To the Graduate Council:

I am submitting herewith a thesis written by J. T. Greer entitled "Design Considerations for Automated Electron Beam Welding." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

J. M. Bailey, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Vice Provost
and Dean of The Graduate School

DESIGN CONSIDERATIONS FOR AUTOMATED ELECTRON BEAM WELDING

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

J. T. Greer

December 1988

ACKNOWLEDGMENTS

The author thanks Dr. J. M. Bailey, Major Professor, for his many fruitful discussions and suggestions. Recognition and appreciation are due to Dr. J. M. Googe, Professor and Electrical Engineering Department Head, and Dr. W. L. Greene, former Electrical Engineering Head, for their encouragement, instruction, and personal friendship. Finally, the author wishes to express deep appreciation to his wife, Tina, whose love and devotion have helped the author more than words can express.

To quote Robert Lee, author of the book *Optimal Estimation and Control*, "Before God, all men are equally foolish, for we cannot begin to comprehend all the infinite mysteries surrounding us. If there be a grain of wisdom in this work, may it be to His glory, for He alone hath provided me with this splendid opportunity for learning."

ABSTRACT

The wide interest in electron beam (EB) manufacturing applications has intensified because of interest in high-productivity and high-quality manufacturing. Applications for EB welding in space, defense projects, Strategic Defense Initiative space stations, and the construction of NASA's space modules have also intensified the interest for automated EB welding systems. A control system that will provide "robust" automation and exception handling requires extensive knowledge of the process. For complex, multivariable, nonlinear, nonstationary processes, various techniques are usually required to capture the knowledge necessary to handle process exceptions.

This thesis develops a methodology and hierarchical control structure and applies it to the EB welding control problem. The EB gun controller is specifically used to illustrate the methodology and the mapping into the proposed hierarchical control and analysis structure. Special attention is given to the process evaluator element of the hierarchical analysis structure, which utilizes a combination of qualitative, procedural, and mathematical methods to evaluate and to handle process exceptions. The EB gun, filament controller, and emission controller are modeled and integrated together so that the system/subsystem issues can be evaluated.

TABLE OF CONTENTS

SECTION	PAGE
I. INTRODUCTION	1
Overview	1
Applications	5
II. HIERARCHICAL CONTROL ARCHITECTURE	7
Description of Architecture	7
Application to Electron Beam Welding	10
III. MODELING OF NONLINEAR SUBSYSTEMS	12
Nonlinear Electron Emission Model	12
Electron Beam Gun Controller Simulation	24
IV. ANALYSIS OF SUBSYSTEM PERFORMANCE	35
Exception Handling	35
Performance Evaluation	41
V. DISCUSSION OF RESULTS	56
VI. SUMMARY AND CONCLUSIONS	58
BIBLIOGRAPHY	60
APPENDIX	63
VITA	130

LIST OF FIGURES

FIGURE PAGE

1. HIERARCHICAL CONTROL ARCHITECTURE 2
2. ELECTRON BEAM WELDING SYSTEM/SUBSYSTEMS 6
3. DECISION MODEL 8
4. EB GUN FILAMENT DIAGRAM 15
5. DIAGRAM OF FILAMENT 19
6. DIAGRAM FOR THERMAL MODEL 21
7. RADIANT COOLING OF FILAMENT 25
8. EMISSION CURRENT (EB GUN CURRENT) WITH RADIANT COOLING OF FILAMENT 26
9. BLOCK DIAGRAM OF FILAMENT CONTROLLER 28
10. EMISSION, FILAMENT CURRENT STEP RESPONSE WITH FILAMENT PID NOT PROPERLY TUNED 29
11. EMISSION, FILAMENT CURRENT STEP RESPONSE WITH BOTH PIDs PROPERLY TUNED 31
12. BLOCK DIAGRAM OF ELECTRON BEAM GUN CONTROLLER 32
13. BLOCK DIAGRAM OF CLOSED-LOOP EBGC 33
14. REGION EXCLUSION ERROR DETECTION AND MODE CONTROL 37
15. EXEMPLARY RESULTS OF RLS ALGORITHM WITH PROCESS EXCEPTION 39
16. RESPONSE TO HV ARCING, STEADY-STATE CONDITION 40
17. EMISSION, FILAMENT CURRENT STEP RESPONSE WITHOUT MODE CONTROL 42
18. EMISSION, FILAMENT CURRENT STEP RESPONSE WITH MODE CONTROL 44

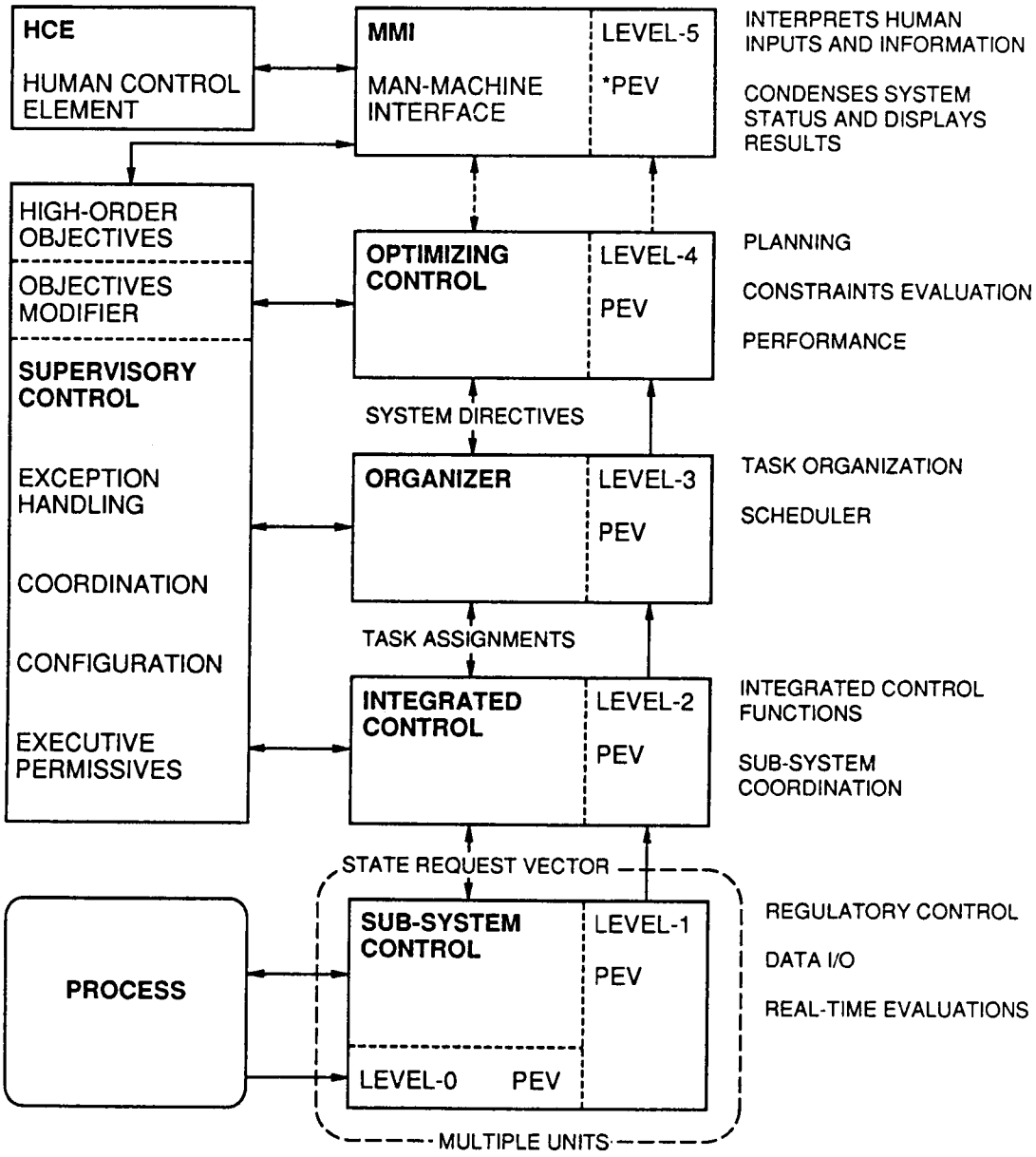
19. RESPONSE TO HV ARCING, STEADY-STATE CONDITION WITH MODE CONTROL 45
20. SIMULATION OF ACTIVATION, 50-ms DURATION 46
21. EMISSION, FILAMENT CURRENT RESPONSE WITH ACTIVATION 48
22. ACTIVATION RECOVERY SCHEME USING MODE SWITCHING 50
23. COMBINATION ARCING AND ACTIVATION WITHOUT RECOVERY SCHEME 52
24. COMBINATION ARCING AND ACTIVATION WITH FULL RECOVERY 53
25. COMBINATION ARCING AND ACTIVATION, GAUSSIAN NOISE WITH FULL RECOVERY 55

I. INTRODUCTION

Overview

Given a relatively complex, nonlinear, time-variant, multivariable process, the real-time decision-making process must include methods and procedures for handling all situations encountered with consistent, optimal, stable, and reliable control. The control system must handle not only the steady-state situation but also process exceptions, variability, and additional process information that is needed to handle process excursions. The automation of the electron beam welder (EBW) has several aspects of the nonlinear and nonstationary system previously described and requires that the controller be capable of making control decisions that will ensure that the final product, the welded material, will be of acceptable quality. The major subsystems of the EBW are: (1) the electron beam (EB) and controller; (2) the vision, image processing, and inspection system; (3) the robotics and motion system; and (4) the environment (vacuum) control system. To meet the objectives of the EBW, a central controller is needed that can make optimal decisions and also be able to receive information from the operator, interpret it, and incorporate it into the control situation. In particular, the control system is usually required to handle exceptions with minimal operator intervention; nevertheless, the operator is considered as a vital and necessary element within the control hierarchy. The operator, however, has limited closed-loop, continuous, active control capability. To better understand the qualitative and quantitative aspects of the required level of interaction with the operator, a hierarchical control architecture has been proposed and partially tested.

Figure 1 shows the overall architecture in terms of its major functional blocks and a partial description of the functions of each block. This architecture is useful in determining where to place the necessary control "intelligence" to accomplish



*PEV-PROCESS EVALUATOR

FIGURE 1
HIERARCHICAL CONTROL ARCHITECTURE

the objectives. The lower levels of the architecture are more closely related to the real-time operation of the process. For example, the lower levels within the control hierarchy may be dominated by closed-loop, set-point control where the main objective is to minimize the mean square error (MSE) of a single process variable and set-point reference. In contrast, a higher-level process control function may be dominated by planning and task organization; the control functions are not necessarily associated with specific hardware locations. Communications between levels are important but will not be addressed in this thesis.

Most complex decisions are resolved through the ordering of information as indicated by this proposed architecture. This is not the only acceptable model for complex control decisions, nor is it a complete model; rather, it is a workable model to be studied and analyzed. An architecture that solves all control situations is neither possible nor desirable. Solutions are based on the particular goals and needs of a given process; however, any workable control architecture does have distinguishable characteristics that can be used as reference and design tools when considering the general approach to complex and/or highly automated control problems.

One important observation of the architecture (Fig. 1) shows that the process evaluation is a systematic procedure that ensures that the level of control decisions matches the level of control objectives. The process evaluator (PEV) is the functional name for this part of the structure that concentrates data, with the appropriate analysis required to produce the process status for a given layer of the architecture. This will be explained and illustrated in later sections.

Another important area within the control hierarchy is the supervisory control (SC) functions. Some of the functions are exception handling, executive permissives, coordination, and interpretation of information from the operator. One of the most important SC functions, and one of the hardest to implement with full confidence, is the exception-handling feature. This feature monitors the process

and organizes other control functions such that the process objectives are being met, even when a portion of the process, subprocess, and/or instrumentation has failed or has been degraded. For example, if one of the process variables has momentary outages due to power loss caused by a power supply restart (arc recovery), the state controller may have to change to an alternate control mode and/or suspend a portion of the closed-loop control until the process variable can be restored. In other cases, the process variable can be estimated from other process data, and control is allowed to continue under these conditions until the process variable is restored. A simulation of this and other SC functions will be analyzed in this thesis.

The highly nonlinear and nonstationary nature of the process compounds the control problem, and consequences of these deviations from the linear, stationary case deserve special attention. The need for an "intelligent" controller typically increases as the process becomes more nonlinear and/or nonconsistent. The prediction of future states, based on present conditions, can be hindered and could require at least one of the following techniques for resolution: (1) real-time model verification, (2) real-time parameter estimations, (3) analysis of controller performance, (4) operator intervention under certain control situations, (5) complex modeling of the process, or (6) acceptance of degraded performance. A nonlinear process is used for the simulation of the hierarchical control problem analyzed in this thesis, namely the EBW.

This thesis does not describe a systematic method for characterizing a general class of complex control problems; however, the proposed hierarchy does fit many of the continuous and batch processes found in industry and could be expanded to include systematic approaches to multivariable, nonlinear, and nonstationary systems.

Applications

To better illustrate and evaluate the implementation of a hierarchical control architecture, a simulation of a nonlinear subprocess, requiring exception handling to maintain process performance, coupled with a multivariable process state model, will be evaluated. The control problem used to analyze the implementation of the proposed hierarchical control architecture is an automated EBW. The total problem requires several very sophisticated real-time analysis functions to ensure the quality of the final weld, including pattern recognition techniques, infrared imaging analysis, etc. The simulation will primarily concentrate on control and modeling of the EB gun and on prediction and real-time evaluation of EB gun process exceptions, based upon process data. The image processing functions are not simulated because this technology is available as a stand-alone system and will be modeled as a PEV element of the hierarchical control architecture. The process status, from the PEV, will be simulated as needed to illustrate its impact on the other control elements. Figure 2 shows a general overview and functional block diagram of a complete electron beam welding system (EBWS). Only a small portion of the overall system will be modeled, but the concepts could be applied when considering or designing a totally integrated system.

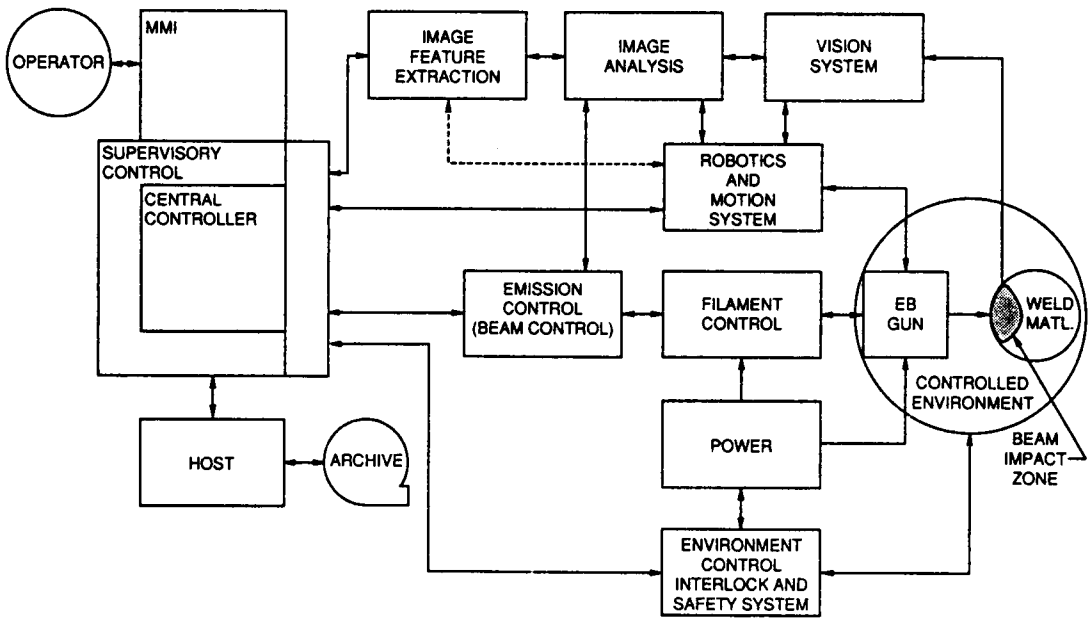


FIGURE 2
ELECTRON BEAM WELDING SYSTEM/SUBSYSTEMS

II. HIERARCHICAL CONTROL ARCHITECTURE

Description of Architecture

The purpose of an architectural outline can be twofold: (1) to organize information, control functions, and data such that the hardware aspects are emphasized and/or (2) to show the functional requirements as they relate to the decision hierarchy. The hierarchical architecture proposed here is intended to provide guidance of the second type and will be used as a model for decision classification. The other distinction of this architecture is the PEV hierarchical analysis structure.

The first step in attacking a complex problem, such as that of the EBWS, is to recognize that several objectives must be met and that the relationships among these objectives are not necessarily simple or obvious. It would be advantageous to have a systematic approach that could be applied at any level of the architecture and provide compatibility of information throughout the system. The following paragraphs will introduce a suggested format for this approach. At any level of the architecture, similar control processes must occur to accomplish all of the given objectives and to adhere to the constraints. Furthermore, ideally the objectives would be met in some optimal fashion. At least in principle, the following statement will be hypothesized: Given any physical system or process, the eminent domain to effect the state of the process (partially or fully), and the desire to impose goals or constraints, then the problem can be posed as an optimization problem. With this as an assumption, the following approach will be introduced. Figure 3 illustrates this decision model with the following elements: (1) objectives, (2) evaluation of the process state, (3) decision process, control law, etc., and (4) interaction with the process, including the ability to observe and modify the process. The PEV is given a special identity because of its equal importance to the

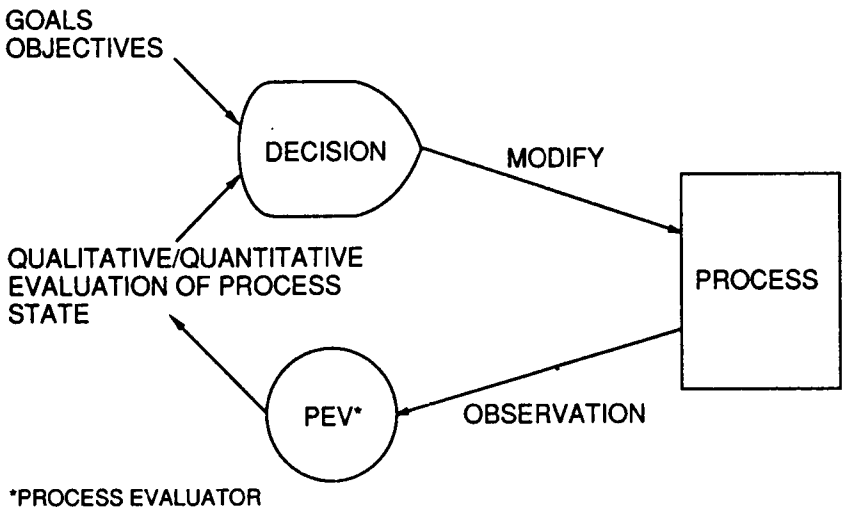


FIGURE 3
DECISION MODEL

decision hierarchy, and it also has a structure of hierarchical analysis. The PEV concentrates data and primarily provides the controller with evaluations of the process performance that can be related to the process goals and objectives. A couple of examples will help verify the concept. If, for example, at the lowest part of the architecture, there existed a single control variable that could be controlled by using a proportional-integral-derivative (PID) loop controller and if the process variables were observable, then the objective function might be viewed as the set-point demand, the PEV could be viewed as the measurement and conversion to engineering units of the process variable, and the decision could be viewed as the action of the PID loop controller upon the error signal (e.g., the error signal is defined as the set point minus the process variable.)

In a more complex situation, the objectives might be minimizing process cost, improving efficiency, etc. The PEV concept, in the context of optimal control, can be used as a reference model (real-time analysis) that predicts process performance. This model could include mathematical and procedural models, decision trees, information databases, and measurements from the process that would be used by the PEV to arrive at the performance estimate. The control decisions would be derived, either directly or heuristically, from the PEV estimates. The manipulation of the control variables would direct the process to the predicted optimal state.

Typically, for a complex, nonlinear process, the performance index cannot be expressed in quadratic form or numerically "closed form" bilinear equations, and furthermore it does not represent a "unique" solution to the optimization problem. The "qualitative" representation of the process performance and reasoning is not an "exact" science; however, certain methodologies can be used to describe workable "qualitative" spaces. A very good introduction to qualitative process (QP) theory is presented in *Qualitative Reasoning About Physical Systems* (Bobrow et al., 1985). Several of the analysis techniques described in QP theory are embodied within the concept and definition of the PEV.

Application to Electron Beam Welding

By using the architecture as a guide, the EBWS can be mapped into the control functions as indicated by the proposed architecture. The total solution to a fully automated EBWS is an ongoing development effort, and the system will probably not be totally automated within the next several years; however, great improvements can be made in performance, and the level of automation can be upgraded such that the EBWS will be reliable, flexible, and easy to operate. The architecture can give meaning and relevance to what would otherwise be a collection of very diverse subsystems, each with its own set of very complex problems. For example, the robotic vision and motion systems are in the infancy phase of development, with several key pattern and object recognition problems still remaining; also, the current state-of-the-art robotics and robotics vision technologies are not sufficiently developed to meet the requirements for the highly automated operations. Accepting these development limitations and assuming that improvements will be incremental, this thesis offers only a partial tool to accomplish one of these incremental steps.

The electron beam gun controller (EBGC) is a subsystem of the EBWS, and the same hierarchical approach used for the EBWS can also be applied to the EBGC. The EBGC has basically all of the control elements that the overall system has; for example, the EBGC has a supervisory control section, PEV structure, state controller, etc. In Sect. III, a model of the EB gun is developed that shows the dynamic behavior of the nonlinear filament-emission relationship. The model can be used to evaluate the performance of a given control philosophy, with special attention given to the simulation of process perturbations, excursions, etc. A good illustration of an application of the PEV in the implementation of the EBGC would be a PEV that would detect when process excursions have occurred, provide status information that can be used to ensure optimal recovery (from the excursion), and

provide information that would be used in quality control, quality assurance (QA), etc.

The usefulness of a systematic, unified approach for assimilating the primary control tasks, subsystems, and integrated control functions is the main reason for having, and adhering to, a systematic integration scheme.

III. MODELING OF NONLINEAR SUBSYSTEMS

Nonlinear Electron Emission Model

The EB current and power are crucial elements in the control scheme in terms of producing a good-quality, consistent weld. The area of beam impact, where the beam energy is concentrated and focused, will be referred to as the "heat-affected zone" or "fusion zone." The beam power, in effect, determines the weld temperature and the temperature distribution at the fusion zone, both of which affect the metallurgic and physical aspects of the final weld. Depending on the size of the part, the material(s), and the speed at which the weld is being made, a relatively narrow band of acceptable beam currents and beam powers will accomplish the desired results. Any instabilities and/or perturbations within the EBGC can therefore have a profound effect on the weld quality. Two of the most common perturbations that can occur in the EBGC are (1) the loss of high-voltage bias, caused by arcing, within the EB gun and (2) filament[1] activations. Because of impurities, background and residual gases, micro-imperfections on the metallic surfaces, dirt, foreign material in the vacuum vessel, etc., there will be occasional arcing. Sometimes the first arc will dislodge impurities, internal gasses, etc., which triggers a sequence of arcs. Depending on the duration of the arc sequence, the temperature of molten metal (at the fusion zone) may drop enough to cause serious imperfections in the weld. Since the arcing can never be totally eliminated, the methodology of the controller should at least: (1) minimize the arc's effects and causes, (2) attempt compensation, if possible, and (3) alert the operator, central controller, etc., that an excursion has occurred and describe the location and severity of the possible flaw. Records of events that occurred during the excursion could be used for quality control and specialized inspection by demand (e.g.,

[1] Filament activation is sometimes referred to as filament "poisoning."

when an excursion occurs). "Development and Application of High Power Electron Beam Welding" (Kita et al., 1986) contains a classification scheme for arcing that is useful when characterizing the type, and source, of the most common arc discharges, namely: (1) micro-arcing, partial capacitor (i.e., the high-voltage power supply filtering capacitors) discharges, (2) single arcing, full capacitor discharge, (3) multi-arcing, a sequence of single arcs (can be initiated by single arcs), and (4) soft vacuum discharges, caused by local ionization and partially, or fully, sustained current conduction (can be initiated by multiple arcs).

In the case of filament activation, this is a less-frequent event than gun arcing; however, activation represents a major perturbation in the system in terms of the normal filament-to-emission relationship. Gun activation is generally caused by impurities within the system that become ionized directly upon impact with the EB. Once ionized, these particles can be accelerated toward the gun cathode and can be deposited directly upon the filament. Typically, the work function (of the foreign material) will be different than that of the cathode and can cause a significant change in the current density at the area where the impurities have lodged. See the Appendix for a short description of the relationship between thermionic emission current density and work function. The Richard-Dushman equation (see the Appendix) clearly shows the direct dependency between work function and current density. When an activation occurs, the relationship between filament current and emission current will be detected by the EBGC and compensation will occur. Typically, the activation will not remain for any appreciable time because the filament is typically in the range of 2500 K, and for most materials at that temperature the vapor pressure will cause a fairly fast dissipation. There are cases where the activation could remain considerably longer due to oxide deposition and/or chemical reaction with the filament; however, this type of activation is not as common and will not be addressed in this thesis. For classification purposes, activations will fall into one of two categories: (1) short-term activation,

lasting less than 10 s, and (2) long-term activation, lasting more than 10 s. As stated, only short-term activations will be addressed in this thesis; however, the techniques for detecting an activated filament will be applicable for both cases. Typically, once an activation is detected, it is treated as a class 1 activation; after repeated unsuccessful attempts to clear the activation, it will then be declared a class 2 activation. The steps required to clear the more severe activation usually require that the high-voltage bias be removed (i.e., by turning off the beam) and that the filament temperature be raised to enhance the evaporation of the foreign material; otherwise, the filament may have to be replaced, etc. Although the class 2 activation will not be simulated here, a complete control philosophy for the EBGC should include class 2 condition handling.

The problems that result from the short-term activations are not as evident but can cause micro-exclusions that may not be detected until the weld process has been completed and the post-inspection has been performed. Additionally, a short-term activation could cause a marginally stable system to fluctuate and/or experience a "ringing," which would affect the beam quality, position, etc.

Considering these types of process exceptions, it is necessary to have the EBGC respond (correctly) with minimum impact to the weld quality. To facilitate the design of a robust controller, an energy-balance, dynamic model was constructed for use in developing methodologies that handle the wide variety of process exceptions.

Figure 4 shows a typical gun configuration used for EBGC system. The model was constructed to accommodate the most general case, to allow for specification of key parameters (such as filament type, filament dimensions, etc.), and to ensure that the model would exhibit good closure with measured EB gun data (e.g., dynamic and steady-state). The model does not estimate or optimize the beam characteristics at the fusion zone. In a full EBW design, the characteristics of the beam at the fusion zone would be of prime importance. This type of modeling

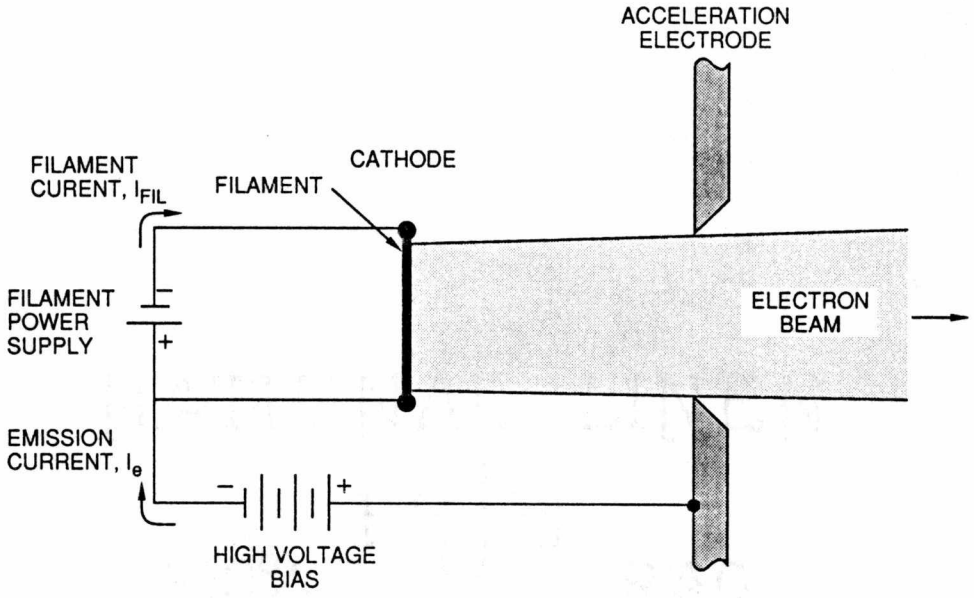


FIGURE 4
EB GUN FILAMENT DIAGRAM

could be done and coupled to the gun emission model, but it would require calculations of electron trajectories, beam divergence, electron energy distributions, etc. As stated, the gun model is primarily used to address issues directly related to the EBGC; however, activations and arcing do have a direct impact on the beam divergence, focus (spot size), thermal aberrations, beam position, etc. Therefore, the control philosophy of the EBGC can be primarily directed at minimizing the immediate effects and still be consistent with the system philosophy.

The first steps in developing the electron emission model were to identify the determining equations from the partial differential heat transfer equations, to make realistic assumptions about the boundary conditions, and to build the model at the lowest dimension possible and still obtain a good representation. The assumptions are as follows:

1. the gun filament is housed within a vacuum environment and is maintained at a pressure of at least 10^{-4} torr (torr is defined as 1/760 of a standard atmosphere),
2. the filament is the cathode (e.g., the electron-emitting surface),
3. the temperature of the filament and the surface area of the filament primarily determine the electron emission current,
4. the gun is operated in a thermionic emission limited mode (space charge limitation is relatively small),
5. the primary heat loss from the filament is from radiant heat loss (conduction loss is relatively small),
6. the filament cathode if assumed to be almost fully enclosed by a constant temperature enclosure,
7. the filament is heated directly by the filament current (e.g., resistive heating), and
8. the temperature distribution along the filament is relatively constant.

Based on these assumptions, it is possible to formulate a one-dimensional model. A two-dimensional finite-difference model would be necessary if not for assumption 8; however, that level of detail is not directly needed for the EBGC and exception handling. One additional heating phenomenon included in the one-dimensional model is not inherent in this model but is inherent in the two-dimensional case: the filament heating due to the emission current. To accommodate this heating effect, the average current increase is assumed to be equal to half of the total emission current. This average is calculated and added to the filament current heating term (i.e., the I^2R term) as needed by the heat balance calculation. The heat balance terms are (1) resistive heat, (2) radiant heat loss, and (3) conductive heat loss. The resistive heat is produced from the filament current; however, the model will also have the capability of specifying filament voltage as the forcing function. The filament current will then be calculated from the resistance of the filament and used for the normal I^2R heating calculation. The radiant heat loss is determined by the temperature of the filament and the temperature of the background that the surface is viewing. The view factor for the radiant loss term will not have to be calculated from the EB gun geometry since the filament cathode is assumed to be almost fully enclosed (assumption 6).

The differential heat equation, including conduction, is

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \dot{q} = \frac{\partial}{\partial t}(\rho C_p T) \quad , \tag{1}$$

where ρ = density, C_p = specific heat, and T = temperature in Kelvin.

By using assumption 8, the conduction term can be set to zero:

$$\frac{\partial T}{\partial x} = 0 \quad . \tag{2}$$

The resulting equation now becomes

$$\dot{q} = \frac{\partial}{\partial t}(\rho C_p T) \quad . \tag{3}$$

Both the density and specific heat are functions of temperature. The specific heat has a relatively stronger dependence than does the density. Since the density

remains fairly constant with temperature, the specific heat will be calculated, based on the current temperature estimate, for each iteration.

If these assumptions are used and if positive heat is defined as heat gain, the heat generation term, $\dot{q}$, can be written as:

$$\dot{q} = q_{I^2R} - q_{\text{rad}} - q_{\text{cond}} \quad . \tag{6}$$

Calculations will be done for a unit length (unit segment) as shown by Figs. 5a and 5b. The total resistance will be calculated by summing all of the segments. Actually, the overall filament can be considered as a single segment having the general shape as depicted in Fig. 5c, where d and w are specified. Note that, if d and w are equal, then the geometry is a circular filament.

To calculate $\dot{q}$, the physical properties of the filament that are dependent on temperature must be estimated. Since data of this type are easily found in tabular form and because highest accuracy is a primary concern, particularly during the first stage of the modeling effort, a cubic spline method will be used to estimate these properties.

To expand $\dot{q}$:

$$q_{I^2R} = (\text{filament current}) \cdot R \quad , \tag{7}$$

where

R = resistivity $\cdot\, l/A_c$,

l = filament length,

A_c = cross sectional area of filament,

$$q_{\text{rad}} = A_s \cdot \sigma \cdot \epsilon \cdot (T^4 - T_{\text{ref}}^4) \quad , \tag{8}$$

where ϵ, emissivity, is a function of temperature, A_s is surface area, and T_{ref} is temperature reference.

$$q_{\text{cond}} = k_1 \cdot (T - T_{\text{ref}}) \tag{9}$$

where k_1 is constant and depends on the specific gun design; it is used to model the heat loss via the filament support and electrical connections and can be used to

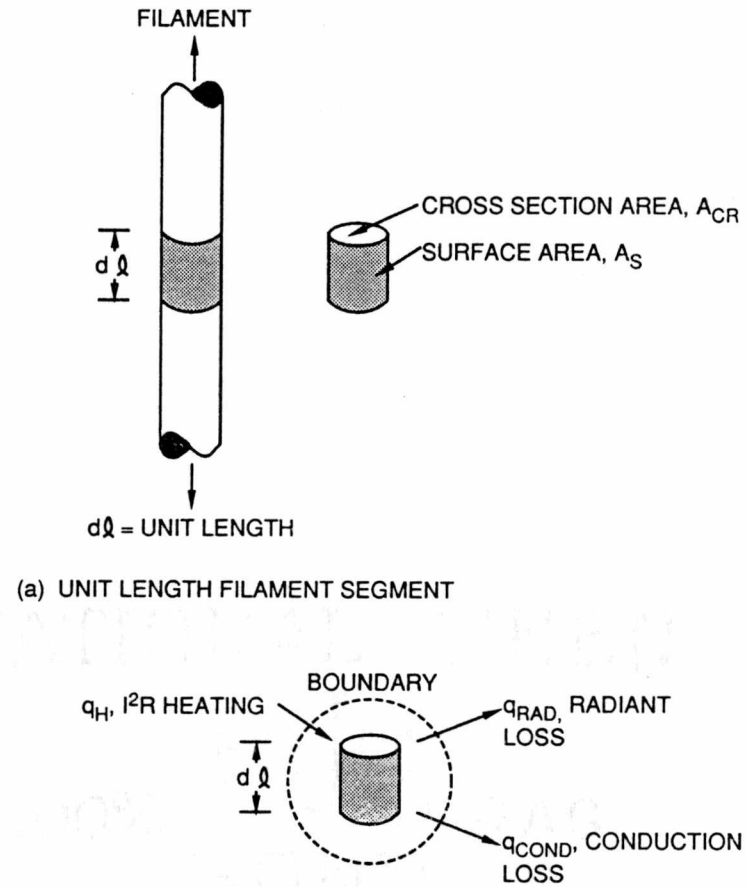


FIGURE 5
DIAGRAM OF FILAMENT

model the heat loss via the convection losses (e.g., in the event of high background pressure).

Converting Eq. (3) into a backward difference equation and solving for $T(n+1)$ results in the following equation:

$$T(n+1) = T(n) + (\dot{q} \cdot \Delta t)/(\rho \cdot V_p \cdot C_p) \quad ,$$

where $T(n+1)$ and $T(n)$ are the filament temperatures at time steps $n+1$ and n respectively, ρ is the density of the filament temperature, V_p is the volume per unit length, C_p is the specific heat at constant pressure, and Δt is the differential time.

Since the difference equation has several nonlinear temperature dependencies, the average temperature over one time step will be used for the calculation of temperature-dependent properties:

$$\overline{T} = [T(n+1) + T(n)]/2 \quad .$$

Therefore, emissivity, resistivity, and specific heat will be calculated using $\overline{T}$.

Figure 6 shows the flow diagram used to solve the difference equation and to calculate the filament and emission currents. Each time step must attain an energy balance convergence before continuing to the next time step so that the nonlinear effects will be not be propagated outside the time step. In other words, the energy balance terms must converge before continuing to the next time step.

After the temperature of the filament is estimated, the Richard-Dushman equation can be used to estimate the total emission. The model is constructed to utilize a cubic spline, if tabular data are used for the emission current density versus temperature. The Sckottky effect is not included in these calculations because of its relatively small effect; however, the effective work function reduction could easily be calculated and included, if secondary effects are being considered. The Sckottky effect is based on the increase of emission current density due to the electric field present at the surface of the cathode. The magnitude of the normal field gradient is included in the Richard-Dushman equation and is described in the Appendix.

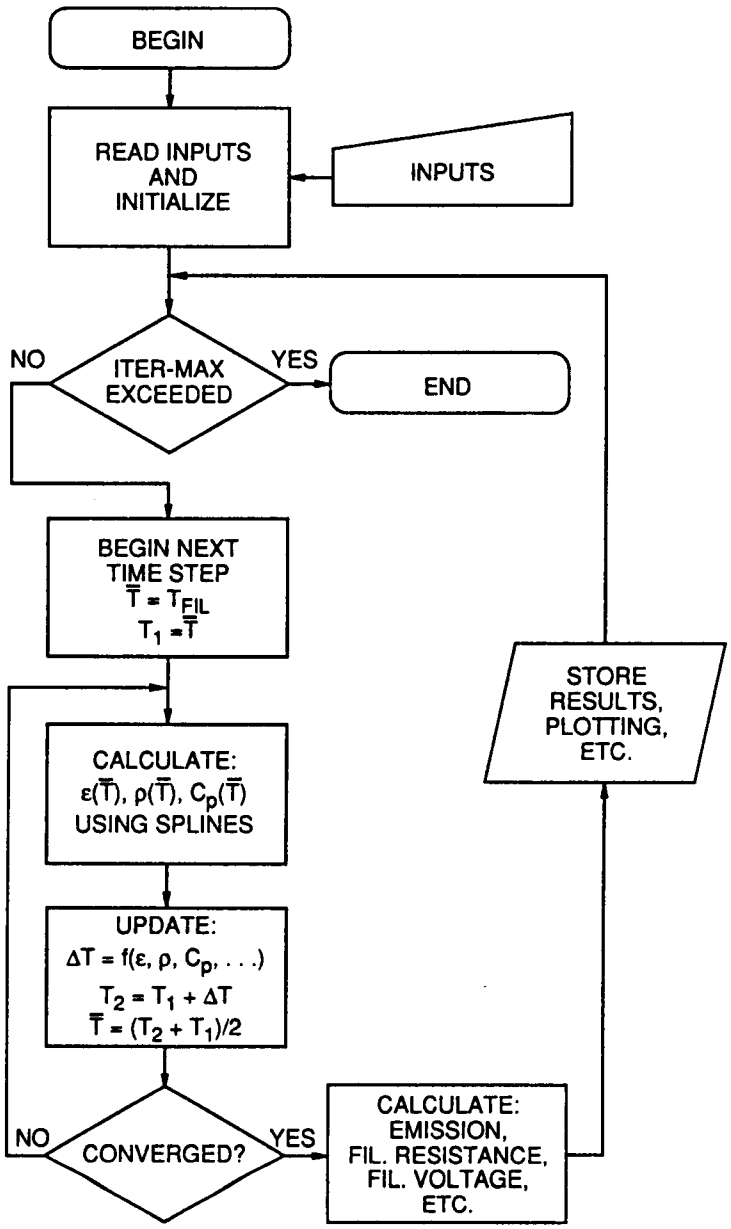


FIGURE 6
DIAGRAM FOR THERMAL MODEL

Once the emission model was completed and tested, a closed-loop system was simulated by using the widely accepted PID loop controller to control the filament current. The emission control unit, which also uses a cascaded PID loop, was also included in the model. To achieve and verify model closure, the thermal/emission model was tested in the open-loop configuration, using the filament current as the forcing function. The filament voltage forcing function was needed so that the filament step-down transformer could be modeled as a two-port network. The filament voltage is not typically used as the controlled or set-point variable because a step voltage across a cool filament will result in a very high filament current. The cool filament has a low filament resistance at low temperatures (e.g., below 1000 K). The filament current forcing function offers a more stable and predictable transfer relationship between input and output. The main objective is to maintain a constant emission current with the least susceptibility to variations in process parameters, operational demands, environmental perturbations, etc. The filament current controller works well for this purpose and, typically, is a stand-alone controller that is implemented with only conventional analog circuitry.

Activations are simulated by changing the work function and Richardson coefficient of the Richard-Dushman equation. The simulation of the activation is done in three parts: (1) initial deposition of the foreign material, (2) the coated state, the sustained emission from the surface of the foreign material, and (3) the evaporation stage, where the material has begun to deplete and the emission current density has begun to return to the previous value. To model the activation process, step 1 is modeled simply by instantaneously changing the work function as the activation begins; step 2 is modeled by sustaining the new values of the work function and Richardson coefficient for a predetermined amount of time (i.e., time of suspension is related to amount of material deposited); and step 3 is modeled by an exponential decay rate for emission current difference. The activation simulation will require the specification of starting time, duration time, decay rate, and

peak emission current multiplication factor (i.e., the ratio of the normal expected emission current to the activated filament current). Typically, the activation current is greater (multiplication factor greater than one) than the normal emission current before activation. The change in the localized current density is, of course, based on the new work function and Richardson coefficient for that material; however, the area of the activation and the mass of the foreign material cannot be measured in real time. Since the exact characteristics of the complex physical phenomena of an activation cannot be quantified and the activation operates in a random, highly unpredictable manner, the multiplication factor will be sufficient to study the effects of activation. As stated, real-time observation of the dynamics of the activation can be done only through the normal measurement and characterization of the filament current versus emission current. Multiplication factors are chosen to simulate typical metallic or oxide material that would normally be present in the EBWS, with a relatively small area of the filament area actually being coated with the foreign material. Most EB guns are designed to minimize the back-streaming of ions and to ensure that they are not likely to impact the filament; however, since activation cannot be totally eliminated, it is important that the EBGC be able to handle this type of exception.

The simulation of an arc is accomplished by setting the emission current to zero while maintaining all other thermal calculations. The high-voltage power supply (HVPS) usually has arc protection, delay, and recovery circuits. The HVPS will detect an arc by sensing an arc surge (current), which will trigger the arc recovery circuitry and immediately turn off the HVPS. The HVPS remains off for a predetermined amount of time (typically for a few milliseconds) and then turns back on and continues to check for current surges. This arc suppression sequence is primarily designed to protect the HVPS and not designed for optimal EB recovery. The HVPS cut-back time is important because it allows the arc-generated ions to dissipate so that the EB can be returned to its previous power level. If the EB

were turned on too soon after the initial arc, the localized ions could initiate the type 4 arc.

Electron Beam Gun Controller Simulation

Once the general numeric structure of the emission model was established, several test cases were constructed to simulate closed-loop performance, first without arcing and/or activations and then with the addition of process exceptions. The supervisory and PEV functions are utilized to provide limited exception handling for the EBGC system.

Figures 7 and 8 show the strong dependence on emission dynamics and filament size. The temperature of the filament is initially set to 2700 K (higher than the normal operating range but used to show the full dynamic range) and is allowed to cool radiantly; Fig. 8 shows the (log base 10) emission current versus time. Since the dynamics of the filament emission are so greatly dependent on the radiant cooling of the filament and the filament size, a circular filament with a diameter of 0.1 cm will be used for all subsystem comparisons. The filament length will be selected and fixed at 1 cm, and this value will be used for all of the subsystem evaluations so that meaningful comparisons can be made. Additionally, the filament emission will be based on the current density expected from a pure tungsten filament cathode. Beam steering and focus control are not considered in the model. The total calculated emission current will be used; however, most gun designs will not emit the theoretical maximum due to a variety of limitations (see Chapter 8 in Pierce, 1954). Basic EB gun parameters versus geometry are well described in Pierce's book (a classic theoretical reference to EB design considerations). The actual beam current in most systems will be approximately half of the calculated values in this thesis (with the same filament dimensions and filament current); however, without a full model and description of the electron optics,

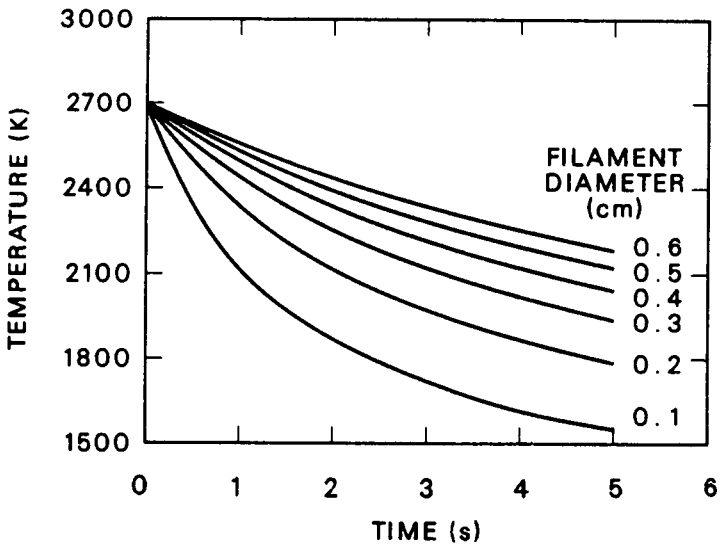


FIGURE 7
RADIANT COOLING OF FILAMENT

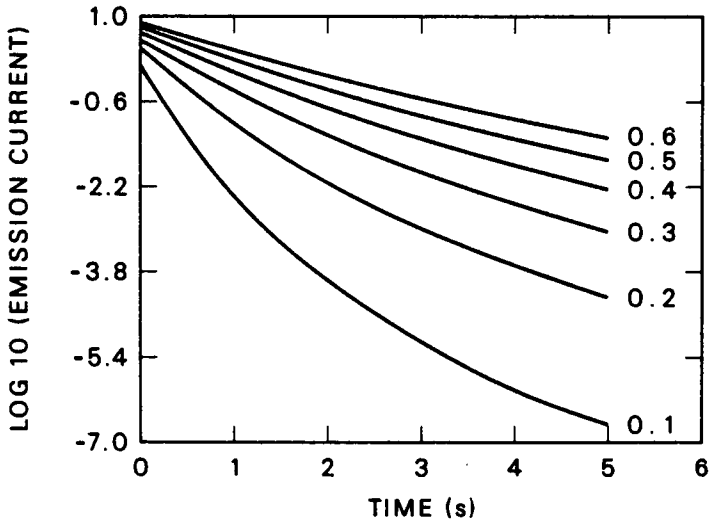


FIGURE 8
EMISSION CURRENT (EB GUN CURRENT) WITH
RADIANT COOLING OF FILAMENT

etc., an emission model is all that is necessary to model the system in terms of its measured values. The filament is assumed to be a pure tungsten filament cathode with direct filament heating and low heat loss due to the filament support (and electrical connections, etc.). Model closure of the thermal model and comparisons to theoretical emission current dynamics are discussed in the Appendix.

Figure 9 shows the equivalent diagrams of each element of the filament controller that will be used for the simulation. Typical control elements would include: (1) the power electronics necessary to convert the electrical AC power (e.g., 240-V single-phase or 480-V three-phase step-down transformer) to an acceptable, high-current, AC power; (2) a rectifier circuit; and (3) a smoothing filter. A single fixed-parameter, dominant-single-pole transfer function is used with a voltage-dependent current source and effective output impedance. Specific values for the filament controller are listed in the Appendix.

The filament controller model was constructed and run for several conditions and PID tuning parameters. The overshoot in the emission must be minimized because the emission current is proportional to the weld power and because over-shoots in emission current could cause the HVPS to detect an over-current or arc condition. In other words, if the HVPS detects a peak over-current and cuts back the voltage, this would cause an unnecessary perturbation to the system and could cause the HVPS to repeatedly cycle in this (false arc) condition. Figure 10 shows a relatively stable filament current loop that would normally be considered properly tuned. For example, the mean square error (MSE) would be minimized, etc.; however, the filament over-shoot causes a very large, and unacceptable, emission current over-shoot. The amplified and delayed emission current is caused by the very nonlinear thermionic emission phenomena. This causes a slight dilemma: when using only a fixed-variable PID loop, the effective emission current delay, dynamics, and loop gain are changing with the operating conditions. Furthermore, closed-loop performance is optimized only to a narrow band of emission

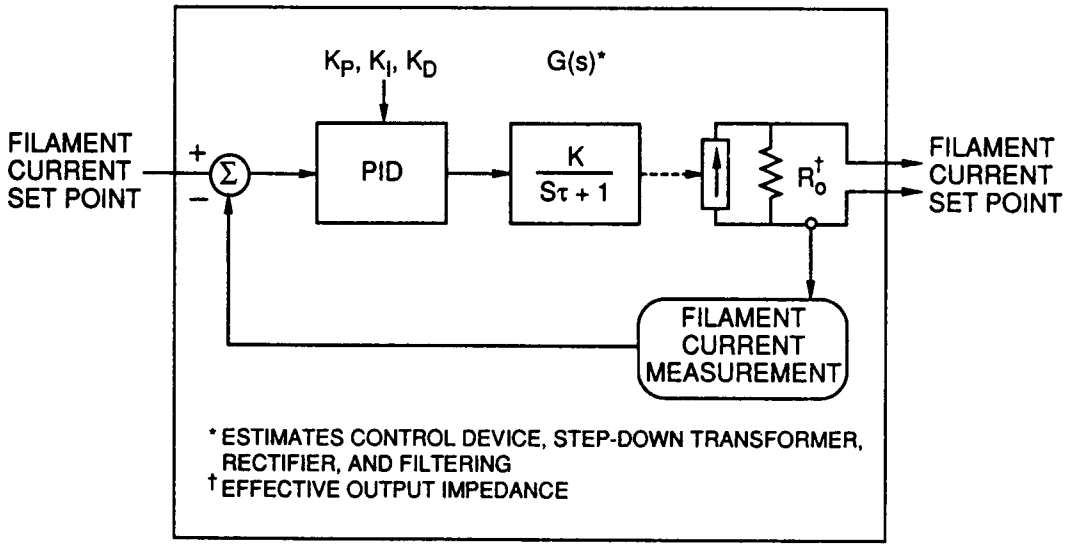


FIGURE 9
BLOCK DIAGRAM OF FILAMENT CONTROLLER

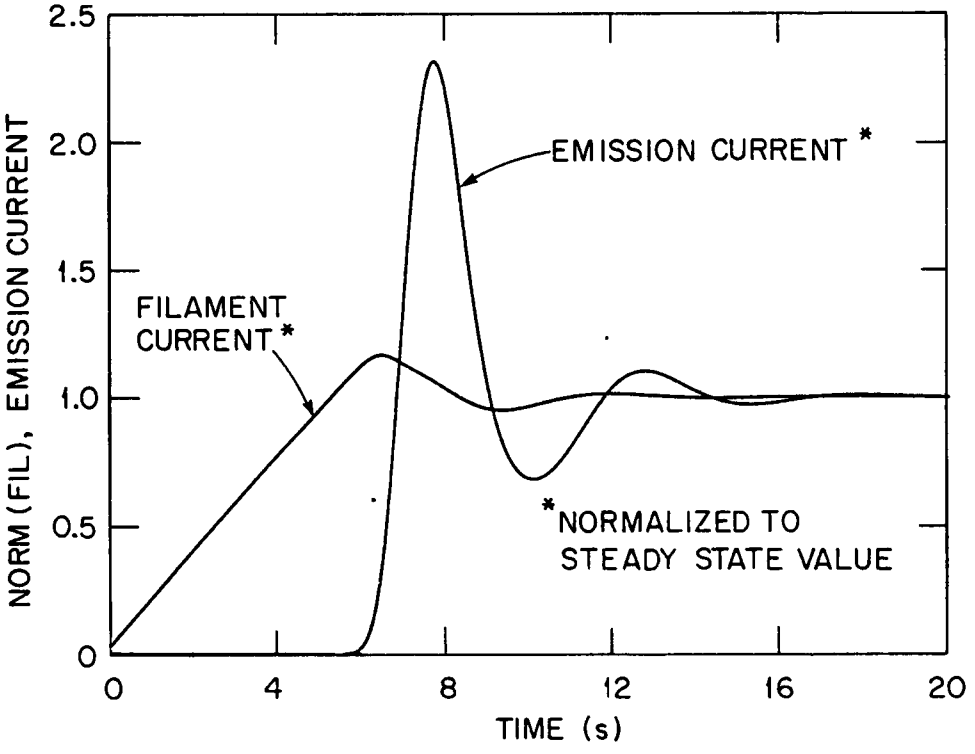


FIGURE 10
EMISSION, FILAMENT CURRENT STEP RESPONSE
WITH FILAMENT PID NOT PROPERLY TUNED

currents. There are several solutions to this dilemma, however; the "mode" control technique will be used for improved system performance and will be discussed in another section.

Use of a standard commercial PID loop controller, without optimization or self-tuning, is assumed in the model. Figure 11 shows the step response to the filament controller with the PID parameters that will be used throughout the rest of this thesis. Two control modes will be considered: filament mode and emission mode. The filament mode assumes that the filament current is specified by the operator or host computer and does not consider the state of the emission. This mode is used for diagnostics, manual control, and/or start-up control.[2] The emission mode is used to control the beam current (and power).

Figures 12 and 13 show the complete EBGC system. Typically, the filament controller, Fig. 12, would be implemented with strictly analog components and the emission controller would be implemented with primarily digital components (i.e., stand-alone microcomputer,.etc.). Figure 13 shows a diagram of the closed-loop EBGC and the equivalent functions during "normal" operation. During exception handling, the EBGC would suspend the PID control actions and execute special control actions to handle exceptions. The model includes all of the control elements in Fig. 13. Control actions, process signals, and exception handling will be simulated as needed. The emission controller, Fig. 12, is assumed to have the necessary hardware (inputs/outputs) to provide the set point for the filament controller and HVPS. In addition, the HVPS signals the EBGC that an arc has occurred and that the HV is being cut back. The "arc detect" signal (Fig. 12) is assumed to be a binary signal that has a relatively fast response, as compared with the cutback time. The emission current is assumed to be measured from the

[2]Start-up control refers to the initial filament heating and out-gassing of the EB gun and the slow "ramping up" of the HV bias to condition the system. Typically, a high arc rate is present before the out-gassing is complete.

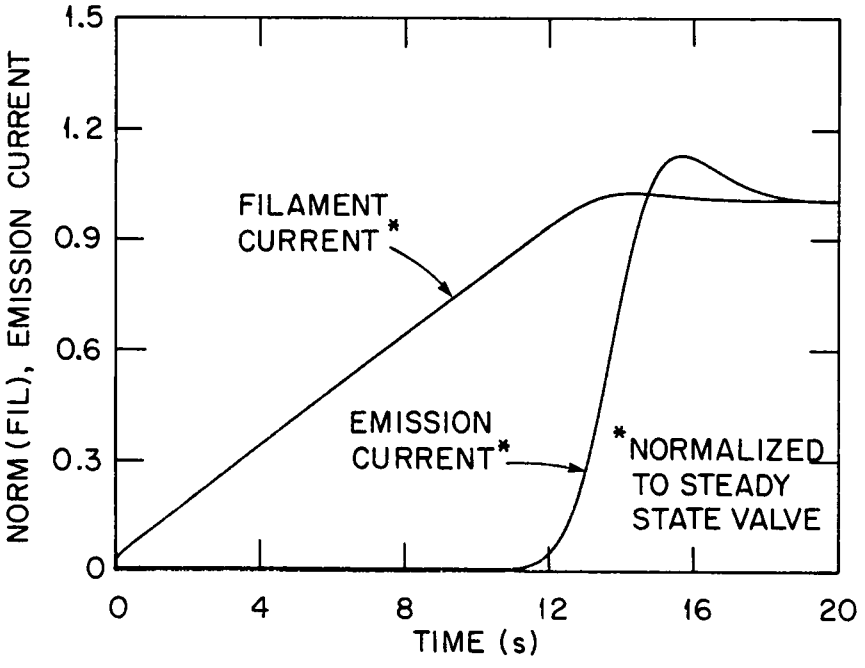


FIGURE 11
EMISSION, FILAMENT CURRENT STEP RESPONSE
WITH BOTH PIDs PROPERLY TUNED

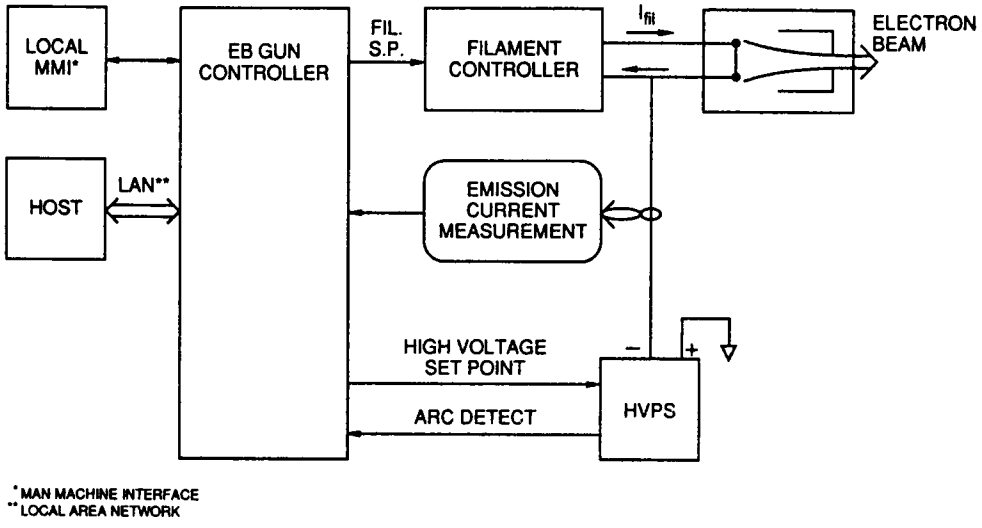
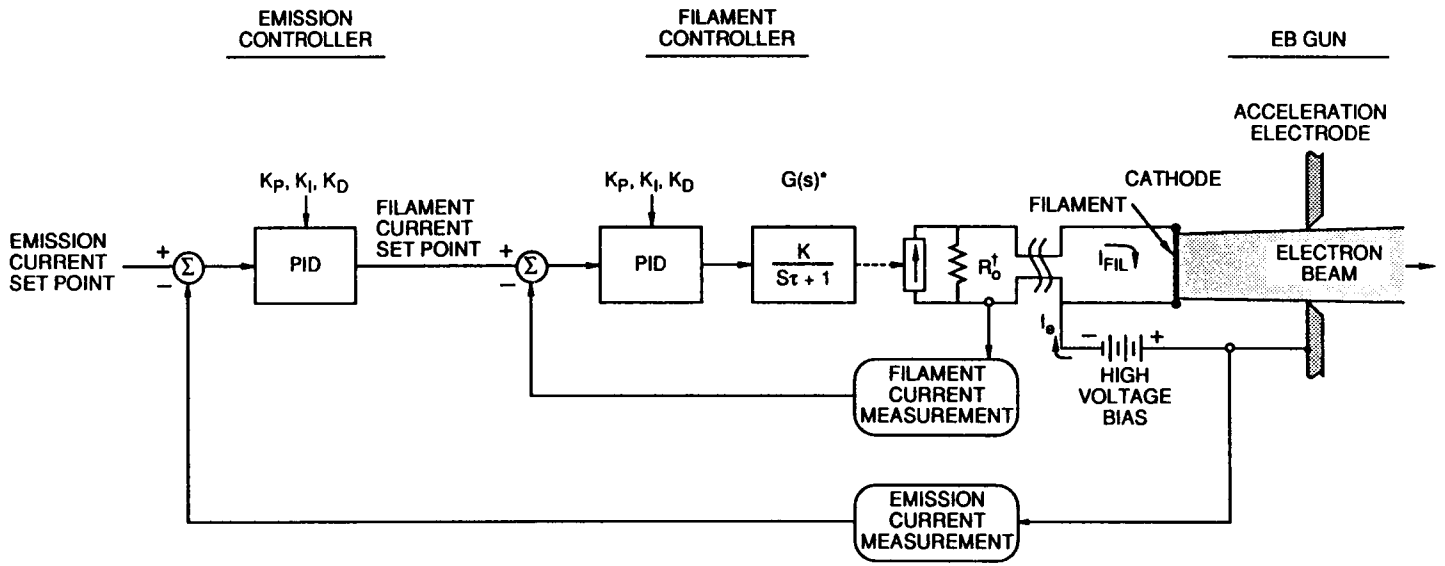


FIGURE 12
BLOCK DIAGRAM OF ELECTRON BEAM GUN CONTROLLER



*ESTIMATES CONTROL DEVICE, STEP-DOWN TRANSFORMER, RECTIFIER, AND FILTERING
†EFFECTIVE OUTPUT IMPEDANCE

FIGURE 13
BLOCK DIAGRAM OF CLOSED-LOOP EBGC

current of the HV bias supply. The filament current supply is assumed to have negligible leakage current and to have sufficient HV isolation at the output so that the emission current is essentially equal to the HVPS bias current.

IV. ANALYSIS OF SUBSYSTEM PERFORMANCE

Exception Handling

Once the closed-loop EBGC was constructed, the ability to handle process exceptions was considered. As stated earlier, the two process exceptions to be considered here are HV arcing and filament activation. The first step was to consider the necessary decisions and control actions that are required to give the maximum probability of success in meeting the objectives (Fig. 3). The objectives have been stated earlier: "to respond to process exceptions with minimal impact to the quality of the weld." This requires a good bit of qualitative reasoning on the part of the designer, but it can be an important aspect when considering a complex control problem where classical and/or deterministic methods are inadequate to handle process exceptions. The usefulness of any method and/or algorithm will be judged by the consistent adherence to the objectives. The PEV can be a single element or multiple units, but its primary purposes are to give relevance to the observations of the process, with respect to the objectives, and to accommodate a predetermined decision process that will compare and/or use the evaluation of the PEV to make the "optimal" decision. In the case of the EBGC, a PEV can be constructed to give status information and detection of the following: (1) normal emission mode, (2) normal filament mode, (3) HV arcing or instrumentation failure, (4) filament activation, and (5) general detection of instrumentation failure and/or fault. The methods for accomplishing these real-time evaluations are numerous and limited only by the imagination; however, the designer typically finds that a single format for representing all of the "knowledge" about the process is inadequate. For example, a database could contain sets of process patterns that were gathered as typical operating conditions but that would not necessarily give any ability to extrapolate outside the normal operating ranges, especially during

an excursion. A mathematical model of the process could be constructed, but there again, this method would not be the best method for containing operational procedures, etc. In other words, several methods of knowledge capture may be necessary to contain the diverse knowledge that is usually required when handling a complex process excursion.

Consider the diagram in Fig. 14 as a method to represent the normal, steady-state, operational regions. For example, if the system is stable, the filament current and emission can be compared with the diagram to establish the initial status of the EBGC. The "normal emission region" could be established from real-time parameter estimation or regression methods. For example, a predefined mathematical model could be formulated from first principles, and the unknown parameters/constants could be estimated from process data. With this model as a reference, this method is very useful in determining when an activation has occurred. If the model is a dynamic model, the requirement that the system be at "steady state" would not be necessary. A dynamic model can be used to compensate and decouple the dynamics.

Once an estimation model has been formulated, the detection of the steady-state condition is possible, and the application of the two methods (steady-state reference shown in Fig. 14 and dynamic model) gives a "robust" prediction. A PEV that uses this methodology was constructed and tested; it was successful in predicting the "correct" status of the EB gun and was used to correctly determine when excursions had occurred.

If the system equations can be estimated in the form $y(t) \quad f(y,\ x)$, then the system can be written as a set of linear differential equations (i.e., if the functions are continuous, etc.), even though x may be nonlinear with respect to y. *Theory and Practice of Recursive Identification* (Ljung and Soderstrom, 1986) is a good resource and programming guide for recursive parameter estimation techniques. There is neither time nor space to elaborate on these methods; however, the re-

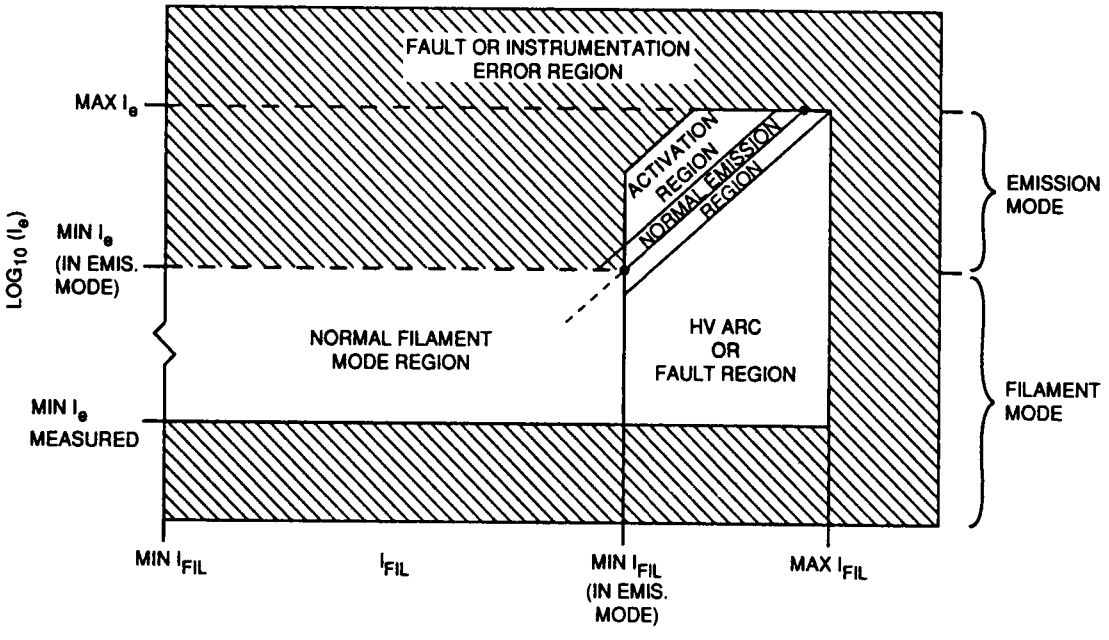


FIGURE 14
REGION EXCLUSION ERROR DETECTION AND MODE CONTROL

cursive least squares (RLS) method can be used to estimate the emission current versus filament difference equation, which could be used to establish the "normal emission region" (Fig. 14), and, furthermore, to provide an empirical, dynamic model of the process. One important point must be made about the utilization of the RLS algorithm: if an excursion occurs during the estimation of the parameters, the RLS algorithm will estimate erroneous process parameters. The usefulness of the PEV concept is again illustrated. The RLS method requires special handling during a process excursion; otherwise, the estimates will be corrupted and/or the RLS algorithm will become divergent (e.g., the covariance matrix will become ill-conditioned).

The principle is the same: there must be a distinction and detection of process excursions, and there must be adequate supervisory control to halt, modify, and/or change control strategies, in real time, to handle the process excursion. Figures 15a and 15b show how the RLS prediction responds to an HV arc situation. Note that the RLS algorithm converges very rapidly, even in the presence of significant process noise, but when the arc occurs, the estimates quickly become corrupted. As stated before, the algorithm can become divergent and/or numerically unstable if it is continually allowed to to iterate during the process exception. Recognizing and designing for process exceptions represent the greatest challenges to the design engineer.

The HV arcing, process perturbation occurs randomly during process operation and causes the emission current to be extinguished. Figure 16 shows a typical response of the closed-loop emission current during an arc sequence. The emission current, as shown in the plot, is quickly extinguished when the HV bias is removed (e.g., during the HVPS cutback) and resumes when the HV is restored. A scheme is developed and implemented that will minimize the impact on the filament and emission currents and minimize the recovery time.

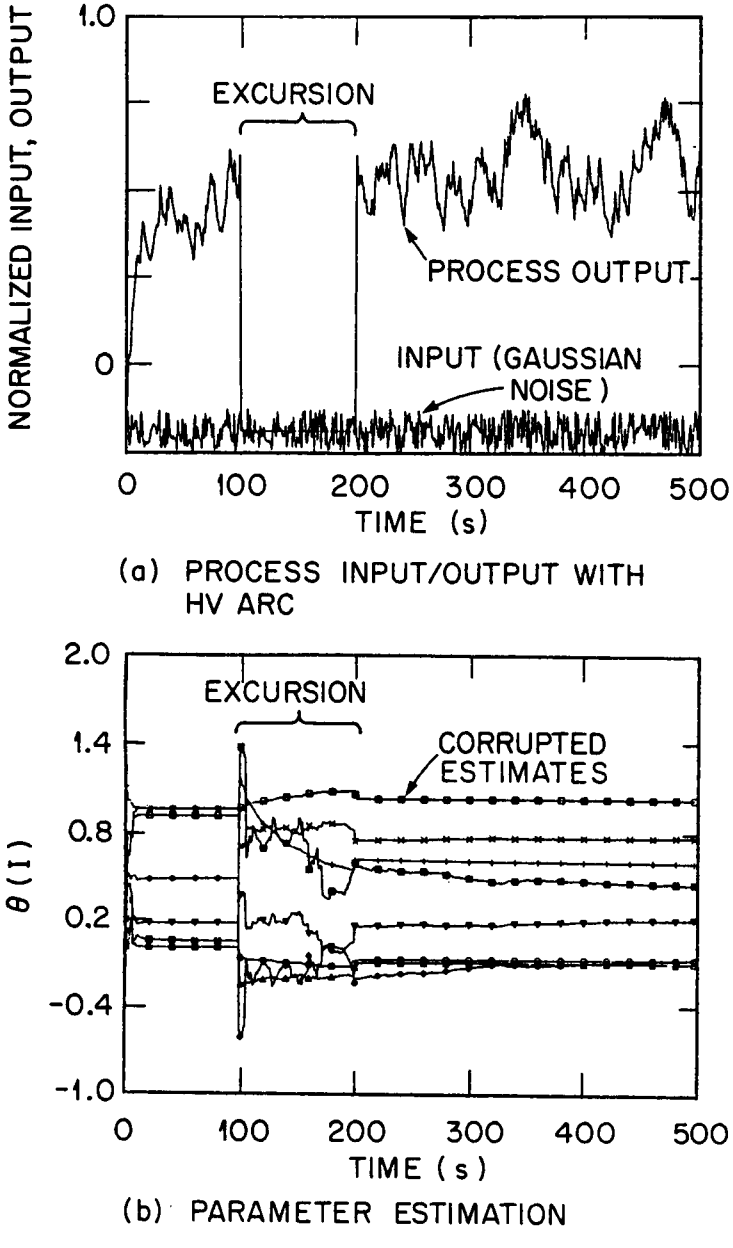


FIGURE 15
EXEMPLARY RESULTS OF RLS ALGORITHM WITH PROCESS EXCEPTION

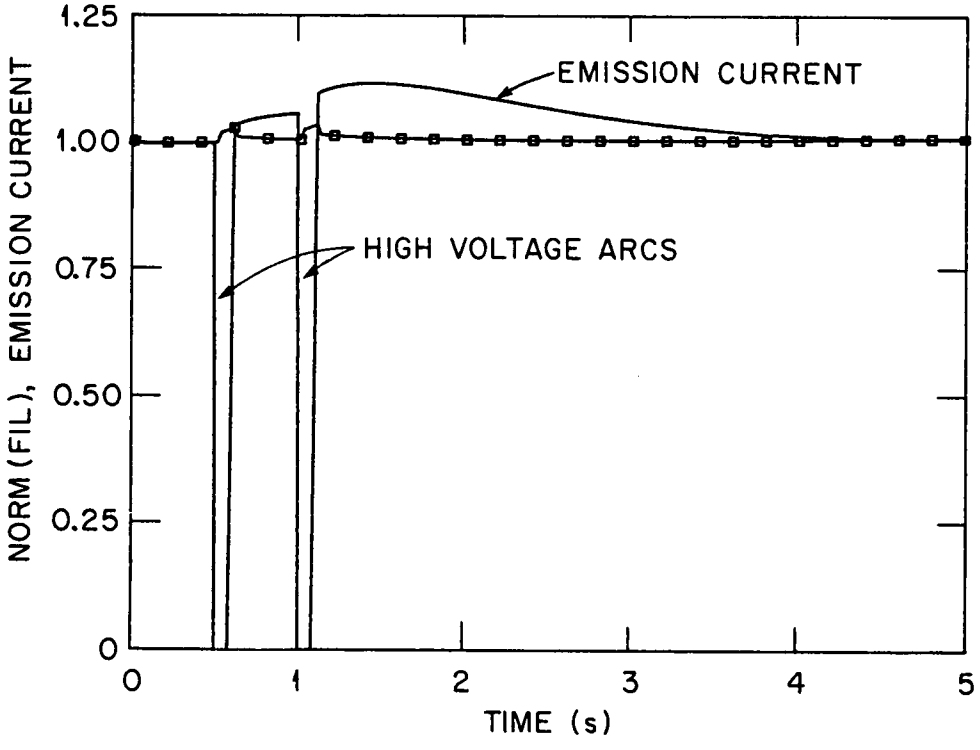


FIGURE 16
RESPONSE TO HV ARCING, STEADY-STATE CONDITION

Activation recovery is also addressed, as is the special case of both activation and arcing occurring at the same time. The next section gives the results and implementation of the proposed recovery schemes. The principle is as follows: (1) identify the regions of normal operation, (2) determine and define the regions of known process excursions, process exceptions, and/or faults, (3) develop a reliable means of detecting the exception regions, (4) determine the proper control actions and procedures to recover from these exceptions, and (5) integrate these control actions into the control structure. Each of the steps described is used to arrive at workable solutions to the arcing and activation process exceptions.

Performance Evaluation

Using the diagram in Fig. 14, a mode control scheme was applied to the closed-loop EBGC system. As stated before, the PID loop controller has several inadequacies when implemented with fixed tuning parameters; however, a self-tuning PID would need special attention during an exception. Figure 17 shows the step response of the emission and filament current versus time, where the emission loop is closed and the filament is initially set to 1000 K. Note that the emission is fairly slow in achieving the desired set-point value, mainly because of the sluggish response of the filament and the resulting thermal delay in the filament temperature. The approximate 12-s time delay could be detrimental to the weld quality and would need to be shortened. One option is to retune the PID; however, the overshoot would increase, and the resulting tuning parameters would not be optimal for the normal emission current values. It should be noted that the emission delay cannot be treated as a linear delay, or fixed-time delay, and that methods such as the Smith Predictor (Smith, O. J. M., 1957) or similar process delay estimators, would be applicable only with special provisions for the nonlinear relationship that exists between the process delay and filament current. There

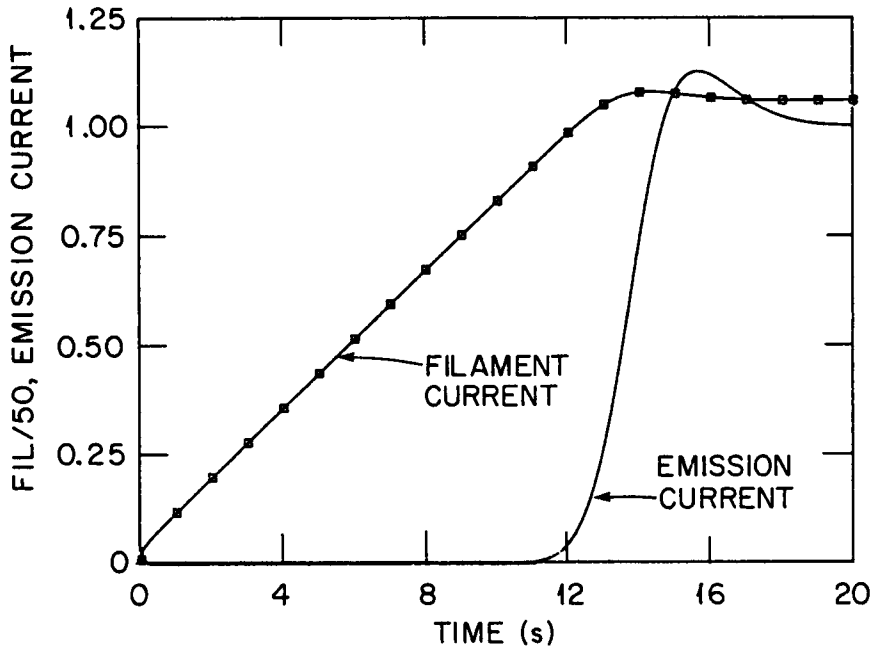


FIGURE 17
EMISSION, FILAMENT CURRENT STEP RESPONSE
WITHOUT MODE CONTROL

again, the process exceptions must be quantified, qualified, and evaluated in order to use these predictors. As stated before, the PEV is an attempt to do just that.

The method used, and the one evaluated in this thesis, is "mode" control, as indicated in Fig. 14. If the emission current is found outside the "normal emission region," then the appropriate control action will be taken. Figure 18 shows the result of utilizing the mode switching scheme, with the same initial conditions as in Fig. 17. The time delay of the emission is reduced by approximately 8 s. Note the cross-over from filament mode to emission mode when the emission exceeds 0.1 A.

Figure 16 shows the filament and emission current response to HV arcing (i.e., at time = 0.5 s and time = 1.0 s). The HVPS provides the HV cut-back signal that is used to determine the control mode. Unlike the other process exception, the HV arcing does not have to be estimated from process data. The control mode is switched from emission mode to filament mode when the arc occurs and the desired results are achieved. The HV arcing recovery scheme is illustrated in Fig. 19.

Filament activation requires a more elaborate method for exception handling and detection. The filament activation was simulated, and methods for supervisory control were considered. Figure 20 shows the steady-state, emission mode response to a simulated activation. Typically, an activation of this magnitude would cause HV cut-back and arc recovery within the HVPS. If, for example, the HVPS cutback current (maximum peak current) were set at 1.2 A, an activation current, like the one shown in Fig. 20, would cause the HVPS to cut back. This combination of excursions will be considered later. Stabilizing the closed-loop, emission mode EBGC system during an activation will require an approach similar to that of the HV arcing and will also utilize the region exclusion technique shown in Fig. 14. One additional problem with the activation detection and correction is the fact that the duration is not as predictable as was the HV arc. Real-time determination

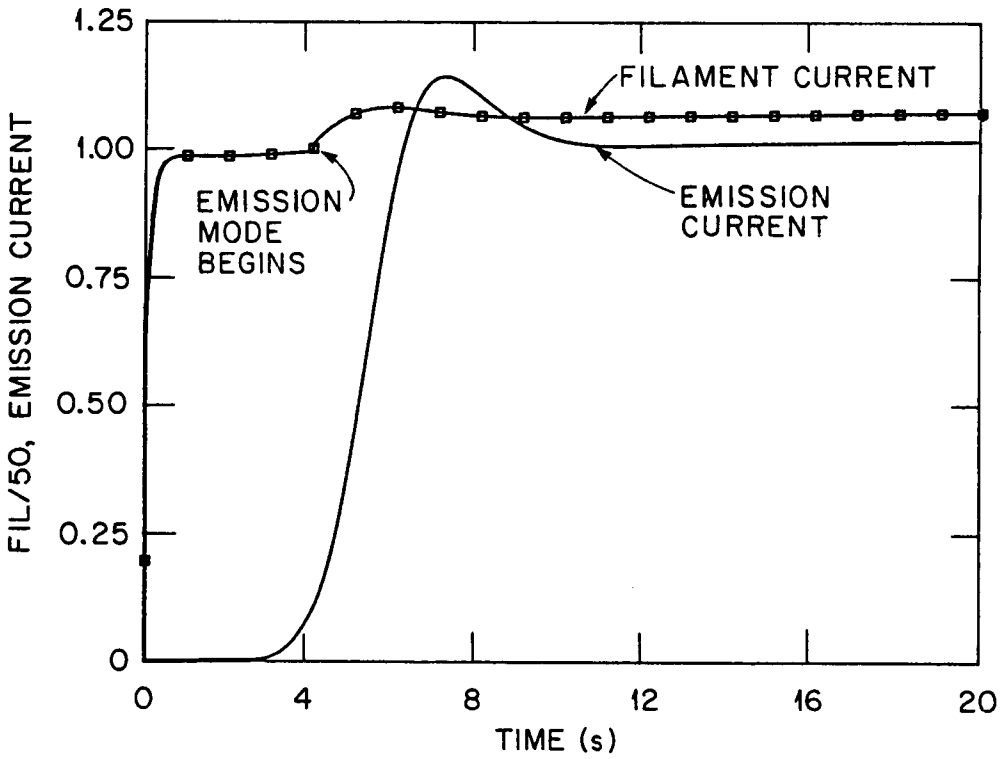


FIGURE 18
EMISSION, FILAMENT CURRENT STEP RESPONSE
WITH MODE CONTROL

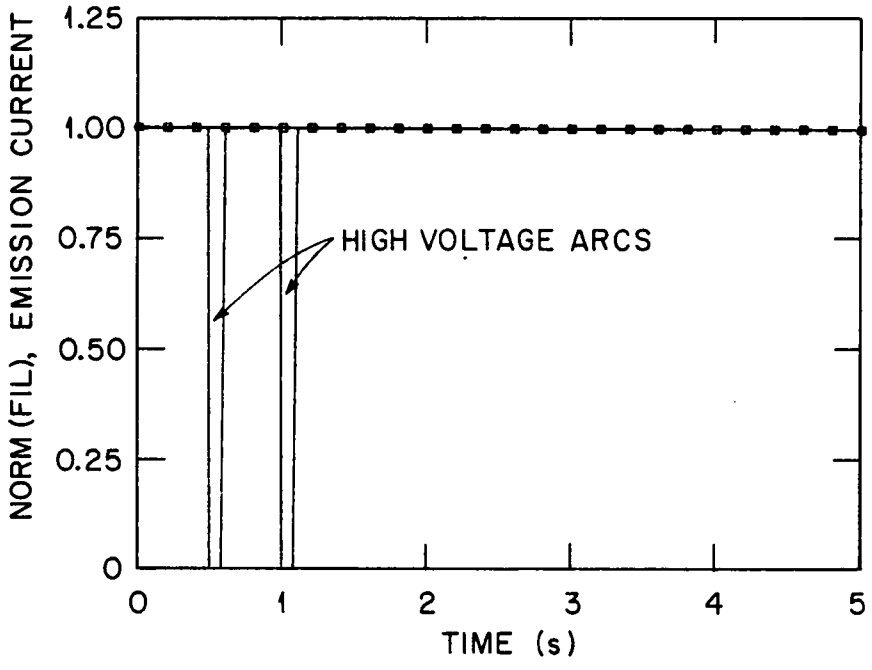


FIGURE 19
RESPONSE TO HV ARCING, STEADY-STATE CONDITION
WITH MODE CONTROL

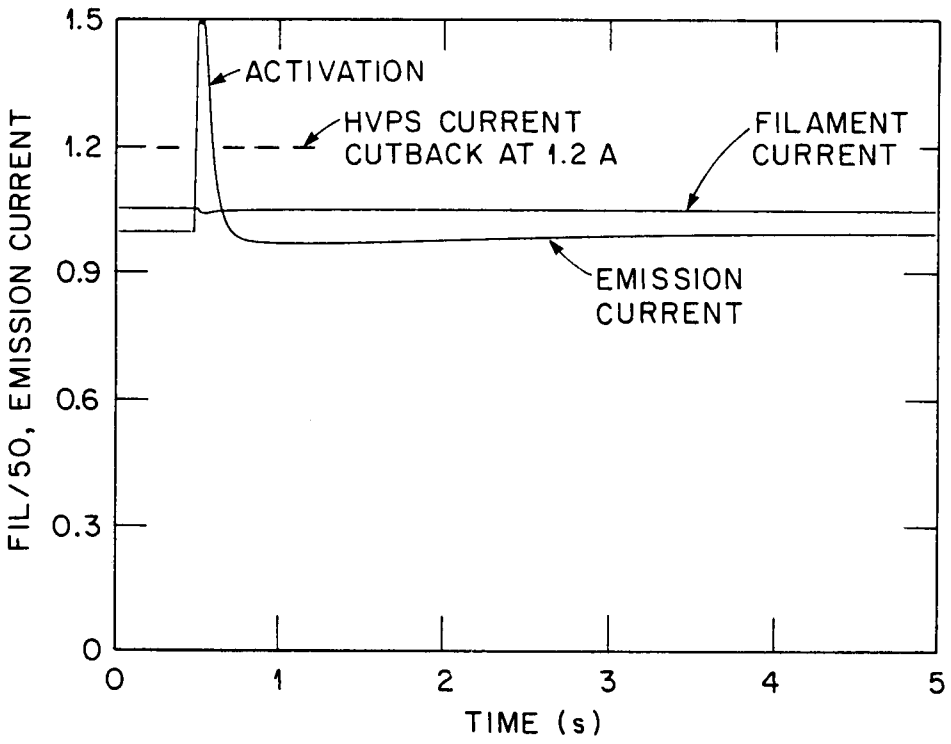


FIGURE 20
SIMULATION OF ACTIVATION, 50-ms DURATION

of the activation and activation type will be required. Corresponding impacts on the beam quality, focus, etc. will not be considered because these qualities are not included in the model. The recovery scheme will consider only the class I activation, as described in Section III.

The activation current can perturb the system in two possible ways: increase the emission current or decrease the current. If a moderate change in emission current is experienced, then constant power can still be maintained by changing the HV bias such that the power in the beam remains constant; however, any sudden increase in the HV bias could cause arcing. Conversely, a decrease in the HV bias typically will not directly cause an arc. Because of this fact, the "de-activation" case will not be addressed in this thesis; furthermore, since the pure tungsten filament material is assumed in these studies, the "de-activation" case is very unlikely. As stated before, very few compounds/elements have work functions that are less than that of pure tungsten.

To minimize the impact of the activated filament, the following scheme will be used: (1) determine normal emission current versus filament current steady-state values, (2) determine the activation region by looking for points that exceed a predetermined threshold and classify all points that are not within the "normal operation region," and, if an exception is suspected, (3) switch to the proper control mode and (4) execute the pre-defined recovery scheme until the exception is resolved and return to step (1) on completion; otherwise, convert to operator (or host) control. If, for example, a class II activation is declared, then the system would be turned over to operator control or host control.

Figure 21 shows the normal response of the emission and filament without exception handling. The activation PEV was used to estimate the activation limit, or threshold, that would represent the boundary line as described in Fig. 14 (i.e., the activation limit as estimated by the activation PEV). In Fig. 21 the control mode remains in the emission mode during the excursion.

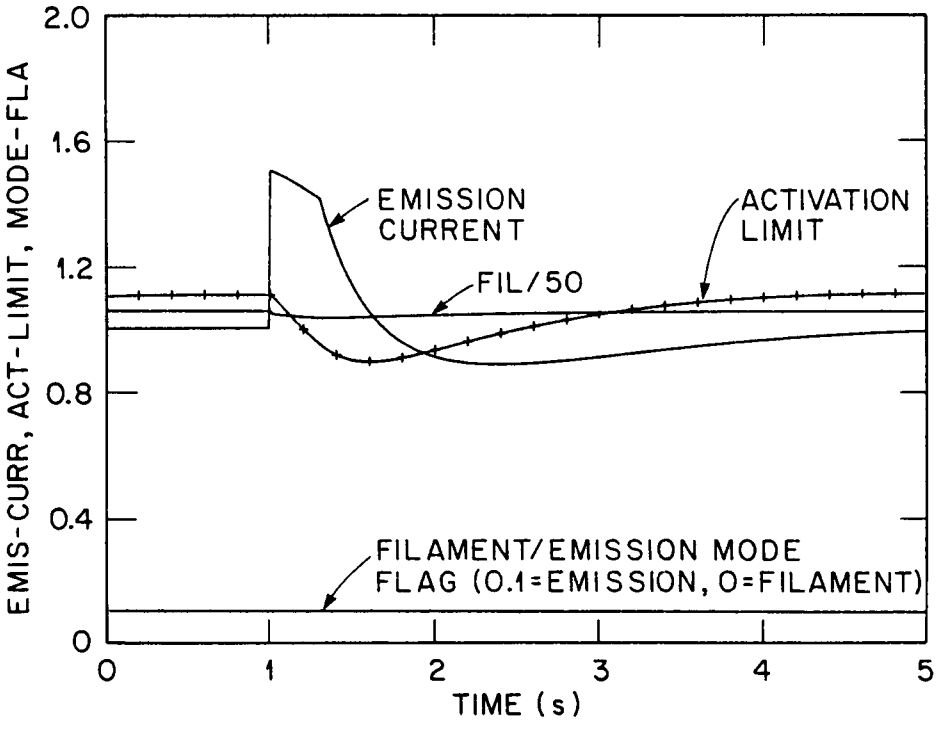


FIGURE 21
EMISSION, FILAMENT CURRENT RESPONSE
WITH ACTIVATION

The PEV typically must be able to run in real time and consequently must be constructed so that the least amount of numerical computation is required. The PEV can usually be developed by using post-process data, or archived data, where the performance of the PEV can be evaluated and tested. Another technique that could be used in the development of the PEV is to implement the PEV in real time and display and/or record the results. The confidence of the predictions could be established, and several techniques could be "tried and tested." The latter approach has been taken and is illustrated in Fig. 21. In other words, the "activation limit" in Fig. 21 is the PEV estimate of the upper threshold, at which point the activation should be declared and shows that the activation PEV prediction is usable. The estimate is done by using an empirical formulation and is described in the Appendix. When considering a large control system, one can appreciate the accumulative computation requirements of the PEV when the number of PEVs becomes quite large. The PEV must be robust, accurate, and most importantly, very efficient.

Once the activation PEV was tested and evaluated, a recovery and exception handling scheme was developed by using the PEV estimate to determine when activation occurs and to switch the "control mode" (i.e., from emission mode to filament mode). Figure 22 shows the implementation of an activation recovery scheme where the activation PEV estimate is used to switch the control mode so that the perturbation to filament current will be minimized. In other words, when the emission current exceeds the activation limit, the EBGC will switch to the emission mode.

The beam power (i.e., $I_e \times \mathrm{HV}$) can be maintained at a constant value, if desired, by using the activation PEV to trigger an appropriate HV set-point change. The HV set point would be adjusted so that the net power in the beam would be maintained at the previous value (before the activation occurred). Since the EB optics are not described in this EBGC model, this option is only mentioned as a

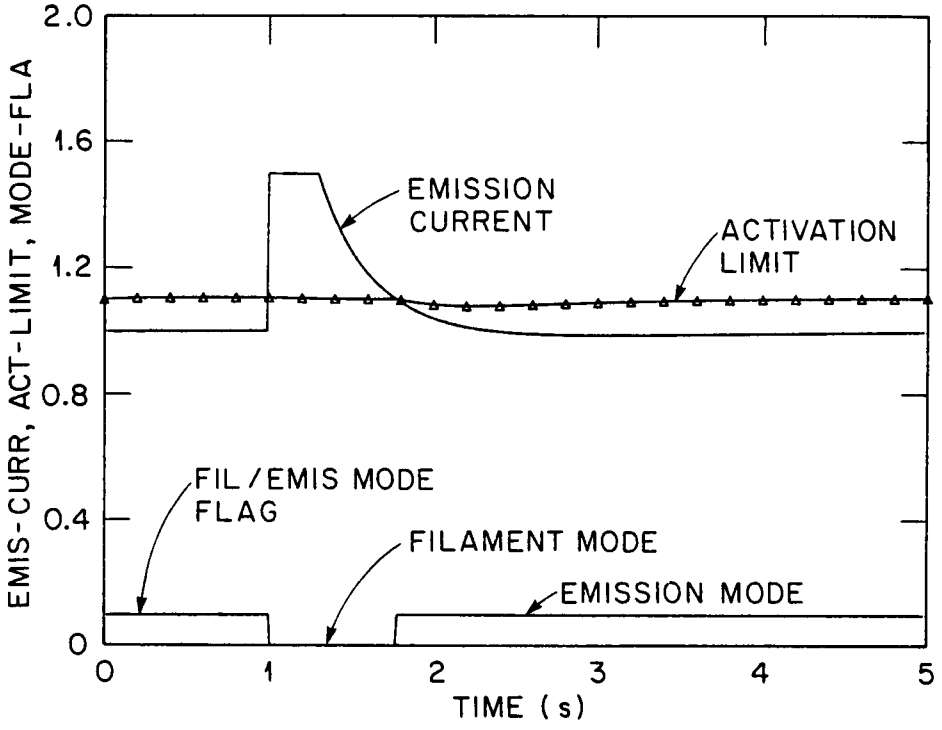


FIGURE 22
ACTIVATION RECOVERY SCHEME USING
MODE SWITCHING

possible design consideration and may not be a viable option when considering a particular system.

Figure 23 shows the combination of effects, HV arcing and activation, occurring together. Note that, when the arc (time = 2.0 s) occurs, the filament is not activated (as far as the EBGC is concerned); however, when the HV is turned back on, the activation is detected. In a real system, this could happen as follows: the filament becomes activated and instantaneously (i.e., with respect to the sampling time of the EBGC) causes the emission to rise, which in turn causes an immediate HV arc down. In Fig. 23, all recovery schemes are disabled to show the full effect of the perturbations. In Fig. 24, the activation limit is displayed and shows how the activation PEV is used directly (i.e., when the measured emission current becomes greater that the activation limit) to predict the activation. The activation flag and control mode are plotted in Fig. 24 to show when the activation detection occurs and when the control mode changes. Note that the control mode is switched when the HV arc occurs and when the activation is detected. The filament current is maintained fairly constant throughout the entire excursion sequence. As stated before, maintaining a constant beam power would require the modification of the HV set point during the activation; however, there are usually other parameters within the beam steering, focus, and beam position controller that can be used to lessen the impact of the class I activation. These considerations are, of course, dependent on specific EB designs; however, as shown in Fig. 1, information and process status developed within the lower-level PEVs would be passed among, or shared with, other layers of the architecture. There is not space here to elaborate on the issues of implementing a real-time distributed database; however, the status of the EBGC would be shared, where appropriate, with other hardware and functional blocks of the architecture.

To further enhance the model closure and to enhance the simulation, the capability to add random Gaussian noise was included. The noise generators simulate

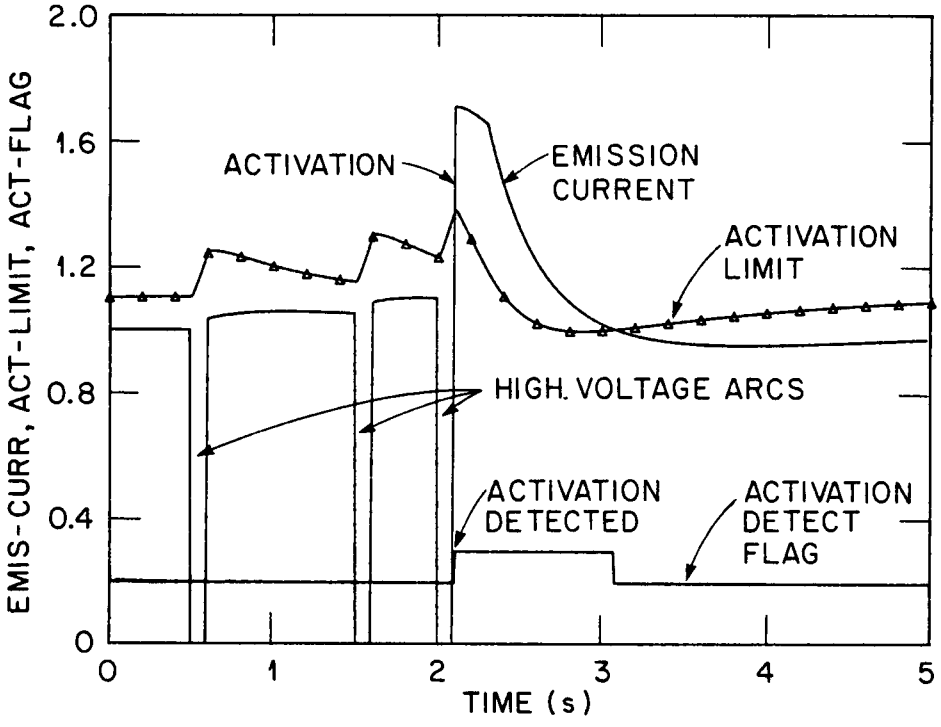


FIGURE 23
COMBINATION ARCING AND ACTIVATION WITHOUT RECOVERY SCHEME

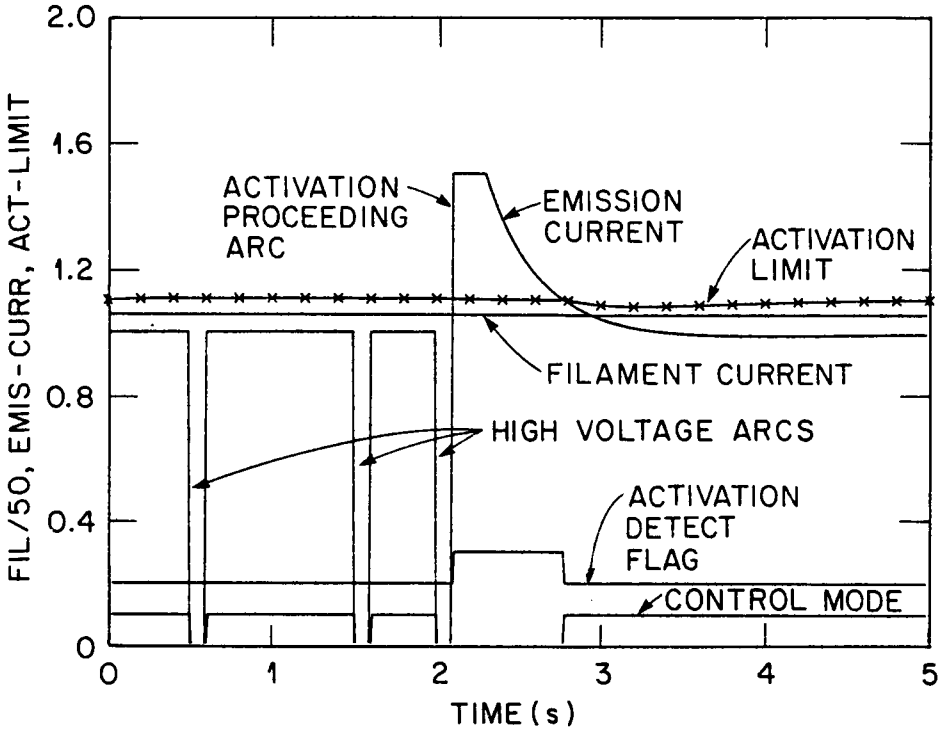


FIGURE 24
COMBINATION ARCING AND ACTIVATION
WITH FULL RECOVERY

normal process noise and thermal noise that is found in all physical processes. The instrumentation noise and data conversion noise are also present, are assumed to be additive with all other process noise, and will be modeled with a single Gaussian noise source. The average (or mean value) of the noise is assumed to be zero. Instrumentation offsets (constant values) are not included in the noise generator. The model will add the noise, when specified, and needs only the standard deviation (S.D.) of the noise to be simulated. Figure 25 shows the same conditions as described in Fig. 24 but with the addition of process noise, where emission current noise S.D. is 0.01 A and filament current noise S.D. is 0.5 A. Filtering techniques for process variables can be used in cases where the background noise is excessive. Most of the measured process variables within the EBW system are typically low noise; however, due to the HV arcing and the large production of X-ray radiation within the vacuum vessel, this can cause a significant electromagnetic interference (EMI) noise problem. Special attention to return-path and instrumentation grounding is important in reducing EMI noise. Transients and EMI coupling and EMI noise are hard to characterize and will not be addressed in this thesis.

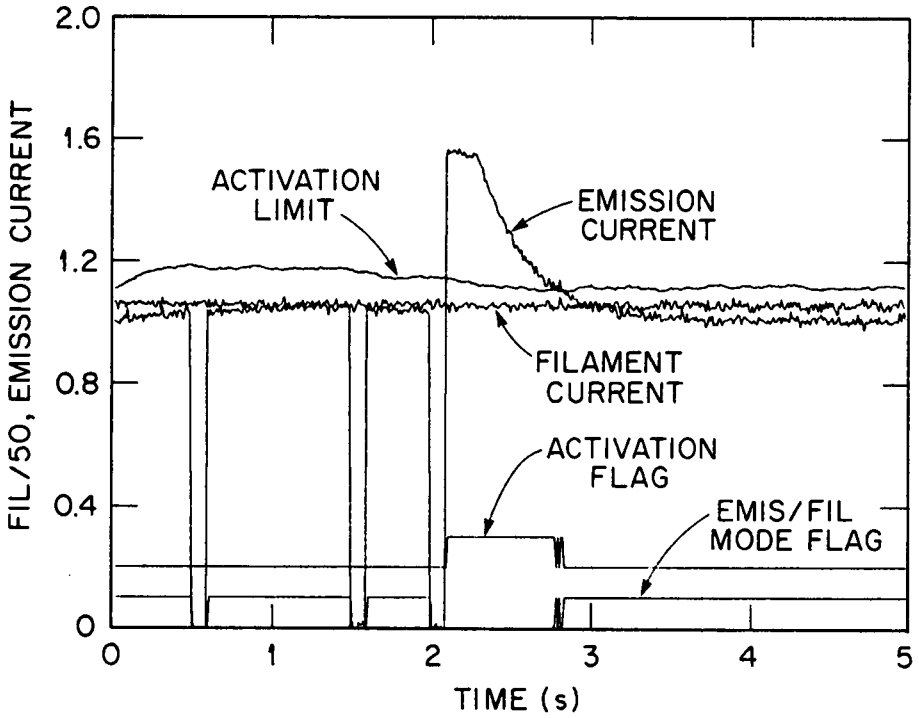


FIGURE 25
COMBINATION ARCING AND ACTIVATION, GAUSSIAN NOISE WITH FULL RECOVERY

V. DISCUSSION OF RESULTS

The application of the hierarchical approach to a complex control problem has been demonstrated and studied. The general problem of the nonlinear, nonstationary case is considered by applying some of the issues found in the automation of the EBWS. The issues related to the EBWS required the utilization of diverse methods for capturing process knowledge and characterization of the process. The handling of process exceptions and supervisory control functions were used to illustrate the principle of the coexistence of the hierarchical process decision and hierarchical process analysis. The PEV was demonstrated as the hierarchical process analysis function, and it provided a mechanism to incorporate the process analysis, with the necessary control decisions, to meet the objectives. Figure 2 contains a block diagram of the control decision model that is used to accomplish closed-loop performance and to handle real-time process exceptions. The control decision model is functionally consistent with each stage of the hierarchical structure as shown in Fig. 1.

The development and application of the PEV were illustrated and applied to the handling of process exceptions, namely HV arcing and filament activation. Typically, handling process exceptions/faults require control actions that are not compatible with, or equivalent to, those control actions that are executed while the process is in the "normal control state" (or steady-state region). The objectives and process evaluations are different for each level of the hierarchy; however, a decision model was presented (Fig. 3) that is consistent with each level of the architecture. The elements discussed were: the process (PEV) analysis derived from the process observations, the objectives, and the resulting decision-making process that executed the predetermined tasks. For the case of the complex system, the objectives are usually represented in terms, formats, and/or procedures that are not purely quantitative. Furthermore, the objectives are based on process

states that are not easily obtainable and/or not easily quantified. For example, for a given process, the objective may be stated as simply as: "minimize cost and increase efficiency for all control situations"; however, success in meeting these objectives requires that the real-time costs and efficiencies can be obtained and that these process evaluations can be stated quantitatively so that control decisions can be made based on these evaluations. The PEV is the functional element that provides the information necessary for executing the appropriate control actions that accomplish the objectives.

The SC actions are those that modify the real-time control law (or strategy) to meet the objectives and to handle the process exceptions. The SC functions were illustrated, by application and simulation, by the EBGC system. The activation and HV arcing exception handling required actions that temporarily changed the control strategy so that the system could recover from the process exception with minimum process upset. Figure 23 illustrates the system response without exception handling, and Fig. 24 shows the end result when applying these exception handling strategies.

VI. SUMMARY AND CONCLUSIONS

The usefulness of an integrated approach was illustrated when addressing the complex issues of exception handling and supervisory control actions. The concepts of hierarchical analysis and qualitative control spaces were introduced as workable control elements of the hierarchical control structure. The PEV was discussed and defined as the hierarchical analysis element of the hierarchical control structure.

The qualitative approach and reasoning were shown to be valuable in the design of a robust controller that can handle a variety of process exceptions. The discrete versus continuous states were coupled in the control decisions and process state determination. Furthermore, the utilization of both qualitative and quantitative information in the control scheme was accomplished in a consistent manner. The concept of the qualitative approach was shown to be applicable when other methods were not sufficient because of their dependency on stationary process parameters. Adaptive techniques, if not properly handled, were also found to be inadequate when process exceptions occur, and process parameter estimation techniques were found to be valid only when the process was currently in the appropriate control mode (illustrated in Figs. 15a and 15b).

A small portion of the EBWS (Fig. 2) was modeled and used to show the approach. A model of the EBGC (Figs. 12 and 13) consisting of the EB gun, filament controller, and emission controller was constructed and included simulations of two process exceptions, namely HV arcing and filament activation. The integrated model was used to apply the concepts as described in Figs. 1 and 3 and to implement specific examples of the real-time PEV and decision model. The concept of "control mode" (Fig. 14) was shown to be useful in establishing a qualitative control space and control philosophy to handle process exceptions.

The application of artificial intelligence techniques, advanced control theory, database management, scheduling, etc., requires an "integrating factor" and common format. The hierarchical control structure requires several of the tools/disciplines just mentioned and encourages the utilization and application of the "right" tool to do the "right" task.

BIBLIOGRAPHY

BIBLIOGRAPHY

Arate, Y., et al. "Development and Application of High Power Electron Beam Welding." *Electron and Laser Beam Welding.* International Institute of Welding. New York, 1986. 149–158.

Bakish, R. and White, S. S. *Handbook of Electron Beam Welding.* John Wiley and Sons, Inc. New York, 1964.

Bobrow, Daniel G., et al. *Qualitative Reasoning About Physical Systems.* Cambridge, Massachusetts: MIT Press, 1985.

Faurre, Pierre. *Elements of System Control Theory.* New York: North-Holland Publishing Company, 1977.

Handbook of Chemistry and Physics. 64th Edition. 1983–1984. Boca Raton, Florida: CRC Press, 1984. E-358 and E-386.

Kita, H., et al. "Development and Application of High Power Electron Beam Welding," *Electron and Laser Beam Welding (Proceedings of the International Conference, Tokyo, Japan, 1986).* First Edition. New York: Pergamon Press, 1986.

Lee, Robert C. K. *Optimal Estimation, Identification, and Control.* Cambridge, Massachusetts: MIT Press, 1964.

Ljung, Lennart, and Soderstrom, Torsten. *Theory and Practice of Recursive Identification.* Cambridge, Massachusetts: MIT Press, 1986.

Lotfi, Zadeh A. *Computing Methods in Optimization Problems—2.* New York: Academic Press, 1969.

McGraw-Hill Encyclopedia of Science and Technology. McGraw-Hill Book Company. 6th Edition, New York. 417–421.

Pierce, J. R. *Theory and Design of Electron Beams.* New York, 1954.

Schwartz, M. M. *Electron Beam Welding.* New York: Welding Research Council, 1974. 196.

Schwartz, Melvin M. *Source Book on Electron Beam and Laser Welding.* American Society for Metals. Metals Park, Ohio, 1981.

Smith, G. D. *Numerical Solution of Partial Differential Equations: Finite Difference Methods.* Third Edition. New York: Oxford University Press, 1985.

Smith, O. J. M. "Closer Control of Loops with Dead Time," *Chem. Eng. Prog.* **53(5)**, 217–9 (1957).

Tanenbaum, Andrew S. *Computer Networks.* Englewood Cliffs, N.J.: Prentice-Hall, 1981.

The Encyclopedia of Physics. Van Nostrand Reinhold Company. New York, 1985.

Thomas, Lindon C. *Fundamentals of Heat Transfer.* New Jersey: Prentice-Hall, 1980.

APPENDIX

Richard-Dushman equation for tungsten:

$$j_e = A_R T^2 \exp\left[-\left(\frac{\phi}{KT} - \frac{B\sqrt{E}}{T}\right)\right] \quad (\mathrm{A/cm^2}) \ ,$$

where j_e is the saturation emission current density,

T = filament temperature (K),

E = normal electric field (V/cm),

A_R = Richardson coefficient,

ϕ = work function,

K = Boltzmann constant,

B = Schottky coefficient.

For tungsten:*

A_R = 59.93 $\mathrm{A/cm^2 \cdot K^2}$,

ϕ = 4.5138 eV,

B = 4.403 $\mathrm{K/(V/cm)^{1/2}}$.

*Applies to nonideal, polycrystalline metal surfaces.

Comparison of cathode materials

Material	Emission (A/cm^2)	Operating temperature (°C)	ψ (eV)	Upper pressure limit
W	0.6	2200	4.55	10^{-4}
Ta	0.5	2000	4.1	10^{-5}
Rh	0.2	2200	4.8	10^{-4}
Thoriated tungsten	1–3	1750	2.6	5×10^{-6}
Oxide coated	0.5	830	1.2	10^{-6}
Dispenser	0.5–10	960–1130	1.5–2.0	5×10^{-6}

PROPERTIES OF TUNGSTEN

Jones and Langmuir, General Electric Review

Temp. °K	Resistivity microhm cm	Electron emission amp./cm²	Evaporation g/cm² sec	Vapor pressure dynes/cm²	Thermal expansion per cent l_o at 293°	Atomic heat cal./g. atom./°C.
300	5.65	—	—	—	.003	6.0
400	8.06	—	—	—	.044	6.0
500	10.56	—	—	—	.086	6.1
600	13.23	—	—	—	.130	6.1
700	16.09	—	—	—	.175	6.2
800	19.00	—	—	—	.222	6.2
900	21.94	—	—	—	.270	6.3
1000	24.93	1.07×10^{-15}	5.32×10^{-34}	1.98×10^{-29}	.320	6.4
1100	27.94	1.52×10^{-13}	2.17×10^{-30}	1.22×10^{-25}	.371	6.4
1200	30.98	9.73×10^{-12}	3.21×10^{-27}	1.87×10^{-22}	.424	6.5
1300	34.08	3.21×10^{-10}	1.35×10^{-24}	8.18×10^{-20}	.479	6.7
1400	37.19	6.62×10^{-9}	2.51×10^{-22}	1.62×10^{-17}	.535	6.8
1500	40.36	9.14×10^{-8}	2.37×10^{-20}	1.54×10^{-15}	.593	7.0
1600	43.55	9.27×10^{-7}	1.25×10^{-18}	8.43×10^{-14}	.652	7.1
1700	46.78	7.08×10^{-6}	4.17×10^{-17}	2.82×10^{-12}	.713	7.2
1800	50.05	4.47×10^{-5}	8.81×10^{-16}	6.31×10^{-11}	.775	7.4
1900	53.35	2.28×10^{-4}	1.41×10^{-14}	1.01×10^{-9}	.839	7.6
2000	56.67	1.00×10^{-3}	1.76×10^{-13}	1.33×10^{-8}	.904	7.7
2100	60.06	3.93×10^{-3}	1.66×10^{-12}	1.28×10^{-7}	.971	7.8
2200	63.48	1.33×10^{-2}	1.25×10^{-11}	9.88×10^{-7}	1.039	8.0
2300	66.91	4.07×10^{-2}	8.00×10^{-11}	6.47×10^{-6}	1.109	8.2
2400	70.39	1.16×10^{-1}	4.26×10^{-10}	3.52×10^{-5}	1.180	8.3
2500	73.91	2.98×10^{-1}	2.03×10^{-9}	1.71×10^{-4}	1.253	8.4
2600	77.49	7.16×10^{-1}	8.41×10^{-9}	7.24×10^{-4}	1.328	8.6
2700	81.04	1.63	3.19×10^{-8}	2.86×10^{-3}	1.404	8.7
2800	84.70	3.54	1.10×10^{-7}	9.84×10^{-3}	1.479	8.9
2900	88.33	7.31	3.30×10^{-7}	3.00×10^{-2}	1.561	9.0
3000	92.04	1.42×10	9.95×10^{-7}	9.20×10^{-2}	1.642	9.2
3100	95.76	2.64×10	2.60×10^{-6}	2.50×10^{-1}	1.724	9.4
3200	99.54	4.78×10	6.38×10^{-6}	6.13×10^{-1}	1.808	9.5
3300	103.3	8.44×10	1.56×10^{-5}	1.51	1.893	9.6
3400	107.2	1.42×10^{2}	3.47×10^{-5}	3.41	1.980	9.8
3500	111.1	2.33×10^{2}	7.54×10^{-5}	7.52	2.068	9.9
3600	115.0	3.73×10^{2}	1.51×10^{-4}	1.53×10	2.158	10.1
3655	117.1	4.79×10^{2}	2.28×10^{-4}	2.33×10	2.209	10.2

Source: Handbook of Chemistry and Physics. 64th Edition. 1983-84. Boca Raton, Florida: CRC Press, 1984.

PARAMETERS USED FOR FILAMENT/EMISSION MODEL

$$\Delta t = 1 \text{ ms}$$

$$\text{Length} = 2.54 \text{ cm}$$

$$T_{\text{INF}} = 800 \text{ K}$$

$$d = 0.1 \text{ cm}$$

$$w = 0.1 \text{ cm}$$

$$q_{\text{COND}} = 0$$

PARAMETERS USED FOR FILAMENT CONTROLLER (SEE FIG. 9)

$$K_p = 2.0$$

$$K_I = 2.0$$

$$K_p = 0.0$$

$$K = 3.0$$

$$\tau = 8 \text{ ms}$$

$$R_0 = 0.01\ \Omega$$

PARAMETERS USED FOR EMISSION CONTROLLER

$$K_p = 1.0$$

$$K_I = 10.0$$

$$K_p = 0.0$$

ACTIVATION PEV

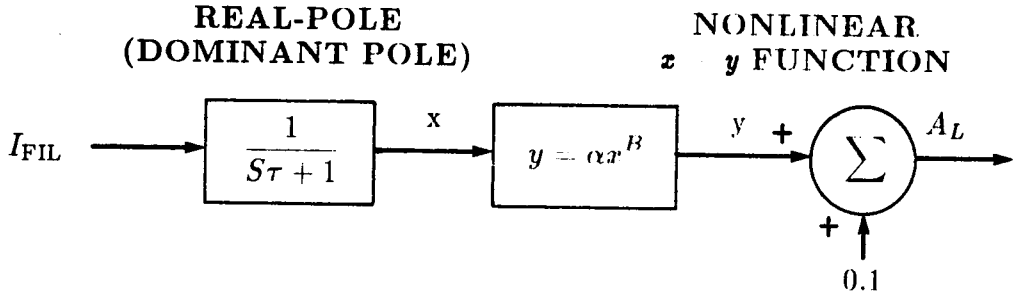
Steady-state values:

Filament Current	Emission Current	Estimate
40	0.025	0.025
45	0.130	0.121
50	0.513	0.495
52.7	1.00	1.00

Steady-state model:

$$\hat{I}_e = (I_{FIL})^{13.366} \cdot (9.691 \times 10^{-24})$$

ACTIVATION LIMIT



α = 9.691×10^{-24}

β = 13.366

τ = 200 ms

A_L = Activation limit = $y + 0.1$

MODEL CLOSURE AND COMPARISON TO THEORETICAL CALCULATIONS OF EMISSION/FILAMENT CURRENT DYNAMICS

(Properties of tungsten are taken from standard tables, CRC manual)

The assumptions described in Chapter III result in the following partial differential equation for radiant cooling:

$$\dot{q}_{\text{rad}} = \frac{\partial}{\partial t}(\rho C_{\text{p}} T) \quad , \tag{A.1}$$

where $\dot{q}_{\text{rad}}$ is the instantaneous heat flow, ρ is the density (g/cm^2), C_{p} is the heat capacity (W $\cdot$ s/g $\cdot$ K), and T is the temperature (K).
For total heat:

$$Q = A_{\text{CR}} \frac{\partial}{\partial t}(\rho C_{\text{p}} T) \quad , \tag{A.2}$$

where A_{CR} is the cross-sectional area (cm^2) and Q is heat flow (W/cm), assuming a pure tungsten filament. Density and specific heat are nonlinear functions of temperature. To transform Eq. (A.2) into standard form, Q will be evaluated at T_{R}. Therefore, $\rho(T_{\text{R}})$ and $C_{\text{p}}(T_{\text{R}})$ will be treated as a constant:

$$Q|_{T_{\text{R}}} = A_{\text{CR}} \rho(T_{\text{R}}) C_{\text{p}}(T_{\text{R}}) \left[\left. \frac{\partial T}{\partial t} \right|_{T_{\text{R}}} \right] \quad . \tag{A.3}$$

Rewriting in standard form:

$$\left. \frac{dT}{dt} \right|_{T_{\text{R}}} = (Q|_{T_{\text{R}}}) \left[\frac{1}{A_{\text{CR}} \cdot \rho(T_{\text{R}}) \cdot C_{\text{p}}(T_{\text{R}})} \right] \quad . \tag{A.4}$$

Since radiant heat loss is assumed to be dominant,

$$Q|_{T_{\text{R}}} = -A_{\text{s}} \sigma \epsilon(T_{\text{R}}) (T_{\text{R}}^4 - T_{\infty}^4) \quad , \tag{A.5}$$

where σ is Boltzmann's constant and $\epsilon(T_{\text{R}})$ is the emissivity, at T_{R}.

Using the Chain Rule:

$$\frac{dI_e}{dt} = \frac{\partial I_e}{\partial T} \cdot \frac{dT}{dt} \quad , \tag{A.6}$$

where I_e is the emission current.

Combining Eqs. (A.4)–(A.6) gives:

$$\left.\frac{dI_e}{dt}\right|_{T_R} = \left(\left.\frac{\partial I_e}{\partial T}\right|_{T_R}\right)\left[\frac{A_s \sigma \epsilon(T_R)}{A_{CR}\rho(T_R)C_p(T_R)}(T_R^4 - T_\infty^4)\right] \quad . \tag{A.7}$$

$$I_e = A_s j_e \quad , \tag{A.8}$$

where j_e is the current density (A/cm^2) and A_s is the surface area for unit length (cm^2).

Using the Richardson-Dushman equation,

$$j_e = A_R T^2 e^{-\alpha/T} \quad (A/cm^2) \quad , \tag{A.9}$$

where α is constant and A_R is Richardson's coefficient; for tungsten, $\alpha = 52,379$ K and $A_R = 59.93$ ($A/cm^2 \cdot K^2$).

$$\frac{\partial j_e}{\partial T} = 2A_R T e^{-\alpha/T} + A_R \alpha e^{-\alpha/T} \quad . \tag{A.10}$$

Rewriting Eq. (A.10) gives:

$$\frac{\partial j_e}{\partial T} = A_R \cdot e^{-\alpha/T} \cdot (2\,T + \alpha) \quad . \tag{A.11}$$

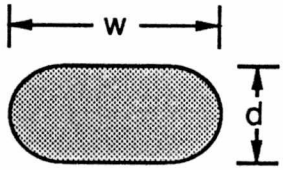
Evaluating at T_R:

$$\left.\frac{\partial j_e}{\partial T}\right|_{T_R} = A_R \cdot e^{-\alpha/T_R} \cdot (2\,T_R + \alpha) \quad . \tag{A.12}$$

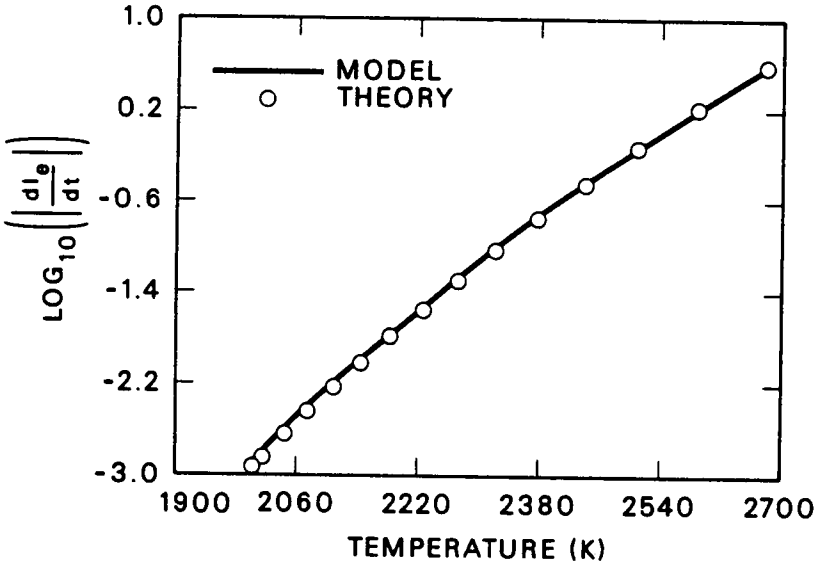
Using Eqs. (A.3), (A.4), (A.6), and (A.12),

$$\begin{aligned} \therefore \left.\frac{dI_e}{dt}\right|_{T_R} = & -A_s^2\left[A_R \cdot e^{-\alpha/T_R} \cdot (2T_R + \alpha)\right] \\ & \cdot \left[\frac{\sigma\epsilon(T_R)}{A_{CR} \cdot \rho(T_R)C_p(T_R)}(T_R^4 - T_\infty^4)\right] \quad , \end{aligned} \tag{A.13}$$

where $A_s = 2w + (\pi - 2)d$; w and d are the filament dimensions:



```
C_____________________________________________________________C
C                                                             C
C       Model Closure Verification                            C
C                                                             C
C       Numerical Calulations for dIe/dt                      C
C        where : dIe/dt = non-linear function                 C
C       Programmer : J.T. Greer                               C
C_____________________________________________________________C
C  As ! Surface Area, CM**2     ;  Acr ! Cross Sec. Area, CM**2
C  w,d ! dim.s of filament, CM  ;  Ar ! Richardson's Coeff
C  alpha ! constant             ;  Tr ! Ref. Temp.
C  Boltz ! Boltzmann's constant ;  emissivity ! emissivity
C  rho ! density                ;  Cp ! Heat Capacity
C  Tinf ! T background, Deg-K
       Parameter (pi = 3.14159265)
       Real As, Acr, w, d, Ar, alpha, Tr, Boltz
       Real emissivity, rho, Cp, Tinf
C
       Data Ar/59.93/, alpha/52379./, Boltz/5.6697e-12/
       Data rho/19.3/ ! relatively constant (compared to Cp)
       Data Tinf/800./
C
       type*,' Enter Width and Depth of filament (CM) : '
       accept*,w,d
C
Cal Surface and CS Area :
              As = 2.*w + (pi-2.)*d
       type*,' Surface Area : ',As
              Acr = (w-d)*d + ((d/2.)**2)*pi
       type*,' Cross Sectional Area : ',Acr
       type*,' Enter Ref. Temp. : '
       accept*, Tr
C
C Open storage file:
!      open(unit=1, file='dym.dat', status='new')
C<< Begin Iteration >>
!      do Tr = 2000.,2700., 10.
Cal emissivity : @Tr
              emissivity  = 0.1012348e-3*( Tr**1.025169 )
       type*,' emissivity = ',emissivity
Cal Cp : @Tr
              Cp = 33.09066*( Tr **0.22 )
       type*,' Cp = ',Cp
              Cp = Cp*1e-3 ! W-MS/gm-K
C
Cal d/dt[Ie]
              Dr = (As**2)*(Ar*exp( -(alpha/Tr) ))*( 2.*Tr + alpha )*
       1 ( (Boltz*emissivity*( Tr**4 - Tinf**4 ) )/(Acr*rho*Cp))
C
       type*,' '
       type*,' Temp = ',Tr,'   dIe/dt = ',Dr
!      write(1,*) Tr,alog10(Dr)
!      enddo
       stop
       end
```



Note: The comparison of the dI_e/dt curves shows very good agreement. The calculation of the dI_e/dt (of the model) from the radiant cool-down curves was calculated by using a first-order difference method. The calculation of dI_e/dt (theoretical) with fixed temperature was done by using first principles and well-verified properties of tungsten. These properties were measured with the highest precision and laboratory conditions and need no further verification. As stated, secondary effects for particular gun design geometries are not considered in this model, and this precludes the use of vendor-published specification sheets for accurate verification of the model.

FORTRAN SOURCE CODE

```
C
C-------------------------------------------------------------------
C
C Simulation Model of Emission Controller Sub-System
C
C     Includes Exception Handling : HV Arcing and Filament Activation
C        Programmer : J.T. Greer
C
C-------------------------------------------------------------------
C Filament - Emission  Model  ; Main Program
C
C           INPUTS:
C             Width of filament
C             Depth of filament
C             Length of filament
C             Filament Material
C             Conductivity constant
C             Radiant temperature (background) reference
C             Emissisity vs temperature data for filament material
C             Resisitivity vs temperature data for filament material
C             Initial Temperature of filament
C             Incremental time
C             Maximum iterations
C
C          OUTPUTS:
C             Emission current verses time
C             Filament Temperature verses time
C
C  ASSUMPTIONS :
C
C  1) the gun filament is housed within an vacuum environment
C  2) the filament is the cathode (e.g. the electron emitting surface),
C  3) the temperature of the filament and the surface area of the filament
C     primarily determine the electron emission current,
C  4) the gun is operated in a thermeionic emission limited mode
C     (space charge limitation is relitively small),
C  5) the primary heat loss from the filament is from radiant heat loss
```

```
C      (conduction loss is relitively small),
C   6) the filament is heated directly by the filament current
C      (e.g. resistive heating), and
C   7) the filament temperature distribution, along the filament,
C      is relitively constant.
C
C
C-------------------------------------------------------------------------
C
          Implicit Real*4 (A-H,L-Z)
          Logical HVarc ! HV arc Flag
          Logical Act_flag ! Activation Flag
          Logical Fil_FF_FLag ! Filament Forcing (for Emis model) Flag
          Logical Emission_mode ! Flag for Emission Mode( else Fil. mode)
          Logical Emission_mode_Req ! Flag for Emission Mode Request
          Logical Mode_switching ! Flag for Automatic Mode Switching
          Character*1 Mode ! Emission Mode/Fil. Mode Flag
          Real Arctimes(10) ! Times that arcs will occur
          Real Para1(3) ! PID parameters for Emission Controller
          Real Para2(3) ! PID parameters for Filament Controller
c
          data iseed/12457/
c Input Parameters and Model Control Instructions:
          write(5,10)
10        format(1h ,
          1 'Enter PID parameters for Emission Controller : (Pp,Pi,Pd) '
          1 '[2,2,0] : '$)
          accept*, para1
          write(5,20)
20        format(1h ,
          1 'Enter PID parameters for Filament Controller : (Pp,Pi,Pd) '
          1 '[1,10,0] : '$)
          accept*, para2
          write(5,30)
30        format(' Enter Sample Time for storage file (ms) [10] : ')
          accept*,sampletime
```

```
      write(5,40)
40    format(' Enter Initial Temp. of Filament (800 - 3000 DEG K ) : '$)
      accept*,Tfil
                    Emission_mode = .false. ! Default
      write(5,50)
50    format(' Do you want Emis. Mode (otherwise Fil. Mode (Y/N) ? : '$)
      read(5,60) itxt
60    format(a1)
                          Emission_mode_Req = .false.
      if((itxt.eq.'Y').or.(itxt.eq.'y')) then
                          Emission_mode_Req = .true.
                          Emission_mode     = .true.
      endif
      type*,' Emission Mode Flag = ',Emission_mode_Req
c
                    Fil_SP_Default = 50.
      if(Emission_mode_Req) then
      write(5,70)
70    format(' Do you want Automatic Emission Mode Switching (Y/N) ? : '$)
      read(5,60) itxt
                    Mode_switching = .false.
      if((itxt.eq.'Y').or.(itxt.eq.'y')) Mode_switching = .true.
      type*,' Mode_switching = ', Mode_switching
      else ! Fil. Mode
      write(5,80)
80    format(' Enter Fil. Curr Set Point : '$)
            accept*,Fil_SP_Default
      endif
c
      write(5,90)
90    format(' Do you want HV arcing ? (Y/N) : '$)
      read(5,60) itxt
                          HVarc = .false.
      if((itxt.eq.'Y').or.(itxt.eq.'y')) HVarc = .true.
      type*,' HVarc Flag = ',HVarc
```

```
          if(HVarc) then
100        write(5,110)
110       format(' How many arcs do you want ? (1 to 10) : '$)
          accept*,num_arcs
           if((num_arcs.gt.10).or.(num_arcs.lt.1)) then
                 type*,' Error in input... Try again.'
                 goto 100
           endif
c
           do i=1,num_arcs
                 write(5,120)
120       format(' Enter Time of HV Arc (to nearest ms) : '$)
          accept*, Arctimes(i)
           enddo
          endif ! (HVarc)
c
          write(5,130)
130       format(' Do you want Activation to occur (Y/N) ? : '$)
          read(5,60) itxt
                                Act_flag = .false.
          if((itxt.eq.'Y').or.(itxt.eq.'y')) Act_flag = .true.
          if(Act_flag) then
          write(5,140)
140       format(' Enter Act_begin, Act_duration, Time_constant : '$)
          accept*,Act_begin, Act_duration, Tau_act
          endif
c
c << Initialize >>
                  Open(unit=1, file='emis.dat', status='new')
          dt = 1. ! Milliseconds
                        Fil_FF_FLag = .true.
                        xLength = 2.54 ! CM ; 1"
                        Tinf = 800. ! deg K
                        Fil_curr = 0.
                        Width = 0.1 ! CM
```

```
                        Depth = Width ! Circular Filament
                        Cond = 0. ! No conductive loss
                        Time_end = 20000. ! 2. Seconds
                       HVPS = 50. ! KV
                        Offtime = 100. ! Milliseconds; HV cutback time
c
         Call FilTmp(Fil_FF_Flag, Dt, Volts, Tfil, Fil_SP_default,
        1 Emis, Resten, xLength, Tinf , Width, Depth, Cond)
c
        do time = 0., Time_end, dt
c <<< Simulation Model >>>
c
                Emis_SP = 1.0
                error_emis = Emis_SP - Emis_curr
c < Determine Emission Mode Switch >
        If ( Mode_switching ) then
         If(( Emis_curr.ge.0.1).and.( Emission_mode_req )) then
          Emission_mode = .true.
         else
          Emission_mode = .false.
         endif
        else ! Always in Emission Mode
          Emission_mode = .true.
        endif
c
        If( Emission_mode )  then
c Closed loop Emission Control
                            y0 = Fil_curr ! Bounceless Transfer
                Fil_SP = PID_1(dt, error_emis, init1, y0, para1 )
        else
c  Fil. Set Point
                 Fil_SP = Fil_SP_Default
        endif

         Fil_SP = Alimit(0., 60., Fil_SP )
```

```
c
c                       Resten = Restance of filament ; Current time
                          If(Resten.le.0) stop' Error!...Fil. Res = zero.'
          Fil_curr = Fil_control( init2, dt, Fil_SP , Resten , para2)
!         type*,' Fsp, Filcurr = ',Fil_sp, Fil_curr
c
           Call FilTmp(Fil_FF_Flag, Dt, Volts, Tfil, Fil_SP_default,
          1 Emis, Resten, xLength, Tinf , Width, Depth, Cond)
c
           Emis = Alimit(0.0, 3., Emis ) ! Limited by Power Supply
           Fil_curr = Alimit(0., 200., Fil_curr )
c
c << HV arcing simulation >>
          if(HVarc) then
                          Trig = 0.
                  do i=1,num_arcs
                          timedif = abs(time - Arctimes(i) )
                  if( timedif.lt.0.1)  then
                          Trig = 1.
                  endif
                  enddo
c
                  HV_status = HV_arc( time,  Trig, Offtime )
                  HV = HVPS * HV_status
          else
                  HV_status = 1.0 ! No arcs ; don't simulate any.
                  HV = HVPS
          endif
c
          if(Act_flag) then
           Emis_curr = Emis * HV_status *
          1(1.+Activation(dt, time, Act_Begin, Act_duration, Tau_act)/2.)
          else
           Emis_curr = Emis * HV_status
          endif
```

```
c
                if(amod( time, SampleTime).eq.0.) then ! Record every 100 ms
          write(5,*) time/1000., Fil_curr, Emis_curr, volts, Resten
          write(1,*) time/1000., Fil_curr, Emis_curr, volts, Resten
                endif
c
          enddo ! time
999       continue ! end
          close(unit=1)
c Store Final Values:
          open(unit=3, file='emis_final.dat', status='new')
          write(3,*) ' Time_end = ',time
          write(3,*) ' Fil Temp = ',Tfil
          write(3,*) ' Fil Curr = ',Fil_curr
          write(3,*) ' Fil Volt = ', Volts
          write(3,*) ' Emis     = ', Emis_curr
          write(3,*) ' Restance = ', Resten
          close(unit=3)
          stop
          end
c
          Real Function   Fil_control( init, dt, Fil_SP , Resten , Para)
c_____________________________________________________________________
c    This Subroutine Models the Filament Power Supply and other
c      Control Elements, for example :
c           the Step-Down Isolation Transformer, PID controller, etc.
c     Programmer : J.T. Greer
c
          Real dt, Fil_sp, Resten
          Real Para(3) ! PID parameters
          Real tau , Gain
          Real Rs ! Output impedance of Filament Power Supply
          data Rs/.01/ ! Effective output impedance of Fil. Power Supply ;ohms
          data tau/8.e-3/! 8 ms ;         Rise time of "     "     "
```

```
          data Gain/3./  ! The max. voltage output of the Fil. Power Supply
c                          is 3 volts.
c
          realpl(y , tau0, xn) = y +
          1       (xn - y)*(1.0 - exp(-dt/(1000.*tau0))) ! dt : Milliseconds
c        realpl() Implements the Transfer Function : 1./( s*tau + 1. )
c                 where s = Laplace variable.
c
          Error = ( Fil_SP - Filcurr)/100. ! Instrumentation Scaling ,
c                                          where 1volt : 100 amps
                               y0 = 0. ! Zero Fil. Curr at time=0.
           Error = Alimit( -3., 3., Error ) ! Sat. Limit
          Fil_drive = PID_2(dt, Error, init, y0, para)
c
          Fil_volt_output =realpl( Fil_volt_output, tau, Fil_drive*Gain)
c Saturation limit:
          Fil_volt_output = Alimit( 0., Gain, Fil_volt_output ) ! Max. Volt
calulate Filcurr:
          Filcurr =  Fil_volt_output/ (Resten + Rs )
c Saturation limit:
          Filcurr = Alimit( 0., 300., Filcurr ) ! Fil. PS saturation limit
          Fil_control = Filcurr
          return
          end
c
          Real Function HV_arc( Time, Trig, Offtime )
c_____________________________________________________________________

c This Routine simulates the HV arcing.
c    When "Trig" .gt. 0.5, then the arc is triggered
c     and the HV is "arced-down" for the time-out period, "Offtime".
c    At the end of the time-out period, the HV is returned to its
c     previous condition.
c
          Logical Arced
```

```
            Real dt, Trig, Offtime
            data Arced/.false./
      c
            If ( Trig.gt.0.5) then
                  Tbegin = Time
                  Arced = .true.
            endif
      c Wait for (Tbegin + Offtime) before resumming:
            If((Arced).and.(Time.ge.Tbegin+Offtime)) Arced = .false.
      c
            If ( Arced ) then
                  HV_arc = 0.
            else
                  HV_arc = 1.
            endif
            return
            end
      c
            Real Function  Activation(dt,time, Act_Begin, Act_duration, Tau_act)
      c_____________________________________________________________________
      c
      c Simulates Filament Activation
      c
            Logical Activated ! Activation Flag
            Real Act_Begin ! Time when activation begins
            Real Act_duration ! Sustained duration
            Real Tau_act ! Time constant of decay
      c
            realpl(y , tau0, xn) = y +
            1       (xn - y)*(1.0 - exp(-dt/(1000.*tau0))) ! dt : Milliseconds
      c
            if( (.not. Activated).and.( time. ge. Act_begin ) ) then
                  T0 = time
                  Activated = .true.
                  step = 1.
                  force = 1.
```

```
            endif
c
            if( (Activated).and.( time.ge. T0 + Act_duration ) ) then
                    force = 0.
                    step = 1.
            endif
                    step = realpl( step, Tau_act, force )
            Activation = step
            return
            end
c
            Real Function   Alimit(xmin, xmax, x)
c_____________________________________________________c
c Simulates  clamping and/or saturation
c   where x is the input,
c         xmin = minimum allowable value, and
c         xmax = maximum    "       "
c
            Real xmin, xmax, x, a
c
            if(x.lt.xmin) then
                    a = xmin
            else if(x.gt.xmax) then
                    a = xmax
            else
                    a = x
            endif
            Alimit = a
            return
            end
c
            Real Function   PID_1(dt, xin, init, y0, para)
c_____________________________________________________c
c Implimentes a standard PID alg.
c   When init .eq. 0, then PID is reset.
```

```
c     Integrator output is set to y0.
c      (used for bounceless transfer)
c
          Real xin, dt, xlast, para(3)
          Integer init
c
          if(init.eq.0) then
                  u = y0/ (para(1)*para(2))
                  xlast = xin
          endif
c
          u = u + xlast*(dt/1000.)
           u = Alimit(-60., 60., u )
          dx =  (xin - xlast)/(dt/1000.)
          xlast = xin
c
          PID_1 = para(1)*(xin + para(2)*u + para(3)*dx)
                  init = init + 1
          return
          end
c
          Real Function   PID_2(dt, xin, init, y0, para)
c_____________________________________________________________________c
c Implimentes a standard PID alg. for Filament Controller
c   When init .eq. 0, then PID is reset.
c    Integrator output is set to y0.
c     (used for bounceless transfer)
c
          Real xin, dt, xlast, para(3)
          Integer init
c
          if(init.eq.0) then
                  u = y0/ (para(1)*para(2))
                  xlast = xin
          endif
```

```
c
      u = u + xlast*(dt/1000.)
       u = Alimit( -3., 3., u ) ! Saturation Limit
      dx =  (xin - xlast)/(dt/1000.)
      xlast = xin
c
      PID_2 = para(1)*(xin + para(2)*u + para(3)*dx)
              init = init + 1
      return
      end
c
      Subroutine FilTmp(FF_Fil, Dt, Volts, Tfil, FilAmp,
      1 Emis, Resten, xLength, Tinf , Width, Depth, Cond)
C
C
C    MAIN TEMPERATURE MODEL :                                  C
C       Tungsten Filament Transient Analysis                   C
C        Radiation Cooling and Electrcal Heating               C
C          Programmer : J.T. Greer                             C
C                                                              C
C            FF_Fil = Forcing Function Flag                    C
C             if .True.  then Filamp = Forcing Function        C
C             if .False. then Volts  =    "       "            C
C                                                              C
C            Dt    = DELTA- Time between iterations            C
C            Volts= Voltage across filament                    C
C            Tfil = Temperature  of Filament                   C
C            FilAmp   = Filament Current                       C
C            Emis = Emission Current                           C
C            Resten = Resistance of filament                   C
C            xLength = Length of filament - CM                 C
C            Tinf = Radiant Background Temperature             C
C            Width = Width of filament , CM                    C
C            Depth = Depth of filament,  CM                    C
C                                                              C
```

```
C       Outputs :                                                       C
C            Emission Current                                           C
C            Temperature  of Filament                                   C
C       Range :   300 DEG K  <  Tfil < 3000 DEG K                       C
C
C-------------------------------------------------------------------
C
        Implicit Real*4 (A-H,O-Z)
        Parameter (pi = 3.141592654)
        Parameter ( i1 = 16 ) ! Number of Emissivity Points
        Parameter ( i2 = 28 ) ! Number of Resisivity Points
        Parameter ( i3 = 21) ! Number of Emission Points
        Parameter ( i4 = 27) ! Number of Cp Points
        Logical FF_Fil, Tcheck
        Dimension Xemis(i1),Yemis(i1),B(i1),C(i1),D(i1)
        Dimension Xres(i2), Yres(i2), B1(i2),C1(i2),D1(i2)
        Dimension Xi(i3),Yi(i3),      B2(i3),C2(i3),D2(i3)
        Dimension Xcp(i4),Ycp(i4),    B3(i4),C3(i4),D3(i4)
c
c       EMISSIVITY OF TUNGSTEN AS FUNCTION OF TEMP.(K), 12
        DATA Xemis/300., 773.,1273.,1773.,1900.,2000.,2100.,2200.,
        1         2300.,2400.,2500.,2600.,2700.,2800.,2900.,3000./
        DATA Yemis/0.032, 0.071,0.15,0.23,0.249,0.260,0.270,0.279,
        1      0.288,0.296,0.303,0.311,0.318,0.323,0.329,0.334/
c
c       ELECTRICAL RESISTIVITY OF TUNGSTEN AS FUNCTION OF TEMP.
        DATA Xres/300.,400.,500.,600.,700.,800.,900.,1000.,
        1       1100.,1200.,1300.,1400.,1500.,1600.,1700.,
        2       1800.,1900.,2000.,2100.,2200.,2300.,2400.,
        3       2500.,2600.,2700.,2800.,2900.,3000./
        DATA Yres/5.65,8.06,10.56,13.23,16.09,19.00,21.94,24.93,
        1       27.94,30.98,34.08,37.19,40.36,43.55,46.78,
        2       50.05,53.35,56.67,60.06,63.48,66.91,70.39,
        3       73.91,77.49,81.04,84.70,88.33,92.04/
C
c--- Emission Data : Electron Emission, EMIS/CM**2 ---      C
```

```
      DATA Xi/300.,1100.0,1200.0,1300.0,1400.0,1500.0,1600.0,
     1       1700.0,1800.0,1900.0,2000.0,2100.0,2200.0,
     2       2300.0,2400.0,2500.0,2600.0,2700.0,2800.0,
     3       2900.,3000./
c
      DATA Yi/0.,1.52E-13,9.73E-12,3.21E-10,6.62E-9,9.14E-8,
     1       9.27E-7,7.08E-6,4.47E-5,2.28E-4,1.00E-3,
     2       3.93E-3,1.33E-2,4.07E-2,1.16E-1,2.98E-1,
     3       7.16E-1,1.63,3.54,7.31,14.2/
c---     SPECIFIC HEAT W-MSEC/G-K'
c       Cp : Specific Heat WATTS-MSEC/GRAM-K
      DATA Xcp/300.0,400.0,500.0,600.0,700.0,800.0,900.0,
     2       1000.0,1100.0,1200.0,1300.0,1400.0,1500.0,
     3       1600.0,1700.0,1800.0,1900.0,2000.0,2100.0,
     4       2200.0,2300.0,2400.0,2500.0,2600.0,2800.0,
     5       2900.0,3000.0/
c
      DATA Ycp/133.3,135.6,137.9,140.3,142.7,145.0,147.5,149.9,
     2       152.4,154.8,157.4,159.9,162.4,165.0,167.6,170.2,
     3       172.8,175.4,178.1,180.8,183.5,186.2,188.9,191.7,
     4       192.0,194.0,198.5/
      DATA Tmin/300./,Tmax/3000./
C
C       Initialize Constants
C---     FILAMENT IN CM
C       As : Surface Area per unit length ; CM**2/CM
       As = 2.0*Width  +  Depth*(1.14159)
C       RHO : Density ;  GRAMS/CM**3
       RHO   = 19.3
C       V : Volume per unit length ;  CM**3/CM
       V_per_CM = Depth*(Width-Depth) + ((Depth/2.0)**2)*pi
C       SIGMA : Stefan-Boltzmann Constant
       SIGMA = 5.6697E-12
Check Range of Tfil:
       if( Tcheck(Tfil, Tmin, Tmax) )
```

```
          1  stop ' Error in Filtmp()... Tfil exceeds limits!'
C
          N1 = I1
Call Spline for Emissivity :
          CALL Splin(N1,Xemis,Yemis,B,C,D)
Call Spline for Resisivity :
          N2 = I2
          CALL Splin(N2,Xres,Yres,B1,C1,D1)
Call Spline for Emission :
          N3 = I3
          CALL Splin(N3,Xi,Yi,B2,C2,D2)
Call Spline for Specific Heat ; Cp
          N4 = I4
          CALL Splin(N4,Xcp,Ycp,B3,C3,D3)
C <<< Begin Iterations >>>
c  Insure Tfil .gt. Tinf :
          IF(Tfil.LE.Tinf) Tfil=Tinf+ .01
c
          T1=Tfil
          TMEAN = T1
C <<< Begin Time Step Iteration >>>
          iter = 0
20        continue
          iter = iter + 1
Calulate Emissisity :
          E1    = Splin_Eval (N1,TMEAN,Xemis,Yemis,B,C,D)
Calulate Resistivity :
          RE    = Splin_Eval (N2,TMEAN,Xres ,Yres,B1,C1,D1)
          RES   = (RE * 1.0E-6)/V_per_CM
Caluate Specfic Heat (constant pressure) :
          CP    = Splin_Eval(N4,T1,Xcp,Ycp,B3,C3,D3)
C
Calulate Qrad :
           Qrad     = As*SIGMA*E1*((TMEAN**4)-(TINF**4))
Calulate Q-conductance
```

```
          Qcon=(TMEAN-TINF)*cond
C
          if(FF_Fil) then
C     HEATING BY I**2*R !
           P = (FilAmp**2)*RES + ((EMIS/(2*xLENGTH))**2)*RES
          else
C     HEATING BY V**2/R !
                  VolCM=abs(VOLTS)/xLENGTH
           P = (VolCM**2)/RES + ((EMIS/(2*xLENGTH))**2)*RES
          endif
C Update T :
           Qsum = P - Qrad - Qcon
          T2     = T1 + ( Qsum *Dt) /(CP*RHO*V_per_CM)
c                      Tinc = ( Qsum *Dt) /(CP*RHO*V_per_CM)
c         type*,' Qsum , Tinc = ',Qsum, Tinc, V_per_CM
          TM1    = TMEAN
          TMEAN = (T2+T1)/2.0
Check for convergence :
          IF (ABS(TM1-TMEAN).LE. 0.1)  GO TO 30
          GO TO 20
30        CONTINUE
          T1=T2
          Tfil=T1
c <<< Converged >>>
Calulate the total emission current:
          EMIS=Splin_Eval(N3,Tfil,Xi,Yi,B2,C2,D2)
c
          EMIS=EMIS*As*xLENGTH
Cal: Resistance of Filament
          RE=Splin_Eval(N2,Tfil,Xres,Yres,B1,C1,D1)
          RES=(RE*1.E-6)/V_per_CM
          RESTEN=RES*xLENGTH
          if(FF_Fil) then
           Volts = RESTEN*FilAmp
             IF(Volts.LE.0.0) Volts=0.0
```

```
          else
           FilAmp=abs(VOLTS)/RESTEN
             IF(FILAMP.LE.0.0) FLAMP=0.0
          endif
c         type*, iter
          RETURN
          END
C
          Logical Function    Tcheck( T, Tmin, Tmax )
C_____________________________________________________________
Checks the range of T : Tmin to Tmax,  Tcheck = .false.
C                                      else Tcheck = .true.
C
          Tcheck = .true.
          if((T.ge. Tmin).and.(T.le.Tmax)) Tcheck = .false.
          return
          end
C_____________________________________________________________
C  Cubic Splin Alg.
C
C  The Coefficients B(I), C(I), and D(I), I=1,2,...,N are computed
C  for a cubic interpolating spline
C
C    S(X) = Y(I) + B(I)*(X-X(I)) + C(I)*(X-X(I))**2 + D(I)*(X-X(I))**3
C
C    for  X(I) .LE. X .LE. X(I+1)
C
C  INPUT..
C    N = The number of data points or knots (N.GE.2)
C    X = The abscissas of the knots in strictly increasing order
C    Y = The ordinates of the knots
C
C  OUTPUT..
C    B, C, D  = Arrays of spline coefficients as defined above.
C
```

```
C  Using  P  to denote differentiation,
C    Y(I) = S(X(I))
C    B(I) = SP(X(I))
C    C(I) = SPP(X(I))/2
C    D(I) = SPPP(X(I))/6  (derivative from the right)
C
C  The accompanying function subprogram  "Splin_Eval" can be used
C  to evaluate the Spline.
C
C-----------------------------------------------------------------------
C
      SUBROUTINE Splin (N, X, Y, B, C, D)
      INTEGER N
      Real*4 X(N), Y(N), B(N), C(N), D(N)
      Real*4 T
      Integer NM1, IB, I
C
      NM1 = N-1
      IF ( N .LT. 2 ) Return
      IF ( N .LT. 3 ) GO TO 50
C
C  Set up Tridiagonal System
C
C  B = Diagonal, D = Offdiagonal, C = Right Hand Side.
C
      D(1) = X(2) - X(1)
      C(2) = (Y(2) - Y(1))/D(1)
      DO 10 I = 2, NM1
      D(I) = X(I+1) - X(I)
      B(I) = 2.*(D(I-1) + D(I))
      C(I+1) = (Y(I+1) - Y(I))/D(I)
      C(I) = C(I+1) - C(I)
10    CONTINUE
C
C  End Conditions.  Third Derivatives AT  X(1)  AND  X(N)
C  Obtained from divided differences
```

```
C
      B(1) = -D(1)
      B(N) = -D(N-1)
      C(1) = 0.
      C(N) = 0.
      IF ( N .EQ. 3 ) GO TO 15
      C(1) = C(3)/(X(4)-X(2)) - C(2)/(X(3)-X(1))
      C(N) = C(N-1)/(X(N)-X(N-2)) - C(N-2)/(X(N-1)-X(N-3))
      C(1) = C(1)*D(1)**2/(X(4)-X(1))
      C(N) = -C(N)*D(N-1)**2/(X(N)-X(N-3))
C
C  Forward Elimination
C
15    DO 20 I = 2, N
      T = D(I-1)/B(I-1)
      B(I) = B(I) - T*D(I-1)
      C(I) = C(I) - T*C(I-1)
20    CONTINUE
C
C  Back Substitution
C
      C(N) = C(N)/B(N)
      DO 30 IB = 1, NM1
      I = N-IB
      C(I) = (C(I) - D(I)*C(I+1))/B(I)
30    CONTINUE
C
C  C(I) is now the sigma(I) of the text
C
C  Compute Polynomial Coefficients
C
      B(N) = (Y(N) - Y(NM1))/D(NM1) + D(NM1)*(C(NM1) + 2.*C(N))
      DO 40 I = 1, NM1
      B(I) = (Y(I+1) - Y(I))/D(I) - D(I)*(C(I+1) + 2.*C(I))
      D(I) = (C(I+1) - C(I))/D(I)
```

```
            C(I) = 3.*C(I)
40          Continue
            C(N) = 3.*C(N)
            D(N) = D(N-1)
            Return
C
50          B(1) = (Y(2)-Y(1))/(X(2)-X(1))
            C(1) = 0.
            D(1) = 0.
            B(2) = B(1)
            C(2) = 0.
            D(2) = 0.
            Return
            END
C_____________________________________________________________________
C
C  Splin_Eval
C
C  This Subroutine Evaluates the Cubic Spline Function
C
C  Splin_Eval = Y(I) + B(I)*(U-X(I)) + C(I)*(U-X(I))**2 + D(I)*(U-X(I))**3
C
C    where  X(I) .LT. U .LT. X(I+1), USING HORNER'S RULE
C
C  IF  U .LT. X(1) then (I = 1) is used.
C  IF  U .GE. X(N) then (I = N) is used.
C
C  INPUT..
C    N = The Number of data points
C    U = The Abscissa at which the Spline is to be evaluated
C    X,Y = The arrays of data Abscissas and Ordinates
C    B,C,D = Arrays of Spline Coefficients Computed by Splin
C
C  If  U  is not in the same interval as the previous call, then a
C  binary search is performed to determine the proper interval.
```

```
C
C
      Real*4 Function Splin_Eval(N, U, X, Y, B, C, D)
      Integer N
      Real*4 U, X(N), Y(N), B(N), C(N), D(N)
      Integer I, J, K
      Real*4 DX
      DATA I/1/
      IF ( I .GE. N ) I = 1
      IF ( U .LT. X(I) ) GO TO 10
      IF ( U .LE. X(I+1) ) GO TO 30
C
C  Binary Search
C
10    I = 1
      J = N+1
20    K = (I+J)/2
      IF ( U .LT. X(K) ) J = K
      IF ( U .GE. X(K) ) I = K
      IF ( J .GT. I+1 ) GO TO 20
C
C  Evaluate Spline
C
30    DX = U - X(I)
      Splin_Eval = Y(I) + DX*(B(I) + DX*(C(I) + DX*D(I)))
      RETURN
      END
```

```
C_____________________________________________________________________
C
C Steady State ; MAIN PROGRAM
C Simulation Model of Emission Controller Sub-System
C
C     Includes Exception Handling : HV Arcing and Filament Activation
C      Programmer J.T. Greer
C_____________________________________________________________________
C
          Implicit Real*4 (A-H,L-Z)
          Parameter (pi = 3.141592654)
          Logical HVarc ! HV arc Flag
          Logical Act_flag ! Activation Flag ; for simulation
          Logical Act_Recovery ! Use Act. recovery scheme
          Logical Act_detect ! Activation has been detected
          Logical SS_flag ! Steady State Flag ; used for Act. Recovery
          Logical Fil_FF_FLag ! Filament Forcing (for Emis model) Flag
          Logical Emission_mode ! Flag for Emission Mode( else Fil. mode)
          Logical Emission_mode_Req ! Flag for Emission Mode Request
          Logical Mode_switching ! Flag for Automatic Mode Switching
          Real Mode ! Emis/Fil mode flag for storage file
          Real Arctimes(10) ! Times that arcs will occur
          Real Para1(3) ! PID parameters for Emission Controller
          Real Para2(3) ! PID parameters for Filament Controller
c
          data iseed/12457/, iseed1/15731/, iseed2/92871/
          data Sigma_emis/0./, Sigma_fil/0./
c Real Pole Line Function:
c  Implements : 1/( s*tau + 1 ) Transfer Function, s=Laplacian Var.
          realpl(y , tau0, xn) = y +
          1       (xn - y)*(1.0 - exp(-dt/(1000.*tau0))) ! dt : Milliseconds
c
          write(5,10)
10        format(1h ,
          1 'Enter PID parameters for Emission Controller : (Pp,Pi,Pd) ',
```

```
     1 '[2,2,0] : '$)
       accept*, para1
       write(5,20)
20     format(1h ,
     1 'Enter PID parameters for Filament Controller : (Pp,Pi,Pd) ',
     1 '[1,10,0] : '$)
       accept*, para2
       write(5,30)
30     format(' Enter Sample Time for storage file (ms) [10] : ')
       accept*,sampletime
c      write(5,40)
c40    format(' Enter Initial Temp. of Filament (800 - 3000 DEG K ) : '$)
c      accept*,Tfil
       Tfil = 2667.257
c
       write(5,50)
                              Emission_mode = .false. ! Default
50     format(' Do you want Emis. Mode (otherwise Fil. Mode (Y/N) ? : '$)
       read(5,60) itxt
60     format(a1)
                              Emission_mode_Req = .false.
       if((itxt.eq.'Y').or.(itxt.eq.'y')) then
                              Emission_mode_Req = .true.
                              Emission_mode     = .true.
       endif
       type*,' Emission Mode Flag = ',Emission_mode_Req
c
       write(5,70)
70     format(' Do you want Automatic Emission Mode Switching (Y/N) ? : '$)
       read(5,60) itxt
                     Mode_switching = .false.
       if((itxt.eq.'Y').or.(itxt.eq.'y')) Mode_switching = .true.
       type*,' Mode_switching = ', Mode_switching

       write(5,80)
```

```
80      format(' Do you want HV arcing ? (Y/N) : '$)
        read(5,60) itxt
                              HVarc = .false.
        if((itxt.eq.'Y').or.(itxt.eq.'y')) HVarc = .true.
        type*,' HVarc Flag = ',HVarc
        if(HVarc) then
90       write(5,100)
100     format(' How many arcs do you want ? (1 to 10) : '$)
        accept*,num_arcs
         if((num_arcs.gt.10).or.(num_arcs.lt.1)) then
               type*,' Error in input... Try again.'
               goto 90
         endif
c
         do i=1,num_arcs
               write(5,110)
110     format(' Enter Time of HV Arc (to nearest ms) : '$)
        accept*, Arctimes(i)
         enddo
        endif ! (HVarc)
c
        write(5,120)
120     format(' Do you want Activation to occur (Y/N) ? : '$)
        read(5,60) itxt
                             Act_flag = .false.
        if((itxt.eq.'Y').or.(itxt.eq.'y')) Act_flag = .true.
        if(Act_flag) then
        write(5,130)
130     format(' Enter Act_begin, Act_duration, Time_constant (ms) : '$)
        accept*,Act_begin, Act_duration, Tau_act
                     Tau_act = Tau_act/1000. ! convert to ms
c
        write(5,140)
140     format(' Do you want Activation Recovery (Y/N) ? : '$)
        read(5,60) itxt
```

```
                        Act_Recovery = .false.
          if((itxt.eq.'Y').or.(itxt.eq.'y')) Act_Recovery = .true.
          endif
          write(5,150)
150       format(' Do you want Gaussian Randon Noise added to '
          1 'Emission (Y/N) ? : '$)
          read(5,60) itxt
          if((itxt.eq.'Y').or.(itxt.eq.'y')) then
          write(5,160)
160       format(' Enter S.D. of Noise (sigma) for Emission : '$)
          accept*, Sigma_emis
          type*,' Gaussian Random Noise will be added to Emission where :'
          type*,'   Sigma (S.D.) = ',Sigma_emis
          endif
c
          write(5,170)
170       format(' Do you want Gaussian Randon Noise added to '
          1 'Fil. Curr. (Y/N) ? : '$)
          read(5,60) itxt
          if((itxt.eq.'Y').or.(itxt.eq.'y')) then
          write(5,180)
180       format(' Enter S.D. of Noise (sigma) for Fil. Curr. : '$)
          accept*, Sigma_fil
          type*,' Gaussian Random Noise will be added to Fil. Curr. where :'
          type*,'   Sigma (S.D.) = ',Sigma_fil
          endif
c
c << Initialize >>
c
                  Open(unit=1, file='emis.dat', status='new')
          dt = 1. ! Milliseconds
                        Fil_FF_FLag = .true.
                        xLength = 2.54 ! CM ; 1"
                        Tinf = 800. ! deg K
                        Fil_curr = 52.7
```

```
                        Width = 0.1 ! CM
                        Depth = Width ! Circular Filament
                        Cond = 0. ! No conductive loss
                        Time_end = 5 000. ! ms  ; or 5. Seconds
                        Fil_SP_Default = 52.7
                       HVPS = 50. ! KV
                        Offtime = 100. ! Milliseconds; HV cutback time
c
         Call FilTmp(Fil_FF_Flag, Dt, Volts, Tfil, Fil_curr,
        1 Emis, Resten, xLength, Tinf , Width, Depth, Cond)
c
        do time = 0., Time_end, dt
c <<< Begin Simulation Model >>>
c
                Emis_SP = 1.0 ! amps
                error_emis = Emis_SP - Emis_curr
c < Determine Emission Mode Switch >
        If (Emission_mode_req)  then ! Request Emision Mode
c
          Emission_mode = .true.
         If( (Mode_switching).and.( Emis_curr.le.0.1) ) then
          Emission_mode = .false.
         endif
c
        else ! Emission Mode Request = .false.
          Emission_mode = .false.
        endif
c
c << PEV for Activation >>
                        Emis_initial = 1.0
          Call PEV_Act_detect (dt, time, Fil_curr, Emis_curr, Act_detect,
        1 Emis_initial, EmisThres)
        if((Act_Recovery).and.( Act_detect ).and.(HV_status.gt..5)) then
c If HV arcing, then don't declare activation. That's already done.
                  Emission_mode = .false. ! handle Activation
```

```
            endif
c
          If ( .not.Emission_mode_req) Emission_mode = .false. ! Fil. Mode
c
c <<< Begin Simulation >>>
c  Control mode:
          If( Emission_mode )  then
                               mode = .1 ! E-Mode Flag, for storage file
c Closed loop Emission Control
                               y0 = Fil_curr ! Bounceless Transfer
                  Fil_SP = PID_1(dt, error_emis, init1, y0, para1 )
          else
                               mode = 0. ! F-Mode Flag, for storage file
c  Fil. Set Point
                  Fil_SP = Fil_SP_Default
          endif

           Fil_SP = Alimit(0., 60., Fil_SP )
c
c                     Resten = Restance of filament ; Current time
                        If(Resten.le.0) stop' Error!...Fil. Res = zero.'
          Fil_curr = Fil_control( init2, dt, Fil_SP , Resten , para2)
!         type*,' Fsp, Filcurr = ',Fil_sp, Fil_curr
c
           Call FilTmp(Fil_FF_Flag, Dt, Volts, Tfil, Fil_curr,
          1 Emis, Resten, xLength, Tinf , Width, Depth, Cond)
c
           Emis = Alimit(0.0, 3., Emis ) ! Limited by Power Supply
           Fil_curr = Alimit(0., 200., Fil_curr )
c
c << Noise Generator >> ; Gaussian Random (white) Noise
c  Simulates instrumentation and process noise
                               ymean = 0.
           Fil_curr = Fil_curr +
          1        Gauss(iseed2, Sigma_fil, ymean)
```

```
c << HV arcing simulation >>
        if(HVarc) then
                        Trig = 0.
                do i=1,num_arcs
                        timedif = abs(time - Arctimes(i) )
                if( timedif.lt.0.1)  then
                        Trig = 1.
                endif
                enddo
c
                HV_status = HV_arc( time,  Trig, Offtime )
                HV = HVPS * HV_status
        else
                HV_status = 1.0 ! No arcs ; don't simulate any.
                HV = HVPS
        endif
c
        if(Act_flag) then
         Emis_curr = Emis * HV_status *
        1(1.+Activation(dt, time, Act_Begin, Act_duration, Tau_act)/2.)
        else
         Emis_curr = Emis * HV_status
        endif
c << Noise Generator >> ; Gaussian Random (white) Noise
c  Simulates instrumentation and process noise
                        ymean = 0.
         Emis_curr = Emis_curr +
        1       Gauss(iseed1, Sigma_emis, ymean)
c
                if(amod( time, SampleTime).eq.0.) then ! Record every 100 ms
                        activ = .2
                if(act_detect)  activ = .3
        write(5,*)
        1 time/1000., Fil_curr, Emis_curr, mode, activ,Emisthres
        write(1,*)
```

```
            1 time/1000., Fil_curr/50., Emis_curr, mode, activ,Emisthres
                  endif
c
            enddo ! time
999         continue ! end
            close(unit=1)
c Store Final Values:
            open(unit=3, file='emis_final.dat', status='new')
            write(3,*) ' Time_end = ',time
            write(3,*) ' Fil Temp = ',Tfil
            write(3,*) ' Fil Curr = ',Fil_curr
            write(3,*) ' Fil Volt = ', Volts
            write(3,*) ' Emis     = ', Emis_curr
            write(3,*) ' Restance = ', Resten
            close(unit=3)
            stop
            end
c
            Subroutine PEV_Act_detect(dt, time, Fil_curr, Emis_curr, Act_detect,
            1 Emis_initial, EmisThres)
c_____________________________________________________________________________
c PEV (Process Evaluator) for Activaion Determination and Estimation
c   of "Activation Limit"
c    Input  - FIl_curr, Emis_curr
c
c    Output - Logical ; PEV_Act_detect ; .true./.false.
c     Programmer : J.T. Greer
c
c      If Activation then Act_detect = .true., otherwise .false.
         Integer Ipass
         Real dt, time
         Real Emis_curr_est, Emis_est, Fil_curr, Emis_curr
         Logical Act_detect
         realpl(y , tau0, xn) = y +
         1       (xn - y)*(1.0 - exp(-dt/(1000.*tau0))) ! dt : Milliseconds
```

```
          data ipass/0/
          data Tau_emis/.2/! Seconds (200ms) time constant (estimated)
          data Tol/0.1/ ! 100ma Tolerance
c Initialize:
                    if(ipass.eq.0) then
                           Emis_curr_est = Emis_initial
                           ipass = 1
                    endif
c Begin ;
                    Act_detect = .false.
c << Estimate (Empirically) Emis_curr >>
          Emis_est = (9.691e-24) * ( Fil_curr ** 13.366 )
c
c < Estimate Dynamics ; Thermal Lag, etc. >
          Emis_curr_est = realpl(Emis_curr_est , tau_emis, Emis_est)
c
Calulate Act. Threshold :
                    Threshold = Emis_curr_est + Tol
                           EmisThres = Threshold
c
Check for Activation :
          if( Emis_curr. gt. Threshold ) Act_detect = .true.
c
          return
          end
```

```
c
c RLS    -    Recursive Parameter Estimator ; MAIN
c_____________________________________________________________________c
c
c Programmer: J.T. Greer
c   Shows how Process Excursions effects the RLS alg.
c     (Simulated HV arc )
c
          parameter (idim = 8)
          Character*1 GRAPH(51,101),SYMBOL,BLANK, iaver_flag
          Character*20 file1,file2
          Real u(idim), y, alpha, P(idim,idim), Cscale
          Real theta(idim), theta_ref(idim), theta_last(idim)
          character filename*20, null*20
          logical learn_these, learn, digital, correct
          integer status, iter_max, iter_required, ipred
          integer iupdates_tot, iupdates
          integer xydisp(64,64)
          integer ia(8,8)
          real patt_dat(8), upatt_dat
          real pattx(600), patty(600)
          data null/'                    '/
c
          type*,' Enter idisp (1=type theta, 0=no) : '
          accept*,idisp
          type*,' Enter the number of patterns to be used for RLS :'
          accept*,learn_these
c
          write(5,10)
10        format(' Enter pattern filename : '$)
          read(5,20) filename
20        format(a20)
                  if(filename.eq.null) then
                  open(unit=2, file='pattern_nl.dat', status='old')
                  else
```

```
                    open(unit=2, file=filename, status='old')
                    endif
c Get Learning Set:
                    num_patt = 0
          do i=4,500
          read(2,*,end=55) pattx(i), patty(i)
                    num_patt = num_patt + 1
          enddo
55        continue
          if(num_patt.eq.0) stop ' Error in data collection...STOP PROGRAM.'
          type*,num_patt,' Learning patterns loaded into memory.'
c
c <<RPE>>>
          type*,' Enter alpha : '
          accept*,beta
                    Cscale = 10000.
                    init = 0
                    sum_sqr_err = 0.
                    sum_err = 0.
                    isum = 0
          do i=4, learn_these
c
                    u(1) = patty(i-1)
                    u(2) = patty(i-2)
                    u(3) = pattx(i-1)
                    u(4) = 1.
c set u(): x,y
                    y = patty(i)
                    x = pattx(i)
                    u(5) = x
                    u(6) = x**2
                    u(7) = x**3
                    u(8) = x**4
c         call   dispv(idim, u, ' U : ')
          s = Dotp( idim, theta, u)
```

```
                diff = s-y
                        sum_sqr_err = sum_sqr_err + diff**2
                        sum_err = sum_err + diff
                        isum = isum + 1
              st = Dotp( idim, theta, theta)
c             type*,' pred, patt = ',s,y
c
c
                if(beta.le.0.) then
                        alpha = 1./float(i)
                else
                        alpha = beta
                endif
c
              call RLS(idim, theta, u, y, alpha, P, Cscale, init)
c
              write(50,*) theta, E_norm(idim, p)
              type*,' Iter = ',init,'  G1, G2, = ',gamma1, gamma2
              if(idisp.gt.0)call dispv(idim,theta, ' Theta : ')
c
              gamma2 = Tmag( idim, theta )
              gamma1 = Conv(idim, theta, theta_last)
              call Setequal( idim, theta_last, theta)
c
              enddo ! i
c Statistics:
              if(isum.ne.1)  then
               sigma = ( (sum_sqr_err - (sum_err**2)/float(isum-1)) /float(isum-1)
)
              else
               sigma = 0.
              endif
              type*,' Sigma (error) = ',sigma
check:
              do i=5,num_patt
                      u(1) = patty(i-1)
```

```
                u(2) = patty(i-2)
                u(3) = pattx(i-1)
                u(4) = 1.
c set u(): x,y
                y = patty(i)
                x = pattx(i)
                u(5) = x
                u(6) = x**2
                u(7) = x**3
                u(8) = x**4
        s = Dotp( idim, theta, u)
                y = patty(i)
        diff = abs( s - y )
        if(i.eq.learn_these) type*,' End of Learning Set'
        type*,i,' [patt:pred] = ',y,s,diff
        enddo
        type*,' Theta : ',theta
        stop' Stop Program.'
        end
c
        Real Function   E_norm(idim, p)
c_____________________________________________________c
c Eucl. Norm of Matrix p
c
        Real p(idim, idim)
c
                sum = 0.
        do i=1,idim
        do j=1,idim
                sum = sum + p(i,j)**2
        enddo
        enddo
                E_norm = sqrt( sum )
        return
        end
```

```
c
      Real Function   Tmag( idim, x )
c_____________________________________________c
      Real x(idim)
c
            sum = 0.
            summag = 0.
      do i=1,idim
            sum = sum + x(i)
            summag = summag + x(i)**2
      enddo
            if(summag.eq.0.) then
                  Tmag = sum
            else
                  Tmag = sign(1.,sum)*((sum**2)/summag)
            endif
      return
      end
c
      Subroutine      Setequal( idim, x, y)
c_____________________________________________c
      Real x(idim), y(idim)
c
      do i=1,idim
            x(i) = y(i)
      enddo
      return
      end
c
      Real Function   Conv(idim, x1, x2)
c_____________________________________________c
      Real x1(idim), x2(idim)
c  Used to detect convergence
```

```
                sum = 0.
                summag = 0.
        do i=1,idim
                sum = sum + ( x1(i) - x2(i) )**2
                summag = summag + x1(i)**2
        enddo
                if(summag.eq.0.) then
                        Conv = sum
                else
                        Conv = sum/summag
                endif
        return
        end
c
        Real Function   Dotp( idim, u, v)
c_______________________________________________________c
c Performs Dot Product of u and v vectors
c
        real u(idim), v(idim)
c
                sum = 0.
        do i=1,idim
                sum = sum + u(i)*v(i)
        enddo
                Dotp = sum
        return
        end
c
        Real Function   Vmax(idim, w)
c_______________________________________________________c
c Finds element in vector w
c
        real    w(idim), sum
c
                sum = w(1)
```

```
        do i=2,idim
                sum = amax1( sum, w(i) )
        enddo
                Vmax = sum
        return
        end
c
        Real Function   Vmag(idim,w)
c_____________________________________________________c
c Eucl. Norm of vector w
c
        real    w(idim), sum
c
                sum = 0.
        do i=1,idim
                sum = sum + w(i)**2
        enddo
                Vmag = sqrt( sum )
        return
        end
c
c
c
        Subroutine      Dispv(N,B, LABEL)
c_____________________________________________________c
C Displays a Nx1 vector with label
c
        IMPLICIT        REAL    (A-H,O-Z)
        REAL    B(101)
        INTEGER N
        CHARACTER*10    LABEL
C
        TYPE*,' '
        IF(N.LT.5) THEN         ! use simple format
                WRITE(5,*) LABEL,(B(M),M=1,N)
```

```
            ELSE ! use multi-line format :
                    ICOL = 1
   10        CONTINUE
                    IEND = ICOL + 4
                    IF(IEND.GT.N) IEND=N
             IF(ICOL.EQ.1) THEN
                    WRITE(5,20) LABEL,(B(J),J=ICOL,IEND,1)
             ELSE
                    WRITE(5,30) (B(J),J=ICOL,IEND,1)
             ENDIF
                    ICOL = ICOL + 5
            IF(IEND.LT.N) GO TO 10
           ENDIF
           RETURN
   20      FORMAT(1H ,A10,1X,10(G12.4,1X))
   30      FORMAT(1H ,11X,10(G12.4,1X))
           END
   c
   c
   c_____________________________________________________________________
c
   c
   c   Recursive Least Squares Method
   c    Reference Ljung and Soderstrom: Theory and Practice of
   c                                    Recursive Identification, 1983
   c_____________________________________________________________________
c
   c
   c Programmer: J.T. Greer
   c Date: June 30, 1988
   c
           Subroutine RLS(idim, theta, u, y, alpha, P, Cscale, init)
   c
           integer init
           real theta(idim), u(idim), y, P(idim,idim), alpha
           real a(10), b(10) ! Temp. storage
   c
```

```
c Initialize :
          if(init.eq.0) then
                    do i=1,idim
                    do j=1,idim
                              P(i,j) = 0.
                              if(i.eq.j) P(i,j) = Cscale
                    enddo
                              theta(i) = 0.
                    enddo
          endif
c
                    init = init + 1
c Begin Recursive Alg:
Calulate a,b,c0:
                    c0 = 0.
          do i=1,idim
                    a(i) = 0.
                    b(i) = 0.
          do j=1,idim
                    a(i) = a(i) + P(i,j)*u(j)
                    b(i) = b(i) + P(j,i)*u(j)
          enddo
                    c0 = c0 + a(i)*u(i)
          enddo
                    c0 = 1./(c0 + 1./alpha)
calulate Pred:
                    Pred = 0.
          do i=1,idim
                    Pred = Pred + theta(i)*u(i)
          enddo
c Update theta():
          do i=1,idim
                    theta(i) = theta(i) + a(i)*( y - Pred )*c0
          enddo
c Update P(,):
```

```
            do i=1,idim
            do j=1,idim
                    P(i,j) = P(i,j) - a(i)*b(j)*c0
            enddo
            enddo
c
            return
            end
c
c Gaussian Number Generator
c   Two variable "Even" to "Normal" Transformation
c      Gauss() has output sequence whose distribution
c        models a "Gaussian White" Noise Generator
c        where : sigma = S.D.
c                ymean = Average value
c
c     Programer : J.T. Greer
c
            real function gauss(i, sigma, ymean)
            parameter (pi = 3.141592654)
c x1 and x2 are independent and have "even" distributions
                    x1 = ran(i)
                    x2 = ran(i)
c  y1 and y2 are independent and have "normal" distributions
c           y1 = sqrt( -2.0 * log( x1 ) )*( cos( 2.0*pi*x2 ) )
            y2 = sqrt( -2.0 * log( x1 ) )*( sin( 2.0*pi*x2 ) )
            gauss = y2* sigma + ymean
            return
            end
```

```
C_____________________________________________________________C
C Plotting Package  - Interactive    ; MAIN PROGRAM
C  Compatible with several formats
C       Programmer : J.T. Greer
C_____________________________________________________________C
C
      parameter (imax = 1000)
      parameter (idimy = 10)
      parameter (pi = 3.141592654)
      logical grid_flag ! plotting flag
      logical bar_flag  !   "
      logical Sflag, XYplot
      integer idev ! device number
      integer idevice ! plot terminal device
      integer iplotpara(idimy) ! plot parameters
      integer iplot(idimy)     !  "   flag
      character filename*20, ofilename*20, label*60, idate*9,iterm_in*3
      character plotlabel*30, xlabel*30, ylabel*30
      character file1*20,file2*20
      character*6 iplt_dev(11) ! plotting device number
      byte ilabel(60)
      equivalence (label, ilabel)
      real x(imax),y(imax), z(imax)   ! x=real input, y&z used for plots
      real yp(imax, idimy)
      real xdata(imax)        ! used in DGP
      real xindex(imax)       ! plot var
      data iplotpara/10*0/! plot parameters
      data iplot/10*0/! activates plots
      data XYplot/.false./
      data plot/.true./ , idev/5/, grid_flag/.false./
      data icur/0/, delt/100.e-6/
      data iplt_dev/'TK4006','TK4010','TK4012','TK4013','TK4015','TK4105',
     1             'TK4107','HP2623','HP2627','HP2647','HP2648'/
c
      iseed = int( secnds(0.0) )
```

```
40        continue ! begin command interpretation
c <<
          call Menu_full (imax, idev )
c >>
45        continue
          write(idev,60)
60        format(' Enter command : '$)
          read(5,70, end=9999) iterm_in
70        format(a3)
          call upper( iterm_in )
c
          call  command_parse( iterm_in, iselect)
c
          goto( 100, 200, 300, 400, 500, 600, 700, 800, 900,
          1 9999,9999) iselect
c              'TER','REA','PLO','PPA','GRI','BAR','SEL','NOR','Pxy'
c         1 'EX ','EXI'/
c
          write(idev,*)' Error in command, re-enter.'
          goto 40
c-------------------------------------------------------c
100       continue
c <<< Get Device , Plotting Terminal >>>
                  idevice = 0 ! assume default ; 4107
          type*,' Index       Terminal Type'
          type*,' -----       -------------'
          do i=1,11
          type 110,i,iplt_dev(i)
110       format(1h ,1x,i4,10x,a6)
          enddo
          type*,' '
          write(5,120)
120       format(1h ,'Enter Terminal index : '$)
          accept*,idevice
          if((idevice.lt.1).or.(idevice.gt.11)) then
```

```
                type*,' ?  Error in device type. Default remains.'
                goto 45
        endif
        type*,' Device is set to ',iplt_dev(idevice)
                goto 45
c
c-------------------------------------------------------c
c <<< Read data file >>>
200     continue
                idata_size = 1
        write(idev,210)
210     format(' Select Data Format : '/
        1'   1 - [       ydata ]'/
        2'   2 - [xdata, ydata ]'/
        3'   3 - [     y1, y2, ... yi ]'/
        4'   4 - [ x,  y1, y2, ... yi ]'/
        5'  Enter data type : '$)
        accept*,idata_type
        if((idata_type.lt.1).or.(idata_type.gt.4)) then
                type*,' Error in input! Return to main.'
                goto 40
        else if((idata_type.eq.3).or.(idata_type.eq.4)) then
                Write(idev,220)
220     format(' Enter number of y(i)''s  : '$)
                accept*,idata_size
                if(idata_size.gt.idimy) then
                type*,' Data size too big!  Re-enter.'
                goto 40
                endif
        endif
c
        do i=1,idimy
         if(i.le.idata_size) then
                iplot(i) = 1
         else
```

```
                iplot(i) = 0
             endif
           enddo
           write(idev,230)
230        format(' Enter data filename : '$)
           read(5,240,end=999) ofilename
240        format(a20)
c
c
           if(ofilename.eq.'ex') goto 40
           open(unit=1, file=ofilename , status='old' ,err=998)
c Read data file :
                   npts = 0
           do 250 i = 1,imax
            if(idata_type.eq.1) then
                   read(1,*,end=290) yp(i,1)
                   npts = npts + 1
                   x(npts) = npts
            else if(idata_type.eq.2) then
                   read(1,*,end=290) x(i), yp(i,1)
                   npts = npts + 1
            else if(idata_type.eq.3) then
                   read(1,*,end=290) (yp(i,k),k=1,idata_size)
                   npts = npts + 1
                   x(npts) = npts
            else if(idata_type.eq.4) then
                   read(1,*,end=290) x(i), (yp(i,k),k=1,idata_size)
                   npts = npts + 1
            endif
250        continue
290        continue
           close( unit=1 )
           if(npts.lt.2) goto 295
c Set data types :
           write(idev,*)npts,' data points read.'
```

```
                goto 40
295     continue ! error, not enough points
        write(idev,*) ' Error in reading file... Return to main.'
                goto 40
c-------------------------------------------------------c
c <<< Plot data >>>
300     continue
315     continue
        write(idev,320)
320     format(' Screen plot or  File (s/f) : '$)
        read(5,325,end=999) itxt
325     format(a1)
        if(itxt.eq.'ex') goto 40
        if((itxt.eq.'s').or.(itxt.eq.'S')) then
                Sflag = .true.
        else if((itxt.eq.'f').or.(itxt.eq.'F')) then
                Sflag = .false.
        else
                write(idev,*)' ? Command error ... re-enter.'
                goto 315
        endif
c
!  plot
c
        if(XYplot) then ! XY plot
                xmin = yp(1,iselx)
                xmax = xmin
                ymin = yp(1,isely)
                ymax = ymin
        do i=1,npts
                        xmin = amin1( xmin, yp(i,iselx) )
                        xmax = amax1( xmax, yp(i,iselx) )
                        ymin = amin1( ymin, yp(i,isely) )
                        ymax = amax1( ymax, yp(i,isely) )
        enddo
```

```
          else ! standard plot
                  xmin = x(1)
                  xmax = xmin
          do i=1,idimy
           if(iplot(i).gt.0) then
                  ymin = yp(1,i)
                  ymax = ymin
                  goto 345
           endif
          enddo
345       continue
          do i=1,npts
                          xmin = amin1( xmin, x(i) )
                          xmax = amax1( xmax, x(i) )
                  do j=1,idimy
           if(iplot(j).gt.0) then
                          ymin = amin1( ymin, yp(i,j) )
                          ymax = amax1( ymax, yp(i,j) )
           endif
                  enddo
          enddo
          endif ! end standard plot
c
          write(idev,*) ' xmin = ',xmin,'   xmax = ',xmax
          write(idev,*) ' ymin = ',ymin,'   ymax = ',ymax
          write(idev,*) ' '
          write(idev,342)
342       format(' Do you want auto-scaling (y/n) : '$)
          read(5,325,end=999) itxt
          if(itxt.eq.'ex') goto 40
          if( (itxt.eq.'Y').or.(itxt.eq.'y')) then ! Auto scale
c "Y" Auto-scaling
c                 y0 = round(ymin)
                   y0 = (ymin)
c                 yend = round(ymax)
```

```
                    yend = (ymax)
c "X" Auto-scaling
c                  x0 = round(xmin)
                    x0 = (xmin)
                    xend = (xmax)
c                  xend = round(xmax)
          else ! Not auto-scaled
c input limits
          write(idev,360)
360       format(' Enter Xmin and Xmax : '$)
          accept*, x0, xend
          write(idev,350)
350       format(' Enter Ymin and Ymax : '$)
          accept*, y0, yend
          endif
c
           write(idev,365)
365       format(' Enter plot label Heading (up to 30 chars) : '$)
           read(5,370,end=999) plotlabel
370       format(a30)
           if(plotlabel.eq.'ex') goto 40
           write(idev,375)
375       format(' Enter X label (up to 30 chars) : '$)
           read(5,370,end=999) xlabel
           write(idev,378)
378       format(' Enter Y label (up to 30 chars) : '$)
           read(5,370,end=999) ylabel
c<<< Begin Plotting >>>
          if( ((xend-x0).eq.0.).or.((yend-y0).eq.0) ) then
                stop' Error in plot setup...Xend=X0.'
          endif
          if(Sflag) then
           call screen_setup(idevice, ierror)
           call lib$put_output(char(27)//'%!0') ! put terminal in TEK mode
           call lib$put_output(char(27)//'[2J') ! erase screen
```

```
            else
             call comprs
            endif
c
            call page(11.0, 8.5)
c           call futura
            call shdchr(90., 1, .002, 1)
            call height(0.17)
            call area2d( 8.0, 6.0 )
c Label plot:
            call headin( ' $', 100, 1., 2)
             call date( idate )
             label(1:7) = 'Date : '
             label(8:16) = idate
                    label(17:33) = '     Title  :    '
             label(34:59) = plotlabel
             label(60:60) = '$'
            call headin( ilabel , 100, 1., 2)
            call xintax
                          label = xlabel
                          lenght = len(xlabel) +1
                          label(lenght:lenght) = '$'
                   call xname( ilabel, 100)
                          label = ylabel
                          lenght = len(ylabel) +1
                          label(lenght:lenght) = '$'
                   call yname( ilabel, 100)
c Scale axis:
                                  ystep = (yend - y0)/5.0
                                  xstep = (xend - x0)/5.0
            call graf (x0, xstep, xend, y0, ystep, yend )
            call frame
c           call bars( -.1)
c
c Stuff Ploting Arrays:
```

```
C
          if(XYplot) then ! Plot Yi vs Yj
           Do J=1,npts
                   z(j) = yp(j,iselx)
          if(z(j).lt.x0) z(j) = x0
          if(z(j).gt.xend) z(j) = xend
                   y(j) = yp(j,isely)
          if(y(j).lt.y0) y(j) = y0
          if(y(j).gt.yend) y(j) = yend
           enddo
c Plot :
                  call linear
                  if(bar_flag) call bars(-.1)
                  if(grid_flag) call grid(1,1)
c           call sclpic(.01) ! scale size of marker
                  icur = iplotpara(iselx)
            CALL CURVE(z, y ,npts, icur)
c           call sclpic(1.0)
                  XYplot = .false.
c
          else ! standard plot
          Do i=1,idimy ! plot all curves
          if( iplot(i).gt.0 ) then
           Do J=1,npts
                   y(j) = yp(j,i)
          if(y(j).lt.y0) y(j) = y0
          if(y(j).gt.yend) y(j) = yend
           enddo
c Plot :
                  call linear
                  if(bar_flag) call bars(-.1)
                  if(grid_flag) call grid(1,1)
c           call sclpic(.01) ! scale size of marker
                  icur = iplotpara(i)
            CALL CURVE(x, y ,npts, icur)
```

```
c          call sclpic(1.0)
          endif
          enddo ! finished curves
          endif
c First define secondary axis ;
c         call yintax
c         call ygraxs( -360.0, 90.0, 180.0, 6.0, 'Phase angle$',-100,8.0,0.0)
c         call dash
c         CALL CURVE(xindex, z ,npts, icur)
          call endpl(0)
          call donepl
          call lib$put_output(char(27)//'%!2') ! put terminal in EDIT mode
          call lib$put_output(char(27)//'[2J') ! erase screen
          goto 40
c-------------------------------------------------------c
c <<< PPA >>> ! Plotting Parameters
400       continue
          write(idev,*)' ICUR is used by "CURVE" to plot arrays.'
          write(idev,*)'  if ICUR=0, points connected, no symbols.'
          write(idev,*)'  if ICUR=n, points connected, symbols every ',
          1 'nth point.'
          write(idev,*)'  if ICUR=-n, points not connected, symbols every nth.
,
410       continue
          write(idev,420)
420       format(' Enter curve to set ICUR(i)  [0=exit] : '$)
          accept*,isel
          if(isel.lt.1) goto 45
          write(idev,430) isel, iplotpara(isel)
430       format(' Enter ICUR(',i2,') = ',i1,',   Enter New value  : '$)
          accept*,iplotpara(isel)
          goto 410
c end of ICUR set .
c <<< Set Grid : >>>
500       continue
          write(idev,*)' Do you want GRID ? (y/n)'
```

```
          read(5,325,end=999) itxt
                  grid_flag = .false.
          if((itxt.eq.'y').or.(itxt.eq.'Y')) grid_flag = .true.
          write(idev,*)' Grid flag = ',grid_flag
          goto 45
c <<< Set Bars : >>>
600       continue
          write(idev,*)' Do you want BARS ? (y/n)'
          read(5,325,end=999) itxt
                  bar_flag = .false.
          if((itxt.eq.'y').or.(itxt.eq.'Y')) bar_flag = .true.
          write(idev,*)' Bar_flag = ',bar_flag
          goto 45
c
c <<< Plot curve select : >>>
700       continue
          write(idev,*)' Select the curves you want plotted :'
          write(idev,*)'  Iplot(i) = 1   ! Curve "i" will be plotted.'
          write(idev,*)'  Iplot(i) = 0   ! Curve "i" will not be plotted.'
710       continue
          write(idev,720)
720       format( ' Enter curve selection (0=exit) : '$)
          accept*,isel
          if(isel.lt.1) goto 45
          write(idev,740) isel, iplot(isel)
740       format(' Iplot(',i2,') = ',i1,',   Enter New value  : '$)
          accept*,iplot(isel)
          goto 710
c
c <<< Normalize >>>
800       continue
          Do i=1,idimy ! plot all curves
          if( iplot(i).gt.0 ) then
                          zmax = abs(yp(1,i))
           Do J=1,npts
```

```
                zmax = amax1( zmax, abs(yp(j,i)) )
             enddo
                if(zmax.eq.0.) goto 820
             Do J=1,npts
                 yp(j,i) = yp(j,i)/zmax
             enddo
            type*,' Normalize curve#',i,'  ymax = ',zmax
            endif
820         continue
             enddo
            type*,' Finished Normalizing.   Return to main.'
            goto 45
c
c <<< Plot XY >>>
900         continue
            if(xyplot) then
            type*,' Warning : Plot xy flags alreay  = TRUE'
            type*,'  XYPLOT_flag : FALSE '
            xyplot = .false.
            else
            write(idev, 910)
910         format(' Enter curve select for X axis : '$)
            accept*,iselx
            write(idev, 920)
920         format(' Enter curve select for Y axis : '$)
            accept*,isely
            xyplot = .true.
            endif
            goto 45
c
998         continue ! error message
            write(idev,*)'? Error in filespec ; return to main.'
            goto 45
c
999         continue ! exit, ^z detected.
```

```
        goto 45
9999    continue ! end program
        call lib$put_output(char(27)//'%!2') ! put terminal in EDIT mode
        call lib$put_output(char(27)//'[2J') ! erase screen
        stop
        end
c
        Real Function   Round(x)
        real x
        if(x.gt.0) then
                round = 1. + float(int(x+0.5))
        else if(x.lt.0) then
                round = -1. - float(int(x-0.5))
        else
                round = 0.
        endif
        return
        end
c
        Subroutine      Menu_full (imax, idev )
c
        integer idev, imax
        write(idev,10) imax
10      format(1h ,/'[  Plotting Package - ',i7,' points max ] '/
        1'  Command       Discription'/
        1'  ------------------------------------------------'/
        1'  TER  -  Terminal Device Select (Tek 4107 default)'/
        1'   Rea -  Read data file'/
        1'   Plo -  Plot data array'/
        1'    PPA   - Plotting Parameters ; Grid, curve, etc.'/
        1'    Grid  - Draw Grid.'/
        1'    Bars  - Bars drawn instead of lines.'/
        1'    SEL   - Select which curves to plot.'/
        1'    Nor   - Normalize selected curves.'/
        1'    Pxy   - XY plot (Yi vs Yj).'/
```

```
          1'   (^Z or Exit - exit program)'/)
c
          return
          end
c
          Subroutine      command_parse( iterm_in, iselect)
c
          parameter (ncoms = 11)
          character iterm_in*3
          character*3     command(ncoms)
          data command/'TER','REA','PLO','PPA','GRI','BAR','SEL',
          1'NOR','PXY','EX ','EXI'/
c
c Look for command :
          do i=1,ncoms
                  if( command(i).eq. iterm_in ) then ! found
                          iselect = i
                          return
                  endif
          enddo
can not find command :
                          iselect = 0
                          return
          end
c
           Subroutine     screen_setup(idevice)
c Set up Screen plot :
          integer idevice
          goto(10,20,30,40,45,50,60,70,80,90,100), idevice
          call tk41(07) ! default
          return
10        call tk4006(960)
          return
20        call tk4010(960)
          return
```

```
30        call tk4012(960)
          return
40        call tk4013(960)
          return
45        call tk4015(960,1)
          return
50        call tk41(05)
          return
60        call tk41(07)
          return
70        call hp2623
          return
80        call hp2627(0)
          return
90        call hp2647
          return
100       call hp2648
          return
          end
c
          subroutine upper( chr )
c Puts chr in uppercase letters only
          character*3 chr
c
          do 10 i=1,3
                  ival = ichar( chr(i:i) )
                  if(ival.gt.95) chr(i:i)= char( ival - 32 )
10        continue
          return
          end
```

VITA

J. T. (Jesse Thomas) Greer was born in Natchez, Mississippi, on March 30, 1952. His father, Jesse N. Greer, relocated to Clinton, Tennessee, in 1954. J. T. graduated from Clinton High School in 1970 and graduated from Tennessee Institute of Electronics in 1972. From 1972 to 1976, he owned and managed a television and appliance sales and service business. In 1978, he began working for Union Carbide Corporation in Oak Ridge, Tennessee. His work and interest in process control and electronics led to his pursuit of a B.S. degree in Electrical Engineering; the degree was awarded in December 1984 by the University of Tennessee. In January 1985, he began pursuing an M.S. degree in Electrical Engineering at UT. J. T. is currently working for Martin Marietta Energy Systems, Inc., in Oak Ridge, where he is involved in research and applications of artificial intelligence and knowledge capture techniques. Future plans, the Lord willing, include additional graduate-level education and a graduate teaching career.