

2000

GasTurbnLab: A Problem solving Environment for simulating Gas Turbines

Sanford Fleeter

Elias N. Houstis

Purdue University, enh@cs.purdue.edu

Report Number:

00-005

Fleeter, Sanford and Houstis, Elias N., "GasTurbnLab: A Problem solving Environment for simulating Gas Turbines" (2000).
Computer Science Technical Reports. Paper 1483.
<http://docs.lib.purdue.edu/cstech/1483>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

**GASTURBNLAB: A PROBLEM SOLVING
ENVIRONMENT FOR SIMULATING GAS TURBINES**

**Sanford Fleeter
John R. Rice
Ann C. Catlin
Elias N. Houstis
Chen Zhou**

**CSD TR #00-005
February 2000**

GasTurbnLab: A Problem Solving Environment for Simulating Gas Turbines

Sanford Fleeter*
John Rice†
Ann Catlin‡

Elias Houstis†
Chen Zhou‡

Abstract

GasTurbnLab is a problem solving environment which, someday, could evolve into a complete simulation of a gas turbine engine. The initial PSE involves only the principal power producing components: the compressor, combustor and high pressure turbine. Even this piece stretches the capabilities of current computer and simulation technology. The general design is based on agents which can solve the relevant PDEs or mediate interface conditions between PDE solving agents. This collaborating PDE solvers approach has the potential for (a) allowing the reuse of legacy simulation code, (b) providing parallelization of these codes without a complete rewrite, (c) flexibility in using and load balancing networks of machines. We describe how we make two large legacy Fortran codes (ALE3D and KIVA) collaborate to carry out the simulation. This is done as a parallel computation without rewriting or parallelizing these codes. The GasTurbnLab user interface is based on the geometry of the engine and allows an engineer to carry out a simulation without knowing the details of the PSE software architecture or how the parallelization is achieved.

Key words: gas turbines, problem solving environments, simulation, surge, rotating stall, high cycle fatigue, interface relaxation, software architecture, agent based computing, combustion, unsteady flows.

AMS subject classifications: .

1 Gas Turbines

The GasTurbnLab project is to develop a problem solving environment (PSE) for studying gas turbines. They have about 30,000 parts (1600 of which rotate very rapidly) and extreme operating conditions (10–50,000 rpm, 500–1000°, 20–50 atmospheres pressure). Important physical phenomena takes place on scales from 10 microns to 10 meters. An accurate simulation of an entire engine is beyond the capability of software systems or computers likely to be available in this decade. This project thus focuses on the “power train” of an engine: the compressor, combustor and turbines. The goals of the project are to advance **PSE technology** and understand better the phenomena of **stall, surge and turbine blade failure**. These phenomena involve the unsteady interaction of the fluids, combustion and structures in the main gas flow. These interactions produce stress loadings on blades resulting in **high cycle fatigue failures**. They can also produce stall and surge which almost instantaneously shut down the engine. The hope is to understand the interaction mechanisms and processes well enough to design instrumentation that can detect the approach of failures and allow corrective actions to prevent engine failure before it happens.

*Dept. of Mechanical Engineering, Purdue University, W. Lafayette, IN 47907, U.S.A., fleeter@cs.ecn.purdue.edu

†Dept. of Computer Sciences, Purdue University, W. Lafayette, IN 47907, U.S.A., [{enh, jrr, acc}@cs.purdue.edu"> {enh, jrr, acc}@cs.purdue.edu](mailto)

‡Dept. of Engineering, Purdue University, Calumet, IN 46322, U.S.A., gzhou@calumet.purdue.edu

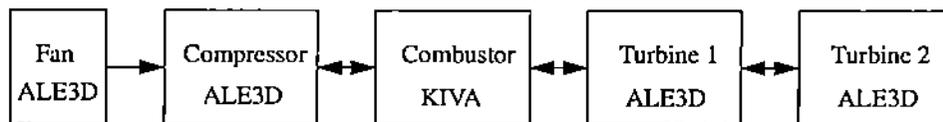


Figure 1: Cross section of an engine in GasTurbnLab. Only the gas flow “power train” of the engine is considered.

The user interface of GasTurbnLab is based on the geometry of the gas flow part of an engine as diagrammed in Figure 1. Conceptually there are five “active” components: fan, compressor, combustor, high pressure turbine, and low pressure turbine separated by “passive” areas of gas mixing. There are mechanical couplings between the fan and low pressure turbine and between the compressor and high pressure turbine, which are not simulated directly; the total force is simply balanced between the component pairs. The software and simulation components also use this geometry modularity, any particular simulation consists of a set of objects (geometry plus simulator) which partition the engine. The simulation of the fan, compressor and turbines use a tailored version ALE3D [2], the simulations of the combustor use a tailored version of KIVA [11] and the simulation of the gas mixing use a CFD solver from PELLPACK [3]. GasTurbnLab has three partial differential equation (PDE) solvers collaborate to find a solution to the overall composite multi-physics PDE problem. Different simulations may involve different subsets of the turbine.

2 The Numerical Simulation

The basis for the simulation consists of three PDE solvers: ALE3D, KIVA and PELLPACK. ALE3D (Arbitrary Lagrangian – Eulerian/3D) is a finite element solver able to handle both fluids and solids simultaneously. This code developed by Lawrence Livermore National Laboratories is able to model the three-dimensional, time dependent coupled fluid-structure interactions of turbo-machinery blade rows that occur in the flow regimes of gas turbines. This is done without phase-lagging errors. The approach uses the Lagrangian perspective for the solid structure and the Eulerian perspective for the fluid; the numerical method is analyzed by Bendiksen [1]. The current version has about 200,000 lines of Fortran code. ALE3D has been tailored for turbines and validated in [2]; this version is used in GasTurbnLab and is sometimes called the Turbo-machinery Aero-Mechanics code (TAM-ALE3D), we refer to this version as ALE3D also here.

KIVA is a widely used code to simulate combustion, especially in diesel engines. It was developed at Los Alamos National Laboratories and currently has about 50,000 lines of code. The GasTurbnLab group has adapted it to use the geometry of a simple gas turbine and will further enhance it to handle more realistic geometries. A current weak point of KIVA is its very simplified model of the fuel spray and flame front; these are modeled on a phenomenological basis, rather than by a direct simulation of the physical and chemical processes. KIVA does simulate the fluid flow adequately and allows for a wide variety of chemical reactions within the combustion chamber.

There are two CFD codes within PELLPACK that can be used within GasTurbnLab: NPARC3-D [4] and ITGFS [7]. They use finite volume techniques to model compressible flows.

These PDE solvers collaborate together to model the gas flows, blade interactions and combustion with a turbine. The method of interface relaxation [5] is quite simple to describe:

- Step 1:** Guess at the solution values, derivatives, etc., on all the interfaces between subdomains of the simulation.
- Step 2:** Solve each PDE exactly on its subdomain using boundary conditions selected from the guesses. There are more interface values available than can be used in solve the PDEs.
- Step 3:** Compare the solution values across each interface and improve them (using a relaxation formula) so as to better satisfy all the interface conditions.
- Iterate:** Steps 2 and 3 until convergence

A variety of relaxation formulas have been proposed in the past 10–12 years and some of them have performed quite well in experiments. This method is extremely difficult to analyze mathematically. The overview by Rice [6] gives reference to the mathematical analysis and experiments with interface relaxation. Figure 2 gives a simplified schematic of the subdomains and solvers used in GasTurbnLab. The interface conditions between the subdomains are the continuity of the gas flow properties (velocities, pressure, mass, ...).

The GasTurbnLab simulations are intractable without parallel computing. Current runs of KIVA or ALE3D on a single current fast processor take several days, or weeks even, for one subdomain, e.g., two blades of a compressor

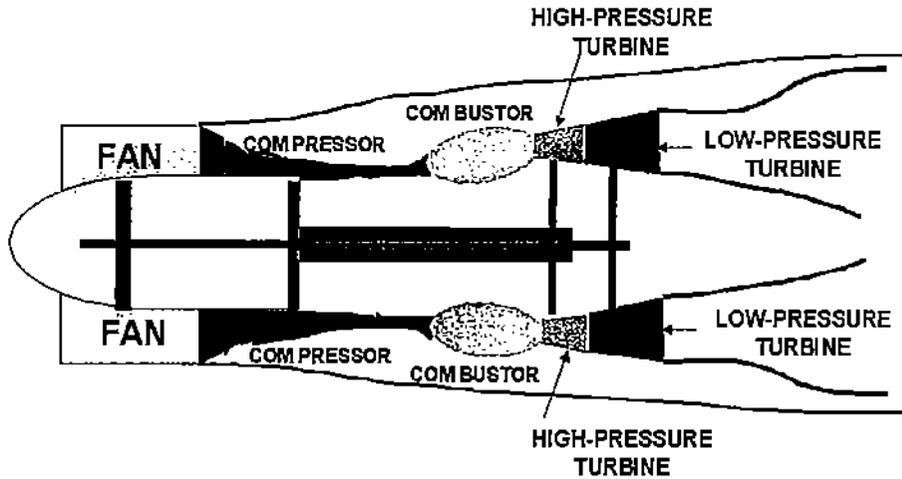


Figure 2: Schematic of the networks of PDE solvers in GasTurbnLab. The PELLPACK solvers may be used to connect these components and the mechanical connection between the Fan and Turbine 2 and between the Compressor and Turbine 2 are not shown.

or a single simplified combustor (real engines may have 5 or 6 of these). The collaborating PDE solvers method used in GasTurbnLab leads naturally to parallelism *without rewriting the PDE codes*. Suppose, for example, one has a compressor fan with 8 rows of blades. If one uses cylindrical symmetry then the 8×17 blades can be simulated with 8 copies of ALE3D, one for each blade row. These can execute on 8 different machines with a relatively low overhead cost for communication. Simulations of unsteady interactions are the principal goal of GasTurbnLab and for these the assumption of cylindrical symmetry does not hold. Thus, with 136 different machines working in parallel, one can execute 136 copies of ALE3D, again with low communication overhead. The software architecture of GasTurbnLab is explicitly designed to accommodate such parallel execution.

The similar approach can be used with the combustors and KIVA. It is straightforward to use 5 copies of KIVA if there are 5 combustors, but this does not provide enough parallelism. However, one can slice a combustor into, say, 10 pieces along planes perpendicular (approximately) to the flow. Such planes already are defined by the mesh KIVA uses in the simulation of a single combustor and a single copy of KIVA can easily use pairs of such planes as its gas inlet and outlet. A simple calculation shows that this approach, applied to ALE3D, KIVA and PELLPACK, allows one to use 500 to 1000 processors in parallel for the simulation of a gas turbine. And this can be done with minimal communication overhead provided the underlying computing facility is able to handle a set of, say, 1000 processors where each communicates with a few others once in awhile. The hardware infrastructure available to GasTurbnLab is much smaller, it consists of computational grid with a IBM SP-2 (28 processors), an SGI Origin (32 processors), an Intel PC cluster running Solaris (128 processors) and a few workstations. This is enough to test the effectiveness of this approach to parallel computation, but not enough for a full engine simulation in parallel.

A deeper analysis of the physics and these codes suggests that another factor of 5 in parallelism or so can be obtained this way. For example, ALE3D actually creates internally 5 submeshes to simulate one blade. One subdomain is inside the blade and the other four surround it. A similar exploitation of the natural geometry appears practical inside a combustor. Thus the collaborating PDE solvers approach might allow as many as 5000 processors (copies of KIVA, ALE3D or PELLPACK) to be executed in parallel.

This gain in parallelism is certainly worthwhile, but it might not be enough. First, much of the gain is by making the problem bigger rather than by making the computation faster. That is, instead of simulating two blades, one is simulating the gas flow through a whole engine. This means that the parallelism in the computation remains coarse grained. Second, this partition of the computation is entirely in the spatial variables and unsteady interactions require relatively long times (dozens to hundreds to thousands of turbine revolutions). Third, the above discussion ignores that the time steps natural for KIVA are about 1/6 to 1/4 those for ALE3D. While we know how to handle this difference now, we do not know whether this difference might grow as more turbine components are simulated, more accuracy is needed, or more realistic chemistry models are used in the combustion.

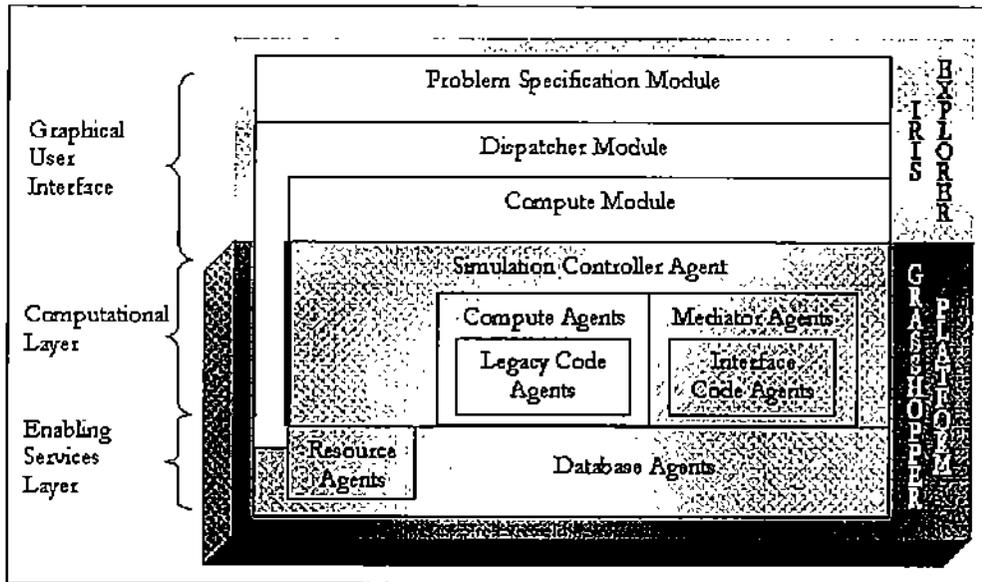


Figure 3: The architecture of the GasTurbnLab PSE. The two main components in the IRIS Explorer and Grasshopper Agent Platform. Within these components there are further layers of the PSE software.

3 The Problem Solving Environment

The PSE interface is based on the turbine geometry. At the top level one sees all or part of Figure 1. At any level one sees the current domain and its subdomains. One can visualize the current solution on this domain (if there is one yet) and related information, e.g., ranges of physical variables. In the future we plan to provide this visualization dynamically during a simulation. At the bottom level one can also examine many aspects of the simulation model, e.g., the mesh, the time step, key software parameters, etc. Similarly, information about the computational environment and performance are available for the components at various levels, e.g., machines used, machine utilization, communication connections and traffic, numerical convergence measures, etc.

The software architecture is one of software agents collaborating to solve the overall composite PDE that models the turbine behavior. Some agents are PDE solvers with one solver from among ALE3D, KIVA or PELLPACK. Others handle the communication between solvers and implement the interface relaxation. This architecture is a natural adoption of the PELLPACK [3] and SciAgents [6] architectures for collaborating PSEs. This architecture is shown in Figure 3. The *problem specification module* contains the user interface components. The *dispatch module* processes the subdomain structure and selects processors from the computational grid for the compute agents which contain the PDE solvers. It has algorithms to optimize performance based on network and processor properties. This module displays its selection to the user who can override the automatic selections. The *compute module* chooses mediator agents based on the interface conditions and connectivity of the problem subdomains. It then launches the simulation computation and controls the execution. Simulation results are connected to the *visualization module* (IRIS Explorer) where they may be viewed, plotted or saved.

The Explorer interface provides user access to all PSE components that interact with the user (problem specification, processor allocation, simulation control, and solution visualization). The Grasshopper [8] distributed agent environment facilitates the agent based simulation paradigm. It is implemented in Java, supports a range of communication protocols and is compliant with OMG standard MASIF for mobile agents.

The legacy Fortran codes (ALE3D, KIVA and PELLPACK) are made into Grasshopper agents by a series of wrappers. The core is a Fortran PDE solver which is wrapped by C code. This is then wrapped by Java code which, in turn, is wrapped to become a Grasshopper agent. A simpler alternation is to take the C code and create a server which can be accessed by client agents of the Grasshopper system. Both approaches are practical and the best choice depends on the nature of the legacy code; if it has a multitude of external interactions, the the client-server approach is better.

A similar PSE architecture is described in a companion paper [10] in the proceedings.

4 Engineering Design

To better understand the transient performance of gas turbine engines, the GasTurbnLab simulation will model the fan, compressor, combustor and turbine. This will capture the compressor-combustor interactions that occur during rotating stall and surge. This simulation includes the effects of three-dimensional, turbulent viscous flows and their interactions with combustion and the engine structure. Note that peak power performance occurs very near to unstable flow, so the move from "normal" operation to stall and then surge is not far. The key question is to identify quantities that indicate a turbine is moving toward an unstable flow regime. The best current experiments show measurable effects predicting unstable flow about 20 revolutions before the instability starts. A 50,000 rpm, that is about 2 or 3 hundredths of a second warning. It is believed that smaller, more localized precursors to unstable flow exist and, if they could be identified, one can visualize sensors that detect these precursors and initiate some corrective action automatically.

Small and very short term excursions into the unsteady flow regime are believed to occur frequently in high performance turbines. These events do not cause the engine to actually stall, but they do produce transients that reduce performance and, more importantly, produce extreme load on the turb machinery blades. These events, in turn, produce high cycle fatigue failure (a blade breaks off) due to blade vibrations.

References

- [1] O.O. Bendiksen, *A new approach to computational aeroelasticity*, AIAA-91-0939, (1991), pp. 00-00.
- [2] D.A. Gottfried and S. Fleeter, *Prediction of unsteady cascade aerodynamics by an arbitrary Lagrangian-Eulerian finite element method*, AIAA-98-3746, (1991), pp. 00.00.
- [3] J.R. Rice, S. Weerawarana, A.C. Catlin, P. Papachiou, K.Y. Wang, and M. Gaitatzes, *PELLPACK: A problem solving environment for PDE based applications on multicomputer platforms*, ACM Trans. Math. Software, 24 (1998), pp. 20-73.
- [4] G.K. Cooper, R.R. Jones, G.D. Power, J.R. Sirbaugh, C.F. Smith, and C.E. Towne, *A user's guide to NPARC, Version 2.0*, NASA Lewis Res. Ctr. and Arnold Engr. Dev. Ctr., (1994).
- [5] J.R. Rice, P. Tsompanopoulou, and E.A. Vavalis, *Interface relaxation for elliptic partial differential equations*, (2000), to appear.
- [6] J.R. Rice, *An agent based architecture for solving partial differential equations*, SIAM News, 31(6), (1998), pp. 1-7.
- [7] S. Zhang, *Molecular-mixing measurements and turbulent-structure visualizations in a round jet with tabs*, Ph.D. Thesis, School of Aero. Astro., Purdue Univ., (1995).
- [8] *The Grasshopper agent platform*, IKV++, <http://www.ikv.de>, GmbH, Kurfurstendamm, (1998).
- [9] S. Markus, E.N. Houstis, A.C. Catlin, J.R. Rice, P. Tsompanopoulou, E.A. Vavalis, D. Gottfried, and K. Su, *An agent based netcentric framework for multidisciplinary problem solving environments (MPSE)*.
- [10] E.N. Houstis, N. Dhanjani, J.R. Rice, and A.C. Catlin, *The WebPDELab server: A problem solving environment for partial differential equation applications*, Proc. 16th IMACS World Congress, (2000) to appear.
- [11] A.A. Amsden, *KIVA-3V, A block-structured KIVA program for engines with vertical or canted valves*, Tech. Rpt. LA-13313-MS, Los Alamos Nat. Lab., (1997).