

Rating Transition Matrix Project

Project Lab, The University of Chicago

Ki Hyun, Charlotte Wang, Peizhe Huang, Hannah Wang

Outline

1. Project background
2. Bridging Past and Future Project
3. Rating Transition Matrix
4. Summary of DA / WA / QOG / EM algorithms for time-scaled RTM
 - a. Principles of the approaches
 - b. RTM calculated and error check with current RTM
 - c. Pros / Cons of each method
5. Conclusion

1. Project Background

❖ Input:

- Historically estimated annual rating transition matrices published by rating agencies

❖ Approach:

- Estimate time-scaled rating transition matrix through four algorithms:
 - Diagonal Adjustment (DA)
 - Weighted Adjustment (WA)
 - Quasi-Optimization Generator (QOG)
 - Expectation–Maximization (EM)

❖ Goal:

- Estimate a time-scaled Rating Transition Matrix (RTM) from annual data
- Ensure reliable probabilities for transitions during time-scaling

2. Bridging Past and Future Project

- **Past efforts:** Executed CVA and FVA analysis under rating-dependent CSA and CDS; confronted difficulties with granular rating factor simulations on a weekly grid
- **RTM Scaling Issues:** Encountered mathematical inconsistencies when scaling the probability matrix to weekly measures
 - Negative probabilities
 - Complex numbers
- **Current Goal:** Address these issues by adopting methods (DA / WA / QOG / EM) to estimate a probability matrix at any time scale, ensuring mathematical consistency and overcoming the issues faced with granular scaling of RTMs

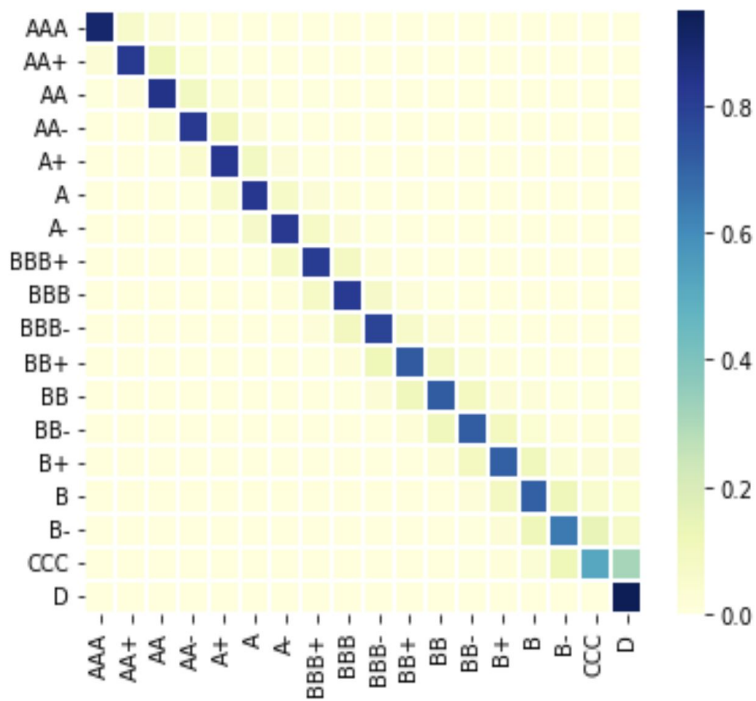
3. Rating Transition Matrix

- **Definition of RTM:**
 - Rating transition matrices represent the probabilities of migrating from one credit rating to another over a certain period, typically one year
- **RTM's Role in xVA Calculation:**
 - RTM is crucial for determining xVA (including CVA and FVA from previous quarter's project) as it influences the estimation of CDS survival probabilities and CSA thresholds
- **Advanced Methods Need:**
 - Techniques like WA / DA / QOG / EM provide more robust modeling of rating transitions than the standard power root method

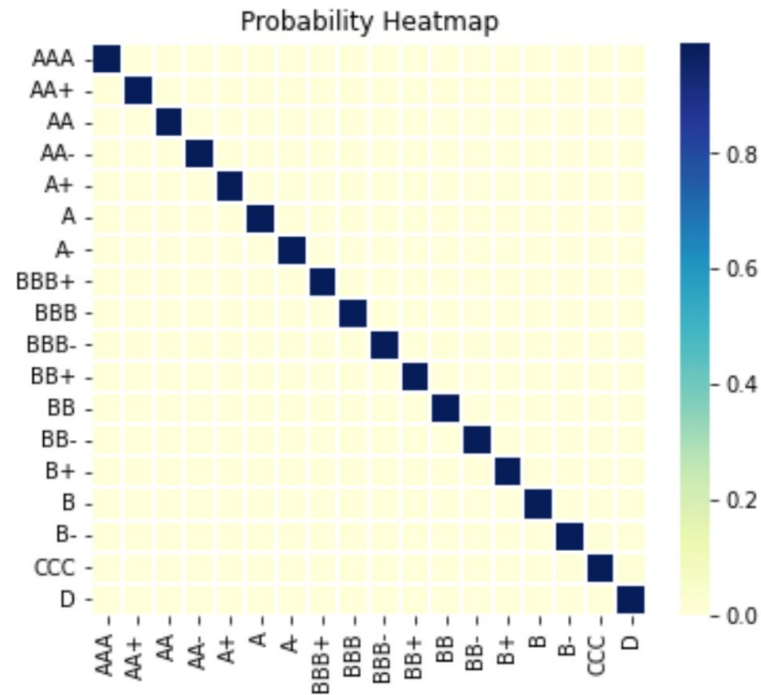
3. Rating Transition Matrix

Comparison of Probability grid: Original RTM vs Weekly Scaled by 52nd power root

Heatmap of **original** RTM

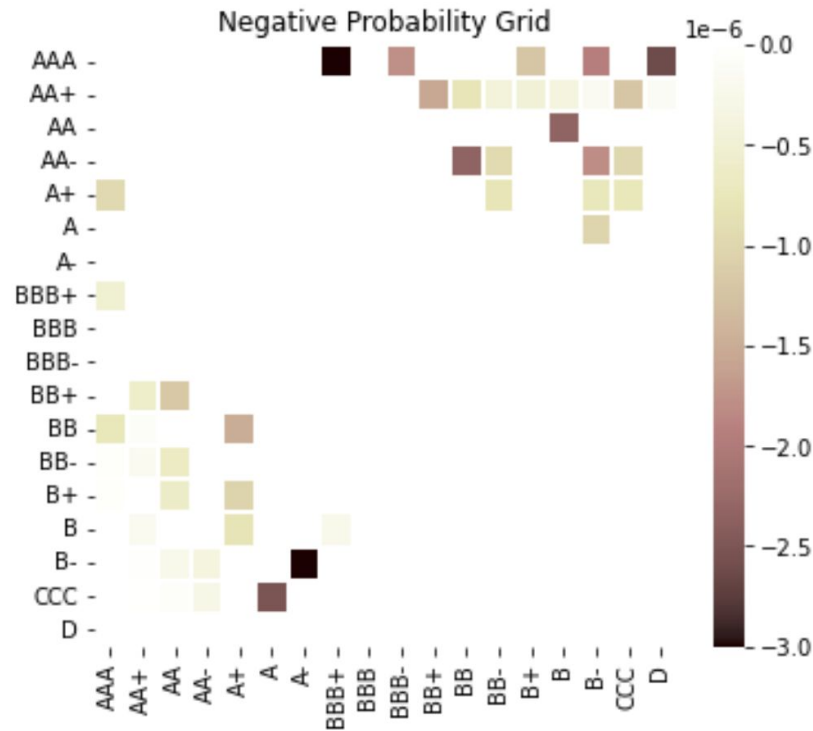


Heatmap of **weekly scaled** RTM

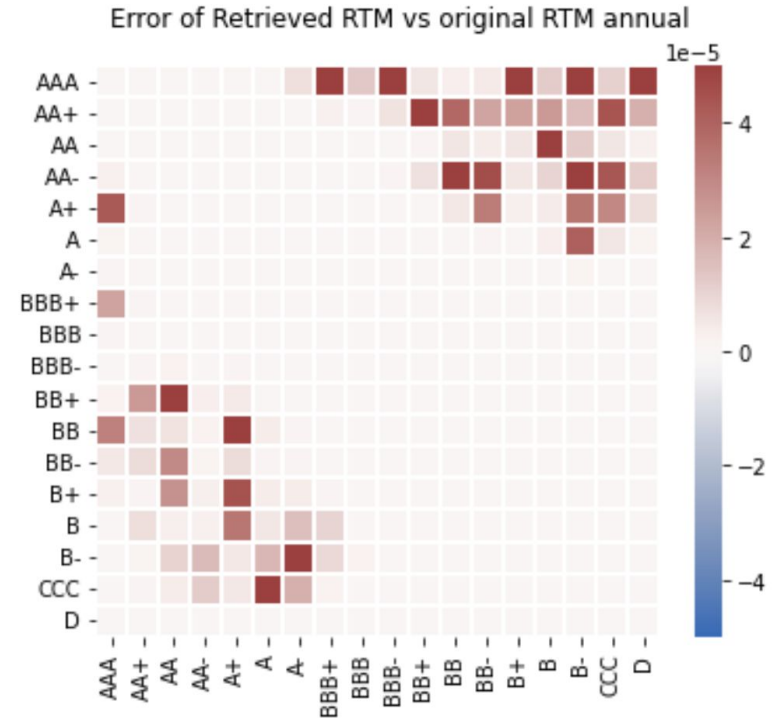


3. Rating Transition Matrix

Weekly scaled RTM by 52nd power root



Retrieved annual RTM floored at zero



4. Algorithms for Time-Scaled RTM

4.1 Generator

By calculating matrix exponentials of a generator matrix Q , we can obtain a transition probability matrix for **an arbitrary time period**.

Definition:

$$q_{ij} = \lim_{\Delta t \rightarrow 0} \frac{\mathbb{P}(X(\Delta t) = j | X(0) = i)}{\Delta t}$$

$$Q = \begin{bmatrix} -q_1 & q_{12} & \cdots & q_{1K} \\ q_{21} & -q_2 & \cdots & q_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ q_{K1} & q_{K2} & \cdots & -q_K \end{bmatrix}$$

Properties:

- $\sum_{j=1}^K q_{ij} = 0$, for $1 \leq i \leq K$
- $0 \leq -q_{ii} = q_i \leq \infty$, for $1 \leq i \leq K$
- $q_{ij} \geq 0$ for $1 \leq i, j \leq K$ with $i \neq j$.

$$\begin{aligned} \mathbf{P}(t, s) &\approx (\mathbf{I} + \Delta t \mathbf{Q})^m \\ &= \left(\mathbf{I} + \frac{(s-t)}{m} \mathbf{Q} \right)^m. \end{aligned}$$

However, because of the embeddability problem mentioned before, generator matrix **does not exist** sometimes.

4.2 Diagonal Adjustment (DA) and Weighted Adjustment (WA)

Principle:

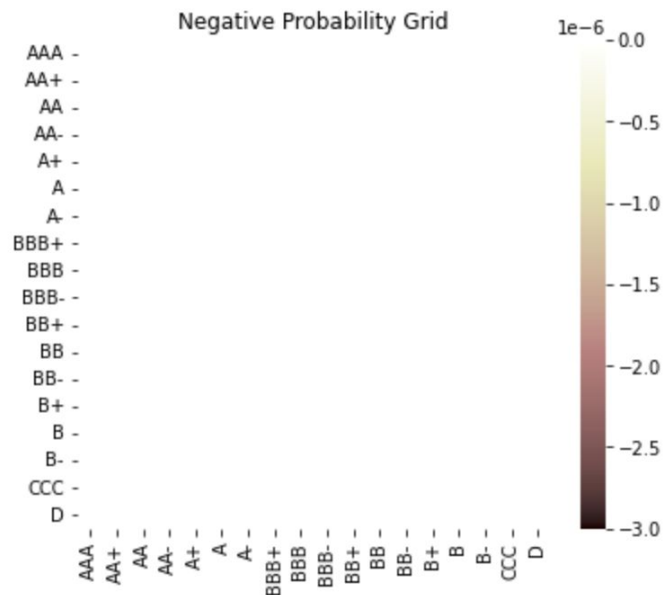
Both two algorithms take a generator and **floor the generator at zero**.

- DA adjusts only the **diagonal components** to offset the impact from flooring (i.e. all negative off-diagonal elements in each row are replaced with zero and the diagonal elements are re-calculated as the negative sum of the non-diagonal elements).
- WA adjusting each row of the generator by **weighted scale** (i.e. any negative off-diagonals are set to zero, but all the other non-zero elements, together with the diagonal elements, are modified to ensure that the matrix has rows that sum to zero).

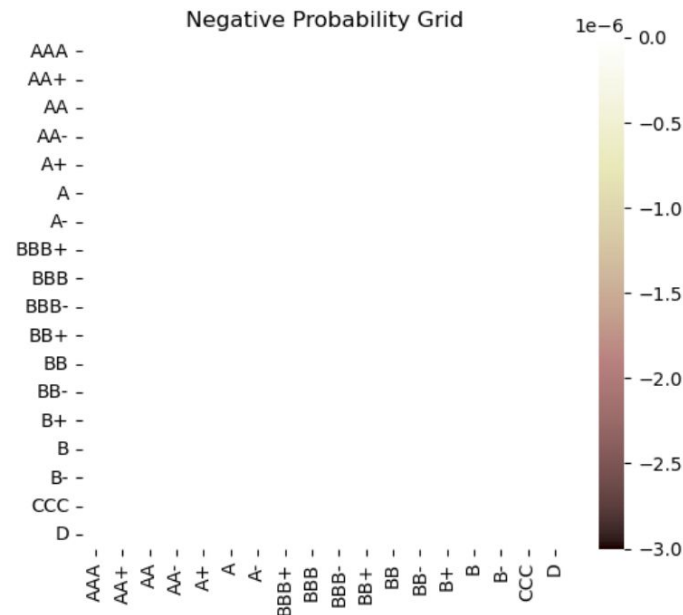
Both algorithms make the new generator **meet requirements** and be an RTM generator.

4.2 Diagonal Adjustment (DA) and Weighted Adjustment (WA)

- Probability Grid



Probability Grid of DA

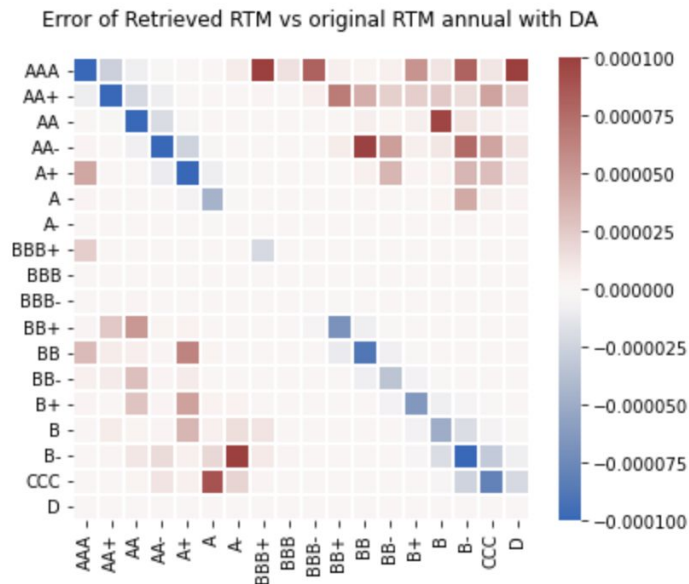


Probability Grid of WA

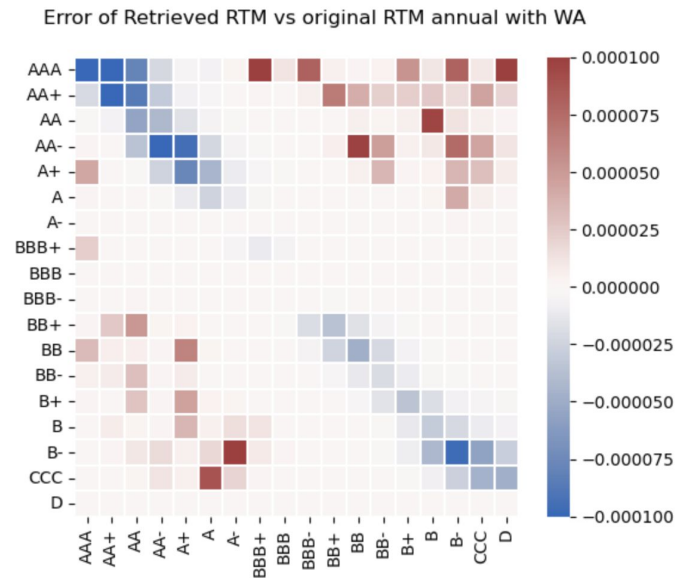
With both DA and WA algorithms, we can obtain weekly scaled matrix **without negative probabilities**.

4.2 Diagonal Adjustment (DA) and Weighted Adjustment (WA)

- Error Analysis



Error with DA



Error with WA

We can see in both algorithms, probabilities in top right area and bottom left area are added up for **compensation of lost diagonal probabilities**. Compared to DA algorithm, WA algorithm gives compensation not only at diagonal components, but also around the diagonal, because DA adjusts only the diagonal components, while WA adjusts each row by weighted scale. ¹²

4.2 Diagonal Adjustment (DA) and Weighted Adjustment (WA)

Pros and Cons:

- ❖ **Pros:** They are **easy** to be implemented and understood. They do **solve the problem** and make probabilities be positive.
- ❖ **Cons:** They do not make any mathematical senses and **drop information**. We are not sure how they work when encountering **complex numbers**.

4.3 Quasi-Optimization Generator (QOG)

Principle:

The QOG method incorporates **distance-minimizing optimization** and aims to find a generator matrix that approximately minimizes the sum of squared deviations between the observed transition counts and the expected transition counts implied by the estimated generator matrix.

$$\min_{Q \in Q} \left\| Q - \log(\tilde{P}) \right\|$$

"Quasi" optimization

- it actually minimizes the distance between the generator and the log of the transition matrix for computational simplification

4.3 Quasi-Optimization Generator (QOG)

Optimization:

The optimization in the QOG method is subject to the constraints that the off-diagonal entries of the generator matrix must be non-negative, and the rows of the generator matrix must sum to zero. These constraints ensure that the estimated generator matrix is valid according to the definition of a continuous-time Markov process.

(condition 2.5 in Bank of Japan paper, closed on each row)

4.3 Quasi-Optimization Generator (QOG)

- **Code Overview** (Bank of Japan paper page 14-15)

```

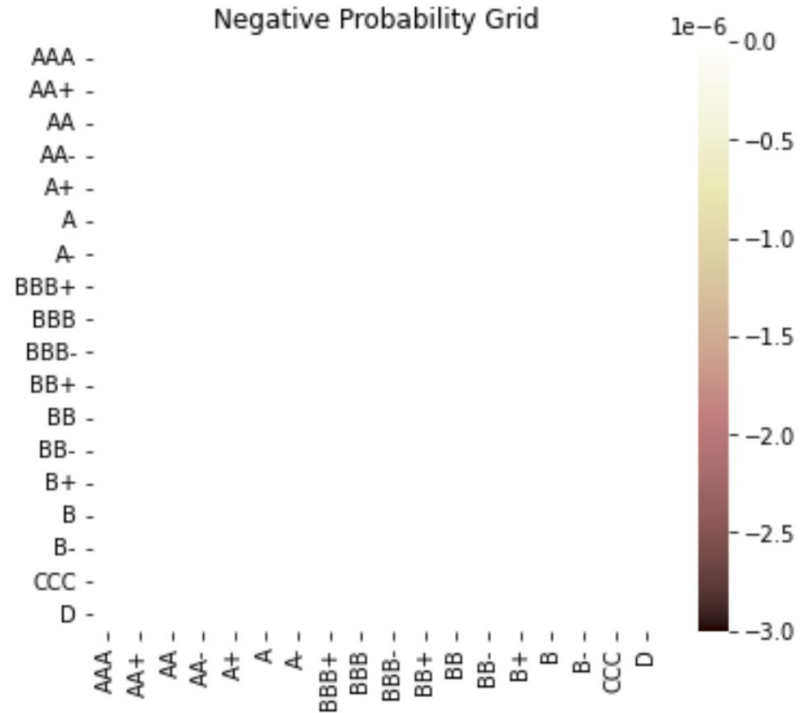
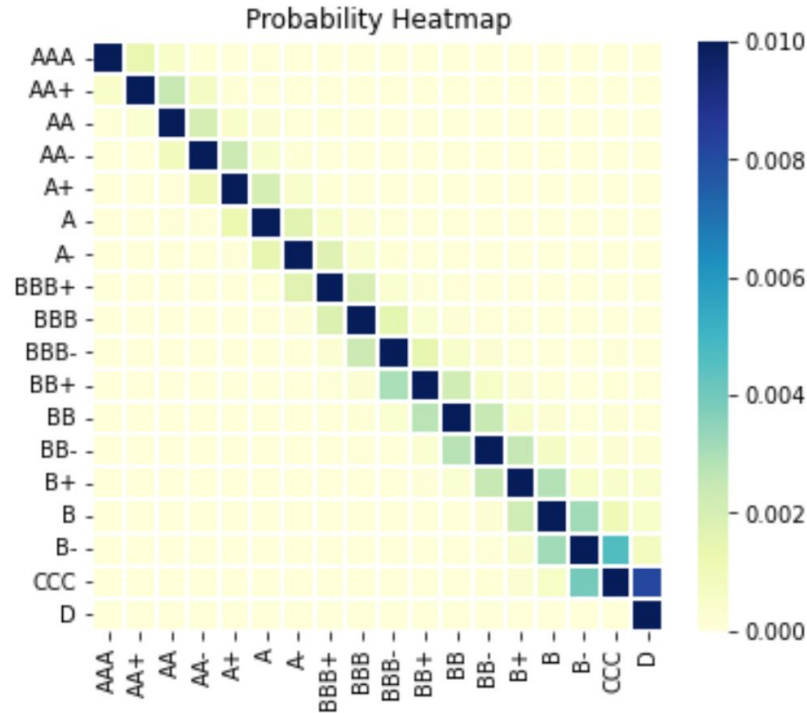
1 def qog_algorithm(p_matrix):
2     # Step 1: Construct b
3     K = p_matrix.shape[0]
4     a = lg.logm(p_matrix)
5     lambd = -np.mean(a)
6     b = a + lambd*np.ones((K,K))
7
8     # Step 2: Permute b
9     sorted_indices = np.argsort(b, axis=1)
10    a_hat = np.array([b[i,sorted_indices[i]] for i in range(K)])
11
12    # Step 3: Find m*
13    q_matrix=np.zeros((K,K))
14    for i in range(0,K):
15        m_star = 2
16        while m_star <= K-1 and ((K-m_star+1)*a_hat[i,m_star] - np.sum(a_hat[i,m_star:]) - a_hat[i,0]) < 0:
17            m_star += 1
18        # Step 4: Construct z*
19        z_star = np.zeros(K)
20        z_star[1:m_star] = 0
21        z_star[m_star:] = a_hat[i,m_star:] - ((np.sum(a_hat[i,0]) + np.sum(a_hat[i,m_star:]))) / (K-m_star+1))
22        z_star[0] = a_hat[i,0] - ((np.sum(a_hat[i,0]) + np.sum(a_hat[i,1:]))) / (K-m_star+1))
23        # Step 5: Inverse permutation to get the generator matrix Q
24        q_matrix_row = np.zeros(K)
25        for j in range(K):
26            q_matrix_row[sorted_indices[i][j]] = z_star[j]
27        q_matrix[i,:] = q_matrix_row
28    return q_matrix
29

```


4.3 Quasi-Optimization Generator (QOG)

- Probability Grid

Heat Map of Weekly Rating Transition Matrix with QOG (S&P adj)



4.3 Quasi-Optimization Generator (QOG)

- **Error analysis**
 - RTM generated minus RTM given
 - **RED** - overestimate / **BLUE** - underestimate

Higher ratings (e.g., AAA, AA+, AA) -> substantial errors

Lower ratings (e.g., D) -> smaller or no errors

Lack of a diagonal pattern in the QOG method's errors signifies a **more even distribution of error**

Better at generalizing across various ratings compared to the DA / WA method

Nature of credit events (high credit ratings relatively more stable / changes are difficult to predict accurately)



4.3 Quasi-Optimization Generator (QOG)

- **QOG method & Convergence Check**
 - QOG method **does NOT** suffer from convergence issues like EM and MCMC methods
 - QOG formulates the estimation problem as a quadratic programming problem, ensuring convergence based on quadratic programming algorithms
 - Careful implementation and validation of the QOG method are necessary to ensure reliable results
 - Potential errors or convergence issues can arise from data preprocessing, initial value selection, and optimization techniques in the overall RTM estimation process

4.3 Quasi-Optimization Generator (QOG)

Pros and Cons:

❖ Pros:

- more **robust** estimation of the generator matrix (distance-minimizing)
- more **accurate** when scaling to different time intervals (estimates the generator matrix directly)

❖ Cons:

- **computationally intensive** compared to previous methods
 - especially for large state spaces due to solving optimization problem
- **sensitive** to the choice of the optimization algorithm

4.4 Expectation-Maximization (EM)

- **Principle:**

The EM algorithm is an iterative method used to find maximum likelihood estimators when the data is incomplete either in the form of missing data or latent variables. The algorithm works by iterating between the Expectation-step (E-step) and the Maximization-step (M-step) until the convergence of the likelihood function is achieved.

4.4 Expectation-Maximization (EM)

- **E-Step - Estimating hidden variables:**

Assuming some initial set of parameters, the E-step is to construct a complete data by replacing unobserved data with their respective expected values conditional on the observed data. In the E-step, we use the current estimates to infer the missing information.

4.4 Expectation-Maximization (EM)

- **M-Step - Maximizing likelihood:**

The M-step is to implement the maximum likelihood estimation using the complete data constructed in the E-step. This step maximizes the likelihood of the observed and imputed data. New estimates updated by the M-step are then used as the parameters in the next E-step.

4.4.1 Particular Problem: Generator Estimation from RTMs

- For our problem, the observed process is a discrete time Markov Chain for each obligor, e.g. Obligor X rated 'AAA' was observed rated 'AA' after one year, while the unobserved process to estimate is a continuous Markov Chain for each obligor.
- Parameters we wish to estimate: the entries in the generator
- The likelihood of a continuous time fully observed Markov chain with Generator Q involves :

$K_{ij}(T)$: the number of jumps from i to j in the interval $[0, T]$,

$S_i(T)$: the holding time of state i in the interval $[0, T]$

- With discrete data, we don't know the exact values of $K_{ij}(T)$ and $S_i(T)$, thus cannot solve the maximum likelihood function directly. However, using the EM algorithm, we can replace $K_{ij}(T)$ and $S_i(T)$ with their respective conditional expectation in the E-Step and then found the maximum likelihood estimator as usual in the M-Step

4.4.1 Particular Problem: Generator Estimation from RTMs

- Review of the procedure:
 - (i) Take an initial intensity matrix \mathbf{Q}
 - (ii) While the convergence criteria is not met
 - (1) E-step: calculate $\mathbb{E}_{\mathbf{Q}}[K_{ij}(T)|\mathbf{P}]$ and $\mathbb{E}_{\mathbf{Q}}[S_i(T)|\mathbf{P}]$
 - (2) M-step: set $q'_{ij} = \mathbb{E}_{\mathbf{Q}}[K_{ij}(T)|\mathbf{P}]/\mathbb{E}_{\mathbf{Q}}[S_i(T)|\mathbf{P}]$, for all $i \neq j$ and set q_{ii} appropriately
 - (3) Set $\mathbf{Q} = \mathbf{Q}'$ (where \mathbf{Q}' is the matrix of q' 's) and return to E-step
 - (iii) End while and return \mathbf{Q}

4.4.2 Implementation - Early Stopping

- Implementing early stopping can help us avoid unnecessary computations in situations where further iterations may not significantly improve the results. It is essential to strike a balance between achieving accurate estimates and avoiding excessive computational resources or time.

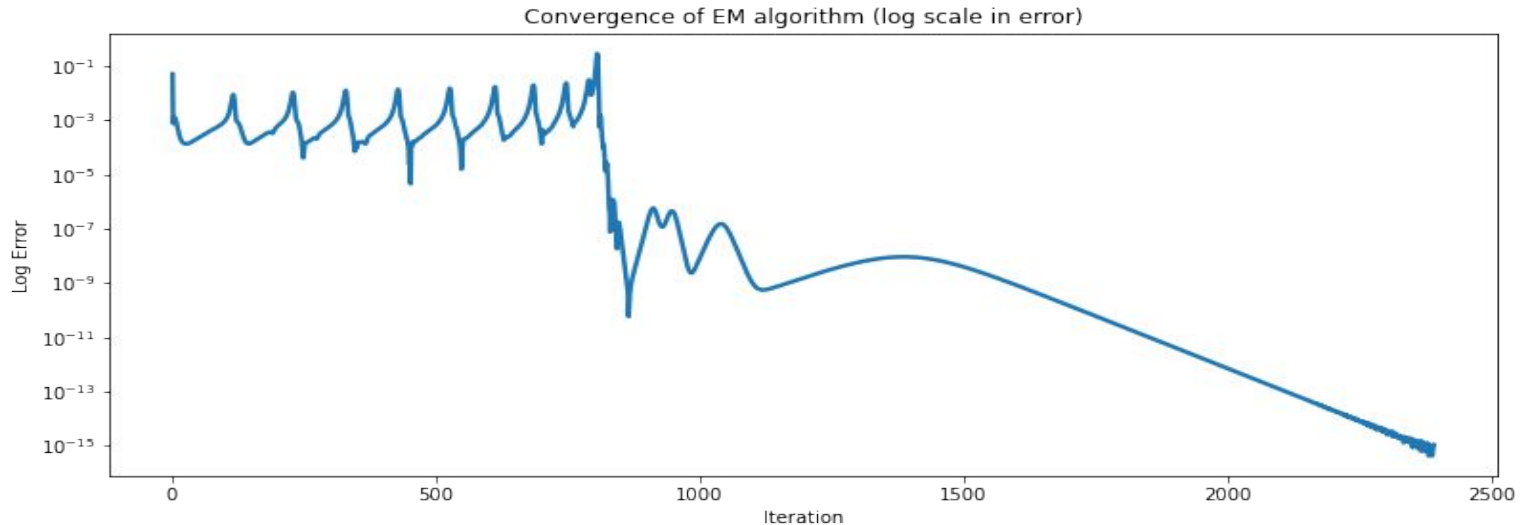
4.4.2 Implementation - Early Stopping

- Define a stopping criterion: Determine a criterion that will indicate when to stop the iterations, such as the change in the expected log-likelihood or specify a maximum number of iterations.
- With a given threshold error, we can check the convergence of the expected log-likelihood for early stopping.

```
# Check convergence of the expected log-likelihood for early stopping
logli = 0.0
for i in range(n):
    for j in range(n):
        if q[i, j] > 0 and i != j:
            logli = logli + (ENijT[i, j] * np.log(q[i, j]) - q[i, j] * ERiT[i])
loglivec.append(logli)
rel_error = abs(loglivec[-1] - loglivec[-2])/abs(loglivec[-2]) if k > 0 else np.nan
if rel_error <= epsil:
    q = qtilde
    break
```

4.4.3 Choice of initial matrix

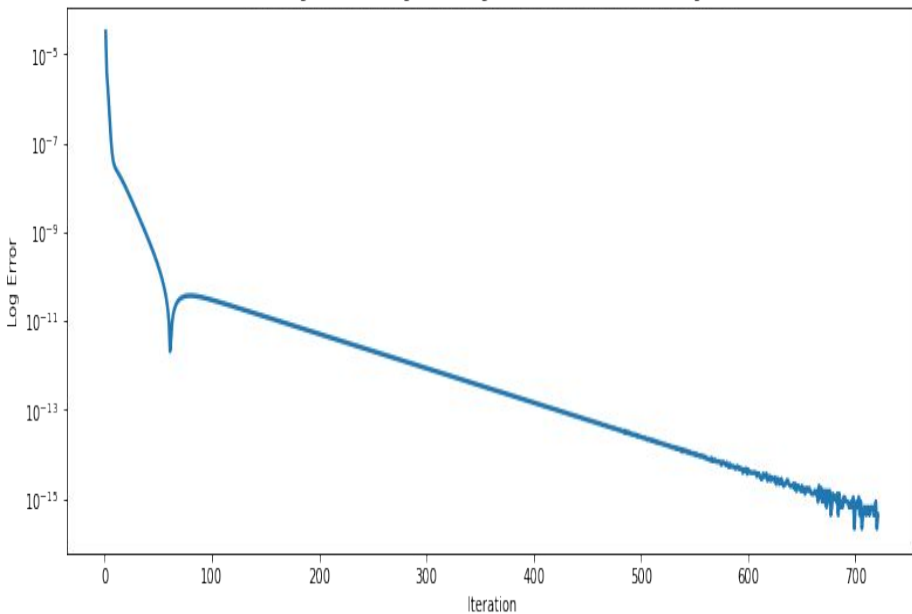
- Null initial guess - probably not good guess as take nearly 800 iterations before starting to converge



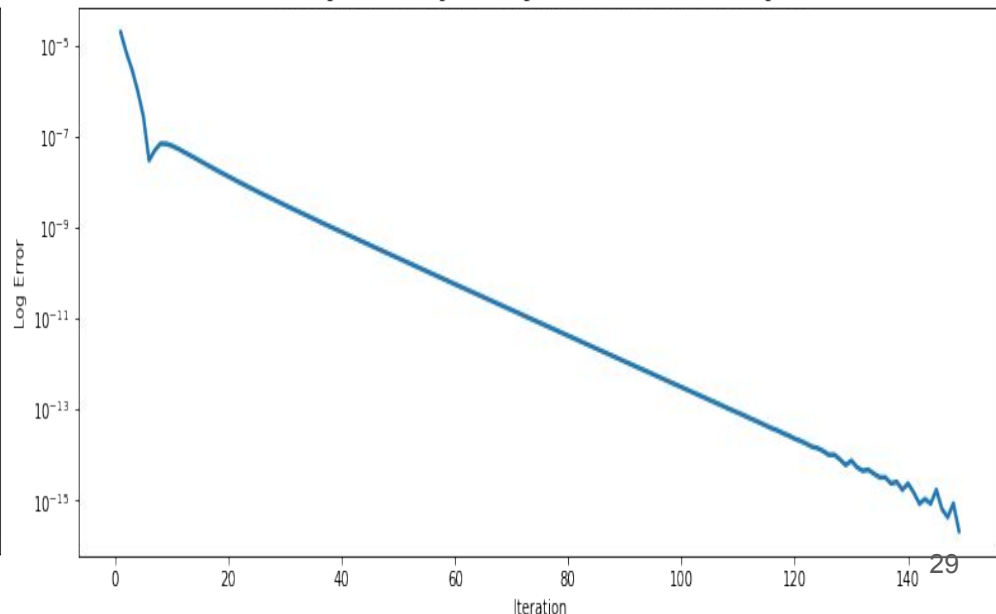
4.4.3 Choice of initial matrix

- WA as initial guess (left) / QOG as initial guess(right) - better than null guess, takes fewer iteration steps under the same threshold of error

Convergence of EM algorithm (log scale in error); WA as initial guess



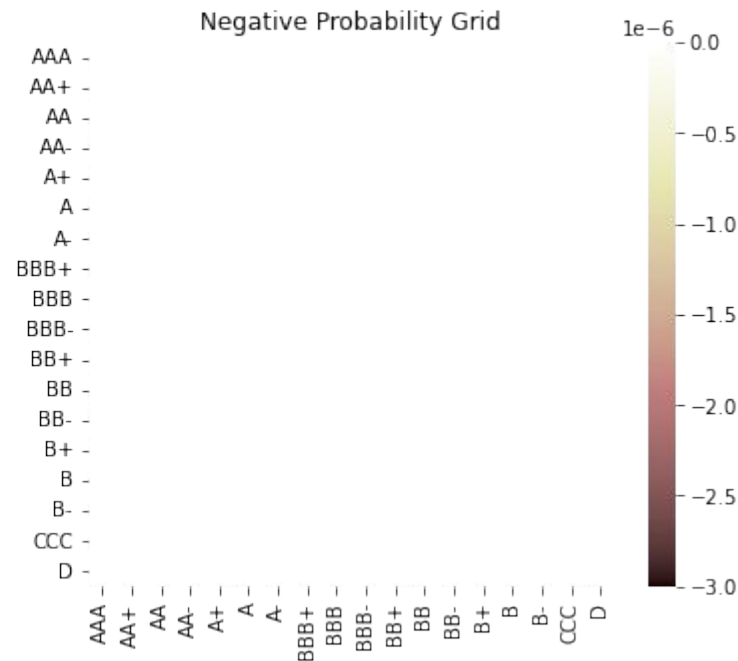
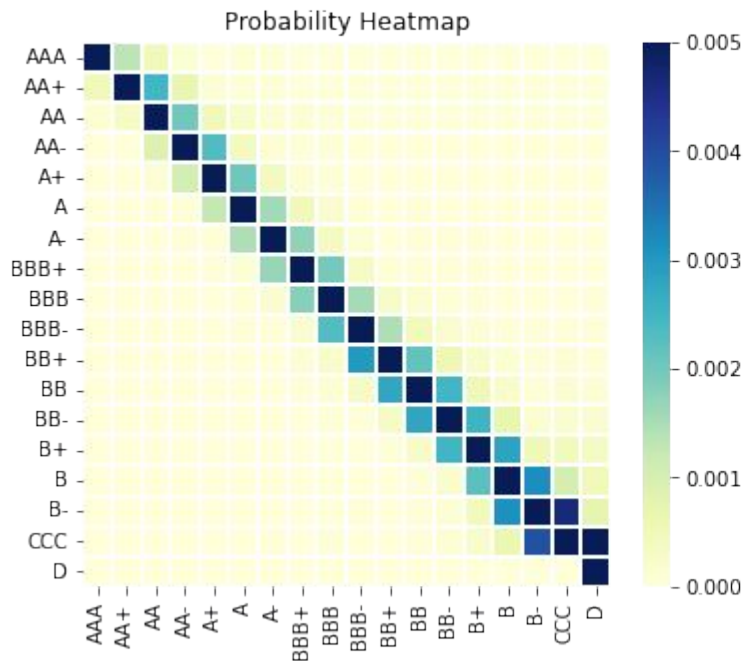
Convergence of EM algorithm (log scale in error); QOG as initial guess



4.4.4 Validation

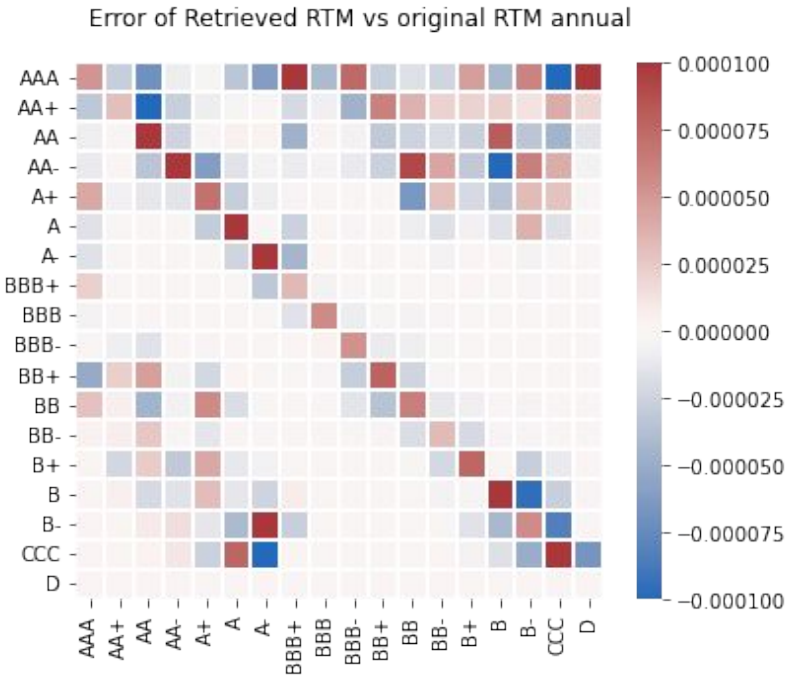
- Probability Grid (using QOG as initial guess here), the weekly transition matrix doesn't have any negative entries

Heat Map of Weekly Rating Transition Matrix with EM (S&P adj)



4.4.4 Validation

- WA as initial guess (left) / QOG as initial guess(right)



4.4.5 Pros & Cons

Pros and Cons:

❖ Pros:

- It's a statistical algorithm and embeds a **strong robustness** property for the estimator contrary to the deterministic algorithms, i.e. the likelihood is far less sensitive to small changes in the underlying RTM
- Unlike DA/WA method, which drops information by directly floored the negative entries at zero, the EM algorithm **uses all the information** (data).
- Like DA/WA/QOG, it's also **a continuous model** and provide consistent estimates.

❖ Cons:

- **Sensitivity to initialization**: Poor initial guess may lead to slower convergence or local optimization rather than global optimization . Careful initialization or employing other techniques, such as multiple random starts, may be helpful.
- **Slower** than other three methods as the EM algorithm involves complex calculations such as calculating the integrals of matrix exponentials in the expectations

5. Conclusion

❖ 5.1 Significant found:

❖ DA/WA:

- Easy to be implemented and understood
- Drop information and can not deal with complex numbers

❖ QOG:

- Provides continuous-time scaled generator, eliminating the need for convergence checks
- Computationally intensive, requires precise data preprocessing, and involves complex algorithmic steps

❖ EM:

- Continuous time scaled generator with nice statistical properties such as robustness
- Computationally expensive, involves calculations of matrix exponential integrals and thus slower than other three methods
- Sensitive to choice of initialization, and needs for convergence check

5. Conclusion

5.2 Future Steps

Continuation on **MCMC (Markov Chain Monte Carlo)** Sampling Method:
Principle:

- A type of Bayesian Inference used for estimating the generator matrix, leveraging Prior and Likelihood Distributions, and Observed Data to derive the Posterior distribution for new data sampling.

Evaluation:

- Current MCMC implementation struggles with convergence, causing widespread errors.
- Weak priors may be exacerbating the issue; computational constraints limit further iterations.
- Future improvements include revising the MCMC algorithm, strengthening convergence checks, and diversifying simulations across multiple Markov Chains.

Appendix

(4.4MCMC)

4.4 MCMC (Markov Chain Monte Carlo) Sampling Method

Another way to estimate the generator matrix is the MCMC method, which is one type of Bayesian Inference. For **Bayesian Inference**, you need the below:

- ***Prior Distribution***
- ***Likelihood Distribution***
- ***Observed Data***

With the above three, you will be able to attain a ***Posterior distribution*** of the parameters to sample new data.

4.4 MCMC (Markov Chain Monte Carlo) Sampling Method

In our case,

- **Prior Distribution:**

- **Gamma distribution** on the **non-diagonal entries** of the generator matrix [Bladt & Sorensen, 2005]
- **Endogenous** diagonal entries [Bladt & Sorensen, 2005] (i.e., *negative of the sum of each row's non-diagonal entries*)

- **Likelihood Distribution:**

- **Poisson distribution** on the *holding time* (with respect to the diagonal entries)
- **Multinomial distribution** on the *rating jump* (with respect to the non-diagonal entries)

This would be a **conjugate prior** (i.e., **posterior distribution \sim Gamma**)

[Bladt & Sorensen, 2005]

4.4 MCMC (Markov Chain Monte Carlo) Sampling Method

However,


- ***Observed data*** on **credit ratings changes** are **sparse** and **rare**
- ⇒ **No reliable observed data** on ***continuous rating changes*** to conduct direct Bayesian Inference

Solution:

⇒ **Markov Chain Monte Carlo Gibbs Sampling**

[Inamura, 2006] [Bladt & Sorensen, 2005]

4.4.1 Gibbs Sampling Methodology

- 
1. Sample ***Generator Matrix*** from the prior distribution.
 2. Sample ***holding time*** from the diagonal entries of the Generator Matrix.
 3. For holding time less than time-step (Δt):
Sample the ***rating jump*** from the non-diagonal entries
 4. After all simulations, **specify** the ***posterior distribution***
 5. Sample ***Generator Matrix***, again, from the posterior distribution.

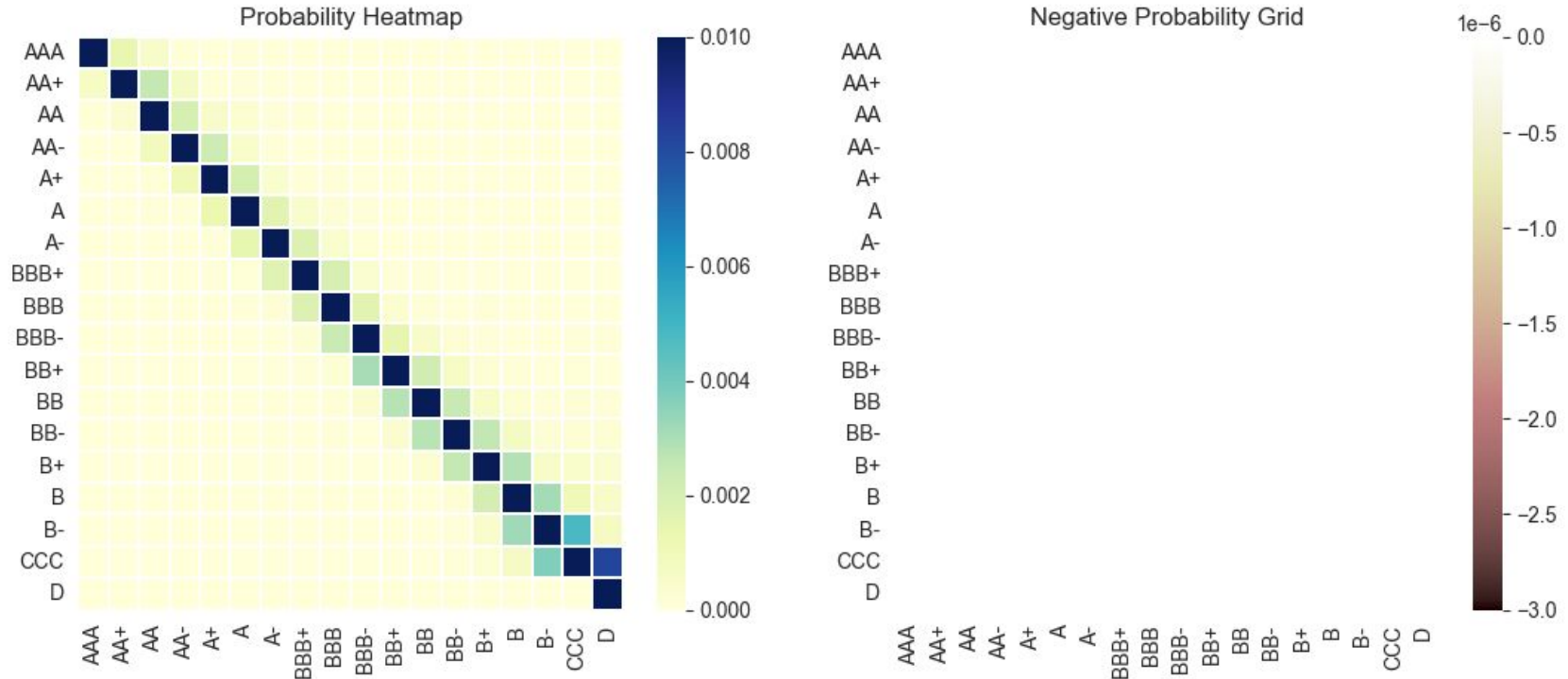
4.4.2 Tuned Prior Approach

- Fit a ***gamma distribution*** on the original generator matrix from the RTM
- Save the β or “**scale**” parameter values from the fitted gamma distribution
- Two ways to tune the α or “**shape**” parameters
 - Assume each non-diagonal entries of the **original generator matrix** as the **mean** of the gamma distribution.
 - Problem: Some of the tuned α values are **NEGATIVE**
 - Solution: **Floor** the α values to 0
 - Assume each non-diagonal entries of the **EM algorithm generator matrix** as the **mean** of the gamma distribution. (There are ***no negative tuned α values***)

* **Gamma mean = α / β**

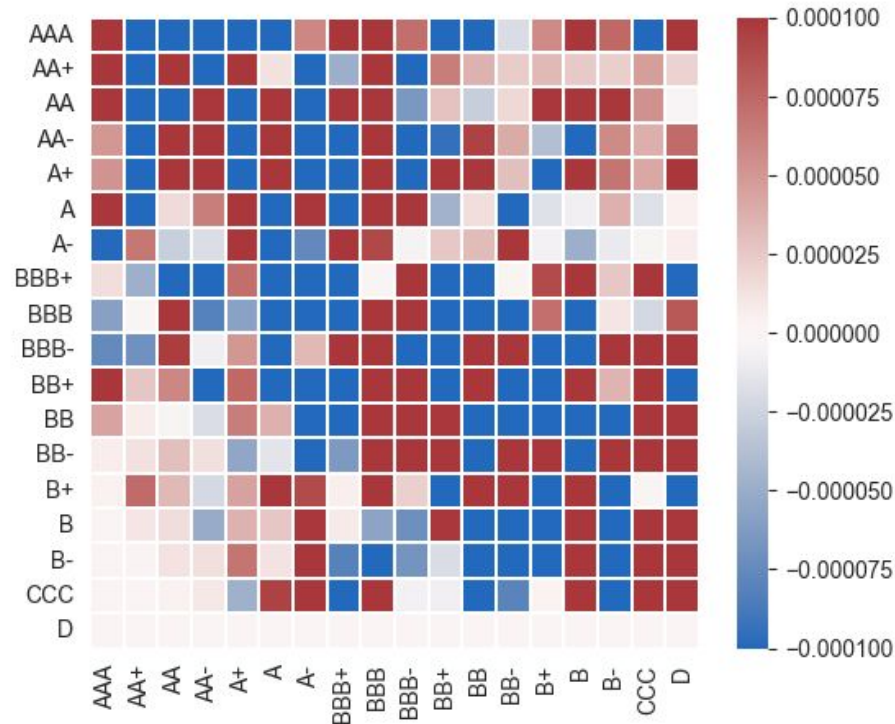
4.4.2.1 Tuned Prior (Floored) Validation [S&P 500]

Heat Map of Weekly Rating Transition Matrix with MCMC - Tuned Prior (Floored) (S&P adj)



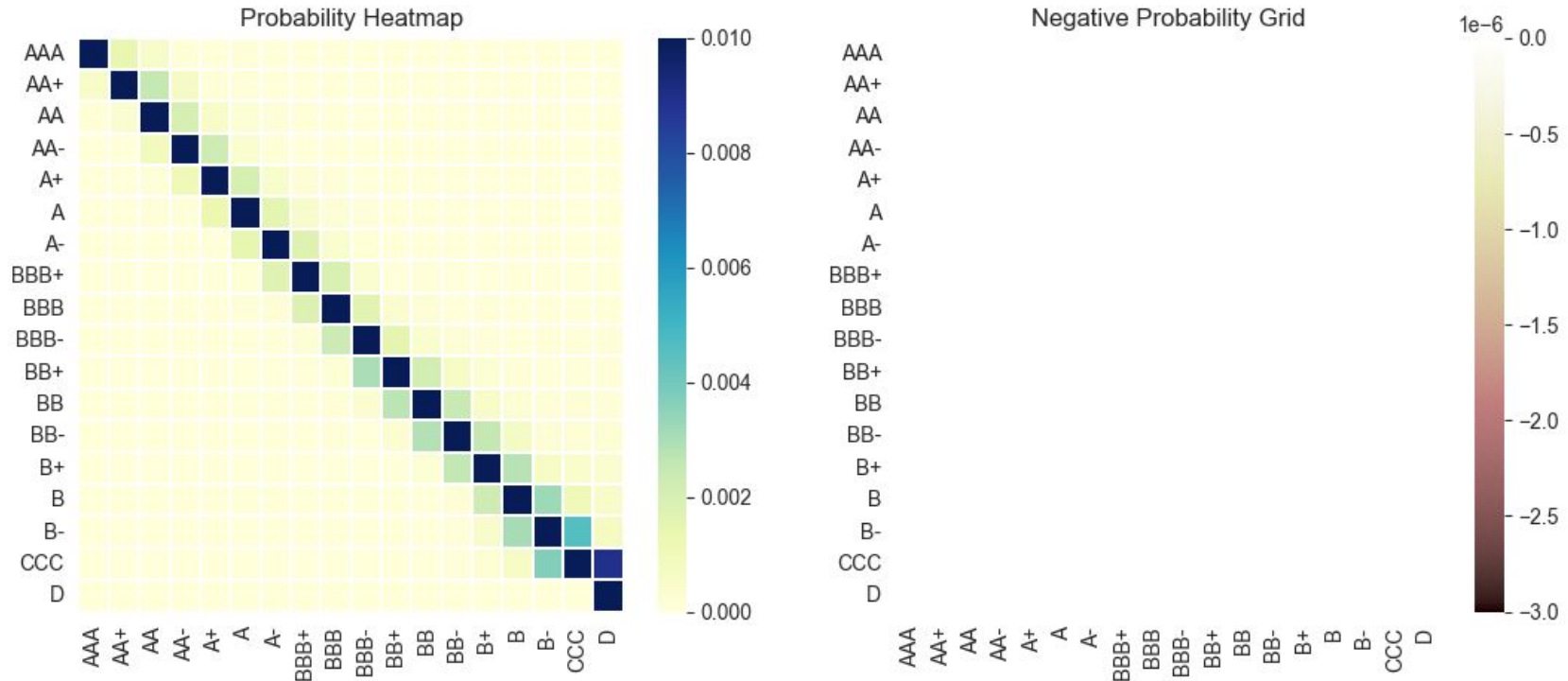
4.4.2.1 Tuned Prior (Floored) Validation [S&P 500]

Error of Retrieved RTM vs original RTM annual (Tuned Floored Prior)



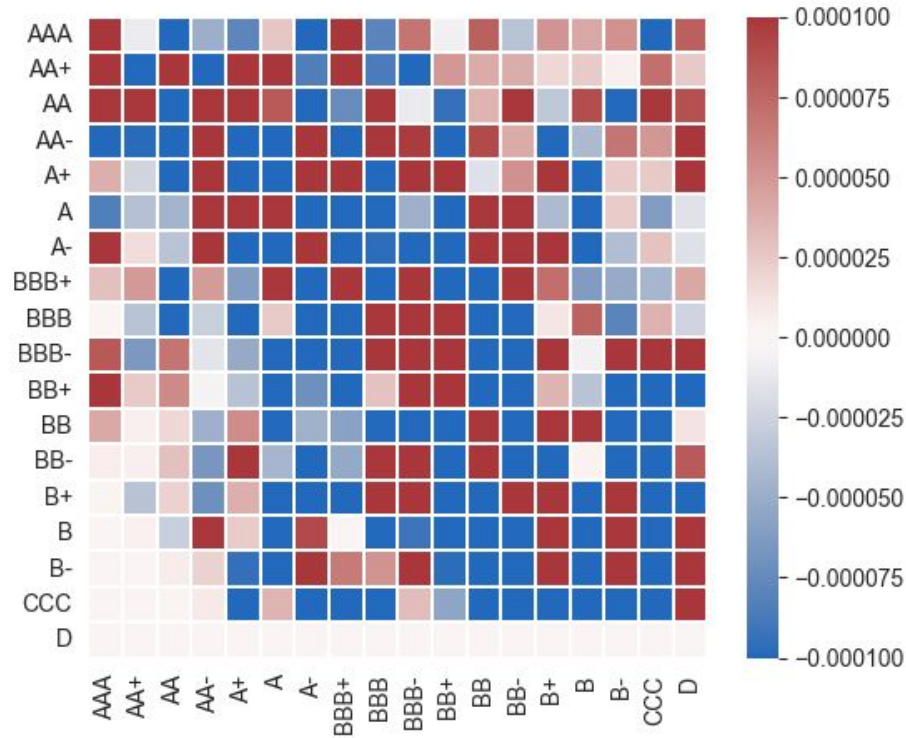
4.4.2.1 Tuned Prior (from EM) Validation [S&P 500]

Heat Map of Weekly Rating Transition Matrix with MCMC - Tuned Prior (from EM) (S&P adj)



4.4.2.1 Tuned Prior (from EM) Validation [S&P 500]

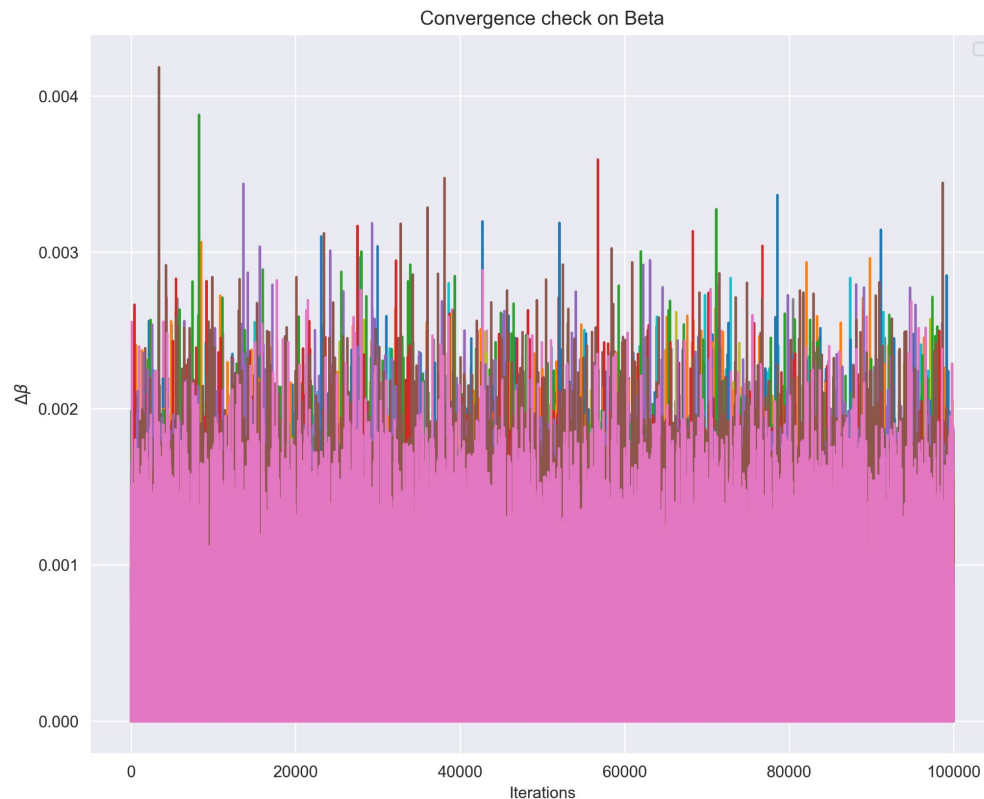
Error of Retrieved RTM vs original RTM annual (Prior from EM)



4.4.3 External Generator Approach

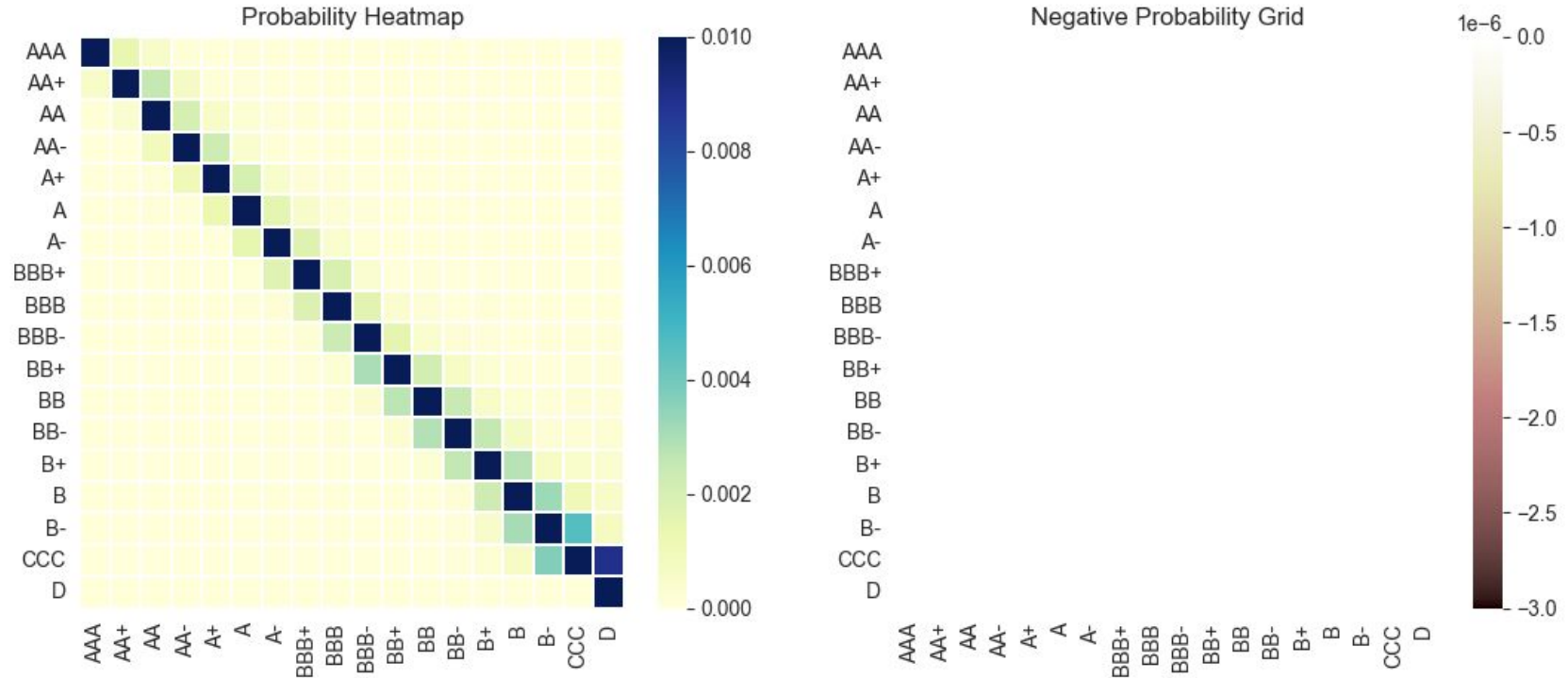
- Choose a set of *weak gamma priors*
- Generate **sample data** based on the original generator matrix from the RTM (i.e., Step 2~3 in the MCMC algorithm)
- Update the parameters from the posterior distribution
- Repeat the MCMC algorithm until **convergence** is achieved.
- Set the pre-convergence iteration as the *burn-in*
- Average the **generator matrix** sampled after the burn-in rate

4.4.3.1 Convergence Check on β (100,000 iterations)

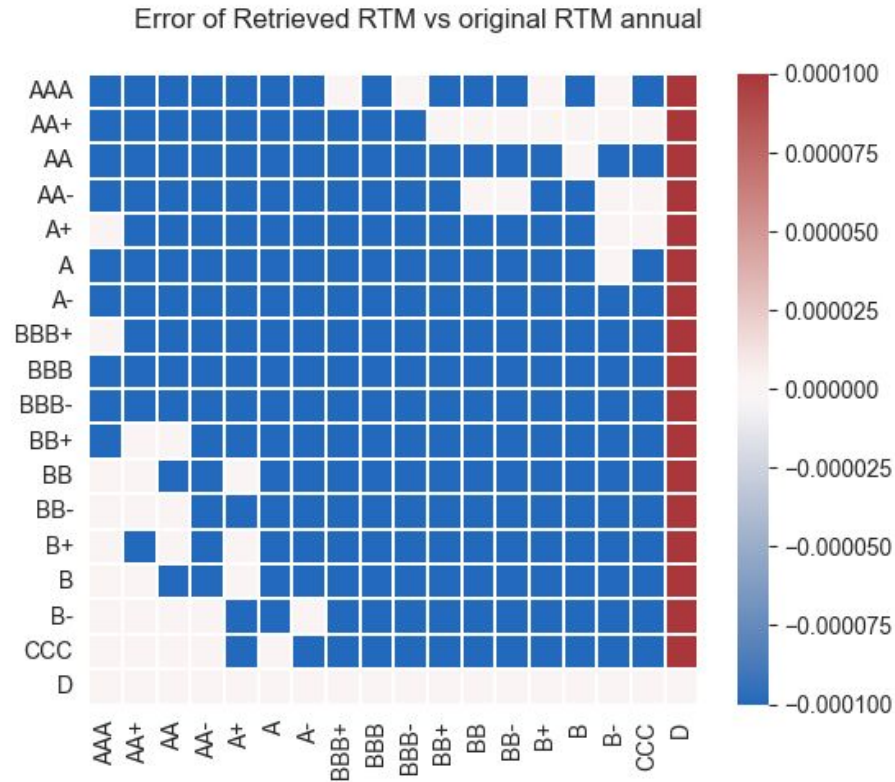


4.4.3.2 Validation (last 20,000 iterations)

Heat Map of Weekly Rating Transition Matrix with MCMC - Tuned Prior (from EM) (S&P adj)



4.4.3.2 Validation (last 20,000 iterations)



4.4.4 Evaluation

- Convergence has **not been reached**.
- Concerns:
 - Absence of convergence leads to **large errors** across all entries.
 - Weak priors may be **too weak**
 - **Computational limitations** on more iterations.
- Future steps...
 - Revision of the code on MCMC algorithm
 - More robust check for convergence
 - Separating simulation to various Markov Chains rather than continuously updating posterior parameters on one chain.