```
using Plots, Symbolics, Nemo, JLD2
```

[A, B, x, a_, b_, c_, d_]

```
@variables A,B,x,a_,b_,c_,d_
```

[0.981, 0.266, -0.823, 0.0238]

```
a,b,c,d = [ 0.981, 0.266, -0.823, 0.0238]
```

1.0048

```
begin
    C = a*d-b*c
    D = b-d
    E = a+d
end
```

expr =

$$0.47859A + 0.21892\left(1 - A + B - x\right) + 0.247A\left(0.0238 - x\right) + \left(1 - A + B - x\right)\left(0.981 - x\right)\left(0.($$

```
expr = (a-x)*(d-x)*(1-A+B-x)-b*c*(1-A+B-x)+b*A*(1-c-d)-A*(1-a-b)*(d-x)
```

exprb =

$$AB\left(-0.2422 - x\right) + \left(-0.266\left(-0.823 - A\right) + \left(0.981 - A - x\right)\left(0.0238 - x\right)\right)\left(1 + B - x\right)$$

```
exprb = B*A*(d-x-b)+(B+1-x)*((d-x)*(a-A-x)-b*(c-A))
```

exprc =

$$0.24227 + 0.2422A + 0.24227B - 1.2471x + 0.7578Ax - 1.0048Bx + 2.0048x^2 - x^2A + x^2B -$$

```
exprc = C + D*A + C*B - (C+E)*x + (1-D)*A*x - E*B*x + (1+E)*x^2 - A*x^2 + B*x^2 - x^3
```

exprb2 =

$$0.24227 + 0.2422A + 0.24227B - 1.2471x + 0.7578Ax - 1.0048Bx + 2.0048x^2 - x^2A + x^2B -$$

```
exprb2 = expand(exprb)
```

expr2 =

$$0.24227 + 0.2422A + 0.24227B - 1.2471x + 0.7578Ax - 1.0048Bx + 2.0048x^2 - x^2A + x^2B -$$

```
expr2 = expand(expr)
```

sol =

$$\left[\frac{4514408266476185}{6755399441055744} - \frac{1}{3}A + \frac{1}{3}B + \sqrt[3]{\frac{88829417906467827179388663900287475243052352 9}{30828550162448733430858976940109094945867327078 4}} + \frac{123736066}{912708432}\right.$$

```
sol = symbolic_solve(exprb2, x)
```

$$\left[\frac{4514408266476185}{6755399441055744} - \frac{1}{3}A + \frac{1}{3}B + \sqrt[3]{\frac{88829417906467827179388663900287475243052352 9}{30828550162448733430858976940109094945867327078 4}} + \frac{123736066}{912708432}\right.$$

```
simplify(sol)
```

[202]

```
begin
    A0v = -2:0.01:1
    B0 = 0
    l1 = [substitute(sol[1], Dict([A => A0, B=> B0])) for A0 in A0v];
    l2 = [substitute(sol[2], Dict([A => A0, B=> B0])) for A0 in A0v];
    l3 = [substitute(sol[3], Dict([A => A0, B=> B0])) for A0 in A0v];
    d1 = findall(diff(abs.(l1) .> 2) .!= 0.0 .&& abs.(diff(abs.(l1))) .<  0.01)
    d2 = findall(diff(abs.(l2) .> 2) .!= 0.0 .&& abs.(diff(abs.(l2))) .<  0.01)
    d3 = findall(diff(abs.(l3) .> 2) .!= 0.0 .&& abs.(diff(abs.(l3))) .<  0.01)
    f1 = findall(diff(abs.(imag(l1)) .> 1e-12) .!= 0.0 )
    f2 = findall(diff(abs.(imag(l2)) .> 1e-12) .!= 0.0 )
    f3 = findall(diff(abs.(imag(l3)) .> 1e-12) .!= 0.0 )
end
```

[]

```
l1[f1]
```

[0.01]

```
A0v[f2]
```

[0.01]

```
A0v[f3]
```

```
begin
    p1 = scatter(A0v, c=:blue, ms=2, msw = 0, real(l1), label="")
    scatter!(A0v, c=:blue, ms=2, msw = 0, real(l2), label="")
    scatter!(A0v, c=:blue, ms=2, msw = 0, real(l3), label="")
    scatter!(A0v[d1],real(l1[d1]), ms=3, c=:green, label="")
    scatter!(A0v[d2],real(l2[d2]), ms=3, c=:green, label="")
    scatter!(A0v[d3],real(l3[d3]), ms=3, c=:green, label="")
    for d in d1 plot!([A0v[d],A0v[d]],[-1.2,1.5],c=:black,label=""); end
    for d in d2 plot!([A0v[d],A0v[d]],[-1.2,1.5],c=:black,label=""); end
    for d in d3 plot!([A0v[d],A0v[d]],[-1.2,1.5],c=:black,label=""); end
    p2 = scatter(A0v, c=:red, ms=2, msw = 0, imag(l1), label="")
    scatter!(A0v, c=:red,ms=2, msw = 0, imag(l2), label="")
    scatter!(A0v, c=:red,ms=2, msw = 0, imag(l3), label="")
    scatter!(A0v[d1],imag(l1[d1]), ms=3, c=:green, label="")
    scatter!(A0v[d2],imag(l2[d2]), ms=3, c=:green, label="")
    scatter!(A0v[d3],imag(l3[d3]), ms=3, c=:green, label="")
    for d in d1 plot!([A0v[d],A0v[d]],[-1.2,1.5],c=:black,label=""); end
    for d in d2 plot!([A0v[d],A0v[d]],[-1.2,1.5],c=:black,label=""); end
    for d in d3 plot!([A0v[d],A0v[d]],[-1.2,1.5],c=:black,label=""); end
    p3 = scatter(real(l1), c=:blue, ms=2, msw = 0, imag(l1), label="")
    scatter!(real(l2), c=:blue, ms=2, msw = 0, imag(l2), label="")
    scatter!(real(l3), c=:blue, ms=2, msw = 0, imag(l3), label="")
    plot!(cos.(0:pi/100:2*pi),sin.(0:pi/100:2*pi),label="")
    plot(p1,p2,p3,layout=(1,3),size=(1200,300))
end
```

Error message

```
InterruptException:
```

```
begin
    A0_v = -2.5:0.001:0.5
    B0_v = 0.1:-0.001:-2.5
    critical_points = []
    for A0 in A0_v
        for n=1:3
            find_critical!(critical_points2,sol[n],A0_v2,B0;threshold=0.3)
        end
    end
end
```

Error message

> InterruptException:

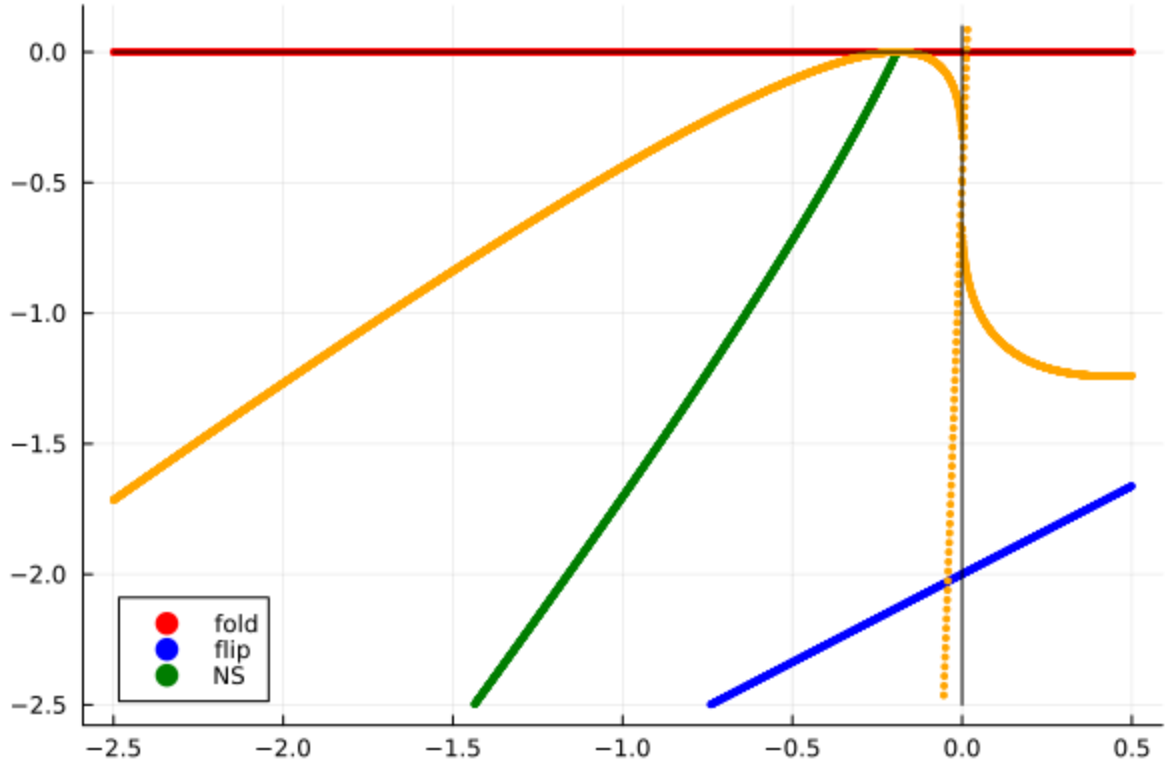
```
begin
    A0_v2 = 0.05:-0.0005:-0.07
    B0_v2 = 0.1:-0.001:-2.5
    critical_points2 = []
    for B0 in B0_v2
        for n=1:3
            find_critical!(critical_points2,sol[n],A0_v2,B0;threshold=0.3)
        end
    end
end
```

```
#jldsave("criticalpoints.jld2";critical_points,A0_v,B0_v)
```

```
cp = JLDFile /home/camilo/AdaptiveTapping/criticalpoints.jld2 (read-only)
     ├─🔢 critical_points
     ├─🔢 A0_v
     └─🔢 B0_v
```

```
cp = jldopen("./criticalpoints.jld2")
```

```
begin
    critical_points = cp["critical_points"]
    A0_v = cp["A0_v"]
    B0_v = cp["B0_v"]
end;
```




```
begin
    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="")
    #ns2 = @. critical_points2[getindex(critical_points2,3) == 3]
    #scatter!(getindex.(ns2,1),getindex.(ns2,2),m=:cross,ms=2,c=:green,label="NS")
    #fc2 = @. critical_points2[getindex(critical_points2,3) == 4]
    #scatter!(getindex.(fc2,1),getindex.(fc2,2),m=:cross,ms=2,c=:orange,label="")
    plot!(A0_v,A0_v*0,c=:black,label="")
    plot!(B0_v*0,B0_v,c=:black,label="")
end
```

```
find_critical! (generic function with 1 method)
```

```
function find_critical!
(critical_points::Vector{Any},sol,A0v::StepRangeLen,B0::Float64;threshold::Float64=0.01)
    lambda = [substitute(sol, Dict([A => A0, B=> B0])) for A0 in A0v];
    dd = findall(diff(abs.(lambda) .> 1) .!= 0.0 .&& abs.(diff(abs.(lambda))) .<
    threshold)
    ff = findall(diff(abs.(imag(lambda)) .> 1e-12) .!= 0.0 )
    for d in dd
        if abs(imag(lambda[d])) < 1e-12
            if real(lambda[d]) > 0
                # fold
                push!(critical_points,[A0v[d],B0,1])
            else
                #flip
                push!(critical_points,[A0v[d],B0,2])
            end
        else
            # Neimark Sacker
            push!(critical_points,[A0v[d],B0,3])
        end
    end
    for f in ff
        push!(critical_points,[A0v[f],B0,4])
    end
    return nothing
end
```

```
find_critical! (generic function with 2 methods)
```

```
function find_critical!
(critical_points::Vector{Any},sol,A0::Float64,B0v::StepRangeLen;threshold::Float64=0.01)
    lambda = [substitute(sol, Dict([A => A0, B=> B0])) for B0 in B0v];
    dd = findall(diff(abs.(lambda) .> 1) .!= 0.0 .&& abs.(diff(abs.(lambda))) .<
    threshold)
    ff = findall(diff(abs.(imag(lambda)) .> 1e-12) .!= 0.0 )
    for d in dd
        if abs(imag(lambda[d])) < 1e-12
            if real(lambda[d]) > 0
                # fold
                push!(critical_points,[A0,B0v[d],1])
            else
                #flip
                push!(critical_points,[A0,B0v[d],2])
            end
        else
            # Neimark Sacker
            push!(critical_points,[A0,B0v[d],3])
        end
    end
    for f in ff
        push!(critical_points,[A0,B0v[f],4])
    end
    return nothing
end
```