AIAA 99-0110
Demonstration of CLIPS as an Intelligent Front-End for POST

I. A. Budianto
J. R. Olds
N. C. Baker
Georgia Institute of Technology
Atlanta, GA

37th Aerospace Sciences Meeting and Exhibit

Jan. 11-14, 1999 / Reno, NV

Demonstration of CLIPS as an Intelligent Front-End for POST

Irene A. Budianto[†]
Dr. John R. Olds[*]
Dr. Nelson C. Baker[‡]
Georgia Institute of Technology
Atlanta, GA 30332

ABSTRACT

Most of the analysis codes used in the design of aerospace systems are complex, requiring some expertise to set up and execute. POST, which is used in many conceptual design studies to compute space vehicle performance characteristics, often encounters numerical difficulties in solving the defined trajectory problem. Usually POST fails to converge when its control variables are given a bad set of initial guesses, causing the trajectory to remain in the infeasible design region throughout the computations. The user then analyzes the output produced and relies on a set of heuristics, typically gained from experience with the program, to determine the appropriate modification to the problem setup that will guide POST in finding a feasible region and eventually converge to a solution.

The potential benefits of employing knowledge-based system within a design environment have long been well known. Various methods of utilization have been identified. As a post-processing guide, an expert system can distill information obtained from an analysis code, such as POST, into knowledge. The system then can emulate the human analyst's decision-making capability based on this collected knowledge.

This paper describes the implementation of POST expertise in a knowledge-based system called CLIPS and demonstrates the feasibility of utilizing this integrated system as a design tool. The automation of POST executions and CLIPS' output evaluation and decision-making is shown to potentially reduce design cycle time. In addition, verification of the decision process and of the quality of the results is easily attained in CLIPS.

NOMENCLATURE

CLIPS	C Language Integrated Production System
CPU	Central Processing Unit
ctha	optimization indicator from POST
HTHL	Horizontal Take-off Horizontal Landing
KBS	Knowledge-Based System
LH2	Liquid Hydrogen
LOX	Liquid Oxygen
P2	weighted constraint error from POST
POST	Program to Optimize Simulated Trajectories
q	dynamic pressure (psf)
RBCC	Rocket-Based Combined Cycle
SSDL	Space Systems Design Laboratory
SSTO	Single-Stage-to-Orbit
wopt	weighting factor for optimization variable
α	angle of attack

INTRODUCTION

Background

Many current research works in the field of aerospace are geared towards improving the design phase of these complex systems [1]. One of the

---

[†] *Palace Knight Fellow, School of Aerospace Engineering, Student member AIAA*

[*] *Assistant Professor, School of Aerospace Engineering, Senior member AIAA*

[‡] *Associate Professor, School of Civil & Environmental Engineering*

major thrust areas is to reduce design cycle times and costs. This can either shorten the system's overall development time or allow for more trade-off studies to be conducted, resulting in a more extensive selection of alternative optimum designs.

Utilization of artificial intelligence within the design environment has received much attention in the past. In particular, knowledge-based systems (KBS) have been shown to be well suited for various design applications [2, 3, 4]. For example, an expert system can serve as a post-processing guide. It can be encoded to analyze the results of an analysis code and, based on the information obtained and the set of knowledge imbedded in the system, determine the appropriate next step without the need for human intervention. This automation of information processing and decision-making promises to shorten design cycle times.

Program to Optimize Simulated Trajectories, or POST [5], plays an important role in many space vehicle conceptual design studies. POST, created by NASA and Lockheed-Martin, is widely available and has become an industry standard tool for solving various trajectory optimization problems, including constant dynamic pressure trajectories for airbreathing vehicles. As a design analysis tool, POST provides the performance characteristics necessary to determine the feasibility and viability of a vehicle concept.

POST is a generalized event-oriented trajectory simulation program capable of optimizing a user-specified objective function, subject to certain conditions and constraints, by determining the appropriate values of the control variables. Within a namelist input file, the user structures the trajectory by a logical sequence of events and defines the model of the vehicle as well as the necessary constraints and conditions. The user is further required to specify the parameters and control variables. POST generates a text output file that summarizes the results of the simulation.

Motivation

Analyzing simulated trajectories computed by POST requires knowledge about flight mechanics and vehicle performance. Producing a numerical solution with POST, on the other hand, also demands the type of expertise that comes with practice. As with other numerical simulation programs, POST at times encounters difficulties in solving the defined problem. If the control variables are given a bad set of initial values, many times the program fails to find a feasible region.

Expert users of POST have developed a variety of heuristics to aid the program in overcoming these numerical difficulties, so as to converge to a solution. For example, when POST fails to concurrently satisfy the constraints and converge to an optimum solution with a given set of initial control variable values, one common practice is to first remove optimization from the problem, requiring for POST to simply target the specified conditions and satisfy the given constraints. Then usually the analyst would remove some of the "less crucial" constraints from the problem. Once a feasible region of the design space for this simplified problem is found, the new control variable values obtained are used as initial values for solving the original problem. Thus expert POST users essentially must learn various tricks to help the program explore the design space in search of the set of control variable values that will result in a trajectory that satisfies the optimum criteria and meets all the constraints and conditions.

Implementation of this type of expertise in a knowledge-based system, such as CLIPS (C Language Integrated Production System [6]), that interacts with POST can prove to be very valuable, particularly in a conceptual design environment. The integrated system promises a "plug-and-play" POST capability, which, to a certain extent, can free the expert analyst from the task of "babysitting" the simulations. Consequently, this expert POST has the potential to greatly reduce design cycle time and cost, especially as the whole design process is becoming more automated. The ability to verify the decision process and the quality of the solution must not be compromised, however. This is accomplished partly by CLIPS' explanation facility that supplies the user with an account of all steps that led to the final solution. The user must, in addition, review the final trajectory solution.

Problem Formulation

The main objective for this work is to create a prototype of the CLIPS-POST integrated system and

demonstrate its feasibility for solving an airbreathing ascent trajectory, which involves a constant dynamic pressure phase. This type of trajectory simulation and analysis is very complex due to the fact that the thrust produced by airbreathing engines varies with altitude, Mach number and dynamic pressure and that the vehicle aerodynamic characteristics are continuously changing [7]. POST often has difficulties solving this type of problem.

The expert system's main duty is to assist POST in finding the control variable values that will result in an optimum, without requiring programming modifications to POST itself. It is assumed that the user has created an error-free namelist input file (POST version 5.1) and has successfully obtained a usable nominal (iteration 1) trajectory. A long list of experiential knowledge exists to help users with this nontrivial step but is not implemented for the purpose of this demonstration work. In the future, however, the development of an expert system that will work with a generic POST input file is envisioned.

PROTOTYPE SYSTEM

System Overview

As shown in Figure 1, POST input and output files form the natural interface to CLIPS. This expert POST has been designed such that the parsing of the files as well as the system calls to execute POST are directly performed by CLIPS. Thus, when the integrated system is implemented in a space vehicle design environment, CLIPS acts as the front-end to POST. The end user of this integrated system is prompted for the input file name of the problem to be solved, herein referred to as the original problem setup. This is the extent of the user interaction required by the prototype system.

The iterative process begins with the first execution of POST and parsing of the resulting output file to obtain a set of information needed by CLIPS. This includes the problem setup (e.g. constraints, conditions, optimization flags, tolerances, control variables) and the numerical results (e.g. constraint error, optimization indicator value, objective value history, final values of the control variables, warning and error messages). Copies of the original files are created to keep them free from any modifications made by CLIPS. The inference engine of CLIPS then emulates the processing strategy of an expert POST user.

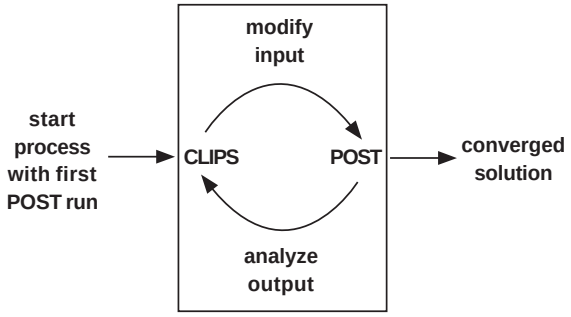

Figure 1. Diagram to illustrate the process of the CLIPS-POST integrated system.

When POST fails to converge, the user would first identify the "symptoms." For example, what error messages are produced by POST? Is the iteration limit exceeded? How do the objective function and the dependent variable values behave over POST iterations? Then according to the information gathered and the user's expertise, he/she makes the appropriate changes to the problem setup and reruns POST with the modified input file in an effort to aid POST in finding a better set of control variable values. If POST still fails to converge, return to the "symptom identification" step by analyzing the new output. Else, retain the control variable values from the last POST run and use them with the original input deck.

Similarly, CLIPS' inference engine must match the set of gathered data (facts) with the knowledge already imbedded in the system. The strategy used for prioritizing these activated (matched) rules is set to simplicity. CLIPS makes the appropriate modifications to the input deck, reruns it in POST and studies the output. The iterative process continues until a successful POST execution is obtained.

In the meantime, an account of all the modifications is kept in memory during each session and is referred to by CLIPS at every step. Each time POST converges, the resulting control variable values are retained. The system backtracks through the cumulated records in the database of past modifications and runs the previous problem setup with these control values. This sequential process, as

has been proven many times manually in actual practice, will eventually lead to a convergence of the original problem.

Process Knowledge Control Structure

The task of the expert POST system can be hierarchically ordered into several sub-problems, as is graphically represented in Figure 2. The problem is considered solved when the POST input and output files are parsed, optimum criterion is met and the constraints and targets are satisfied. Similarly, optimality of the solution requires nonsingular sensitivity matrix (the Jacobian matrix consisting of the partials of each of the constraints with respect to the control variables), a converging objective value history and an improving weighted-error constraint history.

The typical constraints and targeting conditions that are involved in the ascent trajectory simulation of an airbreathing vehicle are maximum wing normal forces, the constant dynamic pressure segment, final altitude and final flight path angle. Each of these constraints forms a sub-problem for the "Constraints Met" node.

Following the strategy used by an expert POST user in examining a POST output, the backward-chaining tree illustrated in Figure 2 is utilized to structure the knowledge process of the CLIPS-POST system. Each node in this network is a hypothesis that can be considered completed only if all its child nodes have been completed (proven). According to the hypotheses that remain unproven, CLIPS can make an intelligent decision on the appropriate modifications to the POST input deck.

The mechanism for keeping a history of input file modifications takes advantage of the dynamic linking capability of a sequence method. The chain of sequence nodes (linked list) is built up as new modifications are made. Thus, each time POST fails to converge to a solution, a new input setup is created and a new node, representing this modified setup, is added at the end of the linked list.

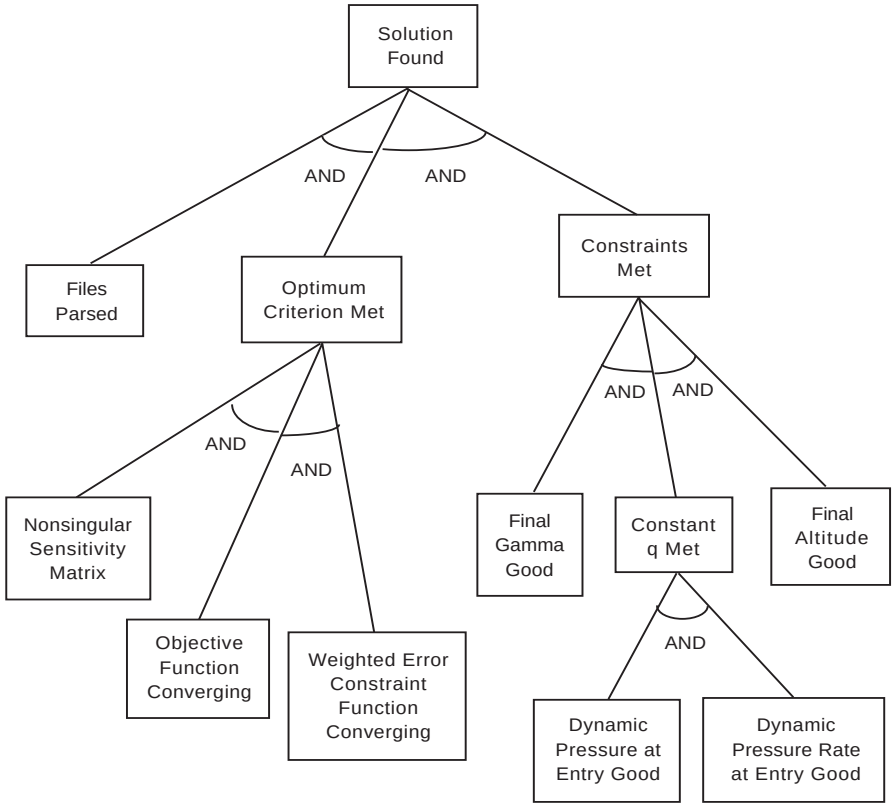

Figure 2. Diagram illustrating the backward-chaining tree structure for the knowledge process control of the CLIPS-POST system.

Table 1. A sample of the domain knowledge collection

RULE NO.	RULE CONDITIONS	CONCLUSIONS / ACTIONS
1	POST optimum criterion, *ctha*, is satisfied; the weighted error constraint, *P2*, indicates that all constraints are satisfied.	Problem is solved and solution is found.
2	The problem involves a constant dynamic pressure segment and has not already been simplified; POST fails to target the conditions and meet the constraints; POST cannot improve *P2*.	Change to target-only mode and eliminate constraints that apply beyond the constant dynamic pressure phase.
3	The dynamic pressure experienced by the vehicle as it enters constant *q* segment is too low (high).	Increase (decrease) the first few angles of the pitch control table applied during the phase before entering the constant *q* phase.
4	The altitude at the end of the trajectory is too low (high).	Increase (decrease) the second and third angles of the pitch control table applied at the end of the trajectory.
37	POST encounters a singular constraint sensitivity matrix.	Reduce take-off angle of attack.
38	POST encounters no change in state.	Check *wopt* and change if necessary.
39	All constraints are satisfied; *ctha* indicates POST has not converged; objective value is not changing much.	Problem is solved and solution is found.
40	Convergence is reached with the original POST input setup; big jumps are observed for the values of optimum criterion, *ctha*.	Re-run the POST input file with the latest control variable values.

Each sequence node is assigned the backward-chaining process knowledge, explained previously. This node is considered complete only when the top goal, "Solution Found," within it is achieved (i.e., when POST successfully converges with this particular input file).

Each time a sequence node is completed (i.e., POST converges and solution is found), the system traverses the linked list backward and pops off the node containing information on the previous problem setup. From this, CLIPS recreates the POST input setup and runs it with the control variable values obtained from the latest converged POST run. This process of modifying and recreating POST input files continues until the original problem formulation is reached and converged, completing the expert system's task.

Context Organization

The context, or data structure, for this expert POST system must be organized such that it works well with the process knowledge. Object templates were created to handle the types of information needed by the system. For example, a template called "POST-input-setup" stores important information about the problem setup. It contains attributes such as *ndepvr* (the number of dependent variables involved in the problem), *nindvr* (the number of control variables), *dt* (integration step), etc. Each object also contains an attribute called *input_ID*, which identifies it as an object belonging to a particular sequence node. In most cases, this attribute can serve as the primary key (i.e., it is sufficient to uniquely identify the object). The variable and parameter names used by POST (e.g. *ndepvr*, *P2*, *ctha*, etc.) are retained in the CLIPS

context. Furthermore, every effort is made to separate input and derived (output) data.

Domain Knowledge

The collection of domain knowledge to be used in the CLIPS-POST system can be categorized according to the sub-problem it addresses. For example, the rules needed for reading and writing POST input files are contained within "Files Parsed" node and can be activated only when this goal is active. On the other hand, the rules that check on the behavior of the objective function are sectioned within "Objective Function Converging" node.

The main sources for this domain knowledge are the authors' own experiences. Additionally, several colleagues in the field who have extensive experience with solving airbreathing ascent trajectories with POST were also interviewed for their expertise. A sampling of the captured heuristics is listed in Table 1. All of the knowledge can be represented in the form of **IF (**condition**) THEN (**conclusion/action**).** If a certain combination of facts exist, then a specified course of action takes place.

Computing Environment

Both CLIPS and POST are compiled on a Silicon Graphics Octane with operating system IRIX 6.4. This workstation runs on a R10000/250 MHz IP30 processor. Installing the two programs on the same machine eases automation of the CLIPS-POST interactions. Thus the user, having setup the problem in POST, can run the expert POST system simply by running the CLIPS program and supplying the input file name at the prompt.

PROTOTYPE EVALUATION

The CLIPS-POST prototype was tested on several airbreathing vehicles. The parameters for these reference vehicles, as summarized in Table 2, are fixed along with the engine and aerodynamic models. The results vary in terms of the number of required iterations, computer processing time and the computed optimal value. The computing requirements depend on the complexity of the problem (i.e., the number of variables and events involved) and the quality of the initial values for the control variables. Furthermore, this type of trajectory optimization is nonlinear and, if not constrained properly, is not unimodal. Thus, depending on the starting point used, the solution found may only be a relative (local) extremum.

Table 2. Summary of test vehicle parameters

Vehicle Parameters	*Hyperion*	*Argus*	*SSTO1*
Gross Wt (lbs)	824,500	653,800	1,000,000
Initial T/W	0.7	0.7	1.1
Rocket Isp_{vac} (s)	462	462	460
Sref (ft^2)	6300	3004	4500

Description of Hyperion

Figure 3 shows a rendering of *Hyperion* as it orbits the earth. This advanced single-stage-to-orbit (SSTO) fully reusable launch vehicle is the product of investigation by graduate students of the Space Systems Design Laboratory (SSDL) at Georgia Institute of Technology. The mission for this un-piloted HTHL (horizontal take-off horizontal landing) concept is to deliver 20 klb of payload to Space Station orbit (220 nmi circular orbit at 51.6° inclination) from Kennedy Space Center.

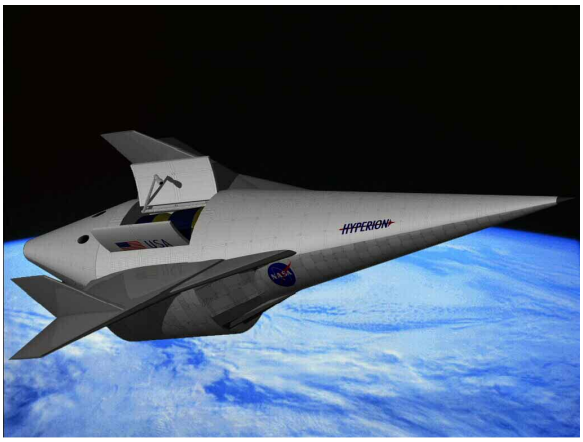

Figure 3. Artist's rendering of *Hyperion*

Hyperion utilizes five LOX/LH2 ejector scramjet RBCC engines as its primary propulsion. The vehicle operates in ejector mode up to Mach 2, where transition to ramjet mode begins. At Mach 3 it

intercepts a 2000 psf dynamic pressure boundary and the vehicle airbreathes (ramjet/scramjet mode) up to Mach 10 while maintaining this *q* limit. The switch to rocket mode occurs at Mach 10.

The trajectory optimization problem involves maximizing the vehicle burnout weight at the end of the simulation. The POST input setup begins at take-off and ends as the vehicle reaches the 50 nmi x 220 nmi insertion orbit. The independent (control) variables are relative pitch angles to control vehicle attitude before and after the constant *q* phase.

Result for Hyperion

Several CLIPS-POST runs for *Hyperion* were conducted with various combinations of initial control variable values. All successfully converged but only two are presented in this paper. A summary of these computations is given in Table 3.

"Test 1" was one case where with the original input setup, the changes in the control parameters made by POST produced no change in the state parameters. CLIPS modified *wopt* (Rule 39 in Table 1) and re-ran POST, which yielded a successful convergence. To juxtapose the CPU requirements of the integrated system and those of individual POST runs, a timing device was setup. The execution of the three POST files themselves, in this case, took a total runtime of 263.14 seconds. This difference of less than 82 seconds was the time required for CLIPS to set up the problems, to run POST, to parse the files, to decide on the appropriate modification and, finally, to recognize that a solution had been found.

For "Test 2," POST was given a set of initial guesses for the control variables, with which the program fails to converge initially (the *P2* value obtained is very large). CLIPS' inference engine applied several rules that eventually guided POST to convergence. It first turned POST to target-only mode and reduced the number of constraints by removing from the problem those that go into effect after the constant dynamic pressure phase (Rule 2 in Table 1). This allowed POST to concentrate on achieving and maintaining the constant *q* path. When POST still failed to reach convergence with this simplified problem setup, CLIPS changed some of the pitch control angles to help the trajectory find the dynamic pressure boundary (Rule 3).

Once POST converged this simplified problem, the resulting control values were used as initial points for the original problem. This process was repeated and terminated because there was no improvement on the objective value as observed by CLIPS. Here, the inference engine applied Rule 39: Although the final *ctha* value does not meet the optimum criteria of being greater than or equal to 89.9°, if the objective value improves very little, then consider the problem solved.

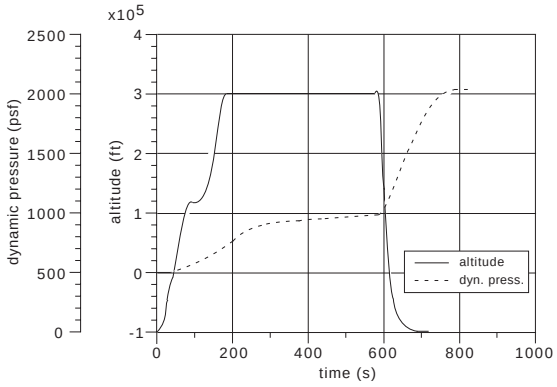

Figure 4. Dynamic pressure and altitude profiles for optimized *Hyperion* trajectory (Test 1).

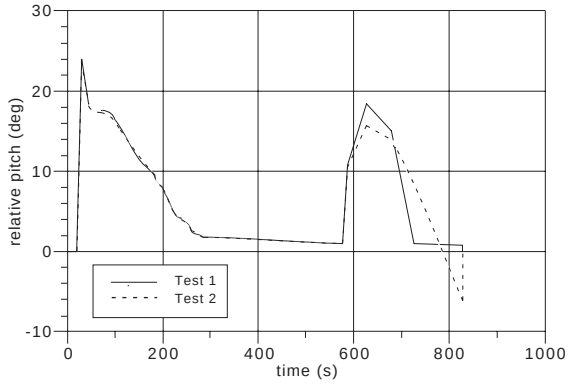

Figure 5. Relative pitch angle histories for optimized *Hyperion* trajectory.

The discrepancy between the optimum burnout weights computed from the two *Hyperion* test cases is less than 0.5%. Figure 4 plots the calculated dynamic pressure and altitude profiles from "Test 1." "Test 2" produced similar trajectory plots. The resulting final pitch control angles differ slightly but have the same general trend, as shown by Figure 5.

Description of Argus

Under NASA's Highly Reusable Space Transportation (HRST) program, a study was conducted at Georgia Tech's SSDL on the highly reusable SSTO launch vehicle concept called *Argus* [8]. Figure 6 depicts an illustration by Pat Rawlings featured in reference 9. *Argus* uses a magnetically-levitated sled and track system, the *Maglifter*, to provide take-off velocity of 800 fps. The main propulsion system consists of two RBCC engines (LOX/LH2 supercharged ejector ramjet). The engines operate similarly to those of *Hyperion*, in that they progressively move from supercharged ejector to fan-ramjet to ramjet modes. One difference is that *Argus* does not have scramjet capability and thus, it airbreathes only until Mach 6. At this point, the vehicle performs a pull-up maneuver and transitions to rocket mode. It continues its ascent to a 50nmi x 100 nmi insertion orbit at 28.5° inclination.

Figure 6. *Argus* by Pat Rawlings

The trajectory simulation problem for *Argus* is also to maximize the vehicle burnout weight. The mission is to deliver 20 klb of cargo autonomously to low earth orbit. The constant dynamic pressure trajectory flown is at 1500 psf.

Result for Argus

In addition to varying the initial control variable setting, *Argus* was also tested several different take-off attack angles (α's). In "Test 1," the vehicle left the *Maglifter* with 7.5° angle of attack. As shown in Table 3, the initial control values given allowed POST to obtain successful convergence unaided by CLIPS. The CPU requirement for the one POST run was 118.8 seconds. In cases such as this, CLIPS simply matched the conditions for Rule 1 listed in Table 1 and terminated the process.

For "Test 2," *Argus* initially was given a take-off attack angle of 15° and initial control values that produce a flight profile in the infeasible region. CLIPS simplified the setup by constraint elimination (Rule 2), which resulted in singularity of the sensitivity matrix of one of the dependent variables. This is typically due to lack of controls to affect the wing normal force constraint violated during take-off. CLIPS remedied this problem by reducing the take-off α (Rule 37). The CLIPS-POST system obtained the final converged trajectory by a combination of lowering take-off angle and simplifying the problem setup. The CLIPS-POST system terminated when it converged the original problem with a reduced take-off α of 10.935°, after a total of 13 POST runs.

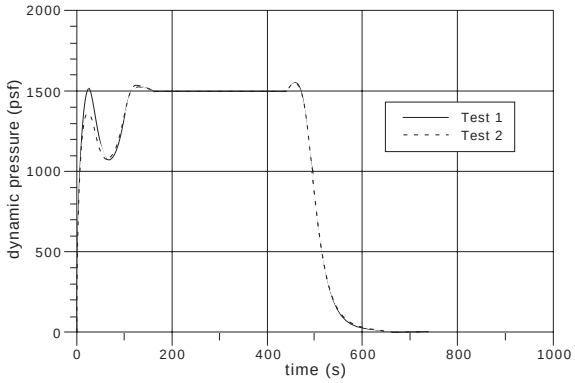

Figure 7. Dynamic pressure profiles for optimized *Argus* trajectories

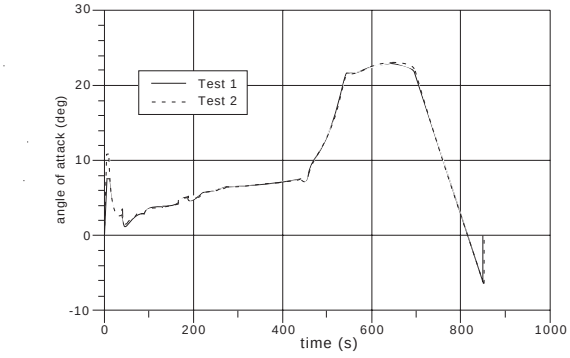

Figure 8. Angle of attack histories for optimized *Argus* trajectories

The resulting dynamic pressure profiles for the two test cases are shown in Figure 7. The lower take-off angle of attack of "Test 1" caused a higher jump in dynamic pressure experienced before the constant *q* segment. Figure 8 illustrates that the angle of attack profiles computed by POST for the two test cases are very similar. The biggest discrepancy was found in the beginning of the trajectories, where the take-off values differed by 3.435°.

The integrated CLIPS-POST system achieved successful convergence for all tested cases (given a usable nominal trajectory). The discrepancies between all of the computed optimal values were on the order of less than 0.5%.

Description of SSTO1

The integrated CLIPS-POST system was also tested with a generic SSTO sample POST input deck, referred herein as *SSTO1*. This winged-body vehicle is powered with pure rockets, eliminating the need for a constant *q* segment during its ascent flight. Again, the trajectory problem is to maximize the final vehicle weight.

The trajectory simulation begins with a vertical take off of the single stage vehicle and ends when it reaches the 50 nmi x 100 nmi x 28.5° insertion orbit. The control (independent) parameters given to POST are two inertial pitch rates and the time at which transition between these rates occur. The orbital destination is specified by providing POST with the desired final altitude and inertial velocity. The parameters for this SSTO are given in Table 2.

Result for SSTO1

Solving the trajectory problem for this vehicle required less computing time than *Hyperion* or *Argus*. This is as expected, since the setup for *SSTO1* is much simpler, involving fewer control variables and fewer constraints.

For "Test 1" of *SSTO1*, POST was given good initial guesses for its control variables, allowing it to reach convergence with the original setup. However, CLIPS applied Rule 40 and re-ran the problem without seeing any more improvements.

With the original input setup in "Test 2a," POST was able to satisfy the constraints but not the optimization criteria, because no change in state was observed by POST. The CPU time for this first POST run is 3330.0 seconds. CLIPS then changed the weighting factor, *wopt*, for the optimization variable (Rule 38) and terminated the process when it could not further improve the objective value. The total CPU time was 3764.1 seconds.

Table 3. Summary of test results.

	Hyperion		*Argus*		*SSTO1*	
	Test 1	Test 2	Test 1	Test 2	Test 1	Test 2b
Total CPU Time (s)	348.3	3026.6	152.9	1101.9	88.0	515.4
No. of Modifications	1	6	0	6	1	1
Total No. of POST Runs	3	13	1	13	3	3
Final Constraint Error, *P2*						
Initial POST Run	1.525E-01	6.915E+17	1.564E-01	8.391E+04	3.740E-01	3.785E-02
Final POST Run	3.721E-01	1.200E-02	---	7.717E-01	3.740E-01	3.785E-02
Final Opt. Indicator, *ctha*						
Initial POST Run	89.83°	---	89.90°	---	89.9°	---
Final POST Run	90.0°	88.66°	---	89.97°	90.0°	90.0°
Final Opt. Value, $W_{burnout}$ (lbs)						
Initial POST Run	163,562.06	97,481.05	114,796.54	113,243.00	122,715.44	122,382.57
Final POST Run	163,596.17	163,993.92	---	114,878.83	122,715.44	122,382.57

"Test 2a" produced a final burnout weight of 98,494.34 lbs. This gives a very significant difference of almost 20% from that of "Test 1." This can be explained by the fact that the problem is not properly constrained.

A re-testing ("Test 2b") was conducted by adding another constraint to the *SSTO1* problem, namely final orbital inclination of 28.5°. The initial control values of "Test 2a" was used. The results of "Test 1" and "Test 2b" are presented in Table 3. As shown, a difference in optimum value of less than 0.5% was obtained.

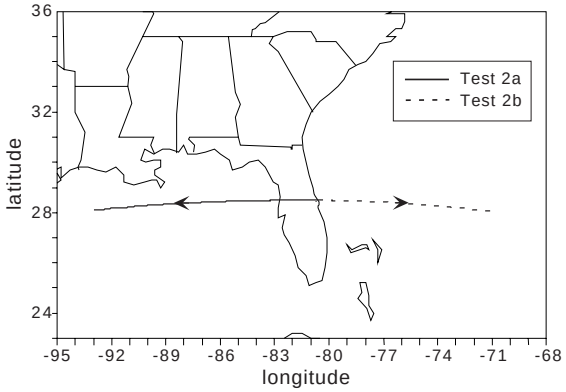

Figure 9. Flight profiles for optimized *SSTO1* trajectories

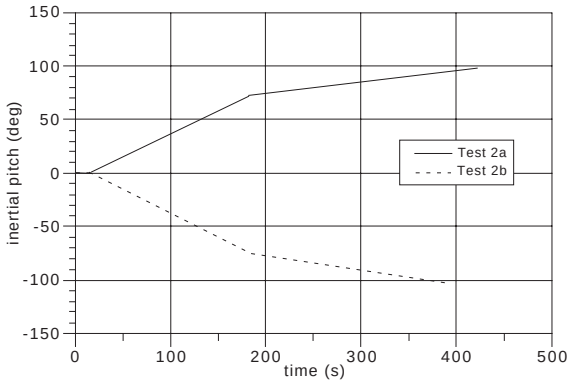

Figure 10. Inertial pitch angle histories for optimized *SSTO1* trajectories

Figure 9 illustrates the vehicle's ground tracks for "Test 2a" and "Test 2b." As shown, the final orbits are different for these two cases, where one is inclined at 28.5° while the other is at 151.5° (a retrograde orbit). The constraints involving the final altitude and velocity were satisfied in both trajectories. Figure 10 compares the inertial pitch angles of the two vehicles (the final inertial roll and yaw angles are 0° for both cases).

CONCLUSION

Summary

The techniques used by expert POST users are not procedural in nature. Much of the rules are heuristics and require symbolic reasoning. Thus this problem is not algorithmic and the implementation of the expertise in a knowledge-based environment is an appropriate choice, as proven by the prototype developed in this research work. Successful convergence was obtained for all tested cases, demonstrating the robustness of the integrated system. The prototype's versatility to work with different types of trajectory problem has also been confirmed with the results for *SSTO1*.

Furthermore, this expert POST system can prove to be a very valuable tool to the trajectory analyst of a space vehicle conceptual design team. The time and cost savings per design cycle will easily outweigh the overhead (more CPU time and memory requirements) of CLIPS integration. Especially, when the vehicle design undergoes major changes, it is often the case that POST will have difficulties converging to a solution. This CLIPS-POST system provides the capability to automatically produce a solution without consuming the analyst's time. In addition, the CLIPS interface can be used by an engineer who is familiar with the capabilities of POST but who is not necessarily an expert user, either as a means of training or simply as a front-end to POST. Note that all interactions with CLIPS were by means of the input and output files, thus requiring no modification to the original programming of POST.

Future Considerations

There are still many additions that can improve the performance of the current system. The prototype developed here is fairly robust in its present state. The executions of POST are done directly from within CLIPS using the `system` command. This delivers the call to execute POST to the operating system and suspends CLIPS processes as it waits for

POST to finish the run. This task can perhaps be done in a more sophisticated manner, by taking advantage of process controls, such as `fork`, provided by the system. This can give CLIPS the capability to check the status of a POST run so that a decision can be made whether to continue or terminate the process (i.e., if no improvement is made with the current run), ultimately to save computing time.

The set of knowledge already imbedded in this CLIPS-POST prototype is not at all comprehensive. The rules relied on by expert POST users in generating a usable nominal trajectory have not been implemented. Also, the heuristics for analyzing the resulting trajectory, if encoded into CLIPS, may allow for automatic checking of the solution. Different types of trajectory problems utilize different sets of heuristics. The current system contains approximately 90% of existing knowledge for converging airbreathing vehicle trajectories. But only ~10% of existing rules for solving direct ascent of a pure rocket vehicle, for example, has been encoded in the prototype developed here. Perhaps the test case for *SSTO1* can be improved further. Other types of trajectory problems are orbital maneuvers and planet reentry.

A potential utilization for an integrated CLIPS-POST system not explored in this research work is for finding the global optimum, or at least for finding better optimal value. The results for *SSTO1* especially can demonstrate that often many relative extrema exist, if the problem has not been properly constrained. The expert system can be encoded to explore a variety of design points, for example by grid search method, keeping the trajectory that produces the best objective function value.

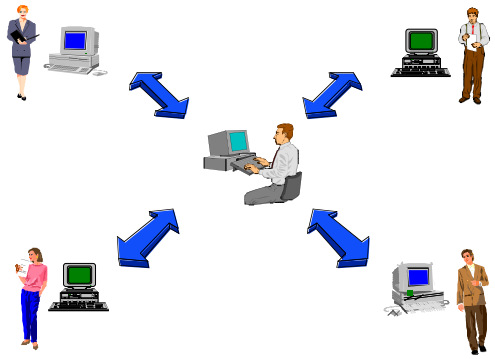

Figure 11. Integrated design framework

Finally, the integration of the CLIPS-POST system within an automated design framework is left for future work. Using the CLIPS front-end to POST within a computing architecture described in reference 10 (either using web-based or UNIX interfaces) can be very beneficial, investing relatively the same amount of effort, if not less, as compared to simply integrating POST to the framework. Figure 11 graphically depicts an example of this integrated design framework where automatic execution of several analysis codes can be performed by one user, typically a system analyst. Disciplinary experts are involved off-line to guide the solution and can be geographically located at multiple sites. Furthermore, the analysis codes can be mounted on different types of computer platforms.

ACKNOWLEDGEMENTS

The authors would like to thank Laura Ledsinger and Peter Bellini for sharing their POST expertise. John Bradford supplied vehicle information for *Hyperion* and *Argus*.

REFERENCES

1. AIAA Technical Committee for MDO, "Current State of the Art of Multidisciplinary Design Optimization," AIAA White Paper, approved by AIAA Technical Activities Committee, Washington, D.C., September 1991.

2. Thurston, D. L. and Carnahan, J. V., "Intelligent Evaluation of Designs for Manufacturing Cost," Chapter 17, *Concurrent Engineering: Automation, Tools, and Techniques*, edited by Kusiak, A., John Wiley & Sons, New York, 1993.

3. Marx, W. J., Schrage, D. P., and Mavris, D. N., "An Application of Artificial Intelligence for Computer-Aided Design and Manufacturing," International Conference on Computational Engineering Science, Mauna Lani, HI, July 1995.

4. Wesley, L. P. and Lee, J. D., "Toward an Integrated CFD Expert System Environment,"

AIAA 98-1005, 36th Aerospace Sciences Meeting and Exhibit, Reno, NV, January 1998.

5. Brauer, G. L., Cornick, D. E., and Stevenson, R., *Program to Optimize Simulated Trajectories (POST)*, Final Report for NASA contract NAS1-18147, Martin-Marietta Corp., September 1990.

6. Giarratano, J. C. and Riley, G. D., *Expert Systems: Principles and Programming*, Third Edition, PWS Publishing Company, Boston, 1998.

7. Olds, J. R. and Budianto, I. A., "Constant Dynamic Pressure Trajectory Simulation with POST," AIAA 98-0302, 36th Aerospace Sciences Meeting and Exhibit, Reno, NV, January 1998.

8. Olds, J. R. and Bellini, P. X., "*Argus*, a Highly Reusable SSTO Rocket-Based Combined Cycle Launch Vehicle with Maglifter Launch Assist," AIAA 98-1557, 8th International Space Planes and Hypersonic Systems and Technologies Conference, Norfolk, VA, April 1998.

9. Mankins, J. C., "Lower Costs for Highly Reusable Space Vehicles," *Aerospace America*, March 1998, pp. 36 – 42.

10. Acton, D. E. and Olds, J. R., "Computational Frameworks for Collaborative Multidisciplinary Design of Complex Systems," AIAA 98-4942, 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, September 1998.