BUILDING A TOOL TO BUILD SYSTEMS

B. Moll*, Wm. J. Garland and T.J. Kennett

Department of Engineering Physics

McMaster University

The development of an interactive computer-aided software tool, named EQNWRITE, to help in the setup and manipulation of engineering mass, momentum and energy equations is discussed in this paper. The software tool is a building tool for building mathematical models of process systems. For a given problem, EQNWRITE enables the user to interactively derive and simplify the governing equations in the desired reference frame and coordinate system. At the user's request, specified mathematical transformations are performed, automatically and resulting equations are displayed. EQNWRITE allows the user to better perceive the physical significance of each term in the symbolic mathematical equations, which describe the dominant properties of a system, as they are being derived and simplified, while leaving the detailed mathematical transformations to the computer.

EQNWRITE was programmed in the symbol manipulation language, MUSIMP and developed using some of the capabilities of the microcomputer based computer algebra system, MUMATH. Major programming features and capabilities of MUSIMP and MUMATH are briefly reviewed, followed by a discussion of the specific automatic transformations implemented in EQNWRITE. Finally, engineering examples are given showing the use of the transformation functions and operators.

---

* Presently with ESSO Petroleum Canada, 55 St. Clair Ave. W., Toronto, Canada.

Presented at the 11th Simulation Symposium on Reactor Dynamics and Plant Control, Royal Military College, Kingston, Ontario, April 22-24, 1985.

1.0 Introduction

The concept of an interactive computer-aided software tool to help in the set up and manipulation of equations is proposed here. Conceptually, for a given problem, the tool is to enable the user to interactively derive and simplify the governing equations in the desired reference frame and coordinate system (ie. Cartesian, cylindrical or spherical). At the user's request, specified mathematical transformations are to be performed automatically and resulting equations are to be displayed.

Using this tool, general mathematical similarities and differences between the differential equations describing the transport phenomena would be better highlighted. Also, the user can go back to basics to derive an equation rather than just modifying an existing simplified equation as the modification may not be compatible with simplifications already employed. For example, an empirical relation is often used to account for momentum flux in one dimensional pipe flow. However, momentum flux can be explicitly expressed in the more general two dimensional equations.

Overall, this software tool is a building tool for building mathematical models of process systems. It allows the user to better perceive the physical significance of each term in the symbolic mathematical equations, which describe the dominant properties of a system, as they are being derived and simplified, while leaving the detailed mathematical transformations or operations to the computer.

The development of EQNWRITE, the interactive computer-aided software tool capable of building and manipulating symbolic mathematical relationships for the modelling of transport phenomena such as mass transfer, heat transfer and fluid flow, required tools and techniques used for symbolic processing as opposed to those used for numeric processing. EQNWRITE was programmed in the symbol manipulation language, MUSIMP and developed using some of the capabilities of the microcomputer based computer algebra system, MUMATH. Major programming features and capabilities of MUSIMP and MUMATH are briefly reviewed, followed by a discussion of some of the specific automatic transformations implemented in EQNWRITE. As an example of how EQNWRITE works, the derivation of the heat conduction equation for determining the temperature distributions in boiler tube walls, piping walls, reactor fuel pencils etc., from the thermal form of the energy equation is presented.

2.0 MUMSIMP - Symbol Manipulation Language

MUSIMP (Micro Symbolic IMPlementation Language) is an interpretive programming language developed by Albert Rich of the Soft Warehouse Co. (reference 1), during 1977, for implementing computer algebra systems and other artificial intelligence applications. It is a variant of LISP but has been optimized for the manipulation of symbolic mathematical expressions and improved for readability (ie. it uses prefix, infix and postfix notation rather than the nested parenthetical syntax of LISP).

All functions and data in MUSIMP are made up of atoms and nodes which are the basic MUSIMP data objects. Atoms are either names or integers. A name is a letter followed by zero or more letters or digits. A node is used to join two data objects together. It is made up of two cells, each of which points to a MUSIMP data object. When a name is initially created, its value is set to the name itself.

The primary data structure in MUSIMP is a binary tree. As the name implies, it is a structure with a root, branches and leaves. The first node is the root. The root and each subsequent node of the tree branches in two directions. Atoms are the leaves of the tree.

A list structure is a special form of a binary tree. It is a sequence of nodes whose first cell points to the data objects and whose second cells link the sequence together.

Any symbol manipulation language must be able to build and take apart symbolic expressions. The MUSIMP functions, FIRST and REST are used to take lists apart. FIRST returns the first element in a list and REST returns a list minus the first element. Different combinations of these two functions can be created to extract other elements from a list. Constructor functions build new data structures. ADJOIN takes two objects (ie. objects may be atoms, binary trees or lists) and creates a new node. LIST takes one or more arguments, evaluates them and returns a list with them as elements. APPEND takes one or more arguments, all of which must be lists and generates a list from the elements of each list.

Other functions include comparator functions, mathematical operators, boolean logical operators and string functions.

There are three programming techniques available to implement new user defined functions or to expand on the capabilities of available functions. They include (1) function definitions, (2) property list additions, and (3) driver modifications.

Function definitions in MUSIMP bear a resemblance to their counterparts in traditional structured programming languages and their major differences are due to an emphasis on expressions rather than statements. Expressions can be seen as specifying procedures for the evaluation of functions.

Another method of implementing transformations on expressions is to use the MUSIMP data base property list primitives to assign new properties to existing functions. This can be done without having to understand, rewrite or consult the built-in definition. Any name including operator and function names can have a number of property values indexed according to various property keys. A name's property list is a list of data pairs. The first cell of the node is called the "key" and the second cell of the node is an expression called the "associated information". This property concept makes it possible for a name to have many values each of which is associated with a specific key.

This type of programming is known as "data driven" programming in the field of artificial intelligence. There are generally two main types of programming for type specific functions. Placing specific information into general purpose functions is one method and the other is placing type information on property lists. In the latter, functions are fetched when an argument of the type appears.

The last method of implementing transformations involves the MUSIMP DRIVE function. The DRIVE function cyclically prompts, parses, evaluates and deparses expressions. It does not automatically update the assignments made to unbound variables in previous expressions containing these variables. Extensions could be made to the driver functions to implement the above step in the evaluation mechanism.

3.0 MUMATH - Computer Algebra System

MUMATH is an interactive symbolic math system which is used to transform and simplify formulas into equivalent formulas, either automatically or at the explicit request of the user. It is organized into a nonredundant additive hierarchy of twenty-two files. Some of the capabilities of the MUMATH system include indefinite precision rational arithmetic, matrix algebra, trigonometric and logarithmic manipulation, symbolic integration, symbolic summation and the solution of nonlinear differential equations. All symbolic math expressions are represented internally as atoms, functional forms or operator forms. A functional form is a list with the function name as the first item followed by successive operands as the subsequent items of the list.

Besides using function definitions for transforming expressions, MUMATH algorithms use rule definitions. In MUMATH rarely are all the rules for a function or an operator placed in one big function definition. The main disadvantage in this method is that every time a new function or operator is defined, the user must go back and insert new lines in every function that interacts with the new function or operator. Also, the last file read would destroy the extension provided by the previous file unless both files contained the same extension. To solve the redundancy and constraint problems with file combinations, a MUMATH function definition only treats the trivial cases and then searches the function name's property list for an appropriate transformation rule. In MUMATH, a transformation rule is stored as a replacement function on the property list of a top level operator keyed to a second level operator. Each rule is a separate function which returns false if it isn't applicable.

The next section discusses some of the specific automatic transformations implemented in EQNWRITE using the capabilities of MUSIMP and MUMATH. EQNWRITE automates some of the mathematical operations and identities used in transport phenomena modelling.

4.0 EQNWRITE

EQNWRITE was developed on the IBM-PC microcomputer. Before EQNWRITE can be read in, a system file ENVIRON.SYS containing preselected available MUMATH capabilities must be loaded in first. A minimum of 70K of memory is required to run ENVIRON.SYS and EQNWRITE. The preselected capabilities include algebra, vector algebra, vector calculus transformations and solving an equation for an unknown variable.

4.1 Integral Operators and Time Derivatives

The operators $III, $II and $I were introduced first in EQNWRITE to represent volume, surface and line integrals respectively. These operators were declared as prefix and matchfix (ie. operator requires a delimiter) operators. For example, the delimiter for the $III operator is "dV". After having established the operators (ie. programming in the appropriate MUSIMP instructions on how to parse and output expressions containing the integral operators) an associated function for each was created, which specified how to apply the operator to its operand. No evaluation of the operands was required in the function except for the case when the operand is equal to zero. It was assumed that during the initial stages of the derivation of the basic equations, evaluation of integrals is not required. However, for future consideration, evaluation of integrals can be implemented by building flags into the associated functions of the integral operators. Evaluation of the integrals would then occur when the flags are set to some predefined values. Integration rules would be stored on the property list of the integral operators.

EQNWRITE implemented two time derivatives which exist when analyzing fluid flow systems. The partial time derivative, $\partial\psi/\partial t$, measures the rate of change of a property, ψ, with respect to time when the observer is at a fixed point in space. This observation is known as the Eulerian point of view. The substantial time derivative, $D\psi/Dt$, represents the rate of change of ψ with respect to time that an observer would measure while moving at the same velocity of the fluid. This is also known as the Lagrangian point of view.

In EQNWRITE, the substantial time derivative, D/Dt, was declared as a prefix operator with an associated function. To represent the partial time derivatives, the MUMATH global variable, DIFVAR (ie. differentiation variable) was declared equal to 't', and the operator, '\', was used to represent differentiation with respect to the variable, t. Therefore, the expression, ψ\, would be equivalent to $\partial\psi/\partial t$.

4.2 Reference Frame Transformations

Two mathematical identities, Reynold's Transport Theorem and Gauss' Divergence Theorem, are implemented in EQNWRITE using similar MUMATH programming capabilities. The Reynold's transport theorem is used to transform substantial derivatives of volume integrals into equivalent expressions consisting of volume integrals of Eulerian derivatives. Mathematically, the theorem states:

$$\frac{D}{Dt} \int\int\int_V \psi(t)\, \mathbf{v} \cdot \mathbf{n}\, dS + \int\int\int_V \partial\psi/\partial t \, dV$$

where:	ψ	= field variable
	V	= arbitrarily chosen volume
	D/Dt	= substantial time derivative
	v	= velocity of the field variable
	$\partial/\partial t$	= local time derivative
	n	= unit vector normal to the surface
	s	= surface bounding the volume, V.

In EQNWRITE, the left hand side of the general form of the theorem was stored on the function name's property list keyed to the indicator, 'OLD. The right hand side of the theorem was stored on the same property list but keyed to the indicator, 'NEW. When the user enters an equation to be transformed to the Reynold's Transport Transformation function, RTT, then any occurrences of the substantial derivative of a volume integral appearing in the left hand side of the incoming equation is replaced by an expression containing Eulerian derivatives. This is also done for the right hand side of the incoming equation. Finally the two sides of the equations are equated and returned.

A second way of implementing the Reynold's Transport Theorem would have been to declare the theorem as a rule on the property list of the operator, "D/Dt" keyed to the integral operator, "$III". Then everytime an expression with the top level operator as "D/Dt" and the second level operator as "$III" is entered, the expression gets transformed into one containing Eulerian derivatives.

A function called GDT was created to implement Gauss' Divergence Theorem, a mathematical identity which transforms a surface integral into a volume integral. Mathematically stated, the theorem is:

$$\int\int_S \mathbf{A}\cdot\mathbf{n}\,dS = \int\int\int_V \nabla\cdot\mathbf{A}\,dV$$

where $\mathbf{A}$ = any vector

∇ = Del operator

(i.e. $\nabla = \frac{\partial}{\partial x} i + \frac{\partial}{\partial y} j + \ldots$)

Similar to the RTT function, the left hand side of the theorem is stored on the property list of GDT keyed to OLD and the transformation is stored on the property list under the key, 'NEW. The theorem could also have been implemented as a rule. The rule would be stored as a function on the property list of "$II" keyed to the DOT operator.

4.3 Operator Deleter

Macroscopic balances describe input/output relationships (ie. the amount of a variable being transferred) whereas microscopic balances are used for the calculation of profiles of such variables as velocity, temperature and concentration within a system. The function, DELINT, was created to transform the macroscopic form of a conservation equation (ie. integral form) to its microscopic form (ie. differential form), provided all the integrals in the equation are of the same type (ie. volume integrals). The assumption inherent in this transformation is that the macroscopic equation refers to an arbitrary volume.

Given an equation, DELINT, initially checks that all the terms on a given side of the equation is in the correct integral form. For a given side of the equation, if all integrals have the same integral operator, then DELINT through a combination of MUSIMP symbolic extractor and constructor functions, removes the integral operator from each term. If not all the integrals have the same integral operator or if one of the terms is not an integral then FALSE is returned. Then a check is made to see if a side has been assigned to FALSE and if so, the original equation is returned. Otherwise, the transformed halves of the equation are equated and returned.

4.4 Grouping Terms

When working with integral equations where each term is an integral, it is desirable at times to group together integrands with similar attributes. For example, if in a system we were considering both the liquid and gas phase for the macroscopic total mass balance, then it would be desirable to have a function which could group the integrands for the local time derivative terms together, the integrands for the advection terms together and the integrands representing the local sum of sources minus the local sum of sinks together as opposed to a function which simply groups all the integrands together.

One basic way of accomplishing this is to pass the expression to be transformed to a function along with a list of terms in that expression which are to be grouped. The function GROUP, was created to do the above and the following example illustrates how it works.

If the variable EQN is assigned the following value:

```
EQN: A + B + C - D + E - F - G + H $
```

and we wanted to group the first, third, fourth and eighth terms together (ie. A, C,D,H), then the following command would be entered:

```
GROUP (EQN,LIST(1,3,4,8)).
```

The output would be:

$$A + C - D + H + (B + E - F - G).$$

Another approach for grouping terms with similar attributes is to initially tag each term with an identifying label and then have a function which groups all terms with identical labels together. The ADDINTG function was created specifically to group integrals with identically tagged integrands together. Basically, ADDINTG checks the property list of TAG under different keys (note in this case, terms in an expression are the key names) and groups together the terms with the same associated information.

The appendix of this paper gives an example of an interactive session with EQNWRITE. The example illustrates how some of the MUSIMP and MUMATH capabilities were used to derive the heat conduction equation for determining temperature distributions in such areas as boiler tube walls, piping walls and reactor fuel pencils.

In summary, a program, EQNWRITE, was written in MUSIMP and developed using some of the capabilities of the computer algebra system, MUMATH, to derive and simplify engineering mass, momentum and energy equations. The capabilities of EQNWRITE include (1) transformation of an equation from the Lagrangian reference frame to the Eulerian reference frame, (2) deletion of a common integral operator from all terms in an equation under specific conditions, (3) implementation of the partial time derivative and substantial time derivative and (4) functions to regroup terms according to user requirements.

The capabilities mentioned above are, however, only a small subset of what the software tool should be capable of performing in the future. Some possible future developments include (1) the set up of a database containing empirical correlations, (2) the implementation of tensor operations, (3) the creation of functions which can format equations for various numerical methods, and (4) the implementation of functions in such a manner that they can be parameterized for different engineering domains.

More details can be found in reference 2.

References:

1. "The Mumath-83 Symbolic Mathematic System Reference Manual", The Softwarehouse Co., P.O. Box 11174, Honolulu, Hawaii, 96828-0174.
2. "Computer-Aided Derivation of Transport Phenomena Equations", B.Quon-Moll, M.Eng. Phys. Thesis, McMaster University, Hamilton, Ontario, 1985.

APPENDIX: EXAMPLE TERMINAL SESSION WITH EQNWRITE

The thermal form of the conservation of energy equation is transformed to the heat conduction equation below.

```
%Example 3:
```

Derivation of the heat conduction equation for determining the temperature distributions in boiler tube walls, piping walls, reactor fuel pencils etc. from the thermal form of the energy equation.

The following variables appear in the thermal form of the energy equation:

p	= density,
e	= internal energy,
v	= velocity vector,
q	= heat flux vector,
E	= internal heat sources and sinks,
P	= pressure,
TAU	= shear stress.

Declaring the above variables' TYPES and DEPENDENCIES .... %

```
;

Abort, Break, Continue, DOS? C

?

v: 'v $

?

NONSCALAR (v,q) $

?

DEPENDS (q ('X,'Y,'Z,t)) $

?
```

```
%The thermal form of the energy equation is ...

THERMAL: "$III" (p*e) "dV" + "$II" (p*e*v DOT n) "dS" = =

          -"$II" q DOT n "dS" + "$III" E "dV" + "$III" (TAU GRAD v) "dV"

          -"$III" (P DIV v) "dV" ;

@: $III (e p) 'dV + $II e p n DOT v dS = = $III E dV - $III P DIV v dV + $III

TAU GRAD v dV - $II n DOT q dS

Abort, Break, Continue, DOS? C

?

%If the fluid velocity, v, is set to zero ...  %

v: 0 $

?

THERMAL: EVAL (THERMAL):

@: $III (e p) dV = = $III E dV - $II n DOT q dS

%Converting surface integrals to volume integrals, and then dropping the volume integrals

the microscopic form of the equation is obtained ....   %

THERMAL: GDT (THERMAL, 1, q):

@: $III (e p) dV = = $III E dV  - $III DIV q dV

Abort, Break, Continue, DOS? C

?

THERMAL: DELINT (THERMAL, '"$III"):

@: (e p) ' = = E - DIV q

Abort, Break, Continue, DOS? C

?

%The left hand side term of THERMAL, from thermodynamics, can be expressed as a function

of temperature, T and heat capacity, Cv as shown below ....   %

DEPENDS (T ('X, 'Y, 'Z, t)) $

?
```

```
NEWpe: (p*Cv) * T';

@: p Cv (T)'

Abort, Break, Continue, DOS? C

?

%Substituting the above expression into THERMAL .... %

THERMAL: SUB (THERMAL, SECOND(THERMAL), NEWpe);

@: p Cv(T)' == E - DIV q

%From Fourier's law for heat conduction, the heat flux vector is equal to the following .... %

q: - k GRAD T ;

@: -k GRAD T

Abort, Break, Continue, DOS? C

?

%Expanding THERMAL further using the above substitution .... %

VECEXPD: 3 $

?

THERMAL: EVAL (THERMAL):

@: p Cv(T)' == E + k DIV GRAD T

Abort, Break, Continue, DOS? C

?

%The above equation is now in the form useful for determining temperature distribution.
Correlations specific for a case can be substituted in for Cv and k.
```