```julia
using Plots, PlutoUI, DynamicalSystems, Images, LinearAlgebra, JLD2
```

$$p_{n+1} = a\,[\,p_n - (\,T_n - u_n\,)\,] + b\,(\,y_n - T_n\,) + P_1^{(3)}(\,p_n, y_n\,)$$

$$y_{n+1} = c\,[\,p_n - (\,T_n - u_n\,)\,] + d\,(\,y_n - T_n\,) + P_2^{(3)}(\,p_n, y_n\,)$$

$$T_{n+1} = A\,[\,p_n - (\,T_n - u_n\,)\,] + B\,(\,T_n - \tau\,) + T_n$$

?

$$u_{n+1} = T_n$$

$$e_n = p_n - (\,T_n - T_{n-1}\,)$$

```julia
begin
    img = load("./PRETypeI.png")
    img2 = load("./ParameterAB.png")
    img3 = load("./Ecuaciones.png")
end
```

```
adaptive (generic function with 1 method)
```

```julia
function adaptive(u,par,t)
    p, y, T, s = u
    a, b, c, d, α, β, γ, δ, A, B, τ = par
    e = p - (T - s)
    pn = a*e + b*(y-T) + α*e^3 + β*e*(y-T)^2 + γ*(y - T)^3
    yn = c*e + d*(y-T) + T + δ*e^2
    Tn = A*e + B*(T - τ) + T
    sn = T
    return  SVector(pn, yn, Tn, sn)
end
```

```
adaptives (generic function with 1 method)
```

```julia
function adaptives(u,par,t)
    q, y, T = u
    a, b, c, d, α, β, γ, δ, A, B, τ = par
    e = q - T
    x = y - T
    qn = a*e + b*x + + α*e^3 + β*e*x^2 + γ*x^3 + T
    yn = c*e + d*x + δ*e^2 + T
    Tn = A*e + B*(T - τ) + T
    return  SVector(qn, yn, Tn)
end
```

```
adaptive3 (generic function with 1 method)
```

```julia
function adaptive3(u,par,t)
    e, x, s = u
    a, b, c, d, α, β, γ, δ, A, B, τ = par
    en = (a-A)*e + b*x + + α*e^3 + β*e*x^2 + γ*x^3 - B*s
    xn = (c-A)*e + d*x + δ*e^2 - B*s
    sn = A*e + (B+1)*s
    return  SVector(en, xn, sn)
end
```

```
p = [0.981, 0.266, -0.823, 0.0238, -2.21e-5, -7.84e-5, 5.34e-5, 0.00335, -0.2, 0.0, 500.0]
```

```julia
# parameters for Type I
p = [ 0.981, 0.266, -0.823, 0.0238, -2.21e-5, -7.84e-5, 5.34e-5, 3.35e-3, A, B, 500]
```

```
p3 = [0.981, 0.266, -0.823, 0.0238, -2.21e-5, -7.84e-5, 5.34e-5, 0.00335, -0.2, 0.0, 0.0]
```

```julia
p3 = [ 0.981, 0.266, -0.823, 0.0238, -2.21e-5, -7.84e-5, 5.34e-5, 3.35e-3, A, B, 0]
```

```
u0 = [0.0, 500.0, 550.0, 500.0]
```

```julia
u0 = [0, 500, Tpost, 500.0]
```

```julia
ds = DiscreteDynamicalSystem(adaptive, u0, p);
```

```
u02 = [500.0, 500.0, 550.0]
```

```julia
u02 = [500.0, 500.0, Tpost]
```

```julia
ds2 = DiscreteDynamicalSystem(adaptives, u02, p);
```

```
u03 = [-50.0, -50.0, 50.0]
```

```julia
u03 = [500.0-Tpost, 500.0-Tpost, Tpost-τ]
```

```julia
ds3 = DiscreteDynamicalSystem(adaptive3, u03, p3);
```

```julia
tr, t = trajectory(ds, 90);
```

```julia
tr2, t2 = trajectory(ds2, 90);
```

```
StateSpaceSets.StateSpaceSet{3, Float64, StaticArraysCore.SVector{3, Float64}}: [StaticArraysCore.
```

```julia
tr2
```

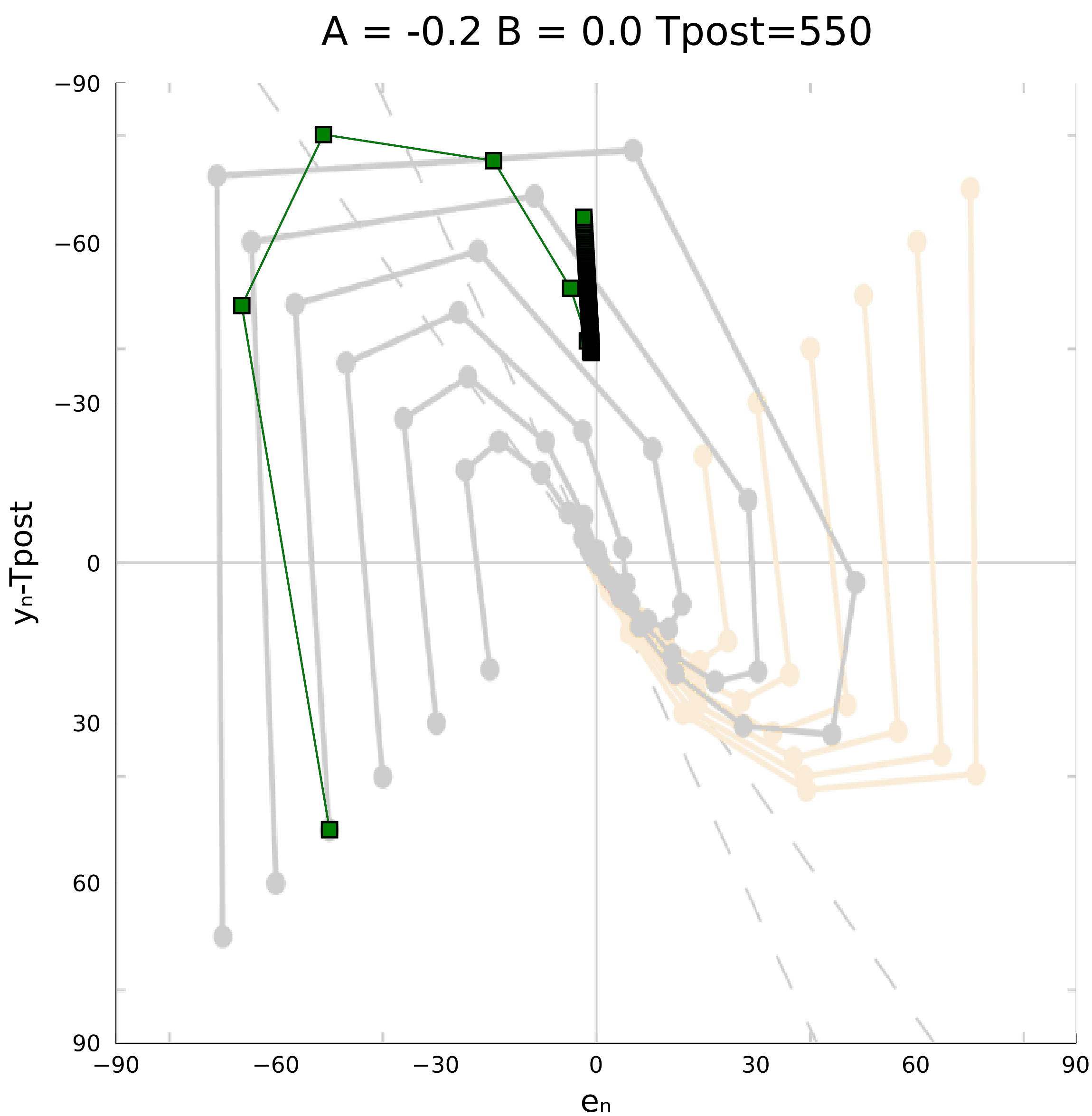
```
-0.623
```

```julia
c-A
```

```julia
tr3, t3 = trajectory(ds3, 90);
```

```
StateSpaceSets.StateSpaceSet{3, Float64, StaticArraysCore.SVector{3, Float64}}: [StaticArraysCore.
```

```
tr3
```




```
begin
    plot(LinRange(-109,97,size(img)[1]),LinRange(-110,99,size(img)[2]),img,yflip=true)
    plot!(Matrix(tr)[:,1]-(Matrix(tr)[:,3]-Matrix(tr)[:,4]),-Matrix(tr)[:,2] .+ Tpost,m=:circle,c=:red,label="")
    plot!(Matrix(tr2)[:,1]-Matrix(tr2)[:,3],-Matrix(tr)[:,2] .+ Tpost,m=:cross,c=:blue,xlabel="eₙ",ylabel="yₙ-Tpost",title="A = $A B = $B Tpost=$Tpost",size=(600,600),xlims=(-90,90),ylims=(-90,90),label="")
    plot!(Matrix(tr3)[:,1],-Matrix(tr3)[:,2]-Matrix(tr3)[:,3].-τ.+Tpost,m=:square,c=:green,label="")
end
```

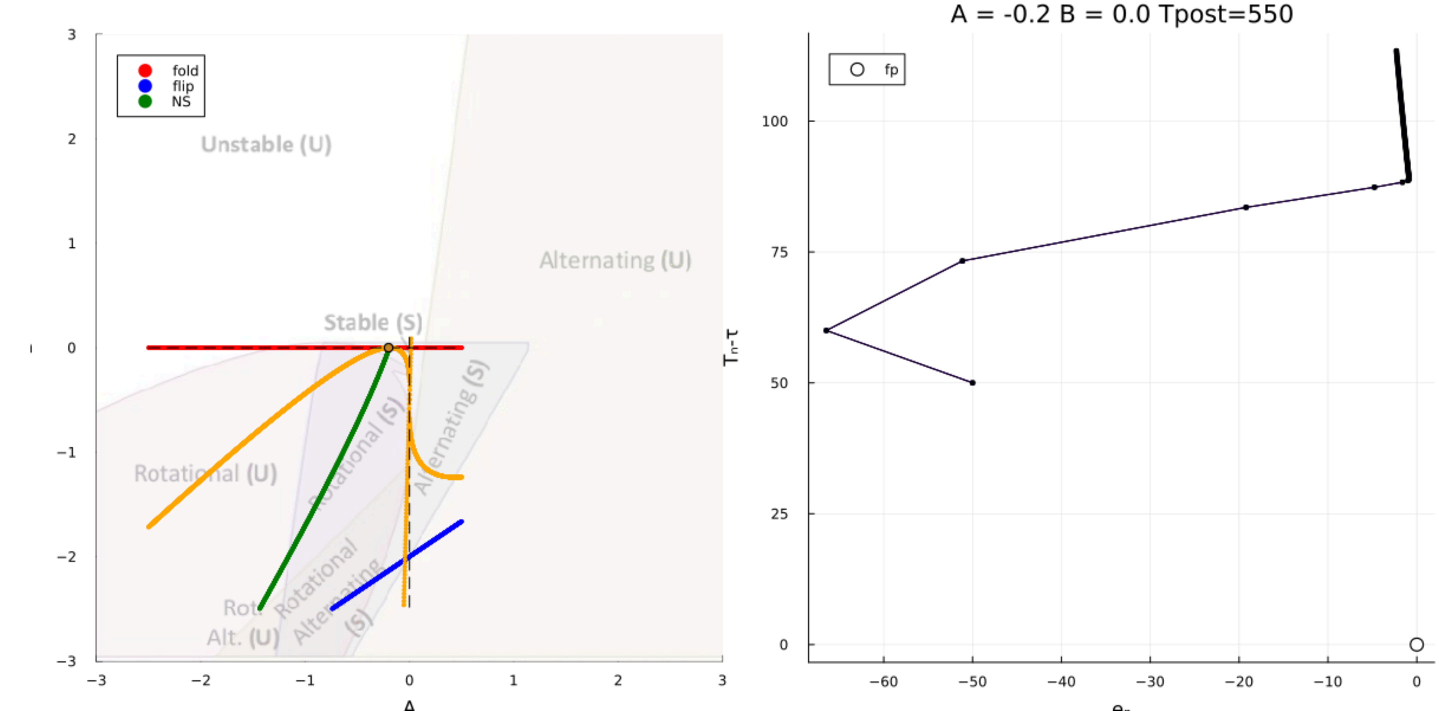
```
0.1:-0.001:-2.5
```

```
begin
    cp = jldopen("./criticalpoints.jld2")
    critical_points = cp["critical_points"]
    A0_v = cp["A0_v"]
    B0_v = cp["B0_v"]
end
```

A : -0.2

B : 0.0

Tpost : 550




```
begin
    p1 = plot(LinRange(-3.4,3.3,size(img2)[1]),LinRange(-3.4,3.3,size(img2)[2]),img2,yflip=false)
    fold = @. critical_points[getindex(critical_points,3) == 1]
    scatter!(getindex.(fold,1),getindex.(fold,2),ms=2,msw=0,c=:red,label="fold")
    flip = @. critical_points[getindex(critical_points,3) == 2]
    scatter!(getindex.(flip,1),getindex.(flip,2),ms=2,msw=0,c=:blue,label="flip")
    ns = @. critical_points[getindex(critical_points,3) == 3]
    scatter!(getindex.(ns,1),getindex.(ns,2),ms=2,msw=0,c=:green,label="NS")
    fc = @. critical_points[getindex(critical_points,3) == 4]
    scatter!(getindex.(fc,1),getindex.(fc,2),ms=2,msw=0,c=:orange,label="")
    plot!(A0_v,A0_v*0,c=:black,ls=:dash,label="")
    plot!(B0_v*0,B0_v,c=:black,ls=:dash,label="")
    scatter!([A],[B],ylims=(-3,3),xlims=(-3,3),xlabel="A",ylabel="B",label="")
    p2 = plot(Matrix(tr)[:,1]+Matrix(tr)[:,4]-Matrix(tr)[:,3],Matrix(tr)[:,3].-τ,m=:circle,ms=2,c=:red,label="")
    plot!(Matrix(tr2)[:,1]-Matrix(tr2)[:,3],Matrix(tr2)[:,3].-τ,m=:cross,ms=2,c=:blue,label="")
    plot!(Matrix(tr3)[:,1],Matrix(tr3)[:,3],m=:square,ms=2,c=:black,xlabel="eₙ",ylabel="Tₙ-τ",title="A = $A B = $B Tpost=$Tpost",label="")
    scatter!([0],[0],c=:white,ms=6,label="fp")
    plot(p1,p2,layout=(1,2),size=(1200,600))
end
```

```
[0.191231, 1.0, 1.01357]
```

```
eigen(M).values
```

```
3×3 Matrix{Float64}:
  0.33719    -0.57735  0.542157
 -0.93906    -0.57735  0.606455
  0.0668518  -0.57735  0.581617
```

```
eigen(M).vectors
```

```
[0.0563322, -0.0816985, -1.58199]
```

```julia
lyapunovspectrum(ds2,60)
```

```
[0.981, 0.266, -0.823, 0.0238, -2.21e-5, -7.84e-5, 5.34e-5, 0.00335, -0.2, 0.0, 500.0]
```

```julia
a, b, c, d, α, β, γ, δ, A_, B_, τ = p
```

```
M = 3×3 Matrix{Float64}:
   0.981  0.266   -0.247
  -0.823  0.0238   1.7992
  -0.2    0.0      1.2
```

```julia
M = [a b 1-(a+b); c d 1-(c+d); A_ 0 (1-A_+B_)]
```