NTNU – Trondheim
Norwegian University of
Science and Technology

Improving Accuracy of Computational Light Scattering Simulations by Improved Numerical Integration

Jørgen Røysland Aarnes

Master of Science in Physics and Mathematics
Submission date: June 2014
Supervisor: Ingve Simonsen, IFY

Norwegian University of Science and Technology
Department of Physics

Abstract

Numerical integration plays an important role in many large scale physics simulations, yet the accuracy of such computations are often overlooked when the integrator routine is implemented—if the integration result itself is not a part of the simulation output. Errors from inaccurately computed integrals may propagate through the computations and severely impact the final result.

The aim of this thesis is to improve the accuracy of the numerical approximations of two classes of integrals. These integrals are computed in large numbers in a Maxwell Eqs. solver—developed to study light scattering from randomly rough surfaces. Previously, they have been computed with a midpoint rule, a crude, yet efficient integration scheme.

An automatic integrator based on the Gauss-Patterson nested quadrature scheme has been developed. The particular quadrature was chosen to minimize the number of function evaluations needed to reach a requested accuracy. The integrator routine performed well for general-purpose integration, but was deemed inefficient when studied in detail for the two classes of integrals from the Maxwell Eqs. solver. This was mainly due to the accuracy of the integrator being limited by an inaccurate surface representation used in the integrand functions' argument, related to a coarse discretization in the light scattering system. A more appropriate integrator was a Gauss-Patterson based rule evaluation quadrature routine. This routine approximates an integral using either three of seven function evaluations, regardless of the complexity of the computed integral. The accuracy of the computed integrals was improved by several orders of magnitude, when computed with the rule evaluation routine rather than the midpoint rule—with a modest increase in the number of function evaluations, from one to three or seven.

When a measure of energy conservation was used to study the impact of the rule evaluation routine incorporated in the Maxwell Eqs. solver, however, the expected accuracy improvement was not reached. This may be due to the energy conservation measure not being an appropriate accuracy measure in the simulations. The computational time increase, on the other hand, was less than expected, with an average factor of 1.54 for single precision and 2.77 for double precision computations.

Sammendrag

Numerisk integrasjon spiller en viktig rolle i mange fysikksimuleringer, men presisjonen i slike utregninger blir ofte oversett når integratoren implementes – hvis ikke integrasjonsresultatet i seg selv er en del av simuleringsresultatet. Feil fra unøyaktig beregnede integraler kan forplante seg gjennom simuleringen og merkbart påvirke det endelige resultatet.

Målet med denne avhandlingen er å forbedre presisjonen i numerisk utregning av to klasser med integraler. Integralene regnes i stort antall i en Maxwells ligninger-løser, Maxwell1D – utviklet for å studere lysspredning fra ru overflater. Tidligere er de blitt tilnærmet med midtpunktregelen, en unøyaktig, men effektiv integrasjonsmetode.

En automatisk integrator basert på Gauss-Pattersons nøstede kvadratur er utviklet. Kvadraturet er valgt for å minimere antall funksjonsevalueringer som trengs for å nå en forespurt presisjon i integrasjonen. Den automatiske integratoren oppnådde gode resultater for generell integration, men ble ansett som lite effektiv da den ble studert i detalj for de to klassene av integraler som inngår i Maxwell1D. Dette skyldes hovedsakelig at presisjonen i integratoren blir begrenset av en unøyaktig overflaterepresentasjon som inngår i argumentet til integranden, i forbindelse med en grov diskretisering i lysspredningssystemet. En mer hensiktsmessig integrator var en Gauss-Patterson basert regelevalueringsrutine. Denne approksimerer et integral med tre eller syv funksjonsevalueringer, uavhengig av kompleksiteten av det gjeldende integralet. Presisjonen var flere størrelsesorden bedre da de studerte integralene ble regnet med integratoren, fremfor med midtpunktetregelen. Dette med en moderat økning i antall funksjonsevaluering, fra en til tre eller syv.

Da et mål på energikonservering i systemet ble anvendt for å studere virkningen av regelevalueringsrutinen implementert i Maxwell1D innfridde imidlertid ikke integratoren forventningene, med tanke på forbedret presisjon. Dette kan skyldes at energikonserveringsmålet ikke er et passende mål på økt presisjon i simuleringene. Økningen i kjøretid var lavere enn ventet, med en gjennomsnittlig økning med faktor 1,54 for enkelt presisjon og 2,77 for dobbel presisjon.

Preface

This thesis is submitted in fulfillment of the requirements for a master degree in physics and mathematics at the Department of Physics at the Norwegian University of Science and Technology (NTNU). The work included in this thesis was mainly done during the spring semester of 2014, from late January to early June. The initial development and results presented in Chap. 1 is based on a project done in preparation to writing a master thesis, during the autumn of 2013. All tables and figures in Chap. 1 where produced during the autumn project, but beyond this, the documentation has been re-written and revised, to better suit the format and quality requirements of this thesis.

Some source code is included in the appendices of the thesis. The format is, however, not appropriate to utilize the programs developed (nor are the programs included in full). If this code is of interest, e.g., for direct use or further development, do not hesitate to contact me by email or by similar means.

The thesis is original, unpublished, independent work by the author, J. R. Aarnes. The Maxwell Eqs. solver mentioned numerous times throughout the thesis is a software developed by Ingve Simonsen and several collaborators. I. Simonsen, Professor at the Department of Physics at NTNU, has had the role of advisor, supervisor and motivator during the time spent working with this thesis. I would like extend my acknowledgment and thanks to him for encouraging support and guidance, in a time of where we both have had a lot on our minds.

I would also like to express a special thanks to Maria B. Hesjedal, fellow student, colleague and partner, for reminding me that there is a lot more to each day than academic pursuits and struggles.

Contents

Introduction

Evaluation and application of integrals is not only a central theme of mathematics, in large scale physics simulation one often needs to evaluate a vast number of integrals. These integrals may often not be solved analytically, and even if they could they might require tedious handling of a lot of special cases and many lines of code not applicable for general cases. Numerical integration techniques, or rather, numerical integrator routines are therefore of great importance in such simulations. Integrator routines that will approximate an integral at the accuracy required by the user—without stalling the program due to poor computational speed—are particularly useful.

A vital part of an integrator routine is the numerical integration formula, also known as the quadrature scheme. There are many quadrature schemes of different complexity and sophistication, available for implementation through descriptions in research articles and books, or even directly applicable through numerical libraries and software packages. As a computational scientist encounters a problem where numerical evaluation of an integral is needed, a common approach to the problem is to either use a library routine or write a integrator routine based on a simple quadrature scheme. At a later time the scientist might return to the integral in question. This usually is the case if

a) the numerical integration constitutes a bottleneck, with regards to computational speed, in the developed software, or
b) the output of the software is not sufficiently accurate, and it is suspected that the fault lies in the numerical evaluation of the integral.

One way to deal with such problems is to optimize the existing integrator. However, in optimizing a numerical integrator, there will always be a trade-off between efficiency (computational cost) and accuracy. If the scientist's problem is that the numerical integration constitutes a bottleneck in the software, a less strict accuracy requirement may solve the problem. To increase the accuracy, on the other hand, the scientist might increasing the number of integrand function evaluations on the integration interval or including a convergence criteria in the routine—a required accuracy to reach before returning the approximated integral result. None of these methods guarantee a more accurate result, but they close to guarantee an increase in the computational cost of the numerical integration.

Simple integrator routines are often very inefficient if high accuracy results are required, and reducing the computational cost of a library routine often is very hard—as it is most likely highly optimized out of the box. In many cases, analyzing the problem at hand and replacing the integrator with a routine using a different quadrature scheme, more appropriate for the specific integrals that are computed, is better strategy than to optimize the existing integrator.

The problem that is dealt with in this thesis is the numerical evaluation of a large number of integrals in the software package *Maxwell1D*—a Maxwell Eqs. solver developed to study light scattering from randomly rough surfaces. Specifically, the aim of this thesis is to improve the accuracy of the numerical approximations of these integrals, while keeping the computational cost at an acceptable level.

The integrals in Maxwell1D may be considered as two classes of integrals, or two problem families. One that involves the zeroth order and one that involves the first order Hankel function of the first kind, in the integrand function. The integrals in a class differ only in the integration intervals where they are evaluated. All integration intervals are small. Previously the integrals have been approximated by the midpoint rule, that is, by a simple and crude integrator that evaluates the integrand function in a single point on the integration interval. Such a quadrature scheme represents an extreme in the trade-

off between efficiency and accuracy. The midpoint rule integrator does not constitute a bottleneck in the software (at least not one that can be removed without drastically changing the functionality of the software), but is highly unstable with regards to the output's accuracy. With this in mind a choice has been made: To replace the midpoint rule integration by a quadrature routine that increases the accuracy of the computed integral, thus, inevitably, also increasing the computational cost of the integral evaluation.

In order to let the user of the Maxwell Eqs. solver choose the accuracy of the computed integrals an automatic integrator is appropriate, as automatic integrators are numerical integrator routines that aim to reach a requested accuracy before returning the result. Usually this is done by increasing the number of points where the integrand function is evaluated until the approximated result has an error estimate that is sufficiently small. Many such integrators exist, and they often make use of adaptive integration techniques: Focusing the computational power on the parts of the integration interval where the numerical result is hardest to compute accurately. Unfortunately, with adaptive integration the amount of information that may be re-used, when increasing the number of points where the integrand function is evaluated, is small. This impacts the routine's efficiency, especially if the evaluation of the integrand function itself is computationally expensive—as is the case with the Hankel functions in the integrand.

An automatic integrator routine that uses a nested quadrature scheme, on the other hand, allows for re-use of all integrand function evaluations when the number of points where the integrand function is evaluated is increased. One such scheme is the Gauss-Patterson nested quadrature scheme developed as an extension to existing Gaussian quadratures. The high cost of evaluating the integrand function in the integrals at hand, and the need keep computational cost down while increasing the accuracy of the approximated integrals—as the competing routine is a midpoint rule integrator—makes the Gauss-Patterson nested quadrature scheme a natural choice of quadrature scheme for the integrator developed in this thesis.

The work done during the time spent with this thesis can be separate into three parts. These parts:

1. Initial development – Writing code for the automatic integrator routine utilizing the Gauss-Patterson quadrature scheme and testing the routine for a variety of integrals. This general-purpose testing establish the all over performance characteristics of the integrator routine, not only giving a first impression of the routines performance, but also ruling out certain classes of integrals as unsuitable for the developed integrator to compute.
2. Customization and performance – Taking measures to specialize the routine for the two types of integrals that occur frequently in Maxwell1D. This includes deriving analytic expressions for these integrals from the theory of light scattering from rough surfaces, an in-depth analysis of the integrand functions, customizing the developed integrator routine and rigorously testing for the integrals that it is now specialized to compute. The test are is done with parameter configurations that are as realistic as possible, that is, as close to the actual computations the routine will do when implemented in the Maxwell Eqs. solver.
3. Application – Final customization, based on the results from the previous parts, and utilization of the routine in the Maxwell Eqs. solver.

As the documentation of this process is rather extensive, a chapter based structure—based on the these three parts—is regarded appropriate for this thesis. All relevant theory and methods are presented in the chapter where it is first used. The third and final chapter also includes concluding remarks and suggestions for further work.

Chapter 1

Initial development

The general one-dimensional definite integral is given by

$$I = \int_a^b f(x)\mathrm{d}x, \tag{1.1}$$

where $f(x)$ is the integrand function and $-\infty \leq a \leq b \leq \infty$ are the left and right integration limits, respectively. Such an integral often needs to be evaluated numerically, for different reasons. The integral may be incapable of being evaluated on closed form, or—if it exists—arriving at the analytic solution of the integral may require tedious and time-consuming pen and paper derivations (which often requires even an expert to battle through a large number of errors following the algebraic manipulation), or the integrand, $f(x)$, may not be known precisely (perhaps it is only given in tabular form, for certain values of x). As the applications of numerically approximated integrals on the form of Eq. (1.1) are vast, so are the number of methods to compute them.

This chapter includes the general-purpose development of the automatic integrator routine based on the Gauss-Patterson quadrature scheme. Testing is a central part of the development of an integrator routine, and the test presented in this chapter are battery experiments—testing the integrator routine on a set of integrals with varying properties in order to establish a basis for the expected all-over performance of the routine. The implemented routine and routine testing may seem too comprehensive for the specific task that constitutes the motivation for developing it. It is, however, expected that the automatic integrator routine will perform well for a variety of integrals, and this chapter will not only serve as a starting point for the specialized analysis in the chapters to come, but also as documentation for the routine, making it may easier to use for other problems than those in the Maxwell Eqs. solver.

Initially, theory about numerical integration is presented, both terminology and general concepts. Some terms, like integrator routine and a routine's efficiency, have been used already, but lack a proper definitions. Following this, the basic algorithms that constitute the integrator routine are described. Further, some notes on how the integrator routine is used is included. The methods used for testing the integrator are described in detail, with the purpose of simplifying the repeatability of the experiments performed. Results for the tests are presented and discussed towards the end of the chapter.

1.1 Theoretical background on numerical integrators

1.1.1 Quadratures

When employed as an approximation to a definite integral, a sum en the form

$$I_n \equiv \sum_{i=1}^{n} w_i f(x_i) \tag{1.2}$$

is called a numerical *quadrature* or a numerical *integration formula*. The n distinct points x_i are called the *nodes* or *abscissae* and the quantities w_i are called the *coefficients* or *weights* [8, 15]. As the sum of Eq. (1.2) denotes a general quadrature, the basic problem of numerical integration is to find integration points (abscissae and related weights) that minimizes the difference between the computed sum and the actual integral result for a large class of functions, $f(x)$ [15]. The n abscissae and n weights provide $2n$ degrees of freedom in the construction of a quadrature.

A measure of the accuracy of a quadrature is the polynomial degree of precision. A quadrature of degree d is exact for all polynomial integrands of degree $\leq d$ (and not exact for all polynomials of degree $> d$) [28].

1.1.2 Newton-Cotes formulae and Gaussian quadratures

A particularly simple quadrature scheme is the *midpoint rule*. This is an *open Newton-Cotes formula*, that is, the quadrature does not include the integration interval's endpoints in the calculations (open formula) and it belongs to a class of quadrature formulae with equally spaced abscissae (Newton-Cotes formulae).

Unique for this quadrature is that it approximate the definite integral by evaluating the integrand function in a single point—the midpoint of the integration interval—and it uses the size of the integration interval as the quadrature coefficient. The midpoint rule is expressed by:

$$M_1 = (b-a) f(\frac{b-a}{2}). \tag{1.3}$$

The result is a simple and crude approximation of the integral, with degree of precision equal to one [14]. More commonly used Newton-Cotes formulae are the trapezoidal rule and Simpson's rule, and their *composite/extended* versions (for details see, e.g., Ref. [8,29]). As the abscissa are equally spaced in all Newton-Cotes formulae these quadrature schemes have in common that the degree of freedom in constructing them is reduced by a factor of 2. In general an n-point Newton-Cotes formula is exact for polynomials of degree at most n or $n-1$, depending on whether n is odd or even, and whether the formula is open or closed [15].

A more sophisticated class of numerical quadratures, than the Newton-Cotes formulae, are the Gaussian quadratures. The quadratures in this class differ from Newton-Cotes formulae in the respect that the abscissae are not equidistant. As the Gaussian quadratures utilize all $2n$ degrees of freedom, inherent in the choice of n abscissa and n weights, the Gaussian quadratures are the quadratures with the highest possible degree of precision, $d = 2n-1$ [15, 28].

The theory of the Gaussian quadratures is closely related to the theory of orthogonal polynomials (for details see, e.g., Chaps. 22 & 25 in [2]). The simplest of these is the Legendre-polynomial, related to the construction of the Gauss-Legendre quadrature formula. This quadrature approximates the definite integral with integration limits $a = -1$

and $b = 1$ (Eq. (25.4.29) in [2]). Any definite integral on the form of Eq. (1.1) may be re-scaled to fit these integral limits by

$$\int_a^b f(x)\mathrm{d}x = \frac{b-a}{2}\int_{-1}^{1} f(\frac{b-a}{2}u + \frac{b+a}{2})\mathrm{d}u. \tag{1.4}$$

Hence, the integral is approximated by the Gauss-Legendre quadrature by

$$\int_a^b f(x)\mathrm{d}x \approx \frac{b-a}{2}\sum_1^n w_i f(\frac{b-a}{2}u_i + \frac{b+a}{2}), \tag{1.5}$$

where the weights and abscissae may be calculated from the orthogonal polynomial. These values for the Gauss-Legendre quadrature—and several others—are well known tabulated values, and may be found in, e.g., [34].

Essentially, the degree of precision of the Gaussian quadratures is twice that of the Newton-Cotes formulae, for the same number of integration points. This does not, however, guarantee twice the accuracy when using Gaussian quadratures, as high degree of precision is not the same as high accuracy (this is highly dependent on the integrand function). Due to their simplicity, the Newton-Cotes formulae are popular quadrature choices for scientist who need to approximate an integral, but don't bother to include library routines in their software. However, the Gaussian quadratures dominate the majority of integrator libraries, due to their superior accuracy and stability.

1.1.3 Integrator routines

The term integrator routine or quadrature routine may be applied to any type of numerical software that approximates a definite integral numerically. Such routines range from a simple line of code that implements the midpoint rule, to large, complex software packages that subdivide the integration interval and increase the number of integration points for each interval stepwize, while updating a measure for the accuracy of the calculations. Using the classification of Lyness and Kaganove [20] the mentioned integrator routines are examples of a *rule evaluation quadrature routine* and an *automatic quadrature routine*, respectively.

Rule evaluation quadrature routines apply to quadrature schemes that are computed with a given number of integration points. A general algorithm for a rule evaluation quadrature routine, with known abscissae and weights, is given by Alg. 1. The input values a, b, f and n are the left and right limit of integration, the integrand function and the number of integration points, respectively. The tables w and x provide tabulated values for the weights and abscissae, respectively, and I_n is the approximated integral.

Algorithm 1 Rule evaluation quadrature routine

```
function REQR(a, b, f, n)
    I_n ← 0
    for i ← 1, n do
        I_n ← I_n + w(i) * f(x(i))
    end for
    return I_n
end function
```

A simple integrator routine on the form of Alg. 1 will carry out all n iterations regardless of the form of the integrand function, $f(x)$. This makes the efficiency of the integrator

routine constant, as efficiency in a quadrature routine usually is measured in terms of the number of function evaluations used by the routine. This does not necessarily mean that the routine has a constant *computational cost*. That is, the required computational time used to perform the numerical integration (more precisely, the number of CPU clock cycles to perform the call and return-branch involving the integrator routine) may vary depending on the integrand function that is evaluated. Usually, however, the efficiency and computational time of an integrator routine is closely related, making a rule evaluation quadrature routine predictable with regards to computational cost. This predictability in one of the main advantage of such a simple integrator routine, compared to more sophisticated integrators. However, since the rule evaluation quadrature routine performs the same operations regardless of $f(x)$ being a linear or a highly oscillatory function, and since the user does not know which value of n to use such a routine may

a) waste computational resources for integrand functions that require few points to be approximated accurately, or
b) compute the quadrature sum with too few integration points to get an accurate result, for an integral that requires more integration points to be approximated accurately.

In addition, no measure of the accuracy of the computed result is returned; the routine does not make any effort in order to obtain such a measure.

Automatic quadrature routines overcome these difficulties—to some extent—by requesting an accuracy requirement from the user, rather than the number of integration points to use in the computations. The accuracy requirement is used in order to return the computed result when one of two *termination criteria* are met. Either a convergence criteria is validated as true or a maximum number of integration points is reached (failure to converge). A widely used convergence criteria, introduced by de Boor [9, 10], ends the computation when

$$|I_n - I| \leq \max\{\varepsilon_{abs}, \varepsilon_{rel} I\}, \tag{1.6}$$

where ε_{abs} and ε_{rel} are the requested absolute and relative accuracy, respectively. Hence, in an automatic quadrature routine the number of integration points used in the computation of a sufficiently accurate result is determined at run-time.

Unfortunately, introducing the accuracy requirement also introduces quite a few challenges in the integrator routine. The quadrature error, that is, the left hand side in Eq. (1.6) can only be approximated, as the exact integral result I is unknown (the opposite would rule out the need for a numerical integrator routine). A convergence criteria that may be implemented in practice is

$$|I_n - I_m| \leq \max\{\varepsilon_{abs}, \varepsilon_{rel} I_n\}, \tag{1.7}$$

where I_m is an approximation of I. I_m may be computed with the same quadrature scheme as I_n, for $n \neq m$, or with a different quadrature scheme, such that two quadratures schemes are employed in the same integrator routine. The left hand side of Eq. (1.7) is called the integrator routine's *error estimate*. As this estimate is calculated with approximations of the integral I, the final result—returned when the convergence criteria is validated as *true*—is not guaranteed to be withing the user's requested accuracy. This introduces a third criteria of performance, in addition to efficiency and accuracy, for an automatic quadrature routine: The reliability.

Further, as the convergence criteria is evaluated at run-time it introduces the need for a more sophisticated algorithm than Alg. 1, with increased computational time being spent outside the integrand function evaluation and summation. This, in addition to

the surcharge related to the need for a quadrature error estimate[1] (not unusual with a surcharge factor of 3 to 4 [19]) leads to a substantially more costly algorithm than the rule evaluation quadrature routine that computes a result with the same accuracy.

Even though the automatic integrator routine cannot guarantee a correct result, within the requested accuracy, well documented automatic integrators are often specified to work well for certain classes of integrand functions. The increased computational cost of an automatic quadrature routine may save the user of such a routine many hours of testing for an optimal number of integration points. As stated by Davis and Rabinowitz [8]: "The aim of an automatic integration scheme is to relive the person who has to compute an integral of any need to think".

The outline of an algorithm for automatic integration that utilizes the convergence criteria in Eq. (1.7), with known weights and abscissae, is given in Alg. 2. Each time a loop-iteration of the while-loop is performed the number of integration points n is increased (drawn from a table, N, of increasing integers). Each iteration is called a *level of integration*. As I_m represents the approximated quadrature on the previous level of integration, this algorithm is an automatic integrator where a single quadrature scheme is utilized. It can be seen that the *for*-loop executed at each level of integration is identical to the *for*-loop executed in the rule evaluation quadrature routine of Alg. 1. This is known as the rule evaluation part of the algorithm, while the part where the termination criteria are evaluated is known as the strategic section of the code [20].

Algorithm 2 Automatic quadrature routine

function AQR$(a, b, f, \varepsilon_{abs}, \varepsilon_{rel}, n_{max})$
  $I_n \leftarrow w(1) * f(x(1))$
  $I_m = R$ ▷ Some number $R \neq I_n$
  $j = 1$
  $n \leftarrow N(1)$
  while $(|I_n - I_m| > \max\{\varepsilon_{abs}, \varepsilon_{rel} I_n\}$ **and** $n < n_{max})$ **do**
    $I_m \leftarrow I_n$
    $I_n \leftarrow 0$
    for $i \leftarrow 1, n$ **do**
      $I_n \leftarrow I_n + w(i) * f(x(i))$
    end for
    $j = j + 1$
    $n \leftarrow N(j)$ ▷ $N(j) < N(j+1)$
  end while
  return I_n
end function

1.1.4 Automatic integrator routine classification

A more sophisticated, and widely used, variant of the automatic integrator in Alg. 2 is the *adaptive* quadrature routine. An automatic integrator is called adaptive if the points at which the integrand is evaluated are chosen in a way that depends on the nature of the integrand function itself [8]. In practice, this means that the position of the integration points at the Nth level of integration depend on the information gathered from the previous

[1] The error estimate is argued to be a good estimate for the less accurate of the approximations in Eq. (1.7), yet it is applied as an overly cautious error estimate for the more accurate of the approximations [17].

integration levels [28]. This typically involves dividing the integration interval into smaller intervals—and some of these into even smaller intervals—until the aggregated error of the approximation of the integral I on all intervals is within a specified accuracy, or the maximum number of subintervals is reached. [12].

The integrator in Alg. 2 is a *non-adaptive, iterative* integrator. In an iterative scheme, successive approximations of the integral are computed until the result is in agreement with the requested accuracy. In contrast, in a *non-iterative* scheme an initial approximation is computed and provide information to a second approximation, which is taken as the final answer [8]. Adaptive integrators typically use non-iterative quadrature schemes in order to compute two approximations of the integral on each subinterval, allowing them to get a measure of error in the approximation needed to determine whether the routine should converge with the current interval set-up.

A major disadvantages of using Gaussian quadratures in automatic integrator routines is that the weights and abscissae of one integration level are distinct from those of all other integration levels (except that the midpoint appears as an abscissa in all levels with an odd number of integration points). Thus, in proceeding from a computation of I_m to I_n, with $n > m$, almost all the information obtained in computing the result at the previous level of integration is lost [8]. To overcome this disadvantage Kronrod [16] developed a technique to augment an n-point Gauss-Legendre quadrature with $n + 1$ integration points—a so called optimal extension of an n-point Gaussian quadrature. By first finding a numerical approximation of the integral for n points, all function evaluations and abscissae can be reused when computing the quadrature with $2n + 1$ points. Such a quadrature scheme is known as a *nested* quadrature scheme, and is common in non-iterative, adaptive integrators.

By generalizing of the Kronrod rules, Patterson [26] constructed a sequence of nested formulae suitable for iterative, non-adaptive integrators. He started with a 3-point Gauss-Legendre rule and added further rules by the optimal extension of the respective predecessor. Hence, with the first level having three integration points, the following levels will have 7, 15, 31, 63... points, where all function evaluations and abscissae from one level are reused in all the following levels. Quadrature schemes of this types are known as *Gauss-Patterson* quadratures and are claimed to be highly efficient compared to other iterative methods [27].

The nested nature of the Gauss-Kronrod and Gauss-Patterson quadratures comes at a price. The degree of precision is reduced compared to the Gauss-Legendre rules of an equal amount of integration points. The sequence of nested levels starting at a 3-point Gauss-Legendre rule with $d = 5$, has a second level that is a 7-point Kronrod-extension with $d = 11$, and levels following this that are Patterson-extensions with $d = (3n+1)/2$ [8].

1.1.5 Errors in quadrature routines

In numerical integration, in general, two types of error occur. The first error, and possibly the most obvious one, is the *truncation error*, E_t. The truncation error arises from the fact that the sum in Eq. (1.2) is only approximately equal to the integral in Eq. (1.1), for a finite number of integration points, n:

$$\int_a^b f(x)\mathrm{d}x = \sum_{i=1}^{n} w_i f(x_i) + E_t. \tag{1.8}$$

The second error is the *round-off error*, E_r, which arises from the fact that the sum in Eq. (1.2) is affected by to the limitations in accuracy of the computer[2]. Hence, the

[2]These limitations are known as floating point precision limitations.

computed sum, S_{comp}, is:

$$S_{comp} = \sum_{i=1}^{n} w_i f(x_i) + E_r. \tag{1.9}$$

The limitation in the computed sum being an approximation of the actual sum is not only valid for numerical quadratures, but for all summation performed by a computer. In practice, round-off errors are usually negligible. However, with a lot of integration points and high accuracy requirements the round-off errors may be taken out of the negligible category [8].

When utilizing quadrature schemes that involve irrational numbers for weights and abscissae (as is the case for most Gaussian quadratures) additional errors arise from the cut-off point in the numerical representation of these numbers on a computer. As these errors arise from the floating point precision limitations, they usually do not represent significant errors in the computed approximation. However, as the abscissae are used in the computation of the integrand function values, functions where they do have significant impact may be constructed. These errors are avoided by increasing the floating point precision in the computation (increasing the computational cost of the integrator). If errors in the integrand function values do occur, they may have noticeable impact on the computation, especially for automatic integrators where such errors may wreck the efficiency of the routine [18].

Automatic integrator routines are not only susceptible to efficiency issues. The limited reliability of the convergence criteria described in Sec. 1.1.3 give rise to erroneous behavior with respect to the accuracy of the result. As mentioned, as both I_m and I_n in Eq. (1.7) are approximations of the integral at hand, a returned result for a converged automatic integrator is not guaranteed to be within the users requested accuracy. This may be because the quadrature scheme utilized in the integrator is not sufficiently accurate for the integral computed, or it may be because the integrator converges at a too low level of integration (when the correct result could have been obtained at a higher level of integration). The first of these sources of error may be avoided by only using the automatic integrator on classes of integrand function for which the quadrature has been thoroughly tested and regarded as an appropriate routine. The second source of error is harder to avoid, and harder to detect. It it is often the result of minor changes in the choice of integrand function, and *parametric studies* are useful for evaluating the reliability of the automatic integrator, in this respect (see Chap. 2).

It is possible to make the convergence criteria stricter, thus making the automatic integrator routine more reliable. This will, however, affect the efficiency of the routine, and is not recommended unless deemed necessary after rigorous testing.

It is important to be aware of the fact that automatic quadrature should not be expected to be completely reliable. This is often overlooked by novice users of quadrature routines, and under-communicated in literature. As an integrator routine only evaluates the integrand in a finite number of selected points on the interval of integration, the routine has no information about the function outside these points. Function behavior off these points is simply missed unless it is built into the assumptions about the program [28]. It is not possible to prevent the misuse of an integrator routine by giving it inappropriate input information, e.g., specially constructed integrand functions with sharp peaks between the abscissae, or zeros at all abscissae.

1.2 Algorithm development and testing

1.2.1 Quadrature scheme and routine choices

Many state of the art automatic integrator routines utilize adaptive quadrature schemes. These are proven to be both efficient, accurate and reliable, for a large number of challenging integrand functions (see, e.g., Ref. [12, 13, 28]). The main motivation for developing this integrator is, however, not to make an integrator routine with the best all-over performance, nor the most accurate results for very hard integrands. The motivation is to improve the calculations of a large number of integrals, computed in the physics simulation software Maxwell1D. The integrals have previously been computed with a midpoint rule, hence, the computational cost will increase when an alternative quadrature routine is utilized. The increased accuracy of the approximated integrals, when replacing the midpoint rule by a more accurate quadrature scheme, is, however, hopefully substantial without wrecking the efficiency of the simulations.

The quadrature scheme used in the developed automatic integrator routine is a Gauss-Patterson quadrature. The integrator routine is a non-adaptive, iterative automatic integrator with a nested nature. It has a total of eight nested levels, with the 3-point Gauss-Legendre rule as level one. Hence, levels one to eight have 3, 7, 15, 31, 63, 127, 255 and 511 points, respectively. In addition, a zeroth level of integration, with a single integration point, is included, to enable calculating an error estimate at level one—enabling convergence with as few as three integration points. The routine differs from the automatic quadrature routine in Alg. 2 in the respect that it reuses the abscissae, the function evaluations and the integral approximation of each integration level, when proceeding to the next level of integration. A previous implementation[3] of this quadrature scheme has shown promising results, regarding efficiency for smooth and oscillatory, non-singular integrand functions [27].

The convergence criteria in use is the one defined in Eq. (1.7), with I_n and I_m being the results of the current and the previous level of integration, respectively. This is a rather unsophisticated termination criteria, that requires rigorous testing before being accepted as sufficiently reliable. The automatic integrator routine returns an approximation of integral if the convergence criteria is met, or if no satisfactory result is computed after completing the 511-point level of integration (level eight).

Neither the abscissae nor the weights are calculated in the module, they are all hard-coded. The tabulated values have been found at [6]. The interval for which the values are given is an open interval: $\langle -1.0, 1.0 \rangle$; the integration limits given by the user of the routine are re-scaled by Eq. (1.5). Consider Fig. 1.1, where the abscissae for levels zero to four are plotted. The figure illustrates the nested nature of the integration points, i.e., the re-use of abscissa points from one integration level to all subsequent levels. The nested nature is valid for all levels in the integrator routine.

Due to allocation and de-allocation of memory being a costly process, the entire arrays of abscissae and weights are allocated at the routine start-up. The arrays are build in a counter-intuitive way, such that the elements reads and writes are done in a contiguous fashion, to minimize performance issues due to CPU cache-misses.

The complete main subroutine of the integrator module is included in Appx. A.1.

[3] This implementation allows for adaptive and non-adaptive interval subdivision. It is, however, described as sufficiently powerful to rarely invoke subdivision, such that the it might be regarded more as a emergency procedure for very hard integrals.

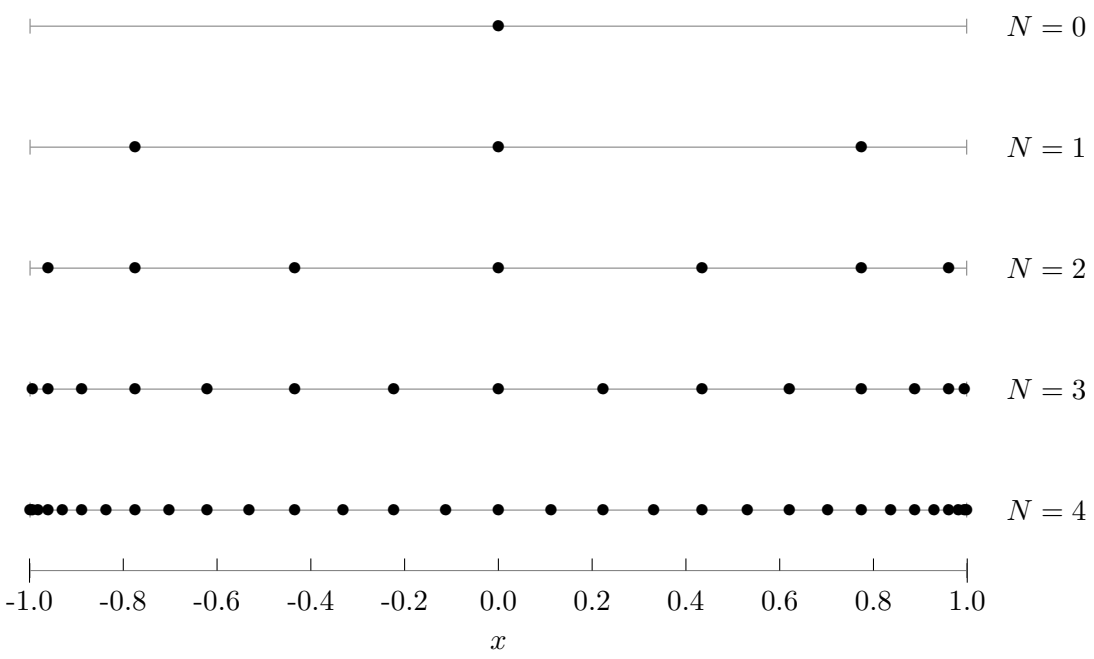

Figure 1.1: Nested nature of the abscissae used by the automatic integrator GPat. Levels zero ($N = 0$) to four ($N = 4$) included in the figure.

1.2.2 Usage

The implementation is written in Fortran90 and compiled with the GNU Fortran compiler, gfortran version 4.7.3, on a Linux 64-bit PC. All necessary functions and subroutines for the computation of integrals with real integrand functions are implemented in a module named *integrator_module*, in the file *integrator.f90*. The only external module used is a precision module modeled after the *nr-type* in [29], included at the start of the module by the statement:

```
use SFL_Precision, only : sp, dp, wp=>dp
```

where *sp* and *dp* refer to single precision and double precision, respectively. The *wp* variable is used as an operator throughout the program, to allow the user to switch between single and double precision, simply by having *wp* point to the appropriate precision parameter (double precision in the line of code included).

The name of the subroutine performing the numerical integration is *GPat*, named such to reflect the use of the Gauss-Patterson quadrature scheme. A call to GPat requires an integrand function, a derived data type *integrator_info*, the integration limits, and a result variable, as parameters. Hence, when calling the integrator from a separate source file, the following line must be included (in addition to the *SFL_Precision* statement):

```
use integrator_module, only: GPat, integrator_info
```

The *only* statement is optional, but highly recommended to avoid conflicts of function and subroutine names throughout the program.

The derived data type *integrator_info* contains information about the requested accuracies, convergence and the estimated error:

```
type :: integrator_info
  real(wp)    :: abs_accuracy
  real(wp)    :: rel_accuracy
  logical     :: converged
  integer     :: n_evals
```

```
    real(wp)   :: est_error
  end type integrator_info
```

The parameters *abs_accuracy* and *rel_accuracy* are the requested absolute accuracy and the requested relative accuracy, respectively. These two parameters must be initialized to numerical values before passing the derived data type to GPat; they are constant throughout the numerical integration. At least one of the parameters *abs_accuracy* and *rel_accuracy* must be non-zero, to allow the integrator to converge. The parameter *converged* is a logical data type confirming convergence if returned as *true*, *n_evals* is the number of function evaluations used to reach convergence[4], and *est_error* is the estimated error at routine exit. These three parameters are set at run-time. If convergence is not reached, *converged* is returned as false, *n_evals* and *est_error* are returned as the number of function evaluations and the estimated error at the highest level of integration.

In addition to *integrator_module*, a module that is capable of computing integrals with complex integrand functions has been developed. The module *integrator_module_cplx* is similar to *integrator_module*, but differs in some respects:

- The routine GPat, the derived data type *integrator_info*, the module and the file containing the module are named with a suffix *_cplx*, to separate them from the automatic integrator applicable for real integrals.
- The *est_error* parameter in the information data type is declared as a complex data type, *complex(wp)*, such that the error estimate for both the real and the imaginary part of the result is returned.
- The input function and the output result are complex data types.
- When checking for convergence the real and the imaginary part of the result are tested independently. Both parts must yield estimated errors within the accuracy requirements for the routine to converge.

When calling the integrator from a separate source file, to numerically integrate a complex function, the following line must be included, in addition to the *SFL_Precision* statement:

```
use integrator_module_cplx, only: GPat_cplx, integrator_info_cplx
```

Again, the *only* statement is optional, but recommended.

1.2.3 Comments on general-purpose testing

The test performed in this chapter are performed with the purpose of testing the automatic integrator routine with regards to all-over performance, that is, how it performs as a general-purpose integrator. Later on the routine will be modified to fit the integrals in the Maxwell Eqs. Solver, and will no longer be classified as a general-purpose integrator. Such a customization is, however, merely one possible use of the integrator routine. The general-purpose testing will serve as a foundation for the expected strengths and weaknesses of the integrator routine, revealing classes of integrand function where it may prove efficient and reliable, thus, making it a integrator routine applicable for numerical integration beyond the use in this thesis.

There is no general consensus in literature on a single best method of testing integrator routines. There exists a wide variety of test, that are generally classified as either battery experiments or parametric studies [11, 21, 30]. Battery tests typically consist of performing several integrations on a group of functions with widespread characteristics. These

[4]This equals the number of integration point at the level of convergence, for a fully nested quadrature scheme.

typically include functions that are well-behaved, have sharp peaks, have singularities at the endpoints or within the integration interval, are highly oscillating, or have discontinuities. In parametric studies an integrator is tested for problem families, that is, classes of integrand functions whose members differ only in a parameter λ. Once a problem family is chosen, as many members of this family as possible are included in the tests. While the results for battery test usually specify how accurate, and at what cost, the result for each of the different types functions are computed, it is common in many parametric studies to derive a score or a statistical distribution function to describe the performance of the routine for a specified problem family.

A drawback of battery testing is that as only a few elements of the same class of integrand functions are tested, there possibly exist functions very close those tested that will be computed quite differently by the integrator routine. Battery testing alone may therefore yield inconclusive results. However, as such spot tests are relatively cheap to compute, many different integrals may be considered. This gives important pointers to where the integrand routine at hand performs well, and—perhaps more importantly—where it is inefficient, inaccurate or unreliable.

Parametric studies, on the other hand, yield a more complete result for the specific problem family that is tested. Such test are, however, much more expensive than the battery tests, and therefore not optimal for initial, general-purpose testing of an integrator routine where a wide range of integrand functions are tested. For this reason only battery tests are provided in this chapter. Parametric studies will be used when testing for the specific integrals that constitute the motivation for developing this integrator. More on this in Chaps. 2.

1.2.4 Battery tests

The test performed are:

- Checking for consistency of tabulated weights.
- Testing if, and at what integration level, the convergence criteria is met.
- Comparing the actual error of the numerical result to the requested accuracy.
- Comparing the efficiency of the integrator routine, at numerous accuracy requirements, to equivalent computations done with library routines.

With the exception of checking for consistency of tabulated weights, all the listed tests are battery experiments. In these experiments the automatic integrator routine is tested for the integrals in Tab. 1.1, for different accuracy requirements. When testing for actual error, only integrals 1–12 are used, as the closed form solutions do not exist for integrals 13 and 14. All tests in this chapter are performed with the floating point precision set to double.

None of the test integrals in Tab. 1.1 contain singularities within the limits of integration. One of the integrals, integral 11, does have a singularity at the endpoint. This is allowed, since the Gauss-Patterson quadrature is an open quadrature formula, but is expected to be a challenging integral to compute with GPat.

When comparing actual error to the requested accuracy it is important be aware of the fact that even though the analytic results are exact, they will suffer from limited precision when implemented on a computer (as all floating point representations of real numbers do). Hence, an analytic result implemented with double precision is no longer the exact mathematical result. In practice, this means that the approximated integrals cannot have an accuracy beyond the floating point precision in use, even if the difference between the double precision analytic result and the computed integral is zero.

Table 1.1: Test integrals of different complexity.

#	Integral
1	$\int_0^{\pi} \sin(x)\mathrm{d}x$
2	$\int_0^{\pi} \sin^2(x)\mathrm{d}x$
3	$\int_{-1}^{1} \sin^2(x)\cos^3(15x)\mathrm{d}x$
4	$\int_{-1}^{1} e^x\mathrm{d}x$
5	$\int_{-1}^{1} e^{-x}\mathrm{d}x$
6	$\int_0^{3} xe^{-x^2}\mathrm{d}x$
7	$\int_0^{1.01} (71x^{178} - 0.5x^{39} + 1.2x^7)\mathrm{d}x$
8	$\int_0^{1} \frac{x}{x^4+1}\mathrm{d}x$
9	$\int_0^{1} \sqrt{x}\mathrm{d}x$
10	$\int_{10^{-4}}^{1} \sqrt{x}\mathrm{d}x$
11	$\int_0^{1} \log(x)\mathrm{d}x$
12	$\int_{10^{-4}}^{1} \log(x)\mathrm{d}x$
13	$\int_0^{1} \frac{\sin(100\pi x)}{100\pi x}\mathrm{d}x$
14	$\int_0^{1} \left(\frac{\sin(10\pi x)}{10\pi x}\right)^5 \mathrm{d}x$

As mentioned, the efficiency of a quadrature routine is usually measured in terms of the number of function evaluations used by the routine. Such a measure has certain limitations, as it does not take into account the computational cost of extra overhead due to memory allocation, convergence checking or possible integration interval subdivision. However, such parts of the integrator routine may be optimized at a later time, but the number of function evaluations needed for the routine to converge remain the same unless the quadrature scheme is fundamentally modified or replaced by another scheme. The number of function evaluations needed by an integrator routine to converge therefore serves well as a measure of computational cost when comparing different automatic integrator routines. For such comparison, three of the automatic integrator routines from the FORTRAN77 library QUADPACK are employed. The QUADPACK routines used are [28]:

- DQAG: A double precision globally adaptive integrator. Allows for a choice between six pairs of Gauss-Kronrod quadrature formulae for the rule evaluation component.

Table 1.2: Double precision sum of all weights at each level of integration in the automatic integrator GPat. The interval of integration is $[-1, 1]$.

Level	Sum of weights
0	2.0000000000000000
1	2.0000000000000000
2	2.0000000000000000
3	2.0000000000000000
4	2.0000000000000000
5	2.0000000000000000
6	2.0000000000000004
7	2.0000000000000000
8	2.0000000000000027

Three of these are used during testing. The routine is recommended for use if the integrand function is smooth, or if it is of non-specific oscillatory type, without singularities.

- DQAGS: A double precision general-purpose integrator based on globally adaptive interval subdivision in connection with extrapolation. The routine is recommended for use if the integrand function has endpoint singularities.
- DQNG: A double precision non-adaptive integrator, using a nested Patterson scheme, similar to the one used by the GPat routine. The routine has four levels of integration, with integration points 10, 21, 43 and 87, for levels one to four, respectively. It is recommended for use if the integrand function is smooth.

1.3 Initial results

1.3.1 Consistency of tabulated weights

The sum of all weights at each level of integration is presented in Tab. 1.2. It can be seen that the results differ somewhat, in the least significant digits. The variations are present in the upper levels of integration.

1.3.2 Convergence and reliability

The integrator's levels of integration where convergence is reached, for the definite integrals in Tab. 1.1, are presented in Tab. 1.3[5]. The results are given for requested relative accuracy set to 10^{-5}, 10^{-10} and 10^{-15}, in turn, and requested absolute accuracy set to zero. An integral that does not reach convergence for at a certain accuracy is represented by the letter F—for failure to converge—in stead of an integer representing the level at convergence. Recall that the number of function evaluations to reach convergence at level one to eight is 3, 7, 15, 31, 63, 127, 255 and 511, respectively. Convergence results for more requested accuracies within the range $\varepsilon_{rel} = 10^{-1}$–$10^{-15}$ are included Appx. B.1.

As the analytic results are known for integrals 1–12 in Tab. 1.1, the results of the numerical integration may be tested for actual accuracy. This makes these integrals well-suited for use in studying the reliability and efficiency of the automatic integrator routine's error estimate. Figures 1.2–1.6 depict the actual relative error of the numerical results

[5]Table 1.3 includes the level of integration only, to increase the readability. When comparing GPat to other integrator routines, it is more appropriate to consider the number of function evaluations directly. This is how the results are presented in Sec. 1.3.3.

Table 1.3: The number of function evaluations to reach convergence, for the automatic integrator GPat, for requested relative accuracies 10^{-5}, 10^{-10} and 10^{-15}. An integration level marked by F denotes failure to converge at the requested accuracy. All runs are with double precision.

	$\varepsilon_{rel} = 10^{-5}$	$\varepsilon_{rel} = 10^{-10}$	$\varepsilon_{rel} = 10^{-15}$
Integral	Integration level		
1	3	4	4
2	3	4	4
3	6	6	8
4	3	3	4
5	3	3	4
6	3	4	4
7	5	6	8
8	3	4	5
9	5	8	F
10	4	7	8
11	7	F	F
12	6	8	8
13	7	7	F
14	5	6	7

computed with the automatic integrator, plotted against the requested relative accuracy used as input. Each figure contains several of the integrals from Tab. 1.1, grouped by the nature of the integrand functions:

- Integrals 1–3 in Fig. 1.2
- Integrals 4–6 in Fig. 1.3
- Integrals 7 and 8 in Fig. 1.4
- Integrals 9 and 10 in Fig. 1.5
- Integrals 11 and 12 in Fig. 1.6

The results have been computed by 800 runs of the integrator GPat, for each integral, with requested relative accuracy being incremented with a factor of 0.95 for each iteration. For runs with numerical results equal to the analytic result, the actual relative error is of the order of the machine precision. To be able to display these points in the figure, actual relative error is set to 10^{-17}, that is, below the theoretical best relative accuracy achieved for non-exact result[6]. In each figure a dashed line is included, to separate the two regions of the figure: The region where the actual relative error is larger than the requested relative accuracy (upper region), and the region where the actual relative error is the smallest of the two (lower region). The curves in Figs. 1.2–1.6 are augmented with points (filled circles), where the points on each curve in the same figure have different offsets. The purpose of this is to increase readability for the regions where two or more curves overlap.

1.3.3 Efficiency

The number of function evaluations needed to reach convergence for the definite integrals in Tab. 1.1, using the automatic integrator routines DQNG, DQAG, DQAGS and GPat,

[6]This constant is known as machine epsilon, which is $1.19 \cdot 10^{-7}$ for single precision and $2.22 \cdot 10^{-16}$ for double precision in Fortran90 at the computer where the experiments are conducted.

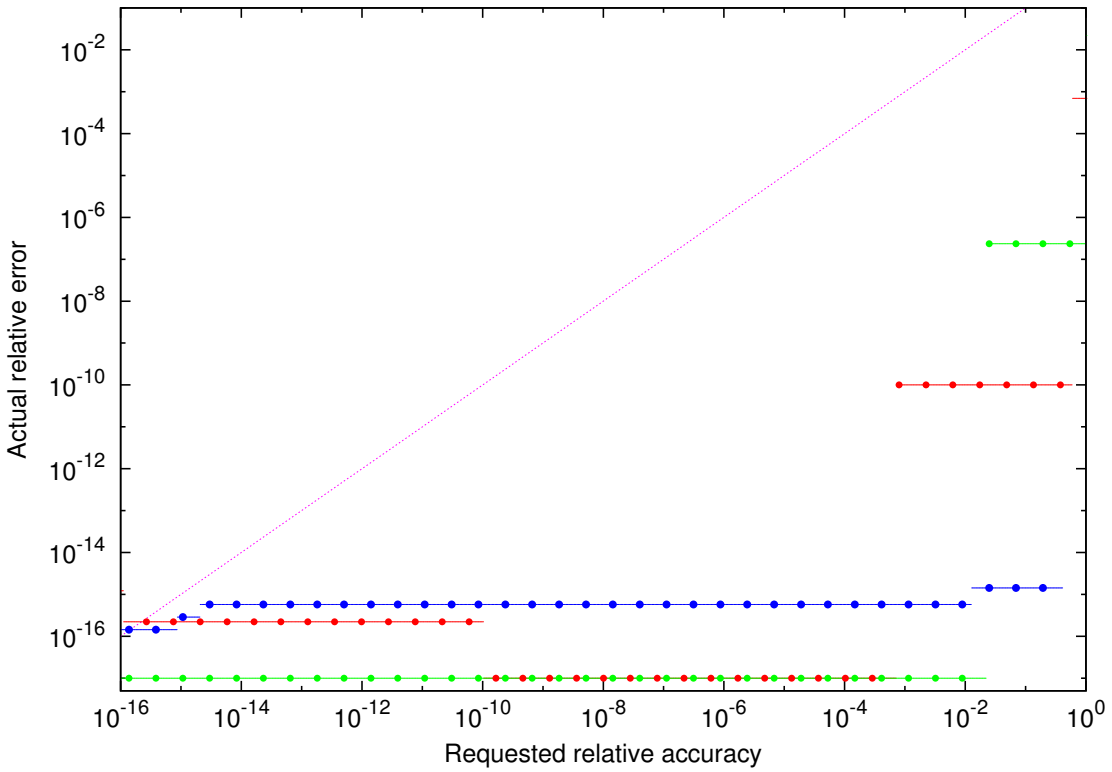

Figure 1.2: Actual relative error plotted against requested relative accuracy, for integrals 1 (red), 2 (green) and 3 (blue) from Tab. 1.1. The dashed pink line separate the region of the figure where the actual relative error is larger than the requested relative accuracy (upper region), from the region where the actual relative error is smaller than the requested relative accuracy (lower region). For numerical results equal to the analytic result the error is set to 10^{-17}, such that these values are visible within the range of the y-axis. To be able to view overlapping curves, the curves are augmented with filled circles with different offsets.

with a requested relative accuracy $\varepsilon_{rel} = 10^{-5}$, are presented in Tab. 1.4. Integration with DQAG has been performed with input value for the parameter *key* set to 1, 3 and 6, in turn. This corresponds to subintervals with local, non-iterative integration rules of Gauss-Kronrod pairs with 7–15 points, 15–31 points, and 30–61 points, respectively. The corresponding results for requested relative accuracies 10^{-10}, and 10^{-15} are included in Appx. B.2.

In order to get a better understanding of the computational performance of each of the automatic integrators, for different requested relative accuracies, averaged results are presented in Fig. 1.7. The plotted values give an average measure of the efficiency for the integrator routines for all integrals by

$$p_j(\varepsilon_{rel}) = \frac{1}{k_j} \sum_{i=1}^{k_j} \frac{n_{i,j}}{\bar{n}_i}, \tag{1.10}$$

where k_j is the number of integrals the integrator routine j converges for, at a requested relative accuracy ε_{rel}, $n_{i,j}$ is the number of function evaluations needed to reach convergence for integral i by integrator j and $\bar{n}_i$ is the average number of function evaluations used by all integrator routines that converged for integral i (both for the requested accuracy ε_{rel}).

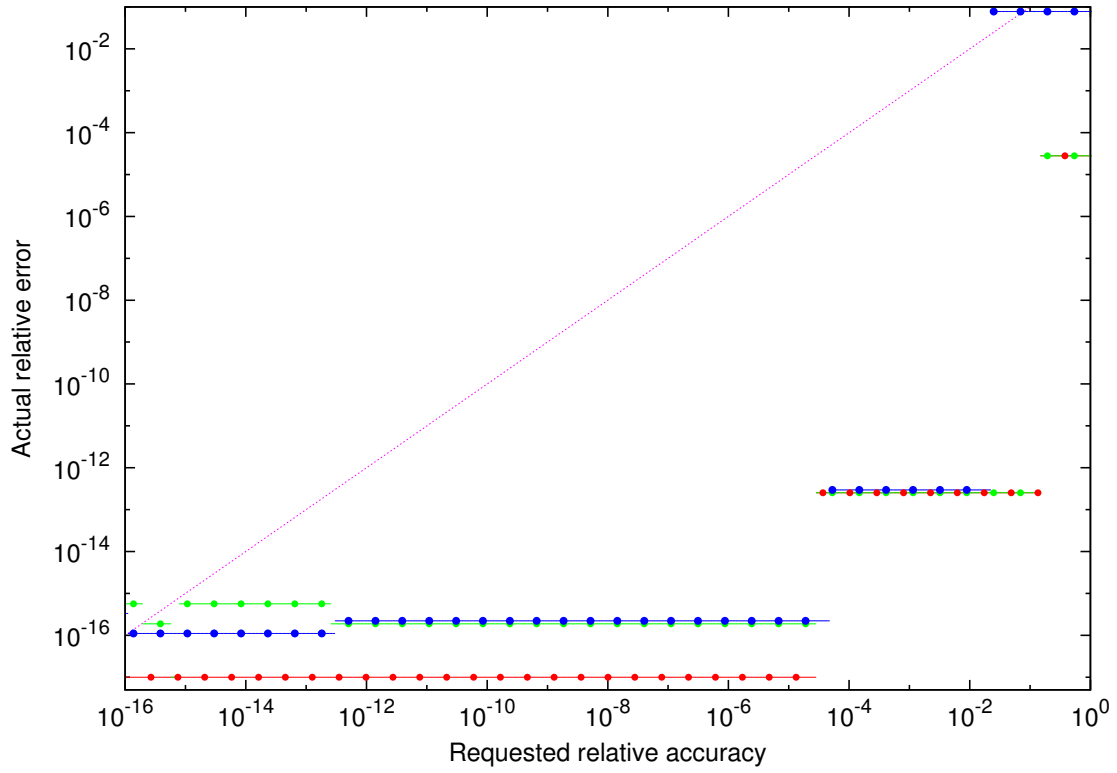

Figure 1.3: Same as Fig. 1.2, but for integrals 4 (red), 5 (green) and 6 (blue) in Tab. 1.1.

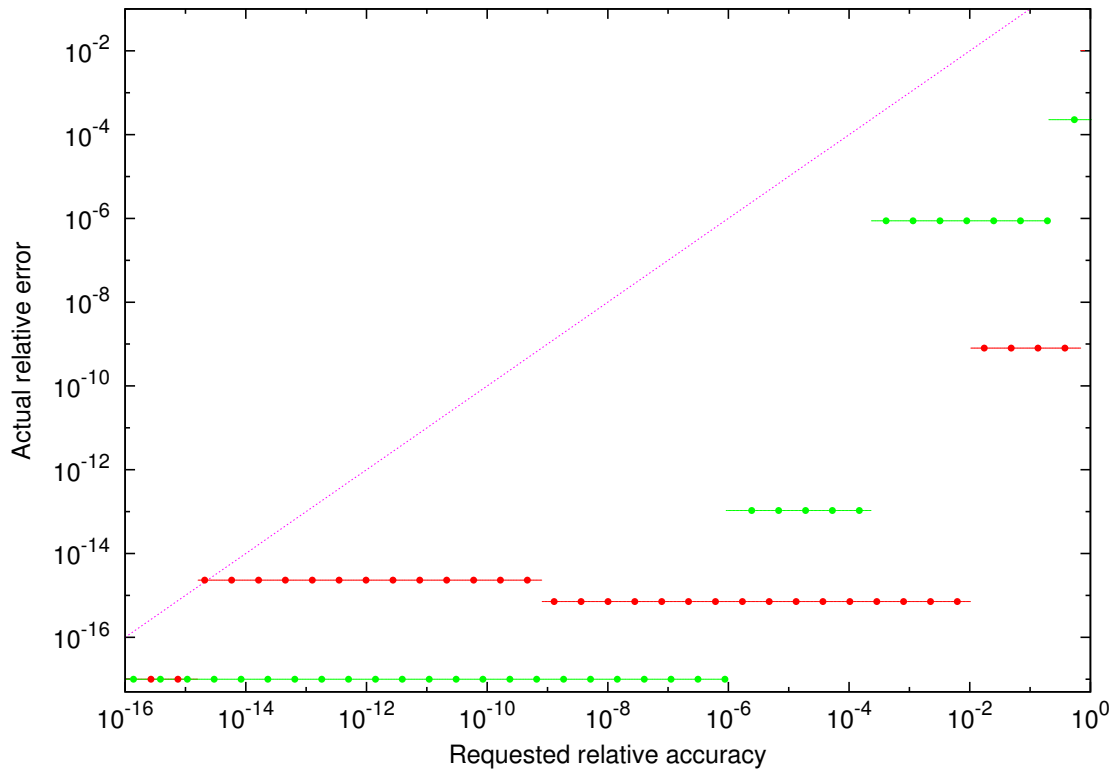

Figure 1.4: Same as Fig. 1.2, but for integrals 7 (red) and 8 (green) in Tab. 1.1.

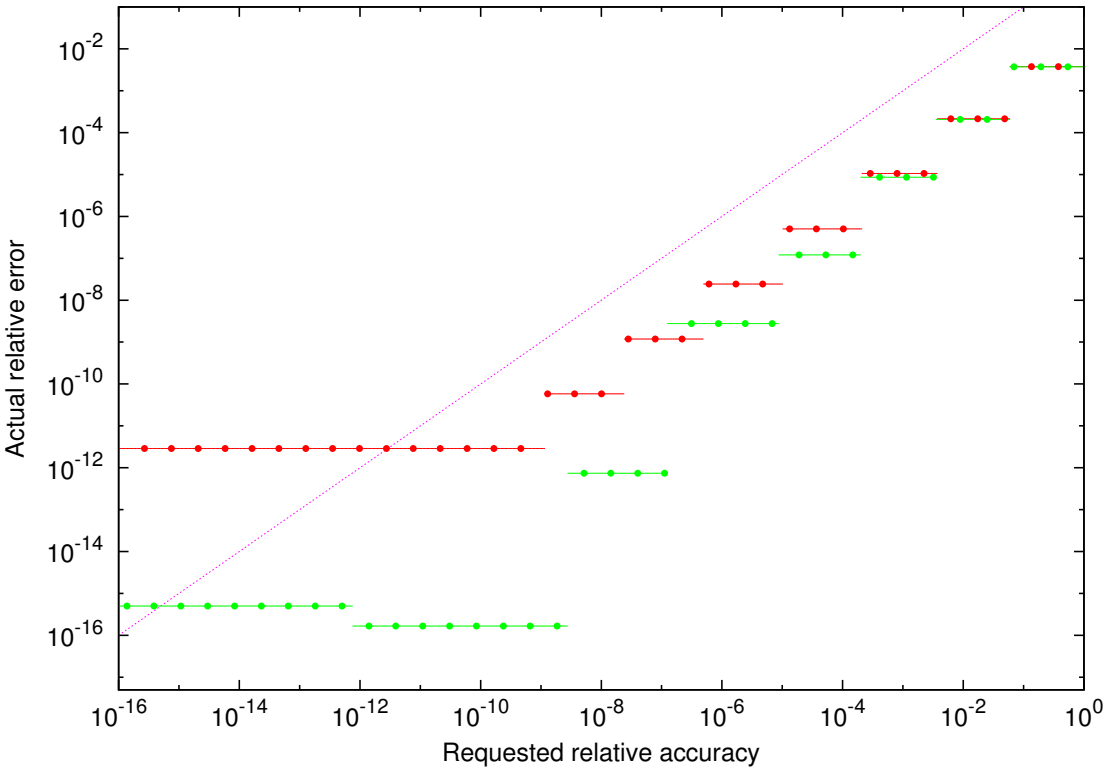

Figure 1.5: Same as Fig. 1.2, but for integrals 9 (red) and 10 (green) in Tab. 1.1.

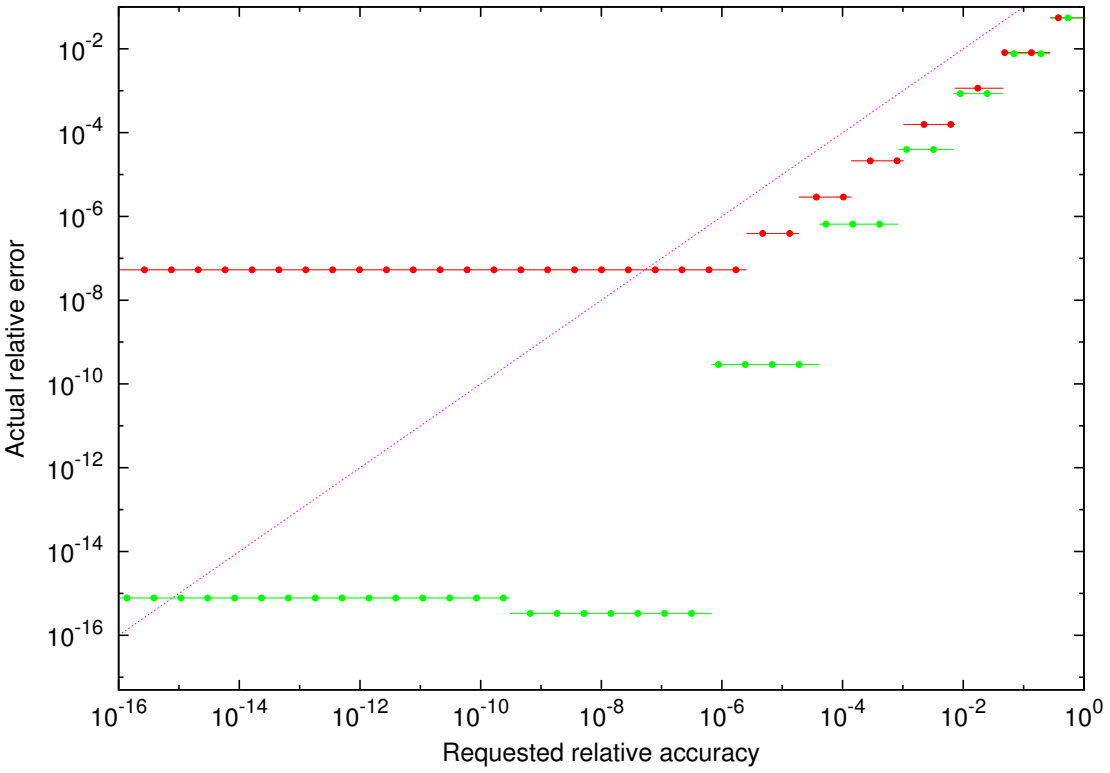

Figure 1.6: Same as Fig. 1.2, but for integrals 11 (red) and 12 (green) in Tab. 1.1.

Table 1.4: The number of function evaluations needed to reach convergence for the integrals in Tab. 1.1, with four different automatic integrator routines. DQNG, DQAG and DQAGS are QUADPACK library routines, and GPat is the integrator implemented here. Subscripts 1, 3 and 6 on the DQAG routine denote the value of a parameter *key* taken as input by the routine, corresponding to three different integration rules used by this integrator. F denotes a routine that is unable to converge for the requested accuracy. Requested relative accuracy is 10^{-5}.

Integral	$DQAG_1$	$DQAG_3$	$DQAG_6$	DQAGS	DQNG	GPat
1	15	31	61	21	21	15
2	15	31	61	21	21	15
3	465	217	183	315	87	127
4	15	31	61	21	21	15
5	15	31	61	21	21	15
6	45	31	61	21	21	31
7	135	155	61	147	87	63
8	15	31	61	21	21	15
9	255	341	305	231	87	63
10	225	279	305	231	87	31
11	525	1023	1647	231	F	255
12	315	465	671	357	F	127
13	1125	261	427	1113	F	255
14	165	155	61	147	87	63

The results are computed for requested relative accuracy, $\varepsilon_{rel} = 10^{-l}$, where $l = 1, 2, ..., 15$, and requested absolute accuracy set to zero. To increase readability the plotted points are joined by lines. A ratio below one indicates better than average performance at the specific requested accuracy.

1.4 Discussion – first impressions and expectations

The automatic integrator routine GPat has been developed as a general-purpose integrator. At this point, no measures have been taken to specialize the routine for implementation in the Maxwell Eqs. solver, beyond the choice of a nested quadrature scheme. This is due to the expectation that the automatic integrator routine can be applied to a variety of problems where efficient numerical integration is needed.

1.4.1 Illustrative tests

The results from the consistency test are primarily of illustrative value:

The summed weights in Tab. 1.2 are not enough to determine whether the weights are correct or not (the weights are well known, tabulated values, and they are known beforehand to be correct). However, in addition to showing consistency (in summing all weights to 2.0), this test has some pedagogic value. Two of the sums are examples of round-off errors, described in Sec. 1.1.5. The occurrence of the deviation in the result at high levels does not come as a surprise, as round-off errors are most likely to occur when the number of elements in the computed sum are large. The results in Tab. 1.2 are computed adding elements from index zero to n (n being the number of integration points at level N), using a specific counter-intuitive indexing. When summing the weights in an array aligned equivalent to abscissae from left to right the sums for the three highest levels

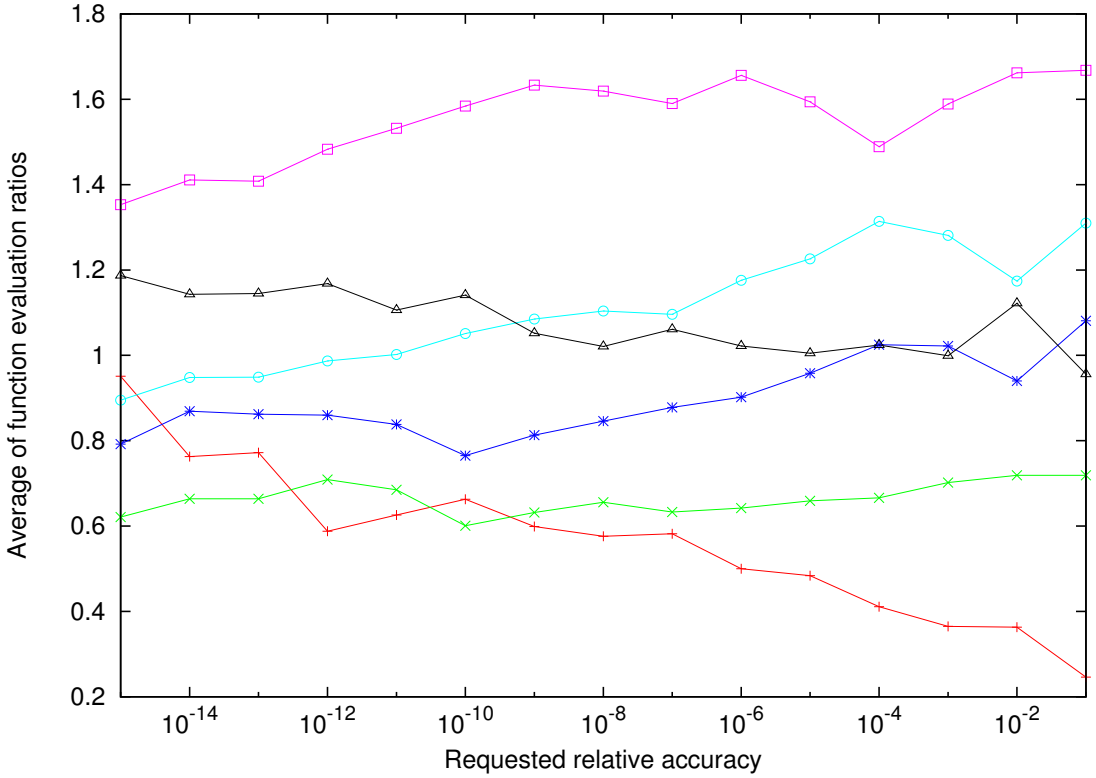

Figure 1.7: Averaged ratios of the integrator routines' efficiency, computed by Eq. (1.10) for the integrals in Tab. 1.1. Results for the automatic integrators GPat (red, +), DQNG (green, x), DQAGS (blue, *), DQAG with $key = 1$ (black, △), DQAG with $key = 3$ (cyan, ○) and DQAG with $key = 6$ (pink, □). The requested absolute accuracy is zero. The computed values are joined by lines to increase readability.

differed from the ones in Tab. 1.2. Neither of the summing conventions are necessarily better than the other, regarding round-off errors.

1.4.2 Automatic integrator routine convergence

The automatic integrator routine converged for all test integrals in Tab. 1.1 at requested relative accuracy $\varepsilon_{rel} = 10^{-5}$, and for most of the integrals at higher accuracy requirements. The three integrals that were too hard for the integrator to compute at requested relative accuracy $\varepsilon_{rel} = 10^{-15}$ where integrals 9, 11 and 13. Both integrals 9 and 11 have integrand functions with derivatives that go to infinity at the left integration limit. In addition, the integrand function in integral 11 has a singularity at this limit. By moving the left limit of integration slightly, from zero to 10^{-4}, the results are significantly improved. This can be seen from convergence of integrals 10 and 12, for all requested relative accuracies. Integral 13 has a highly oscillatory integrand function. The numerical integration of the normalized sinc-function converges up to and including a requested relative accuracy 10^{-14} (see Tab. B.1 and B.2). The failure to converge at $\varepsilon_{rel} = 10^{-15}$ is not surprising, due to the high frequency oscillations making it a hard integral to approximate numerically.

These results may be improved, that is, the integrator may strive for convergence for all integrals at high accuracy requirements, by implementing interval subdivision or by increasing the number of nested integration levels, in the automatic integrator routine. Interval subdivision can be done by either a non-adaptive or an adaptive approach. Non-

adaptive, by simply splitting the interval in two or more parts, and calculating a numerical approximation by an equal amount of integration points for all intervals (indexing these calculations as higher levels than those previously calculated). Adaptive, by focusing the computational power on the subintervals that have the worst error estimates, while the results from the converged subintervals simply are idle until all subinterval computations reach an accuracy requirement. Both strategies would lead to a significant loss of information when going from the highest level of integration with one interval to the first level of integration on a subdivided abscissa; there is no optimal extension of points allowing for a nested nature when employing interval subdivision. Neither are therefore recommended for the implementation here, as efficiency is a priority. Including more nested levels of integration is not recommended either, at this time. 511 integration points is already a quite a few (and probably more than needed in the Maxwell Eqs. solver), especially when compared to the midpoint rule used in Maxwell1D, that has a single integration point. The integrals that GPat cannot compute should rather be computed with another numerical integrator, that use, e.g., adaptive integration or a more appropriate quadrature scheme.

1.4.3 Error estimate – reliability and efficiency

Figures 1.2–1.6 depict the actual error of the numerical integration for different requested accuracies. The figures are split in two by a dashed line, separating the region where the actual relative error is larger than the requested relative accuracy from where the actual relative error is smaller than the requested relative accuracy. Ideally, the plotted results should be below, yet close to, the line separating the two regions for all requested relative accuracies. Such behavior is not achieved, and—as mentioned in Sec. 1.1.5—should not necessarily be expected by an automatic integrator. If the ideal results were achieved, they would imply that the error estimate was either precise, or overly cautious, for all requested relative accuracies for the tested integrals. In the figures, there are some deviations that need to be addressed. These include deviations from the lower half of the figures (requested accuracy not reached) and deviations where the actual error is much smaller than the accuracy requirement (possibly an overly cautious error estimate).

The most significant deviations from the lower half of the plot occur in Figs. 1.5 and 1.6, for integrals 9 and 11, respectively. Comparing these results to the convergence results in Tabs. 1.3 and B.1–B.2 reveals that actual relative error is only larger than the requested relative accuracy in regions where the integrator does not converge. Furthermore, the points where the curves cross the region separator in Figs. 1.5 and 1.6 are close to the tabulated values where the integrator fails to reach convergence for the first time, for the specific integrals. Hence, the results do not indicate that the error estimate is inaccurate for these integrals. On the contrary, the actual relative error in Figs. 1.5 and 1.6 is reduced stepwise, and lies close to (yet below) the region separator for all converging accuracies—indicating that the integrator is reliable and efficient, for integrals 9–12.

Test integral 6, with accuracy results plotted in Fig. 1.3, deviates somewhat from the lower region for precision between 10^{-2} and 10^{-1}. This is certainly not because the integrator does not converge in this region, but because error estimate allows for convergence at a too low level of integration, in this particular case. As mentioned in Sec. 1.1.5, such errors often occur for rather arbitrary changes in the integrand function (or, as in this case, for a rather arbitrary range of requested accuracies), and are hard to avoid. If such errors do occur for a large part of computed integrals, they might, however, indicate a malfunction in the strategic section of the integration algorithm. This is not the case for the tests performed in this chapter.

An aspect in the figures depicting the actual error, related to the efficiency of the

routine, is the regions where the actual error is much smaller than the accuracy requirement. Such behavior is noticeable in Figs. 1.2–1.4. The behavior may result from the error estimate being overly cautious, and constitutes much of the surcharge paid for automatic integration based on a nested quadrature scheme. The error estimate computed at a certain level of integration, is a reasonable estimate for the error at the previous level of integration. However, as it was not available at the previous level a surcharge of n/m (where n and m are the number of integration points at integration level N and $N-1$, respectively) is added to the computational cost of the automatic quadrature routine[7]. This surcharge is the price that is paid for using an automatic quadrature routine, rather than a rule evaluation quadrature routine. The additional computational cost is expected to rise rapidly if the error estimate, at some point, fail to be a reasonable estimate for the error at the previous integration level.

1.4.4 Round-off errors

Besides the error estimate performance, another aspect is worth commenting when comparing the requested accuracy to the actual error in Figs. 1.2–1.6. In all figures the actual relative error is reduced stepwise, as the input value for the requested relative accuracy is more demanding (i.e., closer to zero). This is the expected behavior as the truncation error (Eq. (1.8)) is reduced when more points are included in the numerical approximation of the integral (the integrator goes to a higher integration level to reach convergence). However, in each figure, as the requested relative accuracy approaches a smaller value, at least one of the computed integrals go from an actual relative error at one level, to a higher actual relative error at a higher level of integration. This is due to the round-off error (Eq. (1.9)) dominating over the truncation error in these cases: The actual error is increased, since increasing the number elements in the computed sum increases the round-off error more than truncation error is reduced. When computing the test integrals 1–12 in Tab. 1.1, with the GPat routine, the effect is negligible from a user point of view. This is due to the maximum number of points being quiet small, compared to other automatic integrators[8], and the floating point precision in use being double precision. None the less, this does illustrate the fact that more integration points do not necessarily equal a more accurate result. This type of error is expected to have an increased impact in single precision computations.

If interval subdivision is implemented in order to strive for convergence for hard integrals with GPat, the round-off error is expected to get an increased impact in the numerical integration. Hence, interval subdivision will not guarantee increased accuracy or increased range of integral handled by GPat.

1.4.5 Comparing integrator routine efficiency

Comparing the number of function evaluations the implemented integrator needs to reach convergence—for different accuracy requirements and integrand functions—to the corresponding results for different automatic integrators from the QUADPACK library, gives an important pointer towards the computational cost of the implemented routine. As QUADPACK is a well-established, well-documented, and much used numerical integrator library, it's routines are well-suited to be used in benchmarking of GPat.

[7] This is valid for a nested quadrature scheme, and is expected to be considerably higher for a non-nested scheme.

[8] E.g., adaptive Gaussian quadratures with a high limit on the number of subintervals, or automatic integrators based on Newton Cotes formulae, with high accuracy demands

As seen in Tabs. 1.4 and B.3 no single integrator routine outshines the others for all test integrals. Rather, the best performance for some of the integrals is achieved by one integrator, while for other integrals another routine is optimal. In this respect, one remark should be made: The user of an automatic integrator library does not know beforehand which of the routines will give optimal performance for the problem at hand. Since running all routines for the problem and selecting afterwards which one to use rarely is a good strategy, average results for different types of integrand functions and accuracies are useful. As is documentation that provides guidelines for which integrals an integrator routine is expected to compute accurately and efficiently.

The number of function evaluations needed to converge for the test integrals indicate good performance by GPat. Especially for relative accuracy input of 10^{-5}, where it has the best, or very close to the best, efficiency of the automatic integrators tested, and far better than the average integrator performance. As the relative accuracy input is set to 10^{-10} and 10^{-15}, in turn, the performance compared to the QUADPACK routines is somewhat degrading, but still not bad. The worst results are for the most demanding accuracy input. But although GPat does not stand out as the best routine for many of the integrals for this input, it is close to, or better than the average QUADPACK routine performance for the integrals where convergence is reached. An exception is for integral 7, where the number of function evaluations to reach convergence for GPat is the highest of the automatic integrators.

It should once again be brought to the readers attention that unlike the routines DQAG and DQAGS, GPat does not converge for all test integrals, and neither does DQNG. Both GPat and DQNG show good performance regarding efficiency, but the price they pay for this is that they are less versatile than the routines they are compared with.

Consider the averaged ratios plotted in Fig. 1.7. The figure shows that for requested relative accuracies $\varepsilon_{rel} \geq 10^{-9}$, GPat is more efficient than all QUADPACK routines utilized, when the average ratios defined by Eq. (1.10) are considered. For $10^{-15} < \varepsilon_{rel} < 10^{-9}$ it is among the two best performing routines, with competition from DQNG. For $\varepsilon_{rel} = 10^{-15}$ GPat drops to a fourth best average ratio. This is probably due to numerical computation of integral 13 with GPat failing to converge at this ratio, while it converges for $\varepsilon_{rel} = 10^{-14}$ with substantially less function evaluations than its competitors (255 with GPat, compared to the average for all converged integrators being 1639).

Among the integrators that converge for all integrals at all accuracies used during testing, the DQAGS routine has the overall best performance. This routine performs especially well for integrals with endpoint singularities in the integrand function[9], or in the integrand function's derivative: As seen in Tabs. 1.4 and B.3 the routine does not increase the number of function evaluations needed to evaluate integrals 9 and 11, when going from $\varepsilon_{rel} = 10^{-5}$ to $\varepsilon_{rel} = 10^{-15}$. The number of function evaluations to reach convergence for integrals 10 and 12, however, do increase (past those of integral 9 and 11), even though these integrals are assumed to be simpler to compute numerically. This leads to the speculation that some integrals might be handled as special cases by this routine, leading to very good performance at high accuracy demands.

1.4.6 Beyond general-purpose integration

The integrator reaches convergence for all test integrals at requested relative accuracy set to 10^{-5}. For high accuracy inputs a few of the tested integrals are to difficult for the integrator to compute. This is due to the nature of the integrand functions, making the integrals hard to compute numerically. The integrand function in the integrals in

[9] As stated in the routine's documentation.

Maxwell1D are known to be oscillatory, and expensive to compute. An in-depth analysis of these functions is necessary, in order to relate them to the results in this chapter.

The tests of the error estimate indicate that the automatic routine is reasonably reliable, as very few converged routine calls return results that are not within the requested accuracy. It is more likely that the error estimate is overly cautious, demanding more function evaluations than necessary, especially at low accuracy requirements. The computational cost related to the surcharge in an automatic integrator should be reduced, if possible, when the routine is customized to the Maxwell Eqs. solver.

Round-off errors are observed for high accuracy demands, but are not noticeable by the user for the integrals tested. As round-off errors usually have greater impact for single precision than for double precision computations, test for both these floating point precision will be included in the performance analysis in Chap. 2.

The efficiency comparison with the QUADPACK routines indicate above average performance by GPat. This indicates that the choice of quadrature is appropriate, as efficiency is a priority in the Maxwell Eqs. solver (particularly since the previous integration, with the midpoint rule, is of very low computational cost). Averaged results and battery test do, however, have limited value for a user that wishes to apply a numerical integrator routine on a specific type of integrals. Especially when the number of integrals that are to be computed is large and have similar characteristics. To be able to cope with such problems an integrator should be tested thoroughly for the particular problem family that the integrals represent. To this end, parametric studies of the automatic integrator routine are appropriate.

Chapter 2

Customization and performance analysis

With the automatic integrator developed and tested for general-purpose use, the time has come for a look at the specific tasks for which the routine has been developed. The Maxwell Eqs. solver *Maxwell1D* is a large software package—developed to study light scattering from randomly rough surfaces. The software package is well documented and thoroughly tested. Maxwell1D uses a midpoint scheme in the computation of a large number of integrals, an estimate that has proven too crude despite the integration intervals being small. The need for an improved numerical scheme for computing these integrals has arisen.

Several factors complicate the computations of the integrals in Maxwell1D, making it hard to find an appropriate library routine for computing the integrals, and not trivial to customize an existing routine to compute them. These are:

- The integral involves complex integrand functions and complex integrand function arguments.
- The arguments to the integrand function depends on the surface profile where it is evaluated. Thus, the integrand function $f(x) \Rightarrow f(x_1, x_3)$.
- The integrand function evaluations are dependent on surface elements both inside and outside the interval of integration.
- The surface is discrete, and each integration interval only include a single point (the midpoint).
- The surface may or may not have stochastic properties.
- A lot of integrals will be computed ($\Theta(N^2)$[1] integrals).
- Certain intervals of integration contains a singularity in the midpoint ($\Theta(N)$ integrals).
- The surface may possess overhangs, i.e., be a reentrant surface where x_3 cannot be expressed by a single-valued function of x_1.

In this chapter the focus is drawn from general numerical integration done with GPat, to adapting the integrator to specific problem families and analyzing the performance of the integrator after this customization.

Following this introduction, theory related to rigorous simulations of light scattering from rough surfaces is briefly presented, culminating in the two types of integrals that pose a challenge in Maxwell1D. Further, testing methods that belong in the category parametric studies of an integrator routine are presented. The derived integrals are studied

[1] Asymptotic notation Big Theta [7]

in detail, both on the surface as a whole and for single integration intervals. To this end, three surface profiles are used as test profiles. One weakly concave surface and two sinusoidal surfaces, representing a smooth, a rough and a very rough surface. Before the parametric studies of both the midpoint scheme and the GPat routine are performed, the integrals from Maxwell1D are re-written to a problem family formulation, appropriate for such studies. In addition to the midpoint scheme and GPat, a third alternative for numerical integration is suggested, based on the study of the integrand function on the small integration intervals and observations done in Chap. 1. Several approaches to deal with the challenges related to the discrete surface representation in the Maxwell Eqs. solver are suggested. These are relevant both for GPat and for the alternative integrator routine. A comprehensive amount of test results for the different integrator routines are presented and thoroughly discussed, before suggestions are made in regards to which routine that should be implemented in Maxwell1D.

2.1 Modeling light scattering from rough surfaces

There does not exist an analytic non-perturbative theory for modeling light scattering from rough surfaces of arbitrary roughness. Simonsen [31] argues that this due to higher order scattering process being important for strongly rough surfaces, and that the total field on the surface at some point depends on the total field in other locations on the surface (non-local effects).

To be able to model light scattering from rough surfaces a rigorous numerical simulation method is therefore necessary. The simulations in Maxwell1D is based on a method that derives a set of coupled integral equations for the source functions, the field and its normal derivatives on the surface. The method is well-documented in literature (see, e.g., [23,31]), and is presented in brevity here. The integrals derived here do, however, differ from published literature in the respect that the theory for scattering from reentrant surfaces [24] is combined with the general notation of [31]. This is necessary in order to obtain generalized analytic expressions for the integrals computed in the Maxwell Eqs. solver, applicable in computations for both multi-valued and single-valued surfaces systems, for both s and p-polarized incident light.

2.1.1 The scattering problem and the related integral equations

Consider the scattering geometry depicted in Fig. 2.1. The plane of incidence is the x_1x_3-plane, and the effective one-dimensional surface[2] is illuminated with an electromagnetic wave of frequency ω. The surface is represented by a continuous vector-valued function $\boldsymbol{R}(\xi,\eta)$ in the x_1x_3-plane. The incident light is either s or p-polarized, thus, one non-trivial field component is sufficient to fully describe the electromagnetic field. The primary field for the one-dimensional surface can be written as [31]:

$$\Phi_v(\boldsymbol{r}|\omega) = \begin{cases} H_2(\boldsymbol{r}|\omega), & v = p, \\ E_2(\boldsymbol{r}|\omega), & v = s, \end{cases} \tag{2.1}$$

where $H_2(\boldsymbol{r}|\omega)$ and $E_2(\boldsymbol{r}|\omega)$ denote the magnetic and electric field in the x_2-direction, respectively [31], and $\boldsymbol{r} = (x_1, x_3)$.

The primary field in the regions above and below the scattering interface, denoted by $\Phi_v^+(\boldsymbol{r}|\omega)$ and $\Phi_v^-(\boldsymbol{r}|\omega)$, respectively, satisfy Helmholtz equations. It can be shown (see,

[2]Surfaces with horizontal variations along one coordinate only [22]. In this case, the surface is constant along the x_2-direction.

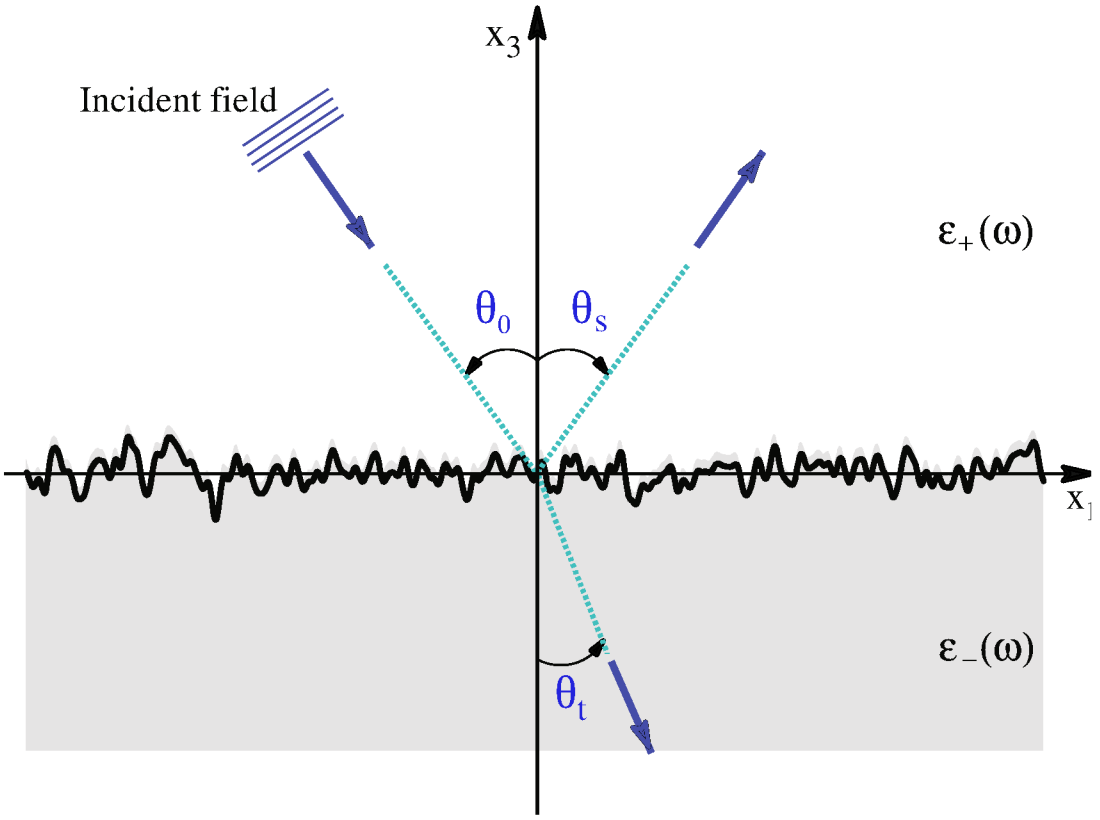

Figure 2.1: The scattering geometry used throughout this chapter and the next for a incident wave of angle θ_0 and frequency ω being scattered from a rough surface by an angle θ_s (or transmitted by an angle θ_t). The surface is the interface between the media of a frequency dependent dielectric function $\varepsilon_+(\omega)$ and $\varepsilon_-(\omega)$, respectively. Although not depicted in the figure, the interface may have overhangs (multi-valued surface profile). The figure is adapted after Ref. [31].

e.g., Ref. [23, 31]), that by applying Green's second integral theorem to the two regions and making use of the equations of continuity for the field and its normal derivative across the surface, result in expressions for the scattered field from an integral over the limiting surface involving the source functions [24]:

$$\mathcal{F}_v(\boldsymbol{R}) = \Phi_v^+(\boldsymbol{R}), \tag{2.2}$$

$$\mathcal{N}_v(\boldsymbol{R}) = \gamma(\boldsymbol{R})\hat{n}(\boldsymbol{R}) \cdot \left[\frac{\partial}{\partial \boldsymbol{r}}\Phi_v(\boldsymbol{r})\right]\Big|_{r=R} \tag{2.3}$$

where $\hat{n}(\boldsymbol{R})$ is the (normalized) surface normal, $\gamma(\boldsymbol{R})$ is the length of the unnormalized surface normal and

$$\frac{\partial}{\partial \boldsymbol{r}} = \frac{\partial}{\partial x_1}\hat{x}_1 + \frac{\partial}{\partial x_3}\hat{x}_3 \tag{2.4}$$

is the gradient vector in the x_1x_3-plane. The source functions can be expressed by the following set of inhomogeneous, coupled integral equations:

$$\mathcal{F}_v(\boldsymbol{R}) = \mathcal{F}_v^{inc}(\boldsymbol{R}) + \frac{1}{4\pi}\lim_{\delta\to 0^+}\int_{-\infty}^{\infty} \mathrm{d}S'\Big\{ \left[\hat{n}(\boldsymbol{R}')\boldsymbol{\nabla}_{\boldsymbol{r}} G_+(\boldsymbol{R}+\delta\hat{n}(\boldsymbol{R})|\boldsymbol{r})\right]_{\boldsymbol{r}=\boldsymbol{R}'} \times \mathcal{F}_v(\boldsymbol{R}') - G_+(\boldsymbol{R}+\delta\hat{n}(\boldsymbol{R})|\boldsymbol{R}')\frac{\mathcal{N}_v(\boldsymbol{R}')}{\gamma(\boldsymbol{R}')}\Big\}, \tag{2.5}$$

$$0 = \lim_{\delta\to 0^+}\frac{1}{4\pi}\int_{-\infty}^{\infty} \mathrm{d}S'\Big\{ \left[\hat{n}(\boldsymbol{R}')\boldsymbol{\nabla}_{\boldsymbol{r}} G_-(\boldsymbol{R}+\delta\hat{n}(\boldsymbol{R})|\boldsymbol{r})\right]_{\boldsymbol{r}=\boldsymbol{R}'} \mathcal{F}_v(\boldsymbol{R}') - \frac{\kappa_v^-}{\kappa_v^+} G_-(\boldsymbol{R}+\delta\hat{n}(\boldsymbol{R})|\boldsymbol{R}')\frac{\mathcal{N}_v(\boldsymbol{R}')}{\gamma(\boldsymbol{R}')}\Big\}, \tag{2.6}$$

where

$$G(\boldsymbol{r}|\boldsymbol{r}')_\pm = i\pi H_0^{(1)}\left(\sqrt{\varepsilon_\pm(\omega)}\frac{\omega}{c}|\boldsymbol{r}-\boldsymbol{r}'|\right), \tag{2.7}$$

is the Green's function for the regions + and − of the electric permittivity $\varepsilon_\pm$. The speed of light is c, and

$$\kappa_v^\pm = \begin{cases} \varepsilon_\pm(\omega), & v = p, \\ \mu_\pm(\omega), & v = s, \end{cases} \tag{2.8}$$

is used to generalize the notation. The function $H_0^{(1)}$ that occurs in the Green's function is the zeroth order Hankel function of the first kind [2].

By computing the source functions $\mathcal{F}_v$ and $\mathcal{N}_v$, the scattering amplitude, the transmission coefficient (if defined), and physical observable quantities, e.g., the mean differential reflection and transmission coefficients, can be computed [31]. The details of the computation of these physical observables are not included here, as the focus of this thesis is limited to improving the numerical approximation of integrals that occur when Eqs. (2.5) and (2.6) are solved. In order to compute the source functions the integral equations are converted to matrix equations, by discretizing the spatial variables $\boldsymbol{R}(\xi,\eta)$ and $\boldsymbol{R}'(\xi,\eta)$, and approximating the resulting integrals with an appropriate quadrature scheme. To the author's knowledge, no publications treat this without using a midpoint scheme for these computations. This is where the computations in Maxwell1D are expected to be insufficiently accurate. This problem will be treated in detail, but first analytic expressions for integrals must be derived.

2.1.2 Surface parametrization and discretization

Mendoza-Suáres and Méndez [24] provide a formal procedure for parametrization of the surface profile, such that the components (ξ,η) of the vector-valued function representing the surface profile may be expressed as functions of the arch length, s, of the surface Γ. A similar approach is taken here, yet with the parameter s not being restricted to representing the arch length of the curve Γ. The surface profile may, nonetheless, be represented by

$$\boldsymbol{R}(s) = [\xi(s), \eta(s)]\,. \tag{2.9}$$

Contrary to the parameter used in [24], the relation to the surface element $\mathrm{d}S$ is

$$\mathrm{d}S = \sqrt{[\xi'(s)]^2 + [\eta'(s)]^2}\mathrm{d}s = \gamma(s)\mathrm{d}s, \tag{2.10}$$

where $\xi'(s) = \left(\frac{\mathrm{d}\xi}{\mathrm{d}s}\right)$ and $\eta'(s) = \left(\frac{\mathrm{d}\eta}{\mathrm{d}s}\right)$ have been used for short.

This enables the normalized surface normal and normal derivative to be written as functions of s, by

$$\hat{n}(s) = \frac{1}{\gamma(s)} \left[-\eta'(s), \xi'(s)\right], \tag{2.11}$$

$$\hat{n}(s)\frac{\partial}{\partial \boldsymbol{r}} = \frac{1}{\gamma(s)} \left(-\eta'(s)\frac{\partial}{\partial x_1} + \xi'(s)\frac{\partial}{\partial x_3}\right), \tag{2.12}$$

The parametric interval is chosen to $[-L_s/2, L_s/2]$, where L_s represent the total length of the parametric space spanned by s. This may be chosen to an appropriate length, e.g., the arch length of the surface $L_T = \int_\Gamma \mathrm{d}S$ or the system width L_x, $x_1 \in [-L_x/2, L_x/2]$, depending on the desired spacing of grid points. The surface grid is defined by

$$\boldsymbol{R}_i = \left[\xi(s_i), \eta(s_i)\right], \qquad s_i = -\frac{L_s}{2} + \left(i - \frac{1}{2}\right)\Delta s, \qquad i = 1, 2, ..., N \tag{2.13}$$

with discretization size $\Delta s = L_s/N$. The source functions are assumed to be slowly varying functions over a grid cell, and are therefore put outside the integral, allowing Eqs. (2.5) and (2.6) to be written as matrix equations evaluated $\boldsymbol{R}(s_m)$, denoted the *observation point*, by

$$\mathcal{F}(s_m) = \mathcal{F}(s_m)^{inc} + \sum_{n=1}^{N} \left[\mathcal{A}^+_{mn}\mathcal{F}(s_n) - \mathcal{B}^+_{mn}\mathcal{N}(s_n)\right], \tag{2.14}$$

$$0 = \sum_{n=1}^{N} \left[\mathcal{A}^-_{mn}\mathcal{F}(s_n) - \mathcal{B}^-_{mn}\mathcal{N}(s_n)\right]. \tag{2.15}$$

Moreover, the matrix elements $\mathcal{A}^\pm_{mn}$ and $\mathcal{B}^\pm_{mn}$ are defined as integrals over grid cells of size Δs. These can be derived from Eqs. (2.5) and (2.6) by splitting the integrals into N integrals of size Δs, and by and using the transformation $\mathrm{d}S = \gamma(s)\mathrm{d}s$. The resulting expressions are

$$\mathcal{A}^\pm_{mn} = \int_{s_n-\Delta s/2}^{s_n+\Delta s/2} \mathrm{d}s\ A_\pm(s_m|s) = \int_{-\Delta s/2}^{\Delta s/2} \mathrm{d}u\ A_\pm(s_m|s_n+u), \tag{2.16}$$

$$\mathcal{B}^\pm_{mn} = \int_{s_n-\Delta s/2}^{s_n+\Delta s/2} \mathrm{d}s\ B_\pm(s_m|s) = \int_{-\Delta s/2}^{\Delta s/2} \mathrm{d}u\ B_\pm(s_m|s_n+u), \tag{2.17}$$

where the integrand functions $A_\pm(s_m|s)$ and $B_\pm(s_m|s)$, expressed in terms of the Green's function of Eq. (2.7), are

$$A_\pm(s_m|s) = \lim_{\delta\to 0^+} \frac{\gamma(s)}{4\pi} \left[\hat{n}(s)\frac{\partial}{\partial \boldsymbol{r}} G_\pm(\boldsymbol{R}(s_m) + \delta\hat{n}(s_m)|\boldsymbol{r})\right]_{\boldsymbol{r}=\boldsymbol{R}(s)}, \tag{2.18}$$

$$B_\pm(s_m|s) = \lim_{\delta\to 0^+} \frac{1}{4\pi} G_\pm(\boldsymbol{R}(s_m) + \delta\hat{n}(s_m)|\boldsymbol{R}). \tag{2.19}$$

The integrand function $A_\pm$ written in terms of the zero order Hankel functions of the first kind is

$$A_\pm(s_m|s) = \lim_{\delta\to 0^+} \frac{i\gamma(s)}{4} \left[\hat{n}(s)\frac{\partial}{\partial \boldsymbol{r}} H_0^{(1)}\left(\sqrt{\varepsilon_\pm(\omega)}\frac{\omega}{c}|\boldsymbol{R}(s_m) + \delta\hat{n}(s_m) - \boldsymbol{r}|\right)\right]_{\boldsymbol{r}=\boldsymbol{R}(s)}. \tag{2.20}$$

The derivative of the zeroth order Hankel function of the first kind is $\frac{\partial}{\partial z}H_0^{(1)}(z) = -H_1^{(1)}(z)$ (given by Eqs. (9.1.3) and (9.1.28) in [2]), where $H_1^{(1)}$ is the first order Hankel function of the first kind. By this, and Eqs. (2.11) and (2.12)), the expressions for the integrand functions can be written out in full. The integrand functions are

$$\begin{aligned} A_\pm(s_m|s) = \lim_{\delta\to 0^+}\left(-\frac{i}{4}\right)\varepsilon_\pm\frac{\omega^2}{c^2}\frac{H_0^{(1)}\left(\chi_\pm(s_m|s)\right)}{\chi_\pm(s_m|s)}\Big[&\left(\xi(s_m)-\delta_\eta-\xi(s)\right)\eta'(s) \\ &-\left(\eta(s_m)+\delta_\xi-\eta(s)\right)\xi'(s)\Big], \end{aligned} \tag{2.21}$$

$$B_\pm(s_m|s) = \lim_{\delta\to 0^+}\frac{i}{4}H_0^{(1)}\left(\chi_\pm(s_m|s)|\right), \tag{2.22}$$

where

$$\chi_\pm(s_m|s) = \sqrt{\varepsilon_\pm}\frac{\omega}{c}\sqrt{\left[\xi(s_m)-\delta_\eta-\xi(s)\right]^2+\left[\eta(s_m)+\delta_\xi-\eta(s)\right]^2}, \tag{2.23}$$

and

$$\delta_\eta = \delta\frac{\eta'(s_m)}{\gamma(s_m)}, \qquad \delta_\xi = \delta\frac{\xi'(s_m)}{\gamma(s_m)}, \tag{2.24}$$

are used to simplify the notation.

By taking the limit $\delta \to 0^+$, integrals are obtained that allow the computation off-diagonal matrix elements, $\mathcal{A}_{mn}^{\pm}\Big|_{n\neq m}$ and $\mathcal{B}_{mn}^{\pm}\Big|_{n\neq m}$, directly by Eqs. (2.16)–(2.17) and (2.21)–(2.23). The diagonal elements, on the other hand, need to be handled with care, as both the zeroth and the first order Hankel functions have singularities for an input argument equal to zero. Fortunately, these singularities are integrable. Approximate expressions for the elements with singularities can be obtained by using the small argument asymptotic expansion of the Hankel function, and Taylor expanding the Hankel function's argument around s_m for $m = n$. The procedure is described in detail in the appendix of [31], for single-valued surface functions. To allow for multi-valued surface functions, equivalent expressions to those in [31] are derived for the parametric surface representation. The derivation is rather technical, and is included in full in Appx. C.1. The results are:

$$\mathcal{A}_{mm}^{\pm} = \frac{1}{2}+\Delta s\frac{\eta''(s_m)\xi'(s_m)-\xi''(s_m)\eta'(s_m)}{4\pi\gamma(s_m)^2}, \tag{2.25}$$

$$\mathcal{B}_{mm}^{\pm} = \frac{i}{4}\Delta s H_0^{(1)}\left(\sqrt{\varepsilon_\pm}\frac{\omega}{c}\frac{\gamma(s_m)\Delta s}{2e}\right). \tag{2.26}$$

2.1.3 Single-valued surface profile

If the surface profile can be represented by a single valued function of x_1 the surface profile simplifies to

$$\boldsymbol{r} = \left[x_1, \zeta(x_1)\right], \tag{2.27}$$

where $\zeta(x_1)$ is the surface coordinate along the x_3-direction, expressed as a function of x_1. By using the grid definition of Eq. (2.13) with equidistant sampling in the x_1-direction (by setting $L_s = L_x$), with $\boldsymbol{R}_i = \boldsymbol{r}_i\left[x_1(s_i), \zeta(x_1(s_i)\right]$, $\xi(s) = x_1(s)$ and $\eta(s) = \zeta(x_1(s))$ the off-diagonal elements can be expressed by

$$\mathcal{A}_{mn}^{\pm} = \int_{x_n-\Delta x/2}^{x_n+\Delta x/2}\mathrm{d}x A_\pm(x_m|x), \tag{2.28}$$

$$\mathcal{B}_{mn}^{\pm} = \int_{x_n-\Delta x/2}^{x_n+\Delta x/2}\mathrm{d}x B_\pm(x_m|x), \tag{2.29}$$

where the integral is over $\Delta x = \Delta s|_{L_s=L_x} = L_x/N$. The integrand functions are

$$A_{\pm}(x_m|x) = \left(-\frac{i}{4}\right)\varepsilon_{\pm}\frac{\omega^2}{c^2}\frac{H_1^{(1)}(\chi_{\pm}(x_m|x))}{\chi_{\pm}(x_m|x)}\left[(x_m - x)\zeta(x) - (\zeta(x_m) - \zeta(x))\right], \tag{2.30}$$

$$B_{\pm}(x_m|x) = \frac{i}{4}H_0^{(1)}(\chi_{\pm}(x_m|x)), \tag{2.31}$$

where

$$\chi_{\pm}(x_m|x)) = \sqrt{\varepsilon_{\pm}}\frac{\omega}{c}\sqrt{(x_m - x)^2 + (\zeta(x_m) - \zeta(x))^2}, \tag{2.32}$$

and $x_m = x_1(s_m)$, $x_n = x_1(s_n)$ and $x = x_1(s)$ are used for short.

Similarly, the diagonal elements are simplified to

$$\mathcal{A}_{mm}^{\pm} = \frac{1}{2} + \frac{\Delta x}{4\pi}\frac{\zeta''(x_m)}{1 + \zeta'(x_m)^2}, \tag{2.33}$$

$$\mathcal{B}_{mm}^{\pm} = \frac{i}{4}\Delta x H_0^{(1)}\left(\sqrt{\varepsilon_{\pm}}\frac{\omega}{c}\frac{\sqrt{1 + \zeta'(x_m)^2}\Delta x}{2e}\right), \tag{2.34}$$

which are consistent with the diagonal elements in [23, 31]. By evaluating Eqs. (2.28) and (2.29) by the midpoint rule, the off-diagonal elements are also consistent with these publications.

This concludes the derivation of the integrals for which the automatic integrator is developed. By numerical integration and linear algebra techniques, Eqs. (2.14)–(2.17) (alternatively, the simpler equations for single-valued surface functions) may be solved in order to obtain the source functions. The numerical integration of these specific integrals will now be studied in detail.

2.2 Parametric studies of numerical integrator routines

The task of the numerical integrator is narrowed down, from general-purpose integration to integrating the two sets of integrals defined by Eqs. (2.16) and (2.17). The method of parametric studies of a numerical integrator routine allows for a more comprehensive knowledge of the integrator routine's performance, when a class of integrand functions, known as a problem family, is studied. Two such methods are described here, the integrator's performance profile and statistical distribution function. To be able to utilize these techniques in relation to the integrals of interest, the integrals in the matrix elements $\mathcal{A}_{mn}^{\pm}$ and $\mathcal{B}_{mn}^{\pm}$ are re-written to a problem family formulation.

2.2.1 Performance profiles

A performance profile is constructed for a problem family, by varying one parameter in the integrand function and computing the numerical integral of as many values of this parameter as possible. An example of a one-parameter problem family is the one used in the introduction of the performance profile concept [20]:

$$I(\lambda) = \int_1^2 \frac{0.1}{(x - \lambda)^2 + 0.01}\mathrm{d}x, \qquad 1 \leq \lambda \leq 2. \tag{2.35}$$

A performance profile is the actual error in the numerical approximation of an integral formulated such as that in Eq. (2.35) plotted against the parameter λ, for a constant

user requested accuracy, ε_{quad}[3]. A single performance profile is not sufficient to evaluate the quality of an automatic quadrature routine, but several such profiles, for different accuracies, will indicate the quadrature routine's performance with regards to accuracy and reliability for a certain class of integrand functions. The performance profile of an automatic integrator will typically be a jagged curve, with discontinuities as the integrator converges at a low level of integration for certain values of λ, and at a high level for others. In contrast, the performance profile of a rule evaluation quadrature routine is usually quite smooth (at least in the sense that the curve is continuous).

2.2.2 An integrator routine's statistical distribution function

Based on the nature of the performance profile, Lyness and Kaganove [21] developed a technique to simplify the process of benchmarking and evaluating the performance of automatic integrator routines. The technique relies heavily on a so called *statistical distribution function*, a natural extension of the performance profile approach as it links together the accuracy, efficiency and reliability of an automatic integrator. It does, however, increase the time used to compute the test results, as a lot of experimental runs are needed.

Consider the problem family defined by Eq. (2.35). When analyzing an integrator routine's performance for such a problem family, by parametric studies, the quantities of interest are: The requested accuracy, ε_{quad}, the actual accuracy of the integrator's output, ε_{act}, the problem family parameter, λ, and the number of function evaluations to converge, n_{eval}. The requested accuracy and the problem family parameter are input values to the numerical integrator, and the two remaining quantities are dependent on these. Hence, $\varepsilon_{act} = \varepsilon_{act}(\lambda, \varepsilon_{quad})$ and $n_{eval} = n_{eval}(\lambda, \varepsilon_{quad})$. As described in the previous section, the plot of ε_{act} as a function of λ, for a constant ε_{quad}, constitute the performance profile.

As a basis for the distribution function evaluation technique, two measures that treat the problem family as a whole are introduced. These are the average function value count, $n_{eval}(\varepsilon_{quad})$, and the statistical distribution function, $\phi(x; \varepsilon_{quad})$:

$$n_{eval}(\varepsilon_{quad}) = \frac{1}{\lambda_+ - \lambda_-} \int_{\lambda_-}^{\lambda_+} n_{eval}(\lambda, \varepsilon_{quad}) \mathrm{d}\lambda, \tag{2.36}$$

$$\phi(x; \varepsilon_{quad}) = \frac{1}{\lambda_+ - \lambda_-} \int_{\lambda_-}^{\lambda_+} \theta(x - |\varepsilon_{act}(\lambda, \varepsilon_{quad})|) \mathrm{d}\lambda, \tag{2.37}$$

where λ_- and λ_+ are the limits of the parameter λ for the particular problem family (for the problem family in Eq. (2.35) these are 1 and 2, respectively), x is an accuracy requirement (may or may not be equal to the requested accuracy), and $\theta(t)$ is the piecewise constant unit step function [2],

$$\theta(t) = \begin{cases} 0 & t < 0 \\ \frac{1}{2} & t = 0 \\ 1 & t > 0 \end{cases}. \tag{2.38}$$

It has been argued that the statistical distribution function is a more useful evaluation tool than, say, the root mean square average of ε_{act}, due to the nature of the quadrature routine's error. In general, a good automatic quadrature routine computes results within the requested accuracy most of the time, but may fail dramatically for a few members

[3]The user requested error, ε_{quad}, may represent the relative or absolute error, or a combination of the two

of a problem family. A root mean square average would be unduly influenced by such individual wild values [20, 21].

The statistical distribution function, on the other hand, supplies the user of such a technique with a probability for reaching an accuracy requirement x in the approximated integral—a success probability—when numerically approximating a member of the specific problem family with a requested accuracy ε_{quad}. Hence, $0 \leq \phi(x, \varepsilon_{quad}) \leq 1$. Each requested accuracy has a related average function value count, linking the reliability and accuracy to the efficiency of the computations.

2.2.3 Problem family formulation and challenges

The integrals in the Maxwell Eqs. solver are well-suited to be re-written to a more appropriate form for parametric studies, that is, to a problem family formulation. As the test performed in this chapter will consist of single-valued surface profile function[4], the integrals defined by Eqs. (2.28) and (2.29) will be considered. The notations $x_m = x_1(s_m)$, $x_n = x_1(s_n)$ and $x = x_1(s)$ will be used to shorten the expressions, when considered appropriate. A problem family formulation of these integrals can be obtained by letting the midpoint in each integration interval represent the problem family parameter, λ. Thus, the matrix elements $\mathcal{A}^{\pm}_{mn}$ and $\mathcal{B}^{\pm}_{mn}$ constitute two different problem families with the problem family parameter

$$x_n \to \lambda \quad \Rightarrow \quad \lambda \in \left[-\frac{L}{2} + \frac{\Delta x}{2}, -\frac{L}{2} + \frac{3\Delta x}{2}, ..., \frac{L}{2} - \frac{3\Delta x}{2}, \frac{L}{2} - \frac{\Delta x}{2} \right]. \tag{2.39}$$

The choice of λ as a part of the integration limits may seem counter-intuitive, but with a variable transformation it is evident that the integrals take on a form similar to the problem family example in Eq. (2.35). The transformation is:

$$\mathcal{A}^{\pm}_m(\lambda) = \int_{\lambda-\Delta x}^{\lambda+\Delta x} \mathrm{d}x \; A_{\pm}(x_m|x) = \int_{-\Delta x/2}^{\Delta x/2} \mathrm{d}u \; A_{\pm}(x_m|u+\lambda), \tag{2.40}$$

$$\mathcal{B}^{\pm}_m(\lambda) = \int_{\lambda-\Delta x}^{\lambda+\Delta x} \mathrm{d}x \; B_{\pm}(x_m|x) = \int_{-\Delta x/2}^{\Delta x/2} \mathrm{d}u \; A_{\pm}(x_m|u+\lambda), \tag{2.41}$$

where $A_{\pm}$ and $B_{\pm}$ are the integrand functions defined by Eqs. (2.30) and (2.31). The two problem families are made up of all the intervals of integration for a surface realization with system width L_x. They differ from the problem family formulation of Eq. (2.35) in some important aspects:

- The parameter λ takes on a finite number of discrete values. This limits the number of members in the problem family, making it simpler to generate performance profiles and distribution functions.
- The problem families are not uniquely defined. A change in the permittivity, wavelength, discretization or surface profile will result in a similar, yet different, problem family. Hence, each of the two problem family in Eqs. (2.40) and (2.41) consist of an infinite number of problem families, meaning that each performance profile and statistical distribution function is only valid for a specific domain with a specific parameter configuration and surface profile.

[4] Although developed to handle multi-valued surface profiles, it is necessary to limit the analysis to single-valued surface profiles to allow the correct results to be computed with the mathematical software Maple.

The latter of these aspects is a major disadvantage, as it practically reduces the parametric studies to battery experiments for different parameter configurations and surface profiles. The problem family formulation is, however, still useful, as the parametric studies it is used in represent a systematic way of testing the integrator routine for different configurations. This simplifies comparing the performance of different integrator routines for several surface profiles and parameter configurations.

As the parameter λ only takes on a finite number of discrete values the integrals in the quadrature routine evaluation measures in Eqs. (2.36) and (2.37) are simply computed by averaging the results over all possible λ-values—practically approximating the integrals by a extended midpoint rule[5]. This means that, similarly to the method used in used in [21], the routine evaluation measures integrals are approximated by

$$n_{eval}(\varepsilon_{quad}) = \frac{1}{m}\sum_{i=1}^{m} n_{eval}(\lambda_i, \varepsilon_{quad}), \tag{2.42}$$

and

$$\phi(x, \varepsilon_{quad}) = \frac{1}{m}\sum_{i=1}^{m} \theta(x - |\varepsilon_{act}(\lambda_i, \varepsilon_{quad}|), \tag{2.43}$$

but contrary to the method described in the research article, the values λ_i for $i = 1, 2, ..., m$ are not chosen using a random number generator. Rather, λ_i are all the possible values for λ (as defined in Eq. (2.39)) for the specific test system . Hence, m equals the number of grid points in the discretized light scattering interface.

In order to generate values for ε_{act}, used in both performance profiles and statistical distribution functions, the approximated integral results must be compared to the correct results. The integrals in the problem families defined by Eqs. (2.40) and (2.41) cannot be evaluated on closed form, so to arrive at results that are correct the mathematical software Maple is used. The requested relative accuracy of the results computed in Maple is set to 10^{-25}, and extended precision[6] is used to support 30 digit numbers in the computations. The requested relative accuracy in Maple is stricter than the possible accuracy of numerical integration for single or double precision in GPat. It is set to 10^{-25} such that the result computed in Maple hopefully are correct in the sense that the relative error is less than the smallest possible error in the double precision computations (making they results appear correct for single and double floating point precision). There is, however, no guarantee that the integration with Maple reach the requested accuracy, even though Maple's *int*-function may utilize several different quadrature schemes in doing so.

2.3 Problem analysis and integrator routine testing

The motivation for this thesis rest on the foundation that the current method used for computing the integrals in Eqs. (2.16) and (2.17) is not sufficiently accurate, and that the numerical accuracy may be improved by changing quadrature scheme and integrator routine.

In this sections the foundation of these assumptions is looked at closer, and configurations used for testing them are described. This includes a more comprehensive look on the integrand functions of interest for a few test systems and exploring challenges related to the use of a more sophisticated quadrature scheme than the midpoint rule, for the surface discretized by Eq. (2.13). In addition, an alternative integrator routine is suggested,

[5]This is in contrast to the Monte Carlo integration used by Lyness and Kaganove [21] when introducing the statistical distribution function technique.

[6]Extended precision between the basic floating point formats double and quadruple precision [1]

Table 2.1: Parameter configuration and surface profile functions for the three systems used in the parametric studies. Single-valued surface profiles where the x_3 coordinate given as a function x_1 by $\zeta(x_1)$. The systems have wavelength λ, electric permittivity ε, discretization size Δx and system width L_x.

Profile	$\zeta(x_1)$	λ	ε	Δx	L_x
Smooth	$0.0001x^2$	0.600 μm	1.0	0.1λ	50 λ
Rough	$0.1\lambda\sin(2\pi x_1)$	0.600 μm	1.0	0.1λ	50 λ
Very rough	$\sin(x_1)$	0.600 μm	1.0	0.1λ	50 λ

as a compromise between the efficiency of the midpoint scheme and the accuracy of the automatic integrator routine.

The integrator routine comparison is mainly focused on the off-diagonal elements in $\mathcal{A}^{\pm}_{mn}$ and $\mathcal{B}^{\pm}_{mn}$, i.e., the elements that do not contain singularities. As there are N diagonal elements for every N^2 matrix elements the diagonal elements should, however, not be neglected. A strategy for improving the accuracy in the computation of the diagonal elements is presented towards the end of this section.

2.3.1 Test systems and a closer look at the integrand function

As mentioned in Sec. 2.2.3, the problem families are not uniquely defined. It is therefore necessary to perform parametric studies for several configurations of the variables in the argument to the integrand function in Eqs. (2.40) and (2.41). For simplicity, the parameter configuration of electric permittivity, ε, wavelength, λ, discretization size, Δx, and system width, L_x, is held constant for all experiments in this chapter, while three different surface profiles are considered. These profiles represent a smooth, a rough, and a very rough surface profile, respectively[7], where the main focus will be on the rough surface. The surface coordinates of each surface is given by $\boldsymbol{r} = [x_1, \zeta(x_1)]$. Table 2.1 list the parameter configuration and surface profile functions of the three test systems. The surface profiles of the three configurations are in addition presented graphically in Fig. 2.2.

Figure 2.3 depicts the integrand function defined by Eq. (2.30), for the three configurations listed in Tab. 2.1 with the observation point $x_m = 0$, and x_n going from $-L_x/2+1/2$ to $L_x/2-1/2$ (i.e., x ranges from $-L_x/2$ to $L_x/2$). The integrand is strongly affected by the choice of surface in the system. This is not the case for the integrand function defined by Eq. (2.31). A single plot for this function, with the same observation point placement and x-range, is seen in Fig. 2.4 for the rough surface system in Tab. 2.1. It is close to identical for the two other systems (see Appx. C.2 for comparison).

The figures show the highly oscillatory behavior of the integrand functions. With the discretization size as specified in Tab. 2.1 the integrands are, however, quite smooth within an integration interval. In the specified test systems, the number of grid points on the discretized surfaces $N = 500$.

The smoothness of several integration intervals can be seen in Fig. 2.5. The figure depicts the real and imaginary part of a scaled version of the integrand function A_+[8] in $\mathcal{A}^{+}_{mn}$ for a the rough surface profile system defined in Tab. 2.1. The integration intervals in the figure have midpoints at distances $|x_m - x_n| = \{\Delta x,\, 2\Delta x,\, 5\Delta x,\, 10\Delta x, 20\Delta x\}$ from the observation point. To simplify the comparison of the computed integrand function

[7] The classification of a surfaces roughness will be studied in detail in Chap. 3

[8] Notice that the subscript + is used in stead $\pm$. This is not a typo, but meant to reflect that only a single medium (vacuum) is used in Tab. 2.1, hence the functions are only considered in the region of electric permittivity ε_+ in Fig. 2.1.

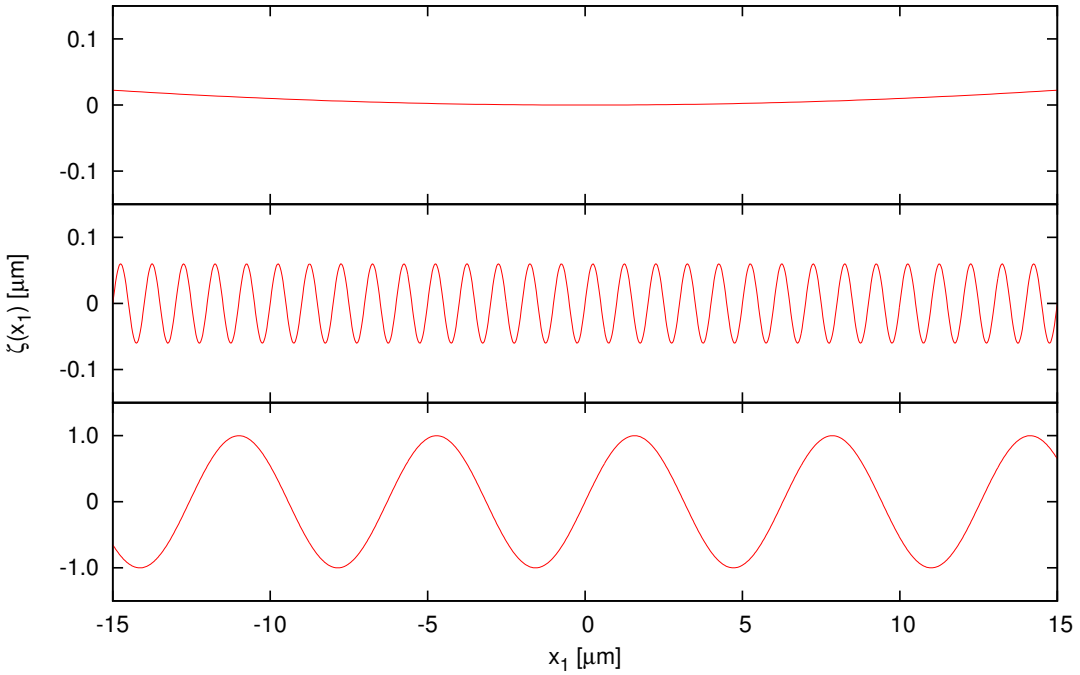

Figure 2.2: The smooth (top), rough (mid) and very rough (bottom) surfaces of Tab. 2.1. Note that the scale on the vertical axis is not equal for all surfaces, making the two upper surfaces seem much rougher compared to the bottom surface, than when viewed on equally scaled axis.

values at different integration intervals the plotted values have been scaled by subtracting the value of the integrand function at the midpoint of each interval:

$$\alpha(x, x_n) = A_+(0|x) - A_+(0|x_n), \tag{2.44}$$

such that all plotted curves are zero at $x = x_n$. Similar plots, for a scaled version of the integrand function in the $\mathcal{B}^+_{mn}$ are included in Appx. C.3.

The integrand functions are quite smooth on the integration intervals—due to the small interval size—but the curves in Figs 2.5 and C.4 are not linear. The deviation from a linear approximation varies from interval to interval, something which is expected to be reflected in the midpoint rule approximated results. Figures 2.4 and C.1 indicate that the integration in the matrix elements $\mathcal{B}^\pm_{mn}$ will be most challenging for the elements close to the observation point (near the diagonal elements), but will not be very much affected by the surface profile. The integration in the matrix elements $\mathcal{A}^\pm_{mn}$, on the other hand, will depend heavily on the surface profile (considering the large difference between the plotted curves in Figs. 2.3), and can therefore not be assumed to be simpler to compute in specific areas.

2.3.2 Evaluating the performance of the integrator routines

The midpoint rule integration is a simple rule evaluation quadrature routine that uses the grid points of the surface profile to approximate each integral with a single integrand function evaluation. As no accuracy requirement is requested by a rule evaluation routine, computing the statistical distribution function for the midpoint rule results in a single curve. The corresponding performance profile to this curve contains a more detailed view of the information about the midpoint rule performance. One such performance profile is

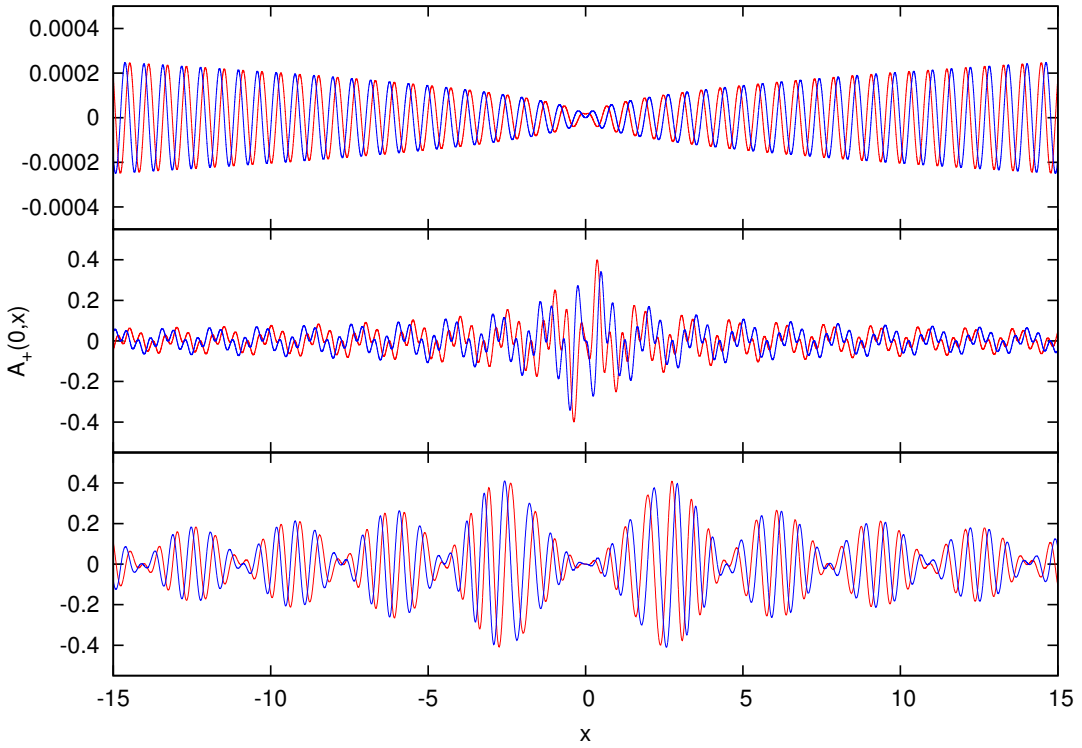

Figure 2.3: Integrand function $A_+(x_s|x)\Big|_{x_s=0}$ given by Eq. (2.30) plotted for the smooth (top), rough (mid) and very rough (bottom) surfaces defined in Tab. 2.1. The real (red) and imaginary (blue) parts of the functions are plotted in separate curves.

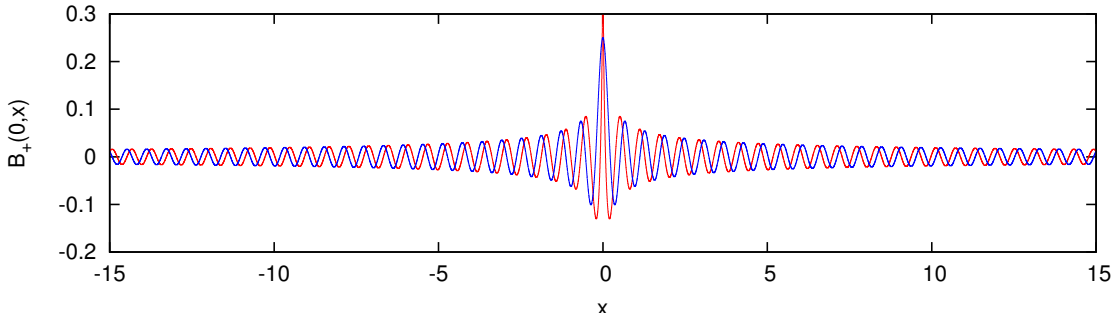

Figure 2.4: Integrand function $B_+(x_s|x)\Big|_{x_s=0}$ given by Eq. (2.31) plotted for the rough surface defined in Tab. 2.1. The real (red) and imaginary (blue) part of the function are plotted in separate curves.

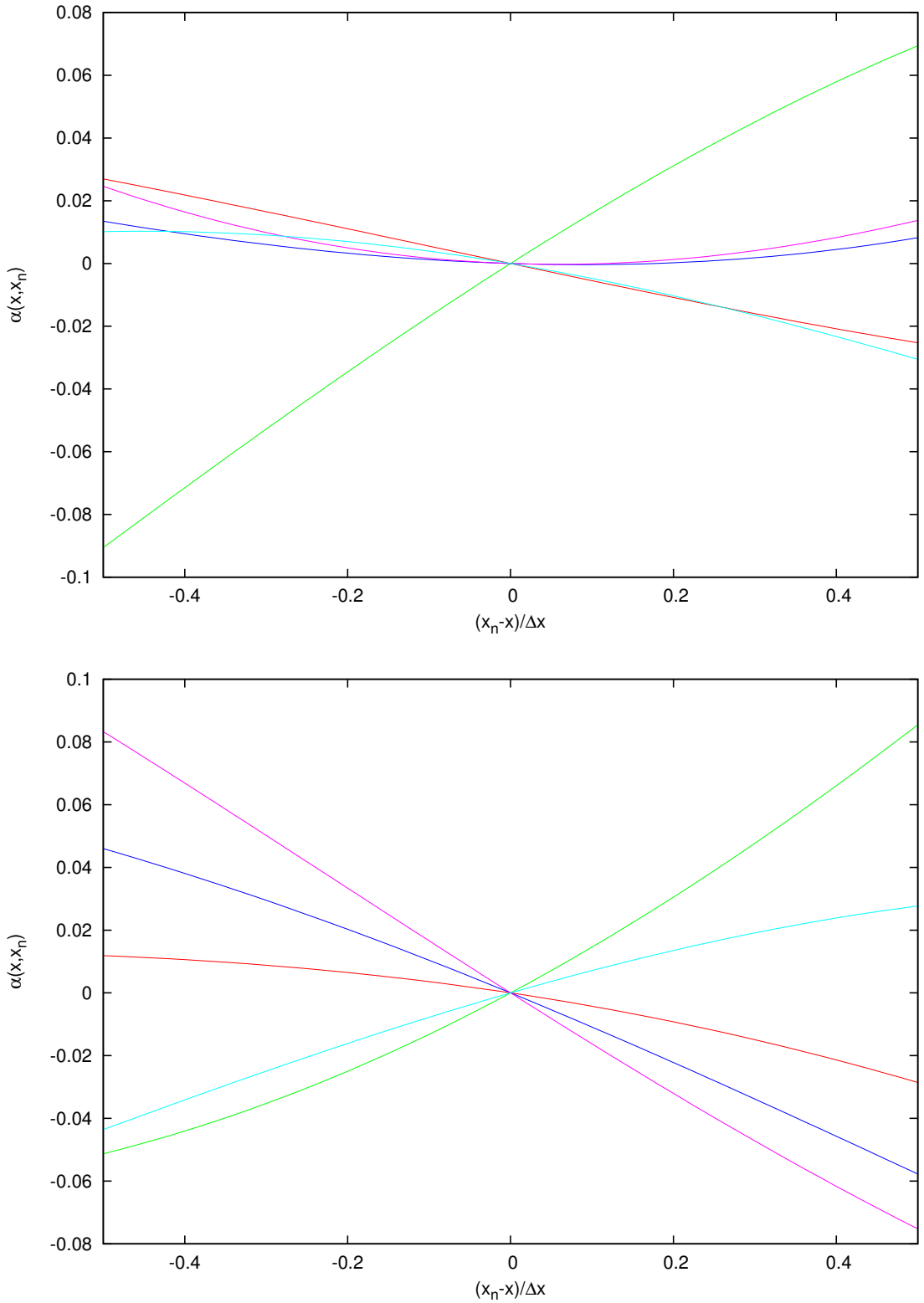

Figure 2.5: Scaled integrand function $\alpha(x, x_n)$ given by Eqs. (2.30) and Eq. (2.44), plotted for integration intervals with midpoints at distances from the observation points (in units of Δx) of 1 (red), 2 (blue), 5 (green), 10 (pink) and 20 (cyan). The curves are for the rough surface system of Tab. 2.1, with observation point $x_m = 0.0$. The complex integrand function is separated into a real part (top) and an imaginary part (bottom).

necessary for each of the problem families, for every parameter and surface configuration tested.

For the automatic integrator routine the statistical distribution function, $\phi(x, \varepsilon_{quad})$, is a more appropriate evaluation tool, than performance profiles, as several requested accuracies can be included in each plot of ϕ. The disadvantage of the statistical distribution function is that each point in a graphical representation of $\phi(x, \varepsilon_{quad})$ requires a computation of the integral in Eq. (2.37), which again requires the computation the actual error, $\varepsilon_{act}(\lambda, \varepsilon_{quad})$ for many λ-values (at the requested accuracy at the specific point). These computational time of generating the statistical distribution function are, however, nowhere near the time consuming numerical integration in Maple. It is therefore a priority to gain as much knowledge about the integrator routines performances for each of the test systems in Tab. 2.1, rather than to expand the number of systems.

While the midpoint rule simply computes the matrix elements $\mathcal{A}^{\pm}_{mn}$ and $\mathcal{B}^{\pm}_{mn}$ by directly using the data for the surface coordinates and their first order derivatives, available for each value for m and n, the Gauss-Patterson quadrature scheme in GPat requires the integrand function in each matrix element to be evaluated outside the midpoint (thus, outside the related grid point). There are several possible ways to approximate these values, by making use of the data available in the grid point data structure (or expanding it, if necessary). Three different methods are used during this testing, listed by increasing complexity and computational cost:

- Taylor expansion in the grid point, making use of the first and second order derivatives included in the grid point data container. Allows for second order Taylor approximation of $\xi(s)$ and $\eta(s)$[9], and first order Taylor approximation of $\xi'(s)$ and $\eta'(s)$.
- Higher order Taylor expansion in the grid point, by increasing amount of data included in the grid point data structure to include the third order derivatives of $\xi(s)$ and $\eta(s)$. Allows for third order Taylor approximation of the surface coordinates, and second order Taylor approximation of their first order derivatives.
- Hermite interpolation, by passing information about the neighboring grid points to the integrator routine. Allows computation of a eighth and a fifth order Hermite polynomial[10] for approximating the points and their first order derivatives, respectively, without expanding the amount of data included in the grid point. Divided difference method used when computing the interpolatory polynomials [5].

Statistical distribution functions are generated for GPat for the three different methods of approximating the surface profile mentioned. In addition, such functions are generated for an analytic surface profile representation. As the statistical distribution function is developed for comparison of automatic integrator routines it serves well for comparing the performance of the different surface approximations.

In contrast to the computations done with the midpoint rule, the accuracy of results computed with GPat are noticeably impacted by the choice of floating point precision. Therefore, both single and double floating point precision results are presented.

[9] Notice that $\xi(s)$ and $\eta(s)$ are considered, as here. This is due to the approximate surface representation being necessary regardless of the surface profile defined by single-valued or multi-valued function.

[10] Strictly speaking, the interpolation uses osculating polynomials, where the special case of Hermite polynomials are in use for the approximation of the derivatives only. The terminology of generalized Hermite interpolation is, however, consistent with literature, for interpolation with the use of osculating polynomials of higher derivatives [33]. The generalized Hermite polynomials enable both the first and second order derivatives to be used in approximating the coordinates on the surface.

2.3.3 A alternative integrator routine

The intervals of integration are small for the integrals in Eqs. (2.16) and (2.17), and the scaled versions of the integrand function (Figs. 2.5 and C.4) indicate that the integrals are fundamentally different from most of the test integrals in Tab. 1.1. Even though the functions have similar characteristics to the highly oscillatory sinc-integrals (integrals 13 and 14 in Tab. 1.1) when considered on the system width scale (Figs. 2.3 and 2.4), they are similar to low order polynomials within each integration interval.

This motivates the introduction of a third integrator routine. The routine should be much simpler than the automatic integrator GPat, but more sophisticated than the midpoint approximation. To this end a rule evaluation routine utilizing a Gaussian quadrature scheme is expected to be both efficient and accurate. For simplicity, such a routine is constructed by utilizing the rule evaluation part of GPat for two choices for the number of integrand points. These choices, and their corresponding degree of precision, are:

$$\begin{aligned} n &= 3, \quad d = 5, \\ n &= 7, \quad d = 11. \end{aligned} \tag{2.45}$$

Hence, the choices are equivalent to convergence at level one and two in GPat, respectively. The routine is referred to as GPatREQR (Gauss-Patterson Rule Evaluation Quadrature Routine[11]). Exact data for the surface profile only being available in the midpoint of each integration interval poses a challenge for GPatREQR, as in GPat (as described in the previous subsection).

2.3.4 Handling the singularities

The midpoint integrator routine in use in the Maxwell Eqs. solver handles the matrix elements that contain singularities by treated them as special cases. These special cases are computed directly by the Eqs. (2.25) and (2.26).

The automatic integrator routine GPat is not developed to handle singularities within the integration interval. There are, however, several ways to customize the integrator routine to handle such integrals. A simple, and efficient, way to improve accuracy in the computations of the $\mathcal{A}^{\pm}_{mm}$ and $\mathcal{B}^{\pm}_{mm}$ is to split the integration interval containing the singularity into three part, where the center part contains the singularity. The diagonal elements can then be computed by

$$S_{\mathcal{A}} = \mathcal{A}^{\pm}_{mm}\Big|_{\Delta s=2\delta} + \int_{s_m-\Delta s}^{s_m-\delta} \mathrm{d}s\ A_{\pm}(s_m|s) + \int_{s_m+\delta}^{s_m+\Delta s} \mathrm{d}s\ A_{\pm}(s_m|s), \tag{2.46}$$

$$S_{\mathcal{B}} = \mathcal{B}^{\pm}_{mm}\Big|_{\Delta s=2\delta} + \int_{s_m-\Delta s}^{s_m-\delta} \mathrm{d}s\ B_{\pm}(s_m|s) + \int_{s_m+\delta}^{s_m+\Delta s} \mathrm{d}s\ B_{\pm}(s_m|s), \tag{2.47}$$

where $\mathcal{A}^{\pm}_{mm}\Big|_{\Delta s=2\delta}$ and $\mathcal{B}^{\pm}_{mm}\Big|_{\Delta s=2\delta}$ are the diagonal elements defined by Eqs. (2.25) and (2.26), evaluated for $\Delta s = 2\delta$. The integrand functions are those defined by Eqs. (2.21) and (2.22), and $0 < \delta \ll \Delta s$ is dependent on the precision used in the computations, to ensure that $\delta > 0$.

The assumption that diagonal elements are computed more accurately by evaluating $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$, than by computing the matrix elements directly by Eqs. (2.25) and (2.26) is

[11] The rule evaluation routine utilized with $n = 3$ is, strictly speaking, a Gauss-Legendre quadrature scheme, as non of Patterson's optimal extensions are applied for three integration points (see Sec. 1.1.4).

checked, by treating the single-valued surface profile equivalents of Eqs. (2.46) and (2.47) as problem families, where parameter $\lambda = x_m$. In this way, all N elements containing singularities, for a certain test configuration, are members of the problem family. However, as neither Maple nor the QUADPACK routine DQAGS are able to compute the diagonal elements $\mathcal{A}^{\pm}_{mm}$ analytically, the approximation in Eq. (2.46) is used in the Maple computations as well as in the computations with GPat.

2.4 Results – Performance profiles and distribution functions

Three test systems are defined in Tab. 2.1, with surfaces ranging from smooth to very rough. As the amount of data produced by parametric studies are very large, the focus of this section is the results from studying the rough surface with surface profile function $\zeta(x_1) = 0.1\lambda \sin(2\pi x_1)$, where λ is the wavelength (not to be confused with the problem family parameter with the same symbol). When deemed necessary, the results for the two other test systems are included in the appendices and referred to in this section.

In all systems studied, the observation point, x_m, is in the leftmost integration interval (except when the diagonal elements are studied). Hence, $x_m = -\frac{L_x}{2} + \frac{\Delta x}{2} = -14.97$. This allows the reader to view the performance profile from left to right as integrals computed with increasing distance from the singular diagonal element. Figures depicting the integrand functions with $x_m = -14.97$ are included in Appx. C.2.

2.4.1 Midpoint rule approximations

Figures 2.6 and 2.7 show performance profiles for the rule evaluation quadrature routine that utilizes the midpoint rule, for the problem families defined by Eqs. (2.40) and (2.41), respectively, for the rough surface system with surface profile listed in Tab. 2.1. Performance profiles are given for the actual error $\varepsilon_{act} = I_{mid} - I$ (where I_{mid} is the midpoint approximated result and I is the result computed with Maple) and the actual relative error $\varepsilon_{act,rel} = \left|\frac{I_{mid}-I}{I}\right|$.

Performance profiles for the midpoint rule integrator for the system with smooth surface and the system with very rough surface, defined in Tab. 2.1, are included in Appx. D.1. For both systems the singularity placement is the same as the one used for generating Figs. 2.6 and 2.7. In addition, statistical distribution function curves for the midpoint rule approximated integrals are included in Appx. D.2. As a rule evaluation quadrature routine only generated one distribution function curve for each configuration and problem family[12] the curves for the three surface profiles systems are plotted in the same figures.

2.4.2 Automatic integrator routine results for an analytic surface profile representation

The results in Fig. 2.8 show statistical distribution functions for the automatic integrator routine GPat for the real part of the problem families defined by Eqs. (2.40) and (2.41). The test system is again the rough system in Tab. 2.1 with the observation point in the leftmost integration interval. Both single and double precision results are included, and the accuracies in the figure are relative accuracies. The equivalent distribution function curves for the imaginary parts of the problem families are included in Appx. D.2. The

[12]Strictly speaking two curves, in this case, one for the real part and one for the imaginary part of the result.

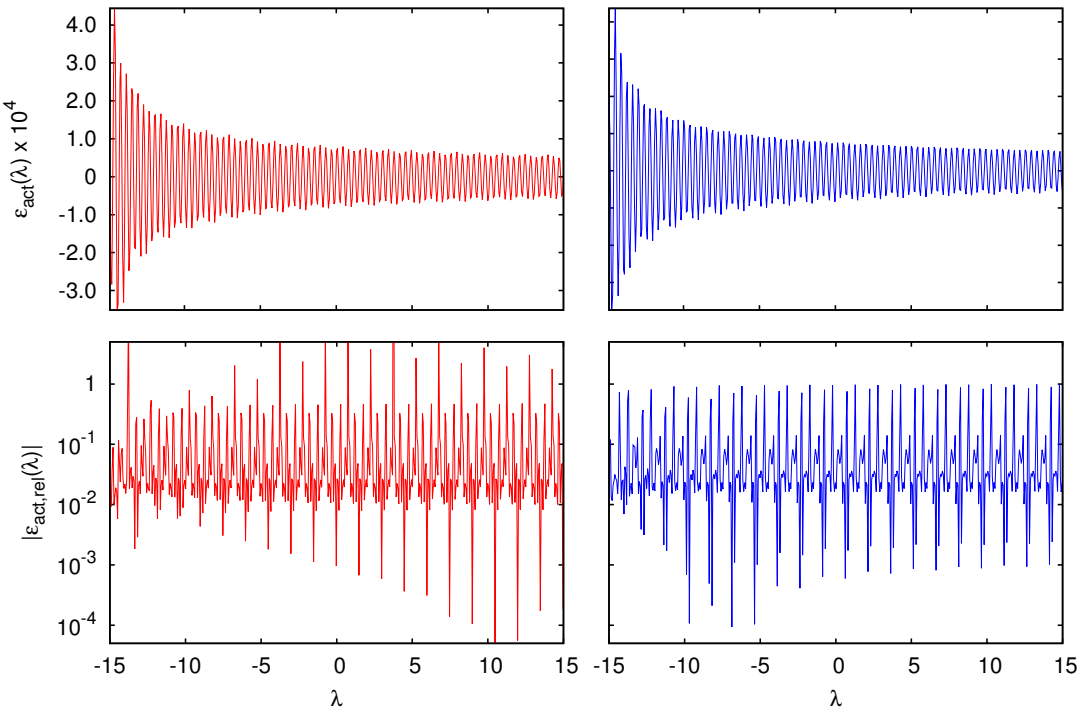

Figure 2.6: Performance profiles for the midpoint rule integrator, for the problem family defined by Eq. (2.40), with the rough surface system defined in Tab. 2.1 and $x_m = -14.97$. The performance profile is given for the real part (red) and imaginary part (blue). The actual error ε_{act} and actual relative error $\varepsilon_{act,rel}$ results are included.

differences between them and the results in Fig. 2.8 are negligible, and they are included for completeness.

Curves for several requested relative accuracies are plotted in each figure. These are not chosen arbitrarily. Consider the definition of an integrator routine's statistical distribution function, $\phi(x;\varepsilon_{quad})$ in Eq. (2.37). It is expected that, for an iterative, non-adaptive automatic integrator using the convergence criteria of Eq. (1.7), $\phi(x;\varepsilon_{quad})$ will have the property $\phi(x;\varepsilon_{quad,2}) \geq \phi(x;\varepsilon_{quad,1})$ for $|\varepsilon_{quad,2}| \leq |\varepsilon_{quad,1}|$ for all x. Due to round-off errors, however, this is not the case when ε_{act} approaches the machine epsilon[13]. Despite this, as most $\phi(x;\varepsilon_{quad})$ for $\varepsilon_{quad} \in \langle \varepsilon_{quad,1}, \varepsilon_{quad,2} \rangle$ will lie between the plotted curves of $\phi(x;\varepsilon_{quad,1})$ and $\phi(x;\varepsilon_{quad,2})$ the curves plotted in Fig. 2.8 are chosen to indicate how the statistical distribution function looks for a large range of ε_{quad}. The statistical distribution functions overlap in large x-ranges, for a large range of requested accuracies. Few curves are therefore included.

Statistical distribution functions have been generated to evaluate the integrator routine GPat's performance of approximating the two problem families for the smooth and the very rough surface systems defined in Tab. 2.1. To avoid an overwhelming amount of data, only selected figures that depict these results are included in Appx. D.2. The statistical distribution functions related to the problem family formulation of the $\mathcal{B}^{\pm}_{mn}$ matrix elements are not among them, as they are close to identical to those for the rough surface. This is not unexpected, considering the similarity of the integrand functions of these matrix elements for the three test systems.

[13]Recall that machine epsilon is $1.19 \cdot 10^{-7}$ for single precision and $2.22 \cdot 10^{-16}$ for double precision, in Fortran90 at the computer where the experiments are conducted

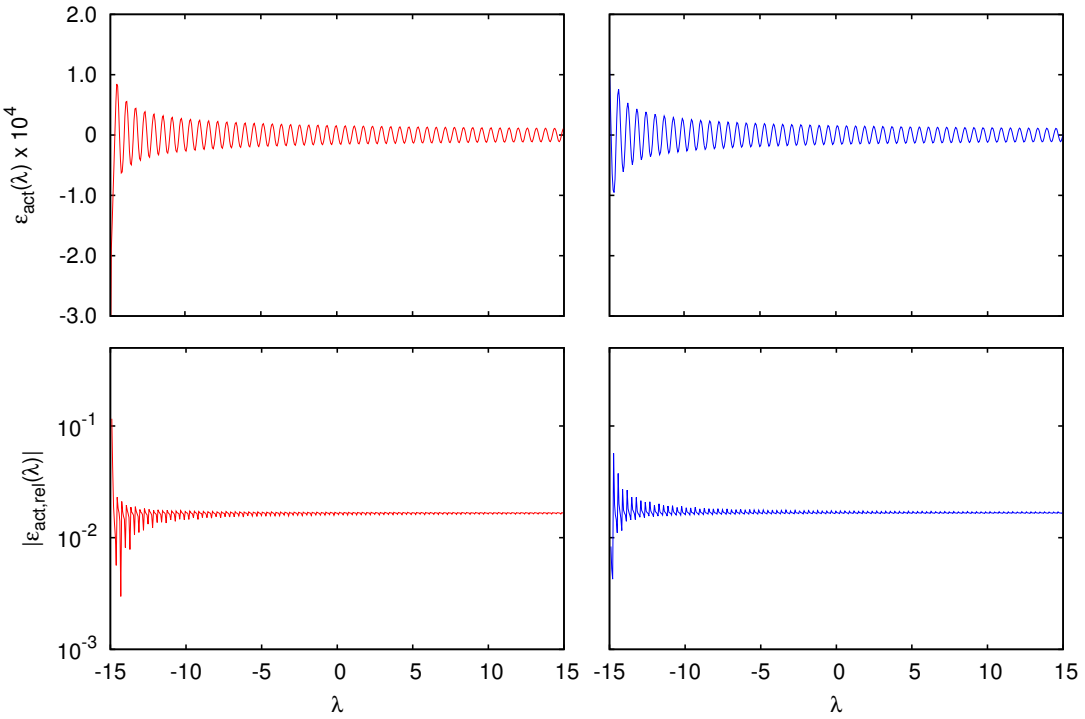

Figure 2.7: Performance profiles for the midpoint rule, for the problem family defined by Eq. (2.41), with the same properties as Fig. 2.6.

The average function value count, $n_{eval}(\varepsilon_{quad})$, used by GPat to compute the integrals in the problem family defined by Eq. (2.40), with the rough surface system, can be seen in Fig. 2.9. The corresponding results for the problem family defined by Eq. (2.40) are included in Appx. D.2. The requested absolute accuracy is set to zero for all the computations. It can be seen that the average function value count used by GPat is very similar in the two problem families tested. The variations depending on the surface profile used are also very small. Figures depicting $n_{eval}(\varepsilon_{quad})$ for the smooth and the very rough surfaces in Tab. 2.1 are not included, as they are qualitatively equal to Figs. 2.9 and D.9.

2.4.3 GPat integration with different surface representations

The results for the statistical distribution function with an analytic surface representation serves as a reference for comparison—as the theoretical best statistical distribution functions that may be achieved with the GPat for a given surface profile. The generated distribution functions for approximated surface representations are more interesting, as they are the results that are achievable with GPat in the Maxwell Eqs. solver (where the analytic surface representation is not available). Similar to the results plotted in Figs. 2.8 and D.8 many requested relative accuracies generate overlapping curves for $\phi(x, \varepsilon_{quad})$, for the different surface representations. The statistical distribution function resulting from GPat when using a particular surface approximation is, however, not necessarily overlapping with the results computed with different surface representations.

Figure 2.10 depicts the statistical distribution function for the real part of the problem families defined by Eqs. (2.40) and (2.41), with the rough surface profile from Tab. 2.1, computed with GPat. Both single and double floating point precision results are included. The single point precision results are computed with $\varepsilon_{quad} = 10^{-4}$, while the double

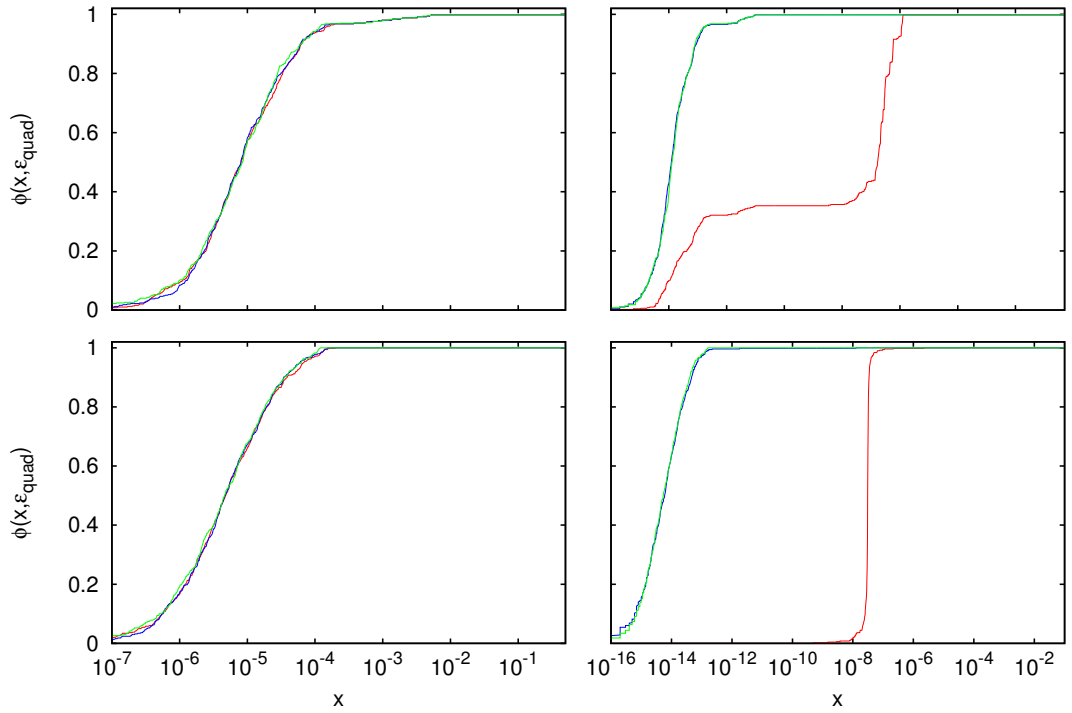

Figure 2.8: Statistical distribution functions for GPat, for the real part of the problem families defined by Eqs. (2.40) (top) and (2.41) (bottom), for the rough surface system in Tab. 2.1. Single precision results (left) given for ε_{quad} equal to 10^{-1} (red), 10^{-3} (blue) and $1.19{\cdot}10^{-7}$ (green). Double precision results (right) given for ε_{quad} equal to 10^{-1} (red), 10^{-2} (blue) and $2.22 \cdot 10^{-16}$ (green). All curves overlap to some extent in the single precision results, while the results for the two most demanding accuracy requirements overlap in the double precision results.

precision computations use $\varepsilon_{quad} = 10^{-10}$. Again, the accuracy $\varepsilon_{quad} = \varepsilon_{rel}$, while the absolute accuracy is set to zero. The singularity placement is $x_m = -14.97$.

The differences between the results plotted in Fig 2.10 and the corresponding results for the imaginary parts of the problem families considered are negligible. These results are therefore not included. The statistical distribution function for the smooth and the very rough surface systems, on the other hand, have some qualitative differences from the ones in Fig. 2.10. Figures depicting these results are included in Appx. D.2.

The average function value count related to the results plotted in Figs. 2.10 is barely impacted by the choice of surface profile approximation and problem family. The single precision computations yield the results $n_{eval}(10^{-4}) \approx 8$ and $n_{eval}(10^{-4}) = 7.0$ for all surface representation, for the problem families defined by Eq. (2.40) and Eq. (2.41), respectively. For the double precision computations $n_{eval}(10^{-9}) = 15.0$ for all four surface representations, for both problem families.

2.4.4 Errors in the surface approximations

As the test surfaces in Tab. 2.1 are all deterministic surfaces, where the vertical variations are given as a function of the x_1-coordinate, by $\zeta(x_1)$, an error bound in each surface approximation used can be calculated. An upper bound for Taylor approximated surfaces can be computed by the remainder term in Taylor's theorem [4]. Similarly, an error

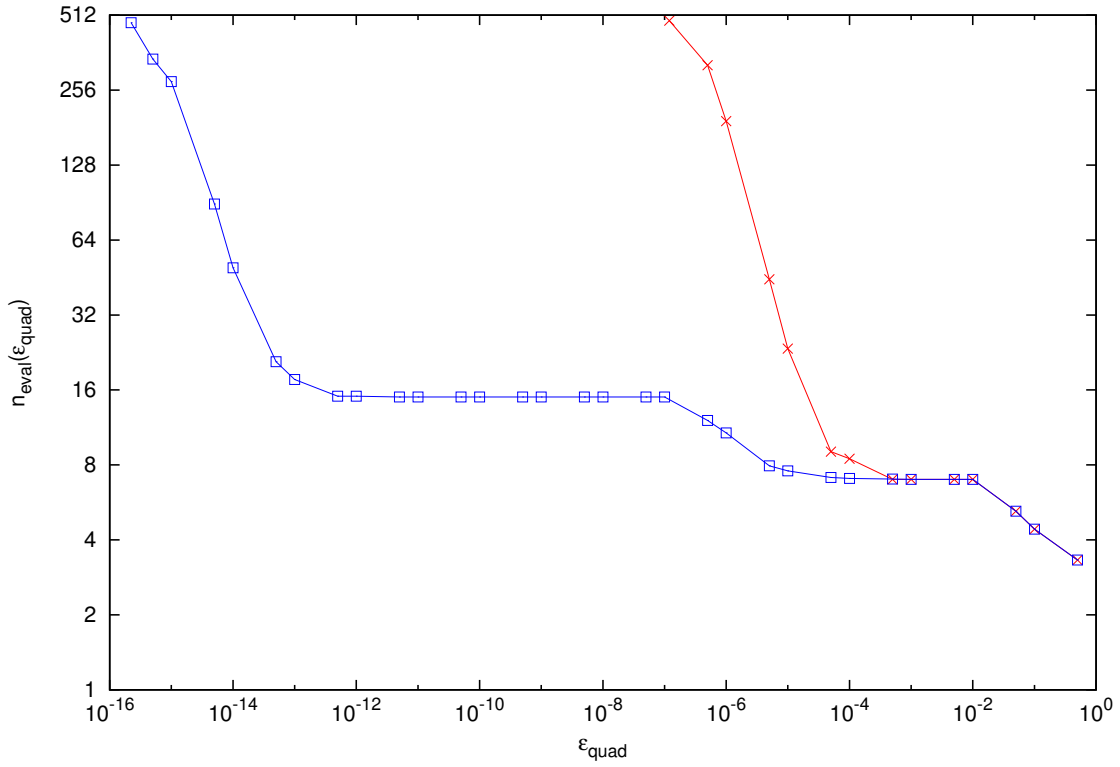

Figure 2.9: The average function value count as a function of requested relative accuracy, $\varepsilon_{quad} = \varepsilon_{rel}$ (as $\varepsilon_{abs} = 0$), for the problem family in Eq. (2.40) computed with GPat, for the rough surface system defined in Tab. 2.1. Single precision (red, x) and double precision (blue, □) results.

function exists for the generalized Hermite polynomials constructed. The error bounds for the surface approximations are given by[14]

$$E_T(x) = \frac{\zeta^{k+1}(c)}{(k+1)!}(x-a)^{k+1}, \tag{2.48}$$

$$E_H(x) = \frac{\zeta^{d}(c)}{d!}\prod_{i=1}^{n}(x-x_i)^{m+1}, \tag{2.49}$$

where E_T and E_H is the error bound for Taylor and Hermite approximations, respectively. The order of the Taylor expansion is k. The number of points, x_i in the Hermite polynomial is n, where each point has m derivatives. The integer d is the sum of points and derivatives in total, hence, $d = n(1+m)$. The Taylor series is constructed around the point a. The high order derivatives of $\zeta(x)$ is evaluated in c, where $a \leq c \leq x$ in Eq. (2.48) and $x_0 \leq c \leq x_n$ in Eq. (2.49).

By choosing c and x to maximize $|E_T|$ and $|E_H|$, upper error bounds for the approximated surface coordinates and first order derivatives can be computed. These are presented in Tab. 2.2.

[14] E_H in Eq. (2.49) is not a generalized expression for the error in an osculating polynomial, but valid with the notation used here.

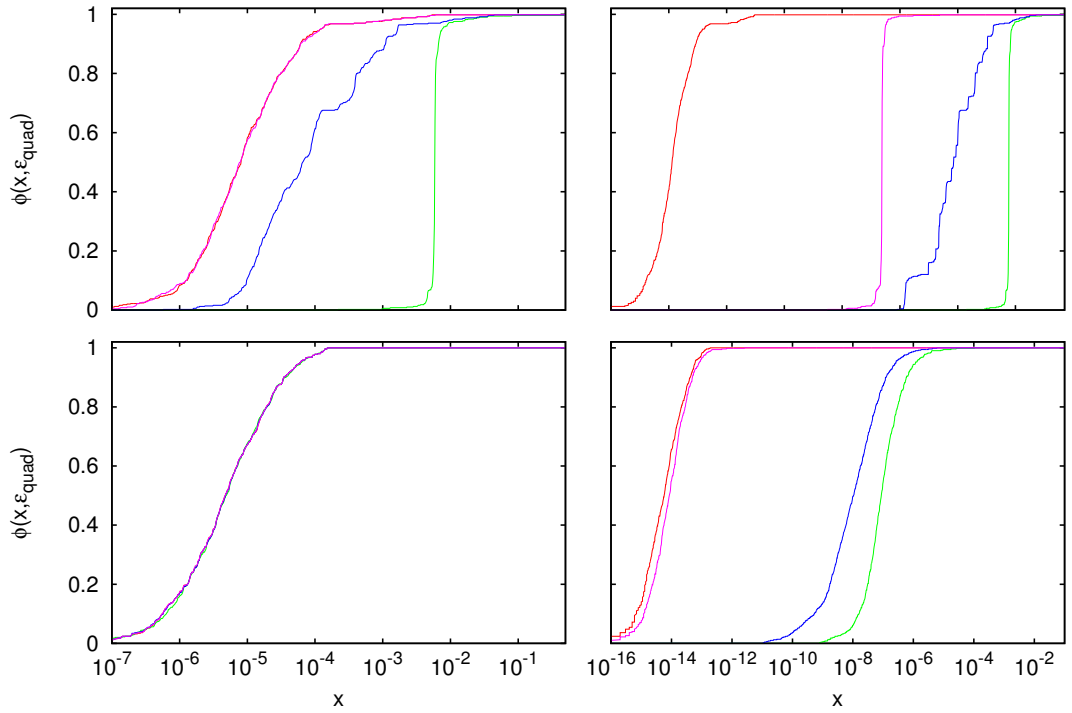

Figure 2.10: Statistical distribution functions for the real part of the problem families defined by Eqs. (2.40) (top) and (2.41) (bottom) computed with GPat, for single precision (left, $\varepsilon_{quad} = 10^{-4}$) and double precision (right, $\varepsilon_{quad} = 10^{-10}$). The rough surface profile function (see Tab. 2.1) is represented analytically (red), by second order Taylor approximation (green), by third order Taylor approximation (blue) and by Hermite interpolation (pink). The Hermite interpolated surface representation result overlap with the analytic results in the top left figure. All curves overlap in the bottom left figure.

2.4.5 Rule evaluation routine performance

Experiments have been conducted for the rule evaluation quadrature routine GPatREQR's evaluation of the problem families in Eqs. (2.40) and (2.41) for the test systems listed in Tab. 2.1. The position of the observation point has not been altered ($x_m = -14.97$). Figure 2.11 depicts performance profiles for GPatREQR when called with $n = 3$, for the real part of the problem family defined by Eq. (2.40), with the rough surface system. The performance profile of the rule evaluation routine with the three different approximation of the surface profiles are shown, as is the profile for an analytic surface representation. Single floating point precision is used in the computations, and only relative actual error plots are included in the figure. The corresponding performance profiles for $n = 7$ with double precision are given in Fig. D.5 in Appx. D.1. In addition, performance profiles for single precision and three integration points, generated for the real part of the problem family defined by Eq. (2.41) are included in Appx.D.1.

The results in Fig. 2.11 are rather messy. A more appropriate format of the data, increasing both readability and the amount of information in each figure is, once again, the routine's statistical distribution functions. As there is no requested accuracy in GPatREQR, the statistical distribution functions are a lot simpler than the ones generated for GPat. For each part of the problem families in Eqs. (2.40) and (2.41) one curve for $n = 3$ and one curve for $n = 7$ for each surface representation is sufficient to fully describe the

Table 2.2: Upper error bounds for the approximated x_3-coordinates and first order derivatives, with the different surface representations. The surface profile functions are from Tab. 2.1, and the wavelength $\lambda = 0.600\mu$m.

Surface	$\zeta(x_1)/\zeta'(x_1)$	2. order Taylor $\lvert E_T\rvert_{max}$	3. order Taylor $\lvert E_T\rvert_{max}$	Hermite int. $\lvert E_H\rvert_{max}$
Smooth	$0.0001x_1^2$	0	0	0
	$\frac{\mathrm{d}}{\mathrm{d}x_1}0.0001x_1^2$	0	0	0
Rough	$0.1\lambda\sin(2\pi x_1)$	$6.7\cdot10^{-4}$	$3.2\cdot10^{-6}$	$1.4\cdot10^{-12}$
	$\frac{\mathrm{d}}{\mathrm{d}x_1}0.1\lambda\sin(2\pi x_1)$	$6.7\cdot10^{-3}$	$4.3\cdot10^{-4}$	$2.2\cdot10^{-7}$
Very rough	$\sin(x_1)$	$4.5\cdot10^{-6}$	$3.4\cdot10^{-8}$	$1.5\cdot10^{-18}$
	$\frac{\mathrm{d}}{\mathrm{d}x_1}\sin(x_1)$	$4.5\cdot10^{-4}$	$4.5\cdot10^{-6}$	$9.2\cdot10^{-12}$

routine's performance for each test system. Figures 2.12 and 2.13 depict such results for the real part of the problem families defined by Eqs. (2.40) and (2.41), respectively. The surface profile is again rough, and the singularity placement is the same as in Fig. 2.11. The figures depicts result for $n = 3$ and $n = 7$, for both single and double precision. Distribution function for surface profiles represented analytically, by Taylor approximations and by Hermite interpolation are included. In addition a single curve that represents the statistical distribution function for the midpoint rule is included to simplify comparing the routines. The corresponding results for the imaginary part of Eqs. (2.40) and Eq. (2.41) are not included, as the difference between these results and those depicted in Figs. 2.12 and 2.13 are negligible.

The performance of GPatREQR is affected by the surface profile in the test system. The statistical distribution function for the integrator routine for the smooth and the very rough surface systems in Tab. 2.1, for the same integrals and parameter configurations as those in Fig. 2.12 and 2.13 are included in Appx. D.2.

2.4.6 Singularity computations

The statistical distribution function format is suitable for comparing the potential improvement of computing the diagonal matrix elements by Eqs. (2.46) and (2.47). Figures 2.14 and 2.15 depict the success probability in reaching an accuracy x, when computing the diagonal matrix elements $\mathcal{A}^{\pm}_{mm}$ and $\mathcal{B}^{\pm}_{mm}$, respectively, by these equation with $\delta = 100\epsilon_{machine}$ (with $\epsilon_{machine}$ denoting the floating point precision dependent machine epsilon). The results computed with an analytic surface representation have been computed with the automatic integrator routine, with $\varepsilon_{rel} = 10^{-4}$ and 10^{-10} for single and double precision computations, respectively. The results for approximate surface representations have, on the other hand, been computed with GPatREQR with seven integration points for each of the two integrals in each of the diagonal elements approximations $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$. The results computed directly with Eqs. (2.25) and (2.26) are also included in the figures.

The corresponding results, with three integration points in the rule evaluation routine GPatREQR, are included in Appx. D.3.

As two integrals are computed in each evaluation of the elements $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$, the average function value count may exceed 511—for the curves computed with GPat and

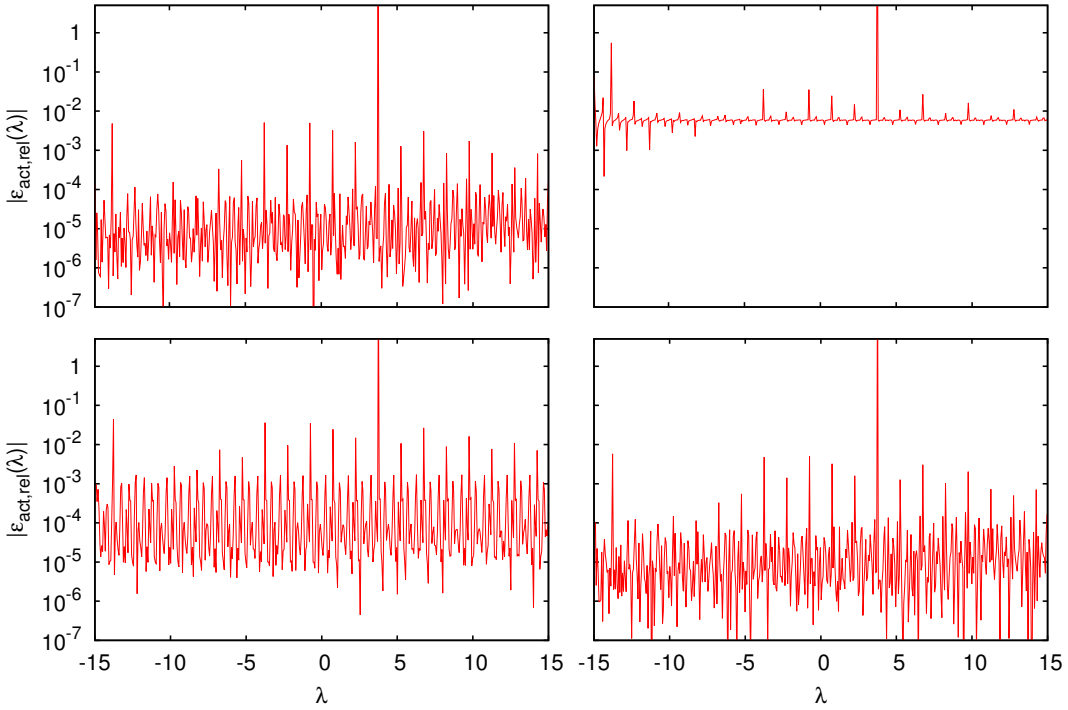

Figure 2.11: Performance profiles for the rule evaluation quadrature routine GPatREQR with three integration points ($n = 3$) and single floating point precision, for the real part of the problem family defined by Eq. (2.40). The test system is the rough surface system in Tab. 2.1, with and $x_m = -14.97$. The performance profiles are given for surface representations: analytic (top, left), second order Taylor approximation (top, right), third order Taylor approximation (bottom, left), hermite interpolation (bottom, right)

an analytic surface representation. This is indeed the case. For the computed curves in Fig. 2.14 the average function value count is $n_{eval}(10^{-4}) = 949$ (single precision) and $n_{eval}(10^{-10}) = 738$ (double precision). The average function value count related to the results in Fig. 2.15 are $n_{eval}(10^{-4}) = 126$ and $n_{eval}(10^{-10}) = 1022$.

2.5 Discussion – appropriate specializations and error estimate breakdown

The integral equation method in rigorous numerical simulation of light scattering from rough surfaces involve computing a large number of definite integrals that cannot be solved on closed-form. Numerical computation of these integrals have been done with several integrator routines, for deterministic surfaces of different roughness. As the scattering surface is discretized with only one grid point in each integration interval, a surface approximation outside the grid points is necessary to compute the intervals with all but the simplest quadrature scheme (the midpoint rule). Several surface approximations have been used during testing.

2.5.1 Integrand functions and midpoint rule integration

The integrand functions in the numerical simulation may be divided into two classes, containing the zeroth and the first order Hankel function of the first kind, respectively,

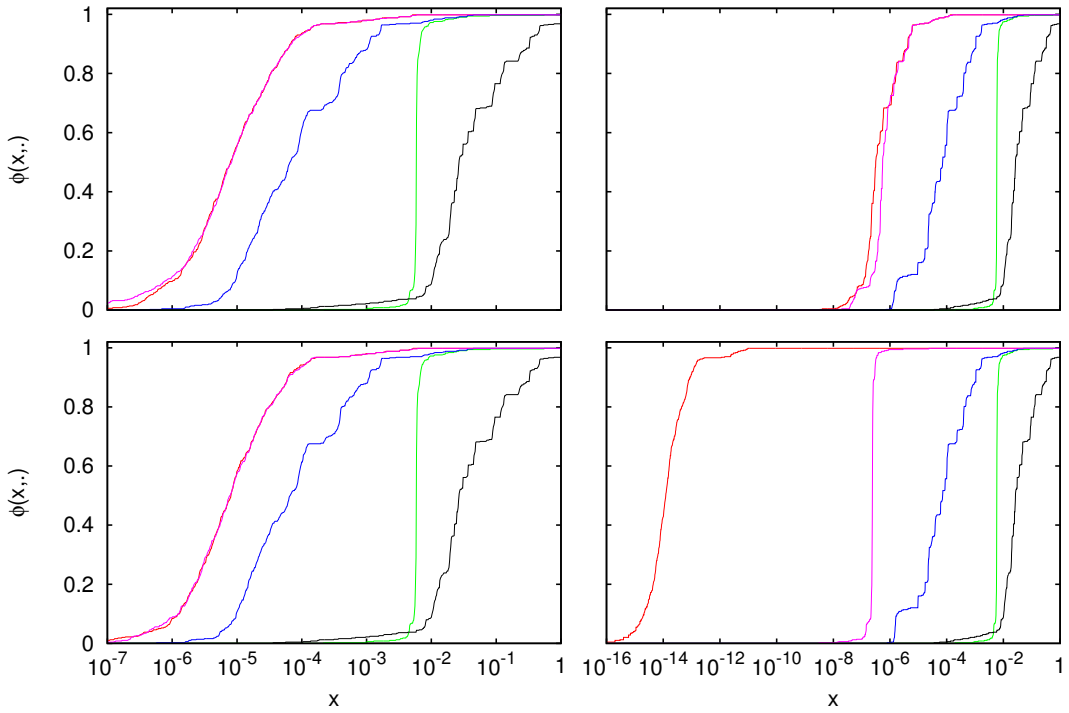

Figure 2.12: Statistical distribution functions for GPatREQR with $n = 3$ (top) and $n = 7$ (bottom), for the real part of the problem family defined by Eq. (2.40), with the rough surface test system in Tab. 2.1 and $x_m = -14.97$. Single precision (left) and double precision (right) results are shown. The curves are results for surface representations: analytic (red), second order Taylor approximation (green), third order Taylor (blue), hermite interpolation (pink). In addition the midpoint rule's distribution function is included (black). The red and pink curves overlap, to a large extent, in all but the bottom right frame.

and are highly oscillatory when considered on a length scale comparable to the size of the scattering surface. However, Figs 2.5 and C.4 show that the integrands—though not quite linear—are smooth when considered on an integration interval scale. The functions are indeed less oscillatory than the ones used for battery experiments in the development of the automatic integrator GPat, in Chap. 1, despite the surface profile being rough.

In order to perform parametric studies of the integrator routines, the integrals from the Maxwell Eqs. solver have been re-written to a problem family formalism. Figures 2.6–2.7, D.1–D.2 and D.3–D.4 depict performance profiles for midpoint rule integration of the problem families defined by Eqs. (2.40) and (2.41) for a rough and a smooth and a very rough surface profile, respectively. The results are not in favor of the midpoint approximation, especially not for the numerical integration of the integrals with the rough surface profile.

The performance profiles in Fig. 2.6 indicate that, even though the error in the approximation abate as the integration is performed farther away from the observation point, the relative error does not. Among the rightmost integration intervals there are several peaks where the actual relative error is larger than one[15]! Most λ-values do, however, correspond to approximations with less error than this. This can be seen from the statistical distribution function of the midpoint rule integrator, in Fig. D.7. The probability of the

[15]This value is affected by the actual result being close to zero, at the particular interval, making small absolute deviations very large on a relative scale

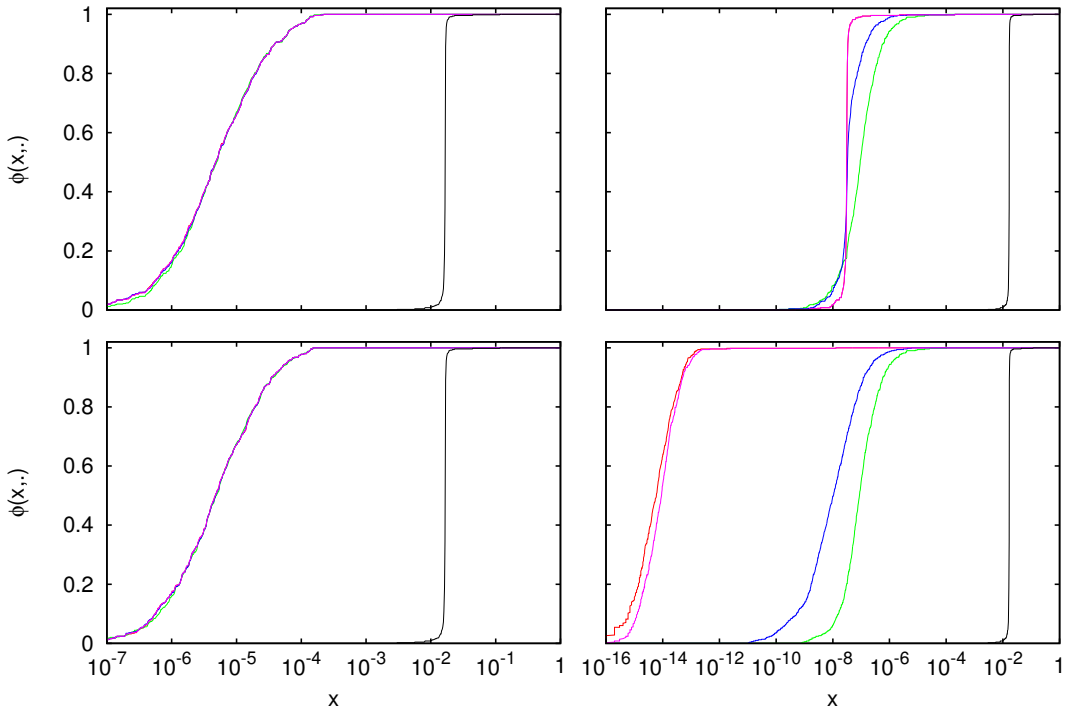

Figure 2.13: Same as Fig. 2.12, but for the problem family defined by Eq. (2.41). The curves for all surface representation overlap in the single precision results (left), and the Hermite interpolation and the analytic representation overlap in the right frames.

approximated result being computed with a relative accuracy $< 10^{-1}$ is $\approx 80\%$ for the $\mathcal{A}^{\pm}_{mn}$ matrix elements, for the rough surface system in Tab. 2.1. The probability is higher for the test systems with the smooth and very rough surface profiles.

The midpoint rule integration of the problem family defined by Eq. (2.41) is much better. The results are not only more accurate, they are also much more stable—with regards to expected output accuracy. As seen in Figs. 2.7, D.2 and D.4, the accuracy of the result is hardly effected by the surface profile in the system. As expected, when considering the integrand functions in Fig. 2.4 and C.1, the results are least accurate close to the observation point. The result seems to converge at a relative accuracy $\varepsilon_{act,rel} \approx 2\%$ when the distance between the integration interval midpoint and the observation point is $> 5\mu$m, for the tested parameter configurations.

Regardless of the surface profile used in testing, and which of the two problem family considered, the probability of computing an integral with better relative accuracy than than 10^{-2} is close to zero. The main advantage of the midpoint scheme (besides, perhaps, that it is a trivial integrator to program) is that it is very cheap to compute. This efficiency does, however, come at a high price. With a relative error $\varepsilon_{act,rel} > 10^{-2}$ for the majority of the computed integrals, and with peaks of disastrous values for certain intervals, the midpoint rule integration is both inaccurate and unreliable.

2.5.2 Inefficiency of an automatic quadrature routine

Consider Figs. 2.8 and D.8 that depict the statistical distribution function for the automatic integrator GPat, for an analytic representation of the rough surface profile. These results constitute the theoretical best performance of the GPat integrator for this sur-

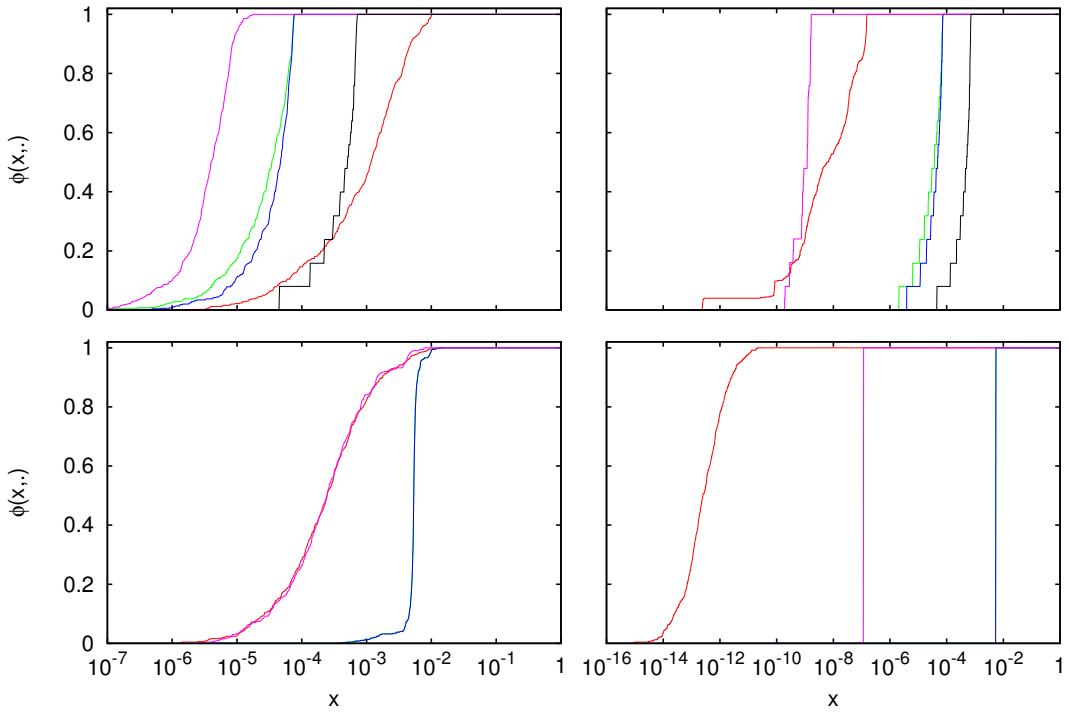

Figure 2.14: Statistical distribution functions for the computation of the diagonal matrix elements $\mathcal{A}^{\pm}_{mm}$, with different surface representations and integration routines. GPat is used with the analytic surface representation (red), GPatREQR with $n = 7$ for the second order Taylor (green), third order Taylor (blue) and Hermite interpolated (pink) surface representation. Equation (2.46) is used with these routines, while the direct method (black) computes the elements with Eq. (2.25). Single precision (left) and double precision (right), for the real part (top) and imaginary part (bottom) of the results. The blue and green curves overlap in the two bottom frames, as do the red and pink curve in the bottom left frame. The black curve is zero in the to bottom frames, for all x.

face, when implemented in the Maxwell Eqs. solver, as the surface will be represented by approximations rather than the analytic values in the simulation software. The plotted curves for selected relative accuracy requirements ε_{quad} show a peculiar behavior.

For the single precision results there is negligible differences in the success probability for reaching any accuracy requirement, when going from a requested relative accuracy $\varepsilon_{quad} = 10^{-1}$ to $\varepsilon_{quad} = 1.19 \cdot 10^{-7}$. The efficiency of the routine, on the other hand, is significantly reduced when increasing the accuracy requirement (Figs. 2.9 and D.9). The number of function evaluations needed to reach convergence increases steeply as ε_{quad} approaches the machine epsilon, without noticeable impact on the accuracy of the computed integrals. The success probability is close to one for $x > 10^{-4}$ but below 0.2 for $x > 10^{-6}$, for all requested accuracies. Hence, the error estimate is overly cautious for the less strict requested accuracies, yet unable to reach a satisfying results when high accuracy is requested, at single precision. The single precision efficiency can be drastically improved by setting $\varepsilon_{abs} = 1.19 \cdot 10^{-7}$ as a fail-safe in the routine. The difference in the success probability is negligible, but the average function value count will be < 8 for all requested relative accuracies. This is, however, practically the same as setting $\varepsilon_{quad} \in 10^{-5}$–$10^{-3}$, as the computed integrals are of order 10^{-4}–10^{-2}.

The double precision results are similar, yet the statistical distribution function does

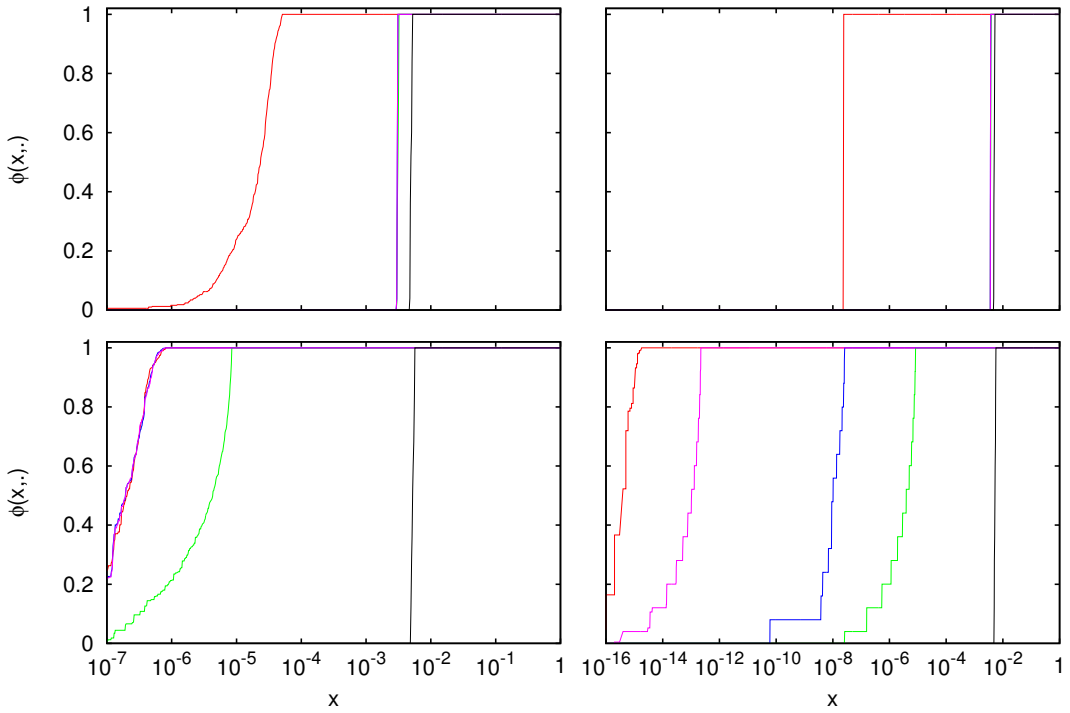

Figure 2.15: Same as Fig. 2.14, but for the diagonal elements $\mathcal{B}^{\pm}_{mm}$, computed by Eq. (2.47) with the Gauss-Patterson routines, and by Eq. (2.26) with the direct method. The blue, green and pink curves overlap in the top frames, as do the red, blue and pink cures in the bottom left frame.

change noticeably when ε_{quad} is reduced from 10^{-1} to 10^{-2} (but not if it is reduced further). The increased success probability at the stricter accuracy requirement is, however, not in the region where it is required. As the success probability is 1.0 for $\Phi(x, 10^{-1})$ for x-values as small as 10^{-6}, increasing the number of integration points to converge is not necessary to compute the integral sufficiently accurate by the automatic routine. An increase of the level of convergence should, ideally, not happen (on average) before the requested accuracy is of order 10^{-7}. This inefficiency is similar to the one discussed in Sec. 1.4.3, where the error estimate of the tested integrals required a surcharge of ≈ 2 for the computation with GPat. Of the integrals used in the battery experiments described in Chap. 1, the members of the problem families defined by Eqs. (2.40) and (2.41) are most similar to the sine integrals (integrals 1 and 2 in Tab. 1.1). These integrals were among the ones where the error estimate was indeed overly cautious, increasing the number of function evaluation long before it was necessary to reach the requested accuracy. This indicates that the error estimate in the automatic integrator is too strict for smooth integrand functions, noticeably impacting the efficiency when such integrals are computed.

The efficiency of the double precision computations is much better than that of the single precision computations when $\varepsilon_{quad} < 10^{-4}$. This is probably due to the occurrence of round-off errors at a much earlier stage in the single precision computations, impacted by the fact that the computed integrals are of the order 10^{-4}–10^{-2}. Similarly to the single precision efficiency, the double precision efficiency for very strict accuracy requirements ($\varepsilon_{quad} < 10^{-13}$) may be significantly improved, without effecting the accuracy of the output, by setting the requested absolute accuracy to the epsilon value ($2.22 \cdot 10^{-16}$ for double precision).

The results depicted by Figs. D.10 and D.11 are very similar to those in Fig. 2.8. This indicates that the above discussion is valid for the smooth and the very rough surface profile systems in Tab. 2.1 as well.

2.5.3 Error estimate breakdown

To be able to measure how the automatic quadrature routine will perform in Maxwell1D it is necessary to go beyond the theoretical results with an analytic surface representation. Each computed integral contains only the surface information in a single point, the midpoint. The information available in the grid point is the coordinates, and their first and second order derivatives. In order to compute the numerical integral with a more sophisticated quadrature scheme than the midpoint rule, it is necessary to evaluate the integrand function outside the midpoint of the integration interval. The matrix elements $\mathcal{A}_{mn}^{\pm}$ and $\mathcal{B}_{mn}^{\pm}$ given by Eqs. (2.16) and (2.17) require both of the coordinates of the surface in the x_1x_3-plane to evaluate the integrand function. In addition, the $\mathcal{A}_{mn}^{\pm}$-elements require the first order derivatives of the coordinates.

Several surface approximations have been tested. These include second and third order Taylor expansions for the coordinates (with related first and second order approximations of the derivatives) and Hermite interpolation (see Sec. 2.3.2 for details). The statistical distribution function curves in Figs. 2.10 and D.12–D.13 indicate that an upper bound on the x-value that give reasonable success probabilities is strongly affected by the surface representation in use. In the figures $\varepsilon_{rel} = 10^{-4}$ and 10^{-10} have been used for single and double precision, respectively, and $\varepsilon_{abs} = 0$. This choice for requested relative accuracy is rather arbitrary, as the statistical distribution functions of all the tested surface representations have similar characteristics as those discussed for an analytic surface profile. That is, the plotted curves would be close to identical for any other choices of $\varepsilon_{rel} \geq 10^{-1}$ for single precision, and $\geq 10^{-2}$ for double precision.

Consider the double precision results in Fig. 2.10. The statistical distribution function curves for the real part of the problem family defined by Eq. (2.40) illustrate the limitations in the surface representations explicitly. While the second order Taylor approximated surface coordinates are limited to having a high probability of reaching accuracies of $x > 10^{-2}$, the third order approximations are quite successful for $x > 10^{-3}$. The Hermite interpolated coordinates (and derivatives) allow the automatic integrator success probabilities of ≈ 1 for $x \geq 10^{-7}$. The actual errors in the computations are several magnitudes larger than the requested accuracy $\varepsilon_{quad} = 10^{-10}$, for all the surface representations for the problem family related to the matrix elements $\mathcal{A}_{mn}^{\pm}$.

The results obtained for the smooth surface system are fundamentally different. The close to flat surface simplifies the approximation of the surface coordinates and derivatives greatly. It can be seen in Fig. D.12 that even the simplest surface representation—the second order Taylor approximation—reaches the theoretical maximum accuracy in this case, for both single and double precision, for both of the two problem families studied. The results for the very rough surface are qualitatively more similar to those for the rough surface. Figure D.13 shows that the different surface representations yield different accuracy of the output (at least for the double precision computations). These results are, however, not as grim as those for the rough surface system, as the second and third order Taylor expansions have success probabilities of approximately 1 for $\varepsilon_{quad} = 10^{-3}$ and $\varepsilon_{quad} = 10^{-4}$, respectively. The Hermite interpolated surface representation allow for computations with success probabilities comparable to those achieved with the analytic surface representations, for the very rough surface system.

The double precision computations of the problem family defined by Eq. (2.40), with

the rough surface in Tab. 2.1, represent the most extreme impact of the limitations in the surface approximations among the generated distribution functions. The statistical distribution function generated for the problem family defined by Eq. (2.41) show some of the same properties, the Taylor approximations being the weaker surface representations and the Hermite interpolation closer to the analytic surface representation, but in a far less extreme way. For double precision computations of the rough surface system the second and third order Taylor approximations have success probabilities of ≈ 1 for $x \geq 10^{-6}$ and $x \geq 10^{-7}$, respectively, and the Hermite interpolated results are close to those of the analytic surface representation. The single precision results show that all surface representations are equal to the analytic results for the problem family defined by Eq. (2.41) in all three test configurations. There is negligible differences between the actual accuracy of the real and imaginary parts of the result.

What is observed in the statistical distribution function plots generated for the different surface representation is a breakdown of the error estimate's relation to the actual error. That is, a point where the automatic integrator routine fails to be reliable with respect to the requested accuracy. In the extreme cases, with second order Taylor approximated coordinates in computing the matrix elements $\mathcal{A}_{mn}^{\pm}$ for the rough surface, the integrator is limited to return results within the user requested accuracy for a $\varepsilon_{rel} \geq 10^{-2}$—regardless of whether single or double floating point precision is used. This is a result of errors in the input to the integrand function (as the coordinates are not sufficiently accurate), which result in erroneous integrand function evaluation. Beyond resulting in a limited accuracy of the result, this may wreck the efficiency of an automatic integrator routine, as mentioned in Sec. 1.1.5. The automatic routine increase the number of function evaluations without an increase in the numerical integral accuracy. Lyness [18] argues that while inaccurate function evaluations will limit the accuracy of a result obtained with a rule evaluation routine as well as with an automatic quadrature routine, automatic integrators may in addition suffer from disastrous efficiency behavior. The average function value count of the automatic integrator is very similar to that in Fig. 2.9, regardless of the surface representation. In the extreme cases, say, for $\varepsilon_{rel} = 10^{-6}$ requested in the single precision computation of $\mathcal{A}_{mn}^{\pm}$ with a second order Taylor approximated surface, this means that GPat computes the integral with 511 integration points, arriving at the same accuracy it would reach with 3 points. This is a surcharge factor of 170, and a grand argument for setting a minimum $\varepsilon_{abs} \neq 0$ as a fail-safe in the routine.

The source of error is not statistical fluctuations in the integrand function, round-off errors due to strict accuracy requirements or convergence at a too low level of integration. Considering the accuracy performance of GPat for the analytic surface representation it is apparent that it is the approximated surface coordinates and derivatives that leads to a breakdown of the error estimate, as the error is calculated for a surface given by the approximation at hand, not the analytic surface profile. The limited accuracy of the computed integrals, for the different surface approximations, are related to the error bounds of each surface representation, as seen in Tab. 2.2. There is a strong correlation between the error bound and the statistical distribution function curves. This can be seen by considering, e.g., the Hermite interpolated derivatives of the surface profiles for a rough and a very rough surface, together with Figs. 2.8 and D.11. For the rough surface, the double precision results of the problem family related to the $\mathcal{A}_{mn}^{\pm}$ matrix elements falls sharply $x \simeq 3 \cdot 10^{-7}$ and the upper error bounds for $\zeta'(x_1)$ for this system is $2.1 \cdot 10^{-7}$. Similarly, for the same matrix elements for the very rough surface, the sharp fall in the statistical distribution function for the Hermite interpolated surface representation occur at $x \simeq 8 \cdot 10^{-11}$ while the upper error bound for the x_3-derivative is $9.1 \cdot 10^{-12}$.

As the problem family related to the $B_{mn}^{\pm}$ matrix elements does not contain the deriva-

tives of the surface coordinates in the integrand function, and since the upper error bounds of the coordinates themselves are several magnitudes below those of the derivatives, the performance of GPat for this problem family is much better for the different surface representations. The exception is for the smooth surface, where all approximated surface representations are exact. For this system the success probabilities are equal, regardless of the surface representation in use (as seen in Fig. D.12).

2.5.4 A simpler approach

Performance profiles for for the rule evaluation quadrature routine GPatREQR for single precision computations with $n = 3$, and double precision with $n = 7$, can be seen in Fig. 2.11 and D.5, respectively. The profiles are for the real part of the problem family defined by Eq. (2.40) with the rough surface test system in Tab. 2.1. In Fig. 2.11 it can be seen that the Hermite interpolated surface representation performs just as well as the analytic surface representation. The second and third order Taylor approximations are, on the other hand, not sufficiently accurate to reach an actual relative error average of order 10^{-6}, but rather averages around $\approx 10^{-2}$ and 10^{-4},respectively. In Fig. D.5 it can be seen that the theoretical performance of the rule evaluation technique is of order 10^{-15}–10^{-13} with as few as 7 integration points. This is not nearly achieved by the surface representations, which differ somewhat more in these computations, than in the single precision performance profiles. The performance profile figures are rather messy, but they highlight an important difference between the midpoint rule and the three or seven point Gauss-Patterson quadrature routine: That computing outputs that are quite stable, with regards to output accuracy, is not a task only achievable by an automatic integrator routine.

By considering the performance profiles in Fig. D.6—the single precision computations with $n = 3$ of the problem family defined by Eq. (2.41)—another difference between the simple and the sophisticated quadrature schemes is highlighted. The midpoint performance for computing the problem family members related to the matrix elements $\mathcal{B}^{\pm}_{mn}$ is very bad close to the observation point, and it stabilizes as the distance from the integration interval to the integration point increases. This is not the case for the GPatREQR routine. The performance profiles indicate that the variations in the output accuracy are quite random, along the surface, yet always remaining close to, or below, 10^{-4} regardless of the distance from the observation point.

Figures 2.12 and 2.13 depict statistical distribution functions for GPatREQR , for the two problem families and the rough surface system, with different surface representations, number of integration points and floating point precision. The plotted curves reveal that the three and seven point integration with the Gauss-Patterson quadrature scheme yield results with negligible differences, for the single precision computations. Thus, if the automatic integrator routine GPat convergences at a higher level than level one for these integrals, at single precision, it is most likely a waste of computational time. This argument holds for the smooth and the very rough surface systems as well, as seen from the distribution functions in Fig. D.14–D.17. As discussed, the reason for the automatic integrator routine converging at a too high level of integration is both related to the surcharge paid in relation to the convergence criteria, and the error estimate breakdown. When a large number of similar integrals are computed in a numerical software the surcharge cost in an automatic integrator routine will have increased impact on the software performance, favoring the use of a rule evaluation quadrature routine.

The double precision results are similar, depending on the surface representation in the computations. For the Taylor approximated surface profiles there is no accuracy

increase when going from three to seven integration points for the problem family related to the $\mathcal{A}_{mn}^{\pm}$ matrix elements, and a small increase for the problem family related to $\mathcal{B}_{mn}^{\pm}$, for the rough and the very rough surface profiles. For the smooth surface system in Tab. 2.1, the accuracy is substantially improved when the number of points is increased, as the theoretical best achievable results are reached at $n = 7$ (see Figs. D.14 and D.15). The integration using the Hermite interpolated surface representation does improve the accuracy significantly, when increasing the number of integration points to seven, at double precision, regardless of which surface system that is studied.

A useful aspects of the figures depicting the statistical distribution functions of the GPatREQR routine, is identifying at what number of function counts the upper bound of the accuracy, for each surface representation, is reached. Comparing Fig. 2.10 to Figs. 2.12 and 2.13 reveals that the statistical distribution function is not improved—for single precision computations—when going beyond three integration points, regardless of the choice of surface representations. The same goes for double precision computations, if a second order Taylor approximation is used to compute the surface coordinates. With a third order Taylor approximation the upper bound is reached at three integration points for computing the matrix elements $\mathcal{A}_{mn}^{\pm}$ and seven for $\mathcal{B}_{mn}^{\pm}$. The Hermite interpolated surface representation does, on the other hand, benefit from evaluating the integrand functions at seven point within the integration intervals, at double precision, for both problem families. The reason for this is that the upper bound on the magnitude of the accuracy, when Hermite interpolation is used, is much higher. The same behavior is seen for the very rough surface system, but not for the smooth surface system (where, as mentioned, all surface representation reach the theoretical max at $n = 3$ for single precision and $n = 7$ for double precision).

The rule evaluation quadrature routine evidently beats the performance of the automatic integrator routine by far. The accuracy of the results are the same, if care is taken when choosing the number of integration points, yet the reliability and efficiency is a lot better. The efficiency is improved as there is no surcharge related to the rule evaluation routine, and the routine avoids the disastrous behavior that GPat has for accuracy requirements close to the machine epsilon—where the number of integration points are increased rapidly, with no noticeable accuracy increase. In addition, the computational cost of termination criteria evaluation and error estimation is avoided, and the overhead related to memory allocation is significantly reduced (as much smaller tables of abscissae and weights are needed). The routine may be considered more reliable, as the accuracy and the computational cost is easier to predict, than for the automatic integrator, and since the routine does not make any promises regarding output accuracy. Promises that—as seen in this chapter—are hard to keep for an automatic routine when the arguments to the integrand function are not exact.

2.5.5 Diagonal matrix elements

Consider Figs. 2.14 and D.18, depicting the probability of a computed diagonal elements of $\mathcal{A}_{mn}$ having an accuracy $\geq x$, for different integrators and approximation methods. The real part of the diagonal elements computed directly with Eq. (2.25) (the scheme utilized in the midpoint integrator) is fairly accurate, at least much more so than the off-diagonal elements computed with the midpoint rule. It is, however, evident that the accuracy of the diagonal elements $\mathcal{A}_{mm}^{\pm}$ can be improved by several orders of magnitude by Eq. (2.46), and a suitable integrator routine.

A suitable routine is not, in this case, the automatic integrator GPat—at least in the computations with single precision. The real part of the output is, in fact, less accurate

when computed with GPat and Eq. (2.46). This is probably due to round-off errors in the result, occurring as the automatic integrator is unable to converge at a low level of integration (related to the troublesome behavior of the integrand function close to the singularity). The rule evaluation quadrature routine GPatREQR, on the other hand, approximates the diagonal elements more accurately than the midpoint scheme and GPat, whether three or seven integration points are used[16]. Again, the Taylor approximated surface representation yields just as good results for $n = 3$ as for $n = 7$ in GPatREQR.

The imaginary part of the diagonal elements is zero, when computed with Eq. (2.25). This is a crude approximation. Even though $\mathrm{Im}(\mathcal{A}^{\pm}_{mm}) \ll \mathrm{Re}(\mathcal{A}^{\pm}_{mm})$[17] the imaginary part of the diagonal elements is not more than one or two orders of magnitude smaller than the off-diagonal elements. The results are, as expected, more accurate when computed with GPat and GPatREQR. In this case, the automatic integrator results are the most accurate. Do, however, keep in mind that only GPat is computed with an analytic surface representation, thus having an unfair advantage over the rule evaluation routine.

The results in Figs. 2.15 and D.19 are quite different. In these figures the results from the automatic integrator routine are less impacted by round-off errors (the number of function evaluation is not as large when computing $S_{\mathcal{B}}$, for single precision, as when computing $S_{\mathcal{A}}$). A more alarming aspect of the results is that the accuracy of the real part of the diagonal elements $\mathcal{B}^{\pm}_{mm}$ is barely improved when they are computed with Eq. (2.47) and GPatREQR with Taylor approximated surface representations, rather than directly by Eq. (2.26). In fact, it can be seen in Fig. D.19 that the accuracy of the real part is worse, when the diagonal elements are computed with GPatREQR, with three function evaluations for each integral in Eq. (2.47), and with Taylor approximated surface. This is the first occurrence of the accuracy of the result achieved with the second order Taylor approximated surface representations being different for $n = 7$ and $n = 3$. It can be seen in, e.g., Fig. 2.4 that it is the real part of the integrand function in $\mathcal{B}^{\pm}_{mn}$ that has a singularity at the observation point. The integrand functions troublesome behavior close to the singularity is probably the cause of the increased number of function evaluations required to reach an accurate result.

The handling of the singularity in the diagonal elements $\mathcal{B}^{\pm}_{mn}$ can be compared to how the lower integration limit of integral 11 in Tab. 1.1 was shifted to the right, to avoid an endpoint singularity. The accuracy of the resulting integral, integral 12, was a lot better. It was, however, one of the integrals that required a lot of function evaluations to converge, even at quite low accuracy requirements of ($n_{eval} = 63$ for $\varepsilon_{rel} = 10^{-5}$ in Tab. 1.3). This might also be the case for the integrals in Eq. (2.47).

2.5.6 Choosing an appropriate integrator routine

Curves for the midpoint rule performance are included in the statistical distribution function plots of GPatREQR depicted by Figs. 2.12 and 2.13. It can be seen that by increasing the number of integration points from one to three, the accuracy of the result is increased by several orders of magnitude—even with the weakest performing surface approximation. These results are in strong favor of replacing the midpoint scheme with the rule evaluation quadrature routine. Comparing Figs. 2.6 and 2.7 with Figs. 2.11 and D.6 further argues for such a change, as the unstable and low accuracy of the midpoint rule output (especially for $\mathcal{A}^{\pm}_{mn}$) induces the need for a change, not only for integrals close to the singularity, but

[16] Three and seven integration points correspond to six and fourteen function evaluations, respectively, as two integrals are computed for each diagonal element.

[17] $\frac{\mathrm{Im}(\mathcal{A}^{\pm}_{mm})}{\mathrm{Re}(\mathcal{A}^{\pm}_{mm})} \approx 10^{-4}$–$10^{-3}$

integrals computed along the entire surface. For the $\mathcal{B}^{\pm}_{mn}$ matrix elements, the errors are largest close to the observation point. The relative error in the midpoint approximations does, however, not drop below 10^{-2} for integrals with integration intervals far from this point. If this would be an acceptable error in the computed integrals, it would be sufficient to improve the quadrature scheme for integrals close to the integration point, for these elements. This would, however, require an algorithm for selecting which integrals should be computed with the midpoint scheme, and which should be computed with a more sophisticated quadrature scheme.

It was expected that an automatic quadrature routine could do such a selection, eliminating the need for an algorithm that couples distance from the observation point, the roughness of the surface profile, and the number of integration points needed in the computations. As the efficiency of the developed automatic quadrature routine has not been satisfactory, and since a more sophisticated algorithm that chooses the appropriate number of integration points will increase the computational cost of the numerical integration, it is expected that the best performance will be achieved by using a rule evaluation quadrature routine based on GPatREQR for all integration intervals. The routine will be implemented in two versions in Maxwell1D. One version for medium accuracy and one for high accuracy requirements. The medium accuracy version will be an implementation of the Gauss-Patterson quadrature scheme with three points with a second order Taylor approximated surface profile. This version is expected to perform well in single precision calculations, as it improves accuracy with a minimal increase in the computational cost. The high accuracy version will be an implementation of the Gauss-Patterson quadrature scheme with seven integration points, using Hermite interpolation of the nearest neighbors to approximate the surface coordinates and derivatives. The high accuracy scheme is expected to perform well, with respect to accuracy in double precision computations. The computational cost is, however, expected to increase with this version, mainly because of the complexity of the computations related to the Hermite interpolated surface approximation.

The reason for using Hermite interpolation, and not the third order Taylor approximation in the high accuracy numerical integration is not only a matter of the impressive accuracy of the interpolated surface representation. It is also a matter of convenience. The third order Taylor approximation would require modifications in the data structures used in the Maxwell Eqs. solver, modifications that would increase the communication cost in the software—as more data is passed back and forth—regardless of which of the versions of integrator that is used at a certain time. The increased computational cost would therefore not be limited to the numerical integration part of the program, but in all instances where the grid point data structures are used. The modifications would also increase the memory allocated by the software. This is not a desirable option, as the software is already quite memory intensive.

The implemented numerical integrators will handle the intervals containing singularities as special cases. They will, however, not make use of $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$ defined by Eqs. (2.46) and (2.47) in doing this, but rather compute the diagonal elements directly by Eqs. (2.16) and (2.17). This is due to the unexpected performance of the GPatREQR routine for the $\mathcal{B}^{\pm}_{mm}$ elements. It is believed that the accuracy of the diagonal elements can be improved by being computed by $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$, but the cost of this improvement (and the magnitude of the accuracy improvement) requires further study before being implemented in the Maxwell Eqs. solver.

Up till now, the efficiency of a quadrature routine has been strictly determined by the number of function evaluations used by the routine. In this respect, the medium accuracy version implemented in Maxwell1D should be approximately twice as efficient as the high

accuracy version. This is, however, not expected to be the case, as the Hermite polynomials needed in the high accuracy version are quite expensive to compute, while the second order Taylor approximation are very inexpensive. To gain a more accurate understanding of the computational cost related to the different integrator routines, the wall-clock time of the Maxwell Eqs. software will be used as the computational cost measure when the routines are implemented in Maxwell1D.

Chapter 3

Application

After rigorous testing some early assumptions about the numerical computation of the integrals in the Maxwell Eqs. solver Maxwell1D have fallen through, as a simple rule evaluation quadrature routine performed way better than the automatic quadrature routine GPat. The results from the preceding chapter indicate that the rule evaluation quadrature routine—based on the Gauss-Patterson quadrature scheme with three and seven points, respectively—reaches the upper bound of the possible accuracy of the output. The upper bound is related to the non-analytic surface representation in the integrals, complicating the computations and reducing the accuracy of the result. The accuracy is closely related to how the surface points are approximated outside the midpoint of the integration interval, as a poor approximation of the surface coordinates negates the effect of increasing the number of function evaluations within the integration interval.

In this chapter the implementation of the rule evaluation quadrature routine in the Maxwell Eqs. solver is described and tested. The accuracy and efficiency of the changes in Maxwell1D are compared to the original version of the software, where the integrals are computed with the midpoint rule.

Following this introduction, theory related to randomly rough surfaces is presented, and a measure of accuracy of Maxwell1D is introduced. Further, experiments are conducted for light scattering from rough surfaces. Both dielectric and metallic interfaces with vacuum are tested. The results are discussed, and a closing section, with summary, concluding remarks and suggestions for further work, ends this final chapter of the thesis.

3.1 Randomly rough surfaces and unitarity calculations

In Chap. 2 the performance of several integrator routines was measured for the integrals computed in the Maxwell Eqs. solver. The computations where done for three deterministic surfaces, representing a smooth, a rough and a very rough surface, respectively. The light scattering simulation software may very well use such surfaces in the simulations, however, as rough surfaces that are found in nature are normally correlated randomly rough surfaces [31], this is the main area of application of Maxwell1D.

In this section a formal definition of a rough surfaces is introduced. Further, the statistical description of randomly rough surfaces is presented. The theory presented is limited to Gaussian random surfaces without reentrant behavior, where the terminology of [31] is, once again, used. Such a limitation is deemed necessary to avoid a too comprehensive theory section in this chapter. Only measures related to the experiments conducted with the improved numerical integration routines in Maxwell1D are introduced.

To be able to measure the accuracy of the numerical integrators in Maxwell1D the *unitarity* is introduced. This is a measurable output of the Maxwell Eqs. solver, which

is expected to be affected by an improved quadrature scheme. However, as numerical integration in the Maxwell Eqs. solver is merely one of many operations necessary to study light scattering from randomly rough surfaces (though, one of the major operations, at least in terms of computational cost), the unitarity is no absolute measure of performance. It is, however, expected to give an indication of the integrator performance, for better or for worse.

3.1.1 Characterizing surface roughness

The roughness of a surface is made up of two parts—the height of features above and below the mean surface level, and the lateral separation of these features [22]. Considered together with properties of the incident light, for nanoscale rough surfaces, the height variations along the surface defines whether the surface is smooth or rough, while the lateral separations determines whether the roughness is fine or coarse. It is convenient to define measures that provide limits to what is defined as a rough surface, and what separates fine and coarse roughness. Such measures do exist, but mainly has the function of indicating the roughness of the surface that is studied—there is no abrupt transition from a smooth to a rough surface.

Consider an effective one-dimensional rough surface defined by the vector-valued function $\boldsymbol{r}(x_1, \zeta(x_1))$. The specular phase difference between two electromagnetic waves being scattered at the points x_1 and x_1' at the surface is

$$\Delta\phi = 2|\boldsymbol{k}||\zeta(x_1) - \zeta(x_1')|\cos\theta_0, \tag{3.1}$$

where $|\boldsymbol{k}| = 2\pi/\lambda$ is the modulus of the wave vector of incident light of wavelength λ, and θ_0 is the angle of the incident light as measured from the normal to the mean surface. If $\Delta\phi \ll \pi$ the scattered waves will be in phase (or close to), and interfere constructively. If $\Delta\phi \simeq \pi$ they will be out off phase and interfere destructively, and less energy will be scattered into the specular direction as compared to the situation where the two waves interfere constructively [31]. This motivates the Rayleigh criteria, that defines a smooth surface in terms of the phase difference

$$\Delta\phi < \frac{\pi}{2}. \tag{3.2}$$

Thus, a rough surface has $\Delta\phi > \pi/2$. As mentioned, this limit is no absolute measure of roughness, and a surface with $\Delta\phi = \pi/4$ may very well be described as rough, e.g., if it is compared to a surface with $\Delta\phi \simeq 0$. The Rayleigh criteria first and foremost has the function of defining surfaces that have $\Delta\phi \ll \pi/2$ as smooth and $\Delta\phi \gg \pi/2$ as rough.

For randomly rough surfaces the height difference in Eq. (3.2) is replaced by a measure of the typical height fluctuation. An appropriate measure is the root-mean-square (rms) height, δ, in the statistical description of the surface. Hence, the Rayleigh criterion for a randomly rough surface can be expressed as [31]

$$R_a = |\boldsymbol{k}|\delta\cos\theta_0 < \frac{\pi}{4}, \tag{3.3}$$

where the Rayleigh parameter $R_a < \pi/4$ defines a random surface as smooth.

In the lack of a corresponding criteria to distinguish between fine and coarse roughness a similar expression to the one defined in [22] is used[1]:

$$a|\boldsymbol{k}|\sin\alpha \begin{cases} \ll 1, & \text{fine,} \\ \gg 1, & \text{coarse,} \end{cases} \tag{3.4}$$

[1]The refractive index, n_0, is left out of the expression, to simplify them for the cases treated in this chapter—light incident from vacuum (where $n_0 = 1$).

where a is the transverse correlation length of a randomly rough surface and the incident and scattering angle is limited within an angle α relative to the surface normal.

While the expressions in Eqs. (3.2)–(3.4) are measures to characterize surface roughness, it should be emphasizes, once more, that they are not absolute measures and do not constitute strict boundaries of what is a rough surface and what is not. Nanoscale surface roughness is a concept related to the properties of light scattered from the surface, e.g., the transition from specular to diffuse scattering for increased surface roughness (see, e.g., Ref. [31]), concepts that are beyond the scope of this thesis. The main importance of these expressions, as presented here, is to relate the concepts of roughness and coarseness to the measures of rms-height, correlation length and the wavelength of the incident light.

3.1.2 Statistical description of randomly rough surfaces

The choice of coordinate system is such that the surface profile function is planar on average, i.e.,

$$\langle \zeta(x_1) \rangle = 0. \tag{3.5}$$

It is assumed that the surface is *ergodic* [25] such that the average in Eq. (3.5) may be considered the ensemble average, a more convenient measure than the spatial average as many surface realizations are considered. It is further assumed that the surfaces studied are *stationary*, that is, that the statistical properties of the surface are independent of which portion of the surface that is used in their determination [31].

Under the assumption of stationary, for a surface with horizontal variation along the surface described by a single-valued function $\zeta(x_1)$, the height-height correlation function of the surface is

$$\langle \zeta(x_1)\zeta(x_1') \rangle = \delta^2 W(|x_1 - x_1'|), \tag{3.6}$$

where δ again denote the rms-height of the surface profile function, and $W(|x_1|)$ is the normalized height auto-correlation function [31].

For a Gaussian random surface all the statistical properties of the surface is available from Eqs. (3.5) and (3.6), as the higher order moments of its probability density function either vanish (odd moments) or are related to the second moment (even moments). The Gaussian auto-correlation function takes the form

$$W(|x_1|) = \exp\left(-\frac{x_1^2}{a^2}\right), \tag{3.7}$$

where a is the transverse correlation length [31]. The rms-height and transverse correlation length are the only two necessary properties to generate a correlated random surface of Gaussian type in Maxwell1D.

In order to get an intuitive picture of how the surface height varies along light scattering system the rms-slope, s, and the mean distance between consecutive peaks and valleys along the x_1-direction, $\langle D \rangle$, are useful. For a Gaussian random surface these properties are given by [31]

$$s = \sqrt{2}\frac{\delta}{a}, \tag{3.8}$$

$$\langle D \rangle = \frac{\pi}{\sqrt{6}} a. \tag{3.9}$$

3.1.3 Energy conservation

Related to the energy conservation of light scattered from rough surfaces, the scattering and transmission unitarities may be defined as [31]

$$\mathcal{U}_v^{sc}(\theta_0,\omega) = \int_{-\pi/2}^{\pi/2} \left\langle \frac{\partial R_v}{\partial \theta_s} \right\rangle \mathrm{d}\theta_s, \tag{3.10}$$

$$\mathcal{U}_v^{tr}(\theta_0,\omega) = \int_{-\pi/2}^{\pi/2} \left\langle \frac{\partial T_v}{\partial \theta_t} \right\rangle \mathrm{d}\theta_t, \tag{3.11}$$

where $\mathcal{U}_v^{sc}$ and $\mathcal{U}_v^{tr}$ express the fraction of incident light scattered from and transmitted through the surface, respectively[2]. The incident light, of frequency ω, incident angle θ_0 and polarization v, is scattered with angle θ_s and transmitted with angle θ_t. The differentials in Eqs. (3.10) and (3.11) are the differential reflection coefficient and the differential transmission coefficient, respectively. The scattered and transmitted fields, and their differential forms, can be computed numerically by the integral equation approach described in Sec. 2.1. For details the interested reader is once more referred to Refs. [23,31].

The link to energy conservation becomes clear from the total unitarity in the system, for a system where the media are non-absorbing ($\mathrm{Im}[\varepsilon(\omega)] = 0$):

$$\mathcal{U}_v(\theta_0,\omega) \equiv \mathcal{U}_v^{sc}(\theta_0,\omega) + \mathcal{U}_v^{tr}(\theta_0,\omega) = 1. \tag{3.12}$$

For practical purposes, the expressions for unitarity may be used as a measure of the quality of the numerical simulations in the Maxwell1D (or a similar light scattering simulation software). It should, however, be stressed that the condition in Eq. (3.12) is only a necessary condition for the simulations to fulfill, and its satisfaction does not guarantee that the simulations are correct [32].

3.2 Implementation and testing

Based on the results and discussion in the preceding chapters, two rule evaluation integrator versions of GPat are implemented in the Maxwell Eqs. solver. The versions are for medium and high accuracy, respectively, where the high accuracy version is expected to be considerably more computationally expensive than the medium accuracy version. These two versions are described briefly in this section. As are the experimental set-ups used in the numerical experiments performed with Maxwell1D. This includes relevant surface parameters, the random surfaces' statistical properties, and the system's numerical properties

3.2.1 Implemented quadrature routines

As mentioned in Sec. 2.5, the two integrator routines selected to be implemented in the Maxwell Eqs. solver are rule evaluation quadrature routines. The routines differ in the amount of integration points used to evaluate an integral and the techniques applied to approximated the surface coordinates outside the integration interval's midpoint. From here on out, the integrator routines will be referred to as *GPatFast* and *GPatAcc*, named

[2]Strictly speaking, the unitarity is a property possessed by a scattering medium that is a perfect reflector only. The notion of a transmission unitarity is more precisely a matter of energy conservation, which is a property of a (non-absorbing) transmitting and a scattering media alike.

Table 3.1: Dielectric properties of the media used in the numerical simulations.

Medium	ε
Vacuum	1.0
Glass	2.25
Silver	-18.0

such to reflect the quadrature scheme in use and the characteristic properties distinguishing the two.

The routine GPatFast uses three integration points to approximate an integral, and a second order Taylor approximation to compute the two points outside the grid point (the midpoint, with known coordinates and derivatives, is one of the three integration points). GPatAcc, on the other hand, uses seven integration points, and a highly accurate (yet computationally costly) Hermite interpolation scheme to approximate the surface coordinates. The computed interpolating polynomials are of order eight and five, used in computing the coordinates and their first order derivatives, respectively. Both GPatFast and GPatAcc are structured like the general rule evaluation quadrature routine of Alg. 1 in Chap. 1, yet with a more extensive set of operations for each iteration in the *for*-loop—related to approximating the surface profile. Both integrator routines are implemented to handle reentrant surfaces and applicable for both single and double precision computations.

To reduce the computational cost of the rule evaluation routines, the matrix elements $\mathcal{A}^{\pm}_{mn}$ and $\mathcal{B}^{\pm}_{mn}$ defined by Eqs. (2.16) and (2.17) are computed in the same integrator routine call, for each (m, n)-iteration of Maxwell1D. This reduces both the memory allocation in the computations and the expected computational cost of the Hankel function evaluation[3]. This action is possible since, with a rule evaluation routine, both integrals are guaranteed to be computed with the same amount of integration points.

The intervals containing singularities are handled outside the integrator routines. The previous implementation of these computations, directly calculating such elements by Eqs. (2.25) and (2.26), is not altered.

The source code of the routine GPatFast is included in Appx. A.2. Readability of the code is prioritized, rather than optimization for low computational cost, in the version included in the appendices.

3.2.2 Experiments and experimental set-up

Two different (numerical) experimental set-ups are used in the simulations that are run with Maxwell1D, with the new integrator routines implemented. The systems differ in the medium that scatters the incident light. System 1 is a vacuum-glass system, where transmission is the dominating effect. System 2 is a vacuum-silver[4] system, where all incident light is reflected. For simplicity, the magnetic permeability is set to 1.0 in all media. As is the imaginary part of the dielectric function, such that the media do not absorb any of the incoming radiation. The relative electric permittivity for each medium is given in Tab. 3.1.

In all but a few initial test runs, Gaussian random surfaces are used as interfaces between the media in the system. Thirty different surface realizations are generated for

[3]The Hankel functions are computed with the routines CBESH (single precision) and ZBESH (double precision) from the AMOS routines included in the SLATEC library. The routines compute a sequence of Hankel functions of different order [3]. The routines are presumably faster when called for two elements, once, than for one element, twice.

[4]The silver medium used is a simplified version of a real silver medium.

each surface parameter configuration, for the Gaussian surfaces. This is deemed sufficient to characterize the systems as ergodic. The properties of the these interfaces differ in the different experiments that have been conducted, by varying a single parameter in each experiment.

A few test runs introduce the result section in this this chapter. These runs are the light scattering simulations with the test systems of Chap. 2, as defined in Tab. 2.1. The deterministic surfaces are tested with both the system dominated by transmission and the system dominated by scattering. The test runs are for fixed parameter values, and will serve as a basis for the discussion to come—as these are the only systems where the performance of the integrator routines is known for certain.

As mentioned in [32], one of the uses of the energy conservation property in the system (expressed here by Eq. (3.12)) is to make sure that the chosen discretization interval is sufficiently fine. Experiments are performed to test how fine discretization is needed to get a unitarity sufficiently close to one in the Maxwell Eqs. solver, by starting at a small discretization size and increasing it stepwise, thus, increasing the grid's coarseness. It is expected that the new integrator routines implemented in Maxwell1D will allow for larger discretization sizes than the previous numerical integration did, as the integrals are significantly more accurately computed. If this is the case, the effect on computational cost by increasing the number of integration points may be negated by a reduced number of intervals of integration. This is due to the number of intervals of integration being related to the number of grid points, which is set at run-time by the relation $N = L/\Delta s$ (where L is the system width, for the case of equidistant sampling in x_1-direction). For each grid point $\Theta(N)$ integrals are computed in Maxwell1D—thus, the number of integrals will be reduced significantly if the discretization size is increased.

Another aspect of the performance of the Maxwell Eqs. solver expected to be improved with the new numerical integrators utilized is how the unitarity is affected by changing the roughness of the surface. Two types of experiments have been conducted to test for this. One by stepwise increasing the rms-height of the surface—going from a smooth to an increasingly rough surface—and one by increasing the correlation length of the Gaussian surfaces—reducing the coarseness of the rough surface.

It is expected that the computational time of the Maxwell Eqs. solver will increase significantly when going from a midpoint integration scheme to the Gauss-Patterson integrators (with the same surface discretization). This assumption is checked by performing several experiments and averaging the wall-clock time used by Maxwell1D, for each of the three integrator routine in use[5]. Such results are computed for different discretization intervals, to explore the possible efficiency boost of increasing the distance between each point on the discretized grid.

The parameters used in the four different experiments are listed in Tab. 3.2. Although the experimental differ, by varying one parameter in each experiment, they have several features in common. It can be seen that the parameters are varied about a system with incident light of wavelength $\lambda = 0.600\mu$m, system width $L = 30$mum, discretization size $\Delta s = 0.1\lambda$, rms-height $\delta = 0.1\lambda$ and correlation length 0.5λ. With a Rayleigh parameter of $R_a = \pi/5$ the surface of such a system is smooth (as $R_a < \pi/4$), yet close to the limit that defines a rough surface. The mean distance between consecutive peaks and valleys for the system is $\langle D \rangle \simeq 0.38\mu$m and the rms-slope is $s = 0.28$. By comparing this to the roughness characteristics of the test surfaces used in Chap. 2 in Tab. 3.3, it can be

[5]The wall-clock time will, off course, be affected by the work done in the Maxwell Eqs. solver, besides computing the matrix elements. This work is, however, constant regardless of the integration scheme in use, and the matrix element computations are expected to be the single most time consuming part of the software.

Table 3.2: Parameter configuration in experiments conducted with the Maxwell Eqs. solver Maxwell1D. Consists of the wavelength, λ, and angle of incidence, θ_0, of the incident light, the system width, L, the discretization size, Δs, and the rms-height, δ, and correlation length, a, of the Gaussian random surface. The incident beam is an s-polarized Gaussian beam of width $L/3$ for all experiments.

Experiment [#]	λ [μm]	θ_0 [°]	L [μm]	Δs [μm]	δ [μm]	a [μm]
1	0.600	0, 25	30	0.020–0.600	0.060	0.300
2	0.600	0, 25	30	0.060	0.002–1.000	0.300
2	0.600	0, 25	30	0.060	0.060	0.00001–1.000
4	0.600	0	30	0.030–0.300	0.060	0.0300

Table 3.3: Roughness characteristics of test surfaces used in Chap. 2, with variations in the x_3-coordinate defined by $\zeta(x_1)$ and λ in Tab. 2.1. Phase difference $\Delta\phi$ calculated by Eq. (3.2) for normal incidence, with x_1 and x_1' being coordinates of mean and maximum values of $\zeta(x_1)$, respectively.

$\zeta(x_1)$ [μm]	$\Delta\phi$ [rad]	$\langle D \rangle$ [μm]	s [-]
$0.0001x^2$	$\pi/20$	-	0.0015
$0.06\sin(2\pi x_1)$	$2\pi/5$	0.5	0.244
$\sin(x_1)$	$20\pi/3$	π	0.637

seen that the system in the experiments is very similar to the test system described as rough in Tab. 2.1[6]. Indeed, by considering Fig. 3.1, depicting the rough surface in Tab. 2.1 and the randomly rough surface described here, the similarity can be seen. However, the differences between a deterministic, sinusoidal surface and a Gaussian random surface with similar roughness characteristics are striking.

3.3 Maxwell1D results

The experiments described in the previous section have been conducted, and the results are presented here. In all but one of the figures and tables depicting the results, the results presented are for simulations performed with single precision in the computations utilizing the midpoint scheme and the integrator routine GPatFast, and double precision in the computations utilizing GPatAcc. Comparing these calculation may appear to be a comparison of apples and oranges, but the reader should be assured that this is not the case. The floating point precision have been chosen based on the analysis in Chap. 2, regarding the expected precision of the computed integrals—where only the integrator routine GPatAcc[7] could potentially make use of the floating point precision beyond single precision. These expectations have been tested, and confirmed.

The exception to this use of floating point precision is in the efficiency results, where the time used by Maxwell1D with the midpoint integrator, for both single and double

[6]It can be seen that this system is, in fact, not a rough surface system according to the definition in Eq. (3.2). It is, however, much more rough than the system described as smooth in Tab. 2.1, and it is the system with the fines roughness of the test systems.

[7]Referred to as GPatREQR with seven integration points and Hermite interpolated surface representation, in Chap. 2.

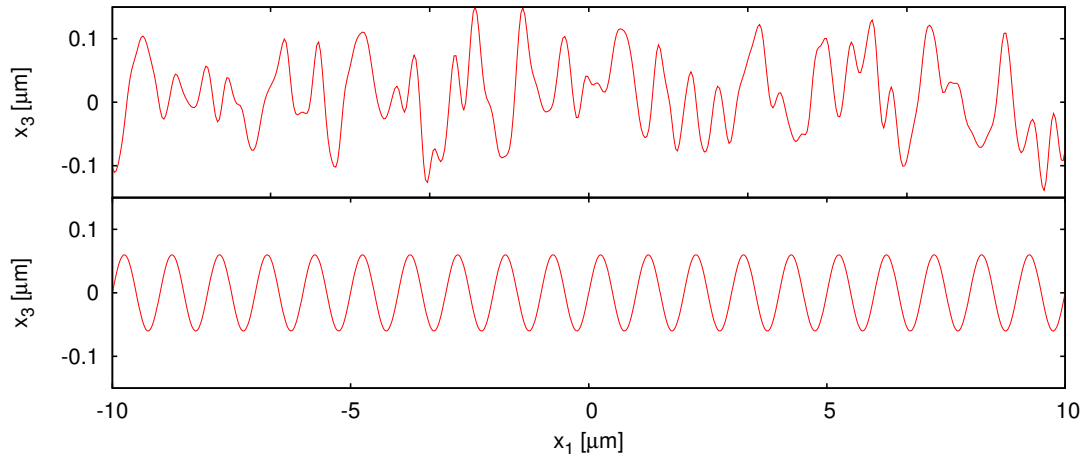

Figure 3.1: A Gaussian random surface with $\delta = 0.060\mu$m and $a = 0.030\mu$m (top), and a deterministic sinusoidal surface with x_3-variations given by $\zeta(x_1) = 0.06\sin(2\pi x_1)$.

Table 3.4: The deviation from the theoretical unitarity value (Eq. (3.13)), of the deterministic surface systems and parameter configurations in Tab. 2.1. Both the vacuum-glass (V-G) and the vacuum-silver (V-S) set-ups have been used in the light scattering simulations with Maxwell1D. Single precision used in midpoint and GPatFast integration, double precision used in GPatAcc integration.

Surface	Midpoint		GPatFast		GPatAcc	
	V-G	V-S	V-G	V-S	V-G	V-S
Smooth	$7.8 \cdot 10^{-4}$	$1.3 \cdot 10^{-2}$	$1.8 \cdot 10^{-3}$	$3.7 \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$	$3.7 \cdot 10^{-3}$
Rough	$8.9 \cdot 10^{-4}$	$1.3 \cdot 10^{-2}$	$1.4 \cdot 10^{-4}$	$2.4 \cdot 10^{-3}$	$4.2 \cdot 10^{-4}$	$2.5 \cdot 10^{-3}$
Very rough	$3.2 \cdot 10^{-3}$	$1.4 \cdot 10^{-2}$	$7.8 \cdot 10^{-3}$	$1.7 \cdot 10^{-2}$	$7.7 \cdot 10^{-3}$	$1.7 \cdot 10^{-2}$

precision, is included. This is done to allow for a study of the time increase due to the new quadrature schemes and surface representations—not due to the change from single to double precision (in the case of GPatAcc).

3.3.1 Accuracy

The results for the initial test runs of the Maxwell Eqs. solver, with the deterministic surface systems with horizontal coordinates given by $\zeta(x_1)$ in Tab. 2.1 can be seen in Tab. 3.4. The results have been generated for both the vacuum-glass and the vacuum-silver system. The parameter configurations in the Maxwell1D simulations are the same as those defined in Tab. 2.1—with the exception of the electric permittivity, which is dependent on the medium the source functions are computed in. The deterministic surfaces are referred to as smooth, rough and very rough, even though it can be seen in Tab. 3.3 that only one of the surfaces is rough by the definition in Eq. (3.2). To simplify readability, the results presented in Tab. 3.4 are the deviation of the unitarity computed with Maxwell1D and the theoretical value of the unitarity. This quantity, ΔU, is simply calculated by

$$\Delta U = |1 - \mathcal{U}_v|, \tag{3.13}$$

where $\mathcal{U}_v$ is the unitarity defined by Eq. (3.12).

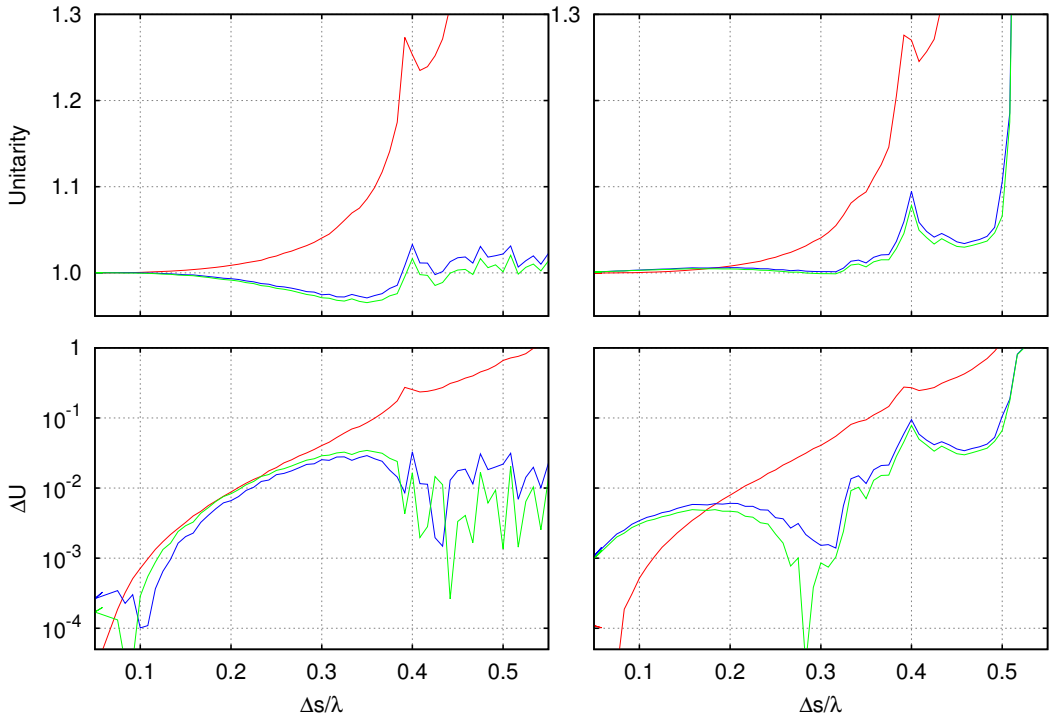

Figure 3.2: Unitarity of light illuminating the vacuum-glass system for increasing discretization size, as defined in experiment 1 in Tab. 3.2. The unitarity (top) and the absolute deviation from the theoretical unitarity value (bottom), as defined by Eqs. (3.12) and (3.13), respectively, depicted in the figures. The angle of incidence is $\theta_0 = 0^o$ (left) and $\theta_0 = 25^o$ (right). The results are given for the midpoint routine (red), the medium accuracy routine (blue) and the high accuracy routine (green).

Figures 3.2 and 3.3 depict the unitarity and the deviations from the theoretical unitarity value for the vacuum-glass system and the vacuum-silver system, respectively, for Gaussian random surfaces with increasingly large discretization size (experiment 1 in Tab. 3.2). The results have been generated by increasing Δs from 0.050λ to 0.550λ by steps of 0.005μm.

By increasing the rms-height in the test systems, the performance of the Maxwell Eqs. solver can be studied as the surface in the test systems change from smooth to rough. The mean slope, s, in the system is proportional with the rms-height (see Eq. (3.8)), and increasing the rms-height for a fixed correlation length will, therefore, also indicate how the software performs for increasing s. Figures 3.4 and 3.5 depict the unitarity and deviation from the theoretical unitarity value, for a vacuum-glass and a vacuum-silver systems, respectively. The parameters configuration is as specified in experiment 2 in Tab. 3.2. The rms-height is incremented from 0.002 to 1.000 in steps of 0.0005 (only results for rms-height up to 0.500 are included in the vacuum-glass system). The unitarity results are plotted against the Rayleigh parameter (defined in Eq. (3.3)). This directly relates the rms-height, δ, to the surface's roughness[8].

The correlation length in a Gaussian random surface does not only affect the mean slope of the surface—the mean difference between consecutive peaks and valleys is proportional to the correlation length alone. Increasing the correlation length increases the coarseness of

[8] As the Rayleigh parameter is also dependent on the incident angle of the scattered light, the scale on the horizontal axis will not be equal in all frames in the figures.

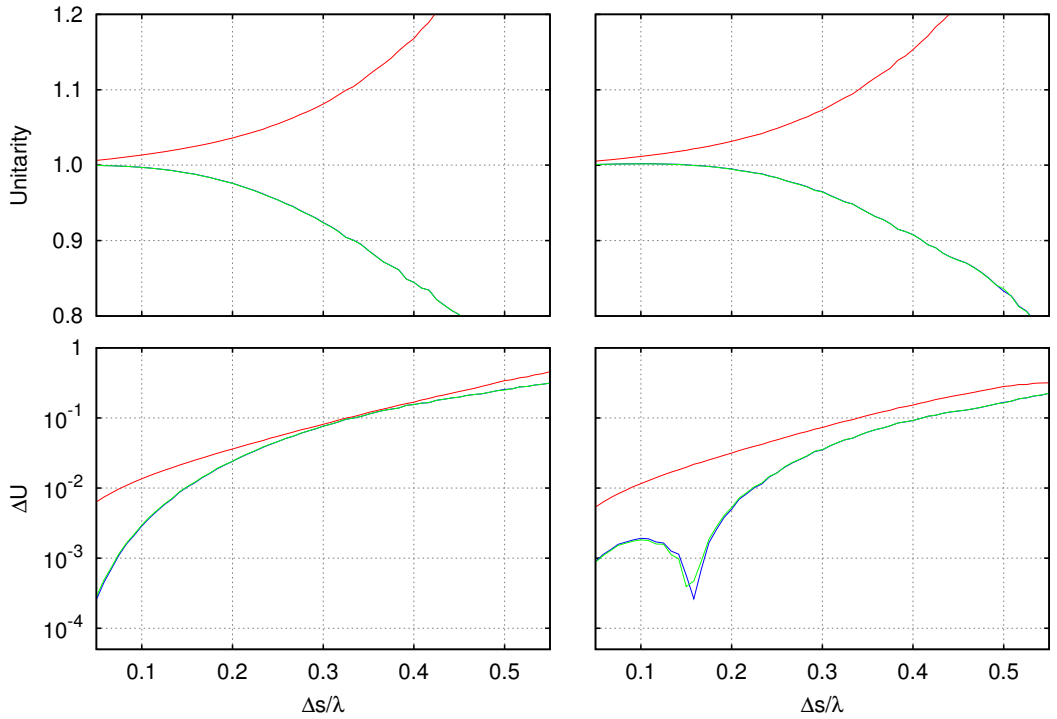

Figure 3.3: Unitarity of the light illuminating the vacuum-silver system. Otherwise the same as Fig. 3.2.

the surface's roughness, as seen in Eq. (3.4). The resulting unitarity output of Maxwell1D for Gaussian surfaces of increasing coarseness, with parameter configurations equal to that of experiment 3 in Tab. 3.2, can be seen in Figs. 3.6 and 3.7. The figure depict result for scattering in the vacuum-glass system and in the vacuum-silver system, respectively, for both the unitarity and the deviation from the theoretic unitarity value. The results have been generated by increasing the correlation length from 10^{-5} to 1.0 by even steps on a logarithmic scale.

3.3.2 Efficiency

Figure 3.8 depicts the average time used in performing 30 surface realizations with related scattering simulations in the Maxwell Eqs. solver, with the three different integrator routines. The time is plotted against the discretization size used in the computations, relative to the wavelength of the incident light. Both double and single precision results are included for Maxwell1D with the midpoint rule in use, yet only single precision results for GPatFast and double precision results for GPatAcc.

The results in Fig. 3.8 are for the vacuum-glass system. They are, however, representative for the vacuum-system system, as changing one of the media in the system does not impact the computational time noticeably.

On average, the computational time is increased by a factor of 1.54 when changing the numerical integration, from being computed with the midpoint scheme to the integrator routine GPatFast, in Maxwell1D. This is less than the average increase factor of 1.62 when going from single to double precision with the midpoint integrator. The computational time increased by a factor of 2.77, on average, when going from double precision computations with the midpoint rule to double precision computations with the GPatAcc

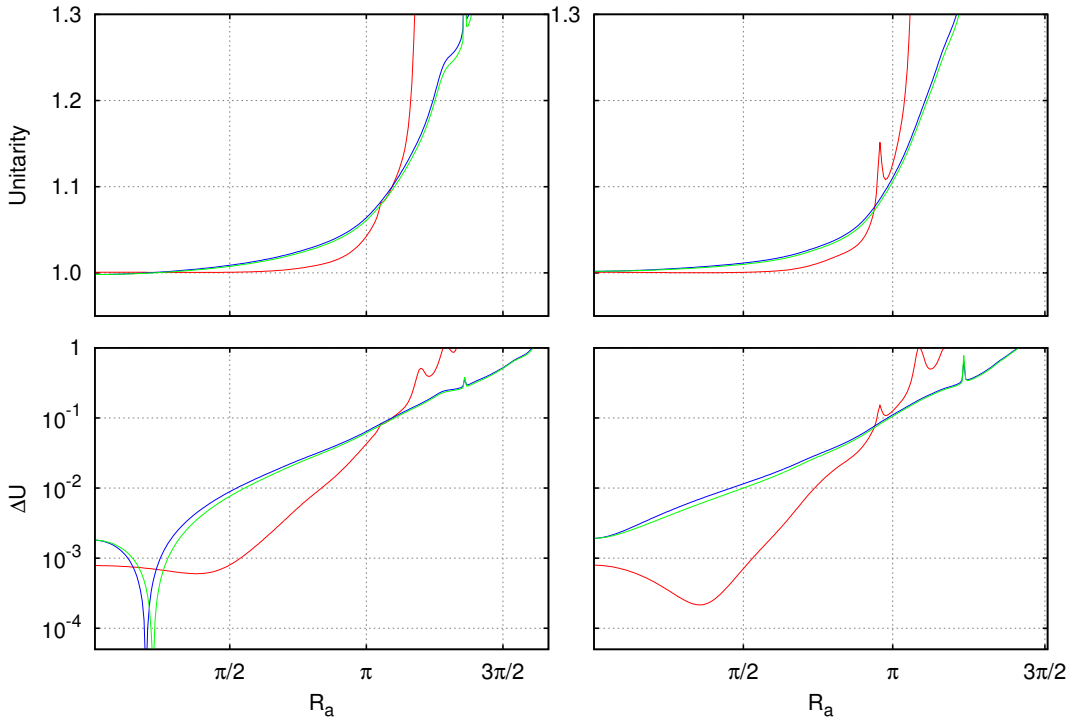

Figure 3.4: Unitarity of light illuminating the vacuum-glass system with increasing rms-height, as defined in experiment 2 in Tab. 3.2, plotted against the Rayleigh parameter defined by Eq. (3.3), for $0.002\mu\text{m} < \delta < 0.5\mu\text{m}$. Otherwise the same as Fig. 3.2.

integrator in the light scattering simulations (a factor of 4.50 compared to single precision with the midpoint integrator).

3.4 Discussion – round up

In this chapter the results for a few experiments conducted in the Maxwell Eqs. solver have been presented. The conducted experiments are of different nature, with the purpose of shedding light on the performance change with the new numerical integration schemes implemented. The experiments described in this chapter are by no means acid tests for the performance of the Maxwell Eqs. solver Maxwell1D, but are hoped to give indications of the integrators' performance in the light scattering simulations, nonetheless.

The amount of output data from the Maxwell Eqs. solver is large, but non of the output variables directly evaluates the accuracy of the computed matrix elements, $\mathcal{A}^{\pm}_{mn}$ and $\mathcal{B}^{\pm}_{mn}$. Rather, the numerically evaluated integrals in these matrix elements are used in solving the matrix equations and computing the source functions, as described in Sec. 2.1.

The unitarity is a quantity computed in the Maxwell Eqs. solver, related to the energy conservation in the system. It was expected that this would be an appropriate performance measure to indicate the accuracy change in the light scattering simulations with the new integrator routines implemented. The performance with regards to efficiency has simply been studied by using the wall-clock time as the variable of interest. This is assumed to be a more accurate measure of efficiency, than the number of function evaluations used by the routines—as much of the work done by GPatFast and GPatAcc is related to approximating the surface coordinates for the abscissa outside the integration interval's midpoint.

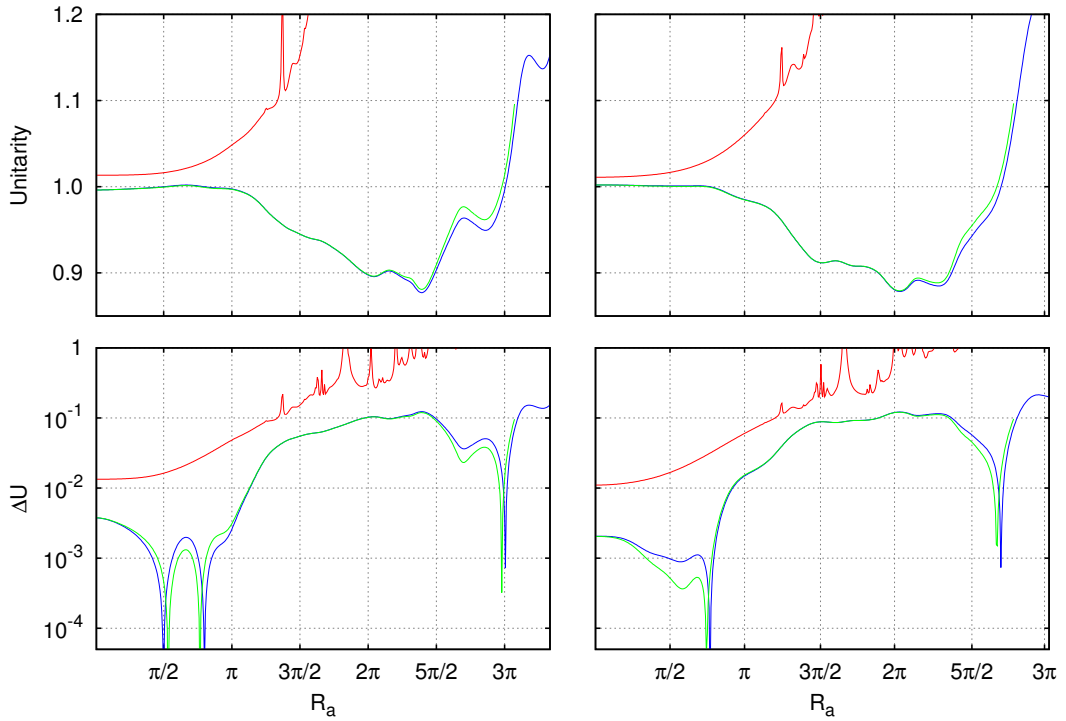

Figure 3.5: Unitarity of the light illuminating the vacuum-silver system for $0.002\mu\text{m} < \delta < 1.0\mu\text{m}$. Otherwise the same as Fig. 3.4.

3.4.1 Test runs

Consider the deviation from the unitarity value of one, in Tab. 3.4, for the deterministic test systems used in the performance analysis in Chap. 2. From the results obtained in the previous chapter the error in the computation of the matrix elements are known to be reduced by several orders of magnitude, when going from a midpoint integrator to the Gauss-Patterson rule evaluation routines. The unitarity calculations in Maxwell1D for the deterministic surfaces do, however, not reflect this.

In Tab. 3.4 the deviations from the theoretical value of the unitarity seems rather arbitrary. The deviations are similar in the two integrator routines GPatFast and GPatAcc, but not necessarily better than the ones computed with the midpoint integrator. The results are improved for the smooth and the rough surface systems, but worse for the very rough surface. The only computations that show significant improvement in the computed unitarity is the vacuum-silver system with the smooth and the rough surfaces. These results may indicate that the unitarity not necessarily is an appropriate accuracy performance measure. As computing the unitarity requires several steps from the computed matrix elements to the final result, it is possible that some erroneous behavior outside the integrator routines may effect the measure. The results in Tab. 3.4 are, however, far from sufficient to draw such a conclusion, and the unitarity is used as a performance measure in the experiments following these test runs.

3.4.2 Discretization, rms-height and correlation length

Consider Figs. 3.2 and 3.3. It can be seen that the unitarity computed by Maxwell1D seems to converge towards one, for small discretization intervals. However, the deviation from this value again reveals that ambiguous nature of this measure.

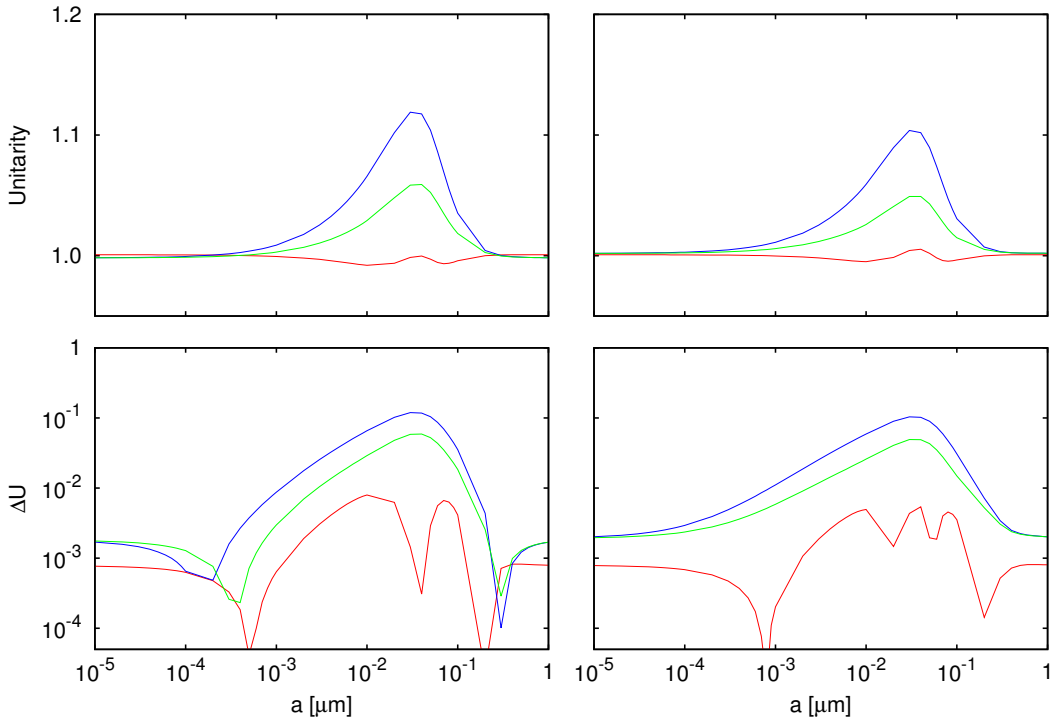

Figure 3.6: Unitarity of light illuminating the vacuum-glass system computed with Maxwell1D for increasing correlation length, as defined in experiment 3 in Tab. 3.2. Otherwise the same as Fig. 3.2.

For small discretization sizes in Fig. 3.2 the results computed with the midpoint scheme has the smallest error in the computed unitarity. For larger sizes, however, the Gauss-Patterson integrators are superior. A peculiar aspect of these results is that the results from Maxwell1D with the midpoint approximated integrals are hardly affected by the incidence angle of the light. However, the computed unitarity is (on average) increased for increased incident angle in Maxwell1D with the Gauss-Patterson integrators in use (thus, increasing the error in the vacuum-glass system where the unitarity is larger than one, and decreasing it in the vacuum-silver system, where the unitarity is less than one). Similar to the results computed for the deterministic surface systems, the deviation from the theoretical unitarity value is less in the glass-silver system with the new integrator routines (Fig. 3.3), compared to the results with the midpoint scheme. This system is, however, more noticeably affected by the increase of discretization interval size, than the glass-vacuum system.

The results in Figs. 3.4 and 3.5 show similar characteristics, yet with more dramatic differences between the integrator routines. For the increasing rms-height in the systems, the unitarity result in the Maxwell1D-version with the midpoint integrator is closest to one, in the vacuum-glass system. In the vacuum-silver system, on the other hand, the versions with the new integrator routines implemented have unitarity values closest to the theoretical unitarity value.

Similar results are also achieved for the experiments with fixed discretization size and rms-height, and varying correlation length. These results, seen in Figs. 3.6 and 3.7, do, however, show some unexpected behavior. For both the vacuum-glass and the vacuum-silver systems, the results generated with the new integrator routines have a distinct, wide peak in the unitarity error, for a certain range of correlation lengths (the width of

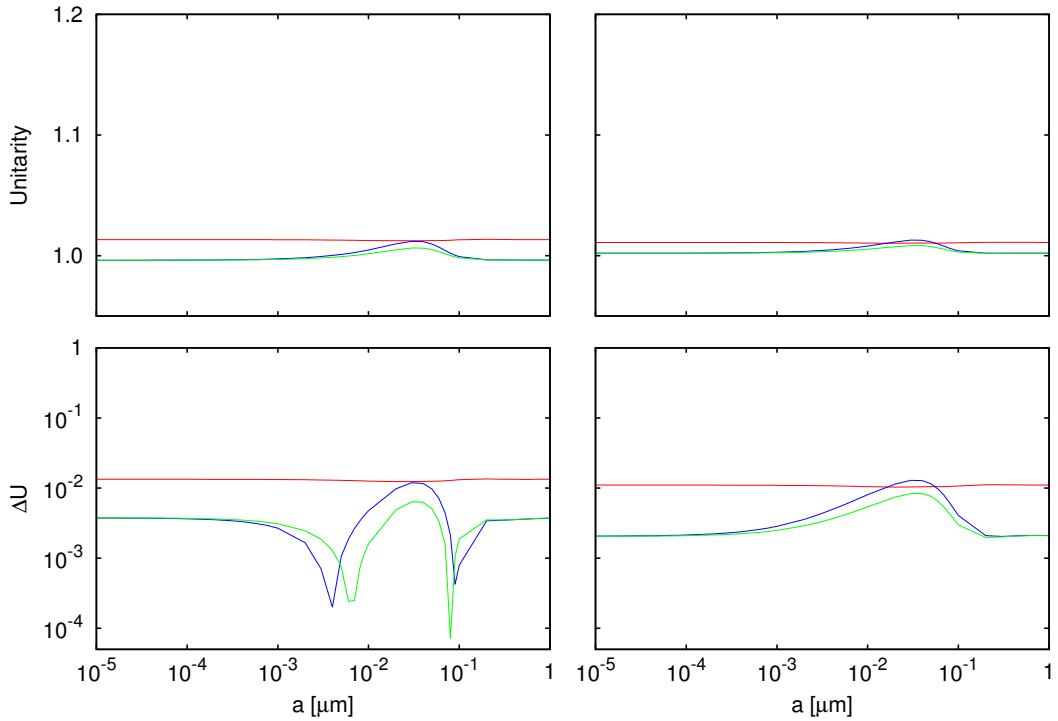

Figure 3.7: Unitarity of the light illuminating the vacuum-silver system. Otherwise the same as Fig. 3.6.

this peak is not the same in the two systems). The peak is greatest in the vacuum-glass system. The unitarity results for the different correlation lengths are quite surprising. In the performance analysis in Chap. 2 the hardest integrals to compute were not the ones in the very rough system—with the largest amplitude—but the rough system integrals—with the finest roughness of the test systems. Based on the performance of the integrators in these systems it should be expected that the unitarity is computed more accurately for large correlation lengths than for small, as a finer roughness increases the oscillations of the surface profile. Moreover, the integrals computed with the midpoint rule should be expected to be increasingly inaccurate for decreasing correlation lengths. If this is the case, the results in Fig. 3.6 and 3.7—showing a unitarity hardly affected by the correlation length for Maxwell1D with the midpoint rule integration—may indicate that the accuracy of the computed integrals is not as important as expected. Perhaps the error in the midpoint evaluated integrals is such that it averages out to a low error, when the matrix system in Eqs. (2.14) and (2.15) are solved by linear algebra techniques.

3.4.3 The performance measure and possible sources of error

The unitarity has been used as a measure of correctness of the Maxwell Eqs. solver, as there is no quantity in in the software's output that is directly linked to the accuracy of the computed integrals. Unfortunately it seams that this measure is not sufficiently unambiguous for such use. Not only does the software, by the unitarity measure, perform contrary to several expectations for the software's performance, it also computes this quantity less accurately for some the deterministic surfaces from Chap. 2—where the performance of the new integrator routines is known to be far superior to that of the midpoint routine. If the unitarity is, however, an appropriate measure for the performance

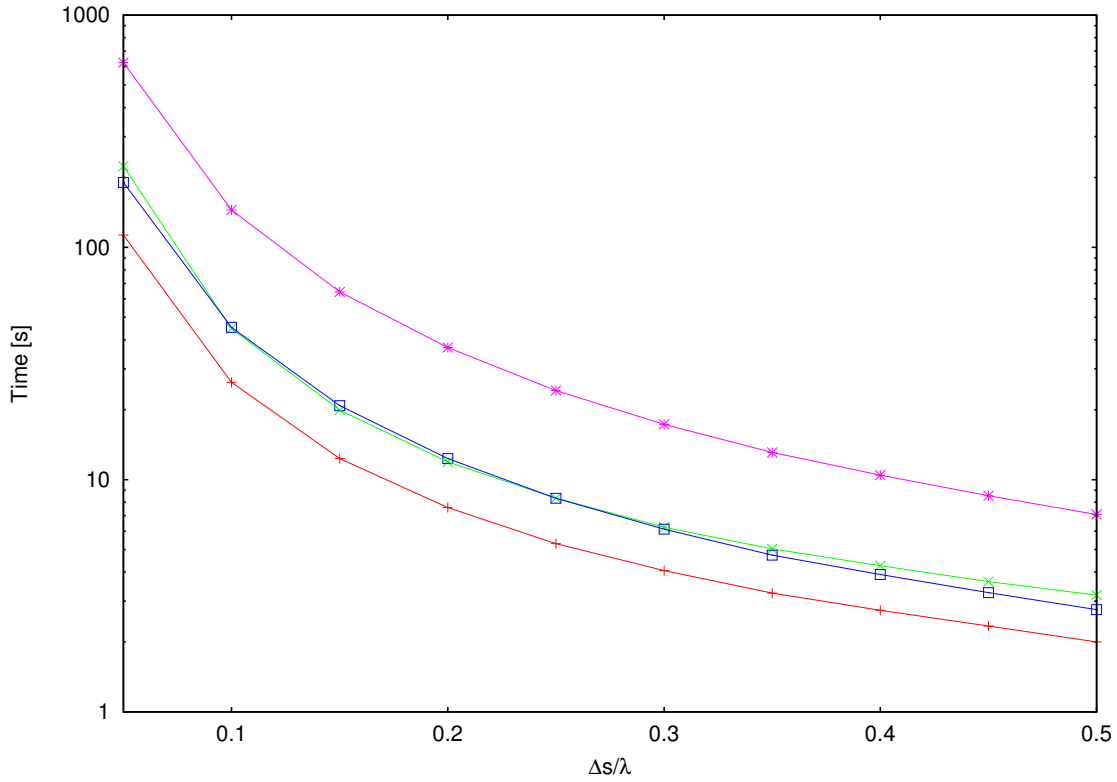

Figure 3.8: The time used to perform the light scattering simulations in Maxwell1D, with thirty surface realizations, for a vacuum-glass system of increasing discretization size (experiment 4 in Tab. 3.2). Results for Maxwell1D using the midpoint scheme with single precision (red, +), GPatFast with single precision (blue, x), midpoint scheme with double precision (green, □) and GPatAcc with double precision (pink, *), plotted as a function of the discretization size, Δs, relative to the wavelength, λ, of the incident light.

of the new integrator routines, the results are not very uplifting. In this case the all-over performance is not improved by replacing the midpoint integration scheme. The unitarity is, in fact, computed less accurately in some of the experiments conducted, particularly for the vacuum-glass system. In the vacuum-silver system the unitarity is closer to one, in the simulations with GPatFast and GPatAcc, than in the simulations with the midpoint scheme. As all light is scattered in this system—contrary to the vacuum-glass system where both scattering and transmission occur—this may indicate that a possible erroneous behavior of the unitarity measure itself comes from the part of the unitarity computation related to transmission (Eq. (3.11)).

An aspect that affects the performance of the Gauss-Patterson routines, as seen in Chap. 2, is the approximate surface representation outside the grid points. It is, however, not believed that these approximations is the reason for the failure to reach significantly improved unitarity values in the Maxwell Eqs. solver. The potential for such a failure is there—the errors of inaccurately approximated coordinates and derivatives could propagate through the Maxwell1D-simulations, and reduce the all over performance considerably. If this was the case, the results would, however, look different in a couple of respects:

- In all results presented in Figs. 3.2–3.7 the unitarity computed with Maxwell1D using GPatFast and GPatAcc are very similar. The error bounds in Tab. 2.2 give

an indication of the different accuracy that is expected by the approximated surface representations. The second order Taylor expansion (used by GPatFast) and the generalized Hermite interpolation (used by GPatAcc) have error bounds that are several orders of magnitude apart, for the test surfaces in Chap. 2. Even though the surfaces in this chapter have stochastic properties, the Hermite interpolated surface approximation is expected to approximate the surface coordinates accurate to several digits more than the Taylor approximated surface. Thus, if the surface representation was to blame the results should differ much more when computed with the two different integrator routines GPatFast and GPatAcc.

- The results for the deterministic surfaces, presented in Tab. 3.4, show similar characteristics as those in the figures depicting the unitarity of random surfaces. Again, the difference between the two surface representations is small, and, for two of the surfaces in the vacuum-glass system, the unitarity from Maxwell1D with the midpoint rule employed is the most accurate. These integrals, at least on the vacuum side of the interface, are known for certain to be considerably more accurately computed with the Gauss-Patterson integrators (as seen in the parametric studies in Chap. 2).
- The fact that the performance of the new integrator routines is systematically worse for the vacuum-glass system than for the vacuum-silver system, with identical surface configurations, more than any of the statements above, disprove that the surface representations are to blame for the inaccurate unitarity results.

Another aspect that may affect the accuracy in the Maxwell Eqs. solver is the possibility of errors in the derived analytic expressions for the integrals over generalized surfaces, in the matrix elements expressed by Eq. (2.16) and (2.17). These equations are, however, not only consistent with the single-valued surface profile integrals in Eqs. (2.33) and (2.29), that have been derived in published literature [31] as well as here. They are also consistent with the previous implementation in Maxwell1D—the Gauss-Patterson integrators are reduced to the midpoint rule, if a single integration points is used in the numerical integration. It is, therefore, deemed unlikely that this is the source of the unexpected computed unitarity values.

3.4.4 Efficiency

So far the performance of the new integrator routines implemented in the Maxwell Eqs. solver has been discussed in regards to the accuracy of the computations. An accuracy improvement is the main motivation for this work, but, as mentioned previously, the price paid for an eventual improvement is far from unimportant.

Consider Fig. 3.8, depicting the computational time used to perform the light scattering simulations in Maxwell1D for different integrator routines and floating point precision. The aspect of importance in this figure is not the time in second used by the Maxwell Eqs. solver (as this is highly machine dependent), but, rather, the factor the time is increased by using a different integrator routine in the light scattering simulations or by reducing the discretization interval size. It can be seen that the increase of computational time is, in fact, quite modest—considering that the single point midpoint scheme is replaced by integrator routines with three and seven integration point, where the coordinates of the points are computed by different approximation techniques.

The average increase of a factor of 1.54 when going from the midpoint scheme to the integrator routine GPatFast in Maxwell1D is less than expected, as the integration is presumably the single most computationally costly part of the Maxwell Eqs. solver. As seen in Fig. 3.8, the increase in computational time is largest when the integrator routine GPatAcc is used. This is as expected as the number of integration points is highest in this

routine, and as it uses the computationally expensive Hermite interpolation to approximate the surface coordinates. The average increase factor of 2.77 (for the double precision computations) is far less than expected—especially when considering the increased amount of work done by this routine compared to GPatFast (the Hermite interpolation requires the computation of two eight order and two fifth order polynomials, by the divided difference method, for each of the $\Theta(N^2)$ integrals that are computed).

The computational time of the software is rapidly reduced when the discretization size is increased. Recall that the number of grid points of the discretized surface is computed in Maxwell1D by $N = L/\Delta s$ (where L is the system width or the arc length of the surface) and that there are $\Theta(N)$ integrals for each grid point. A large reduction of computational time by an increase of Δs should, by this, be expected. An element to take note of is how the computational time of Maxwell1D can, in fact, be reduced with the new integrator routines. It can be seen in Fig. 3.8 that if the implementation of GPatFast will allow for an increase of the discretization size from, say, 0.1λ to 0.15λ, without a reduction in the output's accuracy, the computational time of Maxwell1D is reduced, compared to a discretization size of 0.1λ with the midpoint scheme. A similar computational time reduction with GPatAcc implemented will require a significantly larger increase in the discretization size, possibly too large to avoid the accuracy of the output being degraded.

3.4.5 Application of the new integrator routines

The results in this chapter are not wholeheartedly in favor of employing the integrators GPatFast and GPatAcc in the Maxwell Eqs. solver. The experiments show that the unitarity output of Maxwell1D is closer to one in some experiments, and further from one in some, with these integrators implemented rather than the simple midpoint scheme. This is quite surprising, when compared to the performance analysis in the previous chapter, which is in strong favor of using the rule evaluations quadrature routines based on the Gauss-Patterson quadrature scheme to improve accuracy of the specific types of integrals in Maxwell1D. This may be due to the unitarity not being an appropriate measure of the integrators' performance in the light scattering simulations. The measure may be affected by the computations preceding or following the numerical evaluation of the integrals, or even by possible errors in the source code of Maxwell1D. Another possibility is that there are errors in the source code of the implemented integrators, or in the analytic expressions of the integrand functions.

If the routines should be utilized, despite of this, an increase in the computational time of the software should be expected. This increase is, however, quite modest, and may possibly be negated by changing the sampling properties of simulations. How this will effect the accuracy of the output is uncertain, due to the problem related to the unitarity measure. Figures 3.2 and 3.3 do, however, indicate that the Maxwell Eqs. solver is somewhat more resistant to increased spacing in the discrete grid with the new integrator routines, than with the single point midpoint scheme. No boundary has been defined here, for a sufficiently accurate unitarity value in the system. If such a boundary was set to a modest accuracy requirement of, say, 10^{-2}, it can be seen that for the parameter configuration in experiment 1 in Tab. 3.2 higher values for Δs would be accepted with GPatFast and GPatAcc than with the midpoint scheme implemented in Maxwell1D. Unfortunately, the Maxwell Eqs. solver would fail to reach such an accuracy requirement with the Gauss-Patterson in use for rough surface with $R_a > \pi/2$, in the system dominated by transmission (Fig. 3.4).

3.5 In closing

The work presented in this thesis has been presented in three parts, to reflect development and learning that has taken place throughout the process. The goal for the tasks performed was to improve the accuracy of the numerical approximation of two classes of integrals computed in a light scattering simulation software. In aspiring for this goal different strategies of reaching it has been explored, and some have been rejected.

The starting point for the work performed was that an automatic integrator was the appropriate tool to improve the numerical accuracy of the computed integrals. Such an integrator was developed, based on the Gauss-Patterson nested quadrature scheme—a quadrature scheme expected to be highly efficient for many types of integrand functions. In the general-purpose testing of this integrator routine these expectations where confirmed, despite the fact that the error estimate analysis indicated that the integrator routine's convergence criteria was too strict for several of the tested integrals (leading to convergence with more function evaluations than necessary). However, while the integrator was among the best performing, in terms of efficiency, it was not as versatile as some of the adaptive library routines it was compared with.

After the development and testing for general-purpose integration the focus was shifted to the specialized integration in the Maxwell Eqs. solver. This required the derivation of analytic expressions for the integrals that were to be computed, and a closer look at the resulting integrand functions. While these functions were highly oscillating when considered on a surface width scale, the small integration intervals in the specific integrals resulted in slowly varying, smooth integrands within the limits of integration. With this in mind a new integrator routine was introduced, in addition to the automatic integrator, as it was expected that the automatic integrator would be unable to terminate the computations at a sufficiently low level of integration to avoid stalling the software. The new integrator routine was a rule evaluation quadrature routine, based on the same quadrature scheme as the automatic integrator, with the choice of three or seven integration points.

As the integrand functions at hand involved Hankel functions of the first kind, there were singularities in the intervals where the distance between the observation point and the scattered field was zero (the diagonal matrix elements). These singularities were, fortunately, integrable, and equations to evaluate these directly were derived. A method to improve the accuracy of the integrals containing singularities was tested, and indicated an accuracy improvement for the diagonal matrix elements. However, as only one surface representation was tested with the new singularity handling strategy, and as the improvement was not large for all the diagonal elements (there was, in fact, a reduction in the accuracy for some matrix elements, if few integration points were used), the direct singularity handling was utilized in stead of the supposedly improved strategy.

A new problem arose when the integrals from the Maxwell Eqs. solver were tested. As in any numerical software, the physical phenomenon studied in Maxwell1D relies on a discretized grid with a finite number of point. In this case, these points are the midpoint in each integration interval, making the use of a midpoint rule to approximate the integrals very convenient. With the use of a more sophisticated quadrature scheme than the midpoint rule these points (and their first order derivatives) had to be approximated outside the integration interval midpoint. To this end, three different surface approximations where tested. These were, by increasing accuracy and complexity, second order Taylor approximation, third order Taylor approximation, and generalized Hermite interpolation.

The results from extensive testing of the specialized routines countered some of the early assumptions on improving the accuracy in the Maxwell Eqs. solver. Contrary to the results in the general-purpose testing, very little accuracy improvement was achieved by

increasing the number of integration points beyond three or seven points, in the single and double precision computations, respectively. The accuracy was to a large extent limited by the error in the approximate surface representations outside the grid points. Based on this analysis the rule evaluation routine was deemed the far superior to the automatic integrator, with respect to efficiency, and to the midpoint scheme, with respect to accuracy. Two rule evaluation routines were, therefore, implemented in the Maxwell Eqs. solver, one using three integration points and a second order Taylor approximation for the surface coordinates (for use in single precision computations), and one using seven integration points and high order Hermite polynomials for to approximate the surface (for use in double precision computations). With these surface representations both the coordinates of each grid point, and their first and second order derivatives, were utilized to approximate the surface. Fortunately, this information was stored in a surface data structure in the software, such that no modification of Maxwell1D outside the integrator module was necessary (beyond including and calling the integrator routines).

The rule evaluation routines were tested with different set-ups in Maxwell1D. The experiments included varying the grid point spacing, rms-height and correlation length, as well as testing with the deterministic surfaces used in the preceding performance analysis. To measure a possible accuracy improvement in the software an energy conservation property known as the unitarity was used, while the averaged wall-clock time was used to measure efficiency. Unfortunately, the results from these test were inconclusive.

All experiments were conducted for both a vacuum-glass interface and a vacuum-silver interface. An apparent systematic behavior of the experiments was that the unitarity measure was more accurately computed (closer to one) with the new integrator routines for the vacuum-silver system. In the vacuum-glass system there was a larger variation in the results. For some experiments the systems unitarity was noticeably worse in the vacuum-glass system, with the Gauss-Patterson based integrator routines, than with the midpoint rule. Surprisingly, this behavior was also the result for the deterministic surface systems analyzed previously—systems where the new integrator routines were known to be far more accurate than the midpoint rule integrator. This indicated that the unitarity was not an appropriate measure of the correctness of the system, that there were errors in the implementation or that the computations preceding or following the numerical integration affected the unitarity measure in a negative way.

While the expected significant accuracy improvement was not observed in the experiments, the increase in computational time of the software was less than expected. There was an average increase in computational time of a factor 1.54 for the single precision computations (with three integration points and Taylor approximated surface) and 2.77 for the double precision computations (with seven integration points and Hermite interpolated surface). The increase in the single precision computations could possibly be negated, resulting in a reduction in the total computational cost of the Maxwell Eqs. solver. Such a reduction is related to a possible increase in the systems grid spacing with a more accurate integration scheme, significantly reducing the number of matrix elements to be computed.

Based on the inconclusive nature of the results, the Gauss-Patterson integrator routines are not recommended to replace the midpoint scheme in the Maxwell Eqs. solver, at this time. This is unfortunate, as the integrator routines showed great promise in the parametric studies performed during the customization and performance chapter of this thesis, and should by all means be expected to perform the integration significantly more accurate than the simple midpoint rule—with a moderate computational cost increase.

3.5.1 Further work

In the first part of the integrator routine development, the automatic integrator based on the Gauss-Patterson quadrature scheme performed very well for several of the tested integrals. Even though the automatic integrator was regarded as unsuited for the numerical integration performed in Maxwell1D, at this time, it is still a numerical integrator with promising all over performance for both accuracy and efficiency. The automatic integrator may be utilized in other numerical software where numerical integration is needed, as it is developed as a general-purpose integrator. However, as battery test alone do not give a sufficiently broad platform to evaluate an integrator's performance, further parametric studies should be conducted before the automatic integrator is incorporated in a numerical library or implemented in a different numerical software packages. The results could be used in benchmarking the performance against existing automatic routines. Further, expansions to the initial implementation of the automatic Gauss-Patterson integrator may include singularity handling, allowing for integration interval subdivision (as a fail safe measure for very hard integrals) and increasing the number of nested levels in the routine.

As mentioned, the developed rule evaluation routines are, at this point, not recommended for use in the Maxwell Eqs. solver. Before the Gauss-Patterson routines are incorporated into Maxwell1D, it is recommended that further testing is performed. This should be done with a different performance measure than the unitarity, e.g., by studying the scattering and transmission field for simple interfaces, and comparing them to other simulations or experimental results. Such a comparison may reveal flaws in the simulations performed with the new routines implemented, or worse, in the software itself—regardless of the integration scheme used. If there is, in fact, erroneous parts in the Maxwell Eqs. solver, improving these are expected to reveal that the Gauss-Patterson integrators are superior to the midpoint scheme. Alternatively, if the varying unitarity accuracy is due to, say, round-off errors or truncation errors in parts of the routine not directly related to computing the matrix elements $\mathcal{A}^{\pm}_{mn}$ and $\mathcal{B}^{\pm}_{mn}$, the physics observables computed in the simulations may be improved with the Gauss-Patterson integrators in use, despite of the unitarity measure not necessarily being closer to one.

Depending on the required accuracy in the light scattering simulations, further experiments with different discretization sizes could be a way of optimizing the software for computational speed, with the new integrator routines in use. The experiments with increasing grid point spacing indicated that the Gauss-Patterson routines made Maxwell1D somewhat more flexible regarding the choice of integration interval size. If this interval could be increased from around one tenth of the wavelength to one fifth, or even larger sizes, the computational time of running the software would be significantly reduced.

The strategy to improve the singularity handling in the routine was not implemented together with the rule evaluation routines—the singularities were computed by the direct method. It expected that the computation of the integrals containing singularities may be improved by the non-direct method proposed here. The method requires further testing for multiple surface profiles and parameter configurations to be deemed reliable.

If the routines, after further testing, are incorporated in the Maxwell Eqs. solver, it is recommended to include the use of flags related to the choice of integrator routine, in the calling sequence of the numerical software. This would allow the user to choose which version of the integrator routines to use a certain run, without the need for code modification and compilation. By setting the three point routine with Taylor approximated surface representation as the default routine, a simple flag, e.g., *-acc* could be used by the user to run Maxwell1D with the seven point routine (with Hermite interpolated surface coordinates) if higher accuracy is needed.

Bibliography

[1] IEEE Standard for Floating-Point Arithmetic. Technical report, Microprocessor Standards Committee of the IEEE Computer Society, August 2008.

[2] M. Abramowitz and I.A. Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Applied mathematics series. Dover Publications, 1964.

[3] D.E. Amos. Algorithm 644: A portable package for bessel functions of a complex argument and nonnegative order. *ACM Transactions on Mathematical Software*, 12(3):265–273, 1986.

[4] G.B. Arfken and H.J. Weber. *Mathematical Methods for Physicists*. Mathematical Methods for Physicists. Elsevier, 2005.

[5] R.L. Burden and J.D. Faires. *Numerical Analysis*. Brooks/Cole, Cengage Learning, 2011.

[6] J. Burkardt. Patterson rule – gauss-patterson quadrature rules. `http://people.sc.fsu.edu/~jburkardt/f_src/patterson_rule/patterson_rule.html`, Revised: 2013-02-16. Retrieved: 2013-09-17.

[7] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction To Algorithms*. MIT Press, 2001.

[8] P.J. Davis and P. Rabinowitz. *Methods of Numerical Integration*. Computer Sciences and Applied Mathematics. Academic Press, 1975.

[9] C. De Boor. Cadre: An algorithm for numerical quadrature. *Mathematical Software, J.R. Rice ed.*, pages 417–449, 1971.

[10] C. De Boor. On writing an automatic integration algorithm. *Mathematical Software, J.R. Rice ed.*, pages 201–209, 1971.

[11] P. Favati, G. Lotti, and F. Romani. Testing automatic quadrature programs. *Calcolo*, 27(3-4):169–193, 1990.

[12] W. Gander and W. Gautschi. Adaptive quadrature – revisited. *BIT Numerical Mathematics*, 40(1):84–101, 2000.

[13] P. Gonnet. *Adaptive quadrature re-revisited*. Ph.D. dissertation, ETH Zürich, 2009.

[14] S. Haber. Midpoint quadrature formulas. *Mathematics of Computation*, pages 719–721, 1967.

[15] E. Isaacson and H.B. Keller. *Analysis of Numerical Methods*. Dover Books on Mathematics Series. Dover Publications, 1994.

[16] A.S. Kronrod. *Nodes and weights of quadrature formulas: sixteen-place tables*. Consultants Bureau, 1965.

[17] D.P. Laurie. Practical error estimation in numerical integration. *Journal of Computational and Applied Mathematics*, 12:425–431, 1985.

[18] J.N. Lyness. The effect of inadequate convergence criteria in automatic routines. *The Computer Journal*, 12(3):279–281, 1969.

[19] J.N. Lyness. When not to use an automatic quadrature routine. *SIAM Review*, 25(1):63–87, 1983.

[20] J.N. Lyness and J.J. Kaganove. Comments on the nature of automatic quadrature routines. *ACM Transactions on Mathematical Software*, 2(1):65–81, 1976.

[21] J.N. Lyness and J.J. Kaganove. A technique for comparing automatic quadrature routines. *The Computer Journal*, 20(2):170–177, 1977.

[22] A.A. Maradudin. *Light Scattering and Nanoscale Surface Roughness*. Nanostructure Science and Technology. Springer, 2007.

[23] A.A. Maradudin, T. Michel, A.R. McGurn, and E.R. Méndez. Enhanced backscattering of light from a random grating. *Annals of Physics*, 203(2):255–307, 1990.

[24] A. Mendoza-Suárez and E.R. Méndez. Light scattering by a reentrant fractal surface. *Applied optics*, 36(15):3521–3531, 1997.

[25] A. Papoulis and S.U. Pillai. *Probability, random variables, and stochastic processes*. McGraw-Hill electrical and electronic engineering series. McGraw-Hill, 2002.

[26] T.N.L. Patterson. The optimum addition of points to quadrature formulae. *Mathematics of Computation*, 22(104):847–856, 1968.

[27] T.N.L. Patterson. Algorithm 468: algorithm for automatic numerical integration over a finite interval [d1]. *Communications of the ACM*, 16(11):694–699, 1973.

[28] R. Piessens. *Quadpack: a subroutine package for automatic integration*. Springer series in computational mathematics. Springer-Verlag, 1983.

[29] W.H. Press. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[30] I. Robinson. A comparison of numerical integration programs. *Journal of Computational and Applied Mathematics*, 5(3):207–223, 1979.

[31] I. Simonsen. Optics of surface disordered systems. *The European Physical Journal Special Topics*, 181(1):1–103, 2010.

[32] I. Simonsen, A.A. Maradudin, and T.A. Leskova. Scattering of electromagnetic waves from two-dimensional randomly rough perfectly conducting surfaces: The full angular intensity distribution. *Physical Review A*, 81(1):013806, Jan 2010.

[33] A. Spitzbart. A generalization of hermite's interpolation formula. *The American Mathematical Monthly*, 67(1):42–46, 1960.

[34] A.H. Stroud and D. Secrest. *Gaussian quadrature formulas*. Prentice-Hall series in automatic computation. Prentice-Hall, 1966.

Appendices

Appendix A

Fortran 90 source code

A.1 Automatic integrator GPat

```
!-----------------------------------------------------------------------
! GPat is the main subroutine in the module, calling upon several other
! subroutines and functions to compute the value of the definite integral
!
! Input:
!   func  -- integrand function, external function on the form:
!              function func(x)
!                 implicit none
!                 real(wp) func
!                 real(wp) x
!                 func = something(x)
!              end function func
!   a     -- left integration limit
!   b     -- right integration limit
!   info  -- derived data type integrator_info
!
! Output:
!   integral_curr -- the computed numerical quadrature
!   info  -- the integrator_info type
!-----------------------------------------------------------------------

subroutine GPat(func, a, b, info, integral_curr)
  implicit none

  real(wp)              , external       :: func
  real(wp)              , intent(in)     :: a
  real(wp)              , intent(in)     :: b
  type(integrator_info), intent(inout)  :: info
  real(wp)              , intent(out)    :: integral_curr

  !---- Local variables and arrays ----!
  integer, parameter       :: maxlength = 511
  integer, parameter       :: maxlevel = 8

  real(wp), dimension(maxlength)       :: x
  real(wp), dimension(maxlength)       :: w
  real(wp), dimension(maxlength)       :: F

  real(wp)           :: alpha
  real(wp)           :: beta
  real(wp)           :: x_scaled

  integer            :: lvl_int_curr
```

```
  integer             :: lvl_int_prev
  integer             :: lvl_count_curr
  integer             :: lvl_count_prev
  integer             :: iter

  real(wp)            :: integral_prev

  ! Initializing predifined x-array.
  call build_abscissa(x)

  ! Rescale factors for to adjust integral limints to -1, 1
  alpha = (b - a)*0.5_wp
  beta = (a + b)*0.5_wp

  ! Set default value of convergence level and converged
  info%converged = .false.
  info%n_evals = 511;

  ! Performing level 0 calculation to prepare for do loop excecution
  w(1) = 2._wp
  x_scaled = alpha*x(1) + beta
  F(1) = alpha*func(x_scaled)
  integral_curr = w(1)*F(1)

  lvl_count_prev = 1

  ! Iterative, non-adaptive automatic integration loop
  do lvl_int_curr = 1,maxlevel
    integral_prev = integral_curr

  ! Get the number of elements in the integration at this level.
    lvl_count_curr = get_level_count(lvl_int_curr)

  ! Build weight array for the current level
    call build_weights(w,lvl_count_curr)

  ! Calculate the unknown function values for current level.
    do iter = lvl_count_prev+1, lvl_count_curr
      x_scaled = alpha*x(iter)+beta
      F(iter) = alpha*func(x_scaled)
    end do

  ! Sum function values times weights to an approximation of the integral
    integral_curr = sum(w(1:lvl_count_curr)*F(1:lvl_count_curr))

  ! Check for convergence
    call convergence_check(integral_curr, integral_prev, info)

  ! Exit do-loop if convergence is reached
    if(info%converged) then
      info%n_evals = lvl_count_curr
      exit
    end if

    lvl_count_prev = lvl_count_curr
    lvl_int_prev = lvl_int_curr

  end do
end subroutine GPat
```

A.2 GPatFast

```
!-----------------------------------------------------------------------
! GPatFast is a rule evaluation quadrature routine using a three point
! Gauss-Patterson quadrature to approximate the definite integral of the
! diagonal matrix elements of A and B in Maxwell1D. Uses second order Taylor
! approximation to represent the surface coordiantes outside the grid point
!
! Input:
!   xi1_n   -- x1 coordinate of grid point
!   D1xi1_n -- First order derivative of x1 coordinate of grid point
!   D2xi1_n -- Second order derivative of x1 coordinate of grid point
!   xi3_n   -- x3 coordinate of grid point
!   D1xi3_n -- First order derivative of x3 coordinate of grid point
!   D2xi3_n -- Second order derivative of x3 coordinate of grid point
!   xi1_m   -- x1 coordinate of observation point
!   xi3_m   -- x3 coordinate of observation point
!   delta   -- grid point spacing
!   nwc     -- physical properties of the system: sqrt(varepsilon)*omega/c
!
! Output:
!   integral_A -- computed matrix element A_mn
!   integral_B -- computed matrix element B_mn
!-----------------------------------------------------------------------

subroutine GPatFast(xi1_n, D1xi1_n, D2xi1_n, xi3_n, D1xi3_n, D2xi3_n, xi1_m,
    xi3_m, delta, nwc, integral_A, integraL_B)
  Use Math_Lib_Wrapper,    only : Hankel_Function      ! Hankel function
   libraries
  implicit none

  !---- Input/output variables ----!
  real(wp)    , intent(in)    :: xi1_n
  real(wp)    , intent(in)    :: D1xi1_n
  real(wp)    , intent(in)    :: D2xi1_n
  real(wp)    , intent(in)    :: xi3_n
  real(wp)    , intent(in)    :: D1xi3_n
  real(wp)    , intent(in)    :: D2xi3_n
  real(wp)    , intent(in)    :: delta
  real(wp)    , intent(in)    :: xi1_m
  real(wp)    , intent(in)    :: xi3_m
  complex(wp), intent(in)    :: nwc
  complex(wp), intent(out)   :: integral_A
  complex(wp), intent(out)   :: integral_B

  !---- Local variables and arrays ----!

  real(wp), dimension(3)      :: x
  real(wp), dimension(3)      :: w
  complex(wp), dimension(3)   :: A
  complex(wp), dimension(3)   :: B

  complex(wp), dimension(2)   :: Hankel

  real(wp)           :: alpha
  real(wp)           :: x_scaled

  integer            :: iter

  real(wp)     :: xi1_x
  real(wp)     :: xi3_x
```

```
    real(wp)     :: D1xi1_x
    real(wp)     :: D1xi3_x
    real(wp)     :: dxi1
    real(wp)     :: dxi3
    real(wp)     :: dR
    complex(wp)  :: arg

    ! Rescale factors for to adjust integral limints to -1, 1. Beta = 0._wp
    alpha = delta*0.5_wp

    ! Building abscissae and weights arrays
    x(1) = 0._wp
    x(2) = -0.7745966692414834_wp
    x(3) = 0.7745966692414834_wp

    w(1) = 0.8888888888888888_wp
    w(2) = 0.5555555555555556_wp
    w(3) = 0.5555555555555556_wp

    ! Performing midpoint evaluation
    dxi1 = xi1_m - xi1_n
    dxi3 = xi3_m - xi3_n
    dR = sqrt(dxi1**2 + dxi3**2)
    arg = nwc*dR

    Hankel  = Hankel_Function(arg,0._wp,2)

    A(1) = alpha*(-imu*0.25_wp)*nwc*(Hankel(2)/dR)*(dxi1*D1xi3_n - dxi3*
      D1xi1_n)
    B(1) = alpha*(imu*0.25_wp)*Hankel(1)

    ! Performing Taylor approximation and off-midpoint function evaluations
    do iter = 2, 3
      x_scaled = alpha*x(iter)

      xi1_x   = xi1_n + D1xi1_n*x_scaled + (D2xi1_n*0.5_wp)*(x_scaled**2._wp)
      D1xi1_x = D1xi1_n + D2xi1_n*x_scaled
      xi3_x   = xi3_n + D1xi3_n*x_scaled + (D2xi3_n*0.5_wp)*(x_scaled**2._wp)
      D1xi3_x = D1xi3_n + D2xi3_n*x_scaled

      dxi1 = xi1_m - xi1_x
      dxi3 = xi3_m - xi3_x
      dR = sqrt(dxi1**2 + dxi3**2)
      arg = nwc*dR

      Hankel  = Hankel_Function(arg,0._wp,2)

      A(iter) = alpha*(-imu*0.25_wp)*nwc*(Hankel(2)/dR)*(dxi1*D1xi3_x - dxi3*
      D1xi1_x)
      B(iter) = alpha*(imu*0.25_wp)*Hankel(1)
    end do

    ! Computing quadrature sums
    integral_A = sum(w*A)
    integral_B = sum(w*B)

  end subroutine GPatFast
```

Appendix B

Tables for convergence and efficiency

B.1 Integration routine convergence

Table B.1: Integration level at convergence, for requested relative accuracies 10^{-1}–10^{-5}. An integration level marked by F denotes failure to converge at the requested accuracy. All runs are with double precision.

	$\varepsilon_{rel} = 10^{-1}$	$\varepsilon_{rel} = 10^{-2}$	$\varepsilon_{rel} = 10^{-3}$	$\varepsilon_{rel} = 10^{-4}$	$\varepsilon_{rel} = 10^{-5}$
Integral	Integration level				
1	2	2	2	3	3
2	2	3	3	3	3
3	6	6	6	6	6
4	2	2	2	2	3
5	2	2	2	2	3
6	2	2	3	3	3
7	4	5	5	5	5
8	2	2	2	3	3
9	1	2	3	4	5
10	2	2	3	4	4
11	2	3	4	6	7
12	2	3	5	6	6
13	6	7	7	7	7
14	4	4	5	5	5

B.2 Integrator routine efficiency

The number of function evaluations needed to reach convergence for the definite integrals in Tab. 1.1, using the automatic integrator routines DQNG, DQAG, DQAGS and GPat, with a requested relative accuracies 10^{-10} and 10^{-15}, are presented in Tabs. B.3. Integration with DQAG has been performed with input value for the parameter *key* set to 1, 3 and 6, in turn.

Table B.2: Same as Tab. B.1, but with requested relative accuracies 10^{-6}–10^{-15}.

Integral	$\varepsilon_{rel} = 10^{-6}$	$\varepsilon_{rel} = 10^{-7}$	$\varepsilon_{rel} = 10^{-8}$	$\varepsilon_{rel} = 10^{-9}$	$\varepsilon_{rel} = 10^{-10}$
	Integration level				
1	3	3	3	3	4
2	3	4	4	4	4
3	6	6	6	6	6
4	3	3	3	3	3
5	3	3	3	3	3
6	3	4	4	4	4
7	5	5	5	5	6
8	3	4	4	4	4
9	5	6	7	8	8
10	5	6	6	7	7
11	8	F	F	F	F
12	7	7	7	8	8
13	7	7	7	7	7
14	6	6	6	6	6
Integral	$\varepsilon_{rel} = 10^{-11}$	$\varepsilon_{rel} = 10^{-12}$	$\varepsilon_{rel} = 10^{-13}$	$\varepsilon_{rel} = 10^{-14}$	$\varepsilon_{rel} = 10^{-15}$
	Integration level				
1	4	4	4	4	4
2	4	4	4	4	4
3	7	7	7	7	8
4	3	3	4	4	4
5	3	3	4	4	4
6	4	4	4	4	4
7	6	6	6	6	8
8	4	4	5	5	5
9	F	F	F	F	F
10	7	7	8	8	8
11	F	F	F	F	F
12	8	8	8	8	8
13	7	7	7	7	F
14	6	6	6	6	7

Table B.3: The number of function evaluations needed to reach convergence for the integrals in Tab. 1.1, with four different automatic integrator routines. DQNG, DQAG and DQAGS are QUADPACK routines, and GPat is the integrator implemented here. Subscripts 1, 3 and 6 on the DQAG routine denote value of a parameter *key* taken as input by the routine, corresponding to three different integration rules used by this integrator. F denotes a routine that is unable to reach the requested accuracy. Requested relative accuracy is 10^{-10} and 10^{-15}.

			$\varepsilon_{rel} = 10^{-10}$			
Integral	$DQAG_1$	$DQAG_3$	$DQAG_6$	DQAGS	DQNG	GPat
1	15	31	61	21	21	31
2	45	31	61	21	21	31
3	765	217	183	315	87	127
4	15	31	61	21	21	15
5	15	31	61	21	21	15
6	75	31	61	21	21	31
7	165	217	183	189	87	127
8	45	31	61	21	21	31
9	585	1023	1647	231	F	511
10	345	527	793	399	F	255
11	1005	2015	3599	231	F	F
12	375	589	915	483	F	511
13	1905	961	915	1323	F	255
14	555	403	183	357	F	123
			$\varepsilon_{rel} = 10^{-15}$			
Integral	$DQAG_1$	$DQAG_3$	$DQAG_6$	DQAGS	DQNG	GPat
1	45	31	61	21	21	31
2	45	31	61	21	21	31
3	1005	465	183	651	F	255
4	15	31	61	21	21	31
5	15	31	61	21	21	31
6	105	31	61	63	43	63
7	315	217	183	231	87	511
8	105	31	61	63	43	63
9	945	1705	2989	231	F	F
10	375	651	1037	483	F	511
11	1725	3069	5673	231	F	F
12	555	713	1159	525	F	511
13	3765	1271	915	2457	F	F
14	975	465	183	651	F	255

Appendix C

Problem analysis

C.1 Deriving expression for the diagonal matrix elements

Consider the integrals in the matrix elements defined by Eqs. (2.16) and (2.17). For the diagonal elements the integrand functions will be on the form $A_\pm(s_m|s_m+u)$ and $B_\pm(s_m|s_m+u)$, and the related argument to the Hankel functions is $\chi_\pm(s_m|s_n+u)$. As the arguments in $\chi_\pm$ are vector-valued surface functions, a starting point for the evaluation of the diagonal elements is to write $\xi(s_m+u)$ and $\eta(s_m+u)$ as

$$\xi(s_m+u) = \xi(s_m) + \xi'(s_m)u + \frac{\xi''(s_m)}{2}u^2 + ..., \tag{C.1}$$

$$\eta(s_m+u) = \eta(s_m) + \eta'(s_m)u + \frac{\eta''(s_m)}{2}u^2 + ..., \tag{C.2}$$

by Taylor expansion. This allows the Hankel function argument to be re-written to

$$\begin{aligned}
\chi(s_m|s_n+u) &= \sqrt{\varepsilon_\pm}\frac{\omega}{c}\left[\left(-\delta_\eta - \xi'(s_m)u - \frac{\xi''(s_m)}{2}u^2 - ...\right)^2\right. \\
&\qquad \left. + \left(\delta_\xi - \eta'(s_m)u - \frac{\eta''(s_m)}{2}u^2 - ...\right)^2\right]^{1/2} \\
&= \sqrt{\varepsilon_\pm}\frac{\omega}{c}\left[\delta_\eta^2 + \xi'(s_m)^2u^2 + 2\delta_\eta u\xi'(s_m) + \delta_\xi^2 + \eta'(s_m)^2u^2 - 2\delta_\xi u\eta'(s_m) + ...\right]^{1/2} \\
&= \sqrt{\varepsilon_\pm}\frac{\omega}{c}\left[u^2\gamma(s_m)^2 + \delta^2 + ...\right]^{1/2} \\
&= \sqrt{\varepsilon_\pm}\frac{\omega}{c}|u|\gamma(s_m) + ...,
\end{aligned} \tag{C.3}$$

for a small argument u inside the interval Δs with a singularity at the midpoint. The appropriate small argument expansions of the Hankel functions are [4]

$$H_0^{(1)}(z) = \frac{2i}{\pi}\left(\ln\frac{z}{2} + \gamma\right) + 1 + \mathcal{O}(z^2\ln z), \tag{C.4}$$

$$\frac{H_1^{(1)}(z)}{z} = -\frac{2i}{\pi}\frac{1}{z^2} + \frac{i}{\pi}\left(\ln\frac{z}{2} + \gamma + \frac{1}{2}\right) - \frac{1}{2} + \mathcal{O}(z^2\ln z), \tag{C.5}$$

where γ is the Euler's constant (not to be confused with the normalizing factor $\gamma(s)$ of Eq. (2.10)). The integrals in the diagonal elements $\mathcal{A}_{mm}^\pm$ and $\mathcal{B}_{mm}^\pm$ may be re-written by

Eqs. (C.1)–(C.5). To the leading term, the diagonal elements of $\mathcal{A}$ can be expressed by

$$\begin{aligned}
\mathcal{A}^{\pm}_{mm} &= \lim_{\delta\to 0^+} -\frac{i}{4}\varepsilon_{\pm}\frac{\omega^2}{c^2}\int_{-\Delta s/2}^{\Delta s/2} \mathrm{d}u \left[-\frac{2i}{\pi}\frac{1}{\chi(s_m|s_m+u)} + ...\right] \\
&\qquad \times \left[(\eta'(s_m)+...)\left(-\delta_\eta - u\xi'(s_m) - \frac{u^2}{2}\xi''(s_m) + ...\right)\right. \\
&\qquad\quad \left. - (\xi'(s_m)+...)\left(-\delta_\xi - u\eta'(s_m) - \frac{u^2}{2}\eta''(s_m) + ...\right)\right] \\
&= \lim_{\delta\to 0^+} -\frac{i}{4}\varepsilon_{\pm}\frac{\omega^2}{c^2}\int_{-\Delta s/2}^{\Delta s/2} \mathrm{d}u \left[-\frac{2i}{\pi}\frac{1}{\chi(s_m|s_m+u)} + ...\right] \\
&\qquad \times \left[-\gamma(s_m)\delta - \frac{u^2}{2}\left(\eta''(s_m)\xi'(s_m) - \xi''(s_m)\eta'(s_m)+\right) + ...\right] \\
&\simeq \lim_{\delta\to 0^+} \frac{1}{2\pi}\int_{-\Delta s/2}^{\Delta s/2} \mathrm{d}u \left(\frac{\gamma(s_m)\delta}{\gamma(s_m)^2u^2+\delta^2} + \frac{\eta''(s_m)\xi'(s_m) - \xi''(s_m)\eta'(s_m)}{2\gamma(s_m)^2}\right) \\
&= \lim_{\delta\to 0^+} \frac{\gamma(s_m)}{2\pi}\int_{-\Delta s/2\delta}^{\Delta s/2\delta} \mathrm{d}w \frac{1}{\gamma(s_m)^2u^2+1} + \frac{\eta''(s_m)\xi'(s_m) - \xi''(s_m)\eta'(s_m)}{4\pi\gamma(s_m)^2}\int_{-\Delta s/2}^{\Delta s/2}\mathrm{d}u \\
&= \frac{1}{2\pi}\left[\tan^{-1}\left(\gamma(s_m)w\right)\right]_{w=-\frac{\Delta s}{2\delta}}^{\frac{\Delta s}{2\delta}} + \Delta s\frac{\eta''(s_m)\xi'(s_m) - \xi''(s_m)\eta'(s_m)}{4\pi\gamma(s_m)^2} \\
&= \frac{1}{2} + \Delta s\frac{\eta''(s_m)\xi'(s_m) - \xi''(s_m)\eta'(s_m)}{4\pi\gamma(s_m)^2}.
\end{aligned} \tag{C.6}$$

A similar, yet simpler, procedure yields the diagonal elements for the $\mathcal{B}$ matrix:

$$\begin{aligned}
\mathcal{B}^{\pm}_{mm} &\simeq 2\frac{i}{4}\int_{0}^{\Delta s/2} \mathrm{d}u\ H_0^{(1)}\left(\sqrt{\varepsilon_{\pm}}\frac{\omega}{c}u\gamma(s_m)\right) \\
&= \frac{i}{2}\int_{0}^{\Delta s/2} \mathrm{d}u\ \left[\frac{2i}{\pi}\left(\ln\left(\frac{\sqrt{\varepsilon_{\pm}}\frac{\omega}{c}\gamma(s_m)u}{2}\right)+\gamma\right)+1+...\right] \\
&= \frac{i}{2}\frac{\Delta s}{2}\left[\frac{2i}{\pi}\left(\ln\left(\sqrt{\varepsilon_{\pm}}\frac{\omega}{c}\frac{\gamma(s_m)\Delta s}{4e}\right)+\gamma\right)+1+...\right] \\
&= \frac{i}{4}\Delta s H_0^{(1)}\left(\sqrt{\varepsilon_{\pm}}\frac{\omega}{c}\frac{\gamma(s_m)\Delta s}{2e}\right)
\end{aligned} \tag{C.7}$$

C.2 Integrand function

Figure C.1 depicts the integrand function defined by Eq. (2.31), for the three surface configurations listed in Tab. 2.1 with the observation point $x_m = 0.0$.

The integrand functions defined by Eqs. (2.30) and (2.31), with the observation point in the leftmost integration interval ($x_m = -14.97$), can be seen in Figs. C.2 and C.3, respectively. The figures include plots for all three configurations listed in Tab. 2.1.

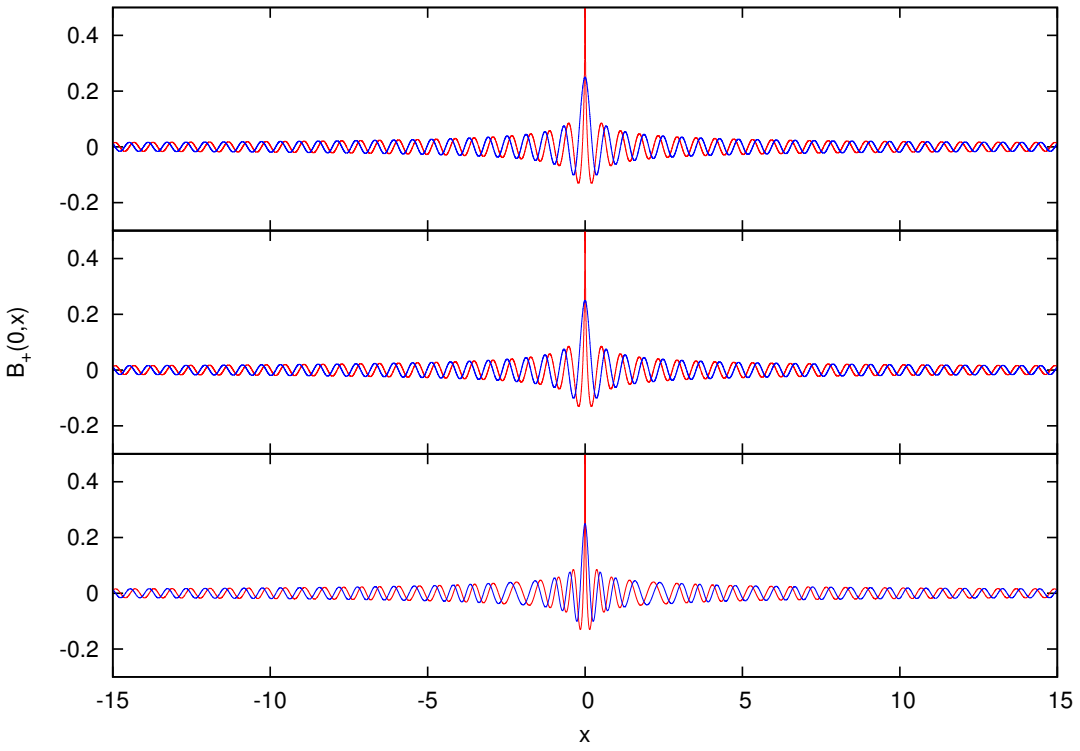

Figure C.1: Integrand function $B_+(x_s|x)\Big|_{x_s=0}$ defined by Eq. (2.31) plotted for the smooth (top), rough (mid) and very rough (bottom) surfaces defined in Tab. 2.1. The real (red) and imaginary (blue) parts of the functions are separated.

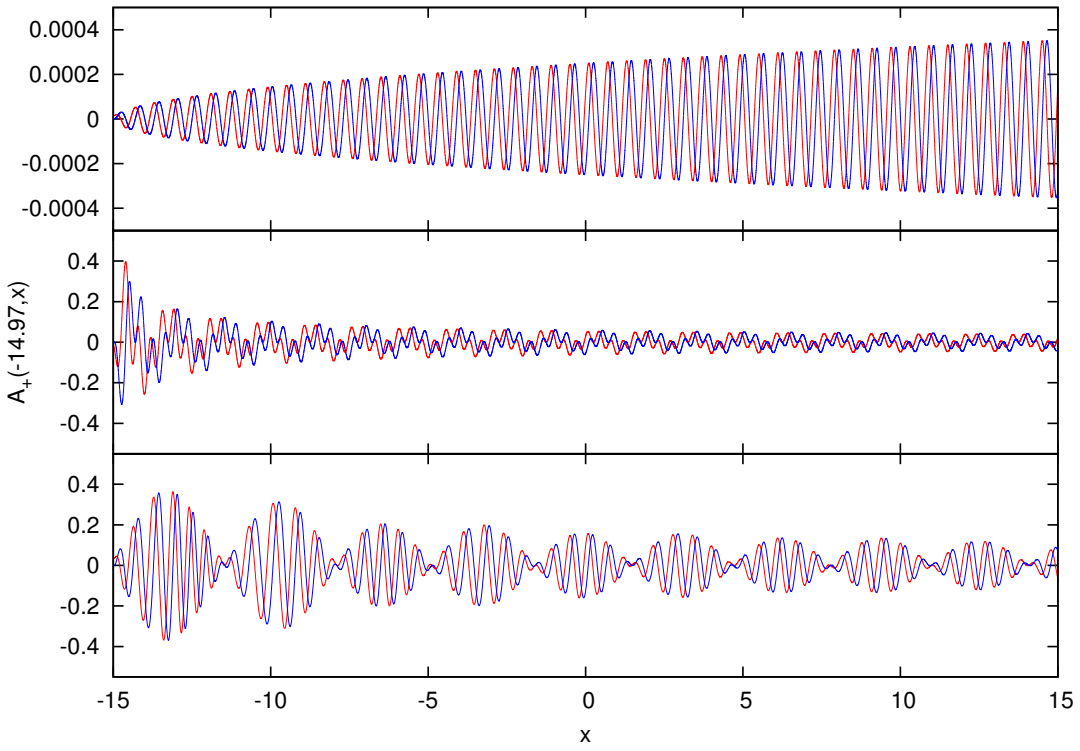

Figure C.2: Integrand function $A_+(x_m|x)\Big|_{x_m=-14.97}$, defined by Eq. (2.30), plotted for the smooth (top), rough (mid) and very rough (bottom) surfaces defined in Tab. 2.1. The real (red) and imaginary (blue) parts of the functions in separate curves.

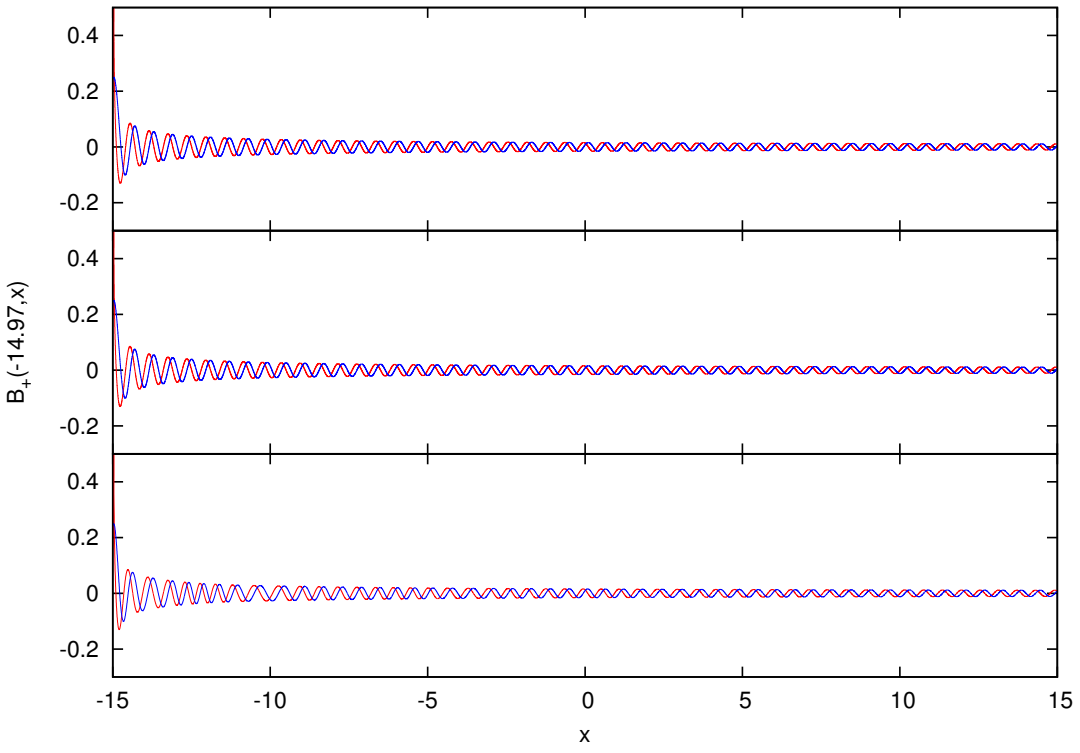

Figure C.3: Same as Fig. C.2, but for the integrand function $B_+(x_m|x)\Big|_{x_m=-14.97}$ defined by Eq. (2.31).

C.3 Integration intervals

Figure C.4 depicts the integrand function $B_+(x_m|x)$, defined by Eq. (2.31), for several selected integration intervals. The figure depicts the real and imaginary part (in separate subfigures) of a scaled version of the integrand function for the rough system configuration in Tab. 2.1. The distances from the intervals' midpoints to the observation point are $|x_m - x_n| = \{\Delta x,\ 2\Delta x,\ 5\Delta x,\ 10\Delta x, 20\Delta x\}$. The observation point $x_m = 0.0$. The computed $B_+(x_m|x)$-values have scaled by subtracting the value of the function at the midpoint of the interval:

$$\beta(x, x_n) = B_+(0|x) - B_+(0|x_n)), \tag{C.8}$$

such that all plotted curves are zero at $x = x_n$.

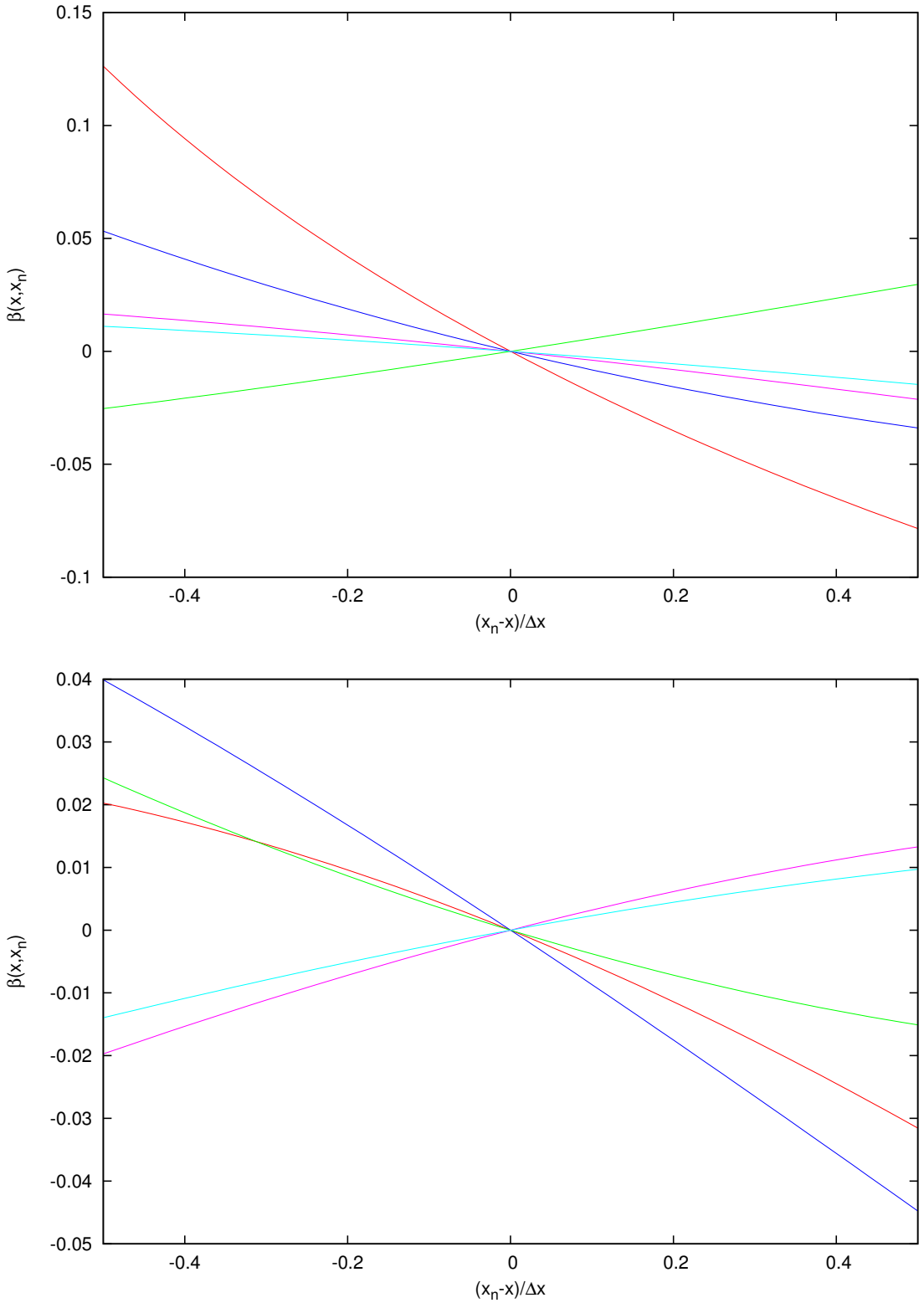

Figure C.4: Scaled integrand function $\beta(s, s_n)$ given by Eqs. (2.31) and Eq. (C.8), plotted for integration intervals with midpoints at distances from the observation points (in units of Δx) of 1 (red), 2 (blue), 5 (green), 10 (pink) and 20 (cyan). The curves are for the rough surface system of Tab. 2.1, with observation point $x_m = 0.0$. The complex integrand function is separated into a real part (top) and an imaginary part (bottom).

Appendix D

Parametric studies

D.1 Performance profiles

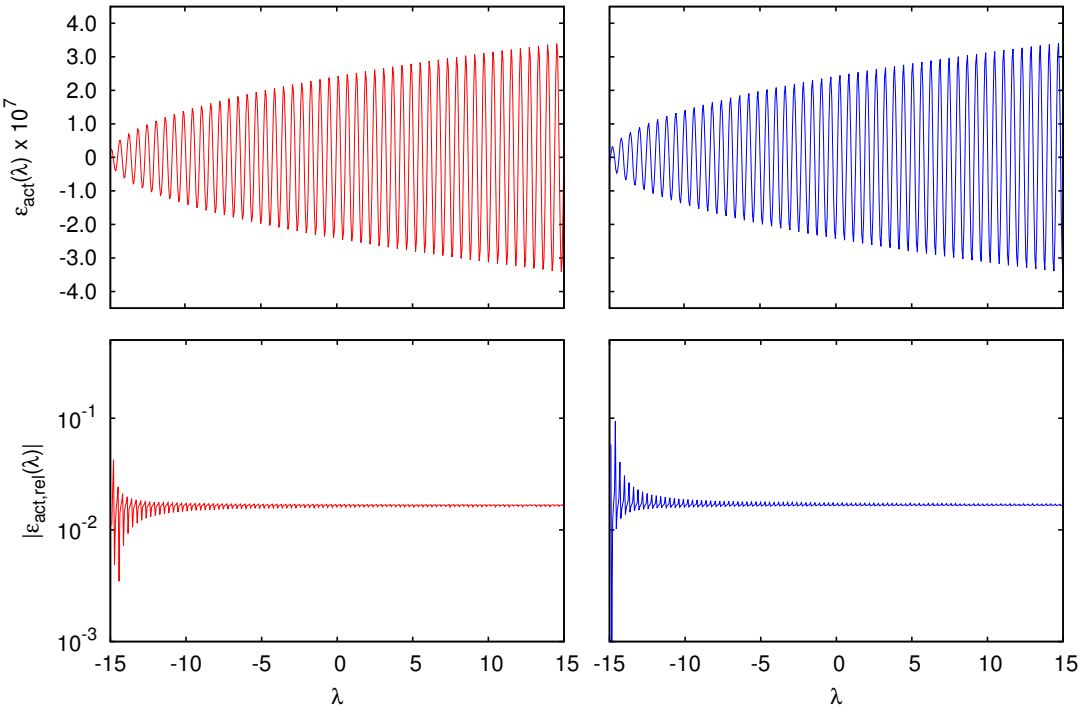

Figure D.1: Performance profiles for the midpoint rule integrator, for the problem family defined by Eq. (2.40), with the smooth surface system defined in Tab. 2.1 and $x_m = -14.97$. The performance profile given for the real part (red) and imaginary part (blue). The actual error ε_{act} and actual relative error $\varepsilon_{act,rel}$ are displayed. The figures of each column share horizontal axis, and the figures in each row share vertical axis.

Figures D.1 and D.2 depict performance profiles for the rule evaluation quadrature routine that utilizes the midpoint rule, for the problem families defined by Eqs. (2.40) and (2.41), respectively, with the smooth surface system defined in Tab. 2.1. The observation point is in the leftmost integration interval, $x_m = -14.97$. Both the real and imaginary parts of the problem families are included in the figures. The equivalent performance profiles for the test system with a very rough surface profile, as defined in Tab. 2.1, are given in Figs. D.3 and D.4.

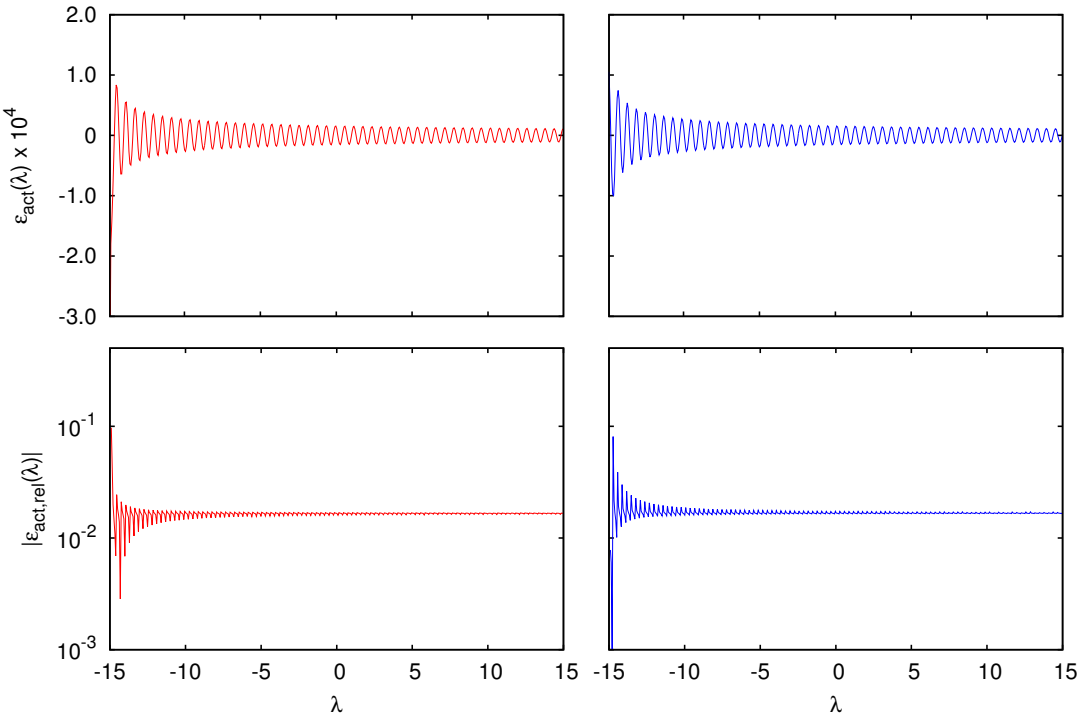

Figure D.2: Performance profiles for the midpoint rule, for the problem family defined by Eq. (2.41). Otherwise the same as Fig. D.1.

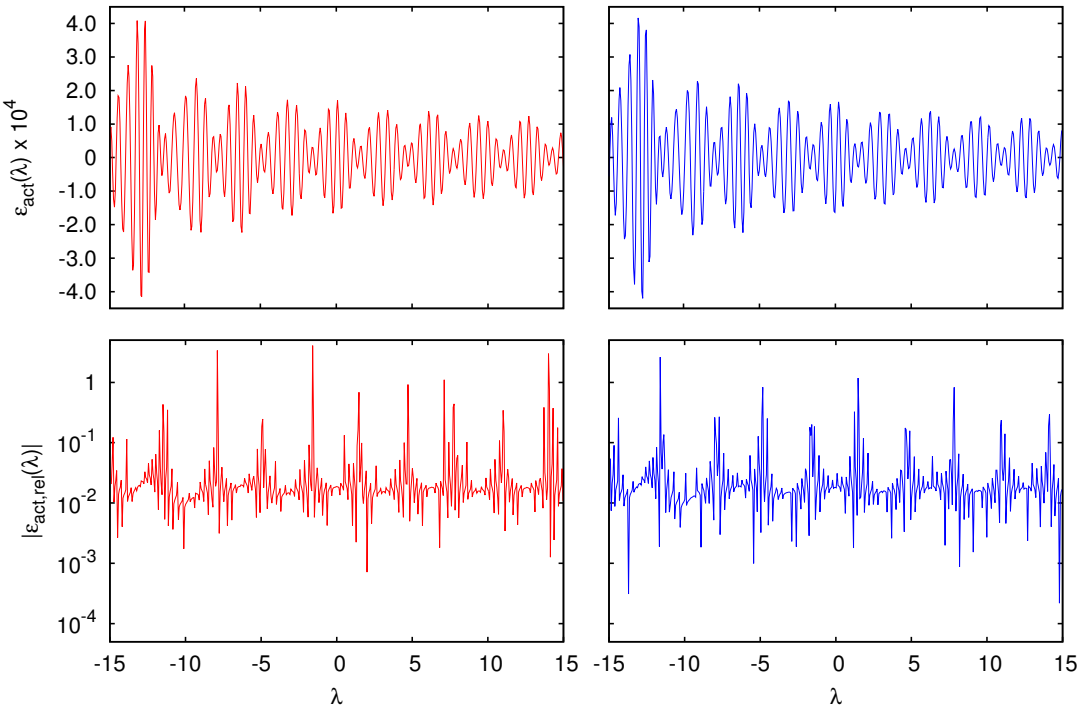

Figure D.3: Same as Fig. D.1, but for test system defined as very rough in Tab. 2.1.

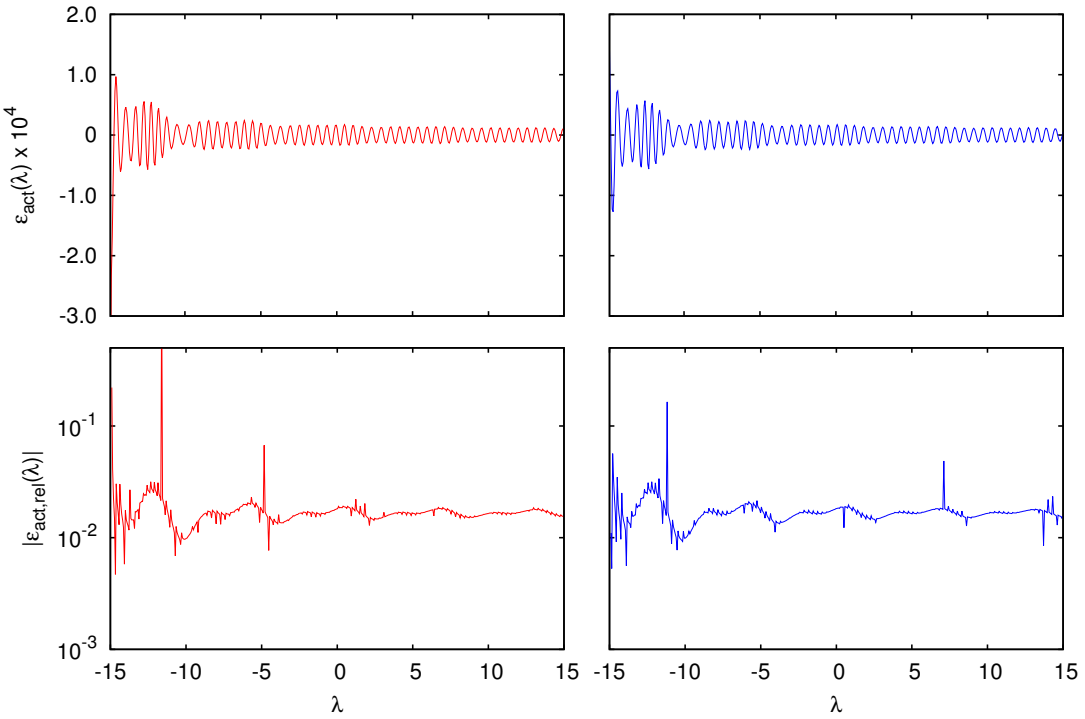

Figure D.4: Same as Fig. D.2, but for test system defined as very rough in Tab. 2.1.

Figure D.5 depicts performance profiles for GPatREQR when called with $n = 7$, for the real part of the problem family defined by Eq. (2.40), with the rough surface system. The performance profile of the rule evaluation routine with the three different approximation of the surface profiles are shown, as is the profile for an analytic surface representation. Double floating point precision is used in the computations, and only relative actual error plots are included in the figure.

Single precision performance performance profiles for GPatREQR with three integration points, generated for the real part of the problem family defined by Eq. (2.41) can be seen in Fig. D.6

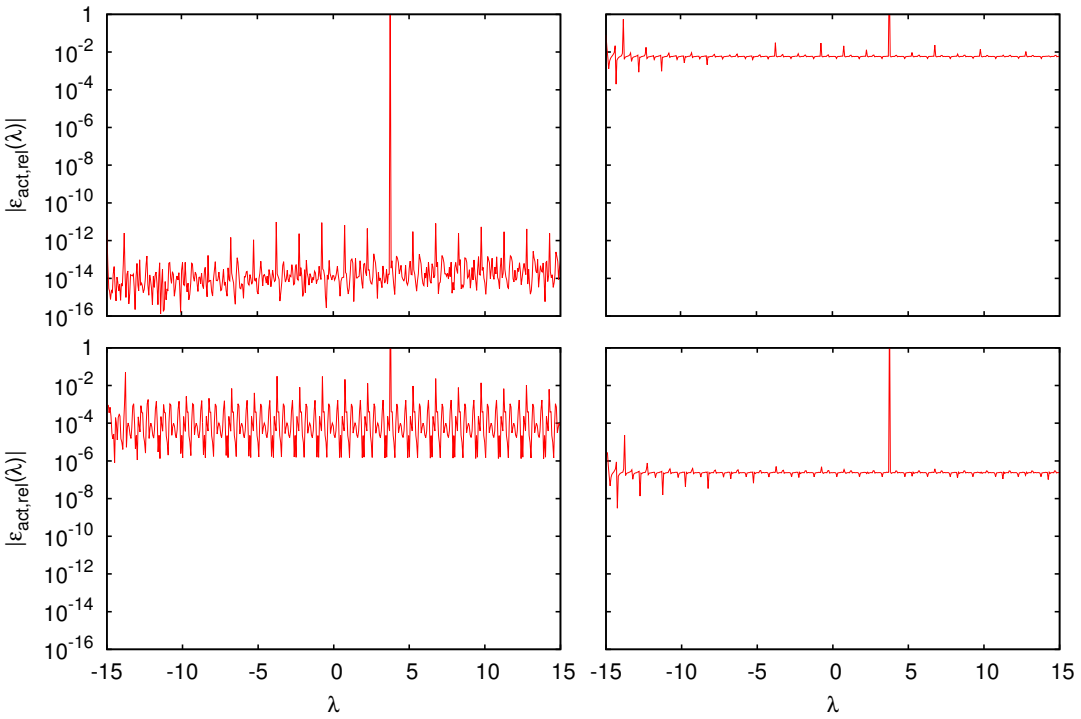

Figure D.5: Performance profiles for the rule evaluation quadrature routine GPatREQR with seven integration points ($n = 7$) and double floating point precision, for the real part of the problem family defined by Eq. (2.40). The test system is the rough surface system in Tab. 2.1, with and $x_m = -14.97$. The performance profiles are given for surface representations: analytic (top, left), second order Taylor approximation (top, right), third order Taylor (bottom, left), hermite interpolation (bottom, right)

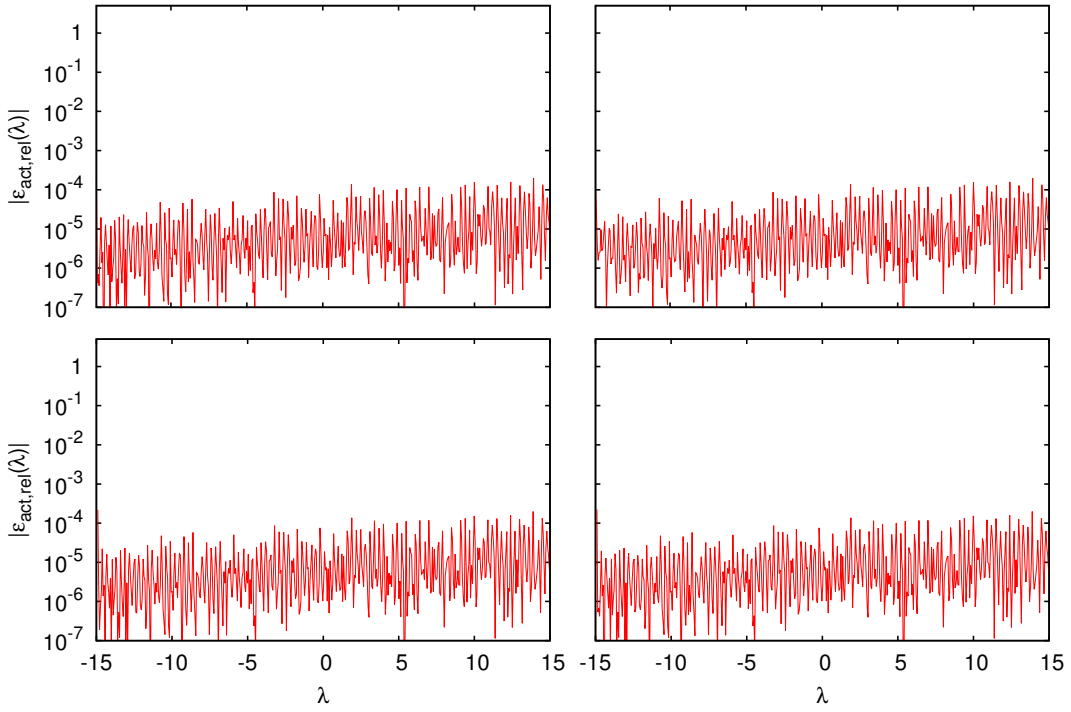

Figure D.6: Same as Fig. D.5, but with three integration points ($n = 3$) and single floating point precision, for the real part of the problem family defined by Eq. (2.41).

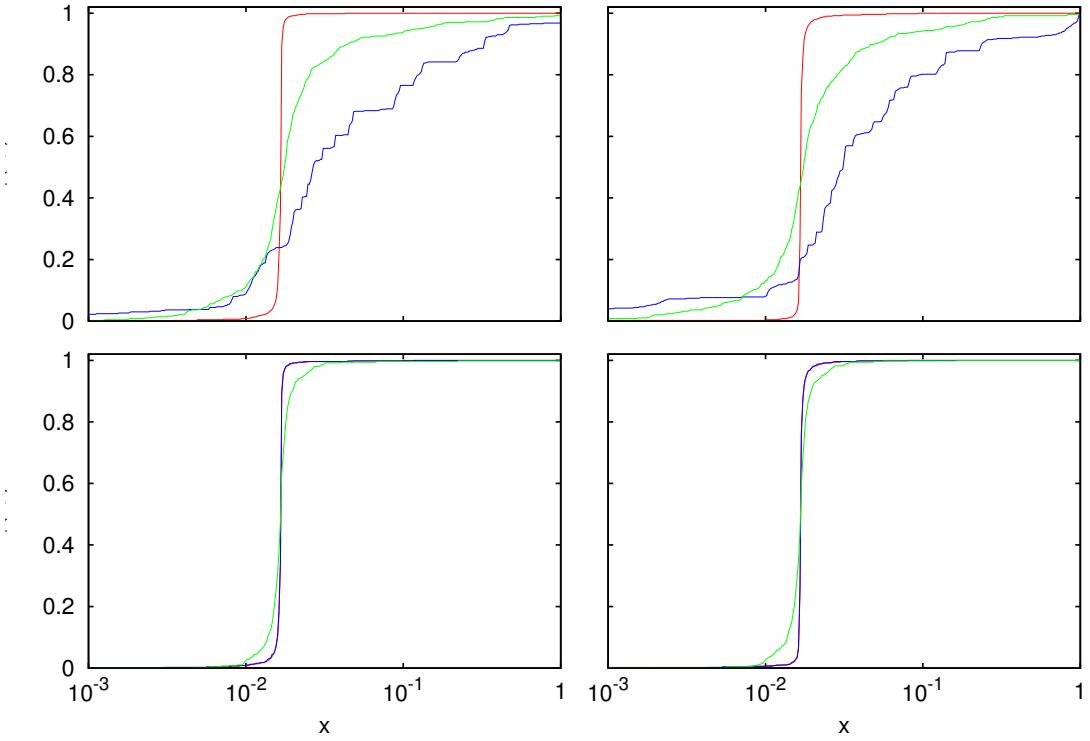

Figure D.7: Statistical distribution functions for the midpoint rule integrator routine, generated for the problem families defined by Eqs. (2.40) (top) and (2.41) (bottom). The distribution functions are generated for the real part (left) and imaginary part (right) of the approximated integrals, and for each of the surface profile systems in Tab. 2.1. This include the smooth surface (red), the rough surface (blue) and the very rough surface (green). The results almost completely overlap for the smooth and the rough surface systems in the bottom figures.

D.2 Statistical distribution functions

Figure D.7 depicts the statistical distribution functions for the real and imaginary parts of the problem families defined by Eqs. (2.40) and (2.41), for all test systems given in Tab. 2.1.

Figure D.8 show statistical distribution functions for the automatic integrator routine GPat for the imaginary part of the problem families defined by Eqs. (2.40) and (2.41). The test system is again the rough system in Tab. 2.1 with the observation point in the leftmost integration interval. Both single and double precision results are included. The requested accuracy $\varepsilon_{quad} = \varepsilon_{rel}$ and $\varepsilon_{abs} = 0$.

The average function value count, $n_{eval}(\varepsilon_{quad})$, used by GPat to compute the integrals in the problem family defined by Eq. (2.41), with the rough surface system defined in Tab. 2.1, can be seen in Fig. D.9.

Figures D.10 and D.11 depict statistical distribution functions for the automatic integrator GPat, generated for the real part of the problem family defined by Eq. (2.40). The figures are for the smooth surface profile and a very rough surface profile, respectively, as defined in Tab. 2.1. Both single and double precision results are included. The requested accuracy $\varepsilon_{quad} = \varepsilon_{rel}$ and $\varepsilon_{abs} = 0$. The corresponding results for the imaginary part of the problem family is not include, as they are close to identical to those in Figs. D.10 and

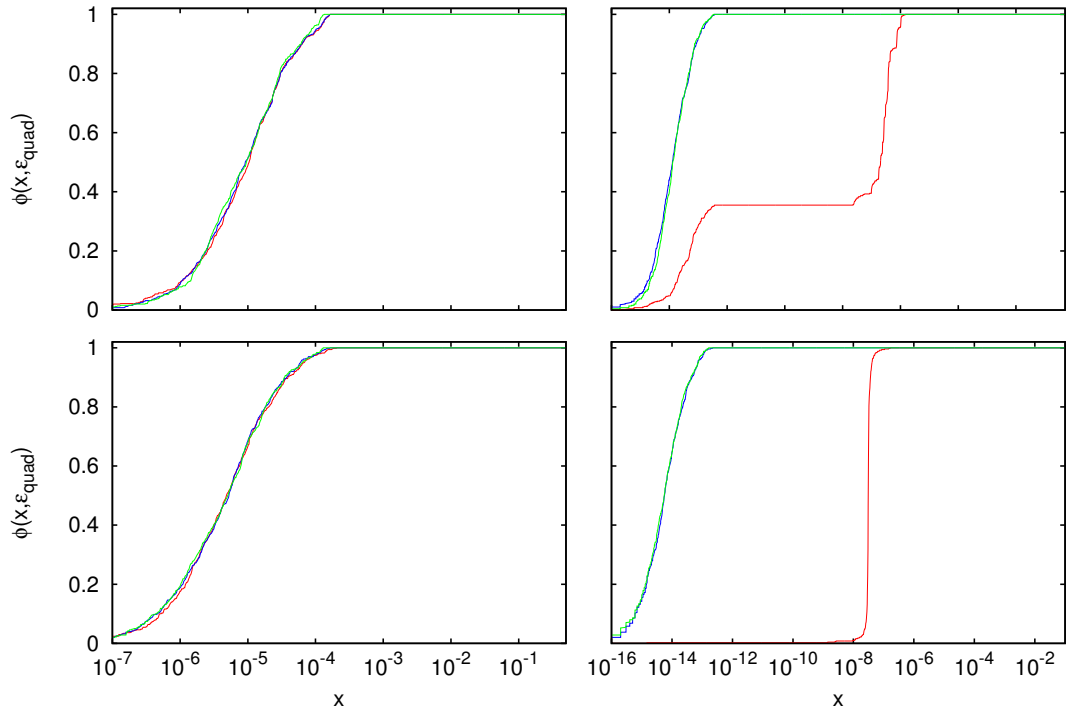

Figure D.8: Statistical distribution functions for GPat, for the imaginary part of the problem families defined by Eqs. (2.40) (top) and (2.41) (bottom), for the rough surface system in Tab. 2.1. Single precision results (left) given for ε_{quad} equal to 10^{-1} (red), 10^{-3} (blue) and $1.19 \cdot 10^{-7}$ (green). Double precision results (right) given for ε_{quad} equal to 10^{-1} (red), 10^{-2} (blue) and $2.22 \cdot 10^{-16}$ (green). All curves overlap to some extent in the single precision results, while the results the two most demanding accuracy requirements overlap in the double precision results.

D.11.

Figures D.12 and D.13 depict the statistical distribution function for the real part of the problem families defined by Eqs. (2.40) and (2.41), computed with GPat for the smooth and very rough surface profile, respectively, defined in Tab. 2.1. Both single and double floating point precision results are included. The single point precision results are computed with $\varepsilon_{quad} = 10^{-4}$, while the double precision computations use $\varepsilon_{quad} = 10^{-10}$. Again, the accuracy $\varepsilon_{quad} = \varepsilon_{rel}$, while the absolute accuracy is set to zero. The observation point is in the leftmost interval of integration.

Figures D.14 and D.15 depict statistical distribution functions for the rule evaluation routine GPatREQR for the real part of the problem families defined by Eqs. (2.40) and (2.41), respectively, for the smooth surface profile in Tab. 2.1. The observation point is $x_m = -14.97$. The figures depicts result for $n = 3$ and $n = 7$, for both single and double precision. Distribution functions for surface profile represented analytically, by Taylor approximations and by Hermite interpolation are included. In addition a single curve that represents the statistical distribution function for the midpoint rule is included. The corresponding results for the very rough surface system in Tab. 2.1 can be seen in Figs. D.16 and D.17.

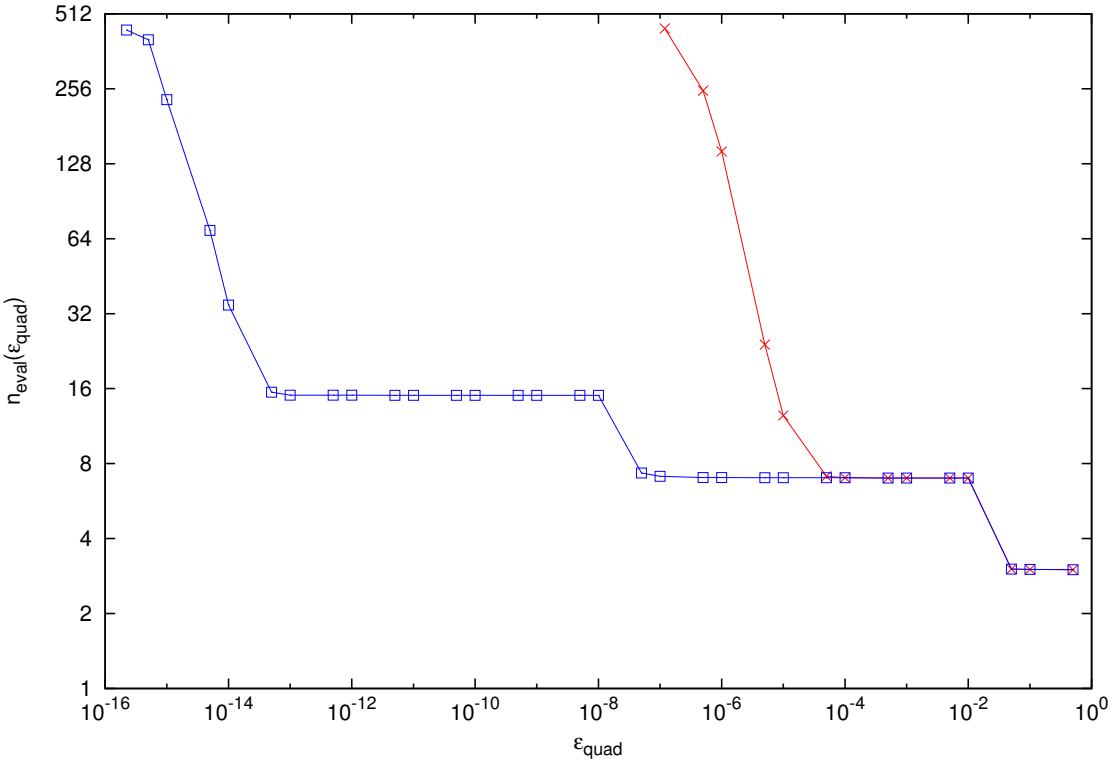

Figure D.9: The average function value count as a function of requested relative accuracy, for the problem family in Eq. (2.41) computed with GPat, for the rough surface system defined in Tab. 2.1. Single precision (red, x) and double precision (blue, □) results.

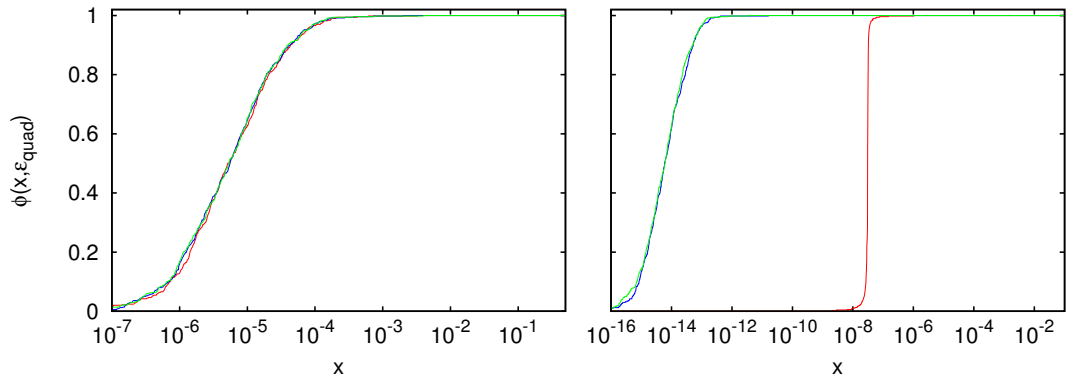

Figure D.10: Statistical distribution functions for GPat, for the real part of the problem families defined by Eq. (2.40), for the smooth surface system in Tab. 2.1. Otherwise the same as Fig. D.8.

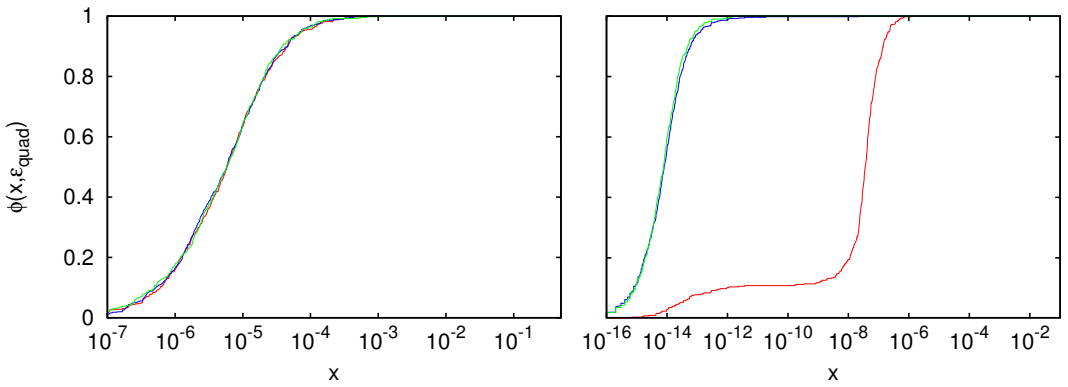

Figure D.11: Statistical distribution functions for GPat, for the real part of the problem families defined by Eq. (2.40), for the very rough surface system in Tab. 2.1. Otherwise the same as Fig. D.8.

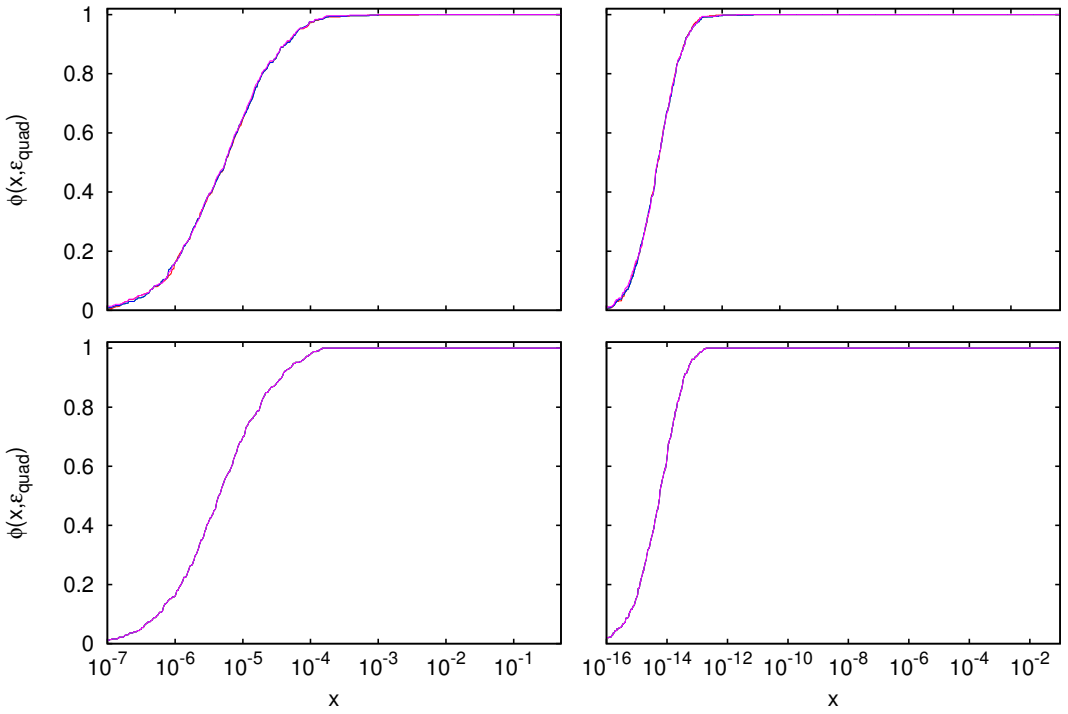

Figure D.12: Statistical distribution functions for the real part of the problem families defined by Eqs. (2.40) (top) and (2.41) (bottom) computed with GPat, for single precision (left, $\varepsilon_{quad} = 10^{-4}$) and double precision (right, $\varepsilon_{quad} = 10^{-10}$). The smooth surface profile (see Tab. 2.1) is represented analytically (red), by second order Taylor approximation (green), by third order Taylor approximation (blue) and by Hermite interpolation (pink). All curves overlap in the plotted results.

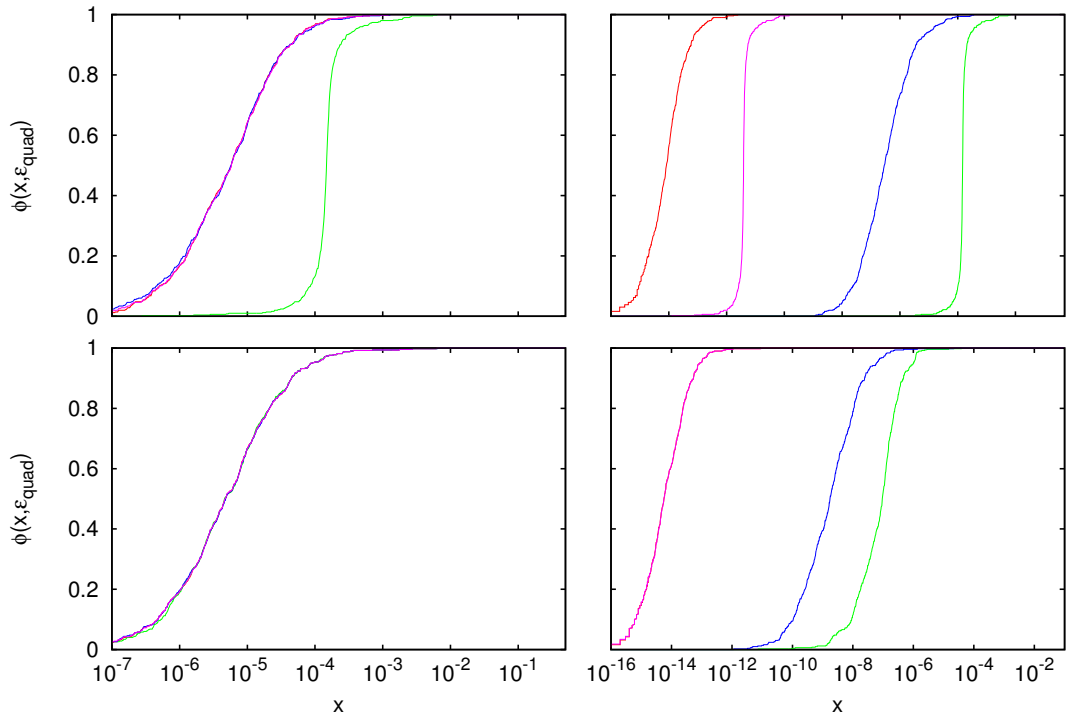

Figure D.13: Same as Fig. D.12, but for the very rough surface system in Tab. 2.1. Overlapping curves occur in three subfigures: The red, blue and pink in top left; all in bottom left; the red and pink in bottom right.

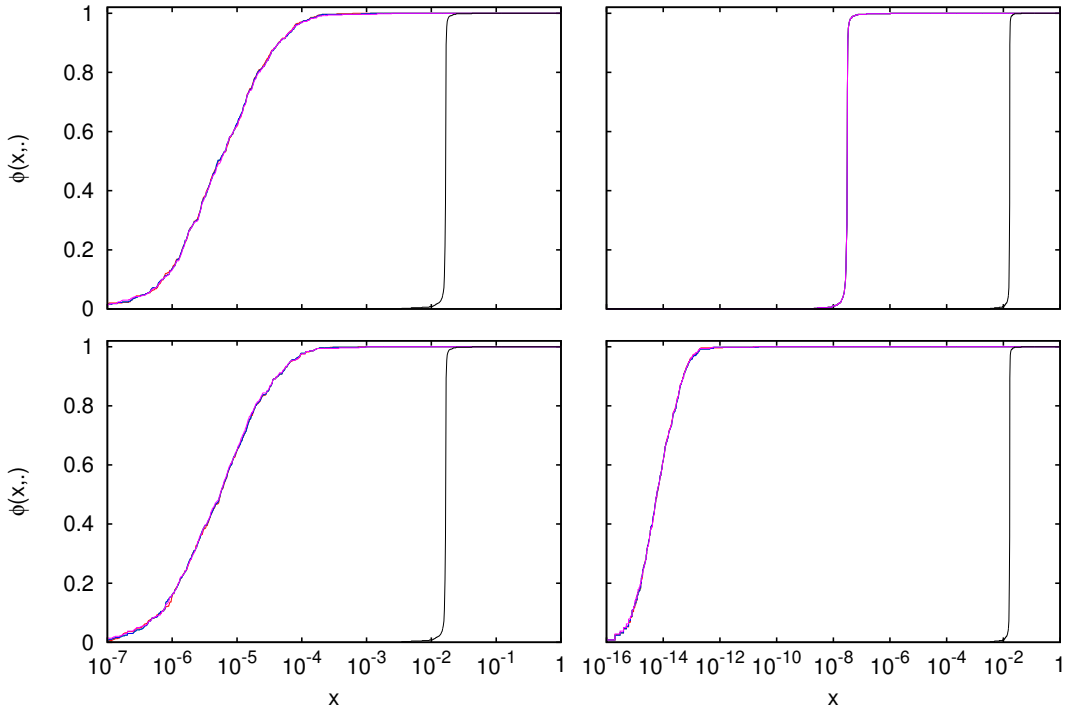

Figure D.14: Statistical distribution functions for GPatREQR with $n = 3$ (top) and $n = 7$ (bottom), for the real part of the problem family defined by Eq. (2.40), with the smooth surface test system in Tab. 2.1 and $x_m = -14.97$. Single precision (left) and double precision (right) results are shown. The curves are results for surface representations: analytic (red), second order Taylor approximation (green), third order Taylor (blue), hermite interpolation (pink). In addition a the midpoint rule's distribution function is included (black). All GPatREQR computed curves overlap in the figures.

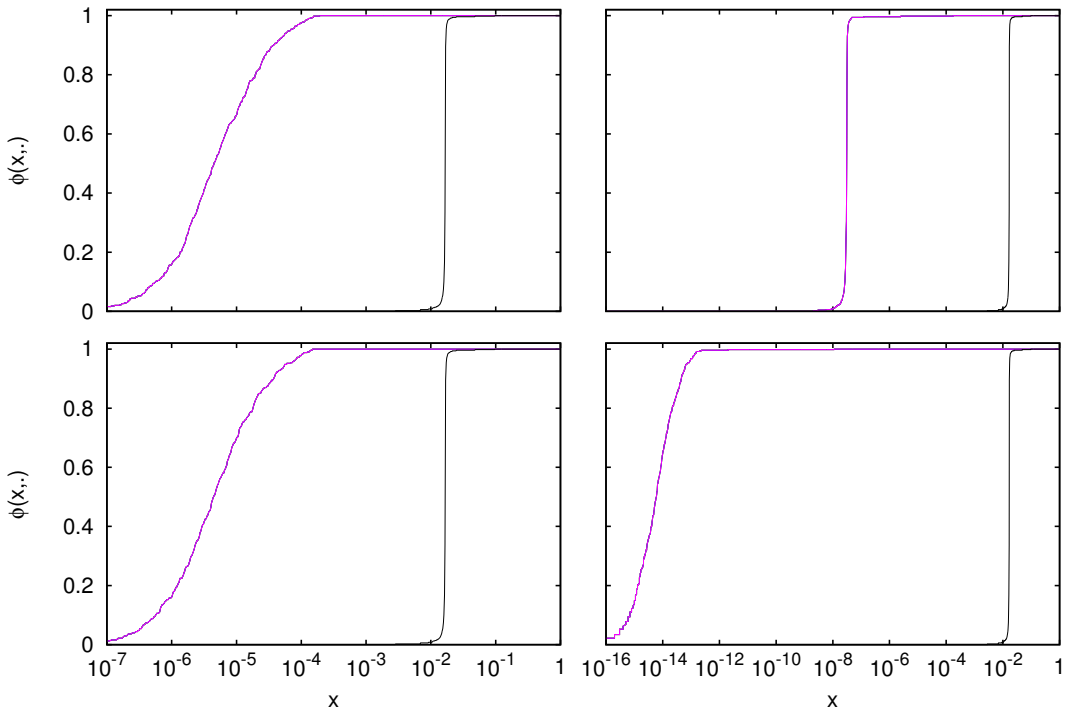

Figure D.15: Same as Fig. D.14, but for the problem family defined by Eq. (2.41).

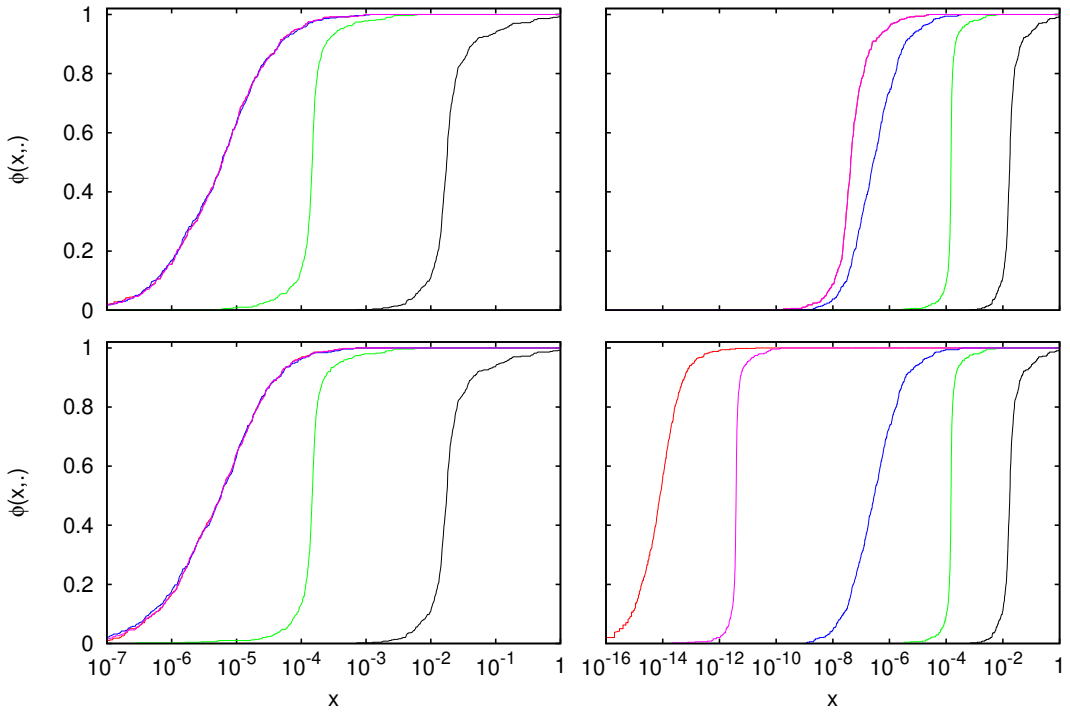

Figure D.16: Same as Fig. D.14, but for the very rough test system in Tab. 2.1. The red, blue and pink curves overlap in the left figures, and the red and pink curves in addition overlap in the top right figure.

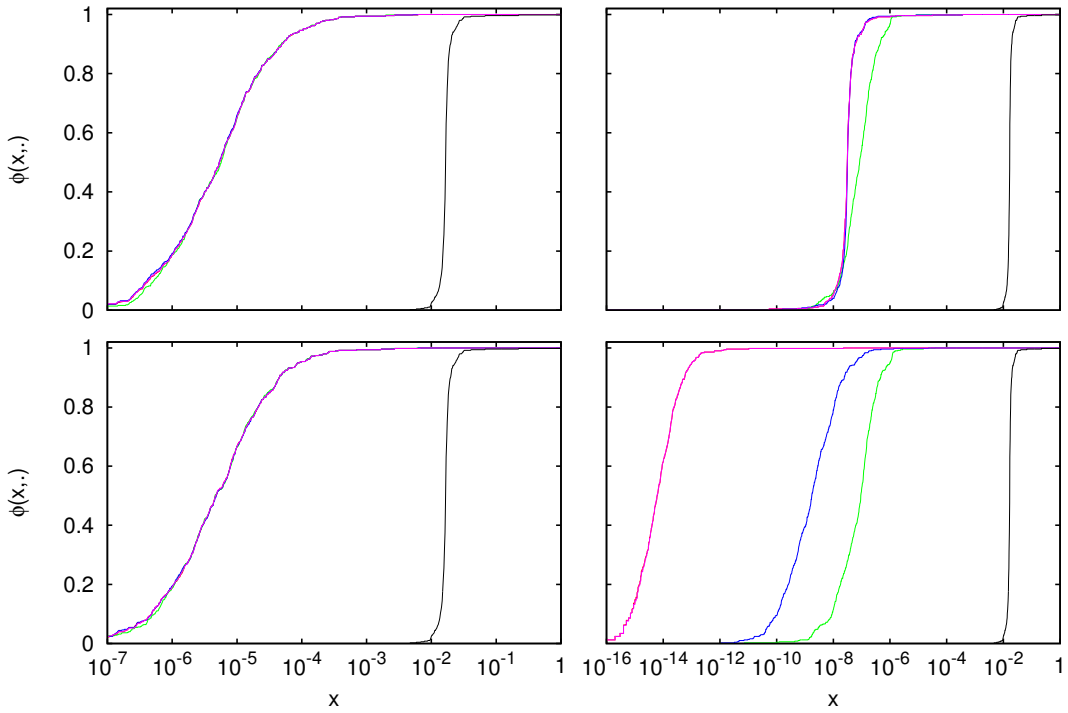

Figure D.17: Same as Fig. D.16, but for the problem family defined by Eq. (2.41). All GPatREQR computed curves overlap in the left frames, while the red, blue and pink overlap in the top right frame. The red and punk in addition overlap in the bottom right figure.

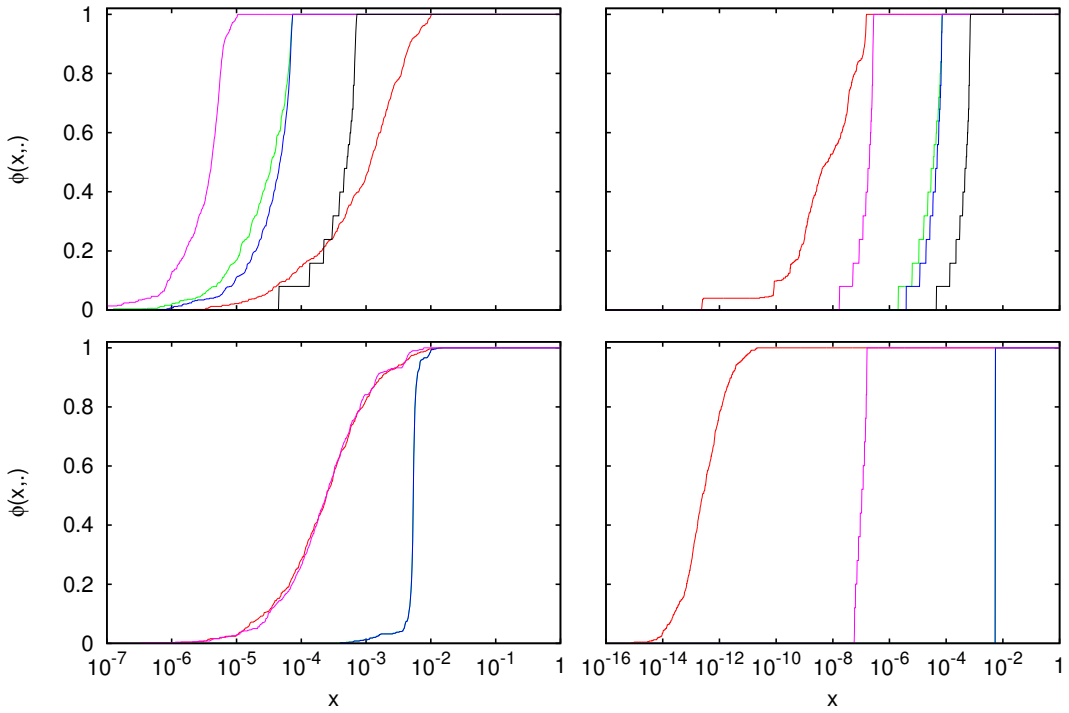

Figure D.18: Statistical distribution functions for the computation of the diagonal matrix elements $\mathcal{A}^{\pm}_{mm}$, with different surface representations and integration routines. GPat is used with the analytic surface representation (red), GPatREQR with $n = 3$ for the second order Taylor (green), third order Taylor (blue) and Hermite interpolated (pink) surface representation. Equation (2.46) is used with these routines, while the direct method (black) computes the elements with Eq. (2.25). Single precision (left) and double precision (right), for the real part (top) and imaginary part (bottom) of the results.

D.3 Diagonal element computations

Figures D.18 and D.19 depict the success probability in reaching an accuracy x, when computing the diagonal matrix elements $\mathcal{A}^{\pm}_{mm}$ and $\mathcal{B}^{\pm}_{mm}$, respectively. The results computed with an analytic surface representation have been computed with the automatic integrator routine, with $\varepsilon_{rel} = 10^{-4}$ and 10^{-10} for single and double precision computations, respectively. The Taylor approximated and Hermite interpolated surface representations have, on the other hand, been used with GPatREQR with three integration points for each of the two integrals in each of the diagonal elements $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$. The results computed directly with Eqs. (2.25) and (2.26) are also included in the figures.

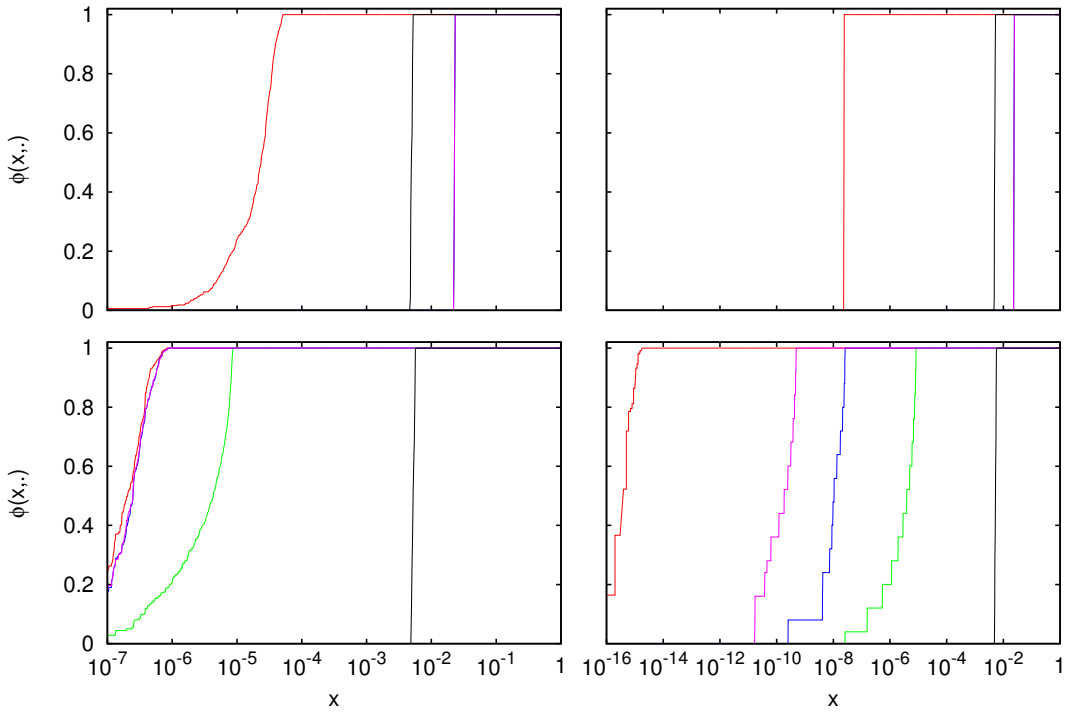

Figure D.19: Same as Fig. D.18, but for the diagonal elements $\mathcal{B}^{\pm}_{mm}$, computed by Eq. (2.47) with the Gauss-Patterson routines, and by Eq. (2.26) with the direct method.