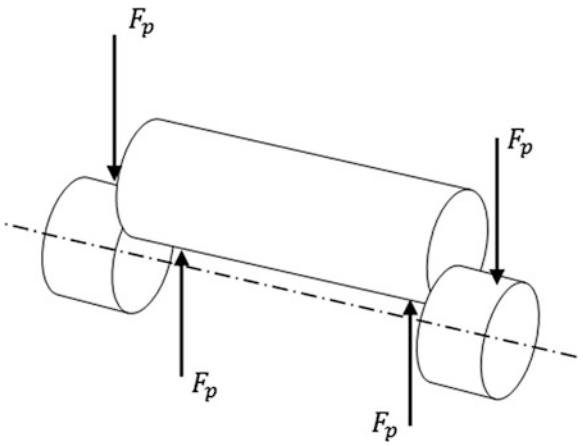


Fig. 11 Pin under double shear

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

$$\sigma_b = \frac{M_{\max}}{z} \tag{69}$$

$$\sigma_b = \frac{F_p\left[\frac{t_1}{3} + \frac{t_2}{2}\right]}{\frac{\pi}{32} d_p^3} \tag{70}$$

10 Mechanism Dynamics Analysis with Virtual Mechanism Modeling and Simulation

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.

10.1 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

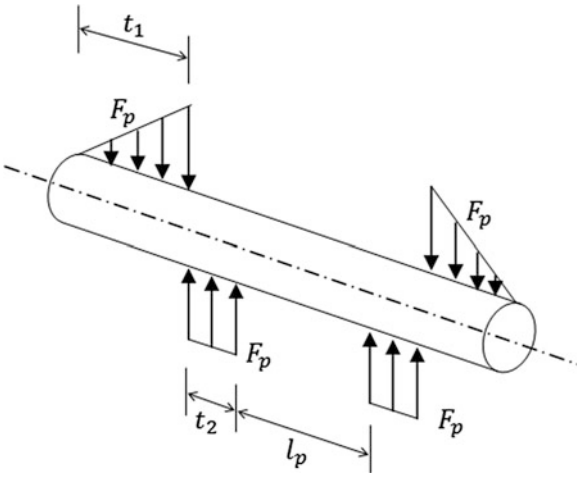


Fig. 12 Bending load distribution on pins

using the previously calculated linear dimensions in the SimMechanics modeling environment of MATLAB®. Using a skeleton mechanism 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 consider the inertia effects.

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. For this approach to be viable in the automation of design processes, two general requirements have to be met. The first is that all linear dimensions need to be defined either numerically or symbolically. The second 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 it is possible to assign inertia properties to these bodies in the initial stages as mentioned above, the procedure adopted in this case study uses a different approach, deemed more suitable for cyclic modifications. The first round of the simulations starts with weightless and high-stiffness rigid bodies; further 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 the SimMchanics® simulation environment, supports only SolidWorks® and Pro/E®, but not UG NX. To overcome this issue, SolidWorks is used as an intermediate file transfer tool in exporting the 3D models to SimMechanics.

10.2 Simulink Model Construction

Figure 13 shows the major components of an excavator arm mechanism while Fig. 14 shows the representation of the same mechanism by linear components. The latter model is constructed in SimMechanics environment using the standard rigid body and joint library.

10.3 Boom Construction

The boom as shown in Fig. 15 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.

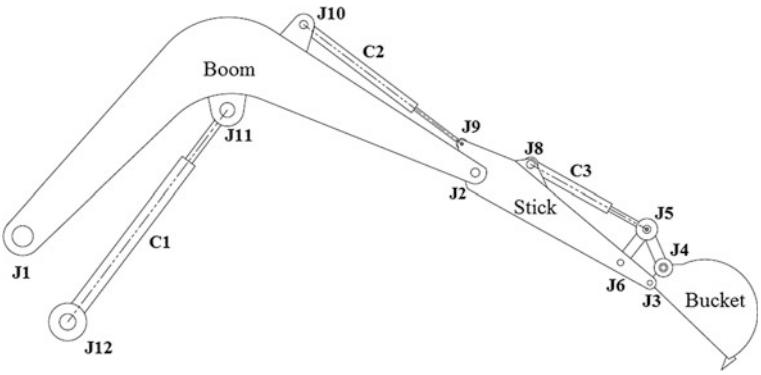


Fig. 13 Typical excavator arm mechanism

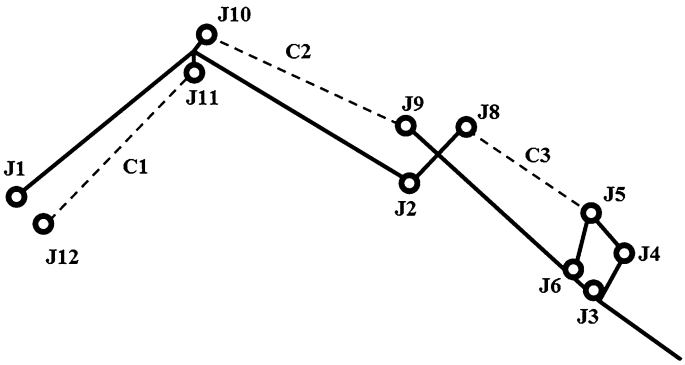
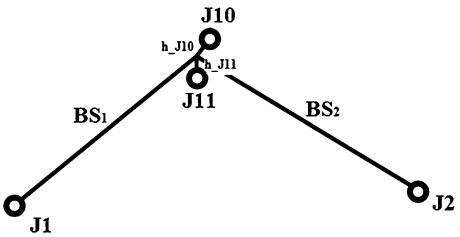


Fig. 14 Skeleton representation of excavator arm mechanism

Fig. 15 Skeleton representation of boom structure



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 (Fig. 16).

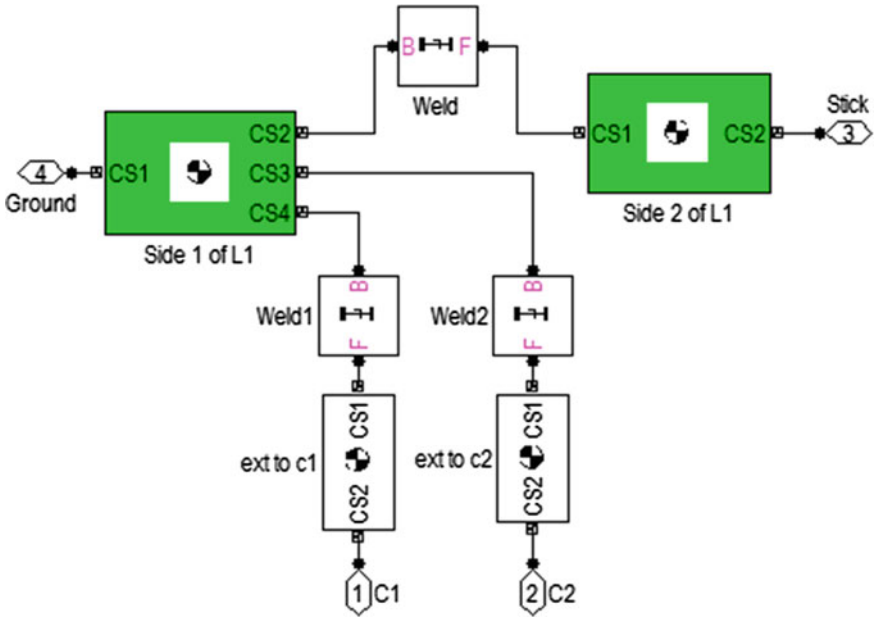


Fig. 16 SimMechanics model of boom

10.4 Stick Construction

The stick has four joints: J2 connects the stick 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 (Figs. 17, 18).

10.5 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 (Figs. 19, 20).

The application point of the ground reaction force is selected in such a way that the overall mechanism will be subject to severe loading conditions. Twisting movements about the *x*-axis and bending movements about the *y*- and *z*- axes register maximum readings when the digging force is applied on the bucket at a

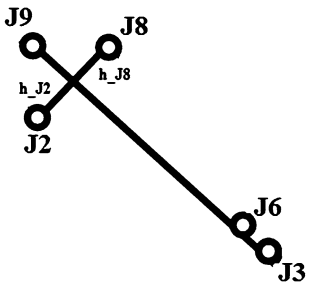


Fig. 17 Skeleton representation of stick

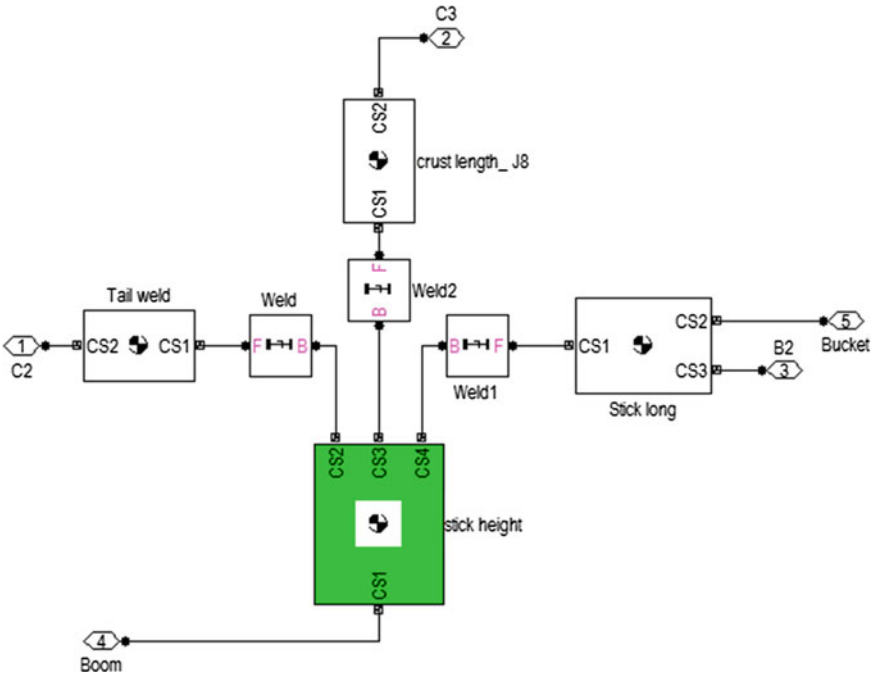


Fig. 18 SimMechanics model of stick

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 Fig. 21.

10.6 Hydraulic Cylinders

Hydraulic cylinders are represented by simple weightless rigid links solely 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

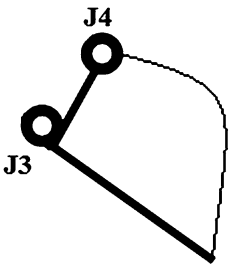


Fig. 19 Bucket schematics

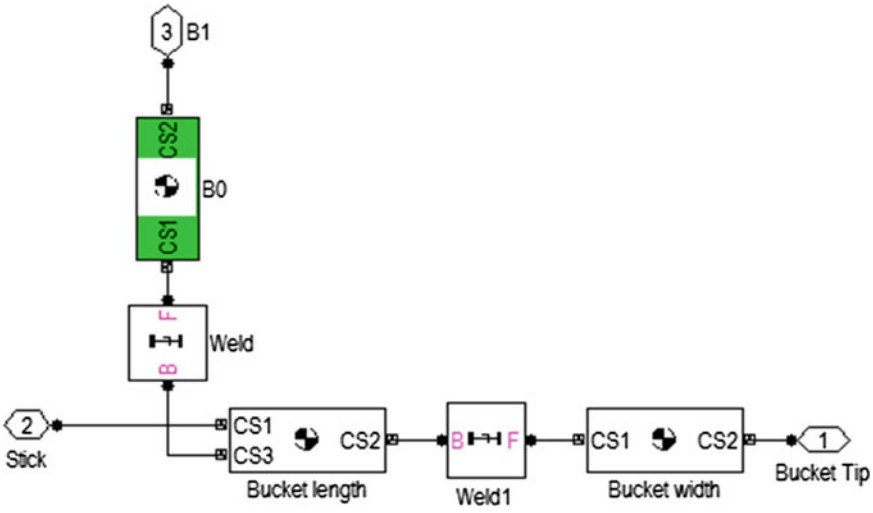


Fig. 20 Bucket SimMechanics model

cylinders. However, this task is beyond the scope of this research and will not be discussed here.

10.7 Transition Four-Bar Linkages

The 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 one more joint needs 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

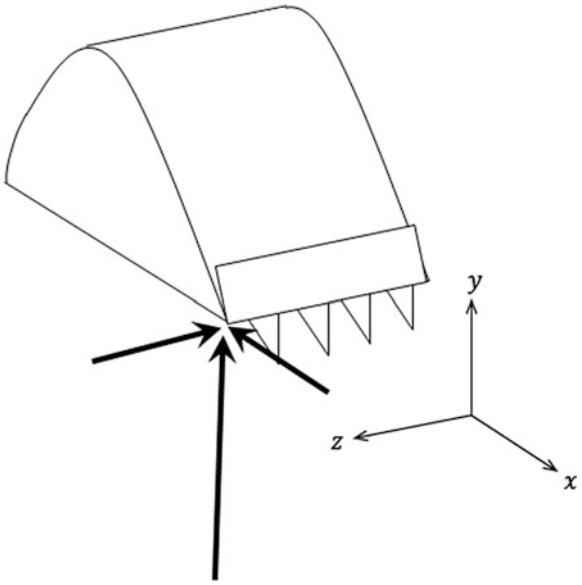


Fig. 21 Location of application point of digging force

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 22 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 two seconds 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 23 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.

11 Example Simulation Analysis

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, which are designed to show the operational procedures of the two proposed methods and their corresponding results.

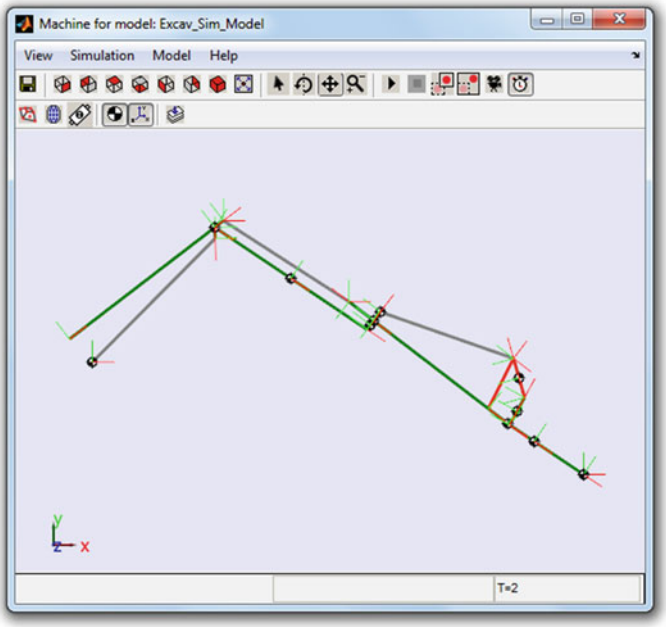


Fig. 22 Real-time simulation environment

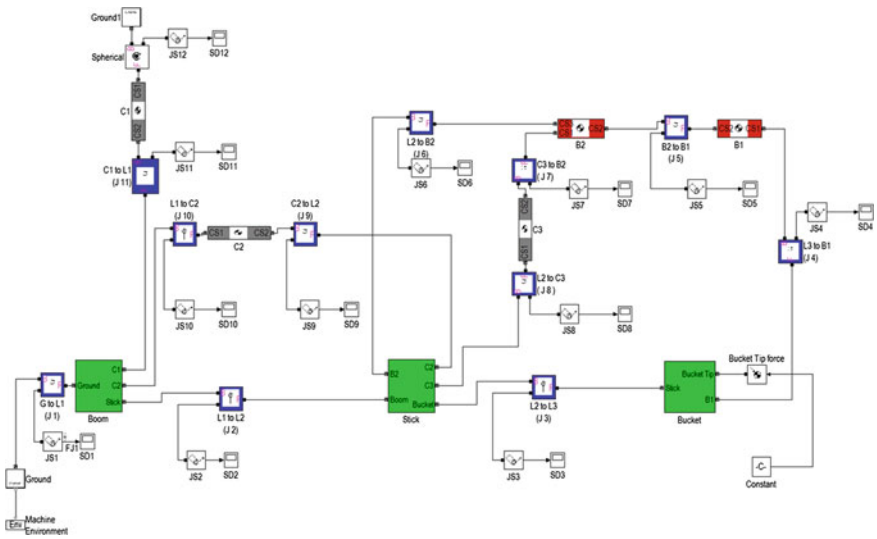


Fig. 23 SimMechanics diagram for an excavator arm mechanism

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 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 for the module handling the dimensional synthesis is a set of required output configurations of the excavator arm mechanism. As discussed before, these values are checked for compatibility to ensure 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 is started by defining the necessary geometric and non-geometric constraints. In addition to these 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_kpsiTS_kpsiYS_MPaTS_MPaElong_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 = 1.0e + 004 *(0.5644, -1.9908, 0)*

12 Feature-Based CAD Embodiment

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® 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. Similar to 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 locations of the shared directories were determined with consideration for the possibility of different tasks being performed on different systems. A free Internet file storage service 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, as 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 an 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];};
```

12.1 Reusability of Functions

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

An effort was undertaken 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 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, and B-spline curves from data points.

12.2 Boom Modeling

The modeling of the boom part is initialized by creating a blank NX.prt file using the function

_Create_New_Part_File(char file_path[UF_CFI_MAX_FILE_NAME_SIZE])

After creating a blank CAD modeling environment, the next step 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 (Fig. 24).

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:

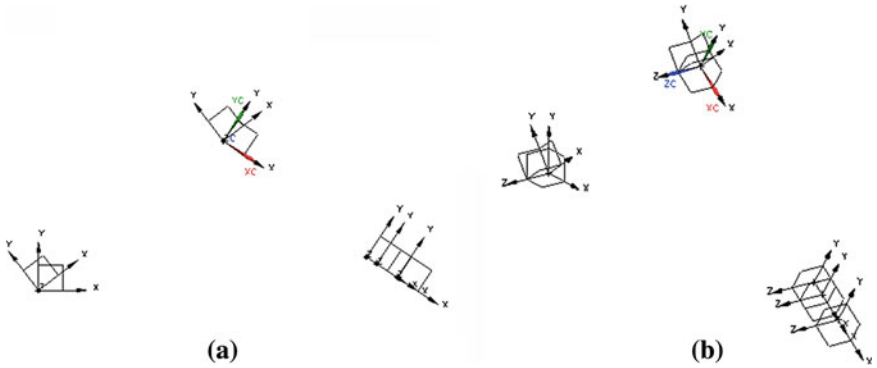


Fig. 24 Boom coordinate system (CSYS) features

- *_CSYS_origin_and_direction(void)*
- *_CSYS_offset(tag_t referece_datum_CSYS,*
  const double linear_offset[3],
  const double angular_offset[3],
  bool operation_sequence)

The optimization result vectors for the two sides of the boom exported from MATLAB® were saved. 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 (Fig. 25).

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 (Fig. 26).

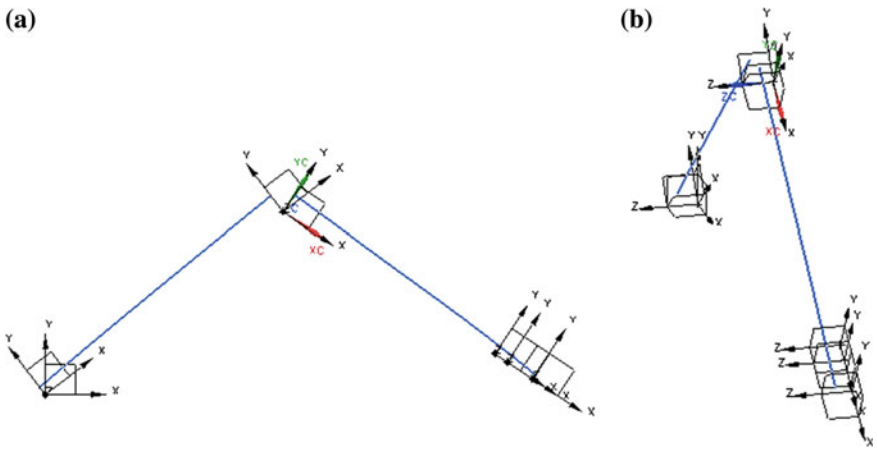


Fig. 25 Boom B-spine curve features

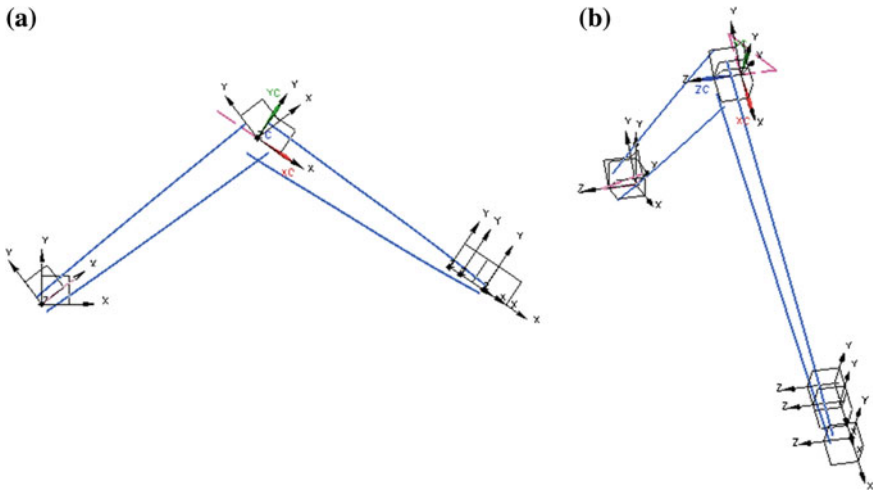


Fig. 26 Evolvement of features

The function used for this purpose is:

_Mirror_a_Curve_through_a_Plane(tag_t Curve_Tag, tag_t Plane_ Tag)

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 _Trim_Curve_by_Datum_Plane(
                tag_t     Curve_Tag,
                tag_t     Datum_Plane_Tag,
                int        which_end);

tag_t _Trim_Curve_by_Curves(
                tag_t     Target_curve_Tag,
                tag_t     Tool_curve1,
                tag_t     Tool_curve2);

tag_t _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 (Fig. 27).

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,

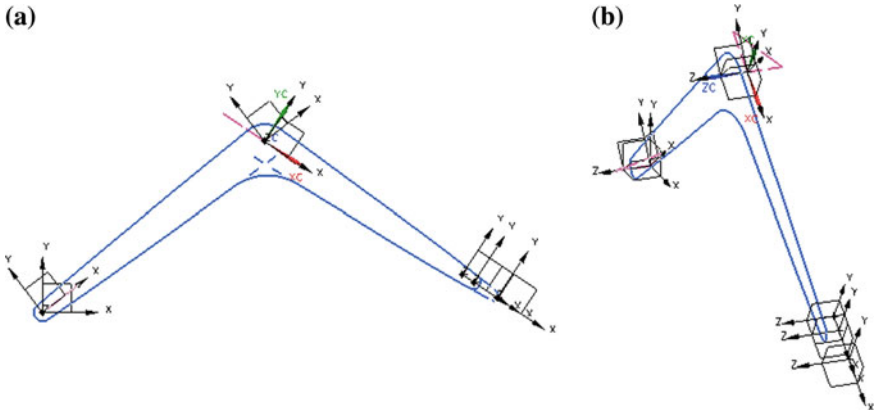


Fig. 27 Joined curve features

shown by the green line in Fig. 28, is projected onto the vertical *x*–*y* plane to form a guideline 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.

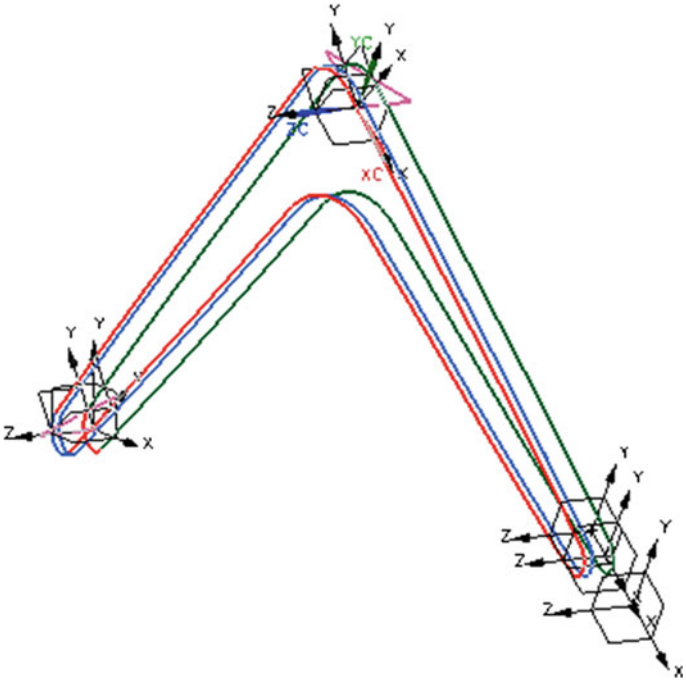


Fig. 28 Embodiment features

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

The third closed curve, colored green in Fig. 28, is created in a very similar procedure as that 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 _Join_Curves(tag_t *curves,int n)
tag_t _Point_from_Spline(tag_t curve_Tag, int line_end)
tag_t _Lines_from_two_points(tag_t point1, tag_t point2)
tag_t _SWEEP_2_guides(tag_t Guide_s1, tag_t Guide_s2, tag_t Section)
tag_t _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 through the plate walls and one that is welded externally to the boom structure.

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 (Fig. 29).

Functions used for this purpose include:

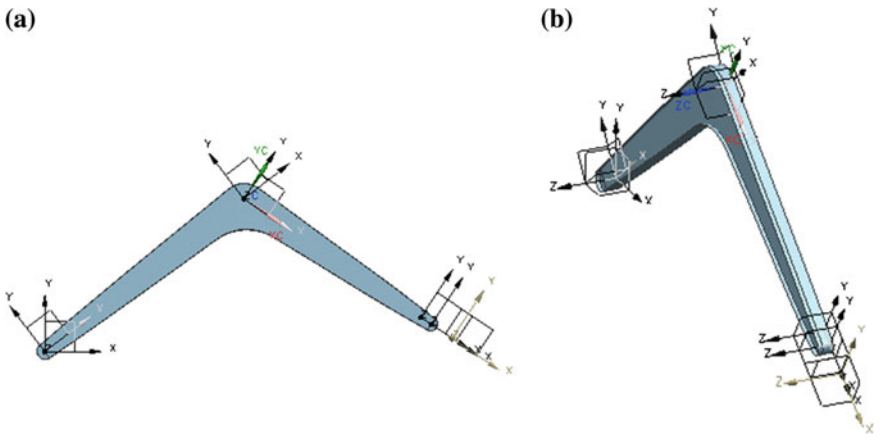


Fig. 29 Sheet body features

```
tag_t    _SKETCHES_J2_adopter(
         char     name[30],
         tag_t    Refrence_CSYS,
         int      Plane_num,
         int      Axis_num)

         tag_t    _ARC_on_sketch(tag_t Reference_CSYS)

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

tag_t _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 _Extrude(tag_t Connected_Curve, char* limit[2])

tag_t _SKETCH_J11(
         tag_t    Reference_CSYS,
         int      Plane_num,
         tag_t    line_Tag,
         tag_t    bridge_tag)

tag_t _SKETCH_J12(
         tag_t    Reference_CSYS,
         int      Plane_num,
         tag_t    Curve_Tag);
```

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 (Figs. 30, 31).

Custom functions used for these operations include:

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

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 to create the final boom solid body shown by Fig. 32.

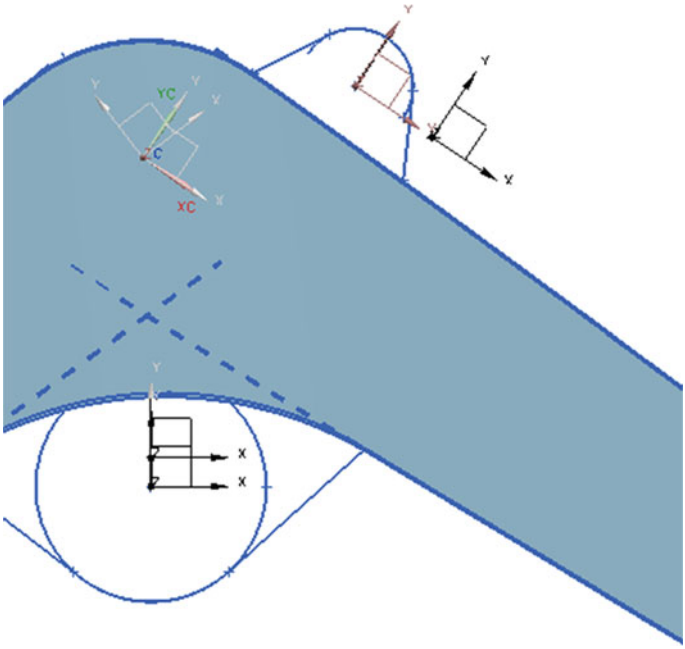


Fig. 30 Hinge joint profiles construction

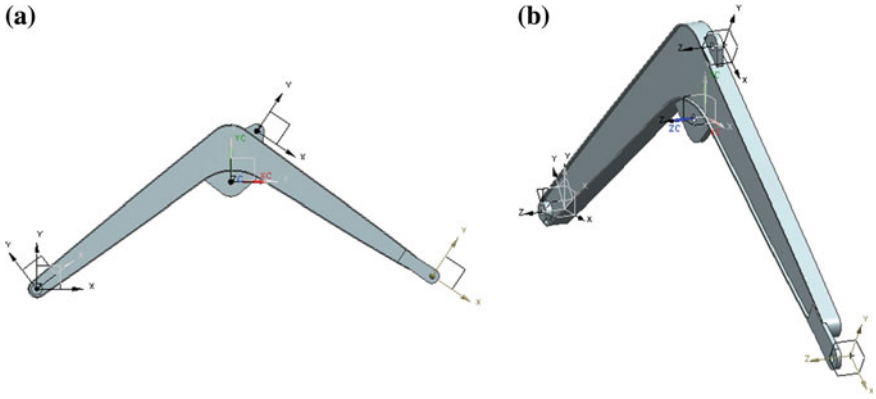


Fig. 31 Solid body features

12.3 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.

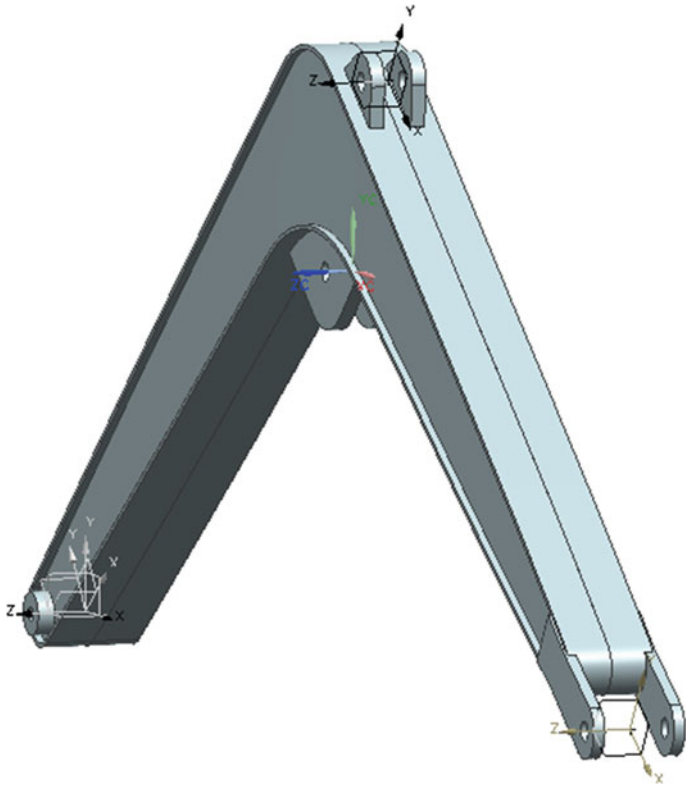


Fig. 32 Final CAD model of an excavator boom

As was 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 were imported from the intermediate files using similar procedures. User-defined CSYSs were created at the joint locations and some critical locations necessary for the creation of sketches.

Generally, the stick construction procedure is easier than that of the boom because of the parallelism of the stick CSYS features (see Fig. 33).

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 Fig. 34.

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 35 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

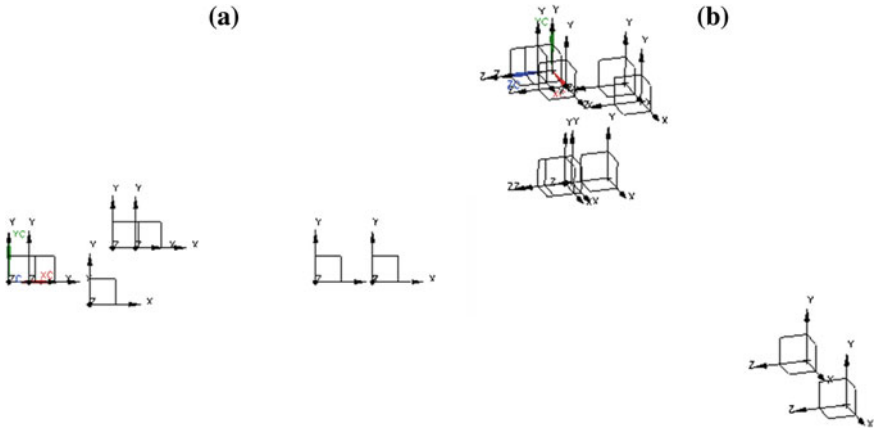


Fig. 33 Stick coordinate system features

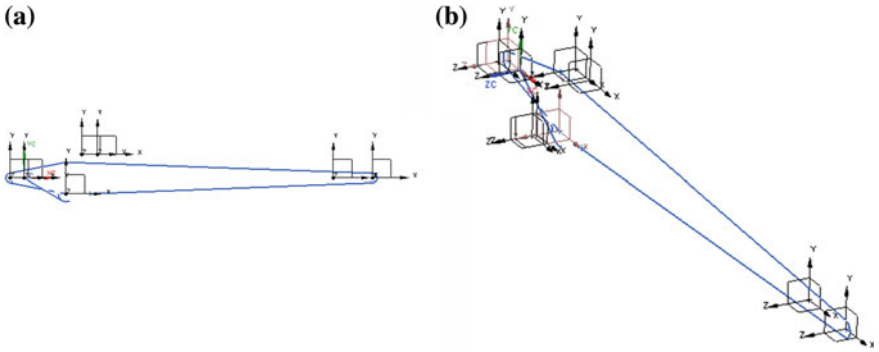


Fig. 34 Feature evolvement

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.

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 (Fig. 36).

The 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 with consideration for necessary inference tolerances. Joints are created using similar procedure as used in the boom modeling. The final left half of the boom is shown in Fig. 37.

The final complete stick solid body feature is created by merging individual extruded and thickened solid features into a single part and mirroring this part

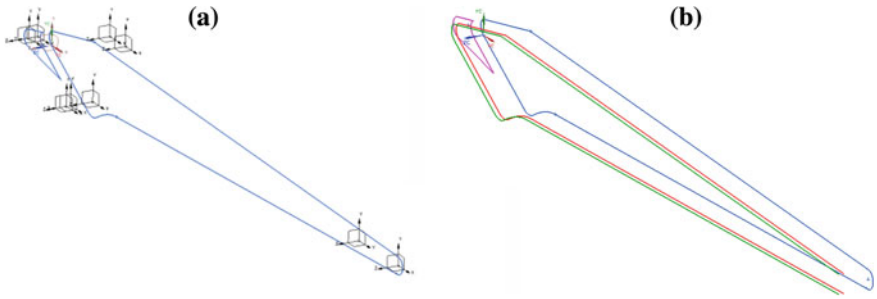


Fig. 35 Embodiment features

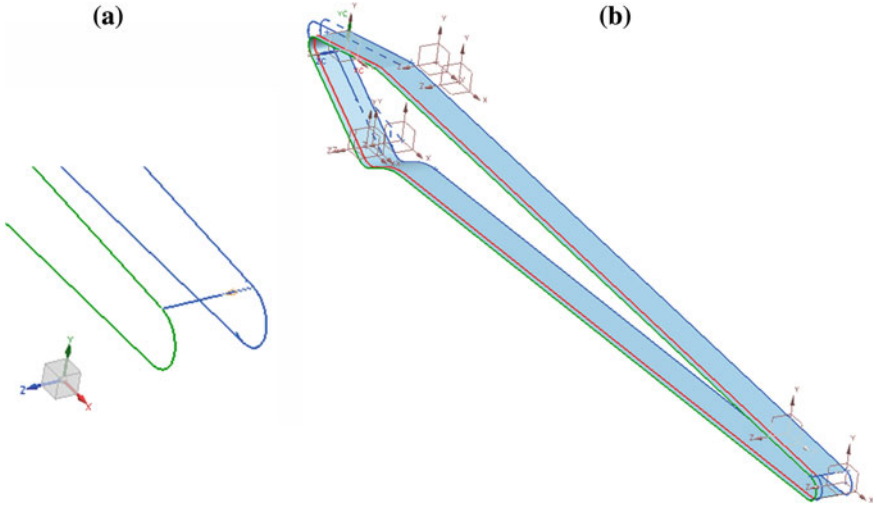


Fig. 36 Stick sheet body features

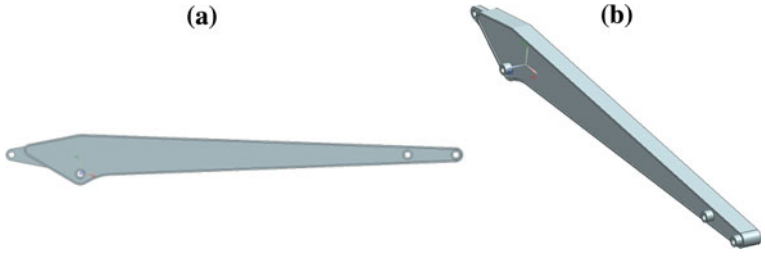


Fig. 37 Stick solid body feature

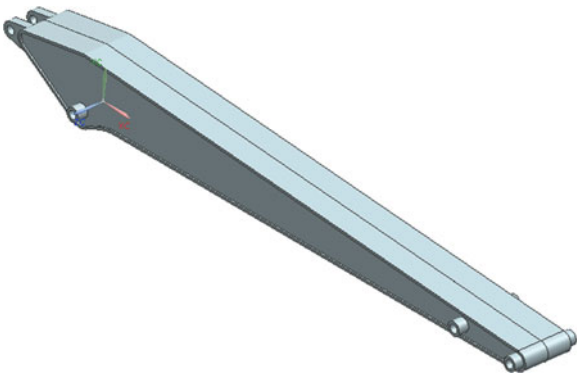


Fig. 38 Final CAD model of an excavator arm stick

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 Fig. 38.

13 Conclusions

The proposed generative feature-based conceptualization method for product development is a promising response to addressing the issues found in 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 through insightful configuration optimization and embodiment development. The method also provides much-needed 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 authors' opinion, regardless how intelligent a design system may be in the future, human design expertise is always required; the developed system can only support decision making more effectively with some productivity tools. The case study presented proves that the knowledge-driven feature-based conceptual design approach can handle traditionally complex challenges such as machine linkage optimization problems.

The proposed hybrid optimization-ANN dimensional synthesis method introduced in the previous chapter has greatly increased the reliability of calculated solutions. Training the ANN with larger size of data collected from the existing products in the market is believed to produce solutions reflective of common design intents and common industrial trends. The hybrid ANN-Optimization method has been proven to provide satisfactory results by generating close initial solutions of the linkage dimensions). The optimization procedure that was also introduced in the

previous chapter was employed to calculate the final linkage dimensions) that satisfy the customer's specification feature requirements. This hybrid method can assist in generating optimal solutions in an integrated design environment.

Together with the previous chapter, this feature-based smart mechanism design method provides a novel approach to 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, as 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.

14 Future Works

Formal definitions of generic conceptual design features need to be investigated such that embedded engineering rules, constraints, data representations, and 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 such as 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 be automatically 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 be similarly used in designing other aspects of the product. For example, the conceptual level design of hydraulic circuit subsystem in the case study can also be modeled and solved under the proposed data and information management scheme by implementing its own conceptualization contents.

Acknowledgments The presented research work was partially supported by Canada Natural Sciences and Engineering Research Council of Canada (NSERC) discovery grant (No. 355454-09) and the University of Alberta GRF (G121140079) grant.

References

1. Basak H, Gulesin M (2004) A feature based parametric design program and expert system for design. Math Comput Appl 9:359–370
2. Bronsvoort WF, Bidarra R, Van DM (2010) The increasing role of semantics in object modeling. Comput Aided Des Appl 7:431–440
3. Danjou S, Lupa N, Koehler P (2008) Approach for automated product modeling using knowledge-based design features. Comput Aided Des Appl 5:622–629

4. Durupt A, Remy S, Ducellier G (2010a) KBRE: a knowledge based reverse engineering for mechanical components. Comput Aided Des Appl 7:279–289
5. Durupt A, Remy S, Ducellier G (2010b) Knowledge based reverse engineering: an approach for reverse engineering of a mechanical part. J Comput Inf Sci Eng 10:044501
6. Larsen S, Jensen CG (2009) Converting topology optimization results into parametric CAD models. Comput Aided Des Appl 6:407–418
7. Li M, Zhang YF, Fuh JYH (2010) Retrieving reusable 3D CAD models using knowledge-driven dependency graph partitioning. Comput Aided Des Appl 7:417–430
8. Lourenco D, Oliveira P, Noort A (2006) Constraint solving for direct manipulation of features. Artif Intell Eng Des Anal Manuf 20:369–382
9. Ma YS, Britton GA, Tor SB (2007) Associative assembly design features: concept, implementation and application. Int J Adv Manuf Technol 32:434–444
10. Ma YS, Chen G, Thimm G (2008) Change propagation algorithm in a unified feature modeling scheme. Comput Ind 59:110–118
11. Ma YS, Tang S, Au CK (2009) Collaborative feature-based design via operations with a fine-grain product database. Comput Ind 60:381–391
12. Mantyla M, Nau D, Shah J (1996) Challenges in feature-based manufacturing research. Commun ACM 39:77–85
13. Myung S, Han S (2001) Knowledge-based parametric design of mechanical products based on configuration design method. Expert Syst Appl 21:99–107
14. Ong SK, Shen Y (2009) A mixed reality environment for collaborative product design and development. CIRP Annals: Manuf Technol 58:139–142
15. Pratt MJ, Anderson BD (2001) A shape modelling applications programming interface for the STEP standard. Comput Aided Des 33:531–543
16. Riou A, Mascle C (2009) Assisting designer using feature modeling for lifecycle. Comput Aided Des 41:1034–1049
17. Shigley JE, Misheke CR (2001) Mechanical engineering design. McGraw-Hill Higher Education, Inc, New York
18. Singh DK, Jebaraj C (2008) Feature-based design for process planning of the forging process. Int J Prod Res 46:675–701
19. Solazzi L (2010) Design of aluminium boom and arm for an excavator. J Terrramech 47:201–207
20. Ter Hofstede AHM, Lippe E, Van DW (1997) Applications of a categorical framework for conceptual data modeling. Acta Inform 34:927–963
21. Thakur A, Banerjee AG, Gupta SK (2009) A survey of CAD model simplification techniques for physics-based simulation applications. Comput Aided Des 41:65–80
22. Van Der Meiden HA, Bronsvoort WF (2009) Modeling families of objects: review and research directions. Comput Aided Des Appl 6:291–306
23. Verdes D, Stan S, Manic M (2009) Kinematics analysis, workspace, design and control of 3-RPS and TRIGLIDE medical parallel robots. 2nd conference on human system interactions. Catania, Italy
24. Wang H, Zhou X, Qiu Y (2009) Feature-based multi-objective optimization algorithm for model partitioning. Int J Adv Manuf Technol 43:830–840
25. Wang Q, Li J, Wu B (2010) Live parametric design modifications in CAD-linked virtual environment. Int J Adv Manuf Technol 50:859–869
26. Wong LM, Wang GG (2003) Development of an automatic design and optimization system for industrial silencers. J Manuf Syst 22:327–339
27. Ye X, Liu H, Chen L (2008) Reverse innovative design: an integrated product design methodology. Comput Aided Des 40:812–827
28. Yoon J, Manurung A (2010) Development of an intuitive user interface for a hydraulic backhoe. Autom Constr 19:779–790