

Exhibit A

A permanent-magnet rotor demonstration in two parts

Postulate: Sustained kinetic motion from a dynamic magnetic field

Paul Gene Conlan

Zuze Project LLC

CDVEF.com

October 2026

Scope

We postulate sustained kinetic motion from a dynamic magnetic field.

Exhibit A is the rig built to test it: one rotor disc, permanent magnets only, no coils and no power supply. It runs in two parts. Part One suspends the disc and addresses the fall-off problem. Part Two drives rotation.

This document states the postulate and the demonstration. It argues nothing further.

Part One — Suspension and the fall-off problem

Part One holds the rotor without contact, so that friction does not mask what Part Two measures.

Two systems hold it. A magnetic channel around the rim faces like poles across a small gap on three surfaces — above, beside, and below. A coaxial ring pair near the axis carries the weight by repulsion.

This is the fall-off problem. Earnshaw's theorem states that static magnets cannot restore in every direction at once, so one axis will always be unstable. The model in Appendix A computes stiffness per surface and shows which axis that is. A rotor held in every direction but one, with that one carried by a minimal low-friction guide, is a clean platform for Part Two.

Part Two — Rotation

Part Two drives the suspended rotor with the same permanent magnets, set at an angle.

Eric Laithwaite's linear induction work demonstrated magnetic thrust along a straight track. Part Two wraps that geometry around a cylinder: a stationary magnet channel surrounding the rim, angled so the interaction force carries a tangential component. Only the tangential part turns the disc.

$$\mathbf{F}_t = \mathbf{F} \cdot \hat{\mathbf{t}}, \quad \tau = r \mathbf{F}_t$$

Four variables govern the result: the stator magnet angle θ , the working gap g , the rotor speed ω , and the rotor's moment of inertia I .

The measurement

The test maps torque against rotor position, $\tau(\phi, \theta, g)$, with the rotor locked at each step so that the field is measured rather than momentum. Each sweep covers a full interaction, approach through exit, because the opposing force lives on the exit side. The first matrix sweeps θ from 0° to 60° in 10° steps at three gap settings.

Each sweep produces one number, the net work over a complete revolution:

$$W = \oint \tau \, d\phi$$

A second test records spin-down time from a fixed starting speed, compared against a control rotor in the same housing with the stator magnets removed.

The gate geometry

The rotation cell used for the full-revolution run:

- Rotor: 18 in disc, 24 rim stations, rotor weight about 18.5 N.
- Magnets: 1/4 in N52 cubes, $B_r = 1.44$ T, modeled as point-dipole grids as in Appendix A.
- Working gap: 1/4 in.
- Gate angle: swept from 0° to 45° , with 20° as the reference case.
- Run: torque integrated over the full 360° , all 24 stations, without resetting rotor speed.

Result

The rotor crosses every gate. Speed varies through each station as the rotor enters and leaves the field: starting at 300 rpm, it ranged from about 299.97 to 301.23 rpm.

After a full revolution, the net magnetic work was zero to numerical precision at every angle tested. At the 20° reference case:

$$W_{\text{cycle}} = \oint_0^{2\pi} \tau \, d\phi \approx -7.8 \times 10^{-16} \text{ J}$$

With no losses, the rotor returns to exactly 300.00 rpm. With any real losses, it slows.

Standing condition

This geometry demonstrates lift, centering, and gate traversal. It does not yet demonstrate sustained motion. For the postulate to hold, the full cycle must return more energy than it loses:

$$\oint \tau \, d\phi > E_{\text{loss}}$$

For any arrangement of fixed permanent magnets, the closed-path integral is zero. A positive integral requires a field that changes during the cycle, and any input that changes it must be counted in the energy budget.

This is the record as of October 2026.

Appendix A: Exhibit A, Part One — static stability model

This model computes force and stiffness on the rotor from each magnetic surface separately: upper lip, wall, lower lip, and the coaxial lift pair. The rotor is an 18 in disc with 24 rim stations, each holding three 1/4 in N52 cubes faced by a matching stator channel across a 1/4 in gap, plus a 4 in OD lift ring pair near the axis. Magnets are modeled as grids of point dipoles, so both force and torque (including tilt) are included.

The final check line, $2 \times \text{radial} + \text{vertical}$, is a built-in self-test. For any arrangement of static magnets, Earnshaw's theorem requires the translational stiffnesses to sum to approximately zero. A reader running this code can therefore confirm the result is physically consistent, and see directly which axes are restoring and which are not.

```
"""
```

```
Exhibit A, three-face magnetic channel (static, no rotation).
```

```
Rotor: 18 in disc. At each of N stations around the rim, three rotor cubes:
    top cube (N up), radial cube (N out), bottom cube (N down).
```

```
Stator channel: matching cubes facing them across GAP:
```

```
    upper lip (N down), wall (N in), lower lip (N up) -> like poles face on
all three surfaces.
```

```
Plus the 4 in coaxial lift ring pair near the axis.
```

```
All magnets as grids of point dipoles. Force AND torque computed, so tilt is
```

```

included.
"""

import numpy as np, time
MU0 = 4e-7*np.pi; IN = 0.0254; G = 9.81; Br = 1.44; M = Br/MU0
K = MU0/(4*np.pi)

# ----- parameters -----
DISC_D, DISC_T = 18*IN, 0.25*IN
N_ST = 24          # stations around rim
C = 0.25*IN        # cube edge
GAP = 0.25*IN      # gap on all three channel faces
SUB = 4            # sub-dipoles per cube edge
LIFT_OD, LIFT_ID, LIFT_H = 4*IN, 2*IN, 0.5*IN
G_LIFT = 0.0824    # lift gap from Part One equilibrium
# -----
R = DISC_D/2

def cubes(centres, dirs):
    s = (np.arange(SUB)+.5)/SUB*C - C/2
    g = np.stack(np.meshgrid(s, s, s, indexing='ij'), -1).reshape(-1, 3)
    P = (centres[:, None, :] + g[None]).reshape(-1, 3)
    Mo = np.repeat(dirs*M*C**3/SUB**3, len(g), axis=0)
    return P, Mo

def channel():
    th = 2*np.pi*np.arange(N_ST)/N_ST
    er = np.stack([np.cos(th), np.sin(th), 0*th], 1); ez = np.array([0, 0,
1.0])
    rc = R - C/2
    zt = C          # rotor top cube centre (stacked on radial
cube)
    rot_c = np.concatenate([rc*er + zt*ez, rc*er, rc*er - zt*ez])
    rot_d = np.concatenate([np.tile(ez, (N_ST, 1)), er, np.tile(-ez, (N_ST,
1))])
    zs = zt + C + GAP
    sta_c = np.concatenate([rc*er + zs*ez, (R+GAP+C/2)*er, rc*er - zs*ez])
    sta_d = np.concatenate([np.tile(-ez, (N_ST, 1)), -er, np.tile(ez, (N_ST,
1))])
    return cubes(rot_c, rot_d), cubes(sta_c, sta_d)

def lift_ring(ztop, sign, na=72, nr=4, nz=4):
    r = LIFT_ID/2 + (np.arange(nr)+.5)/nr*(LIFT_OD-LIFT_ID)/2
    z = ztop - (np.arange(nz)+.5)/nz*LIFT_H

```

```

t = 2*np.pi*(np.arange(na)+.5)/na
rr, zz, tt = [a.ravel() for a in np.meshgrid(r, z, t, indexing='ij')]
P = np.stack([rr*np.cos(tt), rr*np.sin(tt), zz], 1)
dV = (np.pi*((LIFT_OD/2)**2-(LIFT_ID/2)**2)*LIFT_H)/len(rr)
return P, np.tile([0, 0, sign*M*dV], (len(rr), 1))

def FT(rot, sta):
    """Total force and torque (about disc centre) on rotor dipoles from
    stator dipoles."""
    (Pr, Mr), (Ps, Ms) = rot, sta
    F = np.zeros(3); T = np.zeros(3)
    for i in range(0, len(Pr), 256):
        p, m2 = Pr[i:i+256], Mr[i:i+256]
        r = p[:, None] - Ps[None]; rn2 = np.sum(r*r, 2, keepdims=True); rn =
np.sqrt(rn2)
        m1 = Ms[None]
        a = np.sum(m1*r, 2, keepdims=True); b = np.sum(m2[:, None]*r, 2,
keepdims=True)
        c = np.sum(m1*m2[:, None], 2, keepdims=True)
        f = (3*K/rn**5*(a*m2[:, None] + b*m1 + c*r - 5*a*b*r/rn2)).sum(1)
        B = (K*(3*a*r/rn**5 - m1/rn**3)).sum(1)
        F += f.sum(0); T += (np.cross(p, f) + np.cross(m2, B)).sum(0)
    return F, T

def move(rot, dx=0, dz=0, tilt=0):
    """Tilt about x-axis through disc centre, then translate."""
    P, Mo = rot; c, s = np.cos(tilt), np.sin(tilt)
    Rx = np.array([[1, 0, 0], [0, c, -s], [0, s, c]])
    return P@Rx.T + [dx, 0, dz], Mo@Rx.T

(rotC, staC) = channel()
rotL = lift_ring(-DISC_T/2, -1) # under disc, N down
staL = lift_ring(-DISC_T/2 - LIFT_H - G_LIFT, +1) # below, N up -> repel

# split channel by face so each surface is reported separately
n = N_ST*SUB**3
face = {'upper lip': slice(0, n), 'wall': slice(n, 2*n), 'lower lip':
slice(2*n, 3*n)}
sub = lambda arr, sl: (arr[0][sl], arr[1][sl])
parts = {k: (sub(rotC, v), sub(staC, v)) for k, v in face.items()}
parts['lift pair'] = (rotL, staL)

t0 = time.time()

```

```

print("CHECK centred:")
for k, (ro, st) in parts.items():
    F, T = FT(ro, st); print(f" {k:10s} F = {np.round(F, 3)} N")
print(f" (lift pair F_z should equal disc weight ~18-19 N)\n")

d, a = 1e-3, np.radians(0.25)
print("STIFFNESS by surface (restoring = +)")
print(f"'surface':11s} {'radial N/m':>11s} {'vertical N/m':>13s} {'tilt  
N*m/rad':>13s}")
tot = np.zeros(3)
for k, (ro, st) in parts.items():
    kx = -(FT(move(ro, dx=d), st)[0][0] - FT(move(ro, dx=-d), st)[0][0])/(2*d)
    kz = -(FT(move(ro, dz=d), st)[0][2] - FT(move(ro, dz=-d), st)[0][2])/(2*d)
    kt = -(FT(move(ro, tilt=a), st)[1][0] - FT(move(ro, tilt=-a), st)[1][0])/(2*a)
    tot += [kx, kz, kt]
    print(f"{k:11s} {kx:11.0f} {kz:13.0f} {kt:13.1f}")
print(f"'TOTAL':11s} {tot[0]:11.0f} {tot[1]:13.0f} {tot[2]:13.1f}")
print(f"check 2*radial + vertical = {2*tot[0]+tot[1]:.0f} (translations must  
sum ~0)")
print(f"\n[{time.time()-t0:.0f} s]")

```