# Financial Statistics, fall 2023 - Project Report

Eliot Montesino Petrén, el6183mo-s, 990618-9130, LU Faculty of Engineering F19 , eliot.mp99@gmail.com,
Linus Särud, li6008sa-s, 990612-1539, LU Faculty of Engineering F19 , linus@sarud.se

## I. Task 1 (Track A) - Derive the student-$t$ GAS model

### A. Introduction of the GAS framework

The generalized autoregressive score (GAS) framework generates a family of models that addresses the limitations of traditional time series models in capturing the dynamic properties of financial data. Financial markets, when empirically studied exhibit characteristics such as time-varying volatility, volatility clustering, heteroscedastic returns and fat tails, which some models fail to adequately capture, especially more traditional time series models. The GAS framework tries to add flexibility and robustness and thus offers a more accurate representation of such dynamics compared to more traditional time series models. When the GAS framework is applied on assumed gaussian distributed returns, the GARCH(1,1) model can be obtained. Eq 1 is the gaussian GAS model parameter dynamics, eq 2 is the GAS model rewritten to look more similar to the GARCH(1,1) form [3]. When assuming a student-$t$ distribution instead it will not be exactly the same result.

$$f_{t+1} = \omega + A_1 \left(y_t^2 - f_t\right) + B_1 f_t \tag{1}$$

$$f_{t+1} = \omega + A_1 y_t^2 + (B_1 - A_1) f_t \tag{2}$$

### B. GAS models specification

A model from the GAS framework assumes autoregressive dynamics and parameters are updated dynamically based on a score function. Data is generated from the observation density seen in eq 3.

$$y_t \sim p\left(y_t \mid f_t, \theta, \mathcal{F}_{t-1}\right) \tag{3}$$

Where $y_t$ are returns, $f_t$ denotes dynamic parameters, $\theta$ denotes constant parameters and $\mathcal{F}_{t-1}$ is filtration of all available information up to the current time. The autoregressive dynamics are found in the time varying parameter assumed in the model as stated in eq 4.

$$f_{t+1} = \omega + \sum_{i=1}^{p} A_i s_{t-i+1} + \sum_{j=1}^{q} B_j f_{t-j+1} \tag{4}$$

The scaled score function (eq 5) is based on the score function (eq 6) (a log-likelihood function derivated w.r.t $f_t$) and it is utilized to increase the likelihood iteratively in the parameter estimation.

$$s_t = S_t \nabla_t \tag{5}$$

$$\nabla_t = \frac{\partial \log p\left(y_t \mid f_t, \theta, \mathcal{F}_{t-1}\right)}{\partial f_t} \tag{6}$$

Where $S_t$ is a scaling parameter, usually using the Fischer information matrix, $S_t = \left(I^F\right)^{-1}$. The task is to derive the student-$t$ GAS model, i.e an application of the above.

### C. Student-t GAS model derivation

Assume that returns are heteroscedastically student-$t$ distributed, see eq 7.

$$y_t = \sigma_t z_t^{t(\nu)} \tag{7}$$

Let define the time-varying parameter as eq 8.

$$f_t = \sigma_t^2 \tag{8}$$

Rescale student's t to get the same format as in the task. This ensures that $\sigma^2$ is the variation.

$$\begin{aligned} \eta_t &= \sqrt{\frac{\nu-2}{\nu}}\epsilon_t, \\ y_t &= \sigma_t \eta_t, \end{aligned} \tag{9}$$

The probability density function (PDF) of the student-$t$ distribution is given in eq 10.

$$f(x) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\pi\nu}\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{x^2}{\nu}\right)^{-(\nu+1)/2} \tag{10}$$

Now combine the rescaling in eq 9 into the PDF of eq 10 and obtain the density function, $p\left(y_t \mid f_t, \theta, \mathcal{F}_{t-1}\right)$. Thereafter take the log of the density function and obtain the log density in eq 11.

$$\begin{aligned} &\log p\left(y_t \mid f_t, \theta, \mathcal{F}_{t-1}\right) = \\ &= \log\Gamma\left(\frac{\nu+1}{2}\right) - \log\Gamma\left(\frac{\nu}{2}\right) - \frac{1}{2}\log(\pi) - \frac{1}{2}\log(\nu-2) \\ &- \frac{1}{2}\log\left(\sigma_t^2\right) - \frac{(\nu+1)}{2}\log\left(1 + \frac{y_t^2}{(\nu-2)\sigma_t^2}\right) \end{aligned} \tag{11}$$

Now derive the log density function in the equation above w.r.t $f_t$ and obtain the score function for the student-$t$ GAS model, see eq 12.

$$\begin{aligned} \nabla_t &= \frac{\partial \log p\left(y_t \mid f_t, \theta, \mathcal{F}_{t-1}\right)}{\partial \sigma_t^2} = \\ &= \frac{(\nu+1)}{2}\left(1 + \frac{y_t^2}{(\nu-2)\sigma_t^2}\right)^{-1} \frac{y_t^2}{(\nu-2)\sigma_t^4} - \frac{1}{2\sigma_t^2} \end{aligned} \tag{12}$$

Lastly, the scaling parameter $S_t$ has to be derived and the inverse Fisher information matrix will be utilized, see def in eq 13

$$S_t = \left(I^F\right)^{-1} = -\mathrm{E}_{t-1}\left[\nabla_t \nabla_t'\right]^{-1} \tag{13}$$

The definition, eq 13, is the variance of the score, this is the same as eq 14.

$$-\mathrm{E}_{t-1}\left[\frac{\partial^2 \ln p\left(y_t \mid f_t, \theta, \mathcal{F}_{t-1}\right)}{\partial \sigma_t^2 \partial \sigma_t^2}\right]^{-1} \tag{14}$$

Thus, we have to perform a second partial derivation, then take the expectational of this expression and lastly the inversion, which is the most simple step just turning the expression around. The two preceding steps require a lot more work though. Finally we obtain the scaling parameter, the inverse Fisher information matrix of the student-$t$ GAS model in eq , 15.

$$S_t = \left(I^F\right)^{-1} = \frac{2\sigma_t^4(\nu+3)}{\nu} \tag{15}$$

A weight term will be defined (eq 16) in order to simplify the final summary of the student-$t$ GAS model.

$$w_t = \left(1 + \frac{y_t^2}{(\nu-2)}\right)^{-1} \frac{(\nu+1)}{(\nu-2)} \tag{16}$$

Finally the score function, inverse of the information matrix, and factor recursion (after simplifications) are seen in eq 17.

$$\nabla_t = \frac{(\nu+1)}{2}\left(1 + \frac{y_t^2}{(\nu-2)f_t}\right)^{-1} \frac{y_t^2}{(\nu-2)f_t^2} - \frac{1}{2f_t}$$

$$S_t = \frac{2f_t^2(\nu+3)}{\nu} \tag{17}$$

$$f_{t+1} = \omega + A\frac{(\nu+3)}{\nu}\left(w_t y_t^2 - f_t\right) + Bf_t$$

Thus marking the end of this task. Note the similarities between the student-$t$ GAS model compared to the gaussian GAS model in eq 1 and the differences: namely the weight term $w_t$ and $\frac{(\nu+3)}{\nu}$.

## II. Task 2 - Calibration of implied volatility

### A. The Black-Scholes model and implied volatility

Implied volatility and calibration is (strictly numerically and mathematically speaking) essentially trying to reverse the Black-Scholes (BLS) model to instead try to find the volatility of the underlying asset, rather than trying to find out the price of a European options contract. Calibration is finding the implied volatility with some method and numerical approach and this can be done with a non-linear least squares or a non-linear filtering approach.

Economically, the implied volatility can be interpreted as the constant volatility of the underlying asset of a contract during the entire time period of the maturity *estimated by the buyers and sellers on the options market.* The BLS formula is non-linear and we expect non-linear estimation in this task. The BLS formula is seen in eq 18.

$$\begin{aligned} C\left(S_t, t\right) &= N\left(d_+\right) S_t - N\left(d_-\right) K e^{-r(T-t)} \\ d_+ &= \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right] \\ d_- &= d_+ - \sigma\sqrt{T-t} \end{aligned} \tag{18}$$

Where $C$ is the contract price, $S$ is the value of the underlying asset, $K$ is the strike price, $T$ is the maturity, $t$ is the current time (time to maturity is $\tau = T - t$), r is the risk-free interest rate, $\sigma$ is the volatility and $d_+$ and $d_-$ are the auxillary functions.

Note that market data provides (somewhat, more on that later) direct measurements of all the parameters in the BLS formula except for the volatility. This is because the volatility is in a way rather a theoretical concept. The volatility is a latent process and it cannot be measured directly, unlike the other parameters, and an economist has to resort to other methods and this is why the implied volatility is common, also enforced by the fact that the BLS formula is a famous and common model in finance as well. The volatility of a underlying asset, say a stock, could of course also be estimated without the BLS formula, but that is not the task here. The task is to find the implied volatility by finding a volatility that fits the data, the observed market prices over the entire timespan, thus estimating the $\sigma^{impl}$

The BLS model is derived from assumptions of the underlying asset as geometric brownian motion in a stochastic differential equation applied for the problem of pricing of European option contracts, but the details or the history of the BLS model is not relevant to discuss further. It could be useful to state however and keep in mind that the BLS model is forward-looking and assumes that the volatility of the underlying asset is constant between the time of when the contract is offered on the market up to the maturity date, even though the volatility is more accurately modeled as not constant during the entire maturity. Therefore it is important to differentiate the idea of the "true volatility" of the underlying asset and the implied volatility, hence denoted as $\sigma$ and $\sigma^{impl}$. These volatilities are, *not the same thing.* Let's begin.

### B. Pre-processing data and assumptions: Illiquid options market and smile

Task 2 is given as eq 19

$$C_t^{\text{Market}}(S_t, K, \tau_t) = C_t^{\text{BLS}}(S_t, K, \tau_t, r_f, \sigma^2) + \eta_t \tag{19}$$

The first step in a modeling process is of course to inspect the data and to make necessary pre-processing. In order to calibrate $\sigma^{impl}$ we first need to decide on values for all parameters in the BLS model except for $\sigma$.

The complete arguments of how the parameter values was chosen will be given here. The main task is what we assume is the "true" value of a said parameter in the BLS model, as there are no single quoted market prices but they come in bids and asks.

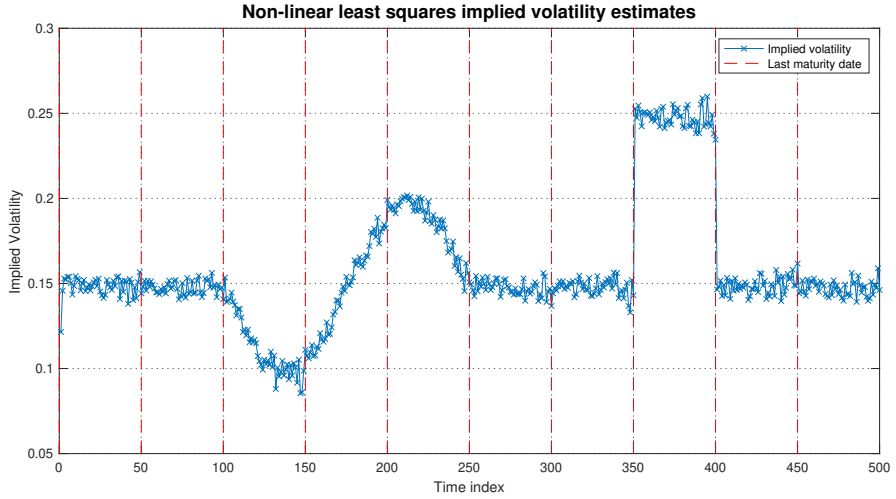


Fig. 1. The calibrated implied volatility over the data set using the least squares approach.

The bids and asks are assumed to be distributed around the true fair market price, as the the one who makes bids makes a bid favorable for themselves and the other way around is true for the one who asks for a price, so the true fair price must lie in between. The data is from an illiquid European options market however so we have to make smart choices here, as there are not enough observations in order to fit a probability density function to the bid-ask spread. The latter is in line with "the law of large numbers".

Starting with the underlying asset, $S_t$. The spread is almost constant over the whole time series, $S_t^{spread} = abs(S_t^{ask} - S_t^{bid}) \approx 0.1$. and we assume mid price. Note here that the value of the underlying asset is the same no matter the contract.

$$S_t^{market} = \frac{S_t^{ask} + S_t^{bid}}{2}$$

The strike is unique for every contract and we must decide how we should react when there is more than one contract available. Here the the concept of volatility surface and smile is relevant. As explained before regarding the BLS model $\sigma^{impl}$ is not a perfect estimator model and empirically it has been observed that a 3D plot of calibrated implied volatility dependent on the strike and the maturity gives a non-flat volatility surface, in contrast what the BLS model would optimally give. Regarding strike especially, the implied volatility is generally higher for moneyness further away from $moneyness = 1$, where:

$$moneyness = K_{t,i}/S_t^{market} \tag{20}$$

Moneyness is a measurement of how much the contract is out of money or in the money or on the money. The 2D relationship of $\sigma_{impl}$ and $K$ forms a "smile" with a minimum of $\sigma^{impl}$ when the contract is on the money, thus the term smile. [5].

Because of smile, the contract that is closest to $moneyness = 1$ (on the money) will be selected when there are two contracts available. By doing so we are trying to reduce the skew that is introduced with smile and we try to keep the estimated fair market prices as consistent possible. Once a contract is selected, $K^{market}$ has been chosen as well.

Lastly, the mid price is assumed for the contract price as well following the same argumentation as with the $S^{market}$. The $C_t^{spread} = abs(C_t^{ask} - C_t^{bid}) \approx 0.2$ for the whole dataset as well and the constant ask-bid spreads indicate a constant uncertainty in the market prices (and in $S_t^{market}$).

$$C_t^{market} = \frac{C_t^{ask} + C_t^{bid}}{2}$$

Now remains calibration.

### C. Calibration approaches and the non-linear least squares approach

There are several ways to approach this non-linear time series problem and they could be divided into two groups: non-linear least squares and non-linear filtration.

The least-squares approach in general is to minimize the value found in equation 21 according to [4].

$$L_t^{WLS}(\theta) = \sum_{s=1}^{t}\sum_{i=1}^{N_s} \lambda_{s,i}\left(C_s^*(K_i,\tau_i) - C_s^{Model}(K_i,\tau_i;\theta)\right)^2,$$
$$\hat{\theta}_t = \underset{\theta\in\Theta}{\operatorname{argmin}}\ L_t^{WLS}(\theta). \tag{21}$$

However, in our case we were given that we should use BLS as model, a restriction really. The only params $\theta$ we have to optimize is then $\sigma^{impl}$ since the other params are decided from preprocessing. This means that we do not need to sum over time. As there are no restrictions on $\sigma^{impl}$ given we will always be able to find a value that set equation 22 to zero, meaning that there is no need for weights either. It should be noted that there are a few options in the data where this is not possible to achieve, which we interpret as a negative $\sigma^{impl}$. In these instances we set the $\sigma^{impl} = 0$ as that is the economically best interpretation, compared to the alternative to assume $\sigma_t^{impl} = \sigma_{t-1}^{impl}$ that would be the best interpretation regarding the true $\sigma$ (though in the end there

was always an alternative options contract available that had a bid that yielded a market price so the final $\sigma^{impl}$-time series had no zero elements in the end). Remember that $\sigma^{impl}$ is the volatility through the eyes of the actors on the market, not an actual perfect depiction of the true volatility of the underlying asset, which is of course $\sigma > 0$.

$$(C_s^{market}(K_i, \tau_i) - C_s^{BLS}(K_i, \tau_i; \theta)) \tag{22}$$

The implementation in Matlab is simply to loop over all options and using fzero to find the $\sigma$ that makes equation 22 close enough to zero and afterwards extract the $\sigma^{impl}$-time series. The final result is seen in fig 1. This is algorithm is also actually the Matlab function "blsimpv" from the financial toolbox, which inspired us to abandon the first least squares approach we had when we realized this way was better and produce the same results as argued above. It might been better just to use blsimpv as it is based on numerical optimizations as referenced in the source code to "Let's be rational"[2], but we wanted more control just to be sure, and we did want to avoid the feeling of "cheating".

### *D. The non-linear filtration approach: Extended Kalman filter*

For the filtration approach we decided to go with an extended Kalman Filter (EKF) that yielded satisfactory results. A Kalman filter estimates latent process, for example volatility, as discussed earlier. The EKF can be used on non-linear problems. The Kalman filter is only valid for linear modeling, but in the EKF the non-linearity of the BLS model has been linearized, thus making it an attractive choice.

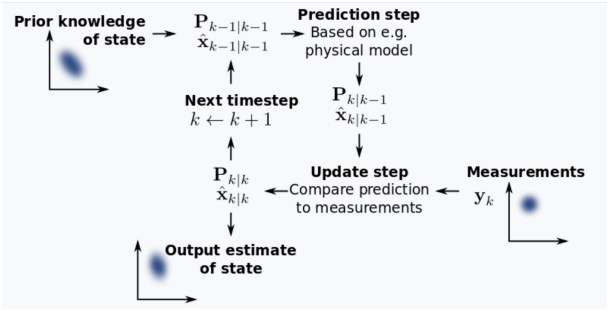


Fig. 2. The Kalman filter algorithm that tracks the estimated state of the system and the variance or uncertainty of the estimate. The estimate is updated using a state transition model and measurements. [1]

Our implementation of the EKF uses a similar approach as the measurement function plugged into the Kalman filter. We store the $\hat{\sigma}_t^{impl}$ as a internal state with the goal of minimizing the discrepancy between model price and market price at each time point. The discrepancy is used to calculate the Kalman gain $K_t$, that decides how much we the value of $\hat{\sigma}_t^{impl}$ should be updated to minimize the value in the update step. Please see figure 2 for a general visual illustration of the Kalman filter and the steps. The EKF we used was implemented as specified in the **Prediction Step** in eq 23 and the **Update Step** in eq 24.

$$\begin{aligned} \hat{\sigma}_{t|t-1} &= \hat{\sigma}_{t-1|t-1} \\ P_{t|t-1} &= P_{t-1|t-1} + Q \end{aligned} \tag{23}$$

$$\begin{aligned} y_t &= C_t^{market} - C_t^{\text{BLS}}(S_t, K_t, \tau_t, r_f, \sigma_{t-1}^2) \\ H_t &= \left.\frac{\partial C_t^{\text{BLS}}(S_t, K_t, \tau_t, r_f, \sigma_{t-1}^2)}{\partial \sigma}\right|_{\sigma=\hat{\sigma}_{t|t-1}} \\ S_t &= H_t P_{t|t-1} H_t^\top + R \\ K_t &= P_{t|t-1} H_t^\top S_t^{-1} \\ \hat{\sigma}_{t|t} &= \hat{\sigma}_{t|t-1} + K_t y_t \\ P_{t|t} &= (I - K_t H_t) P_{t|t-1} \end{aligned} \tag{24}$$

$$\begin{aligned} H_t &= S \cdot \phi(d_1) \cdot \sqrt{\Delta t} \\ d_1 &= \frac{\ln(S/K) + (r + 0.5 \cdot \sigma^2) \cdot \Delta t}{\sigma \cdot \sqrt{\Delta t}} \end{aligned} \tag{25}$$

Where $P$ is the estimate covariance, $Q$ is noise covariance, $R$ is measurement noise covariance, $K$ is the Kalman gain, $y_t$ is the measurement residual, $H$ is the linearized BLS formula (more about this soon), $\phi$ is the gaussian probability density function, $I$ the indentity matrix. Observe that it is only one state being estimated, therefore scalars are handled rather than matrices.

In eq 24 we see the state prediction (corresponds to a AR(1)) and covariance prediction. The mentioned linearization is made with the Jacobian that is operated on the BLS model formula, this is known as the vega, see eq 25. The vega is a measure of the sensitivity of an option's price to changes in the volatility of the underlying asset. In the context of the Black-Scholes model, the vega is defined as the partial derivative of the option price with respect to the volatility of the underlying asset. This is what makes the Kalman filter able to work for non-linear measurement equations!

If we assume that $C_t^{spread}$ is Gaussian distributed we could assume the spread to be one standard deviation above and below mean and chose $R = \sigma^2 = (\frac{C_t^{spread}}{2*\sqrt{3}})^2$ to tune the model. The initial sigma was set to the by-eye observed mean of the time series, to set some kind of valid starting point. The rest of the EKF parameters were set to the identity due to no apparent reasons found yet to what they should be otherwise, but we have some ideas that will be posed soon. Different combinations were tested however to see how the EKF performed, the performance was roughly the same no matter the parameter valus (except for extreme values of course).

TABLE I
EKF PARAMETERS

| | |
|---|---|
| $R$ | $(\frac{0.2}{2*\sqrt{3}})^2$ |
| $\sigma_t$ | 0.15 |
| Rest | $I$ |

This filter was iterated over the full dataset and the $\hat{\sigma}_t^{impl}$ was extracted into a time series, see fig 3

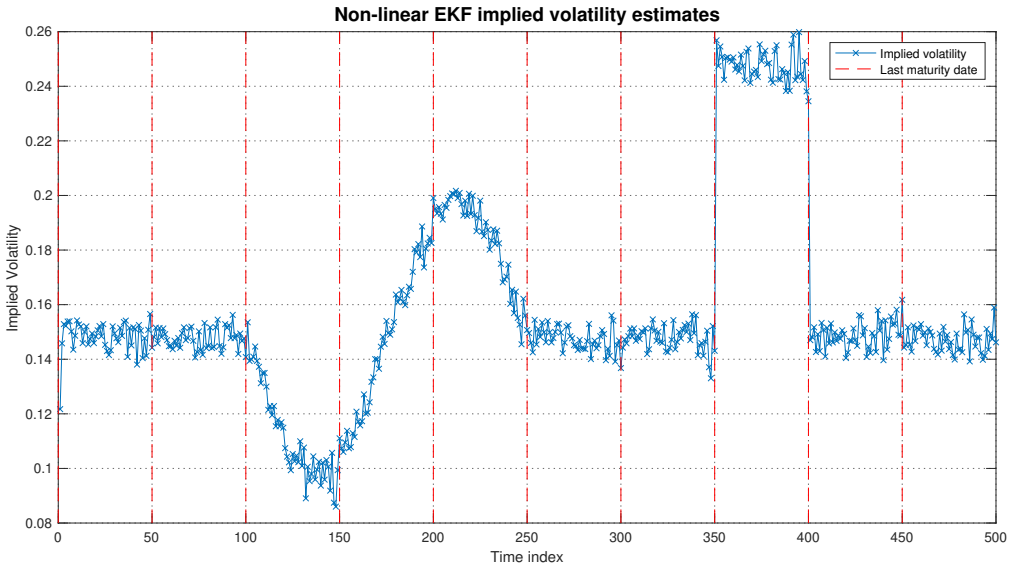


Fig. 3. The calibrated implied volatility over the data set using the EKF

### *E. Comparison of the least squares and filtering approaches*

Finally the two approaches should be compared against each other and the results that they produce. Quantitatively speaking, the results are very similar although the filtering approach using an EKF has some inherent and unique benefits that should be brought up and could be further built on, especially when considering a more general problem because the EKF offer more efficiency and versatility and in a kind of a win-all in comparison to the alternative.

In this case both approaches (where we are restricted to use the BLS model) provided close to almost exactly the same result in the two $\sigma^{impl}$-time series, please see fig 4. In a residual plot (not included here) it is apparent that they differ in the ability of adapting to non-linear changes with large residuals such as at the contract starting at 350. Both approaches feels quite natural in this task, where the least squares approach is the easiest to implement as it does not have parameters to consider in tuning (even though not much tuning was there to be made).

The least squares approach becomes more or less a straight up ”inverted BLS formula” in this case (solving for the volatility in the formula). There are no closed form solutions for this inversion but the calibration was solved with optimization. There was even a function in Matlab ”blsimpv” that performs the task in a similar manner. The problem is also very natural to solve with a Kalman filter as this type of filtering is used for latent processes in partially observed models, fitting volatility perfectly. Since the final time series is basically the same it

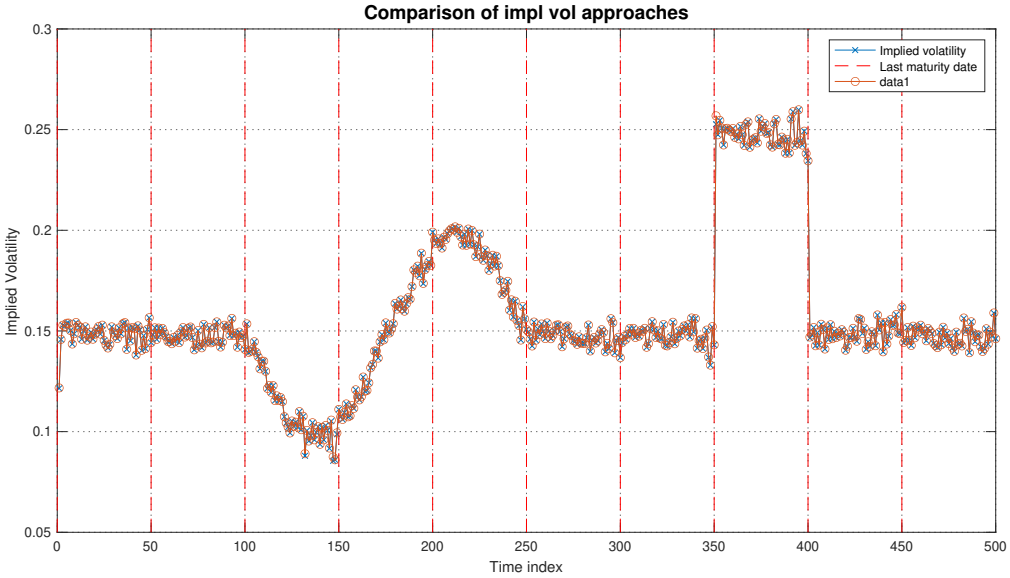


Fig. 4. The calibrated implied volatility from both approaches plotted on top of each other. They are very similar and seem to optimize against the same values. ”data1” is the implied volatility from the EKF.

is shown that the Kalman filter was can perform "at least as good" as the least squares approach, so to speak, but it has deeper possibilities that was not utilized to the extent it could have. This was also partly due to the data at hand. In retrospect it is clear that EKF is a superior estimator in comparison.

Despite the results they differ underneath greatly in capabilities. The filtration was expected to provide a more "smooth" time series as it takes previous values into account and here it assumes the volatility dynamic as an AR(1) process. At an earlier stage this meant that the EKF always calibrated the $\sigma^{impl} > 0$ even in cases where the least squares one could return zero volatilities (mentioned earlier). The Kalman filter has the distinct quality of being able utilize old data to improve its' estimate, while the least squares approach does not have this capability at all for essentially point wise estimation and it is a quite drastic assumption to make to say the least. If we would add complexity and see a smoother time series it would be assumed to be a good sign as the volatile and "edgy" time series are likely a sign of over-fitting against noise. For the least squares approach noise is not reduced more than what preprocessing is able to offer but it must instead be included in the time series. This must mean that noise unfortunately is not minimized much further in the case of the EKF since the results are almost identical. Despite this, the EKF has the capability to separate process and measurement noise and rule them out, because they are built into the model.

The EKF comes with the benefits to be able to detect jumps and other non-linear characteristics apparent from the final results of the non-linear process from the BLS model but these capabilities come with least squares naturally. As mentioned earlier, the Kalman filter approach also has the ability to assume volatility dynamics and a GARCH(p,q) could be implemented for process and noise dynamics (to account for heteroscedacity is a good trait!). However, all of these extra implementations require time and more fine tuning as to compared to the least squares approach. Assuming high order processes with bad tuning could smooth-out the implied volatility even further so that it fails to capture sudden non-linear changes, so fine tuning could also work against you if you have bad guesses, though one would prefer the non-linear filtration approach after we have made this task we would say. The EKF algorithm runs notably faster in Matlab, indicating lower time complexity as well (since the least squares has to do many calculation to converge to every point and the Kalman filter rather "moves around" with steps.)

There are more potential the benefits of the non-linear filtering approach, the bid-ask spread could also be incorporated into the measurement noise $R$ of the Kalman filter. This seems to note have made a large difference in our case, especially since the the bid-ask spread was constant indicating a constant uncertainty in the market. In the same spirit the time to maturity is speculated to also could be Incorporated into the Kalman filter by adjusting a parameter ($K$, $R$) depending on the time to maturity left that is in line with the volatility surface in regards of the relationship between $\sigma^{impl}$ and $\tau$.

## III. Task 3 - Modelling using SDE:s

The Lotka-Volterra model is an ODE that models the dynamics of the populations of a hunter and a prey species. The preys are food to the hunters, thus negatively impacting the population of the preys and positively impacting the population of the hunters and the preys live off some other food other outside of the model at a constant rate. It is an interesting model that has stable solutions (when assuming no stochastic processes) that show how the populations can oscillate over time and how they depend on each other and time-invariant phase space plots showcases this especially.

The Lotka-Volterra model can however be expanded to an SDE where stochastic dynamics are incorporated into the model, which one could argue is a more accurate depiction of reality. In the Lotka-Volterra SDE the populations are assumed to vary over time as GBM. In this task a Lotka-Volterra SDE models the Tuna and Garfish populations of the Baltic sea, defined in eq 26.

$$\begin{aligned} \mathrm{d}G_t &= (\alpha G_t - \beta G_t T_t)\,\mathrm{d}t + \sigma_G G_t\ \mathrm{d}W_t^{(G)} \\ \mathrm{d}T_t &= (\delta G_t T_t - \gamma T_t)\,\mathrm{d}t + \sigma_T T_t\ \mathrm{d}W_t^{(T)} \end{aligned} \tag{26}$$

Now the task remains to estimate the parameters that decide the dynamics of the hunters (Tuna) and the preys (Garfish) from real data of the populations $G_t$ and $T_t$ with a suitable SDE numerical estimator. The parameters to estimate are: $\alpha$ $\beta$, $\sigma_G$, $\delta$, $\gamma$ and $\sigma_T$.

The data provided is low frequency data and gathered biyearly $\Delta t = 6/12$.

We decided to solve it using Euler Maruyama discretization and then simulated maximum likelihood estimation. The discretized equation is 27.

$$\begin{aligned} G_{t+\Delta t} &= G_t + (\alpha G_t - \beta G_t T_t)\Delta t + \sigma_G G_t \Delta W_t^{(G)} \\ T_{t+\Delta t} &= T_t + (\delta G_t T_t - \gamma T_t)\Delta t + \sigma_T T_t \Delta W_t^{(T)} \end{aligned} \tag{27}$$

According to the Pedersen algorithm we know that

$$p_\theta(x_t|x_s) = \mathbb{E}_\theta\left[p_\theta(x_t|x_\tau)|\mathcal{F}(s)\right], \quad t > \tau > s \tag{28}$$

So our solution was to simple between every sample we have generate k paths $x_\tau^k \sim p(x_\tau|x_s)$ and computing the average of that. We transformed that to a log likelihood and summed it up over the whole dataset. Then we let Matlab fminsearch optimize the different parameters to maximize the total likelihood.

Combined with 27 we get the joint likelihood function that we are maximizing as .

$$\begin{aligned} L(\theta) =& N(G_t, G_\tau + (\alpha G_\tau - \beta G_\tau T_\tau)\Delta\tau, \sigma_G G_\tau\sqrt{\Delta\tau}) \\ & \cdot N(T_t, T_\tau + (\delta G_t T_t - \gamma T_t)\Delta\tau, \sigma_T T_\tau\sqrt{\Delta\tau}) \end{aligned} \tag{29}$$

This we can easily convert to the negative log likelihood function. We can analytically calculate a direct solution as it would be unnecessary to take the log of a exponent. That was done in the following code block.

```
EG = sim_G(:,end-1) + (alpha * sim_G(:,
    end-1) - beta * sim_G(:,end-1) .*
    sim_T(:,end-1)) * delta_t;
ET = sim_T(:,end-1) + (delta * sim_G(:,
    end-1) .* sim_T(:,end-1) - gamma *
    sim_T(:,end-1)) * delta_t;
SG = sigma_G * sim_G(:,end-1) * sqrt(
    delta_t);
ST = sigma_T * sim_T(:,end-1) * sqrt(
    delta_t);

avgG = -0.5 * log(2 * pi * SG.^2) -0.5*((
    G_t(i) - EG).^2./(SG.^2));
avgT = -0.5 * log(2 * pi * ST.^2) -0.5*((
    T_t(i) - ET).^2./(ST.^2));

avgG = mean(avgG);
avgT = mean(avgT);
```

To allow the optimizer to execute several iterations we generated all random numbers before calling likelihood function, meaning that we continued to optimize over the same random numbers. This is a technique called common random numbers.

To make sure that our solver worked we wrote a function to simulate data similar to the one given in the task. Unfortunately we had a code error in the simulation which we did not discover until very late in the process, making us believe our general method did not work properly since it failed to optimize the parameters that we know as correct. To solve this we tried several different variations on the simulated maximum likelihood as well as different variants of importance sampling, none that resulted in better values.

One attempt on a simple importance sampling method we tried was to discard to very high and low probabilities. This was easily implemented by replace the call to the Matlab function $mean$ with $trimmean$ which we used to calculate the mean after discarding the five percent lowest and five percent highest scores. However, it was discarded when it did not seem to produce better results.

After solving on the real data we simulated new data based on the found parameters to confirm that the data is reasonable. In figure 5 we can see twenty instances of simulated data against the given data. As we make a stochastic simulation each instance will be slightly different. We find it very fascinating how the data can vary this much and yet we are able to find the parameters.

It is reasonable that the populations in the simulations diverge over time, as the collective variance increase for Brownian motion processes over time. Another validation for our results is that the simulations seem to have roughly the same frequency in the population oscillations.

The values our solver found can be found in table II.

TABLE II
LOTKA-VOLTERRA MODEL PARAMETERS

| Parameter | Value |
|---|---|
| Alpha | 0.2125 |
| Beta | 0.4137 |
| Delta | 1.4636 |
| Gamma | 1.9268 |
| Sigma_G | 0.0799 |
| Sigma_T | 0.1180 |

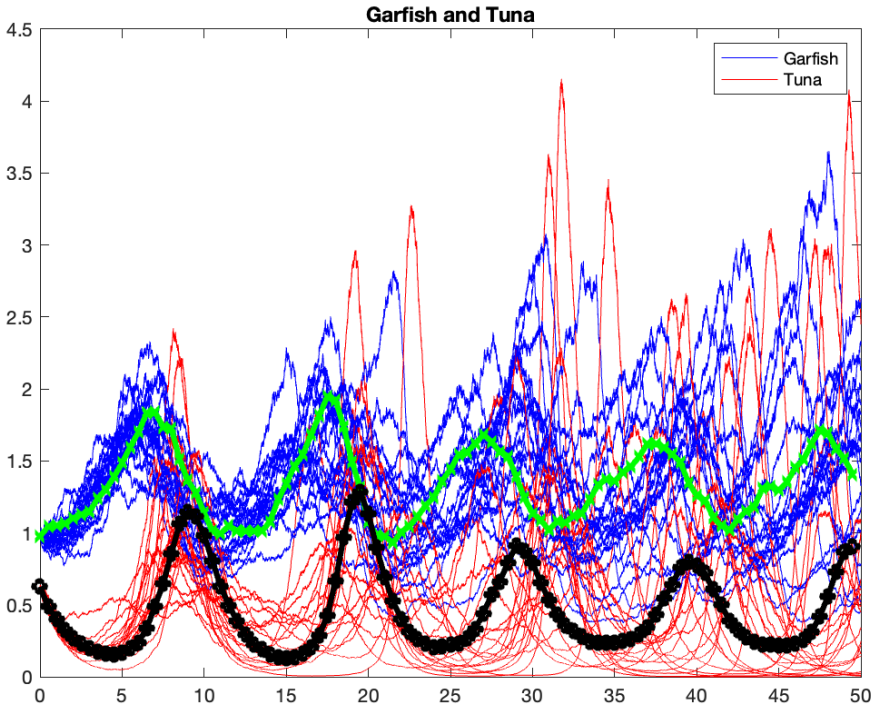


Fig. 5. Twenty simulations based on the parameters from the table compared with the given data.

In our solver we also implemented the restriction that our parameters must be positive. This comes from a physical understanding of the Lotka Volterra-equation. We also set the sigmas to always be positive. Lastly a constraint was that the amount of animals always need to be positive, which we solved by setting $y_\tau$ to $y_{\tau-1}$ if it ever went into negatives.

### A. *Extra!*

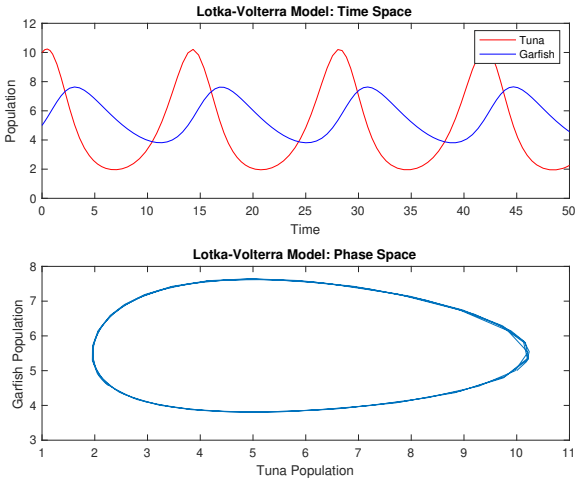


Fig. 6. A Lotka-Volterra ODE that generates a stable oscillation in time-space and a closed loop in the phase-space plot.

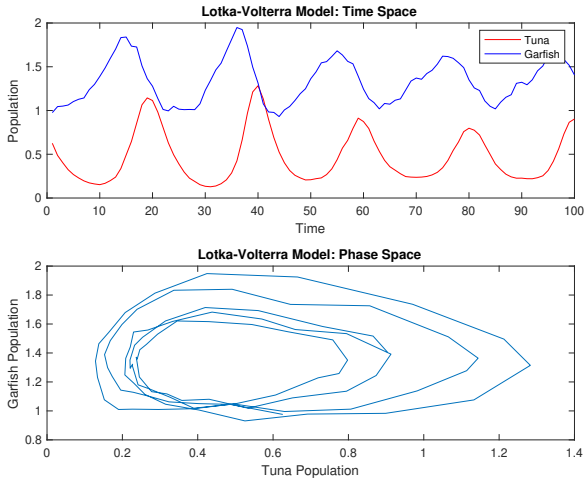


Fig. 7. The given data of the populations in a time-space and phase-space plots. The swirl is not as apparent.

For fun, phase space plots were also made, see fig 6 to fig 8. Observe how the stochastic characteristics in the SDE Lotka-Volterra add risk for prey and predator populations to exit a stable oscillation illustrated in the phase-space by a swirl. The SDE Lotka-Volterra is thus able to incorporate natural risk and stochastic nature to explain how can populations randomly

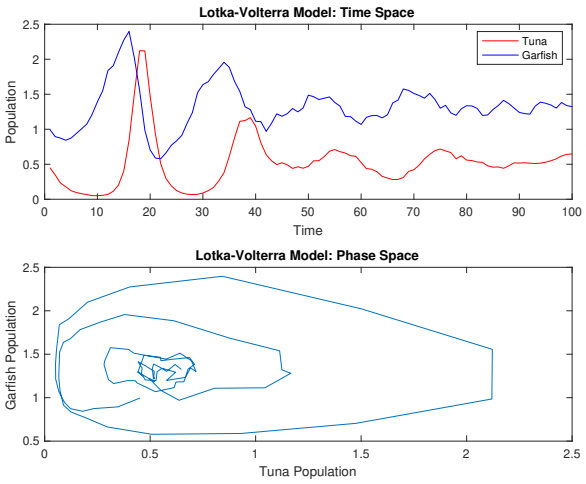


Fig. 8. One selected sim-generated time series of the populations with time-space and phase-space plots to illustrate a swirl in the phase-space. The populations seems to stabilize to a reduced oscillation.

change. For stable solutions of an ODE the phase space form a predictable closed loop.

## IV. References


[1] Wikipedia contributors. *Kalman filter*. 2024. URL: https://en.wikipedia.org/wiki/Kalman_filter (visited on 01/11/2024).

[2] Peter Jäckel. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/wilm.10395.

[3] Erik Lindström. "FMSN60/MASM18 Financial Statistics Lecture 6: Advanced Variance Models". Presented at Centre for Mathematical Sciences, Lund University. 2023.

[4] Erik Lindström et al. "Sequential Calibration of Options". English. In: *Computational Statistics and Data Analysis* 52 (2008), pp. 2877–2891. ISSN: 1872-7352. DOI: 10.1016/j.csda.2007.08.009.

[5] Christoph Reisinger. *Calibration of Derivative Pricing Models*. Lecture presented at [Mathematical Institute of University of Oxford], June 2018. June 2018.