Multi-Period portfolio Optimization using asymmetric GARCH Model

Chirag Jindal
170225
chiragji@iitk.ac.in

Akhilesh Chauhan
170070
akhilc@iitk.ac.in

***Abstract*—In the following work, we consider dynamic portfolio over discrete time periods. We implement the techniques given in [1] to a real world financial data. We propose an improvement to the given model using adaptive variance. We use an asymmetric GARCH Model to forecast volatility. We compare the costs using both the approaches.**

I. Introduction

The word portfolio and optimization as we now understand weren't clubbed together until recent times. For decades, the dividend discount model was considered as the best way to price assets and to buy assets that were best priced and making a basket of such securities was known as a portfolio, risk in fact wasn't even considered as a parameter in financial texts until the 1950s, making way for Harry Markowitz's Modern Portfolio theory and CAPM (capital asset pricing model) formalizing the risk-reward trade-off as we know today. With the advancements in technology and computation capabilities, algorithmic trading has really taken off. We are now, able to trade electronically at frequencies of milliseconds. Much of the portfolio management these days is done using trading algorithms. There is a boom in the number of firms which just work on algorithmic trading.

A. Mathematical Formulation

We consider a dynamic portfolio comprising of N securities. We assume that the prices vary in discrete time intervals. We take the portfolio positions over T such time intervals. We start from an initial portfolio value of $\boldsymbol{p}_0$. Let the value of our portfolio at time t be $\boldsymbol{p}_t$.Therefore, $\boldsymbol{p}_t$ would be a $N \times 1$ vector where each row would denote the amount invested in the security i.Note here that if a row is less than zero, then it would mean that we have a short position in the security i. We want to reach a value of final portfolio. The objective is to minimise the net amount of cash inflow. Let the returns in each security between time t-1 to t be $\boldsymbol{r}_t$. We assume that the returns are log-normal. Let the trade value at time t be $\boldsymbol{g}_t$. Then,

$$\boldsymbol{p}_{t+1} = diag(\boldsymbol{r}_{t+1}) * (\boldsymbol{p}_t + \boldsymbol{g}_t)$$

Here, $\boldsymbol{p}_t + \boldsymbol{g}_t$ would be our portfolio just after the time t. Also the new value of our portfolio will be given by a scalar v_t,

$$v_t = \mathbf{1}^T \boldsymbol{p}_t$$

Now the money that we cash in to the portfolio at time t is given by $c_t(\boldsymbol{p}_t, \boldsymbol{g_t})$.Note here that it is not equal to the net value of trade as there are other costs involved. It includes the value of trade,transaction costs, shorting cost, quadratic cost ,Risk penalty(the cost worth of taking risk). Let's look into the various components of cash inflow:

- **Value of trade:** It accounts for the total value of trade done at time t, and is given as:
$$\mathbf{1}^T \boldsymbol{g}_t$$
- **Transaction cost:** It is a linear cost which is associated with each transaction and is given by:
$$\boldsymbol{k}^T |\boldsymbol{g}_t|$$
where $\boldsymbol{k}(i)$ is the proportionality constant for the security i.
- **Shorting Cost:** There is an additional proportional cost associated with shorting a security, this is known as shorting cost and is given by:
$$\boldsymbol{s}^T (\boldsymbol{g}_t)_-$$
where $\boldsymbol{s}^T$ is the proportionality constant and $(x)_- = abs\{\min\{x, 0\}\}$
- **Quadratic Cost:** In order to keep the value of our trades lower, we have a quadratic costs which relatively punishes very large amounts of trades. It is given as:
$$\boldsymbol{g}_t^T diag(\boldsymbol{q}) \boldsymbol{g}_t$$
where $\boldsymbol{q}$ is the proportionality constant.
- **Risk Penalty:** It is not an actual cost. We want to maximise the risk adjusted return which is why, we take the risk as an extra cost in the objective function. It is given by:
$$\alpha(\boldsymbol{p}_t + \boldsymbol{g}_t)^T \Sigma (\boldsymbol{p}_t + \boldsymbol{g}_t)$$
where α is the proportionality constant, Σ is the co-variance of the return distribution.

Note that all of the costs given above are convex in nature. Now the net cash-in at time t will be given by,

$$c_t(\boldsymbol{p}_t, \boldsymbol{g}_t) = \mathbf{1}^T \boldsymbol{g}_t + \boldsymbol{k}_t^T |\boldsymbol{g}_t| + \boldsymbol{s}_t^T (\boldsymbol{g}_t)_- + \boldsymbol{q}_t^T \boldsymbol{g}_t^2 + \alpha_t (\boldsymbol{p}_t + \boldsymbol{g}_t)^T \Sigma (\boldsymbol{p}_t + \boldsymbol{g}_t)$$

So, the objective of our problem is essentially to just minimise the net cash-in to the portfolio, that is

$$\min_{\boldsymbol{p}_t, \boldsymbol{g}_t} \sum_{t=0}^{T} c_t(\boldsymbol{p}_t, \boldsymbol{g}_t)$$
$$s.t. \quad \boldsymbol{g}_t \in C_t$$

Where C_t is the constraint set for the trades. There can be many possible constraints to the trades being executed, some examples being,

- Unconstrained : This means $C_t \in \mathbb{R}^N$
- Long Only : As the name suggests, we can only take long positions. Mathematically it can be expressed as:

$$C_t \in \mathbb{R}_+^N$$

- Leverage Limit : It essentially puts a limit to the fraction of shorting trades that can be done. It is mathematically expressed as:

$$C_t \in \{\boldsymbol{g} | \mathbf{1}^T(\boldsymbol{g})_- \leq \eta \mathbf{1}^T \boldsymbol{g}\}$$

where η is the limit.

II. Methods

A. Approximate Dynamic Programming

We can think of the above optimization problem as a backward recursion problem. The idea behind the approach being that at a time t, we do not have any idea about what is going to happen in the future. It will essentially become a similar problem starting from the time point t. This is why we try to associate a value function at every time point.This is a dynamic programming based method to solve the recursive problem. The Value function $V_t(\boldsymbol{p})$ denotes the optimal cost from the time point t to T with the current portfolio $\boldsymbol{p}$. Now as the returns are log normal with know mean and co-variance, value functions will also be random, we need to minimize the expectations.

$$V_t(\boldsymbol{p}) = \inf_{\boldsymbol{g}} (c_t(\boldsymbol{p}, \boldsymbol{g}) + \mathbb{E} V_{t+1} \left(diag(\boldsymbol{r}_{t+1})(\boldsymbol{p} + \boldsymbol{g})\right))$$

As it is clear from above, $V_t(\boldsymbol{p})$ will start from time point t and minimize the sum of cash-in at t and the expected cash in from time t+1 to T.

$$V_{T+1}(\boldsymbol{p}) = 0$$

As we do not need to put any cash-in after the time T. Now to compactly write the above expression of recursion, we express it as a bellman operator[1].

$$V_t = \tau_t V_{t+1} \text{for t= T,T-1,....,1,0}$$

Now as already shown in [1] and [2], the bellman operator is monotone and it preserves convexity and quadratic nature. That is,

$$f \leq g \quad \Rightarrow \quad \tau_t f \leq \tau_t g$$

Now as it is not possible to solve for the value function, we approximate it by a quadratic function and try to get a lower bound to the value function using the approximation. Let us call the approximate value function as $V_t^{lb}(p)$.

$$V_t^{lb}(p) = \begin{bmatrix} p & 1 \end{bmatrix} \begin{bmatrix} X_t & x_t \\ x_t^T & y_t \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix}$$

is a quadratic function in p. To solve for V_t^{lb}, we need to construct quadratic functions such that $V_t^{lb} \leq \tau_t V_{t+1}^{lb}$ and $V_{T+1} = 0$. Now using the monotonicity, we can see that,

$$V_T^{lb} \leq \tau_t V_{T+1}^{lb} \leq \tau_t V_{T+1}$$

$$V_T^{lb} \leq V_T$$

Now by recursion, we can see that,

$$V_t^{lb} \leq V_t \text{ for t =1,..,T}$$

Now as shown in [1], the above problem can be converted into LMI constraints and finding V_t^{lb}, can be converted to an SDP with the objective function being,

$$max_{X_t, x_t, y_t} V_0^{lb}(\boldsymbol{p}_0)$$

$$max_{X_t, x_t, y_t} \frac{1}{2} \boldsymbol{p}_0^T X_0 \boldsymbol{p}_0 + x_0^T \boldsymbol{p}_0 + y_0$$

Intuition behind this objective function is that we need to tighten our lower bound. Now as we can see that the above is an SDP as the objective function is linear in $\{X_t, x_t, y_t\}$.
Now after obtaining a lower bound on the value function, we can replace the value function by its lower bound, we get the approximate dynamic programming method. At every time point t, we get the optimal trading policy ($\psi^t(p)$) by solving:

$$\psi^t(p) = \arg\min_{\boldsymbol{g}} (c_t(\boldsymbol{p}, \boldsymbol{g}) + \mathbb{E} V_t^{lb}(diag(\boldsymbol{r}_{t+1})(\boldsymbol{p} + \boldsymbol{g})))$$

Now as we have approximate the value function using a quadratic bound, the above problem becomes a QP. However, point to be noted here is that, as we are approximating the value function, this solution is sub-optimal. Hence the ADP algorithm essentially boils down to 2 types of computations:

- **Offline Computation:** This requires solving an SDP before starting trading to get the approximate value function. This is generally calculation intensive.
- **Online Computation:** This requires solving QPs at each time step to get the optimal policies at each step. This needs to be done during the trading and it is generally very fast to solve. t

B. Model Predictive Control

It is another possible sub-optimal solution to our optimisation problem. The idea here is that, as the future returns are unknown, to find the optimal policy at time t, we are unaware of the future returns. So we replace the future returns by their expected values. Now the resulting problem is a convex problem and can be easily solved. Now let $\mathbb{E}\boldsymbol{r}_t = \bar{\boldsymbol{r}}_t$, then the optimal policy $\psi^t(\boldsymbol{p}_t)$ can be solved from:

$$\min_{\boldsymbol{l}_\tau, \boldsymbol{g}_\tau} \sum_{\tau=t}^{T} c_\tau(\boldsymbol{l}_\tau, \boldsymbol{g}_\tau)$$

$$\text{subject to } \boldsymbol{l}_{\tau+1} = diag(\bar{\boldsymbol{r}}_{\tau+1})(\boldsymbol{l}_\tau + \boldsymbol{g}_\tau) \text{ for } \tau = t, .., T$$

$$\boldsymbol{l}_t = \boldsymbol{p}_t$$

The above optimisation problem solves for the optimal trades $\boldsymbol{g}_t^*, ..., \boldsymbol{g}_T^*$. Now at time point t, we execute the optimal trade $\boldsymbol{g}_t^*$ and then we solve the problem again starting from time t+1 using the current portfolio value $\boldsymbol{p}_{t+1}$. It may look like a very crude approximation, but as we reach closer to the terminal time, the approximation becomes more accurate. The above computation needs to be done online during the trading period. This computation might be infeasible if T is large. To combat this problem, we sometimes truncate the MPC to a fixed length(say

T_0) and we solve the above problem till time $\min\{t+T_0, T\}$. This is known as truncated model predictive control. Interesting thing to note is that as already proven in [1], MPC is just a special case of ADP with a particular approximation of the value function.

C. Asymmetric GARCH Model

GARCH is a process to estimate or forecast volatility of financial assets. Most to the financial institutes use this model to estimate their instantaneous risk and take decision according to this. But distribution of GARCH model is symmetric so they fail to capture leverage effect in their model. To encounter this problem, they extend this model to non-linearity. Out of many non-linear extension we use GJR model. It is represented by GJR(P,Q) where P is degree/maximum lag of GARCH coefficient in the model and Q is degree/maximum lag of ARCH and leverage coefficient in the model. Other than that we can also specify the distribution which our returns follow like Gaussian, student t-distribution etc. As already given in [3], mathematically we can write it as :-

$$y = \mu + \epsilon_t$$

where, $\epsilon_t \sim \mathbb{N}(0, \sigma_t^2)$

$$\sigma_t^2 = C + (\alpha_1\sigma_{t-1}^2 + ... + \alpha_P\sigma_{t-P}^2) + (\beta_1\epsilon_{t-1}^2 + ... + \beta_Q\epsilon_{t-Q}^2) \\ (\gamma_1\epsilon_{t-1}^2\mathbb{I}(\epsilon_{t-1} < 0) + ... + \gamma_Q\epsilon_{t-Q}^2\mathbb{I}(\epsilon_{t-Q} < 0))$$

where C is constant, α's are the GARCH lag coefficient, β's are the ARCH lag coefficients and γ's are the leverage lag coefficients.

Now in order to update our estimates of future return statistics, we employ GJR model. We used an already implemented function in matlab econometrics toolbox to use the GJR model as given in [4]. In ADP and MPC policy we used GJR model to forecast the volatility of our financial assets and optimize according to current stress level of market.

In ADP, as we know that SDP is offline process, so in SDP we forecast our volatility for whole time period once before optimization and then find value function according to the predicted volatility. And during online computation, we forecast our volatility for just next step and take decisions according to that.

In MPC, everything is optimized online. No computation is done offline. So, we need to forecast volatility for the entire remaining period in online process and then use these predictions in the objective function for the MPC policy.

III. Data and Parameters

A. Real Data

We take the initial and terminal portfolio of 0.We downloaded daily stock prices of N = 30 financial securities from here. This securities comprise of 30 major stocks from the Nifty-50 pool like HDFC,AXIS,WIPRO,TCS etc. We took around 900 data points over a period of 4 years(2017-2021). We converted the price series to daily log returns. Using the log returns data, we calculated the sample version of mean vector and variance - co-variance matrix(using correlation function) of the normal distribution. We then processed the entire data and stored it column-wise into a single csv file (log_returns_full.csv).

B. Parameters selection/generation

We have generated the proportionality constants for the costs as

$$\kappa = 0.005 * rand(n, 1)$$

$$s = 0.001 * rand(n, 1)$$

$$q = .01 * rand(n, 1)$$

$$\alpha = .5$$

$$\eta = .3$$

We used these values because absolute trading cost is nearly $.1\%$, shorting cost must be in the range of loan rate which is nearly 15% per year so for each trading day it will be in the range of $.05\%$, quadratic cost is only for penalising large amounts of trade, risk penalty cost define our risk averse behavior and lastly the leverage limit constant puts a limit on taking too much leverage. Note here that we have used random functions to generate these constants because we wanted different proportionality constants for different securities.

IV. Simulations

We take T= 100 for our simulations. We split the data-set into 2 parts, Train and Test data. For the train data, we took all the data except the latest 300 days. The latest 300 days are left out as a test set. We split the test set into 3 parts of size 100 each.We call these periods 1,2 and 3. We use the train data to tune for the mean and co-variance matrices of log returns and also in the volatility estimation using GJR model. We then test the model on the 3 periods and compare the performances of the models for different constraints.

The different constraints we test for are:

- Unconstrained
- Long only
- Leverage Limit

We have already explained them above.

For the coding part, we have taken the help of code given in [6], cvx v1.22 given in [7], and cvxgen given in [8]. We modified the code in [6] to work with real data as it was written keeping returns as random. We also added the dynamic volatility using the GJR model function in [4] in the econometrics toolbox.

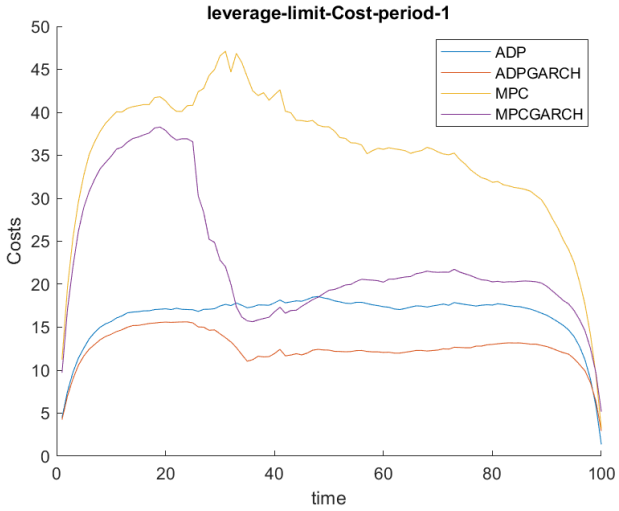
The optimization techniques we tried are ADP, MPC, ADP-GARCH and MPCGARCH. The online running time of MPC was the fastest, followed by MPCGARCH, ADP and then ADP-GARCH.However in the offline process, ADPGARCH is faster than ADP by around 10-15%. The times for the GARCH variants were comparatively higher as they required extra computations as compared to the normal variants. The time taken (in seconds) on our system by the different methods is as given:

Time Taken(sec)	MPC	ADP	MPCGARCH	ADPGARCH
Unconstrained	1.3	780.87	239.85	604.63
Leverage Limit	5.42	664.83	278.64	644.69
Long Only	4.48	582.54	265.2	528.2

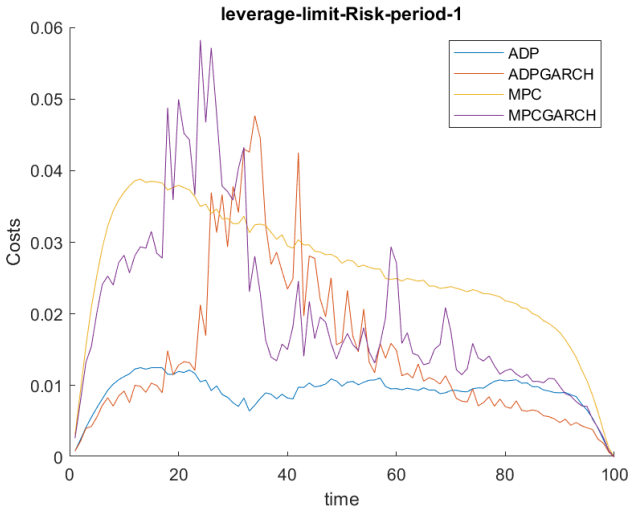
Note that this time might vary depending upon the machine being used. We have a Ryzen 5 CPU with a GTX1650 GPU.

We have tried to compare various strategies based on their cash-in and the risk penalty. Lower the cash-in , the better. Net Negative cash-in would imply that we have earned some profit overall, and positive cash-in would imply a loss. Lower the net risk penalty, the better.

V. Observations



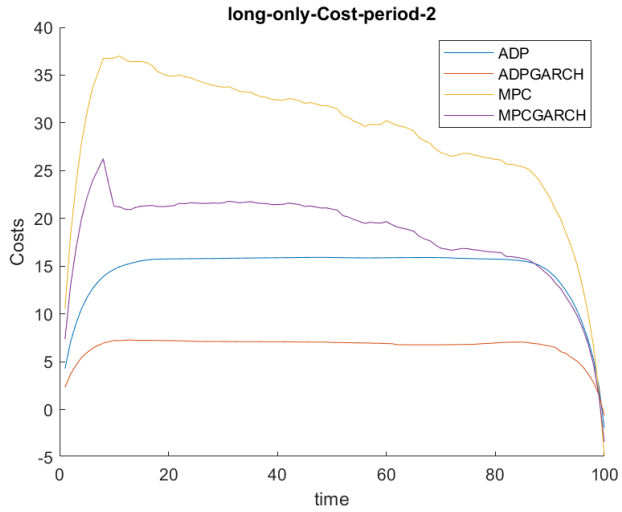
(a) comparing costs of different models



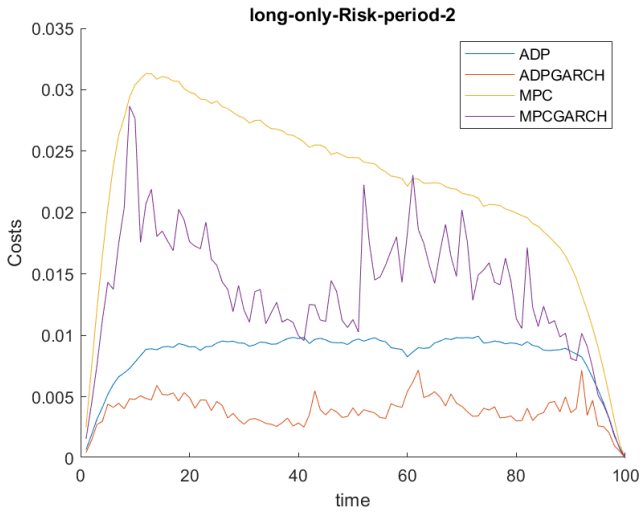
(b) comparing risk penalty of different models

Fig. 1: Under leverage limit constrain over period 1

We are showing plots for some cases here for reference. All the plots are available in the code file. Fig 1(a)(cumulative cost) and Fig 1(b)(daily risk penalty) is simulation result of leverage limit constraint over period 1 which is the period when the financial market crashed due to corona-virus pandemic. As a result, stress/volatility in the stock market drastically increased. In above figure it is evident that the normal methods i.e., ADP and MPC do not react because they are not considering that volatility of stock market is changing(increasing) with time whereas in GARCH methods, they are reacting to the high volatility. This is why they are trying to cash-out as soon as they as realising it. The modified GARCH helps them to reach to such sudden events in the market. But reaction of ADPGARCH and MPCGARCH is different in magnitude because in ADPGARCH most of calculation is offline(before starting trading period), so they are not reacting much whereas in MPCGARCH the entire optimization is done in online mode(during trading interval) so they are reacting more intensely. Also evident in the risk penalty, there are heavy spikes in the GARCH methods due to such extraordinary volatility in the market.



(a) comparing costs of different models

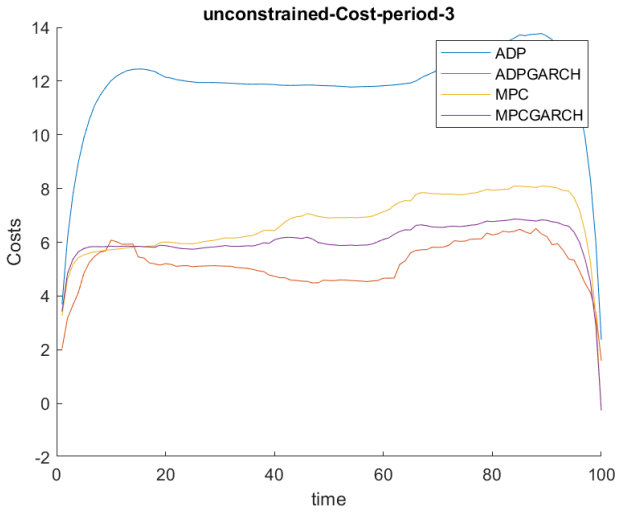


(b) comparing risk penalty of different models

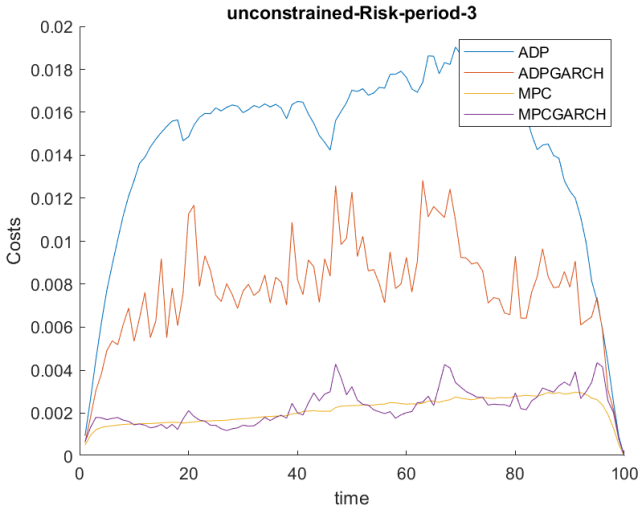
Fig. 2: Under long only constrain over period 2

Fig 2(a)(cumulative cost) and Fig 2(b)(daily risk penalty) is simulation result of Long only constraint over period 2(When stock market is recovering).As a result, volatility is less than normal condition and no small event is much affecting stock market. So, all the models seem to be behaving similarly but they

are not allowing same amount of investment in the market. For GARCH model as the earlier period(period 1) was much more volatile so they allow less investment than normal models.



(a) comparing costs of different models



(b) comparing risk penalty of different models

Fig. 3: Under no constraint over period 3

Fig 3(a)(cumulative cost) and Fig 3(b)(daily risk penalty) is simulation result of unconstrained optimization over period 3(When no impact of previous event is present and no big event happened). Unlike non-GARCH models, GARCH models are both buying and selling securities during middle intervals according to market movement and try to find best position according to market movement.

Other than that in risk penalty we can see that ADPGARCH risk penalty is greater than others(ADP and MPCGARCH) when their position is almost the same. Because it might be expecting some event might happen in the near future which might cause a crash similar to the covid crash(Maybe its expecting a second wave). This is because of what it has learned from the past .MPCGARCH might also expecting that's why it also showing some movement in risk penalty, but as it is optimized during online mode, its reacting less to what has already happened in the past.

VI. Results and Conclusions

We have listed all the results in TABLE I. It contains parameters like

- **Maximum cost :** It corresponds to the maximum cash-in during the entire period. It essentially corresponds to total net money being invested in the entire period.
- **Total Cost :** It means the net amount given into the market. Negative total cost means that we have gained money and vice versa.(Lower the better)

We can see from the table that even the normal methods(ADP, MPC) appear to be working well when applied to real world data. The appear to given consistent positive returns with exception to period 1. If trained well, these methods can even be used in real portfolio management applications.

As we can see from TABLE I, although the GARCH variants are more responsive to the market activites, the normal methods seem to out perform the GARCH variants. The risk penalties are more appropriate modelled by the GARCH models.

VII. Future Work

We can work on some better ways to inculcate market volatility in the model. We can also work on speeding up the run-time of the algorithms as they are not very well optimised to run on powerful systems. The algorithms take about 10% of the CPU usage. They can be allocated more cores and the entire process can be speed-en up. It can also be allocated to GPU which can do it in a parallel manner.

References

[1] Boyd, Stephen. (2014). Performance Bounds and Suboptimal Policies for Multi-Period Investment. 1. 1-72. 10.1561/2400000001.

[2] Wang, Y., O'Donoghue, B., and Boyd, S. (2015), Approximate dynamic programming via iterated Bellman inequalities. Int. J. Robust Nonlinear Control, 25, 1472– 1496. doi: 10.1002/rnc.3152.

[3] Dima Alberg, Haim Shalit & Rami Yosef (2008) Estimating stock market volatility using asymmetric GARCH models, Applied Financial Economics, 18:15, 1201-1208, DOI: 10.1080/09603100701604225

[4] https://in.mathworks.com/help/econ/gjr.html

[5] https://www.investing.com

[6] https://web.stanford.edu/∼boyd/papers/port_opt_bound.html

[7] http://cvxr.com/cvx/

[8] https://cvxgen.com/docs/index.html

		Period 1			Period 2			Period 3		
Constraint	Model	Max Cost	Tot Cost	R. Penalty	Max Cost	Tot Cost	R. Penalty	Max Cost	Tot Cost	R. Penalty
Leverage L.	ADP	18.566	1.378	0.918	18.280	-2.189	1.265	19.718	1.243	1.320
	ADPG.	15.619	2.924	1.447	8.957	-0.240	0.658	9.560	0.921	0.653
	MPC	47.099	3.206	2.598	38.436	-3.977	2.727	40.413	1.414	2.693
	MPCG.	38.283	5.182	2.016	25.905	-2.456	1.607	25.678	1.164	1.634
Long Only	ADP	16.654	1.784	0.590	15.914	-1.914	0.840	15.953	-0.209	0.857
	ADPG.	14.929	3.071	0.993	7.268	-0.620	0.386	7.717	-0.077	0.420
	MPC	47.070	4.166	2.157	37.016	-4.823	2.231	38.098	-1.941	2.226
	MPCG.	37.451	6.111	1.788	26.249	-3.413	1.372	25.490	-1.414	1.403
Unconst.	ADP	11.416	0.828	0.971	11.377	-2.426	1.371	13.773	2.374	1.436
	ADPG.	11.303	2.533	1.577	4.715	-0.429	0.743	6.512	1.589	0.776
	MPC	9.927	0.546	0.214	7.821	-1.579	0.213	8.098	-0.271	0.211
	MPCG.	7.756	1.432	0.418	6.781	-1.302	0.249	6.863	-0.266	0.227

TABLE I: Detailed Comparison of Various Methods