Rapid Flight Control Prototyping - Steps Toward Cooperative Mission-Oriented Capabilities

Vladimir Dobrokhodov[1], Kevin Jones[2], and Isaac Kaminer[3]

***Abstract*— The paper describes the latest advancements in the development of the Rapid Flight Control Prototyping system that were motivated primarily by the need to enable cooperative missions of multiple unmanned aerial vehicles and by the desire to explore and enhance the capabilities of human operators to oversee the collaborative behaviors of multiple heterogeneous UAVs. The evolution of the system is driven by the mission level objectives and supported on one hand by the progress in miniature sensors, computational power, communication and portable energy technologies and on the other hand by the advanced capabilities of embedded control and communication oriented software. As a result the developed system enables rapid design, onboard integration and in-flight verification of multiple UAVs collaborative concepts that seemed impossible just a couple of years ago. Advantages of the designed system are illustrated by a number of scenarios that were recently developed and verified in flight by multiple cooperative UAVs. The paper concentrates on presenting the motivation and the conceptual design ideas which drive the evolution of flight prototyping platform.**

I. INTRODUCTION

High operational utility of a single unmanned aerial vehicle (UAV) has been proven in recent years on many occasions. Single UAV has been primarily used in various ISR missions that have been designed around the capabilities of a single platform, in turn the onboard instrumentation were adjusted to facilitate a specific ISR utility of interest. When properly designed the aerial platform provided a single unique autonomous capability however producing only a humongous amount of raw data but intelligence content was still missing. The fact that the onboard intelligence is limited by various factors (probably it makes sense defining them here) resulted in the "Large Data Problem": a misleading exchange of quality for quantity. In real life scenarios this resulted in the raw data being streamed in close to real-time pace however the human analysts not being capable to process this amount of data in weeks after the mission. In response to this bottleneck of capabilities the R&D community has focused its efforts on the faster processing and automated analysis. However, getting to the data faster, and utilizing more data through automated tools cannot solve some of the most difficult problems, among them is the precision and resolution of data suitable for further use in predictive analytics. Thus, in order to eliminate the creation and need for large data sets, an intelligent high-precision and resolution capability for sensing and surveillance needs to be developed for time critical applications.

Overall, the vision of how to overcome the intelligence bottleneck and to develop a rich data set capability seems to be in the development of distributed platforms, both in the sense of distributed computational intelligence (onboard and ground) as well as in distributing the airborne sensory capabilities that would leverage the features of multiple complementary sensors. Enabling this level of collaborative autonomy, considering given state of the art in communication, computational power and power technologies, requires significant advances to be made in the areas of distributed mission planning, coordinated control, human-machine interaction, distributed data processing to name a few. Some of these capabilities need to be transitioned from the laboratory environment onboard of multiple collaborative UAVs and proven working in flight in a realistic setup.

As steps toward developing and implementing these capabilities in flight, the Unmanned Aerial Systems laboratory at NPS has been developing a Rapid Flight Control Prototyping system (RFCPS). Over years of evolution the RFCPS system has arrived to the state capable of implementing a number of the desired solutions in flight. The key capabilities of the system enable verifiable (repeatable) execution of the multiple heterogeneous UAVs cooperative missions, onboard data preprocessing, in-flight information streaming and sharing; some of them are running onboard while always providing the required level of collision avoidance and flight safety in a multiple UAV sense. When a mission is planned ahead it creates a set of collaborative tasks accounting for the mission objective along with the capabilities of UAV platforms. When executed in flight by multiple UAVs the context-driven raw data preprocessing is followed by an intelligent reaction. Therefore, focused and context adaptive tasking that assumes collaboration among multiple players results in significantly lower volumes of high quality data.

The RFCPS system is evolutionary built around capabilities of an advanced autopilot connected over a real-time communication link with a secondary set of controllers responsible for a variety of tasks including real-time flight-critical control, raw data pre-processing, and robust communication tasks. Rapid hardware prototyping that "mimics" CAD drawings in strong and lightweight materials, new microcontrollers and high speed and precision actuators enable quick integrating onboard of various sensors. Novel high-

[1]V.N. Dobrokhodov is a Research Associate Professor at the Department of Mechanical and Aerospace Engineering, Naval Postgraduate School, Monterey, CA, 93943 vldobr@nps.edu

[2]K.D. Jones is a Research Associate Professor at the Department of Mechanical and Aerospace Engineering, Naval Postgraduate School, Monterey, CA, 93943 jones@nps.edu

[2]I.I. Kaminer is a Professor at the Department of Mechanical and Aerospace Engineering, Naval Postgraduate School, Monterey, CA, 93943 kaminer@nps.edu

bandwidth wireless IP communication link allows beyond the LOS in-flight transmission of sensory information and the R&D telemetry without any need to utilize the primary CC link. New software solutions implementing advanced cooperative GNC algorithms, robust communication messaging and onboard sensory data processing enable cooperative flight of multiple heterogeneous UAVs and their interaction with human operators. Thus, the systems engineering effort of designing and integrating various components resulted in a system capable to rapidly prototype a mission of unknown complexity.

The paper addresses the underlying design concepts implemented by the RFCPS system, the presentation focuses on architectural ideas and the results achieved by the setup. Thus, Section II of the paper outlines two basic types of rapid control prototyping systems that were initiated at the UAV lab: they include OS-based and a System on a Chip (SOC) designs. While the latter one is briefly presented, the primary focus of this publication is given to the OS-based architectures. Illustrating the Model Based Design (MBD) paradigm and the rapid verifiable code generation that are provided by the Simulink and Simulink Coder tools, the xPC-based design is presented as a first step in transitioning a theoretical concept to real flight. The steps of transitioning the same solution (already flown) to the Linux-based semi-industrial implementation are presented next. Chapter IV demonstrates a set of multiple cooperative UAV missions that illustrate the developed solutions in flight.

II. UAV Software Prototyping Solutions

Over the years of control algorithms development and fight experimentation it has been observed that in order to transition a theoretically proven result in flight a number of solutions of various engineering fields need to simultaneously converge to a unifying design. The list of those disciplines varies depending on the objectives of the project. However the core remains the same and includes the following: materials and structural design accompanied with hardware prototyping, aero and flight dynamics, GNC disciplines, onboard electronics and sensors integration, communication and signal processing, and software engineering. To guarantee safe and verifiable flight experimentation and timely results it is often convenient to split the design process into a number of phases and then concentrate on developing a reliable solution at each step.

One of the key elements in building a reliable solution of an onboard flight control system is based on utilizing a set of tools that enable control system design, automatic code generation and its formal verification and validation. The MathWorks's MatLab/Simulink and the Simulink coder are the tools that provide a well-developed model based design (MBD) approach to the modeling and simulation of various engineering systems and processes. Control system design is especially well-supported by Simulink and a number of control oriented toolboxes. Capabilities of the Simulink coder (formerly called the Real-Time Workshop) significantly extend the applicability of solutions developed by utilizing the core capabilities of the MatLab-Simulink environment. The code, that is auto-generated by the system can be easily customized to address the specifics of the control design task, available hardware (sensors and actuators) and the airborne platform requirements. A number of code generation options addressing specific microprocessor architectures is provided by the Simulink coder.

As a results of control prototyping and fight experimentation a number of solutions has been developed and flight tested, see the hierarchy of possible computational platforms in Fig. 1. All of the developed solutions were driven by the objective of utilizing a set of software tools specifically developed for the design of real-time control algorithms and their formal software verification.

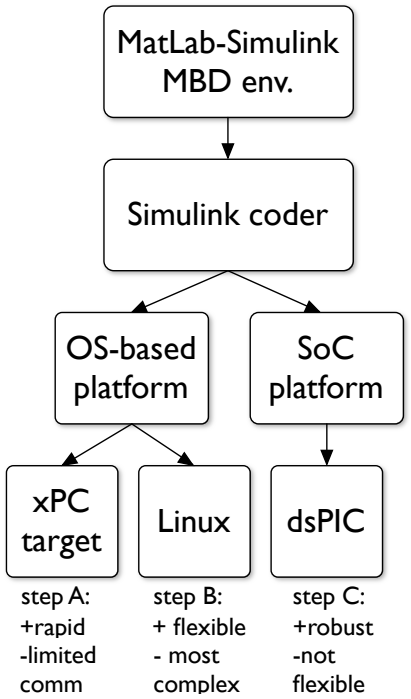

Fig. 1. Hierachy of computational platforms.

When a design task requires developing a standalone solution (feedback controller) that is self sufficient (does not depend on additional services), can be executed by a microprocessor capable of communicating with external sensors (measure states) and actuators (execute control commands), and does not require frequent modification, then a system on a chip (SOC) is one of the viable options. The generated code that is usually called a firmware is uploaded to the chip memory once and then enables the code execution at every time when the system is turned on. The SOC design approach that starts with pure Simulink modeling, code generation and implementation on one of the most widely used microcontrollers (dsPIC) was recently demonstrated in a number of successful flight tests, see Ref[]. The systems built on SOC approach are very capable and

robust, however do not enable in-flight modifications of the onboard algorithms. Although this might seem unnecessary, the flexibility and convenience of the "modification at any time" option is very desirable especially considering that modern UAV platform can provide 2-3 hours of a single flight endurance thus potentially enabling multiple "code modifications" to be flight tested if necessary. Thus, the SOC approach is usually considered as a final step of algorithm integration, when the code is proven working and there is no need for further immediate modification.

On the other hand, the specifics of R&D work frequently require modifications of the underlying algorithms. Thus the "xPC embedded target" option, provided by the Simulink Coder is one of the best solutions enabling frequent and rapid code modification and reintegration. An xPC target environment is a hard real-time operating system that executes the real time code (task) that is built by the coder supporting a number of computational platforms. The solution provides various services typical for an operating system while supporting preemptive mechanism for real time tasks scheduling and execution. The RFCPS system utilizes this option for standard *Intel*x86 processor. While the xPC embedded target option is very flexible, it still requires "manual" coding in a number of occasions. Since it is hard to expect that every new piece of hardware has the same communication protocol and is supported by Mathwork's, one of the primary needs for manual coding does arise when a new communication protocol need to be implemented. Simulink coder enables integration of customized algorithms by means of s-functions; the process is time consuming and error prone, however it is well documented and straightforward.

The desire to concentrate on rapid and verifiable flight implementation of new theoretical results and well-developed legacy codes, given in a variety of programming languages, while spending less time on the code development and reintegration resulted in targeting a new computational platform. The high efficiency of Linux operating system and high computational power of modern miniature computers lead to integrating this system into flight experimentation. Furthermore, the objectives of collaborative missions and distributed computation require integration of messaging and synchronization mechanisms that are robust to network failures and communications dropouts. A number of industrial-grade software packages that enable communication across multiple networking nodes while explicitly addressing the messaging protocols and algorithms that are resilient to faults across local and wide area networks have been recently developed, see [SPREAD],[MOOS]. Therefore, the most recent modification of the RFCPS included the implementation of Simulink/Coder built algorithms into the Linux operating system. The capabilities of Simulink to utilize communication buses and features of the Simulink coder allow generating C/C++ libraries that are directly suitable for compilation and execution on Linux. Software API of the SPREAD toolkit enables efficient messaging across various tasks executed across multiple wirelessly connected UAVs. No manual modification of the library code is necessary, however the tasks scheduling file (main) needs to explicitly define the sampling rate of each of the code components and the messaging mechanism. Special consideration is now given to scheduling of the flight-critical and not critical tasks. One of the initially developed scheduling solution utilizes a simple function call to the CPU time clock. Comparing the clock time with the required execution rate enabled efficient implementation of the required sampling of various tasks (daemons) in Linux. The most recent solution that adopts the same auto-generated code and is based on more rigorous real time scheduling capabilities of ROS/OROCOS operating system [ROS],[OROCOS] has been also tested in the laboratory environment.

A. RFCPS Architecture

To enable the desired collaborative functionality the RFCPS system representing a single UAV is comprised of the onboard and the ground segment, see Figure 2 .The onboard segment is built around an autopilot that brings the basic functionality of a UAV flight to the system. The current implementation of RFCPS utilizes the Piccolo series autopilots and the corresponding ground control station that enables simultaneous operation of multiple UAVs but does not have any cooperative mission execution functionality prebuilt.

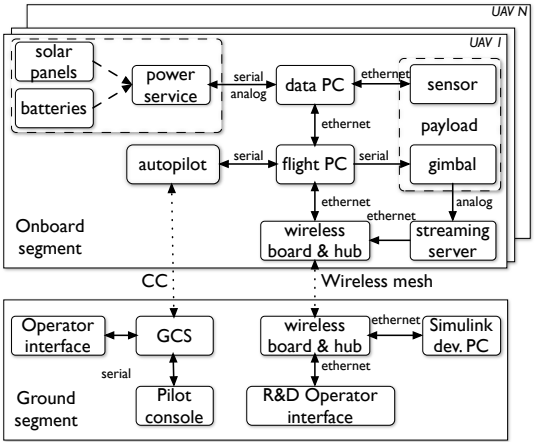

Fig. 2. Single autonomous platform as a building block of RFCPS.

The autopilot is connected over a full duplex serial link to a flight control PC104 computer. In xPC target version, a set of communication drivers is custom built that enable hard real time communication with the AP. The drivers enable *send command* and *read states* functionality and are written as s-functions; they are automatically compiled into a binary executable file by the Simulink coder at the stage of code generation. As soon as the UAV telemetry becomes available it is first used to implement the desired collaborative behavior and to control the onboard utility sensor (for example - a high resolution camera) and its platform; for most of the ISR type missions it is common to have the imagery sensors installed on a gimbaled platform. The telemetry data is also distributed across the local onboard and the ground

ethernet network for the secondary use. In particular, the data PC104 computer utilizes the AP telemetry data and the utility sensor information to preprocess the raw data and extract the intelligent data of interest. A very appealing feature of this data PC is that it runs a Linux operating system that enables a number of convenient services and provides very flexible access to a variety of sensors. Unlike the xPC target system, the integration a new sensor becomes a matter of integrating a new driver that is usually provided by the sensor manufacturer while integrating a new piece of hardware that is not supported by the xPC target is often a significant effort. As it will be shown in the following chapter this data computer runs algorithms that are primarily not time-critical and do not affect the stability of UAV flight. In the case of utilizing video cameras as ISR utility sensors the onboard instrumentation also contains a miniature video streaming server; although it is a single pice of hardware, it is comprised of a frame grabber taking the analog video feed and a web server that enables live streaming of the video input into the IP based network. Since all of the onboard components are connected over the hight speed (up to 1GigE) ethernet a number of convenient capabilities of the code development become available; some of the not flight-critical algorithms can be stopped, modified and updated while still in flight thus providing the desired flexibility for the research.

The last significant onboard component is the power management system. Depending on the specific objectives of a project it allows integrating into onboard information system a set of solar panels and batteries. As it will be demonstrated in the following chapter, the integration of high-output solar cells onboard enables harvesting solar radiation thus significantly extending endurance of autonomous thermal soaring gliders. Continuos monitoring of the current state of the energy of all onboard sources enables RFCPS system to optimally plan and execute the mission.

The ground segment of RFCPS is comprised of two parts; the first part is a standalone ground control station (GCS), an operator interface (OI) computer and a command and control (CC) link provided by the autopilot manufacturer, and the second part is the research environment that is customized according to the needs of a specific mission under development. The fact that the onboard software is separated into flight safety critical and not flight critical defines the configuration of the R&D development environment. As an example, the sensory data processing algorithms that are not flight critical are run on a data PC; two primary OS systems are currently used - the Ubuntu Linux and ROS operating systems. The flight critical tasks are run under either xPC target or the Linux/ROS(OROCOS) environments. Although the hardware of both PC104 computers is identical (for convenience) the task based separation is preserved to guarantee safety of flight experimentation and reliability of the experimental results.

To preserve the rigor of theoretical development while transitioning the results in flight the core of cooperative flight control algorithms is developed utilizing advanced capabilities of the MatLab/Simulink environment. The Model Based Design (MBD) paradigm supported by a number of code verification and automatic code generation tools is what makes the transition very reliable. The algorithms are designed and verified in Simulink with minimal or no use of low level programming. When debugged, the xPC target environment is used first to enable the Hardware In the Loop (HIL) simulation; Piccolo AP is well-supported with this functionality. Therefore, integrating and verifying the newly built algorithms becomes straightforward; at the same time a number of data processing scripts can be built ahead of time to facilitate quick flight data analysis. After confirming the desired performance of cooperative mission in HIL, the same xPC based code is transformed into real time executable and then flown in the air. After a series of flight experiments in different flight conditions the algorithms and the operational environment mature enough to allow building a set of standalone libraries for future Linux or ROS integration. The volume limitation of the paper does not allow to describe all the benefits of this new direction. However it is sufficient to note that the same development concept is followed and the same code that is built by the Simulink coder is used to transition the flight validated solution into a highly reliable Linux/ROS based computational platform. An interested reader is referred to Ref for mored details.

III. Experimental Results

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

A. Coordinated Road Search by Multiple UAVs

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. Cognitive Autonamition

C. Autonomous Thermal Soaring Gliders

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as 3.5-inch disk drive.
- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.

- Do not mix complete spellings and abbreviations of units: Wb/m2 or webers per square meter, not webers/m2. Spell out units when they appear in text: . . . a few henries, not . . . a few H.
- Use a zero before decimal points: 0.25, not .25. Use cm3, not cc. (bullet list)

D. Conclusion

The equations are an exception to the prescribed specifications of this template. You will need to determine whether or not your equation should be typed using either the Times New Roman or the Symbol font (please no other font). To create multileveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled. Number equations consecutively. Equation numbers, within parentheses, are to position flush right, as in (1), using a right tab stop. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in

$$\alpha + \beta = \chi \tag{1}$$

Note that the equation is centered using a center tab stop. Be sure that the symbols in your equation have been defined before or immediately following the equation. Use (1), not Eq. (1) or equation (1), except at the beginning of a sentence: Equation (1) is . . .

E. Some Common Mistakes

- The word data is plural, not singular.
- The subscript for the permeability of vacuum ?0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter o.
- In American English, commas, semi-/colons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an inset, not an insert. The word alternatively is preferred to the word alternately (unless you really mean something that alternates).
- Do not use the word essentially to mean approximately or effectively.
- In your paper title, if the words that uses can accurately replace the word using, capitalize the u; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones affect and effect, complement and compliment, discreet and discrete, principal and principle.
- Do not confuse imply and infer.
- The prefix non is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the et in the Latin abbreviation et al..
- The abbreviation i.e. means that is, and the abbreviation e.g. means for example.

IV. USING THE TEMPLATE

Use this sample document as your LaTeX source file to create your document. Save this file as **root.tex**. You have to make sure to use the cls file that came with this distribution. If you use a different style file, you cannot expect to get required margins. Note also that when you are creating your out PDF file, the source file is only part of the equation. *Your TEX → PDF filter determines the output file size. Even if you make all the specifications to output a letter file in the source - if you filter is set to produce A4, you will only get A4 output.*

It is impossible to account for all possible situation, one would encounter using TEX. If you are using multiple TEX files you must make sure that the "MAIN" source file is called root.tex - this is particularly important if your conference is using PaperPlaza's built in TEX to PDF conversion tool.

A. Headings, etc

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced. Styles named Heading 1, Heading 2, Heading 3, and Heading 4 are prescribed.

B. Figures and Tables

Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation Fig. 1, even at the beginning of a sentence.

TABLE I
AN EXAMPLE OF A TABLE

One	Two
Three	Four

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity Magnetization, or Magnetization, M, not just M. If including units in the label, present them within parentheses. Do not label axes only with units.

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi TIFF or EPS file, with all fonts embedded) because, in an document, this method is somewhat more stable than directly inserting a picture.

Fig. 3. Inductance of oscillation winding on amorphous magnetic core versus DC bias magnetic field

In the example, write Magnetization (A/m) or Magnetization A[m(1)], not just A/m. Do not label axes with a ratio of quantities and units. For example, write Temperature (K), not Temperature/K.

V. CONCLUSIONS

A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

ACKNOWLEDGMENT

The preferred spelling of the word acknowledgment in America is without an e after the g. Avoid the stilted expression, One of us (R. B. G.) thanks . . . Instead, try R. B. G. thanks. Put sponsor acknowledgments in the unnumbered footnote on the first page.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

REFERENCES

[1] G. O. Young, Synthetic structure of industrial plastics (Book style with paper title and editor), in Plastics, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 1564.

[2] W.-K. Chen, Linear Networks and Systems (Book style). Belmont, CA: Wadsworth, 1993, pp. 123135.

[3] H. Poor, An Introduction to Signal Detection and Estimation. New York: Springer-Verlag, 1985, ch. 4.

[4] B. Smith, An approach to graphs of linear forms (Unpublished work style), unpublished.

[5] E. H. Miller, A note on reflector arrays (Periodical styleAccepted for publication), IEEE Trans. Antennas Propagat., to be publised.

[6] J. Wang, Fundamentals of erbium-doped fiber amplifiers arrays (Periodical styleSubmitted for publication), IEEE J. Quantum Electron., submitted for publication.

[7] C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.

[8] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, Electron spectroscopy studies on magneto-optical media and plastic substrate interfaces(Translation Journals style), IEEE Transl. J. Magn.Jpn., vol. 2, Aug. 1987, pp. 740741 [Dig. 9th Annu. Conf. Magnetics Japan, 1982, p. 301].

[9] M. Young, The Techincal Writers Handbook. Mill Valley, CA: University Science, 1989.

[10] J. U. Duncombe, Infrared navigationPart I: An assessment of feasibility (Periodical style), IEEE Trans. Electron Devices, vol. ED-11, pp. 3439, Jan. 1959.

[11] S. Chen, B. Mulgrew, and P. M. Grant, A clustering technique for digital communications channel equalization using radial basis function networks, IEEE Trans. Neural Networks, vol. 4, pp. 570578, July 1993.

[12] R. W. Lucky, Automatic equalization for digital communication, Bell Syst. Tech. J., vol. 44, no. 4, pp. 547588, Apr. 1965.

[13] S. P. Bingulac, On the compatibility of adaptive controllers (Published Conference Proceedings style), in Proc. 4th Annu. Allerton Conf. Circuits and Systems Theory, New York, 1994, pp. 816.

[14] G. R. Faulhaber, Design of service systems with priority reservation, in Conf. Rec. 1995 IEEE Int. Conf. Communications, pp. 38.

[15] W. D. Doyle, Magnetization reversal in films with biaxial anisotropy, in 1987 Proc. INTERMAG Conf., pp. 2.2-12.2-6.

[16] G. W. Juette and L. E. Zeffanella, Radio noise currents n short sections on bundle conductors (Presented Conference Paper style), presented at the IEEE Summer power Meeting, Dallas, TX, June 2227, 1990, Paper 90 SM 690-0 PWRS.

[17] J. G. Kreifeldt, An analysis of surface-detected EMG as an amplitude-modulated noise, presented at the 1989 Int. Conf. Medicine and Biological Engineering, Chicago, IL.

[18] J. Williams, Narrow-band analyzer (Thesis or Dissertation style), Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, 1993.

[19] N. Kawasaki, Parametric study of thermal and chemical nonequilibrium nozzle flow, M.S. thesis, Dept. Electron. Eng., Osaka Univ., Osaka, Japan, 1993.

[20] J. P. Wilkinson, Nonlinear resonant circuit devices (Patent style), U.S. Patent 3 624 12, July 16, 1990.