Appendix A

Sophia Command Assistance

A.1 Introduction

This appendix contains examples of a number of basic Sophia commands. The command arguments are typical cases, e.g. if the name of a typical generalized coordinate is called for the example might use q3 for q_3. The Sophia tools are designed to work in a normal manner with Maple, thus they typically define an object which must be assigned to a symbol for storage or further processing. Some of the Sophia tools control the environment by setting global variables. Thus frame information is stored in direction cosine matrices with the naming convention, Rframe1frame2. Also certain sets are defined that record the frame names used or substitution lists used in differentiation operations. The tools can also be used as functions in normal Maple programming.

A.2 Installation of Sophia

The software included with this book is on a disk in PC readable format. The file of Sophia programs is in text format and should be easy to transform to other formats. The software, together with other information and programs, is also available on a server available to internet users. The address is ftp.mech.kth.se. Login as anonymous and use your computer mail address as your password. Look in the directory ‘sophia’.

Copy the text file, SoPhIa.txt, into a directory which can be accessed by your Maple software. For example, on a Macintosh, put the file into the Maple folder. Start Maple as usual. Use the Maple ‘read’ command with the file path and name as argument. On a Macintosh the keyboard combination command-2 will bring up a selection box from which the appropriate file can be selected. Acceptance of the file will cause the path and name to be printed on the console input line after the read command. The Maple input line would be something like, read ‘mypath:SoPhIa.txt‘; Note the backquotes and the semicolon! Pressing enter will read in the file. Ignore the warnings which are due to reading in Maple’s linear algebra package, which is used by Sophia. You can now proceed with your work session. Note that it takes between 30 seconds and several minutes (depending on your system) to read in the routines. To avoid this time delay read in SoPhIa.txt as above, then type save sophia.m This will create a Maple 'm' file which can be read into your workspace in a few seconds. The next time you use Sophia just start your Maple session and read the 'm' file.

The included version of Sophia is for MapleV release 3. Older versions of Maple do not use the 'global' declaration in procedures. If you have an older version of Maple you will have to use a text file to remove all lines in Sophia's procedures that contain a 'global' declaration.

There may be newer information and other material on the distribution disk or in the ftp directory. The SoPhIa.txt file contains a number of procedures that are not documented in this text. Some are help procedures, needed to achieve the objectives of the documented procedures. Others are extensions that are useful for dealing with particular problem situations. Most of the procedures contain some comments on their function.

While it is possible to use Maple to carry out numerical integration of equations of motion derived with Sophia, it is frequently better to export the results to other programs. The ftp sight should be consulted for Sophia extensions that help in this task, for example in export of results to the popular Matlab program or to a 'c' program.

A.3 Setting Frame Information

A common situation that arises is that the frame relation is known between some frames A and B, and between B and C, but not between A and C. If such a transformation is called for, Sophia will automatically detect the intermediate frame B and compute the relations between A and C.

- &rot [A,B,3,q_1]
- chainSimpRot([[A,B,3,q_1],[B,C,1,q_2]])
- dircos(B,A,$\boldsymbol{b}_1 \cdot \boldsymbol{a}_1, \boldsymbol{b}_1 \cdot \boldsymbol{a}_2, \boldsymbol{b}_1 \cdot \boldsymbol{a}_3, \boldsymbol{b}_2 \cdot \boldsymbol{a}_1, \boldsymbol{b}_2 \cdot \boldsymbol{a}_2, \boldsymbol{b}_2 \cdot \boldsymbol{a}_3, \boldsymbol{b}_3 \cdot \boldsymbol{a}_1, \boldsymbol{b}_3 \cdot \boldsymbol{a}_2, \boldsymbol{b}_3 \cdot \boldsymbol{a}_3$)

A.4 Declaring Functional Dependence

- dependsTime($q_1, q_2, q_3, u_1, u_2, u_3$)
- &kde n → *Standard set up in which 2n variables q and u are set to depend on time and kde is assigned so that the generalized speeds are the generalized velocities, i.e. the time derivatives of the generalized coordinates.*
- clearVars() *clears all time dependencies that have been declared*

A.5 Evectors

A.5.1 Construction and Selection

It is important to make sure that the chosen frame name does not conflict with other names in the work space, either defined as part of the particular task or by Maple. For example the letter 'D' is a Maple operator. Frame names need not be a single letter, for example names such as FrameNewton are acceptable if somewhat lengthy. The basic procedure for defining a dyad is also included.

- A &ev $[v_1, v_2, v_3]$ → v = $v_1\mathrm{a}_1 + v_2\mathrm{a}_2 + v_3\mathrm{a}_3$ *This sets up an Evector from the components in frame A.*
- A &> j → $\boldsymbol{a}_j$ *This operator creates a base vector for any frame.*
- &vPart v → $[v_1, v_2, v_3]$ *extracts the array of components. Note that this is an array, not a list!*
- &fPart eV → frame name *extracts the frame name of the representation of the vector*
- v &c j → v_j *extracts the jth component in the frame in which the vector is represented.*
- Edyad($d_{11}, d_{12}, d_{13}, d_{21}, d_{22}, d_{23}, d_{31}, d_{32}, d_{33}$,N) → $\begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix}$ *sets up the representation of a dyad in the given frame*

A.5.2 Changing Representations

- N &to ev *Transforms the representation from whatever frame is current for the object to the stated frame. This works for both Evectors and Edyads.*
- &simp ev *Carries out simplification operations on the components of the Evector or Edyad. This is not done automatically as it can be a costly operation.*

A.5.3 Algebra

The following operations provide the normal algebra of vectors and dyads in three dimensional Euclidian space.

- s &** v → $s\boldsymbol{v}$ *Simply the multiplication of a scalar (number) into a vector, i.e. all components of the vector are multiplied by the scalar.*

- v1 &++ v2 → $\boldsymbol{v}_1 + \boldsymbol{v}_2$ *The addition of two vectors. Sophia takes care of transforming the vectors representations to consistent frames before addition. The result is represented in the frame that the last vector was represented in. Multiple additions may be put in the same statement, however it is a good idea to use parenthesis to group other operations.*
- v1 &xx v2 → $\boldsymbol{v}_1 \times \boldsymbol{v}_2$ *This is the normal vector cross product.*
- v1 &o v2 → $\boldsymbol{v}_1 \cdot \boldsymbol{v}_2$ *This is the normal 'dot' product. The object on the left in a product may be an Evector or Edyad.*
- &VtoD v → $Dyad(\boldsymbol{v})$ *Produces the so called Dyad of a Vector. That is the dot product of this object with another vector is equivalent to the cross product of the original vector with the other vector.*
- &DtoV D → $Vect(\frac{1}{2}(D - D^T))$ *This takes the antisymmetric part of an Edyad and converts it to a vector.*

A.5.4 Differentiation

The symbol 't' is reserved for time. Sophia represents time derivatives by appending the appropriate number of t's to the quantity. Sophia requires the declaration of variables which depend on time in order to set up certain substitution lists that are used in carrying out differentiations. The system is setup for first and second order derivatives (all that need to appear in formulating equations of motion).

- &dt s → $\frac{ds}{dt}$ *This is for differentiation of a scalar expression. Quantities such as q1 have derivative q1t.*

  A &fdt v → $\frac{^{A}dv}{dt}$ *This is the important 'frame differentiation operator'.*
- $^{N}\boldsymbol{\omega}^{A}$ &fdtAV v → $\frac{^{N}dv}{dt}$ *For frame differentiation in terms of a given expression for the angular velocity. This can be useful in the process of looking for a suitable set of kinematic differential equations.*

A.5.5 Kinematic Quantities

The following are to provide assistance in obtaining basic kinematic information from frame information, i.e. it assumes the frames have been specified in terms of suitably declared variables.

- A &aV B → $^{A}\boldsymbol{\omega}^{B}$ *The angular velocity is computed in terms of the coordinate information.*
- aA(A,B) → $^{A}\boldsymbol{\alpha}^{B}$ *The angular acceleration is computed in terms of the coordinate information.*

- aA(A,B,wAB) → $\frac{^{B}d}{dt}{}^{A}\boldsymbol{\omega}^{B}$ *An alternative computation of the angular acceleration in terms of an expression for the angular velocity, which is invoked as an optional third argument to aA.*
- FFV(N,rB,wNB,rQ) → $^{N}\boldsymbol{v}^{B(Q)}$ *Useful when examining rolling constraints, this function gives the frame fixed velocity. The inputs are the base frame, the position of the index point of frame B, the angular velocity of frame B in N and the position of the special point, typically the point of contact between body B and a point in frame N.*
- `[N,B] &ffv [rB,rQ]`

  → $^{N}\boldsymbol{v}^{B(Q)}$ *An alternate version of the above which does not require the angular velocity. The left argument list is the base and body frame. The right list are the positions of the index and object points.*

A.5.6 Rigid Body Properties

Tools for determining body properties needed in formulating equations of motion, such as the center of mass and the moment of inertia dyad.

- `B &cm [[m1,r1],[m2,r2],[m3,r3]]`

  → center of mass of system of mass points represented in frame B. *Note there must be at least two frames known to the system or an error will occur. It should be realized that this operator is useful for a composite set of bodies. The individual center of mass can be treated as an equivalent point object in the computation of the systems mass center.*
- `m &mpI  r`

  → moment of inertia of mass point located at position $\boldsymbol{r}$. *The result is in the frame in which r is given.*
- `B &msI [[m1,r1],[m2,r2],[m3,r3]]`

  → moment of inertia of a system of mass points. *The comments above regarding the mass center also applies to this.*
- `[m,Icm] &-> r`

  → transfers moment of inertia, Icm is the moment of inertia about center of mass and r is the vector from the cm point to the new point. *Sometimes called the 'parallel axes or Stiener's theorem.*

A.5.7 Equipollent Systems and Screw Transformations

The following are helpful in finding the equivalent force and torque due to a system of forces and torques acting on a rigid body. In setting up the equations of motion it is frequently required to obtain the torque about the mass center or some other index point in each of the involved bodies. The screw object in Sophia is a list of two Evectors. The first object in the list may typically be a torque or velocity, the second a force or angular velocity. For Equipollent Force Systems we generally consider torques and forces.

- A &screw [[r1,R1],[r2,R2],[r3,R3]] → [T,R] *The left argument is the frame in which the result will be represented. The right argument is a list of lists. The latter lists have two members, the point of application of a force and the force. The point of application is an Evector giving the displacement from some index point. The force is also an Evector. The index point is the same for all the forces and the resultant screw object is in terms of that point. The typical case is that the index point is the mass center of a body and the forces are all the applied forces acting on the body at various points.*

- [T,R] &ss rs → [T',R] *The displacement vector rs runs from the point of the screw on the left of the operator to another point. The result is a screw which is based on the new point. Note that this also applies to the pair [v,ω], i.e. provides an expression for the velocities at two points in a rigid body relative to another frame.*

A.6 Redundant Coordinates

This proceedure is for both redundant coordinates and nonholonomic constraints. The first argument is a list of the chosen independent generalized speeds. The second argument is the set of kinematic differential equations for the problem with the redundant coordinates. The third argument is the set of velocity constraint relations. The fourth argument is the name used for the original generalized speeds, the fifth argument is the name used for the transformed generalized speeds. It is implicit that a subset of the new generalized speeds is identically zero. The latter expresses the transformed constraint relation.

- ReducedSpeeds(gslist,kde,vce,u,w) → rtkde,rigst *The output of the procedure is a sequence of two sets. The first set is the reduced transformed kinematic differential equations. The second is the reduced inverse generalized speed transformation. Both of these sets are useful in placing the problem in terms of the reduced generalized speeds and their time derivatives as discussed in chapter 6.*

A.7 KMvectors

The KMvectors are the representations of Kvectors in Maple. They form a vector space and they provide the key to finding constraint free equations of motion. The following are the basic operations for dealing with these objects.

A.7.1 Construction and Selection

- &KM [v1,v2,v3] → $\boldsymbol{v}^{<}$ *forms the Kvector from the Evector components. The user need not worry about what representation is used for the latter.*

- &sPartKM vK → K, *number of Evectors in vK It is important to make sure that in any specific problem one has the right size Kvector!*

- &vPartKM vK → $[\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3]$ *It is frequently useful to have a list of the component vectors to work with as a Maple object.*

A.7.2 Algebra of KMvectors

- s &*** vK → $s\boldsymbol{v}^{<}$ *scaler multiplication*

- vK1 &+++ vK2 → $\boldsymbol{v}_1^{<} + \boldsymbol{v}_2^{<}$ *The addition of Kvectors.*

- vK1 & - - - vK2 → $\boldsymbol{v}_1^{<} - \boldsymbol{v}_2^{<}$ *Sometimes it is useful to represent the subtraction operation directly.*

- vK &O tau1 → $\boldsymbol{v}^{<} \bullet \boldsymbol{\tau}_1^{<}$ *inner product of KMvectors*

- &Ksimp vK → *simplification of a KMvector*

A.7.3 Tangent Space Operations

- *Use is made of the basic velocity expansion equation:*

$$\boldsymbol{v}^{<} = \sum_{j=1}^{j=n} u_j \boldsymbol{\tau}_j^{<} + \boldsymbol{\tau}_t^{<}$$

- *The convention is assumed that the kinematic differential equations are stored in a set called kde.*

- *It is also convenient to start all generalized coordinates and speeds with the same variable symbol, e.g. q for generalized coordinates and u for generalized speeds are typical. Thus q1, q2 and q3; u1, u2 and u3.*

- KMtangents(vK,u,3) → $[\boldsymbol{\tau}_1^{<}, \boldsymbol{\tau}_2^{<}, \boldsymbol{\tau}_3^{<}]$ *A list of the three independent tangent vectors based on the generalized speeds u1,u2 and u3.*

- vKtime(vK,u,3) → $\boldsymbol{\tau}_t^<$ *Note that this is printed out when using the KMtangents command. It can be used as a rough means of checking for errors*

- tau &kane fK → $[F_1, F_2, F_3]$ *A list of the Generalized Active Forces as defined by Kane*

A.7.4 KMvector Calculus

- N &Kfdt vK → $\frac{{}^{N}d}{dt}\boldsymbol{v}^<$ *This carries out the basic frame differentiation operation on all components at once.*

- N &rKfdt rK → *This special form of the time differentiation operator is intended for use on a KM position vector made up of positions of frame origins and possibly frame names. These become angular velocities* ${}^{N}\boldsymbol{\omega}^{Fr}$ *for frame Fr.*

A.8 SKvectors

These are lists of KM vectors or 'super' KMvectors which are useful in the theory of transformations of tangent vector base sets developed in chapter 3.

A.8.1 SKvectors Constructors and Selectors

- &SK [tau1,tau2,tau3] → τ

- tau &SKc 2 → $\boldsymbol{\tau}_2^<$

- &SKn tau → 3

A.8.2 SKvector Algebra

- s &**** tau → $s\tau$

- beta &++++ tau → $\beta + \tau$

- beta & - - - - tau → $\beta - \tau$

A.8.3 Linear Operators on SKvectors

- BetaTau &++** tau → $[\boldsymbol{\beta}_1^<, \boldsymbol{\beta}_2^<, \boldsymbol{\beta}_3^<]$ *BetaTau is an ordinary matrix and this gives the linear transformation from the tangent vectors* τ *to* β

A.8.4 Projection

- GK := &Kmetric tau → $G_{ij} = \boldsymbol{\tau}_i^< \bullet \boldsymbol{\tau}_j^<$ *The metric coefficients as a matrix for the subspace generated by* τ

- &cv tau → τ_c *The linear transformation using the matrix inverse(G) generates the recipricol bases for the subspace.*

- tauC &<> wK → $[c_1, c_2, c_3]$ where

$$\dot{\boldsymbol{\Pi}}_* \bullet \boldsymbol{w}^< = c_1\boldsymbol{\tau}_1^< + c_2\boldsymbol{\tau}_2^< + c_3\boldsymbol{\tau}_3^<$$

- $[c_1, c_2, c_3]$ &SKsum tau → $\boldsymbol{\Pi}_* \bullet \boldsymbol{w}^<$

- *Note that this procedure can be used to sum up any list of KM vectors e.g. the velocity expansion gives the velocity KMvector as:*

  ([u1,u2,u3] &SKsum tau) &+++ VKtime(vK,u,3)

- *Note that the generalized speeds are simply the projections of the of the velocity Kvector on the reciprocal basis appropriate to the corresponding tangent Kvectors*

A.8.5 The Direct Kinematic Problem

For holonomic systems a set of kinematic differential equations implies certain relationships between the coefficient matrix and the inhomogeneous column matrix.

- *The General Form of The Kinematic Differential Equation*

$$\dot{q}_i = \sum_{j=1}^{n} Y_{ij}u_j + X_i$$

- *The velocity expansion equation after a transformation from a coordinate basis* τ *to an arbitrary basis* β.

$$\boldsymbol{v}^< = u^T\beta + \boldsymbol{\beta}_t^<$$

  Note that the 'generalized speeds' for the coordinate basis are simply the generalized velocities based on the chosen generalized coordinates $\dot{q}$.

- *The transformation relations are:*

$$W^{\beta\tau} = Y^T$$

$$\beta = W^{\beta\tau}\tau$$

$$\boldsymbol{\beta}_t^< = \boldsymbol{\tau}_t^< + X^T\beta.$$

- *The Maple-Sophia2 implementation of the above is:*

```
> BetaTau := transpose(Y):
> beta := BetaTau \&++** tau:
> betatK := tautK \&+++ ( X &SKsum tau):
```

- CoefficientArray(kdeList,gsList) → coefficient matrix *The arguments are the kinematic differential equations put into ordered form as a list and a list of the generalized speeds. The result is the coefficient matrix that appears in the kinematic differential equations. See section 3.10*

- SourceList(kdeList,gsList) → inhomogeneous terms in the kinematic differential equations *See above and section 3.10*

A.9 Series Expansions

These procedures are tools for producing series expansions in terms of a small parameter. The idea is to do this directly as part of the derivation of the equations of motion. In most cases expression complexity growth will make it impossible to derive approximate equations from the exact equations. This is because of the impossiblility of deriving exact equations using a given set of computer resources. The procedure calls for a user defined set, called *disturbance* below. This set is a group of substitutions that implement a particular variable scaling in terms of some small parameter. The expansions are in terms of the parameter. This must be reflected by the expansion of direction cosine matrices, which is accounted for in the procedure SetApproximateRotation.

- SetApproximateRotationdisturbance,eps,n → *creates approximate direction cosine matrices to order n in the parameter eps. The set, disturbance, must contain a set of scaling relations in terms of the parameter, e.g.* $\{q1 = eps * q1, u1= eps * u1\}$.

- EvectorSeries(Evec,eps,n) → expansion of the Evector, Evec to order n in eps.

- KvectorSeries(Kvec,eps,n) → expansion of the Kvector, Kvec to order n in eps.

- SvectorSeries(Svec,eps,n) → expansion of the Svector, Svec to order n in eps.

- B &Ato Evec → Evec in frame B to order n.

- A &Afdt Evec → frame based derivative of Evec to order n.

- A &AKfdt Kvec → frame based derivative of Kvec to order n.

- tauK &kane pKt → a list of scalar expressions obtained by taking the fat dot product of each of the members of the set tauK with the Kvector pKt.

A.10 Global Variables

Sophia uses some special global variables to store important information. The user should know about these both to avoid name conflicts and to make other use of the information.

- RfAfB *When a frame relation is defined direction cosine matrices are formed. They are always named RfAfB, where fA and fB are the frame names. If in the course of a computation the system needs a new matrix and has the information to compute it, new names will be assigned as needed. All defined frames are placed in a Global set.*

- SetOfFramesUsed → {{fA,fB},{fA,fC},{fB,fC}} *All the frames known to the system. Each set implies the direction cosine matrices in both directions.*

- kde *In some cases it is assigned by the system, but in general should always be used for the set of kinematic differential equations.*

- toTimeFunction *is used by the system to transform from the time expression form where a derivative is indicated by an appended t, to the functional form where it is indicated by the Maple differentiation expression and functions have explicit time dependencies, for example q1 becomes q1(t), q1t becomes diff(q1(t),t). The user can use subs(toTimeFunction,expr) to convert an expression to the functional form for further processing in Maple, for example in Maples differential equation package.*

- toTimeExpression *The opposite of the above conversion.*

A.11 Comments

Some useful or special considerations when using Sophia in Maple.

- The Maple command *subs(set,expression)* works on fairly general objects, for example KMvectors.

- Always remember the Maple 'last name evaluation rule' for arrays. To get at the actual array one needs to use op(arrayName). arrayName will not evaluate to the array object!

- The toTimeFunction set allows easy conversion of Sophia's time expressions to Maple's functional forms. This is very useful if one wants to use Maple's numerical differential equation solver.

- Consult the Maple help files and manuals to obtain information on how to extract coefficient matrices from sets and lists of linear equations. This is helpful when moving information out of Maple to other applications.

- There is a Sophia package for interacting with Matlab. Parts of this may be obtained by FTP transfer to the server given in the preface to this book.
- Sophia contains many more functions then discussed here or even in the text. Many are used to set up the main functions. The user may find it interesting to explore the procedures for ideas in extending the system.

Appendix B

Annotated List of References and Computer Software

I consider any of these books worth having, even if it is unlikely you will want to read it from cover to cover. Wherever possible I include ISBN numbers to help you in ordering the book, but try to specify a soft cover or international student version to keep cost down. If you are going to use mechanics professionally in your work you will want to own most of these books, but you can start by browsing through a library copy. The comments are personal and should be looked at in that light. Whatever is said it should be realized that mention of a book indicates that I consider it worth looking at and even owning. The computer software represents, at the time of writing, typical modern approaches to numerical mechanics and the use of computer algebra in mechanics.

B.1 Books

- `Angeles, J.`
  `Rational Kinematics`
  `Springer-Verlag 1988`
  `ISBN 0-387-96813-X`

  Modern kinematics makes extensive use of mathematics. This book shows that rigid body kinematics can be put on a firm footing by the use of linear algebra. Theorems and abstraction take priority over computational details.

- `Arnold, V.I.`
  `Mathematical Methods of Classical Mechanics, Second Edition`
  `Springer Verlag 1989`
  `ISBN 0-387-96890-3`

  The acknowledged definitive reference giving the modern mathematicians view of classical analytical mechanics. Concentration is on conservative Hamiltonian