(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2004-527027
(P2004-527027A)
(43)公表日　平成16年9月2日(2004.9.2)

(51)Int.Cl.[7]	F I	テーマコード（参考）
G06F 19/00	G06F 19/00 110	

審査請求 未請求　予備審査請求 有　（全 111 頁）

(21)出願番号	特願2002-561758(P2002-561758)
(86)(22)出願日	平成13年11月2日(2001.11.2)
(85)翻訳文提出日	平成15年5月2日(2003.5.2)
(86)国際出願番号	PCT/US2001/051360
(87)国際公開番号	WO2002/061662
(87)国際公開日	平成14年8月8日(2002.8.8)
(31)優先権主張番号	60/245,731
(32)優先日	平成12年11月2日(2000.11.2)
(33)優先権主張国	米国(US)
(31)優先権主張番号	60/245,688
(32)優先日	平成12年11月2日(2000.11.2)
(33)優先権主張国	米国(US)
(31)優先権主張番号	60/245,730
(32)優先日	平成12年11月2日(2000.11.2)
(33)優先権主張国	米国(US)

(71)出願人　503163561
プロテイン　メカニクス，　インコーポレイテッド
アメリカ合衆国　カリフォルニア　９４０４１，　マウンテン　ヴュー，　ホープ　ストリート　２７８，　スイート　シー
(74)代理人　100078282
弁理士　山本　秀策
(74)代理人　100062409
弁理士　安村　高明
(74)代理人　100113413
弁理士　森下　夏樹

最終頁に続く

(54)【発明の名称】分子モデリングにおける解析的ヤコビアン計算のための方法

(57)【要約】
物理系の動力学についての計算における陰的積分法で使用される解析的ヤコビアンを得る方法。この方法を用いて、数値的ヤコビアンの少なくとも２倍の有効桁数を有するヤコビアンが計算され得る。これはまた、運動方程式の非線形ステージ方程式を解くためにより少ない反復回数が必要とされるため、より効率的な陰的積分法を生じる。計算におけるスピードの向上は、分子モデリングにおいて非常に有用である。本発明はまた、ねじれ角、剛体多体系によってモデリングされ得る、分子の挙動、または任意の物理系をシミュレートするための、コンピュータコードを提供する。陰的積分器を使用する、コンピュータコードにおけるモジュールは、上記のような解析的ヤコビアンを含む。

【特許請求の範囲】
【請求項１】
分子の挙動をモデリングする方法であって、該方法は、以下：
該分子について、ねじれ角、剛体多体系モデルを選択する工程であって、該モデルが、運動方程式を有する、工程；
陰的積分器を選択する工程；および
解析的ヤコビアンを生成して、該陰的積分器が該運動方程式を積分して、該分子の挙動の計算を得る、工程、
を包含する、方法。
【請求項２】
前記解析的ヤコビアンが、前記運動方程式の残差形式の解析的ヤコビアンから導出される、請求項１に記載の方法。
【請求項３】
請求項２に記載の方法であって、前記解析的ヤコビアンＪが、以下：
【表１】

$$J=\begin{pmatrix}\frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial u}\\ \frac{\partial \dot{u}}{\partial q} & \frac{\partial \dot{u}}{\partial u}\end{pmatrix}\triangleq\begin{pmatrix}J_{qq} & J_{qu}\\ J_{uq} & J_{uu}\end{pmatrix};\ \text{および}$$

$$J_{qq}=\frac{\partial \dot{q}}{\partial q}=\frac{\partial (Wu)}{\partial q}\ \text{および}\ J_{qu}=\frac{\partial \dot{q}}{\partial u}=W$$

$$J_{uq}=\frac{\partial \dot{u}}{\partial q}=-M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial q}\ \text{および}\ J_{uu}=\frac{\partial \dot{u}}{\partial u}=-M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial u}$$

を含み、ここで、ｑは、一般化座標であり、ｕは一般化速度であり、Ｗはジョイントマップ行列であり、そしてＭは質量行列であり、そしてρｕは、前記運動方程式の動力学的残差であり、そしてｚは、－Ｍ－１ρｕ（ｑ、ｕ、０）である、方法。
【請求項４】
前記陰的積分器を選択する工程がＬ安定性積分器を含む、請求項３に記載の方法。
【請求項５】
物理系の挙動をシミュレートする方法であって、該方法は、ねじれ角、剛体モデルを用いて該物理系をモデリングする工程であって、該モデルは、運動方程式を有する工程；および
陰的積分器を用いて該運動方程式を積分する工程；
を包含し、該陰的積分器が解析的ヤコビアンを有し、該物理系の該挙動の計算を得る、方法。
【請求項６】
前記解析的ヤコビアンが前記運動方程式の残差形式の解析的ヤコビアンから導出される、請求項５に記載の方法。
【請求項７】
請求項６に記載の方法であって、前記解析的ヤコビアンＪは、以下：

【表２】

$$J=\begin{pmatrix}\frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial u}\\ \frac{\partial \dot{u}}{\partial q} & \frac{\partial \dot{u}}{\partial u}\end{pmatrix}\triangleq\begin{pmatrix}J_{qq} & J_{qu}\\ J_{uq} & J_{uu}\end{pmatrix};\ \text{および}$$

$$J_{qq}=\frac{\partial \dot{q}}{\partial q}=\frac{\partial (Wu)}{\partial q}\ \text{および}\ J_{qu}=\frac{\partial \dot{q}}{\partial u}=W$$

$$J_{uq}=\frac{\partial \dot{u}}{\partial q}=-M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial q}\ \text{および}\ J_{uu}=\frac{\partial \dot{u}}{\partial u}=-M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial u}$$

を含み、ここで、ｑは、一般化座標であり、ｕは一般化速度であり、Ｗはジョイントマップ行列であり、そしてＭは質量行列であり、そしてρｕは、前記運動方程式の動力学的残差であり、そしてｚは、－Ｍ－１ρｕ（ｑ、ｕ、０）である、方法。

【請求項８】

前記陰的積分器がＬ安定性積分器を含む、請求項７に記載の方法。

【請求項９】

分子の挙動をシミュレーションするためのコンピュータコードであって、該コードが、以下：

該分子のねじれ角、剛体多体系モデルのための第１のモジュールであって、該モデルが、運動方程式を有する、第１のモジュール；および

陰的積分器が解析的ヤコビアンを使用して該運動方程式を積分し、該分子の挙動の計算を得る、第２のモジュール；

を備える、コンピュータコード。

【請求項１０】

前記解析的ヤコビアンが前記運動方程式の残差形式の解析的ヤコビアンから導出される、請求項９に記載のコンピュータコード。

【請求項１１】

請求項１０に記載のコンピュータコードであって、前記解析的ヤコビアンＪが、以下：

【表３】

$$J=\begin{pmatrix}\frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial u}\\ \frac{\partial \dot{u}}{\partial q} & \frac{\partial \dot{u}}{\partial u}\end{pmatrix}\triangleq\begin{pmatrix}J_{qq} & J_{qu}\\ J_{uq} & J_{uu}\end{pmatrix};\ \text{および}$$

$$J_{qq}=\frac{\partial \dot{q}}{\partial q}=\frac{\partial (Wu)}{\partial q}\ \text{および}\quad J_{qu}=\frac{\partial \dot{q}}{\partial u}=W$$

$$J_{uq}=\frac{\partial \dot{u}}{\partial q}=-M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial q}\ \text{および}\ J_{uu}=\frac{\partial \dot{u}}{\partial u}=-M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial u}$$

を含み、ここで、ｑは、一般化座標であり、ｕは一般化速度であり、Ｗはジョイントマップ行列であり、そしてＭは、質量行列であり、そしてρｕは、前記運動方程式の動力学的残差であり、そしてｚは、－Ｍ－１ρｕ（ｑ、ｕ、０）である、コンピュータコード。

【請求項１２】

前記陰的積分器がＬ安定性積分器である、請求項１１に記載のコンピュータコード。
【請求項１３】
物理系の挙動をシミュレーションするためのコンピュータコードであって、該コードは、以下：
該系のねじれ角、剛体多体系モデルのための第１のモジュールであって、該モデルが、運動方程式を有する、第１のモジュール；および
陰的積分器が解析的ヤコビアンを使用して該運動方程式を積分し、該分子の挙動の計算を得る、第２のモジュール；
を備える、コンピュータコード。
【請求項１４】
前記解析的ヤコビアンが前記運動方程式の残差形式の解析的ヤコビアンから導出される、請求項１３に記載のコンピュータコード。
【請求項１５】
請求項１４に記載のコンピュータコードであって、前記解析的ヤコビアンＪが、以下：
【表４】

$$J=\begin{pmatrix}\frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial u}\\ \frac{\partial \dot{u}}{\partial q} & \frac{\partial \dot{u}}{\partial u}\end{pmatrix}\triangleq\begin{pmatrix}J_{qq} & J_{qu}\\ J_{uq} & J_{uu}\end{pmatrix};\quad \text{および}$$

$$J_{qq}=\frac{\partial \dot{q}}{\partial q}=\frac{\partial (Wu)}{\partial q}\ \text{および}\quad J_{qu}=\frac{\partial \dot{q}}{\partial u}=W$$

$$J_{uq}=\frac{\partial \dot{u}}{\partial q}=-M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial q}\ \text{および}\quad J_{uu}=\frac{\partial \dot{u}}{\partial u}=-M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial u}$$

を含み、ここで、ｑは、一般化座標であり、ｕは一般化速度であり、Ｗはジョイントマップ行列であり、そしてＭは、質量行列であり、そしてρｕは、前記運動方程式の動力学的残差であり、そしてｚは、－Ｍ－１ρｕ（ｑ、ｕ、０）である、コンピュータコード。
【請求項１６】
前記陰的積分器がＬ安定性積分器を含む、請求項１５に記載のコンピュータコード。
【発明の詳細な説明】
【０００１】
（関連出願の引用参照）
本出願は、仮特許出願第６０／２４５，７３０号（２０００年１１月２日に出願）；および、さらに、第６０／２４５，６８８号（２０００年１１月２日に出願）；第６０／２４５，７３１号（２０００年１１月２日に出願）；ならびに第６０／２４５，７３４号（２０００年１１月２日に出願）の優先出願日の利益を享受し、これらの出願の全てを本明細書中で参考として援用する。
【０００２】
（発明の背景）
本発明は分子モデリングの分野に関し、より詳細には、高分子の動力学的モデリングおよび静的解析のためのコンピュータにインプリメントされた方法に関する。
【０００３】
分子モデリングの分野は、コンピュータによって多くの複雑な分子系の、その運動を首尾よくシミュレート（分子動力学法、すなわち（ＭＤ））し、エネルギー極小状態または静止状態（静的解析）を決定してきた。代表的な分子モデリング用途としては、酵素－リガンドドッキング、分子拡散、反応経路、相転移、およびタンパク質のフォールディングの

研究が挙げられる。生命科学、ならびに、製薬、ポリマー、および化学産業の研究者が、複雑な分子の化学的過程の性質を理解し、それに従って新規の薬物および材料を設計するために、それらの技術を使用し始めている。本来、これらのツールの受容は、現実のものを表現するときの結果の精度、モデルリングされ得るべき分子系のサイズおよび複雑さ、および解が得られる速度を含む、いくつかの要因に基づく。多くの計算の精度は実験と比較され、一般的に特定の境界内では適切であると理解される。しかし、従来技術におけるこれらのツールの使用は、適度な大きさの分子または分子系でさえも、それらをモデリングして有用である十分な長さの分子の時間履歴を得るために膨大な計算パワーを、必要としてきた。

【０００４】

時間積分を含む分子モデリング課題についての計算の複雑さの２つの原因が存在する。

【０００５】

１．特定の分子モデルであって、構成原子の位置、速度、および質量特性、構成原子間の原子間力、ならびにそれらの原子と構成原子の周囲環境との間の相互作用を記述するために使用される、分子モデル；ならびに

２．経時的にモデルを前進させるために使用される、特定の数値的方法。

時刻は、最終時刻に達するまで、非常に短い間隔（これは、時間刻みと呼称される）分だけ、繰り返し進められる。

【０００６】

実質的な研究は、例えば、以下のように分子モデルについての計算負荷を低減することにおいて完結した：剛体仮定条件で高次のモードを拘束することによって、モデルの複雑性の低減すること、剛体性または伸縮性の下部構造分解（ｓｕｂｓｔｒｕｃｔｕｒｉｎｇ）によってモデルを単純化すること、Ｎ次動力学、有効性のある非明示的な溶媒モデル、および力場モデルについての多重極法（ｍｕｌｔｉｐｏｌｅ　ｍｅｔｈｏｄ）（例えば、市販のＭＢＯ（Ｎ）Ｄソフトウエアパッケージに関する米国特許第５，４２４，９６３号を参照のこと））。同時係属出願である、「ＭＥＴＨＯＤ　ＦＯＲ　ＬＡＲＧＥ　ＴＩＭＥ　ＳＴＥＰＳ　ＩＮ　ＭＯＬＥＣＵＬＡＲ　ＭＯＤＥＬＩＮＧ」と題する、米国特許出願番号＿＿＿＿、および「ＭＥＴＨＯＤ　ＦＯＲ　ＲＥＳＩＤＵＡＬ　ＦＯＲＭ　ＩＮ　ＭＯＬＥＣＵＬＥＲ　ＭＯＤＥＬＩＮＧ」と題する、米国特許出願番号＿＿＿＿（これらの両方は、同日付けで出願され、そして、上で引用した仮特許出願の優先権を主張し、本譲渡受人に譲渡され、そしてこれらは、本明細書中において参考として援用される）によって、分子モデリングおよび数値的方法におけるさらなる改善が記載される。

【０００７】

分子シミュレーションが非常に遅い主要な原因は、現在の数値的方法が非常に小さな時間刻み（代表的に、１～１０フェムト秒（10^{-15}～10^{-14}秒））を必要することである。採用された各時間刻みは、特定の分子モデルの新しい状態（各原子についての位置および運動）の計算を必要とし、次いで、力の新しいセットの計算は、この新しい状態から生じる。例えば、高分子の複雑な挙動（例えば、タンパク質のフォールディング）の分子動力学シミュレーションは、代表的には、少なくとも１マイクロ秒～数秒（数分でさえも）の時間範囲に及ぶ必要がある。現在、通常用いられている技術によって、これは、コンピュータシミュレーションにおいて10^{9}～10^{16}の時間刻みをとる必要性を生じる。この状態、および特に力の、刻み当たりの計算は、問題の規模が大きくなるにつれて非常に高くなる。今日利用可能な最高速のコンピュータを用いたとしても、数ヶ月、数年、または数百年のコンピュータ処理時間が、適度な大きさの系であってもこのような問題を解決するために必要とされる。

【０００８】

これまで、これらの小さな時間刻みは、分子結合の振動において見出される非常に高い振動数の存在下で精度を維持するための要求の、不可避な必要条件であることが分子動力学者によって広く信じられてきた。例えば、Ｌｅａｃｈによる、Ｍｏｌｅｃｕｌａｒ　Ｍｏｄｅｌｌｉｎｇ　Ｐｒｉｎｃｉｐｌｅｓ　ａｎｄ　Ａｐｐｌｉｃａｔｉｏｎｓ，１９９６

，ｐ．３２８；Ｂｅｒｅｎｄｓｅｎによる、Ｃｏｍｐｕｔａｔｉｏｎａｌ Ｍｏｌｅｃｕｌａｒ Ｄｙｎａｍｉｃｓ：Ｃｈａｌｌｅｎｇｅｓ，Ｍｅｔｈｏｄｓ，Ｉｄｅａｓ；Ｄｅｕｆｌｈａｒｄら（編）による、Ｓｐｒｉｎｇｅｒ，１９９９，ｐｇ．１８；Ｒａｐａｐｏｒｔによる、Ｔｈｅ Ａｒｔ ｏｆ Ｍｏｌｅｃｕｌａｒ Ｄｙｎａｍｉｃｓ Ｓｉｍｕｌａｔｉｏｎ，Ｃａｍｂｒｉｄｇｅ，１９９５（訂正と共に再版された１９９８年、ｐ．５７）；および米国特許第５，４２４，９６３号を参照のこと。

【０００９】

しかし、この常識的な確信は誤りである。数値解析のうちのコンピュータサイエンス下位区分によって、高振動数が存在するが、それがほとんど関心事とならない問題に対する、数値積分法の広範な理論が生まれてきた。これらの問題は、「スティッフ（ｓｔｉｆｆ）」問題と呼称される（例えば、ＨａｉｒｅｒおよびＷａｎｎｅｒ、Ｓｏｌｖｉｎｇ Ｏｒｄｉｎａｒｙ Ｄｉｆｆｅｒｅｎｔｉａｌ Ｅｑｕａｔｉｏｎｓ ＩＩ：Ｓｔｉｆｆ ａｎｄ Ｄｉｆｆｅｒｅｎｔｉａｌ－Ａｌｇｅｂｒａｉｃ Ｐｒｏｂｌｅｍｓ，第２版、Ｓｐｒｉｎｇｅｒ，１９９６を参照のこと）。これらの場合においては、時間刻みを制限するものは、これらの積分法の安定性であって、その必要とされる解の精度ではない。無条件な安定性を有する積分法が存在し、このことは、理論上、これらの積分法が任意の大きな時間刻みを採用することができることを意味する。このような方法は、「Ｌ安定性」と呼称される数学的な特性を有する。所謂「陰的」な積分法のみが、Ｌ安定性であり得るが、ほとんどの陰的積分法が、現実的に「Ｌ安定性」ではない。上で引用した同時係属出願である、「ＭＥＴＨＯＤ ＦＯＲ ＬＡＲＧＥ ＴＩＭＥＳＴＥＰＳ ＩＮ ＭＯＬＥＣＵＬＡＲ ＭＯＤＥＬＩＮＧ」と題する、米国特許出願番号＿＿＿＿は、陰的であり、特にＬ安定性である積分法の使用を取り扱う。

【００１０】

本発明は、陰的積分器（特にＬ安定性積分器）が大きな時間刻みをとることを可能にするための別の重要な局面、すなわち、「ヤコビアン」と呼称される陰的積分法に必要とされる成分を計算するためのより正確な方法を取り扱う。

【００１１】

しかし、ＭＤシミュレーションに対する高振動数の分子振動の同じ問題が、ヤコビアンの計算についても問題を生じる。例えば、２つの原子の電子軌道関数が重なるために生成される斥力性のファンデルワールス力は、分子内で説明されなければならない。この量子力学的効果は、しばしば、所謂、Ｌｅｎｎａｒｄ－Ｊｏｎｅｓポテンシャル（Ｒａｐａｐｏｒｔ，ｏｐ．ｃｉｔ．）によって、近似され、このポテンシャルは、この斥力を、$1/r^{13}$に比例するものとしてモデリングする。ここで、ｒは、原子の中心間距離である。これは、非常にスティッフ（ｓｔｉｆｆ）である原子間力であり、これは、分子動力学（ＭＤ）シミュレーションの特徴であり、そして、これは、シミュレーションの間に、時間を前進させるために使用される任意の数値積分スキームについて特別な困難を引き起こす。結果として、かつ、上述したように、殆どの従来技術のＭＤ積分スキームは、これらの非常にスティッフな斥力により生成される高振動数によって限定される時間刻みを有する。そして、特に、この原子間力のスティッフ性は、特定の必要とされるヤコビ行列を形成する困難を顕著に増強する。このようなヤコビ行列は、直上で引用した同時係属中の出願に記載されるように、任意の安定性陰的積分スキームの必要不可欠な構成要素である。

【００１２】

全ての汎用目的のシミュレーションコードは、以下のようにヤコビアンを数値的に計算するルーチンを提供する。

【００１３】

【数１】

$$\dot{y} = f(y,t)$$

を積分するため所与の式について、所望のヤコビアンは、

【００１４】

【数２】

$$J = \partial f / \partial y$$

であり、以下のように数値的計算される：

【００１５】

【数３】

$$J = \frac{\Delta f}{\Delta y}$$

ここで

$$\Delta f = f|_{y=y_2} - f|_{y=y_1}$$
$$\Delta y = y_2 - y_1$$

。

【００１６】

ここで、摂動Δｙは、十分な結果を与えるように選択される必要があり、そして特にスティッフ（ｓｔｉｆｆ）な系について選択することがで有り得ることに留意のこと。対照的に、解析的ヤコビアンは、直接的、すなわち、この場合アルゴリズム的に、所望の導関数の方程式について解くことによって、計算される。

【００１７】

線形化されたモデルは、通常は、単純な系に対して解析的に生成される。このような線形化は、通常、平衡解の周囲で実行される。ＡＣＳＬ（Ａｄｖａｎｃｅｄ Ｃｏｎｔｉｎｕｏｕｓ Ｓｉｍｕｌａｔｉｏｎ Ｌａｎｇｕａｇｅ）（ＡＣＳＬ Ｒｅｆｅｒｅｎｃｅ Ｇｕｉｄｅ, Ｍｉｔｃｈｅｌｌ Ｇａｕｔｈｉｅｒ ａｎｄ Ａｓｓｏｃｉａｔｅｓ, １９９６）およびＳＰＩＣＥ（サーキットシミュレーションパッケージ）（Ｒ．Ｋｉｅｌｋｏｗｓｋｉ，Ｉｎｓｉｄｅ ＳＰＩＣＥ，ＭｃＧｒａｗ－Ｈｉｌｌ，１９８８）のようなパッケージにおいて、線形化の前に平衡解析を行うことは、一般的である。このような線形化は、数値的に実行される。

【００１８】

対照的に、本発明のヤコビアンは、微分方程式の瞬間の解（非平衡）をについての線形化を表現し、そして、解析的に生成される。解析的ヤコビアンを生成するための別の従来技術のアプローチは、任意の方程式を記号的に微分し得るＡＤＩＦＯＲ（Ａｕｔｏｍａｔｉｃ Ｄｉｆｆｅｒｅｎｔｉａｔｉｏｎ ｏｆ Ｆｏｒｔｒａｎ）（Ｃ．Ｂｉｓｃｈｏｆら、ＡＤＩＦＯＲ ２．０ Ｕｓｅｒｓ’Ｇｕｉｄｅ Ａｒｇｏｎｎｅ Ｎａｔｉｏｎａｌ Ｌａｂｏｒａｔｏｒｙ，１９９８）のような自動化微分ツールを使用することであることに留意するべきである。これらのツールは、実際問題として、本発明をインプリメントするために使用され得る。しかし、この方程式の構造は、その計算を最小化し、丸めおよび項の相殺に起因する誤差を避け、そして、解かれる問題のサイズを限定し得る「方程式スウェル（ｅｑｕａｔｉｏｎ ｓｗｅｌｌ）」を回避するように適切に活用されなければならない。

【００１９】

むしろ、本発明は、分子モデルの運動方程式の有効な陰的積分法（Ｌ安定性積分器を含む）についての解析的ヤコビアンの計算を可能にする。

【００２０】

（発明の要旨）

本発明は、分子の挙動をモデリングする方法を提供する。この方法は、その分子について、ねじれ角、剛体多体系モデルを選択する工程であって、このモデルが、運動方程式を有する工程；陰的積分器を選択する工程；および解析的ヤコビアンを生成して、陰的積分器が運動方程式を積分して分子の挙動の計算を得る、工程を、有する。

解析的ヤコビアンＪは、運動方程式の残差形式の解析的ヤコビアンから導出され、そして

、これは、以下：

【００２１】

【数４】

$$J=\begin{pmatrix}\frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial u}\\ \frac{\partial \dot{u}}{\partial q} & \frac{\partial \dot{u}}{\partial u}\end{pmatrix}\triangleq\begin{pmatrix}J_{qq} & J_{qu}\\ J_{uq} & J_{uu}\end{pmatrix};\qquad \text{および}$$

$$J_{qq}=\frac{\partial \dot{q}}{\partial q}=\frac{\partial (Wu)}{\partial q}\ \text{および}\quad J_{qu}=\frac{\partial \dot{q}}{\partial u}=W$$

$$J_{uq}=\frac{\partial \dot{u}}{\partial q}=-M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial q}\ \text{および}\ J_{uu}=\frac{\partial \dot{u}}{\partial u}=-M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial u}$$

のように記述される。ここで、ｑは一般化座標、ｕは一般化速度、Ｗはジョイントマップ行列であり、そしてＭは質量行列であり、そしてρｕは、上記運動方程式の動力学的残差であり、そしてｚは、－Ｍ－１ρｕ（ｑ、ｕ、０）である。この方法はまた、ねじれ角、剛体多体系によってモデリングされ得る任意の物理系について使用され得る。

【００２２】

本発明はまた、ねじれ角、剛体多体系によってモデリングされ得る、分子の挙動、または任意の物理系をシミュレートするための、コンピュータコードを提供する。陰的積分器を使用する、コンピュータコードにおけるモジュールは、上記のような解析的ヤコビアンを含む。

【００２３】

（特定の実施形態の説明）

（本発明の一般的な説明）

モデリングした分子系のシミュレーションにおいて時間を前進させるために使用される数値的方法は、積分器、または積分法と呼ばれる。この積分は、この分子系の支配的な運動方程式を差分化することによって、前進する。陰的積分器の場合、この工程は、結合型非線形代数方程式（「ステージ（ｓｔａｇｅ）」方程式）の集合を生じ、そして、これらの方程式の解は、次の時間刻みにおけるこの分子系の状態ベクトルを生成する。

【００２４】

本発明は、このステージ方程式の解法を支援する。原子間力が、短距離にわたって非常に迅速に変動するので、原子間のトラジェクトリーを正確に伝達することが難しい。原子の位置のわずかな誤差が、このステージ方程式を満たす際に大きな誤差を生じる。このヤコビアンがこのステージ方程式を反復的に解くために使用されるので、不正確なヤコビアンが正確な解からかけ離れた試験的な解（ｔｒｉａｌ　ｓｏｌｕｔｉｏｎ）を生じる。このことによって、この試験的な解の撤回に至り、そしてこのシミュレーションを妨げる。

【００２５】

数値的ヤコビアンは、その有効数字の半分のみで正確であり得る。解析的ヤコビアンは、しばしば、その最後の桁以外の全てにおいて正確であり得る。この結果の利点は、この積分器が、特定の間隔のシミュレーションを実施するためにずっと少ない時間刻みを取ることが可能であり、この積分法の理論的な安定性の完全な享受が可能となる。

【００２６】

このヤコビアンを生成することの容易性または困難性は、この支配的な方程式を生成するために使用される定式化に決定的に依存する。例えば、グローバル（ｇｌｏｂａｌ）デカルト定式化は、非常に限定された陽的座標依存性を有する方程式を生じる。このような定式化に対する解析的ヤコビアンの生成は周知である。他方で、内部座標（これにおいて、各分子サブユニットの位置は、それより前のサブユニットの位置に対して表現される）を

独立変数として使用することは、大きな利点を有する。この方法は、内部座標の任意の選択でモデリングされた任意の分子について最も価値があり、そして、特に、残基間の「ねじれ角」を内部座標として用いる、タンパク質のモデルまたは他の高分子とともに使用される場合に、価値がある。ねじれ角で定式化された系についての解析的ヤコビアン生成するためのアルゴリズムは、考案するのが極めて困難である。しかし、本発明は、この課題を達成する。

【００２７】

本発明は、以下のような、一見、多項式アルゴリズムが与えられない問題に取り組む：内部座標、および特にねじれ角を使用する定式化のための解析的ヤコビアンの生成（これは、一般的に非実用的であると考えられている）。数値的ヤコビアンよりも正確であることに加え、この解析的ヤコビアンはまた、生成されることが（コンピュータパワーにおいて）安価である。本発明はまた、状態微分のヤコビアンが、計算トルク法（ｃｏｍｐｕｔｅｄ　ｔｏｒｑｕｅ　ｍｅｔｈｏｄ）のヤコビアンに逆行列を適用することによって計算されるという重要な結果を認識する。この結果は、コンピュータ処理時間およびアルゴリズムを構築する労力の有意な節約となる。

【００２８】

さらに、ねじれ角、内部座標を使用する多体系定式化についての解析的ヤコビアンを生成するこの方法は、一般的なＭＢＳの文献においてこれまで見出されていない。本発明は、任意のねじれ角ＭＢＳ定式化のために使用され得、このねじれ角ＭＢＳは、以下を含むが必ずしもこれらに限定されない、分子シミュレーション以外の他の多数の学問分野にも適用され得る：機械システム、ロボットシステム、乗物システム、またはヒンジ接続剛体のセットとして記述され得る他の任意のシステム。

【００２９】

（説明の概観）

好ましい実施形態は、いくつかの節に分割される。節の第１のセットは、ＭＤシミュレーションアーキテクチャ、多体系（ＭＢＳ）の定義、およびＭＢＳ方程式の残差形式を、引き続き説明のために記載する。節の次のセットは、ヤコビアンの定義、陰的積分法におけるヤコビアンの役割、および残差形式を用いた解析的ヤコビアンの計算法を、議論する。数値的ヤコビアンに対する解析的ヤコビアンの優れた精度および能力もまた示される。解析的ヤコビアンの計算法におけるさらなる有効性が議論され、特に、剛体ＭＢＳをこのヤコビ行列のサイズを「減縮」するために利用すること、およびこのＭＢＳのトポロジー構造を利用して不必要な計算を取り除くことを議論する。

【００３０】

常微分方程式（ＯＤＥ）を解くために、従来技術のほとんどは、直接形式、すなわち、以下：

【００３１】

【数５】

$$\dot{y} = f(y,t)$$

（ここで、

【００３２】

【数６】

$$\dot{y}$$

は、

【００３３】

【数７】

$$\frac{dy}{dt}$$

を意味する）
で表現された方程式を使用してきた。生体分子系についての運動方程式は、この形式（そして、これは直接形式と呼ばれる）に投入され得る。分子モデリングにおいて、本発明に対する全ての先行技術は、この直接形式を使用してきた。すなわち、
【００３４】
【数８】

$$\dot{q} = Wu,\ \dot{u} = M^{-1} f$$

であり、ここで、ｑおよびｕは、それぞれ、一般化座標および一般化速度であり、その結果、従来のＯＤＥの解法が適用され得る。しかし、これは、次数（Ｎ）～次数（N^3）の次数（オーダー）の浮動小数点演算（これは、使用されるアルゴリズムに依存し、ここで、Ｎはこの系の自由度である）を費やすＭ（この系の質量を表す）の逆行列を必要とする。なぜならば、この方程式の固有の形式が、この一般化速度の微分の間の慣性カップリング（ｉｎｅｒｔｉａｌ　ｃｏｕｐｌｉｎｇ）を生じるためである。すなわち、この方程式は、以下：
【００３５】
【数９】

$$\dot{q} - Wu = 0,\ M\dot{u} - f = 0$$

の形式で、固有に生成され、ここで、質量行列Ｍは、一般化座標ｑに陽的に依存しており、すなわち、Ｍ＝Ｍ（ｑ）である。この事実は、この状態微分が、経時的に積分器によって必要とされる各時刻での質量行列を形成し、そして効率的にこの質量行列を因数分解することを要求している。
【００３６】
一般化ジョイントマップ行列（ｇｅｎｅｒａｌｉｚｅｄ　ｊｏｉｎｔ　ｍａｐ　ｍａｔｒｉｘ）Ｗは、ブロック対角であり、そしてこれはまた、座標に依存する（つまり、Ｗ＝Ｗ（ｑ）である）が、これは重大な計算的な損失を有さない。
【００３７】
本発明に従って、分子系の方程式を解くための解法は、残差形式で表現され、直接的にこの状態微分を生成する慣例的な工程を回避する。この残差形式は、以下の工程を有する：１）この解法の変数の差分化（ｄｉｓｃｒｅｔｉｚａｔｉｏｎ）。この差分化の特別な形式が、時間において分子モデルを前進させるために使用される特定の陰的積分法によって、決定づけられる。陰的積分法は、この残差形式に続く。陰的積分法、特に、Ｌ安定性積分器、および他の高度に安定性である積分器（例えば、陰的Ｅｕｌｅｒ法、Ｒａｄａｕ５、ＳＤＩＲＫ３、ＳＤＩＲＫ４、他の陰的Ｒｕｎｇｅ－Ｋｕｔｔａ法およびＤＡＳＳＬ）または他の陰的多段法がまた、分子モデリングについての他の利点を提供する。例えば、上記に引用された、同一日付けで出願された、「ＭＥＴＨＯＤ　ＦＯＲ　ＬＡＲＧＥ　ＴＩＭＥＳＴＥＰＳ　ＩＮ　ＭＯＬＥＣＵＬＡＲ　ＭＯＤＥＬＩＮＧ」と題する、米国特許出願番号＿＿＿＿＿を参照のこと。具体的な単純な例として、陰的Ｅｕｌｅｒ積分法を使用する場合、この差分化は、以下に従う：
【００３８】
【数１０】

$$\dot{q} = (q_n - q_{n-1})/h,\ \dot{u} = (u_n - u_{n-1})/h$$

ここで、ｈはその時間刻みである。
【００３９】
２）この残差方程式への置換：
【００４０】
【数１１】

$$\begin{pmatrix}\rho_q\\ \rho_u\end{pmatrix}=\begin{pmatrix}\dot{q}-W(q)u\\ M(q)\dot{u}-f(t,q,u)\end{pmatrix}$$

３）q_nおよびu_nについてのこの得られた非線形代数方程式
【００４１】
【数１２】

$$\begin{pmatrix}\rho_q\\ \rho_u\end{pmatrix}=0$$

の解法。
【００４２】
運動学的残差ρ_qは、陰的積分器によって生成された推定
【００４３】
【数１３】

$$\dot{q}$$

を、以下で詳説されるようなこの分子モデルのジョイントを決定するためのルーチンによって計算された微分と比較する。この残差の第２行は、動力学的残差ρ_uであり、これは、推定された
【００４４】
【数１４】

$$\dot{u}$$

が運動方程式を満たす度合いを決定する。
【００４５】
系の質量行列Ｍおよびいわゆる「無バイアスヒンジトルク（ｂｉａｓ－ｆｒｅｅ　ｈｉｎｇｅ　ｔｏｒｑｕｅ）」ｆは、両方とも状態依存性である。無バイアスヒンジトルクは、残差ルーチンへと運ばれる計算された
【００４６】
【数１５】

$$\dot{u}$$

ベクトルが零（０）である場合に、動力学的残差によって生成される。一般的に、このヒンジ加速度は、適用された力、ジョイントトルク、および運動誘導性効果（例えば、Ｃｏｒｉｏｌｉｓ力、および遠心力）に応答するものである。この系が静止しており、かつジョイントトルクにのみ供されている場合、無バイアス状態にあると考えられる。現実的な入力を伴う実際の系は、バイアスのある入力と等価なジョイントトルクの集合を計算することによって無バイアス状態へと換算され得る。両方の集合は、同じヒンジ加速度を生じる。
【００４７】
残差形式の好ましい実施形態は、その分子モデルのための次数（Ｎ）ねじれ角、剛体について示される。以下の節によって、基礎的な定義から分子モデルが展開され、そして、このモデルの運動を計算するためにこのモデルがどのように使用しされるかを示す。第１に、分子モデリングシミュレーションのための、全コンピュータコードアーキテクチャが、記載される。次いで、次数（Ｎ）ねじれ角、剛体多体系が、使用された記数法、基準配置、物体間のジョイントの定義、一般化座標、および一般化速度に従って、導出される。動力学状態についてのこのアプローチは、Ｔ．Ｒ．Ｋａｎｅ（Ｄｙｎａｍｉｃｓ、第３版、１９７８）によって使用されたアプローチと類似している。
【００４８】

最後に、この残差形式の結果を使用した解析的ヤコビアンの好ましい実施形態が開発された。

【００４９】

（分子動力学シミュレーションアーキテクチャ）

本発明に従うモデリングモジュールのための、ソフトウェアの汎用システムアーキテクチャ４８およびそのいくつかのプロセスを、図１に示す。大きい矩形ブロックの各々は、ソフトウエアモジュールを表し、矢印はソフトウエアモジュール間を通過する情報を表す。このソフトウエアシステムアーキテクチャは、モデラー（ｍｏｄｅｌｅｒ）モジュール５０、生化学的コンポーネントモジュール５２、物理学的モデルモジュール５４、解析モジュール５６、および可視化モジュール５８を有する。これらのモジュールのいくつかの詳細は、以下に説明される。他のモジュールが、公で利用可能である。

【００５０】

モデラーモジュール５０は、ユーザが特定の分子系を規定する物理学的パラメータを入力するインターフェースを提供する。インターフェースはグラフィックまたはデータファイル入力（またはその両方）を有し得る。生物学的コンポーネントモジュール５２は、分子系の特定の数学モデルのためのモデラー入力を変換し、モデリングされる系の、分子、力場、および溶媒のそれぞれを、数学的モデリングするための、変換サブモジュール６０、６２、および６４に、分割される。例えば、Ｔｉｎｋｅｒ（Ｊａｙ Ｐｏｎｄｅｒ，「ＴＩＮＫＥＲ Ｕｓｅｒ’ｓ Ｇｕｉｄｅ」，Ｖｅｒｓｉｏｎ３．８，２０００年１０月，Ｗａｓｈｉｎｇｔｏｎ Ｕｎｉｖｅｒｓｉｔｙ，Ｓｔ．Ｌｏｕｉｓ，ＭＯ）を含む、利用可能ないくつかのモデラーおよび生物学的コンポーネントモジュールが存在する。

【００５１】

生物学的コンポーネントモジュール５２から変換された物理パラメータを用いて、物理学的モデルモジュール５４は、分子系を数学的に定義する。多体系サブモジュール６６が、モジュール５４の中心に存在する。物理モデルモジュール５４および可視化モジュール５８と通信する解析モジュール５６は、物理学的モデルモジュール５４によって規定された分子系の計算モデルに解を提供する。解析モジュール５６は、物理学的モデルモジュール５４の異なる方程式を積分する、積分器サブモジュール６８のセットからなる。積分器サブモジュール６８は、経時的に分子系を前進させ、さらに分子系の極小エネルギー配置を決定する際に使用される静的解析を提供する。解析モジュール５６および積分器サブモジュールジュール６８は、本発明の内容のほとんどを含み、以下で詳細に説明される。

【００５２】

可視化モジュール５８は、生物学的コンポーネントモジュール５２および解析モジュール５６からの入力情報を受信し、ユーザに分子系の三次元画像表示および分子系に対して得られた解を提供する。多くの可視化モジュールが現在利用可能であり、例えば、ＶＭＤ（Ａ．Ｄａｌｋeら、「ＶＭＤ Ｕｓｅｒ’ｓ Ｇｕｉｄｅ」，Ｖｅｒｓｉｏｎ１．５，２０００年６月、Ｔｈｅｏｒｅｔｉｃａｌ Ｂｉｏｐｈｙｓｉｃｓ Ｇｒｏｕｐ，Ｕｎｉｖｅｒｓｉｔｙ ｏｆ Ｉｌｌｉｎｏｉｓ，Ｕｒｂａｎａ，Ｉｌｌｉｎｏｉｓ）である。

【００５３】

記載されたソフトウェアコードは、Ｐｅｎｔｉｕｍ（登録商標）ＩＩＩマイクロプロセッサおよびＰｅｎｔｉｕｍ（登録商標）ＩＶマイクロプロセッサ（これえらは、Ｉｎｔｅｌ Ｃｏｒｐｏｒａｔｉｏｎ（Ｓｎｔａ Ｃｌａｒａ、Ｃａｌｉｆｏｎｉａ）によって製造される）を搭載したＰＣのような、従来のパーソナルコンピュータにて実行される。このことは、計算を実行させるためにスーパーコンピュータを使用する、分子モデリングにおける多くの現在の努力と対照をなす。糖残、さらなるスピードの改善が、より速いコンピュータ上で記載したソフトウェアを実行させることによって、得られる。

【００５４】

（分子モデルおよび多体系の説明）

サブモジュール６８における、以下で説明される積分器は、多体系（ｍｕｌｔｉｂｏｄｙ ｓｙｓｔｅｍ；ＭＢＳ）に関する分子モデルの運動を記述する方程式のセットについて

演算する。以下に詳細に説明された積分法の計算を支援するために、本発明に従って、ねじれ角、剛体モデルが、対象とする分子系を記述するために使用される。内部座標（選択された、一般化座標および一般化速度）が分子の状態を記述するために使用され得る。

【００５５】

このＭＢＳは、モデリングされるされる分子系を構成する、原子および有効な剛体性の結合の抽象化であり、かつ、シミュレーションによって扱われる問題についての重要な特徴を失うことなく、実際の物理学系、その環境下にある分子を単純化するように選択される。図１に示された一般的なシステムアーキテクチャに関して、このＭＢＳは、原子間の静電荷または他のエネルギー的相互作用を含まないだけでなく、この分子を囲む溶媒のモデルも含まない。力場は、生化学的コンポーネントのサブモジュール６２でモデリングされ、そして、溶媒は、生化学的コンポーネントのサブモジュール６４でモデリングされる。

【００５６】

図２は、対象分子のＭＢＳのツリー構造を示す。このＭＢＳの基本的な抽象化は、ヒンジ接続された剛体１７０の１つ以上の集合の抽象化である。剛体は、その物体を構成する全ての粒子が互いに対して固定された位置を有する物理的な物体の数学的抽象化である。伸縮または他の相対運動は許容されない。ヒンジ接続は、２つの剛体間の許容される相対運動を規定する数学的抽象化である。これらの剛体およびヒンジ接続の例は、以下に説明される。

【００５７】

基本物体（ｂａｓｅ　ｂｏｄｙ）１７２と呼ばれる１以上の物体は、運動力学的状態が、グラウンド１７４上の基準点に対して直接参照される点において固有の状態を有する。系のグラフは、１つ以上の「ツリー」である。ツリーの重要な特性は、任意の物体から任意の他の物体への経路が固有であり、すなわち、このグラフはループを含まないということである。ツリー内の物体は、ｎ個存在する（この出発点（ｂａｓｅ）は、標識１を有する）。ツリー内の物体は、規則的な標識が割り当てられ、このことによって、これらの標識が、基本物体から任意のリーフ物体（ｌｅａｆ　ｂｏｄｙ）１７６までの任意の経路上で決して減少しないことが意味される。リーフ物体は、単一の他の物体のみに接続される物体である。規則的な標識は、標識ｎをリーフ物体１７８の１つ（少なくとも１つでなければならない）に割り当てることによって達成され得る。この物体がグラフから除去される場合、その場合のツリーは、ｎ－１物体を有する。次いで、標識ｎ－１は、そのリーフ物体１８０のうちの１つに割り当てられ、全ての物体が標識化されるまで、このプロセスが繰り返される。これはまた、系内の任意の残っているツリーに対して行われる。

【００５８】

物体間の関係を維持することを支援するために、整数型の関数が使用され、系の各物体について内側物体（ｉｎｂｏａｒｄ　ｂｏｄｙ）を記録する。各出発点（ｂａｓｅ）対しては、内側物体が基準（ｇｒｏｕｎｄ）であり、物体ｋ（１８４）についてのペアレント物体すなわち内側物体１８２（ｉ）は、ｉ＝ｉｎｂ（ｋ）として参照される。さらに、記号Ｎは、慣性系、すなわち基準系１７４をいう。上付き文字Ｏは、座標原点（０，０，０）をいう。

【００５９】

ある点から別の点までのベクトルについての記号は２点の名前を含む。従って、r^{PQ}は、点Ｐから点Ｑまでのベクトルである。基準系内の点の速度を表すベクトルは、以下のように点および基準系の名称を含む：$^{N}v^{P}$。以後導入される特定の記号は、２つの基準系を関連付ける。この場合では、記号は２つの系の名前を含む。従って、$^{i}C^{k}$は、系ｉにおける系ｋの配向のための方向余弦行列である。この記号はそのペアレント系における代表的物体のための方向余弦行列を指す。従って、$^{i}C^{k}$（ｊ）は、問題にしている現実の物体ｊを示す。左および右の上付き文字は、物体のインデックスを変更しない。これはまた、他の記号についても真である。アスタリスクは、転置を示す（例えばH^{*}（ｋ））。ベクトルの上のチルダ記号は、３×３歪対称クロス乗積、すなわち

【００６０】

【数１６】

$$\tilde{v}w \triangleq v \times w$$

を表す。

【００６１】

【数１７】

$$\underline{\underline{E}}_i$$

は、ｉ×ｉの恒等行列であり、そして

【００６２】

【数１８】

$$\underline{0}_i$$

は、長さｉの零ベクトルであり、

【００６３】

【数１９】

$$\underline{\underline{0}}_i$$

は、ｉ×ｉの零行列である。

【００６４】

（モデルの剛体）

図３は、ＭＢＳのサンプル「ツリー」の基準配置１９０を示す。１つを超えるツリーが許容される。各物体の点は、そのヒンジポイントＱとして表される。例えば、点Q_k　１８６は、物体ｋ　１８４のヒンジポイントである。座標軸の固定された集合は、慣性系１９８に確立される。ＭＢＳの任意の配置は、その基準配置１９０として選択される。この配置における慣性座標軸の写像は、各物体内で物体に固定された軸の集合を確立するために使用される。基準配置では、各ヒンジポイントＱはＰ（そのペアレント物体（または拡張された物体）の点）と一致する。各物体に対して、点Ｐは、物体の内側ヒンジポイントと呼ばれる。そのため、物体ｋ　１８４についての内側ヒンジポイント　P_k　１８８は、ペアレント物体ｉ　１８２に固定された点である。各基本物体の内側ヒンジポイントは、グラウンドに固定された点Ｏ　１９２である。図２に示された拡大図は、点Q_k　１８６が、本体ｋ　１８４に固定され、点P_k　１８８はペアレント物体ｉ　１８２に固定されることをより明確に示す。

【００６５】

ヒンジポイントの位置は、各物体についての定ベクトルである

【００６６】

【数２０】

$$\mathbf{d}(k)$$

１９４を規定し、これはまた

【００６７】

【数２１】

$$r^{Q_i P_k}$$

と記載され得る。物体ｋについてのベクトルは、そのペアレント物体ｉに固定される。このベクトルは、物体ｉのヒンジポイントから物体ｋの内側ヒンジポイントの範囲にある。ベクトル

【００６８】

【数２２】

$\mathbf{d}(1)$

１９６は、慣性系原点から第１の基本体の内側ヒンジポイント（またグラウンド内に固定された点）の範囲にあり、
【００６９】
【数２３】
r^{OQ_1}

と記載され得る。
【００７０】
物体に対して、ｍ（ｋ）、
【００７１】
【数２４】
$\mathbf{p}(k)$, および $\underline{\underline{\mathbf{I}}}_{Q_k}(k)$

は、そのヒンジ点Q_kに対する物体ｋの質量の特性を規定する。これらはそれぞれ、質量、第１質量モーメント、その物体の座標系にある、そのヒンジポイントについての物体の慣性行列である。粒子の分布を構成する剛体について、その質量特性は、予備処理モジュールによって計算される定数である。それらの計算の詳細は、ＫａｎｅによるＴ．Ｒ．Ｄｙｎａｍｉｃｓ，第３版、１９７８年１月、Ｓｔａｎｆｏｒｄ Ｕｎｉｖｅｒｓｉｔｙ，Ｓｔａｎｆｏｒｄ，ＣＡ等の標準的な参考文献において見出され得る。
【００７２】
ヒンジポイントQ_kに対する物体ｋの空間的な慣性Ｍ（ｋ）は、対称６×６行列
【００７３】
【数２５】

$$M(k)=\begin{bmatrix}\underline{\underline{\mathbf{I}}}_{Q_k}(k) & \tilde{\mathbf{p}}(k)\\ -\tilde{\mathbf{p}}(k) & m(k)\underline{\underline{E}}_3\end{bmatrix}$$

によって与えられる。
【００７４】
この系における各ジョイントは幾何学的データによって説明される。例えば、ピンジョイントは、そのジョイントによって接続された２つの物体に固定された軸によって特徴付けられる。ジョイントついての具体的なデータは、その型に依存する。番号ｎ、ｉｎｂ関数、系の質量特性、ベクトル
【００７５】
【数２６】
$\mathbf{d}(k)$

、およびジョイント幾何学データ（ジョイント型を含む）は系パラメータを構成する。
【００７６】
（モデルのジョイントおよび一般化座標）
図４Ａ～図４Ｃは、ＭＢＳの好適な実施形態のジョイントの定義（スライダージョイント１００、ピンジョイント１０２、およびボールジョイント１０４）を示す。各ジョイントは、内側ヒンジポイントP_k　１０８に関してヒンジポイントQ_k　１０６の並進変位または回転変位を可能にする。これらの変位は、物体ｋについての一般化座標ｑ（ｋ）１１０によりパラメータ表示される。ちなみに、一般化座標は一般化量の一例であり、回転特性および並進特性の両方を有する量を指すことに留意されるべきである。例えば、点において作用する一般化力は、力ベクトルおよびトルクベクトルの両方からなる。スライダージョイント１００についての一般化座標ｑ（ｋ）は滑り変位ｘ　１１２である。ピンジョ

イント１０２についての一般化座標ｑ（ｋ）は、角変位θ　１１４である。ボールジョイント１０４についての一般化座標ｑ（ｋ）は、Ｅｕｌｅｒパラメータ（ε_1，ε_2，ε_3，ε_4）１１６である。

【００７７】

ジョイントの各々は、ピン、スライダー、またはボールジョイント；あるいはこれらのジョイントの組み合わせであり得る。フリージョイント、Ｕジョイント、シリンダージョイント、およびベアリングジョイントを含むがこれらに限定されない、多くの他のジョイント型が可能である。例えば、基本物体内側ヒンジポイントから基本物体ヒンジポイントまでのベクトルの慣性測定数、ｑ（ｋ）＝（ｘ，ｙ，ｚ）は、３つの直交スライダージョイントとして、グラウンドにおける基本物体の変位を表現する。フリージョイントは、ボールジョイントと結合された３つの直交スライダージョイントからなり、全部で６つの自由度を有する。

【００７８】

全ての物体に対する一般化座標の集合は、系の一般化座標であるベクトル

【００７９】

【数２７】

$$\mathbf{q}$$

を含む。

【００８０】

特定のジョイントについての一般化座標が与えられると、以下の２つの量：

ジョイント並進ベクトル

【００８１】

【数２８】

$$r^{P_k Q_k}(k)$$

およびペアレントにおける本体ｋに対する方向余弦行列

${}^{i}C^{k}$（ｋ）

が形成される。

この並進ベクトル

【００８２】

【数２９】

$$r^{P_k Q_k}(k)$$

は、ペアレント物体の座標系における物体ｋの内側ヒンジポイントＰから物体ｋのヒンジポイントＱへのベクトルを表す。これらの計算の詳細は、ジョイント型に依存し、容易に導出される。この記述の目的のために、系の一般化座標で与えられた

【００８３】

【数３０】

$$r^{P_k Q_k}(k)$$

および${}^{i}C^{k}$（ｋ）を生成する関数の呼び出しが、前提となる。

【００８４】

導入されたように、各物体についてのヒンジポイントの選択は任意である。しかし、賢明な選択によって、問題が大幅に簡略化される。例えば、ピンジョイントに対して、このヒンジポイントは、ジョイントの軸上にある点として選択されるべきである。この選択に対して、点Ｐおよび点Ｑは、ジョイント角度の全ての値に対して一致したままであるため、このジョイントの並進は０である。点Ｑがこの軸からの離れて選択される場合、点ＰおよびＱは、以下：

【００８５】

【数３１】

$$r^{P_kQ_k}(k) = \lambda \times r^{OQ_k} \sin\theta - (1-\cos\theta)\left(\underline{\underline{E}}_3 - \lambda\lambda^*\right) r^{OQ_k}$$

のように互いに対して移動し、ここで、

【００８６】

【数３２】

$$\lambda$$

はジョイント軸の単位ベクトルであり、θはそのジョイント角度であり、

【００８７】

【数３３】

$$r^{OQ_k}$$

は、軸上の任意の点から点Ｑへのベクトルである。

【００８８】

ピンジョイントおよびボールジョイントについて、軸上の点がヒンジポイントとして常に選択される。これらのジョイントに対して、並進ベクトル

【００８９】

【数３４】

$$r^{P_kQ_k}(k)$$

は０である。

【００９０】

スライダージョイントについて、並進ベクトル

【００９１】

【数３５】

$$r^{P_kQ_k}(k)$$

は

【００９２】

【数３６】

$$q(k)\lambda .$$

である。

【００９３】

ピンに対する方向余弦行列は、

【００９４】

【数３７】

$$^{i}C^{k}(k) = \underline{\underline{E}}_3 \cos\theta + \tilde{\lambda}\sin\theta + \lambda\lambda^*(1-\cos\theta) .$$

である。スライダーに対する方向余弦行列は、

【００９５】

【数３８】

$$\underline{\underline{E}}_3$$

である。

【００９６】

（モデルの一般化速度）

ペアレントｉにおいて測定された物体ｋのヒンジポイントの一般化速度を、一般化速度の集合ｕ（ｋ）によってパラメータ表示された${}^{i}\mathrm{V}^{k}$（ｋ）とすると

【００９７】

【数３９】

$$ {}^{i}V^{k}(k)=\begin{pmatrix} {}^{i}\omega^{k}(k) \\ {}^{i}v^{o_k}(k) \end{pmatrix}=H^{*}(k)u(k) $$

であり、

ここで、行列Ｈ（ｋ）は、このジョイントに対してジョイントマップと呼ばれる。これは６×n_u（ｋ）の行列であり、ここで、n_u（ｋ）は、そのジョイントに対する自由度の数（ピンまたはスライダに対して１、ボールに対して３、フリージョイントに対して６）である。Ｈ（ｋ）は、一般的には座標ｑに対する依存性を有し得る。ジョイントについての一般化速度が与えられると、ジョイントマップは、チャイルド（子）物体の系において表現されたジョイントの線形速度および角速度を生成する。これらのジョイントとして、本発明者らは、

【００９８】

【数４０】

$$ H(k)=\begin{bmatrix} \underline{\lambda} & 0 & 0 & 0 \end{bmatrix},\ \text{ピン} $$

$$ H(k)=\begin{bmatrix} 0 & 0 & 0 & \underline{\lambda} \end{bmatrix},\ \text{スライダー} $$

$$ H(k)=\begin{bmatrix} \underline{\underline{E}}_3 & \underline{\underline{0}}_3 \end{bmatrix},\ \text{ボール} $$

$$ H(k)=\begin{bmatrix} \underline{\underline{E}}_3 & \underline{\underline{0}}_3 \\ \underline{\underline{0}}_3 & {}^{i}C^{k}(k) \end{bmatrix},\ \text{フリー} $$

を使用する。

この全ての物体についての一般化速度の集合は、ベクトル

【００９９】

【数４１】

$$ \boldsymbol{u} $$

、すなわち、この系についての一般化座標を含む。前述のように、（ｑ，ｕ）および特定のジョイント型が与えられた場合、ベクトル${}^{i}\mathrm{V}^{k}$（ｋ）を生成し得る関数の呼び出しが前提となる。以下の微分：

【０１００】

【数４２】

$$ \dot{q}(k)=\dot{q}(q(k),u(k)) $$

を計算し得る関数の呼び出しがまた、前提となる。このルーチンは、以下の一般化位置座標の時間微分：

【０１０１】

【数４３】

$$ \dot{q}=W(q)u $$

を生成し、ここで、Ｗ（ｑ）は、

【０１０２】

【数４４】

$$ \dot{q} $$

およびｕに関連するブロック対角行列であり、以下のジョイント型：
【０１０３】
【数４５】
ピンジョイント、スライダージョイントについて $\dot{q} = u$

ボールジョイントについて
$$\begin{bmatrix} \dot{\varepsilon}_1 \\ \dot{\varepsilon}_2 \\ \dot{\varepsilon}_3 \\ \dot{\varepsilon}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \varepsilon_4 & -\varepsilon_3 & \varepsilon_2 \\ \varepsilon_3 & \varepsilon_4 & -\varepsilon_1 \\ -\varepsilon_2 & \varepsilon_1 & \varepsilon_4 \\ -\varepsilon_1 & -\varepsilon_2 & \varepsilon_4 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

ここで、$q = [\varepsilon_1 \quad \varepsilon_2 \quad \varepsilon_3 \quad \varepsilon_4]^*$ および $u = [\omega_1 \quad \omega_2 \quad \omega_3]^*$

およびフリーのジョイントに依存するブロックの各々は、３つのスライダージョイントおよび１つのボールジョントの組み合わせである。ボールジョイントについて、３つのｕと関連する４つの
【０１０４】
【数４６】

$\dot{q}$

（Ｅｕｌｅｒパラメーターの微分）が存在することに留意のこと。
【０１０５】
同様に、${}^{i}A^{k}$（ｋ）、すなわち、そのペアレント（ｐａｒｅｎｔ）の物体ｋのヒンジポイントの一般化加速度は、以下：
【０１０６】
【数４７】
$${}^{i}A^{k}(k) = \begin{pmatrix} {}^{i}\alpha^{k}(k) \\ {}^{i}a^{O_k}(k) \end{pmatrix} = H^{*}(k)\dot{u}(k)$$

によって与えられる。
【０１０７】
ここで計算されるのは、この記述目的のための、分子系の、これらの一般化座標
【０１０８】
【数４８】

$\boldsymbol{q}$:

、一般化速度
【０１０９】
【数４９】

$\boldsymbol{u}$

、内部座標である。この代表的な慣性座標（ｘ，ｙ，ｚ）およびこれらの慣性座標系における速度を用いて取り組むよりも、対象分子系についての計算が軽減される。
【０１１０】
（運動方程式の計算）
記載のように例示的な剛体多体系、ねじれ角モデルが与えられると、運動方程式が計算され得る。本発明に従って、このＭＢＳ分子モデルの運動は、残差形式によって、決定される。この残差形式法は、「第１」運動学的計算と呼称される計算を必要とし、「第２」運動学的計算とこの「第１」運動学的計算を区別し、ここで、第２の運動学的計算は、直接形式（これは、比較目的のためにこの説明に含まれる）によってさらに必要とされる。
【０１１１】
（分子モデルのための第１運動学的計算）

第１の運動学的計算において、この分子系の内部座標、すなわち、
【０１１２】
【数５０】
$$(q,u,\dot{u})$$

およびこの系のパラメータが与えられると、分子モデルの各剛体ｋについての、以下の位置、速度および加速度の運動学的状態が計算される（ちなみに、第１運動学的計算は、残差形式法について実施された場合、ここで、
【０１１３】
【数５１】
$$\dot{u}$$

が、推定された解として、投入され、次いで、この積分法がこの解を、正確な解へと精緻化することに留意すべきである。対照的に、
【０１１４】
【数５２】
$$\dot{u}$$

が、直接形式について使用される場合には、零（０）に設定される。このことは、後の２つ方法の説明において明確に示される）。
【０１１５】
物体ｋの各々について、
【０１１６】
【数５３】
$$
\begin{aligned}
&{}^{N}C^{k}(k), r^{Q_iQ_k}(k), r^{OQ_k}(k), {}^{i}\phi^{k}(k),\\
&{}^{N}\omega^{k}(k), {}^{N}v^{Q_k}(k), V(k),\\
&{}^{N}\alpha^{k}(k), {}^{N}a^{Q_k}(k), A(k)
\end{aligned}
$$

を計算する。これらの計算は、反復的に実行され、基準の物体の各々から開始され、その細部へと進行する。
【０１１７】
${}^{N}C^{k}$（ｋ）、すなわち、基準（ｇｒｏｕｎｄ）におけるその物体ｋについての方向余弦行例が、以下のように定義される：
${}^{N}C^{k}$（１）＝${}^{i}C^{k}$（１）
${}^{N}C^{k}$（ｋ）＝${}^{N}C^{k}$（ｋ）${}^{i}C^{k}$（ｋ），ｋ＝２，．．．ｎ，ｉ＝ｉｎｂ（ｋ）
。
${}^{i}C^{k}$（ｋ）は、上記のジョイントルーチンから得られる。
【０１１８】
そのペアレント系において表現される場合、物体ｋのペアレントのヒンジポイントQ_iからの、物体ｋのペアレントのヒンジポイントQ_kへの位置ベクトルである、
【０１１９】
【数５４】
$$r^{Q_iQ_k}(k)$$

は、以下：
【０１２０】
【数５５】
$$r^{Q_iQ_k}(k)=\mathbf{d}(k)+r^{P_kQ_k}(k),\quad k=1,\ldots n$$

のように定義される。
【０１２１】
【数５６】

$$r^{P_k Q_k}(k)$$

は、ジョイントルーチンから得られる。
【０１２２】
グローバル系として表現される場合、慣性系の原点Ｏから物体ｋのヒンジポイントＱ$_k$への位置ベクトルである
【０１２３】
【数５７】

$$r^{OQ_k}(k)$$

は、以下：
【０１２４】
【数５８】

$$r^{OQ_k}(1) = r^{Q_i Q_k}(1)$$

$$r^{OQ_k}(k) = r^{OQ_k}(i) + {}^{N}C^{k}(i) r^{Q_i Q_k}(k), \quad k = 2,\ldots n, \quad i = inb(k)$$

のように定義される。
【０１２５】
物体ｋについての剛体の変換演算子$^{i}\phi^{k}$は、以下：
【０１２６】
【数５９】

$$ {}^{i}\phi^{k}(k) = \begin{pmatrix} {}^{i}C^{k}(k) & \tilde{r}^{Q_i Q_k}(k)\,{}^{i}C^{k}(k) \\ \underline{\underline{0}}_3 & {}^{i}C^{k}(k) \end{pmatrix}, \quad k = 1,\ldots n$$

のように定義される。
【０１２７】
物体ｋの系で表現される場合、そのヒンジポイントでの物体ｋについて空間ベクトルＶ（ｋ）は、以下：
【０１２８】
【数６０】

$$V(1) \triangleq \begin{pmatrix} {}^{N}\omega^{k}(1) \\ {}^{N}v^{Q_k}(1) \end{pmatrix} = {}^{i}V^{k}(1)$$

$$V(k) \triangleq \begin{pmatrix} {}^{N}\omega^{k}(k) \\ {}^{N}v^{Q_k}(k) \end{pmatrix} = {}^{i}\phi^{k*}(k) V(i) + {}^{i}V^{k}(k), \quad k = 2,\ldots n, \quad i = inb(k)$$

のように定義される。
【０１２９】
物体ｋの系で表現される場合の、そのヒンジポイントにおける、物体ｋについての空間的加速度Ａ（ｋ）は、以下：
【０１３０】
【数６１】

$$A(1) \triangleq \begin{pmatrix} {}^N\alpha^k(1) \\ {}^N a^{Q_k}(1) \end{pmatrix} = {}^i A^k(1)$$

$$A(k) \triangleq \begin{pmatrix} {}^N\alpha^k(k) \\ {}^N a^{Q_k}(k) \end{pmatrix} = \overline{A} + \begin{pmatrix} \tilde{\omega} & \underline{\underline{0}}_3 \\ \underline{\underline{0}}_3 & 2\tilde{\omega} \end{pmatrix} {}^i V^k(k) + {}^i A^k(k), \quad k = 2,\ldots n, \quad i = inb(k)$$

で定義され、ここで、

【０１３１】

【数６２】

$$\overline{A} = {}^i\phi^{k*}(k)A(i) + \begin{pmatrix} \underline{0}_3 \\ {}^i C^{k*}(k)\left({}^N\omega^k(i) \times {}^N\omega^k(i) \times r^{Q_i Q_k}(k)\right) \end{pmatrix}$$

$$\omega = {}^i C^{k*}(k)\,{}^N\omega^k(i)$$

である。

当然、これらの計算は、所望される場合、全て、単一の経路で計算され得る。

【０１３２】

時間刻み１増分についてのこれらの工程を完結した後に、ＭＢＳは、任意の物体の任意のポイントについて、（一般化）位置、（一般化）速度または（一般化）加速度の情報を計算するための運動論的要求に助力を提供する。これは、標準的な剛体式を使用して、その物体についてのヒンジ量に関する任意のポイントについて必要とされる情報を計算することによってなされる。

【０１３３】

（残差計算）

上記の第１運動学的計算を使用して、残差形式法のための残差計算が決定され得る。残差形式およびその分子モデリングへの適用の詳細な説明は、「ＭＥＴＨＨＯＤ　ＦＯＲ　ＲＥＳＩＤＵＡＬ　ＦＯＲＭ　ＩＮ　ＭＯＬＥＣＵＬＡＲ　ＭＯＤＥＬＩＮＧ」と題される、同一日に出願された上述の同時係属中の米国特許出願番号において見出される。この計算は、上で与えられたベクトル

【０１３４】

【数６３】

$$\begin{pmatrix} \rho_q \\ \rho_u \end{pmatrix}$$

の２つの部分からなる。第１の部分は、ρ_qと呼称される、運動学的残差であり、そして、第２の部分は、ρ_uと呼称される、動力学的残差である。この運動学的残差は、

【０１３５】

【数６４】

$$\dot{q}$$

（これは、（陰的）積分サブモジュール６６から投入される）と、各ジョイントによって計算された導関数との間の差から計算される：

【０１３６】

【数６５】

$$\dot{q} - W(q)u = \rho_q$$

。

【０１３７】

動力学的残差がまた、計算される。分子モデルの所定状態、すなわち、所定の

【０１３８】

【数６６】

$$(q,u,\dot{u})$$

および系のパラメータを用いて開始して、プログラムルーチンは、ＭＢＳの「環境」をモデリングする。このようなルーチンは、コンピュータモデリング分野の当業者にとって容易に利用可能であり、作製可能であり得る。このルーチンは、積分サブモジュール６６の形態によって決定され、そして通過する、値（ｑ，ｕ）をとり、（状態依存的な）そのヒンジポイントQ_kにおける物体ｋについて適応された空間的力

【０１３９】

【数６７】

$$T(k)=\begin{pmatrix} T_{Q_k}(k) \\ F(k) \end{pmatrix}$$

、物体ｋについてのヒンジトルクσ（ｋ）を返す。その後、物体ｋについての一般化速度ｕ（ｋ）に関するこの動力学的残差ρ_u（ｋ）は、以下の工程１～３：　１．投入された（ｐａｓｓｅｄ－ｉｎ）状態の値

【０１４０】

【数６８】

$$(q,u,\dot{u})$$

を用いて、上記のように残差形式によって分子モデルのための計算を実施する

２．ｎ個の物体を有するモデルにおける各物体ｋについての空間的負荷収支

【０１４１】

【数６９】

$$\hat{T}(k)$$

の生成：

【０１４２】

【数７０】

$$\hat{T}(k)=M(k)A(k)+\begin{pmatrix} {}^N\tilde{\omega}^k(k)\left(\underline{\mathbf{I}}_{\mathcal{Q}}(k)\,{}^N\omega^k(k)\right) \\ {}^N\tilde{\omega}^k(k)\left({}^N\omega^k(k)\times\mathbf{p}(k)\right) \end{pmatrix}-T(k)$$

$$k=1,\ldots,n$$

３．ρ_u（ｋ）の計算

【０１４３】

【数７１】

$$\begin{aligned} &\text{for } k=n \text{ to } 2 \text{ by } -1 \\ &\quad \rho_u(k)=H(k)\hat{T}(k)-\sigma(k) \\ &\quad i=inb(k) \\ &\quad \hat{T}(i)\mathrel{+}={}^i\phi^k(k)\hat{T}(k) \\ &\text{end} \\ &\rho_u(1)=H(1)\hat{T}(1) \end{aligned}$$

によって、計算される。

【０１４４】

残差形式法は、微分方程式が満たす程度を評価する。零（０）の残差は、適応された力が、慣性力とバランスをとることを示す。しかし、このことは、この系が静的平衡にあるこ

とを意味せず、それよりもむしろ、適用された力が、状態（ｑ，ｕ）にある系に対して適用された場合に、所定の

【０１４５】

【数７２】

$\dot{u}$

を再形成することを示す。この残差は、適用された力および慣性力とバランスをとるために必要とされる、さらなるヒンジトルクとして解釈され得る。文献によると、この方法は、逆方向動力学（ｉｎｖｅｒｓｅ　ｄｙｎａｍｉｃｓ）、または計算されたトルクの方法のいずれかとして公知である。これは、

【０１４６】

【数７３】

$\dot{u}$

が全て規定された場合を支配する。この点において、残差形式について必要とされる計算は、完全である。この残差ρ_qおよびρ_uは、積分器サブモジュール６８の中の陰的積分器によって直接的に使用される。

【０１４７】

（分子モデルのための第２運動学的計算）

直接形式法を実行するために、第１の運動学的計算が必要とされる。これらのさらなる計算は、第２運動学的計算と称する。Ｐ（ｋ）、Ｄ（ｋ）、${}^{i}\Psi^{k}$（ｋ）、${}^{i}K^{k}$（ｋ）の値が、以下のように計算する：

１．上述したような残差形式によって分子モデルについての計算を実施（すなわち、第１運動学的計算）

２．各物体ｋの有節（ａｒｔｉｃｕｌａｔｅｄ）物体の慣性Ｐ（ｋ）の初期化。

【０１４８】

Ｐ（ｋ）＝Ｍ（ｋ）、ｋ＝１，．．．，ｎ

３．以下の対象の生成

【０１４９】

【数７４】

for $k = n$ to 2 by -1

$D(k) = H(k)P(k)H^{*}(k)$

$G = P(k)H^{*}(k)D^{-1}(k)$

$\overline{\tau} = \underline{\underline{E}}_6 - GH(k)$

${}^{i}\psi^{k}(k) = {}^{i}\phi^{k}(k)\overline{\tau}$

${}^{i}K^{k}(k) = {}^{i}\phi^{k}(k)G$

$i = inb(k)$

$P(i) \mathrel{+}= {}^{i}\psi^{k}(k)P(k){}^{i}\psi^{k*}(k)$

end

$D(1) = H(1)P(1)H^{*}(1)$

これらの量の関数依存性は、一般化座標であるｑついてのみである。従って、この第１運動学的計算は、第２運動学的計算を見越してプログラムされる。

【０１５０】

（前進動力学計算）

最後に、

【０１５１】

【数７５】

$\dot{u}$

がこの分子の、以下のように、内側、次いで、外側に、及ぶことよって、計算される：
【０１５２】
【数７６】

$z(k) = \underline{0}_6, \quad k = 1,\ldots n$
for $k = n$ to 2 by -1
$\varepsilon(k) = \rho_u(k) - H(k)z(k)$
$\upsilon(k) = D^{-1}(k)\varepsilon(k)$
$i = inb(k)$
$z(i) \mathrel{+}= {}^{i}\psi^{k}(k)z(k) + {}^{i}K^{k}(k)\rho_u(k)$
end
$\varepsilon(1) = \rho_u(1) - H(1)z(1)$
$\upsilon(1) = D^{-1}(1)\varepsilon(1)$
$\dot{u}(1) = \upsilon(1)$
$\delta(1) = H^{*}(1)\upsilon(1)$
for $k = 2$ to n
$i = inb(k)$
$\delta(k) = {}^{i}\psi^{k*}(k)\delta(i) + H^{*}(k)\upsilon(k)$
$\dot{u}(k) = \upsilon(k) - {}^{i}K^{k*}(k)\delta(i)$
end

。
第１運動学的計算および第２運動学的計算、ならびに前進的動力学計算を使用して、この直接形式法が利用可能である。
【０１５３】
（運動方程式のための直接形式法）
直接形式法は、現在の状態（ｑ，ｕ）を取り、そしてその微分
【０１５４】
【数７７】

$(\dot{q}, \dot{u})$

を上記アルゴリズムを使用して計算し、このアルゴリズムは、時間を前進する積分法によって使用される。
【０１５５】
状態（ｑ，ｕ）が与えられると、
以下：
【０１５６】
【数７８】

$(\dot{q}, \dot{u})$

を計算する。
１．上記のようにジョイント特異的なルーチンを使用して
【０１５７】
【数７９】

$\dot{q}$

を計算する
２．上記の第１運動学的計算を、
【０１５８】
【数８０】

$$\dot{u}=0$$

を用いて実施する
３．上記のように、残差ρ$_u$を生成する
４．残差ρ$_u$＝－ρ$_u$を否定する
５．第２運動学的計算を実施する
６．上記前進動力学計算を使用して、
【０１５９】
【数８１】

$$\dot{u}$$

を計算する。
【０１６０】
直接形式法は、この系について作用する適用された力に応答してヒンジ加速度
【０１６１】
【数８２】

$$\dot{u}$$

を生成する。図５は、残差形式法および直接形式法の計算工程を概説する。
（陰的積分法におけるヤコビアン）
分子（例えば、タンパク質）をモデリングするＭＤ方程式は、多体系（ＭＢＳ）としてインプリメントされる。これらの方程式は、Ｎｅｗｔｏｎの法則を表し、そして、以下の微分方程式：
【０１６２】
【数８３】

$$\dot{y}=f(y,t)$$

の集合によって、表現される。この微分方程式は、Ｎ次多体系動力学法のスイートを使用して、インプリメントされる。経時的にこの方程式を前進させるために、本発明に従って、数値積分の陰的方法、特にＬ安定性陰的積分法（例えば、陰的Ｅｕｌｅｒ法、Ｒａｄａｕ５法およびＳＤＩＲＫ３法）が使用され得る。
【０１６３】
この積分過程の重要な成分は、この微分方程式のヤコビアンの生成である。これは、以下：
【０１６４】
【数８４】

$$J\triangleq\frac{\partial f}{\partial y}$$

である。この関数ｆは、それ自体、陽的な定式よりはむしろ、アルゴリズムによって計算さるので、このヤコビアン計算は、実質的な量の仕事を表現する。最も単純なアプローチにおいて、このヤコビアンは、この微分ルーチンを区別することによって数値的に形成され得る。これは、扱いにくい操作である。なぜならば、このヤコビアンの質は、丸め誤差と打ち切り誤差との間の諸条件の兼ね合いであるからである。結果における、代表的に半分の実際的な精度が、異なるスキームの中で十分な摂動サイズを選択するこのよって保持される。実際には、しかし、これは、実行することは困難である。

【０１６５】
しかし、この支配的な方程式の構造を活用して、ヤコビアンの計算を改善し得る。この例示的な多体系動力学法は、これを例示する。関連するアルゴリズムは数値的な方法がこの式を実行するために使用されたとしても、正確な微分を計算する。得られる微分は、丸め誤差および検討中の多体系の条件付けに依存する量による誤差の範囲にある。しかし、何一つ近似は、この方程式のレベルには含まれていない。
【０１６６】
一般に、Ｇ、つまり、この陰的積分器のＮｅｗｔｏｎループにおいて使用される反復行列は、Ｇ＝Ｅ－αＪの形式を有し、ここで、Ｅは、単位行列であり、αは、時間刻みのスカラー関数である。陰的積分器の記載について、同一日付けで出願された、「ＭＥＴＨＯＤ ＦＯＲ ＬＡＲＧＥ ＴＩＭＥＳＴＥＰＳ ＩＮ ＭＯＬＥＣＵＬＡＲ ＭＯＤＥＬＩＮＧ」と題する、上述の参照した米国特許出願番号＿＿＿＿＿を参照のこと。刻み幅の変化は、Ｇを因数分解（ｒｅｆａｃｔｏｒ）することを要求するが、しかし、Ｊを再形成（ｒｅｆｏｒｍ）することは必要としない。Ｊを再度形成することは、ヤコビアンが新状態で必要とされる場合のみ、必要とされる。Ｇは、Ｎｅｗｔｏｎループ内での線形性の副次的問題において使用される。以下のように解かれる：
【０１６７】
【数８５】

$$G\Delta y = -r(y_n^i),$$
$$y_n^{i+1} = y_n^i + \Delta y$$

ここで、ｒ（ｙ）は、特定の陰的積分法についての残差関数である。
【０１６８】
後ほど示されるように、Ｊは特別な構造を有しており、この構造は、Ｇに受け継がれている。これは、Ｇを用いた方程式の解法が、この構造が活用された場合に、少ない損失で実施され得ることを意味する。
【０１６９】
ヤコビアンの質は、積分器における打ち切りから得られる非線形性方程式を解く能力に影響を与える。Ｎｅｗｔｏｎループが解けないことによって、追跡工程の撤回、および積分時間刻み工程の減少を要求する。この時間刻みは、Ｎｅｗｔｏｎループにおける失敗よりも、むしろ精度によって制御されるべきある。
【０１７０】
（解析的ヤコビアンの計算）
ヤコビアンＪは、運動方程式の線形化を表す行列である。通常は、動力学的系を支配する方程式は、平衡状態、あるいは、定常運動状態の周囲で線形化される。この場合、方程式が、任意の状態の周りで線形化され、そして、全ての可能性のある寄与する項が、展開されるべきである。Ｊを以下のその分割に関してＪを記述することは、慣例的である；
【０１７１】
【数８６】

$$J(q,u) = \begin{pmatrix} \frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial u} \\ \frac{\partial \dot{u}}{\partial q} & \frac{\partial \dot{u}}{\partial u} \end{pmatrix} \triangleq \begin{pmatrix} J_{qq} & J_{qu} \\ J_{uq} & J_{uu} \end{pmatrix}$$

。
【０１７２】
（J_{qq}およびJ_{qu}の構造）
この
【０１７３】

【数８７】

$$\dot{q}$$

の方程式は、

【０１７４】

【数８８】

$$\dot{q} = Wu$$

であり、ここで、行列Ｗは、ブロック対角の構造を有している。各ブロックは、そのジョイント型に依存する。ピンジョイントおよびスライダージョイントは、１×１恒等ブロックを生じる。ボールジョイントは、Ｅｕｌｅｒパラメータおよび物体の角速度測定数（一般化速度）に関してＥｕｌｅｒパラメータ微分を表現する４×３ブロックを生じる。

【０１７５】

上記の

【０１７６】

【数８９】

$$\dot{q}$$

の方程式から、ヤコビ行列の２つの分割が見出される：

【０１７７】

【数９０】

$$J_{qq} = \frac{\partial W(q)}{\partial q} u$$

$$J_{qu} = W$$

。

【０１７８】

これらの方程式は、象徴的な目的のみであると解釈される。実際において、行列Ｗを陽的に生成する必要はない。非ゼロのブロック対角要素は、運動学的残差についての前述の節で議論されたように満たされる。

【０１７９】

（J_{uq} および J_{uu} の構造）

微分

【０１８０】

【数９１】

$$\dot{u}$$

は、さらに複雑である。なぜならば、

【０１８１】

【数９２】

$$\dot{u} = -M^{-1}\rho_u$$

であるので、

【０１８２】

【数９３】

$$\frac{\partial\left(-M^{-1}\rho_u\right)}{\partial q} \text{ および } \frac{\partial\left(-M^{-1}\rho_u\right)}{\partial u}$$

が計算されなければならない（ρ_uが以前に展開された動力学的ルーチンの残差であることに留意のこと）。ここで、数値解析の分野からの重要な結果は、この逆行列の微分を回避するのに使用される。

【０１８３】

Ａ（ｘ）ｙ（ｘ）＝ｂ（ｘ）であるとすると、ｙ（ｘ）＝Ａ$^{-1}$（ｘ）ｂ（ｘ）あると記載し得る。ｙ（x_0）＝ｚが既知であり、かつ、

【０１８４】

【数９４】

$$\frac{\partial y(x)}{\partial x}\Big|_{x=x0}$$

の値が得られなければならない場合、本発明者らは、

【０１８５】

【数９５】

$$\frac{\partial y}{\partial x}\Big|_{x=x0} = A^{-1}(x0)\frac{\partial}{\partial x}\big(b(x)-A(x)z\big)\Big|_{x=x0}$$

を有し、ここで、ｚ＝Ａ$^{-1}$ｂは、この方程式の右辺を形成するときに固定される。上述の多体系ルーチンに適用された場合、

【０１８６】

【数９６】

$$\frac{\partial \dot{u}}{\partial x} = -\frac{\partial}{\partial x}\big(M^{-1}\rho_u(q,u,0)\big) = -M^{-1}\frac{\partial}{\partial x}\rho_u(q,u,z)$$

であり、ここで、

【０１８７】

【数９７】

$$z \triangleq M^{-1}\rho_u(q,u,0)$$

である。この結果は、Ｍ$^{-1}$（これは、超行列（ｈｙｐｅｒｍａｔｒｉｘ）である）の微分を計算することを回避する。この逆行列は、「引き出される」。上記の方程式において、「ｘ」は、一般化座標ｑか、または一般化速度ｕであり、計算されるヤコビアン分割に依存する。

【０１８８】

要約すると、ブロックJ_{uq}およびJ_{uu}を計算するために、３つの工程が続く：

１．所定の（ｑ，ｕ）は、直接法を使用して

【０１８９】

【数９８】

$$z \triangleq -M^{-1}\rho_u(q,u,0)$$

を計算する。これは、単純に

【０１９０】

【数９９】

$$\dot{u}$$

を現在の状態から計算することをいい、そして、これを変数ｚとして保存する。

【０１９１】

２．動力学的残差ルーチンの解析的ヤコビアンを計算する。この工程において、行列

【０１９２】

【数１００】

$$\frac{\partial}{\partial x}\rho_u(q,u,z)$$

が形成される。この量は、明らかに工程１で計算されたベクトルｚに依存する。この工程における残差ρ$_u$の数値的値は、発明者らは、この運動方程式の共通解のまわりでヤコビアンを計算しているので、各要素に対して零（０）であることを留意のこと。この残差ヤコビアンの分割Ｊ$_{uq}$およびＪ$_{uu}$は、上記「ｘ」について「ｑ」および「ｕ」で置換することによって得られる。

３．工程２の結果を、質量行列を用いて後退代入法することによって、所望のブロックを得る。この後退代入法操作は、残差ベクトルを、

【０１９３】

【数１０１】

$$\dot{u}$$

ベクトルへと処理することによって、直接形式ルーチンにおいて解決される。第２運動学的工程は、１度のみ実施される必要がある。なぜならば、この後退代入法は、この状態の名目上の値で実施されるからである。実際に、この第２運動学的ルーチンは、ｚを計算している間に、工程１で呼び出され、そしてこの変数はキャッシュに入れられる。

【０１９４】

言葉で表すと、本発明の微分ルーチンのヤコビアンは、本発明者らの残差ルーチンのヤコビアンの後退代入法によって実施され得る。この残差ヤコビアンは、この微分ヤコビアンよりもずっと容易に計算される。上記の工程２および工程３は、以下の節で導出される。

【０１９５】

（残差ヤコビアン）

残差ヤコビアンの計算は、力学についての残差形式に密接に関連し、ここで以下のように要約する：

１．位置および速度に依存する運動データを計算する、外側への（ｏｕｔｂｏａｒｄ）パスを実行する。

２．原子間力を発生させ、そしてこれらを物体に作用する空間的負荷に強化する、力ルーチンを呼び出す。

３．加速度レベル値を（入力した

【０１９６】

【数１０２】

$$\dot{u}$$

を使用して）計算し、そして慣性力を工程２からの空間的負荷と組み合わせる、別の運動パスを実行する。

４．各結合における残差を発生させる内向パスを実行する。このパスは、問題の結合の外側物体の「ネスト」について、（空間的）慣性力と適用された力との合力を再帰的に計算する。残差は、結合マップデータによって与えられる、結合の自由度の合力の投影である。

【０１９７】

高レベルにおいて、残差の計算は、２種類の力（「運動力」および外力）に依存すると考えられ得る。運動力は、多体系によって直接計算される。外力は、種々の原子間力（例えば、静電力および溶媒）を計算する力モデリングルーチンから、多体系に対して利用可能である。ヤコビアンを計算する場合に、類似の手順に従う。多体系は、運動力のヤコビアンを構築し、そしてこれを外力のヤコビアンと組み合わせる。

【０１９８】

（力場の作用への分配）

分子に作用し得る多くの力が存在し、そしてこれらの力は、種々の固有座標系（これらは、この特定の力の「効果」に対して最も好都合である）において計算され得る。例えば、静電力の項は、多極方法および球座標を使用して計算され得、共有結合の項は、ねじれ角および結合角に関して計算され得、そして溶媒力の項は、グローバルデカルト座標において計算され得る。残差の計算の間に、これらの力は、それらの固有座標系からＭＢＳ座標系へと変換される。

【０１９９】

同じ交換が、ヤコビアンを計算するために起こる。ネイティブのヤコビアンは、その固有座標において、ＭＢＳ座標にされる。このことは、固有座標とＭＢＳ一般化座標との間での変換のために、連鎖法則の使用を必要とする。各効果が、その関数値およびヤコビアンを同時に計算することが重要である。なぜなら、同じ項の多くが、それぞれ計算される必要があるからである。各効果は、空間負荷T_{effect}（ｋ）のセットに変換され、ここで、ｋは、系における一般的な物体の指数である。これらの効果の合計は、記号Ｔ（ｋ）を与えられる。

【０２００】

（ヤコビアンを残差ヤコビアンにする効果）

高レベルにおいて、残差ルーチンは、以前に、以下の等式：

$\rho_u = H\phi(MA-T)$

から実行された。この実行は、次数（Ｎ）の方法（これは、上記等式からすぐに明らかである）を使用する。この等式において、Ｔは、多体系の旋回に作用する空間的負荷のベクトルであり、ここで、各要素は、空間的負荷である（１つの力および１つのトルクからなる６ベクトル）。これは実際に、慣性負荷または純粋なヒンジ負荷以外の全ての効果を表す。括弧内の項は、各物体に対する負荷のバランスを表す。第一項は、慣性力であり、そして次の項は、空間的負荷である。Ｍ（ｋ）Ａ（ｋ）は、代表的な物体に対する空間的慣性力である。これは、物体の質量特性および物体の旋回の空間的な加速度から構築される。この空間的な加速度は、残差ルーチンが先行する力学ルーチンによって実行される前に、計算される。演算子Ｈφは、次数（Ｎ）の内向パスを実行するルーチンにおいて、実行される。

【０２０１】

上記等式によって実行される計算の詳細についての全てを知ることなしにさえ、

【０２０２】

【数１０３】

$$\frac{\partial T}{\partial x}$$

（空間的負荷ヤコビアン（効果ヤコビアン））の、

【０２０３】

【数１０４】

$$\frac{\partial \rho_u}{\partial x}$$

（残差ヤコビアン）に対する寄与が、すぐに推定され得る：

【０２０４】

【数１０５】

$$\frac{\partial \rho_u}{\partial x} = -H\phi\frac{\partial T}{\partial x} + \ldots$$

ここで、．．．は、項が効果ヤコビアンを含まないことを表す。再度、「ｘ」についてのｑまたはｕは、ヤコビアンのどの分配が計算されるかに依存して、置換される。

【０２０５】

残差ヤコビアンにおける効果ヤコビアンの役割は、マルチ物体等式における効果の役割と同じである。このことは、Ｔが、残差ρ_uに対して、

【０２０６】

【数１０６】

$$\frac{\partial T}{\partial x}$$

の列が

【０２０７】

【数１０７】

$$\frac{\partial \rho_u}{\partial x}$$

の列に寄与する様式と同じ様式で寄与することを意味する。両方が、同じ演算子Ｈφによって処理される。これは、非常に重要な点である。なぜなら、このことは、ヤコビアン計算のこの部分に対して新たな方法が必要とされないことを意味するからである。（異なる方法は、

【０２０８】

【数１０８】

$$\frac{\partial T}{\partial x}$$

を得ることを必要とする）。

【０２０９】

従って、効果ヤコビアンを考慮して、その寄与は、この効果ヤコビアンに対してそのもとの残差ルーチンを実行すること、この効果ヤコビアンをマルチ列セットの空間的負荷ベクトルとして処理することによって、残差ヤコビアンに組み立てられる。このことは、この等式の直線性の線形性である結果である。

【０２１０】

残差ヤコビアンの列は、残差ベクトルが順方向力学ルーチンにおいて果たすものと同じ役割を、微分ヤコビアンルーチンにおいて果たす。力学ルーチンは、このルーチンが受信するデータベクトルに対して後退代入法（ｂａｃｋ－ｓｏｌｖｅ）を実行し、そしてどのデータであるかを知る必要がなく、そのデータに対してどの演算が実行されているかのみを知る必要がある。これは、全てのルーチンに当てはまる。

【０２１１】

．．．の項を追加することによって、連鎖法則を使用し、等式全体を以下のように示す：

【０２１２】

【数１０９】

$$\frac{\partial \rho_u}{\partial x} = H\phi\left(\frac{\partial (MA)}{\partial x} - \frac{\partial T}{\partial x}\right) + H\frac{\partial (\phi z)}{\partial x} + \frac{\partial (Hy)}{\partial x}$$

このレベルにおいて、ヤコビアンに対して４つの寄与（慣性力、空間的な力、ならびに演算子φおよびＨにおける変化に起因する寄与）が存在する。量ｚおよびｙは、それぞれ（ＭＡ－Ｔ）およびφｚを表し、これらは、最後の２つの項を詳説する間、一定に維持される。これらの項の数値は、残差計算から既に利用可能である。上記等式についての別の観察は、演算子φがｑのみに依存し、そしてｕには依存しないことである。従って、この項は、分配J_{uu}が計算されている間、一定のままである。同様に、空間負荷は、その連続した効果に分離しえ、これらのいくつかは、ｕに依存しない。一般に、このことは、マルチ物体等式の知識が、計算を最適化するために利用され得る。

【０２１３】
ここまでで、一旦項が計算されたら
【０２１４】
【数１１０】
$$\frac{\partial T}{\partial x}$$

をどう取り扱うかが記載されてきたが、この項をどのように形成するかの説明はなされていない。これらの詳細を、以下の節に示す。
【０２１５】
（効果ヤコビアンの計算および残差ヤコビアンとの組み合わせ）
ここまでは、ヤコビアン計算の高レベルの説明がなされた。これらの計算は、それを行うためのアルゴリズムの特色が非常に強い。先に記載された順方向力学ルーチンについてもまた存在したような、作業のための非常に異なる段階が存在する。そこで、原子間力の計算は、明らかに、ボトルネックの工程である。しかし、力を取り扱うためのマルチ物体の等式におけるオーバーヘッドは、かなり小さい。この場合、呼び出しが、力ルーチンに対してなされ、そしてこのルーチン内において起こることは、無視された。ヤコビアンのこととなると、この局面はさほど正確ではない。
【０２１６】
呼び出しが、効果ヤコビアンを得るために依然としてなされるが、効果ヤコビアンが残差ヤコビアンに組み立てられ得る前に必要とされる、多数の処理が存在する。力場を扱ってヤコビアンを作成する詳細は、次の節に網羅されており、そして静電力を組み込む例が開発される。他の全ての付加は、類似の発達に従う。
【０２１７】
（例の効果としての静電力）
静電力の基本的な前提は、２つの荷電粒子間の力が、以下：
【０２１８】
【数１１１】
$$F_{ij} = \kappa \frac{q_i q_j}{r_{ij}^2} \hat{r}_{ij}$$

であることである。これは、古典的な逆二乗則である。F_{ij}（粒子ｊに起因して粒子ｉに作用する力）は、これらの粒子の電荷に依存し、反対に荷電した粒子については引力であり、同じ荷電については反発力である。記号
【０２１９】
【数１１２】
$$\hat{r}_{ij}$$

は、粒子ｉから粒子ｊへの方向の単位ベクトルである；r_{ij}は、これらの粒子間の距離である；ｋは、力の強度に関連する、単位に依存する定数である。もちろん、粒子ｊに作用する力は、粒子ｉに作用力に等しく、そしてその逆である。従って、上で与えたクーロンの法則によって相互作用する粒子の収集を考慮すると、各粒子に作用する正味の力は、対ごとの力を合計することによって計算される。この例に対して、これらの力は、グローバルデカルト座標において計算される。
【０２２０】
原子間力が使用できる状態で、マルチ物体力が発生し得る。各物体の粒子に作用する力の系は、各物体の旋回において作用する空間的負荷によって置換される。原子間力は、物体を固定した基礎において最初に表現され、次いで、その力が結合する特定の原子のステーション座標を使用して、その旋回にシフトされる。
【０２２１】

F_{ij}は、ベクトルである。dr_{ij}に対するF_{ij}（粒子の相対位置の小さな変化）の微分は、

【０２２２】

【数１１３】

$$\underline{\underline{D}}_{ij}$$

（テンソル）である。これは：

【０２２３】

【数１１４】

$$dF_{ij} = \underline{\underline{D}}_{ij} \bullet dr_{ij}$$

となるようなものである。クーロン力に対して、テンソルは：

【０２２４】

【数１１５】

$$\underline{\underline{D}}_{ij} = \kappa \frac{1}{r_{ij}^3}\left(\underline{\underline{E}}_3 - 3\hat{r}_{ij}\hat{r}_{ij}^*\right)$$

である。

【０２２５】

（いくつかの観測）

１．力ヤコビアンは、大きさ$n_{atoms} \times n_{atoms}$の大きさの行列である。各要素は、３×３のテンソルである。（ｉ，ｊ）ブロックは、原子ｊの位置の小さな変化に対する原子ｉへの力の微分を与える。一般に、全ての力のモデルが、分析処理のための固有ヤコビアン方法を支持するために必要とされる。

２．力ヤコビアンについての格納要件は、急速に非実用的になる。このことは、原子間の対ごとに作用する全ての力のヤコビアンが、物体間の対ごとに作用するように、減少するかまたは「縮小」される場合の、界面の「縮小」の概念を導く。

３．クーロン力は対ごとの相互作用であるので、各力が、ヤコビアン全体における２つのブロックに寄与する。従って、各力は対照的な費用で処理され、そしてヤコビアン全体は、原子の数の二乗（すなわち、次数（N^2））に比例する費用で計算される。このことは、力自体の計算費用と同じである。これは、解析的ヤコビアンを計算するために非常に良い結果である。数値のヤコビアンは、その状態の要素が摂動するごとに、新たな力の計算を必要とする。これは、数値のヤコビアンの費用に、三乗の成長（すなわち、次数（N^3））を導く。従って、解析的ヤコビアンは、数値のヤコビアンより計算が安価であり、そしてより正確である。

４．ヤコビアンの計算は、力を計算するルーチンと同じルーチンにおいて、好都合になされる（同時計算）。しかし、これは代表的に、力の計算よりはるかに少ない頻度でなされる必要がある。従って、フラッグが使用されて、必要な場合にのみヤコビアンの計算を誘発する。

【０２２６】

（変位勾配へのカップリング）

（固有の）力ヤコビアンが得られたら、そのヤコビアンをさらに処理することが必要である。このことは、多体系が相対座標を使用して形成されるという事実に起因する。連鎖法則が各原子間力Ｆ（ｋ，ｉ）に適用されて、「変位勾配の計算」と称される。これは、物体ｋに属する原子ｉに対するグローバルデカルト力を表す。

【０２２７】

【数１１６】

$$\frac{\partial F(k,i)}{\partial q_j} = \sum_{p=1}^{nbod} \sum_{s \in p} \frac{\partial F(k,i)}{\partial r(p,s)} \frac{\partial r(p,s)}{\partial q_j}$$

ここで、ｒ（ｐ，ｓ）は、物体「ｐ」上の原子「ｓ」の位置である。

【０２２８】

この合計の第一項は、計算されたばかりの力ヤコビアンの要素を選択する。量

【０２２９】

【数１１７】

$$\frac{\partial r(p,s)}{\partial q_j}$$

は、変位勾配の要素である。代表的な項は、一般化座標における小さな変化に起因する、原子の位置の変化を与える。この項は、厳密には、力の計算とは無関係の運動量であることに注目のこと。従って、力ヤコビアンは一回計算され得、次いで、多体系における各座標に対する連鎖法則によって、連続的に再処理され得る。この工程は、行列ベクトルの多重を必要とする。なぜなら、

【０２３０】

【数１１８】

$$\frac{\partial r_j}{\partial q_s}$$

は、ｎａｔｏｍｓの実体（各３ベクトル）を有する列ベクトルであり、そしてヤコビアンは、ｎａｔｏｍｓ×ｎａｔｏｍｓの正方行列であり、ここで、各要素は３×３のテンソルであるからである。

【０２３１】

この計算を改善することが可能である。なぜなら、変位ヤコビアンにおける実体の多くは、０であることが既知であるからである。このことは、特定のヒンジを漸増させることによって、その系の全ての原子が変位するのではなく、変位したヒンジの外側でありかつ問題のヒンジから外れていない原子のみが変位するという事実に起因する。例えば、基本の物体を回転させることによって、その系の全ての原子の変化が誘導される。しかし、任意の末端物体においてねじれ角を摂動させることにより、その末端物体に属する原子のみの変化が誘導される。従って、大まかに言えば、およそ半分の仕事が、計算を最適化させることによって節約され得る。この減少は、厳密に知識に基づくアプローチから生じる。

【０２３２】

（界面の縮小）

Ｔ（ｋ）（物体ｋに対する空間的負荷）を形成するプロセスにおいて、この負荷は、物体ｋに属する原子に対して作用する原子間力から生じる。各力は、グローバル座標から局所座標に変換され、次いで物体旋回にシフトする。この手順の簡潔な言及は、以下である：

【０２３３】

【数１１９】

$$T(k) = \sum_{i \in k} \phi(k,i) T(k,i)$$

演算子φ（ｋ，ｉ）は：

【０２３４】

【数１２０】

$$\phi(k,i) = \begin{bmatrix} \underline{\underline{E}}_3 & \tilde{\rho}(k,i) \\ \underline{\underline{0}}_3 & \underline{\underline{E}}_3 \end{bmatrix} \begin{bmatrix} {}^N C^k(k) & \underline{\underline{0}}_3 \\ \underline{\underline{0}}_3 & {}^N C^k(k) \end{bmatrix}^*$$

である。ここで、ρ（ｋ，ｉ）は、物体ｋ上の原子ｉの固定されたステーション座標である。新たな値Ｔ（ｋ，ｉ）が現れることに注目のこと。これは、原子間力が、原子ｉの原子位置における空間的負荷に変わっただけのものである。

【０２３５】

【数１２１】

$$T(k,i)=\begin{bmatrix} \underline{0}_3 \\ F(k,i) \end{bmatrix}$$

原子の空間的負荷の第一の要素は、０である。なぜなら、個々の原子に対する力場によって、トルクが付与されないからである。

【０２３６】

ここで、Ｔ（ｋ）は、原子間力を物体の空間的負荷に関連付ける。従って、この等式の微分は、差示的原子間力と差示的空間的負荷とを関連付ける：

【０２３７】

【数１２２】

$$\frac{\partial T(k)}{\partial q_j}=\sum_{i\in k}\phi(k,i)\frac{\partial T(k,i)}{\partial q_j}+\frac{\partial \phi(k,i)}{\partial q_j}T(k)$$

$$=T_1(k)+T_2(k)$$

。この等式の第二項T_2（ｋ）は、この節の最後に議論される。なぜなら、この項は、空間的負荷を含むが、付加微分を含まないからである。このことは、この項が一般に、空間的負荷がどのように計算されるかを心配せずに処理され得ることを意味する。

【０２３８】

Ｔ（ｋ）の定義をT_1（ｋ）に置換すると：

【０２３９】

【数１２３】

$$T_1(k)=\sum_{i\in k}\phi(k,i)\sum_{p=1}^{nbod}\sum_{s\in p}\frac{\partial T(k,i)}{\partial r(p,s)}\frac{\partial r(p,s)}{\partial q_j}$$

$$=\sum_{p=1}^{nbod}\sum_{s\in p}\frac{\partial T(k)}{\partial r(p,s)}\frac{\partial r(p,s)}{\partial q_j}$$

であり、ここで、記号

【０２４０】

【数１２４】

$$\frac{\partial T(k)}{\partial r(p,s)}\triangleq\sum_{i\in k}\phi(k,i)\frac{\partial T(k,i)}{\partial r(p,s)}$$

は、現在、行が減少された力ヤコビアンの要素である。このヤコビアンは、差示的（物体）空間的負荷を、差示的原子変位に直接関連付ける。再度、ｒ（ｐ，ｓ）は、物体ｐ上の原子ｓのグローバルな位置を表す。項

【０２４１】

【数１２５】

$$\frac{\partial T(k)}{\partial r(p,s)}$$

は、各原子間力ヤコビアン要素を、原子のφ（ｋ，ｉ）行列によって重み付けされた、減少したヤコビアンにおける行き先要素に合計することによって、形成される。行が減少したヤコビアンの各要素は、６×３行列である。従って、力ヤコビアンの行は、縮小されて

いる。この縮小は、以下の記録において明らかである：分子は物体指数のみを有し、一方で分母は物体指数と原子指数との両方を有する。１つの物体あたりの原子の数に依存して、行の減少は、差示的な空間的負荷が形成されなければならない場合に、格納時間と実行時間との両方を節約する。

【０２４２】

行を減少させる手順は、残差ヤコビアンを計算する前に１回のみなされる必要があることに、注目のこと。残差を実行するオーバーヘッドは、形成されなければならない、より小さな行列ベクトル積の減少した費用によって、オフセットを超える。

【０２４３】

【数１２６】

$$\frac{\partial T(k)}{\partial r(p,s)}$$

を形成する際に、原子間力ヤコビアンの要素を節約する必要がないことに注目のこと。すなわち、各要素

【０２４４】

【数１２７】

$$\frac{\partial T(k,i)}{\partial r(p,s)}$$

は、その

【０２４５】

【数１２８】

$$\frac{\partial T(k)}{\partial r(p,s)}$$

に対する寄与が計算される間に利用可能であることのみが必要である。従って、大きな力ヤコビアンの１より多くの要素は、同時には必要とされない。

【０２４６】

代表的な原子ｒ（ｐ，ｓ）のグローバル座標は、ｒ（ｐ）（物体ｐの旋回のグローバル座標）、およびρ（ｐ，ｓ）（原子のステーション座標）の観点で計算される：

ｒ（ｑ，ｓ）　＝　ｒ（ｐ）＋$^{N}C^{k}$（ｐ）ρ（ｐ，ｓ）

微分することによって、本発明者らは、以下：

【０２４７】

【数１２９】

$$\frac{\partial r(p,s)}{\partial q_j} = \frac{\partial r(p)}{\partial q_j} - \left({}^{N}C^{k}(p)\tilde{\rho}(p,s)\right)\lambda(p)$$

を見出した（いくらかは、有限回転の運動から生じる）。

【０２４８】

この等式を、さらなる等式λ（ｐ，ｓ）＝λ（ｐ）で拡大し、そしてｗ（空間的微分）

【０２４９】

【数１３０】

$$w \triangleq \begin{bmatrix} \lambda \\ \dfrac{\partial r}{\partial q} \end{bmatrix}$$

を定義すると、その結果は、以下である：

【０２５０】

【数１３１】

$$w(p,s)=\begin{bmatrix}\lambda(p,s)\\ \dfrac{\partial r(p,s)}{\partial q_j}\end{bmatrix}=\phi(p,s)^{*}\begin{bmatrix}\lambda(p)\\ \dfrac{\partial r(p)}{\partial q_j}\end{bmatrix}$$

ベクトルλは、各物体に対する配向の変化の割合を発生させる場合に、中断され得る。これは、空間における各点においてポテンシャルが変化し得るという意味で、場の量である。純粋な回転（変形なし）を受ける剛体については、この回転によって影響を受ける各物体に対して、これは一定である。λを計算するための次数（Ｎ）アルゴリズムは、厳密には、後の節に記載される。

【０２５１】

上記等式をT_1（ｋ）についての等式に適用することによって、最後の換算公式：

【０２５２】

【数１３２】

$$T_1(k)=\sum_{p=1}^{nbod}\frac{\partial T(k)}{\partial w(p)}\frac{\partial w(p)}{\partial q_j}$$

$$\frac{\partial T(k)}{\partial w(p)}\triangleq\sum_{s\in p}\begin{bmatrix}\underline{0}_3 & \dfrac{\partial T(k)}{\partial r(p,s)}\\ \underline{0}_3 & \end{bmatrix}\phi^{*}(p,s)$$

が得られる。

換算ヤコビアン

【０２５３】

【数１３３】

$$\frac{\partial T(k)}{\partial w(p)}$$

の各要素は、ここで６×６行列である。還元ヤコビアンの要素は、物体の旋回における空間的負荷を、その系における別の旋回において起こる空間的微分（旋回の変位勾配へのカップリングの後）に関連付ける。

【０２５４】

行の減少は、各物体における全ての原子間力を強化し、その系における全ての原子の変位微分にカップリングされた、空間的負荷の微分を残した。列の減少は、全ての原子変位微分を強化し、空間的旋回微分にカップリングされた、空間的負荷微分を残した。従って、力ヤコビアンは、「ネイティブ」な形態に変わる。還元ヤコビアンを用いる作業は、空間的負荷微分の計算を、１つの物体あたりの原子の数のおよそ２乗で加速させる。このスピードアップは、１００倍以上に容易に近付き得る。

【０２５５】

この節は、例として静電力を使用して、原子レベルでどのように力ヤコビアンが構築されるかを示す。このヤコビアンは、差示的原子間力を差示的原子変位に関連付ける。界面収縮の概念を導入して、差示的空間負荷は、行を減少させた力ヤコビアンを介して、差示的原子変位に関連付けられることが示される。計算に対する別の改善は、最後に、完全に収縮したヤコビアンを生じ、これは、差示的食うか敵変位に対する差して器空間的負荷が繰るかも。

【０２５６】

ここで、力ヤコビアン

【０２５７】

【数１３４】

$$\frac{\partial T(k)}{\partial r(p)}$$

が、上記の空間的変位勾配にカップリングされなければならない。行列ベクトル多重度は、この工程を実行し、そしてその系における物体の数（原子の数ではない）と規模を合わせる。

第２項Ｔ$_2$（ｋ）は、以下：

【０２５８】

【数１３５】

$$T_2(k) = \sum_{i \in k} \frac{\partial \phi(k,i)}{\partial q_j} T(k,i)$$

によって与えられ、そしてもとの空間的負荷Ｔ（ｋ）および演算子φ（ｋ，ｉ）を含む：

【０２５９】

【数１３６】

$$\phi(k,i) = \begin{bmatrix} \underline{\underline{E}}_3 & \tilde{\rho}(k,i) \\ \underline{\underline{0}}_3 & \underline{\underline{E}}_3 \end{bmatrix} \begin{bmatrix} {}^N C^k(k) & \underline{\underline{0}}_3 \\ \underline{\underline{0}}_3 & {}^N C^k(k) \end{bmatrix}^*$$

従って

$$T_2(k) = \sum_{i \in k} \begin{bmatrix} {}^N dC^k(k) & \tilde{\rho}(k,i)\,{}^N dC^k(k) \\ \underline{\underline{0}}_3 & {}^N dC^k(k) \end{bmatrix}^* T(k,i)$$

ここで、

$${}^N dC^k(k) = {}^N dC^k(i)\ {}^i C^k(k) + {}^N C^k(i)\ {}^i dC^k(k)$$

は、基部物体から外向きに、繰り返し計算され、そして

【０２６０】

【数１３７】

$${}^i dC^k(k) = \frac{\partial {}^i C^k(k)}{\partial q(k)} dq(k) \quad k = 1, \ldots n$$

ここで、ｄｑ（ｋ）は、次の節で定義され、そして

【０２６１】

【数１３８】

$$\frac{\partial {}^i C^k(k)}{\partial q_k}$$

（物体間方向余弦行列の偏微分）は、物体を接続するジョイントの型の関数である：

【０２６２】

【数１３９】

ピン $\dfrac{\partial^i C^k(k)}{\partial q_k} = -\underline{\underline{E}}_3 \sin(q_k) + \tilde{\lambda}\cos(q_k) + \lambda\lambda^* \sin(q_k)$

スライダー： $\dfrac{\partial^i C^k(k)}{\partial q_k} = \underline{\underline{0}}_3$

ボールおよびフリー： $\dfrac{\partial^i C^k(k)}{\partial \varepsilon_1} = 2\begin{bmatrix} 0 & \varepsilon_2 & \varepsilon_3 \\ \varepsilon_2 & -2\varepsilon_1 & -\varepsilon_4 \\ \varepsilon_3 & \varepsilon_4 & -2\varepsilon_1 \end{bmatrix}$

$$\frac{\partial^i C^k(k)}{\partial \varepsilon_2} = 2\begin{bmatrix} -2\varepsilon_2 & \varepsilon_1 & \varepsilon_4 \\ \varepsilon_1 & 0 & \varepsilon_3 \\ -\varepsilon_4 & \varepsilon_3 & -2\varepsilon_2 \end{bmatrix}$$

$$\frac{\partial^i C^k(k)}{\partial \varepsilon_3} = 2\begin{bmatrix} -2\varepsilon_3 & -\varepsilon_4 & \varepsilon_1 \\ \varepsilon_4 & -2\varepsilon_3 & \varepsilon_2 \\ \varepsilon_1 & \varepsilon_2 & 0 \end{bmatrix}$$

$$\frac{\partial^i C^k(k)}{\partial \varepsilon_4} = 2\begin{bmatrix} 0 & -\varepsilon_3 & \varepsilon_2 \\ \varepsilon_3 & 0 & -\varepsilon_1 \\ -\varepsilon_2 & \varepsilon_1 & 0 \end{bmatrix}$$

次の節において、力ヤコビアンを、慣性力ヤコビアンと組み合わせて、残差ルーチンのヤコビアンを最終的に形成する。

【０２６３】

（残差ヤコビアン）

先行する節は、力ヤコビアン

【０２６４】

【数１４０】

$$\frac{\partial T(k)}{\partial w(p)}$$

を形成し、これは、空間的力の微分を形成するために、空間的変位勾配をカップリングしなければならない。この節は、空間的変位勾配の形成および残差ルーチンのヤコビアンの形成を記載する。

【０２６５】

ヤコビアン全体をカップリングするための残差アルゴリズムが、記載される。ヤコビアンアルゴリズムは、正確には、ヤコビアンを計算するように設定されない。自動差示ルーチンにおいて代表的であるように、これは、行列ベクトル積Ｊ$_{uq}$ｄｑ＋Ｊ$_{uu}$ｄｕを、ベクトルｄｑおよびｄｕについての任意の入力値について計算する。実際に、ヤコビアンを計算するために、「ヤコビアンルーチン」が効果的に、一連のブールベクトル（１つの入力が１に設定され、そして他の全ての入力場０に設定されたベクトル）を用いて繰り返し呼び出される。各呼び出しが、ヤコビアンの対応する列を発生させる。これらの工程のうちのいくつかは、残差形式方法または直接形式方法（順方向力学計算）について既に計算されたかまたは計算されているが、本明細書中では明瞭にするために再形成されていることに、注目のこと。

１．所定の（ｑ，ｕ）が、直接形式方法を使用して、

【０２６６】

【数１４１】

$$z \triangleq -M^{-1}\rho_u(q,u,0)$$

を計算する。また、

【０２６７】

【数１４２】
$$\dot{u} = z$$

を設定し、そしてＡ（ｋ）、次いでρ$_u$を再計算し、これは、
【０２６８】
【数１４３】
$$\hat{T}(k)$$

を再計算する。
２．収縮を実行して、完全に行および列が減少した力ヤコビアン
【０２６９】
【数１４４】
$$\frac{\partial T(k)}{\partial w(p)}$$

を、「界面構築」の節に記載されるように計算する：
【０２７０】
【数１４５】
$$\frac{\partial T}{\partial w} = \Phi \frac{\partial T}{\partial r} \Phi^*$$

。
以下の工程３～１０は、Ｊ$_{u\ q}$の列を満たすために使用される：
３．
【０２７１】
【数１４６】
$$\dot{q}$$

の項の解析的ヤコビアン分配
【０２７２】
【数１４７】
$$J_{qq} = \frac{\partial W(q)}{\partial q} u$$

$$J_{qu} = W$$

を、第１運動学的計算のために必要なルーチンと類似の結合ルーチンを使用して計算する。
４．ｑ（位置量の微分）および空間的勾配についての項を計算する：
先に記載した方法を使用して、特定の結合特異的場を満たした。これらの量は、$^{i}C^{k}$（ｋ）（物体内方向余弦行列）、
【０２７３】
【数１４７Ａ】
$$r^{Q_i Q_k}(k)$$

（各物体についてのスパンベクトル）、およびＨ（ｋ）（各物体の内側への（ｉｎｂｏａｒｄ）結合についてのジョイントマップ）からなった。これらの量の各々の微分についていうために、添え字ｄがその記号の名称に付加されて、この参照を一般的にする。従って、$^{i}dC^{k}$（ｋ）とは、方向余弦行列$^{i}C^{k}$（ｋ）の微分を意味する。
【０２７４】

各物体間方向余弦行列（および全ての結合特異的）量は、個々の結合の一般化座標のみに依存する。従って、${}^{i}dC^{k}$（ｋ）は、この微分が物体ｋについての座標のいずれかに対して微分がとられる場合にのみ、０ではない。試験中の反復を適切に「シード」するために、ベクトルｄｑが、このルーチンに入力される。ヤコビアン計算のために、本発明者らは、１つの入力を１に、そして他の全ての入力を０に設定した。次いで、必要とされた予備的な量を、代表的なループによって作製した：

【０２７５】

【数１４８】

$$ {}^{i}dC^{k}(k) = \frac{\partial {}^{i}C^{k}(k)}{\partial q(k)} dq(k) \quad k = 1,\ldots n $$

方向余弦行列の偏微分を、分析的に生成し、そして上記「界面収縮」の節で表すように表示する。これらの偏微分は、計算されているヤコビアンの特定の列に依存しない。ｄｑの特定の入力を１に設定し、そして残り全てを０に設定することは、シード量の正しいサブセットの消滅の効果を有する。

【０２７６】

【数１４９】

$$ \frac{\partial r^{\Omega_i \Omega_k}(k)}{\partial q_k} $$

（物体間スパンベクトルの偏微分）は、

【０２７７】

【数１５０】

$$ \frac{\partial r^{\Omega_i \Omega_k}(k)}{\partial q_k} = \lambda(k), \text{ スライダー} $$

$$ \frac{\partial r^{\Omega_i \Omega_k}(k)}{\partial q_k} = \underline{\underline{0}}_{3x4}, \text{ ボール} $$

$$ \frac{\partial r^{\Omega_i \Omega_k}(k)}{\partial q_k} = \underline{0}_3, \text{ ピン} $$

$$ \frac{\partial r^{\Omega_i \Omega_k}(k)}{\partial q_k} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{ フリー} $$

によって与えられる。

λ（ｋ）は、本明細書中で、親の物体に接続する、物体のスライド軸をいう。

【０２７８】

【数１５１】

$$ \frac{\partial H(k)}{\partial q_k} $$

（結合マップの偏微分）は、

【０２７９】

【数１５２】

$$\frac{\partial H(k)}{\partial q_k} = \underline{0}_6^*, \quad \text{ピン、スライダー}$$

$$\frac{\partial H(k)}{\partial q_k} = \begin{bmatrix} \underline{0}_3 & \underline{0}_3 \end{bmatrix}, \quad \text{ボール}$$

$$\frac{\partial H(k)}{\partial q_k} = \begin{bmatrix} \underline{0}_3 & \underline{0}_3 \\ \underline{0}_3 & \dfrac{\partial {}^iC^k(k)}{\partial q_k} \end{bmatrix}, \quad \text{フリー}$$

である。

上記偏微分の定義を用いて、反復は、以下のループでシードされる：

【０２８０】

【数１５３】

for k = 1 to *nbod*

$${}^i dC^k(k) = \frac{\partial {}^iC^k(k)}{\partial q_k} dq(k)$$

$$dr^{Q_i Q_k}(k) = \frac{\partial r^{Q_i Q_k}(k)}{\partial q_k} dq(k)$$

$$dH(k) = \frac{\partial H(k)}{\partial q_k} dq(k)$$

end

これらのループの実行の後に、全ての物体は、

【０２８１】

【数１５４】

${}^i dC^k(k)$, $dr^{Q_i Q_k}(k)$, および $dH(k)$

を有し、これらの物体間微分量は利用可能である。

【０２８２】

空間的変位勾配計算において必要とされる１つの新たな量もまた、計算される。これは、界面収縮の節からのλ（ｋ）であり、摂動した結合の外側の、各物体についての配向の変化の割合を発生させる、回転軸である。ここで、この変数は、記号ｄθ（ｋ）を与えられ、各物体についての差示的な回転軸は、以下である：

【０２８３】

【数１５５】

for k = 1 to nbod

$d\theta(k) = \lambda(k)\, dq(k), \quad pin$

$d\theta(k) = \underline{0}_3, \quad slider$

$d\theta(k) =$ not needed for *ball*, *free*

end

一連のオイラーパラメータに対する任意の摂動は、純粋な回転を生じないので、ヤコビアンの対応する列を計算する場合に、列の収縮は使用され得ない。行が減少した力ヤコビアンは、依然として使用される（そして使用されなければならない）。

【０２８４】

反復をシードした後に、

【０２８５】

【数１５６】

$^{N}dC^{k}(k)$, $dr^{OO_k}(k)$, $^{i}d\phi^{k}(k)$, $d\lambda(k)$

が計算される：

【０２８６】

【数１５７】

$$^{N}dC^{k}(1) = {}^{i}dC^{k}(1)$$

$$dr^{OO_k}(1) = dr^{Q_1Q_k}(1)$$

$$^{i}d\phi^{k}(1) = \underline{0}_6$$

$$d\lambda(1) = d\theta(1)$$

for $k = 2$ to $nbod$

$$^{N}dC^{k}(k) = {}^{N}dC^{k}(i)\,{}^{i}C^{k}(k) + {}^{N}C^{k}(i)\,{}^{i}dC^{k}(k)$$

$$dr^{OO_k}(k) = dr^{OO_k}(i) + {}^{N}dC^{k}(i)r^{Q_iQ_k}(k) + {}^{N}C^{k}(i)dr^{Q_iQ_k}(k)$$

$$^{i}d\phi^{k}(k) = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$d\lambda(k) = {}^{i}C^{k*}(k)d\lambda(i) + d\theta(k)$$

end

ここで

$$a = {}^{i}dC^{k}(k),\ b = d\tilde{r}^{Q_iQ_k}(k)\,{}^{i}C^{k}(k) + \tilde{r}^{Q_iQ_k}\,{}^{i}dC^{k}(k),$$

$$c = \underline{0}_3,\ d = {}^{i}dC^{k}(k)$$

５．ｑ（速度の微分）を計算する：

結合角の変化に起因する、結合速度の変化の割合を計算するループは、以下のプロセスで開始する：

【０２８７】

【数１５８】

for $k = 1$ to $nbod$

$$^{i}dv^{k}(k) = dH^{*}(k)u(k)$$

end

この量は、結合角の変化に起因する、結合速度の変化の割合である。明らかに、マップが座標依存性を含む結合についてのみ、この量は０ではない。自由結合について、一般化された速度は、結合の配向に依存する相対線形速度を生じる。

【０２８８】

$^{i}dv^{k}$（ｋ）、ｄＶ（ｋ）を計算した後に、各物体の空間的速度の微分が計算される。これは、以下のループによってなされる：

【０２８９】

【数１５９】

$$dV(1) = {}^{i}dv^{k}(1)$$

for $k = 2$ to $nbod$

$$dV(k) = {}^{i}d\phi^{k*}V(i) + {}^{i}\phi^{k*}dV(i) + {}^{i}dv^{k}(k)$$

end

６．力ヤコビアンを空間的変位勾配にカップリングして、T_1（ｋ）を計算する

【０２９０】

【数１６０】

for k = 1 to *nbod*

$$T_1(k) = \sum_{p=1}^{nbod} \frac{\partial T(k)}{\partial w(p)} \frac{\partial w(p)}{\partial q_j}$$

end

７．力ヤコビアンＴ$_2$（ｋ）の第二項を計算して、Ｔ$_1$（ｋ）に添える：

【０２９１】

【数１６１】

for k= 1 to *nbod*

$$dT(k) = T_1(k) + \sum_{i \in k} \begin{bmatrix} {}^N dC^k(k) & \tilde{\rho}(k,i)\,{}^N dC^k(k) \\ \underline{0}_3 & {}^N dC^k(k) \end{bmatrix}^* T(k,i)$$

end

８．ｑ（加速度関連項の微分）を計算する：

再度、このプロセスは、

【０２９２】

【数１６２】

$${}^i da^k(k) = dH^*(k)\dot{u}(k)$$

を計算するループで開始する：

【０２９３】

【数１６３】

for $k = 1$ to *nbod*

$${}^i da^k(k) = dH^*(k)\dot{u}(k)$$

end

これは、座標の変化に起因する、結合加速度の変化である。次いで、ｄＡ（ｋ）（各物体の空間的加速度の微分）が、以下のように計算される：

【０２９４】

【数１６４】

$$dA(1) = {}^i da^k(1)$$

$${}^i V^k(k) \rightarrow \begin{bmatrix} {}^i\omega^k(k) \\ {}^i v^{Q_k}(k) \end{bmatrix}, \quad {}^i dV^k(k) \rightarrow \begin{bmatrix} {}^i d\omega^k(k) \\ {}^i dv^{Q_k}(k) \end{bmatrix}$$

$$V(k) \rightarrow \begin{bmatrix} {}^N\omega^k(k) \\ {}^N v^{Q_k}(k) \end{bmatrix}, \quad dV(k) \rightarrow \begin{bmatrix} {}^N d\omega^k(k) \\ {}^N dv^{Q_k}(k) \end{bmatrix}$$

ここで、→は、６ベクトルが２つの３ベクトルに分解されることを表し、

Ｋ＝２～ｎｂｏｄについて、

【０２９５】

【数１６５】

$$dt1 \triangleq {}^{i}dC^{k*}(k)\left({}^{N}\omega^{k}(i)\times\left({}^{N}\omega^{k}(i)\times r^{Q_iQ_k}(k)\right)\right)+$$
$${}^{i}C^{k*}(k)\left(\begin{array}{l}\left({}^{N}d\omega^{k}(i)\times\left({}^{N}\omega^{k}(i)\times r^{Q_iQ_k}(k)\right)\right)+\\ \quad\left({}^{N}\omega^{k}(i)\times\left({}^{N}d\omega^{k}(i)\times r^{Q_iQ_k}(k)\right)\right)+\\ \left({}^{N}\omega^{k}(i)\times\left({}^{N}\omega^{k}(i)\times dr^{Q_iQ_k}(k)\right)\right)\end{array}\right)$$
$$da \triangleq {}^{i}d\phi^{k*}(k)A(i)+{}^{i}\phi^{k*}(k)dA(i)+\begin{bmatrix}0_3\\ dt1\end{bmatrix}$$
$$\omega j \triangleq {}^{i}C^{k*}(k)\,{}^{N}\omega^{k}(i)$$
$$d\omega j \triangleq {}^{i}dC^{k*}(k)\,{}^{N}\omega^{k}(i)+{}^{i}C^{k*}(k)\,{}^{N}d\omega^{k}(i)$$
$$dt2 \triangleq d\omega j\times{}^{i}\omega^{k}(k)+\omega j\times{}^{i}d\omega^{k}(k)$$
$$dt3 \triangleq 2d\omega j\times{}^{i}v^{Q_k}(k)+2\omega j\times{}^{i}dv^{Q_k}(k)$$
$$dA(k)=da+\begin{bmatrix}dt2\\ dt3\end{bmatrix}+{}^{i}da^{k}(k)$$
end

ここで

【０２９６】

【数１６６】

$\triangleq$

によって導入される記号は、ｄＡ（ｋ）の計算の後には必要とされない一時的な変数を意味する

空間的加速度の微分を計算した後に、

【０２９７】

【数１６７】

$$d\hat{T}(k)$$

（空間的慣性力の微分）の計算が実行される：

【０２９８】

【数１６８】

for $k=1$ to $nbod$

$$dv1 \triangleq {}^{N}d\omega^{k}(k)\times\underline{\underline{\mathbf{I}}}_{Q_k}(k)\cdot{}^{N}\omega^{k}+{}^{N}\omega^{k}(k)\times\underline{\underline{\mathbf{I}}}_{Q_k}(k)\cdot{}^{N}d\omega^{k}$$
$$dv2 \triangleq {}^{N}d\omega^{k}(k)\times\left({}^{N}\omega^{k}(k)\times\mathbf{p}(k)\right)+{}^{N}\omega^{k}(k)\times\left({}^{N}d\omega^{k}(k)\times\mathbf{p}(k)\right)$$
$$d\hat{T}(k)=M(k)dA(k)+\begin{bmatrix}dv1\\ dv2\end{bmatrix}-dT(k)$$
end

９．ｄρ（ｋ）（物体ｋに対する結合残差微分）を計算する：

【０２９９】

【数１６９】

for $k = nbod$ to 1

$d\rho(k) = dH(k)\hat{T}(k) + H(k)d\hat{T}(k) - d\sigma(k)$

$i = inb(k)$

if i >0

$d\hat{T}(i) = d\hat{T}(i) + {}^{i}d\phi^{k}(k)\hat{T}(k) + {}^{i}\phi^{k}(k)d\hat{T}(k)$

end

end

このルーチンを実行した後に、ｄρ（ｋ）に格納された値は、残差ヤコビアン

【０３００】

【数１７０】

$$\frac{\partial}{\partial q}\rho_u(q,u,z)$$

の新たな列である。

１０．質量行列を用いて、先行する工程の結果

【０３０１】

【数１７１】

$$\frac{\partial \rho}{\partial q}$$

を後退代入法に供して、所望の

【０３０２】

【数１７２】

$$\frac{\partial \dot{u}}{\partial q}:$$

$$\frac{\partial \dot{u}}{\partial q} = -M^{-1}\frac{\partial \rho}{\partial q}$$

を得る。

後退代入法の演算は、残差ベクトルを

【０３０３】

【数１７３】

$$\dot{u}$$

ベクトルに処理することによって、直接形式方法において達成される。第２運動学的計算は、ヤコビアン全体に対して１回のみ実施される必要がある。なぜなら、この後退代入法は、その状態の正常値においてなされるからである。実際に、第２運動ルーチンは、ｚを計算している間に、変数が依然として隠されるように、工程１において呼び出されなければならない。

【０３０４】

以下の工程１１～１３は、J_{uu}の列を満たすために使用される：

１１．ｕ（速度の微分）を計算する：

このルーチンは、入力ベクトルｄｕを採用し、そして${}^{i}ｄｖ^{k}$（ｋ）＝Ｈ＊（ｋ）ｄｕ（ｋ）を計算する。次いで、ｄＶ（ｋ）（各物体の空間的速度の微分）が計算される：

【０３０５】

【数１７４】

$dV(1) = {}^{i}dv^{k}(1)$
for $k = 2$ to $nbod$
$dV(k) = {}^{i}\phi^{k*}(k)dV(i) + {}^{i}dv^{k}(k)$
end

１２．速度により誘導される微分
【０３０６】
【数１７５】
$d\hat{T}(k)$

を計算する。本明細書中に提示されるように、このルーチンは、速度に依存する外部負荷がない場合に対して特殊化される。残っている項は、内力のみの変化に起因するものである。外部負荷の変化が存在する場合でさえも、以前のように、ｄＴ（ｋ）の寄与を含むために必要とされるのみである。
【０３０７】
【数１７６】

$$dA(1) = \begin{bmatrix} \underline{0}_3 \\ \underline{0}_3 \end{bmatrix}$$

for $k = 2$ to $nbod$

$$dt1 \triangleq {}^{i}C^{k*}(k)\left(\begin{array}{l}\left({}^{N}d\omega^{k}(i)\times\left({}^{N}\omega^{k}(i)\times r^{Q_iQ_k}(k)\right)\right)+\\ \left({}^{N}\omega^{k}(i)\times\left({}^{N}d\omega^{k}(i)\times r^{Q_iQ_k}(k)\right)\right)\end{array}\right)$$

$$da \triangleq {}^{i}\phi^{k*}(k)dA(i) + \begin{bmatrix} \underline{0}_3 \\ dt1 \end{bmatrix}$$

$$\omega j \triangleq {}^{i}C^{k*}(k)\,{}^{N}\omega^{k}(i)$$

$$d\omega j \triangleq {}^{i}C^{k*}(k)\,{}^{N}d\omega^{k}(i)$$

$$dt2 \triangleq d\omega j \times {}^{i}\omega^{k}(k) + \omega j \times {}^{i}d\omega^{k}(k)$$

$$dt3 \triangleq 2d\omega j \times {}^{i}v^{Q_k}(k) + 2\omega j \times {}^{i}dv^{Q_k}(k)$$

$$dA(k) = da + \begin{bmatrix} dt2 \\ dt3 \end{bmatrix}$$

end

空間的加速度の微分
【０３０８】
【数１７７】
$d\hat{T}(k)$

を計算した後に、空間的慣性力の微分が計算される：
【０３０９】
【数１７８】

for $k = 1$ to $nbod$

$$dv1 \triangleq {}^{N}d\omega^{k}(k) \times \underline{\underline{\mathbf{I}}}_{Q_k}(k) \cdot {}^{N}\omega^{k} + {}^{N}\omega^{k}(k) \times \underline{\underline{\mathbf{I}}}_{Q_k}(k) \cdot {}^{N}d\omega^{k}$$

$$dv2 \triangleq {}^{N}d\omega^{k}(k) \times \left({}^{N}\omega^{k}(k) \times \mathbf{p}(k)\right) + {}^{N}\omega^{k}(k) \times \left({}^{N}d\omega^{k}(k) \times \mathbf{p}(k)\right)$$

$$d\hat{T}(k) = M(k)dA(k) + \begin{bmatrix} dv1 \\ dv2 \end{bmatrix}$$

end

１３．ｄρ（ｋ）（物体ｋについての結合残差微分）を計算する：

【０３１０】

【数１７９】

for $k = nbod$ to 1

$$d\rho(k) = H(k)d\hat{T}(k) - d\sigma(k)$$

$$i = inb(k)$$

if i >0

$$d\hat{T}(i) = d\hat{T}(i) + {}^{i}\phi^{k}(k)d\hat{T}(k)$$

end

end

このルーチンを実行した後に、ｄρ（ｋ）に格納される値は、残差ヤコビアン

【０３１１】

【数１８０】

$$\frac{\partial}{\partial u}\rho_u(q,u,z)$$

の新たな列である。

１４．質量行列を用いて、先行する工程の結果

【０３１２】

【数１８１】

$$\frac{\partial \rho}{\partial u}$$

を後退代入法し、所望の

【０３１３】

【数１８２】

$$\frac{\partial \dot{u}}{\partial u}:$$

$$\frac{\partial \dot{u}}{\partial u} = -M^{-1}\frac{\partial \rho}{\partial u}$$

を得る。

【０３１４】

後退代入法の演算は、残差ベクトルを

【０３１５】

【数１８３】

$$\dot{u}$$

ベクトルに処理することによって、直接方法において達成される。第２運動学的計算は、１回のみ実施される必要がある。なぜなら、この後退代入法は、その状態の正常値におい

てなされるからである。実際に、第二運動ルーチンは、ｚを計算している間に、変数が依然として隠されるように、工程１において呼び出されなければならない。

【０３１６】

上記工程は、力がｑへの依存のみを有する限り、解析的ヤコビアンの計算を完了する。これは、全ての原子間力がポテンシャル関数から誘導される、古典的な状況に当てはまる。速度依存性の力（例えば、単純な粘度ダンピング）に当てはまるためには、上記工程のいくつかが、以下のように改変される必要がある：

上記工程２において、本発明者らはまた、収縮した速度ヤコビアン

【０３１７】

【数１８４】

$$\frac{\partial T(k)}{\partial \dot{r}(k)}$$

を計算する必要がある。これは、ブロックの対角であり、これもまた計算されなければならない。

【０３１８】

上記工程６において、Ｔ$_1$（ｋ）の計算は、収縮された速度ヤコビアンで増大されなければならない：

【０３１９】

【数１８５】

for k = 1 to *nbod*

$$T_1(k) = \sum_{p=1}^{nbod} \frac{\partial T(k)}{\partial w(p)} \frac{\partial w(p)}{\partial q_j} + \sum_{i \in k} \frac{\partial T(k)}{\partial \dot{r}(k,i)} \frac{\partial \dot{r}(k,i)}{\partial q_j}$$

end

ここで

$$\frac{\partial \dot{r}(k,i)}{\partial q_j} = {}^N dC^k(k)\left[{}^N v^{Q_k}(k) + {}^N \omega^k(k) \times \rho(k,i)\right] +$$
$${}^N C^k(k)\left[{}^N dv^{Q_k}(k) + {}^N d\omega^k(k) \times \rho(k,i)\right]$$

次いで、工程１１の後に工程が追加される。この工程は、工程１１ａと呼ばれる。この新たな工程は、ｄＴ（ｋ）を計算する：

【０３２０】

【数１８６】

$$dT(k) = \sum_{i \in k} \frac{\partial T(k)}{\partial \dot{r}(k,i)} \frac{\partial \dot{r}(k,i)}{\partial u_j}$$

ここで

$$\frac{\partial \dot{r}(k,i)}{\partial u_j} = {}^N C^k(k)\left[{}^N dv^{Q_k}(k) + {}^N d\omega^k(k) \times \rho(k,i)\right]$$

上記工程１２を実行する間、

【０３２１】

【数１８７】

$$d\hat{T}(k)$$

についての最後のループは、速度依存性の力の微分ｄＴ（ｋ）を除くことによって、改変される：

【０３２２】
【数１８８】

$$\text{for } k = 1 \text{ to } nbod$$

$$dv1 \triangleq {}^{N}d\omega^{k}(k) \times \underline{\underline{\mathbf{I}}}_{Q_k}(k) \cdot {}^{N}\omega^{k} + {}^{N}\omega^{k}(k) \times \underline{\underline{\mathbf{I}}}_{Q_k}(k) \cdot {}^{N}d\omega^{k}$$

$$dv2 \triangleq {}^{N}d\omega^{k}(k) \times \left({}^{N}\omega^{k}(k) \times \mathbf{p}(k)\right) + {}^{N}\omega^{k}(k) \times \left({}^{N}d\omega^{k}(k) \times \mathbf{p}(k)\right)$$

$$d\hat{T}(k) = M(k)dA(k) + \begin{bmatrix} dv1 \\ dv2 \end{bmatrix} - dT(k)$$

$$\text{end}$$

これらの工程の残りは、同じままである。
【０３２３】
図６は、上で詳細に記載された、解析的ヤコビアン方法の実行工程を要約する。
【０３２４】
図７は、例示的なＭＤ系についての解析的ヤコビアンの正確さに対する、数値ヤコビアンの正確さのプロットを示す。摂動が完全に選択された最良の場合において、数字ヤコビアンからの一般化座標（ｑ）および一般化速度（ｕ）についての正確さの数字（線１５２によって示される）は、依然として、線１５０によって示される解析的ヤコビアンのものの半分のみであった。
【０３２５】
（さらなる実施形態）
本発明は、上記に記載された実施例に加えて、多くの実施形態を含む。以下の列挙は、他の実施形態および適用を有する。
【０３２６】
・ヤコビアンに含まれる力の次数
ヤコビアンに含まれる力の任意の次数としては、次数（Ｎ）、次数（$Ｎ^{2}$）、次数（$Ｎ^{3}$）、および次数（$Ｎ^{4}$）が挙げられるが、これらに限定されない。次数（Ｎ）の力場の例は、次数（$Ｎ^{2}$）である直接方法ではなく、第一の多極展開方法を使用する、静電力場である（例えば、Ｇｒｅｅｎｇａａｒｄ，Ｔｈｅ Ｒａｐｉｄ Ｅｖａｌｕａｔｉｏｎ ｏｆ Ｐｏｔｅｎｔｉａｌ Ｆｉｅｌｄｓ ｉｎ Ｐａｒｔｉｃｌｅ Ｓｙｓｔｅｍｓ，Ｍａｓｓａｃｈｕｓｅｔｔｓ Ｉｎｓｔｉｔｕｔｅ ｏｆ Ｔｅｃｈｎｏｌｏｇｙ Ｄｉｓｓｅｒｔａｔｉｏｎ，１９８８を参照のこと）。
【０３２７】
・直接形式のための解析的ヤコビアン
支配的な等式が直接形式である場合、いわゆる「順方向力学」形態の等式が得られる。この形式において、これらの等式は、状態ベクトルおよび適用される効果を処理し、そしてその系においてモデル化された結合の各々において加速度を発生させる。
【０３２８】
【数１８９】

$$\dot{u} = M^{-1}\left(f\right)$$

次いで、ヤコビアンは、状態ベクトルの要素に関して、加速度の偏微分を表す。好ましい実施形態は、これらの偏微分を計算するための、いくつかのアルゴリズム方法を示す。これらの方法は正確であり、そして微分を形成するために数値的近似を利用しない。
【０３２９】
直接形式は、ヤコビアンの
【０３３０】
【数１９０】
$\dot{u}$

分配
【０３３１】
【数１９１】

$$J_{uq} = \frac{\partial\left(M^{-1}(f)\right)}{\partial q}$$

および

$$J_{uu} = \frac{\partial\left(M^{-1}(f)\right)}{\partial u}$$

を、
【０３３２】
【数１９２】
$\dot{u}$

関数を計算する関数の対アルゴリズム（ａｌｇｏｒｉｔｈｍｉｃ　ｃｏｕｎｔｅｒｐａｒｔ）を使用することによって、生じる。
【０３３３】
Ｍ－１ｆの計算が、次数（Ｎ）の浮動小数点演算（ｆｌｏｐｓ）において、質量行列の演算子の逆を利用することによって、達成される：
【０３３４】
【数１９３】

$$M^{-1} = \left[I - H\Psi K\right] D^{-1} \left[I - H\Psi K\right]^{T}$$

ここで、全てのブロック行列は、第２運動学的計算において先に規定され、そして各因子は、次数（Ｎ）ｆｌｏｐｓにおいてｎベクトルに適用され得る演算子を表す。
【０３３５】
【数１９４】

$$\dot{u} = \left[I - H\Psi K\right] D^{-1} \left[I - H\Psi K\right]^{T} f$$

であるので、連鎖法則から、
【０３３６】
【数１９５】

$$\begin{aligned} J_{uq} = {} & \left[I - H\Psi K\right] D^{-1} \left[I - H\Psi K\right]^{T} \partial f / \partial q + \\ & \partial\left(\left[I - H\Psi K\right]\right) / \partial q \; D^{-1} \left[I - H\Psi K\right]^{T} f + \\ & \left[I - H\Psi K\right] \; \partial\left(D^{-1}\right) / \partial q \left[I - H\Psi K\right]^{T} f + \\ & \left[I - H\Psi K\right] D^{-1} \; \partial\left(\left[I - H\Psi K\right]^{T}\right) / \partial q \; f \end{aligned}$$

であり、Ｊｕｕについても同様である。
【０３３７】
従って、本発明は、上記項の各々を閉じた形態で計算するアルゴリズムを作成する際の、直接形式の等式のために使用され得る。
【０３３８】
従って、本発明は、多くの利点を提供する。解析的ヤコビアンは、よりずっと正確である（有効数字の桁の２倍）。直接形式ではなく残差形式からのカップリングは、よりずっと効率的である。「原子の数」から「物体の数」への、行および列の「収縮」は、力ヤコビアン行列の大きさを減少させる。ヤコビアン計算は、各列が摂動した場合には過剰に高い次数であるよりむしろ、力の計算と同じ次数である。従って、力が次数（Ｎ^{3}）の演算子

で計算される場合、例えば、数値ヤコビアンは、次数（N^4）を必要とし、一方で解析的ヤコビアンは、次数（N^3）の演算子のみを必要とする。ヤコビアン計算におけるループ構造の範囲を制御することによって、計算はなおさらに減少され得る（外側物体についてちょうど計算する）。

【０３３９】

従って、上述のものは、本発明の実施形態の完全な説明であるが、種々の改変、変更および均等物が、作製および使用され得ることが、明らかであるはずである。従って、上記説明は、添付の特許請求の範囲の組み合わせおよび境界によって規定される、本発明の範囲を限定するとはみなされるべきではない。

【図面の簡単な説明】

【図１】

図１は、本発明に従うソフトウエアシステムアーキテクチャのブロックモジュール図である。

【図２】

図２は、本発明に従う分子モデルの多体系のツリー構造を示す。

【図３】

図３は、図２の多体系の基準配置を示す。

【図４Ａ】

図４Ａは、図２の多体系における２つの物体間のスライダージョイントを示す。

【図４Ｂ】

図４Ｂは、図２の多体系における２つの物体間のピンジョイントを示す。

【図４Ｃ】

図４Ｃは、図２の多体系における２つの物体間のボールジョイントを示す。

【図５】

図５は、解析的ヤコビアン計算について使用される残差形式法についての一般的な計算工程を示す。

【図６】

図６は、解析的ヤコビアン計算のための一般的計算工程を要約するチャートである。

【図７】

図７は、数値的ヤコビアンを上回る、解析的ヤコビアンの有効数字のプロットである。

【図１】

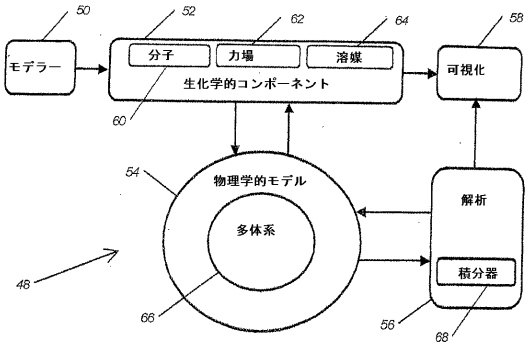

Fig. 1

【図５】

ρ_q および ρ_u を計算するための残差形式法	$\dot{q}$ および $\dot{u}$ を計算するための直接形式法
１．第１運動学的計算および運動学的残差 $\rho_q(k)$ を計算する ２．各物体に関する空間負荷収支 $\hat{f}(k)$ を生成する ３．動力学的残差 $\rho_u(k)$ を計算	１．ジョイント特異的ルーチンを使用して $\dot{q}$ を生成する ２．$\dot{u}=0$ を用いて第１運動学的計算を実行する ３．残差 ρ_u を生成 かつ $\rho_u=-\rho_u$ を否認 ４．第２運動学的計算 ５．前進動力学 を使用して $\dot{u}$ を計算

方法の比較

Fig. 5

【図６】

解析的ヤコビアン法

１．運動学的ルーチンの解析的 ヤコビアンを計算：

$$J_{qq}=\frac{\partial(Wu)}{\partial q} \quad \text{および} \quad J_{qw}=W$$

２．直接法を使用して $z\triangleq -M^{-1}\rho_u(q,u,0)$ を計算

３．動力学的残差ルーチンの 解析的ヤコビアンを計算：

$$\frac{\partial}{\partial q}\rho_u(q,u,z) \text{ および } \frac{\partial}{\partial u}\rho_u(q,u,z).$$

４．第２運動学的工程由来の z についてこの動力学的ルーチンの解析 的ヤコビアンについての後退 代入法をおこなう：

$$J_{uq}=\frac{\partial \dot{u}}{\partial q}=-M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial q} \quad \text{および}$$

$$J_{uu}=\frac{\partial \dot{u}}{\partial u}=-M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial u}$$

Fig. 6

【図７】

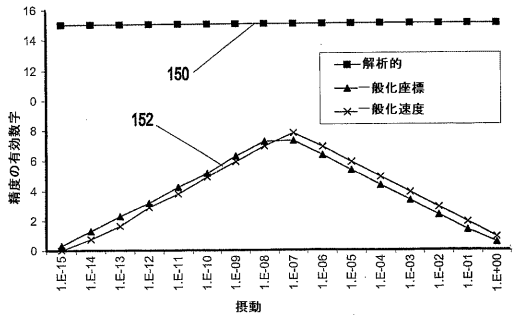

Fig. 7

【国際公開パンフレット】

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 August 2002 (08.08.2002)

PCT

(10) International Publication Number
WO 02/061662 A1

(51) International Patent Classification⁷: G06F 19/00

(21) International Application Number: PCT/US01/51360

(22) International Filing Date:
2 November 2001 (02.11.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/245,731	2 November 2000 (02.11.2000)	US
60/245,688	2 November 2000 (02.11.2000)	US
60/245,730	2 November 2000 (02.11.2000)	US
60/245,734	2 November 2000 (02.11.2000)	US

(71) Applicant *(for all designated States except US)*: **PROTEIN MECHANICS, INC.** [US/US]; 278 Hope Street, Suite C, Mountain View, CA 94041 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **ROSENTHAL, Dan, E.** [US/US]; 718 Edge Lane, Los Altos, CA 94024 (US).

(74) Agents: **LIEBESCHUETZ, Joe** et al.; Townsend and Townsend and Crew LLP, Two Embarcadero Center, 8th floor, San Francisco, CA 94111 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 02/061662 A1

(54) Title: METHOD FOR ANALYTICAL JACOBIAN COMPUTATION IN MOLECULAR MODELING

(57) Abstract: A method for obtaining analytic Jacobians used in implicit integration methods in the computations for the dynamics of a physical system. With this method, the Jacobian with at least twice the number of digits of accuracy as a numerical Jacobian can be computed. This also results in the implicit integration method being more efficient because a smaller number of iterations are required to solve the nonlinear stage equations of the equations of motion, as well as the ability to take larger timesteps. This speedup in computation is very useful in molecular modeling.

METHOD FOR ANALYTICAL JACOBIAN COMPUTATION IN MOLECULAR MODELING

CROSS-REFERENCES TO RELATED APPLICATIONS

This application is entitled to the benefit of the priority filing dates of Provisional Patent Application No. 60/245,730, filed 2000 Nov. 2; and in addition, No. 60/245,688, filed 2000 Nov. 2; No. 60/245,731, filed 2000 Nov. 2; and No. 60/245,734, filed 2000 Nov. 2; all of which are hereby incorporated by reference.

BACKGROUND OF THE INVENTION

The present invention is related to the field of molecular modeling and, more particularly, to computer-implemented methods for the dynamic modeling and static analysis of large molecules.

The field of molecular modeling has successfully simulated the motion (molecular dynamics or (MD)) and determined energy minima or rest states (static analysis) of many complex molecular systems by computers. Typical molecular modeling applications have included enzyme-ligand docking, molecular diffusion, reaction pathways, phase transitions, and protein folding studies. Researchers in the biological sciences and the pharmaceutical, polymer, and chemical industries are beginning to use these techniques to understand the nature of chemical processes in complex molecules and to design new drugs and materials accordingly. Naturally, the acceptance of these tools is based on several factors, including the accuracy of the results in representing reality, the size and complexity of the molecular systems that can be modeled, and the speed by which the solutions are obtained. Accuracy of many computations has been compared to experiment and generally found to be adequate within specified bounds. However, the use of these tools in the prior art has required enormous computing power to model molecules or molecular systems of even modest size to obtain molecular time histories of sufficient length to be useful.

There are two sources of computational complexity for molecular modeling tasks involving time integration:

1. The particular molecular model which is used to describe the locations, velocities and mass properties of the constituent atoms, the inter-atomic forces

between them, and the interactions between the atoms and their surrounding environment; and

2. The particular numerical method used to advance the model through time. Time is advanced repeatedly by very short intervals, called timesteps, until a final time has been reached.

Substantial work has been completed in reducing the computational load for molecular models, such as the reduction of model complexity by constraining higher order modes with rigid body assumptions, simplifying the model with rigid or flexible substructuring, Order(*N*) dynamics, efficient implicit solvent models, and multipole methods for the force field models (see, for example, U.S. Patent No. 5,424,963 on the commercial MBO(N)D software package). Co-pending applications, U.S. Appln. No. _____, entitled "METHOD FOR LARGE TIMESTEPS IN MOLECULAR MODELING," and U.S. Appln. No._____, entitled "METHOD FOR RESIDUAL FORM IN MOLECULAR MODELING," both of which are filed of even date, claim priority from the previously cited provisional patent applications and which are assigned to the present assignee, and which are incorporated by reference herein, describe further improvements in molecular models and numerical methods.

The primary reason molecular simulations are so slow is that current numerical methods require very small timesteps, typically between 1 and 10 femtoseconds (10^{-15} to 10^{-14} seconds). Each timestep taken requires the computation of a new *state* (position and motion for each atom) of the particular molecular model, and then computation of the new set of forces resulting from the new state. For example, molecular dynamics simulations of the complex behavior of large molecules, such as the folding of a protein, typically need to cover a time span from at least a microsecond up to several seconds or even minutes. With techniques currently in common use, this results in the requirement to take 10^{9} to 10^{16} timesteps in the computer simulation. The per-step computations of the state, and especially the forces, grow very expensive as the problem size increases. Even with the fastest computers available today, months, years or even centuries of computer time are required to solve such problems even for systems of modest size.

Heretofore, it has been widely believed by molecular dynamicists that these small timesteps are an inevitable requirement of the need to maintain accuracy in the presence of the very high frequencies to be found in vibrations of molecular

bonds. For example, see Leach, *Molecular Modelling Principles and Applications*, 1996, p. 328; Berendsen, in *Computational Molecular Dynamics: Challenges, Methods, Ideas* Deuflhard et al. (ed.), Springer, 1999, pg. 18; Rapaport, *The Art of Molecular Dynamics Simulation*. Cambridge, 1995, reprinted with corrections 1998, p. 57; and U.S. Patent No. 5,424,963.

This common-sense belief is incorrect, however. The computer science sub-discipline of numerical analysis has produced an extensive theory of numerical integration for problems in which high frequencies exist but are of little interest. These problems are termed "stiff" problems (see, for example, Hairer and Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems, 2nd ed.*, Springer, 1996). In these cases, it is the *stability* of the integration method, not the required solution *accuracy*, which limits the timestep. Integration methods exist which have *unconditional* stability, meaning that in theory they can take arbitrarily large timesteps. Such methods have a mathematical property called "L-stability." Only so-called "implicit" integration methods *can* be L-stable, but very few implicit integration methods actually *are* L-stable. Previously cited co-pending U.S. Patent Appln. No. _____, entitled "METHOD FOR LARGE TIMESTEPS IN MOLECULAR MODELING," covers the use of implicit and in particular L-stable integration methods.

The present invention covers another critical aspect of allowing the implicit and in particular L-stable integrators to take large timesteps, namely, more accurate methods for computing required components of the implicit integration methods called "Jacobians".

But the same problem of high frequency molecular vibration for MD simulations causes problems for the calculation of Jacobians. For example, the repulsive van der Waal's forces that are generated as the electron orbitals of two atoms begin to overlap must be accounted for in a molecule. This quantum mechanical effect is often approximated by the so-called Lennard-Jones potential (Rapaport, *op. cit.*), which models the repulsive force as being proportional to $1/r^{13}$, where r is the distance between the centers of the atoms. This is an extreme stiff interatomic force which is characteristic of molecular dynamics (MD) simulations and poses particular difficulty for any numerical integration scheme used to advance time during the simulation. As a result and as mentioned previously, most prior art MD

integration schemes have timesteps limited by the high frequencies generated by these extremely stiff repulsive forces. And in particular, the stiffness of the atomic forces greatly magnify the difficulty of forming certain required Jacobian matrices. Such Jacobian matrices are a necessary ingredient of any stable implicit integration scheme, such as described in the immediately cited co-pending application.

All general-purpose simulation codes provide routines to compute Jacobians numerically as follows. For a given equation to integrate $\dot{y} = f(y,t)$, the desired Jacobian is $J = \partial f / \partial y$ and is numerically computed:

$$J = \frac{\Delta f}{\Delta y}$$

where

$$\Delta f = f|_{y=y_2} - f|_{y=y_1}$$
$$\Delta y = y_2 - y_1$$

Note that the perturbation Δy has to be selected to give a good result and may be difficult to choose, especially for stiff systems. In contrast, analytic Jacobians are computed by solving directly, or in this case algorithmically, for the equation of the desired derivatives.

Linearized models are regularly produced analytically for simple systems. Such linearization is usually performed around an equilibrium solution. It is common in such packages as ACSL (Advanced Continuous Simulation Language), (*ACSL Reference Guide*, Mitchell Gauthier and Associates, 1996), and SPICE (a circuit simulation package), (R. Kielkowski, *Inside SPICE*, McGraw-Hill, 1998) to perform equilibrium analysis followed by linearization. Such linearization is performed numerically.

In contrast, the Jacobian of the present invention represents linearization about an instantaneous solution of the differential equations (non-equilibrium) and is generated analytically. It should be noted that another prior art approach to generating analytic Jacobians is to use automated differentiation tools such as ADIFOR (Automatic Differentiation of Fortran) (C. Bischof, et. al., *ADIFOR 2.0 Users' Guide*, Argonne National Laboratory, 1998) that can symbolically differentiate arbitrary equations. These tools could be used to implement this invention in practice. However, the structure of the equations must be exploited properly to minimize the computations, to avoid errors due to round off and term

cancellations, and to avoid "equation swell" which could limit the size of problems solved.

Rather, the present invention allows for the calculation of analytic Jacobians for the effective implicit integration, including L-stable integrators, of the equations of motion of molecular models.

SUMMARY OF THE INVENTION

The present invention provides for a method of modeling the behavior of a molecule. The method has the steps of selecting a torsion angle, rigid multibody model for said molecule, the model having equations of motion; selecting an implicit integrator; and generating an analytic Jacobian for the implicit integrator to integrate the equations of motion so as to obtain calculations of the behavior of the molecule. The analytic Jacobian J is derived from an analytic Jacobian of the Residual Form of the equations of motion and is described as:

$$J = \begin{pmatrix} \frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial u} \\ \frac{\partial \dot{u}}{\partial q} & \frac{\partial \dot{u}}{\partial u} \end{pmatrix} \triangleq \begin{pmatrix} J_{qq} & J_{qu} \\ J_{uq} & J_{uu} \end{pmatrix}; \text{ and}$$

$$J_{qq} = \frac{\partial \dot{q}}{\partial q} = \frac{\partial (Wu)}{\partial q} \text{ and } J_{qu} = \frac{\partial \dot{q}}{\partial u} = W$$

$$J_{uq} = \frac{\partial \dot{u}}{\partial q} = -M^{-1} \frac{\partial \rho_u(q,u,z)}{\partial q} \text{ and } J_{uu} = \frac{\partial \dot{u}}{\partial u} = -M^{-1} \frac{\partial \rho_u(q,u,z)}{\partial u}$$

where q are the generalized coordinates, u are the generalized speeds, W is a joint map matrix and M is the mass matrix and ρ_u is the dynamic residual of the equations of motion, and z is $-M^{-1}\rho_u(q,u,0)$. The method can also be used for any physical system which can be modeled by a torsion angle, rigid multibody system.

The present invention also provides for the computer code for simulating the behavior of a molecule, or any physical system, which can be modeled by a torsion angle, rigid multibody system. A module in the computer code with an implicit integrator includes the analytic Jacobian as described above.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a representational block module diagram of the software system architecture in accordance with the present invention;

Fig. 2 illustrates the tree structure of the multibody system of the molecular model according to the present invention;

Fig. 3 illustrates the reference configuration of the Fig. 2 multibody system;

Fig. 4A illustrate a sliding joint between two bodies of the Fig. 2 multibody system; Fig. 4B illustrate a pin joint between two bodies of the Fig. 2 multibody system; Fig. 4C illustrate a ball joint between two bodies of the Fig. 2 multibody system;

Fig. 5 summarizes general computational steps for the Residual Form method and Direct Form methods used for the analytic Jacobian computation;

Fig. 6 is a chart which summarizes the general computational steps for the analytic Jacobian;

Fig. 7 is a plot of digits of accuracy versus perturbation to show the accuracy of analytic Jacobian over the numerical Jacobian.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

GENERAL DESCRIPTION OF THE PRESENT INVENTION

The numerical method used to advance time in the simulation of a modeled molecular system is called the integrator, or integration method. The integration proceeds by discretizing the governing equations of motion of the molecular system. In the case of an implicit integrator, this step results in a set of coupled nonlinear algebraic equations (the "stage" equations) and the solution of these equations yields the state vector for the molecular system at the next time step.

The present invention aids the solution of the stage equations. Because the atomic forces vary so rapidly over short distances, it is difficult to propagate atomic trajectories accurately. Small errors in the atomic positions lead to gross errors in the satisfaction of the stage equations. Since the Jacobian is used solve the stage equations iteratively, an inaccurate Jacobian leads to trial solution that are far from the correct solution. This leads to retraction of trial solutions and hinders the simulation.

Numerical Jacobians may be correct in only half their significant digits. An analytical Jacobian will often be correct in all but the last digit. The benefit of this result is that the integrator can take far fewer time steps to simulate the specified interval, allowing full exploitation of the theoretical stability of the integration method.

The ease or difficulty in producing the Jacobian depends crucially upon the formulation used to produce the governing equations. For instance, global Cartesian formulations produce equations with very limited explicit coordinate dependence. Producing an analytic Jacobian for such a formulation is well known. On the other hand, using internal coordinates (in which each molecular subunit's location is expressed relative to an earlier subunit's location) as independent variables has great benefits. This method is most valuable for any molecule modeled with any choice of internal coordinates, and in particular when used with protein models or other polymeric molecules using "torsion angles" between the residues as internal coordinates. An algorithm for producing an analytic Jacobian for a system formulated in torsion angles is extremely difficult to devise. However, the present invention achieves this task.

The present invention addresses a seemingly intractable problem: production of the analytic Jacobian for a formulation using internal coordinates, and specifically torsion angles, which is generally thought to be impractical. In addition to being more accurate than numerical Jacobians, the analytic Jacobians are also cheaper (in computing power) to produce. The present invention also recognizes a key result that the Jacobian of the state derivatives can be computed by applying a matrix inverse to the Jacobian of the computed torque method. This result allows significant savings in computer time and effort to construct the algorithm.

Furthermore, this method of producing analytic Jacobians for multibody system formulations using torsion angle, internal coordinates has not been seen in the general MBS literature. The present invention can be used for any torsion angle MBS formulation, which can be applied to many other disciplines besides molecular simulations, including, but not necessarily limited to, mechanical systems, robotic systems, vehicle systems, or any other system which could be described as a set of hinge-connected rigid bodies.

OVERVIEW OF DESCRIPTION

The preferred embodiment is divided into several sections. The first set of sections describes the MD simulation architecture, the multibody system (MBS) definitions, and Residual Form of the MBS equations for the subsequent descriptions. The next set of sections discusses the definition of the Jacobian, its role in the implicit integration method, and the computation of the analytic Jacobian using the Residual Form. Also shown is the superior accuracy and performance of the analytic Jacobian vs. the numerical Jacobian. Further efficiencies in the computation of the analytic Jacobian are discussed, specifically, exploiting the rigid body MBS to "contract" the size of the Jacobian matrix, and exploiting the topological structure of the MBS to eliminate unnecessary computations.

To solve ordinary differential equations (ODEs), most of the prior art have used equations expressed in the Direct Form, i.e., $\dot{y} = f(y,t)$ (where $\dot{y}$ means $\frac{dy}{dt}$). The equations of motion for a biomolecular system can be cast into this form (and called the Direct Form). In molecular modeling, all prior art known to the present inventors have used the Direct Form. That is, $\dot{q} = Wu$, $\dot{u} = M^{-1}f$, where q and u are generalized coordinates and speeds respectively, so that conventional ODE solution methods can be applied. However, this requires a matrix inversion of M (representing the mass of the system) at a cost of Order(N) to Order(N^3) floating point operations (depending on algorithm used, where N is the number of degrees of freedom in the system), since the natural form of the equations gives rise to inertial coupling between the derivatives of the generalized speeds. That is, the equations are most naturally produced in the form $\dot{q} - Wu = 0$, $M\dot{u} - f = 0$, where M, the mass matrix, depends explicitly upon the generalized coordinates q, i.e., $M = M(q)$. This fact requires forming and effectively factoring the mass matrix each time the state derivatives are needed by the integrator in integrating the equations of motion over time. The generalized joint map matrix W is block diagonal and, although it is also dependent on the coordinates $W = W(q)$, it does not have a significant computational cost.

In accordance with the present invention, a method for the solution of the equations of a molecular system is expressed in Residual Form to bypass the customary step of producing the state derivatives directly. The Residual Form method has the following steps:

1) Discretization of the solution variables. The specific form of discretization is dictated by the particular implicit integration method used to advance the molecular model in time. Implicit integration follows from the Residual Form. Implicit integration, especially L-stable integrators and other highly stable integrators, such as implicit Euler, Radau5, SDIRK3, SDIRK4, other implicit Runge-Kutta methods, and DASSL or other implicit multistep methods, also provide other advantages for molecular modeling. See, for example, the above-cited U.S. Patent Appln. No. _____, entitled "METHOD FOR LARGE TIMESTEPS IN MOLECULAR MODELING," filed of even date. As a particularly simple example, when used with implicit Euler integration, the discretization is as follows:

$$\dot{q} = (q_n - q_{n-1})/h,\ \dot{u} = (u_n - u_{n-1})/h$$

where h is the timestep.

2) Substitution into the residual equations:

$$\begin{pmatrix} \rho_q \\ \rho_u \end{pmatrix} = \begin{pmatrix} \dot{q} - W(q)u \\ M(q)\dot{u} - f(t,q,u) \end{pmatrix}$$

3) Solution of the resulting nonlinear algebraic equations $\begin{pmatrix} \rho_q \\ \rho_u \end{pmatrix} = 0$ for q_n and u_n.

The kinematic residual ρ_q compares an estimated $\dot{q}$ generated from the implicit integrator to the derivatives computed by the routines for determining the joints of the molecular model, which is described in greater detail below. The second row of the residual is ρ_u, the dynamic residual, which determines the degree to which an estimated $\dot{u}$ satisfies the equations of motion.

The system mass matrix M and the so-called 'bias-free hinge torque' f are both state dependent. The bias-free hinge torque is generated by the dynamic residual routine when the calculated $\dot{u}$ vector passed to the residual routine is zero. In general, the hinge accelerations are a response to applied forces, joint torques, and motion-induced effects (such as Coriolis and centrifugal forces.) If the system were at rest, and subjected only to joint torques, it would be considered in a bias-free state. The real system with its actual inputs can be reduced to a bias-free state by computing

a set of joint torques equivalent to the biased inputs. Both sets produce the same hinge accelerations.

The preferred embodiment of the Residual Form is shown for an Order(N) torsion-angle, rigid-body for the molecular model. The following sections develop the molecular model from basic definitions and show how the model is used to compute the motion of the model. First, the overall computer code architecture for the molecular model simulation is described. Then an Order(N) torsion-angle, rigid multibody system is derived, along with notation used, the reference configuration, the definitions of the joints between the bodies, generalized coordinates, and generalized speeds. This approach for dynamics is similar to that used by T. R. Kane (*Dynamics*, 3rd ed., 1978.)

Finally, the preferred embodiment of the analytic Jacobians using the results of the Residual Form is developed.

MOLECULAR DYNAMICS SIMULATION ARCHITECTURE

The general system architecture 48 of the software and some of its processes for modeling molecules in accordance with the present invention are illustrated in Fig. 1. Each large rectangular block represents a software module and arrows represents information which passes between the software modules. The software system architecture has a modeler module 50, a biochem components module 52, a physical model module 54, an analysis module 56 and a visualization module 58. The details of some of these modules are described below; other modules are available to the public.

The modeler module 50 provides an interface for the user to enter the physical parameters which define a particular molecular system. The interface may have a graphical or data file input (or both). The biochem components module 52 translates the modeler input for a particular mathematical model of the molecular system and is divided into translation submodules 60, 62 and 64 for mathematical modeling the molecule(s), the force fields and the solvent respectively of the system being modeled. There are several modeler and biochem components modules available including, for example, Tinker (Jay Ponder, TINKER User's Guide, Version 3.8, October 2000, Washington University, St. Louis, MO).

With the translated physical parameters from the biochem components module 52, the physical model module 54 defines the molecular system

mathematically. At the core of the module 54 is a multibody system submodule 66. The analysis module 56, which communicates with the physical model module 54 and the visualization module 58, provides solutions to the computational models of the molecular systems defined by the physical model module 54. The analysis module 56 consists of a set of integrator submodules 68 which integrate the differential equations of the physical model module 54. The integrator submodules 68 advance the molecular system through time and also provide for static analyses used in determining the minimum energy configuration of the molecular system. The analysis module 56 and its integrator submodules 68 contain most of the subject matter of the present invention and are described in detail below.

The visualization module 58 receives input information from the biochem components module 52 and the analysis module 56 to provide the user with a three-dimensional graphical representation of the molecular system and the solutions obtained for the molecular system. Many visualization modules are presently available, an example being VMD (A. Dalke, *et al.*, VMD User's Guide, Version 1.5, June 2000, Theoretical Biophysics Group, University of Illinois, Urbana, Illinois).

The described software code is run on conventional personal computers, such as PCs with Pentium III or Pentium IV microprocessors manufactured by Intel Corporation of Santa Clara, California. This contrasts with many current efforts in molecular modeling which use supercomputers to perform calculations. Of course, further speed improvements can be obtained by running the described software on faster computers.

MOLECULAR MODEL AND MULTIBODY SYSTEM DESCRIPTION

The integrators described below in the submodule 68 operate upon a set of equations which describe the motion of the molecular model in terms of a multibody system (MBS). To aid the computation of the integration methods described in detail below, a torsion angle, rigid body model is used to describe the subject molecule system, in accordance with the present invention. Internal coordinates (selected generalized coordinates and speeds) are used to describe the states of the molecule.

The MBS is an abstraction of the atoms and effectively rigid bonds that make up the molecular system being modeled and is selected to simplify the actual physical system, the molecule in its environment, without losing the features

important to the problem being addressed by the simulation. With respect to the general system architecture illustrated in Fig. 1, the MBS does not include the electrostatic charge or other energetic interactions between atoms nor the model of the solvent in which the molecules are immersed. The force fields are modeled in the submodule 62 and the solvent in the submodule 64 in the biochem components module 52.

Fig. 2 illustrates the tree structure of the MBS of a subject molecule. The basic abstraction of the MBS is that of one or more collections of hinge-connected rigid bodies 170. A rigid body is a mathematical abstraction of a physical body in which all the particles making up the body have fixed positions relative to each other. No flexing or other relative motion is allowed. A hinge connection is a mathematical abstraction that defines the allowable relative motion between two rigid bodies. Examples of these rigid bodies and hinge connections are described below.

One or more of the bodies, called base bodies 172, have special status in that their kinematics are referenced directly to a reference point on ground 174. The system graph is one or more "trees". An important property of a tree is that the path from any body to any other body is unique, i.e., the graph contains no loops. The bodies in the tree are n in number (the base has the label 1). The bodies in the tree are assigned a regular labeling, which means that the body labels never decrease on any path from the base body to any leaf body 176. A leaf body is one that is connected to only a single other body. A regular labeling can be achieved by assigning the label n to one of the leaf bodies 178 (there must be at least one). If this body is removed from the graph, the tree now has $n-1$ bodies. The label $n-1$ is then assigned to one of its leaf bodies 180, and the process is repeated until all the bodies have been labeled. This is also done for any remaining trees in the system.

To help maintain the relationship between the bodies, an integer function is used to record the inboard body for each body of the system. The inboard body for each base is ground and i, the parent or inboard body 182 for body k 184, is referred to as $i = inb(k)$. Additionally, the symbol N refers to the inertial, or ground frame 174. A superscript O refers to the ground origin (0,0,0).

The symbol for the vector from one point to another contains the name of the two points. Thus, r^{PQ} is the vector from the point P to point Q. A vector representing the velocity of a point in a reference frame contains the name of the point

and the reference frame: ${}^{N}v^{P}$. Certain symbols to be introduced later relate two reference frames. In this case, the symbol contains the name of two frames. Thus, ${}^{i}C^{k}$ is the direction cosine matrix for the orientation of frame k in frame i. This symbol refers to the direction cosine matrix for a typical body in its parent frame. Thus, ${}^{i}C^{k}(j)$ indicates the actual body j in question. The left and right superscripts do not change with the body index. This is also true for the other symbols. An asterisk indicates the transpose: $H^{*}(k)$, for example. A tilde over a vector indicates a 3 by 3 skew-symmetric cross product matrix: $\tilde{v}w \triangleq v \times w$. $\underline{\underline{E}}_i$ is an i by i identity matrix., and $\underline{0}_i$ is a zero vector of length i and $\underline{\underline{0}}_i$ is an i by i zero matrix.

Rigid Bodies of the Model

Fig. 3 illustrates the reference configuration 190 of a sample "tree" of the MBS. More than one tree is allowed. A point of each body is designated as Q, its hinge point. For example point Q_k 186 is the hinge point for body k 184. A fixed set of coordinate axes is established in the inertial frame 198. An arbitrary configuration of the MBS is chosen as its reference configuration 190. While in this configuration the image of the inertial coordinate axes is used to establish a set of body-fixed axes in each body. In the reference configuration each hinge point Q is coincident with P, a point of its parent body (or extended body.) For each body, point P is called the body's inboard hinge point. So the inboard hinge point P_k 188 for body k 184 is a point fixed in its parent body i 182. The inboard hinge point for each base body is a point O 192 fixed in ground. The expanded view that shown in Fig. 2 more clearly shows that point Q_k 186 is fixed in body k 184 and point P_k 188 is fixed in parent body i 182.

The hinge point locations define $\mathbf{d}(k)$ 194, a constant vector for each body, and can also be written $r^{Q_i P_k}$. The vector for body k is fixed in its parent body i. It spans from the hinge point for body i to the inboard hinge point for body k. The vector $\mathbf{d}(1)$ 196 spans from the inertial origin to the first base body's inboard hinge point (also a point fixed in ground), and can be written r^{OQ_1}.

For a body, $m(k)$, $\mathbf{p}(k)$, and $\underline{\underline{\mathbf{I}}}_{Q_k}(k)$ define the mass properties of body k for its hinge point Q_k. These are, respectively, the mass, the first mass moment, and the inertia matrix of the body for its hinge point in the coordinate frame

of the body. For a rigid body made up of a distribution of particles, the mass properties are constants that are computed by a preprocessing module. The details of these computations can be found in standard references, such as Kane, T.R., *Dynamics, 3rd Ed.*, January 1978, Stanford University, Stanford, CA.

Let $M(k)$, the spatial inertia of body k for its hinge point Q_k, be given by the symmetric 6 by 6 matrix

$$M(k) = \begin{bmatrix} \underline{\underline{\mathbf{I}}}_{Q_k}(k) & \tilde{\mathbf{p}}(k) \\ -\tilde{\mathbf{p}}(k) & m(k)\underline{\underline{E}}_3 \end{bmatrix}$$

Each joint in the system is described by geometric data. For instance, a pin joint is characterized by an axis fixed in the two bodies connected by the joint. The particular data for a joint depends on its type. The number n, the *inb* function, the system mass properties, the vectors $\mathbf{d}(k)$, and the joint geometric data (including joint type) constitute the *system parameters*.

Joints and Generalized Coordinates of the Model

Figs. 4A-4C illustrate the joint definitions of the preferred embodiment of the MBS: the slider joint 100, the pin joint 102, and the ball joint 104. Each joint allows translational or rotational displacement of the hinge point Q_k 106 relative to the inboard hinge point P_k 108. These displacements are parameterized by $q(k)$ 110, the generalized coordinates for body k. In passing, it should be noted that generalized coordinates are examples of generalized quantities, which refer to quantities that have both rotational character and translational character. For instance, a generalized force acting at a point consists of both a force vector and a torque vector. The generalized coordinate $q(k)$ for the slider joint 100 is the sliding displacement x 112. The generalized coordinate $q(k)$ for the pin joint 102 is the angular displacement θ 114. The generalized coordinate $q(k)$ for the ball joint 104 is the Euler parameters $(\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4)$ 116.

Each joint may be a pin, slider, or ball joint; or a combination of these joints. Many other joint types are possible, including, but not limited to, free joints, U-joints, cylindrical joints, and bearing joints. For instance, $q(k) = (x, y, z)$, the inertial measure numbers of the vector from the base body inboard hinge point to the base body hinge point express the base body displacement in ground as three

orthogonal slider joints. A free joint consists of three orthogonal slider joints combined with a ball joint, and has the full 6 degrees of freedom.

The collection of generalized coordinates for all the bodies comprises the vector $\mathbf{q}$, the generalized coordinates for the system.

Given the generalized coordinates for a particular joint, two quantities: $r^{P_k Q_k}(k)$, the joint translation vector and ${}^{i}C^{k}(k)$, the direction cosine matrix for body k in its parent are formed. The translation vector $r^{P_k Q_k}(k)$ expresses the vector from the inboard hinge point P of body k to the hinge point Q of body k, in the coordinate frame of the parent body. Details of these computations depend on the joint type and can be easily derived. For purposes of this description, access to a function that can generate $r^{P_k Q_k}(k)$ and ${}^{i}C^{k}(k)$ given the system generalized coordinates is assumed.

As introduced, the choice of hinge point for each body is arbitrary. However, judicious choice greatly simplifies matters. For instance, for pin joints the hinge point should be chosen as a point on the axis of the joint. For this choice points P and Q remain coincident for all values of the joint angle, so the joint translation is zero. If the point Q is chosen at a distance from the axis, points P and Q move relative to each other:

$$r^{P_k Q_k}(k) = \boldsymbol{\lambda} \times r^{OQ_k} \sin\theta - (1-\cos\theta)\left(\underline{\underline{E}}_3 - \boldsymbol{\lambda}\boldsymbol{\lambda}^*\right) r^{OQ_k}$$

where $\boldsymbol{\lambda}$ is the joint axis unit vector, θ is the joint angle, and r^{OQ_k} is the vector from any point on the axis to point Q.

For pin joints and ball joints, a point on the axis is always chosen as the hinge point. For these joints the translation vector $r^{P_k Q_k}(k)$ is zero.

For a slider joint, the translation vector $r^{P_k Q_k}(k)$ is $q(k)\boldsymbol{\lambda}$.

The direction cosine matrix for a pin is

$${}^{i}C^{k}(k) = \underline{\underline{E}}_3 \cos\theta + \tilde{\boldsymbol{\lambda}} \sin\theta + \boldsymbol{\lambda}\boldsymbol{\lambda}^*(1-\cos\theta).$$

The direction cosine matrix for a slider is $\underline{\underline{E}}_3$.

Generalized Speeds of the Model

Let ${}^{i}V^{k}(k)$, the generalized velocity of the hinge point of body k measured in its parent i, be parameterized by $u(k)$, a set of generalized speeds. Then:

$${}^{i}V^{k}(k) = \begin{pmatrix} {}^{i}\omega^{k}(k) \\ {}^{i}v^{Q_k}(k) \end{pmatrix} = H^*(k)u(k)$$

Here, the matrix $H(k)$ is called the joint map for this joint. It is a $n_u(k)$ by 6 matrix, where $n_u(k)$ is the number of degrees of freedom for the joint (1 for a pin or slider, 3 for a ball, 6 for a free joint). $H(k)$ can, in general have dependence on coordinates q. Given the generalized speeds for the joint, the joint map generates the joint linear and angular velocity, expressed in the child body frame. The following are used for the joints:

$$H(k) = [\hat{1} \quad 0 \quad 0 \quad 0], \ pin$$
$$H(k) = [0 \quad 0 \quad 0 \quad \hat{1}], \ slider$$
$$H(k) = [\underline{\underline{E}}_3 \quad \underline{\underline{0}}_3], \ ball$$
$$H(k) = \begin{bmatrix} \underline{\underline{E}}_3 & \underline{\underline{0}}_3 \\ \underline{\underline{0}}_3 & {}^iC^k(k) \end{bmatrix}, \ free$$

The collection of generalized speeds for all the bodies comprises the vector $\boldsymbol{u}$, the generalized coordinates for the system. As before, access to a function that can generate the vector ${}^iV^k(k)$ given (q,u) and a specific joint type, is assumed. Access to a function that can compute the derivatives $\dot{q}(k) = \dot{q}(q(k),u(k))$ is also assumed. This routine generates the time derivative of the generalized position coordinates:

$$\dot{q} = W(q)u$$

where $W(q)$ is a block diagonal matrix that relates $\dot{q}$ and u, with each block depending upon the joint type:

$\dot{q} = u$ for pin joint, slider joint

$$\begin{bmatrix} \dot{\varepsilon}_1 \\ \dot{\varepsilon}_2 \\ \dot{\varepsilon}_3 \\ \dot{\varepsilon}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \varepsilon_4 & -\varepsilon_3 & \varepsilon_2 \\ \varepsilon_3 & \varepsilon_4 & -\varepsilon_1 \\ -\varepsilon_2 & \varepsilon_1 & \varepsilon_4 \\ -\varepsilon_1 & -\varepsilon_2 & \varepsilon_4 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \text{ for ball joint}$$

where $q = [\varepsilon_1 \quad \varepsilon_2 \quad \varepsilon_3 \quad \varepsilon_4]^*$ and $u = [\omega_1 \quad \omega_2 \quad \omega_3]^*$

and a free joint is a combination of 3 slider joints and one ball joint. Note that there are 4 $\dot{q}$'s (derivatives of the Euler parameters) associated with 3 u's for ball joints.

Similarly, ${}^iA^k(k)$, the generalized acceleration of the hinge point of body k in its parent, is given by:

$${}^iA^k(k) = \begin{pmatrix} {}^i\alpha^k(k) \\ {}^ia^{O_k}(k) \end{pmatrix} = H^*(k)\dot{u}(k)$$

It is these generalized coordinates q, and generalized speeds u, the internal coordinates for purposes of this description, of the molecular system which are calculated. Rather than working with the typical inertial coordinates (x , y, z) and speeds in these inertial coordinate systems, calculations for the subject molecular system are reduced.

CALCULATIONS OF THE EQUATIONS OF MOTION

With the exemplary rigid multibody, torsion angle model described, the equations of motion can now be calculated. In accordance with the present invention, the motion of the MBS molecular model is determined by the Residual Form. The Residual Form method requires calculations termed the "first" kinematic calculations to distinguish them from the "second" kinematic calculations, which are further required by the Direct Form (which is included in this description for purposes of comparison).

First Kinematic Calculations for the Molecular Model

In the first kinematic calculations, given the internal coordinates of the molecular system, $(q,u,\dot{u})$ and the system parameters, the following position, velocity and acceleration kinematics are computed for each rigid body k of the molecular model. (In passing, it should be noted that when the First Kinematic calculations are done for the Residual Form method, the $\dot{u}$ is passed in as a guess of the solution which the integration method then refines to the correct solution. In contrast, $\dot{u}$ is set to zero when used for the Direct Form method. This is shown clearly in the later descriptions of the two methods.)

For each body k compute:

$$
\begin{aligned}
&{}^{N}C^{k}(k), r^{O_k O_k}(k), r^{OO_k}(k), {}^{i}\phi^{k}(k),\\
&{}^{N}\omega^{k}(k), {}^{N}v^{O_k}(k), V(k),\\
&{}^{N}\alpha^{k}(k), {}^{N}a^{O_k}(k), A(k)
\end{aligned}
$$

These computations are done recursively, starting from each base body and progressing to the leaves.

${}^{N}C^{k}(k)$, the direction cosine matrix for body k in ground is defined as:

$$
\begin{aligned}
&{}^{N}C^{k}(1) = {}^{i}C^{k}(1)\\
&{}^{N}C^{k}(k) = {}^{N}C^{k}(i)\,{}^{i}C^{k}(k), \quad k = 2,\ldots n, \quad i = inb(k)
\end{aligned}
$$

${}^{i}C^{k}(k)$ comes from the joint routine described above.

$r^{Q_i Q_k}(k)$, the position vector from Q_i, the hinge point of the parent of body k to Q_k, the hinge point of body k, expressed in the parent frame, is defined as:

$$r^{Q_i Q_k}(k) = \mathbf{d}(k) + r^{P_i Q_k}(k), \quad k = 1,\ldots n$$

$r^{P_i Q_k}(k)$ comes from the joint routine.

$r^{OQ_k}(k)$, the position vector from the inertial origin O to Q_k, the hinge point of body k, expressed in the global frame, is defined

$$r^{OQ_k}(1) = r^{Q_i Q_k}(1)$$

$$r^{OQ_k}(k) = r^{OQ_k}(i) + {}^N C^k(i) r^{Q_i Q_k}(k), \quad k = 2,\ldots n, \quad i = inb(k)$$

${}^i\phi^k(k)$, the rigid body transformation operator for body k is defined

$${}^i\phi^k(k) = \begin{pmatrix} {}^i C^k(k) & \tilde{r}^{Q_i Q_k}(k)\,{}^i C^k(k) \\ \underline{0}_3 & {}^i C^k(k) \end{pmatrix}, \quad k = 1,\ldots n$$

$V(k)$, the spatial velocity for body k at its hinge point, expressed in the frame of body k, is defined

$$V(1) \triangleq \begin{pmatrix} {}^N\omega^k(1) \\ {}^N v^{Q_k}(1) \end{pmatrix} = {}^i V^k(1)$$

$$V(k) \triangleq \begin{pmatrix} {}^N\omega^k(k) \\ {}^N v^{Q_k}(k) \end{pmatrix} = {}^i\phi^{k*}(k) V(i) + {}^i V^k(k), \quad k = 2,\ldots n, \quad i = inb(k)$$

$A(k)$, the spatial acceleration for body k at its hinge point, expressed in the frame of body k, is defined

$$A(1) \triangleq \begin{pmatrix} {}^N\alpha^k(1) \\ {}^N a^{Q_k}(1) \end{pmatrix} = {}^i A^k(1)$$

$$A(k) \triangleq \begin{pmatrix} {}^N\alpha^k(k) \\ {}^N a^{Q_k}(k) \end{pmatrix} = \overline{A} + \begin{pmatrix} \tilde{\omega} & \underline{0}_3 \\ \underline{0}_3 & 2\tilde{\omega} \end{pmatrix} {}^i V^k(k) + {}^i A^k(k), \quad k = 2,\ldots n, \quad i = inb(k)$$

where

$$\overline{A} = {}^i\phi^{k*}(k) A(i) + \begin{pmatrix} \underline{0}_3 \\ {}^i C^{k*}(k)\left({}^N\omega^k(i) \times {}^N\omega^k(i) \times r^{Q_i Q_k}(k) \right) \end{pmatrix}$$

$$\omega = {}^i C^{k*}(k)\, {}^N\omega^k(i)$$

Of course, the computations can all be computed in a single pass if desired.

After completing these steps for one incremental time step, the MBS can service kinematics requests to compute the (generalized) position, velocity, or acceleration information for any point of any body. This is done by computing the required information for any point in terms of the hinge quantities for its body, using standard rigid body formulas.

Residual Computation

With the first kinematic calculations described above, the residual computation for the Residual Form method can be determined. A detailed description of the Residual Form and its application to molecular modeling is found in the previously cited co-pending U.S. Patent Appln. No._____, entitled, "METHOD FOR RESIDUAL FORM IN MOLECULAR MODELING," filed of even date. This computation fills in two partitions of the vector $\begin{pmatrix} \rho_q \\ \rho_u \end{pmatrix}$ given previously. The first partition is called ρ_q, the kinematic residual, and the second partition is called ρ_u, the dynamic residual. The kinematic residual is computed from the difference between a $\dot{q}$, which is passed-in from the (implicit) integration submodules 66, and the derivative computed by each joint:

$$\dot{q} - W(q)u = \rho_q$$

The dynamics residual is also computed. Starting with a given state of the molecular model, i.e., given $(q,u,\dot{u})$ and the system parameters, a program routine models the 'environment' of the MBS. Such routines are readily available to, or can be created by, practitioners in the computer modeling field. The routine takes the values (q,u) determined by and passed in from the integration submodules 66 and returns (the state-dependent) $T(k) = \begin{pmatrix} T_{Q_k}(k) \\ F(k) \end{pmatrix}$, the applied spatial force for a body k at its hinge point Q_k, and $\sigma(k)$, the hinge torque for the body k. The dynamics residual, $\rho_u(k)$, associated with generalized speeds $u(k)$ for the body k is then computed by the following steps:

1. Perform the calculations for the molecular model by the Residual Form as described above with the passed-in state values $(q,u,\dot{u})$;
2. Generate $\hat{T}(k)$, the spatial load balance for each body k in the model having n bodies:

$$\hat{T}(k) = M(k)A(k) + \begin{pmatrix} {}^N\tilde{\omega}^k(k)\left(\underline{\underline{\mathbf{I}}}_Q(k)\,{}^N\omega^k(k)\right) \\ {}^N\tilde{\omega}^k(k)\left({}^N\omega^k(k)\times \mathrm{p}(k)\right) \end{pmatrix} - T(k)$$

$k = 1,\ldots n$

3. Compute $\rho_u(k)$

for $k = n$ to 2 by -1

$\rho_u(k) = H(k)\hat{T}(k) - \sigma(k)$

$i = inb(k)$

$\hat{T}(i) \mathrel{+}= {}^{i}\phi^{k}(k)\hat{T}(k)$

end

$\rho_u(1) = H(1)\hat{T}(1)$

The Residual Form method evaluates the extent to which the system differential equations are satisfied. Zero residual indicates that the applied forces are in balance with the inertia forces. However, this does not mean the system is in static equilibrium, but rather that the applied forces would reproduce the given $\dot{u}$ when applied to the system in the state (q,u). The residuals can be interpreted as that additional hinge torque needed to balance the applied and inertia forces. In the literature this method is known as either inverse dynamics, or the method of computed torques. It governs the case where the $\dot{u}$ are all prescribed. At this point all the computations required for the Residual Form are complete. The residuals ρ_q and ρ_u are used directly by the implicit integrator in the integrator submodule 68.

Second Kinematics Calculations for the Molecular Model

To carry out the Direct Form method, calculations in addition to the first kinematics calculations are required. These additional calculations are termed the second kinematics calculations. The values $P(k), D(k), {}^{i}\psi^{k}(k), {}^{i}K^{k}(k)$ are computed as follows:

1. Perform the calculations for the Molecular Model by the Residual Form as described above, i.e., the first kinematics calculations.
2. $P(k)$, the articulated body inertia of each body k, is initialized.

$$P(k) = M(k), \quad k = 1,\ldots,n$$

3. The objects below are then generated:

for $k = n$ to 2 by -1

$D(k) = H(k)P(k)H^*(k)$

$G = P(k)H^*(k)D^{-1}(k)$

$\bar{\tau} = \underline{\underline{E}}_6 - GH(k)$

${}^i\psi^k(k) = {}^i\phi^k(k)\bar{\tau}$

${}^iK^k(k) = {}^i\phi^k(k)G$

$i = inb(k)$

$P(i) \mathrel{+}= {}^i\psi^k(k)P(k){}^i\psi^{k*}(k)$

end

$D(1) = H(1)P(1)H^*(1)$

The functional dependence of these quantities is only upon the generalized coordinate q. Therefore, the first kinematics calculations are programmed in anticipation of performing the second kinematics calculations.

Forward Dynamics Calculations

Finally, $\dot{u}$ is calculated by sweeping inboard, then outboard, of the molecule:

$z(k) = \underline{0}_6, \quad k = 1, \ldots n$

for $k = n$ to 2 by -1

$\varepsilon(k) = \rho_a(k) - H(k)z(k)$

$\upsilon(k) = D^{-1}(k)\varepsilon(k)$

$i = inb(k)$

$z(i) \mathrel{+}= {}^i\psi^k(k)z(k) + {}^iK^k(k)\rho_a(k)$

end

$\varepsilon(1) = \rho_a(1) - H(1)z(1)$

$\upsilon(1) = D^{-1}(1)\varepsilon(1)$

$\dot{u}(1) = \upsilon(1)$

$\delta(1) = H^*(1)\upsilon(1)$

for $k = 2$ to n

$i = inb(k)$

$\delta(k) = {}^i\psi^{k*}(k)\delta(i) + H^*(k)\upsilon(k)$

$\dot{u}(k) = \upsilon(k) - {}^iK^{k*}(k)\delta(i)$

end

With the First and Second Kinematics Calculations, and the Forward Dynamics Calculations, the Direct Form method is available.

Direct Form Method for the Equations of Motions

The Direct Form method takes the current state (q,u) and computes the derivatives $(\dot{q},\dot{u})$ using the above algorithms, which are then used by the integration method to advance time.

Given: (q,u)

Compute: $(\dot{q},\dot{u})$

1. Compute $\dot{q}$ using joint specific routine as above
2. Perform above First Kinematics Calculations with $\dot{u}=0$
3. Generate residuals ρ_u as above
4. Negate the residuals $\rho_u = -\rho_u$
5. Perform Second Kinematics Calculations
6. Compute $\dot{u}$ using Forward Dynamics Calculations above

The Direct Form method produces the hinge accelerations $\dot{u}$ in response to the applied forces acting on the system. Fig. 5 summarizes the computation steps of the Residual Form method and the Direct Form method.

JACOBIANS IN IMPLICIT INTEGRATION

The MD equations which model a molecule (such as a protein), are implemented as a multibody system (MBS). These equations represent Newton's Laws and are expressed as a set of differential equations $\dot{y} = f(y,t)$. The differential equations are implemented using a suite of Order(N) multibody dynamics methods. To advance the equations in time, in accordance with the present invention, an implicit method of numerical integration is used, in particular, L-stable implicit integration methods, such as implicit Euler, Radau5, and SDIRK3.

An important ingredient of this integration process is formation of the Jacobian of the differential equations. This is

$$J \triangleq \frac{\partial f}{\partial y}$$

Since the function f is itself computed by an algorithm rather than by an explicit formula, the Jacobian computation represents a substantial amount of work. In the simplest approach, the Jacobian can be formed numerically by differencing the derivative routine. This is a delicate operation because the quality of the Jacobian is a tradeoff between round-off and truncation errors. Typically half the working

precision in the result is retained by choosing a good perturbation size in the difference scheme. In practice, though, this is difficult to do.

However, the structure of the governing equations may be exploited to improve the Jacobian computation. The exemplary multibody dynamics methods illustrate this. The algorithms involved compute exact derivatives, even though numerical methods are used to execute the formulas. The derivatives obtained are in error by amounts that depend upon round off and the conditioning of the multibody system under consideration. But no approximations are involved at the equation level.

In general, G, the iteration matrix used in the Newton loop of the implicit integrator has the form $G = E - \alpha J$, where E is the identity matrix and α is be some scalar function of the timestep. See the previously referenced U.S. Patent Appln. No. _____, entitled "METHOD FOR LARGE TIMESTEPS IN MOLECULAR MODELING," filed of even date, for a description of implicit integrators. Changes in step size require refactoring G, but not reforming J. Reforming J is needed only when the Jacobian is needed at a new state. G is used in a linear subproblem within a Newton loop. The following is solved:

$$G\Delta y = -r(y_n^i),$$
$$y_n^{i+1} = y_n^i + \Delta y$$

where $r(y)$ is the residual function for that particular implicit integration method.

As shown later, J has a special structure, which is inherited by G. This means that equation solving with G can be done at reduced cost, if the structure is exploited.

The quality of the Jacobian affects the ability to solve the nonlinear equations resulting from discretization in the integrator. Failure to solve the Newton loop may require retraction of a trail step and reduction of the integration time step. The timestep should be controlled by accuracy, rather than failures in the Newton loop.

Computation of Analytic Jacobian

The Jacobian J is a matrix which represents a linearization of the equations of motion. Normally, the governing equations for a dynamical system are linearized around an equilibrium state, or perhaps a state of steady motion. In this

case, the equations are linearized around an arbitrary state so all possible contributing terms should be developed. It is customary to describe J in terms of its partitions:

$$J(q,u)=\begin{pmatrix}\frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial u}\\ \frac{\partial \dot{u}}{\partial q} & \frac{\partial \dot{u}}{\partial u}\end{pmatrix}\triangleq\begin{pmatrix}J_{qq} & J_{qu}\\ J_{uq} & J_{uu}\end{pmatrix}$$

Structure of J_{qq} AND J_{qu}

The $\dot{q}$ equation is $\dot{q}=Wu$, where the matrix W has a block-diagonal structure. Each block depends upon the joint type. Pins and slider joints give rise to 1 by 1 identity blocks. A ball joint generates a 4 by 3 block that expresses the Euler parameter derivatives in terms of Euler parameters and body angular velocity measure numbers (generalized speeds).

From the $\dot{q}$ equation above, these two partitions of the Jacobian matrix are formed:

$$J_{qq}=\frac{\partial W(q)}{\partial q}u$$

$$J_{qu}=W$$

These equations are to be interpreted for symbolic purposes only. In practice, there is no need to generate the matrix W explicitly. The non-zero block diagonal elements are filled in as discussed in the previous section on the kinematic residual.

Structure of J_{uq} AND J_{uu}

The $\dot{u}$ derivatives are more complicated. Since $\dot{u}=-M^{-1}\rho_u$, $\frac{\partial\left(-M^{-1}\rho_u\right)}{\partial q}$ and $\frac{\partial\left(-M^{-1}\rho_u\right)}{\partial u}$ must be computed. (Note that ρ_u is the residual of the dynamics routine developed earlier). Here a key result from the field of Numerical Analysis is used to avoid the derivative of the matrix inverse.

Suppose $A(x)y(x)=b(x)$, then we can write $y(x)=A^{-1}(x)b(x)$. If $y(x_0)=z$ is known, and the value of $\frac{\partial y(x)}{\partial x}\Big|_{x=x0}$ must be obtained, we have

$$\frac{\partial y}{\partial x}\Big|_{x=x0}=A^{-1}(x0)\frac{\partial}{\partial x}\left(b(x)-A(x)z\right)\Big|_{x=x0}$$

where $z = A^{-1}b$ is held fixed when forming the right-hand side of the equation.

Applied to the described multibody routines,

$$\frac{\partial \dot{u}}{\partial x} = -\frac{\partial}{\partial x}\left(M^{-1}\rho_u(q,u,0)\right) = -M^{-1}\frac{\partial}{\partial x}\rho_u(q,u,z)$$

where $z \triangleq M^{-1}\rho_u(q,u,0)$. This result avoids the computing of the derivative of M^{-1}, which is a hypermatrix. The matrix inverse is "pulled-out". In the above equation, "x" is either the generalized coordinate q or the generalized speed u, depending on the Jacobian partition that is to be computed.

In summary, to compute the blocks J_{uq} and J_{uu}, three steps are followed:

1. Given (q,u) compute $z \triangleq -M^{-1}\rho_u(q,u,0)$ using the Direct Form method. This simply says to compute $\dot{u}$ from the current state and save it as the variable z.
2. Compute the analytic Jacobian of the dynamics residual routine. In this step, the matrix $\frac{\partial}{\partial x}\rho_u(q,u,z)$ is to be formed. This quantity clearly depends upon the vector z computed in Step 1. Note that the numerical value of the residual ρ_u in this step is zero for each element since we are computing the Jacobian around a consistent solution of the motion equations. The partitions J_{uq} and J_{uu} of the Residual Jacobian are obtained by substituting "q" and "u" for the "x" above.
3. Back-solve the result of Step 2 with the mass-matrix to obtain the desired block. The back-solve operation is accomplished in the Direct Method routine by processing a residual vector into a $\dot{u}$ vector. The Second Kinematics Step only needs to be performed once, since the back-solves are done at the nominal value of the state. In fact, the Second Kinematics routine must have been called in Step 1 while computing z, so the variables should still be cached.

In words, the Jacobian of our derivative routine can be formed by back-solving the Jacobian of our residual routine. The residual Jacobian is much simpler to compute than the derivative Jacobian. Steps 2 and 3 above are derived in the following sections.

The Residual Jacobian

The computation of the Residual Jacobian is closely related to the Residual Form method for dynamics, which is summarized here:

1. Perform an outboard pass that computes the kinematic data that depends upon position and velocity.
2. Make a call to the force routine which generates the atomic forces and consolidates them into spatial loads acting on the bodies.
3. Perform another kinematic pass that computes acceleration level quantities (using passed-in $\dot{u}$), and combines inertia forces with the spatial loads from Step 2.
4. Perform an inward pass that generates the residual at each joint. This pass recursively computes the resultant of the (spatial) inertial and applied forces for the 'nest' of bodies outboard of the joint in question. The residual is the projection of the resultant on the joint's degrees of freedom, given by the joint map data.

At a high level the residual computation can be considered to depend upon two kinds of forces: 'motion forces' and external forces. The motion forces are computed directly by the multibody system. The external forces are available to the multibody system from a force modeling routine that computes the various interatomic forces such as electrostatics and solvents. A similar procedure is followed when computing the Jacobian. The multibody system builds the Jacobian of the motion forces, and combine it with the Jacobian of the external forces.

Partitioning the Force Field into Effects

There are many forces that may be acting on the molecule, and these forces may be computed in various intrinsic coordinate frames that are most convenient for that particular force "effect". For example, electrostatic terms may be computed using multipole methods and spherical coordinates, covalent terms may be computed in terms of torsion and bond angles, and solvent forces may be computed in global Cartesian coordinates. During the computation of the Residuals, these forces are transformed from their intrinsic coordinate frame to the MBS coordinates.

The same exchange occurs to compute Jacobians. The native Jacobians in their intrinsic coordinates are be brought into the MBS coordinates. This requires the use of the chain-rule to transform between intrinsic and the MBS

generalized coordinates. It is important that each effect co-computes its function value and Jacobian, because many of the same terms are needed for each computation. Each effect is transformed into a set of spatial loads $T_{effect}(k)$, where k is the index of a generic body in the system. The totality of these effects is given the symbol $T(k)$.

Effect Jacobians Brought into the Residual Jacobian

At a high level, the residual routine was previously implemented from the equation

$$\rho_u = H\phi(MA - T)$$

The implementation uses Order(N) methods which are immediately obvious from the equation above. In this equation T is a vector of spatial loads acting on the pivots of the multibody system, where each element is a spatial load (a 6-vector composed of one force and one torque). It actually represents all effects other than inertia loads or pure hinge loads. The term in parentheses represents the load balance for each body. The first term is the inertia force, the next is the spatial load. $M(k)A(k)$ is the spatial inertia force for a typical body. This is built from the body mass properties and the spatial acceleration of the body pivot. The spatial acceleration is computed before the residual routine is executed by the Forward dynamics routine. The operator $H\phi$ is implemented in a routine that performs an Order(N) inward pass.

Even without knowing anything about the details of the computation implied by the equation above, the contribution of $\frac{\partial T}{\partial x}$, the spatial load Jacobian (the effect Jacobian) to $\frac{\partial \rho_u}{\partial x}$, the residual Jacobian, can be immediately inferred:

$$\frac{\partial \rho_u}{\partial x} = -H\phi\frac{\partial T}{\partial x} + \ldots$$

The ... refers to terms not involving the effect Jacobian. Again, q or u for "x", is substituted, depending upon which partition of the Jacobian is being computed.

The role of an effect Jacobian in the residual Jacobian is the same as the role of the effect in the multibody equations. This means that T contributes to the residual ρ_u in the same way that a column of $\frac{\partial T}{\partial x}$ contributes to $\frac{\partial \rho_u}{\partial x}$. Both are processed by the same operator $H\phi$. This is a crucial point because it means that no

new method is needed for this part of the Jacobian computation. (A different method is need to obtain $\frac{\partial T}{\partial x}$).

Thus, given the effect Jacobian, its contribution is assembled into the residual Jacobian by operating on it with the original residual routine, treating it as a multi-column set of spatial load vectors. This is a direct consequence of the linearity of the equations.

The columns of the residual Jacobian play the same role in the derivative Jacobian routine as the residual vectors play in the Forward Dynamics routine. The dynamics routine performs a back-solve on a data vector it receives, and doesn't need to know what the data is, just what operation to perform on it. This applies to all the routines.

Adding the ... terms, the chain rule is used to show the whole equation:

$$\frac{\partial \rho_u}{\partial x} = H\phi\left(\frac{\partial (MA)}{\partial x} - \frac{\partial T}{\partial x}\right) + H\frac{\partial (\phi z)}{\partial x} + \frac{\partial (Hy)}{\partial x}$$

At this level, there are four contributions to the Jacobian: the inertia forces, the spatial forces, and contributions due to changes in the operators ϕ and H. The quantities z and y refer to $(MA - T)$, and ϕz, respectively, which are held fixed while evaluating the last two terms. The numerical values of these terms are already available from the residual computation. Another observation about the above equation is that the operator ϕ depends only upon q, but not u. Thus, this term remains constant while computing the partition J_{uu}. Similarly, the spatial load can be split into its constituent effects, some of which do not depend upon u. In general, this means that knowledge of the multibody equations can be exploited to optimize the computations.

Up to now, what to do with $\frac{\partial T}{\partial x}$ once the term has been computed has been described, but a description of how to form the term has not been made. These details are in the following sections.

Computing the Effect Jacobians and Combining with the Residual Jacobian

So far a high-level description of the Jacobian computation has been given. It can be seen that the computation has a very algorithmic flavor to it. There are very distinct phases to the task, just as there were also for the Forward Dynamics

routine described previously. There, computation of atomic forces is clearly the bottleneck step. Yet the overhead in the multibody equations for dealing with forces is fairly small. In that case, a call was made to the force routine, and what occurs inside the routine was ignored. When it comes to the Jacobian, this aspect is less true.

A call is still made to obtain the Effect Jacobian, but there is a lot of processing needed before the Effect Jacobian can be assembled into the Residual Jacobian. The details of dealing with force fields to produce Jacobians are covered in the next sections and an example of incorporating electrostatics is developed. All other loads follow a similar development.

Electrostatics as an Example Effect

The basic premise of electrostatics is that the force between two charged particles is

$$F_{ij} = \kappa \frac{q_i q_j}{r_{ij}^2} \hat{r}_{ij}$$

This is a classical inverse square law. F_{ij}, the force acting on particle i due to particle j, depends upon the charge of the particles, attractive for oppositely charged particles, repulsive for like charges. The symbol $\hat{r}_{ij}$ is a unit vector directed from particle i to particle j; r_{ij} is the distance between the particles; κ is a unit-dependent constant related to the strength of the forces. Of course, the force acting on particle j is equal and opposite to that acting on the particle i. Hence, given a collection of particles interacting through Coulomb's Law given above, the net force acting on each particle is computed by summing the pair-wise forces. For this example, the forces are computed in global Cartesian coordinates.

With the atomic forces in hand, the multibody forces can be generated. The system of forces acting on the particles of each body is replaced by a spatial load acting at the pivot of each body. The atomic forces are first expressed in a body-fixed basis, and then shifted to the pivot using the station coordinates of the particular atom to which the force is bound.

F_{ij} is a vector. The derivative of F_{ij} with respect to dr_{ij}, small changes in the particle's relative positions is $\underline{\underline{D}}_{ij}$, a tensor. It is such that $dF_{ij} = \underline{\underline{D}}_{ij} \bullet dr_{ij}$. For Coulomb force the tensor is

$$\underline{\underline{D}}_{ij} = \kappa \frac{1}{r_{ij}^3}\left(\underline{\underline{E}}_3 - 3\hat{r}_{ij}\hat{r}_{ij}^T\right)$$

Some observations:

1. The Force Jacobian is a matrix of size $n_{atoms} \times n_{atoms}$. Each element is a 3 by 3 tensor. The (i,j) block gives the derivative of force on atom i with respect to small changes in the position of atom j. In general, every force model is required to support an intrinsic Jacobian method for analytical processing.
2. Storage requirements for the Force Jacobian quickly become impractical. This leads to the notion of interface "contraction" where the Jacobian of all the forces acting pair-wise between the atoms are reduced or "contracted" to acting pair-wise between the bodies.
3. Because the Coulomb force is a pair-wise interaction, each force contributes to two blocks in the overall Jacobian. Thus, each force is processed at constant cost, and the overall Jacobian is computed at a cost proportional to the number of atoms squared, i.e., Order(N^2). This is the same as the computational cost of the force itself! This is a rather good result for computing analytic Jacobians. A numerical Jacobian requires a fresh force computation each time an element of the state is perturbed. This leads to cubic growth, i.e., Order(N^3), in the cost of the numerical Jacobian. Hence, the analytic Jacobian is much cheaper to compute as well as more accurate than a numerical Jacobian.
4. The computation of the Jacobian is conveniently done in the same routine that computes the force (co-computation). However, it typically needs to be done far less often than the force computation. Therefore, a flag can be used to trigger the Jacobian calculation only when needed.

Coupling to the Displacement Gradient

Having obtained the (intrinsic) Force Jacobian, it is necessary to process the Jacobian further. This is due to the fact that the multibody system is formulated using relative coordinates. The chain rule is applied to each atomic force $F(k,i)$ and is called "coupling to the displacement gradient". This denotes the global Cartesian force on atom i, which resides on body k.

$$\frac{\partial F(k,i)}{\partial q_j} = \sum_{p=1}^{nbod} \sum_{s \in p} \frac{\partial F(k,i)}{\partial r(p,s)} \frac{\partial r(p,s)}{\partial q_j}$$

where $r(p,s)$ is the position of atom "s" on body "p".

The first term in the sum selects an element of the Force Jacobian which was just computed. The quantity $\frac{\partial r(p,s)}{\partial q_j}$ is an element of the displacement gradient. A typical term gives the change in an atom's position due to a small change in a generalized coordinate. Note that this term is strictly a kinematical quantity having nothing to do with the force computation. Thus, the Force Jacobian can be computed once and then continually reprocessed by the chain rule for each coordinate in the multibody system. This step represents a matrix vector multiply, since $\frac{\partial r_j}{\partial q_s}$ is a column vector with n_{atoms} entries (each a 3 vector), and the Jacobian is a square matrix $n_{atoms} \times n_{atoms}$, where each element is a 3 by 3 tensor.

It is possible to improve this computation, since many of the entries in the displacement Jacobian are known to be zero. This is due to the fact that incrementing a particular hinge does not displace every atom in the system, but only those outboard of the displaced hinge, and not disjoint from the hinge in question. For instance, rotating the base body induces a change in all atoms of the system. But perturbing a torsion angle on any terminal body induces a change only to those atoms resident in the terminal body. Therefore, roughly speaking, about half the work may be saved by optimizing the computation. This reduction comes from a strictly knowledge-based approach.

Interface Contraction

In the process of forming $T(k)$, the spatial load on body k, the load comes from the atomic forces acting on atoms that belong to body k. Each force is transformed from global to local coordinates, and then shifted to the body pivot. A concise statement of this procedure is:

$$T(k) = \sum_{i \in k} \phi(k,i) T(k,i)$$

The operator $\phi(k,i)$ is

$$\phi(k,i) = \begin{bmatrix} \underline{\underline{E}}_3 & \tilde{\rho}(k,i) \\ \underline{\underline{0}}_3 & \underline{\underline{E}}_3 \end{bmatrix} \begin{bmatrix} {}^N\underline{\underline{C}}^k(k) & \underline{\underline{0}}_3 \\ \underline{\underline{0}}_3 & {}^N\underline{\underline{C}}^k(k) \end{bmatrix}^*$$

where $\rho(k,i)$ are the fixed station coordinates of atom i on body k. Note that the new quantity $T(k,i)$ appears. This is just the atomic force turned into a spatial load at the atomic position of atom i:

$$T(k,i) = \begin{bmatrix} \underline{0}_3 \\ F(k,i) \end{bmatrix}$$

The first element of the atomic spatial load is zero because there is no torque exerted by the force field on individual atoms.

Now, $T(k)$ relates atomic forces to body spatial loads. So, the derivative of this equation relates differential atomic forces to differential spatial loads:

$$\frac{\partial T(k)}{\partial q_j} = \sum_{i \in k} \phi(k,i) \frac{\partial T(k,i)}{\partial q_j} + \frac{\partial \phi(k,i)}{\partial q_j} T(k)$$

$$= T_1(k) + T_2(k)$$

The second term, $T_2(k)$, in this equation is discussed at the end of this section, as it involves the spatial loads, but not the load derivatives. This means the term can be treated generically, without worrying how the spatial loads were computed.

Substituting the definition of $T(k)$, into $T_1(k)$:

$$T_1(k) = \sum_{i \in k} \phi(k,i) \sum_{p=1}^{nbod} \sum_{s \in p} \frac{\partial T(k,i)}{\partial r(p,s)} \frac{\partial r(p,s)}{\partial q_j}$$

$$= \sum_{p=1}^{nbod} \sum_{s \in p} \frac{\partial T(k)}{\partial r(p,s)} \frac{\partial r(p,s)}{\partial q_j}$$

where now the symbol $\frac{\partial T(k)}{\partial r(p,s)} \triangleq \sum_{i \in k} \phi(k,i) \frac{\partial T(k,i)}{\partial r(p,s)}$ is an element of the *row-reduced* Force Jacobian. This Jacobian relates differential (body) spatial loads directly to differential atomic displacements. Again, $r(p,s)$ refers to the global position of the atom s on the body p. The term $\frac{\partial T(k)}{\partial r(p,s)}$ is formed by summing each atomic Force Jacobian element into the destination element in the reduced Jacobian, weighted by the atoms' $\phi(k,i)$ matrix. Each element of the row-reduced Jacobian is a 6 by 3 matrix. Hence the rows of the Force Jacobian have been contracted. The contraction is evident in the notation: the numerator has only a body index, while the denominator has both a body and an atom index. Depending on the number of atoms per body, the

row reduction can provide a savings in both storage and execution time when differential spatial loads must be formed.

Note that the row-reduction procedure only needs to be done once before computing the Residual Jacobian. The overhead of performing the reduction is more than offset by the reduced cost of the smaller matrix vector products which must be formed.

Note that in forming $\frac{\partial T(k)}{\partial r(p,s)}$ there is no need to save the elements of the atomic Force Jacobian. That is, each element $\frac{\partial T(k,i)}{\partial r(p,s)}$ only needs to be available while its contribution to $\frac{\partial T(k)}{\partial r(p,s)}$ is being computed. So, more than one element of the big Force Jacobian is not required at a time.

The global coordinates of a typical atom, $r(p,s)$, are computed in terms of $r(p)$, the global coordinates of body p's pivot, and $\rho(p,s)$, the station coordinates of the atom:

$$r(p,s) = r(p) + {}^{N}C^{k}(p)\rho(p,s)$$

By differentiating we find, (with some results from the kinematics of finite rotations):

$$\frac{\partial r(p,s)}{\partial q_j} = \frac{\partial r(p)}{\partial q_j} - \left({}^{N}C^{k}(p)\tilde{\rho}(p,s)\right)\lambda(p)$$

Augmenting this equation with the additional equation $\lambda(p,s) = \lambda(p)$ and defining w, the spatial derivative

$$w \triangleq \begin{bmatrix} \lambda \\ \frac{\partial r}{\partial q} \end{bmatrix}$$

the result is:

$$w(p,s) = \begin{bmatrix} \lambda(p,s) \\ \frac{\partial r(p,s)}{\partial q_j} \end{bmatrix} = \phi(p,s)^{*} \begin{bmatrix} \lambda(p) \\ \frac{\partial r(p)}{\partial q_j} \end{bmatrix}$$

The vector λ can be interpreted as generating a rate of change of orientation for each body. It is a field quantity, in the sense that it can potentially vary at each point in space. For rigid bodies undergoing pure rotations (without deformation), it is constant

for each body affected by the rotation. An Order(N) algorithm for computing λ recursively is described in a latter section.

With the application of the above equation to the equation for $T_1(k)$, the final reduction formula is reached:

$$T_1(k) = \sum_{p=1}^{nbod} \frac{\partial T(k)}{\partial w(p)} \frac{\partial w(p)}{\partial q_j}$$

$$\frac{\partial T(k)}{\partial w(p)} \triangleq \sum_{s \in p} \begin{bmatrix} 0_3 & \frac{\partial T(k)}{\partial r(p,s)} \\ 0_3 & \end{bmatrix} \phi^*(p,s)$$

Each element of $\frac{\partial T(k)}{\partial w(p)}$, the reduced Jacobian, is now a 6 by 6 matrix. An element of the reduced Jacobian relates the spatial load at the pivot of a body to a spatial derivative occurring at another pivot in the system (after coupling to the displacement gradient of the pivots).

The row reduction consolidated all the atomic forces on each body, leaving the spatial load derivatives coupled to displacement derivatives of all the atoms in the system. The column reduction consolidated all the atomic displacement derivatives, leaving the spatial load derivatives coupled to spatial pivot derivatives. Thus, the Force Jacobian is recast into a "native" form. Working with the reduced Jacobian speeds up computation of the spatial load derivatives by roughly the square of the number of atoms per body. This speedup can easily approach a factor of 100 or more.

This section shows, using electrostatics as an example, how to build an atomic-level Force Jacobian. This Jacobian relates differential atomic forces to differential atomic displacements. With the introduction of the idea of interface contraction, differential spatial loads are shown to be related to differential atomic displacements through a row-reduced Force Jacobian. Another improvement to the computation finally results in a fully contracted Jacobian that relates differential spatial loads to differential spatial displacements.

Now, the Force Jacobian $\frac{\partial T(k)}{\partial r(p)}$ must be coupled to the spatial displacement gradient described above. A matrix vector multiply performs this step, and scales with the number of bodies in the system, not the number of atoms.

The second term, $T_2(k)$, is given by:

$$T_2(k) = \sum_{i \in k} \frac{\partial \phi(k,i)}{\partial q_j} T(k,i)$$

and involves the original spatial loads $T(k)$ and derivative of the operator $\phi(k,i)$:

$$\phi(k,i) = \begin{bmatrix} \underline{\underline{E}}_3 & \tilde{\rho}(k,i) \\ \underline{\underline{0}}_3 & \underline{\underline{E}}_3 \end{bmatrix} \begin{bmatrix} {}^N C^k(k) & \underline{\underline{0}}_3 \\ \underline{\underline{0}}_3 & {}^N C^k(k) \end{bmatrix}^*$$

Thus

$$T_2(k) = \sum_{i \in k} \begin{bmatrix} {}^N dC^k(k) & \tilde{\rho}(k,i)\,{}^N dC^k(k) \\ \underline{\underline{0}}_3 & {}^N dC^k(k) \end{bmatrix}^* T(k,i)$$

where

$${}^N dC^k(k) = {}^N dC^k(i)\ {}^i C^k(k) + {}^N C^k(i)\ {}^i dC^k(k)$$

is computed recursively from the base body outward and

$${}^i dC^k(k) = \frac{\partial {}^i C^k(k)}{\partial q(k)} dq(k) \quad k = 1, \ldots n$$

where $dq(k)$ is defined in the next section, and $\frac{\partial {}^i C^k(k)}{\partial q_k}$, the partial derivative of the interbody direction cosine matrix is a function of the joint type connecting the bodies:

$$pin: \quad \frac{\partial {}^i C^k(k)}{\partial q_k} = -\underline{\underline{E}}_3 \sin(q_k) + \tilde{\lambda} \cos(q_k) + \lambda \lambda^* \sin(q_k)$$

$$slider: \quad \frac{\partial {}^i C^k(k)}{\partial q_k} = \underline{\underline{0}}_3$$

$$ball \text{ and } free: \quad \frac{\partial {}^i C^k(k)}{\partial \varepsilon_1} = 2 \begin{bmatrix} 0 & \varepsilon_2 & \varepsilon_3 \\ \varepsilon_2 & -2\varepsilon_1 & -\varepsilon_4 \\ \varepsilon_3 & \varepsilon_4 & -2\varepsilon_1 \end{bmatrix}$$

$$\frac{\partial {}^i C^k(k)}{\partial \varepsilon_2} = 2 \begin{bmatrix} -2\varepsilon_2 & \varepsilon_1 & \varepsilon_4 \\ \varepsilon_1 & 0 & \varepsilon_3 \\ -\varepsilon_4 & \varepsilon_3 & -2\varepsilon_2 \end{bmatrix}$$

$$\frac{\partial {}^i C^k(k)}{\partial \varepsilon_3} = 2 \begin{bmatrix} -2\varepsilon_3 & -\varepsilon_4 & \varepsilon_1 \\ \varepsilon_4 & -2\varepsilon_3 & \varepsilon_2 \\ \varepsilon_1 & \varepsilon_2 & 0 \end{bmatrix}$$

$$\frac{\partial {}^i C^k(k)}{\partial \varepsilon_4} = 2 \begin{bmatrix} 0 & -\varepsilon_3 & \varepsilon_2 \\ \varepsilon_3 & 0 & -\varepsilon_1 \\ -\varepsilon_2 & \varepsilon_1 & 0 \end{bmatrix}$$

In the next Section the Force Jacobians are combined with the Inertia Force Jacobians to finally form the Jacobian of the Residual Routine.

The Residual Jacobian

The previous Section describes the formation of the Force Jacobian $\frac{\partial T(k)}{\partial w(p)}$, which must be coupled to the spatial displacement gradient, in order to form the derivative of the spatial forces. This section describes the formation of the spatial displacement gradient and the formation of the Jacobian of the residual routine.

The recursive algorithms for computing the entire Jacobian are described. The Jacobian algorithm is not actually set up to compute the Jacobian. As is typical of automatic differentiation routines, it computes the matrix vector product $J_{uq}dq + J_{uu}du$ for arbitrary passed-in values of the vectors dq and du. In practice, to compute the Jacobian, the "Jacobian Routine" is effectively called repeatedly with a series of Boolean vectors (a vector with one entry set to 1 and all other entries set to zero.) Each call generates the corresponding column of the Jacobian. Note that some of the steps have already been or are being computed for the Residual Form method or the Direct Form method (the Forward Dynamics Calculations), but are reproduced here for clarity.

1. Given (q,u) compute $z \triangleq -M^{-1}\rho_u(q,u,0)$ using the Direct Form method. Also set $\dot{u} = z$ and recompute $A(k)$, then ρ_u, which recomputes $\hat{T}(k)$.
2. Perform contraction to compute the fully row- and column-reduced Force Jacobian, $\frac{\partial T(k)}{\partial w(p)}$ as described in section, "Interface Contraction":

$$\frac{\partial T}{\partial w} = \Phi \frac{\partial T}{\partial r} \Phi^*$$

Steps 3 through 10 below are used to fill the columns of J_{uq}:

3. Compute the analytic Jacobian partitions of the $\dot{q}$ terms:

$$J_{qq} = \frac{\partial W(q)}{\partial q} u$$

$$J_{qu} = W$$

using joint routines similar to those needed for the First Kinematics Calculations.

4. Compute q derivatives of position quantities and terms for spatial gradient:

Previously described methods were used to fill in certain joint-specific fields. These

quantities consisted of ${}^{i}C^{k}(k)$, the interbody direction cosine matrices, $r^{Q,Q_k}(k)$, the spanning vector for each body, and $H(k)$, the joint map for each body's inboard joint. To refer to the derivative of each of these quantities, a prefix '*d*' is added to the symbol name to make this reference generically. Thus, ${}^{i}dC^{k}(k)$ means the derivative of the direction cosine matrix ${}^{i}C^{k}(k)$.

Each interbody direction cosine matrix (and all the joint-specific) quantities depend only on the generalized coordinates of an individual joint. Thus, ${}^{i}dC^{k}(k)$ is nonzero only when the derivative is taken with respect to any of the coordinates for body *k*. To properly 'seed' the recursions being studying, a vector dq is passed in to the routine. For Jacobian computation we set one entry is set to 1, and all the other entries to 0. Then, the needed preliminary quantities are generated by a typical loop:

$$ {}^{i}dC^{k}(k) = \frac{\partial {}^{i}C^{k}(k)}{\partial q(k)} dq(k) \quad k = 1,\ldots n $$

The partial derivatives of the direction cosine matrices are generated analytically and displayed in the section, "Interface Contraction" above. These partial derivatives do *not* depend upon the particular column of the Jacobian that is being computed. Setting a particular entry of dq to 1, and all the rest to 0 has the effect of annihilating the correct subset of the seed quantities.

$\frac{\partial r^{Q,Q_k}(k)}{\partial q_k}$, the partial derivative of the interbody spanning vector is given by

$$ \frac{\partial r^{Q,Q_k}(k)}{\partial q_k} = \lambda(k), \quad slider $$

$$ \frac{\partial r^{Q,Q_k}(k)}{\partial q_k} = \underline{\underline{0}}_{3x4}, \quad ball $$

$$ \frac{\partial r^{Q,Q_k}(k)}{\partial q_k} = \underline{0}_3, \quad pin $$

$$ \frac{\partial r^{Q,Q_k}(k)}{\partial q_k} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad free $$

$\lambda(k)$ here refers to the body's sliding axis which connects it to its parent body.

$\frac{\partial H(k)}{\partial q_k}$, the partial derivative of the joint map is

$$\frac{\partial H(k)}{\partial q_k} = \underline{0}_6^*, \quad pin, slider$$

$$\frac{\partial H(k)}{\partial q_k} = \begin{bmatrix} \underline{\underline{0}}_3 & \underline{\underline{0}}_3 \end{bmatrix}, \quad ball$$

$$\frac{\partial H(k)}{\partial q_k} = \begin{bmatrix} \underline{\underline{0}}_3 & \underline{\underline{0}}_3 \\ \underline{\underline{0}}_3 & \dfrac{\partial {}^iC^k(k)}{\partial q_k} \end{bmatrix}, \quad free$$

With the above definitions of the partial derivatives, the recursions are seeded with the following loop:

for k = 1 to *nbod*

$${}^i dC^k(k) = \frac{\partial {}^iC^k(k)}{\partial q_k} dq(k)$$

$$dr^{Q_k Q_k}(k) = \frac{\partial r^{Q_k Q_k}(k)}{\partial q_k} dq(k)$$

$$dH(k) = \frac{\partial H(k)}{\partial q_k} dq(k)$$

end

After execution of these loops, all bodies have ${}^i dC^k(k)$, $dr^{Q_k Q_k}(k)$, and $dH(k)$, their interbody derivative quantities available.

One new quantity needed in the spatial displacement gradient computation is also computed. This is $\lambda(k)$ from the section on Interface Contraction, the rotation axis that generates the rate of change of orientation for each body outboard of a perturbed joint. Here, this variable is given the symbol $d\theta(k)$, the differential rotation axis for each body is

for k = 1 to nbod

$d\theta(k) = \lambda(k)\, dq(k), \quad pin$

$d\theta(k) = \underline{0}_3, \quad slider$

$d\theta(k) =$ not needed for *ball*, *free*

end

Since arbitrary perturbations to a set of Euler parameters do not produce a pure rotation, column contraction cannot be used when computing the corresponding column of the Jacobian. The row-reduced Force Jacobian can still (and must) be used..

After seeding the recursions, ${}^N dC^k(k)$, $dr^{OQ_k}(k)$, ${}^i d\phi^k(k)$, $d\lambda(k)$ are computed:

$^{N}dC^{k}(1) = {}^{i}dC^{k}(1)$

$dr^{OQ_k}(1) = dr^{Q_iQ_k}(1)$

$^{i}d\phi^{k}(1) = \underline{0}_6$

$d\lambda(1) = d\theta(1)$

for $k = 2$ to *nbod*

$^{N}dC^{k}(k) = {}^{N}dC^{k}(i)\,{}^{i}C^{k}(k) + {}^{N}C^{k}(i)\,{}^{i}dC^{k}(k)$

$dr^{OQ_k}(k) = dr^{OQ_i}(i) + {}^{N}dC^{k}(i)r^{Q_iQ_k}(k) + {}^{N}C^{k}(i)dr^{Q_iQ_k}(k)$

$$^{i}d\phi^{k}(k) = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$d\lambda(k) = {}^{i}C^{k*}(k)d\lambda(i) + d\theta(k)$

end

where

$a = {}^{i}dC^{k}(k)$, $b = d\tilde{r}^{Q_iQ_k}(k)\,{}^{i}C^{k}(k) + \tilde{r}^{Q_iQ_k}\,{}^{i}dC^{k}(k)$,

$c = \underline{0}_3$, $d = {}^{i}dC^{k}(k)$

5. Compute q derivatives of velocity:

A loop that computes the rate of change of joint velocity due to change in joint angle starts the process:

for $k = 1$ to *nbod*

$^{i}dv^{k}(k) = dH^{*}(k)u(k)$

end

This quantity is the rate of change of joint velocity due to change in joint angle. Obviously, it is nonzero only for joints whose map contains coordinate dependence. For free joints, the generalized speeds produce relative linear velocity that depends upon the joint orientation.

After computing $^{i}dv^{k}(k)$, $dV(k)$, the derivative of the spatial velocity of each body, is computed. This is done by the following loop:

$dV(1) = {}^{i}dv^{k}(1)$

for $k = 2$ to *nbod*

$dV(k) = {}^{i}d\phi^{k*}V(i) + {}^{i}\phi^{k*}dV(i) + {}^{i}dv^{k}(k)$

end

6. Couple Force Jacobian to spatial displacement gradient to compute $T_i(k)$

for k = 1 to *nbod*

$$T_i(k) = \sum_{p=1}^{nbod} \frac{\partial T(k)}{\partial w(p)} \frac{\partial w(p)}{\partial q_j}$$

end

7. Compute second term of the Force Jacobian $T_2(k)$ and append to $T_1(k)$:

for k= 1 to *nbod*

$$dT(k) = T_1(k) + \sum_{i \in k} \begin{bmatrix} {}^N dC^k(k) & \tilde{\rho}(k,i)\,{}^N dC^k(k) \\ \underline{0}_3 & {}^N dC^k(k) \end{bmatrix} T(k,i)$$

end

8. Compute q derivatives of acceleration-related terms:

Again the process starts with a loop that computes ${}^i da^k(k) = dH^*(k)\dot{u}(k)$:

for k = 1 to *nbod*

${}^i da^k(k) = dH^*(k)\dot{u}(k)$

end

This is the change in joint acceleration due to a change in coordinate. Then, $dA(k)$, the derivative of the spatial acceleration of each body is computed.

$$dA(1) = {}^i da^k(1)$$

$${}^i V^k(k) \rightarrow \begin{bmatrix} {}^i\omega^k(k) \\ {}^i v^{O_k}(k) \end{bmatrix}, \quad {}^i dV^k(k) \rightarrow \begin{bmatrix} {}^i d\omega^k(k) \\ {}^i dv^{O_k}(k) \end{bmatrix}$$

$$V(k) \rightarrow \begin{bmatrix} {}^N\omega^k(k) \\ {}^N v^{O_k}(k) \end{bmatrix}, \quad dV(k) \rightarrow \begin{bmatrix} {}^N d\omega^k(k) \\ {}^N dv^{O_k}(k) \end{bmatrix}$$

where $\rightarrow$ means the 6 vector is decomposed into two 3 vectors,

for k = 2 to *nbod*

$$dt1 \triangleq {}^i dC^{k*}(k)\left({}^N\omega^k(i) \times \left({}^N\omega^k(i) \times r^{OiO_k}(k)\right)\right) +$$

$${}^i C^{k*}(k)\begin{pmatrix} \left({}^N d\omega^k(i) \times \left({}^N\omega^k(i) \times r^{OiO_k}(k)\right)\right) + \\ \left({}^N\omega^k(i) \times \left({}^N d\omega^k(i) \times r^{OiO_k}(k)\right)\right) + \\ \left({}^N\omega^k(i) \times \left({}^N\omega^k(i) \times dr^{OiO_k}(k)\right)\right) \end{pmatrix}$$

$$da \triangleq {}^i d\phi^{k*}(k)A(i) + {}^i\phi^{k*}(k)dA(i) + \begin{bmatrix} \underline{0}_3 \\ dt1 \end{bmatrix}$$

$$\omega j \triangleq {}^i C^{k*}(k)\,{}^N\omega^k(i)$$

$$d\omega j \triangleq {}^i dC^{k*}(k)\,{}^N\omega^k(i) + {}^i C^{k*}(k)\,{}^N d\omega^k(i)$$

$$dt2 \triangleq d\omega j \times {}^i\omega^k(k) + \omega j \times {}^i d\omega^k(k)$$

$$dt3 \triangleq 2d\omega j \times {}^i v^{O_k}(k) + 2\omega j \times {}^i dv^{O_k}(k)$$

$$dA(k) = da + \begin{bmatrix} dt2 \\ dt3 \end{bmatrix} + {}^i da^k(k)$$

end

The symbols introduced here with $\triangleq$ are meant to be temporary variables not needed after computation of $dA(k)$.

After computing the spatial acceleration derivatives, the computation of $d\hat{T}(k)$, the spatial inertia force derivatives, is performed:

for $k = 1$ to $nbod$

$$dv1 \triangleq {}^{N}d\omega^{k}(k) \times \underline{\underline{\mathbf{I}}}_{0_k}(k) \cdot {}^{N}\omega^{k} + {}^{N}\omega^{k}(k) \times \underline{\underline{\mathbf{I}}}_{0_k}(k) \cdot {}^{N}d\omega^{k}$$

$$dv2 \triangleq {}^{N}d\omega^{k}(k) \times \left({}^{N}\omega^{k}(k) \times \mathbf{p}(k)\right) + {}^{N}\omega^{k}(k) \times \left({}^{N}d\omega^{k}(k) \times \mathbf{p}(k)\right)$$

$$d\hat{T}(k) = M(k)dA(k) + \begin{bmatrix} dv1 \\ dv2 \end{bmatrix} - d\hat{T}(k)$$

end

9. Compute $d\rho(k)$, the joint residual derivative for body k:

for $k = nbod$ to 1

$$d\rho(k) = dH(k)\hat{T}(k) + H(k)d\hat{T}(k) - d\sigma(k)$$

$$i = inb(k)$$

if i >0

$$d\hat{T}(i) = d\hat{T}(i) + {}^{i}d\phi^{k}(k)\hat{T}(k) + {}^{i}\phi^{k}(k)d\hat{T}(k)$$

end

end

After executing this routine, the values stored in $d\rho(k)$ are the new column of the Residual Jacobian $\frac{\partial}{\partial q}\rho_u(q,u,z)$.

10. Back-solve the $\frac{\partial \rho}{\partial q}$ result of previous step with the mass-matrix to obtain the desired $\frac{\partial \dot{u}}{\partial q}$:

$$\frac{\partial \dot{u}}{\partial q} = -M^{-1}\frac{\partial \rho}{\partial q}$$

The back-solve operation is accomplished in the Direct Form method routine by processing a residual vector into a $\dot{u}$ vector. The Second Kinematics Calculations only needs to be performed once for the whole Jacobian, since the back-solves are done at the nominal value of the state. In fact, the Second Kinematics routine must have been called in Step 1 while computing z, so the variables should still be cached.

Steps 11 through 13 below are used to fill the columns of J_{uu}:

11. Compute u derivatives of velocity:

This routine takes a passed-in vector du and computes ${}^{i}dv^{k}(k) = H^{*}(k)du(k)$. Then, $dV(k)$, the derivative of the spatial velocity of each body, is computed:

$$dV(1) = {}^{i}dv^{k}(1)$$
$$\text{for } k = 2 \text{ to } nbod$$
$$dV(k) = {}^{i}\phi^{k*}(k)dV(i) + {}^{i}dv^{k}(k)$$
$$\text{end}$$

12. Compute the velocity-induced derivative $d\hat{T}(k)$. As presented here, the routine is specialized for the case of no velocity dependent external loads. The surviving terms are those due to changes in inertia forces alone. Even if there were changes in external loads, it would only be required to include the contribution of $dT(k)$ as before.

$$dA(1) = \begin{bmatrix} 0_3 \\ 0_3 \end{bmatrix}$$
$$\text{for } k = 2 \text{ to } nbod$$
$$dt1 \triangleq {}^{i}C^{k*}(k)\begin{pmatrix} \left({}^{N}d\omega^{k}(i) \times \left({}^{N}\omega^{k}(i) \times r^{Q_i Q_k}(k)\right)\right) + \\ \left({}^{N}\omega^{k}(i) \times \left({}^{N}d\omega^{k}(i) \times r^{Q_i Q_k}(k)\right)\right) \end{pmatrix}$$
$$da \triangleq {}^{i}\phi^{k*}(k)dA(i) + \begin{bmatrix} 0_3 \\ dt1 \end{bmatrix}$$
$$\omega j \triangleq {}^{i}C^{k*}(k)\,{}^{N}\omega^{k}(i)$$
$$d\omega j \triangleq {}^{i}C^{k*}(k)\,{}^{N}d\omega^{k}(i)$$
$$dt2 \triangleq d\omega j \times {}^{i}\omega^{k}(k) + \omega j \times {}^{i}d\omega^{k}(k)$$
$$dt3 \triangleq 2d\omega j \times {}^{i}v^{Q_k}(k) + 2\omega j \times {}^{i}dv^{Q_k}(k)$$
$$dA(k) = da + \begin{bmatrix} dt2 \\ dt3 \end{bmatrix}$$
$$\text{end}$$

After computing the spatial acceleration derivatives, $d\hat{T}(k)$, the spatial inertia force derivatives, is computed:

for $k = 1$ to $nbod$

$dv1 \triangleq {}^{N}d\omega^{k}(k) \times \underline{\underline{\mathbf{I}}}_{Q_k}(k) \cdot {}^{N}\omega^{k} + {}^{N}\omega^{k}(k) \times \underline{\underline{\mathbf{I}}}_{Q_k}(k) \cdot {}^{N}d\omega^{k}$

$dv2 \triangleq {}^{N}d\omega^{k}(k) \times \left({}^{N}\omega^{k}(k) \times \mathbf{p}(k)\right) + {}^{N}\omega^{k}(k) \times \left({}^{N}d\omega^{k}(k) \times \mathbf{p}(k)\right)$

$$d\hat{T}(k) = M(k)dA(k) + \begin{bmatrix} dv1 \\ dv2 \end{bmatrix}$$

end

13. Compute $d\rho(k)$, the joint residual derivative for body k:

for $k = nbod$ to 1

$d\rho(k) = H(k)d\hat{T}(k) - d\sigma(k)$

$i = inb(k)$

if i >0

$d\hat{T}(i) = d\hat{T}(i) + {}^{i}\phi^{k}(k)d\hat{T}(k)$

end

end

After executing this routine the values stored in $d\rho(k)$ are the new column of the Residual Jacobian $\frac{\partial}{\partial u}\rho_u(q,u,z)$.

14. Back-solve the $\frac{\partial \rho}{\partial u}$ result of previous step with the mass-matrix to obtain the desired $\frac{\partial \dot{u}}{\partial u}$:

$$\frac{\partial \dot{u}}{\partial u} = -M^{-1}\frac{\partial \rho}{\partial u}$$

The back-solve operation is accomplished in the Direct Method routine by processing a residual vector into a $\dot{u}$ vector. The Second Kinematics Calculations only need to be performed once, since the back-solves are done at the nominal value of the state. In fact, the Second Kinematics routine must have been called in Step 1 while computing z, so the variables should still be cached.

The above steps complete the computation of the analytic Jacobian as long as the forces only have dependence on q. This accommodates the classical situation where all atomic forces are derivable from a potential function. To accommodate velocity-dependent forces, such as simple viscous damping, some of the above steps need to be modified as follows:

In Step 2 above, we also need to compute the contracted velocity Jacobian $\frac{\partial T(k)}{\partial \dot{r}(k)}$, which is block diagonal, must also be computed.

In Step 6 above, the computation of $T_1(k)$ must be augmented with the contracted velocity Jacobian:

for k = 1 to *nbod*

$$T_1(k) = \sum_{p=1}^{nbod} \frac{\partial T(k)}{\partial w(p)} \frac{\partial w(p)}{\partial q_j} + \sum_{i \in k} \frac{\partial T(k)}{\partial \dot{r}(k,i)} \frac{\partial \dot{r}(k,i)}{\partial q_j}$$

end

where

$$\frac{\partial \dot{r}(k,i)}{\partial q_j} = {}^N dC^k(k)\left[{}^N v^{Q_k}(k) + {}^N \omega^k(k) \times \rho(k,i)\right] + {}^N C^k(k)\left[{}^N dv^{Q_k}(k) + {}^N d\omega^k(k) \times \rho(k,i)\right]$$

A step is then added after Step 11, which is called Step 11a. This new step computes $dT(k)$:

$$dT(k) = \sum_{i \in k} \frac{\partial T(k)}{\partial \dot{r}(k,i)} \frac{\partial \dot{r}(k,i)}{\partial u_j}$$

where

$$\frac{\partial \dot{r}(k,i)}{\partial u_j} = {}^N C^k(k)\left[{}^N dv^{Q_k}(k) + {}^N d\omega^k(k) \times \rho(k,i)\right]$$

While executing Step 12 above, the last loop for $d\hat{T}(k)$ is modified by subtracting the velocity-dependent force derivative $dT(k)$:

for $k = 1$ to *nbod*

$$dv1 \triangleq {}^N d\omega^k(k) \times \underline{\underline{\mathbf{I}}}_{Q_k}(k) \cdot {}^N \omega^k + {}^N \omega^k(k) \times \underline{\underline{\mathbf{I}}}_{Q_k}(k) \cdot {}^N d\omega^k$$

$$dv2 \triangleq {}^N d\omega^k(k) \times \left({}^N \omega^k(k) \times \mathbf{p}(k)\right) + {}^N \omega^k(k) \times \left({}^N d\omega^k(k) \times \mathbf{p}(k)\right)$$

$$d\hat{T}(k) = M(k) dA(k) + \begin{bmatrix} dv1 \\ dv2 \end{bmatrix} - dT(k)$$

end

The rest of the Steps remain the same.

Fig. 6 summarizes the operational steps of the Analytic Jacobian method, which has been described in detail above.

Fig. 7 shows a plot of the accuracy of the numerical Jacobian versus the accuracy of the analytic Jacobian for an exemplary MD system. In the best case in which the perturbation was perfectly selected, the digits of accuracy for the generalized coordinates (q) and generalized speeds (u) from the numerical Jacobian, illustrated by line 152, were still only half that of the analytic Jacobian, illustrated by line 150.

ADDITIONAL EMBODIMENTS

The present invention has many embodiments besides the examples described above. The list below has other embodiments and applications:

- Order of Forces included in Jacobian

Any order of the forces to be included in the Jacobian, include, but not limited to Order(N), Order(N^2), Order(N^3), and Order(N^4). An example of an Order(N) force field would be an electrostatic force field using fast multi-pole expansion methods (see, for example, Greengaard, *The Rapid Evaluation of Potential Fields in Particle Systems*, Massachusetts Institute of Technology Dissertation, 1988) rather than direct method which is Order(N^2).

- Analytic Jacobian for Direct Form

When the governing equations are in Direct Form, the so-called "forward dynamics" form of the equations is obtained. In this form, the equations process a state vector and applied efforts and generate the acceleration at each of the joints modeled in the system.

$$\dot{u} = M^{-1}(f)$$

The Jacobian then represents the partial derivatives of the accelerations with respect to elements of the state vector. The preferred embodiment shows several algorithmic methods for computation of these partial derivatives. The methods are exact and do not utilize numerical approximations to form derivatives.

The Direct Form produces the $\dot{u}$ partitions of the Jacobian:

$$J_{uq} = \frac{\partial\left(M^{-1}(f)\right)}{\partial q}$$

and

$$J_{uu} = \frac{\partial\left(M^{-1}(f)\right)}{\partial u}$$

by using an algorithmic counterpart to the function which computes the $\dot{u}$ function.

The computation of $M^{-1}f$ is accomplished in Order(N) floating point operations (flops) by utilizing the operational inverse of the Mass Matrix:

$$M^{-1} = [I - H\Psi K]D^{-1}[I - H\Psi K]^T$$

where all the block matrices are defined previously in the Second Kinematics Calculations, and each factor represents an operator that can be applied to an n-vector in Order(N) flops. Since $\dot{u} = [I - H\Psi K]D^{-1}[I - H\Psi K]^T f$, from the chain rule:

$$\begin{aligned} J_{uq} = {} & [I - H\Psi K]D^{-1}[I - H\Psi K]^T \partial f / \partial q + \\ & \partial([I - H\Psi K]) / \partial q \; D^{-1}[I - H\Psi K]^T f + \\ & [I - H\Psi K] \; \partial(D^{-1}) / \partial q \; [I - H\Psi K]^T f + \\ & [I - H\Psi K]D^{-1} \; \partial([I - H\Psi K]^T) / \partial q \; f \end{aligned}$$

and similarly for J_{uu}.

Thus the present invention can be used for equations of the Direct Form in producing algorithms that compute each of the above terms in closed form.

Hence the present invention provides many advantages. Analytical Jacobians are much more accurate (with twice the number of significant digits). Computing from the Residual Form instead of Direct Form is much more efficient. The "Contraction" of rows and columns from "number of atoms" to "number of bodies" reduces the size of the force Jacobian matrices. Jacobian computations are of the same order as computation of forces, rather than an extra order higher if each column has to be perturbed. Thus, if the forces are computed in Order(N^3) operations, for example, a numerical Jacobian requires Order(N^4), whereas an analytic Jacobian requires only Order(N^3) operations. By controlling the range of loop structure in Jacobian calculations, computations can reduced even further (just compute for outboard bodies).

Therefore, while the foregoing is a complete description of the embodiments of the invention, it should be evident that various modifications, alternatives and equivalents may be made and used. Accordingly, the above description should not be taken as limiting the scope of the invention which is defined by the metes and bounds of the appended claims.

WHAT IS CLAIMED IS:

1. A method of modeling the behavior of a molecule, comprising

selecting a torsion angle, rigid multibody model for said molecule, said model having equations of motion;

selecting an implicit integrator; and

generating an analytic Jacobian for said implicit integrator to integrate said equations of motion so as to obtain calculations of said behavior of said molecule.

2. The method of claim 1 wherein said analytic Jacobian is derived from an analytic Jacobian of the Residual Form of the equations of motion.

3. The method of claim 2 wherein said analytic Jacobian J comprises

$$J = \begin{pmatrix} \frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial u} \\ \frac{\partial \dot{u}}{\partial q} & \frac{\partial \dot{u}}{\partial u} \end{pmatrix} \triangleq \begin{pmatrix} J_{qq} & J_{qu} \\ J_{uq} & J_{uu} \end{pmatrix}; \text{ and}$$

$$J_{qq} = \frac{\partial \dot{q}}{\partial q} = \frac{\partial (Wu)}{\partial q} \text{ and } J_{qu} = \frac{\partial \dot{q}}{\partial u} = W$$

$$J_{uq} = \frac{\partial \dot{u}}{\partial q} = -M^{-1} \frac{\partial \rho_u(q,u,z)}{\partial q} \text{ and } J_{uu} = \frac{\partial \dot{u}}{\partial u} = -M^{-1} \frac{\partial \rho_u(q,u,z)}{\partial u}$$

where q are the generalized coordinates, u are the generalized speeds, W is a joint map matrix and M is the mass matrix and ρ_u is the dynamic residual of the equations of motion, and z is $-M^{-1}\rho_u(q,u,0)$.

4. The method of claim 3 wherein said implicit integrator selecting step comprises an L-stable integrator.

5. A method of simulating the behavior of a physical system, comprising

modeling said physical system with a torsion angle, rigid multibody model, said model having equations of motion; and

integrating said equations of motion with an implicit integrator; said implicit integrator having an analytic Jacobian to obtain calculations of said behavior of said physical system.

6. The method of claim 5 wherein said analytic Jacobian is derived from an analytic Jacobian of the Residual Form of the equations of motion.

7. The method of claim 6 wherein said analytic Jacobian J comprises

$$J = \begin{pmatrix} \frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial u} \\ \frac{\partial \dot{u}}{\partial q} & \frac{\partial \dot{u}}{\partial u} \end{pmatrix} \triangleq \begin{pmatrix} J_{qq} & J_{qu} \\ J_{uq} & J_{uu} \end{pmatrix}; \text{ and}$$

$$J_{qq} = \frac{\partial \dot{q}}{\partial q} = \frac{\partial (Wu)}{\partial q} \text{ and } J_{qu} = \frac{\partial \dot{q}}{\partial u} = W$$

$$J_{uq} = \frac{\partial \dot{u}}{\partial q} = -M^{-1} \frac{\partial \rho_u(q,u,z)}{\partial q} \text{ and } J_{uu} = \frac{\partial \dot{u}}{\partial u} = -M^{-1} \frac{\partial \rho_u(q,u,z)}{\partial u}$$

where q are the generalized coordinates, u are the generalized speed, W is a joint map matrix and M is the mass matrix and ρ_u is the dynamic residual of the equations of motion, and z is $-M^{-1}\rho_u(q,u,0)$.

8. The method of claim 7 wherein said implicit integrator comprises an L-stable integrator.

9. Computer code for simulating the behavior of a molecule, said code comprising

a first module for a torsion angle, rigid multibody model of said molecule, said model having equations of motion; and

a second module for an implicit integrator to integrate said equations of motion with an analytic Jacobian to obtain calculations of said behavior of said molecule.

10. The computer code of claim 9 wherein said analytic Jacobian is derived from an analytic Jacobian of the Residual Form of the equations of motion.

11. The computer code of claim 10 wherein said analytic Jacobian J comprises

$$J = \begin{pmatrix} \frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial u} \\ \frac{\partial \dot{u}}{\partial q} & \frac{\partial \dot{u}}{\partial u} \end{pmatrix} \triangleq \begin{pmatrix} J_{qq} & J_{qu} \\ J_{uq} & J_{uu} \end{pmatrix}; \text{ and}$$

$$J_{qq} = \frac{\partial \dot{q}}{\partial q} = \frac{\partial (Wu)}{\partial q} \text{ and } J_{qu} = \frac{\partial \dot{q}}{\partial u} = W$$

$$J_{uq} = \frac{\partial \dot{u}}{\partial q} = -M^{-1} \frac{\partial \rho_u(q,u,z)}{\partial q} \text{ and } J_{uu} = \frac{\partial \dot{u}}{\partial u} = -M^{-1} \frac{\partial \rho_u(q,u,z)}{\partial u}$$

where q are the generalized coordinates, u are the generalized speed, W is a joint map matrix and M is the mass matrix and ρ_u is the dynamic residual of the equations of motion, and z is $-M^{-1}\rho_u(q,u,0)$.

12. The computer code of claim 11 wherein said implicit integrator comprises an L-stable integrator.

13. Computer code for simulating the behavior of a physical system, said code comprising

a first module for a torsion angle, rigid multibody model of said system, said model having equations of motion; and

a second module for an implicit integrator to integrate said equations of motion with an analytic Jacobian to obtain calculations of said behavior of said system.

14. The computer code of claim 13 wherein said analytic Jacobian is derived from an analytic Jacobian of the Residual Form of the equations of motion.

15. The computer code of claim 14 wherein said analytic Jacobian J comprises

$$J = \begin{pmatrix} \frac{\partial \dot{q}}{\partial q} & \frac{\partial \dot{q}}{\partial u} \\ \frac{\partial \dot{u}}{\partial q} & \frac{\partial \dot{u}}{\partial u} \end{pmatrix} \triangleq \begin{pmatrix} J_{qq} & J_{qu} \\ J_{uq} & J_{uu} \end{pmatrix}; \text{ and}$$

$$J_{qq} = \frac{\partial \dot{q}}{\partial q} = \frac{\partial (Wu)}{\partial q} \text{ and } J_{qu} = \frac{\partial \dot{q}}{\partial u} = W$$

$$J_{uq} = \frac{\partial \dot{u}}{\partial q} = -M^{-1} \frac{\partial \rho_u(q,u,z)}{\partial q} \text{ and } J_{uu} = \frac{\partial \dot{u}}{\partial u} = -M^{-1} \frac{\partial \rho_u(q,u,z)}{\partial u}$$

where q are the generalized coordinates, u are the generalized speed, W is a joint map matrix and M is the mass matrix and ρ_u is the dynamic residual of the equations of motion, and z is $-M^{-1}\rho_u(q,u,0)$.

16. The computer code of claim 15 wherein said implicit integrator comprises an L-stable integrator.

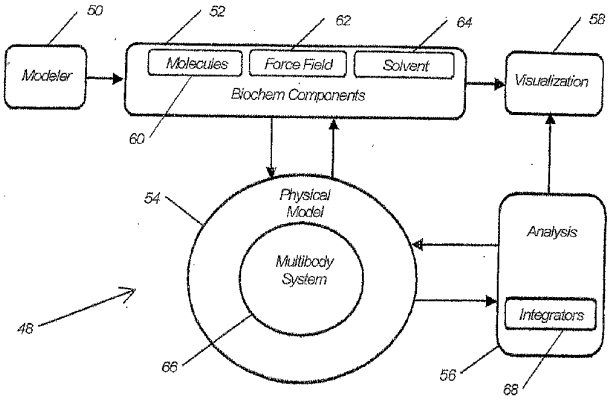

Fig. 1

Residual Form Method to compute ρ_q and ρ_u	Direct Form Method to compute $\dot{q}$ and $\dot{u}$
1. Compute the First Kinematics Calc. and the kinematic residual $\rho_q(k)$ 2. Generate $\hat{T}(k)$, the spatial load balance for each body 3. Compute dynamic residual $\rho_u(k)$	1. Compute $\dot{q}$ using joint specific routines 2. Perform First Kinematics Calc. with $\dot{u}=0$ 3. Generate residuals ρ_u and negate $\rho_u = -\rho_u$ 4. Perform Second Kinematics Calc. 5. Compute $\dot{u}$ using Forward Dynamics

Comparison of Methods

Fig. 5

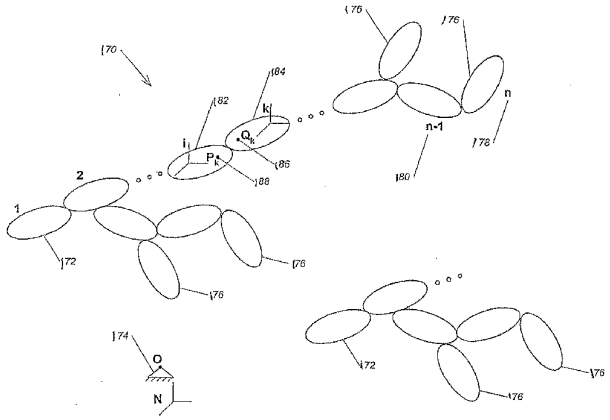
170
176
176
184
182
k
n
n-1
178
180
i
Q_k
P_k
186
188
2
1
172
176
176
174
O
N
172
176
176

Fig. 2

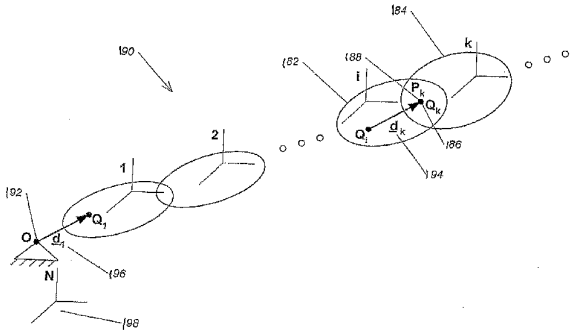
190
184
k
182
188
i
P_k
Q_k
d_k
Q_i
186
194
2
1
192
Q_1
O
d_1
N
196
198

Fig. 3

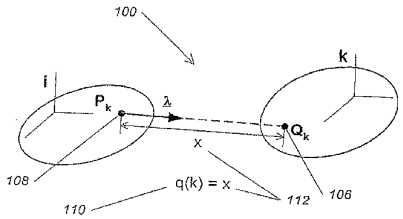

Fig. 4A

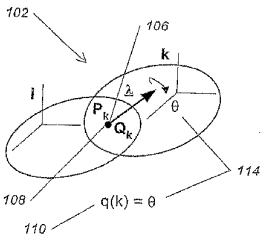

Fig. 4B

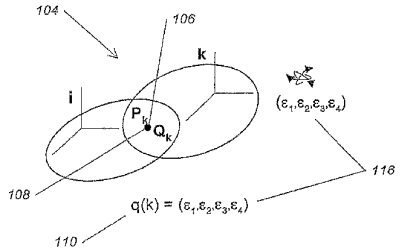

Fig. 4C

Analytic Jacobian Method

1. Compute the analytic Jacobians of the kinematics routines:

$$J_{qq} = \frac{\partial (Wu)}{\partial q} \quad \text{and} \quad J_{qu} = W$$

2. Compute $z \triangleq -M^{-1}\rho_u(q,u,0)$ using the Direct Method

3. Compute the analytic Jacobians of the dynamics Residual routine:

$$\frac{\partial}{\partial q}\rho_u(q,u,z) \text{ and } \frac{\partial}{\partial u}\rho_u(q,u,z).$$

4. Backsolve for the analytic Jacobian of the dynamics routine using results for z from the Second Kinematics step:

$$J_{uq} = \frac{\partial \dot{u}}{\partial q} = -M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial q} \quad \text{and}$$

$$J_{uu} = \frac{\partial \dot{u}}{\partial u} = -M^{-1}\frac{\partial \rho_u(q,u,z)}{\partial u}$$

Fig. 6

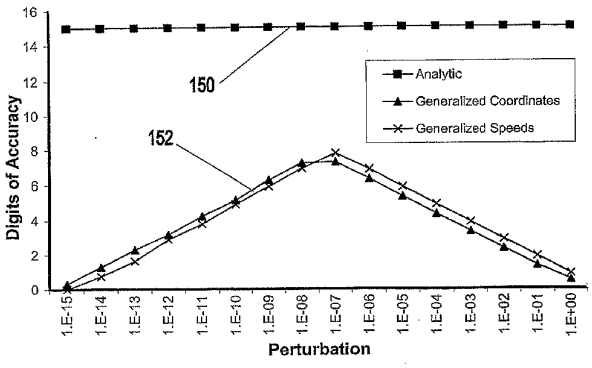

Fig. 7

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/51360

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G 06 F 19/00
US CL : 702/27, 703/2, 707/1-10
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)
U.S. : 702/27, 703/2, 707/1-10

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Non-Patent Literature

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WEST, STN, USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X, P	US 6,150,179 A (WENT) 21 November 2000. See column 2, lines 11-12 and 34-48; column 48, lines 10-14, 30-33 and 47-52; column 52, lines 5-9 and 24-31; column 54, lines 12-67; column 55, lines 1-67; column 56, lines 1-67; column 57, lines 1-56.	1-16
X	US 5,799,312 A (RIGOUTSOS) 25 August 1998. See column 14, lines 66-67; column 21, lines 35-39; column 35, lines 63-67 and column 36, lines 1-6.	1-16
Y, P	US 6,253,166 B1 (WHITMORE et al.) 26 June 2001. See entire document, especially column 23, lines 66-67 and column 24, lines 1-18.	1-16
Y,P	MOROKUMA et al. Model studies of the structures, reactivities, and reaction mechanisms of metalloenzymes. IBM J. RES & DEV. May-July 2001, Volume 45, Nomber 3/4, pages 367-395. See page 384, column 2, lines 35-38.	1-16

☐ 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 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
19 May 2002 (19.05.2002)	14 JUN 2002
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer Nikolai M Galitsky, PhD.
Facsimile No. (703)305-3230	Telephone No. (703)308-0196

Form PCT/ISA/210 (second sheet) (July 1998)

フロントページの続き

(31)優先権主張番号 60/245,734
(32)優先日 平成12年11月2日(2000.11.2)
(33)優先権主張国 米国(US)

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,CH,CY,DE,DK,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NO,NZ,OM,PH,PL,PT,RO,RU,SD,SE,SG,SI,SK,SL,TJ,TM,TR,TT,TZ,UA,UG,US,UZ,VN,YU,ZA,ZW

(72)発明者 ローゼンサル, ダン イー.
アメリカ合衆国 カリフォルニア ９４０２４, ロス アルトス, エッジ レーン ７１８