# Real UNNS “echo” form

## A Concrete, Applied Example

A concrete, applied example that shows how one can do what’s usually done with complex numbers **entirely in real UNNS “echo” form**. It's done in two staples side-by-side:

1. a (damped) harmonic oscillator (physics), and
2. a single Fourier mode / phasor (signal processing),

both as **real 2×2 rotation–dilation recursions** (echo cycles), no $i$i required.

---

# 1) Harmonic oscillator via UNNS echo (no complex)

## 1.1 Undamped SHO

Continuous equation:

$$x''(t) + \omega^2 x(t) = 0, \quad x(0) = x_0, \quad x'(0) = v_0.$$

x″(t)+ω2x(t)=0,x(0)=x0,x′(0)=v0.

Classically you’d write $x(t) = A\cos(\omega t) + B\sin(\omega t)$x(t)=Acos(ωt)+Bsin(ωt) or use $e^{i\omega t}$eiωt.
In UNNS, we keep it **real** and step by a fixed time $h > 0$h>0 using the **exact flow**:

Define the state $s(t) = \begin{bmatrix} x(t) \\ v(t)/\omega \end{bmatrix}$s(t)=[x(t)v(t)/ω]. Over step $h$h,

$$s(t+h) \;=\; R(\theta)\, s(t), \qquad R(\theta) \;=\; \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \quad \theta = \omega h.$$

s(t+h)=R(θ)s(t),R(θ)=[cosθ−sinθsinθcosθ],θ=ωh.

This matrix is the **quarter-turn echo engine** (a planar rotation). No complex algebra, same physics.

**Discrete UNNS recursion (exact):**

$$\begin{bmatrix} x_{n+1} \\ v_{n+1}/\omega \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_n \\ v_n/\omega \end{bmatrix}, \qquad \theta = \omega h.$$

[xn+1vn+1/ω]=[cosθ−sinθsinθcosθ][xnvn/ω],θ=ωh.

- Energy is preserved: $x_n^2 + (v_n/\omega)^2$xn2+(vn/ω)2 is constant (pure rotation).
- This is exactly what $e^{i\omega t}$eiωt does, but as a **real rotation**.

## 1.2 Damped SHO

$$x'' + 2\zeta\omega\, x' + \omega^2 x = 0, \qquad 0 < \zeta < 1.$$

x″+2ζωx′+ω2x=0,0<ζ<1.

Exact step over $h$h:

$$s_{n+1} \;=\; \rho\, R(\theta_d)\, s_n, \quad \rho = e^{-\zeta\omega h}, \quad \theta_d = \omega\sqrt{1-\zeta^2}\, h.$$

sn+1=ρR(θd)sn,ρ=e−ζωh,θd=ω1−ζ2
√
h.

So it’s a **rotation–dilation**: same rotation, multiplied by $\rho < 1$ρ<1. Still all-real.

- Amplitude decays like $\rho^n = e^{-\zeta\omega n h}$ρn=e−ζωnh.
- Phase advances by $\theta_d$θd each step.

This is exactly the complex solution $e^{(-\zeta\omega \pm i\omega_d)t}$e(−ζω±iωd)t, but **as a real echo**.

### Tiny pseudocode (language-agnostic)

```
given x0, v0, ω, ζ (damping ∈[0,1)), step h, steps N
θ  = ω*sqrt(max(1-ζ*ζ, 0))*h
ρ  = exp(-ζ*ω*h)
c  = cos(θ); s = sin(θ)

X = x0
U = v0/ω  # scaled velocity
for n = 1..N:
    X_new =  ρ*( c*X + s*U)
    U_new =  ρ*(-s*X + c*U)
    X, U  =  X_new, U_new
    # physical velocity v = ω*U
```

**Why this matters:** all computations are real; you can run this on hardware that doesn’t want complex arithmetic. Same fidelity.

---

# 2) Single Fourier mode / “phasor” without $i$i

Suppose a discrete signal has a pure tone with angular step $\theta$θ. Classically: $e^{in\theta}$einθ.
In UNNS/real arithmetic, use the **second-order real recursion** (Chebyshev form):

$$y_{n+1} = 2\cos\theta\, y_n - y_{n-1},$$

yn+1=2cosθyn−yn−1,

with, say, $y_0 = 1,\ y_1 = \cos\theta$y0=1,y1=cosθ. Then $y_n = \cos(n\theta)$yn=cos(nθ), and the quadrature $q_n = \sin(n\theta)$qn=sin(nθ) is tracked by the companion rotation:

$$\begin{bmatrix} y_{n+1} \\ q_{n+1} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} y_n \\ q_n \end{bmatrix}.$$

[yn+1qn+1]=[cosθ−sinθsinθcosθ][ynqn].

Again: pure rotation in $R^2$R2. No $C$C needed.

**Under damping / windowing** (e.g., exponential window), just multiply by $\rho < 1$ρ<1 each step — exactly like the damped oscillator:

$$\begin{bmatrix} y_{n+1} \\ q_{n+1} \end{bmatrix} = \rho \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} y_n \\ q_n \end{bmatrix}.$$

[yn+1qn+1]=ρ[cosθ−sinθsinθcosθ][ynqn].

---

## Where this is immediately useful

- **Education:** demystify complex numbers — show “imaginary” as **real rotation**.
- **Embedded / GPU computing:** keep pipelines real; avoid complex datatypes.
- **Signal processing:** phasors and Fourier modes as real recursions (helps on constrained hardware).
- **Physics sims:** oscillators, wave packets, AC circuits, normal modes — all via rotation–dilation.

---

## Bonus: “UNNS repair/normalization” hook

If your data are noisy or quantized, insert a UNNS operator after each step:

- **Collapse/threshold:** zero out tiny components (noise floor).
- **Inlaying:** snap $(y, q)$(y,q) to a target lattice (e.g., Gaussian/Eisenstein) for numerical robustness.
- **Damping:** apply $\rho$ρ to control energy/entropy.

These keep the real recursion **stable** while preserving the dominant resonance (frequency).