File SFCODES2.DOC contains the following bookmarks. Select the topic you are trying to find from the list and double click the highlighted text.

File SFCODES2.DOC Table of Contents

XIII. Automated Tuning Programs

The Poisson Superfish code distribution includes several programs for automatically tuning accelerating cavities. The programs include DTLfish, CCLfish, CDTfish, ELLfish, MDTfish, RFQfish, and SCCfish. Since these codes share many features with one another, this section uses an integrated approach to the tuning-code documentation. Separate subsections for each program describe any significant differences among the codes, such as in the cavity shapes, control-file keywords, etc.

Each tuning program sets up input files for the codes Automesh, Fish, and SFO (collectively known as Superfish). These codes are included as subroutines within the main tuning program. A tuning program runs Superfish repetitively, varying the cavity geometry to tune each cavity to a specified frequency. The control files can define large numbers of problems that will run unattended.

A. The tuning programs and their cavity shapes

This section discusses the types of problems solved by each tuning program. It includes one or more figures that define all the geometrical parameters you can use to specify the cavity shape. A special utility program FScale scales parameters in the tuning program control files for a new rf frequency.

1. DTLfish: a tuning program for drift-tube linacs

DTLfish sets up the geometry for drift-tube linac (DTL) cells. The DTL cell is a figure of revolution about the beam axis. DTLfish assumes a symmetric cell, and therefore sets up Superfish runs for only half the cell. The symmetry plane is in the gap center between the two drift-tube noses. The control file defines up to 100 problems. DTLfish tunes the cell by adjusting either the cavity diameter, drift-tube diameter, gap, or face angle.

a. DTLfish cavity shape

Figure XIII-1 shows the outline of the right half of a DTL cell. Figure XIII-2 shows more detail near the drift-tube nose. The lower left corner is the center of the cell. The full gap is g and the full length is L. The bore radius is R_b. The full cavity diameter is D and the drift-tube diameter is d. The face angle α_f is the angle that the drift-tube face makes with the vertical. Note the difference between the face angle in Figure XIII-2 and the cone angle α_c (see Figure XIII-4) used in coupled-cavity linac problems. The cone angle is the angle that the nose makes with the horizontal. There are three circular arcs on the drift-tube profile. The corner radius R_c connects the outer end of the straight face-angle segment with the straight segment at the drift-tube diameter. The inner-nose radius R_i connects the drift-tube bore to an optional vertical flat segment of length F. The outer-nose radius R_o connects the flat segment on the nose with the face-angle segment.

Programs CCLfish, MDTfish, and CDTfish use the same parameters for the drift-tube nose shown in Figure XIII-2. In CDTfish the cavity-wall nose can be different from the nose on the internal drift tubes. Thus, CDTfish uses symbols R_{di}, R_{do}, and F_d for the internal drift tubes to distinguish them from the parameters describing the cavity nose.

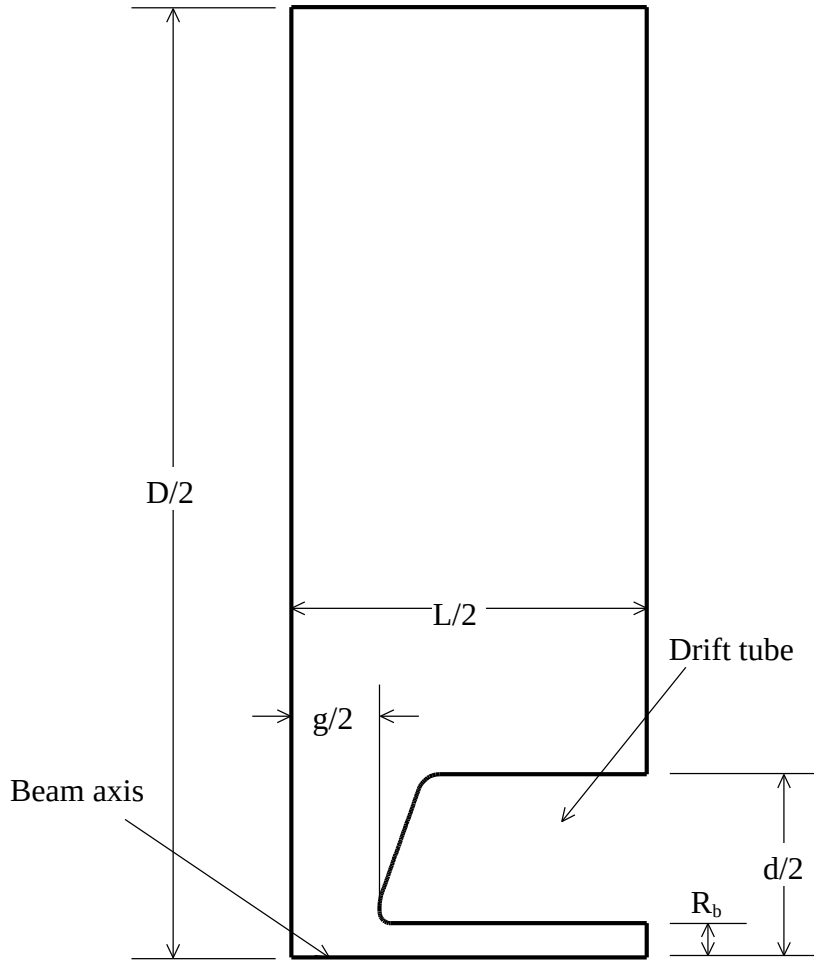

Figure XIII-1. The DTL half cell set up by the code DTLfish.
The cell is a figure of revolution about the beam axis. at the bottom of the figure. The left edge is a symmetry plane. Figure XIII-2 show more details near the drift-tube nose.

b. DTLCells, companion program for generating each Parmila cell

Program DTLCells is a companion program for DTLfish that creates a new input file for DTLfish containing cell geometries interpolated from a set of representative cells. In addition, DTLCells creates two text files containing dimensions needed for construction of the drift tubes. One file lists parameters with linear dimensions in centimeters and the other uses inches.

DTLCells reads the same input file that DTLfish reads. The code assumes that the DTL half cells have already been tuned by an appropriate DTLfish session (all start codes should be negative). The code reads the tuned geometry from the original DTLfish control file and then linearly interpolates geometric parameters that vary as a function of the cell length (or particle velocity βc since the cell length is nβλ, where n is an integer). The problems in the original file must be sequential in order of increasing cell length.

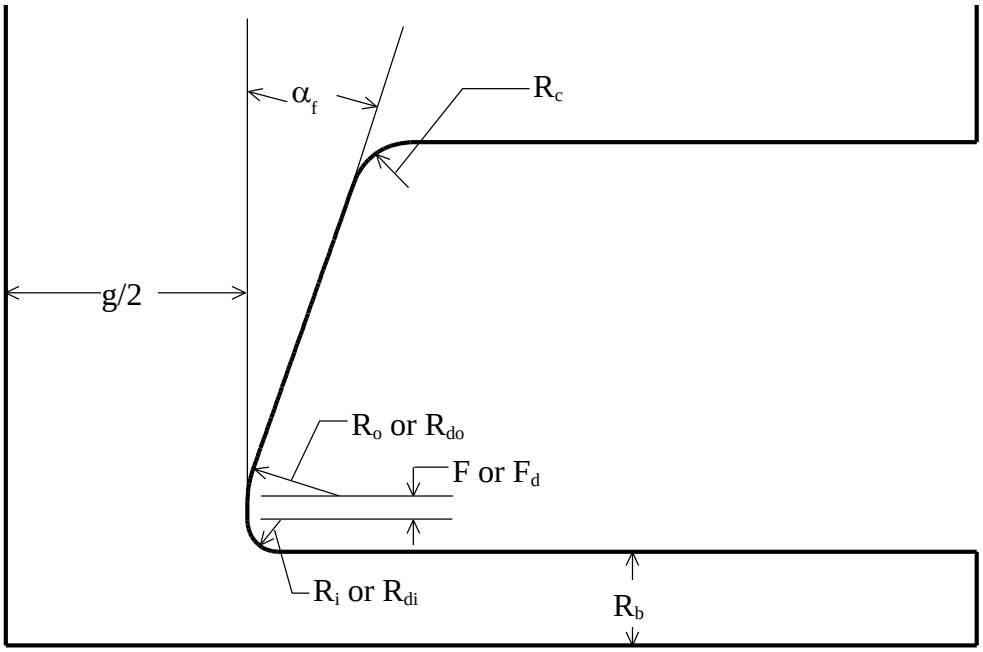

Figure XIII-2. Detail near the drift-tube nose.
This drift-tube nose is from the DTL cell shown in Figure XIII-1. Programs DTLfish, CCLfish, MDTfish, and CDTfish use similar parameters to describe the drift-tube nose.

Program DTLCells uses the setting for the InitialEnergy in the DTLfish control file, which specifies the starting energy for a DTL tank. Although DTLfish does not use the InitialEnergy keyword, it will duplicate its setting in new control files for completed jobs.

The InitialEnergy setting refers to data in the output file Parmila.OUT created by the Parmila design code. The keyword indicates the starting energy of a tank on the "initial" line just before the list of geometric parameters by cell number. An "initial" line appears at the start of both commonly used tables of parameters in Parmila.OUT. One table is titled "linout subroutine 1" and lists the dynamics parameters for the design. The other table is titled "linout subroutine 4" and lists geometric data needed for construction. The DTLCells program requires "linout subroutine 4," the geometric data. After finding this table with the correct starting energy, the code continues reading cell lines until reaching the end of the table. An error occurs if the representative data does not span a large enough range of particle velocities to cover the range in the Parmila.OUT table.

DTLCells counts the number of "linout subroutine 4" tables that it encounters until finding the one with the correct starting energy. The code sets the tank number parameter to this that count number, and includes the tank number first on every line of output. This feature allows you to later merge multiple files into a single file that defines the drift-tube geometry in every tank of a multi-tank DTL. The names of the output text files include the tank number. Thus, for the cm-dimension files, T1Geo_cm.TXT contains data for tank 1, T2Geo_cm.TXT for tank 2, and so on.

Table XIII-1. Column headings in drift-tube geometry table.

Heading	Description
Tank#	Tank number.
DT#	Drift-tube number (cell number for CellLen and GapLen columns).
CellLen	Length of the cell than includes the upstream half of this drift tube.
Zstem	Longitudinal position of the drift-tube center from the upstream end wall rf surface.
FA_1	Face angle on the upstream face of the drift tube.
FA_2	Face angle on the downstream face of the drift tube.
Flat_1	Vertical flat length on the upstream face of the drift tube.
Flat_2	Vertical flat length on the downstream face of the drift tube.
Rin_1	Inner nose radius on the upstream face of the drift tube.
Rin_2	Inner nose radius on the downstream face of the drift tube.
Rout_1	Outer nose radius on the upstream face of the drift tube.
Rout_2	Outer nose radius on the downstream face of the drift tube.
DT_Len	Length of the drift tube.
Rb_1	Bore radius in the upstream half of the drift tube.
Rb_2	Bore radius in the downstream half of the drift tube.
Rc_1	Corner radius on the upstream end of the drift tube.
Rc_2	Corner radius on the downstream end of the drift tube.
Diam	Diameter of the drift tube.
GapLen	Length of the gap just upstream of this drift tube.
GL	The magnetic GL product for the quadrupole magnet in the drift tube.

A drift-tube geometry table contains 20 columns of data (see Table XIII-1) that completely define the shape of each drift tube in the tank, including the two half drift tubes mounted on the end walls. Drift tube 0 refers to the half drift tube on the upstream end wall, and, for a tank with N cells, drift tube N refers to the half drift tube on the downstream end wall. If M is an arbitrary cell number ranging from 1 to N, then drift tube M follows gap M, which is assumed to be contained within cell M. Drift tube stems are centered longitudinally on the drift tubes. The zero for longitudinal position is at the upstream end wall rf surface. The output file whose name ends with "_cm" will contain linear dimensions in centimeters. The file whose name ends with "_in" with contain linear dimensions in inches. In both output files, the face angles are in degrees from vertical, and the GL product is in Tesla. A zero value for the GL product means that the drift tube does not contain a quadrupole magnet. The algebraic sign of GL defines the orientation of the magnet for positively charged particles: positive GL focuses in the x (horizontal) direction and defocuses in y, negative GL focuses in the y (vertical) direction and defocuses in x.

2. CCLfish: a tuning program for coupled-cavity linacs

CCLfish sets up the geometry for coupled-cavity linac (CCL) cells. The CCL cell is a figure of revolution about the beam axis. CCLfish assumes a symmetric cell, and therefore sets up Superfish runs for only half the cell. The symmetry plane is in the gap center between the two noses. The control file defines up to 100 problems for a particular *Shape* (defined by control file keyword CAVITY_shape) from Table XIII-2. CCLfish tunes the cell by adjusting either the cavity diameter, septum thickness, gap, or cone angle. This program also will tune buncher cavities and disk-loaded waveguide cells.

Table XIII-2. Cavity types in CCLfish.

Shape	Description
0	Standard coupled-cavity linac cell shown in Figure XIII-3.
1	Disk-loaded waveguide cavity shown in Figure XIII-7.
2	Cavity with special end wall shape shown in Figure XIII-8.

For coupled-cavity cells, engineering considerations usually dictate the septum thickness. It is probably not a good idea to vary it to tune a CCL cell. You can vary the septum thickness to tune a stand-alone buncher cavity. CCLfish does not allow a negative septum thickness because it would imply that the outer part of the cavity extends beyond the bore-tube extension in the longitudinal direction. This situation usually results in an unphysical boundary condition on the right side of the cavity. If you vary the septum thickness to tune a cell, CCLfish will discontinue the problem if the septum thickness becomes zero or negative before the next Superfish run.

a. Shape 0, standard coupled-cavity linac cell

Figure XIII-3 shows the outline of the right half of a "standard" coupled-cavity linac cell. Figure XIII-4 shows more detail near the nose. The lower left corner is the center of the cell. The full gap is g and the full length is L. The bore radius is R_b. The full cavity diameter is D and the full septum thickness is s.

The CCLfish program uses a control-file entry for the outer corner radius R_{co}, but not for the equator flat F_{Eq}. The code computes the value of F_{Eq} after completing a solution and it reports its value in a comment line in the control file for completed jobs. When using program CDTfish for a full-cell calculation of a CCL end cavity (usually with a bore-tube extension), you can use the value of the equator flat supplied by CCLfish. The CDTfish code computes the outer corner radius from the equator flat entry.

The cone angle α_c is the angle that the nose face makes with the horizontal. There are three circular arcs on the drift-tube profile. The cavity inner corner radius R_{ci} connects the outer end of the straight cone-angle segment with the vertical segment along the septum. The inner nose radius R_i connects the bore tube to an optional vertical flat segment of length F. The outer nose radius R_o connects the flat segment on the nose with the cone-angle segment.

b. Optional tuning surfaces for standard coupled-cavity linac cell

CCLfish provides two types of optional tuning surfaces, which we call tuning rings. An "equator tuning ring" is a bump on the outer cavity wall at the midplane or equator of the cavity. A "wall tuning ring" is a raised bump (or an indentation) on the end wall of the cavity, usually just below the outer corner radius. For either type of tuning ring, you can either define all of the geometrical parameters for the feature, or you can request a particular frequency effect Δf_{Ring}, in which case the code determines the ring thickness. The CDTfish code includes the same optional features.

Figure XIII-5 defines the geometrical parameters for a equator tuning ring in the CCL cavity and Figure XIII-6 shows the geometry for a wall tuning ring. Control file keyword RING_TYPE sets the value of *Ring* (see Table XIII-3), which defines the type of tuning

ring and the method for determining the ring thickness. If *Ring* = 0, no tuning ring will appear in the cavity regardless of the settings for other tuning ring parameters. If *Ring* = 1, 2, 3, or 4, then the control file must contain nonzero values for at least the width W_{Ring} and the chamfer angle α_{Ring}. For *Ring* = 1 or 3, specify a nonzero value for the thickness T_{Ring}; and for *Ring* = 2 or 3, specify a nonzero value for the frequency effect Δf_{Ring}.

Table XIII-3. Tuning ring options in CCLfish.

Ring	Description
0	No tuning ring.
1	Equator ring of a given thickness.
2	Equator ring of a given frequency effect.
3	Wall rings of a given thickness.
4	Wall rings of a given frequency effect.

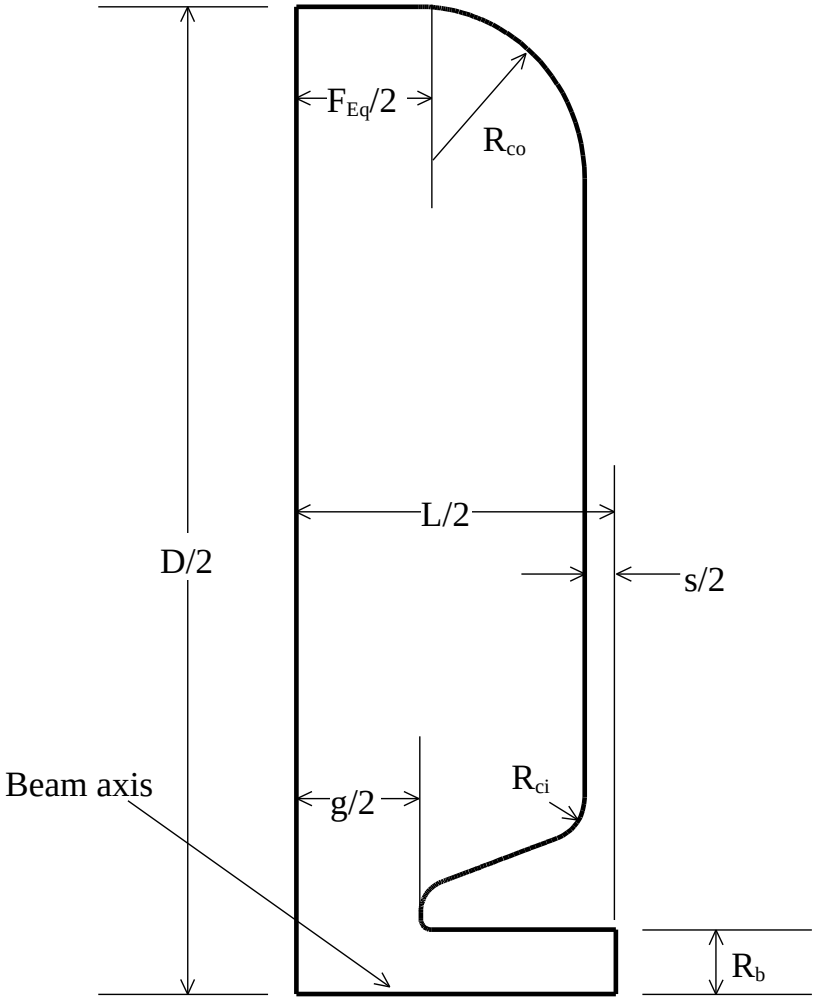

Figure XIII-3. The CCL half cell set up by the code CCLfish.
The cell is a figure of revolution about the beam axis, which is at the bottom of the figure. The left edge is a symmetry plane. Figure XIII-4 show more details near the nose.

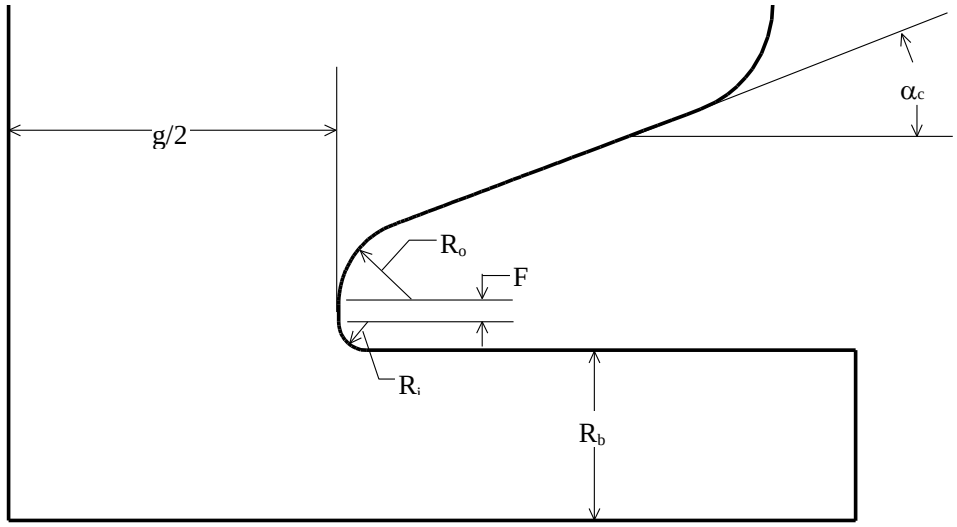

Figure XIII-4. Detail near the nose in a CCL cell.
This nose is from the CCL cell shown in Figure XIII-3. Programs DTLfish, CCLfish, MDTfish, and CDTfish all use the same parameters for the drift-tube nose.

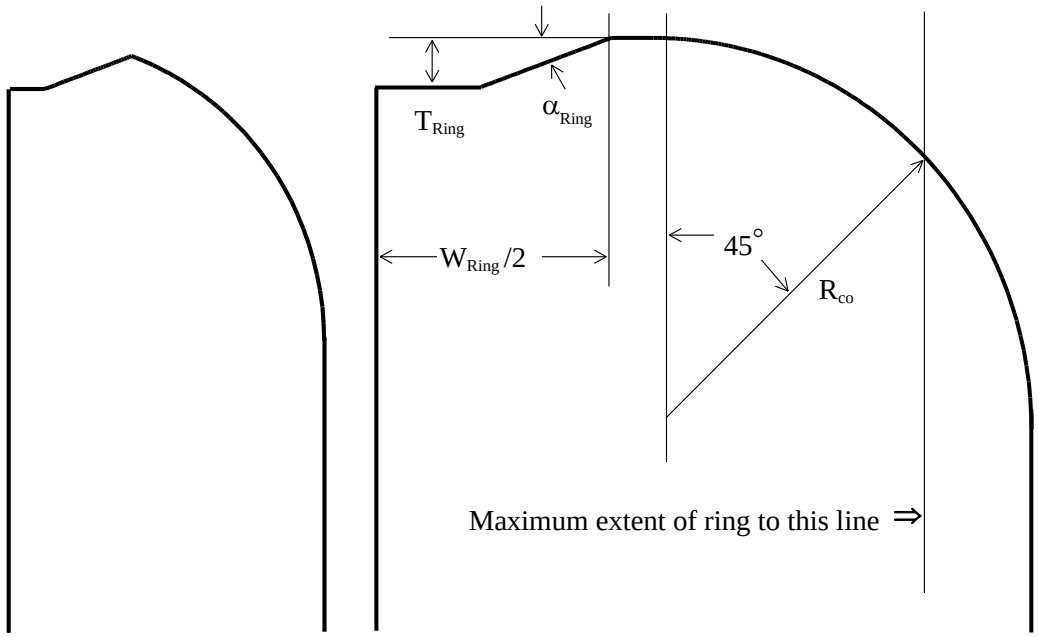

Figure XIII-5. Optional equator tuning ring in CCL cavities.
Parameter *Ring* = 1 or 2 selects an equator ring. In cavities designed for low-velocity particles (left), the ring would typically intersect the outer corner radius. In longer cavities (right), the ring may not reach to the corner arc and instead intersect the flat segment at the cavity wall.

For a specified ring thickness T_{Ring} CCLfish tunes the cavity with the ring feature already included in the geometry. After finishing this problem, the code creates another Automesh input file with the ring removed from the cavity to determine the actual

frequency effect of the ring. By default, the code will delete the extra files (xxx.AM, xxx.SEG, xxx.SFO, xxx.PMI, and xxx.T35) generated by this procedure. If you wish to save these files, set DeleteNoRingFiles = No in file SF.INI. The extra files will have letter "A" appended to the original filenames.

A table of initial estimates of the frequency effect for various ring thickness appears in the log file. This table is based upon Slater perturbation calculations for the completed cavity design. The estimates are usually accurate to a few percent. The actual tuning ring effect Δf_{Ring} (from the additional Superfish run) appears in the control file of completed jobs.

For a specified tuning effect Δf_{Ring} (instead of a thickness), CCLfish subtracts Δf_{Ring} from the target frequency to arrive at an intermediate target frequency for a cavity that has no tuning ring. The program then tunes that cavity to the intermediate target frequency. After finishing this part of the problem, the code uses the Slater perturbation method to estimate of the frequency effect for several values of the ring thickness. A table in the log file lists these estimates up to a maximum thickness (see below). Interpolation from this table provides the starting point for the first Superfish run with the ring in the geometry. After one or two more iterations, the cavity frequency converges to the original target frequency. The final ring thickness that tuned the cavity to the desired frequency appears in the control file of completed jobs created by the code. If the control file has a nonzero setting for the ring thickness, the code will adopt it for the initial thickness if the value is within 50% of the code's thickness estimate.

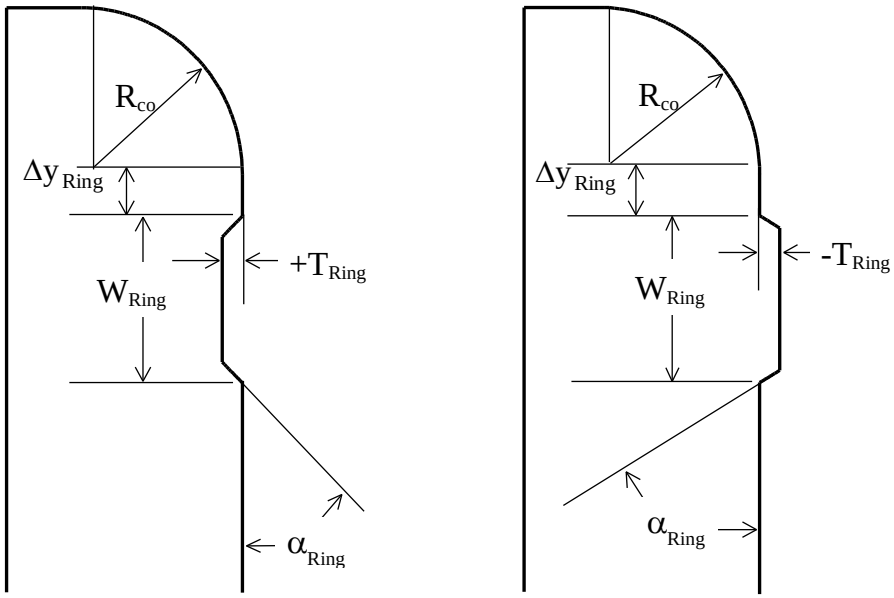

Figure XIII-6. Optional wall tuning rings in CCL and CCDTL cavities. Parameter *Ring* = 3 or 4 selects a wall ring. A bump on the cavity wall corresponds to positive thickness and an indentation corresponds to negative thickness. In both cases, a zero angle implies no tuning ring.

The code imposes a few restrictions on tuning ring parameters in order to keep the problem manageable. Bear in mind that the goal of the tuning ring feature is to provide

just a few MHz of tuning range, typically less than 1% of the operating frequency. This range is sufficient to compensate for the frequency effects of machining tolerances and for uncertainties in frequency estimates of 3-dimensional effects such as coupling slots. The restrictions are as follows:

- The maximum thickness of an equator tuning ring is limited to a fraction of the full cavity length. (This length does not include the septum thickness.) The default fraction is 15%, but it may be set to any value between 5% and 50% using variable EquatorRingLimit in file SF.INI.
- The maximum positive thickness of each ring on the cavity end walls is limited to a fraction of the half cavity length. The default fraction is 15%, but it may be set to any value between 5% and 50% using variable WallRingLimit in file SF.INI.
- A negative thickness wall tuning ring cannot protrude more than halfway into the half septum. For example, if s = 1.0cm, then $T_{Ring} \geq -0.25$ cm.
- The width of an equator ring must be less than the distance between the 45-degree points on the outer corner arcs.
- Wall tuning rings (including the height offset Δy_{Ring}) must fit between the outer corner radius and the inner corner radius.
- The angle α_{Ring} cannot exceed 90 degrees. (If $\alpha_{Ring} = 0$ there is no ring; if $\alpha_{Ring} = 90$ degrees, the ring has no chamfer.)
- The setting *Ring* = 2 or 4 cannot be used with START code 1. To design a tuning ring of a given frequency effect the cavity must first be tuned without the ring.

The tuning rings shown in several figures for the CCL and CCDTL are from some work on 805-MHz cavities at Los Alamos in 1999. Figure XIII-5, gives the dimensions of some equator tuning rings that produce rather large tuning effects. For the CCDTL, we were trying to maintain a larger cavity diameter at the coupling slots on the ends of the cavity (not shown in the figures) than would be possible without an equator tuning ring. Thus, a large portion of the tuning ring will remain in the CCDTL cavity after tuning.

For the CCL, we designed the cavities with a tuning-ring frequency effect of 2.5 MHz. For short CCL cavities ($\beta \sim 0.4$), an equator tuning can interfere with the coupling slot, so the wall tuning ring is a better choice. If a cavity has an equator ring, the expectation is that after coarse tuning (before brazing) about half of the initial thickness will remain in most cavities. The 0.5-cm thickness shown in Figure XIII-5, which gives an effect over 6 MHz, is therefore ~2.5 times the actual thickness required in the design. If a cavity has a wall tuning ring, the target frequency could correspond to a flat surface provided that sufficient tuning is available by machining an indentation in the wall and that the indentation does not interfere with any coolant passages in the septum.

The angle of the chamfer depends mainly on the manufacturing process. The CCL half cells require a shallow angle on equator rings to allow a lathe cutting tool access deeper in the cell than the extent of the ring. The CCDTL cavities can be made of three longitudinal pieces brazed together, so tool access to the equator ring is not an issue.

Table XIII-4. Equator tuning ring effects in three cavities.

Cavity property	CCDTL at $\beta = 0.29$	CCL at $\beta = 0.40$	CCL at $\beta = 0.80$
Length L (cm)	16.20	7.45	14.90
Diameter D (cm)	22.75	26.00	26.00
Ring thickness (cm)	0.50	0.50	0.50
Ring width (cm)	6.70	2.50	5.00
Ring angle (degrees)	30.0	20.0	20.0
Ring effect (MHz)	10.0	6.47	6.65

c. Shape 1, disk-loaded waveguide cell

Figure XIII-7 shows a disk-loaded waveguide commonly used for high-energy electron accelerators. This cavity includes no nose near the bore. To select this shape the keyword CAVITY_shape = 1 must appear in the control file. The code ignores settings for the inner corner radius R_{ci}, the cone angle α_c, and the drift-tube nose parameters R_i, R_o, and F. The septum-thickness parameter s specifies the full width of the annular disks between cells. The inner radius of the disk has a circular cross section of radius s/2. The outer corner radius is optional and may be omitted by setting $R_{co} = 0$. CCLfish will tune this cavity by varying only the diameter D.

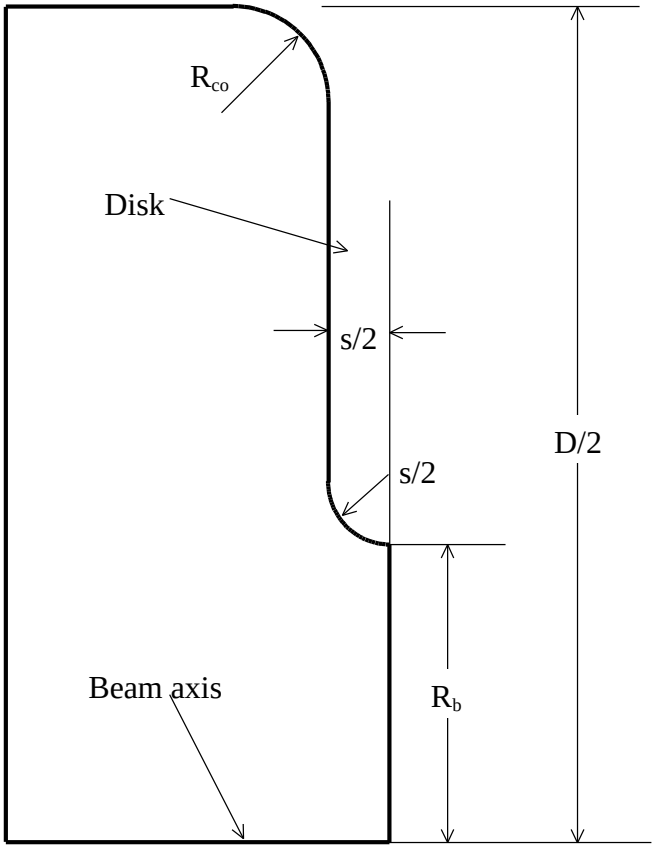

Figure XIII-7. A section of a disk-loaded waveguide.
The left edge is a symmetry plane with a Neumann boundary condition. The right edge has a Dirichlet boundary condition.

d. Shape 2, special end-wall shape

Figure XIII-8 shows a shape option in CCLfish that one might use in certain special applications, such as buncher cavities. To select this shape the keyword CAVITY_shape = 2 must appear in the control file. Enter values for the end-wall radius of curvature R_W and the height H_W above the beam axis of the center of curvature. The rest of the details of this geometry use the same geometrical parameters as the standard CCL cell. For this type of cell there is no vertical wall segment. The entire end wall is an arc that connects the nose cone to the cavity outer wall. The arc will not be tangent to either connecting segment.

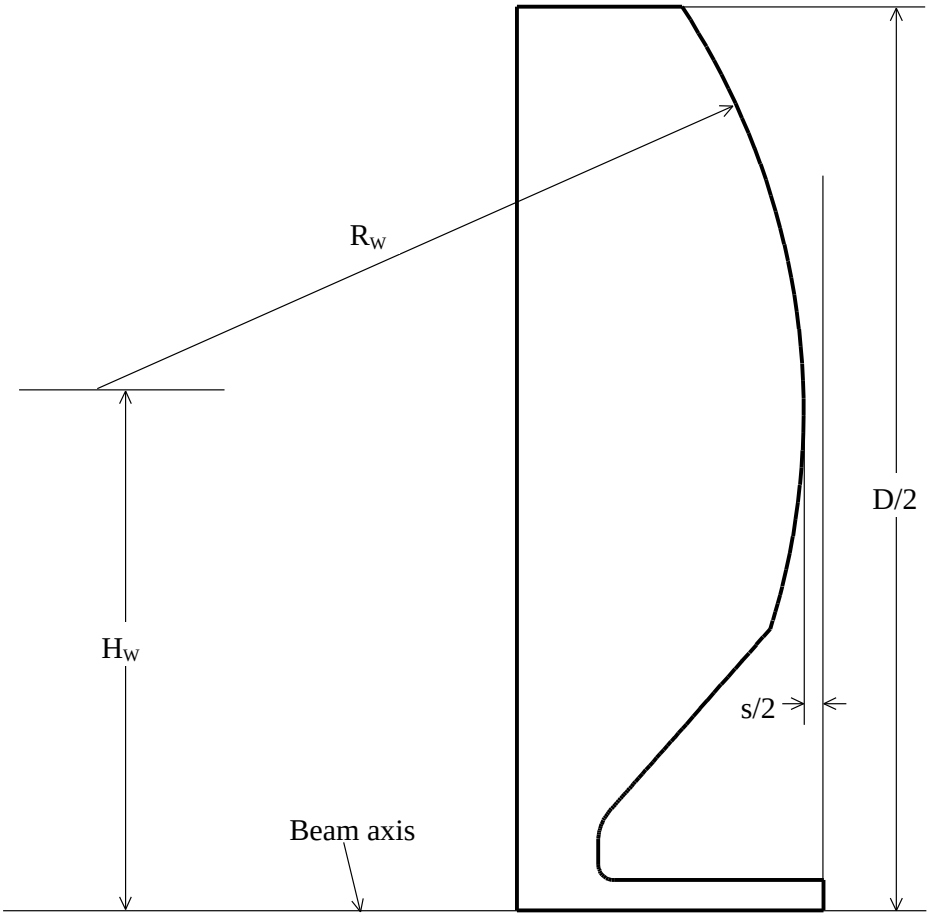

Figure XIII-8. A CCLfish cavity with special shape parameters.
The special shape parameters are R_W, the wall radius of curvature, and H_W, the height the arc center above the beam axis.

e. CCLCells, companion program for generating each Parmila cell

Program CCLCells is a companion program for CCLfish that creates a new input file for CCLfish containing cell geometries interpolated from a set of representative cells.

CCLCells reads the same input file that CCLfish reads. The code assumes that the CCL half cells have already been tuned by an appropriate CCLfish session (all start codes should be negative). The code reads the tuned geometry from the original CCLfish

control file and then linearly interpolates geometric parameters that vary as a function of the cell length (or particle velocity βc since the cell length is nβλ/2, where n is an integer). The problems in the original file must be sequential in order of increasing cell length.

Program CCLCells uses the setting for keywords InitialEnergy and FinalEnergy in the CCLfish control file. Keyword InitialEnergy specifies the starting energy for the rf structure, and FinalEnergy specifies a maximum kinetic energy above which no more cells will be generated. Although CCLfish does not use these keywords, it will duplicate their settings in new control files for completed jobs.

The InitialEnergy setting refers to data in the output file Parmila.OUT created by the Parmila design code. The keyword indicates the starting energy of a CCL structure on the "initial" line just before the list of parameters by cell number. This "initial" line appears at the start of the Parmila.OUT table titled "linout subroutine 1." The table lists the dynamics parameters for the design including the kinetic energy at the end of the cell, the cell length, and the design axial field E_0. These three parameters are the only ones used by CCLCells. Once program CCLCells finds one of the table with the correct starting energy, the code continues reading cell lines until reaching the end of the table or finding a line with a higher energy than the FinalEnergy setting. In a typical CCL design, several lines will have the same cell length. CCLCells will generate only one problem for a given cell length. An error occurs if the representative data does not span a large enough range of particle velocities to cover the range in the Parmila.OUT table.

3. ELLfish: a tuning program for elliptical cavities

ELLfish sets up the geometry for so-called elliptical cavities, which are often used in superconducting applications. These cavities feature an elliptical segment near the bore radius. The elliptical cavity is a figure of revolution about the beam axis. ELLfish can tune either a symmetric half cell or an asymmetric full cell that includes an attached bore tube. The control file defines up to 50 problems. ELLfish tunes the cell by adjusting either the cavity diameter, outer radius of curvature, or the angle that the straight side of the cavity makes with the vertical.

a. Symmetric half cavity

Figure XIII-9 shows a cross section of the right half of an elliptical cavity. This symmetric half-cavity calculation tunes the interior cavities of a superconducting multicell cavity. The lower left corner is the center of the cell. The left edge is the cavity midplane and the bottom edge is the beam axis. For the half-cavity calculation, the boundary conditions are Neumann at the left edge and Dirichlet at the right edge.

The bore radius is R_b, the cavity diameter is D. The full cavity length is $L = \beta_g\lambda/2$, where the subscript g stand for "geometrical." Thus, the problem geometry length for a half-cavity is L/2. The cavity wall includes a flat segment of length F_{Eq} at the equator, an elliptical dome with semiaxes b_D and a_D, a sloping straight segment at angle α_w from the vertical, another ellipse near the iris with semiaxes b_I and a_I, and a flat segment of length F_I at the iris. The flat segments at the equator and at the iris are optional (i.e., they can have zero length). The circular and elliptical arcs both connect tangent to the sloping line segment. The ELLfish program uses control-file entries for the dome vertical semiaxis b_D,

the dome-ellipse aspect ratio a_D/b_D, the iris-ellipse aspect ratio a_I/b_I, and the tangent angle α_w. The code computes the values of a_D, a_I, and b_I as well as the angles θ_D and θ_I of lines from the centers of the ellipses to the intersections with the sloping line. To create a circular dome, set b_D equal to the desired radius and specify $a_D/b_D = 1$.

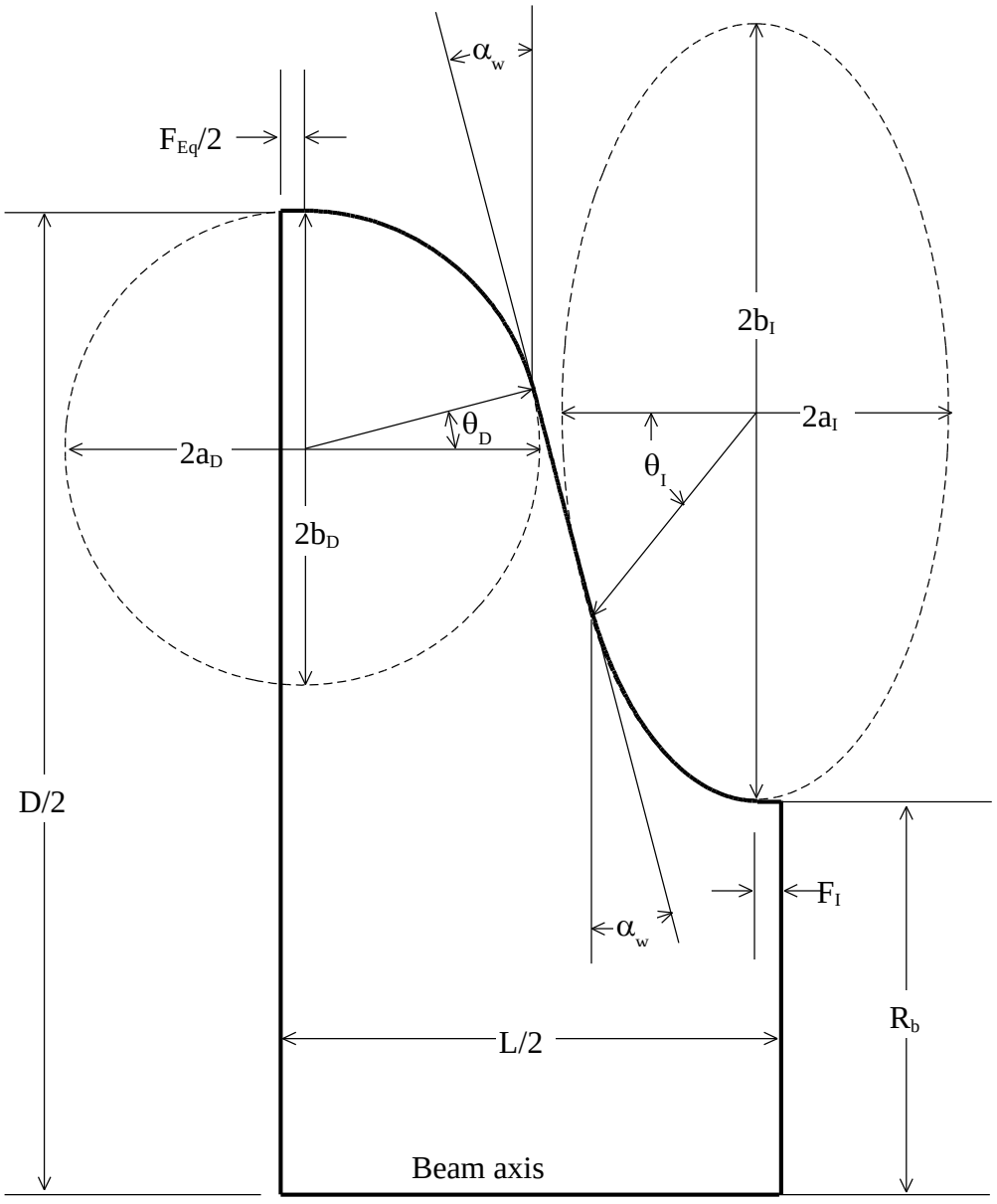

Figure XIII-9. The right half of an elliptical cavity.
The left edge is a symmetry plane with a Neumann boundary condition. The right edge has a Dirichlet boundary condition.

b. Full cavity with attached bore tube

Figure XIII-10 shows a full-cell version of the cavity from Figure XIII-9. The designer uses this type geometry to tune the end cell of a multicell structure. All of the geometric parameters on the right side of the cavity may have different values from the corresponding parameters on the left side. The bore tube of radius $R_{b,R}$ extends a distance δT_R beyond the nominal right edge of the cavity. If necessary, the geometry can include a second bore tube extension of length δT_2 and radius $R_{b,2}$. The cavity length is $L = \beta\lambda/2$, so the full length of the problem geometry is $L + \delta T_R + \delta T_2$. The code uses a Dirichlet boundary condition on the left and right edges of the problem geometry. On the right side, the bore tube should be long enough so the fields fall to negligible values at the edge of the geometry. Neither Dirichlet nor Neumann boundaries would be appropriate at this edge. Since these are the only choices in Superfish, the code must use one of them. The bore tube is long enough if the fields and resonant frequency do not change appreciably if you manually change the setting of NBSRT = 0 to NBSRT = 1 in the Automesh input file and rerun the problem.

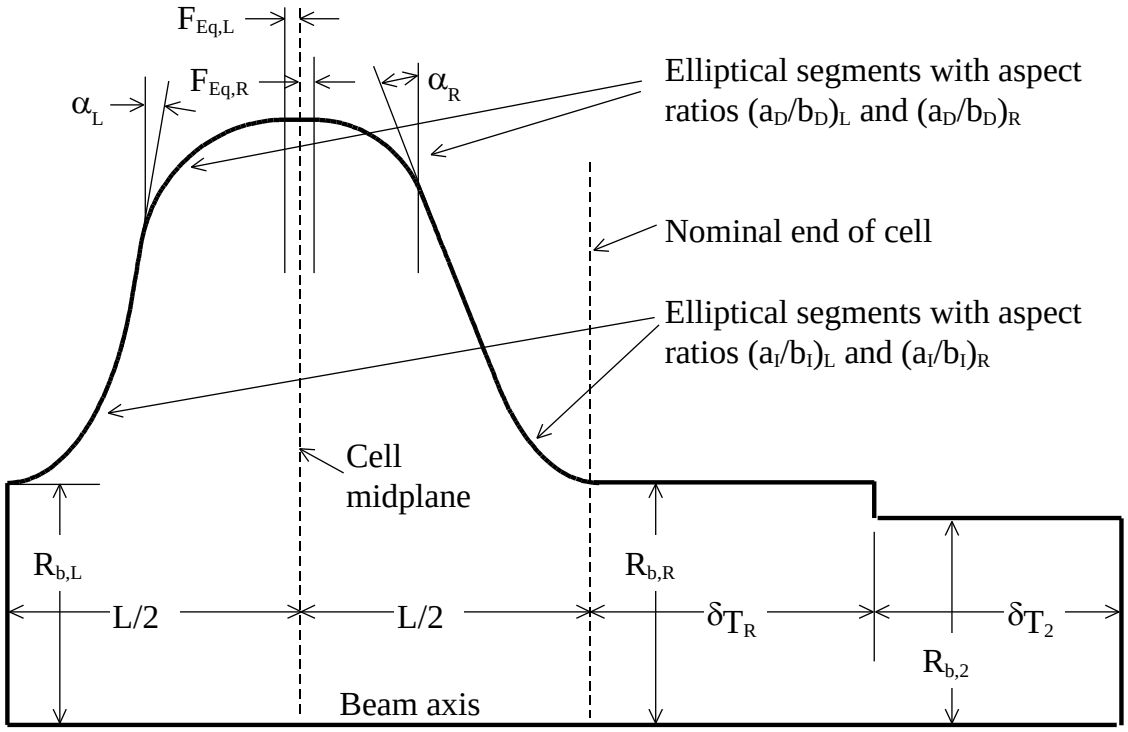

Figure XIII-10. Full-cell elliptic cavity.
The shape of the cavity wall on left and right can be different. The ELLfish program can tune full cavities by adjusting the right-side radius or the right-side wall angle.

The only common parameters on left and right sides are the cavity diameter D and the half lengths L/2. As shown in Figure XIII-10, all other parameters, including the ellipse aspect ratios, can have different values on the left and right sides. For the iris ellipse, even if $(a_I/b_I)_L = (a_I/b_I)_R$, the actual size of the two ellipse segments may be different depending on choices for the other geometrical parameters. The wall angle is α_L on the left and α_R on the right; the dome ellipse parameters are $b_{D,L}$ and $(a_D/b_D)_L$ on the left and $b_{D,R}$ and $(a_D/b_D)_R$ on the right; and the equator flat is divided into $F_{Eq,L}$ on the left and $F_{Eq,R}$ on the

right. If only a value for F_{Eq} appears in the control file, then $F_{Eq,L} = F_{Eq,R} = F_{Eq}/2$. The iris flat, which appears in Figure XIII-9, but not in Figure XIII-10, is $F_{I,L}$ on the left and $F_{I,R}$ on the right. The flat segments at the equator and iris are optional (i.e., they can have zero length).

Often, an electric antenna on the bore tube just outside the end cell of a multicell cavity supplies the rf power. ELLfish allows different values for the bore radius on left and right sides of the end cell. The designer can find the bore radius of the internal cells that provides the desired cell-to-cell coupling and independently adjust the external bore radius for adequate coupling to the rf power supply. Using a smaller second bore radius $R_{b,2}$ can reduce the size of quadrupole magnets between cavities.

Program ELLfish can tune the full-cell cavity by adjusting either the outer diameter D, the dome size $b_{D,R}$ (length of vertical semi-axis), or the right-side wall angle α_R. The usual procedure is to allow ELLfish to tune the interior half cell by adjusting the cavity diameter for chosen settings of the other geometrical parameters. For the end cell, the designer fixes the geometry of the left side at the values found for interior cells and tunes the cell by adjusting the shape of the right-side cavity wall. The recommended method is to tune the full cavity by varying the right-side dome size $b_{D,R}$.

c. ELLCAV, companion program for generating the multicell-cavity file

After having optimized and tuned both the internal and external cells of a multicell structure, you can run program ELLCAV to generate an Autofish input file for half of the multicell cavity. This tool facilitates computing the higher-order longitudinal modes of the cavity from which you can compute the cell-to-cell coupling. Another use of the ELLCAV-generated input file is the creation of a family of PMI files for supplying transit-time-factor data to Parmila.

ELLCAV reads the same input file that ELLfish reads. The code assumes that the internal cell and the external cell have already been tuned by appropriate ELLfish sessions. The code reads the resulting tuned geometry from an ELLfish control file. ELLCAV uses the NumberOfCells keyword in the ELLfish control file, which specifies N_{Cells}, the total number of cells in the cavity. If N_{Cells} is even, the Autofish input file will contain $N_{Cells}/2$ cells; if N_{Cells} is odd, it will contain $N_{Cells}/2 + 1$ cells, where the first cell is a half cell. Although ELLfish does not use the NumberOfCells keyword, it does duplicate its setting in new control files for completed jobs.

The initial Automesh input file written by ELLCAV has the left-edge boundary condition appropriate for the π mode. In order to find all N_{Cells} resonant modes for the TM_{010} band it will be necessary to run Autofish first with one, then the other, boundary condition at the left edge. For example, consider the 8-cell structure depicted in Figure XIII-11. The left edge is the middle of the cavity. The top view shows the 805-MHz π mode of this cavity, which corresponds to a Dirichlet boundary condition at the left edge. The bottom view shows the 790-MHz zero mode.

The remaining six modes of the TM_{010} band lie between these two frequencies. If you do not know the mode frequencies accurately enough to start a resonance search on each mode, we suggest performing a frequency scan over the range of interest.

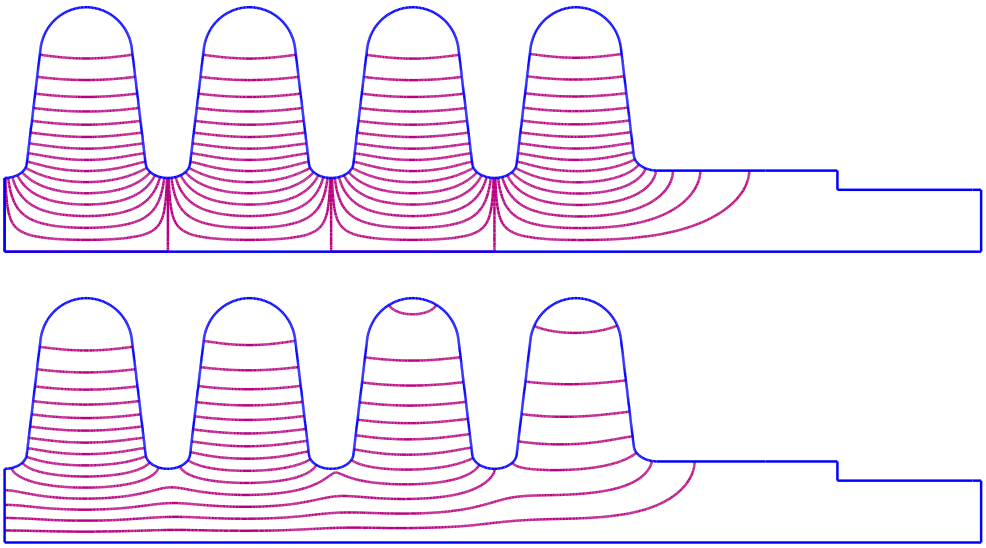

Figure XIII-11. Right half of an 8-cell cavity.
The π mode (top) has the setting NBSLF = 0 for a Dirichlet boundary condition and the 0 mode (bottom) has NBSLF = 1 for a Neumann boundary condition.

4. MDTfish: a tuning program for multiple-cell drift-tube linacs

MDTfish sets up the geometry for entire drift-tube linac (DTL) tanks or multiple-cell sections of a DTL. The DTL cell is a figure of revolution about the beam axis. The control file defines up to 20 problems. MDTfish tunes the cavity either by adjusting the cavity diameter, or by proportionally adjusting the gap in all cells. You can include the effects of stems and post couplers in the power and frequency calculations done by SFO.

a. MDTfish cavity shape

Figure XIII-12 shows an example problem with three cells. The dashed lines show the boundaries between cells. (When viewing this on-line some details may not show up because of the screen resolution.) Numbers in parentheses refer to the cell number. The details near the drift-tube noses are the same as in Figure XIII-2 for a DTLfish half cell. The drift-tube face angles, inner and outer nose radii, and flat lengths between nose radii can all be different on each side of a gap. Figure XIII-12 shows face angles α_L and α_R only for the first gap. The full gap is g and the full length is L. R_b is the bore radius. The full cavity diameter is D and the drift-tube diameter is d for all the cells.

One of the goals in developing the MDTfish code was to find ways to retune the end of a DTL tank to repair a particular type of frequency error. To preserve properties of a high quality beam, a DTL designer may adjust the synchronous phase of the cells on both ends of the tank to make up for the missing gap(s) between tanks. More negative values of the synchronous phase provide more longitudinal focusing at the expense of some loss in acceleration. One way to tune the end is to move the end wall. The example in Figure XIII-12 shows a shift δW_L of the tank's left end wall. The end wall shifts δW_L and δW_R can be positive or negative. A positive value of δW_L moves the end wall into the tank. A

positive value of δW_R on the other end would moves the end wall away from the tank. The left and right beam-tube lengths can only be positive numbers. This example shows a left beam-tube of length δT_L

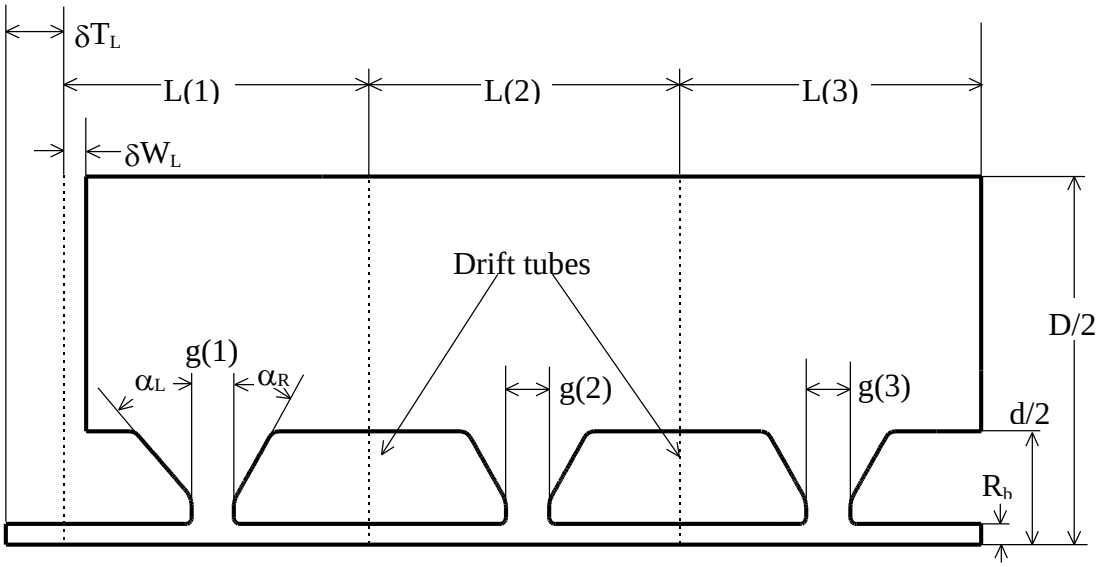

Figure XIII-12. Three full cells of a drift-tube linac setup by MDTfish. Numbers in parentheses refer to cell numbers. This example shows different face angles in cell 1 and a shift of the tank's left end wall.

b. MDTCells, companion program, converts DTLfish input to MDTfish input

Program MDTCells reads a DTLfish input file and creates a MDTfish input file containing all the cells found in the DTLfish file.

5. CDTfish: a tuning program for coupled-cavity drift-tube linacs

CDTfish sets up the geometry for coupled-cavity drift-tube linac (CCDTL) cavity. The CCDTL cavity is a figure of revolution about the beam axis. It resembles a CCL cavity, but with one or more internal drift tubes. The control file defines up to 50 problems.

CDTfish sets up either full cavities or half cavities with optional extensions of the bore tube on either side. It tunes the cavity by adjusting either the cavity diameter or the drift-tube gaps. To correct for longitudinal asymmetries in the gap fields, the code can use information from the SFO transit-time-factor integrals to align the cell's electrical centers. This adjustment consists of shifting the gap locations by an amount $\delta g(i)$ until the electrical center coincides with the desired cell centers. The user also can fix the values of $\delta g(i)$ and tune the cavity without the automatic adjustment. This program also will tune ordinary CCL cells (without internal drift tubes).

a. Coupled-cavity drift-tube linac cavity shapes

Figure XIII-13 shows a full cavity of a one-drift-tube (2-gap) CCDTL. This particular example includes the optional bore-tube extension δT_R on the right-hand side of the cavity. Thus, this geometry simulates the end cavity of a structure consisting of two or more one-drift-tube cavities between drift spaces. Refer to Figure XIII-2 for details of the

drift-tube-nose shape. The nose on the cavity wall is the same as in a CCLfish problem. The equator flat F_{Eq} and septum thickness s determine the cavity's outer corner radius R_{co}. The control file does not include an entry for R_{co}.

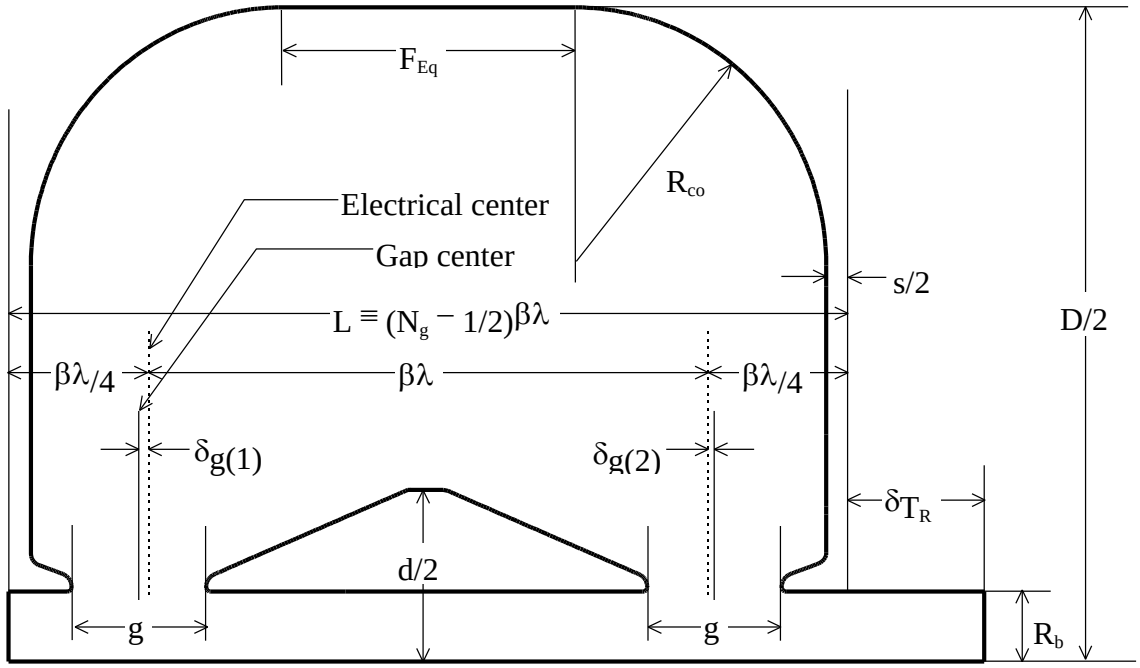

Figure XIII-13. Full cavity of a one-drift-tube, 2-gap CCDTL.
Program CDTfish defines the nominal cavity length L in terms of a half integral multiple of βλ (3βλ/2 in this example). The full geometry is longer by an amount equal to the bore-tube extension. Both drift-tube gaps are exactly the same length. The code optionally moves the gap centers to make the cell's electrical centers βλ apart and βλ/4 from the nominal edges of the cavity.

The full length of the problem geometry is $L + \delta T_L + \delta T_R$, where L is defined in terms of the number of gaps N_g and βλ. In this figure $\delta T_L = 0$. The code ignores any bore-tube extension δT_L on the left side of the cavity for half-cavity problems. For a half cavity, the left edge of the problem is in the center of the cavity and has a Neumann boundary condition. For a full cavity, the code applies a Dirichlet boundary condition on the left edge of the bore hole. If the problem includes a bore-tube extension on either end of the cavity, the bore tube should be long enough so the fields fall to negligible values within the bore-tube length. The boundary condition at these edges is always set to Dirichlet, even though neither Dirichlet nor Neumann boundaries would be appropriate. Since these are the only choices in Superfish, the code must use one of them. The bore tube is long enough if the fields and resonant frequency do not change appreciably if you manually change the setting of NBSRT = 0 to NBSRT = 1 in the Automesh input file and rerun the problem.

Figure XIII-14 shows a half cavity for a two-drift-tube (3-gap) CCDTL. More detail near the drift tube appears in Figure XIII-15. For symmetric cavities with an odd number of gaps, the electrical center of the middle gap is always in the exact center of the cavity. For half-cavity calculations CDTfish assumes a symmetric shift of the end gaps. This

example also includes the optional bore-tube extension δT_R on the right-hand side of the cavity. Thus, this geometry simulates a structure consisting of individual two-drift-tube cavities separated by drift spaces. To model an interior cavity in a chain of cavities, set $\delta T_R = 0$.

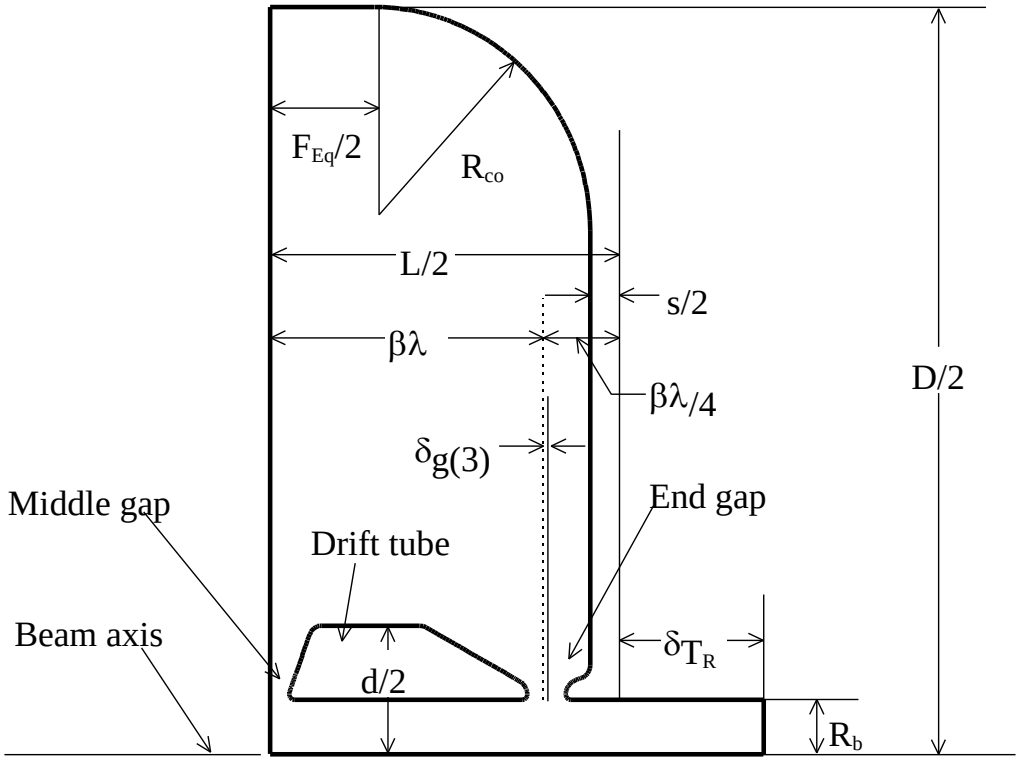

Figure XIII-14. Half of a two-drift-tube, 3-gap CCDTL cavity.

Figure XIII-15 shows details near the drift tube. CDTfish defines the nominal cavity length L in terms of a half integral multiple of βλ (5βλ/2 for the full cavity represented by this half cavity). The actual geometry is longer then L/2 by an amount equal to the bore-tube extension. All drift-tube gaps are exactly the same length. The code optionally moves the center of the end gap to make the cell's electrical centers βλ apart. The end-gap electrical center is βλ/4 from the nominal edge of the cavity.

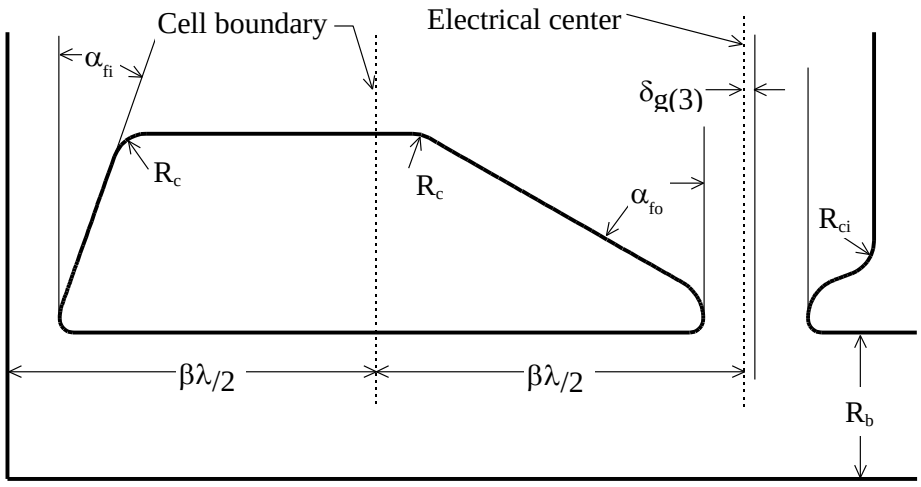

Figure XIII-15. Details near the drift tube in a CCDTL cavity.
This drift tube is from the 3-gap CCDTL cavity shown in Figure XIII-14. The cell boundary between the middle cell and the end cell is βλ/2 from the center of the cavity and βλ/2 from the electrical center of the end cell.

Note the use of different inner and outer face angles. CDTfish uses α_{fo} for drift tubes facing the cavity wall, and α_{fi} for drift tubes facing another drift tube. The idea here is to make the voltage gain E_0L_{cell} approximately equal in all the cells, where E_0 is the average electric field and L_{cell} is the length of a cell. Note that CDTfish does not adjust the face angles for this condition. After a CDTfish run, the user can run SF7 to plot $E_z(z)$. Also, the SFO output lists the values of E_0L_{cell} for each cell.

CDTfish provides two types of optional tuning surfaces, which we call tuning rings. An "equator tuning ring" is a bump on the outer cavity wall at the midplane or equator of the cavity. A "wall tuning ring" is a raised bump (or an indentation) on the end wall of the cavity, usually just below the outer corner radius. For either type of tuning ring, you can either define all of the geometrical parameters for the feature, or you can request a particular frequency effect Δf_{Ring}, in which case the code determines the ring thickness.

The CCLfish code includes the same optional feature. Please refer to the CCLfish discussion of tuning rings for more details. Figure XIII-6 shows the geometry for a wall tuning ring for either CCL or CCDTL cavities. Figure XIII-16 shows an equator tuning ring in a CCDTL cavity. Control file keyword RING_TYPE sets the value of *Ring* (see Table XIII-5), which defines the type of tuning ring and the method for determining the ring thickness. If *Ring* = 0, no tuning ring will appear in the cavity regardless of the settings for other tuning ring parameters. If *Ring* = 1, 2, 3, or 4, then the control file must contain nonzero values for at least the width W_{Ring} and the chamfer angle α_{Ring}. For *Ring* = 1 or 3, specify a nonzero value for the thickness T_{Ring}; and for *Ring* = 2 or 3, specify a nonzero value for the frequency effect Δf_{Ring}.

For a specified ring thickness T_{Ring} CDTfish tunes the cavity with the ring feature already included in the geometry. After finishing this problem, the code creates another Automesh input file with the ring removed from the cavity to determine the actual frequency effect of the ring. By default, the code will delete the extra files (xxx.AM, xxx.SEG, xxx.SFO, xxx.PMI, and xxx.T35) generated by this procedure. If you wish to

save these files, set DeleteNoRingFiles = No in file SF.INI. The extra files will have letter "A" appended to the original filenames.

Table XIII-5. Tuning ring options in CDTfish.

Ring	Description
0	No tuning ring.
1	Equator ring of a given thickness.
2	Equator ring of a given frequency effect.
3	Wall rings of a given thickness.
4	Wall rings of a given frequency effect.

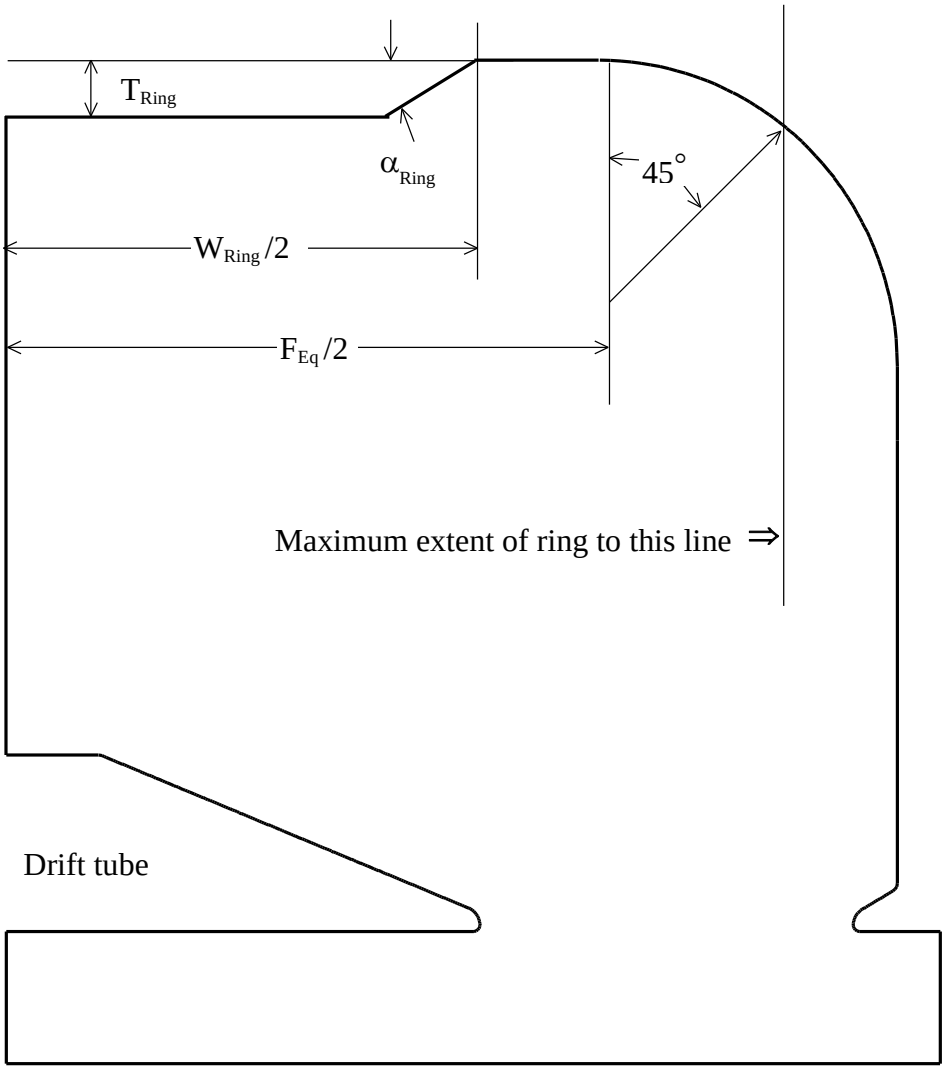

Figure XIII-16. Optional equator tuning ring in a CCDTL cavity.
The ring, which increases cavity frequency must fit entirely within boundary shown.

6. RFQfish: a tuning program for radio-frequency quadrupoles

RFQfish sets up the geometry of a radio-frequency quadrupole linac (RFQ). RFQfish assumes a four-fold symmetry and therefore sets up Superfish runs for only one quadrant of the RFQ. The control file defines up to 50 basic cell shapes. For each basic shape, the code tunes up to 50 cells by adjusting one of several geometrical parameters. If the control file adjusts a different geometrical parameter for the first problem of a basic shape the code will use the value of that parameter for subsequent problems.

a. RFQfish cavity shape

Figure XIII-17 shows the outline of an RFQ quadrant set up by RFQfish. Figure XIII-18 shows more detail near the vane tips. The voltage difference between adjacent vane tips (at the peak of the rf waveform) is called the gap voltage V_g. The user supplies the gap voltage for each problem so that RFQfish can normalize the fields. The field normalization affects only the output generated by the SFO part of the Superfish calculation.

The lower left corner is the RFQ beam axis. RFQ designers often refer to the unmachined tip of the vane as the "vane blank." Before the numerically controlled machining of the tip, the vane blank would have a rectangular cross section in these views. The break-out angle α_{bk} for the tool bit cutting the vane is usually about 10 degrees. Thus, the half width B_W of the vane blank must always exceed the vane-tip radius of curvature ρ for cross sections at any longitudinal position along the vane. Without this feature it would be very difficult to machine the tip without leaving a ridge along one side of the vane. A raised ridge would be especially undesirable because of the high electric field near the tip.

The vane-blank depth B_D is the distance from the RFQ axis to the vertex of angle α_1. At this point, the vane width increases at distances farther from the RFQ axis until it reaches the limit set by W_s, the half width of the vane shoulder. The shoulder segment of length L_s is parallel to the vane axis. At the end of the shoulder segment farthest from the RFQ axis is the vertex of angle α_2. Again, the vane width increases farther from the RFQ axis until it reaches the limit set by W_b, the half width of the vane base.

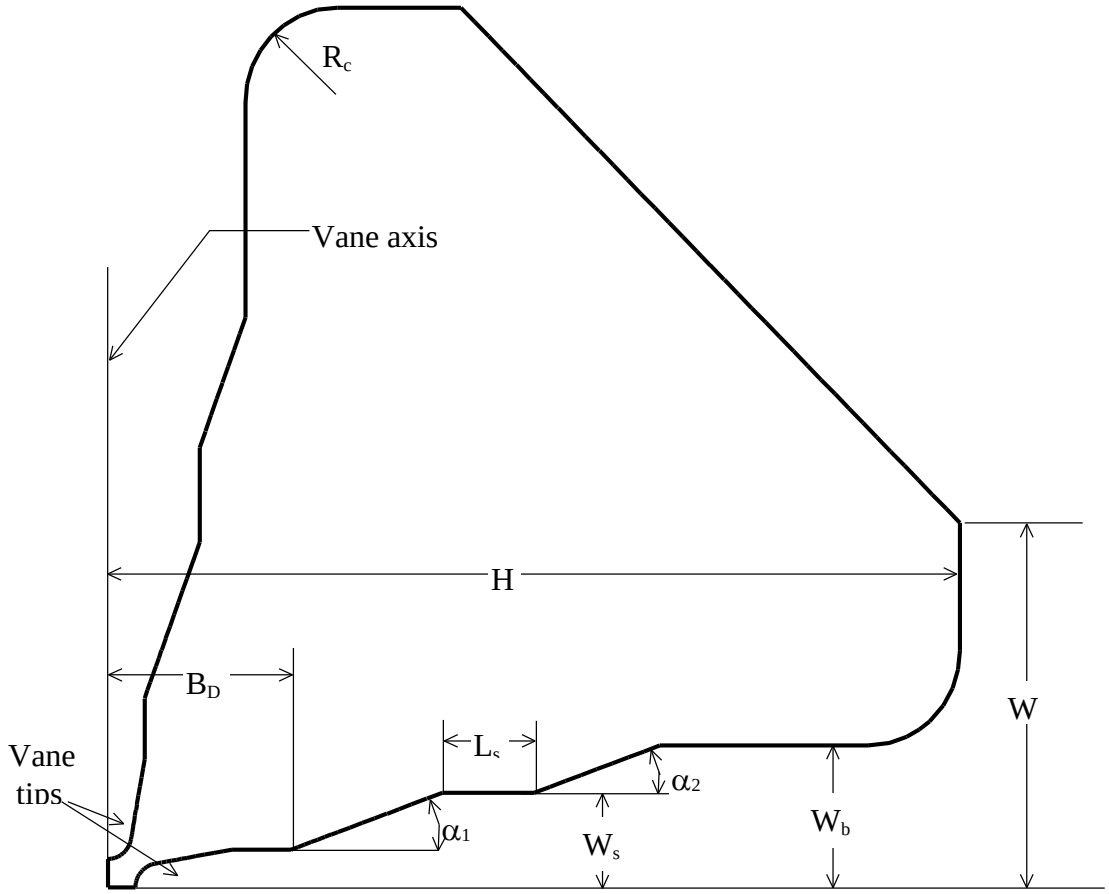

Figure XIII-17. Cross section of one quadrant of an RFQ cavity.
The beam axis is into the page at the lower left-hand corner of the figure. Refer to Figure XIII-18 for details near the vane tips.

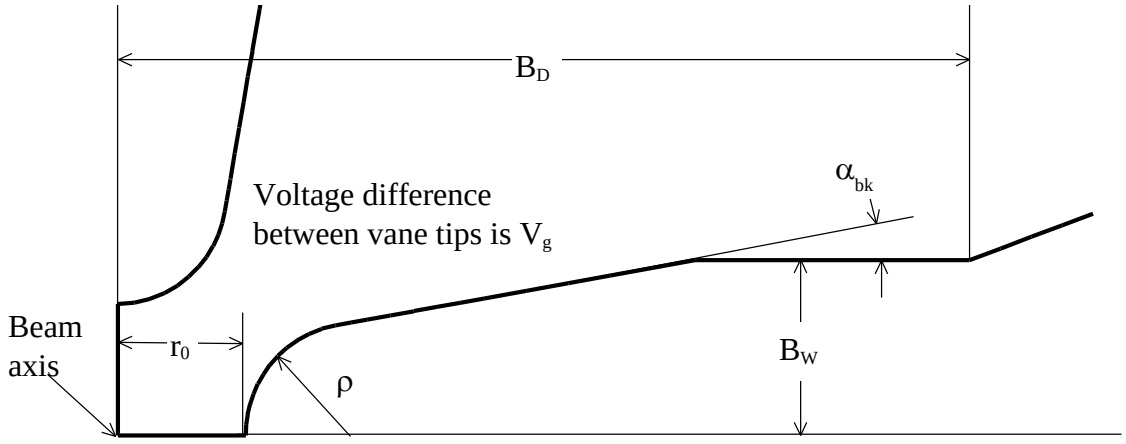

Figure XIII-18. Details near the vane tips for the RFQ quadrant.
These vane tips are from the RFQ quadrant shown in Figure XIII-17. Both tips are a distance r_0 from the beam axis. RFQfish sets up the geometry at the point of quadrupole symmetry and does not include the effects of the longitudinal vane-tip modulation.

7. SCCfish: a tuning program for side coupling cavities

SCCfish sets up the geometry for side coupling cavities typically used in the construction of coupled-cavity linacs designed by the codes CCLfish and CDTfish. The side coupling cell is a figure of revolution and contains a central tuning post. SCCfish can design either symmetric half cells or full cells with different shapes for left and right halves. The control file defines up to 100 problems. SCCfish tunes the cell by adjusting either the cavity diameter or the tuning post length. Figure XIII-19 shows the right-half outlines of two side coupling cells. The face angle α_f, the cavity arcs R_{co} and R_{ci}, and the tuning-post arcs R_{po} and R_{pi} are all optional features.

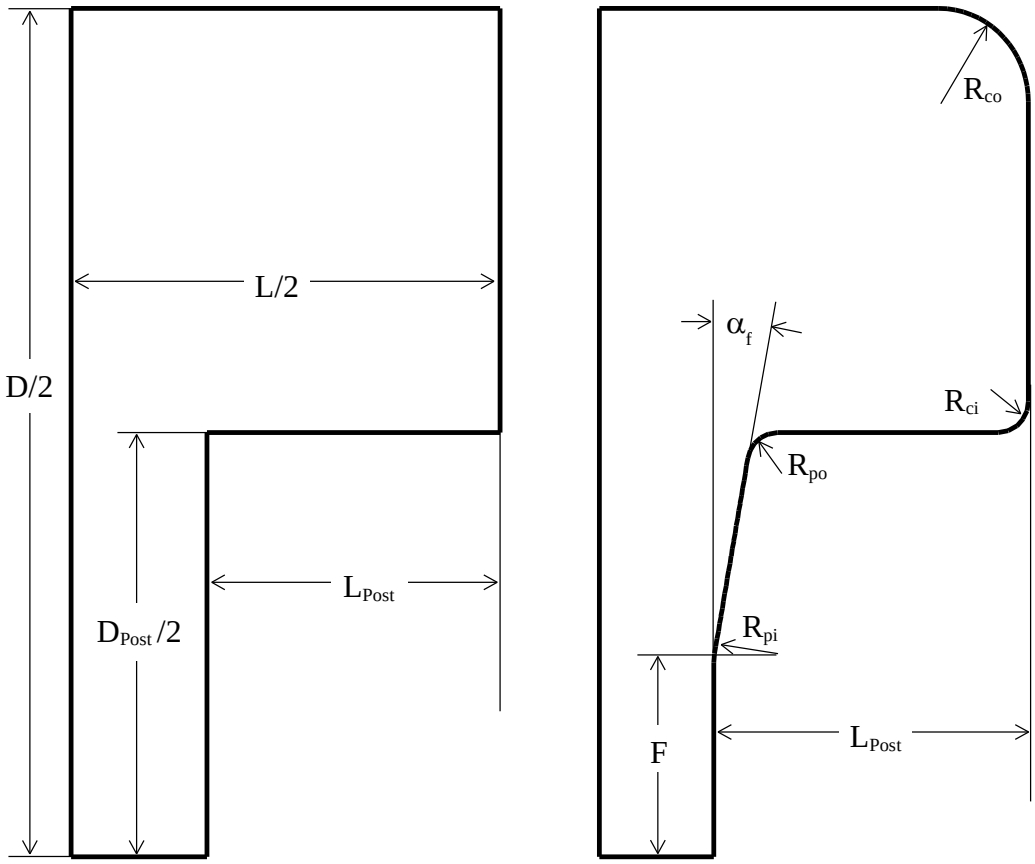

Figure XIII-19. Side coupling cavities.
The left side shows the simplest possible cavity. The most complex cavity with several arcs and a face angle appears on the right side. The figure shows only the fight side of a cavity. The code allows independent specification on left and right for all these features except the diameter.

B. Starting and stopping the tuning programs

To start a tuning code, double-click on a file with the code's default extension. These filename extensions are the first three letters of the program name. For example, double-clicking on files with extension DTL launches program DTLfish. To open the file in

Notepad, right click on the control file (with extension CCL, CDT, ELL, DTL, MDT, RFQ, or SCC) and choose Edit. You can also use a command line to start the code as in the example:

DTLfish *controlfile*

where *controlfile* is the name of the control file. This file defines the problem geometries and also contains information for constructing sequences of filenames for different problems. If the command line contains no filename the program will display the standard Windows Open dialog.

1. Using the Esc key to stop a tuning code or a Superfish run

Type END to stop the tuning code at any program prompt. You can press the ESC key to stop a Superfish run on a problem either during the mesh optimization in Automesh or during Fish execution. After pressing Esc you will have three options.

- Finish on cycle xx (Automesh mesh optimization) or after this iteration (Fish resonance search), save the solution, then stop normally.
- Continue the calculation.
- Stop now without saving the solution (the default).

Automesh can only stop on the next cycle (reported as "xx" in the above menu) for which it checks for convergence. If there are more problems pending when you choose Stop, then the tuning code asks whether you want to abort the entire session. Select Yes (the default) to stop the program, or type No to proceed to the next problem.

Suppose Fish is running and you can tell that the frequency is already very close to the final frequency that Fish will ultimately find for an early run on a problem. For example, letting Fish run to completion may take an extra iteration to get the slope of $D(k^2)$ within the established convergence limits. You can save some computation time by pressing Esc and then selecting Finish to force Fish to quit after the current iteration.

C. Session, problem, and run: tuning program terminology

The terms session, problem, and run (see Table XIII-6) describe different parts of the calculation. A session refers to one execution of a tuning program. During a session, the code works on problems defined in the control file. A problem involves one or more runs of the Superfish codes.

Table XIII-6. Tuning program terminology.

Term	Meaning
SESSION	A single execution of the tuning program.
PROBLEM	A cell or cavity to be tuned by varying a geometrical parameter.
RUN	A Superfish calculation involving codes Automesh, Fish, and SFO.

D. Files used in the tuning codes

Complete file specifications consist of the filename and an extension separated by a dot (.). Each tuning program works with the files listed in Table XIII-7. The Superfish codes themselves generate other files associated with each program during a run.

Table XIII-7. Tuning program files.

Type of file	Default Name	Default Extension
Control file	First 3 letters of code	First 3 letters of code
Control file of completed jobs	*controlfile*nn	First 3 letters of code
Log file	*controlfile*	LOG
Automesh NAMELIST input file	*problemfile*	AM
SFO input file	*problemfile*	SEG
SFO output file	*problemfile*	SFO
Binary solution file	*problemfile*	T35

The LOG file has the same name as the control file, but with extension LOG. The name of the *problemfile* depends on the FILEname_prefix and SEQuence_number settings in the control file. The code make filenames by adding various extensions to the *problemfile* root name for the each problem.

1. The control file and tuned-data file

The control file defines the problem geometries and it also contains information for constructing sequences of filenames for different problems. Both the default name and extension consists of the first three letters of the program name. For example, the default name of the CCLfish control file is CCL.CCL. The file can have any name or extension. The program first looks for the exact filename you entered. If it cannot find that file, it looks for the file you entered with no extension. If that file does not exist, it looks for the file you entered with the code's default extension (CCL, CDT, ELL, DTL, MDT, or RFQ).

To launch a tuning programs automatically, double-click on the control file that uses code's default extension. For example, double-clicking on files with extension DTL launches program DTLfish. To open the file in Notepad, right click on the control file and choose Edit.

The control file of completed jobs is a copy of the control file modified to reflect completed jobs. The name of this file is the same as the control file plus a two-digit number. For example, if the original control file is CCL.CCL, the first tuned-data control file would be CCL00.CCL, the next one would be CCL01.CCL, and so forth. For completed jobs, each tuning program makes the START code negative. If you manually interrupt a tuning program, or if the computer crashes, you can restart the code using the most recent control file of completed jobs.

```
Sample problems for tuning drift-tube linac cells.
Resonant frequency = 425 MHz
Adjusting cavity diameter, currently =          43.0000000

&REG KPROB=1,                              ; Superfish problem
MAT=1,                                     ; Material air or empty space
FREQ=425.,                                 ; Mode frequency
FREQD=425.,                                ; Design frequency, for transit-time factors
BETA=0.861331800081509D-001,               ; Design beta
NBSUP=1,NBSLO=0,NBSRT=1,NBSLF=1,           ; Boundary conditions
LINES=1,                                   ; Fix internal points on line regions
ICYLIN=1,                                  ; X=>Z,Y=>R, cylindrical coordinates
NORM=0,                                    ; Normalize to EZERO
EZERO=4400000.,                            ; Accelerating field
DTL=1,                                     ; Cavity is drift-tube linac
RMASS=-2.,                                 ; Rest mass value or indicator
EPSO=0.1E-05,                              ; Mesh optimization convergence parameter
TEMPC=29.000000,                           ; Temperature, degrees C
XDRI=0.79795058,YDRI=4.135766,             ; Drive point
DSLOPE=1,                                  ; Force at least 2 iterations
; X line-region physical locations:
XREG=0.,0.4,1.39604792616125,1.69604792616125,
; X line-region logical locations:
KREG=0,5,25,28,
KMAX=36,                                   ; Column number for X = XMAX
; Y line-region physical locations:
YREG=0.,0.707106781186548D-001,0.293933982822018,4.13576601717798,
4.48931940777126,5.1964261889578,
; Y line-region logical locations:
LREG=0,3,7,118,123,128,
LMAX=186, &                                ; Row number for Y = YMAX

; Start of boundary points
&PO X=0.,Y=0. &
&PO X=0.,Y=21.5 &
&PO X=3.0378915,Y=21.5 &
&PO X=3.0378915,Y=4.0297 &
&PO X=1.29604792616124,Y=4.0297 &
&PO NT=2,X0=1.29604792616124,Y0=3.5297,
X=-0.498097349045873,Y=0.435778713738291D-001 &
&PO X=0.551236723120183,Y=0.753325616392989 &
&PO NT=2,X0=0.875,Y0=0.725,
X=0.,Y=-0.325 &
&PO X=3.0378915,Y=0.4 &
&PO X=3.0378915,Y=0. &
&PO X=0.,Y=0. &
```

Figure XIII-20. Typical input file for program Automesh.

2. The tuning program LOG file

The LOG file is a running log of the tuning-code session. The code writes various data to this file before and after Superfish runs and after the cell is tuned. The name of the log file is the same as the control file except that the extension is LOG.

3. Automesh input file for a Superfish run

For each problem, a tuning program creates an Automesh input file that has extension AM. The tuning program creates this file using information from the control file. The Automesh input file in Figure XIII-20 is from a DTLfish run. You can edit the AM file and manually run the stand-alone Superfish codes, if desired.

4. SFO file, SFO output file

Program SFO writes file *problemfile*.SFO. The SFO output file includes a summary of cavity parameters. A tuning program uses data from this file after the first Superfish run on a new problem to calculate the change needed in the varied parameter. The tuning program uses frequency-shifts that appear in columns labeled dF/dZ or dF/dR in the output summary. SFO reports frequency shifts that correspond to 1-mm changes in cavity dimensions that tend to enlarge the empty space in the cavity. For example, consider the drift-tube-linac cell shown in Figure XIII-1. To compute the frequency shifts, the outer diameter moves out (dR is positive), and the drift-tube diameter moves in (dR is negative). For an increase in the gap, the drift-tube nose in Figure XIII-1 moves to the right (dZ is positive).

5. SEG file, Input file for program SFO

The tuning code creates file *problemfile*.SEG, which contains data entries for program SFO that cannot be included in the Automesh input file. For example, if the Automesh input file is PROB1.AM, then the SFO input file will be PROB1.SEG. The SEG file contains the segment-number data that SFO needs to calculate power and fields on metal surfaces. The tuning program writes different a SEG file for each problem. EndData

End

Figure XIII-21 shows a sample SEG file for a DTL cell with several breaks in the mesh size in both X and Y directions.

```
NumberOfHalfStems = 1
StemRadius = 0.9525000
FieldSegments                    ; Negative numbers are drift-tube stem segments
7 8 9 10
 –11 –12 –13 –14
15 16 17 18 19 20 21 22 23
EndData
End
```

Figure XIII-21. Typical SEG file for program SFO.
The code DTLfish created this example. All of the tuning programs include comments in the SFO input files they create to identify the entries.

The negative values on line 7 of the file indicate that SFO should calculate the power and frequency shifts for the stem. DTLfish always includes the stem (or stems) rather than the end wall of the cavity. If you need to calculate the power on an end wall, simply omit the minus signs and rerun SFO. If necessary, enter the filename and extension of the binary solution file that corresponds to the cell of interest. For example, to rerun SFO on solution generated for input file PROB2.AM, first edit file PROB2.SEG as appropriate, then enter the following command line to start SFO:

```
SFO   PROB2.T35
```

It is not necessary to include the name of the solution file if this particular file was the last one generated or the last one used by another Poisson Superfish code. See the discussion of SF.INI variable AutoOpenTAPE35 for more information. The example discussed here is from a DTLfish problem. The other tuning codes write similar SEG files with differences appropriate to the type of cavity involved.

E. Control file keywords common to all tuning programs

The tuning programs recognize certain keywords in the control file. Keywords can be in upper, lower, or mixed case. Most names are self explanatory. The next section discusses the keywords common to most of the programs, noting exceptions where appropriate. Lists of keywords show the shortest possible abbreviations in capital letters. Dimensions where appropriate are in square brackets. See Tuning program examples for some sample input files for each tuning program.

Table XIII-8 lists the valid keywords common to all the tuning programs. Those in the first group should appear only once in the control file. The rest can be different for each problem. Control-file keywords specific to each program are found in:

- Table XIII-14 for DTLfish,
- Table XIII-15 for CCLfish,
- Table XIII-16 for ELLfish,
- Table XIII-17 for MDTfish,
- Table XIII-19 for CDTfish,
- Table XIII-20 for RFQfish, and
- Tble XIII-22 for SCCfish.

Table XIII-8. Control file keywords common to all the tuning programs.

Keyword	Description
TITLE	Next line starts the problem description.
ENDTitle	Indicates the end of the problem description.
; !	These characters in column one indicate a comment line.
PARTICLE	Particle type (e+, e–, H+, H–, D+, or D–), H+ is default (except SCCFSH).
REST_mass	Rest mass energy for a nonstandard particle (except SCCFSH).
RESISTivity	Normal conductor bulk resistivity [Ω-cm].
TEMPerature	Normal conductor temperature and reference settings.
R_Surface	RF surface resistance [Ω].
SUPERConductor	Superconductor temperature, Tc, and residual resistance.
PLOTting	(Currently disabled).
FILEname_prefix	Can include a drive letter and path.
SEQuence_number	A one-to-four-digit integer.
FREQuency	Starting and target frequency [MHz].
NEWFREQuency	New frequency [MHz] used only in Fscale program.
DELTA_frequency	Tolerance for the tuned frequency [MHz].
MESH_size	Smallest mesh size [cm].
INCrement	Mesh-size ratio between adjacent regions (except SCCfish).
YINCrement	Optional Y mesh increment (except RFQfish, SCCfish, ELLfish).
E0_Normalization	E_0 field normalization [MV/m] (except RFQfish, SCCfish).
E0T_Normalization	E_0T field normalization [MV/m] (except RFQfish, SCCfish).
ENDFile	End of the control file.

1. TITLE and ENDTitle, Defining the problem description

You can enter a problem description consisting of up to nine 80-character lines between lines containing the keywords TITLE and ENDTitle. The tuning code adds one more line to your description. This last line contains the current value of the parameter used to tune the cavity. The problem description is stored in the binary solution file and the Superfish codes print the description in the output files.

The code ignores blank lines in the problem description. If a line in the TITLE/ENDTitle section starts with a comment indicator (semicolon or exclamation mark), the code reads the line and includes it in all Automesh input files. Automesh ignores these comment lines and they will not appear in output files written by Automesh, Fish or SFO. If you include more than nine lines, the program stores only the first nine lines. An input file should contain only one TITLE/ENDTitle section. If a file contains multiple sections, the code uses the last one in the file.

2. Control-file comment-line indicators

You may include comment lines in the control file after a semicolon (;) or exclamation mark (!). The program ignores everything after a comment indicator. It also ignores blank lines. Blank lines and comment lines may appear anywhere in the file.

3. ENDFile, The last line in the control file

ENDFile is the last line in the control file read by a tuning program. You can add more information to the file without using comment line indicators.

4. PARTICLE and REST_mass, Particle type for kinetic energy calculation

PARTICLE and REST_mass are keywords in the control file. For most tuning programs, SFO reports the kinetic energy of the particle that corresponds to the particle velocity used in the transit-time integrals. The exception is RFQfish, which does not deal with particle kinetic energies. These keywords affect the value of RMASS that the tuning program writes in the Automesh input file.

The default rest mass energy is that of a proton (H+). Use the PARTICLE or REST_mass keyword to enter a different rest-mass energy. Standard particles in SFO are electrons, hydrogen ions, and deuterium ions. For a standard particle, use the PARTICLE keyword followed by the appropriate symbol in Table XIII-9.

Table XIII-9. Symbols for some common ions.

Symbol	Rest mass energy (MeV)
e+	0.510998918
e–	0.510998918
H+	938.272029
H–	939.294012
D+	1875.61282
D–	1876.63480
He3++	2808.39142
He4++	3727.37917
Mu+	105.6583692
Mu–	105.6583692

The control-file line “PARTICLE H–” selects the negative hydrogen ion. For nonstandard particles, use the REST_mass keyword followed by the particle’s rest mass energy in MeV. For example, to specify the mass of a singly-charged boron-11 positive ion, use the line:

```
REST_mass                10254.67
```

PARTICLE or REST_mass should appear only once in the control file. The codes ignore all but the last occurrence of either keyword in the control file.

5. Keywords specifying the surface resistance

The tuning programs use the control-file keywords in Table XIII-10 to indicate how to calculate the surface resistance for rf power calculations in SFO. These keywords affect the value of several Poisson Superfish problem variables that the tuning program writes in the Automesh input file.

Use only one keyword at a time for an entire session. Only the last of these four keywords in the control file has any effect. The default surface is room-temperature copper, which

has a bulk resistivity of 1.7241 μΩ-cm. This value corresponds to the 100% IACS resistivity for copper at 20 degrees C. IACS is the International Annealed Copper Standard. [Note: the nominal value 1.7 μΩ-cm was used in older versions of SFO until September, 1995.] The SFO chapter includes a discussion of the calculations used for normal conductors and for superconductors. To correct for the temperature of normal-conducting surfaces, use the TEMPerature keyword followed by up to four of the parameters listed in Table XIII-11(the last three are optional). These parameters are the Poisson Superfish problem variables used in SFO for normal conductors (when IRTYPE is equal to 0).

Table XIII-10. Control file keywords for specifying the surface resistance.

Keyword	Description
RESISTivity	Normal conductor bulk resistivity in Ω-cm.
TEMPerature	Normal conductor temperature and reference settings.
SUPERConductor	Superconductor temperature, Tc, and residual resistance.
R_Surface	RF surface resistance in Ω.

Table XIII-11. Normal-conductor parameters used with the TEMPerature keyword.

Name	Default value	Description
TEMPC	20.0	Operating temperature for normal conductors.
TEMPR	20.0	Reference temperature.
RHOR	1.7241D–6	Reference resistivity (Ω-cm) at temperature TEMPR.
ALPHAT	3.9300D–3	Temperature coefficient of resistance at TEMPR.

For example, if a copper cavity will operate at a temperature of 40 degrees C, include the following line in the control file:

```
TEMPerature          40
```

For other normal conducting materials, include values for the reference temperature TEMPR, the resistivity at TEMPR, and the material's temperature coefficient of resistance. You also can use the RESISTivity keyword for a known resistivity. This entry sets IRTYPE = 3 and affects the value of RHO that the tuning program writes in the Automesh input file. No temperature correction is applied in this case.

To specify a superconducting surface, use the SUPERConductor keyword followed up to three of the (optional) parameters listed in Table XIII-12. These parameters are the Poisson Superfish problem variables used in SFO for superconductors when IRTYPE is equal to 1.

Table XIII-12. Parameters used with the SUPERConductor keyword.

Name	Default Value	Description
TEMPK	2.0	Operating temperature in degrees K.
TC	9.2	Critical temperature in degrees K.
RESIDR	1.0D–8	Residual resistance in Ω.

The last method in the tuning codes for specifying the surface resistance is to enter the known surface resistance using the R_Surface keyword. This entry sets IRTYPE = 2 and affects the value of RS that the tuning program writes in the Automesh input file.

6. Filenames for each problem in the control file

Two control-file keywords specify the names of the Automesh input file and related Superfish input and output files for subsequent problems. FILEname_prefix is a character string to which the tuning program appends consecutive sequence numbers. SEQuence_number is the starting sequence number. The prefix can include a drive letter and path. Valid sequence numbers are from 1 to 9999. The tuning code adds appropriate extensions to the name of the *problemfile* to make several unique filenames for each problem. For example, suppose these two lines appear in the control file:

```
FILEname_prefix      PROB
SEQuence_number      101
```

Table XIII-13 shows the names of the various files created by the tuning program for the first two problems. The sequence number continues to increment through all problems in the control file unless you redefine it before the START line of a problem. Be careful if you set the sequence number to a smaller value than used previously. The tuning code will overwrite files of the same name.

Table XIII-13. Files created by the tuning program.

First problem	Second problem	Description
PROB101.AM	PROB102.AM	Automesh input files.
PROB101.SEG	PROB102.SEG	SFO input files.
PROB101.SFO	PROB102.SFO	SFO output files.
PROB101.PMI	PROB102.PMI	SFO Parmila data files.
PROB101.T35	PROB102.T35	Binary solution files.

Programs CCLfish and CDTfish may produce a second set of files if the problem geometry includes the optional tuning ring. If the control file specifies the ring thickness T_{Ring} these codes first tune the cavity with the ring feature already included in the geometry. After finishing this problem, CCLfish or CDTfish creates another Automesh input file with the ring removed from the cavity to determine its frequency effect. The extra files will have letter “A” appended to the original filenames.

7. Setting the target frequency and tolerance

The FREQuency keyword defines the starting frequency for Superfish runs and the target frequency for the tuned cell. The DELTA_frequency keyword defines the tolerance for the tuned frequency. Both have dimensions of MHz. In each tuning program, you can vary one of several geometrical parameters to tune the cavity. The tuning program will continue to set up more Superfish runs on each problem until the calculated frequency is within the specified tolerance of FREQuency, or until 10 attempts have failed to tune the cavity. You should relax the tolerance for crude meshes. If the tolerance is too tight, the tuning algorithm may fail. Changes in the mesh from one run to the next can cause

frequency differences. The tuning program works closely with the Fish root finder solver to speed the tuning operation by trying to eliminate some unnecessary iterations in Fish. During the resonance search, the Fish program reports in the LOG file information about the method it used to generate the next frequency estimate.

The NEWFREQuency keyword defines a new target frequency for use by the utility program Fscale, which creates a new control file with all linear dimensions scaled by the ratio of frequencies. The tuning programs ignore this keyword.

8. Setting the mesh size in the problem geometry

MESH_size is the smallest mesh size in cm to use in the Superfish calculation and INCrement is the mesh-size ratio between adjacent regions. Program SCCfish does not divide the mesh into regions of different size. The smallest mesh occurs near the smallest features in the problem geometry, usually where the electric fields are largest and change most rapidly. The tuning code increases the mesh size in other places if it can significantly reduce the total number of mesh points by doing so. Changing the mesh size adds more segments for the SFO calculation of power and surface fields. The tuning code writes the SEG file, which contains the segment-number data that SFO needs to calculate power and fields on metal surfaces.

The default ratio between mesh sizes in adjacent regions is 2.0. You can enter a different factor with the INCrement line. For example, to specify a mesh size of 0.05 cm with only 50% increases in the mesh size between regions, use these lines:

```
MESH_size       0.05
INCrement       1.5
```

If the INCrement is 1.0 or smaller, the code will not use line regions. Instead, the Automesh input file will include a setting for variable DX, the spacing of mesh triangles in the X direction. The value of DX will equal the setting for MESH_size in the control file. Automesh determines the appropriate size for DY. Of course, the smaller the mesh dimension, the more mesh points in the region, and the longer it takes to do a Superfish calculation. As a guide, try to make the mesh four or five times smaller than the smallest radius of curvature on the drift-tube nose. Avoid large increments (e.g., ~3 or more). See also Using line regions to control the mesh size in Automesh.

The optional keyword YINCrement allows independent control over the X and Y mesh intervals in programs DTLfish, CCLfish, CDTfish and MDTfish. For example, if you do not want any X line regions, use these lines:

```
INCrement       1
YINCrement      2
```

a. Controlling line regions through the drift tube in DTLfish and MDTfish

SF.INI variable AllowYregInGap determines whether programs DTLfish and MDTfish insert a Y line regions through the face angle segment on the drift tubes. The code checks if there is sufficient room for several rows of the next larger mesh triangles between the outer nose radius and the corner radius. One reason not to use a Y line region though the

drift tube faces would be when trying to make the MDTfish code mesh a series of DTL cells in the same manner as DTLfish meshed the geometry. If the cells have differences in the drift-tube geometry, then DTLfish could place the Y line region at different locations, so it would be impossible for MDTfish to generate the same mesh.

9. Linking parameters to previously calculated values

You can use the keyword PREVIOUS or REPLACE (instead of a number) for many of the parameters in the control file. At run time on the problem of interest, the code will substitute the value of the parameter from the previous run just completed. This practice allows more flexibility in the design process. The only difference between PREVIOUS and REPLACE is that the PREVIOUS keyword will also appear in the control file of completed jobs. Then, on subsequent sessions that you start from a code-created control file, the program will continue to make the substitution. When you use REPLACE, the substituted value is a permanent replacement and the code writes the numerical value into the control file of completed jobs.

10. Setting the preferred normalization method

Keywords E0_Normalization and E0T_Normalization in the control file specify the field normalization used by SFO for programs CCLfish, CDTfish, DTLfish, ELLfish, and MDTfish. These programs all are concerned with cavities that accelerate particles using TM_{010}-like rf fields. Neither the RFQfish nor the SCCfish control files use these normalization keywords. RFQfish uses the gap voltage V_g to normalize the fields, and SCCfish always uses E_0 = 1.0 MV/m.

Keyword E0_Normalization specifies a value for E_0, the average axial electric field for the cell. E0T_Normalization specifies a value for the product E_0T, where T is the transit time factor. Only one value is needed in the control file. If both values appear, the code uses the preference specified in SF.INI as in these examples:

```
[DTLfish]
Normalization = E0

[DTLfish]
Normalization = E0T
```

Each tuning code checks the preference in SF.INI if the control file contains neither or both lines that set the field normalization. Using only one of these keywords for a problem overrides the SF.INI setting. In the absence of any settings in either SF.INI or the control file, the codes normalize to E_0 = 1.0 MV/m. The SF.INI file distributed with the codes has preferences set to E_0 in the [DTLfish] and [MDTfish] sections, and to E_0T in the [CCLfish], [ELLfish], and [CDTfish] sections.

The SF.INI Normalization setting is important because after finishing a problem the control file of completed jobs will contain both normalization settings. For example, suppose DTLfish completed a run for which it normalized to E_0 = 5.0 MV/m and the calculated transit-time factor was 0.8. The tuned-data file will include the following lines for that completed problem:

```
E0_Normalization        5
E0T_Normalization       4
```

Provided you make no other changes to the geometrical parameters for this problem, it does not matter which method of normalization the code uses if you run this problem again by removing the minus sign on the START codes. However, if you modify the geometry in some way, it may matter, because the new geometry could result in a different transit-time factor. If SF.INI has a preference for E_0, then the code will again normalize to E_0 = 5.0 MV/m, and get a new value for the product E_0T.

11. Saving binary solution files for each completed problem

SF.INI variable SaveTAPE35 sets whether tuning codes save a binary solution file for each problem. The default is to create files with extension T35 and retain them after completing the solution. Saving the solution files allows you view the mesh and field lines later using WSFplot, or (for example) to renormalize the fields in SFO to a different value of E_0 or E_0T. One reason you might choose not to save the solution files would to make more disk space available for other files. However, you would then need to compute the solution again to view the fields in WSFplot or to run another postprocessor such as SF7. If variable SaveTAPE35 = No, then the tuning program deletes each solution file after completing the solution. The updated AM files and the SFO files contain the results of each solution, but there are no binary solution files in the directory after the tuning-code session. Each tuning program can have its own setting for SaveTAPE35. For example, the following settings configure CCLfish to discard solution files and CDTfish to save solution files:

```
[CCLfish]
SaveTAPE35 = No

[CDTfish]
SaveTAPE35 = Yes
```

F. Control-file keywords specific to each tuning code

The tables in the following sections list the control-file keywords specific to each tuning program. Keywords common to all the programs appear in Table XIII-8. Control-file keywords specific to each program are found in:

- Table XIII-14 for DTLfish,
- Table XIII-15 for CCLfish,
- Table XIII-16 for ELLfish,
- Table XIII-17 and Table XIII-18 for MDTfish,
- Table XIII-19 for CDTfish,
- Table XIII-20 and Table XIII-21 for RFQfish, and
- Tble XIII-22 for SCCfish,

The symbols that appear in these tables for the cavity dimensions appear in the figures for each type of problem. See Tuning program examples for some sample input files for each tuning program.

The tuning codes list all the geometrical setup parameters near the beginning of the SFO output files. These parameters come from current values specified by the control-file keywords. (This list does not include parameters that do not change value during a tuning-code session such as the particle type, surface resistance options, etc.) The utility program SFOTABLE can include up to seven of these parameters as columns in a Tablplot file created from a family of SFO files.

1. DTLfish control-file keywords

The DTLfish control file defines the geometry for up to 100 problems. Table XIII-14 lists the control-file keywords used by DTLfish. The code also uses the common keywords listed in Table XIII-8. DTLfish is one of several tuning codes that has "paired keywords" for specifying the cavity length and gap length. Be sure to review the section that discusses these paired quantities.

Table XIII-14. DTLfish control-file keywords

Keyword	Symbol	Description
INITIALEnergy	W_i	Initial energy of a DTL tank used by program DTLCells.
BETA	β	Particle velocity relative to the speed of light.
LENGTH	L	Cavity length [cm].
DIAMeter	D	Cavity diameter [cm].
BORE_radius	R_b	Bore radius [cm].
G_OVER_Beta_lambda	$g/\beta\lambda$	Ratio of gap length to $\beta\lambda$.
GAP_Length	g	Gap length between noses [cm].
E0_Normalization	E_0	E_0 value to use as the field normalization [MV/m].
E0T_Normalization	E_0T	Product of E_0T to use as the field normalization [MV/m].
CORNER_radius	R_c	Radius at the outer end of the face-angle segment [cm].
OUTER_nose_radius	R_o	Radius at the inner end of the face-angle segment [cm].
INNER_nose_radius	R_i	Radius near the bore [cm].
FLAT_length	F	Length of straight segment between the nose radii [cm].
FACE_angle	α_f	Drift-tube face angle relative to vertical [degrees].
DRIFT_TUBE_Diameter	d	Diameter of the drift tube at its midpoint [cm].
PHASE_length	$\Phi_{1/2}$	For a half cell [degrees] (default is 180 degrees).
STEM_Diameter	d_{Stem}	Diameter of the drift-tube stem [cm].
STEM_Count	N_{Stem}	Number of drift tube stems.
START	N_v	Starts a problem using current parameters.

2. CCLfish control-file keywords

The CCLfish control file defines the geometry for up to 100 problems. Table XIII-15 lists the control-file keywords used by CCLfish. The code also uses the common keywords listed in Table XIII-8. CCLfish is one of several tuning codes that has "paired keywords" for specifying the cavity length and gap length. Be sure to review the section that discusses these paired quantities.

Table XIII-15. CCLfish control-file keywords.

Keyword	Symbol	Description
INITIALEnergy	W_i	Initial energy of a CCL section used by program CCLCells.
FINALEnergy	W_f	Final energy of a CCL section used by program CCLCells.
CAVITY_shape	*Shape*	Defines the type of cavity for all problems in a session.
PI_mode		Use a Dirichlet boundary at the end of the bore tube.
ZERO_mode		Use a Neumann boundary at the end of the bore tube.
BORE_radius	R_b	Bore radius [cm].
BETA	β	Particle velocity relative to the speed of light.
LENGTH	L	Cavity length [cm].
DIAMeter	D	Cavity diameter [cm].
G_OVER_Beta_lambda	$g/\beta\lambda$	Ratio of gap length to $\beta\lambda$.
GAP_Length	g	Gap length between noses [cm].
E0_Normalization	E_0	E_0 value to use as the field normalization [MV/m].
E0T_Normalization	E_0T	Product of E_0T to use as the field normalization [MV/m].
OUTER_CORNer_radius	R_{co}	Radius at the cavity outer wall [cm].
INNER_CORNer_radius	R_{ci}	Radius at the outer end of the cone-angle segment [cm].
ENDWALL_Radius	R_W	End-wall radius of curvature [cm] (*Shape* = 2).
ENDWALL_Center	H_W	Height of end-wall center of curvature [cm] (*Shape* = 2).
OUTER_NOSE_radius	R_o	Radius at the inner end of the cone-angle segment [cm].
INNER_NOSE_radius	R_i	Radius near the bore [cm].
FLAT_length	F	Length of straight segment between the nose radii [cm].
CONE_angle	α_c	Drift-tube cone angle relative to horizontal [degrees].
SEPTUM_thickness	s	Thickness of the web or septum between cavities [cm.]
PHASE_length	$\Phi_{1/2}$	For a half cell [degrees] (default is 90 degrees).
RING_TYPE	*T-Ring*	Defines a tuning ring in a standard CCL cell (*Shape* = 0).
RING_Width	W_{Ring}	Tuning ring width [cm] (*Shape* = 0).
RING_DY	Δy_{Ring}	Tuning ring distance below corner radius [cm] (*Shape* = 0).
RING_Angle	α_{Ring}	Tuning ring angle with a horizontal line [cm] (*Shape* = 0).
RING_Effect	Δf_{Ring}	Frequency effect of the tuning ring [MHz] (*Shape* = 0).
RING_Thickness	T_{Ring}	Thickness of the tuning ring [cm] (*Shape* = 0).
START	N_v	Starts a problem with the current parameters.

a. Selecting either Neumann or Dirichlet boundary conditions in CCLfish

Keywords ZERO_mode and PI_mode control the boundary condition at the end of the bore tube. Keyword PI_mode selects the Dirichlet boundary condition for which electric fields are parallel to the surface and is the default setting in CCLfish. The Dirichlet boundary condition is appropriate for coupled cavities operating in the π mode. Keyword ZERO_mode selects the Neumann boundary condition for which electric fields are

perpendicular to the surface. This mode is appropriate if the electric field in adjacent cavities has the same sign as the cavity under consideration.

The PI_mode or Dirichlet boundary condition is the default setting for buncher cavity calculations. However, for a structure in which there is no adjacent cavity, neither the Dirichlet nor the Neumann boundary condition is appropriate. If the bore tube is not long enough compared to its radius, then the mode frequency and field distribution calculated by Superfish will be incorrect. The electric fields of the buncher cavity should terminate on the inside of the bore. You can judge whether the bore tube is long enough in the following way. Run the problem twice, once with each boundary condition. If the difference between the mode frequencies for these two runs is negligible, the bore tube is long enough. If these frequencies differ significantly, increase the bore-tube length without changing the cavity shape by adding the same length to both the cavity length and the septum thickness.

3. ELLfish control-file keywords

The ELLfish control file defines the geometry for up to 50 problems. Table XIII-16 lists the control-file keywords used by ELLfish. The code also uses the common keywords listed in Table XIII-8. ELLfish is one of several tuning codes that has "paired keywords" for specifying the cavity length and gap length. Be sure to review the section that discusses these paired quantities.

Several geometrical parameters allow either a single entry for both sides of a full cavity or separate entries for the left and right sides. For example, if keyword DOME_B appears (setting the value of b_D), the code uses b_D to set the values of both $b_{D,L}$ and $b_{D,R}$. Keyword entries appearing later in the control file (but before the problem's START line) of either LEFT_ DOME_B or RIGHT_ DOME_B take precedence over the DOME_B entry. On the very first problem of a session, if only the entry RIGHT_ DOME_B appears, then ELLfish sets $b_D = b_{D,R}$ and $b_{D,L} = b_{D,R}$. The code then uses these setting as the default values for subsequent runs.

Rules similar to the dome-size rules apply to the wall angles (α_w, α_L, α_R), bore radii (R_b, $R_{b,L}$, $R_{b,R}$), dome ellipse aspect ratios (a_D/b_D, $(a_D/b_D)_L$, $(a_D/b_D)_R$), iris ellipse aspect ratios (a_I/b_I, $(a_I/b_I)_L$, $(a_I/b_I)_R$), equator flats (F_{Eq}, $F_{Eq,L}$, $F_{Eq,R}$), and iris flats (F_I, $F_{I,L}$, $F_{I,R}$). However, note that F_{Eq} is the full length of a symmetrically placed flat at the equator. If only the entry EQUATOR_flat appears, then ELLfish sets $F_{Eq,L} = F_{Eq}/2$ and $F_{Eq,R} = F_{Eq}/2$. On the first problem, if only the entry RIGHT_Equator_flat appears, then ELLfish sets $F_{Eq,L} = F_{Eq,R}$ and $F_{Eq} = 2F_{Eq,R}$.

If the ELLfish control file includes settings for BETASTART, BETASTOP, BETASTEP, and if BETATABLE = 1 or 2, then ELLfish creates a Tablplot file of transit-time factors versus particle velocity β for each cavity in the control file. There is one file for each ELLfish problem. If BETATABLE = 1, then Z_C in the transit-time integrals is the geometric center of the cavity. If BETATABLE = 2, Z_C is the electrical center. (This distinction is important only for asymmetric full-cell problems.) The Tablplot files are produced by the SFO section of ELLfish. The code uses the control-file settings to define

variables BETA1, BETA2, DBETA, and IBETA in the Automesh input file. There are more options available when running the stand-alone SFO program.

4. MDTfish control-file keywords

The MDTfish control file defines the geometry for up to 20 problems. Table XIII-17 lists the control-file keywords used by MDTfish. The code also uses the common keywords listed in Table XIII-8.

Table XIII-16. ELLfish control-file keywords.

Keyword	Symbol	Description
NumberOfCells	N_{Cells}	Number of cells in a cavity (used in ELLCAV program).
BETA	β	Particle velocity relative to the speed of light.
LENGTH	L	Cavity length [cm].
DIAMeter	D	Cavity diameter [cm].
E0_Normalization	E_0	E_0 value to use as the field normalization [MV/m].
E0T_Normalization	E_0T	Product of E_0T to use as the field normalization [MV/m].
BORE_radius	R_b	Bore radius [cm].
LEFT_BORE_radius	$R_{b,L}$	Bore radius at the left edge of a full cell [cm].
RIGHT_BORE_radius	$R_{b,R}$	Bore radius at the right edge of a full cell [cm].
DOME_B	b_D	Vertical semi-axis of dome ellipse [cm].
LEFT_DOME_B	$b_{D,L}$	Dome-ellipse vertical semi-axis on the left side of cavity [cm].
RIGHT_DOME_B	$b_{D,R}$	Dome-ellipse vertical semi-axis on the right side of cavity [cm].
DOME_A/B	a_D/b_D	Ratio of the dome ellipse semiaxes.
LEFT_DOME_A/B	$(a_D/b_D)_L$	Left-side ratio of the dome ellipse semiaxes.
RIGHT_DOME_A/B	$(a_D/b_D)_R$	Right-side ratio of the dome ellipse semiaxes.
WALL_Angle	α_w	Angle of wall flat segment to the vertical [cm].
LEFT_Wall_angle	α_L	Left-side flat-segment angle [cm].
RIGHT_Wall_angle	α_R	Right-side flat-segment angle [cm].
EQUATOR_flat	F_{Eq}	Full length of straight segment at the equator [cm].
LEFT_Equator_flat	$F_{Eq,L}$	Length of the equator flat on the left side [cm].
RIGHT_Equator_flat	$F_{Eq,R}$	Length of the equator flat on the right side [cm].
IRIS_flat	F_I	Length of both straight segments at the iris [cm].
LEFT_Iris_flat	$F_{I,L}$	Length of the iris flat on the left side [cm].
RIGHT_Iris_flat	$F_{I,R}$	Length of the iris flat on the right side [cm].
IRIS_A/B	a_I/b_I	Ratio of the iris ellipse semiaxes.
LEFT_ IRIS_A/B	$(a_I/b_I)_L$	Left-side ratio of the iris ellipse semiaxes.
RIGHT_ IRIS_A/B	$(a_I/b_I)_R$	Right-side ratio of the iris ellipse semiaxes.
RIGHT_BEAM_tube	δT_R	Length of bore tube extension on the right side [cm].
SECOND_BEAM_tube	δT_2	Length of a second bore tube on the right side [cm].
SECOND_TUBE_radius	$R_{b,2}$	Radius of second bore tube extension on right side [cm].
BETASTART	β_1	Starting velocity for T versus β table.
BETASTOP	β_2	Ending velocity for T versus β table.
BETASTEP	$\Delta\beta$	Velocity increment for T versus β table.
BETATABLE		.
START	N_v	Starts a problem with the current parameters.

Table XIII-17. MDTfish control-file keywords.

Keyword	Symbol	Description
BORE_radius	R_b	Bore radius [cm].
DIAMeter	D	Cavity diameter [cm].
LEFT_WALL_shift	δW_L	Low-energy end-wall relative to start of cell 1 [cm].
RIGHT_WALL_shift	δW_R	High-energy end-wall relative to end of last cell [cm].
LEFT_BEAM_tube	δT_L	Length of bore tube extension on the low-energy end [cm].
RIGHT_BEAM_tube	δT_R	Length of bore tube extension on the high-energy end [cm].
PHASE_per_cell	Φ	Phase length of a full cell [degrees] ($1\beta\lambda$ is 360 degrees).
STEM_DIAMeter	d_{stem}	Drift-tube stem diameter [cm].
STEM_data		Lines following contain optional stem parameters.
POST_data		Lines following contain optional post-coupler parameters.
CELL_data	N_v	Lines following contain DTL cell parameters.
ENDData		End of a data table.

a. Entering the cell data in MDTfish

The table of entries between CELL_data and ENDData has the columns listed in Table XIII-18. Most of these parameters in this table are the same ones used to describe an individual half cell in DTLfish or the cell properties in CDTfish. This 10-column table can have up to 50 lines, one line for each cell in the problem. Each CELL_data table corresponds to new problem. The file can contain up to 20 problems. Other keywords can appear before CELL_data sections to redefine parameters common for the next problem.

Table XIII-18. Data columns after the CELL_data keyword.

Column	Symbol	Description
1	Z_c	Gap center from left edge of cell [cm].
2	g	Gap length [cm].
3	α_L	Left face angle [degrees].
4	α_R	Right face angle [degrees].
5	R_c	Corner radius [cm].
6	$R_{i,L}$	Left inner nose radius [cm].
7	$R_{i,R}$	Right inner nose radius [cm].
8	F_L	Left nose flat length [cm].
9	F_R	Right nose flat length [cm].
10	$R_{o,L}$	Left outer nose radius [cm].
11	$R_{o,R}$	Right outer nose radius [cm].
12	L	Cell length [cm].
13	E_0	Design E_0 [MV/m].

The CELL_data line includes the start code, which indicates the parameter to vary to tune the cavity to the desired frequency. For example:

CELL_data N_v

If N_v is 0, 1, or missing, then MDTfish runs Superfish to calculate the frequency, but it does not tune the cell to the target frequency. Table XIII-26 gives all the options for the

start code on the CELL_data line. If N = 2, MDTfish varies the cavity diameter, and if N = 3, the code varies the drift-tube gaps. Negative values serve as place holders for completed jobs or jobs to be skipped during the present session.

The code computes the length of the cavity from the individual cell lengths and the values specified earlier for the end-wall positions and bore-tube extensions. The cell lengths often come from an accelerator design code such as Parmila. The cell lengths together with the longitudinal positions of the gaps determine the placement of the drift tubes in the tank. Program SFO uses the cell lengths in the E_0 and transit-time integrals.

b. Stem and post-coupler data in MDTfish

Two tables of entries in the MDTfish control file contain optional information used by SFO to calculate frequency shifts and power losses for drift-tube stems and post couplers. Keywords STEM_data and ENDData bracket a table containing the drift-tube stem configuration. Keywords POST_data and ENDData bracket a table containing the post-coupler configuration. Each table has 3 columns. The first column in each table is the drift-tube number for the stem or post coupler. For stem data, the other two columns are the number of stems connected to the drift tube and the stem diameter. All the stems at a given drift tube are assumed to have the same diameter. For post-coupler data, the other two columns are diameter and length of the post coupler. A DTL always has only one post coupler at a given drift tube location.

Drift-tube numbers refer to full drift tubes. Note that there is one less drift tube than cells in a cavity. A one-cell cavity such as a buncher has no full drift tubes. Drift tube 1 follows gap 1 in the cavity. Drift-tube numbers −1 and −2 refer to the low-energy and high-energy end walls.

The default configuration has one stem per drift tube plus dummy half stems on the end walls and no post couplers. The POST_data table provides a way to include frequency-shift and power calculations for the post couplers. Only those post couplers that appear in the table will be included. You can include dummy half posts on the end walls by using the special drift-tube numbers (−1 and −2) for the end walls.

If the STEM_data table appears in the control file, it supersedes the default stem configuration and the stem diameter provided with the STEM_DIAMeter keyword. The table provides a way to include frequency-shift and power calculations for multiple-stem DTLs. Only those stems that appear in the table will be included. You can omit an end-wall half stem by excluding it from the table.

c. Example stem configurations

Suppose the cavity of interest has 8 cells (7 full drift tubes). The following examples show how to specify several stem configurations. Example 1 shows the default stem configuration where all stems have the same diameter and there is one stem per drift tube and a half stem on each end wall.

```
STEM_data  ;Example 1.
–1      1    1.0
1       1    1.0
2       1    1.0
3       1    1.0
4       1    1.0
5       1    1.0
6       1    1.0
7       1    1.0
–2      1    1.0
ENDData
```

Example 2 is a two-stem DTL with 2.54-cm diameter stems, but no half stems on the end walls.

```
STEM_data  ;Example 2.
1    2    2.54
2    2    2.54
3    2    2.54
4    2    2.54
5    2    2.54
6    2    2.54
7    2    2.54
ENDData
```

Example 3 is a three-stem DTL with different stem diameters. The table indicates two 2.25-cm diameter stems on every drift tube and one 1.75-cm stem on the even-numbered drift tubes. It includes dummy half stems for both large stems on the end walls.

```
STEM_data  ;Example 3.
–1   2    2.25
1    2    2.25
2    2    2.25
3    2    2.25
4    2    2.25
5    2    2.25
6    2    2.25
7    2    2.25
–2   2    2.25
2    1    1.75
4    1    1.75
6    1    1.75
ENDData
```

d. Example post-coupler configurations

The following examples show how to specify post-coupler configurations. Example 1 specifies 1-cm-diameter, 7.5-cm-long post couplers at every drift tube, with dummy half posts on the end walls.

```
POST_data   ;Example 1.
–1   1.0     7.50
1    1.0     7.50
2    1.0     7.50
3    1.0     7.50
4    1.0     7.50
5    1.0     7.50
6    1.0     7.50
7    1.0     7.50
–2   1.0     7.50
ENDData
```

Example 2 specifies 1-cm-diameter post couplers at the even-numbered drift tubes, with as-built varying lengths, and with dummy half posts on the end walls.

```
POST_data   ;Example 2.
–1   1.0     7.50
2    1.0     7.48
4    1.0     7.51
6    1.0     7.55
–2   1.0     7.50
ENDData
```

5. CDTfish control-file keywords

The CDTfish control file defines the geometry for up to 50 problems. Table XIII-19 lists the control-file keywords used by CDTfish. The code also uses the common keywords listed in Table XIII-8. Many of these parameters are the same ones used to describe drift-tube cells in DTLfish and MDTfish. The outer cavity shape uses many of the parameters found in CCLfish. CDTfish is one of several tuning codes that has “paired keywords” for specifying the cavity length and gap length. Be sure to review the section that discusses these paired quantities.

a. Full-cavity or half-cavity problems and the number of gaps

CDTfish will set up either full cavities or half cavities depending on which keyword appears in the control file. The control-file keyword for N_g refers to the number of gaps for a full cavity for both types of calculation. The full length L of the problem geometry in CDTfish is related to the number of gaps N_g as follows:

$$L = \left(N_g - \frac{1}{2} \right) \beta\lambda ,$$

where $\beta\lambda$ is the distance traveled by the design particle in one rf period. The particle velocity β refers to an average value for the cell. CDTfish, like the tuning programs DTLfish and CCLfish, designs symmetric cavities. It is standard practice in cavity design

to ignore the change in velocity of the particle across the cavity. The total length of the problem geometry includes any bore-tube extensions δT_L and δT_R.

b. Gap-shift parameters in the CDTfish control file

CDTfish optionally corrects for longitudinal asymmetries in the gap fields by adjusting the location of the gap centers. This correction can be important for beam-dynamics calculations that use the drift-kick method (such as in the program Parmila). CDTfish makes the correction automatically when the start code is $N_v = 2$ or $N_v = 3$. For start codes $N_v = 4$ or $N_v = 5$, CDTfish uses the supplied values of the shifts $\delta g(i)$ on the GAP_Shift line, but does not try to adjust them during the calculation. For cavities tuned by this method, CDTfish runs SFO one additional time with the reference position Z_c for transit-time-factor integrals located at the each cell's electrical center. CDTfish obtains the location of the electrical center with respect to the original value of Z_c from the last completed Superfish run on the problem. You can force the code to use the cell's geometric center with the following setting in file SF.INI:

```
[CDTfish]
Zcenter = geometric
```

After tuning a cavity with start codes $N_v = 2$ or $N_v = 3$, CDTfish writes two lines in the control file of completed jobs. The GAP_Shift line contains the final values of the shifts $\delta g(i)$ applied to the gaps for $i = 1, N_g$. The ORIGINAL_Shift line contains the values $\delta g_o(i)$, which correspond to the values $\delta g(i)$ applied after the first SFO calculation. For the first Superfish run, the cavity has the nominal gap geometry for which $\delta g(i) = 0$ for all i. On subsequent Superfish runs, δg_o serves as a calibration of the effectiveness of moving the gaps.

CDTfish uses information from the SFO transit-time-factor integrals to calculate values for $\delta g(i)$ that align the cell's electrical centers with the nominal gap centers. You should only use the GAP_Shift and ORIGINAL_Shift lines in a CDTfish control file when recalculating fields for a problem previously solved by the code. If you adjust anything in the cell geometry, delete the GAP_Shift and ORIGINAL_Shift lines and let the code recalculate these quantities for the new geometry.

Table XIII-19. CDTfish control-file keywords.

Keyword	Symbol	Description
FULL_cavity		Set up the geometry of a full symmetric cell.
HALF_cavity		Set up the geometry of a the right half of a symmetric cell.
BORE_radius	R_b	Bore radius [cm].
BETA	β	Particle velocity relative to the speed of light.
LENGTH	L	Cavity length [cm].
DIAMeter	D	Cavity diameter [cm].
G_OVER_Beta_lambda	$g/\beta\lambda$	Ratio of gap length to $\beta\lambda$.
GAP_Length	g	Gap length between noses [cm].
E0_Normalization	E_0	E_0 value to use as the field normalization [MV/m].
E0T_Normalization	E_0T	Product of E_0T to use as the field normalization [MV/m].
NUMBER_of_gaps	N_g	Number of gaps in a full cavity.
GAP_Shift	$\delta g(n)$	Gap shifts for n = 1, N_g [cm].
ORIGINAL_Shift	$\delta g_o(n)$	Original gap shifts reported by SFO for nominal geometry [cm].
EQUATOR_flat	F_{Eq}	Length of straight section centered on the cavity outer wall [cm].
LEFT_BEAM_tube	δT_L	Length of bore tube extension on the low-energy end [cm].
RIGHT_BEAM_tube	δT_R	Length of bore tube extension on the high-energy end [cm].
INNER_CORNer_radius	R_{ci}	Radius joining cavity wall and straight face-angle segment [cm].
OUTER_nose_radius	R_o	Radius at the inner end of the straight face-angle segment [cm].
INNER_nose_radius	R_i	Radius near the bore [cm].
FLAT_length	F	Length of straight segment between the nose radii [cm].
CONE_angle	α_c	Cone angle on wall noses relative to horizontal [degrees].
SEPTUM_thickness	s	Thickness of the web or septum between cavities [cm].
DT_Diameter	d	Diameter of the drift tube at its midpoint [cm].
DT_CORNer_radius	R_c	Radius at the outer end of the face-angle segment [cm].
DT_OUTER_NOSE_radius	R_{do}	Radius at the inner end of the face-angle segment [cm].
DT_INNER_NOSE_radius	R_{di}	Radius near the bore [cm].
DT_FLAT_length	F_d	Length of straight segment between the nose radii [cm].
DT_STEM_Diameter	d_{Stem}	Diameter of the drift-tube stem [cm].
DT_STEM_Count	N_{Stem}	Number of stems on each drift tube.
DT_FACE_angle	α_f	Drift-tube face angle relative to vertical [degrees].
DT_OUTER_FACE_angle	α_{fo}	Face-angle for drift-tube faces next to outer gaps [cm].
DT_INNER_FACE_angle	α_{fi}	Face-angle for drift-tube faces next to inner gaps [cm].
RING_TYPE	*T-Ring*	Defines a tuning ring in a standard CCL cell.
RING_Width	W_{Ring}	Tuning ring width [cm].
RING_DY	Δy_{Ring}	Tuning ring distance below corner radius [cm].
RING_Angle	α_{Ring}	Tuning ring angle with a horizontal line [cm].
RING_Effect	Δf_{Ring}	Frequency effect of the tuning ring [MHz].
RING_Thickness	T_{Ring}	Thickness of the tuning ring [cm].
START	N_v	Starts a problem using current parameters.

During a CDTfish session, the code writes information about the gap-shift adjustments in the log file. For example, after the first Superfish run on a problem, CDTfish wrote the following data:

Gap-shift data

Cell	Nom. Center	Z0	Current δg	Total δg	Multiplier	δf
1	–4.925162	–4.793938	–0.131224	–0.131224	1.000000	–2.154500
2	4.925162	4.879617	0.045545	0.045545	1.000000	–0.752212

The "Nom. Center" column lists the nominal center of the cells and "Z0" lists the electrical centers calculated in SFO. The next column is just the difference between the "Z0" and "Nom. Center." After another Superfish run, the "Multiplier" column includes information that the code learned after applying the shifts calculated after the first run. The magnitude of the shifts applied at this iteration are equal to the "Current δg" time the "Multiplier." This is reflected in the cumulative applied shift reported in the "Total δg" column.

Gap-shift data

Cell	Nom. Center	Z0	Current δg	Total δg	Multiplier	δf
1	–4.925162	–4.901023	–0.024139	–0.160804	1.225421	–0.532477
2	4.925162	4.923182	0.001979	0.047615	1.045436	–0.036102

After a third iteration, the code refines the value of the Multiplier. After each iteration, the gap adjustments and their corresponding frequency shifts become smaller. When tuning a cavity, CDTfish takes into account the expected frequency shifts caused by adjusting the gaps.

Gap-shift data

Cell	Nom. Center	Z0	Current δg	Total δg	Multiplier	δf
1	–4.925162	–4.924636	–0.000526	–0.161451	–1.230349	–0.011865
2	4.925162	4.925127	0.000035	0.047651	1.046234	–0.000639

c. Using unequal face angles on drift tubes

CDTfish uses the face angle α_{fo} for drift tubes facing the cavity wall, and α_{fi} for drift tubes facing another drift tube. This feature allows the designer nearly to equalize the voltage gain E_0L_{cell} in all the cells, where E_0 is the average electric field and L_{cell} is the length of a cell.

SFO reports the value of E_0L_{cell} for each cell in its output file. You may need to try a few choices for α_{fo} and α_{fi} to get the desired results. You may not have complete freedom in choosing these face angles because of the way CDTfish constructs the drift tubes. The code draws each half of a drift tube separately. The entire face-angle segment plus its corner arc of radius R_c must fit on the same side of the cell boundary. The upper limit on the face angle depends on several parameters including the drift-tube length (equal to L_{cell} – g), drift-tube diameter d, bore radius R_b, corner radius R_c, nose radii R_{di} and R_{do}, and flat length F_d.

6. RFQfish control-file keywords

The RFQfish control file defines the geometry for up to 2500 problems. Table XIII-20 lists the control-file keywords used by RFQfish. The code also uses the common keywords listed in Table XIII-8.

Table XIII-20. RFQfish control-file keywords.

Keyword	Symbol	Description
RFQ_mode		Use a Dirichlet boundary on the left side of geometry.
DIPOLE_mode		Use a Neumann boundary on the left side of geometry.
PARameterName		Name of optional parameter in SFO file.
CORNer_radius	R_c	Outer corner radius (or curved-wall radius).
BREAK_out_angle	α_{bk}	Break-out angle from tip radius to vane-blank width.
RHO/R0	ρ/r_0	Ratio of tip radius RHO to aperture R0.
BLANK_Width	B_W	Half width of the blank on which the tip is machined.
BLANK_Depth	B_D	Depth of the vane blank from the RFQ axis.
VANE_ANGLE_1	α_1	Vane angle from the vane blank to the shoulder.
VANE_ANGLE_2	α_2	Vane angle from the shoulder to the base.
DATA		Lines following contain RFQ tuning parameters.
ENDData		End of the data table.

a. Defining the RFQfish tuning parameters

Each line between DATA and ENDData represents an RFQ cavity problem and contains the data listed in Table XIII-21. RFQfish writes a line in the Superfish problem description containing values of these parameters. The code also includes the data in the SFO file for use by the program SFOtable. The A DATA table can have up to 50 lines. Each line corresponds to a separate RFQ cell that the code will tune to the target frequency. The file can contain up to 50 DATA tables. Other keywords can appear before DATA sections to redefine parameters common to all problems within the next section. Only parameters in columns 3 through 8 can vary to tune the cell. If N_v is 1 or 2, then RFQfish runs Superfish to calculate the frequency, but it does not attempt to tune the cell. A nonzero entry for V_g is optional. If V_g is zero, then SFO normalizes fields to $E_0 = 1.0$ MV/m, where E_0 is the average electric field between the beam axis and the vane tip. A nonzero entry for vane width W also is optional. If W = 0, then the code substitutes $W = H \tan(\pi/8)$. Finally, there is no shoulder if $\alpha_2 = 0$. In this case, the entries for L_s and W_s are ignored. (Include two place holders for L_s and W_s.)

Table XIII-21. Data columns after the DATA keyword in the RFQfish control file.

Column	Symbol	SFO-file keyword	Description
1	r_0	R0	Average bore radius [cm].
2	V_g	Vgap	Gap voltage [MV].
3	ρ	Rho	Radius of curvature of vane tip [cm].
4	W_s	Wsh	Vane shoulder half width [cm].
5	L_s	Lsh	Vane shoulder length [cm].
6	W_b	Wbase	Vane base half width [cm].
7	H	Hvane	Vane height [cm].
8	W	Wvane	Vane half width [cm].
9	N_v		START code for RFQfish.

The average bore size of the RFQ cell is r_0. It is the distance in cm from the beam axis to the vane tip. In an RFQ with modulated vanes it is the radial aperture at the point of quadrupole symmetry. RFQfish ignores the vane-tip modulations, so r_0 is simply the distance between the cavity axis and the vane tip. A value for r_0 must appear in column 1 of a DATA table. It is not one of the adjustable parameters. The vane-tip radius of curvature is ρ in cm. The keyword RHO/R0 specifies the ratio of ρ/r_0 at the point of quadrupole symmetry. If you specify a value for the ratio RHO/R0, then RFQfish uses it to calculate ρ for each problem geometry. If RHO/R0 is specified, then use zero as a placeholder in the DATA table. If you do not specify RHO/R0, then you must supply the value of ρ.

The Data/EndData section may include a tenth column supplied by the user. Define the name of this data column on the ParameterName line near the beginning of the input file. If present, the value of the tenth column will appear in the list of setup parameters in the SFO file. Program SFOtable will include the data in its output plot file, if the SFOtable input file lists the name of the entry on an Include line. For example, suppose you add a column of longitudinal coordinates z that corresponds to each RFQfish problem. The RFQfish input file includes the line

```
ParameterName          z(cm)
```

The SFOtable input file might include the following lines:

```
Output_file            RFQ_A
Filename_prefix        RFQ_A
SEQuence_numbers       1 to 25
CartesianCoordinates
Include                R0
Include                Vgap
Include                Rho
Include                z(cm)        Abscissa
EndFile
```

The appearance of the word “Abscissa” on one of the Include lines makes that column of data the default abscissa when Tablplot opens the resulting plot file. The line CartesianCoordinates tells SFOtable not to search for parameters that SFO calculates for problems with cylindrical symmetry. Without this entry, SFOtable will be unable to read the RFQfish-generated SFO files correctly.

b. Field normalization in RFQfish

RFQfish uses the gap voltage V_g to normalize the fields to a particular value of the electric field for power calculations. For a nonzero value of V_g, the code normalizes the data so that $E_0 = V_g/2r_0$ where E_0 is the average electric field in the RFQ bore from the axis at r = 0 to the vane tip. RFQfish uses a default value of E_0 = 1.0 MV/m if the V_g entry is zero.

The code integrates the field between points (x, y) = (0, 0) and (r_0, 0). Since the axial field is zero for the quadrupole mode, the voltage difference from the axis to the vane tip is just half of the gap voltage V_g. We do not integrate between the points of closest approach of two vane tips. For the rf fields, the voltage difference at the points of closest approach will be reduced slightly from the results of an electrostatic calculation, which makes use of the quasistatic approximation. Thus the normalization performed by program SFO along y = 0 will agree with assumptions made in the RFQ design codes PARI and PARMTEQ.

7. SCCfish control-file keywords

The SCCfish control file defines the geometry for up to 100 problems.

Table XIII-22 lists the control-file keywords used by SCCfish. The code also uses the common keywords listed in Table XIII-8. Parameters that include a “LEFT_” or “RIGHT_” prefix take precedence over the keyword without the prefix.

Table XIII-22. SCCfish control-file keywords.

Keyword	Sym.	Description
LENGTH	L	Cavity length [cm].
DIAMeter	D	Cavity diameter [cm].
POST_Length	L_{Post}	Tuning post length for both sides of the cavity [cm].
LEFT_POST_Length	L_{Post}	Tuning post length on the left side of a full cavity.
RIGHT_POST_Length	L_{Post}	Tuning post length on the right side of a full cavity.
POST_Diameter	D_{Post}	Tuning post diameter on both sides [cm].
LEFT_POST_Diameter	D_{Post}	Tuning post diameter on the left side.
RIGHT_POST_Diameter	D_{Post}	Tuning post diameter on the right side.
OUTER_CORNer_radius	R_{co}	Radius at the cavity outer wall on both sides [cm].
LEFT_OUTER_CORNer_radius	R_{co}	Radius at the cavity outer wall on the left side .
RIGHT_OUTER_CORNer_radius	R_{co}	Radius at the cavity outer wall on the right side.
INNER_CORNer_radius	R_{ci}	Radius at cavity end wall and tuning post on both sides [cm].
LEFT_INNER_CORNer_radius	R_{ci}	Radius at cavity end wall and tuning post on the left side
RIGHT_INNER_CORNer_radius	R_{ci}	Radius at cavity end wall and tuning post on the right side.
OUTER_POST_radius	R_{po}	Radius at the end of the tuning post on both sides [cm].
LEFT_OUTER_POST_radius	R_{po}	Radius at the end of the tuning post on the left side.
RIGHT_OUTER_POST_radius	R_{po}	Radius at the end of the tuning post on the right side.
INNER_POST_radius	R_{pi}	Radius between face angle and vertical flat on both sides [cm].
LEFT_INNER_POST_radius	R_{pi}	Radius between face angle and vertical flat on the left side .
RIGHT_INNER_POST_radius	R_{pi}	Radius between face angle and vertical flat on the right side.
FLAT_length	F	Length of tuning-post vertical flat segment on both sides [cm].
LEFT_FLAT_length	F	Length of tuning-post vertical flat segment on the left side.
RIGHT_FLAT_length	F	Length of tuning-post vertical flat segment on the right side.
FACE_angle	α_f	Post face angle relative to vertical on both sides [degrees].
LEFT_FACE_angle	α_f	Post face angle relative to vertical on the left side.
RIGHT_FACE_angle	α_f	Post face angle relative to vertical on the right side.
START	N_v	Starts a problem with the current parameters.

G. Drift-tube nose shape in cylindrically symmetric cavities

Programs DTLfish, CCLfish, MDTfish, and CDTfish tune cylindrically symmetric accelerating cavities. These codes use the same basic nose shape for the separate drift tubes and for the nose on the cavity end walls. The inner-nose radius R_i defines an arc that starts tangent to the drift-tube bore and ends tangent to the optional vertical segment of length F. Cavity designers have found that for some applications a short flat on the drift-tube nose helps to reduce the peak surface electric field. If there is no flat segment, then the inner-nose arc connects to the outer-nose arc of radius R_o. The inner-nose arc subtends an angle of 90 degrees. The outer-nose arc starts tangent to either the flat segment or to the inner-nose arc. It ends tangent to the face-angle or cone-angle segment. If F = 0 and $R_i = R_o$, then the code draws only one continuous arc connecting the bore segment to the face-angle or cone-angle segment. Note the difference between the face angle α_f, which is an angle with the vertical, and the cone angle α_c, which is an angle with the horizontal.

CDTfish uses the same symbols as above for the cavity-wall noses, but different symbols for the noses on internal drift tubes. These nose shapes can be different and the symbols

correspond to different keywords in the CDTfish control file. The radii R_{di} and R_{do} are the drift-tube inner and outer nose radii, respectively, and are analogous to R_i and R_o. The length of the flat F_d is the analogue to the flat F on the cavity nose. For CCDTL cavities with three or more gaps, CDTfish allows asymmetric face angles for the drift tubes near each cavity wall. If they appear in the control file, CDTfish uses the inner and outer face angles α_{fi} and α_{fo} instead of α_f.

1. CCLfish cavities with no nose

In CCLfish, if CAVITY_shape = 1, the geometry is a simple disk-loaded structure, and the cell has no nose. A vertical line segment connects the outer-corner radius with the radius at the bottom of the septum (or disk). Only the cavity diameter D can vary to tune this type cell.

H. Cell lengths and gap lengths in tuning codes

The tuning codes allow you to enter some parameters in different ways. These "paired quantities" include

- the cavity length L and particle velocity β, and
- the gap length g and the ratio g/βλ.

In DTLfish, CCLfish, ELLfish, and CDTfish you can specify either physical lengths in cm for the cell and gap lengths or you can have the code calculate these quantities in terms of the particle velocity β. For ELLfish, this statement applies only to the length since the ELLfish geometry does not have a gap. In MDTfish, the individual cell lengths and gap lengths appear in the CELL_DATA table for each problem.

Keyword LENGTH is the full length L (in cm) of the cell or cavity along the beam axis. GAP_Length is the full gap length g in cm. See the figures for the DTL cell, CCL cell, elliptical cavity, and CCDTL cavity. Keyword BETA is the particle velocity β relative to the speed of light. The cell length has no default value, so a value for either LENGTH or BETA must appear in the control file before the first START line. If BETA appears, the code ignores L and calculates the cell length according to the equations:

$$L = \beta\lambda\frac{\Phi_{1/2}}{180} \qquad \text{for the DTL and CCL cells,}$$

$$L = \frac{2N_g - 1}{2}\beta\lambda \qquad \text{for the CCDTL cavity, or}$$

$$L = \beta\lambda/2 \qquad \text{for the elliptical cavity,}$$

where $\lambda = c/f$, c the speed of light in cm/s, f is the target resonant frequency of the cavity in Hz, $\Phi_{1/2}$ is the value of PHASE_length, the change in RF phase angle in degrees as the synchronous particle crosses the half cell used in the Superfish runs, and N_g is the number of gaps in the CCDTL cavity. We use the ½ subscript for the phase length of DTL and CCL cells to emphasize that the keyword for phase length corresponds to the problem geometry of a half cell, and not to the full length of the cell. In DTLfish, the default value

of $\Phi_{1/2}$ is 180 degrees, which corresponds to a 1βλ DTL cell. In CCLfish, the default value of $\Phi_{1/2}$ is 90 degrees, which corresponds to the usual CCL cell of length βλ/2.

Like the cell length, the gap length can be entered in terms of the particle velocity. Keyword G_OVER_Beta_lambda is the ratio g/βλ. The gap is one of the parameters that each program can adjust automatically to tune a cell. To vary the gap, use the appropriate START code described in the next section. After a problem has been solved, all four values for L, β, g, and g/βλ will appear in the control file of completed jobs. If both L and β appear the next time you run the code, it uses β to calculate the cavity length. Similarly, if both g and g/βλ appear, the code uses g/βλ to calculate the gap. Before restarting the code, you can either delete the lines you don't need or make them comment lines.

I. Using the START code to tune a cavity

The START code N_v tells a tuning program what to vary in order to tune the cavity to the target frequency. In DTLfish, CCLfish, ELLfish, CDTfish, and SCCfish the START code appears on the START line, which starts a calculation on the next problem. In MDTfish, the START code is on the CELL_data line for the table of entries defining each cell of a problem. In RFQfish, the START code appears on each line in the DATA section.

The following subsections give all the options for the START code in each tuning program. If N_v is 0, 1, or missing, then the tuning program runs Superfish to calculate the frequency without attempting to tune the cell to the target frequency. For RFQfish $N_v = 2$ also has the same effect. Allowed higher values of N_v vary a cell parameter, and negative values serve as place holders for completed jobs or jobs to be skipped during the present session. Making the START code negative to skip a problem provides a convenient way to redo part of a series of problems, keeping the same filenames and sequence numbers used earlier. The program includes skipped problems in the set of filenames it generates even if it does not run Superfish on the problem. The control file of completed jobs includes negative START codes for the jobs that were completed.

1. START codes for DTLfish

START codes for CCLfish

Table XIII-24 lists the START codes for CCLfish if the variable CAVITY_shape = 0 er cavity). The START code Nac cell) or 2 (special wall shape, buncher cavity). The START code Nv appears on the START line for each problem.

For CAVITY_shape = 1 (iris-loaded waveguide), only D can vary to tune the cell. In this same effect and behave as if START = 2.

Table XIII-23 lists the START codes for DTLfish. The START code N_v appears on the START line for each problem.

2. START codes for CCLfish

Table XIII-24 lists the START codes for CCLfish if the variable CAVITY_shape = 0 (standard coupled-cavity linac cell) or 2 (special wall shape, buncher cavity). The START code N_v appears on the START line for each problem.

For CAVITY_shape = 1 (iris-loaded waveguide), only D can vary to tune the cell. In this case, START codes 2, 3, 4, and 5 all have the same effect and behave as if START = 2.

Table XIII-23. START codes for DTLfish.

START code	Description
1	No tuning, calculate the fields and resonant frequency.
2	Adjust the cavity diameter D to tune the cell.
3	Adjust the drift-tube diameter d to tune the cell.
4	Adjust the gap g between the noses to tune the cell.
5	Adjust the drift-tube face angle α_f to tune the cell.
–1	Skip this problem.
–2	Skip this problem previously tuned by adjusting D.
–3	Skip this problem previously tuned by adjusting d.
–4	Skip this problem previously tuned by adjusting g.
–5	Skip this problem previously tuned by adjusting α_f.

Table XIII-24. START codes for CCLfish.

START code	Description
1	No tuning, calculate the fields and resonant frequency.
2	Adjust the cavity diameter D to tune the cell.
3	Adjust the septum thickness s to tune the cell.
4	Adjust the gap g between the noses to tune the cell.
5	Adjust the drift-tube cone angle α_c to tune the cell.
–1	Skip this problem.
–2	Skip this problem previously tuned by adjusting D.
–3	Skip this problem previously tuned by adjusting s.
–4	Skip this problem previously tuned by adjusting g.
–5	Skip this problem previously tuned by adjusting α_c.

3. START codes for ELLfish

Table XIII-25 lists the START codes for ELLfish. The START code N_v appears on the START line for each problem. The recommended procedure is to use $N_v = 2$ to tune half cells that occur in the interior part of a multicell structure and then use $N_v = 3$ to tune the full cell on the end of the structure. Fixing the outer diameter of the end cell at the value determined for the interior cells usually simplifies manufacturing the cavity. The full-cell calculation must includes a sufficiently long beam tube of the right side of the problem geometry.

The $N_v = 4$ and $N_v = 5$ options tune the end cell by varying the wall angle α_w for a half cavity or α_R for a full cavity. The difference between these two options is that the code allows the point of tangency with the bore tube to move for $N_v = 5$, but not for $N_v = 4$. The $N_v = 5$ option requires a nonzero length for the right-side beam tube. When using this method, ELLfish holds fixed the ellipse dimensions a and b and allows the point of tangency with the bore tube to move longitudinally. For all the other tuning options only the ratio a/b is fixed and the code solves for the values of a and b for an elliptical segment that ends at distance L/2 from the cavity midplane.

Table XIII-25. START codes for ELLfish

START code	Description
1	No tuning, calculate the fields and resonant frequency.
2	Adjust the cavity diameter D to tune the cell.
3	Adjust the outer corner radius R_c (R_R for full cavities) to tune the cell.
4	Adjust the slope α_w (α_R for full cavities) to tune the cell.
5	Adjust the slope α_w (α_R for full cavities) fixing the ellipse semiaxes a and b.
–1	Skip this problem.
–2	Skip this problem previously tuned by adjusting D.
–3	Skip this problem previously tuned by adjusting R_c or R_R.
–4 or –5	Skip this problem previously tuned by adjusting α_w or α_R.

4. START codes for MDTfish

Table XIII-26 lists the START codes for MDTfish, which can adjust either the cavity diameter or all of the drift-tube gaps to tune the cavity. The START code N_v appears on the CELL_data line in MDTfish. If you use $N_v = 3$ to adjust the gaps, the code adjusts all the gaps simultaneously. Since the gap length may vary from cell to cell, the code proportionally adjusts the gaps.

Table XIII-26. START codes for MDTfish.

START code	Description
1	No tuning, calculate the fields and resonant frequency.
2	Adjust the cavity diameter D to tune the cavity.
3	Adjust the gaps g between the noses to tune the cavity.
–1	Skip this problem.
–2	Skip this problem previously tuned by adjusting D.
–3	Skip this problem previously tuned by adjusting g.

5. START codes for CDTfish

Table XIII-27 lists the START codes for CDTfish, which can adjust either the cavity diameter or all of the drift-tube gaps to tune the cavity. The START code N_v appears on the START line in CDTfish. If you use $N_v = 3$ or $N_v = 5$ to adjust the gaps, the code adjusts all the gaps simultaneously. In CDTfish all the gaps are identical. For start codes $N_v = 2$ and $N_v = 3$, CDTfish uses data from program SFO to adjust the gap centers to make the transit-time integral S = 0.

Table XIII-27. START codes for CDTfish.

START code	Description
1	No tuning, calculate the fields and resonant frequency
2	Adjust the cavity diameter D to tune the cavity, while simultaneously adjusting the gap centers to make the transit-time integral S = 0.
3	Adjust the gaps g between the noses to tune the cavity, while simultaneously adjusting the gap centers to make the transit-time integral S = 0.
4	Adjust the cavity diameter D to tune the cavity, but do not adjust the gap centers.
5	Adjust the gaps g between the noses to tune the cavity, but do not adjust the gap centers.
–1	Skip this problem.
–2	Skip this problem previously tuned by adjusting D and gap centers.
–3	Skip this problem previously tuned by adjusting g and gap centers.
–4	Skip this problem previously tuned by adjusting D only.
–5	Skip this problem previously tuned by adjusting g only.

6. START codes for RFQfish

Table XIII-28 lists the START codes for RFQfish. The START code N_v appears in the last column of the DATA section. The first and second columns (r_0 and V_g) are not adjustable parameters. If N_v is 0, 1, 2, or missing, then RFQfish runs Superfish to calculate the frequency, but it does not attempt to tune the cell. For RFQ vanes without shoulders ($\alpha_2 = 0$), only ρ, W_b, H, or W can vary to tune the cell.

Table XIII-28. START codes for RFQfish.

START code	Description
1 or 2	No tuning, calculate the fields and resonant frequency.
3	Adjust the vane-tip radius ρ to tune the cavity.
4	Adjust the vane shoulder half width W_s to tune the cavity.
5	Adjust the vane shoulder length L_s to tune the cavity.
6	Adjust the vane base half width W_b to tune the cavity.
7	Adjust the vane height H to tune the cavity.
8	Adjust the vane half width W to tune the cavity.
–1 or –2	Skip this problem .
–3	Skip this problem previously tuned by adjusting ρ .
–4	Skip this problem previously tuned by adjusting W_s.
–5	Skip this problem previously tuned by adjusting L_s.
–6	Skip this problem previously tuned by adjusting W_b.
–7	Skip this problem previously tuned by adjusting H.
–8	Skip this problem previously tuned by adjusting W.

7. START codes for SCCfish

Table XIII-24 lists the START codes for SCCfish. The START code N_v appears on the START line for each problem.

Table XIII-29. START codes for SCCfish.

START code	Description
1	No tuning, calculate the fields and resonant frequency.
2	Adjust the cavity diameter D to tune the cell.
3	Adjust the gap (one or both post lengths) to tune the cell.
4	For full cells, Adjust the right post length $L_{Post,R}$ to tune the cell.
5	For full cells, Adjust the left post length $L_{Post,L}$ to tune the cell.
–1	Skip this problem.
–2	Skip this problem previously tuned by adjusting D.
–3	Skip this problem previously tuned by adjusting the gap (L_{Post}).
–4	Skip this problem previously tuned by adjusting $L_{Post,R}$.
–5	Skip this problem previously tuned by adjusting $L_{Post,L}$.

J. Drift-tube stems in DTLfish, MDTfish, and CDTfish

Programs DTLfish, MDTfish, and CDTfish provide the stem diameter d_{Stem} to SFO for frequency shift and power calculations. Superfish assumes cylindrical symmetry, and therefore cannot include the stem in the calculation of fields and resonant frequency. SFO calculates the frequency shift caused by the presence of the stems. It also calculates the power dissipated on the stem. These calculations are applications of the Slater perturbation theorem using fields calculated by Fish in the region occupied by the stem. Stem power and frequency shifts appear in a table near the end of the SFO output file.

MDTfish instructs SFO to include dummy half stems on the end walls of the cavity. For half stems, SFO calculates the net power added to the total power because the half stem covers up part of the end wall. The value of d_{Stem} entered on the STEM_diameter line has no effect if a STEM_data table specifies the stem configuration. In DTLfish and CDTfish, you can specify multiple stems on each drift tube by entering a value for N_{Stem} in the control file.

K. Converting old DATA/ENDDATA sections to the new format

Some very old versions of DTLfish and CCLfish used a table of entries between keywords DATA and ENDData. Each line in the table corresponded to a separate problem. The present DTLfish and CCLfish codes do not use keywords DATA, ENDData, EZERO, and EZEROT. The desired field normalization appears on either the E0_Normalization line or the E0T_Normalization line.

Table columns in DTLfish included the cavity length, cavity diameter, drift-tube diameter, gap, face angle, START code, and normalization parameter. Table columns in CCLfish were the same except that the septum thickness appeared in place of the drift-tube diameter and the cone angle replaced the face angle. The normalization parameter was linked to the presence of one of the keywords EZERO or EZEROT elsewhere in the control file.

1. Converting an old DTLfish file to the new format

Consider the following fragment from an old DTLfish control file.

```
EZERO
DATA
$ 1:L 2:D 3:d 4:g 5:ϕ            vary # EZERO[T]
 10.40477 48.13 14.25 2.567 5 4. 1
 13.56929 48.13 14.25 3.802 5 4. 1
 15.61927 48.13 14.25 4.679 5 4. 1
 17.40835 48.13 14.25 5.493 5 4. 1
ENDDATA
```

Other lines from this file remain unaffected. Next is a replacement section that converts this fragment to the new control-file format. Only the parameters that change need appear before the next START line.

```
E0_Normalization        1
LENGTH                  10.40477
DIAMeter                48.13
GAP_Length              2.567
DRIFT_TUBE_Diameter     14.25
FACE_angle              5
START                   4

LENGTH                  13.56929
GAP_Length              3.802
START                   4

LENGTH                  15.61927
GAP_Length              4.679
START                   4

LENGTH                  17.40835
GAP_Length              5.493
START                   4
```

2. Converting an old CCLfish file to the new format

Consider the following fragment from an old CCLfish control file.

```
OUTER_CORNer_radius     2.25
EZEROT

DATA
; 1:L                   2:D 3:septum        4:g     5:α     vary # EZERO[T]
 7.5 31 1.524 2.492 20 4. 1.73333
 8.0 31 1.524 2.750 20 4. 1.73333
ENDDATA
```

```
OUTER_CORNer_radius     2.5

DATA
; 1:L                   2:D 3:septum        4:g     5:α     vary # EZERO[T]
 8.5 31 1.524 3.009 20 4. 1.73333
 9.0 31 1.524 3.269 20 4. 1.73333
ENDDATA
```

Other lines from this file remain unaffected. Next is a replacement section that converts this fragment to the new control-file format. Only the parameters that change need appear before the next START line.

```
E0T_Normalization       1.73333
OUTER_CORNer_radius     2.25
LENGTH                  7.5
DIAMeter                31
GAP_Length              2.492
SEPTUM_thickness        1.524
CONE_angle              20
START                   4

LENGTH                  8
GAP_Length              2.750
START                   4

OUTER_CORNer_radius     2.5
LENGTH                  8.5
GAP_Length              3.009
START                   4

LENGTH                  9
GAP_Length              3.269
START                   4
```