Portfolio Optimization in an Artificial Financial Market

Clemens Czech, Evgeni Kochmann and Stephan Küchlin

May 28, 2010

Contents

Contents i

1. Motivation and Outline 1

1.1. Introduction 1

1.2. Outline 1

2. Portfolio Theory 3

2.1. Risk and Return 3

2.2. Diversification 4

2.3. The Efficient Frontier 5

2.4. Efficient Portfolios with a Riskless Asset — the Capital Market Line 5

3. Spread of Information 7

3.1. Swarm Intelligence 7

3.2. Gravitational Model 7

4. Proposed Model 11

4.1. Price modeling with geometric Brownian motion 11

4.1.1. Geometric Brownian Motion 11

4.1.2. Implementation 11

4.1.3. Limits to the Model 12

4.2. Portfolio Space and Coordinate Transform 12

4.2.1. Portfolio Space 12

4.2.2. Coordinate Transform 13

4.3. Gravitational Model for the Flow of Information 15

4.4. Randomization of Movement & Agent Update 16

4.5. Post Processing 18

5. Results 19

5.1. Typical Simulation 19

5.2. Observations of the general Behaviour of the Model 21

6. Conclusion 22

7. Future Work 23

A. MATLAB - Code 24

1. Motivation and Outline

1.1. Introduction

When holding a portfolio of stocks, every investor would like to maximize his expected rate of return. But unfortunately, according to the basic principles of Portfolio theory, a higher rate of return is closely linked to a higher risk, as there has to be a reason for an issuer of bonds for example to pay a higher interest rate. The interest rate can be thought of as composed of the risk free rate and a risk premium. But Portfolio theory (see section 2) also shows us, in which way diversification of a portfolio can lead to minimization of risk while maintaining the same level of expected rate of return. However, on a stock market, the future rate of return as well as the future volatility can only be estimated. So the catch is: how to know exactly just how volatile a particular asset is, and what it's expected rate of return μ might be. Could it be possible to substitute the requirement of perfect assessment of the assets one want's to invest in with different, maybe more accessible information? Even companies that earn money by means of trying to predict risk and volatility of different investment vehicles have proven to not alway be reliable. Our idea was to examine whether a set of simple rules would enable agents to invest rationally—that is, mean variance efficient—in assets they had no information about whatsoever. Instead, they would only be able to observe the choices made by others and their respective success.

1.2. Outline

A popular piece of advice amongst market operators is "Stay with the trend, the trend is your friend". By simulating an agent based artificial financial market, we study whether agents can optimize their respective portfolio towards an efficient composition when the only information they have is the composition of the portfolios held by other agents, as well as their respective "wealth".

The simulated market will consist of N agents that can invest in an entirely risk less and two risky assets. Their specific portfolio composition is thus equivalent to a point in a three dimensional *portfolio space* (see section 4.2.1). The entire market is considered to be *large* (i.e. consisting in total of $NN >> N$ agents, of which N are modeled), meaning that the investment choice of the agents modeled will not impact the prices of the respective assets. Instead, the prices of the respective assets at each period are superimposed.

The core concept of this model is how the agents choose to change the weights of the respective assets in their portfolio: A change of weighting in an agents portfolio constitutes a change of position in portfolio space. This change of position is driven by a "gravitational force" between the agents: each agent behaves like a body in space, it's mass corresponding to the total portfolio value and its position to how this value is distributed between the assets. The agents will thus eventually move towards the most successful strategy. The choice of this interaction mechanism may seem arbitrary, but in fact, it is just a way of modeling information exchange between the agents. To model an uncertainty component in the information and at the same time provide the model with a means to adapt, react and find new strategies, a Brownian motion component is superposed onto the agents' movement. At the end of a given number of simulation steps, the chosen strategies of the agents are compared to what *would have been* a rational choice if the agents *had known* the expected rate of return and standard deviation of the price curves their assets were subject to.

2. Portfolio Theory

2.1. Risk and Return

An asset's return is defined as:

$$r(t) = \frac{price(t+1)}{price(t)} \tag{2.1}$$

where t signifies a discrete timestep.

Rational investors always choose Mean-Variance efficient portfolios, i.e. for a given volatility, the investor will maximize the portfolio's expected rate of return.

We want to analyze the behavior of a rational investor in a market with two risky assets A_1 and A_2 where A_2 has a higher expected rate of return, but is also more volatile. For our analysis, we need to define some measures of performance and risk for a portfolio of two risky assets:

- **The expected rate of return μ_p**

  $$\mu_p = E(r_p) = E(\omega_1 r_1 + \omega_2 r_2) = \omega_1 E(r_1) + \omega_2 E(r_2) \tag{2.2}$$

  where $E(\cdot)$ is the discrete expected value operator and ω_i is the weight of asset i in the portfolio (i.e., proportion of the total value of the portfolio invested in asset i). μ_p is thus simply the weighted average of μ_1 and μ_2.

- **The variance σ_p^2 and the standard deviation $\sigma_p = \sqrt{\sigma_p^2}$**

  $$\begin{aligned}\sigma_p^2 = Var(r_p) &= E\left[(r_p - E(r_p))^2\right] \\ &= E\left[\langle \omega_1 r_1 + (1-\omega_1) r_2 - (\omega_1 E(r_1) + (1-\omega_1)E(r_2))\rangle^2\right] \\ &= E\left[\langle \omega_1 (r_1 - E(r_1)) + (1-\omega_1)(r_2 - E(r_2))\rangle^2\right] \\ &= \omega_1^2 E\left[(r_1 - E(r_1))^2\right] + (1-\omega_1)^2 E\left[(r_2 - E(r_2))^2\right] \\ &+ 2\omega_1^2 (1-\omega_1) E\left[(r_1 - E(r_1))(r_2 - E(r_2))\right] \\ &= \omega_1^2 Var(r_1) + (1-\omega_1)^2 Var(r_2) + 2\omega_1(1-\omega_1) Cov(r_1, r_2)\end{aligned} \tag{2.3}$$

  where $Cov(r_1, r_2)$ denotes the covariance between the two assets and is defined as:

  $$Cov(r_1, r_2) = \sigma_{1,2} = E\left[(r_1 - E(r_1))(r_2 - E(r_2))\right]$$

  The standard deviation is an (imperfect) measure for the risk of an asset or portfolio.

In our market, $E(r_{A_1}) < E(r_{A_2})$ and $\sigma(r_{A_1}) < \sigma(r_{A_2})$, because, as stated in section 1.1, A_2 will only be attractive to investment if a higher risk premium is payed.

2.2. Diversification

From the definitions given in section 2.1—in particular the non linear relationship between portfolio risk and asset risks—one can deduce an interesting property: by holding a combination of two assets, it can be possible to obtain a portfolio that is less volatile than the least volatile asset in the portfolio. Diversification may even allow for the same expected rate of portfolio return at a lower risk. This leads us to another important measure of portfolio risk:

- **The correlation coefficient ρ**

$$\rho_{1,2} = \frac{Cov(r_1, r_2)}{\sigma_{r_1} \cdot \sigma_{r_2}} \tag{2.4}$$

  The correlation coefficient tells us how two assets are related to each other with respect to the movement of their respective rates of return and indicates the possibility of risk reduction by diversification.

One needs to be aware, however, that ρ only measures linear relations. Using equation 2.4, we can re-write the equation for the portfolio variance σ_p^2 2.3 as:

$$\sigma_p^2 = \omega_1^2 \sigma_{r_1}^2 + (1-\omega_1)^2 \sigma_{r_2}^2 + 2\omega_1(1-\omega_1)\sigma_{r_1}\sigma_{r_2}\rho_{1,2} \tag{2.5}$$

A unitary correlation coefficient $\rho_{1,2} = 1$ would mean that there is a perfect positive linear relation between the two assets, i.e., the rate of return of asset A_2 at any given time would be perfectly described by a positive linear function of the rate of return of asset A_1:

$$r_2(t) = a + |b| \cdot r_1(t)$$

It is evident that in the case of $\rho_{1,2} = 1$, there is no possibility of risk reduction via diversification. The lower limit of the portfolio risk will be the volatility of the least risky asset.

A perfectly negative correlation ($\rho_{1,2} = -1$), however, offers the possibility of maximum risk reduction benefit from diversification, because the rates of return will always move in the opposite direction.

$$r_2(t) = a - |b| \cdot r_1(t)$$

Thus, one asset would allow an investor to perfectly "hedge" himself against the risk of the other. A correlation of $\rho = -1$ is purely hypothetical and cannot be observed on the markets.

If ρ were to be equal to zero, there would be no linear relationship between the assets. Theoretically, this can only be the case if one of the assets' standard deviation is equal to zero. Benefits from diversification, however, would still be possible. These properties are illustrated by the plot of equation 2.5 for $\sigma_1^2 = .25$ and $\sigma_2^2 = .5$ in figure 1, showing the portfolio variance as a function of the weight of asset 1 (ω_1) and the correlation coefficient $\rho_{1,2}$

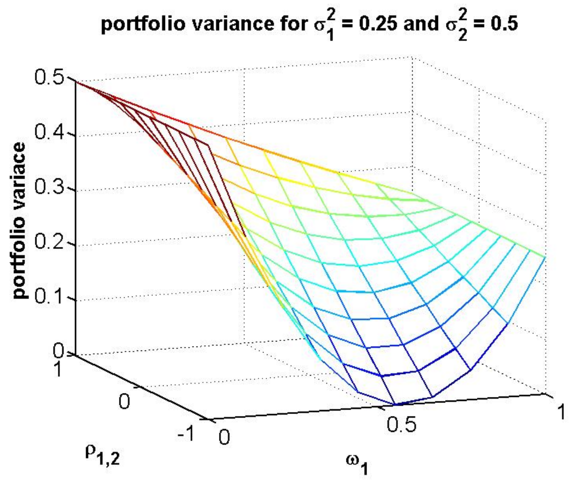


Figure 1: Portfolio variance as a function of asset weighting and correlation coefficient

2.3. The Efficient Frontier

Given two assets A_1 and A_2 with $E(r_{A_1}) < E(r_{A_2})$ and $\sigma(r_{A_1}) < \sigma(r_{A_2})$, we can plot the locations of all possible portfolios composed of these two assets in the σ_p, μ_p plane.

$$\sigma_p^2(\mu_p, \mu_1, \mu_2) = \left[\frac{\mu_p - \mu_2}{\mu_1 - \mu_2}^2\right] \sigma_1^2 + \left[\frac{\mu_1 - \mu_p}{\mu_1 - \mu_2}\right]^2 + 2 \cdot \left[\frac{(\mu_p - \mu_2)(\mu_1 - \mu_p)}{\mu_1 - \mu_2}\right] \cdot Cov(r_1, r_2) \quad (2.6)$$

The resulting curve will be a lying parabola. It's upper branch is called *efficient frontier* (EF). All portfolios on the EF are mean-variance efficient—they maximize the rate of return for a given risk. Portfolios on the lower branch are inefficient, an investor could always reach a higher rate of return at the same risk. The choice of a portfolio by a rational investor will always lie on the EF. It's location on the EF depends on the risk he investor is willing to take.

2.4. Efficient Portfolios with a Riskless Asset — the Capital Market Line

If we introduce a riskless asset A_f with a fixed rate of return r_f into the market, the hyperbola remains the same. What we change compared to the previous situation is, in fact, the new possibility of riskless lending and borrowing. If we want to mark the new asset in the σ_p, μ_p plane, it will have the coordinates $(0, r_f)$. Now, any efficient portfolio can be obtained as a linear combination of two funds: the risk free asset and the optimal risky portfolio T, which is the portfolio on the tangency point between the hyperbola

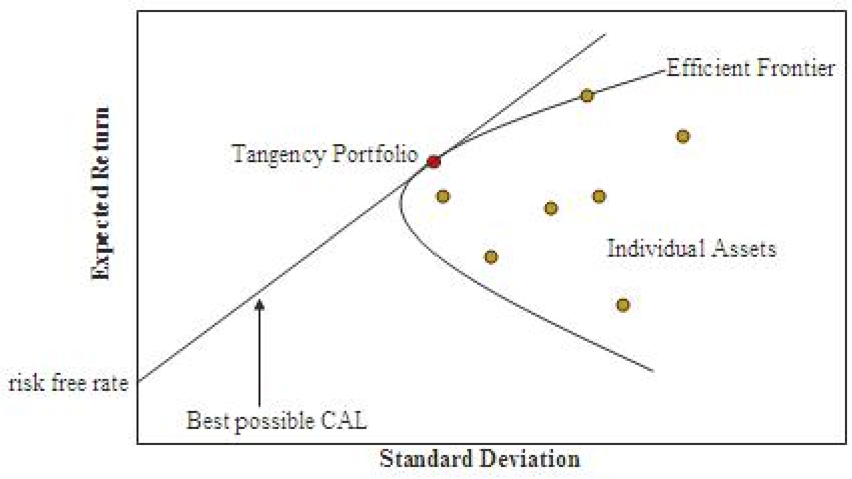


Figure 2: σ_p, μ_p plane, CML and the EF (from Wikipedia)

and a straight line passing through the risk-free rate. T is also referred to as the *market portfolio*. The tangent line passing through the location of the riskless asset and the market portfolio is called *capital market line* (CML). The CML is given by:

$$E(r_c) = r_f + \sigma_c \frac{E(r_m) - r_f}{\sigma_m} \tag{2.7}$$

where the subscripts m and c denote properties of the market portfolio and a portfolio on the CML, respectively.

A rational investor will buy a share of the market portfolio and a share of the riskless asset—he will choose a point on the CML—depending on his risk aversion. In our model—as described later in section 4.5—we evaluate the agents' "performance" by plotting their portfolios in the σ_p, μ_p plane and comparing them to the location of the CML which we obtain from calculating the observed expected rate of return and observed standard deviation of the individual assets.

3. Spread of Information

3.1. Swarm Intelligence

Since the financial theories are explained know it would be interesting to know how such a market with trading investors can be modeled in a proper way. How can a market and asset prices be generated and provided to the actors. But the main question is how to emulate the decisions an investor would make to maximize his profit.

These participators introduced in the previous sections can be modeled as agents with an artificial intelligence. This artificial intelligence can be explained as the ability to follow simple rules one provides to them. By following these boundary conditions the agents form a swarm they can interact in with one another but also with their environment. As there is no additional intervention to their activity these interactions lead to a global behavior which pretends to be intelligent even though the individuals are doing that unconsciously. This global behavior is thus called *Swarm Intelligence* (*SI*).

Depending on the rules one provides, the agents are able to adapt collectively – even though they are acting self-seeking – to a provided situation (in our case the trend of different assets). Hence, one can equalize *SI* with the collective adaption to local or global circumstances. Therefore each agent should be able to respond to quality factors in the environment (*quality principle*) (Kennedy et al., 2001).

These quality factors can be interpreted as informations. The ability of each agent to respond to it can be taken as ability to gather informations, evaluate them and make a decision. The spread of these informations can be asymmetric or perfect for each agent. This has a big effect on the behavior of each individual of the swarm and therefrom on the swarm itself. So one can imagine that it is very important to know how information is distributed along the market to model these interactions.

But how should the responses to information of the investors be modeled in detail? One approach called *Gravitational Model* is explained in the following section.

3.2. Gravitational Model

Since our market is assumed to be perfect in the spread of information the transmission of these between agents can be modeled in several ways. Our approach is to handle the agents and their information as masses in a multidimensional space (Rashedi et al., 2009). The dimension n is directly linked to the number of assets one agent can choose from for his portfolio. As the specific weights always sum up to 1 the space the agents are able to interact in is characterized as a plane in $\mathbb{R}^n$:

- **Portfolio Space**

$$x_1 + x_2 + x_3 + ... + x_n = 1 \tag{3.1}$$

where x_i is the weight of the *ith* asset in the individual portfolio.

These weights also specify the position of one agent in this space.

- **Position of the *jth* Agent**

$$X_j = [x_j^1, x_j^2, ..., x_j^n] \tag{3.2}$$

where X_j is a vector corresponding to the agent's position.

One can easily see that the positions are directly determined by the composition of each portfolio as the weight x_i^j reflects the location in the *jth* dimension.

The idea of the interaction follows the Newtonian law of gravity, where the force F between two objects is described by their masses (m_1 and m_2) and the distance r between them.

$$F = g\frac{m_1 m_2}{r^2} \qquad \text{with } g = 6.67428 * 10^{-11}\frac{Nm^2}{kg^2} \tag{3.3}$$

This resulting force turns into a movement of both object. The acceleration of each is described as the force divided by its mass.

$$a = \frac{F}{m} \tag{3.4}$$

By integrating 3.4 twice one gets the object's displacement dx in dependency of time. For a specified time step dt the offset results in:

$$dx = \frac{1}{2}\frac{F}{m}dt^2 \tag{3.5}$$

An example for the interaction between 4 masses is shown in the following figure

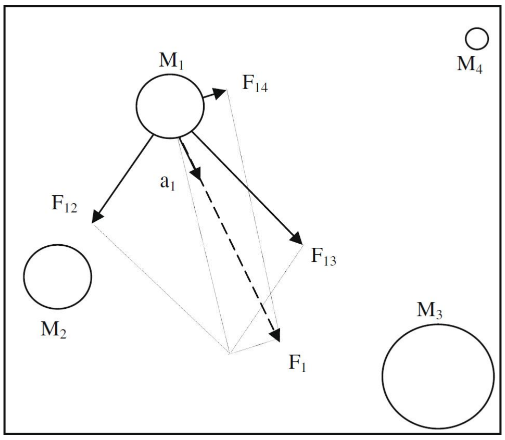


Figure 3: Attraction of mass 1 by three other masses resulting in the force F_1 (Rashedi et al., 2009)

Applying this law to our problem one has to imagine the success of each agents choice in portfolio composition as his mass. These masses interact between each other in the spanned space. As one can see from equation 3.3 heavier masses have higher influence than lighter ones. This means that agents with better portfolio choices have a higher effect on agents with worse than vice versa, so the actors with less successful portfolios will thereby move towards the compositions with higher profits. Since all agents and their masses attract each other this effect causes a global movement towards the better choices in portfolio composition (3.1).

This means related to a real market that each investor first informs himself about the payoffs reached by other portfolios and then decides to adapt his own slightly to the more profitable ones (*quality principle*). The individuals act clearly self-seeking as their only intention is to optimize their own portfolios but looking at it from a distance the actors on this market cooperate by using a direct form of communication to find the most profitable choice (*SI* 3.1).

Considering a market with N agents and adding a time dependency one has to redefine eq. 3.3 and 3.5 as follows:

- **Force on the jth Agent at time t**

$$F_j(t) = \sum_{\substack{k=1 \\ k\neq j}}^{N} F_{jk}(t) = g \sum_{\substack{k=1 \\ k\neq j}}^{N} \frac{m_j(t)m_k(t)}{||X_j(t) - X_k(t)||^3}(X_j(t) - X_k(t)) \tag{3.6}$$

where $F_j(t)$ is the Force exerted by all agents on the jth agent and $||X_j(t) - X_k(t)||$ the distance between the jth and the kth agent.

- **Displacement of the jth Agent induced by $F_j(t)$ for a specific time step dt**

$$dX_j(t) = \frac{1}{2}\frac{F_j(t)}{m_j(t)}dt^2 \tag{3.7}$$

Additionally one can add random offsets to the calculated displacements in order to avoid stagnation in one particular point. This would excite agents to give other solutions a chance instead of just adapting to the first best one. The resulting displacement of one agent could then be described as follows:

- **Final displacement**

$$dX_j^{res} = dX_j(t) + \frac{1}{m_j(t)} randX \tag{3.8}$$

where $randX$ is a random vector added to the dislocation induced by $F_j(t)$.

This random offset does not necessarily have to be regulated by the agents mass. But it is recommended to add this moderation as one clearly wants the heavier masses to trust more in their choices than the lighter ones.

Assuming a given initial distribution of the agents in the space the procedure can be summarized like followed:

1. calculate the forces acting on each agent with eq. 3.6 at a certain time t
2. determine the corresponding displacements applying eq. 3.7 for a specific timestep dt
3. add a random offset to each displacement considering the agents mass at time t
4. update the masses of each agent looking at the profit they made with the new portfolio composition at the time $t + dt$
5. check if your end criterion is reached – if not go back to step 1

The end criterion can be varying. One possibility is to check whether the agents are already arranged in one point. This makes sense since every next step of iteration would not result in a high gain of improvement (only random offset). Another option is to check if a certain wealth is achieved or as explained in 2.3 an efficient portfolio is reached by one of the agents. However in our model (section 4) the criterion is just a time simulating the days of trading in one year.

4. Proposed Model

4.1. Price modeling with geometric Brownian motion

4.1.1. Geometric Brownian Motion

In order to be able to situate our simulations and results into a realistic environment, the chart-development is simulated by a widely recognized model used in finance. The asset-price is a state-value that corresponds ideally to the amount of money buyers are willing to pay for what could be a share of a corresponding business. Its value fluctuates along with supply and demand of the asset on the market. Depending on the size of the market these parameters are linked to a innumerable number of events happening on every possible timescale. Thus the prediction or reconstruction of the time-dependent asset-price is not possible nor is that within the scope of our work. It is on the other hand possible to simulate course developments that show the same characteristics as real courses.

Brownian Motion is a time-continuous stochastic process closely related to normal distribution. Over a period of time the final displacement of an entity moving by Brownian Motion is zero. This property is used to model stock prices with a given return and variance, that still present similar characteristic fluctuations as a real chart.

A Geometric Brownian Motion is defined by the following differential equation:

$$dS_t = \mu S_t dt + \sigma S_t dW_t, \qquad t \in [0, \infty] \tag{4.1}$$

where S_t is the asset price at time t, μ is the expected rate of return, σ is the expected variance, W_t corresponds to Brownian motion.

Integrating equation 4.1 with separation of the variables leads to:

$$\int_0^t \frac{dS_u}{S_u} du = \mu t + \sigma W_t \tag{4.2}$$

A different approach of solving the equation using Ito's lemma gets us the exact solution of the GBM (Croce (2009)):

$$S_t = S_0 e^{\left(\mu - \frac{1}{2}\mu^2\right)t + \sigma W_t} \tag{4.3}$$

4.1.2. Implementation

Using the *MATLAB* class *gbm* and the function *gbm.simBySolution* we create sample course developments for the simulation of the agent-based financial market.

```
% simulate random chart by geometric brownian motion
```

```
GBM = gbm(diag(asset(1,:)), diag(asset(2,:)),'StartState', S_start);
[S,~] = GBM.simBySolution(n_tsteps-1, 'DeltaTime', dt);
```

where *asset* is a matrix containing the expected return of the assets in the first line and their expected variance in the second line, the vector *S_start* defines the initial asset values, *n_tsteps* is total number of timesteps simulated, *dt* is the timestep-length, and S is a matrix giving the price for all asset and every timestep.

4.1.3. Limits to the Model

As mentioned earlier this chapter, the implemented model computes the fluctuation by normal distribution, meaning that low deviations occur with a considerably higher probability than high deviations. Recent studies show that this assumption is critical. Normal distribution is not a good approximation of reality in financial markets[1]. The fluctuations on real markets are highly divergent and "more random" than the approximation with normal distribution. The Gaussian (normal distribution) model predicts a stock market crash to happen approximately every millennium, whereas empirical studies show that they occur every 38 years (Rachev and Mittnik (2006)). Considering this fact would impose the implementation of a so-called *fat-tail* normal distribution, which is not done in our application.
Still, since the current model is not designed to observe agent behavior during stock market crashes, the used model is adequate for this analysis.

Also the asset prices are evaluated independently of the agents actions on the artificial financial market. This simplification is appropriate assuming a relatively low transaction volume compared to the total shareholder value of the fictional enterprises. Although we did not further discuss an elaborate *supply and demand* model, this could be subject to an extension of the model (Kuehn and Neu, 2008).

4.2. Portfolio Space and Coordinate Transform

4.2.1. Portfolio Space

In our modeled market three different assets are traded. So the portfolio of every agent consists out of three parameters. No agent is allowed to store all or part of his capital other than in his asset-portfolio. Thus the sum of the price of each asset times the number of shares is the total capital an agent has invested and available for trade:

$$m_1 + m_2 + m_3 = m_{tot} \tag{4.4}$$

[1]Benoit Mandelbrot (Rachev and Mittnik (2006))

where m_i is the value of the shares an agent holds of the *ith* asset, and m_{tot} is the sum of m_i over the number of assets(3).

Dividing by the total capital(m_{tot}) gives a new form of the equation:

$$x_1 + x_2 + x_3 = 1 \tag{4.5}$$

where x_i is the weight of the *ith* asset in the individual portfolio. Since at every discrete time t all values in 4.5 are defined for every agent, the agents actual portfolio composition can be situated in a plane in $\Re^3$(see fig.4).

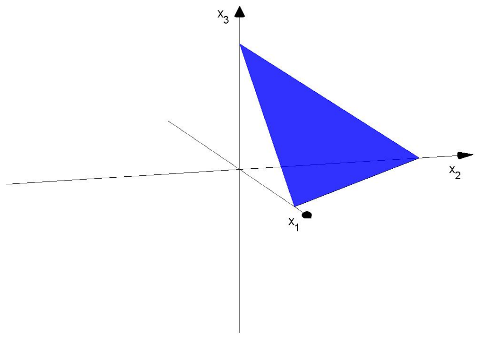


Figure 4: Portfolio Space

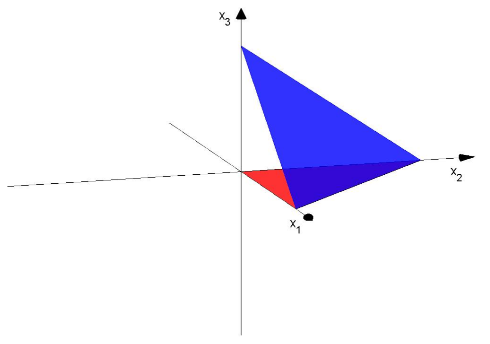
4.2.2. Coordinate Transform

As a plane is sufficiently defined by two parameters it is self-evident to describe the portfolio composition for example by:

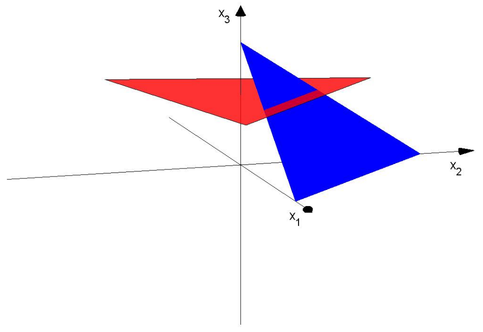
$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{4.6}$$

$$x_3 = 1 - x_1 - x_2 \tag{4.7}$$

The plane corresponding to this projection is visible in fig.5a.
For the calculation of the gravitational forces between the agents(see 4.3) this transformation is not of great interest, we chose to perform a different transformation which enables to proceed with the displacement calculation in new coordinates without considering any geometric conversion caused by the transformation. After the position update the values are transformed back into the original coordinates used in 4.5.
The used transformation rotates the portfolio plane to a plane parallel to the (x_1, x_2)-plane with an offset $x_{3,offset}$. This is performed by using a Givens rotation in the plane defined by the normal vector $\vec{n} = [1 - 1\ 0]^T$ and the angle $\alpha = \arccos \frac{1}{\sqrt{3}}$ (see fig:5b).



(a) Portfolio Space with one possible transformation



(b) Portfolio Space with rotated plane

Figure 5

$$G(\alpha) = \begin{pmatrix} \frac{1}{2}(1+\cos\alpha) & \frac{1}{2}(\cos\alpha - 1) & -\frac{1}{\sqrt{2}}\sin\alpha \\ \frac{1}{2}(\cos\alpha - 1) & \frac{1}{2}(1+\cos\alpha) & -\frac{1}{\sqrt{2}}\sin\alpha \\ \frac{1}{\sqrt{2}}\sin\alpha & \frac{1}{\sqrt{2}}\sin\alpha & \cos\alpha \end{pmatrix} \tag{4.8}$$

is the Matrix defining the rotation, $\alpha_{1,2}$ will be $\pm \arccos \frac{1}{\sqrt{3}}$ performing the transformation, respectively the back transformation.

$$\begin{aligned} G(\alpha_1) \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} &= \begin{pmatrix} x_1 \frac{1}{2}(\cos\alpha_1 + 1) + x_2 \frac{1}{2}(\cos\alpha_1 - 1) - x_3 \frac{1}{\sqrt{2}}\sin\alpha_1 \\ x_1 \frac{1}{2}(\cos\alpha_1 - 1) + x_2 \frac{1}{2}(\cos\alpha_1 + 1) - x_3 \frac{1}{\sqrt{2}}\sin\alpha_1 \\ x_1 \frac{1}{\sqrt{2}}\sin\alpha_1 + x_2 \frac{1}{\sqrt{2}}\sin\alpha_1 + x_3 \cos\alpha_1 \end{pmatrix} \\ &= \begin{pmatrix} \nu(x_1, x_2, x_3) \\ \eta(x_1, x_2, x_3) \\ x_{3,offset} \end{pmatrix} \end{aligned} \tag{4.9}$$

where $\nu(x_1, x_2, x_3)$ and $\eta(x_1, x_2, x_3)$ are the new coordinates used for further manipulation and $x_{3,offset}$ is the offset of the plane and by definition constant ($x_{3,offset} \cong 0.5774$). After the displacements have been calculated and applied, the transformation is performed again in opposite direction, giving updated x_1, x_2 and x_3 in the original portfolio space. Note that portfolio compositions with negative weights are possible, as our model allows unlimited short sales[2].

[2] "In finance, short selling (also known as shorting or going short) is the practice of selling assets, usually securities, that have been borrowed from a third party (usually a broker) with the intention of buying identical assets back at a later date to return to the lender. The short seller hopes to profit from a decline in the price of the assets between the sale and the repurchase, as the seller will pay less to buy the assets than the seller received on selling them. Conversely, the short seller will incur a loss if the price of the assets rises." Wikipedia (2010)

In the $MATLAB$-code the transformation of the portfolio coordinates is done by calling the $transform$-function.

```
function [nu eta x_{3,offset}] = transform([x_1 x_2 x_3], logic)
```

where *nu*, *eta* and x_1, x_2, x_3 are the new, correspondingly the old coordinates and *logic* is a boolean with the value $1(true)$ to perform the transformation and value $0(false)$ for the back transformation.

4.3. Gravitational Model for the Flow of Information

This part of the code is applying the gravitational model introduced in 3.2 to the masses the agents reached in the last time period. Therefore two for-loops where implemented. The first one just takes the ith agents and hands him to the second inner one. This loop calculates the forces F_{ij} corresponding to the attraction of the jth agent and the displacement dX_i resulting from it 3.2. After calculating this for every other agent it returns the final displacement to the outer loop which switches to the next actor.

The first step to achieve this is to transform the received data in to the new coordinates 4.2.2 since there is no need to consider the third dimension then. Subsequently the distance between each agent is calculated by the differences of the position in each coordinate:

```
   diff_nu = n_ax(i,1)-n_ax(j,1);
   diff_eta = n_ax(i,2)-n_ax(j,2);

   r = sqrt(diff_nu^2+diff_eta^2);
```

where $n_{ax}(i,1)$ is the ith agents's position in ν, $n_{ax}(i,2)$ the one in η and r the corresponding absolute distance between agent i and j. It should be mentioned that there is a condition for the distance r as when it is to small it is set to a value $rMin$. The reason is that if a distance becomes to small the force (eq. 3.3) gets very strong and so does the displacement of the agent. He overshoots the other agent's position contradictory to the fact that he should place there.

Then the force F induced by the two masses is determined and split up into the ν- and η-components using an angle α:

```
   F = gamma*m*M/r^2;

   alpha = atan2(diff_nu,diff_eta);

   F_nu = sin(alpha)*F;
   F_eta = cos(alpha)*F;
```

where *gamma* is one of our tuning parameters. It is chosen to be optionally adjustable in order to control the magnitude of the attraction between the agents. Also it is to be mentioned that all the masses used in this function are normalized by the highest and the lowest one. They are additionally multiplied by a so called *reward factor*. This factor is another tuning parameter as it should be possible to control the weighting of the successes achieved in the previous round. This results in a higher inertia of agents with better choices in portfolio composition.

```
mMax = max(A(:,4));
mMin = min(A(:,4));

m = 1+(A(i,4)-mMin)/(mMax-mMin)*rewardFact;
```

where $A(i,4)$ is the mass of the *ith* agent.
For the first time step of the program all agents have the same initial mass. This would cause a division by 0 and thereby a infinite mass for each agent. To avoid this a *if*-condition was added to the structure which skips this normalization in the first time interval.

The next step is to calculate the displacement caused by the forces. This is done using eq. 3.5 for each component:

```
d_nu = 0.5*F_nu/m*dt^2;
d_eta = 0.5*F_eta/m*dt^2;

Pos_nu = Pos_nu-d_nu;
Pos_eta = Pos_eta-d_eta;
```

where Pos_{nu} and Pos_{eta} are the updated positions in each coordinate after one time step. The method implemented in the code differs from the one explained in 3.2. The structure is inversed. Instead of summing up all the forces and calculating a total displacement induced by it our code sums up the single displacements caused by each agent. Anyway the effects are perfectly equal.

The new positions are then stored in a matrix which is handed to the function that adds the random offset (eq. 3.2) and updates the agents new masses for the next time step *dt*.

4.4. Randomization of Movement & Agent Update

To update the agents from time step $t-1$ to t, the following steps have to be performed in a loop that iterates over all agents:

1. After the gravitational model has determined an offset in the new coordinates, this change has to be passed on the agents.
2. As described in section 1.2, a stochastic motion component is added to the gravitation offset.
3. The change of the assets respective prices from one period to the next will shift the agents' weighting of the assets in their portfolio. This change is computed and also added, along with the new mass.

The first step is trivial and consists of the back transformation of the offset coordinates. The stochastic motion component is best added in the new coordinates: For each agent, an angle is drawn from the uniform distribution on the interval $[0, 2\pi)$ and a radius from the normal distribution with an agent specific standard deviation. The radius is scaled by the inverse of the agent's mass because we believe it to be reasonable that the more successful an agent becomes, the less willing he is to "experiment" with new strategies and the better the quality of information he will receive. A parameter ρ scales the radius and allows to tune the influence of the stochastic motion. Angle and radius are first transformed to Cartesian coordinates on the portfolio plane, and then, by a call to the transform function back, to three dimensional portfolio space.

$$\vec{o} = \begin{bmatrix} r \cdot \cos\theta & r \cdot \sin\theta & 0 \end{bmatrix}$$

A consistency check allows the definition of a limit to the amount of short sales the agents are allowed to make. If the new weights exceed this limit, the process is iterated a finite number of times. This also allows for an error check as the program will exit with an error message if it could't find acceptable weights. If successful, the offsets are added to the agent's weights.

The weighting change caused by price change is a bit tricky to calculate. First, the new total portfolio value ("mass") is calculated:

$$mass(t) = \sum_{\text{all assets } i} \omega_i(t-1) \times mass(t-1) \times (1 + r_i(t-1)) \tag{4.10}$$

where $r_i(t)$ is the rate of return of asset i at time t and $\omega(t)$ its weight in the agent's portfolio, respectively.

Next, the new weights are determined:

$$\omega_i(t) = \omega_i(t-1) \times \frac{mass(t-1)}{mass(t)} \times (1 + r_i(t-1)) \tag{4.11}$$

The update of the agent matrix is now complete.

4.5. Post Processing

We now seek to compare both initial and final strategy to the optimal risk/return ratio given by the modern portfolio theory. To be able to do so the following steps have to be completed. As expected variance and return of the simulated charts do not exactly coincide with the value we set earlier, these stochastic characteristics have to be determined. Then the actual correlation and covariance have to be calculated[3]. The efficient frontier is defined by the properties of only the two risky assets2.3 and is calculated with the function:

```
[X,Y,W]=portopt(R , cov , 1e4);
```

where X,Y are vectors defining the efficient frontier in a (σ_p,μ_p)-plane, R is the observed annual return of the risky assets and *cov* is the covariance between them. $1e4$ defines the resolution of the X and Y vectors.

In the next step we calculate the differential quotients of every discrete point on the efficient frontier(A high resultion as defined above is advantageous at this point.) and identify the tagency portfolio, which enable us to draw the Capital Market Line2.4.
The investment strategy of every agent has to be situated in the expected return over expected variance space:

```
[PortRisk, PortReturn] = portstats(muObs, covA, weightsA);
```

where *PortRisk* and *PortReturn* are columns containing the expected risk/variance and return of every agent, *muObs* is the observed return of the simulated assets, *covA* the covariance between the agents returns, and *weightsA* the final investment strategy for every agent.

PortRisk and *PortReturn* are then plotted in the same figure as the CML and the Markowitz bullet.

[3]The different charts are simulated completely independently which would result in a correlation matrix equal to the idendity matrix, but as correlation is highly probable to happen during the simulated time, this definition has to be redefined.

5. Results

The presented results were all computed with the parameters given in table 5.1. The effects of the tunable parameters are explained in 4.3 and 4.4 respectively. The choice of the fixed values is of course the decision of the user. The actual configuration for our approach is explained as follows:

- the number of agents operating on the market should be a trade-off between the computational effort and the probability to converge on a meaningful solution.
- the number of simulated days is the number of trading days in a year.
- the initial agent portfolio value can be explained as the initial wealth of the agents. But since the wealth equals the mass of each agent this value is irrelevant because of the normalization of the masses (4.3).
- the number of assets available for portfolio composition is chosen to be three as it is the best number for visualization (4.2.1)

Fixed Values		Tuning Parameters
Number of Agents	500	Gravitational Constant γ
Number of simulated Days	250	Stochastic Offset Scale ρ
Initial Agent Portfolio Value	20	Reward Factor
Number of Assets	3	Short Sale Limit
		Minimum Range of Attraction

Table 5.1: Model Parameters

5.1. Typical Simulation

For the simulation of the Price chart, we had to set expected return and expected variance of the assets, according to how we wanted the price evolution to look like (see table 5.2). Following is the analysis of a typical simulation, the used tuning parameters are given in 5.3.

Asset	Expected Return	Expected Variance
Fixed Interest Asset	0.02	0
Risky Asset 1	0.08	0.5
Risky Asset 2	0.18	1.5

Table 5.2: Available Assets

Tuning Parameters	Value
Gravitational Constant γ	$4e-6$
Stochastic Offset Scale ρ	1.5
Reward Factor	100
Short Sale Limit	20
Minimum Range of Attraction	0.2

Table 5.3: Tuning Parameters

First, the agents start collecting in three to five subclusters after 50-100 days(see figure 6. These subclusters then merge until typically, only one remains after about 200 iterations (see figure 7). The results show a clear migration towards the efficient Frontier - visible in figure 8.

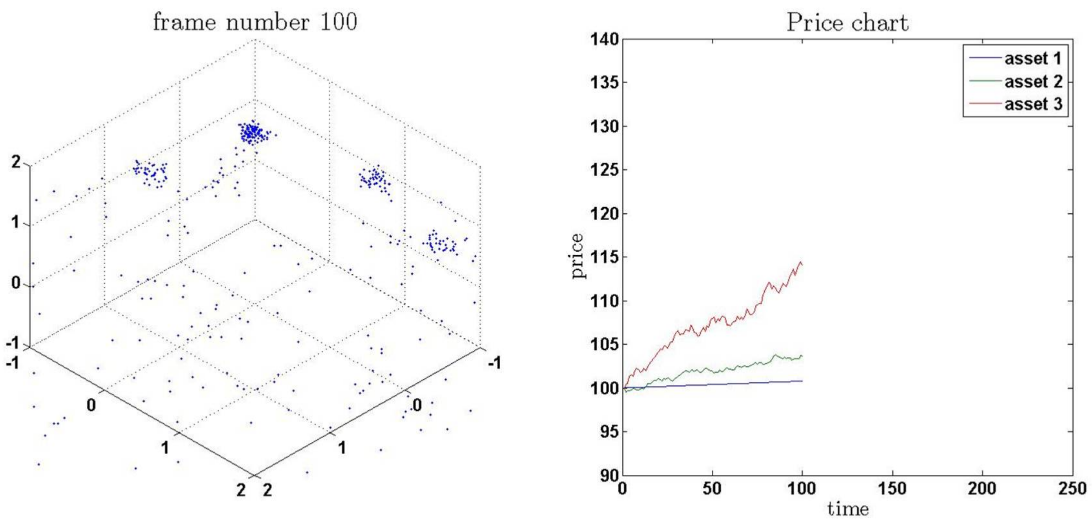


Figure 6: Weights and Price chart at time = 100 days

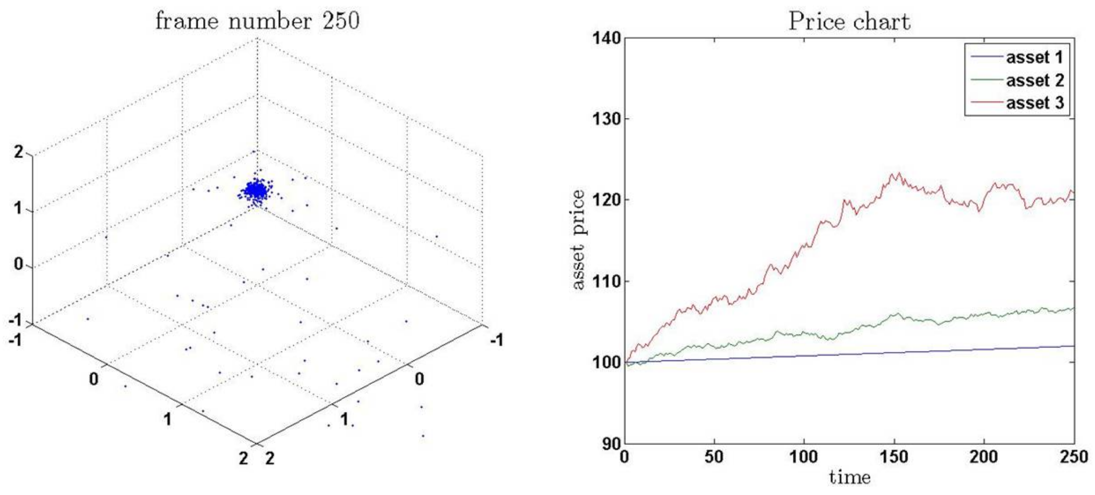


Figure 7: Weights at End of Simulation and Price chart

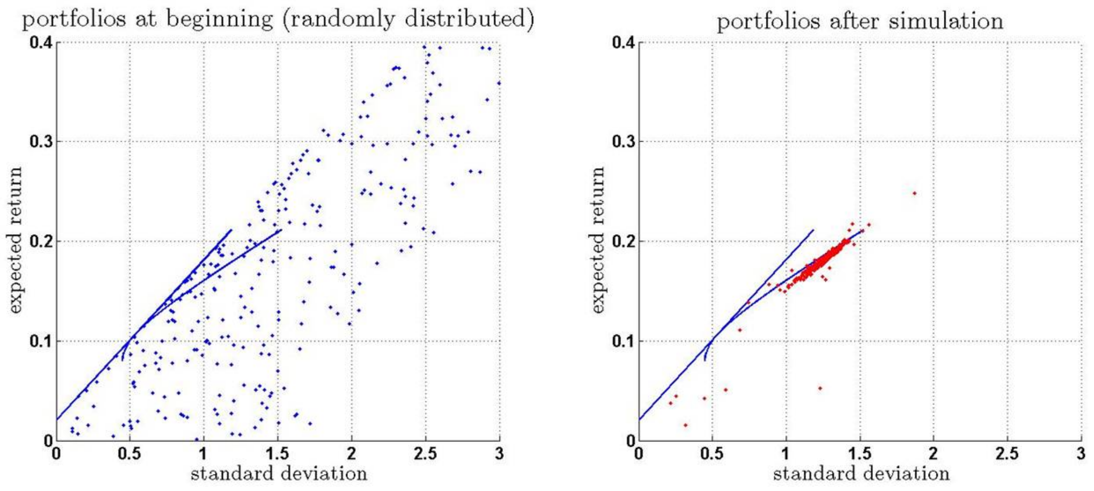


Figure 8: μ and σ at Beginning and End of Simulation

5.2. Observations of the general Behaviour of the Model

- The model has difficulties following a rapid change of prices. This could be interpreted as the impact of a crisis on herding behaviour.
- The model is sensitive to its development in the beginning of the simulation.
- In a typical simulation, the agents will collect at the efficient frontier. This means that they do not make use of the potential to sell the $1st$ asset short to buy more of the $3rd$.

6. Conclusion

An agent based simulation of an artificial financial maket was implemented in MATLAB. The agents were programmed to change their respective portfolio composition according to a weighted average of the compositions of all other agents, the weight beeing their respective success with their strategy. This was done by implementing a gravitational model in portfolio space. The simulation results clearly indicate a migration of the agents from randomly distributed initial portfolios to more efficient—that is, return maximizing for a given risk—ones. The position of the final cluster indicates that this simulation predicts a neglect of the opportunity to maximize return by short-selling the weak performing asset. It is, howver, so close to the efficient frontier and market line that it is doubtful whether any improvement beyond this point would be realistic, because the performance gauges are stochastic properties of the stock charts which change slightly on every iteration, only converging to fixed values as time goes to infinity.

These results, however, have to be interpreted with care. The stability of the simulation depends strongly on the tuning parameters. If the stochastic offset is too large or the gravitational constant too low, no cluster will form. In the opposite case, the model will converge on the first iteration and be entirely unable to adapt to changes in the prices. This means that it would be very hard to mimic the behaviour of the agents in the real market, because the "market gravitational constant" is unknown, so one does not know whether the strategy would be successfull. This, however, can be said about nearly all information available from which one might try to predict a successful strategy.

The results surely do not contradict the general advice "go with the trend" but rather support it. Gathering information about the strategies of successful players is a good piece of advice. Relying exclusively on that information, however, would make an investor voulnarable to bursting price bubbles and other possible negative effects of hearding behaviour on markets.

7. Future Work

In Order to achieve a better approximation of the real market behavior several modifications could be implemented into the existing code.

First of all the price modeling introduced in 4.1 is unsatisfying since the development of the chart is purely a stochastic function which does not reflect the trading that occurs on the market. Supplementing it with a so called interactive GBM would add a dependency of the demand to each asset chart. One approach is presented in Kuehn and Neu (2008).

Another simplification is the equality of the agents in matters of willingness to take risks. Looking at a real market investors not always want to give portfolios a chance which differ a lot from their own one. Applying this tendency may provide better solutions since several agents would have the leaning to try out new choices in portfolio composition. These could be more effective and successful than the best one achieved thus far on the present market.

The next step would be a modification to the gravitational model (4.3). In Rashedi et al. (2009) the possibility of cutting out the lightest masses in each time step so that they can be attracted by the masses left but cannot attract them. This addition would ignore the worst portfolio compositions in each round in order to avoid movement into this direction – even if this effect is almost insignificant.

Also in order to check the compatibility of our model with real market data we would like to input existing stock charts and see the behavior of the agents according to the price history.

A. MATLAB - Code

```
%---------------------------------------------%
%----Modeling Social Systems with Matlab-----%
%-----------------FS 2010--------------------%
%---------------------------------------------%
%--------Artificial Financial Market---------%
%--with Application to Portfolio Theory and--%
%-------------Herding Behaviour--------------%
%---------------------------------------------%
%-----C. Czech E. Kochmann S. Kuechlin-------%
%---------------------------------------------%

%---------------------------------------------%
clear all; close all; clc;
%---------------------------------------------%

%---------------------------------------------%
% model tuning
%---------------------------------------------%

%number of agents
nAgents = 500;

% gravitational constant
gamma = .002*1/nAgents;
%
rho = 1.5;
% short sale limit
maxShorts = 20;
% timestep
dt = 1;
% trading days p.a.
times = 250;
initVal = 20;
% reward factor
rewardFact = 100;
% expected annual returns and variances
asset = [.02 .08 .18; 0 0.5 1.5];
```

```
% minimal distance to another agent
rMin = 0.2;
rMax = 20;

%--------------------------------------------%

% generate a random chart
P = stockchart(asset, times, dt);
% convert the price history to daily returns
Ret = price2ret(P,1:times,'Periodic');

% initialize agent matrix
A = initA(nAgents,initVal,times);

f = figure('name','weights');
set(f, 'position', [400 300  1024 500]);
figure(1);
set(gca,'nextplot','replacechildren');


for t =2:times

    if t==2
        newPos = first_grav_update(A(:,1:4,t-1), gamma, dt, rMin, rMax);
    else
        newPos = grav_update2(A(:,1:4,t-1), gamma, dt, rMin, rMax, rewardFact);
    end

    A(:,:,t) = update_agents(A(:,:,t-1),newPos,Ret(t-1,:),rho,maxShorts);

   subplot(1,2,1);
   scatter3(A(:,1,t),A(:,2,t),A(:,3,t),5*ones(nAgents,1),[0 0 1],'filled');

   view(135,45);
   axis([-1 2 -1 2 -1 2]);
   title(gca,['frame number ' int2str(t)]);

   subplot(1,2,2)
   plot(P(1:t,:));
   axis([0 times 90 140]);
   title('Price')

    F(t-1) = getframe(gcf);

end

post_process(asset,A,Ret,P);
movie2avi(movie(F), 'test.avi','compression','none','fps',24)
```

```
function S = stockchart(asset, n_tsteps,dt)

% initializing
n_days = 250; %trading days pa

asset(:,:)=(1+asset(:,:)).^(1/n_days)-1;

% simulate random chart by geometric brownian motion
GBM = gbm(diag(asset(1,:)), diag(asset(2,:)),'StartState', [100 100 100]');
[S,dummy] = GBM.simBySolution(n_tsteps-1, 'DeltaTime', dt);


end
```

```
function A = initA(nAgents,initVal,times)

A = zeros(nAgents,5,times);
transA = [-3+6*rand([nAgents,2]) repmat(0.5774,nAgents,1) ];
A(:,:,1) = [transform(transA,2) repmat([initVal,1],nAgents,1)];

end
```

First Gravitational Update

```
function newPos = grav_update2 (A, gamma, dt, rMin, rMax)

%transforming the A(t) data (portfolio composition) into new coordinates:
n_ax = transform (A(:,1:3),1);

N = size(A,1);

%calculating the displacement in new coordinates by force attraction for
%each agent:

newPos = zeros(N,3);
newPos(:,3) = n_ax(:,3);
for i = 1:N

        %initial values for the displacement:
    Pos_nu = n_ax(i,1);
    Pos_eta = n_ax(i,2);

    %calculating the attraction of each agent on the ith one:
    for j = 1:N

        if j==i

            continue;

        else
            %force of attraction immitated by gravitation (for each
            %coordinate:
            diff_nu = n_ax(i,1)-n_ax(j,1);
            diff_eta = n_ax(i,2)-n_ax(j,2);
            r = sqrt(diff_nu^2+diff_eta^2);

            if r<=rMin
                continue;
            end

            if r>= rMax
                continue;
            end

            F = gamma*A(i,4)*A(j,4)/r^2;

            alpha = atan2(diff_nu,diff_eta);

            F_nu = sin(alpha)*F;
            F_eta = cos(alpha)*F;

            %displacement in effect of force (for each coordinate):
            d_nu = 0.5*F_nu/A(i,4)*dt^2;
            d_eta = 0.5*F_eta/A(i,4)*dt^2;

            Pos_nu = Pos_nu-d_nu;
            Pos_eta = Pos_eta-d_eta;


        end
    end

    %saving the final displacement for each agent:
    newPos(i,1) = Pos_nu;
    newPos(i,2) = Pos_eta;
end
```

```
function newPos = grav_update2 (A, gamma, dt, rMin, rMax, rewardFact)

%transforming the A(t) data (portfolio composition) into new coordinates:
n_ax = transform (A(:,1:3),1);

%determining the number of agents involved:
N = size(A,1);

%determining the most and less successful portfolio from the previous round:
mMax = max(A(:,4));
mMin = min(A(:,4));

%calculating the displacement in new coordinates by force attraction for
%each agent:

%initial matrix for new positions:
newPos = zeros(N,3);

%last coordinate stays equal in new coordinates
newPos(:,3) = n_ax(:,3);

for i = 1:N

    %normalizing the ith agents mass and multiplying reward factor:
    m = 1+(A(i,4)-mMin)/(mMax-mMin)*rewardFact;

    %initial values for the displacement:
    Pos_nu = n_ax(i,1);
    Pos_eta = n_ax(i,2);

    %calculating the attraction of each agent on the ith one:
    for j = 1:N

        %skipping the agents own mass
        if j==i

            continue;

        else
```

```
%normalizing the mass of the jth attracting agent:
M = 1+(A(j,4)-mMin)/(mMax-mMin)*rewardFact;

%differences in position for each dimension in new coordinates:
diff_nu = n_ax(i,1)-n_ax(j,1);
diff_eta = n_ax(i,2)-n_ax(j,2);

%absolute value of distance:
r = sqrt(diff_nu^2+diff_eta^2);

%controlling overshoot:
if r<=rMin
    continue;
end

if r>= rMax
    continue;
end

%force from gravitaion law for the ith and the jth mass:
F = gamma*m*M/r^2;

%direction of the force
alpha = atan2(diff_nu,diff_eta);

%splitting up into the composites in new coordinates:
F_nu = sin(alpha)*F;
F_eta = cos(alpha)*F;

%displacement in effect of force (for each coordinate):
d_nu = 0.5*F_nu/m*dt^2;
d_eta = 0.5*F_eta/m*dt^2;

%adding the displacement to the current position of the agent:
Pos_nu = Pos_nu-d_nu;
Pos_eta = Pos_eta-d_eta;
```

```
        end
    end

    %saving the final displacement for each agent:
    newPos(i,1) = Pos_nu;
    newPos(i,2) = Pos_eta;
end
```

```
 function A_new = update_agents(A,newPos,Ret,rho,maxShorts)

%-------------------------------------------------------------------------%
% the function "update_agents"
% 1. takes the new portfolio weights proposed by the gravitational
% model as input in matrix "newPos" and updates the portfolios accordingly
% 2. adds a stochastic component to the movement in "weighting space"
% by randomly off-setting the agents' positions taking into account
% consistency restrictions (number of times the entire portfolio may be
% shorted in parameter "maxShorts"), the agents current "mass"/total
% asset value and their specific sigma. this is tuned by parameter "rho"
% 3. takes the price change of the stocks as return(t) (1x3)
% and updates the agents' mass.
% this means that "trading" and price change occur at seperate times
% 4. returns the updated agent matrix A_new
%-------------------------------------------------------------------------%

% vector of old portfolio total values
 oldMassV = A(:,4);

% 1.
 newWeightsM = transform(newPos,2);

% 2.

% number of agents
nAgents = size(A,1);

% traverse matrix to add specific stochastic offset
    % choose a random offset magnitude from normal distribution
    % useing abs to get a positive value
    % with agent specific sigma
    % inversely proportional to agent's mass
    % "light" agents will then behave more erratic than "heavy"
    % alpha is a tuning parameter to scale offsets
    % choose a random offset direction as angle in weight eigenspace
    % from uniform distribution on [0,2*pi]

for i = 1:nAgents
```

```
mass = oldMassV(i);
% weights before stochastic component is added
newWeightsV1 = newWeightsM(i,:);
sigma = A(i,5);

consistent = 0;

% do this until a consistent offset ( all weights >= maxShorts)
% is found
count = 0;
while  ~consistent

    r = rho * (1/mass) * abs(normrnd(0,sigma));
    theta = 2*pi*rand(1);
    offset = [r*cos(theta) r*sin(theta) 0];
    % consistecy is checked in cartesian coordinates
    % vector containing new weights after adding the stochastic
    % component
    newWeightsV = newWeightsV1 + transform(offset,2);

    count = count +1;

    if count > 100000

       disp('error: couldnt find acceptable weights');
       return
    end


    if all(newWeightsV >= -1*maxShorts)
        consistent = 1;
    end



end

% finally update agent's weighting
```

```
    newWeightsM(i,:) = newWeightsV;


end

% 3.

% update agent's mass
pFact = Ret + [1 1 1];
% pfact is the price change as factor
% new mass is given by newmass = oldmass*sum(weight(i)*pfact(i))
 newMassV = oldMassV.*(newWeightsM*pFact');


% new weights are given by (asset i, agent j)
% newweight(i,j) = oldweight(i,j)*oldmass(j)/newmass(j)*pfact(i)
  invMassFactM = repmat((oldMassV./newMassV),[1 3]);
  pFactM = repmat(pFact,[nAgents 1]);
  newWeightsM = newWeightsM.*invMassFactM.*pFactM;

% 4.
 A_new = [newWeightsM newMassV A(:,5:end)];

 end
```

```
function [] = post_process(asset,A,Ret,~)

% observed mean annual return
muObs           = (mean(Ret)+1).^250-1;

% observed annual standard deviation
sigmaObs        = (std(Ret)+1).^250-1;

% EFFICIENT FRONTIER
% calculate covariance and return of risky assets
sigmaHyp        = sigmaObs(2:end);
corrHyp         = corr(Ret(:,2:3),Ret(:,2:3));
R               = muObs(2:end);
cov2            = corr2cov(sigmaHyp , corrHyp);
% calculate efficient frontier data
[X,Y,W]         = portopt(R , cov2 , 1e4);

% CAPITAL MARKET LINE
% calculate the derivate of the efficient frontier
diff            = zeros(size(X));
for i=1:(length(X)-1)
    diff(i)     = (Y(i+1)-Y(i))/(X(i+1)-X(i));
end

% identification of tangency portfolio for CML
B               = abs((Y-asset(1,1))./X - diff);
[~,i]           = min(B);

% calculate agents portfolio coordinates resulting from simulation in figure
covCml          = corr2cov(sigmaObs,corr(Ret,Ret));
[PortRisk, PortReturn] = portstats(muObs, covCml, A(:,1:3,end));
[PortRiskInit, PortReturnInit] = portstats(muObs, covCml, A(:,1:3,1));

% PLOT

f = figure('name','mu-sigma');
set(f, 'position', [400 300  1024 500]);

% final distribution
```

```
subplot(1,2,2);
axis([0 3 0 max(max(PortReturn),max(PortReturnInit))]);
hold on;
% draw markowitz bullet
line(X,Y)
% draw capital market line
line([0,(max(muObs)-asset(1,1))/diff(i)],[asset(1,1),max(muObs)])
% situate final portfolios in plot
plot(PortRisk,PortReturn,'.r','MarkerSize',5)

title('portfolios after simulation')
xlabel('standard deviation')
ylabel('expected return')
hold off;

% initial distribution
subplot(1,2,1);
xlabel('standard deviation')
ylabel('expected return')
axis([0 3 ...
    0 max(max(PortReturn),max(PortReturnInit))]);
hold on;
% draw markowitz bullet
line(X,Y)
% draw capital market line
line([0,(max(muObs)-asset(1,1))/diff(i)],[asset(1,1),max(muObs)])
% situate initial portfolios in plot

plot(PortRiskInit,PortReturnInit,'.b','MarkerSize',5)
title('portfolios at beginning (randomly distributed)');
hold off;

end
```

```
function B = transform(D, const)

if const == 1
    alpha = acos(1/sqrt(3));
else
    alpha = -acos(1/sqrt(3));
end

Rot = [0.5*(1+cos(alpha)) 0.5*(cos(alpha)-1) -1/sqrt(2)*sin(alpha);...
       0.5*(cos(alpha)-1) 0.5*(1+cos(alpha)) -1/sqrt(2)*sin(alpha);...
       1/sqrt(2)*sin(alpha) 1/sqrt(2)*sin(alpha) cos(alpha)];

B = D*Rot';

end
```

References

R. Croce. Primer on the Use of Geometric Brownian Motion in Finance. *http://web.econ.ohio-state.edu/ croce/Geometricf*, 2009.

J. Kennedy, R.C. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann, 2001.

R. Kuehn and P. Neu. Intermittency in an interacting generalization of the geometric Brownian motion model. *Journal Of Physics A: Mathematical And Theoretical*, 41, 2008.

S. Rachev and S. Mittnik. Abschied von der Glockenkurve. *Risiko Manager*, 11/06:20–22, 2006.

E. Rashedi, H. Nezamabadi, and S. Saryazdi. GSA: A Gravitational Search Algorithm. *Information Sciences*, 179:2232–2248, 2009.

Wikipedia. Short (finance) — wikipedia, the free encyclopedia, 2010. URL `http://en.wikipedia.org/w/index.php?title=Short_(finance)&oldid=363792176`. [Online; accessed 25-May-2010].