Appendix A
Projects

This appendix features five computer projects that cover the most important aspects of the theoretical material in this book: (1) convergence and extrapolation, (2) finite differences in the frequency domain for a 1D electromagnetic problem, (3) finite-difference time-domain for a 3D electrodynamic problem, (4) finite elements for a 2D eigenvalue problem, and (5) method of moments for a 2D electrostatic problem. Each computer project is presented in terms of a continuum formulation of the physical situation, and for some of the problems a part of its numerical treatment is also included. Next, a number of assignments are listed such as the derivation of the complete numerical scheme, the computer implementation, and a sequence of numerical tests. The assignments can be assessed by, for example, (1) a written report that is reviewed, (2) oral examination with or without access to a computer, or (3) a presentation in front of a larger group of students and teachers. Access to a computer allows for an exploratory and interactive testing of the computer implementation that may be difficult to achieve otherwise.

In the written or oral assessment, it is important that students attempt to provide mathematical and logical arguments to support the choices, derivations, implementations, results, and conclusions that constitute the computer project work. Such a presentation could consist of the following parts:

- Description of the continuum problem.
- Description of the numerical algorithm by means of suitable derivations together with their computer implementation. Note that it is essential that the computer implementation be well documented: (1) input variables for each function, (2) output variables for each function, (3) purpose and usage for each function, and (4) some sort of overall description of the program and its functions and scripts.
- Presentation of the numerical tests together with interpretations of the results that relate to the theoretical foundation in terms of, for example, the order of convergence, extrapolation, and estimation of the numerical error.
- Conclusion of the computer project work, which may include (1) brief summary of the work, (2) the main results, and (3) important implications of these results.

T. Rylander et al., *Computational Electromagnetics*, Texts in Applied Mathematics 51, DOI 10.1007/978-1-4614-5351-2,

A.1 Convergence and Extrapolation

A.1.1 Problem Description

In electrostatic problems, the surface charge density is singular close to sharp metal edges or corners. For a computation of the capacitance, this is problematic when the total charge on such a conducting body must be computed to determine the capacitance. Consider a situation with a given surface charge density that can be described without errors by means of an analytical formula. For example, the surface charge density close to the edge of a parallel plate capacitor varies as $x^{-(\pi-\beta)/(2\pi-\beta)}$, where x is the distance to the edge and β is the angle subtended by the metal edge as described in Sect. 7.2.3. This leads to integrals of the type

$$\int_a^b x^\xi dx, \tag{A.1}$$

where $0 \leq a < b$. Here, the case $a > 0$ yields a regular integrand and, from a mathematical viewpoint, this type of integrand features the general behavior of the surface charge distribution on a smooth metal surface. Furthermore, $a = 0$, in combination with $\xi < 0$, yields a singular integrand, which corresponds to the surface charge distribution in the vicinity of a sharp metal edge. (Note that according to the derviations in Sect. 7.2.3, an infinitely thin metal plate with $\beta = 0$ yields $\xi = -1/2$, which is the smallest possible value for ξ for such an electrostatic situation. However, we also investigate other cases, such as $\xi = -3/2$, subsequently since it provides additional understanding of some mathematical difficulties that are associated with singularities.)

A.1.2 Assignments

Implementation

Write a program that evaluates the integral (A.1) by means of midpoint integration. The program should allow for midpoint integration with n subintervals of length $h = (b-a)/n$. In what follows, the value of the numerically evaluated integral is denoted by $I_{\text{midp}}(h)$. The analytically calculated value of the integral (A.1) is denoted I_0.

Numerical Test #1

Execute your program with $a = 1$ and $b = 2$ for $\xi = -3/2, -1/2, 1/2, 3/2$.

- Evaluate the absolute error $e(h) = |I_{\text{midp}}(h) - I_0|$ as a function of the resolution controlled by h, where you should use the expression for I_0 that you get

from analytical integration. Does the computed result converge to the analytical answer? What is the order of convergence? Does the order of convergence agree with what you expect from an analysis of the problem?
- Extrapolate the numerically computed result to zero cell size: fit the function $I_{\text{model}}(h) = c_0 + c_\alpha h^\alpha$ to the computed data under the assumption that the constants c_0, c_α, and α are unknown. How well does the model $I_{\text{model}}(h)$ compare with the computed data $I_{\text{midp}}(h)$? Compare the extrapolated value c_0 and the estimated order of convergence α with the analytical results. Test your extrapolation method on different sets of computed data. How do the data influence the possibilities for accurate extrapolation?

Numerical Test #2

Execute your program with $a = 0$ and $b = 2$ for $\xi = -3/2, -1/2, 1/2, 3/2$.

- Evaluate the absolute error $e(h) = |I_{\text{midp}}(h) - I_0|$ as a function of the resolution controlled by h, where you should use the expression for I_0 that you get from analytical integration. Does the computed result converge to the analytical answer? What is the order of convergence? Does the order of convergence agree with what you expect from an analysis of the problem?
- Extrapolate the numerically computed result to zero cell size: fit the function $I_{\text{model}}(h) = c_0 + c_\alpha h^\alpha$ to the computed data under the assumption that the constants c_0, c_α, and α are unknown. How well does the model $I_{\text{model}}(h)$ compare with the computed data $I_{\text{midp}}(h)$? Compare the extrapolated value c_0 and the estimated order of convergence α with the analytical results. Test your extrapolation method on different sets of computed data. How do the data influence the possibilities for accurate extrapolation?

A.2 Finite Differences in the Frequency Domain

A.2.1 Problem Description

Consider an electromagnetic plane wave that propagates toward a large flat window of glass, as shown in Fig. A.1. We wish to compute the reflected and transmitted wave. The glass window has a thickness of $2a$.

The material parameters are $\epsilon(x)$, $\mu(x) = \mu_0$, and $\sigma(x)$ in the glass, where $-a \le x \le a$. The medium outside the window is air with $\epsilon(x) = \epsilon_0$, $\mu(x) = \mu_0$, and $\sigma(x) = 0$ for $|x| > a$. The total electric field satisfies the differential equation

$$-\frac{d^2 E_z(x)}{dx^2} + \mu_0 \left[j\omega\sigma(x) - \omega^2 \epsilon(x) \right] E_z(x) = 0. \tag{A.2}$$

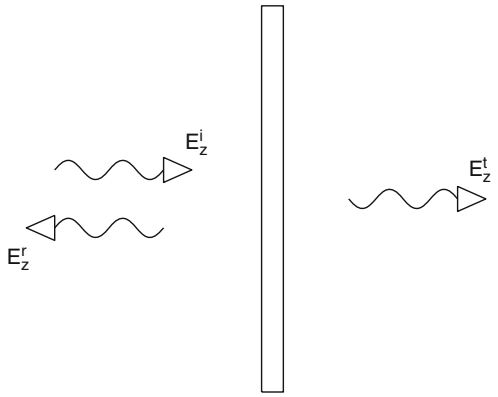

Fig. A.1 Glass window of thickness $2a$ with an incident field E_z^i, reflected field E_z^r, and transmitted field E_z^t

A.2.2 Assignments

Here we introduce some of the techniques used in electromagnetic scattering problems. One such technique is to impose the incoming wave by matching an expression for the incoming wave to the numerical solution in the vacuum region. The matching is done in a vacuum (outside the scatterer) where we know analytically how the incoming field behaves. In a similar manner, the reflected and transmitted waves can be described in the vacuum region. Following this procedure for our 1D problem we can do the matching of the discretized interior region to the fields outside at two points $x = \pm b$. We need to have $b > a$, so that the matching points are in the vacuum.

Formulate Boundary Condition

The first task is to derive the appropriate boundary conditions at $x = -b$ and $x = b$ analytically. Here is some guidance.

Let the incident field be $E_z^{\mathrm{i}}(x) = E_0^{\mathrm{i}} \exp(-jk_0x)$. Introduce the reflected field as $E_z^{\mathrm{r}}(x) = E_0^{\mathrm{r}} \exp(+jk_0x)$ and the transmitted field as $E_z^{\mathrm{t}}(x) = E_0^{\mathrm{t}} \exp(-jk_0x)$. According to these expressions, the total field is (1) the superposition of the incident and reflected field for the region $x < -a$ and (2) equal to the transmitted field for the region $x > a$. Equation (A.2) yields the dispersion relation $k_0 = \omega/c_0$ for the vacuum region, where $c_0 = 1/\sqrt{\epsilon_0\mu_0}$.

What is a priori known and unknown in the expressions above? How can this information be used to formulate the appropriate boundary conditions at $x = -b$ and $x = b$, respectively? The boundary conditions should only involve already known information (such as the incident wave) and the desired solution E_z and its first derivative, where E_z is the total electric field. (Consequently, we wish to find boundary conditions that do not explicitly involve the scattered electric field

since this is unknown before the scattering problem is solved.) Note that b is quite arbitrary as long as it is larger than a.

Generate Grid and System Matrix

We use two different grids:

`G1` The first grid is chosen such that the material interfaces $x = \pm a$ fall in between grid points. We use the grid points $x_n = (n + \frac{1}{2})\Delta x$ with $\Delta x = a/N$, an integer $N \geq 3$, and $n = -2N, -2N + 1, \ldots, 2N - 1$.

`G2` The second grid is chosen such that the material interfaces $x = \pm a$ fall on grid points. We use the grid points $x_n = n\Delta x$ with $\Delta x = a/N$, an integer $N \geq 3$, and $n = -2N, -2N + 1, \ldots, 2N$.

These grids discretize a region of (roughly) length $4a$ with at least three grid points in each vacuum region outside the window.

Denote the unknowns at the grid points by ζ_n, i.e., $E_z(x_n) = \zeta_n$. Discretize the differential equation (A.2) and your boundary conditions using finite differences, both with an error that is proportional to h^2. The boundary condition involving no higher than first derivatives is best centered on the half-grid. Alternatively, a numerical boundary condition can be formulated on the integer grid if more than two grid points are used.

Using the boundary conditions and the differential equation, we have a system of linear equations $\mathbf{Az} = \mathbf{b}$ to solve, where $\mathbf{z} = [\zeta_1, \zeta_2, \ldots, \zeta_{N_{\text{gp}}}]^T$ and N_{gp} is the number of grid points. Write down the matrix $\mathbf{A}$ and the right-hand-side vector $\mathbf{b}$ for the special case where $N = 3$ for the discretization `G1`. What are the similarities and differences when you follow the same procedure for the discretization `G2`? What may the implications be and how can you handle this?

Find a way to compute the reflection coefficient R and transmission coefficient T from the numerical solution, where the following definitions are used:

$$R = \frac{E_0^{\text{r}}}{E_0^{\text{i}}}, \tag{A.3}$$

$$T = \frac{E_0^{\text{t}}}{E_0^{\text{i}}}. \tag{A.4}$$

Implementation

Implement your numerical algorithm for an arbitrary $N \geq 3$ and both discretizations `G1` and `G2`. Given input variables that describe the physical situation and its discretization, the implementation should yield the reflection and transmission coefficients as output variables.

Numerical Test #1

Test your implementation on the case where the glass window has constant relative permittivity ϵ_r and conductivity σ. The reflection and transmission coefficients can be calculated analytically in this case and are given by

$$R = \frac{(k_0^2 - k_1^2)}{\Delta} e^{j2ak_0}(e^{j4ak_1} - 1),$$
$$T = \frac{k_0 k_1}{\Delta} 4e^{j2a(k_0+k_1)}, \tag{A.5}$$

where $\Delta = (k_0 + k_1)^2 e^{j4ak_1} - (k_0 - k_1)^2$, $k_0 = \omega/c_0$, and $k_1 = \sqrt{\epsilon_r k_0^2 - j\omega\mu_0\sigma}$.

Use the thickness $a = 2$ cm in combination with the constant material parameters $\epsilon_r = 2.5$ and $\sigma = 0.02$ S/m.

For the frequency $\omega = 3 \cdot 10^9$ rad/s, compute R and T numerically on the discretization `G1` for a set of appropriately chosen values of N. Do the numerically computed values of R and T converge toward the analytical values? Which order of convergence do you find?

Numerical Test #2

Now repeat the preceding test for the discretization `G2`. Did this change the convergence properties? If so, why? Incidentally, how do you choose the permittivity at $x = \pm a$?

Numerical Test #3

Also, compute R and T as functions of frequency between 0.1 and 10 GHz. You can use your favorite discretization (`G1` or `G2`) with a fixed value of N. How does the error change with respect to the frequency? Explain your findings.

Numerical Test #4

Compute R and T as a function of frequency between 0.1 and 10 GHz for an inhomogeneous window in the region $|x| \leq a$ with the following material parameters:

$$\sigma(x) = 0.02\, p(x),$$
$$\epsilon(x) = \epsilon_0 \left[1 + 5p(x)\right],$$

where

$$p(x) = 1 - \left(\frac{x}{a}\right)^2$$

is a parabolic profile with $p(\pm a) = 0$.

A.3 Finite-Difference Time-Domain Scheme

A.3.1 Problem Description

Waveguides and filters are important components of many complex microwave systems. Here we consider the characteristic features for some relatively simple structures that provide a filtering functionality in waveguides. The quantities of interest are the reflection and transmission coefficients as a function of frequency. In what follows, we will limit the discussion to waveguide structures with rectangular cross sections and a finite-difference time-domain (FDTD) scheme [80, 93].

The FDTD model deals with the part of the waveguide that contains the filtering structure. At each of the two ends of the filter, a shorter section of a rectangular waveguide is attached and truncated by a port for computational modeling purposes. (The physical waveguide would normally extend beyond the ports, but that part is not included in the computational model considered here.)

Modal Representation for a Rectangular Waveguide

In an air-filled rectangular waveguide with the transverse dimensions L_x and L_y, we can decompose the electric and magnetic fields into *transverse electric* (TE) and *transverse magnetic* (TM) modes; see [19] for a detailed discussion. Each mode has its own propagation constant

$$k_z = \sqrt{\left(\frac{\omega}{c}\right)^2 - k_t^2}, \tag{A.6}$$

where k_t^2 are the eigenvalues of the transverse problem for H_z (TE case) or E_z (TM case), i.e.,

$$k_t^2 = \left(\frac{n_x \pi}{L_x}\right)^2 + \left(\frac{n_y \pi}{L_y}\right)^2. \tag{A.7}$$

For TE modes, n_x and n_y are nonnegative integers that satisfy $n_x + n_y > 0$. For TM modes, both n_x and n_y are positive integers.

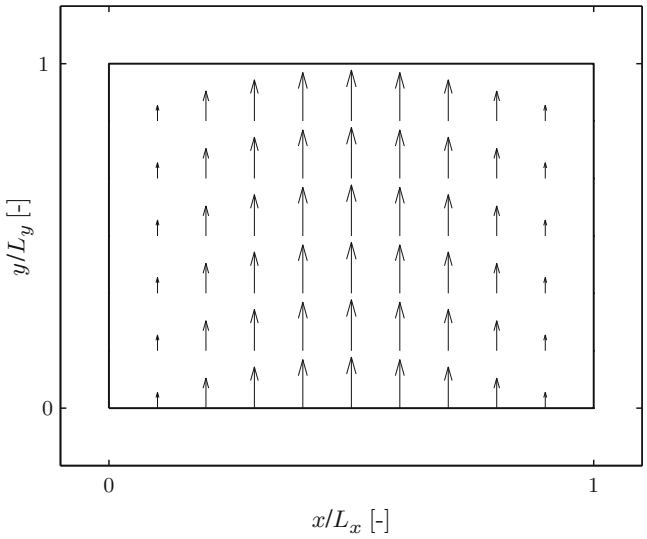

Fig. A.2 Modal field of TE_{10} mode

Numbering the modes from 1 to ∞, we can express the electric field in the waveguide as a superposition of both TE and TM modes by

$$\boldsymbol{E}(x, y, z, t) = \sum_{m=1}^{\infty} V_m(z, t)\, \boldsymbol{e}_m(x, y), \tag{A.8}$$

where $V_m(z, t)$ is the modal amplitude, or *voltage*, of mode m (which can be either a TE or a TM mode) and $\boldsymbol{e}_m(x, y)$ is its modal field. The modal field, $\boldsymbol{e}_m(x, y)$, for the TE_{10} mode is shown in Fig. A.2.

In what follows, we will consider situations where the frequency range of interest and the dimensions (L_x, L_y) of the waveguide are chosen such that k_z is real for the TE_{10} mode and imaginary for all other modes. Thus, the only mode that propagates is the TE_{10} mode. All the other modes are evanescent and decay exponentially along the waveguide axis. Consequently, the TE_{10} mode is the only mode present at a sufficiently large distance from any source or irregularity in the waveguide that may excite higher-order modes.

Computation of Scattering Parameters

A filter can be characterized in terms of its reflection and transmission coefficient, and in a more general setting, these are often referred to as the scattering parameters, or simply the S-parameters. Fig. A.3 shows a rectangular waveguide (without the filtering structure) that is truncated at two ports for computational modeling

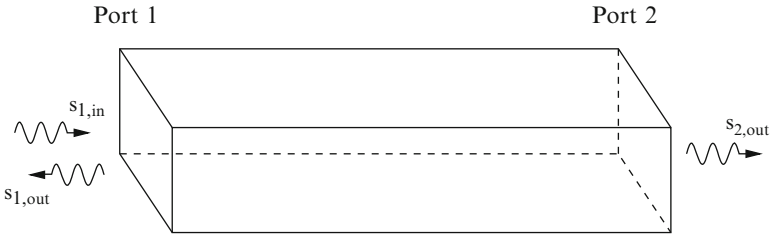

Fig. A.3 Illustration of incoming and outgoing waves

purposes. The S-parameters can be computed given the relation between the amplitudes of the TE_{10} mode at the ports: (1) an incident wave is launched at one port, (2) the reflected wave is recorded at the same port, and (3) the transmitted wave is recorded at the other port.

Let $s_{1,\text{in}}(t)$ be the amplitude of the incoming TE_{10} wave at port 1, and let $s_{1,\text{out}}(t)$ and $s_{2,\text{out}}(t)$ be the amplitudes of the outgoing TE_{10} waves at ports 1 and 2, respectively. The Fourier transform of these signals gives $S_{1,\text{in}}(\omega)$, $S_{1,\text{out}}(\omega)$, and $S_{2,\text{out}}(\omega)$. The relation between the amplitudes at the two ports is usually described by the so-called S-parameters:

$$S_{11}(\omega) = \frac{S_{1,\text{out}}(\omega)}{S_{1,\text{in}}(\omega)}, \tag{A.9}$$

$$S_{12}(\omega) = \frac{S_{2,\text{out}}(\omega)}{S_{1,\text{in}}(\omega)}. \tag{A.10}$$

The scattering parameter S_{11} is recognized as the reflection coefficient and S_{12} as the transmission coefficient. A further extension to an n-port network is rather straightforward and for such cases it is convenient to represent the S-parameters in matrix form, an $n \times n$ scattering matrix $\mathbf{S}$ with the elements S_{ij}.

Numerical Modeling

The interior of the waveguide is discretized by an FDTD grid [80, 93]. A wave can be launched at one of the ports and then propagated through the waveguide by means of Maxwell's equations represented by the FDTD scheme applied to the grid in between the ports. Consequently, a filtering structure can be modeled in detail by the FDTD scheme and its reflection and the transmission coefficient computed from the fields at the ports.

The special type of boundary condition that is required at the ports is already implemented in the MATLAB program provided as a starting point for the tasks below. However, it is useful and interesting to have some understanding of the

port algorithm. The algorithm is briefly summarized as follows (see [3] for further details):

- At each time step, n, we extract the transverse electric field *one cell away* from the port boundary. Let us denote this by $\boldsymbol{E}_t|^n_{p,q,N_z-1}$. Clearly, this field can be represented as a superposition of waveguide modes that propagate along both directions of the waveguide. Subsequently, we consider for simplicity a port that does not have an incident wave.
- With this result we can compute the voltages $V_m|^n_{N_z-1}$ of the different modes m one cell away from the boundary:

$$V_m|^n_{N_z-1} = \sum_{p,q} \Delta x\, \Delta y\, \boldsymbol{E}_t|^n_{p,q,N_z-1} \cdot \boldsymbol{e}_m|_{p,q}. \tag{A.11}$$

  For a waveguide port without an incident field, the decomposed field only consists of waveguide modes that are propagating away from the interior of the computational domain. For a port with an incident field, we could easily compute the incident field at the plane one cell away from the port boundary. Then we subtract the incident field from the total field to get the field associated with modes that are propagating away from the computational domain.
- Each mode can be modeled by a 1D wave equation:

$$\frac{\partial^2 V_m}{\partial z^2} - \frac{1}{c_0^2}\frac{\partial^2 V_m}{\partial t^2} - h_m^2 V_m = 0, \tag{A.12}$$

  which can be discretized as

$$V_m|^{n+1}_r = A V_m|^n_r + B(V_m|^n_{r+1} + V_m|^n_{r-1}) - V_m|^{n-1}_r, \tag{A.13}$$

  where $V_m|^n_r \equiv V_m(r\Delta z, n\Delta t)$ and

$$B = \left(\frac{c_0 \Delta t}{\Delta z}\right)^2,$$
$$A = 2 - 2B - (c_0 \Delta t\, h_m)^2.$$

  The impulse response $I_m|^n = V_m|^n_1$ for this 1D wave equation can be computed from Eq. (A.13). Given this impulse response, we can use a 1D convolution to compute the voltages *on* the boundary that coincides with the port:

$$V_m|^n_{N_z} = V_m|^n_{N_z-1} * I_m|^n = \sum_{j=1}^{n}, V_m|^{n-j}_{N_z-1} \cdot I_m|^j. \tag{A.14}$$

- Now, we know the modal voltages on the port boundary. The total electric field on the boundary is a linear combination of the modal fields

$$\boldsymbol{E}|^{n}_{p,q,N_z} = \sum_{m=1}^{M} V_m|^{n}_{N_z}\, \boldsymbol{e}_m|_{p,q}, \tag{A.15}$$

and this solution is explicitly written into the FDTD grid before the next update of the interior grid points that are located inside the computational domain.

Implementation: A Point of Departure

The following supporting MATLAB script and functions are set up such that they can be used directly as a basis for the solution of the assignments that follow. Here is a brief description of each MATLAB file:

- `Main.m`: Setup of the problem with allocation of memory for variables that store the electromagnetic fields, port information, excitation pulse, etc. Given the setup of the computational problem, the routine contains a loop that should include the update expressions of the FDTD scheme in the bulk of the computational domain. Further, extraction of the scattering parameters and addition of possible metal objects are included in this routine.
- `ComputeTEModes.m` : Computes the transverse electric field for the TE modes and the corresponding cutoff frequencies associated with the FDTD discretization.
- `ComputeTMModes.m` : Computes the transverse electric field for the TM modes and the corresponding cutoff frequencies associated with the FDTD discretization.
- `ComputeIR.m`: Computes the impulse response for all the modes included in the analysis.

If you prefer to use another programming language, these MATLAB files could also be used as a point of departure. In that case, you will have to implement this functionality in the language of your choice. It deserves to be emphasized that if you prefer to work with MATLAB, it would be useful to read and study programs written by other programmers since that would give you the opportunity to develop your programming style and find new solutions to programming problems that you may encounter. In what follows, it is assumed that the functionality provided by `Main.m`, `ComputeTEModes.m`, `ComputeTMModes.m`, and `ComputeIR.m` is in place.

A.3.2 Assignments

Consider an empty waveguide with the dimensions:

$$L_x = 40.0 \text{ mm},\ L_y = 22.5 \text{ mm}, \text{and } L_z = 160.0 \text{ mm}. \tag{A.16}$$

The waveguide ports are located at $z = 0$ mm and $z = 160.0$ mm. The waveguide is excited by a TE_{10} mode at $z = 0$ by a Gaussian-modulated sinusoidal pulse, which contains energy in the frequency interval from 4 to 7 GHz. The port located at $z = 0$ is transparent to the reflected field, which essentially corresponds to the rectangular waveguide's continuing indefinitely for the region $z < 0$. Similarly, the port located at $z = L_z$ is transparent to the transmitted field, which continues to propagation in the positive z-direction as if the waveguide continued indefinitely for the region $z > L_z$. The MATLAB implementation (which may be used as a point of departure) is set up for these particular dimensions and this excitation.

Implementation

Implement the update loops for Faraday's and Ampère's law according to the FDTD scheme in three dimensions for an empty rectangular waveguide. It is sufficient to use a cell size of $h = 2.5$ mm for the following tests, which makes the computational domain consist of $16 \times 9 \times 64$ cells. However, it is often useful to make an implementation such that it is possible to change the cell size to allow for convergence studies.

Numerical Test #1

What is the expected reflected $s_{1,\mathrm{out}}(t)$ and transmitted $s_{2,\mathrm{out}}(t)$ solution for an empty waveguide given the Gaussian excitation pulse? Test your code and see if the result is what you expected. What is the cutoff frequency of the TE_{10} mode? Which mode has the second lowest cutoff frequency, and what frequency is that?

Implementation

Implement a postprocessing step that transforms the time-domain scattering amplitudes $s_{1,\mathrm{out}}(t)$ and $s_{2,\mathrm{out}}(t)$ to their corresponding frequency-domain quantities, and provide code that evaluates the scattering parameters (A.9) and (A.10).

Numerical Test #2

Verify that your implementation of the postprocessing step works as expected for the empty waveguide. You can calculate the analytical scattering parameters (A.9) and (A.10), which makes a careful comparison feasible. Comment on your findings by an interpretation of the numerical errors.

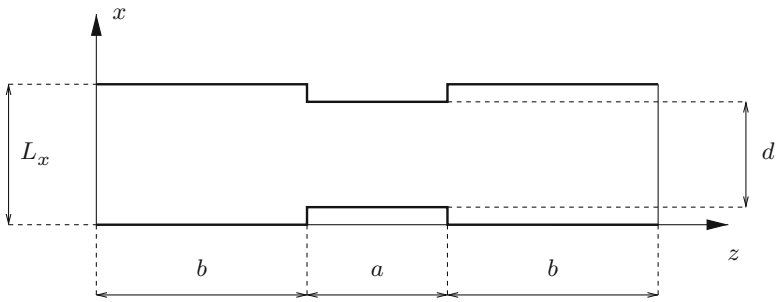

Fig. A.4 Waveguide with a narrow midsection

Implementation

Change the program so that you can analyze a waveguide that has a somewhat more narrow midsection, as shown in Fig. A.4. The dimensions are $a = 4\,\text{cm}$, $b = 6\,\text{cm}$, and $d = 3\,\text{cm}$. The geometry is independent of the y-coordinate and symmetric with respect to the plane $x = L_x/2$. (The extra walls of the waveguide are also perfectly conducting.) It may be noted that this problem can also be solved by a 2D code, should such a code be available, for comparative purposes.

Numerical Test #3

Compute $|S_{11}(\omega)|$ and $|S_{12}(\omega)|$ for this modified geometry. Comment on your findings and provide an explanation of the transmission (and reflection) as a function of frequency.

Implementation

A bit more challenging problem is a metallic block placed "on the floor" of the waveguide, as shown in Fig. A.5, which contains a top and side view of the metallic block indicated by the shaded region. The dimensions in Fig. A.5 are $a = 2\,\text{cm}$, $b = 7\,\text{cm}$, $d = 0.5\,\text{cm}$, and $h = 1.75\,\text{cm}$. The geometry is again symmetric with respect to the plane $x = L_x/2$, but this geometry yields a full 3D problem.

Numerical Test #4

Compute $|S_{11}(\omega)|$ and $|S_{12}(\omega)|$ for this modified geometry and comment on your findings. In fact, the added metal block yields a filtering structure with pass band characteristics. Can you explain the physics that causes this resonance? The field computation may give some information about the resonance.

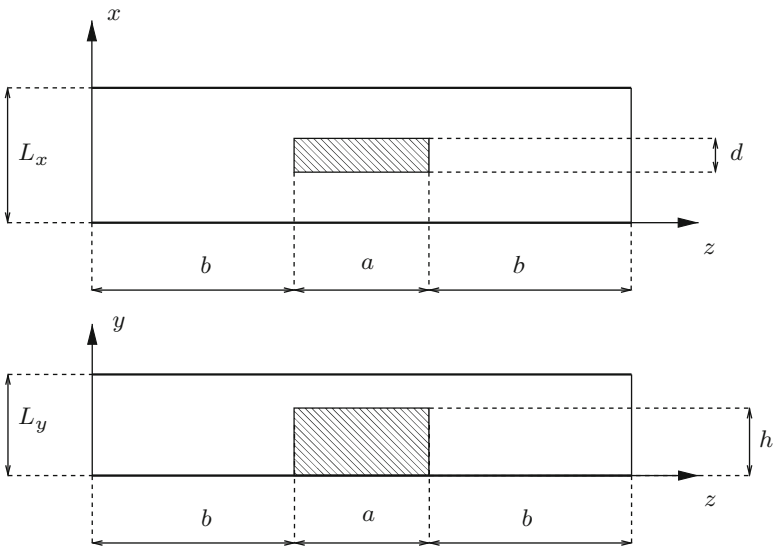

Fig. A.5 Waveguide with metal block placed "on the floor"

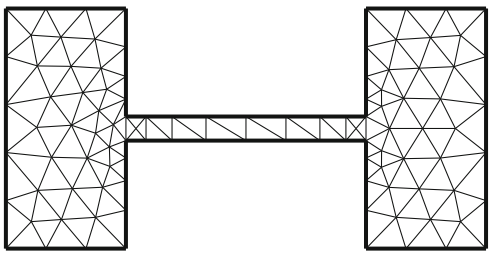

Fig. A.6 Cross section of a ridge waveguide discretized by *triangular finite elements*

A.4 Finite Element Method

A.4.1 Problem Description

A ridge waveguide has a cross section designed to allow for a single mode of propagation over larger bandwidths than a rectangular waveguide. A typical cross section of a ridge waveguide is shown in Fig. A.6. In what follows, the cross section of the waveguide is denoted by S and its closed boundary by L.

To compute the cutoff frequencies, we solve the eigenvalue problem

$$-\nabla^2 H_z = k_t^2 H_z \qquad \text{in } S, \tag{A.17}$$

$$\hat{\boldsymbol{n}} \cdot \nabla H_z = 0 \qquad \text{on } L \tag{A.18}$$

for the TE modes. For the TM modes we solve

$$-\nabla^2 E_z = k_t^2 E_z \qquad \text{in } S, \tag{A.19}$$

$$E_z = 0 \qquad \text{on } L, \tag{A.20}$$

where S is the interior of the waveguide's cross section and L its boundary. The transverse wave number is denoted by k_t and the longitudinal wave number by k_z, i.e., $k^2 = (\omega/c_0)^2 = k_t^2 + k_z^2$. More information on the theory of waveguides can be found in the literature, cf. [19].

Implementation: A Point of Departure

The following supporting MATLAB script and functions are set up such that they can be used directly as a basis for the solution of the assignments that follow. Here is a brief description of each MATLAB file:

- `Main.m` : Reads the grid, assembles the matrices, and solves the eigenvalue problem.
- `CmpElMtx.m` : Empty function where you can implement the computation of the element matrices.
- `ReadGrid.m` : Empty function where you can implement the reading of the meshes.
- `VisualizeMode.m` : Empty function where you can implement the visualization of the eigenmodes. This function should visualize two fields: (1) the field component parallel to the axis of the waveguide by means of colors and (2) its curl by vectors.

If you prefer to use another programming language, these MATLAB files could also be used as a point of departure. In that case, you will have to implement this functionality in the language of your choice. It deserves to be emphasized that if you prefer to work with MATLAB, it would be useful to read and study programs written by other programmers as that would give you the opportunity to develop your programming style and find new solutions to programming problems that you may encounter. In what follows, it is assumed that the functionality provided by `Main.m`, `CmpElMtx.m`, `ReadGrid.m`, and `Visualize.m` is in place.

Meshes

The MATLAB files come with meshes, which are stored in plain text files. The meshes discretize both a rectangular waveguide and a ridge waveguide. For each geometry, there are three discretizations that can be used for convergence studies.

- Meshes for rectangular waveguides of width $L_x = 2$ cm and height $L_y = 1$ cm.

  - `grid_rectangular_res1.txt`—coarse mesh
  - `grid_rectangular_res2.txt`—once hierarchically refined mesh
  - `grid_rectangular_res3.txt`—twice hierarchically refined mesh
- Mesh for a ridge waveguide of outer dimensions 2 and 1 cm. The spacing between the teeth is 0.1 cm, and their width is 1 cm. The mesh is shown in Fig. A.6.
  - `grid_ridge_res1.txt` – coarse mesh
  - `grid_ridge_res2.txt` – once hierarchically refined mesh
  - `grid_ridge_res3.txt` – twice hierarchically refined mesh

If you prefer to create your own meshes, there is a number of choices: (1) use a commercial tool, (2) use a freely available tool such as Triangle [73, 74], or (3) write your own mesh generator.

A.4.2 Assignments

Weak Form

Use the two eigenvalue problems as a point of departure to show that the corresponding weak forms can be written as

$$\int_S \nabla w_i \cdot \nabla H_z \, dS = k_t^2 \int_S w_i H_z \, dS,$$
$$\int_S \nabla w_i \cdot \nabla E_z \, dS = k_t^2 \int_S w_i E_z \, dS.$$

Here, the z-component of the electric and magnetic fields are expanded in and tested by nodal basis functions ϕ_i. The testing must be done in accordance with the boundary conditions. Observe that for the TE modes we have a Neumann boundary condition, while for the TM modes we have a Dirichlet boundary condition. What are the implications of this? How do the two problems differ from each other?

Element Matrices

The two eigenvalue problems can be expressed in the form $\mathbf{Ax} = k_t^2 \mathbf{Bx}$. Here, $\mathbf{A}$ and $\mathbf{B}$ are square matrices having different dimensions for the two different eigenvalue problems. To assemble the matrices $\mathbf{A}$ and $\mathbf{B}$, we sum contributions from each triangle, i.e., we need to compute the following integrals:

$$A_{ij}^e = \int_{S^e} \nabla \phi_i \cdot \nabla \phi_j \, dS,$$

$$B_{ij}^e = \int_{S^e} \phi_i \phi_j \, dS.$$

To evaluate these integrals, you may find the following formula useful:

$$\int_{S^e} (\phi_1^e)^\alpha (\phi_2^e)^\beta (\phi_3^e)^\gamma \, dS = 2S^e \frac{\alpha!\beta!\gamma!}{(\alpha+\beta+\gamma+2)!}, \tag{A.21}$$

where S^e is the area of element e. Here, the constants α, β, and γ are nonnegative integers.

Implementation

Implement the functionality of `ReadGrid.m`. This function takes the name of the file that stores the mesh as the argument. Then it reads the data in this file and returns the mesh in a format that is ready to use for the remaining program.

Numerical Test #1

Compute the 20 lowest k_t and their corresponding cutoff frequencies for the rectangular waveguide and compare to the analytical expression

$$k_t = \frac{\omega}{c_0} = \sqrt{\left(\frac{\pi n_x}{L_x}\right)^2 + \left(\frac{\pi n_y}{L_y}\right)^2}, \tag{A.22}$$

where $n_x = 0, 1, \ldots$ and $n_y = 0, 1, \ldots$, excluding $n_x = n_y = 0$.

- Perform a convergence test for the lowest eigenmode. Does this cutoff frequency converge to the analytical value? What is the order of convergence?
- Visualize the five lowest eigenmodes. Do the eigenmodes compare well with their analytical counterparts?

Numerical Test #2

Compute the 20 lowest k_t and their corresponding frequencies for the ridge waveguide. How do the two lowest cutoff frequencies change as compared to the rectangular waveguide? Compare the ratio f_2/f_1 between the two lowest frequencies ($f_2 > f_1$) for the ridge and rectangular waveguides. Can you explain the reason for this difference? How could you influence the ratio f_2/f_1 by changing the geometry?

Numerical Test #3

Perform a convergence test for the lowest eigenmode of the ridge waveguide. What is the extrapolated cutoff frequency? What is the order of convergence? Do you achieve an optimal order of convergence? If not, why is the order of convergence reduced?

Numerical Test #4

For the ridge waveguide, visualize both the longitudinal field component E_z together with $\nabla \times (\hat{\boldsymbol{z}} E_z)$ for the five lowest eigenmodes. Also, visualize both the longitudinal field component H_z together with $\nabla \times (\hat{\boldsymbol{z}} H_z)$ for the five lowest eigenmodes. Comment on how your visualizations relate to the cutoff frequencies. How do the eigenmodes compare with the rectangular waveguide modes?

A.5 Method of Moments

A.5.1 Problem Description

We seek the capacitance per unit length of a circular conducting wire placed over the middle of a conducting strip. The wire radius is a, and the center of the wire is located at a height h over the strip. The strip has width w and thickness b. The geometry is assumed to be two-dimensional and the structure is located in free space. For the computation of the capacitance per unit, it is therefore appropriate to use the method of moments.

For a capacitor where the two plates have different shape, the two plates must have opposite total charges. However, they will not in general have opposite potentials. Since Poisson's equation is linear, the charges and potentials must satisfy the linear relation

$$q_1 = C_{11} V_1 + C_{12} V_2,$$
$$q_2 = C_{21} V_1 + C_{22} V_2,$$

where q_i are the charges on the plates and V_i their potentials. The matrix elements C_{ij} are referred to as capacitance coefficients. Note that the capacitance coefficients in themselves are not physically very relevant and that they depend on how the normalizing distance is chosen in the expression for the potential from a line charge. The coefficients can be computed by considering two cases where V_1 and V_2 take linearly independent values, e.g.,

1. $V_1 = 1$ and $V_2 = 0$ yield the values for C_{11} and C_{21}.
2. $V_1 = 0$ and $V_2 = 1$ yield the values for C_{12} and C_{22}.

To define the capacitance in cylindrical geometry, we consider cases where the net charge is zero. This implies that $q_1 = -q_2 = q$. Then the capacitance is defined as

$$C = \frac{q}{V_1 - V_2}. \tag{A.23}$$

A.5.2 Assignments

First express the net capacitance C analytically in terms of the capacitance coefficients C_{11}, C_{12}, C_{21}, and C_{22}.

Implementation

Write a program to evaluate the capacitance of a general 2D, two-conductor capacitor and geometry part to generate the geometry described previously. For the 2D method of moments, you can use the MATLAB function `2DMoM.m` described in Sect. 7.2.2.

Numerical Test #1

Test the program by running the case $a = 1\,\text{mm}$, $b = 0.5\,\text{mm}$, $d = 2\,\text{mm}$, and $w = 20\,\text{mm}$. Determine the order of convergence and make an extrapolation to zero grid size within 1 % accuracy.

Numerical Test #2

Verify that your result for C is reasonably close to the analytical result for an infinite plate, i.e., $w = \infty$. (The analytical result can be derived by means of image theory for a circular cylinder next to an infinitely large ground plane, where details can be found in [19].) For this part, give the extrapolated coefficients C_{ij} as intermediate results.

Numerical Test #3

Keeping all the other parameters the same, set $w = 0.5\,\text{mm}$. Check the order of convergence and extrapolate to zero grid size, with at most 1 % error.

Numerical Test #4

Now you have some idea of what resolution is required. Based on this experience, compute and plot the capacitance as a function of width for the interval $0.2\,\text{mm} < w < 5\,\text{mm}$.

Appendix B
A Collection of the Lowest-Order Finite Elements

This book describes a number of different types of approximations that exploit representations in terms of finite elements. Useful information concerning these approximations is collected in this appendix for a range of typical element shapes: (1) the triangle, (2) the quadrilateral, (3) the tetrahedron, (4) the prism, and (5) the hexahedron. The following types of information are provided:

- Domain for the reference element.
- Expressions for the nodes (vertices) of the element.
- Definition of the edges of the element.
- Definition of the faces of the element should the element be a volume element.
- The lowest-order gradient-conforming basis functions that belong to the function space $H(\text{grad})$.
- The lowest-order curl-conforming basis functions that belong to the function space $H(\text{curl})$.
- The lowest-order divergence-conforming basis functions that belong to the function space $H(\text{div})$.
- The quadrature rule that can integrate quadratic polynomials on the reference element sufficiently well to yield an FEM with an error that is proportional to h^2.

Here, the function space $H(\text{grad})$ consists of all functions ϕ that satisfy

$$\int_{\Omega} |\nabla \phi|^2 + \phi^2 d\Omega < \infty, \tag{B.1}$$

where the computational domain Ω is one, two, or three dimensional. Similarly, the function space $H(\text{curl})$ consists of all functions $\boldsymbol{F}$ that satisfy

$$\int_{\Omega} |\nabla \times \boldsymbol{F}|^2 + |\boldsymbol{F}|^2 d\Omega < \infty, \tag{B.2}$$

T. Rylander et al., *Computational Electromagnetics*, Texts in Applied Mathematics 51, DOI 10.1007/978-1-4614-5351-2,

and the function space $H(\mathrm{div})$ consists of all functions $\boldsymbol{F}$ that satisfy

$$\int_{\Omega} |\nabla \cdot \boldsymbol{F}|^2 + |\boldsymbol{F}|^2 d\Omega < \infty. \tag{B.3}$$

Each gradient-conforming basis function φ_i (also referred to as nodal basis functions) is equal to unity at one node in the element and zero at all the other nodes. The node where the basis function evaluates to unity has the same index as the basis function itself. Consequently, there is exactly one nodal basis function for each node in the element. This can be expressed compactly as

$$\varphi_i(\boldsymbol{r}_j) = \delta_{ij}, \tag{B.4}$$

where δ_{ij} is the Kronecker delta that gives $\delta_{ij} = 1$ for $i = j$ and $\delta_{ij} = 0$ for $i \neq j$.

Each curl-conforming basis function $\boldsymbol{N}_i$ (also referred to as edge basis functions or edge elements) has a nonzero tangential component along one edge of the element and a zero tangential component along all the other edges of the element. Consequently, there is exactly one edge basis function associated with each edge in the element. This property of the curl-conforming basis functions listed in this appendix can be expressed compactly as

$$\hat{t}_j \cdot \boldsymbol{N}_i(\boldsymbol{r}_j) = \frac{1}{L_i}\delta_{ij} \quad \forall\, \boldsymbol{r}_j \in \text{edge } j, \tag{B.5}$$

where $\hat{\boldsymbol{t}}_j$ is a unit vector tangential to the edge such that it points from the start node of the edge to the end node of edge i. Further, L_i is the length of edge i. Thus, it is easy to construct a basis function with a unit tangential component along its associated edge by the scaling $L_i \boldsymbol{N}_i$.

Each divergence-conforming basis function $\boldsymbol{M}_i$ (also referred to as a face basis function for volume elements) has a nonzero normal component on each edge (for surface elements) or face (for volume elements) and has a zero normal component on all the other edges/faces of the element. Consequently, there is exactly one divergence-conforming basis function for each edge/face of an element. This property of the divergence-conforming basis functions listed in this appendix can be expressed compactly as

$$\hat{\boldsymbol{n}}_j \cdot \boldsymbol{M}_i(\boldsymbol{r}_j) = \frac{1}{L_i}\delta_{ij} \quad \forall\, \boldsymbol{r}_j \in \text{edge } j \tag{B.6}$$

for divergence-conforming basis functions on surface elements, i.e., the triangle and the quadrilateral. Here, $\hat{\boldsymbol{n}}_j$ is the outward unit normal to edge j. Similarly, this property of the divergence-conforming basis functions listed in this appendix can be expressed compactly as

$$\hat{\boldsymbol{n}}_j \cdot \boldsymbol{M}_i(\boldsymbol{r}_j) = \frac{1}{A_i}\delta_{ij} \quad \forall\, \boldsymbol{r}_j \in \text{face } j \tag{B.7}$$

for divergence-conforming basis functions on the volume elements tetrahedron, prism, and hexahedron. Here, $\hat{\boldsymbol{n}}_j$ is the outward unit normal to face j. Furthermore, A_i is the area of face i. Thus, it is easy to construct a basis function with unit normal component by the scaling $L_i \boldsymbol{M}_i$ for the surface elements and $A_i \boldsymbol{M}_i$ for the volume elements.

The collection of MATLAB programs associated with this book also contains a program that exploits Symbolic Math Toolbox in MATLAB to calculate the basis functions listed in the following discussion, where this program can also visualize the basis functions on the reference elements.

B.1 2D Elements

B.1.1 Triangle

The linear reference triangle occupies the surface bounded by $0 \leq u \leq 1 - v$ and $0 \leq v \leq 1$. It has three vertices and three edges.

The nodes for the reference element are given by

$$\begin{aligned} \boldsymbol{r}_1 &= [0, 0, 0], \\ \boldsymbol{r}_2 &= [+1, 0, 0], \\ \boldsymbol{r}_3 &= [0, +1, 0]. \end{aligned}$$

The edges for the reference element are listed in Table B.1.

Linear Basis Functions for $H(\mathbf{grad})$

The node basis functions for the reference element are given by

$$\begin{aligned} \varphi_1 &= 1 - u - v, \\ \varphi_2 &= u, \\ \varphi_3 &= v. \end{aligned}$$

Table B.1 Definition of edges for triangular reference element: node 1—start node of edge; and node 2—end node of edge

Edge	Node #1	Node #2
1	1	2
2	2	3
3	3	1

Table B.2 Quadrature points for reference triangle

Point	u-coordinate	v-coordinate
1	$6.666666666666667 \times 10^{-1}$	$1.666666666666667 \times 10^{-1}$
2	$1.666666666666667 \times 10^{-1}$	$6.666666666666667 \times 10^{-1}$
3	$1.666666666666667 \times 10^{-1}$	$1.666666666666667 \times 10^{-1}$

Table B.3 Quadrature weights for reference triangle

Point	Weight
1	$1.666666666666667 \times 10^{-1}$
2	$1.666666666666667 \times 10^{-1}$
3	$1.666666666666667 \times 10^{-1}$

Linear Basis Functions for $H(\mathbf{curl})$

The edge basis functions for the reference element are given by

$$\begin{aligned}
\boldsymbol{N}_1 &= \varphi_1 \tilde{\nabla} \varphi_2 - \varphi_2 \tilde{\nabla} \varphi_1 = (1-v)\hat{\boldsymbol{u}} + u\hat{\boldsymbol{v}}, \\
\boldsymbol{N}_2 &= \varphi_2 \tilde{\nabla} \varphi_3 - \varphi_3 \tilde{\nabla} \varphi_2 = -v\hat{\boldsymbol{u}} + u\hat{\boldsymbol{v}}, \\
\boldsymbol{N}_3 &= \varphi_3 \tilde{\nabla} \varphi_1 - \varphi_1 \tilde{\nabla} \varphi_3 = -v\hat{\boldsymbol{u}} + (u-1)\hat{\boldsymbol{v}}.
\end{aligned}$$

Linear Basis Functions for $H(\mathbf{div})$

The face basis functions for the reference element are given by

$$\begin{aligned}
\boldsymbol{M}_1 &= \boldsymbol{N}_1 \times \hat{\boldsymbol{w}} = u\hat{\boldsymbol{u}} + (v-1)\hat{\boldsymbol{v}}, \\
\boldsymbol{M}_2 &= \boldsymbol{N}_2 \times \hat{\boldsymbol{w}} = u\hat{\boldsymbol{u}} + v\hat{\boldsymbol{v}}, \\
\boldsymbol{M}_3 &= \boldsymbol{N}_3 \times \hat{\boldsymbol{w}} = (u-1)\hat{\boldsymbol{u}} + v\hat{\boldsymbol{v}},
\end{aligned}$$

where $\hat{\boldsymbol{w}} = \hat{\boldsymbol{u}} \times \hat{\boldsymbol{v}}$.

Quadrature Rule

The quadrature points in Table B.2 and the corresponding weights in Table B.3 yield a quadrature rule that integrates quadratic polynomials exactly on the reference triangle.

Table B.4 Definition of edges for the quadrilateral reference element: Node #1—start node of the edge; and Node #2—end node of the edge

Edge	Node 1	Node 2
1	1	2
2	2	3
3	3	4
4	4	1

B.1.2 Quadrilateral

The linear reference quadrilateral occupies the surface bounded by $-1 \leq u \leq 1$ and $-1 \leq v \leq 1$. It has four vertices and four edges.

The nodes for the reference element are given by

$$\begin{aligned} \boldsymbol{r}_1 &= [-1, -1, 0], \\ \boldsymbol{r}_2 &= [+1, -1, 0], \\ \boldsymbol{r}_3 &= [+1, +1, 0], \\ \boldsymbol{r}_4 &= [-1, +1, 0]. \end{aligned}$$

The edges for the reference element are given in Table B.4.

Linear Basis Functions for $H(\mathbf{grad})$

The node basis functions for the reference element are given by

$$\begin{aligned} \varphi_1 &= \psi^-(u)\,\psi^-(v) = \frac{1}{4}(1-u)(1-v), \\ \varphi_2 &= \psi^+(u)\,\psi^-(v) = \frac{1}{4}(1+u)(1-v), \\ \varphi_3 &= \psi^+(u)\,\psi^+(v) = \frac{1}{4}(1+u)(1+v), \\ \varphi_4 &= \psi^-(u)\,\psi^+(v) = \frac{1}{4}(1-u)(1+v). \end{aligned}$$

Here, we use the basis functions

$$\begin{aligned} \psi^-(\xi) &= \frac{1}{2}(1-\xi), \\ \psi^+(\xi) &= \frac{1}{2}(1+\xi). \end{aligned}$$

Linear Basis Functions for $H(\mathrm{curl})$

The edge basis functions for the reference element are given by

$$\boldsymbol{N}_1 = \psi^-(v)\tilde{\nabla}\psi^+(u) = \frac{1}{4}(1-v)\hat{\boldsymbol{u}},$$
$$\boldsymbol{N}_2 = \psi^+(u)\tilde{\nabla}\psi^+(v) = \frac{1}{4}(1+u)\hat{\boldsymbol{v}},$$
$$\boldsymbol{N}_3 = \psi^+(v)\tilde{\nabla}\psi^-(u) = -\frac{1}{4}(1+v)\hat{\boldsymbol{u}},$$
$$\boldsymbol{N}_4 = \psi^-(u)\tilde{\nabla}\psi^-(v) = -\frac{1}{4}(1-u)\hat{\boldsymbol{v}}.$$

Linear Basis Functions for $H(\mathrm{div})$

The face basis functions for the reference element are given by

$$\boldsymbol{M}_1 = \boldsymbol{N}_1 \times \hat{\boldsymbol{w}} = -\frac{1}{4}(1-v)\hat{\boldsymbol{v}},$$
$$\boldsymbol{M}_2 = \boldsymbol{N}_2 \times \hat{\boldsymbol{w}} = \frac{1}{4}(1+u)\hat{\boldsymbol{u}},$$
$$\boldsymbol{M}_3 = \boldsymbol{N}_3 \times \hat{\boldsymbol{w}} = \frac{1}{4}(1+v)\hat{\boldsymbol{v}},$$
$$\boldsymbol{M}_4 = \boldsymbol{N}_4 \times \hat{\boldsymbol{w}} = -\frac{1}{4}(1-u)\hat{\boldsymbol{u}}.$$

Quadrature Rule

The quadrature points in Table B.5 and the corresponding weights in Table B.6 yield trapezoidal integration for the reference quadrilateral, by means of the product of two 1D trapezoidal integration rules. This quadrature rule provides mass lumping for rectangular elements and, despite the fact that it cannot integrate quadratic

Table B.5 Quadrature points for reference quadrilateral

Point	u-coordinate	v-coordinate
1	−1.000000000000000	−1.000000000000000
2	1.000000000000000	−1.000000000000000
3	1.000000000000000	1.000000000000000
4	−1.000000000000000	1.000000000000000

Table B.6 Quadrature weights for reference quadrilateral

Point	Weight
1	1.000000000000000
2	1.000000000000000
3	1.000000000000000
4	1.000000000000000

polynomials exactly, yields a, FEM with second order of convergence for a piecewise linear approximation of the field.

B.2 3D Elements

B.2.1 Tetrahedron

The linear reference tetrahedron occupies the volume bounded by $0 \leq u \leq 1-v-w$, $0 \leq v \leq 1-w$, and $0 \leq w \leq 1$. It has four vertices, six edges, and four triangular faces.

The nodes for the reference element are given by

$$\begin{aligned} \boldsymbol{r}_1 &= [0,0,0], \\ \boldsymbol{r}_2 &= [+1,0,0], \\ \boldsymbol{r}_3 &= [0,+1,0], \\ \boldsymbol{r}_4 &= [0,0,+1]. \end{aligned}$$

The edges for the reference element are given in Table B.7.

The faces for the reference element are given in Table B.8.

Linear Basis Functions for $H(\mathbf{grad})$

The node basis functions for the reference element are given by

$$\begin{aligned} \varphi_1 &= 1-u-v-w, \\ \varphi_2 &= u, \\ \varphi_3 &= v, \\ \varphi_4 &= w. \end{aligned}$$

Table B.7 Definition of edges for tetrahedral reference element: node 1—start node of edge; node 2—end node of edge

Edge	Node 1	Node 2
1	1	2
2	2	3
3	3	1
4	1	4
5	2	4
6	3	4

Table B.8 Definition of faces for tetrahedral reference element: node 1—first node of face; node 2—second node of face; and node 3—third node of face

Face	Node 1	Node 2	Node 3
1	3	2	1
2	1	2	4
3	2	3	4
4	3	1	4

Linear Basis Functions for $H(\mathbf{curl})$

The edge basis functions for the reference element are given by

$$\begin{aligned}
\boldsymbol{N}_1 &= \varphi_1\tilde{\nabla}\varphi_2 - \varphi_2\tilde{\nabla}\varphi_1 = (1-w-v)\hat{\boldsymbol{u}} + u\hat{\boldsymbol{v}} + u\hat{\boldsymbol{w}},\\
\boldsymbol{N}_2 &= \varphi_2\tilde{\nabla}\varphi_3 - \varphi_3\tilde{\nabla}\varphi_2 = -v\hat{\boldsymbol{u}} + u\hat{\boldsymbol{v}},\\
\boldsymbol{N}_3 &= \varphi_3\tilde{\nabla}\varphi_1 - \varphi_1\tilde{\nabla}\varphi_3 = -v\hat{\boldsymbol{u}} + (u+w-1)\hat{\boldsymbol{v}} - v\hat{\boldsymbol{w}},\\
\boldsymbol{N}_4 &= \varphi_1\tilde{\nabla}\varphi_4 - \varphi_4\tilde{\nabla}\varphi_1 = w\hat{\boldsymbol{u}} + w\hat{\boldsymbol{v}} + (1-v-u)\hat{\boldsymbol{w}},\\
\boldsymbol{N}_5 &= \varphi_2\tilde{\nabla}\varphi_4 - \varphi_4\tilde{\nabla}\varphi_2 = -w\hat{\boldsymbol{u}} + u\hat{\boldsymbol{w}},\\
\boldsymbol{N}_6 &= \varphi_3\tilde{\nabla}\varphi_4 - \varphi_4\tilde{\nabla}\varphi_3 = -w\hat{\boldsymbol{v}} + v\hat{\boldsymbol{w}}.
\end{aligned}$$

Linear Basis Functions for $H(\mathbf{div})$

The face basis functions for the reference element are given by

$$\begin{aligned}
\boldsymbol{M}_1 &= 2(\varphi_3\tilde{\nabla}\varphi_2 \times \tilde{\nabla}\varphi_1 + \varphi_2\tilde{\nabla}\varphi_1 \times \tilde{\nabla}\varphi_3 + \varphi_1\tilde{\nabla}\varphi_3 \times \tilde{\nabla}\varphi_2)\\
&= 2[u\hat{\boldsymbol{u}} + v\hat{\boldsymbol{v}} + (w-1)\hat{\boldsymbol{w}}],\\
\boldsymbol{M}_2 &= 2(\varphi_1\tilde{\nabla}\varphi_2 \times \tilde{\nabla}\varphi_4 + \varphi_2\tilde{\nabla}\varphi_4 \times \tilde{\nabla}\varphi_1 + \varphi_4\tilde{\nabla}\varphi_1 \times \tilde{\nabla}\varphi_2)\\
&= 2[u\hat{\boldsymbol{u}} + (v-1)\hat{\boldsymbol{v}} + w\hat{\boldsymbol{w}}],\\
\boldsymbol{M}_3 &= 2(\varphi_2\tilde{\nabla}\varphi_3 \times \tilde{\nabla}\varphi_4 + \varphi_3\tilde{\nabla}\varphi_4 \times \tilde{\nabla}\varphi_2 + \varphi_4\tilde{\nabla}\varphi_2 \times \tilde{\nabla}\varphi_3)\\
&= 2[u\hat{\boldsymbol{u}} + v\hat{\boldsymbol{v}} + w\hat{\boldsymbol{w}}],
\end{aligned}$$

Table B.9 Quadrature points for reference tetrahedron

Point	u-coordinate	v-coordinate	w-coordinate
1	$5.854101966249685 \times 10^{-1}$	$1.381966011250105 \times 10^{-1}$	$1.381966011250105 \times 10^{-1}$
2	$1.381966011250105 \times 10^{-1}$	$5.854101966249685 \times 10^{-1}$	$1.381966011250105 \times 10^{-1}$
3	$1.381966011250105 \times 10^{-1}$	$1.381966011250105 \times 10^{-1}$	$5.854101966249685 \times 10^{-1}$
4	$1.381966011250105 \times 10^{-1}$	$1.381966011250105 \times 10^{-1}$	$1.381966011250105 \times 10^{-1}$

Table B.10 Quadrature weights for reference tetrahedron

Point	Weight
1	$4.166666666666666 \times 10^{-2}$
2	$4.166666666666666 \times 10^{-2}$
3	$4.166666666666666 \times 10^{-2}$
4	$4.166666666666666 \times 10^{-2}$

$$\begin{aligned} \boldsymbol{M}_4 &= 2(\varphi_3 \tilde{\nabla}\varphi_1 \times \tilde{\nabla}\varphi_4 + \varphi_1 \tilde{\nabla}\varphi_4 \times \tilde{\nabla}\varphi_3 + \varphi_4 \tilde{\nabla}\varphi_3 \times \tilde{\nabla}\varphi_1) \\ &= 2[(u-1)\hat{\boldsymbol{u}} + v\hat{\boldsymbol{v}} + w\hat{\boldsymbol{w}}]. \end{aligned}$$

Quadrature Rule

The quadrature points in Table B.9 and the corresponding weights in Table B.10 yield a quadrature rule that integrates quadratic polynomials exactly on the reference tetrahedron.

B.2.2 Prism

The linear reference prism occupies the volume bounded by $0 \leq u \leq 1-v$, $0 \leq v \leq 1$, and $-1 \leq w \leq 1$. It has six vertices, nine edges, two triangular faces, and three quadrilateral faces.

The nodes for the reference element are given by

$$\begin{aligned} \boldsymbol{r}_1 &= [0, 0, -1], \\ \boldsymbol{r}_2 &= [+1, 0, -1], \\ \boldsymbol{r}_3 &= [0, +1, -1], \\ \boldsymbol{r}_4 &= [0, 0, +1], \\ \boldsymbol{r}_5 &= [+1, 0, +1], \\ \boldsymbol{r}_6 &= [0, +1, +1]. \end{aligned}$$

The edges for the reference element are given in Table B.11.

The faces for the reference element are given in Table B.12.

Table B.11 Definition of edges for pyramidal reference element: node 1—start node of edge; node 2—end node of edge

Edge	Node 1	Node 2
1	1	2
2	2	3
3	3	1
4	1	4
5	2	5
6	3	6
7	4	5
8	5	6
9	6	4

Table B.12 Definition of faces for tetrahedral reference element: node 1—first node of face; node 2—second node of face; node 3—third node of face; node 4—fourth node of face (for three quadrilateral faces only)

Face	Node 1	Node 2	Node 3	Node 4
1	3	2	1	–
2	1	2	5	4
3	2	3	6	5
4	3	1	4	6
5	4	5	6	–

Linear Basis Functions for $H(\mathrm{grad})$

The node basis functions for the reference element are given by

$$\varphi_1 = \phi_1(u,v)\,\psi^-(w) = \frac{1}{2}(1-u-v)(1-w),$$

$$\varphi_2 = \phi_2(u,v)\,\psi^-(w) = \frac{1}{2}u(1-w),$$

$$\varphi_3 = \phi_3(u,v)\,\psi^-(w) = \frac{1}{2}v(1-w),$$

$$\varphi_4 = \phi_1(u,v)\,\psi^+(w) = \frac{1}{2}(1-u-v)(1+w),$$

$$\varphi_5 = \phi_2(u,v)\,\psi^+(w) = \frac{1}{2}u(1+w),$$

$$\varphi_6 = \phi_3(u,v)\,\psi^+(w) = \frac{1}{2}v(1+w).$$

We have the basis functions

$$\phi_1 = 1-u-v,$$

$$\phi_2 = u,$$

$$\phi_3 = v$$

that vary in the plane perpendicular to the cylinder axis of the prism and, thus, depend only on u and v. Further, the two basis functions

$$\psi^- = \frac{1}{2}(1-w),$$
$$\psi^+ = \frac{1}{2}(1+w)$$

vary along the cylinder axis with the coordinate w.

Linear Basis Functions for $H(\mathrm{curl})$

The edge basis functions for the reference element are given by

$$\boldsymbol{N}_1 = \boldsymbol{n}_1(u,v)\,\psi^-(w) = \frac{1}{2}\left[(v-1)(w-1)\hat{\boldsymbol{u}} - u(w-1)\hat{\boldsymbol{v}}\right],$$
$$\boldsymbol{N}_2 = \boldsymbol{n}_2(u,v)\,\psi^-(w) = \frac{1}{2}\left[v(w-1)\hat{\boldsymbol{u}} - u(w-1)\hat{\boldsymbol{v}}\right],$$
$$\boldsymbol{N}_3 = \boldsymbol{n}_3(u,v)\,\psi^-(w) = \frac{1}{2}\left[v(w-1)\hat{\boldsymbol{u}} - (u-1)(w-1)\hat{\boldsymbol{v}}\right],$$
$$\boldsymbol{N}_4 = \phi_1(u,v)\,\tilde{\nabla}\psi^+(w) = \frac{1}{2}(1-v-u)\hat{\boldsymbol{w}},$$
$$\boldsymbol{N}_5 = \phi_2(u,v)\,\tilde{\nabla}\psi^+(w) = \frac{1}{2}u\hat{\boldsymbol{w}},$$
$$\boldsymbol{N}_6 = \phi_3(u,v)\,\tilde{\nabla}\psi^+(w) = \frac{1}{2}v\hat{\boldsymbol{w}},$$
$$\boldsymbol{N}_7 = \boldsymbol{n}_1(u,v)\,\psi^+(w) = \frac{1}{2}\left[-(v-1)(w+1)\hat{\boldsymbol{u}} + u(w+1)\hat{\boldsymbol{v}}\right],$$
$$\boldsymbol{N}_8 = \boldsymbol{n}_2(u,v)\,\psi^+(w) = \frac{1}{2}\left[-v(w+1)\hat{\boldsymbol{u}} + u(w+1)\hat{\boldsymbol{v}}\right],$$
$$\boldsymbol{N}_9 = \boldsymbol{n}_3(u,v)\,\psi^+(w) = \frac{1}{2}\left[-v(w+1)\hat{\boldsymbol{u}} + (u-1)(w+1)\hat{\boldsymbol{v}}\right],$$

where

$$\boldsymbol{n}_1 = \phi_1\tilde{\nabla}\phi_2 - \phi_2\tilde{\nabla}\phi_1,$$
$$\boldsymbol{n}_2 = \phi_2\tilde{\nabla}\phi_3 - \phi_3\tilde{\nabla}\phi_2,$$
$$\boldsymbol{n}_3 = \phi_3\tilde{\nabla}\phi_1 - \phi_1\tilde{\nabla}\phi_3$$

are the edge elements on the triangular cross section of the prism perpendicular to the cylinder axis.

Linear Basis Functions for $H(\mathrm{div})$

The face basis functions for the reference element are given by

$$\begin{aligned}
\boldsymbol{M}_1 &= 4\psi^-(w)\tilde{\nabla}\psi^-(w) = (w-1)\hat{\boldsymbol{w}},\\
\boldsymbol{M}_2 &= \frac{1}{2}\boldsymbol{m}_1(u,v) = \frac{1}{2}\left[u\hat{\boldsymbol{u}} + (v-1)\hat{\boldsymbol{v}}\right],\\
\boldsymbol{M}_3 &= \frac{1}{2}\boldsymbol{m}_2(u,v) = \frac{1}{2}\left[u\hat{\boldsymbol{u}} + v\hat{\boldsymbol{v}}\right],\\
\boldsymbol{M}_4 &= \frac{1}{2}\boldsymbol{m}_3(u,v) = \frac{1}{2}\left[(u-1)\hat{\boldsymbol{u}} + v\hat{\boldsymbol{v}}\right],\\
\boldsymbol{M}_5 &= 4\psi^+(w)\tilde{\nabla}\psi^+(w) = (w+1)\hat{\boldsymbol{w}},
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{m}_1 &= \boldsymbol{n}_1 \times \hat{\boldsymbol{w}},\\
\boldsymbol{m}_2 &= \boldsymbol{n}_2 \times \hat{\boldsymbol{w}},\\
\boldsymbol{m}_3 &= \boldsymbol{n}_3 \times \hat{\boldsymbol{w}}.
\end{aligned}$$

Quadrature Rule

The quadrature points in Table B.13 and the corresponding weights in Table B.14 yield a quadrature rule that integrates quadratic polynomials exactly in the plane perpendicular to the axis of the prism. However, the quadrature rule exploits trapezoidal integration along the prism axis and, consequently, provides mass lumping along the cylinder axis for straight prisms. Despite the fact that this quadrature rule for a prism cannot integrate quadratic polynomials exactly, it yields an FEM with second order of convergence for a piecewise linear approximation of the field.

B.2.3 Hexahedron

The linear reference hexahedron occupies the volume bounded by $-1 \le u \le 1$, $-1 \le v \le 1$, and $-1 \le w \le 1$. It has 8 vertices, 12 edges, and 6 quadrilateral faces.

The nodes for the reference element are given by

Table B.13 Quadrature points for reference prism

Point	u-coordinate	v-coordinate	w-coordinate
1	$6.666666666666667 \times 10^{-1}$	$1.666666666666667 \times 10^{-1}$	−1.000000000000000
2	$1.666666666666667 \times 10^{-1}$	$6.666666666666667 \times 10^{-1}$	−1.000000000000000
3	$1.666666666666667 \times 10^{-1}$	$1.666666666666667 \times 10^{-1}$	−1.000000000000000
4	$6.666666666666667 \times 10^{-1}$	$1.666666666666667 \times 10^{-1}$	1.000000000000000
5	$1.666666666666667 \times 10^{-1}$	$6.666666666666667 \times 10^{-1}$	1.000000000000000
6	$1.666666666666667 \times 10^{-1}$	$1.666666666666667 \times 10^{-1}$	1.000000000000000

Table B.14 Quadrature weights for reference prism

Point	Weight
1	$1.666666666666667 \times 10^{-1}$
2	$1.666666666666667 \times 10^{-1}$
3	$1.666666666666667 \times 10^{-1}$
4	$1.666666666666667 \times 10^{-1}$
5	$1.666666666666667 \times 10^{-1}$
6	$1.666666666666667 \times 10^{-1}$

Table B.15 Definition of edges for hexahedral reference element: node 1—start node of edge; node 2—end node of edge

Edge	Node 1	Node 2
1	1	2
2	2	3
3	3	4
4	4	1
5	1	5
6	2	6
7	3	7
8	4	8
9	5	6
10	6	7
11	7	8
12	8	5

$$
\begin{aligned}
\boldsymbol{r}_1 &= [-1, -1, -1], \\
\boldsymbol{r}_2 &= [+1, -1, -1], \\
\boldsymbol{r}_3 &= [+1, +1, -1], \\
\boldsymbol{r}_4 &= [-1, +1, -1], \\
\boldsymbol{r}_5 &= [-1, -1, +1], \\
\boldsymbol{r}_6 &= [+1, -1, +1], \\
\boldsymbol{r}_7 &= [+1, +1, +1], \\
\boldsymbol{r}_8 &= [-1, +1, +1].
\end{aligned}
$$

The edges for the reference element are given in Table B.15.
The faces for the reference element are given in Table B.16.

Table B.16 Definition of faces for tetrahedral reference element: node 1—first node of face; node 2—second node of face; node 3—third node of face; node 4—fourth node of face

Face	Node 1	Node 2	Node 3	Node 4
1	4	3	2	1
2	1	2	6	5
3	2	3	7	6
4	3	4	8	7
5	4	1	5	8
6	5	6	7	8

Linear Basis Functions for $H(\mathbf{grad})$

The node basis functions for the reference element are given by

$$\varphi_1 = \psi^-(u)\,\psi^-(v)\,\psi^-(w) = \frac{1}{8}(1-u)(1-v)(1-w),$$

$$\varphi_2 = \psi^+(u)\,\psi^-(v)\,\psi^-(w) = \frac{1}{8}(1+u)(1-v)(1-w),$$

$$\varphi_3 = \psi^+(u)\,\psi^+(v)\,\psi^-(w) = \frac{1}{8}(1+u)(1+v)(1-w),$$

$$\varphi_4 = \psi^-(u)\,\psi^+(v)\,\psi^-(w) = \frac{1}{8}(1-u)(1+v)(1-w),$$

$$\varphi_5 = \psi^-(u)\,\psi^-(v)\,\psi^+(w) = \frac{1}{8}(1-u)(1-v)(1+w),$$

$$\varphi_6 = \psi^+(u)\,\psi^-(v)\,\psi^+(w) = \frac{1}{8}(1+u)(1-v)(1+w),$$

$$\varphi_7 = \psi^+(u)\,\psi^+(v)\,\psi^+(w) = \frac{1}{8}(1+u)(1+v)(1+w),$$

$$\varphi_8 = \psi^-(u)\,\psi^+(v)\,\psi^+(w) = \frac{1}{8}(1-u)(1+v)(1+w).$$

Here, we use the basis functions

$$\psi^-(\xi) = \frac{1}{2}(1-\xi),$$

$$\psi^+(\xi) = \frac{1}{2}(1+\xi).$$

Linear Basis Functions for $H(\mathbf{curl})$

The edge basis functions for the reference element are given by

$$\boldsymbol{N}_1 = \psi^-(v)\,\psi^-(w)\tilde{\nabla}\psi^+(u) = \frac{1}{8}(1-v)(1-w)\hat{\boldsymbol{u}},$$

$$\boldsymbol{N}_2 = \psi^+(u)\,\psi^-(w)\tilde{\nabla}\psi^+(v) = \frac{1}{8}(1+u)(1-w)\hat{\boldsymbol{v}},$$

$$\boldsymbol{N}_3 = \psi^+(v)\,\psi^-(w)\tilde{\nabla}\psi^-(u) = -\frac{1}{8}(1+v)(1-w)\hat{\boldsymbol{u}},$$

$$\boldsymbol{N}_4 = \psi^-(u)\,\psi^-(w)\tilde{\nabla}\psi^-(v) = -\frac{1}{8}(1-u)(1-w)\hat{\boldsymbol{v}},$$

$$\boldsymbol{N}_5 = \psi^-(u)\,\psi^-(v)\tilde{\nabla}\psi^+(w) = \frac{1}{8}(1-u)(1-v)\hat{\boldsymbol{w}},$$

$$\boldsymbol{N}_6 = \psi^+(u)\,\psi^-(v)\tilde{\nabla}\psi^+(w) = \frac{1}{8}(1+u)(1-v)\hat{\boldsymbol{w}},$$

$$\boldsymbol{N}_7 = \psi^+(u)\,\psi^+(v)\tilde{\nabla}\psi^+(w) = \frac{1}{8}(1+u)(1+v)\hat{\boldsymbol{w}},$$

$$\boldsymbol{N}_8 = \psi^-(u)\,\psi^+(v)\tilde{\nabla}\psi^+(w) = \frac{1}{8}(1-u)(1+v)\hat{\boldsymbol{w}},$$

$$\boldsymbol{N}_9 = \psi^-(v)\,\psi^+(w)\tilde{\nabla}\psi^+(u) = \frac{1}{8}(1-v)(1+w)\hat{\boldsymbol{u}},$$

$$\boldsymbol{N}_{10} = \psi^+(u)\,\psi^+(w)\tilde{\nabla}\psi^+(v) = \frac{1}{8}(1+u)(1+w)\hat{\boldsymbol{v}},$$

$$\boldsymbol{N}_{11} = \psi^+(v)\,\psi^+(w)\tilde{\nabla}\psi^-(u) = -\frac{1}{8}(1+v)(1+w)\hat{\boldsymbol{u}},$$

$$\boldsymbol{N}_{12} = \psi^-(u)\,\psi^+(w)\tilde{\nabla}\psi^-(v) = -\frac{1}{8}(1-u)(1+w)\hat{\boldsymbol{v}}.$$

Linear Basis Functions for $H(\mathbf{div})$

The face basis functions for the reference element are given by

$$\boldsymbol{M}_1 = \frac{1}{2}\psi^-(w)\tilde{\nabla}\psi^-(w) = -\frac{1}{8}(1-w)\hat{\boldsymbol{w}},$$

$$\boldsymbol{M}_2 = \frac{1}{2}\psi^-(v)\tilde{\nabla}\psi^-(v) = -\frac{1}{8}(1-v)\hat{\boldsymbol{v}},$$

$$\boldsymbol{M}_3 = \frac{1}{2}\psi^+(u)\tilde{\nabla}\psi^+(u) = +\frac{1}{8}(1+u)\hat{\boldsymbol{u}},$$

$$\boldsymbol{M}_4 = \frac{1}{2}\psi^+(v)\tilde{\nabla}\psi^+(v) = +\frac{1}{8}(1+v)\hat{\boldsymbol{v}},$$

Table B.17 Quadrature points for reference hexahedron

Point	u-coordinate	v-coordinate	w-coordinate
1	−1.000000000000000	−1.000000000000000	−1.000000000000000
2	1.000000000000000	−1.000000000000000	−1.000000000000000
3	1.000000000000000	1.000000000000000	−1.000000000000000
4	−1.000000000000000	1.000000000000000	−1.000000000000000
5	−1.000000000000000	−1.000000000000000	1.000000000000000
6	1.000000000000000	−1.000000000000000	1.000000000000000
7	1.000000000000000	1.000000000000000	1.000000000000000
8	−1.000000000000000	1.000000000000000	1.000000000000000

Table B.18 Quadrature weights for reference hexahedron

Point	Weight
1	1.000000000000000
2	1.000000000000000
3	1.000000000000000
4	1.000000000000000
5	1.000000000000000
6	1.000000000000000
7	1.000000000000000
8	1.000000000000000

$$\boldsymbol{M}_5 = \frac{1}{2}\psi^-(u)\tilde{\nabla}\psi^-(u) = -\frac{1}{8}(1-u)\hat{\boldsymbol{u}},$$
$$\boldsymbol{M}_6 = \frac{1}{2}\psi^+(w)\tilde{\nabla}\psi^+(w) = +\frac{1}{8}(1-w)\hat{\boldsymbol{w}}.$$

Quadrature Rule

The quadrature points in Table B.17 and the corresponding weights in Table B.18 yield trapezoidal integration for the reference hexahedron by means of the product of three 1D trapezoidal integration rules. This quadrature rule provides mass lumping for brick-shaped elements and, despite the fact that it cannot integrate quadratic polynomials exactly, yields an FEM with second order of convergence for a piecewise linear approximation of the field.

Appendix C
Large Linear Systems

C.1 Sparse Matrices

Many CEM problems require the solution of large linear systems of equations. This is generally the case for the finite element method (FEM), both for frequency- and time-domain applications. In realistic 3D applications, the number of unknowns can be in the range of tens of thousands to several millions. For the largest systems, direct inversion is seldom possible, and iterative methods are needed. Here, we will introduce some routines for large linear systems.

Below, we give a MATLAB function that assembles the sparse system that we solved using Gauss–Seidel iterations in the capacitance calculation in Chap. 3. The study was then limited to a 50×50 grid. With the assembled system we can use more efficient methods and therefore use higher resolutions. For this 2D problem, the direct solver invoked by "\" in MATLAB performs very well.

We write the discretized problem as $\mathbf{Af} = \mathbf{s}$ and use the MATLAB function `setAs` listed below to set $\mathbf{A}$ and $\mathbf{s}$. Note that this script was written so as to make very few references to the sparse matrix. This is faster than referencing the individual elements in the sparse matrix, because each reference requires a function call, which is quite slow.

```
% ------------------------------------------------------------
% Set up matrix A and right-hand side s
% ------------------------------------------------------------
function [A, s] = setAs(a, b, c, d, n, m)

% Arguments:
%    a    =  width of inner conductor
%    b    = height of inner conductor
%    c    =  width of outer conductor
%    d    = height of outer conductor
%    n    = number of points in the x-direction (horizontal)
%    m    = number of points in the y-direction (vertical)
% Returns:
%    A    = matrix on sparse storage format
%    s    = right-hand side on sparse storage format
```

T. Rylander et al., *Computational Electromagnetics*, Texts in Applied Mathematics 51, DOI 10.1007/978-1-4614-5351-2,

```
hx = 0.5*c/n;           % Grid size in x-direction
na = round(0.5*a/hx);   % Number of cells for half width of
                        % inner conductor
hy = 0.5*d/m;           % Grid size in y-direction
m  = round(0.5*d/hy);   % Number of cells for half height of
                        % outer conductor
mb = round(0.5*b/hy);   % Number of cells for half height of
                        % inner conductor
p  = 1;                 % Potential on inner conductor

%          The upper right corner is discretized
%
%               ---------------------+
%                         c/2        |
%                                    |
%               ------------+        | d/2
%                   a/2     |        |
%                           | b/2    |
%                           |        |
%
%                     (Dimensions)
%
%
%              The nodes are numbered like this
%        y
%        ^
%        |
% (m-1)hy| (m-1)n+1 (m-1)n+2 (m-1)n+3  ...    mn
%        |     :        :        :            :
%   2hy  |   2n+1     2n+2     2n+3    ...    3n
%    hy  |    n+1      n+2      n+3    ...    2n
%     0  |     1        2        3     ...     n
%           ---------------------------------------> x
%              0       hx      2 hx        (n-1)hx
%
%                    (Discretization)

N  = n * m;     % Total number of unknowns.
cx = hx^-2;
cy = hy^-2;

% Generate a matrix with N = m*n rows (-> nodes on the grid),
% and five columns, one for each nonzero diagonal of A.
% The first column gives contribution from nodes beneath.
% The second column gives contribution from nodes to the left.
% The third column gives self-contribution.
% The fourth column gives contribution from nodes to the right.
% The fifth column gives contribution from nodes above.

% The following lines assume some knowledge of MATLAB.  If you
% feel uncertain, insert the 'keyboard' command.  This causes
% MATLAB to stop.  Then execute lines by 'dbstep' and examine
% the result.
```

```
C = repmat([cy cx -2*(cx+cy) cx cy], N, 1);

% Find indices of nodes that are not surrounded by four interior
% nodes.
idx0R = n:n:N-n;          % Nodes with     V = 0 to the right
idxNB = na+2:n;           % Nodes with dV/dy = 0 beneath
idxNL = 1+n*(mb+1):n:N;   % Nodes with dV/dx = 0 to the left

idx1C = repmat((1:na+1)', 1, mb+1) + repmat((0:n:n*mb),na+1,1);
                          % 'x-index + n*(y-index-1)' for all
idx1C = idx1C(:)';        % nodes on (or inside) the inner
                          % conductor where V = 1
                          % and convert to row vector

C(idx1C,[1 2 4 5]) = 0;
C(idx1C, 3) = 1;
C(idx0R, 4) = 0;
C(idxNB, 5) = 2*cy;
C(idxNL, 4) = 2*cx;
C(idxNL, 2) = 0;

% Find the nonzero elements (si) of each column and the
% corresponding row indices (ii).  Do not include elements
% corresponding to nodes outside the grid.
[i1,j,s1] = find(C(n+1:end,   1)); % The first 'nc' nodes have no
                                   % neighbors beneath
[i2,j,s2] = find(C(1+1:end,   2)); % The first node has no
                                   % neighbor to the left
[i3,j,s3] = find(C(  1:end,   3));
[i4,j,s4] = find(C(  1:end-1, 4)); % The last node has no
                                   % neighbor to the right
[i5,j,s5] = find(C(  1:end-n, 5)); % The last 'nc' nodes have no
                                   % neighbors above

% Put the elements (si) into a sparse matrix. The first input
% are row indices, the second is column indices and the third
% is the elements.
A = sparse([i1+n; i2+1; i3; i4; i5], ...
           [i1; i2; i3; i4+1; i5+n], ...
           [s1; s2; s3; s4; s5], N, N);
s = sparse(idx1C', 1, p, N, 1);
```

C.2 Solvers for Large Sparse Systems of Equations

As we already mentioned, the 2D discretized Laplace equation can be solved in MATLAB by direct inversion `f = A\s`. For 2D problems, direct methods are generally very competitive, unless the problems are very large. However, for 3D problems, iterative solvers are often more efficient. We will here give a brief overview of solvers for sparse linear systems of equations that are used in CEM.

C.2.1 Direct Solvers

In direct methods, a complete factorization (e.g., an LU decomposition) of the matrix $\mathbf{A}$ is done. Clever reordering of the rows and the columns of $\mathbf{A}$ plays an important role; a good reordering scheme can reduce the operation count and the memory requirements for the factorization by more than an order of magnitude. In MATLAB, one can, for example, use column approximate minimum degree permutation, `colamd` (for nonsymmetric matrices), or symmetric approximate minimum degree permutation, `symamd` (for symmetric matrices), to reorder matrices. However, when the backslash operator "\" is invoked, this is done automatically.

A major advantage of direct methods compared to iterative methods is that since a complete factorization is done, additional right-hand sides can be solved for with low additional cost. Another advantage is that direct methods generally are less sensitive to ill conditioning and can be used where many iterative methods fail to converge.

However, both time and memory requirements scale unfavorably with problem size; hence direct methods become prohibitively expensive for very large problems. Often the memory requirements are the limiting factor.

Efficient, freely available algorithms for direct factorization and reordering of sparse matrices include UMFPACK [23], SuperLU [24], TAUCS [84], and METIS [44].

C.2.2 Iterative Solvers

The matrices that result from finite element discretizations of Poisson's equation (1.3) or the time-domain version of the curl-curl equation (6.86) are symmetric and positive definite. For such systems, iterative so-called Krylov methods (see Appendix D) generally work very well.

However, to speed up the convergence of the iterative algorithm, it is very useful to precondition the matrix. The idea of preconditioning is to find an approximate inverse of $\mathbf{A}$, say $\mathbf{M}^{-1}$, and multiply $\mathbf{Af} = \mathbf{s}$ by the approximate inverse from the left. If $\mathbf{M}^{-1}\mathbf{A} \approx \mathbf{I}$, the iterative solver will converge much faster. The choice of preconditioner generally has a much stronger effect on the speed of convergence than the choice of Krylov method. A choice that often works well is the so-called incomplete LU decomposition, in which $\mathbf{M} = \mathbf{LU} \approx \mathbf{A}$, with $\mathbf{L}$ a lower triangular and $\mathbf{U}$ an upper triangular matrix. Then $\mathbf{M}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}$, which is inexpensive to apply if $\mathbf{L}$ and $\mathbf{U}$ are sparse. When $\mathbf{A}$ is symmetric, the factorization can be made such that $\mathbf{U} = \mathbf{L}^T$, and this is called incomplete Cholesky decomposition. The degree of incompleteness can be specified by how much fill-in is allowed in $\mathbf{L}$ and $\mathbf{U}$, that is, how many extra nonzero elements $\mathbf{L}$ and $\mathbf{U}$ have in comparison with $\mathbf{A}$. In MATLAB, this is controlled by setting a relative tolerance below which

elements in **L** and **U** are dropped. This tolerance is chosen as a compromise between good accuracy of the decomposition (favored by a small tolerance) and minimizing memory and CPU time for a matrix multiplication (which is favored by a high tolerance).

Also in the case with incomplete factorizations, it is strongly recommended to reorder the rows and columns of **A** before the incomplete factorization is computed.

Another, less complicated, preconditioner is symmetric successive overrelaxation (SSOR) [7], in which the preconditioning matrix **M** never is stored explicitly. Hence the memory requirements are smaller when SSOR is used as a preconditioner instead of some incomplete factorization of **A**.

An important note is that for the time-harmonic version of the curl-curl equation, and for low-frequency eddy current computations (Sect. 6.8.3), the null-space of the curl operator causes problems for the Krylov methods, and therefore more advanced preconditioners [26, 27, 46] are required.

Reliable implementations of Krylov methods and preconditioners are available, e.g., in the PETSc library [6]. Also MATLAB provides implementations of many popular Krylov methods.

C.2.3 Multigrid Methods

The multigrid (MG) method [33, 91] was introduced about four decades ago, but has only very recently been applied to Maxwell's equations [36]. The MG method can be used either as an iterative solver on its own, or as a very efficient preconditioner for iterative Krylov methods. It greatly improves the convergence rate of iterative solvers for large sparse matrices that occur in differential equation formulations. In fact, the convergence rate can be made independent of the cell size h, rather than to scale as some power of h.

The underlying principle is the observation that for the Laplace equation, the "short-wavelength error" (which varies on the scale of the grid) is reduced quickly by local operations (known as smoothers) such as Jacobi or Gauss–Seidel iterations; see Sect. 3.1.1. However, the long-wavelength error is reduced much more slowly by the smoothers. Since such error has short wavelength with respect to a *coarser* grid, one expects that this error can be reduced more rapidly on a coarse grid. Therefore, the basic idea of MG is to introduce a hierarchy of grids, starting from the finest one, and try to improve the solution on the finer grid by looking for a correction from the coarser grid. Optimally, the coarsest grid has only a small number of cells, and a direct solver can be used at a low computational cost.

So far, MG is used mostly for electrostatic and magnetostatic problems [59, 70] and transient eddy current problems. Generally, MG is among the most efficient solvers [33, 59] for Laplace-type equations. However, little research on MG has been devoted to fully electromagnetic problems, such as time-harmonic problems for eddy current computations [8, 35]. Certain difficulties (due to the null-space of the curl-curl operator) are encountered when this method is applied to the full

Table C.1 Capacitance vs. cell size for finite difference solution on larger grids

n	$h \times 10^2$	C [pF/m]
50	2.000	90.78080 583
100	1.000	90.68006 976
200	0.500	90.64044 979
300	0.333	90.62961 567
400	0.250	90.62481 230

Maxwell's equations. For wave problems, another complicating aspect is that the coarsest grid must resolve the wavelength $\lambda \propto 1/f$, which limits the hierarchy of grids and therefore the recursive MG algorithm.

C.3 Capacitance Calculation on Larger Grids

With the more efficient solvers we can extend the capacitance calculation of Sect. 3.1 to much larger grids. Results for grids up to 400 by 400 are shown in Table C.1.

One can estimate the order of convergence from formula (2.4) for 100, 200, and 400 points, and the order of convergence in h comes out as 1.341. This is close to the asymptotic result 4/3, which occurs for the 270° corners. If we do polynomial fits to $h^{4/3}$, the extrapolated value is 90.6145 pF/m. It should be pointed out that a *higher-order* fit to noninteger powers of h, such as $h^{4/3}$, is not an optimal representation, because the regular parts of the solution contribute errors that scale as h^2. Nevertheless, the extrapolation has added three figures of accuracy. If we tried to achieve this accuracy by a single calculation with uniform refinement of the grid, we would have to decrease h by more than a factor of 100, and the execution time would increase by at least 100^3, that is, one million times. Evidently, extrapolation can be a very efficient way of increasing the accuracy. In the chapter on finite elements, we show that the accuracy can also be improved by adaptive grid refinement, which aims at increasing the resolution in regions where the solution varies rapidly.

Appendix D
Krylov Methods

Here, we will discuss some iterative methods for solving large linear systems of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{D.1}$$

For large 3D problems, it is generally too demanding to use a direct solver. Iterative, so-called Krylov methods are often a much better choice for these problems. Multigrid methods, which we discussed very briefly in Sect. C.2.3, have proven even more efficient for many problems but will not be discussed here.

D.1 Projection Methods

In projection methods, one minimizes the residual

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x} \tag{D.2}$$

by an approach similar to the Galerkin and Petrov–Galerkin methods for finite elements. The vector $\mathbf{x}$ will be constructed as a sum of basis vectors $\mathbf{v}$; $\mathbf{x} = \mathbf{x}_0 + \sum_{i=1}^{m} \mathbf{v}_i y_i$, and y is an array of coefficients. This can be written compactly by introducing the matrix $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m)$ and the column vector $\mathbf{y} = (y_1, y_2, \ldots, y_m)^T$:

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{V}\mathbf{y}. \tag{D.3}$$

The vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$ span a space K_m of "basis" vectors. Similarly, one chooses a space L_m of "test" vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m$ and demands that on the mth step of the iteration the residual $\mathbf{r}_m$ be orthogonal to all vectors in L_m. If $K_m = L_m$, this is Galerkin's method; otherwise, it is a Petrov–Galerkin method.

The most important part of the iteration is the choice of the search directions $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$. The simplest case is that in which $\mathbf{A}$ is real and symmetric. The old-fashioned "steepest descent" method chooses the increment directions $\mathbf{v}_i$ in

T. Rylander et al., *Computational Electromagnetics*, Texts in Applied Mathematics 51, DOI 10.1007/978-1-4614-5351-2,

the gradient direction of the error functional $(\mathbf{x} - \mathbf{x}_{\text{exact}})^T \mathbf{A}(\mathbf{x} - \mathbf{x}_{\text{exact}})$, on every step of the iteration. It turns out that this is a bad strategy. When the matrix $\mathbf{A}$ is positive definite and symmetric, the number of iterations for the steepest descent method scales as the condition number of $\mathbf{A}$, that is, the ratio of largest to smallest eigenvalues, $\kappa = \lambda_{\max}/\lambda_{\min}$.

D.2 Krylov Methods

A better strategy is to generate the increment directions as $\mathbf{r}_0$, $\mathbf{A}\mathbf{r}_0$, $\mathbf{A}^2\mathbf{r}_0$, ..., $\mathbf{A}^{m-1}\mathbf{r}_0$, where $\mathbf{r}_0$ is the first residual. Then K is called a *Krylov space*. The Arnoldi algorithm does exactly this and projects out components of the new $\mathbf{v}$'s to keep them orthonormal.

1. *Choose a vector* $\mathbf{v}_1$ *of norm 1*
2. *For* $j = 1, 2, \ldots, m$, *Do:*
3. $\quad h_{ij} = (\mathbf{A}\mathbf{v}_j, \mathbf{v}_i)$ *for* $i = 1, 2, \ldots, j$
4. $\quad \mathbf{w}_j = \mathbf{A}\mathbf{v}_j - \sum_{i=1}^{j} h_{ij}\mathbf{v}_i$
5. $\quad h_{j+1,j} = (\mathbf{w}_j, \mathbf{w}_j)^{1/2}$
6. $\quad$ *If* $h_{j+1,j} = 0$ *then Stop*
7. $\quad \mathbf{v}_{j+1} = \mathbf{w}_j / h_{j+1,j}$
8. *EndDo*

GMRES is Arnoldi's method followed by a minimization of $(\mathbf{r}, \mathbf{r})$. This is a reliable method, and it has the nice property that the error decreases monotonically with the iteration number. The disadvantage of GMRES is that one needs to store all the incremental directions $\mathbf{v}_1, \ldots, \mathbf{v}_m$ to do the minimizations. Therefore, it can become very memory-demanding if the number of iterations is large. To circumvent the memory problem, one can restart GMRES after a certain number of iterations (typically 5 to 50). However, at the restart, orthogonality is lost.

There are cleverer ways of generating the incremental directions $\mathbf{v}$. The standard method, which assumes that $\mathbf{A}$ is symmetric, is the Lanczos method. Here it suffices to save three increment directions.

1. *Choose a start vector* $\mathbf{v}_1$ *of norm 1.*
2. *Set* $\beta_1 = 0$, $\mathbf{v}_0 = 0$
3. *For* $j = 1, 2, \ldots, m$, *Do:*
4. $\quad \mathbf{w}_j = \mathbf{A}\mathbf{v}_j - \beta_j \mathbf{v}_{j-1}$
5. $\quad \alpha_j = (\mathbf{w}_j, \mathbf{v}_j)$
6. $\quad \mathbf{w}_j = \mathbf{w}_j - \alpha_j \mathbf{v}_j$
7. $\quad \beta_{j+1} = (\mathbf{w}_j, \mathbf{w}_j)^{1/2}$. *If* $\beta_{j+1} = 0$ *then Stop*
8. $\quad \mathbf{v}_{j+1} = \mathbf{w}_j / \beta_{j+1}$
9. *EndDo*

This makes all the vectors $\mathbf{v}_i, i = 1, 2, \ldots,$ orthogonal (in infinite-precision arithmetic). With finite precision, orthogonality may be lost if the iteration runs many steps. Consequently, the iteration may have to be restarted.

A method that is related to the Lanczos method is the conjugate gradient (CG) method, where one keeps going in orthogonal directions. At least with infinite-precision arithmetic, this method can guarantee convergence when the number of steps equals the number of unknowns. The CG method for a symmetric $\mathbf{A}$ can be written as follows:

1. *Compute* $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 = \mathbf{r}_0$
2. *For* $j = 0, 1, \ldots,$ *until convergence, Do:*
3. $\quad \alpha_j = (\mathbf{r}_j, \mathbf{r}_j)/(\mathbf{A}\mathbf{p}_j, \mathbf{p}_j)$
4. $\quad \mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j$
5. $\quad \mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j$
6. $\quad \beta_j = (\mathbf{r}_{j+1}, \mathbf{r}_{j+1})/(\mathbf{r}_j, \mathbf{r}_j)$
7. $\quad \mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$
8. *EndDo*

An advantage of the CG method is that one does not store the whole history of incremental directions. For positive definite symmetric matrices, the required number of iterations for CG is proportional to the square root of the condition number of the matrix.

D.3 Nonsymmetric A

Lanczos Biorthogonalization

The symmetric Lanczos algorithm can be extended to nonsymmetric matrices. The biorthogonal Lanczos algorithm constructs a pair of biorthogonal bases

$$\mathbf{v}_1, \mathbf{A}\mathbf{v}_1, \ldots, \mathbf{A}^{m-1}\mathbf{v}_1,$$

$$\mathbf{w}_1, \mathbf{A}^T\mathbf{w}_1, \ldots, (\mathbf{A}^T)^{m-1}\mathbf{w}_1,$$

with the orthogonality property $(\mathbf{v}_i, \mathbf{w}_j) = \delta_{ij}$. The procedure can be written as follows:

1. *Choose two vectors* $\mathbf{v}_1, \mathbf{w}_1$ *such that* $(\mathbf{v}_1, \mathbf{w}_1) = 1$.
2. *Set* $\beta_1 = \delta_1 = 0$, $\mathbf{v}_0 = \mathbf{w}_0 = 0$
3. *For* $j = 1, 2, \ldots, m,$ *Do:*
4. $\quad \alpha_j = (\mathbf{A}\mathbf{v}_j, \mathbf{w}_j)$
5. $\quad \hat{\mathbf{v}}_{j+1} = \mathbf{A}\mathbf{v}_j - \alpha_j \mathbf{v}_j - \beta_j \mathbf{v}_{j-1}$
6. $\quad \hat{\mathbf{w}}_{j+1} = \mathbf{A}^T\mathbf{v}_j - \alpha_j \mathbf{w}_j - \delta_j \mathbf{w}_{j-1}$
7. $\quad \delta_{j+1} = |(\hat{\mathbf{v}}_{j+1}, \hat{\mathbf{w}}_{j+1})|^{1/2}$. *If* $\delta_{j+1} = 0$ *Stop*
8. $\quad \beta_{j+1} = (\hat{\mathbf{v}}_{j+1}, \hat{\mathbf{w}}_{j+1})/\delta_{j+1}$

9. $\mathbf{w}_{j+1} = \hat{\mathbf{w}}_{j+1}/\beta_{j+1}$, $\mathbf{v}_{j+1} = \hat{\mathbf{v}}_{j+1}/\delta_{j+1}$
10. *EndDo*

BICG and QMR

Relatively new methods are the biconjugate gradient (BICG) and quasi-minimal residual (QMR) algorithms. BICG is a generalization of CG to nonsymmetric matrices. BICG generates the space of test vectors from powers of $\mathbf{A}^T$ rather than of $\mathbf{A}$, so this is a Petrov–Galerkin method. The BICG method works as follows ($\mathbf{x}^*$ denotes the complex conjugate of $\mathbf{x}$):

1. *Set* $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$. *Choose* $\mathbf{r}_0^*$ *so that* $(\mathbf{r}_0, \mathbf{r}_0^*) \neq 0$
2. *Set* $\mathbf{p}_0 = \mathbf{r}_0$, $\mathbf{p}_0^* = \mathbf{r}_0^*$
3. *For* $j = 0, 1, \ldots,$ *until convergence, Do:*
4. $\alpha_j = (\mathbf{r}_j, \mathbf{r}_j^*)/(\mathbf{A}\mathbf{p}_j, \mathbf{p}_j^*)$
5. $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j$
6. $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{p}_j, \quad \mathbf{r}_{j+1}^* = \mathbf{r}_j^* - \alpha_j \mathbf{A}\mathbf{p}_j^*$
7. $\beta_j = (\mathbf{r}_{j+1}, \mathbf{r}_{j+1}^*)/(\mathbf{r}_j, \mathbf{r}_j^*)$
8. $\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j, \quad \mathbf{p}_{j+1}^* = \mathbf{r}_{j+1}^* + \beta_j \mathbf{p}_j^*$
9. *EndDo*

QMR uses the Lanczos procedure to generate the incremental directions but still manages to avoid saving the $\mathbf{v}$'s. Finally, QMR minimizes a quantity that is related to (but not quite the same as) the residual. Hence the name "quasi." QMR does not require storage of the $\mathbf{v}$ vectors. As long as it does not lose orthogonality, it is probably the most useful of the iterative schemes for nonsymmetric matrices. In case the method loses orthogonality, QMR can be restarted using the last $\mathbf{x}$ as a starting point.

A disadvantage of both BICG and QMR is that they also use the transpose of the matrix $\mathbf{A}$. Improvements in which $\mathbf{A}^T$ is eliminated are called BICGSTAB and TFQMR (transpose-free QMR).

D.4 Preconditioning

For good efficiency, the iterative solver must in general be combined with a preconditioner; i.e., (D.1) is multiplied by some approximate inverse of $\mathbf{A}$ from the left. This can strongly improve the convergence. A preconditioner that often works, and is commonly used for eddy current calculations, is the incomplete LU decomposition; see Appendix C.2.2. Iterative methods are described in [4, 6, 7, 34, 67].

References

1. T Abboud, J C Nédélec, and J Volakis. Stable solution of the retarded potential equations. *17th Annual Review of Progress in Applied Computational Electromagnetics, Monterey, CA*, pages 146–151, 2001.
2. M Abramowitz and I A Stegun. *Handbook of Mathematical Functions*. National Bureau of Standards, 1965.
3. F Alimenti, P Mezzanotte, L Roselli, and R Sorrentino. A revised formulation of model absorbing and matched modal source boundary conditions for the efficient FDTD analysis of waveguide structures. *IEEE Trans. Microwave Theory Tech.*, 48(1):50–59, January 2000.
4. O Axelsson. *Iterative Solution Methods*. New York, NY: Cambridge University Press, 1994.
5. C A Balanis. *Advanced Engineering Electromagnetics*. New York, NY: John Wiley & Sons, 1989.
6. S Balay, W Gropp, L Curfman McInnes, and B Smith. The portable, extensible toolkit for scientific computation. http://www-unix.mcs.anl.gov/petsc/petsc-2/, 2005.
7. R Barret, M Berry, T F Chan, J Demmel, J Donato, J Dongarra, V Eijkhout, R Pozo, C Romine, and H Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, 1994. available at: `ftp://ftp.netlib.org/templates/templates.ps`.
8. R Beck and R Hiptmair. Multilevel solution of the time-harmonic Maxwell's equations based on edge elements. *Int. J. Numer. Meth. Engng.*, 45(7):901–920, 1999.
9. J P Bérenger. A perfectly matched layer for the absorption of electromagnetic waves. *J. Comput. Phys.*, 114(2):185–200, October 1994.
10. J Bey. Tetrahedral grid refinement. *Computing*, 55(4):355–378, 1995.
11. M J Bluck and S P Walker. Time-domain BIE analysis of large three-dimensional electromagnetic scattering problems. *IEEE Trans. Antennas Propagat.*, 45(5):894–901, May 1997.
12. Alain Bossavit. *Computational Electromagnetism*. Boston, MA: Academic Press, 1998.
13. M M Botha and J M Jin. On the variational formulation of hybrid finite element–boundary integral techniques for electromagnetic analysis. *IEEE Trans. Antennas Propagat.*, 52(11):3037–3047, November 2004.
14. A C Cangellaris and D B Wright. Analysis of the numerical error caused by the stair-stepped approximation of a conducting boundary in FDTD simulations of electromagnetic phenomena. *IEEE Trans. Antennas Propagat.*, 39(10):1518–1525, October 1991.
15. F X Canning. Improved impedance matrix localization method. *IEEE Trans. Antennas Propagat.*, 41(5):659–667, May 1993.
16. F X Canning and K Rogovin. Fast direct solution of standard moment-method matrices. *IEEE Antennas Propagat. Mag.*, 40(3):15–26, June 1998.

T. Rylander et al., *Computational Electromagnetics*, Texts in Applied Mathematics 51, DOI 10.1007/978-1-4614-5351-2,

17. M Celuch-Marcysiak and W K Gwarek. Generalized TLM algorithms with controlled stability margin and their equivalence with finite-difference formulations for modified grids. *IEEE Trans. Microwave Theory Tech.*, 43(9):2081–2089, September 1995.
18. Z Chen, M M Ney, and W J R Hoefer. A new finite-difference time-domain formulation and its equivalence with the TLM symmetrical condensed node. *IEEE Trans. Microwave Theory Tech.*, 39(12):2160–2169, December 1991.
19. D K Cheng. *Fundamentals of Engineering Electromagnetics*. Reading, MA: Addison-Wesley, 1993.
20. W C Chew, J M Jin, E Michielssen, and J Song. *Fast and Efficient Algorithms in Computational Electromagnetics*. Norwood, MA: Artech House, 2001.
21. R Coifman, V Rokhlin, and S Wandzura. The fast multipole method for the wave equation: A pedestrian prescription. *IEEE Antennas Propagat. Mag.*, 35(3):7–12, June 1993.
22. D B Davidson. *Computational Electromagnetics for RF and Microwave Engineering*. Cambridge: Cambridge University Press, second edition, 2011.
23. T Davis. UMFPACK. http://www.cise.ufl.edu/research/sparse/umfpack/, 2005.
24. J W Demmel, J R Gilbert, and X S Li. SuperLU. http://crd.lbl.gov/~xiaoye/SuperLU/, 2005.
25. S J Dodson, S P Walker, and M J Bluck. Costs and cost scaling in time-domain integral-equation analysis of electromagnetic scattering. *IEEE Antennas Propagat. Mag.*, 40(4):12–21, August 1998.
26. R Dyczij-Edlinger and O Biro. A joint vector and scalar potential formulation for driven high frequency problems using hybrid edge and nodal finite elements. *IEEE Trans. Microwave Theory Tech.*, 44(1):15–23, 1996.
27. R Dyczij-Edlinger, G Peng, and J F Lee. A fast vector-potential method using tangentially continuous vector finite elements. *IEEE Trans. Microwave Theory Tech.*, 46(6):863–868, 1998.
28. K Eriksson, D Estep, P Hansbo, and C Johnson. *Computational Differential Equations*. New York, NY: Cambridge University Press, 1996.
29. R Garg. *Analytical and Computational Methods in Electromagnetics*. Norwood, MA: Artech House, 2008.
30. W L Golik. Wavelet packets for fast solution of electromagnetic integral equations. *IEEE Trans. Antennas Propagat.*, 46(5):618–624, May 1998.
31. R D Graglia. On the numerical integration of the linear shape functions times the 3-D Green's function or its gradient on a plane triangle. *IEEE Trans. Antennas Propagat.*, 41(10):1448–1455, October 1993.
32. D J Griffiths. *Introduction to Electrodynamics*. Upper Saddle River, NJ: Prentice-Hall, third edition, 1999.
33. W Hackbusch. *Multi-Grid Methods and Application*. Berlin: Springer-Verlag, 1985.
34. W Hackbush. *Iterative Solution of Large Sparse Linear Systems of Equations*. New York, NY: Springer-Verlag, 1994.
35. V Hill, O Farle, and R Dyczij-Edlinger. A stabilized multilevel vector finite-element solver for time-harmonic electromagnetic waves. *IEEE Trans. Magn.*, 39(3):1203–1206, 2003.
36. R Hiptmair. Multigrid method for Maxwell's equations. *SIAM J. Numer. Anal.*, 36(1):204–225, 1998.
37. W J R Hoefer. The transmission-line method – theory and applications. *IEEE Trans. Microwave Theory Tech.*, 33(10):882–893, October 1985.
38. T J R Hughes. *The finite element method: linear static and dynamic finite element analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1987.
39. P Ingelström. *Higher Order Finite Elements and Adaptivity in Computational Electromagnetics*. PhD thesis, Chalmers University of Technology, Göteborg, Sweden, 2004.
40. J M Jin. *The Finite Element Method in Electromagnetics*. New York, NY: John Wiley & Sons, 1993.
41. J M Jin. *The Finite Element Method in Electromagnetics*. New York, NY: John Wiley & Sons, second edition, 2002.
42. J M Jin. *Theory and Computation of Electromagnetic Fields*. New York, NY: John Wiley & Sons, 2010.

43. P B Johns. A symmetrical condensed node for the TLM method. *IEEE Trans. Microwave Theory Tech.*, 35(4):370–377, April 1987.
44. G Karypis. METIS. http://www-users.cs.umn.edu/~karypis/metis/, 2005.
45. P S Kildal, S Rengarajan, and A Moldsvor. Analysis of nearly cylindrical antennas and scattering problems using a spectrum of two-dimensional solutions. *IEEE Trans. Antennas Propagat.*, 44(8):1183–1192, August 1996.
46. Y Q Liu, A Bondeson, R Bergström, C Johnson, M G Larson, and K Samuelsson. Eddy-current computations using adaptive grids and edge elements. *IEEE Trans. Magn.*, 38(2):449–452, March 2002.
47. N K Madsen and R W Ziolkowski. A three-dimensional modified finite volume technique for maxwell's equations. *Electromagnetics*, 10(1-2):147–161, January-June 1990.
48. P Monk. *Finite Element Methods for Maxwell's Equations*. Oxford: Clarendon Press, 2003.
49. P B Monk. A comparison of three mixed methods for the time dependent Maxwell equations. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1097–1122, September 1992.
50. A Monorchio and R Mittra. A hybrid finite-element finite-difference time-domain (FE/FDTD) technique for solving complex electromagnetic problems. *IEEE Microw. Guided Wave Lett.*, 8(2):93–95, February 1998.
51. J C Nédélec. Mixed finite elements in R^3. *Numer. Math.*, 35(3):315–341, 1980.
52. N M Newmark. A method of computation for structural dynamics. *J. Eng. Mech. Div., Proc. Am. Soc. Civil Eng.*, 85(EM 3):67–94, July 1959.
53. S Owen. Meshing Research Corner. http://www.andrew.cmu.edu/user/sowen/mesh.html, 2005.
54. A F Peterson, S L Ray, and R Mittra. *Computational Methods for Electromagnetics*. New York, NY: IEEE Press, 1997.
55. P G Petropoulos, L Zhao, and A C Cangellaris. A reflectionless sponge layer absorbing boundary condition for the solution of Maxwell's equations with high-order staggered finite difference schemes. *J. Comput. Phys.*, 139(1):184–208, January 1998.
56. A J Poggio and E K Miller. Integral equation solutions of three-dimensional scattering problems. *Computer Techniques for Electromagnetics*, Oxford: Pergamon:159–264, 1973.
57. S M Rao and D R Wilton. Transient scattering by conducting surfaces of arbitrary shape. *IEEE Trans. Antennas Propagat.*, 39(1):56–61, January 1991.
58. S M Rao, D R Wilton, and A W Glisson. Electromagnetic scattering by surfaces of arbitrary shape. *IEEE Trans. Antennas Propagat.*, AP-30(3):409–418, May 1982.
59. S Reitzinger and M Kaltenbacher. Algebraic multigrid methods for magnetostatic field problems. *IEEE Trans. Magn.*, 38(2):477–480, 2002.
60. D J Riley and C D Turner. VOLMAX: A solid-model-based, transient volumetric Maxwell solver using hybrid grids. *IEEE Antennas Propagat. Mag.*, 39(1):20–33, February 1997.
61. V Rokhlin. Rapid solution of integral equations of classical potential theory. *J. Comput. Phys.*, 60(2):187–207, 1985.
62. V Rokhlin. Rapid solution of integral equations of scattering theory in two dimensions. *J. Comput. Phys.*, 86(2):414–439, 1990.
63. T Rylander and A Bondeson. Stability of explicit-implicit hybrid time-stepping schemes for Maxwell's equations. *J. Comput. Phys.*, 179(2):426–438, July 2002.
64. T Rylander and J M Jin. Perfectly matched layer for the time domain finite element method. *J. of Comput. Phys.*, 200(1):238–250, October 2004.
65. T Rylander, T McKelvey, and M Viberg. Estimation of resonant frequencies and quality factors from time domain computations. *J. of Comput. Phys.*, 192(2):523–545, December 2003.
66. B P Rynne. Instabilities in time marching methods for scattering problems. *Electromagnetics*, 6(2):129–144, 1986.
67. Y Saad. *Iterative methods for sparse linear systems*. Boston, MA: PWS Publishing, 1996.
68. M N O Sadiku. *Numerical Techniques in Electromagnetics with MATLAB*. Boca Raton, FL: CRC Press, third edition, 2009.
69. M Salazar-Palma, T K Sarkar, L E Garcia-Castillo, T Roy, and A Djordjevic. *Iterative and Self-Adaptive Finite-Elements in Electromagnetic Modeling*. Norwood, MA: Artech House, 1998.

70. M Schinnerl, J Schöberl, and M Kaltenbacher. Nested multigrid methods for the fast numerical computation of 3D magnetic fields. *IEEE Trans. Magn.*, 36(4):1557–1560, 2000.
71. R Schuhmann and T Weiland. Stability of the FDTD algorithm on nonorthogonal grids related to the spatial interpolation scheme. *IEEE Trans. Magn.*, 34(5):2751–2754, September 1998.
72. X Q Sheng and W Song. *Essentials of Computational Electromagnetics*. Singapore: John Wiley & Sons, 2012.
73. J R Shewchuk. Trianlge – a two-dimensional quality mesh generator and delaunay triangulator. http://www.cs.cmu.edu/~quake/triangle.html.
74. J R Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. *Lecture Notes in Computer Science*, 1148:203–222, May 1996.
75. P P Silvester and R L Ferrari. *Finite Elements for Electrical Engineers*. New York, NY: Cambridge University Press, second edition, 1990.
76. P D Smith. Instabilities in time marching methods for scattering: cause and rectification. *Electromagnetics*, 10(4):439–451, October–December 1990.
77. J M Song and W C Chew. The fast Illinois solver code: requirements and scaling properties. *IEEE Comput. Sci. Eng.*, 5(3):19–23, July–September 1998.
78. J M Song, C C Lu, and W C Chew. Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects. *IEEE Trans. Antennas Propagat.*, 45(10):1488–1493, October 1997.
79. J M Song, C C Lu, W C Chew, and S W Lee. Fast Illinois solver code (FISC). *IEEE Antennas Propagat. Mag.*, 40(3):27–34, June 1998.
80. A Taflove. *Computational Electrodynamics: The Finite-Difference Time-Domain Method*. Norwood, MA: Artech House, 1995.
81. A Taflove, editor. *Advances in Computational Electrodynamics: The Finite-Difference Time-Domain Method*. Norwood, MA: Artech House, 1998.
82. A Taflove and S C Hagness. *Computational Electrodynamics: The Finite-Difference Time-Domain Method*. Norwood, MA: Artech House, second edition, 2000.
83. P Thoma and T Weiland. Numerical stability of finite difference time domain methods. *IEEE Trans. Magn.*, 34(5):2740–2743, September 1998.
84. S Toledo, D Chen, and V Rotkin. TAUCS, A Library of Sparse Linear Solvers. http://www.tau.ac.il/~stoledo/taucs/, 2005.
85. D A Vechinski and S M Rao. A stable procedure to calculate the transient scattering by conducting surfaces of arbitrary shape. *IEEE Trans. Antennas Propagat.*, 40(6):661–665, June 1992.
86. R L Wagner and W C Chew. A study of wavelets for the solution of electromagnetic integral equations. *IEEE Trans. Antennas Propagat.*, 43(8):802–810, August 1995.
87. J J H Wang. *Generalized Moment Methods in Electromagnetics*. New York, NY: John Wiley & Sons, 1991.
88. K F Warnick. *NUMERICAL METHODS FOR ENGINEERING - An Introduction Using MATLAB and Computational Electromagnetics Examples*. Raleigh, NC: SciTech Publishing, 2011.
89. J P Webb. Hierarchal vector basis functions of arbitrary order for triangular and tetrahedral finite elements. *IEEE. Trans. Antennas Propagat.*, 47(8):1244–1253, 1999.
90. T Weiland. Time domain electromagnetic field computation with finite difference methods. *Int. J. Numer. Model. El.*, 9(4):295–319, July-August 1996.
91. P Wesseling. *An Introduction to Multigrid Methods*. Chichester: John Wiley & Sons, 1992.
92. R B Wu and T Itoh. Hybrid finite-difference time-domain modeling of curved surfaces using tetrahedral edge elements. *IEEE Trans. Antennas Propagat.*, 45(8):1302–1309, August 1997.
93. K S Yee. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Trans. Antennas Propagat.*, AP-14(3):302–307, May 1966.
94. K S Yee and J S Chen. The finite-difference time-domain (FDTD) and the finite-volume time-domain (FVTD) methods in solving Maxwell's equations. *IEEE Trans. Antennas Propagat.*, 45(3):354–363, March 1997.
95. K S Yee, J S Chen, and A H Chang. Numerical experiments on PEC boundary condition and late time growth involving the FDTD/FDTD and FDTD/FVTD hybrid. *IEEE Antennas Propagat. Soc. Int. Symp.*, 1:624–627, 1995.

Index

A
A (vector potential), 165
ABC, *see* absorbing boundary conditions
absorbing boundary conditions, 5, 85
adaptivity, **110**
  MoM, 199
Ampère's law, 3, 69
amplification factor, 50
assembling, **101**

B
B (magnetic flux), 4
barycentric coordinates, 103
basis functions
  for edge elements, 128
  nodal, 96
boundary conditions, **4**
  absorbing, 85
  Dirichlet, 98, 116
  essential, 98
  FDTD, PEC, 75
  homogeneous, 98
  natural, 98
  Neumann, 98
  perfectly matched layer, 85
  Robin, 98, 116
  Sommerfeld, 183

C
c (speed of light), 4, 63
capacitance
  computing, 23
  definition, 23
CEM, *see* computational electromagnetics
CFIE, *see* combined field integral equation
CFL condition, *see* Courant–Friedrichs–Levy condition
CG, *see* conjugate gradient method
collocation, 190
combined field integral equation, 209
computational electromagnetics, 1, 2
conjugate gradient method, 277
convergence, **11**
Coulomb gauge, 169
Courant condition, 66
Courant–Friedrichs–Levy condition, 66
curl-curl equation, 7, 43, 115
  weak form, 116

D
D (electric displacement), 4
Derivative operators
  D_x and D_{xx}, 29
differential equations solvers, 224
Dirac delta function, 187
Dirichlet boundary condition, *see* boundary conditions
dispersion relation, 7, 41
  exact, 8
  FDTD, 3D, 81–83
  numerical, 65
dissipation, 44, 101
divergence conforming elements, 207

E
E (electric field), 4
eddy current calculations, 165
  3D, 168
edge elements, **115**
  on bricks and tetrahedra, 118

T. Rylander et al., *Computational Electromagnetics*, Texts in Applied Mathematics 51, DOI 10.1007/978-1-4614-5351-2,

edge elements (*cont.*)
on rectangles, 117
on triangles, 128
EFIE, *see* electric field integral equation
eigenmode, 44
eigenvalue problems, **43**, 74, 120, 130, 144
eigenvalues
frequency-domain calculation of, 46
time-domain calculation of, 49
electric field integral equation, 205
element matrix, **105**, 122
energy, **5**
density, 5
electrostatic, 5, 108
magnetostatic, 6
ϵ (electric permittivity), 4
variable, 71
ϵ_0 (free-space electric permittivity), 4
ϵ_r (relative electric permittivity), 4
error estimation, 17
a posteriori, 12, 111
error indicator, 111
explicit time-stepping, *see* time-stepping, explicit
extrapolation, **12**

F
Faraday's law, 3, 69
fast Fourier transform, 55
fast multipole method, 228
FDTD, *see* finite-difference time-domain
FEM, *see* finite element method
FFT, *see* fast Fourier transform
finite difference derivatives of complex exponentials, **27**
across two cells, 30
on staggered grids, 30
second-order, 32
finite differences, **19**
across two cells, 20
first-order derivative, 20, 34
higher-order approximations, 34
noncentered, 20
on staggered grids, 20
second-order derivative, 20, 37
finite element
definition, 95
finite element method, **93**, 226
mixed-order, 113, 144
numerical integration, 143
reference element, 138
relation to FDTD, 115, 119–120
time-dependent problems, 163
finite integration technique, 227
finite volume method, 225
finite-difference time-domain method, **63**, 225
3D, 72
integral interpretation, 77
solenoidal magnetic flux density, 79
unit cell, 72
FIT, *see* finite integration technique
FMM, *see* fast multipole method
FVTD, *see* finite volume method

G
Galerkin's method, 95, 176, 191
gauge condition, 203
gauge transformation, 170, 202
Gauss's theorem, 4
Gauss–Seidel iteration, 22
Gaussian integration, *see* numerical integration
Green's function, **187**
3D electrostatics, 187
for the vector potential, 203
grid
square, 21
staggered, 20, 30, 31, 69, 70
unstructured, *see* mesh
group velocity, 8, 30

H
$\boldsymbol{H}$ (magnetic field), 4
Halléns equation, 211
Helmholtz equation
1D, 41, 45, 96
1D, discretized, 47
2D, 98
2D weak form, 100
hybrid methods, 230

I
initial conditions
discretized wave equation, 65
eigenfrequency calculation, 76
for Ampère's law, 6
for Faraday's law, 6
for FDTD, 71
for the vector wave equation, 7
integral equation solvers, 227
integral equations, 9
charge density, 186
integration by parts, 44
internal resonances, 206

J
$\boldsymbol{J}$ (electric current density), 4
j (imaginary unit), 7
$\boldsymbol{J}_s$ (surface current), 5
Jacobi iteration, 22

K
k (wavenumber), 7, 29

L
λ (wavelength), 29
Laplace transform, 55
Laplace's equation
  2D, 20
  2D, discretized, 22
  iterative solution of, 22
  quadratic form, 108
leap-frog, *see* time-stepping
Lorentz gauge, 202
low-frequency approximation, **8**, 165, 168
lumped, *see* mass lumping

M
magic time step, 65, 67, 83
magnetic field integral equation, 208
magnetostatics, 165
mass lumping, 115, 164
mass matrix, 121
MATLAB
  1D FD, Helmholtz equation, 47
  1D FD, wave equation, 52
  2D FD, capacitance of coaxial cable, 24
  2D FEM, edge elements, 130
  2D FEM, nodal elements, 107
  2D MoM, 194
  3D FDTD, 74
  3D mixed-order FEM, 144
  Padé approximation, 57
Maxwell's equations, 3
  1D, 113
  frequency domain, 43
mesh, 94
  computer representation, **106**
  generation of, 107
mesh refinement
  adaptive, 111
  uniform, 111
method of weighted residuals, 178
method of moments, **185**
MFIE, *see* magnetic field integral equation
midpoint integration, *see* numerical integration
mixed elements, 115
MLFMA, *see* multilevel fast multipole algorithm
MoM, *see* method of moments
μ (magnetic permeability), 4
  variable, 71
μ_0 (free-space magnetic permeability), 4
multigrid methods, 273
multilevel fast multipole algorithm, 228
μ_r (relative magnetic permeability), 4

N
$\boldsymbol{N}_i$, *see* basis functions for edge elements
near-to-far-field transformation, 87
Neumann boundary condition, *see* boundary conditions
Newmark scheme, 164
nodal basis functions, *see* basis functions, 100
nodal basis funtions, 103
nodal elements, 100
numerical dispersion, 65, 83
numerical integration
  for log-singularity, 200
  Gaussian integration, 18, 200
  midpoint integration, 12, 200
  Simpson's rule, 13
  trapezoidal rule, 200
Nystrom method, 190

O
ω (angular frequency), 7
open region problems, 84
overrelaxation, 23, 273

P
Padé approximation, 56
PEC, *see* perfect electric conductor
penalty term, 169
perfect electric conductor, 5
perfectly matched layer, 5, 85
Petrov–Galerkin method, 176
phase velocity, 8
ϕ (electrostatic potential), 9
φ_i (nodal basis function), 100
PML, *see* perfectly matched layer
point matching, 190
Poisson's equation, 3, 185
Prony's method, 58

Q
quadratic form, 172
  for Laplace's equation, 108

R
Rao–Wilton–Glisson elements, 207
Rayleigh–Ritz method, 175
reciprocity, 191
resolution, 11
ρ (electric charge density), 4
ρ_s (surface charge density), 5
Robin boundary condition, *see* boundary conditions

S
self-adjoint, 44, 45
σ (electric conductivity), 4
simplex coordinates, 103
Simpson's rule, *see* numerical integration
skin effect, 167
solenoidal magnetic flux density, condition of, 3
Sommerfeld boundary condition, 183
spurious modes, **31**, 115
SSOR, *see* symmetric successive overrelaxation
stability analysis, **50**, 65
  von Neumann, 50, 51
stability limit
  FDTD, 3D, 82
  general, 50
  wave equation, 1D, 66
staggered grids, *see* grid
staircase approximation, 3, 63, 225
stationary point, 172
stiffness matrix, 121
Stokes's theorem, 5
symmetric successive overrelaxation, 273

T
t (time), 4
Taylor expansion, 14, 19
TE modes, *see* transverse electric modes
tent functions, 96, 113
time-stepping
  explicit, 49, 63
  leap-frog, 50
  Newmark, 164
  stability limit, *see* stability limit
TLM, *see* transmission line method
TM modes, *see* transverse magnetic modes
top-hat functions, 113
transmission line method, 226
transverse electric modes, 61
transverse magnetic modes, 61
trial functions, *see* basis functions

U
ungauged formulation, 170

V
variational methods, **171**
vector Helmholtz equation, 7
vector wave equation, 7
von Neumann stability analysis, *see* stability analysis

W
wave equation
  1D, 7, 41, 52
  1D, discretized, 52, 64
  analytical solution,1D, 7
wavenumber, 7, 29
  analytical, 1D, 45
  numerical, 30
weak form
  Helmholtz equation, 1D, 97
  Helmholtz equation, 2D, 100
  vector Helmholtz equation, 116
weighted residuals, 190

Y
Yee cell, 72
Yee scheme, *see* finite-difference time-domain