VFI example

This notebook shows an example of value function iterations.

```
In[138]:= x = 0; Remove["Global`*"]; DateList[Date[]] // Most

Out[138]= {2020, 4, 27, 17, 21}
```

Discrete-Time Growth Example

Assume that the production function is Cobb-Douglas (A is chosen so that steady state is k=1)

In[139]:=
```
Clear[f, ftay]; f0[k_] = A k^α; A = 1/(α β);
α = 0.25; β = 0.90;
ftay[k_] = Series[f0[k], {k, 0.01, 2}] // Normal;
f[k_] = If[k ≤ 0.01, ftay[k] // Evaluate, f0[k]]
```

Out[142]=
```
If[k ≤ 0.01, 1.40546 + 35.1364 (-0.01 + k) - 1317.62 (-0.01 + k)^2, f0[k]]
```

and that the utility function is the log function

In[143]:=
```
u0[c_] = Log[c];
utay[c_] = Series[u0[c], {c, 0.01, 2}] // Normal;
u[c_] = If[c ≤ 0.01, utay[c] // Evaluate, u0[c]]
```

Out[145]=
```
If[c ≤ 0.01, -4.60517 + 100. (-0.01 + c) - 5000. (-0.01 + c)^2, u0[c]]
```

Closed-form solutions for value function and consumption functions

In[146]:= $\text{Vtrue}[k\_] = -\frac{\alpha\,\text{Log}[k]}{-1+\alpha\beta} - \frac{\text{Log}\left[\frac{1-\alpha\beta}{\alpha\beta}\right]}{-1+\beta}$

Out[146]= 12.3676 + 0.322581 Log[k]

In[147]:= $\theta = 1 - \alpha\beta$; Ctrue[k_] = θ f0[k]

Out[147]= $3.44444\,k^{0.25}$

```
In[148]:= Plot[Vtrue[k], {k, kmin, kmax}]
```

Plot: Limiting value kmin in {k, kmin, kmax} is not a machine–sized real number.

```
Out[148]= Plot[Vtrue[k], {k, kmin, kmax}]
```

Set algorithm parameters

Choose range

```
In[149]:= kmin = 0.2; kmax = 1.5;
```

Choose approximation nodes

```
In[150]:= npts = 14; δk = (kmax - kmin) / (npts - 1);
nodes = Table[x, {x, kmin, kmax, δk}];
```

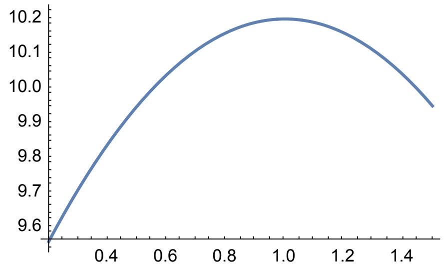
Specify basis functions for value function approximation

```
In[152]:= powers = Table[x^i, {i, 0, 4}];
```

Set initial guess

In[153]:=
```
cmin = f[kmin] - kmin;
valinit[x_] = -(x - 1)^2 + u[cmin] / (1 - β);
Plot[valinit[x], {x, kmin, kmax}]
```

Out[155]=



Define Bellman operator

Define newval[x] which computes the new value of of V[x] given by the RHS of the Bellman equation.

In[158]:=
```
newval[x_] := FindMaximum[
    (* Objective *)
        u[c] + β val[f[x] - c],
    (* Initial guess *)
        {c, css},
    AccuracyGoal → 6][[1]]
```

First VFI

Set val[x] to the initial guess.

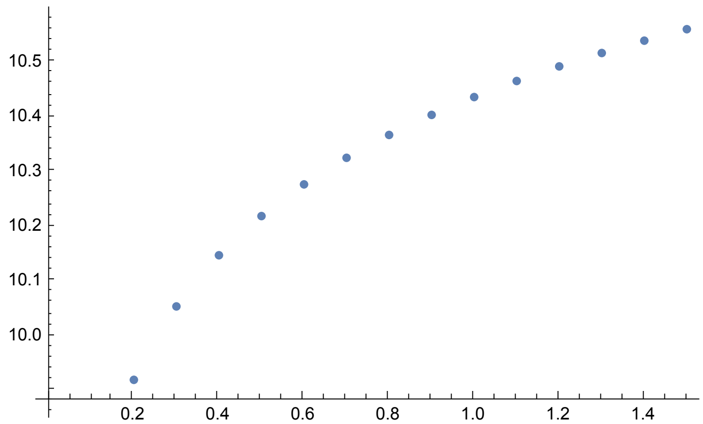
In[188]:=
```
val[x_] = valinit[x];
```

Do first VFI

newval[nodes[[i]]] is the Bellman maximum when x = nodes[[i]]. We create a Table of these pairs

In[189]:=
```
newvs = Table[{nodes[[i]], newval[nodes[[i]]]}, {i, 1, Length[nodes]}];
ListPlot[newvs]
```

Out[190]=



Compute new value function

Use Fit, Mathematica’s regression command, to fit a polynomial to the data where data is

In[191]:=
```
newvs // TableForm
```

Out[191]//TableForm=

0.2	9.91894
0.3	10.0531
0.4	10.1467
0.5	10.2183
0.6	10.2762
0.7	10.3248
0.8	10.3666
0.9	10.4033
1.	10.4359
1.1	10.4652
1.2	10.4919
1.3	10.5164
1.4	10.539
1.5	10.5599

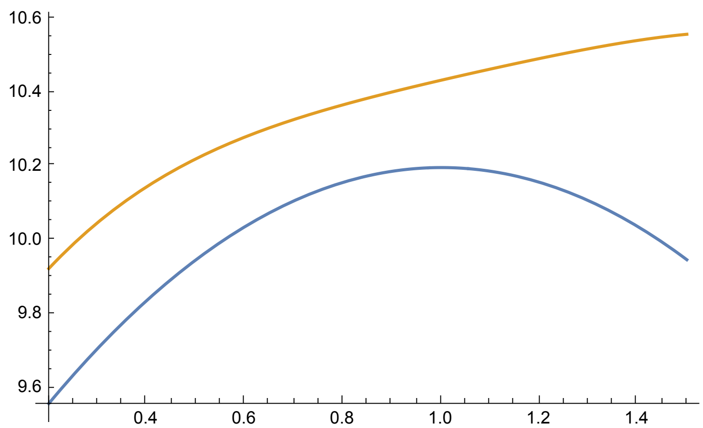
and basis functions are

In[192]:=
```
powers
```

Out[192]= $\{1, x, x^2, x^3, x^4\}$

In[193]:=
```
Clear[val];
val[x_] = Fit[newvs, powers, x];
Plot[{valinit[x], val[x]}, {x, kmin, kmax}]
```

Out[195]=



Define VFI script

Define a value function iteration command

In[196]:=
```
vfi := (
  (* Collect new values*)
     newvs = Table[
             {nodes[[i]], newval[nodes[[i]]]},
          {i, 1, Length[nodes]}];
  (* Compute new value function, and plot it*)
  Clear[val]; val[x_] = Fit[newvs, powers, x];
  Plot[{Vtrue[x], val[x]}, {x, kmin, kmax}, PlotRange → {10, 13}])
```

We have done one iteration; so set iter

In[197]:=
```
iter = 1;
```

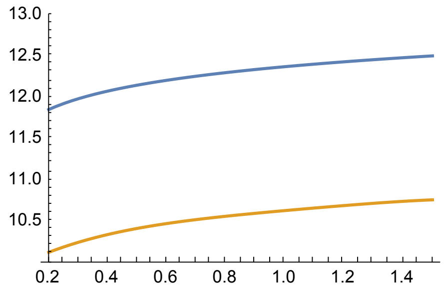
Now iterate

In[198]:=
```
Print["iteration number:"]; iter = iter + 1
vfi
```

```
iteration number:
```

Out[198]= 2

Out[199]=

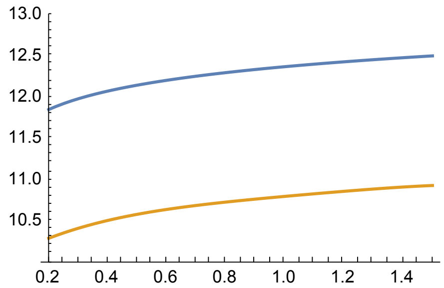


In[200]:=
```
Print["iteration number:"]; iter = iter + 1
vfi
```

```
iteration number:
```

Out[200]= 3

Out[201]=

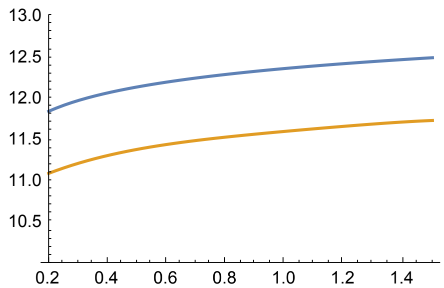


```
Do[iter = iter + 1; vfi, {6}];
Print["iteration number:"];
iter = iter + 1
vfi
```

iteration number:

Out[203]= 10

Out[204]=

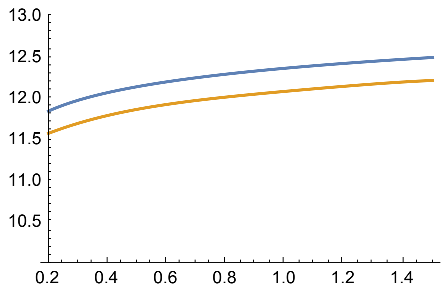


```
Do[iter = iter + 1; vfi, {9}];
Print["iteration number:"];
iter = iter + 1
vfi
```

iteration number:

Out[206]= 20

Out[207]=

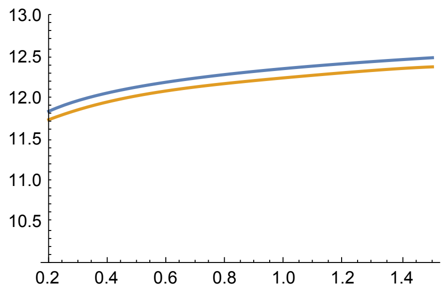


```
Do[iter = iter + 1; vfi, {9}];
Print["iteration number:"];
iter = iter + 1
vfi
```

iteration number:

Out[209]= 30

Out[210]=



```
Do[iter = iter + 1; vfi, {19}];
Print["iteration number:"];
iter = iter + 1
vfi
```

iteration number:

Out[212]= 50

Out[213]=