asasim: Adaptive sampling for electromagnetic simulations

Sørensen, Kristian Tølbøl

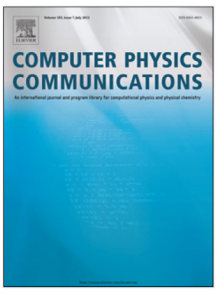
Published in:
Computer Physics Communications

Link to article, DOI:
10.1016/j.cpc.2018.10.017

Publication date:
2019

Document Version
Peer reviewed version

Accepted Manuscript



`asasim`: Adaptive sampling for electromagnetic simulations

Kristian Tølbøl Sørensen

PII: S0010-4655(18)30365-5
DOI: https://doi.org/10.1016/j.cpc.2018.10.017
Reference: COMPHY 6643

To appear in: *Computer Physics Communications*

Received date : 5 November 2017
Revised date : 10 October 2018
Accepted date : 14 October 2018

Please cite this article as: K.T. Sørensen, `asasim`: Adaptive sampling for electromagnetic simulations, *Computer Physics Communications* (2018), https://doi.org/10.1016/j.cpc.2018.10.017

`asasim`: Adaptive Sampling for Electromagnetic Simulations

Kristian Tølbøl Sørensen[a,*]

[a]*DTU Nanotech, Technical University of Denmark, Ørsteds Plads building 345C, 2800 Kgs. Lyngby, Denmark.*

Abstract

For simulations of electromagnetic resonance spectra, where the locations of spectral features are unknown, and for wide-band simulations in general, a substantial number of wavelengths must be simulated for acceptable resolution, increasing computation time. This problem is exacerbated for spectra containing narrow-band features, as a high spectral resolution is required to even detect them. To address this challenge, a heuristic algorithm is presented for electromagnetic simulations, which adaptively refines the local resolution of spectral features during a simulation. The method supports parallel processing and plugs in with existing simulation systems, such as rigorous coupled-wave analysis (RCWA). It can routinely reduce the computational load by two orders of magnitude.

Keywords: lorentzian; adaptive; resolution; rcwa; PACS 40; PACS 41

PROGRAM SUMMARY

Program Title: `asasim`

Program Files doi: `http://dx.doi.org/10.17632/d6gty7kr2x.1`

Licensing provisions: CC By 4.0

Programming language: MATLAB

Nature of problem: Simulations are challenging when information is needed both on a long scale (broad interval) and on a short scale (high local resolution), such as wide-band electromagnetic spectra containing narrow-band features. When resolution is insufficient, narrow-band features may be downright absent from the spectrum, if neighboring points are simulated on either side of a narrow peak. When local resolution is sufficient, it will necessarily be excessive in flat regions, wastefully increasing computation time.

Solution method: The presented method enables adaptive resolution, which ensures that

*Corresponding author.

E-mail address: kristian@tolbol.dk

all peaks of a given minimum width are always detected and maximally resolved, while feature-less regions remain minimally resolved. An optimum point spacing is derived for lorentzian peaks (descriptive of optical resonances) and is applied to optimize computation time.

1. Introduction

The ability to accurately simulate light-matter interactions in nanostructures has enabled breakthroughs in areas as diverse as optical biosensors [1], pigment-free coloration [2], and solar cells [3]. Rigorous coupled-wave analysis (RCWA) is a popular semi-analytical method for electromagnetic simulations originally described by Moharam and Gaylord in 1981 [4]. However, the method is computationally demanding, and this can be a limitation for high-resolution, wide-band simulations. This can be particularly problematic for optimization methods, such as particle swarm optimization [5] or genetic algorithms [6], where an extensive number of simulations in a many-dimensional parameter space should be performed. Such endeavors would benefit from increased simulation efficiency.

The challenges of multiscale simulations have similarly been encountered in other fields, where more intricate schemes have been demonstrated, e.g., for elastodynamic shock propagation[7] or particle–particle interactions[8].

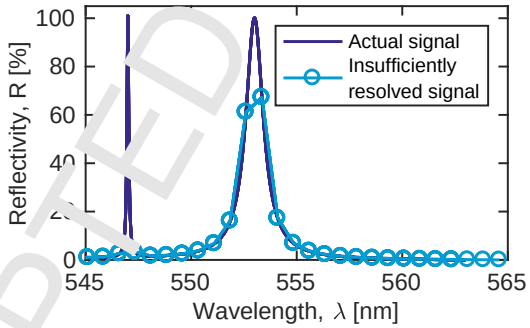


Figure 1: Illustration of detection problem. At insufficient resolution, the peak at 547 nm is not registered at all, as the neighboring points may randomly fall on either side of it. The much wider peak at 553 nm is certain to be detected at this resolution, but it is still poorly resolved.

Simulation time can be reduced by simply reducing the spectral resolution. However, when simulation resolution becomes too low, narrow peaks may not even be detected, as illustrated in figure 1. Depending on whether a data point happens to

fall on the narrow peak, it may or may not register as a bump, but there will be a risk that the peak is completely absent from the simulation, which can be quite misleading.

As also illustrated in figure 1, peaks of sufficient width will surely be detected, as at least one point will always fall on the peak. However, resolution may still be too low to properly resolve its lineshape, which is commonly a simulation goal. An obvious solution is to increase the spectral resolution of the simulation, proportionally increasing simulation time, but this would result in an unnecessarily high resolution in the flat parts of the spectrum. Figure 2A illustrates this central issue, i.e., spectrally flat regions are over-emphasized, whereas regions with features are under-emphasized. Ideally, an initial simulation should only be fine-grained enough that the presence of a peak would always be detected, and peak regions should then be further resolved to the desired resolution.

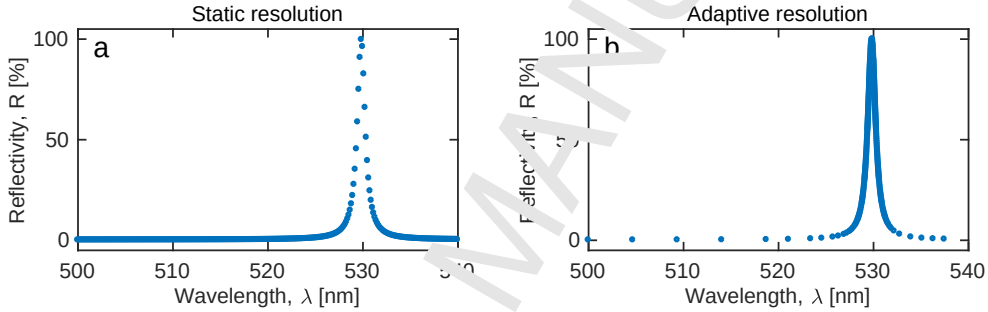


Figure 2: The same peak simulated with 209 points using a) static resolution, and b) adaptive resolution. The static method yields a high information density off-peak, whereas the adaptive method emphasizes the peak region.

Here, a heuristic MATLAB-algorithm is demonstrated for achieving adaptive resolution in electromagnetic simulations. This effectively reverses the information emphasis to lay on the spectral features rather than the background, as illustrated in figure 2b. Furthermore, optimal parameters are derived, and the speed of the method is evaluated.

2. Examples and installation

At its core, the `asasim` algorithm simply replaces the per-wavelength `for`-loop of a typical electromagnetic simulation system. Thus, instead of statically looping over all wavelengths in an interval with identical spacing, as is the common approach, `asasim` evaluates which regions to refine during runtime, thereby only refining regions

where features are present. Therefore, in principle, any script that calls a simulation function from a `for`-loop can be integrated with `asasim`. The system performs best for "needle-in-a-haystack"-type simulations, where narrow-band features are found in a broad interval. As this is common for electromagnetic resonance spectra, these are the focus of this paper, although the system will likely also be applicable to many other topics of simulation. This section explains how `asasim` is integrated with existing simulation systems in MATLAB 2016b, running on a MacBook Pro (2.4 GHz Intel Core i5, 8 GB 1600 MHz DDR3 RAM).

2.1. A simple test

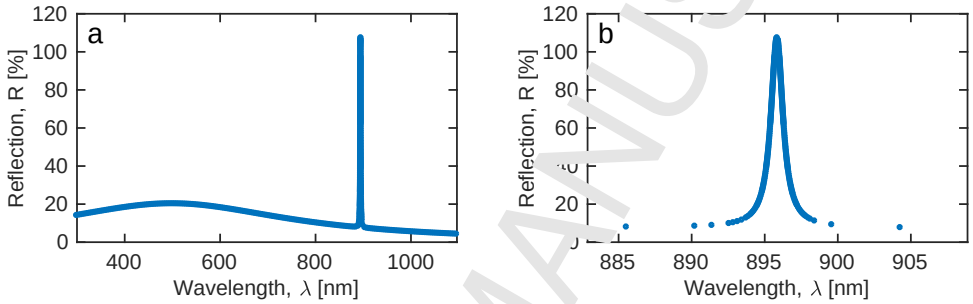


Figure 3: Output of example script, which imitates an electromagnetic simulation at a random position a) within a wide interval, with b) narrow line width. The resolution varies from 4.6 pm on the peak, to 4.6 nm in the flat parts of the spectrum. Achieving the same resolution with static spacing would require 370× as many points.

As actual electromagnetic simulations can be complex to set up and time-consuming to run, a test-function was written that imitates an actual simulation function, but returns values from an analytical evaluation of lorentzians. The script `asasim_Example_1_Imitator.m` exemplifies how the `asasim` method may be integrated with an arbitrary simulation function. A lorentzian of narrow line width (0.5 nm) is simulated at a random location within the broad interval 300–1100 nm, on a slightly sloped background, given by a broad lorentzian. The time to run is ∼1 second, and figure 3 shows the results.

2.2. Photonic crystal slab sensor at varying angles of incidence

GD-Calc is a MATLAB-package for RCWA, which can be downloaded from `http://kjinnovation.com/GD-Calc.html` and installed as per the instructions given on the website. The workhorse of the GD-Calc package is the function `gdc.m`, which takes three inputs: a `Grating`-struct with the relevant optical grating parameters and

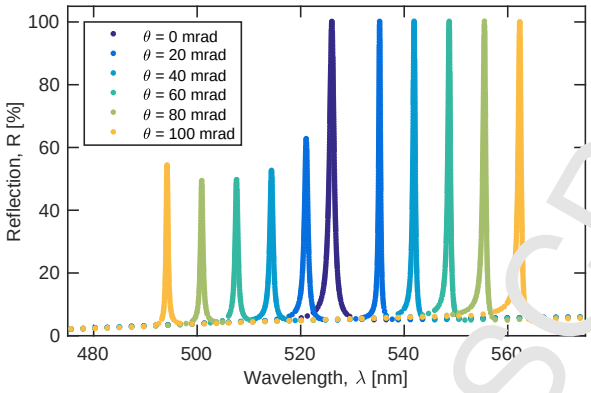


Figure 4: Example simulation of PCS sensor at varying angles of incidence. Each simulation has a variable resolution between 2 pm and 2342 pm depending on the local region.

geometric layout, an `IncField`-struct with information about the incoming field, and an `order`-matrix, specifying the diffraction orders to be used in the calculations. The function `gdcWrapper` has been written to interface `asasim` with GD-Calc. It defines these necessary three inputs and calls gdc.m to get a reflectivity value `R` at the given input wavelength `w`. An additional `FullOutput` struct contains all diffraction efficiencies. `asasim` uses this reflectivity value `R` to evaluate whether points neighboring `w` should be further resolved. The `asasim` system was recently used with RCWA to substantiate experimental observations of waveguide core swelling [9] in a photonic crystal slab (PCS) sensor. Here, the simulation parameters describe a linear grating of period 368 nm, duty cycle 50% and grating height 100 nm, illuminated at an angle θ. The model incorporates refractive index dispersion data for all three materials constituting the sensor, namely a cladding layer of Efiron PC409AP (Luvantix, Korea), a nano-structured core layer of HI01XP (micro resist technologies, Germany) and water as superstrate. In the example given here, simulations are performed in a broad wavelength interval of 450–850 nm, with angles of incidence in steps of 0.02 rad between 0 and 0.1 rad. The full example code is given in `asasim_Example_2_GDC.m`, and the result is shown in figure 4. It should be noted that `asasim` is incapable of displaying a progress bar during runtime due to parallel processing working asynchronously, and because the wavelengths to simulate are being refined adaptively, and are thus not known *a priori*. Instead, the total number of points calculated is displayed for each round to indicate progress. On the computer used here, the total run time for all 6 angles is ∼10 min.

2.3. Scattering cross section of plasmonic nanoparticles

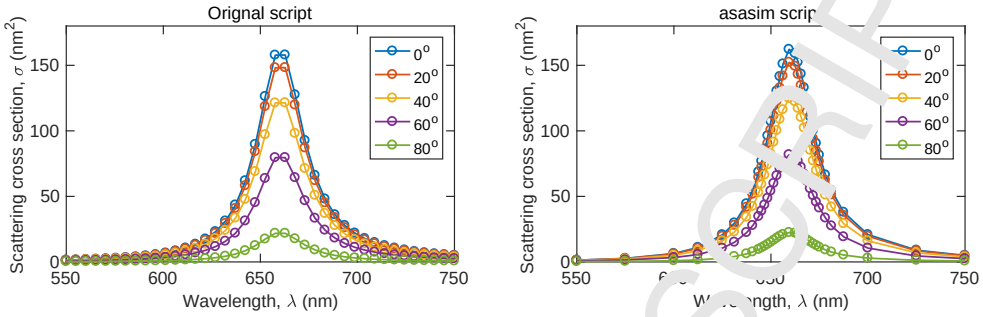


Figure 5: Simulation performed using the `mnpbem`-package. a) Using the default static spacing places 200 points, many of them off-peak. b) With adaptive resolution, the peaks are better resolved using only 165 points.

To demonstrate the versatility of `asasim`, it was also integrated with the excellent class `mnpbem`[10], available at `http://cpc.cs.qub.ac.uk/summaries/AEKJ_v3_0.html`, which is used for calculating scattering cross sections of plasmonic nanoparticles near surfaces. An example script, based the demonstration `demospecstat1.m` of that toolbox, is given in `asasim_Example_3_MNP.m`. Here, the per-wavelength `for`-loop is quite simply replaced by a call to `asasim` using `mnpWrapper`, which is functionally three lines long. As mentioned in section 2.4, the wrapper must take two inputs and return two outputs, and it serves as an example of how simply a wrapper function can be written. The resulting graph is shown in figure 5, and although there is perhaps limited gain from using `asasim` for simulating such wide peaks in a limited interval, the example illustrates how additional simulation systems can be interfaced using a wrapper function. The code has a typical runtime of ∼3 seconds.

2.4. Installation

The contents of `asasim.zip` should be decompressed to a folder on the MATLAB search path, such as `MATLAB/Toolboxes/asasim`. With the MATLAB-folder as the working directory, scripts should always contain the command `addpath(genpath(pwd))` in order to add all files in all subfolders to the search path. This line is included in all example files.

The main system for adaptive resolution is now installed, and is ready to be interfaced with an existing simulation system. In general, conversion of a simulation script (listing 1) into a version utilizing adaptive resolution (listing 2), is a three-step operation:

Listing 1: Original code

```
dx = 1; % desired x-axis resolution
x = 400:dx:700;
for ix = 1:numel(x)
  y(ix) = simulateR(x(ix),input1,input2,input3);
end
plot(x,y);
```

Listing 2: asasim-implementation, illustrating how a wrapper-function may be defined.

```
minHalfWidth = 0.1;
dy = 0.01; % desired y-axis resolution
dx = optimalLorentzianSpacing(minHalfWidth,dy);
xInitial = 400:dx:700;
wrapFunction = @(x,S) simulateR(x,S.in1,S.in2,S.in3);
SimInput = struct('in1',input1,'in2',input2,'in3',input3);
[x,y] = asasim(xInitial,dy,wrapFunction,SimInput);
plot(x,y);
```

1. Define wrapper. Copy the content of the per-wavelength `for`-loop into a wrapper function.
2. Determine initial point spacing. For peaks described as lorentzians ranging from 0 to 100% intensity, the optimal point spacing can be calculated automatically using the `optimalLorentzianSpacing` function.
3. Replace the per-wavelength `for`-loop by a call to `asasim`.

Each of these steps will be elaborated next in the context of concrete application examples.

The purpose of the wrapper function is to bridge the simulation system with `asasim`, and must take two inputs, namely the simulation coordinate `x` (e.g., wavelength) and a struct of additional parameters. The wrapper function passes these parameters on to the simulation function(s), using the specific syntax of that simulation system. In general, the content of a per-wavelength `for`-loop can often just be excised and placed in a wrapper function, and the `for`-loop itself is then replaced by a call to `asasim`. This was first exemplified in the simple case above, where only a single hypothetical function `simulateR` was called from within the `for`-loop. Subsequent examples demonstrated how more involved cases could be handled.

For simulations where the peak-shape resembles a lorentzian with a maximum intensity of 100%, the optimum initial point spacing `dx` can be calculated by calling `optimalLorentzianSpacing`, where `minHalfWidth` is the minimum lorentzian half-width to be detected, and `yRes` is the desired y-axis resolution. Using this point

spacing, a vector of points to be calculated initially is then defined as `xInitial = x1:dx:x2`, where `x1` and `x2` represent the extremes of the interval to simulate. Alternatively, if the spectrum does not contain lorentzian peaks at 100% maximum, arbitrary values for `dy` and `xInitial` can be defined manually, using a custom point spacing `dx`. The following section explains how and why an optimal point spacing is found.

3. Background

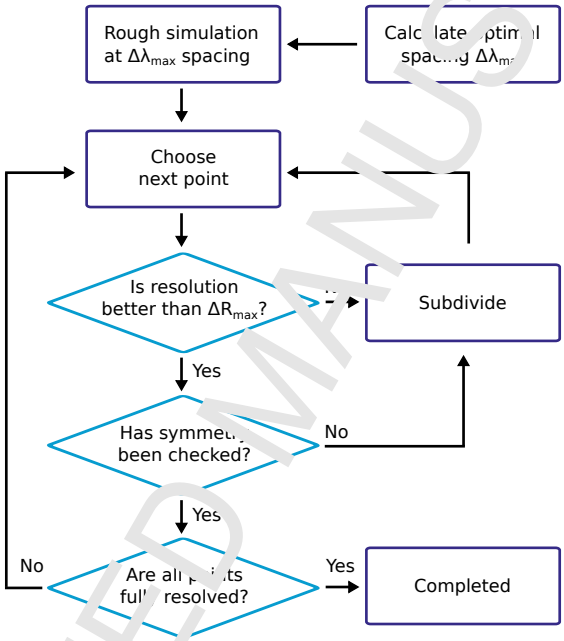


Figure 6: Working principle behind the algorithm. An optimal point spacing is calculated using equation 5, and an initial rough simulation is performed in order to detect all relevant features. Then, each peak is further resolved by adaptive subdivisioning until the desired resolution is achieved. At this point, a symmetry-check is performed in order to ensure that two points with similar values are not just placed symmetrically around a peak.

A simplified illustration of the working principle is presented in figure 6, with the goal of producing a (λ, R)-spectrum using $\Delta R_{\max}$ (goal resolution) and $\gamma_{\min}$ (minimum peak half-width) as input parameters. First, an optimum initial point spacing $\Delta\lambda_{\max}$ is automatically calculated, such that any peak of a given minimum half-width $\gamma_{\min}$ is certain to cause a perturbation exceeding the threshold $\Delta R_{\max}$ within

the interval, flagging the region for further refinement. Then, a rough simulation is performed, with the purpose of detecting all spectral features of interest.

After each round of simulations, the difference ΔR between neighboring points is evaluated, to identify regions that exceed $\Delta R_{\max}$. These will be further refined by subdivisioning. New points inherit the ΔR-value of its parent in that round, such that subdivisioning does not continue indefinitely.

When a point no longer exceeds $\Delta R_{\max}$, it is subdivided one last time as a symmetry-check. It is entirely possible for two points to be placed symmetrically around a peak, in which case their difference ΔR could be zero, without the upper part of the peak having been resolved. The symmetry-check safe-guards against this. If the difference still does not exceed $\Delta R_{\max}$, the region is considered fully resolved. Thus, once a peak is detected, i.e., at least one point satisfies the criterion $\Delta R > \Delta R_{\max}$, the entire peak always becomes fully resolved.

For the sake of argument, consider a lorentzian at an arbitrary location, i.e., $\lambda_0 = 0$, normalized so that $R(\lambda_0) = 100\%$:

$$R(\lambda) = \frac{1}{1 + \frac{\lambda^2}{\gamma^2}} \tag{1}$$

$$\lambda = \sqrt{\gamma^2 \left(\frac{1}{R} - 1 \right)} \tag{2}$$

Because of the subdivisioning-scheme employed, whenever a peak is detected, it is certain to come out fully resolved. Detection in this context entails that the perturbation from a peak causes two neighboring points, spaced apart by $\Delta\lambda$ on the first axis, to have a sufficient difference on the second axis, $\Delta R > \Delta R_{\max}$. The narrowest peak of half-width γ that is certain to be detected is then a peak that is so narrow, that its perturbation only just causes ΔR between any two neighboring points to exceed $\Delta R_{\max}$, even when the peak is placed right between those two neighboring points, such as points a and b in figure 7. In this case, ΔR is zero between them, so for detection, the difference to the *next* neighbor (point c) must instead satisfy $\Delta R > \Delta R_{\max}$. If the distance from the peak center to the first symmetrically placed neighbor (point b) is $\lambda_1 = \Delta\lambda/2$, then the distance to its next neighbor (point c) must be $3\lambda_1$. Thus,

$$\Delta R = R_1 - R_2 \tag{3}$$

$$= \frac{1}{1 + \frac{\lambda_1^2}{\gamma^2}} - \frac{1}{1 + \frac{(3\lambda_1)^2}{\gamma^2}} \tag{4}$$

This does not have a simple solution for the optimal point spacing $\Delta\lambda = 2\lambda_1$, but it can be isolated as

$$\Delta\lambda = \left(\frac{2}{3} \sqrt{\frac{\gamma^2(4 - 5\Delta R) + 2\sqrt{2}\sqrt{\gamma^4(2\Delta R^2 - 5\Delta R + 2)}}{\Delta R}} \right) \tag{5}$$

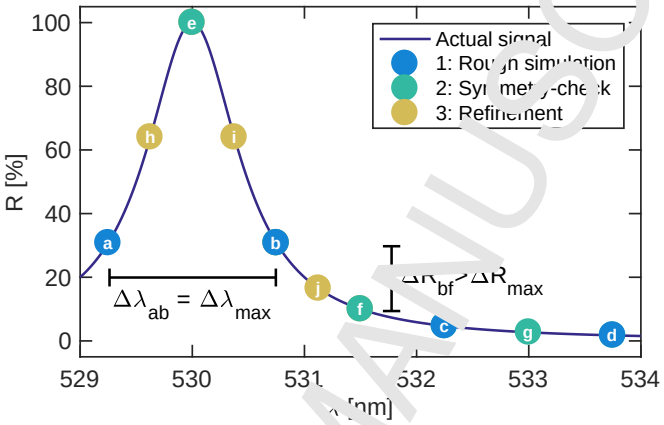


Figure 7: Illustration of points placed in the first three rounds of simulations. a-d) An initial rough-simulation is performed in order to detect all relevant features. e-g) A symmetry-check is performed in order to ensure that two points with similar values are not just placed symmetrically around the peak (like points a and b). h-j) Peak refinement continues until a desired resolution is achieved. Note that here, the spectrum beyond $\lambda > 531.3$ nm is therefore not refined after round two.

4. Discussion

The optimum ratio between the parameters $\Delta\lambda_{\max}$ and $\Delta R_{\max}$ for a given $\gamma_{\min}$ is given by equation 5, and shown in figure 8. Parameter-sets below the curve are sub-optimal in the sense that they cause more simulation points to be calculated than necessary, wastefully increasing computation time. Sets above the curve will only serendipitously resolve peaks of a given half-width $\gamma_{\min}$. As an example, if a simulation is to be performed with a 5% resolution on the y-axis, and the expected minimum half-width $\gamma_{\min}$ is 1 nm, the optimum spacing on the x-axis is 8.2 nm. At a resolution of 0.5%, the spacing can be 26.6 nm, while still detecting the perturbation from the narrow peak.

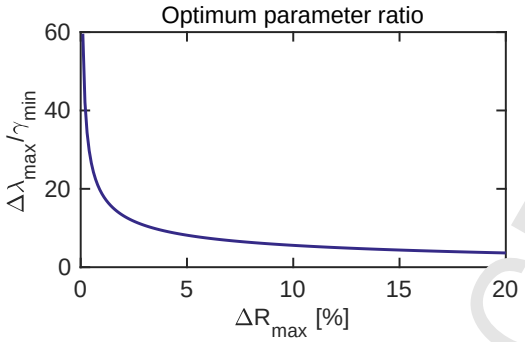


Figure 8: Optimum ratio of simulation parameters.

Decreasing $\Delta R_{\max}$ causes more points to be calculated on the R-axis, but this also increases $\Delta\lambda_{\max}$, reducing the number of points to be calculated initially on the λ-axis. In the simplest possible model, consider a spectrum only containing a single lorentzian peak with a half-width of γ. The number of points simulated on the fully resolved peak is $N_R = 2/\Delta R$, and the number of points to be simulated statically across the spectrum is $N_\lambda = (\lambda_{\max} - \lambda_{\min})/\Delta\lambda$. The total number of points to be simulated adaptively is

$$N = \frac{2}{\Delta R_{\max}} \frac{\lambda_{\max} - \lambda_{\min}}{\Delta\lambda_{\max}} \tag{6}$$

Using this equation for estimating the number of simulation points, the two methods were compared for speed as shown in figure 9. It is clear that the adaptive method is generally a couple of orders of magnitude faster than the static method. The figure also illustrates the computational optimum (curve minimum) for the ΔR and thus $\Delta\lambda$ parameters, which depend on the minimum necessary peak half-width $\gamma_{\min}$. Time-optimal parameters should be determined by combination of equations 5 and 6 and solving $\frac{dN}{d\Delta R_{\max}} = 0$, but this becomes rather unwieldy. As figure 9 indicates, the total number of simulated points does not vary steeply for similar values of ΔR, and so the choice of resolution is perhaps more a question of preference.

For comparison, in order to achieve the same R-axis resolution with static sampling as with adaptive sampling, the static first-axis point spacing must equal the smallest distance between two points separated by $\Delta R_{\max}$ on the second axis. For example, to resolve a peak of half-width $\gamma = 0.5$ nm at $\Delta R = 1\%$ resolution on the steepest part in a 800 nm interval, the adaptive resolution varies between 0.0046 nm and 4.6 nm depending on the local spectral features. To achieve a static resolution of 0.004 nm, more than 170,000 points would be required. With adaptive

resolution, the same is achieved with 468 points, making the simulation 374× faster.

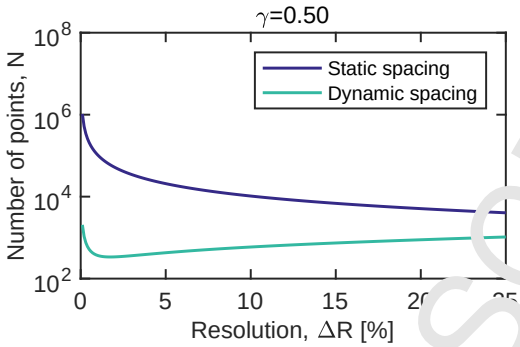


Figure 9: Number of points simulated as function of R-axis resolution.

Apart from reducing computation time, a main advantage of `asasim` is the decreased amount of *a priori* information needed. As spectrally flat regions are computed very fast, the precise spectral positions of features need not be known beforehand in order to simulate the narrow regions of relevance. Furthermore, whereas one might have to iteratively adjust first-axis resolution in order to achieve the desired second-axis resolution, with this method, resolution is decided for the second-axis directly.

While the amount of necessary *a priori* information is reduced, a rough estimate of the smallest realistic half-width is still required. The consequences of choosing a poor value for this parameter was discussed in section 4. For very dense spectra, e.g., containing many closely spaced peaks such as interference patterns, the same amount of points may end up being simulated as using static spacing. Furthermore, when the background is strongly sloped, e.g. the resonance peak of interest resembles a bump on a larger and much broader peak, the background also becomes highly resolved. This is partially the case in figure 5.

Because the x-axis resolution varies across all spectra, direct comparison between spectra or presentation of data as an image will require interpolation. This is quite simply achieved using the built-in MATLAB-function `interp1`, as exemplified below.

5. Summary

In summary, the presented method allows high-speed, high-resolution simulation of narrow-band spectral features in a broad range, with no *a priori* information about the location of spectral features. In one example, the number of points necessary to simulate was reduced from >170,000 to 468, with an accompanying reduction in

Listing 3: Interpolation

```
dx = 0.01; % desired x-axis resolution
interpolatedX = min([Sim.xs]):dx:max([Sim.xs]);
interpolatedY = zeros(numel(Sim),numel(
   interpolatedX));
for iSim = 1:numel(Sim)
  interpolatedY(iSim,:) = interp1(Sim(iSim).xs,
     Sim(iSim).ys,interpolatedX,'linear');
end
imagesc(interpolatedX,1:numel(Sim),interpolatedY);
```

computation time from ∼20 hours to ∼3 minutes. The solution is tailored to simulations of electromagnetic spectra containing Lorentzian features, but can trivially be adapted to other simulation functions and other lineshapes. The solution easily plugs into existing simulation systems, and interfacing to three different simulation systems was demonstrated by examples.

Acknowledgements

The author gratefully acknowledges funding from the Danish Council for Strategic Research, DSF, under Grant Agreement No. 10-092322 (PolyNano).

References

[1] K. Byun, S. Kim, D. Kim, Design study of highly sensitive nanowire-enhanced surface plasmon resonance biosensors using rigorous coupled wave analysis., Optics express 13 (10) (2005) 3737–3742. doi:10.1364/OPEX.13.003737.

[2] E. Højlund-Nielsen, J. Clausen, T. Mäkela, L. H. Thamdrup, M. Zalkovskij, T. Nielsen, N. Li Pira, J. Ahopelto, N. A. Mortensen, A. Kristensen, Plasmonic Colors: Toward Mass Production of Metasurfaces, Advanced Materials Technologies 1 (7) (2016) 1600054. doi:10.1002/admt.201600054.
URL http://doi.wiley.com/10.1002/admt.201600054

[3] S. A. Boden, D. M. Bagnall, Optimization of moth-eye antireflection schemes for silicon solar cells, Progress in Photovoltaics: Research and Applications 18 (3) (2010) 195–203. doi:10.1002/pip.951.

[4] M. G. Moharam, T. K. Gaylord, Rigorous coupled-wave analysis of planar-grating diffraction, Journal of the Optical Society of America 71 (7) (1981) 811. doi:10.1364/JOSA.71.000811.
URL https://www.osapublishing.org/abstract.cfm?URI=josa-71-7-811

[5] M. Shokooh-saremi, R. Magnusson, Particle swarm optimization and its application to the Design of Diffraction Grating Filters, Optics Letters 32 (8) (2007) 894–896.

[6] C. Wang, S. Yu, W. Chen, C. Sun, Highly Efficient Light-Trapping Structure Design Inspired By Natural Evolution, Scientific Reports 3 (1) (2013) 1025. doi:10.1038/srep01025.
URL http://www.nature.com/articles/srep01025

[7] B. Rouet-Leduc, K. Barros, E. Cieren, V. Elango, C. Junghans, T. Lookman, J. Mohd-Yusof, R. S. Pavel, A. Y. Rivera, D. Roehm, A. L. McPherson, T. C. Germann, Spatial adaptive sampling in multiscale simulation, Computer Physics Communications 185 (7) (2014) 1857–1864. doi:10.1016/j.cpc.2014.03.011.
URL http://dx.doi.org/10.1016/j.cpc.2014.03.011

[8] O. Awile, F. Büyükkeçeci, S. Reboux, I. F. Sbalzarini, Fast neighbor lists for adaptive-resolution particle simulations, Computer Physics Communications 183 (5) (2012) 1073–1081. doi:10.1016/j.cpc.2012.01.003.
URL http://dx.doi.org/10.1016/j.cpc.2012.01.003

[9] K. T. Sørensen, C. B. Ingvorsen, L. H. Nielsen, A. Kristensen, Effects of water-absorption and thermal drift on a polymeric photonic crystal slab sensor, Optics Express 26 (5) (2018) 5416–5422. doi:10.1364/OE.26.005416.

[10] J. Waxenegger, A. Trügler, U. Hohenester, Plasmonics simulations with the MNPBEM toolbox: Consideration of substrates and layer structures, Computer Physics Communications 193 (2015) 138–150. arXiv:1412.5467, doi:10.1016/j.cpc.2015.03.023.
URL http://dx.doi.org/10.1016/j.cpc.2015.03.023