Bachelorproject

Nonlinear Optics with Surface Plasmons

by Lars Guski
& Martin Cramer Pedersen

June 2007

Contents

1 Summary in Danish - Resumé på dansk **2**

2 Introduction **3**

3 Plasmons **3**

4 Single Plasmon **5**
4.1 Singlemode singleplasmon model 5
4.2 Applying the model 6

5 Calculations - Single Plasmon **7**
5.1 Constructing the algorithm 7
5.2 Initial conditions 10
5.3 Results 11

6 Multiple Plasmons **13**
6.1 Multimode multiplasmon model 13
6.2 Applying the model 14

7 Calculations - Multiple Plasmons **15**
7.1 Constructing the algorithm 15
7.2 Initial conditions 16
7.3 Results 17

8 Comparing the results **19**

9 Conclusion **21**

10 Appendix **23**
10.1 Appendix A 23
10.2 Appendix B 24
10.3 Appendix C 25
10.4 Appendix D 32

11 References **40**

1 Summary in Danish - Resumé på dansk

På baggrund af "traditionel" kvanteoptik og en endimensionel model for et system bestående af en enkelt plasmon og et atom forsøgte vi at generalisere os frem til en mere kompleks model for et system bestående af flere (eller rettere to) plasmoner og det samme (to-niveau) atom.
Målet var at opstille en række sandsynlighedsamplituder for diverse potentielle sluttilstande. Metoden skulle gerne være den samme som i den simple model med en plasmon.
Som udgangspunkt opstillede vi en Hamilton, der gerne skulle beskrive vekselvirkningen mellem plasmon og atom. Derefter forsøgte vi at konstruere nogle tilstande, vi kunne få den selvsamme Hamilton til at virke på. Slutteligt benyttede vi begge disse i den tidsafhængige Schödingerligning, der gerne skulle give os de ønskede sandsynligheder.
Med udgangspunkt i bølgepakker med form som Gaussfordelinger (dette var begyndelsesbetingelserne til vores differentialligninger) løste vi nu Schrödingerligningen og var i stand til at summere vores resultater op, enten ved et plot af sandsynlighedsamplituder for forskellige typer af bølgefunktioner eller ved at lave et filmklip, hvori bølgepakken ramte atomet, hvorefter pakken delte sig i en reflekteret del og en transmitteret del.
Ved at gennemføre en række af de ovenstående simulationer (både for en plasmon og for to plasmoner) var vi nu i stand til at konstruere en graf for sandsynligheden for transmission kontra bredden af bølgepakken, vi sendte ind.
Jo bredere vores bølgepakke var i positionsrummet var, des smallere (og derfor des mere sandsynligt, at plasmonen var på resonansfrekvensen og derfor des mere vekselvirkende - dette er et resultat af Fouriertransformationen) var bølgepakken.
Hele formålet med projektet var nu at sammenligne de to grafer (en for en plasmon og en for to plasmoner) for transmissionen versus bølgepakkens bredde.
Forventningen var (eller nærmere den klassiske forventning ville være), at hvis sandsynligheden for at transmittere en plasmon ved en given bølgepakkebredde var f.eks. 50% for en-plasmon-scenariet, så ville sandsynligheden for at transmittere begge plasmoner være 25% (man forventer naturligvis $50\%^2$) i to-plasmon-scenariet.
Men ved nærmere inspektion af vores grafer viste det sig, at dette ikke var tilfældet.
Vi opnåede nogle resultater, der bedst af alt kunne fortolkes således, at hvis begge plasmoner var i nærheden af resonansfrekvensen, så var de begge i stand til at vekselvirke med atomet, men da atomet kun er "bygget" til at vekselvirke med en plasmon, svækkedes den egentlige vekselvirkning (og derved muligheden for refleksion) en del.
Med andre ord så vi altså to plasmoner vekselvirke med hinanden - dog gennem vekselvirkningen med et atom - dette er íkke desto mindre en stykke vej fra den traditionelle opfattelse, at den enkelte plasmon ikke kan "se" andre plasmoner (eksempelvis er lyset fra en lampe ikke påvirket af lyset fra en anden lampe).
Der var altså en ganske udemærket årsag til, at projektet faldt under kategorien "ikke-lineær optik".

2 Introduction

> *Both matter and radiation possess a remarkable duality of character, as they sometimes exhibit the properties of waves, at other times those of particles. Now it is obvious that a thing cannot be a form of wave motion and composed of particles at the same time - the two concepts are too different.*
>
> Werner Heisenberg, 1930

Luckily for this project, Heisenberg was not entirely correct in the statement above. A few years after 1930 he (and most other parts of the scientific society; or rather those that had not already done so) embraced this new way of thinking and understanding particles and light. This way of thinking is now referred to as particle-wave-duality.
This "concluded" a discussion, that had been going on for hundreds of years. One group believed that Young's interferenceexperiments and Newton's experiments concerning the dispersion of sunlight through a prism proved, that light was electromagnetic waves, while the other group claimed, that Compton's experiments along with Einstein's photoelectric effect were arguments enough to prove that light was indeed massless particles.
In the late 1920's and the early 1930's Heisenberg, Niels Bohr, Paul Dirac, and Ernst Schrödinger developed, what is now known as the quantummechanical description of the nature of atoms and other particles and paused the discussion for the time being.
Today, quantumphysical methods are used all over the world to all sorts of technology and experiments, while the theory itself is widely considered one of the most successful theories in modern physics due to remarkably accurate and useful results.
In this project we will attempt to treat a simple quantumoptical system using traditional quantummechanical calculations combined with numerical solutions of the numerous differential equations, that will rise, when we attempt to apply our model to the system.
The system in question will be an atom in a cavity being hit by one or two photons (or rather plasmons, as we will stick to a onedimensional approximation of the problem).
The particle, we call plasmon, is the physical name of a photon contained in a onedimensional enviroment. Though we were first surprised by the amount of articles and literature on system like the ones, we studied, it became clear to us, that this kind of particle is rather important, since it is found everywhere in modern technology. An example would be optical fibers used for data transmission.
As one was taught in the more simple courses concerning particle physics and quantum optics, the photon is not interacting with other photons. The same is the case for plasmons as these are nothing but a special kind of photons.
We do, however, also know that if one plasmon were to excite an atom, another plasmon would not be capable of doing the same thing (until the atom had relaxed again, that is).
So in a sense we should be able to make plasmons interact with each other through the dynamics of the plasmon-atom-connection.
The wavefunction of one plasmon would be dependant of the wavefunction of another plasmon - and this is the interaction we are looking for. We shall aim our efforts in this report towards describing this interaction-like behaviour - by simulating the whereabouts of the plasmons (or rather the energy brought into the system by the plasmons).

3 Plasmons

There are two ways of describing the plasmon - as there will always be with any quantumoptical particle due to the particle-wave-duality of most quantumdynamical objects.
In the particle-picture a plasmon is a photon travelling along a wire. The allowed wavevectors (the correct term when dealing with particles would be momentumvectors, since the word wavewector implicates a wave) of these are confined to the direction parallel to the wire. These plasmons may

or may not interact with any atoms along their path on the wire due to the traditional photoelectric effect.
In the wave-picture we can consider plasmons as fluctuations in the electromagnetic field surrounding the wire. These have the traditional properties of any other electromagnetic wave, but no wavecomponents travelling orthogonal to the wire are permitted.

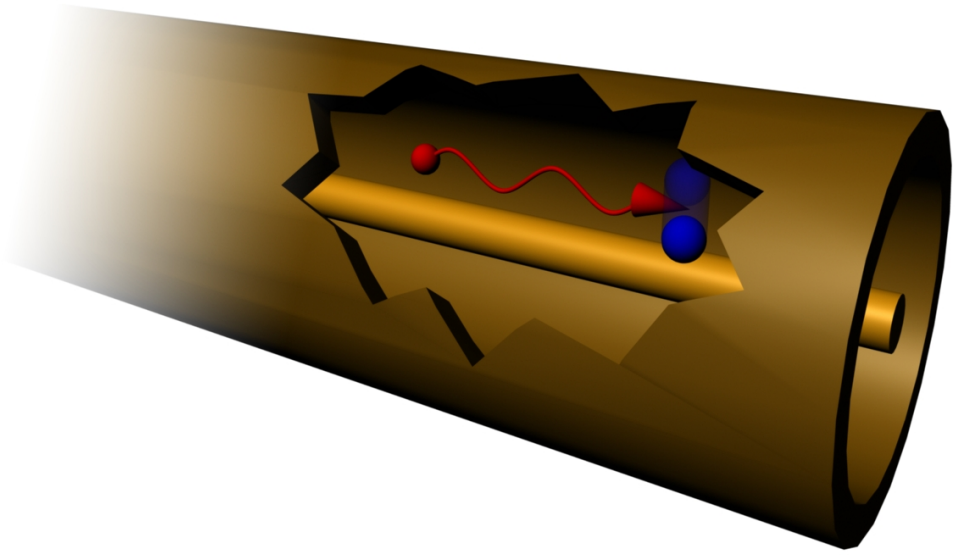

Figure 3.1 - A plasmon inside a coaxial cable interacting with an two-level atom. This clearly gives rise to our one-dimensionel approximation as the plasmon only propagate in the direction of the cable[1].

As we have tried to indicate on figure 3.1, we will be dealing with a onedimensional approximation (this would be the x-direction on the figure), since a plasmon is contained in a wire. This ensures that our plasmons will hit and interact with the atom and that we can discard the possibility of any kind of emission perpendicular to the wire.
As one might have guessed, this is the reason for the name "surface" plasmons. In this project, the plasmons exist on the surface of the wires in a coaxialcable. In two dimension they would exist on the surface of a conductor.
This is the very reason, why this project is about plasmons rather than the more general case - the photon.
When dealing with a photon in a 3D-enviroment, we have to consider all sorts of issues like "will the photon hit the atom - or come close enough to interact?", "will emission be possible in all directions?", and "will we still be able to approximate with a discrete wavespace, now that we have three times as many dimensions?"; we would have to incorporate the crosssection of the atom into our calculations and severely complicate our wavevectorspace. And the actual dynamics of plasmon-atom-interaction is exactly the same as photon-atom-interaction.
We will be concerning ourselves mainly with the case, where two plasmons propagate towards the atom as this gives rise to a number of phenomena that cannot be observed in the case of the lonely plasmon. The price is paid in anaytical compatibility, as we are forced to solve the two-plasmon case numerically. Furthermore we are forced to abandon the continuous wavevectorspace, as MatLab has obvious trouble accomprehending infinite potential results.
The purpose of these calculations would be to observe the different possible ways of reflected and/or

[1]Thanks to Martin Troels Eberhardt for creating this picture.

transmitted plasmons - i.e. two left-propagating plasmons enter; how many right-propagating plasmons exit?
We have further simplified our model by assuming that the atom is a socalled "two-level-atom"; meaning that once the atom is excited no further excitations are possible. There is only two energylevels; $|b\rangle$ and $|a\rangle$. And we have negleted the possibility of the atom using both plasmons in the excitation. Though one will see later that the atom is capable of absorbing the first plasmon, then emitting it, and still having time to absorb the second one - this will happen, when the potential distance between the two plasmons increases.
Though we have chosen to limit this project to a maximum of two plasmons, there really is no upper limit to the number of excitation, we might be operating with.
But keep in mind, that the processing power (and thus time) required to do the following simulation will grow rapidly, as the complexity of the system increases.
A good way of representing the problem would be a traditional Feynman-diagram:

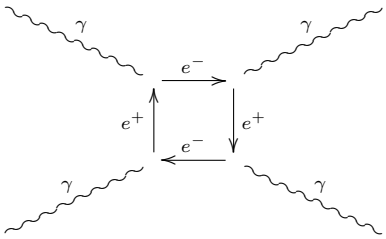

Figure 3.2 - The synopsis of this project contained in this Feynman-diagrams.

The object of the project is to show that a system consisting of an atom and two plasmons obey the dynamics described in the diagram above, rather than just the traditional linear dynamics.

4 Single Plasmon

4.1 Singlemode singleplasmon model

The model, which we are going to apply to the single-plasmon case, can be found in its full extent in [5].
The dynamics of a system consisting of a single plasmon and a single atom is more or less equal to the model found in [6] or [7].
The Hamiltonian of a two-level (a low-energy-level $|b\rangle$ and a high-energy-level $|a\rangle$) atom in a multimode, electromagnetic field is given by[2]:

$$\begin{aligned}\hat{H} &= \hat{H}_{atom} + \hat{H}_{field} + \hat{H}_{interaction} \\ &= \frac{1}{2}\hbar\omega_0\left(\sigma_{aa} - \sigma_{bb}\right) + \sum_k \hbar\omega_k\left(\hat{a}_k\hat{a}_k^\dagger + \frac{1}{2}\right) + \sum_k \hbar g_k\left(\sigma_+\hat{a}_k + \sigma_-\hat{a}_k^\dagger\right)\end{aligned}$$

We have used the abbreviations:

$$\sigma_{ij} = |i\rangle\langle j| \qquad g_k^{ij} = \frac{M_{ij}\cdot\hat{\epsilon}_k E_k}{\hbar} \qquad \text{and} \qquad \sigma_+ = |a\rangle\langle b| \qquad \sigma_- = |b\rangle\langle a|$$

For a singlemode field (one kind of k-mode) this equation reduces to ($\sigma_z = \sigma_{aa} - \sigma_{bb}$ - and the zeropoint of the energy in the atom is set exactly between the two energy levels.):

$$\hat{H} = \frac{1}{2}\hbar\omega_0\sigma_z + \hbar\omega\hat{a}\hat{a}^\dagger + \hbar g\left(\sigma_+\hat{a} + \sigma_-\hat{a}^\dagger\right)$$

[2]More on the various operators in Appendix A.

Notice that we have chosen to neglect the 1/2-term representing the energy of the zeromode. The only difference between a system consisting of a photon and an atom or a plasmon and an atom is contained in the factor, g_k.
As mentioned before g is given by:

$$g_k = \frac{M_{ab} \cdot \epsilon_k E_k}{\hbar}$$

Where M_{ab} is the dipole-matrix element for a transistion from state $|a\rangle$ to state $|b\rangle$. E_k is the energy in the mode k and ϵ_k is the polarization of the lightfield.
>From [6] we find, that

$$E_k = \sqrt{\frac{\hbar\omega_k}{2\epsilon_0 V}}$$

As we can see the interactive coupling ($\hat{H}_{int}$) between the atom and the plasmon/photon is proportional with $\sqrt{\frac{1}{V}}$. Since the plasmon is contained in a wire, the effective volume will be much smaller. The interaction will therefore be much stronger.

4.2 Applying the model

An atom in a singlemode field has the wavefunction:

$$|\Psi(t)\rangle = c_{a,n}(t)\,|a,n\rangle + c_{b,n+1}(t)\,|b,n+1\rangle$$

Using our Hamiltonian, our waveequation and the time-dependant Schrödingerequation:

$$i\hbar\frac{\partial|\Psi\rangle}{\partial t} = \hat{V}\,|\Psi\rangle$$

where[3] ($\hat{V}$ is the timeevolved $\hat{H}_{int}$):

$$\begin{aligned}\hat{V} &= \hbar g\left(\sigma_+\hat{a}e^{i\Delta t} + \sigma_-\hat{a}^\dagger e^{-i\Delta t}\right)\\ \Delta &= \omega - \omega_0\end{aligned}$$

We can write the following differential equations for the timedependant probability amplitudes:

$$\begin{aligned}\frac{\partial c_{a,n}(t)}{\partial t} &= -ig\sqrt{n+1}e^{i\Delta t}c_{b,n+1}\\ \frac{\partial c_{b,n+1}(t)}{\partial t} &= -ig\sqrt{n+1}e^{-i\Delta t}c_{a,n}\end{aligned}$$

Based on these equations we can derive the following solutions for the probability amplitudes:

$$\begin{aligned}c_{a,n}(t) &= \left(c_{a,n}(0)\left(\cos\frac{\Omega_n t}{2} - \frac{i\Delta}{\Omega_n}\sin\frac{\Omega_n t}{2}\right) - c_{b,n+1}(0)\,\frac{2ig\sqrt{n+1}}{\Omega_n}\sin\frac{\Omega_n t}{2}\right)e^{\frac{i\Delta t}{2}}\\ c_{b,n+1}(t) &= \left(c_{b,n+1}(0)\left(\cos\frac{\Omega_n t}{2} + \frac{i\Delta}{\Omega_n}\sin\frac{\Omega_n t}{2}\right) - c_{a,n}(0)\,\frac{2ig\sqrt{n+1}}{\Omega_n}\sin\frac{\Omega_n t}{2}\right)e^{-\frac{i\Delta t}{2}}\end{aligned}$$

where:

$$\Omega_n^2 = \Delta^2 + 4g^2(n+1)$$

[3] See Appendix B for detailed derivation of the timeevolution.

If our system indeed only consists of one plasmon, we can put $|n\rangle = |0\rangle$ and $|n+1\rangle = |1\rangle$ to find the probability amplitudes in this case.
We find that:

$$\begin{aligned} c_{a,0}(t) &= \left(c_{a,0}(0)\left(\cos\frac{\Omega_0 t}{2} - \frac{i\Delta}{\Omega_0}\sin\frac{\Omega_0 t}{2}\right) - c_{b,1}(0)\frac{2ig}{\Omega_0}\sin\frac{\Omega_0 t}{2}\right)e^{\frac{i\Delta t}{2}} \\ c_{b,1}(t) &= \left(c_{b,1}(0)\left(\cos\frac{\Omega_0 t}{2} + \frac{i\Delta}{\Omega_0}\sin\frac{\Omega_0 t}{2}\right) - c_{a,0}(0)\frac{2ig}{\Omega_0}\sin\frac{\Omega_0 t}{2}\right)e^{-\frac{i\Delta t}{2}} \end{aligned}$$

This is the "standard" way of treating the singlemode-field-problem.
We will in the next section try to develop a method of calculating the timeevolution of a system consisting of an atom and a plasmon capable of "obtaining" a variety of different frequencies (or rather wavenumbers).
In reality the singlemode field is an impossibility due to imperfection of the constructed system, i.e. different ways of broadening the emission and absorbtion spectra of the atom or inability to construct complete "pure"/monochromatic beams (or rather plasmons that is not a coherent state).
One problem with this calculation is, that there can be no geometric interpretation of the equations above - they do not contain information of the whereabouts of the plasmon. This is another good reason to turn to numerical calculation instead of analytical.

5 Calculations - Single Plasmon

5.1 Constructing the algorithm

The purpose of this project is to construct an algorithm capable of simulating the quantumphysical dynamics of any given plasmon system under given initial conditions (this will later be limited to Gaussian waves - but only for this project - "feeding" the algorithm a different inital state will be just as possible, we just had to limit ourselves to one type of initial conditions).
The way to do this will be programming at set of scripts to be run by MatLab.
MatLab is chosen due to its matrix-based calculations methods, which corresponds well to the operator mathematics in quantum physics.
The fundamental part of our calculations will be observations of the interactions between different states.
In order to do this we will need Schrödingers equations along with a way of identifying the different combinations of photons and/or the atom.
MatLab has an algorithm ,`ode45`, that can evaluate a set of coupled partial differential equations based only on the equation itself and a set of boundary conditions. Recalling the timedependant Schrödinger equation

$$i\hbar\frac{\partial\Psi}{\partial t} = \hat{H}\Psi$$

we see that if we can put this into MatLabs syntax, we will have an expression that can be solved by `ode45`. The solutions will then contain the timeevolution of the system.
The important aspect in these processes will be keeping track of the different states while constructing a Hamiltonian-like operator, which will be one of the base elements of our description.
To descibe the dynamics of a quantumoptical system, we will have to describe the coupling from one state to another.
As written before our Hamiltonian is given by

$$\begin{aligned} \frac{\hat{H}}{g} &= \sum_k \sum_{i=L,R}\left(\sigma_- \hat{a}^\dagger_{k,i} + \sigma_+ \hat{a}_{k,i}\right) + \sum_n \sum_{j=L,R}\frac{\hbar c k_n}{g}\hat{a}^\dagger_{n,j}\hat{a}_{n,j} \\ &= \hat{V} + \hat{V}^\dagger + \hat{D} \end{aligned}$$

where $\hat{V}$ and $\hat{V}^\dagger$ describes the coupling terms of the operator and $\hat{D}$ describes the field properties of the system. We will use the notation:

$$\Delta\omega = \frac{\hbar c k_n}{g}$$

The interpretation of this quantity is explained in figure 5.1.

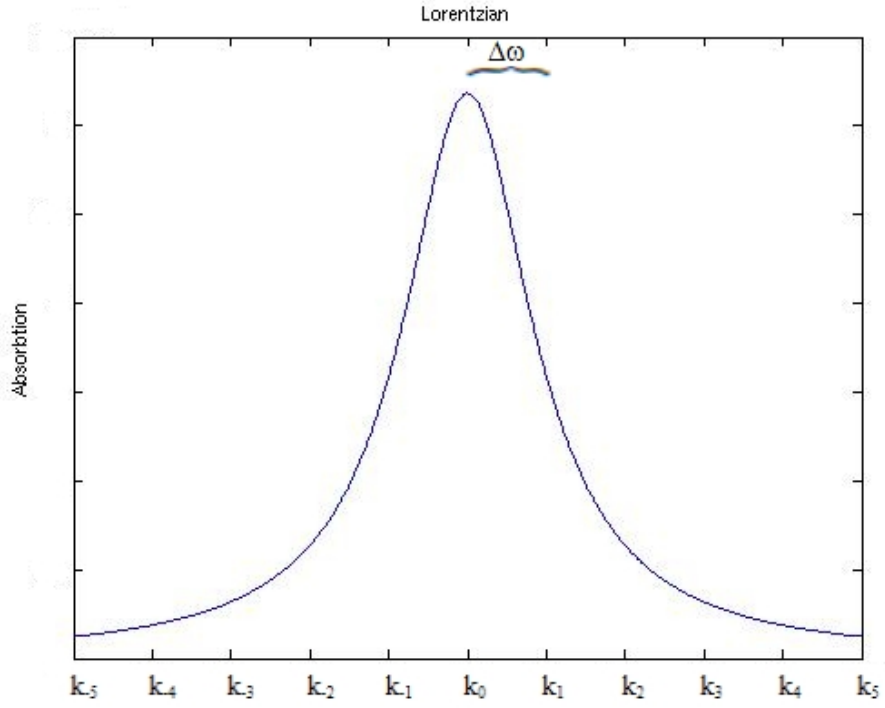

Figure 5.1 - The graph drawn is a traditional Lorentzian representing absorbtion of an energylevel in the atom. The properties (and importance) of $\Delta\omega$ should be pretty obvious.

A note should be made on the units (that we more or less neglect through the entire project). Observing the following property of the Shrödingerequation

$$i\hbar\frac{\partial\Psi}{\partial gt} = \frac{\hat{H}}{g}\Psi$$

we get an idea of the timescale, we should be looking at; and to calculate this more explicitly, we seek to approximate the quantity gt.
First of all we notice that $g = \hbar c k_n/\Delta\omega$. As we observe a discrete wavespace (more on this later), we can write k in terms of "cavity-length" such that $k = \frac{2\pi}{L}$.
Writing L as velocity times time (T is the size of the timescale, that should correspond to one period in a cavity of length, L) and substitute, we see that

$$g = \hbar c\frac{2\pi}{L} = \hbar c\frac{2\pi}{cT}$$

setting $\hbar = 1$,[4] we now obtain the relation:

$$gT = \frac{2\pi}{\Delta\omega}$$

[4]Standard atomic units.

This should give us the boundries of our calculations. If we set $\Delta\omega = 0.1$, our calculations and simulations should be constrained to the timeinterval, $t \in [0; 62.8]$.
All this is important, as MatLab will automatically assume periodic boundary conditions unless told otherwise. If we leave the system "on" in longer timeintervals, we would observe the transmitted and reflected wave hit the atom once again (and again and again) causing some not-so-easy-to-interpret graphs.
As we are dealing with numerical operations, a continous wavevectorspace becomes an impossibility, as it would require an "$(\infty \times \infty)$-matrix" to describe the different couplings of the system. We turn to a discrete space consisting of n_{max} possible states.
Since our plasmon is capable of propagating either left or right, we now have n_{max} potential states for left-propagating plasmons and n_{max} potential states for right-propagating plasmons, summing up to a total of $2n_{max} + 1$ different states (the "+1" represents the state in which the atom has absorbed the plasmon).
Please notice that through this entire project, plasmons will be characterized by the direction in which they are heading and not their position in relation to the atom.
Due to periodic boundary conditions, our positionspace will repeat itself, hence making the actual position of the plasmon hard to express; i.e. a rightmoving plasmon located to the right of the atom will after a certain amount of time be located to the left of the atom without having actually been in the vicinity of the atom.
The abundance of different states will then be:

Leftpropagating plasmons	Rightpropagating plasmons	State of the atom	Abundance
0	0	Excited	1
0	1	Ground	n_{max}
1	0	Ground	n_{max}

In order to make MatLab understand the coupling, we need a delicate system to enumerate the different states. We define the ground state as being the state with $L = 1$.
The states with left-propagating plasmons will be enumerated $L \in [2; n_{max} + 1]$ and the states with rightpropating will be enumerated $L \in [n_{max} + 2; 2n_{max} + 1]$.
This way, we will be able to choose a central wavenumber, k_0 (preferably the resonance vector, but not necesarily; though any other choice would not make much sense), and the observe the dynamics of the nearby wavevectors - the size of the word "nearby" is based on the quantity $\Delta\omega$ and the number of states included in out calculations, n_{max}.
In other words our Hamiltonian covers an "area" in wavevectorspace corresponding to $\left\{k_0 - \frac{n_{max}-1}{2}, \ldots, k_0, \ldots, k_0 + \frac{n_{max}-1}{2}\right\}$.
In this discrete wavevectorspace our Hamiltonian becomes a $((2n_{max} + 1) \times (2n_{max} + 1))$-matrix. The diagonal elements of this matrix will consist of the energy relative to k_0.
Using the algorithms described in Appendix C, we can construct an operator, whose matrixrepresentation is given by (we have chosen $n_{max} = 5$; this number will usually be much, much larger, but it should give an idea of the properties of the matrix, altogether with $\Delta\omega = 0.1$):

$$\hat{H} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -0.2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -0.1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 \end{pmatrix}$$

The top row and the left column represent the coupling between the excited state of the atom to any of the potential quantumoptical states.

And as stated the diagonal is the energy relative to the central k, which would then be $(4;4)$ for a leftgoing k_0-plasmon and $(9;9)$ for a rightgoing plasmon.
Using the operator described above in our Schrödinger-equation we get an idea of the challenge, we are about to give MatLab.
Applying the operator to a wavevector, Ψ, consisting of $2n_{max}+1$ elements, we should have the same number of coupled differential equations. Setting $n_{max} = 51$ to solve over 100 differential equation - this number will escalate drastically, when we move to the dual plasmon case.

5.2 Initial conditions

Whenever one is solving a differential equation, one will need a set of boundary conditions and a set of initial condition in order to obtain a unique solution for the (set of) equations.
We will focus our efforts on a wave of the following type:

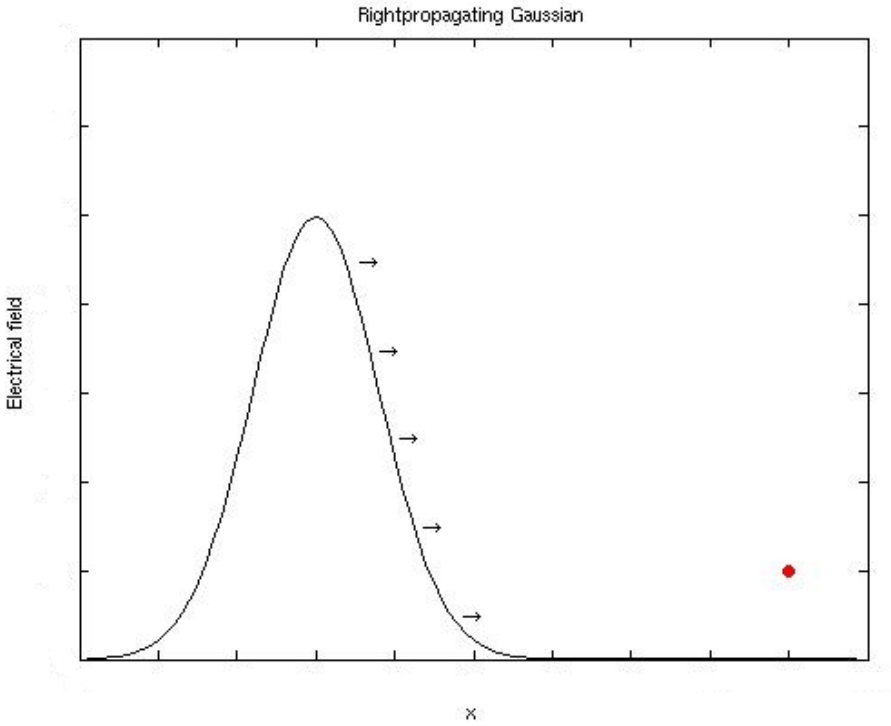

Figure 5.2 - Rightpropagating Gaussian and an atom. This is the first and foremost initial condition used in our program. The Gaussian wave is "easily" obtained with a traditional pulselaser.

The standard Gaussian wave can be written as:

$$E(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{x-\mu}{2\sigma}\right)^2}$$

Once this is done, we can plug this initial condition into `ode45` and calculate the timeevolution of such a system.
We are now faced with a bit of a problem. The function written above exists in positionspace, whereas our Hamiltonian governs the dynamics in wavevectorspace. Long story short, we need to Fouriertransform this Gaussian in order to obtain a usable initial condition for our equations.
Luckily, MatLab is capable of doing this for us, using the operation `FFT`, which is short for Fast Fourier Transform.
Furthermore we shall make use of the command `FFTSHIFT`, which is used to keep our function within the boundries of the cavity (i.e. shifting the parts of the function that exceed the cavity to the other side).

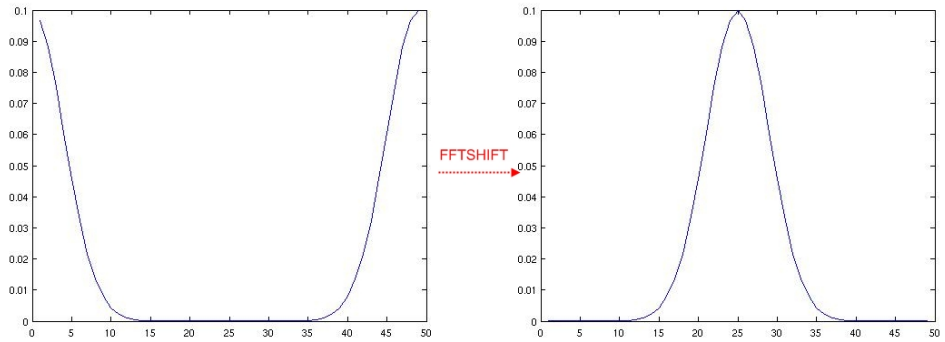

Figure 5.3 - The very useful result of the FFTSHIFT-command.

Whenever one uses `FFT` on a vector, one will obtain the Fouriertransformed vector - the elements, however, will be shifted in the way indicated above. Using the command `FFTSHIFT` will restore the order of the elements.
There are two key parameters describing a Gaussian wave; σ is governing the width of our Gaussian and μ is describing the location of the center of the wave (keep in mind that our atom is located in $x = 0$).
The rest of the dynamics of the system is left completely to the mercy of the Schrödingerequation and our Hamiltonian.
This concludes our list of quantities and functions necessary to construct our string of programs and we (well, MatLab actually) should now be ready for accepting the task of numerically calculating the different properties of the singleplasmon multimode case.
In traditional quantumphysical notation we need a creationoperator with the property:

$$\sum_k \Phi_k \hat{a}_k^\dagger |0\rangle = \hat{a}_\Phi^\dagger |0\rangle = |1_\Phi\rangle$$

In this case Φ would be the Fouriertransform of the initial condition (the Gaussian wave). The quantity $|1_\Phi\rangle$ represents the wavefunction, where we have one plasmon in a Φ-state.
This way we can write our resulting state as:

$$|1_\Phi\rangle = \sum_k c_k |1_k\rangle$$

The information regarding our initial state can now be kept in a string (actually, a vector), where the k'th element is c_k.
This vector will later serve as initial conditions for our Schrödinger equation, where we will use it as $\Psi(t = 0)$.

5.3 Results

For the simulation with one plasmon, we have chosen only to send in a plasmon from the right side (this would be a leftpropagating plasmon), since obvious symmetry arguments would yield the same result for rightpropagating plasmons.
One way of displaying the dynamics of the system would be to derive the probability amplitudes for either of the three kinds of potential states (atom in ground state, leftpropagating plasmon, or rightpropagating plasmon).
The following two pictures represent the solution to a system with initial conditions being a leftpropagating Gaussian wave with $\sigma = 2$ and a interaction charaterized by $\Delta\omega = 2$.

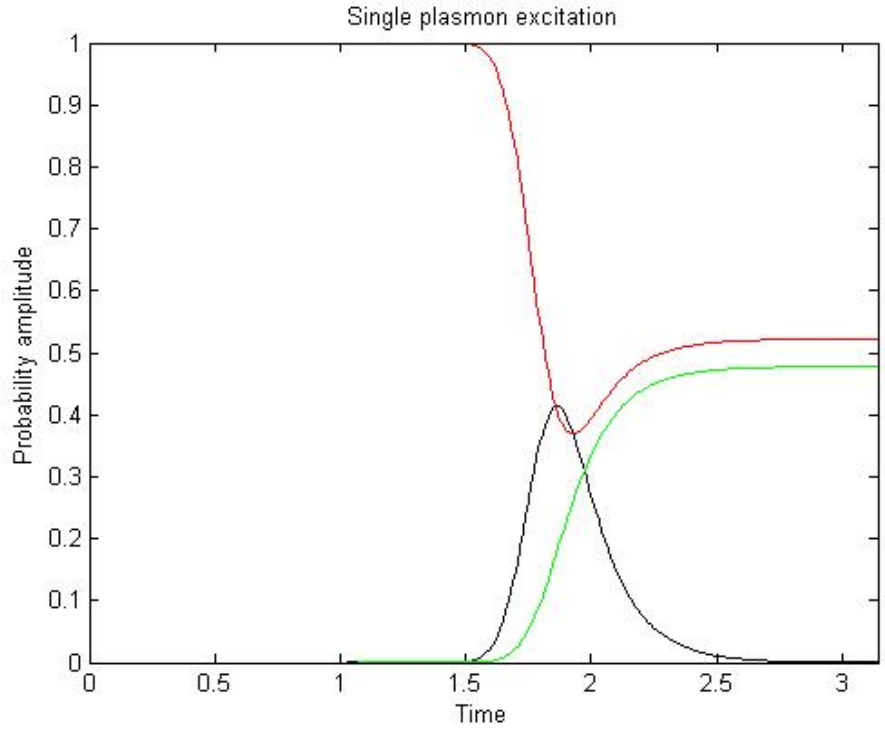

Figure 5.4 - The evolution of our system with respect to time. The black graph corresponds to the probability for having an excited atom, the green and the red graph covers the probability of having a rightpropagating or a leftpropagating plasmon, respectively.

This picture clearly shows the requirements for reflecting a wave, as it is quite obvious that a reflected plasmon is actually a plasmon that has been absorbed by the atom and emitted in the direction from whivh it came (there is no way of having a rightgoing plasmon until the probability of an excited atom begins to rise).
Another way of displaying our resuls could be in a movie-like fashion, where we could observe a travelling wave in positionspace and transmitted and reflected wave.
The best way of picturing this is in a cartoon-like fashion[5] like this:

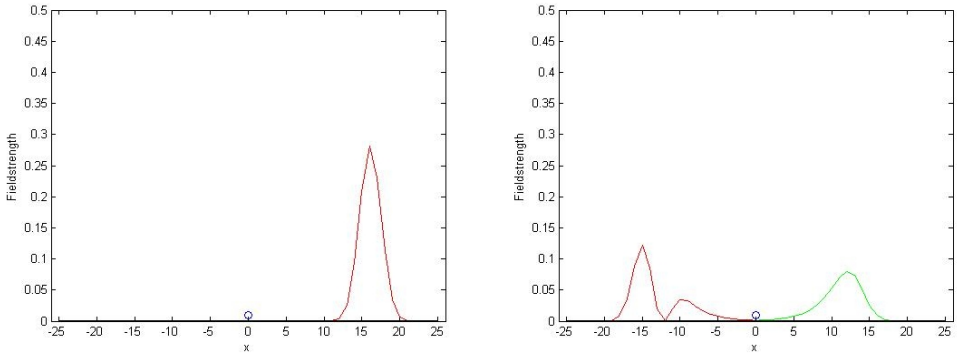

Figure 5.5 - Before-and-after-picture of the interaction between a leftpropagating Gaussian wave and an atom. The red graphs are travelling to the left and the green graph is travelling to the right.

[5]The actual "movieclip" can be found at
http://www.fys.ku.dk/∼mcp/Bachelorprojekt/SinglePlasmonMovie.wmv

Comparing these pictures to the graph in figure 5.4, we can see, that we (obviously) start out with a leftpropagating wave and end up with about 55% chance of having a leftpropagating plasmon (approximately the integral under the red curve in the second picture) and roughly 45% chance of having a rightpropagating plasmon (the area under the green curve in the second picture as a reference).
In other words figure 4.4 and figure 4.5 pretty clearly state the same facts about our system.
For the singleplasmon scheme we obtained the following graph by running our simulation numerous times (30 to be more exact), each time with a different initial value for the width of the incoming wave (this would be the parameter σ):

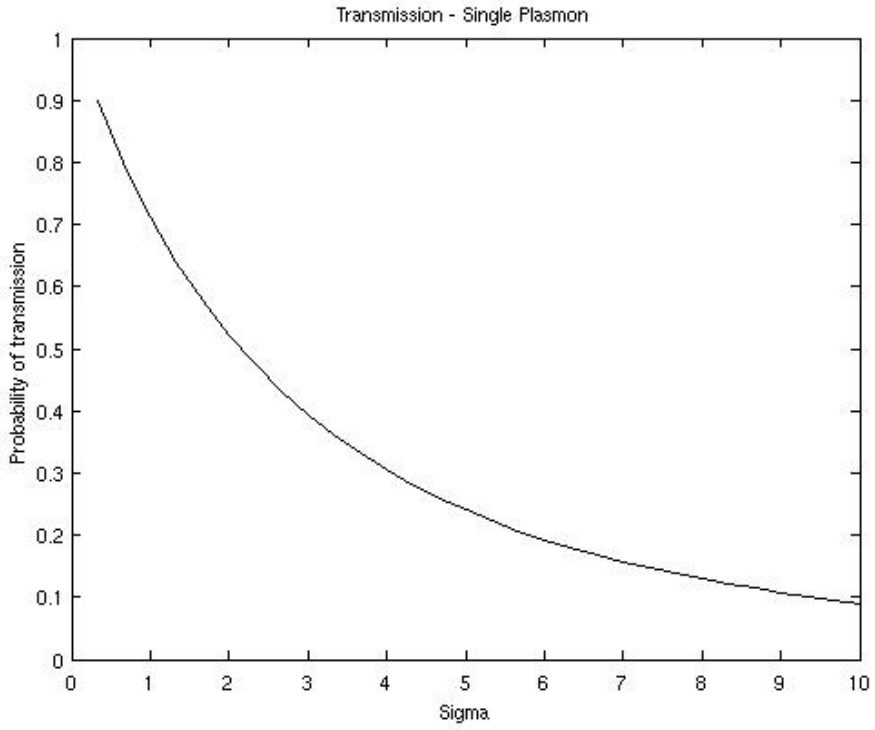

Figure 5.6 - The relation between the width of the incoming wavepacket versus the likeliness of full transmission of the plasmon for states containing a single plasmon.

The shape of this graph is pretty wellknown and expected. We obviously want the transmission to be low, when the Gaussian is wide (in positionspace), as this will result in a narrow Gaussian in wavevectorspace and therefore a high chance of interaction and hence reflection.

6 Multiple Plasmons

6.1 Multimode multiplasmon model

In this section we are going to derive a model describing a system consisting of an atom as before and many plasmons travelling along a wire.
We adopt the same sort of Hamiltonian as in the previous section. But as one might suspect this one will be more detailed and complex than the previous one. $\hat{H}_{atom}$ remains unchanged, but our $\hat{H}_{field}$ now contains a term for each plasmon. The real challenge is to construct $\hat{H}_{interaction}$, which should now contain every possible way the plasmons can interact with our atom.
The generalized Hamiltonian for a system containing a multimode multiphoton field and a twolevel

atom - a more general descibtion can be found in [8]:

$$\begin{aligned}\hat{H} &= \frac{1}{2}\hbar\omega_0\left(\sigma_{aa}-\sigma_{bb}\right)+\sum_k \hbar\omega_k\left(\hat{a}_{k,R}\hat{a}^\dagger_{k,R}+\hat{a}_{k,L}\hat{a}^\dagger_{k,L}+\frac{1}{2}\right)+\\ &\sum_k \hbar g_k\left(\sigma_+\prod_k\left(\hat{a}_{k,R}\right)^{n_R}\left(\hat{a}^\dagger_{k,R}\right)^{m_R}+\sigma_-\prod_k\left(\hat{a}_{k,R}\right)^{m_R}\left(\hat{a}^\dagger_{k,R}\right)^{n_R}\right)+\\ &\sum_k \hbar g_k\left(\sigma_+\prod_k\left(\hat{a}_{k,L}\right)^{n_L}\left(\hat{a}^\dagger_{k,L}\right)^{m_L}+\sigma_-\prod_k\left(\hat{a}_{k,L}\right)^{m_L}\left(\hat{a}^\dagger_{k,L}\right)^{n_L}\right)\end{aligned}$$

where n is the number of photon annilihated when the atom is exited, m the number created when relaxed.
It should be emphasized that the Hamiltonian is strongly nonlinear.
This Hamiltonian appears to couple the desired states (this becomes excedingly clear, if we choose the number of plasmons involved to be 1 or 2) and furthermore contains terms familiar to the singleplasmon Hamiltonian, which will make the derivations a bit easier.

6.2 Applying the model

In order to use the timedependent Schrödingerequation we will need the timeevolved version of the interaction part of out Hamiltonian. This is, however, easier said than done.
We suspect something along the lines of the timeevolved interaction operator from the singleplasmon model, but since we now have the factors like $\left(\hat{a}_{k,R}\right)^{n_R}$ and $\left(\hat{a}^\dagger_{k,L}\right)^{m_L}$, our timeevolution turns out to go less smooth. We end up with[6]:

$$\begin{aligned}\hat{V}=\hat{U}(t)\,\hat{H}_{int} &= \sum_k \hbar g_k\left(\sigma_+ e^{-i\omega_0 t}\prod_k\left(\hat{a}_{k,R}\right)^{n_R}\left(\hat{a}^\dagger_{k,R}\right)^{m_R}e^{i\omega_k(n_R-m_R)t}+\right.\\ &\qquad\left.\sigma_- e^{i\omega_0 t}\prod_k\left(\hat{a}_{k,R}\right)^{m_R}\left(\hat{a}^\dagger_{k,R}\right)^{n_R}e^{-i\omega_k(n_R-m_R)t}\right)+\\ &\sum_k \hbar g_k\left(\sigma_+ e^{-i\omega_0 t}\prod_k\left(\hat{a}_{k,L}\right)^{n_L}\left(\hat{a}^\dagger_{k,L}\right)^{m_L}e^{-i\omega_k(n_L-m_L)t}+\right.\\ &\qquad\left.\sigma_- e^{i\omega_0 t}\prod_k\left(\hat{a}_{k,L}\right)^{m_L}\left(\hat{a}^\dagger_{k,L}\right)^{n_L}e^{i\omega_k(n_L-m_L)t}\right)\end{aligned}$$

This operator should be ready for the Scrödingerequation, so without further delay, we use our two components (timeevolved interaction-part of the Hamiltonian and a state consisting of an atom and two travelling wavepackets, one from the left and one from the right).
At this point we are faced with the limitations of the analytical way of solving these problems. The next step will quite simply be very, very tedious and very, very long.
Writing down the Schrödinger equations (and even worse: the solutions to these) will take forever, as we have to be aware of the many different potential detunings and combinations of wavevectors. A singlemode field is just not possible in the same way as in the single plasmon case.
In other words doing this by hand/analytically is a bad idea. We therefore skip directly to the MatLab-based way of solving this, as we suspect the numerical method to be the "right" way of doing things.
We will, however, be using the Hamiltonian above as an important tool during the calculations.

[6]See Appendix B for detailed derivation of the timeevolution

7 Calculations - Multiple Plasmons

7.1 Constructing the algorithm

We will adopt a discrete subspace of wavenumbers, n, spanding from 1 to n_{max}.
The abundance of different states will then be:

Leftpropagating plasmons	Rightpropagating plasmons	State of the atom	Abundance
1	0	Excited	n_{max}
0	1	Excited	n_{max}
2	0	Ground	$\frac{1}{2}n_{max}(n_{max}+1)$
0	2	Ground	$\frac{1}{2}n_{max}(n_{max}+1)$
1	1	Ground	n_{max}^2

Or $3n_{max} + 2n_{max}^2$ in total; we call this number n_{total}.
Please notice that the case where two leftmoving plasmons has $k_1 = 4$ and $k_2 = 7$ is not any different from the state, where $k_1 = 7$ and $k_2 = 4$.
The number of different states is then:

$$N_{\text{States with two leftmoving plasmons}} = \sum_{i=1}^{n_{max}} i = \frac{n_{max}^2 + n_{max}}{2}$$

The same is obviously the case for two rightmoving plasmons. It is, however, not the case, when the plasmons are moving in opposite directions, as we will be able to tell plasmon 1 apart from plasmon 2.
Once again we will make use of the following abbreviations and properties:

$$\Delta\omega = \frac{\hbar c k_n}{g} \qquad gT = \frac{2\pi}{\Delta\omega}$$

In other words we will aim our calculations at the same timeframe as in the previous chapters (concerning the systems with one plasmon). $\Delta\omega$ is the same parameter as before and has the same physical meaning. Combined with the knowledge of the width of the wave, these two cover the information about the size of the interaction.

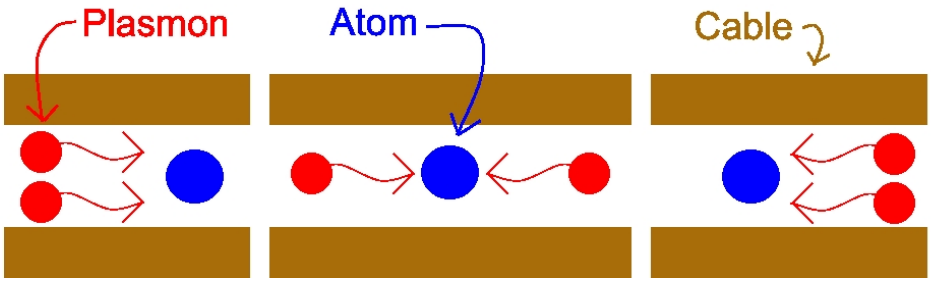

Figure 7.1 - Three of the five different kinds of possible states in the dualplasmon model; each of the plasmons is contained in a discrete wavespace with n_{max} different possibilies. The other two consist of the atom in an excited state and a plasmon travelling either left of right.

In order to create the matrix $\hat{H}$ to suit our needs, we will need it to be of the dimension $n_{total} \times n_{total}$. This way we can construct it in such a way that we can connect every pair of coupled states with a coupling constant in the offdiagonal elements corresponding to the transistion in statespace.
That being said we plan on containing combined wavenumbers of the n'th state in the (n, n) element.

The generalized version of $\hat{H}$ for the multiple plasmon is:

$$\hat{H} = \begin{pmatrix} D_1 & 0 & C_1 & 0 & C_3 \\ 0 & D_2 & 0 & C_2 & C_4 \\ C_1^\dagger & 0 & D_3 & 0 & 0 \\ 0 & C_2^\dagger & 0 & D_4 & 0 \\ C_3^\dagger & C_4^\dagger & 0 & 0 & D_5 \end{pmatrix}$$

The submatrices C_n and $C_n^\dagger$ contain the information of the different possible couplings. The elements of these matrices will be either 0, if no coupling, 1, if a normal couping, and $\sqrt{2}$, if there is a coupling based on stimulated emission.
The C_1- and C_2-matrices have the dimensions $(\frac{1}{2}n_{max}(n_{max}+1)) \times (n_{max})$. The C_3 and C_4 are $(n_{max}^2) \times (n_{max})$-matrices
D_n represents a diagonal matrix with elements corresponding to the energy in the field relative to the resonance of the atom. D_1 and D_2 have dimension $(n_{max}) \times (n_{max})$, D_3 and D_4 $(\frac{1}{2}n_{max}(n_{max}+1)) \times (\frac{1}{2}n_{max}(n_{max}+1))$, and D_5 is $(n_{max}^2) \times (n_{max}^2)$.
And our MatLab-scripts can be found in Appendix D.
The mathematical form of the matrix will be more or less the same as the previous case, which is:

$$\frac{\hat{H}}{g} = \sum_k \sum_{i=L,R} \left(\sigma_- \hat{a}_{k,i}^\dagger + \sigma_+ \hat{a}_{k,i} \right) + \sum_n \sum_{j=L,R} \frac{\hbar c k_n}{g} \hat{a}_{n,j}^\dagger \hat{a}_{n,j}$$

When using this matrix in our Schrödingerequation, we obtain a huge number of differential equations (an equation for the timeevolution of every single state). When we set $n_{max} = 51$, we are indirectly asking MatLab to solve more than 5000 coupled differential equations. These calculations has taken a decent amount of time to perform.

7.2 Initial conditions

As done earlier for single-plasmon-systems, we will need to consider our initial states the same way we did in the chapter for single plasmons. Once again we will be sending Gaussian wavepackets at the atom, however, this time two plasmons will be in the state of the Gaussian beam, not one. This will give rise to a number of necessary precautions, as we will now have to be more careful with the normalization and our statespace.
First of all we need to be sure that when the two wavenumbers of the two plasmons are the same, we will need to adjust by multiplying by $\sqrt{2}$. Furthermore the very enumeration of the states is far more complicated than in the singleplasmon case - we will need an elaborate system to convert our Gaussian in positionspace to a distribution in wavevectorspace. Not just the Fouriertransform as earlier.
Once again we need a creationoperator (but only one, since we will be sending two plasmons in the same state against the atom).
We want our operator to have the property:

$$\left(\sum_k \Phi_k \hat{a}_k^\dagger \right)^2 |0\rangle = \hat{a}_\Phi^{\dagger 2} |0\rangle = |2_\Phi\rangle$$

Now things turn a bit more complicated this time, as we will run into trouble, whenever the two plasmons have the same k.
The actual expression of the resulting state would be:

$$|2_\Phi\rangle = \sum_{k_1 > k_2} \sqrt{2} \Phi_{k_1} \Phi_{k_2} |1_{k_1 k_2}\rangle + \sum_{k_1 = k_2} \Phi_{k_1}^2 |1_{k_1 k_2}\rangle$$

And once again we will be able to write our as a sum of the complete set of wavevectors:

$$|2_\Phi\rangle = \sum_k c_{k_1 k_2} |1_{k_1 k_2}\rangle$$

These coefficients will be used as elements in the vector, we will use in our Schrödingerequation.
One more note on the initial states needs to be made. The enumeration of our states also plays a bigger part in this chapter. In the singleplasmon-model we had the luck, that the number of a state represented the energy relative to the resonant state; i.e. in state number 30 the plasmon has less energy than state number 42.
This is not the case now. First of all we have two parameters representing the energy instead of one.
We need to design a way of representing a two-parameter-state with only one parameter[7].
The best way of describing this is done graphically. Please notice that $k_1 > k_2$ at all times, since we would otherwise be counting some states twice. This way of enumerating the states is only used for states, where both plasmons are propagating in the same direction.

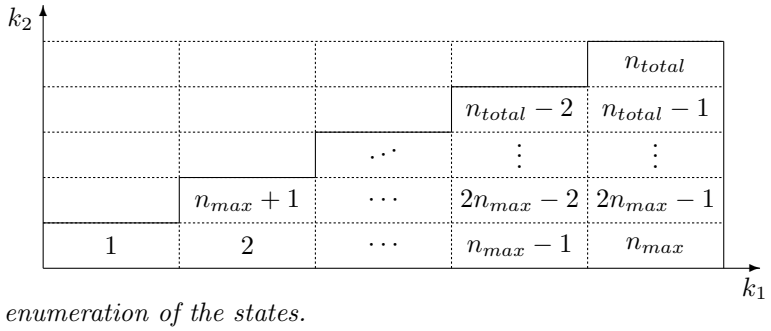

Figure 7.2 - Our enumeration of the states.

Combining the way of enumerating our states with the expression for $|2_\Phi\rangle$, we can now construct the desired initial state.
Notice, that the initial state in positionspace is equal to the initial state of the singleplasmon model. All we really do is adding another plasmon to the same state. This is one of the reasons, why we have not made another "movieclip" of the events in positionspace. It would not be much different from the one, we have already made.

7.3 Results

Once again we turn our attention to graphing the probabilities for having our system in different states.
As one might have guessed, we need more graphs, as we have more states than before. These are, however, best explained by the graph itself.
The following graph is made by setting $n_{max} = 51$, $\sigma = 2$, $\Delta\omega = 2$, and $\mu = 0$:

[7]The program Integerfromarray.m does this in our script. The code can be found in Appendix D.

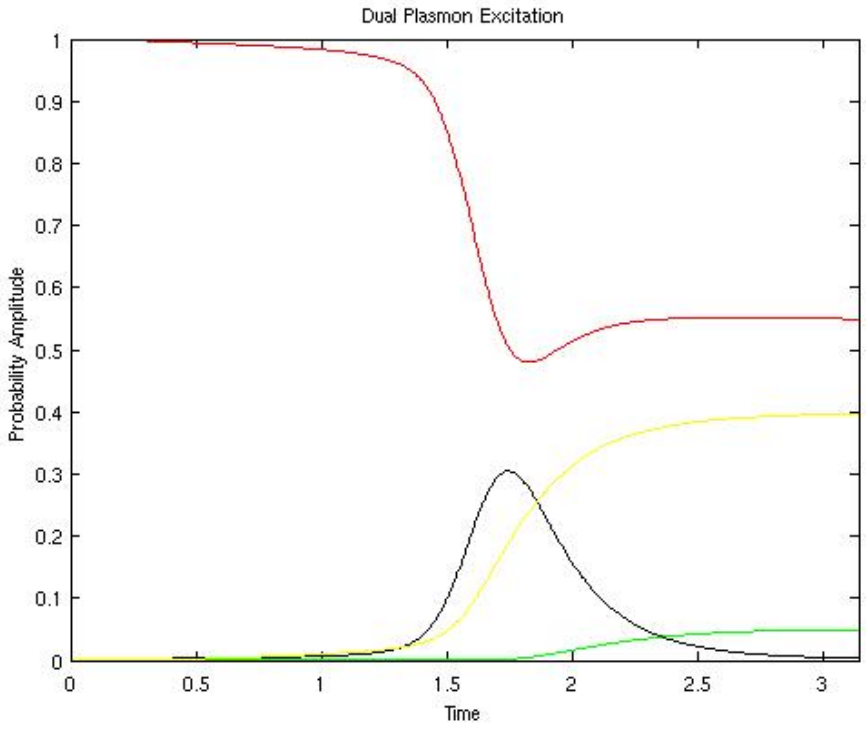

Figure 7.3 - The black graph represents the probability of having atomic excitation. Notice, that it peaks, when the center of the wave crosses the atom. The yellow graph contains the information of the likeliness that one plasmon is travelling to the left and the other one propagating right. The red graph is the probability of having two plasmons propagating left and the green graph represent the same for rightmoving plasmons.

This figure tells us, what we already know. A reflected plasmon is clearly a plasmon, that has interacted with the atom and has been emitted back in the direction from which it came. The complicated part of this graph is, that both plasmons can find the time to interact with the atom (the green graph is above zero at some point - this will become exceedingly clear later on).
In this case the probability of having "full transmission" is approximately 55% (the last points on the red graph).
As we did in the previous resultsection, we have also constructed a graph of the probability of transmission of both plasmon versus the width of the incoming wave.

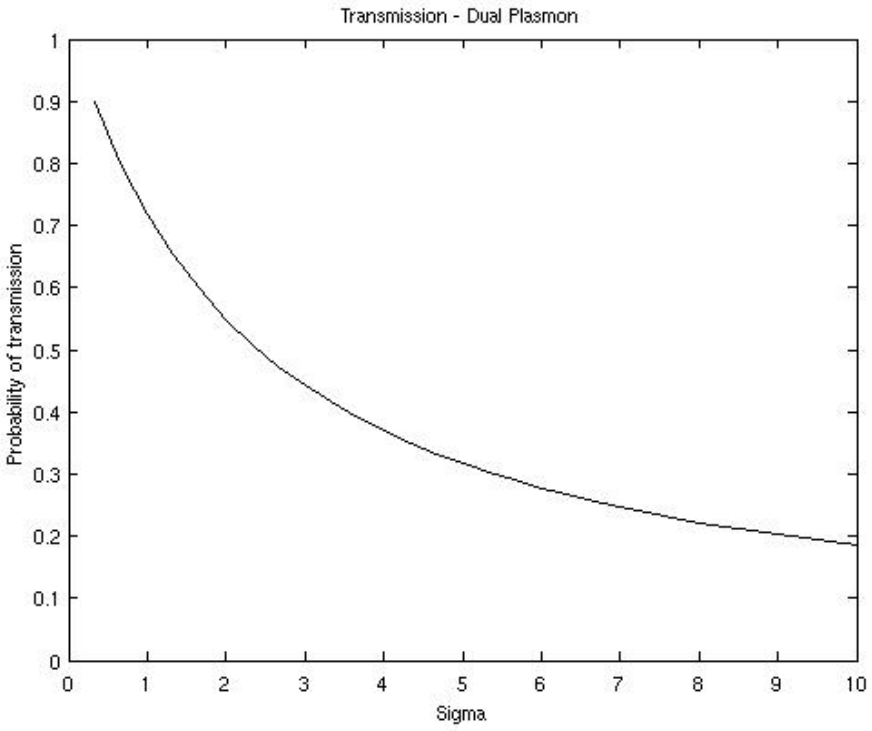

Figure 7.4 - A $(\sigma, P\,(\textit{Full transmission}))$-graph of states containing two plasmons.

Once again the shape of this graph is highly anticipated for exactly the same reasons as before.
We could have made a similar cartoonlike figure, but chose not to. An unfortunate feature of the states in the dualplasmonmodel is, that where the states for single plasmons moved in a direction that could be seen directly from the states position in the statevector, this is not the case for multiple plasmon. The states, where the plasmons travel in different directions can not be Fouriertransformed by MatLab. We would have to do the Fouriertransformations manually.
So even though creating a movie is not impossible at all, it would have been rather timeconsuming and have lots of lots of tricky calculations.
We chose to prioritize the actual results above this.
This concludes our examinations of the dualplasmonsystem. We now turn our attention towards comparing the singleplasmon scheme and the dualplasmon scheme.

8 Comparing the results

We have constructed two graphs, one for the singleplasmon case and one for the dualplasmon case. The graphs has been created by varying the width of the Gaussian (known as σ) and measuring the total probability for being in a state, where the directions of the velocity of the plasmons are preserved.
And now for the final result of the project:

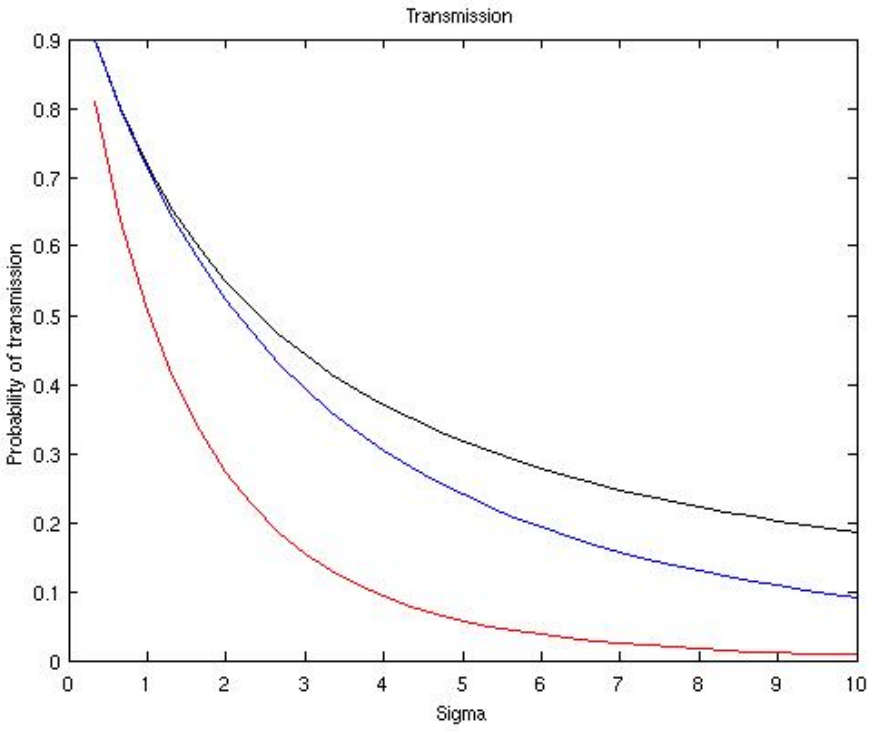

Figure 8.1 - The black graph is the same as the one in figure 7.4. The blue graph is the values of the graph in figure 5.6. The red is simply the blue graph squared. Please notice that the black and the blue graph are very close to each other for small values of σ. They do, however, not cross!

The most important part of the picture above is to nice that the values of the black graph is (way) above both the other graph.
As mentioned before one would expect the probability of full transmission of two plasmons to be the same as the probability of full transmission of one plasmon squared, if we were dealing with linear optics.
A quick look on the graphs tells us that this is not the case. Not even close to being the case actually. Whereas the red graph is close to zero as about $\sigma = 10$, the black one is approximately 0.2. This is a remarkable difference in the model constructed from linear optics and the actual results of the simulation.
The first and most important conclusion to draw here is simply that the fact that more plasmons are present, limits the likeliness that the plasmons are actually capable of interacting with the atom (please notice, that (Probability of full transmission) $\propto$ (Interaction with the atom)$^{-1}$). Another potential (although rather farfetched and "unphysical") model could be that one plasmon would interact, while the other one would pass straight through as though the atom was were there.
If this were the case the blue and the black graph would be equal each other. This is obviously not the case.
In other words we have "solved" the problem presented in figure 3.2. We are dealing with mechanics more complicated than traditional optics. There is quite simply a more complicated effect taking place during the interaction. It should be pretty clear that we left the realm of linear optics the second, we turned our attention to the dualplasmon system.
The cause of the effect is a harder to figure out, than to actually show the existence of such. Intuitively speaking, one imagines, that both plasmons interact a little with the atom, and therefore none of them makes an actual "real" photoelectric interaction. In a very vague and unclear sense the plasmons are blocking the interaction of each other.
Finally we have been asked to plot our graph with respect to g. The relation between the units

has appeared to be:

$$gT = \frac{2\pi}{\Delta\omega}$$

Using $c = 1$ and $\Delta\omega = 2$, we can obtain an expression between distances in positionspace and the corresponding coupling constants:

$$x = \frac{\pi}{g}$$

As σ is a parameter existing in positionspace, we can now construct the following graph:

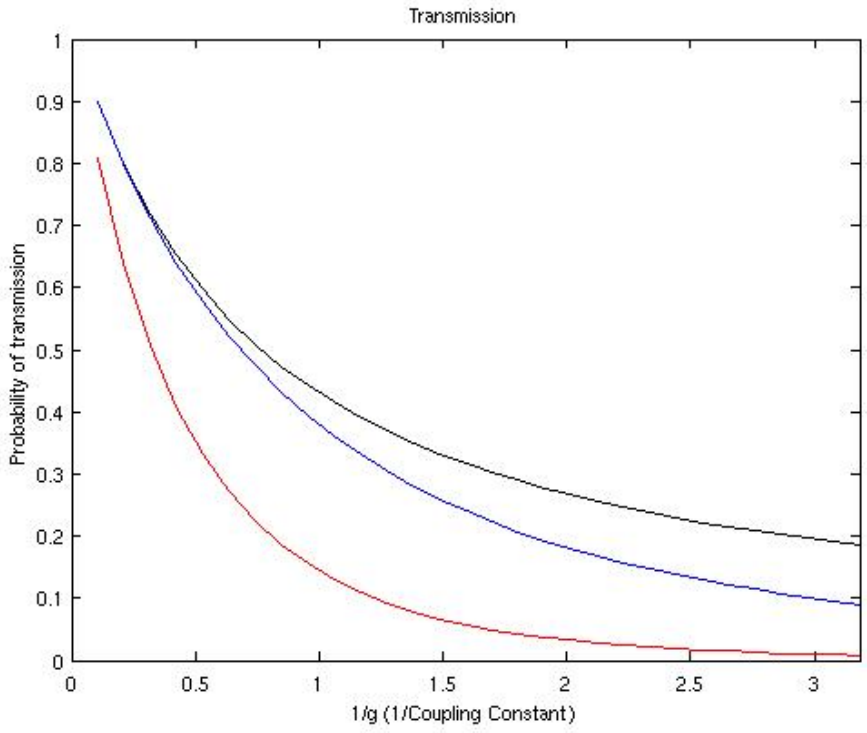

Figure 8.2 - The values from figure 8.1 plotted versus the width of the Gaussian in relation to the coupling constant, g. This is probably a more correct manner of displaying our results, as the parameter σ is a rather ambiguous quantity. It is highly dependant on n_{max} and the interpretation is closely related to $\Delta\omega$. The factor, g, is a simpler and more "physical" quantity.

There are no new conclusions to be drawn here (all we really did was rescaling the categoryaxis). It is merely the "right" way of doing things, as we have now taken units a bit back into consideration.

9 Conclusion

The synopsis of this project was investigating the nonlinear effects of a system described so many times in this report.
Through simulations of these systems we have (in our own opinion) aquired results showing that clearly sticking to linear optics simply will not suffice in these cases.
And even though the effects caused by this nonlinearity are too complicated to describe and understand yet, our simulations show that they are unquestionably there.
The project has gone relatively smooth. The biggest troublemaker during the process has undoubtedly been MatLab and MatLab's unpleasant syntax. The actual physics behind the issues

at hand are not too tricky (the linear ones, that is) to understand and once one is familiar with the Hamiltonians and the Schrödingerequation, the dynamics of the system are understood quite easily.
That being said, the obtained results are, as mentioned, far from fully explained.
Most of our time has gone by inventing, constructing, and interpreting the algorithms found in Appendix C and D. Countless failed attempts have been made to construct usable initial conditions and Hamiltonians. Every failed process did leave behind a little more understanding of the numerical method we (and MatLab) were using.
This is an obvious advantage in this project. We were never in doubt, whether a result was close to being correct or completely wrong, the graph would simply be physically uninterpretable (i.e. the wave interact instantly/constantly with the atom or perhaps not at all). This was very helpful, as debugging our scripts was necessary.
The results themselves seem to speak a bit for themselves. Some features in our graphs are easy to interpret (i.e. the wave hits the atom at the right time or the probabilies for reflection differing from zero at the correct parts of the graph). Other part of our results, this would be explaining the events or effects caused by nonlinearity, are simply close to nondoable.
Nonetheless, we feel as if our goals have been fulfilled and that our initial questions have been answered. We found (more or less expected) results that seems to be quite a good fit in the nonlinear model.
In any way it has been an interesting assignment and a challenging procedure to work with.

10 Appendix

10.1 Appendix A

Our operators and their applications
In our basic treatment of the atom and the field we can simplify the formulas and physical interpretation by using operators to describe the dynamics of the system. The state of the atom is descibed by our raising and lowering operators, σ_+ and σ_-.
In standard Dirac-notation, these are given by $|a\rangle\langle b|$ and $|b\rangle\langle a|$ respectively, where $|a\rangle$ and $|b\rangle$ represent the exited and the ground state of an atom.
The effect of these operators become clear, when we let them operate on the eigenstates of an atom:

$$\sigma_+ |b\rangle = |a\rangle \langle b| \; b\rangle = |a\rangle \qquad \sigma_- |a\rangle = |b\rangle \langle a| \; a\rangle = |b\rangle$$

As one might have suspected, the raising operator excites the atom, while the lowering operator relaxes it.
The field is described by the creation- and the annihilationoperator, $\hat{a}_k^\dagger$ and $\hat{a}_k$. Using traditional canonical variables these can be written as (can be found in [9] and [7]):

$$\hat{a}_k = \frac{1}{\sqrt{2m_k\hbar\omega_k}} (m_k\omega_k\hat{x}_k + i\hat{p}_k) \qquad \hat{a}_k^\dagger = \frac{1}{\sqrt{2m_k\hbar\omega_k}} (m_k\omega_k\hat{x}_k - i\hat{p}_k)$$

In textbook-examples these operators are meant to work on a harmonic oscillator with discrete modes. In this project we use them to treat the socalled Fock-states, where $|n\rangle$ represents a state with n photons (in a cavity).
The effect of these operators are best viewed, when they are applied to a Fock-state:

$$\hat{a}_k |n\rangle = \sqrt{n} \, |n-1\rangle \qquad \hat{a}_k^\dagger |n\rangle = \sqrt{n+1} \, |n+1\rangle$$

As we can see, there is a reason for the names of these operators.
One can obtain another important operator using these two:

$$\hat{n}_k = \hat{a}_k^\dagger \hat{a}_k$$

The effect of this operator is pretty simple:

$$\hat{n}_k |n_k\rangle = n_k |n_k\rangle$$

The operator simply tells us, how many particles can be found in the k'th mode.
This adds a minor interpretation to the Hamiltonian of a quantummechanical electromagnetic field, as it becomes clear that all it really means is:

$$E_{total} = \sum_k n_k \cdot \hbar\omega_k$$

10.2 Appendix B

Timeevolution of our operators
During our derivations in the theoretical parts of the project, we often use timeevolved operators, namely the annihilation- and the creationoperator ($\hat{a}_k$ and $\hat{a}_k^\dagger$ respectively) for the field and the raising and lowering operator (σ_+ and σ_- respectively) for the atom.
When timeevolving these operators, we apply a time-evolution operator, which is described in [9]. The effect this operator will have on our four operators can easily be computed:

$$\begin{aligned}
\hat{U}(t)\,\sigma_+ &= \sigma_+ e^{i\omega_0 t} \\
\hat{U}(t)\,\sigma_- &= \sigma_- e^{-i\omega_0 t} \\
\hat{U}(t)\,\hat{a}_k &= \hat{a}_k e^{i\omega_k t} \\
\hat{U}(t)\,\hat{a}_k^\dagger &= \hat{a}_k^\dagger e^{-i\omega_k t}
\end{aligned}$$

These timeevolutions play an important role in our way of proceeding through our calculations, as these make our use of the timedependant Scrödingerequation possible.
Another (and perhaps more correct) way of deriving these expressions would be to use Heisenberg's commutatorrelations.
This would hopefully yield the same results as above, so we have not performed the calculations; it is the traditional way of obtaining the timeevolution of an operator, though.

10.3 Appendix C

Single plasmon MatLab scripts
The scripts written below can be found at http://www.fys.ku.dk/∼mcp/MatLab/SinglePlasmon. Our program consists of the following scripts. In this case we have chosen $n_{max} = 51$, $\Delta\omega = 2$, and we solve for $t \in [0, \pi]$.
Our initial conditions consist of a leftpropagating Gaussian wave with parameters $\sigma = 2$ and $\mu = -10$.
The main MatLab-script of our program is:

```
clear;
clc;

%% Inputs
global nmax
nmax = 51;

global sigma
sigma = 2;

global mu
mu = -10;

global Delta
Delta = 2;

%% Program
global ti
ti = 0;
global tf
tf = 2*3.14169/Delta;
global N
N = 2*nmax + 1;

Hamiltonian;
disp('Hamiltonian constructed...')

InitialLeft;
% InitialRight;
disp('Initial conditions constructed...')

global psi
global T
[T,psi] = ode45(@Schr,[ti tf], psiini);
disp('Schroedingerequation solved...')

%% Outputs
ProbabilityAmplitude
% Movie
```

First things first. The lines in the beginning are used to specify the different quantities needed to perform the calculation.

After this a Hamiltonian is constructed using the following string:

```
global Delta
global nmax
global N

D = sparse(N,N);
V = D;

for j = 2:1:nmax+1;
    aarray = Arrayfrominteger(j);
    D(j,j) = aarray(2)-(nmax+1)/2;
end

for j = nmax+2:1:N
    aarray = Arrayfrominteger(j);
    D(j,j) = aarray(3)-(nmax+1)/2;
end

for j = 1
    aarray = Arrayfrominteger(j);
    if aarray(1)==1
        for k = 1:1:nmax
            barray = [0,k,0];
            targetone = Integerfromarray(barray);
            V(targetone,j) = 1;
        end
        for l = 1:1:nmax
            barray = [0,0,l];
            targetone = Integerfromarray(barray);
            V(targetone,j) = 1;
        end
    end
end

D(1,1) = 0;

global H
H = V + V' + Delta*D;
```

The two function, `Integerfromarray` and `Arrayfrominteger`, are supportfunctions, enumerating the different states. They are obviously each others inverse function.

`IntegerFromArray` is:

```
function f = Integerfromarray(barray);

global nmax
global N

if barray(1) == 0;
    if barray(3) == 0;
        f = barray(2)+1;
    else
        f = nmax+1+barray(3);
    end
else
    f = 1;
end
```

And the `ArrayFromInteger`-function consists of the following m-file:

```
function f = Arrayfrominteger(j);

global nmax
global N

if j == 1;
    f = [1,0,0];
else
    if j <= nmax+1;
        f = [0,j-1,0];
    else
        f = [0,0,j-(nmax+1)];
    end
end
```

Our Main.m-program now constructs the initial conditions (a Gaussian wave) using either the InitialRight.m-file or the InitialLeft.m-file. These are almost identical except for the obvious fact that InitialLeft constructs a leftpropagating wave and InitialRight constructs a rightpropagating.

InitialLeft.m looks like the following in textformat:

```
global nmax
global sigma
global mu
global N

for m = 1:1:nmax
    l(m) = 0;
    r(m) = 0;
end

%% Leftpropagating Gaussian wave
for m = 1:1:nmax
    l(m) = 1/(sigma*sqrt(2*3.14159))*exp(-((m-(nmax+1)/2)-mu)^2/(2*sigma^2));
end

lf = fft(l);
lfs = fftshift(lf);

global psiini
psiini = [0 lfs r];
```

The function forces a Fouriertransformed Gaussian into the leftpropagating part of a vector later to be used in the Schrödinger equation.
And now for the core of the program - solving the differential equations - using `ode45`.
The function `@Schr` is given by:

```
function dpsidt = Schr(t,psi);
global H

dpsidt = -i*H*psi;
```

We have constructed our program in such a way that it is capable of making two different kinds of outputs, one being a graph of the likeliness of being in one kind of state or another and the other being a smaller movie of the whereabouts of our Gaussian wave and the resulting reflected and transmitted wave.
The graph is constructed by the following script - called ProbabilityAmplitude.m:

```
global psi
global T
global ti
global tf
global nmax
global N

for m = 1:1:N;
    for n = 1:1:length(psi(:,1));
        psi2(n,m) = abs(psi(n,m))*abs(psi(n,m));
    end
end
```

```
for m = 1:1:N;
    for n = 1:1:length(psi(:,1));
        A(n) = 1/sum(psi2(n,:));
        psi2n(n,m) = A(n)*psi2(n,m);
    end
end
disp('Preparations for plotting complete...')

%% Probability of atomic excitation
for m = 1:1:length(psi(:,1));
    psi2e(m) = psi2n(m,1);
end

plot(T,psi2e,'k')
hold on

%% Probability of having a leftgoing plasmon
for m = 2:1:nmax+1
    psi2left(:,m-1) = psi2n(:,m);
end

for m = 1:1:length(psi(:,1));
    psi2l(m) = sum(psi2left(m,:));

end

plot(T,psi2l,'r')
hold on

%% Probability of having a rightgoing plasmon
for m = nmax+2:1:N
    psi2right(:,m-nmax-1) = psi2n(:,m);
end

for m = 1:1:length(psi(:,1));
    psi2r(m) = sum(psi2right(m,:));
end

plot(T,psi2r,'g')

%% All at once
axis([ti tf 0 1])
title('Single plasmon excitation');
xlabel('Time');
ylabel('Probability amplitude');
disp('Plotting complete...')
```

The program plots the normalized square of the outputs from the solutions of the differential equations (keep in mind that (probability)$\propto |\psi|^2$).

The movie is made using these lines - referred to as Movie.m:

```
global psi
global T
global ti
global tf
global nmax
global N

%% Calculations
for m = 1:1:nmax
    x(m) = m-(nmax+1)/2;
end

for m = 1:1:length(T)
        gr(m) = psi(m,1);
    for n = 2:1:nmax+1
        left(m,n-1) = psi(m,n);
    end
    for n = nmax+2:1:N
        right(m,n-nmax+1) = psi(m,n);
    end
end

for m = 1:1:length(T)
    lefts(m,:) = fftshift(left(m,:));
    rights(m,:) = fftshift(right(m,:));
    le(m,:) = fft(lefts(m,:));
    ri(m,:) = conj(fft(conj(rights(m,:))));
end

for m = 1:1:nmax;
    for n = 1:1:length(T);
        gr2(n) = abs(gr(n))^2;
        le2(n,m) = abs(le(n,m))^2;
        ri2(n,m) = abs(ri(n,m))^2;
    end
end

for m = 1:1:length(T)
    sumx2(m,:) = le2(m,:)+ri2(m,:)+gr2(m);
end

for m = 1:1:nmax;
    for n = 1:1:length(T);
        A(n) = 1/sum(sumx2(n,:));
        le2n(n,m) = A(n)*le2(n,m);
        ri2n(n,m) = A(n)*ri2(n,m);
    end
end
```

```
for m = 1:1:length(T)
    le2ns(m,:) = fftshift(le2n(m,:));
    ri2ns(m,:) = fftshift(ri2n(m,:));
end
disp('Preparations for plotting complete...')

%% Plot
for m = 1:10:length(T)
    plot(0,0.009,'bo')
    hold all
    plot(x,ri2ns(m,:),'g')
    hold all
    plot(x,le2ns(m,:),'r')
    axis([-(nmax+1)/2 (nmax+1)/2 0 0.5])
    xlabel('x')
    ylabel('Fieldstrength')
    pause(0.5)
    hold off
end
disp('Plotting complete...')
```

This script is a bit trickier.
First of all we have to transform back to positionspace (this is done relatively easy using `FFT` and `FFTSHIFT`. After that we plot the same normalized probabilities (they differ greatly from the ones in ProbabilityAmplitude, as these are in an entirely different Hilbertspace).

10.4 Appendix D

Multiple plasmons MatLab scripts
The follwing scripts can be found at http://www.fys.ku.dk/∼mcp/MatLab/MultiplePlasmon.
The structure of the dualplasmon simulation is much like the one of the single plasmon simulation. The different scripts and functions fulfill the same task as the one by the same name in the other simulation.
In this case we have chosen $n_{max} = 51$, $\Delta\omega = 2$, such that we solve for $t \in [0, \pi]$.
Our initial conditions consist of a leftpropagating Gaussian wave with parameters $\sigma = 2$ and $\mu = 0$.
The main MatLab-script of our program is:

```
clear;
clc;

%% Inputs
global nmax
nmax = 11;

global sigma
sigma = 2;

global mu
mu = 0;

global Delta
Delta = 2;

%% Program
global ti
ti = 0;
global tf
tf = 2*3.14169/Delta;
global N
N = 2*nmax + (nmax*(nmax+1)) + nmax^2;

Hamiltonian;
disp('Hamiltonian constructed...')

InitialLeft;
% InitialRight;
disp('Initial conditions constructed...')

global psi
global T
[T,psi] = ode45(@Schr,[ti tf], psiini);
disp('Schroedingerequation solved...')

%% Outputs
ProbabilityAmplitude
```

The string used to construct the Hamiltonian is (not surprisingly) a lot more complicated than the previous.
First of all the part of the matrix that describes the coupling is no longer filled with either zeroes or ones, but also with $\sqrt{2}$. Furthermore, the actual number of nonzero elements is much, much

larger than before.

```
global nmax
global N
global Delta

D = sparse(N,N);
V = D;

for m = 1:1:nmax;
    aarray = Arrayfrominteger(m);
    D(m,m) = aarray(2)-(nmax+1)/2;
    if aarray(1) == 1
        for n = 1:1:nmax
            if m == n
                barray = [1 m 0 0 0];
                darray = [0 m 0 n 0];
                targetone = Integerfromarray(barray);
                targetthree = Integerfromarray(darray);
                V(targetone,targetthree) = 1;
            else
                barray = [1 m 0 0 0];
                darray = [0 m 0 n 0];
                targetone = Integerfromarray(barray);
                targetthree = Integerfromarray(darray);
                V(targetone,targetthree) = 1;
            end
        end
        for n = 1:1:m
            if m == n
                barray = [1 m 0 0 0];
                carray = [0 m n 0 0];
                targetone = Integerfromarray(barray);
                targettwo = Integerfromarray(carray);
                V(targetone,targettwo) = sqrt(2);
            else
                barray = [1 m 0 0 0];
                carray = [0 m n 0 0];
                targetone = Integerfromarray(barray);
                targettwo = Integerfromarray(carray);
                V(targetone,targettwo) = 1;
            end
        end
    end
end
```

```
for m = nmax+1:1:2*nmax;
aarray = Arrayfrominteger(m);
D(m,m) = aarray(4)-(nmax+1)/2;
if aarray(1) == 1
        for n = 1:1:nmax
            if m-nmax == n
                barray = [1 0 0 m-nmax 0];
                carray = [0 n 0 m-nmax 0];
                targetone = Integerfromarray(barray);
                targettwo = Integerfromarray(carray);
                V(targetone,targettwo) = 1;
            else
                barray = [1 0 0 m-nmax 0];
                carray = [0 n 0 m-nmax 0];
                targetone = Integerfromarray(barray);
                targettwo = Integerfromarray(carray);
                V(targetone,targettwo) = 1;
            end
        end
        for n = 1:1:m-nmax
            if m-nmax == n
                barray = [1 0 0 m-nmax 0];
                darray = [0 0 0 m-nmax n];
                targetone = Integerfromarray(barray);
                targetthree = Integerfromarray(darray);
                V(targetone,targetthree) = sqrt(2);
            else
                barray = [1 0 0 m-nmax 0];
                darray = [0 0 0 m-nmax n];
                targetone = Integerfromarray(barray);
                targetthree = Integerfromarray(darray);
                V(targetone,targetthree) = 1;
            end
        end
    end
end

for m = 2*nmax+1:1:2*nmax+nmax*(nmax+1)/2;
    aarray=Arrayfrominteger(m);
    D(m,m) = (aarray(2)+aarray(3))-(2*nmax+2)/2;
end

for m = 2*nmax+nmax*(nmax+1)/2+1:1:2*nmax+nmax*(nmax+1);
    aarray=Arrayfrominteger(m);
    D(m,m) = (aarray(4)+aarray(5))-(2*nmax+2)/2;
end

for m = 2*nmax+nmax*(nmax+1)+1:1:2*nmax+nmax^2+nmax*(nmax+1);
    aarray=Arrayfrominteger(m);
    D(m,m) = (aarray(2)+aarray(4))-(2*nmax+2)/2;
end

global H
H = V + V' + Delta*D;
```

As one might suspect, the support functions will also turn out to be far more complicated than the ones keeping track of the singleplasmon states.
`Integerfromarray` is:

```
function f = Integerfromarray(barray);
global nmax
global N

if barray(1) == 1;
    if barray(2) == 0;
        f = nmax + barray(4);
    else
        f = barray(2);
    end
else
    if barray(2) == 0;
        f = 3*nmax+nmax*(nmax-1)/2+(nmax-(barray(5)))*(barray(5)-1)...
                          .+barray(4)+(barray(5)*(barray(5)-1))/2;
    else
        if barray(4) == 0;
            f = 2*nmax+(nmax-barray(3))*(barray(3)-1)+barray(2)...
                                .+(barray(3)*(barray(3)-1))/2;
        else
            f = 4*nmax+nmax*(nmax-1)+(barray(4)-1)*nmax+barray(2);
        end
    end
end
```

And the `Arrayfrominteger`-function consists of the following m-file:

```
function f = Arrayfrominteger(j);
global nmax
global N

if j <= nmax;
    m = j;
    f = [1,m,0,0,0];
else
    if j <= 2*nmax;
        m = j - nmax;
        f = [1,0,0,m,0];
    else
        if j < 2*nmax+nmax*(nmax+1)/2+1;
            for r = 1:1:nmax
                for s = 1:1:r
                    if Integerfromarray([0 r s 0 0]) == j
                        k = r;
                        g = s;
                    else
                    end
                end
            end
            f = [0,k,g,0,0];
        else
            if j <= 2*nmax+nmax*(nmax+1);
                for r = 1:1:nmax
                    for s = 1:1:r
                        if Integerfromarray([0 0 0 r s]) == j
                            k = r;
                            g = s;
                        else
                        end
                    end
                end
                f = [0,0,0,k,g];
            else
                for r = 1:1:nmax
                    for s = 1:1:nmax
                        if Integerfromarray([0 r 0 s 0]) == j
                            k = r;
                            g = s;
                        else
                        end
                    end
                end
                f = [0,k,0,g,0];
            end
        end
    end
end
```

Our Main.m-program now constructs the initial conditions (a Gaussian wave) using either the InitialRight.m-file or the InitialLeft.m-file. These are almost identical except for the obvious fact,

that InitialLeft constructs a leftpropagating wave and InitialRight constructs a rightpropagating. The important part is to make sure the `FFT` and `FFTSHIFT` is used properly.
InitialLeft.m looks like the following in text:

```
global nmax
global sigma
global mu
global N
global tau

for m = 1:1:nmax
    exl(m) = 0;
    exr(m) = 0;
end

for m = 1:1:nmax*(nmax+1)/2
    r(m) = 0;
    ini(m) = 0;
end

for m = 1:1:nmax^2
    b(m) = 0;
end

%% Leftpropagating Gaussian wave
for m = 1:1:nmax
    gau(m) = 1/(sigma*sqrt(2*3.14159))*exp(-((m-(nmax+1)/2)-mu)^2/(2*sigma^2));
end

gauf = fft(gau);
gaufs = fftshift(gauf);

for m = 1:1:nmax
    for n = 1:1:m
        if n == m
            ini(Integerfromarray([0 m n 0 0])-2*nmax) = gaufs(m)*gaufs(n);
        else
            ini(Integerfromarray([0 m n 0 0])-2*nmax) = sqrt(2)*gaufs(m)*gaufs(n);
        end
    end
end

global psiini
psiini = [exl exr ini r b];
```

The events in this program are described in the chapter on the initial states used to solve the multiple plasmon problem.
The governing equation of the entire problem is still `@Schr`.
The function handle `@Schr` is:

```
function dpsidt = Schr(t,psi);
global H

dpsidt = -i*H*psi;
```

The final graph is constructed by the following script - called ProbabilityAmplitude.m - it does exactly what it sounds like - collecting the different probabilities and plotting them:

```
global psi
global T
global ti
global tf
global nmax
global N
global tau

for m = 1:1:N;
    for n = 1:1:length(psi(:,1));
        psi2(n,m) = abs(psi(n,m))^2;
    end
end

for m = 1:1:N;
    for n = 1:1:length(psi(:,1));
        A(n) = 1/sum(psi2(n,:));
        psi2n(n,m) = A(n)*psi2(n,m);
    end
end
disp('Preparations for plotting complete...')

%% Probability of atomic excitation
for m = 1:1:2*nmax
    psi2ex(:,m) = psi2n(:,m);
end

for m = 1:length(psi(:,1));
    psi2e(m) = sum(psi2ex(m,:));
end

plot(T,psi2e,'k')
hold on

%% Probability of atomic excitation and leftpropagating plasmon
% for m = 1:1:nmax
%     psi2exl(:,m) = psi2n(:,m);
% end
%
% for m = 1:length(psi(:,1));
%     psi2l(m) = sum(psi2exl(m,:));
% end
%
% plot(T,psi2l,'m')
% hold on
```

```
%% Probability of atomic excitation and rightpropagating plasmon
% for m = nmax+1:1:2*nmax
%     psi2exr(:,m-nmax) = psi2n(:,m);
% end
%
% for m = 1:length(psi(:,1));
%     psi2r(m) = sum(psi2exr(m,:));
% end
%
% plot(T,psi2r,'c')
% hold on

Probability of having two leftpropagating plasmons
m = 2*nmax+1:1:2*nmax+nmax*(nmax+1)/2
    psi2lele(:,m-2*nmax) = psi2n(:,m);
end

for m = 1:length(psi(:,1));
    psi2ll(m) = sum(psi2lele(m,:));
end

plot(T,psi2ll,'r')
hold on

%% Probability of having two rightpropagating plasmons
for m = 2*nmax+nmax*(nmax+1)/2+1:1:2*nmax+nmax*(nmax+1)
    psi2riri(:,m-2*nmax-nmax*(nmax+1)/2) = psi2n(:,m);
end

m = 1:length(psi(:,1));
psi2rr(m) = sum(psi2riri(m,:));
end
(T,psi2rr,'g')
hold on

%% Probability of having one plasmon propagating in each direction
for m = 2*nmax+nmax*(nmax+1)+1:1:2*nmax+nmax^2+nmax*(nmax+1)
    psi2leri(:,m-2*nmax-nmax*(nmax+1)) = psi2n(:,m);
end
for m = 1:length(psi(:,1));
    psi2lr(m) = sum(psi2leri(m,:));
end

plot(T,psi2lr,'y')
on
%% All at once
('Time');
ylabel('Probability Amplitude');
axis([ti tf 0 1])
title('Dual Plasmon Excitation');
disp('Plotting complete...')
```

11 References

References

[1] Strong coupling of single emitters to surface plasmons - D. E. Chang, A. S. Sørensen, P. R. Hemmer & M. D. Lukin - Physics Review Letters - 24. march 2006

[2] Quantum Optics with Surface Plasmons - D. E. Chang, A. S. Sørensen, P. R. Hemmer & M. D. Lukin - Physics Review Letters - 4. august 2006

[3] Capacitive Coupling of Atomic Systems to Mesonic Conductors - Anders S. Sørensen, Casper H. van der Wal, Lilian I. Childress & Mikhail D. Lukin - Physics Review Letters - 13. february 2004

[4] Near-field photonics: surface plasmon polaritons and localized surface plasmons - Anatoly V. Zayats & Igor I. Smolyanimov - Journal of Optics: Pure and Applied Optics - 5. october 2003

[5] Single photon switching with nano-scale optical surface plasmons - D. E. Chang, A. S. Sørensen & M. D. Lukin - Physics Review Letters - 26. june 2006

[6] Quantum Optics - Marlan O. Scully & M. Suhail Zubairy - Cambridge University Press - 1997

[7] Lecture Notes for Quantum Optics - Jan W. Thomsen - 2006

[8] On the multiphoton multimode interaction between a twolevel atom and a quantum electromagnetic radiation field - Avia Rosenhouse-Dantsker - Journal of Modern Optics, vol. 39 - 1992

[9] An Introduction to Quantum Mechanics - Andrew D. Jackson - 2005