EUROTEV-Report-2007-035

DEVELOPMENT OF CIRCUITS, CONTROLS AND SYSTEM MODELS FOR THE SYNCHRONIZATION AND STABILIZATION OF THE ILC CRAB CAVITIES*

G. Burt, R. Carter, A. Dexter#, R. Jenkins, I. Tahir, Cockcroft Institute, Lancaster University, Lancaster, LA1 4YR, UK

C. Beard, P. Goudket, A. Kalinin, L. Ma, P. McIntosh, ASTeC/Cockcroft Institute, STFC, Daresbury WA4 4AD, UK

ABSTRACT

The ILC reference design report (RDR) recommends a 14 mrad crossing angle for the positron and electron beams at the IP. A matched pair of crab cavity systems are required in the beam delivery system to align both bunches at the IP. The use of a multi-cell, 3.9 GHz dipole mode superconducting cavity is proposed, derived from the Fermilab CKM cavity being developed as a beam slice diagnostic [1]. Dipole-mode cavities phased for crab rotation are shifted by 90^o with respect to similar cavities phased for deflection. Uncorrelated phase errors of 0.086^o (equivalent to 61 fs) for the two cavity systems, gives an average of 180 nm for the relative deflection of the bunch centres. For a horizontal bunch size $\sigma_x = 655$ nm, a deflection of 180 nm reduces the ILC luminosity by 2%. The crab cavity systems are to be placed ~30 m apart and synchronization to within 61 fs is required; this is on the limit of what is presently achievable. This paper describes LLRF circuits under development at the Cockcroft Institute for proof of principle experiments planned on the ERLP at Daresbury and on the ILCTA test beamline at FNAL. Simulation results for stabilisation performance are also given.

Published in the Proceedings of PAC 07, Albuquerque USA, June 25-29, 2007

(Two additional graphs and a table presented during poster session are included.

The code and the derivation of the envelope equations have been added in appendices.)

*Work supported by the EC under the FP6 “Research Infrasctructure Action - Structuring the European Research Area” EUROTeV DS Project Contract no.011899 RIDS

a.dexter@lancaster.ac.uk

PRELIMINARY LLRF LAYOUT

There are a wide range of possible technologies and configurations that might be used for the crab cavity LLRF system. The circuits being developed are for proof of principle. Important criteria for the current system are flexibility and accurate synchronisation in the absence of beam based calibration. For superconducting cavities, the microphonics which act to spoil phase synchronism are independent for each cavity. Accordingly we chose separate amplifiers and controllers for each cavity. In order to synchronize the crab cavities we must therefore provide accurately synchronized timing signals very close to the output couplers of each cavity. The cavities then need to be stabilized with respect to the local timing signal. The timing error between cavity phases has three components, one from the synchronization of the timing signals and two from the cavity to timing signal synchronization. Our initial target is a cavity to a timing signal synchronization of 20 fs and a timing signal to timing signal synchronization also of 20 fs. Timing synchronization of 10 fs has recently been claimed at LBLN using mode locked lasers [2]. Our current system uses an RF interferometer which is not expected to achieve this performance but which could be easily replaced with such a system. The development system sketched in figure 1 employs digital phase detection permitting absolute measurement with minimal calibration issues, 16 bit A to D conversion at 100 MSPS, DSP control algorithm implementation giving maximum flexibility and IQ modulation of the RF drive.

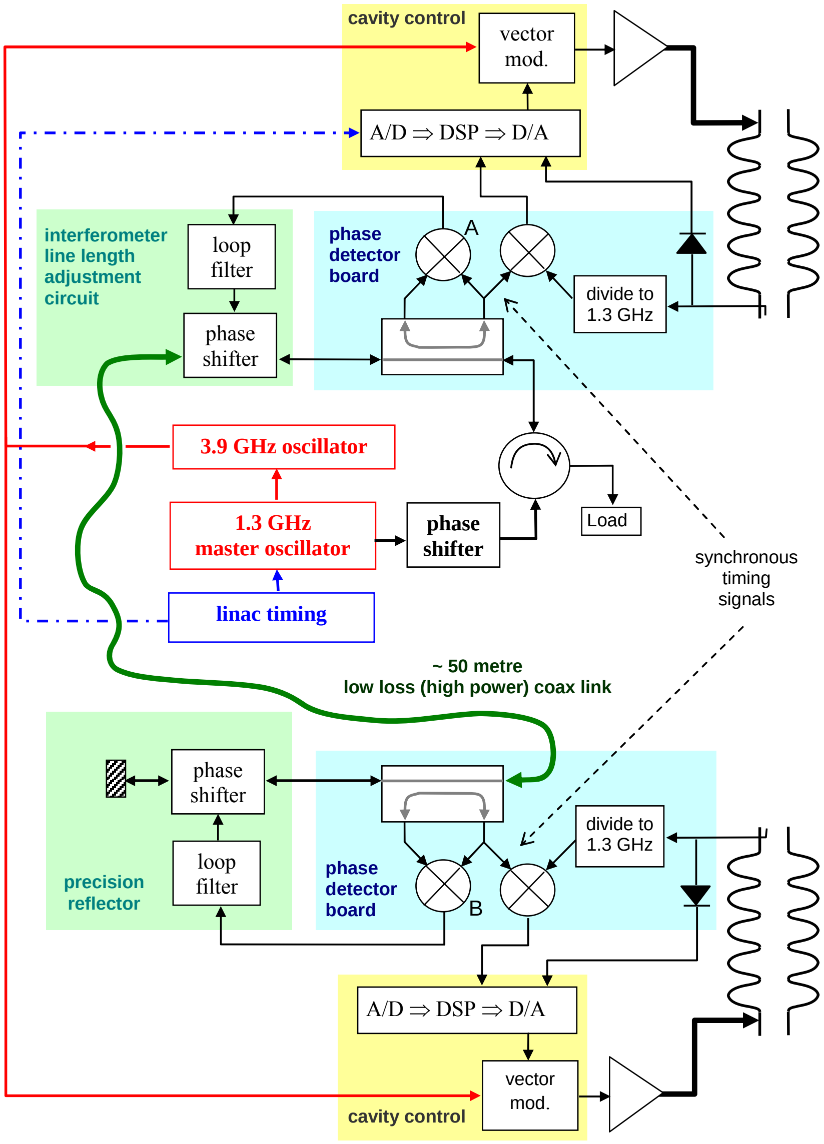


Figure 1: Synchronization layout.

CAVITY MODEL

The accuracy to which the phase of each cavity can be stabilized against unpredictable components of the disturbing influences of microphonics and beam-loading, will depend on how accurately the phase can be measured and the maximum control system gain consistent with stability. High gain RF control stability of the TESLA cavities has recently been discussed by Vogel [3]. The gain stability limit for the control system will depend on loop delay. For an analogue control loop, delay will arise from input filtering and amplifier bandwidth. For the digital control system being proposed here, control loop delay depends additionally on ADC, DAC and DSP processing time. Compensation for the additional loop delay of a digital controller is made by opportunities for sophisticated input filtering, real time variation of control parameters and anticipation of repetitive disturbances. Calculation indicates that to get the required performance, digital processing delay needs to be less than about 1 μs. The anticipated performance of the control system can be modelled numerically. We apply the standard equations for cavity filling and numerically integrate the envelope equations [4] for anticipated microphonics and worst case beam-loading. Figure 2 shows the effective (diagonalized) equivalent circuit for a multi-cell cavity driven via a coupler. Values L_i , R_i , C_i are chosen so that each parallel resonator represents a cavity mode and are determined from modal frequencies, Q and R/Q values.

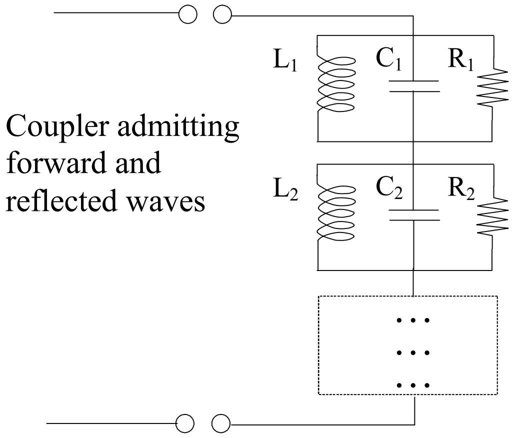


Figure 2: Cavity model.

One therefore solves

$$\frac{d^2V_m}{dt^2} + \frac{\omega_m}{Q_{om}}\frac{dV_m}{dt} + \frac{1}{Q_{em}}\omega_m\sum_{j=1}^{N}\frac{dV_j}{dt} + \omega_m^2 V_m = \frac{2\omega_m}{Q_{em}}\frac{d}{dt}\{\mathcal{F}\exp(-j\omega t)\}$$

where V_m , ω_m , Q_{om} and Q_{em} are the voltage, frequency, unloaded Q and Q external for the mth mode respectively and $\mathcal{F}$ is the amplitude of the forward wave in the coupler, ω is the drive frequency. This set of equations must be solved numerically as the modal frequencies are functions of time as determined by microphonics and the modal voltages take a step change each time a bunch passes through the cavity. These equations cannot be accurately integrated over the fill time and bunch train time which amounts to at least 10^7 RF cycles. Instead one solves for the real and imaginary parts of an amplitude function [4] determined by the equation

$$V_m(t) = \{A_{mr}(t) + jA_{mi}(t)\}\exp\{-j\omega t\}$$

After neglecting second time derivatives of the slowly varying amplitude functions A_{mr} and A_{ri} and also terms of order $(1/Q_e)^2$ one obtains the envelope equations

$$\frac{\dot{A}_{mr}}{\omega_m} + \frac{1}{4Q_{om}}\left(\frac{\omega_m^2}{\omega^2}+1\right)A_{mr} + \frac{1}{4Q_{em}}\sum_{j=1}^{N}\left(\frac{\omega_j^2}{\omega^2}+1\right)A_{jr} - \left(\frac{\omega_m}{\omega}-\frac{\omega}{\omega_m}\right)\frac{A_{mi}}{2} = -\frac{1}{Q_{em}}\left(\frac{\dot{\mathcal{F}}_i}{\omega} - \mathcal{F}_r\right)$$

$$\frac{\dot{A}_{mi}}{\omega_m} + \frac{1}{4Q_{om}}\left(\frac{\omega_m^2}{\omega^2}+1\right)A_{mi} + \frac{1}{4Q_{em}}\sum_{j=1}^{N}\left(\frac{\omega_j^2}{\omega^2}+1\right)A_{ji} + \left(\frac{\omega_m}{\omega}-\frac{\omega}{\omega_m}\right)\frac{A_{mr}}{2} = \frac{1}{Q_{em}}\left(\frac{\dot{\mathcal{F}}_r}{\omega}+\mathcal{F}_i\right)$$

For the planned 9 cell cavity, modes will be designated as $\text{mode} = \pi(1-m/9)$ so that m = 0 gives the π mode.

The real and imaginary parts of the forward wave are determined by the controller which must reduce A_{0i} to zero and hold A_{0r} at a steady level V_{sp} as appropriate for required kick. At this stage we have no detailed knowledge of system disturbances hence use of a proportional integral controller is appropriate for the model. Explicitly we take

$$\mathcal{F}_r\left(t+t_{delay}\right) = c_{pr}\left(V_{sp} - \overline{\sum_m A_{mr}}\right) + c_{ir}\int_{-\infty}^{t} dt\left(V_{sp} - \overline{\sum_m A_{mr}}\right)$$

$$\mathcal{F}_i\left(t+t_{delay}\right) = -c_{pi}\overline{\sum_m A_{mi}} - c_{ii}\int_{-\infty}^{t} dt\,\overline{\sum_m A_{mi}}$$

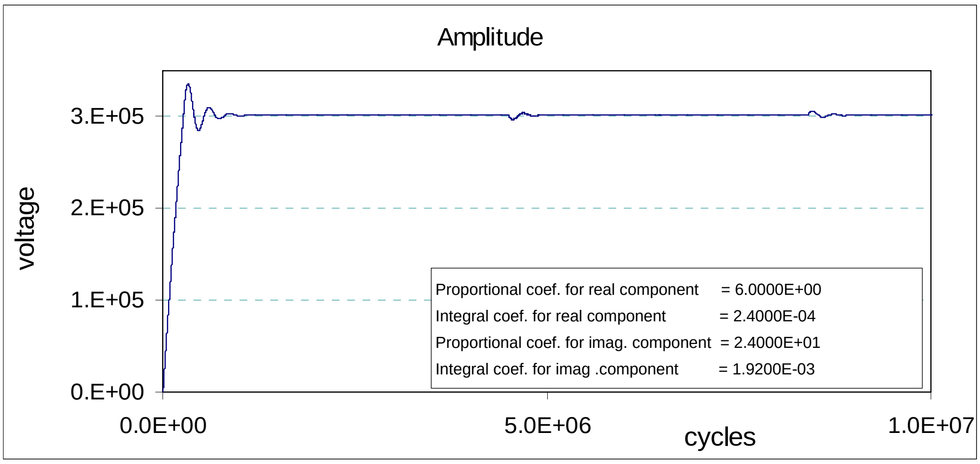
where c_{pr} , c_{pi} , c_{ir} and c_{ii} are the controller coefficients and t_{delay} represents time delays in the digital processor. One wishes to control the π mode, however the measured cavity voltage will be some weighted time average of the summed excitation of the modes. As the weightings for this average have yet to be determined, the modes are sampled with equal weight through a low pass filter having a time constant equal to the sampling rate. The amplifier is modelled as a second order filter with a time constant determined from its bandwidth. Beamloading of the π mode with the ILC time structure is included. Wakefield calculations have shown that excitation of other modes by the beam is small. Monochromatic cavity vibration appropriate to the most prominent frequency measure on prototype CKM cavities is included.

CALCULATION PARAMETERS

Drive frequency in GHz	3.9 GHz	
Centre cavity frequency in GHz	3.9 GHz	
Number of cavity modes	1 or 3	3 for figs. 9 - 12
Cavity Q factor	1.0e9	
External Q factor	3.0e6	
Cavity R over Q (2xFNAL=53 per cell)	53 Ω per cell	
Energy per cell	28.4 mJ	
Maximum Amplifier Power per Cell	1200 W	
Maximum beam offset	0.6 mm	
Maximum random bunch timing error	712 fs	0 for figs. 3 - 7
Beam offset frequency	0 or 2 kHz	0 for figs. 3 - 7
Bunch Charge	3.2 nC	
RF cycles between bunches	1200 (~ 308 ns)	
Bunch train length	1 ms	
Cavity frequency shift for microphonics	600 Hz	
Cavity vibration frequency	230 Hz	
Initial vibration phase	20^{o}	
Phase measurement error	taken as zero	Except in fig. 12
Time delay for digital control (latency)	1 μs	
Control update frequency	1 μs	
Amplifier bandwidth	10 MHz	
Real part proportional control coefficient	6.0	× Gain factor in fig. 9
Real part integral control coefficient	2.4e-4	× Gain factor in fig. 9
Imag. part proportional control coefficient	24	× Gain factor in fig. 9
Imag. part integral control coefficient	1.92e-3	× Gain factor in fig. 9

RESULTS

Figures 3-7 show cavity voltage, drive amplitude and cavity phase during cavity filling and the passage of an ILC bunch train with an offset of 0.6 mm after 4.5 million 3.9 GHz, RF cycles. In this calculation only the cavity π mode is modelled. Phase measurement errors are not included at this stage. The cavity has a microphonic frequency of 230 Hz which shifts the RF frequency by 600 Hz. The "in phase" drive amplitude follows the beam loading whilst the "phase quadrature" drive amplitude compensates for microphonics as expected. If the beam offset is permitted to oscillate, then for perfect bunch timing the phase response remains as in figure 7. Figure 8 shows the influence of bunch timing errors when the bunch offset oscillates at 2 kHz. Note that the amplifier bandwidth was taken as 10 MHz throughout. For all figures except 9 the gain was 36% below the point of instability [5]. Figure 9 shows RMS phase error as a function of gain. Figures 10 and 11 show the effect of including the 8π/9 mode at +2.2 MHz and the 7π/9 mode at +8.9 MHz in addition to the π at 3.9 GHz mode. Interestingly, beam-loading is seen to couple to the phase error when additional modes are included in the model. Figure 11 represents a worst case control scenario in the absence of phase measurement errors. Using digital phase detectors and for the bandwidth appropriate to the model, the measurement jitter after division to 1.3 GHz will be of the order of 10 milli-degrees. Figure 12 shows relative to figure 11 the effect of random phase measurement errors of +/- 10 milli-degrees and amplitude errors of +/- 0.2%. Figure 12 suggests that stabilisation to 30 milli-degrees ~ 20 fs looks feasible.



Proportional coef. for real component = 6.0000E+00
Integral coef. for real component = 2.4000E-04
Proportional coef. for imag. component = 2.4000E+01
Integral coef. for imag .component = 1.9200E-03

Figure 3 One mode and steady 0.6 mm beam offset

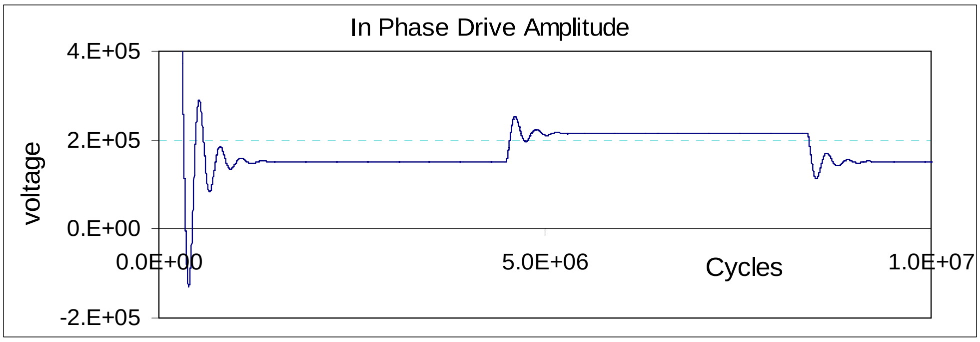


Figure 4 One mode and steady 0.6 mm beam offset

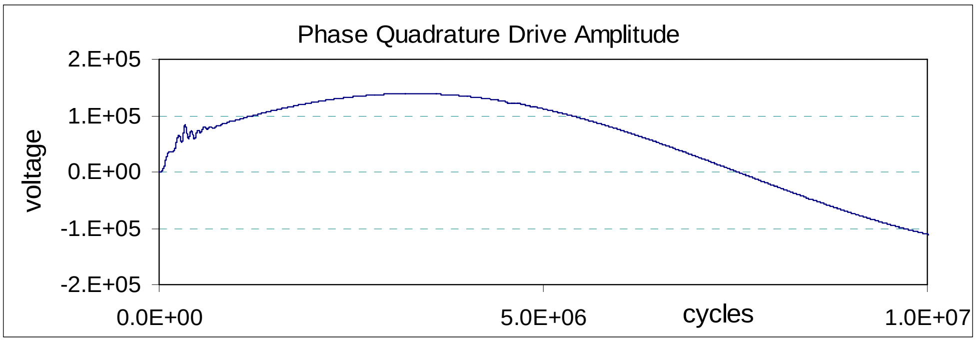


Figure 5 One mode and steady 0.6 mm beam offset

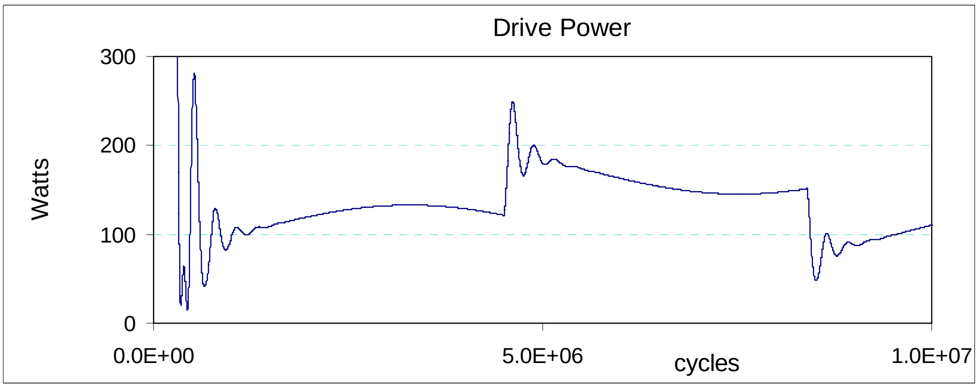


Figure 6 One mode and steady 0.6 mm beam offset

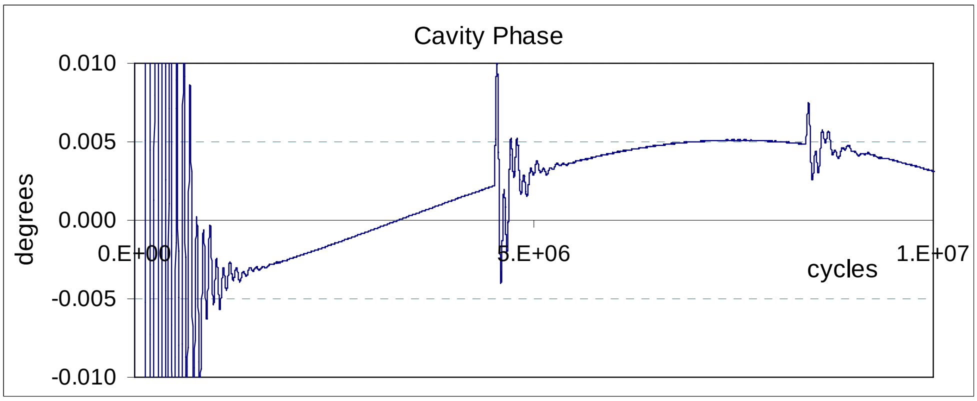


Figure 7 One mode and steady 0.6 mm beam offset

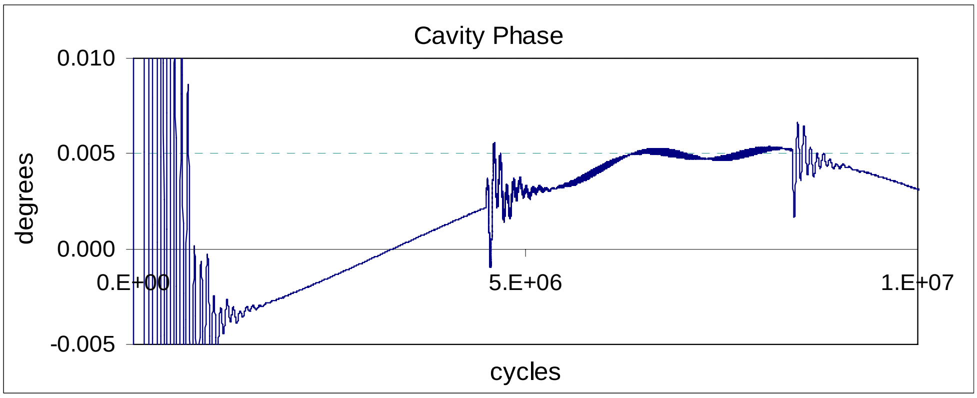


Figure 8 One mode and 0.6 mm oscillating beam offset and with 712 fs random, bunch timing errors

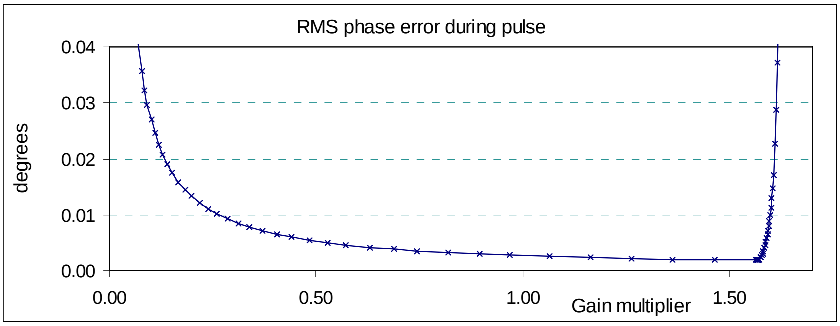


Figure 9 RMS phase performance as function of gain for three modes

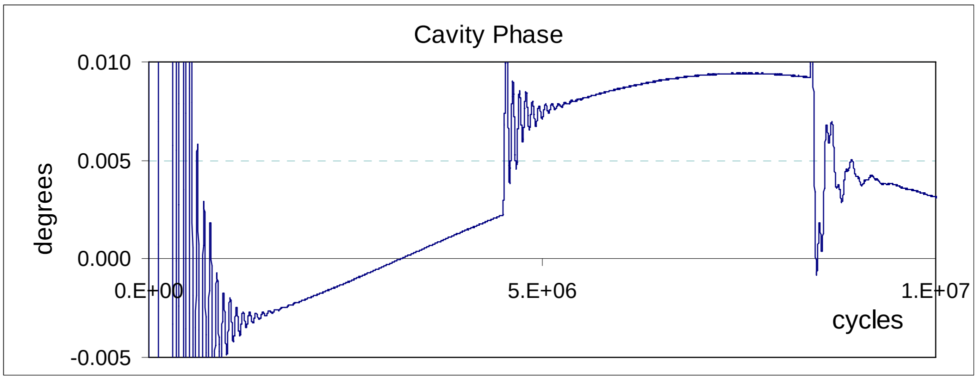


Figure 10 Three modes and steady 0.6mm beam offset

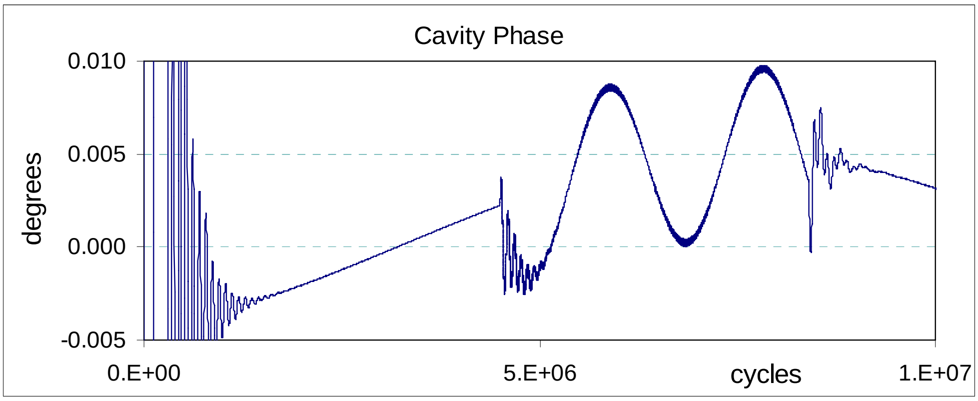


Figure 11 Three modes and 0.6mm oscillating beam offset with 712 fs random, bunch timing errors

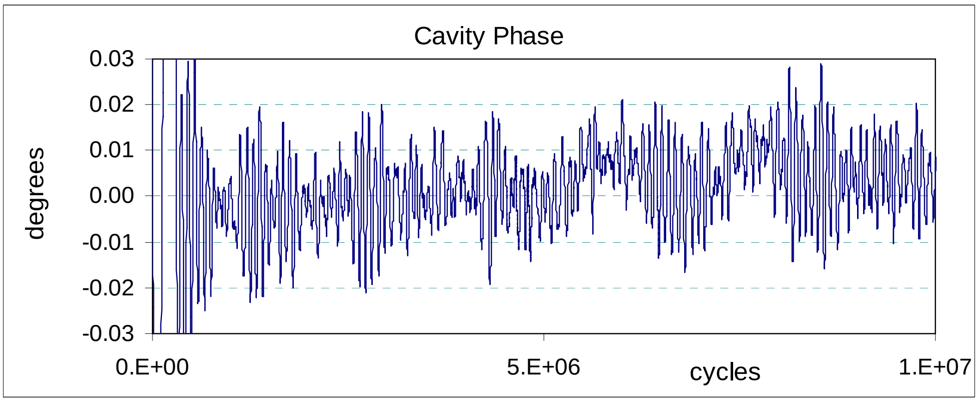


Figure 12 As figure 11 but with random phase measurement errors of +/- 10 milli-degrees

FUTURE WORK

The LLRF circuits are soon to be tested by driving two adjacent cavities, independently in a vertical cryostat. As the cavities are adjacent it will be possible to make an independent measurement of relative cavity phase jitter.

ACKNOWLEDGEMENTS

The authors would like to thank L. Bellantoni for his helpful comments and provision of data for this study which is taking place under the envelope of the ILC crab cavity collaboration that includes FNAL as a major contributor. The authors would also like to thank Stefan Simrock and Brian Chase for their advice on the development of the LLRF system.

The work here has been supported in part by the EC under Contract no.011899 RIDS and in part by UK Science and Technology Facilities Council (STFC).

REFERENCES

[1] M. McAshan, R. Wanzenberg, RF Design of a Transverse Mode Cavity for Kaon Separation, FERMILAB-TM-2144, May 2001.

[2] S. Lidia, “Progress in high precision timing and synchronization at LBNL”, ERL07, Daresbury 2007

[3] E. Vogel, “High Gain proportional rf control stability at TESLA cavities”, Phys Rev. Special Topics – Accelerators and Beams 10, 052001 (2007)

[4] H. Henke, M. Filtz, “Envelope equations for transients in linear chains of resonators”, PAC 1993

[5] H. Imsieke, A. Kholdyni, S.N. Simrock, “Time delay Compensation for the digital RF control at the TESLA test facility” LINAC 98

APPENDIX 1 DERIVATION OF ENVELOPE EQUATIONS

The time evolution of a cavity mode obeys the same differential equation as a parallel lumped circuit as shown in figure 2. At the terminal the voltage in the coupler must equal the voltage in the lumped circuit. Assuming that the coupler can be represented in some sense as a waveguide then along this waveguide the voltage and current satisfies the equations

$$\frac{\partial V}{\partial z} = -L_{wg}\frac{\partial I}{\partial t} \qquad \text{and} \qquad \frac{\partial I}{\partial z} = -C_{wg}\frac{\partial V}{\partial t} \qquad \text{hence} \qquad \frac{\partial^2 V}{\partial z^2} = L_{wg}C_{wg}\frac{\partial^2 V}{\partial t^2}$$

where C_{wg} is the capacitance per unit length and L_{wg} is the inductance per unit length.

For a fixed frequency source of angular frequency ω the voltage along the waveguide is given as

$$V(z,t) = \mathcal{F}\exp\{j(kz-\omega t)\} + \mathcal{R}\exp\{-j(kz+\omega t)\} \tag{1}$$

where $k = \omega\sqrt{L_{wg}C_{wg}}$

and $\mathcal{F}$ is the amplitude of the forward wave and $\mathcal{R}$ is the amplitude of the reflected wave.

The current on the waveguide is therefore given as

$$I(z,t) = \mathcal{F}\frac{\omega C_{wg}}{k}\exp\{j(kz-\omega t)\} - \mathcal{R}\frac{\omega C_{wg}}{k}\exp\{-j(kz+\omega t)\}$$

which can be written as

$$I(z,t) = \frac{1}{Z_{wg}}\left[\mathcal{F}\exp\{j(kz-\omega t)\} - \mathcal{R}\exp\{-j(kz+\omega t)\}\right] \tag{2}$$

$$\text{where } Z_{wg} = \sqrt{\frac{L_{wg}}{C_{wg}}} \tag{3}$$

If the terminal between the cavity and the waveguide is at z = 0 then the current in the waveguide equals the sum of the currents through the equivalent circuit components of each series resonator hence

$$\frac{1}{L_i}\int V_i dt + C_i\frac{dV_i}{dt} + \frac{V_i}{R_i} = \frac{1}{Z_{wg}}\{\mathcal{F} - \mathcal{R}\}\exp(-j\omega t) \tag{4}$$

From (1) the voltage at z = 0 is given as

$$V = \sum_{\text{modes}} V_i = (\mathcal{F} + \mathcal{R})\exp(-j\omega t) \tag{5}$$

Eliminating the reflected power between (4) and (5) gives

$$\frac{1}{L_i}\int V_i dt + C_i\frac{dV_i}{dt} + \frac{V_i}{R_i} + \frac{1}{Z_{wg}}\sum_{j=1}^{N} V_j = \frac{2\mathcal{F}}{Z_{wg}}\exp(-j\omega t) \tag{6}$$

If the coupling to different modes is dissimilar then Z_{wg} takes a different value for each mode.

This equation determines the modal voltages in the cavity as a function of the amplitude of the forward wave in the waveguide.

Now define the natural frequency of the i^{th} mode as

$$\omega_i = \frac{1}{\sqrt{L_i C_i}} \tag{7}$$

To evaluate Z_{wg} we write

$$Q_{ei} = \frac{\omega_i U_{stored}}{U_{emitted}} = \frac{\frac{1}{2}\omega_i\, C_i\, V_i^2}{\frac{1}{2}\left(V_i^2 / Z_{wg}\right)} = \omega_i\, Z_{wgi} C_i \tag{8}$$

$$Q_{oi} = \frac{\omega_i U_{stored}}{U_{diss}} = \frac{\frac{1}{2}\omega_i\, C_i\, V_i^2}{\frac{1}{2}\left(V_i^2 / R_i\right)} = \omega_i\, R_i\, C_i \tag{9}$$

Hence dividing (8) and (9) we have that

$$\frac{Q_{ei}}{Q_{oi}} = \frac{Z_{wgi}}{R_i} \tag{10}$$

which can be re-arranged as

$$Z_{wgi} = \left(\frac{R}{Q}\right)_{oi} Q_{ei} \tag{11}$$

i.e. Z_{wg} is the product of the external *Q* with the R/Q of the bare cavity. Note that Z_{wg} is not that of the physical waveguide from the RF generator as represented in figure 2 by the transmission line from the current source to the transformer. The transformer models the coupler which transforms the voltage.

Differentiation of (6) and division by C_i gives

$$\frac{d^2V_i}{dt^2} + \frac{\omega_i}{\omega_i\, R_i\, C_i}\frac{dV_i}{dt} + \frac{\omega_i}{\omega_i Z_{wg} C_i}\sum_{j=1}^{N}\frac{dV_j}{dt} + \frac{1}{L_i\, C_i} V_i = \frac{2\,\omega_i}{\omega_i Z_{wgi} C_i}\frac{d}{dt}\left\{\mathcal{F}\exp(-j\omega t)\right\} \tag{12}$$

and using (7), (8) and (10) in (11) gives

$$\frac{d^2V_i}{dt^2} + \frac{\omega_i}{Q_{oi}}\frac{dV_i}{dt} + \frac{1}{Q_{ei}}\omega_i\sum_{j=1}^{N}\frac{dV_j}{dt} + \omega_i^2\, V_i = \frac{2\,\omega_o}{Q_e}\frac{d}{dt}\left\{\mathcal{F}\exp(-j\omega t)\right\} \tag{13}$$

The Envelope Equations

The solution of the differential equation (13) is only fast with a modern PC. For high Q calculations it is difficult to obtain phase prediction with an accuracy of milli-degrees. A faster solution technique is to assume a solution of the form

$$V_m(t) = \left\{A_{mr}(t) + jA_{mi}(t)\right\}\exp\left\{-j\omega t\right\} \tag{14}$$

where $A_{mr}(t)$ and $A_{mi}(t)$ are slowly varying functions of time and ω is the drive frequency. A similar form is assumed for $\mathcal{F}$. It will be seen that that when the assumed solution is substituted into the differential equation second derivatives can be neglected. Differentiating the RHS of (13) and renaming the index for the modes as m to avoid confusion with imaginary parts we get

$$\frac{d^2V_m}{dt^2} + \frac{\omega_m}{Q_{om}}\frac{dV_m}{dt} + \omega_m^2\, V_m + \frac{\omega_m}{Q_{em}}\sum_{j=1}^{N}\frac{dV_j}{dt} = \frac{2\,\omega_m}{Q_{em}}\left(\dot{\mathcal{F}} - j\omega\mathcal{F}\right)\exp(-j\omega\, t) \tag{15}$$

Differentiation of (14) gives

$$\frac{dV_m}{dt} = \left(\dot{A}_{mr} - j\omega A_{mr} + j\dot{A}_{mi} + \omega A_{mi}\right)\exp\{-j\omega t\} \qquad (16)$$

second differentiation gives

$$\frac{d^2V_m}{dt^2} = \left(\ddot{A}_{mr} - 2j\omega\,\dot{A}_{mr} - \omega^2 A_r + j\ddot{A}_{mi} + 2\omega\,\dot{A}_{mi} - j\omega^2 A_{mi}\right)\exp\{-j\omega t\} \qquad (17)$$

Substituting (16) and (17) in (15), cancelling a factor of $\exp(-j\omega t)$ and separating into real and imaginary parts gives the envelope equations as

$$\left(\ddot{A}_{mr} + 2\omega\,\dot{A}_{mi} - \omega^2 A_{mr}\right) + \frac{\omega_m}{Q_{om}}\left(\dot{A}_{mr} + \omega A_{mi}\right) + \frac{\omega_m}{Q_{em}}\sum_{j=1}^{N}\left(\dot{A}_{jr} + \omega A_{ji}\right) + \omega_m^2 A_{mr} = \frac{2\omega_m}{Q_{em}}\left(\dot{\mathcal{F}}_r + \omega\mathcal{F}_i\right) \qquad (18a)$$

$$\left(\ddot{A}_{mi} - 2\omega\,\dot{A}_{mr} - \omega^2 A_{mi}\right) + \frac{\omega_m}{Q_{om}}\left(\dot{A}_{mi} - \omega A_{mr}\right) + \frac{\omega_m}{Q_{em}}\sum_{j=1}^{N}\left(\dot{A}_{ji} - \omega A_{jr}\right) + \omega_m^2 A_{mi} = \frac{2\omega_m}{Q_{em}}\left(\dot{\mathcal{F}}_i - \omega\mathcal{F}_r\right) \qquad (19a)$$

Now ω_m and ω are big numbers whilst $\dot{A}$ and $\ddot{A}$ are not so big as A is a slowly varying envelope function. This means that in (18a) and (19a) the second derivative terms i.e. $\ddot{A}$ are an order of magnitude smaller than all the other terms, i.e. we retain order ω and ω^2 though we shall see that order ω^2 cancels to order ω. Eliminating 2nd derivatives in (18a) and (19a) they become

$$2\omega\,\dot{A}_{mi} + \frac{\omega_m}{Q_{om}}\dot{A}_{mr} + \frac{\omega_m}{Q_{em}}\sum_{j=1}^{N}\dot{A}_{jr} + \frac{\omega\omega_m}{Q_{om}}A_{mi} + \frac{\omega\omega_m}{Q_{em}}\sum_{j=1}^{N}A_{ji} + \left(\omega_m^2 - \omega^2\right)A_{mr} = \frac{2\omega_m}{Q_{em}}\left(\dot{\mathcal{F}}_r + \omega\mathcal{F}_i\right) \qquad (18b)$$

$$-2\omega\,\dot{A}_{mr} + \frac{\omega_m}{Q_{om}}\dot{A}_{mi} + \frac{\omega_m}{Q_{em}}\sum_{j=1}^{N}\dot{A}_{ji} - \frac{\omega\omega_m}{Q_{om}}A_{mr} - \frac{\omega\omega_m}{Q_{em}}\sum_{j=1}^{N}A_{jr} + \left(\omega_m^2 - \omega^2\right)A_{mi} = \frac{2\omega_m}{Q_{em}}\left(\dot{\mathcal{F}}_i - \omega\mathcal{F}_r\right) \qquad (19b)$$

In superconducting cavities Q_{Lm} and Q_{em} are always going to be large ($\sim 10^6$) hence terms in $1/Q^2$ can always be neglected where the are multiplied by the same power of ω. This observation allows us to diagonalise the derivative terms of (18b) and (19b) by substituting (18b) into (19b) and (18b) into (19b) neglecting powers $1/Q^2$. This gives

$$2\omega \dot{A}_{mi} + \frac{1}{2Q_{om}}\frac{\omega_m}{\omega}\left(\omega_m^2-\omega^2\right)A_{mi} + \frac{1}{2Q_{em}}\frac{\omega_m}{\omega}\sum_{j=1}^{N}\left(\omega_j^2-\omega^2\right)A_{ji} + \frac{\omega\omega_m}{Q_{om}}A_{mi} + \frac{\omega\omega_m}{Q_{em}}\sum_{j=1}^{N}A_{ji} + \left(\omega_m^2-\omega^2\right)A_{mr} = \frac{2\omega_m}{Q_{em}}\left(\dot{\mathcal{F}}_r + \omega\mathcal{F}_i\right) \tag{18c}$$

$$-2\omega \dot{A}_{mr} - \frac{1}{2Q_{om}}\frac{\omega_m}{\omega}\left(\omega_m^2-\omega^2\right)A_{mr} - \frac{1}{2Q_{em}}\frac{\omega_m}{\omega}\sum_{j=1}^{N}\left(\omega_j^2-\omega^2\right)A_{jr} - \frac{\omega\omega_m}{Q_{om}}A_{mr} - \frac{\omega\omega_m}{Q_{em}}\sum_{j=1}^{N}A_{jr} + \left(\omega_m^2-\omega^2\right)A_{mi} = \frac{2\omega_m}{Q_{em}}\left(\dot{\mathcal{F}}_i - \omega\mathcal{F}_r\right) \tag{19c}$$

Combing terms and normalising the differentials in (18c) and (19c) and swapping the order of (18c) and (19c) gives

$$\frac{\dot{A}_{mr}}{\omega_m} + \frac{1}{4Q_{om}}\left(\frac{\omega_m^2}{\omega^2}+1\right)A_{mr} + \frac{1}{4Q_{em}}\sum_{j=1}^{N}\left(\frac{\omega_j^2}{\omega^2}+1\right)A_{jr} - \left(\frac{\omega_m}{\omega}-\frac{\omega}{\omega_m}\right)\frac{A_{mi}}{2} = -\frac{1}{Q_{em}}\left(\frac{\dot{\mathcal{F}}_i}{\omega}-\mathcal{F}_r\right) \tag{20}$$

$$\frac{\dot{A}_{mi}}{\omega_m} + \frac{1}{4Q_{om}}\left(\frac{\omega_m^2}{\omega^2}+1\right)A_{mi} + \frac{1}{4Q_{em}}\sum_{j=1}^{N}\left(\frac{\omega_j^2}{\omega^2}+1\right)A_{ji} + \left(\frac{\omega_m}{\omega}-\frac{\omega}{\omega_m}\right)\frac{A_{mr}}{2} = \frac{1}{Q_{em}}\left(\frac{\dot{\mathcal{F}}_r}{\omega}+\mathcal{F}_i\right) \tag{21}$$

For Runge Kutta Solution we write these equations in the form

$$\dot{A}_{mr} = -\frac{\omega_m}{4Q_{om}}\left(\frac{\omega_m^2}{\omega^2}+1\right)A_{mr} - \frac{\omega_m}{4Q_{em}}\sum_{j=1}^{N}\left(\frac{\omega_j^2}{\omega^2}+1\right)A_{jr} + \left(\omega_m^2-\omega^2\right)\frac{A_{mi}}{2\omega} - \frac{\omega_m}{\omega Q_{em}}\left(\dot{\mathcal{F}}_i - \omega\mathcal{F}_r\right) \tag{22}$$

$$\dot{A}_{mi} = -\frac{\omega_m}{4Q_{om}}\left(\frac{\omega_m^2}{\omega^2}+1\right)A_{mi} - \frac{\omega_m}{4Q_{em}}\sum_{j=1}^{N}\left(\frac{\omega_j^2}{\omega^2}+1\right)A_{ji} - \left(\omega_m^2-\omega^2\right)\frac{A_{mr}}{2\omega} + \frac{\omega_m}{\omega Q_{em}}\left(\dot{\mathcal{F}}_r + \omega\mathcal{F}_i\right) \tag{23}$$

APPENDIX 2 CODE

```
c     Driven oscillator with beamloading, microphonics & delayed PI control on drive
c     ***************************************************************************
c     Still requires a more realistic amplifier model

c     Last modified 13th June 2007
c     This version includes measurement model, amplifier model and multi-mode cavity

c     This program solves the envelope equations for
c     C*dV/dt+V/Zext+(1/L)*Integral(V*dt)=2*Forw*cos(wd*t+psi)/Zext

c     which are formed by seting V=(Ar+j*Ai)*exp(-j*wd*t) and neglecting second
c     derivatives of Ar and Ai

c     Forw is the amplitude of the forward wave and is determined by a PI controller

c     Differential equations are solved by 4th order Runge Kutta

c234567890123456789012345678901234567890123456789012345678901234567890123456789012

      integer*4 jic, jm
      parameter(jic=100000)
      parameter(jm=3)

      real*8 V_set_point(jm), vjump, drive_amp, drive_max, V_max_point
      real*8 energy_set_point
      real*8 Vkick, Max_power, drive_pow, drive_phase, bunch_phase, c
      real*8 Offset, bunch_phase_err, cos_err, sin_err, bunch_charge
      real*8 vdr, Vdr_last, vdi, Vdi_last, vdr_dot, vdi_dot
      real*8 wd, wc(jm), wc0(jm), QC(jm), QE(jm)
      real*8 wa, QA, ROQ, amp_bandwidth
      real*8 pi, period, t0, dt, t2, tstart, hdt, dt6, time
      real*8 time_step, time_last, time_step1, time_step2
      real*8 fc(jm), fd
      real*8 f0(jm), fe(jm), g1(jm), g2(jm), g3(jm)
      real*8 ga1, ga2, ga3, ga4, ga5
      real*8 fa1, fa2, fa3, fa4
      real*8 AR(jm), AI(jm), AR_sum, AI_sum
      real*8 amp_AR0, amp_AR
      real*8 amp_AI0, amp_AI
      real*8 amp_AR_dot, amp_AI_dot
      real*8 outr(jm), outi(jm)
      real*8 AR0_meas, AI0_meas, rand_phase, rand_mag, mag_factor
      real*8 meas_phase_jitter, meas_amp_jitter, meas_phase_jitter_deg
      real*8 amplitude(jm), phase(jm), phase_deg(jm)
      real*8 driver(jic), drivei(jic)
      real*8 dr_ramp, di_ramp, dr_flat, di_flat
      real*8 cntr_delay, update_interval
      real*8 c_prop_Ar, c_intl_Ar, c_prop_Ai, c_intl_Ai
      real*8 sumr, sumi, Rerror, Ierror, summax
      real*8 cycle
      real*8 xrandom
      real*8 random
      real*8 aa
      real*8 offset_freq, w_offset
      real*8 pulse_length
      real*8 max_power_used
      real*8 drive_freq, cavity_freq, cavity_vib_freq, cavity_freq_shift
      real*8 initial_vib_phase
      real*8 wshift, wcav, vph
      real*8 sum_sq_ph_err(jm), sum_sq_amp_err(jm), amp_err(jm), value
      real*8 gain
      real*8 Qmeas, AR0_filter, AI0_filter
      real*8 testr, testi
      real*8 mode_coupling(jm)

      integer*4 n_iterations, ncycle, icycle, j, js, nprint, nwrite
      integer*4 its_per_cycle, idelay, ic_delay, ic
      integer*4 nbeam_on, ibeam, nbeam_off, ndetail, nbeam
```

```
      integer*4 cycles_per_train
      integer*4 count, control_update, cnt_param, nsettle
      integer*4 normalise
      integer*4 n_vec_mod
      integer*4 nerror, modes

      character*80 anything

      logical lrandom

      intrinsic abs, atan, cos, sin, sqrt, acos, asin
      external random, RK, RKM

      pi=4.0*atan(1.0d00)

      c=2.998d08

c     Read input data from file
c     *************************
      open(file='indata.txt',unit=45,status='old')
      open(file='outdata.txt',unit=46,status='modify')

900   format(a)

      read(45,900) anything
      nerror=952
      read(anything(41:),*,err=3000) drive_freq
      write(*,952) drive_freq
      write(46,952) drive_freq
      fd=drive_freq*1.0d09
952   format('Drive frequency in GHz                  =',f11.3,' GHz')

      read(45,900) anything
      nerror=953
      read(anything(41:),*,err=3000) cavity_freq
      write(*,953) cavity_freq
      write(46,953) cavity_freq
      fc(1)=cavity_freq*1.0d09
953   format('Centre cavity frequency in GHz          =',f11.3,' GHz')

      modes=3
      write(46,9539) modes
9539  format('Number of cavity modes                  =',i2)

      fc(2)=fc(1)+2.0d+06
      fc(3)=fc(1)+9.0d+07
      mode_coupling(1)=1.0d00
      mode_coupling(2)=0.5d00
      mode_coupling(3)=0.25d00

      mode_coupling(2)=0.75d00
      mode_coupling(3)=0.25d00

      wd=2.0d00*pi*fd
      do 61 js=1,modes
        wc0(js)=2.0d00*pi*fc(js)
61    continue

      aa=0.5d00*c*pi/wd
c     reference radius to get voltage setpoint (aa*w/c)=pi/2

      read(45,900) anything
      nerror=954
      read(anything(41:),*,err=3000) QC(1)
      write(*,954) QC(1)
      write(46,954) QC(1)
954   format('Cavity Q factor                         =',1pe11.4)
      do 62 js=1,modes
       QC(js)=QC(1)
62    continue
```

```
      read(45,900) anything
      nerror=955
      read(anything(41:),*,err=3000) QE(1)
      write(*,955) QE(1)
      write(46,955) QE(1)
955   format('External Q factor                       =',1pe11.4)
      do 63 js=1,modes
        QE(js)=QE(1)/mode_coupling(js)
63    continue

      read(45,900) anything
      nerror=956
      read(anything(41:),*,err=3000) ROQ
      write(*,956) ROQ
      write(46,956) ROQ
956   format('Cavity R over Q   (2xFNAL=53 per cell)  =',f11.3,' ohms')

      read(45,900) anything
      nerror=957
      read(anything(41:),*,err=3000) energy_set_point
      value=energy_set_point*1000.0d00
      write(*,957) value
      write(46,957) value
957   format('Energy point  ILC crab~0.0284J per cell)=',f11.3,' mJ')

      do 52 js=1,modes
        v_set_point(js)=0.0d00
52    continue

      v_set_point(1)=(aa*wd/c)*sqrt(energy_set_point*wd*ROQ)
      value=v_set_point(1)/1000.0d00

      write(*,958) value
      write(46,958) value
958   format('Amplitude set point                     =',f11.3,' kV')

      read(45,900) anything
      nerror=959
      read(anything(41:),*,err=3000) Max_power
      write(*,959) Max_power
      write(46,959) Max_power
959   format('Maximum Amplifier Power per cell        =',f11.3,' W')

      drive_max= sqrt(2.0d00*QE(1)*ROQ*Max_power)
c****** drive_max is the maximum amplitude of the forward wave ********

      V_max_point=2.0d00*drive_max
      value=V_max_point/1000.0d00
      write(*,960) value
      write(46,960) value
960   format('Maximum voltage set point (no beam)     =',f11.3,' kV')

      read(45,900) anything
      nerror=961
      read(anything(41:),*,err=3000) offset
      write(*,961) offset
      write(46,961) offset
961   format('Maximum beam offset                     =',f11.3,' mm')

      read(45,900) anything
      nerror=962
      read(anything(41:),*,err=3000) bunch_phase_err
      write(*,962) bunch_phase_err
      write(46,962) bunch_phase_err
962   format('Maximum bunch phase error               =',f11.3,' deg')

      read(45,900) anything
      nerror=963
      read(anything(41:),*,err=3000) xrandom

      lrandom=.false.
```

```
      if(xrandom.gt.0.5d00) lrandom=.true.
      if(lrandom)then
        read(45,900) anything
c       this line contains a offset period which is not used
        offset_freq=0.0d00
        write(*,963)
        write(46,963)
963     format('Random offset')
      else
        read(45,900) anything
        nerror=964
        read(anything(41:),*,err=3000) offset_freq
        write(*,964) offset_freq
        write(46,964) offset_freq
964     format('Beam offset frequency                   =',
     +  f11.3,' Hz')
      end if
      w_offset=2.0d00*pi*offset_freq

      read(45,900) anything
      nerror=965
      read(anything(41:),*,err=3000)  bunch_charge
      value=bunch_charge*1.0d09
      write(*,965)  value
      write(46,965)  value
965   format('Bunch charge (ILC=3.2 nC)               =',f11.3,' nC')
      cos_err=cos(bunch_phase_err*pi/180.0d00)
      sin_err=sin(bunch_phase_err*pi/180.0d00)
      vjump=0.5d00*(aa*wd/c)*(offset*1.0d-03*wd/c)*wd*ROQ*bunch_charge
      write(*,966) vjump
966   format('Voltage jump at t=0                     =', 1pe11.4,' V')

      read(45,900) anything
      nerror=967
      read(anything(41:),*,err=3000) ibeam
      write(*,967) ibeam
      write(46,967) ibeam
967   format('RF cycles between bunches               =', i11)

      read(45,900) anything
      nerror=969
      read(anything(41:),*,err=3000) pulse_length
      value=pulse_length*1000.0d00
      write(*,969) value
      write(46,969) value
969   format('Bunch train length                      =',f11.3,' ms')

      read(45,900) anything
      nerror=970
      read(anything(41:),*,err=3000) cavity_freq_shift
      write(*,970) cavity_freq_shift
      write(46,970) cavity_freq_shift
970   format('Cavity frequency shift from microphonics=',f11.3,' Hz')
      wshift=2.0d00*pi*cavity_freq_shift

      read(45,900) anything
      nerror=971
      read(anything(41:),*,err=3000) cavity_vib_freq
      write(*,971) cavity_vib_freq
      write(46,971) cavity_vib_freq
971   format('Cavity vibration frequency              =',f11.3,' Hz')
      wcav=2.0d00*pi*cavity_vib_freq

      read(45,900) anything
      nerror=972
      read(anything(41:),*,err=3000) initial_vib_phase
      write(*,972) initial_vib_phase
      write(46,972) initial_vib_phase
972   format('Initial vibration phase (degrees)       =',f11.3,' deg')
      vph=initial_vib_phase*pi/180.0d00
```

```
      read(45,900) anything
      nerror=973
      read(anything(41:),*,err=3000) meas_phase_jitter_deg
      write(*,973) meas_phase_jitter_deg
      write(46,973) meas_phase_jitter_deg
973   format('Phase measurement error(degrees)        =',f11.5,' deg')
      meas_phase_jitter=meas_phase_jitter_deg*2.0d00*pi/360.00d00

      read(45,900) anything
      nerror=974
      read(anything(41:),*,err=3000) meas_amp_jitter
      write(*,974) meas_amp_jitter
      write(46,974) meas_amp_jitter
974   format('Fractional err in amplitude measurement =',f11.5)

      read(45,900) anything
      nerror=968
      read(anything(41:),*,err=3000) cntr_delay
      idelay=cntr_delay*fd
      if(idelay.gt.jic) idelay=jic
      if(idelay.lt.1) idelay=1
      write(*,968) cntr_delay
      write(46,968) cntr_delay
968   format('Time delay (latency) for control system =',1pe11.4,' s')

      read(45,900) anything
      nerror=975
      read(anything(41:),*,err=3000) update_interval
      write(*,975) update_interval
      write(46,975) update_interval
975   format('Control update interval                 =',1pe11.4,' s')

      control_update=fd*update_interval

      read(45,900) anything
      nerror=9766
      read(anything(41:),*,err=3000) gain
      write(*,9766) gain
      write(46,9766) gain
9766  format('Gain constant for controller            =',f9.4)

      read(45,900) anything
      nerror=9767
      read(anything(41:),*,err=3000) amp_bandwidth
      write(*,9767) amp_bandwidth
      write(46,9767) amp_bandwidth
9767  format('Amplifier bandwidth                     =',1pe11.4)

      n_vec_mod=20
c     number of RF cyles over which the vector modulator ramps to a new value

      QA=fd/amp_bandwidth
c     Amplifier Q factor

c     PI Controller coefficients
c     **************************
      cnt_param=2
      if(cnt_param.eq.1)then
        c_prop_ar= 250.00d-8*QE(1)
        c_intl_ar=   0.01d-8*QE(1)
        c_prop_ai= 250.00d-8*QE(1)
        c_intl_ai=   0.01d-8*QE(1)
      else
c       increased values for Ai
        c_prop_Ar= 500.00d-8*QE(1)*gain
        c_intl_Ar=   0.02d-8*QE(1)*gain
        c_prop_Ai=2000.00d-8*QE(1)*gain
        c_intl_Ai=   0.16d-8*QE(1)*gain
      endif

      write(46,976) c_prop_ar
```

```
976   format('Proportional coef for real component    =',1pe11.4)
      write(46,977) c_intl_ar
977   format('Integral coef for real component        =',1pe11.4)
      write(46,978) c_prop_ai
978   format('Proportional coef for imag component    =',1pe11.4)
      write(46,979) c_intl_ai
979   format('Integral coef for imag component        =',1pe11.4)

      close(unit=45,status='keep')


      write(*,*) '     '

      icycle=0

      cycles_per_train=1.0e-3*fd
c     number of cycle in a bunch train = 1ms * frequency

      if(QE(1).lt.3.0d06)then
        nbeam_on=2.0d00*QE(1)
      else
        nbeam_on=1.5d00*QE(1)
      end if
      if(nbeam_on.gt.10000000)then
        nbeam_on=10000000
c       limit size for plotting in Excel
      end if
      nbeam_on=4500000
      nbeam_off=nbeam_on+cycles_per_train
      nsettle=0.05d00*cycles_per_train
      ncycle=nbeam_off+0.5d00*cycles_per_train
      ndetail=0.5d00*(nbeam_on+nbeam_off)
c     cycle when next bunch is due

      its_per_cycle=1
c     time iterations per cycle

      period=2.0d00*pi/wd

      tstart=0.0d00
      dt=period/its_per_cycle
      n_iterations=its_per_cycle*ncycle

      open(file='results_os.txt',unit=40,status='modify')
      open(file='wave_os.txt',unit=41,status='modify')

      t0=tstart
      amplitude(1) =0.0d00
      vdr = 0.0d00
      vdi = 0.0d00
      time_step = 1.0d00

      write(40,930)
930   format(9x,'cycle',10x,'time',5x,'amplitude',6x,'phase',
     +       11x,'Rcontrol',6x,'Icontrol',6x,'DrvPower',6x,'DrvPhase')

      write(41,931)
931   format(8x,'time',5x,'    field')

      hdt=dt*0.5d00
      dt6=dt/6.0d00

      Qmeas=dt/update_interval
c     filter parameter for measurement model

      do 31 js=1,modes
       AR(js) = 0.0d00
       AI(js) = 0.0d00
31    continue
      amp_AR = 0.0d00
      amp_AI = 0.0d00
```

```
      AR0_filter=0.0d00
      AI0_filter=0.0d00

      nprint=0
      nwrite=1

      sumr=0.0d00
      sumi=0.0d00
      summax=drive_max/c_intl_Ar

      cycle=0.0d00
c     when cycle gets to one the number of cycles=icycle increments

      do 10 ic=1,idelay
        driver(ic)=0.0d00
        drivei(ic)=0.0d00
10    continue

      nbeam=nbeam_on
      max_power_used=0.0d00
      normalise=0

      do 70 js=1,modes
        sum_sq_ph_err(js)=0.0d00
        sum_sq_amp_err(js)=0.0d00
70    continue

      ic=2
      ic_delay=1
      count=control_update

c     evaluate amplifier coefficients
c     *****************************
      wa=wd
      ga1=(2.0d00*wd/wa)**2+(1.0d00/QA)**2
      ga2=((wd/wa)**2+1.0d00)*(1.0d00/QA)
      ga3=((1.0d00/QA)**2-(2.0d00*wd/wa)*(wa/wd-wd/wa))*(wd/wa)
      ga4=1.0d00/(QA*QA)
      ga5=2.0d00*(wd/wa)*(1.0d00/QA)

c********** the coefficients of ga4 and ga5 need checking               ***********
c********** and also the QA of the amplifier with respect to QE and QL  ***********

      fa1=wa*ga2/ga1
      fa2=wa*ga3/ga1
      fa3=ga4/ga1
      fa4=ga5/ga1

      do 1 j=1,n_iterations

        t2=t0+dt

c       set instantaneous cavity frequency and evaluate cavity coefficients
c       *******************************************************************
        do 30 js=1,modes
          wc(js)=wc0(js)+wshift*sin(wcav*t0+vph)

          f0(js)=wc(js)/(4.0d00*QC(js))
          fe(js)=wc(js)/(4.0d00*QE(js))

          g1(js)=(1.0d00+(wc(js)/wd)**2)
          g2(js)=(wc(js)**2-wd**2)/(2.0d00*wd)
          g3(js)=wc(js)/(QE(js)*wd)
30       continue


c  Measurement Model
c  *****************
c  Note that the actual measurement model deteriorates at the flash ADC combination points
c  Still need to put this in.
```

```
c  Low Pass Filter to average cavity fields over period between DSP re-calculating the control action
c  ****************************************************************************************************
c  There will be some sort of low pass filter between the cavity probe and the device that measures
c  amplitude and phase. If the device that measures amplitude and phase delivers its output by means of
c  an ADC to a DSP then it is sensible to chose the low pass cut-off equal to the sample rate.
C  The next few lines averages over modes and implements a first order low pass filter.

        AR_sum=0.0d00
        AI_sum=0.0d00
        do 73 js=1,modes
          AR_sum=AR_sum+AR(js)*mode_coupling(js)
          AI_sum=AI_sum+AI(js)*mode_coupling(js)
73      continue

        AR0_filter=AR0_filter*(1.0d00-Qmeas)+AR_sum*Qmeas
        AI0_filter=AI0_filter*(1.0d00-Qmeas)+AI_sum*Qmeas


        if(count.ge.control_update)then
c    measurement errors depend on bandwidth. The measurement errors that should be inserted are those
c    appropriate to the control update frequency.

          count=1
          rand_phase=meas_phase_jitter*(2.0d00*random()-1.0d00)
          rand_mag=1.0d00 + meas_amp_jitter*(2.0d00*random()-1.0d00)
          mag_factor=rand_mag/sqrt(1.0d00+rand_phase**2)
          AR0_meas=mag_factor*(AR0_filter+rand_phase*AI0_filter)
          AI0_meas=mag_factor*(AI0_filter-rand_phase*AR0_filter)

c         PI Control
c         **********
          vdr_last=vdr
          vdi_last=vdi
          time_last=t0
          Rerror=V_set_point(1)-AR0_meas
          Ierror=-AI0_meas
          sumr=sumr+fd*Rerror*dt*control_update
          sumi=sumi+fd*Ierror*dt*control_update
          if(sumr.gt.summax)  sumr=summax
          if(sumr.lt.-summax) sumr=-summax
          if(sumi.gt.summax)  sumi=summax
          if(sumi.lt.-summax) sumi=-summax

          Vdr=c_prop_Ar*Rerror+c_intl_Ar*sumr
          Vdi=c_prop_Ai*Ierror+c_intl_Ai*sumi

c         limit drive output
c         ******************
          drive_amp=sqrt(Vdr**2+Vdi**2)
          if(drive_amp.gt.drive_max)then
            Vdr=vdr*drive_max/drive_amp
            Vdi=vdi*drive_max/drive_amp
          end if

c         evaluate driver terms in differential equations
c         ***********************************************
          time_step1=n_vec_mod*dt
          time_step2=t0-time_last
          if(time_step1.lt.time_step2)then
            time_step=time_step1
          else
            time_step=time_step2
          end if
          if(time_step.le.1.0e-12) time_step=1.0d00
          vdr_dot=(vdr-vdr_last)/time_step
          vdi_dot=(vdi-vdi_last)/time_step

c         evaluate driver while change is being made and after completion
c         ***************************************************************
c         everytime the DSP re-calculates the drive, the vector modulator
c         ramps the drive to the new value.
```

```
          dr_ramp=(fa3*(vdr_dot+wd*Vdi)-fa4*(vdi_dot-wd*Vdr))
          di_ramp=(fa3*(vdi_dot-wd*Vdr)+fa4*(vdr_dot+wd*Vdi))
          dr_flat=( fa3*wd*Vdi+fa4*wd*Vdr)
          di_flat=(-fa3*wd*Vdr+fa4*wd*Vdi)
        end if

c       evaluate drive - this stays the same if the controller has not recalculated
c       ***************************************************************************
        if(count.le.20)then
          driver(ic_delay)=dr_ramp
          drivei(ic_delay)=di_ramp
        else
          driver(ic_delay)=dr_flat
          drivei(ic_delay)=di_flat
        endif
c       ********************************************************
c       *  The drive action is delayed by the processing time  *
c       ********************************************************

        count=count+1

c       amplifier model
c       ***************
c       The amplifier is modelled as another cavity hence one needs to solve a
c       second set of envelope functions.


c       4th order Runge Kutta integration for amplifier
c       ***********************************************
        amp_AR0=amp_AR
        amp_AI0=amp_AI

        call RK(t0,hdt,dt6,amp_AR,amp_AI,fa1,fa2,driver(ic),drivei(ic))

        testr=amp_AR-vdr
        testi=amp_AI-vdi

        amp_AR_dot=(amp_AR-amp_AR0)/dt
        amp_AI_dot=(amp_AI-amp_AI0)/dt

        do 32 js=1,modes
          outr(js)=-g3(js)*(amp_AI_dot-wd*amp_AR)
          outi(js)= g3(js)*(amp_AR_dot+wd*amp_AI)
32      continue


c       4th order Runge Kutta integration for cavity
c       ********************************************
        call RKM(t0,hdt,dt6,AR,AI,f0,fe,g1,g2,outr,outi,modes)

        do 51 js=1,modes
          amplitude(js) = sqrt(AR(js)*AR(js)+AI(js)*AI(js))
          if(abs(AR(js)).gt.1.0d-9)then
            phase(js) = atan(AI(js)/AR(js))
          else
            phase(js)=0.5d00*pi
          end if
          phase_deg(js)=phase(js)*180.0d00/pi
          if((j.gt.nbeam_on+nsettle).and.(j.lt.nbeam_off))then
            sum_sq_ph_err(js)=sum_sq_ph_err(js)+phase_deg(js)**2
            amp_err(js)=amplitude(js)-v_set_point(js)
            sum_sq_amp_err(js)=sum_sq_amp_err(js)+amp_err(js)**2
            normalise=normalise+1
          endif
51      continue

        time = t2*1.0d09

        ic=ic+1
        ic_delay=ic_delay+1
        if(ic_delay.eq.idelay+1) ic_delay=1
```

```
        if(ic.eq.idelay+1) ic=1

c       output data to files and screen
c       ******************************
        if(j.gt.nprint)then
          write(*,901) time, amplitude(1), phase_deg(1), vdr, vdi
901       format('t=',f11.3,' amp=',f11.1,
     +           ' ph=', e11.4,' Rcntr=', f10.1,' Icntr=', f9.1)
          nprint=nprint+100000*its_per_cycle
        end if

        if(j.gt.nwrite)then
          drive_amp=sqrt(Vdr**2+Vdi**2)
          drive_phase=180.0d00*asin(Vdi/drive_amp)/pi
          drive_pow=0.5d00*drive_amp**2/(QE(1)*ROQ)
          write(40,902) icycle, time, amplitude(1), phase_deg(1),
     +    vdr, vdi,drive_pow,drive_phase
902       format(2x,i12,3x,1pe11.4,3x,1pe11.4,3x,e13.6,3x,1pe11.4,
     +          3x,1pe11.4,3x,1pe11.4,3x,1pe11.4)

          if(icycle.gt.nbeam_on)then
           if(drive_pow.gt.max_power_used) max_power_used=drive_pow
          end if

          if((icycle.gt.ndetail).and.(icycle.lt.ndetail+20000))then
            nwrite=nwrite+20*its_per_cycle
          elseif(icycle.lt.nbeam_on)then
            nwrite=nwrite+2000*its_per_cycle
          elseif((icycle.gt.nbeam_on).and.
     +           (icycle.lt.(nbeam_on+400000)))then
            nwrite=nwrite+100*its_per_cycle
          else
            nwrite=nwrite+2000*its_per_cycle
          end if
        end if

        if((j.gt.0).and.(j.lt.5000))then
          time = t2*1.0d09
          write(41,903) time, AR(1), testr, testi
903       format(3x,f11.3,3x,f11.5,3x,1pe11.5,3x,1pe11.5)
        end if

c       Provision for more than one time step per cycle
c       ***********************************************
        cycle=cycle+1.0d00/its_per_cycle
        if(cycle.gt.0.9999d00)then
          icycle=icycle+cycle*1.00001d00
          cycle=0.0d00
        end if

c       Beam loading of fundamental mode only
c       *************************************
        if(icycle.gt.nbeam)then
         if((icycle.gt.nbeam_on).and.(icycle.le.nbeam_off))then
            bunch_phase=bunch_phase_err*pi/180.0d00
            Vkick=Vjump*cos(w_offset*t2)
            if(lrandom)then
              bunch_phase=bunch_phase*random( )
              Vkick=Vjump*random( )
            end if
            cos_err=cos(bunch_phase)
            sin_err=sin(bunch_phase)
            AR(1)=AR(1)-Vkick*cos_err**2
            AI(1)=AI(1)-Vkick*sin_err*cos_err
            nbeam=nbeam+ibeam
          endif
        endif

        t0=t2
1     continue
```

```
      write(*,9977) max_power_used
      write(46,9977) max_power_used
9977  format('maximum power delivered                 =', f9.2)

      value=sqrt(sum_sq_ph_err(1)/normalise)
      write(*,9978) value
      write(46,9978) value
9978  format('In pulse rms phase err                  =', f10.5,
     +       ' degrees')

      value=100.0d00*sqrt(sum_sq_amp_err(1)/normalise)/v_set_point(1)
      write(*,9979) value
      write(46,9979) value
9979  format('In pulse rms amplitude err              =', f10.5,' %')

      value=100.0d00*sqrt(sum_sq_amp_err(2)/normalise)/v_set_point(1)
      write(*,9980) value
      write(46,9980) value
9980  format('Relative excitation of 2nd mode         =', f10.5,' %')

      value=100.0d00*sqrt(sum_sq_amp_err(3)/normalise)/v_set_point(1)
      write(*,9981) value
      write(46,9981) value
9981  format('Relative excitation of 3rd mode         =', f10.5,' %')

      close(unit=40,status='keep')
      close(unit=41,status='keep')
      close(unit=46,status='keep')

1000  stop

3000  print*,'error ',nerror,' reading value'
      go to 1001

1001  close(unit=45,status='keep')
      close(unit=46,status='keep')
      go to 1000
      end

      subroutine RK(t0,hdt,dt6,AR0,AI0,f1,f2,outr,outi)
c     *************************************************
      real*8 AR0, AR1, AR2, AR3, AR4, AI0, AI1, AI2, AI3, AI4
      real*8 DAR1, DAR2, DAR3, DAR4, DAI1, DAI2, DAI3, DAI4
      real*8 f1, f2, t0, t1, t2, dt6, hdt, outr, outi

      t1  = t0+hdt
      DAR1=-f1*AR0-f2*AI0+outr
      AR1 = AR0+hdt*DAR1
      DAI1=-f1*AI0+f2*AR0+outi
      AI1 = AI0+hdt*DAI1

      DAR2=-f1*AR1-f2*AI1+outr
      AR2 = AR0+hdt*DAR2
      DAI2=-f1*AI1+f2*AR1+outi
      AI2 = AI0+hdt*DAI2

      DAR3=-f1*AR2-f2*AI2+outr
      AR3 = AR0+2.0d00*hdt*DAR3
      DAI3=-f1*AI2+f2*AR2+outi
      AI3 = AI0+2.0d00*hdt*DAI3

      t2 = t1+hdt
      DAR4=-f1*AR3-f2*AI3+outr
      AR4 = AR0+dt6*(DAR1+2.0d00*(DAR2+DAR3)+DAR4)
      DAI4=-f1*AI3+f2*AR3+outi
      AI4 = AI0+dt6*(DAI1+2.0d00*(DAI2+DAI3)+DAI4)

      AR0=AR4
      AI0=AI4

      return
```

```
      end

      subroutine RKM(t0,hdt,dt6,AR0,AI0,f0,fe,g1,g2,outr,outi,modes)
c     **************************************************************
      integer*4 jm
      parameter(jm=3)

      real*8 AR0(jm), AR1(jm), AR2(jm), AR3(jm), AR4(jm)
      reAL*8 AI0(jm), AI1(jm), AI2(jm), AI3(jm), AI4(jm)
      real*8 DAR1(jm), DAR2(jm), DAR3(jm), DAR4(jm)
      real*8 DAI1(jm), DAI2(jm), DAI3(jm), DAI4(jm)
      real*8 f0(jm), fe(jm), g1(jm), g2(jm)
      real*8 outr(jm), outi(jm)
      real*8 t0, t1, t2, dt6, hdt, sum

      integer*4 j1, j2, modes

      t1      = t0+hdt

      do 11 j1=1, modes
         sum=0.0d00
         do 12 j2=1, modes
            sum=sum+g1(j2)*AR0(j2)
12       continue
         sum=f0(j1)*g1(j1)*AR0(j1)+fe(j1)*sum
         DAR1(j1)= - sum + g2(j1)*AI0(j1) + outr(j1)
         AR1(j1) =   AR0(j1) + hdt*DAR1(j1)

         sum=0.0d00
         do 13 j2=1, modes
            sum=sum+g1(j2)*AI0(j2)
13       continue
         sum=f0(j1)*g1(j1)*AI0(j1)+fe(j1)*sum
         DAI1(j1)= - sum - g2(j1)*AR0(j1) + outi(j1)
         AI1(j1) =   AI0(j1)+hdt*DAI1(j1)
11    continue

      do 21 j1=1, modes
         sum=0.0d00
         do 22 j2=1, modes
            sum=sum+g1(j2)*AR1(j2)
22       continue
         sum=f0(j1)*g1(j1)*AR1(j1)+fe(j1)*sum
         DAR2(j1)= - sum + g2(j1)*AI1(j1) + outr(j1)
         AR2(j1) =   AR0(j1) + hdt*DAR2(j1)

         sum=0.0d00
         do 23 j2=1, modes
            sum=sum+g1(j2)*AI1(j2)
23       continue
         sum=f0(j1)*g1(j1)*AI1(j1)+fe(j1)*sum
         DAI2(j1)= - sum - g2(j1)*AR1(j1) + outi(j1)
         AI2(j1) =   AI0(j1)+hdt*DAI2(j1)
21    continue

      do 31 j1=1, modes
         sum=0.0d00
         do 32 j2=1, modes
            sum=sum+g1(j2)*AR2(j2)
32       continue
         sum=f0(j1)*g1(j1)*AR2(j1)+fe(j1)*sum
         DAR3(j1)= - sum + g2(j1)*AI2(j1) + outr(j1)
         AR3(j1) =   AR0(j1) + 2.0d00*hdt*DAR3(j1)

         sum=0.0d00
         do 33 j2=1, modes
            sum=sum+g1(j2)*AI2(j2)
33       continue
         sum=f0(j1)*g1(j1)*AI2(j1)+fe(j1)*sum
         DAI3(j1)= - sum - g2(j1)*AR2(j1) + outi(j1)
         AI3(j1) =   AI0(j1) + 2.0d00*hdt*DAI3(j1)
```

```
31    continue

      t2 = t1+hdt

      do 41 j1=1, modes
         sum=0.0d00
         do 42 j2=1, modes
            sum=sum+g1(j2)*AR3(j2)
42       continue
         sum=f0(j1)*g1(j1)*AR3(j1)+fe(j1)*sum
         DAR4(j1)= - sum + g2(j1)*AI3(j1) + outr(j1)
         AR4(j1) = AR0(j1) +
     +            dt6*(DAR1(j1)+2.0d00*(DAR2(j1)+DAR3(j1))+DAR4(j1))

         sum=0.0d00
         do 43 j2=1, modes
            sum=sum+g1(j2)*AI3(j2)
43       continue
         sum=f0(j1)*g1(j1)*AI3(j1)+fe(j1)*sum
         DAI4(j1)= - sum - g2(j1)*AR3(j1) + outi(j1)
         AI4(j1) = AI0(j1) +
     +            dt6*(DAI1(j1)+2.0d00*(DAI2(j1)+DAI3(j1))+DAI4(j1))
41    continue

      do 5 j1=1, modes
        AR0(j1)=AR4(j1)
        AI0(j1)=AI4(j1)
5     continue

      return
      end
```