USING AN EXTENSIBLE OBJECT-ORIENTED QUERY LANGUAGE IN MULTIBODY SYSTEM ANALYSIS

Claes Tisell
Machine Elements
Department of Machine Design
Royal Institute of Technology
Stockholm, Sweden

Kjell Orsborn
Intelligent Engineering Systems Laboratory
Department of Civil & Environmental Engineering
Massachusetts Institute of Technology
Cambridge, USA

Abstract

As the modern software tools produce large amount of engineering data the demand for efficient data management may be met by integrating database technology with engineering applications. This approach is taken in MECHAMOS, which is a previously reported system for symbolic and numeric multibody system (MBS) analysis. This work focuses on the high level analysis performed with the available query language in MECHAMOS. The data management is considerably improved in this system compared to a traditional MBS analysis tool. For instance MECHAMOS can easily combine and compare MBS data not only within the same MBS model but also over several MBS models each governing different equations of motion. To avoid redundant computations in such analyses a simplified materialisation mechanism is implemented. Examples are given on combining and comparing both symbolic and numeric MBS data

1. Introduction

The rapid development of computer technology and software tools has enabled engineers to build larger models and to perform more advanced analyses on these models. Consequently this generates large amounts of heterogeneous engineering data. Therefore sharing and combining heterogeneous engineering data as well as transforming engineering data to a suitable form for further analysis have become important issues in the design process today. To meet these demands, full availability of data and efficient data access are important. One way to accomplish this is to access data through a query language, which is faster and requires less coding than using a conventional programming language [1]. This will add to the requirements on future generation engineering applications to supply database technology and a general query language for data management.

In the field of multibody system (MBS) analysis the development of computer technology has, apart from larger models, also opened the possibility to perform the analysis in the symbolic domain and move on to the numerical domain in a later stage of the analysis. A previously reported system for MBS analysis based on object-relational database technology [2] shows how the future requirements can be met with increased availability of MBS data and how the data management is facilitated. In this system, named MECHAMOS, symbolic and numeric MBS data is fully available through a general query language and can be put into a suitable form for further analysis. A similar approach is taken in [3] where a system for finite element analysis (FEAMOS) is based on the same database technology as MECHAMOS.

This work focus on the MBS analysis in MECHAMOS where large amount of MBS data can be generated, compared and searched for by taking advantage of the data management capabilities and the extensible query language integrated into the application.

2. The MECHAMOS System

MECHAMOS is an MBS analysis tool based on the object-relational database management system (ORDBMS) AMOS II [4-6]. This provides MECHAMOS with a general object oriented and extensible query language (QL) for accessing the MBS data. In MECHAMOS the mathematical capabilities of AMOS II has been extended with Matlab [7] and MapleV [8] enabling the system to perform both numerical and symbolical MBS analysis. MBS data is passed from the query language to Matlab and Maple through a general application-programming interface using LISP, C and Java and is thus not based on file transfer. However, some MBS results are stored on file due to limitations in the architecture of Matlab and Maple. In Figure 1 the MECHAMOS system is illustrated with base data and the extended mathematical capabilities through Matlab and MapleV.

The MBS analysis is based on Kane's equations which aims towards a minimal set of constraint reaction free equations of motion in a minimal set of generalised coordinates. To obtain these equations MECHAMOS utilise the SOPHIA system [9-10], which is a set of routines, implemented in MapleV for vector algebra and vector analysis including routines supporting Kane's equations. This provides MECHAMOS with an efficient tool for formulating the equations of motion on symbolic form and the resulting equations is put into a form for efficient numeric evaluation.

The MBS data is divided into MBS concepts in a basic form called *base data* and is stored in the database. Further, the database also contains MBS concepts derived from these base data. The former can be further divided into data on component level and data on system level. On the component level six quantities has to be stored for each rigid body in the mechanism:

$$m \quad \mathbf{r} \quad \mathbf{F}_a \quad \mathbf{J}_{\mathrm{A}} \quad fA \quad \mathbf{M}_a \tag{1}$$

where the first three items represent the translational part and are the *mass*, *position vector* of the centre of mass and the applied *force vector*. The representations of the rotational part are the *moments of inertia* dyad, a body fixed *reference frame* and the applied *moment vector*. On system level the *generalised coordinates* (*q*), *generalised speeds* (*u*) and their relations (kinematic differential equations, kde) has to be declared. Further, the gravity vector and reference frames are also defined on system level. To enable numerical analysis and simulations, different sets of numerical values of the parameters and initial values are also defined.

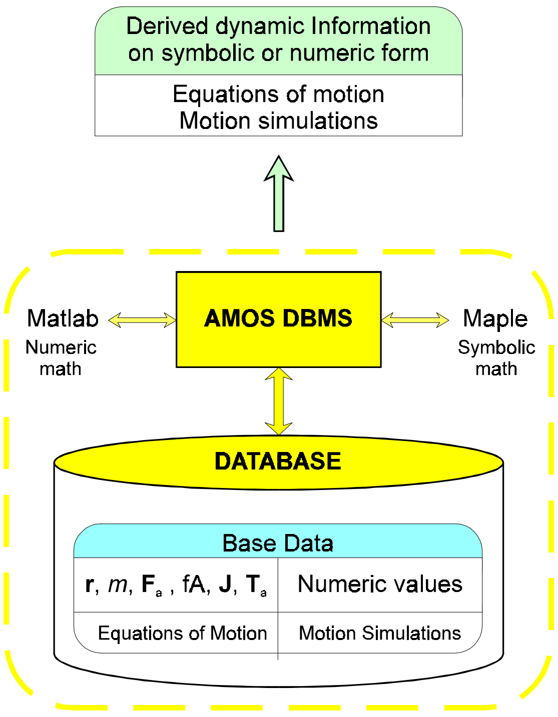

Figure 1. *The MECHAMOS system selects dynamic information derived from base data through a general QL.*

3. Data Retrieval and Materialisation

With a system based on this technology large amounts of MBS data (velocities, momentum, potential and kinetic energies, trajectories etc.) can be easily and automatically generated through the QL. This MBS data can be retrieved on many different forms. A *scalar* quantity is of course represented on scalar form whereas a *vector* quantity is either a vector representation in a given reference frame or a scalar representing the magnitude of the vector. In the combined *symbolic* and *numeric* domains, MBS data can be retrieved on pure symbolic form or with the numeric values of the parameters substituted and thus maintaining the state variables and their derivatives on symbolic form. Selecting scalars and vectors on pure numeric form requires a numeric solution of the equations of motion (commonly referred to as a simulation). The retrieved object is then a numeric sequence with one value for each time step of the integration.

Different forms of MBS data require various computational efforts. For instance, to obtain the velocity vector on symbolic form for a given body *B* in a given MBS model the necessary sequence of operations is

*B*11 declare time dependent coordinates (*q* and *u*)
*B*12 declare the kinematic differential equations
*B*13 declare relations between reference frames
*B*14 differentiate the position vector with respect to time and relative to the inertial frame

For MBS models with reasonable degrees of freedom these symbolic computations (*B*1*x*) are usually simple and fast to execute. To select the same velocity vector on pure numerical form, a simulation has to be performed for a given set of parameter values and a given set of initial conditions. The necessary sequence of operations for the numerical case (*B*2*x*) is shown below where each of the *B*21 and *B*22 operations usually involves costly computations.

*B*21 derive the equations of motion for the MBS model on symbolic form
*B*22 solve these equations numerically to obtain numeric values for the state space variables (*q* and *u*)
*B*23 get the velocity vector symbolically (*B*11-*B*14 above)
*B*24 evaluate the symbolic vector numerically for each time step of the integration

In a traditional MBS analysis software the user accesses the MBS data through a command menu structure that usually activates function calls in some programming language. The possibility of composing user-defined macros is also supported in most traditional software. This type of software usually allow analysis of one MBS model at a time and thus perform *B*11-*B*13 and *B*21-*B*22 once in the beginning of the session. Various MBS data is then derived from the model and with the current set of numerical parameter values (e.g. *B*23-*B*24 for different vectors).

In comparison, MECHAMOS has several MBS models that are simultaneously available in the database and the MBS data can be accessed or derived on a higher level through a general query language. To derive the velocity vector of body *B* on symbolic form the following AMOSQL query is processed

```
MA1> SELECT velvec(b)
     FROM  body b
     WHERE name(b) = "B"
       AND name(model_oid(b)) =
           "Sliding Pendulum";
OID[0x0:2474]
MA2>
```

where `MA1>` is the prompt and the number indicates the generation of the database population. The query searches the database for all objects of type `body` where the attribute `name` equals `"B"` and this object is related to an object of type `MBS_model` named `"Sliding Pendulum"`. For that object of type `body` the derived function `velvec` derives the velocity vector using the information stored in the object. The result is an object identifier (OID) to an object of the type `euclidean_vector` containing the velocity vector on symbolic form and the reference frame in which the vector is represented. It is the derived function `velvec` that performs the entire sequence of operations to derive the vector by further AMOSQL queries (i.e. *B*11-*B*14).

To derive the magnitude of the velocity vector on pure numeric form, the name of a set of parameter values is also required. This is done to enable numeric integration of the equations. The select statement that derives the vector is shown below where the derived function `velvec` performs *B*23 in the symbolic world whereas `numseq` first perform *B*21 in the symbolic world and then moves to the numeric world by performing the operations *B*22 and *B*24.

```
MA2> SELECT numseq(magvec(velvec(b)), n)
     FROM  body b, mbs_numeric n
     WHERE name(b) = "B"
       AND name(model_oid(b)) =
           "Sliding Pendulum"
       AND name(n) = "s20"
       AND name(model_oid(n)) =
           "Sliding Pendulum";
OID[0x0:2478]
MA3>
```

The result is an object identifier (OID) to an object of the type `scalar_sequence` containing the magnitude of the velocity vector on pure numeric form with one numeric value for each time step of the integration.

To illustrate the need and the importance of *function materialisation* [11] in the DBMS consider another body, named *C*, in the same MBS model. Assume that the velocity vector of this body *C* is selected on pure numerical form with the same set of parameter values and initial conditions as for body *B* above. The corresponding operations to perform this task are named *C*11-*C*14 and *C*21-*C*24 respectively. The computations in *C*11-*C*13 and *C*21-*C*22 are then identical with their body *B* counterparts (assuming that the involved base data has not changed between the occasions). It is thus not necessary to perform *C*11-*C*13 and *C*21-*C*22 when the second velocity vector is derived.

In a traditional MBS software this is normally taken care of by the user since he/she performs the operations in an appropriate sequence and the users insight that the equations and the numerical results of the simulation are the same in the two cases. In a system like MECHAMOS, where MBS data is accessed and derived on a higher level and only when needed for a specific query, the result of the query must be independent of previously posed queries and results. This implies that each query has to derive all the necessary data and thus, the entire sequence of operations must be performed through nested subqueries.

With function materialisation supported by the DBMS, the query results are temporarily saved. If a query is posed a second time and the relevant base data is not changed the DBMS looks up the result of the query rather than process the query again which means that unnecessary computation can be avoided.

Currently, true functional materialisation is not supported in AMOS II. However a specialised temporary storage mechanism which provides a simplified materialisation functionality has been implemented in MECHAMOS. This implementation covers the most frequent operations (*B*11-*B*13) and the most computational costly operations (*B*21 and *B*22). It is obtained by introducing an object keeping track of which MBS model or numeric data that occupies the Maple and Matlab workspace. It also includes a file numbering mechanism to handle the part of the MBS results stored on file. True functional materialisation should also handle the ability of detecting a change in the stored data and to determine which computational results that are effected by that change in data and thus no longer valid. This is not implemented in MECHAMOS.

The conclusion from this section is that the derived functions in the database have to include the entire sequence of operations required to select the MBS data, e.g. B21-B24. Materialisation capabilities and query optimisation in the DBMS will then select the necessary operations, detect the queries already processed and do the operations in an efficient order.

4. Examples of Combining and Comparing MBS data

To illustrate how MBS data is combined and compared in MECHAMOS a planar sliding pendulum is taken as an example. The system (illustrated in Figure 2) has two degrees of freedom described by two generalised coordinates (q_1, q_2) and two generalised speeds (u_1, u_2) with the *kde* chosen on the simplest form, i.e. $\dot{q}_j = u_j$. A spring-damper system with zero equilibrium length is inserted between a moving wall and the slider. The motion of the wall is governed by a constrained function in time and described by

$$f_1(t) = A_0 \sin(\omega\, t) \tag{2}$$

This yields the applied force vector on the slider (body *A*)

$$\mathbf{F}_a^{<A} = k_1\big(f_1(t) - q_1\big)\mathbf{n}_1 + c_1\big(\dot{f}_1(t) - u_1\big)\mathbf{n}_1 - m_A g \mathbf{n}_2 \tag{3}$$

The other required base data for each rigid body according to Equation (1) is easily formulated by inspecting the illustration in Figure 2.

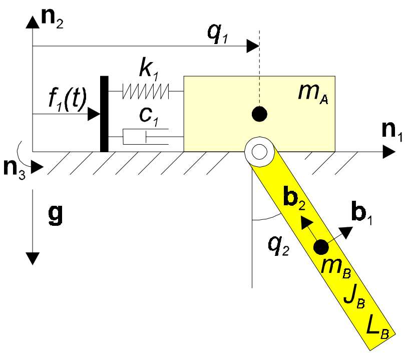

Figure 2. *A sliding pendulum example.*

The parameters in the model are {m_A, m_B, J_B, L_B, k_1, c_1, A_0, ω, g} and their numeric values are all equal 1 except for g = 9.81 [m/s^2] and ω ranging from 0.1 to 4.0 [rad/s]. This results in a total of 29 numerical sets (with only ω varying). The initial conditions are all zero and the integration time corresponds to approximately 10 cycles. The absolute and relative tolerances are 10^{-6} and 10^{-3} respectively. Note that the dimensions of the slider are not effecting the dynamics since the slider is constrained to translate only. This concludes the definition of base data to enable the MECHAMOS system to perform MBS analysis. A more thoroughly discussion of the required base data is found in [2] where a *crank-roller* is taken as an example. A shortened version of the text file that populates MECHAMOS with this model is found in Appendix 1. Most of the numerical sets are here left out.

The sliding pendulum example was used earlier to illustrate the selection statements in the previous section. Looking at the resulting object of the first query we find the velocity vector on symbolic form represented in frame *fB*

```
MA3> representation(OID[0x0:2474]);
{"cos(q2)*u1+1/2*u2*LB","-sin(q2)*u1",0}
MA3> name(frame_oid(OID[0x0:2474]));
"fB"
MA4>
```

The second query, the magnitude of the vector on pure numeric form, is graphically represented in Figure 3. This plot is obtained by

```
MA4> plot(OID[0x0:2480],OID[0x0:2478]);
MA4>
```

assuming that the sequence of times of the integration points is stored in the object `numeric_sequence` with the object identifier `OID[0x0:2480]`. The title is obtained manually in a command like fashion through the QL.

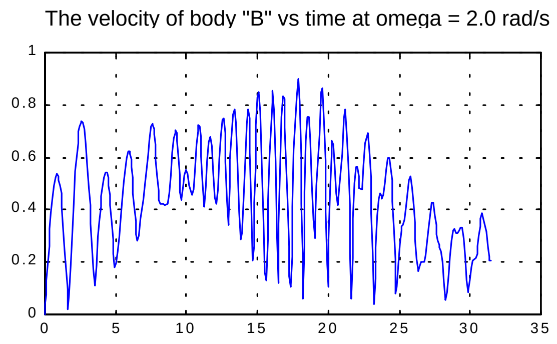

Figure 3. *A graphic representation of the numeric sequence representing the velocity of the pendulum.*

In the remaining of this section further examples are given on selecting MBS data for the sliding pendulum model through the QL. Selecting MBS data usually involve combination and comparison of data and is not as strict as may be indicated by the subsections below.

4.1 Combining Symbolic MBS data

The ability to combine data is important in MBS analysis. Symbolic MBS data is usually combined within an MBS model where data for different bodies can be combined to obtain data for a system of bodies. Typically the derivation of the equations of motion involves the combination of data from individual bodies in the mechanism and their interrelations to obtain equations for the entire set of bodies. Combining and comparing symbolic MBS data over several models requires some *submodel* strategy to avoid violation of coordinate and reference frame definitions.

As a first example of combining symbolic MBS data, the centre of mass for the mechanism is retrieved through the QL. The common centre of mass for a group of bodies is simply a mass-weighted mean value of the position vectors and is thus given by

$$\mathbf{r}^* = \frac{\sum m_i \mathbf{r}^{<i}}{\sum m_i} \tag{4}$$

The position vector, $\mathbf{r}^{<i}$, of the *i*:th body is given by the derived function `posvec(body)` and its mass, m_i, by the stored function `mass_oid(body)`. Selecting and combining the appropriate data according to Equation (4) is obtained with the AMOSQL query

```
MA4> SELECT div(sum(ev),sum(so))
     FROM  mbs_model m,
              bag of euclidean_vector ev,
              bag of scalar_object so
     WHERE name(m) = "Sliding Pendulum"
       AND ev = (SELECT mul(mass_oid(b),
                            posvec(b))
                 FROM  body b
                 WHERE model_oid(b) = m)
       AND so = (SELECT mass_oid(b)
                 FROM  body b
                 WHERE model_oid(b) = m);
OID[0x0:2487]
MA5>
```

This query has two subqueries that determine the nominator and the denominator and both returns a bag of objects. A bag is a collection of objects in no specific order. These bags are summed separately by the overloaded aggregation operator `sum` and then the nominator vector is divided by the denominator scalar. The result is an object of type `euclidean_vector` containing the position vector of the centre of mass. Two of the stored functions of the object reveals the vector on symbolic form represented in frame *fB*.

```
MA5> representation(OID[0x0:2487]);
{"cos(q2)*q1" , "-1/2*(2*sin(q2)*mA*q1+
  2*mB*sin(q2)*q1+mB*LB)/(mA+mB)" , 0}
MA5> name(frame_oid(OID[0x0:2487]));
"fB"
MA6>
```

Another example of combining MBS data is to determine the kinetic energy for a selected group of bodies. For each body in the MBS system the kinetic energy consists of a translational part and a rotational part

$$K_i = \frac{m_i}{2}\mathbf{v}^{<i}\cdot\mathbf{v}^{<i} + \frac{1}{2}\boldsymbol{\omega}^{<i}\cdot\mathbf{J}_i\cdot\boldsymbol{\omega}^{<i} \tag{5}$$

This equation is implemented as a derived function for the object type `body` named `kinegy(body)`. The object contains information and refers to other objects containing the information necessary to derive the kinetic energy (i.e. mass, velocity, angular velocity and moments of inertia). For a system of bodies, the total kinetic energy is the sum of the kinetic energy for the individual bodies expressed as

$$K = \sum K_i \tag{6}$$

and is retrieved by the following AMOSQL query

```
MA6> SELECT sum(so)
     FROM  bag of scalar_object so,
              mbs_model m
     WHERE name(m) = "Sliding Pendulum"
       AND so = (SELECT kinegy(b)
                 FROM  body b
                 WHERE model_oid(b) = m);
OID[0x0:2515]
MA7> scalar(OID[0x0:2515]);
"1/2*mB*cos(q2)*u1*u2*LB + 1/8*mB*u2^2*LB^2 +
 1/2*mB*u1^2 + 1/2*u2^2*JB + 1/2*mA*u1^2"
MA7>
```

This query returns an OID to a `scalar_object` containing the total kinetic energy for the selected system of bodies on symbolic form. In this case this is all the bodies in the MBS model "`Sliding Pendulum`" and the query can be defined as a derived function on the `mbs_model` type as

```
MA7> CREATE FUNCTION kinegy(mbs_model m)
                     -> scalar_object AS
       SELECT sum((SELECT kinegy(b)
                   FROM  body b
                   WHERE model_oid(b) = m));
OID[MBS_MODEL.KINEGY->SCALAR_OBJECT:2470]
MA8>
```

The function `kineqy` is an example of an overloaded derived function that is defined on the type `body` and returning the kinetic energy for the body. The same concept, `kineqy`, is also defined on the type `mbs_model` returning the total kinetic energy for the entire set of bodies in the MBS model. Thus, depending on the type of object, the appropriate operations are performed to derive the kinetic energy.

In MECHAMOS this overloading is also applied on the centre of mass example above where the derived function `posvec(mbs_model)` yields the position vector for the centre of mass of the MBS model and `posvec(body)` yields the position vector of the body's centre of mass.

4.2 Combining Numeric MBS data

Combining numeric MBS data is preferably done over several sets of numeric values. This opens up the possibility to study a mechanism over a parameter space. In the sliding pendulum example the different numerical sets vary only in the frequency of the moving wall. Plotting the maximum amplitude of the generalised coordinates for different ω values yields information about critical frequencies of the mechanism. This is accomplished with the following AMOSQL query

```
MA8> SELECT applot_o("omega", max(ss), n)
     FROM  mbs_numeric n,
              scalar_sequence ss
     WHERE name(model_oid(n)) =
           "Sliding Pendulum"
       AND ss = numseq("q1",n);
MA9>
```

This query searches the database for all sets of numeric values (`mbs_numeric`) which are defined on the "`Sliding Pendulum`" model. For each of these numerical sets the string "`q1`" is evaluated numerically to obtain a sequence of numeric values of the coordinate q_1. The maximum value in this sequence is then plotted versus ω. The result is the upper curve in Figure 4 below. The lower curve is obtained by simply append a plot of the minimum value (`min`) of the same numeric sequence. In Figure 5 this is done for the generalised coordinate q_2 ("q2").

As mentioned earlier it is the derived function `numseq` that performs the sequence of operations corresponding to *B*21-*B*24. The equations of motion (*B*21) have already been derived in a previous query (generation `MA2>`) and are thus not derived again since the result is available to the system. Solving the equations (*B*22) for each numerical set are performed for all the numerical sets in question except for the set with $\omega = 2.0$ [rad/s] since this has already been performed (again in generation `MA2>`). The symbolic expression to be evaluated (*B*23) is given in this case and thus not derived. Finally the numeric evaluation (*B*24) is performed for each of the numerical set.

To obtain the q_2 plot in Figure 5 only the numeric evaluation (*B*24) is performed since B21-B22 is already performed for the involved MBS model and the numerical parameter sets.

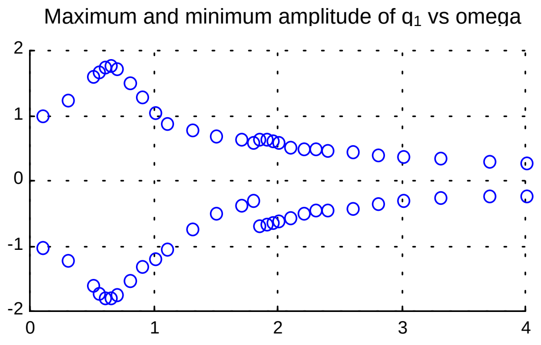

Figure 4. *The maximum and minimum amplitude of the slider position for all defined numerical sets in the MBS model.*

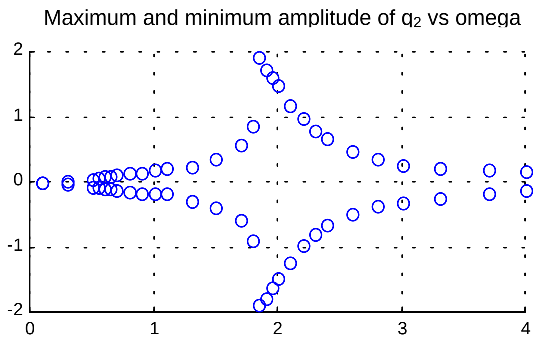

Figure 5. *The maximum and minimum amplitude of the pendulum angle for all defined numerical sets in the MBS model.*

4.3 Comparing Numeric MBS data

Selection conditions are crucial in query processing. In the examples above this comparison has been performed on abstract characteristics such as names of objects, OIDs etc. In this section comparison and selection will be based on simulation results and are therefore more computationally intensive.

Figures 4 and 5 show a graph over the maximum and minimum amplitude of q_1 and q_2 versus ω for the entire set of parameter values defined on the `"Sliding Pendulum"` model. Finding where in the ω-range the maximum amplitude of q_1 occurs is done with the following AMOSQL query where `equation()` composes a string of the parameter name and its value.

```
MA12> SELECT
      <equation(scalareqn_oid(n,"omega")), n>
      FROM  mbs_numeric n
      WHERE name(model_oid(n)) =
            "Sliding Pendulum"
        AND max(numseq("q1",n)) >= maxagg(
            (SELECT max(numseq("q1",n2))
             FROM  mbs_numeric n2
             WHERE name(model_oid(n2)) =
                   "Sliding Pendulum"));
<"omega=0.65",OID[0x0:1675] >
MA13>
```

The query compare MBS data from different simulations (which are reduced to numeric evaluations since the simulation results are already available to the system) and the result is a tuple of the ω-value and the OID associated to the `mbs_numeric` containing the numeric information. The corresponding query for q_2 yields the result

```
<"omega=1.85",OID[0x0:1965] >
```

and these two results correspond well to the maximum values found in Figures 4 and 5.

The amplitude of the pendulum angle q_2 is shown in Figure 6 (the plot is based on the `OID[0x0:1965]`) and it is seen that this angle exceeds $\pi/2$. The final question is if there are other numerical sets, i.e. ω-values, where the q_2 angle also exceeds $\pi/2$. This is answered with the following AMOSQL query and its result can also be verified in Figure 5

```
MA15> SELECT
      equation(scalareqn_oid(n,"omega"))
      FROM  mbs_numeric n,
                 scalar_sequence ss
      WHERE name(model_oid(n)) =
            "Sliding Pendulum"
        AND ss = numseq("q2",n)
        AND max(ss) > 3.14159/2;
"omega=1.95"
"omega=1.85"
"omega=1.9"
MA16>
```

A closing remark to this section is that the examples and queries presented here are general and can be used on other MBS models defined in the database without any hands on programming efforts. The MECHAMOS system is an MBS analysis tool based on database technology. This provides the system with general capabilities for combining and comparing large amounts of data.

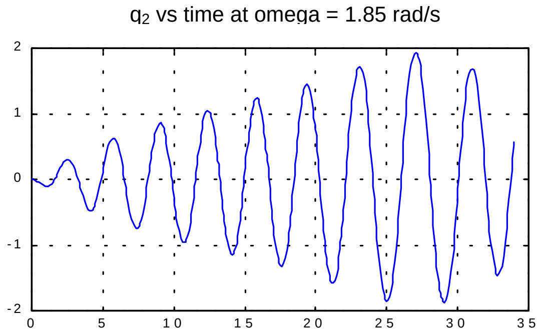

Figure 6. *A plot of the pendulum angle, q_2, for the numeric object yielding the maximum amplitude of the same q_2.*

5. Discussion and Conclusions

This work demonstrates the benefits of integrating MBS analysis in a database environment. The available query language enables MBS analysis on a higher level compared to traditional MBS analysis tools.

The differences in retrieval of symbolic and numeric MBS data are discussed and from this discussion the importance of function materialisation emerges. The materialisation capabilities avoid redundant and unnecessary computations and the simplified implementation in MECHAMOS is focusing on computational intensive operations and the most frequent operations in the system. The importance of this implementation is shown in some examples along with examples of combining and comparing of both symbolic and numeric MBS data.

Some of the advantages of the available database technology in MBS analysis found in this work are:

- *Several MBS models available to the analyst.* In traditional MBS software each model is analysed separately and usually each set of parameter values for the model are also analysed separately. The results of these separate analyses must then be stored and compared in a later stage of the analysis where the MBS models are no longer available to the analyst. This paper has presented and shown through examples how the MECHAMOS system handles MBS analysis and processes queries over several models. The examples have only ranged over several sets of parameter values within the same MBS model but it is evident that extending the query processing to range over several MBS models is supported in MECHAMOS.

- *Improved MBS data management.* The large number of defined parameter sets (a total of 29) involved in the sliding pendulum example shows how the available database technology facilitates the data management of the MBS analysis in MECHAMOS. Through the available query language, the MBS analysis is performed on a higher level compared to a traditional MBS analysis software. Further, the MBS data is not generated in the beginning of the session as in traditional tools but generated along the analysis as the data is required to proceed the analysis.

- *Extend the optimisation analysis to include several MBS models.* In a traditional optimisation process a given physical system is analysed in a parameter space where each set of parameter values represent a variant of the system. The aim of the optimisation is to determine the parameter values giving a maximum or minimum value of some response. With the possibility to easily analyse several MBS models simultaneously the optimisation can be performed over different physical concepts i.e. different technical solutions to obtain a similar functional response. The different concepts gives different equations of motion and each of the concepts may also contain variants in terms of different parameter sets. This has not been shown in the examples of this work but it is possible to perform such analyses in MECHAMOS.

- *Combining data between engineering disciplines.* This work has focused on MBS analysis but in a wider perspective where MBS analysis represents one discipline among others (e.g. CAD, FEA, CFD etc.) the ability to combine engineering data can even be of greater importance. Finite element analysis (FEA) may need to combine geometrical data from a CAD model and dynamic loads from an MBS model to perform a structural analysis. This issue is further discussed in [2].

Possible future development directions for MECHAMOS has also been discussed in [2]. This work supports the idea that the available database capabilities are an important factor for a successful implementation of the following suggestions:

- *Automatic submodel assembly*. Implementing a submodel representation and automatic assembly strategies will improve the MBS analysis in MECHAMOS considerable. Functional descriptions of physical components (e.g. motors, gearboxes, couplings etc.) can then be stored in the database as submodels. MECHAMOS should then be able to combine these submodels and derive MBS data for the assembled system. This implies that MECHAMOS automatically can derive the different concepts discussed in the optimisation section above. MECHAMOS should then be able to select the assembled system that is optimal in some sense.

- *Trajectory analysis.* Grossman [12-13] presents a strategy for storing trajectories in a database and search these trajectories to obtain information of the system described by the trajectory. In "path-planning problems", the path closest to a given desired path of a given point in the mechanism can be selected. Further given a trajectory that is a periodic orbit, other trajectories in the database can then be compared and those, which also have a periodic orbit, can be identified. The MECHAMOS system generates trajectories by deriving the equations of motion and solving these numerically. These trajectories can then be searched and the trajectories that fulfil certain criteria's could be detected.

Acknowledgement

This research has been sponsored by The Swedish Foundation for International Cooperation in Research and Higher Education and by The Swedish National Board for Industrial and Technical Development.

Kjell Orsborn is currently on leave from The Engineering Databases and Systems Laboratory at The Department of Computer and Information Science, Linköping University, Linköping, Sweden.

References

[1] Takizawa, M., *Distributed Database System JDDBS*, JARECT Computer Science & Technologies, Vol 7, OHMSHA & North Holland (publ.), 262-283, 1983.

[2] Tisell, C., Orsborn, K., *A System for Multibody Analysis Based on Object-Relational Database Technology*, Proceedings of the First International Conference on Engineering Computational Technology (EST '98), Edinburgh, Scotland, 18-20 August 1998. In Advances in Engineering Computational Technology, Topping, B.H.V. (ed.), Civil-Comp Press, ISBN 0-948749-55-5, 1998, p. 273-285.

[3] Orsborn, K., *On Extensible and Object-Relational Database Technology for Finite Element Analysis Applications*, PhD Thesis No 452, Department of Computer and Information Science, Linköping University, Linköping, Sweden, 1996, ISBN 91-7871-827-9, ISSN 0345-7524.

[4] Fahl, G., Risch, T., Sköld, M., *AMOS - An Architecture for Active Mediators*, The International Workshop on Next Generation Information Technologies and Systems (NGITS' 93), Haifa, Israel, June 28-30, 1993, p. 47-53.

[5] Flodin, S., Karlsson, J., Orsborn, K., Risch, T., Sköld, M., Werner, M., *AMOS User's Guide*, EDSLAB, Linköping University, Linköping, 1997, http://www.ida.liu.se/labs/edslab/amos/amosdoc.html.

[6] Flodin, S., Karlsson, J., Risch, T., Sköld, M., Werner, M., *AMOS System Manual*, EDSLAB, Linköping University, Linköping, 1997.

[7] Matlab (version 5), *The Math Works Inc.*, 24 Prime Park Way, Natick, MA 01760-1500.

[8] Maple (release 4), *Waterloo Maple Inc.*, 57 Erb Street W. Waterloo, Ontario Canada N2L 6C2.

[9] Lesser, M.B., *The Analysis of Complex Nonlinear Mechanical Systems, A Computer Algebra Assisted Approach*, World Scientific, Singapore, 1995.

[10] Lesser, M.B., *SOPHIA, A Proscriptive Language for Mechanics*, Proceedings of the 36th SIMS Simulation Conference, Stockholm 1994.

[11] Kemper, A., Kilger, C., Moerkotte, G., *Function Materialization in Object Bases: Design, Realization, and Evaluation*, IEEE Transactions on Knowledge and Data Engineering, v. 6 n. 4, August 1994, p. 587-608.

[12] Grossman, R., *Querying Databases of Trajectories of Differential Equations I: Data Structures for Trajectories*, Proceedings of the 23:d Annual Hawaii International Conference on System Sciences. Volume 2: Software Track., Jan 2-5 1990.

[13] Grossman, R., *Querying Databases of Trajectories of Differential Equations Ii: Index Functions*, Fourth NASA Workshop on Computational Control of Flexible Aerospace Systems, NASA Conference Proceedings, Number 10 065; Part 1, L. W. Taylor, Jr.; editor, NASA Langley Research Center, 1991, pp35-39.

Appendix 1.

```
/*----------------------------------------*/
/* This file populates the MECHAMOS system */
/* with the Sliding Pendulum model  990515 */
/* C - Claes Tisell, Machine Design, KTH   */
/*----------------------------------------*/

/* Create & populate the mbs_model object  */
SET :mod6 =constr_model('Sliding Pendulum',2,2);
constr_kdeeq(:mod6, "q1t", "u1");
constr_kdeeq(:mod6, "q2t", "u2");
constr_timedepeq(:mod6,"f1","A0*sin(omega*t)");
constr_irf(:mod6, "fN");
constr_arf(:mod6, "fB", {"fN","fB",3,"q2"});
constr_gravityvec(:mod6, VECTOR(0,"-g",0),"fN");
constr_parameter(:mod6,
{"mA","mB","JB","LB","k1","c1","A0","omega","g"});
set_pval(:mod6, "g", 9.81);

/*----------------------------------------*/

/**  Create and populate BODY "A"  **/
SET :b61  = constr_body(:mod6,"A","mA","fN");
constr_idyad(:b61,VECTOR(0,0,0,0,0,0),"fN");
constr_posvec(:b61,VECTOR("q1",0,0), "fN");
constr_aforvec(:b61,VECTOR("-k1*(q1-f1)",0,0),"fN");
constr_aforvec(:b61,VECTOR("-c1*(u1-f1t)",0,0),"fN");

/**  Create and populate BODY "B"  **/
SET :b62 = constr_body(:mod6,"B","mB","fB");
constr_idyad(:b62,VECTOR(0,0,"JB",0,0,0),"fB");
constr_posvec(:b62,VECTOR("q1",0,0), "fN");
constr_posvec(:b62,VECTOR(0,"-LB/2",0),"fB");

/*----------------------------------------*/

/* Create & populate "mbs_numeric" objects */

SET :s601 = constr_mbsnum(:mod6, "s01");
set_pval(:s601, "omega", 0.1);
set_time(:s601, 0, 2*3.1415927*10/0.1);

SET :s603 = constr_mbsnum(:mod6, "s03");
set_pval(:s603, "omega", 0.3);
set_time(:s603, 0, 2*3.1415927*10/0.3);

SET :s605 = constr_mbsnum(:mod6, "s05");
set_pval(:s605, "omega", 0.5);
set_time(:s605, 0, 2*3.1415927*10/0.5);

SET :s6055 = constr_mbsnum(:mod6, "s055");
set_pval(:s6055, "omega", 0.55);
set_time(:s6055, 0, 2*3.1415927*10/0.55);

SET :s606 = constr_mbsnum(:mod6, "s06");
set_pval(:s606, "omega", 0.6);
set_time(:s606, 0, 2*3.1415927*10/0.6);

SET :s6065 = constr_mbsnum(:mod6, "s065");
set_pval(:s6065, "omega", 0.65);
set_time(:s6065, 0, 2*3.1415927*10/0.65);

SET :s607 = constr_mbsnum(:mod6, "s07");
set_pval(:s607, "omega", 0.7);
set_time(:s607, 0, 2*3.1415927*10/0.7);

SET :s608 = constr_mbsnum(:mod6, "s08");
set_pval(:s608, "omega", 0.8);
set_time(:s608, 0, 2*3.1415927*10/0.8);

SET :s609 = constr_mbsnum(:mod6, "s09");
set_pval(:s609, "omega", 0.9);
set_time(:s609, 0, 2*3.1415927*10/0.9);

SET :s610 = constr_mbsnum(:mod6, "s10");
set_pval(:s610, "omega", 1.0);
set_time(:s610, 0, 2*3.1415927*10/1.0);

SET :s611 = constr_mbsnum(:mod6, "s11");
set_pval(:s611, "omega", 1.1);
set_time(:s611, 0, 2*3.1415927*10/1.1);

SET :s613 = constr_mbsnum(:mod6, "s13");
set_pval(:s613, "omega", 1.3);
set_time(:s613, 0, 2*3.1415927*10/1.3);

SET :s615 = constr_mbsnum(:mod6, "s15");
set_pval(:s615, "omega", 1.5);
set_time(:s615, 0, 2*3.1415927*10/1.5);

SET :s617 = constr_mbsnum(:mod6, "s17");
set_pval(:s617, "omega", 1.7);
set_time(:s617, 0, 2*3.1415927*10/1.7);

SET :s618 = constr_mbsnum(:mod6, "s18");
set_pval(:s618, "omega", 1.8);
set_time(:s618, 0, 2*3.1415927*10/1.8);

SET :s6185 = constr_mbsnum(:mod6, "s185");
set_pval(:s6185, "omega", 1.85);
set_time(:s6185, 0, 2*3.1415927*10/1.85);

SET :s619 = constr_mbsnum(:mod6, "s19");
set_pval(:s619, "omega", 1.9);
set_time(:s619, 0, 2*3.1415927*10/1.9);

SET :s6195 = constr_mbsnum(:mod6, "s195");
set_pval(:s6195, "omega", 1.95);
set_time(:s6195, 0, 2*3.1415927*10/1.95);

SET :s620 = constr_mbsnum(:mod6, "s20");
set_pval(:s620, "omega", 2.0);
set_time(:s620, 0, 2*3.1415927*10/2.0);

SET :s621 = constr_mbsnum(:mod6, "s21");
set_pval(:s621, "omega", 2.1);
set_time(:s621, 0, 2*3.1415927*10/2.1);

SET :s622 = constr_mbsnum(:mod6, "s22");
set_pval(:s622, "omega", 2.2);
set_time(:s622, 0, 2*3.1415927*10/2.2);

SET :s623 = constr_mbsnum(:mod6, "s23");
set_pval(:s623, "omega", 2.3);
set_time(:s623, 0, 2*3.1415927*10/2.3);

SET :s624 = constr_mbsnum(:mod6, "s24");
set_pval(:s624, "omega", 2.4);
set_time(:s624, 0, 2*3.1415927*10/2.4);

SET :s626 = constr_mbsnum(:mod6, "s26");
set_pval(:s626, "omega", 2.6);
set_time(:s626, 0, 2*3.1415927*10/2.6);

SET :s628 = constr_mbsnum(:mod6, "s28");
set_pval(:s628, "omega", 2.8);
set_time(:s628, 0, 2*3.1415927*10/2.8);

/* and so on for the rest of the numeric objects */
/* That is for omega = 3.0, 3.3, 3.7 and 4.0     */
```