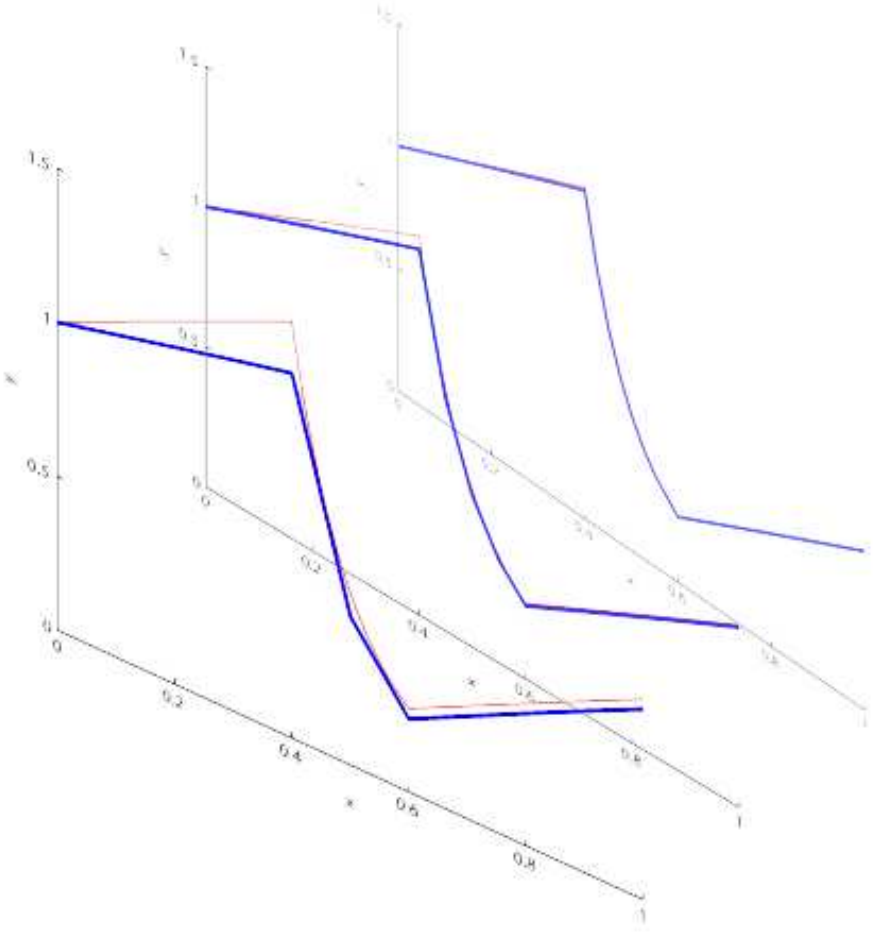
Adaptive mesh refinement for neutron transport

TN2982 Bachelor Thesis
M.J. van Nijhuis 1260154
supervisor: D. Lathouwers

August 2011



Abstract

A study of an adaptive mesh refinement (AMR) algorithm to solve the neutron transport equation is presented in this report. The goal is to see whether and how AMR can be used to solve the neutron transport equation. AMR is a method in which the grid is refined or coarsened depending on the error of the solution on the elements. The error estimate is a least squares functional in this case. The numerical solutions found using AMR have a higher accuracy and resolution while they require less memory and have shorter calculation times, which is cheaper.
In this report two test problems are used to test the AMR algorithm. It turns out that the AMR implementation in the algorithm works, but the functional is not a good error estimate, because it doesn't refine all regions with a large error. The solution to the differential equation is not the right one, because the least squares functional used for solving is not the most accurate way of finding a solution.

Contents

Chapter 1

Introduction

Neutron transport is studied by a lot of scientists and engineers. It is often approximated by numerical methods. The neutron transport equation is a difficult equation with a lot of parameters. Accuracy is more and more requested, which means finer grids have to be used. Finer grids use more computer memory and have longer computation times. The fast increasing use of these tools makes it necessary to have shorter calculation times in simulation runs on computers. This is cheaper and therefore research is done to find ways to keep or even improve the accuracy and reduce the numerical cost.
Adaptive mesh refinement (AMR) is nowadays used in many engineering disciplines. Adaptivity is used since the late 1970's [3]. It is based on the idea that in order to achieve high accuracy, a uniformly fine mesh is not necessarily required. The computational grid only needs to be fine in areas where the solution is rough and can be coarse in regions where the solution is smooth. Therefore larger cells suffice in those areas. In this report it is all about the refinement of the grid.

Two test problems were used in this research. Both are written in Matlab code. The first was an area with not so much changes in the cross section and source. The second problem was a more difficult one with more changes. The first test problem was used to find out if the algorithm does what it has to do. The second test problem was to find out if the algorithm refines the expected areas.

This project was done as a bachelor thesis for the applied physics bachelor of the University of Technology Delft. The goal of this project was to investigate whether and how the adaptive mesh refinement algorithm can be used to solve the neutron transport equation.

A quick overview of the report is provided here. First the mathematics and theory of neutron transport and the finite element method will be explained. In the next chapter the AMR is discussed and the algorithm is also described. The fourth chapter describes the first test problem and discusses the results. In the fifth chapter the second test problem is presented and its results are discussed. The sixth chapter gives a short introduction to the mathematics and theory for the two-way model. This is a more realistic problem because also neutron flux to the left and scatter are incorporated. The last chapter discusses the conclusions, whether and how AMR can be used to solve the neutron transport equation. It also contains information about further research that can be done on the AMR algorithm.

Chapter 2

Mathematics and theory of neutron transport and the finite element method

This chapter describes the neutron transport equation and the finite element method.

2.1 Neutron transport

In this section the neutron transport equation will be explained. It is a complex equation and to be able to solve it a simplification is made. The neutron transport equation is an exact differential equation in which the various mechanisms by which neutrons can be gained or lost from an arbitrary volume V within the system is balanced. In essence, the neutron transport equation claims that neutrons gained equals neutrons lost. It is formulated as follows [1]

$$\underbrace{\frac{1}{v}\frac{\partial\psi(\mathbf{r},E,\hat{\mathbf{\Omega}},t)}{\partial t}}_{1}+\underbrace{\hat{\mathbf{\Omega}}\cdot\nabla\psi(\mathbf{r},E,\hat{\mathbf{\Omega}},t)}_{2}+\underbrace{\Sigma_t(\mathbf{r},E)\psi(\mathbf{r},E,\hat{\mathbf{\Omega}},t)}_{3}$$
$$=\int_{4\pi}d\hat{\mathbf{\Omega}}'\int_0^\infty\underbrace{dE'\Sigma_s(E'\rightarrow E,\hat{\mathbf{\Omega}}'\rightarrow\hat{\mathbf{\Omega}})\psi(\mathbf{r},E',\hat{\mathbf{\Omega}}',t)}_{4}+\underbrace{s(\mathbf{r},E,\hat{\mathbf{\Omega}},t)}_{5} \quad (2.1)$$

In which

- E, the specific energy,
- $\mathbf{r}$, the position vector,
- $\hat{\mathbf{\Omega}}$, the unit vector in the direction of motion,
- $v(E)$, the neutron velocity,
- t, the time,
- $\psi(\mathbf{r},E,\hat{\mathbf{\Omega}},t)$, the angular neutron flux, integration over all angles gives the neutron scalar flux,
- $\Sigma_t(\mathbf{r},E)$, the macroscopic total cross section, this includes all possible interactions,

- $\Sigma_s(E' \to E, \hat{\mathbf{\Omega}}' \to \hat{\mathbf{\Omega}})$, the macroscopic scatter cross section, this characterises the rate at which neutrons scatter from $(E', \hat{\mathbf{\Omega}}')$ to $(E, \hat{\mathbf{\Omega}})$,
- $s(\mathbf{r}, E, \hat{\mathbf{\Omega}}, t)$, the source term.

In this equation the first term is the time rate of change of neutrons in the system. The second term describes the leakage of neutrons into or out of the control volume. The third term accounts for all the neutrons that have a collision in that control volume. The fourth term is the term that describes in-scattering. These are the neutrons that enter this area of the control volume as a result of scattering interactions in another volume. The fifth and last term is a source.

Some simplifications can be made to the previous equation. Assuming a steady-state situation the first term in equation 2.1 equals zero. Also by looking at a single energy group the integral over the energy can be removed. This simplification is formulated as follows

$$\hat{\mathbf{\Omega}} \cdot \nabla\psi(\mathbf{r}, \hat{\mathbf{\Omega}}) + \sigma_t\psi(\mathbf{r}, \hat{\mathbf{\Omega}}) = \int \sigma_s(\hat{\mathbf{\Omega}} \cdot \hat{\mathbf{\Omega}}')\psi(\mathbf{r}, \hat{\mathbf{\Omega}}')d\hat{\mathbf{\Omega}}' + s(\mathbf{r}, \hat{\mathbf{\Omega}}) \tag{2.2}$$

Equation 2.2 can be reduced to the following equation

$$\mathcal{L}\psi = S \tag{2.3}$$

in which

$$\mathcal{L} = \hat{\mathbf{\Omega}} \cdot \nabla + \sigma_t - \int \sigma_s(\hat{\mathbf{\Omega}} \cdot \hat{\mathbf{\Omega}}')d\hat{\mathbf{\Omega}}' \tag{2.4}$$

The problem of interest in this bachelor thesis is the one dimensional case where only neutron transport to the right is taken into account, $\hat{\mathbf{\Omega}}$ reduces to 1 and ∇ reduces to $\mathrm{d}/\mathrm{d}x$. The scattering interactions of the neutrons are neglected (i.e. $\int \sigma_s(\hat{\mathbf{\Omega}} \cdot \hat{\mathbf{\Omega}}')d\hat{\mathbf{\Omega}}' = 0$), so $\mathcal{L}$ can be written for this problem as follows

$$\mathcal{L} = \frac{\mathrm{d}}{\mathrm{d}x} + \sigma_t \tag{2.5}$$

This differential equation has to be solved, which will be explained in the next sections.

2.2 Finite element method

The finite element method (FEM) is a numerical technique which can be used to find an approximate solution to a partial differential equation by rewriting it into a minimisation problem. The region of interest is divided into a number of discrete elements. These elements form the grid on which the solution is approximated. In FEM the solution is based on a linear combination of predefined functions, the basis functions. These basis functions can be of any form, for example linear or quadratic. In FEM it is possible to use non-uniform grids. This makes FEM the perfect choice for adaptive mesh refinement (AMR), where the grid will be non-uniform. Also FEM only uses information on the elements it is considering at the time. This makes it a particularly useful tool for computer implementation. Boundary conditions can be easily handled in FEM. They are just implemented in the matrix equation. They are therefore simpler to implement than in more classical finite difference methods.

In this project a one dimensional medium of length L is considered. This length L is divided into N elements of size h. These elements are connected by so-called nodes on which the solution will be determined. The minimisation problem here is the least squares functional. The minimisation of this functional leads to a matrix equation in which the solution of this equation gives an approximation to the differential equation. AMR is used later on. The algorithm starts with an uniform grid, which will be refined based on the local least squares functional value. A large functional value should indicate a large deviation from the exact solution, so here the grid will be refined. This algorithm will be explained in more detail in chapter 3. The first step in FEM is choosing the basis functions, which will be shown below.

2.2.1 Tent functions

The basis functions which have to be chosen before the problem can be solved are linear equations in this case. The solution will be an exponential curve, so a piecewise linear approximation is created using tent functions.

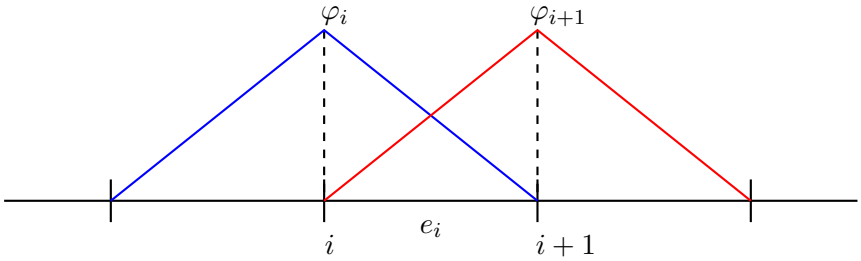


Figure 2.1: *Graphical display of the tent functions which are used as basis functions for the FEM method. A linear combination of these tent functions will provide the solution.*

They are described on element i with the formulas below and visualised in figure 2.1.

$$\varphi_i = 1 - \frac{1}{h}(x - x_i) \tag{2.6}$$

$$\varphi_{i+1} = \frac{1}{h}(x - x_i) \tag{2.7}$$

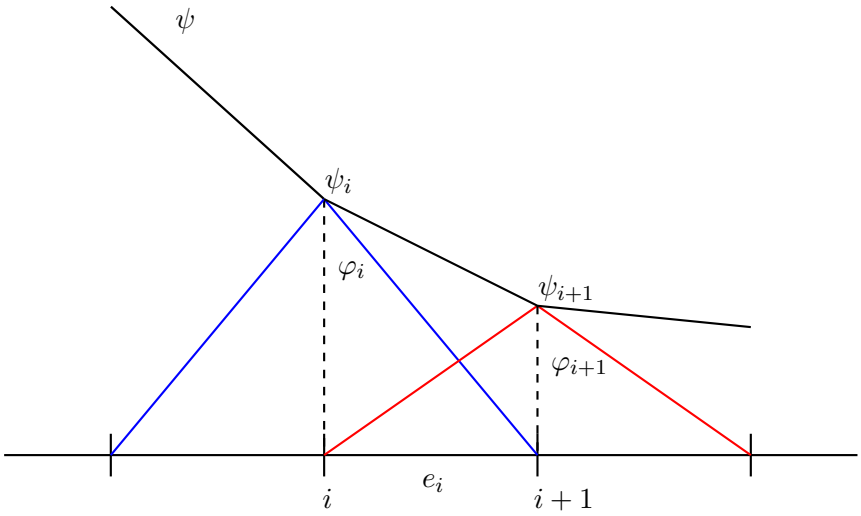


Figure 2.2: *Graphical display of the tent functions that build ψ. When both tents are added (per element) the solution (plotted in black) is found.*

On each element two halved tents are defined which contribute to the total solution. The right part of φ_i and the left part of φ_{i+1} are the contributing parts on element i, from here shortened as φ_i and φ_{i+1}. When multiplied by the solution on the nodes the summation of both tent functions results in the linear approximation on that element. It can be seen in figure 2.2 that only φ_i and φ_{i+1} are contributing to the solution on element i. The numerical solution on element i can hence be written as

$$\psi_{e_i}(x) = \psi_i \varphi_i(x) + \psi_{i+1}\varphi_{i+1}(x) \tag{2.8}$$

2.2.2 Numerical solution of the least squares functional

The functional that has to be minimised is the least squares functional. The squares of the errors made in solving every single equation are added to get the functional. The optimal solution is the one when the sum of squared residuals is a minimum. The least squares functional is given by

$$\mathcal{F}[\psi] = \int_0^L (\mathcal{L}\psi - S)^2 \, \mathrm{d}x \tag{2.9}$$

The exact solution to the differential equation makes the functional equal to zero. The numerical solution must be chosen as such that the functional is as close to zero as possible. This can be done by approximating the solution with a linear combination of basis functions φ_j, in which j is the number of nodes, and then minimizing the functional.

$$\psi(x) = \sum_j \psi_j \varphi_j(x) \tag{2.10}$$

By substituting this sum the only unknowns in the integrand are the coefficients ψ_j. The value of ψ_j is the value of the solution on node j. Hence to minimise the functional the derivative with respect to each individual ψ_k must be equal to zero. The index k represents the number of unknowns.

$$\frac{\partial \mathcal{F}[\psi]}{\partial \psi_k} = 0 \tag{2.11}$$

or

$$\frac{\partial}{\partial \psi_k} \int_0^L (\mathcal{L}\psi - S)^2 \, \mathrm{d}x = 0 \tag{2.12}$$

The integrand of equation 2.9 can be written as follows with the help of equations 2.4 and 2.10

$$\begin{aligned}
(\mathcal{L}\psi - S)^2 &= \left(\frac{\mathrm{d}\psi}{\mathrm{d}x} + \sigma\psi - S \right)^2 \\
&= \left(\frac{\mathrm{d}\psi}{\mathrm{d}x} \right)^2 + \sigma^2\psi^2 + 2\sigma\psi\frac{\mathrm{d}\psi}{\mathrm{d}x} - 2S\frac{\mathrm{d}\psi}{\mathrm{d}x} - 2\sigma S\psi + S^2 \\
&= \underbrace{\left(\sum_j \psi_j \frac{\mathrm{d}\varphi_j}{\mathrm{d}x} \right)^2}_{A} + \underbrace{\sigma^2 \left(\sum_j \psi_j \varphi_j \right)^2}_{B} + \underbrace{2\sigma \left(\sum_j \psi_j \varphi_j \right) \left(\sum_j \psi_j \frac{\mathrm{d}\varphi_j}{\mathrm{d}x} \right)}_{C} \\
&\quad \underbrace{- 2S \left(\sum_j \psi_j \frac{\mathrm{d}\varphi_j}{\mathrm{d}x} \right)}_{D} \underbrace{- 2\sigma S \left(\sum_j \psi_j \varphi_j \right)}_{E} + \underbrace{S^2}_{F}
\end{aligned} \tag{2.13}$$

The order of integration and differentiation can be reversed. This means the derivative of equation 2.13 with respect to ψ_k is needed, in which k is the node number. The derivative of part A is

$$\frac{\partial}{\partial \psi_k} \left(\sum_j \psi_j \frac{\mathrm{d}\varphi_j}{\mathrm{d}x} \right)^2 = 2 \left(\sum_j \psi_j \frac{\mathrm{d}\varphi_j}{\mathrm{d}x} \right) \frac{\mathrm{d}\varphi_k}{\mathrm{d}x} \tag{2.14}$$

The derivative of part B is

$$\frac{\partial}{\partial \psi_k} \sigma^2 \left(\sum_j \psi_j \varphi_j \right)^2 = 2\sigma^2 \left(\sum_j \psi_j \varphi_j \right) \varphi_k \tag{2.15}$$

The derivative of part C is

$$\frac{\partial}{\partial \psi_k} 2\sigma \left(\sum_j \psi_j \varphi_j \right) \left(\sum_j \psi_j \frac{\mathrm{d}\varphi_j}{\mathrm{d}x} \right) = 2\sigma \left(\sum_j \psi_j \frac{\mathrm{d}\varphi_j}{\mathrm{d}x} \right) \varphi_k + 2\sigma \left(\sum_j \psi_j \varphi_j \right) \frac{\mathrm{d}\varphi_k}{\mathrm{d}x} \tag{2.16}$$

The derivative of part D is

$$\frac{\partial}{\partial \psi_k} 2S \left(\sum_j \psi_j \frac{\mathrm{d}\varphi_j}{\mathrm{d}x} \right) = 2S \frac{\mathrm{d}\varphi_k}{\mathrm{d}x} \tag{2.17}$$

The derivative of part E is

$$\frac{\partial}{\partial \psi_k} 2\sigma S \left(\sum_j \psi_j \varphi_j \right) = 2\sigma S \varphi_k \tag{2.18}$$

The derivative of part F is

$$\frac{\partial}{\partial \psi_k} S^2 = 0 \tag{2.19}$$

After substituting these derivatives and putting the sum in front of the integral, the result is

$$\begin{aligned} \sum_j \psi_j \int_0^L \left[\frac{\mathrm{d}\varphi_k}{\mathrm{d}x} \frac{\mathrm{d}\varphi_j}{\mathrm{d}x} + \sigma^2 \varphi_k \varphi_j + \sigma \varphi_k \frac{\mathrm{d}\varphi_j}{\mathrm{d}x} + \sigma \varphi_j \frac{\mathrm{d}\varphi_k}{\mathrm{d}x} \right] \mathrm{d}x \\ = \int_0^L \left[S \left(\frac{\mathrm{d}\varphi_k}{\mathrm{d}x} + \sigma \varphi_k \right) \right] \mathrm{d}x, \qquad k = 1, \ldots, N+1 \end{aligned} \tag{2.20}$$

This can also be written shortly as

$$\sum_j \psi_j \int_0^L \left[\left(\mathcal{L}\varphi_k \right) \left(\mathcal{L}\varphi_j \right) \right] \mathrm{d}x = \int_0^L \left[S \left(\mathcal{L}\varphi_k \right) \right] \mathrm{d}x \qquad k = 1, \ldots, N+1 \tag{2.21}$$

Now consider an element where the solution is influenced by two unknowns: ψ_i and ψ_{i+1}. For ψ_i equation 2.21 reduces to

$$\psi_i \int_{e_i} \left[\mathcal{L}\varphi_i \right]^2 \mathrm{d}x + \psi_{i+1} \int_{e_i} \left[\left(\mathcal{L}\varphi_i \right) \left(\mathcal{L}\varphi_{i+1} \right) \right] \mathrm{d}x = \int_{e_i} \left[S \left(\mathcal{L}\varphi_i \right) \right] \mathrm{d}x \tag{2.22}$$

For ψ_{i+1} equation 2.21 reduces to

$$\psi_i \int_{e_i} \left[\left(\mathcal{L}\varphi_{i+1} \right) \left(\mathcal{L}\varphi_i \right) \right] \mathrm{d}x + \psi_{i+1} \int_{e_i} \left[\mathcal{L}\varphi_{i+1} \right]^2 \mathrm{d}x + = \int_{e_i} \left[S \left(\mathcal{L}\varphi_{i+1} \right) \right] \mathrm{d}x \tag{2.23}$$

Together they form an element matrix equation for element e_i which is shown below.

2.2.3 Element matrices and vectors

The solution to equation 2.3 can be obtained using matrices. In order to construct these matrices it is necessary to evaluate the integrals in equation 2.20. Since the tent functions are defined per element, the natural way to do this is element by element. Per element at most four integrals are contributing and thus different from zero. These four integrals are stored in a small matrix, the element matrix. For element i the equations for node $k = i$ and node $k = i + 1$ are needed to construct the element matrix. The result is:

$$A_{e_i} = \begin{bmatrix} a_{i,i} & a_{i,i+1} \\ a_{i+1,i} & a_{i+1,i+1} \end{bmatrix} \tag{2.24}$$

In which

$$a_{i,i} = \int_{e_i} \left(\left(\frac{\mathrm{d}\varphi_i}{\mathrm{d}x} \right)^2 + \sigma^2 \varphi_i^2 + 2\sigma \varphi_i \frac{\mathrm{d}\varphi_i}{\mathrm{d}x} \right) \mathrm{d}x \tag{2.25}$$

$$a_{i,i+1} = \int_{e_i} \left(\frac{\mathrm{d}\varphi_i}{\mathrm{d}x} \frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x} + \sigma^2 \varphi_i \varphi_{i+1} + \sigma \varphi_i \frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x} + \sigma \frac{\mathrm{d}\varphi_i}{\mathrm{d}x} \varphi_{i+1} \right) \mathrm{d}x \tag{2.26}$$

$$a_{i+1,i} = \int_{e_i} \left(\frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x} \frac{\mathrm{d}\varphi_i}{\mathrm{d}x} + \sigma^2 \varphi_{i+1} \varphi_i + \sigma \varphi_{i+1} \frac{\mathrm{d}\varphi_i}{\mathrm{d}x} + \sigma \frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x} \varphi_i \right) \mathrm{d}x \tag{2.27}$$

$$a_{i+1,i+1} = \int_{e_i} \left(\left(\frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x} \right)^2 + \sigma^2 \varphi_{i+1}^2 + 2\sigma \varphi_{i+1} \frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x} \right) \mathrm{d}x \tag{2.28}$$

In the same way the element vector is created using Newton Cotes integration. Equation 2.20 is discretised, but S is still a continuous function. Newton Cotes integration can be used to discretise the source. The source function is approximated by a linear combination of the same basis functions. In this equation the $S(x_j)$ is the value of the source on that node.

$$S(x) = \sum_j S(x_j)\varphi_j(x) \tag{2.29}$$

Because the source is now also written as a linear combination of basis functions, the right hand side $\mathbf{b}$ can be found.

$$\mathbf{b}_{e_i} = \begin{bmatrix} b_i \\ b_{i+1} \end{bmatrix} \tag{2.30}$$

In which

$$\begin{aligned} b_i = \int_{e_i} \Bigg[&\left(\frac{\mathrm{d}\varphi_i}{\mathrm{d}x}(x_i) + \sigma\varphi_i(x_i) \right) S(x_i)\varphi_i \\ &+ \left(\frac{\mathrm{d}\varphi_i}{\mathrm{d}x}(x_{i+1}) + \sigma\varphi_i(x_{i+1}) \right) S(x_{i+1})\varphi_{i+1} \Bigg] \, \mathrm{d}x \end{aligned} \tag{2.31}$$

$$\begin{aligned} b_{i+1} = \int_{e_i} \Bigg[&\left(\frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x}(x_i) + \sigma\varphi_{i+1}(x_i) \right) S(x_i)\varphi_i \\ &+ \left(\frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x}(x_{i+1}) + \sigma\varphi_{i+1}(x_{i+1}) \right) S(x_{i+1})\varphi_{i+1} \Bigg] \, \mathrm{d}x \end{aligned} \tag{2.32}$$

Using the definitions of the φ some simplifications can be made, which results in an easier right hand side:

$$\mathbf{b}_{e_i} = \begin{bmatrix} \int_{e_i} \left[\left(-\frac{1}{h} + \sigma \right) S(x_i)\varphi_i - \frac{1}{h} S(x_{i+1})\varphi_{i+1} \right] \mathrm{d}x \\ \int_{e_i} \left[\frac{1}{h} S(x_i)\varphi_i + \left(\frac{1}{h} + \sigma \right) S(x_{i+1})\varphi_{i+1} \right] \mathrm{d}x \end{bmatrix} \tag{2.33}$$

To fill the matrix A and vector $\mathbf{b}$ four 2x2 matrices are created. These matrices contain the different integrals which have to be computed, shown in equations 2.25 until 2.28. The matrices containing the integrals are listed below

$$P_1 = \begin{bmatrix} \int_{e_i} \frac{\mathrm{d}\varphi_i}{\mathrm{d}x}\frac{\mathrm{d}\varphi_i}{\mathrm{d}x} \, \mathrm{d}x & \int_{e_i} \frac{\mathrm{d}\varphi_i}{\mathrm{d}x}\frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x} \, \mathrm{d}x \\ \int_{e_i} \frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x}\frac{\mathrm{d}\varphi_i}{\mathrm{d}x} \, \mathrm{d}x & \int_{e_i} \frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x}\frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x} \, \mathrm{d}x \end{bmatrix} = \begin{bmatrix} \frac{1}{h} & -\frac{1}{h} \\ -\frac{1}{h} & \frac{1}{h} \end{bmatrix} \tag{2.34}$$

$$P_2 = \begin{bmatrix} \int_{e_i} \frac{\mathrm{d}\varphi_i}{\mathrm{d}x}\varphi_i \, \mathrm{d}x & \int_{e_i} \frac{\mathrm{d}\varphi_i}{\mathrm{d}x}\varphi_{i+1} \, \mathrm{d}x \\ \int_{e_i} \frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x}\varphi_i \, \mathrm{d}x & \int_{e_i} \frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x}\varphi_{i+1} \, \mathrm{d}x \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.35}$$

$$P_3 = \begin{bmatrix} \int_{e_i} \varphi_i \frac{\mathrm{d}\varphi_i}{\mathrm{d}x} \, \mathrm{d}x & \int_{e_i} \varphi_i \frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x} \, \mathrm{d}x \\ \int_{e_i} \varphi_{i+1} \frac{\mathrm{d}\varphi_i}{\mathrm{d}x} \, \mathrm{d}x & \int_{e_i} \varphi_{i+1} \frac{\mathrm{d}\varphi_{i+1}}{\mathrm{d}x} \, \mathrm{d}x \end{bmatrix} = P_2^T = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.36}$$

$$P_4 = \begin{bmatrix} \int_{e_i} \varphi_i^2 \, \mathrm{d}x & \int_{e_i} \varphi_i\varphi_{i+1} \, \mathrm{d}x \\ \int_{e_i} \varphi_{i+1}\varphi_i \, \mathrm{d}x & \int_{e_i} \varphi_{i+1}^2 \, \mathrm{d}x \end{bmatrix} = \begin{bmatrix} \frac{h}{3} & \frac{h}{6} \\ \frac{h}{6} & \frac{h}{3} \end{bmatrix} \tag{2.37}$$

In the algorithm it is now easy to pick the needed integral, because they are all stored.

2.2.4 Matrix coefficients

Once all element matrices and vectors are computed, they can be added to create the large matrix A and the right hand side $\mathbf{b}$ [2]. To explain how the large matrix A is constructed a simple example is shown. Suppose the problem has three elements and therefore four nodes with four unknowns which results in a 4x4 matrix. The element matrix for an arbitrary element is given by

$$A_{e_i} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{2.38}$$

And suppose that the element vector is of the shape

$$\mathbf{b}_{e_i} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{2.39}$$

First element matrix 1 and element vector 1 are assembled

$$A^1 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{b}^1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.40}$$

Next element matrix 2 and element vector 2 are added

$$A^2 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{b}^2 = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 0 \end{bmatrix} \tag{2.41}$$

Finally this process is repeated for element matrix 3 and element vector 3

$$A = A^3 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}, \quad \mathbf{b} = \mathbf{b}^3 = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 1 \end{bmatrix} \tag{2.42}$$

It can be seen that this results in a tridiagonal matrix A. The equation which has to be solved is of the form:

$$A\boldsymbol{\Psi} = \mathbf{b} \tag{2.43}$$

This can easily be done using software like Matlab.

2.2.5 Boundary condition

Before the solution can be calculated, a boundary condition is needed. It is implemented after the large matrix A is created. The boundary condition is

$$\psi_1 = \psi_{left} \tag{2.44}$$

In which ψ_{left} is the number of neutrons coming in at the left side of the volume per unit of time per squared meter. Implementing the boundary condition is done by setting the first value of the first row of matrix A to one and the rest of that row to zero. The first value of the right hand side $\mathbf{b}$ is set to ψ_{left}. Now the linear system of equations is complete and it can be solved.

Chapter 3

Adaptive mesh refinement

In this chapter the adaptive mesh refinement method will be explained. The error estimate is shown and an overview of the used algorithm is given. The basic components of the AMR algorithm are error estimation, regridding and interpolation. This will all be explained in this chapter.

3.1 Error estimate

The purpose of error estimation is to identify and tag elements where additional refinement is needed. The adaptive mesh refinement approach adds new refinement grids in regions where the error is estimated to be large. In this case it splits elements in two new elements with half the size of the original element. Once a new set of elements is generated, the source values and material properties are redefined for the new grid. Then a new matrix equation is generated and solved. After the calculation of the solution the contribution to the least squares functional (equation 2.9) is computed. It is repeated here

$$\begin{aligned} \mathcal{F}[\psi] &= \int_0^L (\mathcal{L}\psi - S)^2 \, \mathrm{d}x \\ &= \sum_{i=1}^{N} \int_{e_i} (\mathcal{L}\psi - S)^2 \, \mathrm{d}x \end{aligned} \tag{3.1}$$

It is also possible to compute the exact solution for this problem, which will be shown in the next chapter. The real error can be calculated by subtracting the exact solution from the numerical solution (equation 2.8). It is also squared and integrated because it has to be a good representation of the real error and it has to be comparable with the functional. This real error is called ϵ.

$$\begin{aligned} \epsilon &= \int_0^L (\psi_{\text{exact}} - \psi_{\text{numerical}})^2 \, \mathrm{d}x \\ &= \sum_{i=1}^{N} \int_{e_i} (\psi_{\text{exact}} - \psi_{\text{numerical}})^2 \, \mathrm{d}x \end{aligned} \tag{3.2}$$

3.2 Overview of the algorithm

The subdivision of the algorithm and the implementation of the adaptive mesh refinement are explained in this section.

3.2.1 Computation of the solution

The calculation of the solution is an automated process in the algorithm. The algorithm can be divided into three steps, this can be seen in figure 3.1.

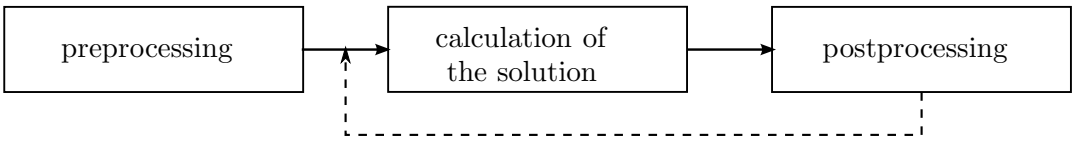


Figure 3.1: *Overview of the structure of the algorithm*

The part in which the preprocessing is done can be split into the following steps:

- define the length of the slab
- define the number of elements
- define the material properties
- define ψ_{left}
- set the needed accuracy
  - set the percentage of elements which have to be halved
  - set the number of times the algorithm will run

The most important part is the definition of the problem.

The part in which the solving is done can be split into the following steps:

- read input parameters and grid
- then for all elements
  - compute element matrix and element vector
  - add element matrix to large matrix
  - add element vector to large vector
- apply boundary condition
- solve system of equations
- save results for postprocessing

The most important step is the computation of the large matrix and vector.

The part in which the postprocessing is done can be split into the following steps:

- for all elements
  - calculate contribution to functional (error estimation)
  - tag elements with largest contribution to functional
- halve the tagged elements (regridding)
- interpolate source values and material properties from old grid to new grid
- if the number of times the loop has to run is not reached yet, start the solving process again

As said before, the steps typical for AMR are the error estimation, regridding and interpolation

3.2.2 Implementation of AMR

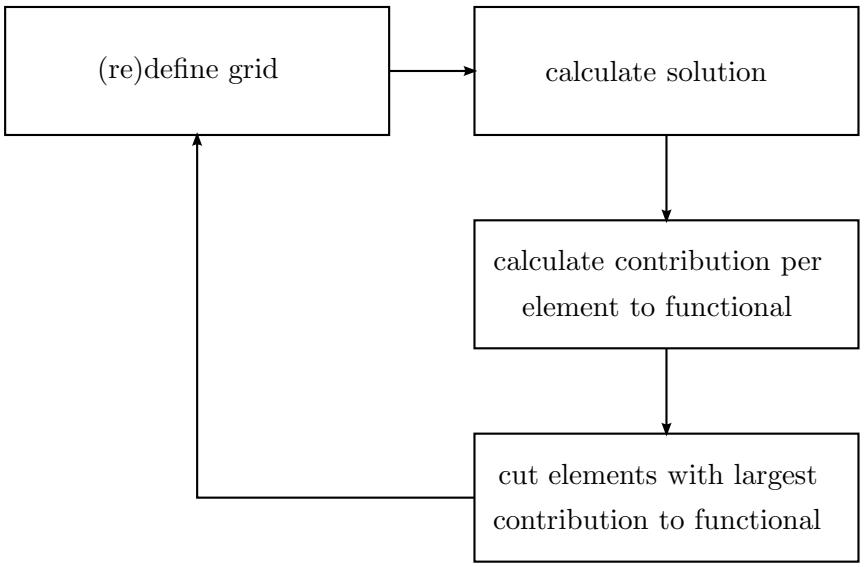


Figure 3.2: *Overview of the implementation of AMR in the algorithm used in this report*

After calculation of the solution, the contribution to the functional is determined per element. If the contribution of an element to the functional is large compared to other elements, the element is split into two elements with equal size. The algorithm, which can be seen in figure 3.2, sorts the contribution to the functional from largest to smallest. The elements with the largest contribution are split. Depending on the needed accuracy a percentage of the total number of elements is halved several times. This means that the size of the elements, h, is different for different elements. The result is that the matrices in equations 2.34 and 2.37 will change every time elements are halved.

3.3 Advantages and disadvantages of AMR

In this section an overview of the advantages and disadvantages of the adaptive mesh refinement is provided. First the advantages are discussed, then the disadvantages.

3.3.1 Advantages

Some problems have strong varying material properties and sources in some places. In those regions a coarse grid will not give a very good result. With AMR it is possible to refine the grid automatically in those regions where the solution is more interesting. The opposite is also true. In the regions where the solution is less interesting, the grid may stay coarse. With AMR it is possible to do those things at the same time.

A second advantage is the amount of memory and run time needed for the computation. Without AMR the grid is refined everywhere, also in places where it is not necessary. This takes up a lot of memory space. Also the matrix equation is larger, because there are more elements on the grid. The result is more computation time. AMR can improve on both the memory use and computation time. The matrix equation is smaller, hence less memory is needed. Also the smaller matrix equation is solved faster.

The third advantage has to do with the numerical accuracy. Coarse grids give a large error compared to fine grids. By refining the grid in regions where gradients are large, the numerical error can be reduces and hence the accuracy improved.

3.3.2 Disadvantages

A disadvantage of AMR is the more complex code. There is an algorithm needed to find the elements where the functional is largest and subsequently cuts the element in two new elements. This means the algorithm has to search for 'bad' elements, make two new elements, update the grid, update the source function and the material properties and then the process starts again. This kind of code is more difficult to program than a normal FEM method.

Another disadvantage is the communication between different levels in the algorithm. All the modules have to speak with each other and have to provide feedback for previous modules. This makes the structure of the code more complex.

A last disadvantage is the need to solve the matrix equation more than once. With a normal FEM method one chooses a grid, solves and the computation is ready. With AMR one starts with a grid which can be coarser than with normal FEM. The algorithm will then refine the grid in places where it is needed and solves again until the needed accuracy is obtained. It is of course the question whether the extra solving steps weigh up against the amount of elements in normal FEM.

Chapter 4

Test problem 1

In this chapter the fist test problem will be discussed. First the general solution to the differential equation will be given. This will be followed by the description of this test problem. Afterwards the results for the first test problem will be shown and discussed.

4.1 General solution to the differential equation

The differential equation which has to be solved is equation 2.3 and will be repeated here

$$\frac{\mathrm{d}\psi}{\mathrm{d}x} + \sigma\psi = S \tag{4.1}$$

This equation is of the form

$$\frac{\mathrm{d}\psi}{\mathrm{d}x} + p(x)\psi = q(x) \tag{4.2}$$

in which

$$\begin{aligned} p(x) &= \sigma \\ q(x) &= S \end{aligned} \tag{4.3}$$

This can be solved using an integrating factor μ

$$\mu = e^{\int^x p(x)\,\mathrm{d}x} \tag{4.4}$$

The solution becomes

$$\begin{aligned} \psi &= \frac{1}{\mu}\left(\int \mu q(x)\,\mathrm{d}x + c\right) \\ &= e^{-\int^x p(x')\,\mathrm{d}x'}\left(\int e^{\int^x p(x')\,\mathrm{d}x'} q(x)\,\mathrm{d}x + c\right) \end{aligned} \tag{4.5}$$

in which c is the arbitrary constant of integration, which can be found by applying the boundary condition.

4.2 Definition of the test problem

In this section the schematic overview of this test problem is shown. The exact solution is also given. This is necessary to check whether the algorithm gives the right solution.

4.2.1 Schematic overview

The first test problem is a relatively easy problem to solve. Its primary purpose is to check if the algorithm works as expected. The first problem to explore the AMR technique consists of a linear source, an area with a low σ, then an area with a high σ and finally again a region with a low σ. Figure 4.1 gives an overview of the situation.

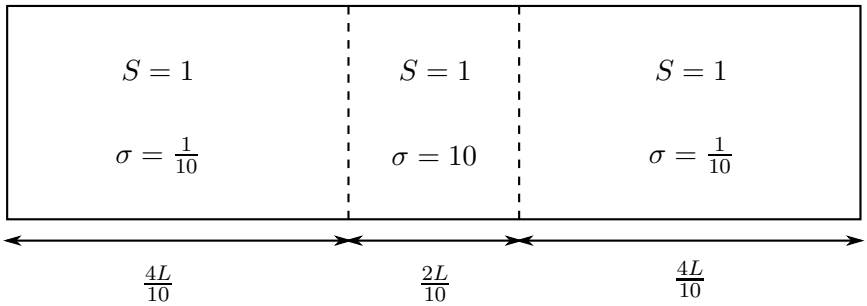


Figure 4.1: *Schematic overview of the test problem, a one dimensional medium. Neutrons are coming in from the left and travelling to the right. The number of neutrons coming in from the left side define the boundary condition.*

4.2.2 Exact solution

In equation 4.5 it is required that the functions $p(x)$ and $q(x)$ are continuous, which is not the case. Therefore the problem is split into three separate problems which can be solved using this integration factor. The length L of the slab is one. The complete solution for ψ is found when all three separate solutions are combined

$$\psi(x) = \begin{cases} 10 - 9e^{-x/10} & \text{if } 0 \leq x \leq \frac{4}{10} \\ \frac{1}{10} + \left(9\frac{9}{10} - 9e^{-\frac{1}{25}}\right) e^{-10(x-4/10)} & \text{if } \frac{4}{10} \leq x \leq \frac{6}{10} \\ 10 + \left(9\frac{9}{10}e^{-2} - 9e^{-\frac{1}{25}} - 9\frac{9}{10}\right) e^{-\frac{x-6/10}{10}} & \text{if } \frac{6}{10} \leq x \leq 1 \end{cases} \tag{4.6}$$

The solution can be plotted, which is shown in figure 4.2

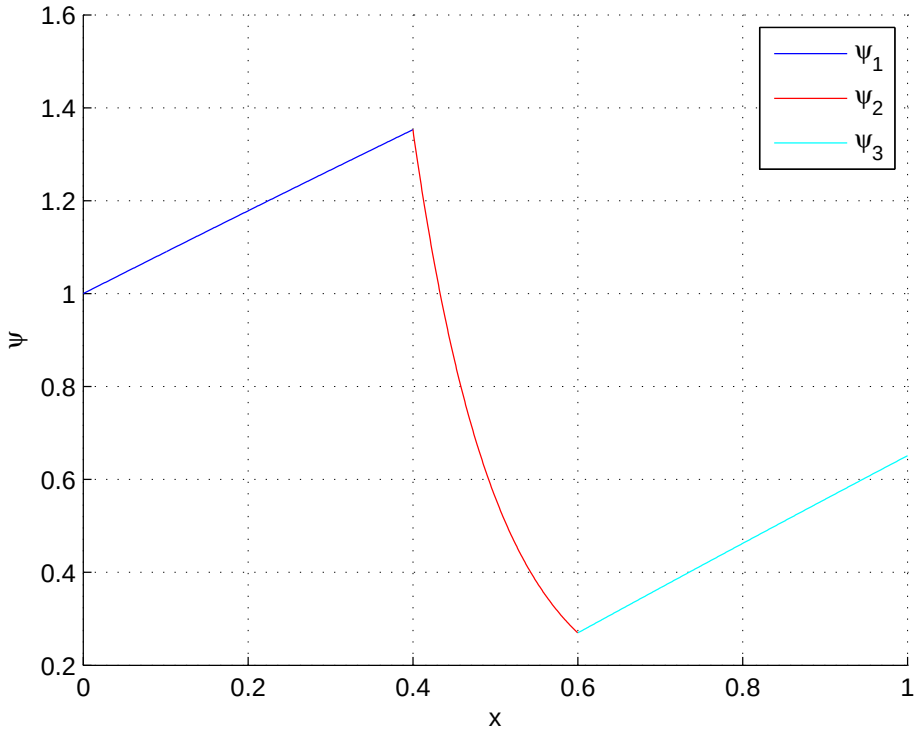


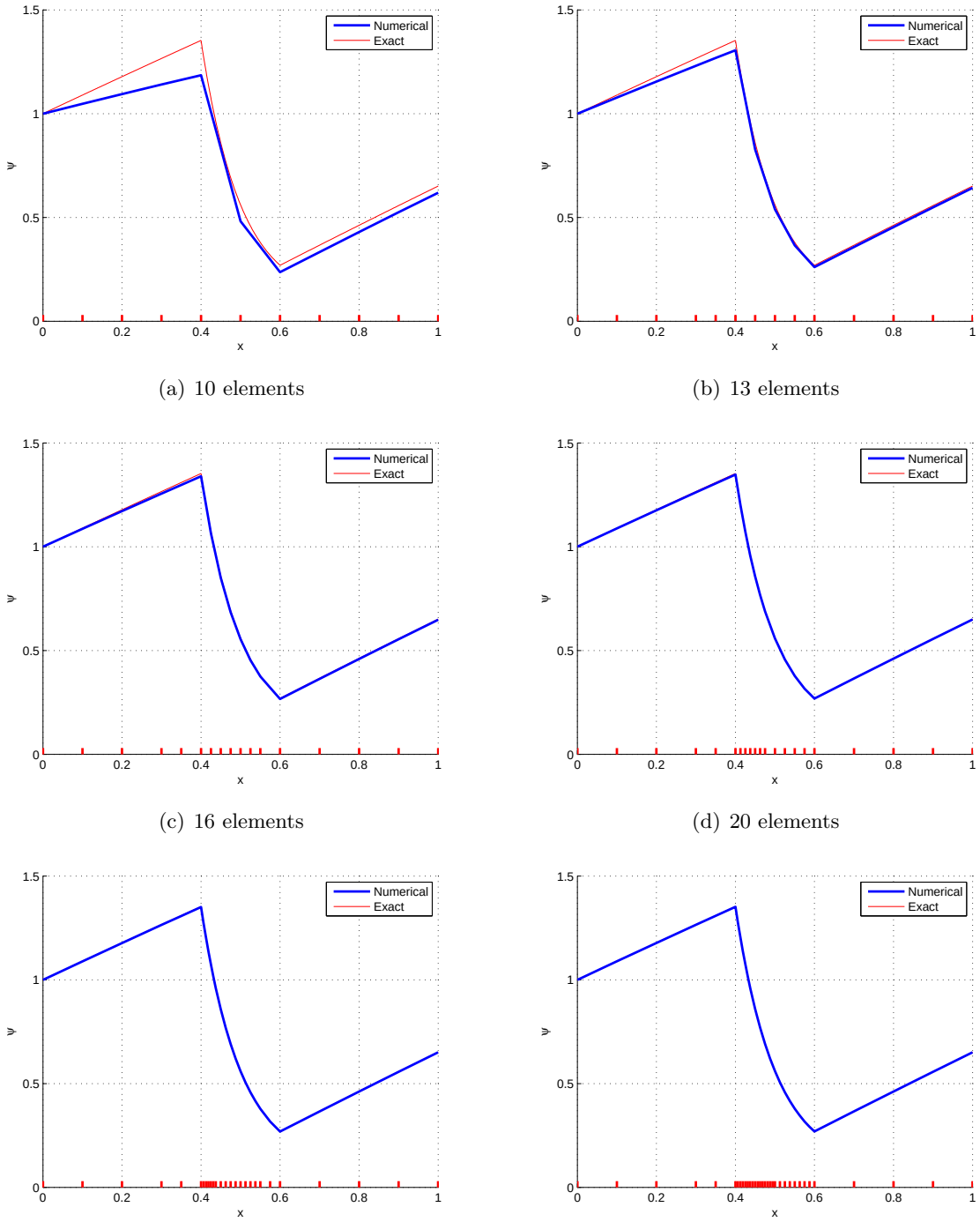
Figure 4.2: *Solution of the first test problem. This is the exact solution which is calculated above. It is used in the comparison of the exact and numerical solution of this test problem. The three separate solutions are plotted in three different colours.*

4.3 Results

In this section the results for the first test problem will be shown. First a comparison of the exact and numerical solution will be given. After that the real error and least squares functional will be compared to check if the least squares functional is a well defined error estimate. Finally the adaptive and uniform mesh refinement will be shown.

4.3.1 Comparison of exact and numerical solution

In figure 4.3 the numerical solution to the differential equation 2.3 is shown starting with ten elements. The algorithm has five cutting rounds. The vertical red lines on the x axis represent the element boundaries. This makes it easy to see where the algorithm has halved an element. The exact solution is also plotted. The first plot shows the numerical solution with ten elements. By adding more elements on specific locations determined by the algorithm the numerical solution converges nice and quick to the exact solution.
In figure 4.3 it is visible that the algorithm works quite well to get the solution, even with only ten elements. The mesh is refined in the area with a high σ because the changes are large there. This is what was expected.



(a) 10 elements

(b) 13 elements

(c) 16 elements

(d) 20 elements

(e) 26 elements

(f) 33 elements

Figure 4.3: *Comparison of the numerical and exact solution of the first test problem. The initial number of elements is ten and in the end 33 (after five times cutting 30% of the elements). The numerical solution is plotted in blue and the exact solution in red. On the x-axis the distribution of elements is shown with small red lines.*

In figure 4.4 the real error (the difference between the exact and numerical solution on the nodes) is shown for 33 elements adaptively spaced. The error at the left side is zero, as it should be due to the boundary condition. The error then grows to the maximum value, which is at the sharp transition in cross section σ. The error then becomes smaller in the region where the algorithm has refined the grid.
The difference between the exact and numerical solution for 33 elements shows that the solution is the least accurate in the area with the steep slope due to the large σ $\left(\frac{4}{10} \leq x \leq \frac{6}{10}\right)$. This is also the area where the algorithm has refined, but the real error is still large there. To the right of that area the real error is also larger than on the left. The functional is large in the area with the large σ and therefore there is no refinement in the other regions.

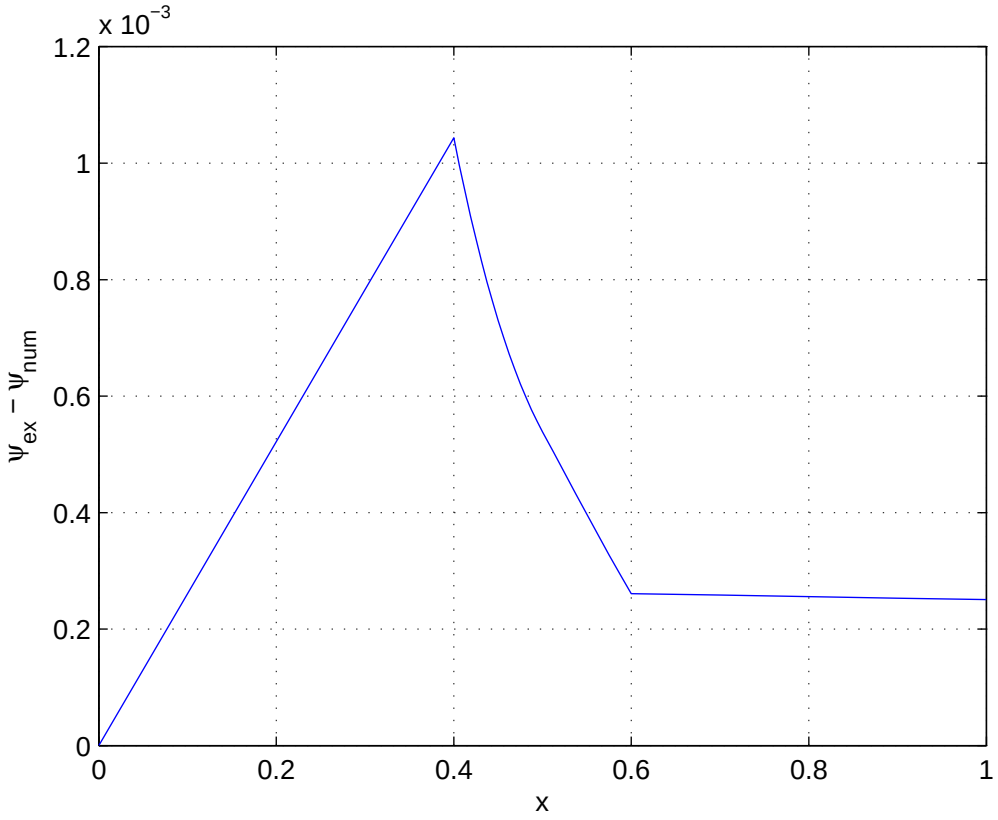
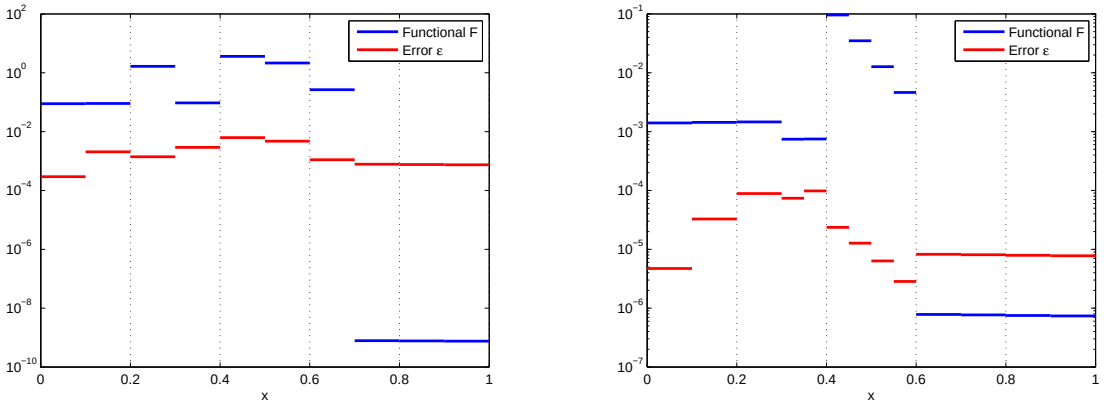


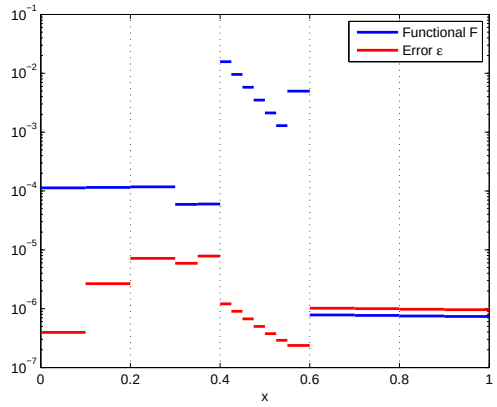
Figure 4.4: *Difference between the exact and numerical solution on the nodes for 33 elements on an adaptive grid.*

In figure 4.5 the functional F and error ϵ are plotted. The y-axis is logarithmic and note that the scale is not the same in each graph. The values of the functional and ϵ are not correlated at all. This seems to be the case for all grid refinement steps. Sometimes the error decreases while the functional increases. The functional increases in the area with the high σ. The area left to that needs refinement, but is not refined. The functional is therefore not a representative measure for the error per element. An explanation is that the definition of the functional and the error ϵ is quite different. The functional has nothing to do with the exact solution and vice versa. Hence these measures for error can hardly be compared.

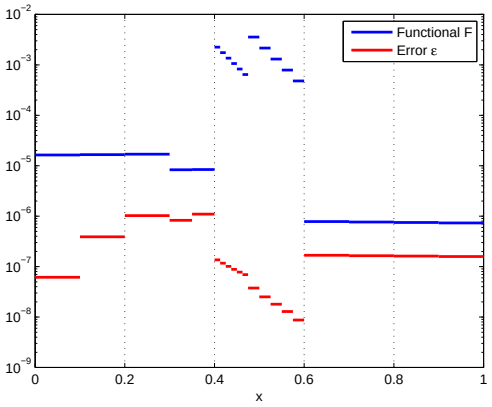


(a) 10 elements

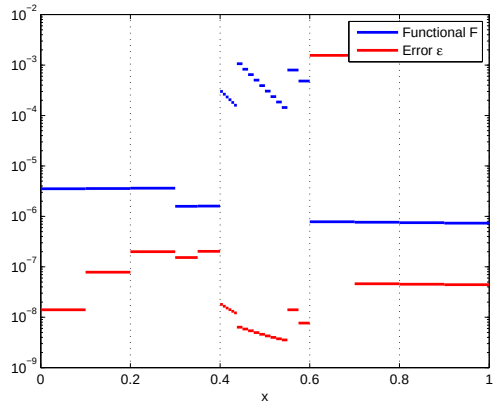
(b) 13 elements



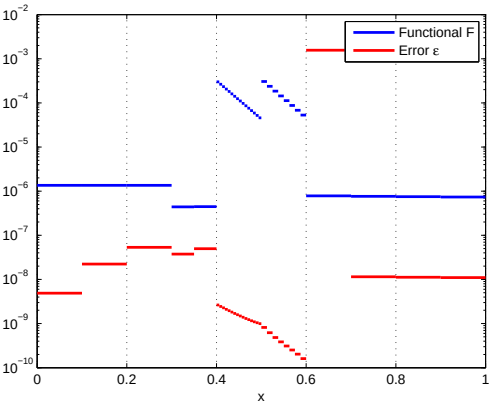
(c) 16 elements



(d) 20 elements



(e) 26 elements



(f) 33 elements

Figure 4.5: *Functional F and error ϵ of the first test problem are plotted. The initial number of elements is ten and in the end it is 33 (after five times cutting 30% of the elements). The functional is plotted in blue and the error ϵ in red.*

4.3.2 Comparison of adaptive and uniform grid

In figure 4.6 the total functional and error are plotted for an uniform grid and an adaptive grid. The y-axis has a logarithmic scale. Noticeable is that the functional of the adaptive grid decreases much faster, which shows that the use of an adaptive grid is useful.
For the same functional value a lot more elements are needed in the uniform grid, about 3 to 5 times as much. This means that the computation time is longer and therefore more expensive. The ϵ and functional do not show the same behaviour. For the uniform grid the ϵ alternates more or less around a trendline with the same shape as the functional for the uniform grid. For the adaptive grid the ϵ is very variable. It decreases and increases and finally grows larger than the functional for the adaptive grid. It can be concluded that the functional is not a good error estimate. But even for a uniform grid the ϵ alternates, which shows that the solution is not very accurate and therefore the solving method (with the least squares functional) is not the best one.

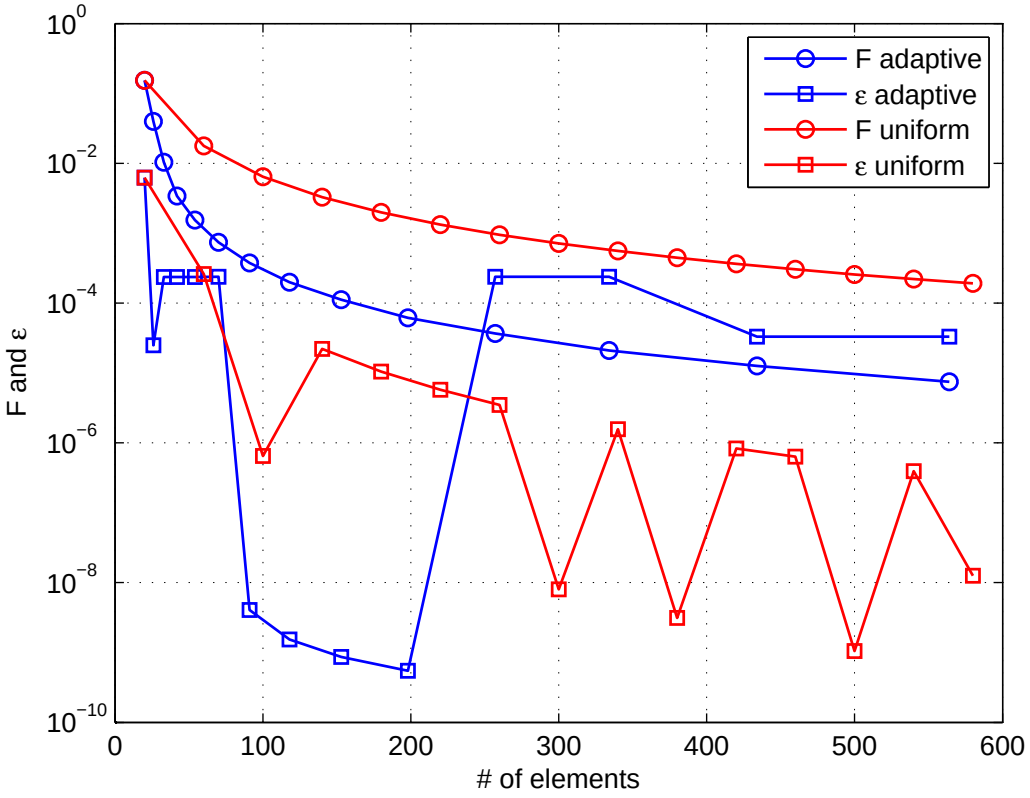


Figure 4.6: *Functional F and error ϵ for an adaptive and uniform grid. The functional is plotted with circles and the error ϵ with squares. The plot for the adaptive grid is shown in blue and the plot for the uniform grid in red. The y-axis has a logarithmic scale.*

Chapter 5

Test problem 2

The second test problem is a more difficult problem. It contains regions with major changes. The purpose of this is to see whether the algorithm refines in the expected areas.

5.1 Definition of the test problem

Analogue to the previous chapter, first the schematic overview of this test problem will be given followed by the exact solution.

5.1.1 Schematic overview

The second problem to explore the AMR technique consists of one region with a large source and other regions without source. The cross section is low in the region with the large source and high in a region without source. This creates a volume in which there are major changes in the absorption and production of neutrons. Figure 5.1 gives an overview of the situation. The length L of the slab is one.

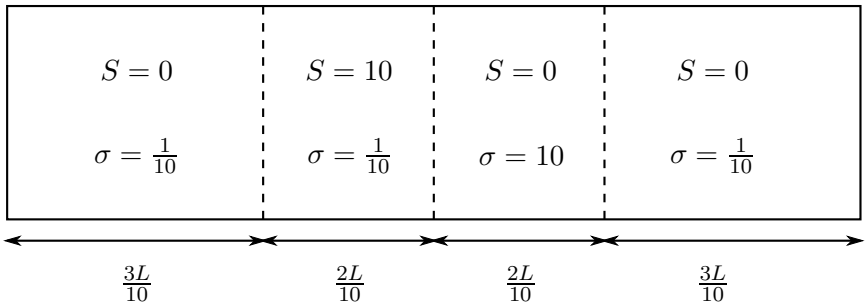


Figure 5.1: *Schematic overview of the test problem, a one dimensional medium. Neutrons are coming in from the left and travelling to the right. The number of neutrons coming in from the left side define the boundary condition.*

5.1.2 Exact solution

Again because of continuity this problem is split into four parts. The complete solution for ψ is found when all four separate solutions are combined

$$\psi(x) = \begin{cases} e^{-x/10} & \text{if } 0 \leq x \leq \frac{3}{10} \\ 100 + \left(e^{-\frac{3}{100}} - 100\right) e^{-\frac{x-3/10}{10}} & \text{if } \frac{3}{10} \leq x \leq \frac{5}{10} \\ \left(100 + e^{-\frac{1}{20}} - 100e^{-\frac{1}{50}}\right) e^{-10(x-5/10)} & \text{if } \frac{5}{10} \leq x \leq \frac{7}{10} \\ e^{-2}\left(100 + e^{-\frac{1}{20}} - 100e^{-\frac{1}{50}}\right) e^{-\frac{x-7/10}{10}} & \text{if } \frac{7}{10} \leq x \leq 1 \end{cases} \tag{5.1}$$

The solution can be plotted, which is shown in figure 5.2

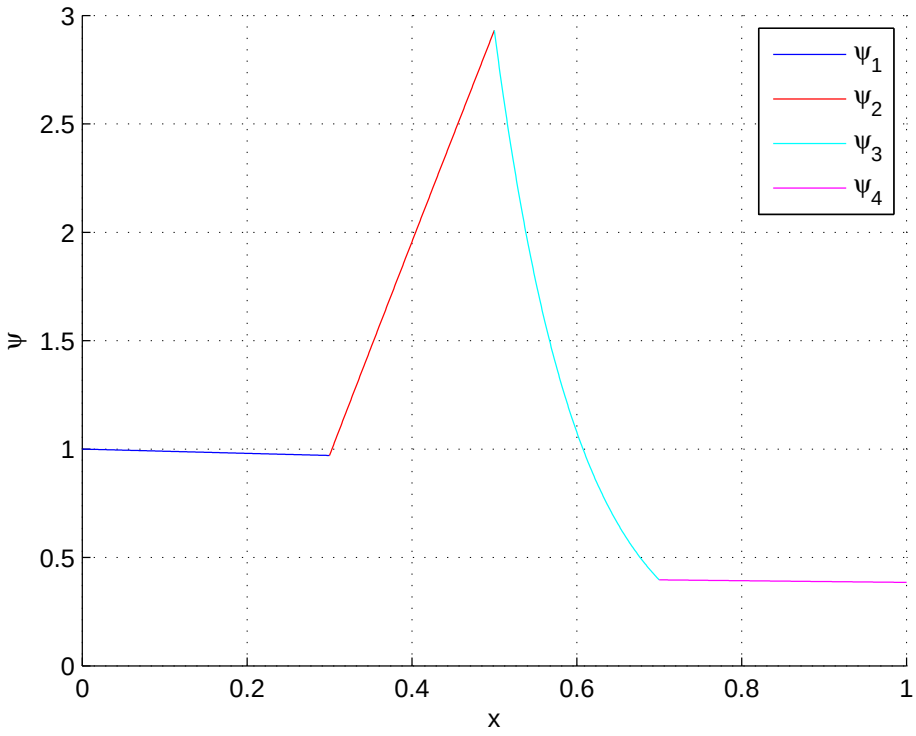


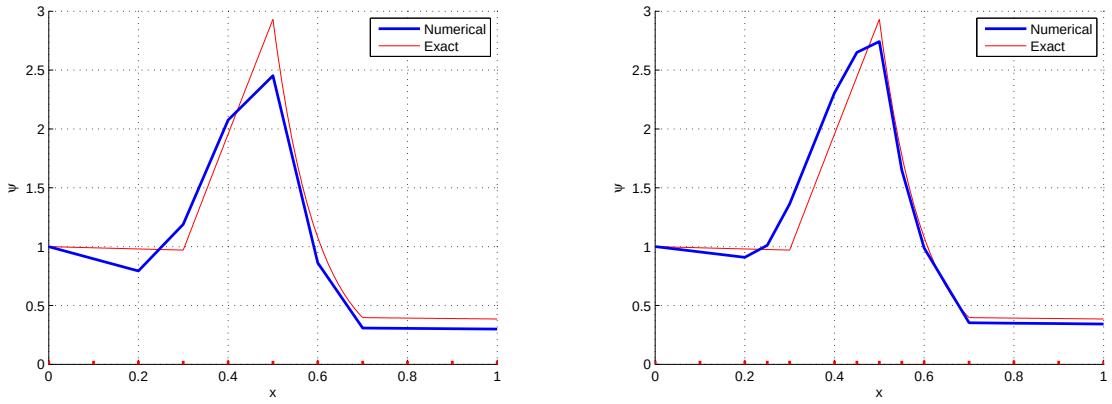
Figure 5.2: *Solution of the second test problem. This is the exact solution which is calculated above. It is used in the comparison of the exact and numerical solution of this test problem.*

5.2 Results

In this section the results for the results for this test problem will be shown. The results will be compared with the exact solution and a comparison of the adaptive and uniform mesh refinement is given.

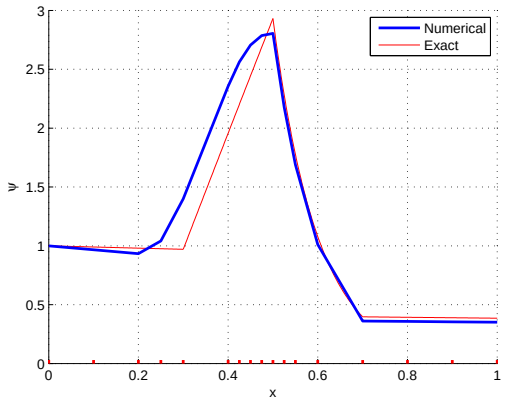
5.2.1 Comparison of exact and numerical solution

Figure 5.3 shows the solution of test problem two starting with ten elements and then refining five times. The first solution is, considering the small amount of elements, quite accurate, but unfortunately the solution in the region $\frac{2}{10} \leq x \leq \frac{5}{10}$ starts to deviate from the exact solution. It can also be seen that the algorithm didn't refine the grid in the middle of this region, which would partly explain the deviation. The algorithm doesn't work very well for this test problem. The least squares method used for solving is not scaled. It is influenced too much by the value of σ_t. The result is that the solution is not the best one.

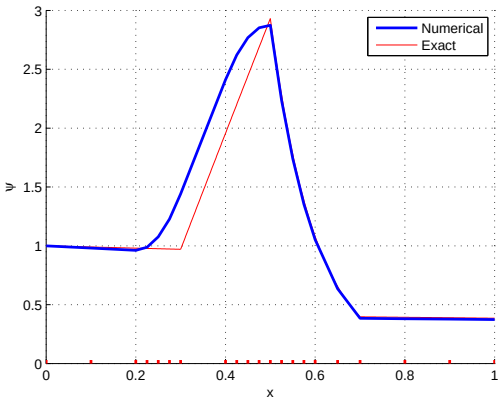


(a) 10 elements

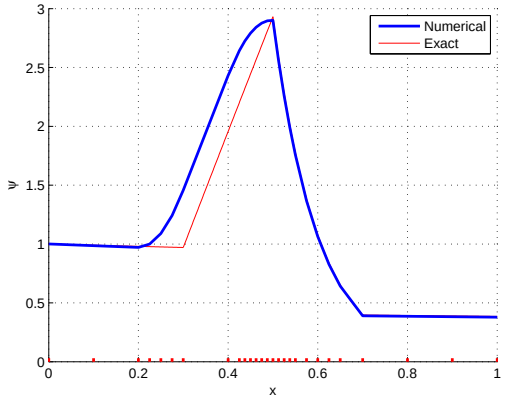
(b) 13 elements



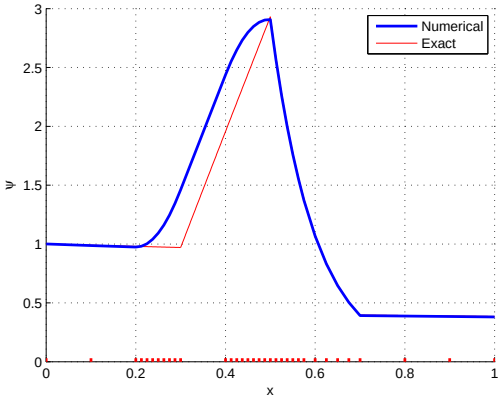
(c) 16 elements



(d) 20 elements



(e) 26 elements



(f) 33 elements

Figure 5.3: *Comparison of the numerical and exact solution of the second test problem. The initial number of elements is ten and in the end 33 (after five times cutting 30% of the elements). The numerical solution is plotted in blue and the exact solution in red. On the x-axis the distribution of elements is shown with small red lines.*

The difference between the exact and numerical solution can be found in figure 5.4. This plot supports what the solutions in figure 5.3 already showed. The deviation is quite large in the area where the algorithm didn't cut $\left(\frac{3}{10} \leq x \leq \frac{5}{10}\right)$. Based on the error the algorithm should refine the grid in this region. Based on the functional it didn't.

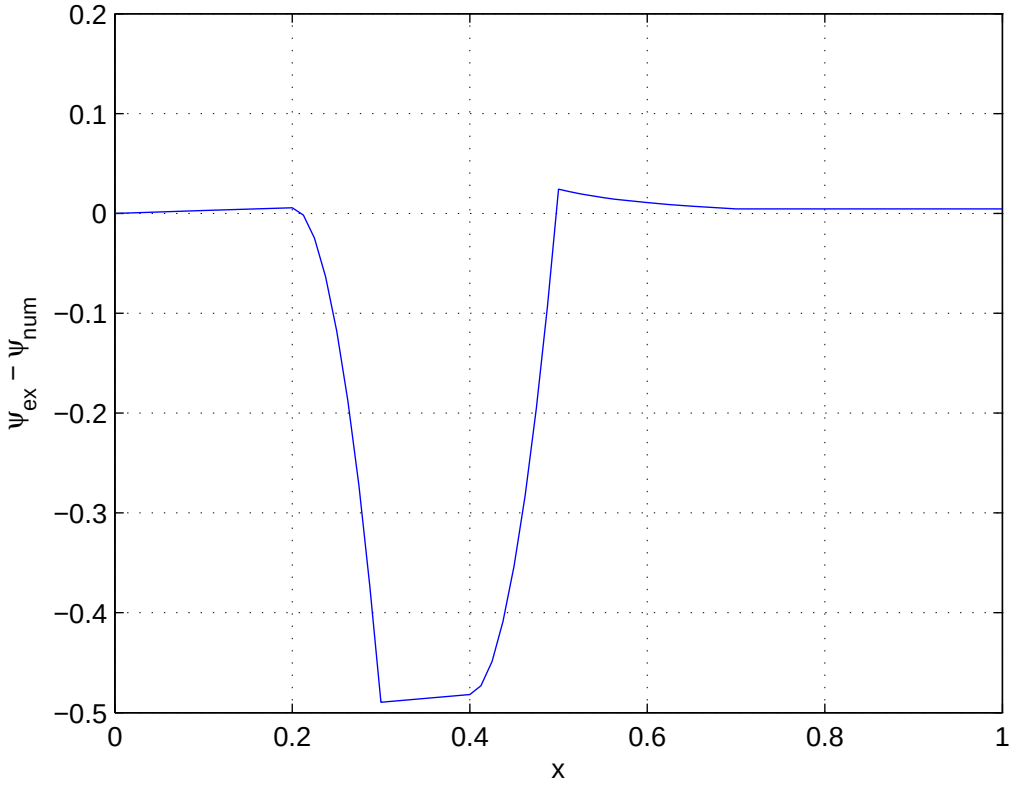
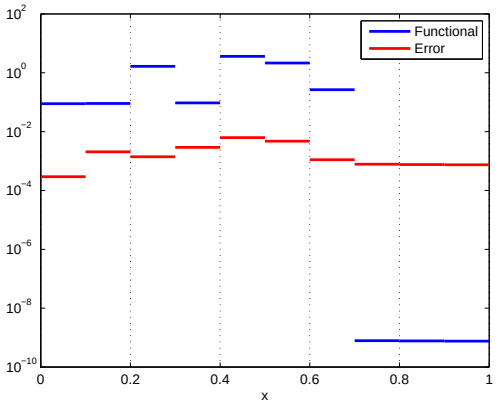
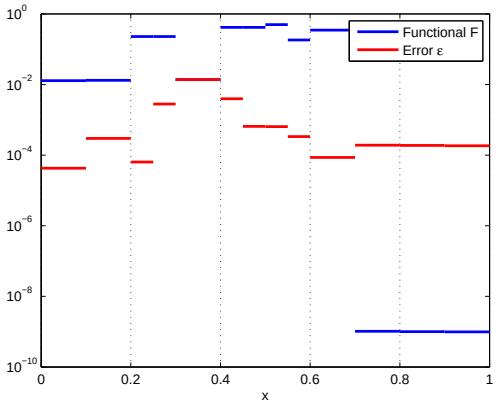


Figure 5.4: *Difference between the exact and numerical solution on the nodes for 33 elements on an adaptive grid.*

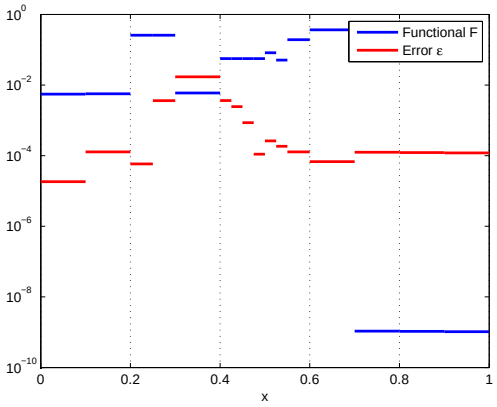
In figure 5.5 the functional F and error ϵ are plotted. The y-axis is logarithmic and note that the scale is not the same in each graph. Again there appears to be no correlation at all between the the two errors.
It is visible that the functional is always larger than the error in the area with $\sigma = 10$ and $S = 0$ $\left(\frac{5}{10} \leq x \leq \frac{7}{10}\right)$ so there it will continue to refine. The real error is larger for $\left(\frac{3}{10} \leq x \leq \frac{4}{10}\right)$, but because the functional is the criterion for refinement, it doesn't refine there. The functional is therefore not a good error estimate for this test problem.



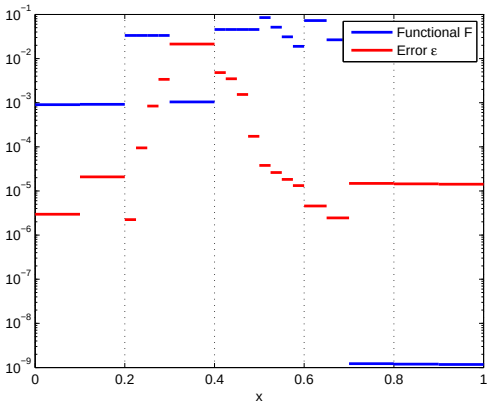
(a) 10 elements



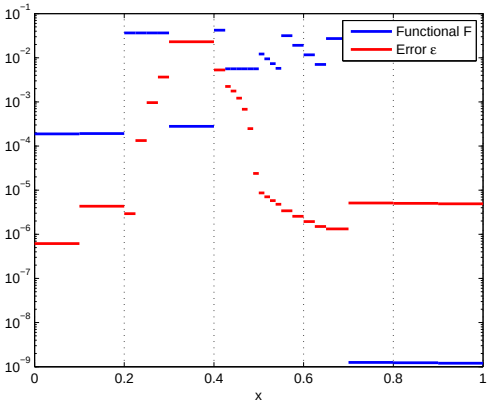
(b) 13 elements



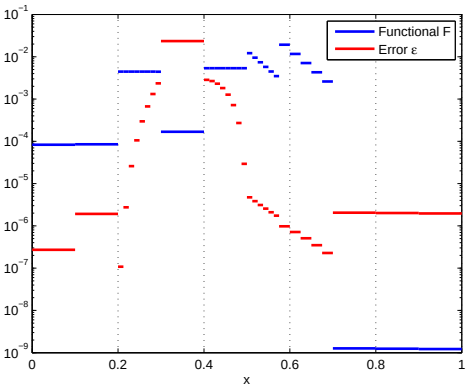
(c) 16 elements



(d) 20 elements



(e) 26 elements



(f) 33 elements

Figure 5.5: *Functional F and error ϵ of the second test problem are plotted. The initial number of elements is ten and in the end it is 33 (after five times cutting 30% of the elements). The functional is plotted in blue and the error ϵ in red.*

5.2.2 Comparison of adaptive and uniform grid

In figure 5.6 the same behaviour can be observed as in figure 4.6. The functional of the adaptive grid decreases faster than the functional for the uniform grid in both graphs. For the same value of the functional a lot more elements are needed in the uniform grid, almost 4 or 5 times as much. This means that the computation time is longer and therefore more expensive.

It was already concluded that the functional is not a good error estimate in this test problem and that is confirmed here again. The region that did need refinement was not refined, which made the solution worse. The ϵ behaves the same as in figure 4.6. For the uniform grid it alternates around a trendline with the same shape as the functional for the uniform grid. The adaptive grid also results in an ϵ which decreases, increases and finally reaches an equilibrium larger than the functional value. From this plot it can also be concluded that the solving method is not the best one.

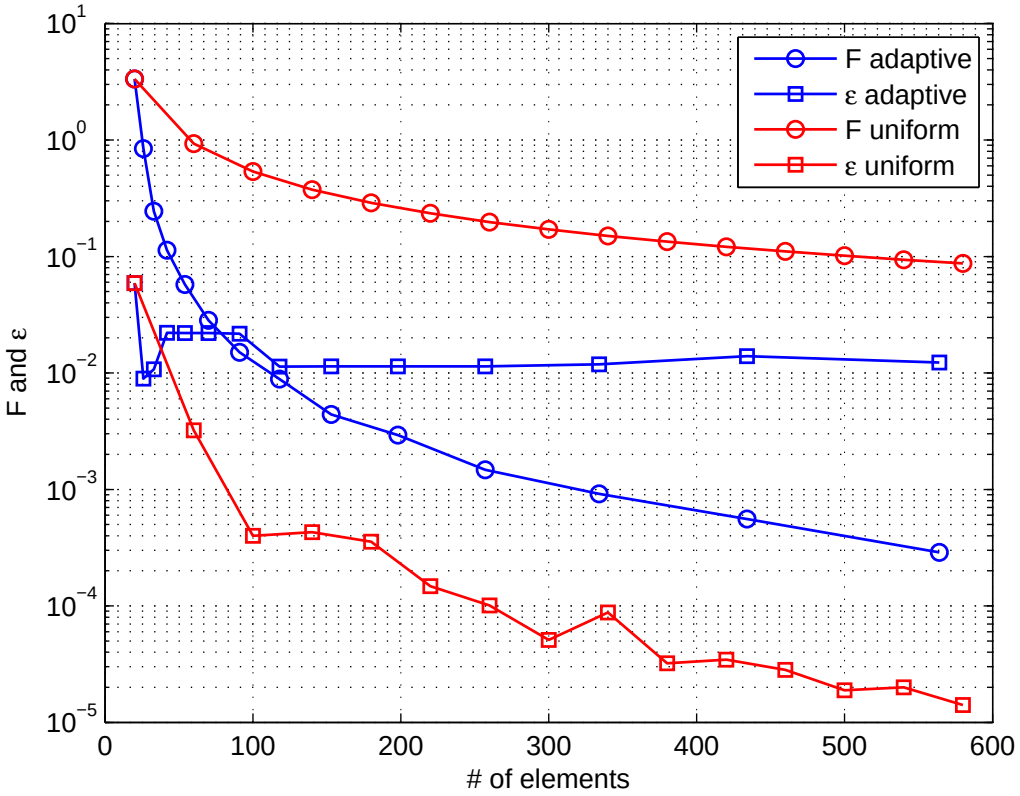


Figure 5.6: *Functional F and error ϵ for an adaptive and uniform grid. The functional is plotted with circles and the error with squares. The plot for the adaptive grid is shown in blue and the plot for the uniform grid in red. The y-axis has a logarithmic scale.*

Chapter 6

Short introduction to mathematics and theory of the two-way model

The research done and described in this report only considers the total cross section. Also only the neutron flux to the right was considered. In this chapter a short introduction to the mathematics and theory of the two-way model for neutron transport will be described. This can be used for further research.

6.1 Cross section for absorption and scatter

When a neutron approaches an atomic nucleus, it either scatters or is absorbed. The absorption cross section is a measure for the probability of an absorption process while the scatter cross section is a measure for the probability of a scatter process. When these cross sections are added the total cross section is found.

$$\sigma_t = \sigma_a + \sigma_s \tag{6.1}$$

During the research done only the total cross section was taken into account. To get a fully representative model it is required to describe the neutron interactions also in terms of absorption and scatter.

6.2 Mathematical description of σ_a and σ_s

The scattered neutrons can be seen as a source from the other side of an element. This results in two equations that have to be solved instead of one. The part of the source due to scatter can hence be written as

$$\begin{aligned} s_{\text{scat}} &= s^+_{\text{scat}} + s^-_{\text{scat}} \\ &= \sigma_s \left(\psi^+ + \psi^-\right) \\ &= \sigma_s \psi \end{aligned} \tag{6.2}$$

In which ψ^+ is the neutron flux to the right and ψ^- is the neutron flux to the left. These two added represent the total neutron flux.

In the neutron transport equation (equation 2.1) scatter does occur, so the operator $\mathcal{L}$ will change. It is repeated here.

$$\mathcal{L} = \hat{\mathbf{\Omega}} \cdot \nabla + \sigma_t - \int \sigma_s(\hat{\mathbf{\Omega}} \cdot \hat{\mathbf{\Omega}}')d\hat{\mathbf{\Omega}}' \tag{6.3}$$

The operator $\mathcal{L}$ consists of $\hat{\mathbf{\Omega}}$, the unit vector in the direction of motion. For the neutron flux to the right this reduces to 1. For the neutron flux to the left this reduces to -1. The ∇ reduces to $\mathrm{d}/\mathrm{d}x$, because it is a one-dimensional case. The scatter is taken into account this time, so that part can not be neglected. Because the process of scatter is isotropic, half of the scatter is to the left and half of the scatter is to the right.
The source term s is specified per equation. The first equation considers the neutron flux to the right, so the s^+_{ext} is the source on the left of the medium. The second equation considers the neutron flux to the left, so there the s^-_{ext} is the source on the right of the medium.
The result is the next system of equations

$$\frac{\mathrm{d}\psi^+}{\mathrm{d}x} + \sigma_t\psi^+ = s^+_{ext} + \frac{\sigma_s}{2}\left(\psi^+ + \psi^-\right) \tag{6.4}$$

$$-\frac{\mathrm{d}\psi^-}{\mathrm{d}x} + \sigma_t\psi^- = s^-_{ext} + \frac{\sigma_s}{2}\left(\psi^+ + \psi^-\right) \tag{6.5}$$

To be able to solve them they have to be rewritten to

$$\frac{\mathrm{d}\psi^+}{\mathrm{d}x} + \sigma_t\psi^+ - \frac{\sigma_s}{2}\left(\psi^+ + \psi^-\right) = s^+_{ext} \tag{6.6}$$

$$-\frac{\mathrm{d}\psi^-}{\mathrm{d}x} + \sigma_t\psi^- - \frac{\sigma_s}{2}\left(\psi^+ + \psi^-\right) = s^-_{ext} \tag{6.7}$$

It is visible that ψ^+ and ψ^- occur in both equations 6.6 and 6.7. This results in a coupled algorithm.

Unfortunately, during this project there was no time remaining to further elaborate the mathematics or to solve this two-way model.

Chapter 7

Conclusion and discussion

We have studied the application of adaptive mesh refinement in nuclear reactor physics to solve the neutron transport equation. In this chapter the most important results from the first and second test problem will be repeated and an overall conclusion is given. There are also recommendations for further research.

7.1 Test problem 1

The most important result from this test problem is visible in figure 4.6. There is no relation between the functional and the ϵ for the adaptive grid, so the functional is not a good measure for the error. It can also be seen that for the uniform grid the solution is not always getting better, because the ϵ alternates. The way of solving with a least squares functional and the FEM doesn't give the correct solution.

7.2 Test problem 2

The most important result from this test problem is shown in figure 5.4. Based on the error the algorithm should refine the grid in this region, based on the functional it didn't. There is no correlation at all between the functional and the error.

7.3 Overall conclusion

In conclusion, the adaptive mesh refinement method is a good method to find a solution to the simplified neutron transport equation. It solves the problem quicker than the uniform grid with the same accuracy. This means that it is also cheaper. It takes some extra time because the solution has to be calculated a few times, but it is quicker then the calculation for an uniform grid in which the smallest elements are of the same size.

It is not applicable with this functional as error estimate, because the functional is influenced too much by the values of σ and S. This was visible in the both test problems where there were different areas with large fluctuations in the values of σ or S. It seems that cutting 30% of the elements is not enough. The elements in the part where the neutron flux decreases are in the 30% of the elements which are halved. The region with increasing neutron flux prior to the region with decreasing neutron flux is never refined because the functional is too small there.

Besides that the functional is not a good error estimate, the way of finding the solution is not the best one. The least squares functional used for the computation of the solution is influenced too much by the value of σ_t.

7.4 Further research

The most important part to investigate is a good error estimate. This is something to examine more thoroughly. The functional is not the best error estimate, so it is necessary to investigate something else which better represents the real error. The algorithm has to refine more in regions with a steep slope, so taking the derivative as criterion might be better then the current functional.
The least squares functional used for the computation of the solution doesn't give the best result. It might be a solution to scale it with the σ_t to find a better solution for the differential equation.
In this report only the neutron flux to the right was described. In chapter 6 an introduction to the mathematics and theory of the two-way model is given. To obtain a more realistic model this has to be implemented in the algorithm.

Bibliography

[1] J.J. Duderstadt and L.J. Hamilton, *Nuclear reactor analysis*, Wiley, 1976.

[2] J. van Kan, A. Segal, and F. Vermolen, *Numerical methods in scientific computing*, VSSD, 2005.

[3] Yaqi Wang, Wolfgang Bangerth, and Jean Ragusa, *Three-dimensional h-adaptivity for the multigroup neutron diffusion equations*, Physics in nuclear energy **51** (2008), 543–555.