(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 July 2017 (06.07.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/117607 A1

(51) International Patent Classification:
G06F 17/50 (2006.01)

(21) International Application Number:
PCT/US2017/012049

(22) International Filing Date:
3 January 2017 (03.01.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/273,719 31 December 2015 (31.12.2015) US
62/273,767 31 December 2015 (31.12.2015) US

(71) Applicant: **ESI GROUP** [FR/FR]; 100-102 Avenue de Suffren, 75015 Paris (FR).

(72) Inventors: **SAHA, Bhaskar**; 701 Mendocino Way, Redwood City, CA 94065 (US). **BOBROW, Daniel, G.**; 376 Addison Avenue, Palo Alto, CA 94301 (US). **DE KLEER, Johan**; 685 Paco Drive, Los Altos, CA 94024 (US). **JANSSEN, William, C.**; 230 Jason Way, Mountain View, CA 94043 (US). **KURTOLGLU, Tolga**; 2762 Sunbonnet Court, San Jose, CA 95125 (US).

(74) Agent: **CORNELY, John, P.**; Fay Sharpe LLP, The Halle Building, 5th Floor, 1228 Euclid Avenue, Cleveland, OH 44115-1843 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

(54) Title: AUGMENTATION OF VIRTUAL MODELS

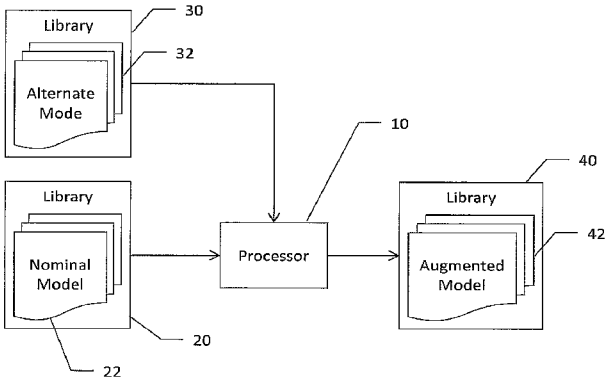

FIG. 1

(57) Abstract: A method is disclosed for automatically generating an augmented model of a physical component. The method includes: reading an input model into a processor, the input model describing a nominal mode of operation for a physical component modeled by the input model; parsing with the processor the input model to generate a parse thereof; analyzing with the processor the parse of the input model; and automatically writing with the processor an augmented model for the physical component from the input model based on the analysis, the augmented model describing both (i) the nominal mode of operation for the modeled physical component and (ii) at least one alternate mode of operation for the modeled physical component which is different from the nominal mode of operation. Suitably, the analyzing is conducted according to at least one of a first approach and a second approach. The first approach operates to detect if the input model is susceptible to one or more kind of fault including: (i) catastrophic, (ii) power flow, and (iii) parametric kinds of faults; and the second approach operates to match specific facets of a component's dynamic behavior, and appropriately modify those matched facets to reflect the dynamics of a fault mechanism.

WO 2017/117607 A1

Patent Application for:

AUGMENTATION OF VIRTUAL MODELS

[0001] The present application claims the benefit of U.S. Provisional Patent Application No. 62/273,719, file 12/31/2015 and U.S. Provisional Patent Application No. 62/273,767, filed 12/31/2015, both of which are incorporated herein by reference in their entirety.

BACKGROUND

[0002] The present inventive subject matter relates generally to the art of cyber-physical component and/or system modeling. Particular but not exclusive relevance is found in connection with object-oriented and/or other like modeling computer languages, e.g., such as Modelica, etc. The present specification accordingly makes specific reference thereto at times. However, it is to be appreciated that aspects of the present inventive subject matter are also equally amenable to other like applications and/or environments.

[0003] Modeling languages, e.g., such as Modelica, are commonly used to generate virtual models of and/or to simulate physical components and/or systems for design, analysis and/or testing purposes. For example, the physical systems and/or components that are modeled may be fairly complex or rather simple and can include, e.g., electrical circuits or automotive vehicles or other physical systems. For example, the open source Modelica Standard Library (MSL) includes models of approximately 1280 components in domains, e.g., such as electronics, mechanics and fluids. Cyber-physical components are built from or depend upon the synergy of computational and physical components, such as system controllers that implement control software on hardware platforms, as found in vehicles, assembly lines and other engineered systems. For example, the Modelica_LinearSystems2 library includes models for continuous and discrete controllers. The MSL itself includes models for different variants of Proportional, Integral and Differential (PID) controllers. These Modelica models and the like have come to be widely used more and more in various projects. In this disclosure, the term "physical component" is used to mean either or both "physical" and "cyber-physical component".

[0004] In practice, however, the MSL models and the like generally describe and/or represent only one mode of operation for the component being modeled, namely, the nominal or correct behavior of the component being modeled. That is to say, such models commonly do not describe faulty or other non-nominal behaviors. For example, a fault mode may be a short or open circuit or a motor that fails to run or a slipping brake, etc. Any number of difference mechanisms, e.g., such as wear, material fatigue, corrosion, etc., may lead to the faulty operation of a modeled component. Often, it may be desirable to test and/or analyze components operating in faulty or otherwise non-nominal modes, e.g., for diagnostic purposes. However, insomuch as conventional MSL models and the like generally do not describe such faulty and/or otherwise non-nominal modes, diagnostics and/or other testing of modeled components operating in such faulty and/or otherwise non-nominal modes can be problematic.

[0005] Furthermore, damage process models express damage due to failure mechanisms that are active in system components because of their service environments as a function of, for example, such parameters as loads, materials' properties, and dimensions. Running reliability simulations under specific system configurations and usage conditions can be overly time and/or labor intensive.

[0006] One work related to the present specification, which is explicitly incorporated herein by reference in its entirety, is U.S. Patent Application No. 13/967,503, filed August, 15, 2013, of Saha, et al.

[0007] In general, a new and/or improved method and/or system or apparatus is disclosed herein for automatically and/or semi-automatically generating component models including non-nominal behavior modes. Additionally, it allows investigation of the effect of component fault at the system level that may possibly be used to improve the system design in terms of robustness, resiliency and reliability.

BRIEF DESCRIPTION

[0008] This Brief Description is provided to introduce concepts related to the present inventive subject matter. The summary is not intended to identify essential features of the claimed subject matter nor is it intended for use in determining or limiting the scope of the claimed subject matter. The embodiments described below are not intended to be

exhaustive or to limit the invention to the precise forms disclosed in the following detailed description. Rather, the embodiments are chosen and described so that others skilled in the art may appreciate and understand the principles and practices of the present inventive subject matter.

[0009] In accordance with one embodiment, a method is provided for automatically generating an augmented model of a physical component. The method includes: reading an input model into a processor, the input model describing a nominal mode of operation for a physical component modeled by the input model; parsing with the processor the input model to generate a parse thereof; analyzing with the processor the parse of the input model; and automatically writing with the processor an augmented model for the physical component from the input model based on the analysis, the augmented model describing both (i) the nominal mode of operation for the modeled physical component and (ii) at least one alternate mode of operation for the modeled physical component which is different from the nominal mode of operation. Suitably, the analyzing is conducted according to at least one of a first approach and a second approach. The first approach operates to detect if the input model is susceptible to one or more kind of fault including: (i) catastrophic, (ii) power flow, and (iii) parametric kinds of faults; and the second approach operates to match specific facets of a component's dynamic behavior, and appropriately modify those matched facets to reflect the dynamics of a fault mechanism.

[0010] In accordance with another embodiment, a system is provided for automatically generating an augmented model of a physical component. The system includes: a processor operative to: read in an input model, the input model describing a nominal mode of operation for a physical component modeled by the input model; parse the input model to generate a parse thereof; analyze the parse of the input model; and automatically write an augmented model for the physical component from the input model based on the analysis, the augmented model describing both (i) the nominal mode of operation for the modeled physical component and (ii) at least one alternate mode of operation for the modeled physical component which is different from the nominal mode of operation. Suitably, analyzing the parse is conducted according to at least one of a first approach and a second approach. The first approach operates to detect if the input model is

susceptible to one or more kind of fault including: (i) catastrophic, (ii) power flow, and (iii) parametric kinds of faults; and the second approach operates to match specific facets of a component's dynamic behavior, and appropriately modify those matched facets to reflect the dynamics of a fault mechanism.

[0011] Numerous advantages and benefits of the inventive subject matter disclosed herein will become apparent to those of ordinary skill in the art upon reading and understanding the present specification. It is to be understood, however, that the detailed description of the various embodiments and specific examples, while indicating preferred and/or other embodiments, are given by way of illustration and not limitation. Many changes and modifications within the scope of the present invention may be made without departing from the spirit thereof, and the invention includes all such modifications.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The following detailed description makes reference to the figures in the accompanying drawings. However, the inventive subject matter disclosed herein may take form in various components and arrangements of components, and in various steps and arrangements of steps. The drawings are only for purposes of illustrating exemplary and/or preferred embodiments and are not to be construed as limiting. Further, it is to be appreciated that the drawings may not be to scale.

[0013] FIGURE **1** is a diagrammatic illustration showing an exemplary model transformation system suitable for practicing aspects of the present inventive subject matter.

[0014] FIGURE **2** is a flow chart showing an exemplary method for performing a model transformation process in accordance with aspects of the present inventive subject matter.

[0015] FIGURE **3** shows an exemplary workflow in accordance with aspects of the present inventive subject matter, using MBD analysis for determining component fatigue life conditioned on mission usage and design configuration.

[0016] FIGURE **4** shows a graphical representation of a simulation test course, with the course height along the vertical axis (in inches) and the course distance along the horizontal axis (in inches).

[0017] FIGURES **5A** and **5B** illustrate sample, acceleration time histories for a trench crossing and 1 meter wall step down.

[0018] FIGURE **6** shows a sample von Mises stress contour of an exemplary 4 x 4 x 4 bracket.

[0019] FIGURES **7, 8** and **9** show the maximum accelerations at the bracket in X, Y, and Z directions respectively as functions of speed and bracket stiffness.

[0020] FIGURES **10, 11** and **12** show the percentage errors for the maximum lateral, longitudinal and vertical accelerations as a function of vehicle speed and Power Pack mass scaling factor.

DETAILED DESCRIPTION

[0021] For clarity and simplicity, the present specification shall refer to structural and/or functional elements, relevant standards, algorithms and/or protocols, and other components, algorithms, methods and/or processes that are commonly known in the art without further detailed explanation as to their configuration or operation except to the extent they have been modified or altered in accordance with and/or to accommodate the preferred and/or other embodiment(s) presented herein. Moreover, the apparatuses and methods disclosed in the present specification are described in detail by way of examples and with reference to the figures. Unless otherwise specified, like numbers in the figures indicate references to the same, similar or corresponding elements throughout the figures. It will be appreciated that modifications to disclosed and described examples, arrangements, configurations, components, elements, apparatuses, methods, materials, etc. can be made and may be desired for a specific application. In this disclosure, any identification of specific materials, techniques, arrangements, etc. are either related to a specific example presented or are merely a general description of such a material, technique, arrangement, etc. Identifications of specific details or examples are not intended to be, and should not be, construed as mandatory or limiting unless specifically designated as such. Selected examples of apparatuses and methods are hereinafter disclosed and described in detail with reference made to the figures.

[0022] In general, in accordance with at least one embodiment, there is disclosed herein a method, system and/or apparatus which is operative to augment models of

physical system components with fault and/or other non-nominal modes. Suitably, models including the fault and/or other non-nominal modes are automatically or semi-automatically generated from models of the components which otherwise describe only the nominal or correct behaviors of the components. Note, as used herein FAME is an acronym for Fault-Augmented Model Extension. However, it is to be appreciated that the augmented models may be extended as described herein to include other alternate modes that are not "fault" related strictly speaking, but are rather related to other non-nominal or alternate operating modes.

[0023] In accordance with one suitable embodiment, existing open source libraries (e.g., the Modelica Standard Library (MSL), libraries developed by third parties and other libraries) are extended to allow the compact introduction of fault models while preserving the availability of existing nominal behavior specifications when no fault has been introduced into a specific instance of a model. More specifically, there is created a basic set of non-nominal behavioral models that scale up to the libraries of nominal component models, e.g., in the MSL, libraries developed by third parties and/or other like libraries. The methodology allows fault models to be inserted into new components created by various third parties and/or other Modelica library developers.

[0024] The FAME model transformation process leverages the JModelica Modelica parser framework, and the JastAdd technology on which it is built, to inject faults into the nominal component model library. A Java program incorporating JastAdd and JModelica runs over the supplied library, recognizes fault-susceptible component models, re-writes them as appropriate to provide fault behavior, and outputs the modified library to a new location.

[0025] The following is a simple model class in Modelica, for an electrical capacitor, which is susceptible to both *ElectricalOpen* and *ElectricalShort* faults.

```
model Capacitor "Ideal linear electrical capacitor"
     extends Modelica.Electrical.Analog.Interfaces.OnePort;
     parameter Modelica.SIunits.Capacitance C(start=1)
     "Capacitance";
     equation
          i = C*der(v);
end Capacitor;
```

[0026] The forgoing is for example an un-augmented or simple capacitor model.

[0027] To analyze the forgoing, a suitable embodiment operates to detect or otherwise determine what faults it is susceptible to, and rewrite it so that these faults could be simulated in instances of this class. In general, an exemplary embodiment operates to look for three classes of faults: (i) catastrophic, (ii) power flow, and (iii) parametric. Catastrophic faults typically change the dynamics of a component into something completely different. For example, in the electrical domain, the common catastrophic fault is an open circuit. Power flow faults affect one or more of the power flows to or from the component. For example, in electrical components, a common power flow fault is a short circuit of some magnitude. Parametric faults typically reflect the shifting of a supposedly fixed parameter. For example, in the case of a capacitor, the capacitance.

[0028] Once the faults have been identified to which a component is susceptible, then behavioral descriptions are added to that component for each of the identified faults. The model can then be run in any of these fault modes, and for non-catastrophic faults, the severity of the fault can be specified.

[0029] With respect to fault injection, one suitable approach relies on understanding patterns of power flow through a system built with the components, and has the advantage of working with models specified quite abstractly, and in the face of modeling inconsistencies. This approach is described as an "external" approach below. Suitably, the external approach may be augmented with a manually compiled list of declarative specification of desired parametric faults, which are injected automatically. This allows modeling of faults more specifically and precisely.

[0030] Another suitable approach relies on matching specific facets of a component's dynamic behavior, and appropriately modifying those facets to reflect the dynamics of fault mechanisms. In contrast to the external approach, this method is referred to as an "internal" approach. It can provide more detailed modeling of the fault behaviors, but it can be more difficult to specify accurately.

[0031] JModelica is an open-source Modelica tool chain. JModelica consists of a Java/JastAdd parser for the Modelica language, along with various simulation and analysis tools, written in both Java and Python.

[0032] JastAdd is a Java-based implementation of Knuth's *attribute grammars*, in which nodes in the abstract syntax tree (AST) can have *attributes*, the values of which are defined by *equations*. Each attribute can be either *synthesized* (defined by an equation attached to the node itself), or *inherited* (defined by an equation in an ancestor node). JastAdd supports *reference* attributes, which is an attribute that has as its value another node in the AST. This allows arbitrary graphs to be *woven* through the AST. It also allows its attributes to be *parameterized*, which means that the *equation* has unbound variables which bind to parameters when the equation is evaluated. It also allows *collection* attributes; multi-valued attributes the value of which can be contributed to by various other AST nodes. And it supports *circular* reference attributes, which provide a way of breaking referential cycles.

[0033] JastAdd is Java-based, so *equations* are implemented as Java methods, and attribute access is via calls to those methods. It also extends Java itself with various constructs, notably *aspects*. JastAdd aspects support *intertype declarations* for AST classes. An intertype declaration is a declaration that appears in an aspect file, but that actually belongs to an AST class (like an attribute equation). The JastAdd system reads the aspect files and weaves the intertype declarations into the appropriate AST classes. It supports both *declarative* aspects (.jrag files), and *imperative* aspects (.jadd files). Declarative aspects add new attributes, equations, and rewrites; imperative aspects add only Java methods and variables. Rewrites of the AST can also be specified; they replace an AST node of type A with a node of type B, optionally only when some condition C is true. New AST subtrees can be created and specified as attribute values ("non-terminal attributes").

[0034] In one suitable embodiment, the so called external approach, the JModelica parser is used to parse each library file, then examine each model class (non-partial Modelica class definitions with the restriction *model*) found in the parse tree for instances of the following connector classes:

```
Modelica.Electrical.Analog.Interfaces.Pin
Modelica.Mechanics.Rotational.Interfaces.Flange_a
Modelica.Mechanics.Rotational.Interfaces.Flange_b
```

```
Modelica.Mechanics.Translational.Interfaces.Flange_a
Modelica.Mechanics.Translational.Interfaces.Flange_b
Modelica.Fluid.Interfaces.FluidPort
Modelica.Thermal.HeatTransfer.Interfaces.HeatPort
```

[0035] These are instances of Modelica connector classes which contain two variables, one of an *effort* type, such as voltage or pressure, and another of a *flow* type, such as current or mass flow rate. The analysis suitably incorporates both direct components of the model class, as well as components of each inherited class used by the model class. If any of the above components are found, the parse tree for a replacement class for the model class is created, and the model class's parse tree is replaced with that new parse tree.

[0036] Finally, the JModelica "FormattedPrettyPrint" aspect is used to recreate the now-fault-enabled Modelica source code for the model from the AST.

[0037] In another exemplary embodiment (i.e., the internal approach), each generic fault or alternate mode, e.g., such as an "electrical short", is modeled with an aspect which provides a subclass of a more generic class, e.g., "Fault". Each Fault or alternate mode subclass provides a predicate, e.g., such as:

```
static boolean may_occur_with (FullClassDecl klass);
```

which, given a FullClassDecl (a node in the JModelica AST which represents the definition of a Modelica class), says whether that FullClassDecl might be susceptible to this fault (or the alternate mode may otherwise be applicable).

[0038] Suitably, the implementation of the "may_occur_with" predicate is dependent on matching some pattern defined for the fault/alternate mode against the parse of the model, as given by the FullClassDecl node of the AST.

[0039] In addition, each Fault or alternate mode subclass suitably contributes an instance of itself to the "possibleFaults" collection attribute on FullClassDecl, if that FullClassDecl is susceptible to it, using, e.g., the JastAdd "contributes" mechanism. For instance, for a Fault subclass ElectricalShort, it might look like the following:

```
FullClassDecl contributes (ElectricalShort(this))
        when (ElectricalShort.may_occur_with(this))
        to FullClassDecl.possibleFaults()
        for this;
```

[0040] In practice, more fault/alternate modes can be added by adding aspects to the collection of fault aspects, and re-compiling the program.

[0041] Suitably, to inject the faults and/or alternate modes, JModelica FullClassDecl node type is extended to include a number of additional attributes, e.g., in the following manner:

```
coll Set <Fault> FullClassDecl.possibleFaults()
        [new HashSet<Fault>()] with add root List;

syn lazy boolean ClassDecl.faultEligible() = false;

syn lazy boolean FullClassDecl.faultEligible() {
        if (isUnknown() || hasPartial()
                || !(getRestriction().isBlock() || getRestriction().isModel())) {
                return false;
        }
        return (possibleFaults().size() > 0);
}
```

[0042] Then, the "rewrite" capability of JastAdd, for example, may be used to re-write the AST representing the model, the FullClassDecl node, to add fault behavior to that model. For example, the rewrite rule may look something like the following:

```
rewrite FullClassDecl {

    when (faultEligible()) // faults possible for this type?
```

```
        to FullClassDecl {
            for (Fault f : possibleFaults()) {
                // add an enumeration value for each fault this component
                // is susceptible to
                List<ComponentDecl> fault_comps = f.fault_components();
                if (fault_comps != null)
                    for (ComponentDecl c : fault_comps) {
                        addComponentDeclNoTransform(c);
                    }
            }
            // rewrite the equations as an if-equation
            setEquationList(rewriteEquationsToAddFaults());
            return this;
        }
}
```

[0043] Suitably, this rewrite may be applied automatically as part of the parsing process, where appropriate, which will result in the fault/alternate behavior clauses being injected into the parse of the model. In practice, the equation re-writing method, rewriteEquationsToAddFaults(), takes the nominal equations for the model, and wraps them in one branch of a Modelica if-equation for that case, then adds additional branches to the if-equation for each possible fault/alternate mode, along with the equations for that fault/alternate mode. In one suitable embodiment, those fault/alternate equations may be obtained by calling a method on the Fault and/or alternate subclass instance. In Modelica, an if-equation is a common way of implementing a conditional equation.

[0044] Finally, the JModelica "FormattedPrettyPrint" aspect and/or capability, for example, may be used to recreate the now-fault-enabled Modelica source code for the model from the AST.

[0045] For example, in the case of either approach, in one suitable embodiment, the overall program for injecting faults and/or alternate modes into the models may be something like the following (in pseudo-code):

```
for file in recurseOver (libraryTree):
    outputFile = figureOutputName (outputDir, file)
    parsedVersion = parseFile (file)
    prettyPrint (parsedVersion, outputFile)
```

[0046] Once the faults and/or alternate modes have been added, regular Modelica simulation can be used to assess their effects. In practice, the augmented models can be used for a wide variety of diagnostic applications. Suitably, all the augmented models are written in standard Modelica. Thus, the initial values, parameters and fault/alternate modes can all be set in a Modelica wrapper. For example, some uses of the new models for both design and diagnostic purposes include, but are not limited to:

- Given a set of initial conditions, parameter values and faults, determine whether a system requirement is met.
- Given a set of initial conditions, faults and component ages, determine the conditional probability that the system meets a requirement.
- Given a set of initial conditions, faults, observations, and ages for all but one of the components, determine the age of the remaining component.
- Standard model-based diagnosis.
- Fault mode and effects analysis.
- Fault mode and effects criticality analysis.
- Determine which component(s) damage most effects whether system meets requirements.
- Determining MTTF.

[0047] Suitably, every model class definition which contains faults is replaced with a new class definition, namely, a Modelica model class subsuming the original model class and adding declarative behavior to allow simulation of the faults.

[0048] If a class model is found to be susceptible to one or more faults, then the class is re-written. An encapsulated enumerated type is defined, listing the various fault modes of the class, along with the "nominal" mode. A discrete mode parameter of this new type is defined, defining the mode in which an instance of the class is operating. An if-equation is added, so that each operating mode can define its own dynamics. For example, this may look like the following:

```
if operating_mode == OperatingModes.Nominal then
     // original set of equations
     ...
elif operating_mode == OperatingModes.Fault1 then
     // equations for dynamics of Fault1 operation
     ...
else
     assert(False, "Invalid operating mode detected!");
end if;
```

[0049] The set of equations which apply in each fault mode is expressed in the appropriate branch of the foregoing if-equation. As the operating mode type is a Modelica parameter, the selected branch will not change during simulation and compilers can optimize the equation.

[0050] Suitably, each new class connects its instance of a power interface component through an added variable power dissipation component which in nominal mode dissipates no power. For example, if the original model class is an electrical component, and the power interface instance is an instance of the class Pin, the appropriate power dissipation component would be an instance of FAME.DynamicDampers.Electrical, with the damping parameter set to 0.

[0051] In addition, if the new class contains multiple power interfaces of the same type, it may also contain variable power conductance components connecting each pair of compatible connector components, nominally set to conduct no power. Connector components are "compatible" if they are of the same type, or inherit from the same type.

For example, if the original model class is an electrical component which contains an instance "p" of the connector type PositivePin and an instance "n" of the connector type NegativePin, both of which are subtypes of Pin, there would be an instance of FAME.Bridges.Electrical connecting those two instances, with the bridging amount set to the very small value of Modelica.Constants.eps.

[0052] The process may also suitably flatten the superclasses of the model into the rewritten class, and introduce two new externally visible components, FAME_operating_mode and FAME_fault_amount, as well as an enumerated type giving the possible faults for this component, FAME_OperatingModes.

[0053] Faults which manifest as power flow anomalies are modeled by a simple change to these two variables. For instance, as seen above, an electrical short can be modeled by setting FAME_operating_mode to FAME_OperatingModes.Electrical_Short, and FAME_fault_amount to 1.

[0054] For example, the above-referenced un-augmented or simple capacitor would in accordance with execution of the external approach become something like the following (i.e., an augmented capacitor model):

```
model Capacitor
     import FAME;
// locally defined classes in Capacitor
     final encapsulated type FAME_OperatingModes =
          enumeration(Nominal, Drift, Electrical_Short,
                    Electrical_Leak, Electrical_Break);
// components of Capacitor
     Modelica.SIunits.Voltage v;
     Modelica.Electrical.Analog.Interfaces.PositivePin p;
     FAME.DynamicDampers.ElectricalWithoutConnectEquations_
damper_p;
     parameter Modelica.SIunits.Capacitance C;
     Modelica.Electrical.Analog.Interfaces.NegativePin n;
     FAME.DynamicDampers.ElectricalWithoutConnectEquations_
damper_n;
```

```
    Modelica.SIunits.Current i "Current flowing from pin p
to pin n";
    FAME.DynamicBridges.Electrical _bridge_p_n;
    parameter FAME_OperatingModes
FAME_operating_mode=FAME_OperatingModes.Nominal;
    Modelica.Blocks.Interfaces.RealInput
FAME_fault_amount;
protected
    Modelica.SIunits.Capacitance FAME__C;
// algorithms and equations of Capacitor
equation
    i = FAME__C*der(v);
    v = _damper_p.port_b.v-_damper_n.port_b.v;
    0 = _damper_p.port_b.i+_damper_n.port_b.i;
    i = _damper_p.port_b.i;
    connect(p,_damper_p.port_a);
    FAME__C = C*(1.0-FAME_fault_amount);
    connect(n,_damper_n.port_a);
    connect(p,_bridge_p_n.port_a);
    connect(_bridge_p_n.port_b,n);
    if FAME_operating_mode==FAME_OperatingModes.Nominal
    then
        FAME_fault_amount = 0.0;
    end if;
    if FAME_operating_mode==FAME_OperatingModes.Nominal
    then
        _damper_p.damping = 0.0;
        _damper_n.damping = 0.0;
        _bridge_p_n.bridging = 0.0;
    elseif
    FAME_operating_mode==FAME_OperatingModes.Electrical_Short
    then
        _damper_p.damping = 1.0;
        _damper_n.damping = 1.0;
        _bridge_p_n.bridging = 1.0;
```

```
    elseif
    FAME_operating_mode==FAME_OperatingModes.Electrical_Leak
    then
        _damper_p.damping = FAME_fault_amount;
        _damper_n.damping = FAME_fault_amount;
        _bridge_p_n.bridging = FAME_fault_amount;
    elseif
    FAME_operating_mode==FAME_OperatingModes.Electrical_Break
    then
        _damper_p.damping = 1.0;
        _damper_n.damping = 1.0;
        _bridge_p_n.bridging = 0.0;
    elseif FAME_operating_mode==FAME_OperatingModes.Drift
    then
        _damper_p.damping = 0.0;
        _damper_n.damping = 0.0;
        _bridge_p_n.bridging = 0.0;
    end if;
end Capacitor;
```

[0055] In one optional embodiment, the system reads in a table of parametric faults, for example, each row of which describes a particular fault in which a supposedly fixed parameter changes during the operation of the component. For each fault, this provides the fault mode, the Modelica class, the specific parameter component of the class, the Modelica type of the parameter, and a Modelica expression describing the change in the parameter as a function of the variable FAME_fault_amount. Suitably, parametric faults are handled by introducing a new continuous variable, prefixed with "FAME_". An equation is added to set this variable to the value computed by the function specified by the fault table. Accordingly, references to the original parameter are replaced with an expression with references to this new variable.

[0056] The so called internal approach functions by detecting patterns in the dynamics of the model class, which indicate its susceptibility to a particular fault. Where such susceptibility is detected, the equations describing the dynamics of that model are then re-written in such a way as to allow that fault to be modeled.

[0057] In one suitable embodiment, for each fault and/or alternate mode, a susceptibility/applicability pattern is specified that, when matched by a model class, indicates that the class may exhibit the fault. In practice, these patterns are suitably defined in an appropriate language that describes the kinds of components the class may have, and how those components are interrelated. For example, to describe the common fault of clutch or brake slippage caused by surface wear, which results in a loss of torque through the clutch or brake, the system looks for model classes which have two variables representing torque, where one of the variables is the product of the other variable and another variable representing a component of friction. For example, this may look like the following:

```
component(tau1,type=Modelica.SIunits.Torque)
component(tau2,type=Modelica.SIunits.Torque)
component(friction,type=Modelica.SIunits.CoefficientOfFriction)
equation(eq1,tau1,product(tau2,friction))
```

where "tau1" represents a first torque variable, "tau2" represents a second torque variable, "friction" represents the component of friction variable and "eq1" represents an equation which relates the foregoing variable to one another.

[0058] Accordingly, the system suitably deems any model class satisfying these four constraints to be potentially subject to this slippage fault mode. That is to say, when the foregoing constrains are detected in a model class, that model class is deemed to be subject to the designated fault mode and accordingly rewritten.

[0059] Note that the names assigned to variables in the patterns are suitably independent of the names assigned to the variables in the Modelica source code. The pattern name bindings are preserved in the case of a match to be used in the modification of the model dynamics.

[0060] Suitably, before matching, the Modelica model class is *flattened*, i.e., all *superclasses* are expanded, and all record and connector subcomponents are expanded.

[0061] In general, the pattern language can be fairly simple. For example, there are two operators, *component* and *equation*, which match components and equations, respectively.

[0062] For example, the component *operator* has the form:

- `component` *`cname qualifier-list`*

where *cname* is an identifier to be assigned to the matching component (not the name defined by the component declaration), and *qualifier-list* is a comma-separated list of qualifiers which when matched result in the constraint being satisfied. For example, two qualifiers which may be specified are *type*, which is the fully-qualified name of a Modelica class, or *prefix*, which is the type prefix for the component declaration.

[0063] The *equation* operator, for example, may have the form:

- `equation` *`ename cname rhs-expression`*

where *ename* is an identifier to be assigned to the matching component, *cname* is bound in one of the *component* operators, and *rhs-expression* defines a partial constraint on the right-hand side of the matching equation.

[0064] In practice, instances of *rhs-expression* may be just a *cname*, or built up with the following primitives:

- `product` *`cname-or-exp`* `[` *`cname-or-exp`* `...` `]` – product of the named components or subexpressions (while other elements, including other components, may be involved in the product, they may be ignored);
- `sum` *`cname-or-exp`* `[` *`cname-or-exp`* `...` `]` – sum of the named components or subexpressions (while other elements, including other components, may be involved in the sum, they may be ignored);
- `quotient` *`numerator-cname-or-exp denominator-cname-or-exp`* – an expression formed of the numerator divided by the denominator; and/or
- `difference` *`minuend-cname-or-exp subtrahend-cname-or-exp`* – an expression formed of the minuend minus the subtrahend.
- `literal` *`modelica-literal`* – a literal Modelica expression.
- `function` *`cname-or-exp [ cname-or-exp ... ]`* – some arbitrary function of the specified *cname-or-exp* elements.

[0065] In addition to the patterns used to identity susceptible/applicable classes, each fault and/or alternate mode suitably contains a description of an "edit program", used to modify the class to add the ability to simulate that fault. For example, these edit programs may be written in the following primitives:

- `add-component` *`cname type`* – add a new component, internally referred to as *cname* (the actual Modelica name for it will be generated by the editor program), of the specified Modelica *type.* This variable's scope will be "protected".
- `replace-component` *`cname1 cname2 [ cname ]`* – replace instances of the variable referred to by the metaname *cname1* with the variable referred to by the metaname *cname2* in all equations. If the optional *cname* is specified, only modify the equation referred to by that metaname.

- `add-equation` *`ename cname exp`* – add a new equation, internally referred to as *ename*, which binds *cname* to the specified *exp.* The operators for *exp* use the same primitives that are defined for the pattern language above.
- `remove-equation` *`ename`* – remove the equation referred to by the metavariable *ename.*
- `type` *`cname`* – returns the Modelica type of the variable referred to with the metaname *cname.*

[0066] For example, the edit program for the clutch or brake wear fault described earlier may have the form:

add-component (mu-prime, type=type(mu))
add-component (slip-factor, type="Real (min=0.0, max=1.0)")

replace-component (mu, mu-prime)

add-equation (eq2, mu-prime, product (mu, difference (1.0, slip-factor)))

[0067] As another example, a complete fault pattern for the fault Electrical_Open (an open circuit) might look this:

component (v , type="Modelica.SIunits.Voltage")
component (i, type="Modelica.SIunits.Current")
equation(eq1, i, function(v))

remove-equation (eq1)
add-equation (eq2, i, literal ("Modelica.Constants.eps"))

[0068] In accordance with aspects of one exemplary embodiment, if a class model is found to be susceptible to one or more faults, the class is re-written. An encapsulated enumerated type is defined, listing the various fault modes of the class, along with the "nominal" mode. A discrete mode parameter of this new type is defined, defining the mode in which an instance of the class is operating. The set of equations for the class are replaced with a single if-equation, e.g., of the form:

```
if operating_mode == OperatingModes.Nominal then
< original set of equations >
elif operating_mode == OperatingModes.Fault1 then
< equations for dynamics of Fault1 operation >
...
else
assert(False,"Invalid operating_mode detected!");
end if;
```

[0069] For example, the above-referenced capacitor, re-written to be able to simulate *short* and *open* faults, becomes something like the following:

```
model Capacitor
     extends Modelica.Electrical.Analog.Interfaces.OnePort;

     encapsulated final type OperatingModes =
     enumeration (Nominal, ElectricalOpen, ElectricalShort);
     parameter OperatingModes
     operating_mode=OperatingModes.Nominal;
```

```
    parameter Modelica.SIunits.Capacitance C(start=1);

    equation
        if operating_mode == OperatingModes.Nominal then
            i = C*der(v);
        elseif operating_mode == OperatingModes.ElectricalOpen
        then
            i = Modelica.Constants.eps;
        elseif operating_mode ==
        OperatingModes.ElectricalShort then
            v = Modelica.Constants.eps;
        end if;
end Capacitor;
```

[0070] Note the use of Modelica.Constants.eps, which is Modelica's notion of a very small value, just as Modelica.Constants.inf is Modelica's idea of a very large value. This used this to avoid numerical instability associated with the use of zero, in the simulation phase.

[0071] With reference now to FIGURE **1**, there is shown in general a computer 10 or other like processor which executes or otherwise performs a model transformation process, e.g., either automatically or semi-automatically. Input thereto is a first library 20 of nominal component models 22. For example, the library 20 may include a collection of models 22, in which each model 22 describes and/or otherwise represents a physical component, e.g., including the operation and/or behavior of the component. Suitably, the library 20 may include one or more such models 22, e.g., representing components that are to be employed and/or interact together in a system, or the library 20 may include some other suitable collection of such models 22. In practice, each model 22 in the input library 20 may comprise and/or be represented by lines of code (e.g., written in the Modelica language or another such suitable language) or the like stored in a file or otherwise in a suitable memory or other data storage device accessible by the computer or processor 10. Suitably, at least one of the models 22 (but alternately more or all of the models 22) in the library 20 describes or otherwise simulates only a nominal operation and/or behavior of the physical component modeled thereby, e.g., such nominal operation

and/or behavior may correspond to the correct operation and/or behavior of the modeled component. In one simple example, the library 20 may include only one model 22, and in more complex examples, it may include a great many models 22.

[0072] As another input, a second library 30 may include a collection of salient alternate modes and/or mechanisms 32 applicable to the modeled components of the library 20. For example, each alternate mode and/or mechanism 32 of the library 30 may describe and/or represent a fault mode and/or mechanism or some other non-nominal or alternate mode of operation and/or behavior. In practice, each alternate mode 32 may be applicable to one or more modeled components in the library 20, while potentially not being applicable to others. For example, a short circuit fault mode may be applicable to a number of different electronic components (e.g., such as a capacitor and/or inductor), while not being applicable to mechanical components (e.g., such as a brake). Suitably, each alternate mode 32 in the input library 30 may comprise and/or be represented by lines of code (e.g., written in the Modelica language or another such suitable language) or the like stored in a file or otherwise in a suitable memory or other data storage device accessible by the computer or processor 10. For example, a fault mechanism can be used to express an underlying cause of a fault, while a fault mode can be used to express the abnormal operation or behavior of a component being modeled. Consider, for purposes of illustration, the exemplary fault mechanisms of a rusty shaft and a disconnected lead wire in relation to a motor which is being modeled. The corresponding fault modes resulting from the aforementioned mechanisms may be a shaft that is harder to turn (i.e., as compared to when the motor is otherwise operating normally) and a motor that simply does not run, respectively.

[0073] In one suitable embodiment, for example, fault modes and mechanisms may be captured and/or otherwise identified for each subject component being modeled. These may be organized, optionally along with fault probabilities, in a suitable taxonomy so that corresponding behavioral models can be built consistently. Accordingly, a fault mechanism behavior schema may in turn be developed in view of the aforementioned taxonomy. For example, the schema may define faulty behavior at the highest abstraction level in a generic sense and use the concept of inheritance to specify behavior for the lower-level fault mechanisms. This approach allows for flexibility in defining behavior for

subclasses of fault mechanisms. Consider the case of a fault mechanism identified as "wear." This may represent a high-level category of different subclasses of wear, e.g., such as abrasive, impact or corrosive wear. Suitably then, the subclasses (of wear in this example) inherit and expands upon the higher-level behavior. For example, the behavior of wear representing a loss of material is expanded to include, e.g., the frictional behavior in the subclass of abrasive wear. The result of this process is that fault modes for different components can call upon the same fault mechanism, i.e., underlying faulty behavioral model. Moreover, these subclasses may cover the expression of a fault mechanism in different domains, e.g., such as mechanical, hydraulic, electronic, etc.

[0074] Of course, more broadly, an alternate mode may not represent faulty operation and/or behavior at all but rather may simply represent some other non-nominal operation and/or behavior of the component being modeled.

[0075] In any event, the processor 10 generates and/or otherwise outputs a library 40 of augmented component models 42. In practice, each model 42 in the output library 40 may comprise and/or be represented by lines of code (e.g., written in the Modelica language or another such suitable language) or the like output to a file or otherwise which is stored in a suitable memory or other data storage device accessible by the computer or processor 10. Suitably, in accordance with the model transformation process executed by the computer and/or processor 10, an augmented component model 42 is output for each input nominal component model 22 from the library 20, with the augmented component models 42 including a description and/or representation of not only the nominal operation and/or behavior of the modeled component but also including a description and/or representation of one or more or all of the alternate modes of operation and/or behavior applicable to the component being modeled. Suitably, the output augmented component models 42 of the library 40 are fully compatible with their corresponding input nominal component models 22, e.g., in terms of their interfaces, inputs, outputs and parameters. Accordingly, for example, the augmented component models 42 may simply be plugged-in to and/or replace the nominal component models 22 for testing and/or analysis of fault and/or other alternative modes of operation and/or behavior of various components.

[0076] In one exemplary embodiment, the computer and/or processor 10 may execute a computer program or the like (e.g., stored in a suitable memory or other data storage device accessible by the computer/processor 10) to carry out the aforementioned model transformation process. FIGURE **2** is a flow chart illustrating steps of one suitable embodiment of the model transformation process 100. Optionally, the aforementioned program executed by the computer/processor 10 may be written in a general-purpose, class-based, object-oriented computer programming language, e.g., such as a Java or the like, which incorporates the JModelica platform or the like and the Java/JastAdd parser or the like. Suitably, the program runs over the input library 20 and recognizes those nominal component models 22 which are susceptible the various faults and/or alternate modes 32 of the library 30, e.g., by identifying specified patterns associated with the respective alternate modes 32 in the variables, equations and/or other content of the models 22. That is to say, if the variables, equations and/or other content of a given model 22 match (e.g., within a given tolerance) a specific pattern which is associated with a given alternate mode 32, then that alternate mode 32 is deemed applicable to the given model 22. Consequently, the model 22 is in essence rewritten to include a description of the applicable alternate mode 32, and the new augmented model 42 is output.

[0077] Referring now more specifically to FIGURE **2**, the model transformation process 100 suitably begins at step 102, with the reading in of one of the models 22 from the library 20.

[0078] At step 104, the model 22 is parsed, and at step 106, the parsed model is examined and/or analyzed to look for specified content, patterns and/or clues therein which are associated with an alternative mode 32. In practice, a given model 22 will generally comprises one or more lines of code, including, e.g., equations, parameters, variables and/or other content, which defines the component being modeled along with its nominal operation and/or behavior. From this content, clues and/or patterns can be detected and/or discerned which match (e.g., within some degree of tolerance) certain content, clues and/or patterns associated with faults and/or other non-nominal or alternate modes. To illustrate, the content may include elements, clues or patterns which suggest that a given model 22 describes an electrical component, e.g., such as a capacitor. Indeed, the model 22 may include an equation or parameter or other content from which

it can be determined that the model 22 is in fact a capacitor. To continue with the example, it may be known that this type of component, i.e., an electrical component in general and more specifically a capacitor, is susceptible to certain faults and/or certain alternate modes are applicable thereto, e.g., a short or an open circuit, etc. That is to say, the content of the model 22 sufficiently matches a pattern associated with the respective fault and/or alternate mode 32.

[0079] At decision step 108, if no pattern and/or clues are found which sufficiently match those sought, then the process 100 branches to step 110, otherwise if a pattern is found, then the process 100 continues to step 112.

[0080] At step 110, the model 22 is essentially left unaltered. That is to say, the model 22 may be essentially left as is. In this case, the corresponding model 42 in the output library 40 will be essentially the same as the input model 22.

[0081] Conversely, at step 112, the model 22 is rewritten and/or edited into a new model 42 so as to describe, not only the nominal mode of operation originally contained in the model 22, but also to describe the alternate mode associated with matched pattern.

[0082] In practice, it is to be appreciated that suitably each model 22 of the input library 20 is so processed, e.g., in turn or in parallel. Additionally, it is to be appreciated that in practice each model 22 may be susceptible to zero or one or more of the faults and/or alternate modes 32 described in the library 30. That is to say, the content of the model 22 may match none or one or more different patterns associated with the various faults and/or other alternate modes 32.

[0083] Suitably, upon rewriting (or editing) of a model 22 to include alternate modes (i.e., to achieve the corresponding augmented model 42), the description of and/or code for each applicable alternate mode may be injected or otherwise inserted into the model. For example, that description or code may be obtained from the applicable alternate modes 32 and may include one or more equations and/or functions which represent and/or model the alternate mode in question. In practice, every model class definition which contains faults or alternate modes is replaced with a new class definition, e.g., a Modelica model class subsuming the original model class and adding behavior to allow simulation of the faults and/or alternate modes. Additionally, the augmented model 42 is provided with a control mechanism and/or suitable code to permit the selection of the

particular mode in which the model 42 will operate for a given simulation. That is to say, the aforementioned control mechanism and/or code allows the augmented model 42 to be selectively run in any one of the modes, be it the nominal mode from the original model 22 or one of the alternate modes added by the rewrite/editing. In this way, alternative dynamics are enabled for each operating mode.

[0084] For example, an input model 22 of a capacitor may look something like the aforementioned un-augmented or simple capacitor model (written in the Modelica language). Accordingly, the output augmented model 42 (also written in the Modelica language) generated from such an input model 22 may look like the aforementioned augmented capacitor model.

[0085] More specifically, in the forgoing example, it can be seen that four faults and/or alternate modes were found applicable to the original model 22. In particular, in the augmented model 42, an encapsulated "enumerate" type is defined in which the alternate modes labeled as Drift, Electrical_Short, Electrical_Leak and Electrical_Break are listed along with the Nominal mode. A discrete mode parameter of this new type is defined, defining the mode in which an instance of the class is operating. Accordingly, a conditional equation is then employed and/or written to permit dynamic selection of a particular operating mode. Suitably, a set of equations which apply in each alternate mode is expressed in the appropriate branch of this conditional equation. As the operating mode type is, e.g., a Modelica parameter, the selected branch generally will not change during simulation and compilers can optimize this equation.

[0086] In one embodiment (as referenced above), the power flow analysis may depend on identification of standard power interfaces represented in the input model 22. Suitably, these are instances of Modelica connector classes and/or the like, e.g., such as the Pin class in Modelica's package of analog electrical interface types. In practice, such power interfaces generally contain two variables, one of an "effort" type, such as "Voltage" or "Pressure," and another of a "flow" type, such as "Current" or "MassFlowRate." In one suitable embodiment, the analysis examines both the directly defined components of the model class as well as components of each inherited class used by the model class. Suitably, every model class definition which contains such an instance is wrapped in a "shell" class definition, which is, e.g., a new Modelica model class containing the original

model class, and containing an instance of that model class, as well as instances of each connector component, parameter component and constant component found in that original class.

[0087] In addition, the system (e.g., the computer or processor 10) can optionally read in a table of parametric faults and/or alternate modes, each row of which describes a particular fault or alternate mode in which a supposedly fixed parameter changes during the operation of the component. For each fault or alternate mode therein, the table or row thereof provides the fault/alternate mode, the Modelica class, the specific parameter component of the class, the Modelica type of the parameter, and a Modelica expression describing the change in the parameter as a function of a variable, e.g., such as FAME_fault_amount. Suitably, e.g., the aforementioned table or the like may reside is a file or the like stored or otherwise saved in a memory or other suitable data storage device that is accessible by the computer or processor 10.

[0088] In one suitable embodiment, each new shell class connects its instance of a power interface and/or connector component to the original power interface and/or connector component in its instance of the original model class, through an added variable power dissipation component, e.g., which in the nominal mode is set to dissipate no power. For instance, if the original model class is an electrical component, and the power interface instance is an instance of the class Pin, the appropriate power dissipation component may be an instance of FAME.DynamicDampers.Electrical, with the damping parameter set to 0.

[0089] Additionally, if the shell class contains multiple power interfaces and/or connector components of the same type, it may also contain variable power conductance components connecting each pair of compatible connector components, nominally set to conduct no power. In general, connector components are deemed "compatible" if they are of the same type, or inherit from the same type. For instance, if the original model class is an electrical component which contains an instance "p" of the connector type PositivePin and an instance "n" of the connector type NegativePin, both of which are subtypes of Pin, there would be generated an instance of FAME.Bridges.Electrical connecting those two instances, with the conductance or bridging amount set to a very small value of Modelica.Constants.eps.

[0090] As shown above, the augmented model 42 contains an example of a capacitor model with the aforementioned dampers and bridges added. Suitably, the model transformation process also flattens (removes all hierarchy) the superclasses of the model into the rewritten class, and introduces two new externally visible components, FAME_operating_mode and FAME_fault_amount, as well as an enumerated type giving the possible faults and/or alternate operating modes for this component, FAME_OperatingModes. Suitably, faults and/or other alternate modes which manifest as power flow anomalies can be modeled by a change to these two variables. For example, as shown above in the output augmented model 42, an electrical short may be modeled by setting FAME_operating_mode equal to FAME_OperatingModes.Electrical_Short, and FAME_fault_amount equal to 1.

[0091] As shown above, parametric faults and/or other like alternate modes are handled by introducing a new continuous variable, e.g., prefixed with "FAME_". Suitably, an equation is added to set this variable to the value computed by the function specified by the above-mentioned table of parametric faults and/or alternate modes. Accordingly, references to the original parameter are replaced with an expression which references to this new variable. For example, as shown in the above example, the parameter C (in the input model 22) is replaced with the expression C*(1-FAME_fault_amount) (in the output augmented model 42).

[0092] In general, in accordance with at least one further embodiment, there is disclosed herein a method, system and/or apparatus which is operative to generate damage-parameter maps and use the same as a proxy for damage process models.

[0093] In general, as disclosed herein, a damage process models express damage due to failure mechanisms that are active in system components because of their service environments as a function of, for example, such parameters as loads, materials' properties, and dimensions. The damage state of a component reflects the service usage and history derived from context models. Uncertainties about parameters derived from context and component models and about damage process model parameters are characterized and incorporated in stochastic simulations. In accordance with one suitable embodiment, the parameterized simulation results are represented as damage-parameter maps (e.g., tables) that are then used to look up the expected damage in a

user-specified component for a user-specified usage context without running the underlying stochastic damage process simulations again.

[0094] One example of a system component for which a failure risk or reliability analysis may be performed is an engine/drivetrain mounting bracket of a vehicle. Such brackets are subject to fatigue failure due to time-varying loads resulting from traveling over rough terrain, vibration, or impacts. In practice, other failure mechanisms may also be analyzed and/or different system components may likewise be analyzed. Suitably, the analyses involve the use different engineering analysis tools and/or software. For example, an evaluation of the risk of fatigue failure of a mounting bracket may include:

- Generating bracket force histories using Multi-Body Dynamics (MBD) software;
- Obtaining stresses in the bracket using Finite Element Analysis (FEA) software; and
- Evaluating failure probability and bracket life using Materials Probabilistic Fatigue Analysis (PFA) software.

[0095] In accordance with one suitable embodiment, a backend workflow 200 (which is optionally invisible to the user) in accordance with aspects of the present inventive subject matter is shown in FIGURE 3. To demonstrate, the workflow 200 is described herein in conjunction with a seed design vehicle (SDV) based on a given set of design parameters. However, in practice, the workflow 200 may be otherwise employed to achive similar results for other applications. In one suitable embodiment, the backend workflow 200 may progress as follows:

[0096] As shown, a MBD simulation model 202 is first developed. In the present example, the SDV that was developed included springs that represented the brackets that attached an engine and transmission assemblies to a hull. More generally, design configurations 204 and mission definitions 206 may be given as inputs to the workflow 200.

[0097] In particular, mass properties are gathered, for example, from a computer aided design (CAD) model. As shown, the CAD information is obtained from the design configurations 204 by the MBD model 202. Alternately, they may be estimated, e.g., assuming a uniform density box. Suitably, the MBD model is created based on the CAD model, for example, of the SDV and its components. In the case of the present example,

suspension spring/damping characteristics and mounting bracket location and stiffness are determined, and along with the selection of terrain/obstacle profiles, reasonable vehicle speed(s) are also determined. In general, the design parameters 208 (such as the foregoing) are obtained from the design configurations 204. As shown, the design parameters 206 are used in the generation one or more parameterized maps 300, e.g., such as a parameterized fatigue map.

[0098] As shown, the MBD simulation model 202 and mission definitions 206 are employed to achieve a set of results 210, namely, according to the present example, acceleration/force time histories at points-of-interest. Suitably, the results for are post-processed, e.g., including power pack mounting brackets for the current effort. In accordance with the present example, a mission force history is defined by concatenating the force histories that are generated from the MBD simulation. For example, in the present case, the SDV is driven over: Perryman 3, 12″ high half round bump, a 30″ step down, and a 96″ wide trench crossing. Using Dynamic Analysis and Design System (DADS) software, force histories (e.g., of the bracket in the present instance) are derived, for example, from DADS simulations of the SDV. The processed results 210 are in turn used to generate the parameterized map 300. In the present example, the parameterized map 300 is used to determine the component life 302.

[0099] In the present example, a class of angle brackets for 2 materials and 3 angle dimensions and varying angle and gusset thicknesses were defined. For each bracket design, linear static Finite Element (FE) analyses were performed to derive the maximum stress in the brackets due to the highest force in the bracket mission history. An allowable stress factor for each bracket was calculated by dividing the yield stress of the material by the maximum stress in that bracket. An outcome of the FE analyses were the bracket stiffness in the three directions. Stiffnesses were represented via springs in the DADS vehicle simulation analyses.

[00100] In turn, a fatigue failure simulation model was modified to generate damage parameter maps for given values of Allowable Stress Factor (ASF), for a given material of the bracket due to the bracket force mission histories derived from the DADS simulations. The ASF equations for bracket thicknesses t_a and t_g were derived by curve

fitting the bracket FE analyses results. Bracket stiffness equations were also derived in a similar fashion.

[00101] The bracket force histories that were used for the damage simulations were derived from MBD simulations of the SDV using DADS software. The mission force history was defined by concatenating the bracket force histories that were generated from Perryman 3, 12" High half round bump, 30" step down, and 96" wide trench crossing DADS simulations of the SDV. FIGURE 4 shows a graphical representation of the course, with the course height along the vertical axis (in inches) and the course distance along the horizontal axis (in inches).

[00102] From the histories of the four brackets that support the engine and transmission assembly, the Front Left (FL) bracket histories were chosen to define the mission since they had the highest force peaks. One example mission was defined to be a distance of 28,800 miles based on 1200 hour operation, in accordance with Fast Adaptable Next-Generation (FANG) vehicle program. In the history, the 28 second Perryman 3 was repeated for the full duration (212826 times) and the wall step up, step down, and trench crossing events were repeated every 10 miles (2880 times). FIGURES 5A and 5B illustrates sample, acceleration time histories for the trench crossing and 1 meter wall step down.

[00103] In one suitable embodiment, a fatigue model generates damage parameter maps at different probabilities of occurrence for a component, e.g., given its cyclic load history, geometry, material properties and certain uncertainties. The fatigue model is based on strain vs. life (ε vs. N) behavior of the material due to cyclic loads. The inputs to the fatigue model are the force history, the material data and the governing parameters for the component, e.g., for the bracket in the present case. Monte-Carlo (M-C) loops are performed for a number of random trials to derive the damage at desired probabilities of occurrence for a set of parameter values. In a single M-C trial, damage is derived by the following steps. The input load history is first transformed to a stress time history using a load-to-stress transformation described by the geometry or ASF. Next, the stress range and number of cycles are identified, e.g., using a rainflow counting technique or algorithm. If the stress range of a cycle is above the yield strength, the total strain, ε, will include the elastic and plastic strain. Suitably, Neuber's rule is used to transform the elastic stress to

a total strain range using the stress-strain curve for the material. The Walker relation is then used to correct the total strain range to account for the mean strain of the cycle. The damage due to total strain is derived using the strain vs. life model. Finally, the total damage due to each cycle is accumulated using Miner's rule to find the damage due to the entire force history.

[00104] FIGURE 6 shows a sample von Mises stress contour of a 4 x 4 x 4 bracket. The maximum stress of 39.4 ksi is due to the largest force of 18300 lbs and the yield stress for stainless steel 17-4PH is 170 ksi. Hence, the ASF = 170/39.4 = 4.31.

[00105] In one example, stainless steel 17-4PH H900 and aluminum 6061-T6 materials where used in the bracket damage simulations. The properties of the materials used in the bracket damage simulations were obtained from MIL-HNDBK-5J (Department of Defense Handbook: Metallic Materials and Elements for Aerospace Vehicle Structures (31 Jan 2003)) and other suitable material's data files. The stress rupture properties given in Table 1 are the yield strengths for alloys. The A and B "basis" values for these materials (from published literature) were used to derive the mean, standard deviation, and the coefficient of variations for the yield stress and describe the strength uncertainty. The total strain vs. life data are used in the bootstrapping materials model approach to capture fatigue life uncertainty implied by the data set.

Probability of Failure	LIFE (Number of Missions) Allowable Stress Factor (ASF)										
	1.00	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	-	-
0.0001	1.54E-03	6.18E-02	3.37E-01	9.34E-01	1.75E+00	4.02E+00	7.96E+00	1.19E+01	1.96E+01	0.00E+00	0.00E+00
0.0002	1.74E-03	7.08E-02	3.95E-01	1.24E+00	2.36E+00	5.41E+00	9.84E+00	1.63E+01	2.72E+01	0.00E+00	0.00E+00
0.0003	1.95E-03	7.36E-02	4.59E-01	1.36E+00	2.75E+00	6.56E+00	1.22E+01	1.98E+01	3.25E+01	0.00E+00	0.00E+00
0.0004	2.06E-03	7.89E-02	4.81E-01	1.45E+00	3.53E+00	7.97E+00	1.34E+01	2.29E+01	3.99E+01	0.00E+00	0.00E+00
0.0005	2.14E-03	8.32E-02	5.15E-01	1.54E+00	4.04E+00	8.79E+00	1.47E+01	2.69E+01	4.41E+01	0.00E+00	0.00E+00
0.0006	2.18E-03	8.81E-02	5.40E-01	1.63E+00	4.38E+00	9.41E+00	1.61E+01	2.93E+01	4.84E+01	0.00E+00	0.00E+00
0.0007	2.28E-03	9.20E-02	5.54E-01	1.74E+00	4.68E+00	1.00E+01	1.74E+01	3.23E+01	5.44E+01	0.00E+00	0.00E+00
0.0008	2.33E-03	9.53E-02	6.02E-01	1.84E+00	4.96E+00	1.04E+01	1.89E+01	3.40E+01	5.83E+01	0.00E+00	0.00E+00
0.0009	2.36E-03	9.72E-02	6.20E-01	1.92E+00	5.19E+00	1.07E+01	2.04E+01	3.57E+01	6.18E+01	0.00E+00	0.00E+00
0.001	2.42E-03	9.94E-02	6.43E-01	2.06E+00	5.33E+00	1.17E+01	2.18E+01	3.69E+01	6.42E+01	0.00E+00	0.00E+00
0.002	2.88E-03	1.17E-01	8.08E-01	2.80E+00	7.54E+00	1.67E+01	3.11E+01	5.46E+01	9.21E+01	0.00E+00	0.00E+00
0.003	3.17E-03	1.29E-01	9.18E-01	3.28E+00	9.36E+00	1.97E+01	3.79E+01	7.10E+01	1.18E+02	0.00E+00	0.00E+00
0.004	3.38E-03	1.38E-01	1.02E+00	3.68E+00	1.05E+01	2.30E+01	4.49E+01	8.48E+01	1.40E+02	0.00E+00	0.00E+00
0.005	3.54E-03	1.45E-01	1.09E+00	4.05E+00	1.16E+01	2.59E+01	5.23E+01	9.56E+01	1.62E+02	0.00E+00	0.00E+00
0.006	3.69E-03	1.52E-01	1.16E+00	4.36E+00	1.28E+01	2.85E+01	5.79E+01	1.09E+02	1.81E+02	0.00E+00	0.00E+00
0.007	3.84E-03	1.58E-01	1.24E+00	4.69E+00	1.38E+01	3.13E+01	6.28E+01	1.18E+02	2.01E+02	0.00E+00	0.00E+00
0.008	3.98E-03	1.64E-01	1.31E+00	4.99E+00	1.47E+01	3.42E+01	6.78E+01	1.28E+02	2.21E+02	0.00E+00	0.00E+00
0.009	4.12E-03	1.70E-01	1.37E+00	5.32E+00	1.57E+01	3.62E+01	7.29E+01	1.40E+02	2.40E+02	0.00E+00	0.00E+00
0.01	4.25E-03	1.75E-01	1.42E+00	5.59E+00	1.67E+01	3.86E+01	7.84E+01	1.48E+02	2.56E+02	0.00E+00	0.00E+00
0.02	5.14E-03	2.19E-01	1.92E+00	7.99E+00	2.45E+01	6.05E+01	1.27E+02	2.53E+02	4.37E+02	0.00E+00	0.00E+00
0.03	5.83E-03	2.53E-01	2.32E+00	1.01E+01	3.17E+01	7.93E+01	1.73E+02	3.47E+02	6.12E+02	0.00E+00	0.00E+00
0.04	6.42E-03	2.81E-01	2.67E+00	1.20E+01	3.82E+01	9.83E+01	2.19E+02	4.37E+02	7.82E+02	0.00E+00	0.00E+00
0.05	6.93E-03	3.07E-01	2.99E+00	1.39E+01	4.50E+01	1.17E+02	2.64E+02	5.30E+02	9.70E+02	0.00E+00	0.00E+00
0.06	7.45E-03	3.30E-01	3.33E+00	1.57E+01	5.16E+01	1.34E+02	3.09E+02	6.27E+02	1.15E+03	0.00E+00	0.00E+00
0.07	7.90E-03	3.55E-01	3.64E+00	1.75E+01	5.79E+01	1.53E+02	3.56E+02	7.23E+02	1.34E+03	0.00E+00	0.00E+00
0.08	8.38E-03	3.77E-01	3.93E+00	1.91E+01	6.40E+01	1.71E+02	4.00E+02	8.26E+02	1.53E+03	0.00E+00	0.00E+00
0.09	8.85E-03	4.00E-01	4.22E+00	2.08E+01	7.09E+01	1.90E+02	4.45E+02	9.32E+02	1.73E+03	0.00E+00	0.00E+00
0.1	9.29E-03	4.23E-01	4.52E+00	2.25E+01	7.73E+01	2.09E+02	4.95E+02	1.04E+03	1.95E+03	0.00E+00	0.00E+00
0.2	1.36E-02	6.43E-01	7.55E+00	4.07E+01	1.50E+02	4.35E+02	1.07E+03	2.33E+03	4.52E+03	0.00E+00	0.00E+00
0.3	1.85E-02	8.96E-01	1.12E+01	6.30E+01	2.43E+02	7.39E+02	1.85E+03	4.17E+03	8.39E+03	0.00E+00	0.00E+00
0.4	2.47E-02	1.20E+00	1.56E+01	9.16E+01	3.67E+02	1.15E+03	2.96E+03	6.90E+03	1.42E+04	0.00E+00	0.00E+00
0.5	3.28E-02	1.60E+00	2.14E+01	1.30E+02	5.40E+02	1.72E+03	4.60E+03	1.11E+04	2.30E+04	0.00E+00	0.00E+00
0.6	4.34E-02	2.12E+00	2.92E+01	1.84E+02	7.92E+02	2.60E+03	7.19E+03	1.75E+04	3.72E+04	0.00E+00	0.00E+00
0.7	5.79E-02	2.86E+00	4.07E+01	2.67E+02	1.19E+03	4.04E+03	1.14E+04	2.84E+04	6.20E+04	0.00E+00	0.00E+00
0.8	7.94E-02	3.97E+00	5.96E+01	4.07E+02	1.90E+03	6.71E+03	1.95E+04	4.99E+04	1.12E+05	0.00E+00	0.00E+00
0.9	1.16E-01	6.04E+00	9.91E+01	7.27E+02	3.58E+03	1.33E+04	4.11E+04	1.07E+05	2.50E+05	0.00E+00	0.00E+00
1	7.78E-01	5.29E+01	1.47E+03	1.91E+04	2.34E+05	5.15E+05	2.80E+06	1.79E+07	3.43E+07	0.00E+00	0.00E+00

Table 1. Sample Damage-Parameter Map for 17-4PH900 Steel Bracket

[00106] The damage-parameter map allows a system designer or design tester, to explore the reliability consequences of different design choices, and to assess the probability of catastrophic fatigue failure as a result of mission stress. For example, in accordance with the bracket example described herein, the approach and/or system described herein aids in discovering the answers to the following questions:

- What is the probability of fatigue failure in the bracket if it is used to support the engine in the SDV, where the vehicle is run, for example, for 1200 hours over the terrain profile described above; and
- If changes are made to the dimensions or the material of the bracket, what difference does that make?

[00107] Above, it has been described how stochastic physics-of-failure models are pre-simulated in a Monte Carlo framework incorporating design uncertainties (material choices and geometry) as well as the expected spectrum of usage over the lifetime of a component. Suitably, the results stored as damage-parameter maps which can then be indexed by model material and geometric parameters and level of usage. In accordance with one suitable embodiment, the described approach and/or system is expanded to account for variances in the material properties as well as manufacturing processes (i.e., tolerance analysis). The expanded process is similar to the one described above except that for simulation set up the system would account for samples not only from design choices (materials and geometry) but also from distributions of material properties and geometric dimensions arising due to variations in manufacturing processes.

[00108] In practice, this multidimensional sampling space means that the number of simulations grows exponentially with the number of distributions to sample from. However, this can be countered by speed up methods developed for Monte Carlo simulations.

[00109] In one suitable embodiment, the system is further adapted to interpolate over a cloud of MBD point simulations over a desired set of points-of-interest to determine the acceleration at those points-of-interest for an unknown set of design or operational parameters. More specifically, MBD simulations are conducted to numerically determine the loads (accelerations) on points of interest in a system (e.g., suspension components in a car) based on the operation environment (e.g., terrain/road) and the operational parameters (e.g., speed). These loads are subsequently used in damage process models to determine the probability of failure in components at any given level of usage (e.g., suspension failure after 100K miles over potholed roads using city drive profiles). In practice, these simulations can be computation and labor intensive to set up and run. Hence, conventionally only a small number of points of interest in a system and a small number of operational profiles are simulated. However, this limited simulation testing can lead to unexpected failures in deployed systems. Therefore, in accordance with one exemplary embodiment of the system and/or method employed herein there is implemented a process of interpolating over a cloud of MBD point simulations over a desired set of points-of-interest to determine the loads and/or accelerations at those

points-of-interest for an unknown set of design or operational parameters, e.g., of a vehicle. Suitably, the system and/or method exploits the linear structural dynamics of a mass-spring-damper-based MBD model to interpolate over the pre-computed accelerations at system points-of-interest for parameter sets uniformly sampled from the parameter space.

[00110] MBD analyses help in fast exploration of reliability in large design spaces. In particular, MBD analysis is useful for determining component fatigue life conditioned on mission usage and design configuration.

[00111] Suitably, in accordance with aspects of one exemplary embodiment, the ability is provided to determine component loads at arbitrary locations within a design configuration (e.g., front-engine vs mid- or rear-engine configuration for a car). By interpolating over a cloud of MBD point simulations, a user is able to explore the entire design space in a more systematic and comprehensive manner resulting in more efficient, reliable and robust designs.

[00112] The following evidence illustrates the value of using interpolation to determine 3-axis accelerations for an arbitrary point design from a multi-dimensional cloud of 3-axis accelerations derived from MBD simulations over a set of point designs that span the range of the test design point. The cross-validation study presented here assumes that all point designs considered belong to a tracked personnel carrier vehicle class. The point-of-interest where the accelerations are computed is the bracket element connecting the Power Pack to the chassis. It is assumed that there are 4 identical brackets supporting the Power Pack.

[00113] In the present example, the 125-point cloud considers MBD simulations quantized by the following 3 simulation parameters:

- Vehicle speed over Perryman 3 terrain: the values simulated are 5, 8, 11, 14, and 17 mph;
    - Bracket vertical stiffness: the values simulated are 3.49x105, 8.55 x105, 1.71 x106, 2.14 x106, and 2.83 x106 lb/in; and
- Power Pack mass scaling factor: the values simulated are 0.5, 0.75, 1, 1.25, and 1.5 (x Nominal).

[00114] FIGURES 7, 8 and 9 show the maximum accelerations at the bracket in X, Y, and Z directions respectively as functions of speed and bracket stiffness.

[00115] A 3D spline interpolation was used to estimate the accelerations at known simulation points (i.e., for a given set of simulation parameters) and the error with respect to the simulated value was computed as a percentage of the simulated value. FIGURES 10, 11 and 12 show the percentage errors for the maximum lateral, longitudinal and vertical accelerations as a function of vehicle speed and Power Pack mass scaling factor. Given the relatively low errors (i.e., most errors less than 5%), the interpolation approach to determining 3-axis accelerations for an arbitrary point design from a pre-simulated multi-dimensional cloud of 3-axis accelerations is considered valid.

[00116] The above methods and/or apparatus have been described with respect to particular embodiments. It is to be appreciated, however, that certain modifications and/or alteration are also contemplated.

[00117] In any event, it is to be appreciated that in connection with the particular exemplary embodiment(s) presented herein certain structural and/or function features are described as being incorporated in defined elements and/or components. However, it is contemplated that these features may, to the same or similar benefit, also likewise be incorporated in other elements and/or components where appropriate. It is also to be appreciated that different aspects of the exemplary embodiments may be selectively employed as appropriate to achieve other alternate embodiments suited for desired applications, the other alternate embodiments thereby realizing the respective advantages of the aspects incorporated therein.

[00118] It is also to be appreciated that any one or more of the particular tasks, steps, processes, methods, functions, elements and/or components described herein may suitably be implemented via hardware, software, firmware or a combination thereof. In particular, the processor 10 may be embodied by a computer or other electronic data processing device that is configured and/or otherwise provisioned to perform one or more of the tasks, steps, processes, methods and/or functions described herein. For example, a computer or other electronic data processing device embodying the processor 10 may be provided, supplied and/or programmed with a suitable listing of code (e.g., such as source code, interpretive code, object code, directly executable code, and so forth) or

other like instructions or software or firmware, such that when run and/or executed by the computer or other electronic data processing device one or more of the tasks, steps, processes, methods and/or functions described herein are completed or otherwise performed. Suitably, the listing of code or other like instructions or software or firmware is implemented as and/or recorded, stored, contained or included in and/or on a non-transitory computer and/or machine readable storage medium or media so as to be providable to and/or executable by the computer or other electronic data processing device. For example, suitable storage mediums and/or media can include but are not limited to: floppy disks, flexible disks, hard disks, magnetic tape, or any other magnetic storage medium or media, CD-ROM, DVD, optical disks, or any other optical medium or media, a RAM, a ROM, a PROM, an EPROM, a FLASH-EPROM, or other memory or chip or cartridge, or any other tangible medium or media from which a computer or machine or electronic data processing device can read and use. In essence, as used herein, non-transitory computer-readable and/or machine-readable mediums and/or media comprise all computer-readable and/or machine-readable mediums and/or media except for a transitory, propagating signal.

[00119] Optionally, any one or more of the particular tasks, steps, processes, methods, functions, elements and/or components described herein may be implemented on and/or embodiment in one or more general purpose computers, special purpose computer(s), a programmed microprocessor or microcontroller and peripheral integrated circuit elements, an ASIC or other integrated circuit, a digital signal processor, a hardwired electronic or logic circuit such as a discrete element circuit, a programmable logic device such as a PLD, PLA, FPGA, Graphical card CPU (GPU), or PAL, or the like. In general, any device, capable of implementing a finite state machine that is in turn capable of implementing the respective tasks, steps, processes, methods and/or functions described herein can be used.

[00120] Additionally, it is to be appreciated that certain elements described herein as incorporated together may under suitable circumstances be stand-alone elements or otherwise divided. Similarly, a plurality of particular functions described as being carried out by one particular element may be carried out by a plurality of distinct elements acting independently to carry out individual functions, or certain individual functions may be split-

up and carried out by a plurality of distinct elements acting in concert. Alternately, some elements or components otherwise described and/or shown herein as distinct from one another may be physically or functionally combined where appropriate.

[00121] In short, the present specification has been set forth with reference to preferred embodiments. Obviously, modifications and alterations will occur to others upon reading and understanding the present specification. It is intended that the invention be construed as including all such modifications and alterations insofar as they come within the scope of the appended claims or the equivalents thereof.

[00122] What is claimed is:

CLAIMS

1. A method for automatically generating an augmented model of a physical component, said method comprising:

reading an input model into a processor, said input model describing a nominal mode of operation for a physical component modeled by said input model;

parsing with said processor said input model to generate a parse thereof;

analyzing with said processor the parse of said input model; and

automatically writing with said processor an augmented model for the physical component from said input model based on said analysis, said augmented model describing both (i) the nominal mode of operation for the modeled physical component and (ii) at least one alternate mode of operation for the modeled physical component which is different from the nominal mode of operation;

wherein said analyzing is conducted according to at least one of a first approach and a second approach;

wherein said first approach operates to detect if the input model is susceptible to one or more kind of fault including: (i) catastrophic, (ii) power flow, and (iii) parametric kinds of faults; and

wherein the second approach operates to match specific facets of a component's dynamic behavior, and appropriately modify those matched facets to reflect the dynamics of a fault mechanism.

2. The method of claim 1, wherein: (i) a catastrophic kind of fault represents a change in the dynamics of the component such that the component functions as something completely different from what it is, (ii) a power flow kind of fault affects one or more of the power flows to or from the component, and (iii) a parametric kind of fault reflects a shifting of a supposedly fixed parameter of the component.

3. The method of claim 1, wherein said analyzing in accordance with the first approach comprises:

searching said parse for instances of at least one of power interfaces or connector classes, such that if at least one of a power interface or a connector class is found, then an alternate mode related thereto is described in the augmented model.

4. The method of claim 1, wherein said analyzing in accordance with the second approach comprises:

detecting a pattern in said parse which matches a target pattern associated with the alternate mode to be described in said augmented model, such that if a pattern is detected which matches said target pattern, then the associated alternate mode is described in the augmented model.

5. The method of claim 1, wherein said writing comprises:

providing a control mechanism by which one of the modes of operation described in the augment model is selected for use in a simulation employing said augmented model.

6. The method of claim 1, further comprising:

obtaining a mission definition for a system of one or more components;

obtaining a design configurations representing the system of components;

extracting design parameters from the design configuration;

extracting information from the design parameters;

establishing a Multi-Body Dynamics (MBD) simulation model based on the extracted information and running the MBD simulation model using the mission definition to achieve a set of results; and

generating at least one parametric map based on the achieved set of results and extracted design parameters.

7. The method of claim 6, wherein said parametric map reflects a fatigue of a component within the system and said method further comprising:

determining a fatigue life of said component from said parametric map.

8. The method of 7, wherein the parametric map relates (i) a probability of fatigue failure of the component as a result of stress incurred due to exposure of the component to conditions reflected by the mission definition, to (ii) an array of design parameters.

9. The method of claim 8, wherein the MBD simulation model is run a plurality of times to achieve a multi-point cloud of results where the points in the cloud represent an array of different design parameters, and the method further comprises:

interpolating values for the results between the points in the cloud.

10. A system for automatically generating an augmented model of a physical component, said system comprising:

a processor operative to:

read in an input model, said input model describing a nominal mode of operation for a physical component modeled by said input model;

parse said input model to generate a parse thereof;

analyze the parse of said input model; and

automatically write an augmented model for the physical component from said input model based on said analysis, said augmented model describing both (i) the nominal mode of operation for the modeled physical component and (ii) at least one alternate mode of operation for the modeled physical component which is different from the nominal mode of operation;

wherein analyzing the parse is conducted according to at least one of a first approach and a second approach;

wherein said first approach operates to detect if the input model is susceptible to one or more kind of fault including: (i) catastrophic, (ii) power flow, and (iii) parametric kinds of faults; and

wherein the second approach operates to match specific facets of a component's dynamic behavior, and appropriately modify those matched facets to reflect the dynamics of a fault mechanism.

11. The system of claim 10, wherein: (i) a catastrophic kind of fault represents a change in the dynamics of the component such that the component functions as something completely different from what it is, (ii) a power flow kind of fault affects one or more of the power flows to or from the component, and (iii) a parametric kind of fault reflects a shifting of a supposedly fixed parameter of the component.

12. The system of claim 10, wherein said analyzing in accordance with the first approach comprises:

searching said parse for instances of at least one of power interfaces or connector classes, such that if at least one of a power interface or a connector class is found, then an alternate mode related thereto is described in the augmented model.

13. The system of claim 10, wherein said analyzing in accordance with the second approach comprises:

detecting a pattern in said parse which matches a target pattern associated with the alternate mode to be described in said augmented model, such that if a pattern is detected which matches said target pattern, then the associated alternate mode is described in the augmented model.

14. The system of claim 10, wherein said processor is further operative to:

obtain a mission definition for a system of one or more components;

obtain a design configurations representing the system of components;

extract design parameters from the design configuration;

extract information from the design parameters;

establish a Multi-Body Dynamics (MBD) simulation model based on the extracted information and running the MBD simulation model using the mission definition to achieve a set of results; and

generate at least one parametric map based on the achieved set of results and extracted design parameters.

15. The system of claim 14, wherein said parametric map reflects a fatigue of a component within the system and said method further comprising:

determining a fatigue life of said component from said parametric map.

16. The system of claim 15, wherein the parametric map relates (i) a probability of fatigue failure of the component as a result of stress incurred due to exposure of the component to conditions reflected by the mission definition, to (ii) an array of design parameters.

17. The system of claim 16, wherein the MBD simulation model is run a plurality of times to achieve a multi-point cloud of results where the points in the cloud represent an array of different design parameters, and the method further comprises:

interpolating values for the results between the points in the cloud.

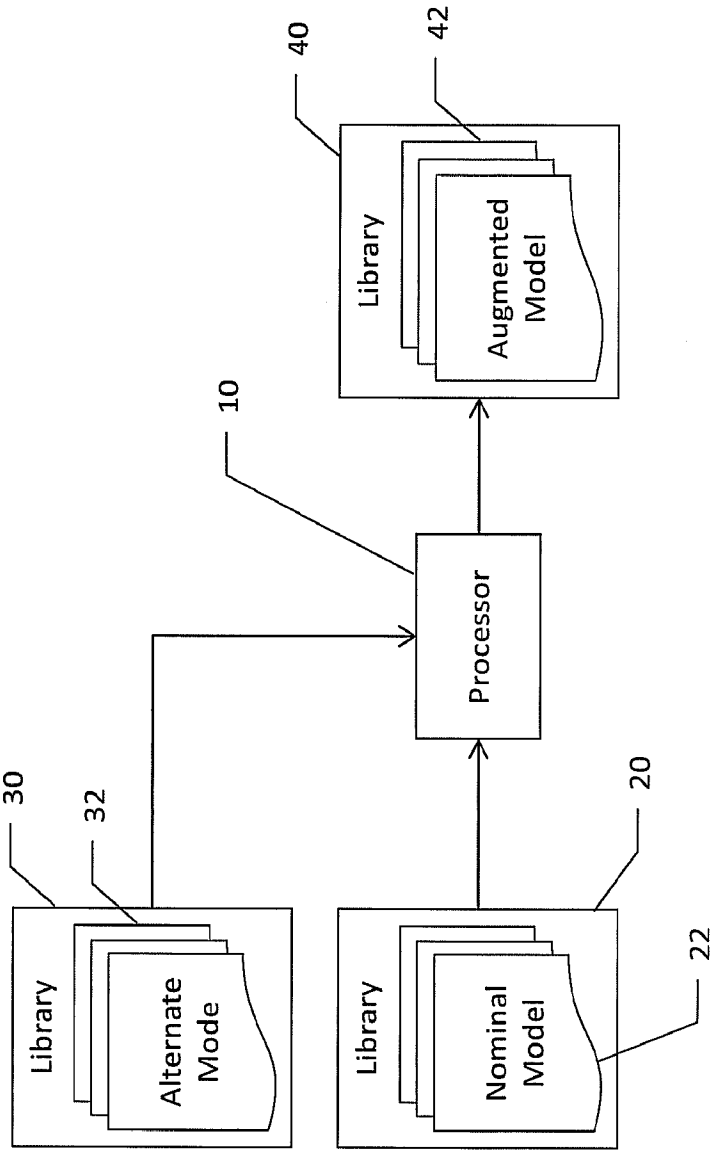
40
42
Library
Augmented Model
10
Processor
30
32
Library
Alternate Mode
20
Library
Nominal Model
22

FIG. 1

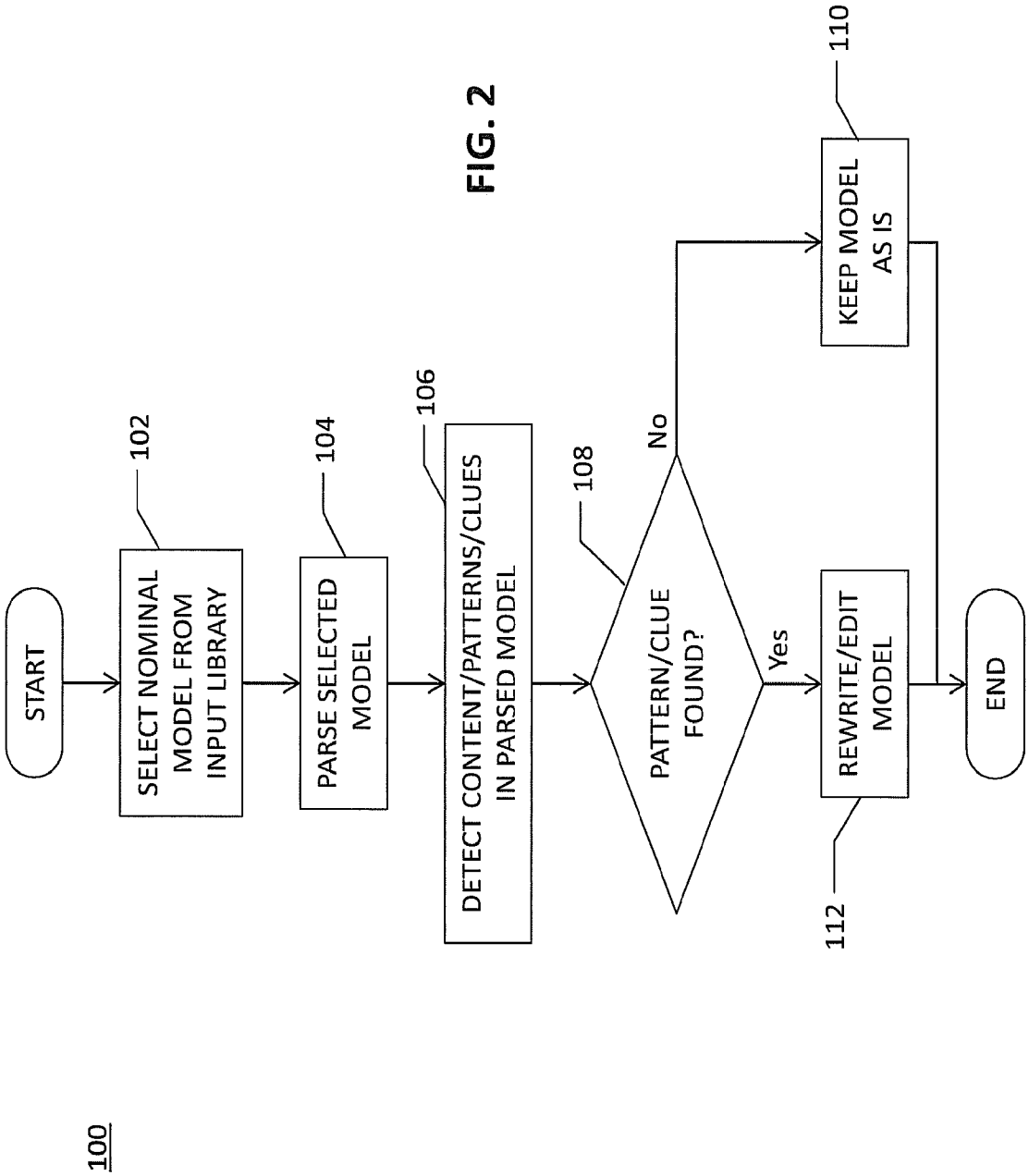
100
START
SELECT NOMINAL MODEL FROM INPUT LIBRARY
102
PARSE SELECTED MODEL
104
DETECT CONTENT/PATTERNS/CLUES IN PARSED MODEL
106
PATTERN/CLUE FOUND?
108
No
Yes
KEEP MODEL AS IS
110
REWRITE/EDIT MODEL
112
END

FIG. 2

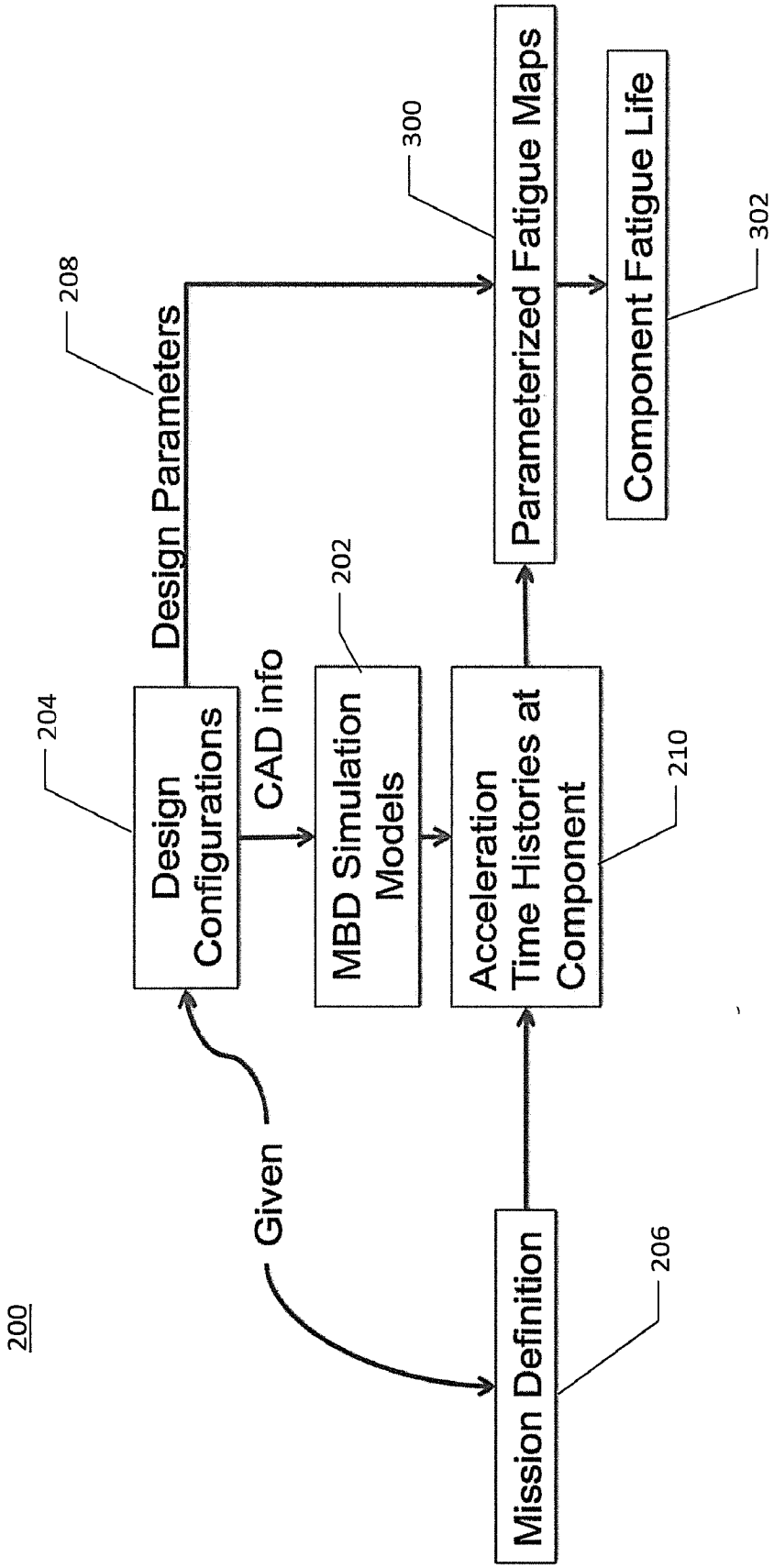
200
Design Configurations
204
CAD info
MBD Simulation Models
202
Acceleration Time Histories at Component
210
Parameterized Fatigue Maps
300
Component Fatigue Life
302
Design Parameters
208
Given
Mission Definition
206

FIG. 3

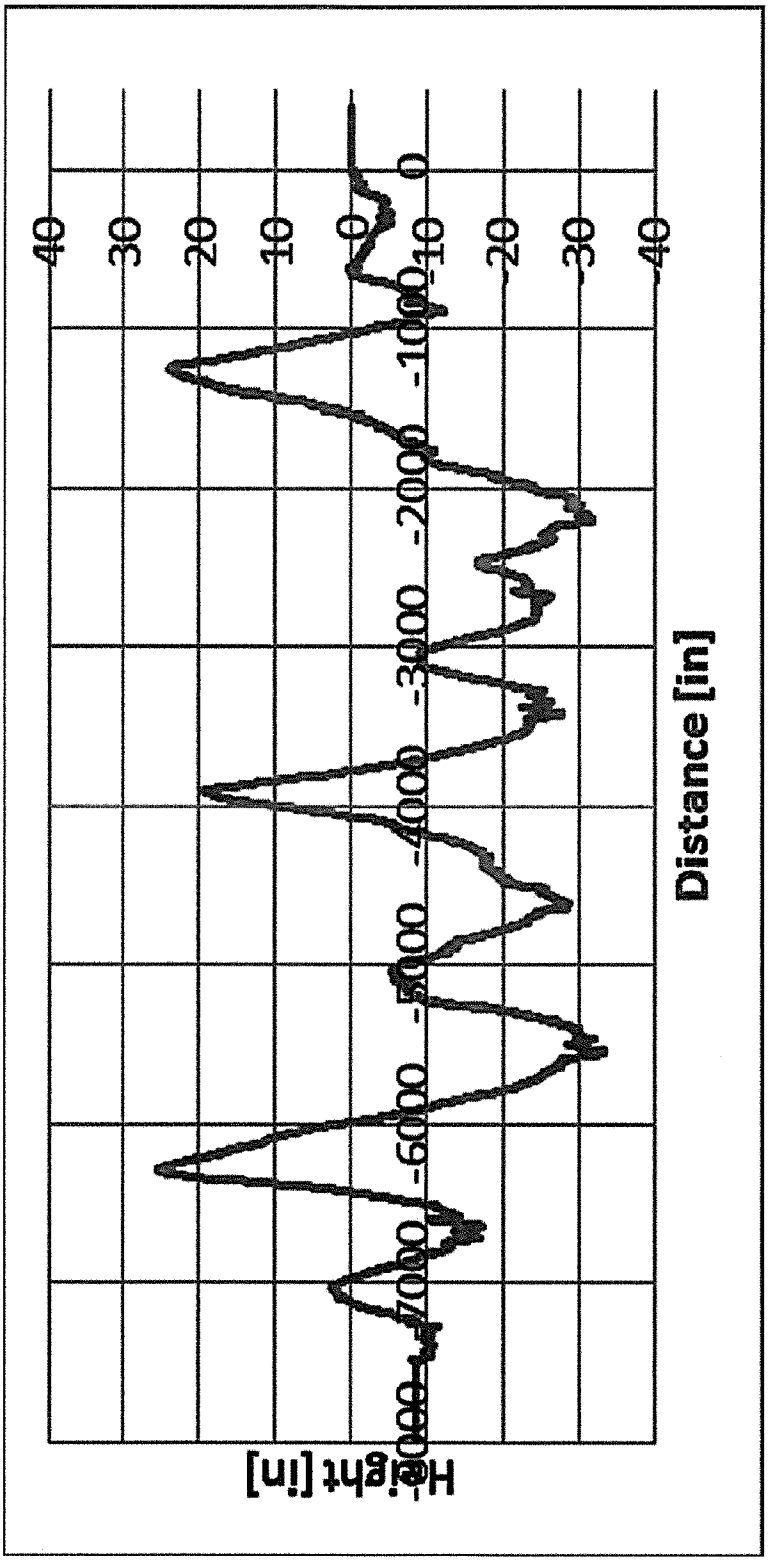
Height [in]
40
30
20
10
0
-10
-20
-30
-40
0
-1000
-2000
-3000
-4000
-5000
-6000
-7000
Distance [in]

FIG. 4

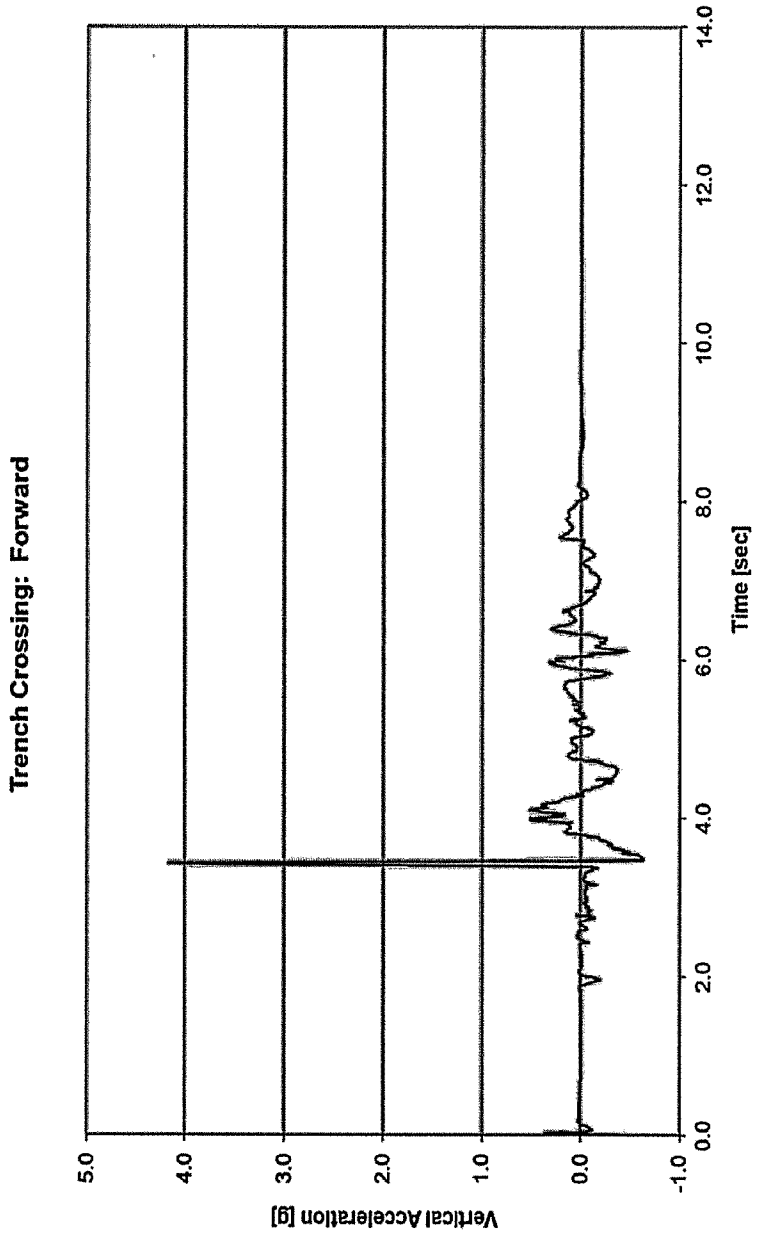
Trench Crossing: Forward
Vertical Acceleration [g]
5.0
4.0
3.0
2.0
1.0
0.0
-1.0
0.0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
Time [sec]

FIG. 5A

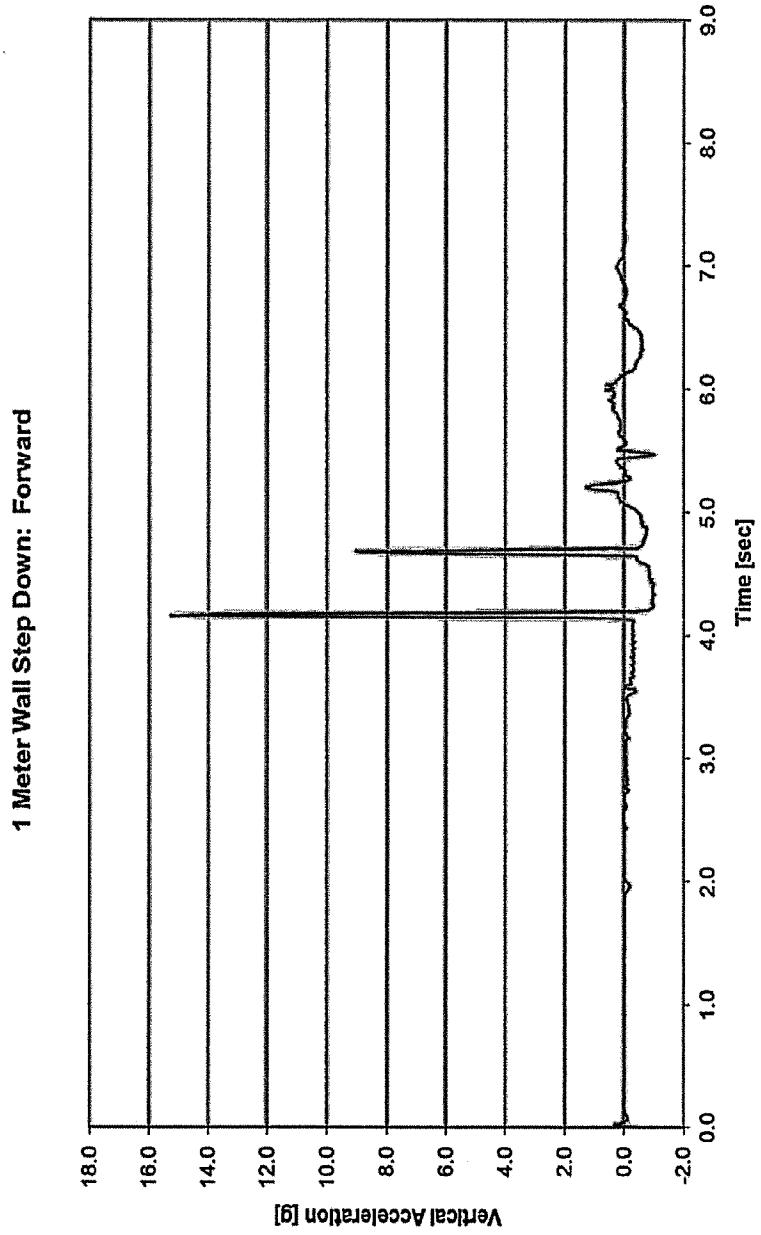
1 Meter Wall Step Down: Forward
Vertical Acceleration [g]
18.0
16.0
14.0
12.0
10.0
8.0
6.0
4.0
2.0
0.0
-2.0
Time [sec]
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0

FIG. 5B

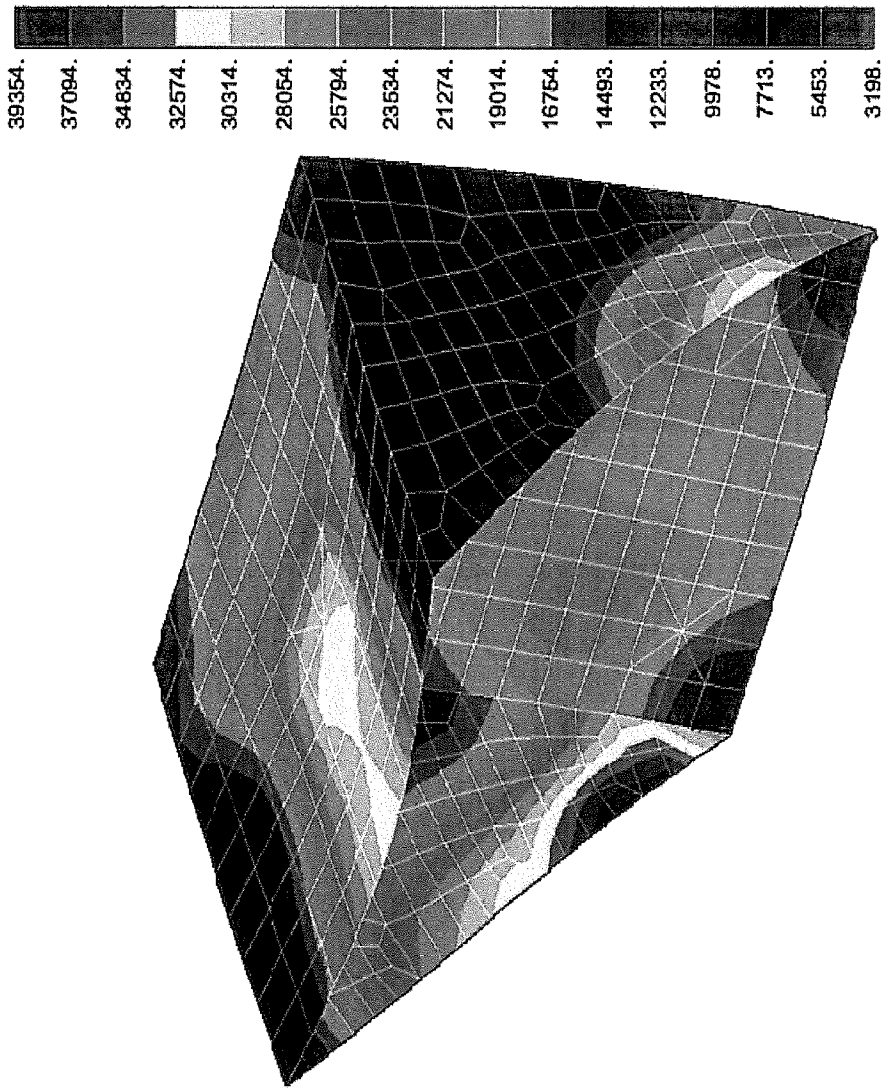
39354.
37094.
34834.
32574.
30314.
28054.
25794.
23534.
21274.
19014.
16754.
14493.
12233.
9978.
7713.
5453.
3198.

FIG. 6

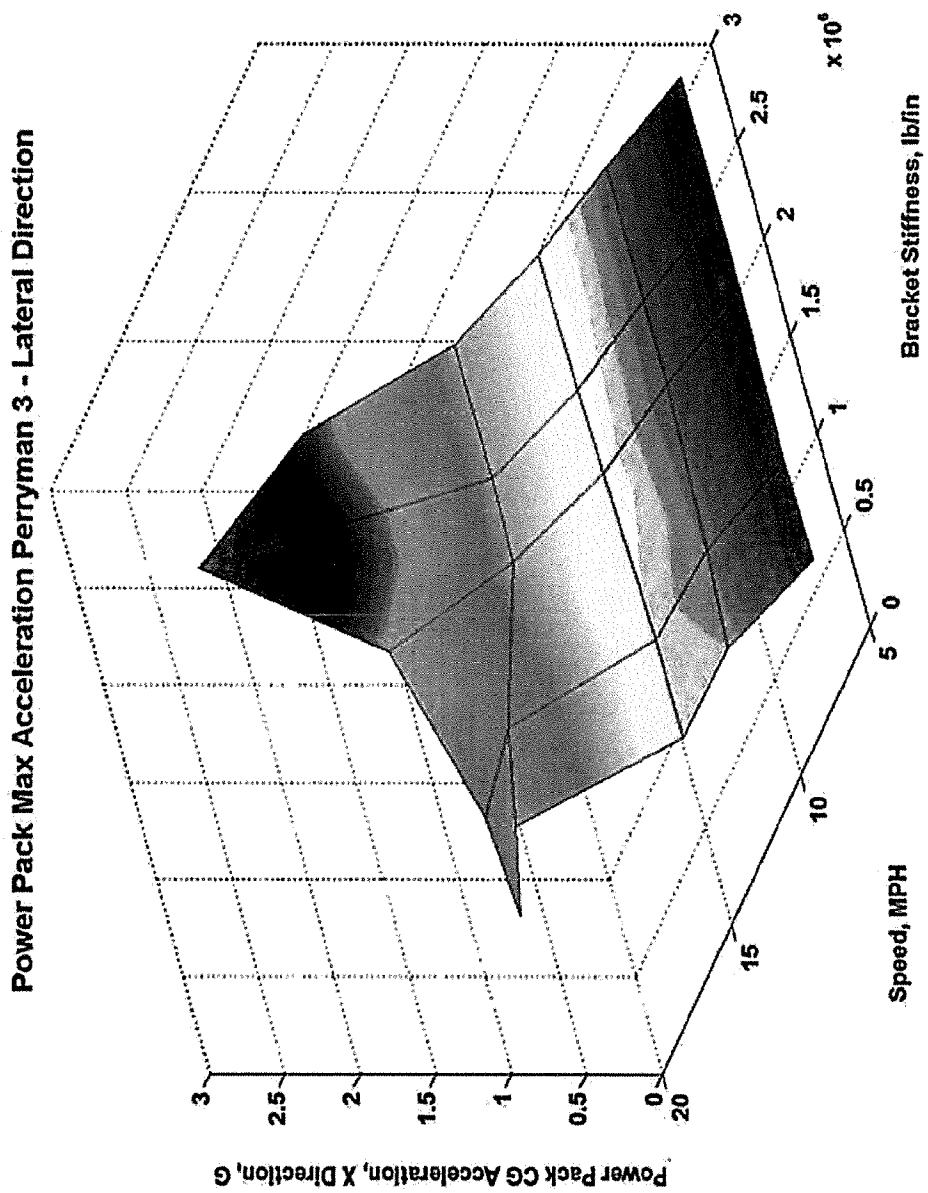
Power Pack Max Acceleration Perryman 3 - Lateral Direction
Power Pack CG Acceleration, X Direction, G
3
2.5
2
1.5
1
0.5
0
20
15
10
5
0
Speed, MPH
0.5
1
1.5
2
2.5
3
x 10^6
Bracket Stiffness, lb/in

FIG. 7

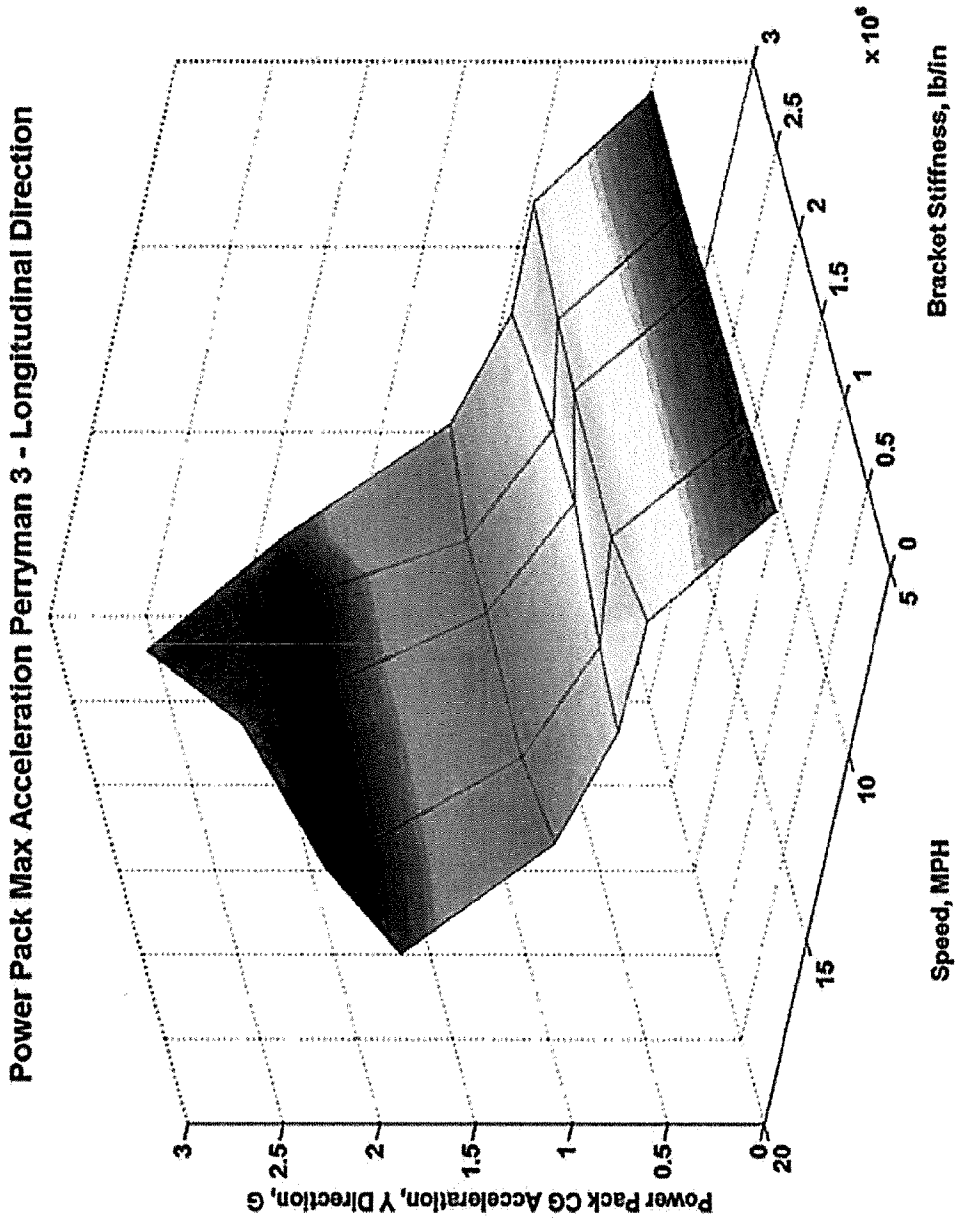
Power Pack Max Acceleration Perryman 3 - Longitudinal Direction
Power Pack CG Acceleration, Y Direction, G
0
0.5
1
1.5
2
2.5
3
Speed, MPH
0
5
10
15
20
Bracket Stiffness, lb/in
x 10^6
0
0.5
1
1.5
2
2.5
3

FIG. 8

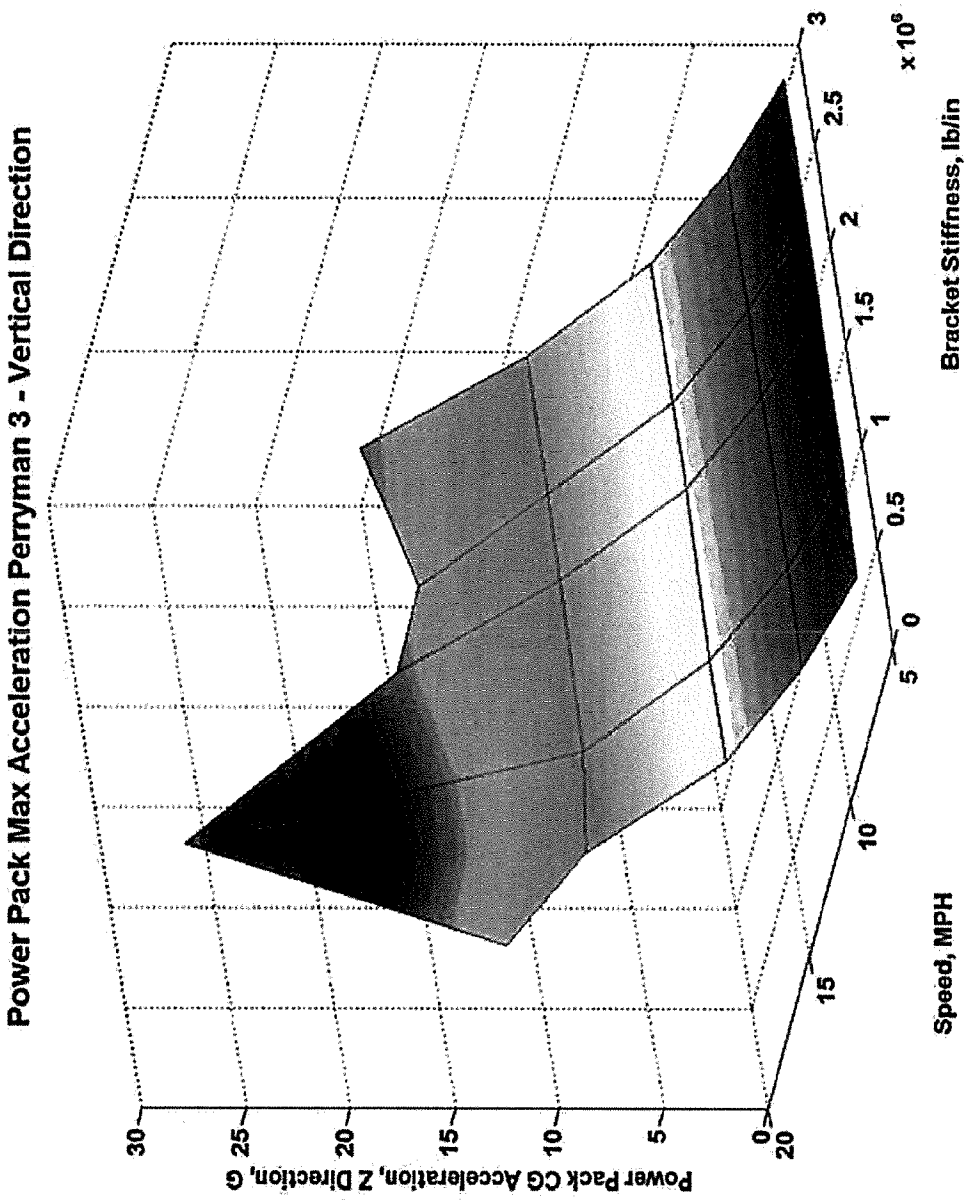
Power Pack Max Acceleration Perryman 3 - Vertical Direction
Power Pack CG Acceleration, Z Direction, G
30
25
20
15
10
5
0
20
15
10
5
0
Speed, MPH
0.5
1
1.5
2
2.5
3
x 10^6
Bracket Stiffness, lb/in

FIG. 9

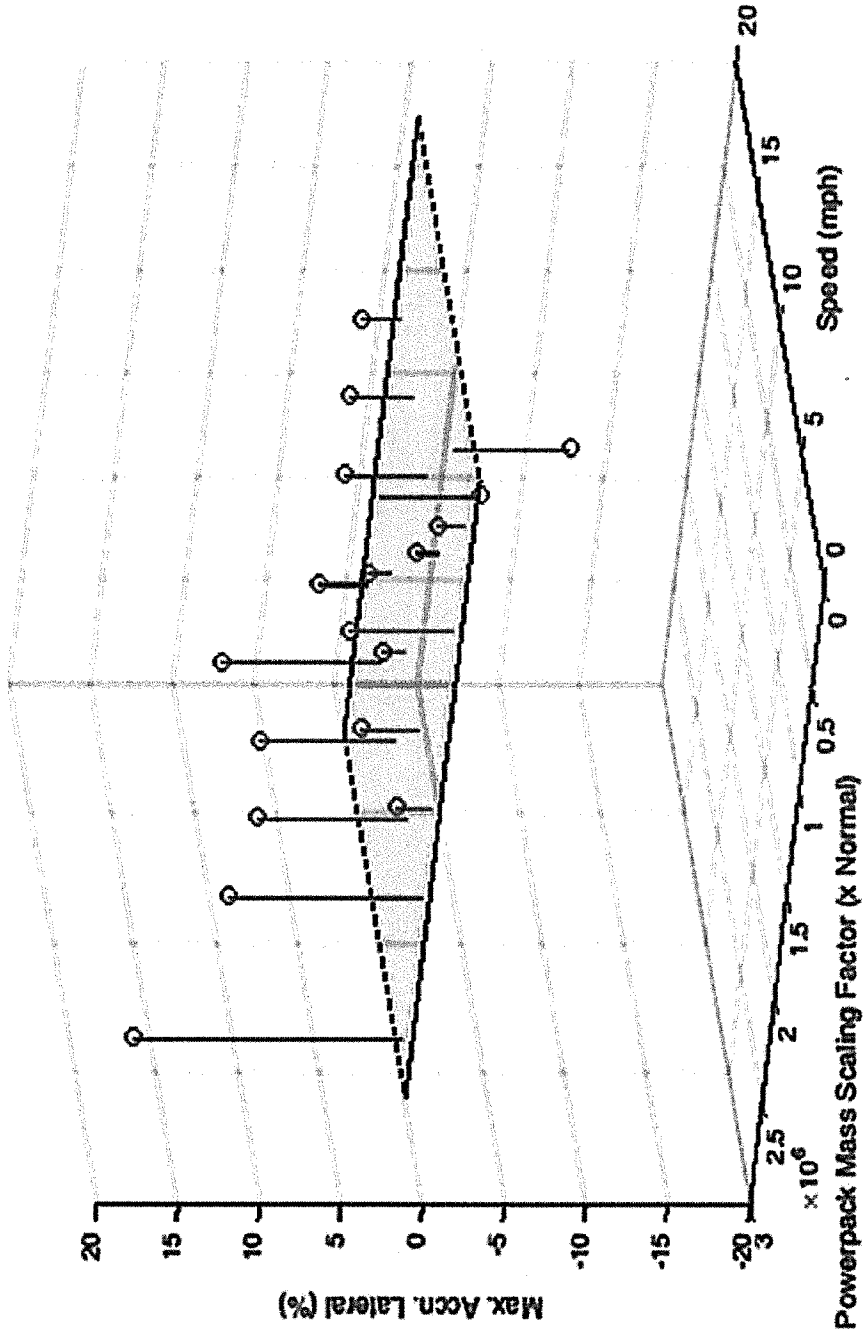
Max. Accn. Lateral (%)
20
15
10
5
0
-5
-10
-15
-20
Powerpack Mass Scaling Factor (x Normal)
0
0.5
1
1.5
2
2.5
×10^6
Speed (mph)
0
5
10
15
20

FIG. 10

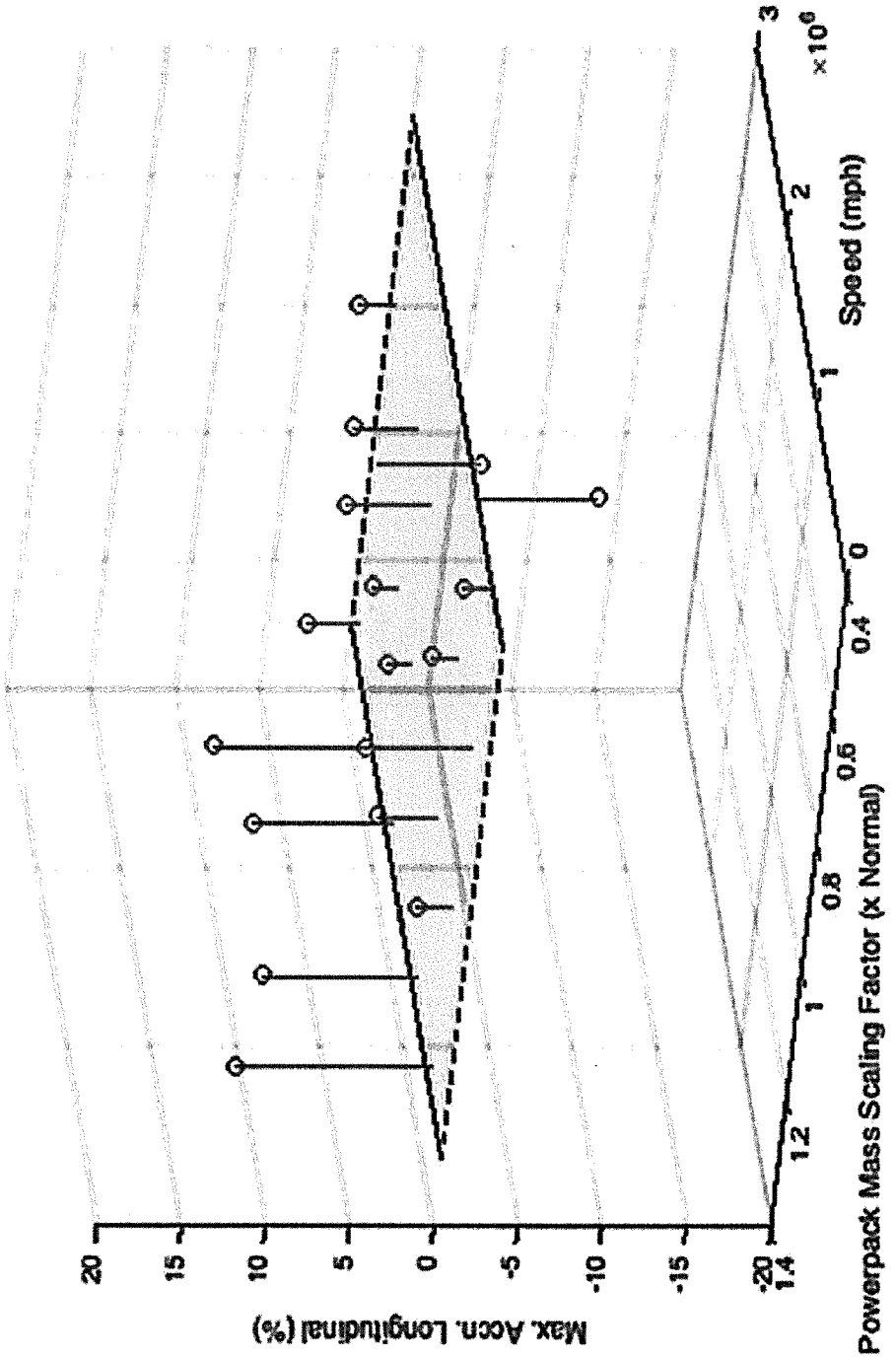
Max. Accn. Longitudinal (%)
20
15
10
5
0
-5
-10
-15
-20
Powerpack Mass Scaling Factor (x Normal)
1.4
1.2
1
0.8
0.6
0.4
0
Speed (mph)
×10^6
1
2
3

FIG. 11

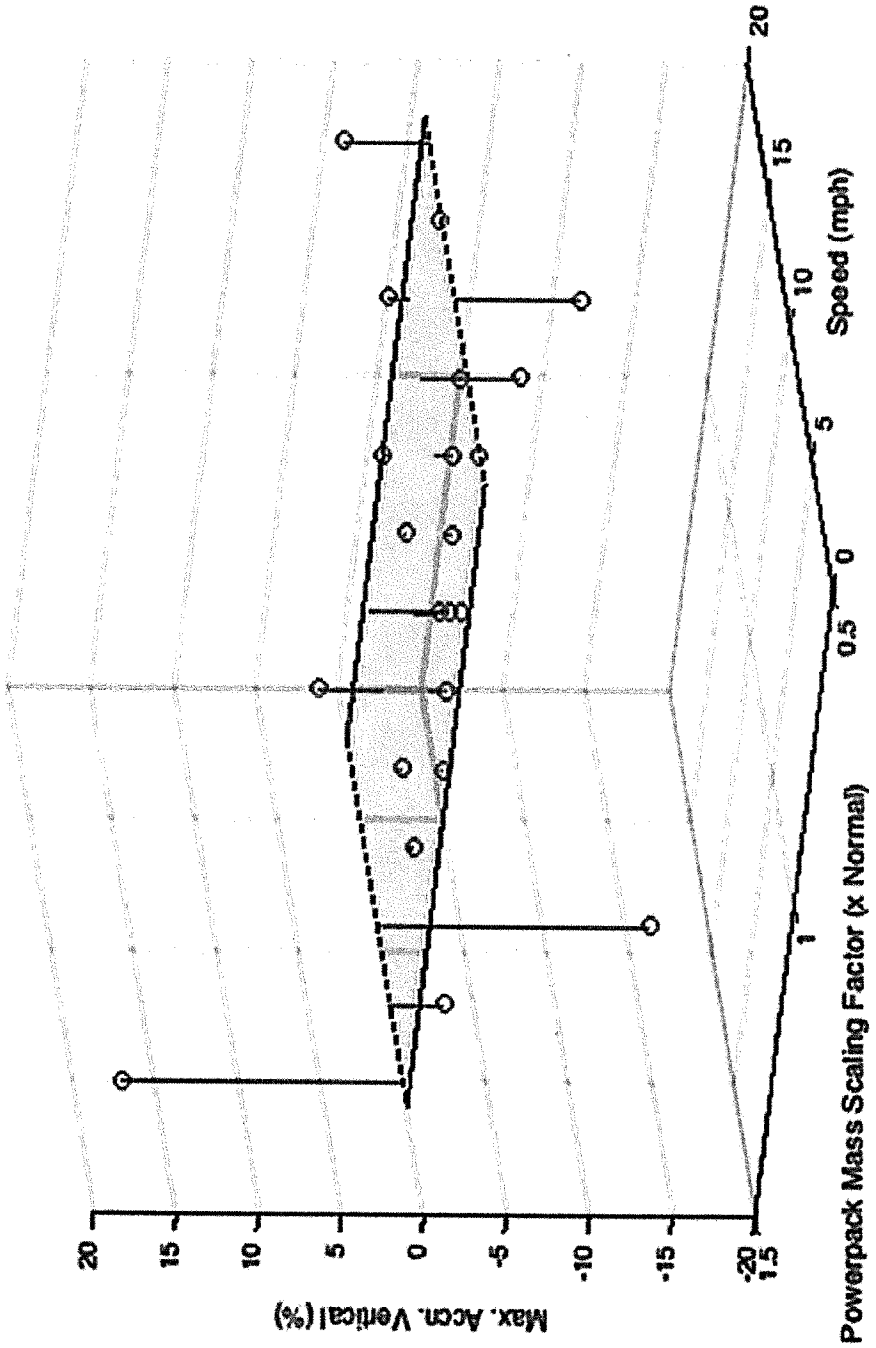
Max. Accn. Vertical (%)
20
15
10
5
0
-5
-10
-15
-20
Speed (mph)
20
15
10
5
0
Powerpack Mass Scaling Factor (x Normal)
0.5
1
1.5

FIG. 12

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US17/12049

A. CLASSIFICATION OF SUBJECT MATTER
IPC - G06F 17/50 (2017.01)
CPC - G06F 17/5009, 17/40, 8/35

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
See Search History document

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
See Search History document

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
See Search History document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2015/0051890 A1 (PALO ALTO RESEARCH CENTER INCORPORATED) 19 February 2015; abstract; paragraphs [0031], [0035], [0036], [0038], [0039]	1-5, 10-13
----		----
Y		6-9, 14-17
Y	US 2013/0268254 A1 (BRIDGESTONE CORPORATION) 10 October 2013; paragraphs [0043], [0045], [0046], [0047], [0052], [0053], [0061]	6-9, 14-17
Y	US 2003/0114995 A1 (SU, H et al.) 19 June 2003; abstract; paragraph [0026]	7-9, 15-17
Y	US 2005/0197814 A1 (ARAM, L et al.) 8 September 2005; paragraph [0050]	9, 17

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
2 March 2017 (02.03.2017)	24 MAR 2017
Name and mailing address of the ISA/ Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-8300	Authorized officer Shane Thomas PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774