A System for the Symbolic Analysis of Problems in Engineering Mechanics

Philip H Todd, Robin J. Y. McLeod, Marcia Harris

Saltire Software
PO Box 1565, Beaverton OR 97075,
U.S.A.
(503) 622-4055 Fax: (503) 622-4537

Abstract

In this paper, we present a system for the symbolic solution of problems in engineering mechanics. Of critical importance is a sub-system for maintaining and manipulating quantities containing unevaluated intermediate variables Without this subsystem, intermediate expression expansion makes symbolic mechanics impractical for all but trivial problems. With the subsystem, readable symbolic solutions may be derived for mechanics problems of textbook complexity and above. More complex symbolic solutions in a form amenable to code generation may be derived for mechanics problems of the complexity found in a practical engineering context.

Introduction

In a previous paper [5] a method was described for performing a symbolic analysis of dimensioned engineering drawings. Analysis of the constraint graph associated with the drawing [6] 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 P1) on line L3 distance b from point P2.
Construct a line (line L5) through point P2 angle e degrees to line L3.
Construct a point (point P4) on line L5 distance a 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.

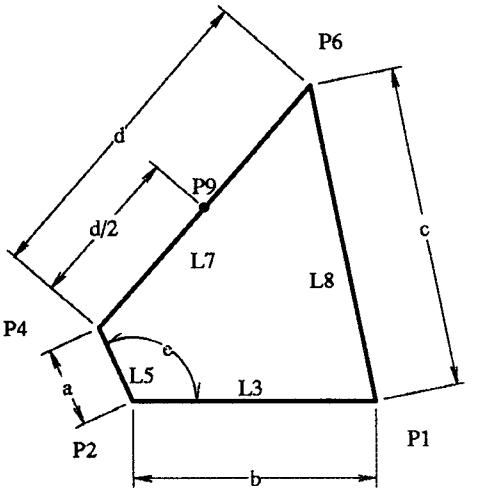

Figure 1: A four bar linkage specified by the lengths of three sides and one angle.

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 constructions are described in [5].

In the previous work, expressions for the coordinates of the drawing's points and the coefficents 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. Attempts to perform analysis of the engineering mechanics of a model subsequent to the derivation of its symbolic geometry exacerbated this problem. In this paper, we report the design and implementation of a symbolic engineering mechanics system which uses intermediate variables throughout the analysis. This eliminates intermediate expression expansion, because the size of the expressions is known a priori. Furthermore, the growth in number of the new intermediate variables is linear in problem size, as opposed to the exponential growth in expression complexity which is experienced without their use.

Intermediate Variables

A comparison of typical output of a computer algebra system with mathematics generated by a mathematician elicits the observation that the mathematician, lacking the machine's facility for analyzing extremely long symbolic expressions, imposes more structure on his mathematics. He does this,

ISAAC 94 - 7/94 Oxford England UK

typically, by extracting subexpressions and renaming them. A single complicated expression is in this way transformed into a simpler expression along with a series of equations defining terms in the expression. The mathematician applies the typically human resources of domain knowledge, intuition, experience, pattern recognition and rules of thumb to this process.

In the context of Constructive Variational Geometry, two different types of symbolic description may be derived from the primary abstract description provided by the construction sequence. The first is a description where the coordinates of each point and the coefficients of each line are expressed purely in terms of the independent variables. The second, and the approach followed here, is a description where intermediate variables are introduced and may appear in the expressions for the coordinates of the points and the coefficients of the lines.

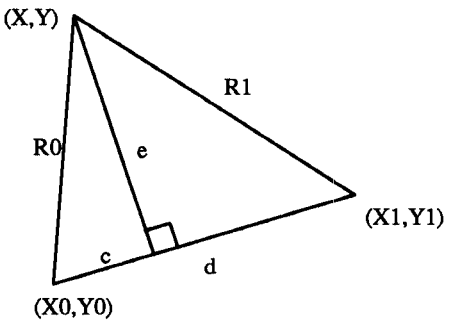

Figure 2: The intermediate variables c, d, e are added in the construction for a point (X,Y) distance R0 from (X0,Y0), R1 from (X1,Y1).

The intermediate variables are introduced during the process of performing the individual constructions. For example the construction for a point (X,Y) distance R_0 from point (X_0,Y_0) and distance R_1 from point (X_1,Y_1) consists of the sequence of formal expressions described in figure 2.

In performing the construction, the system merely substitutes the correct names for the placeholders (X_0,Y_0) and (X_1,Y_1), and generates new names for the intermediate variables c, d, e.

If none of the intermediate variables are subsequently eliminated, then the size of expressions is known a priori simply to be the size of the formal expressions. Also, as the algebra system is not performing any extensive symbolic manipulation, the time taken to perform the geometric construction is short. The set of equations generated using this approach, however, is long and the number of variables introduced may be excessive. Further, the algebra system is given no opportunity to perform any simplification on the expressions as they evolve.

$$d = \sqrt{(X_1 - X_0)^2 + (Y_1 - Y_0)^2}$$

$$c = \frac{d^2 + R_0^2 - R_1^2}{2d}$$

$$e = \sqrt{R_0^2 - c^2}$$

$$X = X_0 + \frac{c(X_1 - X_0) + e(Y_1 - Y_0)}{d}$$

$$Y = Y_0 + \frac{c(Y_1 - Y_0) + e(X_0 - X_1)}{d}$$

The strategy employed in the implementation of our system is to generate an initial symbolic geometry description with a large number of intermediate variables and judiciously to eliminate variables in order to achieve an elegant or "simple" description of the mathematics. Specifically, we apply a measure of complexity to the expression defining each intermediate variable; if this measure falls below a pre-set threshold, the variable is eliminated.

In our prototype system the measure of complexity used is the total number of independent or dependent variables in the expression. For example if a threshhold value of 2 is set, then any intermediate variable which is a constant, or an expression containing a single variable is eliminated.

Geometrical quantities are frequently more succinct when expressed as an implicit rather than an explicit equation. For example, $r^2 = x^2 + y^2$ is slightly preferable to $r = \sqrt{x^2 + y^2}$. Differentiating the implicit form, yields a substantially simpler expression than differentiating the explicit form: $rr' = xx' + yy'$ as opposed to $r' = \frac{xx' + yy'}{\sqrt{x^2 + y^2}}$. On the other hand, the explicit form is needed for evaluation and substitution. Hence it is advantageous to maintain both an implicit and an explicit description of each variable.

In our system, the intermediate variables are maintained in a table (the Intermediate Variable Table or IVT) containing, for each variable: (a) the name of the variable, (b) the left hand side of an implicit definition of the variable, (c) the right hand side of an implicit definition of the variable, (d) the right hand side of an explicit definition of the variable. When a new variable is installed in the table, an implicit definition is given, this is solved by the system to yield an explicit definition.

Shown below is the Intermediate Variable Table description of point P9 in the four bar linkage depicted in Figure 1. In this description, a, b, c, d, and e are input constraint values as drawn on Figure 1. P6x and P6y are the x and y coordinates of point P6, P9x and P9y are the coordinates of point P9. P6c, P6d and P6e are intermediate variables introduced by the system in the process of constructing point P6. These intermediate variables correspond to the variable c,d and e in the definition of the construction for a point a given distance from two known points described in figure 2 above.

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

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

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

$$P6x = b + \frac{P6c(a\cos(e) - b) + aP6e\sin(e)}{P6d}$$

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

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

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

Notice the absence of intermediate variables corresponding to the points P1 and P2, and to all the lines in the figure. While such variables were present in the initial verbose IVT, the system's automatic simplification has eliminated these variables from the table.

Critical to the analysis of problems in engineering mechanics is the ability to differentiate expressions involving intermediate variables. As the intermediate variables may conceal functional dependencies , it is necessary for the IVT to implement its own differentiation. In differentiating an expression y with respect to t, each variable x contained in the expression is inspected. Since the chain rule for implicit differentiation contains the term $\frac{dy}{dx}\cdot\frac{dx}{dt}$, x is checked for membership in the IVT. If x is not contained in the IVT, x is considered constant and hence dx/dt is 0. If x is contained in the IVT, then dx/dt is non zero and the term must be included in the expression for dy/dt. It is important to note that a new intermediate variable, "dx_by_dt", say, is introduced into the IVT to represent dx/dt. In the expression for dy/dt the term $\frac{dy}{dx}\cdot dx\_by\_dt$ is introduced. Elsewhere in the IVT, both an implicit and an explicit equation for x exists. The implicit equation is differentiated to provide an implicit equation for the new variable dx_by_dt. This implicit equation is solved to establish an explicit equation for dx_by_dt. Using the implicit definition in this way results in simpler expressions for the derivatives.

For example, consider the following IVT:

Implicit	Explicit
$x = t^2$	$x = t^2$
$y = \cos(t)$	$y = \cos(t)$
$z^2 = x^2 + y^2$	$z = \sqrt{x^2 + y^2}$

differentiating z with respect to t yields this IVT:

Implicit	Explicit
$x = t^2$	$x = t^2$
$y = \cos(t)$	$y = \cos(t)$
$z^2 = x^2 + y^2$	$z = \sqrt{x^2 + y^2}$
$\frac{dx}{dt} = 2t$	$\frac{dx}{dt} = 2t$
$\frac{dy}{dt} = -\sin(t)$	$\frac{dy}{dt} = -\sin(t)$
$z\frac{dz}{dt} = x\frac{dx}{dt} + y\frac{dy}{dt}$	$\frac{dz}{dt} = \frac{x\frac{dx}{dt} + y\frac{dy}{dt}}{z}$

Differentiating a second time from the implicit equation gives the following table entry for the second derivative of z:

$$\frac{d^2z}{dt^2} = \frac{x\frac{d^2x}{dt^2} + \left(\frac{dx}{dt}\right)^2 + y\frac{d^2y}{dt^2} + \left(\frac{dy}{dt}\right)^2 - \left(\frac{dz}{dt}\right)^2}{z}$$

By contrast, a process where the explicit equation was differentiated at each step would yield the following expression:

$$\frac{d^2z}{dt^2} = \frac{\left(x\frac{d^2x}{dt^2} + \left(\frac{dx}{dt}\right)^2 + y\frac{d^2y}{dt^2} + \left(\frac{dy}{dt}\right)^2\right)(x^2 + y^2) - \left(x\frac{dx}{dt} + y\frac{dy}{dt}\right)^2}{(x^2 + y^2)^{\frac{3}{2}}}$$

Engineering Mechanics

Given a capability for the symbolic analysis of dimensioned drawings, an engineering mechanics system may be built in a manner described formally as follows [7].

Let **u** be the vector of free parameter values (for example in the double pendulum of figure 4, **u**=(a,b)). Let there be m points of interest in our drawing and n lines of interest (a point of interest is a mass center or a point of force application, a line of interest is a line with non zero moment of inertia or a line of torque application). The above geometric analysis yields the position of point i ($1 \le i \le m$) as a function of these constraints: $(\xi_i(\mathbf{u}),\eta_i(\mathbf{u}))$. Let μ_i be the mass of point i and let (ϕ_i , ψ_i) be the force applied to point i. The angle of line j ($1 \le j \le n$) may also be derived again as a function of the free constraints: $\theta_j(u)$. Let I_j be the moment of inertia of line j and let T_j be the applied torque.

Let **x** be a generalized position vector, **m** the generalized mass vector and **f** the generalized force vector defined as follows:

$$x_{2i-1} = \xi_i, \quad m_{2i-1} = \mu_i, \quad f_{2i-1} = \phi_i \quad (1 \le i \le m)$$
$$x_{2i} = \eta_i, \quad m_{2i} = \mu_i, \quad f_{2i} = \psi_i \quad (1 \le i \le m)$$
$$x_{2m+j} = \theta_j, \quad m_{2m+j} = I_j, \quad f_{2m+j} = T_j \quad (1 \le j \le n)$$

Let p be the generalized momentum vector defined as follows:

$$p_i = m_i \sum_j \frac{\partial x_i}{\partial u_j} \cdot \frac{du_j}{dt}$$

At kinetostatic equilibrium, the reaction pseudoforce F_j in parameter u_j is defined by the equation:

$$F_j = \sum_i (f_i - \frac{dp_i}{dt}) \cdot \frac{\partial x_i}{\partial u_j}$$

Depending on the nature of parameter u_j, this pseudoforce has different interpretations. If u_j is the length of a line, then the pseudoforce represents the force needed to prevent the length of the line from changing. If u_j is an angle, the pseudoforce represents the torque required to maintain the value of the angle. While the pseudoforce is well defined even for parameters which do not directly constrain lengths or angles, the physical interpretation may be less obvious.

The computation of pseudoforces in parameters represents a solution to the inverse dynamics problem: that of deriving forces for given known motion. The forward dynamics problem, that of deriving motion from given applied forces, is conveniently phrased in variational geometry terms as follows. Given a set of applied forces **f** and parameter values **u** and velocities **u'**, find expressions for parameter accelerations **u''** for a subset of parameters which are free to accelerate in response to the unbalanced forces. A solution to the forward dynamics problem may be formulated from the observation that while in dynamic equilibrium, the free constraints must transmit zero force. Hence if constraint j is free,

$$\sum_i (f_i - \frac{dp_i}{dt}) \cdot \frac{\partial x_i}{\partial u_j} = 0$$

Now, differentiating p gives:

$$\frac{dp_i}{dt} = m_i \sum_k \frac{\partial x_i}{\partial u_k} \cdot \frac{d^2 u_k}{dt^2} + m_i \sum_k \frac{\partial^2 x_i}{\partial u_k^2} \cdot \left(\frac{du_k}{dt} \right)^2$$

Hence the equation of motion may be written as the following linear system for **u''**:

$$\sum_k \left(\sum_i m_i \frac{\partial x_i}{\partial u_j} \frac{\partial x_i}{\partial u_k} \right) \frac{d^2 u_k}{dt^2} = \sum_i \left(f_i - m_i \sum_k \frac{\partial^2 x_i}{\partial u_k^2} \cdot \left(\frac{du_k}{dt} \right)^2 \right) \frac{\partial x_i}{\partial u_j}$$

For examples where there are several free parameters, it is perhaps most useful for a symbolic mechanics system to present the equations of motion in the above form and leave solution of the linear system to be done numerically. For examples with one or two free parameters, the system may be solved symbolically to give direct acceleration expressions.

System Implementation

Our symbolic engineering mechanics system has three major components. (i) The front end is a customized version of Saltire Software's Analytix package [8]. Analytix is a constraint based numerical engineering analysis package; as such it has all the user interface elements necessary for the graphical specification of problems in engineering mechanics. (ii) The front end accesses OEM Maple as a Dynamic Linked Library and has its own textual interface with Maple. (iii) A further system component comprises the intermediate variable table and the symbolic geometry and dynamics code. This is implemented as a Maple library, which is called by OEM Maple.

To specify a mechanics problem, the user sketches the geometry in Analytix and adds symbolic constraints to specify distances and angles. The user then adds external forces or torques, masses or moments of inertia to the figure. When the model is specified, the user opens a Maple analysis window. At this point the symbolic geometry code is invoked and an IVT created containing the coordinates of all the points and the coefficients of all the lines in the figure, along with such intermediate variables as were added by the system.

Two different engineering mechanics functions are available to the user. Reaction(x) gives the pseudoforce in parameter x. InstallDiffEq([x_0,...,x_n]) installs new variables A_x_0,...,A_x_n representing the accelerations of x_0,...,x_n into the IVT. 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.

A further component of our system is a code generation routine which creates a C function for the evaluation of one or more variables defined in the IVT. The resulting C function has as outputs the requested variables, as inputs all the independent variables required to evaluate the outputs and as locals all the intermediate variables, plus any intermediate variables generated by the Maple optimizer. For example, applied to the 4 bar linkage example of figure 1, the following routine was generated for computing the acceleration of the crank angle A_e:

```
void CrankAcc( double a, double b, double e, double d, double g, double
V_e, double c, double *A_e)
{
      double t133;
      double P6c;
      double t122;
...    158 more lines of code    ...
}
```

In the above, a, b, c, d, e are parameters of the model, g is the gravitational acceleration and V_e is the velocity of parameter e. A_e is the output acceleration.

Examples

Our first example derives the formula for the torque in a piston crank with unit force applied at the piston.

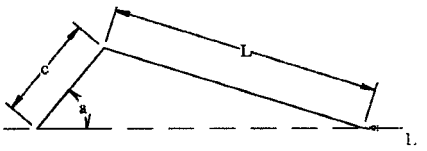

Figure 3: A crank piston with unit force applied to the piston.

From the user's perspective, this problem is solved as follows. The user draws the model in the form depicted in Figure 3, then invokes the symbolic mechanics system. He introduces the variable *torque* into the IVT table using the defining expression torque=Reaction(a). Without further user intervention, our symbolic mechanics system gives the following expression for torque.

$$P8a^2 = L^2 - c^2\sin(a)^2$$

$$2P8a\frac{dP8a}{da} = -2c^2\sin(a)\cos(a)$$

$$torque = -c\sin(a) + \frac{dP8a}{da}$$

In the above expressions, P8a is a geometric intermediate variable introduced by the system.

Our second example considers a model of a double pendulum with unit link lengths and with unit masses at the end of each link (figure 4). Without simplification, the output from the InstallDiffEq function had 30 intermediate variables. The use of an automatic IVT simplification routine generated the following expressions:

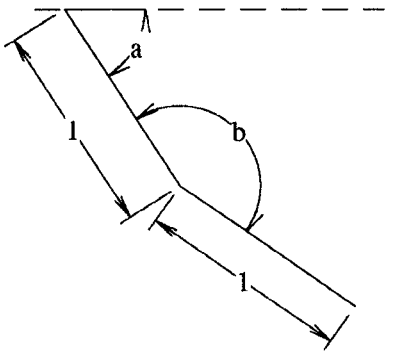

Figure 4: A double pendulum model has free parameters a and b.

$$fbJ2 = V\_a^2\sin(b) - 4\cos(a+b)g$$

$$fbJ1 = 8\cos(a)g - 2V\_aV\_b\sin(b) - V\_b^2\sin(b) - 4\cos(a+b)g$$

$$AJ11 = 3 - 2\cos(b)$$

$$AJ21 = -\cos(b) + 1$$

$$A\_b = \frac{AJ11fbJ2 - AJ21fbJ1}{AJ11 - AJ21^2}$$

$$A\_a = \frac{AJ21fbJ2 - fbJ1}{AJ11 - AJ21^2}$$

In the above, V_a, A_a, V_b and A_b are the velocities and accelerations of angles a and b, while AJ11, AJ21, fbJ1, fbJ2 are intermediate variables introduced by the system.

The IVT simplification routine works by adaptively eliminating intermediate variables contained in the expressions for intermediate variables then looking for simplification in these expressions. A measure of the complexity of the IVT is used to direct this process; currently the measure used is the sum of a weighted operation count for each expression in the IVT. We are currently working on the development of more sophisticated complexity measures.

The examples shown above are small, and hence it is quite possible to eliminate all the intermediate variables while retaining a workable symbolic description. This is emphatically not the case, however, for even slightly more complex examples.

Concluding Remarks

Research in the computer algebra community on the symbolic representation of geometry has centered on the topic of automatic geometry theorem proving, for example [2,4]. For the interpretation of engineering problems, it is more important to derive formulas than prove theorems. Wu and Chou have developed techniques for automatic derivation of geometric formulas [1]. These are based on their methods for geometric theorem proving. In recent work [3] Chou and Gau have begun to apply their techniques to problems in the geometric analysis of mechanisms, however performance is a major issue in the practical application of these techniques in an engineering context.

Two important uses of a symbolic description of an engineering problem are design documentation and code generation. In design documentation, the behavior of a system is captured in the form of a mathematical description. This should be human-readable and suitable for archiving in a paper form, for publishing and for incorporating in patent applications. In code generation, the behavior of the system is captured as a module of computer source code. This code may then be incorporated in a computer simulation. In both of these uses, the presence of intermediate variables in the symbolic description is quite acceptable, indeed beneficial. This is in contrast to the theorem-proving domain where, to achieve the sought cancellation, intermediate variables must typically be eliminated.

A symbolic mathematics system, such as Maple does not consider a sequence of equations as having a collective meaning; common subexpressions which Maple may contain are maintained for the purposes of optimizing the storage of a single expression. On the other hand, the approach presented in this paper maintains the collective meaning of a sequence of equations, allowing the development of an efficient symbolic mechanics system.

Acknowledgement

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

Bibliography

1. Chou S.C. (1987) "A Method for the Mechanical Derivation of Formulas in Elementary Geometry" *Journal of Automated Reasoning 3, 291-299.*
2. Chou S.C. (1987) *Mechanical Geometry Theorem Proving*, Kluwer Academic Publishers, Hingham, MA.
3. Chou S.C. & X.S. Gao (1989) "Automated Reasoning in Mechanics" *Technical Report* 89–11, *Dept of Computer Science*, University of Texas at Austin.
4. Kutzler B. & S. Sifter (1986) "Automated Geometry Theorem Proving Using Buchberger's Algorithm", in *Proceedings of the 1986 Symposium on Symbolic and Algebraic Computation* pp 209-214.
5. Todd P. and Cherry G., (1989) "Symbolic Analysis of Planar Drawings", *Lecture Notes in Computer Science Vol 358 (P. Gianni ed.) 344-355.*
6. Todd P. (1989) "A k-tree generalisation that characterises consistency of dimensioned engineering drawings", *SIAM Journal of Discrete Mathematics 2:255-261.*
7. Todd P. (1992) "A Constructive Variational Geometry Based Mechanism Design Software Package" ASME DE-Vol 46, Mechanism Design & Synthesis, pp 267 - 273.
8. Todd P. and Kemper H. (1993) Analytix Version 2.0 Users' Manual, Saltire Software, Beaverton OR.