ToyExample

February 3, 2020

1 Model

1.1 Oil producer's problem

$$\pi = \max \begin{cases} \overbrace{R_o + R_g}^{R_t} - c_g & \text{gather} \\ R_o - (c_f + \tau) & \text{flare + tax} \\ V_a & \text{alternative} \end{cases}$$

We then have

$$Flare \succsim Gather \iff (c_f + \tau) \geq c_g - R_g$$

$$Alt \succsim Gather \iff V_a \geq R_o + R_g - c_g$$

$$Alt \succsim Flare \iff V_a \geq R_o - (c_f + \tau)$$

Define indifference thresholds

$$c_f^*(c_g) = c_g - R_g - \tau \qquad V_a^*(c_g) = R_o + R_g - c_g$$

Derivatives wrt c_g, τ are

$$\frac{dc_f^*}{dc_g} = 1 \qquad \frac{dc_f^*}{d\tau} = -1 \qquad \frac{dV_a^*}{dc_g} = -1 \qquad \frac{dV_a^*}{d\tau} = 0$$

Assume that flaring costs & outside options are uniformly distributed w/ mass equal to upper limit: $c_f \sim \bar{c}_f \times U(0, \bar{c}_f)$ and $V_a \sim \times \bar{V}_a U(0, \bar{V}_a)$. Then quantities outcomes and derivatives wrt c_g are the follwoing:

Gathering -

$$G = V_a^*(\bar{c}_f - c_f^*) \qquad G' = -[V_a^* + (\bar{c}_f - c_f^*)]$$

Flaring -

$$F = c_f^*[V_a^* + \frac{1}{2}c_f^*] \qquad F' = V_a^*$$

Alternative/Outside Option

$$A = \bar{c}_f(\bar{V}_a - V_a^*) - \frac{1}{2}(c_f^*)^2 \qquad A' = (\bar{c}_f - c_f^*)$$

1.2 Processor profits

$$W = c_g G - \kappa(G)$$

In the competitive case we have

$$c_g = \kappa'(G)$$

But in a monopolistic case, FOC imply a markup

$$c_g = \kappa'(G) - \underbrace{\frac{G}{G'}}_{<0} > \kappa'(G)$$

Under a constant MC, $\kappa(G) = \kappa_0 G$, we end up with a quadratic function of c_g^*, and then we solve for the lower quadratic root. The algebra is awful to sign $\frac{dc_g}{d\tau}$ under monopolistic competition… so I do it numerically.

1.3 Analysis

Monopolistic behavior results in a MUCH bigger markup on processing, and more flaring. When we impose a tax, the monopolist actually *increases* the price of its services.

The "outside option" V^a will include other processors… so we can think about this model as a case of a Nash-Bertrand equilibrium a price-setting game with monopolistic competition (as one would do in BLP…). Practically speaking, we could think about a processor on a network who knows they face a downward-sloping demand curve because the wells close to them are unlikely to switch to an alternative, far-away processor if the processor raises the price a bit.

2 Installaion of package

Install `ToyFlaringModel.jl` in Julia, and then "instantiate" it to install dependencies Note that the `]` will put Julia into "package" mode, so you won't need the second `]`

```
]dev https://github.com/magerton/ToyFlaringModel.jl
]instantiate ToyFlaringModel
```

3 Installing Jupyter notebooks

To run this notebook, you'll need to install IJulia (see https://github.com/JuliaLang/IJulia.jl).

Once you install Julia, you'll need to install IJulia with `]add IJulia`. Then open up the Julia command line and ask for a Jupyter notebook

```
using IJulia
jupyterlab()
```

You can also create a shortcut on Windows. For example, my shortcut `Target` is `D:\libraries\julia\conda\3\Scripts\jupyter-lab.exe`, and the `Start in` location is `E:\projects\`

[1]:
```julia
using ToyFlaringModel
using Plots
using ForwardDiff
```

```
gr(fmt=:png)
using Plots: pdf
```

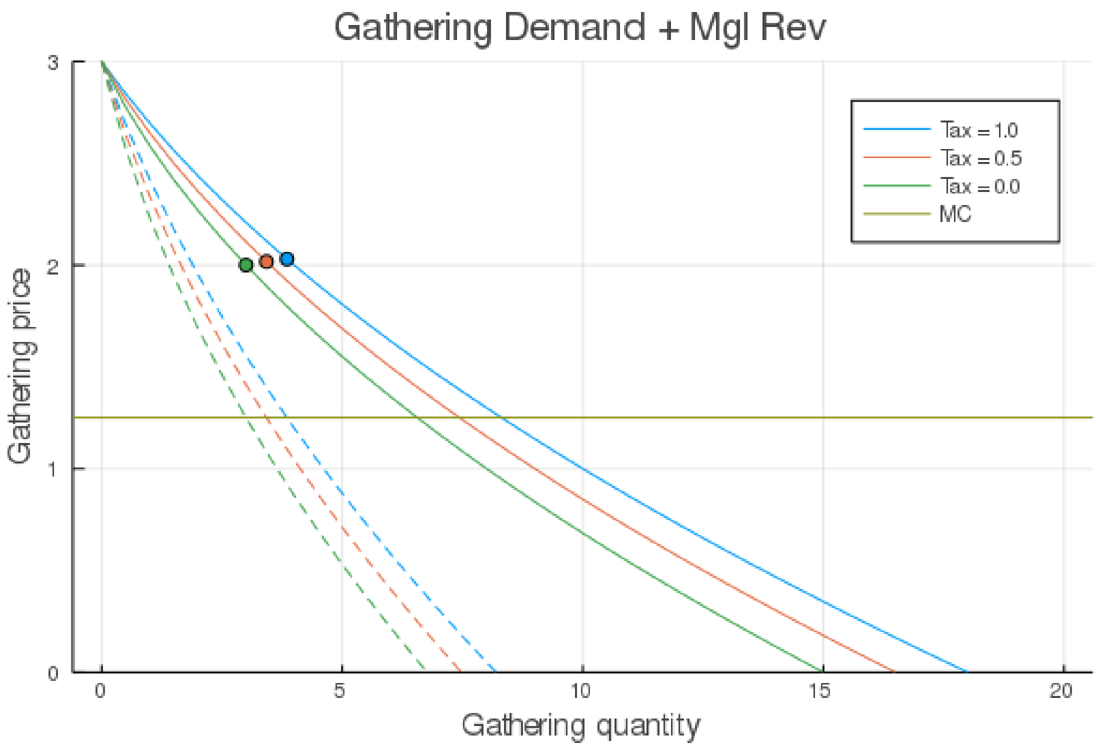
```
 Info: Recompiling stale cache file
D:\software_libraries\julia\compiled\v1.2\ToyFlaringModel\l2jD5.ji for
ToyFlaringModel [8a476941-f787-44af-bcc4-be9956bd24c8]
 @ Base loading.jl:1240
```

[2]:
```
mc = FlaringModel(;monop=false)
mm = FlaringModel(;monop=true)
```

[2]:
```
FlaringModel(2.0, 1.0, 4.0, 4.0, 1.25, true)
```

[3]:
```
using ToyFlaringModel: Ro, Rt, Rg, cbar

# is negative root of quadratic
function inversedemand(m,g,t)
    B = -(Rt(m) + Rg(m) + t + cbar(m))
    C = Rt(m)*(Rg(m)+t+cbar(m)) - g
    x = (- B - sqrt(B^2 - 4*C))/2
    return x
end

inversedemandp(m,g,t) = ForwardDiff.derivative(x -> inversedemand(m,x,t), g)

mr(m,g,t) = inversedemandp(m,g,t)*g + inversedemand(m,g,t)

qs = 0:0.5:20
taxes = [1, 1/2, 0]

p = plot(;title="Gathering Demand + Mgl Rev", ylab="Gathering price", xlab =␣
 ↪"Gathering quantity", ylim=(0,Inf))
for (i,t) in enumerate(taxes)
    plot!(p, qs, q -> inversedemand(mm,q,t), label="Tax = $t", linecolor=i)
    plot!(p, qs, q -> mr(mm,q,t),            label="",         linecolor=i,␣
 ↪linestyle=:dash)
    scatter!(p,[Gather(mm,t)], [Cost(mm,t)], label="",         color=i )
end
hline!(p,[mm.k0], label="MC")
png(p,"gathering-demand.png")
pdf(p,"gathering-demand.pdf")
plot(p)
```

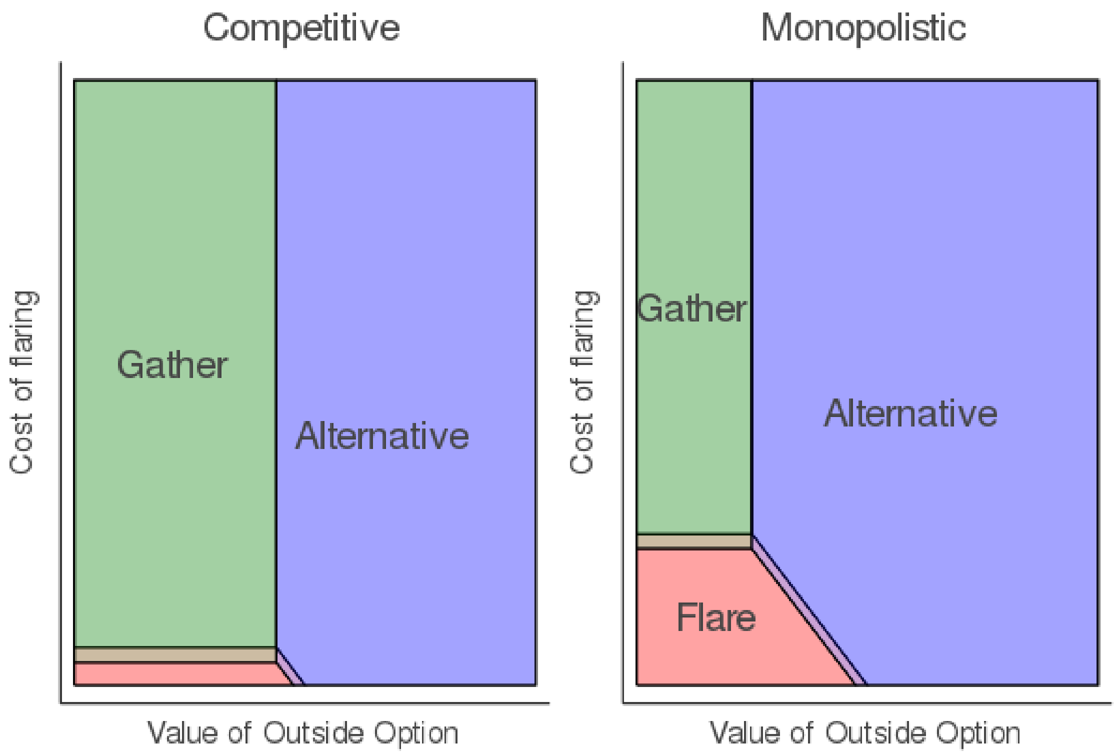
[3]:



[4]:
```
function PlotModel(m::FlaringModel, t0=0, t1=0.1)
    p = plot(; xticks=0, yticks=0, xlab="Value of Outside Option", ylab="Cost␣
↪of flaring")
    plot!(p, GatherShape(m,t0); fillalpha=0.2, label="", fillcolor=:green)
    plot!(p, FlareShape( m,t0); fillalpha=0.2, label="", fillcolor=:red)
    plot!(p, AltOptShape(m,t0); fillalpha=0.2, label="", fillcolor=:blue)

    plot!(p, GatherShape(m,t1), fillalpha=0.2, label="", fillcolor=:green)
    plot!(p, FlareShape( m,t1), fillalpha=0.2, label="", fillcolor=:red)
    plot!(p, AltOptShape(m,t1), fillalpha=0.2, label="", fillcolor=:blue)

    annotate!([GatherLab(m,t0)])
    annotate!([AltOptLab(m,t0)])

    return p
end

p1 = PlotModel(mc, 0, 0.1)
title!(p1, "Competitive")
p2 = PlotModel(mm, 0, 0.1)
title!(p2, "Monopolistic")
annotate!(p2, [FlareLab(mm,0.1)])
p3 = plot(p1,p2)
```

```
png(p3,"competitive-vs-monopolistic.png")
pdf(p3,"competitive-vs-monopolistic.pdf")
plot(p3)
```

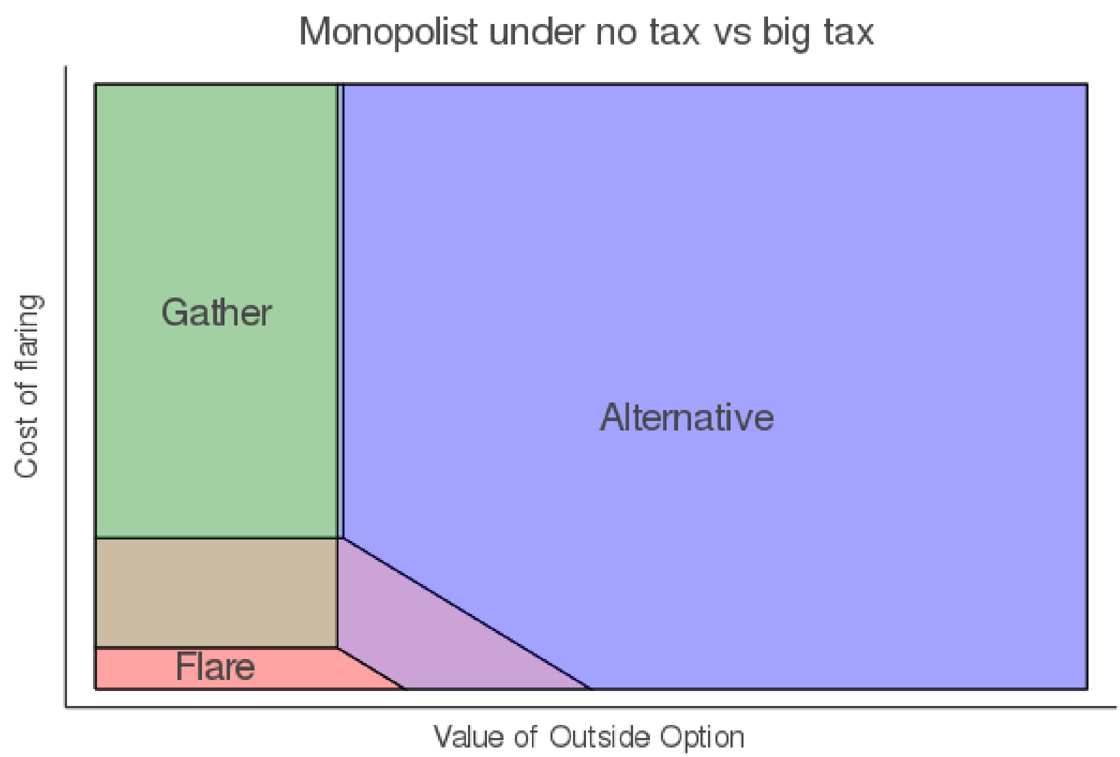
[4]:



[5]:

```
p = PlotModel(mm, 0, 0.75)
title!(p, "Monopolist under no tax vs big tax")
annotate!(p, [(0.5, 0.15, "Flare")])
png(p, "monopolist-with-tax.png")
pdf(p, "monopolist-with-tax.pdf")
plot!(p)
```

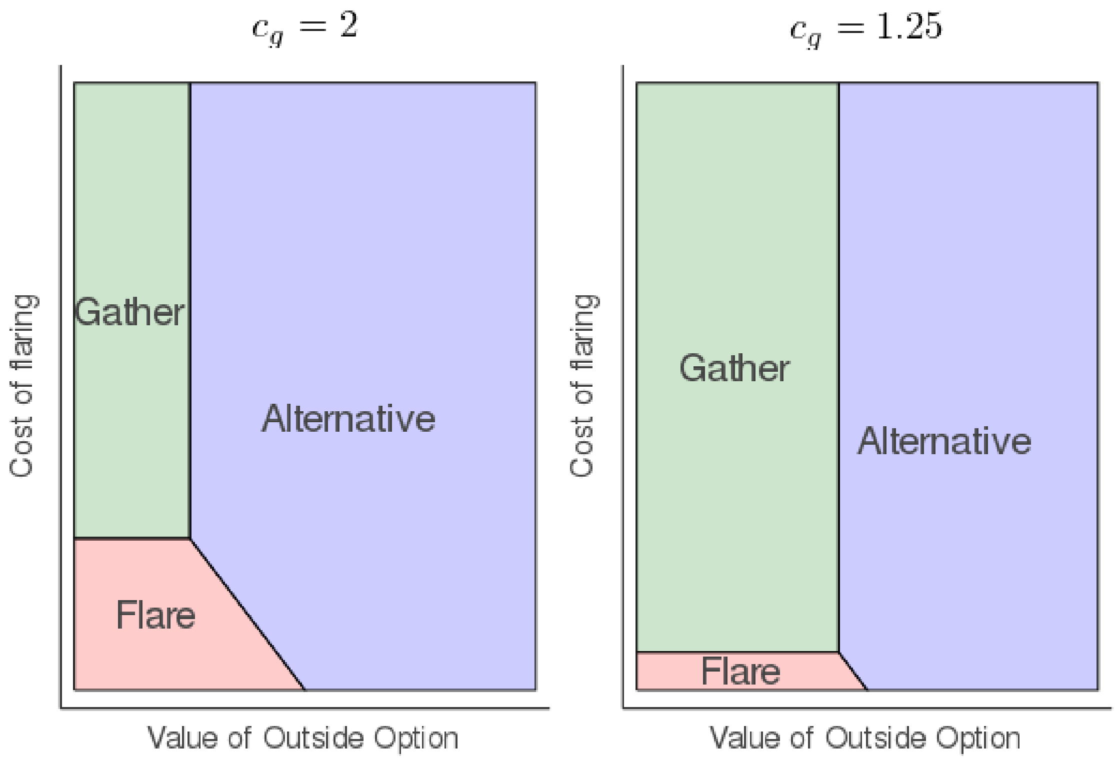
[5]:



[6]:
```
c0 = 2
c1 = 1.25

gs(c) = GatherShape(mc,c,0)
fs(c) = FlareShape(mc,c,0)
as(c) = AltOptShape(mc,c,0)

p = plot(; title="\$c_g = 2\$",   xticks=0, yticks=0, xlab="Value of Outside␣
↪Option", ylab="Cost of flaring")
q = plot(; title="\$c_g = 1.25\$", xticks=0, yticks=0, xlab="Value of Outside␣
↪Option", ylab="Cost of flaring")

plot!(p, gs(c0); fillalpha=0.2, label="", fillcolor=:green)
plot!(p, fs(c0); fillalpha=0.2, label="", fillcolor=:red)
plot!(p, as(c0); fillalpha=0.2, label="", fillcolor=:blue)
annotate!(p,[GatherLab(mc,c0,0)])
annotate!(p,[AltOptLab(mc,c0,0)])
annotate!(p,[FlareLab( mc,c0,0)])

plot!(q, gs(c1); fillalpha=0.2, label="", fillcolor=:green)
plot!(q, fs(c1); fillalpha=0.2, label="", fillcolor=:red)
plot!(q, as(c1); fillalpha=0.2, label="", fillcolor=:blue)
annotate!(q,[GatherLab(mc,c1,0)])
```

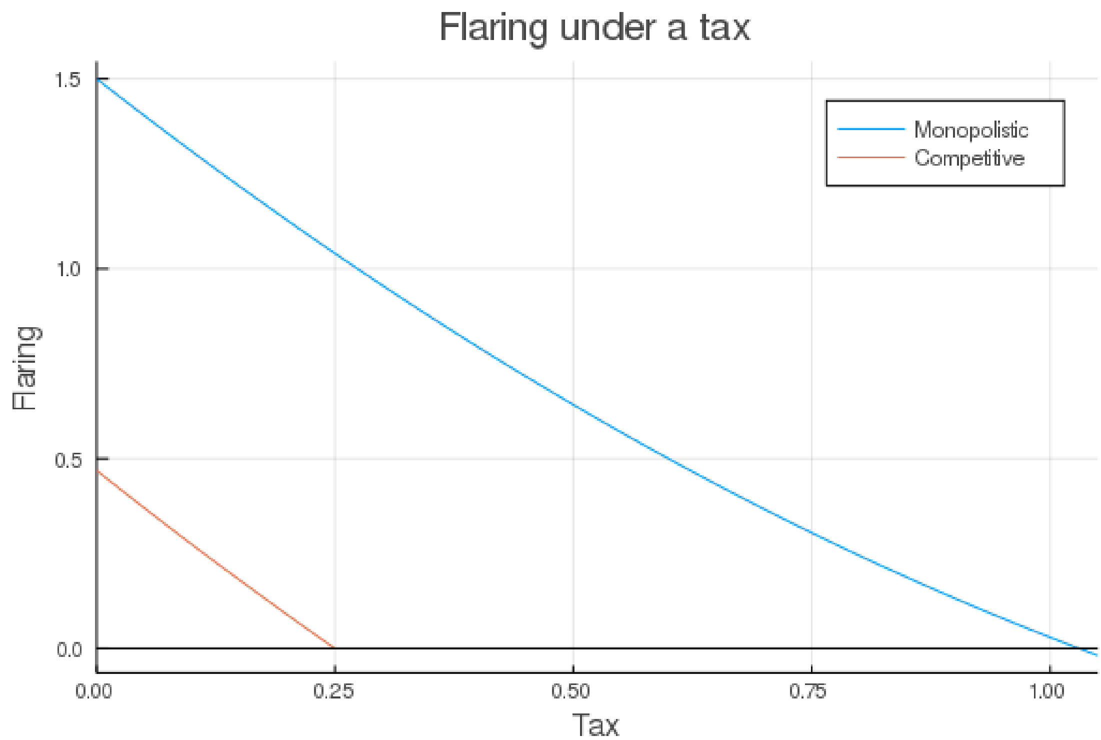
```
annotate!(q,[AltOptLab(mc,c1,0)])
annotate!(q,[FlareLab( mc,c1,0)])

pp = plot(p,q)
png(pp, "gather-flare-alt-under-different-costs.png")
pdf(pp, "gather-flare-alt-under-different-costs.pdf")
plot(pp)
```

[6]:



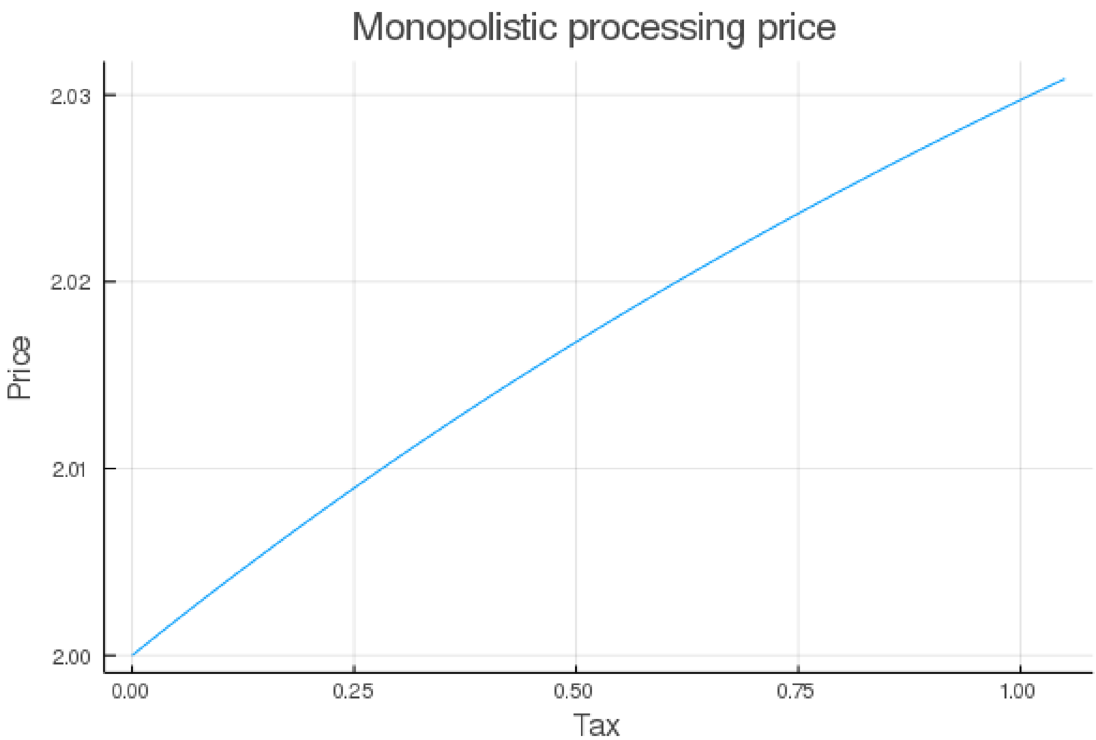
[7]:
```
taxes = 0:0.01:1
p = plot(; title="Flaring under a tax", xlab="Tax", ylab="Flaring",␣
 ↪xlim=(0,Inf))
plot!(p, 0:0.01:1.05, t -> Flare(mm, t), label="Monopolistic")
plot!(p, 0:0.01:0.25, t -> Flare(mc, t), label="Competitive")
hline!(p, [0]; linecolor=:black, label="")
png(p, "flaring-under-tax.png")
pdf(p, "flaring-under-tax.pdf")
plot!(p)
```

[7]:



[8]:
```
p = plot(; title="Monopolistic processing price", xlab="Tax", ylab="Price")
plot!(p, 0:0.01:1.05, t -> Cost(mm, t), label="")
png(p, "monopolist-overshifts-tax.png")
pdf(p, "monopolist-overshifts-tax.pdf")
plot(p)
```

[8]:


Monopolistic processing price
Price
Tax
2.03
2.02
2.01
2.00
0.00
0.25
0.50
0.75
1.00