A brief look at waveguide periodic structures using Fenics and Octave models

Technical Report · July 2022

DOI: 10.13140/RG.2.2.28961.68961

1 author:

George Slade
Orban Microwave Products
68 PUBLICATIONS **220** CITATIONS

SEE PROFILE

A brief look at waveguide periodic structures using Fenics and Octave models

G. William Slade

Abstract

This work started off as an experiment with using the Fenics 2019 finite element solver suite to handle periodic boundary conditions. In the course of this work, we assembled a moment method integral equation solver to verify the solutions coming from the finite element method. The moment method uses a spectral domain decomposition method that is comact and fast (that for this problem, may be preferable to the FEM for solution). However, the finite element method is suitable for handling very complicated geometries and is recommended for its generality, despite its computational resource requirements.

In this paper, we briefly review dispersion relationships for periodic structures as well as the basics of the moment method solution. We also give the modified variational formulation for the finite element solution that permits the use of periodic boundary conditions. An example calculation for a waveguide inductive wall is presented as a numerical experiment. The finite element and integral equation software is listed in the appendices.

I. Introduction

In the design of filters, antennas, traveling (or slow wave) devices, one often encounters periodic structures. These structures often exhibit interesting dispersion characteristics such as stopbands, slow wave behavior and negative group velocities. In this short paper, we give a bit of background to the problem and show how the FEnICS finite element package can be used to generate solutions by enforcing a periodic boundary condition that satisfies the Floquet-Bloch requirement. In fact, the original purpose of this work was to study the implementation of periodic boundary conditions in Fenics 2019 [1]. We compute the field profiles and dispersion relationships using both the full field eigenvalue method [2] as well as the scattering/transmission (S/T) parameter method of concatenating two-port unit cells. Note that in a waveguide, many modes may propagate at once. The full field method permits proper solutions where all modes are accounted for. The S parameter method assumes only a single propagating mode (and that th eports are sufficiently decoupled from one another with respect to evanescent modes). Note that the S parameter method can be augmented to treat multiple modes, but we did not do that here.

A. The periodic cell & the eigenvalue problem

The periodic structure involves stacking each identical cell one after the other. Mathematically speaking, it is convenient to use wave transmission parameters to model this. Consider the forward traveling waves (a^+, b^+) and negative traveling waves (a^-, b^-) in Figure 1. The block represents the waveguide discontinuity in one of the repeating cells.

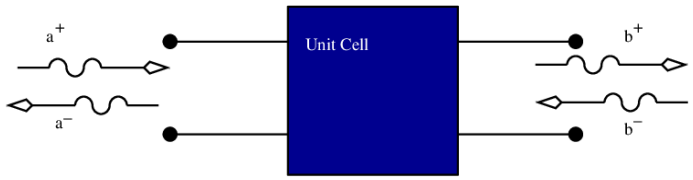


Fig. 1: The waveguide discontinuity unit cell showing forward and backward traveling waves.

The transmission parameters relate the port wave coefficients via

$$\left\{ \begin{array}{c} a^+ \\ a^- \end{array} \right\} = \left[\begin{array}{cc} T_{11} & T_{12} \\ T_{21} & T_{22} \end{array} \right] \left\{ \begin{array}{c} b^+ \\ b^- \end{array} \right\} \tag{1}$$

The matrix elements are given in terms of the S parameters by

$$T_{11} = \frac{1}{S_{21}} \tag{2}$$

$$T_{21} = -T_{12} = \frac{S_{11}}{S_{21}} \tag{3}$$

$$T_{22} = S_{21} - \frac{S_{11}^2}{S_{21}} \tag{4}$$

after rearranging the S parameter matrix.

In the periodic structure, the left port is equivalent to a phase-shifted version of the right port wave coefficients yield the eigenvalue problem:

$$e^{\gamma l} \left\{ \begin{array}{c} b^+ \\ b^- \end{array} \right\} = \left[\begin{array}{cc} T_{11} & T_{12} \\ T_{21} & T_{22} \end{array} \right] \left\{ \begin{array}{c} b^+ \\ b^- \end{array} \right\} \tag{5}$$

The periodic (Floquet) propagation factor γ is calculated from the eigenvalue problem

$$\det \left[\begin{array}{cc} T_{11} - e^{\gamma l} & T_{12} \\ T_{21} & T_{22} - e^{\gamma l} \end{array} \right] = 0 \tag{6}$$

which yields the characteristic equation

$$y^2 - (T_{11} + T_{22})y + 1 = 0 \tag{7}$$

where $y = e^{\gamma l}$. (Note that $T_{11}T_{22} - T_{12}T_{21} = 1$.) The solution is

$$e^{\gamma l} = \frac{T_{11} + T_{22}}{2} \pm \sqrt{\left(\frac{T_{11} + T_{22}}{2} \right)^2 - 1} \tag{8}$$

Setting

$$\cosh \gamma l = \frac{T_{11} + T_{22}}{2} \tag{9}$$

recovers the exponential in (8) when you recognize that

$$\cosh \gamma l \pm \sqrt{\cosh^2 \gamma l - 1} = \cosh \gamma l \pm \sinh \gamma l = e^{\pm \gamma l}. \tag{10}$$

Hence, the phase shift (and attenuation) suffered by the wave as it passes through the periodic network is given by

$$\gamma l = \operatorname{arccosh} \left(\frac{T_{11} + T_{22}}{2} \right). \tag{11}$$

Phase velocity along the structure is

$$v_p = \frac{k_0 c}{\Im \gamma} \tag{12}$$

and the group velocity

$$v_g = \frac{c}{dk_0/d\Im \gamma}. \tag{13}$$

II. Mode matching solution

Here, we take a look at the specific problem of an inductive waveguide diaphragm in a WR-90 rectangular waveguide (X-band, 8-12GHz, cross-section dimensions 0.9 x 0.4 in). We choose this structure because will scatter the TE_{10} mode only into other TE modes, thereby simplifying the mathematical treatment. This allows us to construct a fast, simple Octave code to solve the periodic problem, illustrating the techniques that could be implemented to solve the more general (full mode set) case. We assume that the metal diaphragm is centered at the $z = 0$ plane in the waveguide. This symmetry means we only need to

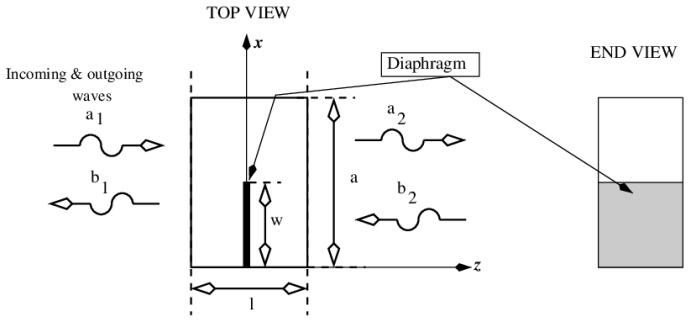


Fig. 2: The inductive waveguide septum with the metal vane centered longitudinally in the unit cell.

compute S_{11} and S_{21}. The TE_{m0} forward travelling modes in the waveguide transverse fields are given by

$$E_y = a_m \sin\left(\frac{m\pi x}{a}\right) e^{-\Gamma_m z} \tag{14}$$

$$H_x = -a_m \frac{\Gamma_m}{jk_0\eta_0} \sin\left(\frac{m\pi x}{a}\right) e^{-\Gamma_m z}, \tag{15}$$

where a is the waveguide width, k_0 is the free space wave number, η_0 is the free space wave impedance (377Ω) and the mode coefficients for mode m are a_m. By matching the transverse electric fields before and after the diaphragm (they must be continuous) and accounting for a unit current source in the plane of the diaphragm, the Green's function for the TE_{m0} electric field is found to be

$$G(x, x', z) = -\frac{jk_0\eta_0}{ab} \sum_m \frac{1}{\Gamma_m} e^{-\Gamma_m |z|} \sin\left(\frac{m\pi x}{a}\right) \sin\left(\frac{m\pi x'}{a}\right), \tag{16}$$

where

$$\Gamma_m = \sqrt{\left(\frac{m\pi}{a}\right)^2 - k_0^2} \tag{17}$$

and is, in general, complex. The source current distribution is found using field equivalence (Figure 3) by placing a short-circuit plane at $z = 0$ and evaluating the equivalent surface current over the area corresponding to the aperture not covered by the diaphragm using

$$\mathbf{J}_{inc} = \mathbf{n} \times \mathbf{H} \tag{18}$$

or using (15)

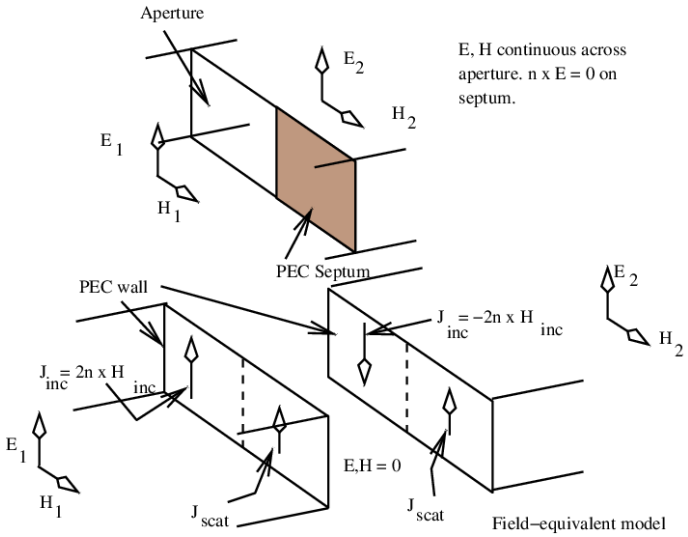


Fig. 3: The field equivalence model for defining the aperture and scattering currents.

$$J_{inc} = -\frac{2\Gamma_1}{jk_0\eta_0} \sin\left(\frac{\pi x}{a}\right), \tag{19}$$

over $w \leq x \leq b$ and is in the y direction only (hence dropping the vector notation) and the mode coefficient a_1 is set to unity (and dropped from (19).

We find the total electric field in the diaphragm region by applying the known aperture current (due to the incident wave) and that scattered by the (unknown) currents J_s on the diaphragm

$$E_y = -\frac{jk_0\eta_0}{a}\sum_m \frac{1}{\Gamma_m} e^{-\Gamma_m|z|}\sin\left(\frac{m\pi x}{a}\right)\left\{\frac{-2a_1\Gamma_1}{jk_0\eta_0}\int_w^a \sin\left(\frac{m\pi x'}{a}\right)\sin\left(\frac{\pi x'}{a}\right)\,dx' + \int_0^w J_{scat}(x')\sin\left(\frac{m\pi x'}{a}\right)\,dx'\right\} \tag{20}$$

We now have an equation that relates the total electric field to the incident field and an unknown current. In order to solve this equation for the unknown current, we expand the current in terms of a set of basis functions ϕ_i, where the current is expressed as a function of x on the vane:

$$J_{scat}(x') = \sum_{q=1}^{N} c_q\phi_q(x'), \tag{21}$$

where the aim is to solve for the set of coefficients c_q.

In order to solve for J_{scat} in (20), we use the Galerkin method with the current basis functions as testing functions on the septum boundary at $z = 0$, viz.:

$$0 = \int_0^w \phi_p(x)E_y(x)\,dx \tag{22}$$

This has the effect of invoking the boundary condition $E_y = 0$ on the metal vane. The evaluation of the integrals in (22) yield the following matrix equation, which we can now solve for the unknown diaphragm currents c_n,

$$\{b_p\} = [A_{pq}]\{c_q\} \tag{23}$$

where

$$[A_{pq}] = \sum_m \frac{1}{\Gamma_m}\int_0^w \phi_p(x)\sin\frac{m\pi x}{a}\,dx\int_0^w \phi_q(x')\sin\frac{m\pi x'}{a}\,dx' \tag{24}$$

$$\{b_m\} = \sum_p \frac{2\Gamma_1}{jk_0\eta_0\Gamma_m}\int_0^w \phi_p(x)\sin\frac{m\pi x}{a}\,dx\int_w^a \sin\frac{\pi x'}{a}\sin\frac{m\pi x'}{a}\,dx' \tag{25}$$

We can reduce the computational effort if we realize that the intergrals can be expressed as outer products of the Fourier Sine transforms of the basis functions. If we have

$$\hat{\phi}_{pm} = \int_0^w \phi_p(x)\sin\frac{m\pi x}{a}\,dx \tag{26}$$

which we only need to evaluate once in frequency-sweep applications. The matrix equation terms are constructed using the following matrix multiplications:

$$[A_{pq}] = [\hat{\phi}_{pm}][\text{diag}(1/\Gamma_m)][\hat{\phi}_{mq}]^T \tag{27}$$

$$\{b_p\} = [\hat{\phi}_{pm}][\text{diag}(2\Gamma_1/jk_0\eta_0\Gamma_m)]\{\hat{J}_{m\ inc}\} \tag{28}$$

Once we have the currents on the diaphragm, the S parameters are calculated using

$$S_{11} = S_{22} = -e^{-\Gamma_1^* l}\left(1 + \left(\hat{J}_{1\ inc} + \sum_p \hat{\phi}_{p1}c_p\right)\frac{jk_0\eta_0}{\Gamma_1 a}\right) \tag{29}$$

$$S_{21} = S_{12} = -e^{-\Gamma_1^* l}\left(\hat{J}_{1\ inc} + \sum_p \hat{\phi}_{p1}c_p\right)\frac{jk_0\eta_0}{\Gamma_1 a} \tag{30}$$

Because of the symmetry of the unit cell geometry in our chosen geometry, $S_{11} = S_{22}$ and due to the electromagnetic reciprocity inherent in this linear, passive network, $S_{21} = S_{12}$. It is possible to compute the scattering into other modes by

computing the higher spectral terms:

$$Mode2 = -e^{-\Gamma_2^* l}\left(\hat{J}_{2\ inc} + \sum_p \hat{\phi}_{p2} c_p\right)\frac{jk_0\eta_0}{\Gamma_2 a} \tag{31}$$

$$Mode3 = -e^{-\Gamma_3^* l}\left(\hat{J}_{3\ inc} + \sum_p \hat{\phi}_{p3} c_p\right)\frac{jk_0\eta_0}{\Gamma_3 a} \tag{32}$$

$$\tag{33}$$

where the current spectral terms are

$$J_{m\ inc} = \frac{2\Gamma_1}{jk_0\eta_0}\int_w^w \sin\left(\frac{\pi x}{a}\right)\sin\left(\frac{m\pi x}{a}\right)\,dx \tag{34}$$

III. Finite element solution

A. Propagation farctor via S parameters

As in the moment method, we start off by computing the S parameters from the FEM electromagnetic field solution. However, instead of working with a current solution, we use the electric field solution provided by the finite element solver. The reflection is given by [3]

$$S_{11} = S_{22} = \frac{\iint_{\partial\Omega_{\mathrm{InPort}}}(\mathbf{E}-\mathbf{E}^{\mathrm{inc}})\cdot\mathbf{e}_1\,dS}{\iint_{\partial\Omega_{\mathrm{InPort}}}\mathbf{E}^{\mathrm{inc}}\cdot\mathbf{e}_1\,dS}, \tag{35}$$

where $\mathbf{E}^{\mathrm{inc}}$ is the incident mode (forward traveling wave) and $\mathbf{e}_1$ is the fundamental mode field distribution. The integration domain is over the input waveguide port. The transmission through the cell is

$$S_{21} = S_{12} = \frac{\iint_{\partial\Omega_{\mathrm{OutPort}}}\mathbf{E}\cdot\mathbf{e_1}\,\mathbf{dS}}{\iint_{\partial\Omega_{\mathrm{OutPort}}}\mathbf{E}^{\mathrm{inc}}\cdot\mathbf{e}_1\,dS}, \tag{36}$$

In this case, the integration is carried out over the waveguide output port surface. The wave transmission matrix is computed and the eigenvalues are found exactly as in the preceding section.

Presently, using the Fenics finite element solver, we can only treat ports that permit a single mode. This is because Fenics does not permit one to build modal projection operators on the boundary in a straightforward way [3]. Use of high-order absorbing boundary conditions (which are a bit unwieldy) [4], [5] or perfectly matched layers [6] may be suitable to be studied at a later date.

B. Full periodic solution

Using the finite element method, we can generate a fully periodic solution by making the following substitution for the electric field variable [7]

$$e^{-\Gamma z}\mathbf{E} \rightarrow \mathbf{E}', \tag{37}$$

and enforce a strict periodic solution at the ends of the periodic cell (see Figure 4). Both $\mathbf{E}$ and Γ are complex numbers and must be treated accordingly. The curl of the electric field is modified accordingly:

$$\nabla\times\mathbf{E} = \nabla e^{\Gamma z}\times\mathbf{E}' + e^{\Gamma z}\nabla\times\mathbf{E}' = e^{\Gamma z}\left(\Gamma\hat{\mathbf{z}}\times\mathbf{E}' + \nabla\times\mathbf{E}'\right). \tag{38}$$

Hence, the weak form of the vector Helmholtz equation yields

$$\mathcal{L} = \iiint\left((\Gamma\hat{\mathbf{z}}\times\mathbf{E}' + \nabla\times\mathbf{E}')^*\cdot(\Gamma\hat{\mathbf{z}}\times\mathbf{E}' + \nabla\times\mathbf{E}') - k_0^2\mathbf{E}'^*\cdot\mathbf{E}'\right)\,dV \tag{39}$$

where the asterisk indicates complex conjugation. Minimization of this functional form yields an eigenvalue problem fo k_0^2. The boundary conditions on $\mathbf{E}'$ are exactly equivalent to those invoked for $\mathbf{E}$. Solving the eigenvalue problem directly produces the dispersion relation $k_0(\Gamma)$.

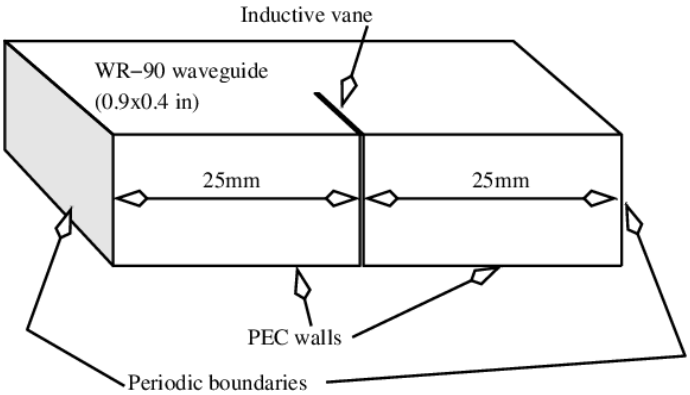


Fig. 4: The waveguide discontinuity unit cell showing the perodic boundaries on the waveguide ends.

This method does not rely on modal expansion at the ports, but recognizes that the field distribution at the ports can contain many modes and the only restriction is that the total field at one port be proportional to the field at the other port via a complex phase factor. This means that the ports can be positioned close to the waveguide discontinuities, where high-order evanescent modes can contribute to the port total field. The S parameter method relies on the fact that the fields of only a single mode dominate at the concatenated ports, hence all evanescent modes should be permitted to decay before encountering the port. This means that the unit cell must be long enough to ensure dominance of th efundamental mode at the ports for the S-parameter method.

The periodic boundary condition in full periodic solution is invoked by equating the degrees of freedom on the two end ports. This requires special handling of the port DoFs (see Appendix B) as well as ensuring that the mesh on one side is equivalent to the mesh on the other side (See Appendix A).

IV. Numerical solutions

For the finite element solutions, we must generate a mesh that exhibits geometric symmetry on the periodic faces. (See Appendix A for GMSH code to generate periodic mesh.) Figure 5 shows the result from the code in the appendix.

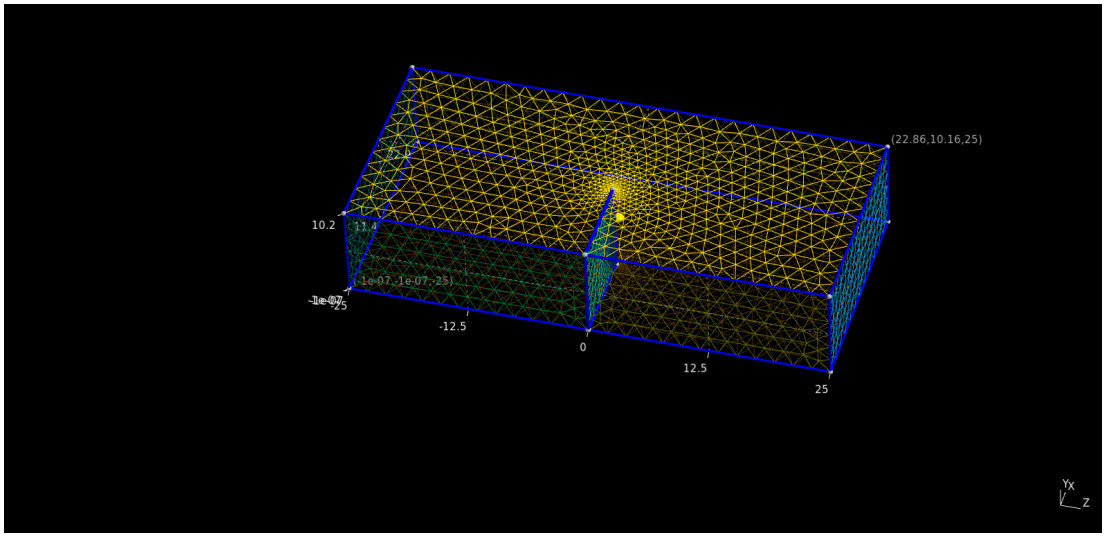
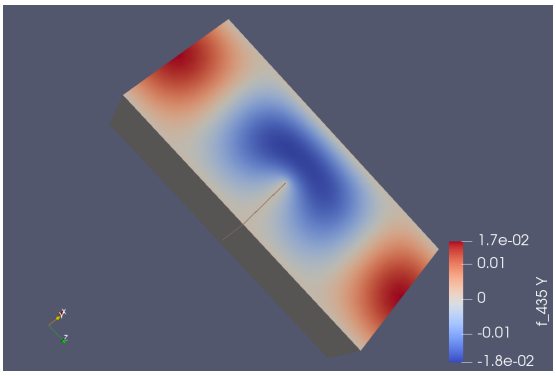
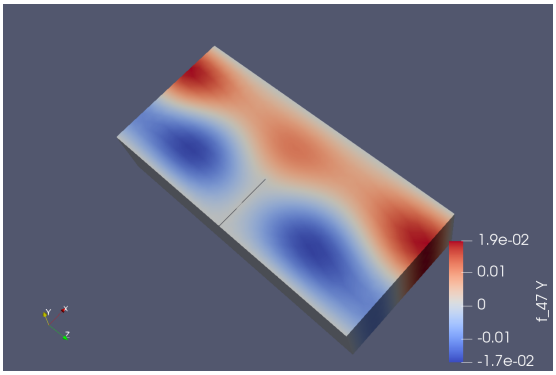


Fig. 5: The waveguide discontinuity unit cell mesh with periodic end faces.

Using the code in Appendix B, we generate solutions for the first several eigenvalues (that correspond to the frequency of operation). We can verify that the solutions are periodic by plotting the computed eigenfunctions, seen in Figure 6. Note that the low order field solution is dominated by TE_{10} field pattern on the ports (in Figure 6a). The higher order solution in Figure 6b has port fields that are a mixture of TE_{10} and TE_{20} modes. The S parameter technique will not work for this case (without



(a) Low order eigenfunction.



(b) Higher order periodic mode, showing TE_{10} and TE_{20} mode propagation.

Fig. 6: Two mode examples that use the fully periodic boundary condition on the waveguide ends.

some modification [7]). Basically, the field solution is indeed periodic at the port boundaries and we can have confidence in the solution.

We use the same mesh to solve the problem where we launch a wave from the top left and compute the reflected and transmitted fractions. The field profile is seen in Figure 7. The resulting S parameters are used to calculate the periodic structure dispersion characteristics.

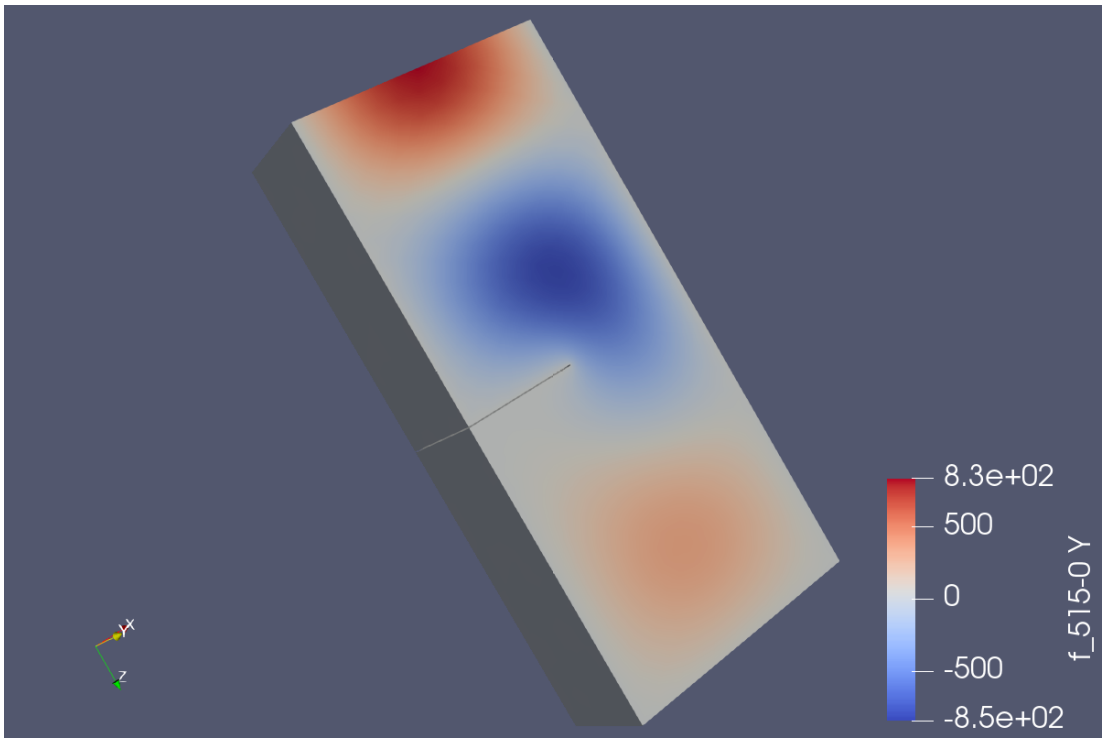


Fig. 7: Waveguide solution using S parameter method.

S parameters are also calculated using the mode matching method as further verification. The resulting dispersion data from all methods is overlayed in the plot in Figure 8.

The mode matching results are indicated by the green lines, and we clearly see the propagation and forbidden bands. The blue points are the results of the S-parameter FEM. The red points are the results of the fully periodic FEM solution. All points and curves coincide, highlighting the likelihood that the simulation results correspond to reality. The set of red points at the top of the plot is the mode that corresponds to the field plot in Figure 6b, which contains a mixture of TE_{10} and TE_{20} modes. The empty waveguide dispersion relationship is displayed as well (the black curve).

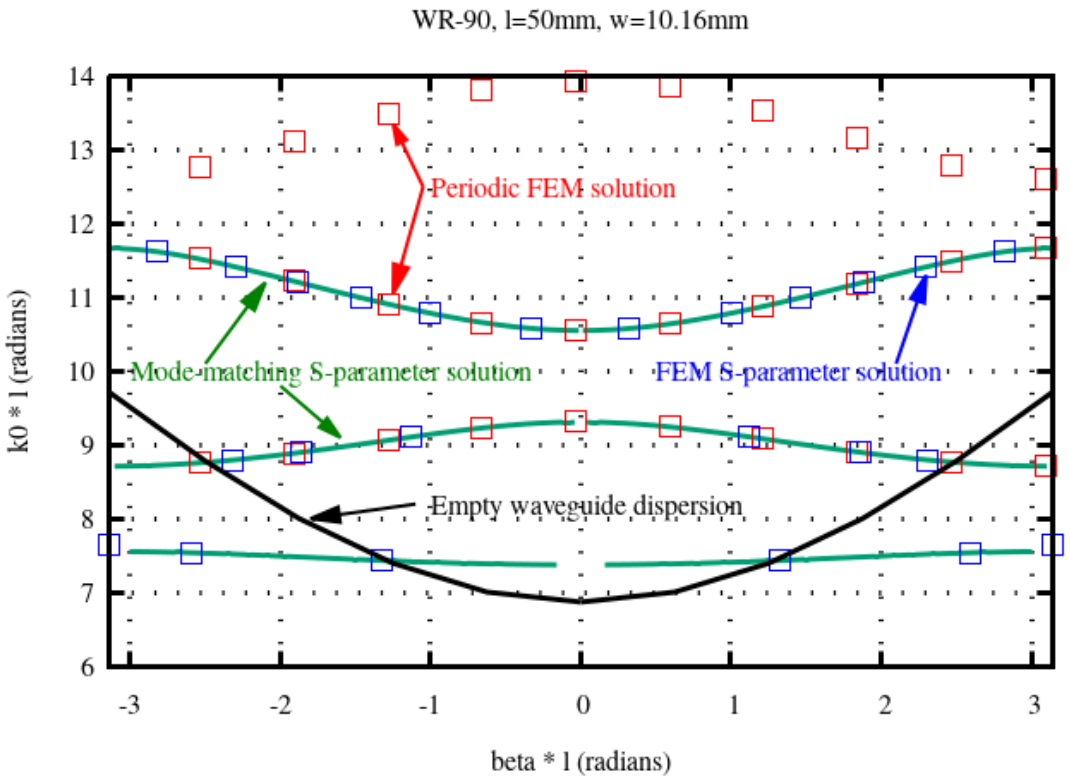


Fig. 8: The resulting dispersion properties for all three modeling methods..

The bottom of the empty waveguide curve shows the cutoff point of the empty waveguide. The first transmission mode has a somewhat higher cutoff frequency, which is consistent with the shunt inductive behavior of the inductive vanes (highpass/band-pass response). The red points at the top were not compared with the S parameter solution, because the S parameter simulations as developed, did not permit multimode solutions.

V. Conclusions

The application of periodic boundary conditions to Fenics FEM models worked well for this basic electromagnetic problem. The solutions coincided well with both moment-method and S-parameter finite element solutions. Note that the periodic boundary solution is also applicable in two and three dimensions, but we save that problem for another day.

References

[1] The Fenics Project, https://fenicsproject.org/.
[2] P. P. Silvester and R. L. Ferrari, *Finite Elements for Electrical Engineers*, Cambridge Uni. Press, 1996.
[3] G. Slade, "Solving the time harmonic Maxwell Equations in 3-D using the Fenics finite element package", Available at Researchgate.
[4] A. Ditkowski and D. Gottlieb, "On the Enquist-Majda absorbing boundary condition for hyperbolic systems," Available at Researchgate.
[5] A. Bayliss and E. Turkel, "Radiation boundary condition for wave like equations", *Comm. Pure Appl. Math.*,33(6), pp. 707-725, Nov. 1980.
[6] J. P. Berenger, "A Perfectly Matched Layer for the Absorption of Electromagnetic Waves," *J. Comp. Phys.*, vol. 114, no. 2, pp. 185-200, Elsevier, Oct. 1994.
[7] R. E. Collin, *Field Theory of Guided Waves,* McGraw-Hill, 1960.

VI. Appendix A: GMSH code for periodic mesh

```
// Inductive diaphragm in rect waveguide with periodic end BCs
a = 22.86;   // WR-90
b = 10.16;
w = 10.0;   // diaphragm width
```

```
l = 50.0;  // section length
t = 0.10; // Diaphragm thickness

eps = 1.0e-5;
refin = 0.03;

SetFactory("OpenCASCADE");
Box(1) = {0, 0, -l/2, a, b, l};
Box(2) = {0, 0, -t/2, w, b, t};
BooleanDifference(3) = {Volume{1}; Delete;}{Volume{2}; Delete;};

// Periodic boundary
Periodic Surface{7} = {5} Translate{0, 0, 50};

p = Point In BoundingBox{w-eps, -eps, -t/2-eps, w+eps, b+eps, t/2+eps};
MeshSize{p()} = refin;

Physical Volume("Waveguide") = {3};
Mesh.CharacteristicLengthMax = 2.0;
Mesh.CharacteristicLengthMin = 0.02;
Mesh.Algorithm3D = 4;
```

VII. Appendix B: Fenics Python code for periodic simulation

```
import meshio
from mpi4py import MPI as nMPI
from dolfin import *
import os, sys, traceback

tol = 1.0e-3


class PEC(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary

class PeriodicBoundary(SubDomain):
    def inside(self, x, on_boundary):
        return near(x[2], 25.0, tol) and on_boundary

    def map(self, x, y):
        if(near(x[2], -25.0, tol)):
                y[0] = x[0]
                y[1] = x[1]
                y[2] = x[2] + 50.0 # The two waveguide ends
        else:
            y[0] = -1000
            y[1] = -1000
            y[2] = -1000

class Slave(SubDomain):
    def inside(self, x, on_boundary):
        return near(x[2], -25.0, tol) and on_boundary

mesh = Mesh()
with XDMFFile("mesh.xdmf") as infile:
    infile.read(mesh)
mvc = MeshValueCollection("size_t", mesh, 3)
```

```
with XDMFFile("mesh.xdmf") as infile:
    infile.read(mvc, "VolumeRegions")
cf = cpp.mesh.MeshFunctionSizet(mesh, mvc)


info(mesh)

# Mark boundaries
sub_domains = MeshFunction("size_t", mesh, mesh.topology().dim() - 1)
sub_domains.set_all(3)
pec = PEC()
pec.mark(sub_domains, 0)
pbc = PeriodicBoundary()
pbc.mark(sub_domains, 1)
slv = Slave()
slv.mark(sub_domains, 2)

File("BoxSubDomains.pvd").write(sub_domains)

dk = 0.0125
for m in range(21):
#for m in range(1):
   comm = nMPI.COMM_WORLD
   mpi_rank = comm.Get_rank()


   gamma_r = Constant(0.0)
   gamma_i = Constant(dk*m)
# Set up function spaces
# For low order problem
   cell = tetrahedron
   ele_type = FiniteElement('N1curl', cell, 2, variant="integral") # H(curl) element
       for EM
   V2 = FunctionSpace(mesh, MixedElement([ele_type, ele_type]),
       constrained_domain=pbc)
   V = FunctionSpace(mesh, ele_type)
   u_r, u_i = TrialFunctions(V2)
   v_r, v_i = TestFunctions(V2)


#surface integral definitions from boundaries
   ds = Measure('ds', domain = mesh, subdomain_data = sub_domains)
# Volume regions
   dx_wg = Measure('dx', domain = mesh, subdomain_data = cf, subdomain_id = 1)

#Boundary condition dictionary
   boundary_conditions = {0: {'PEC' : (0, 0, 0)},
                          1: {'PBC': 1}}

   n = FacetNormal(mesh)
   az = Constant((0, 0, 1))

#Build PEC boundary conditions for real and imaginary parts
   bcs = []
   for i in boundary_conditions:
       if 'PEC' in boundary_conditions[i]:
           bc = DirichletBC(V2.sub(0), boundary_conditions[i]['PEC'], sub_domains, i)
           bcs.append(bc)
```

```
            bc = DirichletBC(V2.sub(1), boundary_conditions[i]['PEC'], sub_domains, i)
            bcs.append(bc)

    A = PETScMatrix()
    B = PETScMatrix()

    a = (inner((curl(v_r) + gamma_r * cross(az, v_r) - gamma_i * cross(az, v_i)), \
            (curl(u_r) + gamma_r * cross(az, u_r) - gamma_i * cross(az, u_i))) + \
            inner((curl(v_i) + gamma_r * cross(az, v_i) + gamma_i * cross(az, v_r)), \
            (curl(u_i) + gamma_r * cross(az, u_i) + gamma_i * cross(az, u_r))) \
            ) * dx_wg
    b = (inner(v_r, u_r) + inner(v_i, u_i)) * dx_wg
    L_dummy = (inner(Constant((0, 0, 0)), v_r) + inner(Constant((0,0,0)), v_i)) *
        dx_wg
    assemble_system(a, L_dummy, bcs, A_tensor = A) # Do this to get symmetric
        application of boundary conditions
    assemble_system(b, L_dummy, bcs, A_tensor = B)

    eigenSolver = SLEPcEigenSolver(A, B)
    eigenSolver.parameters["spectrum"] = "target_magnitude"
    eigenSolver.parameters["problem_type"] = "gen_hermitian"
    eigenSolver.parameters["spectral_transform"] = "shift-and-invert"
    eigenSolver.parameters["spectral_shift"] = 0.05
    eigenSolver.parameters["tolerance"] = 1.0e-14
    eigenSolver.parameters["maximum_iterations"] = 250
    eigenSolver.parameters["solver"] = "krylov-schur"
    eigenSolver.parameters["verbose"] = True
    print(eigenSolver.parameters.str(True))
    N=5
    eigenSolver.solve(N)

    for i in range(min(N, eigenSolver.get_number_converged())):
        r, c, rx, cx = eigenSolver.get_eigenpair(i)
        if mpi_rank == 0:
            kk = sqrt(r)
            print("Eigenvalue {0:<f} = {1:<f}".format(i, kk))
```

VIII. Appendix C: Octave code for moment method solution

```
clear all;

a = 22.86
w = 10.16

kx = pi / a
l = 50.0 # Section length
nmd = 1.0;

eta0 = 377.0

N = 15; # Number of current segments
M = 101; # Number of spectral terms
P = 5001; # Number of frequency points
fstart = 7.0; # Starting frequency (GHz)
df = 0.001; # Freq step

# Build spectral terms once
xp = linspace(0, w, N+1);
```

```
for m = 1:N+1
   yp = zeros(1,N+1);
   yp(m) = 1.0;
   printf("Basis function = %d\n", m);
   fflush(stdout);
   for n = 1:M
      ker = @(x) interp1(xp, yp, x, "linear").*sin(n*pi*x/a);
      if(m == 1)
         [s(n,m), err, np] = quadcc(ker, 0, w, 1.0e-6, [xp(m+1)]);
      else if(m == N+1)
         [s(n,m), err, np] = quadcc(ker, 0, w, 1.0e-6, [xp(m-1)]);
         else
            [s(n,m), err, np] = quadcc(ker, 0, w, 1.0e-6, [xp(m-1), xp(m), xp(m+1)]);
         endif
      endif
    endfor
endfor
for m = 1:M
  Cker = @(x) sin(pi * x / a) .* sin(m * pi * x / a);
  C(m) = 2.0 * quadcc(Cker, w, a, 1.0e-6);
endfor
printf("Finished DFT, Compute dispersion curves.\n");
fflush(stdout);
# Step thru frequencies
for q = 0:P
  f0 = fstart + df * q #freq (GHz)
  k0 = 2.0 * pi * f0 / 300.0;
  kk(q+1) = k0;
  for n = 1:M
     g(n) = sqrt(kx * kx * n * n - k0 * k0);
  endfor
  CCoef = sqrt(kx * kx - k0 * k0) / (I * k0 * eta0);
  # Incident field in aperture
  for m = 1:N+1
    for n = 1:M
      st(n, m) = s(n, m) / g(n);
    endfor
  endfor


# The matrix elements summing over the spectral terms!
   A = st'*s;

#Source integral (RHS) again summing over the spectral terms
   rhs = (C * CCoef) * st;

#Current on metal septum
   cc = inverse(A) * rhs'

# Calculate S params

  S11 = -exp(-conj(g(1))*1)*(1.0 + ((-C(1)*CCoef) + s(1,:) * cc)  * I * k0 * eta0 /
      (a * g(1)))
  S21 = -exp(-conj(g(1))*1)*((-C(1)*CCoef) + s(1,:) * cc)  * I * k0 * eta0/ (g(1) *
      a)

# Scattering into second and third modes
```

```
    Sm2 = -exp(-conj(g(2))*1)*(-(C(2)*CCoef) + s(2,:) * cc)  * I * k0 * eta0/ (g(2) *
        a)
    Sm3 = -exp(-conj(g(3))*1)*(-(C(3)*CCoef) + s(3,:) * cc)  * I * k0 * eta0/ (g(3) *
        a)

#T parameters for concatenation
    T(1,1) = S21 - S11*S11/S21;
    T(1,2) = S11 / S21;
    T(2,1) = - T(1,2);
    T(2,2) = 1.0 / S21;
    Pt = [0.0, 1.0; 1.0, 0.0]; # Permutation
    Ti = Pt * inverse(T) * Pt; # Transformed for forward concatenation

    gl(q+1) = acosh((T(1,1) + T(2,2)) / 2);
    fflush(stdout);
  endfor
  fp = fopen("Dispersion.txt", "w");
  prow = 1;
  for q = 1:P+1
     if(abs(real(gl(q))) < 1.0e-4)
        prow = 1;
        fprintf(fp, "%f %f %f\n", kk(q), real(gl(q)), imag(gl(q)));
      else
        if(prow == 1)
           fprintf(fp, "\n");
           prow = 0;
        endif
     endif
  endfor
  fclose(fp);
```