Moderntensor: Detailed Documentation on Core Mechanisms and Formulas (Updated Version Based on Design Decisions)

Moderntensor Development Team

April 3, 2025

1 Abstract

This document provides an overview and detailed explanation of the *updated* mathematical formulas and core mechanisms governing Moderntensor—a decentralized Artificial Intelligence (AI) platform built on the Cardano blockchain. The mechanisms discussed include incentive distribution, performance evaluation, penalty handling, weight calculation, resource allocation, and DAO governance. Each formula is clearly described, with its meaning, related parameters, operational mechanics, and interactions with other system components explained. A practical example (conceptual simulation) is also provided to illustrate the operational flow.

2 Introduction

Moderntensor is a decentralized AI platform that leverages the power of the Cardano blockchain to create a transparent, fair, and efficient environment for training, validating, and rewarding AI models. The platform aims to establish a network where Miners (who provide computational resources and train models) and Validators (who verify the Miners' work) interact and are incentivized in a rational manner. This document focuses on analyzing the *updated* formulas that govern the system's economic, evaluation, and governance mechanisms, aiming to enhance stability, objectivity, and resilience against undesirable behaviors. Understanding these mechanisms is crucial for ensuring the sustainable and equitable development of the Moderntensor network.

3 Core Mechanisms (Updated)

The following mechanisms work in coordination to regulate participant behavior and ensure the efficient operation of the network.

3.1 Incentive Distribution

This mechanism determines how rewards (e.g., the project's native tokens) are distributed to Miners and Validators based on their contributions and reputation.

3.1.1 Incentive for Miners (`Incentive_miner`)

Formula:

$$Incentive_{miner}(x) = f_{sig}(trust_{score}(x)) \times \frac{\sum_{j=0}^{m}(W_x \times P_{xj})}{\sum_{i=0}^{n}\sum_{j=0}^{m}(W_i \times P_{ij})}$$

Where,

$$f_{sig}(y) = \frac{L}{1 + e^{-k(y-x_0)}}$$

is the sigmoid function.

Meaning: The reward received by Miner x is proportional to their trust level (adjusted via the sigmoid function f_{sig}), the total performance evaluated by Validators (P_{xj}) weighted by the Miner's weight (W_x), and normalized based on the total weighted contribution of all Miners in the system.

Parameters:

- `trust_score(x)`: Historical trust score of Miner x (from 0 to 1), reflecting reliability based on past activity.
- `f_sig(y)`: Sigmoid function mapping the trust score.
  - L: Maximum value of the function (e.g., 1.0). **(To be determined)**
  - k: Slope of the sigmoid function (e.g., 5-15). **(To be determined)**
  - x_0: Inflection point of the function (e.g., 0.5 or 0.6). **(To be determined)**
- W_x: Weight of Miner x, reflecting their importance or potential contribution (see Formula 3.3.1).
- P_{xj}: Performance score of Miner x as evaluated by Validator j (see Formula 3.2.5, where $P_{miner\_adjusted}$ is the aggregated result).
- Denominator: Total weighted contribution ($W \times P$) of all Miners, ensuring a fixed reward pool (if applicable).

Operation:

1. The trust score `trust_score(x)` is fed into the sigmoid function f_{sig}. This function stabilizes rewards by reducing sensitivity at very high or low trust scores, compared to linear multiplication.
2. The Miner x's performance is aggregated from Validator evaluations (P_{xj}) and multiplied by their weight W_x.
3. The result is normalized by dividing by the total weighted contribution of the system.
4. Finally, it is multiplied by the trust score factor (post-sigmoid) to determine the final reward.
5. **Interaction:** This reward provides direct economic motivation for Miners to maintain high performance (P_{xj}) and a good trust score (`trust_score`).

3.1.2 Incentive for Validators (`Incentive_validator`)

Formula:

$$Incentive_{validator}(v) = f_{sig}(trust_{score}(v)) \times \frac{W_v \times E_v}{\sum_{u \in V}(W_u \times E_u)}$$

Where $f_{sig}(y)$ is the same sigmoid function as above.

Meaning: The reward received by Validator v is proportional to their trust level (via the sigmoid function f_{sig}), their weight W_v, and their evaluation performance E_v, normalized based on the total weighted contribution of all Validators.

Parameters:

- `trust_score(v)`: Historical trust score of Validator v.
- `f_sig(y)`: Sigmoid function with parameters L, k, x_0. **(To be determined, may be the same or different from the Miner's function)**
- W_v: Weight of Validator v (see Formula 3.3.2).

- E_v: Performance score of Validator v (see Formula 3.2.3).
- Denominator: Total weighted contribution ($W \times E$) of all Validators.

Operation:

1. Similar to Miners, the Validator's trust score is processed through the sigmoid function f_{sig}.
2. The Validator's performance E_v is multiplied by their weight W_v.
3. The result is normalized and multiplied by the trust score factor (post-sigmoid).
4. **Interaction:** This incentivizes Validators to maintain good evaluation performance (E_v), increase their stake or long-term participation (affecting W_v), and uphold high credibility (`trust_score`).

3.2 Performance Evaluation

These formulas measure the quality of work performed by Miners and Validators.

3.2.1 Task Completion Rate (`Q_task`)

Formula:

$$Q_{task} = \frac{\sum_t (task_{success,t} \times e^{-\delta(T-t)})}{\sum_t (task_{total,t} \times e^{-\delta(T-t)})}$$

Meaning: Measures the success rate of completed tasks, factoring in time (recent tasks are weighted more heavily).

Parameters:

- `task_success,t`, `task_total,t`: Number of successful/total tasks at time t.
- T: Current time.
- δ: Time decay constant. **(Should be governed by the DAO)**.

Operation: Calculates a weighted average of the success rate over time, with the weight $e^{-\delta(T-t)}$ decreasing for older time points.

3.2.2 Miner Approval Rate (`D_miner`)

Formula:

$$D_{miner}(i) = \frac{\sum(\text{Number of times Miner } i \text{ is approved})}{\text{Total number of times Miner } i \text{ is reviewed}}$$

Meaning: The percentage of Miner i's work approved by Validators.

Operation: A simple division of the number of approvals by the total number of reviews.

3.2.3 Validator Performance Score (`E_validator`)

Formula:

$$E_{validator} = \theta_1 Q_{task\_validator} + \theta_2 Metric\_Validator\_Quality + \theta_3 \times Penalty\_Term(Deviation)$$

Meaning: A composite evaluation of a Validator's quality, combining task completion ability (if applicable), objective evaluation quality, and consensus with the network.

Parameters:

- $\theta_1, \theta_2, \theta_3$: Weight coefficients ($\sum \theta_i = 1$). **(Should be governed by the DAO)**.
- `Q_task_validator`: Task completion rate of the Validator (if they also perform tasks).

- `Metric_Validator_Quality`: An alternative quality metric replacing `accuracy`. Example: "Historical consistency":

$$1 - \text{StdDev}(\text{Evaluation scores of V over the last N rounds})$$

. **(Requires precise formula definition and parameter N)**.

- $Deviation = |Eval - Avg|/\sigma$: Normalized deviation of the current evaluation ($Eval$) from the average (Avg).

- `Penalty_Term(Deviation)`: A decreasing penalty function when deviation exceeds a threshold:

$$Penalty\_Term(dev) = \frac{1}{1 + k' \times \max(0, dev - Threshold_{dev})^p}$$

.

  - `Threshold_dev`: Deviation threshold for triggering a penalty. **(To be determined; may be governed by the DAO)**.
  - k': Coefficient controlling penalty severity. **(To be determined)**.
  - p: Exponent of deviation (e.g., 1 or 2). **(To be determined)**.

Operation:

1. Calculate components: Q_{task} (if applicable), `Metric_Validator_Quality` (per defined formula), and `Penalty_Term` (based on deviation from the average evaluation).
2. Combine these components by multiplying with their respective weights θ_i and summing them.
3. **Interaction:** The $E_{validator}$ score directly impacts the Validator's reward (Section 3.1.2) and weight (Section 3.3.2). It encourages Validators to perform tasks well (if applicable), evaluate consistently/accurately, and align with network consensus (unless justified).

3.2.4 Basic Miner Performance (`P_miner`)

Formula: (Same as Q_{task})

$$P_{miner} = \frac{\sum_t (task_{success,t} \times e^{-\delta(T-t)})}{\sum_t (task_{total,t} \times e^{-\delta(T-t)})}$$

- δ: Decay constant. **(Should be governed by the DAO)**.

Meaning: Measures the Miner's task completion performance, prioritizing recent tasks.

3.2.5 Adjusted Miner Performance (`P_miner_adjusted`)

Formula:

$$P_{miner\_adjusted} = \frac{\sum_v (trust_{score_v} \times P_{miner,v})}{\sum_v trust_{score_v}}$$

Meaning: Aggregates the performance scores $P_{miner,v}$ received by a Miner from Validators v, weighted by the Validators' trust scores $trust_{score_v}$.

Operation: Computes a weighted average of the evaluation scores $P_{miner,v}$, where the weights are the Validators' $trust_{score_v}$. Evaluations from more reputable Validators have greater influence on the final performance score. **Interaction:** $P_{miner\_adjusted}$ is a key input for updating the Miner's Trust Score (Section 3.4.1) and calculating rewards (Section 3.1.1, via P_{xj}).

3.3 Participant Weight Calculation

Weights determine the relative "importance"of each Miner and Validator in various calculations.

3.3.1 Miner Weight (`W_x`)

Formula:

$$W_x = \sum_t P_{miner,t} \times e^{-\delta_W (T-t)}$$

Meaning: Aggregates the historical performance $P_{miner,t}$ of a Miner, with weights decreasing over time. Reflects the Miner's capability and consistent contribution.

Parameters:

- `P_miner,t`: Miner's performance at time t (could be P_{miner} or $P_{miner\_adjusted}$).
- T: Current time.
- `delta_W`: Weight decay constant. **(Should be governed by the DAO)**.

Operation: Computes a weighted sum of performance over time, similar to Q_{task} but not as a ratio. **Interaction:** W_x is directly used in the Miner reward formula (Section 3.1.1).

3.3.2 Validator Weight (`W_validator`)

Formula:

$$W_{validator} = \lambda \times \left(\frac{\log(1 + stake_v)}{\sum \log(1 + stake_u)} \right) + (1 - \lambda) \times \left(\frac{E_{validator}}{\max(E_{avg}, \epsilon)} \right) \times (1 + \log_{10}(time\_participated))$$

Meaning: Determines a Validator's importance based on a balanced combination of financial contribution (stake, log-adjusted to reduce the advantage of large stakes), relative performance ($E_{validator}$ compared to the average E_{avg}), and participation duration.

Parameters:

- `stake_v`: Amount staked by Validator v.
- $\log(1 + s)$: Natural logarithm (ln) or base-10 logarithm to adjust stake influence.
- $E_{validator}$: Performance score of Validator v.
- E_{avg}: Average performance of Validators. ϵ is a small constant. **(Requires definition of E_{avg} calculation)**.
- `time_participated`: Duration of participation (unit to be defined).
- $\log_{10}()$: Base-10 logarithm.
- λ: Balance coefficient between stake and performance (0 to 1). **(Critical, should be governed by the DAO)**.

Operation:

1. Calculate the stake influence component (log-adjusted and normalized).
2. Calculate the performance influence component (normalized by average) and participation duration (via log10).
3. Combine the two components based on the λ coefficient. High λ prioritizes stake; low λ prioritizes performance/time.
4. **Interaction:** $W_{validator}$ directly affects the Validator's reward (Section 3.1.2).

3.4 Trust Score and Fairness Mechanisms

Managing reputation and ensuring fair participation opportunities.

3.4.1 Trust Score Update (`TrustScore_Update`)

Formula:

$$TrustScore_{new} = TrustScore_{old} \times e^{-\delta_{trust} \times time} + \alpha_{effective}(TrustScore_{old}) \times f_{update\_sig}(Score_{new})$$

Where:

- $$\alpha_{effective}(y) = \alpha_{base} \times (1 - k_\alpha \times |y - 0.5|)$$
- $$f_{update\_sig}(s) = \frac{L_{upd}}{1 + e^{-k_{upd}(s - s_0)}}$$

Meaning: Updates the trust score based on the old value (decayed over time `time` without evaluation) and an adjustment based on the latest performance score `Score_new` (processed through the sigmoid function f_{update_sig}), with a variable learning rate $\alpha_{effective}$ (faster in the middle, slower at extremes).

Parameters:

- `TrustScore_old/new`: Old/new trust score.
- `time = time_since_last_evaluation`: Time since the last evaluation.
- `delta_trust`: Trust score decay constant. **(Should be governed by the DAO)**.
- `alpha_base`: Base learning rate. **(To be determined)**.
- `k_alpha`: Learning rate adjustment coefficient. **(To be determined)**.
- `Score_new`: New performance score (e.g., $P_{miner\_adjusted}$ for Miners, $E_{validator}$ for Validators). **(Requires clear source definition)**. If not evaluated, `Score_new`=0.
- `f_update_sig(s)`: Sigmoid function mapping the new score. L_{upd}, k_{upd}, s_0 **(To be determined)**.

Operation:

1. Calculate the remaining trust score after time-based decay.
2. Compute the effective learning rate $\alpha_{effective}$ based on the old trust score.
3. Map the new performance score `Score_new` through the sigmoid function f_{update_sig}.
4. Multiply the result of step 3 by the effective learning rate (step 2).
5. Add the results of steps 1 and 4 to obtain the new trust score.
6. **Interaction:** The new trust score affects rewards (Section 3.1) and selection probability (Section 3.4.2) in subsequent cycles.

3.4.2 Miner Selection Probability (`SelectionProbability`)

Formula:

$$SelectionProbability = trust\_score \times (1 + \beta \times \min(time\_since\_last\_selection, MaxTimeBonusEffect))$$

Meaning: The probability of a Validator selecting a Miner, based on the Miner's `trust_score`, with an additional bonus if the Miner has not been selected recently, capped at a maximum.

Parameters:

- `trust_score`: Current trust score of the Miner.

- `time_since_last_selection`: Number of cycles/time since the Miner was last selected.
- β: Fairness bonus coefficient. **(Should be governed by the DAO)**.
- `MaxTimeBonusEffect`: Maximum cycles/time for the bonus to apply. **(To be determined)**.

Operation: Calculates a bonus value based on waiting time (capped), adds 1, and multiplies by the trust score. This result (or a normalized version) is used by Validators to decide which Miner to select. **Interaction:** This mechanism ensures Miners with temporarily low trust(scores or new participants still have a chance to be selected and improve their reputation, preventing "starvation."

3.5 Penalty Mechanism

Measures to address fraudulent behavior or severe underperformance.

3.5.1 Performance Adjustment (Recovery)

Formula:

$$P_{adjuster\_new} = E_{validator\_new} + \gamma(E_{validator\_base} - E_{validator\_new})$$

- γ: Recovery rate.

Meaning: Allows a Validator's performance score to gradually recover toward a baseline level after a reduction due to non-fraudulent factors.

3.5.2 Stake Slashing (`Stake Slashing`)

Formula:

$$Slash_{amount} = \min(MaxSlashRate \times stake, fraud\_severity \times stake)$$

Meaning: Imposes a direct financial penalty by reducing a portion of the stake upon detection of fraud.

Parameters:

- `stake`: Current stake amount.
- `MaxSlashRate`: Maximum slashing rate. **(Should be governed by the DAO)**.
- `fraud_severity`: Severity level, determined by tiers:
  - **Tier 1 (Minor):** E.g., Deviation $> Threshold_{dev1}$ for N_1 cycles $\rightarrow fraud\_severity = S_1$ (e.g., 0.1).
  - **Tier 2 (Moderate):** E.g., Deviation $> Threshold_{dev2}$ or invalid data submission $\rightarrow fraud\_severity = S_2$ (e.g., 0.3).
  - **Tier 3 (Severe):** E.g., Evidence of coordinated attack $\rightarrow fraud\_severity = S_3$ (e.g., 0.7-1.0).
  - ***(Requires clear definition of Threshold_ dev1/2, N1, and S1/2/3 values. Fraud verification process needs design.)***

Operation: Calculates the stake reduction based on severity, capped at the maximum allowed rate. **Interaction:** A strong economic penalty to deter fraudulent behavior.

3.5.3 Fraud Detection (`Fraud Detection`)

Logic:

- Set `Fraud_Flag = 1` if fraudulent behavior is detected (per severity tiers).
- Basic mechanism: If $Deviation > Threshold_{dev}$ for N_{cycles} cycles → Trigger review or flag Tier 1.
- (***Thresholds*** `Threshold_dev, N_cycles` ***need definition and may be governed by the DAO***).

Meaning: A mechanism to identify suspicious behaviors requiring action.
Operation: Monitors metrics (e.g., evaluation deviation) over cycles. If thresholds are exceeded for a sufficient duration, triggers a review or automatic flagging.

3.5.4 Trust Score Update Due to Fraud

Formula:

$$trust_{score\_new} = trust_{score\_old} \times (1 - \eta \times Fraud\_Flag)$$

Meaning: Significantly reduces the trust score upon confirmed fraud.
Parameters:

- η: Trust score penalty coefficient (e.g., 0.1 - 1.0). **(Should be governed by the DAO)**.
- `Fraud_Flag`: Fraud indicator (0 or 1).

Operation: If `Fraud_Flag=1`, the trust score is reduced by a proportion η. **Interaction:** This reduction immediately lowers the participant's rewards and selection probability.

3.6 Resource Allocation (`Resource Allocation`)

3.6.1 Subnet Resource Distribution (`R_subnet`)

Formula:

$$R_{subnet} = \frac{\sum_{i \in subnet}(W_i \times P_i)}{\sum_{j \in \text{all subnets}}(W_j \times P_j)} \times R_{total}$$

Meaning: Allocates the system's total resources R_{total} to Subnets proportional to the total weighted contribution (weight W times performance P) of their members.
Operation: Calculates the Subnet's contribution ratio relative to the entire system and multiplies by the total resources.

3.7 DAO Governance (`DAO Governance`)

3.7.1 DAO Voting Power (`Voting Power`)

Formula:

$$VotingPower(p) = stake_p \times \left(1 + k_g \sqrt{\frac{time\_staked_p}{total\_time}}\right) \times Lockup\_Multiplier(p)$$

Meaning: Determines influence in governance voting, based on stake amount, staking duration (with diminishing returns via square root), and an additional multiplier for locking stake.
Parameters:

- `stake_p`: Stake amount.
- `time_staked_p`, `total_time`: Staking duration and reference time (**requires clear definition**).

- k_g: Time bonus adjustment coefficient. **(To be determined)**.
- $\sqrt{}$: Square root function.
- `Lockup_Multiplier(p)`: Multiplier for locking stake. E.g., No lock = 1.0, 3-month lock = 1.2, 6-month lock = 1.5, 1-year lock = 2.0. ***(Requires design of lock mechanism and multiplier tiers.)***

Operation: Calculates bonus components from duration and stake locking, then multiplies by the base stake to determine final voting power. **Interaction:** This voting power is used when the community votes on protocol changes, parameter updates (those marked "governed by DAO"), or other significant decisions.

4 System Operation Flow

The Moderntensor system operates in a repeating cycle with the following steps:

1. **Task Assignment:** Validators use `SelectionProbability` (Section 3.4.2) to select Miners for tasks. Key factors are trust score and waiting time.
2. **Task Execution:** Selected Miners receive tasks, use their resources to process them (e.g., training a model segment), and submit results.
3. **Evaluation:** Validators receive Miner results, review them, and assign performance scores $P_{miner,v}$.
4. **Consensus & Adjusted Performance:** The system aggregates evaluations from multiple Validators for the same Miner, calculating the adjusted performance score $P_{miner\_adjusted}$ (Section 3.2.5) based on Validator credibility. Simultaneously, Validator performance ($E_{validator}$, Section 3.2.3) is computed based on evaluation quality and consensus.
5. **Trust Score Update:** Based on evaluation results ($P_{miner\_adjusted}$ or $E_{validator}$ as `Score_new`), trust scores for Miners and Validators are updated per the formula in Section 3.4.1. If not evaluated, trust scores decay over time.
6. **Reward Distribution:** Using the latest trust scores, weights (W_x, W_v), and performance ($P_{miner\_adjusted}, E_v$ the system calculates and prepares rewards for Miners and Validators per Section 3.1 formulas.
7. **Penalty Handling:** If fraud is detected (Section 3.5.3), the system triggers penalties such as Trust Score reduction (Section 3.5.4) and Stake slashing (Section 3.5.2).
8. **Blockchain Update:** Critical state information (new Trust Scores, weights, reward details, stake status, fraud flags, etc.) is securely and transparently recorded on the Cardano blockchain, typically by updating the Datum of UTXOs at smart contract addresses.
9. **DAO Governance:** Periodically or upon proposals, token holders (possibly Miners, Validators, or other users) use their `Voting Power` (Section 3.7.1) to vote on protocol changes, system parameter updates (marked "governed by DAO"), or other key decisions.

This cycle repeats, creating a continuous feedback loop that enables the network to self-regulate, incentivize good behavior, and penalize bad behavior.

5 Practical Example (Conceptual Simulation - Cycle 1)

This section illustrates how the formulas operate in a single cycle, using example parameters to clarify the computation flow.

Initial Assumptions:

- **State (Cycle 0):** Miners M1-M5, Validators V1-V3 with initial Trust Scores (M1=0.9, M2=0.8, M3=0.7, M4=0.6, M5=0.5; V1=0.9, V2=0.8, V3=0.7). `time_since_last_selection = 0` for all Miners. Example weights W_x ($W_{M1} = 2.0, W_{M2} = 1.8, W_{M3} = 1.5, W_{M4} = 1.2, W_{M5} = 1.0$). Example system total value = 50.0.
- **Example Parameters (FOR ILLUSTRATION ONLY):** $\beta = 0.2$, `MaxTimeBonusEffect=10`, $\delta_{trust} = 0.1$, $\alpha_{base} = 0.1$, $k_\alpha = 1.0$, `f_update_sig` ($L_{upd} = 1, k_{upd} = 5, s_0 = 0.5$), `f_sig` ($L = 1, k = 10, x_0 = 0.5$).

Step 1: Task Assignment Validators select Miners based on `SelectionProbability` (Formula 3.4.2):

$$SelectionProbability = trust\_score \times (1 + \beta \times \min(time\_since\_last\_selection, MaxTimeBonusEffect))$$

In this cycle, $time\_since\_last\_selection = 0$ for all Miners.

- M1: $0.9 \times (1 + 0.2 \times 0) = 0.9$
- M2: $0.8 \times (1 + 0.2 \times 0) = 0.8$
- M3: $0.7 \times (1 + 0.2 \times 0) = 0.7$
- M4: $0.6 \times (1 + 0.2 \times 0) = 0.6$
- M5: $0.5 \times (1 + 0.2 \times 0) = 0.5$

Assumed selection (consistent with original example): V1 selects M1, M2; V2 selects M1, M3; V3 selects M2, M4.
Miner M5 is not selected, so its `time_since_last_selection` increases to 1 for the next cycle.

Step 2: Task Execution & Evaluation Miners execute tasks, and Validators evaluate them.
Assumed evaluation scores $P_{miner,v}$:

- V1: M1=0.85, M2=0.9
- V2: M1=0.9, M3=0.75
- V3: M2=0.8, M4=0.65

Step 3: Adjusted Performance Calculate $P_{miner\_adjusted}$ for each evaluated Miner based on Validator trust scores (Formula 3.2.5):

$$P_{miner\_adjusted} = \frac{\sum_v (trust_{score_v} \times P_{miner,v})}{\sum_v trust_{score_v}}$$

- $P_{adj}(M1) = \frac{(0.9 \times 0.85) + (0.8 \times 0.9)}{0.9 + 0.8} \approx 0.873$
- $P_{adj}(M2) = \frac{(0.9 \times 0.9) + (0.7 \times 0.8)}{0.9 + 0.7} \approx 0.856$
- $P_{adj}(M3) = \frac{(0.8 \times 0.75)}{0.8} = 0.75$
- $P_{adj}(M4) = \frac{(0.7 \times 0.65)}{0.7} = 0.65$
- $P_{adj}(M5) = 0$ (Not evaluated)

Step 4: Trust Score Update Use Formula 3.4.1 with $time = 1$ and example parameters.

$$TrustScore_{new} = TrustScore_{old} \times e^{-\delta_{trust} \times time} + \alpha_{effective}(TrustScore_{old}) \times f_{update\_sig}(Score_{new})$$

$$\alpha_{effective}(y) = 0.1 \times (1 - 1.0 \times |y - 0.5|)$$

$$f_{update\_sig}(s) = \frac{1}{1 + e^{-5(s-0.5)}}$$

- **M1 (`Score_new` = 0.873):**

$$\begin{aligned} \alpha_{eff}(0.9) &= 0.06 \\ f_{upd}(0.873) &\approx 0.866 \\ Trust(M1)_{new} &= 0.9 \times e^{-0.1} + 0.06 \times 0.866 \approx \mathbf{0.8663} \end{aligned}$$

- **M2 (`Score_new` = 0.856):**

$$\begin{aligned} \alpha_{eff}(0.8) &= 0.07 \\ f_{upd}(0.856) &\approx 0.855 \\ Trust(M2)_{new} &= 0.8 \times e^{-0.1} + 0.07 \times 0.855 \approx \mathbf{0.7837} \end{aligned}$$

- **M3 (`Score_new` = 0.75):**

$$\begin{aligned} \alpha_{eff}(0.7) &= 0.08 \\ f_{upd}(0.75) &\approx 0.777 \\ Trust(M3)_{new} &= 0.7 \times e^{-0.1} + 0.08 \times 0.777 \approx \mathbf{0.6956} \end{aligned}$$

- **M4 (`Score_new` = 0.65):**

$$\begin{aligned} \alpha_{eff}(0.6) &= 0.09 \\ f_{upd}(0.65) &\approx 0.679 \\ Trust(M4)_{new} &= 0.6 \times e^{-0.1} + 0.09 \times 0.679 \approx \mathbf{0.6040} \end{aligned}$$

- **M5 (`Score_new` = 0):**

$$\begin{aligned} Trust(M5)_{new} &= 0.5 \times e^{-0.1} + \alpha_{eff}(0.5) \times f_{upd}(0) \\ &= 0.5 \times e^{-0.1} + 0.1 \times 0 \approx \mathbf{0.4524} \end{aligned}$$

(Similar calculations needed for Validators based on their E_v)

Step 5: Reward Distribution Use Formula 3.1.1 with $TrustScore_{new}$ and example parameters.

$$Incentive_{miner}(x) = f_{sig}(TrustScore_{new}) \times \frac{W_x \times P_{adj}(x)}{\text{Total System Value}}$$

$$f_{sig}(y) = \frac{1}{1 + e^{-10(y-0.5)}}$$

Example system total value = 50.

- M1 ($Trust \approx 0.8663, P_{adj} \approx 0.873, W = 2.0$):

$$f_{sig}(0.8663) \approx 0.975$$

$$Incentive(M1) \approx 0.975 \times (2.0 \times 0.873/50) \approx \mathbf{0.0340}$$

- M2 ($Trust \approx 0.7837, P_{adj} \approx 0.856, W = 1.8$):

$$f_{sig}(0.7837) \approx 0.945$$

$$Incentive(M2) \approx 0.945 \times (1.8 \times 0.856/50) \approx \mathbf{0.0291}$$

- M3 ($Trust \approx 0.6956, P_{adj} = 0.75, W = 1.5$):

$$f_{sig}(0.6956) \approx 0.876$$

$$Incentive(M3) \approx 0.876 \times (1.5 \times 0.75/50) \approx \mathbf{0.0197}$$

- M4 ($Trust \approx 0.6040, P_{adj} = 0.65, W = 1.2$):

$$f_{sig}(0.6040) \approx 0.740$$

$$Incentive(M4) \approx 0.740 \times (1.2 \times 0.65/50) \approx \mathbf{0.0115}$$

- M5 ($Trust \approx 0.4524, P_{adj} = 0, W = 1.0$):

$$Incentive(M5) = 0$$

(Similar calculations needed for Validators based on E_v and W_v)

Step 6-9: Penalty Handling, Blockchain Update, DAO

- Check if any Validator's *Deviation* exceeds `Threshold_dev` over `N_cycles`. If so, set `Fraud_Flag` and apply penalties (trust reduction, slashing).
- Record new $TrustScore_{new}$ values, calculated rewards, and other states (e.g., `time_since_last_select`) on the blockchain via Datum updates.
- If DAO voting occurs, calculate `VotingPower` for participants based on stake, staking duration, and lock status (Formula 3.7.1).

(Important Note on Example: Numerical results are illustrative only due to assumed parameters. Actual results will depend on final parameter values selected after analysis and simulation.)

6 Further Discussion and Development Directions

Implementing and operating these mechanisms requires continuous monitoring and adjustment. Key focus areas include:

- **Parameter Definition and Refinement:** This is the most critical task. A clear process (including analysis, simulation, and possibly A/B testing on a testnet) is needed to select initial parameters and a mechanism for the DAO to update them later.
- **Advanced Simulation:** Develop more complex simulation models with multiple agents (Miners, Validators) employing different strategies (honest, lazy, attacking, etc.) to assess system stability and resilience.
- **User Interface Design (UI/UX):** Provide intuitive tools and interfaces for participants to track performance, reputation, rewards, and engage in governance easily.
- **Enhanced Fraud Detection Mechanisms:** Beyond relying on evaluation deviation, explore more sophisticated fraud detection methods based on behavioral analysis, machine learning, or cryptographic proofs.
- **Cold Start Problem:** Design specific mechanisms to help new participants integrate into the network (e.g., initial trust score, trial period, initial selection opportunities).
- **Detailed Parameter Documentation:** Create a separate document or glossary clearly defining all system parameters, their meanings, default values (if any), proposed ranges, and which are governed by the DAO.

By continuously improving and adjusting based on real-world data and community feedback, Moderntensor can build a robust, fair, and sustainable decentralized AI platform.