12-2012

Integrated modeling and analysis methodologies for architecture-level vehicle design.

Rostyslav Lesiv
University of Louisville

INTEGRATED MODELING AND ANALYSIS METHODOLOGIES FOR ARCHITECTURE-LEVEL VEHICLE DESIGN

By

Rostyslav Lesiv
M.Eng. Mechanical Engineering, 2006
Ivan Franko National University of Lviv, Ukraine

A Dissertation
Submitted to the Faculty of the
J.B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Department of Mechanical Engineering
University of Louisville
Louisville, Kentucky

December 2012

INTEGRATED MODELING AND ANALYSIS METHODOLOGIES FOR ARCHITECTURE-LEVEL VEHICLE DESIGN

By

Rostyslav Lesiv
M.Eng. Mechanical Engineering, 2006
Ivan Franko National University of Lviv, Ukraine

A Dissertation Approved on November 15, 2012,
by the Following Dissertation Committee:

Glen Prater, Ph.D., Dissertation Director

Matthew P. Castanier, Ph.D., Committee Member

David A. Lamb, Ph.D., Committee Member

Gary M. Osborne, Ph.D., Committee Member

Rammohan K. Ragade, Ph.D., External Committee Member

ACKNOWLEDGEMENTS

The writing of this dissertation was one of the most significant academic challenges I have had to face. I would never have been able to finish my work without the help from my advisor, guidance of the committee members, and support from my wife.

I would like to express my deepest gratitude to my advisor, Dr. Glen Prater Jr., who provided me with excellent guidance, insightful advices throughout the dissertation project, and patiently corrected my writing.

I wish to thank Dr. Lamb, Dr. Castanier, and Dr. Ragade, who served on my dissertation advisory committee, for their insightful comments about vehicle development, software engineering and suggestions for improving my dissertation.

I would like to thank Dr. Osborne, who was always willing to help and give his best suggestions. My research would not have been possible without his helps.

I would also like to thank Robert Meyers, Yindong Yu and Dylan Meador for their invaluable programming support and assistance with learning C#.

Finally, I would like to thank my wife, Svitlana. She was always there cheering me up and stood by me through the good times and bad.

ABSTRACT

INTEGRATED MODELING AND ANALYSIS METHODOLOGIES FOR ARCHITECTURE-LEVEL VEHICLE DESIGN

Rostyslav Lesiv

November 15, 2012

In order to satisfy customer expectations, a ground vehicle must be designed to meet a broad range of performance requirements. A satisfactory vehicle design process implements a set of requirements reflecting necessary, but perhaps not sufficient conditions for assuring success in a highly competitive market. An optimal architecture-level vehicle design configuration is one of the most important of these requirements. A basic layout that is efficient and flexible permits significant reductions in the time needed to complete the product development cycle, with commensurate reductions in cost. Unfortunately, architecture-level design is the most abstract phase of the design process. The high-level concepts that characterize these designs do not lend themselves to traditional analyses normally used to characterize, assess, and optimize designs later in the development cycle.

This research addresses the need for architecture-level design abstractions that can be used to support ground vehicle development. The work begins with a rigorous description of hierarchical function-based abstractions representing not the physical configuration of the elements of a vehicle, but their function within the design space. The hierarchical nature of the abstractions lends itself to object orientation - convenient for

software implementation purposes - as well as description of components, assemblies, feature groupings based on non-structural interactions, and eventually, full vehicles. Unlike the traditional early-design abstractions, the completeness of our function-based hierarchical abstractions, including their interactions, allows their use as a starting point for the derivation of analysis models.

The scope of the research in this dissertation includes development of meshing algorithms for abstract structural models, a rigid-body analysis engine, and a fatigue analysis module.

It is expected that the results obtained in this study will move systematic design and analysis to the earliest phases of the vehicle development process, leading to more highly optimized architectures, and eventually, better ground vehicles. This work shows that architecture level abstractions in many cases are better suited for life cycle support than geometric CAD models. Finally, substituting modeling, simulation, and optimization for intuition and guesswork will do much to mitigate the risk inherent in large projects by minimizing the possibility of incorporating irrevocably compromised architecture elements into a vehicle design that no amount of detail-level reengineering can undo.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Finite element analyses and rigid body analyses are extensively used in the automotive industry as product development tools. These computer models can help predict the mechanical characteristics and performance of a vehicle design prior to fabricating and testing prototypes, resulting in shortened design cycles and reduced costs. While the engineering value of such analyses is very well established, current vehicle development methodologies have an inherent limitation. High-precision analysis models require topologically accurate geometric descriptions as a basis for abstraction. CAD solid models are generally a product of the detail design phase, and are not available during conceptual design. Architecture design is inherently conceptual in nature, qualitative rather than quantitative, and based upon precedent and designers' experience-based intuition. As a result, computer analyses tend to play little role in vehicle concept development and assessment. Optimization based upon detailed CAD models tends to focus on detail level features rather than the fundamental vehicle architecture. There is a compelling need for vehicle modeling and analysis methodologies supporting the quantitative assessment and optimization of vehicle architecture design early in the design process. This work presents a methodology for developing an integrated suite of modeling and analysis tools to meet this need.

1.2 Problem Background

Analysis models all have their basis in some sort of physical description of the design domain. At the architecture/conceptual level, this abstraction is of necessity fundamentally different than a detail-level abstraction. Rather than describing three-dimensional continua of a collection of components, an architecture-level abstraction should describe fundamental function and arrangement, while capturing enough physical detail to be used as the basis for meaningful analyses. In general, these abstractions must include general shape and layout, major spatial features (nominal volumes, in particular), component and subsystem connections (which can be quite simple relative to a detailed CAD model), and inertia properties. They should describe critical subsystems, including, but not limited to primary body structure, suspensions and powertrain, and the human operator and passengers. The design information captured by the abstractions must be available at the very earliest stages of the vehicle developing process so that the model itself functions as a "design space for ideas".

Provided a suitable design, abstraction can be formulated. The architecture model can be used as a starting point for the derivation of analysis models. At a minimum, the analysis models and associated analysis algorithms should be able to determine gross dimensions (length, height, width, ground clearance, closure opening sizes, ergonomic features such as head clearance and reach envelope, cargo/functional compartment volumes, *etc.*), inertia properties (mass, mass moments of inertia), rigid body structural characteristics, forced rigid body response, structural dynamic characteristics, structural static strength, and structural fatigue strength.

Implementation of this list of analysis requirements mandates the use of preprocessing algorithms to create three analysis model families. Simplest of the three is a model for determining gross dimensions, physical sizes, absolute and relative locations, and inertia properties. These spatial, kinematic, inertia (SKI) models can be based closely upon the architecture abstraction, but must include enough information to describe the extent of the design domain, as well as functions that discretize component-like features to calculate composite lengths, areas, volumes, masses, mass centers, and moments of inertia.

A second, more complicated model formulation is required for rigid body analyses. These analyses can be based upon discrete system models with lumped mass/inertia, damping, and stiffness elements. The most critical aspect of the model generation step is the portion of the algorithm used to make discretization decisions. The architecture-abstraction must be complete enough to calculate the corresponding element parameters. Mathematically, the model will consist of coupled linear or nonlinear ordinary differential equations. Complementary analyses for such models include determination of rigid body natural frequencies and modes, frequency response functions, and forced and unforced time domain response.

A third, and final, primary analysis model type is needed to support finite element based structural calculations. As with rigid body models, a critical step in the finite element model synthesis process is discretizing and connecting architecture level components. The discretization algorithm must determine whether a component will be fully structural, with inertia, damping, stiffness characteristics included in element continua, non-structural with inertia effects present, or non-structural without mass.

Surface/panel discretization is a particular concern. Architecture-level analyses are intended to support the design process, and accordingly, should provide near real-time feedback to the designer. Use of finely meshed two- and three-dimensional shell and plate elements dramatically lengthen solution times. As a design tool, the FEA meshing algorithms should be fully automatic, and not require the designer to be an expert in finite element preprocessing.

The rendering conventions used to display the architecture design model can be adapted to display analysis results. Color contours can be superimposed upon the architecture depictions to denote continuum-mapped parameter values (stresses, strains, factors of safety, reliability, etc.). A vehicle rendering can be animated to normal modes, generalized time domain response, and response to terrain inputs.

In addition to being capable of rapid solution times, a viable analysis suite should be capable of preparing architecture analysis models for full vehicles, a primary subsystem (a cab or rolling chassis, for example), and in some cases, single features or components. It must have access to visual renderings that depict the vehicle architecture's physical layout while allowing designer to interact with both the physical design space and analysis results. Individual modules of the analysis suite should be algorithmically integrated. Finally, in order to become a practical vehicle architecture design tool, these methodologies must be implemented as a software package using modern object oriented programming (OOP) techniques that permit the use, programmatic modification, and reuse of data structures, including the methods that will implement the calculations performed as part of individual analyses. Design domain discretization methods, for

example, are required for rendering and SKI, RB, and FE analyses, and should be reused whenever possible.

In addition to software implementation and code reuse benefits, an integrated analyses suite derived from a common set of architecture geometry objects offers another important benefit. It will be possible to perform "composite" analyses that use the results from one analysis as an input or constraint for a different analysis type. Some of these composite analyses will be performed naturally as a result of good object oriented programming practice. SKI analysis methods, for example, will be invoked as part of the discretization process required for formulating rigid body models.

The evaluation of the structural performance of a vehicle model is often performed using static, time-invariant analyses. However, all vehicles are subject to fluctuating loads, and fatigue tolerance is an important design consideration. Not considering the possibility of structural cracking can result in either inefficient, over-designed structures, or structures that are prone to premature cracking. Because fatigue crack propagation is detail geometry dependent, it may not be possible to obtain accurate quantitative results with an architecture-level model. However, qualitative fatigue analyses can be performed. For instance, comparative fatigue analysis can be of interest when choosing between several vehicle configurations. Comparative fatigue analysis can be performed as a composite analysis. Such an implementation would combine three types of analyses: rigid body analysis, static FE analysis and fatigue analysis. In order to obtain proper loading configuration, the resulting joint reactions and inertial forces can be acquired from the rigid body engine. Consequently, resulting forces are used in a static FE analysis, and afterwards in the fatigue engine. Results of the analysis can be presented

as color contour plot showing deterministic and stochastic fatigue reliability characteristics of competing architecture-level vehicle designs.

1.3 Objectives and Scope

The research focuses on developing, refining, and implementing a wide range of methodologies and algorithms for modeling and analyses of a vehicle concept model. Specifically, the dissertation involves five primary objectives:

1. Formally describe an integrated methodology for creating and displaying architecture abstractions comprehensive enough to both represent vehicle architecture at the earliest stages of the design process, and serve as the basis for generating SKI, rigid body, and finite element analysis models. Explain how the integrated design and analysis abstractions can be used to enhance vehicle development processes.
2. Develop, implement, and, optimize algorithms for the surface modeling, rendering and editing of vehicle architecture design models.
3. Formulate a fully automatic meshing module capable of generating finite element models for panels with arbitrary complexity.
4. Implement and optimize a rigid body analysis engine capable of deriving analysis models from generalized architecture concept models, calculating natural frequencies and modes, frequency response functions, and forced and unforced time domain vehicle response.
5. Develop and optimize a fatigue analysis module that is coupled with the static FEA engine, and is capable of visual representation of fatigue reliability contour plots.

Addressing the first objective begins with an explanation of the need for conceptual-level optimization phase as an effective paradigm for reducing vehicle development time and detecting flaws of the architecture early in the development process. A list of appropriate architecture-level analyses is outlined, along with the design information required for each analysis. The corresponding dissertation chapter also includes the systematic description of vehicle component abstractions. All vehicle components can be grouped in the structural and non-structural component categories. For each feature, depending on its functional responsibility, a set of properties sufficient for meaningful architecture-level analysis is discussed. The hierarchical organization and data structure for the vehicle components and systems is represented. Finally, the hierarchical representation of analyses based on the corresponding abstraction level is demonstrated.

The surface modeling algorithm of objective 2 is capable of creating mathematical representations of a surface, as well as its tessellation for visual representation. First, the mathematical model of the B-Spline curve was created and implemented in software. A designer is able to define an arbitrary number of control points on a curve and modify their positions. Next, the algorithm interpolates the corresponding surface using boundary curves. Control over the surface shape is allowed after converting the interpolated patch to the B-Spline type of surface. The algorithm supports the defining of curves on a surface, and thus allowing the modeling of trimmed surfaces. Curves on a surface are treated the same way as B-Spline curves, but in the parametric space of the underlying surface. Finally, a rendering algorithm creates an accurate curvature-dependent tessellation which is rendered using XNA redistributable library.

Panel meshing algorithm forming the basis of objective 3 generates a high-quality grid on an arbitrary trimmed surface. The algorithm accounts for surface curvature by increasing mesh density in highly curved regions. Both structured and unstructured meshing schemes have been implemented. The algorithm takes into account any internal or boundary constraints that may represent points of loading or connection. Finally, to guarantee smooth load transition between rigidly connected panels, the algorithm synchronizes meshing interfaces between the patches of the composite surface.

Rigid body analysis algorithm of objective 4 automatically extracts the rigid body model from the arbitrary vehicle concept model configuration. In order to decrease the computational burden for the solver, the algorithm combines rigidly connected bodies into a single rigid body. Users are provided tools for defining the shape of the terrain profile exciting the response. The algorithm for rigid body analysis detects the driving and braking sets of wheels, for which driving velocity or braking torque can be applied. Results of a simulation are shown as the animation of vehicle driving on terrain or plots of the vehicle displacement, velocity and acceleration as a function of time. In addition, the solution algorithm is used to determine vehicle frequency response characteristics.

The fatigue algorithm uses quasi-static stresses analysis results from the FEA engine to predict the safety margins for an architecture-level structural assembly subjected to time varying load profiles. Both deterministic and stochastic analyses have been implemented in the engine. Stress-life and strain-life approaches have been used in order to provide preliminary qualitative fatigue life estimations for both high-cycle and low-cycle ranges. Fatigue stress concentration is taken into account by applying a stress concentration factor in the regions where two or more structural components are

connected. Results are shown as contour plots of safety factor, reliability, or probability of fatigue failure.

CHAPTER 2

REVIEW OF PREVIOUS WORK

2.1 Vehicle Concept Modeling

Kojima [1] has described the evolution of the vehicle design processes from the conventional approach to the one with an increased emphasis on concept modeling. We can characterize this procedure as a sequence of processes involving planning, design, documentation, evaluation, optimization, and production activities (final flowchart in Figure 1). The loop that is comprised of CAE and "Grand Design" (Kojima's term for architecture-level design) is used to create a good conceptual design proposal. Comparing to the conventional development procedure, expensive iterative evaluation of prototypes can be avoided, hence significantly reducing the expense associated with making prototypes. Furthermore, the author has proposed First Order Analysis (FOA), the analysis tool developed to execute preliminary analyses simultaneously with the model creation. In his implementation, Kojima uses Microsoft Excel spreadsheets to hierarchically distribute automobile components among designers, while the relationships between parts are maintained. FOA serves as a designer's tool for the optimal structure creation by means of the topology optimization. Two central analyses used in FOA are FEA, which provides information about the strains and stresses distribution in a model, and modal analysis that is used to predict vibration modes.

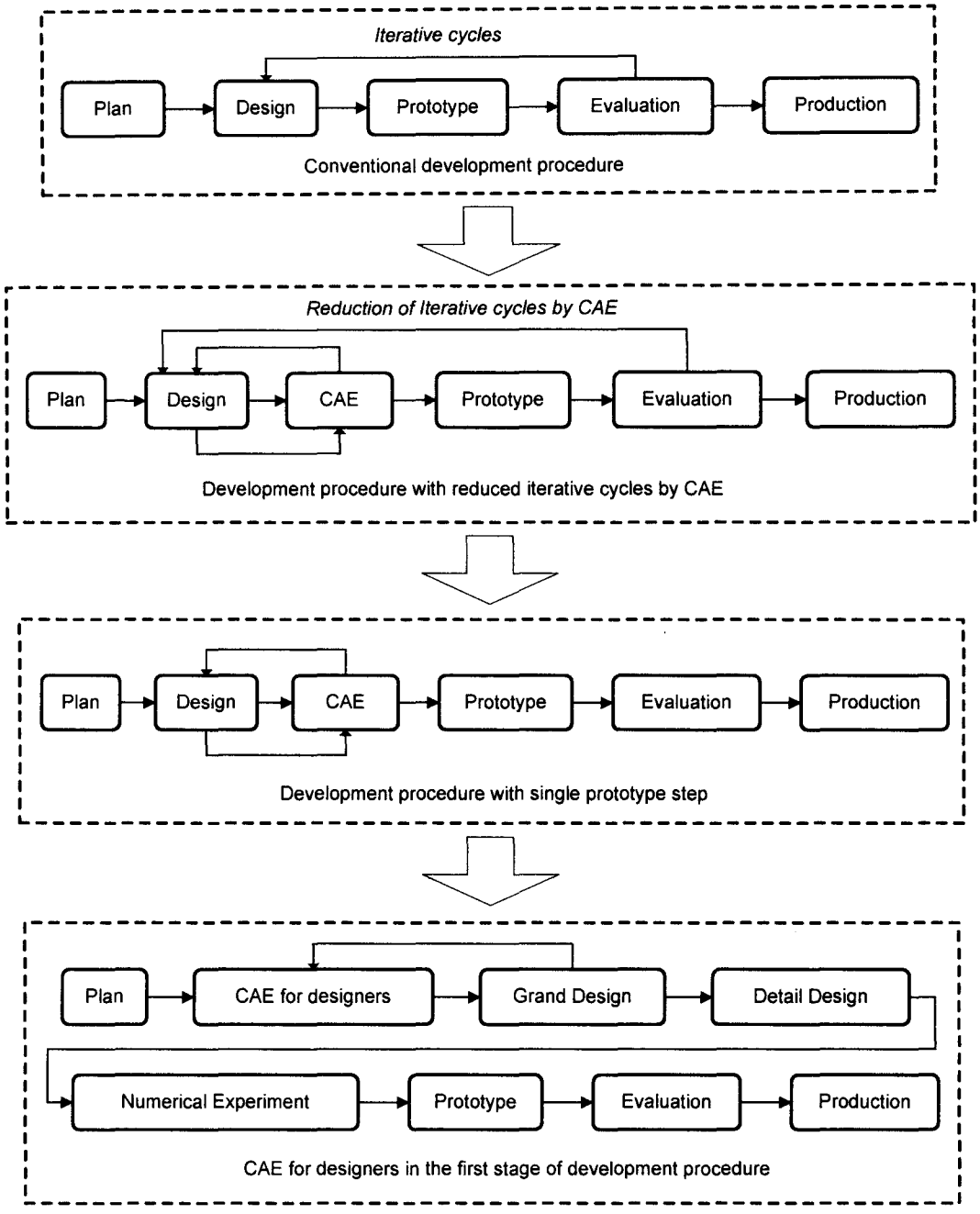


Figure 1. Evolution of the vehicle design process (adapted from Kojima [1]).

Hou et al. [2] has developed the ACD-ICAE (Auto-body Concept Design-Intelligent Computer Aided Engineering) software suitable for the concept design phase of vehicle development. This tool for auto-body modeling, analysis and optimization permits quick creation of a geometric model for a conceptual vehicle design, as well as

generation of FE models. Many types of auto-body templates have been integrated into the software, and template geometry can be modified by changing the control points. Like Kojima' FOA approach, this software is limited to FE-based analysis and optimization techniques. Moreover, ACD-ICAE is not capable of modeling and obtaining analysis responses from other essential components of a vehicle that can be available during the conceptual phase, like suspensions, wheels, powertrain, seats and crew members.

Torstenfelt and Klarbring [3] presented an efficient pre-3D CAD tool for optimization of modules shared among product of the same product family platform. The optimization algorithm is built upon a FE solver; as a result, the optimization can be performed in one run. Car geometry is defined using only points, lines and parametric curves, assuming that no surface information available yet at the current design stage. Alternatively, the effect of surfaces (windows, floor and roof) is modeled as distributed line masses. Features like seats, fuel tank, gearbox, and side doors are treated as point masses. Design variables were divided in two groups: sizing and shape variables. First group is responsible for the cross-sectional geometry and orientation of the beams, while the second group variables control the positioning of beam connection points.

Schelkle and Eslenhans [4] compared the early and series stages of vehicle development process, and explained the importance of the conceptual phase of the design process. It has been pointed out that detailed models are usually too complicated to analyze many of them, thus the resulting model is chosen, in some sense, by chance, based on the limited number of concepts admitted for the analyses. Authors have presented the CAE tool for the vehicle concept modeling and optimization based on the

three steps procedure. First step is topology optimization, which serves to find out where material should be located. In detail, the structural space is defined by subtracting the functional packaging space from the vehicle basic solid. Second step is the parametric concept design level, used for the layout optimization to prepare the body concept model. Finally, uncertainty is taken into account by using the stochastic concept assessment, which serves to optimize the shape, size and material of a model. SFE CONCEPT software [5] has been used for the concept model creation on the second step. The tool assists a designer in developing the surface-based parameterized vehicle geometry within a very short period of time. With the help of the software, a model can be meshed, and analysis can be accomplished using the NASTRAN solver. SFE CONCEPT is an effective instrument for quick generation and modification of concept models.

Bylund in his PhD thesis [6] has developed the tools, which facilitate a designer in developing a simulation-driven design, rather than a simulation-verified. It has been suggested that supporting preliminary analysis by the design engineers at the early stage can save analysis time during the detailed stage. The tools implemented by the author permit two types of analysis for joints and beams: linear and nonlinear FE analyses.

Prater et al. [7, 8] have compared concept finite element vehicle model against detailed model. The resulting NVH parameters correlated well with the result achieved from detail model analysis. Later, it has been recognized that representation of body joints is usually the main difference between concept models. Therefore, efforts have been made to create finite element joint models. Shahhosseini et al. [9] has presented the joint modeling literature review, where many joint modeling techniques have been discussed. Besides that, two concept models with different connection types have been

tested and compared to the detailed model representation. Superelement elastic joints have been used for the first test case, and tri-spring joints for another. For both bending and torsional cases model with tri-spring joint representation is more favorable over the model with superelement elastic joints. Therefore, tri-spring joint representation is more suitable for the analysis at the early stage of vehicle development process.

Somewhat similar research has been done by Osborne et al. [10]. The authors created three light-duty pickup truck box concept models, each incorporating one of the major compliance joint types: rigid single-point connections, distributed rigid connections, and elastic superelement connections. The NASTRAN FE solver was used to estimate bending stiffness, torsional stiffness, fundamental torsional frequency, and fundamental bending frequency. The results of the numerical experiment have been compared to the analogous results from the FE analysis of the detailed box model. The model with the distributed rigid connections demonstrated the best correlation with the detailed model. Coincident node MCJ modeling techniques produced overly stiff results, and thus is inappropriate for the joint modeling. Difficulty with obtaining optimal spring constant and complexity of modeling methodology of elastic joints make them impractical for the use in concept modeling. Additionally, the paper presents a comprehensive description of the overall FE vehicle concept modeling methodologies, using four primary load carrying structural component types: beams, panels, assembly joints, and MCJ.

Donders et al. [11] have given a state-of-art overview of concept CAE methods divided into three categories: methods based on predecessor FE models, methods from scratch, and methods concurrent with the CAD design. The first category of methods use

existing detailed FE models for initial concept model derivation. Drawbacks of the existing model are identified and fixed by changing the initial FE grid. Mesh model can be modified by using mesh morphing or concept modification approaches. Methods concurrent with CAD deal with integrated CAE-CAD tools that allow immediate component modification after analysis has been performed. In the second part of the paper the authors have proposed the beam and joint replacement methodology for concept model creation. Afterward, the optimization process has been used to improve global NVH behavior by modifying the simplified beam and joint elements. Global optimum is found by minimizing an objective function, constructed as a weighted sum of the global modes frequencies. Similarly to the paper [11], the improved beam and joint concept elements have been proposed in the series of works [12, 13, and 14]. The technique introduced by Mundo et al. [12] has been validated using two static load cases. The stiffness of the vehicle under the applied torsion and bending load has been evaluated, and the difference between the original and the concept model was less than 1% for both load cases. In the [14] the authors presented the new connection element RBE2.5 that represents the beam-to-joint connection. The element can be considered as a composition of RBE2 and RBE3 Nastran elements. According to the authors RBE3 element is too flexible and not appropriate to describe rotation of the beam and joints end-section, and RBE2 is too stiff. As a result RBE2.5 element takes the advantages of both RBE2 and RBE3 elements, and is capable of accurate displacement predicting.

Dai and Duan [15] pointed out that there is sufficient information in the concept phase of car design to perform the reasonably accurate analysis for determining essential crash dynamic characteristics. An artificial neural network technique has been applied to

extract the relationship between the crash dynamic characteristics and beam element features. When the expected crash dynamics characteristics are established, after the training process, the method can be used to predict beam element features, based on the relationships derived by neural network.

2.2 Surface Modeling for CAD Applications

Parametric curves and surfaces are fundamental tools for automotive geometric design. There are many books related to CAD modeling and dedicated to parametric curves and surfaces. The field has been actively growing since the 60-s of the last century, however, in this review we will only focus on some works done over the period of the past 15 years, and which are the most relevant to the dissertation.

There is an extensive amount of literature dealing with design and shape modification of parametric surfaces. A number of great handbooks describing the basics of rational curve and surface modeling have been published over the past 20 years by Goldman [16], Farin et al. [17, 18], Salomon [19], and Saxena et al. [20]. Nevertheless, most of the methods described are more theoretical rather than practical. Piegl and Tiller have written a very valuable handbook for practical usage titled "The NURBS book" [21]. This handbook is a comprehensive source of information for those who not only do theoretical work, but also want to have efficient algorithmic implementations. It contains many algorithms that go beyond plain theory and are optimized for efficient calculations.

2.2.1 Curve Modeling

Parameterization. The problem of optimal curve parameterization is important not only for creating a curve with intuitive shape, but also for surface meshing, as well as curve and surface integrals calculation. Chord-length, equally-spaced and centripetal

parameterization techniques are thoroughly investigated, however a few other methods of parameterization can be found in the literature. Most classic parameterization methods don't work well in some cases and create undesirable bulges. As a result, much effort has been done to develop better parameterization schemes. Lim [22] came up with the idea of taking into account the nature of B-spline basic functions. It can be thought of as if parameterization predicts the behavior of the B-spline, thus better shape can be created. Besides that, usage of the fixed knot vector that is independent of data points makes the algorithm faster than other parameterization schemes. The method works well for curves of degree less than four, which is common to have in practice. Goodman, Ong and Sampoli [23, 24] have developed a nearly unit-speed parameterization technique that sequentially interpolates a curve, segment by segment. Later, the curve is optimized using the ratio between control polygon length and the length of data segments as the optimization criteria.

Park, Kim and Lee [25] have developed the method that makes two curves compatible by bringing them to the same reduced knot vector with a better parameterization. After that, curves are refitted based on energy minimization. The technique has been successfully applied to surface modeling, resulting in a better quality of constructed surfaces. However, the method is more computationally intensive.

Shape Modification. It has been suggested that while many commercial CAD programs use control point shape modification, it would be more intuitive to control the shape directly, by changing any point on a curve. For example, a direct shape modification approach proposed by Ishida [26] can be credited. The proposed method is also extendable to the surface shape control.

Juhasz and Hoffmann [27, 28] studied the effect of knot values on the shape of a cubic B-spline. Three methods of local shape modifications of curves subjected to position and tangent constraints have been proposed. As opposed to control point repositioning, these approaches only allow knot value modification, which leads to preserving the convex hull of a control polygon. However, users don't have to deal with knot value modifications directly, simple specifying of the geometric constraints would be sufficient.

Curve Approximation. Renner and Weib [29] have presented two algorithms for approximate computation of B-splines on a surface. The emphasis has been put on the practical efficiency of the proposed algorithms. The first approach is based on an exact representation, but since exact curve has higher degree, the simplified curve is created by degree reduction. The second approach is based on the classic least-square method incorporating surface geometry information. Another algorithm for approximation of a curve on a B-spline surface has been presented by Yang et al. [30]. In this method a given curve is approximated by the polyline, which is later subdivided to achieve certain user-defined tolerance. This approach is different from the other approximation algorithms in a way that the curve completely lays on the corresponding B-spline surface.

Point Inversion. Valuable work on point projection and inversion for NURBS curves and surfaces has been done by Ma and Hewitt [31]. The main idea of the method is to find a good candidate point for further Newton-Raphson solution approximation. Afterward, the surface of interest is subdivided into a set of patches, and the candidate point is picked as the closest vertex of one of the patches. Even though Chen et al. [32]

have shown the counterexample for the above algorithm, the method still works well in most cases and can be efficiently used in practice.

2.2.2 Surface Modeling

Surface Interpolation. Nasri [33] has described a novel method for generating interpolation surfaces from a set of curves by means of the control network subdivision, specifically, subdivisions of the network converge to the limit surface which is the interpolation of curves. It has been noticed that in the case of regular networks this algorithm is identical to the knot insertion, and therefore the same result can be achieved. Farin and Hansford [34] have constructed a new formulation for the discrete Coons patch. In their work the Coons patch is defined as the solution of a linear system. The method produces patches with more desirable shape than regular Coons surfaces. Work [35] presents the new approach for surface interpolation, called NURBS boundary Gregory patch, which is an extension to the standard Gregory patch. The main advantage of the method is that it permits constructing smooth NURBS surfaces over irregular curve meshes. Moreover, the shape of a surface can be modified locally, and continuity between patches is satisfied automatically.

An interesting work, from the point of view of conceptual design, has been presented by Qin and Wright [36]. The authors described a novel scheme for surface design using sketch-based modeling. The base surface is constructed as the standard Coons patch. Progressively, with adding each new internal characteristic curve, the surface is updated to incorporate new features. Finally, surface fairing is achieved by means of reparameterization. This method is very "user-friendly," since it does not

demand accurate curve definition, and as a result, conceptual models can be created very quickly.

Shape Modification. Free-form surface modification has been investigated in the work [37]. Leon and Trompette presented an approach for the freeform surface control by simultaneous modification of more than one control point. This approach gives a new shape modification tool that in some cases can be more preferable for a user, and increases computational performance for high quality surface modeling. In the work [38] Hu et al. have presented a new approach for NURBS surface modification. While the standard NURBS control can be performed by direct repositioning of control points and changing weights, the contribution of this research is the shape control using point, curve and normal vector constraints. The proposed method, which is based on strain energy minimization, indirectly changes control points in a way that guarantees satisfaction of given constraints.

Features on a surface are important for surface construction, especially for the vehicle design process, when it is preferable to stretch a surface without making any changes to the existing cutouts. Zhang et al. [39] have developed a new approach for shape scaling while keeping trimmed features unchanged. According to the authors, the method works well up to a scaling factor of two.

Surface Blending and Decomposition. First order continuity of free-form patches is a quite important topic in CAD design. As new approaches for surface modeling are developed, tools for smooth blending of these surfaces are a necessity. In general, only quadratic and cubic NURBS are blended to have G^1 continuity. Though, Che et al. [40] came up with necessary and sufficient conditions for connecting NURBS surfaces having

arbitrary degrees and knot vectors. A corresponding algorithm for adjacent G^1 NURBS surfaces construction is given in his work. In the work [41] Konno et al. have considered the problem of continuous connection of patches interpolated from a net of curves using NURBS boundary Gregory patches. Adjacent boundaries of each quad of a mesh are brought to the same knot vector without changing the shape. Afterward, each quad is interpolated using NURBS boundary Gregory patch method, producing a smooth surface over the whole net of curves.

Hui et al. [42] have presented a new feature-based approach to trimmed surface decomposition into a set of B-spline patches. All trims on a surface are isolated by means of Voronoi diagram. Next, the algorithm finds optimal connections between Voronoi region curves and trimming curves, thus creating more regular patches. This scheme can be helpful for constructing very complicated trimmed surfaces.

2.2.3 Surface Rendering

Trimmed surface tessellation can be very time-consuming, especially for very complicated shapes. To speed up the triangulation of surfaces, methods based on trimmed curves tracing have been developed [43, 44]. For example, Cripps et al. [43] have proposed the tessellation scheme that works only on active cells of the rectangular grid. Specifically, the algorithm traces a trimmed curve on a surface, detecting all cells that are intersected by the loop. Hence, further tessellation operates only on a small list of detected cells. Hamann and Tsai [45] have proposed a tessellation scheme based on surface decomposition. Using the Voronoi diagram a surface is decomposed into a set of patches. Each patch is a ruled surface, two boundaries of which are horizontal segments, and other two are the trimming curve and the patch boundary curves. After the

decomposition it is trivial to triangulate each of the patches composing the surface. In [46] Piegl and Tiller, to catch the curvature of a surface, subdivided it based on a user-defined tolerance. Rectangles intersecting trimming polygons were merged with segments of those polygons.

A few tessellation methods have been developed to efficiently handle shape modifications. A new efficient tessellation scheme for deformable trimmed surfaces has been proposed in the work [47]. The authors constructed a special data structure that updates only actively deformed regions on a surface. Ng and Tan [48] subdivide a surface with some tolerance, after which they split each newly created region into two triangles. Finally, the trimming curve is converted to a polygon and inserted in the mesh using an edge recovery algorithm, which can be found in some unstructured meshing techniques. The data structure utilized in this work permits rapid modification of the shape, as well as straightforward removal or addition of trimming curves onto a surface.

Generally speaking, tessellation can be done using any algorithm for unstructured meshing, as is shown in [49]. However, unstructured meshing algorithms designed to produce high quality triangulations are excessively slow for most applications, whereas for rendering purposes, low quality triangulation is acceptable.

2.3 Structural FEA Meshing Algorithms

2.3.1 Curve Discretization

There are relatively few papers on density-based curve discretization. One of the earliest attempts to solve the problem of smooth node distribution was presented in the work of Gutierrez et al. [50], where node positions are defined by iteratively solving a system of linear equations. Cunha et al. [51] proposed an algorithm called "smart sizing".

In general, the algorithm is not capable of accurate node distribution with respect to a given density function. Particularly, nodes are generated based on ends' sizing values, therefore ignoring any changes of size density internal to the segment. An advantage of the algorithm is that it takes into account feature proximity, creating good quality edges near sharp corners and trims.

Cuilliere [52] has developed an elegant scheme for 3D parametric curve discretization. Contrary to earlier proposed iterative schemes, Cuilliere's method is direct, and works well for both 3D curves in space and curves on parametric space surfaces. The number of segments to be created is estimated by integrating the inverse of a density function over the curve length. Later, Wu and Wang [53] have improved the Cuilliere's scheme by using the Newton-Raphson method to solve for unknown parametric values for nodes, which made the scheme much faster.

2.3.2 Surface Meshing

Meshing algorithms usually are divided in two categories: structured and unstructured. Currently the most popular structured methods are transfinite interpolation and elliptic smoothing. Transfinite interpolation also sometimes serves as an initial solution for further elliptic grid generation. These methods work well for simple four-sided untrimmed patches with nearly flat shapes. Basic descriptions of approaches for different types of structured grid generation methods can be found in [54, 55, and 56].

Advancing Front. Over 10 years ago Tristano et al. [57] presented an innovative scheme for advancing front surface triangulation. The method utilizes a metric map that estimates the distortion of a surface in the parametric space, hence permitting triangulation of a surface in parametric space. Triangles can be very distorted in

parametric space, but they present a good-quality on the surface. The algorithm shows significant improvement in run time compared to early direct methods that work in 3D space. The method presented in [53] progresses in the opposite direction, starting from the opposite longest boundaries. Corner element generation has been improved by deciding how many elements to generate. The drawback of the algorithm is that it generates elements over the entire surface, regardless of trimmed regions. It can significantly decrease the run time, and the elements created on the edges of trimmed curves can be of poor shape, even though it is usually expected to have well shaped elements near the trimmed regions, considering that these are often regions of stress concentration.

Miranda and Martha [58] have presented a procedure and data structure for storing surface metric information. Instead of the widely used Delaunay background mesh, the authors have proposed to store the local surface metric in a quadtree data structure. Specifically, a background quadtree is created by recursive subdivision of parametric space. Each cell is refined until a specified tolerance will be more than the local surface curvature. In order to avoid large differences in the degree of subdivision between adjacent cells, the algorithm refines them to guarantee a smooth transition. Finally, the method proceeds like the classic advancing front with a metric map analogous to the one defined by Tristano et al. [57].

Quadrilateral Meshing. One of the most popular indirect methods for generating quad only meshes has been proposed by Owen et al. [59]. A quad mesh is generated on top of a triangular mesh by successive triangle transformations into well-shaped quad elements. The method (usually called Q-Morph) gained its popularity thanks to well-

established approaches for smooth triangular meshing. Q-Morph uses advancing front to keep track of active edges and to facilitate smooth transition between newly created quad elements. A few techniques in Q-Morph are borrowed from the paving method, which is another popular but older method [60]. A progress update on paving and its newer version, unconstrained paving, can be found in work by Staten et al. [61].

Maza, Noel, and Leon have proposed a method similar to Q-Morph [62] that use a frontal approach for quad generation on a triangular mesh. To improve overall mesh quality the authors have used a few topological operations, such as edge swapping, element removal, element creation, and a special scheme for treatment of corners and boundaries. Even though topological operators are applied only after the whole quad mesh is generated, the resulting mesh has good quality.

A new method capable of producing quadrilateral meshes with elements aligned along the principal direction of an underlying surface has been developed in [63]. For the alignment process an iterative relaxation scheme has been used. After that, quads are generated by simply discarding non-aligned edges of triangles.

New results have been presented for 2D unstructured grid generation. Sarrate and Huerta [64] presented a scheme that recursively decomposes a domain until a desired element size is reached. A special objective function that includes five criteria for numerical description of the splitting line positioning has been developed. The optimal split line is found by minimizing the objective function on a given domain. Bastian and Li [65] have proposed improvements for the recursive method by considering possible misshaped elements and describing the solutions for each of them. Furthermore, improvement for the algorithmic implementation has been presented.

High-quality mesh generation can be very time consuming for large applications, thus, block decomposition schemes can be of particular interest. For example, in the work [66] the authors have developed a method that decomposes a domain in the fine-shaped regions based on a size map and geometry. After that, each region can be meshed using an arbitrary structured scheme.

Other Meshing Methods. Delaunay triangulation is an example of a popular meshing scheme that falls into the category of unstructured methods [67]. Though the original idea goes back to the works of Dirichlet, Voronoi and Delaunay, its application to the meshing problem was explored only 30 years ago, when a few point insertion algorithms were developed.

Another interesting approach is advancing front circle packing. It has been noticed that circle packing mimics the Voronoi region [68], as a result, the method is somewhat similar to Delaunay approach. Unlike a standard frontal scheme, the algorithm starts from some convenient internal point, hence both bounded and unbounded meshes can be produced [69, 70]. Circle packing works by generating nodes in the centers of tightly placed circles. Element sizing is controlled by changing the radius of a circle.

Meshing with Constraints. Constraint solving for mesh generation is a very important problem. Often complex models contain point and line constraints. Those can be points or lines where load is applied, or lines of a surface connection to other geometric features (for example, surface-to-surface connections).

Park et al. [71] developed a paving-based method capable of taking into account random line constraints. Since the paving is frontal method, the main idea is to generate a loop of segments from a given line constraint. After that, the algorithm works in a way

similar to a regular direct quadrilateral mesh generator, however additional operations are introduced. These new procedures deal with special cases of line constraints, e.g., intersecting lines that form unmeshed hole, crossing lines etc. Alternatively, Lee et al. [72] tried to solve an analogical problem using an indirect method instead. Unfortunately, their algorithm is not generalized to work with intersecting lines.

A new approach for point constraints solution on transfinite meshes has been investigated in the paper [73]. Older methods usually snap the nearest nodes to the constraint points, therefore, poorly-shaped elements can be generated. Even though smoothing techniques can enhance the quality of such meshes, elements quality may still not be satisfactory. The author has proposed an inverse solution to the problem. Having the set of points it is possible to solve for corresponding boundary nodes. It can be easily done with any algorithm for inversing a point on a surface. As a result, the meshing lines will pass through these constraint points, and all generated elements will have a rectangular shape. The drawback of the scheme is that elements with a bad aspect ratio can be created.

Parallel Meshing. Meshing processes can be very time consuming for large models with many elements, thus much work has been devoted to the problem of parallel grid generation [74, 75, 76, 77]. Generally speaking, these methods are based on sub-domain meshing, where a surface is decomposed by adopting some domain decomposition algorithm. Finally, all sub-domains are independent and can be meshed in parallel.

2.3.3 Mesh Smoothing and Optimization

Laplacian mesh relaxation has been the most common mesh optimization technique for over 20 years. Despite being very simple in implementation, the scheme is a powerful

tool for mesh smoothing and produces satisfactory quality meshes. Another family of smoothing methods is optimization-based methods. The main idea is to create a function that represents the distortion of an element. By minimizing the distortion function, the optimal mesh quality is achieved. Canann et al. [78] have suggested the combined approach of Laplacian and optimization-based smoothing. The method utilizes the best features of both schemes and can be applied to triangular, quadrilateral and mixed meshes. Xu and Newman [79] have described a new optimization approach to the torsion spring-based smoothing method [80]. Their approach minimizes the energy of the torsion spring system, specifically, by using the Gauss-Newton optimization scheme. As the authors pointed out, the algorithm permits better optimization for new positions of internal nodes. Therefore, the method is preferred over the standard approach described by Zhou-Shimada in [80]. Most methods that are based on the Laplacian scheme do not preserve the original shape of a surface. After a large number of iterations a surface can be significantly shrank and distorted. Garimella et al. [81] have tried to solve this problem. They proposed a smoothing technique that keeps nodes as close as possible to the original shape, but still allows some deviation. Another method that preserves the original shape of a surface is based on solving a nonlinear fourth order PDE [82]. In short, the solution of this PDE also lies on the original surface, thus the method does not affect the resulting surface shape. Liu et al. [83] have developed a global mesh optimization approach based on the Laplacian technique. The main difference from other Laplacian-based methods is that the proposed algorithm is non-iterative. The problem is formulated as quadratic optimization, and is solved by sparse linear system. Likewise, the given algorithm does not cause surface distortion.

2.4 Rigid Body Response for Vehicle Performance Analysis

2.4.1 Numerical Integration

Potra et al. have presented new linearized trapezoidal scheme [84] for multibody dynamic simulation. Numerical results demonstrated that the method is stable to stiff springs, and can be used with arbitrary time step length. Some authors prefer implicit integration schemes [85, 86, and 87], especially for real-time animation of a complex model with stiff springs. Acceleration based simulators sometimes use a Newmark family algorithm as described in [88, 89, 90]. Stewart and Trinkle in [91] and Anitescu et al. in [92] have developed an implicit scheme for their MLCP formulation. In practice, implementation of an implicit integration scheme can be somewhat complicated. Particularly, all terms should be estimated at the next time step, and it is often done using Tailor expansion formula. Though, derivative estimation can be tricky sometimes. For low cost computers it is preferable to use explicit methods, especially when stiffness is not an issue [93]. Guendelman et al. [94] used simple Euler integration, but improved the sequence for different algorithms processing. Their approach allows clear separation of contacts from collisions without the need for threshold velocity. The method has been successfully validated with the standard theoretical solution for a block sliding on an inclined surface.

It is almost impossible to discern between physically-based and numerical energy dissipation. For accurate multibody simulations it can be of interest to have a method that is capable of energy preservation. In [95] Uhlar and Betsch have introduced the energy momentum consistent time integration scheme. Their approach relies on a

thermodynamically consistent modeling of dissipation. As a result, time stepping preserves total energy balance and completely excludes numerical dissipation.

2.4.2 Error Correction

The penalty-based method probably is the oldest method in the field, and some applications still use it to recover from penetration [96, 97]. When running a simulation with a large time step and without continuous collision detection, significant penetration can take place. The penalty-based method compensates a penetration by means of a virtual spring between the body and the point of penetration. Though this method is easy to implement, it produces some undesirable behavior like jittering, and therefore is not appropriate for physically accurate simulation. Even though better methods exist, Baumgarte stabilization [98] is the most popular so far. The disadvantage of the Baumgarte approach is that it demands special coefficient tweaking, and this coefficient can be different for different systems. Many new methods have been proposed over the last decade. Other error correction methods, as well as review of other popular methods and comparison with Baumgarte, can be found in [99, 100, 101]. Cline and Pai [102] have proposed a new post-stabilization scheme for drift elimination. In their scheme, constraint stabilization and dynamics simulation are unified in the single framework which simplifies the design of software. An advantage of this approach over the Baumgarte stabilization is that it doesn't have any parameters to tune. Muller et al. [103, 104] worked directly with position of rigid body. They skipped the steps with velocity or acceleration calculation when solving constraints, thus avoiding the problems with constraints stabilization. In their formulation, position correction is estimated using first order Taylor expansion.

2.4.3 Rigid Body Paradigms

There are a few different approaches to the rigid body simulations. The earliest formulations were developed using an acceleration-based approach in the works of Pfeiffer et al. [105, 106, 107, 108, 109], Featherstone [110], Baraff [111, 112], and other [113]. In an acceleration-based approach all equations are written in terms of forces and accelerations. The main drawback of this method is that constraints are usually imposed on the position of a body, and not on the acceleration. Particularly, revolute, spherical, prismatic joints and contacts are constraints imposed on positions. Converting position constraints to acceleration constraints is computationally expensive and complex, often producing large equations.

An alternative to the acceleration approach is the use of velocity-based equations. In the velocity-based approach equations are formulated in terms of impulses and velocities. The approach is free of the above described problems native to acceleration-based methods, nevertheless, it has a problem with constraint "drifting". Position constraint equations can be easily converted to the velocity constraint equivalent. This allows satisfying velocity constraints exactly, but position constraints cannot be respected exactly due to the numerical errors. A few error correction methods proposed to overcome this problem with velocity-based approaches will be discussed in the next chapter.

Two families of the velocity-based algorithms can be outlined: constraint-based and impulse-based. Constraint-based methods have been developed in the works of Potra, Anitescu, Trinkle et al. [84, 92, 114, and 115], Sauer, Schomer [116], Kaufman et al. [117, 118]. An impulse-based method has been proposed by Mirtich [119] and later

elaborated by other authors [120, 121, 122, 123, 124, 125, 126, 127, and 94]. At present, most popular free engines like Bullet [128], ODE [129] and 2D engine Box2D [130] use a velocity-based approach.

Some researchers have formulated the equations of motion in terms of displacements. This approach is called position-based, and can be found in the works of Muller et al. [103, 104]. Furthermore, many efforts have been done to combine rigid-body and flexible-body approaches, as in the works [131, 132, and 133].

2.4.4 MLCP Formulation and Solution Methods

MLCP formulations have been presented in [114, 91, and 116]. Using this approach, one doesn't have to detect the specific time of impact. Instead, contacts can be solved using the notations of active and potentially active contacts, and by adding auxiliary complimentary conditions to the MLCP system. The frictional case is handled by approximating the friction cone with a pyramid having an arbitrary number of facets. It has been proven that such a MLCP problem has a solution and can be solved by means of the Lemke algorithm [134, 135]. Tasora [136] has developed a fixed-point iterative method for solving large multibody systems with contacts and frictions. The method works like fixed-point Banach contractive mapping. Later Tasora and Anitescu [137] used fixed point iteration to solve a nonlinear complementary problem. According to the authors, algorithm has O(n) complexity, therefore it makes the scheme very attractive for real-time simulation of large systems.

Projected Gauss-Seidel is another iterative solver that is successfully used in rigid-body simulations [138, 139, and 140]. PGS has a couple advantages comparing to exact Lemke algorithm. Specifically, accuracy can be controlled by setting a number of

iterations, plus it is simpler to understand and implement. Substantial improvement can be attained when using a constraint reaction caching method, also known as "warmstarting" [138]. Finally, PGS can be implemented in a matrix-free fashion as it was described by Catto [141]. In this formulation, PGS is called Sequential Impulse solver. Constraint modeling procedures are thoroughly described in the works of Erleben [142], Garstenauer [143] and Sabana [144].

2.5 Fatigue Analysis for Vehicle Structural Assemblies

As it has been demonstrated by Wannenburg et al. [145], reasonablely accurate vehicle structures lifetime prediction can be achieved by using simple fatigue calculations based on stress-life instead of strain-life approach, and also by using quasi-static FEA instead of dynamic FEA. In this work, two case studies on structures of heavy vehicles have been performed. In both cases fatigue loads were estimated from quasi-static G-loads applied to the models, and static FE analysis has been run to obtain stresses. Zoroufi in his dissertation [146] has done an extensive study of fatigue design and optimization process of automotive components, and showed how it is affected by manufacturing process. Fatigue analysis for wide range of vehicle components has been performed using experimental, numerical and analytical tools. Fatigue behaviors of different materials have been compared. The author pointed out that even at the lower loading level the material undergoes local plastic deformation. Thus use of linear elastic FEA may not be sufficient for reliable fatigue life predictions. Boessio et al. [147] investigated vehicle structures under the random load due to the effect of rough pavement surfaces. Displacements and corresponding stresses in structural members can be defined using dynamic analysis, and lifetime can be estimated by means of the classic stress-life

approach. More comprehensive review of FEA-based fatigue analysis techniques can be found in the work of Abdullah et al. [148].

Several attempts have been made to develop fatigue software [149, 150, 151]. Less recently developed fatigue tool [152], and implemented in commercial software, consists of three component libraries, namely, material library, fatigue engine and damage engine. The goal of their work has been to create a general purpose fatigue tool, which is easily extendable for specific applications.

CHAPTER 3

ARCHITECTURE-LEVEL VEHICLE MODELS AND ANALYSES

3.1 Utility of an Integrated Analytical Approach to Architecture Design

Functional performance goals can be met with a significant decrease in product development time implementing a conceptual design phase. NVH parameters can be effectively estimated during the conceptual phase. In order to obtain a satisfactory design, which leads to shorter design-time and reduced costs, one can utilize analyses early in the development cycle. The early stage of vehicle development is of particular importance, since the designer has the possibility of controlling both the vehicle geometry and performance characteristics. At that early stage the information on the vehicle architecture is not yet complete. Therefore it is possible to make decisions on alternative vehicle concepts. Although, a good concept is not a sufficient condition to establish a successful product.

The full potential of utilizing CAE during the conceptual design stage has not yet been completely investigated in industry. It is difficult but possible to derive appropriate functional and geometric simplifications, which still correlate well with detailed-level functional characteristics. Preliminary analyses allow a designer to compare more alternative architectures, thereby increasing the likelihood of successfully creating a product. An appropriate tool for conceptual architecture modeling and optimization could drastically reduce design-time for the engineer and permit a more simulation-driven development process.

Historically, vehicle design relied primarily on engineers' intuition and experience. New products were derived with gradual changes, as a result, permitting a good understanding of the design, based on the predecessor products. On the other hand, nowadays, there is a vast choice of new materials, including different metal alloys, plastics, and composite materials. Additionally, options are expanding, as a variety of suspension types and hybrid powertrains have been developed. This abundance of distinct functional components and materials for structural components provides greater flexibility for engineers, but limits their intuitive approach to the problem. As the market rapidly grows and new technologies are created, design engineers encounter the problem of meeting all functional standards are sometimes conflicting, or even contradictory. The number of design constraints imposed on a designer has risen significantly over the last few decades.

Therefore, the concept design phase is essential when it is particularly crucial to reduce the amount of time required for vehicle development. For instance, sometimes during the long time spent on detail design optimization, new requirements arise from changes in the market and among competitors. This often leads to expensive design changes to accommodate the new requirements. By delegating the portion of the optimization process to the conceptual design phase, potential flaws of the new product can be revealed early in the development progress, thus reducing the overall design time.

While architecture models have been recently discussed in the literature, and "reduced order" finite element models have been used in vehicle design to shorten analysis cycles, there has been no effort to systematically describe an appropriate abstraction level for vehicle architecture models that function as a design space and as a

starting point for deriving analysis models. Nobody has proposed a practical approach to representing a full-vehicle models at the architecture level, or for "designing a concept" in the similar sense to that used for geometric design. While a stand-alone architecture abstraction might be useful from visualization or comprehension perspective, their true power is tapped only when they are used as the basis for a broad range of analyses supporting conceptual and architecture-level design.

3.2 Analyses Needed for Architecture Optimization

It is vital to conduct all essential analyses during the concept phase. Defects in the concept vehicle architecture lead to poor results in the detailed design phase, increasing the overall vehicle design time due to the increased difficulty and time in modifying the much more complex detailed model geometry. Instead, optimal design parameters can be estimated during the conceptual design phase. The vehicle concept design phase follows the same process as the detailed phase: architecture design, model discretization, analysis and optimization. During the course of developing a vehicle model, different analyses models may be abstracted from the concept model based on the current level of specification. The following analyses are appropriate during the conceptual stage:

- Spatial, kinematic, inertia
- Rigid body dynamics
- Powertrain performance
- Finite element analysis
- Fatigue analysis
- Crashworthiness
- Static stability and rollover analysis

Material selection can be accomplished as the optimization process with a set of weight constraints. Moreover, advanced material selection optimization can be

established by taking into account structural stiffness, modal characteristics, crashworthiness, reliability, etc.

3.2.1 Spatial, Kinematic, Inertia

This is the simplest set of analyses needed to estimate basic characteristics of a proposed concept. Spatial characteristics include gross dimensions, assembly compartment volumes, as well as clearance parameters. Spatial evaluation may begin as soon as geometry of a vehicle or an individual assembly is defined. Usually, for calculation of gross dimensions, only the points determining the geometry are sufficient. Though, in the case of more advanced scenario, where the geometry is defined with B-spline curves and surfaces, corresponding control points have to be included in the spatial analysis. Assembly compartment volumes can be estimated when enclosures are defined in the concept model. Clearance, as part of an ergonomic analysis, provides a designer with information on a crew member's head room, leg room, and moreover, distances from the specified crew member to the specified structural component or sub-assembly. In addition, wheel base, track width, and ground clearances can be evaluated based on the vehicle assembly abstraction.

As soon as closures are defined in an architecture model, kinematic analysis can help to inspect the opening angles, particularly for doors. A designer can specify an angle and inspect if no interference occurs. Similarly, repositioning a crew member body, by adjusting the angles between its body parts, or changing seat orientation, can be useful for predicting ergonomic characteristics of a concept vehicle.

Inertial analysis incorporates estimation of the standard parameters: mass, center of mass, and moments of inertia. When geometric features, like paths, surfaces, and

volumes are designated as structural components the material properties must be assigned to estimate inertial properties of components. Inertial properties of the whole vehicle can be estimated based on the inertial properties of each assembly in the vehicle.

3.2.2 Rigid Body Dynamics

These analyses may incorporate response to standardized inputs, terrain response, and frequency domain response. Once the inertial properties and the assembly connectivity are defined in terms of the connection compliance and DOF, rigid body assessment may be performed by the engineer. For the simplicity of the vehicle concept model, it can be assumed that all beam and panel components within a structural assembly are treated as rigid components, therefore ignoring any flexibility in the vehicle. Usually it is convenient to store these models in an undirected graph data structure, with each node of the graph representing a rigid assembly, and edges representing assembly connections. When two rigid components are connected by a rigidly connected joint, the components can be lumped together forming a single composite body, which significantly reduces the solution cycle time for the numerical simulation. The algorithm loops through all rigid edges in the graph and combines bodies until no rigid edges remain.

3.2.3 Powertrain Performance

Powertrain analysis includes minimum braking distance, maximum acceleration, and fuel efficiency for a vehicle. To perform these calculations a designer should specify the powertrain energy storage, power source, and power transmission assemblies and the energy and power transmission connectivity for the vehicle. Additionally, these types of analyses require determining normal forces at the wheels based on the road grade,

aerodynamic drag, and external loads applied to the vehicle. These forces can be estimated from the static analysis of a vehicle as a multi-body system. Therefore, the reasonably accurate estimation of inertia of each vehicle assembly is required to obtain accurate powertrain performance parameters.

Fuel efficiency analysis assists a design engineer in predicting the fuel consumption for the vehicle that operates at a given speed and a length of time. Many innovative analytically-based methods for calculating fuel consumption of parallel and series hybrid electric vehicles (HEVs) have been proposed. These analytical methods are usually applicable for the concept-level analysis and feature highly accurate fuel consumption results. The input data of the analysis are energy flows and characteristic efficiencies that are derived from the energy flows on selected energy paths of HEVs. Based on these powertrain component's abstractions, vehicle fuel consumption can be obtained.

To avoid collision, the braking performance of vehicles is critical. Therefore, the vehicle components and assemblies influencing the braking performance should be appropriately designed to meet the predetermined braking constraints. For a given test condition, braking performance assessment requires the specification of the vehicle brake and wheel abstractions, inertial model abstractions and power transmission paths to the wheels for each braked wheel. Based on the vehicle abstraction, braking performance for a specified vehicle speed, road grade, and friction coefficient related to road finish can be determined. However, in the context of the preset braking distance regulations, the method based upon the elementary vehicle architecture may not be applicable because it cannot accurately predict the effects of these abstract components. Thus, for the precise analysis a designer should use a more detailed brake model.

Vehicle peak acceleration calculations can be estimated for different hybrid powertrain configurations and road conditions. Torque at the drive wheels is a function of the overall powertrain configuration. It requires powertrain component specifications beginning with the power source and ending at the driven wheels, and the power transmission paths between them.

3.2.4 Finite Element Analysis

At the conceptual design stage, FEA includes static analysis, modal analysis, and strength analysis. Moreover, knowing the endurance characteristics of the material model, fatigue analysis can be performed.

For the FE analysis, accurate representation of non-structural components is unnecessary. Instead, components like powertrain assemblies, fuel tank, power devices, etc., may be represented by simple inertial components with appropriate model connectivity. These inertia components contribute to the load through the assembly connections, which are depicted as key points in the FE model. If any component contains multiple key points at approximately the same location, then the points must be merged. Any rigid portions of a connection are represented as rigid elements, while spring and dampers are modeled with appropriate spring and damper elements.

Vehicle meshing advances at the vehicle level one assembly at a time, and discretize each component within an assembly one at a time. During this process a meshing algorithm tracks node dependencies due to the connections at the various hierarchical levels between assemblies.

3.2.5 Fatigue Analysis

The fatigue engine is an auxiliary tool for the vehicle assembly structural design and based on FEA. Fatigue factor of safety or reliability serves as a constraint during the optimization process. Early estimation of fatigue reliability can help in locating the problematic regions in the assembly before proceeding with the detailed level analysis. Besides that, comparative analysis between different assembly configurations can be useful when making the decision about an appropriate design.

The hierarchical organization of the vehicle architecture abstraction contains sufficient geometric and material information to enable fatigue analyses appropriate for the conceptual design phase. Since fatigue analysis depends only on the FEA, as a result, its level of abstraction is the same as for the FEA.

3.2.6 Crashworthiness

Crashworthiness design is a very important in the automotive industry due to the regulations form the Department of Transportation. Safety of the occupants and the structure's ability to protect them during an impact is considered as one of the primary performance characteristics during the vehicle design. Many approaches for optimal body-structure design and crash analysis, as well as different criteria, including passenger injury probability and deformation of the vehicle structure, are usually utilized while estimating the crashworthiness of a vehicle. Though, analyzing occurs only when very complex finite element models are already created at the detail level. Since, crashworthiness deficiencies are usually introduced early in the vehicle development, it is time-consuming to correct them in the later detailed design phase. This usually leads to expensive optimization procedures, while trying to satisfy all crashworthiness constraints.

In order to meet the crashworthiness requirements, the corresponding analysis must be conducted during the conceptual design phase. Higher abstraction conceptual models can suffice for preliminary structural optimization relevant to crashworthiness. While these conceptual models do not replace detailed models, they can significantly reduce the time required for detailed level optimization as the vehicle architectural layout was already optimized.

Crashworthiness analysis relies on finite element analysis. In particular, non-linear material properties have to be specified. Besides that, the behavior of structures is usually characterized by materials and joints failure models and other related material attributes. Finally, in order to obtain accurate response of structures under crashes, proper inertial forces and contacts must be considered.

3.2.7 Rollover Analysis

Rollover is one of the most dangerous maneuvers for the vehicle crew members. The most common reasons for rollover to occur are driving too fast while turning, traversing a steep slope, collision with another vehicle, and tripping. At the concept phase of the vehicle design only the first two cases can be reasonably estimated in order to establish the stability of the concept vehicle. Both cases can be analyzed using quasi-static rollover of a rigid vehicle or suspended vehicle. Otherwise, with some additional computational time, it is possible to use a multi-body system dynamics approach using a simple terrain.

For the simplest analytical approach that assumes the vehicle is a rigid body, only the center of mass and tread are required, which can be determined from the vehicle model. For the method involving vehicle dynamic simulation, the vehicle abstraction

should be at a level equivalent to that of rigid body analysis. Specifically, assembly inertial properties and assembly connection DOF should be derivable from the vehicle model.

3.3 Feature Abstraction

All components comprising a vehicle, in general can be grouped into two categories: structural and non-structural features. Structural components are designed to carry load in the vehicle, while the non-structural contribute to the load but are considered with other analyses, for example, powertrain analysis, rigid body analysis and others. Description of each of these component types follows below.

3.3.1 Structural Features

Many of the primary structural members in a vehicle body are beam structures. Beam components are designed to carry combinations of forces, bending moments, torsional moments and experience compressive, tensile and shear stresses. Beams are characterized by their profile, length and material properties. Beam geometry can be defined in terms of a curve path type, geometric control points defining the shape, and cross sectional shapes along the curve. In the most general case the beam path is represented by a parametric equation, for example B-spline curve. Additionally, the locations of cross sectional changes should be specified in terms of the parametric coordinates.

Panels are steel sheet metal stampings, and may be of fairly complex geometry. Panels usually are spot-welded together to form an integrated structure. In the most general case panel can be represented as a composition of parametric surfaces. Trimmed parametric B-spline surface is an optimal choice. Cutouts on a surface can be created

using parametric B-spline curves in the parametric domain of the surface. Besides geometric representation of the shape, panel abstraction should include thickness, number of layers, in case of composite material, and material properties.

Connection between beam components in a vehicle body structure is presented as major compliance joint (MCJ) elements. Characteristics of MCJ's are defined by local sheet metal shape and thickness.

The assembly of the panel stampings in to the vehicle body is made mainly with spot welds: a few thousand are needed. Other connection types are also used, such as arc welds, laser welds, clinches, self-piercing rivets, bolts, and glue. The choice of connection method depends on numerous often interrelated criteria such as cost, visual appearance, accessibility, corrosion, stiffness, fatigue and strength. These assemblage joints are classified into three groups: point, path, and surface based connections. Point based connections are modeled by rigid connections at the specified parametric locations on the components. Abstractions for path based connections include weld pitch spacing and parametric range along the connection path. Finally, surface based connections are extended from the path based connections to a second dimension by adding additional weld pitch spacing in the orthogonal parametric coordinate.

3.3.2 Structural Assemblies

Structural assembly consists of rigid and compliant components, and is designed to carry the external loads. Individual panels and beams are joined through a variety of processes, such as welding, riveting, and bonding into assemblies like frames, cabs, doors, cargo structures, etc. Proper joint type selection and assembly designs procedure can improve vehicle structural behavior, optimize manufacturing time, and reduce cost.

Traditionally, assembly design is based on intuition and experience, lacking scientific and engineering argumentation. Therefore, controlling an assembly design during the concept stage is of particular important.

Structural assembly modeling begins with construction of an assembly from simple architecture features and defining spatial relations between them. These features can be beams, panel, inertial component, major compliance joints and assemblage joints. A representation of a structural assembly concept model must represent the architecture layout and connectivity between structural members included in the abstraction. Furthermore, selection of a single feature in the assembly must be available. Model representation must support changes in the relative spatial location of parts, and manipulation of the assembly as a whole.

An assembly connection between two assemblies in a vehicle concept model is represented by defining the connection model type, connection points and connection properties. Connection model types are represented by the standard mechanical joints - fixed, hinge, spherical, prismatic, spring, damper, and combination of these. Connection properties include the DOF, compliance, and damping characteristics associated with each type.

3.3.3 Powertrain Systems

There is huge attention for low emission and independence from the fossil fuel energy sources to decrease global warming on the world. As a result, many configurations of electric vehicles are designed to reduce the emission. These electric vehicles (EV) may include battery electric vehicles (BEV), hybrid electric vehicle (HEV) and hydrogen fuel cell electric vehicle (FCEV) and hydrogen fuel cell plug-in hybrid

vehicles (FCHEV) which is the combination of previous. These powertrain configurations are much more complicated as compared to the conventional internal combustion engine vehicles (ICEVs). The powertrain configuration of HEV can be divided into three types: Series, Parallel and combination HEV's (Figure 2, Figure 3 and Figure 4).

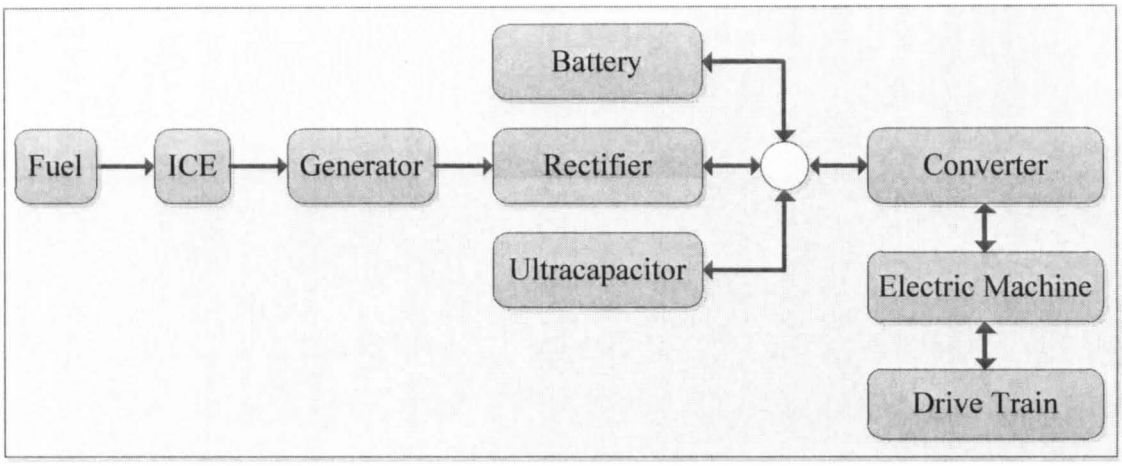


Figure 2. Power flow in a series hybrid powertrain.

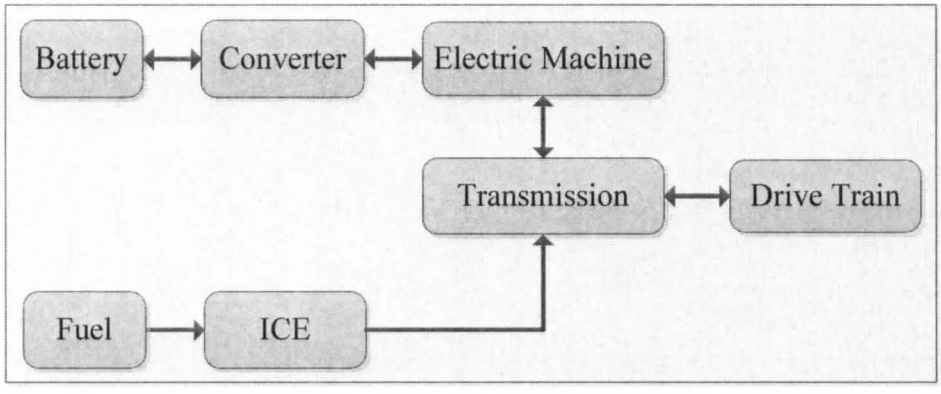


Figure 3. Power flow in a parallel hybrid powertrain.

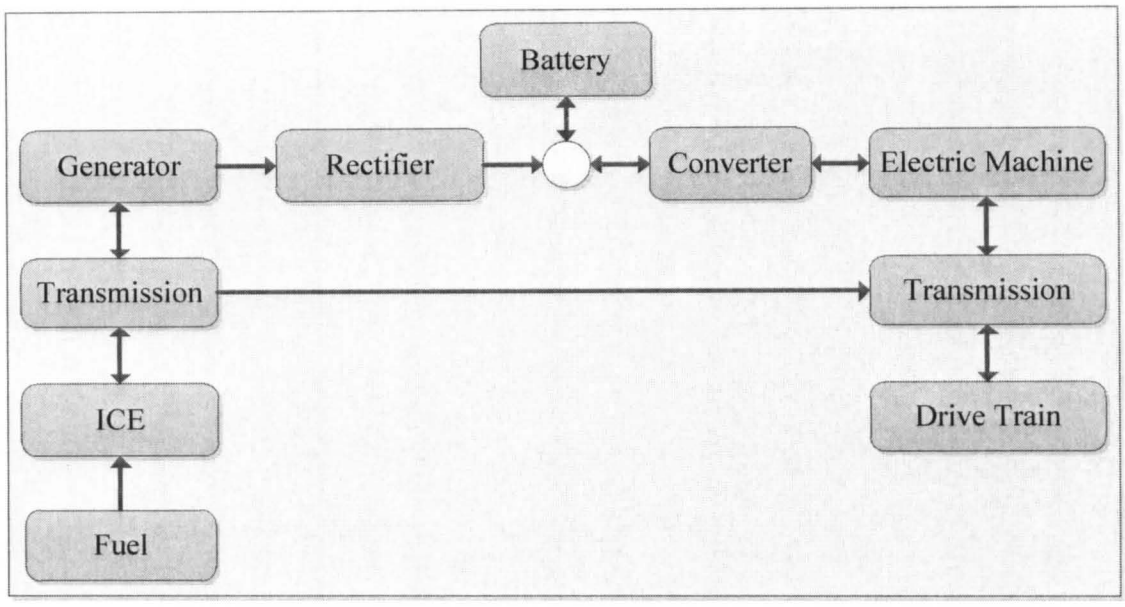


Figure 4. Combination of series and parallel hybrid configurations.

Hybrid powertrains usually may include internal combustion engine, generator, battery packs, rectifier, capacitors, converters and electric motors, etc. Abstraction level for these and other features can be characterized for the concept development stage. Proper abstraction of each of these components is a significant factor in designing the powertrain systems.

Energy storage assembly's abstractions may include fuel tanks and batteries packs. For fuel tank abstraction the basic geometric dimensions should be specified to estimate the capacity. Additionally, fuel type is required to calculate the energy that can be released. Similarly, for battery pack the energy required for the electric motor can be estimated. The necessary properties that should be specified are cell type, weight and size. In turn, cell type defines the characteristic properties such as specific energy, specific power, internal resistance, voltage, and capacity. Recently, ultracapacitors are often used as an assistant to batteries, taking advantage of their high power density, quick charging and discharging capacity, long life, and high efficiency.

Power source assemblies within a vehicle are represented by electric motor, engine and brakes. The electric motor of a hybrid electric powertrain can be defined using several parameters such as the motor size, battery capacity, and type of hybrid powertrain configuration. IC engines are defined by specifying geometric properties - bank angle, number of cylinders, cylinder bore, cylinder spacing, slant angle, crank radius, and piston height parameters.

Power transmitting assembly, including transmissions, differentials, and wheels, transmit power from one assembly to another connected by a power transmission path. Wheels are geometrically represented as a tire and rim. The parameters that need to be defined are geometric properties and tire type. This set of characteristics provides enough information for determining inertia properties, connectivity location and friction coefficient. For transmission, besides general geometric and inertial properties, performance parameters have to be specified. Depending on the transmission type, gear ratios and efficiency can be defined.

Connection path that defines the type of energy flow between components should be abstracted as well. For example, typical energy paths are generated for fuel, electric current flow and power transmission. Correct path type can be automatically determined and validated for the given powertrain configuration.

3.3.4 Miscellaneous Features

Built of structural assemblies, vehicle body also acts as a console, holding the engine, powertrain, interior panels, seats and closures together. These and many other components and assemblies that do not necessarily provide structural support for the vehicle architecture can be modeled as a nonstructural assembly. One of the functions of

these assemblies is to contribute to the inertial forces acting on the vehicle through the connectivity that they provide to the other assemblies in the vehicle. Thus, mass and moments of inertia for the assembly are the abstractions required to be define in the concept model. However, most assemblies can be used for the other types of analyses where more abstractions are needed, for example dimensions for clearance or volumetric calculation analyses. Besides, specialized abstractions required for powertrain performance have been described above.

3.4 Hierarchical Feature Organization

A description of the vehicle architecture follows here, with its relation to other systems in the vehicle. At the architecture/conceptual level, vehicle abstraction is of necessity fundamentally different than a detail-level abstraction. In general, these abstractions must include general shape and layout, major spatial features (nominal volumes, in particular), component and subsystem connections (which can be quite simple relative to a detailed CAD model), and inertia properties. They should describe critical subsystems, including, but not limited to primary body structure, suspensions and powertrain, energy storage and transfer elements, and the human operator and passengers.

Implementation of this list of analysis requirements mandates the use of preprocessing algorithms for determining gross dimensions, areas, and inertia properties (mass, mass centers, mass moments of inertia). Geometrically, vehicle abstraction consists of parametric curves and surfaces, which are later in the design process designated as representations of structural components such as beams and panels. Volumetric enclosures can also be designated as rigid components. During this process, material-related properties are applied for each newly created component in the model.

Higher dimensional properties (gross vehicle measurements, in particular) and inertia properties are calculated based on the material type and geometric specifications. The classes used to calculate these vehicle dimensions and inertia properties represent the most fundamental of the analysis modules used to evaluate the vehicle architecture being designed.

Vehicle architecture development begins with a process of defining an underlying geometry, which is called scaffolding [153]. The functional architecture layout is generated using scaffolding primitives, like points, paths, surfaces and enclosures. Basically, scaffolding is the graphical tool for quick abstraction of the vehicle geometry. Vehicle layout can be quickly modified in the "scaffolding" regime by moving architecture points (Figure 5) and changing the curvature of architecture curves and surfaces.

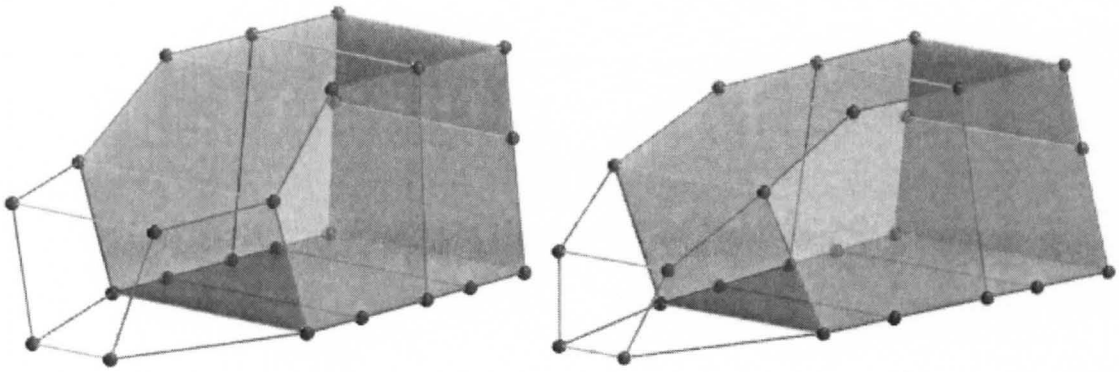


Figure 5. Modification of the geometric model respecting the symmetry constraints (architecture points are depicted in red color).

All geometric primitives in the architecture model can be classified as independent or dependent defined relative to other geometric primitives. Dependent geometric features include:

- points dependent on paths, surfaces, or enclosures;

- paths dependent on paths, surfaces, or enclosures;
- surfaces dependent on surfaces or enclosures;
- enclosures dependent on enclosures.

These dependency relationships between points, paths, surfaces, and enclosures of the scaffolding model are stored in a directed graph date structure. Adding a dependent geometric feature to the directed graph requires specification of the parent primitives, and removal of a geometric primitive requires the removal of any dependent geometric primitives. When the geometry of a geometric feature is changed in the scaffold model, the underlying directed graph is used to update the geometric primitives dependent on the modified primitive.

Yellow lines on the figure represent symmetry constraints, which significantly facilitates the modeling process. These point constraints define relationships between points in the model and assign a rule of how a point can move relative to another point. Available constraints include symmetry with respect to a Cartesian axes, common coordinate constraints, and relative coordinate constraints. Constraints are stored in the directed graph, where nodes represent points and edges represent the dependency of one point upon another.

When specified, these scaffolding primitives can be designated as components, correspondingly beams, panels and nonstructural features (Figure 6). Surfaces may be marked as panel components that carry in-plane loads. Enclosures may be designated as inertial components by marking them as such. Establishing a component model includes specification of materials, beam section shapes, panel thicknesses for the structural member, and inertial properties for the nonstructural. Once a material type from the

material database is specified, the corresponding mechanical properties, such as density, Young's modulus, ultimate and yield stresses are automatically assigned to the corresponding component.

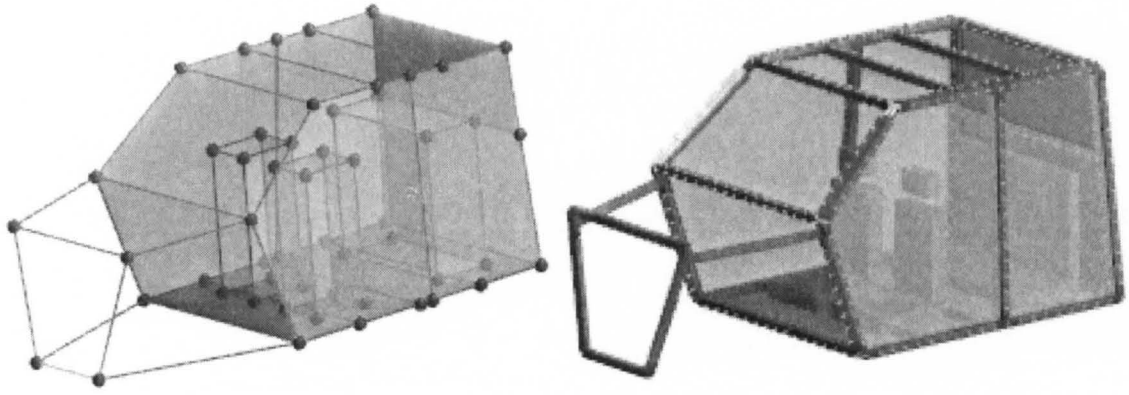


Figure 6. Cab geometric model and corresponding component representation.

At the architecture/conceptual level, vehicle abstraction is of necessity fundamentally different than a detail-level abstraction. In general, these abstractions must include general shape and layout, major spatial features (nominal volumes, in particular), component and subsystem connections (which can be quite simple relative to a detailed CAD model), and inertia properties. They should describe critical subsystems, including, but not limited to primary body structure, suspensions and powertrain, energy storage and transfer elements, and the human operator and passengers. Typical trucks created with our vehicle architecture software are depicted on the Figure 7.

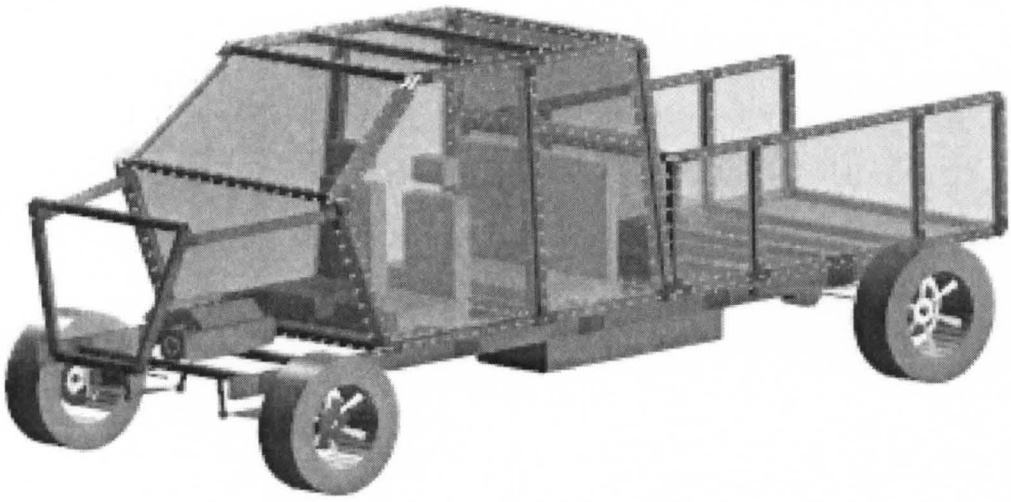


Figure 7. Vehicle concept model renderings for the example cases.

Vehicle abstraction is best represented as a hierarchy ordered according to functions, and to a certain extent, projected physical complexity. On the very top level of the abstraction hierarchy are assemblies and assembly connections. Assembly connections store the connectivity information between assemblies. Further, each assembly can include beam, panel, rigid components, and component connections (Figure 8 and Figure 9). Like assembly connections, component connections provide the connectivity specification between components inside an assembly. Components and connectivity information of each assembly are stored in the graph data structure, where components are represented by nodes and component connection by edges (Figure 10).

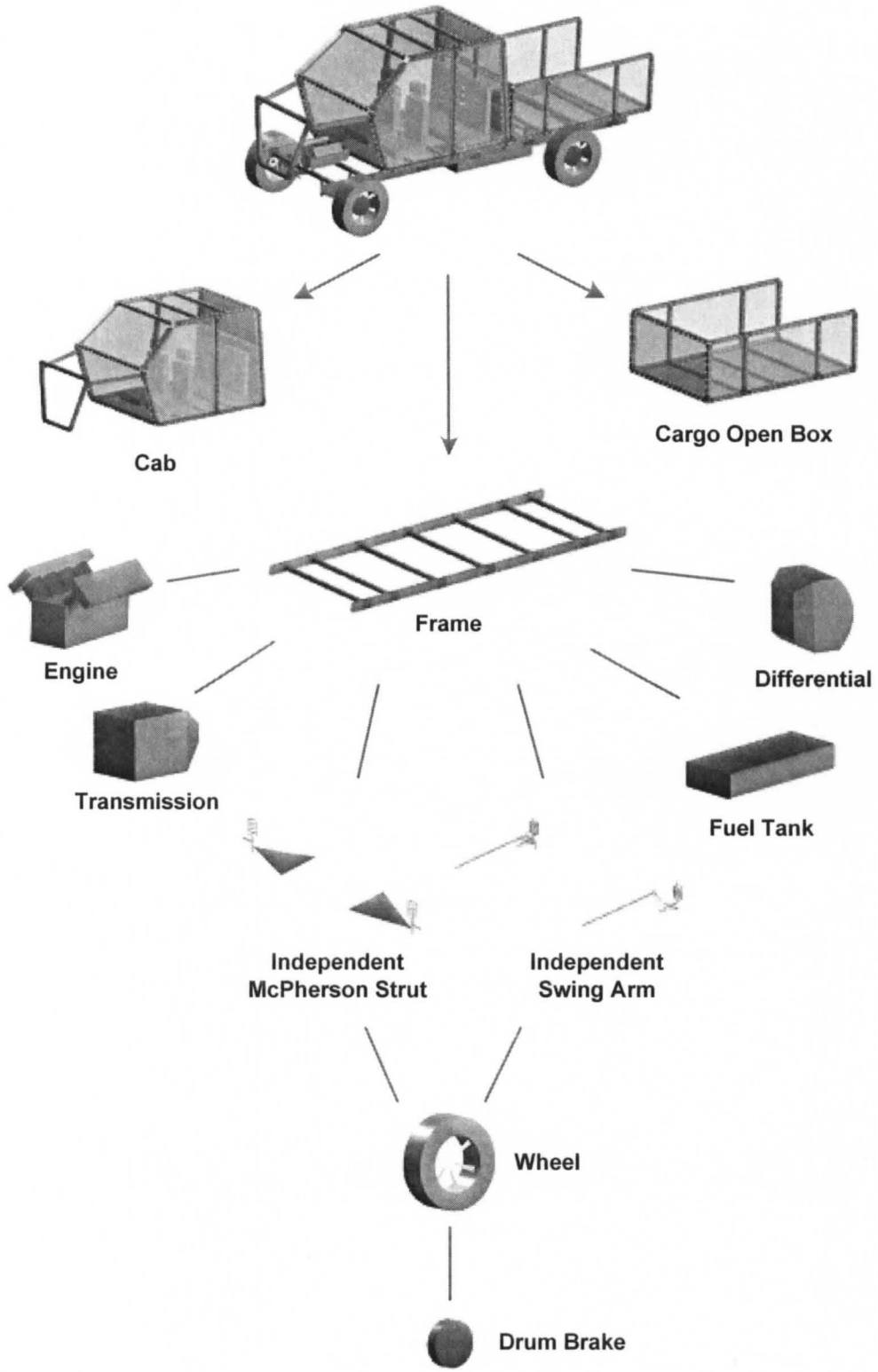


Figure 8. Architecture representation of the vehicle assemblies, sub-assemblies and connections between them.

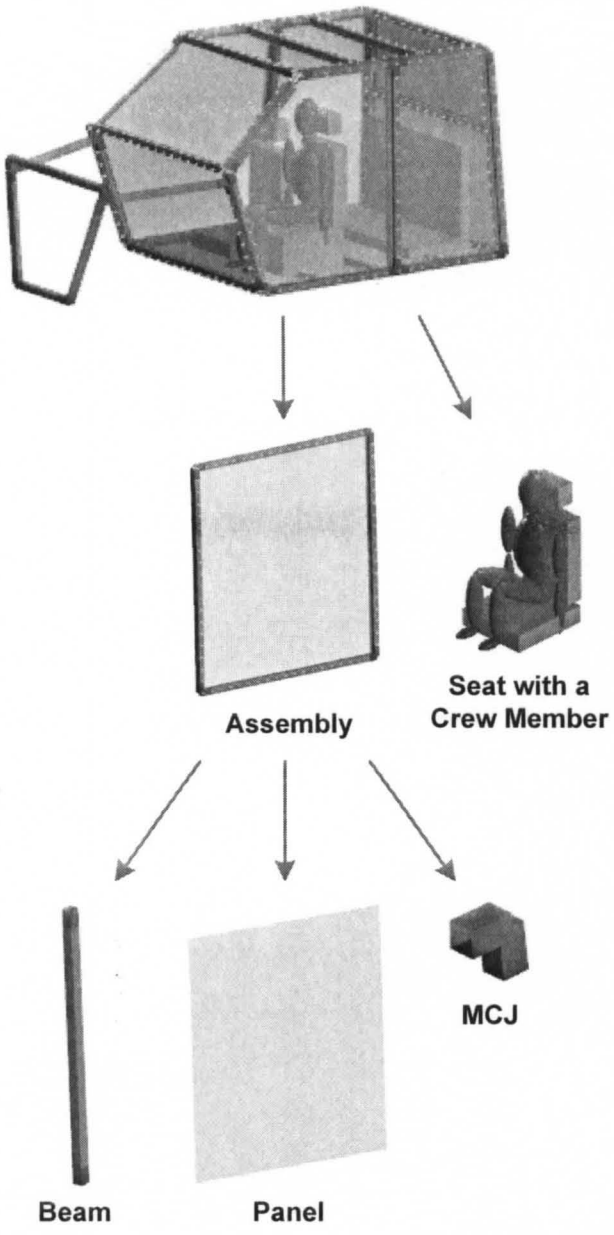


Figure 9. Cab architecture representation.

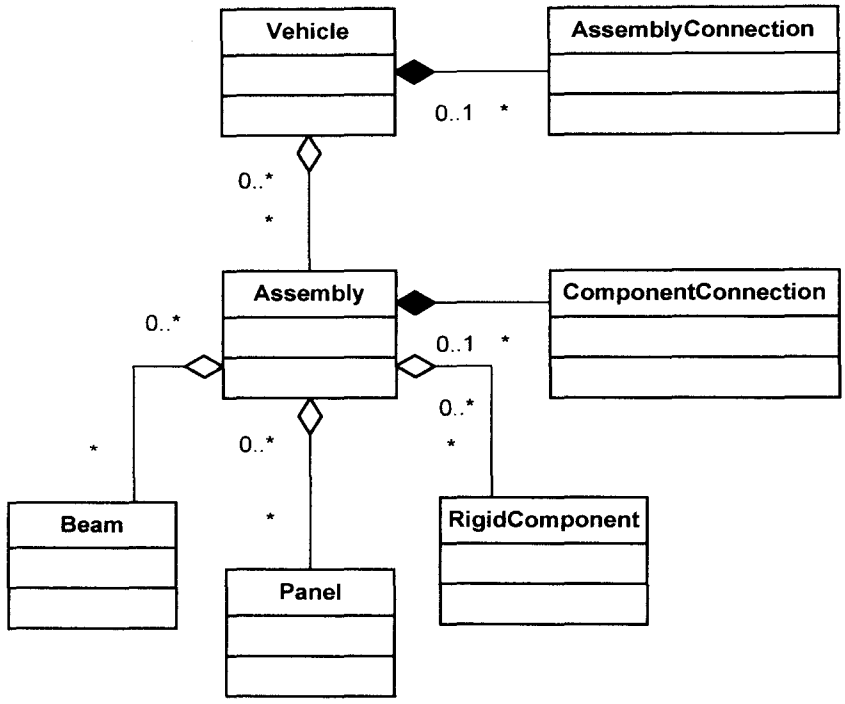


Figure 10. UML class diagram of the vehicle abstraction and aggregation and composition relationships that make up a portion of the vehicle hierarchy (properties and methods not shown).

3.5 Library of Reusable Components and Assemblies

The architecture modeling approach is supported by component library in which reusable component/assembly models can be stored. For a large vehicle design project that involves a number of designers and many similar vehicle architectures, a component library ensures one does not waste his time creating components that already exist. Reusable components can be plugged directly into a vehicle model through their connection interfaces, and modified to meet the design requirements. Furthermore, when developing such a plug-and-play model library, vehicle models can eventually be built by only using component models, with little to no modeling from scratch.

As the library grows, new component models are easier to design. This usually can be achieved by component evolution, where an existing element is modified and saved to a new component file.

In order for a component models library to be properly used, a few conditions should be respected by a team of designers. These conditions are universal for any library of reusable components, and can be outlined as follows:

- Define an appropriate file format that would allow comprehensive function-based component model description
- Component repository should be created, where designers will add new components for further reuse
- Validation of the new components in the library is required in order to prevent contribution of invalid components
- Component help documentation will provide instructions for understanding, proper utilization and modification of component models stored in the repository

Function-based component models require a unique data format that would contain not only the geometric model, but also functional information, such as material properties, inertial properties, connection interface specification, mathematical models, and other engineering, or even aesthetic information. Such function-based representation goes beyond the common CAD model representation, and offers extensive behavioral description of a component.

Establishment of a library for component-based modeling leads to increased automation of model development process, and as a result, to amplified economic benefits such as reduced costs, reduced time, and increased quality. Some broader advantages of using this approach are less obvious. For example, good 'engineering practices' are easier to exchange between different projects by exchanging reusable

components that have been validated and proven to be effective in previous projects. In this way engineering knowledge is not lost when designing from scratch, but is readily available to be efficiently utilized in other projects.

The component library enables quick object-oriented, component-based architecture model development. Proper implementation and usage of the library guarantees that efforts spent finding and extracting components is less than efforts creating them from scratch. Utilizing reusable 'building blocks' from a component library saves much time, prevents double work and keeps the experience across different projects consistent.

3.6 Formal Description of Architecture-level Analyses Hierarchy

Previously in this work we considered a number of analyses which can be utilized during the conceptual design. Each analysis has been discussed with respect to the required level of abstraction of vehicle architecture. As a result of these requirement realizations, we saw very different model detailing levels, ranging from a simple model description for inertia analysis, to much more accurate vehicle specification for the modal and crashworthiness analyses. Obviously, some analyses can be performed much early than others in the vehicle development process. Moreover, some analyses depend on the results of other analyses, and are considered to be sequential, e.g. multibody system dynamics analysis depends on the inertia analysis. Even though not all analyses can be sequentially ordered with respect to each other, they still can be conventionally categorized into three groups based on the corresponding level of vehicle abstraction, Figure 11:

- First phase
- Second phase

- Third phase

Each analysis stage is strongly based on the previous, and deals with more detailed feature descriptions.

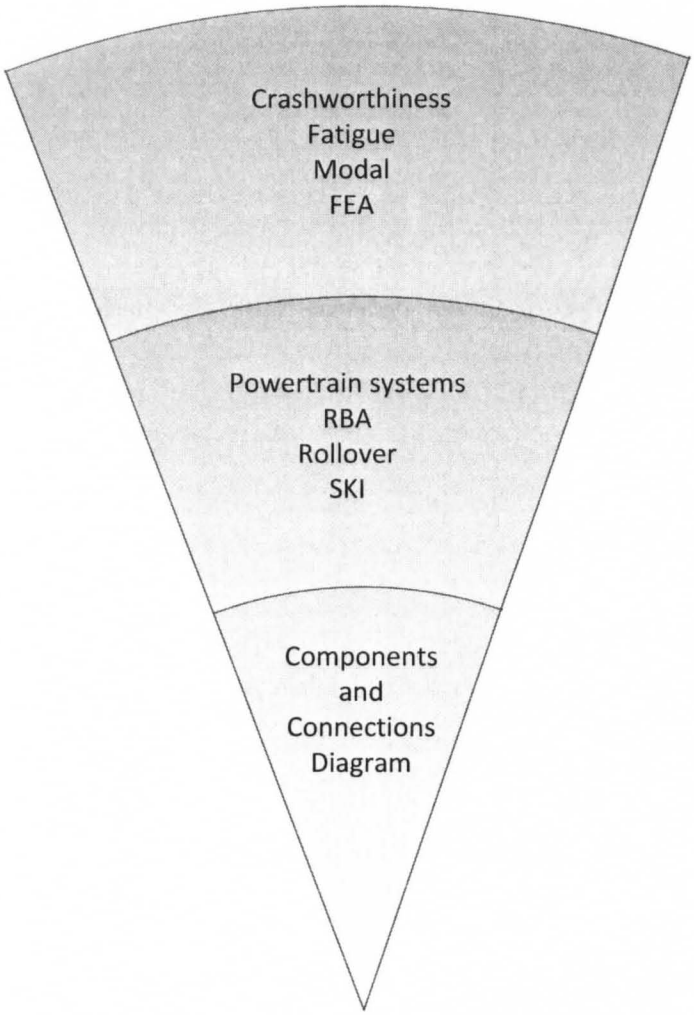


Figure 11. Architecture-level analyses hierarchy.

Concept level design is an iterative process, repeated multiple times over a range of analyses. As we increase detailing of the vehicle architecture, analysis results from the early, more abstract, levels should be updated against the new available design information. Therefore, each stage not only depends but rather "contains" all previous stages. One can notice an analogy with software engineering, where this type of

association corresponds to inheritance between classes. As a result, informally these visual relations between analyses stages can be represented as associations of abstraction between the packages of analysis components, where packages represent analyses stages, Figure 12. This representation reflects the fact that besides new analyses each package contains all analyses from its parent package.

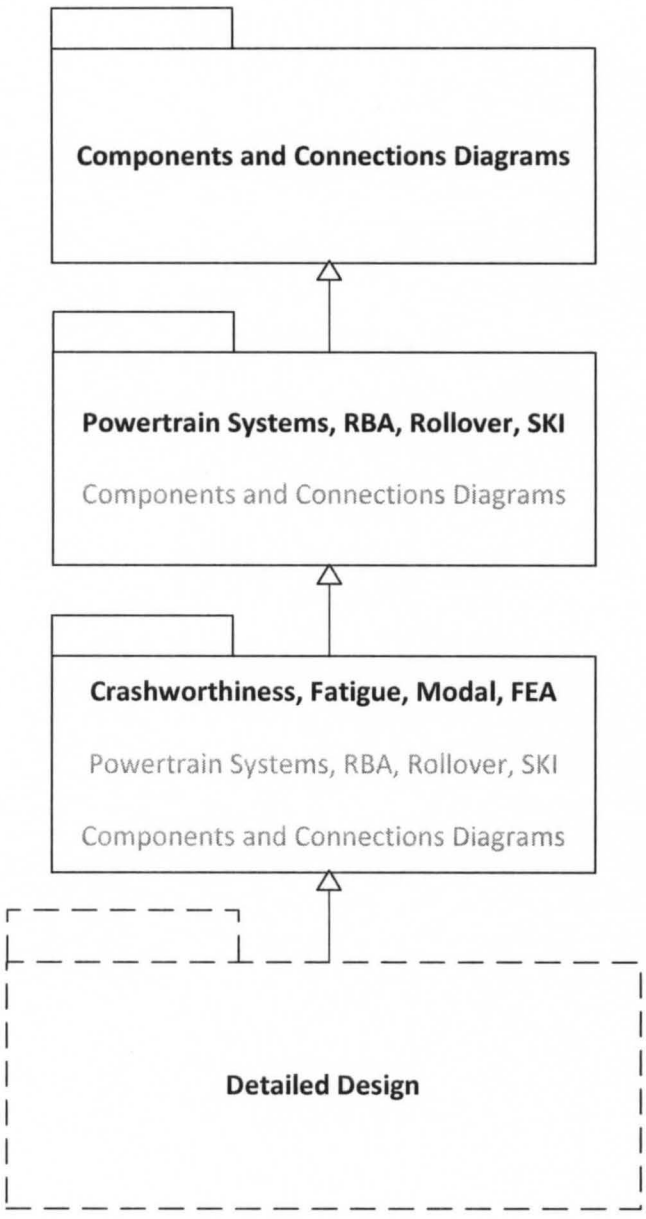


Figure 12. Architecture-level analyses hierarchy represented as association of abstraction.

3.6.1 First Level

In the first very basic category of the analyses hierarchy we deal with the highest level of abstraction. The information about a proposed architecture is extremely sparse, and can be represented on the architecture diagram as a list of components/assemblies and corresponding associations between them. These associations may not represent particular types of connection, but only mark relations between assemblies, and should be properly defined during the following design stages.

On a typical component diagram we can outline a few very general vehicle configuration elements:

- Type of cabin
- Type of passengers' seats
- Type of frame
- Type of front suspension
- Number of axles and type of rear suspension
- Wheel configuration
- Type of cargo box
- Number of axles and type of the front and rear suspensions for each trailer

Figure 13 demonstrates the vehicle component diagram for a six-axle multi-trailer. In the following stage, when further design information is available, the diagram can be updated if necessary. For example, in order to perform the SKI, RBA and Powertrain analyses during the second phase, we need to know specific types of assembly connections and powertrain system components. Based on this information the component diagram can be

updated to include new components/subassemblies and even add the connectivity information between the existing assemblies.

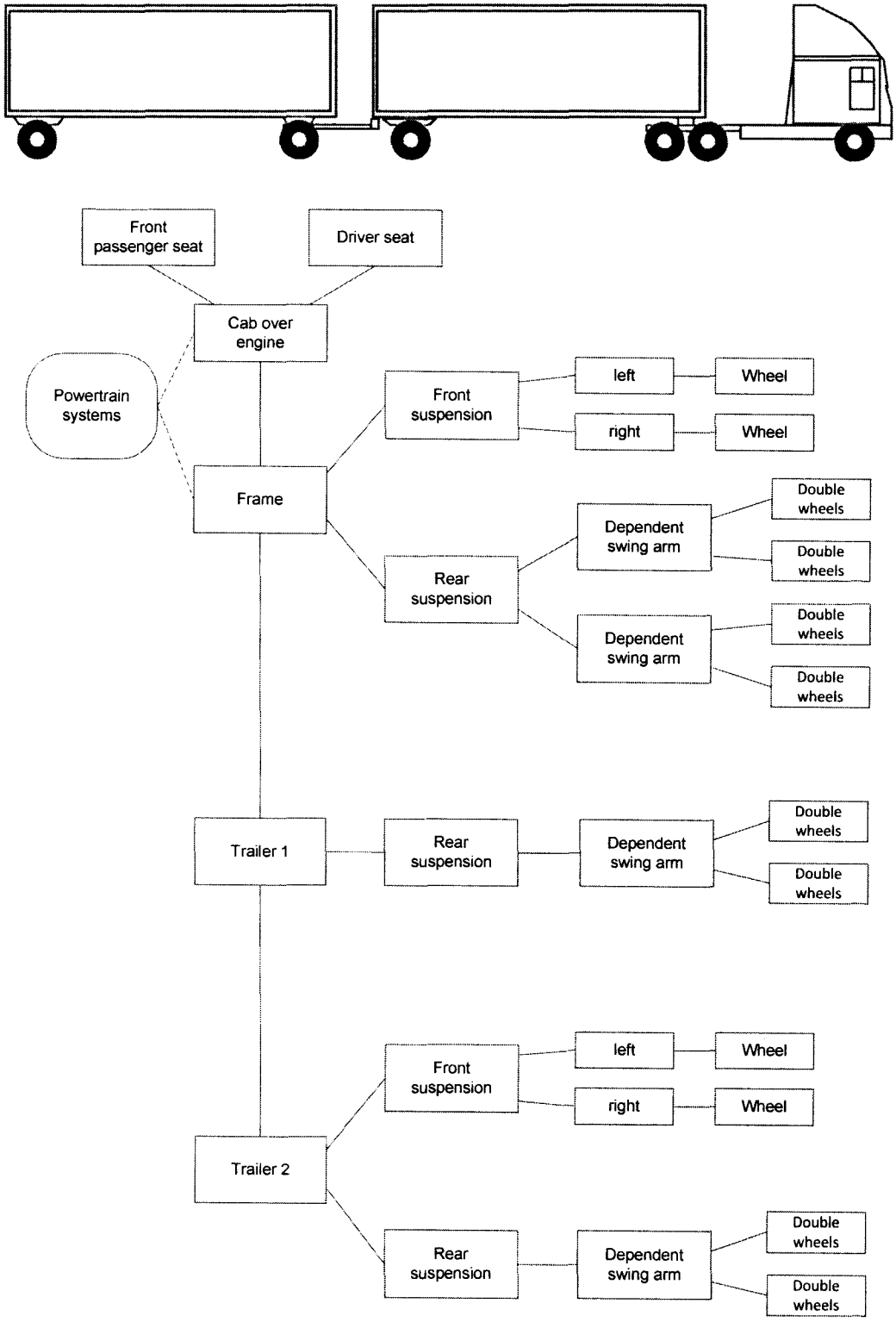


Figure 13. Component diagram.

3.6.2 Second Level

Next stage would begin with defining assemblies' geometry and positioning them respectively to each other in the vehicle architecture. Packaging and basic load path requirements are two design processes that should be done at this level of abstraction in order to avoid potential problems later, during more detailed stages.

Geometric design at this level of the architecture realization is relatively simple and done in the "scaffolding" regime. It involves only linear beam path profiles and plane surface geometry, Figure 14. Material selection and preliminary cross-section definition also must be done at this stage in order to estimate assemblies' inertia properties.

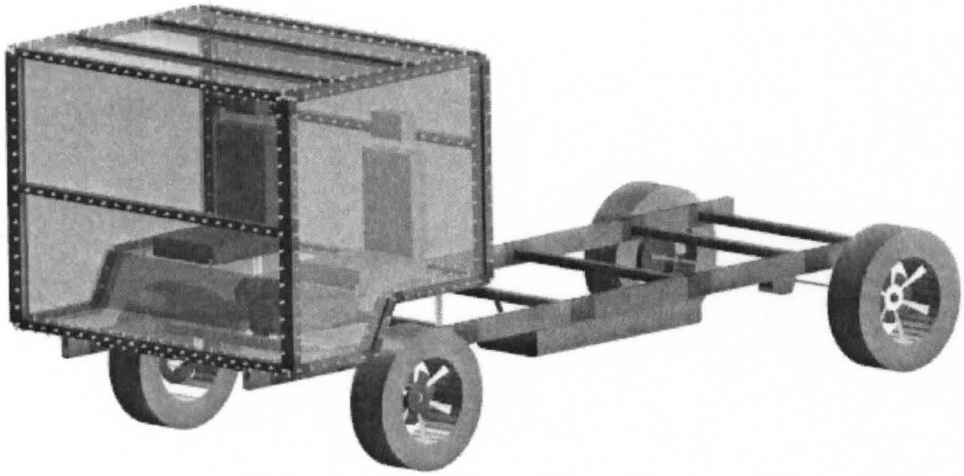


Figure 14. Example of a typical cab-over-engine vehicle designed during the second concept stage.

As a result, at this stage of the vehicle abstraction we have enough data to perform four types of analyses: SKI, Rollover, RBA and Powertrain analysis. Specific assembly connections, as well as suspension types and characteristics should be available at this stage in order to estimate basic driving performance and rigid body modes of vibration using RBA.

This stage is also appropriate for the preliminary powertrain analysis. Simplified definition of powertrain systems is sufficient for acceleration, fuel efficiency and braking performance. Though accurate for acceleration performance, this analysis would be less accurate for fuel efficiency and braking distance estimation. To perform minimum braking distance and fuel efficiency calculation with acceptable accuracy, we need more detailed specification for the relevant structural components. For example, fuel efficiency is a function of aerodynamic drag, which in turn depends on the vehicle load path requirements. Another example is braking performance, where the estimation may not be accurate enough at the earlier design stages, and requires more detailed description of the vehicle brake and wheel models.

3.6.3 Third Level

Next step in the vehicle specification is to add more details to the geometric model. Geometric details are of particular importance for FEA, Fatigue, and especially for modal and crashworthiness analyses. Such details as surface curvature and cutouts can significantly change prediction for the whole body vibrational characteristics. For example, crown panels can have a few times higher natural frequencies comparing to the flat panels with the same dimensions. An accurate prediction of the modal characteristics of a vehicle body is critical in order to avoid major vibration source excitation frequencies by detuning the vehicle body natural frequencies away from the source frequencies.

In case of design for crashworthiness, geometric details are also of particular importance. Depending on the beam's mode of failure, we can see significant variation of the vehicle mass. The optimal failure mode for a beam during the crush event is failure

by crumbling in an accordion fashion. This can usually be achieved by designing a straight beam with crush initiators. Now, consider the situation where the beam is not straight but has a certain level of curvature. In this case, it is more likely that the beam will fail due to plastic hinge formed at the point of highest curvature or stress concentration. For the plastic hinge to generate the same reaction force as the straight beam that fails by crumbling, the beam should be much heavier. Therefore, even very little information about beam and surface curvature would be very useful at the conceptual stage.

Finally, it is expected that this stage contains sufficient details to perform comprehensive powertrain analysis, including fuel efficiency and minimum braking distance estimation. Besides complete powertrain system architecture, the analyses require significantly detailed definition of wheel and brake models, as well as road properties such as friction coefficient and road grade.

CHAPTER 4

SURFACE DESIGN AND ABSTRACTION

The purpose of traditional CAD software packages is to make the representation of a solid model as realistic as possible. Instead of an exact geometric model, a concept model is based on functional feature abstractions. Simplified representation of functional elements (e.g. beams, panels, joints, powertrain components) reflects the designer's intention to assign a specific role for each component. Therefore, in agreement with functional abstractions, concept model rendering also should be function-based rather than geometry-based. The visual representation is a rendering that indicates at a quick glance what a feature does, how it interfaces with adjoining or associated objects, what is its object type, and abstracted property values (Figure 15).

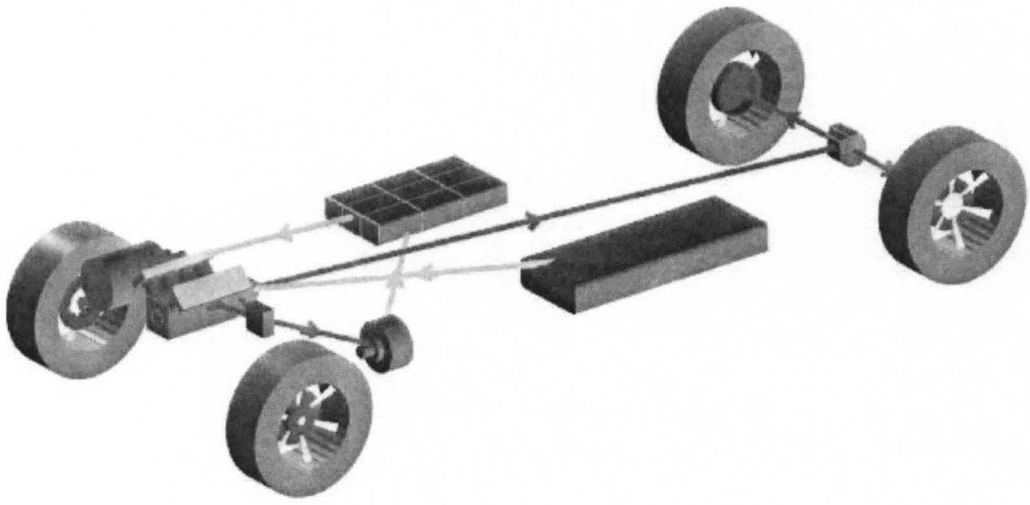


Figure 15. Example of function-cued rendering of vehicle energy and power transmission path abstractions for parallel electric hybrid rear wheel drive powertrains.

This chapter addresses a number of important questions related to the surface characterization at the architecture level. The main purpose of the chapter is to outline the minimum methodology sufficient for development of geometric modeling tools for concept vehicle architecture generation. Such methodology includes algorithms for curve and surface modeling, modification and trimming, as well as surface representation.

Open literature describes the need for coarse surface models, but simply does not address important issues (data structures, optimal curve and surface modeling schemes, rendering without implying details that are not part of the abstraction) associated with architecture-level design and analysis. Simplified models can quickly become more complex than detailed models. As currently realized, they require almost the same level of specialized expertise as that required for traditional models. Surface abstractions, in particular, are problematic. If too much detail is included, the concept model becomes just that: a detailed model. Too little, and the abstraction fails to capture properties that are essential to the architecture optimization process. There is little in the literature about using surface abstractions with an integrated family of design/analysis, and very little description of reusing discretization methods or even data structures for rendering, SKI calculations, and finite element meshing.

Even though conceptual design implies simplified modeling, some non-trivial geometric features that are critical for the vehicle architecture still need to be represented and included in analysis model. Such features and functionality include curvature of curves and surfaces, cutouts on surfaces, shape modification and trimming functions, and also auxiliary functionality to support geometric constraints.

Curves and surfaces are the primitive geometric units required to define the component architecture layout representing the structural components or non-structural elements of the architecture. Curves represent boundaries of surfaces, geometry of beam components, or non-structural geometry. Similarly, surfaces represent boundaries of volumetric features, geometry of panel components, or non-structural geometry.

Beams are characterized by their profiles, which are defined in terms of a curve path type and geometric control points defining the shape. In the most general case the beam path is represented by a parametric equation, for example locally interpolated cubic B-spline. Panels are represented by surfaces and may be of fairly complex geometry. In the most general case panel can be depicted as a composition of trimmed parametric surfaces with cutouts on a surface defined by parametric B-spline curves in the parametric domain of the surface.

4.1 Parametric Curve Modeling Methods

Reasons behind using a parametric surface for surface modeling are well understood and it is hard to overestimate. Neither explicit nor implicit formulation can give such flexibility during the geometric design. Another advantage of having a parametric surface representation can be seen from the point of view of meshing. A parametric surface can be meshed in its 2D domain, i.e. in parametric domain, which is much faster than meshing directly in Cartesian space. Besides its high flexibility and robustness for geometrical operations, parametric curves and surfaces have all the necessary attributes to work well with meshing algorithms. Particularly, a parametric surface provides all the information about the geometry at any point on a surface, such as curvature and metric information for both way mapping.

Geometric modeling of vehicle architecture begins with curve modeling. A curve can be defined in a number of ways including explicit and implicit formulations. The most flexible, robust and widely used in practice formulation, namely a parametric definition, has been chosen. B-spline [46] is the type of parametric curve defined as

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i \tag{4.1}$$

where $0 \le u \le 1$ – parameter, $\mathbf{P}_i$ are the control points, p – degree of the curve, and $N_{i,p}(u)$ are basic functions defined on the knot vector

$$U = \{0,\ldots,0,u_{p+1},\ldots,u_{m-p-1},1,\ldots,1\} \tag{4.2}$$

4.1.1 Local and Global Interpolation

There are a number of algorithms for curve interpolation and approximation using a B-spline. Few of them are based on global or local approximation, such as least square fitting. The drawback of an approximation approach is that it only approaches the curve, but, in general does not pass through a given list of data points. In this respect, curve interpolation is a more appropriate choice, since the constructed curve always goes through the set of data points. The family of interpolation schemes can be divided in two categories: global and local interpolation. Although both methods have been implemented in CMTS, it is obvious that the local scheme is more preferable over the global interpolation.

The drawbacks of global interpolation may be summarized as follows:

- A large part of a curve depends on many neighboring control points, thus the user has poor local control over the shape of a curve;

- No perfect parameterization exists, therefore wiggling artifacts can be seen if a few data points are too close to each other;
- A system of linear equations has to be solved to find control points. This is very time-inefficient, since for many data points the system is very large;
- Every time a user changes a control point, the system has to be resolved.

Local interpolation suffers from none of these problems, thus allowing instant and flexible control over the shape. One approach that deserves special attention is local cubic B-spline interpolation, which sequentially passes through data points and generates a curve (Figure 16). This tool allows creation of arbitrary shapes while avoiding the problems native to the global interpolation schemes, such as curve wiggling. Another important consideration is that the local cubic B-spline interpolation method guarantees generation of constant speed curves, which is essential for computationally effective meshing. It is a very time-efficient method for shape generation. Besides decent local control, this method allows the user to create corners at any data point if necessary. Therefore, any shape can be approximated very accurately using only one curve.

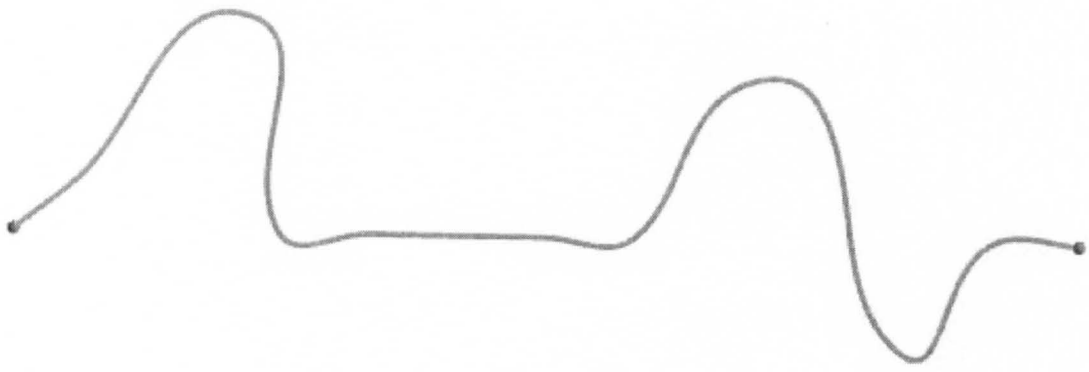


Figure 16. B-Spline curve with 11 interior control points.

4.2 Surface Modeling Methods

The next step is surface modeling. There is a gap in the literature pertaining to automatic surface interpolation from an arbitrary bounding path. Very few algorithms are devoted to this problem, and those available do not always provide a desirable solution. This procedure is usually delegated to the users, who intuitively concatenate or split bounding path segments to create a four sided patch. In order to eliminate the necessity for user interaction, we extend an algorithm proposed by Mitchell [154]. Rather than picking the end points of the segment curves as corners, all curves are split in each internal corner point. It is worth mentioning that the quality of the mapped mesh depends heavily on a side picking algorithm.

Well established approaches for free-form surface modeling exist. As soon as pairs of opposite sides are brought to the same knot vector, a B-spline surface can be generated. Initial positions of the internal data points are usually calculated using Coons interpolation scheme. The local interpolation algorithm for surface modeling is very similar to the corresponding curve modeling algorithm, and performs consecutive patch-by-patch surface generation.

Following is a parametric surface definition:

$$\mathbf{S}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)\mathbf{P}_{i,j} \tag{4.3}$$

where $\mathbf{P}_{i,j}$ is a bidirectional net of control points, p and q - degrees of the surface in u and v - directions correspondingly. Basic functions are defined on the knot vectors

$$U = \{0,\ldots,0,u_{p+1},\ldots,u_{k-p-1},1,\ldots,1\} \tag{4.4a}$$

$$V=\{0,\ldots,0,v_{q+1},\ldots,v_{l-q-1},1,\ldots,1\} \tag{4.4b}$$

Usually, as an input from the user interface, only the boundary of a surface is received. However, the corresponding B-spline surface needs a bidirectional net of control points. This set of control points can be found using bidirectional interpolation of the bounding curves by means of Coons patch approach. Let $\mathbf{a}_0(v)$, $\mathbf{a}_1(v)$ $\mathbf{b}_0(v)$ and $\mathbf{b}_1(v)$ be four bounding curves. Then the Coons surface is defined as following:

$$\mathbf{r}(u,v)=\mathbf{r}_1(u,v)+\mathbf{r}_2(u,v)-\mathbf{r}_3(u,v) \tag{4.5}$$

with

$$\mathbf{r}_1(u,v)=(1-v)\mathbf{b}_0(u)+v\mathbf{b}_1(u) \tag{4.6a}$$

$$\mathbf{r}_2(u,v)=(1-u)\mathbf{a}_0(v)+u\mathbf{a}_1(v) \tag{4.6b}$$

$$\mathbf{r}_3(u,v)=(1-v)(1-u)\mathbf{P}_{00}+vu\mathbf{P}_{11}+u(1-v)\mathbf{P}_{10}+(1-u)v\mathbf{P}_{01} \tag{4.6c}$$

Figure 17 depicts the example of a constructed Coons patch form the four boundary curves. Now, the necessary control net for a B-spline surface local interpolation can be obtained from the Coons surface.

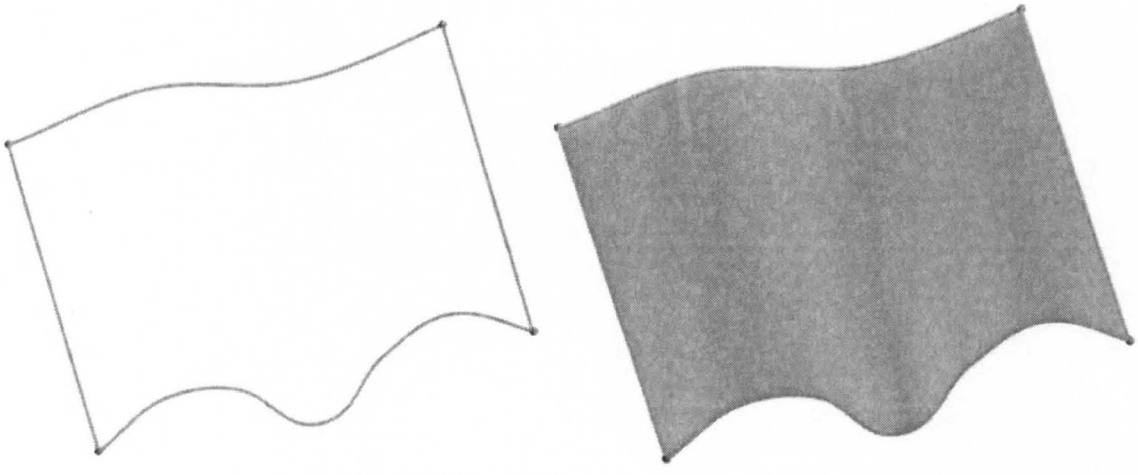


Figure 17. Four boundary curves (left). Interpolated Coons patch (right).

4.3 Curves on a Surface

A cutout is represented by a set of parametric curves $\mathbf{c}(t)$ in the parametric domain (u,v) of the surface: $\mathbf{c}(t)=\{u(t),v(t)\}^T$. In the Euclidian space this curve is defined as $\mathbf{C}(t)=\mathbf{S}(\mathbf{c}(t))=\{x(t),y(t),z(t)\}^T$. Thus the resulting curve is a mapping $\square^1 \rightarrow \square^2 \rightarrow \square^3$. An example of this mapping is shown on the Figure 18.

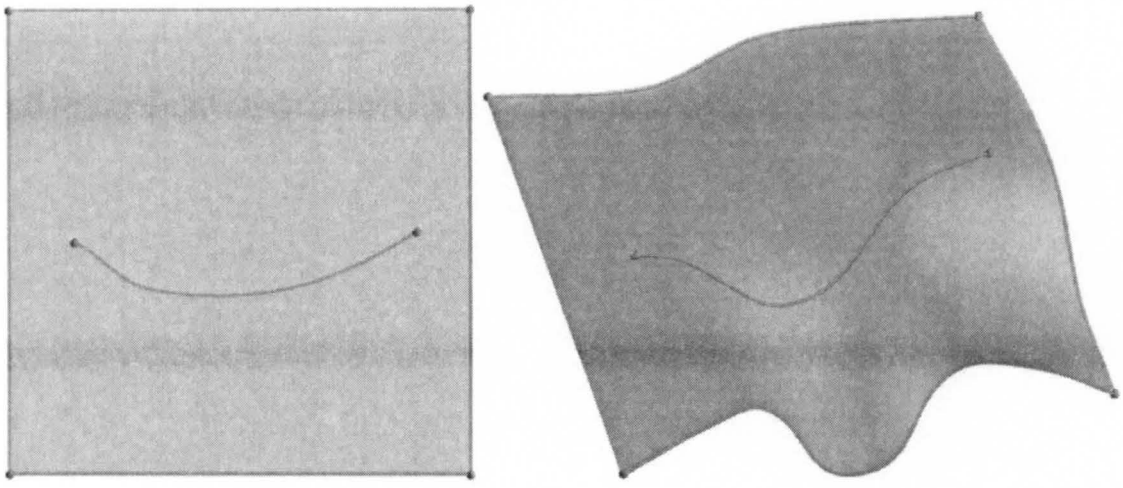


Figure 18. Curve in the parametric space of a surface (left). Curve on the surface (right).

4.4 Curve and Surface Rendering

To render a curve, a recursive method based on subdivision has been used. The criterion for subdivision is the curvature of a curve. Using the same sizing criteria for both curve and surface discretization guarantees synchronization between the bounding curve and the surface rendering.

Trimmed B-spline surfaces have been considered as an important tool for modeling free-form panels in the vehicle modeling process. These surfaces can have very complex shapes and curvatures, as well as arbitrarily trimmed regions. A tessellation algorithm like [45] works well only for relatively flat surfaces, and in case of a more complicated

surface, the algorithm can produce too many triangles in the regions of small curvature, and may not capture the shape in the highly curved regions.

In practice a family of subdivision-based algorithms provides the most accurate approximation of a shape and effectively deals with cutouts. The idea used in the paper of Ng and Tan [48] has been utilized in this work. Even though the authors have developed this method for the deformable shape rendering, it has been found to be very useful for static shapes as well. The algorithm can be divided in three phases. A special data structure, called Background quadtree, is used to store surface sizing information during the first phase of rendering.

The steps in the first phase are as following [58]:

1. Quadtree initialization is based on a given number of control points. The curvature cannot change significantly between two control points.
2. Each initial quad is subdivided to the depth level determined by the local curvature and distortion metric.

During the second phase, each quad is divided in two triangles, thus generating a surface tessellation. Finally, the third phase deals with possible cutouts on the surface. In order to efficiently manipulate triangulations, a special data structure for triangle representation is required. This data structure can look like the one in Figure 19.

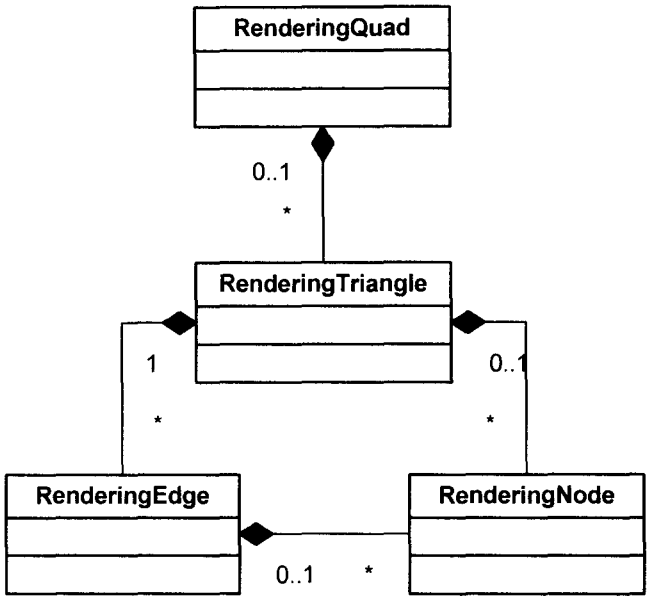


Figure 19. Data structure for the rendering algorithm.

Each edge in a triangle knows about the parent triangle, in other terms, it contains the list of parent triangles. This data structure is necessary for handling cutouts on a surface in a way similar to the quad generation on the triangulated surface. Specifically, an edge recovery algorithm has been used to place trim curves on a surface. The edge recovery algorithm takes a discretized trim curve and accurately merges it with a given surface tessellation, so the resulting cutout follows the discretized curve exactly.

The last step is to remove all of the triangles that are internal to cutouts. It can be implemented as follows:

1. Pick any edge on a cutout, and using a simple inside-outside test, define which triangle adjacent to the edge is inside the cutout.
2. Remember the other two edges of the triangle and delete the current triangle.
3. Execute the above command on the triangles adjacent to the edges from step 2. If the edge is on the cutout boundary don't execute this line.

It is easy to see that the algorithm is recursive and stops when there are no more internal triangles.

Figure 20 demonstrates the application of the above described algorithm. Trimmed surface rendering is shown in Figure 21.

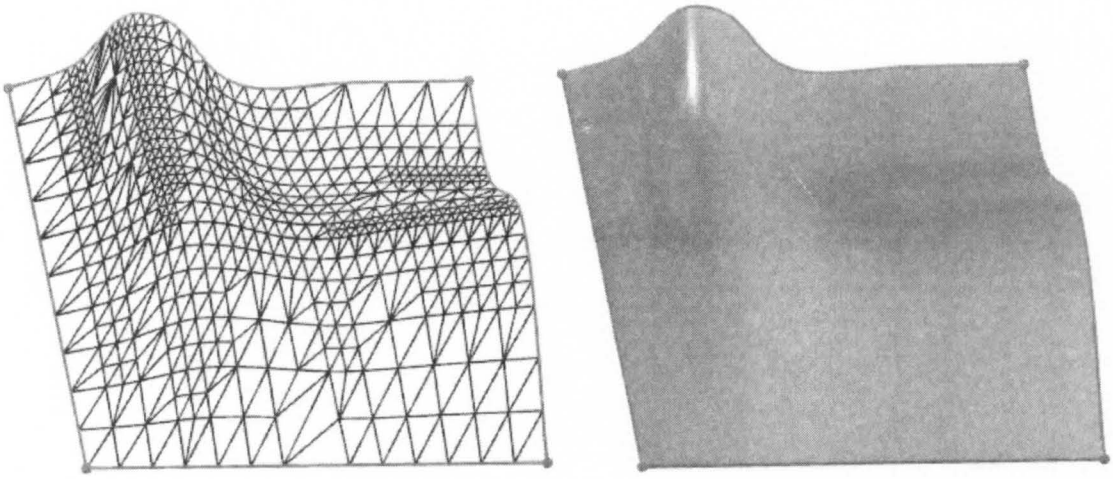


Figure 20. Surface rendering: surface triangulation (left); actual rendering (right).

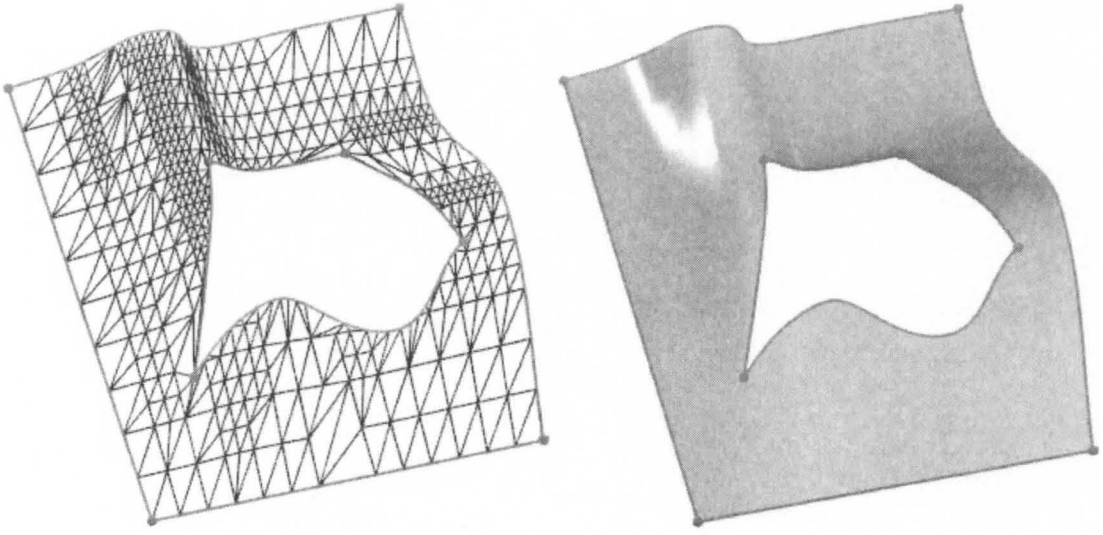


Figure 21. Trimmed surface rendering: surface triangulation (left); actual rendering (right).

The surface rendering routines have been implemented in a comprehensive software package for vehicle concept modeling design and analysis. The rendering in the software is based upon Microsoft XNA, a DirectX-based API for visualization within the

Windows environment. The surface rendering classes support flexible modeling and reusing of discretization methods and data structures for inertia property estimation and mesh generation.

CHAPTER 5

CURVE AND SURFACE DISCRETIZATION

Even though there are many algorithm and strategies for mesh generation, most commercial meshing software requires a significant amount of user input. Decisions must be made about sizing, elements type, grid type, etc. This need for specialist-level input explains the differentiation between "analysts" and "designers" seen in nearly all vehicle design efforts. The concept modeling approach proposed in this work requires most decisions to be made independently of user input. All necessary information can be extracted from the vehicle model. Mesh generation does not change the geometric or mechanical properties of an underlying component model, thus, theoretically, there is no reason for a designer to interact with the meshing module. As a result, a designer performing these analyses should, in principle, not need to be familiar with finite element theory or meshing methodologies.

The objective of this chapter is to describe a mesh generation methodology for a concept vehicle body that consists of a composition of panels represented by surface patches. Usually vehicle body geometric models at concept level can be defined by a collection of flat or nearly flat patches. In this case, mesh generation is relatively simple, and can be performed using a structured mesh generation strategy. However, for patches that contain regions of high curvature, a structured scheme may not be appropriate; hence, an advanced meshing strategy should be utilized. Furthermore, if a surface contains trimmed features, the use of an advanced mesh generation is imperative in order

to approximate the shape and achieve a reasonable accuracy. As a result, both structured and unstructured meshing schemes have been utilized in the concept modeling software package. For a single vehicle concept model being analyzed, the run-time difference between hybrid structured/unstructured strategy and pure unstructured strategy is negligible, since meshing is very coarse comparing to the detailed model. However, the concept modeling methodology allows running multiple analyses, as well as performing transient analyses that require mesh adaptation. In this case, employing a structured scheme, where possible, greatly improves computational efficiency of the meshing module.

The discretization of concept model panel components into a finite element mesh is a difficult task for a number of reasons. As a design tool for concept modeling software, the FEA meshing algorithms should be fully automatic, and not require the designer to deal with finite element preprocessing. The model contains both geometry and connectivity information defined through connection relationships between components, and this information is sufficient to perform automatic model discretization. The algorithm takes into account any internal or boundary constraints that may represent points of loading or connection. Finally, to guarantee smooth load transition between rigidly connected panels, the algorithm synchronizes (if possible) meshing interfaces between the patches of the composite surface.

5.1 Curve Discretization

Curve discretization is an initials step before surface meshing, and most often defines the quality of the finite element model in the most critical regions of a panel, regions of stress concentration. During the concept modeling of a vehicle body, designers

usually create nearly rectangular patches. If these patches do not have any internal features, an algebraic meshing method such as transfinite interpolation can be used. Moreover, for surfaces that can be meshed using algebraic approach, boundaries are discretized into equal arc length elements. The discretization procedure is done directly in the parametric domain, since the parametric curves are unit-speed by design, as described in the previous chapter.

For patches of more complex geometry, unstructured meshing can be used. For the meshing procedures like advancing front, the curvature of the boundary curves has to be respected by using curvature-weighted spacing. Therefore, a more advanced boundary discretization scheme, such as Cuilliere's direct approach [52], should be used. Since by design we have unit-speed curves, discretization occurs in the parametric domain, and thus a substantial speed improvement is achieved comparing to discretization in the Euclidian space. This algorithm starts with the computation of the total number of segments N which need to be created along the length L of the curve with respect to a sizing function. N is obtained by taking it to be equal to the nearest integer to A, where A is

$$A = \int_0^L \frac{\mathrm{d}s}{E_d(s)} \tag{5.1}$$

$E_d(s)$ is a sizing function, and s is the real distance along the curve. Sizing function defines the size of an element in the vicinity of the coordinate s, and usually is based on the curvature at that point. A number of segments N is not an integer in most cases, which means that the length of the last segment of the curve discretization cannot satisfy the sizing function. Thus, we have to find the value of t_{i+1} that satisfies the condition

$$\int_{t_i}^{t_{i+1}} \frac{1}{E_d(t)} \left(\frac{ds}{dt} \right) dt = \Delta A_s \tag{5.2}$$

where

$$\Delta A_s = \frac{A}{N} \tag{5.3}$$

For the 3D curve

$$\frac{ds}{dt} = \frac{d\mathbf{C}(t)}{dt} \tag{5.4}$$

Similarly, using the same approach parametric curves on a surface can be discretized. In this case, the difference is only in calculating the differential of real distance ds along the curve. Since the parametric curve is on the surface $\mathbf{S}(u, v)$ we have the following relation:

$$\frac{ds}{dt} = \sqrt{E\left(\frac{du}{dt}\right)^2 + F\left(\frac{du}{dt}\right)\left(\frac{dv}{dt}\right) + G\left(\frac{dv}{dt}\right)^2} \tag{5.5}$$

E, F and G are the first fundamental quantities of the surface.

Wu and Wang in [53] have improved the computational efforts of the Cuilliere's algorithm. They used a Newton iterative method to find the parametric position t_{i+1} of a next node. Let

$$f(t) = \int_{t_i}^{t} \frac{1}{E_d(t)} \left(\frac{ds}{dt} \right) dt - \Delta A_s \tag{5.6}$$

Solution of the equation $f(t) = 0$ gives us the desired parametric position t_{i+1}. The algorithm is implemented as following:

1. Calculate $f(t_i)$ and $f'(t_i)$, where

$$f'(t_i) = \frac{1}{E_d(t)}\left(\frac{\mathrm{d}s}{\mathrm{d}t}\right) \tag{5.7}$$

2. Compute $t_{i+1} = t_i - f(t_i)/f'(t_i)$.

3. The iteration is finished if $|t_{i+1} - t_i| < \varepsilon_1$ or $|f(t_{i+1})| < \varepsilon_2$.

Sample results for the curve meshing procedure are depicted in Figure 22 and Figure 23.

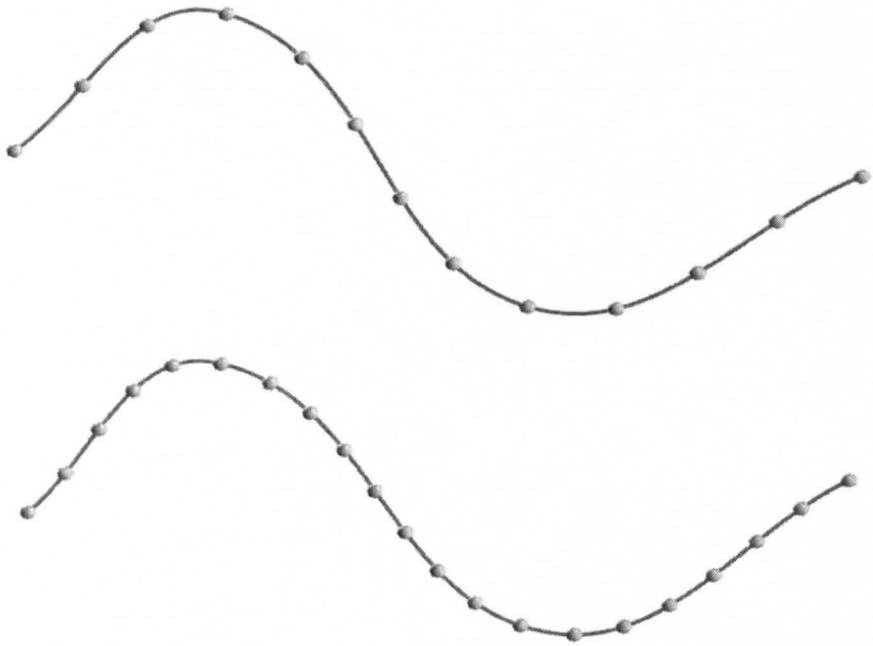


Figure 22. Two sizing levels of equal arc length node distribution.

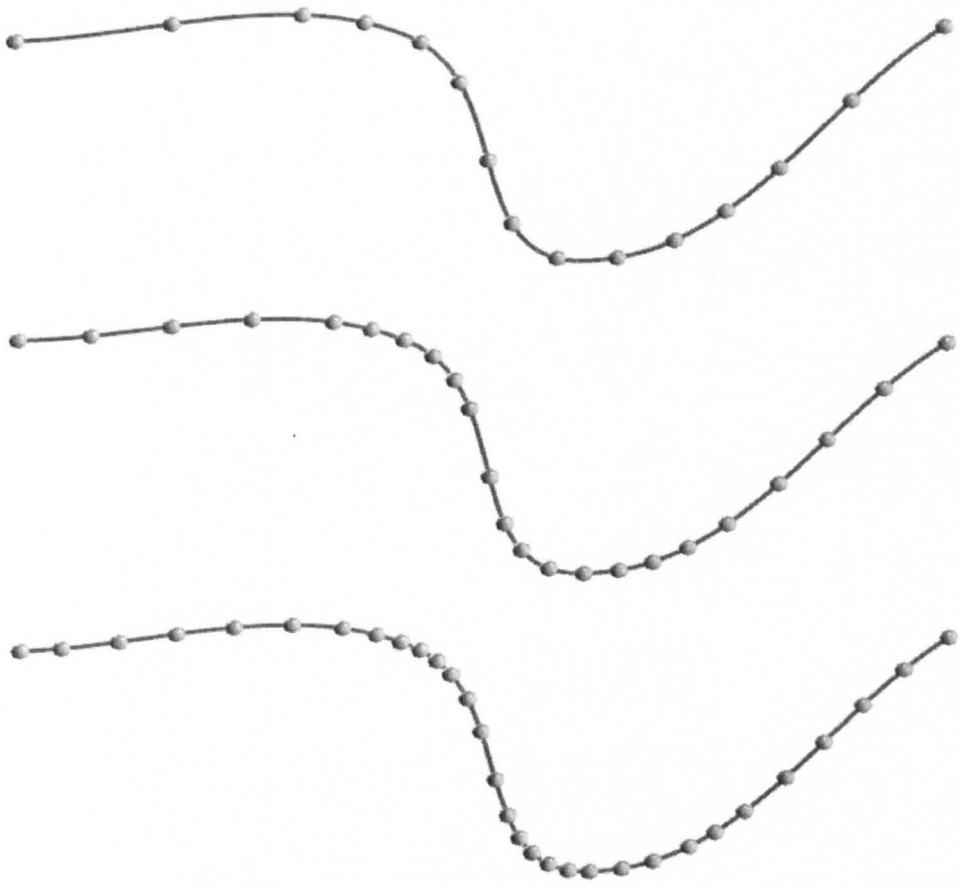


Figure 23. Three sizing levels of curvature-weighted node distribution.

5.2 Algebraic Meshing

In this work we limited ourselves only to the usage of a simple quad mesher based on transfinite interpolation (TFI). Transfinite or mapped meshing is of prime importance for vehicle finite element analyses. The method produces structured orthogonal meshes and is very simple to implement. The grid can be generated either based on four sides of the underlying surface, or a more advanced procedure of corners picking can be used. The quality of the mapped mesh heavily depends on a side picking algorithm and can be significantly improved with an intelligent algorithm.

Since the boundary curve parameterization is constant speed, we divide opposite sides into an equal number of elements in parametric space. All the internal nodes are

interpolated using Coons technics described earlier. An example of the surface transfinite interpolation is shown in Figure 24.

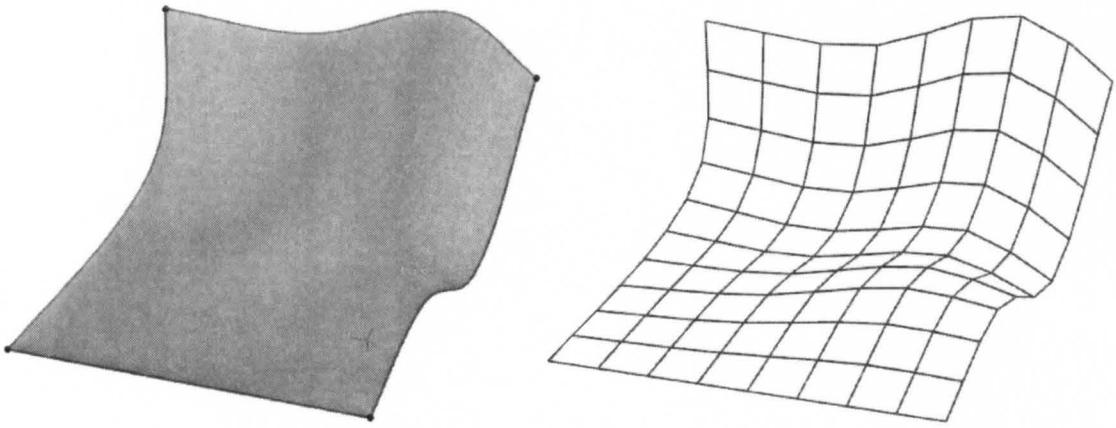


Figure 24. Example of TFI.

5.3 Background Meshing

Background meshing has been shortly introduced in the Chapter 4 with application to surface rendering, and here background meshing is used for a slightly different purpose. Specifically, a background meshing data structure, in this case quadtree [58], stores mapping information and curvature at the surface sampling points. Each node of a quadtree corresponds to assigned points of the surface and contains the mapping and curvature information at this point. Metric data at an arbitrary point can be extracted by interpolating it from the four nodes of the quad that contains this point. Background mesh is generated before proceeding to advancing front analysis. Figure 25 depicts the UML diagram representing the data structure.

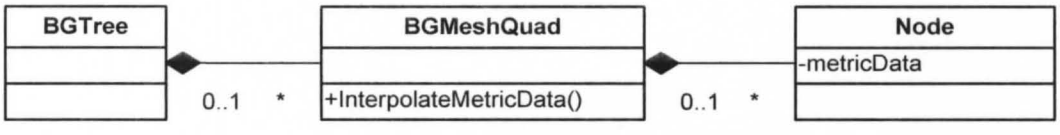


Figure 25. Structure of the background tree.

Background mesh is an important preparation step before using advancing front. The quadtree search is very efficient for retrieving metric information of a surface with high curvature. Example of a surface and its background mesh is shown in Figure 26.

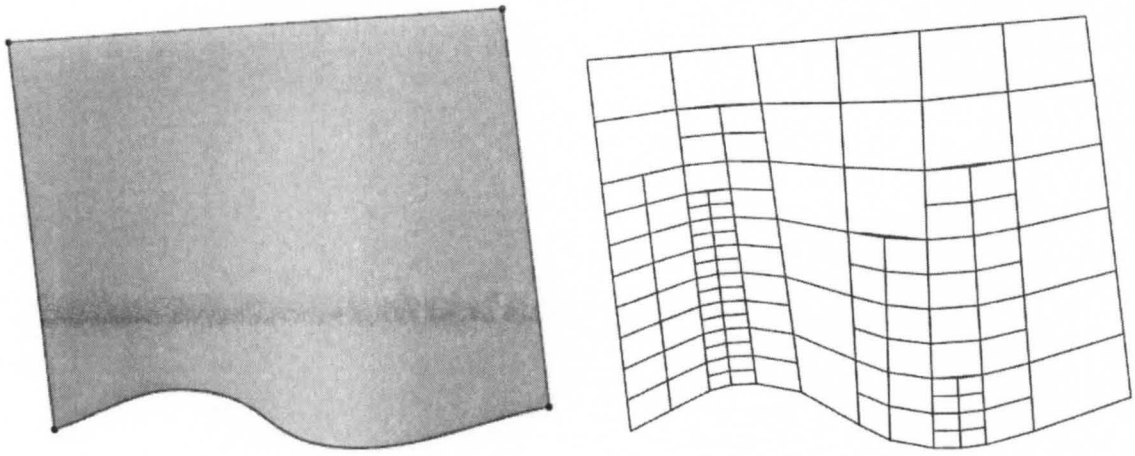


Figure 26. Surface and its background mesh.

An initial quadtree is generated as a grid of size $n \times m$. For each quad the depth is estimated based on the curvature at the middle point of the cell. If the curvature is higher than the predefined threshold, the quad is subdivided into four new quads. The process is recursive and continues until the curvature is less than the threshold or the maximum number of depth is reached.

Additionally, only one level of depth between the neighboring cells is forced. Specifically, a cell of lower depth is refined until the level of difference between neighboring cells will be not more than one.

Each node contains a metric map between 3D surface and its 2D parametric domain. Metric of a tangent plane at every point of a 3D surface, as described in [57], is defined as:

$$[M] = \begin{bmatrix} E & F \\ F & G \end{bmatrix} \tag{5.8}$$

where

$$E=\vec{\tau}_1\cdot\vec{\tau}_1,\quad F=\vec{\tau}_1\cdot\vec{\tau}_2,\quad G=\vec{\tau}_2\cdot\vec{\tau}_2 \tag{5.9}$$

with

$$\vec{\tau}_1=\vec{\sigma}_u(u,v),\quad \vec{\tau}_2=\vec{\sigma}_v(u,v) \tag{5.10}$$

$\vec{\sigma}_u$ and $\vec{\sigma}_v$ are the partial derivatives of the surface at the specific point. Having two points in the parametric domain the corresponding distance on the 3D surface will be

$$D=\sqrt{E(u_1-u_2)^2+2F(u_1-u_2)(v_1-v_2)+G(v_1-v_2)^2} \tag{5.11}$$

Using metric information a unit vector $\vec{N}_{2D}$ perpendicular to the line *AB* can be determined. The vector $\vec{N}_{2D}$ is normal to the line AB in the 3D space of the surface but not perpendicular in the parametric domain (Figure 27).

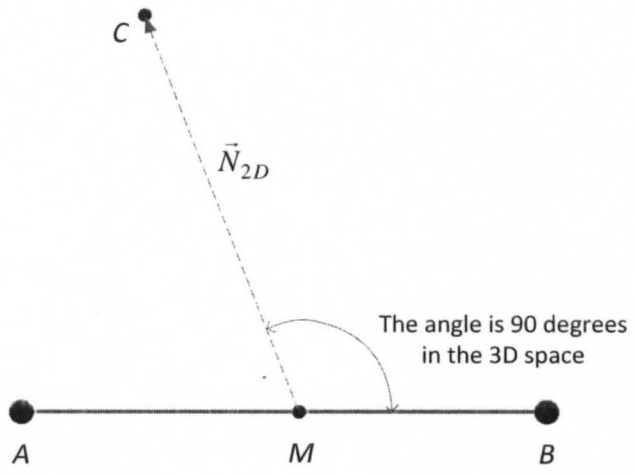


Figure 27. Normal vector to the AB in the 3D space but not necessary perpendicular in 2D.

Vector $\vec{N}_{2D}$ is defined as

$$\vec{N}_{2D}=\begin{Bmatrix}u_{2D}\\ v_{2D}\end{Bmatrix}=\begin{bmatrix}F & G\\ -E & -F\end{bmatrix}\cdot\vec{V}_{AB} \tag{5.12}$$

where $\vec{V}_{AB}$ is a unit vector pointing from A to B. Meshing progresses in the parametric domain, therefore the predefined distance h_{3D} on the surface should be properly mapped to the parametric domain distance h_{2D}:

$$h_{2D} = \frac{h_{3D}}{\sqrt{E \cdot u_{N2D}^2 + 2F \cdot u_{N2D} \cdot v_{N2D} + G \cdot v_{N2D}^2}} \tag{5.13}$$

Based on the above defined mapping a new node on the surface is generated as following:

$$\vec{C} = \vec{m}_{AB} + \vec{N}_{2D} \cdot h_{2D} \tag{5.14}$$

5.4 Advancing Front Meshing

Advancing front meshing algorithm consists of the following steps:

1. Front initialization.
2. Triangle creation.
3. Front updating.
4. Return to step 2 if front is not empty.

5.4.1 Front Initialization

Front initialization process can be divided in a few steps:

1. All exterior edges of the surface boundary should be oriented in the counter-clockwise direction. However, internal cutout boundaries should follow the clockwise direction. In this manner we define the interior and exterior part of a surface. Any interior part always bounded by the counter-clockwise oriented edge loops. These loops form an initial front, which serves as input to the advancing front mesher.

2. It is advantageous for the quality of the grid to proceed with smaller elements first. To accomplish this, edges in the front are sorted according to their 3D size [57].
3. Each edge has to know its neighboring edges. Thus, the class Edge contains its left and right adjacent edges (Figure 28).

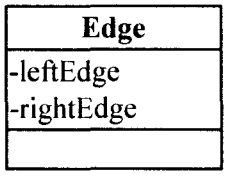


Figure 28. Edge data structure.

5.4.2 Triangle Creation

For each edge in the front a triangle is created by generating or finding the opposite vertex. The process of the triangle generation is described in the algorithm that follows below. More details on this algorithm can be found in [57 and 58]:

1. Using the background quadtree the metric is interpolated in the point M. Based on this metric, the distance MC to the new vertex C in the parametric domain is calculated according to the formula (5.13) and its direction according to the formula (5.12). The direction of the vector MC is normal to the edge AB in the 3D space by design.

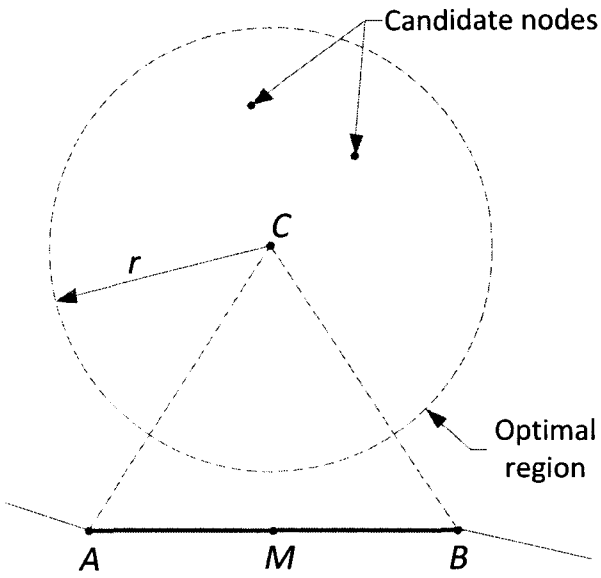


Figure 29. Optimal node finding.

2. An optimal region of radius r is defined as a distance MC multiplied by a certain factor. The algorithm checks if any existing nodes falls into the optimum region. If more than one node already exists in the region, the optimal vertex is chosen based on the best shape factor. If no existing candidate is found a newly generated node C is accepted (Figure 29).
3. Each new triangle is tested to detect any intersection with existing edges. If this is the case, the element is rejected.

Example of the front progressing is demonstrated in Figure 31 for the surface below (Figure 30).

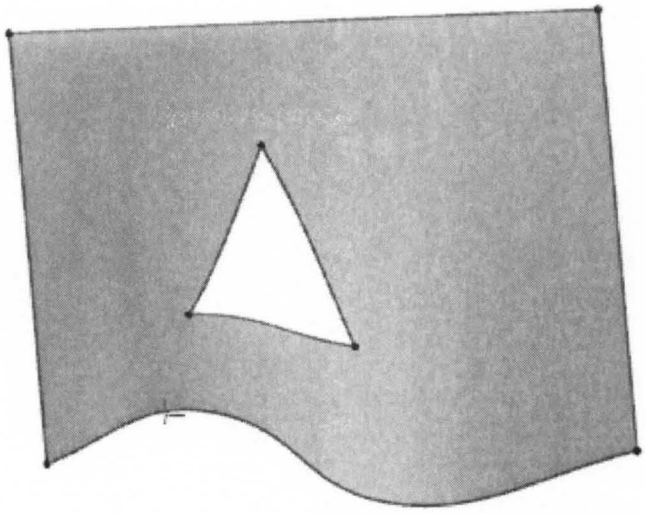


Figure 30. Surface used for advancing front method example.

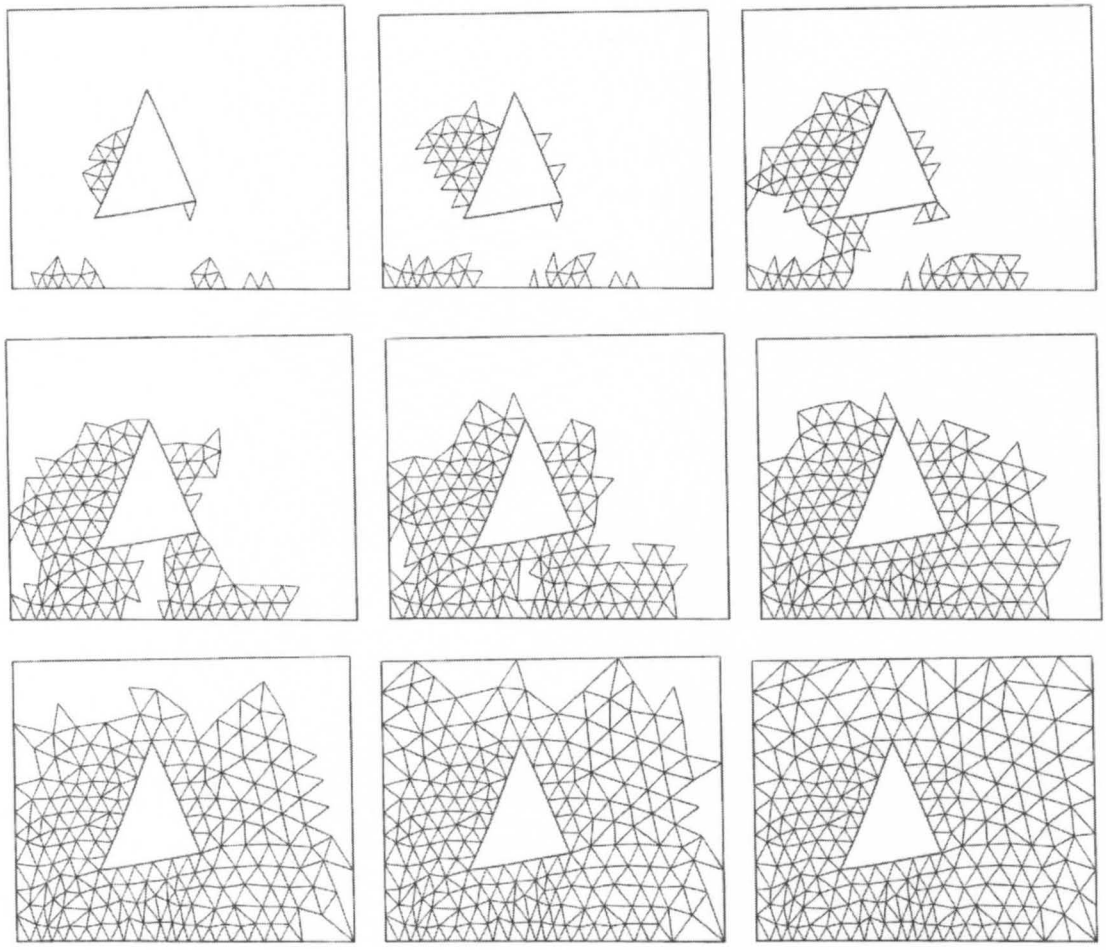


Figure 31. Front progress.

5.4.3 Front Updating

The front has to be updated each time a new element is generated. The base edge and any other edges of the triangle that belong to the front should be deleted. If new edges are not in the front list, they are added to the front. Every time when the algorithm fails to generate a triangle, the base edge is removed from the front list and moved to the list of rejected edges, and will be processed at the very end when the front is empty. If an edge is rejected a second time, the brute force method is used. The method loops through all of the nodes on the front and finds the best shaped triangle.

5.5 Mesh Smoothing

Even though the advancing front usually generates good quality grids without smoothing, a Laplace technique is used to smooth the transition between elements' sizes and thus increase quality of a mesh (Figure 32). This algorithm has been implemented and is used for both algebraic and advancing front meshers [58]. Smoothing is done in parametric space, but takes into account size distortion metrics between parametric and 3D spaces.

$$\mathbf{X}_0^{n+1} = \mathbf{X}_0^n + \varphi \frac{\sum_{i=1}^{m} w_{i0}\left(\mathbf{X}_i^n - \mathbf{X}_0^n\right)}{\sum_{i=1}^{m} w_{i0}} \tag{5.15}$$

here, m is a number of nodes connected to node 0, $\mathbf{X}_0^n$ is a parametric position of node 0 at smoothing pass n, φ - relaxation parameter, w_{i0} is a weighted function between nodes i and 0.

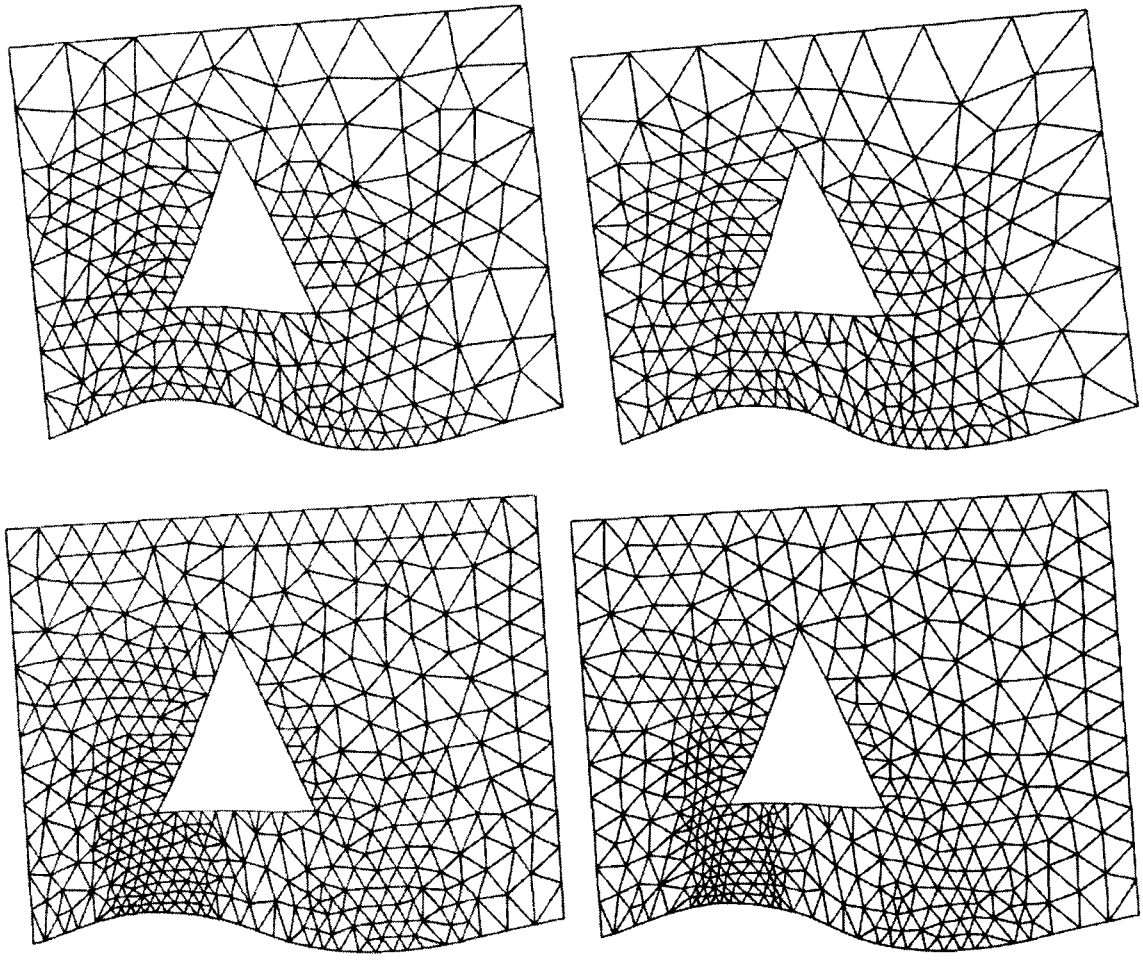


Figure 32. Initial triangulation (left). Smoothed triangulation (right).

5.6 Constraints Handling

Often structures, like vehicle panels, have internal constraints. These constraints can be points of connection for different structural or inertial components, for example seats connected to the floor panel. Also, specific features such as stiffeners can be attached on the plate for reinforcement. In such case, a stiffener is regarded as a line constraint that must be imposed on a mesh, though to simplify the meshing procedure only points of connection can be considered as the constraining points.

Constraints are static nodes on a mesh and they cannot be moved during smoothing or any other operation. For both TFI and advancing front methods the closest nodes in a

mesh are snapped to the constraint point. After that the whole mesh is smoothed to improve the quality of the elements that have been distorted during the constraints handling. For an unstructured mesh the constraints snapping is performed after triangle mesh generation and before quad conversion.

5.7 Hybrid Surface Meshing

Composite surface is meshed on patch basis. A patch without cutouts is meshed using algebraic mesher and a patch with cutouts is meshed using advancing front. Level of meshing depends on a number of key points on the boundaries. Synchronization is done in a way that boundaries are meshed prior to patch meshing. The algorithm iteratively created a data structure that stores boundaries sizing information for each patch in a composite surface.

5.8 Object-Oriented Design and Data Structures

Properly designed data structures are very important for efficient developing and functionality of software and can significantly speed up meshing. Part of the meshing functionality can be delegated to data structures and thus creating more efficient, readable and easy to debug code. The most important data structures used in the mesher can be outlined as follows:

- Node
- Edge
- Triangle element
- Quad element
- Bin data structure
- Quadtree structure

Good OOD is critical for creating a robust and efficient mesher. The meshing module has to be isolated from the rest of software to eliminate its dependence on software redesign or future extension. Optimal interaction between classes is important for creating a fast and reliable mesher. The following UML diagram (Figure 33) demonstrated the classes that have been used in the mesher, and interactions between them:

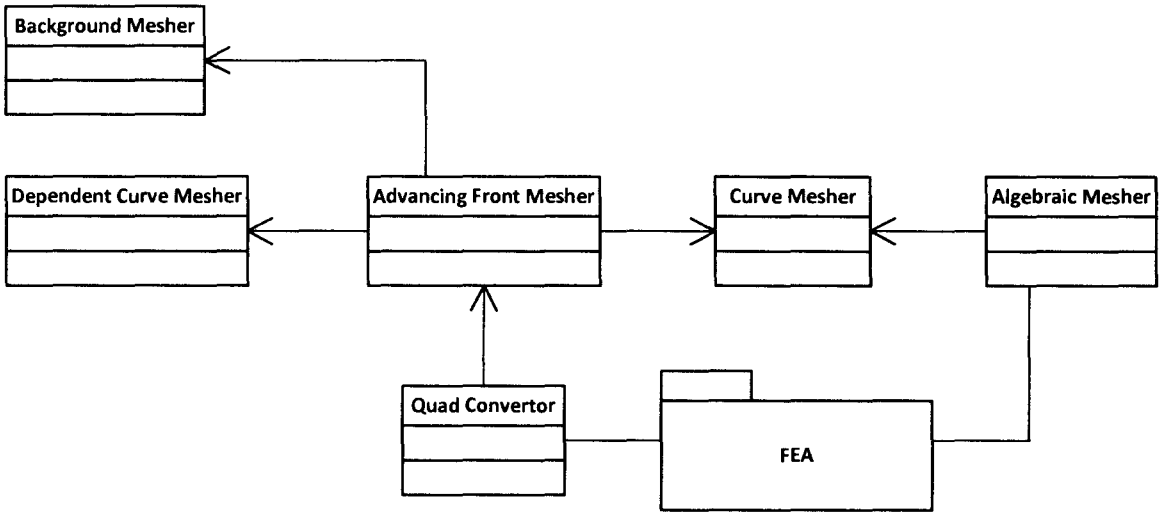


Figure 33. Meshing module UML class diagram.

CHAPTER 6

RIGID BODY SOLUTION ENGINE

This chapter gives a systematic description for the mathematical formulation of system dynamics, solution method and possible Object Oriented Design. Before going directly into the details, we will outline the main building blocks of the engine (Figure 34).

First, one of the simulation paradigms has to be chosen. Velocity-based approach is the optimal choice, combining a relatively simple set of equations, number of well elaborated solution methods, and a fast run time.

Second, the next step is to pick the right solver. It is well know that the formulation of the constrained rigid body problem is the same as the Mixed Linear Complementary Problem in the field of linear programming. There are a number of solution methods available for this problem, which can be divided in two groups of: exact and iterative algorithms. For practical reasons, a good decision would be to go with an iterative solver, since exact algorithms are much slower. The most popular of the iterative solvers is the Projected Gauss-Seidel scheme. Though, in practice we can often see its matrix-free mathematical analogy – Sequential Impulse method ("SI").

Third, scheme for the numerical integration of the equations of motion has to be chosen. Popular methods used in rigid body simulations are explicit, semi-implicit, implicit Euler and Verlet schemes. For this work the Verlet method has been chosen.

Last, for collision detection between wheels and terrain a Ray casting method will be used.

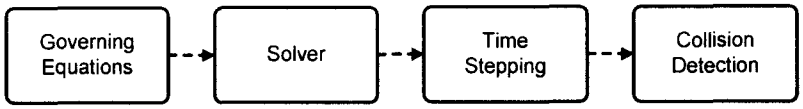


Figure 34. Main process blocks of the rigid body solution engine.

6.1 Equations of Motion

In three-dimensional space, a rigid body position can be determined by 6 coordinates. Three variables describe the position of the body's center of mass and 3 successive Euler angles. Classic Newton-Euler equations of motion of a single rigid body are as following:

$$\mathbf{m}\dot{\vec{v}} = \vec{f} \tag{6.1a}$$

$$\mathbf{I}\dot{\vec{\omega}} = \vec{\tau} - \vec{\omega} \times \mathbf{I}\vec{\omega} \tag{6.1b}$$

$\mathbf{m}$ and $\mathbf{I}$ – mass matrix and inertia tensor, $\vec{f}$ and $\vec{\tau}$ – vector of total force and torque acting on the center of mass, $\vec{v}$, $\vec{\omega}$ – vectors of linear and angular velocity of the center of mass.

To simplify the notation of the above equations we will write these in the following form:

$$\mathbf{M}\dot{\vec{\mathrm{v}}} = \vec{f}_e \tag{6.2}$$

Now $\mathbf{M} = \{\mathbf{m}, \mathbf{I}\}^T$, $\vec{\mathrm{v}} = \{\vec{v}, \vec{\omega}\}^T$ and $\vec{\mathrm{f}}_e = \left\{\vec{f}, \vec{\tau} - \vec{\omega} \times \mathbf{I}\vec{\omega}\right\}^T$.

Besides external forces, a rigid body is often subjected to the contact and joint reaction forces. These forces can be treated similarly:

$$\mathbf{M}\dot{\vec{\mathrm{v}}} = \vec{f}_e + \vec{f}_c + \vec{f}_j + \vec{f}_f \tag{6.3}$$

Contact, friction and joint forces are modeled using Lagrange multipliers:

$$\vec{f}_c = \mathbf{J}_c^T \vec{\lambda}_c \tag{6.4a}$$

$$\vec{f}_f = \mathbf{J}_f^T \vec{\lambda}_f \tag{6.4b}$$

$$\vec{f}_j = \mathbf{J}_j^T \vec{\lambda}_j \tag{6.4c}$$

$\mathbf{J}$, $\vec{\lambda}$ – Jacobian and vector of Lagrange multipliers correspondingly.

6.2 Contact and Friction Modeling

Having the following contact and friction condition

$$\mathbf{J}_c \vec{\mathrm{v}} \geq 0, \ \vec{\lambda}_c \geq 0 \tag{6.5a}$$

$$\mathbf{J}_f \vec{\mathrm{v}} \geq 0, \ \vec{\lambda}_f \geq 0 \tag{6.5b}$$

means that normal velocity of a body cannot be negative, thus we do not allow penetration. Friction force is limited by the normal contact reaction:

$$-\mu\vec{\lambda}_c \leq \vec{\lambda}_f \leq \mu\vec{\lambda}_c \tag{6.6}$$

The term $\mathbf{J}_c\vec{\mathrm{v}}$ can be considered as the projection of the body contact point velocity on the surface normal $\vec{n}$:

$$\mathbf{J}_c \vec{\mathrm{v}} = \vec{n} \cdot \left(\vec{v} + \vec{r}^{\,g} \times \vec{\omega} \right) \tag{6.7}$$

$\vec{r}^{\,g}$ – vector of the global moment arm of the body, connecting center of the mass of a body and its point of contact with a surface.

Analogously, friction Jacobians are shown as follows

$$\mathbf{J}_f^1 \vec{\mathrm{v}} = \vec{t}_1 \cdot \left(\vec{v} + \vec{r}^{\,g} \times \vec{\omega} \right) \tag{6.8a}$$

$$\mathbf{J}_f^2 \vec{\mathrm{v}} = \vec{t}_2 \cdot \left(\vec{v} + \vec{r}^{\,g} \times \vec{\omega} \right) \tag{6.8b}$$

$\vec{t}_1$ and $\vec{t}_2$ are two mutually perpendicular vectors in the tangent plane of the terrain contact point, such that $\vec{t}_1 \perp \vec{n}$ and $\vec{t}_2 \perp \vec{n}$.

Contact Jacobian in the matrix form can be represented as a row of dimensions 1×6. Then

$$\mathbf{J}_c = \left\{\vec{n},\ \vec{n}\times\vec{r}^{\,g}\right\}^T \tag{6.9}$$

And for friction

$$\mathbf{J}_f^1 = \left\{\vec{t}_1,\ \vec{t}_1\times\vec{r}^{\,g}\right\}^T \tag{6.10a}$$

$$\mathbf{J}_f^2 = \left\{\vec{t}_2,\ \vec{t}_2\times\vec{r}^{\,g}\right\}^T \tag{6.10b}$$

Now it will be demonstrated how to solve separately for contact and friction. Integrating the equation (6.2) from the time t_0 to t_1 and solving for the velocity at the moment t_1 we have:

$$\vec{v}_1 = \vec{v}_0 + \mathbf{M}^{-1}\mathbf{J}_c^T\vec{\lambda}_c \tag{6.11}$$

$$\vec{\lambda}_c \geq 0 \tag{6.12}$$

Substituting $\vec{v}_1$ into equation (6.5a) will give us the following:

$$\mathbf{J}_c\vec{v}_0 + \mathbf{J}_c\mathbf{M}^{-1}\mathbf{J}_c^T\vec{\lambda}_c = 0 \tag{6.13}$$

$$\mathbf{m}_c = \frac{1}{\mathbf{J}_c\mathbf{M}^{-1}\mathbf{J}_c^T} \tag{6.14}$$

$\mathbf{m}_c$ is sometimes called effective mass.

Assigning $\vec{s} = \vec{n}\times\vec{r}^{\,g}$, the term $\mathbf{J}_c\mathbf{M}^{-1}\mathbf{J}_c^T$ is equal to

$$\mathbf{J}_c\mathbf{M}^{-1}\mathbf{J}_c^T = \begin{bmatrix}\vec{n} & \vec{s}\end{bmatrix}\begin{bmatrix} m^{-1}\cdot \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}^{-1}\end{bmatrix}\begin{bmatrix}\vec{n} \\ \vec{s}\end{bmatrix} = \left[m^{-1}\cdot\vec{n}^2 + \vec{s}^T\cdot\mathbf{I}^{-1}\cdot\vec{s}\right] \tag{6.15}$$

$\boldsymbol{I}$ – 3×3 identity matrix. Effective mass is

$$\mathbf{m}_c = \frac{1}{\left[m^{-1}\cdot\vec{n}^2 + \vec{s}^T\cdot\mathbf{I}^{-1}\cdot\vec{s}\right]} \tag{6.16}$$

Lagrange multipliers that guaranty no penetration are calculated as

$$\vec{\lambda}_c = -\mathbf{m}_c\left(\mathbf{J}_c\vec{v}_0\right) \tag{6.17}$$

Similarly, in case of friction we have

$$\vec{v}_1 = \vec{v}_0 + \mathbf{M}^{-1}\mathbf{J}_f^T\vec{\lambda}_f \tag{6.18}$$

$$-\mu\vec{\lambda}_c \le \vec{\lambda}_f \le \mu\vec{\lambda}_c \tag{6.19}$$

$$\mathbf{J}_f\vec{v}_0 + \mathbf{J}_f\mathbf{M}^{-1}\mathbf{J}_f^T\vec{\lambda}_f = 0 \tag{6.20}$$

$$\mathbf{m}_f = \frac{1}{\mathbf{J}_f\mathbf{M}^{-1}\mathbf{J}_f^T} \tag{6.21}$$

$$\vec{\lambda}_f = -\mathbf{m}_f\left(\mathbf{J}_f\vec{v}_0\right) \tag{6.22}$$

Using the formula (6.10a) and introducing the notations $\vec{c}_1 = \vec{t}_1\times\vec{r}^{\,g}$ and $\vec{c}_2 = \vec{t}_2\times\vec{r}^{\,g}$ the term $\mathbf{J}_f\mathbf{M}^{-1}\mathbf{J}_f^T$ can be estimated as

$$\begin{aligned}\mathbf{J}_f\mathbf{M}^{-1}\mathbf{J}_f^T &= \begin{bmatrix}\vec{t}_1^T & \vec{c}_1^T \\ \vec{t}_2^T & \vec{c}_2^T\end{bmatrix}\begin{bmatrix} m^{-1}\cdot\boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}^{-1}\end{bmatrix}\begin{bmatrix}\vec{t}_1 & \vec{t}_2 \\ \vec{c}_1 & \vec{c}_2\end{bmatrix} \\ &= \begin{bmatrix} \left[m^{-1}\cdot\vec{t}_1^2 + \vec{c}_1^T\cdot\mathbf{I}^{-1}\cdot\vec{c}_1\right] & \left[m^{-1}\cdot\vec{t}_1^T\cdot\vec{t}_2 + \vec{c}_1^T\cdot\mathbf{I}^{-1}\cdot\vec{c}_2\right] \\ \left[m^{-1}\cdot\vec{t}_2^T\cdot\vec{t}_1 + \vec{c}_2^T\cdot\mathbf{I}^{-1}\cdot\vec{c}_1\right] & \left[m^{-1}\cdot\vec{t}_2^2 + \vec{c}_2^T\cdot\mathbf{I}^{-1}\cdot\vec{c}_2\right]\end{bmatrix}\end{aligned} \tag{6.23}$$

$$\mathbf{m}_f = \begin{bmatrix} \left[m^{-1}\cdot\vec{t}_1^{\,2} + \vec{c}_1^{\,T}\cdot\mathbf{I}^{-1}\cdot\vec{c}_1\right] & \left[m^{-1}\cdot\vec{t}_1^{\,T}\cdot\vec{t}_2 + \vec{c}_1^{\,T}\cdot\mathbf{I}^{-1}\cdot\vec{c}_2\right] \\ \left[m^{-1}\cdot\vec{t}_2^{\,T}\cdot\vec{t}_1 + \vec{c}_2^{\,T}\cdot\mathbf{I}^{-1}\cdot\vec{c}_1\right] & \left[m^{-1}\cdot\vec{t}_2^{\,2} + \vec{c}_2^{\,T}\cdot\mathbf{I}^{-1}\cdot\vec{c}_2\right] \end{bmatrix}^{-1} \tag{6.24}$$

One should notice that this formulation only approximates the Coulomb friction law, while the original formula is non-linear: $\lambda_{f1}{}^2 + \lambda_{f2}{}^2 \le (\mu\lambda_c)^2$. In practice also more accurate linearized models can be used, for example as in [91].

6.3 Impulse Solver

Wheel to terrain surface collision can be modeled using Newton impact law

$$v_a = -e\cdot v_b, \tag{6.25}$$

where v_b and v_a normal to the surface of contact velocities before and after collision. e – restitution coefficient. If $e=0$ we have fully plastic contact, $e=1$ – fully elastic contact.

To calculate normal velocity before the impact, the contact Jacobian can be used

$$v_b = \mathbf{J}_c\vec{\mathrm{v}} = \vec{n}\cdot\left(\vec{v} + \vec{r}^{\,g}\times\vec{\omega}\right) \tag{6.26}$$

$$\vec{v}_1 = \vec{v}_0 + \mathbf{M}^{-1}\mathbf{J}_c^T\vec{\lambda}_c \tag{6.27}$$

$$\mathbf{J}_c\vec{\mathrm{v}} \ge 0,\ \vec{\lambda}_c \ge 0 \tag{6.28}$$

$$\mathbf{J}_c\vec{v}_0 + \mathbf{J}_c\mathbf{M}^{-1}\mathbf{J}_c^T\vec{\lambda}_c = 0 \tag{6.29}$$

$$\mathbf{m}_c = \frac{1}{\mathbf{J}_c\mathbf{M}^{-1}\mathbf{J}_c^T} \tag{6.30}$$

Here $\mathbf{m}_c$ is the same as in the formula (6.16). Desired velocity difference is

$$b = v_a - v_b \tag{6.31}$$

Lagrange multipliers, that guaranty this jump of velocities, are calculated as

$$\vec{\lambda}_c = -\mathbf{m}_c \left(\mathbf{J}_c \vec{v}_0 + b \right) \tag{6.32}$$

Now substitution $\vec{\lambda}_c$ into equation (6.27) will give the velocity $\vec{\mathrm{v}}_1$ of a body after the impact.

6.4 Joint Modeling

Here the joint constraints equations will be described. The basic equation of the bilateral constraint is formulated as follows:

$$\mathbf{J}\vec{\mathrm{v}} = 0 \tag{6.33}$$

Let us introduce some definitions and notations:

- anchor point – the point of connection of two bodies;
- $\vec{r}^{\,l}$ – vector of the local moment arm of the body, connecting center of the mass and anchor point.
- $\vec{r}^{\,g}$ – vector of the global moment arm of the body.
- $\vec{r}^{\,g}$ and $\vec{r}^{\,l}$ are related by the transformation matrix $\mathbf{T}$ as $\vec{r}^{\,g} = \mathbf{T} \cdot \vec{r}^{\,l}$.

6.4.1 Hinge

Hinge joint allows only one degree of freedom, specifically rotation about an arbitrary axis l. Thus we have 5 equations constraining the motion of two bodies. Three of them describe a translational constraint (one vector equation for X, Y, Z degrees of freedom)

$$\vec{v}_1 + \vec{r}_1^{\,g} \times \vec{\omega}_1 - \vec{v}_2 - \vec{r}_2^{\,g} \times \vec{\omega}_2 = 0 \tag{6.34}$$

and the remaining two – prevent rotation

$$\vec{k}_1 \cdot \vec{\omega}_1 - \vec{k}_1 \cdot \vec{\omega}_2 = 0 \tag{6.35a}$$

$$\vec{k}_2 \cdot \vec{\omega}_1 - \vec{k}_2 \cdot \vec{\omega}_2 = 0 \tag{6.35b}$$

$\vec{v}_1$, $\vec{v}_2$, $\vec{\omega}_1$, $\vec{\omega}_2$ – linear and angular velocities of body 1 and 2. $\vec{k}_1$, $\vec{k}_2$ – orthogonal vectors, such that $\vec{k}_1 \perp \vec{l}$ and $\vec{k}_2 \perp \vec{l}$.

The whole Jacobian in matrix form

$$\mathbf{J}_h = \begin{bmatrix} \boldsymbol{I} & \left[\vec{r}_1^{\,g\times}\right] & -\boldsymbol{I} & -\left[\vec{r}_2^{\,g\times}\right] \\ \mathbf{0} & \vec{k}_1^{\,T} & \mathbf{0} & -\vec{k}_1^{\,T} \\ \mathbf{0} & \vec{k}_2^{\,T} & \mathbf{0} & -\vec{k}_2^{\,T} \end{bmatrix} \tag{6.36}$$

Let us consider the example of calculation for the effective mass $\mathbf{m}_h$ for a hinge model. Inverted mass matrix is:

$$\mathbf{M}^{-1} = \begin{bmatrix} m_1^{-1} \cdot \boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_1^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & m_2^{-1} \cdot \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_2^{-1} \end{bmatrix} \tag{6.37}$$

$$\begin{aligned} \mathbf{M}^{-1}\mathbf{J}_h^T &= \begin{bmatrix} m_1^{-1} \cdot \boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_1^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & m_2^{-1} \cdot \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_2^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \vec{0} & \vec{0} \\ \left[\vec{r}_1^{\,g\times}\right]^T & \vec{k}_1 & \vec{k}_2 \\ -\boldsymbol{I} & \vec{0} & \vec{0} \\ -\left[\vec{r}_2^{\,g\times}\right]^T & -\vec{k}_1 & -\vec{k}_2 \end{bmatrix} \\ &= \begin{bmatrix} m_1^{-1} \cdot \boldsymbol{I} & \vec{0} & \vec{0} \\ \mathbf{I}_1^{-1} \cdot \left[\vec{r}_1^{\,g\times}\right]^T & \mathbf{I}_1^{-1} \cdot \vec{k}_1 & \mathbf{I}_1^{-1} \cdot \vec{k}_2 \\ -m_2^{-1} \cdot \boldsymbol{I} & \vec{0} & \vec{0} \\ -\mathbf{I}_2^{-1} \cdot \left[\vec{r}_2^{\,g\times}\right]^T & -\mathbf{I}_2^{-1} \cdot \vec{k}_1 & -\mathbf{I}_2^{-1} \cdot \vec{k}_2 \end{bmatrix} \end{aligned} \tag{6.38}$$

$$\mathbf{J}_h\mathbf{M}^{-1}\mathbf{J}_h^T = \begin{bmatrix} \boldsymbol{I} & \left[\vec{r}_1^{\,g\times}\right] & -\boldsymbol{I} & -\left[\vec{r}_2^{\,g\times}\right] \\ \vec{0}^T & \vec{k}_1^T & \vec{0}^T & -\vec{k}_1^T \\ \vec{0}^T & \vec{k}_2^T & \vec{0}^T & -\vec{k}_2^T \end{bmatrix} \begin{bmatrix} m_1^{-1}\cdot\boldsymbol{I} & \vec{0} & \vec{0} \\ \mathbf{I}_1^{-1}\cdot\left[\vec{r}_1^{\,g\times}\right]^T & \mathbf{I}_1^{-1}\cdot\vec{k}_1 & \mathbf{I}_1^{-1}\cdot\vec{k}_2 \\ -m_2^{-1}\cdot\boldsymbol{I} & \vec{0} & \vec{0} \\ -\mathbf{I}_2^{-1}\cdot\left[\vec{r}_2^{\,g\times}\right]^T & -\mathbf{I}_2^{-1}\cdot\vec{k}_1 & -\mathbf{I}_2^{-1}\cdot\vec{k}_2 \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} \left(m_1^{-1}+m_2^{-1}\right)\cdot\boldsymbol{I} + \\ +\left[\vec{r}_1^{\,g\times}\right]\cdot\mathbf{I}_1^{-1}\cdot\left[\vec{r}_1^{\,g\times}\right]^T + \\ +\left[\vec{r}_2^{\,g\times}\right]\cdot\mathbf{I}_2^{-1}\cdot\left[\vec{r}_2^{\,g\times}\right]^T \end{bmatrix} & \begin{bmatrix} \left[\vec{r}_1^{\,g\times}\right]\cdot\mathbf{I}_1^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right]+ \\ +\left[\vec{r}_2^{\,g\times}\right]\cdot\mathbf{I}_2^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right] \end{bmatrix} \\ \begin{bmatrix} \begin{bmatrix}\vec{k}_1^T \\ \vec{k}_2^T\end{bmatrix}\cdot\mathbf{I}_1^{-1}\cdot\left[\vec{r}_1^{\,g\times}\right]^T + \\ +\begin{bmatrix}\vec{k}_1^T \\ \vec{k}_2^T\end{bmatrix}\cdot\mathbf{I}_2^{-1}\cdot\left[\vec{r}_2^{\,g\times}\right]^T \end{bmatrix} & \begin{bmatrix} \begin{bmatrix}\vec{k}_1^T \\ \vec{k}_2^T\end{bmatrix}\cdot\mathbf{I}_1^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right]+ \\ +\begin{bmatrix}\vec{k}_1^T \\ \vec{k}_2^T\end{bmatrix}\cdot\mathbf{I}_2^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right] \end{bmatrix} \end{bmatrix} \tag{6.39}$$

Now the effective mass for a hinge can be calculated by inverting the matrix $\mathbf{J}_h\mathbf{M}^{-1}\mathbf{J}_h^T$:

$$\mathbf{m}_h = \begin{bmatrix} \begin{bmatrix} \left(m_1^{-1}+m_2^{-1}\right)\cdot\boldsymbol{I} + \\ +\left[\vec{r}_1^{\,g\times}\right]\cdot\mathbf{I}_1^{-1}\cdot\left[\vec{r}_1^{\,g\times}\right]^T + \\ +\left[\vec{r}_2^{\,g\times}\right]\cdot\mathbf{I}_2^{-1}\cdot\left[\vec{r}_2^{\,g\times}\right]^T \end{bmatrix} & \begin{bmatrix} \left[\vec{r}_1^{\,g\times}\right]\cdot\mathbf{I}_1^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right]+ \\ +\left[\vec{r}_2^{\,g\times}\right]\cdot\mathbf{I}_2^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right] \end{bmatrix} \\ \begin{bmatrix} \begin{bmatrix}\vec{k}_1^T \\ \vec{k}_2^T\end{bmatrix}\cdot\mathbf{I}_1^{-1}\cdot\left[\vec{r}_1^{\,g\times}\right]^T + \\ +\begin{bmatrix}\vec{k}_1^T \\ \vec{k}_2^T\end{bmatrix}\cdot\mathbf{I}_2^{-1}\cdot\left[\vec{r}_2^{\,g\times}\right]^T \end{bmatrix} & \begin{bmatrix} \begin{bmatrix}\vec{k}_1^T \\ \vec{k}_2^T\end{bmatrix}\cdot\mathbf{I}_1^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right]+ \\ +\begin{bmatrix}\vec{k}_1^T \\ \vec{k}_2^T\end{bmatrix}\cdot\mathbf{I}_2^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right] \end{bmatrix} \end{bmatrix}^{-1} \tag{6.40}$$

Having effective mass the contribution to velocity from the hinge constraint can be estimated using the following formula

$$\vec{v}_1 = \vec{v}_0 + \mathbf{M}^{-1}\mathbf{J}_h^T\vec{\lambda}_h \tag{6.41}$$

$$\mathbf{M}^{-1}\mathbf{J}_h^T\lambda_h = \begin{bmatrix} m_1^{-1}\cdot\boldsymbol{I} & \vec{0} & \vec{0} \\ \mathbf{I}_1^{-1}\cdot\left[\vec{r}_1^{\,g\times}\right]^T & \mathbf{I}_1^{-1}\cdot\vec{k}_1 & \mathbf{I}_1^{-1}\cdot\vec{k}_2 \\ -m_2^{-1}\cdot\boldsymbol{I} & \vec{0} & \vec{0} \\ -\mathbf{I}_2^{-1}\cdot\left[\vec{r}_2^{\,g\times}\right]^T & -\mathbf{I}_2^{-1}\cdot\vec{k}_1 & -\mathbf{I}_2^{-1}\cdot\vec{k}_2 \end{bmatrix}\begin{bmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\\ \lambda_4\\ \lambda_5\end{bmatrix}$$

$$= \begin{bmatrix} m_1^{-1}\cdot\begin{bmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{bmatrix} \\ \left[\mathbf{I}_1^{-1}\cdot\left[\vec{r}_1^{\,g\times}\right]^T\cdot\begin{bmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{bmatrix} + \mathbf{I}_1^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right]\begin{bmatrix}\lambda_4\\ \lambda_5\end{bmatrix}\right] \\ m_2^{-1}\cdot\begin{bmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{bmatrix} \\ \left[-\mathbf{I}_2^{-1}\cdot\left[\vec{r}_2^{\,g\times}\right]^T\cdot\begin{bmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{bmatrix} - \mathbf{I}_2^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right]\begin{bmatrix}\lambda_4\\ \lambda_5\end{bmatrix}\right] \end{bmatrix} \tag{6.42}$$

6.4.2 Ball Joint

A ball joint is also sometime called spherical joint. The definition of the joint is easy to derive from the definition of the hinge joint, by simply removing the last two equations that constrain rotational degrees of freedom. Therefore, the set of equations are analogical to (6.34).

$$\vec{v}_1 + \vec{r}_1^{\,g}\times\vec{\omega}_1 - \vec{v}_2 - \vec{r}_2^{\,g}\times\vec{\omega}_2 = 0 \tag{6.43}$$

In the matrix form

$$\mathbf{J}_b = \begin{bmatrix} \boldsymbol{I} & \left[\vec{r}_1^{\,g\times}\right] & -\boldsymbol{I} & -\left[\vec{r}_1^{\,g\times}\right] \end{bmatrix} \tag{6.44}$$

$$\mathbf{M}^{-1}\mathbf{J}_b^T = \begin{bmatrix} m_1^{-1}\cdot\boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_1^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & m_2^{-1}\cdot\boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_2^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} \\ \left[\vec{r}_1^{\,g\times}\right]^T \\ -\boldsymbol{I} \\ -\left[\vec{r}_2^{\,g\times}\right]^T \end{bmatrix}$$
$$= \begin{bmatrix} m_1^{-1}\cdot\boldsymbol{I} \\ \mathbf{I}_1^{-1}\cdot\left[\vec{r}_1^{\,g\times}\right]^T \\ -m_2^{-1}\cdot\boldsymbol{I} \\ -\mathbf{I}_2^{-1}\cdot\left[\vec{r}_2^{\,g\times}\right]^T \end{bmatrix} \tag{6.45}$$

$$\mathbf{J}_b\mathbf{M}^{-1}\mathbf{J}_b^T = \begin{bmatrix} \boldsymbol{I} & \left[\vec{r}_1^{\,g\times}\right] & -\boldsymbol{I} & -\left[\vec{r}_1^{\,g\times}\right] \end{bmatrix} \begin{bmatrix} m_1^{-1}\cdot\boldsymbol{I} \\ \mathbf{I}_1^{-1}\cdot\left[\vec{r}_1^{\,g\times}\right]^T \\ -m_2^{-1}\cdot\boldsymbol{I} \\ -\mathbf{I}_2^{-1}\cdot\left[\vec{r}_2^{\,g\times}\right]^T \end{bmatrix}$$
$$= \begin{bmatrix} \left(m_1^{-1}+m_2^{-1}\right)\cdot\boldsymbol{I} + \\ +\left[\vec{r}_1^{\,g\times}\right]\cdot\mathbf{I}_1^{-1}\cdot\left[\vec{r}_1^{\,g\times}\right]^T + \\ +\left[\vec{r}_2^{\,g\times}\right]\cdot\mathbf{I}_2^{-1}\cdot\left[\vec{r}_2^{\,g\times}\right]^T \end{bmatrix} \tag{6.46}$$

Now effective mass for the ball joint can be calculated by inverting the matrix $\mathbf{J}_b\mathbf{M}^{-1}\mathbf{J}_b^T$:

$$\mathbf{m}_b = \begin{bmatrix} \left(m_1^{-1}+m_2^{-1}\right)\cdot\boldsymbol{I} + \\ +\left[\vec{r}_1^{\,g\times}\right]\cdot\mathbf{I}_1^{-1}\cdot\left[\vec{r}_1^{\,g\times}\right]^T + \\ +\left[\vec{r}_2^{\,g\times}\right]\cdot\mathbf{I}_2^{-1}\cdot\left[\vec{r}_2^{\,g\times}\right]^T \end{bmatrix}^{-1} \tag{6.47}$$

Having the effective mass the contribution to velocity from spherical constraint can be estimated using the following formula

$$\vec{v}_1 = \vec{v}_0 + \mathbf{M}^{-1}\mathbf{J}_b^T\vec{\lambda}_b \tag{6.48}$$

$$\mathbf{M}^{-1}\mathbf{J}_b^T\lambda_b = \begin{bmatrix} m_1^{-1}\cdot \boldsymbol{I} \\ \mathbf{I}_1^{-1}\cdot\left[\vec{r}_1^{\,g\times}\right]^T \\ -m_2^{-1}\cdot \boldsymbol{I} \\ -\mathbf{I}_2^{-1}\cdot\left[\vec{r}_2^{\,g\times}\right]^T \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

$$= \begin{bmatrix} m_1^{-1}\cdot \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} \\ \mathbf{I}_1^{-1}\cdot\left[\vec{r}_1^{\,g\times}\right]^T \cdot \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} \\ m_2^{-1}\cdot \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} \\ -\mathbf{I}_2^{-1}\cdot\left[\vec{r}_2^{\,g\times}\right]^T \cdot \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} \end{bmatrix} \tag{6.49}$$

6.4.3 Slider

A slider joint allows only one translational degree of freedom. The joint equations can be formulated as following

$$\vec{\omega}_1 - \vec{\omega}_2 = 0 \tag{6.50}$$

$$\vec{k}_1\cdot\vec{v}_1 + \frac{1}{2}\left(\vec{c}\times\vec{k}_1\right)\cdot\vec{\omega}_1 - \vec{k}_1\cdot\vec{v}_2 + \frac{1}{2}\left(\vec{c}\times\vec{k}_1\right)\cdot\vec{\omega}_2 = 0 \tag{6.51a}$$

$$\vec{k}_2\cdot\vec{v}_1 + \frac{1}{2}\left(\vec{c}\times\vec{k}_2\right)\cdot\vec{\omega}_1 - \vec{k}_2\cdot\vec{v}_2 + \frac{1}{2}\left(\vec{c}\times\vec{k}_2\right)\cdot\vec{\omega}_2 = 0 \tag{6.51b}$$

Where $\vec{c} = \vec{r}_2 - \vec{r}_1$. The first vector equation makes sure there is no relative rotation between body 1 and 2. Last 2 equations guaranty relative sliding only in the direction of axis $\vec{l}$.

Jacobian in the matrix form

$$\mathbf{J}_s = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} & \mathbf{0} & -\boldsymbol{I} \\ \vec{k}_1^{\,T} & \vec{p}_1^{\,T} & -\vec{k}_1^{\,T} & \vec{p}_1^{\,T} \\ \vec{k}_2^{\,T} & \vec{p}_2^{\,T} & -\vec{k}_2^{\,T} & \vec{p}_2^{\,T} \end{bmatrix} \tag{6.52}$$

here $\vec{p}_1 = \frac{1}{2}\left(\vec{c} \times \vec{k}_1\right)$ and $\vec{p}_2 = \frac{1}{2}\left(\vec{c} \times \vec{k}_2\right)$.

$$\begin{aligned} \mathbf{M}^{-1}\mathbf{J}_s^T &= \begin{bmatrix} m_1^{-1} \cdot \boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_1^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & m_2^{-1} \cdot \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_2^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \vec{k}_1 & \vec{k}_2 \\ \boldsymbol{I} & \vec{p}_1 & \vec{p}_2 \\ \mathbf{0} & -\vec{k}_1 & -\vec{k}_2 \\ -\boldsymbol{I} & \vec{p}_1 & \vec{p}_2 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0} & m_1^{-1} \cdot \left[\vec{k}_1 \quad \vec{k}_2\right] \\ \mathbf{I}_1^{-1} & \mathbf{I}_1^{-1} \cdot \left[\vec{p}_1 \quad \vec{p}_2\right] \\ \mathbf{0} & -m_2^{-1} \cdot \left[\vec{k}_1 \quad \vec{k}_2\right] \\ -\mathbf{I}_2^{-1} & \mathbf{I}_2^{-1} \cdot \left[\vec{p}_1 \quad \vec{p}_2\right] \end{bmatrix} \end{aligned} \tag{6.53}$$

$$\begin{aligned} \mathbf{J}_s\mathbf{M}^{-1}\mathbf{J}_s^T &= \begin{bmatrix} \mathbf{0} & \boldsymbol{I} & \mathbf{0} & -\boldsymbol{I} \\ \vec{k}_1^{\,T} & \vec{p}_1^{\,T} & -\vec{k}_1^{\,T} & \vec{p}_1^{\,T} \\ \vec{k}_2^{\,T} & \vec{p}_2^{\,T} & -\vec{k}_2^{\,T} & \vec{p}_2^{\,T} \end{bmatrix} \begin{bmatrix} \mathbf{0} & m_1^{-1} \cdot \left[\vec{k}_1 \quad \vec{k}_2\right] \\ \mathbf{I}_1^{-1} & \mathbf{I}_1^{-1} \cdot \left[\vec{p}_1 \quad \vec{p}_2\right] \\ \mathbf{0} & -m_2^{-1} \cdot \left[\vec{k}_1 \quad \vec{k}_2\right] \\ -\mathbf{I}_2^{-1} & \mathbf{I}_2^{-1} \cdot \left[\vec{p}_1 \quad \vec{p}_2\right] \end{bmatrix} \\ &= \begin{bmatrix} \left[\mathbf{I}_1^{-1} + \mathbf{I}_2^{-1}\right] & \left[\left(\mathbf{I}_1^{-1} - \mathbf{I}_2^{-1}\right) \cdot \left[\vec{p}_1 \quad \vec{p}_2\right]\right] \\ \left[\begin{bmatrix} \vec{p}_1^{\,T} \\ \vec{p}_2^{\,T} \end{bmatrix} \cdot \left(\mathbf{I}_1^{-1} - \mathbf{I}_2^{-1}\right)\right] & \left[\begin{array}{l} \left(m_1^{-1} + m_2^{-1}\right) \cdot \begin{bmatrix} \vec{k}_1^{\,T} \\ \vec{k}_2^{\,T} \end{bmatrix} \cdot \left[\vec{k}_1 \quad \vec{k}_2\right] + \\ + \begin{bmatrix} \vec{p}_1^{\,T} \\ \vec{p}_2^{\,T} \end{bmatrix} \cdot \left(\mathbf{I}_1^{-1} + \mathbf{I}_2^{-1}\right) \cdot \left[\vec{p}_1 \quad \vec{p}_2\right] \end{array}\right] \end{bmatrix} \end{aligned} \tag{6.54}$$

Now effective mass for a slider can be calculated by inverting the matrix $\mathbf{J}_s\mathbf{M}^{-1}\mathbf{J}_s^T$:

$$\mathbf{m}_s = \begin{bmatrix} \left[\mathbf{I}_1^{-1} + \mathbf{I}_2^{-1}\right] & \left[\left(\mathbf{I}_1^{-1} - \mathbf{I}_2^{-1}\right)\cdot\left[\vec{p}_1 \quad \vec{p}_2\right]\right] \\ \left[\begin{bmatrix}\vec{p}_1^T \\ \vec{p}_2^T\end{bmatrix}\cdot\left(\mathbf{I}_1^{-1} - \mathbf{I}_2^{-1}\right)\right] & \left[\begin{array}{l}\left(m_1^{-1} + m_2^{-1}\right)\cdot\begin{bmatrix}\vec{k}_1^T \\ \vec{k}_2^T\end{bmatrix}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right] + \\ + \begin{bmatrix}\vec{p}_1^T \\ \vec{p}_2^T\end{bmatrix}\cdot\left(\mathbf{I}_1^{-1} + \mathbf{I}_2^{-1}\right)\cdot\left[\vec{p}_1 \quad \vec{p}_2\right]\end{array}\right] \end{bmatrix}^{-1} \tag{6.55}$$

Having effective mass the contribution to velocity from slider constraint can be estimated using the following formula

$$\vec{v}_1 = \vec{v}_0 + \mathbf{M}^{-1}\mathbf{J}_s^T\vec{\lambda}_s \tag{6.56}$$

$$\begin{aligned}\mathbf{M}^{-1}\mathbf{J}_s^T\lambda_s &= \begin{bmatrix} \mathbf{0} & m_1^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right] \\ \mathbf{I}_1^{-1} & \mathbf{I}_1^{-1}\cdot\left[\vec{p}_1 \quad \vec{p}_2\right] \\ \mathbf{0} & -m_2^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right] \\ -\mathbf{I}_2^{-1} & \mathbf{I}_2^{-1}\cdot\left[\vec{p}_1 \quad \vec{p}_2\right] \end{bmatrix}\begin{bmatrix}\lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5\end{bmatrix} \\ &= \begin{bmatrix} m_1^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right]\begin{bmatrix}\lambda_4 \\ \lambda_5\end{bmatrix} \\ \left[\mathbf{I}_1^{-1}\cdot\begin{bmatrix}\lambda_1 \\ \lambda_2 \\ \lambda_3\end{bmatrix} + \mathbf{I}_1^{-1}\cdot\left[\vec{p}_1 \quad \vec{p}_2\right]\begin{bmatrix}\lambda_4 \\ \lambda_5\end{bmatrix}\right] \\ -m_2^{-1}\cdot\left[\vec{k}_1 \quad \vec{k}_2\right]\begin{bmatrix}\lambda_4 \\ \lambda_5\end{bmatrix} \\ \left[-\mathbf{I}_2^{-1}\cdot\begin{bmatrix}\lambda_1 \\ \lambda_2 \\ \lambda_3\end{bmatrix} + \mathbf{I}_2^{-1}\cdot\left[\vec{p}_1 \quad \vec{p}_2\right]\begin{bmatrix}\lambda_4 \\ \lambda_5\end{bmatrix}\right] \end{bmatrix}\end{aligned} \tag{6.57}$$

6.4.4 Rigid Joint

Rigid body engine itself does not implement a rigid type of constraint. Instead, we combine any two rigidly connected bodies, creating one body. The algorithm loops through all rigid edges in the graph and combines bodies until no rigid edges are left. For

a newly created body the center of mass and inertial tensor are calculated. The algorithm runs during the preprocessing stage, thus time intensive constraint solving is not needed for rigid joints.

6.5 Mixed Linear Complementary Problem Formulation

In short, the system of equations can be formulated as following

$$\mathbf{M}(\vec{v}_1 - \vec{v}_0) = \mathbf{J}_u^T \vec{\lambda}_u + \mathbf{J}_b^T \vec{\lambda}_b + \vec{f}_e \tag{6.58}$$

$$\mathbf{J}_b \vec{v} = 0, \; -\infty \geq \vec{\lambda}_b \geq +\infty \tag{6.59}$$

$$\mathbf{J}_u \vec{v} \geq 0, \; \vec{\lambda}_u \geq 0 \tag{6.60}$$

Where $\mathbf{J}_b$ – Jacobian that represents bilateral constraints, $\mathbf{J}_u$ – represents unilateral constraints.

$$\vec{v}_1 = \vec{v}_0 + \mathbf{M}^{-1}\mathbf{J}_u^T \vec{\lambda}_u + \mathbf{M}^{-1}\mathbf{J}_b^T \vec{\lambda}_b + \mathbf{M}^{-1}\vec{f}_e \tag{6.61}$$

Satisfying constraint equations (6.59) and (6.60) we have

$$\mathbf{J}_b \vec{v}_0 + \mathbf{J}_b \mathbf{M}^{-1}\mathbf{J}_u^T \vec{\lambda}_u + \mathbf{J}_b \mathbf{M}^{-1}\mathbf{J}_b^T \vec{\lambda}_b + \mathbf{J}_b \mathbf{M}^{-1}\vec{f}_e = 0 \tag{6.62a}$$

$$\mathbf{J}_u \vec{v}_0 + \mathbf{J}_u \mathbf{M}^{-1}\mathbf{J}_u^T \vec{\lambda}_u + \mathbf{J}_u \mathbf{M}^{-1}\mathbf{J}_b^T \vec{\lambda}_b + \mathbf{J}_u \mathbf{M}^{-1}\vec{f}_e \geq 0 \tag{6.62b}$$

Generally speaking, one can solve the system for the $\vec{\lambda}_u$ and $\vec{\lambda}_b$, and update $\vec{v}_1$ in (6.61), although a more elegant way of solving the system will be demonstrated later.

6.6 Time Stepping Scheme

Implicit integration provides computational saving by allowing a large time step, but on the other hand it results in a loss of accuracy and higher frequency response of a system. The visible result is overly damped behavior of rigid bodies. Accuracy is especially important for mechanical simulation of vehicle design, while it can be ignored

for animation purposes. It is possible to decrease the time step in an implicit method so it approaches the size required for stability of an explicit method and for preservation of high frequency response. But this will make an implicit method unusable, since an implicit method is complex in implementation. For reason of accuracy and simplicity we prefer explicit methods.

The Verlet method is a conditionally stable [155] second order method with the computational efficiency of a first order method. The Courant stability condition has to be satisfied in order for the solution not to diverge:

$$\Delta t \leq \frac{2}{\omega_d}, \tag{6.63}$$

with ω_d being the highest natural frequency of the system. For an undamped single spring:

$$\omega_d = \sqrt{\frac{k}{m}}, \tag{6.64}$$

and for springs with dampers

$$\omega_d = \sqrt{\left(\frac{k}{m}\right)^2 - \left(\frac{c}{2m}\right)^2} \tag{6.65}$$

A few papers have described the application of velocity Verlet time stepping schemes [103, 132, 155].

6.7 Collision Detection and TOI Solver

For the conceptual modeling the simplified collision detection has been used. Only collision of wheels with the terrain surface is considered. Ignoring the collision detection between the terrain and other bodies in the vehicle system reduces considerably the run time of the engine. We used the Raycasting against a Minkowski sum approach [157,

158]. Radius of a wheel is added to the radius of the bigger object, in our case it is terrain. Now the problem is much simple: to find distance to the terrain.

A more complicated problem is to find the time of impact (TOI). Let us describe the contact situation. In the beginning of a time step there was no contact, and in the end of a time step at time Δt there is a penetration. The algorithm has to go back to the safe position at time $0 \le t \le \Delta t$ where no contact is occurring (in practice we will allow some threshold for the penetration). This algorithm is sometimes called conservative advancing, since it iteratively comes closer to the contact until a certain threshold is reached.

Let us consider one contact point with position *initialPosition* at the beginning of a time step and position *currentPosition* at the end of the time step. The following pseudo-code describes the TOI calculation:

GetTimeOfImpactFactor(*initialPosition*, *currentPosition*)

{

$directionVector := currentPosition - initialPosition;$

$direction :=$ Normalize($directionVector$);

$contactPosition :=$ RecursiveApproximation($initialPosition$);

$fullDistance :=$ Distance($initialPosition, currentPosition$);

$newDistance :=$ Distance($initialPosition, contactPosition$);

$$factor := \frac{newDistance}{fullDistance};$$

return *factor*;

}

```
RecursiveApproximation(initialPosition)
{
    distance := CalculateClosestDistanceToTerrain(initialPosition);
    if (distance < threshold) return initialPosition;
    else
    {
        initialPosition := distance · direction;
        return RecursiveApproximation(initialPosition);
    }
}
```

The TOI factor f is a real number such that $0 \le f \le 1$. In practice we have to go through all contacts in a list and find the smallest factor f_m. Then the TOI can be calculated as

$$t = f_m \cdot \Delta t \tag{6.66}$$

6.8 Object-Oriented Design and Data Flow for the Rigid Body Engine

The rigid body analysis engine is implemented as a module of a larger suite of analysis tools, and it follows design patterns common to all analyses included in the package. The UML class diagram of Figure 35 depicts this shared structure. Analysis, Engine and PostProcessor are all abstract classes, and interaction between them are also implemented on the abstract level through their interfaces. This allows us to implement a simulation framework that is independent of the underlying simulation method, and where strategy of the implementation can be easily changed by switching to the different concrete class.

In addition to being capable of rapid solution times, a viable analysis suite should be capable of preparing architecture analysis models for full vehicles, a primary subsystem (a cab or rolling chassis, for example), and in some cases singular features or components. It must have access to visual renderings that depict the vehicle architecture's physical layout while allowing a designer to interact with both the physical design space and analysis results. Finally, individual modules of the analysis suite should be algorithmically integrated.

Rigid body analysis can be based upon discrete system models with lumped mass/inertia, damping, and stiffness elements. Mathematically, the model will consist of coupled ordinary differential and algebraic equations. Complementary analyses for such models include determination of rigid body natural frequencies characteristics, and time domain response.

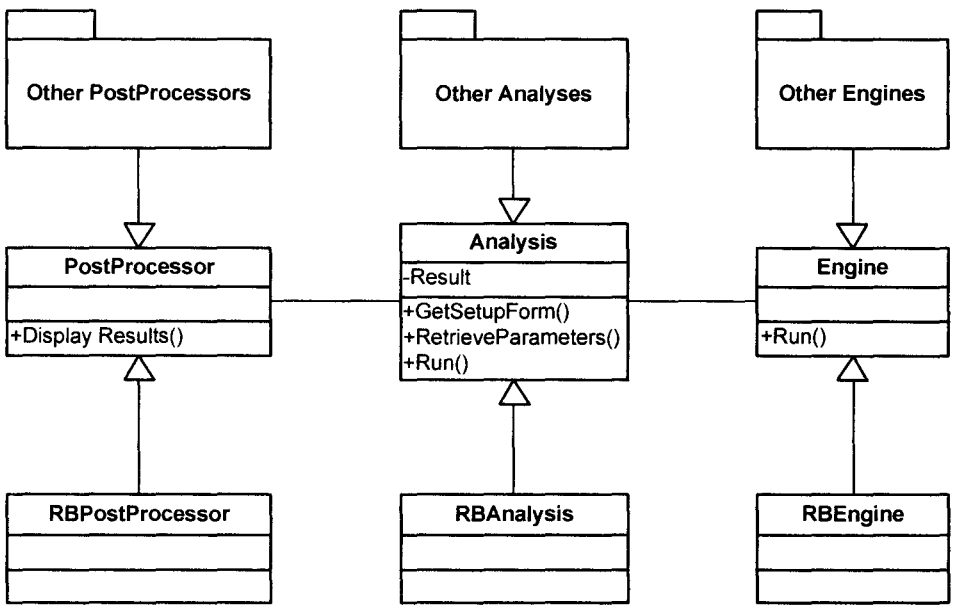


Figure 35. Rigid body module structure.

Many common joint types, including spring, damper, ball-joint, hinge, motor, slider, and limit, are supported by the engine. Like the vehicle abstraction, the resulting

rigid body model is stored in an undirected graph data structure, with each node of the graph representing a rigid component or assembly (class RigidBody), and edges representing component connections (class Edge). Design of the RB Graph is shown on the Figure 36.

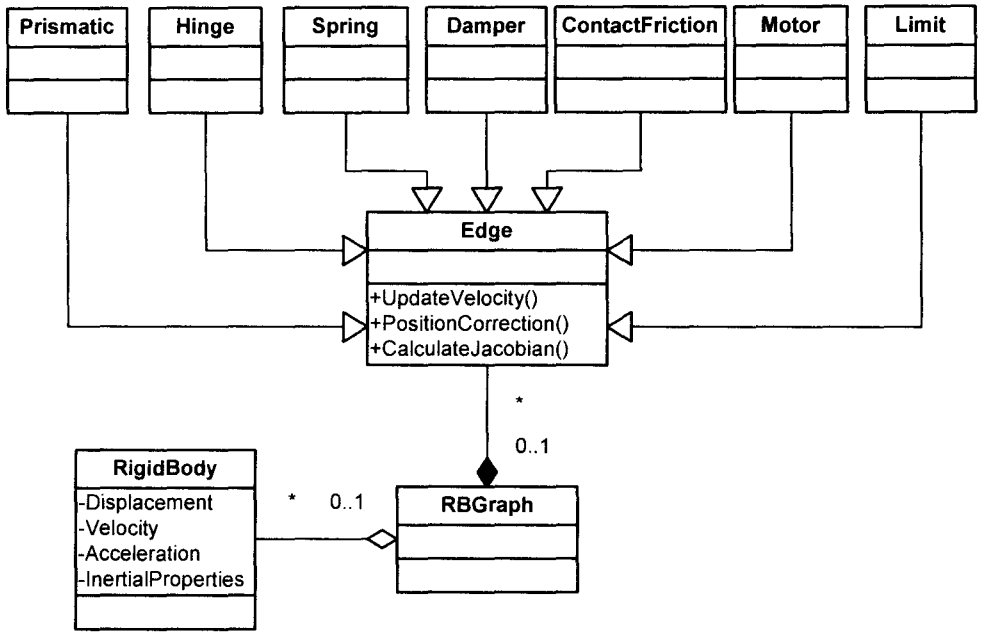


Figure 36. UML diagram of the RBGraph class.

The entire rigid body module consists of the three main classes: RBAnalysis, RBEngine and RBPostProcessor. RBAnalysis is the class responsible for the interaction of the RB module with the rest of the software. Inside RBAnalysis we store input parameters and results of the analysis. The class is also responsible for constructing the rigid body graph and combining any rigidly connected components. RBEngine is the actual multibody system dynamics engine that controls all the subclasses responsible for the multibody system dynamics mathematical model calculations. This class defines the interaction between its subclasses (Figure 37).

RBPostProcessor is the class responsible for the visual representation of results. The simulation results are normally depicted as an animation of the vehicle traversing the input terrain profile. They can also be shown as plots presenting the position, velocity and acceleration of a specified system body as a function of time, or as a state rendering, which represents a rendering of the analyzed vehicle at a given moment in time.

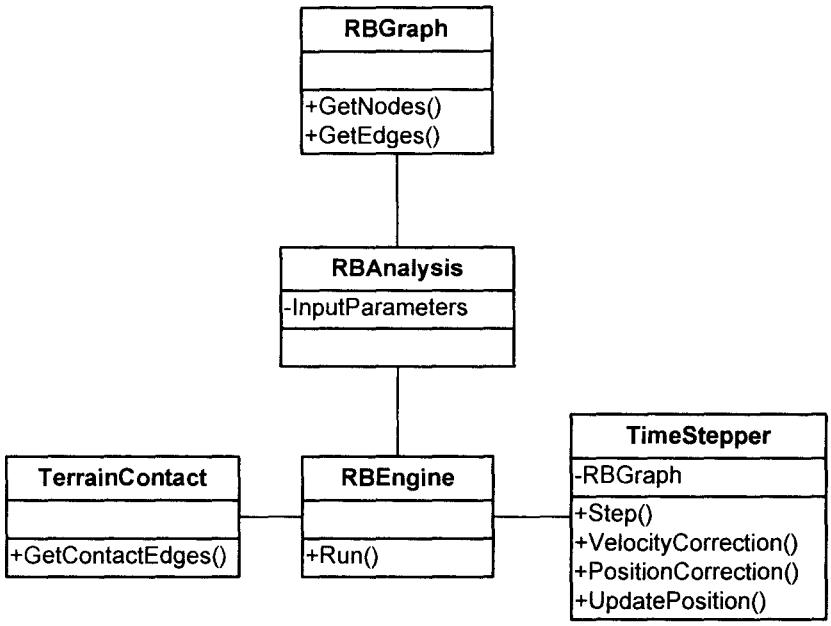


Figure 37. UML diagram of RB Engine interaction with other classes.

Another important class seen in Figure 37 is the TimeStepper, which provides the algorithmic functionality needed to advance numerical integration in time. The class is responsible only for the data flow within one time step (Figure 38).

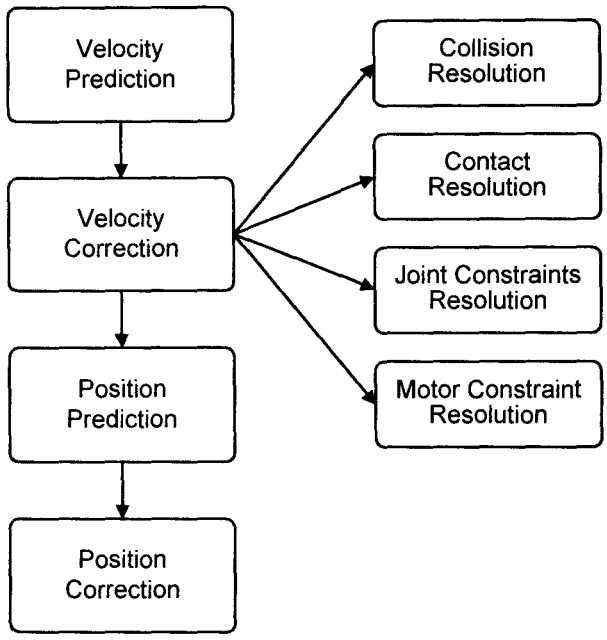


Figure 38. Data flow in the TimeStepper class.

For each time step the algorithm performs checks to see if contact exists between individual wheels and the terrain. Collision detection and time of impact solution methods are implemented in the class TerrainContact. Any such contacts found are stored in the RBGraph as ContactFriction type of edges.

It is often useful to understand the rigid body modal characteristics of a vehicle architecture design. Knowing the system natural frequencies, damping ratios, and response patterns allows the designer to optimize the system behavior by tuning spring and damper characteristic properties. Accordingly, a frequency analysis has been implemented as a part of the rigid body analysis engine (Figure 39).

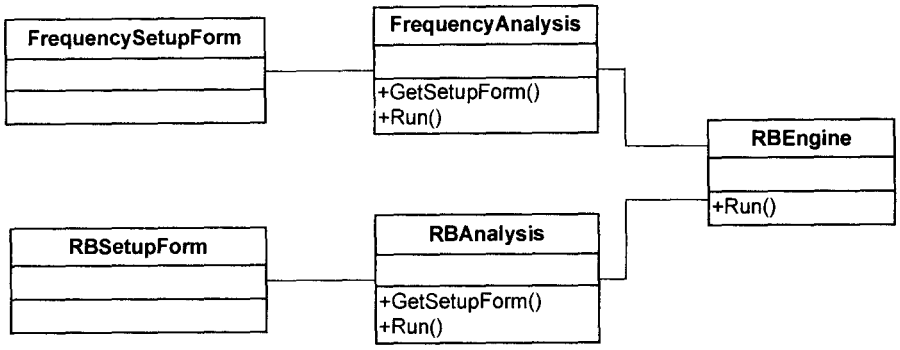


Figure 39. Relation of frequency analysis to rigid body analysis.

6.9 Results and Discussion

The fundamental result from the rigid body analysis module is the kinematic response of the rigid bodies making up the system. Specifically, the simulation yields translational and rotational velocities, positions, and accelerations for the cab, frame, and suspension mass centers. These results can be displayed as time domain response plots (cab velocity magnitude versus time, for example), or represented simultaneously as an animated rendering of the entire vehicle. The response of important non-centroidal points can also be calculated and displayed, as can connection forces. The instantaneous response can be compared with quasi-static responses to emphasize the effects of system compliance and contact. Vehicles can be excited by specifying a torque-speed relationship from the powertrain. Transient or steady-state response can be evaluated. In addition to time-dependent powertrain inputs, the vehicle models can be excited through terrain profile inputs.

To demonstrate the rigid body analysis results we have built concept models for two different vehicles. The first of these models has architecture typical of a Class 3[1] (3,850 - 4,540 kg GVWR) light-duty commercial truck: four-wheel ladder frame, engine-forward

[1] US Department of Transportation Federal Highway Administration (FHWA) commercial truck classifications based on the vehicle's gross vehicle weight rating (GVWR).

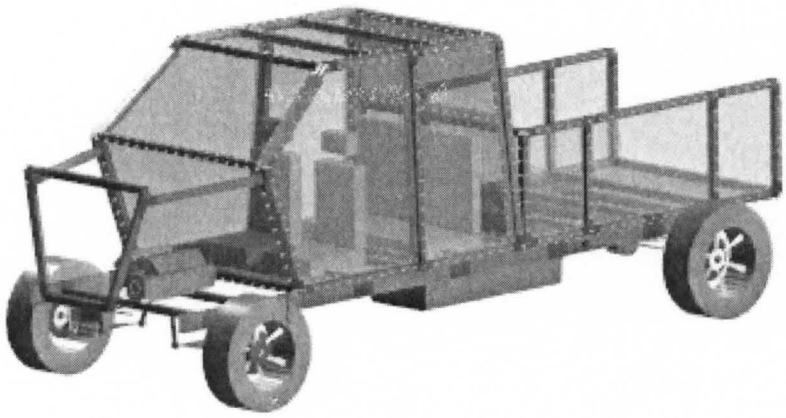
crew cab and open cargo box (Figure 40a). The second model represents the architecture of a Class 7 (11,800 - 14,970 kg GVWR) heavy-duty commercial truck with three axles (two driven), a cab-over-engine layout, and no payload module (Figure 40b).[2]

For the light-duty truck we used Macpherson struts for the front suspension and dependent swing arm as a rear suspension[3] (Figure 40a). Suspension compliance is controlled by tuning the spring stiffness coefficient and damping coefficient. The cab is modeled as single rigid body with rigidly attached seats. It is connected to the frame by means of springs, dampers and prismatic joints. The cargo box is connected rigidly to the frame, thus these two bodies are analyzed as a single rigid body. In addition to the cab, frame, cargo box and payload, suspensions, and wheels, we included a compliantly mounted connected engine and a rigidly mounted fuel tank. For the Class 7 truck, we used a longitudinally-constrained solid axle for the front suspension, and swing arm rear suspensions for both of the rear axles.

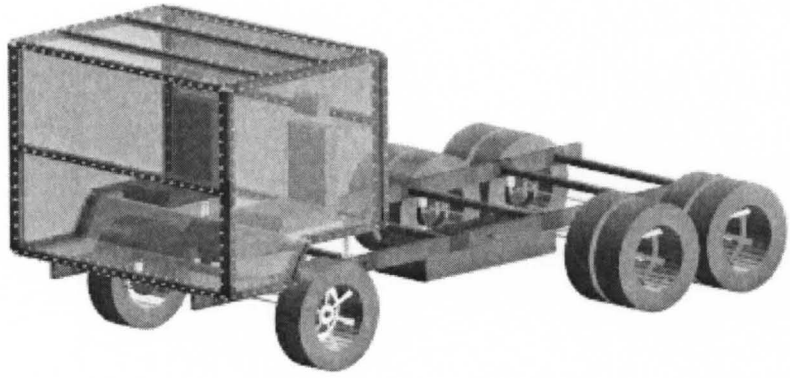
---

[2] The architecture models used for these examples and reproduced in Figure 8 do not represent the full range of architecture features that can be included in the architecture concept model and the derived rigid body analysis model. These models could have included closures, driver/passenger objects, additional geometric detail, and mass/inertia corrections to account for build-out weight.

[3] These suspension configurations were chosen for model validation purposes only, and are not typical of Class 3 trucks.



(a)



(b)

Figure 40. Vehicle concept model renderings for the example cases.[4] (a) Two-axle Class 3 truck with an engine-forward architecture, and (b) three-axle Class 7 truck with a cab-over-engine architecture.

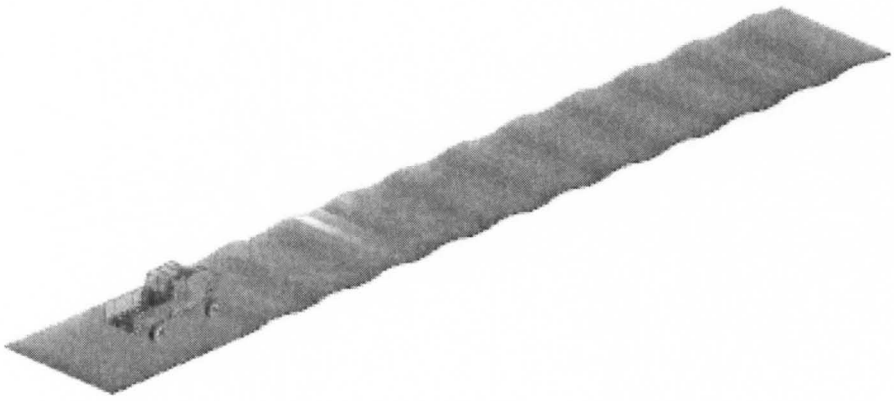


Figure 41. Class 3 truck traversing a 50 m long terrain profile with superimposed periodic bumps.

[4] Like the abstractions themselves, abstraction displays used with vehicle concept models use rendering queues that represent function, rather than topological geometry that does not exist at the concept level. In many cases, functional rendering results in depictions that are reasonably representative of physical appearance. Other visual queues, particularly those related to structural layout (location and configuration of assembly and major compliant joints, for example) are not as intuitive.

To investigate the rigid body response characteristics of the vehicle, we specified a two-dimensional (longitudinal) terrain profile (Figure 41)[5] and applied a desired angular velocity (equivalent to a target speed) and maximum torque to the driving wheels. The maximum torque is calculated based on the properties of the engine defined during the architecture design/modeling process. Braking is simulated by specifying a maximum braking torque and minimum wheel speed of zero (to model the case of lock-up). Braking performance also depends on the road surface, road condition, and tire type, parameters that can also be specified. Simulation rate and accuracy is controlled by defining the time step, number of iterations for the solver, and number of sub-steps iterations for the springs and dampers. The last option is helpful in case of very stiff springs that can lead to solution stability due to the explicit time integration.

Figure 42 shows animation screen captures depicting vehicle state renderings of the example case architecture models as they traverse portions of different terrain profiles. While a multitude of time domain kinematic response plots can be generated, the plot of Class 3 cab vertical displacement in Figure 43 is representative.

[5] Terrain profiles are currently specified using elevation singularity functions of various orders; however, the simulation algorithm can be adapted to support other, more complex approaches.

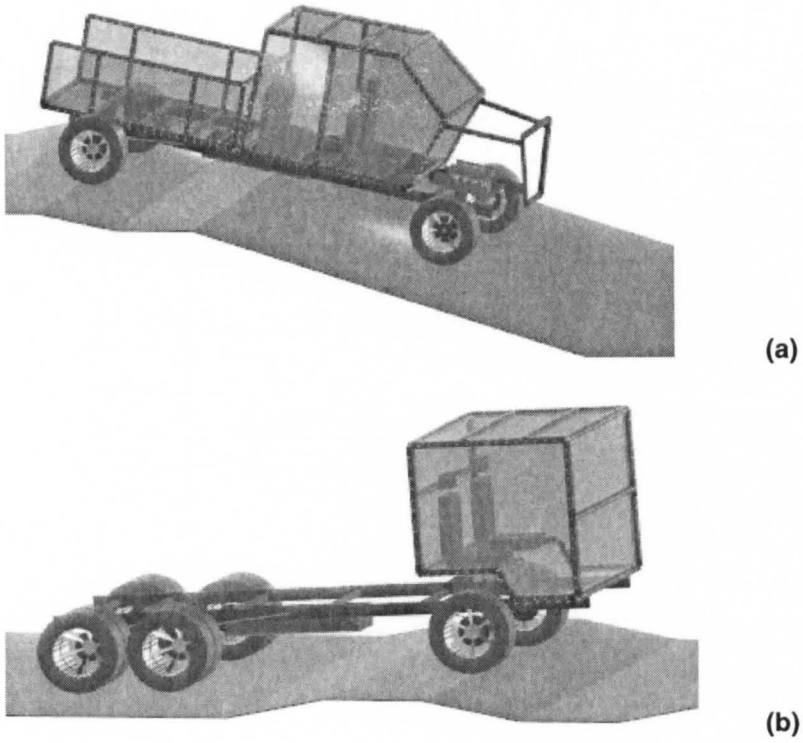


Figure 42. State rendering for the terrain response analysis showing (a) the Class 3 truck model clearing a dip in the terrain surface while driving down a 15 degree incline, and (b) the Class 7 model moving over a bump on a generally flat surface.

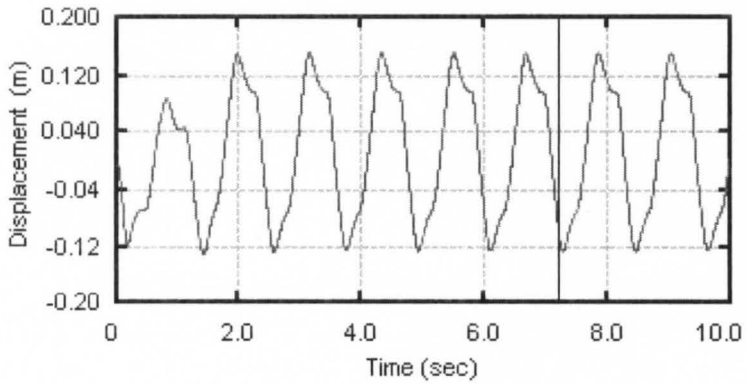


Figure 43. Class 3 truck cab displacement response while traversing a periodic terrain profile.

While a designer's understanding the rigid body time domain response of these models is critical, the frequency domain response can be important as well. Unfortunately, the system of differential equations governing the vehicle response (Eqs. (6.16), (6.59) and (6.60)) can be highly nonlinear, and formulating an appropriate

eigenproblem is quite complicated. The best way of acquiring modal characteristics or frequency response functions for such systems would be to linearize the system equations of motion, express them in standard matrix form, $\mathbf{M}\ddot{y}+\mathbf{C}\dot{y}+\mathbf{K}y=\mathbf{Q}$, and apply appropriate harmonic input/response assumptions. For more details on linearization methods, the interested reader is referred to the work of Negrut and Ortiz [159].

In our application we chose not to directly linearize the system equations; instead, we characterize the frequency domain response by using a discrete Fourier transformation (DFT). After running the rigid body simulation, acceleration responses of each body are used as input data for the DFT algorithm, which yields frequency response spectra for the bodies. DFT response spectra can permit identification of natural frequencies as high as the Nyquist frequency, particularly when transient inputs are specified. The frequency response spectrum (magnitude only) for the Class 3 light duty truck model is shown in Figure 44.

The validly of our DFT/signal processing algorithm has been tested in three different ways. For the first test we compared the DFT scheme against the simple functions that have known sinusoidal components. Next, we built a very simple vehicle configuration consisting of a frame and four wheels connected to the frame with vertically positioned springs. For this configuration eigenvalues were calculated analytically. For both cases we achieved good correlation between analytical results and numerical experiments. Finally, for a few very complex vehicle configurations we have visually inspected the cab and frame vibrations on the slowly animated models. The resulting lowest frequencies from the plot matched all the frequencies obtained by means of visual inspection.

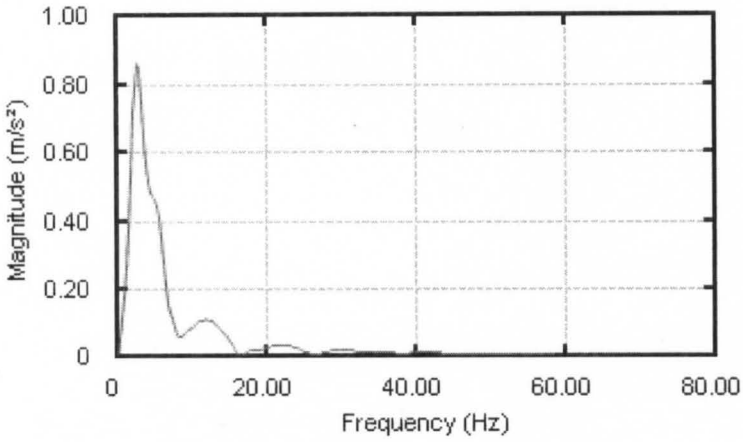


Figure 44. Cab acceleration spectrum magnitude for the Class 3 truck model.

This chapter has presented a description of the methodologies, basic data structures, and implementation algorithms for a rigid body analysis engine that makes up one portion of a comprehensive software package for vehicle concept modeling design and analysis.[6] As noted, the hierarchical organization of the vehicle architecture abstraction is based on the component and feature functionality. These abstractions contain geometric information sufficient to enable direct derivation of parametric models for many analyses appropriate for the conceptual design phase, including rigid body analysis.[7] Like the original vehicle abstraction, derived analysis models encompass the advantages (focus on primary functional attributes, small model size, direct coupling with the design process) and limitations of architecture concept models.

The multibody response engine is fully automated and capable of deriving the rigid body model from the vehicle model abstraction; however, it does have limitations. Flexible body elements are not supported, as is appropriate for rigid body analyses. The collision detection module identifies only contact between the wheels and terrain, while

---

[6] The working title of this package is Concept Modeling Tool Suite (CMTS).

[7] Other analysis types supported in the software include mass, kinematic, and geometric (MKG) properties, structural finite element analysis (including standard NVH calculations), powertrain performance, and ergonomic characteristics, to name just a few.

ignoring other interference. These restrictions aside, the rigid body analysis engine in its initial form is complete and validated. Future enhancement efforts will focus on:

- Continued improvement in computational and rendering speeds.
- Implementation of advanced terrain modeling routines, including support for transversely asymmetric profiles, and profiles with spatially varying friction and restitution coefficients.
- Development of standardized input functions and sampling rates optimized for generation of frequency response functions and modal parameters extraction (in cluding use of standard signal processing techniques).
- Adding support for localized restitution coefficients greater than unity to represent vehicle response to explosions triggered by wheel contact.

CHAPTER 7

FATIGUE ANALYSIS ENGINE

In automotive industry, evaluation of structural performance of a vehicle model is usually done by using monotonous rather than cyclic analysis. However, results obtained from fatigue analysis can be considerably different from that obtained by static FE analysis. Even though, fatigue is one of the most important characteristic that contrast among competing designs, it is often neglected by engineers, resulting in inefficient over-designed structures that can be optimized.

Fatigue can be defined as "the progressive localized permanent structural change that occurs in a material subjected to repeated strains at stresses having a maximum value less than the tensile strength of the material". It is one of the major phenomena that cause mechanical failure in components. It has been estimated that over 50% of failures in many mechanical systems are due to fatigue. Those failures culminate in cracks or fracture after a sufficient number of fluctuations of loads. Fatigue failure prediction is affected by many parameters, so that accurate assessment is difficult to achieve. Therefore, many engineers still rely on prototypes being tested under realistic loading. However, the expense, complexity and time for building these prototypes make it impractical trucks and military vehicles, therefore placing increasing emphasis on analytical durability assessment methods. In the case of special purpose vehicles, for which the low production volumes may exclude the possibility of extensive prototype testing, such emphasis would be even more pronounced.

Previously, during the vehicle design, fatigue was not considered to be critical type of analysis. It has been common for designers to allow large safety factors to reduce the likelihood of fatigue failure, where typically the design life will be one-fifth of the estimated life. Most heavy trucks and military vehicles were made of steel alloys, and designed in the way that maximum load doesn't reach the level of endurance limit. As a result vehicles had infinite fatigue life. However, in automotive industry, significant increase of the demand for lighter, more fuel efficient vehicles, reduced prototype testing iterations, and satisfactory reliability level has promoted the adoption of new materials and alloys. The trend towards reduced vehicle weight has led to the need to work closer to the design limit of the structure.

Accordingly, a tool for quick comparison of competing designs was one of the main motivations of this chapter. The analytical approach during the concept stage reduces design cycle time due to reduced detailed level analysis, allows inexpensive evaluation of changes in geometry, material and loading through performance simulation.

7.1 Theoretical Background

It is common practice in the automotive industry to employ stress-life and strain-life methods, in combination with finite element analyses. Once the stresses from the quasi-static analysis are available, the fatigue life estimation can be performed.

Stress-life approach is usually utilized when analyzing a vehicle component in the high-cycle region. However, when designing components subjected to occasional overloads, strain-life approach gives more accurate predictions. This approach is of particular interest in designing off-road vehicles, particularly for notched components

where cyclic plastic deformation can be significant. This is common for fatigue critical parts suspension components, such as steering knuckle and axle.

For steels, if specimen lifetime is more than 10^6 cycles, it is considered infinite life, since it is unlikely for specimen to fail after 10^6 cycles. Therefore, normally fatigue analyses are performed according to infinite life criteria with $N = 10^6$ cycles, though 10^3 or any other arbitrary number of cycles can be used.

For materials that exhibit endurance limit we can estimate it by means of tensile strength correlation method, as a fraction of ultimate strength. Endurance limit is:

$$S_e' = \varphi \cdot S_{ut} \tag{7.1}$$

For common carbon and low-alloy steels endurance limit is [160, 161]:

$$S_e' = \begin{cases} 0.5 \cdot S_{ut} LN(1, 0.138) \text{ MPa} & S_{ut} \le 1460 \text{ MPa} \\ 740 \cdot LN(1, 0.138) \text{ MPa} & S_{ut} > 1460 \text{ MPa} \end{cases} \tag{7.2}$$

S_{ut} – mean ultimate tensile strength. The symbol $LN(1,C)$ is a unit variate lognormally distributed, with a mean of 1 and a standard deviation C. Endurance limits approximations for other materials can be found in the literature, e.g. [149].

Some materials like aluminum and copper do not have endurance limit. However, for them endurance strength can be estimated at a specified number of cycles-to-failure. It is convenient to approximate fatigue strength in two separate regions, namely, the low-cycle $(1 \le N \le 10^3)$ and high-cycle regions $(10^3 \le N \le 10^6)$. In the high-cycle region we have

$$S_f = \sigma_F' (2N)^b \tag{7.3}$$

Where σ_F' and b are parameters of Manson-Coffin equation,

$$\sigma_F' = S_{ut} + 345\ \text{MPa} \tag{7.4}$$

$$b = -\frac{\log(\sigma_F'/S_e')}{\log(2N_e)} \tag{7.5}$$

N_e - endurance limit. For low-cycle region

$$S_f \geq S_{ut} N^{(\log f)/3} \tag{7.6}$$

$$f = \frac{\sigma_F'}{S_{ut}}\left(2 \cdot 10^3\right)^b \tag{7.7}$$

To account for the difference between endurance limit obtained in the laboratory and in real operating conditions, the Marin equation is used, in deterministic or stochastic form. Thus, the adjusted endurance limit for a particular machine part is

$$S_e = k_a k_b k_c S_e' \tag{7.8}$$

where k_a - surface condition modification factor, k_b - size modification factor (only deterministic), and k_c - load modification factor. The fatigue stress-concentration K factor is applied to the nominal stress σ as $K\sigma$.

Fluctuating stresses is often represented in the form of a sinusoidal function. Therefore, it can be characterized by midrange component σ_m and amplitude component σ_a respectively:

$$\sigma_m = \frac{\sigma_{\max} + \sigma_{\min}}{2} \text{ and } \sigma_a = \left|\frac{\sigma_{\max} - \sigma_{\min}}{2}\right| \tag{7.9}$$

Here, to account for multiaxial stress state, equivalent von Mises stresses $\sigma_{\max}$ and $\sigma_{\min}$ are obtained from finite element analyses, and utilized in the fatigue engine.

Among a number of empirical theories formulated to account for a mean stress conditions, the most popular are Soderberg and Goodman and Gerber formulas. All three criteria have been used in the engine, and it is left to the design engineer to determine, which mean stress theory is best, given the particular situation.

Fatigue factor of safety is

$$n_f = \frac{S_a}{\sigma'_a} \tag{7.10}$$

The first-cycle yield factor of safety is

$$n_y = \frac{S_y}{\sigma'_a} \tag{7.11}$$

A major issue in the vehicle industry is presence of variability in physical properties and manufacturing processes. Deterministic approaches are unable to take into account these variabilities without leading to oversized structures. The necessity of assessing the robustness of a particular design requires a new methodology based on reliability analysis and design optimization through probabilistic models of design variables.

Just as the deterministic failure loci are located by endurance strength and ultimate tensile (or yield) strength, so too are stochastic failure loci located by S_e and by S_{ut} or S_{yt}.

To take into account the effect of mean stress we solve this equation for S_a

$$S_a = \frac{r^2 S_{ut}^2}{2S_e}\left[-1+\sqrt{1+\left(\frac{2S_e}{rS_{ut}}\right)^2}\right] \tag{7.12}$$

$$r = \frac{S_a}{S_m} \tag{7.13}$$

Because of the positive correlation between S_e and S_{ut}, we increment S_e by $C_{Se}S_e$, S_{ut} by $C_{Sut}S_{ut}$, and S_a by $C_{Sa}S_a$, and solve for C_{Sa} to obtain

$$C_{Sa} = \frac{(1+C_{Sut})^2}{1+C_{Se}} \cdot \frac{\left\{-1+\sqrt{1+\left(\frac{2S_e(1+C_{Se})}{rS_{ut}(1+C_{Sut})}\right)^2}\right\}}{\left[-1+\sqrt{1+\left(\frac{2S_e}{rS_{ut}}\right)^2}\right]} - 1 \tag{7.14}$$

This equation can be viewed as an interpolation formula for C_{Sa} which falls between C_{Se} and C_{Sut} depending on load line slope r.

Probability of failure is estimated by the Gaussian distribution:

$$P(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} \cdot e^{\left(-u^2/2\right)} du \tag{7.15}$$

$$z = -\frac{\ln\left(S_a/\sigma'_a\right)}{\left(C_{Sa}^2 + C_{\sigma a}^2\right)^{1/2}} \tag{7.16}$$

There are a few ways of estimating this integral approximately. Tailor series expansion approach to represent Gaussian integral has been used. Because of the nature of this function, using Tailor expansion is very convenient way of estimating the integral. It converges very fast.

By definition Tailor expansion is

$$P(z) = P(0) + P'(0)\cdot z + P''(0)\cdot\frac{z^2}{2!} + P'''(0)\cdot\frac{z^3}{3!} + \cdots \tag{7.17}$$

The first derivative is:

$$P'(u)=\frac{1}{\sqrt{2\pi}}\cdot e^{\left(-\frac{u^2}{2}\right)} \tag{7.18}$$

The higher order derivatives are

$$P''(u)=\frac{1}{\sqrt{2\pi}}\cdot e^{\left(-\frac{u^2}{2}\right)}\cdot(-u) \tag{7.19}$$

$$P'''(u)=\frac{1}{\sqrt{2\pi}}\cdot e^{\left(-\frac{u^2}{2}\right)}\cdot\left(-1+u^2\right) \tag{7.20}$$

Hence whole sequence is

$$P(z)=\frac{1}{\sqrt{2\pi}}\sum_{n=0}^{\infty}\frac{(-1)^n z^{2n+1}}{n!(2n+1)}=\frac{1}{\sqrt{2\pi}}\left(z-\frac{z^3}{3}+\frac{z^5}{10}-\frac{z^7}{42}+\cdots\right) \tag{7.21}$$

For calculation of the above series, the following alternative formulation is more useful:

$$P(z)=\frac{1}{\sqrt{2\pi}}\sum_{n=0}^{\infty}z\prod_{i=1}^{n}\frac{-(2i-1)z^2}{i(2i+1)}=\frac{1}{\sqrt{2\pi}}\sum_{n=0}^{\infty}\frac{z}{2i+1}\prod_{i=1}^{n}\frac{-z^2}{i} \tag{7.22}$$

Base on the probability of failure from fatigue and probability of first-cycle failure, the total probability of failure can be estimated. Since these two probabilities are statistically dependent the total probability is of the form:

$$P=P_y+P_f-P_yP_f \tag{7.23}$$

P_y – probability of yield failure (first-cycle failure), P_f – probability of fatigue failure.

Reliability is:

$$R=1-P \tag{7.24}$$

7.2 Object-Oriented Design and Data Flow

The fatigue analysis engine is implemented as a module of a larger suite analysis tools, and it follows design patterns common to all analyses included in the package. The UML class diagram of Figure 45 depicts this shared structure. Analysis, Engine and PostProcessor are all abstract classes, and interaction between them also implemented on the abstract level through their interfaces. This allows us to implement a simulation framework that is independent of the underlying simulation method, and where strategy of the implementation can be easily changed by switching to the different concrete class.

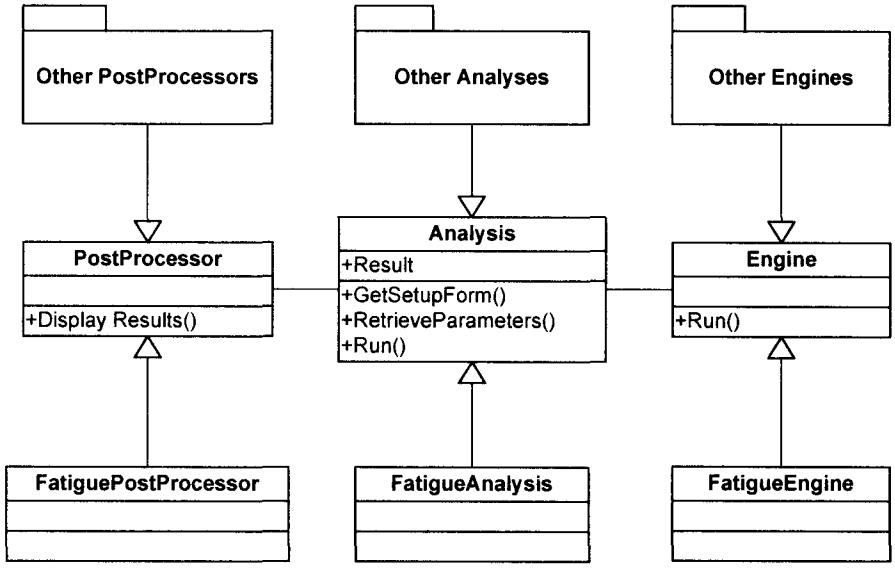


Figure 45. UML class diagram of the fatigue module.

In addition to being capable of rapid solution times, a viable analysis suite should be capable of analyzing full vehicle models, a primary subsystem models (a cab or rolling chassis, for example), and in some cases singular feature or component models. It must have access to visual renderings that depict the vehicle architecture's physical layout while allowing designer to interact with both the physical design space and analysis

results. Finally, individual modules of the analysis suite should be algorithmically integrated.

The entire fatigue analysis module constitutes of the three fundamental classes: FatigueAnalysis, FatigueEngine and FatiguePostProcessor. Each piece has a fundamental function associated with it. FatigueAnalysis is the class responsible for the interaction of Fatigue module with the rest of the software. Inside FatigueAnalysis we store input parameters and results of the analysis. The class is also responsible for employing mesher and FE solver. FatigueEngine is the actual fatigue solver that controls all subclasses responsible for the underlying fatigue mathematical model calculations. This class defines the interaction between its subclasses (Figure 45 show that fatigue consists of sub classes based on deterministic and stochastic analysis). Safety factor and reliability calculations for each node in the mesh are performed in FatigueEngine. More details on the structure and the mathematical background of the engine are presented in the next section.

FatiguePostProcessor is the class responsible for the visual representation of results. The simulation results are normally depicted as a contour plot of the safety factors or failure probabilities of the structure being analyzed.

Fatigue analysis engine is strongly dependent on the FE engine and mesher. NASTRAN solver has been used to perform finite element analysis and calculate stresses induced in a model during the cyclic loading.

7.3 Results and Discussion

The engine has alternatives to choose between deterministic or stochastic fatigue analyses. Fatigue modification factors such as size, load and surface finish may be

specified for each component, and applied as a multiplier to the endurance limit. Input is in the form of a maximum-minimum or fully-reversed load. Output is in the form of contour plots for fatigue safety factor, reliability, as well as yield factor of safety and total reliability. Specifically, the numerical results of simulations can be viewed by clicking on any point of a contour plot.

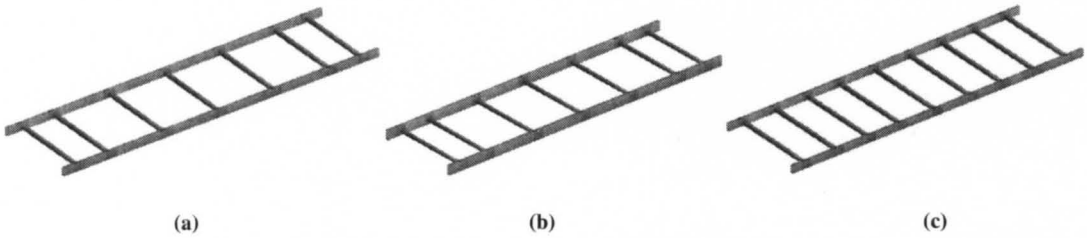
To demonstrate the fatigue analysis results, a few vehicle assembly concept models have been built. The purpose of this chapter is to demonstrate how the comparative fatigue analysis can be performed between alternative configurations of a given structural component or assembly. Fatigue results depend on the variation in size, surface finishing, stress concentration, as well as material characteristics such as Young modulus and ultimate strength. Furthermore, fatigue characteristics strongly influenced by the difference in architecture layouts and body mount points, represented by the points of application of model constraints and loads. Even modest changing of these attributes can have notable effect on the overall fatigue endurance. Lastly, due to the possible variation of the assembly connection points, the difference in the loading that contribute to the model through these connections can be considered.

7.3.1 Ladder Frame

The first of these models is a ladder frame typical for a Class 3[8] (3,850 - 4,540 kg GVWR) light-duty commercial truck. Comparative fatigue analysis has been performed for three configurations of ladder frame (Figure 46). Specifically, first and second models (Figure 46a and Figure 46b) have been compared in case of bending load and later first

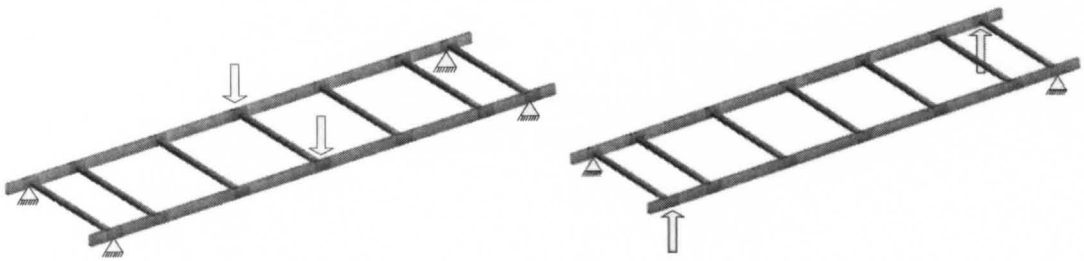
[8] US Department of Transportation Federal Highway Administration (FHWA) commercial truck classifications based on the vehicle's gross vehicle weight rating (GVWR).

and third (Figure 46a and Figure 46c) in torsion. Constraint and load applied in both cases are shown on the Figure 47.



(a) (b) (c)

Figure 46. Example of three frame configurations.



(a) (b)

Figure 47. Bending and torsion load test cases for frame.

For the bending case two ends of each rail were fixed by constraining translational degrees of freedom, and releasing rotational. Load has been applied to the two center points of the rails. Mean and alternating loads are the same and equal to 400 N. First and second model are identical with the exception that the size of rails of the second model is much larger. For both deterministic and stochastic analysis modes, minimum factor of safety and reliability were the lowest for the case of first frame architecture. For the first frame architecture the factor of safety was 1.7, while for the second it was 2.7. Besides comparing two models, quantitative estimation should serve as one of the indicators to decide if the model is appropriate for the concept level design. For example if resulting safety factor or reliability is less than some threshold, it may be a signal that the design should be revised before it is approved for the next phase, where it can be too expensive or impossible to fix the problem.

Corresponding contour plots of the resulting fatigue evaluation of the bending case are depicted on the Figure 48 and Figure 49. On the plots, red color corresponds to the parts of members with low factor of safety or low reliability, and blue color to the parts with high safety factor and reliability.

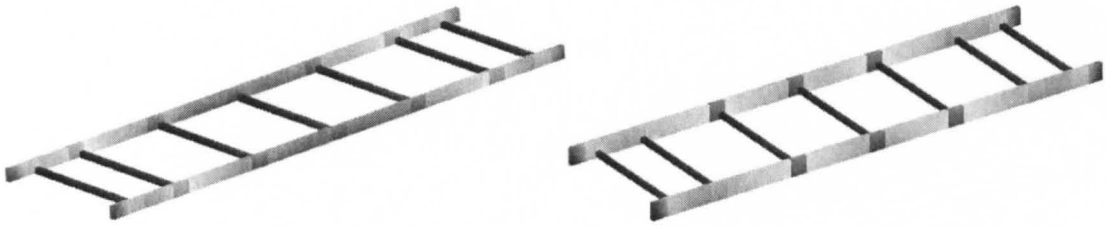


Figure 48. Example of fatigue factor of safety contour plots for two frame configurations (bending case).

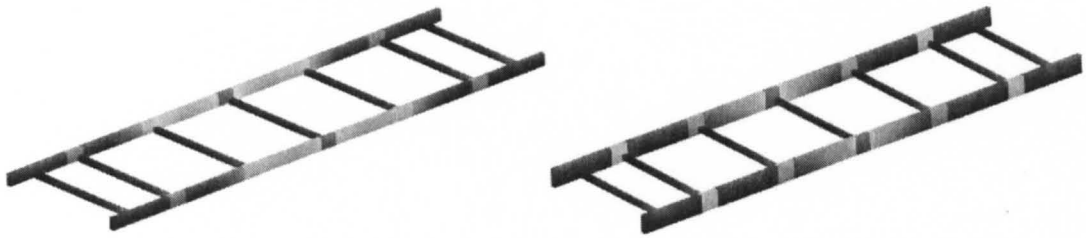


Figure 49. Example of fatigue reliability contour plots for two frame configurations (bending case).

For the torsion case, first and third models have been used (Figure 46a and Figure 46c). Second model is designed to be stiffer by adding two extra cross members of the same size, material and cross-section properties. Two opposite corners have been fixed by constraining all 6 DOF, while 500 N loading of mean and alternating magnitude have been applied to the other opposite corners. Estimated minimum safety factor and reliability for the first model was 1.7 and 80%, and correspondingly, for the second configuration 2.1 and 90%. Contour plots of the resulting fatigue evaluation of the torsion case are depicted on the Figure 50 and Figure 51.

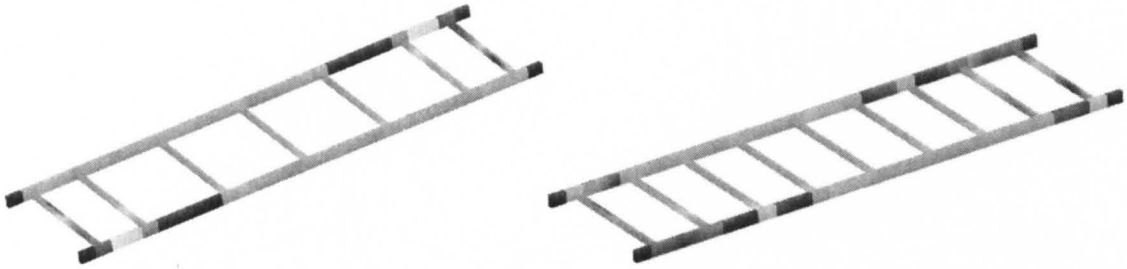


Figure 50. Example of fatigue reliability contour plots for two frame configurations (torsion case).

Figure 51. Example of fatigue factor of safety contour plots for two frame configurations (torsion case).

7.3.2 Engine-Forward Cab

Next, three engine-forward cab configurations for a Class 3 vehicle have been designed (Figure 52). Similarly, as in case of ladder frame, an engine-forward cab assembly has been subjected to the bending and torsion load cases (Figure 53).

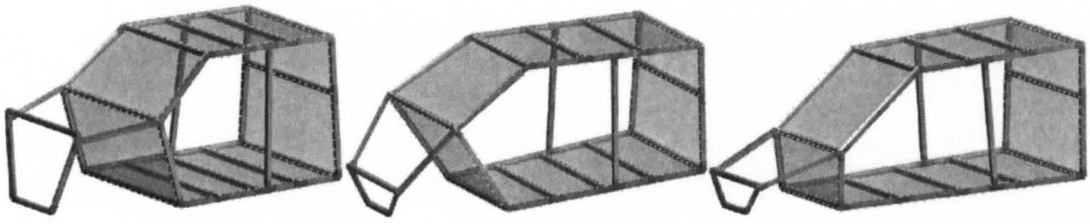
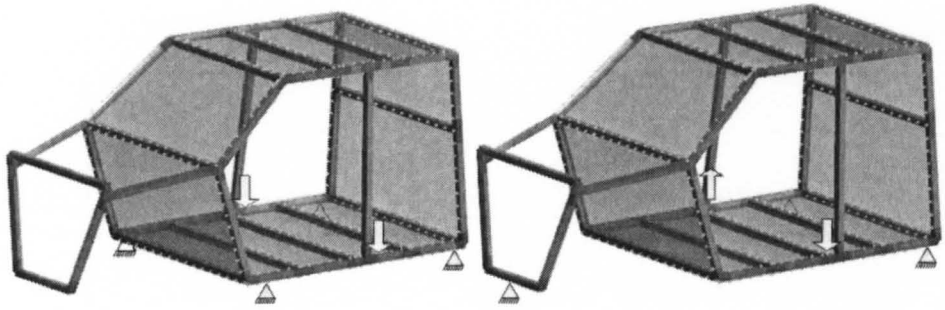


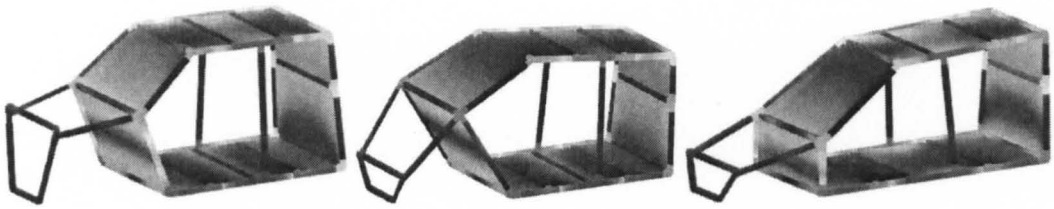
Figure 52. Example of three cab configurations.



(a) (b)

Figure 53. Bending and torsion load test cases for cab.

For the bending case, the four corners of the floor have been constrained (Figure 53a). Loading of 3000 N has been applied to the middle of the each rail on the floor. Estimated minimum safety factor was the smallest in the third model and for the first model was equal to 1.6 (Figure 54c). For the first and second configuration the values were 2.2, and 2.1, correspondingly (Figure 54a and Figure 54b). For the torsional case constraining model to be tested is slightly different than that in case of frame. Here, two rear corners of the floor and middle point of the bottom radiator support beam have been constrained and the loading of 4000 N has been applied to the meddle point of two floor rails. As a result the following numbers were obtained: safety factor of 1.8, 1.7 and 1.25 for first, second and third model correspondingly (Figure 55). On the Figure 56 reliability has been compared only for the first and third models and resulting in the close values of 85% and only 50%. Additionally, the example of safety factor contour plot that guards against yield failure is shown on the Figure 57.



(a) (b) (c)

Figure 54. Example of fatigue factor of safety contour plots for three cab configurations (bending case).

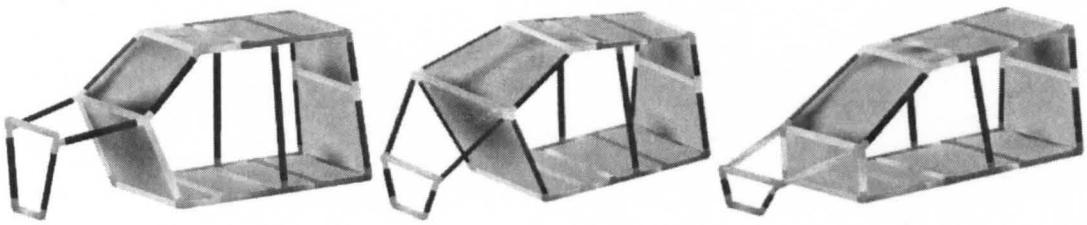


Figure 55. Example of fatigue factor of safety contour plots for three cab configurations (torsion case).

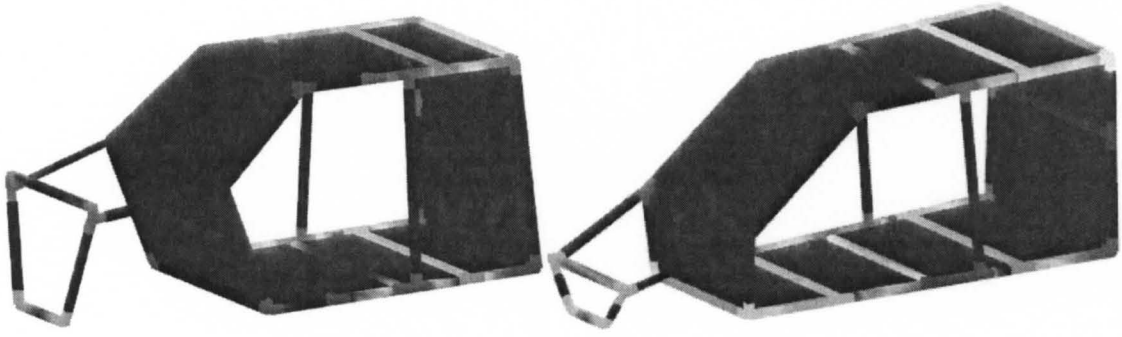


Figure 56. Example of fatigue reliability contour plots for two cab configurations (torsion case).

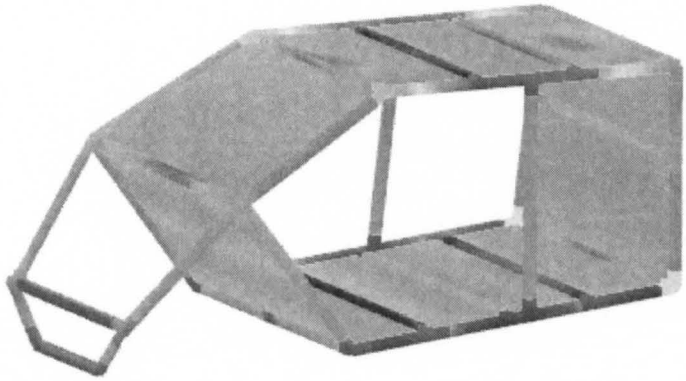
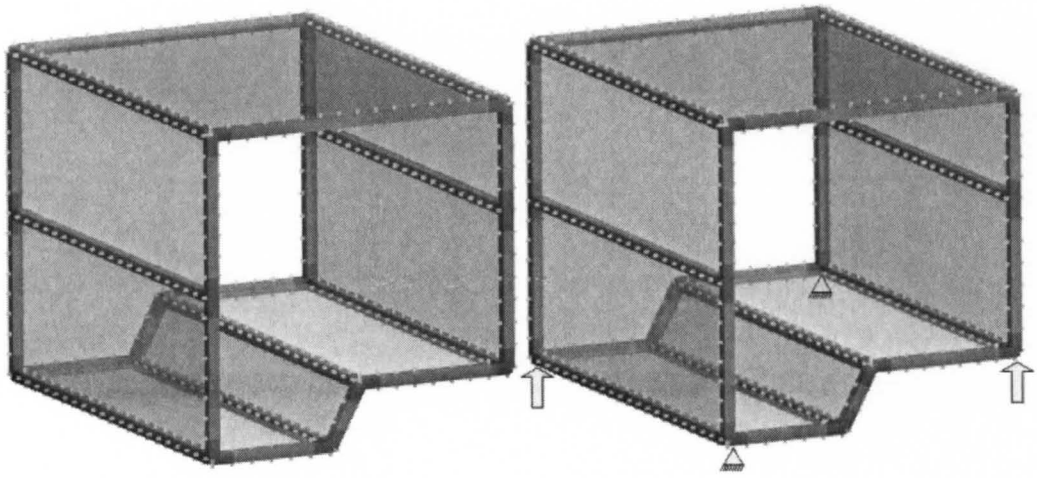


Figure 57. Example of fatigue factor of safety that guards against yield failure (bending case).

7.3.3 Cab-Over-Engine

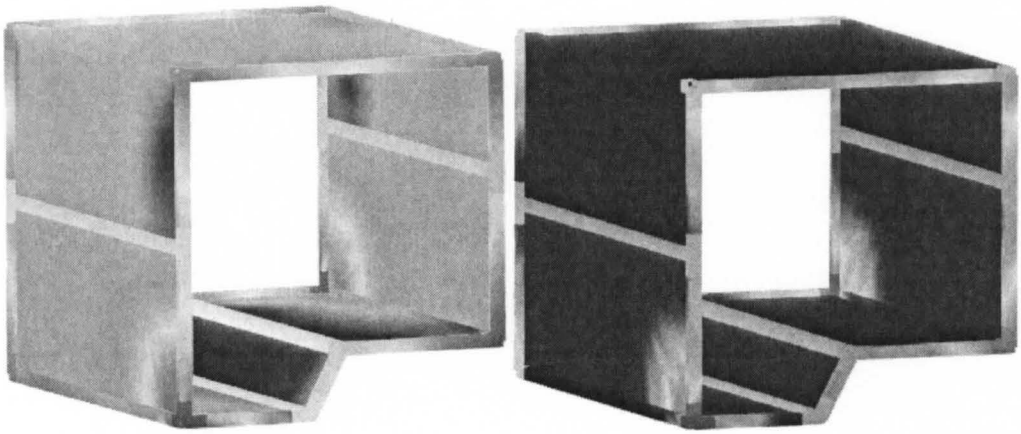
The next model represents the cab-over-engine architecture (Figure 58a) of a Class 7 (11,800 - 14,970 kg GVWR) heavy-duty commercial truck. Torsional loading applied to the cab-over-engine is similarly as in the case of the ladder frame and is shown on the Figure 58b. Opposite corners of the cab floor have been constrained and equal loading of 1500 N for bending and 2000N for torsion cases have been applied the model. Minimum

factor of safety and reliability index obtained from the analysis were 2.2 and 92%. Analysis results for the torsional load case are shown on the Figure 59.



(a) (b)

Figure 58. Example of cab-over-engine configuration.



(a) (b)

Figure 59. Example of fatigue factor of safety (a) and reliability (b) contour plots for cab-over-engine in torsion.

7.3.4 McPherson Strut

Lastly, two McPherson strut models have been created and tested (Figure 60). Constraints and load applied to the model is presented on the Figure 61. Fatigue factor of safety and reliability contour plots are depicted on the Figure 62 and Figure 63.

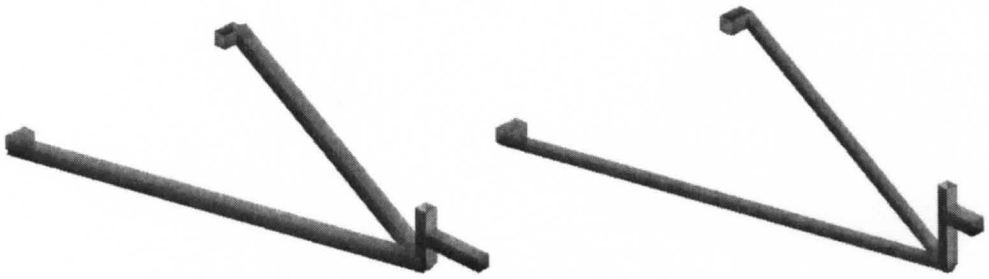


Figure 60. Example of two McPherson strut configurations.

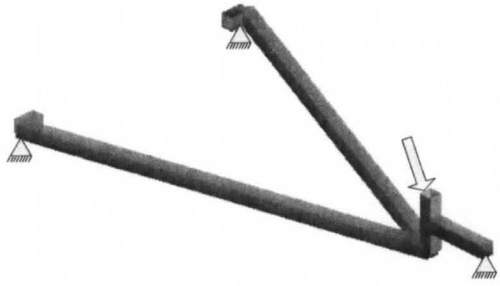


Figure 61. Bending and torsion load test cases for frame.

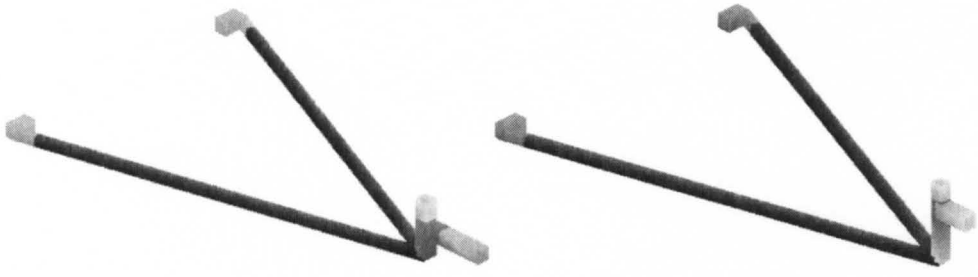


Figure 62. Fatigue factor of safety contour plots for two configurations of McPherson strut.

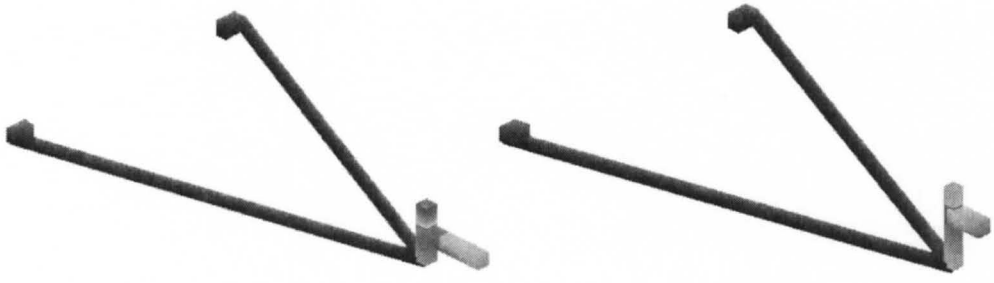


Figure 63. Example of fatigue reliability contour plots for two configurations of McPherson strut.

CHAPTER 8

CONCLUSIONS

The research contained herein provides five fundamental contributions to the state-of-the-art in architecture-level vehicle design, modeling and simulation methodology.

1. Systematic and comprehensive description of abstraction methodologies for vehicle architecture modeling and analyses.
2. Development and algorithmic implementation of geometric modeling and rendering methodology that supports concept-level vehicle design.
3. Methodology development and implementation of a fully-automatic hybrid meshing module that is appropriate for concept finite element mesh generation.
4. Development and software implementation of a rigid body analysis engine, along with the methodologies and data structures for deriving analysis models from architecture models.
5. Framework and algorithmic implementation of fatigue analysis, as composite analysis engine based on finite element solver.

First contribution addresses the abstraction methodologies capable of representing full vehicle architecture models, and used as the starting point for a broad range of analyses appropriate for architecture-level optimization. These abstraction methodologies include descriptions of architecture-level vehicle component models and systems, its hierarchical organization, requirements for architecture-level analyses, as well as "hierarchical" organization of architecture-level analyses.

While architecture models have been recently discussed in the literature, and "simplified" finite element models have been used in vehicle design to shorten analysis cycles, there has been no effort to systematically describe an appropriate abstraction level for vehicle architecture models that function as a design space and as a starting point for deriving analysis models. The resulting chapter 3 describes novel abstraction methodology and provides a number of contributions.

Abstraction methodology provides a list of analyses needed for architecture analysis and optimization. It is critical to conduct all fundamental analyses during the concept phase, since defects in the concept vehicle architecture can lead to poor results in the detailed design phase. The vehicle concept design follows the same process as the detailed design: architecture design, model discretization, analysis and optimization. Moreover, during the course of conceptual analysis of a vehicle model, engineers can utilize the same fundamental set of analyses which are used during the detailed phase. However, results of the conceptual design analyses are rather qualitative, while results of the detailed design analyses are quantitative. Each "simplified" analysis dictates the minimum information required by the corresponding analysis model. This information includes description of geometric level of detailing, material properties and rules of interfacing between components and subassemblies.

Nobody has proposed a practical approach to representing full-vehicle models at the architecture level, or for "designing a concept" in the similar sense to that used for geometric design. In this work the abstraction methodology provides a formal description of components and representation of hierarchical feature organization. In general all components comprising a vehicle are grouped into two categories: structural and non-

structural. Structural components can carry loads in the vehicle, while the non-structural contribute to the load but can't carry it, and are considered with other analyses, for example, powertrain analysis, rigid body analysis, etc. This simplifies the analysis model derivation, where structural components are automatically included into structural analysis models, and non-structural components are used for non-structural analysis models correspondingly.

Vehicle components, body structures, suspensions and powertrain, energy storage and transfer elements, the human operator and passengers are represented as a hierarchy ordered according to features' functions. Components and connectivity information are stored in the graph data structure, where components are represented by nodes and component connections by edges. On the very top level of the abstraction hierarchy are assemblies and assembly connections, and further, each assembly can include beams, panels, rigid components, and component connections.

Besides hierarchical feature organization, abstraction methodology provides a novel formal description of architecture-level analyses hierarchy. This analyses hierarchy reflects the fact that all analyses require different level of abstraction, and thus can be ordered sequentially. Moreover, some analyses depend on the results of other analyses, and are considered to be sequential. As a result, analyses were categorized into three groups based on the corresponding level of vehicle abstraction: first phase, second phase and third phase. Each analysis phase is strongly based on the previous, and deals with more detailed feature descriptions. The first category corresponds to the highest level of abstraction, where a proposed architecture is represented on the architecture diagram as a list of components and associations between them. A typical component diagram outlines

a very basic vehicle configuration, including type of a cabin, passengers' seats, frame, front and rear suspension, wheel configuration and type of a cargo box. Second category deals with the preliminary geometric model, and from the designer's prospective, defines basic load path requirements and packaging of the vehicle model. This initial geometric data is sufficient to estimate spatial, ergonomic and inertial characteristic of the model, and perform RBA and Rollover analyses. Finally, the third category requires the most detailed characterization of a vehicle's architecture. Some analyses under this category, such as crashworthiness and braking performance, are critical for safety of occupants, and therefore require exceptional attention to details in the model.

To summarize, the purpose of concept modeling and analysis methodology described in this work is to surpass limitations of the traditional vehicle design process. During the conceptual stage a vehicle model is considerably simpler, geometric modeling is very quick, and as a result, automotive engineers have additional time for comparison and optimization of conceptual architectures, thus increasing the likelihood of creating a successful product. Primary contributions of the concept modeling and analysis methodology can be outlined as follows:

- Analyses hierarchy for architecture establishment and optimization
- Function-based component abstractions and function-cued visual representation
- Object-oriented design of architecture elements
- Comprehensive full vehicle architecture concept models
- Comprehensive analysis modules

Second contribution addresses a number of important questions related to the geometric model characterization at the architecture level. Chapter 4 discusses the

fundamental methodology required for geometric modeling and representation of vehicle architecture. Curve and surface modeling methodology has been used and implemented in an interactive design interface consistent with vehicle conceptual design information and methods. The interactive interface supports both geometric design and function-cued rendering of a vehicle model during the conceptual design stage. It provides a quick starting point for the development of common vehicle architectures from existing scaffolding geometry common template files. Function-cued rendering provides the visual representation that quickly indicates what a feature does, how it interfaces with associated objects, and what is its object type. Finally, surface modeling methodology is used as a basis for finite element mesh generation.

Contribution number 3 addresses the critical need for a vehicle body grid generator suitable for use in concept modeling software capable of functioning as a design environment for practicing vehicle designers. Methodology development and implementation of a fully-automatic hybrid meshing module that is appropriate for concept finite element curve and surface mesh generation, supports proper handling of structural point and line constraints, and has provision for future mesh adaptation and element conversion. Even though there are many algorithm and strategies for mesh generation, most commercial meshing software requires a significant amount of user input. Decisions must be made about sizing, elements type, grid type, etc. This need for specialist-level input explains the differentiation between "analysts" and "designers" seen in nearly all vehicle design efforts. The concept modeling approach proposed here permits most decisions to be made independently of user input. All necessary information can be extracted from the vehicle model. Mesh generation does not change the geometric

or mechanical properties of an underlying component model, thus, theoretically, there is no reason for a designer to interact with meshing module. As a result, a designer performing these analyses should, in principle, not need to be familiar with finite element theory or meshing methodologies. Besides, intelligent automation of the mesh generation greatly speeds up the overall vehicle development process.

Fourth contribution of this research addresses an important need for an architecture-level multibody response analysis engine. Development and software implementation of a rigid body analysis engine, along with the methodologies and data structures for deriving analysis models from architecture models are discussed. The software implementation tasks encompassed development of user interfaces for defining terrain profile, configuring the analysis parameters, and animation output. Terrain response assessment may be performed by the engineer once the inertial properties and the assembly connectivity are defined in terms of the connection compliance and DOF. It is even possible that the inertial properties can be available prior to the geometric model. While the rigid body engine requires relatively little input data from the architecture model, analysis results can be very important vehicle performance characteristics that are available early in design. In chapter 6, it has been demonstrated how the state-of-art RB engine can be implemented in the scopes of the vehicle concept modeling software. Vehicle architecture abstractions contain geometric information sufficient to enable direct derivation of parametric models for many analyses appropriate for the conceptual design phase, including rigid body analysis. Like the original vehicle abstraction, derived analysis models encompass the advantages (focus on primary functional attributes, small model size, direct coupling with the design process) of architecture concept models. The

multibody response engine is fully automated and capable of deriving the rigid body model from the vehicle model abstraction.

Fifth contribution pertains to development of a framework for a fatigue analysis engine coupled with a FE solver integrated into the concept modeling software. Chapter 7 has presented a description of the methodologies, basic data structures and implementation algorithms, and it has been demonstrated how the tool for quick comparison of fatigue characteristics of competing designs can be implemented as a composite analysis engine that makes up one portion of a comprehensive software package for vehicle concept modeling design and analysis. The hierarchical organization of the vehicle architecture abstraction is based on the component and feature functionality. These abstractions contain sufficient geometric and material information to enable fatigue analyses appropriate for the conceptual design phase. The fatigue engine is an auxiliary tool for the vehicle assembly structural design. Fatigue factor of safety and reliability serve as constraints during the optimization process. Early estimation of fatigue reliability can help locating the problematic regions in the assembly before proceeding with the detailed level analysis. Besides that, comparative analysis between different assembly configurations can be useful when making the decision about appropriate design. The engine implements both deterministic and stochastic fatigue analyses. The analysis implementation includes user interfaces for load case definition, fatigue parameters configuring and resulting output depiction in the form of contour plots for the fatigue safety factor, reliability, as well as yield factor of safety and total reliability.

CHAPTER 9

RECOMMENDATIONS

Even though all the dissertation objectives were achieved, the problem is very broad, and numerous enhancements could be made to improve the abstraction methodology and corresponding software.

While the concept modeling methodology has been extensively described in this work, proper transition from a concept model to the corresponding detailed model would be a great enhancement. Currently there is no direct pathway from the concept model to an equivalent geometric model, and as a result, the process is somewhat disjointed. Once the vehicle architecture is established the engineer has to start building the geometric model from scratch. Providing a transition pathway form concept architecture model files to geometric representations can increase the efficiency of using a concept modeling software as a pre-CAD tool in a vehicle design development. The most obvious advantages of the corresponding transition methodology are:

- Eliminating time spent by the designer when creating a CAD model from scratch based on concept model
- The possibility of using a concept model representations as a starting point for detail-level design after the architecture has been established
- The possibility to view concept architecture designs outside of the concept modeling environment, in external CAD/CAE software packages
- Saving concept models in a format suitable for prototyping

Fully automatic surface mesh generation for finite element concept models is quite complex, and the field of surface mesh generation is still under active research. A number of enhancements can be added to improve the meshing functionality. Currently implemented combination of algebraic and advancing front meshing can be enhanced by utilizing parallel programing algorithms. Both approaches account for connectivity specified within the model, however, constraint handling is still under investigation and new optimized algorithms can be developed. Mesh coarsening algorithms can be considered in order to produce reduced size finite element models, and reduce solution times while remaining consistent with the concept modeling philosophy. Generally speaking, mesh generation for finite element concept models is very complex and can be separated into a number of distinct research topics.

Future enhancement efforts for vehicle dynamics response analysis module will focus on:

- Continued improvement in computational and rendering speeds.
- Implementation of advanced terrain modeling routines, including support for arbitrary 3D profiles, and profiles with spatially varying friction and restitution coefficients
- Development of standardized input functions and sampling rates optimized for generation of frequency response functions and modal parameters extraction (including use of standard signal processing techniques)
- Adding support for localized restitution coefficients greater than unity to represent vehicle response to explosions triggered by wheel contact

- Adding support for flexible body representation in order to properly model behavior of certain types of assemblies, and particularly suspensions.
- Linearize the system of differential equations governing the vehicle response, i.e. express the system equations of motion in standard matrix form, in order to formulate an appropriate eigenproblem and estimate modal characteristics or frequency response functions

Heavy industrial and special purpose vehicles often have a driver observational field of view that is limited by architecture elements, including body structure features, equipment, and payload. Consequently, the resulting blind spots can be a safety hazard when operating the vehicle in high-traffic or urban areas. A field of view analysis module would be a good enhancement for a family of ergonomic analyses appropriate for concept-level vehicle architecture development.

REFERENCES

[1] Kojima Y. (2000) "Mechanical CAE in automotive design," R&D Review of Toyota CRDL, 35, pp. 1-10.

[2] Hou W., Zhang H., Chi R., Hu P. (2009) "Development of an intelligent CAE system for auto-body concept design," Int. J. Automotive Technology, 10, pp. 175-180.

[3] Torstenfelt B., Klarbring A. (2007) "Conceptual optimal design of modular car product families using simultaneous size, shape and topology optimization," Finite Elements in Analysis and Design, 43, pp. 1050-1061.

[4] Schelkle E., Elsenhans H. (2002) "Virtual vehicle development in the concept stage – current status of CAE and outlook on the future," Conference Proceedings for the 3rd Worldwide MSC.Software Aerospace Conference & Technology Showcase.

[5] SFE Concept. < http://sfe1.extern.tu-berlin.de >.

[6] Bylund N. (2004) "Simulation driven product development applied to car body design," PhD thesis, Lulea University of Technology.

[7] Prater G., Shahhosseini A., Kuo E., Mehta P., Furman V. (2005) "Finite element concept models for vehicle architecture assessment and optimization," SAE 2005 World Congress Proceedings, no. 2005-01-1400, Detroit, MI, USA.

[8] Prater G., Shahhosseini A., State M., Furman V., Azzouz M. (2002) "Use of FEA concept models to develop light-truck cab architectures with reduced weight and enhanced MVH characteristics," SAE Technical Paper No. 2002-01-0369.

[9] Shahhosseini A., Prater G., Osborne G., Kuo E., Mehta P. (2010) "Major compliance joint modeling survey for automotive body structures," Int. J. Vehicle Systems Modelling and Testing, 5, pp. 1-17.

[10] Osborne G., Prater G., Shahhosseini A. (2010) "Finite element concept modeling methodologies for pickup truck boxes," Int. J. Heavy Vehicle Systems, 17, pp. 1–17.

[11] Donders S., Takahashi Y., Hadjit R., Langenhove T., Brughmans M., Genechten B., Desmet W. (2009) "A reduced beam and joint concept modeling approach to optimize global vehicle body dynamics," Finite Elements in Analysis and Design, 45, pp. 439-455.

[12] Mundo D., Hadjit R., Donders S., Brughmans M., Mas P., Desmet W. (2009) "Simplified modeling of joints and beam-like structures for BIW optimization in a concept phase of the vehicle design process," Finite Elements in Analysis and Design, 45, pp. 456-462.

[13] Mundo D., Donders S., Hadjit R., Stigliano G., Mas. P., Auweraer H. (2010) "Concept modelling of automotive beams, joints and panels," Proc. World Scientific and engineering Academy and Society (WSEAS) International Conference on FD, FEM, FV and BEM, Romania.

[14] Stigliano G., Mundo D., Donders S., Tamarozzi T. (2010) "Advanced vehicle body concept modeling approach using reduced models of beams and joints," Proc. of ISMA 2010 International Conference on Noise and Vibration Engineering, Belgium. Pp. 4179-4190.

[15] Dai Y., Duan C. (2009) "Beam element modeling of vehicle body-in-white applying artificial neural network". Applied Mathematical Modelling, 33, pp. 2808-2817.

[16] Goldman R. (2003) "Pyramid algorithm: A dynamic programming approach to curves and surface for geometric modeling," Elsevier Science.

[17] Farin G., Hoschek J., Kim M.–S. (2002) "Handbook of computer aided geometric design," Elsevier Science.

[18] Farin G. (1997) "Curves and surfaces for computer aided geometric design: A practical guide," 4th Edition, Academic Press.

[19] Salomon D. (2006) "Curves and surfaces for computer graphics," Springer.

[20] Saxena A., Sahay B. (2005) "Computer aided engineering design," Springer.

[21] Piegl L., Tiller W. (1997) "The NURBS book," Springer.

[22] Lim C.-G. (1999) "A universal parameterization in B-spline curve and surface interpolation," Computer Aided Geometric Design, 16, pp. 407-422.

[23] Goodman T., Ong B., Sampoli M. (1998) "Automatic interpolation by fair, shape-preserving, G^2 space curves," Computer-Aided Design, 30, pp. 813-822.

[24] Goodman T., Ong B. (1997) "Shape preserving interpolation by space curves," Computer Aided Geometric Design, 15, pp. 1-17.

[25] Park H., Kim K., Lee S.-C. (2000) "A method for approximate NURBS curve compatibility based on multiple curve refitting," Computer-Aided Design, 32, pp. 237-252.

[26] Ishida J. (1997) "The general B-spline interpolation method and its application to the modification of curves and surfaces," Computer-Aided Design, 29, pp. 779-790.

[27] Juhasz I., Hoffmann M. (2004) "Constrained shape modification of cubic B-spline curves by mean of knots," Computer-Aided Design, 36, pp. 437-445.

[28] Juhasz I., Hoffmann M. (2003) "Modifying a knot of B-spline curves," Computer Aided Geometric Design, 20, pp. 243-245.

[29] Renner G., Weib V. (2004) "Exact and approximate computation of B-spline curves on surface," Computer-Aided Design, 36, pp. 351-362.

[30] Yang Y.-J., Cao S., Yong J.-H., Zhang H., Paul J.-C., Sun J.-G., Gu H.-J. (2008) "Approximate computation of curves on B-spline surface," Computer-Aided Design, 40, pp. 223-234.

[31] Ma Y., Hewitt W. (2003) "Point inversion and projection for NURBS curves and surface: Control polygon approach," Computer Aided Geometric Design, 20, pp. 79-99.

[32] Chen X.-D., Su H., Yong J.-H., Paul J.-C., Sun J.-G. (2007) "A counterexample on point inversion and projection for NURBS curve," Computer Aided Geometric Design, 24, p. 302.

[33] Nasri A. (1997) "Curve interpolation in recursively generated B-spline surfaces over arbitrary topology," Computer Aided Geometric Design, 14, pp. 13-30.

[34] Farin G., Hansford D. (1999) "Discrete Coons patches," Computer Aided Geometric Design, 16, pp. 691-700.

[35] Konno K., Chiyokura H. (1996) "An approach of designing and controlling free-form surfaces by using NURBS boundary Gregory patches," Computer Aided Geometric Design, 13, pp. 825-849.

[36] Qin S., Wright D. (2006) "Progressive surface modeling scheme from unorganized curves," Computer-Aided Design, 38, pp. 1113-1122.

[37] Leon J., Trompette P. (1995) "A new approach towards free-form surface control," Computer Aided Geometric Design, 12, pp. 395-416.

[38] Hu S.-M., Li Y.-F., Ju T., Zhu X. (2001) "Modifying the shape of NURBS surfaces with geometric constraints," Computer-Aided Design, 33, pp. 903-912.

[39] Zhang C., Zhang P., Cheng F. (2001) "Constrained scaling of trimmed NURBS surfaces based on fix-and-stretch approach," Computer-Aided Design, 33, pp. 103-112.

[40] Che X., Liang X., Li Q. (2005) "G^1 continuity conditions of adjacent NURBS surfaces," Computer Aided Geometric Design, 22, pp. 285-298.

[41] Konno K., Tokuyama Y., Chiyokura H. (2001) "A G^1 connection around complicated curve meshes using C^1 NURBS Boundary Gregory Patches," Computer-Aided Design, 33, pp. 293-306.

[42] Hui K., Wu Y.-B. (2005) "Feature-based decomposition of trimmed surface," Computer-Aided Design, 37, pp. 859-867.

[43] Cripps R., Parwana S. (2011) "A robust efficient tracing scheme for triangulating trimmed parametric surfaces," Computer-Aided Design, 43, pp. 12-20.

[44] Kumar S., Manocha D. (1994) "Efficient rendering of trimmed NURBS surfaces," Computer-Aided Design, 27, pp. 509-521.

[45] Hamann B., Tsai P.-Y. (1995) "A tessellation algorithm for the presentation of trimmed NURBS surfaces with arbitrary trimming curves," Computer-Aided Design, 28, pp. 461-472.

[46] Piegl L., Tiller W. (1997) "Geometry-based triangulation of trimmed NURBS surfaces," Computer-Aided Design, 30, pp. 11-18.

[47] Cheung G., Lau R., Li F. (2003) "Incremental rendering of deformable trimmed NURBS surfaces," Proc. ACM Symposium on Virtual Reality Software and Technology.

[48] Ng W., Tan S. (2000) "Incremental tessellation of trimmed parametric surface," Computer-Aided Design, 32, pp. 279-294.

[49] Cho W., Patrikalakis N., Peraire J. (1998) "Approximate development of trimmed patches for surface tessellation," Computer-Aided Design, 30, pp. 1077-1087.

[50] Gutierrez M., Borst R., Schellekens J., Sluys L. (1995) "An algorithm for mesh rezoning with application to strain localization problems," Computers and Structures, 55, pp. 237-247.

[51] Cunha A., Canann S., Saigal S. (1997) "Automatic boundary sizing for 2D and 3D meshes". In Trends in Unstructured Mesh Generation, ASME, 220, pp. 65–72.

[52] Cuilliere J. (1997) "A direct method for the automatic discretization of 3D parametric curves," Computer-Aided Design, 29, pp. 639-647.

[53] Wu B., Wang S. (2005) "Automatic triangulation over three-dimensional parametric surfaces based on advancing front method," Finite Elements in Analysis and Design, 41, pp. 892-910.

[54] Thompson J., Soni B., Weatherill N. (1999) "Handbook of grid generation," CRC Press.

[55] Frey P., George P.-L. (2000) "Mesh generation: application to finite elements," Hermes Science.

[56] Farrashkhaluat M., Miles J. (2003) "Basic structured grid generation: with an introduction to unstructured grid generation," Butterworth Heinemann.

[57] Tristano J., Owen S., Cannan S. (1998) "Advancing front surface mesh generation in parametric space using a Riemannian surface definition," In proceedings of 7th International Meshing Roundtable, pp. 429-445.

[58] Miranda A., Martha L. (2002) "Mesh generation on high-curvature surfaces based on a background quadtree structure," In proceedings of 11th International Meshing Roundtable, pp. 333-342.

[59] Owen S., Staten M., Canann S., Saigal S. (1998) "Advancing front quadrilateral meshing using triangle transformations," In proceedings of 7th International Meshing Roundtable, pp. 409–428.

[60] Blacker T., Stephenson M. (1991) "Paving: a new approach to automated quadrilateral mesh generation," Int. J. Numer. Meth. Engng., 32, pp. 811-847.

[61] Staten M., Kerr R., Owen S., Blacker T. (2006) "Unconstrained paving and plastering: progress update," In proceedings of 15th International Meshing Roundtable, pp. 469-486.

[62] Maza S., Noel F., Leon J. (1999) "Generation of quadrilateral meshes on free-form surfaces," Computers and Structures, 71, pp. 505-524.

[63] Lai Y.-K., Kobbelt L., Hu S.-M. (2010) "Feature aligned quad dominant remeshing using iterative local updates," Computer-Aided Design, 42, pp. 109-117.

[64] Sarrate J., Huerta A. (2000) "Efficient unstructured quadrilateral mesh generation," Int. J. Numer. Meth. Engng., 49, pp. 1327-1350.

[65] Bastian M., Li B. (2003) "An efficient automatic mesh generator for quadrilateral elements implemented using C++," Finite Elements in Analysis and Design, 39, pp. 905-930.

[66] Egidi N., Maponi P. (2008) "Block decomposition techniques in the generation of adaptive grids," Mathematics and Computers in Simulation, 78, pp. 593-604.

[67] George P.-L., Borouchaki H. (1998) "Delaunay triangulation and meshing: application to finite elements," Hermes.

[68] Shimada K., Gossard D. (1998) "Automatic triangular mesh generation of trimmed parametric surface for finite element analysis," Computer Aided Geometric Design, 15, pp. 199-222.

[69] Lo S., Wang W. (2005) "Generation of finite element mesh with variable size over an unbounded 2D domain," Comput. Methods Appl. Mech. Engrg, 194, pp. 4668-4684.

[70] Wang W., Ming C., Lo S. (2007) "Generation of triangular mesh with specified size by circle packing," Advances in Engineering Software, 38, pp. 133-142.

[71] Park C., Noh J.-S., Jang I.-S., Kang J. (2007) "A new automated scheme of quadrilateral mesh generation for randomly distributed line constraints," Computer-Aided Design, 39, pp. 258-267.

[72] Lee K.-Y., Kim I.-I., Cho D.-Y., Kim T.-W. (2003) "An algorithm for automatic 2D quadrilateral mesh generation with line constraints," Computer-Aided Design, 35, pp. 1055-1068.

[73] Mukherjee N. (2006) "High quality bi-linear transfinite meshing with interior point constraints". In proceedings of 15th International Meshing Roundtable, pp. 309-323.

[74] Topping B., Cheng B. (1999) "Parallel and distributed adaptive quadrilateral mesh generation," Computers and Structures, 73, pp. 519-536.

[75] Sziveri J., Cheng B., Bahreininejad A., Cai J., Thierauf G., Topping B. (1999) "Parallel quadrilateral subdomain generation for finite element analysis," Advances in Engineering Software, 30, pp. 809-823.

[76] Lammer L., Burghardt M. (2000) "Parallel generation of triangular and quadrilateral meshes," Advances in Engineering Software, 31, pp. 929-936.

[77] Ito Y., Shih A., Erukala A., Soni B., Chernikov A., Chrisochoides N., Nakahashi K. (2007) "Parallel unstructured mesh generation by an advancing front method," Mathematics and Computers in Simulation, 75, pp. 200-209.

[78] Cannan S., Tristano J., Staten M. (1998) "An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes," In proceedings of 7th International Meshing Roundtable, pp. 479-494.

[79] Xu H., Newman T. (2006) "An angle-based optimization approach for 2D finite element mesh smoothing," Finite Elements in Analysis and Design, 42, pp. 1150-1164.

[80] Zhou T., Shimada K. (2000) "An angle-based approach to two-dimensional mesh smoothing," In proceedings of 9th International Meshing Roundtable, pp. 373-384.

[81] Garimella R., Shashkov M., Knupp P. (2004) "Triangular and quadrilateral surface mesh quality optimization using local parameterization," Comput. Methods Appl. Mech. Engrg, 193, pp. 913-928.

[82] Schneider R., Kobbelt L. (2001) "Geometric fairing of irregular meshes for free-form surface design," Computer Aided Geometric Design, 18, pp. 359-379.

[83] Liu L., Tai C.-L., Ji Z., Wang G. (2007) "Non-iterative approach for global mesh optimization," Computer-Aided Design, 39, pp. 772-782.

[84] Potra F., Anitescu M., Gavrea B., Trinkle J. (2006) "A linearly implicit trapezoidal method for integrating stiff multibody dynamics with contact, joints and friction," Int. J. Numer. Meth. Engng, 66, pp. 1079-1124.

[85] Funk K., Pfeiffer F. (2003) "A time-stepping algorithm for stiff mechanical systems with unilateral constraints," Proceedings in Applied Mathematics and Mechanics, 2, pp. 228-229.

[86] Arnold M., Fuchs A., Fuhrer C. (2006) "Efficient corrector iteration for DAE time integration in multibody dynamics," Comput. Methods Appl. Mech. Engrg, 195, pp. 6958-6973.

[87] Baraff D. (1997) "Implicit methods: how to not blow up," ACM Transactions on Graphics (SIGGRAPH 1997)

[88] Hughes T. (1987) "The finite element method: linear static and dynamic finite element analysis," Prentice-Hall.

[89] Chentanez N., Alterovitz R., Ritchie D., Cho L., Hauser K. (2009) "Interactive simulation of surgical needle insertion and steering," ACM Transactions on Graphics (SIGGRAPH 2009)

[90] Kane C., Marsden J., Ortiz M., West M. (2000) "Variational integrators and the Newmark algorithm for conservative and dissipative mechanical systems," Int. J. Numer. Meth. Engng, 49, pp. 1295-1325.

[91] Stewart D., Trinkle J. (1996) "An implicit time-stepping scheme for rigid body dynamics with Coulomb friction," Int. J. Numer. Meth. Engng, 39, pp. 2673-2691.

[92] Anitescu M., Potra F., Stewart D. (1999) "Time-spepping for three-dimensional rigid body dynamics," Comput. Methods Appl. Mech. Engrg, 177, pp. 183-197.

[93] Bae D., Lee J., Cho H., Yae H. (2000) "An explicit integration method for realtime simulation of multibody vehicle models," Comput. Methods Appl. Mech. Engrg, 187, pp. 337-350.

[94] Guendelman E., Bridson R., Fedkiw R. (2003) "Nonconvex rigid bodies with stacking," ACM Transactions on Graphics (SIGGRAPH 2003)

[95] Uhlar S., Betsch P. (2010) "On the derivation of energy consistent time stepping schemes for friction afflicted multibody systems," Computers and Structures, 88, pp. 737-754.

[96] Drumwright E. (2008) "A fast and stable penalty method for rigid body simulation," IEEE Transactions on Visualization and Computer Graphics, 14, pp. 231-240.

[97] Wu D. (2000) "Penalty methods for contact resolution," Game Developers Conference.

[98] Baumgarte J. (1972) "Stabilization of constraints and integrals of motion in dynamical systems," Comput. Methods Appl. Mech. Engrg, 1, pp. 1–16.

[99] Braun D., Goldfarb M. (2009) "Eliminating constraint drift in the numerical simulation of constrained dynamical systems," Comput. Methods Appl. Mech. Engrg, 198, pp. 3151–3160.

[100] Parida N., Raha S. (2009) "Regularized numerical integration of multibody dynamics with the generalized alfa method," Applied Mathematics and Computation, 215, pp. 1224-1243.

[101] Blajer W. (1999) "State and energy stabilization in multibody system simulations," Mechanics Research Communications, 26, pp. 261-268.

[102] Cline M., Pai D. (2003) "Post-stabilization for rigid body simulation with contact and constraints," In Proceedings of IEEE International Conference on Robotics and Automation.

[103] Muller M., Heidelberger B., Hennix M., Ratcliff J. (2006) "Position based dynamics," In Proceedings of Virtual Reality Interactions and Physical Simulation.

[104] Muller M. (2008) "Hierarchical position based dynamics," In Proceedings of Virtual Reality Interactions and Physical Simulation.

[105] Glocker C., Pfeiffer F. (1993) "Complementary problems in mutibody systems with planar friction," Archive of Applied Mechanics, 63, pp. 452-463.

[106] Pfeiffer F., Glocker C. (2000) "Contact in mutibody systems," J. Appl. Maths. Mechs, 64, pp. 773-782.

[107] Pfeiffer F. (2001) "Multibody systems with unilateral constraints," J. Appl. Maths. Mechs, 65, pp. 665-670.

[108] Pfeiffer F., Foerg M., Ulbrich H. (2006) "Numerical aspects of non-smooth multibody dynamics," Comput. Methods. Appl. Mech. Engrg, 195, pp. 6891-6908.

[109] Pfeiffer F., Glocker C. (2004) "Multibody dynamics with unilateral contacts," Wiley-VCH.

[110] Featherstone R. (2008) "Rigid body dynamics algorithms," Springer.

[111] Baraff D. (1996) "Linear-time dynamics using Lagrange multipliers," ACM Transactions on Graphics (SIGGRAPH 1996)

[112] Baraff D. (1997) "An introduction to physically based modeling: rigid body simulation," ACM Transactions on Graphics (SIGGRAPH 1997).

[113] Tseng F.-C., Ma Z.-D., Hulbert G. (2003) "Efficient numerical solution of constrained mutibody dynamics systems," Comput. Methods. Appl. Mech. Engrg, 192, pp. 439-472.

[114] Anitescu M., Potra F. (1997) "Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems," Nonlinear Dynamics, 14, pp. 231-247.

[115] Trinkle J., Pang J.-S., Sudarsky S., Lo G. (1997) "On dynamic multi-rigid-body contact problems with Coulomb friction," Zeitschrift fur Angewandte Mathematik und Mechanik, 77, pp. 267-279.

[116] Sauer J., Schomer E. (1998) "A constraint-based approach to rigid body dynamics for virtual reality applications," Proceedings of the ACM Symposium on Virtual Reality Software and Technology, pp. 153-162.

[117] Kaufman D., Edmunds T., Pai D. (2005) "Fast frictional dynamics for rigid bodies," ACM Transactions on Graphics (SIGGRAPH 2005).

[118] Kaufman D., Sueda S., James D., Pai D. (2008) "Staggered projections for frictional contact in multibody systems," ACM Transactions on Graphics (SIGGRAPH Asia 2008).

[119] Mirtich B. (1996) "Impulse-based dynamic simulation of rigid body systems," Ph.D. thesis, University of California, Berkeley.

[120] Bender J., Finkenzeller D., Schmitt A. (2005) "An impulse-based dynamic simulation system for VR applications," Proceedings of Virtual Concept 2005.

[121] Bender J., Schmitt A. (2006) "Constraint-based collision and contact handling using impulses," In Proceedings of the 19th International Conference on Computer Animation and Social Agents.

[122] Bender J. (2008) "Design of a dynamic simulation system for VR applications," Technical Report 2008-13.

[123] Bender J., Bayer D., Diziol R. (2009) "Dynamic simulation of inextensible cloth," IADIS International Journal on Computer Science and Information Systems, 4, pp. 86-102.

[124] Bender J., Schmitt A. (2006) "Fast dynamic simulation of multi-body systems using impulses," In Proceedings of Virtual Reality Interactions and Physical Simulations.

[125] Schmitt A., Bender J. (2005) "Impulse based dynamic simulation of multibody systems: numerical comparison with standard methods," In Proceedings of Automation of Discrete Production Engineering 2005.

[126] Bender J. (2007) "Impulse-based dynamic simulation in linear time," Journal of Computer Animation and Virtual Worlds.

[127] Schmitt A., Bender J., Prautzsch H. (2005) "Impulse-based dynamic simulation of higher order and numerical results," Technical Report 2005-21.

[128] Bullet physics library. < http://bulletphysics.org >.

[129] Open dynamics engine. < http://www.ode.org >.

[130] Box2D physics engine. < http://www.box2d.org >.

[131] Schiehlen W. (2006) "Computational dynamics: theory and applications of multibody systems". European Journal of Mechanics A/Solids, 25, pp. 566-594.

[132] Servin M., Lacoursiere C., Melin N. (2006) "Interactive simulation of elastic deformable materials," ACM Transactions on Graphics (SIGGRAPH 2006).

[133] Ebrahimi S., Eberhard P. (2007) "Aspects of contact problems in computational multibody dynamics," In E. Onate, editor, Multibody dynamics: Computational Methods and Applications, Springer, pp. 23-47.

[134] Lemke C. (1965) "Bimatrix equilibrium points and mathematical programming," Management Science, 11, pp. 681-689.

[135] Hecker C. (2004) "Lemke's algorithm: the hammer in your math toolbox?," GDC 2004.

[136] Tasora A. (2006) "An iterative fixed-point method for solving large complementarity problems in multibody systems," In Proceedings of XVI GIMC.

[137] Tasora A., Anitescu M. (2008) "A fast NCP solver for large rigid-body problems with contacts, friction and joints," In C.L. Bottasso, editor, Multibody Dynamics: Computational Methods and Applications, Springer, pp. 45-55.

[138] Catto E. (2005) "Iterative dynamics with temporal coherence," GDC 2005.

[139] Lacoursiere C. (2008) "Using Gauss-Seidel for multibody problems".

[140] Poulsen M., Niebe S., Erleben K. (2010) "Heuristic convergence rate improvements of the projected Gauss-Seidel method for frictional contact problems," 18th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG 2010: Full Paper Proceedings, pp. 135 – 142.

[141] Catto E. (2009) "Modeling and solving constraints," GDC 2009.

[142] Erleben K. (2004) "Stable, robust, and versatile multibody dynamics animation," Ph.D. thesis, University of Copenhagen.

[143] Garstenauer H. (2006) "A unified framework for rigid body dynamics," Ph.D. thesis, Johannes Kepler University of Linz.

[144] Shabana A. (2001) "Computational dynamics," Wiley-Interscience.

[145] Wannenburg J., Heyns P., Raath A. (2009) "Application of a fatigue equivalent static load methodology durability of heavy vehicle structures," International Journal of Fatigue. 31, pp. 1541-1549.

[146] Zoroufi M. (2004) "Manufacturing process effects on fatigue design and optimization of automotive components – An analytical and experimental study," PhD thesis. University of Toledo.

[147] Boessio M., Morsch I., Awruch A. (2006) "Fatigue liferime estimation of commercial vehicles," Journal of Sound and Vibration, 291, pp. 169-191.

[148] Abdullah S., Al-Asady N., Ariffin A., Rahman M. (2008) "A review on finite element of automotive components," Journal of Applied Sciences, 8, pp. 2192-2201.

[149] Walton D., Prayoonrat S., Taylor S. (1986) "Computer-aided fatigue analysis," Computer-Aided Design. 18, pp. 263-274.

[150] Ong J. (1987) "Fatigue-life predictions on microcomputers," Computer-Aided Design. 19, pp. 95-99.

[151] Aspragathos N. (1988) "Program for torsional dynamic modeling of drive systems and fatigue life prediction," Computer-Aided Design. 20, pp. 555-563.

[152] Hancq D., Walters A., Beuth J. (2000) "Development of an object-oriented fatigue tool," Engineering with Computers. 16, pp. 131-144.

[153] Osborne G., Prater G., Lesiv R., Lamb D., Castanier M. (2011) "An interactive design space supporting development of vehicle architecture concept models," ASME 2011 International Mechanical Engineering Congress and Exposition, paper no. IMECE2011-64510.

[154] Mitchell S. (1997) "Choosing corners of rectangles for mapped meshing," In Proceedings of the 13th Annual Symposium on Computational Geometry, pp. 87-93.

[155] Kacic-Alesic Z., Nordenstam M., Bullock D. (2003) "A practical dynamics system," ACM Transactions on Graphics (SIGGRAPH 2003).

[156] Catto E. (2009) "Numerical integration," GDC 2009.

[157] Bergen G. (2005) "Ray casting against general convex objects with application to continuous collision detection," GDC 2005.

[158] Grootjans R. (2009) "XNA 3.0 Game programming recipes: A problem solution approach," Apress.

[159] Negrut D., Ortiz J. (2006) "A practical approach for the linearization of the constrained multibody dynamics equations," ASME J. Comput. Nonlin. Dyn, 1, pp. 230-239.

[160] Shigley J., Mischke C. (1989) "Mechanical Engineering Design," 5th edition, McGraw-Hill, Inc.

[161] Shigley J., Mischke C. (2004) "Standard Handbook of Machine Design," 3rd edition, McGraw-Hill, Inc.

APPENDICES

APPENDIX A

List of Acronyms

ACD-ICAE: Auto-body Concept Design-Intelligent Computer Aided Engineering

API: Application Programming Interface

BEV: Battery Electric Vehicle

CAD: Computer Aided Design

CAE: Computer Aided Engineering

CMTS: Concept Modeling Tool Suite

DFT: Discrete Fourier Transformation

DOF: Degree of Freedom

EV: Electric Vehicle

FCEV: Fuel Cell Electric Vehicle

FCHEV: Fuel Cell Plug-in Hybrid Electric Vehicle

FE: Finite Element

FEA: Finite Element Analysis

FHWA: Federal Highway Administration

FOA: First Order Analysis

GVWR: Gross Vehicle Weight Rating

HEV: Hybrid Electric Vehicle

IC: Internal Combustion

ICE: Internal Combustion Engine

ICEV: Internal Combustion Engine Vehicle

MCJ: Major Compliance Joint

MKG: Mass, Kinematic, and Geometric

MLCP: Mixed Linear Complementary Problem

NURBS: Non-uniform Rational B-spline

NVH: Noise, Vibration, and Harshness

OOD: Object Oriented Design

OOP: Object Oriented Programming

PDE: Partial Differential Equation

PGS: Projected Gauss-Seidel

RB: Rigid Body

RBA: Rigid Body Analysis

SI: Sequential Impulse

SKI: Spatial, Kinematic, and Inertia

TFI: Transfinite Interpolation

TOI: Time of Impact

UML: Unified Modeling Language

CURRICULUM VITAE

Rostyslav M. Lesiv

University of Louisville
Louisville, KY, USA
rmlesi01@louisville.edu

Education

8/08-present University of Louisville Department of Mechanical Engineering, Louisville, KY. Ph.D. in Mechanical Engineering expected December 2012. GPA: 3.97/4.0.

9/01-7/08 Ivan Franko National University of Lviv, Department of Mechanics and Mathematics, Lviv, Ukraine.
B.Sc. in Mechanical Engineering June 2005. GPA: 3.9/4.0
M.Eng. June 2006. GPA: 4.0/4.0

Experience

8/08-present University of Louisville, Louisville, KY. Graduate Research Assistant, Department of Mechanical Engineering. CMTS Project Research. Developed fatigue analysis engine. Methodology and implementation of free-form surface modeling. Methodology and implementation of free-form trimmed composit surface meshing. Methodology and implementation of free-form trimmed surface tesselation for the rendering engine. Methodology and implementation of multibody dynamics engine.

10/06-7/08 Ivan Franko National University of Lviv, Lviv, Ukraine. Graduate Research Assistant, Department of Mechanics and Mathematics. Investigated different aspects of crack growth in metals subjected to combined action of creep and fatigue. Developed and tested the approach for estimation of residual life-time of thin-walled constructions.

6/06-9/06 Freeze Pro Software, Lviv, Ukraine. Junior Software Engineer. Developed Windows Form User Controls.

8/05-5/06 Ivan Franko National University of Lviv, Lviv, Ukraine. Graduate Research and Thesis. Developed the mathematical model for estimation of the creep-fatigue crack growth rate in metals.

1/05-7/05 Ivan Franko National University of Lviv, Lviv, Ukraine. Undergraduate Theses. Mathematical modeling of stress-strain distribution in the vicinity of wedge composites. Built an analytical solution using the linear theory of elasticity.

Publications

Lesiv R., Prater G., Osborne G., Lamb D., Castanier M. (2011) "Rigid body analysis models derived from vehicle architecture abstractions," ASME 2011 International Mechanical Engineering Congress and Exposition, paper no. IMECE2011-63613.

Osborne G., Prater G., Lesiv R., Lamb D., Castanier M. (2011) "An interactive design space supporting development of vehicle architecture concept models," ASME 2011 International Mechanical Engineering Congress and Exposition, paper no. IMECE2011-64510.

Osborne G., Prater G., Lesiv R., Lamb D., Castanier M. (2011) "Vehicle concept model abstractions for integrated geometric, inertial, rigid body, powertrain, and FE analyses," ASME 2011 International Mechanical Engineering Congress and Exposition, paper no. IMECE2011-63590.

Lesiv R., Prater G., Nester V. (September 2011) "Stochastic fatigue analysis engine for vehicle structural assemblies design," II International Symposium "Modern Problems of Mechanical Engineering," Lutsk, Ukraine, pp. 14-17.

Lesiv R., Prater G. (September 2011) "Rigid body physics engine for concept vehicle dynamics simulation," International Skorobohat'ko Mathematical Conference, Mini-Symposium: "Computing mathematics and modern problems of mechanics, mathematical and theoretical physics," Drohobych, Ukraine.

Lesiv R., Prater G. (September 2010) "Free-form panel modeling and meshing for vehicle architecture abstractions," VIII International Conference on Mathematical Problems of Mechanics of Inhomogeneous Structures, Lviv, Ukraine.

Andrejkiv O., Lesiv R., Dolinska I. (2009) "Dependence of the period of subcritical growth of a creep-fatigue crack on the duration of loading cycles," Materials Science, 45, pp. 494-503.

Andrejkiv O., Lesiv R., Levytska N. (2009) "Crack growth in structural materials under the combined action of fatidue and creep (review)," Materials Science, 45, pp. 1-17.

Yu Y., Lesiv R., Ragade R. (2009) "An XNA-based form control for 3D modeling software," 14th International Conference on Computer Games, Louisville, KY, USA.

Publications - Continued

Lesiv R. (2008) "Lifetime of a plate with system of two-periodic straight-line creep-fatigue cracks," Visnyk LNU, 69, pp. 177-182.

Andrejkiv O., Lesiv R. (2008) "Mathematical model for estimating the period of creep-fatigue crack growth in plates," Mashynoznavstvo, 131, pp. 3-7.

Andrejkiv O., Lesiv R. (2008) "Estimating of residual resource of constructions elements with cracks under high temperature and variable loading," Conference on Modern Problems of Mechanics and Mathematics, Lviv, Ukraine.

Andrejkiv O., Lesiv R. (2007) "Mathematical model for estimating the period of creep-fatigue crack growth in construction materials at high temperature," Acta Mechanica et Automatica, 1, pp. 7-10.

Lesiv R. (2007) Estimation the period of sub-critical creep-fatigue crack growth in plates," XX Open Scientific and Technical Conference of Young Scientists and Specialists of Karpenko Physico-Mechanical Institute of the NAS of Ukraine, Lviv, Ukraine.

Andrejkiv O., Lesiv R. (2007) "Construction of dependence of sub-critical creep-fatigue crack growth period on length of loading cycle," VII Ukrainian-Poland Scientific Symposium, Lviv, Ukraine.

Andrejkiv O., Lesiv R. (2007) "Estimating the period of creep-fatigue crack growth in thin-walled elements of constructions," 8 International Symposium of Ukrainian Mechanical Engineers in Lviv, Lviv, Ukraine.

Andrejkiv O., Sas N., Kit M., Lesiv R. (2006) "Mathematical models for determination of sub-critical crack growth in structural elements under high temperature and long-term loading," VII International Scientific Conference, Lviv, Ukraine.

Teaching Experience

- Fall 2012: ME 206 Mechanics II - Dynamics. Full course responsibility.
- Summer 2012: ME 626 Vehicle Body Structure Design. Full course responsibility.
- Spring 2012: ME 280 Structured and Event-Driven Programming. Full course responsibility.

Honors and Awards

2012 Graduate Dean's Citations (PhD degree)

2006 Graduation with highest honors (B.Sc. and M.Eng degrees).

2006 First place, The All Ukrainian Competition of Student Works in Section "Mathematical Sciences" including Solids Mechanics and Fluid Mechanics, Ukraine.

2005 First place, The Ukrainian National Competition in Theoretical Mechanics, Ukraine.

2004 Second place, The Ukrainian National Competition in Theoretical Mechanics, Ukraine.

2003 Third place, The Ukrainian National Competition in Theoretical Mechanics, Ukraine.

Technical Skills

C#, ANSYS, Maple, Matlab, LabView