

Abstract model rigidity



I'm thinking that maybe a more abstract model might be more useful. For example, estimating the rigidity of a system that has lifetime appointments as life expectancy increases, the idea of being that there may be an exponential or hi. High factor nonlinear relationship, which would suggest against life or

Yeah, I think I see where you're going with this:

if people live longer, "lifetime" appointments mean **slower turnover**, which can make the system **much more rigid** — and maybe *much more* than linearly.

Let me sketch a clean abstract model you could play with.

1. Minimal abstract setup

Say you have:

- N : number of seats (judges, directors, etc.)
- A : typical appointment age
- E : life expectancy (or expected retirement age)
- $T = E - A$: expected tenure length
- λ : average *per-seat* replacement rate (per year)

In a simple world:

$$\lambda \approx \frac{1}{T} = \frac{1}{E - A}$$

Total turnover per year:

$$\Lambda = N\lambda = \frac{N}{E - A}$$

A **naive rigidity index** could be "years per full refresh":

$$R_{\text{linear}} = \frac{1}{\Lambda} = \frac{E - A}{N}$$

So *purely demographically* this is linear in E : double average tenure → double rigidity.

That's the boring baseline.

2. Where the *nonlinearity* comes from

The interesting part is: decision-making isn't just "how often seats turn over," it's "how badly composition can lag behind the environment."

Imagine:

- There's a "current environment" or "current majority preference" $M(t)$
- The institution's actual composition is $C(t)$
- System "misalignment" is $\Delta(t) = \|C(t) - M(t)\|$
- We define **functional rigidity** as something like the long-run average misalignment:

$$R = \mathbb{E}[\Delta(t)]$$

Now assume:

- The environment drifts at some rate σ (social change, technology, norms)
- The institution updates only when seats turn over at rate Λ

In a bunch of simple stochastic / Markov-ish models, you get behavior like:

$$R \propto \frac{\sigma}{\Lambda^\alpha}$$

for some $\alpha \geq 1$.

- If $\alpha = 1$, rigidity grows **linearly** as turnover slows.
- If $\alpha > 1$, rigidity grows **superlinearly** (convex in E).

Intuitively, why might $\alpha > 1$?

- When turnover falls below some critical rate, **coalitions can entrench**:
 - Control agenda-setting
 - Gatekeep successors
 - Shape rules that further reduce effective turnover (e.g., norms against retirement)
- That means **effective** adaptation slows faster than the raw demographic turnover would suggest.

A simple stylized functional form that captures "blowing up" past a threshold:

$$R(E) = \frac{1}{\left(\frac{N}{E-A} - \Lambda_c\right)_+}$$

where Λ_c is a "critical" turnover rate the system needs to keep up, and $(x)_+ = \max(x, 0)$. As E increases and $\frac{N}{E-A}$ approaches Λ_c from above, $R(E)$ shoots up like a **hyperbola**. That's a very strong nonlinearity.

3. Concrete: lifetime vs fixed term

Compare:

1. Lifetime appointments

- Tenure $T_{\text{life}} \approx E - A$
- Turnover $\Lambda_{\text{life}} = N / (E - A)$

2. Fixed term K (e.g., 18-year term)

- Tenure $T_{\text{term}} = K$ (doesn't grow with life expectancy)
- Turnover $\Lambda_{\text{term}} = N / K$ (constant in E)

If rigidity is something like:

$$R \propto \frac{\sigma}{\Lambda^\alpha}$$

then:

- Under lifetime:

$$R_{\text{life}}(E) \propto \sigma \left(\frac{E - A}{N} \right)^\alpha$$

- Under fixed term:

$$R_{\text{term}} \propto \sigma \left(\frac{K}{N} \right)^\alpha \quad (\text{independent of } E)$$

So as E creeps up over decades, **lifetime** rigidity grows like $(E - A)^\alpha$, while **term-limited** rigidity stays flat.

If $\alpha > 1$, even modest increases in life expectancy cause **disproportionately large increases** in rigidity in the lifetime system, while the term-limited system is basically immune to that particular pressure.

That's exactly the kind of "exponential or high-factor nonlinear" you were gesturing at.

4. How you could actually use this

If you wanted to make this a real model:

1. **Pick a small number of seats**, say $N = 9$.

2. **Simulate**:

- Appointment times as a Poisson process with rate $\Lambda(E)$
- An ideology / position for each seat
- A drifting "public preference" process

3. **Measure**:

- Average misalignment over many runs
- How it scales with E under:
 - Lifetime rules

- Fixed K -year terms
4. Fit a simple function $R(E)$ and see if it's roughly linear, quadratic, or more like a blow-up near some critical turnover level.

You'd probably find that even if the **demographic** effect is linear, the **functional** effect (in terms of misalignment, polarization, time to flip majorities, etc.) is steeper and plausibly closer to what you're calling "exponential or high-factor nonlinear."

If you want, next step I can either:

- Turn this into explicit math (e.g., a tiny Markov model with closed-form expected lag), or
- Sketch Python pseudo-code for a quick simulation where you can sweep E and compare lifetime vs fixed-term behavior.



Let's start with the python.

Nice, let's turn it into something you can actually run and poke at.

Below is a **single self-contained Python file** that:

1. Simulates a drifting "environment" (public preference) as a random walk.
2. Simulates a court/board with:
 - **Lifetime tenure** = $E - A$ (life expectancy - appointment age)
 - **Fixed terms** = constant K years
3. Measures "rigidity" as the **average misalignment** between the institution's mean position and the environment.
4. Sweeps over different life expectancies and prints a comparison.

You can save this as **rigidity_sim.py** and run **python rigidity_sim.py**.

```
python
```

```
#!/usr/bin/env python3
"""
rigidity_sim.py
```

```
Toy simulation of institutional "rigidity" under lifetime vs fixed-term
appointments as life expectancy increases.
```

```
Idea:
```

- Environment (public preference) drifts as a random walk.
- Each seat has a fixed "position" set at appointment time.
- New appointees are drawn from a distribution centered on the current environment, plus some noise.
- Rigidity is measured as the average misalignment between the mean seat position and the environment over time.

```
We compare:
```

- Lifetime appointments with `tenure = life_expectancy - appointment_age`
- Fixed terms with a constant `term_length` (e.g., 18 years)

```

"""

from __future__ import annotations

from dataclasses import dataclass
from typing import List, Dict
import numpy as np

@dataclass
class SimConfig:
    n_seats: int = 9
    years: int = 300          # total simulated years
    burn_in: int = 100        # ignore early years for measuring rigidity
    env_sigma: float = 0.20    # year-to-year environment drift (random walk step std)
    appoint_sigma: float = 0.50 # noise of appointee around environment at appointment
    n_runs: int = 50          # independent runs for averaging
    seed: int | None = 123    # base RNG seed (set None for fully random)

def simulate_rigidity(
    tenure_years: int,
    cfg: SimConfig,
) -> float:
    """
    Simulate average misalignment ("rigidity") for a given tenure length.

    Parameters
    -----
    tenure_years : int
        How long each appointment lasts, in years. After this many years,
        a seat is reappointed to a new member whose position is drawn
        around the current environment.
    cfg : SimConfig
        Global simulation settings.

    Returns
    -----
    float
        Average |composition - environment| over time and runs,
        after burn-in.
    """
    assert tenure_years > 0, "tenure_years must be positive"
    rigidities: List[float] = []

    for run in range(cfg.n_runs):
        # Run-specific RNG
        rng = np.random.default_rng(
            None if cfg.seed is None else cfg.seed + run
        )

        # Environment starts at 0 and drifts
        env = 0.0

        # Initial appointments at env=0
        judges = rng.normal(loc=env, scale=cfg.appoint_sigma, size=cfg.n_seats)
        time_since_appointment = np.zeros(cfg.n_seats, dtype=int)

        misalignments: List[float] = []

        for t in range(cfg.years):
            # Environment random walk step
            env += rng.normal(loc=0.0, scale=cfg.env_sigma)

            # Mean position of the institution
            composition = float(judges.mean())

            # Misalignment measure (absolute difference)
            misalign = abs(composition - env)

            if t >= cfg.burn_in:
                misalignments.append(misalign)

            # Advance clock and reappoint where tenure has expired
            time_since_appointment += 1

```

```

        expired = time_since_appointment >= tenure_years
        if np.any(expired):
            n_expired = int(expired.sum())
            # New appointees drawn around *current* environment
            judges[expired] = rng.normal(
                loc=env, scale=cfg.appoint_sigma, size=n_expired
            )
            time_since_appointment[expired] = 0

    if misalignments:
        rigidities.append(float(np.mean(misalignments)))

    return float(np.mean(rigidities)) if rigidities else float("nan")

def explore_life_expectancy(
    appointment_age: int = 50,
    life_expectancies: List[int] | None = None,
    term_length: int = 18,
    cfg: SimConfig | None = None,
) -> List[Dict[str, float]]:
    """
    Sweep over life expectancies and compare rigidity for:

    - lifetime tenure = life_expectancy - appointment_age
    - fixed term = term_length (constant)

    Returns a list of dicts with results for each life expectancy.
    """
    if cfg is None:
        cfg = SimConfig()

    if life_expectancies is None:
        life_expectancies = [70, 75, 80, 85, 90]

    results: List[Dict[str, float]] = []

    for E in life_expectancies:
        tenure_lifetime = E - appointment_age
        if tenure_lifetime <= 0:
            raise ValueError(
                f"life_expectancy={E} <= appointment_age={appointment_age}"
            )

        R_lifetime = simulate_rigidity(tenure_years=tenure_lifetime, cfg=cfg)
        R_fixed = simulate_rigidity(tenure_years=term_length, cfg=cfg)

        results.append(
            {
                "life_expectancy": float(E),
                "appointment_age": float(appointment_age),
                "tenure_lifetime": float(tenure_lifetime),
                "tenure_fixed": float(term_length),
                "rigidity_lifetime": R_lifetime,
                "rigidity_fixed": R_fixed,
            }
        )

    return results

def main():
    # Example usage: compare lifetime vs 18-year terms
    cfg = SimConfig(
        n_seats=9,
        years=300,
        burn_in=100,
        env_sigma=0.20,
        appoint_sigma=0.50,
        n_runs=50,
        seed=123,
    )

    appointment_age = 50
    life_expectancies = [70, 75, 80, 85, 90]

```

```
term_length = 18

results = explore_life_expectancy(
    appointment_age=appointment_age,
    life_expectancies=life_expectancies,
    term_length=term_length,
    cfg=cfg,
)

print("Simulation results (rigidity = avg |composition - environment|):")
print(
    f"  n_seats={cfg.n_seats}, env_sigma={cfg.env_sigma}, "
    f"appoint_sigma={cfg.appoint_sigma}, n_runs={cfg.n_runs}\n"
)
print(
    f"{'E':>5} {'tenure_life':>11} {'tenure_fixed':>13} "
    f"{'R_life':>10} {'R_fixed':>10}"
)
print("-" * 55)

for row in results:
    E = int(row["life_expectancy"])
    tenure_life = row["tenure_lifetime"]
    tenure_fixed = row["tenure_fixed"]
    R_life = row["rigidity_lifetime"]
    R_fixed = row["rigidity_fixed"]
    print(
        f"{E:5d} {tenure_life:11.1f} {tenure_fixed:13.1f} "
        f"{R_life:10.3f} {R_fixed:10.3f}"
    )

if __name__ == "__main__":
    main()
```

If you tell me what environment you're running in (Anaconda / plain Python, whether you'd prefer no NumPy, etc.), I can tweak this to match your setup and/or add a quick plotting function to visualize how rigidity scales with life expectancy.