Chapter 1

The Channel

1.1 The equivalent baseband representation

We can model a wireless channel as a complex number which is changing in time. Here we ignore that the complex data symbol is modulated and then mapped to a carrier frequency and that the channel itself is a real function. And only use the complex Channel representation: // complex channel representation

1.2 SISO

The well known channel model of a wireless communication System is that we have one antenna on each side â receiver and transmitter. This creates a Channel like //y=h*x+n, where one symbol is transmitted per time slot modulated by the channel and noise added.

1.3 MIMO

Over the last decade much research was done on MIMO (Multiple Input Multiple Output) channels. We use the spatial domoin to enhance perfomance with regard to date rates, outage and coverage.

For a SISO channel in a Rayleigh fading channel we bit error rate off

$$P_b(|h|^2) = 1/2\,\mathrm{erfc}\left(\sqrt{|h|^2\frac{\bar{E}_b}{N_0}}\right).$$

If we now send the same signal over N statistically independent channels we get a BER of

$$P_b(|h|^2) = 1/2\,\mathrm{erfc}\left(\sqrt{\sum_{j=1}^{N}|h_j|^2\frac{E_s}{N_0}}\right)$$

for the euivalent SISO channel.

// formula SISO The spatial diversity as the name MIMO already sugges is achieved by using multiple Antennas on both receiver and transmitter side of the channel, which alters our channel form a single Tab //h per time to a Matrix //H. // formula MIMO Channel In a sufficient strong Fading environment and with sufficient antenna separation this Matrix is an all Random //H element of C, NxM Matrix.

Chapter 2

CSI

The question now is, is it possible to increase the Throughput of this channel with knowledge of the channel Matrix H. and to investigate this we use following Channel Representation and we are fully aware of its implications and imperfections.
// description of our channel

2.1 An Optimal Solution

If we are in single user MIMO system and we can assume full CSI (Channel State Information at Transmitter and Receiver) we can fully diagonalize the Channel by using the SVD of H:

$$H = USV^H$$

with U and V unitary and square and S a diagonal matrix consiting of al singular Values of H.
With this knowlage we can multiply our transmit and received vector as follows: $\tilde{y} = U^H y$ and $\tilde{x} = Vx$. and our transmit equation changes to

$$\tilde{y} = U^H USV^H Vx = Sx.$$

Now we have created min(N,M) parallel SISO systems out of our MIMO system. this brings us to the question of how to distribute the transmit power optimally over these systems to acheave the maximal possible ergodic rate.

2.1.1 waterfilling

The well known answer to that question is the water filling Solution described in //tsebook as follows:

2.2 Multiuser Case

Since Water Filling diagonalizese the channel it does not matter what kind of decoder is used on the receiver side, the streams are already independent. But Waterfilling is only feasible in a single user case with full CSIT due to the fact that inter antenna data coordination is needed. So in a multi user scenario we need other methods to decode the channel. We use two well-known Receiver structures:

1. Linear MMSE Receiver
   //formula

2. V-BLAST MMSE or SIC with MMSE (MMSE-SIC)
   //formula (algorithm)
   Generally speaking the SIC receivers provide better Throughput but are more complicated to implement, and harder to adapt.

2.3 Optimizing the Throughput of V-Blast

[Jindal] The Paper proposes an algorithm which optimizes the throughput with an SIC receiver, the papers goal is to solve the problem of beam forming in a Broadcast or Downlink Channel, so exactly the opposite direction than we are discussing in this report. But as //paper has described there is a duality of these two channels and we are able to use the optimized results of the MAC to optimize the BC. So his work is using this duality to solve the easer problem of the MAC and then transform it back to the BC problem, so his work is highly relevant to us.
// Duality description
// abstract of Jindal
// Jindal algorithm

2.4 Comparison

Numerical Gradient Search function We implemented a numerical gradient search for the SIC receiver to compare to the Jindal algorithm in terms of throughput maximization. We do not compare it in terms of complexitiy or speed of convergence. Convergence is guaranteed for the numerical Gradient based on the work of //paper. And the Jindal himself adds the proof of convergence to in his paper.

2.5 Linear MMSE

From there we started to investigate the LMMSE receiver.
//theory

// formulas

2.6 Optimizing

// problems //drawbacks
Fodor describes an algorithm in his //paper to optimize the throughput of a typical Cellular Network Channel, but to improve stability and feasibility of his algorithm he sacrifices accuracy and chooses to use a very week constraint. Which leads to unusable results in comparison to an easy numerical Gradient search of the same Optimum.

2.7 Numerical Gradient

So we will only look at the numerical Gradient for Optimizing the LMMSE receiver. As shown in // Figure there is some pretty big improvement over just equal power distribution, but even for high SNR cases at least one channel is closed, which means that at least one user is not allowed to send any data.

Chapter 3

Gradient search

3.1 Steepest descent

The idea of the steepest descent algorithm is to find the global maximum or minimum of a function by searching along the gradient of the function. We used

```
1. Start at one point 2. Calculate the Gradient 3. Use the gradient to get to
the next point (Step) 4. Iterate
```

two ways to calculate the gradient at a certain point, numerical and analytical:

3.2 Numerical Gradient

The numerical Algorithm calculates the Gradient with numerical means. Which means to increase the input in each dimension one after another by a very small amount and recalculate output. With both results, the original one and the new ones, we can approximate the local gradient very precisely. Analytical Gradient

```
1. Calculate Starting Point 2. 1:K a. Increase input K b. Calculate Point 3.
Calculate Gradient 4. Step to next point 5. Iterate
```

The analytical Algorithm on the other hand uses analytical tools to calculate the gradient form the known function in advance. It uses the gradient function to calculate the exact gradient at each point.

```
Pre: calculate analytical Gradient by hand 1. Calculate gradient at input Point
with the precalculated gradient function 2. Step to the next point 3. Iterate
```

3.3 Analytical Gradient for Throughput Maximization

// calculations: formulas step by step
Implementation with the norm and not just the sum allows us to use the same function in two different ways. Firstly with a //p value of 1, we just calculate the sum Rate for the throughput maximization or sum Rate maximization. Secondly we can set the //p value to something very small like -30 to create a maxmin optimizer. The Optimum would be the //min norm but this is not feasible in the numerical domain and is prone to get stuck in local minimas, here -30 is a good approximation, we tested the algorithm with -100 and achieved results which were closer to the optimum that each channel gets the same Rate but itâs not save to use.

3.4 Adaptive step Size

One Problem that occurred during the simulations was that at some point the gradient search got unstable, in our case it started to fluctuate and stopped moving towards the optimum. The reason of this bad and unwanted behavior lies in the step size of the function, and so there are two ways to fix it. The first and easy way is just to reduce the step size and increase the number of iterations accordingly, this way the problem occurs later and weaker and has less impact on the end result. The second much more complicated solution is to introduce adaptive step size.
Adaptive Step size means, that for each iteration we search the optimal step size which allows us to take the biggest step towards our goal. So it should not be possible to choose a step size which gives us a worse result than we had on the iteration before. With this Algorithm we can prevent any fluctuation in the

```
Start at an arbitrary step size, calculate the next point If next Point is better than last, increase step size If next Point is worse than last, decrease step size Stop if increasing or decreasing does not improve the result anymore
```

gradient and force it to go the fastest possible way. We do not have to calculate the gradient as often and can thus reduce the needed calculation time. Another nice side effect is, that we implicitly know when we reached the maximum or minimum when the step size is decreased to zero without finding any better points than the previous.

3.5 Implementation

//lots of code

3.6 Gradient for power Minimization

// calculations
// differences to the throughput max
// question about convecity?