Life-Cycle Consumption Problems with Borrowing

```
In[801]:=  x = 0; Remove["Global`*"]; Date[] // Most
```

```
Out[801]=  {2020, 3, 10, 9, 6}
```

In this notebook, we examine a variety of life-cycle optimization problems, including borrowing constraints, labor supply, and income taxes.

Life Cycle Consumption - with explicit modeling of savings

The previous examples represented the budget constraint in terms of the present values of consumption and income. This is not adequate in many applications. Therefore, we instead track the wealth of an individual over the course of his life.

```
In[802]:= x = 0; Remove["Global`*"]; Date[]

Out[802]= {2020, 3, 10, 9, 6, 18.102405}
```

Define the objective as before.

```
In[803]:= obj := Sum[β^(t-1) u[Cons[t]], {t, 1, T}]
```

We now keep track of assets.
Let Assets[t] be the assets (or savings) in hand at the beginning of period t.
In period t, the individual earns Wage[t] in wages
consumes Cons[t],
and earns has gross return R on his assets.
Wealth at the beginning of period t+1 equals R Assets[t]+Wage[t]-Cons[t]), where R=1+r.

The dynamic budget constraint is now expressed in terms of the period-by-period changes in wealth, Assets[t].

```
In[804]:= budget := Table[Assets[t + 1] - (R Assets[t] + Wage[t] - Cons[t]) ≤ 0, {t, 1, T}]
```

We express the constraint as a weak inequality. This allows the individual to throw away money at the end of time t. However, they won’t do that so the constraint will bind at the solution.

The initial wealth, at the beginning of period 1, is zero.

```
In[805]:= Assets[1] = 0;
```

The terminal wealth must be nonnegative (you can’t go without paying your debts).

```
In[806]:= bcT := {Assets[T + 1] ≥ 0}
```

We impose a minimum asset level, AssetMin <= 0. If AssetMin=0, then borrowing is not allowed. If AssetMin is sufficiently large in magnitude then an individual can do as much borrowing as he wants as long as he can repay it before time T.

```
In[807]:= AssetBnd := Table[Assets[t] ≥ AssetMin, {t, 2, T}]
```

We add lower bound constraints on consumption

```
In[808]:= LoBndsC := Table[Cons[t] ≥ 0.0001, {t, 1, T}]
```

Construct the collection of all constraints.

```
In[809]:= constraints := Union[budget, bcT, AssetBnd, LoBndsC]
```

List the variables, which now includes both the path of consumption and assets.

```
In[810]:= varsCons := Table[Cons[t], {t, 1, T}]
          varsAssets := Table[Assets[t], {t, 2, T + 1}]
          vars := Union[varsCons, varsAssets]
```

Simple example

Choose an interest rate and a discount factor

```
In[813]:= r = 0.1; R = 1 + r; β = 0.96;
```

Specify a utility function

```
In[814]:= u[c_] = -Exp[-c];
```

Specify the minimum on assets.

```
In[815]:= AssetMin = 0;
```

Specify a horizon

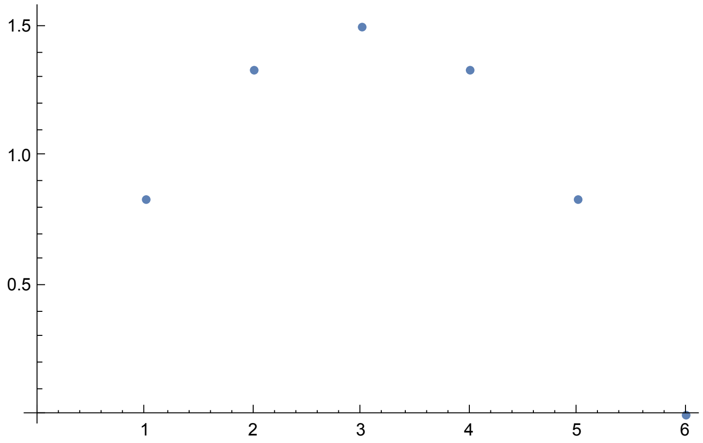
```
In[816]:= T = 6;
```

Specify the income path. We choose a pattern that corresponds to a life-cycle earnings pattern

```
In[817]:= Do[Wage[i] = (T - i) i / T, {i, 1, T}]

In[818]:= Table[Wage[i], {i, 1, T}];
          ListPlot[%]
```

Out[819]=



Display problem elements

```
In[820]:= vars

Out[820]= {Assets[2], Assets[3], Assets[4], Assets[5], Assets[6],
           Assets[7], Cons[1], Cons[2], Cons[3], Cons[4], Cons[5], Cons[6]}

In[821]:= obj
```

Out[821]= $-1.\ e^{-\text{Cons}[1]} - 0.96\ e^{-\text{Cons}[2]} - 0.9216\ e^{-\text{Cons}[3]} - 0.884736\ e^{-\text{Cons}[4]} - 0.849347\ e^{-\text{Cons}[5]} - 0.815373\ e^{-\text{Cons}[6]}$

```
In[822]:= budget // TableForm
Out[822]//TableForm=
        -0.833333 + Assets[2] + Cons[1] ≤ 0
        -4/3 - 1.1 Assets[2] + Assets[3] + Cons[2] ≤ 0
        -3/2 - 1.1 Assets[3] + Assets[4] + Cons[3] ≤ 0
        -4/3 - 1.1 Assets[4] + Assets[5] + Cons[4] ≤ 0
        -5/6 - 1.1 Assets[5] + Assets[6] + Cons[5] ≤ 0
        -1.1 Assets[6] + Assets[7] + Cons[6] ≤ 0

In[823]:= bcT
Out[823]= {Assets[7] ≥ 0}

In[824]:= AssetBnd // TableForm
Out[824]//TableForm=
        Assets[2] ≥ 0
        Assets[3] ≥ 0
        Assets[4] ≥ 0
        Assets[5] ≥ 0
        Assets[6] ≥ 0

In[825]:= constraints // TableForm
Out[825]//TableForm=
        Assets[2] ≥ 0
        Assets[3] ≥ 0
        Assets[4] ≥ 0
        Assets[5] ≥ 0
        Assets[6] ≥ 0
        Assets[7] ≥ 0
        Cons[1] ≥ 0.0001
        Cons[2] ≥ 0.0001
        Cons[3] ≥ 0.0001
        Cons[4] ≥ 0.0001
        Cons[5] ≥ 0.0001
        Cons[6] ≥ 0.0001
        -0.833333 + Assets[2] + Cons[1] ≤ 0
        -4/3 - 1.1 Assets[2] + Assets[3] + Cons[2] ≤ 0
        -3/2 - 1.1 Assets[3] + Assets[4] + Cons[3] ≤ 0
        -4/3 - 1.1 Assets[4] + Assets[5] + Cons[4] ≤ 0
        -5/6 - 1.1 Assets[5] + Assets[6] + Cons[5] ≤ 0
        -1.1 Assets[6] + Assets[7] + Cons[6] ≤ 0
```

Solve the problem

```
In[826]:= sol = FindMaximum[{obj, constraints}, vars];
```

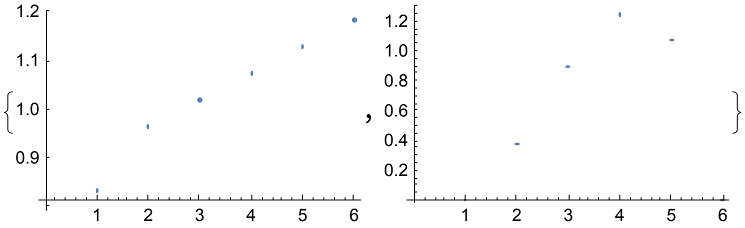
Collect and plot the consumption and asset paths

```
In[827]:= Table[Cons[i], {i, 1, T}] /. sol[[2]]

Out[827]= {0.83333, 0.962553, 1.01704, 1.07153, 1.12602, 1.18051}

In[828]:= {ListPlot[varsCons /. sol⟦2⟧], ListPlot[varsAssets /. sol⟦2⟧]}
```

Out[828]=



A better example

Choose an interest rate and a discount factor

```
In[829]:= r = 0.1; R = 1 + r; β = 0.96;
```

Specify a utility function

```
In[830]:= u[c_] = -Exp[-c];
```

Specify the minimum on assets.

```
In[831]:= AssetMin = 0;
```

Specify a horizon

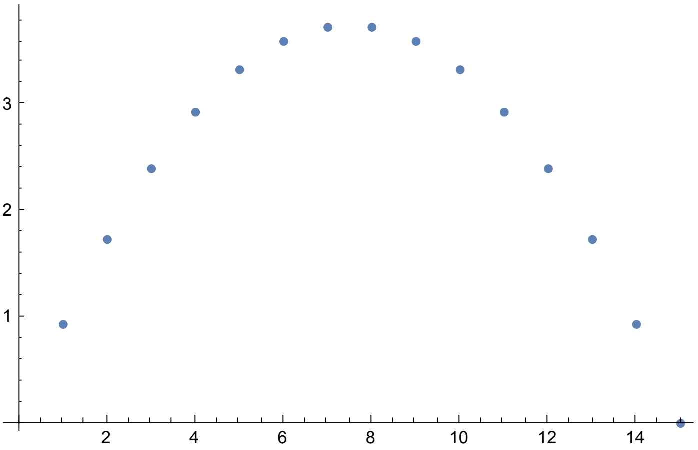
```
In[832]:= T = 15;
```

Specify the income path. We choose a pattern that corresponds to a life-cycle earnings pattern

```
In[833]:= Do[Wage[i] = (T - i) i / T, {i, 1, T}]
```

```
In[834]:= Table[Wage[i], {i, 1, T}];
          ListPlot[%]
```

Out[835]=



Solve the problem

```
In[836]:= sol = FindMaximum[{obj, constraints}, vars];
```

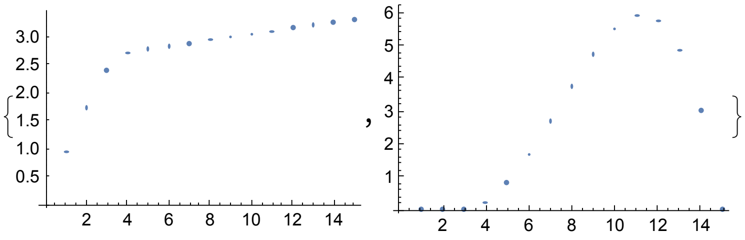
Collect and plot the consumption and asset paths

```
In[837]:= Table[Cons[i], {i, 1, T}] /. sol[[2]]

Out[837]= {0.933333, 1.73333, 2.4, 2.71353, 2.76803, 2.82252, 2.87701,
           2.9315, 2.98599, 3.04048, 3.09497, 3.14945, 3.20394, 3.25843, 3.31292}
```

```
In[838]:= {ListPlot[varsCons /. sol〚2〛], ListPlot[varsAssets /. sol〚2〛]}
```

Out[838]=



Eliminate borrowing constraint

Resolve the problem but with no borrowing constraint

```
In[839]:= AssetMin = -Infinity;
```

```
In[840]:= sol = FindMaximum[{obj, constraints}, vars];
```

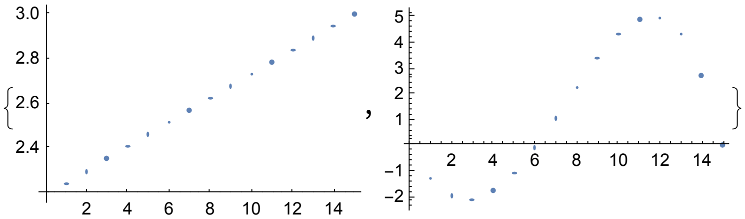
Collect and plot the consumption and asset paths

```
In[841]:= Table[Cons[i], {i, 1, T}] /. sol[[2]]

Out[841]= {2.23659, 2.29108, 2.34557, 2.40005, 2.45454, 2.50903, 2.56352,
           2.61801, 2.6725, 2.72698, 2.78147, 2.83596, 2.89045, 2.94494, 2.99942}
```

```
In[842]:= {ListPlot[varsCons /. sol〚2〛], ListPlot[varsAssets /. sol〚2〛]}
```

Out[842]=



Note that the solution now implies borrowing in the early years, financed by the higher wages in later periods.

Life Cycle Consumption - with savings and elastic labor supply

```
In[843]:= x = 0; Remove["Global`*"]; Date[]

Out[843]= {2020, 3, 10, 9, 6, 19.325736}
```

We now add labor supply decisions to the problem. Cons[t] (Lab[t]) is consumption (labor supply) at time t.
Define the objective, which now has a utility function over consumption and labor supply in each period.

```
In[844]:= obj := Sum[β^(t-1) u[Cons[t], Lab[t]], {t, 1, T}]
```

Define constraints:

```
In[845]:= budget :=
           Table[Assets[t + 1] - (R Assets[t] + Wage[t] × Lab[t] - Cons[t]) ≤ 0, {t, 1, T}]
          bcT := {Assets[T + 1] ≥ 0}
          Assets[1] = 0;
          AssetBnd := Table[Assets[t] ≥ AssetMin, {t, 2, T}]
```

List the variables

```
In[849]:= varsCons := Table[Cons[t], {t, 1, T}]
          varsLab := Table[Lab[t], {t, 1, T}]
          varsAssets := Table[Assets[t], {t, 2, T + 1}]
          vars := Union[varsCons, varsLab, varsAssets]
```

We should also add the lower bound constraints. We allow labor to go down to zero, but not consumption.

```
In[853]:= LoBndsC := Table[Cons[t] ≥ 0.0001, {t, 1, T}]
          LoBndsL := Table[Lab[t] ≥ 0, {t, 1, T}]
          Bnds := Union[LoBndsC, LoBndsL]
```

Now we form a new constraint set combining the budget constraint with the positivity constraints.

```
In[856]:= constraints := Union[budget, bcT, AssetBnd, Bnds]
```

Specify the horizon.

```
In[857]:= T = 15;
```

Specify the wage path. We choose a pattern that corresponds to a life-cycle,

```
In[858]:= Do[Wage[i] = (T - i) i / T, {i, 1, T}]
```

Choose an interest rate and a discount factor

```
In[859]:= r = 0.1; R = 1 + r; β = 0.96;
```

Specify a utility function

```
In[860]:= u[c_, l_] = -Exp[-c] - l^2;

In[861]:= AssetMin = 0;

In[862]:= sol = FindMaximum[{obj, constraints}, vars];
```

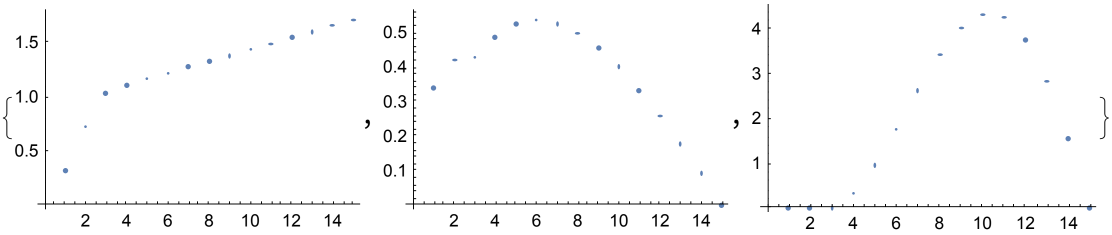
Collect and plot the consumption, labor supply, and asset paths

```
In[863]:= Table[Cons[i], {i, 1, T}] /. sol[[2]]

Out[863]= {0.317173, 0.726484, 1.0291, 1.10376, 1.15825, 1.21274, 1.26723,
           1.32172, 1.3762, 1.43069, 1.48518, 1.53967, 1.59416, 1.64865, 1.70314}

In[864]:= {ListPlot[varsCons /. sol⟦2⟧],
           ListPlot[varsLab /. sol⟦2⟧], ListPlot[varsAssets /. sol⟦2⟧]}
```

Out[864]=



Life Cycle Consumption - with savings, elastic labor supply, and debt

```
In[865]:=  x = 0; Remove["Global`*"]; Date[]

Out[865]=  {2020, 3, 10, 9, 6, 19.668927}
```

This notebook describes a basic life-cycle problem with savings, labor supply, and debt.

Let Cons[t] (Lab[t]) be consumption (labor supply) in period t, where "life" begins at t=1 and continues to t=T. We assume that the utility at time t is u[Cons[t], L[t]] and that utility is discounted at *β*. Therefore, the life-cycle objective is

```
In[866]:=  PVUtil := Sum[β^(t-1) u[Cons[t], Lab[t]], {t, 1, T}]
```

We set the objective for the optimization solver to be PVUtil.

```
In[867]:=  obj := PVUtil;
```

R will be the gross interest rate, which is one plus the interest rate; hence, a dollar saved at time t will be worth R=1+r at time t+1. Let Wage[t] be the labor income in period t. The present values of consumption expenditures and wage income are

```
In[868]:=  PVc := Sum[Cons[t] R^(-t+1), {t, 1, T}]
           PVw := Sum[Wage[t] R^(-t+1), {t, 1, T}]
```

We now keep track of assets.
Let Assets[t] (>= 0) be the assets (or savings) in hand and Debt[t] (>= 0) be debt at the beginning of period t.
In period t, the individual
earns Wage[t] Lab[t] in wages
consumes Cons[t],
borrows B[t] (where negative B[t] represents paying off debt),
earns gross return R on his assets,
and pays (Rb-1) Debt[t] in interest on his debt.
Wealth at the beginning of period t+1 equals R Assets[t]+Wage[t]-Cons[t]), where R=1+r.
Debt at the beginning of period t+1 equals Rb Debt[t] + B[t].

The dynamic budget constraint is now expressed in terms of the period-by-period changes in wealth, Assets[t].

```
In[870]:= budgetAssets :=
           Table[Assets[t + 1] ≤ R Assets[t] + Borrow[t] + Wage[t] × Lab[t] - Cons[t], {t, 1, T}]
```

The second constraint says that Debt[t+1] cannot be less than debt at the end of period t plus interest payments.

```
In[871]:= budgetDebt := Table[Debt[t + 1] ≥ Rd Debt[t] + Borrow[t], {t, 1, T}]
```

COMMENT: Why we use inequality constraints here
Mathematica uses an interior point method. IP methods do not like equality constraints; in fact, they convert them into increasingly tight inequality constraints.

Keep this in mind: If you have a choice between equality and inequality constraints, try both since different solvers will have different preferences over how you formulate the problem.

You initially have neither assets nor debt.

```
In[872]:= Assets[1] = Debt[1] = 0;
```

Assets and Debt must be nonnegative at each time t

```
In[873]:= ADDefn := Table[{Assets[t] ≥ 0, Debt[t] ≥ 0}, {t, 2, T}] // Flatten
```

The terminal wealth must be nonnegative (you can’t take it with you) and terminal debt must be zero.

```
In[874]:= bcT := {Assets[T + 1] ≥ 0, Debt[T + 1] == 0}
```

We impose a maximum debt level, DebtMax.
If DebtMax=0, then no borrowing is allowed.
If DebtMax is sufficiently large then an individual can borrow as he wants as long as he can repay it before time T+1.

```
In[875]:= DebtBnd := Table[Debt[t] ≤ DebtMax, {t, 2, T}]
```

We add lower bound constraints on consumption and labor supply:

```
In[876]:= LoBndsC := Table[Cons[t] ≥ 1 / 1000, {t, 1, T}];
          LoBndsL := Table[Lab[t] ≥ 0, {t, 1, T}]
```

We also include an upper bound, Lmax, on labor supply, representing the maximum number of hours in a day, or year, or whatever the unit of time is.

```
In[878]:= UpBndsL := Table[Lab[t] ≤ Lmax, {t, 1, T}]
```

We put all the bounds in Bnds:

```
In[879]:= Bnds := Union[LoBndsC, LoBndsL, UpBndsL]
```

List the variables:

```
In[880]:= varsCons := Table[Cons[t], {t, 1, T}]
          varsLab := Table[Lab[t], {t, 1, T}]
          varsAssets := Table[Assets[t], {t, 2, T + 1}]
          varsDebt := Table[Debt[t], {t, 2, T + 1}]
          varsBorrow := Table[Borrow[t], {t, 1, T}]
          vars := Union[varsCons, varsLab, varsAssets, varsDebt, varsBorrow]
```

Now we form a new constraint set combining the budget constraint with the positivity constraints.

```
In[886]:= constraints := Union[budgetDebt, budgetAssets, bcT, DebtBnd, ADDefn, Bnds]
```

Numerical example

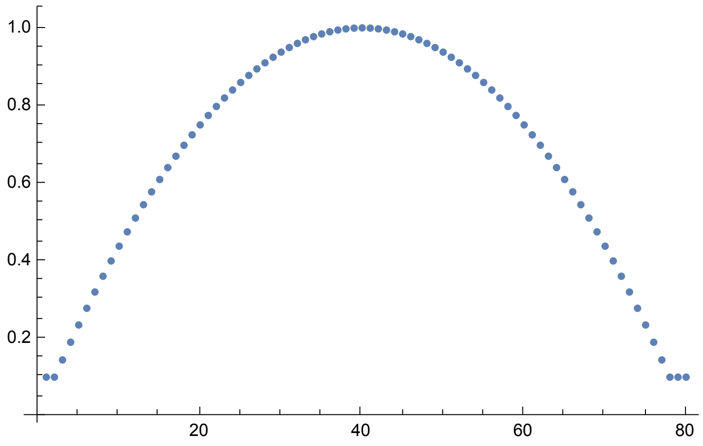
Choose length of life. I choose 80 in this case to represent annual time periods.

```
In[887]:= T = 80;
```

Specify the income path. We choose a pattern that corresponds to a life-cycle. Parabolas are close approximations. Here is a good basic choice. We also adjust the path so that the max wage is 1 at t=T/2. This allows you to change T without affecting the basic wage path.

```
In[888]:= Wage[t_] = Max[1 / 10, (4 t / T) (1 - t / T)];
          lifeW = Table[{t, Wage[t]}, {t, 1, T}];
          ListPlot[lifeW]
```

Out[890]=



Choose interest rates, borrowing and lending, and a discount factor. Large T and β close to 1 implies short units of time. T=80 is similar to assuming a year is the unit of time. In that case we choose the following parameter values.

```
In[891]:= r = 57 / 1000; R = 1 + r ; β = 96 / 100; rd = 8 / 100; Rd = 1 + rd;
```

Our utility functions will have the form

$$c^{1-\gamma}/(1-\gamma) - B\, l^{1+\eta}/(1+\eta)$$

In[892]:= $u[c\_, l\_] := c^{1-\gamma} / (1-\gamma) - B\, l^{1+\eta} / (1+\eta)$

Set parameters of our utility function:

```
In[893]:= B = 1; γ = 3; η = 2; Lmax = 10;
          u[c, l]
```

Out[894]= $-\frac{1}{2\, c^2} - \frac{l^3}{3}$

For our example, we will set DebtMax = 2 (which equals twice the peak wage);

```
In[895]:= DebtMax = 2;
```

Compute the solution

```
In[896]:= solution = FindMaximum[{obj, constraints}, vars];
```

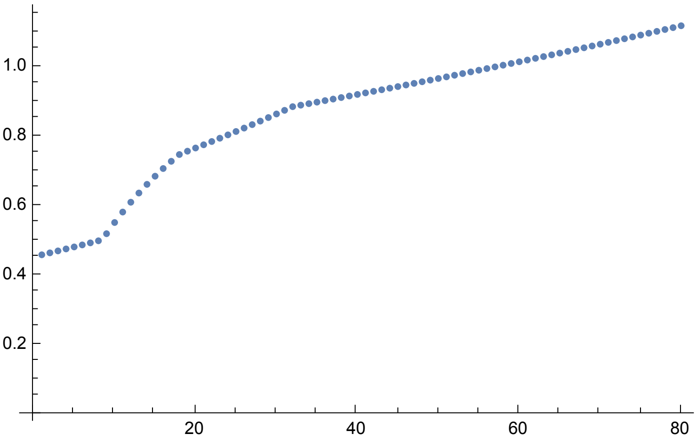
Output

consumption

Output the consumption path, first in a graph and then listing the results.

```
In[897]:= lifeC = Table[{i, Cons[i]}, {i, 1, T}] /. solution[[2]];
          ListPlot[lifeC, PlotStyle -> PointSize[0.01]]
```

Out[898]=



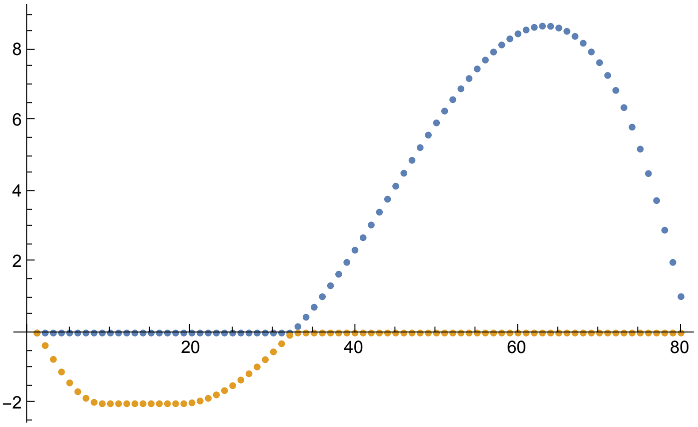
```
In[899]:= varsCons /. solution[[2]];
```

Assets and Debt

We next plot the path for Assets and for Debt, and also output the results

```
In[900]:= lifeS = Table[{i, Assets[i]}, {i, 1, T}] /. solution[[2]];
          lifeD = Table[{i, -Debt[i]}, {i, 1, T}] /. solution[[2]];
          ListPlot[{lifeS, lifeD}, PlotStyle -> PointSize[0.01]]
```

Out[902]=



```
In[903]:= varsAssets /. solution[[2]];
```

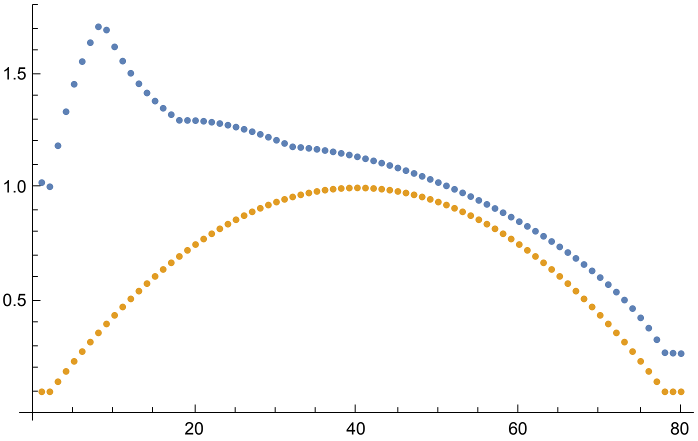
```
In[904]:= varsDebt /. solution[[2]];
```

Wages and labor supply

We now plot both the wage path and the labor supply path. Note that the labor supply behaves very different when the borrowing constraints bind than when they don’t.

```
In[905]:= lifeL = Table[{i, Lab[i]}, {i, 1, T}] /. solution[[2]];
ListPlot[{lifeL, lifeW}, PlotStyle -> PointSize[0.01]]
```

Out[906]=



```
In[907]:= varsLab /. solution[[2]];
```

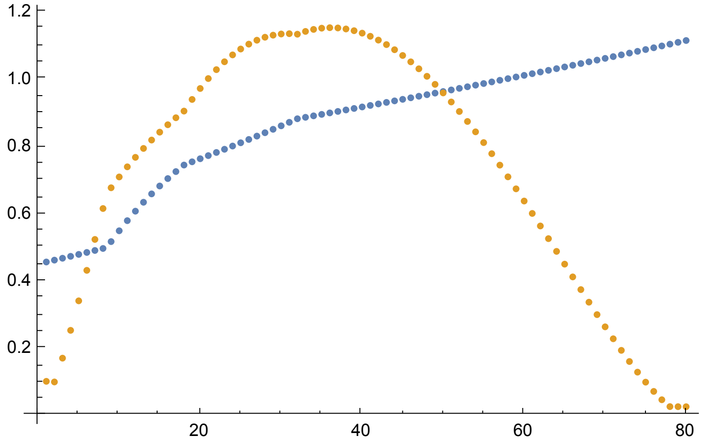
Earnings and consumption

```
In[908]:= earnings = Table[{t, Wage[t] × Lab[t]}, {t, 1, T}] /. solution[[2]];
```

This graph shows how wages, savings and consumption move over the life cycle.

```
In[909]:= ListPlot[{lifeC, earnings}, PlotStyle -> PointSize[0.01], PlotRange → All]
```

Out[909]=



Life Cycle Consumption - with savings, elastic labor supply, and more general utility functions

```
In[910]:= x = 0; Remove["Global`*"]; Date[]

Out[910]= {2020, 3, 10, 9, 6, 20.744118}
```

Our utility functions will often be of the following form:

$$\left(A\,c^{\alpha} + B\,(\text{Lmax} - l)^{\beta}\right)^{\gamma}$$

where Lmax is the maximum amount of time that one can supply as labor. If l represents labor supply then Lmax − l is leisure.
So, the general way to specify this is with parameters A, B, Lmax, α, δ, and γ in a data file.

```
In[911]:= u[c_, l_] := (A c^α + B (Lmax - l)^δ)^γ
```

We need code where we can easily change the utility function. I think GAMS allows you to define simple functions. I have never done this; my students would find this quite useful.

Define the objective with a per period utility function over consumption and labor.

```
In[912]:= obj := Sum[β^(t-1) u[Cons[t], Lab[t]], {t, 1, T}]
```

Our list of bounds will now include an upper bound, Lmax, on labor supply. This will correspond to the maximum number of hours in a day, or year, or whatever the unit of time is.

```
In[913]:= LoBndsC := Table[Cons[t] ≥ 0.0001, {t, 1, T}];
          LoBndsL := Table[Lab[t] ≥ 0, {t, 1, T}]
          UpBndsL := Table[Lab[t] ≤  Lmax - 0.0001, {t, 1, T}]
          Bnds := Union[LoBndsC, LoBndsL, UpBndsL]
```

Same old stuff:

```
In[917]:= budget :=
            Table[Assets[t + 1] - (R Assets[t] + Wage[t] × Lab[t] - Cons[t]) ≤ 0, {t, 1, T}]
           bcT := {Assets[T + 1] ≥ 0}; Assets[1] = 0;
           AssetBnd := Table[Assets[t] ≥ AssetMin, {t, 2, T}]; AssetMin = 0;
           (* List the variables *)
           varsCons := Table[Cons[t], {t, 1, T}]; varsLab := Table[Lab[t], {t, 1, T}]
           varsAssets := Table[Assets[t], {t, 2, T + 1}]
           vars := Union[varsCons, varsLab, varsAssets]
           (* Specify the horizon. *)
           T = 15;
           (* Specify the wage path *)
           Do[Wage[i] = (T - i) i / T, {i, 1, T}]
           (* Choose an interest rate and a discount factor *)
           r = 0.1; R = 1 + r;  β = 0.96;
```

Form set of all constraints.

```
In[926]:= constraints := Union[budget, bcT, AssetBnd, Bnds]
```

A numerical example

Set parameters of our utility function:

```
In[927]:=  Lmax = 24; A = B = 1; α = 1 / 3; δ = 3 / 7; γ = 5 / 6;
```

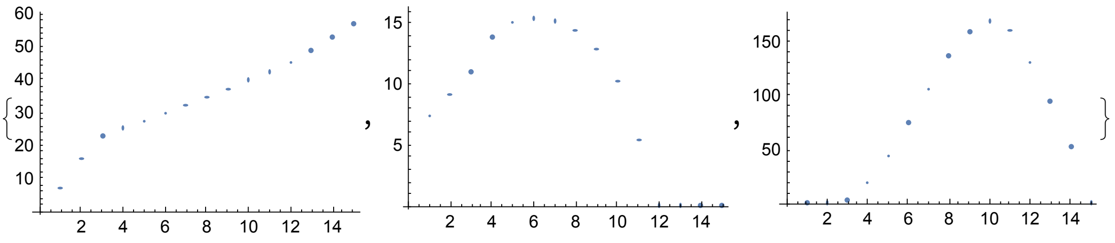
```
In[928]:= u[c, l]
```

Out[928]= $\left(c^{1/3} + (24 - l)^{3/7}\right)^{5/6}$

Solve problem, collect and plot the consumption, labor supply, and asset paths

```
In[929]:= sol = FindMaximum[{obj, constraints}, vars];
           {ListPlot[varsCons /. sol⟦2⟧],
            ListPlot[varsLab /. sol⟦2⟧], ListPlot[varsAssets /. sol⟦2⟧]}
```

Out[930]=


60
50
40
30
20
10
2
4
6
8
10
12
14
15
10
5
2
4
6
8
10
12
14
150
100
50
2
4
6
8
10
12
14