A SYMBOLIC ENGINEERING MECHANICS SYSTEM BUILT ON OEM MAPLE

Philip Todd, Robin McLeod and Marcia Harris
Saltire Software, Beaverton, Oregon

Abstract

In this paper, we present a system which converts graphical descriptions of engineering mechanics problems into closed form equations. We describe the functions we provide for simplifying and manipulating the resulting mathematical equations.

1. Introduction

A major limitation of the use of symbolic mathematics packages, such as Maple, by mechanical engineers is the difficulty of accurately converting a geometry based mechanical engineering problem into a set of algebraic equations which can then be manipulated, solved or visualized by the symbolic mathematics package.

In the last eight years, Variational Geometry has emerged as a natural and powerful language for the expression of mechanical engineering problems [1,2,6], particularly problems in the realm of engineering mechanics. Problems involving the kinematic behavior of mechanisms may be succinctly expressed in the form of a constraint-based variational model of the geometry along with first and second derivatives of the constraints. With an additional capability for handling forces and masses, we add the ability to express problems in the statics and dynamics of mechanisms. The commercial package *Analytix* uses a variational geometry user interface as the means of specifying problems in the dynamic analysis of mechanisms [3-6]. These problems are then solved numerically in the package. Figure 1 shows an Analytix model of a four-bar linkage.

A Constructive Variational Geometry System represents engineering geometry as a sketch with constraints (also called "dimensions") given as labels on the sketch. "Constructive" refers to the classical constructions of Euclid's geometry, for example, the construction of a point of intersection, given two known lines. "Variational" means that numerical information resides in the dimensions, such as distance between points or measure of an angle, which are labeled on the diagram, rather than residing in the coordinates of points; therefore change or motion in the figure can be simply represented as change or variation in a constraint value.

In this paper, we present a system which converts an engineering mechanics problem, expressed as an Analytix model, into a symbolic mathematical representation within Maple. We discuss the functions which were developed to allow the user to manipulate and display this mathematical model. We present several examples to show the process of using the hybrid system.

Our approach allows the engineering mechanics problem to be expressed in the natural graphical terms of a dimensioned sketch, while automatically performing the error prone process of converting the geometrically expressed problem into mathematics.

2. System Architecture

There are three components to the symbolic mechanics package: (i) the front end is a modified version of the Analytix variational geometry package; (ii) a layer of code converts from the Analytix representation of the mechanics problem to a symbolic representation; (iii) a set of Maple functions facilitates user interaction with the symbolic representation.

Copyrighted material

Component (i) is written in C, component (ii) contains C code and Maple code, component (iii) is entirely Maple code.

(i) Analytix

To create a model of a mechanism in the Analytix package, the user first sketches the mechanism. Typically the sketch contains a simplified representation of the mechanism's geometry. Links are usually represented by straight lines between joints or points of force application. Centers of gravity of any links with non-negligible mass are explicitly sketched.

The user then specifies the exact geometry by adding constraints to the drawing. The constraints specify angles or distances on the drawing. For a kinematic analysis, the constraints should represent quantities whose motion is known. In practice, for a single degree of freedom mechanism, all the constraints but one represent quantities which stay fixed throughout the motion of the mechanism. The final constraint represents the driver, whose motion is known. The Analytix system aids the constraint specification process by giving continuous feedback on whether the model is underconstrained, overconstrained, or consistently constrained. The constraint values may be numbers or expressions. Figure 1 shows an Analytix model for a four bar linkage.

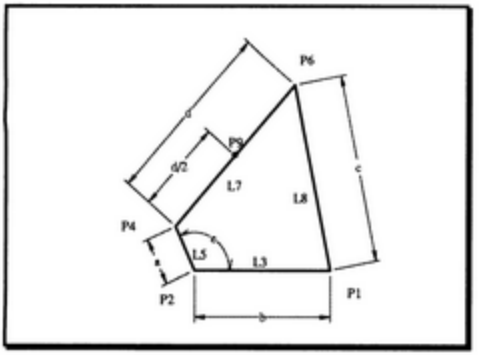

Figure 1: A four bar linkage specified by the lengths of three sides and one angle.

To perform a kinematic analysis, the user specifies the velocities and accelerations of any constraints which are in motion. (In the example of the four bar linkage (figure 1), all constraints would have 0 velocity and acceleration except the driving angle. If the crank is being driven at constant angular velocity v the angle would be given a velocity v and an acceleration 0.) The user also must specify a fixed point and a fixed line.

With the constraint velocities and accelerations given, the model is able to output on demand the position, velocity, and acceleration of any point of the mechanism and the angular velocity and angular acceleration of any line.

Static analysis of a mechanism may be performed in a very natural way in a Variational Geometry system by assuming that constraints carry reactions. Hence not only the motion, but also the statics of the model are defined by the constraints. If loads are applied to a mechanism in Analytix, reaction forces are generated in the constraints. The system will sum the constraint forces which impinge on a point to give the reaction force at that point. The system will integrate the forces on a single line segment to give shear force and bending moment.

Analytix allows external forces to be applied to any point on a mechanism and external torques to be applied to any line. These forces may be constant or they may vary with geometry, time or some other parameter. Analytix also provides conformal force elements: translational and rotational spring-damper-actuators.

To perform inverse dynamics in Analytix, the user specifies the instantaneous motion of the mechanism in the same way as for kinematics, but also applies forces and specifies the mass of points representing mass centers. The user also specifies the moment of inertia of lines which represent rigid links. The user can now generate the same reaction force outputs as for the static analysis, except the inertial forces generated by the accelerating masses will be taken into consideration.

In a dynamic analysis, some constraints represent quantities whose behavior is known in advance - either quantities which remain fixed or drivers, whose motion is given. Other constraints are free to accelerate in response to unbalanced forces present in the model. To set up a dynamic model, the user

Copyrighted material

specifies the fixed constraints and the input drivers in the same way as for kinematic analysis. Free constraints are given initial values and initial velocities and marked as being free to accelerate.

(ii) Creating Symbolic Analytix Models

In a previous paper [4] a method was described for performing a symbolic analysis of dimensioned engineering drawings. Analysis of the constraint graph associated with the drawing led to its description as a sequence of elementary Euclidean constructions, each of which was responsible for constructing a single point or line from its specified relation to known (already constructed) points and lines.

For example, a construction sequence for the four bar linkage of figure 1 is given below.

Construct a point (point P2) at the origin.
Construct a line (line L3) through point P2 in the direction of the x-axis.
Construct a point (point P4) on line L5 distance a from point P2.
Construct a line (line L5) through point P2 angle e degrees to line L3.
Construct a point (point P1) on line L3 distance b from point P2.
Construct a point (point P6) distance c from point P1 and distance d from point P4.
Construct a line (line L8) through points P1 and P6.
Construct a line (line L7) through points P4 and P6.
Construct a point (point P9) on line L7 distance d/2 from point P4.

Each individual construction which is part of the sequence described above has a corresponding set of equations which may be solved to determine the unknown geometrical entity. These are described in [4].

In [4], expressions for the coordinates of the drawing's points and the coefficients of the drawing's lines were established in terms of the independent constraint values alone. With this approach, the system suffered from substantial intermediate expression expansion, and was able to deal only with rather simple drawings. In order to deal with drawings and engineering mechanics problems of arbitrary complexity, it was necessary to maintain a symbolic representation as a nested sequence of expressions with intermediate variables introduced by the system. This eliminated the exponential growth in individual expression complexity. However, as Maple is designed to manipulate single expressions rather than sequences of expressions with collective meaning, this necessitated a layer of code to allow the manipulation of our symbolic representations.

Table 1 shows the system output for the location of point P9. For comparison, table 2 shows the same expressions without intermediate variables.

$$P6d^2 = (a\cos(e) - b)^2 + a^2\sin(e)^2$$

$$P6e^2 = c^2 - \frac{1}{4}\frac{\left(P6d^2 + c^2 - d^2\right)^2}{P6d^2}$$

$$P6c = \frac{1}{2}\frac{P6d^2 + c^2 - d^2}{P6d}$$

$$P6y = \frac{P6c\ a\sin(e)}{P6d} - \frac{P6e\ (a\cos(e) - b)}{P6d}$$

$$P9y = \frac{1}{2}a\sin(e) + \frac{1}{2}P6y$$

$$P6x = b + \frac{P6c\ (a\cos(e) - b)}{P6d} + \frac{P6e\ a\sin(e)}{P6d}$$

$$P9x = \frac{1}{2}a\cos(e) + \frac{1}{2}P6x$$

Table 1: Equations for the curve traced by point P9 on the four bar linkage shown in Figure 1.

Copyrighted material

FullySubstitute(P9x);

$$\frac{1}{2}a\cos(e)+\frac{1}{2}b+\frac{1}{4}\frac{\left((a\cos(e)-b)^2+a^2\sin(e)^2+c^2-d^2\right)(a\cos(e)-b)}{a^2\cos(e)^2-2a\cos(e)b+b^2+a^2\sin(e)^2}+\frac{1}{4}\Big(2c^2a^2\cos(e)^2-4c^2a\cos(e)b+2c^2b^2+2c^2a^2\sin(e)^2-a^4\cos(e)^4+4a^3\cos(e)^3b-6a^2\cos(e)^2b^2-2a^4\cos(e)^2\sin(e)^2+4a\cos(e)b^3+4a^3\cos(e)b\sin(e)^2-b^4-2b^2a^2\sin(e)^2-a^4\sin(e)^4+2d^2a^2\cos(e)^2-4d^2a\cos(e)b+2d^2b^2+2d^2a^2\sin(e)^2-c^4+2c^2d^2-d^4\Big)^{1/2}a\sin(e)\Big/\Big(a^2\cos(e)^2-2a\cos(e)b+b^2+a^2\sin(e)^2\Big)$$

Table 2: Equation for P9x with all the intermediate variables eliminated.

Two different engineering mechanics functions are available to the user. Reaction(x) gives the pseudoforce in parameter x. InstallDiffEq([x_0,...,x_n]) creates new variables A_x_0,...,A_x_n representing the accelerations of x_0,...,x_n. In the process of assigning expressions to these accelerations, a number of new intermediate variables will be introduced (many representing derivatives of existing variables). Velocity variables V_x_0,...,V_x_n will also be introduced and treated as independent input variables. The methods used in these functions are described in [6].

(iii) Interacting With Symbolic Analytix Models

The Intermediate Variable Table (IVT) and the routines to support and manipulate this structure were written in the Maple symbolic programming language and coexist as a package running within Maple. We describe in the following the functions which constitute this library, grouped according to their use.

(a) Adding variables to the IVT

Install(varname,eqn) - Installs a new variable with defining (implicit or explicit) equation eqn.

InstallAreaMoments(name,pointlist) - installs a number of new variables corresponding to area moments of the polygon defined by the points in pointlist.

InstallDerivative(var,dvar) - installs an expression for the derivative of var with respect to dvar.

(b) Extracting Expressions from the IVT

Evaluate(expr) - returns expr with the equation from the IVT substituted for each variable in expr which is in the IVT. This is not done recursively

FullyEvaluate(expr) - A recursive version of the above.

EvaluateF(expr,assgn) - performs a floating point evaluation with numerical assignments given in assgn.

Equations(expr,style) - for each variable v in expr which is in the Intermediate Variable Table, returns the equation defining v. Style specifies whether implicit or explicit equations are desired.

FullEquations(expr,style) - A recursive version of the above

CFunction(fname,varlist) - creates a C function with name fname to evaluate the variables contained in the varlist. Inputs to the function are all the constants in the recursive definition of the variables in varlist.

(c) Displaying the IVT

Show(expr,style) - displays expr and the definitions (recursively) of all the table symbols involved in expr.

Copyrighted material

$$A\_a = -\left(\left(\frac{1}{2} V\_a^2 \sin(2b) - 4\cos(b)\cos(a+b)g - V\_a^2 \sin(b) + 8\cos(a)g - 2\,V\_a\,V\_b\sin(b) - V\_b^2 \sin(b)\right)\Big/\left(-\frac{3}{2}+\frac{1}{2}\cos(2b)\right)\right)$$

$$A\_b = -\left(\left(3\,V\_a^2 \sin(b) - 8\cos(a+b)g - V\_a^2 \sin(2b) + 4\cos(b)g\cos(a+b) + 8\cos(b)\cos(a)g - V\_a\,V\_b\sin(2b) - \frac{1}{2} V\_b^2 \sin(2b) - 8\cos(a)g + 2\,V\_a\,V\_b\sin(b) + V\_b^2 \sin(b)\right)\Big/\left(-\frac{3}{2}+\frac{1}{2}\cos(2b)\right)\right)$$

Table 5: Equations for the accelerations A_a and A_b of the angles a and b. Parameters V_a and V_b are introduced by the system and represent the velocities of a and b, g is the acceleration due to gravity

Table 6 shows the use of CFunction() to create code for evaluating these accelerations. CFunction() automatically derives which variables are inputs and which (any intermediate variables) are locals. The function may be called with the `optimized` option, in which case, further local variables are introduced for any common sub-expressions in the evaluation.

Code generation in CFunction() uses the Maple function C(). In more complex dynamics problems, with dozens of intermediate variables introduced by the mechanics system and dozens more introduced by the use of the `optimized` option, the capability of automatically generating the wrapper code for a C function was essential.

CFunction() Implementation Note

In the case shown above, we desire code for a sequence of equations, namely for an equation representing A_a and an equation representing A_b. The Maple function optimize() will accept a sequence of equations as input, however it is assumed that the last in the sequence of equations is the only one we actually care about. Hence any of the prior equations may be discarded if it is not deemed necessary in computing the final answer. For example, let's assume we want to generate code for x and y from the sequence of equations: [r=2*a, x=r, y=r*sin(t)].

```
optimize([r=2*a,x=r,y=r*sin(t)]);
```

$$y = 2\,a\sin(t)$$

```
CFunction(CalcAcc,[A_a,A_b]);


void CalcAcc(
        double b,
        double a,
        double g,
        double V_b,
                        double V_a,
        double *A_a_,
        double *A_b_)
{
    double A_a;
    double A_b;
     A_b = -(3.0*V_a*V_a*sin(b)-8.0*cos(a+b)*g-
           V_a*V_a*sin(2.0*b)+4.0\
*cos(b)*g
*cos(a+b)+8.0*cos(b)*cos(a)*g-
V_a*V_b*sin(2.0*b)-V_b*V_b*sin(2.0*b)/2-\
8.0*cos(a
)*g+2.0*V_a*V_b*sin(b)+V_b*V_b*sin(b))/(-
3.0/2.0+cos(2.0*b)/2);
    A_a = -(V_a*V_a*sin(2.0*b)/2-
4.0*cos(b)*g*cos(a+b)-V_a*V_a*sin(b\
)+8.0*cos
(a)*g-2.0*V_a*V_b*sin(b)-V_b*V_b*sin(b))/(-
3.0/2.0+cos(2.0*b)/2);
    *A_a_ = A_a;
    *A_b_ = A_b;
}
```

Table 6: CFunction() can be used to create a C function to evaluate these accelerations.

In order to ensure that all the equations in our output sequence are preserved we use the undocumented Maple function `optimize/statseq()` with as its first

Copyrighted material

parameter the list of equations to be optimized and as its second parameter the list of output variables. This ensures these outputs are preserved.

```
`optimize/statseq`([r=2*a,x=r,y=r*sin(t)],[x,y]);
```

$$[r = 2\,a,\, x = r,\, y = r\,\sin(t)]$$

4. Conclusions

We have presented a system which allows engineering mechanics problems to be specified in an extremely intuitive and graphical manner through the medium of a variational geometry system while a mathematical formulation of the problem within Maple is made available to the user. We have thus, for this problem domain, automated the error prone process of mathematical modeling which generally precedes mathematical analysis. The maintenance of an intermediate variable table enabled us to keep the complexity of this mathematical model under control.

5. References

1. Light R., Gossard D., (1982) "Modification of geometric models through variational geometry", CAD 14:209:214

2. Fuller N. & Prusinkiewicz P. (1989) "Applications of Euclidean constructions to computer graphics", Visual Computer 5:53-67

3. Todd P. (1986) "An algorithm for determining consistency and manufacturability of dimensioned drawings" CAD86 pp 36 - 41, Butterworths, London

4. Todd P. and Cherry G., (1989) "Symbolic Analysis of Planar Drawings", Lecture Notes in Computer Science Vol 358 (P. Gianni ed.) 344-355

5. Todd P. (1989) "A k-tree generalisation that characterises consistency of dimensioned engineering drawings", SIAM Journal of Discrete Mathematics 2:255-261.

6. Todd P. (1992) "A Constructive Variational Geometry Based Mechanism Design Software Package" ASME DE-Vol 46, Mechanism Design & Synthesis, pp 267 - 273

Authors

Philip Todd has an M.A. in Mathematics from Cambridge University, an M.S. in Applied Mathematics from Georgia Tech and a Ph.D. in Mathematical Biology from Dundee University. He is the founder and V.P. of Research and Development of Saltire Software. He is currently the principal investigator of an NSF SBIR grant researching automatic mathematical modelling of geometric and mechanical problems.

Robin McLeod received his Ph.D. in Mathematics from Dundee University in 1972. His research interests lie in numerical analysis and applied geometry. He is a Senior Scientist at Saltire Software. His current research involves human readability of symbolic mathematics, and readability-directed simplification.

Marcia Harris received her B.S. in Mathematics from UCLA, and her M.S. in Computer Science from Oregon State University. Her research interests include the symbolic representation of robot paths.

Todd & McLeod can be contacted at Saltire Software, PO Box 1565, Beaverton OR 97075, tel. 503-622-4055

Harris can be contacted at 1334 SE 52nd Street, Portland, OR

Acknowledgement

This research was partially funded by the National Science Foundation under grant number ISI-9223482

Copyrighted material