NTNU
Norwegian University of
Science and Technology

Quasiclassical Theory Beyond 1D: Supercurrents and Topological Excitations

Morten Amundsen

MSc in Physics
Submission date: May 2016
Supervisor: Jacob Rune Wüsthoff Linder, IFY

Norwegian University of Science and Technology
Department of Physics

Summary

The finite element method is used to solve the quasiclassical Usadel equation, valid for diffusive superconducting hybrid systems, in higher dimensions than one. A detailed derivation of the method is given, and it is verified that the results generated are correct by comparing with known solutions. In addition, the numerical routine is applied to several case studies which explore novel phenomena that are intrinsically higher dimensional. Specifically, an analysis of a two dimensional Josephson junction with external flux reveals that vortices present in the system are influenced by the width of the junction and the phase difference between the superconductors, altering both the number of vortices and their positions. Spin-orbit coupling is investigated, where it is found that the symmetries in the induced magnetization and the spincurrents can be predicted. In addition, it is found that spincurrents flow in the system even without any charge current present. Finally, a three dimension model is considered, where superconducting islands are placed on a ferromagnetic substrate with different exchange field distributions. It is found that charge current flowing between the superconductors selects the easiest route, rather than the shortest, avoiding regions of highest magnetization.

Sammendrag

Elementmetoden er brukt til å løse den kvasiklassiske Usadellikningen, gyldig for diffusive superledende hybridstrukturer, i høyere dimensjon enn én. En detaljert utledning av metoden er gitt, og det er bekreftet at resultatene den produserer er korrekte ved å sammenligne med kjente løsninger. I tillegg er den numeriske rutinen anvendt på flere eksempelstudier som utforsker nye fenomener som kun oppstår i høyere dimensjoner. I en analyse av en todimensjonal Josephson-sammenkobling med ytre fluks ble det oppdaget at vorteksene som dukker opp i systemet påvirkes av bredden på sammenkoblingen, samt faseforskjellen mellom superlederne. Dette endrer både antallet vortekser og deres posisjoner. Spinn-bane-kobling undersøkes, hvor det vises at symmetrier i den indusert magnetiseringen og spinnstrømmen kan forutses. I tillegg, ble det funnet at det er spinnstrømmer i systemet selv om det ikke er ladningsstrømmer. Til slutt betraktes en tredimensjonal modell, hvor superledende øyer er plassert på et ferromagnetisk substrat med forskjellig romlig fordeling av utvekslingsfeltet. Det ble demonstrert at ladningsstrømmen velger den enkleste ruten mellom superlederne i stedet for den kortest, ved at områdene med høyest magnetisering unngås.

Preface

This thesis represents the conclusion of the two year Master's degree programme in physics at the Norwegian University of Science and Technology. This work has spanned the final two semesters and amounts to 60 ECTS credits. Much effort has been devoted to explaining the theoretical concepts which make the foundation of the quasiclassical equations, and emphasis has been given those topics which I did not understand upon starting this project. The biggest undertaking was, however, the development of a numerical code capable of solving the Usadel equation in 2D and 3D. This includes the creation, not only of a program that solves the differential equations, but also a number of routines for accepting input, enabling the generation of arbitrary model geometries, as well as output of relevant physical quantities. Excerpts of the code is given in appendix B. The numerical method used has to the extent of my knowledge not been used within quasiclassical theory before, and this posed several challenges, for instance in the treatment of boundary conditions, which I have not encountered before. This work has resulted in an article that was published in the journal *Nature Scientific Reports*[1].

I would like to thank my supervisor, professor Jacob Linder, for all his help the past two years. I have truly enjoyed our weekly meetings, from which I learned a great deal. I also want to thank Jabir Ali Ouassou, who went out of his way to help me overcome problems with my numerical code, as well as providing me with useful tips. Finally, I would like to thank my wife, Helene. Without your constant support and encouragement I would certainly not be where I am today. You have always enabled me to follow my dreams, and for that I am forever grateful.

Contents

Chapter 1

Introduction

When a superconductor is placed in contact with a non-superconducting material, a fascinating phenomenon can take place where superconducting properties are transferred to the normal-state material. In this way, a dissipationless current can flow through a non-superconducting material. This is called the superconducting proximity effect, and was first discovered by Meissner and Ochsenfeld[2]. The theoretical study of this phenomenon means solving the quantum transport problem - a Herculean task that requires simplification. Progress is aided by the application of quasiclassical theory, which is an approximation where only the slowly varying envelope functions of the particles are considered, and fast oscillations ignored. This is reasonable because the experimentally relevant systems for the study of the proximity effect are mesoscopic, i.e., with length scales in an intermediate regime between the macroscopic and the microscopic. Diffusive systems coupled to superconductors, are in the quasiclassical approximation governed by the Usadel equation[3]. Its derivation is given in chapter 4.

A large contributor to the recent upswing in research into the proximity effect is the realization that superconductor-ferromagnet hybrid systems can be used within the field of spintroncs to create dissipationless spin transport[4]. One of the mechanisms by which this can take place is if the ferromagnet has an inhomogeneous spatial distribution of the magnetization[5]. This inhomogeneity may come from layers of several ferromagnets with non-collinear magnetization[6], or be intrinsic to one material. Examples of the latter include domain walls, and the more exotic skyrmion, which is a magnetization pattern with a spiral-like character[7,8]. Another mechanism by which spin transport can take place is through spin-orbit interactions[9,10].

The numerical methods in common use today are limited to one dimensional models or require additional approximations, such as linearization of the differential equations[11]. This poses a challenge. While some phenomena of interest can be described by an effective 1D model, they are generally higher dimensional. In addition, the complexity of the systems considered are expected to increase as the frontier of research within superconducting hybrid structures expands. This leads to more sophisticated theoretical hypotheses and experimental setups. A need has therefore arisen for a numerical tool that can handle problems which have so far been outside the researchers' grasp. *Motivated by this, it has* *been the ambition of this thesis to develop an efficient numerical routine capable of solving the Usadel equation, not only in 2D and 3D, but also without any limitations to the geometries that can be modeled.* This is a non-trivial task, and to the best of our knowledge, only a single work has solved the full, unsimplified Usadel equation in 2D[12]. There is no known attempt at a 3D solution. There is, in other words, a vast landscape of physical effects which remain unexplored due to lack of an appropriate numerical method.

Numerical investigations are of great value to experimental research. They provide an inexpensive way of testing variations of a given experiment, thereby helping to focus research efforts. In addition, numerical analyses are often successful in explaining surprising experimental results. In the following, a few recent notable examples are given which may not be described by a 1D numerical model, thus further illustrating the need for a higher dimensional analysis tool. A study by J. Kim *et al* has shown a geometrical dependence of the strength of the proximity effect in an effectively 2D system with superconducting islands on a metallic substrate[13]. It was found that the strength of the proximity effect increases with the curvature of the studied junction. In another study, the effect of an external magnetic field on superconducting correlations was investigated by Roditchev *et al* on a similar structure[14]. It was found that the magnetic field creates vortices in the normal metal, which are select points where the superconducting correlations disappear. The presence of such vortices creates resistance, as their positions change when currents are applied. It turns out that this can to a large extent be mitigated by creating a lattice of perforations that surrounds the vortices, as was shown by Córdoba *et al*[15]. Finally, a phenomenon which is highly relevant for the field of spintronics is the spin Hall effect, which is the accumulation of spin on lateral surfaces when a charge current passes through a material with spin-orbit coupling[16]. This can occur also for superconducting hybrid structures, thus creating a superconducting spin Hall effect[17]. This was recently verified experimentally by Wakamura *et al*[18].

The numerical method proposed herein is the finite element method. The method was first invented in the 1950s for the solution of problems in structural engineering[19], which remains one of the fields in which it is most popular. The main idea behind the finite element method is that the geometry is discretized rather than the differential operators. This means that the numerical framework is established without referencing the particular geometrical shape. It is this feature that makes the method well suited to handle complicated physical models. A detailed derivation of the method, as well as the specifics regarding its application to the Usadel equation is presented in chapter 5.

The degree of discretization necessary in the finite element method, which determines the amount of computing resources required, is governed by the spatial variations of the solution. A slowly varying physical effect can be described by a cruder discretization. This makes the finite element method particularly well suited to solving quasiclassical equations, where the fast oscillations are neglected. The analytical and the numerical approximations therefore complement each other, which results in a highly efficient solution method. Another characteristic which makes the finite element method appealing is its design philosophy. The method requires benchmark solutions to verify the results, and at the same time it has the descriptive power to accurately recreate experiments. This creates a synergy between theory and practice, which should be the basis for any numerical method.

Chapter 2

Background

In this chapter, fundamental concepts will be briefly reviewed.

2.1 An overview of superconductivity

Superconductivity is a fascinating effect which is entirely quantum mechanical in origin, but still observable on scales visible with the naked eye. As was discovered by H. Kamerlingh Onnes in 1911[20], many normal metals drastically change their behavior when cooled to temperatures close to absolute zero. Below a certain temperature threshold T_c, superconductivity can take place, where the resistance in the material drops to zero beyond any means of measure. Indeed, it is estimated that the decay time of currents circulating in superconductors can be of the order $10^{10^{10}}$ years[21]. Another astonishing phenomenon that is associated with superconductivity is the Meissner effect[2], which is the expulsion of magnetic field lines, thereby hindering the penetration of magnetic fields. This is caused by spontaneously induced currents which appear when a metal is cooled to below T_c. This has the effect that magnetic field lines cancel inside the superconducting material, and add outside, leading to perfect diamagnetism[22].

The physical origin of superconductivity stems from the surprising fact that electrons in a lattice can behave attractively. The manner in which this attraction takes place is illustrated in figure 2.1. The electrons can be thought of as moving freely around in the lattice. However, as they do, they attract the oppositely charged ions. This distorts the lattice, as the ions are dragged out of equilibrium, closer to the electrons. Since the ions are several orders of magnitude heavier than the electrons, these distortions linger long after the electrons have passed by, creating pockets where the density of positive charge is greater. These regions can then attract another electron, hence generating an attractive electron-electron interaction.

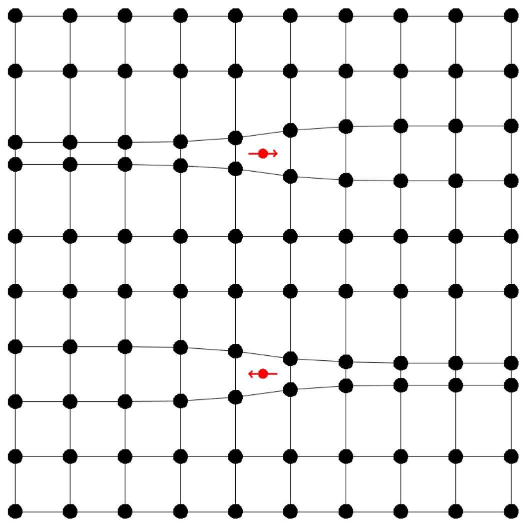


Figure 2.1: Classical illustration of the formation of a Cooper pair. The electrons are marked in red, with arrows indicating their direction of motion.

With an attractive interaction possible, a tendency towards pairing of electrons with opposite spins into what is known as Cooper pairs[23] takes place. However, it is incorrect to say that two particular electrons remain locked in a bound state. Rather, Cooper pairs continually break and reform. For conventional superconductivity, the electron pairs form singlet states, $|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle$.

While the Cooper pairs are not bosons, they gain bosonic mannerisms, which enable them to avoid the Pauli exclusion principle and may therefore all occupy the same state. It turns out that the ground state energy of the Cooper pairs lies below the Fermi surface. As the temperature is cooled below T_c, a phase transition therefore takes place, where Cooper pairs form a fermionic condensate. Much like Bose-Einstein condensates, this allows for the Cooper pairs to attain phase coherence on macroscopic length scales. In addition, there is an energy gap that must be overcome before a Cooper pair can be split, and the electrons scattered. For energies below the gap, there are no available single particle states for the electrons to scatter into, and so they cannot interact with the environment. These mechanisms explain why dissipationless currents that encompass the entire superconductor appear. This picture also explains why superconductivity is a low temperature phenomenon, as thermal vibrations of the lattice would mask the influence of the electrons, and is therefore detrimental to the formation of Cooper pairs.

2.1.1 Characterization of superconductors

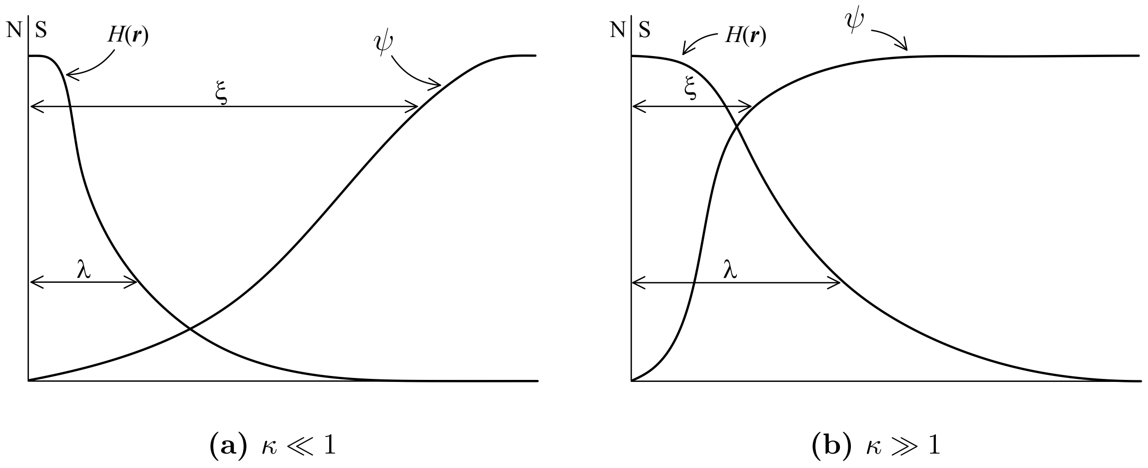
Following the arguments of Pippard[24], the superconducting coherence length ξ is to be found, which is a measure of the length scales over which the superconducting wave functions vary. For an effect taking place at the temperature T_c, only electrons with energies $\sim k_B T_c$ are influenced, where k_B is the Boltzmann constant. This means that the momentum range of the electrons involved can be estimated in terms of the Fermi velocity v_F as $\Delta p \sim \frac{k_B T_c}{v_F}$. By using the Heisenberg uncertainty principle, the spatial uncertainty is found as $\Delta x \leq \frac{\hbar}{2\Delta p}$. Therefore, the superconducting coherence length can be expressed as

$$\xi = \alpha \frac{\hbar v_F}{k_B T_c} \tag{2.1}$$

where α is a numerical constant. The superconducting coherence length can be interpreted as the distance between the electrons in a Cooper pair. Mathematically, it means that a localized change in the wave function will spread out a distance ξ. It is necessary to introduce another length scale, namely the penetration depth λ. When applying a magnetic field H to a superconductor, perfect diamagnetism ensues from the Meissner effect. However, this cancellation of the magnetic field does not happen immediately upon entering the superconductor. Rather, the magnetic field is gradually decreased to zero over a short distance, within which the magnetic field decays as $e^{-r/\lambda}$. From ξ and λ, the Ginzburg-Landau parameter is defined as

$$\kappa = \frac{\lambda}{\xi} \tag{2.2}$$

Equation 2.2 can be used to divide superconductors into two classes. For $\kappa \ll 1$, illustrated in figure 2.2a, the superconducting wave function goes to zero over a distance ξ which is large. This means that there is a greater volume within which Cooper pairs are destroyed, which requires energy. Meanwhile, the larger the penetration depth λ is, the greater is the volume where the magnetic field is not pushed out, thus saving energy. In this case, λ is short, and therefore the net energy contribution to the system from the interface is positive. This means that the energy in the system is minimized by having as few interfaces as possible. Such superconductors are labeled as type I. Type II superconductors correspond to the opposite situation, where $\kappa \gg 1$, as illustrated in figure 2.2b. Here the net energy contribution is negative, so that the superconductor prefers as many interfaces as possible. This materializes in the establishing of an array of vortices, which are localized regions within which the material is in the normal state, and a single flux quantum penetrates[25]. It was found by Ginzburg and Landau that $\kappa = \sqrt{2}$ marks the transition between type I and type II superconductors[26].



(a) $\kappa \ll 1$ **(b)** $\kappa \gg 1$

Figure 2.2: Illustration of the behavior of the magnetic field H and the wave function ψ by a normal metal - superconductor interface for different values of the parameter κ. Adapted from[22].

2.1.2 The proximity effect

Interesting phenomena occur when a superconductor is placed in contact with a metal which is in the normal state. For such a configuration, Cooper pairs may diffuse into the normal state metal so that it attains superconducting properties near the interface. This is called the proximity effect. The leakage of Cooper pairs, in turn, reduces the superconducting correlations near the surface in the superconductor, which is known as the inverse proximity effect. The mechanism by which Cooper pairs enter the normal metal is called Andreev reflection[27,28]. Electrons in the normal metal with energies lower than the superconducting energy gap, may not enter the superconductor, as there are no available states. When such an electron approaches the superconductor, two things can happen. The first is that the electron is reflected at the interface. The other possibility is Andreev reflection, where the electron is transmitted as a Cooper pair, with the reflection of a hole.

In a ferromagnet, the exchange field has a detrimental effect on superconducting correlations. Since the Cooper pair consists of two electrons with opposite spins, the Zeeman effect gives the pair a net momentum $\boldsymbol{q}$. This is illustrated in figure 2.3, where it is seen that the presence of an exchange field has the effect of shifting the bands for electrons with spin up and down with respect to each other. This destroys the phase coherence between Cooper pairs, and therefore also superconductivity. However another exotic phenomenon also takes place. The net momentum $\boldsymbol{q}$ appears as a phase factor in the two-particle wave function. This means that the singlet wave function is changed to[29]

$$|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle \rightarrow |\uparrow\downarrow\rangle\, e^{i\boldsymbol{q}\cdot\boldsymbol{r}} - |\downarrow\uparrow\rangle\, e^{-i\boldsymbol{q}\cdot\boldsymbol{r}} = (|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle)\cos(\boldsymbol{q}\cdot\boldsymbol{r}) + i\,(|\uparrow\downarrow\rangle + |\downarrow\uparrow\rangle)\sin(\boldsymbol{q}\cdot\boldsymbol{r}) \quad (2.3)$$

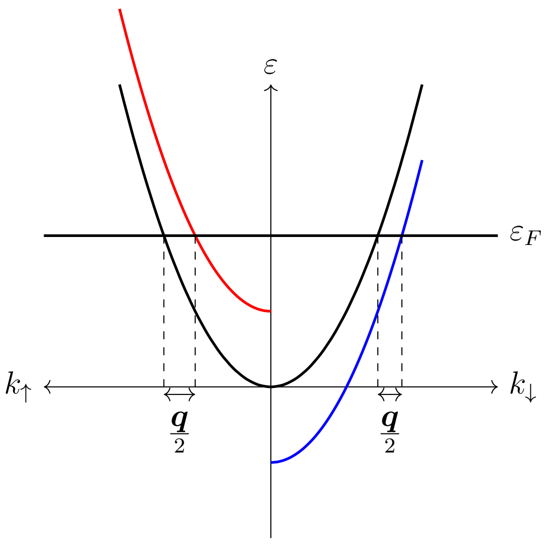


Figure 2.3: Illustration of the splitting of electron bands with opposite spins in the presence of an exchange field.

where $\boldsymbol{r}$ is the position in the perpendicular direction to the interface. This is called a Fulde-Ferrell-Larkin-Ovchinnikov (FFLO) state[30,31]. As can be seen from equation 2.3, upon entering the superconductor, the singlet state Cooper pair begins to oscillate to the spin-zero triplet state. This has exciting implications. Indeed, with the presence of either an inhomogeneous exchange field or spin orbit coupling, the spin-zero triplet state may be rotated to one of the other two triplet states - thus generating equal spin Cooper pairs[5]. Due to the Pauli exclusion principle, this cannot be the normal superconductivity encountered so far. The wave function of the Cooper pair must be odd in exchange of particles, and the spins are now even. For diffusive systems, which is what is of concern here, frequent scatterings results in the wave function being even also in position. The only remaining possibility is that the wave function is odd in time. In other words, an unusual form of superconductivity appears, which is called odd-frequency pairing[32]. This has been recently verified experimentally[33,34].

2.1.3 The Josephson junction

When two different superconductors are joined by a weak link, be it either a normal metal, ferromagnet or isolator, a Josephson junction results[35]. For a normal metal sandwiched between to superconductors, the geometry is shown in figure 2.4.

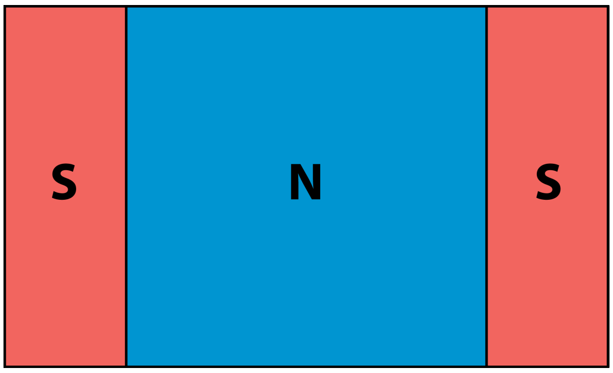


Figure 2.4: Geometry of the Josephson junction.

The Josephson junction has the interesting feature that Cooper pairs can tunnel from one superconductor to the other. If the wave functions residing in the respective superconductors have different phases, supercurrents flow through the weak link - even without the presence of a voltage difference. The supercurrent is given by[22]

$$J = J_c \sin(\gamma) \tag{2.4}$$

where J_c is the maximum supercurrent allowed to flow through the junction and γ is the phase difference. With a magnetic field penetrating perpendicularly to the normal metal present, the situation becomes more complicated. It turns out that the phase difference becomes position dependent, which modifies the expression for the supercurrent to

$$J = J_c \sin\left(\frac{2\pi\Phi}{W\Phi_0}y + \gamma\right) \tag{2.5}$$

where Φ is the applied magnetic flux, W is the width in the transversal direction y and $\Phi_0 = \frac{h}{2e}$ is the flux quantum. It is seen that the supercurrent is zero in a periodic necklace of points in the transversal direction. Around these points, a circulating pattern is found. These points are called Josephson vortices, as they are similar to the Abrikosov vortices found in type II superconductors. The Josephson vortices are, however, of a different nature as they do not have a normal core[22].

2.2 Second quantization

In quantum mechanics, every particle is described as belonging to a state which has associated with it a wave function. A system of several particles is then described by products of these one-particles states. If the particles are identical, they can be divided into two categories; bosons and fermions. The complete, many-body wave function for the two types of particles must be either symmetric or antisymmetric, respectively, and this leads to awkward symmetrization operations, which makes the description of a large number of particles impractical. For this reason, many-particle systems are best described in the language of second quantization. This formalism counts the number of particles in each state, rather than keeping track of each individual particle by single-particle wave functions. Interactions are then described by operators which either create or annihilate particles in a given state. Only fermions, of which the electron is the most famous member, will be of concern in the following. For fermions, only a single particle can occupy any given state. This results in a set of anticommutation relations for the creation operators $c_\nu^\dagger$ and the annihilation operators c_ν:

$$\left\{c_\nu^\dagger, c_{\nu'}\right\} = \delta_{\nu\nu'} \quad \left\{c_\nu, c_{\nu'}\right\} = 0 \quad \left\{c_\nu^\dagger, c_{\nu'}^\dagger\right\} = 0 \tag{2.6}$$

where ν is a set of quantum numbers. In a basis of single-particle position and spin states, $|\boldsymbol{r}, \sigma\rangle$, the creation and annihilation operators are referred to as field operators

$$\psi_\sigma^\dagger(\boldsymbol{r}) = \sum_\nu \langle \nu|\boldsymbol{r}\rangle\, c_{\nu\sigma}^\dagger \quad \psi_\sigma(\boldsymbol{r}) = \sum_\nu \langle \boldsymbol{r}|\nu\rangle\, c_{\nu\sigma} \tag{2.7}$$

By using equation 2.6, the field operators are seen to satisfy the anticommutation relations

$$\left\{\psi_\sigma^\dagger(\boldsymbol{r}), \psi_{\sigma'}(\boldsymbol{r'})\right\} = \delta_{\sigma\sigma'}\delta(\boldsymbol{r}-\boldsymbol{r'}) \quad \left\{\psi_\sigma(\boldsymbol{r}), \psi_{\sigma'}(\boldsymbol{r'})\right\} = 0 \quad \left\{\psi_\sigma^\dagger(\boldsymbol{r}), \psi_{\sigma'}^\dagger(\boldsymbol{r'})\right\} = 0 \tag{2.8}$$

A single-particle operator $\hat{o}$, can be expressed in the second quantization formalism in terms of the field operators as

$$\hat{O} = \sum_\sigma \int d\boldsymbol{r}\psi_\sigma^\dagger(\boldsymbol{r})\hat{o}(\boldsymbol{r})\psi_\sigma(\boldsymbol{r}) \tag{2.9}$$

Similarly, for an operator describing the interaction between two particles, such as the Coulomb interaction, its second quantized equivalent becomes

$$\hat{O} = \sum_{\sigma\sigma'} \int d\boldsymbol{r}d\boldsymbol{r'}\psi_\sigma^\dagger(\boldsymbol{r})\psi_{\sigma'}^\dagger(\boldsymbol{r'})\hat{o}(\boldsymbol{r},\boldsymbol{r'})\psi_{\sigma'}(\boldsymbol{r'})\psi_\sigma(\boldsymbol{r}) \tag{2.10}$$

In the Heisenberg picture, the operators are assumed to depend on time, while the states do not. The time evolution of the field operators is given as

$$\psi_\sigma(\boldsymbol{r}, t) = e^{iHt}\psi_\sigma(\boldsymbol{r}, 0)e^{-iHt} \tag{2.11}$$

The equation of motion that the field operators follow is then given by the Heisenberg equation

$$i\hbar\frac{\partial\psi_\sigma}{\partial t} = [\psi_\sigma, H]\,, \quad i\hbar\frac{\partial\psi_\sigma^\dagger}{\partial t} = \left[\psi_\sigma^\dagger, H\right] \tag{2.12}$$

2.3 Green functions

The Green function is defined as

$$G(\boldsymbol{r}, t, \boldsymbol{r}', t') = -i\langle T\hat{\psi}(\boldsymbol{r}, t)\hat{\psi}^\dagger(\boldsymbol{r}', t')\rangle \tag{2.13}$$

The angle brackets symbolize thermal and quantum average in a certain basis spanned by the eigenstates $|\phi_n\rangle$, and T is the time ordering operator. From this definition it is seen that, if $t > t'$ so that T does not reorder the operators, the Green function is the overlap between the wave function where a particle has been created at point $\boldsymbol{r}'$ at time t' and a particle is destroyed at time $\boldsymbol{r}$ at time t, and the initial wave function. In other words, the Green function gives the correlation between a state where a particle has propagated from $(\boldsymbol{r}', t')$ to $(\boldsymbol{r}, t)$, and the initial state.

The Green function is defined in the Heisenberg picture, so the wave functions $|\phi_n\rangle$ are independent of time. Furthermore, they are unknown, as they are the eigenfunctions of the total Hamiltonian H. Further insight is found by converting to the interaction picture, where both the eigen states and the operators evolve in time. This is done by assuming that the Hamiltonian can be split into two parts, only one of which depends on time; $H(t) = H_0 + V(t)$. The operators then evolve according to the time independent term H_0. The time dependence of the eigenstates is found from the time dependent interaction term $V(t)$. In the interaction picture, a handy tool may be used to describe time evolution, namely the S-matrix $S(t, t')$. It is defined by the relation

$$|\phi_n(t)\rangle = S(t, t')\,|\phi_n(t')\rangle \tag{2.14}$$

Furthermore it is assumed that at $t = -\infty$, the interaction term $V(t)$ has reduced adiabatically to zero, so that Hamiltonian is $H = H_0$ for which the eigenstates $|\phi_n(-\infty)\rangle$ are known. For systems where the interaction term goes to zero also in the limit $t = \infty$, the eigenstates must have returned to their initial, known, states, except for a phase factor i.e.,

$$|\phi_n(\infty)\rangle = S(\infty, -\infty)\,|\phi_n(-\infty)\rangle = e^{i\theta}\,|\phi_n(-\infty)\rangle \tag{2.15}$$

This implies that $\langle S(\infty, -\infty)\rangle_0 = e^{i\theta}$, where the lower index 0 indicates average with respect to the eigenstates $|\phi_n(-\infty)\rangle$.

The field operators in the Heisenberg picture are converted to the interaction picture as

$$\psi^H(\boldsymbol{r},t) = e^{iHt}\psi(\boldsymbol{r},0)e^{-iHt} = e^{iHt}e^{-iH_0t}\psi^I(\boldsymbol{r},t)e^{iH_0t}e^{-iHt} = S(0,t)\psi^I(\boldsymbol{r},t)S(t,0) \quad (2.16)$$

where ψ^H are field operators in the Heisenberg picture, and ψ^I field operators in the interaction picture. The Green function may now be computed by averaging over the known ground state eigenfunctions of H_0, rather than the unknown eigenfunctions of $H(t)$. Inserting equation 2.16 into equation 2.13 gives

$$G(\boldsymbol{r},t,\boldsymbol{r}',t') = -i\langle TS(-\infty,0)S(0,t)\psi(\boldsymbol{r},t)S(t,0)S(0,t')\psi^\dagger(\boldsymbol{r}',t')S(t',0)S(0,-\infty)\rangle_0 \quad (2.17)$$

This may be simplified by using equation 2.15 so that the final expression for the Green function in equilibrium conditions becomes

$$G(\boldsymbol{r},t,\boldsymbol{r}',t') = -i\frac{\langle TS(\infty,-\infty)\psi(\boldsymbol{r},t),\psi^\dagger(\boldsymbol{r}',t')\rangle_0}{\langle TS(\infty,-\infty)\rangle_0} \quad (2.18)$$

It is noted that the S matrices can be moved freely in the numerator of equation 2.18 due to the time ordering operator T. This has been used to collapse the S matrices into one.

2.4 Nonequilibrium Green functions

By looking at equation 2.17 it is seen that the equilibrium Green functions expresses the following: the system starts at $t = -\infty$, defined by the eigenstates of H_0. It is then translated in time to t', when a particle is created, then to t, when a particle is destroyed and finally to $t = \infty$, when the system has returned to its initial state. Thus is found how much the processes at t' and t have altered the initial state, giving the probability amplitude. This approach leans heavily on the assumption that the system returns to its initial state at $t = \infty$, an assumption which is not valid for nonequilibrium phenomena. In such cases, the final states may be drastically altered from the initial states, and are therefore unknown. Rather than computing the Green function along the real time axis, it is instead possible to use a contour as seen in figure 2.5. In equation 2.18 this amounts to replacing $S(\infty,-\infty)$, which translates the system from $-\infty$ to ∞, with an S matrix that first translates the system from $t = -\infty$ to a time τ, $S(\tau,-\infty)$, and an S matrix that translates back again: $S(-\infty,\tau)$. The operators need now be ordered along the contour, so that the time ordering operator T is replaced by the contour ordering operator T_c. The resulting Green function becomes

$$G(\boldsymbol{r},t,\boldsymbol{r}',t') = -i\langle T_cS_c\psi(\boldsymbol{r},t)\psi^\dagger(\boldsymbol{r}',t')\rangle_0 \quad (2.19)$$

where S_c is an S matrix that translates the system from $-\infty$ through t', t, τ and back to $-\infty$.

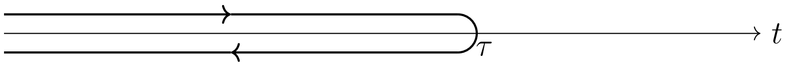
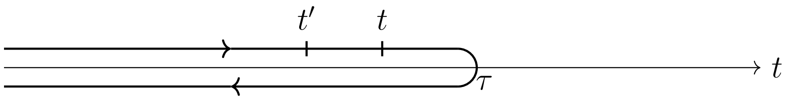


Figure 2.5: Contour used to avoid $t = \infty$. It can be thought of as bending the time axis onto itself at time τ.

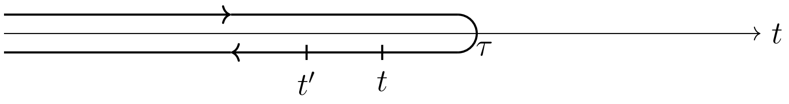
The time τ is arbitrary, and this makes equation 2.19 ambiguous. Depending on where τ is placed relative to t and t', four different versions of the Green function may be extracted:

$$G_t(\boldsymbol{r}, t, \boldsymbol{r}', t') = -i\langle T\psi(\boldsymbol{r}, t)\psi^\dagger(\boldsymbol{r}', t')\rangle \tag{2.20}$$
$$G_{\tilde{t}}(\boldsymbol{r}, t, \boldsymbol{r}', t') = -i\langle \tilde{T}\psi(\boldsymbol{r}, t)\psi^\dagger(\boldsymbol{r}', t')\rangle \tag{2.21}$$
$$G^>(\boldsymbol{r}, t, \boldsymbol{r}', t') = -i\langle \psi(\boldsymbol{r}, t)\psi^\dagger(\boldsymbol{r}', t')\rangle \tag{2.22}$$
$$G^<(\boldsymbol{r}, t, \boldsymbol{r}', t') = i\langle \psi^\dagger(\boldsymbol{r}', t')\psi(\boldsymbol{r}, t)\rangle \tag{2.23}$$

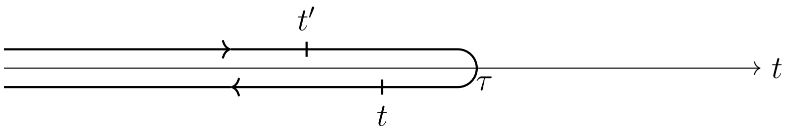
where $\tilde{T}$ is the anti-time ordering operator. In figure 2.6a, both t' and t occur before τ, and so the operators are applied on a part of the contour that is along the time axis. This means that normal time ordering is required, yielding G_t. If t and t' occur after τ, shown in figure 2.6b, the contour is still along the time axis, but oriented in the negative direction so that anti-time ordering is necessary. This gives $G_{\tilde{t}}$. If τ is placed after t', but before t, $G^<$ results. Finally, $G^>$ is found by placing τ after t, but before t'. These configurations are shown in figures 2.6c and 2.6d respectively.



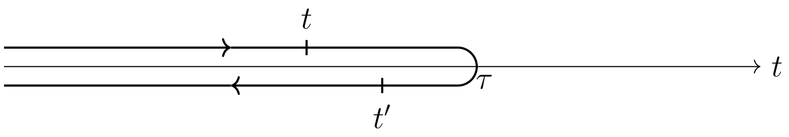
(a) G_t



(b) $G_{\tilde{t}}$



(c) $G^<$



(d) $G^>$

Figure 2.6: The various Green functions that appear due to the deformation of the integration path.

With proper bookkeeping now in place, the artificial parameter τ may be dispatched with by inserting $S(\infty,\tau)S(\tau,\infty) = 1$ into the Green function. This is equivalent to taking the limit $\tau \to \infty$. The resulting contour then consists of two disjoint parts, as illustrated in figure 2.7. This is called the Keldysh contour.

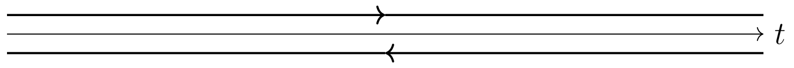


Figure 2.7: The Keldysh contour.

To summarize, the Green function is now defined solely from the initial state, and may be applied to systems where the final states are unknown. The price of this is that it is now necessary to keep track of four Green functions. This may be simplified by introducing matrix notation.

$$G^* = \begin{pmatrix} G_t & G^< \\ G^> & G_{\tilde{t}} \end{pmatrix} \tag{2.24}$$

The Green functions are not linearly independent, and equation 2.24 may be further simplified by performing a linear transformation of the matrix[36], resulting in

$$G = \begin{pmatrix} G^R & G^K \\ 0 & G^A \end{pmatrix} \tag{2.25}$$

where G^R, G^A and G^K are called the retarded, advanced and Keldysh Green functions, respectively. They are defined as

$$G^R(\boldsymbol{r},t,\boldsymbol{r}',t') = -i\theta(t-t')\left\langle\left\{\psi(\boldsymbol{r},t),\psi^\dagger(\boldsymbol{r}',t')\right\}\right\rangle \tag{2.26}$$

$$G^A(\boldsymbol{r},t,\boldsymbol{r}',t') = i\theta(t'-t)\left\langle\left\{\psi(\boldsymbol{r},t),\psi^\dagger(\boldsymbol{r}',t')\right\}\right\rangle \tag{2.27}$$

$$G^K(\boldsymbol{r},t,\boldsymbol{r}',t') = -i\left\langle\left[\psi(\boldsymbol{r},t),\psi^\dagger(\boldsymbol{r}',t')\right]\right\rangle \tag{2.28}$$

2.5 Nambu Formalism

It turns out that for superconducting systems, the diagram perturbation series for the Green function will fail. The reason for this is that such a series expansion is only valid if the perturbed state is qualitatively similar to the unperturbed state[37]. This is not the case for a superconductor, where a phase transition from the normal state has taken place. Cooper pairs may not be arrived at with an expansion of the Green functions introduced so far. To mend this, an extended definition of the Green functions may be used, which includes the field operators for both electrons and holes. This is achieved by introducing vector notation, and is called the Nambu formalism[38]. Including also spin, the field operator in Nambu⊗spin space becomes

$$\psi = \begin{pmatrix} \psi_\uparrow \\ \psi_\downarrow \\ \psi_\uparrow^\dagger \\ \psi_\downarrow^\dagger \end{pmatrix} \tag{2.29}$$

The Green functions now become 4×4 matrices, and are defined as

$$\begin{aligned}
\hat{G}^R(\boldsymbol{r}, t, \boldsymbol{r}', t') &= -i\theta(t-t')\hat{\rho}_3 \left\langle \left\{ \psi(\boldsymbol{r}, t), \psi^\dagger(\boldsymbol{r}', t') \right\}_N \right\rangle && (2.30)\\
\hat{G}^A(\boldsymbol{r}, t, \boldsymbol{r}', t') &= i\theta(t'-t)\hat{\rho}_3 \left\langle \left\{ \psi(\boldsymbol{r}, t), \psi^\dagger(\boldsymbol{r}', t') \right\}_N \right\rangle && (2.31)\\
\hat{G}^K(\boldsymbol{r}, t, \boldsymbol{r}', t') &= -i \left\langle \left[\psi(\boldsymbol{r}, t), \psi^\dagger(\boldsymbol{r}', t') \right]_N \right\rangle && (2.32)
\end{aligned}$$

where the commutation and anticommutation brackets in the Nambu formalism take the form

$$[A, B]_N = AB - (B^T A^T)^T, \quad \{A, B\}_N = AB + (B^T A^T)^T \tag{2.33}$$

Multiplying this out produces the following matrix structures

$$\hat{G}^R = \begin{pmatrix} G^R & F^R \\ \left(F^R\right)^* & \left(G^R\right)^* \end{pmatrix} \tag{2.34}$$

$$\hat{G}^A = \begin{pmatrix} G^A & F^A \\ \left(F^A\right)^* & \left(G^A\right)^* \end{pmatrix} \tag{2.35}$$

$$\hat{G}^K = \begin{pmatrix} G^K & F^K \\ -\left(F^K\right)^* & -\left(G^K\right)^* \end{pmatrix} \tag{2.36}$$

where the submatrices G^X and F^X, $X \in [R, A, K]$ have dimension 2×2 in spin space. The matrices F^X are called anomalous Green functions, and their spin space components are given as

$$\begin{aligned}
F^R_{\sigma\sigma'}(\boldsymbol{r}, t, \boldsymbol{r}', t') &= -i\Theta\left(t-t'\right) \left\langle \left\{ \psi_\sigma(\boldsymbol{r}, t), \psi_{\sigma'}(\boldsymbol{r}', t') \right\} \right\rangle && (2.37)\\
F^A_{\sigma\sigma'}(\boldsymbol{r}, t, \boldsymbol{r}', t') &= i\Theta\left(t'-t\right) \left\langle \left\{ \psi_\sigma(\boldsymbol{r}, t), \psi_{\sigma'}(\boldsymbol{r}', t') \right\} \right\rangle && (2.38)\\
F^K_{\sigma\sigma'}(\boldsymbol{r}, t, \boldsymbol{r}', t') &= -i \left\langle \left[\psi_\sigma(\boldsymbol{r}, t), \psi_{\sigma'}(\boldsymbol{r}', t') \right] \right\rangle && (2.39)
\end{aligned}$$

It is seen that the anomalous Green functions describe correlations between pairs of electrons. The off-diagonal elements of these matrices, which have opposite spins, describe conventional Cooper pairs. The complete Green function in Keldysh⊗Nambu⊗spin space becomes an 8×8 matrix, and is given as

$$\check{G} = \begin{pmatrix} \hat{G}^R & \hat{G}^K \\ 0 & \hat{G}^A \end{pmatrix} \tag{2.40}$$

By insertion, it is seen that the advanced Green function may be expressed in terms of the retarded Green function as

$$\hat{G}^A = -\hat{\rho}_3 \left[\hat{G}^R\right]^\dagger \hat{\rho}_3 \tag{2.41}$$

In addition, when the system considered is in thermal equilibrium, the Keldysh Green function is given as[39]

$$\hat{G}^K = \left(\hat{G}^R - \hat{G}^A\right) \tanh\left(\frac{\varepsilon}{2k_B T}\right) \tag{2.42}$$

where ε is the energy in the system, and T is the temperature. These identities imply that only the retarded Green function needs to be computed to find the entire matrix $\check{G}$ in equilibrium.

Chapter 3

Microscopic theory

In this chapter, the equations of motion that describe superconducting hybrid structures will be derived.

3.1 Superconductivity

Conventional superconductivity is in the following derived from the electron-phonon interaction, thereby motivating the use of the BCS interaction.

3.1.1 Electron-phonon interaction

In a superconductor, pairs of electrons may interact attractively and form so-called Cooper pairs. To understand this counterintuitive fact, it is necessary to explore the interaction between electrons and phonons. For this purpose, a lattice of ions is considered. A particular ion, number j, has its equilibrium position at $\boldsymbol{R}_j^{(0)}$. It may oscillate about this position, and the deviation from equilibrium is called $\boldsymbol{u}_j(t)$. The instantaneous position of ion j at time t then becomes $\boldsymbol{R}_j(t) = \boldsymbol{R}_j^{(0)} + \boldsymbol{u}_j(t)$. The Hamiltonian describing the dynamics of electrons and phonons is[40]

$$H = \sum_{k\lambda} \omega_{k\lambda} a_{k\lambda}^{\dagger} a_{k\lambda} + \sum_i \left[\frac{p_i^2}{2m} + \frac{e^2}{2} \sum_{j \neq i} \frac{1}{r_{ij}} \right] + H_{el-ion} \tag{3.1}$$

where the first term describes the contribution of free phonons, where $a_{k\lambda}^{\dagger}$ and $a_{k\lambda}$ are respectively creation and annihilation operators for a phonon with wavevector $\boldsymbol{k}$, polarization λ and frequency $\omega_{k\lambda}$. The second term is the Coulomb interaction between electrons, and the final term is the interaction between electrons and the ions, which is some unknown potential depending upon the distance between the electrons and the ions:

$$H_{el-ion} = \sum_{ij} V(\boldsymbol{r}_i - \boldsymbol{R}_j) \tag{3.2}$$

Under the assumption that the deviations from equilibrium, $\boldsymbol{u}$, are small, equation 3.2 may be expanded about the equilibrium positions $\boldsymbol{R}_j^{(0)}$ so that

$$H_{el-ion} = \sum_{ij} \left[V(\boldsymbol{r}_i - \boldsymbol{R}_j^{(0)}) - \boldsymbol{u}_j \cdot \nabla V(\boldsymbol{r}_i - \boldsymbol{R}_j^{(0)}) \right] + O(u^2) \tag{3.3}$$

The first term of equation 3.3 includes the effects of the periodic lattice on the electron and are described by Bloch states. The second term includes a coupling to the motion of the ions, and so this is the electron-phonon interaction.

The motion of the ions are oscillatory, and so it is reasonable to expand $\boldsymbol{u}(t)$ in a basis of the eigen states of the non-interacting phonon system, which are normal coordinates. By assuming that the system is a large box of volume V with periodic boundary conditions, this can be written in wave vector space as[40,41]

$$\boldsymbol{u}_j(t) = \sum_{k\lambda} \left(\frac{\hbar}{2MN\omega_{k\lambda}} \right)^{1/2} \boldsymbol{\xi}_{k\lambda} (a_{k\lambda} e^{-i\omega_{k\lambda} t} + a^\dagger_{-k\lambda} e^{i\omega_{k\lambda} t}) e^{i\boldsymbol{k}\cdot\boldsymbol{R}_j^0} \tag{3.4}$$

where M and N is the mass and number of ions respectively. For each wave vector k within the first Brillouin zone and each polarization λ, the phonon creation and annihilation operators are given as $a^\dagger_{k\lambda}$ and $a_{k\lambda}$ respectively. The vector $\boldsymbol{\xi}_{k\lambda}$ denotes the polarization direction of the given phonon mode. By requiring that the displacement operator is Hermitian, it must satisfy the relation $\boldsymbol{\xi}_{k\lambda} = \boldsymbol{\xi}^*_{-k\lambda}$.

Inserting the Fourier transform of $\nabla V(\boldsymbol{r} - \boldsymbol{R}_j^{(0)}) = \sum_q i\boldsymbol{q} V(\boldsymbol{q}) e^{i\boldsymbol{q}\cdot\boldsymbol{r}}$ one gets for the electron-phonon term for a particular $\boldsymbol{r}$, $H_{ep}(\boldsymbol{r})$:

$$H_{ep}(\boldsymbol{r}) = \frac{1}{V} \sum_j \sum_{kq\lambda} \left(\frac{\hbar}{2MN\omega_{k\lambda}} \right)^{1/2} \boldsymbol{\xi}_{k\lambda} \cdot i\boldsymbol{q} \left(a_{k\lambda} e^{-i\omega_{k\lambda} t} + a^\dagger_{-k\lambda} e^{i\omega_{k\lambda} t} \right) V(\boldsymbol{q}) e^{i\boldsymbol{q}\cdot\boldsymbol{r}} e^{i(\boldsymbol{k}-\boldsymbol{q})\cdot\boldsymbol{R}_j^0}$$

Performing the sum over all ions j gives a factor $N\delta_{k,q}$, however as k is restricted to the first Brillouin zone, and q is not, it is necessary to introduce the decomposition $\boldsymbol{q} = \boldsymbol{k}' + \boldsymbol{G}$ where $\boldsymbol{G}$ is the reciprocal lattice vector and $\boldsymbol{k}'$ a vector in 1BZ. This leads to the relation $\boldsymbol{k} - \boldsymbol{k}' = \boldsymbol{G}$.

$$H_{ep}(\boldsymbol{r}) = \frac{i}{V} \sum_{kG\lambda} \left(\frac{N\hbar}{2M\omega_{k\lambda}} \right)^{1/2} \boldsymbol{\xi}_{k\lambda} \cdot (\boldsymbol{k} + \boldsymbol{G}) V(\boldsymbol{k} + \boldsymbol{G}) \left(a_{k\lambda} e^{-i\omega_{k\lambda} t} + a^\dagger_{-k\lambda} e^{i\omega_{k\lambda} t} \right) e^{i(\boldsymbol{k}+\boldsymbol{G})\cdot\boldsymbol{r}} \tag{3.5}$$

In the following, scattering where $\boldsymbol{G} = 0$ is assumed to dominate over the so-called Umklapp processes, where it is not. The second quantized version of the electron-phonon interaction becomes

$$H_{ep} = \sum_{kp\lambda\sigma} M_k c^\dagger_{k+p,\sigma} c_{p,\sigma} \left(a_{k\lambda} + a^\dagger_{-k\lambda} \right) \tag{3.6}$$

with $M_k = i \left(\frac{N\hbar}{2M\omega_{k\lambda}} \right)^{1/2} (\boldsymbol{k} \cdot \boldsymbol{\xi_{k\lambda}}) V(\boldsymbol{k})$. Equation 3.5 describes a process where an electron in a state with momentum $\boldsymbol{k}$ is scattered to a state with momentum $\boldsymbol{k} + \boldsymbol{p}$ by either absorbing a phonon with momentum $\boldsymbol{p}$ or by emitting one with momentum $-\boldsymbol{p}$. Two electrons may then interact as one electron emits a phonon which is absorbed by the other. Such a process may be described by the Feynamn diagram shown in figure 3.1.

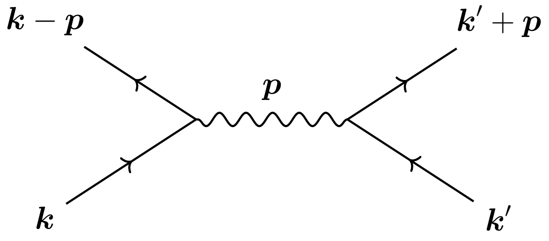

$\boldsymbol{k} - \boldsymbol{p}$
$\boldsymbol{k}' + \boldsymbol{p}$
$\boldsymbol{p}$
$\boldsymbol{k}$
$\boldsymbol{k}'$

Figure 3.1: Feynman diagram for a phonon-mediated electron-electron interaction.

By using the Feynman rules and the free phonon propagator, $D(\boldsymbol{k}, \omega) = \frac{2\omega_{k\lambda}}{\omega - \omega_{k\lambda}}$, one arrives at the following phonon-mediated electron-electron interaction:

$$H_{ep}(\boldsymbol{k}, \omega) = \frac{2|M_{k\lambda}|^2 \omega_{k\lambda}}{\omega - \omega_{k\lambda}} c^\dagger_{k+p,\sigma} c^\dagger_{k'-p,\sigma'} c_{k',\sigma'} c_{k,\sigma} \tag{3.7}$$

This term has exactly the same form as the second quantized two-particle operator, given in equation (2.10). In other words, to account for small oscillations of the ions in the crystal lattice, the pure Coulomb potential between the electrons need only be replaced by an effective potential

$$V_{\text{eff}} = \frac{2|M_{k\lambda}|^2 \omega_{k\lambda}}{\omega - \omega_{k\lambda}} + \frac{e^2}{4\pi\varepsilon_0} \frac{2\pi}{k^2} \tag{3.8}$$

3.1.2 BCS Theory

An illustration of the effective electron-electron interaction is given in figure 3.2a. It is observed that as the frequency of the phonon mediator ω approaches $\omega_{k\lambda}$, V_{eff} diverges. In particular, approaching this limit from below it is seen that the potential becomes increasingly negative. This means that while the Coulomb interaction dominates everywhere else, creating a repulsive interaction, there are certain frequencies for which the electrons feel attraction. It is possible to formulate the equations of motion by using 3.8. This is called strong coupling theory. Another option is to use the simplified model of Bardeen, Cooper and Schrieffer[42] (BCS). It consists of using the potential V_{BCS}, shown in figure 3.2b, consisting of a constant attractive potential for $\omega \leq \omega_D$ and zero otherwise;

$$V_{\text{BCS}} = \begin{cases} -V_0, & \omega \leq \omega_D \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

where $V_0 > 0$ is a constant. This is a good approximation for superconductors where the coupling between electrons and phonons is weak.

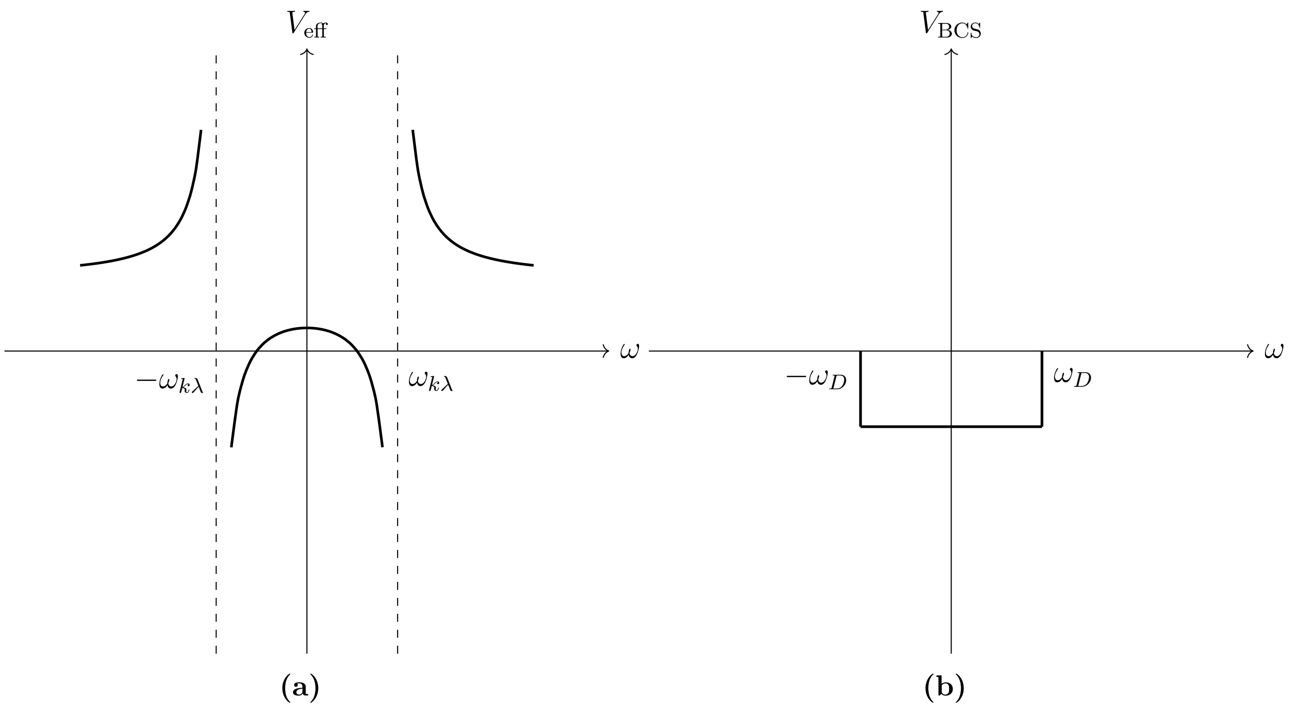


Figure 3.2: The potential used in the development of the microscopic theory of superconductivity. (a) The potential generated by phonon-mediated electron-electron interaction, (b) a simplified potential used in BCS theory.

Another simplification is used next. The potential is attractive for electron energies within a thin shell around the Fermi surface. It is then necessary for both incoming and scattered electrons to have energies within this shell. This can only happen if $\boldsymbol{k}' = -\boldsymbol{k}$. Physically, this means that since a pair of electrons is not attracted to each other directly, but rather to the trail of distorted ions left behind by the other, it is reasonable that the attraction is maximized when the electrons dart past each other in opposite directions. After having passed each other, the electrons quickly separate, thus reducing the Coulomb interaction, leaving only the phonon interactions.

If one assumes that the coupling is so weak that only phonons with energies $\omega \ll \omega_D$ partake in the interaction, the electrons feel a constant potential. It is in this situation reasonable to approximate the effective electron-electron interaction as a point contact in real space; $V_{\text{eff}}(\boldsymbol{r}, \boldsymbol{r}') = -V_0 \delta(\boldsymbol{r} - \boldsymbol{r}')$ The Pauli exclusion principle constrains the spins of the electron pair to be opposite; $\sigma' = -\sigma$, so that the Hamiltonian becomes

$$H = -V_0 \sum_{\sigma} \int d\boldsymbol{r} \psi^{\dagger}_{\sigma}(\boldsymbol{r}) \psi^{\dagger}_{-\sigma}(\boldsymbol{r}) \psi_{-\sigma}(\boldsymbol{r}) \psi_{\sigma}(\boldsymbol{r}) \tag{3.10}$$

A mean-field approach may be applied to equation 3.10 in the order parameter $\Delta(\boldsymbol{r}) = \lambda(\boldsymbol{r}) \langle \psi_{\uparrow}(\boldsymbol{r}) \psi_{\downarrow}(\boldsymbol{r}) \rangle$, where $\lambda(\boldsymbol{r})$ is a real quantity defined to be equal to the constant value $-V_0$ inside a superconductor, and zero elsewhere. This is done by assuming that the deviations from the average pair correlation are small, e.g., $\delta = \psi_{\uparrow}\psi_{\downarrow} - \langle \psi_{\uparrow}\psi_{\downarrow} \rangle \ll 1$, so that one may write

$$\psi_{\uparrow}(\boldsymbol{r}) \psi_{\downarrow}(\boldsymbol{r}) = \langle \psi_{\uparrow}(\boldsymbol{r}) \psi_{\downarrow}(\boldsymbol{r}) \rangle + \delta \tag{3.11}$$

Inserting equation (3.11) into equation (3.10), keeping only first order terms in δ and ignoring constants, the Hamiltonian becomes

$$H = \int d\boldsymbol{r} \left[\Delta(\boldsymbol{r}) \psi^{\dagger}_{\uparrow}(\boldsymbol{r}) \psi^{\dagger}_{\downarrow}(\boldsymbol{r}) + \Delta^{*}(\boldsymbol{r}) \psi_{\downarrow}(\boldsymbol{r}) \psi_{\uparrow}(\boldsymbol{r}) \right] \tag{3.12}$$

The order parameter $\Delta(\boldsymbol{r})$ generally requires self-consistency iterations to compute, as it depends on the solution. Here, superconductors are only to be included as boundary conditions, and so this topic is not pursued further.

3.2 Ferromagnetism

The defining characteristic of a metal is that the Fermi level lies within an electron band. This means that charge carriers can be excited without having to overcome a band gap, as would have been the case for semiconductors. The consequence of this is that within a metal, for energies close to the Fermi surface, the electrons can be thought of as roaming freely throughout the lattice. Far below the Fermi surface, the electrons are localized to the ions that constitute the lattice. To understand what causes a material to be ferromagnetic, it is reasonable to first consider the influence of the localized electrons. This is done by considering to a toy model consisting of a lattice of simplified ions. Belonging to each ion is a single orbital $\phi(\boldsymbol{r})$, containing a single electron. The model can then be considered as half-filled due to the spin-degeneracy. The motivation for considering such a system is that the electron spin can be thought of as a magnetic dipole. Magnetization is then an average over the spins in the material. Reasonable candidates for a system which has a net magnetization are therefore metals whose atoms contain unpaired electrons. The half-filled lattice is then a simplification of a metal with a single unpaired valence electron, and the attraction of the nucleus screened by other, paired (in terms of spin) electrons. The field operators for this system can be written as

$$\psi_\sigma(\boldsymbol{r}) = \sum_i \phi(\boldsymbol{r} - \boldsymbol{R}_i)c_{i\sigma}, \quad \psi^\dagger_\sigma(\boldsymbol{r}) = \sum_i \phi^*(\boldsymbol{r} - \boldsymbol{R}_i)c^\dagger_{i\sigma} \tag{3.13}$$

where $c^\dagger_{i\sigma}$ and $c_{i\sigma}$ are fermion creation and annihilation operators respectively, satisfying the anticommutation relations

$$\left\{c^\dagger_{i\sigma}, c_{j\sigma'}\right\} = \delta_{ij}\delta_{\sigma\sigma'}, \quad \left\{c^\dagger_{i\sigma}, c^\dagger_{j\sigma'}\right\} = \{c_{i\sigma}, c_{j\sigma'}\} = 0 \tag{3.14}$$

It is assumed that the overlap of the orbitals of different ions is negligible, so that

$$\int d\boldsymbol{r}\phi^*(\boldsymbol{r} - \boldsymbol{R}_i)\phi(\boldsymbol{r} - \boldsymbol{R}_j) = \delta_{ij} \tag{3.15}$$

In addition, from the anticommutation relations of the field operators, the following completeness relation is found

$$\sum_i \phi^*(\boldsymbol{r} - \boldsymbol{R}_i)\phi(\boldsymbol{r}' - \boldsymbol{R}_i) = \delta(\boldsymbol{r} - \boldsymbol{r}') \tag{3.16}$$

With the toy model now properly established, the Coulomb interaction between the elctrons is considered. Inserting into equation 2.10 gives

$$H_C = \frac{1}{2}\sum_{ijkl}\sum_{\sigma\sigma'}\int d\boldsymbol{r}d\boldsymbol{r}'V(\boldsymbol{r} - \boldsymbol{r}')\phi^*(\boldsymbol{r} - \boldsymbol{R}_i)\phi^*(\boldsymbol{r}' - \boldsymbol{R}_j)\phi(\boldsymbol{r}' - \boldsymbol{R}_k)\phi(\boldsymbol{r} - \boldsymbol{R}_l)c^\dagger_{i\sigma}c^\dagger_{j\sigma'}c_{k\sigma'}c_{l\sigma} \tag{3.17}$$

It is reasonable that the largest contribution is found for interactions within a single orbital or involving an interchange of orbitals between two electrons. Thus, only processes where $\boldsymbol{R}_i = \boldsymbol{R}_l$ and $\boldsymbol{R}_j = \boldsymbol{R}_k$ or processes where $\boldsymbol{R}_i = \boldsymbol{R}_k$ and $\boldsymbol{R}_j = \boldsymbol{R}_l$ is considered. The Coulomb interaction can then be written as

$$\begin{aligned} H_C = \frac{1}{2}\sum_{ij}\sum_{\sigma\sigma'}\Big\{&\int d\boldsymbol{r}d\boldsymbol{r}'|\phi(\boldsymbol{r} - \boldsymbol{R}_i)|^2V(\boldsymbol{r} - \boldsymbol{r}')|\phi(\boldsymbol{r}' - \boldsymbol{R}_j)|^2c^\dagger_{i\sigma}c_{i\sigma}c^\dagger_{j\sigma'}c_{j\sigma'} \\ &+\int d\boldsymbol{r}d\boldsymbol{r}'V(\boldsymbol{r} - \boldsymbol{r}')\phi^*(\boldsymbol{r} - \boldsymbol{R}_i)\phi^*(\boldsymbol{r}' - \boldsymbol{R}_j)\phi(\boldsymbol{r}' - \boldsymbol{R}_i)\phi(\boldsymbol{r} - \boldsymbol{R}_j)c^\dagger_{i\sigma}c^\dagger_{j\sigma'}c_{i\sigma'}c_{j\sigma}\Big\} \end{aligned} \tag{3.18}$$

By inspecting equation 3.18 it is seen that the first term does not represent spin exchange, but rather a so-called *direct interaction.* This term is not considered further, as its effects are included elsewhere. The *exchange interaction* is given by the second term, and with the use of equation 3.14 it becomes

$$
\begin{aligned}
H_{C2} = &-\frac{1}{2}\sum_{ij}\sum_{\sigma\sigma'}\int d\boldsymbol{r}d\boldsymbol{r'}V(\boldsymbol{r}-\boldsymbol{r'})\phi^*(\boldsymbol{r}-\boldsymbol{R}_i)\phi^*(\boldsymbol{r'}-\boldsymbol{R}_j)\phi(\boldsymbol{r'}-\boldsymbol{R}_i)\phi(\boldsymbol{r}-\boldsymbol{R}_j)c^\dagger_{i\sigma}c_{i\sigma'}c^\dagger_{j\sigma'}c_{j\sigma} \\
&+\sum_i\sum_\sigma\int d\boldsymbol{r}d\boldsymbol{r'}|\phi(\boldsymbol{r}-\boldsymbol{R}_i)|^2V(\boldsymbol{r}-\boldsymbol{r'})|\phi(\boldsymbol{r'}-\boldsymbol{R}_i)|^2c^\dagger_{i\sigma}c_{i\sigma}
\end{aligned}
\tag{3.19}
$$

The second term in equation 3.19 gives no exchange of spins and is disregarded. The first term, however does. This is seen clearly by noting that the spin operators are given in terms of the creation and annihilation operators as

$$
S_i^\alpha = \frac{\hbar}{2}\sum_{\sigma\sigma'}c^\dagger_{i\sigma}\tau^\alpha_{\sigma\sigma'}c_{i\sigma'} \tag{3.20}
$$

where $\alpha \in \{x, y, z\}$ and τ^α is a Pauli matrix. By computation, it is found that

$$
\sum_{\sigma\sigma'}c^\dagger_{i\sigma}c_{i\sigma'}c^\dagger_{j\sigma'}c_{j\sigma} = \frac{1}{2}c^\dagger_{i\sigma}c_{i\sigma}c^\dagger_{j\sigma'}c_{j\sigma'} + \frac{2}{\hbar^2}\boldsymbol{S}_i\cdot\boldsymbol{S}_j \tag{3.21}
$$

Only the second term of 3.21 is relevant for the spin interaction. Inserting it into 3.19 gives the exchange Hamiltonian

$$
H_{ex} = -\sum_{ij}J_{ij}\boldsymbol{S}_i\cdot\boldsymbol{S}_j \tag{3.22}
$$

with the exchange integral J_{ij} given as

$$
J_{ij} = \frac{1}{\hbar^2}\int d\boldsymbol{r}d\boldsymbol{r'}\phi^*(\boldsymbol{r}-\boldsymbol{R}_i)\phi^*(\boldsymbol{r'}-\boldsymbol{R}_j)V(\boldsymbol{r}-\boldsymbol{r'})\phi(\boldsymbol{r'}-\boldsymbol{R}_i)\phi(\boldsymbol{r}-\boldsymbol{R}_j) \tag{3.23}
$$

To establish the sign of J_{ij}, it is observed that that the Coulomb potential can be expressed as

$$
V(\boldsymbol{r}-\boldsymbol{r'}) = \frac{e^2}{4\pi\varepsilon}\int\frac{d\boldsymbol{k}}{(2\pi)^3}\frac{4\pi}{k^2}e^{i\boldsymbol{k}\cdot(\boldsymbol{r}-\boldsymbol{r'})} \tag{3.24}
$$

By switching integration order, it is seen that

$$
J_{ij} = \frac{e^2}{\varepsilon\hbar^2}\int\frac{d\boldsymbol{k}}{(2\pi)^3}\frac{4\pi}{k^2}\left|\int d\boldsymbol{r}\phi^*(\boldsymbol{r}-\boldsymbol{R}_i)\phi(\boldsymbol{r}-\boldsymbol{R}_j)e^{i\boldsymbol{k}\cdot\boldsymbol{r}}\right|^2 \geq 0 \tag{3.25}
$$

For the half-filled lattice of ions, it is thus seen that the exchange integral J_{ij} is always positive. This means that the energy of the system is minimal when all spins are parallel. In other words, this model is ferromagnetic. Hidden in the Coulomb interaction was a spin interaction which comes about due to the Pauli exclusion principle and the limited availability of states. Turning next to electrons close to the Fermi surface, these are only loosely bound to the ions. This suggests the following Hamiltonian for exchange interaction between the localized and non-localized electrons[43]

$$H_{ex} = -\int d\boldsymbol{r} \sum_i J(\boldsymbol{r} - \boldsymbol{R}_i)\boldsymbol{S}(\boldsymbol{R}_i) \cdot \boldsymbol{s}(\boldsymbol{r}) \tag{3.26}$$

where the exchange integral is modeled as a continuous function and the spin density $\boldsymbol{s}(\boldsymbol{r})$ is given as

$$\boldsymbol{s}(\boldsymbol{r}) = \frac{\hbar}{2} \sum_{\sigma\sigma'} \psi_\sigma^\dagger(\boldsymbol{r}) \boldsymbol{\tau}_{\sigma\sigma'} \psi_\sigma(\boldsymbol{r}) \tag{3.27}$$

with $\psi_\sigma^\dagger(\boldsymbol{r})$ and $\psi_\sigma(\boldsymbol{r})$ the field operators of the roaming electrons. The exchange Hamiltonian may then be written as

$$H_{ex} = -\sum_{\sigma\sigma'} \int d\boldsymbol{r} \psi_\sigma^\dagger(\boldsymbol{r}) \boldsymbol{h}(\boldsymbol{r}) \cdot \boldsymbol{\tau}_{\sigma\sigma'} \psi_{\sigma'}(\boldsymbol{r}) \tag{3.28}$$

It is interesting to note that equation 3.28 takes the form of a Zeeman effect, and this is reasonable. When the spins of the localized electrons are oriented in the same direction, they create a magnetic field which the non-localized electrons feel. A real metal is much too complicated to be adequately described by the toy model used in the derivation of the exchange interaction. For this reason, no attempt will be made to compute $\boldsymbol{h}(\boldsymbol{r})$ from first principles. Instead, the interaction is assumed to be of the form of equation 3.28, with $\boldsymbol{h}(\boldsymbol{r})$ given as an input parameter.

3.3 Spin-orbit Coupling

A heuristic explanation for the spin-orbit coupling of electrons is found in the realization that from the point of view of the electrons, the crystal lattice is moving. Furthermore, if the crystal generates a potential V, the electric field may be found as $\boldsymbol{E} = -\nabla V$. Thus, the electron experiences a moving electric field, which in turn creates a magnetic field. The magnetic field couples to the spin of the electron via the Zeeman effect. A more fundamental approach is to consider the Dirac equation[44]

$$c\boldsymbol{\sigma} \cdot \boldsymbol{p}\psi_h + mc^2\psi_e + V\psi_e = (E + mc^2)\psi_e \tag{3.29a}$$

$$c\boldsymbol{\sigma}\cdot\boldsymbol{p}\psi_e - mc^2\psi_h + V\psi_h = (E+mc^2)\psi_h \tag{3.29b}$$

where ψ_e and ψ_h is the wave function for particles and holes respectively. On the right hand side, the rest energy is given explicitly, due to the relativistic nature of the Dirac equation. To apply this equation to condensed matter systems, the limit of $v \ll c$ is to be investigated. Towards this end, equation 3.29b may be solved for ψ_h and inserted into equation 3.29a.

$$\left\{\frac{1}{2m}\boldsymbol{\sigma}\cdot\boldsymbol{p}\left[1+\frac{E-V}{2mc^2}\right]^{-1}\boldsymbol{\sigma}\cdot\boldsymbol{p}+V\right\}\psi_e = E\psi_e \tag{3.30}$$

It is seen that neglecting the c^2-term gives the Schrödinger equation. Going beyond the non-relativistic limit by Taylor expanding and including the next order term gives

$$\left\{\frac{1}{2m}\boldsymbol{p}^2 - \boldsymbol{\sigma}\cdot\boldsymbol{p}\frac{E-V}{4m^2c^2}\boldsymbol{\sigma}\cdot\boldsymbol{p}+V\right\}\psi_e = E\psi_e \tag{3.31}$$

Since $\boldsymbol{p}$ does not commute with V, equation 3.31 becomes

$$\left\{\frac{1}{2m}\boldsymbol{p}^2 - \frac{E-V}{4m^2c^2}\boldsymbol{p}^2 - \frac{\hbar}{4im^2c^2}(\boldsymbol{\sigma}\cdot\nabla V)(\boldsymbol{\sigma}\cdot\boldsymbol{p})+V\right\}\psi_e = E\psi_e \tag{3.32}$$

Setting $E-V \approx \frac{1}{2m}\boldsymbol{p}^2$ and using the identity $(\boldsymbol{\sigma}\cdot\boldsymbol{a})(\boldsymbol{\sigma}\cdot\boldsymbol{b}) = \boldsymbol{a}\cdot\boldsymbol{b} + (\boldsymbol{a}\times\boldsymbol{b})\cdot\boldsymbol{\sigma}$ gives

$$\left\{\frac{1}{2m}\boldsymbol{p}^2 - \frac{1}{8m^3c^2}\boldsymbol{p}^4 - \frac{\hbar}{4im^2c^2}\nabla V\cdot\boldsymbol{p} - \frac{\hbar}{4im^2c^2}\boldsymbol{\sigma}\cdot(\nabla V\times\boldsymbol{p})+V\right\}\psi_e = E\psi_e \tag{3.33}$$

The fourth term of equation 3.33 is the spin-orbit coupling. This is readily seen for a spherically symmetric potential, where it becomes $\propto \frac{dV}{dr}\boldsymbol{L}\cdot\boldsymbol{S}$. In the following, the spin-orbit interaction will be included, while the other relativistic correction terms will be ignored. The Hamiltonian term to be included will thus be

$$H_{SOC} = -\frac{\hbar}{4im^2c^2}\boldsymbol{\sigma}\cdot(\nabla V\times\boldsymbol{p}) \tag{3.34}$$

In a crystal, a major contribution to the potential in equation 3.34 will come from the lattice of ions, which varies in a periodic fashion. By considering a tight binding model, where the electrons are assumed localized to individual lattice points, it can be deduced that for a centrosymmetric crystal, due to symmetry-imposed constraints, the spin-orbit interaction can only take place as a scattering event between different electron bands[45]. In the following, this mechanism for spin-orbit coupling will be ignored, and the discussion restricted to noncentrosymmetric crystals, where the symmetry constraints are relaxed, and an effective single band model is sufficient. One such system is near the surface of a crystal. Here the inversion symmetry of the potential is clearly broken. In fact, the electrons feel the surface in the form of an electric field pointing in the direction of the surface normal $\boldsymbol{n}$, that is $-\nabla V = E\boldsymbol{n}$. Near the surface, the spin-orbit coupling therefore becomes

$$H_R = \alpha(\boldsymbol{n} \times \boldsymbol{p}) \cdot \boldsymbol{\sigma} \tag{3.35}$$

This is the Rashba Hamiltonian[46]. If the material considered has a bulk inversion asymmetry, stemming from the crystal structure, the spin-orbit interaction takes the form of the Dresselhaus Hamiltonian[47]

$$H_D = \beta' \left[p_x \left(p_y^2 - p_z^2\right) \sigma_x + p_y \left(p_z^2 - p_x^2\right) \sigma_y + p_z \left(p_x^2 - p_y^2\right) \sigma_z\right] \tag{3.36}$$

If the crystal is a thin film, it is possible to approximate equation 3.36 by averaging over the thickness direction. Assuming that the film lies in the xy-plane, and integrating over the z-direction gives

$$H_D = \beta \left(p_y\sigma_y - p_x\sigma_x\right) \tag{3.37}$$

From equations 3.35 and 3.37 it is seen that spin-orbit coupling may be described approximatively, at least in some cases, by an interaction which is linear in $\boldsymbol{p}$. Generalizing these results, it is reasonable to write the spin-orbit interaction as

$$H_{SOC} = \sum_{ij} \alpha_{ji}\sigma_i p_j = \boldsymbol{w} \cdot \boldsymbol{p} \tag{3.38}$$

Equation 3.38 incorporates the Rashba and linearized Dresselhaus interactions. Writing out the kinetic energy term of the Hamiltonian in the presence of a magnetic vector potential gives

$$\frac{1}{2m}(\boldsymbol{p} - e\boldsymbol{A})^2 = \frac{1}{2m}(\boldsymbol{p}^2 - 2e\boldsymbol{A} \cdot \boldsymbol{p} + e^2\boldsymbol{A}^2) \tag{3.39}$$

The second term of equation 3.39 is of the form of equation 3.38, and by neglecting the term $(\boldsymbol{w} \cdot \boldsymbol{p})^2$, the spin-orbit Hamiltonian may be included by the replacement

$$\boldsymbol{A} \to e\boldsymbol{A} - m\boldsymbol{w} \tag{3.40}$$

It is emphasized that due to $\boldsymbol{w}$, $\boldsymbol{A}$ has structure in spin-space.

3.4 Impurity scattering

The materials considered are assumed to be dirty in the sense that they contain a large number of randomly distributed impurities, e.g., lattice defects. Their presence is modeled by an impurity potential, given as

$$V_{\mathrm{imp}}(\boldsymbol{r}) = \sum_j U(\boldsymbol{r} - \boldsymbol{r}_j) \tag{3.41}$$

where $U(\boldsymbol{r} - \boldsymbol{r}_j)$ is the potential from a particular impurity that is located at position $\boldsymbol{r}_j$. It is assumed that all impurities are identical. It is a daunting task to include such a potential in any physical theory, due to the impracticably large number of impurities. However, within the Green function formalism, the problem is made much more tractable by averaging over the impurities[48,49]. In this way, the impurities can be included as a perturbation of the Green function[40,50]

$$\check{G}(\boldsymbol{r}, \boldsymbol{r}') = \check{G}^{(0)}(\boldsymbol{r} - \boldsymbol{r}') + \sum_n \check{G}^{(n)}(\boldsymbol{r}, \boldsymbol{r}') \tag{3.42}$$

where $\check{G}^{(n)}(\boldsymbol{r}, \boldsymbol{r}')$ is given as

$$\begin{aligned}\check{G}^{(n)}(\boldsymbol{r}, \boldsymbol{r}') = \prod_{k=1}^{n} \left(\sum_{j_k} \int d\boldsymbol{r}_k \right) \check{G}^{(0)}(\boldsymbol{r} - \boldsymbol{r}_1) u(\boldsymbol{r}_1 - \boldsymbol{r}_{j_1}) \check{G}^{(0)}(\boldsymbol{r}_1 - \boldsymbol{r}_2) \times \ldots \\ \ldots \times \check{G}^{(0)}(\boldsymbol{r}_{n-1} - \boldsymbol{r}_n) u(\boldsymbol{r}_n - \boldsymbol{r}_{j_n}) \check{G}^{(0)}(\boldsymbol{r}_n - \boldsymbol{r}')\end{aligned} \tag{3.43}$$

where $u(\boldsymbol{r}) = \frac{U(\boldsymbol{r})}{V}$ and V is the volume of the material. The inverse Fourier transforms for $\check{G}^{(0)}$ and u are inserted, and the integrals over r_k are performed. Each spatial integral produces one δ-function, and after some tidying up this becomes

$$\begin{aligned}\check{G}^{(n)}(\boldsymbol{r}, \boldsymbol{r}') = \sum_{j_1, \ldots j_n}^{N} \int \frac{d\boldsymbol{p} d\boldsymbol{p}'}{(2\pi\hbar)^6} e^{i(\boldsymbol{p}\cdot\boldsymbol{r} - \boldsymbol{p}'\cdot\boldsymbol{r}')/\hbar} \prod_{k=1}^{n-1} \left(\int \frac{d\boldsymbol{p}_k}{(2\pi\hbar)^3} \right) \times \\ \times e^{-i(\boldsymbol{p}-\boldsymbol{p}_1)\cdot\boldsymbol{r}_{j_1}/\hbar} e^{-i(\boldsymbol{p}_1-\boldsymbol{p}_2)\cdot\boldsymbol{r}_{j_2}/\hbar} \ldots e^{-i(\boldsymbol{p}_{n-1}-\boldsymbol{p}')\cdot\boldsymbol{r}_{j_n}/\hbar} \times \\ \times \check{G}^{(0)}(\boldsymbol{p}) u(\boldsymbol{p} - \boldsymbol{p}_1) \check{G}^{(0)}(\boldsymbol{p}_1) \ldots \check{G}^{(0)}(\boldsymbol{p}_n) u(\boldsymbol{p}_n - \boldsymbol{p}') \check{G}^{(0)}(\boldsymbol{p}')\end{aligned} \tag{3.44}$$

Every term n in the perturbation expansion gives rise to n scattering events. However, these scatterings need not happen on individual impurities. To fully evaluate the sum in equation (3.44), it is necessary to take into account all combinations of multiple scatterings as well. With the contributing terms sorted appropriately, the sum over impurities may be approximated as

$$\sum_{j}^{N} \rightarrow n_{\text{imp}} \int d\boldsymbol{r}_j \tag{3.45}$$

where $n_{\text{imp}} = \frac{N}{V}$. The integration over $\boldsymbol{r}_j$ produces δ-functions which simplify the result. For $n = 1$ one gets

$$\check{G}^{(1)}(\boldsymbol{r}, \boldsymbol{r}') = n_{\text{imp}} \int \frac{d\boldsymbol{p}}{(2\pi\hbar)^3} e^{i\boldsymbol{p}\cdot(\boldsymbol{r}-\boldsymbol{r}')} \check{G}^{(0)}(\boldsymbol{p}) u(0) \check{G}^{(0)}(\boldsymbol{p}) \tag{3.46}$$

For $n = 2$, two terms appear

$$\begin{aligned} \check{G}^{(2)}(\boldsymbol{r}, \boldsymbol{r}') = n_{\text{imp}}^2 \int \frac{d\boldsymbol{p}}{(2\pi\hbar)^3} e^{i\boldsymbol{p}\cdot(\boldsymbol{r}-\boldsymbol{r}')} \check{G}^{(0)}(\boldsymbol{p}) u(0) \check{G}^{(0)}(\boldsymbol{p}) u(0) \check{G}^{(0)}(\boldsymbol{p}) \\ + \int \frac{d\boldsymbol{p}}{(2\pi\hbar)^3} e^{i\boldsymbol{p}\cdot(\boldsymbol{r}-\boldsymbol{r}')} \check{G}^{(0)}(\boldsymbol{p}) \Sigma^{(0)}(\boldsymbol{p}) \check{G}^{(0)}(\boldsymbol{p}) \end{aligned} \tag{3.47}$$

The first term in equation (3.46) is found by restricting $j_1 \neq j_2$, and so this corresponds to a single scattering event per impurity. By comparing with equation (3.46) a pattern emerges. For every order n, there is one term which only depends on $u(0)$. This term describes a trivial interaction with the impurities, and may be included in the chemical potential. In the second term, the zeroth order self-energy has been introduced, which is defined as

$$\Sigma^{(0)}(\boldsymbol{p}) = n_{\text{imp}} \int \frac{d\boldsymbol{p}_1}{(2\pi\hbar)^3} |u(\boldsymbol{p} - \boldsymbol{p}_1)|^2 \check{G}^{(0)}(\boldsymbol{p}_1) \tag{3.48}$$

where it has been used that $u(\boldsymbol{p}_1 - \boldsymbol{p}) = u^*(\boldsymbol{p} - \boldsymbol{p}_1)$. By writing out higher order terms in the perturbation expansion and comparing the results, it is seen that the Green function can be expressed in terms of such self-energy terms in the following way

$$\begin{aligned} \check{G}(\boldsymbol{r}, \boldsymbol{r}') &= \int \frac{d\boldsymbol{p}}{(2\pi\hbar)^3} e^{i\boldsymbol{p}\cdot(\boldsymbol{r}-\boldsymbol{r}')} \Big[\check{G}^{(0)}(\boldsymbol{p}) + \sum_i \check{G}^{(0)}(\boldsymbol{p}) \Sigma^{(i)}(\boldsymbol{p}) \check{G}^{(0)}(\boldsymbol{p}) \\ &\quad + \sum_{i,j} \check{G}^{(0)}(\boldsymbol{p}) \Sigma^{(i)}(\boldsymbol{p}) \check{G}^{(0)}(\boldsymbol{p}) \Sigma^{(j)}(\boldsymbol{p}) \check{G}^{(0)}(\boldsymbol{p}) + \dots \Big] \\ &= \int \frac{d\boldsymbol{p}}{(2\pi\hbar)^3} e^{i\boldsymbol{p}\cdot(\boldsymbol{r}-\boldsymbol{r}')} \left[\check{G}^{(0)}(\boldsymbol{p}) + \check{G}^{(0)}(\boldsymbol{p}) \Sigma(\boldsymbol{p}) \check{G}(\boldsymbol{p}) \right] \end{aligned} \tag{3.49}$$

This is the Dyson equation. The Green function $\check{G}(\boldsymbol{p})$ is the Fourier transformed of $\check{G}(\boldsymbol{r}, \boldsymbol{r}')$ and the total self-energy is defined as the sum of the contribution from all orders

$$\Sigma(\boldsymbol{p}) = \sum_i \Sigma^{(i)}(\boldsymbol{p}) \tag{3.50}$$

Including impurities therefore boils down to finding $\Sigma(\boldsymbol{p})$. Equation (3.50) can be computed in the Born approximation, where only terms where scattering occurs on each impurity twice are included. The result is[50]

$$\Sigma(\boldsymbol{p}) = n_{\text{imp}} \int \frac{d\boldsymbol{p}_1}{(2\pi\hbar)^3} |u(\boldsymbol{p} - \boldsymbol{p}_1)|^2 \check{G}(\boldsymbol{p}_1) \tag{3.51}$$

3.5 The equations of motion

The complete, second quantized Hamiltonian to be considered is given as

$$\begin{aligned} H = &\sum_\sigma \int d\boldsymbol{r} \psi^\dagger_\sigma(\boldsymbol{r}, t) \left[\frac{1}{2m}(\boldsymbol{p} - e\boldsymbol{A})^2 - \mu + V_{imp}(\boldsymbol{r}) \right] \psi_\sigma(\boldsymbol{r}, t) \\ &+ \int d\boldsymbol{r} \left[\Delta(\boldsymbol{r}) \psi^\dagger_\uparrow(\boldsymbol{r}, t) \psi^\dagger_\downarrow(\boldsymbol{r}, t) + \Delta^*(\boldsymbol{r}) \psi_\downarrow(\boldsymbol{r}, t) \psi_\uparrow(\boldsymbol{r}, t) \right] \\ &- \sum_{\sigma\sigma'} \int d\boldsymbol{r} \psi^\dagger_\sigma(\boldsymbol{r}, t) \boldsymbol{h}(\boldsymbol{r}) \cdot \boldsymbol{\tau}_{\sigma\sigma'} \psi_{\sigma'}(\boldsymbol{r}, t) \end{aligned} \tag{3.52}$$

where the spin-orbit coupling is included in the matrix $\boldsymbol{A}$, which therefore gets structure in spin space. It is noted that the exchange field and the superconducting order parameter both appear in equation 3.52. Generally, materials do not exhibit these two phenomena simultaneously, however exceptions do exist. One possibility is if the superconducting material has a thickness that is much less than the penetration depth. An external magnetic field may then enter the superconductor and create spin splitting[51]. Ferromagnetic order can also be accommodated in triplet superconductors[52]. Inserting equation 3.52 into equation 2.12 and using the anticommutation relations in equation 2.8, results in[53]

$$\begin{aligned} i\hbar \frac{\partial}{\partial t} \psi_\sigma(\boldsymbol{r}, t) = &\left[\frac{1}{2m}(\boldsymbol{p} - e\boldsymbol{A})^2 - \mu + V_{imp}(\boldsymbol{r}) \right] \psi_\sigma(\boldsymbol{r}, t) \\ &+ \delta_{\sigma\uparrow} \Delta(\boldsymbol{r}) \psi^\dagger_\downarrow(\boldsymbol{r}, t) - \delta_{\sigma\downarrow} \Delta(\boldsymbol{r}) \psi^\dagger_\uparrow(\boldsymbol{r}, t) \\ &- \sum_{\sigma'} \boldsymbol{\tau}_{\sigma\sigma'} \cdot \boldsymbol{h} \psi_{\sigma'}(\boldsymbol{r}, t) \end{aligned} \tag{3.53}$$

Similarly, the equation of motion for the adjoint field operator is given as

$$\begin{aligned} i\hbar\frac{\partial}{\partial t}\psi_\sigma^\dagger(\boldsymbol{r},t) = &-\left[\frac{1}{2m}(\boldsymbol{p}+e\boldsymbol{A}^*)^2-\mu+V_{imp}(\boldsymbol{r})\right]\psi_\sigma^\dagger(\boldsymbol{r},t) \\ &-\delta_{\sigma\uparrow}\Delta^*(\boldsymbol{r})\psi_\downarrow(\boldsymbol{r},t)+\delta_{\sigma\downarrow}\Delta^*(\boldsymbol{r})\psi_\uparrow(\boldsymbol{r},t) \\ &+\sum_{\sigma'}\boldsymbol{\tau}^*_{\sigma\sigma'}\cdot\boldsymbol{h}\psi^\dagger_{\sigma'}(\boldsymbol{r},t) \end{aligned} \tag{3.54}$$

where the property $\tau_{\sigma'\sigma} = \tau^*_{\sigma\sigma'}$ has been used. In Nambu⊗spin space, the notation may be abbreviated to

$$\begin{aligned} i\hbar\frac{\partial}{\partial t}\hat{\rho}_3\psi(\boldsymbol{r},\boldsymbol{t}) &= \left[\frac{1}{2m}(\boldsymbol{p}\hat{I}-\hat{\boldsymbol{A}})^2+V_{\text{imp}}(\boldsymbol{r})\hat{I}-\mu\hat{I}+\hat{\Delta}(\boldsymbol{r})-\hat{\boldsymbol{\tau}}\cdot\boldsymbol{h}(\boldsymbol{r})\right]\psi(\boldsymbol{r},t) \\ &= \bar{H}(\boldsymbol{r})\psi(\boldsymbol{r},t) \end{aligned} \tag{3.55a}$$

$$\begin{aligned} i\hbar\frac{\partial}{\partial t}\psi^\dagger(\boldsymbol{r},\boldsymbol{t})\hat{\rho}_3 &= \psi^\dagger(\boldsymbol{r},\boldsymbol{t})\left[\frac{1}{2m}(\boldsymbol{p}\hat{I}+\hat{\boldsymbol{A}})^2+V_{\text{imp}}(\boldsymbol{r})\hat{I}-\mu\hat{I}-\hat{\Delta}(\boldsymbol{r})-\hat{\boldsymbol{\tau}}\cdot\boldsymbol{h}(\boldsymbol{r})\right] \\ &= -\psi^\dagger(\boldsymbol{r},t)\bar{H}^\dagger(\boldsymbol{r}) \end{aligned} \tag{3.55b}$$

with $\hat{\boldsymbol{A}} = \text{diag}(\boldsymbol{A}, -\boldsymbol{A}^*)$ and $\hat{\boldsymbol{\tau}} = \text{diag}(\boldsymbol{\tau}, \boldsymbol{\tau}^*)$. The matrix of superconducting order parameters is given as

$$\hat{\Delta}(\boldsymbol{r}) = \begin{pmatrix} 0 & 0 & 0 & \Delta(\boldsymbol{r}) \\ 0 & 0 & -\Delta(\boldsymbol{r}) & 0 \\ 0 & \Delta^*(\boldsymbol{r}) & 0 & 0 \\ -\Delta^*(\boldsymbol{r}) & 0 & 0 & 0 \end{pmatrix} \tag{3.56}$$

The equation of motion for the retarded Green function is found by acting on equation 2.30 from the right with the operator $i\hbar\frac{\partial}{\partial t}\hat{\rho}_3$

$$\begin{aligned} i\hbar\frac{\partial}{\partial t}\hat{\rho}_3\hat{G}^R(\boldsymbol{r},t,\boldsymbol{r}',t') &= i\hbar\frac{\partial}{\partial t}\hat{\rho}_3\left\{-i\theta(t-t')\hat{\rho}_3\left\langle\left\{\psi(\boldsymbol{r},t),\psi^\dagger(\boldsymbol{r}',t')\right\}_N\right\rangle\right\} \\ &= \hbar\delta(t-t')\left\langle\left\{\psi(\boldsymbol{r},t),\psi^\dagger(\boldsymbol{r}',t)\right\}_N\right\rangle \\ &\quad - i\theta(t-t')\hat{\rho}_3\left\langle\left\{i\hbar\frac{\partial}{\partial t}\hat{\rho}_3\psi(\boldsymbol{r},t),\psi^\dagger(\boldsymbol{r}',t')\right\}_N\right\rangle \end{aligned}$$

Inserting equation 3.55a gives

$$i\hbar\frac{\partial}{\partial t}\hat{\rho}_3\hat{G}^R(\boldsymbol{r},t,\boldsymbol{r}',t') = \hbar\delta(t-t')\delta(\boldsymbol{r}-\boldsymbol{r}') - i\theta(t-t')\hat{\rho}_3\left\langle\left\{\bar{H}\psi(\boldsymbol{r},t),\psi^\dagger(\boldsymbol{r}',t')\right\}_N\right\rangle \tag{3.57}$$

Now comes a subtle point. It is desirable to extract the Hamiltonian matrix $\bar{H}$ from the second term, so that $\hat{G}^R$ can be inserted. To accomplish this, $\hat{\rho}_3^2 = \hat{I}$ is inserted between $\bar{H}$ and $\psi(\boldsymbol{r}, t)$. A new Hamiltonian matrix may be defined as $\hat{H} = \hat{\rho}_3 \bar{H} \hat{\rho}_3$ so that the equation of motion becomes

$$\left(i\hbar \frac{\partial}{\partial t} \hat{\rho}_3 - \hat{H}(\boldsymbol{r}) \right) \hat{G}^R(\boldsymbol{r}, t, \boldsymbol{r}', t') = \hbar \delta(t - t') \delta(\boldsymbol{r} - \boldsymbol{r}') \hat{I} \tag{3.58}$$

The new matrix $\hat{H}$ is identical to $\bar{H}$ except that the BCS-term changes sign, i.e., $\hat{\Delta} \to -\hat{\Delta}$ in equation 3.55a. The same procedure applied to the advanced and Keldysh Green functions results in

$$\left(i\hbar \frac{\partial}{\partial t} \hat{\rho}_3 - \hat{H}(\boldsymbol{r}) \right) \hat{G}^A(\boldsymbol{r}, t, \boldsymbol{r}', t') = \hbar \delta(t - t') \delta(\boldsymbol{r} - \boldsymbol{r}') \hat{I} \tag{3.59}$$

$$\left(i\hbar \frac{\partial}{\partial t} \hat{\rho}_3 - \hat{H}(\boldsymbol{r}) \right) \hat{G}^K(\boldsymbol{r}, t, \boldsymbol{r}', t') = 0 \tag{3.60}$$

Finally, the equation of motion for the full Keldysh Green function is

$$\left(i\hbar \frac{\partial}{\partial t} \check{\rho}_3 - \check{H}(\boldsymbol{r}) \right) \check{G}(\boldsymbol{r}, t, \boldsymbol{r}', t') = \hbar \delta(t - t') \delta(\boldsymbol{r} - \boldsymbol{r}') \check{I} \tag{3.61}$$

An equivalent equation of motion is found by acting on equation 2.30 from the left with the operator $-i\hbar \frac{\partial}{\partial t'} \hat{\rho}_3$. The matrix equation in Keldysh space then takes the form

$$\check{G}(\boldsymbol{r}, t, \boldsymbol{r}', t') \left(i\hbar \frac{\partial}{\partial t'} \check{\rho}_3 - \check{H}(\boldsymbol{r}') \right)^\dagger = \hbar \delta(t - t') \delta(\boldsymbol{r} - \boldsymbol{r}') \check{I} \tag{3.62}$$

It is noted that from this derivation, it is found that the adjoint of the redefined Hamiltonian matrix in Nambu$\otimes$spin space becomes $\hat{H}^\dagger = \bar{H}^\dagger$. In other words, the order parameter matrix $\hat{\Delta}$ has the same sign in equation 3.61 and in equation 3.62, even though it is anti-Hermitian, as can be seen from equation 3.56. The unwieldy δ-functions may be removed by subtracting equations 3.61 and 3.62, giving

$$\left(i\hbar \frac{\partial}{\partial t} \check{\rho}_3 - \check{H}(\boldsymbol{r}) \right) \check{G}(\boldsymbol{r}, t, \boldsymbol{r}', t') - \check{G}(\boldsymbol{r}, t, \boldsymbol{r}', t') \left(i\hbar \frac{\partial}{\partial t'} \check{\rho}_3 - \check{H}(\boldsymbol{r}') \right)^\dagger = 0 \tag{3.63}$$

Solution for a bulk superconductor

For a superconductor that is infinite in every direction, the Hamiltonian reduces to

$$\hat{H}(\boldsymbol{r}) = \frac{1}{2m}\boldsymbol{p}^2\hat{I} - \mu\hat{I} - \hat{\Delta}(\boldsymbol{r}) \tag{3.64}$$

Following Ref. [53] and inserting into equation 3.58 gives

$$\left(i\hbar\frac{\partial}{\partial t}\hat{\rho}_3 - \left(\frac{1}{2m}\boldsymbol{p}^2 - \mu\right)\hat{I} + \hat{\Delta}(\boldsymbol{r})\right)\hat{G}^R(\boldsymbol{r}, t, \boldsymbol{r'}, t') = \hbar\delta(t-t')\delta(\boldsymbol{r}-\boldsymbol{r'})\hat{I} \tag{3.65}$$

The Green function must be translation invariant both in space and time, which means that equation 3.65 only depends on the relative coordinates $\boldsymbol{r} - \boldsymbol{r'}$ and $t - t'$. Fourier transforming gives

$$\left(\varepsilon\hat{\rho}_3 - \left(\frac{1}{2m}\boldsymbol{p} - \mu\right)\hat{I} + \hat{\Delta}(\boldsymbol{r})\right)\hat{G}^R(\boldsymbol{p}, \varepsilon) = \hbar\hat{I} \tag{3.66}$$

Inverting this matrix gives

$$\hat{G}^R(\boldsymbol{p}, \varepsilon) = \frac{1}{\varepsilon^2 - \left(\frac{\boldsymbol{p}^2}{2m} - \mu\right)^2 - |\Delta|^2}\left[\varepsilon\hat{\rho}_3 + \left(\frac{\boldsymbol{p}^2}{2m} - \mu\right)\hat{I} + \hat{\Delta}\right] \tag{3.67}$$

Chapter 4

Quasiclassical Theory

The quasiclassical theory is a formalism where one considers only the envelopes of the Green functions, and ignore rapid oscillations. In doing so, information is lost about phenomena that occur on the same scale as the oscillations. However, many interesting physical effects evolve on longer length scales where quasiclassical theory is valid. Below, a detailed derivation of the quasiclassical equations of motion is provided.

4.1 The mixed representation

As a stepping stone towards applying the quasiclassical approximation, the rapidly varying relative motion of the coordinates in the Green function may be separated from the slowly varying center of mass motion by a coordinate transformation. It is assumed that the Hamiltonian contains no explicit time dependence, so that the Green function is translation invariant in time.

$$\begin{aligned} \boldsymbol{r} = \boldsymbol{r_1} - \boldsymbol{r_2}, \quad \boldsymbol{R} = \frac{1}{2}(\boldsymbol{r_1} + \boldsymbol{r_2}) \\ t = t_1 - t_2, \quad T = \frac{1}{2}(t_1 + t_2) \end{aligned} \tag{4.1}$$

Writing out equation 3.63 gives with $\check{G} = \check{G}(\boldsymbol{r}_1, t_1, \boldsymbol{r_2}, t_2)$

$$\begin{aligned} & i\hbar(\frac{\partial}{\partial t_1}\check{\rho}_3\check{G} + \frac{\partial}{\partial t_2}\check{G}\check{\rho}_3) + \frac{\hbar^2}{2m}(\nabla_1^2 - \nabla_2^2)\check{G} - i\frac{\hbar}{m}[\check{\boldsymbol{A}}(\boldsymbol{r_1}) \cdot \nabla_1\check{G} + \nabla_2\check{G} \cdot \check{\boldsymbol{A}}(\boldsymbol{r_2})] \\ & - \frac{1}{2m}[\check{\boldsymbol{A}}^2(\boldsymbol{r_1})\check{G} - \check{G}\check{\boldsymbol{A}}^2(\boldsymbol{r_2})] - [V_{\text{imp}}(\boldsymbol{r}_1) - V_{\text{imp}}(\boldsymbol{r_2})]\check{G} \\ & + [\check{\Delta}(\boldsymbol{r}_1) + \check{\boldsymbol{\tau}} \cdot \boldsymbol{h}(\boldsymbol{r_1})]\check{G} - \check{G}[\check{\Delta}(\boldsymbol{r}_2) + \check{\boldsymbol{\tau}} \cdot \boldsymbol{h}(\boldsymbol{r_2})] = 0 \end{aligned} \tag{4.2}$$

The mixed representation, or Wigner transform, is found by Fourier transforming equation 4.2 in the relative coordinates $\boldsymbol{r}$ and t. This is, however, not entirely straight forward due to the products of spatially varying functions. Such product-terms are treated in appendix A.3. When Fourier transforming, the terms involving derivatives in the coordinates $(\boldsymbol{r}, t_1)$ become

$$\begin{aligned}\int d\boldsymbol{r} d\varepsilon e^{(-i\boldsymbol{p}\cdot\boldsymbol{r}+i\varepsilon t)/\hbar}\frac{\partial}{\partial t_1}\check{\rho}_3\check{G} &= \int d\boldsymbol{r} d\varepsilon e^{(-i\boldsymbol{p}\cdot\boldsymbol{r}+i\varepsilon t)/\hbar}\left(\frac{\partial}{\partial t}+\frac{1}{2}\frac{\partial}{\partial T}\right)\check{\rho}_3\check{G} \\ &= \left(-i\frac{\varepsilon}{\hbar}+\frac{1}{2}\frac{\partial}{\partial T}\right)\check{\rho}_3\check{\mathcal{G}}\end{aligned} \tag{4.3}$$

$$\begin{aligned}\int d\boldsymbol{r} d\varepsilon e^{(-i\boldsymbol{p}\cdot\boldsymbol{r}+i\varepsilon t)/\hbar}\nabla_1^2\check{G} &= \int d\boldsymbol{r} d\varepsilon e^{(-i\boldsymbol{p}\cdot\boldsymbol{r}+i\varepsilon t)/\hbar}\left(\nabla_r+\frac{1}{2}\nabla_R\right)^2\check{G} \\ &= \left(-\frac{\boldsymbol{p}^2}{\hbar^2}+\frac{i}{\hbar}\boldsymbol{p}\cdot\nabla_R+\frac{1}{4}\nabla_R^2\right)\check{\mathcal{G}}\end{aligned} \tag{4.4}$$

$$\begin{aligned}\int d\boldsymbol{r} d\varepsilon e^{(-i\boldsymbol{p}\cdot\boldsymbol{r}+i\varepsilon t)/\hbar}\check{\boldsymbol{A}}(\boldsymbol{r_1})\cdot\nabla_1\check{G} &= \int d\boldsymbol{r} d\varepsilon e^{(-i\boldsymbol{p}\cdot\boldsymbol{r}+i\varepsilon t)/\hbar}\check{\boldsymbol{A}}(\boldsymbol{r_1})\cdot\left(\nabla_r+\frac{1}{2}\nabla_R\right)\check{G} \\ &= \check{\boldsymbol{A}}(\boldsymbol{R})\otimes\left(\frac{i}{\hbar}\boldsymbol{p}+\frac{1}{2}\nabla_R\right)\check{\mathcal{G}}\end{aligned} \tag{4.5}$$

with $\check{\mathcal{G}} = \check{\mathcal{G}}(\boldsymbol{R}, T; \boldsymbol{p}, \varepsilon)$ the Fourier transformed of $\check{G} = \check{G}(\boldsymbol{r}_1, t_1, \boldsymbol{r}_2, t_2)$ in the relative coordinates $\boldsymbol{r}$ and t. Similar results are found also for the $(\boldsymbol{r}_2, t_2)$ coordinates. Inserting into equation 4.2, the resulting equation becomes:

$$\begin{aligned}\frac{i\hbar}{m}\boldsymbol{p}\cdot\left(\nabla_R\check{\mathcal{G}}-\frac{i}{\hbar^2}\left[\check{\boldsymbol{A}}\overset{\otimes}{,}\check{\mathcal{G}}\right]\right)+\left[\varepsilon\check{\rho}_3-\frac{1}{2m}\check{\boldsymbol{A}}^2-V_{\text{imp}}\check{I}-\check{\Delta}-\check{\boldsymbol{\tau}}\cdot\boldsymbol{h}\overset{\otimes}{,}\check{\mathcal{G}}\right]& \\ -\frac{i\hbar}{2m}\left\{\check{\boldsymbol{A}}\overset{\otimes}{,}\nabla_R\check{\mathcal{G}}\right\}&=0\end{aligned} \tag{4.6}$$

4.2 The quasiclassical approximation

The Green function $\check{\mathcal{G}}$ oscillates in the parameter $\boldsymbol{r}$ on a scale of the Fermi wavelength λ_F[54]. Thus, the wavevector $\boldsymbol{p}$ is of the order λ_F^{-1}. The physical quantities of interest, however, evolve on much greater length scales L. This means that $\nabla_R\check{\mathcal{G}}$ must be of the order L^{-1}. From this, it may be concluded that

$$\left\{\check{\boldsymbol{A}}\overset{\otimes}{,}\nabla_R\check{\mathcal{G}}\right\} \ll \boldsymbol{p}\cdot\left[\check{\boldsymbol{A}}\overset{\otimes}{,}\check{\mathcal{G}}\right]$$

and so the anticommutator in equation 4.6 is neglected. Further dimensional analysis uncovers that $\check{\boldsymbol{A}}^2 \ll \boldsymbol{p}\cdot\check{\boldsymbol{A}}$, so that the $\check{\boldsymbol{A}}^2$-term in equation 4.6 is also neglected.

Inserting these approximations into equation 4.6 and using equations (A.19) and (A.20) results in

$$\frac{i\hbar}{m}\boldsymbol{p}\cdot\left(\nabla_R\check{\mathcal{G}}-\frac{i}{\hbar^2}\left[\check{\boldsymbol{A}},\check{\mathcal{G}}\right]\right)+\left[\varepsilon\check{\rho}_3-V_{\text{imp}}\check{I}-\check{\Delta}-\check{\boldsymbol{\tau}}\cdot\boldsymbol{h},\check{\mathcal{G}}\right] \\ -\frac{i\hbar}{2}\left\{\nabla_R\left(V_{\text{imp}}\check{I}+\check{\Delta}+\check{\boldsymbol{\tau}}\cdot\boldsymbol{h}\right),\nabla_p\check{\mathcal{G}}\right\}=0 \tag{4.7}$$

It is possible to define quasiclassical Green functions, where the rapidly varying oscillations have been integrated out. This is done by realizing that most of the particles participating in physical phenomena have a momentum close to the Fermi-momentum $\boldsymbol{p}_F$. The reason for this is that far below the Fermi surface, every state has been filled (for low temperatures), and so the system is noninteracting. Far above the Fermi-surface, there are no particles. For this reason, most of the physics must occur around the Fermi-surface. This materializes as a pronounced peak in the Green function[54]. Integrating over the momentum is then approximately equal to selecting the Fermi-momentum. It is however necessary to keep the directional information of the Fermi momentum, so an appropriate integration parameter is $\xi_p = \frac{\boldsymbol{p}^2}{2m}-\mu$. The quasiclassical Green function is defined as[55,56]

$$\check{g}(\boldsymbol{R},T,\boldsymbol{p}_F,\varepsilon)=\frac{i}{\pi}\int d\xi_p\check{\mathcal{G}}(\boldsymbol{R},T,\boldsymbol{p},\varepsilon) \tag{4.8}$$

A subtle point is that the integral in equation 4.8 generally diverges[50]. Several ways of circumventing this have been suggested. One option is to simply use a high-energy cut-off[57]. Another option is to split the integral into a low-energy and a high-energy contour, and ignore the latter since it does not contribute to the physical quantities of interest[58]. A third option, as introduced by Shelankov[59], is to not perform the integral at all, and instead derive the quasiclassical Green function by considering motion along classical trajectories. All methods give the same results. It is in the following assumed that appropriate measures have been taken in dealing with the divergence. Introducing the quasiclassical Green function in equation 4.8 to equation 4.7 one gets

$$\frac{i\hbar}{m}\boldsymbol{p}_F\cdot\bar{\nabla}\check{g}+\left[\varepsilon\check{\rho}_3-V_{\text{imp}}\check{I}-\check{\Delta}-\check{\boldsymbol{\tau}}\cdot\boldsymbol{h},\check{g}\right]=0 \tag{4.9}$$

where $\bar{\nabla}\check{g}=\nabla\check{g}-\frac{i}{\hbar^2}\left[\check{\boldsymbol{A}},\check{g}\right]$ and $\nabla=\nabla_R$ constitute notational simplifications used from here on. It is observed that the anticommutator in equation 4.9 drops out, as $\check{g}$ is constant in the momentum coordinate p. Equation 4.9 is known as the Eilenberger equation[58].

Due to the presence of the commutators in equation 4.9, the Green function is only determined up to a constant. To fully determine the Green function, a normalization condition is used:

$$\check{g}^2 = \check{I} \tag{4.10}$$

From equations 2.34, 2.35 and 2.36, it is seen that the Green functions in Nambu-space involve complex conjugate submatrices in spin space; $(G^R)^*$, $(F^R)^*$ etc. The complex conjugate Fourier transform in $(\boldsymbol{r}, t)$ of these matrices is found by making the substitutions $\boldsymbol{p} \to -\boldsymbol{p}$ and $\varepsilon \to -\varepsilon$. In addition, upon using the quasiclassical approximation in equation 4.8, a negative sign is introduced due to the imaginary unit. The matrix structure then becomes

$$\hat{g}^R = \begin{pmatrix} g^R & f^R \\ -\tilde{f}^R & -\tilde{g}^R \end{pmatrix} \tag{4.11}$$

$$\hat{g}^A = \begin{pmatrix} g^A & f^A \\ -\tilde{f}^A & -\tilde{g}^A \end{pmatrix} \tag{4.12}$$

$$\hat{g}^K = \begin{pmatrix} g^K & f^K \\ \tilde{f}^K & \tilde{g}^K \end{pmatrix} \tag{4.13}$$

where the tilde notation symbolizes complex conjugation and the substitution $\varepsilon \to -\varepsilon$.

Quasiclassical approximation of a bulk superconductor

Continuing the analysis from section 3.5, the quasiclassical approximation to the solution is given by inserting equation 3.67 into equation 4.8.

$$\hat{g}^R(\varepsilon) = \frac{i}{\pi} \int d\xi \frac{1}{(\varepsilon + i\delta)^2 - \xi^2 - |\Delta|^2} \left[\varepsilon\hat{\rho}_3 + \xi\hat{I} + \hat{\Delta}\right] \tag{4.14}$$

where $\xi = \frac{1}{2m}\boldsymbol{p}^2 - \mu$. The poles have been shifted so that the inverse Fourier transform in time converges. Furthermore, it is noted that the ξ-proportional matrix cancels due to the symmetry. Performing the contour integral yields[53]

$$\hat{g}^R(\varepsilon) = \left[\frac{\text{sgn}(\varepsilon)}{\sqrt{\varepsilon^2 - |\Delta|^2}}\,\theta(\varepsilon^2 - \Delta^2) - \frac{i}{\sqrt{|\Delta|^2 - \varepsilon^2}}\,\theta(\Delta^2 - \varepsilon^2)\right]\left(\varepsilon\hat{\rho}_3 + \hat{\Delta}\right) \tag{4.15}$$

4.3 The dirty limit

The materials considered are assumed to be dirty. This means that there is a high density of impurities, which are randomly located throughout the material. A particle traveling through this material will scatter frequently off these impurities. This has an averaging effect on the direction of the particle, and it is reasonable that the greatest contribution to the Green function is spherically symmetric. Expanding $\check{g}$ in spherical harmonics, retaining only the spherically symmetric and the first directionally dependent component gives:

$$\check{g} \approx \check{g}_s + \hat{p}_F \cdot \check{\boldsymbol{g}}_p \tag{4.16}$$

where $\hat{p}_F$ is a unit vector denoting the direction of the Fermi momentum. The normalization condition of equation 4.10 then becomes

$$\check{g}_s^2 + \left\{\check{g}_s, \hat{p}_F \cdot \check{\boldsymbol{g}}_p\right\} + \left(\hat{p}_F \cdot \check{\boldsymbol{g}}_p\right)^2 = \check{I} \tag{4.17}$$

Neglecting second order terms in $\check{\boldsymbol{g}}_p$ gives the conditions

$$\begin{aligned} \check{g}_s^2 &= \check{I} \\ \left\{\check{g}_s, \hat{p}_F \cdot \check{\boldsymbol{g}}_p\right\} &= 0 \end{aligned} \tag{4.18}$$

Impurity scattering is included by replacing V_{imp} with equation (3.51). The quasiclassical version of this equation is found by the coordinate transformation $\xi_p = \frac{p^2}{2m}$. The integration measure becomes

$$d\boldsymbol{p} = p^2\, dp\, d\Omega_p = (2m)^{3/2}\xi_p^{1/2}\, d\xi_p\, d\Omega_p = 2\pi^2\hbar^3 N(\xi_p)\, d\xi_p\, d\Omega_p \approx 2\pi^2\hbar^3 N_0\, d\xi_p\, d\Omega_p \tag{4.19}$$

where $d\Omega_p$ is the angular integration measure and $N(\xi_p)$ is the density of states for a free electron gas. It may, in accordance with the quasiclassical approximation, be set equal to its value at the Fermi surface: $N(\xi_p) \approx N_0$. Furthermore, $u(\boldsymbol{p} - \boldsymbol{q}) \approx u(\boldsymbol{p} - \boldsymbol{q}_F)$. The quasiclassical version of the impurity scattered electrons then becomes

$$\Sigma(\boldsymbol{p}_F) = -i\pi n_{\text{imp}} N_0 \int \frac{d\Omega_p}{4\pi} |u(\boldsymbol{p}_F - \boldsymbol{q}_F)|^2 \check{g}(\boldsymbol{R}, T, \boldsymbol{q}_F, \varepsilon) \tag{4.20}$$

Inserting equation 4.16 into equation 4.20, the angular dependent component cancels, leaving only the spherically symmetric part:

$$\Sigma = -\frac{i\hbar}{2\tau}\check{g}_s \tag{4.21}$$

with the scattering time τ given as

$$\frac{\hbar}{\tau} = 2\pi n_{\text{imp}} N_0 \int \frac{d\Omega_p}{4\pi} |u(\boldsymbol{p}_F - \boldsymbol{q}_F)|^2 \tag{4.22}$$

Performing an angular average over the direction of the momentum at the Fermi surface, equation 4.9 becomes

$$i\hbar\frac{v_F}{3}\bar{\nabla}\check{\boldsymbol{g}}_p + \left[\varepsilon\check{\rho}_3 - \check{\Delta} - \check{\boldsymbol{\tau}}\cdot\boldsymbol{h}, \check{g}_s\right] = 0 \tag{4.23}$$

The terms which are odd in $\hat{p}_F$ are found by premultiplying with $\hat{p}_F$ and then repeating the angular average

$$i\hbar v_F\bar{\nabla}\check{g}_s + \left[\varepsilon\check{\rho}_3 + \frac{i\hbar}{2\tau}\check{g}_s - \check{\Delta} - \check{\boldsymbol{\tau}}\cdot\boldsymbol{h}, \check{\boldsymbol{g}}_p\right] \approx i\hbar v_F\bar{\nabla}\check{g}_s + \frac{i\hbar}{2\tau}\left[\check{g}_s, \check{\boldsymbol{g}}_p\right] = 0 \tag{4.24}$$

where it has assumed that impurity scattering dominates. The normalization condition in equation 4.18 and equation 4.24 may then be used to eliminate $\check{\boldsymbol{g}}_p$ from equation 4.23. Multiplying equation 4.24 by $\check{g}_s$ gives immediately

$$\check{\boldsymbol{g}}_p = -\tau v_F\check{g}_s\bar{\nabla}\check{g}_s \tag{4.25}$$

Inserting equation 4.25 into equation 4.23 yields the Usadel equation, which is the equation of motion sought after.

$$D\bar{\nabla}\left(\check{g}_s\bar{\nabla}\check{g}_s\right) + i\left[\varepsilon\check{\rho}_3 - \check{\Delta} - \check{\boldsymbol{\tau}}\cdot\boldsymbol{h}, \check{g}_s\right] = 0 \tag{4.26}$$

where the diffusion constant D is defined as

$$D = \frac{1}{3}\hbar v_F^2\tau \tag{4.27}$$

4.4 Boundary conditions

In the quasiclassical approximation, the treatment of boundary conditions at the interface between materials is a challenge. Such a transition occurs on a much smaller length scale than those for which the quasiclassical approximation is valid so it is necessary to use a different approach. Close to the boundary, on a length scale less than the elastic scattering length, the particle motion may be considered as ballistic. This is because the particle, upon entering the material, has not yet encountered any impurities. In the bulk of the sample, the quasiclassical approximation is valid, and so the motion must be diffusive. Between these two regions, on either side of an interface, there must therefore be an area where impurity scattering begins to take hold, converting ballistic motion into diffusive. This is called the isotropization zone. The model used to establish the boundary conditions is illustrated in figure 4.1.

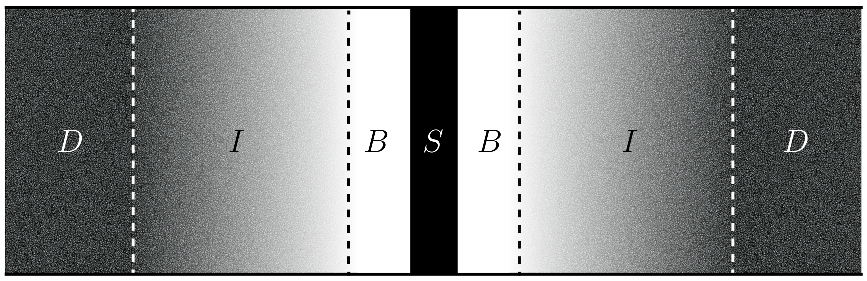


Figure 4.1: Illustration of the model used in the formulation of boundary conditions. The interface (scattering zone) is labeled as S, the ballistic zone as B, the isotropization zone as I and the diffusive zone as D.

The Hamiltonian in the ballistic zone close to the boundary is given as

$$\left[\varepsilon\hat{\rho}_3 + \frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} - H_\perp(\boldsymbol{\rho}) - V_b(x) + \hat{\Delta}(x) + \hat{\boldsymbol{\tau}}\cdot\boldsymbol{h}(x) - V_{\mathrm{imp}}(x)\check{I} + \mu\check{I}\right]\check{G}(\boldsymbol{r},\boldsymbol{r}') \\ = \delta(\boldsymbol{r}-\boldsymbol{r}')\check{I} \quad (4.28)$$

where it is assumed that the only dependence on the transversal coordinate $\boldsymbol{\rho}$ lies in $H_\perp$, which also includes a confinement potential. Parallel to the interface the quasiclassical approximation is valid, whereas in the direction perpendicular to the surface, i.e., in the x-direction, the exact Green function must be used. This means that the transversal variations of the self energies included are much slower than the variations along the x axis, which justifies neglecting the dependence on $\boldsymbol{\rho}$. In the following, spin-orbit coupling and external flux is neglected in the ballistic and isotropization zone, under the assumption that these terms will give a negligible contribution. They are, however, included in the diffusive zone. This is in correspondence with Ref.[12]. The boundary is assumed to lie in the yz-plane and the boundary potential is given by $V_b(x)$, which is zero outside of the boundary region. In the ballistic zone, the following field operators may be defined[60,61]

$$\phi^{\nu}_{\sigma}(\boldsymbol{r}) = \sum_n \frac{\chi_n(\boldsymbol{\rho})}{\sqrt{k_n}}\left(\psi^{\nu+}_{n\sigma}(x)e^{i\nu k_n x} + \psi^{\nu-}_{n\sigma}(x)e^{-i\nu k_n x}\right) = \sum_{nd}\frac{\chi_n(\boldsymbol{\rho})}{\sqrt{k_n}}\psi^{\nu d}_{n\sigma}(x)e^{id\nu k_n x} \quad (4.29)$$

It is seen that the motion may be split into a longitudinal term and a transversal term. The transversal motion is assumed to be quantized by the confinement potential, with wave vectors $\chi_n(\boldsymbol{\rho})$ that satisfy

$$H_\perp\chi_n(\boldsymbol{\rho}) = E_n\chi_n(\boldsymbol{\rho}) \quad (4.30)$$

where ρ is the transversal coordinate and n is the transversal channel. Furthermore, the confinement potential is assumed to be zero everywhere except for at a small layer by the sample edge, so that away from the edges

$$E_n = E_F - \frac{\hbar^2 k_n^2}{2m} \tag{4.31}$$

The operators ${\psi_{n\sigma}^{\nu d}}^\dagger(x)$ and $\psi_{n\sigma}^{\nu d}(x)$ are field operators that respectively create and annihilate a particle at location x, in channel n with spin σ, either in positive x-direction ($d = +1$) or negative ($d = -1$). In addition, electrons and holes move in opposite directions, thus introducing the factor $\nu = \pm 1$. These operators are by definition slowly varying, since they have been separated from the quickly oscillating exponential terms. Since the particles are fermions, the field operators satisfy the regular anticommutation relations

$$\begin{aligned} &\left\{ \left(\psi_{n\sigma}^{\nu d}\right)^\dagger (x), \psi_{m\sigma'}^{\nu d}(x') \right\} = \delta_{nm}\delta_{\sigma\sigma'}\delta(x-x') \\ &\left\{ \psi_{m\sigma'}^{\nu d}(x'), \psi_{m\sigma'}^{\nu d}(x') \right\} = 0 \end{aligned} \tag{4.32}$$

Equation 4.29 may be used to construct the ballistic Green function

$$\check{G}(\boldsymbol{r},\boldsymbol{r}') = \sum_{nmdd'} \tilde{G}_{nm}^{dd'}(\boldsymbol{r},\boldsymbol{r}') = \sum_{nmdd'} \tilde{G}_{nm}^{dd'}(x,x') \frac{\chi_n(\boldsymbol{\rho})\chi_m(\boldsymbol{\rho}')}{\sqrt{k_n k_m}} \tag{4.33}$$

The function $\check{G}(\boldsymbol{r},\boldsymbol{r}')$ is a matrix in spin⊗Nambu⊗Keldysh-space, and is given by the contributions from each channel and direction. The function $\tilde{G}(\boldsymbol{r},\boldsymbol{r}')$ is a matrix in spin⊗Nambu⊗Keldysh⊗direction⊗channel-space. In other words, each element $\tilde{G}_{nm}^{dd'}(\boldsymbol{r},\boldsymbol{r}')$ is a Keldysh matrix. The different matrix structures are listed below.

Direction:

$$\tilde{G}_{nm} = \begin{pmatrix} \tilde{G}_{nm}^{++} & \tilde{G}_{nm}^{+-} \\ \tilde{G}_{nm}^{-+} & \tilde{G}_{nm}^{--} \end{pmatrix} \tag{4.34}$$

Keldysh:

$$\tilde{G}_{nm}^{dd'} = \begin{pmatrix} \tilde{G}_{nm}^{Rdd'} & \tilde{G}_{nm}^{Kdd'} \\ 0 & \tilde{G}_{nm}^{Add'} \end{pmatrix} \tag{4.35}$$

Nambu:

$$\tilde{G}_{nm}^{Rdd'} = \begin{pmatrix} \tilde{G}_{nm}^{Rdd'++} \cdot e^{idk_n x - id'k_m x'} & \tilde{G}_{nm}^{Rdd'+-} \cdot e^{idk_n x + id'k_m x'} \\ \tilde{G}_{nm}^{Rdd'-+} \cdot e^{-idk_n x - id'k_m x'} & \tilde{G}_{nm}^{Rdd'--} \cdot e^{-idk_n x + id'k_m x'} \end{pmatrix} \tag{4.36}$$

Spin:

$$\tilde{G}_{nm}^{Rdd'\nu\nu'} = \begin{pmatrix} \tilde{G}_{nm\uparrow\uparrow}^{Rdd'\nu\nu'} & \tilde{G}_{nm\uparrow\downarrow}^{Rdd'\nu\nu'} \\ \tilde{G}_{nm\downarrow\uparrow}^{Rdd'\nu\nu'} & \tilde{G}_{nm\downarrow\downarrow}^{Rdd'\nu\nu'} \end{pmatrix} \tag{4.37}$$

From equation 4.33 it is seen that

$$H_\perp \check{G}(\boldsymbol{r},\boldsymbol{r}') = \sum_{nmdd'} E_n \tilde{G}^{dd'}_{nm}(\boldsymbol{r},\boldsymbol{r}') \tag{4.38}$$

Inserting equation 4.33 into equation 4.28 and evaluating it in the ballistic zone where $V_b(x) = 0$ gives

$$\sum_{dd'} \left[\rho_3\varepsilon + \frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} - E_n - \hat{\Delta}(x) + \hat{\boldsymbol{\tau}}\cdot\boldsymbol{h}(x) - V_{\text{imp}}(x)\check{I} + \mu\check{I}\right] \tilde{G}^{dd'}_{nm}(\boldsymbol{r},\boldsymbol{r}') \\ = \delta_{nm}\delta(\boldsymbol{r}-\boldsymbol{r}')\check{I} \tag{4.39}$$

The anticommutation relations in equation 4.32 have been used to be able to express equation 4.39 in terms of the individual channel channel contributions $\tilde{G}^{dd'}_{nm}$. The Hamiltonian is now independent of the transversal coordinate, which may be Wigner transformed into

$$\sum_{dd'} \left[\rho_3\varepsilon + \frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} - E_n - \hat{\Delta}(x) + \hat{\boldsymbol{\tau}}\cdot\boldsymbol{h}(x) - V_{\text{imp}}(x)\check{I} + \mu\check{I}\right] \tilde{G}^{dd'}_{nm}(x,x') = \delta_{nm}\delta(x-x')\check{I} \tag{4.40}$$

The Green function in equation 4.40 then also depends on the transversal center of mass coordinate $\rho_c = (\rho+\rho')/2$ but this is omitted for notational brevity. For a particular element of the retarded Green function in Nambu space, equation 4.40 takes the form

$$\sum_{dd'} \left[\rho_3\varepsilon + \frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} - E_n - \hat{\Delta}^{\nu\nu'}(x) + \hat{\boldsymbol{\tau}}\cdot\boldsymbol{h}(x) - V_{\text{imp}}(x)I + \mu I\right] \\ \times \tilde{G}^{dd'\nu\nu'}_{nm}(x,x')\cdot e^{id\nu k_n x - id'\nu' k_m x'} = \delta_{nm}\delta_{\nu\nu'}\delta(x-x')I \tag{4.41}$$

where I is the identity matrix in spin-space. The matrices $\tilde{G}^{dd'\nu\nu'}_{nm}$ are slowly varying, and so the second derivatives of x may be neglected, resulting in

$$\left[\rho_3\varepsilon + i\hbar d\nu v_n\frac{\partial}{\partial x} - \hat{\Delta}^{\nu\nu'}(x) + \hat{\boldsymbol{\tau}}\cdot\boldsymbol{h}(x) - V_{\text{imp}}(x)I\right]\tilde{G}^{dd'\nu\nu'}_{nm}(x,x') = 0, \quad x\neq x' \tag{4.42}$$

where the transversal energy E_n and the chemical potential $\mu \approx E_F$ have been canceled by the k_n^2-term produced by differentiation of the exponential function. The channel velocity is given by $v_n = k_n\hbar/m$. Repeating the procedure for the conjugate equation gives

$$\tilde{G}^{dd'\nu\nu'}_{nm}(x,x')\left[\rho_3\varepsilon - i\hbar d'\nu' v_m\frac{\partial}{\partial x'} - \hat{\Delta}^{\nu\nu'}(x') + \hat{\boldsymbol{\tau}}\cdot\boldsymbol{h}(x') - V_{\text{imp}}(x')I\right] = 0, \quad x'\neq x \tag{4.43}$$

Comparing equation 4.42 and 4.43 in the limit $x \to x'$ reveals that $\tilde{G}_{nm}^{dd'\nu\nu'}$ cannot be continuous in this point. Fortunately, the value of the discontinuity may be identified by fixating x' in equation 4.40 and integrating x over a small interval around it:

$$\int_{x'-\eta}^{x'+\eta} dx \sum_{dd'} \left[\rho_3 \varepsilon + \frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} - E_n - \hat{\Delta}(x) + \hat{\boldsymbol{\tau}} \cdot \boldsymbol{h}(x) - V_{\text{imp}}(x) \check{I} + \mu \check{I} \right] \tilde{G}_{nm}^{dd'}(x, x') = \delta_{nm} \check{I} \tag{4.44}$$

In the limit $\eta \to 0$ what remains is, due to the fundamental theorem of calculus

$$\sum_{dd'} \left[\left(\frac{\partial}{\partial x} \tilde{G}_{nm}^{dd'}(x, x') \right)_{x=x'+\eta} - \left(\frac{\partial}{\partial x} \tilde{G}_{nm}^{dd'}(x, x') \right)_{x=x'-\eta} \right] = \delta_{nm} \frac{2m}{\hbar^2} \check{I} \tag{4.45}$$

Furthermore, the total Green function must be continuous, giving the condition

$$\sum_{dd'} \left[\tilde{G}_{nm}^{dd'}(x+\eta, x) - \tilde{G}_{nm}^{dd'}(x-\eta, x) \right] = 0 \tag{4.46}$$

Evaluating equation 4.45 for each element in Nambu-space, where the exponential terms appear explicitly, gives

$$\begin{aligned} &\sum_{dd'} \left[\left(\frac{\partial}{\partial x} \tilde{G}_{nm}^{dd'\nu\nu'}(x, x') \right)_{x=x'+\eta} - \left(\frac{\partial}{\partial x} \tilde{G}_{nm}^{dd'\nu\nu'}(x, x') \right)_{x=x'-\eta} \right] e^{i(d\nu k_n - d'\nu' k_m)x} \\ &+ \sum_{dd'} \left[\tilde{G}_{nm}^{dd'}(x+\eta, x) - \tilde{G}_{nm}^{dd'}(x-\eta, x) \right] i d\nu k_n e^{i(d\nu k_n - d'\nu' k_m)x} \\ &= \delta_{mn} \delta_{\nu\nu'} \frac{2m}{\hbar^2} I \end{aligned}$$

The derivatives, $\frac{\partial}{\partial x} \tilde{G}_{nm}^{dd'\nu\nu'}$, are neglected under the assumption that they are much smaller than $k_n \tilde{G}_{nm}^{dd'\nu\nu'}$. For $\nu = \nu'$ and $n = m$, one gets for equation 4.45 and 4.46

$$\sum_{dd'} d \left[\tilde{G}_{nm}^{dd'}(x+\eta, x) - \tilde{G}_{nm}^{dd'}(x-\eta, x) \right] e^{i\nu k_n x (d-d')} = -i \frac{2\nu}{\hbar v_n} I \tag{4.47a}$$

$$\sum_{dd'} \left[\tilde{G}_{nm}^{dd'}(x+\eta, x) - \tilde{G}_{nm}^{dd'}(x-\eta, x) \right] = 0 \tag{4.47b}$$

Adding equation 4.47a and 4.47b gives

$$\begin{aligned} &\tilde{G}_{nn}^{++\nu\nu}(x+\eta, x) - \tilde{G}_{nn}^{++\nu\nu}(x-\eta, x) + \left(\tilde{G}_{nn}^{+-\nu\nu}(x+\eta, x) - \tilde{G}_{nn}^{+-\nu\nu}(x-\eta, x) \right) e^{2i\nu k_n x} \\ &= -\frac{i\nu}{\hbar v_n} I \end{aligned} \tag{4.48}$$

Equation 4.48 is supposed to yield a constant. However the Green functions are by design too slowly varying to compensate for the exponential function. This implies that in order for equation 4.48 to be valid, the $+-$-component must be continuous. The result then becomes

$$\tilde{G}^{++\nu\nu}_{nn}(x+\eta,x) - \tilde{G}^{++\nu\nu}_{nn}(x-\eta,x) = -\frac{i\nu}{\hbar v_n} I \tag{4.49}$$

By subtracting equation 4.47a and equation 4.47b on gets by similar arguments that the $-+$-term must be continous and that the $--$-term is given by

$$\tilde{G}^{--\nu\nu}_{nn}(x+\eta,x) - \tilde{G}^{--\nu\nu}_{nn}(x-\eta,x) = \frac{i\nu}{\hbar v_n} I \tag{4.50}$$

Summarizing, the discontinuity at $x = x'$ for an element in Nambu-space becomes

$$\tilde{G}^{dd'\nu\nu'}_{nm}(x+\eta,x) - \tilde{G}^{dd'\nu\nu'}_{nm}(x-\eta,x) = -\delta_{dd'}\delta_{\nu\nu'}\delta_{nm}\frac{id\nu}{\hbar v_n} I \tag{4.51}$$

Equation 4.51 enables the definition of a Green function $\tilde{C}^{dd'\nu\nu'}_{nm}$ which is continuous, since the discontinuity can be explicitly accounted for.

$$\tilde{G}^{dd'\nu\nu'}_{nm}(x,x') = -\frac{i}{2\pi\hbar\sqrt{v_n v_m}}\tilde{C}^{dd'\nu\nu'}_{nm}(x,x') - \delta_{dd'}\delta_{\nu\nu'}\delta_{nm}\frac{id\nu}{2\pi\hbar v_n}\mathrm{sgn}(x-x') I \tag{4.52}$$

The factor in front of $\tilde{C}^{dd'\nu\nu'}_{nm}$ is added to simplify a later result, while also making sure that channel n does not receive a different weight than channel m, which would be unphysical. Moving away from the interface, and into the isotropization zone, the impurity scattering term is given by equation 4.21, where the Green function is approximated to the value at the beginning of the diffusive zone, where the quasiclassical approximation is valid. This is reasonable because the quasiclassical Green function varies on scales much larger than the size of the isotropization zone. Furthermore, impurity scattering dominates the energy ε, the gap parameter Δ and the exchange field $\boldsymbol{h}$, which are all on the same scale, so that these terms may be neglected. The resulting equations in the isotropization zone becomes

$$\left[\frac{\partial}{\partial x} - \frac{d\nu}{2\tau v_n}\check{g}\right]\tilde{G}^{dd'\nu\nu'}_{nm}(x,x') = 0, \quad x \neq x' \tag{4.53a}$$

$$\tilde{G}^{dd'\nu\nu'}_{nm}(x,x')\left[\frac{\partial}{\partial x'} + \frac{d'\nu'}{2\tau v_m}\check{g}\right] = 0, \quad x' \neq x \tag{4.53b}$$

Differentiating equation 4.53a and 4.53b in terms of x and x' respectively, inserting equation 4.53a and 4.53b, and subtracting gives the following partial differential equation

$$\left(\frac{\partial^2}{\partial x^2}-\frac{\partial^2}{\partial x'^2}\right)\tilde{G}^{dd'\nu\nu'}_{nm}(x,x')=\left(\frac{1}{4\tau^2 v_n^2}-\frac{1}{4\tau^2 v_m^2}\right)\tilde{G}^{dd'\nu\nu'}_{nm}(x,x') \tag{4.54}$$

Equation 4.54 is readily solved by using separation of variables, with the requirement that the Green function at the beginning of the isotropization zone, at x_b, should be equal the ballistic Green function. This results in

$$\begin{aligned}\tilde{G}^{dd'\nu\nu'}_{nm}(x,x') = \frac{1}{4}&\left[\left(I-d\nu g^{\nu\nu'}\right)e^{\frac{x-x_b}{2\tau v_n}}+\left(I+d\nu g^{\nu\nu'}\right)e^{-\frac{x-x_b}{2\tau v_n}}\right]\\ \times&\left(-\frac{i}{2\pi\hbar\sqrt{v_n v_m}}\tilde{C}^{dd'\nu\nu'}_{nm}(x_b,x_b)-\delta_{dd'}\delta_{\nu\nu'}\delta_{nm}\frac{id\nu}{2\pi\hbar v_n}\mathrm{sgn}(x-x')I\right)\\ \times&\left[\left(I+d'\nu' g^{\nu\nu'}\right)e^{\frac{x'-x_b}{2\tau v_m}}+\left(I-d'\nu' g^{\nu\nu'}\right)e^{-\frac{x'-x_b}{2\tau v_m}}\right]\end{aligned} \tag{4.55}$$

Equation 4.55 will be used to express the quasiclassical Green functions g in terms of the ballistic Green functions $\tilde{C}^{dd'}_{nm}$. This is done by requiring that $\tilde{G}^{dd'\nu\nu'}$ decreases when moving towards the diffusive zone. On the left side of an interface, inspection of equation 4.55 uncovers the following conditions for the Green function to remain finite as $x \to -\infty$ and $x' \to -\infty$ respectively:

$$\left(\nu\Sigma_z^{(d)}+g_1\right)\left(\tilde{C}_1-\nu\Sigma_z^{(d)}\right)=0 \tag{4.56a}$$

$$\left(\tilde{C}_1+\nu\Sigma_z^{(d)}\right)\left(\nu\Sigma_z^{(d)}-g_1\right)=0 \tag{4.56b}$$

where $\tilde{C}_1$ is short-hand for $\tilde{C}^{dd\nu\nu}_{nn,1}$ and g_1 for $g_1^{\nu\nu}$. On the right hand side, the equivalent conditions become

$$\left(\nu\Sigma_z^{(d)}-g_2\right)\left(\tilde{C}_2+\nu\Sigma_z^{(d)}\right)=0 \tag{4.57a}$$

$$\left(\tilde{C}_2+\nu\Sigma_z^{(d)}\right)\left(\nu\Sigma_z^{(d)}-g_2\right)=0 \tag{4.57b}$$

A lower index of 1 indicates Green functions to the left of the interface, and an index of 2 indicates Green functions to the right. Equations 4.56 and 4.57 have been expanded to direction-space, with $\nu\Sigma_z^{(d)}$ the Pauli-matrix in z-direction, which has the parameter $d\delta_{dd'}$ as elements. It is important to note that the quasiclassical Green functions g have no structure in the direction space, in contrast to the ballistic Green function $\tilde{C}$. This means that a matrix that only has structure in direction space will commute with g, but not with $\tilde{C}$. It is observed that equations 4.56 and 4.57 are diagonal in direction$\otimes$Nambu$\otimes$channel-space. This is due to the discontinuity condition in equation 4.51 which only gives a contribution to the continuous Green function $\tilde{C}$ on the diagonal. The off-diagonal terms produce the trivial result of $\tilde{C}=0$.

The ballistic Green function on the right hand side may be expressed in terms of the ballistic Green function on the left hand side by means of the transfer matrix M:

$$\tilde{C}_2 = M\tilde{C}_1 M^\dagger \tag{4.58}$$

Multiplying equation 4.56a by g_1 from the left and equation 4.57a by $g_1 M^\dagger$ from the left, and by $\left(M^\dagger\right)^{-1}$ from the right, then subtracting the two equations gives

$$C_1 = \left(I + g_1 M^\dagger g_2 M\right)^{-1} \left[2g_1 + \left(I - g_1 M^\dagger g_2 M\right) \nu\Sigma_z^{(d)}\right] \tag{4.59}$$

where it has been used that $M^\dagger \Sigma_z^{(d)} M = \Sigma_z^{(d)}$ owing to flux conservation[62]. There is now one step remaining before the boundary conditions for the Usadel equation may be defined. Having expressed the ballistic Green functions in terms of the quasiclassical Green functions, it is now necessary to relate the ballistic Green functions to a quantity which is conserved across the interface. This is achieved by considering the matrix current, which is defined as[63,60]

$$\check{I}_M(x,\varepsilon) = \lim_{\boldsymbol{r}'\to\boldsymbol{r}} \frac{e^2\hbar}{m} \int d\boldsymbol{\rho} \left(\frac{\partial}{\partial x} - \frac{\partial}{\partial x'}\right) \check{G}(\boldsymbol{r},\boldsymbol{r}') \tag{4.60}$$

With the integrand taking the form of a current density, the integral over the transverse coordinate $\boldsymbol{\rho}$ yields the net matrix current through the surface. Inserting equation (4.33) and equation (4.52), and using the orthonormality of the transverse wave vectors $\chi_n(\rho)$, this becomes

$$\check{I}_M^{\nu\nu'}(x,\varepsilon) = \frac{e^2}{2\pi\hbar} \sum_{ndd'} (\nu d + \nu' d') \tilde{C}_{nn}^{dd'\nu\nu'}(x,x) \tag{4.61}$$

In the isotropization zones, $\nu = \nu'$ and $d = d'$, thus giving

$$\check{I}_M^{\nu\nu'}(x,\varepsilon) = G_q \,\mathrm{Tr}_{n,d} \left[\nu\Sigma_z^{(d)} \tilde{C}^{\nu\nu'}(x,x)\right] \tag{4.62}$$

where $G_q = \frac{e^2}{\pi\hbar}$ is the conductance quantum. Using that $\Sigma_z^{(d)} = M\Sigma_z^{(d)} M^\dagger$ it is seen that

$$\begin{aligned}
\check{I}^{\nu\nu'}_{M,1} &= G_q \operatorname{Tr}_{n,d}\left[\nu\Sigma^{(d)}_z \tilde{C}^{\nu\nu'}_1\right] \\
&= G_q \operatorname{Tr}_{n,d}\left[\nu M\Sigma^{(d)}_z M^\dagger \tilde{C}^{\nu\nu'}_1\right] \\
&= G_q \operatorname{Tr}_{n,d}\left[\nu\Sigma^{(d)}_z M^\dagger \tilde{C}^{\nu\nu'}_1 M\right] \\
&= G_q \operatorname{Tr}_{n,d}\left[\nu\Sigma^{(d)}_z \tilde{C}^{\nu\nu'}_2\right] \\
&= \check{I}^{\nu\nu'}_{M,2}
\end{aligned} \tag{4.63}$$

This shows that the matrix current is indeed conserved across the interface. This is not the case for spin-active boundary conditions, but suitable conditions may still be derived. This was recently done by Eschrig *et al* using a scattering matrix approach[64]. By insertion of equation (4.55) into equation (4.62) it is found that the matrix current is conserved also in the isotropization zone[60]. Furthermore, the matrix current is in the quasiclassical approximation given as[63]

$$\check{I}_{M,j} = \sigma_j A_j \check{g}\nabla\check{g}\cdot\boldsymbol{n} = \frac{L_j}{R_j}\check{g}\nabla\check{g}\cdot\boldsymbol{n} \tag{4.64}$$

where A_j is the cross section through which the current flows and σ_j is the conductance on side j of the interface. The unit vector $\boldsymbol{n}$ is the normal vector of the interface. Here, $\boldsymbol{n}$ is pointing along the x-axis. The bulk resistance of the material considered on side j, is given as $R_j = \frac{L_j}{\sigma_j A_j}$, where L_j is the length of the material in the direction of $\boldsymbol{n}$. Equations (4.59), (4.62) and (4.64) sufficiently define the boundary conditions.

A special case where the boundary conditions take on a particularly easy form is when the transfer matrix only has structure in direction space. This is the case when the boundary is spin-independent as well as identical both for particles and holes. In this case, the transfer matrix M commutes with the quasiclassical Green functions g_1 and g_2 so that they only appear as products $Q = M^\dagger M$. Furthermore, the inverse of Q may be found by

$$I = \Sigma^{(d)}_z\Sigma^{(d)}_z = M\Sigma^{(d)}_z M^\dagger M\Sigma^{(d)}_z M^\dagger = Q\Sigma^{(d)}_z Q\Sigma^{(d)}_z \Rightarrow Q^{-1} = \Sigma^{(d)}_z Q\Sigma^{(d)}_z \tag{4.65}$$

where the last equality is found by multiplying with $M^\dagger$ from the left and $\left(M^\dagger\right)^{-1}$ from the right. This further implies that $\det(Q) = 1$ and that the diagonal elements are identical. Another useful identity is

$$(I + Qg_1g_2)(Q^{-1} + g_2g_1) = \{g_1, g_2\} + Q + Q^{-1} \tag{4.66}$$

Because of equation 4.65, $Q + Q^{-1}$ is diagonal in direction space, and so is the anticommutator. Thus, the inverse of the right hand side of equation 4.66 is particularly easy. Using this relation with equation 4.59 and inserting it into equation 4.62 gives

$$I(x,\varepsilon) = G_q \operatorname{Tr}_{n,d}\left[\frac{(2g_1Q+g_2)\nu\Sigma_z^{(d)} + [g_2,g_1] + Q^{-1} - Q}{\{g_2,g_1\} + Q + Q^{-1}}\right] \tag{4.67}$$

The term $Q^{-1} - Q$ has zero on the diagonals, and may be omitted. Furthermore, $\Sigma_z^{(d)}$-term cancels when taking the trace. It is customary to express the boundary condition in terms of the scattering eigenvalues T_n. It is related to the elements of the matrix Q, q_{ij} via[65]

$$q_{11} + q_{22} = \frac{4}{T_n} - 2 \tag{4.68}$$

Performing the trace, and using equation 4.64, the boundary conditions become

$$\frac{L_j}{R_j}\check{g}\nabla\check{g} = 4G_q \sum_n \frac{T_n\left[\check{g}_2,\check{g}_1\right]}{4\check{I} + T_n\left(\{\check{g}_2,\check{g}_1\} - 2\check{I}\right)} \tag{4.69}$$

Since the Nambu space dependent parameter ν canceled, equation 4.69 have been expanded to Keldysh space. These are the Nazarov boundary conditions[63]. A simpler set of boundary conditions may be found in the tunneling limit, $T_n \ll 1$

$$\check{g}\nabla\check{g} = \frac{1}{L_j\zeta_j}\left[\check{g}_2,\check{g}_1\right] \tag{4.70}$$

where $\zeta_j = \frac{R_I}{R_j}$ and $R_I = \left(G_q \sum_n T_n\right)^{-1}$ is the interface resistance. In other words, the parameter ζ describes physically the ratio between the interface and bulk resistance. These are the Kupriyanov-Lukichev boundary conditions[66].

4.5 Observables

4.5.1 Density of states

The density of states $N(\boldsymbol{R},\varepsilon)$ is found directly from the spectral function $A(\varepsilon)$[67], so that for a particular spin component σ

$$\begin{aligned}
N_\sigma(\boldsymbol{R},\varepsilon) &= \lim_{\boldsymbol{r}\to 0} -\frac{1}{\pi}\,\mathrm{Im}\,G^R_{\sigma\sigma}(\boldsymbol{R},\boldsymbol{r},\varepsilon) \\
&= \lim_{\boldsymbol{r}\to 0} -\frac{1}{\pi}\,\mathrm{Im}\int \frac{d\boldsymbol{p}}{(2\pi\hbar)^3} e^{i\boldsymbol{p}\cdot\boldsymbol{r}/\hbar} G^R_{\sigma\sigma}(\boldsymbol{R},\boldsymbol{p},\varepsilon) \\
&= -\frac{1}{\pi}\,\mathrm{Im}\int \frac{d\boldsymbol{p}}{(2\pi\hbar)^3} G^R_{\sigma\sigma}(\boldsymbol{R},\boldsymbol{p},\varepsilon) \\
&= N_0\,\mathrm{Re}\,g^R_{\sigma\sigma}(\boldsymbol{R},\varepsilon)
\end{aligned} \tag{4.71}$$

where equation 4.8 and 4.19 has been used. When information about separate spin components is unnecessary, the average in spin space may be taken:

$$N(\boldsymbol{R},\varepsilon) = \frac{1}{2}N_0\,\mathrm{Re}\,\mathrm{Tr}\left[g^R(\boldsymbol{R},\varepsilon)\right] \tag{4.72}$$

Density of states in a bulk superconductor

Applying equation 4.72 to equation 4.15 gives

$$N(\varepsilon) = N_0 \frac{|\varepsilon|}{\sqrt{\varepsilon^2 - |\Delta|^2}}\, \theta(\varepsilon^2 - |\Delta|^2) \tag{4.73}$$

The density of states $N(\varepsilon)$ is plotted in figure 4.2, where the energy gap mentioned in section 2.1 is clearly seen. In addition, two pronounced peaks appear at the points $\varepsilon = \pm\Delta$, where $N(\varepsilon)$ formally diverges. This can be thought of as a consequence of all the particles with energies lower than Δ being "pushed out" to where $\varepsilon > \Delta$.

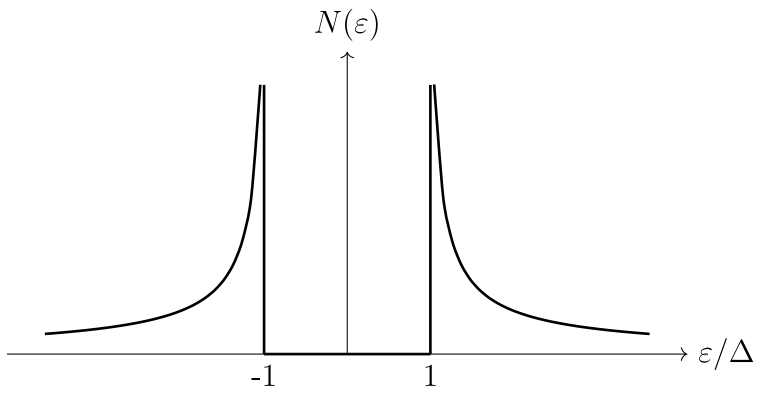


Figure 4.2: Density of states for a bulk superconductor.

4.5.2 Currents

In the second quantization formalism, the charge density is given as

$$\rho_e(\boldsymbol{r}) = e\langle n(\boldsymbol{r})\rangle = e\sum_\sigma \langle \psi_\sigma^\dagger(\boldsymbol{r})\psi_\sigma(\boldsymbol{r})\rangle \tag{4.74}$$

The charge current density must satisfy the continuity equation

$$\frac{\partial}{\partial t}\rho_e(\boldsymbol{r}) + \nabla \cdot \boldsymbol{J}_e(\boldsymbol{r}) = 0 \tag{4.75}$$

The time derivative is found by using the Heisenberg equation, resulting in

$$\begin{aligned}\frac{\partial}{\partial t}\rho_e(\boldsymbol{r}) &= e\sum_\sigma\langle\frac{\partial}{\partial t}\psi^\dagger_\sigma(\boldsymbol{r})\psi_\sigma(\boldsymbol{r})+\psi^\dagger_\sigma(\boldsymbol{r})\frac{\partial}{\partial t}\psi_\sigma(\boldsymbol{r})\rangle\\ &= -\frac{ie\hbar}{2m}\sum_\sigma\left\langle\left(\nabla+\frac{i}{\hbar}\boldsymbol{A}(\boldsymbol{r})\right)^2\psi^\dagger_\sigma(\boldsymbol{r})\psi_\sigma(\boldsymbol{r})-\psi^\dagger_\sigma(\boldsymbol{r})\left(\nabla-\frac{i}{\hbar}\boldsymbol{A}(\boldsymbol{r})\right)^2\psi_\sigma(\boldsymbol{r})\right\rangle\\ &= -\nabla\cdot\frac{ie\hbar}{2m}\sum_\sigma\left\langle\left(\nabla+\frac{i}{\hbar}\boldsymbol{A}(\boldsymbol{r})\right)\psi^\dagger_\sigma(\boldsymbol{r})\psi_\sigma(\boldsymbol{r})-\psi^\dagger_\sigma(\boldsymbol{r})\left(\nabla-\frac{i}{\hbar}\boldsymbol{A}(\boldsymbol{r})\right)\psi_\sigma(\boldsymbol{r})\right\rangle\end{aligned} \tag{4.76}$$

The charge current may then be identified as

$$\boldsymbol{J}_e(\boldsymbol{r}) = \frac{ie\hbar}{2m}\sum_\sigma\left\langle\left(\nabla+\frac{i}{\hbar}\boldsymbol{A}(\boldsymbol{r})\right)\psi^\dagger_\sigma(\boldsymbol{r})\psi_\sigma(\boldsymbol{r})-\psi^\dagger_\sigma(\boldsymbol{r})\left(\nabla-\frac{i}{\hbar}\boldsymbol{A}(\boldsymbol{r})\right)\psi_\sigma(\boldsymbol{r})\right\rangle \tag{4.77}$$

To reformulate this in the language of Green function, the coordinates $(\boldsymbol{r}, t)$ are split into two different coordinate pairs $(\boldsymbol{r}_1, t_1)$ and $(\boldsymbol{r}_2, t_2)$, with equation 4.77 emerging in the limit where they are equal. The introduction of the limit gives a freedom as to which coordinates to assign to which functions, and for a particular choice, the expression for the current becomes

$$\begin{aligned}\boldsymbol{J}_e(\boldsymbol{R}) &= \lim_{\boldsymbol{r}_1,\boldsymbol{r}_2\to\boldsymbol{R}}\frac{ie\hbar}{2m}\sum_\sigma\left\langle\left(\nabla_2+\frac{i}{\hbar}\boldsymbol{A}(\boldsymbol{r_2})\right)\psi^\dagger_\sigma(\boldsymbol{r_2})\psi_\sigma(\boldsymbol{r_1})-\psi^\dagger_\sigma(\boldsymbol{r_2})\left(\nabla_1-\frac{i}{\hbar}\boldsymbol{A}(\boldsymbol{r_1})\right)\psi_\sigma(\boldsymbol{r_1})\right\rangle\\ &= \lim_{\boldsymbol{r}_1,\boldsymbol{r}_2\to\boldsymbol{R}}-\frac{ie\hbar}{2m}\sum_\sigma\left\{\nabla_1-\nabla_2-\frac{i}{\hbar}\left(\boldsymbol{A}(\boldsymbol{r}_1)+\boldsymbol{A}(\boldsymbol{r}_2)\right)\right\}\langle\psi^\dagger_\sigma(\boldsymbol{r}_2)\psi_\sigma(\boldsymbol{r}_1)\rangle\end{aligned} \tag{4.78}$$

The expectation value is related to the Green functions in the following manner, due to the anticommuation relations:

$$\left\langle\psi^\dagger_\sigma(\boldsymbol{r}_2)\psi_\sigma(\boldsymbol{r}_1)\right\rangle = \frac{1}{2}\left\langle\left[\psi^\dagger_\sigma(\boldsymbol{r}_2),\psi_\sigma(\boldsymbol{r}_1)\right]\right\rangle+\frac{1}{2}\langle\delta(\boldsymbol{r}_1-\boldsymbol{r}_2)\rangle = -\frac{i}{2}G^K_{\sigma\sigma}(\boldsymbol{r}_1,\boldsymbol{r}_2)+\frac{1}{2}\langle\delta(\boldsymbol{r}_1-\boldsymbol{r}_2)\rangle \tag{4.79}$$

The charge current in the Keldysh formalism thus becomes

$$\begin{aligned}\boldsymbol{J}_e(\boldsymbol{R}) = \lim_{\boldsymbol{r}_1,\boldsymbol{r}_2\to\boldsymbol{R}}-\frac{e\hbar}{4m}\operatorname{Tr}\Big\{&\left(\nabla_1-\nabla_2-\frac{i}{\hbar}\left(\boldsymbol{A}(\boldsymbol{r}_1)+\boldsymbol{A}(\boldsymbol{r}_2)\right)\right)\\ &\times\left(G^K(\boldsymbol{r}_1,\boldsymbol{r}_2)+\frac{i}{4}I\langle\delta(\boldsymbol{r}_1-\boldsymbol{r}_2)\rangle\right)\Big\}\end{aligned} \tag{4.80}$$

It is observed that equation 4.80 diverges in the limit $\boldsymbol{r}_1 \to \boldsymbol{r}_2$. This is to be expected as the separation into $(\boldsymbol{r}_1, t)$ and $(\boldsymbol{r}_2, t)$ in equation 4.76 allowed the operators to be extracted from the expectation value. Setting $\boldsymbol{r}_1 = \boldsymbol{r}_2$ does not represent the original equation. The limit must then be thought of as allowing $\boldsymbol{r}_1$ to approach $\boldsymbol{r}_2$ arbitrarily close without becoming identical. In this case, the δ-function gives no contribution and the expression remains finite. This may be written as

$$\boldsymbol{J}_e(\boldsymbol{R}) = \lim_{\boldsymbol{r}_1, \boldsymbol{r}_2 \to \boldsymbol{R}} -\frac{e\hbar}{4m} \operatorname{Tr}\left\{ \left(\nabla_1 - \nabla_2 - \frac{i}{\hbar}\left(\boldsymbol{A}(\boldsymbol{r}_1) + \boldsymbol{A}(\boldsymbol{r}_2)\right)\right) G^K(\boldsymbol{r}_1, \boldsymbol{r}_2) \right\} \tag{4.81}$$

The quasiclassical form of equation 4.81 is found by expressing $G^K(\boldsymbol{r}_1, \boldsymbol{r}_2)$ as the inverse Fourier transformed in the relative coordinates $(\boldsymbol{r}, t)$.

$$\begin{aligned}
\boldsymbol{J}_e(\boldsymbol{R}) &= \lim_{\boldsymbol{r}\to 0} -\frac{e\hbar}{4m} \operatorname{Tr}\left\{ \left(2\nabla_r - \frac{i}{\hbar}\left(\boldsymbol{A}(\boldsymbol{R} + \frac{1}{2}\boldsymbol{r}) + \boldsymbol{A}(\boldsymbol{R} - \frac{1}{2}\boldsymbol{r}) \right) \right) \right. \\
&\qquad \left. \times \int \frac{d\boldsymbol{p} d\varepsilon}{(2\pi\hbar)^4} e^{(i\boldsymbol{p}\cdot\boldsymbol{r} - i\varepsilon t)/\hbar} G^K(\boldsymbol{R}, T, \boldsymbol{p}, \varepsilon) \right\} \\
&= -\frac{e}{4\pi m} \int d\varepsilon \operatorname{Tr}\left\{ \int \frac{d\boldsymbol{p}}{(2\pi\hbar)^3} \left(\frac{i}{\hbar}\boldsymbol{p} - \frac{i}{\hbar}\boldsymbol{A}(\boldsymbol{R}) \right) G^K(\boldsymbol{R}, T, \boldsymbol{p}, \varepsilon) \right\} \\
&= -\frac{e}{4m\hbar} N_0 \int d\varepsilon \operatorname{Tr}\left\{ \int \frac{d\Omega}{4\pi} \left(\boldsymbol{p}_F - \boldsymbol{A}(\boldsymbol{R})\right) g^K(\boldsymbol{R}, T, \boldsymbol{p}, \varepsilon) \right\} \\
&= \frac{N_0 e D}{4\hbar} \int d\varepsilon \operatorname{Tr}\left\{ (g\bar{\nabla} g)^K \right\} + \frac{N_0 e}{4m\hbar} \int d\varepsilon \operatorname{Tr}\left\{ \boldsymbol{A} g^K \right\}
\end{aligned} \tag{4.82}$$

In going to the last line of equation (4.82), equations (4.16) and (4.25) have been used. The derivation of equation (4.82) has taken place in spin space and is valid for particles. Expanding to Nambu⊗spin-space, the current of holes is also required. The hole-current flows in the opposite direction as the particle current, and so the expression for the total charge current density of particles and holes becomes

$$\boldsymbol{J}_e = \frac{N_0 e D}{4\hbar} \int d\varepsilon \operatorname{Tr}\left\{ \hat{\rho}_3 (\check{g}\bar{\nabla}\check{g})^K \right\} + \frac{N_0 e}{4m\hbar} \int d\varepsilon \operatorname{Tr}\left\{ \hat{\rho}_3 \hat{\boldsymbol{A}} \hat{g}^K \right\} \tag{4.83}$$

In equation (4.83), the term including $\hat{\boldsymbol{A}}$, called the diamagnetic term, does not give a contribution to the current for the systems considered. The reason for this is that this term is canceled by the high-energy contribution to the quasiclassical ξ-integral[50,67]. No such cancellation is found here because the high-energy part of the integration contour is ignored. However, the removal of the $\hat{\boldsymbol{A}}$-term can be made plausible by the following argument. By performing the trace in Nambu space the integral, denoted I_A, becomes

$$
\begin{aligned}
I_A &= \int_{-\infty}^{\infty} d\varepsilon \,\mathrm{Tr}\left\{\boldsymbol{A} g^K + \boldsymbol{A}^* \tilde{g}^K\right\} \\
&= \int_{-\infty}^{\infty} d\varepsilon \,\mathrm{Tr}\left\{\boldsymbol{A} g^K(\varepsilon)\right\} + \int_{-\infty}^{\infty} \mathrm{Tr}\left\{\boldsymbol{A}^* g^{K*}(-\varepsilon)\right\} \\
&= \int_{-\infty}^{\infty} d\varepsilon \,\mathrm{Tr}\left\{\boldsymbol{A} g^K(\varepsilon) - \boldsymbol{A}^* g^{K*}(\varepsilon)\right\} \\
&= 2i \int_{-\infty}^{\infty} d\varepsilon \,\mathrm{Im}\,\mathrm{Tr}\left\{\boldsymbol{A} g^K\right\}
\end{aligned}
$$

where equation (4.13) has been used, as well as the definition of tilde-conjugation. The transformation $\varepsilon \to -\varepsilon$ was introduced in the second term. The conclusion is that the diamagnetic term gives an imaginary contribution to the current, which is not physical. The final expression for the charge current therefore is

$$
J_e(\boldsymbol{R}) = \frac{N_0 e D}{4\hbar} \int d\varepsilon \,\mathrm{Tr}\left\{\hat{\rho}_3 (\check{g}\bar{\nabla}\check{g})^K\right\} \tag{4.84}
$$

The particles and holes referred to are of course electrons and absence of electrons. Since the electrons have spin, it is possible to define a spin current. The charge current $\boldsymbol{J}_e$ does not yield any information about the transport of spin. The starting point from which to find the spin current is the spin density

$$
\boldsymbol{\rho}_s = \frac{\hbar}{2} \sum_{\sigma\sigma'} \langle \psi_\sigma^\dagger \boldsymbol{\tau}_{\sigma\sigma'} \psi_{\sigma'} \rangle \tag{4.85}
$$

The spin density is a vector, and so the spin current is a tensor. In a normal metal with no spin-orbit coupling, a continuity equation may be established for each component of the spin density and the derivation of the spin current follows analogously to the charge current, resulting in

$$
\boldsymbol{J}_s^x = \frac{1}{8} N_0 D \int d\varepsilon \,\mathrm{Tr}\left\{\hat{\rho}_3 \hat{\tau}_x (\check{g}\bar{\nabla}\check{g})^K\right\} \tag{4.86}
$$

$$
\boldsymbol{J}_s^y = \frac{1}{8} N_0 D \int d\varepsilon \,\mathrm{Tr}\left\{\hat{\rho}_3 \hat{\tau}_y (\check{g}\bar{\nabla}\check{g})^K\right\} \tag{4.87}
$$

$$
\boldsymbol{J}_s^z = \frac{1}{8} N_0 D \int d\varepsilon \,\mathrm{Tr}\left\{\hat{\rho}_3 \hat{\tau}_z (\check{g}\bar{\nabla}\check{g})^K\right\} \tag{4.88}
$$

where the physical interpretation of the z-component is particularly easy to interpret: $\boldsymbol{J}_s^z \propto \boldsymbol{J}_{e\uparrow} - \boldsymbol{J}_{e\downarrow}$. The difference between the contribution from the opposite spins is found, rather than their sum. This means that $\boldsymbol{J}_s^z$ is zero if the current contains equal amounts of spin-$\uparrow$ and spin-$\downarrow$, and non-zero otherwise. Obviously this then gives the spin transport. For a ferromagnet it is more difficult, and the contribution from the exchange field to $\frac{\partial}{\partial t}\boldsymbol{\rho}_s$ will be investigated. Indeed,

$$\frac{\partial \boldsymbol{\rho}_s}{\partial t} = \frac{\hbar}{2}\sum_{\sigma\sigma'} \boldsymbol{\tau}_{\sigma\sigma'} \left\langle \frac{\partial \psi_\sigma^\dagger}{\partial t}\psi_{\sigma'} + \psi_\sigma^\dagger \frac{\partial \psi_{\sigma'}}{\partial t} \right\rangle \tag{4.89}$$

Using the Heisenberg equation, and inserting equation 3.28 gives

$$\begin{aligned} \left(\frac{\partial \rho_s}{\partial t}\right)_{ex} &= \frac{1}{2i}\sum_{\sigma\sigma'\sigma''} \tau_{\sigma\sigma'} \left\langle -\psi_{\sigma''}^\dagger \boldsymbol{h}\cdot\boldsymbol{\tau}_{\sigma''\sigma}\psi_{\sigma'} + \psi_\sigma^\dagger \boldsymbol{h}\cdot\boldsymbol{\tau}_{\sigma'\sigma''}\psi_{\sigma'} \right\rangle \\ &= \frac{1}{2i}\sum_{\sigma\sigma'} \left\langle -\psi_\sigma^\dagger \left[(\boldsymbol{h}\cdot\boldsymbol{\tau})\boldsymbol{\tau}\right]_{\sigma\sigma'}\psi_{\sigma'} + \psi_\sigma^\dagger \left[\boldsymbol{\tau}(\boldsymbol{h}\cdot\boldsymbol{\tau})\right]_{\sigma\sigma'}\psi_{\sigma'} \right\rangle \end{aligned} \tag{4.90}$$

Using that $(\boldsymbol{h}\cdot\boldsymbol{\tau})\boldsymbol{\tau} = \boldsymbol{h} - i\boldsymbol{h}\times\boldsymbol{\tau}$ and that $\boldsymbol{\tau}(\boldsymbol{h}\cdot\boldsymbol{\tau}) = \boldsymbol{h} + i\boldsymbol{h}\times\boldsymbol{\tau}$ one gets

$$\left(\frac{\partial \boldsymbol{\rho}_s}{\partial t}\right)_{ex} = \sum_{\sigma\sigma'} \langle \psi_\sigma^\dagger \left[\boldsymbol{h}\times\boldsymbol{\tau}\right]_{\sigma\sigma'}\psi_{\sigma'}\rangle = \boldsymbol{h}\times\boldsymbol{\rho}_s \tag{4.91}$$

This means that the continuity equation for a given component α, gets an additional term:

$$\frac{\partial \boldsymbol{\rho}_s^\alpha}{\partial t} + \nabla\cdot\boldsymbol{J}_s^\alpha + \left(\frac{\partial \boldsymbol{\rho}_s^\alpha}{\partial t}\right)_{ex} = 0 \tag{4.92}$$

The extra term is the spin torque, which means that the spin current is in general not conserved. However, due to the cross product in equation 4.91 it is seen that there is no spin torque for the component α that is parallel to the exchange field $\boldsymbol{h}$. Therefore, this component of the spin current is conserved even in a ferromagnet. Spin-orbit coupling also influences the spins, making the spin current ill-defined by an additional term in the continuity equation

$$\frac{\partial \boldsymbol{\rho}_s^\alpha}{\partial t} + \nabla\cdot\boldsymbol{J}_s^\alpha + \left(\frac{\partial \boldsymbol{\rho}_s^\alpha}{\partial t}\right)_{ex} + \left(\frac{\partial \boldsymbol{\rho}_s^\alpha}{\partial t}\right)_{\text{SOC}} = 0 \tag{4.93}$$

It is for this reason important to only discuss the spin currents in materials within which it is well defined.

4.5.3 Pair correlation

The pair correlation describes the presence of superconducting correlations in metals. It is defined as

$$\Psi(\boldsymbol{R}) = \langle \psi_\downarrow(\boldsymbol{R})\psi_\uparrow(\boldsymbol{R})\rangle \tag{4.94}$$

Equation 4.94 should not be confused with the superconducting order parameter Δ, which includes a spatially varying function that forces it to be identical to zero outside of superconductors. This is not the case for the pair correlation function, which due to the proximity effect may attain non-zero values also in non-superconducting materials. Expressed with Green functions it becomes

$$\Psi(\boldsymbol{R}) = \lim_{\boldsymbol{r}\to 0} \frac{i}{4}\left(\hat{G}^K_{2,3}(\boldsymbol{R}, T\boldsymbol{r}, t) - \hat{G}^K_{1,4}(\boldsymbol{R}, T, \boldsymbol{r}, t)\right) \tag{4.95}$$

where i and j refers to row i and column j of $\hat{G}^K$. Following the same procedure as for the density of states and the current gives the quasiclassical form of the pair correlation function:

$$\Psi(\boldsymbol{R}) = \frac{1}{8}N_0 \int d\varepsilon \left[\hat{g}^K_{2,3}(\boldsymbol{R}, \varepsilon) - \hat{g}^K_{1,4}(\boldsymbol{R},\boldsymbol{\varepsilon})\right] \tag{4.96}$$

4.5.4 Magnetization

The magnetization is defined as

$$\boldsymbol{M}(\boldsymbol{R}) = \lim_{\boldsymbol{r}_1,\boldsymbol{r}_2\to\boldsymbol{R}} -\frac{g\mu_B}{2}\sum_{\sigma\sigma'} \boldsymbol{\tau}_{\sigma\sigma'}\langle \psi^\dagger_{\sigma'}(\boldsymbol{r}_1)\psi_\sigma(\boldsymbol{r}_2)\rangle \tag{4.97}$$

Using equation 4.79 while allowing for different spins gives

$$\boldsymbol{M}(\boldsymbol{R}) = \lim_{\boldsymbol{r}_1,\boldsymbol{r}_2\to\boldsymbol{R}} -\frac{g\mu_B}{4}\sum_{\sigma\sigma'} \boldsymbol{\tau}_{\sigma\sigma'}\left\{-iG^K_{\sigma'\sigma}(\boldsymbol{r}_1, \boldsymbol{r}_2) + \delta_{\sigma\sigma'}\langle\delta(\boldsymbol{r}_1 - \boldsymbol{r}_2)\rangle\right\} \tag{4.98}$$

The δ-function vanishes, as the trace of the Pauli matrices $\boldsymbol{\tau}$ is zero. To get the magnetization in Nambu$\otimes$spin space, the identity $G^K_{\sigma\sigma'} = -[G^K_{\sigma'\sigma}]^*$, valid in the limit $\boldsymbol{r}_1 \to \boldsymbol{r}_2$, may be used along with the relation $\tau_{\sigma\sigma'} = \tau^*_{\sigma'\sigma}$ to give

$$\begin{aligned}\boldsymbol{M}(\boldsymbol{R}) &= \lim_{\boldsymbol{r}\to 0}\frac{ig\mu_B}{8}\operatorname{Tr}\left\{\tau G^K(\boldsymbol{R}, \boldsymbol{r}) - \tau^*\left[G^K(\boldsymbol{R}, \boldsymbol{r})\right]^*\right\}\\ &= \lim_{\boldsymbol{r}\to 0}\frac{ig\mu_B}{8}\operatorname{Tr}\left\{\hat{\boldsymbol{\tau}}\hat{G}^K(\boldsymbol{R}, \boldsymbol{r})\right\}\end{aligned} \tag{4.99}$$

The quasiclassical version is found by the usual procedure

$$\boldsymbol{M}(\boldsymbol{R}) = \frac{g\mu_B N_0}{8} \int d\varepsilon \, \mathrm{Tr}\left\{\hat{\boldsymbol{\tau}} \hat{g}^K(\boldsymbol{R}, \varepsilon)\right\} \tag{4.100}$$

4.6 Parametrization

To solve the Usadel equation numerically, it is necessary to use a parametrization that takes into account the normalization condition in equation 4.10. One possibility is to use the so-called θ-parametrization, where the Green functions are represented by hyperbolic functions[67]. Another option is the Riccati parametrization[68], where the retarded Green function is written as

$$\hat{g}^R = \begin{pmatrix} N(I + \gamma\tilde{\gamma}) & 2N\gamma \\ -2\tilde{N}\tilde{\gamma} & -\tilde{N}(I + \tilde{\gamma}\gamma) \end{pmatrix} \tag{4.101}$$

where $N = (I - \gamma\tilde{\gamma})^{-1}$, and γ is a 2×2-matrix in spin space. The Riccati parametrization has the advantage that $\gamma \in [0, 1]$, which makes it well suited for numerical solution. By inserting equation 4.101 into equation 4.26, the Usadel equation may be expressed as[69]

$$\begin{aligned} D\left[\nabla^2\gamma + 2\nabla\gamma \cdot \tilde{N}\tilde{\gamma}\nabla\gamma\right] = & -2i\varepsilon\gamma - \Delta\sigma_y + \gamma\Delta^*\sigma_y\gamma - i\boldsymbol{h} \cdot (\boldsymbol{\sigma}\gamma - \gamma\boldsymbol{\sigma}^*) \\ & + D\left[\boldsymbol{A} \cdot \boldsymbol{A}\gamma - \gamma\boldsymbol{A}^* \cdot \boldsymbol{A}^* + 2\left(\boldsymbol{A}\gamma + \gamma\boldsymbol{A}^*\right) \cdot \tilde{N}\left(\boldsymbol{A}^* + \tilde{\gamma}\boldsymbol{A}\gamma\right)\right] \\ & + 2iD\left[\nabla\gamma \cdot \tilde{N}\left(\boldsymbol{A}^* + \tilde{\gamma}\boldsymbol{A}\gamma\right) + \left(\boldsymbol{A} + \gamma\boldsymbol{A}^*\tilde{\gamma}\right) \cdot N\nabla\gamma\right] \end{aligned} \tag{4.102}$$

$$\begin{aligned} D\left[\nabla^2\tilde{\gamma} + 2\nabla\tilde{\gamma} \cdot N\gamma\nabla\tilde{\gamma}\right] = & -2i\varepsilon\tilde{\gamma} - \Delta^*\sigma_y^* + \tilde{\gamma}\Delta\sigma_y^*\tilde{\gamma} + i\boldsymbol{h} \cdot (\boldsymbol{\sigma}^*\tilde{\gamma} - \tilde{\gamma}\boldsymbol{\sigma}) \\ & + D\left[\boldsymbol{A}^* \cdot \boldsymbol{A}^*\tilde{\gamma} - \tilde{\gamma}\boldsymbol{A} \cdot \boldsymbol{A} + 2\left(\boldsymbol{A}^*\tilde{\gamma} + \tilde{\gamma}\boldsymbol{A}\right) \cdot N\left(\boldsymbol{A} + \gamma\boldsymbol{A}^*\tilde{\gamma}\right)\right] \\ & - 2iD\left[\nabla\tilde{\gamma} \cdot N\left(\boldsymbol{A} + \gamma\boldsymbol{A}^*\tilde{\gamma}\right) + \left(\boldsymbol{A}^* + \tilde{\gamma}\boldsymbol{A}\gamma\right) \cdot \tilde{N}\nabla\tilde{\gamma}\right] \end{aligned} \tag{4.103}$$

Included in the energy ε is an inelastic scattering term δ, so that $\varepsilon \to (\varepsilon + i\delta)$[70]. The main purpose of this factor is to provide numerical damping, since there are energy values where the solution diverges. One such instance is the solution for a bulk superconductor at energies $\varepsilon = \pm\Delta$. This can be seen from figure 4.2. In the following, a value of $\delta/\Delta = 10^{-3}$ is used. The Kupriyanov-Lukichev boundary conditions, given in equation (4.70) take on the form

$$\boldsymbol{n} \cdot \nabla\gamma_i = \mp\frac{1}{L_i\zeta_i}(I - \gamma_i\tilde{\gamma}_j)N_j(\gamma_i - \gamma_j) + 2i\boldsymbol{n} \cdot \boldsymbol{A}\gamma_i \tag{4.104}$$

where $\boldsymbol{n}$ is a normal vector to the interface. The negative sign should be used for a boundary where region j is to the right of region i, and vice versa. Boundary conditions for $\tilde{\gamma}_i$ are found by tilde conjugation.

Chapter 5

Numerical Solution Method

In the following, the finite element method is derived for the purpose of solving the Usadel equation in higher spatial dimensions. The Usadel equation consists of eight coupled nonlinear partial differential equations, and is highly non-trivial to solve. Even so, several solution methods exist. A brief summary is given in the following.

The finite difference method

The finite difference methods is the most common numerical solution method for partial differential equations. It makes use of local Taylor expansions to discretize the differential operators. Equation 5.1 is an example of such a discretization for a particular coordinate x. The derivatives in point i are approximated to second order in h.

$$\frac{\partial}{\partial x} f(x) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2h}, \quad \frac{\partial^2}{\partial x^2} f(x) \approx \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} \tag{5.1}$$

In more than one dimensions, each coordinate needs to be simultaneously discretized in a similar fashion. The discretization takes the form of a grid, with derivatives approximated by making use of the surrounding points, as is illustrated in figure 5.1.

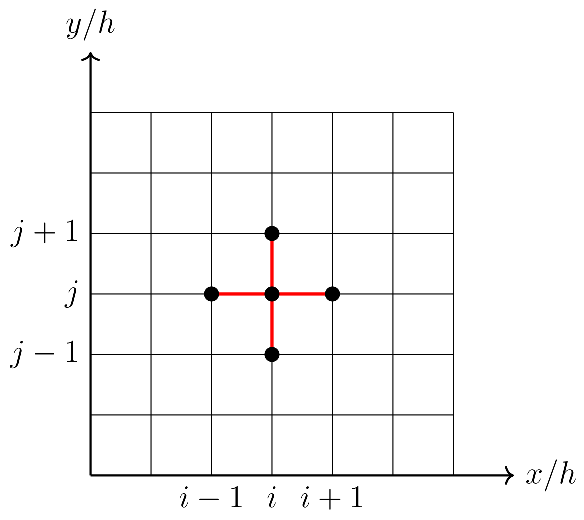


Figure 5.1: Illustration of a two dimensional finite difference numerical scheme.

The discretization itself is performed by inserting equation 5.1 into the PDE to be solved. Several solution methods then exist, with varying degrees of sophistication, which can be divided into two families. Explicit methods formulate the problem in a way where the solution at the next point in space depends only on points where the solution has already been computed. Each step is then generally solved quickly, but the solution is prone to instability. Implicit methods solve for the entire grid simultaneously, thus requiring significantly more memory. For a given grid, these methods are slower than explicit methods, however, as they never become unstable, this may be compensated by allowing for a coarser grid.

A major disadvantage of the finite difference method, as can be gleaned from figure 5.1, is that it requires a rectangular grid. This means that the geometries that can be considered, are restricted to those which can be mapped to a rectangle. Because of this, more complex geometries, such as the inclusion of holes, need to be approximated by a sufficiently fine mesh of rectangles. This quickly becomes a bottleneck in terms of computation time and memory usage. In addition, for every geometry the PDE is to be solved for, the discretization needs to be performed anew. The reason for this is that, while in the bulk of the model, the discretized version of the equations do not vary, the boundary conditions need to be evaluated explicitly.

The finite element method

Rather than discretizing the operators, the finite element method discretizes the geometry. This allows for a much more flexible numerical scheme, with virtually no restrictions on geometry. The main idea behind the finite element method is to restrict the space in which a solution is sought, to a discrete space which is spanned by a set of known test functions. The finite element method will be elaborated on in the following sections.

Other methods

The finite volume method is a method often used in the modeling fluid mechanics, heat transfer and other problems in which a conservation law is followed. The reason why the finite volume method is particularly well suited for these kinds of problems is that the flux between the discretized cells is by definition conserved[71]. A family of very efficient solution methods are the *spectral methods*, which consists of selecting an appropriate basis in which the solution is expanded. This limits the method to problems where an appropriate basis may be inferred, however, once that basis is found, exponential convergence to the exact solution may be found[72]. These methods will not be pursued further herein.

5.1 Finite element theory

This section will endeavor to explain the theory behind the finite element method. To clarify the discussion, the following begins with the much simpler case of a linear partial differential equation (PDE). The main results will then be generalized and applied to the Usadel equation.

5.1.1 Linear equations

The following partial differential equation is defined in a domain Ω with a boundary $\partial\Omega$

$$\begin{cases} \nabla^2 u + f(\boldsymbol{r}) = 0, & u \in \Omega \\ BC[u] = 0, & u \in \partial\Omega \end{cases} \tag{5.2}$$

This is the Poisson equation. Appropriate boundary conditions, $BC[u]$ need to be established for equation 5.2 to be well defined. This treatment will consider two such boundary conditions; the homogeneous Dirichlet boundary conditions and the Neumann boundary conditions. The former is given by

$$u = 0, \quad u \in \partial\Omega \tag{5.3}$$

and latter is given as

$$\boldsymbol{\nu} \cdot \nabla u = g, \quad u \in \partial\Omega \tag{5.4}$$

where $\boldsymbol{\nu}$ is an outwards pointing surface normal. They are of the form of the Nazarov and Kupriyanov-Lukichev boundary conditions, given in equations 4.69 and 4.70 respectively, and are thus particularly relevant.

It is assumed that the solution u to equation 5.2 resides in a Hilbert space V. By multiplying the equation with a testfunction $v \in V$, and integrating over the domain Ω, the *variational* or *weak* formulation is found:

$$\int_{\Omega} d\boldsymbol{r} \nabla u \cdot \nabla v = \int_{\Omega} d\boldsymbol{r} f v + \int_{\partial\Omega} dS \nabla u \cdot \boldsymbol{\nu} v \tag{5.5}$$

where the divergence theorem has been used. For the homogeneous Dirichlet problem, the space V can be defined to have compact support, which means that all elements vanish on the boundary. In other words, the boundary conditions are included in the definition of V. For this reason, the homogeneous Dirichlet boundary conditions are referred to as *essential*. The Neumann boundary conditions are satisfied by inserting equation 5.4 into equation 5.5. In this case, compact support of the solution space may not be defined, and the boundary conditions must be enforced via the equations. These boundary conditions are referred to as *natural*. The restatement of the problem then becomes[73]

$$\text{find } u \in V \text{ such that } a(u, v) = F(v) \ \forall v \in V \tag{5.6}$$

where the bilinear form $a(u, v)$ has been introduced as

$$a(u, v) = \int_{\Omega} d\boldsymbol{r} \nabla u \cdot \nabla v \tag{5.7}$$

and the functional $F(v)$ as

$$F(v) = \begin{cases} \int_{\Omega} d\boldsymbol{r} f v, & \text{Dirichlet} \\ \int_{\Omega} d\boldsymbol{r} f v + \int_{\partial\Omega} dS g v, & \text{Neumann} \end{cases} \tag{5.8}$$

This has not simplified the PDE at all, however it makes for a much better starting point for approximations then the original PDE, particularly because the integral operator is bounded, whereas the differential operator is not. The Galerkin approximation is used next[74], where the infinite dimensional space V, that contains the exact solution, is truncated to a finite dimensional space V^*. The Stone-Weierstrass theorem states that any continuous function that is defined within a closed domain can be approximated to arbitrarily high accuracy by polynomials[75], so an excellent candidate for V^* is the space of polynomials of degree less than or equal to r, $\mathbb{P}_r(\boldsymbol{r})$. Thus, the approximation space gets the desirable quality that $V^* \to V$ as $r \to \infty$. An approximate solution to equation 5.2 is then

$$\text{find } u^* \in V^* \text{ such that } a(u^*, v) = F(v) \ \forall v \in V^* \tag{5.9}$$

The crucial point is then that since V^* is finite, with dimension N, it is spanned by a finite basis of functions ϕ_j, $j = 1, 2, ...N$, and so any element in V^* is a linear combination of these functions

$$u^*(\boldsymbol{r}) = \sum_{j=1}^{N} = u_j^*\phi_j(\boldsymbol{r}), \quad v(\boldsymbol{r}) = \sum_{j=1}^{N} = v_j\phi_j(\boldsymbol{r}) \tag{5.10}$$

Then, for a given component i of v, the discretized version of the PDE becomes

$$\sum_{j=1}^{N} a(\phi_j, \phi_i)u_j^* = F(\phi_i) \tag{5.11}$$

In matrix notation, with $\boldsymbol{F}_i = F(\phi_i)$ and $\boldsymbol{K}_{ij} = a(\phi_j, \phi_i)$ this becomes

$$\boldsymbol{K}\boldsymbol{u}^* = \boldsymbol{F} \tag{5.12}$$

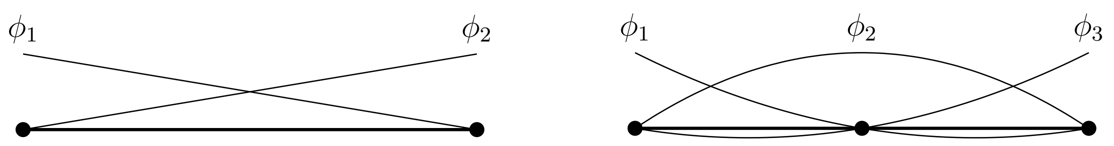
The matrix $\boldsymbol{K}$ is called the stiffness matrix and $\boldsymbol{F}$ the force vector for historical reasons, as the finite element method was first used in structural mechanics. Once $\boldsymbol{K}$ and $\boldsymbol{F}$ have been created, the equation system may be solver for u^*. To establish these quantities, the appropriate basis functions $\phi_j(\boldsymbol{r})$ need to be selected.

5.1.2 Elements, nodes and interpolation

In order to obtain a basis in which to expand the approximate solution, u^*, the finite element method divides the domain Ω into a discrete set of subdomains Ω_e. These subdomains are called elements. On each element, there may be defined a set of N_n points, called nodes, where the solution of the PDE is computed. Within the element there may then be defined N_n functions that interpolate between the nodes. Several possibilities exist when selecting interpolation functions, however the most common choice, which is also what will be used herein, are the Lagrange polynomials. These are in one dimension given as

$$\phi_j(x) = \prod_{\substack{m=1 \\ m \neq j}}^{N_n} \frac{x - x_m}{x_j - x_m} \tag{5.13}$$

By construction, the Lagrange polynomials are linearly independent and satisfy $\phi_i(x_i)\phi_j(x_j) = \delta_{ij}$. Furthermore they span the space $\mathbb{P}_{N_n-1}(x)$, i.e., the space of polynomials of degree $N_n - 1$.



(a) A linear element **(b) A quadratic element**

Figure 5.2: Lagrange polynomials for one dimensional line elements. The dots denote node locations.

Within an element domain Ω_e, the Lagrange polynomials may thus be used to express u^*. It is a requirement for the Galerkin method that the solution is continuous across element boundaries. This is satisfied by enforcing that elements sharing boundaries also share the nodes on the boundary. Since the Lagrange polynomials are completely determined by its values at the nodes, continuity follows. This means that the elements must at least have nodes at its vertices, i.e., at the ends of line elements and corners of quadrilaterals. The choice of Lagrange polynomials has an additional advantage. By construction, the polynomial belonging to a specific node, has only non-zero values in elements that share the node. This makes the resulting matrix sparse, which is highly beneficial in terms of solving the system. The same procedure works also in higher dimensions. For a quadrilateral (that is, a rectangle), with nodes only in the corners, the two dimensional Lagrange interpolation functions are given as products of the one dimensional Lagrange polynomials along respective axes. More complex elements, such as quadrilaterals with internal nodes, triangles and tetrahedra require a bit more thought. In two dimensions, a general way to establish the interpolation functions is to use the Pascal triangle, which visualizes in an intuitive way the monomials that need to be included in order to create interpolation functions for a certain number of nodes.

Element order	r					Pascal triangle				
Constant	0					1				
Linear	1				x		y			
Quadratic	2			x^2		xy		y^2		
Cubic	3		x^3		x^2y		xy^2		y^3	
Quartic	4	x^4		x^3y		x^2y^2		xy^3		y^4

Figure 5.3: Visualization of number of terms in a complete polynomial corresponding to a given element order in two dimensions. The parameter r denotes the degree of the polynomial space.

From figure 5.3, the basis for a, for instance, nine node element can be immediately read off as the nine monomials of lowest order in the Pascal triangle, selected symmetrically about its center line. The latter is necessary to avoid giving the elements a bias towards a given direction.

$$\Phi = a_1 + a_2x + a_3y + a_4x^2 + a_5xy + a_6y^2 + a_7x^2y + a_8xy^2 + a_9x^2y^2 \tag{5.14}$$

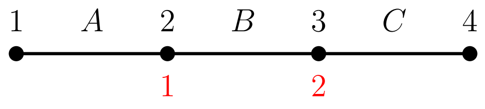
The coefficients a_i are determined by enforcing orthonormality in the nodes. It is possible to use the same procedure in three dimensions, by establishing a Pascal pyramid, however its visualizing power is greatly diminished by the complexity of such a construction. Inserting the interpolation functions into equation 5.11, the stiffness matrix is found as

$$K_{ij}^{(e)} = \int_{\Omega} d\boldsymbol{r} \nabla\phi_i(\boldsymbol{r}) \cdot \nabla\phi_j(\boldsymbol{r}) \tag{5.15}$$

and the right hand side is

$$F_i^{(e)} = \int_{\Omega} d\boldsymbol{r} f(\boldsymbol{r})\phi_i(\boldsymbol{r}) \tag{5.16}$$

In this expressions, all coefficients in the expansion of v in equation 5.10 has been set equal to one; $v_j = 1 \ \forall j$. This is permitted as v was arbitrary. The upper index (e) indicates that equations 5.15 and 5.16 give the contribution from a single element. A global stiffness matrix K and force vector F valid for the entire model needs to be found. However, this follows trivially once all the element contributions have been determined. In fact, every node can be given two labels; a local index, which identifies the interpolation function that belongs to it, and a global index which determines the structure of K and F. The global matrix system is then found by inserting the matrix elements of the local matrix into positions in the global matrix identified by the global node indices. The global stiffness matrix then becomes an $M \times M$ matrix, where M is the total number of nodes in the mesh. An example is shown in figure 5.4 using a mesh consisting of three line elements with linear interpolation functions. It is observed that matrix element k_{22} in the global matrix gets contributions both from element A and B. Similarly, k_{33} gets contributions both from element B and C. Matrix elements k_{13}, k_{14} and k_{24}, as well as their counterparts with reversed indices are zero, since there is no communication between these nodes. This illustrates the sparseness of the global matrix K for larger systems.



$$K_B = \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix} \quad \Rightarrow \quad K = \begin{pmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{pmatrix}$$

Figure 5.4: Illustration of local to global matrix restructuring for a mesh consisting of three linear elements.

5.1.3 Coordinate transformation and numerical integration

There are virtually no limits with regards to the types of geometries that can be described by the finite element method. The only requirement is that it may be adequately represented by a mesh. In general, polygons can be recreated exactly by a mesh. For other types of geometries, such as spheres and splines, the mesh is approximate. This is a feature reminiscent of the finite difference method, however an important difference is that the finite element method allows distortion of the elements to better fit the geometry, with only minor influence on the discretized equations. For this reason, it is necessary to understand how a distorted element can be described.

The shape of the element may be expressed in the same basis as the solution u^*.

$$x = \sum_i x_i \phi_i(\xi), \quad y = \sum_i y_i \phi_i(\eta), \quad z = \sum_i z_i \phi_i(\zeta) \tag{5.17}$$

These elements are called *isoparametric*. Other choices exist, where the polynomial used is either of lower or greater degree than those used in representing the solution, giving *subparametric* and *superparametric* elements respectively. Such element types are less commonly used, and will not be explored further. From equation 5.17 it is seen that the distortion of the element is fully determined by the nodal positions, and so a higher order element may describe more complex geometries. The coordinate transformation transforms the real element to a reference element for which numerical integration is much more convenient. This is illustrated in figure 5.5.

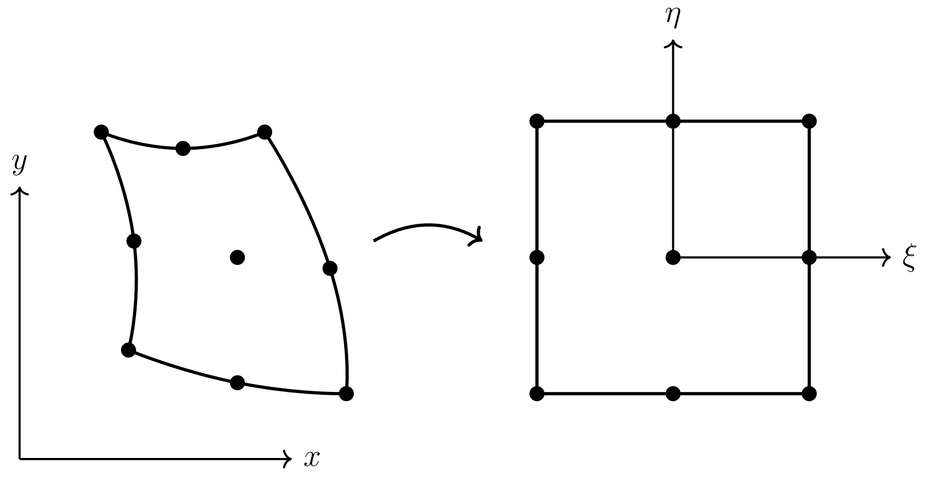


Figure 5.5: Illustration of a transformation to the reference element.

The gradient of the interpolation functions used in equation 5.15 then takes the form

$$\nabla_{\boldsymbol{r}} \phi_i(\boldsymbol{r}) = \boldsymbol{J}^{-1} \nabla_{\boldsymbol{\rho}} \phi_i(\rho) \tag{5.18}$$

where $\boldsymbol{J}$ is the Jacobian matrix for the coordinate transformation, given as

$$\boldsymbol{J} = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \zeta} \end{pmatrix} \tag{5.19}$$

The transformed stiffness matrix becomes, with $\boldsymbol{\rho} = (\xi, \eta, \zeta)$

$$K_{ij}^{(e)} = \int_{\Omega} d\boldsymbol{\rho}\ |J| \Big[\boldsymbol{J}^{-1} \nabla_{\rho} \phi_i(\boldsymbol{\rho})\Big]^T \boldsymbol{J}^{-1} \nabla_{\rho} \phi_j(\boldsymbol{\rho}) \tag{5.20}$$

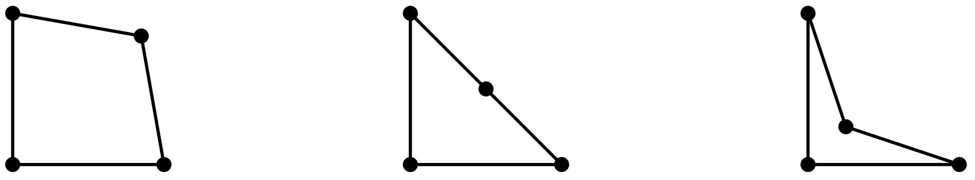
In equation 5.20, factor $|J|= \det(\boldsymbol{J})$ appears due to the coordinate transformation of the integration measure. In addition, matrix notation has been used for the scalar product. The force vector is given as

$$F_i^{(e)} = \int_{\Omega} d\boldsymbol{\rho} |J| f(\boldsymbol{\rho}) \phi_i(\boldsymbol{\rho}) \tag{5.21}$$

The reason the coordinate transformation is so convenient is that the interpolation functions can be defined for the reference element in the coordinates $\boldsymbol{\rho}$. Given that the same element type is used for the entire model, these never change and so may be given in a lookup table for ease of programming. By using the Jacobian matrix, the interpolation functions are transformed to the real geometry. This puts restrictions on the element distortions, as the inverse of the Jacobian matrix has to exist. This means that the Jacobian determinant cannot be zero. In addition, it cannot be negative, as this represents a situation where the element overlaps itself, thus resulting in a coordinate transformation which is ill-defined. In other words, the value of the Jacobian within the elements is a measure of the quality of the mesh. To get a sense of how element distortions influence the Jacobian, a 4 node linear quadrilateral is considered, where only the upper right node is moved, with the other nodes placed in positions equal to corners of the reference element. In this case, the Jacobian determinant is given as

$$J = \frac{1}{4}(1-\eta) + \frac{1}{4}(1-\xi) + \frac{1}{4} x_3 (1+\eta) + \frac{1}{4} y_3 (1+\xi) \tag{5.22}$$

By investigating different locations (x_3, y_3) of the movable node, it is seen from figure 5.6 that this element remains acceptable as long as it is convex.



(a) $J > 0$ **(b)** $J = 0$ **(c)** $J < 0$

Figure 5.6: The Jacobian determinant for different element distortions, evaluated in the upper right corner. Of the three, only (a) is acceptable.

The next step in generating the stiffness matrix and force vector is to compute the integral in equation 5.20 and 5.21. This could in principle be done analytically, as both the interpolation functions and Jacobian matrices are known once the mesh has been created, however this is not done in practice. Instead, numerical integration is used. One option is to use common integration methods such as the trapezoidal or Simpson rule, where the domain to be integrated is divided into equipartitioned intervals where the integrand is evaluated. A better approach is to use the Gauss quadrature, where the sampling points are selected, and the integrand weighted, in a way so as to minimize integration error.

$$I = \int f(\xi, \eta, \zeta)\, d\xi d\eta d\zeta \approx \sum_{ijk} w_i w_j w_k f(\xi_i, \eta_i, \zeta_i) \tag{5.23}$$

The integration points (ξ_i, η_j, ζ_k) are given by the roots of the Legendre polynomials $P_n(\rho)$, and the weights may be calculated by the formula[76]

$$w_i = \frac{2}{(1-\rho_i^2)\left[\frac{d}{d\rho}P_n(\rho_i)\right]} \tag{5.24}$$

with ρ_i one of the coordinates ξ, η or ζ, evaluated in the integration point. The stiffness matrix is then given by

$$K_{ij}^{(e)} = \sum_{klm} w_k w_l w_m |J| \left[\boldsymbol{J}^{-1}\nabla\phi_i(\xi_k, \eta_l, \zeta_m)\right]^T \boldsymbol{J}^{-1}\nabla\phi_j(\xi_k, \eta_l, \zeta_m) \tag{5.25}$$

and the force vector by

$$F_i^{(e)} = \sum_{klm} w_k w_l w_m |J| f(\xi_k, \eta_l, \zeta_m)\phi_i(\xi_k, \eta_l, \zeta_m) \tag{5.26}$$

where the sums are over the integration points. The Gauss quadrature is well suited for polynomial integrands. In fact, for a polynomial of degree n in 1D, a Gauss quadrature of at least $2n-1$ integration points produces the analytical result. It is customary to choose the number of integration points so that the undeformed element of a given degree is integrated exactly. This is called *full integration.* In general, the integration will still be approximate, due to the appearance $\boldsymbol{J}^{-1}$ in equation 5.20, which is not necessarily polynomial.

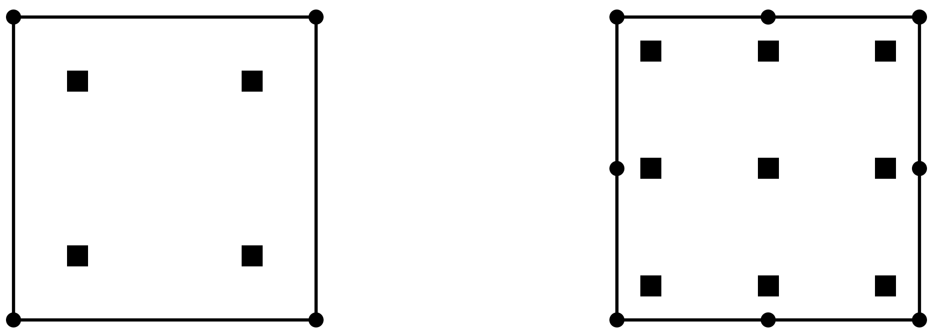


Figure 5.7: Number and position of integration points for a fully integrated (a) linear 4-node element (Q4) (b) quadratic 8-node element (Q8).

Figure 5.7 illustrates an important source of confusion. While solution of the matrix system in equation 5.12 finds u^* in the nodes, the stiffness matrix and the force vector are computed only in the integration points. This is beneficial from a programming perspective. Every element in the mesh is created by a coordinate transformation of the reference element, onto which the integration points are defined. Thus, it is sufficient to evaluate the interpolation functions in these points, as it is only the Jacobian matrix that changes between elements. This makes for highly efficient code.

5.1.4 Nonlinear finite element method and the Usadel equation

For a linear PDE, the solution $\boldsymbol{u}^*$ is found by multiplying equation 5.12 by $\boldsymbol{K}^{-1}$ from the left. For a nonlinear problem, the situation is quite another. The toy problem of equation 5.2 is made nonlinear by allowing the force function to depend on u and ∇u, i.e.,

$$\begin{cases} \nabla^2 u + f(\boldsymbol{r}, u, \nabla u) = 0, & u \in \Omega \\ BC[u] = 0, & u \in \partial\Omega \end{cases} \tag{5.27}$$

In this case, after discretization, equation 5.12 may not be solved, since $\boldsymbol{u}^*$ appears also in $\boldsymbol{F}$. Progress is aided by the realization that equation 5.12 may be generalized. What really happens when the solution space V is replaced by V^*, or $u \rightarrow u^*$, is that in general, equation 5.2 is no longer satisfied. This implies that equation 5.12 may be written as

$$\boldsymbol{K}\boldsymbol{u}^* - \boldsymbol{F} = \boldsymbol{R} \tag{5.28}$$

where $\boldsymbol{R}$ is a residual vector representing the difference between the exact and the approximate solution. Equation 5.28 is solved for $\boldsymbol{R} = 0$. In a linear analysis, this is done in a single step, and so it was unnecessary to include it explicitly in equation 5.12. For the present case, an incremental solution method needs to be used. This is achieved by means of Newton-Raphson iterations.

$$\boldsymbol{u}^*_{n+1} = \boldsymbol{u}^*_n - \left[\frac{\partial \boldsymbol{R}_n}{\partial \boldsymbol{u}^*_n}\right]^{-1} \boldsymbol{R}_n = \boldsymbol{u}^*_n - \left[\boldsymbol{K} - \frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{u}^*_n}\right]^{-1} \boldsymbol{R}_n = \boldsymbol{u}^*_n - \boldsymbol{K}_T^{-1}\boldsymbol{R}_n \tag{5.29}$$

An initial solution $\boldsymbol{u}^*_0$ is guessed upon, and iterations towards equilibrium are performed using equation 5.29. The matrix $\boldsymbol{K}_T^{-1}$ is called the tangent stiffness matrix, a name which is justified by the illustration in figure 5.8. It shows the equilibrium path of the force vector $\boldsymbol{F}$, which is the proper dependence of $\boldsymbol{F}$ on the solution $\boldsymbol{u}^*$. Every Newton-Raphson iteration will then consist of computing the tangent to this curve at the current step, which is used as the stiffness matrix. Solving the system by using equation 5.29 and the formalism developed for the linear finite element method will give a solution that is closer to equilibrium - assuming convergence - and the process is repeated anew. Obviously, figure 5.8 is only a visual aid, as all variables involved are matrices.

A few remarks are in order. In general, the solution of the matrix equations will involve iterations even for the linear finite element method. This is because the stiffness matrix generally is too large to be read into the memory of a computer in its entirety. For this reason, the Newton-Raphson procedure described in this section is often referred to as nonlinear iterations. Thus, for every nonlinear iteration, there are linear iterations to find the solution $\boldsymbol{u}^*_n$. Assuming both the linear and the nonlinear iterations converge to within an acceptable error, one has still only found the approximate solution u^*. This makes the finite element method quite dangerous to use. The solution is only as good as the mesh. If the quality of the mesh is poor, the solution towards which the iterations converge, u^*, might differ significantly from the exact solution of the PDE, u. It is difficult to determine whether the computed solution is correct, as finite element software only reports the error with respect to u^*. This may mislead the analyst into thinking the accuracy is high when this may not be the case. A healthy dose of skepticism towards the computed results is required. One should compare different levels of mesh refinement to determine whether the approximate solution is converging. In addition, the analyst should check whether the boundary conditions are satisfied and that the solution agrees with physical intuition.

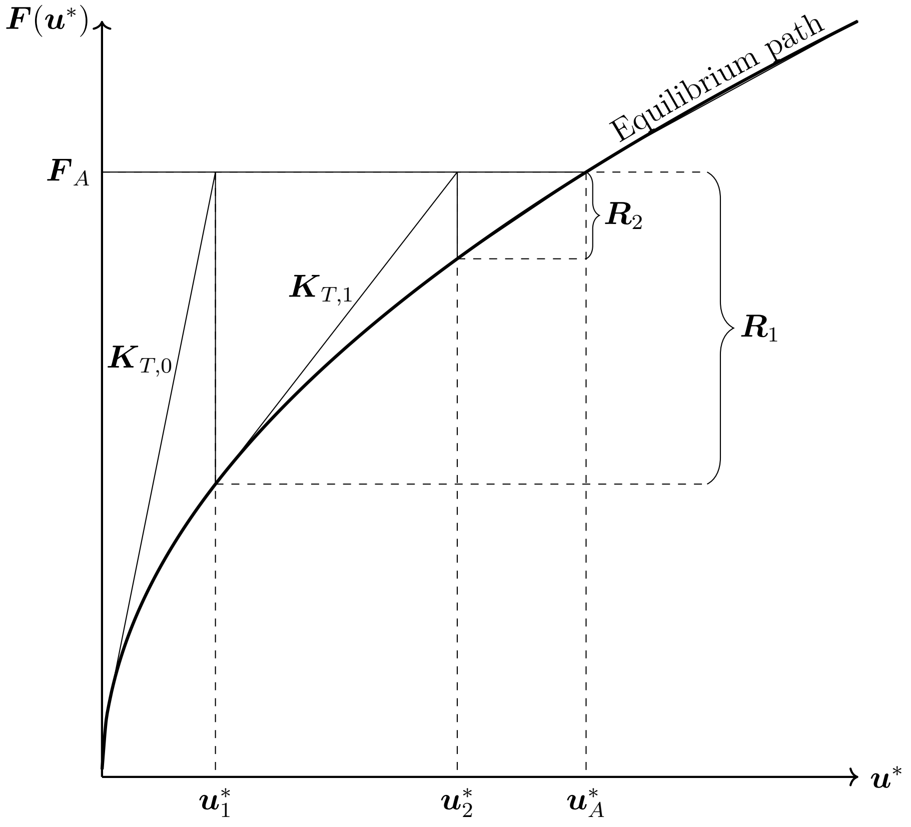


Figure 5.8: Illustration of Newton-Raphson iterations for a given applied force vector F_A.

All the tools required to solve the Usadel equation numerically have now been developed. By inspecting equations 4.102 and 4.103 it is seen that the Usadel equation in the Riccati parametrization is exactly of the form given in equation 5.27. A complicating factor is, however, that γ and $\tilde{\gamma}$ are each 2×2 matrices, thus giving 8 coupled PDEs. To simplify the notation, the elements of these matrices are placed in a single vector

$$\Gamma = (\gamma_{11}, \gamma_{12}, \gamma_{21}, \gamma_{22}, \tilde{\gamma}_{11}, \tilde{\gamma}_{12}, \tilde{\gamma}_{21}, \tilde{\gamma}_{22})^T \tag{5.30}$$

where γ_{ij} and $\tilde{\gamma}_{ij}$ are elements of the γ and $\tilde{\gamma}$ matrices respectively. It is then possible to write the Usadel equation for element α of Γ as

$$\nabla^2 \Gamma^{(\alpha)} = Q^{(\alpha)}(\gamma, \tilde{\gamma}, \nabla\gamma, \nabla\tilde{\gamma}) \tag{5.31}$$

In equation 5.31, the function $Q^{(\alpha)}$ performs the matrix multiplications of equations 4.102 and 4.103, and extracts the appropriate element. Similarly, the boundary conditions given in equation 4.70 may be expressed as

$$\boldsymbol{n} \cdot \nabla\Gamma^{(\alpha)} = B^{(\alpha)}(\gamma, \tilde{\gamma}) \tag{5.32}$$

where $\boldsymbol{n}$ is, as defined in equation 4.70, a unit vector pointing along positive coordinate axes. The nonlinear finite element formulation of the Usadel equation is then given as

$$\Gamma_{n+1} = \Gamma_n - \mathbb{J}_n^{-1}\mathbb{R}_n \tag{5.33}$$

The matrix $\mathbb{R}_n$ is given as

$$\mathbb{R}_n^{(\alpha)} = \sum_{ijk} |J| w_i w_j w_k \left\{ - \left[\boldsymbol{J}^{-1}\nabla\boldsymbol{\Gamma}^{(\alpha)}\right]^T \left[\boldsymbol{J}^{-1}\nabla\boldsymbol{\phi}\right] + \boldsymbol{Q}^{(\alpha)}\boldsymbol{\phi} \right\} + (\boldsymbol{n}\cdot\boldsymbol{\nu}) \sum_{ij} |J| w_i w_j \boldsymbol{B}^{(\alpha)}\boldsymbol{\phi} \tag{5.34}$$

The matrix $\mathbb{J}$ is the Jacobian matrix in 8 dimensional variable space, and appears as a multidimensional generalization of the Newton-Raphson method. The surface integral in equation 5.34 is also to be performed by Gauss quadrature. This Jacobian is not to be confused with the Jacobian matrix $\boldsymbol{J}$ for the coordinate transformation of isoparametric elements. $\mathbb{J}$ is given as

$$\begin{aligned} \mathbb{J}^{(\alpha\beta)} = \frac{\partial \mathbb{R}^{(\beta)}}{\partial \boldsymbol{\Gamma}^{(\alpha)}} &= \sum_{ijk} |J| w_i w_j w_k \left\{ -\delta_{\alpha\beta} \left[\boldsymbol{J}^{-1}\nabla\boldsymbol{\phi}\right]^T \left[\boldsymbol{J}^{-1}\nabla\boldsymbol{\phi}\right] + \frac{\partial \boldsymbol{Q}^{(\beta)}}{\partial \boldsymbol{\Gamma}^{(\alpha)}} \boldsymbol{\phi}^T\boldsymbol{\phi} \right\} \\ &+ (\boldsymbol{n}\cdot\boldsymbol{\nu}) \sum_{ij} |J| w_i w_j \frac{\partial \boldsymbol{B}^{(\beta)}}{\partial \boldsymbol{\Gamma}^{(\alpha)}} \boldsymbol{\phi}^T\boldsymbol{\phi} \end{aligned} \tag{5.35}$$

Technical details regarding implementation

The boundary conditions to be used are given in equation (4.104). If the geometry for which the Usadel equation is solved only has interfaces towards materials where the Green function is constant, they reduce to regular Neumann boundary conditions. This is the case if the adjoining superconductors are much larger than the material considered, so that they can be approximated by the bulk expression given in equation (4.15). Another example is if interfaces towards infinite normal metals are considered. However, if materials on both sides of the interface are to be included in the analysis, the situation is not so simple. An essential assumption within the Galerkin method is that the solution is continuous. This is not the case for the quasiclassical Green function when crossing an interface. Here it is seen that the boundary conditions on one side of the interface, influencing one of the materials, depend on the solution on the other side, and vice versa. One way around this problem is to first solve for one material by inserting an initial guess for the interface boundary conditions, use that solution to compute boundary conditions for the other material, and repeat until convergence is achieved. This will be referred to as the iterative approach.

A different strategy will be employed here, dubbed the direct approach. In this method, disjoint meshes are created, one for each material that is to be included. This means that there will be no elements that connect the different materials. The stiffness matrix then takes the following form

$$\boldsymbol{K}_{\text{tot}} = \begin{pmatrix} \boldsymbol{K}_1 & 0 \\ 0 & \boldsymbol{K}_2 \end{pmatrix} \tag{5.36}$$

with index 1 and 2 referring to the materials on either side of the interface. As there is no coupling between the two matrices, a solution is found in the two meshes independently. By making sure that the nodes on the interface line up, as illustrated in figure 5.9, least square methods may then be used to create a mapping between integration points belonging to elements on opposite sides. In this way, the solution on one side of the interface can be extracted and used in the boundary condition for the solution on the other side. This can be included as a part of the iterative procedure used in solving the equation system, and therefore no additional self-consistency iterations are required. The correct boundary conditions are found to within the error of the numerical solution with a negligible increase in execution time compared to an analysis where the same number of elements is used, but no interface. For one interface and 20×20 elements in each mesh, it was found that the direct approach was as fast as a single self-consistency iteration of the iterative approach. In other words, the direct approach was far superior. However, the dimension of the stiffness matrix increases as 2^n where n is the number of interfaces. Because of this, it is expected that for a system with a large number of interfaces, the direct approach will eventually become less efficient than the iterative approach, as well as require much more memory.

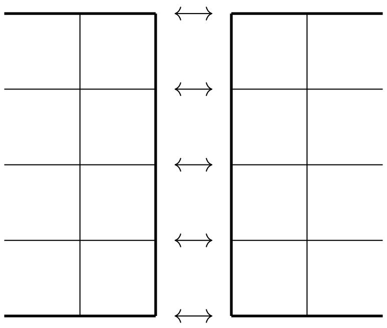


Figure 5.9: Illustration of mesh alignment at an interface between two materials, thus allowing a discontinuous transition.

To compute some of the output the Green functions generate, for instance the currents, the gradient of the solution is needed. At first sight, this seems straight forward. The solution is found as the expansion coefficients of the known interpolation functions, and therefore the gradient of the solution field can in principle be computed. There is, however, a subtle point. The interpolation functions are not directly available, as they are defined for the reference element. What is availabe, are the gradients of these functions, evaluated at the integration points. Meanwhile, the solution is computed in the nodes. For this reason, an extrapolation procedure is necessary to find the gradients at the nodes. This can be done within the finite element formalism by defining the force function as the gradients at the integration points, $f = \nabla\Gamma_q$. Renaming the unknown gradients at the nodes as $b = \nabla\Gamma_n$, the equation to be solved then becomes $b = f$. While this seems like a trivial equality, it is indeed an extrapolation routine. The reason for this is that when passing the equation through the finite element machinery, f is computed at the integration points, but the solution, b, is found at the nodes. This amounts to performing the substitution $\nabla\phi_i \to \phi_i$ and $\nabla\phi_j \to \phi_j$ in equation (5.25).

5.2 Comparison with the finite difference method

The finite element method can be thought of as a generalization of the more familiar finite difference method. To see this, consider the toy problem of equation 5.2 defined on a one dimensional bar discretized by linear elements. The approximate solution within an element is then given by

$$u(x) = u_i\phi_1(x) + u_{i+1}\phi_2(x) = u_i\left(1 - \frac{x}{L_e}\right) + u_{i+1}\frac{x}{L_e} = u_i + \frac{u_{i+1} - u_i}{L_e}x \tag{5.37}$$

It is observed that the interpolation of $u(x)$ is nothing but a Taylor expansion to first order in the element length L_e. The stiffness matrix for the given element is found by using equation 5.15 to be

$$\boldsymbol{K} = \frac{1}{L_e}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{5.38}$$

With, for instance 5 elements, the global stiffness matrix becomes

$$K = \frac{1}{L_e}\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \tag{5.39}$$

If the right hand side of equation 5.2 is sufficiently smooth for the approximation $f(x) \approx f(x_i) = f_i$ to be reasonable, the global force vector for element i becomes $F_i = L_e f_i$. Therefore, the numerical scheme for node i becomes

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{L_e^2} = f_i \tag{5.40}$$

which is precisely the second order accurate finite difference scheme for equation 5.2.

Chapter 6

Applications

A numerical code has been developed based on the method described in chapter 5. It is written in C++ and makes use of the finite element library libMesh[77] and the equation solving toolkit PETSc[78,79,80]. In the following sections, application of the program to 2D and 3D systems is presented. Kupriyanov-Lukichev boundary conditions, given in equation (4.70) are used at interfaces between materials. At all other edges, vacuum is assumed, which imposes the boundary condition that no current can exit the system here, resulting in

$$\boldsymbol{n} \cdot \bar{\nabla}\check{g} = 0 \tag{6.1}$$

where it is recalled that $\bar{\nabla}\check{g} = \nabla - [\boldsymbol{A}, \check{g}]$ and $\boldsymbol{n}$ is the surface normal. Analyses that include external flux and spin-orbit coupling encounter a subtle problem that warrant commenting. Applying these effects to a restricted space means mathematically to include step functions. To give a concrete example, a particular spin-orbit coupling is applied to the half-space $x > 0$

$$H = \alpha\tau_x p_x \theta(x) \tag{6.2}$$

This expression is not Hermitian, as p_x and the Heaviside step function $\theta(x)$ does not commute. To mend this, the Hamiltonian may be symmetrized

$$H = \frac{1}{2}\alpha\tau_x \left(p_x\theta(x) + \theta(x)p_x\right) = \alpha\tau_x p_x\theta(x) + \frac{1}{2}\alpha\tau_x \left[\theta(x), p_x\right] \tag{6.3}$$

The commutator in equation (6.3) has been computed in appendix A.2, resulting in

$$H = \alpha\tau_x p_x\theta(x) + \frac{1}{2}\alpha\tau_x\delta(x) \tag{6.4}$$

In other words, the Hamiltonian for the spin-orbit coupling is made Hermitian by introducing a δ-function term. This means that total Hamiltonian is unaltered in the bulk of the material, but an additional term appears at the boundary. The transfer matrix M, used in the derivation of the boundary conditions in section section 4.4, is easily found for the δ-function potential, however, this leads to spin-dependent boundary conditions, as the transfer matrix now gets structure in spin space. It is assumed that this contribution is small with respect to the regular Kupriyanov-Lukichev contribution, and for this reason, the spin-orbit addition is ignored. It has been verified numerically that the charge current through interfaces is conserved to within numerical precision.

Restricting the magnetic vector potential $\boldsymbol{A}$ in a similar manner, needs to be done in a way which ensures a physical magnetic field, $\boldsymbol{B} = \nabla \times \boldsymbol{A}$. This means that step functions should be introduced so as to not give a contribution to the curl of the vector potential. For a magnetic field in the z-direction, present only for $x > 0$, the vector potential in the Landau gauge becomes

$$A_x = B_z y \theta(x) \tag{6.5}$$

The magnetic field may then be computed as $\boldsymbol{B} = B_z \theta(x) \hat{z}$, as it should. In addition, the Hamiltonian is Hermitian due to the way the vector potential is included. However, during the derivation of the quasiclassical equations, the Coulomb gauge, $\nabla \cdot \boldsymbol{A}$, was assumed, ensuring that $\boldsymbol{A}$ commutes with $\boldsymbol{p}$. This is not satisfied with the imposed restrictions. By ignoring this, and treating $\boldsymbol{A}$ and $\boldsymbol{p}$ as if they commute, it is seen from the previous analysis that this is equivalent to including δ-function terms, present only at the boundary. These terms are readily included in the scattering matrix, and does not alter the form of the boundary conditions.

6.1 Numerical verification: Limiting cases

Since the finite element method has not been used to solve the Usadel equation before, it is necessary to verify that the method does in fact give correct results. This is important regardless of the problem considered, but especially so when nonlinearities are involved.

6.1.1 The Josephson junction

To evaluate the numerical method, the two dimensional Josephson junction will be considered. The normal metal is assumed to be square, and the superconductors large enough to be well described by the bulk superconductor. This means that the superconductors appear only as boundary conditions and no self-consistency iterations are required. On the sides of the normal metal not attached to superconductors, vacuum is assumed.

Systems both with and without an external magnetic field are analyzed. A valuable tool in the verification of the numerical method is to determine whether the solution approaches a constant value as the mesh is refined. Therefore, the current density in the x-direction is given in the middle of the junction for a selection of increasingly fine meshes. This is shown in figure 6.1. The external flux applied for the purpose of investigating convergence is $\Phi = 2\Phi_0$, where $\Phi_0 = \frac{h}{2e}$ is the flux quantum. A convergence plot for the charge current is shown in figure 6.1, with a phase difference between the superconductors of $\phi = \frac{\pi}{2}$. It is seen that convergence is achieved quickly. Without external flux, even with only two elements on each side (4 total), the computed current differs from the 16×16 element analysis by only 2%. With external flux, it is seen that the current changes significantly as the number of elements is increased for 2×2 to 4×4. This signifies that the mesh was too crude to resolve the current density. A denser mesh enables the representation of faster varying solutions, and therefore more complex current densities. With more than 4×4 elements, the current is seen to converge, but requires a larger amount of elements than the analysis without external flux. This illustrates how the solution dictates the mesh density, requiring trial and error. An important point is that the convergence analysis performed on the charge current, gives an upper limit for the amount of elements that are necessary. This is because the gradient is required to compute the currents, which reduces the order of the interpolation polynomials by one, and hence their accuracy. In this case, quadratic elements were used, meaning currents can be described within the elements by linear interpolation. Linear elements would give constant inter-element currents and would require a higher mesh density. Observables such as the density of states do not require the gradient, and converge much faster. In fact, without external flux, 2×2 elements is sufficient.

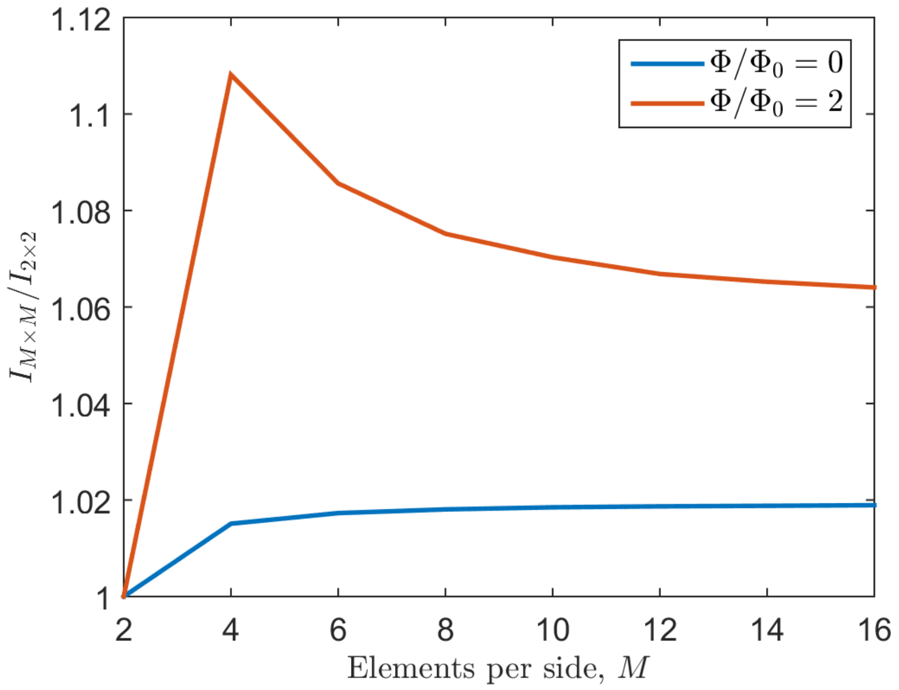
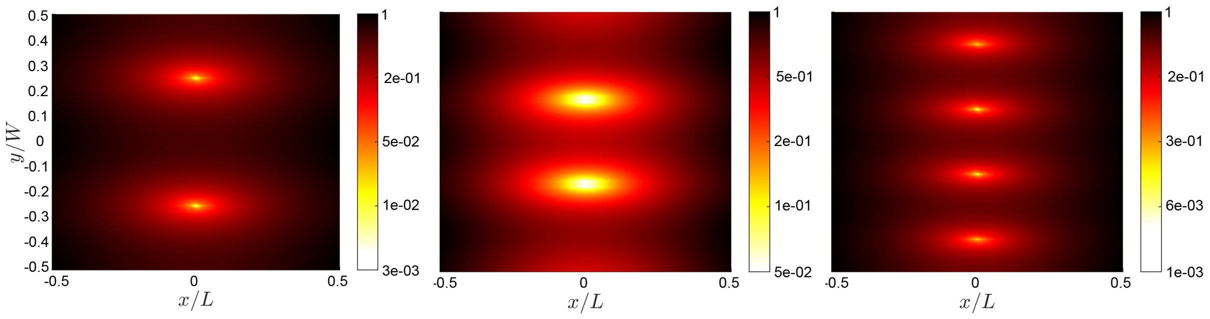


Figure 6.1: Convergence plot for the numerical solution of the Usadel equation on a Josephson junction with and without an external magnetic field.

Another way to evaluate the numerical method, is to use a known benchmark. This will be accomplished by comparison with the works of Bergeret and Cuevas[81,12] who consider a two dimensional Josephson junction with external flux. They solve the Usadel equation numerically using a finite difference scheme, and compute the pair correlation function, given in equation 4.96. They find that for a very wide junction, the external flux produces a linear array of vortices in the transverse direction of the junction, where the pair correlation is zero within numerical accuracy. Furthermore, they find that the location of these vortices can be calculated analytically in the limit of weak proximity effect. The location of vortex n in the transversal direction y is given by

$$y_n = \phi - nW\frac{\Phi_0}{\Phi} \tag{6.6}$$

where W is the width of the junction, and $n = \pm 1, \pm 2, ...$ denotes the vortex number. It is observed that the number of vortices is equal to the number of flux quanta passing through the junction. These results have been recreated for a system where $W = 8\xi$ and $L = 2\xi$ for varying external flux. The results are shown in figure 6.2 and are seen to match the findings of Bergeret and Cuevas well.



(a) $\Phi/\Phi_0 = 2$ (b) $\Phi/\Phi_0 = 3$ (c) $\Phi/\Phi_0 = 4$

Figure 6.2: The pair correlation for a wide Josephson junction with external flux. The width is $W = 8\xi$ and the length is $L = 2\xi$. The pair corelation function is normalized with respect to its value at the interface to the superconductors.

6.1.2 Superconductor-ferromagnet bilayer in 3D

To further test the capabilities of the numerical scheme, a 3D cylindrical superconductor-ferromagnet bilayer is considered. Once again, the superconductor is assumed infinite, so that the bulk expression is valid, which appears only as a boundary condition. In other words, the system can be thought of as a ferromagnetic nanoisland placed atop a superconductor, as shown in figure 6.3. The ferromagnet is otherwise surrounded by vacuum. The solution is expected to be identical to a one dimensional system as there is no explicit radial dependence.

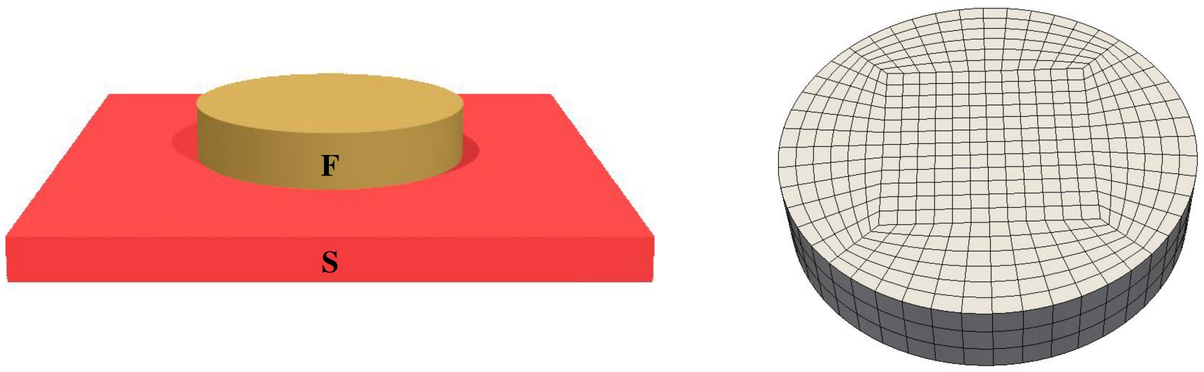


Figure 6.3: Geometry and mesh of a 3D SF bilayer.

Shown in figure 6.3 is the mesh used to discretize the system. This has been done in a particular manner so as to minimize element distortion. Hexagonal 27-node elements are used, which allow for second order interpolation polynomials. Meshing a circular geometry with rectangular elements is a challenge due to high mesh distortions near the center of the model. One possibility is to use an irregular mesh, however accuracy is improved by preserving regularity. The approach used here is to create an inner square of a completely regular mesh, with an outer layer creating the remainder of the cylinder, as can be seen in figure 6.3. The highest mesh distortions are then found near the corners of the inner square, however the Jacobian is far from being ill-defined, as would have been the case if the mesh was designed with elements fanning out from the center. It is also observed how the outer elements are curved according to the curvature of the model. This enables a near-perfect description of the geometry. It is emphasized, however, that this is just an approximation to a circle, as the element shapes are created by parabolas.

The radius of the nanoisland is set to be $R = 2\xi$, where ξ is the superconducting coherence length. The thickness is $L_z = 0.4R$. The resistance ratio for the interface between ferromagnet and superconductor is set to be $\zeta = 3$. The density of states for this system is calculated for an exchange field in the thickness direction, with values $h = 0.3\Delta$, 0.5Δ and 0.7Δ. The value Δ is the size of the energy gap for a bulk superconductor. The results are shown in figure 6.4, and show an enhanced density of states at the Fermi level, i.e., at $\varepsilon = 0$. In addition, a spin-split minigap is found. This is in agreement with numerical investigations of one dimensional systems[82,83,84,85,86,87,88], thereby demonstrating the the method is capable of solving 3D, non-rectangular problems.

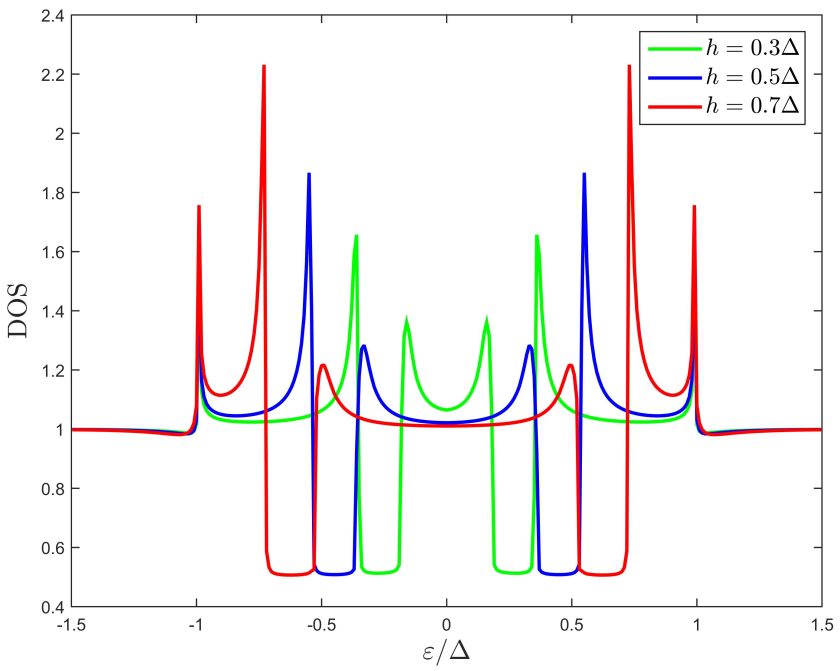
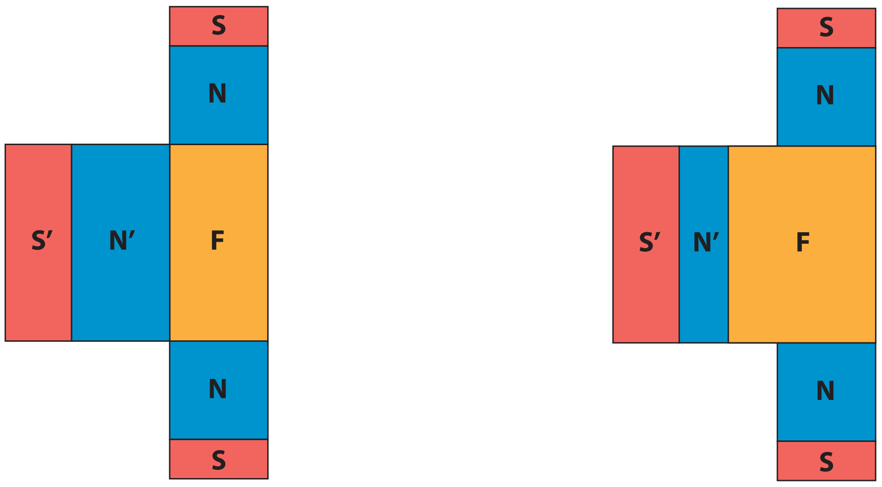


Figure 6.4: Density of states for a 3D cylindrical ferromagnet nanoisland with radius $R = 2\xi$ and thickness $L_z = 0.4R$ The exchange field is in the thickness direction.

6.1.3 An example where the finite element method gives incorrect results

In the previous sections, the finite element method has been shown to converge quickly with mesh refinements and yield correct results when compared to known benchmarks. Next is presented an analysis where the method does not give correct results. The model to be explored is once again a Josephson junction, except the metal sandwiched between superconductors contain a branch, shown in figure 6.5 and referred to as a T-geometry. It consists of both a ferromagnetic and a normal region.



(a) Incorrect (b) Correct

Figure 6.5: Two variations of a T-shaped geometry, one of which displays a poor modeling decision, producing erroneous results.

The exchange field in the ferromagnet is selected to be $h = 0.5\Delta$, oriented perpendicularly to the plane of the T-geometry. The resistance ratio of the Kupriyanov-Lukichev boundary conditions between each interface is $\zeta = 3$. The superconductors at the ends of the two prongs (S) are assumed to be in contact, and have a phase $\phi = \frac{\pi}{2}$ relative to the leftmost superconductor (S'). The results are shown in figure 6.6.

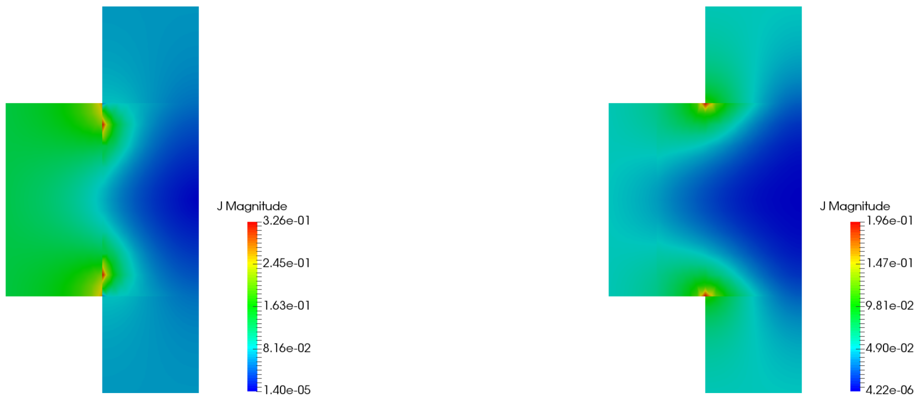


Figure 6.6: The current density in the two geometries, scaled by $J_0 = \frac{N_0 e D \Delta}{4\hbar}$.

At first sight, they seem reasonable. The numerical scheme has converged to within the specified error limit. The current density is symmetrical about the horizontal axis and travels between superconductors with different phase. Furthermore, no current is leaking into the vacuum - which would have been a sure sign that the boundary conditions are not satisfied. In the incorrect model there are, however, two suspicious points by the interface between the ferromagnet and the normal metal where the current density is noticeably higher than the surrounding area. This could indicate that the current is not conserved. To investigate whether the results are physical, the current entering and leaving the system is found by integrated along the interfaces to the superconductors. It is found that the current flowing into the system does not equal the current flowing out, and therefore something is wrong. Further insight is found by distorting the geometry based on the value of the current density, as is shown in figure 6.7 for the component flowing through the N'-F interface.

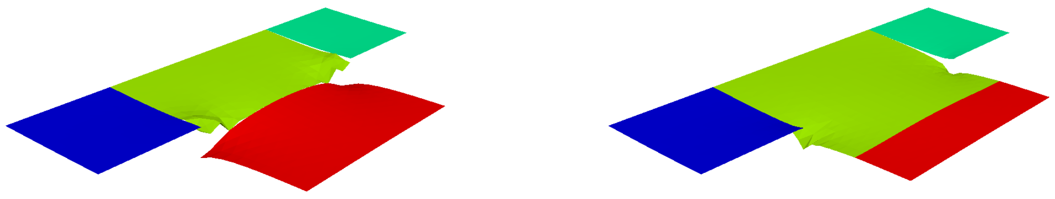


Figure 6.7: Deformed geometry based on the value of the current density flowing through the N'-F interface. It is emphasized that only the green and red areas should be continuous, as the other interfaces impose constraints on different current components.

The current density should be continuous through the interface, meaning that the ferromagnet (F), colored green in figure 6.7, should match the normal metal (N'), colored red. Obviously this is just a handy debugging tool, as the scale of the given distortion has not been given. It is, however, sufficient to identify the source of the problem in this particular case. Indeed, by inspecting the deformations, it is seen that the problem areas are found in the corners of ferromagnet, by the N' interface, as continuity is maintained to a greater degree along the center of the model. This leads to the realization that these two corner nodes are overconstrained, as they have to satisfy conservation of currents across the F-N' interface as well as across the F-N interface. Moving the green-red interface greatly reduces the problem, achieving current conservation.

6.2 Vortices in Josephson junctions

It was shown in section 6.1.1 that the numerical method developed is capable of reproducing results reported for a 2D Josephson junction with external flux. In this section, these results will be explored further. It was claimed that the minima of the pair correlation functions are vortices, i.e, localized regions in which superconductivity is suppressed and flux quanta may penetrate. This can be verified by looking at the density of states, as shown in figure 6.8. Plotted in the transverse direction along the center of the junction, at $x/L = 0$, it shows typical signs of the proximity effect. There is a suppression of the density of states for low energies as well as peaks at $\varepsilon/\Delta = \pm 1$. These features are small due to the pair breaking effect of the magnetic field. Nevertheless, they are noticable. The most interesting feature is, however, that at the exact location of the minima of the pair correlation function, the minigap is closed and the density of states is equal to one.

In other words, at each of these points there is a core which is in the normal state - a tell-tale sign of vortices.

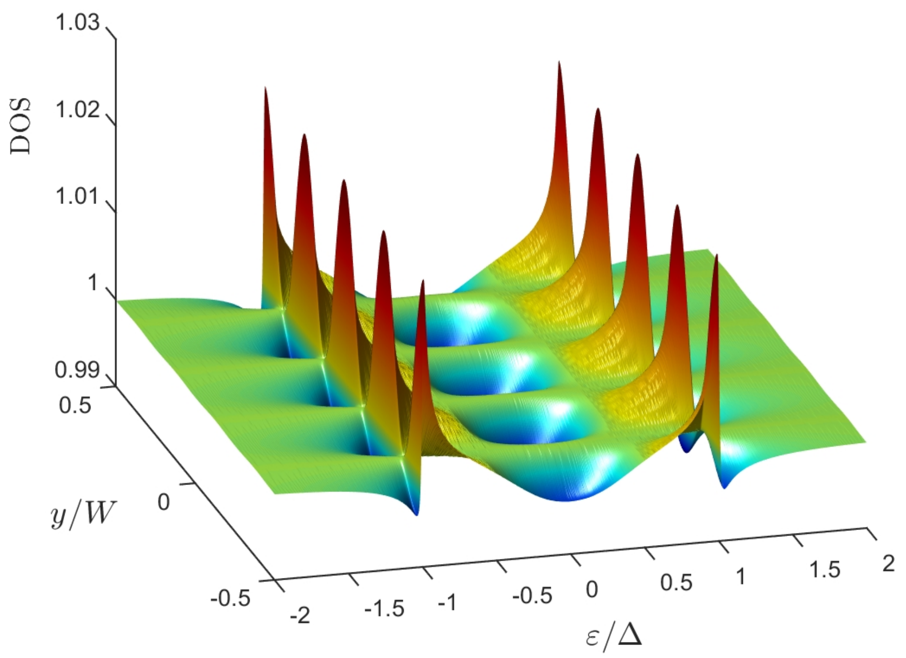


Figure 6.8: Density of state for a wide 2D Josephson junction along the transversal coordinate y.

Further evidence is found by considering the phase of the pair correlation function, which is given as

$$\theta = \arctan\left(\frac{\operatorname{Im}\Psi}{\operatorname{Re}\Psi}\right) \tag{6.7}$$

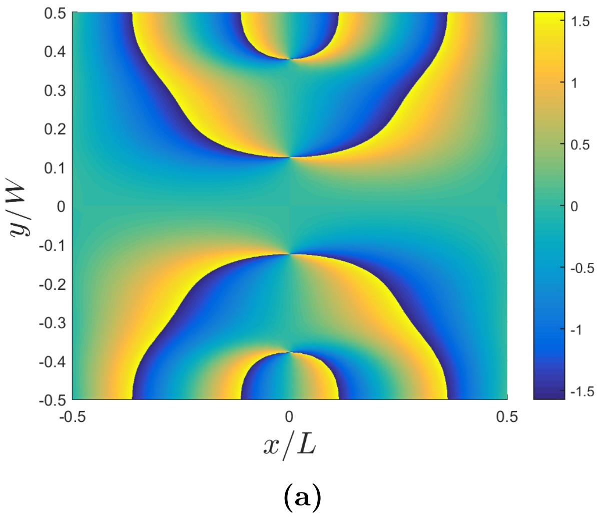
The winding number, or topological charge is found as the change in θ after having traversed along a closed path. This change is found by integrating $\nabla\theta$ along the selected path

$$\oint_C \nabla\theta = 2\pi N \tag{6.8}$$

where N is the winding number of the path C. It is observed that θ is bounded, with a range of $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, but discontinuous where ever $\operatorname{Re}\Psi = 0$. This enables a particularly easy way to compute the integral in equation 6.8. Indeed, for any bounded function $f(x)$ containing discontinuities at discrete location x_i, $i = 1, 2, 3, ...$ the integral of its derivative is found as

$$\begin{aligned}\int_a^b \frac{\partial f}{\partial x}dx &= \lim_{\epsilon\to 0}\int_a^{x_1-\epsilon}\frac{\partial f}{\partial x}dx + \sum_{i+1}^{n-1}\int_{x_i+\epsilon}^{x_{i+1}-\epsilon}\frac{\partial f}{\partial x}dx + \int_{x_n+\epsilon}^{b}\frac{\partial f}{\partial x}dx \\ &= f(b) - f(a) + \lim_{\epsilon\to 0}\sum_{i=1}^{n}\left[f(x_i-\epsilon) - f(x_i+\epsilon)\right]\end{aligned} \tag{6.9}$$

where the fundamental theorem of calculus is used in the last equality. The limit $\epsilon \to 0$ is well defined since the function f is bounded. For a closed path, $f(a) = f(b)$ and the integral becomes a sum of the value of the discontinuities encountered along the path. The phase of the pair correlation function is shown in figure 6.9a. It is seen that for any path encircling a minimum, two discontinuities of value π must be traversed, implying that the topological charge is 1. In figure 6.9b, streamlines of the current density is shown. It is seen that the current circles around the minima, which is also indicative of a normal core. It is concluded that the minima of the pair correlation function are indeed vortices. Furthermore, they bear a striking resemblance to the Josephson vortices. Their location are identical, and the current circulates around the vortices, creating an oscillating pattern with no net charge transport. On the other and, there are dissimilarities. The Josephson vortices do not have normal cores, which these vortices do. In addition, flux quantization is not found. The reason for this is that the numerical method does not take the effect of the screening currents created by the superconducting correlations into account. Such an effect would require a self-consistent computation of the magnetic field, as the screening currents generate a diamagnetic contribution. In the approximation used herein, only currents of low enough magnitude to not influence the external flux can be described, and the applied field is assumed omnipresent.



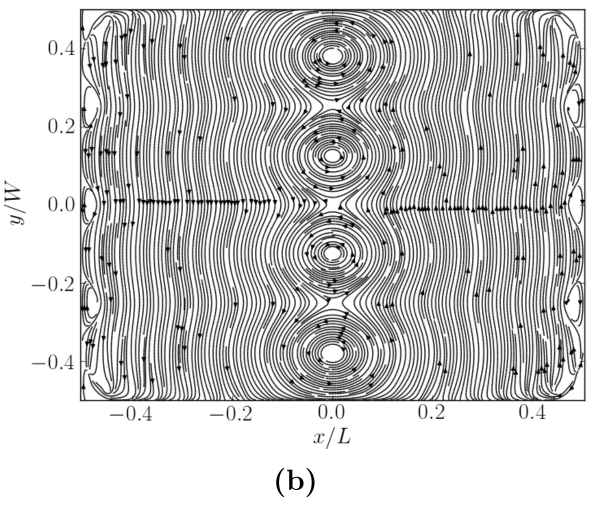


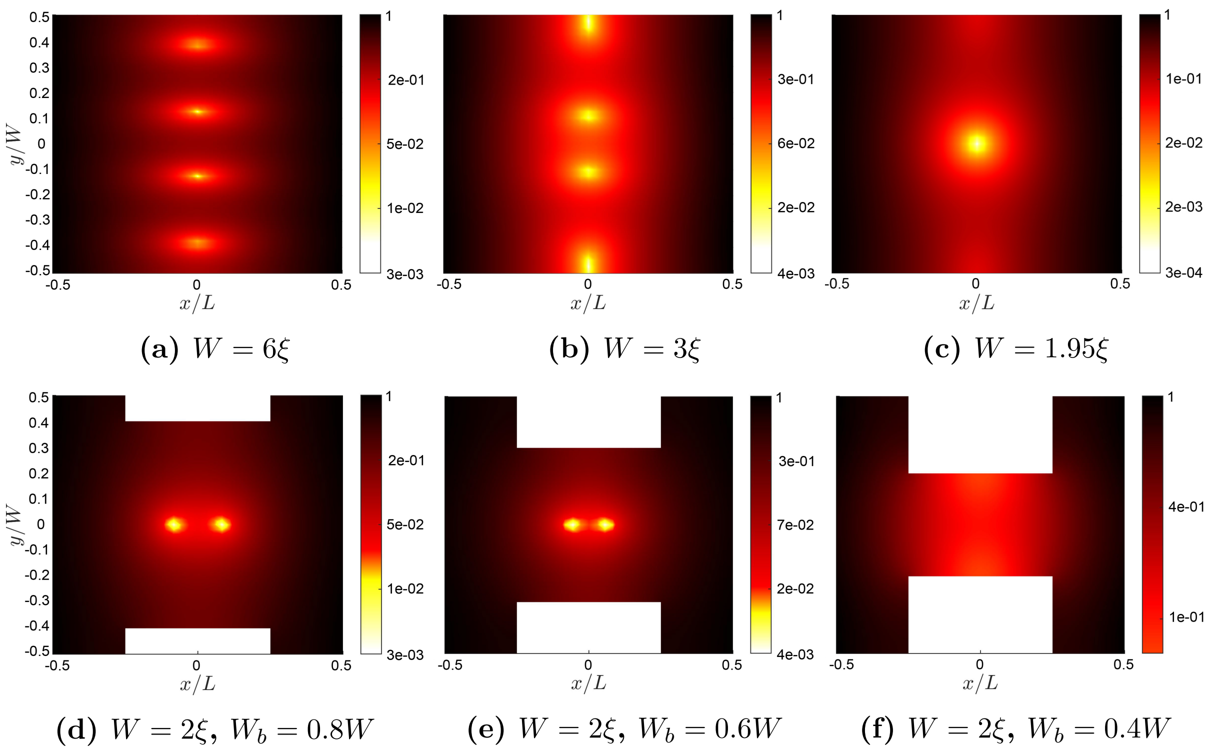
Figure 6.9: The phase of the pair correlation function Ψ and the current density, used in determining the presence of vortex. In (a) is shown the phase of the pair correlation function, and in (b) the direction of the current density.

6.2.1 Geometry dependence

The numerical code developed is now used to go beyond previous works and discover new effects. In Refs.[81,12] it was found that for a wide junction, $W \gg L$, a regular array of vortices was found, as shown in figure 6.2. Meanwhile, for a narrow junction no vortices appear. There is therefore a geometrical dependence in that the narrowing of the junction causes a transition to a vortex-less state. It is interesting to investigate how the vortices leave the system as the width decreases, and it turns out that they do so in a non-trivial way. Figure 6.10a-c shows the absolute value of the pair correlation function as W is reduced. There is no phase difference between the superconductors, and therefore no net current. The length of the system is $L = 2\xi$, where ξ is the superconducting coherence length. The first observation to be made is the appearance of a lattice structure in the vortices for wide junctions. This implies that the vortices repel, as they otherwise would have a tendency to group together. This is a feature they share with the Abrikosov vortices of type II superconductors[25]. As the width is decreased, it is seen that the two outermost vortices translate out of the system. The innermost vortices on the other hand approach each other. This behavior indicates that as the junction becomes narrower, less energy is available, and a threshold is passed after which four vortices is no longer energetically favorable, and so two are expelled.

The fact that two vortices exit the system simultaneously can be explained by the symmetry of the system. With no phase difference, the model is symmetric about both the x and the y axis. This means that the absolute value of the pair correlation function, being a gauge invariant observable, must also exhibit the same symmetries. For this reason, vortices can only appear symmetrically about the origin. When the outermost vortices translate out of the system, this must happen in a way which maintains this symmetry.

The two remaining vortices are seen to be forced closer to each other as W approaches L. This means that the edges of the junction repel the vortices more than they do each other, implying that for the given geometry, the presence of two vortices is energetically favorable regardless of their separation. In fact, for $W = 1.95\xi$, the vortices overlap within numerical precision, as shown in figure 6.10c. To demonstrate the ease with which the finite element method handles non-trivial geometries, further decrease of the width is restricted to a bottleneck region spanning the middle half of the normal metal region, shown in figure 6.10d-f. As the bottleneck width W_b is reduced, something surprising occurs. After having met at the center of the junction, the vortices are seen to separate along the x axis. Further decrease eventually forces the vortices back together at the center of the junction, for then to be expelled vertically in a similar manner as was seen with the first two vortices. This behavior can also be explained by the symmetry. With two vortices present, these must be constrained to be on either the x or the y axis. As the width is reduced, they eventually meet in the center of the junction, that is, at the origin. This is the only location the two vortices can overlap while still maintaining the symmetry of the pair correlation function. However, once in the origin, the vortices are free to separate along the x axis, which reduces the energy in the system. A continued decrease of W_b reduces the available energy, which is compensated by a larger vortex separation. However, the superconductors constitute boundary conditions where the pair correlation function is fixed at a constant value, this means that moving a vortex closer to the superconductors requires a higher curvature of the pair correlation function, which in turn requires more energy. Therefore, a turning point must take place, where the vortices can no longer move further apart as W_b is reduced. The energy requirement of two vortices becomes unsustainable, and the vortices seek out of the system. They may not leave through the superconductors, and are therefore forced back together at the origin, and expelled along the y axis.

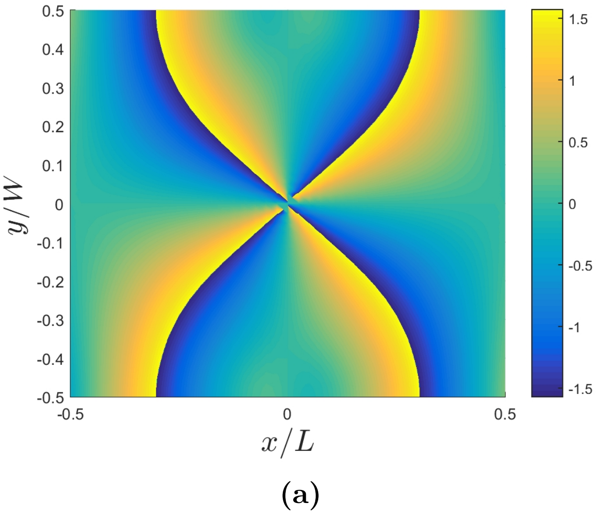


(a) $W = 6\xi$ (b) $W = 3\xi$ (c) $W = 1.95\xi$

(d) $W = 2\xi$, $W_b = 0.8W$ (e) $W = 2\xi$, $W_b = 0.6W$ (f) $W = 2\xi$, $W_b = 0.4W$

Figure 6.10: The absolute value of the pair correlation function for different values of the width W and the bottleneck width W_b.

The configuration where the vortices overlap is worth further investigation. The phase of the pair correlation function is shown in figure 6.11a and a streamline plot of the current density in figure 6.11b. The currents are seen to circulate around the location of the vortices, as is to be expected. The phase plot shows that four lines of discontinuity meet at the origin to within numerical precision, suggesting a topological charge of 2. It is plausible that, due to a frustration effect from the geometry, two $N = 1$ vortices are forcibly merged into a single, highly unstable $N = 2$ vortex. The slightest perturbation of the geometry causes the vortex to split along either the x or the y-axis.



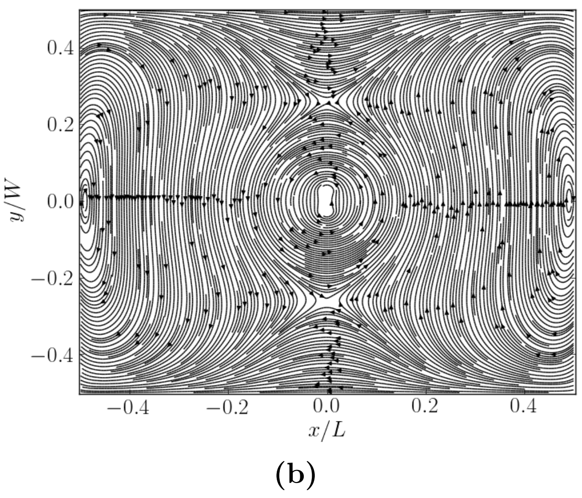
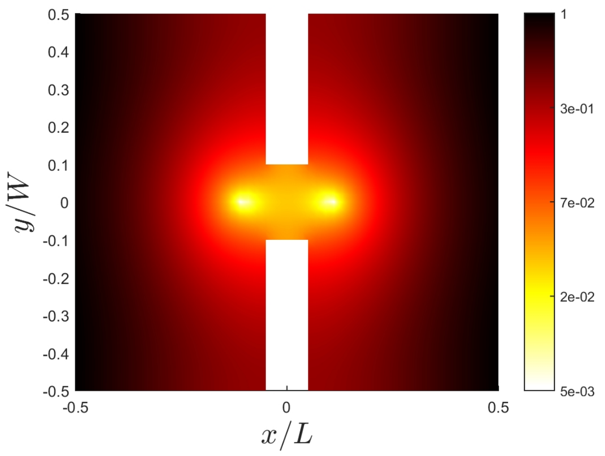
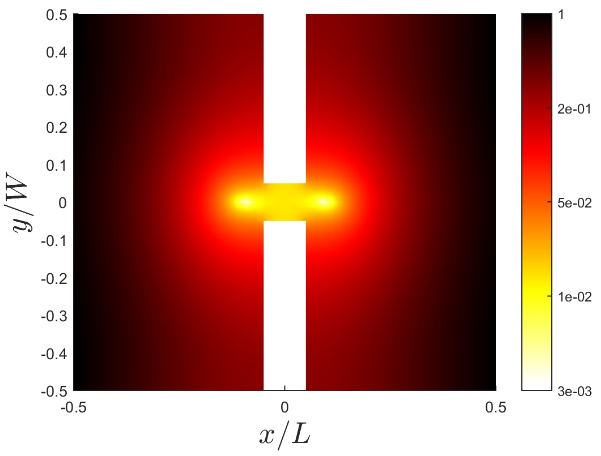


Figure 6.11: Additional information for the configuration where the vortices overlap; $W = 1.95\xi$. In (a) is shown the phase of the pair correlation function and in (b) the direction of the current density

The behavior of the vortices is qualitatively the same if the width W is uniformly decreased, rather than just the bottleneck W_b. It is reasonable to ask why that is the case. If repulsion from the edges force the vortices together at the origin and then away from each other along the x axis, then one should expect the vortices to remain stationary once they are pushed outside the bottleneck and into the wider regions. It turns out that this reasoning is correct. It was not observed for the bottleneck in figure 6.10 because it was too large, $L_b = \frac{L}{2}$ and the turning point came before the vortices reached out to the wide regions. In figure 6.12 is shown a geometry where the bottleneck length is significantly smaller, $L_b = 0.1L$. In this case, the vortices are indeed pushed into the wide regions, where they remain virtually immobile regardless of the bottleneck width. It is observed that for $L_b = \frac{L}{2}$ the vortices had already disappeared as $W_b = 0.4W$, shown in figure 6.10f, whereas with $L_b = 0.1L$, the vortices are present even when $W_b = 0.1W$.



(a) $W_b = 0.2W$



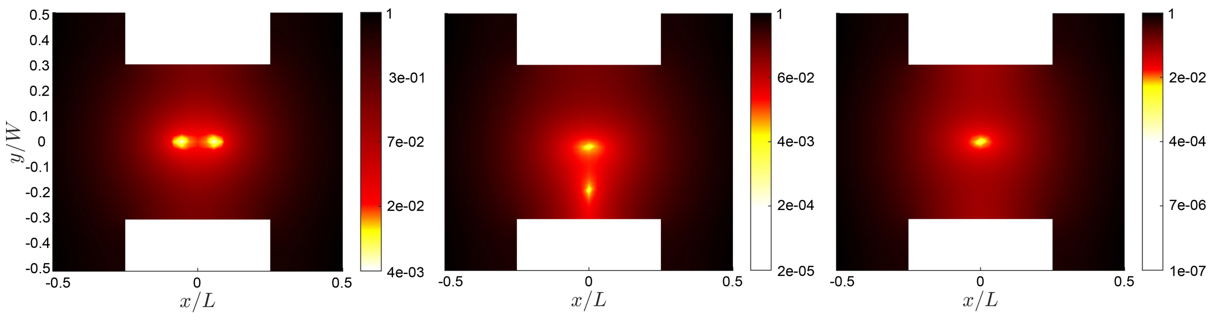
(b) $W_b = 0.1W$

Figure 6.12: An SNS junction with a bottleneck that has a length along the x axis of $L_b = 0.1L$.

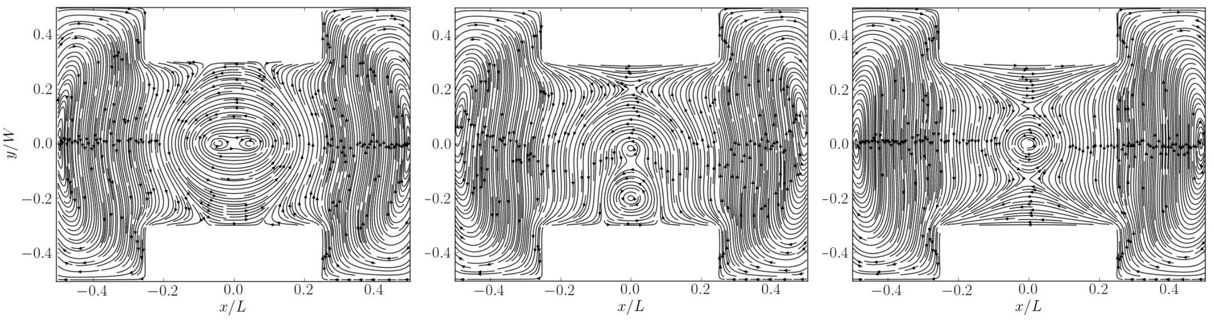
6.2.2 Phase dependence

The influence of varying the phase difference between the superconductors is considered for the bottleneck configuration with $W_b = 0.6W$. It is known that for Josephson vortices, the presence of a phase difference, and thereby induced supercurrent, causes a translation of the vortices along the transversal direction. In other words, the system needs not be symmetric about the x axis. The absolute value of the pair correlation function must still be symmetric about the y axis, however, since the addition of an extra phase factor should have no direct influence on it. However, the resulting supercurrent will influence the vortex locations. For the bottleneck, as the phase difference ϕ is increased from zero, it is seen that the vortices once again begin to converge towards the origin, where they eventually coalesce. A further increase of ϕ causes one of the vortices to translate out of the system, until only a single vortex remains at $\phi = \pi$. It is seen that also for the proximity induced vortices considered herein, there is a tendency for translation of the vortices when ϕ is increased. The vortices must move symmetrically on the x axis, but can move asymmetrically on the y axis as is seen in figure 6.13a-c. The conclusion is that the phase difference between the superconductors can be used to alter, not only the location of the vortices, but also the number of vortices present in the system. In figures 6.13d-f are shown streamline plots of the current density, and in figures 6.13g-i the phase of the pair correlation function, both verifying that minimas in question are in fact vortices.

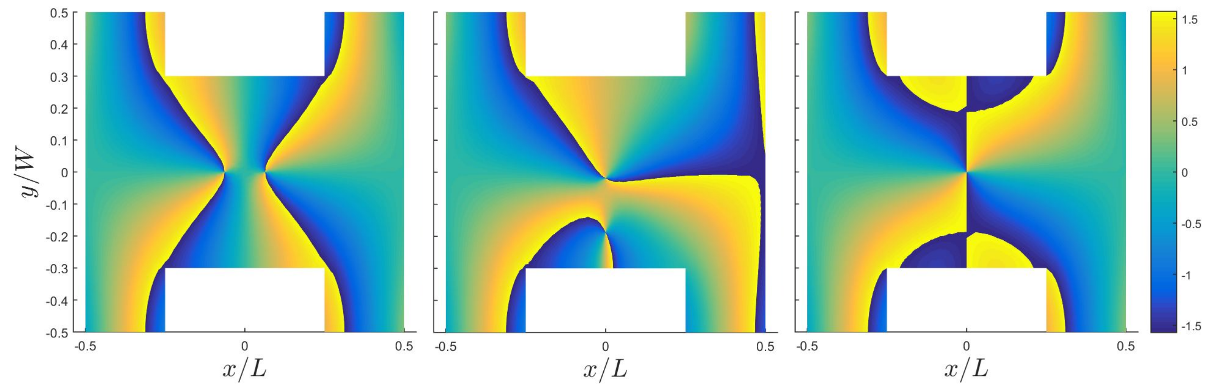
From the streamline plots of the current density in figure 6.13d-f an interesting effect appears which was not visible in the rectangular geometry. Near the vortices, it is seen that the current circulates in a counter-clockwise fashion. However, as is particularly clear in figures 6.13d and 6.13f, the current circulates in the opposite direction around the geometry. In other words, near the vortices, where the superconductng correlations are low, currents are induced by the magnetic field. Far away, screening currents dominate, moving clockwise around the edges of the geometry. It is this mechanism that would cause flux quantization in the vortices, as the external magnetic field would be canceled by the induced field from these currents. By assumption, however, the screening currents do not influence the external magnetic field, thereby causing the abrupt change in the circulation pattern of the current as one moves away from a vortex.



(a) $\phi = 0$ **(b)** $\phi = \frac{\pi}{2}$ **(c)** $\phi = \pi$



(d) $\phi = 0$ **(e)** $\phi = \frac{\pi}{2}$ **(f)** $\phi = \pi$



(g) $\phi = 0$ **(h)** $\phi = \frac{\pi}{2}$ **(i)** $\phi = \pi$

Figure 6.13: The absolute value of the pair correlation function, current density streamlines and the phase of the pair correlation function for a bottleneck with $W_b = 0.6W$ and varying phase differences between the superconductors.

Comments on the numerical solution

In this section, several surface plots of the pair correlation function have been shown, which have demonstrated the presence of vortices in various model geometries and phase differences. By inspecting these plots, it is seen that the value of the minimas are not identical to zero. In fact, their values fluctuate. This is caused by the mesh used, and where the nodes are located in regards to a given minimum. The solution within an element is found by interpolation between the nodes, where the solution is computed. This means that for a highly localized effect, which is entirely contained within a single element, there are limits to how accurately the behavior can be resolved. If higher accuracy is required, a denser mesh should be used in the region in which the effect is located. In the present analyses, more pronounced minimas can be achieved by generating a mesh so that a node is placed in the center of each minimum. This has not been done, however, as the vortices are clearly visible, and the results are complemented by other observables.

6.3 Supercurrents and magnetization in the presence of in-plane spin-orbit coupling

Several works have considered the effect of spin-orbit coupling (SOC) on one dimensional models of superconducting hybrid structures. This restricts the types of systems that can be investigated to those where the spin-orbit interaction lies in a plane orthogonal to the 1D model, i.e., out-of-plane SOC. Here are presented the results for a two dimensional model with in-plane SOC. The geometry is shown in figure 6.14. It consists of a normal metal sandwiched between two ferromagnets with spin-orbit coupling, which is again sandwiched between two superconductors, thus creating a Josephson junction. Kupriyanov-Lukichev boundary conditions are used at every material interface, with a resistance ratio of $\zeta = 3$. The distance between the superconductors is $L = 4\xi$, with the ferromagnets each having a length of $L_F = \xi$. The width of the junctions is $W = 2\xi$.

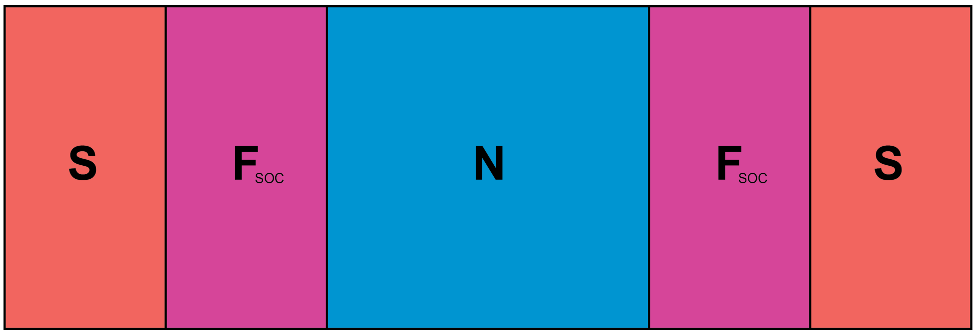
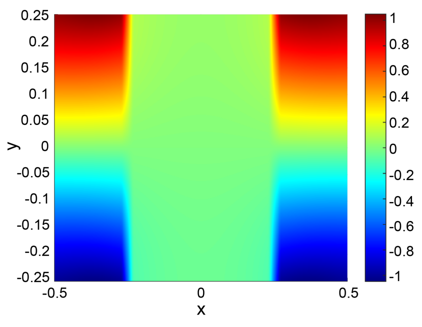


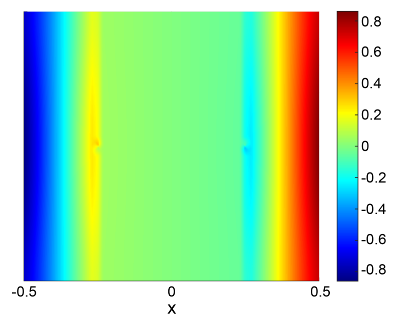
Figure 6.14: The geometry used for the numerical investigation of the effects of in-plane spin-orbit coupling.

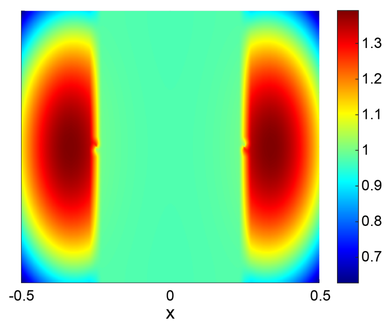
A linear combination between Rashba and Dresselhaus spin-orbit coupling is used, leading to[69]

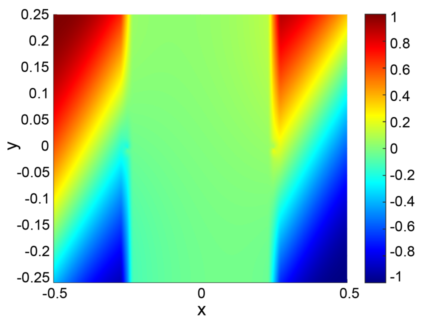
$$\boldsymbol{A} = a\left(\tau_x \cos\chi + \tau_y \sin\chi\right)\hat{x} - a\left(\tau_x \sin\chi + \tau_y \cos\chi\right)\hat{y} \tag{6.10}$$

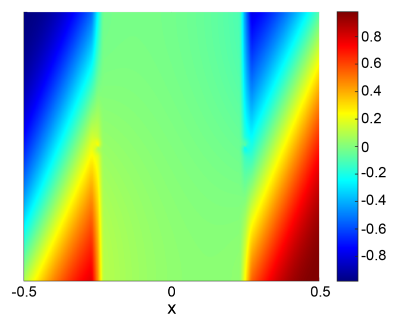
where $\hat{x}$ and $\hat{y}$ are unit vectors in the xy-plane. It is emphasized that the Josephson junction also lies in this plane. For an angle $\chi = 0$, the interaction reduces to a pure Dresselhaus coupling, whereas $\chi = \pm\frac{\pi}{2}$ gives a Rashba coupling. The strength of the interaction is held fixed at $a = 2/\xi$. The induced magnetization M_x, M_y and M_z are calculated by using equation (4.100) and are shown for select values of the spin-orbit angle χ in figure 6.15. It is noted that for the purpose of performing a symmetry analysis, a reduced length of $L = \xi$ is used, while otherwise maintaining the system proportions. The reason for this is that the results become more pronounced. The behavior is qualitatively the same for $L = 4\xi$. It is interesting to note that the angle χ is observable in the symmetry of the magnetization within the regions containing spin-orbit coupling. For $\chi = 0$, M_x is symmetric about the y-axis, M_y about the x-axis and M_z about both axes, as shown in figures 6.15a to 6.15c, respectively. As χ increases, the symmetry axes are rotated accordingly, as can be seen in figures 6.15d to 6.15f. Once $\chi = \frac{\pi}{2}$, the symmetries of M_x and M_y have switched, and M_z is identical to the result for $\chi = 0$. This is shown in figures 6.15g to 6.15i and corresponds to a 90° rotation of the symmetry axes.

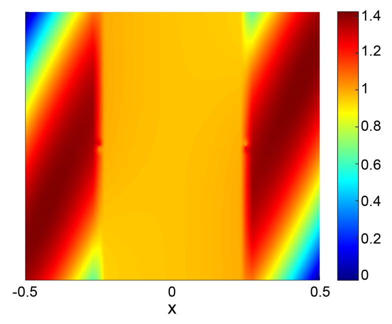

(a) M_x, $\chi = 0$

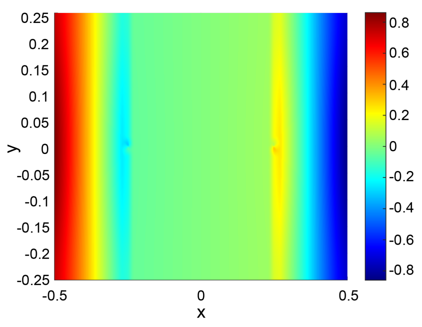

(b) M_y, $\chi = 0$

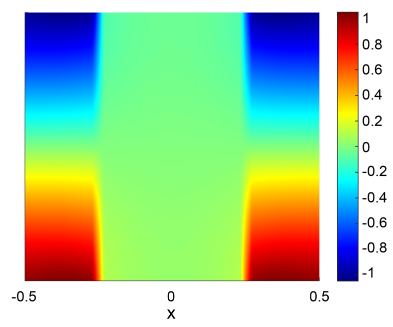

(c) M_z, $\chi = 0$


(d) M_x, $\chi = \frac{\pi}{4}$


(e) M_y, $\chi = \frac{\pi}{4}$


(f) M_z, $\chi = \frac{\pi}{4}$


(g) M_x, $\chi = \frac{\pi}{2}$


(h) M_y, $\chi = \frac{\pi}{2}$

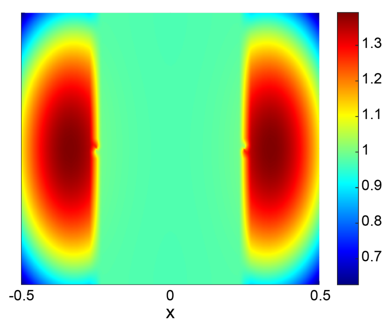

(i) M_z, $\chi = \frac{\pi}{2}$

Figure 6.15: The magnetization for different values of the spin-orbit angle χ. The results are scaled by $M_0 = \frac{g\mu_B N_0 \Delta}{8}$, and the lengths by ξ.

To understand this behavior, it is instructive to simplify the Usadel equation, given in equation (4.102) to a form where its symmetries are more apparent. This is done by considering the weak proximity effect, where it is assumed that the Green function is only slightly perturbed from its value in the normal state. The retarded quasiclassical Green function can then be approximated to

$$\hat{g}^R = \begin{pmatrix} g & f \\ -\tilde{f} & -\tilde{g} \end{pmatrix} \approx \hat{g}_0 + \hat{g}_1 = \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix} + \begin{pmatrix} 0 & f \\ -\tilde{f} & 0 \end{pmatrix} \tag{6.11}$$

where $\hat{g}_0 = \hat{\rho}_3$ is the normal state solution and by definition, the elements of the 2×2 anomalous Green functions, f and $\tilde{f}$ are small, i.e., $f_{ij}, \tilde{f}_{ij} \ll 1$. By comparing with equation (4.101) it is seen that $N = \tilde{N} = I$, $\gamma = \frac{1}{2}f$ and $\tilde{\gamma} = \frac{1}{2}\tilde{f}$. Furthermore, the anomalous Green functions can be parametrized by[89,90,69]

$$f = (f_s I + \boldsymbol{f}_t \cdot \boldsymbol{\tau})\, i\tau_y \tag{6.12}$$

where f_s and $\boldsymbol{f}_t$ are scalars in spin space and $\boldsymbol{\tau}$ is a vector of Pauli matrices. A similar parametrization for $\tilde{f}$ is found by tilde conjugation. Equation (6.12) separates the contribution coming from the singlet configuration of the Cooper pair, represented by f_s, and the part coming from the triplet configuration, given by $\boldsymbol{f}_t$. Inserting into equation (4.102) and by using the trace identities of the Pauli matrices, derived in appendix A.1, the Usadel equation in the weak proximity limit becomes

$$D\nabla^2 f_s + 2i\varepsilon f_s + 2i\boldsymbol{h} \cdot \boldsymbol{f}_t = 0 \tag{6.13a}$$

$$D\left[\nabla^2 f_x - 4a^2 f_x + 4a^2 f_y \sin 2\chi + 4ia\left(\frac{\partial f_z}{\partial x}\sin\chi - \frac{\partial f_z}{\partial y}\cos\chi\right)\right] + 2i\varepsilon f_x + 2ih_x f_s = 0 \tag{6.13b}$$

$$D\left[\nabla^2 f_y - 4a^2 f_y + 4a^2 f_x \sin 2\chi - 4ia\left(\frac{\partial f_z}{\partial x}\cos\chi - \frac{\partial f_z}{\partial y}\sin\chi\right)\right] + 2i\varepsilon f_y + 2ih_y f_s = 0 \tag{6.13c}$$

$$D\left[\nabla^2 f_z - 8a^2 f_z + 4ia\left(\frac{\partial f_y}{\partial x} + \frac{\partial f_x}{\partial y}\right)\cos\chi - 4ia\left(\frac{\partial f_x}{\partial x} + \frac{\partial f_y}{\partial y}\right)\sin\chi\right] + 2i\varepsilon f_z + 2ih_z f_s = 0 \tag{6.13d}$$

By performing the same linearization on the magnetization $\boldsymbol{M}$ one finds that it is zero to the leading order. The next order term is found by expanding $\hat{g}^R$ to higher order in g and $\tilde{g}$ and using equation (4.10). This leads to

$$(\hat{g}_0 + \hat{g}_2)(\hat{g}_0 + \hat{g}_2) = \hat{I} \tag{6.14}$$

where $\hat{g}_2$ is given as

$$\hat{g}_2 = \begin{pmatrix} \mathrm{d}g & f \\ -\tilde{f} & -\mathrm{d}\tilde{g} \end{pmatrix} \tag{6.15}$$

By multiplying out it is found that the matrices $\mathrm{d}g$ and $\mathrm{d}\tilde{g}$ can be expressed in terms of the anomalous Green functions as

$$\mathrm{d}g = \frac{1}{2} f \tilde{f}, \quad \mathrm{d}\tilde{g} = \frac{1}{2} \tilde{f} f \tag{6.16}$$

Inserting equation (6.16) into equation (4.100), and using the identities equation (2.41) and equation (2.42), the magnetization is found to be

$$\boldsymbol{M} = \frac{g\mu_B N_0}{4} \int d\varepsilon \ \mathrm{Re}\,\mathrm{Tr}\left[\boldsymbol{\tau} f \tilde{f}\right] \tanh\frac{\beta\varepsilon}{2} \tag{6.17}$$

Finally, by inserting equation (6.12) and computing the trace, this becomes

$$\boldsymbol{M} = g\mu_B N_0 \int d\varepsilon \ \mathrm{Re}\left[f_s \tilde{\boldsymbol{f}}_t\right] \tanh\frac{\beta\varepsilon}{2} \tag{6.18}$$

The symmetry of the magnetization is therefore found from the symmetry of the product $f_s \tilde{\boldsymbol{f}}_t$. The geometry discussed here is symmetric about the x-axis, and with no phase difference between the superconductors, it is also symmetric about the y-axis. For $\chi = 0$ it is seen from figure 6.15c that M_z is symmetric about both the x- and the y-axis. Based on this, it seems reasonable to make the ansatz that f_s also shares these symmetries, which implies that f_z does as well. This is consistent with equation (6.13a). From a term by term inspection of equation (6.13d) it is deduced that $\frac{\partial f_x}{\partial y}$ and $\frac{\partial f_y}{\partial x}$ must have the same symmetries as f_z. This means that f_x must be antisymmetric about the x-axis and symmetric about the y-axis. Similarly, f_y is symmetric about the x-axis and antisymmetric about the y-axis. These observations are precisely reflected in the plots of M_x and M_y shown in figures 6.15a and 6.15b. For $\chi = \frac{\pi}{2}$, M_z is identical to the results for $\chi = 0$. Using the same ansatz, the symmetries of M_x and M_y may once again be deduced from equation (6.13d). For general χ, the situation is not so simple. From figure 6.15f it is seen that M_z is not symmetric about the x- or y-axis separately, but instead has inversion symmetry, i.e., it is invariant under the transformation $\boldsymbol{r} \to -\boldsymbol{r}$. This must therefore also be the case for the product $f_s \tilde{f}_z$. Furthermore, f_s and $\tilde{f}_z$ must have the same symmetries for equation (6.13a) to be satisfied. This is because a symmetry of the solution must be reflected in the equation. The conclusion is that spin-orbit coupling indirectly influences the singlet component, through the coupling with f_z.

The coupling between f_z and the other two triplet components, f_x and f_y is unique for an in-plane SOC. This has interesting consequences. It turns out that even without a phase difference between the superconductors, a spincurrent can flow through the normal metal, where it is well defined and conserved. Shown in figure 6.16 are the spin currents for different values of the spin-orbit angle χ, ranging from 0 to $\frac{\pi}{2}$.

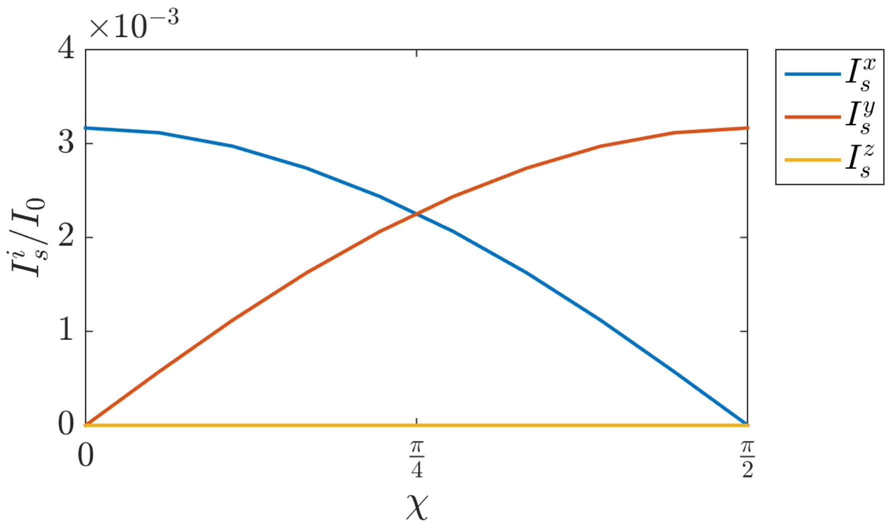


Figure 6.16: Spincurrents for various spin-orbit angles χ. The parameter I_0 is given as $I_0 = \frac{N_0 D\Delta}{8}$.

It is seen that the current of spin polarized in the z-direction, I_s^z, remains identically zero regardless of the value of χ. Moreover, I_s^x and I_s^y vary sinusoidally, perfectly out of phase. This behavior can be explained by the induced magnetization $\boldsymbol{M}$. As figure 6.15 shows, only M_z has a value different from zero in the normal metal. This has been induced by the exchange field in the surrounding ferromagnets, which also points in the z-direction. In the ferromagnets, the spin-orbit coupling generates a magnetization also in the x- and y-direction. By averaging over the magnetization in the ferromagnet and in the normal metal, it is found that for pure Dresselhaus SOC, $\chi = 0$, the net magnetization in the ferromagnet has components in the y- and z-direction, $\langle M_y \rangle \neq 0$ and $\langle M_z \rangle \neq 0$, whereas $\langle M_x \rangle = 0$. In the normal metal, the averaged magnetization only has a component in the z-direction. This means that, due to $\langle M_y \rangle$, there is a torque between the spins in the ferromagnet and those in the normal metal, since the system will favor alignment of spins. In addition, due to the spin-orbit coupling and the exchange field, the spincurrents are not conserved in the ferromagnets, so that they may act as a source and drain. This is important, as the boundary conditions do not allow spin currents to enter the superconductor.

At the left F-N interface, approaching from the left, spincurrents are created that enters the normal metal due to the spin torque caused by the $y \to z$ spin misalignment, in a way which resembles the process of spincurrents passing between misaligned ferromagnets[91,92]. At the right interface, when approaching from the normal metal side, a $z \to y$ spin misalignment is found. Combined with the antisymmetric M_y (figure 6.15b), this means that the spincurrents are directed from the normal metal to the ferromagnet at this interface. The result is that a uniform spincurrent passes through the normal metal. It is noted that while $\langle M_x \rangle$ cancels, there are local variations that also generate spin current densities polarized in the y-direction, however no net spincurrent I_s^y is found upon integration over the cross section,

For $\chi = \frac{\pi}{2}$, the net magnetization lies in the xz-plane, thus generating spincurrents polarized in the y-direction. For arbitrary χ between 0 and $\frac{\pi}{2}$, various proportions of I_s^x

and I_s^y can be generated, as dictated by the distribution of $\boldsymbol{M}$. It is with this physical picture apparent why $I_s^z = 0$, as it is a misalignment relative to the magnetization in the normal metal that causes the spin currents.

The effect of the exchange field is investigated next. For a pure Dresselhaus SOC, the exchange fields in the two ferromagnets is rotated in the yz-plane according to $\boldsymbol{h} = h\left(\hat{y}\sin\theta + \hat{z}\cos\theta\right)$, where $\hat{y}$ and $\hat{z}$ are unit vectors. The results are shown in figure 6.17a for $\theta \in \left[0, \frac{\pi}{2}\right]$. It is observed that the value of I_s^x is influenced by changing θ. The effect is quite faint for the given dimensions, and to obtain an upper limit, results are also shown for a configuration where the distance between the superconductors is reduced to $L = \xi$. This is shown in figure 6.17b, where it is seen that the value of I_s^x nearly doubles.

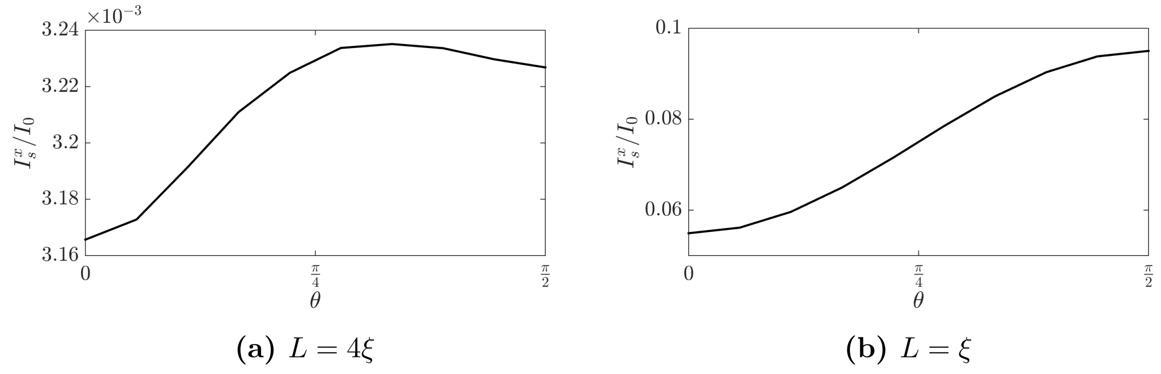


Figure 6.17: Spincurrents for various orientations of the exchange field in the two ferromagnets. The exchange fields are parallel, and given as $\boldsymbol{h} = h\left(\hat{y}\sin\theta + \hat{z}\cos\theta\right)$.

The other polarizations of the spincurrent, I_s^y and I_s^z, are identical to zero independent of θ. This means that as the exchange field rotates, the net induced magnetization rotates accordingly so that only I_s^x is generated. To bolster this argument, the expression for the spincurrents in the normal metal are derived in the weak proximity limit to be

$$\boldsymbol{I}_s = N_0 D t \int dy d\varepsilon \ \mathrm{Im}\left[\boldsymbol{f}_t \times \frac{\partial}{\partial x}\tilde{\boldsymbol{f}}_t\right] \tag{6.19}$$

where t is the thickness of the material. For a spincurrent to be non-zero in the normal metal, two symmetry conditions must be met for pure Rashba and pure Dresselhaus SOC, where f_s and $\boldsymbol{f}_t$ are either symmetric or antisymmetric. Firstly, the integrand in equation (6.19) must be symmetric about the x-axis, otherwise the y-integration cancels. Secondly, the integrand must be symmetric about the y axis. The reason for this is that due to current conservation, the spincurrent must be constant within the normal metal. An antisymmetric spincurrent must be zero on the y-axis, and hence it must be zero in the entire normal metal. A non-zero spincurrent, therefore requires an integrand that is symmetric about both axes. Looking at the y-component of equation (6.19) shows that it depends on the products $f_x \frac{\partial \tilde{f}_z}{\partial x}$ and $f_z \frac{\partial \tilde{f}_x}{\partial x}$. This means that for the symmetry requirements of I_y to be fulfilled, f_x and f_z must have opposite symmetries about the x-axis and equal symmetries about the y-axis. By inspecting figures 6.15a and 6.15c it is seen that this is not the case. A similar analysis shows that I_s^z does not satisfy the requirements either. On the other hand, I_s^x does. The conclusion is that the symmetry of the spin-orbit coupling forces I_s^y and I_s^z to be zero. For a pure Rashba SOC, it can similarly be deduced that $I_s^x = I_s^z = 0$.

It is also interesting to investigate the effect of a phase difference between the superconductors. This is shown for the Dresselhaus SOC in figure 6.18. It is found numerically that the x-polarized spincurrent satisfies the relation $I_s^x = I_1 + I_2 \cos\phi$, which is in agreement with a recent study of a 1D system with perpendicular SOC[93]. On the other hand, I_s^y and I_s^z remain equal to zero, which is a consequence of the 2D in-plane SOC.

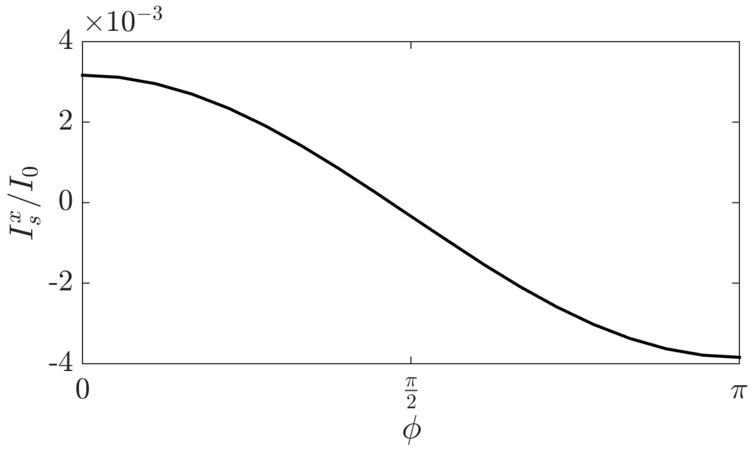


Figure 6.18: Phase dependence of the x-polarized spin current. The other polarizations are identically equal to zero.

6.4 Nanoislands in 3D

A fully 3D configuration is considered next, in which superconducting nanoislands are placed on a ferromagnet. It is emphasized that, to the best of our knowledge, this is the first time the 3D Usadel equation has been solved for a configuration that cannot be reduced to an effective 1D problem. The ferromagnet has dimensions $L_x \times L_y \times L_z = 10\xi \times 7\xi \times \xi$. The superconducting nanoislands have a dimension of $2.5\xi \times 2.5\xi$, which is assumed large enough to approximate them as bulk superconductors. Kupriyanov-Lukichev boundary conditions are once again used at the interfaces between superconductors and ferromagnet, with a resistance ratio of $\zeta = 1.5$. While this seems like too low a value for the Kupriyanov-Lukichev boundary conditions to be valid, a study has shown the results to be accurate also for moderately to highly transparent interfaces[94]. Two nanoislands are placed onto the ferromagnet in two different arrangements. In one configuration, both superconductors are placed on the same side, as is seen in figure 6.19a. In the other, one of the superconductors is moved to the opposite side, shown in figure 6.19b. The motivation for this study is to investigate the supercurrent flowing in the ferromagnet. For this reason, the nanoislands are given a phase difference of $\phi = \frac{\pi}{2}$.

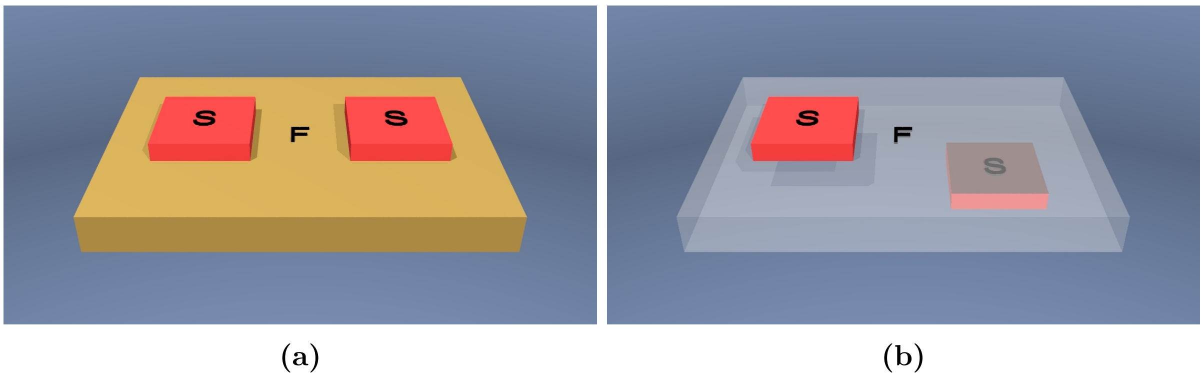


Figure 6.19: The two configurations considered in the 3D analysis of superconducting nanoisland-ferromagnet system.

To create these configurations experimentally, the ferromagnet can be grown on a substrate. For the analysis with both nanoislands on the same side, superconducting leads may then be placed on top of the ferromagnet, as illustrated in figure 6.20. As the important region is located between the superconductors, this setup should be a reasonable representation of the theoretical model. For the configuration with superconductors on opposite sides, one of the superconducting leads must be placed on the substrate, with the ferromagnet grown on top of it.

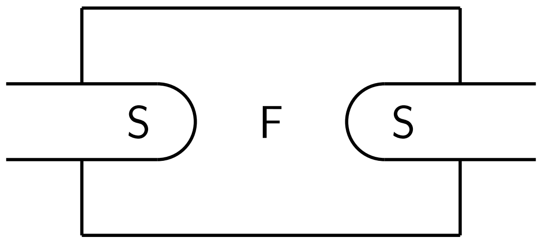


Figure 6.20: Illustration of the experimental setup for the theoretical model.

Two different magnetizations will be considered, a uniform distribution of the exchange field and a non-uniform, as shown in figure 6.21. Both fields are pointing in the thickness direction. The uniform exchange field has a strength of $h = 5\Delta$. The non-uniform exchange field is constant through the thickness of the model, but has a value of 15Δ within a horizontal radius of one coherence length ξ from the center of the ferromagnet. Farther away from the center, the exchange field falls off quickly. Such a distribution can be created experimentally by placing a strong ferromagnet in contact with a normal metal. An approximately constant magnetization will then be induced through the thickness of the metal, in the region beneath the ferromagnet. In the rest of the metal, the ability of the ferromagnet to magnetize quickly abates the farther away one moves, thus creating the distribution shown in figure 6.21.

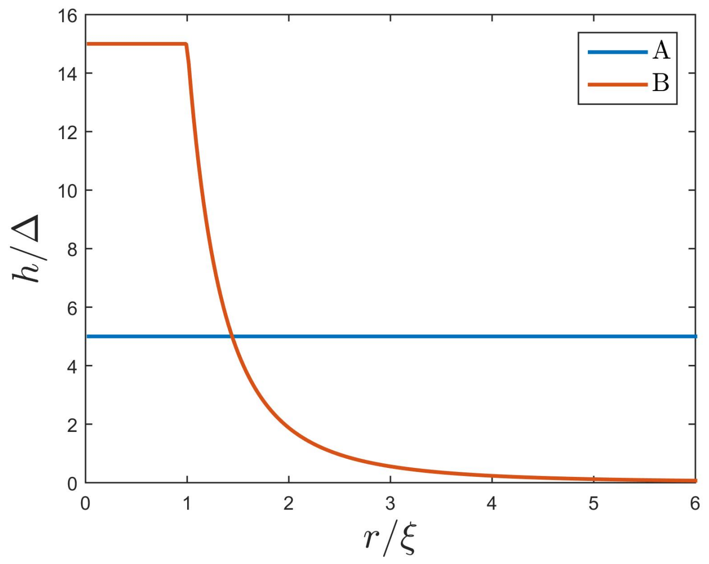


Figure 6.21: The distribution of the exchange field considered.

The charge current density is shown in figure 6.22 for two configurations

- A The superconductors are placed on the same side of the ferromagnet. The exchange field is uniform.
- B The superconductors are placed on opposite sides of the ferromagnet. The exchange field is non-uniform.

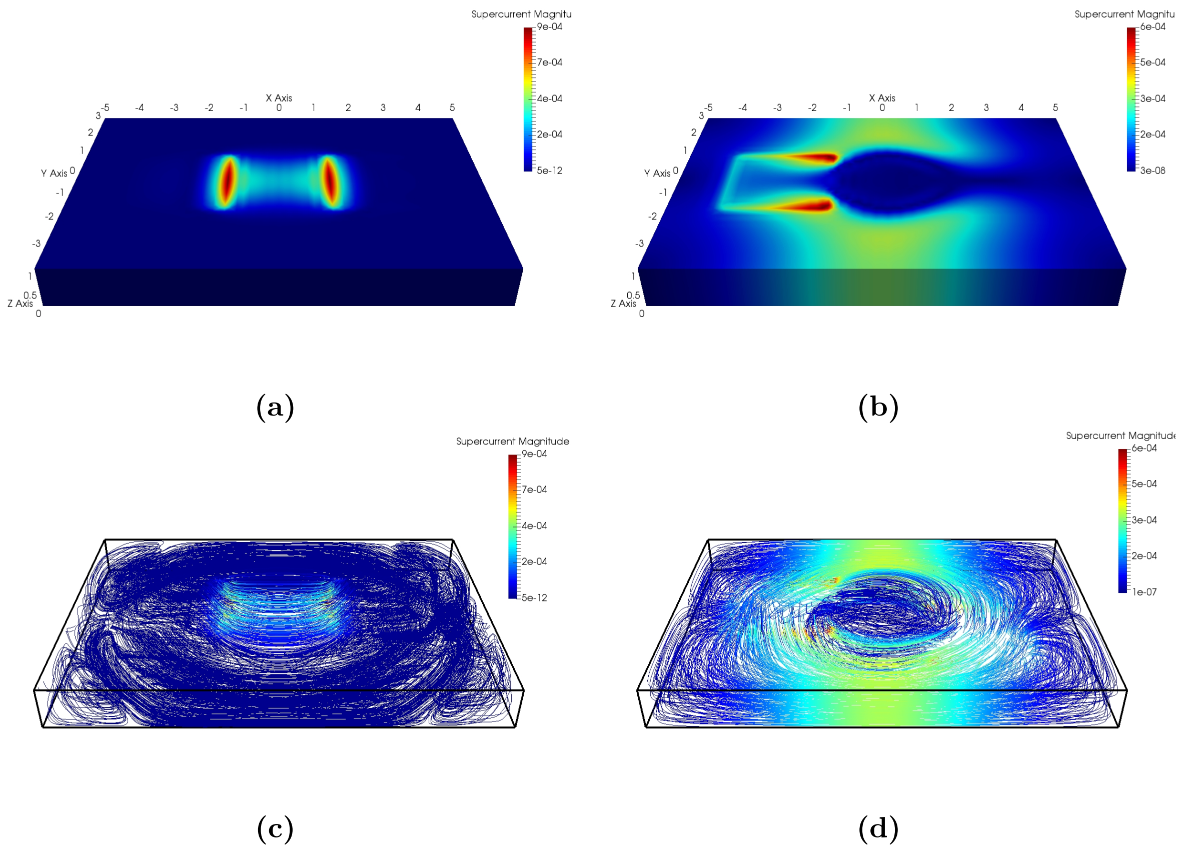
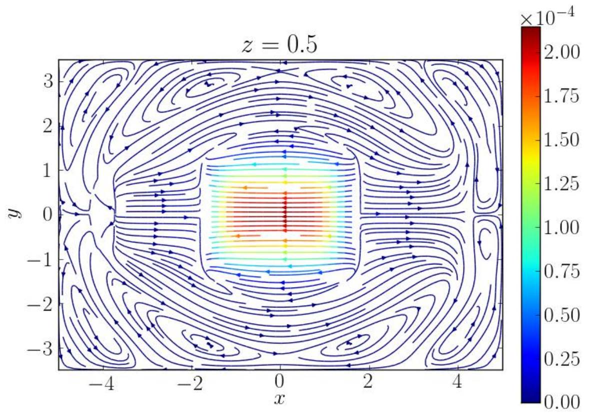
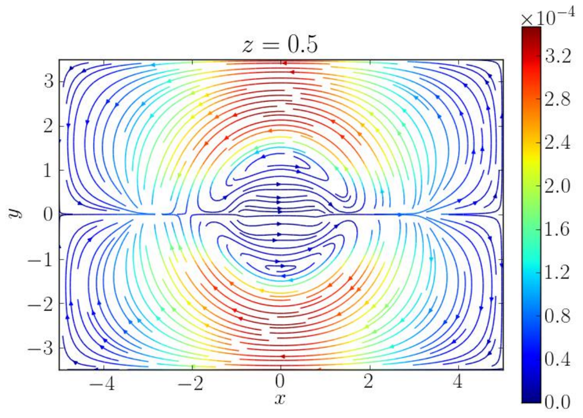


Figure 6.22: Charge current density in the ferromagnet for the two configurations A and B. The phase difference between the superconductors are $\phi = \frac{\pi}{2}$. All currents are scaled by $J_0 = \frac{N_0 e D \Delta}{8\hbar}$, and all lengths by ξ.

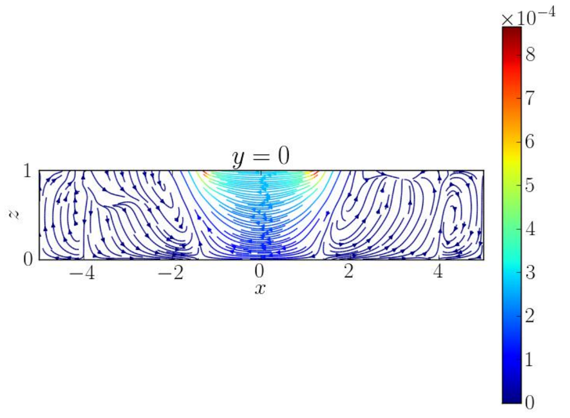
In configuration A, the charge current density is shown in figures 6.22a, 6.22c, 6.23a and 6.23c. It is seen that with a uniform exchange field, the current takes the shortest route between the superconductors. However, since the boundary conditions constrain the currents to enter and exit the superconductors vertically, the current is forced to arc into the thickness of the ferromagnet. This is shown in figure 6.23c, in which the current density is plotted on a slice through the model, along the y-axis. In configuration B, a more interesting pattern in the current density emerges, as can be seen in figures 6.22b, 6.22d, 6.23b and 6.23d. Within a cylinder of radius ξ surrounding the center of the ferromagnet, the exchange field is $h(\boldsymbol{r}) = 15\Delta$. This has the effect that the supercurrent is pushed out from the central region, and instead follows a semicircular path between the superconductors. The exchange field has a pair breaking effect on the Cooper pairs, and therefore functions as a hindrance for the supercurrent. Rather than taking the shortest route, as was the case with uniform magnetization, the supercurrent now takes the easiest route. In other words, the presence of an exchange field influences the supercurrent in an analogous way as a resistance influences a normal current, and may have application in exerting a greater degree of control over the supercurrent path.



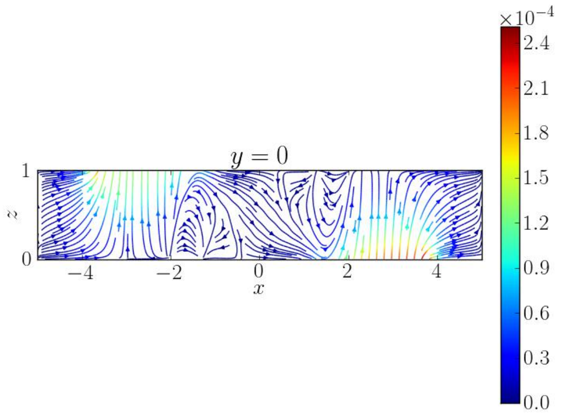
(a)



(b)



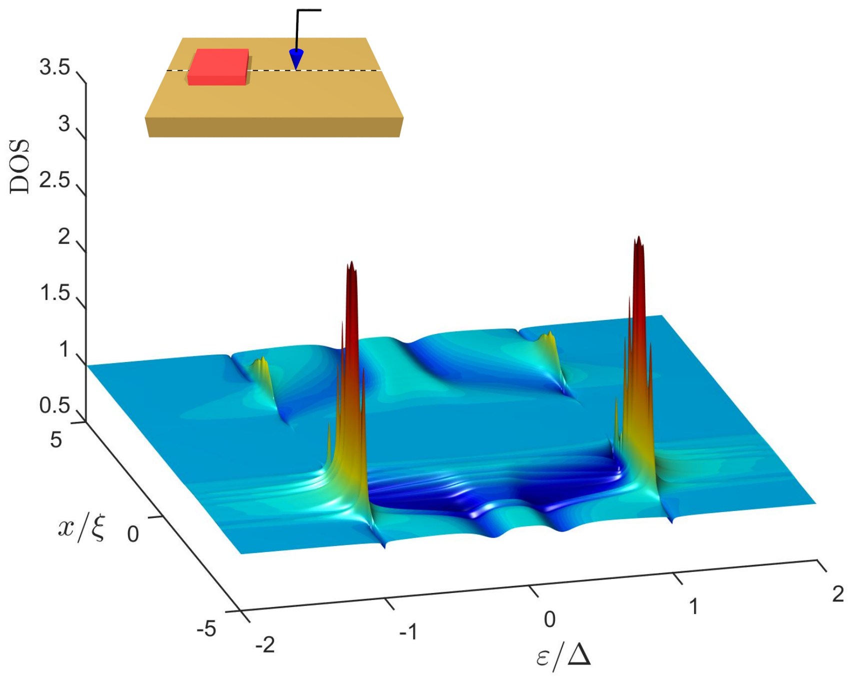
(c)



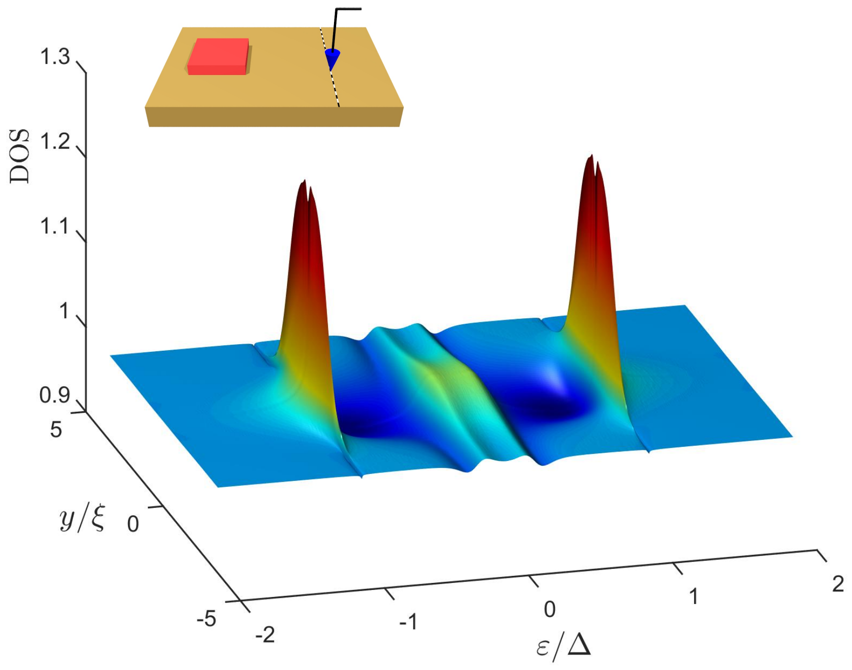
(d)

Figure 6.23: Charge current density in the ferromagnet for the two configurations A and B. The phase difference between the superconductors are $\phi = \frac{\pi}{2}$. All currents are scaled by $J_0 = \frac{N_0 e D \Delta}{8}$, and all lengths by ξ.

The density of states is given for configuration B in figure 6.24 for two different sections on the top surface of the ferromagnet, and is intended to simulate the measurement of $\frac{dI}{dV} \propto DOS$ by scanning tunneling microscopy. In figure 6.24a, it is seen that for the distance which passes directly underneath the superconductor there is found peaks at $\varepsilon = \pm\Delta$, indicating superconducting correlations, as is to be expected. In addition, a slight suppression is found at $\varepsilon = \pm h(\boldsymbol{r})$, i.e., a spin split minigap. Within the region of highest exchange field, it is seen that superconductivity is completely suppressed, and explains why the supercurrent is observed to circumvent this region. In the region above the second superconductor, the signs of the proximity effect reemerges, albeit somewhat damped, as the distance from the superconductor is now greater. Figure 6.24b shows the density of states for a line in the y direction, opposite the lower superconductor. The same modulation appears also here, with the greatest signs of superconducting correlations found in the region directly above the superconductor. Away from the superconductor, the density of states approaches that of a normal metal.



(a)



(b)

Figure 6.24: The density of states for configuration B. The dotted lines symbolize the spatial positions evaluated.

Chapter 7

Conclusion

It has been demonstrated that the finite element method is well suited for solving the quasiclassical Usadel equation. This was illustrated through comparison with known, benchmark solutions. The numerical method was then used to explore new effects that are intrinsically higher dimensional. For a 2D Josephson junction in the presence of a magnetic field, it was found that both the position and number of the vortices generated are influenced by the width of the junction and the phase difference between the superconductors. The vortex positions were seen to be determined by an equilibrium where the distance between the vortices are maximized while maintaining the symmetry of the system.

Another 2D Josephson junction was investigated, which contained ferromagnets with in-plane spin-orbit coupling. For pure Rashba or pure Dresselhaus SOC, the induced magnetization took on a predictable symmetry pattern. In addition, spincurrents were generated even without phase difference between the superconductors. It was observed that these currents occur due to a spin torque effect between different regions of the junctions. Finally, a 3D model was considered, which consisted of superconducting nanoislands placed on a ferromagnet. Charge currents between the islands were explored in the presence of both a homogeneous and inhomogeneous exchange fields. In particular, for the inhomogeneous exchange field it was revealed that the supercurrents avoid the regions of highest magnetization, creating a fully three dimensional current flow.

The numerical method developed opens exciting avenues in terms of future work. As the numerical case studies presented here show, even quite simple models yield interesting results. Therefore, much remains still to explore, such as different geometrical models and spatial distribution of exchange fields. The numerical code is also quite easily extended to nonequilibrium systems, which would allow passing charge currents through the system by means of a voltage difference.

The finite element method used herein to solve the Usadel equation was inspired by the solution techniques used in structural mechanics for the analysis of static problems. However, the method is also popular within the field of structural dynamics, which describes the vibrations of structures, for instance in earthquake design[95]. It is believed that a similar approach may be used to extend the numerical code developed for the solution of the nonlinear quasiclassical equations in 3D to handle time dependent problems as well. Research into time dependent systems by means of quasiclassical theory is at an early stage, although some advances have been made[96,97]. The successful development of a numerical routine capable of solving higher dimensional systems with time stepping would be a great leap forward.

Bibliography

1. Amundsen, M. and Linder, J. *Scientific Reports* **6**, 22765 (2016).
2. Meissner, W. and Ochsenfeld, R. *Naturwissenschaften* **21**(44), 787–788.
3. Usadel, K. D. *Physical Review Letters* **25**(8), 507–509 (1970). PRL.
4. Linder, J. and Robinson, J. W. A. *Nat Phys* **11**(4), 307–315 (2015).
5. Bergeret, F. S., Volkov, A. F., and Efetov, K. B. *Physical Review Letters* **86**(18), 4096–4099 (2001). PRL.
6. Houzet, M. and Buzdin, A. I. *Physical Review B* **76**(6), 060504 (2007). PRB.
7. Skyrme, T. H. R. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* **260**(1300), 127–138 (1961).
8. Nagaosa, N. and Tokura, Y. *Nat Nano* **8**(12), 899–911 (2013).
9. Bergeret, F. S. and Tokatly, I. V. *Physical Review Letters* **110**(11), 117003 (2013). PRL.
10. Bergeret, F. S. and Tokatly, I. V. *Physical Review B* **89**(13), 134517 (2014). PRB.
11. Yokoyama, T. and Linder, J. *Physical Review B* **92**(6), 060503 (2015). PRB.
12. Bergeret, F. S. and Cuevas, J. C. *Journal of Low Temperature Physics* **154**(1-2), 76–76 (2009).
13. Kim, J., Chua, V., Fiete, G. A., Nam, H., MacDonald, A. H., and Shih, C.-K. *Nat Phys* **8**(6), 464–469 (2012).
14. Roditchev, D., Brun, C., Serrier-Garcia, L., Cuevas, J. C., Bessa, V. H. L., Milosevic, M. V., Debontridder, F., Stolyarov, V., and Cren, T. *Nat Phys* **11**(4), 332–337 (2015).
15. Córdoba, R., Baturina, T. I., Sesé, J., Yu Mironov, A., De Teresa, J. M., Ibarra, M. R., Nasimov, D. A., Gutakovskii, A. K., Latyshev, A. V., Guillamón, I., Suderow, H., Vieira, S., Baklanov, M. R., Palacios, J. J., and Vinokur, V. M. *Nat Commun* **4**, 1437 (2013). 10.1038/ncomms2437.
16. Hirsch, J. E. *Physical Review Letters* **83**(9), 1834–1837 (1999). PRL.
17. Saburo, T. and Sadamichi, M. *Japanese Journal of Applied Physics* **51**(1R), 010110 (2012).

18. Wakamura, T., Akaike, H., Omori, Y., Niimi, Y., Takahashi, S., Fujimaki, A., Maekawa, S., and Otani, Y. *Nat Mater* **14**(7), 675–678 (2015).

19. Turner, M. J., Clough, R. W., Martin, H. C., and Topp, L. J. *Journal of the Aeronautical Sciences* **23**(9), 805–823 (1956).

20. Kammerlingh Onnes, H., (1911).

21. Tinkham, M. *Introduction to Superconductivity.* Dover, 2 edition, (1996).

22. Fossheim, K. and Sudbø, A. *Superconductivity. Physics and applications.* John Wiley & sons, Ltd, (2005).

23. Cooper, L. N. *Physical Review* **104**(4), 1189–1190 (1956). PR.

24. Pippard, A. B. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* **216**(1127), 547–568 (1953).

25. Abrikosov, A. A. *Sov. Phys. J. Exp. Theor. Phys.* **5**(6), 9 (1957).

26. Ginzburg, V. L. and Landau, L. D. *Zh. Eksp. Teor. Fiz.* **20** (1950).

27. Andreev, A. F. *Sov. Phys. J. Exp. Theor. Phys.* **19**(5) (1964).

28. Klapwijk, T. M. *Journal of Superconductivity* **17**(5), 593–611.

29. Eschrig, M. *Physics Today* **64**(1), 43–49 (2011).

30. Fulde, P. and Ferrell, R. A. *Physical Review* **135**(3A), A550–A563 (1964). PR.

31. Larkin, A. I.; Ovchinnikov, Y. N. *Sov. Phys. J. Exp. Theor. Phys.* **20** (1965).

32. Schrieffer, J. R., Balatsky, A. V., Abrahams, E., and Scalapino, D. J. *Journal of Superconductivity* **7**(3), 501–504.

33. Di Bernardo, A., Diesch, S., Gu, Y., Linder, J., Divitini, G., Ducati, C., Scheer, E., Blamire, M. G., and Robinson, J. W. A. *Nat Commun* **6** (2015). Supplementary information available for this article at http://www.nature.com/ncomms/2015/150902/ncomms9053/suppinfo/ncomms9053˙S1.html.

34. Kalcheim, Y., Millo, O., Di Bernardo, A., Pal, A., and Robinson, J. W. A. *Physical Review B* **92**(6), 060501 (2015). PRB.

35. Josephson, B. D. *Physics Letters* **1**(7), 251–253 (1962).

36. Keldysh, L. V. *Sov. Phys. J. Exp. Theor. Phys.* **20**(4), 1018– (1965).

37. Mattuck, R. D. *A Guide to Feynman Diagrams in the Many-Body Problem.* Dover Publications, 2 edition, (1992).

38. Nambu, Y. *Physical Review* **117**(3), 648–663 (1960). PR.

39. Kamenev, A. *Field Theory of Non-Equilibrium Systems.* Cambridge Univeristy Press, 1 edition, (2011).

40. Mahan, G. D. *Many-particle physics.* Third edition, (2000).

41. Bruus, H. and Flensberg, K. *Many-Body Quantum Theory in Condensed Matter Physics.* Oxford University Press, (2004).

42. Bardeen, J., Cooper, L. N., and Schrieffer, J. R. *Physical Review* **108**(5), 1175–1204 (1957). PR.

43. Buzdin, A. I. *Reviews of Modern Physics* **77**(3), 935–976 (2005). RMP.

44. Bransden, B. H. and Joachain, C. J. *Quantum Mechanics.* Pearson Education Limited, 2 edition, (2000).

45. Samokhin, K. V. *Physical Review B* **76**(9), 094516 (2007). PRB.

46. Bychkov, Y. A. and Rashba, E. I. *JETP Lett.* **39**(2) (1984).

47. Dresselhaus, G. *Physical Review* **100**(2), 580–586 (1955). PR.

48. Abrikosov, A. A. and Gor'kov, L. P. *Sov. Phys. J. Exp. Theor. Phys.* **35**(8), 9 (1959).

49. Abrikosov, A. A. and Gor'kov, L. P. *Sov. Phys. J. Exp. Theor. Phys.* **36**, 2 (1959).

50. Kopnin, N. B. *Theory of Nonequilibrium Superconductivity*, volume 110 of *International Series of Monographs on Physics.* Oxford University Press, (2001).

51. Meservey, R., Tedrow, P. M., and Fulde, P. *Physical Review Letters* **25**(18), 1270–1272 (1970). PRL.

52. Machida, K. and Ohmi, T. *Physical Review Letters* **86**(5), 850–853 (2001). PRL.

53. Morten, J. P. *Spin and Charge Transport in Dirty Superconductors.* Thesis, (2003).

54. Belzig, W., Wilhelm, F. K., Bruder, C., Schön, G., and Zaikin, A. D. *Superlattices and Microstructures* **25**(5–6), 1251–1288 (1999).

55. Larkin, A. I. and Ovchinnikov, Y. N. *Sov. Phys. J. Exp. Theor. Phys.* **68** (1975).

56. Eliashberg, G. M. *Sov. Phys. J. Exp. Theor. Phys.* **61** (1972).

57. Serene, J. W. and Rainer, D. *Physics Reports* **101**(4), 221–311 (1983).

58. Eilenberger, G. *Zeitschrift für Physik* **214**(2), 195–213.

59. Shelankov, A. L. *Journal of Low Temperature Physics* **60**(1), 29–44.

60. Cottet, A., Huertas-Hernando, D., Belzig, W., and Nazarov, Y. V. *Physical Review B* **80**(18), 184511 (2009). PRB.

61. Blanter, Y. M. and Büttiker, M. *Physics Reports* **336**(1–2), 1–166 (2000).

62. Stone, A. D., Mello, K. A., Muttalib, K. A., and Pichard, J. *Mesoscopic Phenomena in Solids*, volume 30 of *Modern Problems in Condensed Matter Sciences.* Elsevier Science Publishers, (1991).

63. Nazarov, Y. V. *Superlattices and Microstructures* **25**(5–6), 1221–1231 (1999).

64. Eschrig, M., Cottet, A., Belzig, W., and Linder, J. *New Journal of Physics* **17**(8), 083037 (2015).

65. Beenakker, C. W. J. *Reviews of Modern Physics* **69**(3), 731–808 (1997). RMP.

66. Kupriyanov, M. Y. and Lukichev, V. F. *Sov. Phys. J. Exp. Theor. Phys.* **67**(6), 6 (1988).

67. Chandrasekhar, V. *Proximity Coupled Systems: Quasiclassical Theory of Superconductivity*, volume 2 of *The Physics of Superconductors*. Springer-Verlag, (2004).

68. Schopohl, N. and Maki, K. *Physical Review B* **52**(1), 490–493 (1995). PRB.

69. Jacobsen, S. H., Ouassou, J. A., and Linder, J. *Physical Review B* **92**(2), 024510 (2015). PRB.

70. Dynes, R. C., Garno, J. P., Hertel, G. B., and Orlando, T. P. *Physical Review Letters* **53**(25), 2437–2440 (1984). PRL.

71. Eymard, R., Gallouët, T., and Herbin, R. *Finite volume methods*, volume Volume 7, 713–1018. Elsevier (2000).

72. Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes*. Cambridge University Press, third edition, (2007).

73. Quarteroni, A. *Numerical Models for Differential Problems*, volume 8 of *MS&A - Modeling, Simulation & Applications*. Springer Verlag, (2014).

74. Cook, R. D., Malkus, D. S., Plesha, M. E., and Witt, R. J. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons Inc., 4 edition, (2002).

75. Stone, M. H. *Mathematics Magazine* **21**(4), 167–184 (1948).

76. Abramowitz, M. and Stegun, I. A. *Handbook of Mathematical Functions*. Martino Publishing, (2014).

77. Kirk, B., Peterson, J., Stogner, R., and Carey, G. *Engineering with Computers* **22**(3-4), 237–254 (2006).

78. Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Rupp, K., Smith, B. F., Zampini, S., and Zhang, H. (2015).

79. Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Rupp, K., Smith, B. F., Zampini, S., and Zhang, H. Report ANL-95/11 - Revision 3.6, Argonne National Laboratory, (2015).

80. Smith, S. B., Gropp, W. D., McInnes, L. C., and F., B. *Efficient Management of Parallelism in Object Oriented Numerical Software Libraries*. Modern Software Tools in Scientific Computing. Birkhäuser Press, (1997).

81. Cuevas, J. C. and Bergeret, F. S. *Physical Review Letters* **99**(21), 217002 (2007). PRL.

82. Buzdin, A. *Physical Review B* **62**(17), 11377–11379 (2000). PRB.

83. Zareyan, M., Belzig, W., and Nazarov, Y. V. *Physical Review B* **65**(18), 184505 (2002). PRB.

84. Kontos, T., Aprili, M., Lesueur, J., and Grison, X. *Physical Review Letters* **86**(2), 304–307 (2001). PRL.

85. Yokoyama, T., Tanaka, Y., and Golubov, A. A. *Physical Review B* **75**(13), 134510 (2007). PRB.

86. Linder, J., Yokoyama, T., and Sudbø, A. *Physical Review B* **77**(17), 174514 (2008). PRB.

87. SanGiorgio, P., Reymond, S., Beasley, M. R., Kwon, J. H., and Char, K. *Physical Review Letters* **100**(23), 237002 (2008). PRL.

88. Linder, J. and Robinson, J. W. A. *Scientific Reports* **5**, 15483 (2015).

89. Balian, R. and Werthamer, N. R. *Physical Review* **131**(4), 1553–1564 (1963). PR.

90. Mackenzie, A. P. and Maeno, Y. *Reviews of Modern Physics* **75**(2), 657–712 (2003). RMP.

91. Slonczewski, J. C. *Journal of Magnetism and Magnetic Materials* **126**(1), 374–379 (1993).

92. Slonczewski, J. C. *Journal of Magnetism and Magnetic Materials* **159**(1–2), L1–L7 (1996).

93. Jacobsen, S. H., Kulagina, I., and Linder, J. *Scientific Reports* **6**, 23926 (2016).

94. Hammer, J. C., Cuevas, J. C., Bergeret, F. S., and Belzig, W. *Physical Review B* **76**(6), 064514 (2007). PRB.

95. Amundsen, M. *Dynamic Analysis of Offshore Concrete Structures subjected to Earthquake.* Thesis, (2012).

96. Cuevas, J. C., Hammer, J., Kopu, J., Viljas, J. K., and Eschrig, M. *Physical Review B* **73**(18), 184505 (2006). PRB.

97. Houzet, M. *Physical Review Letters* **101**(5), 057009 (2008). PRL.

Appendix A

Useful identities

In this appendix, identities which are used in the main text are derived.

A.1 Trace identities of the Pauli matrices

The Pauli matrices are defined as

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{A.1}$$

They satisfy the following (anti)commutation relation

$$[\sigma_k, \sigma_l] = 2i\varepsilon_{klm}\sigma_m \tag{A.2}$$
$$\{\sigma_k, \sigma_l\} = 2\delta_{kl} I \tag{A.3}$$

where I is the 2×2 unit matrix and ε_{klm} is the Levi-Civita tensor. From equation (A.1) it is seen immediately that

$$\mathrm{Tr}\,\{\sigma_k\} = 0 \tag{A.4}$$

for $k \in \{x, y, z\}$. From equations (A.2) and (A.3) it is found that

$$\mathrm{Tr}\,\{\sigma_k\sigma_l\} = \mathrm{Tr}\,\{\delta_{kl} I + i\varepsilon_{klm}\sigma_m\} = 2\delta_{kl} \tag{A.5}$$

The trace of three Pauli matrices becomes

$$\mathrm{Tr}\left\{\sigma_k\sigma_l\sigma_m\right\} = \mathrm{Tr}\left\{\delta_{kl}\sigma_m + i\varepsilon_{kln}\sigma_n\sigma_m\right\} = 2i\varepsilon_{klm} \tag{A.6}$$

Finally, the trace of four matrices may be computed as

$$\begin{aligned}\mathrm{Tr}\left\{\sigma_k\sigma_l\sigma_m\sigma_n\right\} &= \mathrm{Tr}\left\{\delta_{kl}\sigma_m\sigma_n + i\varepsilon_{klp}\sigma_p\sigma_m\sigma_n\right\} = 2\delta_{kl}\delta_{mn} - 2\varepsilon_{klp}\varepsilon_{pmn} \\ &= 2\delta_{kl}\delta_{mn} - 2\delta_{km}\delta_{ln} + 2\delta_{kn}\delta_{lm}\end{aligned} \tag{A.7}$$

where the identity $\varepsilon_{ijk}\varepsilon_{ilm} = \delta_{jl}\delta_{km} - \delta_{jm}\delta_{kl}$ has been used.

A.2 Commutator between $\theta(x)$ and p_x

The Heaviside step function is defined as

$$\theta(x) = \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases} \tag{A.8}$$

The commutator between $\theta(x)$ and the linear momentum operator in the x-direction, p_x, is sought after. Towards that end, it is useful to introduce an integral representation for the step function

$$\theta(x) = \lim_{\varepsilon\to 0}\frac{1}{2\pi i}\int_{-\infty}^{\infty}\frac{1}{1-i\varepsilon}e^{ixt}dt \tag{A.9}$$

Equivalence between equations (A.8) and (A.9) is readily verified by contour integration. The commutator becomes

$$\left[\theta(x), p_x\right] = \lim_{\varepsilon\to 0}\frac{1}{2\pi i}\int_{-\infty}^{\infty}\frac{1}{1-i\varepsilon}\left[e^{ixt}, p_x\right]dt \tag{A.10}$$

The commutator in the integrand is given as

$$\left[e^{ixt}, p_x\right] = \sum_{n=0}^{\infty}\frac{(it)^n}{n!}\left[x^n, p_x\right] \tag{A.11}$$

Therefore, $[x^n, p_x]$ is needed. It may be found by induction. It is well known that $[x, p_x] = i\hbar$. This means that

$$\left[x^2, p_x\right] = x^2 p_x - p_x x^2 = x(xp_x - p_x x) + i\hbar x = 2i\hbar x \tag{A.12}$$

From this, it is tempting to guess at the formula

$$[x^n, p_x] = ni\hbar x^{n-1} \tag{A.13}$$

It is assumed that this holds. Next it is demonstrated that this is consistent for x^{n+1}:

$$\left[x^{n+1}, p_x\right] = x^{n+1} p_x - p_x x^{n+1} = x(x^n p_x - p_x x) + i\hbar x^n = (n+1)i\hbar x^n \tag{A.14}$$

To summarize, equation (A.13) holds for $n = 1$. Furthermore, if it holds for n, it also holds for $n + 1$. Combining these statements concludes the proof of equation (A.13). Inserting this into equation (A.11) gives

$$\left[e^{ixt}, p_x\right] = \sum_{n=0}^{\infty} \frac{(it)^n}{n!} ni\hbar x^{n-1} = -\hbar t \sum_{n=1}^{\infty} \frac{(it)^{n-1}}{(n-1)!} x^{n-1} = -\hbar t e^{ixt} \tag{A.15}$$

Inserting this into equation (A.10) gives

$$\begin{aligned}[\theta(x), p_x] &= \lim_{\varepsilon\to 0} -\frac{\hbar}{2\pi i}\int_{-\infty}^{\infty} \frac{t}{1-i\varepsilon} e^{ixt} dt = \lim_{\varepsilon\to 0} \frac{i\hbar}{2\pi i}\int_{-\infty}^{\infty} \frac{1}{1-i\varepsilon}\frac{\partial}{\partial x} e^{ixt} dt \\ &= i\hbar \frac{\partial}{\partial x}\theta(x) = i\hbar\delta(x) \end{aligned} \tag{A.16}$$

A.3 Fourier transform of products

The Fourier transform of a product of two functions $f_1\left(\boldsymbol{R} + \frac{1}{2}\boldsymbol{r}\right)$ and $f_2\left(\boldsymbol{R}, \boldsymbol{r}\right)$ is a convolution, which is difficult to handle analytically. However, in the special case considered here, a fortuitous combination of the dependent variables makes a formal solution possible. With $\boldsymbol{R}$ considered as a constant, the Fourier transform of f_2 becomes

$$f_2(\boldsymbol{R}, \boldsymbol{p}) = \int d\boldsymbol{r} e^{-i\boldsymbol{p}\cdot\boldsymbol{r}/\hbar} f_2(\boldsymbol{R}, \boldsymbol{r}) \tag{A.17}$$

Furthermore, a Taylor expansionn of f_1 gives

$$f_1\left(\boldsymbol{R} + \frac{1}{2}\boldsymbol{r}\right) \approx \left(1 + \frac{1}{2}\boldsymbol{r}\cdot\nabla_R\right) f_1(\boldsymbol{R}) \tag{A.18}$$

Therefore, the Fourier transform of the product, denoted as $f_1 \otimes f_2(\boldsymbol{p})$, becomes

$$\begin{aligned} f_1 \otimes f_2(\boldsymbol{p}) &= \int d\boldsymbol{r} e^{-i\boldsymbol{p}\cdot\boldsymbol{r}/\hbar} f_1\left(\boldsymbol{R} + \frac{1}{2}\boldsymbol{r}\right) f_2(\boldsymbol{R}, \boldsymbol{r}) \\ &\approx \int d\boldsymbol{r} e^{-i\boldsymbol{p}\cdot\boldsymbol{r}/\hbar} \left(1 + \frac{1}{2}\boldsymbol{r} \cdot \nabla_{R_1}\right) f_1(\boldsymbol{R}) f_2(\boldsymbol{R}, \boldsymbol{r}) \\ &= \left(1 + \frac{i\hbar}{2}\nabla_p \cdot \nabla_{R_1}\right) f_1(\boldsymbol{R}) \int d\boldsymbol{r} e^{-i\boldsymbol{p}\cdot\boldsymbol{r}/\hbar} f_2(\boldsymbol{R}, \boldsymbol{r}) \\ &= \left(1 + \frac{i\hbar}{2}\nabla_p \cdot \nabla_{R_1}\right) f_1(\boldsymbol{R}) f_2(\boldsymbol{R}, \boldsymbol{p}) \end{aligned} \tag{A.19}$$

Equivalently, for $f_1\left(\boldsymbol{R} - \frac{1}{2}\boldsymbol{r}\right)$ it is found that

$$f_2 \otimes f_1(\boldsymbol{p}) = \left(1 - \frac{i\hbar}{2}\nabla_p \cdot \nabla_{R_1}\right) f_1(\boldsymbol{R}) f_2(\boldsymbol{R}, \boldsymbol{p}) \tag{A.20}$$

Appendix B

Numerical code

B.1 nonlinearsolver.cpp

```
#include <iostream>
#include <algorithm>
#include <sstream>
#include <math.h>
#include <armadillo>
#include <complex>

#include "libmesh/libmesh.h"
#include "libmesh/mesh.h"
#include "libmesh/mesh_generation.h"
#include "libmesh/exodusII_io.h"
#include "libmesh/equation_systems.h"
#include "libmesh/fe.h"
#include "libmesh/quadrature_gauss.h"
#include "libmesh/dof_map.h"
#include "libmesh/sparse_matrix.h"
#include "libmesh/numeric_vector.h"
#include "libmesh/dense_matrix.h"
#include "libmesh/dense_vector.h"
#include "libmesh/linear_implicit_system.h"
#include "libmesh/nonlinear_implicit_system.h"
#include "libmesh/nonlinear_solver.h"
#include "libmesh/perf_log.h"
#include "libmesh/boundary_info.h"
#include "libmesh/utility.h"
#include "libmesh/dense_submatrix.h"
#include "libmesh/dense_subvector.h"
#include "libmesh/elem.h"
#include "libmesh/vtk_io.h"
#include "libmesh/gmsh_io.h"
#include "libmesh/exodusII_io.h"
#include "libmesh/matlab_io.h"
#include "libmesh/parallel_mesh.h"
#include "compute_RJ.h"
#include "nonlinearsolver.h"
#include "MVConverter.h"
#include "BCS.h"
#include "spincurrent.h"
#include "customstructs.h"
#include "magnetization.h"
#include "externalflux.h"
#include "wrap.h"
#include <utility>
#include "Connection.h"
#include "compute_output.h"
```

```
using namespace std;
using namespace arma;
using namespace libMesh;

EquationSystems *_eqsys;
Connection* _connectivity;

void nonlinearsolver(int argc, char** argv, vector<double> Evec, vector<Input> parameters
    ,
                    Meshinput meshinfo, BCinputV BC, int iteration, string prevsol){


    LibMeshInit init (argc, argv);
    Mesh mesh(init.comm());

    double Lz = parameters[0].dimensions[2];
    double inelsc = meshinfo.inelsc;

    if (meshinfo.type == "Default"){
        if (Lz == 0){

            MeshTools::Generation::build_square(mesh, meshinfo.number[0], meshinfo.number
                [1], 0.5, 0.5, 0.5, 0.5, QUAD9);

        } else {

            MeshTools::Generation::build_cube(mesh, meshinfo.number[0], meshinfo.number
                [1], meshinfo.number[2], 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, HEX20);

        }


    } else {


        GmshIO(mesh).read(meshinfo.type.append(".msh"));
        mesh.prepare_for_use();

    }


    Connection connectivity(mesh,BC);
    _connectivity = &connectivity;
    ExodusII_IO(mesh).write("mesh.e");

    EquationSystems eqsys(mesh);
    _eqsys = &eqsys;

    NonlinearImplicitSystem& system = eqsys.add_system<NonlinearImplicitSystem>("Usadel")
        ;
    LinearImplicitSystem& gradsys = eqsys.add_system<LinearImplicitSystem>("Gradients");
    gradsys.attach_assemble_function(compute_gradients);

    if (prevsol == ""){

        ExplicitSystem& PPsys = eqsys.add_system<ExplicitSystem>("PP");

        system.add_variable("g00", SECOND);
        system.add_variable("g01", SECOND);
        system.add_variable("g10", SECOND);
        system.add_variable("g11", SECOND);
        system.add_variable("gt00", SECOND);
        system.add_variable("gt01", SECOND);
        system.add_variable("gt10", SECOND);
        system.add_variable("gt11", SECOND);


        PPsys.add_variable("DOS", SECOND);
        PPsys.add_variable("Jx", SECOND);
        PPsys.add_variable("Jy", SECOND);
        PPsys.add_variable("Jz", SECOND);
        PPsys.add_variable("PC", SECOND);
        PPsys.add_variable("JSXx", SECOND);
    PPsys.add_variable("JSXy", SECOND);
    PPsys.add_variable("JSXz", SECOND);
    PPsys.add_variable("JSYx", SECOND);
    PPsys.add_variable("JSYy", SECOND);
    PPsys.add_variable("JSYz", SECOND);
    PPsys.add_variable("JSZx", SECOND);
    PPsys.add_variable("JSZy", SECOND);
    PPsys.add_variable("JSZz", SECOND);
    PPsys.add_variable("Hx", SECOND);
    PPsys.add_variable("Hy", SECOND);
    PPsys.add_variable("Hz", SECOND);
    PPsys.add_variable("Mx", SECOND);
    PPsys.add_variable("My", SECOND);
    PPsys.add_variable("Mz", SECOND);


    gradsys.add_variable("dgx00",SECOND);
    gradsys.add_variable("dgx01",SECOND);
    gradsys.add_variable("dgx10",SECOND);
    gradsys.add_variable("dgx11",SECOND);
    gradsys.add_variable("dgtx00",SECOND);
    gradsys.add_variable("dgtx01",SECOND);
    gradsys.add_variable("dgtx10",SECOND);
    gradsys.add_variable("dgtx11",SECOND);

    gradsys.add_variable("dgy00",SECOND);
    gradsys.add_variable("dgy01",SECOND);
    gradsys.add_variable("dgy10",SECOND);
    gradsys.add_variable("dgy11",SECOND);
    gradsys.add_variable("dgty00",SECOND);
    gradsys.add_variable("dgty01",SECOND);
    gradsys.add_variable("dgty10",SECOND);
    gradsys.add_variable("dgty11",SECOND);

    gradsys.add_variable("dgz00",SECOND);
    gradsys.add_variable("dgz01",SECOND);
    gradsys.add_variable("dgz10",SECOND);
    gradsys.add_variable("dgz11",SECOND);
    gradsys.add_variable("dgtz00",SECOND);
    gradsys.add_variable("dgtz01",SECOND);
    gradsys.add_variable("dgtz10",SECOND);
    gradsys.add_variable("dgtz11",SECOND);



} else {

    string folder = "Restart/";
    cout << ">>>>>_Reading_" << prevsol << "_<<<<<" << endl;
    eqsys.read(folder.append(prevsol));
    system.update();
    gradsys.update();

}

system.nonlinear_solver  > matvec = compute_RJ;

eqsys.parameters.set<unsigned int>("nonlinear_solver_maximum_iterations") = meshinfo.
    maxiter;
eqsys.parameters.set<Real>("linear_solver_tolerance") = 1.0e 4;


eqsys.parameters.set<vector<Real>>("BCstrength") = BC.strength;
eqsys.parameters.set<vector<Real>>("BCpenalty") = BC.penalty;
eqsys.parameters.set<vector<string>>("BCtype") = BC.type;
eqsys.parameters.set<vector<int>>("BCID") = BC.ID;
eqsys.parameters.set<vector<int>>("BCneighborID") = BC.neighborID;
eqsys.parameters.set<vector<double>>("BCphase") = BC.phase;

vector<Input> zpar = parameters;

if (Lz == 0)
    for (unsigned int i = 0; i < zpar.size(); i++)
```

```
        zpar[i].dimensions[2] = 1.0; // Avoid division by zero


    vector<double> parvec = wrap(zpar);
    eqsys.parameters.set<vector<double>>("parvec") = parvec;


    if (prevsol == ""){

        eqsys.init();

    }

    ofstream summary;
    summary.open("Summary.txt", ios::app);
    const set<string> outputname = { "PP" };
    const set<string> *poutput;

    poutput = &outputname;

    double Eold = Evec[1];

    for (double E : Evec){

        complex<double> Ec(E,inelsc);
        eqsys.parameters.set<Number>("E") = Ec;
        eqsys.parameters.set<Real>("dE") = abs(Eold  E);


        eqsys.get_system("Usadel").solve();
        eqsys.get_system("Gradients").solve();



        cout << "Usadel␣system␣solved␣at␣nonlinear␣iteration␣"
            << system.n_nonlinear_iterations()
            << "␣,␣final␣nonlinear␣residual␣norm:␣"
            << system.final_nonlinear_residual()
            << endl;

        // Post Processing
        //
        //


        if (system.final_nonlinear_residual() > 1.0e 4){

            cx_vec y = zeros<cx_vec>(8);
            InitialGuess(eqsys, y);


        }

        if( system.final_nonlinear_residual() < 1.0e 4){

            summary << iteration << "␣␣␣" << E << "␣␣␣" << system.
                final_nonlinear_residual() << endl;

            postprocessor(eqsys);

            ostringstream filename;

            filename << "Results/solution"
                << iteration
                << ".e";

            ExodusII_IO(mesh).write_equation_systems(filename.str(), eqsys, poutput );


        } else {

            summary << iteration << "␣␣␣" << E << "␣␣␣" << system.
                final_nonlinear_residual() << "␣␣␣" << "DIVERGED" << endl;
        }

        iteration = iteration + 1;
        Eold = E;
    }

    summary.close();
    return;


}


void postprocessor(EquationSystems& es){


    const MeshBase& mesh = es.get_mesh();
    const unsigned int dim = mesh.mesh_dimension();

    complex<double> Ec = es.parameters.get<Number>(”E”);
    double dE = es.parameters.get<Real>(”dE”);
    double T = 0.001;
    double E = real(Ec);

    complex<double> ii(0,1);
    NonlinearImplicitSystem& system = es.get_system<NonlinearImplicitSystem>(”Usadel”);


    vector<double> parvec = es.parameters.get<vector<double>>(”parvec”);
    vector<Input> parameters = unwrapper(parvec);
    const int Nvars = 8;
    vector<unsigned int> vars;
    system.get_all_variable_numbers(vars);

    const DofMap& dof_map = system.get_dof_map();

    FEType fe_type = dof_map.variable_type(vars[0]);
    AutoPtr<FEBase> fe (FEBase::build(dim, fe_type));
    QGauss qrule (dim, fe_type.default_quadrature_order());
    fe  > attach_quadrature_rule(&qrule);

    const vector<Real>& JxW = fe  > get_JxW();
    const vector<vector<Real>>& phi = fe  > get_phi();

    LinearImplicitSystem& gradsys = es.get_system<LinearImplicitSystem>(”Gradients”);


    vector<unsigned int> dvars;
    gradsys.get_all_variable_numbers(dvars);
    sort(dvars.begin(),dvars.end());

    const DofMap& grad_dof_map = gradsys.get_dof_map();


    ExplicitSystem& PPsys = es.get_system<ExplicitSystem>(”PP”);
    vector<unsigned int> PPvars;
    PPsys.get_all_variable_numbers(PPvars);
    sort(PPvars.begin(),PPvars.end());

  //  const unsigned int DOSnum = PPsys.variable_number(”DOS”);
  //
    const unsigned int Jxnum = PPsys.variable_number(”Jx”);
    const unsigned int Jynum = PPsys.variable_number(”Jy”);
    const unsigned int Jznum = PPsys.variable_number(”Jz”);
    const unsigned int PCnum = PPsys.variable_number(”PC”);
    const unsigned int Mxnum = PPsys.variable_number(”Mx”);
    const unsigned int Mynum = PPsys.variable_number(”My”);
    const unsigned int Mznum = PPsys.variable_number(”Mz”);
    const unsigned int JSXxnum = PPsys.variable_number(”JSXx”);
    const unsigned int JSXynum = PPsys.variable_number(”JSXy”);
    const unsigned int JSXznum = PPsys.variable_number(”JSXz”);
const unsigned int JSYxnum = PPsys.variable_number("JSYx");
const unsigned int JSYynum = PPsys.variable_number("JSYy");
const unsigned int JSYznum = PPsys.variable_number("JSYz");

const unsigned int JSZxnum = PPsys.variable_number("JSZx");
const unsigned int JSZynum = PPsys.variable_number("JSZy");
const unsigned int JSZznum = PPsys.variable_number("JSZz");


const DofMap& PP_dof_map = PPsys.get_dof_map();
int n_PPvars = PPsys.n_vars();

vector<dof_id_type> dof_indices;
vector< vector<dof_id_type> > dof_indices_var(Nvars);
vector< vector<dof_id_type> > dof_indices_dvar(3*Nvars);
vector< vector<dof_id_type> > dof_indices_PPvar(n_PPvars);

vector<dof_id_type> dof_indices_Jx;
vector<dof_id_type> dof_indices_Jy;
vector<dof_id_type> dof_indices_Jz;
vector<dof_id_type> dof_indices_PC;

vector<dof_id_type> dof_indices_Mx;
vector<dof_id_type> dof_indices_My;
vector<dof_id_type> dof_indices_Mz;

vector<dof_id_type> dof_indices_JSXx;
vector<dof_id_type> dof_indices_JSXy;
vector<dof_id_type> dof_indices_JSXz;

vector<dof_id_type> dof_indices_JSYx;
vector<dof_id_type> dof_indices_JSYy;
vector<dof_id_type> dof_indices_JSYz;

vector<dof_id_type> dof_indices_JSZx;
vector<dof_id_type> dof_indices_JSZy;
vector<dof_id_type> dof_indices_JSZz;


MeshBase::const_element_iterator  el = mesh.active_local_elements_begin();
const MeshBase::const_element_iterator  end_el = mesh.active_local_elements_end();


cx_vec y(Nvars), dy_x(Nvars), dy_y(Nvars), dy_z(Nvars);

for ( ; el != end_el; ++el){

    const Elem* elem = *el;
    int sid = elem  > subdomain_id();

    vector<double> flux = parameters[sid].ext_flux;
    vector<double> h    = parameters[sid].magnetization;

    for (int var = 0; var < Nvars; var++)
        dof_map.dof_indices(elem, dof_indices_var[var], vars[var]);

    for (int var = 0; var < 3*Nvars; var++)
        grad_dof_map.dof_indices(elem,dof_indices_dvar[var], dvars[var]);

    for (int var = 0; var < n_PPvars; var++)
        PP_dof_map.dof_indices(elem,dof_indices_PPvar[var], PPvars[var]);

    PP_dof_map.dof_indices(elem, dof_indices_Jx, Jxnum);
    PP_dof_map.dof_indices(elem, dof_indices_Jy, Jynum);
    PP_dof_map.dof_indices(elem, dof_indices_Jz, Jznum);
    PP_dof_map.dof_indices(elem, dof_indices_PC, PCnum);

    PP_dof_map.dof_indices(elem, dof_indices_Mx, Mxnum);
    PP_dof_map.dof_indices(elem, dof_indices_My, Mynum);
    PP_dof_map.dof_indices(elem, dof_indices_Mz, Mznum);

    PP_dof_map.dof_indices(elem, dof_indices_JSXx, JSXxnum);
    PP_dof_map.dof_indices(elem, dof_indices_JSXy, JSXynum);
    PP_dof_map.dof_indices(elem, dof_indices_JSXz, JSXznum);

    PP_dof_map.dof_indices(elem, dof_indices_JSYx, JSYxnum);
    PP_dof_map.dof_indices(elem, dof_indices_JSYy, JSYynum);
    PP_dof_map.dof_indices(elem, dof_indices_JSYz, JSYznum);

    PP_dof_map.dof_indices(elem, dof_indices_JSZx, JSZxnum);
    PP_dof_map.dof_indices(elem, dof_indices_JSZy, JSZynum);
    PP_dof_map.dof_indices(elem, dof_indices_JSZz, JSZznum);

    fe > reinit(elem);
    vector<Number> DOS(phi.size()), Jx(phi.size()), Jy(phi.size()), Jz(phi.size());
    vector<Number> Jxold(phi.size()), Jyold(phi.size()), Jzold(phi.size());

    vector<Number> JSXx(phi.size()), JSXy(phi.size()), JSXz(phi.size());
    vector<Number> JSYx(phi.size()), JSYy(phi.size()), JSYz(phi.size());
    vector<Number> JSZx(phi.size()), JSZy(phi.size()), JSZz(phi.size());

    vector<Number> JSXxold(phi.size()), JSXyold(phi.size()), JSXzold(phi.size());
    vector<Number> JSYxold(phi.size()), JSYyold(phi.size()), JSYzold(phi.size());
    vector<Number> JSZxold(phi.size()), JSZyold(phi.size()), JSZzold(phi.size());

    vector<Number> PC(phi.size()), PCold(phi.size()), Hx(phi.size()),Hy(phi.size()),
        Hz(phi.size());
    vector<Number> Mx(phi.size()), My(phi.size()), Mz(phi.size());
    vector<Number> Mxold(phi.size()), Myold(phi.size()), Mzold(phi.size());
    vector<double> xpts(phi.size()), ypts(phi.size());
    vector<vector<Number>> PPout(n_PPvars, vector<Number>(phi.size()));


    for (unsigned int i = 0; i < phi.size(); i++){

        xpts[i] = (*el) > point(i)(0);
        ypts[i] = (*el) > point(i)(1);

        Jxold[i] = PPsys.current_solution(dof_indices_Jx[i]);
        Jyold[i] = PPsys.current_solution(dof_indices_Jy[i]);
        Jzold[i] = PPsys.current_solution(dof_indices_Jz[i]);
        PCold[i] = PPsys.current_solution(dof_indices_PC[i]);

        Mxold[i] = PPsys.current_solution(dof_indices_Mx[i]);
        Myold[i] = PPsys.current_solution(dof_indices_My[i]);
        Mzold[i] = PPsys.current_solution(dof_indices_Mz[i]);

        JSXxold[i] = PPsys.current_solution(dof_indices_JSXx[i]);
        JSXyold[i] = PPsys.current_solution(dof_indices_JSXy[i]);
        JSXzold[i] = PPsys.current_solution(dof_indices_JSXz[i]);

        JSYxold[i] = PPsys.current_solution(dof_indices_JSYx[i]);
        JSYyold[i] = PPsys.current_solution(dof_indices_JSYy[i]);
        JSYzold[i] = PPsys.current_solution(dof_indices_JSYz[i]);

        JSZxold[i] = PPsys.current_solution(dof_indices_JSZx[i]);
        JSZyold[i] = PPsys.current_solution(dof_indices_JSZy[i]);
        JSZzold[i] = PPsys.current_solution(dof_indices_JSZz[i]);

        for (int vars = 0; vars < Nvars; vars++){

            y(vars) = system.current_solution(dof_indices_var[vars][i]);
            dy_x(vars) = gradsys.current_solution(dof_indices_dvar[vars][i]);
            dy_y(vars) = gradsys.current_solution(dof_indices_dvar[Nvars + vars][i]);
            dy_z(vars) = gradsys.current_solution(dof_indices_dvar[2*Nvars + vars][i
                ]);

        }

        cx_mat G = v_to_m(y);
        cx_mat Gt = v_to_mt(y);
        cx_mat dGx = v_to_m(dy_x);
        cx_mat dGtx = v_to_mt(dy_x);
        cx_mat dGy = v_to_m(dy_y);
        cx_mat dGty = v_to_mt(dy_y);
        cx_mat dGz = v_to_m(dy_z);
```

```
            cx_mat dGtz = v_to_mt(dy_z);


            mat I(2,2); I.eye();

            cx_mat Ninv = I    G*Gt;
            cx_mat N = inv(Ninv);
            cx_mat Ntinv = I    Gt*G;
            cx_mat Nt = inv(Ntinv);

            cx_mat f = 2.*N*G;
            cx_mat ft = 2.*Nt*Gt;
            cx_mat df_x = 2.*N*(dGx*Gt + G*dGtx)*N*G + 2.*N*dGx;
            cx_mat df_y = 2.*N*(dGy*Gt + G*dGty)*N*G + 2.*N*dGy;
            cx_mat df_z = 2.*N*(dGz*Gt + G*dGtz)*N*G + 2.*N*dGz;
            cx_mat dft_x = 2.*Nt*(dGtx*G + Gt*dGx)*Nt*Gt + 2.*Nt*dGtx;
            cx_mat dft_y = 2.*Nt*(dGty*G + Gt*dGy)*Nt*Gt + 2.*Nt*dGty;
            cx_mat dft_z = 2.*Nt*(dGtz*G + Gt*dGz)*Nt*Gt + 2.*Nt*dGtz;


            cx_mat g = N*(I + G*Gt);
            cx_mat gt = Nt*(I + Gt*G);
            cx_mat dg_x = 2.*N*(dGx*Gt + G*dGtx)*N;
            cx_mat dg_y = 2.*N*(dGy*Gt + G*dGty)*N;
            cx_mat dg_z = 2.*N*(dGz*Gt + G*dGtz)*N;
            cx_mat dgt_x = 2.*Nt*(dGtx*G + Gt*dGx)*Nt;
            cx_mat dgt_y = 2.*Nt*(dGty*G + Gt*dGy)*Nt;
            cx_mat dgt_z = 2.*Nt*(dGtz*G + Gt*dGz)*Nt;

//              cx_mat Jx_mat = fRt*dfR_x    fR*dfRt_x + 4.*ii*sx*fR*fRt;
            cx_mat Jy_mat = ft*df_y    f*dft_y;
            cx_mat Jz_mat = ft*df_z    f*dft_z;
            cx_mat Ax, Ay, Az;
            SOC A = externalflux(flux, xpts[i], ypts[i]);
            Ax = A.x;
            Ay = A.y;
            Az = A.z;



            Gmat g_mat = compute_g_matrices(y, dy_x, dy_y, dy_z, E, T);
            Gmat g_neg = compute_g_negE(g_mat);
            //Compute Keldysh component for positive energies
            cx_mat gdg_x = g_mat.gtot*g_mat.dgtot_x    ii*g_mat.gtot*(A.x_large*g_mat.gtot
                   g_mat.gtot*A.x_large);
            cx_mat gdg_y = g_mat.gtot*g_mat.dgtot_y    ii*g_mat.gtot*(A.y_large*g_mat.gtot
                   g_mat.gtot*A.y_large);
            cx_mat gdg_z = g_mat.gtot*g_mat.dgtot_z    ii*g_mat.gtot*(A.z_large*g_mat.gtot
                   g_mat.gtot*A.z_large);


            cx_mat gdg_x_k = gdg_x(span(0,3),span(4,7));
            cx_mat gdg_y_k = gdg_y(span(0,3),span(4,7));
            cx_mat gdg_z_k = gdg_z(span(0,3),span(4,7));

            //Compute Keldysh component for negative energies
            cx_mat gdg_x_n = g_neg.gtot*g_neg.dgtot_x    ii*g_neg.gtot*(A.x_large*g_neg.
                gtot    g_neg.gtot*A.x_large);
            cx_mat gdg_y_n = g_neg.gtot*g_neg.dgtot_y    ii*g_neg.gtot*(A.y_large*g_neg.
                gtot    g_neg.gtot*A.y_large);
            cx_mat gdg_z_n = g_neg.gtot*g_neg.dgtot_z    ii*g_neg.gtot*(A.z_large*g_neg.
                gtot    g_neg.gtot*A.z_large);


            cx_mat gdg_x_k_n = gdg_x_n(span(0,3),span(4,7));
            cx_mat gdg_y_k_n = gdg_y_n(span(0,3),span(4,7));
            cx_mat gdg_z_k_n = gdg_z_n(span(0,3),span(4,7));
//              cx_mat gdg_k_y = keldysh_gdg(g_mat.gtot, g_mat.dgtot_y, Ay);
//              cx_mat gdg_k_z = keldysh_gdg(g_mat.gtot, g_mat.dgtot_z, Az);

            mat rho3 = zeros<mat>(4,4);
            mat rho1 = rho3;
```

```
                rho3.submat(0,0,1,1) = I;
                rho3.submat(2,2,3,3) =  I;

                rho1.submat(0,2,1,3) = I;
                rho1.submat(2,0,3,1) = I;


                DOS[i] = real(trace(g))/2;

//
                Jx[i] = dE*trace(rho3*(gdg_x_k + gdg_x_k_n)) + Jxold[i];
                Jy[i] = dE*trace(rho3*(gdg_y_k + gdg_y_k_n)) + Jyold[i];
                Jz[i] = dE*trace(rho3*(gdg_z_k + gdg_z_k_n)) + Jzold[i];


                mat sigma_x, sigma_z;
                cx_mat sigma_y;

                mat S_x = zeros<mat>(4,4);
                cx_mat S_y = zeros<cx_mat>(4,4);
                mat S_z = zeros<mat>(4,4);

                sigma_x << 0 << 1 << endr
                        << 1 << 0 << endr;

                sigma_y << 0 <<  ii << endr
                        << ii << 0  << endr;

                sigma_z << 1 << 0 << endr
                        << 0 <<  1 << endr;
                S_x.submat(0,0,1,1) = sigma_x;
                S_x.submat(2,2,3,3) = sigma_x;

                S_y.submat(0,0,1,1) = sigma_y;
                S_y.submat(2,2,3,3) = conj(sigma_y);

                S_z.submat(0,0,1,1) = sigma_z;
                S_z.submat(2,2,3,3) = sigma_z;


                Mx[i] = dE*trace(S_x*(g_mat.gK + g_neg.gK)) + Mxold[i];
                My[i] = dE*trace(S_y*(g_mat.gK + g_neg.gK)) + Myold[i];
                Mz[i] = dE*trace(S_z*(g_mat.gK + g_neg.gK)) + Mzold[i];




                PC[i] = dE*tanh(1.76*E/(2.*T))*(f(0,1)    conj(ft(0,1)) + conj(ft(1,0))    f
                    (1,0)) + PCold[i];
                double sx =  flux[0];




                JSXx[i] = dE*trace(rho3*S_x*(gdg_x_k + gdg_x_k_n)) + JSXxold[i];
                JSXy[i] = dE*trace(rho3*S_x*(gdg_y_k + gdg_y_k_n)) + JSXyold[i];
                JSXz[i] = dE*trace(rho3*S_x*(gdg_z_k + gdg_z_k_n)) + JSXzold[i];

                JSYx[i] = dE*trace(rho3*S_y*(gdg_x_k + gdg_x_k_n)) + JSYxold[i];
                JSYy[i] = dE*trace(rho3*S_y*(gdg_y_k + gdg_y_k_n)) + JSYyold[i];
                JSYz[i] = dE*trace(rho3*S_y*(gdg_z_k + gdg_z_k_n)) + JSYzold[i];

                JSZx[i] = dE*trace(rho3*S_z*(gdg_x_k + gdg_x_k_n)) + JSZxold[i];
                JSZy[i] = dE*trace(rho3*S_z*(gdg_y_k + gdg_y_k_n)) + JSZyold[i];
                JSZz[i] = dE*trace(rho3*S_z*(gdg_z_k + gdg_z_k_n)) + JSZzold[i];

                vector<double> hfield = magnetization(h,xpts[i],ypts[i]);
                Hx[i] = hfield[0]; Hy[i] = hfield[1]; Hz[i] = hfield[2];


                PPout[0][i] = DOS[i];
                PPout[1][i] = Jx[i];
                PPout[2][i] = Jy[i];
```

```
            PPout[3][i] = Jz[i];
            PPout[4][i] = PC[i];
            PPout[5][i] = JSXx[i];
            PPout[6][i] = JSXy[i];
            PPout[7][i] = JSXz[i];
            PPout[8][i] = JSYx[i];
            PPout[9][i] = JSYy[i];
            PPout[10][i] = JSYz[i];
            PPout[11][i] = JSZx[i];
            PPout[12][i] = JSZy[i];
            PPout[13][i] = JSZz[i];
            PPout[14][i] = Hx[i];
            PPout[15][i] = Hy[i];
            PPout[16][i] = Hz[i];
            PPout[17][i] = Mx[i];
            PPout[18][i] = My[i];
            PPout[19][i] = Mz[i];


            for (unsigned int var = 0; var < n_PPvars; var++)
                PPsys.solution > set(dof_indices_PPvar[var][i],PPout[var][i]);
        }


    }


    PPsys.solution > close();
    PPsys.update();





    return;


}


void InitialGuess(EquationSystems& es, cx_vec value){



    const MeshBase& mesh = es.get_mesh();
    const unsigned int dim = mesh.mesh_dimension();

    NonlinearImplicitSystem& system = es.get_system<NonlinearImplicitSystem>(”Usadel”);


    const int Nvars = 8;
    unsigned int vars[Nvars];
    vars[0] = system.variable_number(”g00”);
    vars[1] = system.variable_number(”g01”);
    vars[2] = system.variable_number(”g10”);
    vars[3] = system.variable_number(”g11”);
    vars[4] = system.variable_number(”gt00”);
    vars[5] = system.variable_number(”gt01”);
    vars[6] = system.variable_number(”gt10”);
    vars[7] = system.variable_number(”gt11”);

    const DofMap& dof_map = system.get_dof_map();

    FEType fe_type = dof_map.variable_type(vars[0]);
    AutoPtr<FEBase> fe (FEBase::build(dim, fe_type));
    QGauss qrule (dim, fe_type.default_quadrature_order());
    fe > attach_quadrature_rule(&qrule);

    const vector<Real>& JxW = fe > get_JxW();
    const vector<vector<Real>>& phi = fe > get_phi();


    vector<dof_id_type> dof_indices;
    vector< vector<dof_id_type> > dof_indices_var(Nvars);
    dof_id_type dof_index = vars[0];

    MeshBase::const_element_iterator el = mesh.active_local_elements_begin();
    const MeshBase::const_element_iterator end_el = mesh.active_local_elements_end();


    for ( ; el != end_el; ++el){

        const Elem* elem = *el;


        for (int var = 0; var < Nvars; var++)
            dof_map.dof_indices(elem, dof_indices_var[var], vars[var]);

        fe > reinit(elem);


            for (unsigned int i = 0; i < phi.size(); i++){

                for (int var = 0; var < Nvars; var++)
                    system.solution > set(dof_indices_var[var][i], value(var) + 1.e 8);

            }


    }


    system.solution > close();
    system.update();

    return;


}

void compute_gradients(EquationSystems& es, const string& /*system_name*/){


    const MeshBase& mesh = es.get_mesh();
    const unsigned int dim = mesh.mesh_dimension();

    NonlinearImplicitSystem& system = es.get_system<NonlinearImplicitSystem>("Usadel");

    const int Nvars = 8;
    vector<unsigned int> vars;
    system.get_all_variable_numbers(vars);
    sort(vars.begin(),vars.end());


    const DofMap& dof_map = system.get_dof_map();

    FEType fe_type = dof_map.variable_type(vars[0]);
    AutoPtr<FEBase> fe (FEBase::build(dim, fe_type));
    QGauss qrule (dim, fe_type.default_quadrature_order());
    fe > attach_quadrature_rule(&qrule);

    const vector<Real>& JxW = fe > get_JxW();
    const vector<vector<Real>>& phi = fe > get_phi();
    const vector<vector<RealGradient>>& dphi = fe > get_dphi();

    LinearImplicitSystem& gradsys = es.get_system<LinearImplicitSystem>("Gradients");


    vector<unsigned int> dvars;
    gradsys.get_all_variable_numbers(dvars);
    sort(dvars.begin(),dvars.end());


    const DofMap& grad_dof_map = gradsys.get_dof_map();

    DenseMatrix<Number> Ke;
    vector<vector< DenseSubMatrix<Number>* > > K;

    K.resize(3*Nvars);

    for (int i = 0; i < 3*Nvars; ++i)
        for (int j = 0; j < 3*Nvars; ++j)
            K[i].push_back( new DenseSubMatrix<Number>(Ke));

    DenseVector<Number> Re;
    vector<DenseSubVector<Number>*> R;

    for (int i = 0; i < 3*Nvars; ++i)
        R.push_back(new DenseSubVector<Number>(Re));


    vector<dof_id_type> dof_indices;
    vector<vector<dof_id_type>> dof_indices_var(Nvars);
    vector<vector<dof_id_type>> dof_indices_dvar(3*Nvars);

    MeshBase::const_element_iterator el = mesh.active_local_elements_begin();
    const MeshBase::const_element_iterator end_el = mesh.active_local_elements_end();



    for ( ; el != end_el; ++el){

        const Elem* elem = *el;

        grad_dof_map.dof_indices(elem,dof_indices);

        for (int var = 0; var < Nvars; var++)
            dof_map.dof_indices(elem,dof_indices_var[var], vars[var]);

        for (int var = 0; var < 3*Nvars; var++)
            grad_dof_map.dof_indices(elem, dof_indices_dvar[var], dvars[var]);

        const unsigned int n_dofs_dvar = dof_indices_dvar[0].size();
        fe > reinit(elem);

// Stiffness matrix
        Ke.resize(dof_indices.size(),dof_indices.size());
        Re.resize(dof_indices.size());

        for (int i = 0; i < 3*Nvars; ++i)
            R[i] > reposition(dvars[i]*n_dofs_dvar, n_dofs_dvar);

        for (int i = 0; i < 3*Nvars; ++i)
            for (int j = 0; j < 3*Nvars; ++j)
                K[i][j] > reposition(dvars[i]*n_dofs_dvar, dvars[j]*n_dofs_dvar,
                    n_dofs_dvar, n_dofs_dvar);


        for (unsigned int qp = 0; qp < qrule.n_points();qp++){
            vector<Number> dgx(Nvars), dgy(Nvars), dgz(Nvars);
            for (unsigned int i = 0; i < phi.size(); i++)
                for (unsigned int var = 0; var < Nvars; var++){

                    dgx[var] += dphi[i][qp](0)*system.current_solution(dof_indices_var[
                        var][i]);
                    dgy[var] += dphi[i][qp](1)*system.current_solution(dof_indices_var[
                        var][i]);
                    dgz[var] += dphi[i][qp](2)*system.current_solution(dof_indices_var[
                        var][i]);

                }

            for (unsigned int i = 0; i < phi.size(); i++)
                for (unsigned int j = 0; j < phi.size(); j++)
                    for (int comp = 0; comp < 3*Nvars; comp++)
                        K[comp][comp] > el(i,j) += JxW[qp]*phi[i][qp]*phi[j][qp];

// Residual vector
```

```
            for (unsigned int i = 0; i < phi.size(); i++){
                int counter = 0;
                for (int comp = 0; comp < 3*Nvars; comp++){

                    if (comp < Nvars){

                        R[comp]  > el(i) += JxW[qp]*dgx[counter]*phi[i][qp];

                    } else if ((comp >= Nvars) && (comp < 2*Nvars)){

                        if (comp == Nvars)
                            counter = 0;

                        R[comp]  > el(i) += JxW[qp]*dgy[counter]*phi[i][qp];

                    } else {

                        if (comp == 2*Nvars)
                            counter = 0;

                        R[comp]  > el(i) += JxW[qp]*dgz[counter]*phi[i][qp];
                    }
                    counter++;
                }
            }
        }


        grad_dof_map.constrain_element_matrix_and_vector(Ke,Re,dof_indices);
        gradsys.matrix  > add_matrix(Ke,dof_indices);
        gradsys.rhs  > add_vector (Re, dof_indices);




    }



    return;


}
```

B.2 compute˙RJ.cpp

```
#include <iostream>
#include <algorithm>
#include <sstream>
#include <math.h>
#include <armadillo>
#include <complex>
#include <utility>

#include "libmesh/libmesh.h"
#include "libmesh/mesh.h"
#include "libmesh/mesh_generation.h"
#include "libmesh/exodusII_io.h"
#include "libmesh/equation_systems.h"
#include "libmesh/fe.h"
#include "libmesh/quadrature_gauss.h"
#include "libmesh/dof_map.h"
#include "libmesh/sparse_matrix.h"
#include "libmesh/numeric_vector.h"
#include "libmesh/dense_matrix.h"
#include "libmesh/dense_vector.h"
#include "libmesh/linear_implicit_system.h"
#include "libmesh/nonlinear_implicit_system.h"
#include "libmesh/nonlinear_solver.h"
#include "libmesh/perf_log.h"
#include "libmesh/boundary_info.h"
#include "libmesh/utility.h"
#include "libmesh/dense_submatrix.h"
#include "libmesh/dense_subvector.h"
#include "libmesh/elem.h"
#include "libmesh/vtk_io.h"
#include "libmesh/gnuplot_io.h"
#include "libmesh/exodusII_io.h"
#include "libmesh/matlab_io.h"
#include "libmesh/fe_interface.h"
#include "compute_RJ.h"
#include "jacobian2.h"
#include "residual.h"
#include "BCS.h"
#include "MVConverter.h"
#include "magnetization.h"
#include "BChandler.h"
#include "externalflux.h"
#include "wrap.h"
#include "Connection.h"

using namespace std;
using namespace arma;
using namespace libMesh;

extern EquationSystems *_eqsys;
extern Connection* _connectivity;

void compute_RJ(const NumericVector<Number>& soln,
                NumericVector<Number>* residual,
                SparseMatrix<Number>* jacobian,
                NonlinearImplicitSystem& /*sys*/){

    //Get global equation system
    EquationSystems &es = *_eqsys;

    //Get connectivity between domains
    Connection &connectivity = *_connectivity;

    //Get mesh and its dimension
    const MeshBase& mesh    = es.get_mesh();
    const unsigned int dim = mesh.mesh_dimension();

    //Get system
    NonlinearImplicitSystem& usadel_system = es.get_system<NonlinearImplicitSystem>("
        Usadel");
```

```
//Get parameters
vector<double> parvec = es.parameters.get<vector<double>>("parvec");
vector<Input> parameters = unwrapper(parvec);
complex<double> E   = es.parameters.get<Number>("E");

//Define constants for convenience
const complex<double> ii(0,1);
const int Nvars = usadel_system.n_vars();

//Get boundary conditions
vector<double> BCstrength = es.parameters.get<vector<Real>>("BCstrength");
vector<double> BCpenalty  = es.parameters.get<vector<Real>>("BCpenalty");
vector<string> BCtype     = es.parameters.get<vector<string>>("BCtype");
vector<int> BCID          = es.parameters.get<vector<int>>("BCID");
vector<double> BCphase    = es.parameters.get<vector<double>>("BCphase");

//Wrap BC into a more convenient structure
int NBC = BCstrength.size();
vector<BCinputS> BC(NBC);

for (int j = 0; j < NBC; j++){

    BC[j].type = BCtype[j];
    BC[j].strength = BCstrength[j];
    BC[j].penalty = BCpenalty[j];
    BC[j].ID = BCID[j];
    BC[j].phase = BCphase[j];

}
//Get variable numbers
vector<unsigned int> vars;
usadel_system.get_all_variable_numbers(vars);
sort(vars.begin(), vars.end());

//Assign element type and integration rule
FEType fe_type = usadel_system.variable_type(vars[0]);

AutoPtr<FEBase> fe(FEBase::build(dim,fe_type));
QGauss qrule(dim, fe_type.default_quadrature_order());
fe > attach_quadrature_rule(&qrule);

AutoPtr<FEBase> fe_face(FEBase::build(dim,fe_type));
QGauss qface(dim  1, fe_type.default_quadrature_order());
fe_face > attach_quadrature_rule(&qface);

AutoPtr<FEBase> fe_neighbor_face(FEBase::build(dim, fe_type));
fe_neighbor_face > attach_quadrature_rule(&qface);

//Define weights*jacobian, shape functions and points
const vector<Real>& JxW                    = fe > get_JxW();
const vector<vector<Real>>& phi            = fe > get_phi();
const vector<vector<RealGradient>>& dphi = fe > get_dphi();
const vector<Point>& q_point               = fe > get_xyz();
const vector<Point>& face_point            = fe_face > get_xyz();

//Define residual vector
DenseVector<Number> Re;
vector< DenseSubVector<Number>* > R;

//Reshape R to allow for several variables
for (int i = 0; i < Nvars; ++i)
      R.push_back( new DenseSubVector<Number>(Re) );

//Define jacobian matrix
DenseMatrix<Number> Ke;
vector<vector< DenseSubMatrix<Number>* >> K;

//Reshape K to allow for several variables
K.resize(Nvars);
for (int i = 0; i < Nvars; ++i)
    for (int j = 0; j < Nvars; ++j)
        K[i].push_back( new DenseSubMatrix<Number>(Ke) );
```

```
//Get DOF map and define DOF indices
const DofMap& dof_map = usadel_system.get_dof_map();
vector<dof_id_type> dof_indices;
vector <vector<dof_id_type> > dof_indices_var(Nvars);
vector <vector<dof_id_type> > neighbor_dof_indices_var(Nvars);

//Initialize residual vector to zero
if(residual)
    residual > zero();

//Get list of boundaries that are connected
vector<int> cID = connectivity.get_connection_IDs();

//Broadcast solution so that neighbor element
//solutions are available in parallel
vector<Number> localized_solution;
soln.localize(localized_solution);



//Get element iterators
MeshBase::const_element_iterator el = mesh.active_local_elements_begin();
const MeshBase::const_element_iterator end_el = mesh.active_local_elements_end();

for ( ; el != end_el; ++el){

    //Get handle for current element
    const Elem* elem = *el;
    int sid = elem > subdomain_id();

    double Lx           = parameters[sid].dimensions[0];
    double Ly           = parameters[sid].dimensions[1];
    double Lz           = parameters[sid].dimensions[2];
    vector<double> flux = parameters[sid].ext_flux;
    vector<double> h    = parameters[sid].magnetization;

    double Eth = 1./Lx/Lx;
    //Assign indices to DOFs of current element
    dof_map.dof_indices(elem, dof_indices);

    //Assign DOF indices belonging to each individual variable
    for (int var = 0; var < Nvars; var++)
        dof_map.dof_indices(elem, dof_indices_var[var], vars[var]);


    //Define number of DOFs for convenience
    const unsigned int n_dofs = dof_indices.size();
    const unsigned int n_dofs_var = dof_indices_var[0].size();

    //Compute element specific data for current element
    fe > reinit(elem);

    //Resize jacobian and residual
    Re.resize(n_dofs);
    Ke.resize(n_dofs, n_dofs);


    //Move variable vectors/matrices to appropriate positions
    //in the global matrix system
    for (int i = 0; i < Nvars; ++i)
        R[i] > reposition(vars[i]*n_dofs_var, n_dofs_var);

    for (int i = 0; i < Nvars; ++i)
        for (int j = 0; j < Nvars; ++j)
            K[i][j] > reposition(vars[i]*n_dofs_var, vars[j]*n_dofs_var, n_dofs_var,
                n_dofs_var);


    //Loop over integration points
    for (unsigned int qp = 0; qp < qrule.n_points(); qp++){

        //Define vectors to hold solution and its derivative
```

```
vector<Number> solution(Nvars);
vector<Gradient> dsolution(Nvars);

//Get coordinates of given integration point
const Real x = q_point[qp](0);
const Real y = q_point[qp](1);

//Get solution and derivative in integration point
for (unsigned int i = 0; i < phi.size(); i++)
    for (unsigned int v = 0; v < Nvars; v++){

        solution[v]  += phi[i][qp]*soln(dof_indices_var[v][i]);
        dsolution[v].add_scaled(dphi[i][qp],soln(dof_indices_var[v][i]));

}

//Define arma vectors to hold solution
cx_vec gvec(Nvars), dgvec_x(Nvars), dgvec_y(Nvars), dgvec_z(Nvars);

//Transfer solution and derivatives
for (int v = 0; v < Nvars; v++){

    gvec(v) = solution[v];
    dgvec_x(v) = dsolution[v](0);
    dgvec_y(v) = dsolution[v](1);
    dgvec_z(v) = dsolution[v](2);

}

//Compute residual and jacobian
cx_vec Res = zeros<cx_vec>(Nvars);
cx_mat Jac = zeros<cx_mat>(Nvars,Nvars);

if (residual)
    Res = RC(gvec, dgvec_x, dgvec_y, dgvec_z, E, Lx, Ly, Lz, flux, h, x, y);

if (jacobian){
    Jac = JC(gvec, dgvec_x, dgvec_y, dgvec_z, E, Lx, Ly, Lz, flux, h, x, y);

}
Real v, p;
RealGradient dv, dp;
for (unsigned int i = 0; i < phi.size(); i++){

    //Define shape functions i
    v  = phi[i][qp];
    dv = dphi[i][qp];

    //Generate residual vector
    if (residual)
        for (unsigned int vars = 0; vars < Nvars; vars++){

            R[vars]  > el(i)  += JxW[qp]*( dgvec_x[vars]*dv(0)
                        (Lx/Ly)*(Lx/Ly)*dgvec_y[vars]*dv(1)
                        (Lx/Lz)*(Lx/Lz)*dgvec_z[vars]*dv(2)
                      + (Res[vars] + 2.*ii*(E/Eth)*gvec[vars])*v);
        }



    if (jacobian)
        for (unsigned int j = 0; j < phi.size(); j++){

            //Define shape functions j
            p  = phi[j][qp];
            dp = dphi[j][qp];

            //Compute linear contribution to jacobian
            Number Kel =  dp(0)*dv(0)    (Lx/Ly)*(Lx/Ly)*dp(1)*dv(1)    (Lx/Lz)
                *(Lx/Lz)*dp(2)*dv(2)
                + 2.*ii*(E/Eth)*v*p;

            //Insert linear contribution into jacobian matrix
```

```
                for (int comp = 0; comp < Nvars; comp++)
                    K[comp][comp]  > el(i,j) += JxW[qp]*Kel;

                //Insert nonlinear contribution into jacobian matrix
                for (int wrt = 0; wrt < Nvars; wrt++)
                    for (int comp = 0; comp < Nvars; comp++)
                        K[comp][wrt]  > el(i,j) += JxW[qp]*Jac(comp,wrt)*v*p;

            }
    }

    //End of sum over qp
}

//Compute boundary conditions
for (unsigned int s = 0; s < elem  > n_sides(); s++)
    if (elem  > neighbor(s) == NULL){

        //Get data from boundary side
        AutoPtr<Elem> side (elem  > build_side(s));

        //Get shape functions, weight and normals on surface
        const vector<Real>& JxW_face = fe_face  > get_JxW();
        const vector<vector<Real>>& phi_face = fe_face  > get_phi();
        const vector<vector<RealGradient>>& dphi_face = fe_face  > get_dphi();
        const vector<Point>& normals = fe_face  > get_normals();
        const vector<vector<Real>>& phi_neighbor_face = fe_neighbor_face  >
            get_phi();

        //Update element data
        fe_face  > reinit(elem,s);

        //Get boundary IDs
        int bc_id = mesh.boundary_info  > boundary_id(elem,s);

        if (find(cID.begin(), cID.end(), bc_id) != cID.end()){

            //Get neighbor element on other side of boundary
            dof_id_type neighbor_elem_ID = connectivity.get_neighbor_element_ID(
                elem  > id());
            const Elem* neighbor_elem = mesh.elem(neighbor_elem_ID);

            //Get neighbor element data on locations matching integration point
                of current element
            vector<Point> qface_neighbor_points;
            FEInterface::inverse_map(elem  > dim(), fe  > get_fe_type(),
                neighbor_elem, face_point, qface_neighbor_points);
            fe_neighbor_face  > reinit(neighbor_elem, &qface_neighbor_points);

            for (int var = 0; var < Nvars; var++)
                dof_map.dof_indices(neighbor_elem, neighbor_dof_indices_var[var],
                     vars[var]);

        }


        for (unsigned int qp = 0; qp < qface.n_points(); qp++){

            //Get surface positions
            const Real x = face_point[qp](0);
            const Real y = face_point[qp](1);

            //Get solution at surfac and at neighbor surface if connectede
            cx_vec solution(Nvars), neighbor_solution(Nvars);

            for (unsigned int i = 0; i < phi_face.size(); i++)
                for (int vars = 0; vars < Nvars; vars++){

                    solution(vars) += phi_face[i][qp]*soln(dof_indices_var[vars][
                        i]);
                    if (find(cID.begin(), cID.end(), bc_id) != cID.end())
                        neighbor_solution(vars) += phi_neighbor_face[i][qp]*
                            localized_solution[neighbor_dof_indices_var[vars][i
```

```
                                    ]];


                        }
                    //Get gamma matrices
                    cx_mat g  = v_to_m(solution);
                    cx_mat gt = v_to_mt(solution);
                    cx_mat gout, gtout;

                    if (find(cID.begin(), cID.end(), bc_id) != cID.end()){

                        gout = v_to_m(neighbor_solution);
                        gtout = v_to_mt(neighbor_solution);

                    }

                    //Loop over shape functions and BCIDs, insert into residual
                    if (residual)
                        for (unsigned int i = 0; i < phi_face.size(); i++)
                            for (int ibc = 0; ibc < NBC; ibc++)
                                if (bc_id == BC[ibc].ID){

                                    //Get BC value
                                    cx_vec value = BChandler(BC[ibc], g, gt, E, flux, x,
                                        y, normals[qp], gout, gtout);

                                    for (int comp = 0; comp < Nvars; comp++)
                                        R[comp]  > el(i) += JxW_face[qp]*BC[ibc].penalty*
                                            value(comp)*phi_face[i][qp];
                                }

                    //Loop over shape functions and BCIDs, insert into jacobian
                    if (jacobian)
                        for (unsigned int i = 0; i < phi_face.size(); i++)
                            for (unsigned int j = 0; j < phi_face.size(); j++)
                                for (int ibc = 0; ibc < NBC; ibc++)
                                    if (bc_id == BC[ibc].ID)
                                        for (int wrt = 0; wrt < Nvars; wrt++){

                                            cx_vec KBC = JBC(wrt, BC[ibc], g, gt, E, flux
                                                , x, y, normals[qp], gout, gtout);

                                            for (int comp = 0; comp < Nvars; comp++)
                                                K[comp][wrt]  > el(i,j) += JxW_face[qp]*
                                                    BC[ibc].penalty*KBC(comp)*phi_face[i
                                                    ][qp]*phi_face[j][qp];
                                        }

                }//End sum over qp

            }//End sum over element sides


        //Add jacobian and residual into global matrix system
        if (residual){

            dof_map.constrain_element_vector(Re, dof_indices);
            residual  > add_vector(Re, dof_indices);
        }

        if (jacobian){

            dof_map.constrain_element_matrix(Ke, dof_indices);
            jacobian  > add_matrix(Ke, dof_indices, dof_indices);
        }

    }

    return;
}
```

B.3 jacobian2.cpp

```
#include ”jacobian2.h”
#include ”magnetization.h”
#include ”externalflux.h”
#include ”MVConverter.h”



cx_mat JC(cx_vec gvec, cx_vec dgvec_x, cx_vec dgvec_y, cx_vec dgvec_z, complex<double> E,
     double Lx, double Ly, double Lz, vector<double> flux, vector<double> h, double x,
    double y){

    //Compute Thouless energy
    double Eth = 1./Lx/Lx;

    //Prepare jacobians
    cx_mat J, Jt;
    cx_mat J_main, Jt_main;
    cx_mat J_A  = zeros<cx_mat>(2,2);
    cx_mat Jt_A = zeros<cx_mat>(2,2);
    cx_mat J_H  = zeros<cx_mat>(2,2);
    cx_mat Jt_H = zeros<cx_mat>(2,2);
    cx_mat Jac(8,8);

    //Prepare matrices involved with differentiation
    cx_mat M(2,2), Mt(2,2), M_t(2,2), Mt_t(2,2);
    mat Lm(2,2);


    //Create gamma matrices and their derivatives
    cx_mat g     = v_to_m(gvec);
    cx_mat gt    = v_to_mt(gvec);
    cx_mat dg_x  = v_to_m(dgvec_x);
    cx_mat dg_y  = v_to_m(dgvec_y);
    cx_mat dg_z  = v_to_m(dgvec_z);
    cx_mat dgt_x = v_to_mt(dgvec_x);
    cx_mat dgt_y = v_to_mt(dgvec_y);
    cx_mat dgt_z = v_to_mt(dgvec_z);

    //Create s matrices
    mat sx, sz; cx_mat sy;
    complex<double> ii(0,1);

    sx <<  0  <<  1  << endr
       <<  1  <<  0  << endr;

    sy <<  0  <<  ii << endr
       << ii  <<  0  << endr;

    sz <<  1  <<  0  << endr
       <<  0  <<  1  << endr;

    //Create N and Nt and its determinant
    mat I(2,2); I.eye();

    cx_mat Ninv = I   g*gt;
    cx_mat N = inv(Ninv);

    cx_mat Ntinv = I   gt*g;
    cx_mat Nt = inv(Ntinv);

    complex<double> D = det(Ninv);

    //Compute external flux
    cx_mat Ax, Ay, Az, A2;
    SOC A = externalflux(flux, x, y);
    Ax = A.x;
    Ay = A.y;
    Az = A.z;
    A2 = Ax*Ax + Ay*Ay + Az*Az;
```

```
//Compute magnetization
vector<double> H = magnetization(h,x,y);


//***********************************************
// Differentiation wrt gamma
//**********************************************

for (int wrt = 0; wrt < 8; wrt++){

    MatrixCompute(&M, &Mt, &M_t, &Mt_t, &Lm, g, gt, wrt);
    complex<double> dD = dDCompute(g,gt,wrt);

    if (wrt < 4){

        //**************************//
        // Differentiation wrt gamma //
        //**************************//

        //Main contribution
        J_main    =                 2.*dg_x*(1./D)*(Mt    dD*Nt)*gt*dg_x
                + 2.*(Lx/Ly)*(Lx/Ly)*dg_y*(1./D)*(Mt    dD*Nt)*gt*dg_y
                + 2.*(Lx/Lz)*(Lx/Lz)*dg_z*(1./D)*(Mt    dD*Nt)*gt*dg_z;

        Jt_main   =                 2.*dgt_x*((1./D)*(M    dD*N)*g + N*Lm)*dgt_x
                + 2.*(Lx/Ly)*(Lx/Ly)*dgt_y*((1./D)*(M    dD*N)*g + N*Lm)*dgt_y;
                + 2.*(Lx/Lz)*(Lx/Lz)*dgt_z*((1./D)*(M    dD*N)*g + N*Lm)*dgt_z;


        //Contribution from external flux and SOC
        J_A   = 2.*ii*(Lm*conj(Ax)*gt*N*dg_x + (Ax + g*conj(Ax)*gt)*(1./D)*(M    dD*N)
            *dg_x + dg_x*(1./D)*(Mt    dD*Nt)*(conj(Ax) + gt*Ax*g) + dg_x*Nt*gt*Ax*Lm)
                2.*ii*(Lm*conj(Ay)*gt*N*dg_y + (Ay + g*conj(Ay)*gt)*(1./D)*(M    dD*N)
                    *dg_y + dg_y*(1./D)*(Mt    dD*Nt)*(conj(Ay) + gt*Ay*g) + dg_y*Nt*gt
                    *Ay*Lm)
                2.*ii*(Lm*conj(Az)*gt*N*dg_z + (Az + g*conj(Az)*gt)*(1./D)*(M    dD*N)
                    *dg_z + dg_z*(1./D)*(Mt    dD*Nt)*(conj(Az) + gt*Az*g) + dg_z*Nt*gt
                    *Az*Lm)
                2.*( (Ax*Lm + Lm*conj(Ax))*Nt*(conj(Ax) + gt*Ax*g) + (Ax*g + g*conj(
                    Ax))*(1./D)*(Mt    dD*Nt)*(conj(Ax) + gt*Ax*g) + (Ax*g+g*conj(Ax))*
                    Nt*gt*Ax*Lm)
                2.*( (Ay*Lm + Lm*conj(Ay))*Nt*(conj(Ay) + gt*Ay*g) + (Ay*g + g*conj(
                    Ay))*(1./D)*(Mt    dD*Nt)*(conj(Ay) + gt*Ay*g) + (Ay*g+g*conj(Ay))*
                    Nt*gt*Ay*Lm)
                2.*( (Az*Lm + Lm*conj(Az))*Nt*(conj(Az) + gt*Az*g) + (Az*g + g*conj(
                    Az))*(1./D)*(Mt    dD*Nt)*(conj(Az) + gt*Az*g) + (Az*g+g*conj(Az))*
                    Nt*gt*Az*Lm)
                A2*Lm + Lm*conj(A2);

        Jt_A = 2.*ii*(gt*Ax*Lm*Nt*dgt_x + (conj(Ax) + gt*Ax*g)*(1./D)*(Mt    dD*Nt)*
            dgt_x + dgt_x*(1./D)*(M    dD*N)*(Ax + g*conj(Ax)*gt) + dgt_x*N*Lm*conj(Ax
            )*gt)
              +2.*ii*(gt*Ay*Lm*Nt*dgt_y + (conj(Ay) + gt*Ay*g)*(1./D)*(Mt    dD*Nt)*
                    dgt_y + dgt_y*(1./D)*(M    dD*N)*(Ay + g*conj(Ay)*gt) + dgt_y*N*Lm*
                    conj(Ay)*gt)
              +2.*ii*(gt*Az*Lm*Nt*dgt_z + (conj(Az) + gt*Az*g)*(1./D)*(Mt    dD*Nt)*
                    dgt_z + dgt_z*(1./D)*(M    dD*N)*(Az + g*conj(Az)*gt) + dgt_z*N*Lm*
                    conj(Az)*gt)
               2.*(conj(Ax)*gt + gt*Ax)*((1./D)*(M    dD*N)*(Ax + g*conj(Ax)*gt) + N*
                   Lm*conj(Ax)*gt)
               2.*(conj(Ay)*gt + gt*Ay)*((1./D)*(M    dD*N)*(Ay + g*conj(Ay)*gt) + N*
                   Lm*conj(Ay)*gt)
               2.*(conj(Az)*gt + gt*Az)*((1./D)*(M    dD*N)*(Az + g*conj(Az)*gt) + N*
                   Lm*conj(Az)*gt);

        //Contribution from magnetization
        J_H = ii*(H[0]*(sx*Lm    Lm*sx) + H[1]*(sy*Lm    Lm*conj(sy)) + H[2]*(sz*Lm
            Lm*sz))/Eth;

    } else {

        //********************************//
        // Differentiation wrt gamma_tilde //
```

```
            //********************************//

            //Main contribution
            J_main  =                  2.*dg_x*( (1./D) * (Mt_t   dD*Nt)*gt + Nt*Lm )*dg_x
                    + 2.*(Lx/Ly)*(Lx/Ly)*dg_y*( (1./D) * (Mt_t   dD*Nt)*gt + Nt*Lm )*dg_y
                    + 2.*(Lx/Lz)*(Lx/Lz)*dg_z*( (1./D) * (Mt_t   dD*Nt)*gt + Nt*Lm )*dg_z
                        ;

            Jt_main =                  2.*dgt_x*(1./D)*(M_t   dD*N)*g*dgt_x
                    + 2.*(Lx/Ly)*(Lx/Ly)*dgt_y*(1./D)*(M_t   dD*N)*g*dgt_y
                    + 2.*(Lx/Lz)*(Lx/Lz)*dgt_z*(1./D)*(M_t   dD*N)*g*dgt_z;

            //Contribution from external flux and SOC
            J_A  =  2.*ii*(g*conj(Ax)*Lm*N*dg_x + (Ax + g*conj(Ax)*gt)*(1./D)*(M_t   dD*N
                 )*dg_x + dg_x*(1./D)*(Mt_t   dD*Nt)*(conj(Ax) + gt*Ax*g) + dg_x*Nt*Lm*Ax*
                 g)
                    2.*ii*(g*conj(Ay)*Lm*N*dg_y + (Ay + g*conj(Ay)*gt)*(1./D)*(M_t   dD*N
                       )*dg_y + dg_y*(1./D)*(Mt_t   dD*Nt)*(conj(Ay) + gt*Ay*g) + dg_y*Nt
                       *Lm*Ay*g)
                    2.*ii*(g*conj(Az)*Lm*N*dg_z + (Az + g*conj(Az)*gt)*(1./D)*(M_t   dD*N
                       )*dg_z + dg_z*(1./D)*(Mt_t   dD*Nt)*(conj(Az) + gt*Az*g) + dg_z*Nt
                       *Lm*Az*g)
                    2.*(Ax*g+g*conj(Ax))*( (1./D)*(Mt_t   dD*Nt)*(conj(Ax) + gt*Ax*g) +Nt
                       *Lm*Ax*g)
                    2.*(Ay*g+g*conj(Ay))*( (1./D)*(Mt_t   dD*Nt)*(conj(Ay) + gt*Ay*g) +Nt
                       *Lm*Ay*g)
                    2.*(Az*g+g*conj(Az))*( (1./D)*(Mt_t   dD*Nt)*(conj(Az) + gt*Az*g) +Nt
                       *Lm*Az*g);

            Jt_A = 2.*ii*(Lm*Ax*g*Nt*dgt_x + (conj(Ax) + gt*Ax*g)*(1./D)*(Mt_t   dD*Nt)*
                 dgt_x + dgt_x*(1./D)*(M_t   dD*N)*(Ax + g*conj(Ax)*gt) + dgt_x*N*g*conj(
                 Ax)*Lm)
                   +2.*ii*(Lm*Ay*g*Nt*dgt_y + (conj(Ay) + gt*Ay*g)*(1./D)*(Mt_t   dD*Nt)*
                       dgt_y + dgt_y*(1./D)*(M_t   dD*N)*(Ay + g*conj(Ay)*gt) + dgt_y*N*g*
                       conj(Ay)*Lm)
                   +2.*ii*(Lm*Az*g*Nt*dgt_z + (conj(Az) + gt*Az*g)*(1./D)*(Mt_t   dD*Nt)*
                       dgt_z + dgt_z*(1./D)*(M_t   dD*N)*(Az + g*conj(Az)*gt) + dgt_z*N*g*
                       conj(Az)*Lm)
                    2.*( (conj(Ax)*Lm + Lm*Ax)*N*(Ax + g*conj(Ax)*gt) + (conj(Ax)*gt + gt*
                       Ax)*(1./D)*(M_t   dD*N)*(Ax + g*conj(Ax)*gt) + (conj(Ax)*gt + gt*Ax
                       )*N*g*conj(Ax)*Lm)
                    2.*( (conj(Ay)*Lm + Lm*Ay)*N*(Ay + g*conj(Ay)*gt) + (conj(Ay)*gt + gt*
                       Ay)*(1./D)*(M_t   dD*N)*(Ay + g*conj(Ay)*gt) + (conj(Ay)*gt + gt*Ay
                       )*N*g*conj(Ay)*Lm)
                    2.*( (conj(Az)*Lm + Lm*Az)*N*(Az + g*conj(Az)*gt) + (conj(Az)*gt + gt*
                       Az)*(1./D)*(M_t   dD*N)*(Az + g*conj(Az)*gt) + (conj(Az)*gt + gt*Az
                       )*N*g*conj(Az)*Lm)
                    conj(A2)*Lm + Lm*A2;

            //Contribution from magnetization
            Jt_H  =  ii*(H[0]*(sx*Lm   Lm*sx) + H[1]*(conj(sy)*Lm   Lm*sy) + H[2]*(sz*Lm
                  Lm*sz))/Eth;

        }

        //Addition of contribution to jacobian
        J  = J_main + J_A + J_H;
        Jt = Jt_main + Jt_A + Jt_H;

        //Reshape into column wrt
        cx_vec Jwrt  = m_to_v(J);
        cx_vec Jtwrt = m_to_v(Jt);

        Jac(span(0,3),wrt) = Jwrt;
        Jac(span(4,7),wrt) = Jtwrt;

    }

    return Jac;


}
```

```
void MatrixCompute(cx_mat* M,cx_mat* Mt,cx_mat* M_t,cx_mat* Mt_t, mat* Lm, cx_mat g,
    cx_mat gt, int index){


    if ((index == 0) || (index == 4)){



        Lm > at(0,0) = 1; Lm > at(0,1) = 0;
        Lm > at(1,0) = 0; Lm > at(1,1) = 0;



    } else if ((index == 1) || (index == 5)){



        Lm > at(0,0) = 0; Lm > at(0,1) = 1;
        Lm > at(1,0) = 0; Lm > at(1,1) = 0;


    } else if ((index == 2) || (index == 6)){



        Lm > at(0,0) = 0; Lm > at(0,1) = 0;
        Lm > at(1,0) = 1; Lm > at(1,1) = 0;


    } else if ((index == 3) || (index == 7)){



        Lm > at(0,0) = 0; Lm > at(0,1) = 0;
        Lm > at(1,0) = 0; Lm > at(1,1) = 1;


    }







    if (index == 0){


        Mt > at(0,0) =  0;        Mt > at(0,1) =  0;
        Mt > at(1,0) =  gt(1,0); Mt > at(1,1) =  gt(0,0);

        M > at(0,0) = 0; M > at(0,1) = gt(0,1);
        M > at(1,0) = 0; M > at(1,1) =  gt(0,0);

    } else if (index == 1){


        Mt > at(0,0) =   gt(1,0); Mt > at(0,1) =  gt(0,0);
        Mt > at(1,0) =  0;         Mt > at(1,1) = 0;

        M > at(0,0) = 0; M > at(0,1) = gt(1,1);
        M > at(1,0) = 0; M > at(1,1) =  gt(1,0);


    } else if (index == 2){

        Mt > at(0,0) =  0;        Mt > at(0,1) =  0;
        Mt > at(1,0) =  gt(1,1); Mt > at(1,1) =  gt(0,1);

        M > at(0,0) =  gt(0,1); M > at(0,1) = 0;
        M > at(1,0) = gt(0,0); M > at(1,1) = 0;
    } else if (index == 3){


        Mt  >  at(0,0) =   gt(1,1); Mt  >  at(0,1) =  gt(0,1);
        Mt  >  at(1,0) =  0;        Mt  >  at(1,1) = 0;

        M  >  at(0,0) =  gt(1,1); M  >  at(0,1) = 0;
        M  >  at(1,0) = gt(1,0); M  >  at(1,1) = 0;



    } else if (index == 4){

        Mt_t  >  at(0,0) =  0; Mt_t  >  at(0,1) =  g(0,1);
        Mt_t  >  at(1,0) =  0; Mt_t  >  at(1,1) =  g(0,0);

        M_t  >  at(0,0) = 0;       M_t  >  at(0,1) = 0;
        M_t  >  at(1,0) = g(1,0); M_t  >  at(1,1) =  g(0,0);


    } else if (index == 5){

        Mt_t  >  at(0,0) =  0; Mt_t  >  at(0,1) =  g(1,1);
        Mt_t  >  at(1,0) =  0; Mt_t  >  at(1,1) =  g(1,0);

        M_t  >  at(0,0) =  g(1,0); M_t  >  at(0,1) = g(0,0);
        M_t  >  at(1,0) = 0;        M_t  >  at(1,1) = 0;



    } else if (index == 6){

        Mt_t  >  at(0,0) =   g(0,1); Mt_t  >  at(0,1) = 0;
        Mt_t  >  at(1,0) =  g(0,0);  Mt_t  >  at(1,1) = 0;

        M_t  >  at(0,0) = 0;       M_t  >  at(0,1) = 0;
        M_t  >  at(1,0) = g(1,1); M_t  >  at(1,1) =  g(0,1);


    } else if (index == 7){

        Mt_t  >  at(0,0) =   g(1,1); Mt_t  >  at(0,1) = 0;
        Mt_t  >  at(1,0) =   g(1,0); Mt_t  >  at(1,1) = 0;

        M_t  >  at(0,0) =  g(1,1); M_t  >  at(0,1) = g(0,1);
        M_t  >  at(1,0) = 0;        M_t  >  at(1,1) = 0;

    }

}

complex<double> dDCompute(cx_mat g, cx_mat gt, int index){

    complex<double> dD;

    if (index == 0){

        dD = g(1,1)*gt(0,0)*gt(1,1)    gt(0,0)    g(1,1)*gt(0,1)*gt(1,0);


    } else if (index == 1){

        dD = g(1,0)*gt(0,1)*gt(1,0)    gt(1,0)    g(1,0)*gt(0,0)*gt(1,1);


    } else if (index == 2){

        dD = g(0,1)*gt(0,1)*gt(1,0)    g(0,1)*gt(0,0)*gt(1,1)    gt(0,1);


    } else if (index == 3){
        dD = g(0,0)*gt(0,0)*gt(1,1)    gt(1,1)    g(0,0)*gt(0,1)*gt(1,0);


    } else if (index == 4){
        dD = g(0,0)*g(1,1)*gt(1,1)    g(0,0)   g(0,1)*g(1,0)*gt(1,1);

    } else if (index == 5){

        dD =  g(1,0)    g(0,0)*g(1,1)*gt(1,0) + g(0,1)*g(1,0)*gt(0,1);

    } else if (index == 6){
        dD =  g(0,1)    g(0,0)*g(1,1)*gt(0,1) + g(0,1)*g(1,0)*gt(0,1);

    } else if (index == 7){
        dD = g(0,0)*g(1,1)*gt(0,0)    g(1,1)    g(0,1)*g(1,0)*gt(0,0);
    }
    return dD;
}
```

B.4 residual.cpp

```
#include <armadillo>
#include "residual.h"
#include "MVConverter.h"
#include "externalflux.h"
#include "magnetization.h"

using namespace arma;
using namespace std;


cx_vec RC(cx_vec gvec, cx_vec dgvec_x, cx_vec dgvec_y, cx_vec dgvec_z, complex<double> E,
     double Lx, double Ly, double Lz, vector<double> flux, vector<double> h, double x,
    double y){


    //Compute Thouless energy
    double Eth = 1./Lx/Lx;

    //Create gamma matrices and their derivatives
    cx_mat g     = v_to_m(gvec);
    cx_mat gt    = v_to_mt(gvec);
    cx_mat dg_x  = v_to_m(dgvec_x);
    cx_mat dg_y  = v_to_m(dgvec_y);
    cx_mat dg_z  = v_to_m(dgvec_z);
    cx_mat dgt_x = v_to_mt(dgvec_x);
    cx_mat dgt_y = v_to_mt(dgvec_y);
    cx_mat dgt_z = v_to_mt(dgvec_z);

    //Create sigma matrices
    mat sx, sz; cx_mat sy;
    complex<double> ii(0,1);

    sx << 0  << 1  << endr
       << 1  << 0  << endr;

    sy << 0  << ii << endr
       << ii << 0  << endr;

    sz << 1  << 0  << endr
       << 0  << 1  << endr;

    //Create N and Nt
    mat I(2,2); I.eye();

    cx_mat Ninv = I    g*gt;
    cx_mat N = inv(Ninv);

    cx_mat Ntinv = I    gt*g;
    cx_mat Nt = inv(Ntinv);

    //Compute external flux
    cx_mat Ax, Ay, Az, A2;
    SOC A = externalflux(flux, x, y);
    Ax = A.x;
    Ay = A.y;
    Az = A.z;
    A2 = Ax*Ax + Ay*Ay + Az*Az;

    //Compute magnetization
    vector<double> H = magnetization(h,x,y);

    //Compute main contribution from the Usadel equation
    cx_mat K_main  = 2.*dg_x*Nt*gt*dg_x + 2.*(Lx/Ly)*(Lx/Ly)*dg_y*Nt*gt*dg_y + 2.*(Lx/Lz)
        *(Lx/Lz)*dg_z*Nt*gt*dg_z;
    cx_mat Kt_main = 2.*dgt_x*N*g*dgt_x + 2.*(Lx/Ly)*(Lx/Ly)*dgt_y*N*g*dgt_y + 2.*(Lx/Lz)
        *(Lx/Lz)*dgt_z*N*g*dgt_z;

    //Compute contribution from external flux and SOC
    cx_mat K_A =  2.*ii*((Ax + g*conj(Ax)*gt)*N*dg_x + dg_x*Nt*(conj(Ax) + gt*Ax*g))
                  2.*ii*((Ay + g*conj(Ay)*gt)*N*dg_y + dg_y*Nt*(conj(Ay) + gt*Ay*g))
```

```
                2.*ii*((Az + g*conj(Az)*gt)*N*dg_z + dg_z*Nt*(conj(Az) + gt*Az*g))
                2.*(Ax*g + g*conj(Ax))*Nt*(conj(Ax) + gt*Ax*g)
                2.*(Ay*g + g*conj(Ay))*Nt*(conj(Ay) + gt*Ay*g)
                2.*(Az*g + g*conj(Az))*Nt*(conj(Az) + gt*Az*g)
                (A2*g   g*conj(A2));

    cx_mat Kt_A = +2.*ii*((conj(Ax) + gt*Ax*g)*Nt*dgt_x + dgt_x*N*(Ax + g*conj(Ax)*gt))
                +2.*ii*((conj(Ay) + gt*Ay*g)*Nt*dgt_y + dgt_y*N*(Ay + g*conj(Ay)*gt))
                +2.*ii*((conj(Az) + gt*Az*g)*Nt*dgt_z + dgt_z*N*(Az + g*conj(Az)*gt))
                 2.*(conj(Ax)*gt+gt*Ax)*N*(Ax + g*conj(Ax)*gt)
                 2.*(conj(Ay)*gt+gt*Ay)*N*(Ay + g*conj(Ay)*gt)
                 2.*(conj(Az)*gt+gt*Az)*N*(Az + g*conj(Az)*gt)
                 (conj(A2)*gt   gt*A2);

    //Compute contribution from magnetization
    cx_mat K_H  =   ii*(H[0]*(sx*g   g*sx)   + H[1]*(sy*g   g*conj(sy))   + H[2]*(sz*g   g
        *sz))/Eth;
    cx_mat Kt_H =   ii*(H[0]*(sx*gt   gt*sx) + H[1]*(conj(sy)*gt   gt*sy) + H[2]*(sz*gt
        gt*sz))/Eth;

    //Add contributions
    cx_mat Kmat  = K_main + K_A + K_H;
    cx_mat Ktmat = Kt_main + Kt_A + Kt_H;

    //Reshape to vector
    cx_vec Kvec  = m_to_v(Kmat);
    cx_vec Ktvec = mt_to_v(Ktmat);

    cx_vec K = join_vert(Kvec,Ktvec);

    return K;

}
```

B.5 BChandler.cpp

```
#include <armadillo>
#include "MVConverter.h"
#include "customstructs.h"
#include "BCS.h"
#include "jacobian.h"
#include "BChandler.h"
#include "libmesh/libmesh.h"
#include "libmesh/point.h"
#include "externalflux.h"

using namespace std;
using namespace arma;

cx_vec BChandler(BCinputS BC, cx_mat g, cx_mat gt, complex<double> E, vector<double> flux
    , double xpts, double ypts, libMesh::Point n, cx_mat gout, cx_mat gtout){

    //BCtype
    //
    //The first element contains information about which BCs to use. Possibilities:
    //KL_BCS = Kupriyanov Lukichev with infinite BCS
    //Trans_BCS = Transparent with infinite BCS
    //Vacuum
    //The second element states whether the outward normal of the boundary is positve or
        negative
    //
    //
    //R
    //
    //Contains the strength of the Kupriyanov Lukichev BC, it is otherwise ignored.


    int Nvars = 8;
    complex<double> ii(0,1);
    cx_vec value(Nvars), y(4), yt(4);
    cx_mat vmat, vtmat, Nout, Ntout;

    double R = BC.strength;
    double Phi = BC.phase;

    mat I(2,2); I.eye();

    double sx = flux[0];
    if (gout.is_empty()){

        BCout(BC, gout, gtout, Nout, Ntout, E, Phi);

    } else {

        cx_mat Noutinv, Ntoutinv;
        Noutinv = I   gout*gtout;
        Ntoutinv = I   gtout*gout;
        Nout = inv(Noutinv);
        Ntout = inv(Ntoutinv);

    }


    cx_mat Ax, Ay, Az, An;
    SOC A = externalflux(flux, xpts, ypts);

    Ax = A.x;
    Ay = A.y;
    Az = A.z;

    An = n(0)*Ax + n(1)*Ay + n(2)*Az;


    if ((BC.type == "KL_BCS_Right") || (BC.type == "KL_Con")){

//          vmat = (1./R)*(I   g*gtout)*Nout*(g   gout)   2.*ii*sx*ypts*g;
//          vtmat = (1./R)*(I    gt*gout)*Ntout*(gt    gtout) + 2.*ii*sx*ypts*gt;
            vmat  = (1./R)*(I    g*gtout)*Nout*(gout    g) + ii*(An*g + g*conj(An));
            vtmat = (1./R)*(I    gt*gout)*Ntout*(gtout    gt)    ii*(conj(An)*gt + gt*An);


    } else if (BC.type == "KL_BCS_Left"){

            vmat  =  (1./R)*(I    g*gtout)*Nout*(g    gout) + 2.*ii*sx*ypts*g;
            vtmat =  (1./R)*(I    gt*gout)*Ntout*(gt    gtout)    2.*ii*sx*ypts*gt;

    } else if (BC.type == "KL_N"){

//          vmat  = (1./R)*(I    g*gtout)*Nout*(gout    g);
//          vtmat = (1./R)*(I    gt*gout)*Ntout*(gtout    gt);
            vmat  =  (1./R)*g + ii*(An*g + g*conj(An));
            vtmat =  (1./R)*gt    ii*(conj(An)*gt + gt*An);

    } else if (BC.type == "Trans_BCS"){

            //Note this is a Dirichlet BC.

            vmat = g    gout;
            vtmat = gt    gtout;


    } else if (BC.type == "Vacuum_Right"){

            vmat =   ii*(An*g + g*conj(An));
            vtmat =   ii*(conj(An)*gt + gt*An);

    } else if (BC.type == "Vacuum_Left"){

            vmat = 2.*ii*sx*ypts*g;
            vtmat =  2.*ii*sx*ypts*gt;

    }


    y = m_to_v(vmat);
    yt = mt_to_v(vtmat);

    value = join_vert(y,yt);


    return value;


}

cx_vec JBC(int index, BCinputS BC, cx_mat g, cx_mat gt, complex<double> E, vector<double>
     flux, double xpts, double ypts, libMesh::Point n, cx_mat gout, cx_mat gtout){

    cx_mat M(2,2), Mt(2,2), M_t(2,2), Mt_t(2,2), vmat(2,2), vtmat(2,2);
    cx_mat Nout, Ntout;
    mat Lm(2,2);
    mat I(2,2); I.eye();
    cx_mat J;
    cx_vec y, yt, BCvalue;
    complex<double> ii(0,1);
    double R = BC.strength;
    double Phi = BC.phase;
    double sx = flux[0];
    vmat.zeros();
    vtmat.zeros();

    cx_mat Ax, Ay, Az, An;
    SOC A = externalflux(flux, xpts, ypts);

    Ax = A.x;
    Ay = A.y;
    Az = A.z;
    An = n(0)*Ax + n(1)*Ay + n(2)*Az;
    MatrixCompute(&M, &Mt, &M_t, &Mt_t, &Lm, g, gt, index);

    if (gout.is_empty()){

        BCout(BC, gout, gtout, Nout, Ntout, E, Phi);

    } else {

        cx_mat Noutinv, Ntoutinv;
        Noutinv = I   gout*gtout;
        Ntoutinv = I   gtout*gout;
        Nout = inv(Noutinv);
        Ntout = inv(Ntoutinv);

    }

    if ((BC.type == "KL_BCS_Right") || (BC.type == "KL_Con")){

        if (index < 4){

//              vmat  = (1./R)*( Lm*gtout*Nout*(g   gout) + (I   g*gtout)*Nout*Lm)   2.*ii*
    sx*ypts*Lm;
            vmat  = (1./R)*( Lm*gtout*Nout*(gout   g)   (I   g*gtout)*Nout*Lm) +ii*(An*Lm
                + Lm*conj(An));

        } else {

//              vtmat = (1./R)*( Lm*gout*Ntout*(gt   gtout) + (I   gt*gout)*Ntout*Lm) + 2.*
    ii*sx*ypts*Lm;
            vtmat = (1./R)*( Lm*gout*Ntout*(gtout   gt)   (I   gt*gout)*Ntout*Lm)   ii*(
                conj(An)*Lm + Lm*An);

        }

    } else if (BC.type == "KL_BCS_Left"){

        if (index < 4){

//              vmat  =  (1./R)*( Lm*gtout*Nout*(gout   g)   (I   g*gtout)*Nout*Lm)   2.*ii
    *sx*ypts*Lm;
            vmat  =  (1./R)*( Lm*gtout*Nout*(g   gout) + (I   g*gtout)*Nout*Lm) + 2.*ii*
                sx*ypts*Lm;

        } else {

//              vtmat =  (1./R)*( Lm*gout*Ntout*(gtout   gt)   (I   gt*gout)*Ntout*Lm) +
    2.*ii*sx*ypts*Lm;
            vtmat =  (1./R)*( Lm*gout*Ntout*(gt   gtout) + (I   gt*gout)*Ntout*Lm)   2.*
                ii*sx*ypts*Lm;

        }

    } else if (BC.type == "Trans_BCS"){

        if (index < 4){

            vmat = Lm*ii/ii;

        } else {

            vtmat = Lm*ii/ii;

        }

    } else if (BC.type == "Vacuum_Right"){

        if (index < 4){

            vmat = ii*(An*Lm + Lm*conj(An));

        } else {

            vtmat =  ii*(conj(An)*Lm + Lm*An);
        }

    } else if (BC.type == ”Vacuum_Left”){

        if (index < 4){

            vmat = 2.*ii*sx*ypts*Lm;

        } else {

            vtmat =  2.*ii*sx*ypts*Lm;

        }



    } else if (BC.type == ”KL_N”){

        if (index < 4){

        vmat  =  (1./R)*Lm + ii*(An*Lm + Lm*conj(An));

        } else {

        vtmat =  (1./R)*Lm   ii*(conj(An)*Lm + Lm*An);

        }
    }

    y = m_to_v(vmat);
    yt = mt_to_v(vtmat);

    BCvalue = join_vert(y,yt);


    return BCvalue;

}

void BCout(BCinputS BC, cx_mat &gout, cx_mat &gtout, cx_mat &Nout, cx_mat &Ntout, complex
    <double> E, double Phi){


    mat I(2,2); I.eye();
    cx_mat Noutinv;
    cx_mat Ntoutinv;

    if ((BC.type == ”KL_BCS_Right”) || (BC.type == ”KL_BCS_Left”) || (BC.type == ”
        Trans_BCS”)){

        cx_vec yBCS = BCS(E, Phi);
        gout = v_to_m(yBCS);
        gtout = v_to_mt(yBCS);
        Noutinv = I   gout*gtout;
        Ntoutinv = I   gtout*gout;
        Nout = inv(Noutinv);
        Ntout = inv(Ntoutinv);


    }


}
```

B.6 externalflux.cpp

```
#include <armadillo>
#include "customstructs.h"

 using namespace std;
 using namespace arma;

SOC externalflux(vector<double> ext_flux, double x, double y){


    SOC A;
    double z = 0;
    complex<double> ii(0,1);
    vector<double> sx(3), sy(3), sz(3);
    double a, chi;
    for (int j = 0; j < 3; j++){

        sx[j] = ext_flux[j];
        sy[j] = ext_flux[j+3];
        sz[j] = ext_flux[j+6];

    }

    a   = ext_flux[9];
    chi = ext_flux[10];

    cx_mat soc_x, soc_y;

    soc_x << 0 << a*exp( ii*chi) << endr
          << a*exp(ii*chi) << 0 << endr;

    soc_y << 0 << a*ii*exp(ii*chi) << endr
          <<  ii*a*exp( ii*chi) << 0 << endr;



    A.x << sx[0]*x + sx[1]*y + sx[2]*z << 0 << endr
        << 0 << sx[0]*x + sx[1]*y + sx[2]*z << endr;

    A.y << sy[0]*x + sy[1]*y + sy[2]*z << 0 << endr
        << 0 << sy[0]*x + sy[1]*y + sy[2]*z << endr;

    A.z << sz[0]*x + sz[1]*y + sz[2]*z << 0 << endr
        << 0 << sz[0]*x + sz[1]*y + sz[2]*z << endr;

/*     A.y << sy*x << 0 << endr
        << 0 << sy*x << endr;

    A.z << sz << 0 << endr
        << 0 << sz << endr;
*/
    A.x += soc_x;
    A.y += soc_y;

    A.x_large = zeros<cx_mat>(8,8);
    A.y_large = zeros<cx_mat>(8,8);
    A.z_large = zeros<cx_mat>(8,8);

    A.x_large.submat(0,0,1,1) = A.x;
    A.x_large.submat(2,2,3,3) =  conj(A.x);
    A.x_large.submat(4,4,5,5) = A.x;
    A.x_large.submat(6,6,7,7) =  conj(A.x);

    A.y_large.submat(0,0,1,1) = A.y;
    A.y_large.submat(2,2,3,3) =  conj(A.y);
    A.y_large.submat(4,4,5,5) = A.y;
    A.y_large.submat(6,6,7,7) =  conj(A.y);

    A.z_large.submat(0,0,1,1) = A.z;
    A.z_large.submat(2,2,3,3) =  conj(A.z);
    A.z_large.submat(4,4,5,5) = A.z;
    A.z_large.submat(6,6,7,7) =  conj(A.z);

    return A;

}
```

B.7 magnetization.cpp

```
#include <vector>
#include <cmath>
#include <complex>
#include <iostream>
using namespace std;



vector<double> magnetization(vector<double> h, double x, double y){

    vector<double> H(3);
    complex<double> u;
    complex<double> ii(0,1);
    double lambda, xc, yc, hval;

    if (h.back() == 1.0){

        for (int i = 0; i < 3; i++)
            H[i] = h[i];

    } else if (h.back() == 2.0){

        hval = h[0];
        lambda = h[1];
        xc = h[2];
        yc = h[3];

        u = ii*lambda/(x   xc   ii*(y   yc));

       H[0] =  hval/(1. + abs(u)*abs(u))*2.*real(u);
       H[1] =  hval/(1. + abs(u)*abs(u))*2.*imag(u);
       H[2] =  hval/(1. + abs(u)*abs(u))*(1.   abs(u)*abs(u));



    } else if (h.back() == 3.0){

        H[0] = 0;
        H[1] = 0;
        H[2] = h[0]*tanh(h[1]*y);


    } else if (h.back() == 4.0){

        H[0] = h[0]*cos(h[1]*x);
        H[1] = h[0]*sin(h[1]*y);
        H[2] = 0;

    } else if (h.back() == 5.0){

        double r = sqrt(x*x + y*y);
        double alpha;

        if (r <= h[3]){

            alpha = 1.0;

        } else {

            alpha = pow(1.0/(1.0 + (r   h[3])),h[4]);

        }

        H[0] = h[0]*alpha;
        H[1] = h[1]*alpha;
        H[2] = h[2]*alpha;

    }
    return H;
}
```

B.8 Connection.cpp

```
#include "libmesh/libmesh.h"
#include "libmesh/mesh.h"
#include "libmesh/dof_map.h"
#include <utility>
#include "Connection.h"
#include "customstructs.h"

 using namespace std;
 using namespace libMesh;


 Connection::Connection(const MeshBase& mesh_in, BCinputV BC) :
     mesh(mesh_in) {

         filter_BC(BC);
         create_connection_map();

     }


//map<dof_id_type, dof_id_type> Connection::create_connection_map(){
void Connection::create_connection_map(){

    //map<dof_id_type, dof_id_type> connection_map;

    for (unsigned int c = 0; c < conIDs.size(); c++){
        map<pair<dof_id_type, unsigned char>, Point> in_centroids, out_centroids;

        MeshBase::const_element_iterator el = mesh.active_elements_begin();
        const MeshBase::const_element_iterator end_el = mesh.active_elements_end();


        int currentID = conIDs[c].first;
        int connectID = conIDs[c].second;

        for ( ; el != end_el; ++el){

            const Elem* elem = *el;

            for (unsigned int side=0; side < elem > n_sides(); side++)
                if (elem > neighbor(side) == NULL){
                    if(mesh.get_boundary_info().has_boundary_id(elem, side, connectID)){

                        UniquePtr<Elem> side_elem = elem > build_side(side);
                        out_centroids[make_pair(elem > id(), side)] = side_elem >
                            centroid();


                    } else if(mesh.get_boundary_info().has_boundary_id(elem, side,
                        currentID)){

                        UniquePtr<Elem> side_elem = elem > build_side(side);
                        in_centroids[make_pair(elem > id(), side)] = side_elem >
                            centroid();



                    }
                }

        }



        map<pair<dof_id_type, unsigned char>, Point>::iterator it = in_centroids.begin();
        map<pair<dof_id_type, unsigned char>, Point>::iterator it_end = in_centroids.end
            ();

        for ( ; it != it_end; ++it){
            Point in_centroid = it  > second;

            map<pair<dof_id_type, unsigned char>, Point>::iterator inner_it =
                out_centroids.begin();
            map<pair<dof_id_type, unsigned char>, Point>::iterator inner_it_end =
                out_centroids.end();

            Real min_distance = numeric_limits<Real>::max();

            for ( ; inner_it != inner_it_end; ++inner_it){

                Point out_centroid = inner_it  > second;

                Real distance = (out_centroid   in_centroid).size();

                if (distance < min_distance){

                    min_distance = distance;
                    connection_map[it  > first.first] = inner_it  > first.first;
                }

            }

        }
    }

}

dof_id_type Connection::get_neighbor_element_ID(dof_id_type elem){


    return connection_map[elem];
}

vector<int> Connection::get_connection_IDs(){

    return connection_ID;

}


void Connection::filter_BC(BCinputV BC){


    int NBC = BC.type.size();

    for (unsigned int i = 0; i < NBC; i++){

        if (BC.type[i] == "KL_Con"){

            int ID1 = BC.ID[i];
            int ID2 = BC.neighborID[i];

            conIDs.push_back(make_pair(ID1, ID2));

            connection_ID.push_back(ID1);

        }
    }


}
```

B.9 compute˙output.cpp

```
#include <armadillo>
#include "customstructs.h"
#include "MVConverter.h"
#include "compute_output.h"

using namespace std;
using namespace arma;
Gmat compute_g_matrices(cx_vec sol, cx_vec dsol_x, cx_vec dsol_y, cx_vec dsol_z, double E
    , double T){



    //Compute gamma matrices
    cx_mat gamma   = v_to_m(sol);
    cx_mat gamma_t = v_to_mt(sol);

    //Compute derivatives of gamma_matrices
    cx_mat dgamma_x   = v_to_m(dsol_x);
    cx_mat dgamma_y   = v_to_m(dsol_y);
    cx_mat dgamma_z   = v_to_m(dsol_z);
    cx_mat dgamma_t_x = v_to_mt(dsol_x);
    cx_mat dgamma_t_y = v_to_mt(dsol_y);
    cx_mat dgamma_t_z = v_to_mt(dsol_z);

    //Compute N and Nt
    mat I(2,2); I.eye();

    cx_mat Ninv = I   gamma*gamma_t;
    cx_mat N = inv(Ninv);
    cx_mat Ntinv = I   gamma_t*gamma;
    cx_mat Nt = inv(Ntinv);

    //Compute g, gt, f, ft and their derivatives
    cx_mat f = 2.*N*gamma;
    cx_mat ft = 2.*Nt*gamma_t;
    cx_mat df_x = 2.*N*(dgamma_x*gamma_t + gamma*dgamma_t_x)*N*gamma + 2.*N*dgamma_x;
    cx_mat df_y = 2.*N*(dgamma_y*gamma_t + gamma*dgamma_t_y)*N*gamma + 2.*N*dgamma_y;
    cx_mat df_z = 2.*N*(dgamma_z*gamma_t + gamma*dgamma_t_z)*N*gamma + 2.*N*dgamma_z;
    cx_mat dft_x = 2.*Nt*(dgamma_t_x*gamma + gamma_t*dgamma_x)*Nt*gamma_t + 2.*Nt*
        dgamma_t_x;
    cx_mat dft_y = 2.*Nt*(dgamma_t_y*gamma + gamma_t*dgamma_y)*Nt*gamma_t + 2.*Nt*
        dgamma_t_y;
    cx_mat dft_z = 2.*Nt*(dgamma_t_z*gamma + gamma_t*dgamma_z)*Nt*gamma_t + 2.*Nt*
        dgamma_t_z;


    cx_mat g = N*(I + gamma*gamma_t);
    cx_mat gt = Nt*(I + gamma_t*gamma);
    cx_mat dg_x = 2.*N*(dgamma_x*gamma_t + gamma*dgamma_t_x)*N;
    cx_mat dg_y = 2.*N*(dgamma_y*gamma_t + gamma*dgamma_t_y)*N;
    cx_mat dg_z = 2.*N*(dgamma_z*gamma_t + gamma*dgamma_t_z)*N;
    cx_mat dgt_x = 2.*Nt*(dgamma_t_x*gamma + gamma_t*dgamma_x)*Nt;
    cx_mat dgt_y = 2.*Nt*(dgamma_t_y*gamma + gamma_t*dgamma_y)*Nt;
    cx_mat dgt_z = 2.*Nt*(dgamma_t_z*gamma + gamma_t*dgamma_z)*Nt;

    cx_mat gR = zeros<cx_mat>(4,4);
    cx_mat gA = zeros<cx_mat>(4,4);
    cx_mat gK = zeros<cx_mat>(4,4);

    cx_mat gtot    = zeros<cx_mat>(8,8);
    cx_mat dgtot_x = zeros<cx_mat>(8,8);
    cx_mat dgtot_y = zeros<cx_mat>(8,8);
    cx_mat dgtot_z = zeros<cx_mat>(8,8);

    cx_mat dgR_x = zeros<cx_mat>(4,4);
    cx_mat dgR_y = zeros<cx_mat>(4,4);
    cx_mat dgR_z = zeros<cx_mat>(4,4);

    cx_mat dgA_x = zeros<cx_mat>(4,4);
    cx_mat dgA_y = zeros<cx_mat>(4,4);
cx_mat dgA_z = zeros<cx_mat>(4,4);

cx_mat dgK_x = zeros<cx_mat>(4,4);
cx_mat dgK_y = zeros<cx_mat>(4,4);
cx_mat dgK_z = zeros<cx_mat>(4,4);



mat rho3 = zeros<mat>(4,4);

rho3.submat(0,0,1,1) = I;
rho3.submat(2,2,3,3) =  I;

gR.submat(0,0,1,1) = g;
gR.submat(0,2,1,3) = f;
gR.submat(2,0,3,1) =  ft;
gR.submat(2,2,3,3) =  gt;

dgR_x.submat(0,0,1,1) = dg_x;
dgR_x.submat(0,2,1,3) = df_x;
dgR_x.submat(2,0,3,1) =  dft_x;
dgR_x.submat(2,2,3,3) =  dgt_x;

dgR_y.submat(0,0,1,1) = dg_y;
dgR_y.submat(0,2,1,3) = df_y;
dgR_y.submat(2,0,3,1) =  dft_y;
dgR_y.submat(2,2,3,3) =  dgt_y;

dgR_z.submat(0,0,1,1) = dg_z;
dgR_z.submat(0,2,1,3) = df_z;
dgR_z.submat(2,0,3,1) =  dft_z;
dgR_z.submat(2,2,3,3) =  dgt_z;

gA =  rho3*trans(gR)*rho3;

dgA_x =  rho3*trans(dgR_x)*rho3;
dgA_y =  rho3*trans(dgR_y)*rho3;
dgA_z =  rho3*trans(dgR_z)*rho3;

gK = (gR   gA)*tanh(1.76*E/(2.*T));

dgK_x = (dgR_x   dgA_x)*tanh(1.76*E/(2.*T));
dgK_y = (dgR_y   dgA_y)*tanh(1.76*E/(2.*T));
dgK_z = (dgR_z   dgA_z)*tanh(1.76*E/(2.*T));

gtot.submat(0,0,3,3) = gR;
gtot.submat(0,4,3,7) = gK;
gtot.submat(4,4,7,7) = gA;

dgtot_x.submat(0,0,3,3) = dgR_x;
dgtot_x.submat(0,4,3,7) = dgK_x;
dgtot_x.submat(4,4,7,7) = dgA_x;

dgtot_y.submat(0,0,3,3) = dgR_y;
dgtot_y.submat(0,4,3,7) = dgK_y;
dgtot_y.submat(4,4,7,7) = dgA_y;

dgtot_z.submat(0,0,3,3) = dgR_z;
dgtot_z.submat(0,4,3,7) = dgK_z;
dgtot_z.submat(4,4,7,7) = dgA_z;

Gmat G;

G.g = g;
G.f = f;
G.gt = gt;
G.ft = ft;

G.gR = gR;
G.dgR_x = dgR_x;
G.dgR_y = dgR_y;
G.dgR_z = dgR_z;
```

```
    G.gA = gA;
    G.dgA_x = dgA_x;
    G.dgA_y = dgA_y;
    G.dgA_z = dgA_z;

    G.gK = gK;
    G.dgK_x = dgK_x;
    G.dgK_y = dgK_y;
    G.dgK_z = dgK_z;

    G.gtot = gtot;
    G.dgtot_x = dgtot_x;
    G.dgtot_y = dgtot_y;
    G.dgtot_z = dgtot_z;

    return G;

}

cx_mat keldysh_gdg(cx_mat g, cx_mat dg, cx_mat A){

    cx_mat Atot, gdg(8,8), gdg_k(4,4);
    Atot = zeros<cx_mat>(8,8);

    Atot.submat(0,0,1,1) = A;
    Atot.submat(2,2,3,3) = A;
    Atot.submat(4,4,5,5) =  conj(A);
    Atot.submat(6,6,7,7) =  conj(A);


    complex<double> ii(0,1);

    gdg = g*dg   ii*g*(A*g   g*A);

    gdg_k = gdg(span(0,3),span(4,7));

    return gdg_k;

}

Gmat compute_g_negE(Gmat G){

    Gmat Gn;
    mat I(2,2); I.eye();
    mat rho1 = zeros<mat>(4,4);

    cx_mat gtot    = zeros<cx_mat>(8,8);
    cx_mat dgtot_x = zeros<cx_mat>(8,8);
    cx_mat dgtot_y = zeros<cx_mat>(8,8);
    cx_mat dgtot_z = zeros<cx_mat>(8,8);

    rho1.submat(0,2,1,3) = I;
    rho1.submat(2,0,3,1) = I;

    Gn.g = conj(G.gt);
    Gn.f = conj(G.ft);
    Gn.gt = conj(G.g);
    Gn.ft = conj(G.f);

    Gn.gR =  rho1*conj(G.gR)*rho1;
    Gn.gA =  rho1*conj(G.gA)*rho1;
    Gn.gK =  rho1*conj(G.gK)*rho1;

    Gn.dgR_x =  rho1*conj(G.dgR_x)*rho1;
    Gn.dgR_y =  rho1*conj(G.dgR_y)*rho1;
    Gn.dgR_z =  rho1*conj(G.dgR_z)*rho1;

    Gn.dgA_x =  rho1*conj(G.dgA_x)*rho1;
    Gn.dgA_y =  rho1*conj(G.dgA_y)*rho1;
    Gn.dgA_z =  rho1*conj(G.dgA_z)*rho1;

    Gn.dgK_x = rho1*conj(G.dgK_x)*rho1;
    Gn.dgK_y = rho1*conj(G.dgK_y)*rho1;
    Gn.dgK_z = rho1*conj(G.dgK_z)*rho1;

    Gn.gtot = gtot;
    Gn.dgtot_x = dgtot_x;
    Gn.dgtot_y = dgtot_y;
    Gn.dgtot_z = dgtot_z;

    Gn.gtot.submat(0,0,3,3) = Gn.gR;
    Gn.gtot.submat(0,4,3,7) = Gn.gK;
    Gn.gtot.submat(4,4,7,7) = Gn.gA;

    Gn.dgtot_x.submat(0,0,3,3) = Gn.dgR_x;
    Gn.dgtot_x.submat(0,4,3,7) = Gn.dgK_x;
    Gn.dgtot_x.submat(4,4,7,7) = Gn.dgA_x;

    Gn.dgtot_y.submat(0,0,3,3) = Gn.dgR_y;
    Gn.dgtot_y.submat(0,4,3,7) = Gn.dgK_y;
    Gn.dgtot_y.submat(4,4,7,7) = Gn.dgA_y;

    Gn.dgtot_z.submat(0,0,3,3) = Gn.dgR_z;
    Gn.dgtot_z.submat(0,4,3,7) = Gn.dgK_z;
    Gn.dgtot_z.submat(4,4,7,7) = Gn.dgA_z;

    return Gn;

}
```