Chapter 2
Convergence

The usage of computational electromagnetics in engineering and science more or less always originates from a physical situation that features a particular problem. Here, some examples of such situations could be to determine the radiation pattern of an antenna, the transfer function of a frequency selective surface, the scattering from small particles or the influence of a cell phone on its user. The physical problem is then described as a mathematical problem that involves Maxwell's equations. In a very limited number of cases, the mathematical problem can be solved analytically such that we have an *exact* solution in closed form. If there exists a solution to the problem that can not be calculated analytically, we can approximate the mathematical problem and pursue an *approximate* solution. In the context of CEM, such an approximate solution is often referred to as a numerical solution, since it typically involves extensive numerical computations in combination with relatively simple analytical expressions. These simple analytical expressions are normally applied to small subdomains of the original problem-domain, where the subdomain solutions are related to each other such that they collectively correspond to the solution to the original problem. The difference between an approximate solution and the exact solution is referred to as the error. It is desirable that the error can be reduced to an arbitrary low level such that the approximate solution *converge* to the exact solution, i.e. the accuracy of the numerical solution improves.

Thus, one must keep in mind that numerical tools never give the exact answer. The accuracy of the numerical result depends on the so-called resolution. Resolution may mean the number of grid points per wavelength in microwave problems, or how well the geometry of an electrical motor is represented by a finite element mesh. If the method works correctly, the computed answer will converge to the exact result as the resolution increases. However, with finite resolution, the error is nonzero, and one must estimate it to ensure that its magnitude is acceptable. This is particularly true for large systems, where it may be hard to resolve details of the geometry or to afford a sufficient number of points per wavelength. Examples of this state of affairs are found in 3D-modeling of electrical motors and generators, large array antennas, and computation of the radar cross sections of aircrafts.

T. Rylander et al., *Computational Electromagnetics*, Texts in Applied Mathematics 51, DOI 10.1007/978-1-4614-5351-2_2,

Applied mathematicians have derived a posteriori error estimates, which can be evaluated after an approximate numerical solution has been computed. However, such error estimates are only beginning to be established for Maxwell's equations, and discussion of these would take us far beyond an introductory course. For further information on this topic, see, e.g., [48, 69]. Nevertheless, error estimates are useful because they can be exploited for adaptive mesh refinement in regions that give large contributions to the error. A simpler method to estimate the error of a given computation is to do a convergence test by increasing the resolution uniformly, finding out the order of convergence, and then extrapolating the computed results to infinite resolution. That is the approach we will follow.

In general, one does not know the order of convergence of a computational method for a given problem a priori. Even though standard centered finite differences or linear finite elements converge with an error of order h^2 (where h is the grid spacing or the cell size) for regular problems, singular behavior of the solution decreases the order of convergence in most application problems. Singularities are introduced by sharp edges and tips of objects such as metallic conductors, dielectrics, and magnetic materials.

2.1 Extrapolation to Zero Cell Size

We will use a very simple problem, namely to calculate the electrostatic potential on the symmetry axis of a uniformly charged square, to illustrate how computed results can be extrapolated to zero cell size. The square is the region $-a < x < a,\ -a < y < a,\ z = 0$, the surface charge density $\rho_s(x, y) = \rho_{s0}$ is constant, and we seek the potential ϕ at two points on the symmetry axis: $(0, 0, a)$ and $(0, 0, 0)$. Using the symmetry, we can write the potential from this charge distribution as

$$\phi(0,0,z) = \frac{\rho_{s0}}{4\pi\epsilon_0}\int_{x'=-a}^{a} dx' \int_{y'=-a}^{a} \frac{dy'}{(x'^2 + y'^2 + z^2)^{1/2}} = \frac{\rho_{s0}}{\pi\epsilon_0} I(z,a),$$

with

$$I(z,a) \equiv \int_{x'=0}^{a} dx' \int_{y'=0}^{a} \frac{dy'}{(x'^2 + y'^2 + z^2)^{1/2}}. \tag{2.1}$$

To do the integral $I(z, a)$ numerically, we split the square into n^2 smaller squares of side $h = a/n$, and on each square, apply a simple integration rule such as midpoint integration

$$\int_x^{x+h} f(x)dx \approx hf\left(x + \frac{h}{2}\right) \tag{2.2}$$

or Simpson's rule

$$\int_x^{x+h} f(x)dx \approx \frac{h}{6}\left[f(x) + 4f\left(x + \frac{h}{2}\right) + f(x+h)\right] \tag{2.3}$$

in two dimensions. The integration can be written as a MATLAB function.

```
% ---------------------------------------------------------------
% Compute potential on symmetry axis of square plate
% ---------------------------------------------------------------
function pot = integr(z, a, n, rule)

% Arguments:
%    z    = the height over the plate
%    a    = the side of the square
%    n    = the number of elements along each side of the plate
%    rule = a string 'midpoint' or 'simpson' that specifies
%           the integration rule
% Returns:
%    pot = the potential at the point (0,0,z)

x  = linspace(0, a, n+1);
y  = linspace(0, a, n+1);
h  = a/n;
zs = z^2;

if (strcmp(rule, 'midpoint'))

  % Midpoint integration
  xs(1:n) = (x(1:n) + h/2).^2;
  ys(1:n) = (y(1:n) + h/2).^2;
  [xxs, yys] = meshgrid(xs,ys);

  int = sum(sum(1./sqrt(xxs + yys + zs)));

elseif (strcmp(rule, 'simpson'))

  % Simpson's rule
  int = 0;
  for i = 1:n
    x1 = x(i)^2; x2 = (x(i) + h/2)^2; x3 = (x(i) + h)^2;
    y1(1:n) = y(1:n).^2;
    y2(1:n) = (y(1:n) + h/2).^2;
    y3(1:n) = (y(1:n) + h).^2;
    int = int + sum(  1./sqrt(x1+y1+zs) + 1./sqrt(x1+y3+zs) ...
                    + 1./sqrt(x3+y1+zs) + 1./sqrt(x3+y3+zs)...
                    + 4./sqrt(x2+y1+zs) + 4./sqrt(x2+y3+zs)...
                    + 4./sqrt(x1+y2+zs) + 4./sqrt(x3+y2+zs)...
                    + 16./sqrt(x2+y2+zs))/36;
  end
```

Table 2.1 Integral $I(1,1)$ from numerical integration with different cell sizes

n [-]	h [m]	$I_{\text{midp}}(1,1)$ [m]	$I_{\text{Simpson}}(1,1)$ [m]
5	0.20000	0.79432 30171	0.79335 94378
7	0.14286	0.79385 04952	0.79335 92042
10	0.10000	0.79359 97873	0.79335 91413
15	0.06667	0.79346 60584	0.79335 91252
20	0.05000	0.79341 92684	0.79335 91225

```
else

  error(['Only midpoint integration and Simpson''s rule are ' ...
         'implemented'])

end

pot = int*h^2;
```

We call this function with $z = a = 1$ [`integr(1,1,n,rule)`] and different numbers of grid points `n` for `rule = 'simpson'` and `'midpoint'`, and then extrapolate the results to zero cell size to get as accurate an answer as possible. The first step is to establish the order of convergence. Table 2.1 shows some results of calling the function for different cell sizes $h = 1/n$.

We can carry out the extrapolation using MATLAB routines, by collecting the values of h, I_{midp}, and I_{Simpson} in vectors. Plotting I_{midp} versus h to some power p, we find an almost straight line for $p = 2$, as shown in Fig. 2.1. This indicates that the midpoint rule gives quadratic convergence, i.e., $I_{\text{midp}}(h) = I_0 + I_2 h^2 + \cdots$ where I_0 is the extrapolated result. The term $I_2 h^2$ in the Taylor expansion of I_{midp} is the dominant contribution to the error when h is sufficiently small, and for such resolutions the higher-order terms in the Taylor expansion can be neglected.

We extrapolate the computed results as a polynomial fit in h^2 using the MATLAB command

```
pfit = polyfit(h.^2,I,m)
```

Here, m is the order of the polynomial, and the extrapolated value of the integral is the coefficient for h^0. [With the MATLAB convention for storing polynomials, this is the $(m+1)$th component of `pfit`]. A first-order fit ($m = 1$) gives the extrapolation $I(1,1) \approx 0.79335\,88818$, second-order ($m = 2$) gives $0.79335\,91208$, and a third-order fit gives $0.79335\,91213$.

The results from the Simpson integration fall on an almost straight line when plotted against h^4, and we conclude that the dominant error scales as h^4. A fit of $I_{\text{Simpson}}(1,1)$ to a linear polynomial in h^4 gives the extrapolation 0.79335 91207, and quadratic and cubic fits give 0.79335 91202.

The correct answer to eight digits is $0.79335\,912$. Extrapolation allows us to establish this degree of accuracy with a rather moderate effort: a second-order fit of the low-order midpoint rule versus h^2, using data computed for rather coarse grids $h \geq 0.05$. This gives eight-digit accuracy of the extrapolation even though the

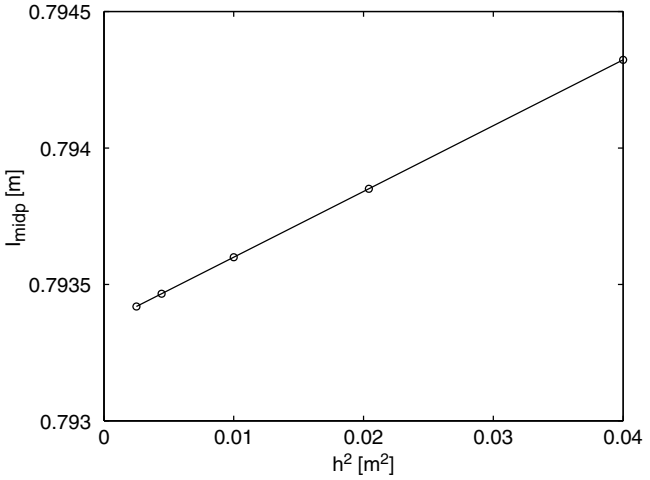

Fig. 2.1 Values of the integral $I(1, 1)$ computed by the midpoint rule, plotted versus h^2

computed data has only three to four correct digits. Thus, extrapolation can bring very significant improvements of accuracy. Another advantage of extrapolation is that it makes us aware of how good the accuracy is. The example shows that good accuracy can also be obtained by using the higher-order Simpson integration, even without extrapolation, on a grid of moderate size.

A simple way to estimate the order of convergence is to carry out computations for a geometric sequence of cell sizes such that $h_i / h_{i+1} = h_{i+1} / h_{i+2}$. Assuming that the lowest-order term in the expansion of the error is sufficient, i.e. $I(h) = I_0 + I_p h^p$, and that the cell sizes form a geometric series, one can then estimate the order of convergence as

$$p = \ln\left[\frac{I(h_i) - I(h_{i+1})}{I(h_{i+1}) - I(h_{i+2})}\right] \Bigg/ \ln\left[\frac{h_i}{h_{i+1}}\right]. \tag{2.4}$$

When applied to the computed results for $h = 0.2, 0.1$ and 0.05, this formula gives $p = 2.002$ for the midpoint rule and $p = 3.985$ for Simpson, indicating that the convergence is quadratic and quartic, respectively, for the two methods.

2.1.1 A Singular Problem

It is instructive to consider a more singular problem, such as the potential on the midpoint of the plate, $z = 0$. Now, the integrand is singular, but the integral is nevertheless convergent. For this problem, Simpson integration gives a divergent result and cannot be used. (This illustrates the fact that high-order methods often

experience difficulties in the presence of singularities.) However, the midpoint integration still works, and for the cell sizes above we find the following values for $I_{\text{midp}}(0,1)$: 1.684320, 1.706250, 1.722947, 1.736083, 1.742700. Plots of I_{midp} versus h^p reveal that the order of convergence is now lower, $p = 1$. Nevertheless, we can still extrapolate using fits to polynomials in h. The results are linear, 1.762015; quadratic, 1.762745; cubic, 1.762748. This integral can be done analytically: $I(0,1) = 2\ln(1+\sqrt{2}) \approx 1.762747$. Thus, despite the singularity, the midpoint rule gives six-figure accuracy with $h \geq 0.05$ and quadratic extrapolation.

Review Questions

2.1-1 What is meant by resolution in the context of numerical computations? Give some examples.
2.1-2 How can the error in a computation be estimated?
2.1-3 What influences the error and the order of convergence?
2.1-4 Give a couple of examples of numerical integration rules and provide a simple comparison. Especially consider the differences for smooth and singular integrands.

2.2 Practical Procedures

The example we have just studied is very simple. Real application problems have more complex geometry than a square, but on the other hand, six-digit accuracy is very rarely needed, or even possible to achieve. Furthermore, numerical results converge in the very regular way we found here only if the grid can be refined uniformly over the whole computational region. When this is not possible, the convergence may be oscillatory, and the extrapolation to zero cell size becomes more difficult. In practice, it is often possible to extract a main power of convergence with the number of grid cells, but the remainder is too oscillatory to be convincingly fit by higher-order polynomials. A more robust and practical procedure for such cases is to use a linear fit of the computed results to h^p, where p is the estimated order of convergence. When the converged answer is not known, but the convergence is sufficiently regular, the order of convergence can be estimated from results for three different resolutions. To ascertain that the estimated order of convergence is not accidental, at least four different resolutions should be used. Once the order of convergence is established, extrapolation to zero cell size can be made by fitting a lowest-order expansion

$$I(h) = I_0 + I_p h^p \tag{2.5}$$

to the computed results.

Review Question

2.2-1 Why can extrapolation to zero cell size be difficult for nonuniformly refined grids?

Summary

- The accuracy of a numerical result depends on resolution. For example, a domain of integration can be divided into segments of size h, and a numerical evaluation of the integral I is then expressed as $I(h) = I_0 + I_p h^p + \cdots$, where I_0 is the exact result, $I_p h^p$ is the dominant error term (provided that h is sufficiently small), and p is the order of convergence.
- The order of convergence p can be estimated from

$$p = \ln\left[\frac{I(h_i) - I(h_{i+1})}{I(h_{i+1}) - I(h_{i+2})}\right] \Bigg/ \ln\left[\frac{h_i}{h_{i+1}}\right],$$

  which requires at least three computations and where $h_i / h_{i+1} = h_{i+1} / h_{i+2}$. The result should preferably be verified for at least four resolutions to ascertain that the estimated p is not accidental.
- A simple method to estimate the error of a given computation is to (i) do a convergence test by uniform grid refinement, (ii) find the order of convergence, and (iii) extrapolate the computed results to zero cell size.
- The order of convergence depend on the method *and* the regularity of the solution. Singular behavior of the solution decreases the order of convergence p in many real-world problems.

Problems

P.2-1 Derive the order of convergence for midpoint integration (2.2) and Simpson's rule (2.3) under the assumption that the integrand is regular. How does a singular integrand influence your derivation?

P.2-2 Show that (2.4) gives an estimate for p. Under what conditions is this estimate accurate?

Computer Projects

C.2-1 Repeat the calculations of $I(1,1)$ and $I(0,1)$, where $I(z,a)$ is defined in (2.1), using two-point Gaussian integration

$$\int_x^{x+h} f(x)dx = \frac{h}{2}\left[f\left(x+\frac{h}{2}\left(1-\frac{1}{\sqrt{3}}\right)\right)+f\left(x+\frac{h}{2}\left(1+\frac{1}{\sqrt{3}}\right)\right)\right]$$

and find the order of convergence.

C.2-2 Calculate the integral $\int_0^1 x^{-\alpha}dx$, with a singular integrand, numerically by dividing the interval into equal elements and applying midpoint integration on each. Investigate the cases $\alpha = 0.5$ and 0.8, find the order of convergence, and extrapolate to zero cell size. The exact integral is $1/(1-\alpha)$.

http://www.springer.com/978-1-4614-5350-5

Computational Electromagnetics

Rylander, Th.; Ingelström, P.; Bondeson, A.

2013, XX, 288 p., Hardcover
ISBN: 978-1-4614-5350-5