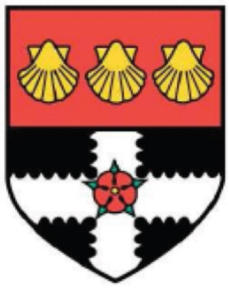


UNIVERSITY OF READING
Department of Mathematics

The University of Reading
The Department of Mathematics and Statistics
MSc Dissertation

SELF - CONSISTENT FIELD CALCULATIONS ON A VARIABLE RESOLUTION GRID

Author:
Melios Michael

Supervisors:
Professor Michael J. Baines
Professor Mark W. Matsen

August 2011
This dissertation is submitted to the Department of Mathematics in partial fulfilment of the requirements for the degree of Master of Science

Declaration

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

Signed .....

Abstract

The Self-Consistent Field Theory, otherwise known as Mean Field Theory, represents interactions by two static fields acting on two polymer segments (A, B). The model corresponds to a melt of AB diblock copolymers subject to a local incompressibility constraint. Exploiting the simplest classical microstructure (Lamellar) the density distributions of the copolymer blocks are computed by applying the Crank-Nicolson algorithm on a uniform mesh. The aim of this dissertation is to increase the numerical efficiency of the calculations by employing an adaptive mesh using subdivision. The dissertation contains a study of the effectiveness of using the h refinement technique on the computation of the copolymer propagators with the equation $\frac{\partial}{\partial s}q(\mathbf{r},s) = \frac{a^2N}{6}\nabla^2 q(\mathbf{r},s) - w(\mathbf{r},s)q(\mathbf{r},s)$. Preliminary studies are carried out using a uniform mesh method and globally refining the mesh before subdividing the mesh in areas of interest. Numerical results of the total partition function and the segment concentration distributions are compared and conclusions drawn on the space size step and local refinement factors.

Acknowledgements

I would like to thank all the staff of the Mathematics Department for being friendly and supportive throughout the duration of my studies. Particular thanks must go to Dr. Peter Sweby and Sue Davis for all their help and for their spontaneous replies and guidance. I would also like to thank Prof. Mark Matsen for his help and supervision in my dissertation. Particular thanks for his help, guidance, conversations, motivation and positivity in all matters and situations whether they reflected on the course or general topics and personal problems must go towards Prof. Mike Baines. Your supervision, ideas, jokes and patience have really been appreciated and inspiring.

Lastly I would like to thank my family and close friends for their understanding and support which made the completion of the course and dissertation easier and possible.

Contents

Chapter 1

Introduction

1.1 Background

A polymer can be defined as a macromolecule. It is essentially constructed from several repeated monomer building blocks or chemical units, linked together into one or more chains [4]. Typically there could be several hundred to several thousand monomer units in a polymer. When the monomers that build a chain are all of the same chemical structure, in other words if they are identical, the polymer is called a Homopolymer. If however it involves two or more chemical distinct monomers then, the result is termed a copolymer. In our current investigation, we will be considering block copolymers, these refer to molecules grouped together as blocks. A linear diblock copolymer is a polymer constructed by attaching one end of a linear homopolymer of one type of chemical units to the end of another linear homopolymer of distinct types, creating a longer but still linear molecule. Symbolically we denote A and B the two distinct types of chemical units or monomer and the portion of the chain which is of type A is referred to as the A block, and similarly the porton of the chain which is of type B is referred to as the B block. Thus, diblock copolymer refers to a copolymer comprised of two blocks of distinct species [6].

Figure 1.1: Classical Microstructures

A larger unit consisting of monomers grouped together each with a specific volume is called a segment. Unlike segments are incompatible and the segment segregates into an A and a B rich domain. This is termed Periodically Structured microstructures and there is a list of such Classical Microstructures e.g Spheres, Cylinders and the Lamella, as shown in Figure 1.1. The simplest of these ordered microphases is the Lamellar (L) phase in which the A and B monomers separate into A-rich and B-rich lamellae . It is observed to occur when the volume fractions of the two monomers are comparable at $f = \frac{1}{2}$, where f is the average volume fraction that distinguishes the number of segments that are of each type. Every AB diblock copolymer is thus characterised by N number of segments and f, the fraction [9] .

1.2 Reasons for Research

Block copolymer melts have become an excellent model for studying fundamental phenomena associated with molecular self-assembly. This interest is driven by the vast industrial and commercial applications of polymer materials.
The field of complex liquids includes a diverse range of molecular systems which involve molecules with two unfavourable contacts that tend to self assemble into ordered microstructures. Manipulating this tendency, researchers have created a large host of important applications as for example the emerging field of nanotechnology. Polymers have become a vital class of materials for the industrial sector. It stands as one of the dominant areas in soft considered matter physics.
Great attention has been given to the equilibrium phase behaviour and with it has emerged a thorough understanding in terms of simple intuitive explanations. The theoretical contributions to these efforts are largely attributed to mean-field calculations on a standard Gaussian model [4].

1.3 The Model

Our model considers individual molecules and it ignores their atomic structure as it does not explain the mesoscale behaviour of the melt. It is chosen because it is simple and valid. It will consist of n identical AB diblock copolymer molecules, N segments and f, the fraction that forms A block.

Using the Self Consistent Field Theory otherwise known as the Mean Field Theory (M-F-T), molecular interactions are replaced by fields which fluctuate since the molecules that create them move. The M-F-T focuses on a single molecule and

represents interactions by two static fields acting on A,B segments. The model corresponds to a melt of AB diblock copolymers subject to a local incompressibility constraint. It uses the applicability of the Gaussian model and develops the necessary statistical mechanics for a single chain which is subject to an external field, $w(\mathbf{r})$. According to the Gaussian model, diblock copolymers are treated as microscopic elastic threats [14].

We should define the following notation that will be used in our model:

- $r_i = i^{th}$ monomer
- $r_\alpha(s)$ = a function explaining the coarse-grained trajectory of the polymer
- s = a parameter indicating the interval of the chain (segment) and is defined within $0 \leq s \leq 1$
- subscript α is ued to label different molecules
- N total number of segments

When discretising the partial differential equation 1.2, s is treated similarly to a time variable and r as the space variable.

Each molecule is parametrised by a variable $\mathbf{s}$ that increases from 0 to 1 along the length. The partition function $\mathbf{Q}$ for a single copolymer experiencing chemical potentials q and q^* that exerts forces, respectively, on the A and B blocks is given by

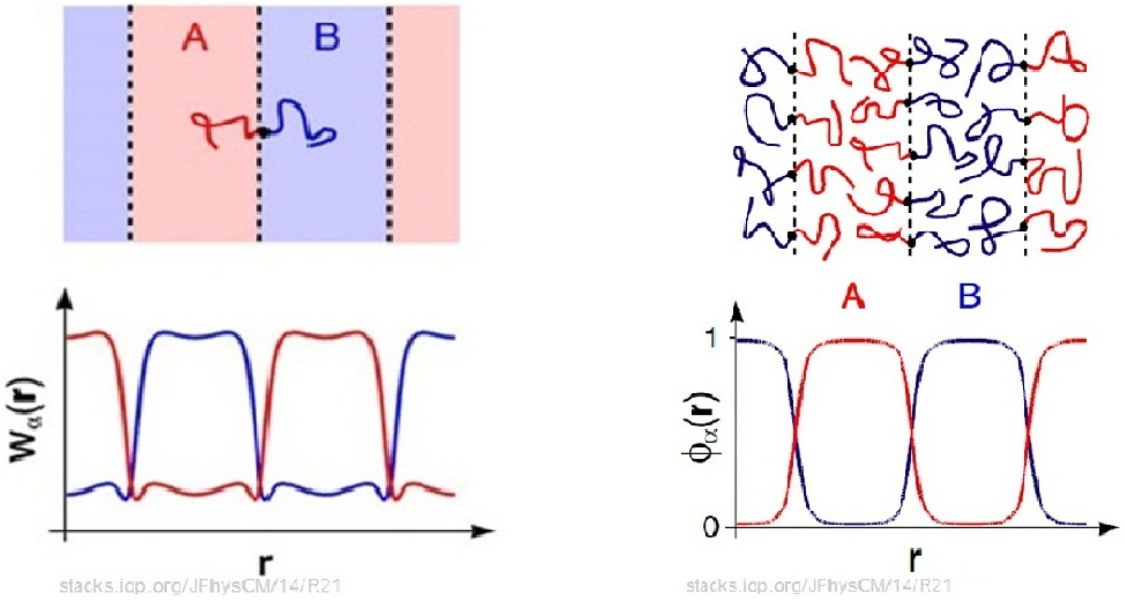
$$Q = \int q(\mathbf{r}, s)q^*(\mathbf{r}, s)dr \tag{1.1}$$

where the copolymer propagator satisfies

$$\frac{\partial}{\partial s}q(\mathbf{r}, s) = \frac{a^2 N}{6}\nabla^2 q(\mathbf{r}, s) - w(\mathbf{r}, s)q(\mathbf{r}, s) \tag{1.2}$$

and can be evaluated starting from $q(r, 0) = 1$. Similarly for $q^*(r, s)$ the differential equation to evaluate is identical but the right hand side is multiplied by -1 as follows

$$\frac{\partial}{\partial s}q^*(\mathbf{r}, s) = -\frac{\alpha^2 N}{6}\nabla^2 q^*(\mathbf{r}, s) + w(\mathbf{r}, s)q^*(\mathbf{r}, s) \tag{1.3}$$



(a) External Fields (b) Segment Concentration Dist.

Figure 1.2: [8]

and starting from $q(r,1) = 1$. The function $w(\mathbf{r}, s)$is given by

$$w(\mathbf{r}, s) = \begin{cases} w_A(\mathbf{r}), & 0 \leq s \leq f \\ w_B(\mathbf{r}), & f \leq s \leq 1 \end{cases} \tag{1.4}$$

For our case, a symmetric diblock copolymer melt, $f = \frac{1}{2}$, the model exhibits a microphase phase separation into a lamellar phase for segregation strengths χN, where χ is the Flory parameter that quantifies the repulsive interaction of the chemical units. The boundary conditions for this problem can either be periodic or reflective Neumann conditions [1] .

$\phi_A(\mathbf{r})$ and $\phi_B(\mathbf{r})$ correspond to ensemble-averaged segment concentration distributions and from the schematic diagram in Figure 1.2b we can see the concentrations of A and B segments at different points $\mathbf{r}$.

The incompressibility assumption is then given by $\phi_A + \phi_B = 1$. The segment concentrations are defined as follows:

$$\phi_A(\mathbf{r}) = \frac{V}{Q}\int_0^f q(\mathbf{r}, s)q^*(\mathbf{r}, s)ds \qquad \phi_B(\mathbf{r}) = \frac{V}{Q}\int_f^1 q(\mathbf{r}, s)q^*(\mathbf{r}, s)ds \tag{1.5}$$

1.4 Aim of this study

The main target of this study is to evaluate the exact equilibrium behaviour of a single diblock copolymer subject to the mean field, therefore it will be with reference to the partial differential equations (PDE) that the copolymer propagators satisfy in 1.2. Clearly it will be far from simple to calculate the truncation errors created by such schemes using analytic methods and as such, we have used other means of testing our results. Applying finite difference schemes based on forward time centred space, Crank-Nicolson solutions to the modified diffusion equation can model these polymeric materials that are used by the self-consistent field theory. Many attempts to solve these equations using the ordinary Crank-Nicolson algorithm on a uniform mesh have been made, but the aim is to increase the numerical efficiency of the calculation by employing an adaptive mesh. In order to solve equation 1.2 we need the establishing of the external fields $w_{\alpha,i}$. The results should be symmetric and the plot should look similar to Figure 1.2a. Other tests to perform are the validation of the incompressibility assumption and that the Q equation 1.1 is constant throughout the whole period. More analytical explanations on these tests will be given later. The preliminary tests using uniform meshes will set the grounds for the adaptive methods and hopefully produce more efficient results where the fields fluctuate and where polymer configurations intersect. As we will see further on, a fortran program will be used to apply the Crank-Nicolson scheme on an adaptive mesh and then results plotted in Matlab.

1.5 Adaptive Methods

The results of the Self-Consistent field equations we are going to evaluate depend on the reliability of the mathematical model and the accuracy of the numerical approximations. Using a standard uniform mesh, to which approximations are required at equally distant points, a large number of points would be required to achieve satisfactory accuracy levels. By doing so the cost of a processing unit that would be able to handle the data involved would be great. In addition it would cause extra unnecessary processing time. On the other hand, too few points would lead to bad approximations and large errors.

An adaptive approach is to achieve, in the most effective way, an approximate solution which is in the range of admissible accuracy (tolerance). In general, there are three forms of adaptive methods, these are the mesh refinement (h) - refining the grid locally (adding grid points), order enrichment (p) - changing the order of accuracy locally and mesh movement (r) - relocating a grid (moving grid points).

"Refining indicators" are often used to identify portions of the domain in need of additional resolution [10].

This dissertation will centre around the use of the local grid refinement. The aim of the h type adaptive procedure is to achieve a higher rate of convergence and thus reach the desired accuracy with minimal cost. Given that we have sufficient information describing where best to refine the mesh, then there are several ways of approach. Using h-refinement, we obtain a new mesh by simply subdiving the intervals.
Using this method we can break the mesh into smaller pieces when necessary and even coarsen the mesh where the solution is very smooth. However, large jumps in mesh size could cause numerical errors and an arbitrary number of unknowns will lead to unknown run times. The number of elements from one refinement to another may increase significantly. Another key point to raise is that the original location of nodes does not alter through successive refinements. Though new nodes are added and old ones may be deleted at each refinement stage, the initial orientation of the elemets does not change and thus highly distorted elements do not appear as when the r-method is used. The h method has to calculate how new nodes get incorporated but does not need to calculate a solution of a PDE to shift the nodes.

All in all the h method is one of the most widely used as it's a self-adaptive technique which requires minimal user interaction to activate the adaptive process. In Chapter 4 we will see how it is implemented in our case when dealing with the Self-Consistent equations.

Chapter 2

Preliminary Stage

We are interested in obtaining the solution to the Self-Consistent Field equations in which the normalised density distributions, $\phi_\alpha(\mathbf{r})$ reflect a lamellar morphology. In this symmetry, $\phi_\alpha(\mathbf{r})$ varies only along one axis, and exhibits a periodic variation. $\phi_\alpha(\mathbf{r})$ can thus be reduced to a function of only one co-ordinate, r, and be periodic. By concentrating exclusively on the lamellar morphology, the shortcomings of the unit-cell approximation used in the early self-consistent efforts were avoided while maintaining a manageable numerical task.
Only configurations where block A and block B are joined should be considered, so the partition function will be restricted to include only those cases. Furthermore, the periodicity of these functions mean that we need only evaluate them over a single period.
The reason for this prelminary stage is to approximate the partial differential equation 1.2 using a less sophisticated scheme and set some grounds for reference when using the Crank Nicolson scheme and adaptive techniques as we progress.

2.1 The Finite Difference Method

As mentioned in section 1.3 the copolymer propagators are defined by the following partial differential equation:

$$\frac{\partial}{\partial s}q(\mathbf{r},s) = \frac{a^2N}{6}\nabla^2 q(\mathbf{r},s) - w(\mathbf{r},s)q(\mathbf{r},s) \tag{2.1}$$

which can also be rewritten by setting $q_{i,j} = q(r_i, s_j)$ as follows:

$$q_s = \frac{a^2N}{6}q_{rr} - w(\mathbf{r},s)q \tag{2.2}$$

The finite differences method will be used to approximate the solution of the diffusion equation 2.1. The basic idea of the finite difference method of solving PDEs is to replace spatial and time derivatives by suitable approximations, then to numerically solve the resulting difference equations. Specifically instead of solving $q(r, s)$ with r continuous, we solve $q_{i,j} \equiv q(r_i, s_j)$ where $r_i \equiv i\delta x$ and $s_j \equiv j\delta s$. We will have a grid similar to Figure 2.1

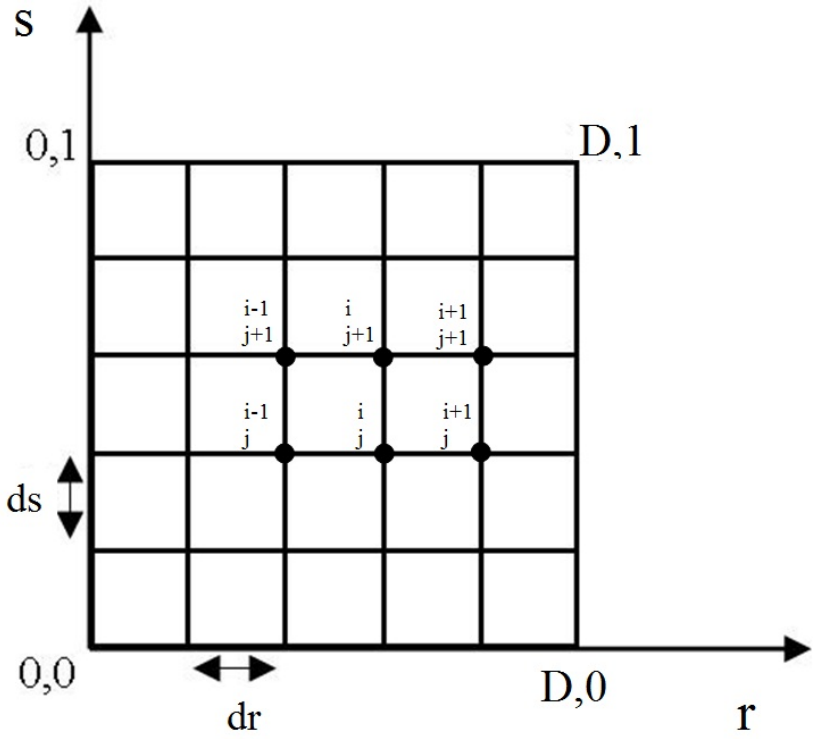


Figure 2.1: Grid

In all numerical solutions the continuous PDE is replaced with a discrete approximation [11]. The word "discrete" means that the numerical solution is known only at a finite number of points in the physical domain. Increasing the number of points not only increases the resolution but also ideally the accuracy of the numerical solution. We will thus create a **mesh**. The mesh is the set of locations where the discrete solution is computed. These points are called nodes. Applying the finite-difference method to a differential equation involves replacing all derivatives with difference formulas that involve only the discrete values associated with positions on the mesh. The rate at which the numerical solution approaches the true solution varies with the scheme.
Derivatives of q are approximated in terms of the values of q at grid points. Since:

$$\frac{\partial q}{\partial r} = \lim_{\Delta r \to 0} \frac{\Delta q}{\Delta r} \tag{2.3}$$

the derivatives evaluated at the grid points $(r, s) = (r_i, s_j)$ can be approximated in many different ways, the simplest being the following:

$$\text{Forward Difference} = \frac{\partial q}{\partial r}|_{r_i,s_j} \simeq \frac{q_{i+1,j} - q_{i,j}}{r_{i+1} - r_i} = \frac{q_{i+1,j} - q_{i,j}}{\delta r} \tag{2.4}$$

$$\text{Backward Difference} = \frac{\partial q}{\partial r}|_{r_i,s_j} \simeq \frac{q_{i,j} - q_{i-1,j}}{r_i - r_{i-1}} = \frac{q_{i,j} - q_{i-1,j}}{\delta r} \tag{2.5}$$

$$\text{Central Difference} = \frac{\partial q}{\partial r}|_{r_i,s_j} \simeq \frac{q_{i+1,j} - q_{i-1,j}}{r_{i+1} - r_{i-1}} = \frac{q_{i+1,j} - q_{i-1,j}}{\delta r} \tag{2.6}$$

The second derivative at the grid point (r_i, s_j) may be approximated from:

$$\frac{\partial^2 q}{\partial r^2} = \lim_{\Delta r \to 0} \frac{\Delta(\frac{\partial q}{\partial r})}{\Delta r} \tag{2.7}$$

Instead of using approximations for $\frac{\partial q}{\partial r}$ in terms of the values of q at r_{i+1}, r_i as for the forward difference or at the points r_i, r_{i-1} as for the backward difference, let's imagine instead that we evaluate it as $r_{i+\frac{1}{2}}, r_{i-\frac{1}{2}}$. Then using the central difference approximations for the spatial derivatives evaluated at these points,

$$\frac{\partial q}{\partial r}|_{r_{i+\frac{1}{2}},s_j} \simeq \frac{q_{i+1,j} - q_{i,j}}{r_{i+1} - r_i} = \frac{q_{i+1,j} - q_{i,j}}{\delta r} \tag{2.8}$$

$$\frac{\partial q}{\partial r}|_{r_{i-\frac{1}{2}},s_j} \simeq \frac{q_{i,j} - q_{i-1,j}}{r_i - r_{i-1}} = \frac{q_{i,j} - q_{i-1,j}}{\delta r} \tag{2.9}$$

Therefore,

$$\frac{\partial^2 q}{\partial r^2}|_{r_i} \simeq \frac{\frac{\partial q}{\partial r}|_{r_{i+\frac{1}{2}}} - \frac{\partial q}{\partial r}|_{r_{i-\frac{1}{2}}}}{r_{i+\frac{1}{2}} - r_{i-\frac{1}{2}}} = \frac{q_{i+1,j} - 2q_{i,j} + q_{i-1,j}}{(\delta r)^2} \tag{2.10}$$

We can approximate time derivatives in the same way. For example, the forward difference approximation for $\frac{\partial q}{\partial s}$ at the grid points (r_i, s_j) is:

$$\frac{\partial q}{\partial s}|_{r_i,s_j} \simeq \frac{q_{i,j+1} - q_{i,j}}{s_{j+1} - s_j} = \frac{q_{i,j+1} - q_{i,j}}{\delta s} \tag{2.11}$$

The error in the differential equation is called the truncation error.
In the computer program in FORTRAN, that we will create, the following notation will be used:

- N_x will be the total number of spatial nodes, including those on the boundary

and dr the size of the space step

- D will denote the total length of space, $0 \leq s \leq D$
- t_{max} the total length of time and as previously mentioned, the segments s behave similarly to the time variable so $t_{max} = 1$
- N_t the total number of time steps and ds the size of time step
- finally $r(i) = (i-1) * dr$ where $dr = \frac{D}{N_{x-1}}$ where$i = 1, ..., N_x$ and $ds = \frac{t_{max}}{N_{t-1}}$

2.2 A fully Explicit scheme

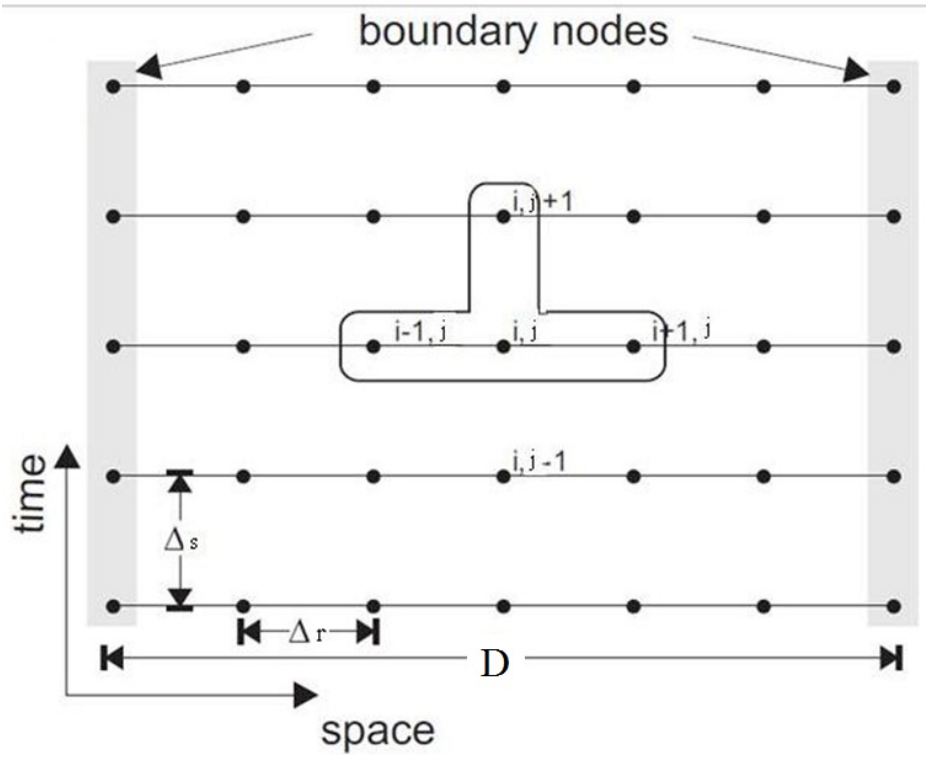


Figure 2.2: [17] Explicit Scheme - Grid

If we use 2.10, 2.11, 2.1 and rearrange we obtain the following difference equation which can be iterated to find the approimate solution to 2.1.

$$q_{i,j+1} = q_{i,j} + \frac{ds}{(dr)^2}(q_{i+1,j} - 2q_{i,j} + q_{i-1,j}) - w_i q_{i,j} \tag{2.12}$$

This equation is called explicit because the computation of q at $s_j + 1$ is completely determined by our computation of q at s_j. This scheme is also called consistent because the finite difference approximations have a truncation error that

approaches zero in the limit that $\delta s \to 0$ and $\delta r \to 0$. The scheme is found to be "consistent", first order in time and second order in space.
In order to guarantee that the scheme will give a good approximation to the true solution of the diffusion equation, when the discretised equation approaches the exact solution then the numerical scheme is termed convergent.
For a linear solution such as the diffusion equation, convergence is dependent on the stability of the numerical scheme, it is termed stable if the amplification factor remains bounded during calculations. According to the *Lax Equivalence theorem* schemes that are convergent are those that are consistent and stable [7].
Therefore, for a properly posed initial value problem for a linear PDE and a consistent finite difference approximation, stability is the necessary and sufficient condition of convergence. The explicit scheme 2.12 is stable and therefore convergent when

$$\frac{\delta s}{(\delta r)^2} \leq \frac{1}{2} \tag{2.13}$$

The main advantage of the explicit scheme is that it's easy to solve numerically. However, the stability condition raises issues as we are bound by the physical problem on having total space length $D = 2.336784$ and total time variable $t_{max} = 1$ and thus restricting us on the choice of total number of space and time steps N_x, N_t. We can therefore conclude that the stability condition and the scheme's first order in time truncation error restrict the accuracy of our numerical result.

2.2.1 Applying the Explicit Scheme

Since the explicit Scheme is relatively easy to solve numerically, we will use a FORTRAN program that will numerically solve the partial differential equation 2.1 that defines the copolymer propagators.

Using the finite difference method as shown in the previous section we can obtain the discretised equation of 2.1,

$$q_{i,j+1} = q_{i,j} + \frac{1}{6}\frac{ds}{(dr)^2}(q_{i+1,j} - 2q_{i,j} + q_{i-1,j}) - w_j q_{i,j} \tag{2.14}$$

For the q^* the PDE is identical to 2.1 but with the right hand side multiplied by -1. Therefore the discretised form is very similar but with a small difference as shown below,

$$q^*_{i,\mathbf{j-1}} = q^*_{i,j} + \frac{1}{6}\frac{ds}{(dr)^2}(q^*_{i+1,j} - 2q^*_{i,j} + q^*_{i-1,j}) - w_j q^*_{i,j} \tag{2.15}$$

In both cases, equations 2.14 and 2.15, we use reflective boundary conditions

which are defined in the program as, $q_{0,j} = q_{2,j}$ and $q_{N_x+1,j} = q_{N_{x-1},j}$. They are defined similarly for q^* which is denoted as q_1 in the program. The initial data for the q propagator is $q(r,0) = 1$ and for the q^* is $q^*(r,N_t) = 1$.

The function w_{α_j}, the external field, is given by

$$w_{\alpha,i} = \sum_{j=1}^{\infty} W_{\alpha,j} f_j(r) \tag{2.16}$$

where α is equal to either A or B depending on the segment and

$$f_j(r) = \sqrt{2}\cos\left(\frac{2\pi jr}{D}\right) \tag{2.17}$$

The fields are calculated over the whole domain using sample data for $w_{\alpha,i}$. Most of the program trials are executed with the use of $\chi N = 100.0$ and a sample of 80 values of w_A and 80 values of w_B. During the computation of q, w_A is used for the first half of the time (s) steps and then w_B for the remaining steps. However for q^*, w_B is used first and w_A afterwards. When plotting the results, the external fields should look similar to Figure 1.2a and indeed the FORTRAN program outputs results for the fields that are plotted using MATLAB and are shown in the figure below.

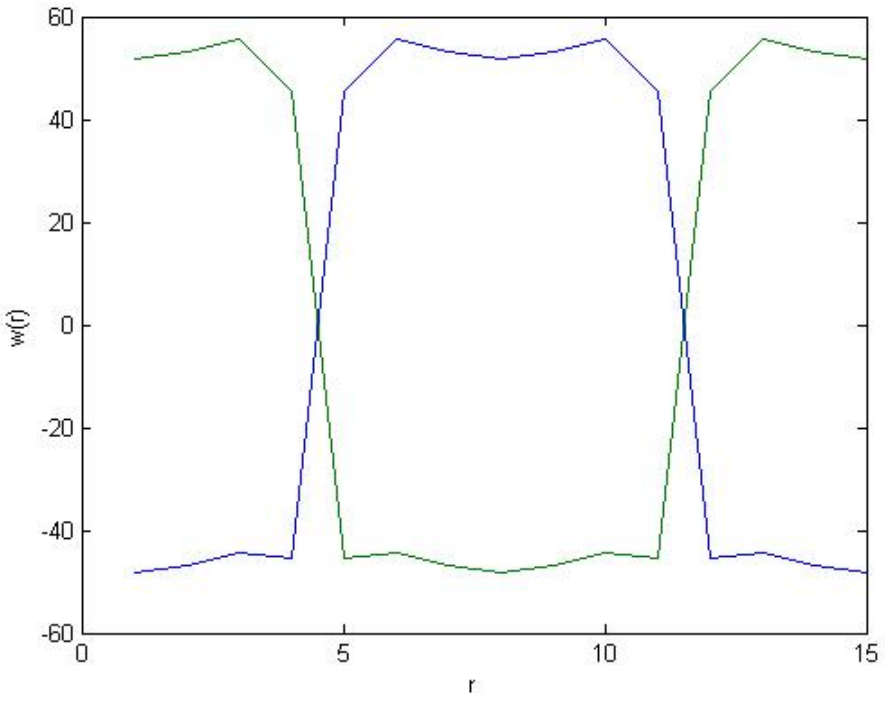


Figure 2.3: External Fields - $N_x = 15$, $N_t = 80$

In Figure 2.4 the blue line denotes w_A and the green line w_B. The external fields have been computed using 15 space steps and 80 time steps for $0 \leq r \leq 2.336784$ and $0 \leq s \leq 1$ and because there are only 15 space steps the plot is not smooth nor is it accurate enough. We therefore try plotting it using $N_x = 100$ and $N_t = 3592$, the results were more sufficient and presented in the figure below.

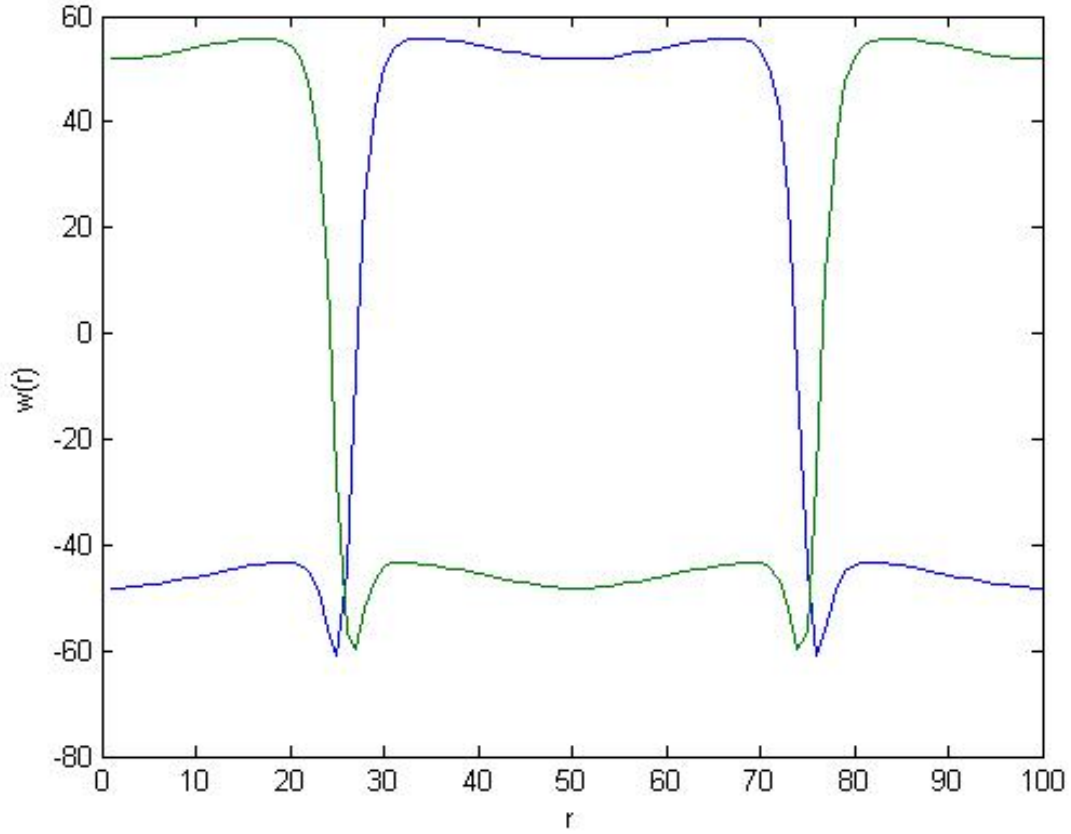


Figure 2.4: External Fields - $N_x = 100$, $N_t = 3592$

The biggest issue with the explicit scheme is the stability condition that needs to be satisfied in order for the scheme to converge, so the total space steps and time steps should be chosen with great care. The FORTRAN program helps tackle the issue by informing the user of the minimum N_t number of steps that correspond to each N_x steps. As an example in Figure 2.5, the user selected 100 space steps and the program notified that at least 3592 time steps should be selected for the stability condition to be met.

Figure 2.5: Program - Stability condition notification

The numerical solution of q when choosing 15 space steps and 80 time steps has been plotted in MATLAB and is shown in Figure 2.6. The results could also be plotted on a logarithmic scale so that the plots show more details in all the domain and the propagator's behaviour, however the propagator plots are only shown for reference. The aim is to compute the segment concentrations with great accuracy and to meet the incompressibility condition

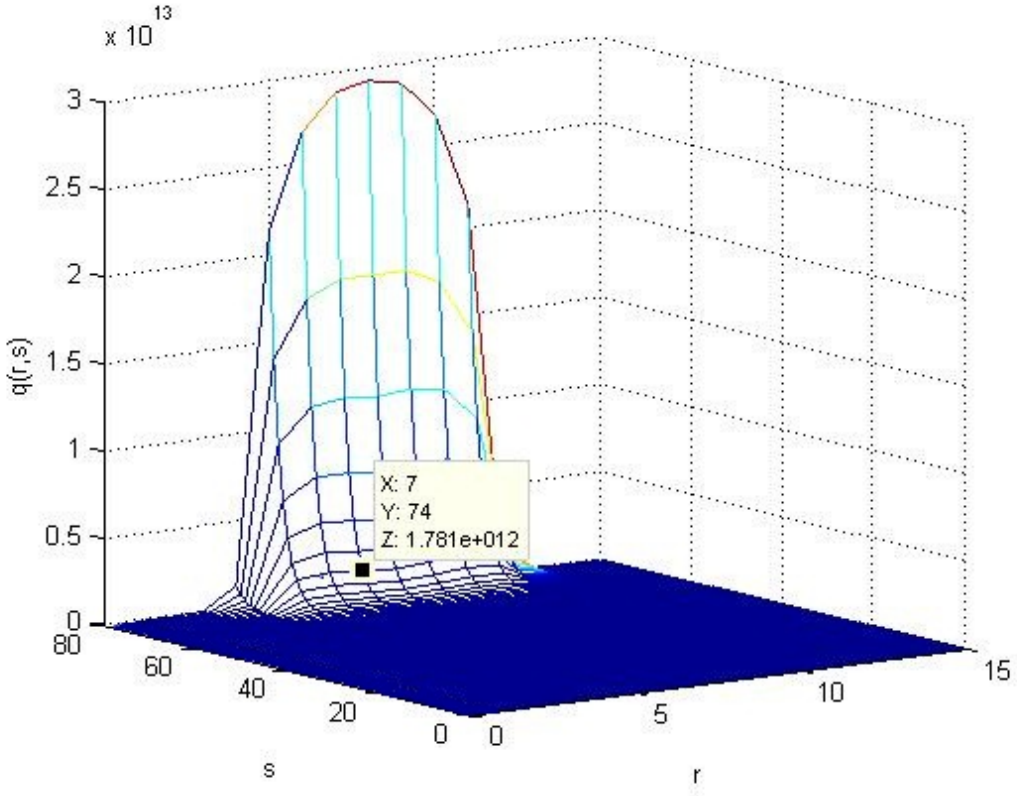


Figure 2.6: q propagator - explicit scheme - $N_x = 15$, $N_t = 80$

In order to check the correctness of the plot, a small test we could perform is to remove the $w_{\alpha,i}$ function, the external fields, from the partial differential equation 2.14. This would transform our PDE in a pure diffusion equation and in fact we would be solving the Heat Equation $\frac{\partial q}{\partial s} = \frac{\partial^2 q}{\partial r^2}$, of which the expected result is known. The discretised form of our PDE would then look like equation 2.18. The fact that our initial data is equal to 1 throughout the whole domain including the boundary points, would lead us to expect a constant value of 1 as a solution and a flat plot 2.7

$$q_{i,j+1} = q_{i,j} + \frac{1}{6}\frac{ds}{(dr)^2}(q_{i+1,j} - 2q_{i,j} + q_{i-1,j}) \tag{2.18}$$

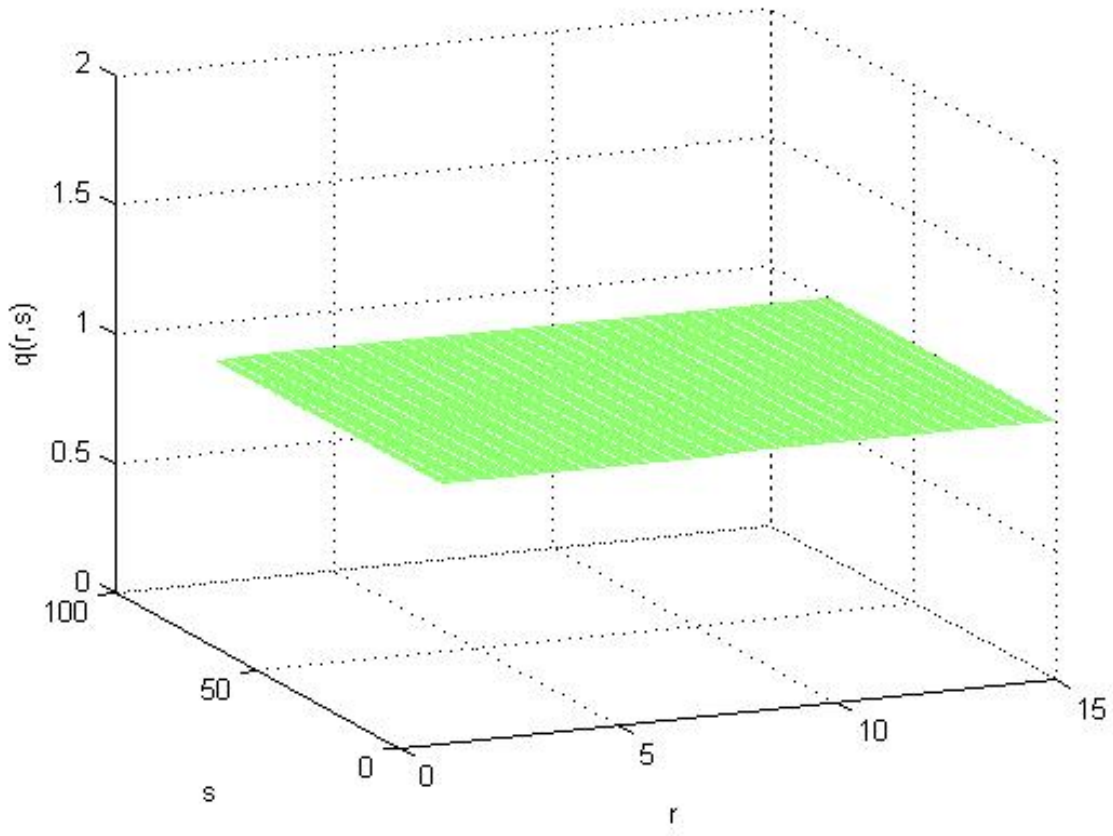


Figure 2.7: q propagator - removing external fields

Now that we have validated that the explicit scheme works correctly, another test run could be to plot the results of the PDE with segragation strength $\chi N = 1$ instead of 100. This would help us understand how the $w_{\alpha,i}$ function affects our results and the overall copolymer propagator. The outcome is shown in figure 2.8

and from a different angle in figure 2.9.

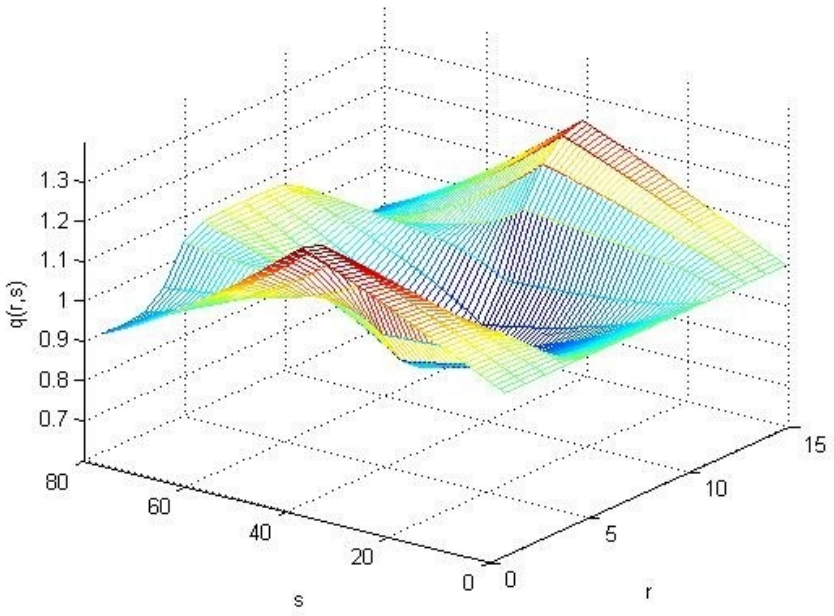


Figure 2.8: q propagator - $\chi N = 1$

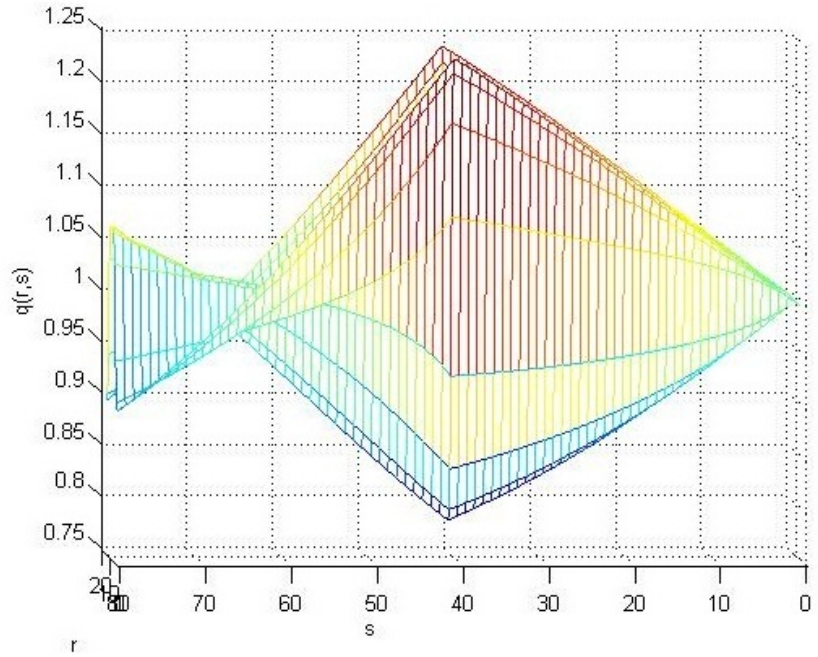


Figure 2.9: q propagator - $\chi N = 1$ - different angle

Looking at the results we could see how the $w_{\alpha,i}$ function chages the plot from Figure 2.7 to Figure 2.9 and then Figure 2.6. We could therefore conclude that the

larger the values of $w_{\alpha,i}$, the larger the values of q will be. An exponential growth appears in the values of q. This could not have been caused by the diffusion part of the equation as all the research concludes that diffusion satisfies the solution which appears in Figure 2.7. We therefore refer to the analytic solution of $\frac{\partial q}{\partial s} = -w_{\alpha,i}q$, which is $q = Ae^{-sw}$. If $w_{\alpha,i} < 0$ then we would, indeed expect exponential growth. There is no restriction in the $w_{\alpha,i}$ values so they could be negative and thus the results in Figure 2.6 are explained. These assumptions and conclusions are important and will help us analyse the results of the Crank Nicolson scheme used later and set the grounds for reference and comparison in the later stages.

We will now consider the q^* propagator and check if its discretised equation behaves in a similar manner and if the $w_{\alpha,i}$ function affects it in the same way as it does for the q propagator. Once more we choose $N_x = 15$ and $N_t = 80$.
The discretised equation 2.15 works backwards, the initial data is given for the last time step, N_t, and the FORTRAN program computes each time step in reverse order to q. The $w_{B,i}$ function is used for the first half of the time steps and then the $w_{A,i}$ function for the remaining steps. The results are plotted in MATLAB and shown in Figure 2.10

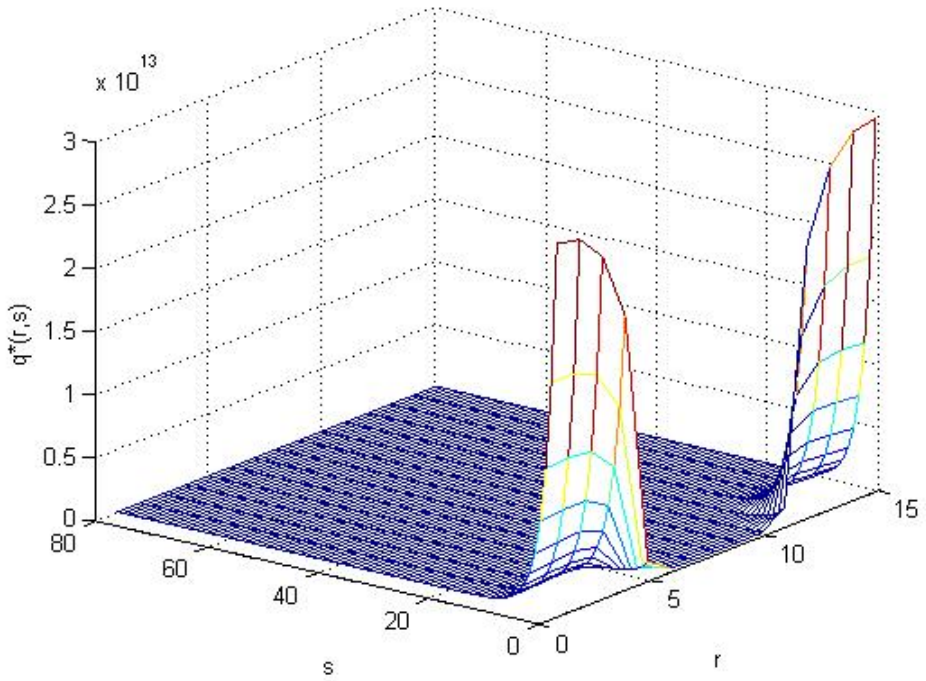


Figure 2.10: q^* propagator $N_x = 15$, $N_t = 80$

We then remove the $w_{\alpha,i}$ function in order to check if the discretisation of the

remaining diffusion equation produces the same result as the q propagator under the same conditions. The discretised equation without the external fields is now,

$$q_{i,\mathbf{j-1}} = q_{i,j} + \frac{1}{6}\frac{ds}{(dr)^2}(q_{i+1,j} - 2q_{i,j} + q_{i-1,j}) \tag{2.19}$$

and the results satisfy our expectations, as shown in Figure 2.11.

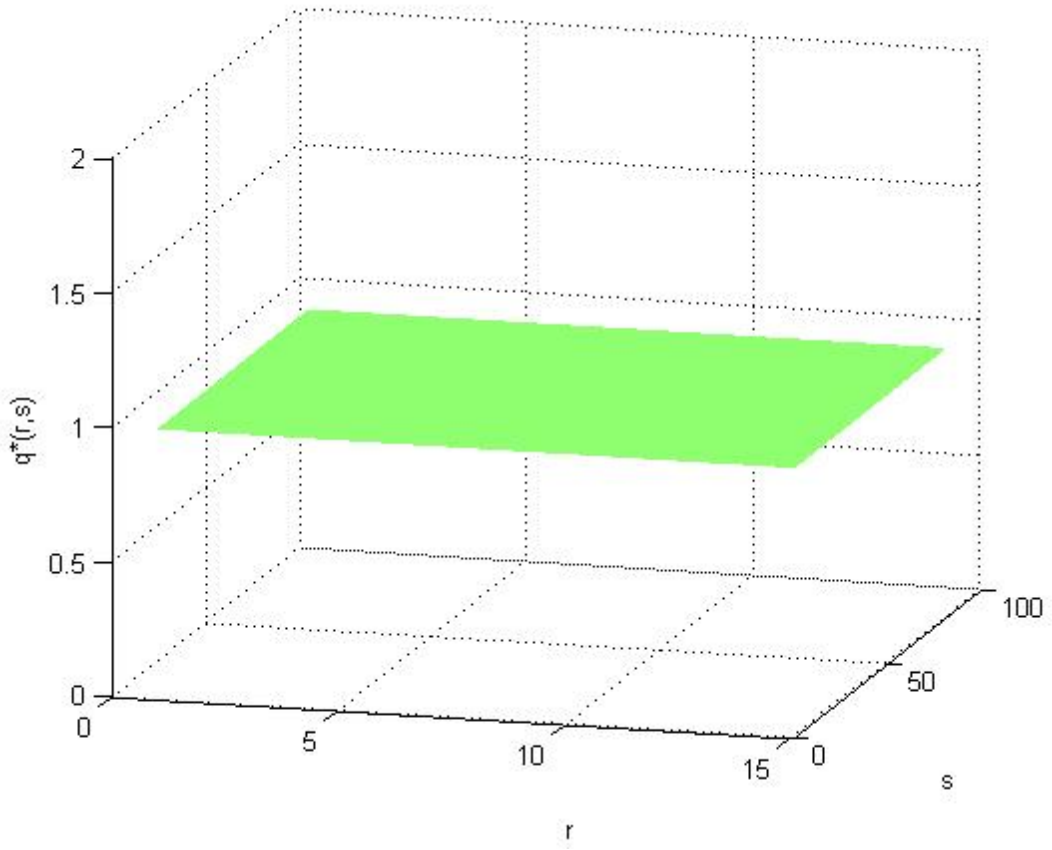


Figure 2.11: q^* propagator - external fields removed

Plotting the results of the PDE with segregation strength $\chi N = 1$ instead of 100 for q^* as well, allow us to make better comparisons between the two propagators. That is because the $w_{\alpha,i}$ function will be weaker and thus it will have less effect on the PDE and will not experience such great exponential growth. The outcome is shown in Figure 2.12 and from a different angle in figure 2.13.

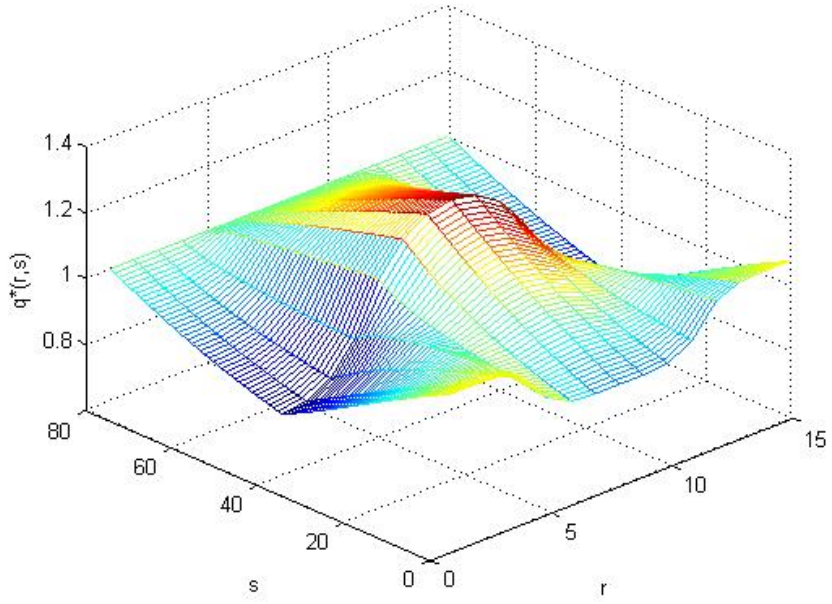


Figure 2.12: q^* propagator - $\chi N = 1$

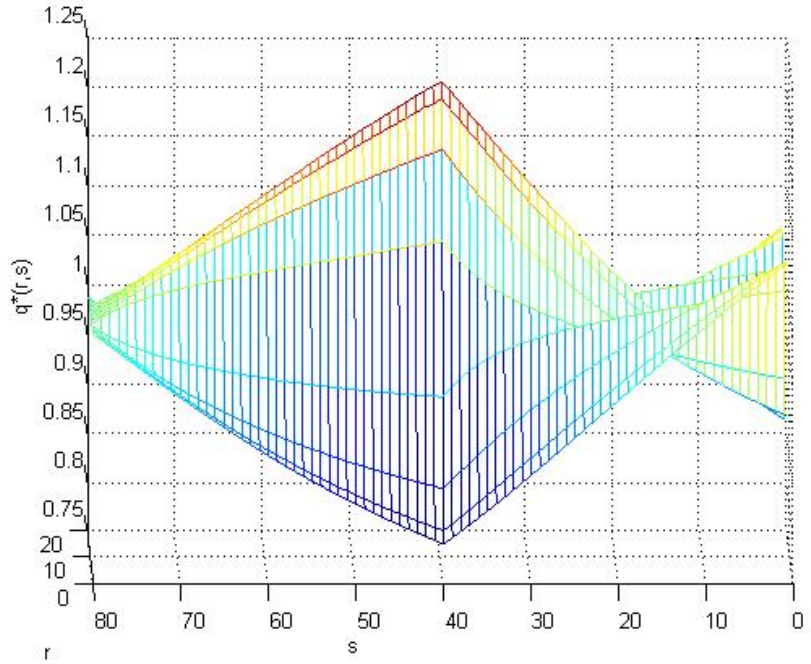


Figure 2.13: q^* propagator - $\chi N = 1$ - different angle

Comparing Figure 2.12 of the q^* copolymer propagator to Figure 2.8 of copolymer propagator q, we can notice two relatively similar plots, with the first starting from 1 in the last time step, $N_t = 80$, and the second one starting from 1 at the initial time step j=0. They are both relatively symmetrical and q^* being almost

equal to q if rotated through 180°. Looking at the plots from a different angle, q^* in Figure 2.13 and q in Figure 2.9, we can notice that they expand smoothly until the space step i=40. The reason for this change is the fact that in both cases the middle space step i=40 is the switching step from $w_{A,i}$ to $w_{B,i}$ and vice versa. However, these figures are valid for segregation strength $\chi N = 1$, when we turn back to $\chi N = 100$ the program outputs the corresponding plots for q and q^* as in Figures 2.6 and 2.10, respectively.

The problems with using the fully explicit scheme are not so serious until we try to numerically solve the segment concentration equations 1.5. The problems would mostly be attributed to the truncation error of the scheme being first order in time. The current issue though is the fact that by increasing the total space steps from 15 to a larger more realistic number e.g $N_x = 1000$ or even more, which would give us better resolution and also the satisfactory levels of accuracy, we would require the use of a huge number of time steps e.g. $N_t = 365533$. This would create a larger error in the calculations but also the processing time that would take to compute the solution for much greater number of space steps would be significantly more and possibly require a bigger computer processor. As an example of the greater accuracy in the numerical solution of the propagators using $N_x = 100$ instead of 15 and $N_t = 3592$ instead of 80, Figures 2.14 and 2.15 present the resullts for q and q^* respectively.

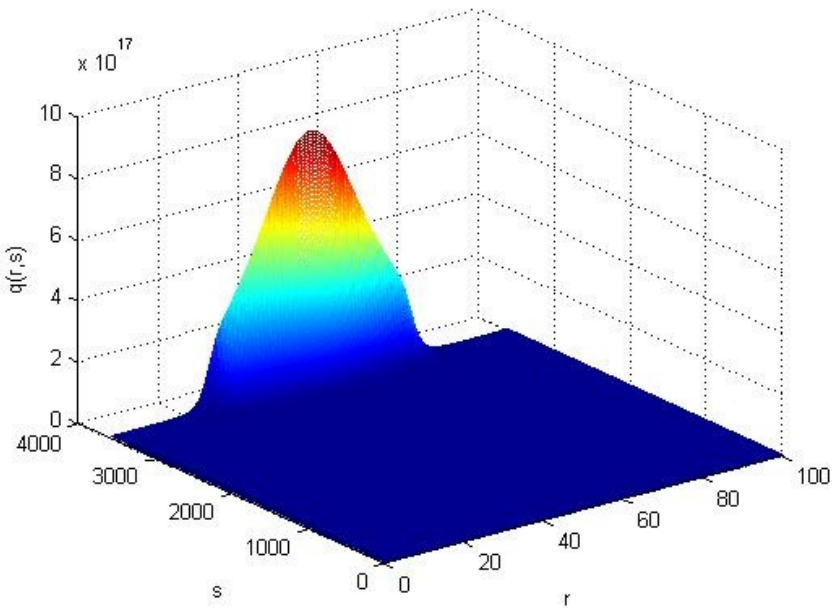


Figure 2.14: q propagator - $N_x = 100$, $N_t = 3592$

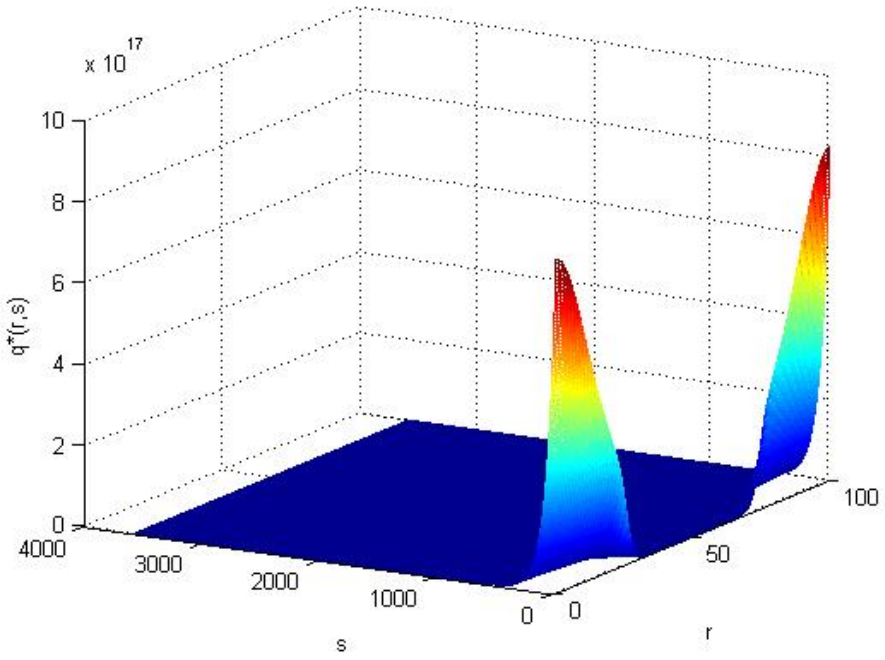


Figure 2.15: q^* propagator - $N_x = 100$, $N_t = 3592$

2.3 Numerical Integration

Once $q(r,s)$ and $q^*(r,s)$ have been calculated using the numerical scheme, we must solve equations 2.20 and 2.21.

$$Q = \int q(\mathbf{r},s)q^*(\mathbf{r},s)dr \tag{2.20}$$

$$\phi_A(\mathbf{r}) = \frac{V}{Q}\int_0^f g(\mathbf{r},s)q^*(\mathbf{r},s)ds \qquad \phi_B(\mathbf{r}) = \frac{V}{Q}\int_f^1 q(\mathbf{r},s)q^*(\mathbf{r},s)ds \tag{2.21}$$

Equation 2.20 indicates the partition function and 2.21 the ensemble-averaged segment concentration distributions and their results should resemble Figure 1.2b. Both equations involve integrating the product of $q(r,s)$ and $q^*(r,s)$,($X = q(r,s).q^*(r,s)$). There are several method of numerical integration of varying accuracy and ease of use. The most commonly used method and the one we are going to apply to our problem is the composite trapezoidal rule.

The trapezoidal rule is a simple formula that estimates this integral. It basically replaces the integral by a discrete sum that can be interpreted as the sum of areas

of trapezoids. A trapezoid is a four-sided region with two opposite sides parallel. The area of a trapezoid is the average length of the parallel sides, times the distance between them.
Given the partition [0,D] we can define the associated trapezoid sum to correspond to the area under the X-line.

The FORTRAN program will read all the numerical results of $q(r,s)$ and $q^*(r,s)$ that were previously produced by the explicit scheme. The two equations differ in the aspect that Equation 2.20 requires the numerical solution to be divided into areas along the space axis ($\mathbf{r}$) whereas Equation 2.21 requires division along the segment axis ($\mathbf{s}$). The trapezoidal rule is then used in both cases. The equations used by the program are 2.22 and 2.23 for the integral equations 2.20 and 2.21, respectively.

$$Q = \sum_{i=1}^{N_x} \frac{1}{2}\Big(\big(q(i+1,j) * q1(i+1,j)\big) + \big(q(i,j) * q1(i,j)\big)\Big) * dr \tag{2.22}$$

$$\phi_\alpha(\mathbf{r}) = \sum_{j=1}^{N_t} \frac{1}{2}\Big(\big(q(i,j+1) * q1(i+1,j+1)\big) + \big(q(i,j) * q1(i,j)\big)\Big) * ds * D \tag{2.23}$$

2.3.1 The Trapezoidal Error

The trapezoidal rule corresponds to approximating the product of the propagators, ($X = q(r,s).q^*(r,s)$) by a straight line on each interval, $dr = \frac{D}{N_x}$. If we look at the Taylor expansion of X, the lowest-order deviation from a straight line is the quadratic $(X")"$ term and this term means that the product deviates from a straight line by at most $\sim \Delta r^2$ within the interval [5].
The corresponding error area is the proportion to $\Delta r^2 \Delta r = \Delta r^3 \sim \frac{1}{N_x^3}$. This is the local error from a single interval. As there are N_x such intervals, the total error should be bounded above by $N_x \frac{1}{N_x^3} = \frac{1}{N_x^2}$. The error of the trapezoidal rule decreases at worst proportional to $\frac{1}{N_x^2}$,

$$\frac{dr^2 D}{12} f"(C_{N_x}) \tag{2.24}$$

The above formula says that the error decreases in a manner that is roughly proportional to dr^2. Thus doubling N_x and halving dr should cause the error to decrease by a factor of ≈ 4.

Although the trapezoidal rule is generally only second-order $\big($the error is $O(dr^2)\big)$, it is highly accurate for periodic functions like the one we are working on.

2.3.2 Applying the Trapezoidal rule

The problems with using the explicit scheme are more obvious in the consequent results of the trapezoidal rule in respect to accuracy.
The solution to the numerical integration of Q should be constant across all time steps. Therefore Q should have the same value throughout the time variable, $\mathbf{s}$, and if we plot the solution of Q against the different time steps we should observe a straight line. However when using the program for space steps $N_x = 15$ and time steps $N_t = 80$ the results are not in the accuracy levels expected, the magnitude of the greatest oscillation is a staggering 120. The propagators $q(r, s)$ and $q^*(r, s)$, as generated by the scheme, are shown in Figures 2.6 and 2.10 and the numerical solution of Equation 2.20 of Q in Figure 2.16.

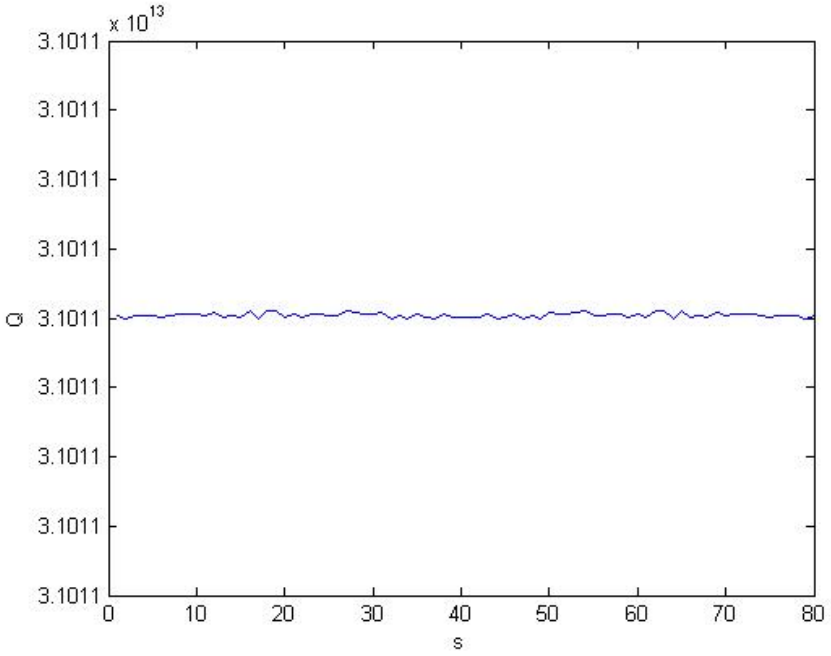


Figure 2.16: Total Partition Function - $N_x = 15$, $N_t = 80$

The segment concentrations that are defined as $\phi_A(\mathbf{r})$ and $\phi_B(\mathbf{r})$ are computed using Equation 2.23. The results of these equations are known and presented in Figure 1.2b. However, due to the small number of points used, the truncation error of the explicit scheme and the trapezoidal error lead us to expect a very weak

approximation of the true result and in fact the numerical solution when plotted in MATLAB is very inaccurate and disapointing, Figure 2.17.

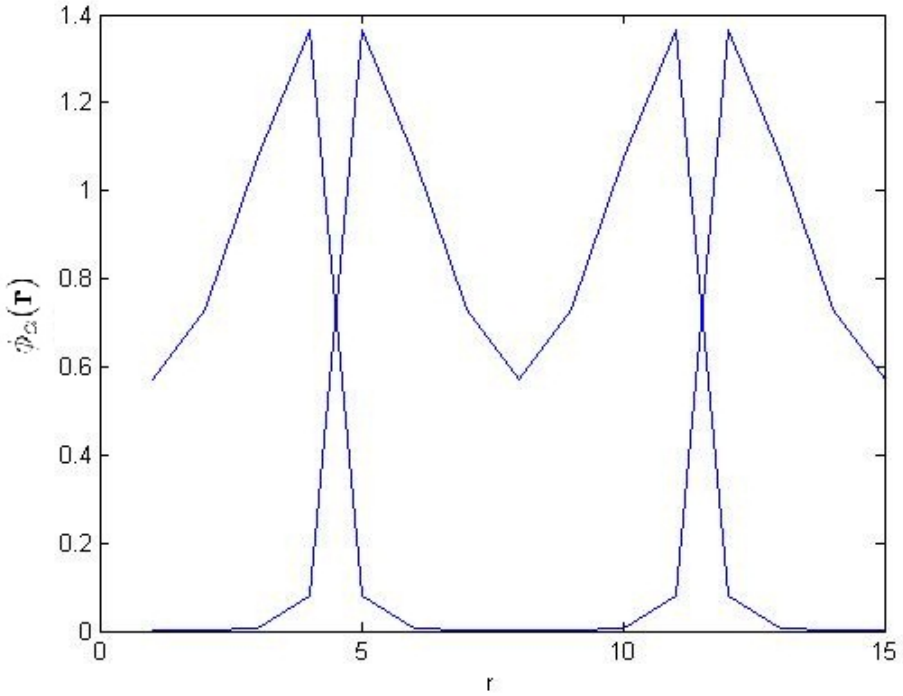


Figure 2.17: Segment Concentration Dist. - $N_x = 15$, $N_t = 80$

Another attempt to improve the accuracy and get more realistic results for the segment concentrations $\phi_\alpha(\mathbf{r})$, is to allocate more space steps, e.g $N_x = 100$ and corresponding time steps to meet the stability condition, $N_t = 3592$.
We should point out that an even greater number of space steps is prefered, however by choosing e.g $N_x = 1000$ we would require $N_t = 365533$ and therefore we would need a stronger processing unit. This problem is another reason why the explicit scheme is not efficient for our problem.
The results for both Q and $\phi_A(\mathbf{r})$, equations 2.22 and 2.23 with $N_x = 100$ and $N_t = 3592$ can be observed in Figure 2.18 and 2.19, accordingly. However we can notice that the concentrations are still lacking accuracy and a small error in the last few digits has caused Q not to be a straight line but in fact include a jump of magnitude 113900032 which is still small compare to the values we are handling. The relative difference is quite small.
All together the explicit scheme is not appropriate for all the reasons mentioned so far, but gives us the background required to use a less unstable scheme and compare

its results.

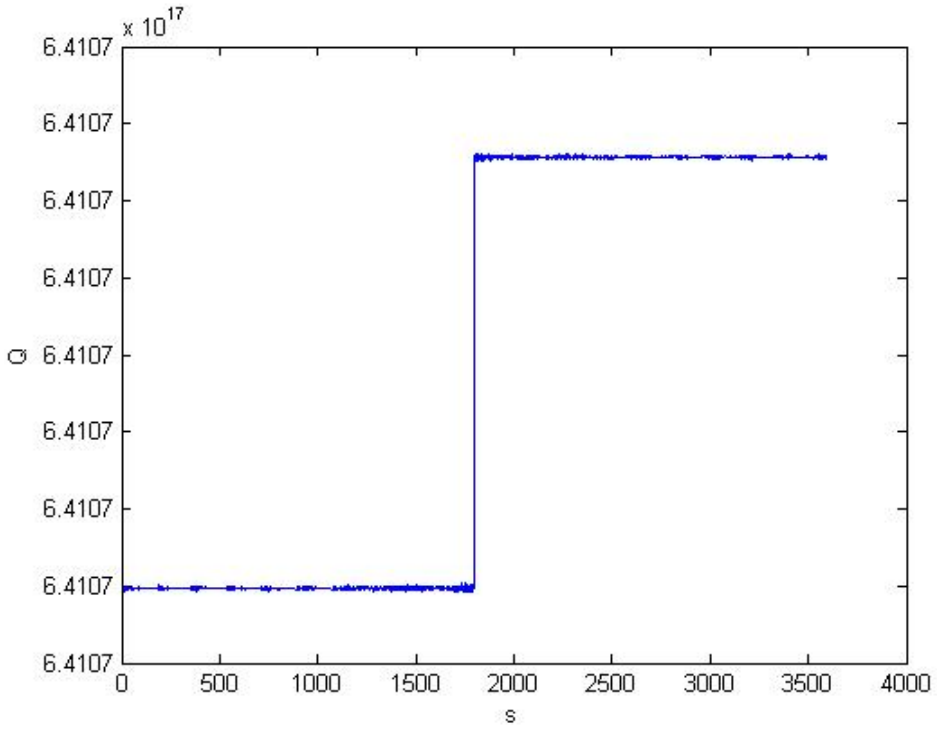


Figure 2.18: Total Partition Function - $N_100 = 100$, $N_t = 3592$

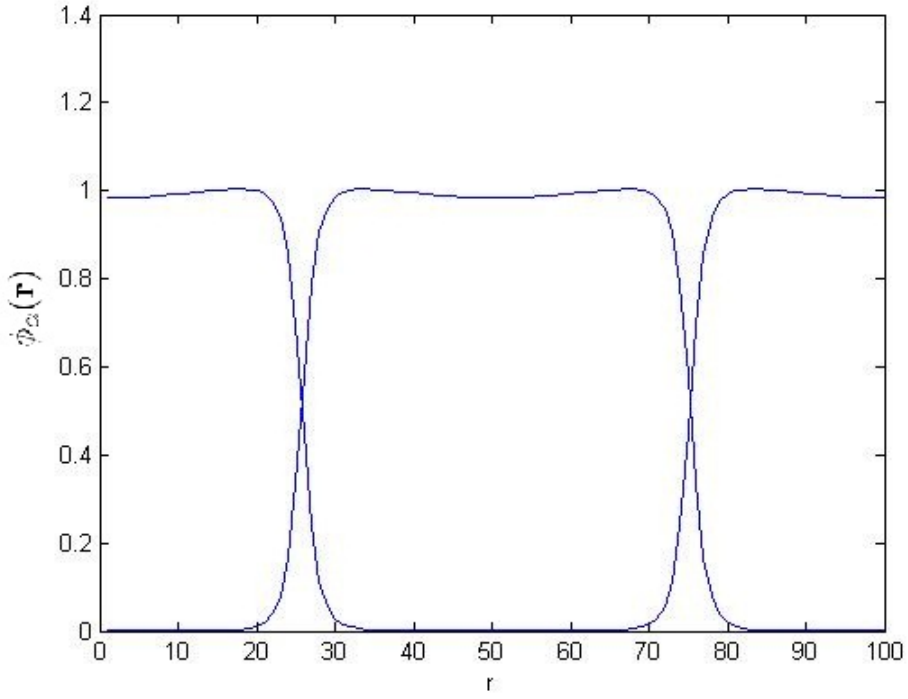


Figure 2.19: Segment Concentration Dist. - $N_100 = 100$, $N_t = 3592$

Chapter 3

The Crank Nicolson Scheme

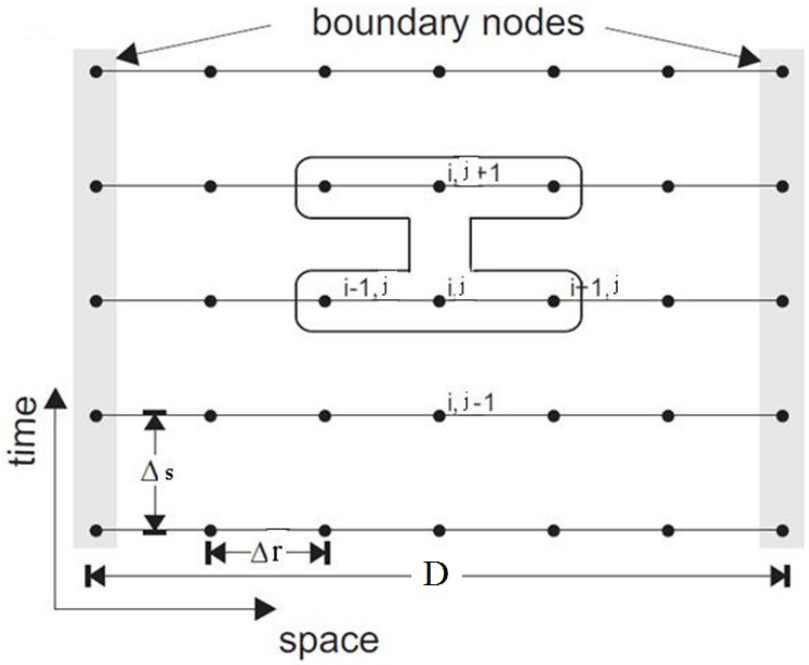


Figure 3.1: [17] Crank Nicolson Grid

The Crank Nicolson scheme is an implicit scheme and was chosen for our problem as it has truncation error $O(\Delta r^2) + O(\Delta s^2)$ which makes it better than other implicit schemes which are less accurate as they have truncation error in time of first order [2]. It is an improvement on the fully implicit scheme as it is an average of the explicit and implicit difference schemes. In addition, unlike the explicit scheme there is no stability condition and therefore we are not bound by any restriction on our choice of N_x and N_t.

$$\frac{\partial q}{\partial s}|_{r_i,s_{j+1}} = \frac{q_{i,j+1} - q_{i,j}}{\Delta s} + O(\Delta s) \tag{3.1}$$

$$\frac{\partial^2 q}{\partial r^2}|_{r_i} = \frac{q_{i-1,j} - 2q_{i,j} + q_{i+1,j}}{\Delta r^2} + O(\Delta r^2) \tag{3.2}$$

Substituting 3.1 and 3.2 into 2.1 and collecting the truncation errors we obtain

$\frac{q_{i,j+1}-q_{i,j}}{\Delta s} = \frac{1}{2}(\frac{1}{6}\frac{q_{i-1,j}-2q_{i,j}+q_{i+1,j}}{\Delta r^2} + \frac{1}{6}\frac{q_{i-1,j+1}-2q_{i,j+1}+q_{i+1,j+1}}{\Delta r^2}) + w_{\alpha,i}\frac{q_{i,j}+q_{i,j+1}}{2} + O(\Delta s^2) + O(\Delta r^2)$

We can notice that the values of q in the above equation from time step j and time step j+1 appear on the right hand side. This equation is used to predict values of q at time j+1 so all values of q at j are assumed to be known. The propagator q is equal to 1 at the first time step as imposed by the initial data.
Rearranging the above equation so that values of q at time j+1 are on the left (L) and values of q at time j are on the right (R) and dropping the truncation error terms we obtain L = R as follows:

$$L = \Big(1.0 + \frac{1}{3}\mu + \frac{\Delta s\ w_{\alpha,i}}{2}\Big)q_{i,j+1} - \frac{1}{6}\mu q_{i-1,j+1} - \frac{1}{6}\mu q_{i+1,j+1} \tag{3.3}$$

$$R = \Big(1.0 - \frac{1}{3}\mu - \frac{\Delta s\ w_{\alpha,i}}{2}\Big)q_{i,j} + \frac{1}{6}\mu q_{i-1,j} + \frac{1}{6}\mu q_{i+1,j} \tag{3.4}$$

where $\mu = \frac{\Delta s}{2\Delta r^2}$.

This equation cannot be rearranged like the explicit equation scheme to obtain a simple algebraic formula for computing for q_i^{j+1} in terms of neighbors like $q_{i+1,j}, q_{i-1,j}$ and $q_{i,j}$.
This equation is one equation in a system of equations for the values of q at the internal nodes of the spatial mesh (i=2,3...,N-1).

The system of equations can be represented in matrix form, the left hand side 3.3 is presented by matrix 3.6. The matrix is tridiagonal and efficient algorithms exist to invert the matrix.
When we perform a von Neumann stability analysis to the scheme by substituting $q_{i,j} = \xi^j e^{ikj\Delta r}$ into the differential scheme, it yields an amplification factor:

$$\xi = \frac{1 - 2m(sin(\frac{k\Delta r}{2}))^2}{1 + 2m(sin(\frac{k\Delta r}{2}))^2} \tag{3.5}$$

where $\mu = \frac{k\Delta s}{(\Delta r)^2}$ and $\xi \leq 1$ for all k.
We can therefore conclude that the Crank Nicolson scheme has unconditional stability just like the fully implicit scheme [13].

The tridiagonal matrix used as part of the Crank Nicolson scheme is,

$$\begin{pmatrix} b_1 & c_1 & 0 & \dots & 0 \\ a_2 & b_2 & c_2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & \dots & 0 & a_N & b_N \end{pmatrix} \tag{3.6}$$

where the coefficients of the interior nodes are:

$a = -\frac{1}{6}\mu$
$b = 1.0 + \frac{1}{3}\mu + \frac{\Delta s\ w_{\alpha,i}}{2}$
$c = -\frac{1}{6}\mu$

Due to the reflective boundary conditions,

a_1 is multiplied by 2
c_{N_x} is also multiplied by 2

In addition, in order to satisfy the reflective boundary conditions on the right hand side, the first space step should be altered from equation 3.4 to 3.7 and for the last space step to equation 3.8.

$$R(1) = \Big(1.0 - \frac{1}{3}\mu - \frac{\Delta s\ w_{\alpha,1}}{2}\Big)q_{1,j+1} + \frac{1}{3}\mu q_{2,j+1} \tag{3.7}$$

$$R(N_x) = \Big(1.0 - \frac{1}{3}\mu - \frac{\Delta s\ w_{\alpha,1}}{2}\Big)q_{1,j+1} + \frac{1}{3}\mu q_{N_{x-1},j+1} \tag{3.8}$$

As for the q^* propagator, the PDE is identical to 2.1 but with the right hand side multiplied by -1. Therefore its discretised form is identical to the one we just

analysed but with opposite signs. However, since the initial data of q^* is at the last time step j=N_t, we need to work backwards to time step j=1 and thus we use the same discretised equations with the same signs, for the right hand side, 3.4, 3.7 and 3.8 and for the left hand side the tridiagonal matrix solver for matrix 3.6.

The external field defined by the function $w_{\alpha,i}$ and equation 2.16 is computed in the same way as explained in the explicit scheme section 2.2.1 . Once more, for propagator q the first half of the time steps use $w_{A,i}$ and the remaining steps $w_{B,i}$ and for propagator q^* vice versa. The plotted results should look similar to Figure 2.4.

We will test run the program of the Crank Nicolson scheme with a uniform mesh to compare the results with that of the explicit scheme. Starting with the propagator q and choosing 15 space steps and 80 time steps, the results should look similar to that of Figure 2.6. The MATLAB plot of the Crank Nicolson results is shown in Figure 3.2 and in fact the result resembles the explicit scheme figure with a small difference which could be attributed to the fact that Crank Nicolson has a truncation error of second order in time whereas the explicit scheme is of first order.

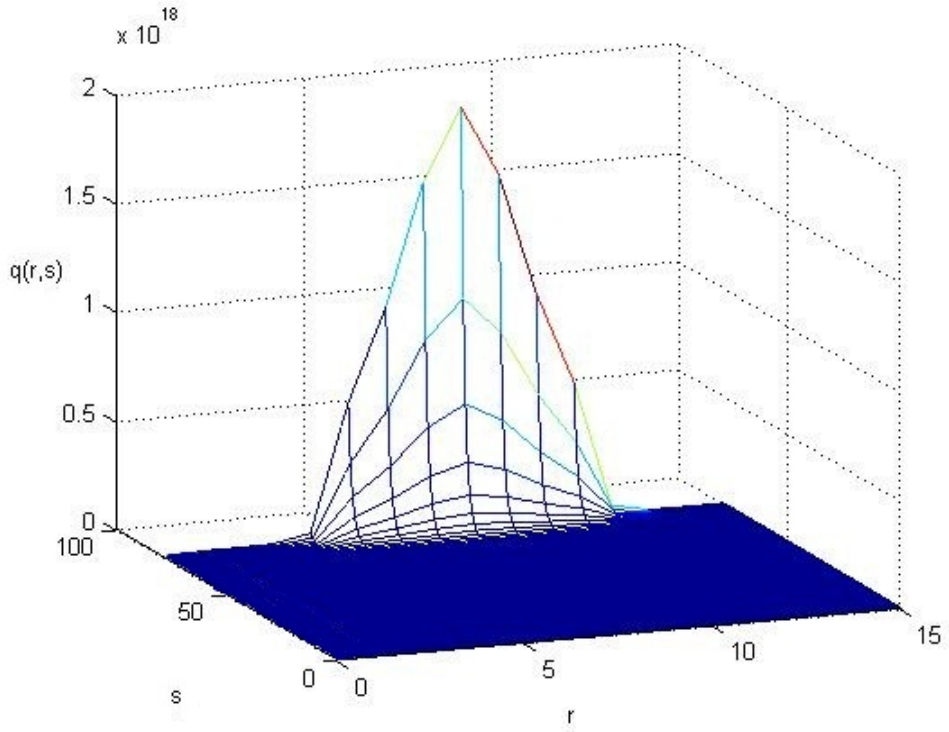


Figure 3.2: q propagator - $N_x = 15$, $N_t = 80$

As with the explicit scheme, we now remove the $w_{\alpha,i}$ function from the discretised

partial differential equation. This would transform our PDE into a pure diffusion equation of which the discretised form is equation 3.9 and the result should look similar to Figure 2.7 of the explicit scheme.

$$\frac{q_{i,j+1} - q_{i,j}}{\Delta s} = \frac{1}{2}(\frac{1}{6}\frac{q_{i-1,j} - 2q_{i,j} + q_{i+1,j}}{\Delta r^2} + \frac{1}{6}\frac{q_{i-1,j+1} - 2q_{i,j+1} + q_{i+1,j+1}}{\Delta r^2}) + O(\Delta s^2) + O(\Delta r^2) \tag{3.9}$$

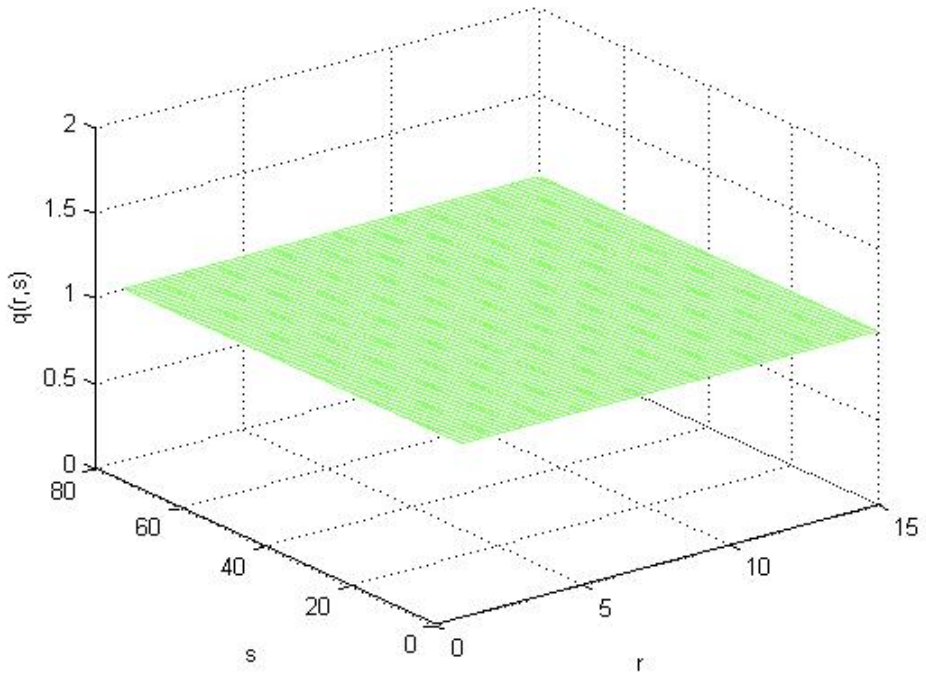


Figure 3.3: q propagator - excluding external fields

We can observe in Figure 3.3 that the scheme outputs a constant value of 1 as the numerical solution of the diffusion equation and thus we can be reassured that the Crank Nicolson scheme works correctly.

Similarly now, for the q^* propagator we choose $N_x = 15$ and $N_t = 90$. The computation should behave in the same manner as for the explicit scheme. We shall start with the initial data at j=N_t and operate in a reverse order to that of q. Our results should look similar to that of Figure 2.10 from the explicit scheme.
As we can see from Figure 3.4, the results are as expected. Better understanding of the difference in accuracy between the numerical results of the explicit scheme and the Crank Nicolson would be made when the total partition function Q and segment concentration $\phi_\alpha(\mathbf{r})$ equations are solved. As a next step we would also like

to validate that the Crank Nicolson scheme works correctly for the computation of q^* by removing the $w_{\alpha,i}$ function and thus solve an ordinary diffusion equation but backwards as the initial data is at the last time step. The solution is 1 throughout the domain and Figure 3.5 presents the results.

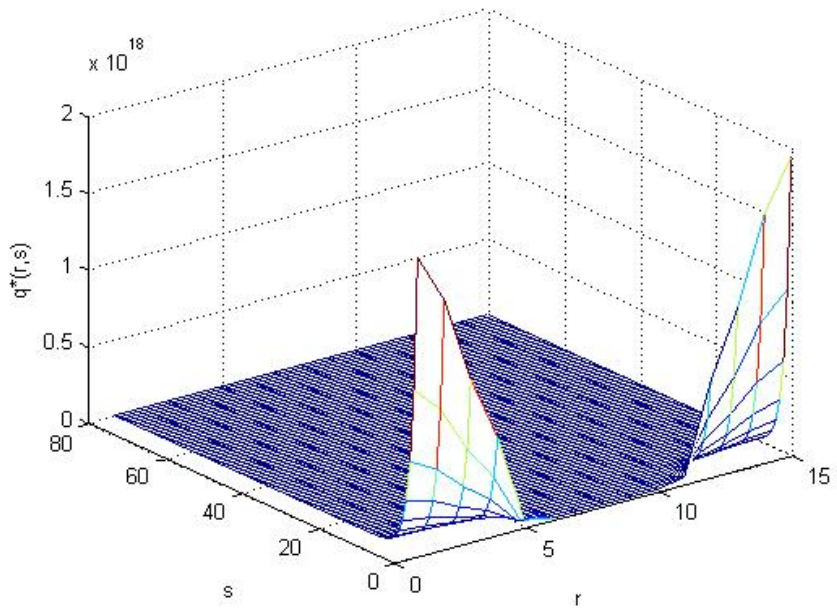


Figure 3.4: q^* propagator - $N_x = 15$, $N_t = 80$

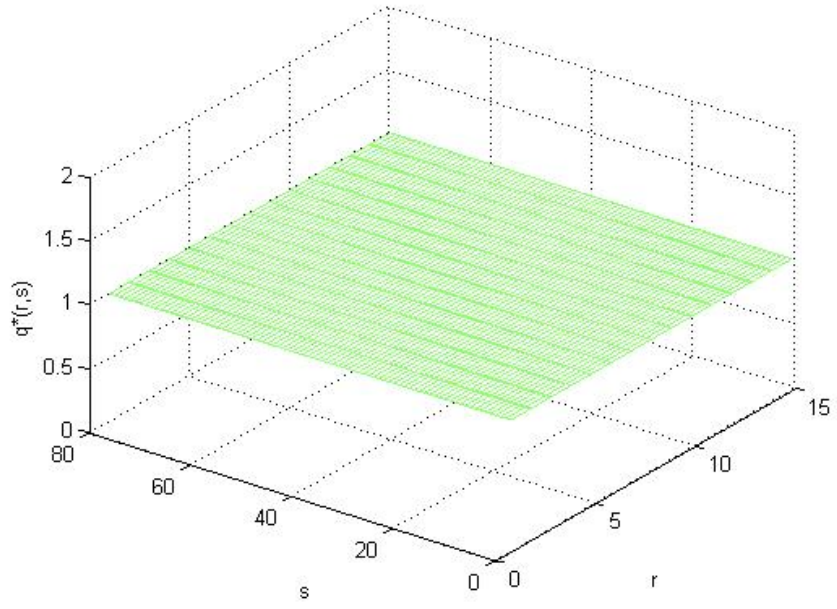


Figure 3.5: q^* propagator - excluding external fields

Unlike the explicit scheme, there is no stability condition that needs to be satisfied, therefore we can choose a larger number of space steps without having to select a huge number of time steps. We now test run the program on a number of space steps that was difficult to run using the explicit scheme. Figures 3.6 and 3.7 show the results of the q and q^* propagators for space steps $N_x = 1000$ and time steps $N_t = 100$.

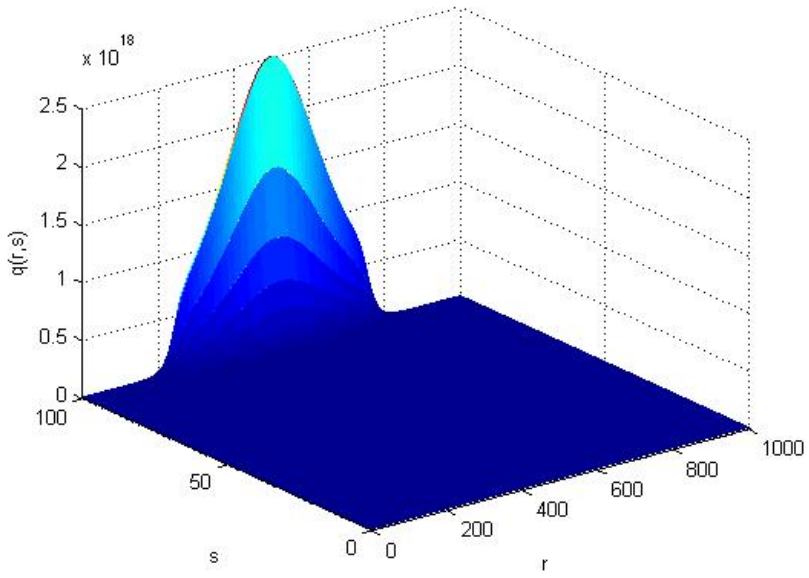


Figure 3.6: q propagator - $N_x = 1000$, $N_t = 100$

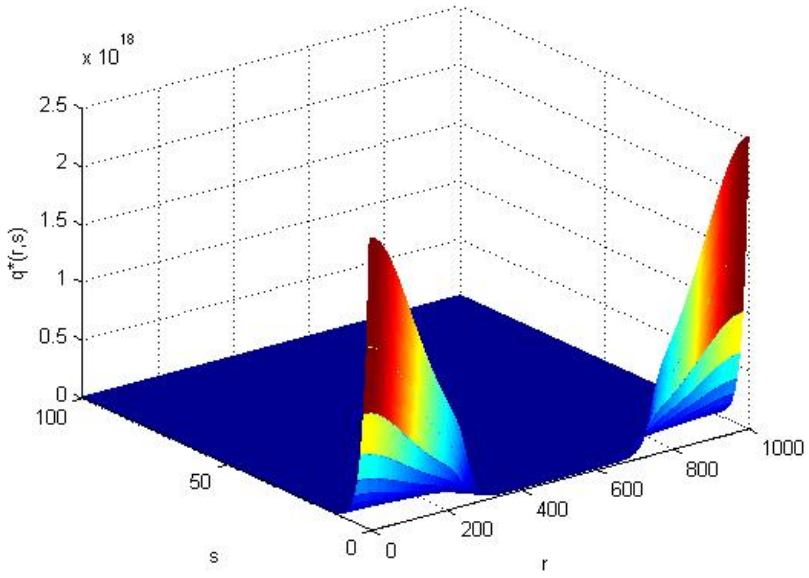


Figure 3.7: q^* propagator - $N_x = 1000$, $N_t = 100$

We repeat the process and select $N_x = 6000$ and $N_t = 1000$. The results of both q and q^* propagators are in Figures 3.8 and 3.9. The increase of resolution is noticeable and we only have to refer to the integral equations of Q and $\phi_\alpha(\mathbf{r})$ to check if the numerical solutions converges to the expected results.

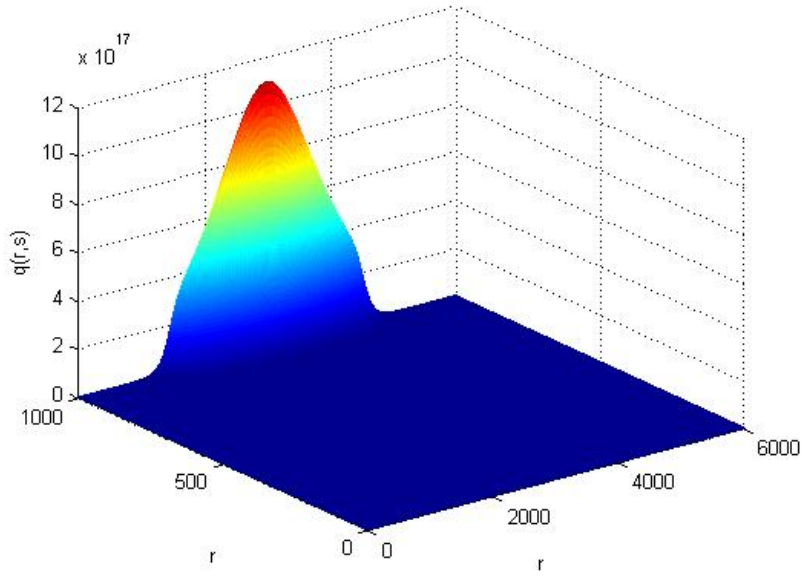


Figure 3.8: q propagator - $N_x = 6000$, $N_t = 1000$

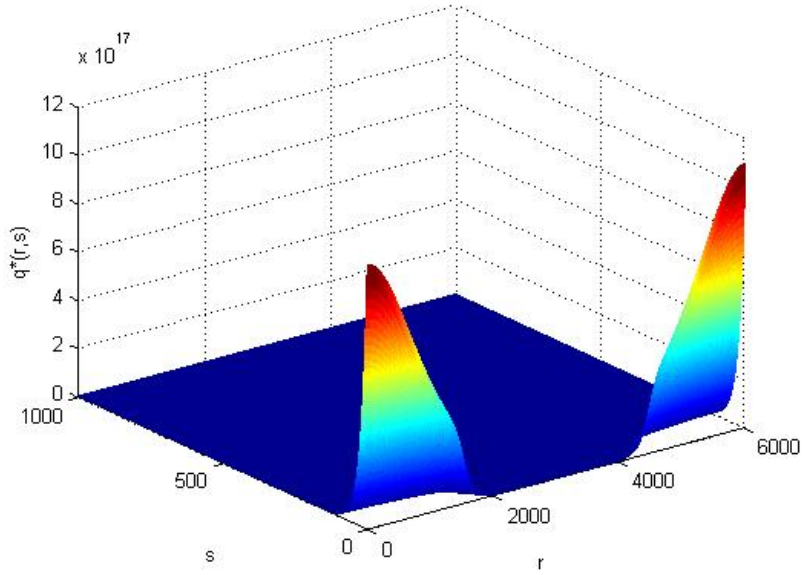


Figure 3.9: q^* propagator - $N_x = 6000$, $N_t = 1000$

Applying the trapezoidal rule to the integral equations for the total partition

function and the segment concentrations, we should observe similar but more accurate results from the explicit scheme. As we explained at an earlier stage, the numerical solution of Q over all time steps (s points) should be constant and therefore we expect a straight line. The segment concentration distributions should resemble the plot in Figure 1.2b.

So far we have shown the results of the copolymer propagators at space steps $N_x = 15$, $N_x = 1000$ and $N_x = 6000$ and the corresponding time steps, $N_t = 80$, $N_t = 100$ and $N_t = 1000$. We will now refer to the numerical solution of the integral equations for each of these results.

Figures 3.10 and 3.11 shows results of the integral equations for $N_x = 15$ and $N_t = 80$. We can clearly notice the greater accuracy of the segment concentrations' results compared to the scheme results in Figure 2.17. But Q still appears to have oscillations and in fact the magnitude of the greatest oscillation is a staggering 4999936. However the values of Q that we are dealing with are much greater and makes this oscillation seem relatively small.

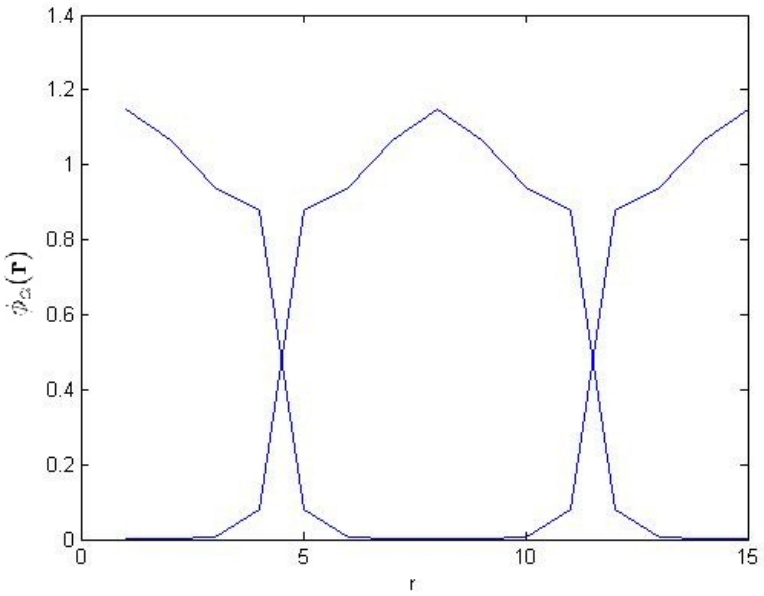


Figure 3.10: Segment Concentration - $N_x = 15$, $N_t = 80$

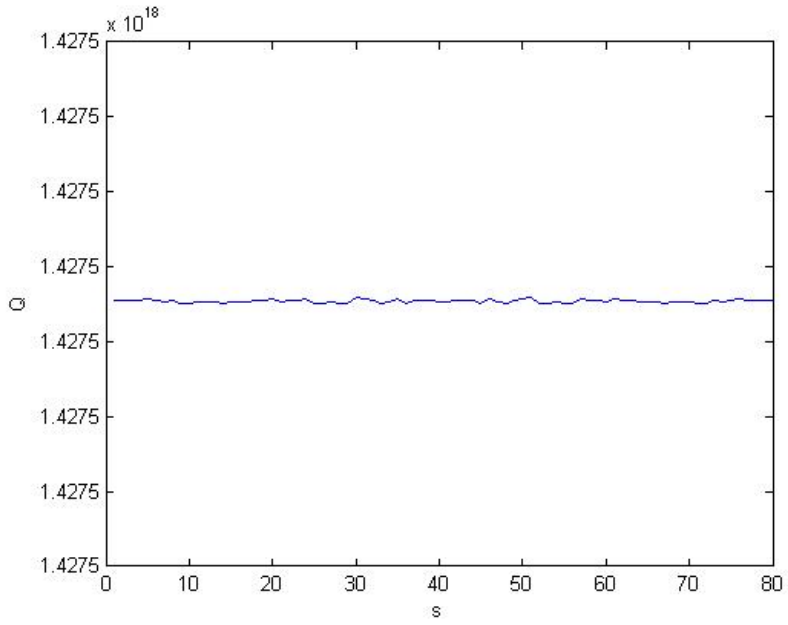


Figure 3.11: Total Partition Function - $N_x = 15$, $N_t = 80$

Further on are the results for $N_x = 1000$ and $N_t = 100$. We can observe an improvement in the $\phi_\alpha(\mathbf{r})$ plot in Figure 3.12, it has become flatter and with the maximum almost at 1 as expected. The Q result has improved as well as there are smaller oscillations. The magnitude of the greatest oscillation is a now 999936. Although the value is still quite large, there is a big decrease in the oscillation magnitudes compared to the previous run using $N_x = 15$ where the greatest oscillation was 4999936.

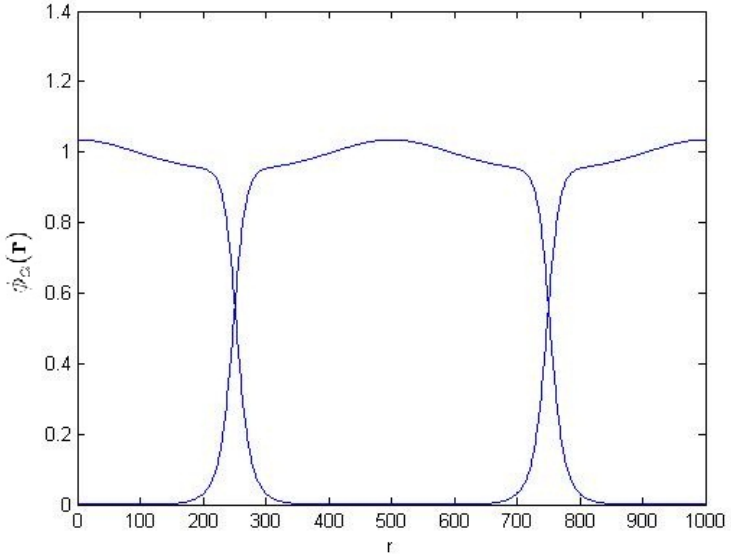


Figure 3.12: Segment Concentration - $N_x = 1000$, $N_t = 100$

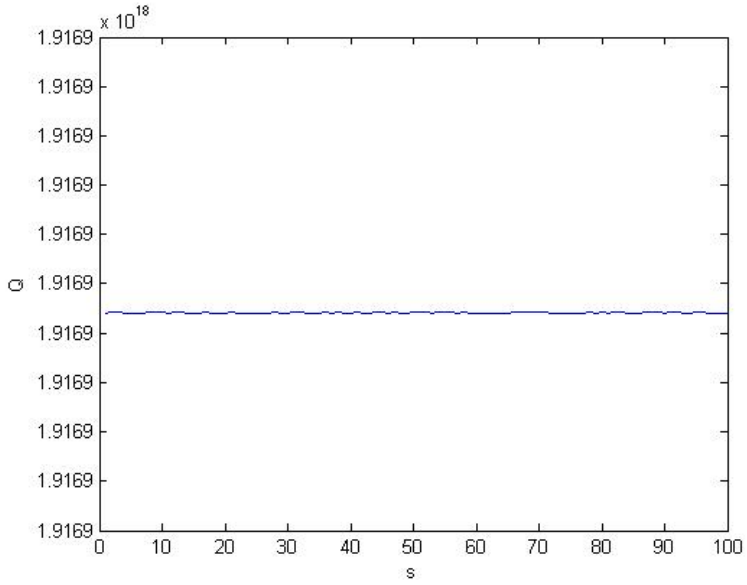


Figure 3.13: Total Partition Function - $N_x = 1000$, $N_t = 100$

Finally, we have the results of $N_x = 6000$ and $N_t = 1000$. This number of points would be impossible using the explicit scheme with the current processing units available. However as we can see Figure 3.14 resembles the expected results shown in Figure 1.2b and Q is almost a straight line. The magnitude of the greatest oscillation has further decreased to 300032. We must note that the values of Q are very large and this difference between them is relatively small.

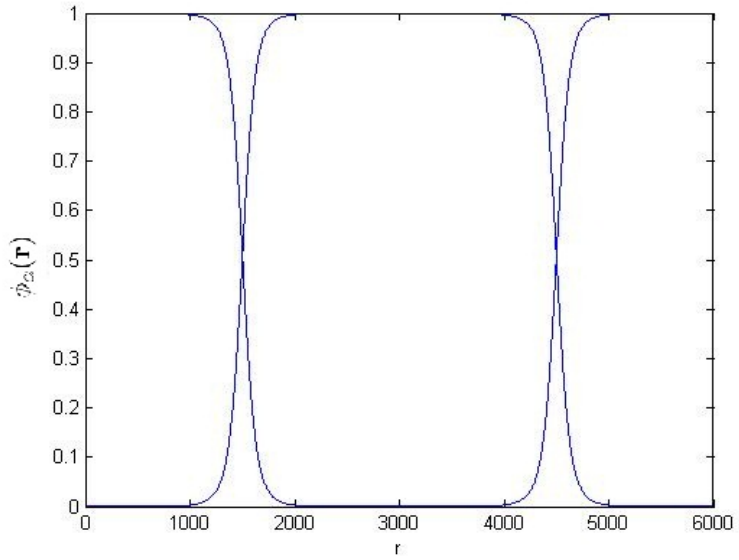


Figure 3.14: Segment Concentration - $N_x = 6000$, $N_t = 1000$

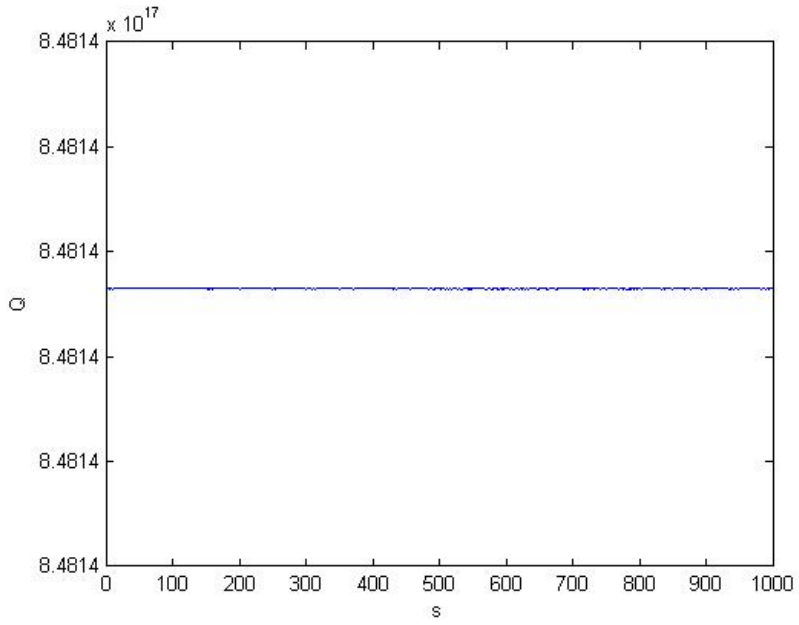


Figure 3.15: Total Partition Function - $N_x = 6000$, $N_t = 1000$

Chapter 4

Grid Refinement

Now that we have created a program that uses the Crank Nicolson scheme and numerically solves our equations on a uniform mesh we would like to increase the level of accuracy at the interface points between the two fields. Having a high level of accuracy at those points is important in order to understand the nature of interactions between the two fields and to manipulate and exploit them in the different industrial and commercial applications. In addition, a uniform grid may be disadvantageous when solutions possess large local gradients. In Chapter 1, we have talked about adaptive methods and specifically about *h refinement*. Using this method we can break the mesh into smaller pieces when necessary and coarsen the mesh where the solution is very smooth if desired [3].

The basic idea of local uniform grid refinement is to cover the spatial domain, D, with nested, finer and finer, locally uniform subgrids so as to accurately resolve steep spatial transitions. This was done to balance the improvement in model accuracy in the area of interest while trying to minimize errors of the refined model and reduce processing time. When very large refinement ratios are used, errors in the model solution in the coarse section might increase. To avoid that phenomenon, the refinement was done in two stages to achieve a cell size suitable for interactions between the two fields for a single molecule. In addition downsizing to an intermediate scale model made the process more computationally efficient [16].
Stage 1 involves recognising the regions that require refining and applying the scheme with a new discretised equation and stage 2 aims to smooth the transition between step size as to avoid big "jumps", but to change step size gradually around the areas of interest. A common approach is the h-refinement, to provide refinement in an area of interest. We achieve this by using a finite-difference grid with variable spacing such that the grid spacing is small where needed and larger away from it.

4.1 Stage 1 - Local grid refinement

The local refinement strategy leads to a new method of discretisation of our partial differential equation,

$$\frac{\partial}{\partial s}q(\mathbf{r},s) = \frac{1}{6}\nabla^2 q(\mathbf{r},s) - w(\mathbf{r},s)q(\mathbf{r},s) \tag{4.1}$$

The non-uniform mesh implies that the space step, dr will not be constant throughout the domain. We therefore have to take into consideration the different sizes of space step at each area of our domain. We can perform this by using the following difference formulas:

$$\frac{\partial q}{\partial s}|_{r_i,s_{j+1}} = \frac{q_{i,j+1} - q_{i,j}}{\Delta s} + O(\Delta s) \tag{4.2}$$

$$\frac{\partial^2 q}{\partial r^2}|_{r_i} = \frac{\frac{q_{i+1,j}-q_{i,j}}{r_{i+1}-r_i} - \frac{q_{i,j}-q_{i-1,j}}{r_i-r_{i-1}}}{r_{i+\frac{1}{2}} - r_{i-\frac{1}{2}}} + O(\Delta r^2) \tag{4.3}$$

using 4.2 and 4.3. Removing the truncation errors we obtain,

$$\frac{q_{i,j+1}-q_{i,j}}{\Delta s} = \frac{1}{12}\frac{\frac{q_{i+1,j}-q_{i,j}}{r_{i+1}-r_i} - \frac{q_{i,j}-q_{i-1,j}}{r_i-r_{i-1}}}{r_{i+\frac{1}{2}}-r_{i-\frac{1}{2}}} + \frac{1}{12}\frac{\frac{q_{i+1,j+1}-q_{i,j+1}}{r_{i+1}-r_i} - \frac{q_{i,j+1}-q_{i-1,j+1}}{r_i-r_{i-1}}}{r_{i+\frac{1}{2}}-r_{i-\frac{1}{2}}} - w_i\frac{q_{i,j}+q_{i,j+1}}{2}$$

At this phase we need to rearrange the above eqation so that the values of q at time j+1 are on the left (L) and values of q at time j are on the right (R) and dropping the truncation error terms we obtain L=R as follows:

$$\begin{aligned} L = &\left(1.0 + \frac{\Delta s}{6}\frac{1}{(r_{i+1}-r_i)(r_{i+1}-r_{i-1})} + \frac{\Delta s}{6}\frac{1}{(r_i-r_{i-1})(r_{i+1}-r_{i-1})} + \frac{\Delta s w_i}{2}\right) q_{i,j+1} \\ &- \frac{\Delta s}{6}\frac{1}{(r_{i+1}-r_i)(r_{i+1}-r_{i-1})}\, q_{i+1,j+1} \\ &- \frac{\Delta s}{6}\frac{1}{(r_i-r_{i-1})(r_{i+1}-r_{i-1})}\, q_{i-1,j+1} \end{aligned} \tag{4.4}$$

$$
\begin{aligned}
R =& \left(1.0 - \frac{\Delta s}{6}\frac{1}{(r_{i+1}-r_i)(r_{i+1}-r_{i-1})} - \frac{\Delta s}{6}\frac{1}{(r_i-r_{i-1})(r_{i+1}-r_{i-1})} - \frac{\Delta s w_i}{2}\right) q_{i,j} \\
&+ \frac{\Delta s}{6}\frac{1}{(r_{i+1}-r_i)(r_{i+1}-r_{i-1})}\, q_{i+1,j} \\
&+ \frac{\Delta s}{6}\frac{1}{(r_i-r_{i-1})(r_{i+1}-r_{i-1})}\, q_{i-1,j}
\end{aligned}
\tag{4.5}
$$

As in the previous discretised equation for the Crank Nicolson under uniform mesh, this equation cannot be rearranged to obtain a simple algebraic formula for computing $q_{i,j+1}$ in terms of neighbours like $q_{i+1,j}$,$q_{i-1,j}$ and $q_{i,j}$. This equation is one equation in a system of equations for the values of q at the internal nodes of the spatial mesh (i=2,3,...N-1).
The system of equations is represented in a matrix form and in fact a tridiagonal matrix, is represented by matrix 4.6,

$$
\begin{pmatrix}
b_1 & c_1 & 0 & \dots & 0 \\
a_2 & b_2 & c_2 & \dots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \dots & a_{N-1} & b_{N-1} & c_{N-1} \\
0 & \dots & 0 & a_N & b_N
\end{pmatrix}
\tag{4.6}
$$

where the coefficients of the interior nodes are:

$a = -\frac{\Delta s}{6}\frac{1}{(r_i - r_{i-1})(r_{i+1}-r_{i-1})}$

$b = 1.0 + \frac{\Delta s}{6}\frac{1}{(r_{i+1}-r_i)(r_{i+1}-r_{i-1})} + \frac{\Delta s}{6}\frac{1}{(r_i-r_{i-1})(r_{i+1}-r_{i-1})} + \frac{\Delta s w_{\alpha,i}}{2}$

$c = -\frac{\Delta s}{6}\frac{1}{(r_{i+1}-r_i)(r_{i+1}-r_{i-1})}$

Due to the reflective boundary conditions,

$a_1$is multiplied by2
$c_{N_x}$is also multiplied by2

Similarly the right hand side as defined by equation 4.5 satisfies the reflecitve boundary conditions by imposing the following equations for the first and final space

step, 4.7 and 4.8 respectively.

$$R(1) = \Big(1.0 - \frac{1}{3}\mu - \frac{\Delta s\ w_{\alpha,1}}{2}\Big)q_{1,j+1} + \frac{1}{3}\mu q_{2,j+1} \tag{4.7}$$

$$R(N_x) = \Big(1.0 - \frac{1}{3}\mu - \frac{\Delta s\ w_{\alpha,1}}{2}\Big)q_{1,j+1} + \frac{1}{3}\mu q_{N_{x-1},j+1} \tag{4.8}$$

where $\mu = \frac{k\Delta s}{(\Delta r)^2}$

As for the q^* propagator the equations are identical but are solved using backward steps, starting from the last space step at $j = N_t$ where the initial data is given and $q^*(r, N_t) = 1$. We need to compute the q^* propagator with great caution as it needs to satisfy condition 4.9, which indicates that the value of any q should be the same as of q^* at the point located half a period further in space and at time step N_{t-s}, where s is the time step of the q.

$$q(r, s) = q^*(r + \frac{D}{2}, 1 - s) \tag{4.9}$$

The external field defined by the function $w_{\alpha,i}$ is computed in the same way, as explained in previous chapters. Once more, for the propagator q the first half of the time steps use w_A and the remaining steps w_B and for the propagator q^* vice versa. Both from the practical and theoretical point of view the first question to address is how to select the regions in D that ought to be refined. Those are the areas of interest and can be distinguished by the points at the interface between the two fields. The fields $w_{\alpha,i}$ are given by equation 2.16 and the results for $N_x = 1000$ and $N_t = 100$ as given by the Crank Nicolson program under the uniform mesh problem are plotted in Figure 4.1.

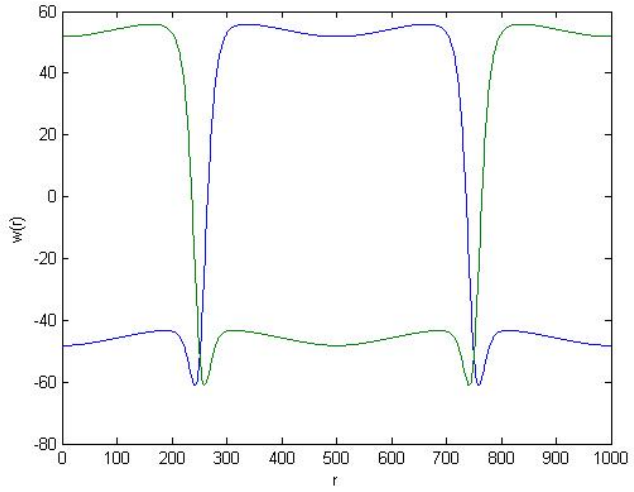


Figure 4.1: External Fields - $N_x = 1000$, $N_t = 100$

As we can see from Figure 4.1 the areas of interest are between $200 \leq r \leq 400$ and $600 \leq r \leq 800$, since those are the regions where the fields intersect. These regions correspond to $N_x = 1000$ steps, we can therefore generalise them for any N_x steps as follows:

- $\frac{N_x}{5} \leq r \leq \frac{2N_x}{5}$
- $\frac{3N_x}{5} \leq r \leq \frac{4N_x}{5}$

Using these inequalities we can specify the areas where we need a finer grid and a coarser grid in the rest of the domain.

The program subdivides the elements in those regions in any even number we choose. As an initial test run we choose to double the points within areas of interest, the space step therefore is divided by a factor of 2, ($\frac{dr}{2}$). Such a grid should be more accurate than a uniform grid calculation. As an example, if we choose $N_x = 1000$ and $N_t = 100$, the total points the adaptive Crank Nicolson program will calculate are $N_x = 1400$ and the results are shown in the feedback table the program produces as shown in Figure 4.2

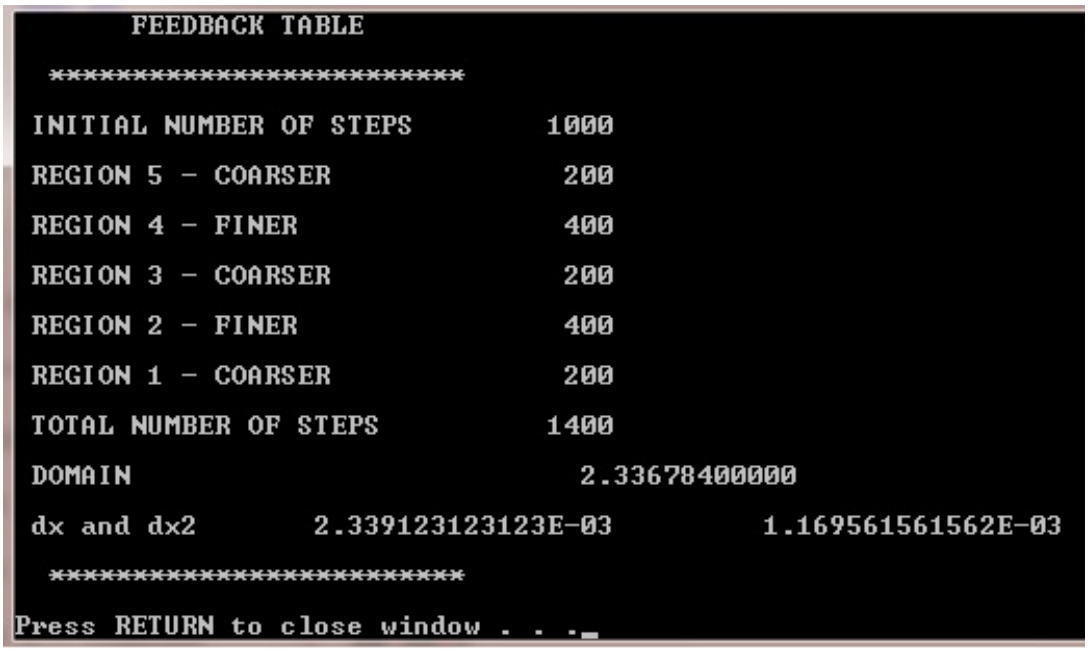


Figure 4.2: Feedback table - Version 6

The propagators q and q^* that are computed from this program are shown in Figures 4.3 and 4.4, respectively.

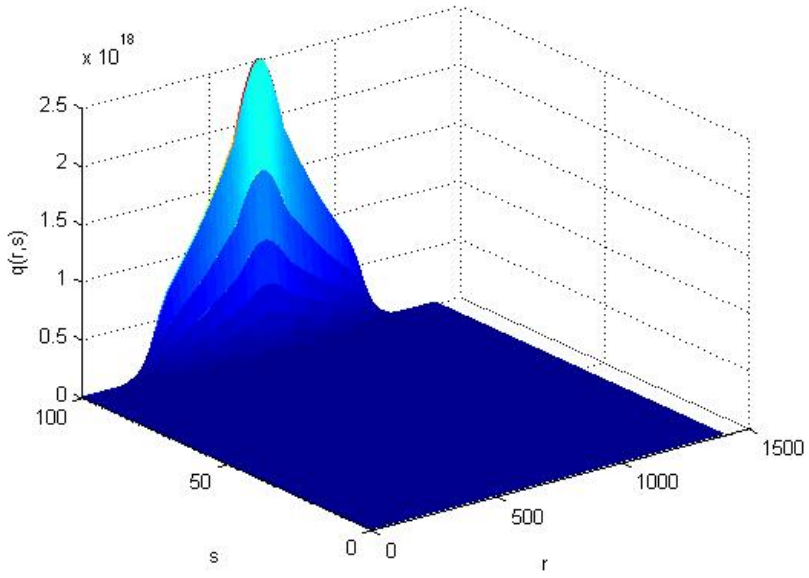


Figure 4.3: q propagator - Version 6 - $N_x = 1400$, $N_t = 100$

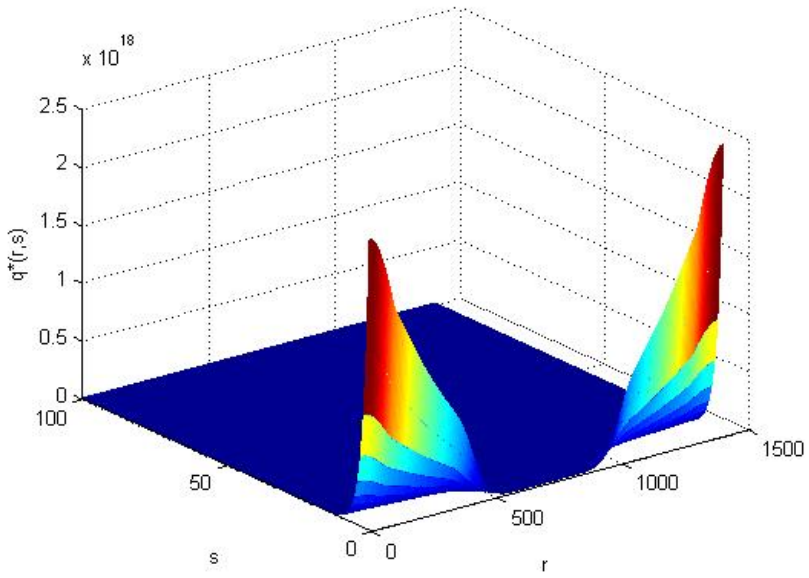


Figure 4.4: q^* propagator - Version 6 - $N_x = 1400$, $N_t = 100$

Finally the trapezoidal rule is used to solve the total partition function Q and the segment concentration distributions $\phi_\alpha(\mathbf{r})$ are presented in Figures 4.5 and 4.6 accordingly.

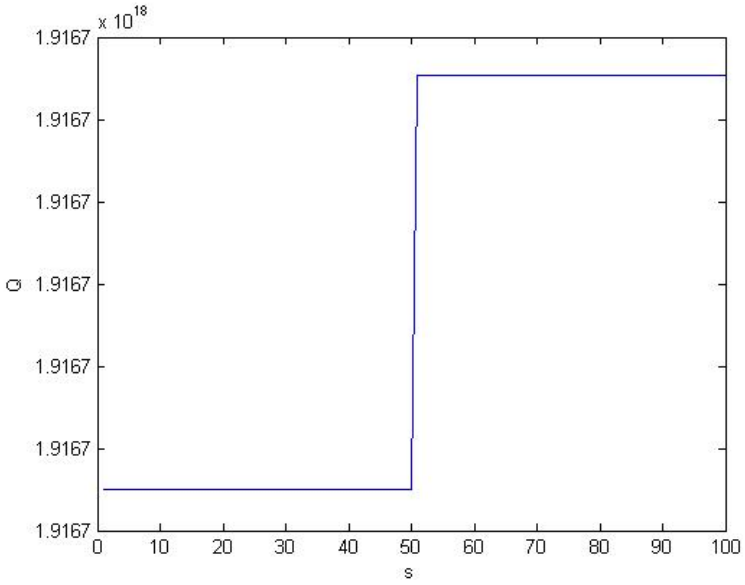


Figure 4.5: Total partition function $N_x = 1400$, $N_t = 100$

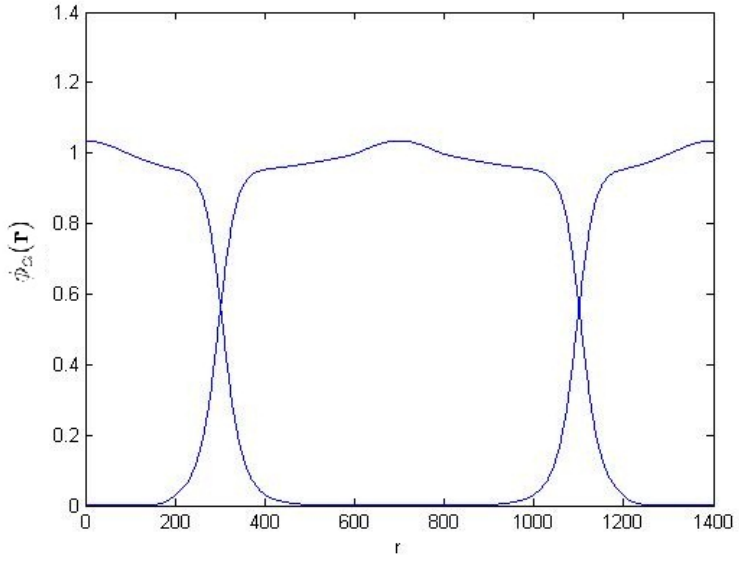


Figure 4.6: Segment concentration Dist. - $N_x = 1400$, $N_t = 100$

The plots look similar if not identical to those of the Crank Nicolson program under a uniform grid in Figures 3.12 and 3.13. The differences will not be obvious until we compare the processing time required to numerically solve the partial differential equations of the propagators and the integral equations, the numerical error using the various schemes and the addition of $\phi_A(\mathbf{r}) + \phi_A(\mathbf{r})$ which should be close if not equal to 1. Q seems to make a jump half way through the steps and the magnitude of the greatest oscillation is quite large, especially with the jump it's

a staggering 2.5130e+011. This might be attributed to the sudden change in size steps from coarser to finer regions.

4.2 Stage 2 - Variable spaced grid

While most schemes can adopt any step size, the question of how to automatically adopt the grid to rapid spatial transitions is much more involved. A common approach to provide more refinement in an area of interest is to use a finite-difference grid with variable spacing. Using a variable spaced grid, the grid spacing is small around the area of interest and gradually increases in size away from the area, out to the boundary of the model domain. This approach reduces the computational time compared to refining the grid over the entire domain (referred to as global refinement). The difference from the program from Stage 1 (which we will refer to as version 6) is that in order to apply the variable spacing (referred to as version 7) we need to introduce two space steps before and after each "jump" from coarser to finer regions. We will call these points *buffer* points and the area they cover as the intermediate regions [12].

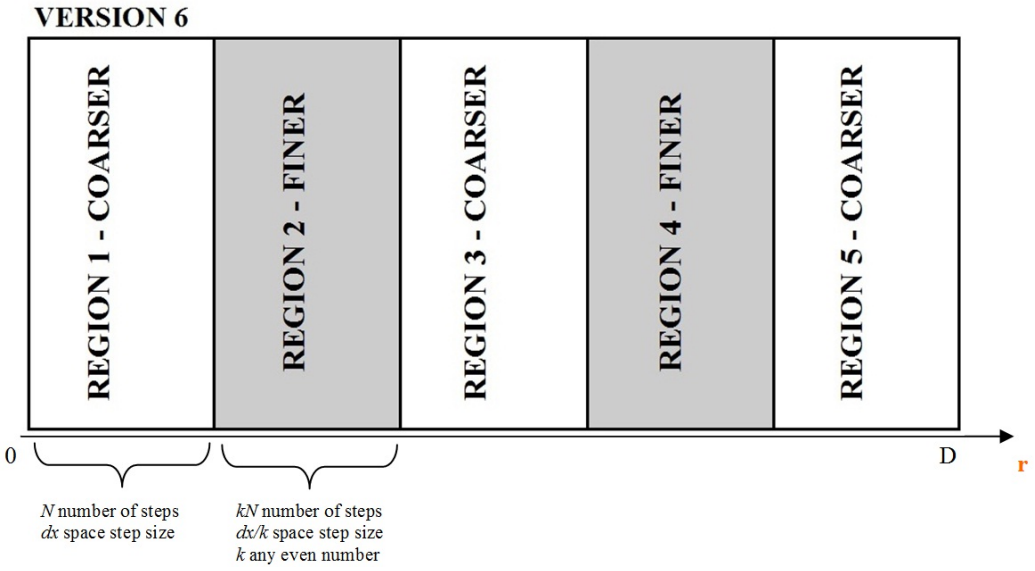


Figure 4.7: Version 6 - Diagram

In Figure 4.7 we can observe the structure of program Version 6 as explained in Stage 1 of this Chapter. It shows the 5 different areas the domain is divided into. The coarse regions will have a number of nodes and space steps depending on the user's preference when the program is run. The user also selects how fine the grid is to become in regions 2 and 4, those are the areas of interest as referred to at earlier

stages. We can select to subdivide the gird by any even factor, obviously the greater the number (k) the finer the grid becomes. As an example if k=2 and there are 200 points in the coarse regions 1, 3 and 5, the fine regions 2 and 4 will have 400 points and step size half ($\frac{dr}{2}$) of that of in the coarse regions. The advantages of this adaptive mesh is that less physical memory will be required for a large number of total space steps, N_x, less processing time and power and therefore less costs imposed compared to having a uniform mesh and using the step size $\frac{dr}{2}$ throughout the whole domain. We will explore and compare the time and accuracy differences between the uniform mesh and different versions of the program in more detail in the next Chapter.

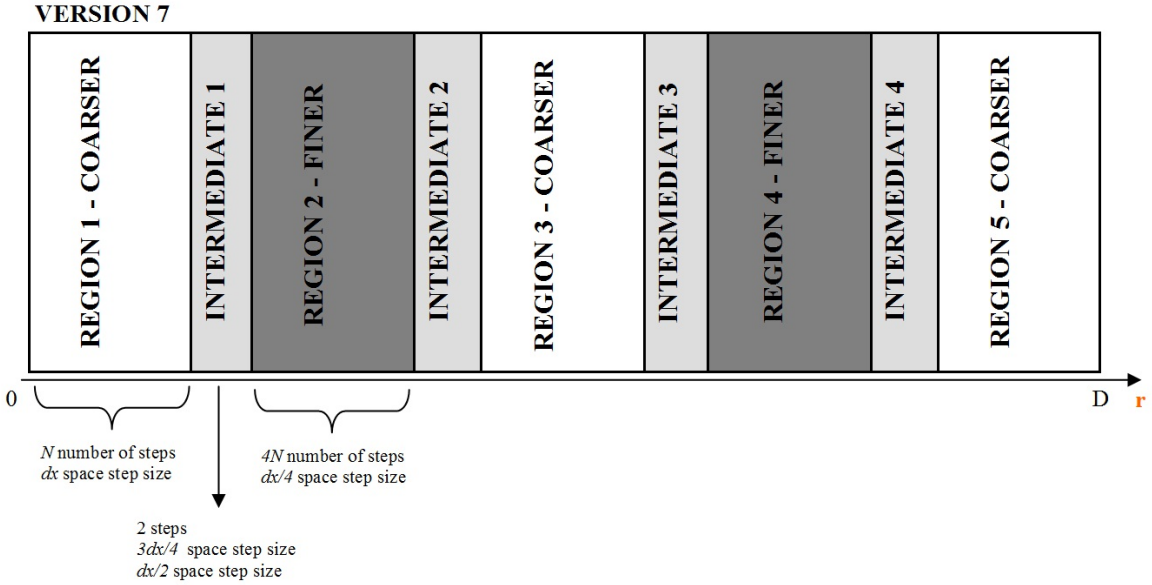


Figure 4.8: Version 7 - Diagram

Program Version 6 is expected to be relatively better than the uniform mesh program, as long as the step size between coarse and fine regions does not differ a lot. The finer the grid in the areas of interest, the greater the accuracy of our results at those points. However, the bigger the "jumps",the greater chances of errors in the calculations. Version 7 of the program tries to tackle this issue by introducing the *intermediate* regions and create a variably spaced grid. Figure 4.8 shows how we tried to employ the variable spaced grid idea in our model. The refined grid uses 3 levels. Level 0 is defined by the user at the start of the program by selecting N_x space steps and thus the step size is created, dr. Level 1 consists of 2 steps, and is termed as the intermediate region which is generated by refining the grid using $\frac{3dr}{4}$ step size and then $\frac{dr}{2}$ for the next step. Similarly Level 2 is the finer region where

the step size is $\frac{dr}{4}$ and the number of steps is 4 times the number in the coarser region. The accuracy of the locally refined calculation depends critically on the initial number of space steps chosen. However, too many points and we will require more processing power and time. We are continuing to investigate the relationship of the grid placement and accuracy.

Version 7 of the program uses the same discretised equations 4.4 and 4.5 as shown in section 4.1. The tridiagonal matrix, matrix2, is used to numerically solve the left hand side 4.4 and the external fields $w_{\alpha,i}$ as explained so far. Boundary conditions are reflective and initial data the same. The q^* propagator uses the same discretised equation but starts from the last time step N_t and works in a backwards procedure. The difference between Version 6 and Version 7 is the extra 2 steps between coarser and finer regions that create the intermediate regions. As a first test run of the program, we choose $N_x = 1001$ and $N_t = 100$ and the results are shown in the feedback table the program produces as in Figure 4.9.

```
 HOW MANY r STEPS?
1001
 HOW MANY s STEPS?
100

 ONE MOMENT PLEASE ...

 STILL PROCESSING

  ************************************************************************
                              FEEDBACK TABLE

  ************************************************************************

 INITIAL NX STEPS              1001
 REGION 5 - COARSER             199
 BUFFER STEP 8 - dx2              1
 BUFFER STEP 7 - dx3              1
 REGION 4 - FINER               798
 BUFFER STEP 6 - dx3              1
 BUFFER STEP 5 - dx2              1
 REGION 3 - COARSER             199
 BUFFER STEP 4 - dx2              1
 BUFFER STEP 3 - dx3              1
 REGION 2 - FINER               798
 BUFFER STEP 2 - dx3              1
 BUFFER STEP 1 - dx2              1
 REGION 1 - COARSER             199
 FINAL Nx STEPS                2201
 DOMAIN                         2.33678400000
 INITIAL SPACE STEP dx        2.336784000000E-03
 dx2 SPACE STEP               1.752588000000E-03
 dx3 SPACE STEP               1.168392000000E-03
 dx4 SPACE STEP               5.841960000000E-04

  ************************************************************************
***Pause:
Enter system command or press ENTER key to restart:
```

Figure 4.9: Feedback Table - Version 7

The propagators q and q^* that are computed from this program are shown in Figures 4.10 and 4.11, respectively.

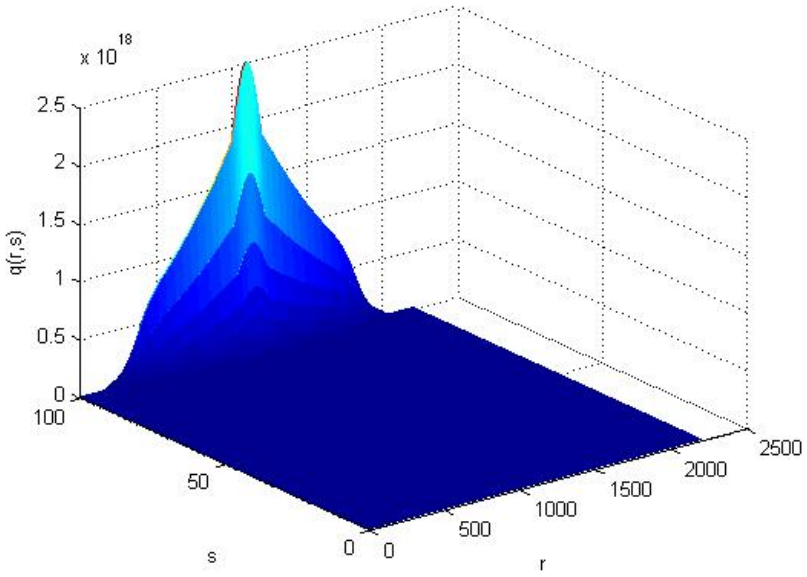


Figure 4.10: q propagator - Version 7

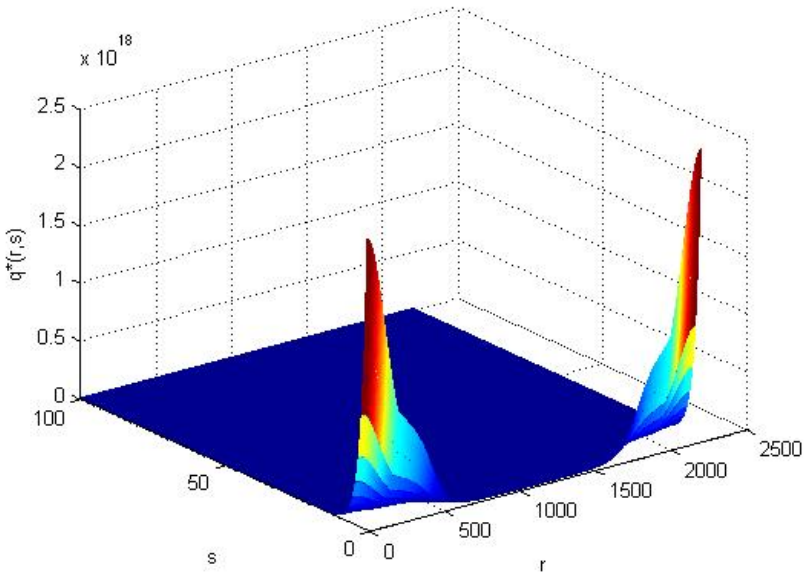


Figure 4.11: q^* propagator - Version 7

We numerically integrate using the trapezoidal rule to find the total partition function Q and the segment concentration distributions $\phi_\alpha(\mathbf{r})$. The solutions are presented in Figures 4.12 and 4.13 accordingly.

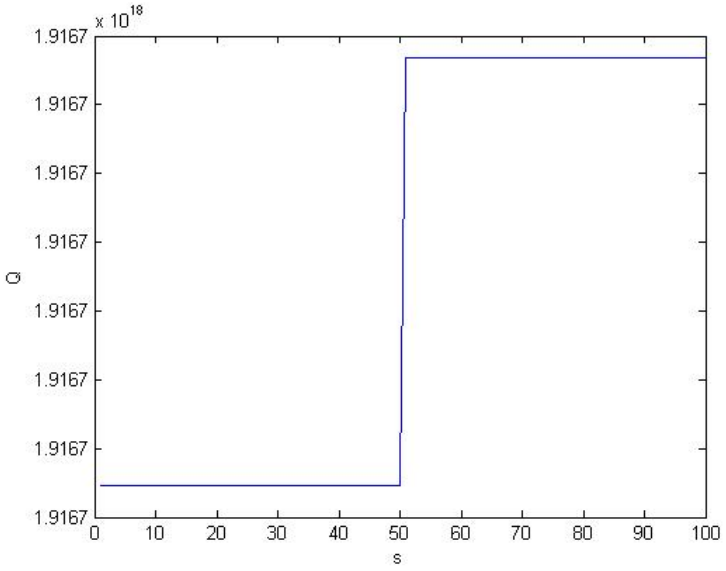


Figure 4.12: Total Partition Function - Version 7

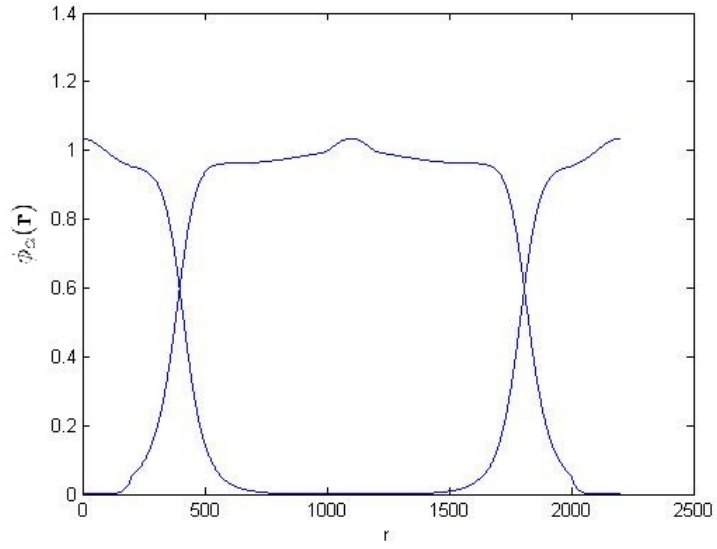


Figure 4.13: Segment Concentration Dist. - Version 7

As in Version 6 of the program, similarly here the plotted results do not differ a lot and we cannot make any conclusions on the efficiency, accuracy and effectiveness of the scheme and program using the variational grid refinement method. The Q plot seems to still have a jump but the magnitude of the greatest oscillation has decreased to 2.500e+011. The more points we use the more accurate this figure will become. In the next chapter we will look at the processing time and accuracy of each program in more detail and try to distinguish the advantages and drawbacks of each one.

Chapter 5

Computational Efficiency

In order for a local refinement to make sense, refined calculations need to show some computational savings. Using a fine grid over the entire domain (referred to as global refinement) can be computationally intensive, both in terms of CPU time and memory requirements. We will be investigating which method is more efficient in terms of accuracy, CPU time and memory requirements. The program using a uniform mesh referred to as Version 5, the program Version 6 that uses a locally refined grid and Version 7 that introduces the concept of a variably spaced grid, were run on the same processing unit, under the same conditions and using a constant total time step, $N_t = 100$ throughout all the tests. Results are shown for refinement ratios of 2, 4 and 8 in Version 6 and compared to the uniform grid. In each case calculations were performed with a refined grid and a uniform grid. Version 7 is still under construction and therefore results can only be compared for the refinement ratio of 4. The uniform grid cell size corresponded with the finest cells in the refined grid. Each calculation was run to the same simulation time and the CPU times were calculated. The CPU times are recorded on the tables below. Our experience has been that the CPU time per iteration does not vary much during a calculation so these timings should be representative.

COMPARISON TABLE using 1000 points as initial steps in Version 6 ... Nt-100

Factor	Version 5 (uniform)				Version 6 (adaptive)				
	dx	Nx	Time	% decrease	dx	dx2	Nx	Time	% dec.
K=2	1.168392E-03	2000	2.854 seconds	0 - original program	2.336784E-03	1.168392E-03	1400	2.387 seconds	16.4

Figure 5.1

As we can see from Table 5.1, by refining the grid at the areas of interest by only a factor of 2, the total CPU time used by the program is decreased by 16.4%. We selected an initial total number of space steps $N_x = 1000$ in Version 6 of the program. The refinement at the areas around the interface of the external fields were double the points at the rest of the domain and the final points became $N_x = 1400$. On the other hand by attempting a global refinement using Version 5, points had to be selected as $N_x = 2000$ so that the size of the space step dr became equal to the step size at the fine regions in Version 6.

Table 5.3 (on the next page) compares all 3 versions of the program. The fine regions of the domain have 4 times the number of points in the coarse regions. However, the difference between Versions 6 and 7 are the extra points. Version 7 creates buffer points in order to establish a gradual refinement of the grid. As we can notice from the table, Versions 6 and 7 required almost the same number of steps, but the latter reduced the time by an extra 3.85%.

We then attempted to increase the total number of space steps but maintaining the refinement factor at $k = 4$. This was done to check if the time is positively correlated to the factor or the number of space steps. However as shown in Table 5.4, although the initial number of points entered in Version 7 increased to 3001, the time reduction was relatively the same as in Table 5.3.

Finally we compared having a fine grid over the whole domain with $N_x = 8000$ steps, using Version 5, to that of an adaptive grid using Version 6 and a factor of refinement $k = 8$. The number of points used by Version 6 was only $N_x = 3800$ and the time was reduced by 44%. This proves how applying the locally refined grid maintains accuracy at the areas of interest and reduces processing time. In addition this reduction of points implies that if a greater number of initial points were used in Version 6, we would not require the processing power that we would using Version 5, therefore costs are reduced as well.

COMPARISON TABLE using 1000 points as initial steps in Version 6 ... Nt=100

Factor	Version 5 (uniform)				Version 6 (adaptive)				
	dx	Nx	Time	% decrease	dx	dx2	Nx	Time	% dec.
K=8	2.923903903 904 E-04	8000	11.294 seconds	0 - original program	2.336784E-03	2.923903903 904 E-04	3800	6.346 seconds	44

Figure 5.2

COMPARISON TABLE using 1001 points as initial steps in Version 7 ... Nt=100 TABLE 5.3

Factor	Version 5 (uniform)				Version 6 (adaptive)					Version 7 (adaptive + buffers)						
	dx	Nx	Time	% decrease	dx	dx2	Nx	Time	% dec.	dx	dx2	dx3	dx4	Nx	Time	% dec.
K=4	5.84196E-04	4000	5.647 seconds	0 - original program	2.336784E-03	5.84196E-04	2200	3.99 seconds	29.3	2.336784E-03	1.752588E-03	1.168392E-03	5.84196E-04	2201	3.775 seconds	33.15

COMPARISON TABLE using 3001 points as initial steps in Version 7 ... Nt=100 TABLE 5.4

Factor	Version 5 (uniform)				Version 6 (adaptive)					Version 7 (adaptive + buffers)						
	dx	Nx	Time	% decrease	dx	dx2	Nx	Time	% dec.	dx	dx2	dx3	dx4	Nx	Time	% dec.
K=4	5.84196E-04	12000	17.083 seconds	0 - original program	2.336784E-03	5.84196E-04	6600	11.388 seconds	33.3	2.336784E-03	1.752588E-03	1.168392E-03	5.84196E-04	6601	11.311 seconds	33.79

Great savings can be made by gradually refining the grid as to the solution approaches the interface points as in Version 7. Variably spaced grids are still being investigated since we have noticed that the more we refine the domain covered by the fine cells, the more time the calculations require. Currently the number of cells at the fine regions, in both Version 6 and Version 7, is rather large compared to the coarser levels. A more accurate analysis of the external fields is required so more precise areas of interest are selected.

5.1 Refinement Errors

Applying the local refinement method, by either using a variably spaced grid or by simply splitting the domain in coarser and finer regions, emphasis is given to specific areas of interest where a finer grid is used. This approach could create errors in calculations of our segment concentration distributions, especially in the parts wher a coarser mesh is used. We therefore want to distinguish the size of the overall error between the results of a globally refined mesh using the uniform mesh program (Version 5) with small size steps and that of a locally refined mesh that has similar size steps only at the finer regions.

As an initial test for error calculations we find the mean sum of $\phi_A(\mathbf{r})$ and $\phi_B(\mathbf{r})$. The incompressibility condition states that $\phi_A(\mathbf{r}) + \phi_B(\mathbf{r}) = 1$. Using numerical schemes such as the Crank Nicolson and the trapezoidal rule lead us to expect a small variance of the solution to this sum. We therefore calculate the mean sum of the segment concentrations across the whole domain using equation 5.1

$$\text{Mean Sum} = \sum_{r=1}^{N_x} \frac{\phi_A(r) + \phi_B(r)}{N_x} \tag{5.1}$$

The results of this sum are shown in the table below, 5.3. N_t remained constant across all programs and equal to 100. The refinement at the finer regions is done using a factor $k = 4$.

Version 5 (uniform)		Version 6 (adaptive)		Version 7 (adaptive + buffers)	
Nx	**Mean**	**Nx**	**Mean**	**Nx**	**Mean**
4004	1.01010730582	2200	1.01273986664	2201	1.02977378573

Figure 5.3: Table

From the table above, we can conclude that the more we refine the grid the greater the error and that can probably be attributed to the coarser areas where less attention is given. However the averages differ only very slightly and the size of the error is not clear and probably not very accurate with this method of measure. We therefore use the error,

$$Error = \sqrt{\frac{1}{D}\int_0^D (\phi_{AB1}(\mathbf{r}) - \phi_{AB2}(\mathbf{r}))^2 dr} \tag{5.2}$$

where $\phi_{AB1}(\mathbf{r})$ is the sum of $\phi_A(\mathbf{r}) + \phi_B(\mathbf{r})$ of the global refinement program which produces the more accurate results throughout the whole domain, since the step size is equal to that of the finer regions in Version 7. $\phi_{AB2}(\mathbf{r})$ is the sum of the segment concentrations as a result of Version 7 that focuses on specific regions of the domain.

The issue with computing the error using equation 5.2 is that each program makes calculations at different points in the domain and the global refinement has a lot more points than the local refinement and we need to use this equation only at the common points. Since we are more interested in the local refinement program, we use its space points as reference where to apply the error equation. However, there might not be an equivalent point in the global refinement program as different size steps are used. We therefore use linear interpolation to find the equivalent $\phi_{AB2}(\mathbf{r})$ at that specific point.

Once we have a $\phi_{AB2}(\mathbf{r})$ equivalent to $\phi_{AB1}(\mathbf{r})$ at each point that the local refinement program produced, we can then apply equation 5.2. The error equation also includes integrating the difference over the two totals over the whole domain. We use the trapezoidal rule to numerically solve this integration in the same way as we did for calculating the total partition function and the segment concentration equations.

We will now compare Version 7, a variably spaced grid that refines the fine re-

gions by a factor of 4 against Version 5, using a globally refined grid that has equal space steps, to that of the fine regions of the locally refined grid. We will select $N_t = 100$ in both programs and the results are shown in Table 5.4

Factor	Version 5 (uniform)		Version 7 (adaptive + buffers)			ERROR
	Nx	dx	Nx	dx	dx4	
4	12004	5.84196E-04	6601	2.336784E-03	5.84196E-04	2.191146698777E-02

Figure 5.4: error table

The results show that the difference between the result of globally refining the grid and using a variably spaced grid is only 2.191146698777E-02. This error is located around the coarser regions of the domain where the locally refined grid program uses bigger space steps.

Table 5.5 performs global refinement on our domain by using smaller and smaller steps. Part 1 of the table increases the total number of space steps from 12000 to 20000 and then 40000 while maintaining the time steps constant. We can observe a negative correlation between processing time and error. As the processing time increases as a consequence of the extra steps and calculations the error decreases. But it's the change in the error that is surprisingly small. The change appears in the seventh significant figure. We therefore perform a different test, as shown in Part 2. Space steps are kept at 6000, while we run the program with 1000, 2000, 4000 and 6000 total time steps. The processing time does increase similarly to Part 1, however the error decreases at each stage to almost half the size of the previous test run. Thus having a sufficient number of total time steps is important. Part 3 shows a change in total time steps but with a smaller number of space steps. The error increases as expected but very slightly. What is even more surprising is that using 6000 total space steps and 1000 time steps does not affect the error as much as using 1000 space steps and 6000 time steps. The error is almost double in the first case. From table 5.5 we can conclude that using 6000 time steps gives us an efficient step size and satisfying results.

Table 5.6 performs a local refinement on our domain. All 3 parts of the table have identical time steps as in the equivalent parts in Table 5.5. However, the space

step size is equal between the two tables at only the fine regions of the domain. The coarser regions in the local refinement process have greater step sizes and are in fact 4 times bigger. We therefore see a big change in the processing time and total space steps required. The error in each test is bigger than the equivalent one using a global refinement but by only a slight variation. The 3 different parts lead us to the same observations and conclusions as in Table 5.5. Therefore, the local refinement method reduces the processing time with a cost of a slight increase in the error but maintains all assumptions and conclusions of the more accurate but more demanding in processing power and memory, global refinement approach.

TABLE 5.5

GLOBAL REFINEMENT USING VERSION 5 (uniform mesh)								
NUMBER OF STEPS		TIME				ERROR (between 1.0 and numerical solution)		
Nx	Nt	q	q*	Integration	Total			
12000	1000	55sec	56sec	45sec	2 min 3sec	4.411909146590 E-03	increase Nx ↓	PART 1
20000	1000	1min 35sec	1min 37sec	1min 18 sec	4min 30sec	4.411722988947 E-03		
40000	1000	5min 10sec	5min 05sec	5min 20sec	15min 35sec	4.411644506548 E-03		
6000	1000	27.6sec	27.9sec	22.5sec	1min 18sec	4.412781177706 E-03	increase Nt ↓	PART 2
6000	2000	55.05sec	55.7sec	45.1sec	1min 45.85sec	2.256139630460 E-03		
6000	4000	- -	- -	- -	- -	1.141328189643 E-03		
6000	6000	4min 30sec	4min 25sec	4min	12min 55sec	7.641263754367 E-04		
1000	6000	27sec	27sec	22sec	1min 16sec	8.050380335667 E-04	increase Nt, smaller Nx ↓	PART 3
1000	10000	45sec	46sec	37sec	2min 8sec	5.012668098887 E-04		

TABLE 5.6

LOCAL REFINEMENT USING VERSION 7 (non-uniform mesh)							
NUMBER OF STEPS		TIME				ERROR (between 1.0 and numerical solution)	
Nx	Nt	q	q*	Integration	Total		
6601	1000	20sec	22sec	12sec	54sec	6.598496944065 E-03	PART 1
11001	1000	41sec	42sec	20sec	1min 43sec	6.598319740119 E-03	
22001	1000	1min 09sec	1min 09sec	35sec	2min 53sec	6.598244678601 E-03	
3301	1000	20sec	20sec	10sec	50sec	6.599321442841 E-03	PART 2
3301	2000	40sec	40sec	20sec	1min 40sec	3.346452611449 E-03	
3301	4000	1min 21sec	1min 21sec	41sec	3min 23sec	1.685533256481 E-03	
3301	6000	- -	- -	- -	- -	1.126681410917 E-03	
551	6000	20sec	20sec	10sec	50sec	1.163793287941 E-03	PART 3
551	10000	34sec	34sec	17sec	1min 25sec	7.152131255992 E-04	

increase Nx

increase Nt

increase Nt smaller Nx

These comparisons show that using a globally refined grid produces more accurate results, which should be more efficient for analysis and applications. However, the error is very small and is attributed to the coarser regions which are not our areas of interest in the domain. The aim of local refinement is to maintain the accuracy at specific areas and thus increase the processing speed. In addition the memory requirements are much less severe for refined grids, provided that the finest levels do not make up a large fraction of the domain. Refering to Table 5.2 the number of total space points used in the globally refined program was a staggering $N_x = 8000$ compared to the locally refined program that only used $N_x = 3800$, that is less than half the points from the uniform mesh program. The important point here is that for each space step, a calculation is required and a value needs to be stored and this process is done for both propagators. We can therefore understand how much physical memory we save by using the refined grid program. Lastly, it is important to mention that at some levels of refinement, it is no longer feasible to compare highly refined grids to uniform equivalent grids because the latter cannot fit within physical memory [15].

Chapter 6

Conclusions and Discussion

Adaptive grid methods are meant for problems possessing rapid local transitions in their solution. By their very nature, these problems remain difficult to solve accurately and cheaply, also when using adaptive grids. The approach we used, of computing on nested, finer and finer, local uniform subgrids, is widely applicable in any number of space dimensions and provides much flexibility in the selection of discretisation schemes.
Accuracy on uniform versus accuracy on non uniform grids, implicit solution costs and no doubt the computer architecture plays an essential role.

While the local refinement approach improves accuracy at selected regions it generally results in introducing extra nodes and thus more computations, this approach can produce finite-difference cells with large aspect ratios, which can lead to numerical errors thus should be used with great caution. This approach demonstrates the shortcomings of traditional finite-difference methods vs finite element methods in that the grid is not flexible. Despite these drawbacks, this method of refinement remains an accurate and viable solution to our problem. A good balance between maintaining satisfactory levels of accuracy and low processing time is therefore essential. Nevertheless, working with these grids is more complex than with uniformly spaced grids but outputs better and more accurate results.

6.1 Further Work

Completing this study, the list of possible alterations we could make is great. The truncation errors of the scheme have not been calculated and their effect on the overall results have been neglected throughout this project. A more accurate method for numerical integration could have been applied rather than the trapezoidal rule.

Version 7 of the program has not been completed to work for any factor of refinement, currently only at $k = 4$. It would be very interesting to compared results of higher orders with the globally refined method but also to check how much time and memory we save compare to Version 6 that does not apply the variably refined grid. Another possible issue is the fact that the complete solution computed over the space-time domain must be kept in storage for error estimation and local refinement purposes. This of course may become an obstacle when the PDE has two or three space dimensions.

Throughout the dissertation project we have seen how to tackle the one-dimensional case, the two- and three- dimensional case is still questionable for applications. The CPU time required to solve those types of problems is certainly of some interest. Solving a higher dimensional problem would allow us to analyse other Classical Microstructures e.g. Spheres and Cylinders as explained in Chapter 1. Attempting to apply the problem on a three-dimensional case could be very challenging and costly.

Bibliography

[1] Hector D. Ceniceros and Glenn H. Fredrickson. Numerical solution of polymer self-consistent field theory. *Multiscale Model. Simul.*, 2(3):452–474, 1993.

[2] Daniel J. Duffy. A critique of the crank nicolson scheme strengths and weaknesses for financial instrument pricing. *WILMOTT magazine*, pages 68–76, 2004.

[3] Joseph E. Flaherty. *Adaptive methods for partial differential equations.* Rensselaer Polytechnic Institute, 1988.

[4] M. Schick G. Gompper. *Soft Matter - Volume 1: Polymer Melts and Mixtures.* Wiley-VCH, 2006.

[5] Ian Gladwell. Numerical integration. *Introduction to Scientific Computing*, 2004.

[6] I. W. Hamle. *Developments in Block Copolymer Science and Technology.* John Wiley and Sons, Lt, 2004.

[7] John Jossey and Anil N. Hirani. Equivalence theorems in numerical analysis:integration, differentiation and integration. 2007.

[8] M W Matsen. The standard gaussian model for block copolymer melts. *Journal of Physics:Condensed Matter*, 14:R21–R47, 2002.

[9] M. W. Matsen and M. Schick. Stable and unstable phases of a diblock copolymer melt. *Physical Review Letters*, 72(16):2660–2663, 1994.

[10] Peter K. Moore. An adaptive h-refinement finite element method for parabolic differential systems in three space dimensions. *SIAM J. Sci. Comput*, 21:1567–1586, 2000.

[11] Gerald W. Recktenwald. Finite-difference approximations to the heat equation recktenwald. *Mechanical engineering (New York, N.Y. 1919)*, 2004.

[12] Mary C. Hill Steffen Mehl and Stanley A. Leake. Comparison of local grid refinement methods for modflow. *Ground Water*, 44(6):792–796, 2006.

[13] Trueman C.W. Sun, C. Unconditionally stable crank-nicolson scheme for solving two-dimensional maxwell's equations. *IET Electronic papers*, 39(7):595–597, 2003.

[14] Jeffrey David Vavasour. Self-consistent mean field theory of the lamellar morphology of binary copolymer-homopolymer blends. January 2000.

[15] J.G. Verwer and R.A. Trompert. Analysis of local uniform grid refinement. *Applied Numerical Mathematics*, 13:251–270, 1993.

[16] Mary Fanett Wheeler William Edward Fitzgibbon. *Computational methods in geosciences*. Society for Industrial and Applied Mathematics, 1992.

[17] ETH Zurich. Explicit versus implicit finite difference schemes, 2009. 4D-Adamello Numerical Modelling shortcourse.