Investment problem

$$\max_{i(t)} \int_0^T e^{-\rho t}\left\{pk(t) - ci(t) - \frac{d}{2}i(t)^2\right\}dt \qquad st \qquad \dot{k}(t) = i(t) - \delta k(t)$$

given boundary conditions

$$k(0) = k_0 \qquad k(T) = k_T \qquad T: \text{ given}$$

Necessary conditions

$$\dot{k} = i - \delta k$$
$$\dot{\psi} = (\rho + \delta)\psi - p$$

```
# can take a while....
# might need to install DifferentialEquations & Plots with
# ]add Differential Equations Plots
begin
    using DifferentialEquations
    using Plots
    using Base.Iterators: product
end
```

```
# tell plots to use GR b/c it's fastest
gr();
```

```
23.33333333333332
```

```
begin
    # parameters
    px=0.5
    c=3
    d=1
    ρ=0.07
    δ=0.05

    # steady state
    ssi = (px / (ρ+δ) - c) / d
    ssk = ssi/δ
end
```

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

```
begin
    doti(k,i) = (ρ+δ)*(i - ssi)
    dotk(k,i) = i - δ*k

    # solvers in DifferentialEquations.jl require functions that
```

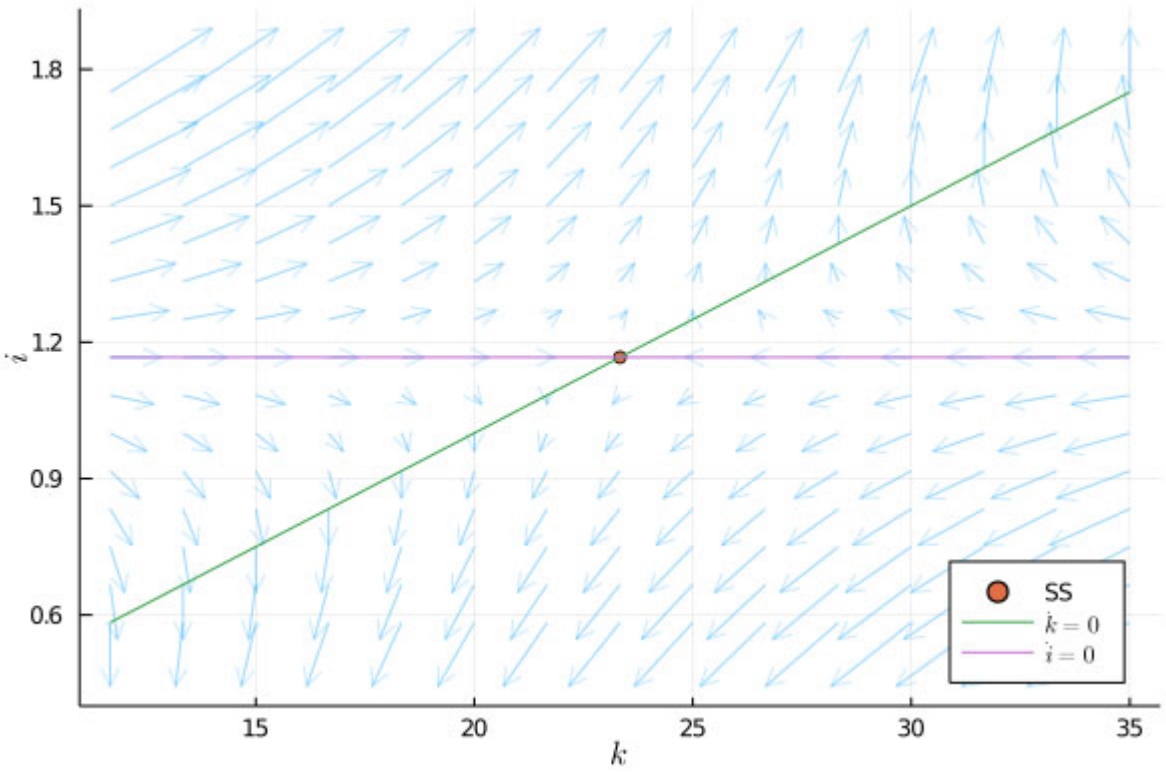
```
    # take the form   'f!(dy, y, params, t)'
    function dot_ki!(dy, y, parm, t)
        k,i = y
        dy[1] = dotk(k,i)
        dy[2] = doti(k,i)
        return dy
    end
end
```

```
15×15 Array{Float64,2}:
  0.0        0.166667      0.333333   0.5        …   2.16667       2.33333
 -0.166667  -2.22045e-16   0.166667   0.333333       2.0           2.16667
 -0.333333  -0.166667      0.0        0.166667       1.83333       2.0
 -0.5       -0.333333     -0.166667   0.0            1.66667       1.83333
 -0.666667  -0.5          -0.333333  -0.166667       1.5           1.66667
 -0.833333  -0.666667     -0.5       -0.333333   …   1.33333       1.5
 -1.0       -0.833333     -0.666667  -0.5            1.16667       1.33333
  ⋮                                                 ⋱
 -1.5       -1.33333      -1.16667   -1.0            0.666667      0.833333
 -1.66667   -1.5          -1.33333   -1.16667    …   0.5           0.666667
 -1.83333   -1.66667      -1.5       -1.33333        0.333333      0.5
 -2.0       -1.83333      -1.66667   -1.5            0.166667      0.333333
 -2.16667   -2.0          -1.83333   -1.66667       -4.44089e-16   0.166667
 -2.33333   -2.16667      -2.0       -1.83333       -0.166667     -4.44089e-16
```

```
# create a "mesh"
begin
    # grid of points in K, I space
    kspace = range(ssk/2, stop = 1.5*ssk, length=15)
    ispace = range(ssi/2, stop = 1.5*ssi, length=15)

    kispace = product(kspace, ispace)
    KK = [k for (k,i) in kispace]
    II = [i for (k,i) in kispace]

    SCALE = 2
    dotII = [doti(k,i) for (k,i) in kispace].*SCALE
    dotKK = [dotk(k,i) for (k,i) in kispace].*SCALE
end
```




```
begin
    plt_quiv = plot(;legend=:bottomright, xlabel="\$k\$", ylabel="\$i\$")

    # motion vectors
    # note, use 'vec' b/c need to input a vector
    quiver!(plt_quiv, vec(KK), vec(II); quiver=(vec(dotKK),vec(dotII)), alpha=0.2)

    # steady state
    # also note that we have to give vectors to scatter
    scatter!(plt_quiv, [ssk],[ssi], label="SS")

    # nullclines
    plot!(plt_quiv, kspace, k -> k*δ, label="\$\\dot k = 0\$")
    plot!(plt_quiv, kspace, k -> ssi, label="\$\\dot i = 0\$")
end
```

```
[36mODEProblem[0m with uType [36mArray{Float64,1}[0m and tType [36mFloat64[0m. In-pla
timespan: (0.0, 25.0)
u0: [12.0, 1.19]
```

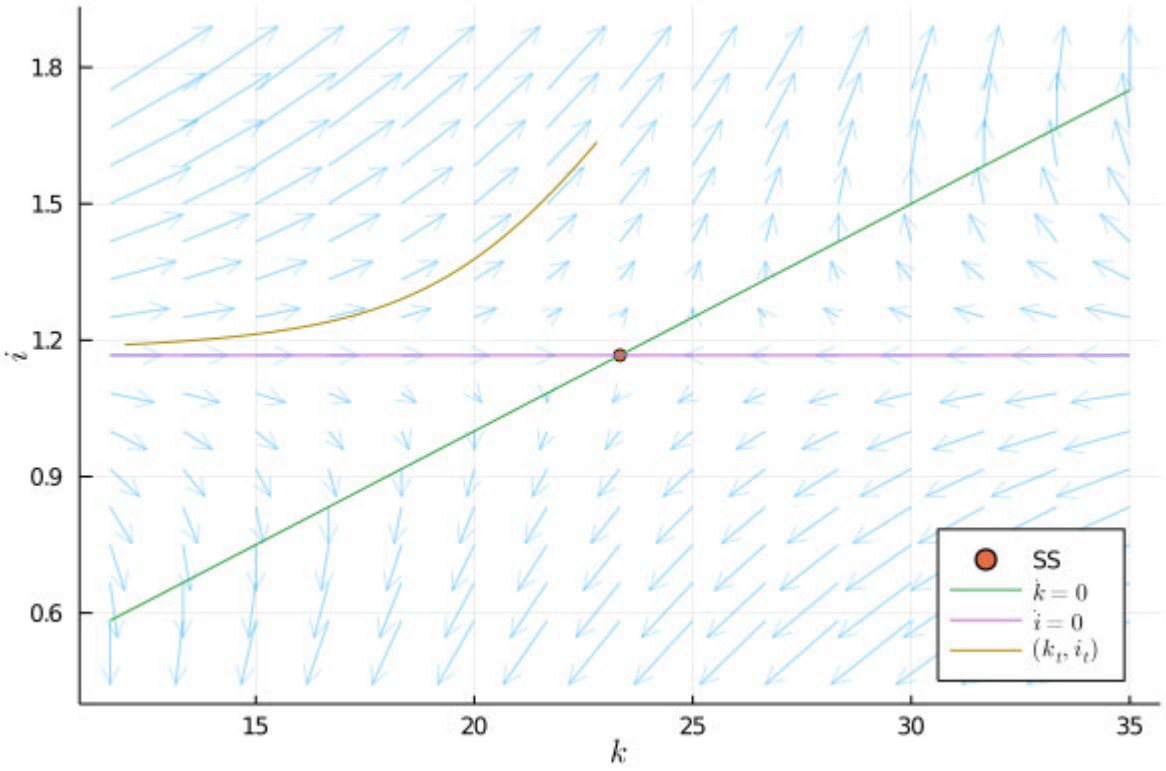
```
begin
    # initial conditions
    # (k₀ i₀)
    y0 = [12.0, 1.19]

    # time-span for problem
    # NOTE: this MUST be in Floats, not Ints!
    tmax = 25.0
    tspan = (0.0, tmax)

    # define a problem
    prob = ODEProblem(dot_ki!, y0, tspan);
end
```

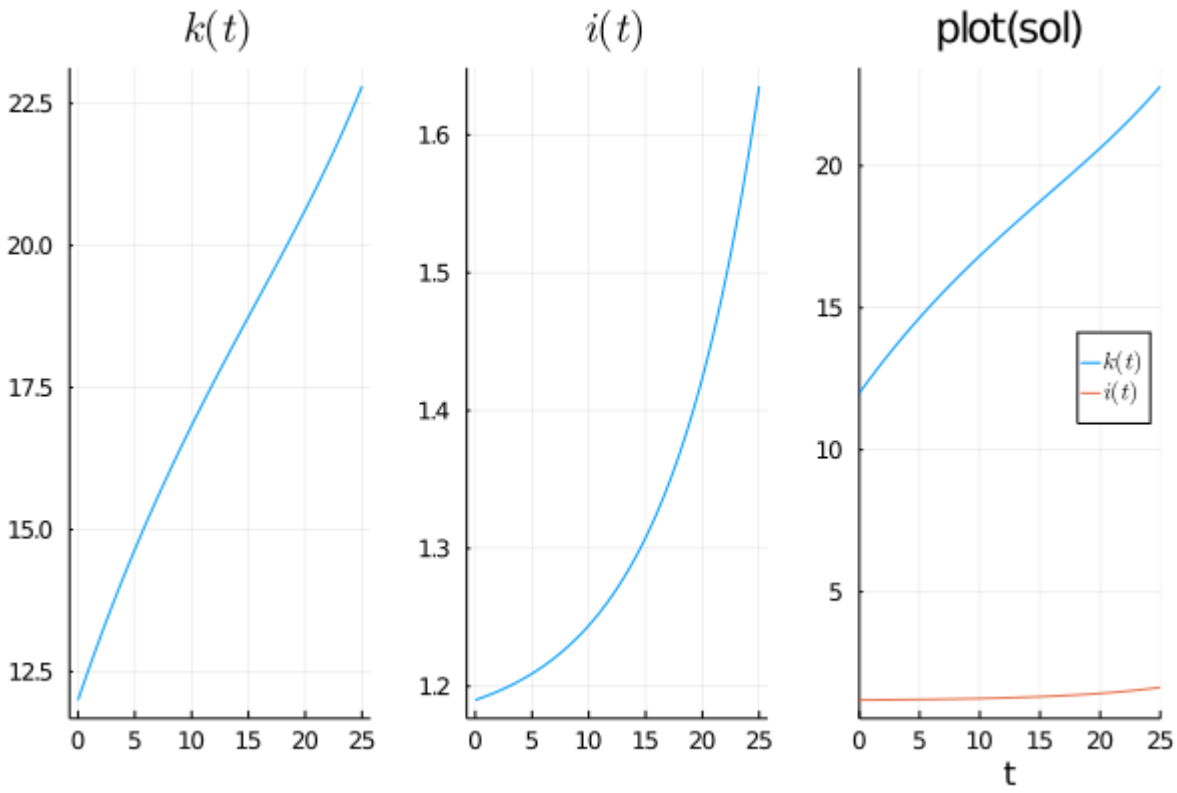
```
sol = retcode: Success
      Interpolation: specialized 4th order "free" interpolation
      t: 7-element Array{Float64,1}:
        0.0
        0.19573878328137712
        2.153126616095148
        6.6929381211746595
       12.54812413836499
       19.311471520641916
       25.0
      u: 7-element Array{Array{Float64,1},1}:
       [12.0, 1.19]
       [12.11497648059986, 1.1905545559837427]
       [13.211198665368997, 1.196879239196716]
       [15.431517391727429, 1.2187598669073956]
       [17.82701675624822, 1.2718442542485149]
       [20.358703731436886, 1.4034711175727574]
       [22.803708674093624, 1.6353150483207328]
```

```
# solve the problem using the 'Tsit5()' method
# Note how much better our tolerances are
sol = solve(prob, Tsit5())
```




```
begin
    TT = 0 : 0.1 : tmax
    kpath = first.(sol.(TT))
    ipath = last.(sol.(TT))

    # need to copy plot b/c of updating in Pluto
    pltq = deepcopy(plt_quiv)
    plot!(pltq, kpath, ipath; label="\$(k_t, i_t)\$")
end
```




```
plot(
    plot(TT, t -> sol(t)[1], title="\$k(t)\$", legend=:none),
    plot(TT, t -> sol(t)[2], title="\$i(t)\$", legend=:none),
    plot(
        sol,
        labels=["\$k(t)\$" "\$i(t)\$"], # NOTE: labels are row matrix!
        legend=:right,
        title="plot(sol)"
    ),
    layout=(1,3)
)
```