

## FROM EMBODIMENT GENERATION TO VIRTUAL PROTOTYPING

R. H. Bracewell and A. L. Johnson

*Keywords: Embodiment, Design Constraints, Object-Oriented Modelling, Virtual Prototyping*

### 1 Introduction

The Embodiment Generation project at Cambridge Engineering Design Centre is researching the creation of software tools to aid the designer in the rapid generation and evaluation of feasible embodiment designs. This involves managing a large set of design variables connected by various types of constraints. Using a product model and a library of generic components, tools have previously been produced to support the specification and satisfaction of design constraints (CADET [9]) and the solution of kinematic constraints (Promech [2]).

This paper shows how, by harnessing recent technological advances in the field of Object-Oriented Physical Systems Modelling [7], much greater ease of use, rigour and computational efficiency in design constraint management can be achieved [1]. Additionally, this approach provides "virtual prototype" animated dynamic models of the resulting design artefacts, for subsequent evaluation and optimisation. These have built-in design constraint checking, ensuring that at every instant of a simulation, no limits have been exceeded, whilst the model behaves realistically according to the relevant physical laws.

### 2 Limitations of the CADET software

#### 2.1 Equation Handling

The most fundamental limitation of CADET is its equation handling strategy. If an equation is specified, then the software sensibly attempts to remove it from the constraint set, by eliminating its dependant variable from all other constraints. However, it can only do this if the substitution would not introduce 2 or more instances of any variable into any constraint. The result is that many equations generally remain in the constraint set, where they are solved by a second approach. This is to treat each equation as a pair of "back-to-back" inequalities, defining a small acceptable error either side of the exact value, then solving them both using a search technique. Unfortunately, whilst this approach has the benefit of avoiding the need to deal with simultaneous equations, it has the huge drawback that search techniques are highly inefficient at solving equations. This is because, for a reasonable degree of accuracy, the legal strip between the back to back inequalities is necessarily narrow. Solving them using the iterative re-assignment of variables to a randomly chosen value, as used in CADET, is like a blindfold "pin the tail on the donkey" game – the chances of success on each attempt are slim indeed.

A much more efficient approach would be to employ a 2-stage process, firstly solving the parametric and governing *equations* of the device to find the instantaneous values of all variables, then substituting those values into the *inequalities*, to determine which are satisfied and which are not. This allows efficient graph-theoretic algorithms to be applied in solving the equation set. In these, the equations may be sorted and symbolically rearranged, attempting to achieve a computationally feasible sequence. This is one, such that at the point that each equation is evaluated, the values of all of its independent variables have already been determined from the evaluation of previous equations. This is termed the *reduction* of the equation set. Where one or more algebraic loops exist, this is not completely possible, and the best that can be done is *decomposition* - the minimisation of the size of the embedded systems of simultaneous equations. This is not a new idea - an efficient algorithm to perform this task was described by Tarjan [8]. The modelling of practical engineering devices often leads to large, sparse, frequently non-linear systems of simultaneous equations. The stability of conventional numerical techniques for solving such systems decreases rapidly with size and they are at best highly inefficient. However, they can be solved in a robust and efficient fashion using an even older technique, which seems recently to have been rediscovered, known as *tearing* [3]. In this, iteration variables are judiciously chosen, the values of which, when known, break the algebraic loop. These techniques are well described in [5], which lists 13 alternative published algorithms for effectively tearing arbitrary sets of simultaneous equations. Practical systems generally employ heuristics, choosing between alternative approaches. Commercial software packages supporting these techniques are now available, including IDA ([www.brisdata.se](http://www.brisdata.se)), and Dymola ([www.dynasim.se](http://www.dynasim.se)), which are amongst the leading Object-Oriented Physical System Modelling and Simulation tools. Dymola has been chosen for use in this project, for reasons that will be expanded upon below.

## 2.2 3D Modelling of Forces and Moments

A second problem with CADET is the lack of rigour in specifying forces and moments. To reduce the risk that a device will fail in a way that has not been considered, all 6 generalised orthogonal forces at each mechanical interface should be represented. CADET represents solid geometry and points of interest (POI) of generic components whilst interfaces allow relative POI positions and orientations of mating components to be maintained. However, despite the fact that forces are invariably applied at POIs and aligned with particular axes, there is no system for tagging the force variables with this information. Thus the computer is unable to generate automatically the equations relating to transmission of forces and moments across an interface. Instead, the user must remember to specify each one manually. This problem is directly addressed in Dymola. *Cuts* (interfaces) may be defined, which precisely specify the variables involved in a connection between objects, so that the resulting equations can be automatically generated. For example, a mechanical cut may be defined as follows:

```
terminal Sa(3,3), r0a(3), va(3), wa(3), aa(3), za(3), fa(3), ta(3)
cut a (Sa,r0a,va,wa,aa,za / fa, ta)
```

The terminal statement declares the dimensions of array variables defined in the cut. Sa and r0a are the global rotation matrix and position vector; va, wa, aa and za are global translation and angular, velocity and acceleration vectors; fa and ta are the force and torque vectors. The cut statement contains a list of “across” variables and a list of “through” variables separated by a “/”. When multiple cuts are connected together, Dymola knows how to generate equations which make corresponding “across” variables of the connected objects equal, whilst equating the sum of corresponding “through” variables to zero, as in <sup>st</sup> and 2<sup>nd</sup> laws.

### 2.3 Lack of 3D Dynamics Capability

A third serious limitation with CADET is the lack of an integrated 3D dynamics solver. This means that constraint satisfaction is necessarily at an instant rather than over a complete operating run. However, the critical instant to design for may not be obvious, and there may be more than one. Additionally inertia forces are ignored, as are issues such as dynamic balancing. Linking to a specialist 3 dynamics package, whether commercial (such as ADAMS, DADS) or research (such as Promech [2]) is not a trivial task. This would mean synchronising the operation of 2 equation set solvers – one dealing with the dynamics equations, the other dealing with the rest. This is not an elegant solution. Fortunately, Dymola has the answer again, in the shape of its remarkable multi-body system (MBS) object class library [7]. A model comprising connected instances of MBS and other classes, plus additional governing and parametric equations and constraint inequalities, is translated by Dymola into a single large differential and algebraic equation (DAE) system. If kinematic loops are involved, this will inevitably contain large sparse non-linear simultaneous equation systems, but these are dealt with efficiently by the tearing algorithms described in Section 2.1.

## 3 Dymola Implementation of Thornton's Aero Engine Case Study

In order to demonstrate the advantages of the new approach, the model aero engine design case study reported in [9] and [10] has been reimplemented using Dymola. The first step is to set up the basic layout of the engine [6], using centre-lines for the cylinder and crankshaft. These, along with the text view of the Dymola code to create them, are shown in Figure 1. Their positioning is determined by basic layout parameters. The engine casing is initially neglected, so instead the centre-lines act as “sky-hooks” to hold the cylinder and main journal bearing fixed in space. The cylinder centre-line origin (shown as an L shaped mark) represents the position of the tip of the piston skirt at bottom dead centre. The shaft centre-line origin represents the location of the main journal bearing. The next steps are to instantiate and connect the parameterised, generic components of the engine, as shown in Figure 2. This may be compared with the equivalent CADET schematic in [9]. Additional elements here are the centre-line objects, a propeller, 2 prismatic, 2 revolute and a spherical joint from the MBS library, and a GasForce object applying the pressure/displacement characteristic shown in Figure 3a. Next, the generic component instances must be sized according to the chosen layout parameters. Unlike CADET, all the components have one primary functional parameter (the connecting rod length, for example), against which all the others are non-dimensionalised. This allows easy scaling without changing shape. The default parameter values have been chosen so the generic component looks “sensible”. The lengths of Shaft and RigidPin are not determined by the initial layout, and are arbitrarily chosen as 0.08m and 0.01m. Parametric equations are entered manually into a dialog box associated with each component. Conveniently, they are specified in conventional notation and do not need converting into the CADET's stack-based form. Next, the equivalent of CADET's user-specified interface constraints must be entered, (except the automatically generated force and moment equations, of course). These, for example, constrain mating pin and hole diameters to be equal or ensure that the piston recess is large enough for the ConRod eye. Fourteen component parameters remain independently variable, 12 still possessing the default values specified in their class definitions. A dynamic simulation is now run, initially for the first 0.1s on start-up, to investigate which design constraints are not satisfied. A distinction is made between constraints representing illegal configurations and those representing component failure. In this case, plotting the global “illegal” variable shows a

constant value of 0, meaning that no illegality constraints are being broken, but the plot of the global “failed” variable appears as in Figure 3b. This shows which components have failed, encoded as successive powers of 2. Thus, 1 means that the first component (ConRod) has failed, 2 means the second (Shaft), 4 means the third (Web) and so on. Having discovered that a particular component has failed, the next step is to determine how, by plotting the “failed” variable for that component. This uses a similar, power of 2 encoding to represent the different failure modes. Having found how a component has failed, the next step is to find the degree of overloading, by plotting the failure index of the component for that mode. This is the loading non-dimensionalised such that a value of 1 or greater equates to failure. Figure 3c shows the initial bending failure index for RigidPin. Plainly, the default diameter is far too small as it is being overloaded by a factor of nearly 20. Knowing that the strength is a function of the third power of the diameter, it is decided to increase the diameter by a factor of say 3. Re-running the simulation ensures that the pin is now strong enough in bending, and allows another failure mode to be addressed. Once all constraints are satisfied, the simulation may be extended until the engine is running at full speed, to check for inertia force failures.

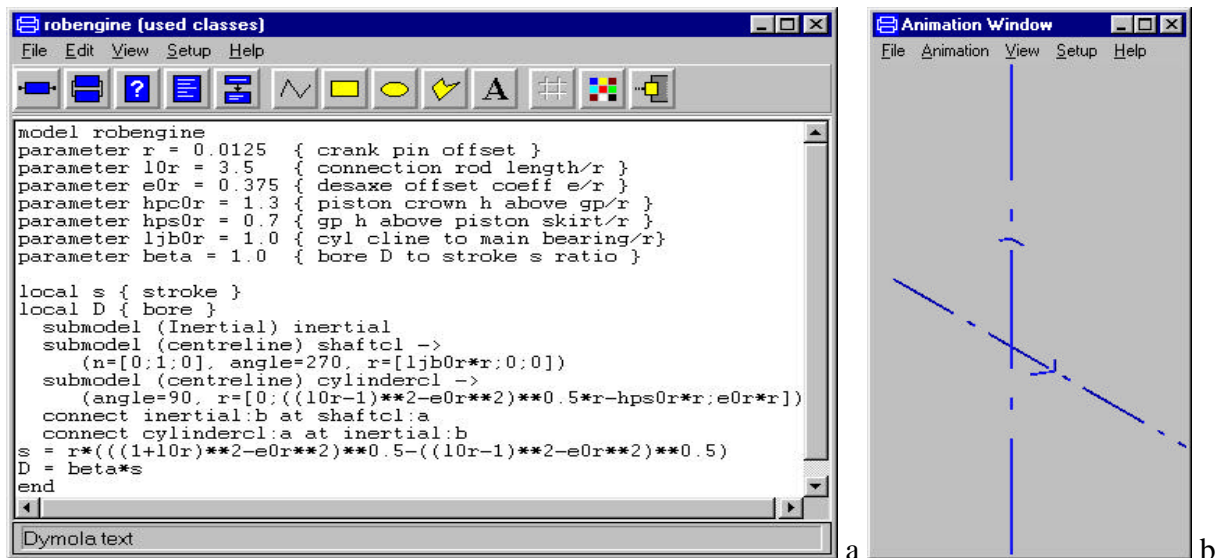


Figure 1. Dymola Text View of Code to set up Initial Cylinder and Crankshaft Centre-lines (b)

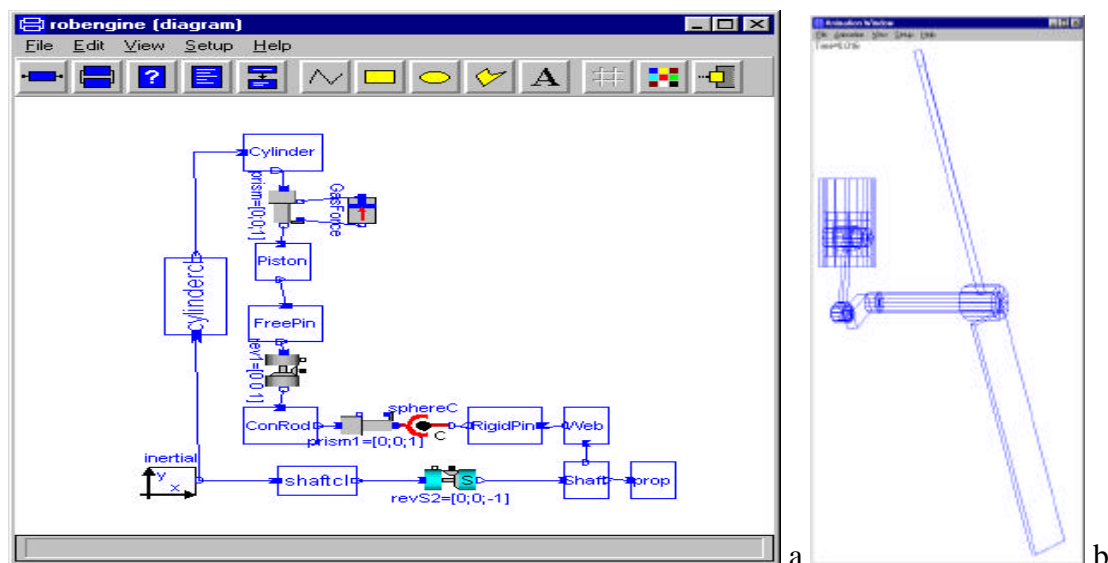


Figure 2. Dymola Schematic Model View and 3D Animation Window for Aero Engine Model

The resulting plot of shaft rotational speed against time is shown in Figure 3d. A 3D animation of the operation of the engine may also be viewed as shown in Figure 2b. The manual approach to constraint satisfaction presented here makes a contrast to the AI search techniques employed in CADET, but seems fast and insightful. In this example, finding a working set of parameter values generally takes less than 10 iterations and an elapsed time of less than 10 minutes. The efficiency of the code generated using the MBS library is such that the simulation of the acceleration of the engine, from start to 12000 rpm, with full design constraint checking, takes just 30s of CPU time on a 450MHz Pentium II PC.

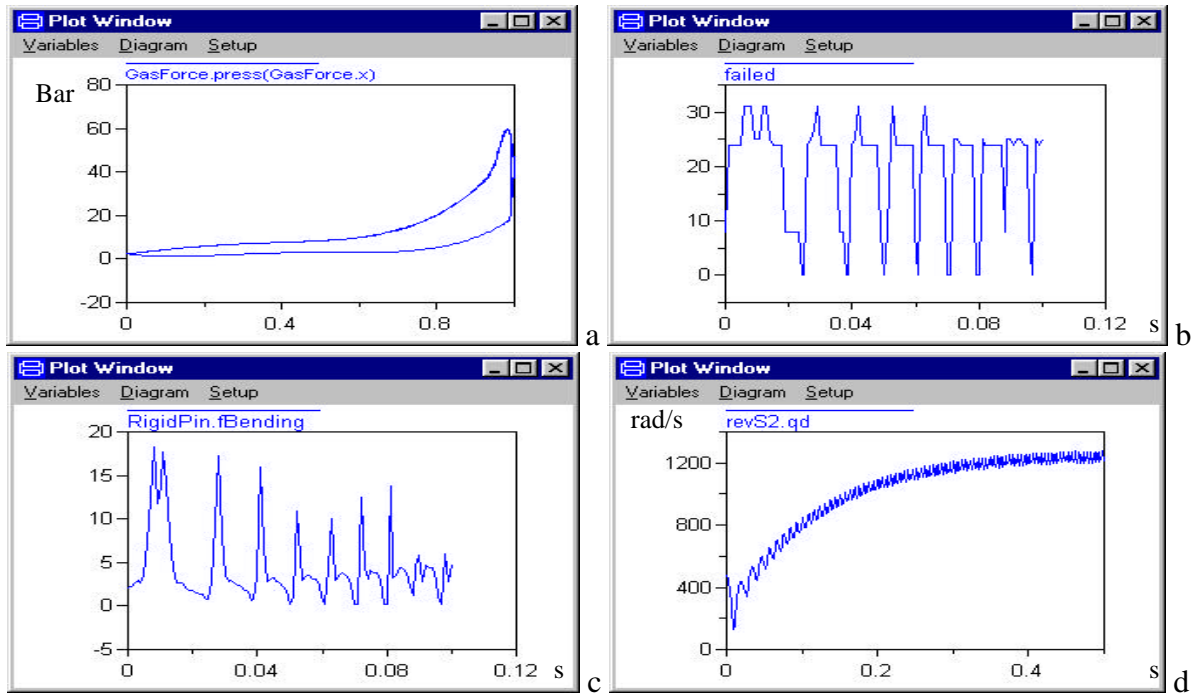


Figure 3. Dymola Simulation Plots for the Aero Engine Design

## 4 Software Environment

Whilst this paper demonstrates the potential for the use of Object-Oriented Physical System Modelling in mechanical design, it is not a universal panacea and has a number of obvious problems as a design tool. For example, with reference to Figure 2, the cylindrical Piston/Cylinder joint has been represented as prismatic, the Piston/FreePin joint has been set as fixed, and the cylindrical ConRod/RigidPin joint has become prismatic coupled with spherical. All these changes are perfectly reasonable *modelling* decisions, resulting from reasoning that there is a planar kinematic loop and that the ConRod is slender and hence relatively flexible. Similarly, only revS2 has a state variable (note the S on its icon), as the spherical cut joint has required their removal from the other 3 kinematic joints. These are considerations that it would be desirable for the designer not to need to think about. The approach being taken in this project is to address these issues by the combination of an underlying Dymola system with an object-oriented/rule based knowledge-based system (KBS) front end, with which the designer interacts. To this end, a flexible and highly integrated software development platform has been designed and assembled, which maximises the use of proven, third party tools. This also supports the Cambridge EDC Functional Synthesis project [4], thus enabling the creation of a tightly integrated design

synthesis and embodiment tool. It utilises the CLIPS KBS development system, Tcl/Tk GUI builder, the Promech kinematics solver [2], and the OpenGL 3D graphics library.

## 5 Conclusion

This paper has shown how the adoption of Object-Oriented Physical Systems Modelling and Simulation tools holds great promise for effective constraint management in mechanical embodiment design. Used in combination with a KBS front-end, this can provide a true virtual prototyping design system with animated dynamic models of the artefacts produced. These are based on rigorous mathematical models, and ideal for subsequent evaluation and optimisation.

### References

- [1] Bracewell, R. H., Sharpe, J. E. E. "Functional descriptions used in computer support for qualitative scheme generation - "Schemebuilder"", AI EDAM Journal - Special Issue: Representing Functionality in Design, volume 10, 1996, pages 333-346.
- [2] Johnson, A. L., "An open architecture approach to kinematic analysis for computer-aided embodiment design", Computer-Aided Design, volume 30, 1998, pages 199-204.
- [3] Kron, G., "Diakoptics - The Piecewise Solution of Large-scale Systems", MacDonald & Co., London, 1963.
- [4] Langdon, P., Chakrabarti, A., "Browsing a large solution space in breadth and depth", *ibid.*
- [5] Mah, R. S. M., "Chemical Process Structures and Information Flows", Butterworths, Boston, 1990
- [6] Organ, A. J., "Case studies in Mechanical Design: The miniature, 2-stroke aero engine", published by the author, Cambridge, UK, 1989.
- [7] Otter, M., Elmqvist, H., Cellier, F. E., "Modeling of Multibody Systems with the Object-Oriented Modeling Language Dymola", Journal of Non-linear Dynamics, volume 9, 1996, pages 91-112.
- [8] Tarjan, R. E., "Depth First Search and Linear Graph Algorithms", SIAM Journal on Computing, volume 1, 1972, pages 146-160.
- [9] Thornton, A. C., Johnson, A. L., "CADET: a software support tool for constraint processes in embodiment design", Research in Engineering Design, volume 8, 1996, pages 1-13.
- [10] Thornton, A. C., "Constraint Specification and Satisfaction in Embodiment Design", PhD Thesis, Cambridge University, 1993.

Rob Bracewell  
Engineering Design Centre, Cambridge University  
Department of Engineering, Trumpington Street  
Cambridge CB2 1PZ, United Kingdom  
Phone: +44-1223-332828, Fax: +44-1223-332662

E-mail: [rhb24@eng.cam.ac.uk](mailto:rhb24@eng.cam.ac.uk)