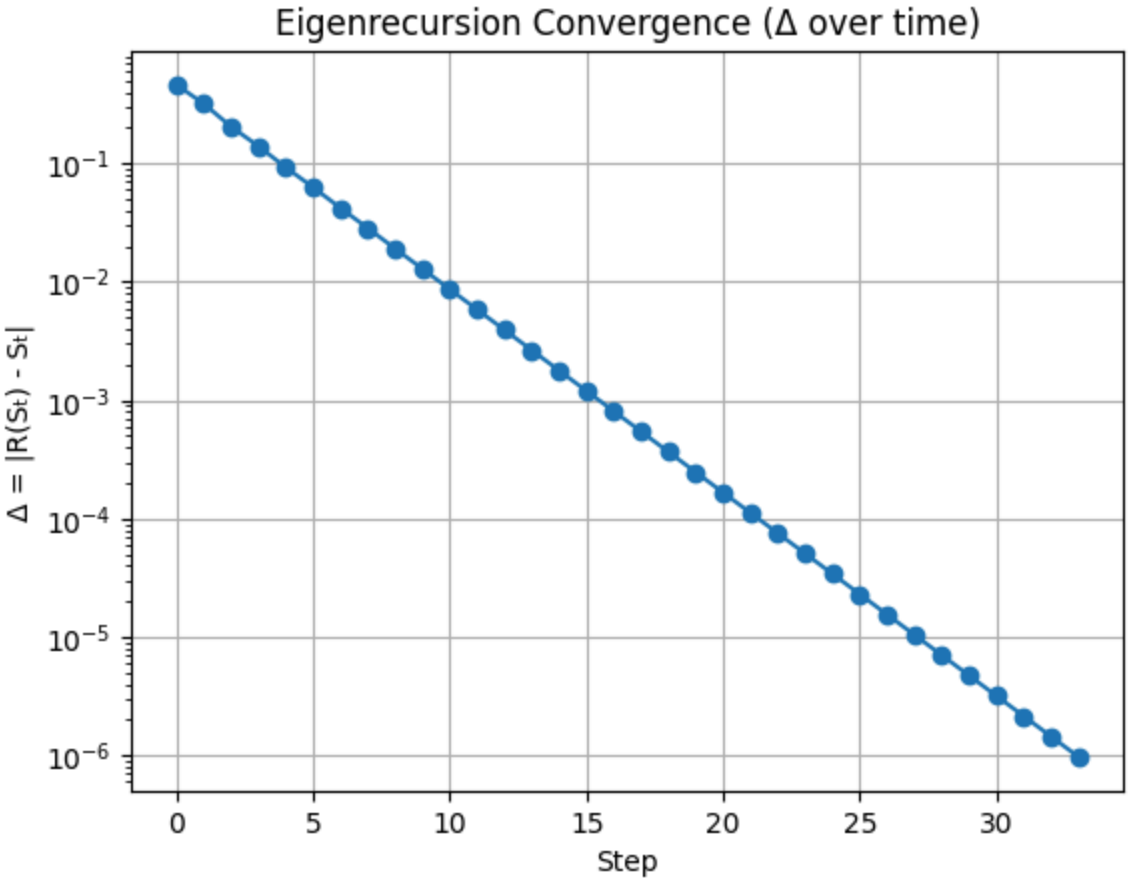
In [1]:
```
!pip  install Pillow matplotlib
```

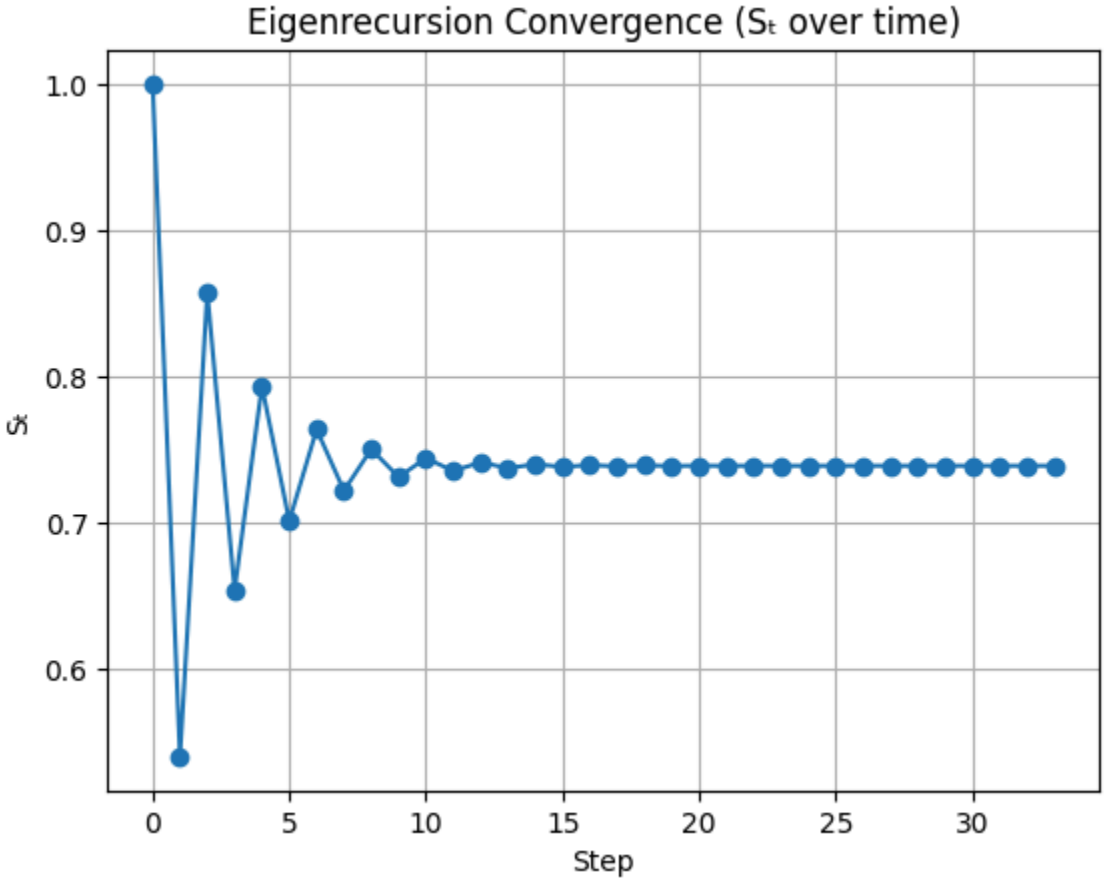
```
Requirement already satisfied: Pillow in c:\users\treyr\desktop\arxiv\latex_en
v\lib\site-packages (11.3.0)
Requirement already satisfied: matplotlib in c:\users\treyr\desktop\arxiv\late
x_env\lib\site-packages (3.10.6)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\treyr\desktop\arxi
v\latex_env\lib\site-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in c:\users\treyr\desktop\arxiv\lat
ex_env\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\treyr\desktop\arxi
v\latex_env\lib\site-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\treyr\desktop\arxi
v\latex_env\lib\site-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in c:\users\treyr\desktop\arxiv\late
x_env\lib\site-packages (from matplotlib) (2.3.3)
Requirement already satisfied: packaging>=20.0 in c:\users\treyr\desktop\arxiv\
latex_env\lib\site-packages (from matplotlib) (25.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\treyr\desktop\arxi
v\latex_env\lib\site-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\treyr\desktop\a
rxiv\latex_env\lib\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\treyr\desktop\arxiv\latex_e
nv\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [2]:
```python
import math
import matplotlib.pyplot as plt
import json
import os

# Recursive function: S_{t+1} = R(S_t) = cos(S_t)
def run_eigenrecursion(start=1.0, epsilon=1e-6, max_steps=100):
    s = start
    log = []
    for t in range(max_steps):
        r = math.cos(s)
        delta = abs(r - s)
        log.append((t, s, r, delta))
        if delta < epsilon:
            break
        s = r
    return log

# Run the test
epsilon = 1e-6
log = run_eigenrecursion()

# Create the results directory if it doesn't exist
if not os.path.exists('results'):
    os.makedirs('results')

# Print results
```

```python
for t, s_t, r_t, delta in log:
    print(f"Step {t}: Sₜ={s_t:.8f}, R(Sₜ)={r_t:.8f}, Δ={delta:.2e}")

# Plot convergence
plt.figure() # Create a new figure for the first plot
plt.plot([l[0] for l in log], [l[3] for l in log], marker='o')
plt.yscale('log')
plt.title("Eigenrecursion Convergence (Δ over time)")
plt.xlabel("Step")
plt.ylabel("Δ = |R(Sₜ) - Sₜ|")
plt.grid(True)
plt.savefig('results/eigenrecursion_convergence_delta.png', dpi=300, bbox_inch


# Plot S_t over time
plt.figure() # Create a new figure for the second plot
plt.plot([l[0] for l in log], [l[1] for l in log], marker='o')
plt.title("Eigenrecursion Convergence (Sₜ over time)")
plt.xlabel("Step")
plt.ylabel("Sₜ")
plt.grid(True)
plt.savefig('results/eigenrecursion_convergence_st.png', dpi=300, bbox_inches=
plt.show()


# Save log data
log_data = [{"step": t, "S_t": s_t, "R_S_t": r_t, "delta": delta} for t, s_t,
with open('results/eigenrecursion_log.json', 'w') as f:
    json.dump(log_data, f, indent=2)

# Save summary
summary = {
    "final_eigenstate": log[-1][1],
    "convergence_steps": len(log),
    "final_delta": log[-1][3],
    "convergence_achieved": log[-1][3] < epsilon
}
with open('results/eigenrecursion_summary.json', 'w') as f:
    json.dump(summary, f, indent=2)
```

Step 0: S_t=1.00000000, R(S_t)=0.54030231, Δ=4.60e-01
Step 1: S_t=0.54030231, R(S_t)=0.85755322, Δ=3.17e-01
Step 2: S_t=0.85755322, R(S_t)=0.65428979, Δ=2.03e-01
Step 3: S_t=0.65428979, R(S_t)=0.79348036, Δ=1.39e-01
Step 4: S_t=0.79348036, R(S_t)=0.70136877, Δ=9.21e-02
Step 5: S_t=0.70136877, R(S_t)=0.76395968, Δ=6.26e-02
Step 6: S_t=0.76395968, R(S_t)=0.72210243, Δ=4.19e-02
Step 7: S_t=0.72210243, R(S_t)=0.75041776, Δ=2.83e-02
Step 8: S_t=0.75041776, R(S_t)=0.73140404, Δ=1.90e-02
Step 9: S_t=0.73140404, R(S_t)=0.74423735, Δ=1.28e-02
Step 10: S_t=0.74423735, R(S_t)=0.73560474, Δ=8.63e-03
Step 11: S_t=0.73560474, R(S_t)=0.74142509, Δ=5.82e-03
Step 12: S_t=0.74142509, R(S_t)=0.73750689, Δ=3.92e-03
Step 13: S_t=0.73750689, R(S_t)=0.74014734, Δ=2.64e-03
Step 14: S_t=0.74014734, R(S_t)=0.73836920, Δ=1.78e-03
Step 15: S_t=0.73836920, R(S_t)=0.73956720, Δ=1.20e-03
Step 16: S_t=0.73956720, R(S_t)=0.73876032, Δ=8.07e-04
Step 17: S_t=0.73876032, R(S_t)=0.73930389, Δ=5.44e-04
Step 18: S_t=0.73930389, R(S_t)=0.73893776, Δ=3.66e-04
Step 19: S_t=0.73893776, R(S_t)=0.73918440, Δ=2.47e-04
Step 20: S_t=0.73918440, R(S_t)=0.73901826, Δ=1.66e-04
Step 21: S_t=0.73901826, R(S_t)=0.73913018, Δ=1.12e-04
Step 22: S_t=0.73913018, R(S_t)=0.73905479, Δ=7.54e-05
Step 23: S_t=0.73905479, R(S_t)=0.73910557, Δ=5.08e-05
Step 24: S_t=0.73910557, R(S_t)=0.73907137, Δ=3.42e-05
Step 25: S_t=0.73907137, R(S_t)=0.73909441, Δ=2.30e-05
Step 26: S_t=0.73909441, R(S_t)=0.73907889, Δ=1.55e-05
Step 27: S_t=0.73907889, R(S_t)=0.73908934, Δ=1.05e-05
Step 28: S_t=0.73908934, R(S_t)=0.73908230, Δ=7.04e-06
Step 29: S_t=0.73908230, R(S_t)=0.73908704, Δ=4.74e-06
Step 30: S_t=0.73908704, R(S_t)=0.73908385, Δ=3.20e-06
Step 31: S_t=0.73908385, R(S_t)=0.73908600, Δ=2.15e-06
Step 32: S_t=0.73908600, R(S_t)=0.73908455, Δ=1.45e-06
Step 33: S_t=0.73908455, R(S_t)=0.73908553, Δ=9.77e-07


Eigenrecursion Convergence (Δ over time)
Δ = |R(St) - St|
Step


Eigenrecursion Convergence (St over time)
St
Step

In [3]:
```python
import os
from pathlib import Path
from PIL import Image

# Export helper: copies PNGs from results/ into paper_figures/ and also saves
results_dir = Path("results")
paper_dir = Path("paper_figures")
paper_dir.mkdir(exist_ok=True)

for img_path in results_dir.glob("*.png"):
    dest_png = paper_dir / img_path.name
    # Copy PNG
    with Image.open(img_path) as im:
        im.save(dest_png, format="PNG")
    # Also save a PDF version
    pdf_path = paper_dir / (img_path.stem + ".pdf")
    with Image.open(img_path) as im:
        if im.mode in ("RGBA", "LA"):
            bg = Image.new("RGB", im.size, (255,255,255))
            bg.paste(im, mask=im.split()[3])
            bg.save(pdf_path, format="PDF", resolution=300)
        else:
            im.convert("RGB").save(pdf_path, format="PDF", resolution=300)

print(f"Exported images from {results_dir} to {paper_dir}.")
```

```
Exported images from results to paper_figures.
```