Can you mathematically model group polarization?

> **Quote:**
>
> Someone should demonstrate this more mathematically, but it seems to me that if you start with a random assortment of identities, small fluctuations plus reactions should force polarization. That is, if a chance fluctuation makes environmentalists slightly more likely to support gun control, and this new bloc goes around insulting polluters and gun owners, then the gun owners affected will reactively start hating the environmentalists and insult them, the environmentalists will notice they're being attacked by gun owners and polarize even more against them, and so on until (environmentalists + gun haters) and (polluters + gun lovers) have become two relatively consistent groups. Then if one guy from the (environmentalist + gun hater) group happens to insult a Catholic, the same process starts again until it's (environmentalists + gun haters + atheists) and (polluters + gun lovers + Catholics), and so on until there are just two big groups.
>
> *— Scott Alexander of AstralCodexTen, in his article Why I Am Not A Conflict Theorist*

This seems like a super interesting problem that one could model with relatively low effort - about one day's worth of focused effort at most. That is, if someone had the mathematical and computational acumen required to run a simulation like this.

Alternatively, you could prompt an LLM to generate the code for you (which i did try), but that didn't really work out all that well. The first draft of the code was decent, but it generated a static final histogram that basically went from a random distribution of opinions among agents to a supremely polarized distribution. Which is like, cool, but all the computing happened in the backend, so does anyone really care? I wanted to see more step-wise evolution so someone could look at it and go "wow! mimetic desire exists!" (I don't know if I'm using this in the correct context but it sounds cool).

I also think that the basic rules of the LLM-generated model are flawed (copy-pasted from my conversation with claude):

1. Initialize random opinions for agents across multiple issues
2. For each time step and each agent:
    1. Identify similar and opposing agents

    2. Move toward similar agents' opinions (social influence)
    3. Move away from opposing agents' opinions (reactive polarization)
    4. Align opinions across issues (issue correlation)
3. Track the history of all opinions for analysis

This looks like you're basically forcing these agents to be polarized, which is not the goal of the simulation. The goal of the simulation is to *only* start with a fundamental rulset of "small fluctuations plus reactions" and then see if this results in group polarization towards the end.

This is my guess of what an *extremely basic* algorithm for this would look like:

1. Create agents
2. Create a list of arbitrary issues, numbered $issue\ 1, issue\ 2, \ldots issue\ n$
3. Let agents have "belief" for or against issues, on a scale of -1 for "against" to 1 for "for", with 0 being neutral
4. Make agents have "affinity" towards each other, which starts off at 0
5. Make agents with similar beliefs on the same issues increase affinity for each other with each chronological step, and agents with dissimilar beliefs decrease affinity with each other with chronological step
6. With each step have a non-zero random chance that an agents beliefs in a random issue is incremented or decremented by 0.1
7. Observe if a number of steps into this simulation, we have polarized groups - or, in mathematical terms, if there's a strong correlation between beliefs in one issue and another. Do most agents that are *for* issue 1 *against* issue 2? Or something along those lines. If that holds true, then we can say that small fluctuations plus reactions do force polarization. This doesn't say anything about human behaviour, because we haven't yet linked this model's fundamental axioms with those of human behaviour, but it's an interesting observation nonetheless.

Super interesting.

Update: I managed to make a working simulation of this! I prompted Claude with the updated algorithm, and it did a surprisingly good job of coding the simulation in one go. I tried with ChatGPT and deepseek before that, but despite underperforming on certain benchmarks and tests, Claude remains the best at coding, at least in my experience.

The initial simulation didn't really reflect my beliefs - it displayed an upward trend initially, but it ended up dipping back down to polarization levels below starting, which was super confusing to me. When I asked claude *why* this could be happening, one of its bullet points prompted me to realize that I was updating the beliefs randomly at *every step*, which is stupid because beliefs have persistence.

This is the word by word modification prompt that I used: "You're not supposed to repeatedly change random beliefs - beliefs should have some persistence. I think something that would work better is an initial randomisation of beliefs and then we let the model run its course and see if that ends up in belief clusters, rather than continuously randomising beliefs because that is just not how it works in the real world. Belief systems have some *persistence*, for a lack of a better word."

And Claude made some changes, the gist of which can be summed up with "In the new version, beliefs are initialized randomly at the start and then only change through social influence." You can see this conversation with Claude here.

I've also deployed this online here.

Once you've played around with that a bit, if you want some help interpreting the output (apart from the line graph - that's pretty self explanatory), here's a technical overview generated by claude with a few modifications (primarily added context, better formatting, and outgoing explanatory links) added by me.

Technical Overview: Agent Belief Polarization Simulation

Model Architecture

The simulation models interactions between agents with beliefs on multiple issues and evolving affinities for each other. Inspired by the following quote by Scott Alexander in his post Why I Am Not A Conflict Theorist

"Someone should demonstrate this more mathematically, but it seems to me that if you start with a random assortment of identities, small fluctuations plus reactions should force polarization. That is, if a chance fluctuation makes environmentalists slightly more likely to support gun control, and this new bloc goes around insulting polluters and gun owners, then the gun owners affected will reactively start hating the environmentalists and insult them, the environmentalists will notice they're being attacked by gun owners and polarize even more against them, and so on until (environmentalists + gun haters) and (polluters + gun lovers) have become two relatively consistent groups. Then if one guy from the (environmentalist + gun hater) group happens to insult a Catholic, the same process starts again until it's (environmentalists + gun haters + atheists) and (polluters + gun lovers + Catholics), and so on until there are just two big groups."

Core Components

1. **Agents**: Each agent has:
    - A belief vector with values [-1, 1] across n issues
    - Affinity values [-1, 1] toward all other agents
2. **Initialization**:
    - Beliefs: Randomly assigned (uniform distribution between -1 and 1)
    - Affinities: All initialized to 0
3. **Update Rules**:
    a) **Affinity Updates**:

    - For each pair of agents i and j:
        - Calculate belief similarity as the *normalized* dot product of belief vectors. The dot product of normalized vectors when calculated numerically is a stable way of detecting how equal two vectors are in relation to direction and sense.
        - Increment affinity(i,j) by similarity × affinity_change_rate
        - Ensure symmetry by setting affinity(j,i) = affinity(i,j)
        - Constrain all affinity values to the range [-1, 1]

    b) **Belief Updates**:
    - For each agent and each issue:
        - For each other agent:
            - If affinity is positive: adjust belief toward the other agent's belief proportional to affinity strength
            - If affinity is negative: adjust belief away from the other agent's belief proportional to affinity strength
        - Constrain all belief values to the range [-1, 1]
    - Apply all updates simultaneously to avoid order effects

Visualization Components

1. Network Graph

- **Nodes**: Agents
- **Node Colors**: Belief on first issue (red = positive, blue = negative)
- **Edges**: Significant affinities (|affinity| > 0.1)
- **Edge Colors**: Blue for positive affinity, red for negative
- **Edge Width**: Proportional to |affinity|

2. Belief Heatmap

- **X-axis**: Issues (1 to n)
- **Y-axis**: Agents
- **Colors**: Red = positive belief (+1), Blue = negative belief (-1)
- **Interpretation**: Each row shows one agent's beliefs across all issues

3. Correlation Matrix

- **Axes**: Issues
- **Colors**: Red = positive correlation, Blue = negative correlation
- **Interpretation**: Shows how beliefs on different issues have become associated
- **Example**: If cell (1,2) is bright red, agents who believe strongly in issue 1 also tend to believe strongly in issue 2

4. Polarization Metrics

- **Correlation of Beliefs**:
  - Calculate correlation matrix between all issue pairs
  - Take the mean of the absolute values of the upper triangular portion, as the correlation matrix is equal across the diagonal. For example, (1, 2) and (2, 1) will have the same correlation value.
  - Higher values indicate stronger correlations between different issues
- **Belief Distance**:
  - For each pair of agents, calculate the Euclidean distance between their belief vectors
  - Compute the average distance across all agent pairs
  - Higher values indicate greater overall separation in belief space

Parameter Specifications

- `num_agents`: Number of agents (default: 40)
- `num_issues`: Number of belief dimensions (default: 5)
- `affinity_change_rate`: Scaling factor for affinity updates (default: 0.05)
- `positive_influence_rate`: Strength of positive influence (default: 0.03)
- `negative_influence_rate`: Strength of negative influence (default: -0.03)

Implementation Notes

1. The simulation runs synchronous updates to avoid order effects
2. No randomization after initialization ensures belief persistence
3. Both positive and negative influence mechanisms are included

4. Belief and affinity values are constrained to the range [-1, 1]
5. The network visualization uses force-directed layout with fixed positions

Basically get yourself out of echo chambers, don't hate things just because people you hate love those things or vice versa, and be self-aware and cognizant of the way your beliefs and perceptions are shaped.

much love,
hari