A much-too-simplified simulation of the Chamley-Judd result (and demonstration that it breaks without constant returns to scale)

Basic setup

First let’s set up our very quick-and dirty framework. We begin by computing the marginal products of capital and labor for an unspecified production function F:

```
In[2]:= mpk[K_, L_, F_] := (∂KAP F[KAP, LAB]) /. {KAP → K, LAB → L}
```

```
In[3]:= mpl[K_, L_, F_] := (∂LAB F[KAP, LAB]) /. {KAP → K, LAB → L}
```

Since factors will be paid their marginal products, we can compute the total take accruing to both factors:

```
In[4]:= totalCapIncome[K_, L_, F_] := K mpk[K, L, F]
```

```
In[5]:= totalWages[K_, L_, F_] := L mpl[K, L, F]
```

In the absence of taxes, we compute the equilibrium quantity of capital. We assume a constant time preference *r*. At equilibrium, the rate of time preference must equal the marginal rate of return to capital:

```
In[6]:= eqmK[L_, r_, F_] := K /. Solve[r == mpk[K, L, F], K][[1]]
```

Now we compute the quantity of capital that would obtain at equilibrium if a tax rate τ is imposed on capital. The equilibrium condition is that the *after-tax* rate of return, $(1 - \tau)$ times the marginal prodict of capital, must equal the time preference *r*:

```
In[7]:= taxedEqmK[L_, r_, τ_, F_] := K /. Solve[r == (1 - τ) mpk[K, L, F], K][[1]]
```

We assume that all taxes raised will be redistributed to workers, providers of *L*, who for the purposes of this exercise are presumed to be entirely distinct from providers of *K*. So, we compute the quantity of redistributed wealth, which is just the quantity of capital at the taxed equilibrium, times the equilibrium marginal product of capital $\frac{r}{(1-\tau)}$ (which gives us pre-tax income), times the tax rate on capital τ:

```
In[8]:= redistributedK[L_, r_, τ_, F_] := taxedEqmK[L, r, τ, F] r/(1 - τ) τ
```

Labor wealth at equilibrium is total wages at equilibrium plus taxes levied against capital providers, if any. We compute both cases, with taxation and redistribution and without:

```
In[9]:= laborWealthWithRedistribution[L_, r_, τ_, F_] :=
         totalWages[taxedEqmK[L, r, τ, F], L, F] + redistributedK[L, r, τ, F]
```

In[10]:=
```
laborWealthNoRedistribution[L_, r_, F_] := totalWages[eqmK[L, r, F], L, F]
```

Finally, we will want to plot the gain to workers from redistributive taxation. Positive values mean workers are better off with redistribution, negative values mean they are made worse off. The tax rate τ will be the indepenent variable, which we will examine in the neighborhood of zero:

In[32]:=
```
redistributiveGainPlot[L_, r_, F_, radius_] :=
 Plot[{(laborWealthWithRedistribution[L, r, τ, F] -
      laborWealthNoRedistribution[L, r, F]), 0},
  {τ, -radius, +radius}, Frame → True, AspectRatio → 1, PlotStyle → {Blue, Red}]
```

"Classic" simulation, constant returns to scale

In the spirit of Judd (1985), we normalize inelastic labor provision to the quantity 1.

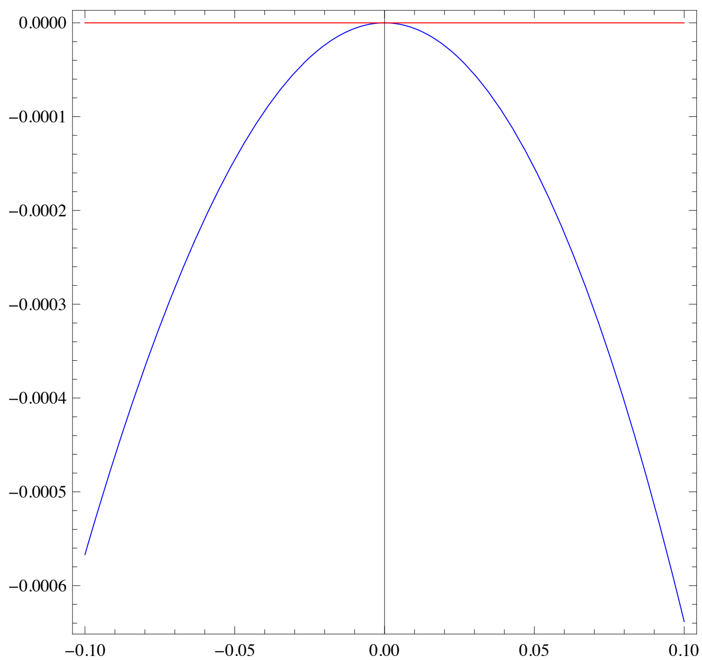
Let's plot under a Cobb-Douglas production function, with three different exponents. All of these are constant-returns-to-scale production functions:

In[19]:=
```
Off[Solve::ifun] (* Suppress warnings about use of inverse functions *)
```

In[20]:=
```
cobbDouglas[K_, L_, α_] := K^α L^(1-α)
```

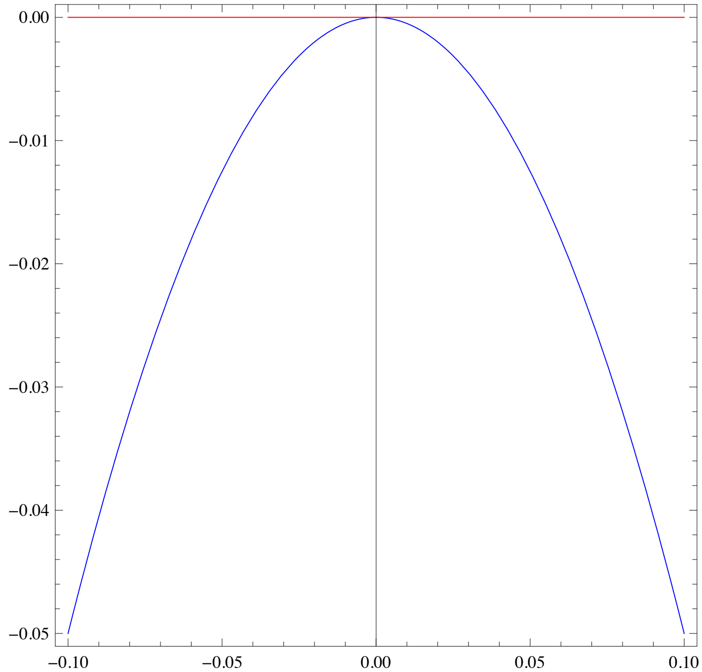
In[33]:=
```
F1[K_, L_] := cobbDouglas[K, L, 0.1]; redistributiveGainPlot[1, 0.05, F1, 0.1]
```

Out[33]=



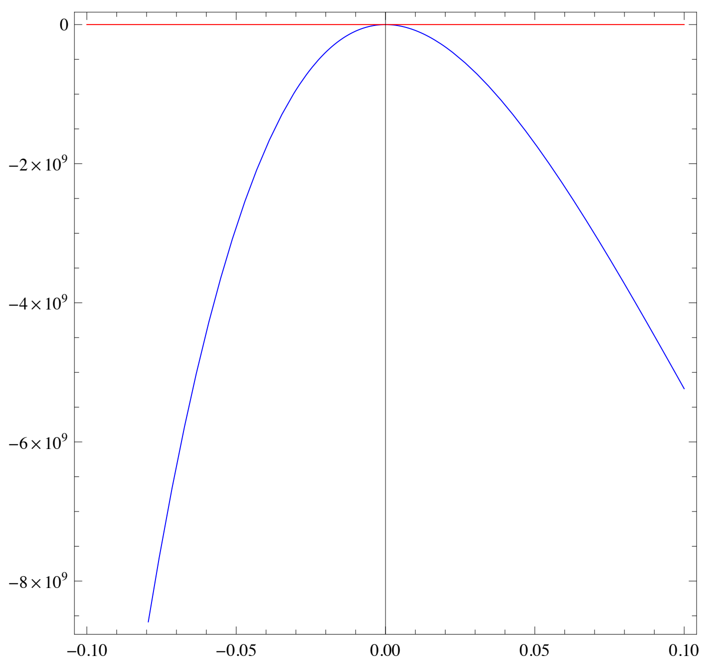
In[34]:=
```
F2[K_, L_] := cobbDouglas[K, L, 0.5]; redistributiveGainPlot[1, 0.05, F2, 0.1]
```

Out[34]=



In[35]:=
```
F3[K_, L_] := cobbDouglas[K, L, 0.9]; redistributiveGainPlot[1, 0.05, F3, 0.1]
```

Out[35]=



Note that in all cases, the redistributive gain is always less than or equal to zero, and is maximized at zero when the tax rate τ is zero. This is the classic Chamley-Judd result. ■

a quick double-check...

Let's verify analytically, as well as graphically, that the optimal tax rate on capital is precisely zero:

In[75]:=
```
wealthFunction =
 FullSimplify[(laborWealthWithRedistribution[L, r, τ, F] - laborWealthNoRedistribution[L, r, F]) /.
   F → (cobbDouglas[#1, #2, α] &)]
```

Out[75]=
$$-\frac{r\,\tau\left(\frac{L^{-1+\alpha}\,r}{\alpha-\alpha\,\tau}\right)^{\frac{1}{-1+\alpha}}}{-1+\tau}+L^{1-\alpha}\,(-1+\alpha)\left(\left(\left(\frac{L^{-1+\alpha}\,r}{\alpha}\right)^{\frac{1}{-1+\alpha}}\right)^{\alpha}-\left(\left(\frac{L^{-1+\alpha}\,r}{\alpha-\alpha\,\tau}\right)^{\frac{1}{-1+\alpha}}\right)^{\alpha}\right)$$

In[76]:=
```
derivWealthFunction = FullSimplify[∂τ wealthFunction ]
```

Out[76]=
$$\frac{L^{1-\alpha}\,\alpha\,(-1+\tau)\left(\left(\frac{L^{-1+\alpha}\,r}{\alpha-\alpha\,\tau}\right)^{\frac{1}{-1+\alpha}}\right)^{\alpha}+\frac{r\,(-1+\alpha+\tau)\left(\frac{L^{-1+\alpha}\,r}{\alpha-\alpha\,\tau}\right)^{\frac{1}{-1+\alpha}}}{-1+\alpha}}{(-1+\tau)^{2}}$$

In[77]:=
```
Simplify[derivWealthFunction, r > 0 && α > 0 && α < 1 && τ == 0]
```

Out[77]= 0

"Noveau" simulation, non-constant returns to scale

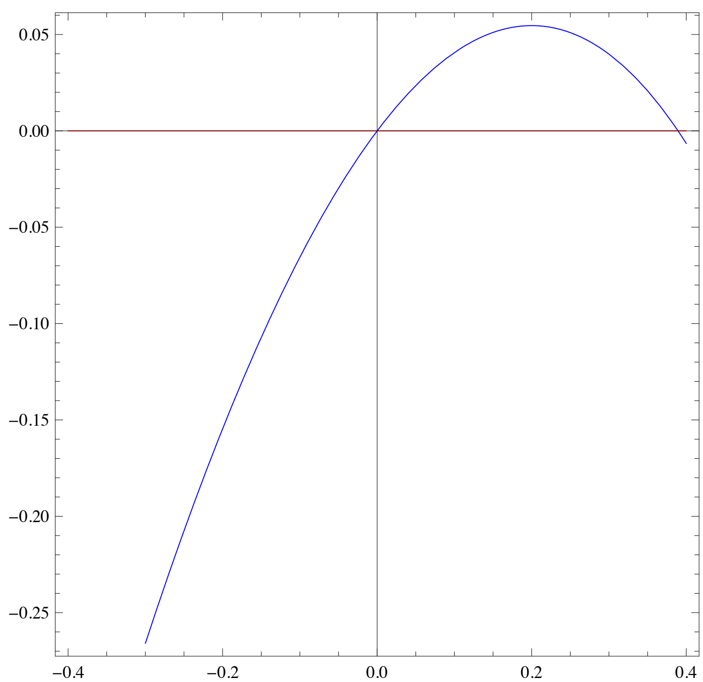
We repeat the previous exercise, but with the following production function:

In[36]:=
```
noveau[K_, L_, α_] := (K L)^α
```

Now when the exponent is less that 0.5, we have a production function with decreasing returns to scale. When the exponent is greater than 0.5, returns to scale are increasing. And, when the exponent is precisely 0.5, we have repeated the Cobb-Douglas, constant-returns-to-scale case:

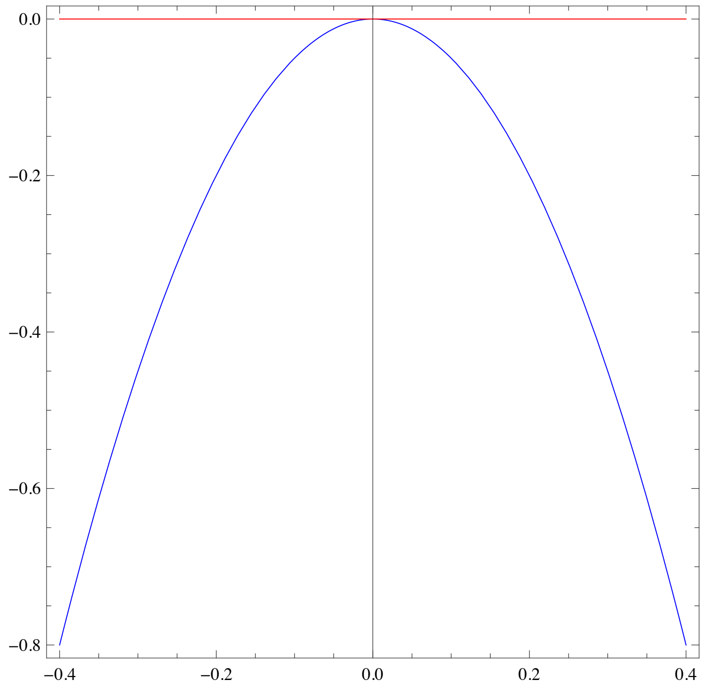
In[41]:=
```
F4[K_, L_] := noveau[K, L, 0.4]; redistributiveGainPlot[1, 0.05, F4, 0.4]
```

Out[41]=



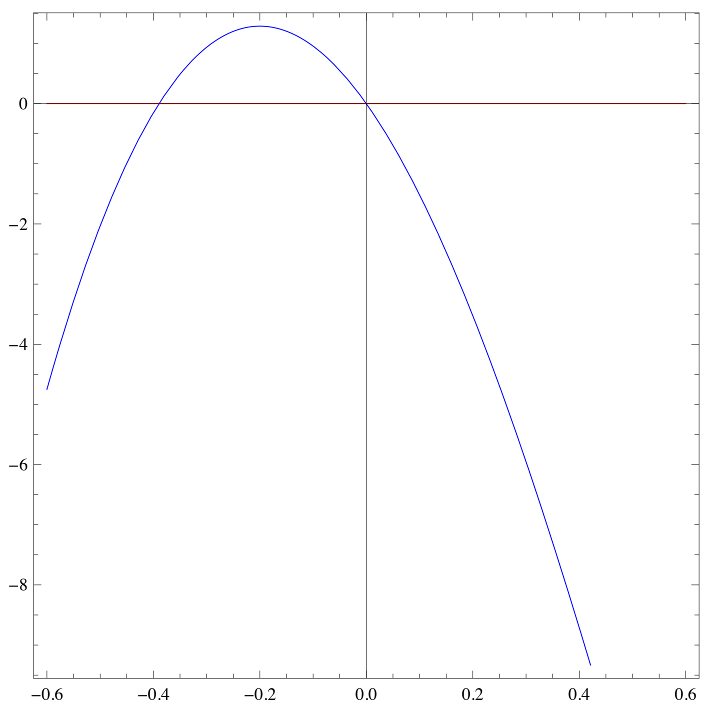
In[42]:=
```
F5[K_, L_] := noveau[K, L, 0.5]; redistributiveGainPlot[1, 0.05, F5, 0.4]
```

Out[42]=



In[43]:=
```
F6[K_, L_] := noveau[K, L, 0.6]; redistributiveGainPlot[1, 0.05, F6, 0.6]
```

Out[43]=



Now the optimal tax on capital (from workers' perspective) depends on the production function. With modestly decreasing returns to scale, the optimal tax rate is positive; with constant returns to scale the optimal tax rate is zero; with modestly increasing returns to scale, the optimal tax rate is negative.

You can get these relationships to change if you explore more radical deviations from constant returns-to-scale. ■