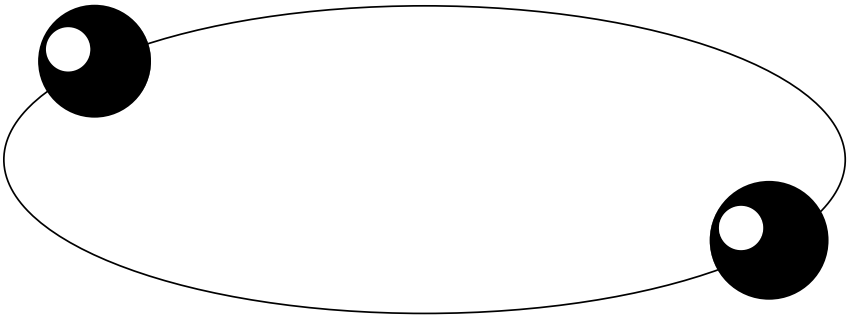
Science Institute
University of Iceland

June 2002

Electrons on a Ring



Sigríður Sif Gylfadóttir

Introduction

In this report the results of a numerical analysis of the properties of a system of electrons are reported. The electrons are confined to a ring and influenced by an external perturbation. We look at the system both with and without an interaction between the electrons. The number of electrons on the ring is varied between 1 and 6.

First we outline the physical description of the system. The numerical procedure used is then shortly described and following that the results are presented.

The main aim of the project was to become familiar with the quantum mechanics of a simple system and the numerical methods used to solve the equations describing this system.

Free electron confined to a ring

The Hamiltonian of a free electron is given by

$$H_0 = \frac{\mathbf{p}^2}{2m}$$

where m denotes the electron mass and $\mathbf{p}$ the momentum operator, which for an electron confined to a ring of radius R is

$$\mathbf{p} = -i\hbar\nabla = -\hat{a}_\theta \frac{i\hbar}{R}\frac{\partial}{\partial\theta}.$$

Inserting that into the Hamiltonian leads to

$$H_0 = -\frac{\hbar^2}{2mR^2}\frac{\partial^2}{\partial\theta^2}$$

which can be further simplified by defining the natural energy scale of the system as $E_* = \hbar^2/2mR^2$

$$H_0 = -E_*\frac{\partial^2}{\partial\theta^2}.$$

The Schrödinger equation

$$H_0\phi = \epsilon\phi \quad \leftrightarrow \quad -E_*\frac{\partial^2\phi}{\partial\theta^2} = \epsilon\phi$$

has the solution

$$\phi(\theta) = Ae^{ik\theta} + Be^{-ik\theta}$$

where $k = \sqrt{\epsilon/E_*}$. The boundary condition is $\phi(\theta + 2\pi) = \phi(\theta)$ giving

$$Ae^{ik(\theta+2\pi)} + Be^{-ik(\theta+2\pi)} = Ae^{ik\theta} + Be^{-ik\theta}$$

$$\Rightarrow \quad Ae^{ik\theta}e^{ik2\pi} + Be^{-ik\theta}e^{-ik2\pi} = Ae^{ik\theta} + Be^{-ik\theta}.$$

For this condition to hold we must have

$$e^{ik2\pi} = e^{-ik2\pi} = 1 \quad \Rightarrow \quad k = \alpha, \ \ \alpha = 0, \ \pm 1, \ \pm 2, \ \ldots$$

and we find that the energy eigenvalues are given by

$$\sqrt{\frac{\epsilon_\alpha}{E_*}} = \alpha \quad \Rightarrow \quad \epsilon_\alpha = E_*\alpha^2.$$

By normalizing the eigenfunctions we find the constants A and B

$$\int \phi^* \cdot \phi \, d\mathbf{r} = 1$$

$$\Rightarrow \quad \int_0^{2\pi} \left(Ae^{-i\alpha\theta} + Be^{-i\alpha\theta}\right)\left(Ae^{i\alpha\theta} + Be^{i\alpha\theta}\right) \; Rd\theta = 1$$

$$\Rightarrow \quad R\int_0^{2\pi} \left(A^2 + 2AB + B^2\right) \; d\theta = 1$$

$$\Rightarrow \quad \left(A^2 + 2AB + B^2\right) \; 2\pi R = 1$$

Since we don't have any other restriction on A and B we can choose $B = 0$. In that case

$$A = \frac{1}{\sqrt{2\pi R}}$$

and the solution becomes

$$\phi_\alpha\left(\theta\right) = \frac{e^{i\alpha\theta}}{\sqrt{2\pi R}}, \qquad \epsilon_\alpha = E_*\alpha^2, \qquad \alpha = 0,\ \pm 1,\ \pm 2,\ \ldots \tag{1}$$

The wave functions $|\alpha\rangle$ form a complete orthonormal basis, thus

$$\sum_\alpha |\alpha\rangle\,\langle\alpha| = 1 \quad \leftrightarrow \quad \sum_\alpha \phi_\alpha^*(\theta)\phi_\alpha(\theta') = \delta(\theta - \theta')$$

and

$$\langle\alpha|\ \alpha'\rangle = \delta_{\alpha\alpha'} \quad \leftrightarrow \quad \int \phi_\alpha^*\phi_{\alpha'} = \delta_{\alpha\alpha'}$$

Many-electron ring

When more than one electron are confined to the ring a mean field approach to the system supplies a term describing the interaction of the electrons (V_{H}) to the Hamiltonian (this term is described in the next section). We are interested in the effects on the system when an external potential $V_{\mathrm{ext}}(\mathbf{r})$ is applied, so the Hamiltonian becomes

$$H = \frac{\mathbf{p}^2}{2m} + V_{\mathrm{ext}}(\mathbf{r}) + V_{\mathrm{H}}(\mathbf{r})$$

giving the Schrödinger equation

$$\{H_0 + V_{\mathrm{ext}}(\mathbf{r}) + V_{\mathrm{H}}(\mathbf{r})\}\, \psi_\alpha(\mathbf{r}) = E_\alpha \psi_\alpha(\mathbf{r})$$

or

$$\{H_0 + V_{\mathrm{ext}}(\mathbf{r}) + V_{\mathrm{H}}(\mathbf{r})\}\, |\alpha) = E_\alpha\, |\alpha)\,.$$

We know the solutions to $H_0 \,|\alpha\rangle = \epsilon_\alpha \,|\alpha\rangle$ and the states $|\alpha\rangle$ form a complete orthonormal basis. Thus the eigenstates of our system can be expanded in this basis

$$|\alpha) = \sum_\gamma |\gamma\rangle \langle\gamma|\alpha\rangle = \sum_\gamma C_{\alpha\gamma}\,|\gamma\rangle \quad \leftrightarrow \quad \psi_\alpha(\mathbf{r}) = \sum_\gamma C_{\alpha\gamma}\phi_\gamma(\mathbf{r})$$

and the Schrödinger equation (2.1) transforms into

$$\sum_\gamma \{H_0 + V_{\mathrm{ext}} + V_{\mathrm{H}}\} C_{\alpha\gamma}\,|\gamma\rangle = E_\alpha \sum_\gamma C_{\alpha\gamma}\,|\gamma\rangle$$

or

$$\begin{aligned} &\sum_\gamma \{\langle\beta|\, H_0 \,|\gamma\rangle + \langle\beta|\,(V_{\mathrm{ext}} + V_{\mathrm{H}})\,|\gamma\rangle\} C_{\alpha\gamma} = E_\alpha C_{\alpha\beta} \\ &\Rightarrow \sum_\gamma \{\epsilon_\gamma \delta_{\gamma\beta} + \langle\beta|\,(V_{\mathrm{ext}} + V_{\mathrm{H}})\,|\gamma\rangle\}\, C_{\alpha\gamma} = E_\alpha C_{\alpha\beta} \qquad (2) \end{aligned}$$

where ϵ_γ is an eigenvalue of H_0. We have thus transformed the Schrödinger equation into an infinite set of algebraic eigenvalue equations without any approximations.

The Hartree approximation

The Coulomb interaction between the electrons is included using the Hartree approximation

$$V_{\mathrm{H}}(\mathbf{r}) = \frac{e^2}{4\pi\epsilon_0\epsilon_r}\int \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}d\mathbf{r}'.$$

where $n(\mathbf{r})$ is the *total* electron density. This means that each electron, as well as interacting with the other electrons present, interacts with itself. This of course is not the physical reality, and can be eliminated by using the Hartree-Fock approximation. However, the Hartree-Fock approximation is harder to implement numerically and for that reason we choose Hartree instead. The electron density is given by

$$n(\mathbf{r}) = \sum_\alpha |\psi_\alpha(\mathbf{r})|^2 f(E_\alpha - \mu) \quad \text{and} \quad \int n(\mathbf{r}')\ d\mathbf{r}' = Ne$$

Ne denoting the number of electrons. For thermal equilibrium the Fermi distribution function f is

$$f(E_\alpha - \mu) = \left[\exp\left(-\frac{E_\alpha - \mu}{k_B T}\right) + 1\right]^{-1}$$

where μ is the chemical potential. We choose to examine the system at a zero temperature, in which case $f(E_\alpha - \mu) \to \theta(\mu - E_\alpha)$, where $\theta(x)$ is the Heaviside function ($f(E_\alpha - \mu) = 1$ if $E_\alpha \leq \mu$). This is a reasonable choice since inside low dimensional semiconductors or metals the temperature in experiments is very nearly zero. This reduces the programming task but can result in computational instability. In our calculations we use the approximation

$$n(\mathbf{r}) = \sum_{\alpha=1}^{N} e\,|\psi_\alpha(\mathbf{r})|^2$$

i.e. we only sum over occupied states. Expanding the wave functions in the basis $|\gamma\rangle$ the density becomes

$$\begin{aligned} n(\theta) &= \sum_{\alpha=1}^{Ne}\left[\sum_\gamma C^*_{\alpha\gamma}\phi^*_\gamma \sum_\beta C_{\alpha\beta}\phi_\beta\right] \qquad \beta,\gamma = 0,\ \pm1,\ \pm2,\ \ldots \\ &= \sum_{\alpha=1}^{Ne}\sum_\gamma\sum_\beta C^*_{\alpha\gamma}C_{\alpha\beta}\left[\frac{e^{-i\gamma\theta}}{\sqrt{2\pi R}}\cdot\frac{e^{i\beta\theta}}{\sqrt{2\pi R}}\right] \qquad (3) \\ &= \frac{1}{2\pi R}\sum_{\alpha=1}^{Ne}\sum_\gamma\sum_\beta C^*_{\alpha\gamma}C_{\alpha\beta}e^{i(\beta-\gamma)\theta}. \end{aligned}$$

We select an interaction kernel of the type

$$V_{\text{int}} = \frac{1}{|\mathbf{r} - \mathbf{r}'|} \rightarrow \frac{V_i}{R\sqrt{\left[\sin\left(\frac{\theta-\theta'}{2}\right)\right]^2 + \mu_s^2}}$$

where V_i is the interaction strength (in units of E_*) and μ_s a dimensionless screening factor to eliminate the singularity at $\theta = \theta'$. This function describes an interaction between the electrons along a cord connecting them, not along the arch. Inserting this, the Hartree term becomes

$$\begin{aligned} V_{\text{H}}(\theta) &= \frac{V_i e^2}{4\pi\epsilon_0\epsilon_r}\int_0^{2\pi} \frac{n(\theta')}{R\sqrt{\sin^2\left(\frac{\theta-\theta'}{2}\right) + \mu_s^2}}\, R\, d\theta' \\ &= \frac{V_i e^2}{4\pi\epsilon_0\epsilon_r}\int_0^{2\pi} \frac{n(\theta')}{\sqrt{\sin^2\left(\frac{\theta-\theta'}{2}\right) + \mu_s^2}}\, d\theta'. \end{aligned} \tag{4}$$

Hereafter we shall use $n^*(\theta) = 2\pi R\, n(\theta)$, so

$$\begin{aligned} V_{\text{H}}(\theta) &= \frac{e^2}{4\pi\epsilon_0\epsilon_r}\frac{V_i}{2\pi R}\int_0^{2\pi} \frac{n^*(\theta')}{\sqrt{\sin^2\left(\frac{\theta-\theta'}{2}\right) + \mu_s^2}}\, d\theta' \\ &= \frac{R}{a_0}\frac{E_* V_i}{\pi}\int_0^{2\pi} \frac{n^*(\theta')}{\sqrt{\sin^2\left(\frac{\theta-\theta'}{2}\right) + \mu_s^2}}\, d\theta' \end{aligned} \tag{5}$$

where $a_0 = 4\pi\epsilon_0\epsilon_r\hbar^2/me^2$ denotes the Bohr radius.

The matrix element $\langle\beta\,|V_{\text{H}}|\,\gamma\rangle$

The contribution of the Hartree term to the matrix element is

$$\begin{aligned} \langle\beta\,|V_{\text{H}}|\,\gamma\rangle &= \int_0^{2\pi} \phi_\beta^*(\theta)\; V_{\text{H}}(\theta)\; \phi_\gamma(\theta)\; R d\theta \\ &= \int_0^{2\pi} \frac{e^{-i\beta\theta}}{\sqrt{2\pi R}}\left[\frac{R}{a_0}\frac{E_* V_i}{\pi}\int_0^{2\pi} \frac{n^*(\theta')}{\sqrt{\sin^2\left(\frac{\theta-\theta'}{2}\right)+\mu_s^2}}\, d\theta'\right]\frac{e^{i\gamma\theta}}{\sqrt{2\pi R}}\; R\, d\theta \\ &= \frac{E_*}{2\pi^2}\frac{R}{a_0}V_i\int_0^{2\pi} e^{i(\gamma-\beta)\theta}\left[\int_0^{2\pi} \frac{n^*(\theta')}{\sqrt{\sin^2\left(\frac{\theta-\theta'}{2}\right)+\mu_s^2}}\, d\theta'\right] d\theta. \end{aligned} \tag{6}$$

The external potential

The external potential is chosen to be

$$V_{\text{ext}}(\theta) = V_0 \left[\cos\left(10\ \theta\right)\right]^2 \quad , \quad \theta \in \left\{ \left[0, \tfrac{\pi}{20}\right], \left[\pi - \tfrac{\pi}{20}, \pi + \tfrac{\pi}{20}\right], \left[2\pi - \tfrac{\pi}{20}, 2\pi\right] \right\}$$

and shown in Fig. 1. The contribution of the external potential to the matrix element is given by

$$\langle \beta \left| V_{\text{ext}} \right| \gamma \rangle = \int_0^{2\pi} \phi_\beta^*(\theta)\ V_{\text{ext}}\ \phi_\gamma(\theta)$$

and results (after tedious calculation) in

$$\langle \beta \left| V_{\text{ext}} \right| \gamma \rangle = \begin{cases} V_0/2 & \beta - \alpha = 0 \\ V_0/4 & \beta - \alpha \in \{-20,\ 20\} \\ \frac{5V_0\ \sin\left([\gamma-\beta]\frac{\pi}{20}\right)\left[\left(-\gamma^2+2\gamma\beta-\beta^2-400\right)\left(1+(-1)^{\gamma-\beta}\right)\right]}{2\pi(\gamma-\beta)(\gamma-\beta-20)(\gamma-\beta+20)} & \text{otherwise.} \end{cases} \tag{7}$$

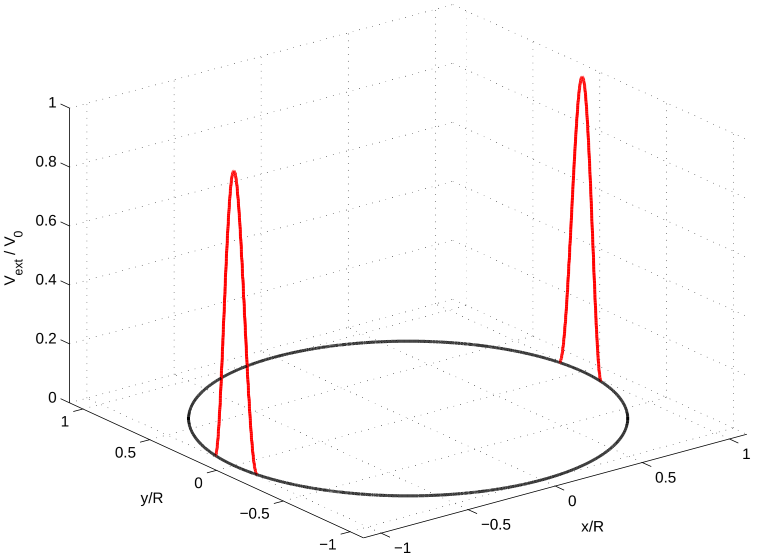


Figure 1: The external potential.

Numerical implementation

To examine the system we need to implement the physical description numerically. We choose to concentrate on the energy eigenvalues as well as the electron density along the ring. In the preceding sections we have seen how this many body problem can successfully be transformed into a relatively simple eigenvalue problem without serious approximations. To solve the system of equations we are forced to truncate the basis of single electron eigenfunctions. The cutoff value depends on the number of electrons on the ring since the accuracy of solutions decreases as we approach the endpoints of the basis. The basis cutoff value is chosen to be $N = 20$ giving a good accuracy when looking at the system with a maximum of 6 electrons. The numerical procedure is then to fill the matrix according to equation (2) using (1), (6), and (7) and solve $A\mathbf{C} = E\mathbf{C}$ where A is given by

$$\begin{bmatrix} \epsilon_{-N}+\langle -N|V_{\mathrm{ext}}+V_{\mathrm{H}}|-N\rangle & \dots & \langle -N|V_{\mathrm{ext}}+V_{\mathrm{H}}|0\rangle & \dots & \langle -N|V_{\mathrm{ext}}+V_{\mathrm{H}}|N\rangle \\ \vdots\ddots & & \vdots & & \vdots \\ \langle 0|V_{\mathrm{ext}}+V_{\mathrm{H}}|-N\rangle & \dots & \epsilon_0+\langle 0|V_{\mathrm{ext}}+V_{\mathrm{H}}|0\rangle & \dots & \langle 0|V_{\mathrm{ext}}+V_{\mathrm{H}}|N\rangle \\ \vdots & & \ddots & & \vdots \\ \langle N|V_{\mathrm{ext}}+V_{\mathrm{H}}|-N\rangle & \dots & \langle N|V_{\mathrm{ext}}+V_{\mathrm{H}}|0\rangle & \dots & \epsilon_N+\langle N|V_{\mathrm{ext}}+V_{\mathrm{H}}|N\rangle \end{bmatrix} \tag{8}$$

The columns of $\mathbf{C}$ give the weight coefficients for the wavefunctions in the basis.

The Hartree term is by far the most computationally expensive term to evaluate as well as being unstable. The solution is found self-consistently by

1. using the initial guess $n_0^*(\theta) = Ne$, i.e. a constant density along the ring.

2. Calculating the matrix elements.

3. Finding the matrix energy eigenvalues (E_j) and eigenvectors $(C_{\alpha j})$.

4. Calculating $n^*(\theta)$ using (3).

5. Mixing the new and old densities according to

   $$n_1^*(\theta) = x \cdot n_0(\theta) + (1-x) \cdot n^*(\theta).$$

   Since the system is quite unstable we use $x = 0.999$.

6. Repeating steps 2-5 until $\sum_{i=0}^{2N} \sqrt{(E_{i+1} - E_i)^2} \leq \delta$ where δ is a tolerance value chosen to be $\delta = 0.5 \cdot 10^{-4}$.

The terms in eq. (2) are scaled with E_*. Lengths do not occur in any equations due to the fact that we work with $n^*(\theta) = 2\pi R\ n(\theta)$, except in (6) where it is scaled with the Bohr radius a_0. This ratio is chosen as $R/a_0 = 5$. The screening factor is given the value $\mu_s = 0.5$. The programs that solve the system equations are written in Fortran90/95.

Results

No interaction

Starting with a computationally efficient case, we look at the system with no interaction between the electrons. The energy spectrum is shown in Fig. 1. As expected the external perturbation lifts the degeneracy of the states. For small perturbation strength (Fig. 2a) the energy increases linearly as the perturbation strength grows. The lower branches of the split states grow linearly over the whole range (Fig. 2b). The upper branches on the other hand are linear in the beginning, but the growth slows down as the perturbation strength increases, avoiding the lower branch above.

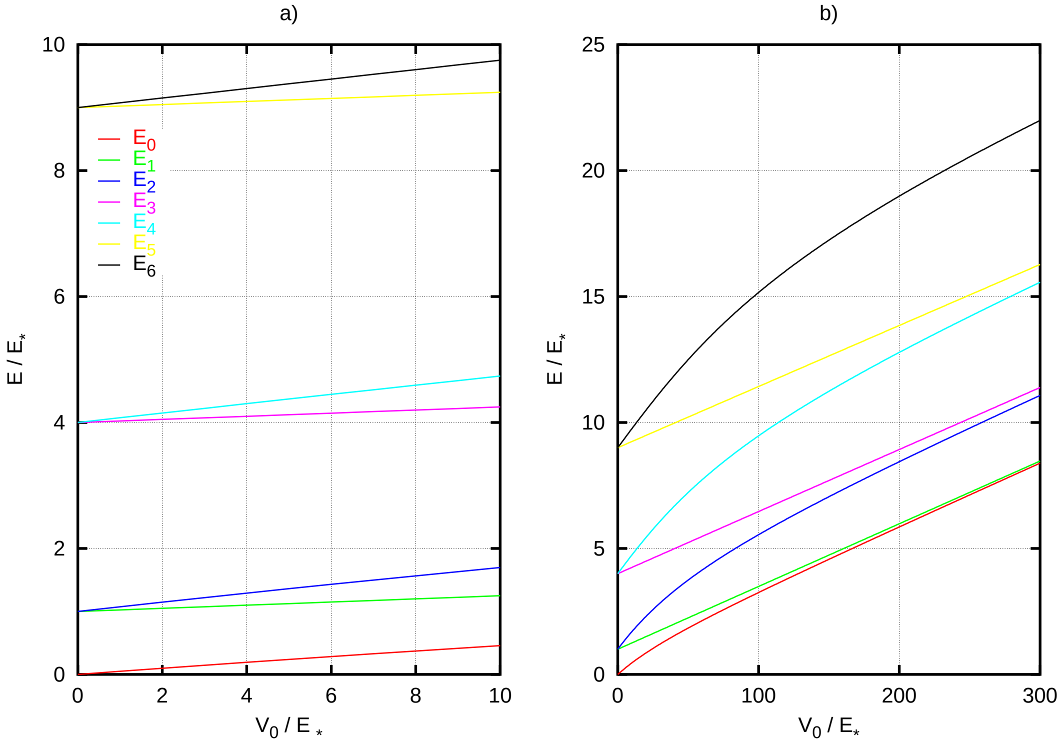


Figure 2: Energy spectrum. a) States $E_0 - E_6$, low values of V_0/E_*. b) States $E_0 - E_6$, high values of V_0/E_*.

In agreement with Fig. 2, Figs. 3a and c show that the energy difference of degenerate states grows linearly with increasing (low) perturbation strength. It is interesting to note (Fig. 3c) that up to $\Delta E_{11,12}$ the lifting increases for higher energy states. However, as we move upwards from $\Delta E_{11,12}$ the situation reverses and higher energy states exhibit a lower lifting of degeneracy.

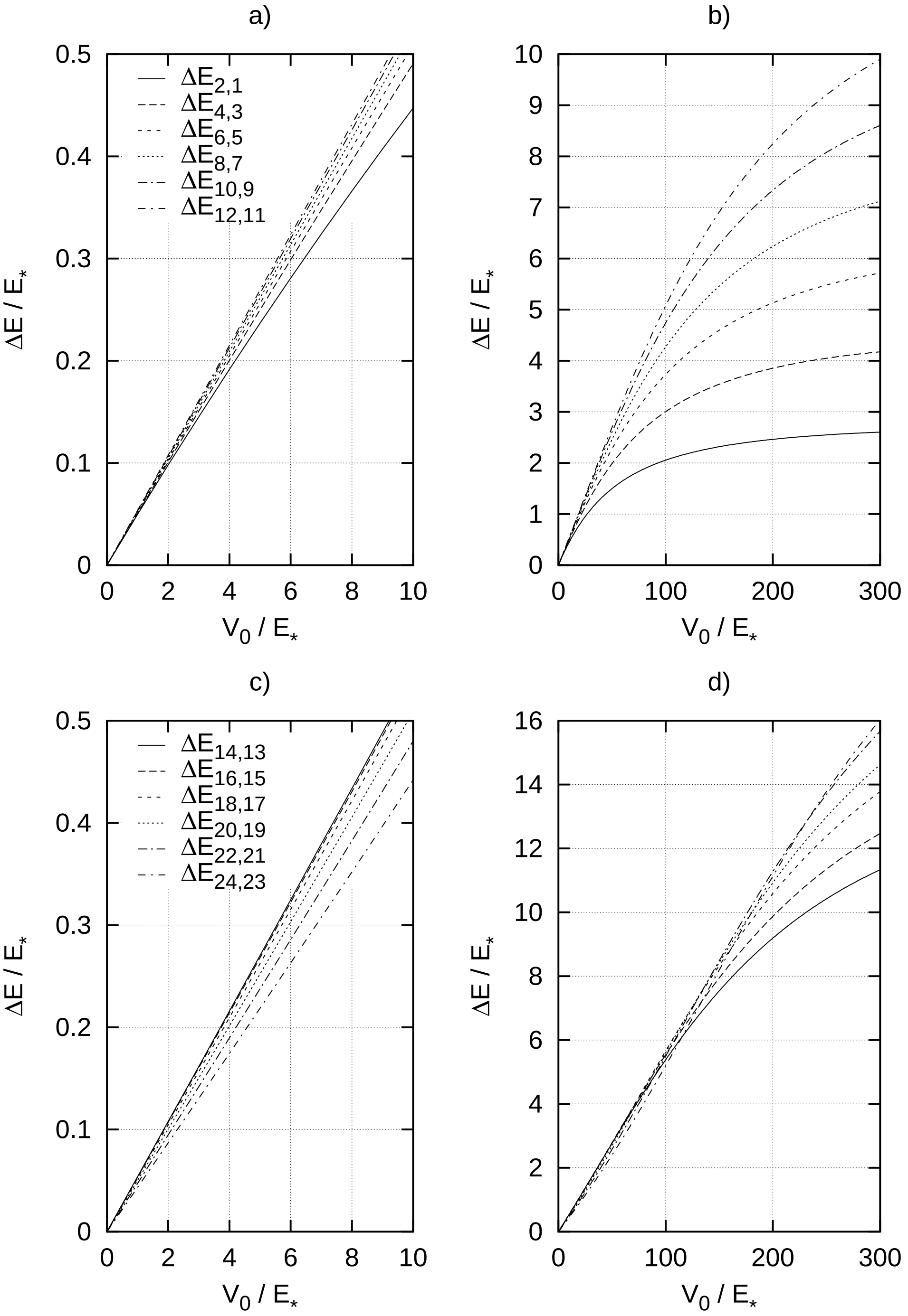


Figure 3: Energy difference of degenerate states. a) States up to $\Delta E_{12,11}$, low values of V_0/E_*. b) States up to $\Delta E_{12,11}$, high values of V_0/E_*. c) States from $\Delta E_{14,13}$, low values of V_0/E_*. d) States from $\Delta E_{14,13}$, high values of V_0/E_*.

The calculations were repeated for a larger basis of eigenfunctions to verify this being true, and not an artifact of using a reduced basis. Obviously the external perturbation has less effect on higher energy states. With increasing strength (Fig. 3b and d) the rate of change of the difference decreases. This is because the growth of the upper branches of split states slows down while the lower branches continue to grow (almost) linearly. In Fig. 3d we see that the higher energy states gradually catch up with the lower states and by $V_0/E_* = 250$ the behaviour is the same as in Fig. 3b. Here the perturbation is strong enough to effect the higher states. Fig. 4 shows the whole range of differences to display better the change that occurs at $\Delta E_{12,11}$. Differences of split states above that are depicted by points.

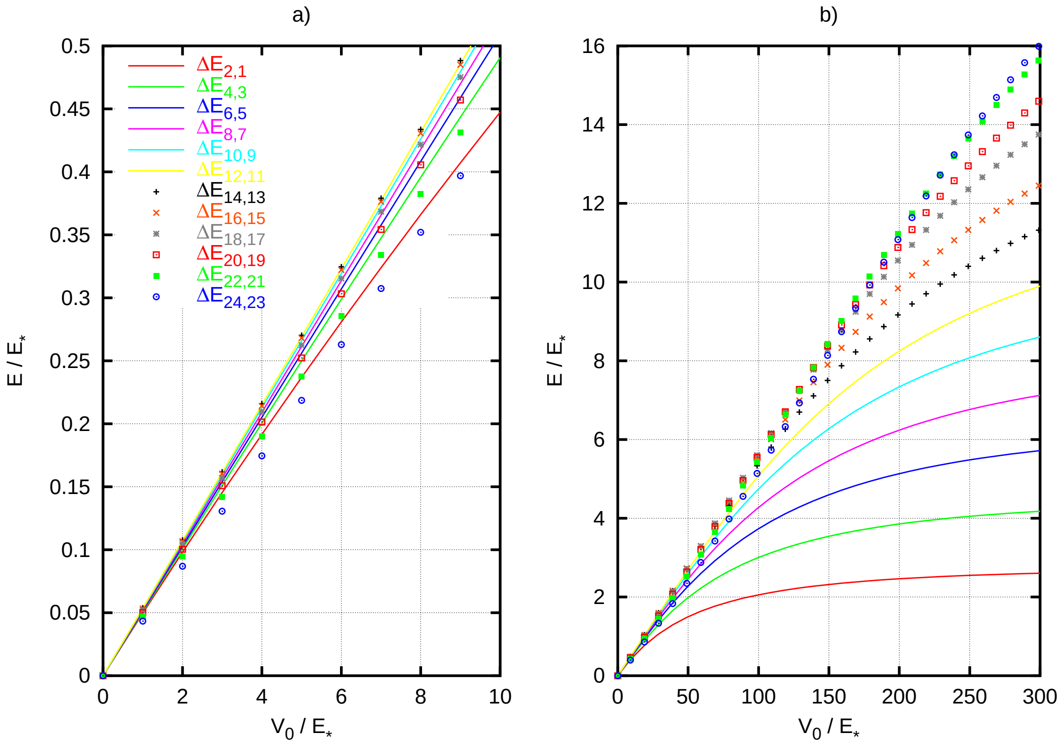
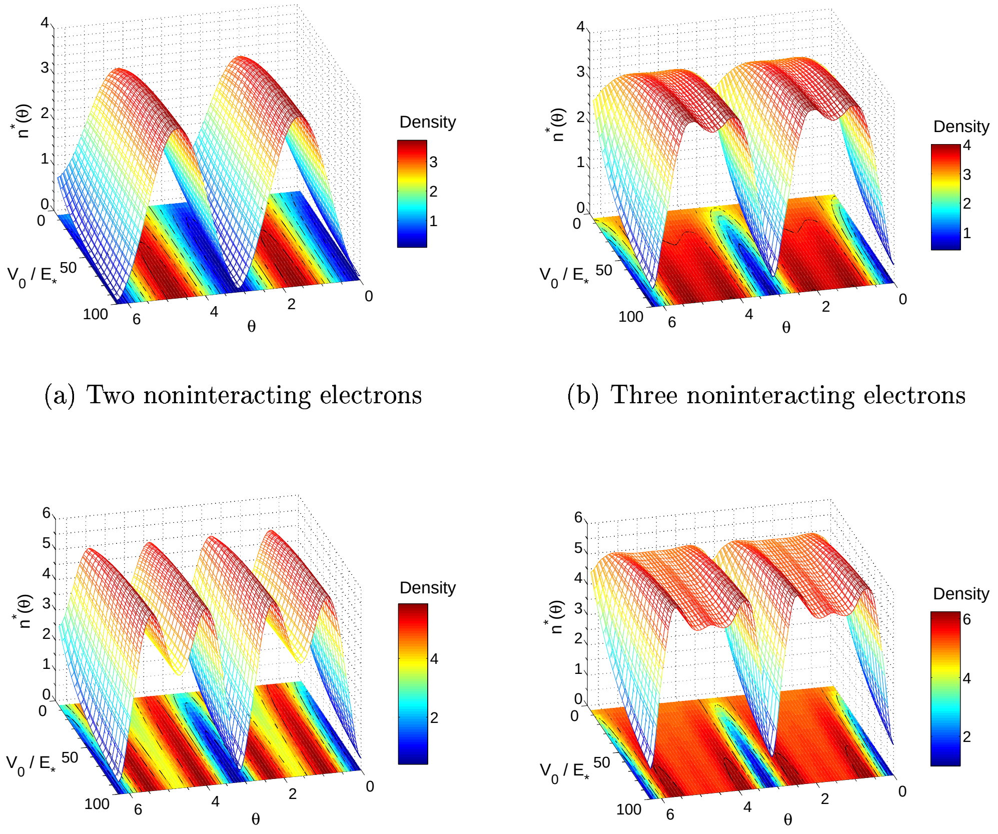


Figure 4: Energy difference of degenerate states. Differences from $E_{14,13}$ drawn with points. a) Low values of V_0/E_*. b) High values of V_0/E_*

The electron density $n^*(\theta)$ along the ring is shown as a function of V_0/E_* in Fig. 5 for 2 - 5 electrons confined to the ring. As expected the electrons avoid the maxima of the external potential increasingly as the strength grows. The electrons strive to arrange symmetrically between the perturbation maxima, which for an even number of electrons is easily achieved. When the number of electrons is odd they encounter great difficulties in distributing evenly around the ring. Instead of symmetric wavelike configuration the density flattens out.

In Fig. 6 the density is shown as a function of angle for selected values of V_0/E_* and 1-6 electrons. As the perturbation strength increases the minima in electron density are gradually drawn further down. The peaks become more prominent as well as confined to a shorter interval, so the electrons pack more closely.



(a) Two noninteracting electrons

(b) Three noninteracting electrons

(c) Four noninteracting electrons

(d) Five noninteracting electrons

Figure 5: Density as a function of the external perturbation strength for noninteracting electrons.

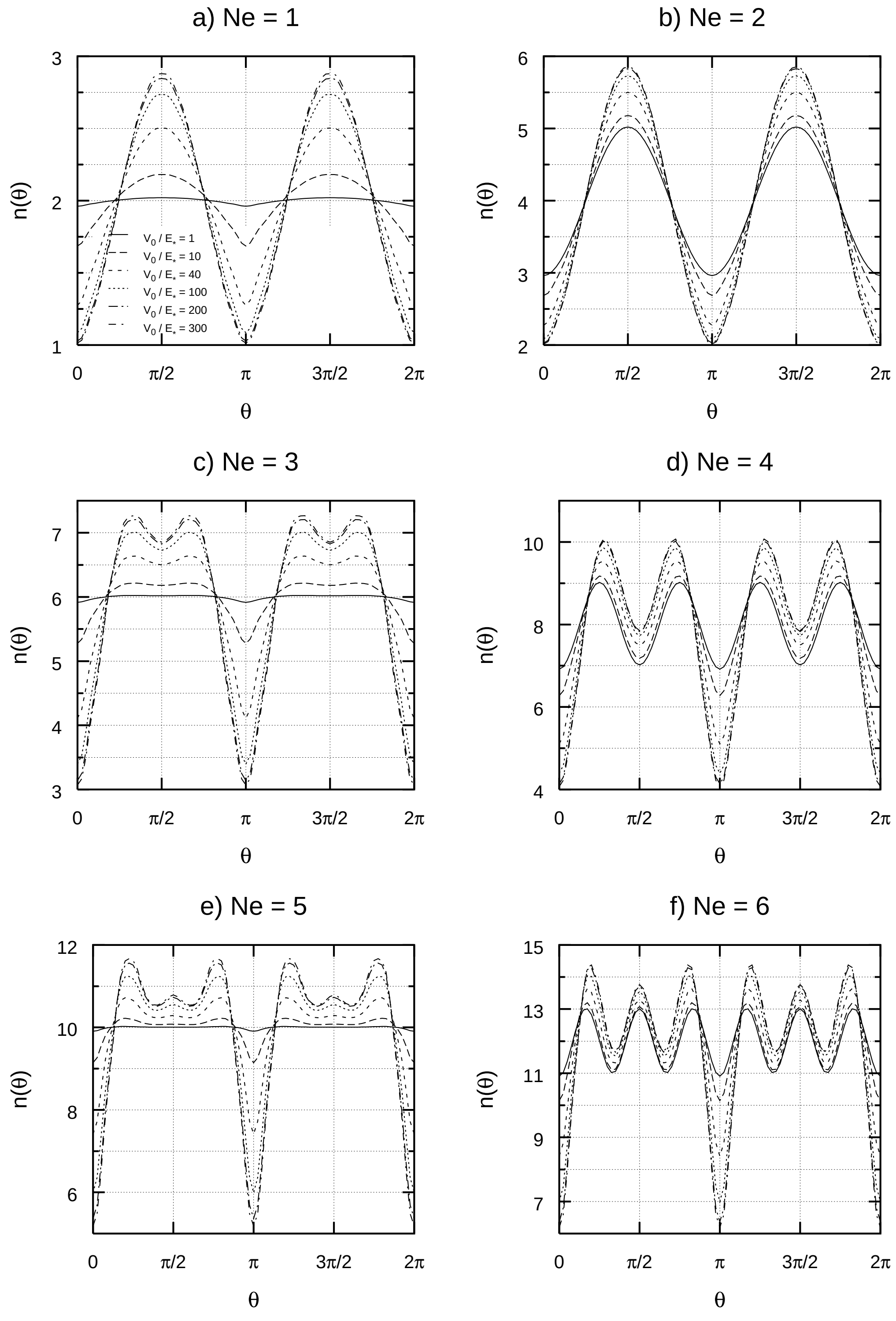


Figure 6: Electron density $n^*(\theta)$ for 1 - 6 noninteracting electrons and selected values of V_0/E_*.

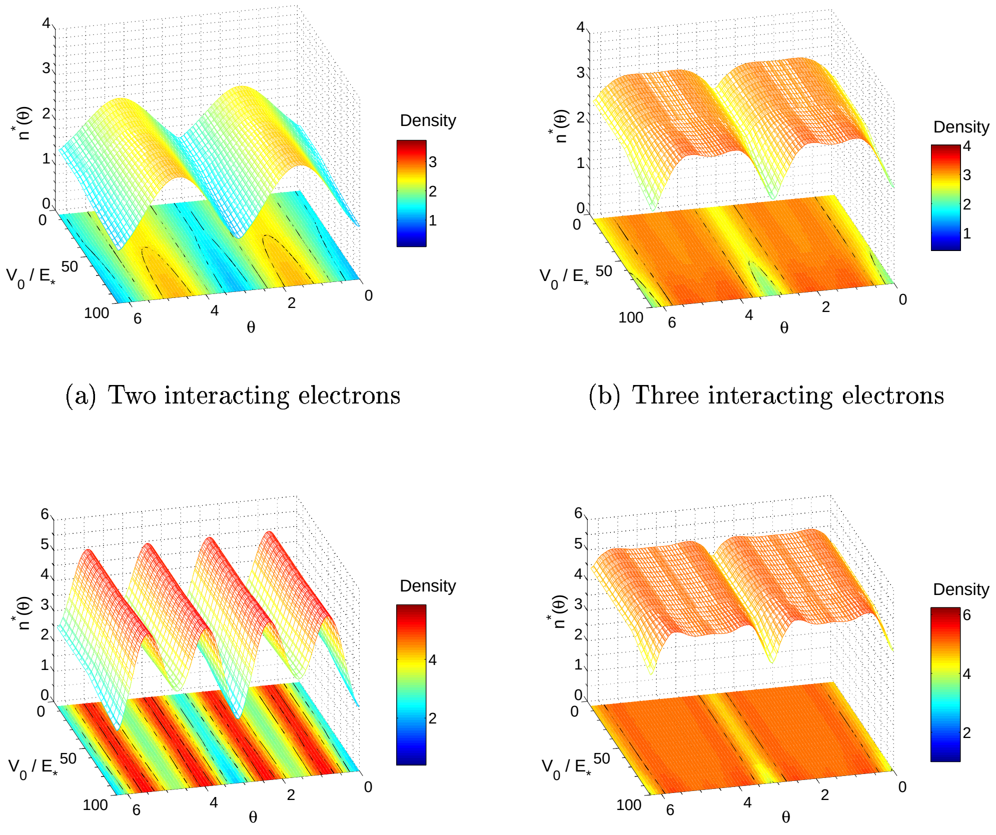
Interaction

For interacting electrons on a ring we look at two cases:

1. A constant interaction strength ($V_i = 0.2625$) and varying perburbation strength ($V_0 = 5,\ 10,\ \ldots,\ 100$).
2. A constant perturbation strength ($V_0 = 10$) and varying interaction strength ($V_i = 0,\ 2,\ \ldots,\ 50$).

The reason for the choice of the low value of V_i in case 1 is that the two terms, perturbation and interaction, are of comparable sizes and compete for influence on the system. In case 2 the interaction dominates, except for the lowest values of V_i.

Fig. 7 shows the density along the ring as a function of the external



(a) Two interacting electrons

(b) Three interacting electrons

(c) Four interacting electrons

(d) Five interacting electrons

Figure 7: Density as a function of the external perturbation strength for interacting electrons and $V_i = 0.2625$.

perturbation strength for 2-5 electrons. To see more clearly the effect of the interaction the color scale is the same as in Fig. 5. Clearly the interaction flattens out the density, the electrons avoid each other and move closer to the potential maxima. As the potential strength grows the electrons gradually move away from the maxima. It must be kept in mind that in the Hartree approximation the electron "sees" the total charge density, including itself. Thus the density in the following figures is more flattened out than in reality. The energy spectrum for 3 electrons is shown in Fig. 8. Comparing it to Fig. 2b, we see that the structure remains the same, the energy just shifts upwards as a result of the interaction. Although not shown, the results for 2, 4 and 5 electrons have the same form, the only difference being the magnitude of the energy.

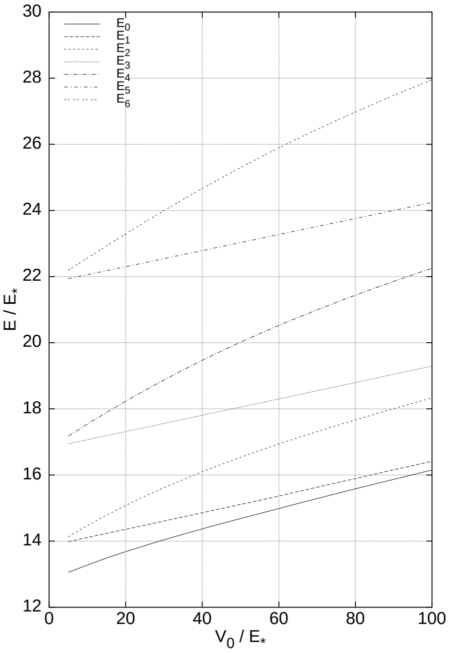
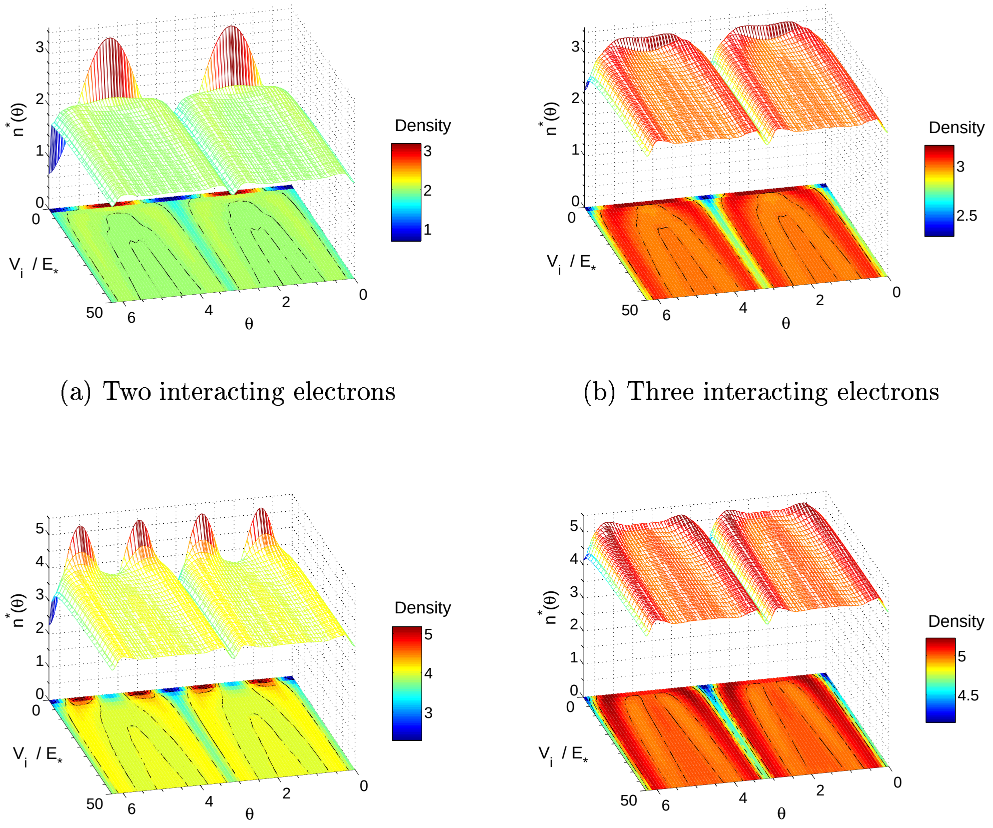


Figure 8: Energy spectrum as a function of the external perturbation for 3 interacting electrons with $V_i = 0.2625$.

The density distribution for case 2 is shown in Fig. 9. In this case the interaction dominates the system, as can clearly be seen in the figure especially for an even number of electrons. The wavelike structure almost vanishes, even for the lowest strength of interaction. As the interaction strength increases, the peaks in density move closer to the potential maxima, and the density distribution looks similar for the different numbers of electrons on the ring.



(a) Two interacting electrons

(b) Three interacting electrons

(c) Four interacting electrons

(d) Five interacting electrons

Figure 9: Density as a function of the interaction strength for 2-5 interacting electrons and $V_0 = 10$.

Summary

This project was an opportunity for me to do two things. First, as a student with but the most basic knowledge of quantum mechanics, the project gave a valuable insight into the subject giving a basis to examine more complex systems. Second, it gave me a chance to move on from Fortran77 to programming in Fortran90/95. I have included the programs written for others who might share my interest and find it valuable. The project ends here but could easily be extended in many directions, for example including a magnetic field or using the more complicated Hartree-Fock approximation for the interaction.

Programs

main: the main program

```
PROGRAM main

INTEGER  :: ON
EXTERNAL :: el_ring_V_ext, el_ring_V_int

ON = 1

SELECT CASE (ON)

   CASE (1)

      CALL el_ring_V_ext()

   CASE (2)

      CALL el_ring_V_int()

   CASE DEFAULT

      PRINT*, 'Error in main'
      STOP

END SELECT

STOP
END PROGRAM main
```

el_ring_V_ext: the subroutine for case 1

```
   SUBROUTINE el_ring_V_ext()

   USE const
   USE tools
   USE ComDensit
   USE angleVs
   USE densVs

   IMPLICIT NONE

! .. VARIABLES ..

   INTEGER, PARAMETER :: fid1 = 25, fid2 = 26, fid3 = 27
   INTEGER            :: i, j, k, INFO, show
   REAL(KIND=8)       :: V_ext, V_int, RMS, RMSOCC, tmp, DensInt

! .. ARRAYS ..

   REAL(KIND=8),    DIMENSION(lda)     :: EIG, EIGOld, EIGN, temp
   REAL(KIND=8),    DIMENSION(3*lda)   :: RWORK
   REAL(KIND=8),    DIMENSION(4*NOP)   :: DeltaDens2, tmp1
   REAL(KIND=8),    DIMENSION(4*NOP)   :: ReDensVOld, ReDensVN, ImDensVOld
   COMPLEX(KIND=8), DIMENSION(2*lda)   :: WORK
   COMPLEX(KIND=8), DIMENSION(lda,lda) :: FYLKI

! .. EXTERNAL ROUTINES ..

   EXTERNAL :: makearr, GausstOs, F02HAF

   OPEN(fid1,file='density.dat',status='new')
   OPEN(fid2,file='energy.dat',status='new')
   OPEN(fid3,file='error.dat',status='new')

   V_int  = 0.2625
   V_ext  = 5.0
   DoDens = 0

!--------------------------------------------------|
! MAKE AN ARRAY OF THETA VALUES WHERE THE DENSITY  |
! AND INTEGRAL OVER THETA AND THETAt' IS EVALUATED. |
!--------------------------------------------------|

   th_inc = 2.D0 * PI / DBLE(NOP)

   CALL GausstOs(Angle,GausW,NOP,4*NOP,th_inc,PI)

!----------------------------------|
! BEGIN BY GUESSING : n*(theta)=Ne |
!----------------------------------|

   CALL makearr(V_ext,V_int,FYLKI,1)

   INFO = 0

   CALL F02HAF('V','U',lda,FYLKI,lda,EIG,RWORK,WORK,2*lda,INFO)

!-------------------|
! CALCULATE DENSITY |
!-------------------|
   DO i = 1 , 4*NOP

      ReDensV(i) = densityf(Angle(i),DBLE(FYLKI(:,:Ne)),AIMAG(FYLKI(:,:Ne)))

   ENDDO

   EIGOld     = EIG
   ReDensVOld = ReDensV

   DoDens = 1 !NOW USE CALC. DENSITY INSTEAD OF n = Ne

!---------------------------------------|
! LOOP THROUGH A NUMBER OF V_EXT VALUES |
!---------------------------------------|

   DO k = 1 , 20

      PRINT*, k

      V_ext = DBLE(k) * 5.0

      RMS    = 1.

      show = 0

!-----------------------------------------|
! FIRST FIND THE NO-INTERACTION SOLUTION  |
!-----------------------------------------|

      CALL makearr(V_ext,V_int,FYLKI,0)

      INFO = 0

      CALL F02HAF('V','U',lda,FYLKI,lda,EIGN,RWORK,WORK,2*lda,INFO)

      DO i = 1 , 4*NOP

         ReDensVN(i) = densityf(Angle(i),DBLE(FYLKI(:,:Ne)),AIMAG(FYLKI(:,:Ne)))

      ENDDO

!------------------------------------|
! NOW FIND THE INTERACTION SOLUTION. |
!------------------------------------|

! LOOP UNTIL THE ENERGY RMS DIFF IS SMALL

      DO WHILE (RMS >= 0.5D-4)

         show = show + 1

         IF (show>1000) EXIT

         FYLKI = 0.

         !FILL MATRIX
         CALL makearr(V_ext,V_int,FYLKI,1)

         INFO = 0

         !SOLVE EIGENVALUE EQ.
         CALL F02HAF('V','U',lda,FYLKI,lda,EIG,RWORK,WORK,2*lda,INFO)

         !PRINT*, SUM(ABS(FYLKI(:,:Ne))**2)
   !CALCULATE NEW DENSITY
   DO i = 1 , 4*NOP

      ReDensV(i)  = densityf(Angle(i),DBLE(FYLKI(:,:Ne)),AIMAG(FYLKI(:,:Ne)))

   ENDDO

   !ENERGY RMS DIFF
   RMS = SQRT(SUM((EIG-EIGOld)*(EIG-EIGOld))/SIZE(EIG))

   !ENERGY RMS DIFF ONLY USING OCCUPIED STATES
   RMSOCC = SQRT(SUM((EIG(:Ne)-EIGOld(:Ne))*(EIG(:Ne)-EIGOld(:Ne)))/Ne)

   !----------------------------------------------------------|
   ! CALCULATE INT _0 ^2PI [ (new density - old density)^2 ]  |
   !----------------------------------------------------------|

   DeltaDens2 = (ReDensV - ReDensVOld)*(ReDensV - ReDensVOld)

   tmp = 0.D0

   DO i = 1,4*NOP

      tmp = tmp + DeltaDens2(i) * GausW(i)

   ENDDO

   DensInt = tmp * (th_inc * 0.5)

   !WRITE ERROR VALUES TO FILE
   WRITE(fid3,FMT='(3(F16.7))') RMS, RMSOCC, SQRT(DensInt)/Ne

   !---------------------------------------------------------|
   ! NOW MIX OLD AND NEW DENSITY -> ONLY USE 1-xtake OF NEW  |
   !---------------------------------------------------------|

   tmp1 = (xtake * ReDensVOld) + ((1.-xtake) * ReDensV)

   ReDensVOld = tmp1
   ReDensV    = tmp1

   EIGOld = EIG

ENDDO

!------------------------------------------------------------|
! NOW THE SOLUTIONS HAVE CONVERGED AND WE'RE PRETTY SURE WE  |
! HAVE THE RIGHT ANSWER. WRITE THE RESULTS TO OUTPUT FILE    |
! AND LET'S TRY A NEW V_0 VALUE.                             |
!------------------------------------------------------------|

DO i = 1 , 4*NOP

   WRITE(fid1,FMT='(4(F16.7))') Angle(i), V_ext, ReDensVN(i), ReDensV(i)

ENDDO

DO i = 1 , 3*Ne

   WRITE(fid2,FMT='(2(F16.7))')   EIG(i), EIGN(i)

ENDDO

WRITE(fid3,FMT='(3(F16.7))') 0, 0, 0  !(FOR DISTINCTION)
```

```
ENDDO

CLOSE(fid1)
CLOSE(fid2)
CLOSE(fid3)

RETURN
END SUBROUTINE el_ring_V_ext
```

el_ring_V_int: the subroutine for case 2

```
   SUBROUTINE el_ring_V_int()

   USE const
   USE tools
   USE ComDensit
   USE angleVs
   USE densVs

   IMPLICIT NONE

!     .. breytur

   INTEGER, PARAMETER     :: fid1 = 25, fid2 = 26, fid3 = 27
   INTEGER                :: i, j, k, INFO, show
   INTEGER, DIMENSION(20) :: NOCONV
   REAL(KIND=8)           :: V_ext, V_int, RMS, RMSOCC, tmp, DensInt

!     .. vigrar og fylki ..

   REAL(KIND=8),    DIMENSION(lda)     :: EIG, EIGOld, EIGN, temp
   REAL(KIND=8),    DIMENSION(3*lda)   :: RWORK
   REAL(KIND=8),    DIMENSION(4*NOP)   :: DeltaDens2, tmp1
   REAL(KIND=8),    DIMENSION(4*NOP)   :: ReDensVOld, ReDensVN, ImDensVOld
   COMPLEX(KIND=8), DIMENSION(2*lda)   :: WORK
   COMPLEX(KIND=8), DIMENSION(lda,lda) :: FYLKI

!     .. external ..

   EXTERNAL :: makearr, Gausst0s, F02HAF

   OPEN(fid1,file='density.dat',status='new')
   OPEN(fid2,file='energy.dat',status='new')
   OPEN(fid3,file='error.dat',status='new')

   V_int  = 0.0
   V_ext  = 10.0
   DoDens = 0

!--------------------------------------------------|
! MAKE AN ARRAY OF THETA VALUES WHERE THE DENSITY  |
! AND INTEGRAL OVER THETA AND THETAť IS EVALUATED. |
!--------------------------------------------------|

   th_inc = 2.D0 * PI / DBLE(NOP)

   CALL Gausst0s(Angle,GausW,NOP,4*NOP,th_inc,PI)

!----------------------------------|
! BEGIN BY GUESSING : n*(theta)=Ne |
!----------------------------------|

   CALL makearr(V_ext,V_int,FYLKI,1)

   INFO = 0

   CALL F02HAF('V','U',lda,FYLKI,lda,EIG,RWORK,WORK,2*lda,INFO)

   PRINT*, SUM(ABS(FYLKI(:,:Ne))**2)
   print*, EIG(1:3*Ne)

!-------------------|
! CALCULATE DENSITY |
!-------------------|

   DO i = 1 , 4*NOP

      ReDensV(i) = densityf(Angle(i),DBLE(FYLKI(:,:Ne)),AIMAG(FYLKI(:,:Ne)))

   ENDDO

   EIGOld     = EIG
   ReDensVOld = ReDensV

   DoDens = 1 !NOW USE CALC. DENSITY INSTEAD OF n=Ne

!---------------------------------------|
! LOOP THROUGH A NUMBER OF V_INT VALUES |
!---------------------------------------|

   DO k = 1 , 26

      PRINT*, k

      V_int = DBLE(k-1) * 2.0

      RMS    = 1.

      show = 0

!-----------------------------------------|
! LOOP UNTIL THE ENERGY RMS DIFF IS SMALL |
!-----------------------------------------|

      DO WHILE (RMS>0.5D-5)

         show = show + 1

         IF (show>1000) EXIT

         FYLKI = 0.

         !FILL MATRIX
         CALL makearr(V_ext,V_int,FYLKI,1)

         INFO = 0

         !SOLVE EIGENVALUE EQ.
         CALL F02HAF('V','U',lda,FYLKI,lda,EIG,RWORK,WORK,2*lda,INFO)

         !CALCULATE NEW DENSITY
         DO i = 1 , 4*NOP

            ReDensV(i)  = densityf(Angle(i),DBLE(FYLKI(:,:Ne)),AIMAG(FYLKI(:,:Ne)))

         ENDDO

         !ENERGY RMS DIFF
         RMS = SQRT(SUM((EIG-EIGOld)*(EIG-EIGOld))/SIZE(EIG))

         !ENERGY RMS DIFF ONLY USING OCCUPIED STATES
         RMSOCC = SQRT(SUM((EIG(:Ne)-EIGOld(:Ne))*(EIG(:Ne)-EIGOld(:Ne)))/Ne)

         !----------------------------------------------------------|
      ! CALCULATE INT _0 ^2PI [ (new density - old density)^2 ]  |
      !----------------------------------------------------------|

      DeltaDens2 = (ReDensV - ReDensVOld)*(ReDensV - ReDensVOld)

      tmp = 0.D0

      DO i = 1,4*NOP

         tmp = tmp + DeltaDens2(i) * GausW(i)

      ENDDO

      DensInt = tmp * (th_inc * 0.5)

      WRITE(fid3,FMT='(3(F16.7))') RMS, RMSOCC, SQRT(DensInt)/Ne

      !---------------------------------------------------------|
      ! NOW MIX OLD AND NEW DENSITY -> ONLY USE 1-xtake OF NEW  |
      !---------------------------------------------------------|

      tmp1 = (xtake * ReDensVOld) + ((1.-xtake) * ReDensV)

      ReDensVOld = tmp1
      ReDensV    = tmp1

      EIGOld = EIG

   ENDDO

   !------------------------------------------------------------|
   ! NOW THE SOLUTIONS HAVE CONVERGED AND WE'RE PRETTY SURE WE  |
   ! HAVE THE RIGHT ANSWER. WRITE THE RESULTS TO OUTPUT FILE    |
   ! AND LET'S TRY A NEW V_0 VALUE.                             |
   !------------------------------------------------------------|

   DO i = 1 , 4*NOP

      WRITE(fid1,FMT='(4(F16.7))') Angle(i), V_int, ReDensV(i)

   ENDDO

   DO i = 1 , 3*Ne

      WRITE(fid2,FMT='(F16.7)')   EIG(i)

   ENDDO

   WRITE(fid3,FMT='(3(F16.7))') 0, 0, 0  !(FOR DISTINCTION)

ENDDO

CLOSE(fid1)
CLOSE(fid2)
CLOSE(fid3)

RETURN
END SUBROUTINE el_ring_V_int
```

makearr: a subroutine that fills the matrix

```
   SUBROUTINE makearr(V0,V_i,arr,H_on)

   USE const
   USE tools

   IMPLICIT NONE

! .. ARGUMENTS ..

   REAL(KIND=8),                                       INTENT(IN)  :: V0, V_i
   INTEGER,                                            INTENT(IN)  :: H_on
   COMPLEX(KIND=8), DIMENSION(-NMAX:NMAX,-NMAX:NMAX), INTENT(OUT) :: arr

! .. VARIABLES ..

   INTEGER      :: i, j

! .. ARRAYS ..

   INTEGER,        DIMENSION(-NMAX:NMAX), PARAMETER :: M=(/(i,i=-NMAX,NMAX)/)
   INTEGER,        DIMENSION(-NMAX:NMAX,-NMAX:NMAX) :: IDE, M1, M2
   COMPLEX(KIND=8),DIMENSION(-NMAX:NMAX,-NMAX:NMAX) :: V_H

! .. EXTERNAL ROUTINES ..

   EXTERNAL :: hartree

   arr = (0.,0.)

!----------------------------------------------------|
!  PUT THE ENERGY EIGENVALUES OF H_0 ON THE DIAGONAL |
!----------------------------------------------------|

   DO i = -NMAX,NMAX

        arr(i,i) = CMPLX( ( DBLE(i) - Phi ) * ( DBLE(i) - Phi ) , 0. )

   ENDDO

   CALL meshgrid(M,M,M1,M2,SIZE(M),SIZE(M))

   IDE = M1 - M2

!-----------------------------------------|
! FILL IN THE EXTERNAL POTENTIAL ELEMENTS |
!-----------------------------------------|
   WHERE (IDE == 0) arr = arr + CMPLX(V0/20.,0.)

   WHERE ((IDE == -20).OR.(IDE == 20)) arr = arr + CMPLX(V0/40.,0.)

   WHERE ((IDE /= 0).AND.(IDE /= -20).AND.(IDE /= 20)) &
      arr = arr + CMPLX( (V0/(4.*PI))*((SIN(IDE*PI/20.))/&
            (IDE*(IDE+20.)*(IDE-20)))*((-(M1*M1)+(2*M1*M2)-(M2*M2)-400)&
            +(((-1)**IDE)*(-(M1*M1)+(2*M1*M2)-(M2*M2)-400))),0.)

!----------------------------------------|
! CALCULATE THE HARTREE TERM AND FILL IN |
!----------------------------------------|

   SELECT CASE (H_on) !SKIP IF H_on .NE. 1

      CASE (1)

         CALL hartree(V_H)

         arr = arr + (R_0*V_i/(2.*PI*PI)) * V_H

   END SELECT

   RETURN
   END SUBROUTINE makearr
```

hartree: a subroutine that calculates the Hartree term

```
   SUBROUTINE hartree(HarArr)

   USE const
   USE NOuter
   USE theta
   USE angleVs

   IMPLICIT NONE

! .. ARGUMENTS ..

   COMPLEX(KIND=8), DIMENSION(-NMAX:NMAX,-NMAX:NMAX), INTENT(OUT) :: HarArr

! .. VARIABLES ..

   INTEGER                               :: i, j
   REAL(KIND=8)                          :: ReH, ImH
   REAL(KIND=8),    DIMENSION(4*NOP)     :: ReI
   COMPLEX(KIND=8), DIMENSION(0:2*NMAX) :: temp

! .. EXTERNAL ROUTINES ..

   REAL(KIND=8), EXTERNAL :: GaussIntI, GaussIntO, HKernel, ReHfunc, ImHfunc

   HarArr = CMPLX(0.,0.)

!---------------------------------------------|
! CALCULATE THE HARTREE INTEGRAL FOR EACH THETA |
!---------------------------------------------|

   DO i = 1 , 4*NOP

      xu = Angle(i)

      ReI(i) = GaussIntI(HKernel,Angle,GausW,4*NOP,th_inc)

   ENDDO

!-------------------------------------|
! CALCULATE THE HARTREE ARRAY ELEMENT |
!-------------------------------------|

   DO i = 0 , 2*NMAX

      NOut = i

      ReH  = GaussIntO(ReHfunc,Angle,GausW,4*NOP,th_inc,ReI)
```

```
     SELECT CASE (i)

        CASE (0)

           ImH = 0.

        CASE DEFAULT

           ImH  = GaussInt0(ImHfunc,Angle,GausW,4*NOP,th_inc,ReI)

     END SELECT

     temp(i) = CMPLX(ReH,ImH)

  ENDDO

!----------------------------------------------------------|
! FILL THE UPPER TRIANGLE OF MATRIX WITH HARTREE ELEMENTS |
!----------------------------------------------------------|

  DO i = -NMAX , NMAX

     DO j = i , NMAX

        HarArr(i,j) = temp(j-i)

     ENDDO

  ENDDO

  RETURN
  END SUBROUTINE hartree
```

GaussInt0: integration of the Hartree function

```
    REAL(KIND=8) FUNCTION GaussInt0(fd,ThetaArr,GausW,NeGsub,pinc,ReIV)

!------------------------------------------------------------|
! FUNCTION GaussInt0 DOES THE (GAUSSIAN) INTEGRATION OF THE  |
! HARTREE ELEMENT (THE OUTER INTEGRAL). THE FUNCTION IS      |
! EVALUATED AT THE ANGLES GIVEN IN ThetaArr, AND WEIGHTED    |
! WITH THE COEFFICIENTS IN GausW.                            |
!------------------------------------------------------------|

    IMPLICIT NONE

    INTEGER,                         INTENT(IN)  :: NeGsub
    REAL(KIND=8),                    INTENT(IN)  :: pinc
    REAL(KIND=8), DIMENSION(NeGsub), INTENT(IN)  :: ThetaArr, GausW, ReIV
    INTEGER                                      :: i2
    REAL(KIND=8), PARAMETER                      :: PI = 3.14159265
    REAL(KIND=8)                                 :: anx, Theta1
    REAL(KIND=8), EXTERNAL                       :: fd

! NB : THE VECTOR ReIV CONTAINs THE INNER INTEGRAL
! AT THE THETA VALUES IN ThetaArr.

    anx = 0.D0

    DO i2 = 1 , NeGsub

       Theta1 = ThetaArr(i2)
       anx    = anx + fd(Theta1,i2,ReIV) * GausW(i2)

    ENDDO

    GaussInt0 = anx * (pinc * 0.5)

    RETURN
    END FUNCTION GaussInt0
```

ReHfunc: the real part of the Hartree function

```
   REAL(KIND=8) FUNCTION ReHfunc(x,ind,ReIV)

!----------------------------------------------------|
! FUNCTION ReHfunc FINDS THE VALUE OF THE REAL PART OF |
! THE HARTREE MATRIX ELEMENT FUNCTION AT THE ANGLE x   |
!----------------------------------------------------|

   USE const
   USE NOuter

   IMPLICIT NONE

   REAL(KIND=8), INTENT(IN)                   :: x
   INTEGER,      INTENT(IN)                   :: ind
   REAL(KIND=8), DIMENSION(4*NOP), INTENT(IN) :: ReIV

   ReHfunc = COS( DBLE(NOut) * x ) * ReIV(ind)

   RETURN
   END FUNCTION ReHfunc
```

ImHfunc: the imaginary part of the Hartree function

```
   REAL(KIND=8) FUNCTION ImHfunc(x,ind,ReIV)

!-----------------------------------------------------|
! FUNCTION ImHfunc FINDS THE VALUE OF THE IMAGINARY PART |
! OF THE HARTREE MATRIX ELEMENT FUNCTION AT THE ANGLE x  |
!-----------------------------------------------------|

   USE const
   USE NOuter

   IMPLICIT NONE

   REAL(KIND=8),                  INTENT(IN) :: x
   INTEGER,                       INTENT(IN) :: ind
   REAL(KIND=8), DIMENSION(4*NOP), INTENT(IN) :: ReIV

   ImHfunc = SIN( DBLE(NOut) * x ) * ReIV(ind)

   RETURN
   END FUNCTION ImHfunc
```

GaussIntI: integration of the Hartree kernel

```
    REAL(KIND=8) FUNCTION GaussIntI(fd,ThetaArr,GausW,NeGsub,pinc)

!------------------------------------------------------------|
! FUNCTION GaussIntI DOES THE (GAUSSIAN) INTEGRATION OF THE  |
! HARTREE TERM (THE INNER INTEGRAL). THE FUNCTION IS         |
! EVALUATED AT THE ANGLES GIVEN IN ThetaArr, AND WEIGHTED    |
! WITH THE COEFFICIENTS IN GausW.                            |
!------------------------------------------------------------|

    IMPLICIT NONE

    INTEGER,                          INTENT(IN)  :: NeGsub
    REAL(KIND=8),                     INTENT(IN)  :: pinc
    REAL(KIND=8), DIMENSION(NeGsub), INTENT(IN)  :: ThetaArr, GausW
    INTEGER                                       :: ifail, i2, irank
    REAL(KIND=8), PARAMETER                       :: PI = 3.14159265
    REAL(KIND=8)                                  :: anx, Theta1
    REAL(KIND=8), EXTERNAL                        :: fd

    anx = 0.D0

    DO i2 = 1 , NeGsub

       Theta1 = ThetaArr(i2)
       anx    = anx + fd(Theta1,i2) * GausW(i2)

    ENDDO

    GaussIntI = anx * (pinc * 0.5)

    RETURN
    END FUNCTION GaussIntI
```

HKernel: the Hartree Kernel

```
   REAL(KIND=8) FUNCTION HKernel(x,ind)

!--------------------------------------------------|
! FUNCTION HKernel FINDS THE VALUE OF THE HARTREE  |
! KERNEL (REAL PART) AT THE ANGLE x                |
!--------------------------------------------------|

   USE const
   USE theta
   USE densVs

   IMPLICIT NONE

   INTEGER, INTENT(IN)       :: ind
   REAL(KIND=8), INTENT(IN)  :: x
   REAL(KIND=8), PARAMETER   :: mu = 0.3
   REAL(KIND=8)              :: densi

   SELECT CASE (DoDens)

      CASE (0)

         densi = DBLE(Ne)

      CASE (1)

         densi = ReDensV(ind)

      CASE DEFAULT

         PRINT*, 'Error in function innra, DoDens = (0,1)'
         RETURN

   END SELECT


   HKernel = (densi) / (SQRT((mu*mu)+((SIN((xu-x)/2.))*(SIN((xu-x)/2.)))))

   RETURN
   END FUNCTION HKernel
```

Modules

const

```
  MODULE const

!-----------------------------------------|
! MODULE const KEEPS THE GLOBAL CONSTANTS |
!-----------------------------------------|

   IMPLICIT NONE

   INTEGER,      PARAMETER    :: Ne = 3, NMAX = 20, lda = 2*NMAX+1
   INTEGER,      PARAMETER    :: NOP = 4*NMAX+1
   REAL(KIND=8), PARAMETER    :: Phi = 0., PI = 3.14159265358979
   REAL(KIND=8), PARAMETER    :: R_0 = 5., xtake = 0.999
   COMPLEX(KIND=8), PARAMETER :: ci = (0.D0,1.D0)

  END MODULE const
```

NOuter

```
  MODULE NOuter

!--------------------------------------|
! MODULE NOuter KEEPS THE EXPONENT FOR |
! USE IN ReHfunc AND ImHfunc           |
!--------------------------------------|

   IMPLICIT NONE

   INTEGER :: NOut

  END MODULE NOuter
```

theta

```
  MODULE theta

!----------------------------------|
! MODULE theta KEEPS THE ANGLE FOR |
! USE IN HKernel                   |
!----------------------------------|

   IMPLICIT NONE

   REAL(KIND=8) :: xu

  END MODULE theta
```

tools

```
 MODULE tools

  IMPLICIT NONE

 CONTAINS

  SUBROUTINE meshgrid(x,y,xo,yo,Nx,Ny)

!--------------------------------------------------|
! SUBROUTINE meshgrid TAKES THE VECTORS x AND y AND |
! MAKES MESH MATRICES                               |
!   xo : THE ROWS CONTAIN x                         |
!   yo : THE COLUMNS CONTAIN y                      |
! THE IDEA IS STOLEN FROM MATLAB                    |
!--------------------------------------------------|

  INTEGER, DIMENSION(:),     INTENT(IN)  :: x, y
  INTEGER,                   INTENT(IN)  :: Nx, Ny
  INTEGER, DIMENSION(Ny,Nx), INTENT(OUT) :: xo, yo
  INTEGER                                :: i

  DO i = 1 , Ny

     xo(i,:) = x(:)

  ENDDO

  DO i = 1 , Nx

     yo(:,i) = y(:)

  ENDDO

  RETURN
  END SUBROUTINE meshgrid

 END MODULE tools
```

angleVs

```
 MODULE angleVs

!-------------------------------------------------|
! MODULE angleVs KEEPS THE ANGLE VALUES, INCREMENT |
! AND WEIGHTS FOR GAUSSIAN INTEGRATION             |
!-------------------------------------------------|

  USE const

  IMPLICIT NONE

  REAL(KIND=8)                   :: th_inc
  REAL(KIND=8), DIMENSION(4*NOP) :: Angle, GausW

 END MODULE angleVs
```

densVs

```
  MODULE densVs

!----------------------------------------|
! MODULE densVs KEEPS THE DENSITY VECTOR |
!----------------------------------------|

   USE const

   IMPLICIT NONE

   INTEGER                        :: DoDens
   REAL(KIND=8), DIMENSION(4*NOP) :: ReDensV

  END MODULE densVs
```

ComDensit

```
  MODULE ComDensit

   IMPLICIT NONE

  CONTAINS

   REAL(KIND=8) FUNCTION densityf(x,ReCvs,ImCvs)

!------------------------------------------------------|
! FUNCTION densityf CALCULATES THE ELECTRON DENSITY:   |
!                                                      |
! n(theta) = sum _{n=1} ^{Ne} (| Psi_n |^2)            |
!                                                      |
! ReCvs AND ImCvs ARE THE REAL AND IMAGINARY PARTS OF  |
! THE WEIGHT COEFFICIENTS FOR THE BASIS THAT Psi IS    |
! EXPANDED IN.                                         |
!------------------------------------------------------|

   USE const

! .. ARGUMENTS ..

   REAL(KIND=8), INTENT(IN)                    :: x
   REAL(KIND=8), INTENT(IN), DIMENSION(lda,Ne) :: ReCvs, ImCvs

! .. LOCAL VARIABLES ..

   INTEGER        :: i, j, k
   COMPLEX(KIND=8) :: temp


   temp = 0.

   DO k = 1 , Ne

      DO i = 1 , lda

         DO j = 1 , lda

            temp = temp +  CMPLX(ReCvs(i,k),ImCvs(i,k)) * &
                           CMPLX(ReCvs(j,k),ImCvs(j,k)) * &
                           EXP(ci * DBLE(j-i) * x)

         ENDDO

      ENDDO

   ENDDO

   densityf = DBLE(temp)

   RETURN
   END FUNCTION densityf

  END MODULE ComDensit
```