Table of Contents

Table of Contents

CHAPTER 1

Introduction

Welcome to the LIGO Single-Mode Acquisition Code!

SMAC is a resonant-cavity interferometer simulator. It takes a set of system parameters (mirror reflectivities, cavity lengths, modulation frequencies, servo gains) and a set of initial conditions (mirror positions and velocities) and generates outputs (demodulated voltages and internal fields) that closely reproduce the outputs an actual interferometer with the same parameters would generate. SMAC produces time traces, showing fringes and fields as the simulated interferometer moves in and out of resonance, much as an oscilloscope does in the laboratory. SMAC works in the time domain, generating open- and closed-loop pseudo-data in the form of signal traces vs. time for essentially any cavity parameter. SMAC can be used to generate frequency-domain transfer functions as well.

SMAC has 2 main advantages, compared to a real interferometer. The first is that it is easily reconfigured, so that new mirror parameters, new cavity lengths, new modulation schemes or new controllers can be rapidly inserted, debugged, and evaluated, without requiring any hardware construction. The second is that SMAC provides details about its response that are not available from real hardware, such as the true amplitude and phase of any field in the cavity, or the inertial position of any mirror. It can separate sidebands from the carrier. These features make it a useful platform for conducting experiments and refining designs that later can be implemented on the real hardware.

SMAC has certain limitations, most of which derive from its primary purpose, which is the design of acquisition controllers for LIGO length control. Some of the current limitations will be removed later.

- It is limited in the interferometer topologies it can handle those currently used in the LIGO project, namely (1) Fabry-Perot interferometers, (2) Coupled cavity interferometers, (3) Recombined interferometers, and (4) Recycled interferometers.
- It is limited to single-mode representation of the light propagating in the cavities. The interferometer is assumed to be well aligned in angle, so that only the TEM_{00} mode is present.
- It is limited in the noise terms it currently incorporates which includes mirror forces and laser phase noise (although other noise sources such as electronics noise or frequency noise could be added in an indirect way, .e.g. shot noise could be added to the demodulator output using a nonlinear control subroutine).

SMAC is not limited in its dynamic range: it is accurate for arbitrary displacements of the interferometer mirrors. It uses a non-linear set of equations that are valid at any operating condition, even very far from resonance.

SMAC provides a choice of temporal frequency resolutions. For cases where motions of the interferometer mirrors are slow compared to the lifetime of the light in the cavity, a static or adiabatic set of cavity equations can be specified which ignores the optical dynamics. For cases where the mirror motions or source phase is varied at high bandwidth, a dynamic set of cavity equations is used. These afford arbitrarily high frequency resolution, at the expense of increased computation time. For multiple-cavity configurations, a mixed static and dynamic set of equations provides a good compromise between frequency resolution and execution time.

SMAC incorporates a graphical user interface to simplify the process of specifying its various parameters, initial conditions, analysis types and operating modes. Data can be entered interactively in several screens, or it can be accessed from datasets that the user can setup or modify external to SMAC. It produces graphical output, in the form of time traces or transfer functions, as specified by the user. SMAC runs in the Matlab engineering environment, so all of its parameters and data (input and output) are available for pre- and post-processing as the user may wish.

This manual is intended to introduce the user to SMAC and its application in typical engineering tasks. The next chapter introduces its capabilities in fairly general terms. The succeeding chapters walk through the SMAC screens, explaining how to set them up, how to use them, and what the different variables mean. SMAC comes with a set of default datasets, which describe standard interferometer configurations (the LIGO gravity wave antenna, the various 40-meter configurations). These are discussed.

A companion report, Mathematical Description of the LIGO Single-Mode Acquisition Code, provides a detailed mathematical description of the simulation, and will be useful for people wishing to extend or modify the code. For users who do not wish to use the SMAC graphical user interface, command-line driven codes are available.

CHAPTER 2

Overview of Capabilities

Before running a simulation, the user must define the interferometer hardware -- its sources, mirrors and detectors. If closed-loop response is desired, the interferometer software (its controllers) must be selected. Modulation schemes must be defined as well, and various SMAC simulation modes must be set. The SMAC user provides this information interactively, through a sequence of graphical menus. These menus include data entries for each parameter, buttons for selecting modes and initiating actions, and links for specifying signal connections.

The mechanics of interacting with SMAC through this graphical user interface are described in later chapters. This chapter provides context, introducing the available options and capabilities, and defining the parameters the user will need to specify.

Interferometer Description

Definition of the interferometer hardware begins with the specification of its basic configuration. This includes its layout: what component where. It includes source and mirror properties, such as wavelength and mirror transmissivity. And it includes modulation parameters, such as modulation frequency, amplitude and depth, number of modulation frequencies, and number of higher-order sidebands. These parameters are introduced in this section.

This version of SMAC provides explicit support for only 2 of the interferometer configurations built by the LIGO project, namely the single Fabry-Perot and the power recycled interferometers. These are illustrated in Figure 1. Other configurations, such as the recombined, Michelson, or coupled-cavity interferometers can be created by choice of input parameters. For instance, setting the transmissivity of the recycling mirror (RM) to 1 results in a recombined interferometer. Further examples are provided in the default parameter sets, described later in this manual.

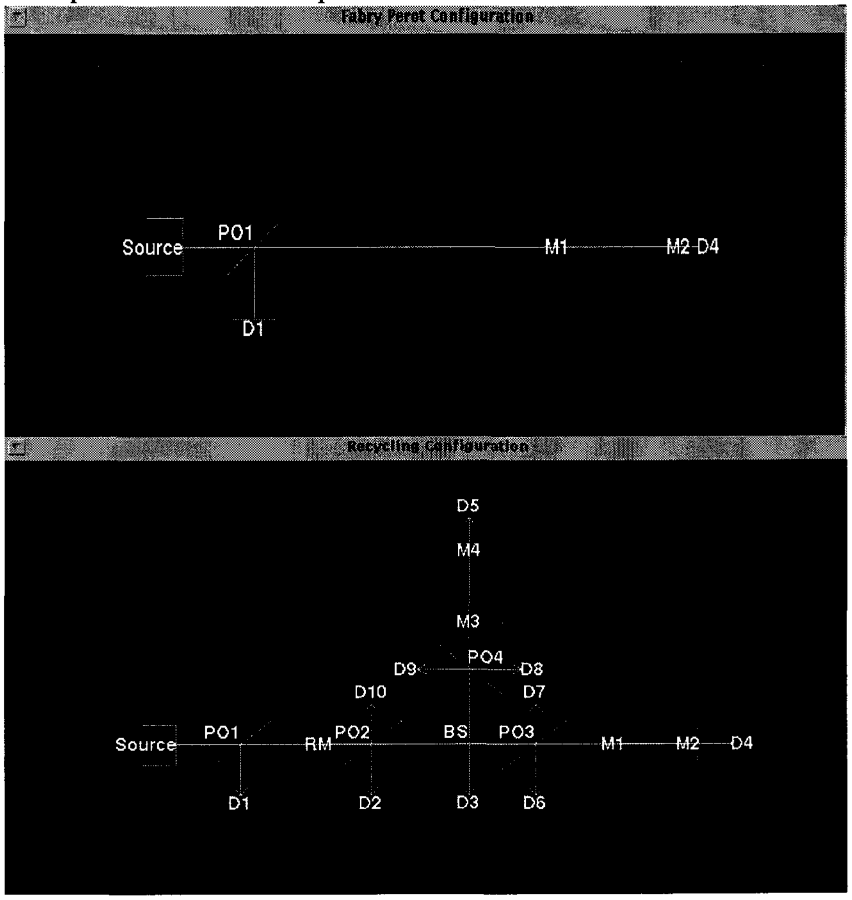


FIGURE 1. SMAC-supported interferometer configurations.

Once a configuration is chosen, the user is asked to provide values for the defining parameters. These parameters include:

- Source (carrier) wavelength in meters.
- Source power in watts. Total power as input into the (lossless) phase modulator. Power levels for the carrier and sidebands are computed by SMAC.
- Nominal front and back cavity lengths in meters. By "nominal" we mean the lengths the cavities have when all the mirror displacement states are zero. These lengths define the reference position of all of the mirrors; the displacement states are the deviations from these reference positions.

  It is currently assumed that the back cavities have the same nominal length. Lengths of all cavities may be adjusted slightly by the program, to ensure that they are a multiple of the wavelength, and to enforce timing constraints.
- Asymmetry length in meters. This parameter defines the difference between the optical path from the recycling mirror (RM) to the arm cavity front mirrors (M1 and M3).
- Power transmissivity and loss for the cavity mirrors. Reflectivity is computed as the greater of R=1-T-L or 0. If transmissivity is set to 1, the mirror effectively is removed, which is one way to alter a baseline configuration.
- Power transmissivity and loss for the detector pickoffs. Reflectivity is computed as the greater of R=1-T-L or 0, unless the transmissivity is set to 1. If the transmissivity is set to 1, a perfect pickoff is assumed and the reflectivity is also set to 1.
- Power transmissivity and loss for the beam splitter. Reflectivity is computed as the greater of R=1-T-L or 0, as for the cavity mirrors.
- Cavity subsample constant. This parameter is used to define the temporal frequency resolution of a particular simulation run, and is discussed in the later section on computational modes.

An important parameter that is hard-wired into SMAC is the value for the speed of light, "c". The value used by SMAC is 2.99792×10^8 meters/sec. Models for high finesse cavities are extremely sensitive to this parameter so care should be taken when comparing other models with SMAC to verify that the comparison model uses the same approximation.

LIGO interferometers use radio-frequency phase modulation and synchronous detection to develop signals proportional to the phase change of the light in the interferometer cavities. The sensitivity of each signal to particular modes of motion of the mirrors, and the robustness of the signals, depends on exactly how the light is modulated, as well as the configuration of the hardware.

The LIGO phase modulation is actuated by a Pockels cell located after the laser and before the first cavity mirror. The effect of the modulation is modeled in SMAC by decomposing the input field into multiple separate fields, each with slightly different wavelength: a carrier frequency plus symmetric sidebands (Ref xxx). The carrier wavelength is that of the laser, and has the most power. Its amplitude is equal to the $J_0(\Gamma)$ Bessel function times the source amplitude (here Γ is the modulation depth). The sideband wavelengths differ by $\pm$ the modulation frequency times the sideband order divided by the carrier frequency, a very small number. There are an infinite number of these sidebands, whose amplitude is $J_i(\Gamma)$, where i is the sideband order. For values of Γ less than 0.1, only the J_1 sidebands will significantly affect the demodulated phase signals. For higher values of Γ, the J_2 and J_3 sidebands can also affect the detected signals.

A modulation scheme is described for SMAC using the following parameters:

- Modulation frequency. SMAC offers the user the choice of using a specific modulation frequency, or of computing an optimal modulation frequency -- that is, a modulation frequency that causes the sidebands to be resonant in the average Michelson arm length. This choice defines the Pound-Drever-Hall detection scheme (ref xxx).
- Optimal modulation frequency multiplier. An optimal modulation frequency can be increased by integral multiples without affecting the resonance condition -- this reduces 1/f noise and is commonly done.
- Modulation depth. This directly affects the energy in the sidebands. For larger values, it may be necessary to compute higher-order sidebands to model the output signals accurately.
- Demodulation reference signal phase for each detector. The electronic modulation reference waveform can be adjusted in phase relative to the phase of the light at individual detectors. In some cases, this adjustment directly controls the sensitivity of particular signals to particular mirror degrees of freedom.
- Number of sidebands carried. The time-varying frequency of the source light induced by the modulation is expressed in SMAC as the sum of separate fields at slightly different frequencies. The light at the source frequency is termed the "carrier," and is generally the strongest field. The other fields are termed "sidebands." For relatively shallow modulation depths, only the first-order sidebands need be carried. For larger modulations depths (of about 0.5 or greater), higher-order sidebands can persist and may show up in the demodulated output. Sidebands of order J1 to J3 may be included in the simulation.

- Number of modulation frequencies. SMAC currently supports 1 carrier, up to 2 modulation frequencies, and sidebands up to J3. The user can select a second set of modulation parameters independently from the choices made for the first set.
- Parallel or serial modulation. When multiple modulation frequencies are specified, exactly how the modulation frequencies are implemented must also be specified. The parallel approach does not introduce any cross products. A limitation of SMAC for the parallel approach is that it does not properly handle sideband frequencies that overlap. This is because the demodulation calculations are done at each frequency independently (e.g. if the J2 sideband of the first modulation frequency overlaps with the second modulation frequency, independent calculations at the same frequency will result). This should not be a problem since interferometer hardware is normally chosen to avoid this situation. Serial modulation *utilizes multiple* Pockels cells in series. The downstream cells introduce modulation, not only of the carrier, but of the sidebands introduced by the upstream cells. Cross-products are then a factor in the demodulation. Presently SMAC does not have the capability to simulate the cross-terms but this feature could be added at a later date (this should not be a problem though since interferometer hardware is normally chosen to avoid this situation).

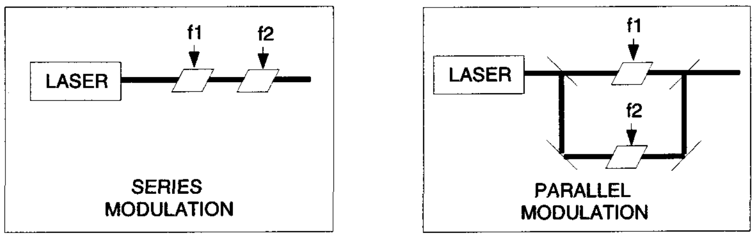


FIGURE 2. Parallel and serial modulation.

The final step in defining the interferometer hardware is to specify initial conditions for the mirror and source states, to select mirror dynamics mode, and to define noise sources. The relevant parameters are:

- Mirror dynamics option. SMAC integrates the mirror mechanical dynamical equations together with cavity optical and servo filter dynamical equations to compute the total simulated response. SMAC offers the choice of free-mass dynamics or pendulum suspension dynamics for the mirrors. The free-mass dynamics option is useful for illustrating ideal behavior of the interferometer, for instance in performing a constant-velocity fringe sweep. For other simulation work, the pendulum suspension dynamics better represents the LIGO system.

Users wishing to try different suspension parameters may do so by modifying the code slightly.

- Initial position and velocity. For each of the several mirrors (and for the source), initial displacements and velocities may be specified. They may be specified so as to excite single modes of the cavity, individually for each mirror, or collectively with any set of values desired. "Mirror modes" refers to coupled motion of particular mirrors that drive particular signals more or less directly.

Specifying controllers

The control servos are the "software" component of the LIGO interferometers. SMAC provides a very general means of linking particular signals to predefined or user-defined servos, and linking the servos to particular mirror and source actuation modes. The particulars of the LIGO servos will be discussed later.

The overall SMAC controller combines multiple servo loops. The user specifies servo type and servo parameters for each of these loops. SMAC provides a default linear servo form, which is used by most controllers. Other controllers may be non-linear in form, by virtue of internal switches or thresholds used to set internal gain factors, or so as to capture non-linearities in the mechanization of the servo. For these servos, SMAC requires the user to provide a FORTRAN file that performs the control computations.

SMAC linear servos are single-input, single-output in form. The inputs are particular detector demodulated voltages, and the outputs are mirror forces (or source phase), including multiple-mirror modal forces. The latter can be used to actuate the common-mode, difference-mode, or other mode of the interferometer, by moving arm cavity end mirrors together.

SMAC nonlinear servos can combine multiple inputs, including demodulated voltages and detector intensity signals. They can provide timed or sequential actions. They can actuate single or multiple mirrors and the source in any combination.

Servo outputs are in units of Volts. The gain conversion from servo output in volts to force applied to the test mass is 1 Newton/Volt. *It is important to note that if the servo designs are based on transfer functions generated in SMAC, the controller* *must contain a Force/Displacement (N/m) conversion since the drive for the transfer functions is displacement, not force.* The gain conversion from servo output in volts to phase-front displacement is 1 meter/Volt ([phase front displacement]=Φ*[carrier wavelength]/[2*π] where Φ is the phase of the light in radians).

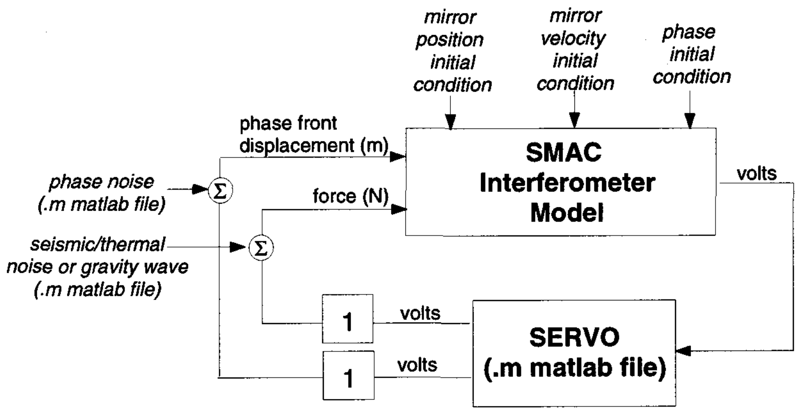

mirror position initial condition
mirror velocity initial condition
phase initial condition
phase front displacement (m)
SMAC Interferometer Model
volts
phase noise (.m matlab file)
Σ
force (N)
seismic/thermal noise or gravity wave (.m matlab file)
Σ
1
volts
SERVO (.m matlab file)
1
volts

FIGURE 3. Block Diagram of Servo System including external inputs.

The parameters the user needs to specify to define an interferometer control loop are:

- Linear or user-supplied nonlinear controller type.

For linear type, the following inputs need to be specified:

- Mirror mode to be driven. The choices here are any of the individual mirrors, the source phase, or 4 composite modes. These composite modes are: the M2+M4 common mode, in which both arm cavities elongate; the M2-M4 differential mode, in which the end mirrors move in opposite directions, so that one cavity gets longer while the other gets shorter; the Michelson common mode, M1+M2+M3+M4; and the Michelson differential mode, -[M1+M2]+[M3+M4].
- Signal to be used. In-phase or quad-phase signals at any of the detectors, synchronous with either modulation frequency, can be specified.

- Controller gain file. SMAC uses controller system matrices of a particular form described on page 66 in the section on Controller Definition. The user must identify the particular .m file containing each controller system matrix. Standard servo gain files are provided as part of the default data sets.

These parameters are set for each control loop, implicitly defining the number of active control loops. Multiple loops are required to fully control a recycled or recombined interferometer. Example servos are provided in the default data set.

Computational modes

SMAC is based on a set of "cavity field equations," time-difference equations that define the propagation of light from mirror to mirror, and from cavity to cavity, within the interferometer. SMAC computes the interferometer response by integrating these equations forward in time, together with the mirror dynamical equations and the servo filter equations. The user can choose different computational modes for this integration, and these are described here.

The cavity equations are written in terms of "circulating field" states, which are the fields just inside each separate cavity within the interferometer. This is illustrated in Figure 4 for a coupled cavity interferometer. In this interferometer, light is injected from the source through the recycling mirror (RM) into the front (or recycling) cavity, which is formed from the recycling mirror and mirror M1. Light that is in the front cavity will tend to remain there, bouncing back and forth between the cavity mirrors. This circulating light is described by E_R, the circulating field in the recycling cavity. The E-field states are complex scalars, representing the integrated transverse-electric field, which is assumed to consist solely of the TEM_{00} mode. If the cavity is near a resonance condition, then the E_R will grow to be quite large.

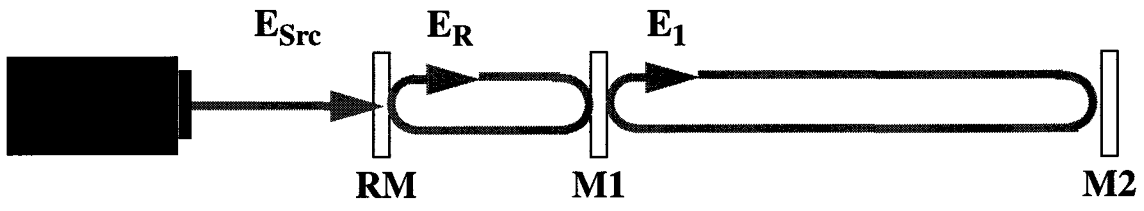


FIGURE 4. Coupled cavity IFO illustrating circulating field states

Some of the light circulating in the recycling cavity leaks out, either back towards the source, or into the arm cavity defined by mirrors M1 and M2. The light circulating in the arm cavity is defined by the circulating field state E_1. Again, if M1 and M2 are near a resonance condition (in this case, if the distance between M1 and M2 is a multiple of half the wavelength of the laser light), E_1 can grow to be quite large. LIGO interferometers are typically setup with the back mirrors (M2 in this case) having extremely high reflectivities, so that most of the light leaking out of the arm cavity returns to the recycling cavity -- most of which is reinjected back into the arm cavity. When everything is in resonance, the recycling cavity functions as a very high reflectivity compound mirror, allowing for quite large fields to be built up in the arm cavity.

The resonance conditions for interferometers are set by the distances between mirrors and the wavelength of the light. Moving the cavity mirrors imparts phase shifts to the circulating fields that strongly affect their amplitude. Very small motions -- within the width of the resonance fringe -- are possible without reducing the circulating fields very much. These distances define the operating range of an interferometer. Larger motions take the interferometer out of resonance, dropping cavity field amplitudes towards zero, and taking the interferometer out of its linear operating region as well. The SMAC cavity field equations incorporate these mirror (and source) phase shifts as driving terms, and remain valid whether the interferometer is in resonance or not.

Propagation of light between mirrors introduces time lags. For instance, if M2 is moved at a time t_0, its effect will not be noticed at M1 until $t=t_0+\tau_{12}$, where τ_{12} is the one-way light time between M2 and M1. It will not affect E1 until even later, at $t=t_0+\tau_{12}+\tau_{R1}$, where τ_{R1} is the one-way light time between M1 and RM.

For another example, see Figure 5. Here it is clearly seen that the field circulating in the recycling cavity at a time t ($E_R(t)$) is a function of the field E_1 at time $t-\tau_{R1}-2\tau_{12}$, of the field E_R at time $t-2\tau_{R1}$, of the source field at the current time t, and of mirror positions δ_{RM} at time $t-\tau_{R1}-\tau_{12}$, δ_{M2} at $t = t-\tau_{R1}$, and δ_{M1} at time t. Integrating the field equations forward in time, it is easily shown that the effect of a particular field or mirror state persists through many cycles of the light in the interferometer, becoming negligible only when the cumulative product of the reflectivities of each bounce brings the power of the affected field below detection noise levels. In LIGO interferometers, this process can take a significant fraction of a second.

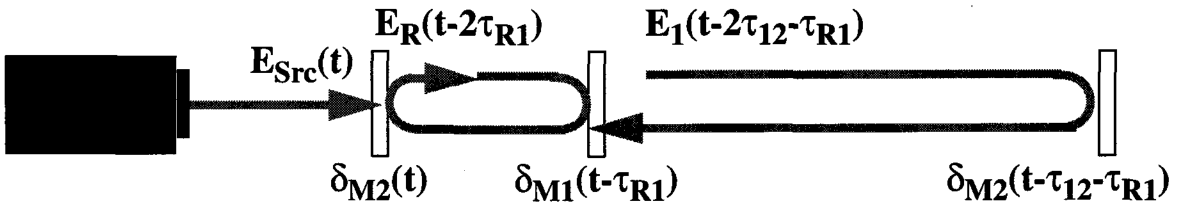

$E_R(t-2\tau_{R1})$
$E_1(t-2\tau_{12}-\tau_{R1})$
$E_{Src}(t)$
$\delta_{M2}(t)$
$\delta_{M1}(t-\tau_{R1})$
$\delta_{M2}(t-\tau_{12}-\tau_{R1})$

FIGURE 5. Coupled cavity IFO illustrating time delays

So free-space propagations introduce time delays into the cavity field equations. These time delays result in optical dynamical effects that must be accounted for in designing high-bandwidth interferometer length controls (though in some other cases they can be ignored). In general, integrating the field equations forward in time with time-varying forcing functions, total cavity response depends on the frequency content of the forcing function. If the frequencies of the forcing function exceed the frequency of the cavity pole of one or more of the cavities, then the dynamics of those cavities will influence the response of the interferometer as a whole.

SMAC offers 3 modes of computation that deal with time-delay effects of the cavity dynamics in different ways.

- The "dynamic" mode resolves times with very high resolution, integrating the cavity equations with a time step equal to the one-way light travel time in the average Michelson arm length. This mode has the highest frequency resolution, and takes the longest time to compute results.
- The "static" mode implements an adiabatic approximation: it assumes that the forcing function frequency is well below any of the cavity poles, consequently the cavities of the interferometer are always in equilibrium. This leads to a closed-form solution of the cavity equations, and frees SMAC from having to use any particular integration times -- integration time will then be chosen based on controller or mechanical dynamical considerations.
- The "mixed" mode can be used in the case of recycled or couple-cavity interferometers, provided the recycling cavity is much shorter than the arm cavity or cavities. In this mode, the recycling cavity is solved using a static approximation, while the dynamics of the arm cavities are captured by running them in

dynamic mode. The integration time used for this mode is an integer subsample of the one-way light travel time in the long arm.

For the full LIGO parameters, where the ratio of the lengths of the front and back cavities exceeds 400, and the controller bandwidths are below the pole of the recycling cavity, the mixed mode is quite accurate and should be preferred. Integration times are forced to be less than or equal to the one-way light time in the back cavities. We have found that the controllers actually need to be integrated faster than this, so we typically use integration times three times smaller than the maximum. This is specified in SMAC using the cavity subsample constant referred to earlier. In the mixed mode, the integration time is set to the one-way light time in the back cavities divided by the (integer) cavity subsample constant.

For the 40-meter in recycling mode, the front and back cavity lengths differ by a factor of approximately 17, and the source phase servos have quite high bandwidths, so the full dynamic mode may be desirable. In the dynamic mode, the simulation integration time step is the one-way light time in the average Michelson arm length. The length of the back cavity is adjusted to ensure that it's one-way light time is an integral multiple of this integration time step.

In comparing dynamic and mixed-mode transfer function calculations of the 40-meter interferometer, we saw differences only in the high frequency open-loop response. The fringe patterns due to a sweep are functionally identical. However, we have seen quantitative differences when simulating lock acquisition. Small differences in mirror trajectories were observed when high bandwidth servos were used to actuate the mirrors during lock acquisition. These differences did not appear to affect the overall performance (lock/doesn't lock). The user may wish to check particular cases using both the mixed and dynamic codes to determine the limits of validity of the mixed mode model for a particular interferometer configuration.

The static mode is useful for understanding the idealized behavior of an interferometer, which is important (for instance) in determining loop-closing sequences for lock-acquisition controllers. It is also useful for quickly verifying the locations and magnitudes of carrier and sideband resonances across multiple-wavelength sweeps, for verifying DC gains, fringe widths and other optical properties of a given interferometer.

External Inputs

For time simulations, the user can input initial conditions on the mirror displacements, mirror velocities, and laser phase front displacement. Simulations can be run with external force inputs driving the test masses. The force inputs are externally generated .mat matlab files and consist of a time series vector of force data for each "driven" degree of freedom. Simulations can also be run with an external phase noise input. An open loop block diagram with external inputs is shown in Figure 6 while the closed loop block diagram is shown in Figure 3.

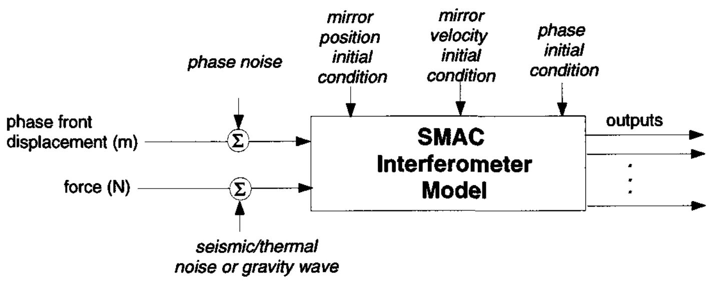

mirror position initial condition
mirror velocity initial condition
phase initial condition
phase noise
phase front displacement (m)
force (N)
Σ
Σ
SMAC Interferometer Model
outputs
seismic/thermal noise or gravity wave

FIGURE 6. **Block Diagram of open loop system including external inputs.**

Transfer function calculations are run with the input drive being test mass displacement and phase front displacement in meters. If controllers are designed using the transfer function calculations generated in SMAC it is important to remember that the transfer function calculations do not include the suspension dynamics (see Figure 7).

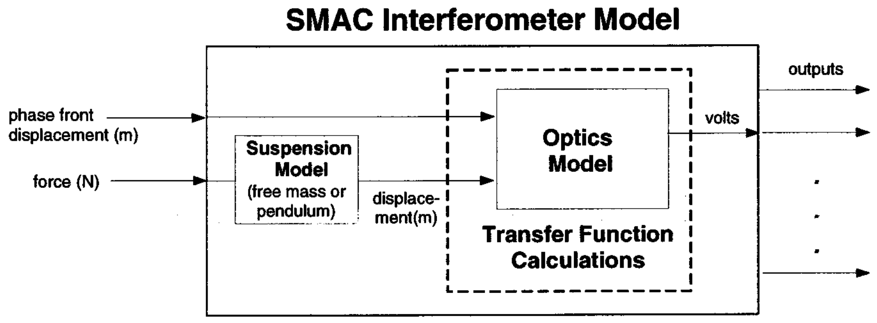

SMAC Interferometer Model
phase front displacement (m)
force (N)
Suspension Model (free mass or pendulum)
displacement(m)
Optics Model
Transfer Function Calculations
volts
outputs

FIGURE 7. Block Diagram showing input and output ports used for transfer function calculation.

Outputs

SMAC provides direct output of the intensities corresponding to the circulating field states in the cavities. The intensities of the carrier and the sidebands are output separately, as described later in this manual.

Besides the circulating field states, SMAC provides direct output of the detector fields and demodulated signal voltages, using 4 different pickoff mirrors, plus the beamsplitter and arm cavity end masses, feeding 10 different detectors. These are illustrated in Figure 8.

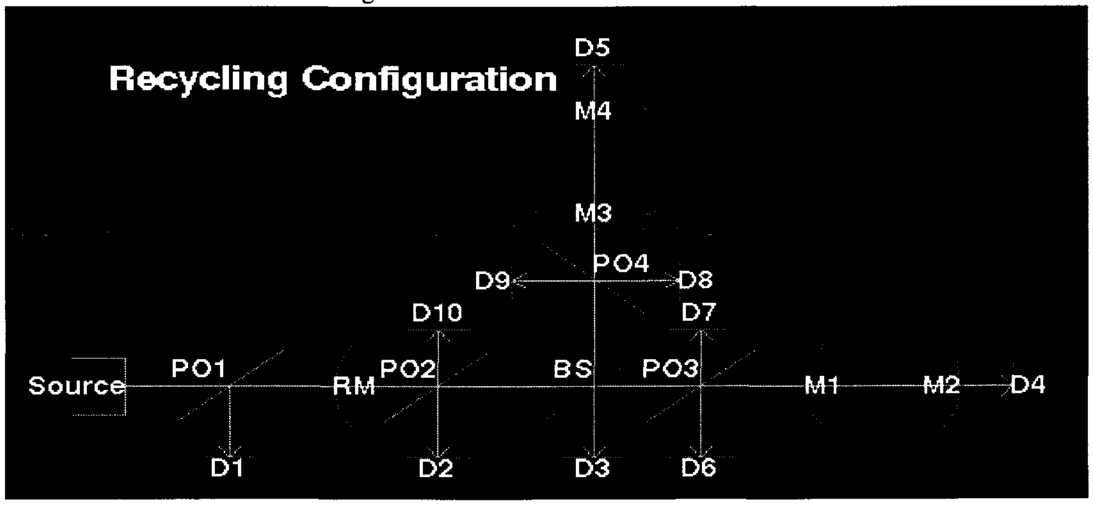


FIGURE 8. Pickoffs and detector fields

Detector D1 is located at the input port of the interferometer. In nominal operation this is a dark port, consequently the field at D1, E_{D1}, is quite small. E_{D1} combines the light leaked back out of the recycling cavity with the prompt reflection of the source light (in the recycled configuration), or the mixed light from the 2 arm cavities (in the recombined configuration). The pickoff for D1 will normally be an isolator for maximum efficiency. This is specified by setting the mirror transmissivity to 1 exactly.

Detectors D2 and D10 sample the recycling cavity fields. D2 sees the bright port of the Michelson interferometer formed by the arm cavities, and D10 directly samples the circulating field. The pickoff for D2 is within the recycling cavity, in the recycling configuration. If used, it will normally be specified as a weak beam splitter. The pickoff for D10 might be the same weak beamsplitter, or it may be implemented using back-scattered light from the recycling mirror.

Detectors D4 and D5 sample the light transmitted by the arm cavity end mirrors, M2 and M4. They provide signals proportional to the arm cavity circulating fields (but slightly delayed).

Detectors D6 and D7 sample the light in the in-line arm of the Michelson recycling cavity. The pickoff PO3 captures light scattered back from mirror M1. If M1 and M2 are wedged, one can model the first reflection (or prompt reflection) of the light from M1 as the light incident on D7 while the second reflection containing phase information about the long arm can be modeled as the light incident on D6. Similarly, detectors D8 and D9 sample the perpendicular leg of the Michelson, with the pickoff for D8 and D9 (PO4) capturing backscatter from M3.

SMAC provides carrier and sideband intensities at each detector as plotting options. These can be useful in verifying power levels and signal sensitivity levels as a function of time.

More importantly, SMAC uses the detector fields to compute the in-phase and quadrature-phase signals that are the primary outputs of a real interferometer. When the interferometer is in resonance, these signals provide a very sensitive measurement of cavity displacements including the gravity wave strain. They are also the input signals to the mirror and laser frequency/phase control servos. SMAC assumes a value for photodetector quantum efficiency of 0.8 electrons/photon, and a gain value of 1 Volt/Amp (the model of the photodetector is the same as used in Twiddle [ref LIGO-T960079-00-R, Twiddle, by Martin Regehr & James Mason].

A summary of the available time-domain outputs:

- Circulating field intensities, carrier and summed sidebands.
- Detector intensities, carrier and summed sidebands.
- Demodulated in-phase and quadrature-phase voltages.
- Mirror displacements and velocities, and equivalent source phase displacement

In addition to the time-domain outputs, SMAC can be used to generate frequency-domain transfer functions, which give the response of the demodulated voltages at each detector as a function of excitation frequency, where the excitation is provided by oscillating mirror position or source phase. These transfer functions are essential for control design and noise analysis applications. They are also useful in validating the accuracy of SMAC models.

Transfer functions are computed using the time-domain code in a sine-sweep mode, driving a particular degree of freedom with a sine wave forcing function, recording its gain at each detector, and then repeating at a new frequency. SMAC then plots the entire gain vs. frequency transfer function for each of the detectors. The process can then be repeated for any other desired mirror or source degree of freedom.

The user can use SMAC to generate an input file with which to drive the "Twiddle" program to generate the same transfer functions. Twiddle uses a direct frequency-response method to compute amplitude and phase transfer functions, and so provides an independent check of SMAC results. This capability is important if the user is concerned (for instance) about whether the mixed-mode or dynamic-mode should be used to model a particular interferometer configuration. Comparing Twiddle and SMAC outputs in the frequency region of interest will indicate whether the mixed-mode approach introduces potential error.

In comparing SMAC and Twiddle results, keep in mind that the Twiddle transfer functions are normalized with respect to carrier wavelength. Twiddle amplitude transfer functions should be multiplied by (carrier wavelength)/2π before comparing them to SMAC results. (This is done for the user automatically if **TFcheck** is used. See Appendix 4). If comparing results near the free spectral range of the arm cavities, it is recommended that the user specify a large cavity subdivision factor when specifying the integration time for SMAC. This will allow SMAC to adequately resolve high-frequency features.

An example

The user can exercise SMAC in several different modes of analysis. These include:

- Static-mode fringe sweep
- Dynamic or mixed-mode fringe sweep and open-loop simulation
- Closed loop simulation
- Transfer function generation

Figures 9-13 show a set of open loop simulations computed for the LIGO gravity wave interferometer. Figure 9 shows the M2 mirror trajectory as it is driven through resonance at a constant velocity of $\lambda/2$/second (the other lengths are kept constant so that the carrier stays resonant in the recycling cavity and the "other" arm). Resulting time traces due to M2 mirror motion are plotted in Figures 10-12 for selected cavity fields, detector fields, and demodulated signals. Figure 13 shows a transfer function calculation for the LIGO gravity wave interferometer from M2-M4 (i.e. L-) actuation to the gravity wave signal (detector 3 quad-phase output). These simulations were run using the mixed mode model.

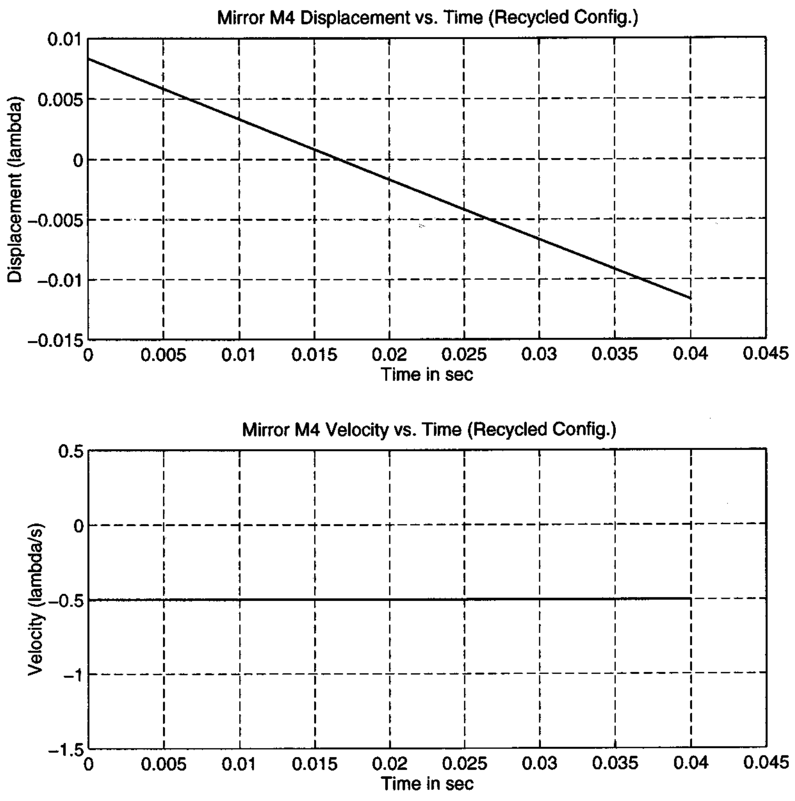


FIGURE 9. Open loop simulation of LIGO interferometer showing M2 trajectory as it sweeps through resonance (i.e. displacement=0).

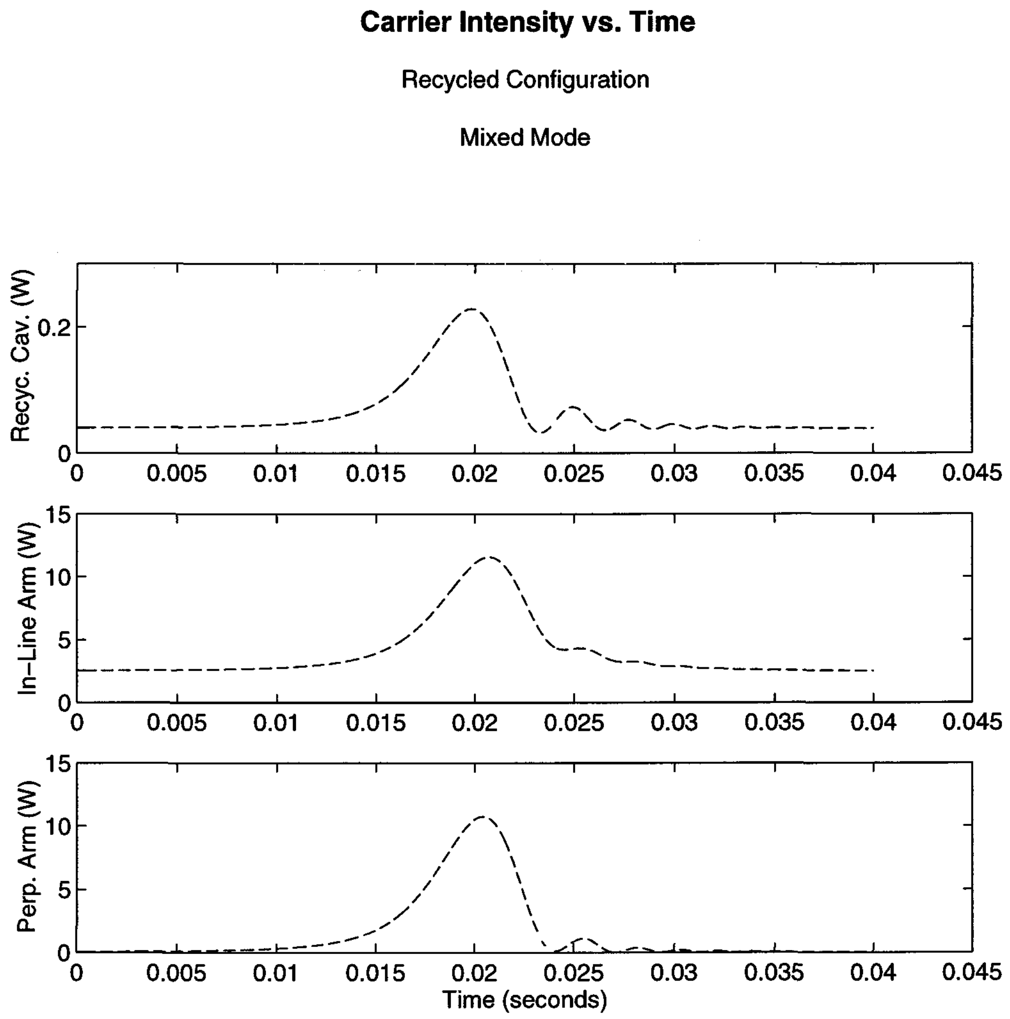


FIGURE 10. Open loop simulation of LIGO interferometer response showing carrier power in the three cavities as M2 sweeps through resonance. The three plots shows power in the recycling cavity, the in-line arm cavity and the perpendicular arm cavity, respectively.

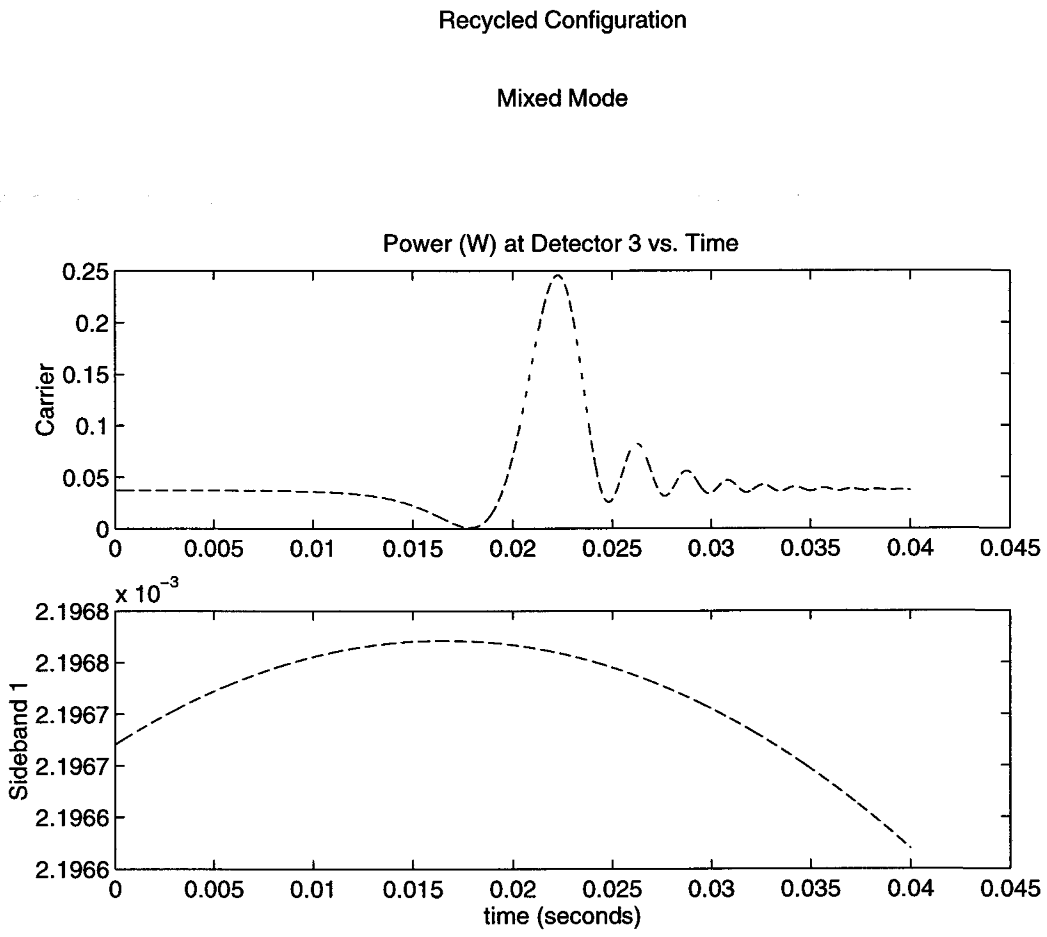


FIGURE 11. Open loop simulation of LIGO interferometer response showing carrier power and sideband power on photodetector 3 as M2 sweeps through resonance.

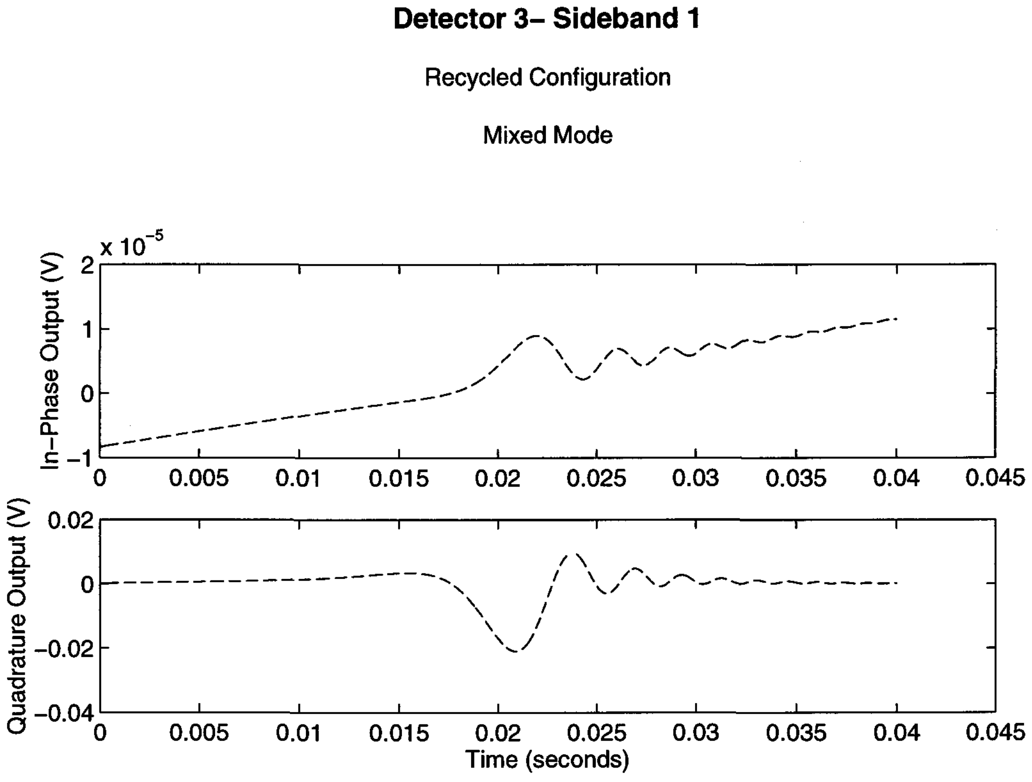


FIGURE 12. Open loop simulation of LIGO interferometer response showing in-phase and quad-phase demodulator output (demodulation frequency = sideband 1 frequency) as M2 sweeps through resonance.

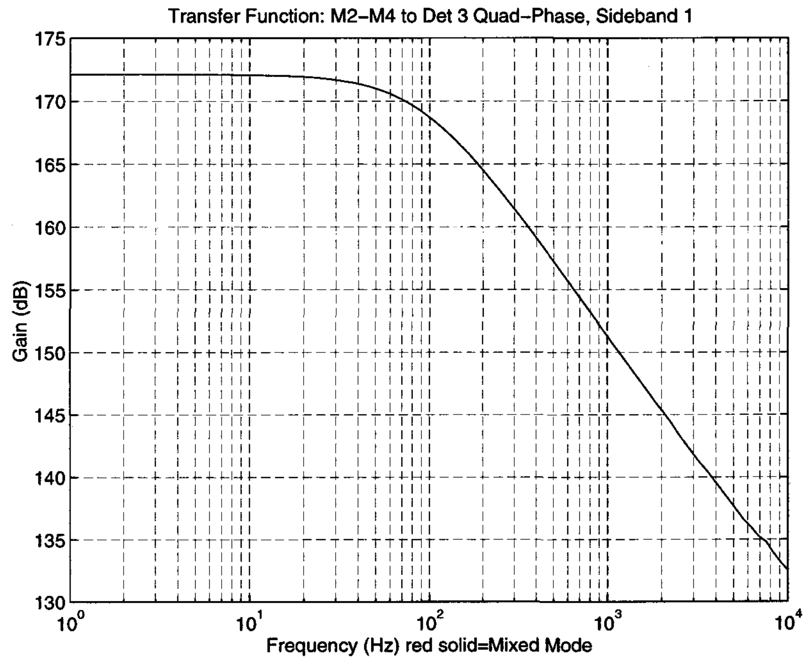


FIGURE 13. Frequency response of LIGO interferometer when carrier is resonant in all cavities and sidebands are resonant in recycling cavity. Transfer function calculated from M2-M4 actuation (L- degree of freedom) to the gravity wave signal (detector 3 quadrature phase output; demodulation frequency=sideband 1 frequency).

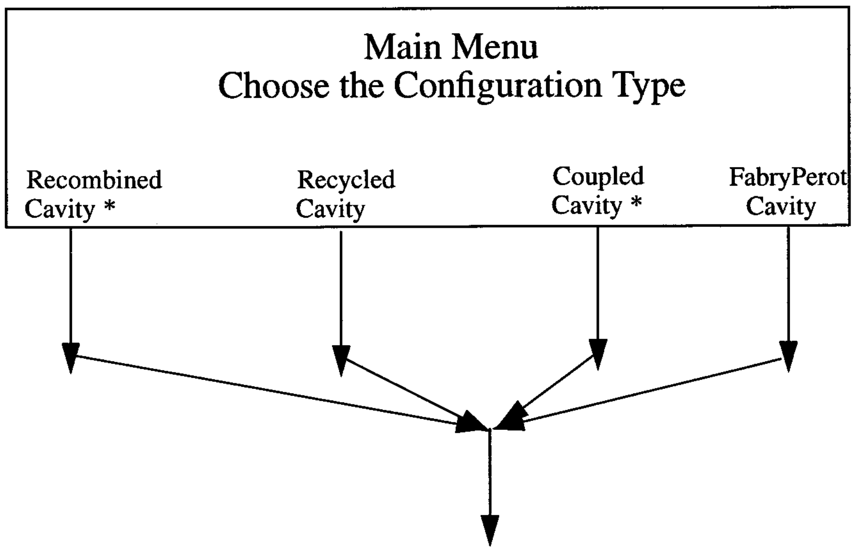
CHAPTER 3

Setting Up A Time Simulation Or Transfer Function Calculation

The user must define several different sets of parameters for a time simulation or transfer function calculation. These include simulation options, cavity parameters, initial conditions, plotting options, and other parameters depending on whether time simulations or transfer functions are being calculated.

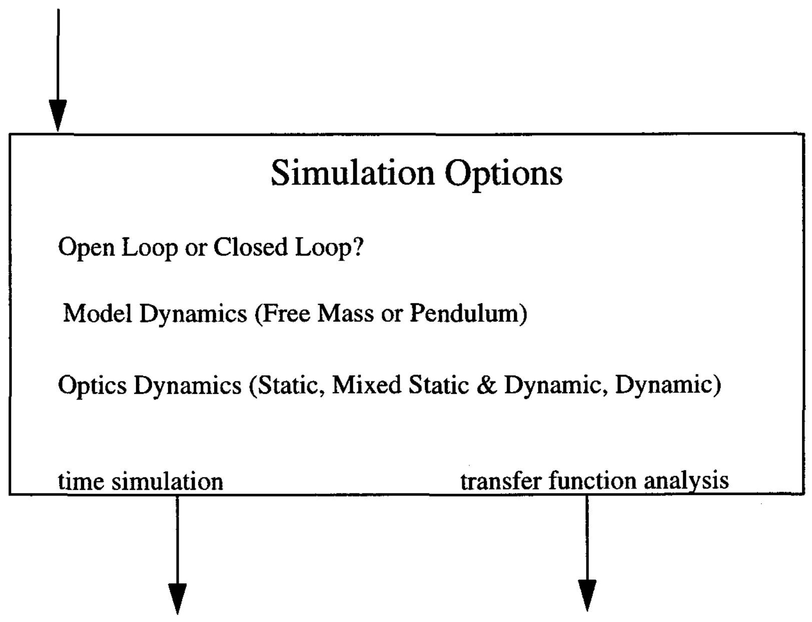
The GUI is designed to walk the user through these required sets of parameters. The GUI also provides the user with default values of the parameters.

The overview of the GUI menus that follow demonstrates the flow of menus. General information about each menu is provided. This section is intended as an overview only. More detailed descriptions of each window, and the parameters specified in each window is given in the following chapter.



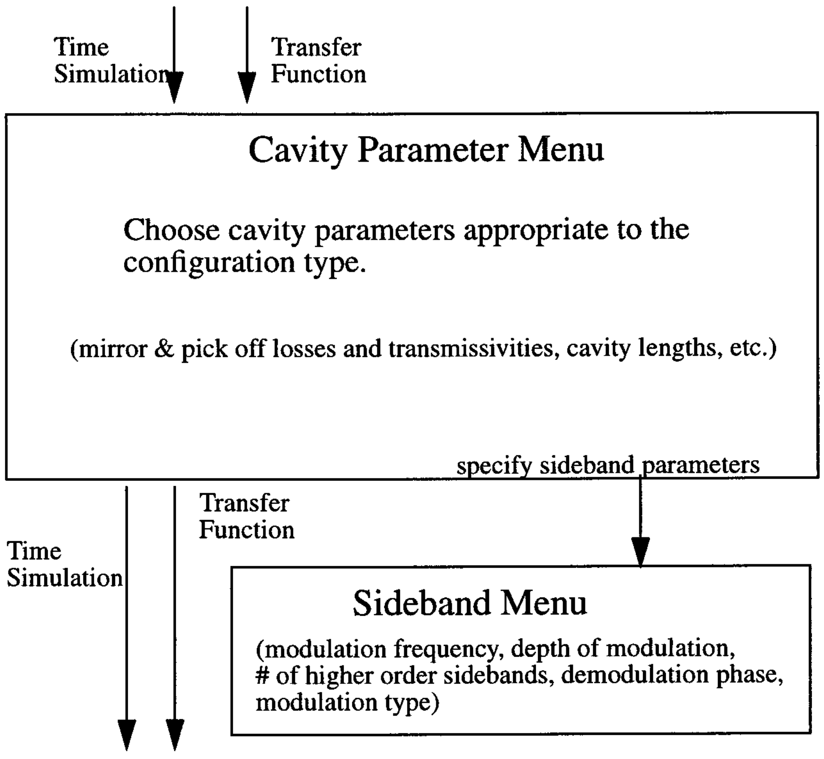
Currently, two configurations have a GUI interface, the recycled configuration and the fabry-perot configuration. GUIs for the recombined cavity and coupled cavity configurations will be implemented in the future. Currently, the recombined cavity and coupled cavity configurations can be run by making appropriate selections of cavity parameters using the recycled configuration GUI. For example, to run the recombined configuration, the user should set the Recycling Mirror Transmissivity equal to 1.

Once the configuration is chosen, simulation options must be selected.


Simulation Options
Open Loop or Closed Loop?
Model Dynamics (Free Mass or Pendulum)
Optics Dynamics (Static, Mixed Static & Dynamic, Dynamic)
time simulation
transfer function analysis

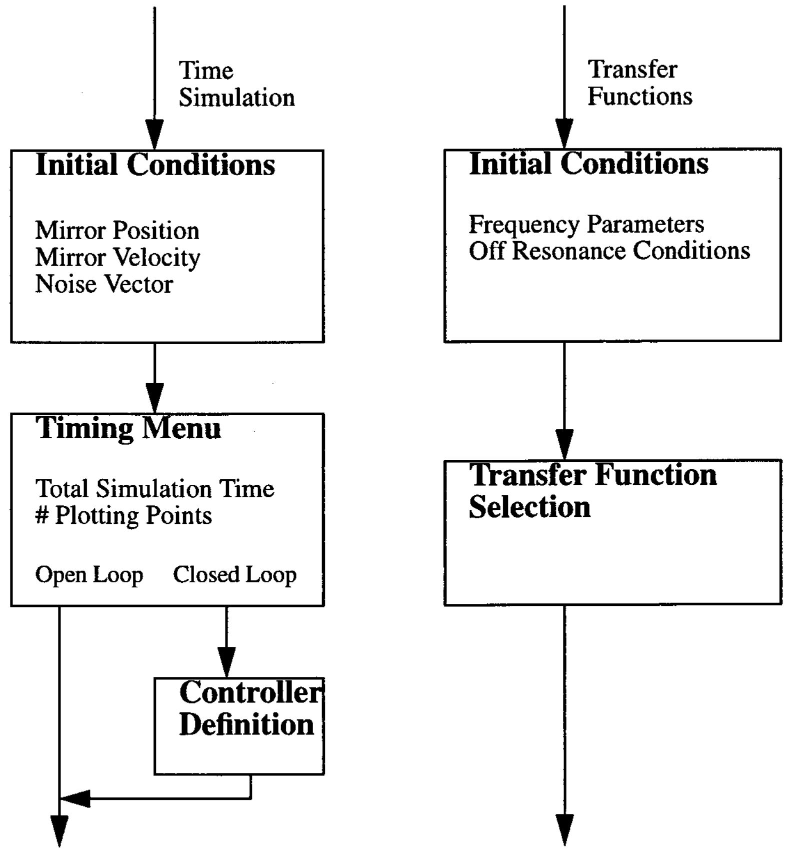
The two main simulation options are a time domain simulation and a transfer function analysis. Other options should also be selected, such as the type of model dynamics desired, the type of optical dynamics, and whether the simulation is open loop or closed loop. Not all combinations are available (eg. Closed Loop Transfer Functions), and not all combinations make sense (eg. Static Optical Dynamic Transfer Functions). The GUI will pop up a warning window if the user selects something that is not currently available or does not make sense.

Once the simulation options are chosen, cavity parameters must be set.


Time Simulation
Transfer Function
Cavity Parameter Menu
Choose cavity parameters appropriate to the configuration type.
(mirror & pick off losses and transmissivities, cavity lengths, etc.)
specify sideband parameters
Transfer Function
Time Simulation
Sideband Menu
(modulation frequency, depth of modulation, # of higher order sidebands, demodulation phase, modulation type)

If the user is interested in selecting sideband properties, there is an option to do so. Otherwise, default properties are used. The default properties included in each of the default cases are described in a later section.

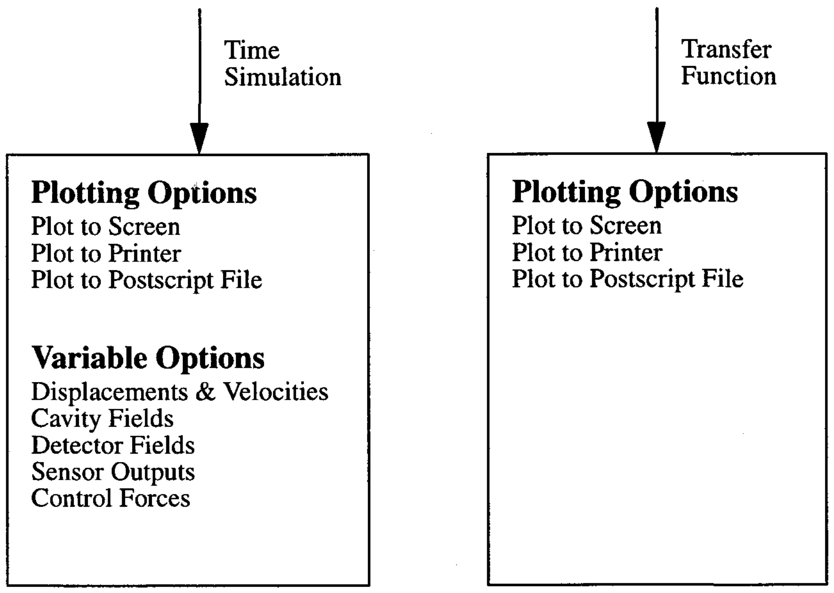
Next, initial conditions and other parameters of the simulation or transfer function calculations are needed. This is done on several menus. For time simulation, the following information is needed, initial position and velocity for the optical elements, disturbance time vectors, timing parameters (total simulation time, etc.), and controller information if applicable. For transfer function calculations, the following information is needed, frequency values, off-resonance conditions, and desired transfer functions.



There are several menu features which facilitate the user in setting up conditions. They will be discussed in the following chapter.

If the simulation is open loop, all controller values are automatically zeroed out.

Finally, plotting options are chosen.



The user can specify which variables should be printed, and whether the plots should be printed to the screen, printed on the printer, or printed to a file.

CHAPTER 4 Menus

This section gives a more detailed description of each of the GUI menus. It begins with helpful hints on using the menus. The section finishes with a description of the menus and the parameters in each menu.

Helpful Hints About Using the GUI

The GUI was developed to make the acquisition modeling program simple to use. Keeping these hints in mind will help beginner users avoid potential problems.

- The user should never destroy a window. They will go away when no longer needed.
- In many cases, when the user is finished with a menu, SMAC completes calculations needed for future windows. These calculations may take a few seconds, depending on the type of workstation that is being used and the load on that workstation. The use should not double click on buttons, as this may case SMAC to abort.
- The buttons are "color coded".
  1. Green buttons will sequence you to the next menu.
  2. Blue buttons are parameters, such as LenAsym.

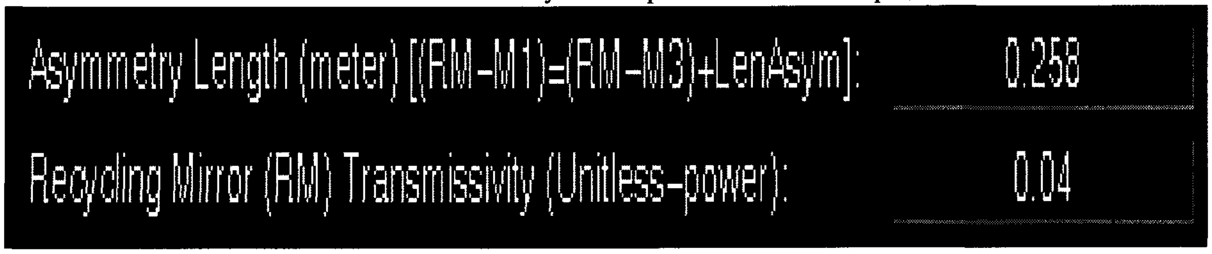
3. Red buttons are special function buttons, such as "Save", "Load", or "Show Configuration".
4. Magenta buttons send the user back to the Simulation Options Menu.

- Previously entered data is preserved if the user goes back to the cavity parameter menu. The data may appear differently than it was entered by the user. For example, if the user entered pi/2, the data may re-appear as 1.5708. Currently, the user must go through each menu again, even though data does not need to be re-entered. This will be modified in the future.
- Many windows have a "Save" button which can be used to store parameters for future runs. There is also a "Save" button at the end of SMAC which allows the user to store the entire run. The "Save" button at the end of SMAC should be used only to preserve data for replotting, etc. *The user should not load a past run, on any of the menus which allow the user to load a saved parameter data set.* Values stored in the past run may be incompatible with the values which have been previously set using the GUI, consequently SMAC results may be wrong.
- "c", the speed of light, is hardwired to 2.99792e8 meters/sec. Comparisons with other codes/results should be done only if the other codes/results use the identical value for the speed of light.

Types of Buttons in the GUI

There are several different types of "buttons" on the GUI.

The first is a button which is merely a descriptive label. For example,



Asymmetry Length (meter) [(RM-M1)=(RM-M3)+LenAsym]: 0.258

Recycling Mirror (RM) Transmissivity (Unitless-power): 0.04

FIGURE 14. Example of descriptive label button & editable button.

The Asymmetry Length (meter) [(RM-M1)=(RM-M3)-LenAsym]: button and the Recycling Mirror (RM) Transmissivity (Unitless-power): button on the cavity parameter menu are only descriptive labels. The user cannot modify these buttons.

The second type of button looks exactly like the descriptive button, but this button can be edited. The 0.258 and 0.04 in the above figure are example of editable buttons. To modify the value in the button, all the user has to do is click on the button, and type the desired value.

The third type of button is called a push button. The user just needs to click on the button to activate it. The pushbuttons are the green and red buttons as shown in the figure below.

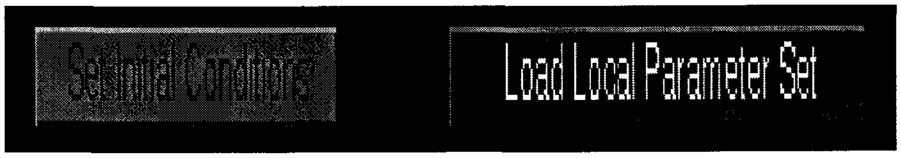


FIGURE 15. Examples of pushbuttons.

The fourth type of button is called a radio button. A radio button is used when only one of several options are allowed, such as an Open Loop or Closed Loop option. The option which has been selected has a white diamond. An example of a radio button is shown in the figure below.

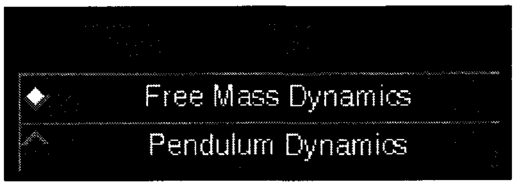


FIGURE 16. Example of a radio button

To change the selected option, the user just has to click on the desired box. For example, to change from Free Mass Dynamics to Pendulum Dynamics, just click on the **Pendulum Dynamics** box, and the display will change as shown in the figure below.

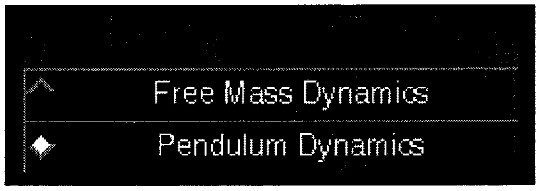


FIGURE 17. Example of a radio button

The fifth type of button is called a check box. Check boxes are very similar to radio buttons except that multiple check boxes can be activated at the same time. Activated check boxes have a white square on the left hand side of the button. A check box may or may not have descriptive words in the check box. The **Activate** button is an example of check boxes without words is shown in the figure below.

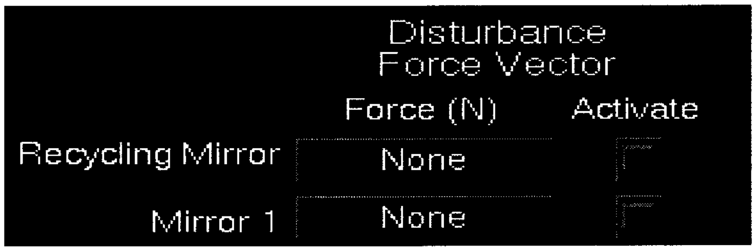


FIGURE 18. Example of Check Boxes Without Text

To signify that the Recycling Mirror Disturbance Vector should be activated, the user just needs to click on the check box, and the square in the check box will turn white. Similarly, to turn "off" a feature, the user should click on the check box a second time, and the square in the check box will return to its original color. An activated check box is shown in the Figure below.

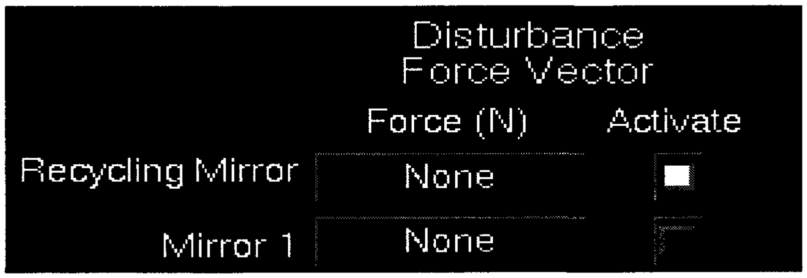


FIGURE 19. **Example of Check Boxes Without Text**

In most cases, when a check box has text, the text in check boxes may change once the check box has been activated. An example of this is shown in the figures below.

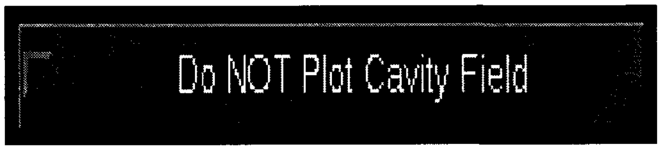


FIGURE 20. **Example of Check Box With Text**

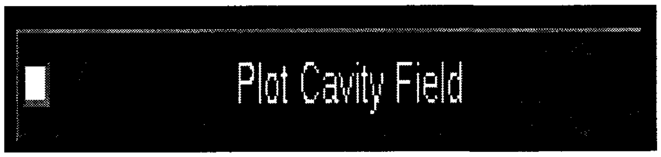


FIGURE 21. **Example of Check Box With Text**

The sixth and final type of box is a popup box. The popup box gives the user several options to choose from, and the currently chosen option is the only one displayed. The popup boxes have rectangles on the right side of the button, as shown in the figure below.

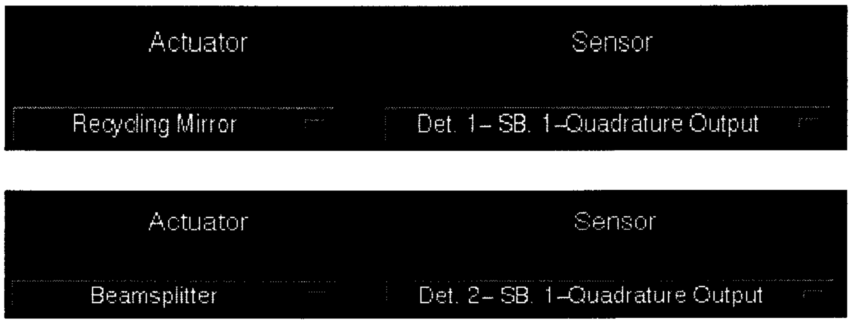

Actuator
Sensor
Recycling Mirror
Det. 1- SB. 1-Quadrature Output
Actuator
Sensor
Beamsplitter
Det. 2- SB. 1-Quadrature Output

FIGURE 22. Different Displays on Popup Boxes

A More Detailed Description of Each GUI Menu

Before beginning SMAC, it is recommended that the user types **clear** at the Matlab prompt to erase out old variables and data. To start the SMAC GUI, the user should type **smac** at the Matlab prompt.

Two different types of analysis can be completed using SMAC - time simulation analysis and transfer function analysis. As shown in the flow diagrams in Chapter 3, some of the GUI menus are the same for the different types of analysis, and some of the GUI menus are different. The menus needed for time simulation analysis are discussed first. Then, the menus needed for transfer function analysis which are not needed in time simulation analysis are discussed.

4.1 Time Simulation

SMAC Main Menu

The first SMAC menu asks the user to select which configuration should be used. Currently, only the Fabry Perot and Recycled Configurations are available for use. The Coupled Cavity and Recombined Configurations can be simulated, but the user must select the Recycled Configuration, and set the cavity variables to appropriate values. If a configuration other than the recycled configuration is chosen, some of the variables that appear in the figures that are presented below will not appear on the GUI.

If the user is uncertain what the configuration names are, a Show Configurations button exists, and will generate drawings of each of the configurations.

The SMAC Main Menu is shown below.

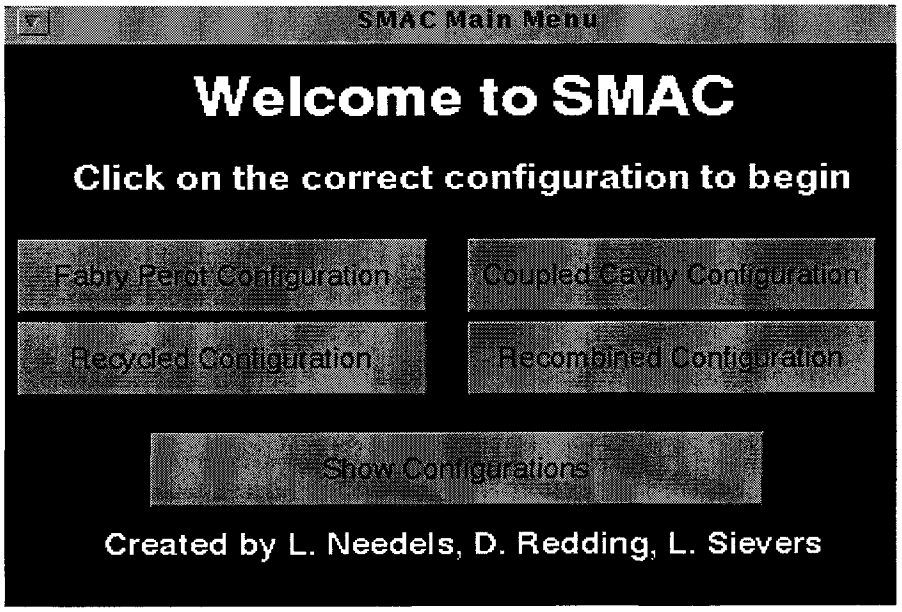

SMAC Main Menu
Welcome to SMAC
Click on the correct configuration to begin
Fabry Perot Configuration
Coupled Cavity Configuration
Recycled Configuration
Recombined Configuration
Show Configurations
Created by L. Needels, D. Redding, L. Sievers

FIGURE 23. SMAC Main Menu

The Recycled Configuration drawing is shown in the figure below.

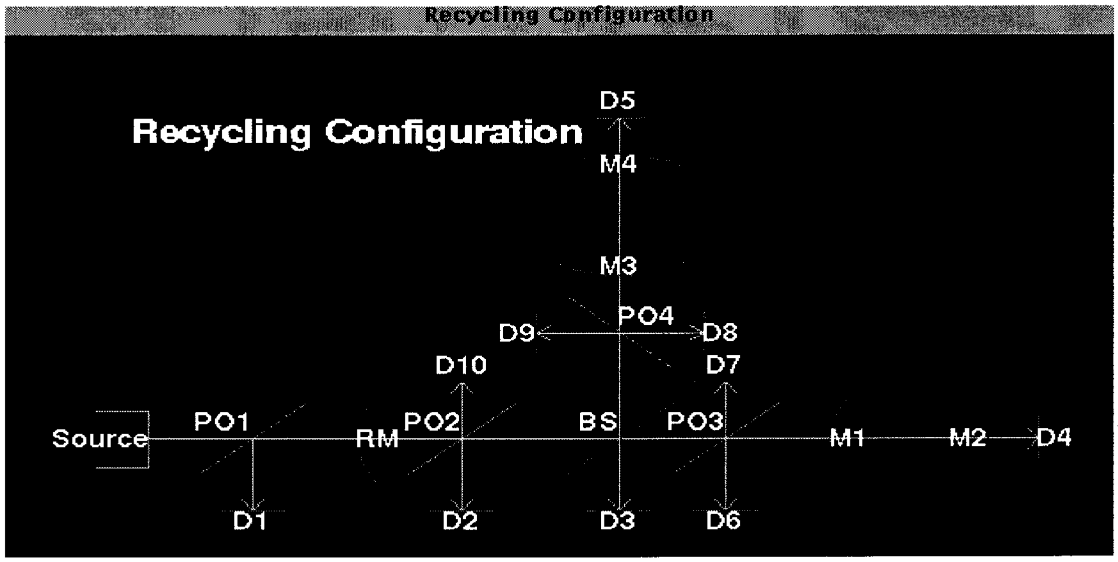


FIGURE 24. **Recycling Configuration**

Distances between optical elements for the recycling configuration are shown in the figure below. SMAC assumes that the RM, BS, PO1, PO2, PO3, PO4 are all at the same location and that the modulation and demodulation for the associated photodetectors is also done at this location. This simplifies the bookkeeping of phase lags in the system; a method adopted from the Twiddle code. These distances are very important to note if demodulator phase settings are to be offset from 0.

Arrows indicating positive mirror translation directions are also included in the figure.

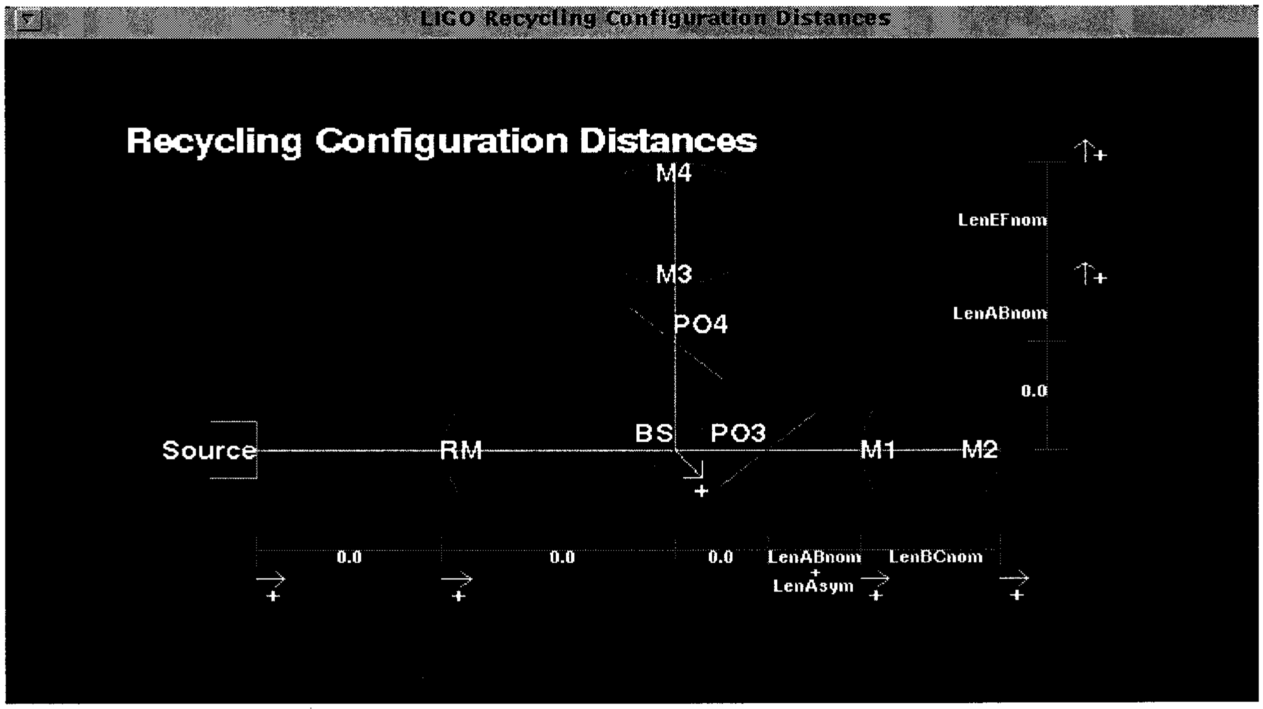
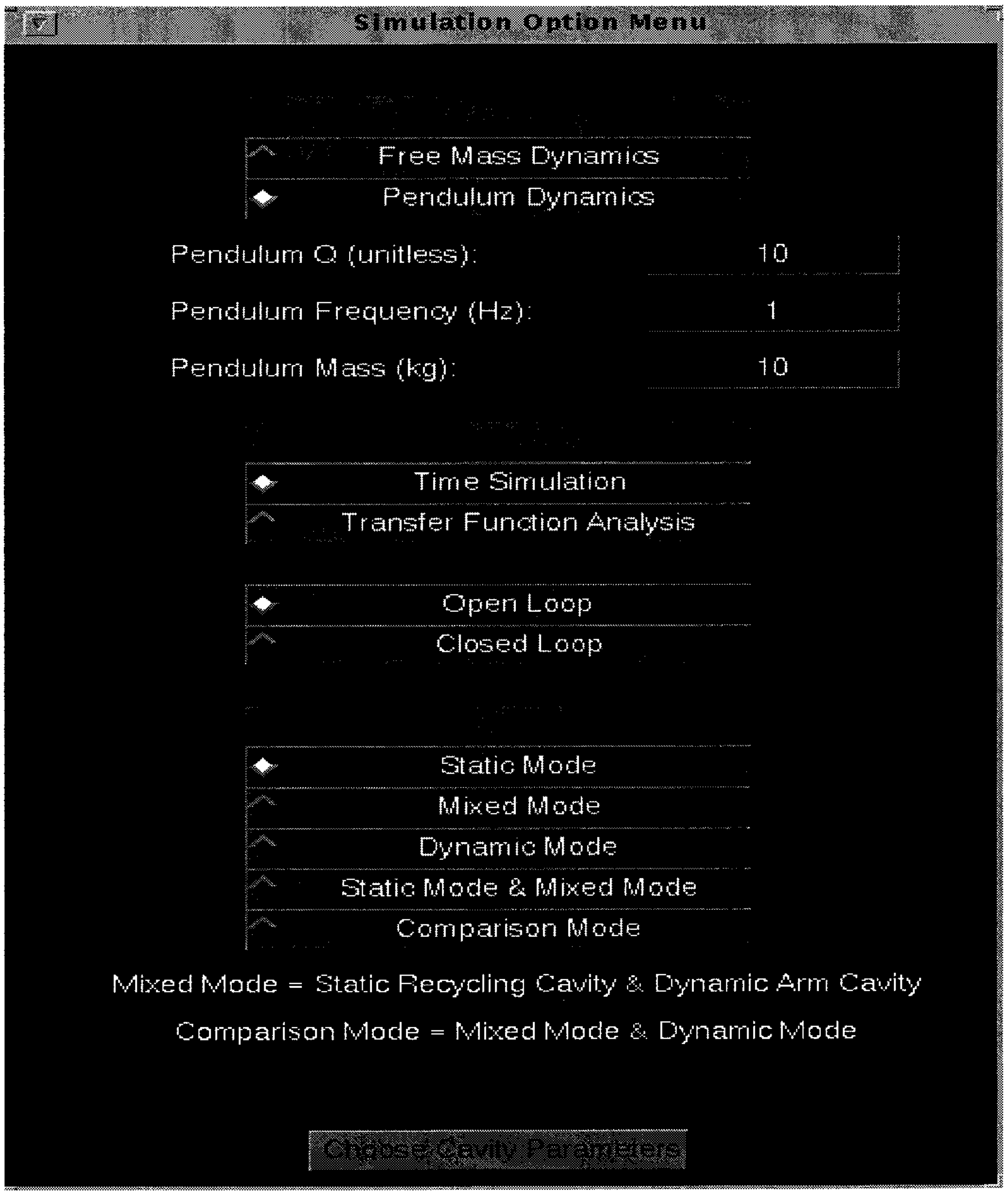


FIGURE 25. Recycling Configuration Distances

Simulation Option Menu

The Simulation Option Menu is used to set up the type of analysis that will be done. The type of Model Dynamics must be chosen (Free Mass Dynamics or Pendulum Dynamics). The Simulation Type must be chosen (Time Simulation or Transfer Function Analysis). An Open Loop or Closed Loop simulation must be chosen. Finally, the type of Optical Dynamics (Static, Mixed, Dynamic) must be specified. Static Mode means that the optical dynamics of the arms and the recycling cavity are modeled using a static model. Mixed Mode means the optical model for the arm cavities is dynamic but the optical model for the recycling cavity is static.The Dynamic Mode means that both the optical dynamics of the arm cavities and the recycling cavity are dynamic. The Dynamic Optics **Dynamic** version is a computationally intensive version of the code and it is recommended that it be used only when the added resolution is required.

The user must also specify the **Pendulum Mass. If Pendulum Dynamics** are chosen, the user must specify the **Pendulum Q, Pendulum Mass,** and **Pendulum Frequency.** Not all combinations of options are currently available and it doesn't make sense to run all combinations (eg. Transfer Function analysis with Static Optical Dynamics). The GUI issues a warning if an inappropriate combination is chosen.



Free Mass Dynamics
[✓ Pendulum Dynamics]

Pendulum Q (unitless): 10

Pendulum Frequency (Hz): 1

Pendulum Mass (kg): 10

[✓ Time Simulation]
Transfer Function Analysis

[✓ Open Loop]
Closed Loop

[✓ Static Mode]
Mixed Mode
Dynamic Mode
Static Mode & Mixed Mode
Comparison Mode

Mixed Mode = Static Recycling Cavity & Dynamic Arm Cavity

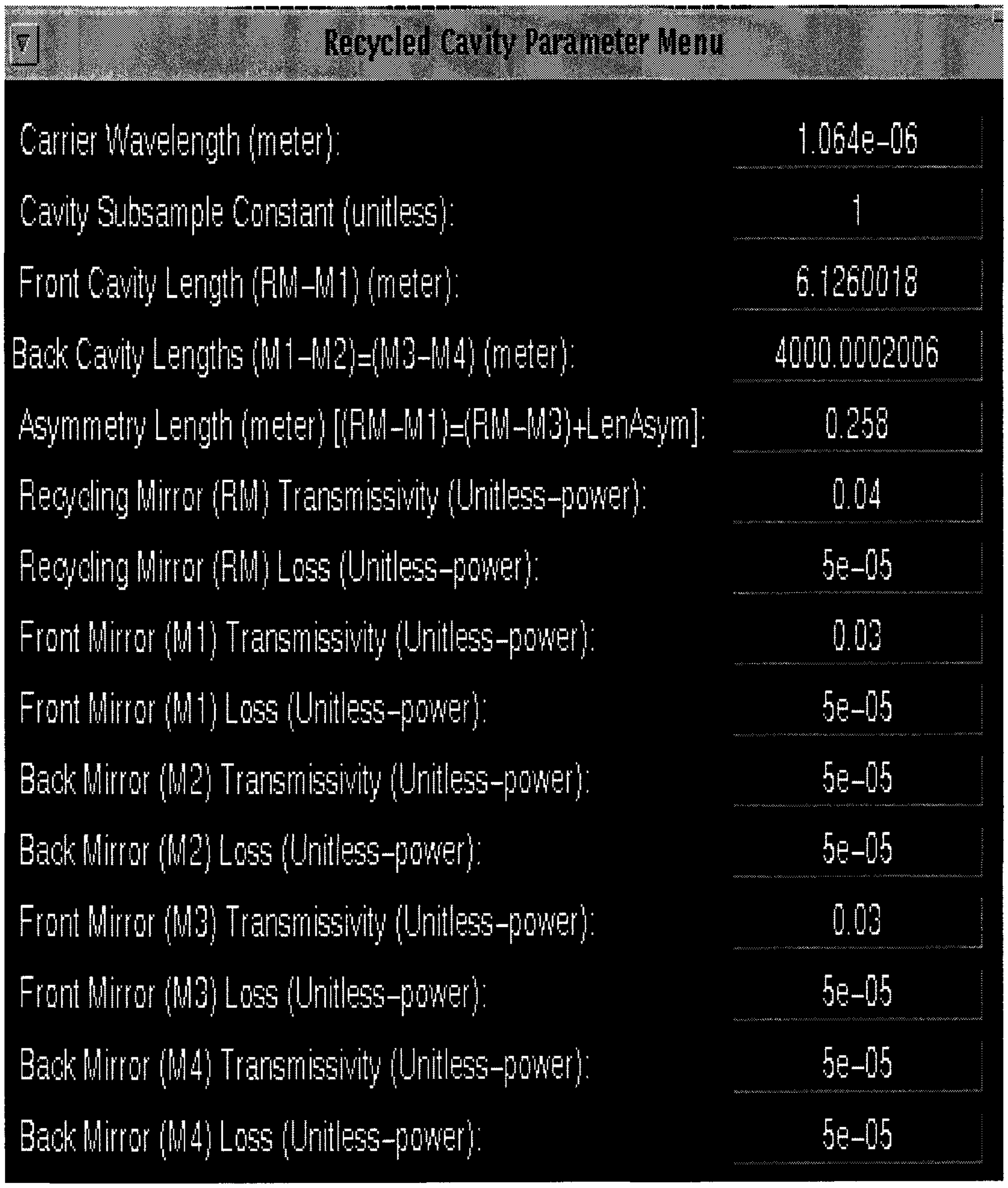
Comparison Mode = Mixed Mode & Dynamic Mode

FIGURE 26. Simulation Option Menu

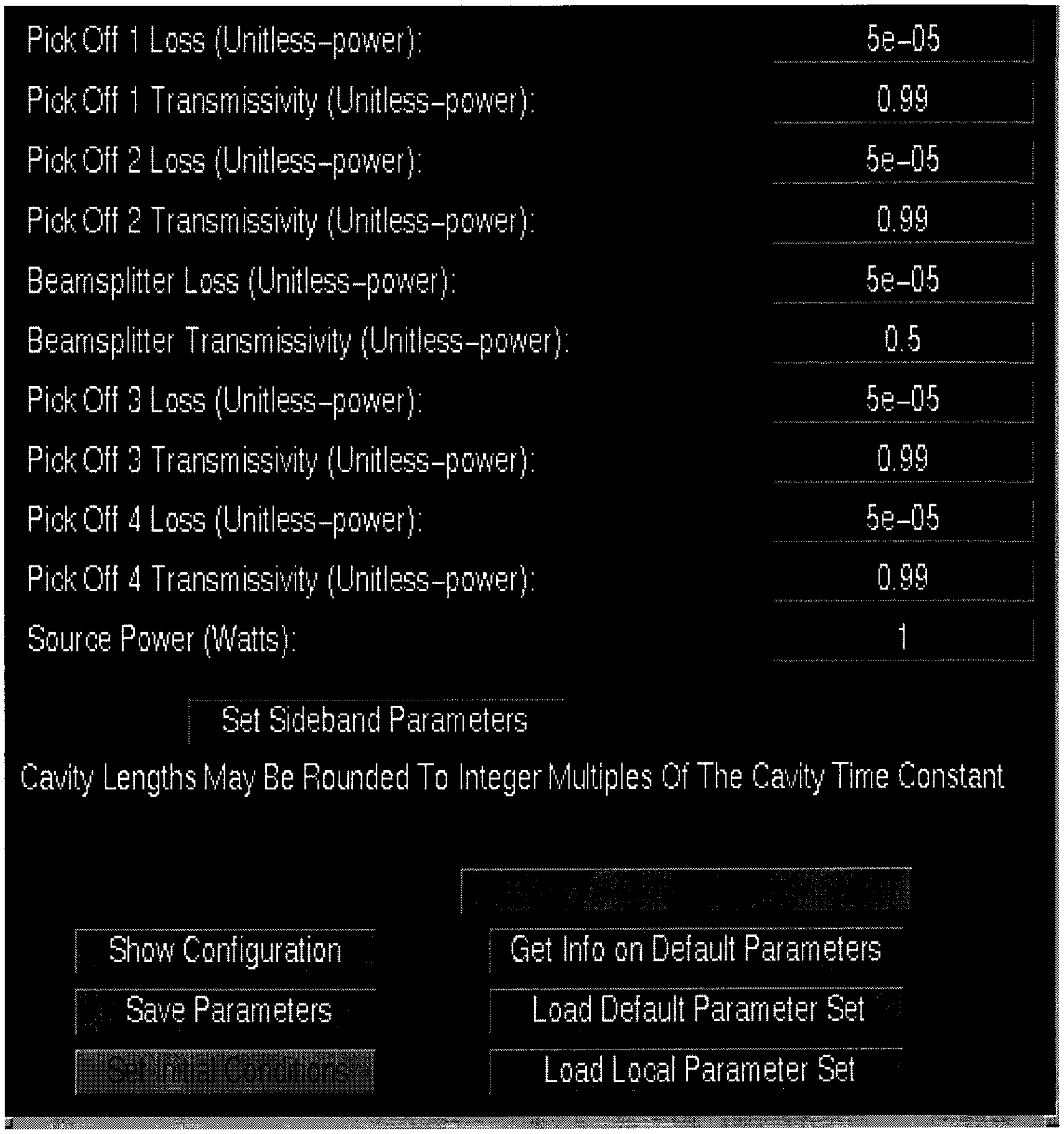
Cavity Parameter Menu

The Cavity Parameter Menu is used to set the properties which describe the cavity.

It is very important to note that “c”, the speed of light is a automatically set for the user. Models for high finesse cavities are invariably sensitive to the approximation used for the value of the speed of light, "c". The value of c used in smac is 2.99792*10^8 meters/second. When comparing other models with SMAC, erroneous conclusions can be drawn if the "other" model doesn't use the identical value of c for the field calculations.



Carrier Wavelength (meter):	1.064e-06
Cavity Subsample Constant (unitless):	1
Front Cavity Length (RM-M1) (meter):	6.1260018
Back Cavity Lengths (M1-M2)=(M3-M4) (meter):	4000.0002006
Asymmetry Length (meter) [(RM-M1)=(RM-M3)+LenAsym]:	0.258
Recycling Mirror (RM) Transmissivity (Unitless-power):	0.04
Recycling Mirror (RM) Loss (Unitless-power):	5e-05
Front Mirror (M1) Transmissivity (Unitless-power):	0.03
Front Mirror (M1) Loss (Unitless-power):	5e-05
Back Mirror (M2) Transmissivity (Unitless-power):	5e-05
Back Mirror (M2) Loss (Unitless-power):	5e-05
Front Mirror (M3) Transmissivity (Unitless-power):	0.03
Front Mirror (M3) Loss (Unitless-power):	5e-05
Back Mirror (M4) Transmissivity (Unitless-power):	5e-05
Back Mirror (M4) Loss (Unitless-power):	5e-05



Pick Off 1 Loss (Unitless-power):	5e-05
Pick Off 1 Transmissivity (Unitless-power):	0.99
Pick Off 2 Loss (Unitless-power):	5e-05
Pick Off 2 Transmissivity (Unitless-power):	0.99
Beamsplitter Loss (Unitless-power):	5e-05
Beamsplitter Transmissivity (Unitless-power):	0.5
Pick Off 3 Loss (Unitless-power):	5e-05
Pick Off 3 Transmissivity (Unitless-power):	0.99
Pick Off 4 Loss (Unitless-power):	5e-05
Pick Off 4 Transmissivity (Unitless-power):	0.99
Source Power (Watts):	1

Cavity Lengths May Be Rounded To Integer Multiples Of The Cavity Time Constant

FIGURE 27. **Cavity Parameter Menu**

TABLE 1. **Cavity Menu Parameters**

Menu entry name	**Internal variable**	**Description**
Carrier Wavelength	lamdaC	Wavelength of laser source (λ)
Cavity Subsample Constant	Lfac	Number of integer τ's per cavity τ. If set to 1, integration time is one-way light travel time in arm cavity (τ). If set to n, the integration time step = τ/n.In dynamic mode, the one-way light travel time corresponds to the average Michelson arm length, and this length cannot be subdivided.
Front Cavity length	LenIdiot	Length between RM and M1 (long arm of Michelson) in meters. Rounded to nearest $\lambda/2$.
Back Cavity Lengths	LenBCnom & LenEFnom	Length between M1 and M2 in meters & length between M3 and M4 in meters. In dynamic mode, the back cavity length is adjusted to an integral number of L_0 (the average recycling cavity length).
Asymmetry Length	LenAsym	Asymmetry length between the In-Line Arm and the Perpendicular Arm (RM-M1) = (RM-M3) + LenAsym
Recycling Mirror (RM) Transmissivity	TA	The fractional amount of power transmitted through the Recycling Mirror (RM)
Recycling Mirror (RM) Loss	AA	The fractional amount of power that is lost during reflection on the Recycling Mirror (RM)
Front Mirror (M1) Transmissivity	TB	The fractional amount of power transmitted through the Front Mirror (M1)
Front Mirror (M1) Loss	AB	The fractional amount of power lost during reflection on the Front Mirror (M1)
Back Mirror (M2) Transmissivity	TC	The fractional amount of power transmitted through the Back Mirror (M2)
Back Mirror (M2) Loss	AC	The fractional amount of power lost during reflection on the Back Mirror (M2)
Front Mirror (M3) Transmissivity	TE	The fractional amount of power transmitted through the Front Mirror (M3)
Front Mirror (M3) Loss	AE	The fractional amount of power lost during reflection on the Front Mirror (M3)

TABLE 1. Cavity Menu Parameters

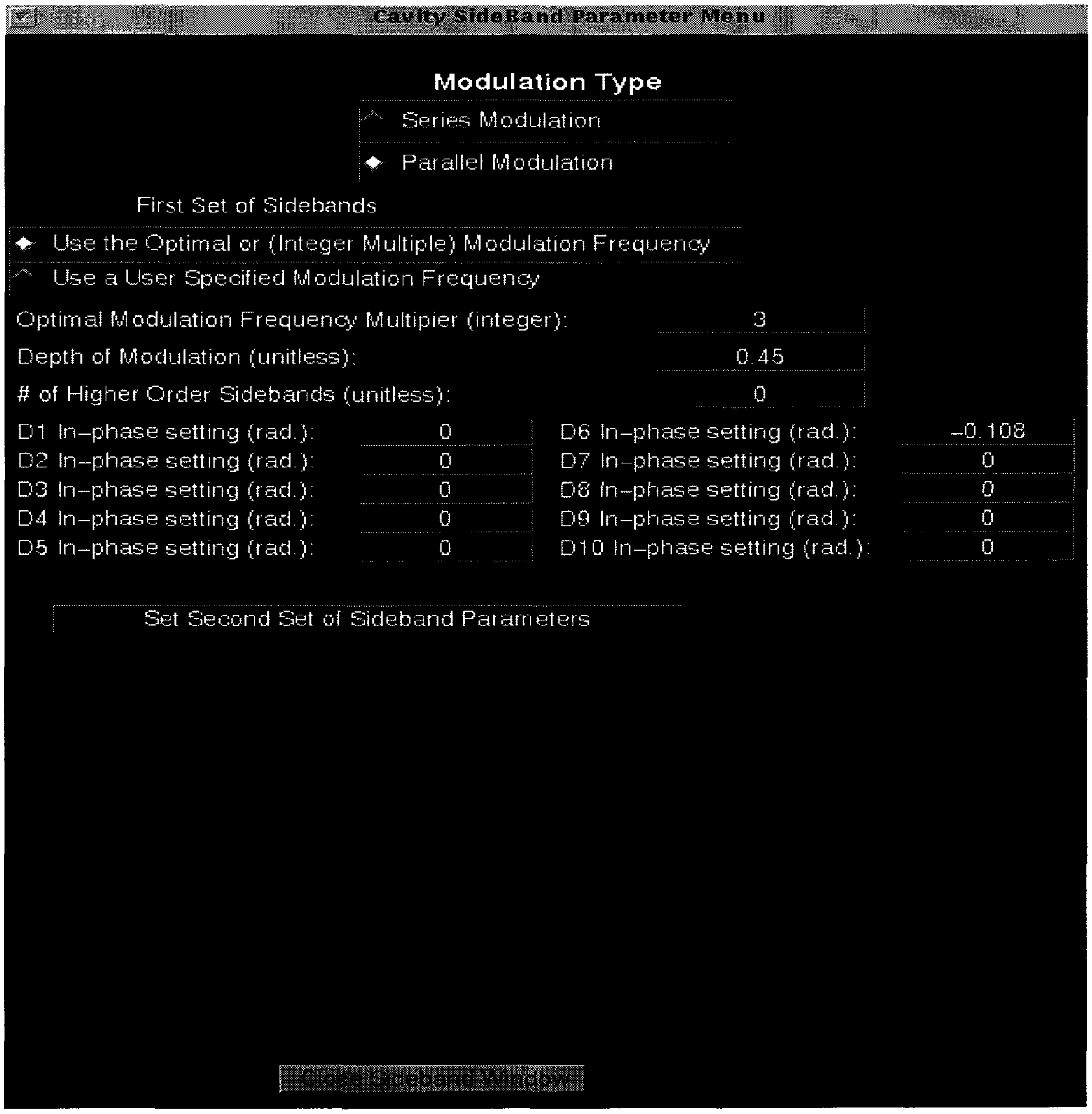
Menu entry name	Internal variable	Description
Back Mirror (M4) Transmissivity	TF	The fractional amount of power transmitted through the Back Mirror (M4)
Back Mirror (M4) Loss	AF	The fractional amount of power lost during reflection on the Back Mirror (M4)
Pick Off 1 Loss	ABS1	The fractional amount of power lost on Pick Off 1 (PO1 at 45^o angle)
Pick Off 1 Transmissivity	TBS1	The fractional amount of power transmitted by Pick Off 1 (PO1). Pick Off 1 becomes an isolator if TBS1=1.0
Pick Off 2 Loss	ABS3	The fractional amount of power lost on Pick Off 2 (PO2 at 45^o angle)
Pick Off 2 Transmissivity	TBS3	The fractional amount of power transmitted by Pick Off 2 (PO2). Pick Off 2 becomes an isolator if TBS3=1.0
Beamsplitter Loss	ABS2	The fractional amount of power lost by the Beamsplitter (BS at 45^o angle)
Beamsplitter Transmissivity	TBS2	The fractional amount of power transmitted by the Beamsplitter (BS)
Pick Off 3 Loss	APO3	The fractional amount of power lost on Pick Off 3 (PO3 at 45^o angle)
Pick Off 3 Transmissivity	TPO3	The fractional amount of power transmitted by Pick Off 3 (PO3) Pick Off 3 becomes an isolator if TBS3=1.0
Pick Off 4 Loss	APO4	The fractional amount of power lost on Pick Off 4 (PO4 at 45^o angle)
Pick Off 4 Transmissivity	TPO4	The fractional amount of power transmitted by Pick Off 4 (PO4) Pick Off 4 becomes an isolator if TBS3=1.0
Source Power	Is	The power of the source in Watts.

A special note should be made about the Pick Offs. When the Pick Off transmissivity is chosen to be 1 or greater, the pick off's function becomes that of an Optical

Isolator. The Pick Off Transmissivity and Reflectivity used by SMAC are calculated to be the following values: T=1-A/2 and R=1-A/2 where A is defined as the Pick Off Loss that appears as a parameter in the Cavity Parameter GUI Menu.

There are several special buttons on the **Cavity Parameter Menu**. The **Show Configuration** button will bring forward a plot showing the layout of the cavities, with names on the mirrors, pickoffs and detectors. The **Save Parameters** button allows the user to save the cavity parameters into a .mat file. Only the cavity parameters are saved. The **Get Info on Default Parameters** displays a description of the default parameter sets (eg. 4 km with green light - recycled configuration or 4 km with IR light - recycled configuration) that are available. The information is displayed in the Matlab window. The user may find it useful to begin with one of the default cases and modify those parameters as needed. More information on the default parameter sets is given in Chapter 5. The **Load Default Parameter Set** button loads one of the default parameter sets. The **Load Local Parameter Set** button loads a parameter set that the user has previously saved.

The **Set Sideband Parameters** button allows the user to modify sideband parameters. The sideband menu is shown below.



Cavity SideBand Parameter Menu

Modulation Type

^ Series Modulation

◆ Parallel Modulation

First Set of Sidebands

◆ Use the Optimal or (Integer Multiple) Modulation Frequency

^ Use a User Specified Modulation Frequency

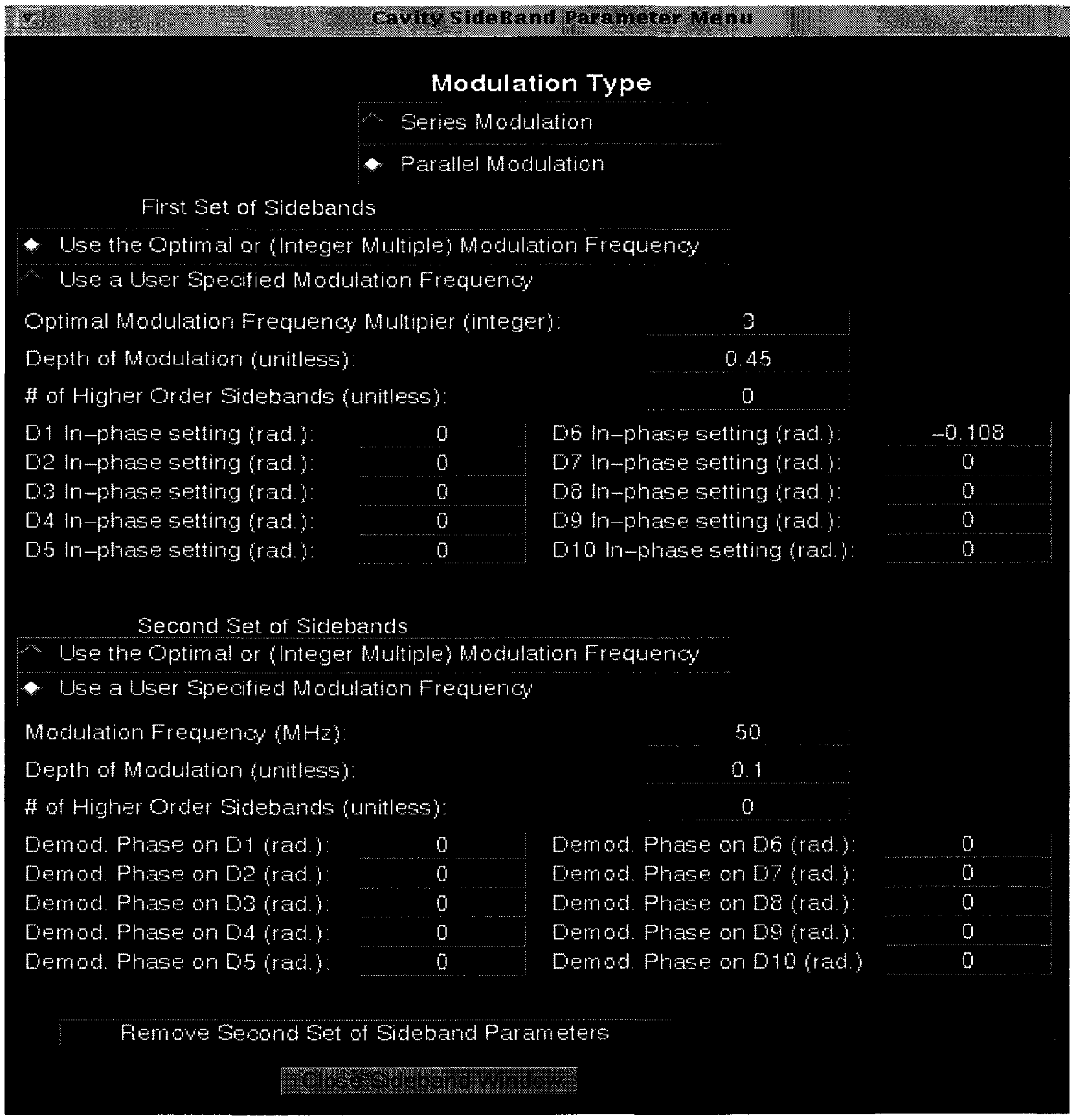
Optimal Modulation Frequency Multipier (integer):	3
Depth of Modulation (unitless):	0.45
# of Higher Order Sidebands (unitless):	0

D1 In-phase setting (rad.):	0	D6 In-phase setting (rad.):	-0.108
D2 In-phase setting (rad.):	0	D7 In-phase setting (rad.):	0
D3 In-phase setting (rad.):	0	D8 In-phase setting (rad.):	0
D4 In-phase setting (rad.):	0	D9 In-phase setting (rad.):	0
D5 In-phase setting (rad.):	0	D10 In-phase setting (rad.):	0

FIGURE 28. **Cavity Sideband Parameter Menu**

The optimal modulation frequency is defined as the frequency such that [(modulation wavelength)/4=average recycling cavity length)]

When the user selects the Set Second Set Of Sideband Parameters, the window looks like the figure below. Also notice that Use a User Specified Modulation Frequency has been selected, and that the Optimal Modulation Frequency Multiplier button has become Modulation Frequency.



Cavity SideBand Parameter Menu

Modulation Type

- Series Modulation
- [✓ Parallel Modulation]

First Set of Sidebands

- [✓ Use the Optimal or (Integer Multiple) Modulation Frequency]
- Use a User Specified Modulation Frequency

Optimal Modulation Frequency Multipier (integer):	3
Depth of Modulation (unitless):	0.45
# of Higher Order Sidebands (unitless):	0

D1 In–phase setting (rad.):	0	D6 In–phase setting (rad.):	–0.108
D2 In–phase setting (rad.):	0	D7 In–phase setting (rad.):	0
D3 In–phase setting (rad.):	0	D8 In–phase setting (rad.):	0
D4 In–phase setting (rad.):	0	D9 In–phase setting (rad.):	0
D5 In–phase setting (rad.):	0	D10 In–phase setting (rad.):	0

Second Set of Sidebands

- Use the Optimal or (Integer Multiple) Modulation Frequency
- [✓ Use a User Specified Modulation Frequency]

Modulation Frequency (MHz):	50
Depth of Modulation (unitless):	0.1
# of Higher Order Sidebands (unitless):	0

Demod. Phase on D1 (rad.):	0	Demod. Phase on D6 (rad.):	0
Demod. Phase on D2 (rad.):	0	Demod. Phase on D7 (rad.):	0
Demod. Phase on D3 (rad.):	0	Demod. Phase on D8 (rad.):	0
Demod. Phase on D4 (rad.):	0	Demod. Phase on D9 (rad.):	0
Demod. Phase on D5 (rad.):	0	Demod. Phase on D10 (rad.)	0

FIGURE 29. **Cavity Sideband Menu**

The Parameters in the Sideband Menu are defined below.

Modulation Type (Series Modulation of Parallel Modulation) defines the method that will be used to modulate the two signals. For more information on Series and Parallel Modulation, see Figure 2.

TABLE 2. Sideband Menu Parameters - First Modulation Frequency

Menu entry name	**Internal variable**	**Description**
Optimal Modulation Frequency Multiplier	FreqMultiplier(1,1)	Integer multiplier which specifies the higher order harmonic of the optimal modulation frequency for the first set of sidebands. Either FreqMultiplier(1,1) or fmod(1,1) can be entered (not both).
Modulation Frequency	fmod(1,1)	User defined modulation frequency for the first set of sidebands. Either FreqMultiplier(1,1) or fmod(1,1) can be entered (not both).
Depth of Modulation	ModAmp(1,1)	The coefficient $J_n(\Gamma)$ are Bessel functions of the first kind. The argument Γ is the Depth of Modulation and n is the order for the first Modulation Frequency
# of Higher Order Sidebands	nSB1	Number of higher order sidebands carried in the first set of sidebands. 0 means that $J1(\Gamma)$ is carried. 1 means that $J1(\Gamma)$ & $J2(\Gamma)$ are carried. 2 means that $J1(\Gamma)$, $J2(\Gamma)$ & $J3(\Gamma)$ are carried.
Dl In-phase setting	Phil(1,1)	Phase setting on demodulator 1 with respect to the first demodulation frequency. Optimal setting for recycling = 0
D2 In-phase setting	Phi3(1,1)	Phase setting on demodulator 2 with respect to the first demodulation frequency. Optimal setting for recycling = 0
D3. In-phase setting	Phi2(1,1)	Phase setting on demodulator 3 with respect to the first demodulation frequency. Optimal setting for recycling = 0
D4 In-phase setting	Phi4(1,1)	Phase setting on demodulator 4 with respect to the first demodulation frequency.
D5. In-phase setting	Phi5(1,1)	Phase setting on demodulator 5 with respect to the first demodulation frequency.

TABLE 2. **Sideband Menu Parameters - First Modulation Frequency**

Menu entry name	**Internal variable**	**Description**
D6. In-phase setting	Phi6(1,1)	Phase setting on demodulator 6 with respect to the first demodulation frequency. Optimal setting for recycling = -2π*LenAsym/(Modulation Wavelength)
D7. In-phase setting	Phi7(1,1)	Phase setting on demodulator 7 with respect to the first demodulation frequency.
D8 In-phase setting	Phi8(1,1)	Phase setting on demodulator 8 with respect to the first demodulation frequency.
D9 In-phase setting	Phi9(1,1)	Phase setting on demodulator 9 with respect to the first demodulation frequency.
D10 In-phase setting	Phi10(1,1)	Phase setting on demodulator 10 with respect to the first demodulation frequency.

If a second modulation frequency is used, the following parameters are also needed.

TABLE 3. **Sideband Menu Parameters - Second Modulation Frequency**

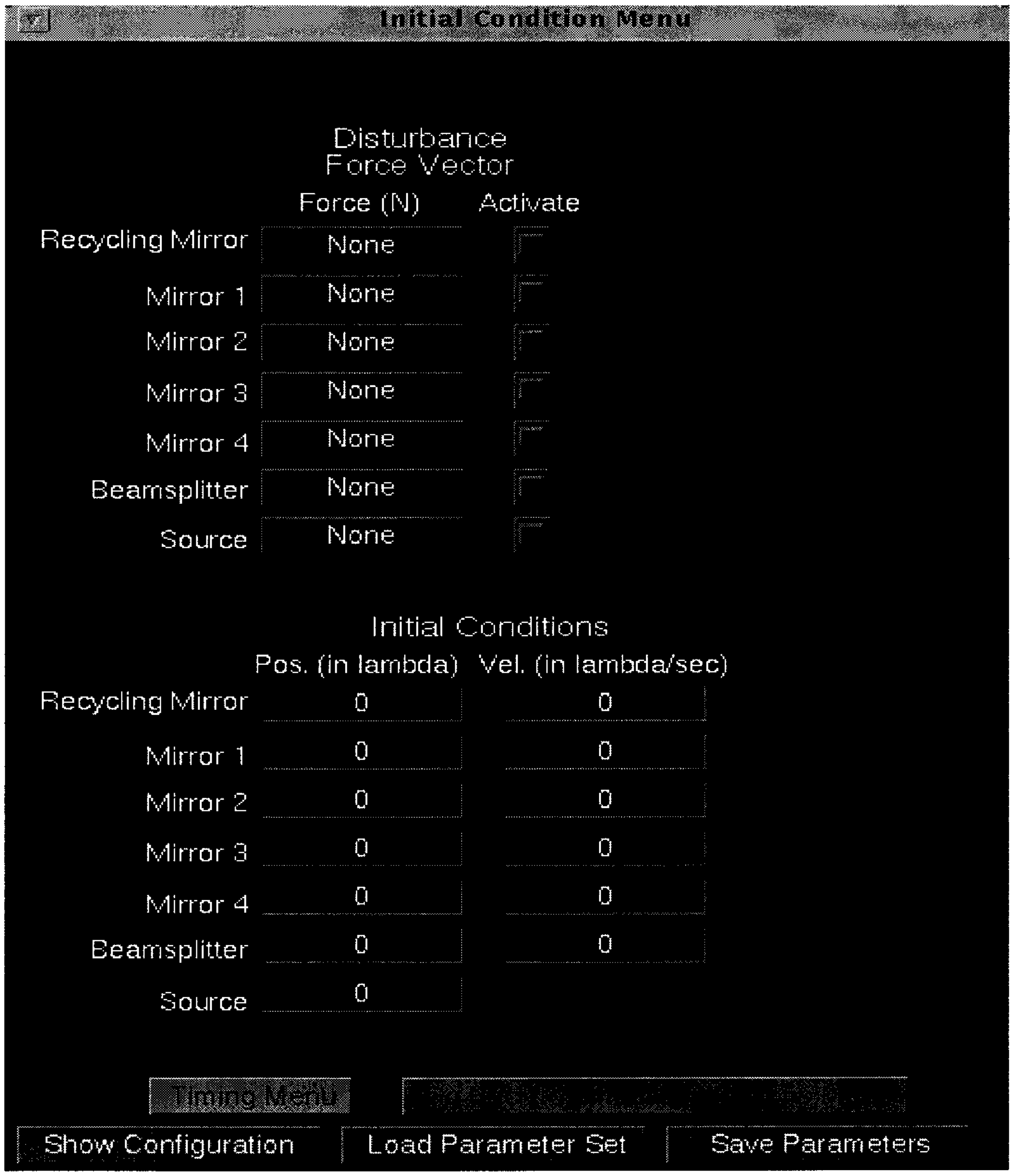
Optimal Modulation Frequency Multiplier	FreqMultiplier(2,1)	Integer multiplier which specifies the higher order harmonic of the optimal modulation frequency for the second set of sidebands. Either FreqMultiplier(2,1) or fmod(2,1) can be entered (not both).
Modulation Frequency	fmod(2,1)	User defined modulation frequency used for the second set of sidebands. Either FreqMultiplier(2,1) or fmod(2,1) can be entered (not both).
Depth of Modulation	ModAmp(2,1)	The coefficient $J_n(\Gamma)$ are Bessel functions of the first kind. The argument Γ is the Depth of Modulation and n is the order for the second Modulation Frequency
# of Higher Order Sidebands	nSB2	Number of higher order sidebands carried in the second set of sidebands. 0 means that J1(Γ) is carried. 1 means that J1(Γ) & J2(Γ) are carried. 2 means that J1(Γ), J2(Γ) & J3(Γ) are carried.
D1 In-phase setting	Phi1(2,1)	Phase setting on demodulator 1 with respect to the second demodulation frequency. Optimal setting for recycling = 0
D2 In-phase setting	Phi3(2,1)	Phase setting on demodulator 2 with respect to the second demodulation frequency.

TABLE 3. Sideband Menu Parameters - Second Modulation Frequency

D3. In-phase setting	Phi2(2,1)	Phase setting on demodulator 3 with respect to the second demodulation frequency.
D4 In-phase setting	Phi4(2,1)	Phase setting on demodulator 4 with respect to the second demodulation frequency.
D5. In-phase setting	Phi5(2,1)	Phase setting on demodulator 5 with respect to the second demodulation frequency.
D6. In-phase setting	Phi6(2,1)	Phase setting on demodulator 6 with respect to the second demodulation frequency.
D7. In-phase setting	Phi7(2,1)	Phase setting on demodulator 7 with respect to the second demodulation frequency.
D8 In-phase setting	Phi8(2,1)	Phase setting on demodulator 8 with respect to the second demodulation frequency.
D9 In-phase setting	Phi9(2,1)	Phase setting on demodulator 9 with respect to the second demodulation frequency.
D10 In-phase setting	Phi10(2,1)	Phase setting on demodulator 10 with respect to the second demodulation frequency.

Initial Conditions Menu

The Initial Condition Menu is used only for Time Simulation Analysis.



Initial Condition Menu

Disturbance
Force Vector

	Force (N)	Activate
Recycling Mirror	None	
Mirror 1	None	
Mirror 2	None	
Mirror 3	None	
Mirror 4	None	
Beamsplitter	None	
Source	None	

Initial Conditions

	Pos. (in lambda)	Vel. (in lambda/sec)
Recycling Mirror	0	0
Mirror 1	0	0
Mirror 2	0	0
Mirror 3	0	0
Mirror 4	0	0
Beamsplitter	0	0
Source	0	

FIGURE 30. **Initial Condition Menu**

The Initial Condition Menu allows the user to input offsets from the resonance conditions and disturbance vectors.

The position initial conditions are distance from resonance conditions in units of lambda. The velocity initial conditions are the initial mirror velocities. The direction of positive initial conditions is shown in Figure 25.

The disturbance inputs are force for the mirror/beamsplitter elements and position for the Source. The name of the file storing the disturbance input is up to the user, however, strict naming conventions exist for the variable storing the data. No other information should be stored in the.mat file other than the disturbance information. The correct variable name for each element is shown in the Table below.

TABLE 4. Correct Name For Disturbance Input

Optical Element	Name
Recycling Mirror	RM_fnoise
Mirror 1	M1_fnoise
Mirror 2	M2_fnoise
Mirror 3	M3_fnoise
Mirror 4	M4_fnoise
Beamsplitter	BS_fnoise
Source	S_fnoise

This variable should be stored in a .mat file. Only this variable should be stored in the .mat file (the Matlab binary format). Time should be stored in the first column of the variable. The disturbance input should be stored in the second column of the variable. The first time step in the disturbance variable should always be at time=0.0. If disturbances are included in the model, The Total Simulation Time (discussed in the following section) is limited by the shortest maximum time included in the disturbance input. Also, since linear interpolation is used, the time/disturbance input should be in chronological order.

Linear interpolation is used to obtain the actual disturbance values put into SMAC. This is very important to the user and the way the disturbances should be implemented. For example, if the user wanted to input a step disturbance between t=0.05

and 0.06 on the recycling mirror, the unwary user might erroneously implement it as shown in the Table below.

TABLE 5. **Wrong Way To Implement Step Input Between t=0.05 & t=0.06**

Time	Disturbance
0.0	0.0
0.05	1.0
0.06	1.0
0.1	0.0

This is incorrect because linear interpolation is used between t=0.0 and t=0.05 and an increasing ramp will be the disturbance input in this case. Similarly, between t=0.06 and t=0.1, a decreasing ramp will be the disturbance input. One way to see errors like this would be to issue the command: `plot(RM_fnoise(:,1),RM_fnoise(:,2))`. The result is shown in the figure below.

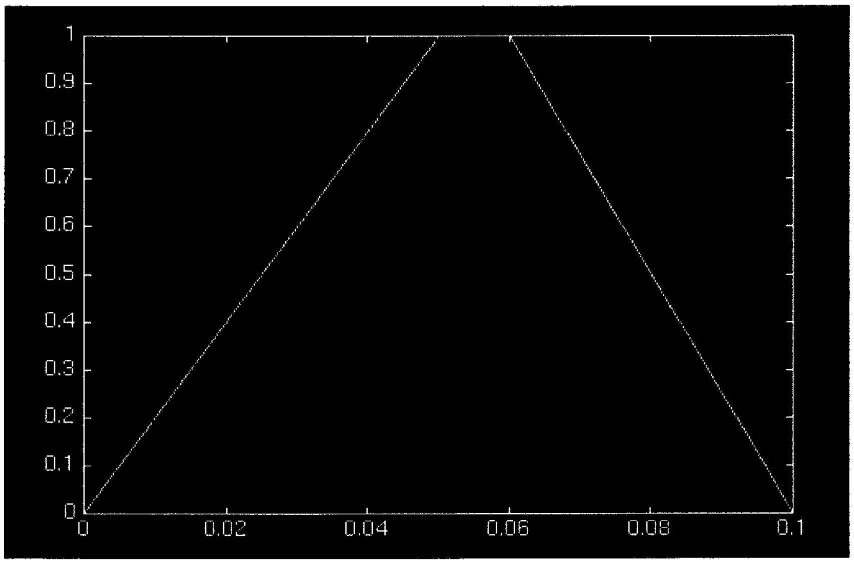


FIGURE 31. **Wrong Way To Implement Step Input Between t=0.05 & t=0.06**

The correct way to input this disturbance is shown in the table and figure below.

TABLE 6. **Right Way To Implement Step Input Between t=0.05 & t=0.06**

Time	**Disturbance**
0.0	0.0
0.05	0.0
0.05	1.0
0.06	1.0
0.06	0.0
0.1	0.0

.

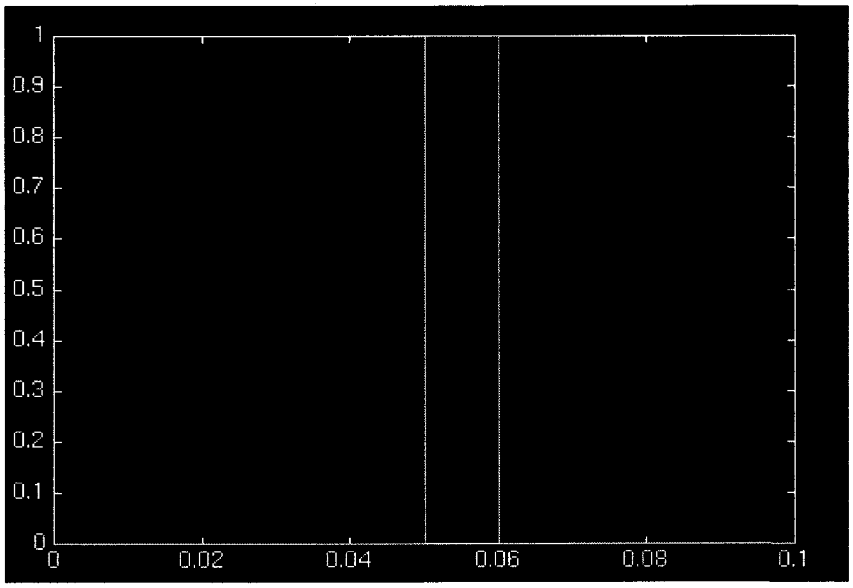


FIGURE 32. **Right Way To Implement Step Input Between t=0.05 & t=0.06**

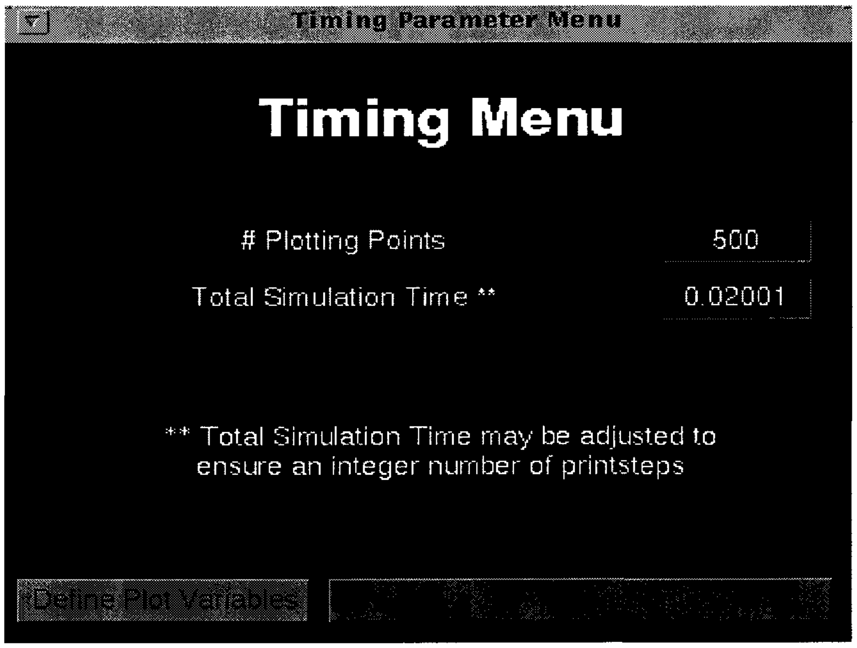
It is important to note that in the above example, if a simulation time step is exactly t=0.05, the value of the disturbance will be 0.0. The first simulation time step that will get the step disturbance is the first time step > 0.05. If the user wants the step to begin at t=0.05, then the second Time value listed in Table 6 should be 0.0499999999. Similarly, if a simulation time step is exactly t=0.06, the value of the disturbance will be 1.0.

Additional difficulty arises if the user wants to input an impulse disturbance. The user must be sure that the disturbance time steps get sampled appropriately during the simulation.

The noise must be activated using the adjacent check box. The direction of positive disturbance is shown in Figure 25.

Timing Parameter Menu

The Timing Parameter Menu is used only during a Time Simulation Analysis. If the user has specified an Open Loop system, the Timing Parameter Menu looks as shown below.

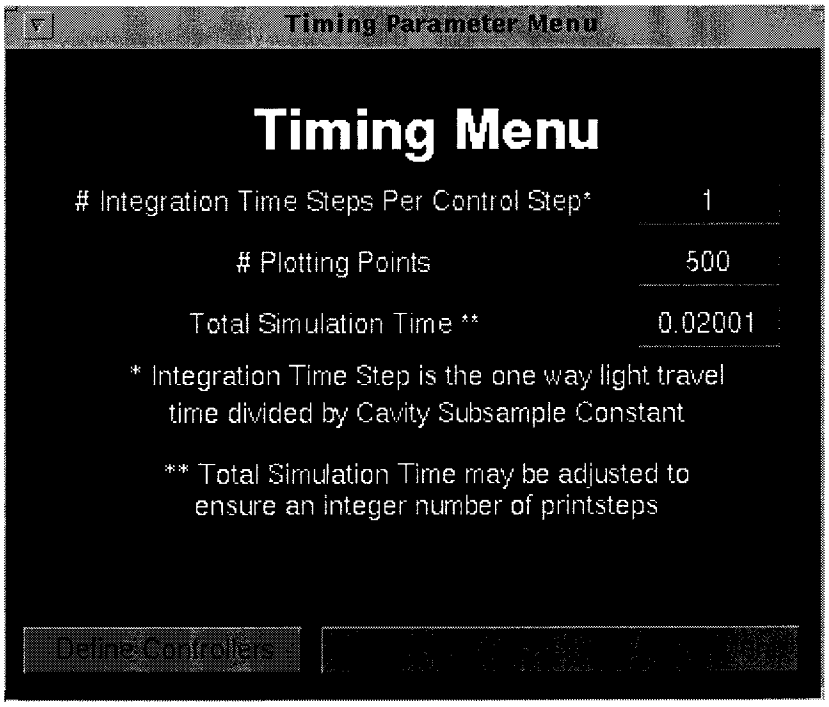


Plotting Points 500
Total Simulation Time ** 0.02001

** Total Simulation Time may be adjusted to ensure an integer number of printsteps

FIGURE 33. Timing Parameter Menu For An Open Loop System

However, if the user has specified a Closed Loop system, the Timing Parameter Menu looks as shown below. The difference between the two is that with an Open Loop simulation, the green button says Pick Plotting Variables and with a Closed Loop simulation, the green button says Define Controllers. There is also one more parameter that needs to be set in the case of closed loop simulation.



Timing Menu

Integration Time Steps Per Control Step* 1

Plotting Points 500

Total Simulation Time ** 0.02001

* Integration Time Step is the one way light travel time divided by Cavity Subsample Constant

** Total Simulation Time may be adjusted to ensure an integer number of printsteps

FIGURE 34. **Timing Parameter Menu For A Closed Loop System**

TABLE 7. Timing Parameter Variables

Menu Entry Name	Internal Variables	Description
# Integration Time Steps Per Control Step	nTauPerControl	Number of τ/Lfac's per control time step (can be used to model sampling in a digital controller)
# Plotting Points	nPrintSteps	Number of points that are plotted
Total Simulation TIme	TtlTime	Total time of simulation in seconds

Increasing the # Plotting Points or the Total Simulation Time will increase the run time of the simulation accordingly.

The Total Simulation Time may be limited if disturbance inputs are included in the model.

Controller Definition Menu

The Controller Menu is used only during a Time Simulation Analysis when closed loop control has been selected. If the user is running an open loop simulation, this menu does not appear and controller values are automatically set for an open loop simulation.

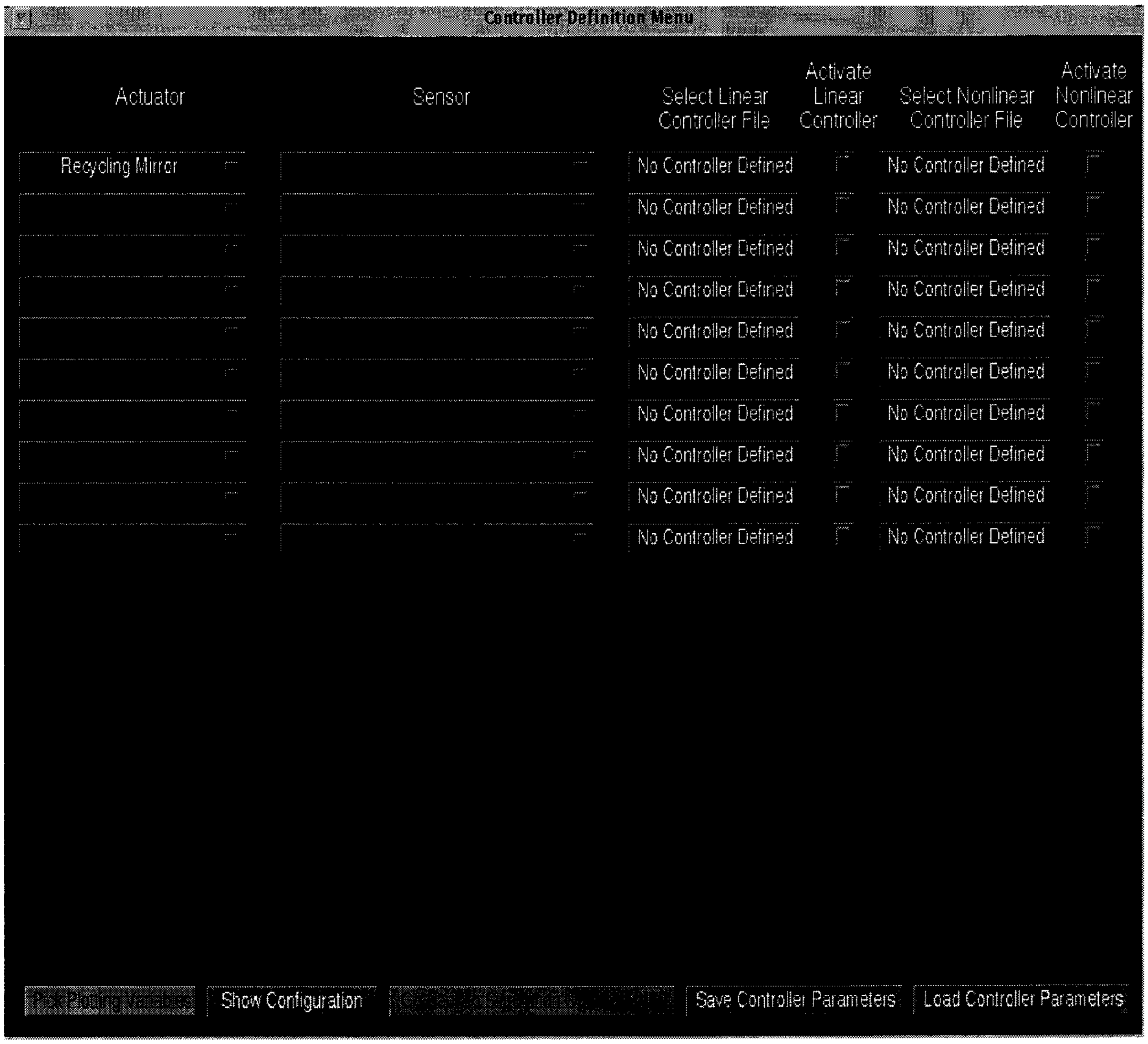
Both linear and nonlinear controllers can be implemented in SMAC.

Certain types of linear controllers can be used in SMAC with a reduced amount of effort on the part of the user. The steps required to add these types of controllers are discussed in this section. Other types of linear controllers and nonlinear controllers are more difficult to implement and are discussed in more detail in Appendix 2.

The criteria that the controllers must meet so that the user can use the easier method of implementation is:

- the controller must be linear
- the controller must be single input, and the input used must be either the demodulated or quadrature output from an available detector
- the controller output must be a scalar, and the actuator must be a single optical element or one of the combinations provided in the GUI

Up to 10 controllers can be implemented in a single simulation run.


Controller Definition Menu
Pick Plotting Variables
Show Configuration
Save Controller Parameters
Load Controller Parameters

Actuator	Sensor	Select Linear Controller File	Activate Linear Controller	Select Nonlinear Controller File	Activate Nonlinear Controller
Recycling Mirror		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	

FIGURE 35. Controller Menu

Linear Controllers

The Actuator and Sensor buttons are pop up buttons. When the user clicks on the Actuator button, the available actuation combinations (Recycling Mirror, Mirror 1, Mirror 2, Mirror 3, Mirror 4, Beamsplitter, Source, M2 + M4, M2 - M4, -(M1 + M2) + (M3 + M4), and (M1 + M2) + (M3 + M4) for the recycling configuration) will appear and the user can select the appropriate actuation. Even though some of the available actuation locations are combinations of mirrors (eg. M2 - M4), this is considered to be a "single output" because the same force value is applied at the different mirrors, in the direction given by the GUI. If the user wants to implement an actuator combination that is not offered (eg. M2 + BS), the user must implement it as a nonlinear controller, even if the controller is a linear controller. Similarly, if the user wants to use a "multi-output" controller where the output values for the controller are distinct, the user must implement it as a nonlinear controller, even if the controller is a linear controller. The user should study Appendix 2 for directions for implementing these types of controllers.

Similarly, when the user clicks on the Sensor button, the available sensor locations will be displayed. The sensors available include the In-Phase signal and the Quadrature signal on each of the available detectors for each of the existing modulation frequencies. The GUI supports "single input" controllers only. If the user wants a "multi-input" controller, the user must implement it as a nonlinear controller, even if it is a linear controller. Similarly, if the user wants to make control decisions based on other sensor input (such as the intensity on detector), the user must implement this as a nonlinear controller. The user is referred to Appendix 2 for more detail on implementing these type of controllers.

The user must also select a linear controller file. When the user clicks on the Select Linear Controller File button, a Matlab window will pop up, forcing the user to specify the .m Matlab file with the linear controller. See below for how to specify the linear controller and what information must be included in the .m Matlab file.

Finally, the user must activate the controller by clicking on the Activate Linear Controller check box. When the linear controller has been activated, a line will be drawn to signify that the linear controller will be used during simulation.

To specify a linear controller, the user must create a .m Matlab file that defines the following variables, ac, bc, cc, dc which are the state variable representation of the controller. The variables must be named ac, bc, cc, and dc. A sample controller file is shown below.

```
% transfer function from controller output in volts to position
% in m/v

z=-2*pi*[1 100 170];
p=-2*pi*[10 1000 208 3000];
gain = 400;

[ac,bc,cc,dc]=zp2ss(z,p,gain);
```

Nonlinear Controllers

If the user wishes to implement a nonlinear controller, the user must also select a nonlinear controller file. When the user clicks on the Specify Nonlinear Controller File button, a Matlab window will pop up, forcing the user to specify the .f Fortran file with the nonlinear controller. See Appendix 2 for more information on nonlinear controllers. The user must also activate the nonlinear controller by clicking on the Active Nonlinear Controller check box. When the nonlinear controller has been activated, a line will be drawn to signify that the nonlinear controller will be used during simulation.

The user must specify a Sensor/Actuator combination and a Linear Controller File even when a nonlinear controller is being used. Even though it is possible for the user to include the sensor/actuator and state space information in the nonlinear Fortran routine, the user must specify these values on the GUI menu.

To specify a nonlinear controller, the user must create a .f Fortran file. The user can use any of the demodulated signals, quadrature signals, detector intensities, and first or second sideband modulation frequency detector intensities as input signals to the controller. The name of the subroutine depends on the position of controller in the GUI. This will be explained further in a following paragraph. Certain declarations are required in a nonlinear .f subroutine, and these are included in detail in Appendix 2.

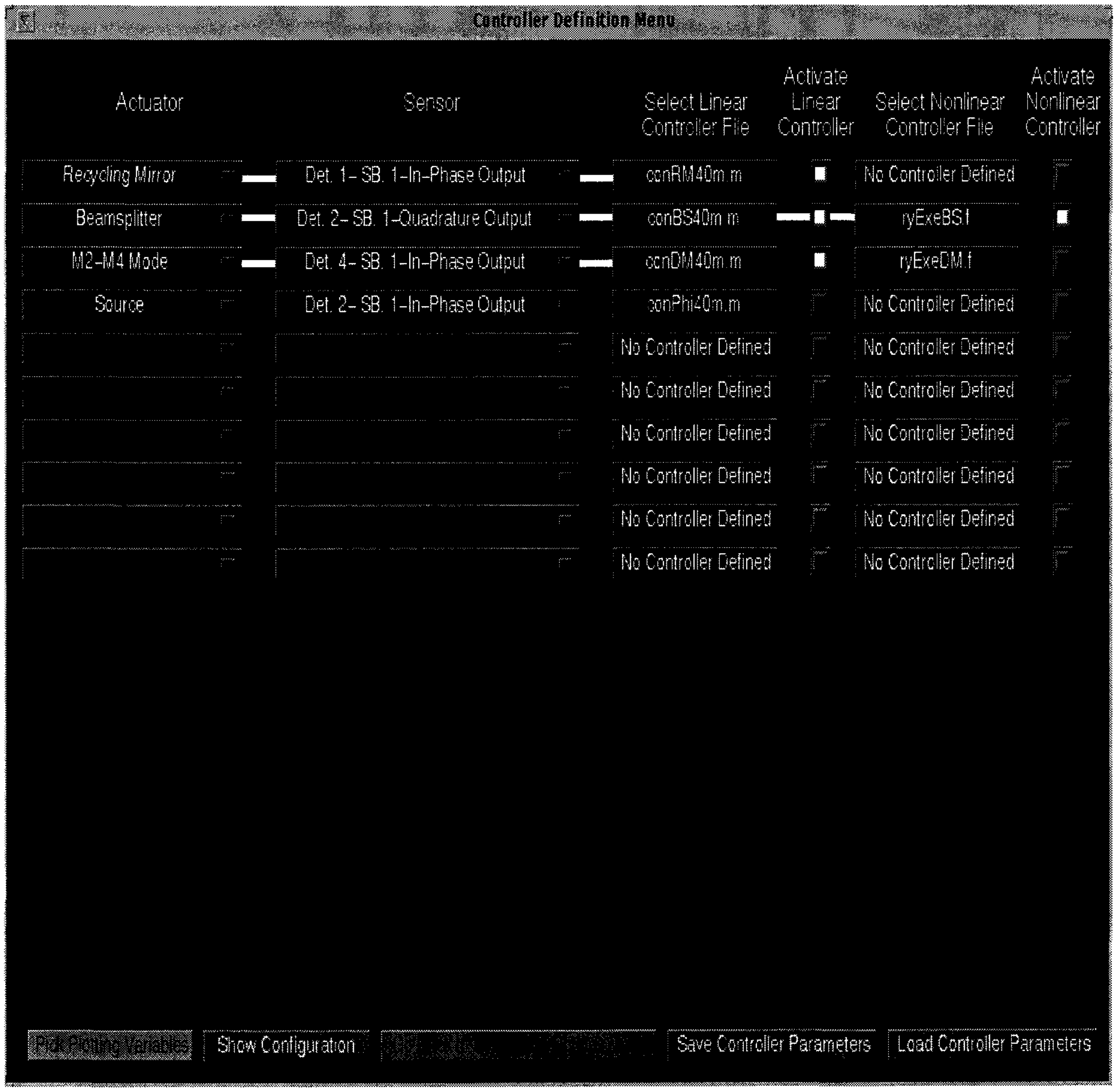
The variables passed in to the subroutine are defined below.

TABLE 8. Variables Passed to the Nonlinear Control Subroutine

fVec	[force applied to RM in kg*m/sec^2;
	force applied to M1 in kg*m/sec^2;
	force applied to M2 in kg*m/sec^2;
	force applied to M3 in kg*m/sec^2;
	force applied to M4 in kg*m/sec^2;
	force applied to BS in kg*m/sec^2;
	equivalent Source displacement in units of m]

i2f	the actuator(s) for this control calculation (as specified in the GUI)
S	User defined linear controller state space model in a combined format. S=[ac bc;cc dc] where "ac" is the state space model, "bc" is a column vector of length n-1, "cc" is a row vector of length n-1, and "dc" is a scalar. This is the state space matrix that was specified in the linear controller .m file.
xA	States in the state space model + an extra zero so the vector is of length n. This is the state vector that was specified in the linear controller .m file.
n	length of vector x, which is equal to the number of states of the state space model + 1
v	v = the value of the signal that drives the controller (the sensor input) (as specified in the GUI)
simTime	simTime = scalar value of time that the simulation is using at this step in the program

In the figure below, 4 controllers have been defined. 2 controllers are linear and 2 controllers are non-linear. However, only 3 of the linear controllers have been activated and only 1 of the non-linear controllers has been activated. ryExeBS.f is the "second" controller on the GUI, so the subroutine must be called ryExeControl2. Similarly, if the second non-linear controller were activated, the subroutine would have to be named ryExeControl3, since it is the "third" controller listed on the GUI. It is easy to tell that a non-linear controller has been activated because the line extends farther than it does in the "linear only" case.


Controller Definition Menu
Pick Plotting Variables
Show Configuration
Save Controller Parameters
Load Controller Parameters

Actuator	Sensor	Select Linear Controller File	Activate Linear Controller	Select Nonlinear Controller File	Activate Nonlinear Controller
Recycling Mirror	Det. 1- SB. 1-In-Phase Output	conRM40m.m	■	No Controller Defined	
Beamsplitter	Det. 2- SB. 1-Quadrature Output	conBS40m.m	■	ryExeBS.f	■
M2-M4 Mode	Det. 4- SB. 1-In-Phase Output	conDM40m.m	■	ryExeDM.f	
Source	Det. 2- SB. 1-In-Phase Output	conPhi40m.m		No Controller Defined	
		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	
		No Controller Defined		No Controller Defined	

FIGURE 36. **Controller Menu**

Plotting Options Menu

The final selection the user needs to make is with regard to plotting options. The main plotting menu allows the user to select whether the plots should be sent to the screen, the printer, or a postscript file for printing at a later time. The main plotting menu also gives the user the opportunity to select which plots should be displayed. The plotting menu also allows the user to **Save Parameters to Text File**, which encapsulates relevant data for future reference. The user will often store a printed copy of this file with the printed copies of plots for easy reference.

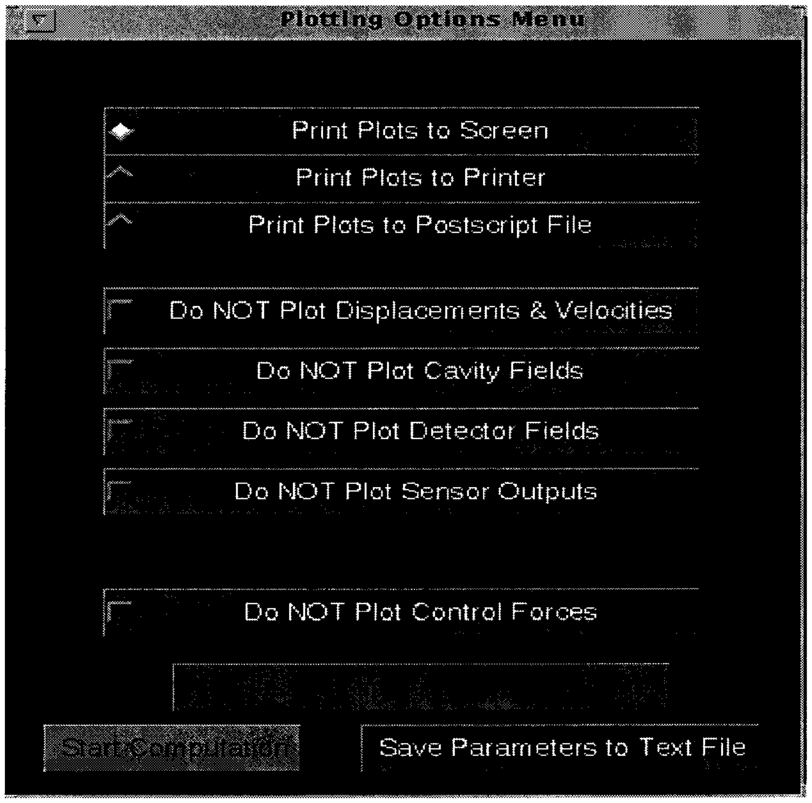

Plotting Options Menu
Print Plots to Screen
Print Plots to Printer
Print Plots to Postscript File
Do NOT Plot Displacements & Velocities
Do NOT Plot Cavity Fields
Do NOT Plot Detector Fields
Do NOT Plot Sensor Outputs
Do NOT Plot Control Forces
Start Computation
Save Parameters to Text File

FIGURE 37. **Plotting Options Menu**

When the user selects some of the options on the **Main Plotting Menu**, an additional window will come up.

The menu for selecting which displacement and velocity variables will be plotted is shown below.

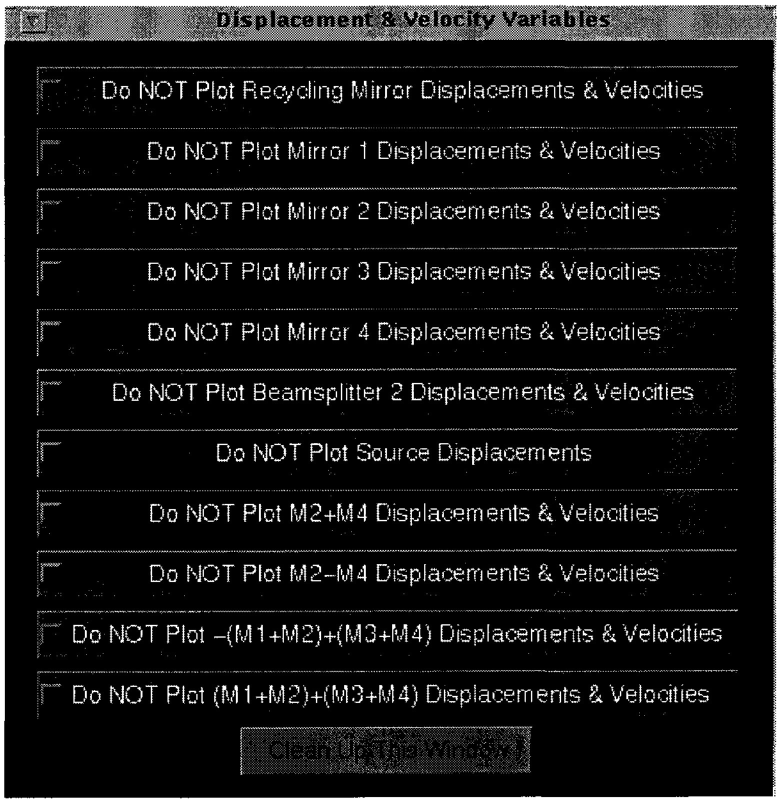
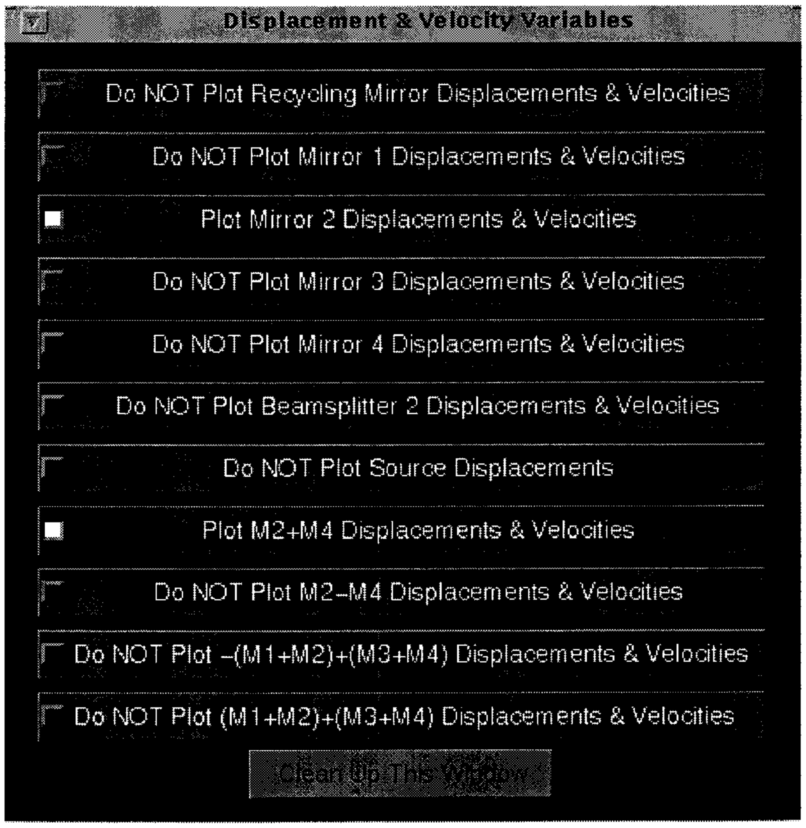

Displacement & Velocity Variables
Do NOT Plot Recycling Mirror Displacements & Velocities
Do NOT Plot Mirror 1 Displacements & Velocities
Do NOT Plot Mirror 2 Displacements & Velocities
Do NOT Plot Mirror 3 Displacements & Velocities
Do NOT Plot Mirror 4 Displacements & Velocities
Do NOT Plot Beamsplitter 2 Displacements & Velocities
Do NOT Plot Source Displacements
Do NOT Plot M2+M4 Displacements & Velocities
Do NOT Plot M2-M4 Displacements & Velocities
Do NOT Plot -(M1+M2)+(M3+M4) Displacements & Velocities
Do NOT Plot (M1+M2)+(M3+M4) Displacements & Velocities
Clean Up This Window!

FIGURE 38. Plot Displacement & Velocity Variables

As with all the plotting menus, the words "Do NOT" will be removed when the user elects to plot the variable.


Displacement & Velocity Variables

☐ Do NOT Plot Recycling Mirror Displacements & Velocities

☐ Do NOT Plot Mirror 1 Displacements & Velocities

■ Plot Mirror 2 Displacements & Velocities

☐ Do NOT Plot Mirror 3 Displacements & Velocities

☐ Do NOT Plot Mirror 4 Displacements & Velocities

☐ Do NOT Plot Beamsplitter 2 Displacements & Velocities

☐ Do NOT Plot Source Displacements

■ Plot M2+M4 Displacements & Velocities

☐ Do NOT Plot M2–M4 Displacements & Velocities

☐ Do NOT Plot –(M1+M2)+(M3+M4) Displacements & Velocities

☐ Do NOT Plot (M1+M2)+(M3+M4) Displacements & Velocities

Clean Up This Window

FIGURE 39. Plot Displacement & Velocity Variables Menu

The menu for selecting which detector intensities will be plotted is shown below. As with all the plotting menus, if a user is running a configuration other than the recycling configuration, not all of the displayed options will appear since not all of the variables are active in all configurations.

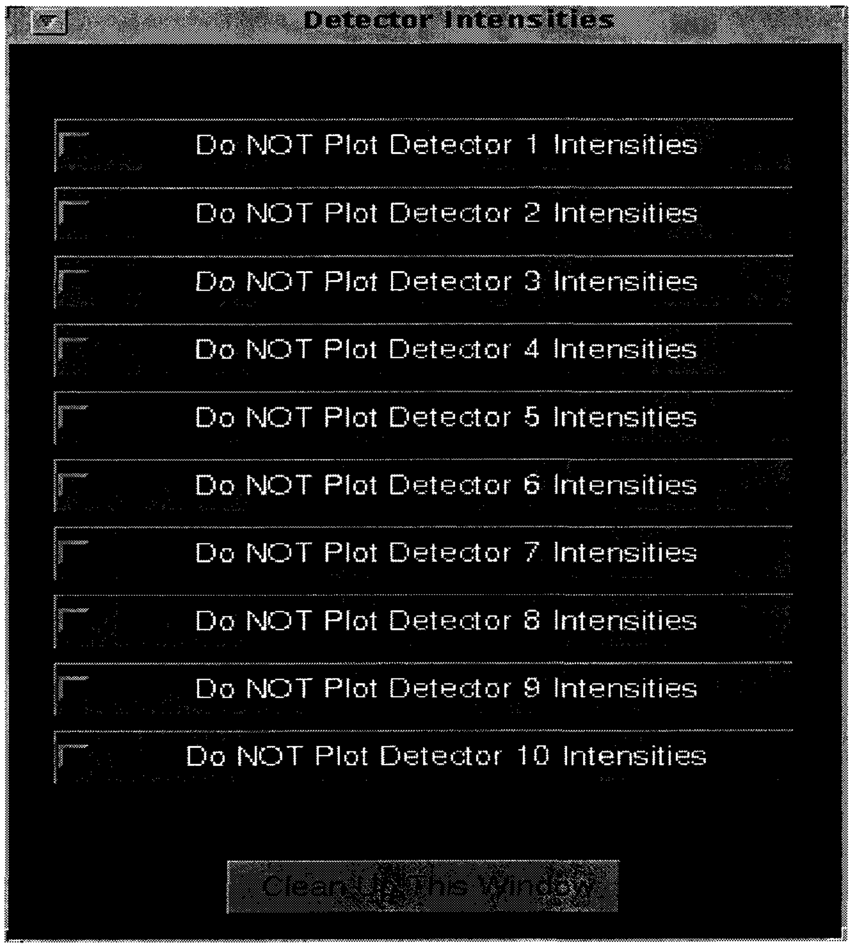

Detector Intensities
Do NOT Plot Detector 1 Intensities
Do NOT Plot Detector 2 Intensities
Do NOT Plot Detector 3 Intensities
Do NOT Plot Detector 4 Intensities
Do NOT Plot Detector 5 Intensities
Do NOT Plot Detector 6 Intensities
Do NOT Plot Detector 7 Intensities
Do NOT Plot Detector 8 Intensities
Do NOT Plot Detector 9 Intensities
Do NOT Plot Detector 10 Intensities
Clean Up This Window

FIGURE 40. **Plot Detector Intensities Menu**

The menu which allows the user to select which sensor variables are to be plotted is shown below. If the user only has one modulation frequency, the right half of the menu will not be visible.

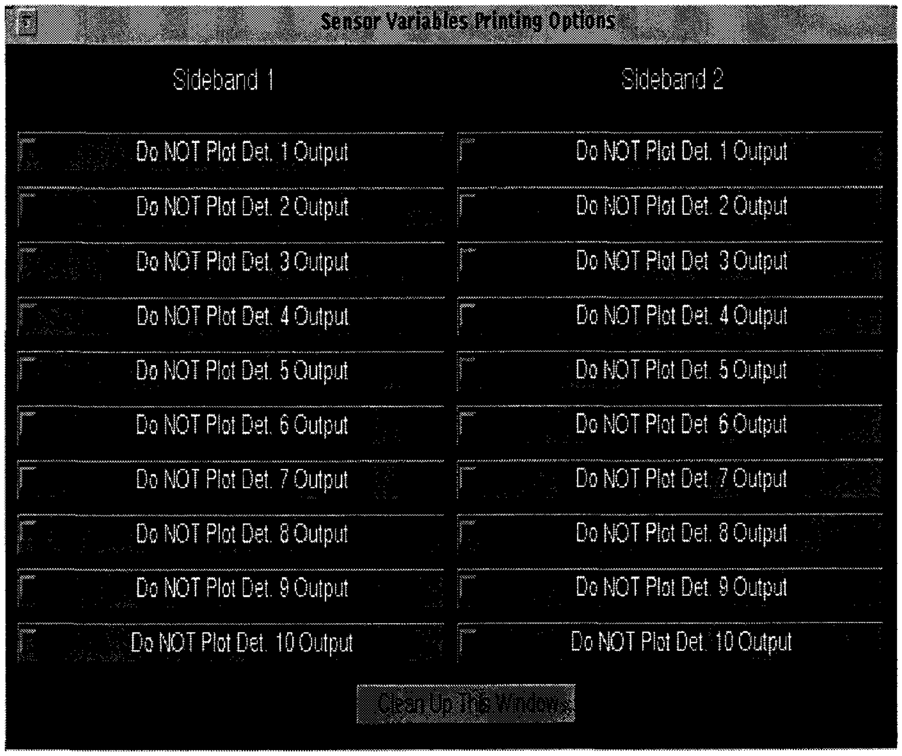

Sensor Variables Printing Options
Sideband 1
Sideband 2
Do NOT Plot Det. 1 Output
Do NOT Plot Det. 2 Output
Do NOT Plot Det. 3 Output
Do NOT Plot Det. 4 Output
Do NOT Plot Det. 5 Output
Do NOT Plot Det. 6 Output
Do NOT Plot Det. 7 Output
Do NOT Plot Det. 8 Output
Do NOT Plot Det. 9 Output
Do NOT Plot Det. 10 Output
Clean Up This Window

FIGURE 41. Plot Sensor Variables Menu

After the plotting options have been chosen, the user should click on the Start Computation button. The time simulation will begin. The messages shown below will appear in the Matlab window which will help the user determine how the simulation is progressing.

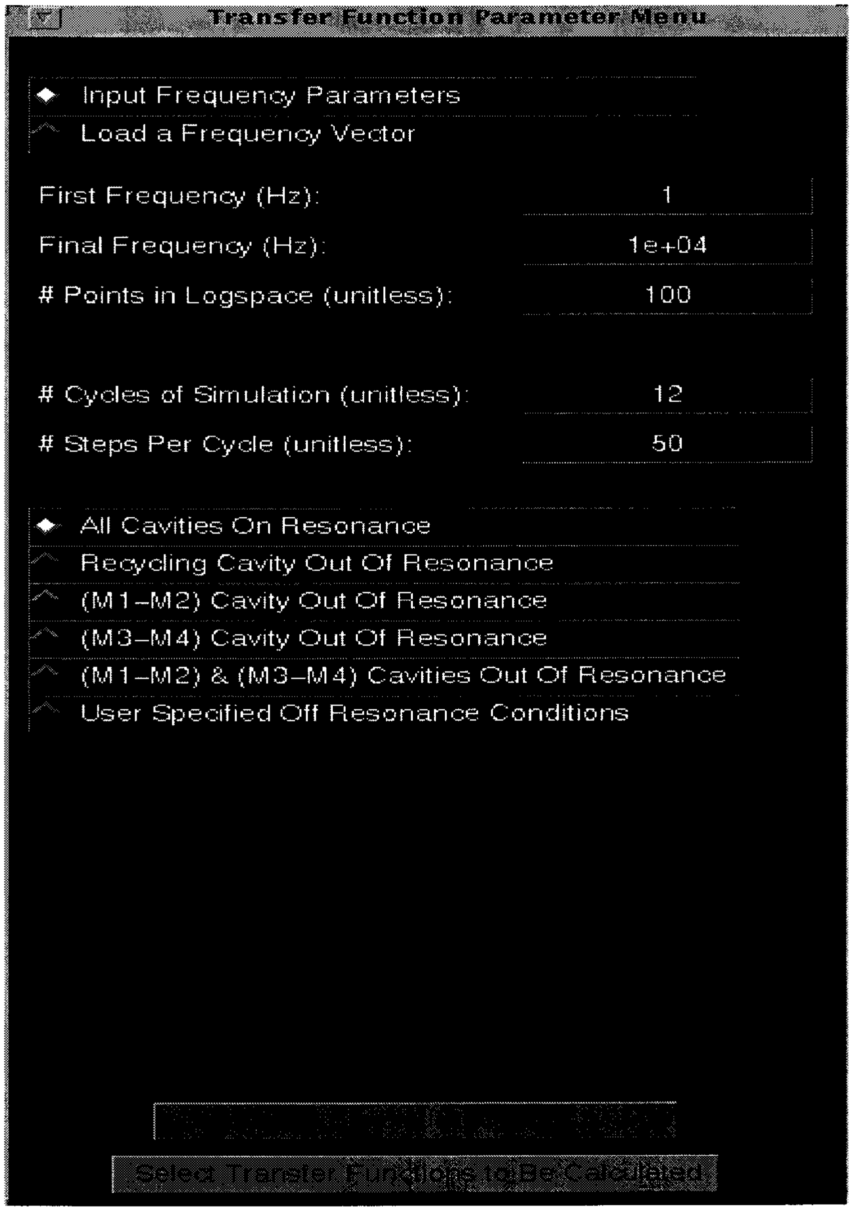
```
Starting Computation

Working with Mixed Mode

Simulation is 10% complete
Simulation is 20% complete
Simulation is 30% complete
Simulation is 40% complete
Simulation is 50% complete
Simulation is 60% complete
Simulation is 70% complete
Simulation is 80% complete
Simulation is 90% complete
Simulation is 100% complete
```

4.2 Transfer Functions

The first several menus needed for calculating transfer functions are the same as for calculating time simulations. First, the user needs to select which configuration is desired from the SMAC Main Menu. This is shown in Figure 23. Second, the user will select the Simulation Options desired. This is shown in Figure 26. Next, the user defines the Cavity Parameters. This is shown in Figure 27. It is here that the menus begin to differ from the time simulation menus.

Transfer Function Parameter Menu

The Transfer Function Parameter Menu allows the user to input parameters relevant to how the transfer functions are calculated. The menu is shown below.



Transfer Function Parameter Menu

[✓ Input Frequency Parameters]
Load a Frequency Vector

First Frequency (Hz):	1
Final Frequency (Hz):	1e+04
# Points in Logspace (unitless):	100
# Cycles of Simulation (unitless):	12
# Steps Per Cycle (unitless):	50

[✓ All Cavities On Resonance]
Recycling Cavity Out Of Resonance
(M1–M2) Cavity Out Of Resonance
(M3–M4) Cavity Out Of Resonance
(M1–M2) & (M3–M4) Cavities Out Of Resonance
User Specified Off Resonance Conditions

[illegible]

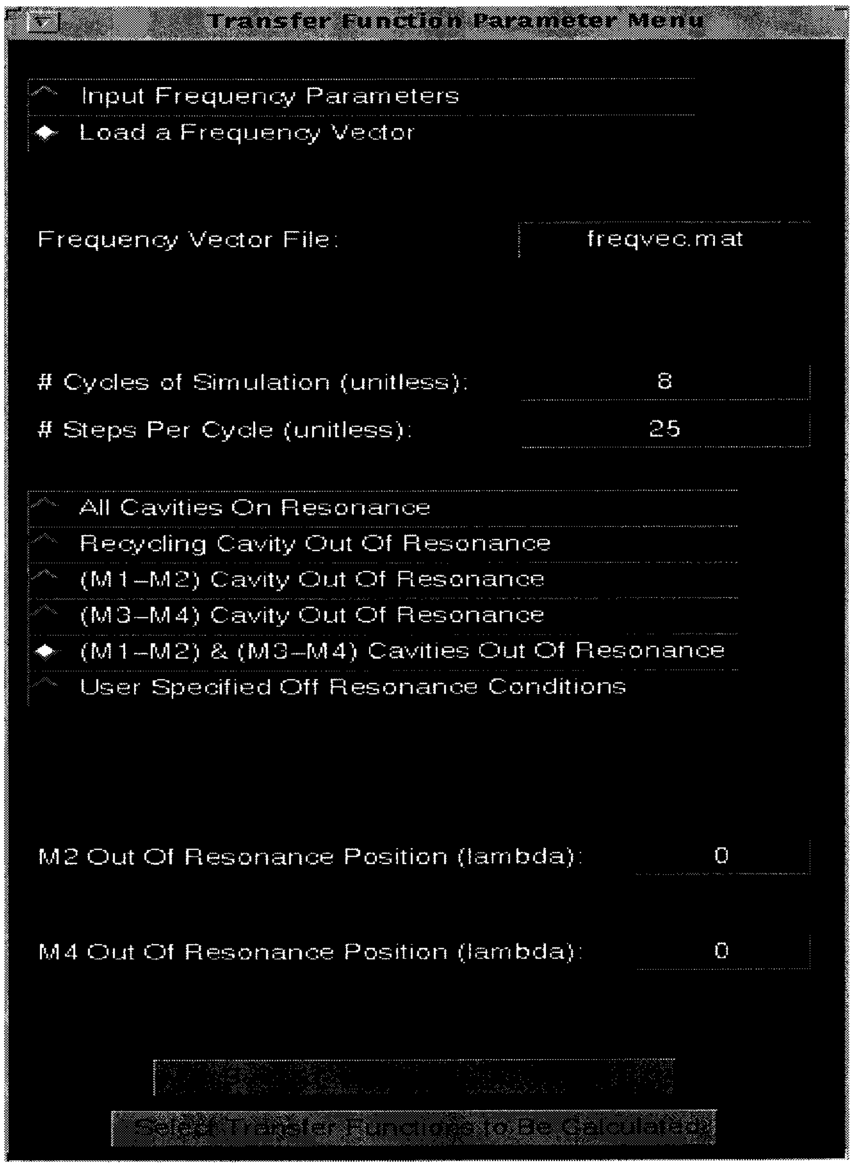
Select Transfer Functions to Be Calculated

FIGURE 42. Transfer Function Parameter Menu

The first vector to be specified is the set of frequencies at which the transfer functions are calculated. The user can specify the frequencies in 2 ways. The user can input frequency parameters (the first frequency, the last frequency, and the number of frequencies at which the transfer functions should be calculated). SMAC will then generate a frequency vector with that number of frequencies evenly spaced in logspace between the first and final frequency specified. The second way to specify the frequency vector is for the user to generate it, and use the GUI to specify the name of the file where it is stored. The user-generated frequency vector must be named FoscVec and it must be stored in a .mat file (which can have any name). No other variables should be stored in this .mat file. The appearance of the GUI when a user-generated frequency file is specified will be shown in the figure below.

The second set of parameters to be specified define the resolution and accuracy of the calculation. The calculation is carried out by injecting a sine wave at the disturbance point for each frequency in FoscVec that is nCycle (# of Cycles of Simulation) long. The number of points simulated in each cycle is nStepsPerCycle (# Steps Per Cycle). At each frequency, the ratio of the max amplitude at the sensing point to the max amplitude at the disturbance point in the final cycle is the calculated transfer function gain. To speed up the calculation time, both of these parameters can be reduced but with a loss of accuracy and resolution.

The third set of menu choices allows the user to specify the mirror locations at which the transfer functions will be calculated. If "All Cavities on Resonance" is chosen, the mirror positions will all be set to their nominal 0 position. If "(M1-M2 & M3-M4)" is chosen, additional buttons will appear on the menu that allow the user to choose how far off resonance M2 and M4 will be set from their nominal 0 position. The off-resonance values are specified in lambda with positive lambda being defined as shown in Figure 25. Off- resonance conditions may be entered as numbers (eg. 0.1), or as expressions (eg. (2*3/pi). The appearance of the GUI when off-resonance conditions are specified is shown below



Input Frequency Parameters
[✓ Load a Frequency Vector]

Frequency Vector File: freqvec.mat

Cycles of Simulation (unitless): 8

Steps Per Cycle (unitless): 25

All Cavities On Resonance
Recycling Cavity Out Of Resonance
(M1–M2) Cavity Out Of Resonance
(M3–M4) Cavity Out Of Resonance
[✓ (M1–M2) & (M3–M4) Cavities Out Of Resonance]
User Specified Off Resonance Conditions

M2 Out Of Resonance Position (lambda): 0

M4 Out Of Resonance Position (lambda): 0

FIGURE 43. Transfer Function Parameter Menu

Select Transfer Function Menu

The user now selects which transfer functions are to be calculated. The user indicates that a transfer function should be calculated by clicking on an actuator and a sensor, (or a sensor and an actuator - order is not important). A line will be drawn between the sensor and actuator to indicate that it has been selected. To unselect a transfer function that has already been selected, the user should click on the sensor/actuator combination a second time, and the line connecting the two will disappear. The user should notice the Connect All and Disconnect All buttons at the bottom of the menu which were added to facilitate the specification of transfer functions when many are desired.

If more than one modulation frequency is being used, the user can toggle between the Sideband 1 menu and the Sideband 2 menu using the button at the bottom of the menu.

Transfer functions are calculated on an actuator by actuator basis. This means that it takes the same amount of time to calculate the

- "Beamsplitter" to "Detector 1 - In-Phase Output"

as it does to calculate the

- "Beamsplitter" to "All Sensor Outputs".

Similarly, it takes approximately twice as long to calculate the

- "Beamsplitter" to "Detector 1 - In-Phase Output"
- "Recycling Mirror" to "Detector 1 - Quadrature Output"

as it does to calculate

- "Mirror 1" to "Detector 1 - In-Phase Output"
- "Mirror 1" to "Detector 2 - In-Phase Output"

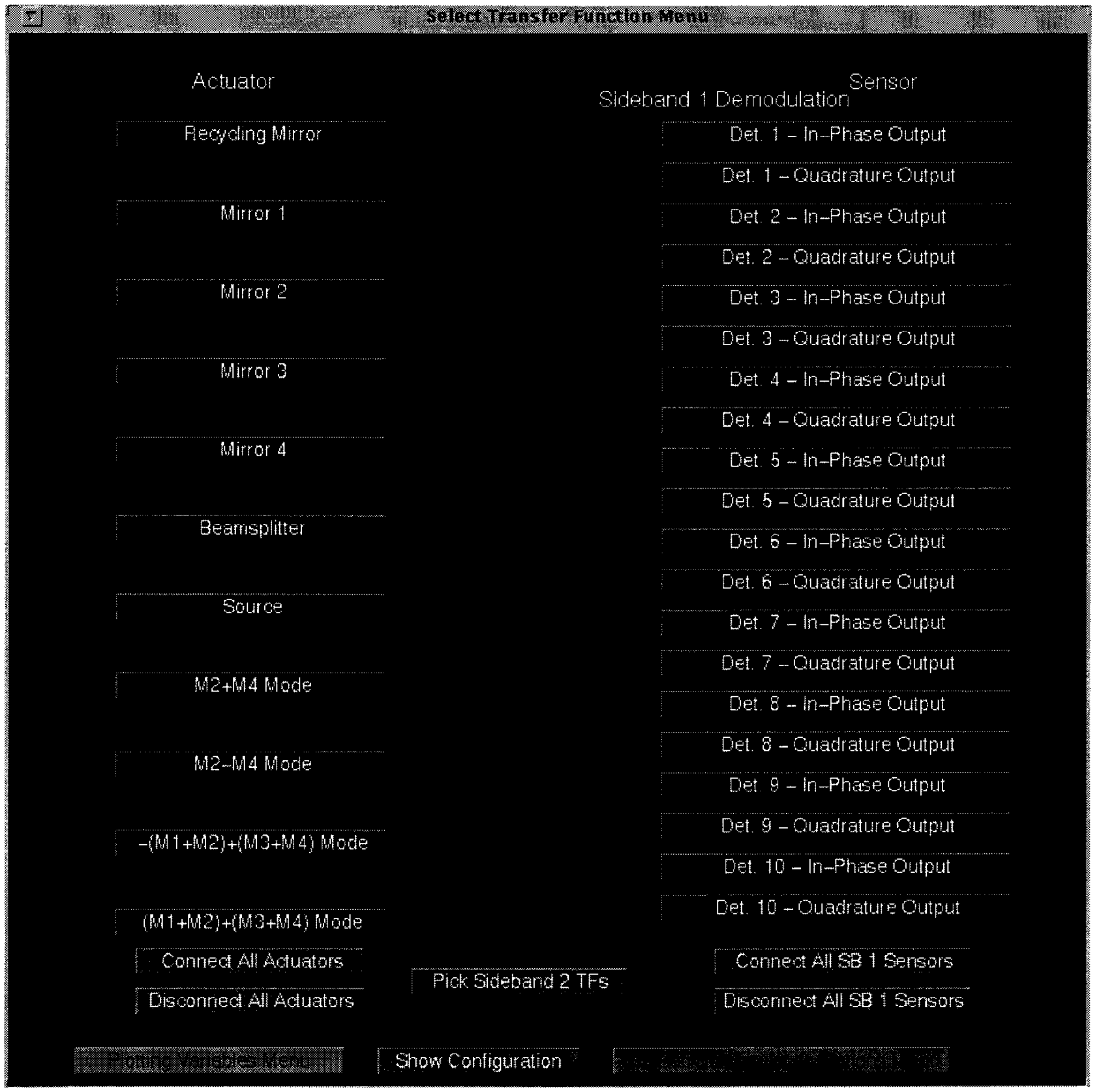


FIGURE 44. Select Transfer Function Menu

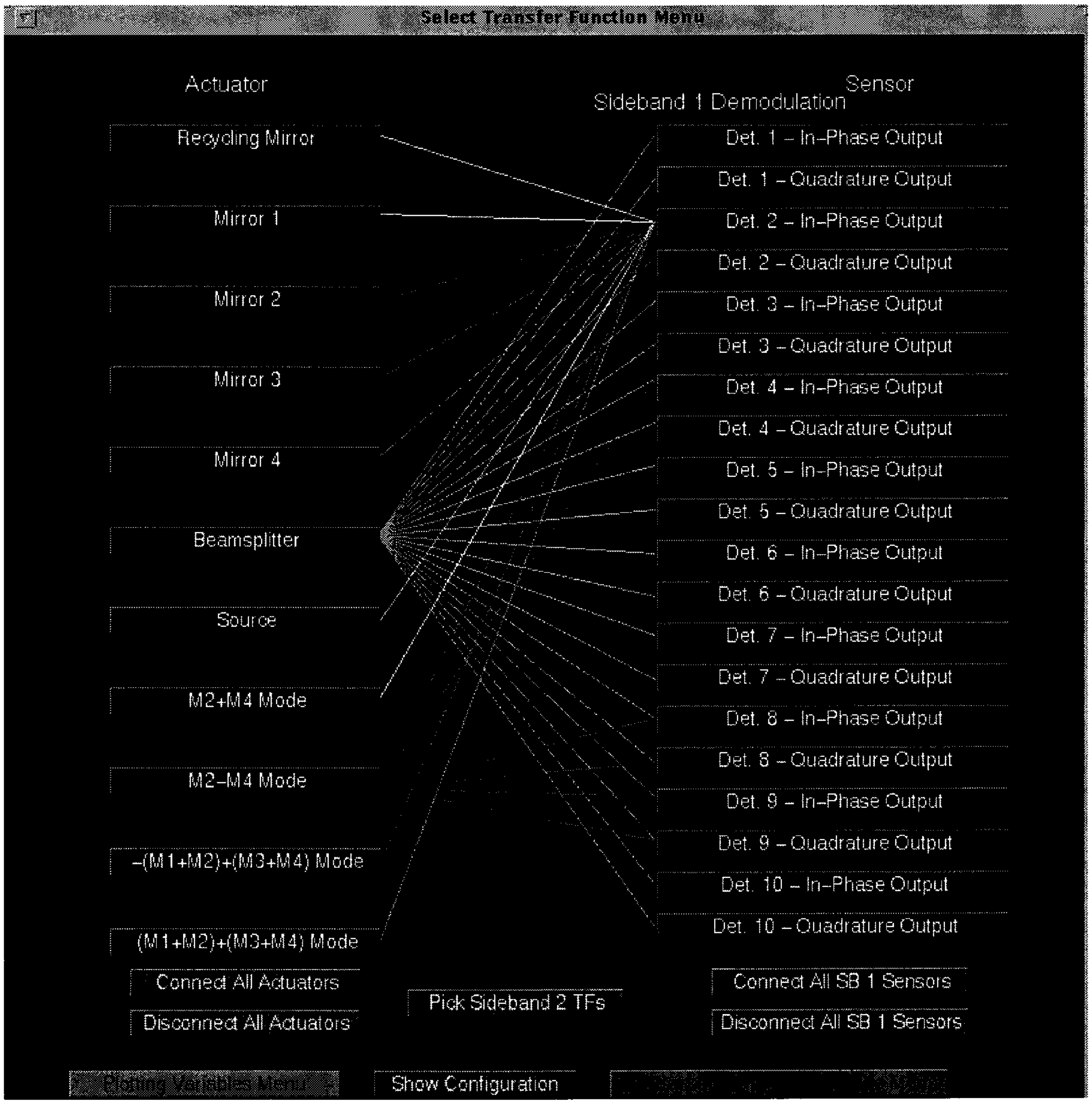


FIGURE 45. Select Transfer Function Parameter Menu

After the transfer functions have been specified, the user defines plotting information.

After the user clicks on the **Start Computation** button, the following messages will appear in the Matlab window. This will help the user determine how the transfer function calculations are progressing. The user should keep in mind that the lower frequency and higher frequency transfer functions may take significantly longer than middle frequency transfer functions. The messages below are based only on the number of frequencies, not the length of time it will take to caluclate the transfer function. For example, 0.0001 Hz is weighted the same as 1000 Hz when the messages are displayed, when in almost all cases, 0.0001 Hz will take significantly longer to calculate than 1000 Hz.

```
>> Starting Computation

Working with Mixed Mode

TF Analysis is 10% complete for 1 of 2 DOF
TF Analysis is 20% complete for 1 of 2 DOF
TF Analysis is 30% complete for 1 of 2 DOF
TF Analysis is 40% complete for 1 of 2 DOF
TF Analysis is 50% complete for 1 of 2 DOF
TF Analysis is 60% complete for 1 of 2 DOF
TF Analysis is 70% complete for 1 of 2 DOF
TF Analysis is 80% complete for 1 of 2 DOF
TF Analysis is 90% complete for 1 of 2 DOF
TF Analysis is 100% complete for 1 of 2 DOF
TF Analysis is 10% complete for 2 of 2 DOF
TF Analysis is 20% complete for 2 of 2 DOF
TF Analysis is 30% complete for 2 of 2 DOF
TF Analysis is 40% complete for 2 of 2 DOF
TF Analysis is 50% complete for 2 of 2 DOF
TF Analysis is 60% complete for 2 of 2 DOF
TF Analysis is 70% complete for 2 of 2 DOF
TF Analysis is 80% complete for 2 of 2 DOF
TF Analysis is 90% complete for 2 of 2 DOF
TF Analysis is 100% complete for 2 of 2 DOF
```

4.3 Continue Analysis Menu

Once the time simulation or transfer functions are calculated and the desired plots have been displayed, a menu will come up which will allow the user to go back to modify specifications.

The appearance of the menu will differ based on whether transfer functions or time simulations were calculated.

The menu for time simulation analysis is shown below.

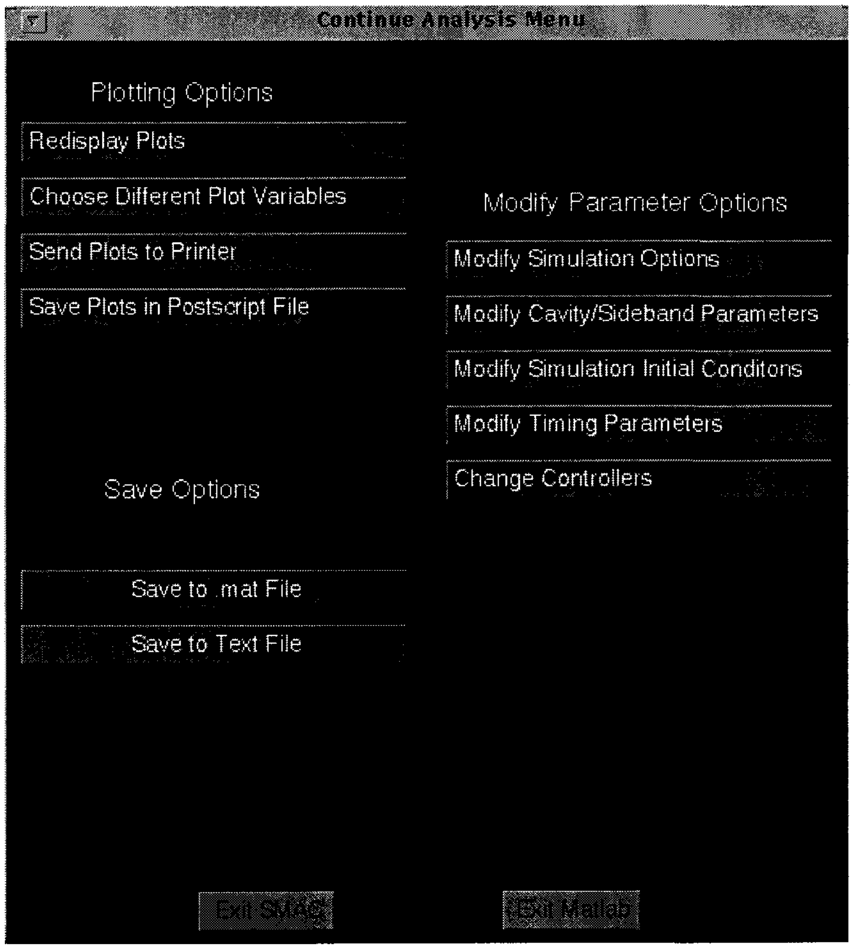


FIGURE 46. Continue Analysis Menu For Time Simulations

The menu for transfer function analysis is shown below.

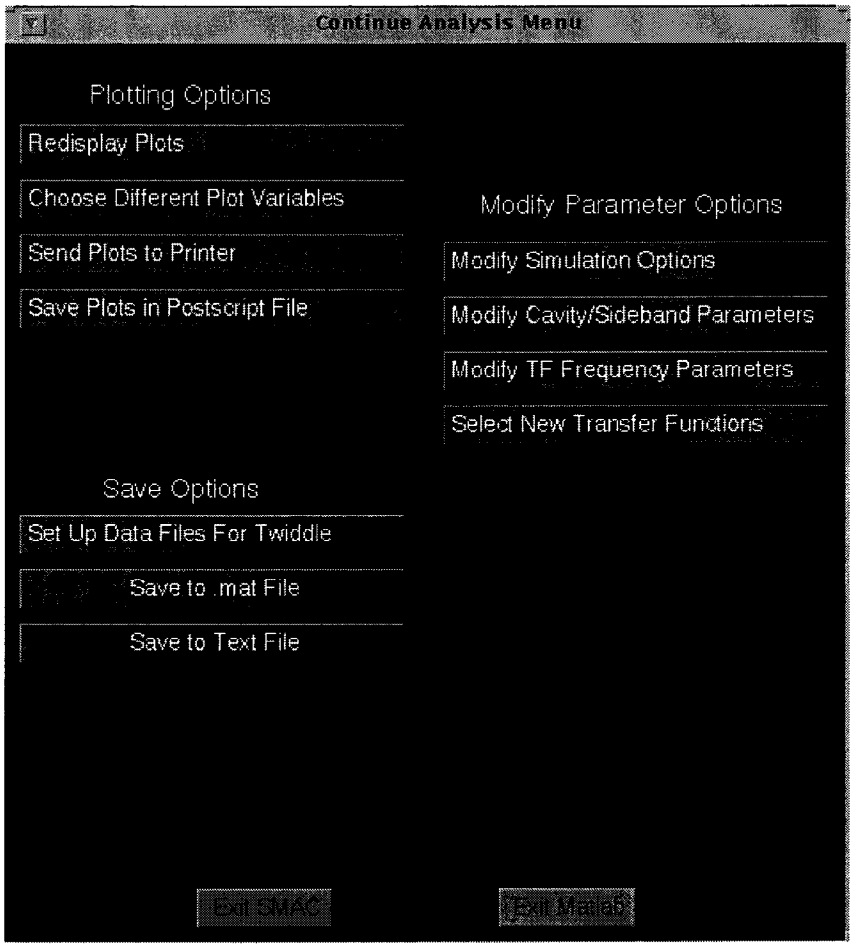

Continue Analysis Menu
Plotting Options
Redisplay Plots
Choose Different Plot Variables
Send Plots to Printer
Save Plots in Postscript File
Modify Parameter Options
Modify Simulation Options
Modify Cavity/Sideband Parameters
Modify TF Frequency Parameters
Select New Transfer Functions
Save Options
Set Up Data Files For Twiddle
Save to .mat File
Save to Text File
Exit SMAC
Exit Matlab

FIGURE 47. Continue Analysis Menu For Transfer Function Analysis

The **Set Up Data Files For Twiddle** button generates the input files for the Mathematica based Twiddle that has been used to verify Transfer Function output. The user should use the **Save to .mat File** button to save the SMAC transfer functions in a Matlab .mat file. Then, the user should use Mathematica to run the twiddle transfer functions. After this has been completed, the user can compare the two by restarting Matlab and using "TFcheck" to run the comparisons. See Appendix 4 for more detail.

The **Save to .mat File** button will save all the data to a Matlab .mat file.

The **Save to Text File** button will generate an ascii text file with the important parameters. This button produces identical results as the **Save Parameters to Text File** button on the plotting menu.

CHAPTER 5

Default Cases

Several default sets of cavity parameters are provided. This section describes the default cases and explains how to access the parameter sets.

Case1.mat = 4 km with green light - recycled configuration

Case2.mat = 4 km with green light - coupled cavity (M1-M2) configuration

Case3.mat = 4 km with green light - coupled cavity (M3-M4) configuration

Case4.mat = 4 km with green light - recombined configuration

Case5.mat = 40-meter with green light - recycled configuration

Case6.mat = 1100-meter with green light - coupled cavity (M1-M2) configuration

Case7.mat = 1100-meter with green light - coupled cavity (M3-M4) configuration

Case8.mat = 40-meter with green light - recombined configuration

Case9.mat = 4 km with IR light - recycled configuration

Appendix 1

Variable Definitions

This appendix lists the SMAC output variables. Users may need this information if the user decides to generate their own plots.

1.1 Transfer Function Variables

FoscVec = the vector of frequencies where the transfer functions are evaluated

The Transfer Function Outputs have the following naming convention, and the names are built up as follows.

The beginning of the name matches the TF gain to the detector as shown below:

g1 = detector 1 - sideband 1

g2 = detector 3 - sideband 1

g3 = detector 2 - sideband 1

gC1 = detector 4 - sideband 1

gF1 = detector 5 - sideband 1

g6_1 = detector 6 - sideband 1

g7_1 = detector 7 - sideband 1

g8_1 = detector 8 - sideband 1

g9_1 = detector 9 - sideband 1

g10_1 = detector 10 - sideband 1
g4 = detector 1 - sideband 2
g6 = detector 3 - sideband 2
g5 = detector 2 - sideband 2
gC2 = detector 4 - sideband 2
gF2 = detector 5 - sideband 2
g6_2 = detector 6 - sideband 2
g7_2 = detector 7 - sideband 2
g8_2 = detector 8 - sideband 2
g9_2 = detector 9 - sideband 2
g10_2 = detector 10 - sideband 2

The next part of the name is

d = demodulated
q = quadrature

The next part of the name is the degree of freedom which is being actuated

1 = Recycling Mirror
2 = Mirror 1
3 = Mirror 2
4 = Mirror 3
5 = Mirror 4
6 = Beam Splitter
7 = Source
8 = M2+M4
9 = M2-M4
10 = -(M1+M2)+(M3+M4)
11 = (M1+M2)+(M3+M4)

Finally, if the mode being run is Mixed Mode, nothing is added to the name, if Dynamics Mode is being run, a capital D is appended.

EXAMPLES

g1q7D=Det. 1, SB. 1, Quadrature, Source Actuation, Dynamic Mode

g6_1d2=Det. 6, SB. 1, InPhase, Mirror 1 Actuation, Mixed Mode

gF2d6=Det. 5, SB. 2, InPhase, Beam Splitter Actuator, Mixed Mode

1.2 Simulation Variables

dtVec = time vector, Mixed Mode or Static Mode
dtVecD = time vector, Dynamic Mode

DISPLACEMENTS AND VELOCITIES:

dzVecA = RM Displacement, Mixed Mode or Static Mode
dzVecAD = RM Displacement, Dynamic Mode
dzVecB = M1 Displacement, Mixed Mode or Static Mode
dzVecBD = M1 Displacement, Dynamic Mode
dzVecC = M2 Displacement, Mixed Mode or Static Mode
dzVecCD = M2 Displacement, Dynamic Mode
dzVecE = M3 Displacement, Mixed Mode or Static Mode
dzVecED = M3 Displacement, Dynamic Mode
dzVecF = M4 Displacement, Mixed Mode or Static Mode
dzVecFD = M4 Displacement, Dynamic Mode
dzVec2 = BS Displacement, Mixed Mode or Static Mode
dzVec2D = BS Displacement, Dynamic Mode
dzVecS = Source Phase, Mixed Mode or Static Mode
dzVecSD = Source Phase, Dynamic Mode

dvVecA = RM Velocity, Mixed Mode or Static Mode
dvVecAD = RM Velocity, Dynamic Mode
dvVecB = M1 Velocity, Mixed Mode or Static Mode

dvVecBD = M1 Velocity, Dynamic Mode
dvVecC = M2 Velocity, Mixed Mode or Static Mode
dvVecCD = M2 Velocity, Dynamic Mode
dvVecE = M3 Velocity, Mixed Mode or Static Mode
dvVecED = M3 Velocity, Dynamic Mode
dvVecF = M4 Velocity, Mixed Mode or Static Mode
dvVecFD = M4 Velocity, Dynamic Mode
dvVec2 = BS Velocity, Mixed Mode or Static Mode
dvVec2D = BS Velocity, Dynamic Mode

CONTROL FORCES:

fVecA = RM control force, Mixed Mode or Static Mode
fVecAD = RM control force, Dynamic Mode
fVecB = M1 control force, Mixed Mode or Static Mode
fVecBD = M1 control force, Dynamic Mode
fVecC = M2 control force, Mixed Mode or Static Mode
fVecCD = M2 control force, Dynamic Mode
fVecE = M3 control force, Mixed Mode or Static Mode
fVecED = M3 control force, Dynamic Mode
fVecF = M4 control force, Mixed Mode or Static Mode
fVecFD = M4 control force, Dynamic Mode
fVec2 = BS control force, Mixed Mode or Static Mode
fVec2D = BS control force, Dynamic Mode
fVecS = Source control force, Mixed Mode or Static Mode
fVecSD = Source control force, Dynamic Mode

CAVITY CIRCULATION INTENSITIES:

IABnd = RM to BS circulation intensity, carrier, static mode
IAB = RM to BS circulation intensity, carrier, mixed mode
IABD = RM to BS circulation intensity, carrier, dynamic mode

IBCnd = M1 to M2 circulation intensity, carrier, static mode
IBC = M1 to M2 circulation intensity, carrier, mixed mode
IBCD = M1 to M2 circulation intensity, carrier, dynamic mode
IEFnd = M3 to M4 circulation intensity, carrier, static mode
IEF = M3 to M4 circulation intensity, carrier, mixed mode
IEFD = M3 to M4 circulation intensity, carrier, dynamic mode

IABsbnd = RM to BS circulation intensity, sidebands, static mode
IABsb = RM to BS circulation intensity, sidebands, mixed mode
IABsbD = RM to BS circulation intensity, sidebands, dynamic mode
IBCsbnd = M1 to M2 circulation intensity, sidebands, static mode
IBCsb = M1 to M2 circulation intensity, sidebands, mixed mode
IBCsbD = M1 to M2 circulation intensity, sidebands, dynamic mode
IEFsbnd = M3 to M4 circulation intensity, sidebands, static mode
IEFsb = M3 to M4 circulation intensity, sidebands, mixed mode
IEFsbD = M3 to M4 circulation intensity, sidebands, dynamic mode

DETECTOR INTENSITIES:

ID1nd = Det. 1 intensity, carrier, static mode
ID1 = Det. 1 intensity, carrier, mixed mode
ID1D = Det. 1 intensity, carrier, dynamic mode
ID3nd = Det. 2 intensity, carrier, static mode
ID3 = Det. 2 intensity, carrier, mixed mode
ID3D = Det. 2 intensity, carrier, dynamic mode
ID2nd = Det. 3 intensity, carrier, static mode
ID2 = Det. 3 intensity, carrier, mixed mode
ID2D = Det. 3 intensity, carrier, dynamic mode
IDC1nd = Det. 4 intensity, carrier, static mode
IDC1 = Det. 4 intensity, carrier, mixed mode
IDC1D = Det. 4 intensity, carrier, dynamic mode
IDF1nd = Det. 5 intensity, carrier, static mode

IDF1 = Det. 5 intensity, carrier, mixed mode

IDF1D = Det. 5 intensity, carrier, dynamic mode

ID6nd = Det. 6 intensity, carrier, static mode

ID6 = Det. 6 intensity, carrier, mixed mode

ID6D = Det. 6 intensity, carrier, dynamic mode

ID7nd = Det. 7 intensity, carrier, static mode

ID7 = Det. 7 intensity, carrier, mixed mode

ID7D = Det. 7 intensity, carrier, dynamic mode

ID8nd = Det. 8 intensity, carrier, static mode

ID8 = Det. 8 intensity, carrier, mixed mode

ID8D = Det. 8 intensity, carrier, dynamic mode

ID9nd = Det. 9 intensity, carrier, static mode

ID9 = Det. 9 intensity, carrier, mixed mode

ID9D = Det. 9 intensity, carrier, dynamic mode

ID10nd = Det. 10 intensity, carrier, static mode

ID10 = Det. 10 intensity, carrier, mixed mode

ID10D = Det. 10 intensity, carrier, dynamic mode

Id1sbnd = Det. 1 intensity, SB 1, static mode

Id1sb = Det. 1 intensity, SB 1, mixed mode

Id1sbD = Det. 1 intensity, SB 1, dynamic mode

Id3sbnd = Det. 2 intensity, SB 1, static mode

Id3sb = Det. 2 intensity, SB 1, mixed mode

Id3sbD = Det. 2 intensity, SB 1, dynamic mode

Id2sbnd = Det. 3 intensity, SB 1, static mode

Id2sb = Det. 3 intensity, SB 1, mixed mode

Id2sbD = Det. 3 intensity, SB 1, dynamic mode

IC1sbnd = Det. 4 intensity, SB 1, static mode

IC1sb = Det. 4 intensity, SB 1, mixed mode

IC1sbD = Det. 4 intensity, SB 1, dynamic mode

IF1sbnd = Det. 5 intensity, SB 1, static mode

IF1sb = Det. 5 intensity, SB 1, mixed mode

IF1sbD = Det. 5 intensity, SB 1, dynamic mode
Id6_1sbnd = Det. 6 intensity, SB 1, static mode
Id6_1sb = Det. 6 intensity, SB 1, mixed mode
Id6_1sbD = Det. 6 intensity, SB 1, dynamic mode
Id7_1sbnd = Det. 7 intensity, SB 1, static mode
Id7_1sb = Det. 7 intensity, SB 1, mixed mode
Id7_1sbD = Det. 7 intensity, SB 1, dynamic mode
Id8_1sbnd = Det. 8 intensity, SB 1, static mode
Id8_1sb = Det. 8 intensity, SB 1, mixed mode
Id8_1sbD = Det. 8 intensity, SB 1, dynamic mode
Id9_1sbnd = Det. 9 intensity, SB 1, static mode
Id9_1sb = Det. 9 intensity, SB 1, mixed mode
Id9_1sbD = Det. 9 intensity, SB 1, dynamic mode
Id10_1sbnd = Det. 10 intensity, SB 1, static mode
Id10_1sb = Det. 10 intensity, SB 1, mixed mode
Id10_1sbD = Det. 10 intensity, SB 1, dynamic mode

Id4sbnd = Det. 1 intensity, SB 2, static mode
Id4sb = Det. 1 intensity, SB 2, mixed mode
Id4sbD = Det. 1 intensity, SB 2, dynamic mode
Id6sbnd = Det. 2 intensity, SB 2, static mode
Id6sb = Det. 2 intensity, SB 2, mixed mode
Id6sbD = Det. 2 intensity, SB 2, dynamic mode
Id5sbnd = Det. 3 intensity, SB 2, static mode
Id5sb = Det. 3 intensity, SB 2, mixed mode
Id5sbD = Det. 3 intensity, SB 2, dynamic mode
IC2sbnd = Det. 4 intensity, SB 2, static mode
IC2sb = Det. 4 intensity, SB 2, mixed mode
IC2sbD = Det. 4 intensity, SB 2, dynamic mode
IF2sbnd = Det. 5 intensity, SB 2, static mode
IF2sb = Det. 5 intensity, SB 2, mixed mode
IF2sbD = Det. 5 intensity, SB 2, dynamic mode

Id6_2sbnd = Det. 6 intensity, SB 2, static mode
Id6_2sb = Det. 6 intensity, SB 2, mixed mode
Id6_2sbD = Det. 6 intensity, SB 2, dynamic mode
Id7_2sbnd = Det. 7 intensity, SB 2, static mode
Id7_2sb = Det. 7 intensity, SB 2, mixed mode
Id7_2sbD = Det. 7 intensity, SB 2, dynamic mode
Id8_2sbnd = Det. 8 intensity, SB 2, static mode
Id8_2sb = Det. 8 intensity, SB 2, mixed mode
Id8_2sbD = Det. 8 intensity, SB 2, dynamic mode
Id9_2sbnd = Det. 9 intensity, SB 2, static mode
Id9_2sb = Det. 9 intensity, SB 2, mixed mode
Id9_2sbD = Det. 9 intensity, SB 2, dynamic mode
Id10_2sbnd = Det. 10 intensity, SB 2, static mode
Id10_2sb = Det. 10 intensity, SB 2, mixed mode
Id10_2sbD = Det. 10 intensity, SB 2, dynamic mode

SENSOR OUTPUT:

Idemod1nd = Det. 1 InPhase Output, SB. 1, static mode
Idemod1 = Det. 1 InPhase Output, SB. 1, mixed mode
Idemod1D = Det. 1 InPhase Output, SB. 1, dynamic mode
Iquad1nd = Det. 1 Quadrature Output, SB. 1, static mode
Iquad1 = Det. 1 Quadrature Output, SB. 1, mixed mode
Iquad1D = Det. 1 Quadrature Output, SB. 1, dynamic mode
Idemod3nd = Det. 2 InPhase Output, SB. 1, static mode
Idemod3 = Det. 2 InPhase Output, SB. 1, mixed mode
Idemod3D = Det. 2 InPhase Output, SB. 1, dynamic mode
Iquad3nd = Det. 2 Quadrature Output, SB. 1, static mode
Iquad3 = Det. 2 Quadrature Output, SB. 1, mixed mode
Iquad3D = Det. 2 Quadrature Output, SB. 1, dynamic mode
Idemod2nd = Det. 3 InPhase Output, SB. 1, static mode
Idemod2 = Det. 3 InPhase Output, SB. 1, mixed mode
Idemod2D = Det. 3 InPhase Output, SB. 1, dynamic mode

Iquad2nd = Det. 3 Quadrature Output, SB. 1, static mode
Iquad2 = Det. 3 Quadrature Output, SB. 1, mixed mode
Iquad2D = Det. 3 Quadrature Output, SB. 1, dynamic mode
IdemodC1nd = Det. 4 InPhase Output, SB. 1, static mode
IdemodC1 = Det. 4 InPhase Output, SB. 1, mixed mode
IdemodC1D = Det. 4 InPhase Output, SB. 1, dynamic mode
IquadC1nd = Det. 4 Quadrature Output, SB. 1, static mode
IquadC1 = Det. 4 Quadrature Output, SB. 1, mixed mode
IquadC1D = Det. 4 Quadrature Output, SB. 1, dynamic mode
IdemodF1nd = Det. 5 InPhase Output, SB. 1, static mode
IdemodF1 = Det. 5 InPhase Output, SB. 1, mixed mode
IdemodF1D = Det. 5 InPhase Output, SB. 1, dynamic mode
IquadF1nd = Det. 5 Quadrature Output, SB. 1, static mode
IquadF1 = Det. 5 Quadrature Output, SB. 1, mixed mode
IquadF1D = Det. 5 Quadrature Output, SB. 1, dynamic mode
Idemod6_1nd = Det. 6 InPhase Output, SB. 1, static mode
Idemod6_1 = Det. 6 InPhase Output, SB. 1, mixed mode
Idemod6_1D = Det. 6 InPhase Output, SB. 1, dynamic mode
Iquad6_1nd = Det. 6 Quadrature Output, SB. 1, static mode
Iquad6_1 = Det. 6 Quadrature Output, SB. 1, mixed mode
Iquad6_1D = Det. 6 Quadrature Output, SB. 1, dynamic mode
Idemod7_1nd = Det. 7 InPhase Output, SB. 1, static mode
Idemod7_1 = Det. 7 InPhase Output, SB. 1, mixed mode
Idemod7_1D = Det. 7 InPhase Output, SB. 1, dynamic mode
Iquad7_1nd = Det. 7 Quadrature Output, SB. 1, static mode
Iquad7_1 = Det. 7 Quadrature Output, SB. 1, mixed mode
Iquad7_1D = Det. 7 Quadrature Output, SB. 1, dynamic mode
Idemod8_1nd = Det. 8 InPhase Output, SB. 1, static mode
Idemod8_1 = Det. 8 InPhase Output, SB. 1, mixed mode
Idemod8_1D = Det. 8 InPhase Output, SB. 1, dynamic mode
Iquad8_1nd = Det. 8 Quadrature Output, SB. 1, static mode
Iquad8_1 = Det. 8 Quadrature Output, SB. 1, mixed mode
Iquad8_1D = Det. 8 Quadrature Output, SB. 1, dynamic mode

Idemod9_1nd = Det. 9 InPhase Output, SB. 1, static mode

Idemod9_1 = Det. 9 InPhase Output, SB. 1, mixed mode

Idemod9_1D = Det. 9 InPhase Output, SB. 1, dynamic mode

Iquad9_1nd = Det. 9 Quadrature Output, SB. 1, static mode

Iquad9_1 = Det. 9 Quadrature Output, SB. 1, mixed mode

Iquad9_1D = Det. 9 Quadrature Output, SB. 1, dynamic mode

Idemod10_1nd = Det. 10 InPhase Output, SB. 1, static mode

Idemod10_1 = Det. 10 InPhase Output, SB. 1, mixed mode

Idemod10_1D = Det. 10 InPhase Output, SB. 1, dynamic mode

Iquad10_1nd = Det. 10 Quadrature Output, SB. 1, static mode

Iquad10_1 = Det. 10 Quadrature Output, SB. 1, mixed mode

Iquad10_1D = Det. 10 Quadrature Output, SB. 1, dynamic mode

Idemod4nd = Det. 1 InPhase Output, SB. 2, static mode

Idemod4 = Det. 1 InPhase Output, SB. 2, mixed mode

Idemod4D = Det. 1 InPhase Output, SB. 2, dynamic mode

Iquad4nd = Det. 1 Quadrature Output, SB. 2, static mode

Iquad4 = Det. 1 Quadrature Output, SB. 2, mixed mode

Iquad4D = Det. 1 Quadrature Output, SB. 2, dynamic mode

Idemod6nd = Det. 2 InPhase Output, SB. 2, static mode

Idemod6 = Det. 2 InPhase Output, SB. 2, mixed mode

Idemod6D = Det. 2 InPhase Output, SB. 2, dynamic mode

Iquad6nd = Det. 2 Quadrature Output, SB. 2, static mode

Iquad6 = Det. 2 Quadrature Output, SB. 2, mixed mode

Iquad6D = Det. 2 Quadrature Output, SB. 2, dynamic mode

Idemod5nd = Det. 3 InPhase Output, SB. 2, static mode

Idemod5 = Det. 3 InPhase Output, SB. 2, mixed mode

Idemod5D = Det. 3 InPhase Output, SB. 2, dynamic mode

Iquad5nd = Det. 3 Quadrature Output, SB. 2, static mode

Iquad5 = Det. 3 Quadrature Output, SB. 2, mixed mode

Iquad5D = Det. 3 Quadrature Output, SB. 2, dynamic mode

IdemodC2nd = Det. 4 InPhase Output, SB. 2, static mode

IdemodC2 = Det. 4 InPhase Output, SB. 2, mixed mode

IdemodC2D = Det. 4 InPhase Output, SB. 2, dynamic mode

IquadC2nd = Det. 4 Quadrature Output, SB. 2, static mode

IquadC2 = Det. 4 Quadrature Output, SB. 2, mixed mode

IquadC2D = Det. 4 Quadrature Output, SB. 2, dynamic mode

IdemodF2nd = Det. 5 InPhase Output, SB. 2, static mode

IdemodF2 = Det. 5 InPhase Output, SB. 2, mixed mode

IdemodF2D = Det. 5 InPhase Output, SB. 2, dynamic mode

IquadF2nd = Det. 5 Quadrature Output, SB. 2, static mode

IquadF2 = Det. 5 Quadrature Output, SB. 2, mixed mode

IquadF2D = Det. 5 Quadrature Output, SB. 2, dynamic mode

Idemod6_2nd = Det. 6 InPhase Output, SB. 2, static mode

Idemod6_2 = Det. 6 InPhase Output, SB. 2, mixed mode

Idemod6_2D = Det. 6 InPhase Output, SB. 2, dynamic mode

Iquad6_2nd = Det. 6 Quadrature Output, SB. 2, static mode

Iquad6_2 = Det. 6 Quadrature Output, SB. 2, mixed mode

Iquad6_2D = Det. 6 Quadrature Output, SB. 2, dynamic mode

Idemod7_2nd = Det. 7 InPhase Output, SB. 2, static mode

Idemod7_2 = Det. 7 InPhase Output, SB. 2, mixed mode

Idemod7_2D = Det. 7 InPhase Output, SB. 2, dynamic mode

Iquad7_2nd = Det. 7 Quadrature Output, SB. 2, static mode

Iquad7_2 = Det. 7 Quadrature Output, SB. 2, mixed mode

Iquad7_2D = Det. 7 Quadrature Output, SB. 2, dynamic mode

Idemod8_2nd = Det. 8 InPhase Output, SB. 2, static mode

Idemod8_2 = Det. 8 InPhase Output, SB. 2, mixed mode

Idemod8_2D = Det. 8 InPhase Output, SB. 2, dynamic mode

Iquad8_2nd = Det. 8 Quadrature Output, SB. 2, static mode

Iquad8_2 = Det. 8 Quadrature Output, SB. 2, mixed mode

Iquad8_2D = Det. 8 Quadrature Output, SB. 2, dynamic mode

Idemod9_2nd = Det. 9 InPhase Output, SB. 2, static mode

Idemod9_2 = Det. 9 InPhase Output, SB. 2, mixed mode

Idemod9_2D = Det. 9 InPhase Output, SB. 2, dynamic mode

Iquad9_2nd = Det. 9 Quadrature Output, SB. 2, static mode

Iquad9_2	= Det. 9 Quadrature Output, SB. 2, mixed mode
Iquad9_2D	= Det. 9 Quadrature Output, SB. 2, dynamic mode
Idemod10_2nd	= Det. 10 InPhase Output, SB. 2, static mode
Idemod10_2	= Det. 10 InPhase Output, SB. 2, mixed mode
Idemod10_2D	= Det. 10 InPhase Output, SB. 2, dynamic mode
Iquad10_2nd	= Det. 10 Quadrature Output, SB. 2, static mode
Iquad10_2	= Det. 10 Quadrature Output, SB. 2, mixed mode
Iquad10_2D	= Det. 10 Quadrature Output, SB. 2, dynamic mode

Appendix 2

Nonlinear Control

This appendix offers a detailed description of each of the pieces of a Fortran nonlinear control routine which can be used in SMAC. This nonlinear controller capability offers the user great freedom in the type of controller which can be used. However, this freedom means that the implementation of the controller may be considered by many users to be complicated.

There are several pieces of the code which are required, regardless of how simple or complicated the controller is. These required pieces will be discussed first. Then, several examples will be explained which will give insight into how powerful a tool this capability is.

The nonlinear control subroutine is similar to any other subroutine, and is highlighted below:

- Subroutine Title: there is a strict naming convention which must be followed for the subroutine title. The name of the file containing the subroutine can be named anything.f
- Subroutine Inputs: 7 variables are passed to the subroutine. There are six "inputs" and one "output". The user must include all 7 variables in the SUBROUTINE statement, and the user must calculate the output, but the user is not obligated to use any of the 6 "input" variables when calculating the "output" variable

- Variable Declarations - this section is used to declare the variable types. Some declarations are required and must be included as shown below. The user must also make the declarations needed for their particular controller
- Output Calculations - This is the section where the user implements the body of code which represents the nonlinear controller. Several examples of types of nonlinear controllers will be presented in the examples which follow

2.1 The Subroutine Title

```
SUBROUTINE ryExeControl2(fVec,i2f,S,x,n,v,simTime)
```

As explained in the section beginning on page 69, the name of the Fortran file can be anything, but, the name of the subroutine must be ryExeControl1, ryExeControl2,..., ryExeControl10 depending on the location on the GUI of this controller. (See Figure 36).

2.2 The Subroutine Inputs

There are 7 variables which are passed to the subroutine, they are

```
C*****************************************************
C This subroutine passes in 7 variables:
C
C  fVec = [
C     force applied to RM in kg*m/sec^2;
C     force applied to M1 in kg*m/sec^2;
C     force applied to M2 in kg*m/sec^2;
C     force applied to M3 in kg*m/sec^2;
C     force applied to M4 in kg*m/sec^2;
C     force applied to BS in kg*m/sec^2;
C     equivalent Source displacement in units of m]
C
C  i2f = the actuator(s) for this control calculation
C    1 = Recycling Mirror
C    2 = Mirror 1
C    3 = Mirror 2
C    4 = Mirror 3
```

ler which was entered on the GUI. The user does not need to use this value, and may use more than one "sensor input", as will be demonstrated in examples below. simTime is the REAL*8 scalar which is the value of time that the simulation is using at this step in the program.

2.3 Variable Declarations

The following are variable declarations which must be included in the subroutine. This portion of the code must be included exactly as shown here. The variables are passed to the controller subroutine through the COMMON BLOCK SIGNALS which must be included in the code. The variables included in the common block below are the variables which can be used in the nonlinear control functions. See Appendix 3 for definitions of these variables.

```
IMPLICIT NONE
INTEGER i2f,n
REAL*8 fVec(7),f,S(n,n),x(n),v,simTime

REAL*8
   & Idemod1_f1,Iquad1_f1,Idemod2_f1,Iquad2_f1,
   & Idemod3_f1,Iquad3_f1,Idemod4_f1,Iquad4_f1,
   & Idemod5_f1,Iquad5_f1,Idemod6_f1,Iquad6_f1,
   & Idemod7_f1,Iquad7_f1,Idemod8_f1,Iquad8_f1,
   & Idemod9_f1,Iquad9_f1,Idemod10_f1,Iquad10_f1,
   & Idemod1_f2,Iquad1_f2,Idemod2_f2,Iquad2_f2,
   & Idemod3_f2,Iquad3_f2,Idemod4_f2,Iquad4_f2,
   & Idemod5_f2,Iquad5_f2,Idemod6_f2,Iquad6_f2,
   & Idemod7_f2,Iquad7_f2,Idemod8_f2,Iquad8_f2,
   & Idemod9_f2,Iquad9_f2,Idemod10_f2,Iquad10_f2,
   & I_D1,I_D2,I_D3,I_D4,I_D5,
   & I_D6,I_D7,I_D8,I_D9,I_D10,
   & I_D1_sb1,I_D2_sb1,I_D3_sb1,I_D4_sb1,I_D5_sb1,
   & I_D6_sb1,I_D7_sb1,I_D8_sb1,I_D9_sb1,I_D10_sb1,
   & I_D1_sb2,I_D2_sb2,I_D3_sb2,I_D4_sb2,I_D5_sb2,
   & I_D6_sb2,I_D7_sb2,I_D8_sb2,I_D9_sb2,I_D10_sb2
```

```
 COMMON /SIGNALS/
& Idemod1_f1,Iquad1_f1,Idemod2_f1,Iquad2_f1,
& Idemod3_f1,Iquad3_f1,Idemod4_f1,Iquad4_f1,
& Idemod5_f1,Iquad5_f1,Idemod6_f1,Iquad6_f1,
& Idemod7_f1,Iquad7_f1,Idemod8_f1,Iquad8_f1,
& Idemod9_f1,Iquad9_f1,Idemod10_f1,Iquad10_f1,
& Idemod1_f2,Iquad1_f2,Idemod2_f2,Iquad2_f2,
& Idemod3_f2,Iquad3_f2,Idemod4_f2,Iquad4_f2,
& Idemod5_f2,Iquad5_f2,Idemod6_f2,Iquad6_f2,
& Idemod7_f2,Iquad7_f2,Idemod8_f2,Iquad8_f2,
& Idemod9_f2,Iquad9_f2,Idemod10_f2,Iquad10_f2,
& I_D1,I_D2,I_D3,I_D4,I_D5,
& I_D6,I_D7,I_D8,I_D9,I_D10,
& I_D1_sb1,I_D2_sb1,I_D3_sb1,I_D4_sb1,I_D5_sb1,
& I_D6_sb1,I_D7_sb1,I_D8_sb1,I_D9_sb1,I_D10_sb1,
& I_D1_sb2,I_D2_sb2,I_D3_sb2,I_D4_sb2,I_D5_sb2,
& I_D6_sb2,I_D7_sb2,I_D8_sb2,I_D9_sb2,I_D10_sb2
```

2.4 The Force Calculations

The final piece of code that needs to be included is the force/distance calculations (the output of the controller). Several examples are included below which demonstrate different possible force calculations.

2.5 Examples

Several different examples of controllers are shown below. The entire subroutine is included in each example. The sections of code which are shown in bold font are the sections unique to each example. The sections or code shown in normal font are the sections which are required pieces of the subroutine which were discussed above. Descriptive information is included in the standard font.

2.5.1 Linear Controller

The first example implements the linear controller which is normally used by SMAC. This is intended to clarify any residual questions about the implementation of controllers without complications of difficult control schemes.

The linear controller implements the standard

xdot = A*x + B*u

y = C*x + D*u

```
      SUBROUTINE ryExeControl2(fVec,i2f,S,x,n,v,simTime)

      IMPLICIT NONE
      INTEGER i2f,n
      REAL*8 fVec(7),f,S(n,n),x(n),v,simTime
```

INTEGER i,j
REAL*8 xn(n),f

```
      REAL*8
     & Idemod1_f1,Iquad1_f1,Idemod2_f1,Iquad2_f1,
     & Idemod3_f1,Iquad3_f1,Idemod4_f1,Iquad4_f1,
     & Idemod5_f1,Iquad5_f1,Idemod6_f1,Iquad6_f1,
     & Idemod7_f1,Iquad7_f1,Idemod8_f1,Iquad8_f1,
     & Idemod9_f1,Iquad9_f1,Idemod10_f1,Iquad10_f1,
     & Idemod1_f2,Iquad1_f2,Idemod2_f2,Iquad2_f2,
     & Idemod3_f2,Iquad3_f2,Idemod4_f2,Iquad4_f2,
     & Idemod5_f2,Iquad5_f2,Idemod6_f2,Iquad6_f2,
     & Idemod7_f2,Iquad7_f2,Idemod8_f2,Iquad8_f2,
     & Idemod9_f2,Iquad9_f2,Idemod10_f2,Iquad10_f2,
     & I_D1,I_D2,I_D3,I_D4,I_D5,
     & I_D6,I_D7,I_D8,I_D9,I_D10,
     & I_D1_sb1,I_D2_sb1,I_D3_sb1,I_D4_sb1,I_D5_sb1,
     & I_D6_sb1,I_D7_sb1,I_D8_sb1,I_D9_sb1,I_D10_sb1,
     & I_D1_sb2,I_D2_sb2,I_D3_sb2,I_D4_sb2,I_D5_sb2,
     & I_D6_sb2,I_D7_sb2,I_D8_sb2,I_D9_sb2,I_D10_sb2


      COMMON /SIGNALS/
```

```
& Idemod1_f1,Iquad1_f1,Idemod2_f1,Iquad2_f1,
& Idemod3_f1,Iquad3_f1,Idemod4_f1,Iquad4_f1,
& Idemod5_f1,Iquad5_f1,Idemod6_f1,Iquad6_f1,
& Idemod7_f1,Iquad7_f1,Idemod8_f1,Iquad8_f1,
& Idemod9_f1,Iquad9_f1,Idemod10_f1,Iquad10_f1,
& Idemod1_f2,Iquad1_f2,Idemod2_f2,Iquad2_f2,
& Idemod3_f2,Iquad3_f2,Idemod4_f2,Iquad4_f2,
& Idemod5_f2,Iquad5_f2,Idemod6_f2,Iquad6_f2,
& Idemod7_f2,Iquad7_f2,Idemod8_f2,Iquad8_f2,
& Idemod9_f2,Iquad9_f2,Idemod10_f2,Iquad10_f2,
& I_D1,I_D2,I_D3,I_D4,I_D5,
& I_D6,I_D7,I_D8,I_D9,I_D10,
& I_D1_sb1,I_D2_sb1,I_D3_sb1,I_D4_sb1,I_D5_sb1,
& I_D6_sb1,I_D7_sb1,I_D8_sb1,I_D9_sb1,I_D10_sb1,
& I_D1_sb2,I_D2_sb2,I_D3_sb2,I_D4_sb2,I_D5_sb2,
& I_D6_sb2,I_D7_sb2,I_D8_sb2,I_D9_sb2,I_D10_sb2
```

The next step implements the "xdot=a*x" part of the controller. Remember that "a" is the (n-1)x(n-1) subset of the S matrix.

```
C  xn=a*x
     DO i=1,n-1
       xn(i)=0d0
       DO j=1,n-1
         xn(i)=xn(i)+S(i,j)*x(j)
       END DO
     END DO
```

Now add on the "+b*u" part of "xdot = a*x + b*u"

```
C  x=xn+b*I
     DO i=1,n-1
       x(i)=xn(i)+S(i,n)*v
     END DO
```

Finally, calculate the output force, y = c *x + d*u

```
C  f=c*x+d*I
```

```
      f=S(n,n)*v
      DO j=1,n-1
       f=f+S(n,j)*x(j)
      END DO
```

Now, put the calculated output force in the correct locations in the output vector fVec. The rows of fVec are defined in Section 2.2 of this Appendix. i2f specifies which actuator set is to be used. It is important to remember to add the old value of fVec to the new one because multiple controllers may be used and control values may already exist in fVec.

```
      IF (i2f.GE.1.AND.i2f.LE.7) THEN
       fVec(i2f)=f+fVec(i2f)
      ELSE IF (i2f.EQ.8) THEN
C M2+M4
       fVec(3)=f+fVec(3)
       fVec(5)=f+fVec(5)
      ELSE IF (i2f.EQ.9) THEN
C M2-M4
       fVec(3)=f+fVec(3)
       fVec(5)=-f+fVec(5)
      ELSE IF (i2f.EQ.10) THEN
C -(M1+M2)+(M3+M4)
       fVec(2)=-f+fVec(2)
       fVec(3)=-f+fVec(3)
       fVec(4)=f+fVec(4)
       fVec(5)=f+fVec(5)
      ELSE IF (i2f.EQ.11) THEN
C (M1+M2)+(M3+M4)
       fVec(2)=f+fVec(2)
       fVec(3)=f+fVec(3)
       fVec(4)=f+fVec(4)
       fVec(5)=f+fVec(5)
      END IF

      RETURN
```

```
END
```

2.5.2 Controller With Sensor Saturation

This example implements a simple linear controller which is normally used by SMAC, with a nonlinear addition of sensor saturation. The sensor saturation value in this example is 1e-6. The sensor/actuator values entered in the GUI are preserved for this example. The actuator is the RM, and the sensed value is the Detector 1 In-Phase signal.

```
    SUBROUTINE ryExeControl2(fVec,i2f,S,x,n,v,simTime)

    IMPLICIT NONE
    INTEGER i2f,n
    REAL*8 fVec(7),f,S(n,n),x(n),v,simTime

    INTEGER i,j
    REAL*8 xn(n),vtmp, satvalue

    REAL*8
  &  Idemod1_f1,Iquad1_f1,Idemod2_f1,Iquad2_f1,
  &  Idemod3_f1,Iquad3_f1,Idemod4_f1,Iquad4_f1,
  &  Idemod5_f1,Iquad5_f1,Idemod6_f1,Iquad6_f1,
  &  Idemod7_f1,Iquad7_f1,Idemod8_f1,Iquad8_f1,
  &  Idemod9_f1,Iquad9_f1,Idemod10_f1,Iquad10_f1,
  &  Idemod1_f2,Iquad1_f2,Idemod2_f2,Iquad2_f2,
  &  Idemod3_f2,Iquad3_f2,Idemod4_f2,Iquad4_f2,
  &  Idemod5_f2,Iquad5_f2,Idemod6_f2,Iquad6_f2,
  &  Idemod7_f2,Iquad7_f2,Idemod8_f2,Iquad8_f2,
  &  Idemod9_f2,Iquad9_f2,Idemod10_f2,Iquad10_f2,
  &  I_D1,I_D2,I_D3,I_D4,I_D5,
  &  I_D6,I_D7,I_D8,I_D9,I_D10,
  &  I_D1_sb1,I_D2_sb1,I_D3_sb1,I_D4_sb1,I_D5_sb1,
  &  I_D6_sb1,I_D7_sb1,I_D8_sb1,I_D9_sb1,I_D10_sb1,
  &  I_D1_sb2,I_D2_sb2,I_D3_sb2,I_D4_sb2,I_D5_sb2,
     & I_D6_sb2,I_D7_sb2,I_D8_sb2,I_D9_sb2,I_D10_sb2


      COMMON /SIGNALS/
     & Idemod1_f1,Iquad1_f1,Idemod2_f1,Iquad2_f1,
     & Idemod3_f1,Iquad3_f1,Idemod4_f1,Iquad4_f1,
     & Idemod5_f1,Iquad5_f1,Idemod6_f1,Iquad6_f1,
     & Idemod7_f1,Iquad7_f1,Idemod8_f1,Iquad8_f1,
     & Idemod9_f1,Iquad9_f1,Idemod10_f1,Iquad10_f1,
     & Idemod1_f2,Iquad1_f2,Idemod2_f2,Iquad2_f2,
     & Idemod3_f2,Iquad3_f2,Idemod4_f2,Iquad4_f2,
     & Idemod5_f2,Iquad5_f2,Idemod6_f2,Iquad6_f2,
     & Idemod7_f2,Iquad7_f2,Idemod8_f2,Iquad8_f2,
     & Idemod9_f2,Iquad9_f2,Idemod10_f2,Iquad10_f2,
     & I_D1,I_D2,I_D3,I_D4,I_D5,
     & I_D6,I_D7,I_D8,I_D9,I_D10,
     & I_D1_sb1,I_D2_sb1,I_D3_sb1,I_D4_sb1,I_D5_sb1,
     & I_D6_sb1,I_D7_sb1,I_D8_sb1,I_D9_sb1,I_D10_sb1,
     & I_D1_sb2,I_D2_sb2,I_D3_sb2,I_D4_sb2,I_D5_sb2,
     & I_D6_sb2,I_D7_sb2,I_D8_sb2,I_D9_sb2,I_D10_sb2
```

Declare the sensor's saturation value.

```
      satvalue=1e-6
```

Check to see if the sensor saturation limit is reached.

```
   C Check for saturation
      IF v .GT. satvalue
       vtmp=satvalue
      ELSE
       vtmp=v
      END

   C xn=a*x
      DO i=1,n-1
       xn(i)=0d0
```

```
        DO j=1,n-1
         xn(i)=xn(i)+S(i,j)*x(j)
        END DO
       END DO

C  x=xn+b*I
       DO i=1,n-1
        x(i)=xn(i)+S(i,n) * vtmp
       END DO
```

Finally, calculate the output force, y = c *x + d*u

```
C  f=c*x+d*I
       f=S(n,n)* vtmp
       DO j=1,n-1
        f=f+S(n,j)*x(j)
       END DO
```

Now, put the calculated output force in the correct locations in the output vector fVec. The rows of fVec are defined in Section 2.2. i2f specifies which actuator is to be used. It is important to remember to add the old value of fVec to the new one because multiple controllers may be used and control values may already exist in fVec.

```
       IF (i2f.GE.1.AND.i2f.LE.7) THEN
        fVec(i2f)=f+fVec(i2f)
       ELSE IF (i2f.EQ.8) THEN
C M2+M4
        fVec(3)=f+fVec(3)
        fVec(5)=f+fVec(5)
       ELSE IF (i2f.EQ.9) THEN
C M2-M4
        fVec(3)=f+fVec(3)
        fVec(5)=-f+fVec(5)
       ELSE IF (i2f.EQ.10) THEN
C -(M1+M2)+(M3+M4)
        fVec(2)=-f+fVec(2)
```

```
   fVec(3)=-f+fVec(3)
   fVec(4)=f+fVec(4)
   fVec(5)=f+fVec(5)
  ELSE IF (i2f.EQ.11) THEN
C (M1+M2)+(M3+M4)
   fVec(2)=f+fVec(2)
   fVec(3)=f+fVec(3)
   fVec(4)=f+fVec(4)
   fVec(5)=f+fVec(5)
  END IF

  RETURN
  END
```

2.5.3 Controller With Sign Flip On Actuator Signal

In this example, a simple linear controller is used, except a sign flip on the feedback signal occurs when the transmitted light on the photodetectors is above a certain threshold value.

The actuator for this controller is the beam splitter and was included in the GUI. The sensor signal for the controller was included in the GUI.

The transmitted light levels on photodetectors D4 & D5 are checked to see if they have both reached the threshold value of 800*12e-6. If both photodetectors are above this value, then the sign on the feedback signal is flipped.

This example demonstrates how it is possible to use sensor information which was not included in the GUI menu (the light on photodetectors D4 & D5). This example also demonstrates how it is possible to cater the controller specifically to an actuator.

```
  SUBROUTINE ryExeControl2(fVec,i2f,S,x,n,v,simTime)

  IMPLICIT NONE
  INTEGER i2f,n
  REAL*8 fVec(7),f,S(n,n),x(n),v,simTime
```

```
 INTEGER i,j
 REAL*8 xn(n), trig_val

 REAL*8
&  Idemod1_f1,Iquad1_f1,Idemod2_f1,Iquad2_f1,
&  Idemod3_f1,Iquad3_f1,Idemod4_f1,Iquad4_f1,
&  Idemod5_f1,Iquad5_f1,Idemod6_f1,Iquad6_f1,
&  Idemod7_f1,Iquad7_f1,Idemod8_f1,Iquad8_f1,
&  Idemod9_f1,Iquad9_f1,Idemod10_f1,Iquad10_f1,
&  Idemod1_f2,Iquad1_f2,Idemod2_f2,Iquad2_f2,
&  Idemod3_f2,Iquad3_f2,Idemod4_f2,Iquad4_f2,
&  Idemod5_f2,Iquad5_f2,Idemod6_f2,Iquad6_f2,
&  Idemod7_f2,Iquad7_f2,Idemod8_f2,Iquad8_f2,
&  Idemod9_f2,Iquad9_f2,Idemod10_f2,Iquad10_f2,
&  I_D1,I_D2,I_D3,I_D4,I_D5,
&  I_D6,I_D7,I_D8,I_D9,I_D10,
&  I_D1_sb1,I_D2_sb1,I_D3_sb1,I_D4_sb1,I_D5_sb1,
&  I_D6_sb1,I_D7_sb1,I_D8_sb1,I_D9_sb1,I_D10_sb1,
&  I_D1_sb2,I_D2_sb2,I_D3_sb2,I_D4_sb2,I_D5_sb2,
&  I_D6_sb2,I_D7_sb2,I_D8_sb2,I_D9_sb2,I_D10_sb2


 COMMON /SIGNALS/
&  Idemod1_f1,Iquad1_f1,Idemod2_f1,Iquad2_f1,
&  Idemod3_f1,Iquad3_f1,Idemod4_f1,Iquad4_f1,
&  Idemod5_f1,Iquad5_f1,Idemod6_f1,Iquad6_f1,
&  Idemod7_f1,Iquad7_f1,Idemod8_f1,Iquad8_f1,
&  Idemod9_f1,Iquad9_f1,Idemod10_f1,Iquad10_f1,
&  Idemod1_f2,Iquad1_f2,Idemod2_f2,Iquad2_f2,
&  Idemod3_f2,Iquad3_f2,Idemod4_f2,Iquad4_f2,
&  Idemod5_f2,Iquad5_f2,Idemod6_f2,Iquad6_f2,
&  Idemod7_f2,Iquad7_f2,Idemod8_f2,Iquad8_f2,
&  Idemod9_f2,Iquad9_f2,Idemod10_f2,Iquad10_f2,
&  I_D1,I_D2,I_D3,I_D4,I_D5,
& I_D6,I_D7,I_D8,I_D9,I_D10,
& I_D1_sb1,I_D2_sb1,I_D3_sb1,I_D4_sb1,I_D5_sb1,
& I_D6_sb1,I_D7_sb1,I_D8_sb1,I_D9_sb1,I_D10_sb1,
& I_D1_sb2,I_D2_sb2,I_D3_sb2,I_D4_sb2,I_D5_sb2,
& I_D6_sb2,I_D7_sb2,I_D8_sb2,I_D9_sb2,I_D10_sb2
```

Declare the trigger value for light levels

```
    trig_val=800*12e-6
```

```
C xn=a*x
    DO i=1,n-1
     xn(i)=0d0
     DO j=1,n-1
      xn(i)=xn(i)+S(i,j)*x(j)
     END DO
    END DO

C x=xn+b*I
    DO i=1,n-1
     x(i)=xn(i)+S(i,n)* v
    END DO
```

Finally, calculate the output force, y = c *x + d*u

```
C f=c*x+d*I
    f=S(n,n)* v
    DO j=1,n-1
     f=f+S(n,j)*x(j)
    END DO
```

We know that the actuator is the RM, so the control value will be placed directly in that location, instead of checking to see what actuator was included in the GUI. It is also here that we check to see if the light on the photodetectors has reached the trigger value.

```
    IF (I_D4 .GE. trig_val .AND. I_D5 .GE. trig_val) THEN
```

```
  fVec(1)=fVec(1)-f
ELSE
  fVec(1)=fVec(1)+f
END IF

RETURN
END
```

2.5.4 Controller Replicating Stuck Actuators

This example is included demonstrate the "power" of the nonlinear controller capability. This is an example of a stuck actuator. Regardless of what the sensor signal is, two actuators are "stuck" and output a constant value. This example is obviously not a controller a user would want to implement, but is included because it demonstrates the flexibility of the nonlinear controller capability.

```
SUBROUTINE ryExeControl2(fVec,i2f,S,x,n,v,simTime)
```

These variable type declarations below are required because the variables are passed into the subroutine even though none of this information will be used.

```
IMPLICIT NONE
INTEGER i2f,n
REAL*8 fVec(7),f,S(n,n),x(n),v,simTime
```

The user should notice that the COMMON BLOCK and declarations for the variables in the COMMON BLOCK are not included here. Since none of these variables are being used, there is no reason to include them in the subroutine.

Similarly, the user should notice that no calculations are included regarding the states of the system or the sensor signals. If the user wanted to implement a Multi-Input and/or Multi-Output Controller, the user would include the state space calculations here.

The stuck actuators are the RM and the BS actuators, so we will place the control value directly in that location, instead of checking to see what actuator was included in the GUI. Also notice that in this example, since the actuators are “stuck”, the control force is not added to the value that previously existed in fVec (fVec(1)=fVec(1)+0.5), but resets the value (fVec(1)=0.5) to the “stuck” value.

```
fVec(1)=0.5
fVec(6)=0.25
```

The user needs to understand a little bit more of how the controllers are implemented in SMAC to fully understand this example. In SMAC, the controllers are called in the order they are listed on the GUI. Since we know this was the second controller listed on the GUI (SUBROUTINE ryExeControl2), this example will accurately model stuck actuators ONLY if the last active controller on the GUI is the second one OR any subsequent active controllers on the GUI do not use the RM and BS actuators. If either of these conditions are not met, then it is possible that the RM and BS actuators may have additional force outputs added to or subtracted from the value the actuator is stuck at.

```
RETURN
END
```

Appendix 3

Control Variables

This appendix lists the SMAC control variables. Users may need this information if non-linear controllers are used.

3.1 Control Variables

DETECTOR INTENSITIES:

I_D1 = Det. 1 intensity, carrier

I_D3 = Det. 2 intensity, carrier

I_D2 = Det. 3 intensity, carrier

I_D4 = Det. 4 intensity, carrier

I_D5 = Det. 5 intensity, carrier

I_D6 = Det. 6 intensity, carrier

I_D7 = Det. 7 intensity, carrier

I_D8 = Det. 8 intensity, carrier

I_D9 = Det. 9 intensity, carrier

I_D10 = Det. 10 intensity, carrier

I_D1_sb1 = Det. 1 intensity, SB 1

I_D3_sb1 = Det. 2 intensity, SB 1

I_D2_sb1 = Det. 3 intensity, SB 1

I_D4_sb1 = Det. 4 intensity, SB 1

I_D5_sb1 = Det. 5 intensity, SB 1

I_D6_sb1 = Det. 6 intensity, SB 1

I_D7_sb1 = Det. 7 intensity, SB 1

I_D8_sb1 = Det. 8 intensity, SB 1

I_D9_sb1 = Det. 9 intensity, SB 1

I_D10_sb1 = Det. 10 intensity, SB 1

I_D1_sb2 = Det. 1 intensity, SB 2

I_D2_sb2 = Det. 2 intensity, SB 2

I_D3_sb2 = Det. 3 intensity, SB 2

I_D4_sb2 = Det. 4 intensity, SB 2

I_D5_sb2 = Det. 5 intensity, SB 2

I_D6_sb2 = Det. 6 intensity, SB 2

I_D7_sb2 = Det. 7 intensity, SB 2

I_D8_sb2 = Det. 8 intensity, SB 2

I_D9_sb2 = Det. 9 intensity, SB 2

I_D10_sb2 = Det. 10 intensity, SB 2

SENSOR OUTPUT:

Idemod1_f1 = Det. 1 InPhase Output, SB. 1

Iquad1_f1 = Det. 1 Quadrature Output, SB. 1

Idemod3_f1 = Det. 2 InPhase Output, SB. 1

Iquad3_f1 = Det. 2 Quadrature Output, SB. 1

Idemod2_f1 = Det. 3 InPhase Output, SB. 1

Iquad2_f1 = Det. 3 Quadrature Output, SB. 1

Idemod4_f1 = Det. 4 InPhase Output, SB. 1

Iquad4_f1 = Det. 4 Quadrature Output, SB. 1

Idemod5_f1 = Det. 5 InPhase Output, SB. 1

Iquad5_f1 = Det. 5 Quadrature Output, SB. 1

Idemod6_f1 = Det. 6 InPhase Output, SB. 1

Iquad6_f1 = Det. 6 Quadrature Output, SB. 1

Idemod7_f1 = Det. 7 InPhase Output, SB. 1

Iquad7_f1 = Det. 7 Quadrature Output, SB. 1

Idemod8_f1 = Det. 8 InPhase Output, SB. 1

Iquad8_f1 = Det. 8 Quadrature Output, SB. 1

Idemod9_f1 = Det. 9 InPhase Output, SB. 1

Iquad9_f1 = Det. 9 Quadrature Output, SB. 1

Idemod10_f1 = Det. 10 InPhase Output, SB. 1

Iquad10_f1 = Det. 10 Quadrature Output, SB. 1

Idemod1_f2 = Det. 1 InPhase Output, SB. 2

Iquad1_f2 = Det. 1 Quadrature Output, SB. 2

Idemod2_f2 = Det. 2 InPhase Output, SB. 2

Iquad2_f2 = Det. 2 Quadrature Output, SB. 2

Idemod3_f2 = Det. 3 InPhase Output, SB. 2

Iquad3_f2 = Det. 3 Quadrature Output, SB. 2

Idemod4_f2 = Det. 4 InPhase Output, SB. 2

Iquad4_f2 = Det. 4 Quadrature Output, SB. 2

Idemod5_f2 = Det. 5 InPhase Output, SB. 2

Iquad5_f2 = Det. 5 Quadrature Output, SB. 2

Idemod6_2 = Det. 6 InPhase Output, SB. 2

Iquad6_2 = Det. 6 Quadrature Output, SB. 2

Idemod7_2 = Det. 7 InPhase Output, SB. 2

Iquad7_2 = Det. 7 Quadrature Output, SB. 2

Idemod8_2 = Det. 8 InPhase Output, SB. 2

Iquad8_2 = Det. 8 Quadrature Output, SB. 2

Idemod9_2 = Det. 9 InPhase Output, SB. 2

Iquad9_2 = Det. 9 Quadrature Output, SB. 2

Idemod10_2 = Det. 10 InPhase Output, SB. 2

Iquad10_2 = Det. 10 Quadrature Output, SB. 2

Appendix 4

Comparison with Twiddle

This appendix describes the steps the user needs to take to compare SMAC results against Twiddle results.

4.1 Modeling Restrictions

The following is a list of capabilities in SMAC which cannot be used when comparisons to Twiddle are made:

- Isolators, e.g. a pick off transmissivity = 1 (Twiddle does not have this capability)
- Transfer Functions are off-resonance (the file which writes out the Twiddle files does not have the capability to add this information to the Twiddle files)
- In-phase settings on the detectors must be equal to 0 (Twiddle does not have the capability to add phase settings to the detectors)
- series modulation - An overview of series and parallel modulation is included in Chapter 2, with more explanation in Figure 2. (Twiddle only uses parallel modulation)
- overlaying modulation frequencies - If a user is modulating with 2 frequencies, these frequencies, and their higher order sidebands (e.g. J2 and J3) must not overlay one another. For example, if the first modulation frequency is 8, and 2 higher order sidebands are carried, the J1, J2, and J3 terms for this modulation frequency are 8, 16, and 24. If the second modulation frequency is 12, and 1

higher order sideband is carried, the J1 and J2 terms for this modulation are 12 and 24. The 24 (J3) of the first modulation frequency overlays the 24 (J2) of the second modulation frequency. (SMAC does not model overlaying modulation frequencies correctly)

If any of these occur, the SMAC routine which generates the Twiddle files will detect the error and will not generate the Twiddle files.

4.2 The Steps In Running The SMAC/Twiddle Comparison

Running the SMAC/Twiddle comparison involves four steps. Each step will be discussed briefly in a section below.

1. run the transfer functions in SMAC (this is done in Matlab)
2. generate the files needed for Twiddle (this is done in Matlab)
3. run the transfer functions in Twiddle (this is done in Mathematica)
4. load/compare the results (this is done in Matlab)

4.2.1 Running The Transfer Functions In SMAC

Running Transfer Function calculations is discussed at great length in Chapter 4.2 These results should be stored in a Matlab .mat file. The user should verify that none of the modeling restrictions discussed in Appendix 4.1 have been violated.

4.2.2 Generating The Files Needed For Twiddle

Once the plotting of the SMAC transfer functions has been completed, the Continue Analysis Menu will appear on the screen. This menu is discussed in Chapter 4.3. On this menu is a Set Up Data Files For Twiddle button.

Twiddle needs 6 files to run; compare_smac, ifo.m, plot.m, twiddle.m, components.m, and operations.m. The first three are generated by SMAC with the Set Up Data Files For Twiddle button. The second three must be copied from the "twiddle" subdirectory of the SMAC matlab toolbox (/home/manx4/acqmodel/twiddle).

Once the user has used the Set Up Data Files For Twiddle button to generate the first three Twiddle files, the user can exit from Matlab.

Since one of the files needed for Twiddle is called plot.m, the user should evaluate whether they wish to make a special subdirectory for these Twiddle files so that the Twiddle plot.m file does not interfere with the Matlab plot command.

4.2.3 Running The Transfer Functions In Twiddle

Once the user has the 6 files needed for twiddle in a common directory, the user should start Mathematica.

To begin running the comparison, the user should type

```
<<compare_smac
```

with a SHIFT-RETURN needed to start the calculations. When this has been done, the title of the Mathematica Window will change to include "Running". Twiddle generates the transfer functions for one actuation mode at a time, and the information is stored in a twid**.dat file, where ** refers to the relevant actuation mode.

4.2.4 Loading/Comparing The SMAC/Twiddle Results

Once the SMAC Transfer Functions have been calculated and the Twiddle Transfer Functions have been calculated, the user can compare the results. It is important that the SMAC .mat file and all of the Twiddle .dat files exist in the same directory.

When the user is ready, the user should enter Matlab, and type

```
TF_check
```

The following GUI menu will come up.

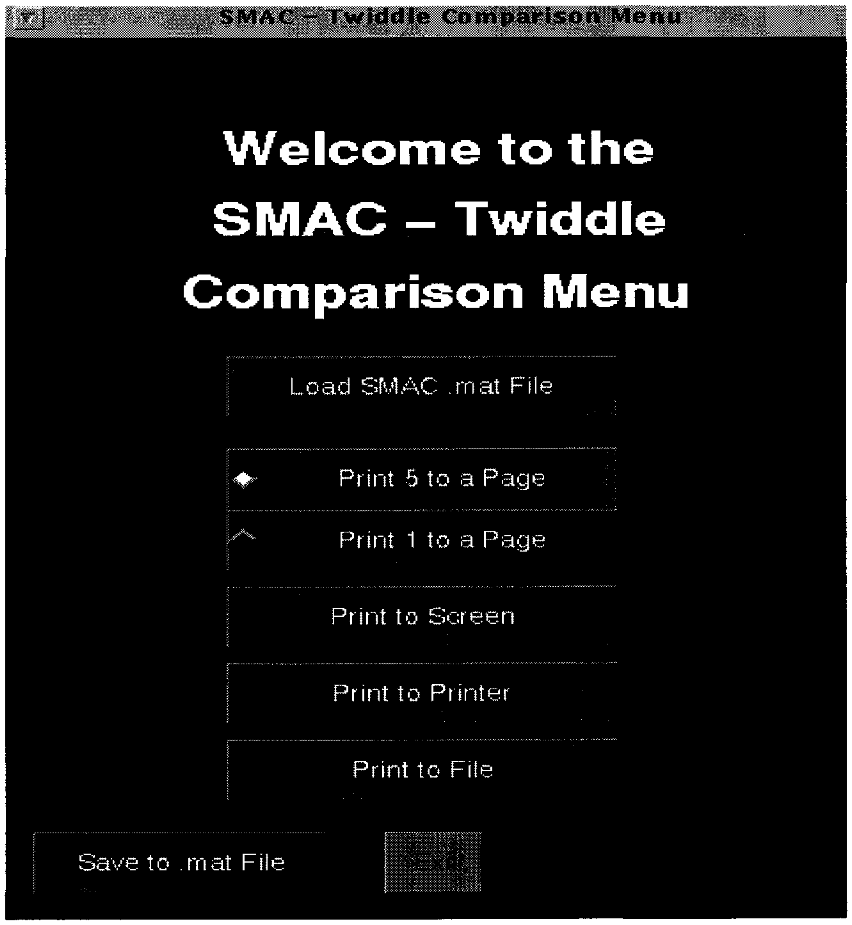


FIGURE 48. SMAC/Twiddle Comparison Menu

The first thing the user should do is Load SMAC .mat File. When the SMAC and Twiddle files are loaded, scaling is automatically done to account for the normalization with respect to carrier wavelength done by Twiddle. The user should then decide if they want the plots 5 to a page, or 1 to a page. If many Transfer Functions have been calculated, it is recommended that the user have them printed 5 to a page. Finally, when the user selects Print To Screen, Print To Printer, or Print to File, the comparison plots will be generated.

4.3 Mismatches Between Twiddle And SMAC

Twiddle is equivalent to the Dynamic model in SMAC so exact comparisons are possible between these two models. During the model validation process we found that the Mixed Mode model in SMAC was equivalent to Twiddle for the 4 km LIGO recycled interferometer parameters and essentially equivalent for the 40 m recycled interferometer parameters.

There is one known instance where the Mixed Mode model in SMAC and the Twiddle results do not match. The discrepancy occurs in transfer functions calculated when Detector 3 is the output (the gravity wave port) and the gain levels are very low. An example of the mismatch is the transfer function calculation from Laser Source to the in-phase demodulated output at Detector 3.

Appendix 5

Installing SMAC

This appendix describes the steps needed to install SMAC.

5.1 Retrieving SMAC Files

SMAC files are stored in a file called SMAC.tar. A default directory called

/home/ligo/toolbox/

should be established for storing SMAC files. Once this directory has been established, the user can untar SMAC.tar in the /home/ligo/toolbox/ directory with the following command

```
tar xvf SMAC.tar
```

In addition to several .m and .f files, 3 subdirectories will be created, "control", "cav_config", and "twiddle".

If this has successfully been completed, the user should go to Section 5.2.

If a /home/ligo/toolbox/ directory cannot be created, two portions of SMAC code will have to be modified. In these 2 sections of SMAC code, the system domain-name is checked, and depending on the result, different paths for various commands are set up so the following commands can be completed.

SMAC checks for two domainnames that were using SMAC at the time it was completed, and paths are hardwired for these two system. If the system domainname is not one of these two, then a default path is used, which is /home/ligo/toolbox/.

Since the user presumably cannot created a /home/ligo/toolbox/ directory, one of two things must be done.

The first, and probably easiest, is change the default directory from /home/ligo/toolbox/ to whatever the user has created.

The second is add a new check for the domainname the user works.

The check for domainname is done in two places so the user will have to make the modification in two files. The first is in the file gomake.m. The second is in the file cavpar_recyc.m.

5.2 Creating .mex Files

Once the SMAC directory have been installed, the user needs to create the mex files used by SMAC. At the unix prompt in the "main" SMAC directory, the user should type the following commands

fmex ryOptics.f ryOpticsG.f control/ryExeControl1.f control/ryExeControl2.f control/ryExeControl3.f control/ryExeControl4.f control/ryExeControl5.f control/ryExeControl6.f control/ryExeControl7.f control/ryExeControl8.f control/ryExeControl9.f control/ryExeControl10.f

and

fmex rzOptics.f rzOpticsG.f control/ryExeControl1.f control/ryExeControl2.f control/ryExeControl3.f control/ryExeControl4.f control/ryExeControl5.f control/ryExeControl6.f control/ryExeControl7.f control/ryExeControl8.f control/ryExeControl9.f control/ryExeControl10.f

This will create ryOptics.mexsol and rzOptics.mexsol (on Solaris systems).

If fmex is no longer available because the upgrade to Matlab5 has been made, the user should change "fmex" to "mex". This change from "fmex" to "mex" will also have to be made in the file gomake.m

5.3 Setting Permissions & Path

Finally, now that all the files have been created, the user should make sure that

- all of the .m files are readable by everyone
- the two .mex files are both readable and executable by everyone
- all of the SMAC directories, including “cav_config”, “control”, and “twiddle” are readable and executable by everyone

The user should also make sure that the “/home/ligo/toolbox/” path is added to everyone’s MATLABPATH.