Preprint
UCRL-JC-139826

Real-Time Control System for Adaptive Resonator

L. Flath, J. An, J. Brase, R. Hurd, M. Kartz, R. Sawvel, D. Silva

This article was submitted to
Society of Photo-Optical Instrumentation Engineers 45th Annual Meeting Exhibition & Education Program, San Diego, CA
July 30 – August 4, 2000

U.S. Department of Energy

Lawrence Livermore National Laboratory

July 24, 2000

Real-time control system for adaptive resonator

Laurence Flath†, Jong An, James Brase, Randall Hurd, Michael Kartz, Robert Sawvel, Dennis Silva
Lawrence Livermore National Laboratory, P.O. Box 808, L-258, Livermore, CA 94550

ABSTRACT

Sustained operation of high average power solid-state lasers currently requires an adaptive resonator to produce the optimal beam quality. We describe the architecture of a real-time adaptive control system for correcting intra-cavity aberrations in a heat capacity laser. Image data collected from a wavefront sensor are processed and used to control phase with a high-spatial-resolution deformable mirror. Our controller takes advantage of recent developments in low-cost, high-performance processor technology. A desktop-based computational engine and object-oriented software architecture replaces the high-cost rack-mount embedded computers of previous systems.

Keywords: Adaptive optics, real-time control systems, unstable resonators

1. INTRODUCTION

As output power levels of solid-state lasers increase, maintaining beam quality becomes increasingly difficult without some form of adaptive wavefront control.[1, 2] The theory of dynamic modification of optical phase is well-studied;[3, 4] however, practical implementation of adaptive optics (AO) has been limited to ground-based astronomical[5, 6] and simple laser configurations.[7, 8] The large expense of AO systems with high spatial resolution (i.e. with the number of active elements in the hundreds and above) remains a major obstacle to wide-spread adoption. Until recently, the cost of the real-time computational hardware required for such systems has been a large fraction of the total system cost. With this adaptive solid-state laser resonator, we are applying our expertise in AO control systems to the new realm of intracavity wavefront sensing/modulation *and* creating portable software that runs the latest in low-cost desktop hardware.

The laser itself is a heat-capacity design for high average power in an unstable resonator configuration.[2] The control system permits continuous operation, pulsed at 20 Hz for several seconds, while maintaining a far-field pattern no greater than three-times the diffraction-limited spot size. Referring to Figure 1, aberrations in the gain medium build with each shot as non-radiative energy from the pumping source is stored (in the form of heat). An intracavity wavefront sensor samples the phase profile. This error information (with respect to the ideal flat wavefront) is then sent to a computer-based control system to determine the appropriate shape to apply on a deformable mirror for the next laser pulse.

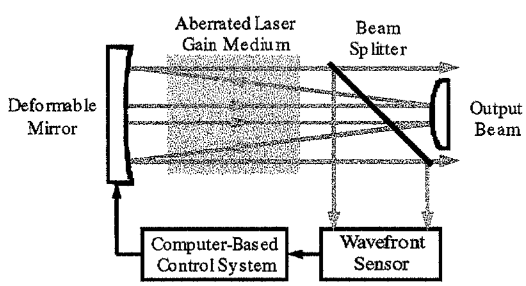

Figure 1. Schematic adaptive resonator.

The core control system for the adaptive resonator consists of real-time code running on two digital signal processors (DSPs). These processors reside in a desktop host computer running a mainstream non-real-time operating system (OS). Additional hardware provides interfaces to the wavefront / diagnostic imaging sensors, deformable mirror control, and laser timing /

† Correspondence: Email: flath1@llnl.gov; Telephone: 925 424 6424; Fax 925 422 1796

interlock control. An object-oriented hardware abstraction layer hides the internal workings of the real-time system, providing a consistent application programming interface (API) to dynamic data-processing on both the DSPs and in host-based threads. This API also furnishes other beneficial services, such as telemetry and diagnostic data streaming. Every attempt was made to minimize platform-dependent code, so as to permit its use in future adaptive optic control systems.

2. SYSTEM DESIGN

As stated previously, with this system we are attempting to meet the real-time control requirements for sustained laser operation with a portable software framework while using low-cost desktop personal computer (PC) hardware. There are several aspects of adaptive optic control systems that apply globally, thus making a fairly generic software API feasible. However, there are specific elements to our system design that require careful consideration.

2.1. Optical Design

The unstable resonator's output wavefront is inherently more uniform than other resonator configurations,[9] and in a confocal arrangement provides a collimated, annular output beam. Previous experiments with a passive (non-lasing) amplifier have demonstrated the feasibility of an adaptive unstable resonator using an intracavity wavefront sensing system.[10]

The laser (Figure 2) consists of an unstable resonator (with primary mirror and secondary mirrors), and a flashlamp-pumped 9-slab heat capacity amplifier. The AO system consists of a wavefront sensor (WFS), and a deformable mirror (DM). Two diagnostic packages, a far-field sensor (FFS) and near-field sensor (NFS) monitor the laser's output beam.

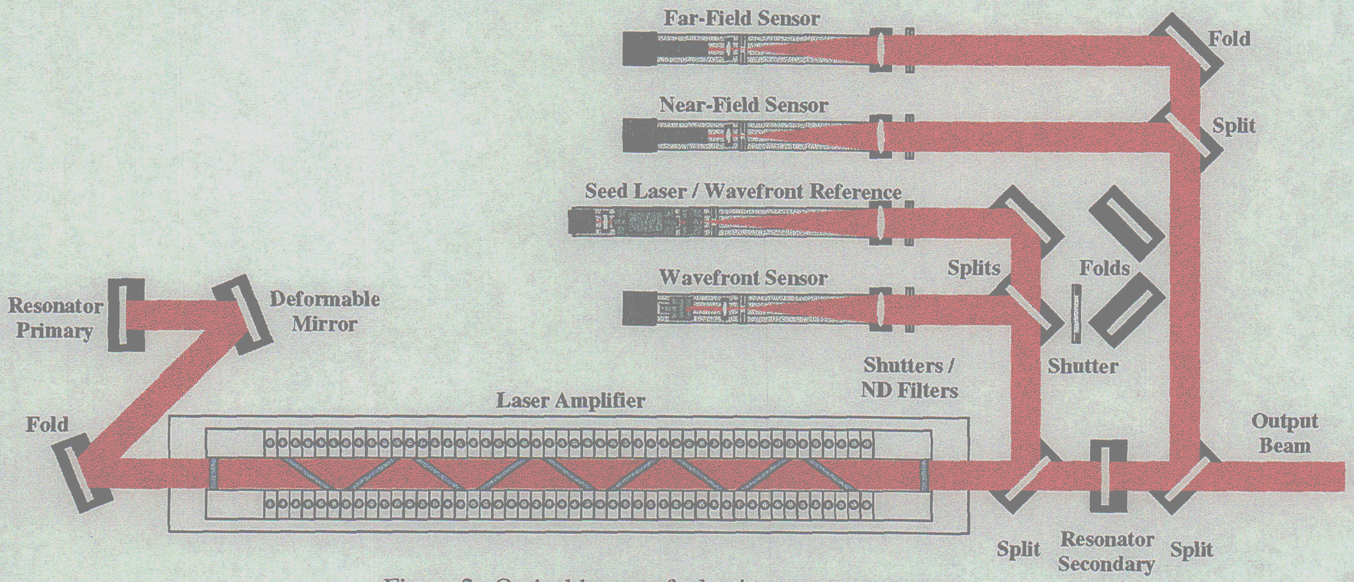

Figure 2. Optical layout of adaptive resonator system.

With a primary of radius R_{PRI} = 30 m, and a secondary radius of R_{PRI} = – 20 m, the confocal resonator arrangement has a magnification M = 1.5, and a cavity length L = 5.0 m, where:[11]

$$M = -R_{PRI} / R_{SEC}\,, \tag{1}$$

and

$$L = (R_{PRI} + R_{SEC}) / 2\,. \tag{2}$$

Our wavefront sensor's lenslet array, with 200 μm lenslet spacing and focal length f = 5 mm (Adaptive Optics Associates), focuses light onto a 256×256, 16 μm pixel frame-transfer CCD (Dalsa CA-D1). The deformable mirror is a Xinetics 150 element device, with 10 mm actuator spacing on a pseudo-hexagonal grid (horizontal and vertical spacings are equal). The

DM actuators are lead-magnesium-niobate (PMN) with ± 4 μm of usable stroke, a Gaussian-like profile function, and a 10% influence on nearest-neighbors. To make best use of the DM's high spatial resolution, tip/tilt control is offloaded to motors on the primary mirror mount. The diagnostic FFS and NFS packages use 1024×1024, 12 μm pixel frame-transfer CCD cameras (Dalsa CA-D7).

The intracavity phase is sampled by a Shack-Hartmann wavefront sensor,[12] which measures the wavefront slope with respect to a reference wavefront (hence, this type of sensor cannot detect piston). The beam-size is reduced via an afocal telescope, and a lenslet array focuses small areas (subapertures) of the beam down to spots on the an imaging sensor. The positions of the spots are calculated using a grayscale centroid algorithm. These locations are then subtracted from the flat-wavefront (reference) spot positions to generate an error signal.

Loop control is achieved by multiplying this error vector by a control matrix, which maps relative wavefront slope to actuator movement on the deformable mirror. At this point, predictive information related to the temporal aberrative behavior of the laser's gain medium is injected into the control data. To insure stability, an integrator function retains a history of the controller's output and allows fine-tuning of the loop's damping factor. Data is then written to the deformable mirror. Figure 3 shows the adaptive resonator's relative arrangement of wavefront sensor, deformable mirror actuators, and beam footprint.

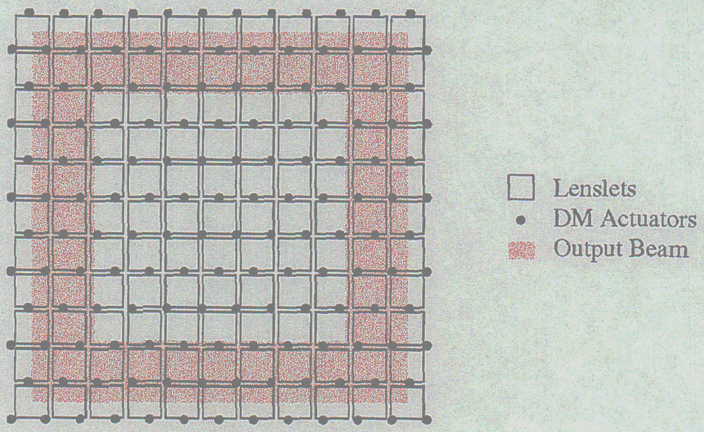

Figure 3. Wavefront control footprint.

2.2. Hardware Design

In the past, control systems for adaptive optics have largely depended on multiprocessor embedded computers, using non-portable, application-specific software.[13] However, recent mainstream commercial processors are broaching the billion instructions-per-second mark, facilitating a complete reasonably-sized AO control system with only a few processors. Also, the cost of hardware and development tools on the PC platform is significantly less than in single-board-computer (SBC) space. Our design relegates non-real-time operations to the PC's processor(s), and the real-time computational requirements are met by auxiliary processors in peripheral component interconnect (PCI) cards.

Our computational engine is split between real-time and non-real-time (host PC) sections; refer to Figure 4. For the real-time component, a modular DSP system from Traquair Data Systems consists of two Texas Instruments 320C6701 167 MHz DSP daughter cards, a frame grabber daughter card for the wavefront sensor, and a digital input/output daughter card for interfacing with the DM, tip/tilt motors, and the laser triggering electronics. The 'C6701 DSPs contain an eight-instruction-wide parallel pipeline (six floating-point, two integer), permitting a theoretical maximum throughput of 1 GFLOP (billion floating-point operations per second) each. These four daughter cards communicate with each other via FIFOs built into a PCI motherboard at up to 167 MB/s. A FIFO connection also permits telemetry, diagnostic, and control data transfer between the real-time system and the host PC over the PCI bus, albeit at a lower data rate. An off-the-shelf dual 450 MHz Pentium III PC with 256 MB of RAM and several GB of magnetic storage provides the non-real-time part of the control system. Diagnostic image data from the NFS and the FFS are collected via two PCI frame-grabbers. Timing of the system is determined by the charging period of the laser amplifier power supplies. Acquisition of the WFS, NFS, and FFS imagery is synchronized to the laser pulse, and loop processing follows immediately. User interface software may either reside on the host PC or on a remote system connected via an Ethernet interface.

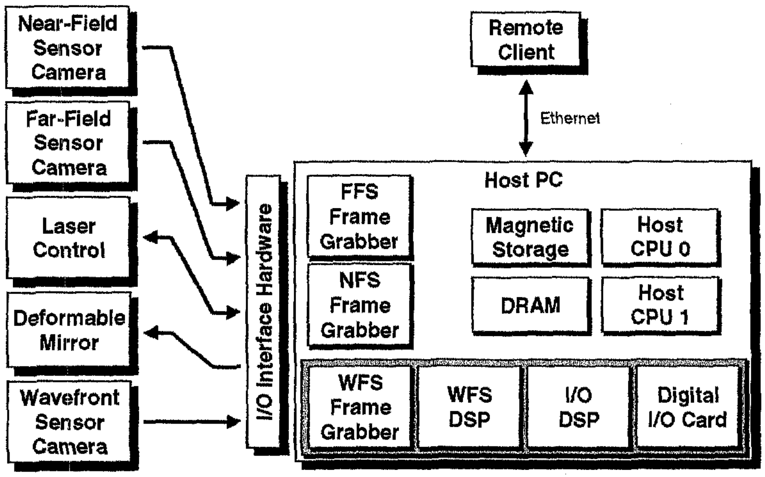

Figure 4. Hardware block diagram.

A small amount of custom hardware is required to interface the sensors, laser signals, deformable mirror, and tip/tilt motors to the control system. Differential signaling is used on all distant devices (cameras, DM), permitting cable lengths greater than 30 m. We fabricated an interface chassis with minimal knowledge of the final signal routing by incorporating complex programmable logic devices (CPLDs) into the data paths; specific protocols and signaling are coded in a hardware description language, compiled, and downloaded whenever a modification is necessary.

2.3. Software Design

Computational hardware is changing at a break-neck pace, while improvements to the software development process have changed little over time.[14] Therefore, to facilitate the goal stated in the introduction of a reusable, platform-independent code base, a three-phase approach was used for the real-time software design. First, an object-oriented application programming interface was designed to abstract basic components of an adaptive control system. This API, written in C++, provides a hardware abstraction layer between user interfaces and the underlying computational engines, operating system services, and interface hardware (Figure 5). Second, the computational engines for an AO control system were implemented in separate threads of execution on the host machine. This allowed testing of algorithmic code and data structures, while providing the benefits of the development and debugging tools available for desktop PCs. Third, the code base from the host was moved over to the DSP development system. Any constructs, such as inter-processor communication, host/DSP messaging, and boot-code loading, were added to the DSP code to provide the same functionality as existed in the host execution threads.

Every effort was made to minimize the differences in the host versus DSP implementations. Some of the features of C++ do not lend themselves well to real-time embedded applications. Thus the TI DSP development environment only supports C and the native processor's assembly language. Originally we were concerned this would mandate significant differences in the code bases, but this turned out not to be the case. Judicious naming conventions and avoiding certain specialized C++ constructs (e.g. multiple inheritance, operator overloading) allowed almost identical source files, with the differences being limited to the existence of scoping operators and the placement of instance-variable declarations.

The host OS is Microsoft Windows NT, so the object-oriented API is implemented in a Win32 dynamic-link library (AOControl.dll). Internally, the objects are written in C++, however, since there are platform- and compiler-specific issues involved with directly exporting member functions in a library, the API is exposed via C-wrapper functions.

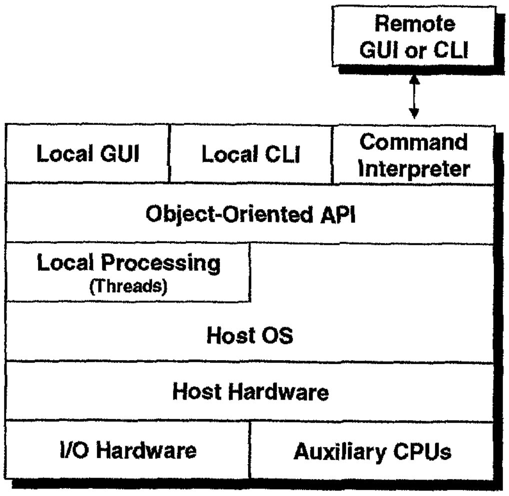

Figure 5. Hardware abstraction topology.

A complete system is configured by instantiating a set of objects, each of which specializes in a particular function in an adaptive optics control system. Referring to Figure 6a, an *AOCamera* encapsulates the functionality of a digital camera; an *AOSensor* converts input data into wavefront measurements; an *AOLoopController* provides common control-law algorithms; an *AOModulator* encapsulates hardware that modifies optical phase. All of these classes are derived from the root object of our AOControl library: *AODataProcessor*. This class provides an interface to a computational engine available to every object in the system, including data transformations (e.g. vector/matrix operations, polynomial functions, lookup tables), real-time parameters (heartbeats, watchdogs), dynamic binding of data-paths both internal to the object and between objects, live diagnostic/telemetry data streaming, extensible markup language- (XML) based object configuration, and numerous debugging hooks. Data flows from the input port of an object, is processed through various computational sub-modules, and is presented at the object's output port, where it is available to the inputs of one or more other objects (Figure 6b). Specific sub-modules may be turned on/off at any time, and may be reordered to suit the application. Constructing a complete system entails 'wiring' the output data port of one object to the input data port of the next.

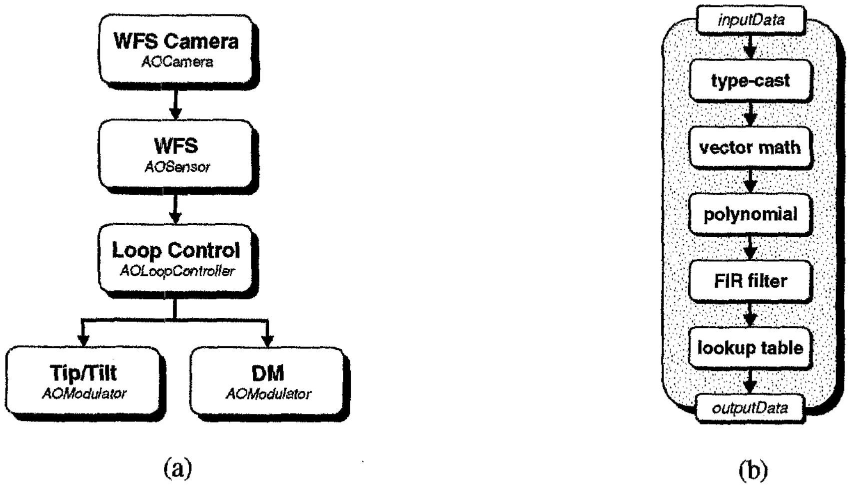

Figure 6. Object-based adaptive optics control. (a) A complete AO control system (italics indicate class name), and (b) an example *AODataProcessor* object with specific internal processing sub-modules enabled.

The user of the library communicates with object instances in a control system by calling accessor member functions, usually in the form of *get/set* pairs. Internal to the objects, these functions then communicate with the computational engine to retrieve the requested information. For host-based objects, semaphores allow communication with the object's processing thread. For remote objects, communication occurs via messaging over the PCI bus to the appropriate DSP. Data processing always takes precedence over information requests.

The definition of "real-time" for our system means that WFS data must be sampled, processed, and written to the adaptive optic devices before the next laser pulse occurs. Analysis of our algorithms and the sizes of our data sets revealed that two DSPs are required to perform all of the required control loop operations (processing data, exporting telemetry/diagnostic information, servicing interrupts) in the available 50 ms.[15] A watchdog timer interrupts loop processing if the execution time exceeds a certain value, insuring that the loop is ready to process the next set of input data -- in this situation, the DM shape is left unchanged. As there is no need for a true pre-emptive scheduler, each DSP runs a tight event loop and processes new data as soon as it is available. This is the same event loop used for the execution thread of host-based objects.

In parallel to the real-time code development, a graphical user interface (GUI) was written in Java to provide a platform-independent means to control the system. Communication with the host PC is facilitated through TCP/IP sockets, the programming implementation of which is relatively straightforward in Java.[16] A C-based command interpreter running on the host configures an object-based control system, and then proceeds to process commands sent over Ethernet. One of the benefits of this design is that sockets work for both local and remote GUI clients (i.e. both situations use identical software), with performance limited only by available bandwidth.

3. RESULTS

At the present time, the majority of the system design is implemented. All DSP/host hardware interfaces, including cameras, deformable mirror, tip/tilt motors, and laser synchronization signaling, are complete and fully tested.

The platform-independent GUI is functional, and runs on both the host PC and remotely via an Ethernet connection. Preliminary tests indicate that the remote option with a dedicated network provides higher performance than running all of the software on the host PC (i.e. using an internal TCP/IP socket for control system ↔ GUI communication). It is postulated that perceived user performance degradation is due more to the CPU demands of the Java virtual machine than network bandwidth.

The choice of exposing our adaptive optics control API though a Win32 dynamic-link library has permitted us a large amount of flexibility; in addition to our C-based command interpreter, C-based test applications are used in console (text) mode, our calibration procedures are performed in Research Systems, Inc.'s Interactive Data Language (IDL), and quick GUIs are easily written in Microsoft's VisualBasic.

Probably the most surprising aspect of developing a nearly identical code base on the Pentium and 'C6701 platforms simultaneously is the relative performance of the two CPUs running our algorithms. While the 'C6701 advertises 1GFLOPs at a clock frequency of 167 MHz, this assumes that the algorithm uses all six internal floating-point pipelines simultaneously. The likelihood of this scenario is extremely rare; disassembly of our high-level code reveals, even at full optimization levels, performance of a single such DSP is more on the order of 300 MFLOPs. Obviously, hand-coding techniques (e.g. loop unwrapping, pipeline optimization, use of intrinsic instructions) can easily double this figure. However, the cost of time spent in coding these very CPU-specific optimizations is easily offset by Moore's Law (a CPU with 50% higher performance is released in a matter of months), especially if custom algorithms cannot be implemented using off-the-shelf libraries. The multi-threaded nature of host–based objects lends itself well to multiprocessor (MP) configurations, and runs very smoothly in our dual-Pentium III PC. Given the complexities of programming / communication with remote embedded CPUs and the associated hardware cost premiums, for future adaptive optics projects we highly recommend a computational engine using the PC platform itself, running a deterministic real-time operating system such as VxWorks or Real-Time Linux. It should be noted though, that larger data sizes than used by our system may still dictate the use of the high-speed I/O and MP configurations available only in real-time hardware; the PC's performance will always lag behind that of state-of-the-art SBC hardware.

4. CONCLUSION

Adaptive optic control systems are essential for high-power solid-state lasers with high output wavefront quality. We have designed and constructed an adaptive resonator using an intracavity wavefront sensor and high spatial resolution deformable mirror. An object-oriented interface to the real-time control processing provides a highly configurable system. It is clear, due to the ever-improving processors in the personal computer space, that the computational engine required for adaptive beam control will become smaller and much lower in cost.

Following the completion of the control system, the next stage of the project involves integration with the laser amplifier hardware. First-light from the adaptive resonator will follow shortly thereafter. A future paper will describe the tests used to verify our control algorithms and report the performance of the operational laser.

ACKNOWLEDGEMENTS

This work was performed under the auspices of the U. S. Department of Energy by the Lawrence Livermore National Laboratory under contract number W-7405-ENG-48. The authors would like to acknowledge the support and technical assistance of G. Albrecht, B. Bauman, B. Bhachu, R. Buff, C. Carrano, C. B. Dane, S. Fochs, D. Gavel, L. Lindberg, W. Manning, E. Stappaerts, C. Thompson, and J. Wintemute. This research was funded by the U.S. Army High-Energy Laser Test Facility under WFO Proposal No. L7378.

REFERENCES

1. W. Koechner, Solid-State *Laser Engineering*, Fourth Edition, pp. 262-278, Springer-Verlag, Berlin, 1996.
2. G. F. Albrecht, S. B. Sutton, E. V. George, W. R. Sooy, and W. F. Krupke, "Solid state heat capacity disk laser," *Laser and Particle Beams*, **16-4**, pp. 605-625, 1998.
3. K. E. Oughstun, "Intracavity adaptive optic compensation of phase aberrations. I: Analysis," *Journal of the OSA*, **71-7**, pp. 862-872, 1981.
4. J. M. Spinhirne, D. Anafi, R. H. Freeman, and H. R. Garcia, "Intracavity adaptive optics. I. Astigmatism correction performance," *Applied Optics*, **20-6**, pp. 976-984, 1981.
5. J. M. Brase, J. An, K. Avicola, H. D. Bissinger, H. W. Friedman, D. T. Gavel, B. Johnston, C. E. Max, S. S. Olivier, R. Presta, D. A. Rapp, J. T. Salmon, K. E. Waltjen, "Adaptive optices at Lick Observatory: System architecture and operations," *Proceedings of the SPIE*, **2201**. 474-488, 1994.
6. R. Dekany, K. Wallace, G. Brack, B. R. Oppenheimer, D. Palmer, "Initial test results from the Palomar 200" adaptive optics system," *Proceedings of the SPIE*, **3126**. 269-276, 1997.
7. R. H. Freeman, R. J. Freiberg, and H. R. Garcia, "Adaptive laser resonator," *Optics Letters*, **2-3**, pp. 61-63, 1978.
8. A. V. Kudryashov, "Intracavity laser beam control," *Proceedings of the SPIE*, **3611**, pp. 32-41, 1999.
9. A. E. Siegman, "Unstable optical resonators," *Applied Optics*, **13-2**, pp. 353-367, 1974.
10. L. Flath, J. An, J. Brase, C. Carrano, C. B. Dane, S. Fochs, R. Hurd, M. Kartz, and R. Sawvel, "Development of adaptive resonator techniques for high-power lasers, " *Proceedings of 2nd International Workshop on Adaptive Optics for Industry and Medicine*, pp. 163-168, July 1999.
11. A. E. Siegmann, *Lasers*, pp. 858-922, University Science Books, Sausalito, 1986.
12. R. K. Tyson, *Principles of Adaptive Optics, Second Edition*, pp. 163-170, Academic Press, San Diego, 1998.
13. R. Q. Fugate, "Progress in atmospheric adaptive optics," *Proceedings of the SPIE*, **3749**. 24-25, 1999.
14. S. McConnell, *Code Complete*, Microsoft Press, Redmond, 1993.
15. K. Tindell, "Deadline Monotonic Analysis," *Embedded Systems Programming*, **13-6**, pp. 20-38, 2000.
16. P. M. Tyma, G. Torok, T. Downing, *Java Primer Plus*, pp. 328-346, Waite Group Press, Corte Madera, 1996.