

Amortisation: What a killer

Teacher Notes and Answers

7 8 9 10 11 12



TI-Nspire CAS



Investigation



Teacher



90 min






Introduction

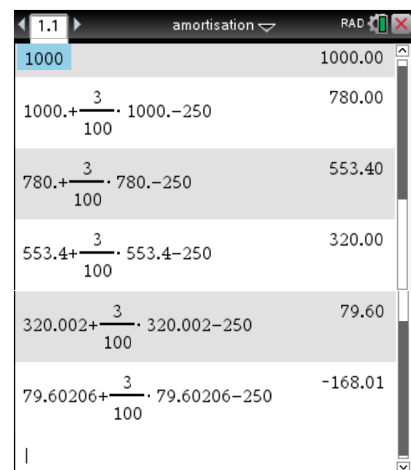
In its original meaning, *amortisation* means ‘to kill’, so the amortisation of a loan can be thought of as the process of killing off a debt. In this task we will look at how amortisation works for loans where the interest is calculated on the balance as it reduces over the life of the loan.

Recursion – baby steps!

Consider a small bank loan for \$1000. The interest rate for this loan is 12% p.a. compounding quarterly. At the end of each quarter, the interest is calculated and added to the value of the loan. In addition a repayment of \$250 is made. The calculator application of the TI-Nspire CAS can be used to find the reducing balance of this loan at the end of each quarter, as follows.

On the TI-Nspire CAS

- Press  > **New Document**, and then select **Add Calculator**
- Press  > **Settings** > **Document Settings** and change **Display Digits** to “Fix 2”.
- Press **OK** and then select **Current** to return to the **Calculator** page.
- Type **1000** and press 
- Type **ans + 3/100*ans – 250** and then press 
- Press  until the balance is a negative number.



Balance	Interest	Repayment	New Balance
1000.00			1000.00
$1000.00 + \frac{3}{100} \cdot 1000.00 - 250$			780.00
$780.00 + \frac{3}{100} \cdot 780.00 - 250$			553.40
$553.40 + \frac{3}{100} \cdot 553.40 - 250$			320.00
$320.00 + \frac{3}{100} \cdot 320.00 - 250$			79.60
$79.60206 + \frac{3}{100} \cdot 79.60206 - 250$			-168.01

The results are shown in the screen shot at right. Note that in each new calculation of the quarterly balance, the calculator replaces ‘ans’ with the previous balance (i.e. the previous ‘answer’).

Question 1.

Answer the following questions about the above loan repayment scenario.

- How many quarterly repayments of \$250 are needed to amortise the loan? **5 payments (5th payment of \$250 will mean the bank owes the client money!)**
- By how much could the final repayment be increased, so that the loan can be *amortised* after 4 payments? **Final payment = \$250 + \$79.60 = \$329.60**
- Explain why making four quarterly repayments of \$250 and a fifth payment at the end of fifth quarter \$79.60 would not amortise the loan.

Using this plan, the loan balance would be \$79.60 after 4 quarters. However, there would be an interest charge in the fifth quarter of $\frac{3}{100} \times \$79.60 = \2.39 which would also have to be paid to fully amortise the loan. (So fifth repayment = $\$79.60 + \$2.39 = \$81.99$)

- d) If V_n and V_{n+1} are used to represent successive loan balance amounts at the end of each quarter, write a recurrence relation that describes how successive balances are calculated.

Different answers are possible, all different forms of the following *recursive* relationship:

new balance value = old balance value + interest – repayment

Here are three variants which are suitable.

$$V_{n+1} = V_n + \frac{3}{100} \times V_n - 250, \quad V_0 = 1000$$

$$V_{n+1} = \left(1 + \frac{3}{100}\right) \times V_n - 250, \quad V_0 = 1000$$

$$V_{n+1} = 1.03 \times V_n - 250, \quad V_0 = 1000$$

Creating an amortisation table with “amortloan”

The previous amortisation example can be analysed further using a short program entitled “amortloan”.

On the TI-Nspire CAS

- Open the TI-Nspire file “amortloan.tns”
- Press and select **amortloan()** to run the program
- Enter the following values when requested.
 - Loan value (\$) = **1000**
 - Interest rate (%p.a.) = **12**
 - No. of payments to show = **4**
 - Payments (\$) = **250**
 - Cpy/Ppy = **4**
 - Adjust final payment (y/n)? **y**

Observe that the program has created a table, and provided two options for fully amortising the loan (if relevant to the scenario).

- An *adjusted* (increased) 4th payment, **or**
- An *extra* (5th) payment

Program output from “amortloan”

```

1.1 | amortloan
-----
amortloan()
Loan value ($) = 1000
Interest rate (% p.a.) = 12
No. payments to show = 4
Payment ($) = 250
Cpy/Ppy = 4
["num" "pmt" "int" "p_red" "bal"]
0.    0.    0.    0.    1000.
1.   250.  30.4  220.  780.
2.   250.  23.4  226.6  553.4
3.   250.  16.6  233.4  320.
4.   250.   9.6  240.4  79.6
Adjust final payment?(y/n) y
Adjusted Amort. Table (last line only)
["num" "pmt" "int" "p_red" "bal"]
4.   329.6  9.6  320.  0.
...or if extra payment (No. 5.)
= $79.6 + (12. % / 4.) / 100 * $79.6 = $81.99
  
```

What is “principal reduction”?

There is a column in the table that needs further explanation. The “**p_red**” column refers to “*principal reduction*”, which is the amount by which the original loan value (or principal) has been reduced in each quarter. As you can see from the table, the principal reduction is calculated as the difference between the repayment and the value of the interest in that quarter. In general, for each time period:

$$\text{principal reduction} = \text{repayment} - \text{interest}$$

Question 2.

How does the value of ‘principal reduction’ change as the loan repayment progresses over time? Explain why this change occurs.

The value of ‘principal reduction’ increases over the life of the loan. This happens because at the start of the loan period, the difference between the repayment value and the interest value is the smallest.

As the loan progresses, this difference increases, and more of the principal is being reduced.

Further analysis of the amortisation table

Once the program has been run, each column of values in the amortisation table is stored for further analysis. Press **VAR** to access them at any time. Here's what each variable represents.

- **num** number of loan balances over loan period
- **pmt** fixed repayment
- **interest** interest payment over the loan period
- **prinreduct** principal reduction amount over loan period
- **balance** loan balance over the loan period
- **amorttab** Final amortisation table
(including adjusted final payment if relevant)

num	pmt	int	p_red	bal
0.00	0.00	0.00	0.00	1000.00
1.00	250.00	30.00	220.00	780.00
2.00	250.00	23.40	226.60	553.40
3.00	250.00	16.60	233.40	320.00
4.00	329.60	9.60	320.00	0.00

The screen shows the output when **var** is pressed, and each of the variables is selected.

[Note: If you select “**amortloan**” it will just relaunch the program]

If you have not yet done so, type “**amortloan()**” to run the program. Enter the values as per the example on page 2, and type ‘y’ to ask the program to adjust the final payment for full amortisation in 4 payments. Then use the **var** key to verify that you get the same values as shown on the screen above right.

Now answer the following questions.

Question 3.

Type **sum(interest)** and then **enter**. What was the result, and what does it represent?

The total interest paid over the life of the loan (\$79.60).

interest	{ 30.00,23.40,16.60,9.60 }
sum(interest)	79.60
prinreduct	{ 220.00,226.60,233.40,320.00 }
sum(prinreduct)	1000.00
sum(pmt)	1079.60
sum(pmt)-sum(interest)	1000.00

Question 4.

Type **sum(prinreduct)** and then **enter**. What was the result, and what does it represent?

The total amount that the principal has been reduced over the life of the loan (from \$1000 to \$0).

Question 5.

Type **sum(pmt)** and then **enter**. What was the result, and what does it represent?

The total value of payments made over the life of the loan (\$1079.60).

Note that it will always be true by definition that

$$\text{sum(prinreduct)} + \text{sum(interest)} = \text{sum(pmt)}$$

Question 6.

Use the program to find out the number of payments required to amortise the above loan if the payment was only \$80/quarter.

It will take 16 payments, or 4 years. The final payment will be slightly less than \$80 (\$72.16) [Note that when using the program, if you enter a larger number of payments than required, it will ignore extra payments if it is able to amortise using fewer payments.]

14.	80.	6.57	73.43	145.68
15.	80.	4.37	75.63	70.05
16.	80.	2.1	77.9	-7.84

Adjust final payment?(y/n) y
Adjusted Amort. Table (last line only)

num	pmt	int	p_red	bal
16.	72.16	2.1	70.06	0.

[Press VAR for more amort. analysis]

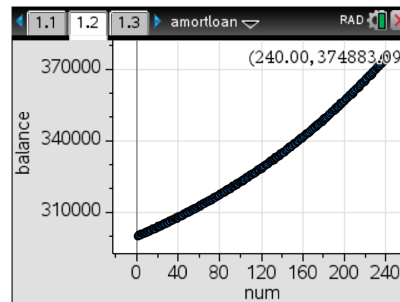
Keeping it real – amortising a housing loan

Of course most loans are for greater amounts of money and take a lot longer to pay off than our simple example. Consider a typical home loan, which might be for \$300 000 with an interest rate 4.75% p.a. (compounded monthly), and is to be amortised over 20 years.

Question 7.

Run the program and try a monthly repayment of \$1000. Explain what happens to the loan balance in this case. Create a scatter plot of the loan balance (**balance**) vs. the number of repayments (**num**) to assist your analysis.

The monthly repayment is less than the interest generated in the first month, and so the loan balance does not reduce – it increases! In such a scenario, the loan will never be paid off, and the debt will increase (for example, if you paid \$1000/month for 20 years, at the end you would still owe \$374,883.09!!!)



Question 8.

Run the program and try 3 other fixed monthly repayments for the loan above. Complete the following table to assess each repayment plan.

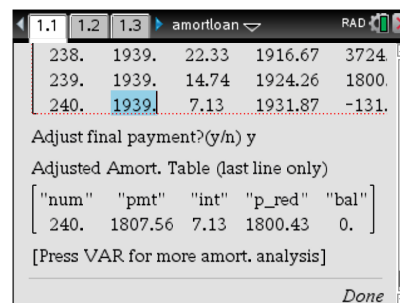
Answers will vary, sample answer follows. (Note: Total Interest can be found using **sum(interest)** after program is run.)

Option	Monthly repayment	How long for amortisation?	Total Interest paid?	Comment
1	1500	398 months (approx. 33 years)	\$295 599	Loan period may be too long, amount of interest to be paid is too high
2	2000	229 months (approx. 19 years)	\$156 034	Possibly best option (close to 20 years).
3	2500	164 months (approx. 14 years)	\$107 768	Amortises quickly, and interest paid is lower, but the monthly repayment is high.

Question 9.

Use the program to find the least monthly repayment (correct to the nearest dollar) that will fully amortise the above loan in exactly 20 years (12 x 20 = 240 payments).

A monthly repayment of just under \$2000 should work. By experimentation, the least monthly repayment that works is \$1939.



Question 10.

Use the program to find the monthly repayment that will mean that you will always have a loan balance of \$300 000 (hint: this is called an ‘interest only’ loan).

The monthly repayment must be the same as the first interest payment (\$1187.50)

Checking with the Finance Solver

[This section makes the assumption that you are already familiar with the Finance Solver, and the meaning and use of the various parameters].

The purpose of the program was to 'look inside' the amortisation process. Now that you have used the TI-Nspire CAS to experiment with the effect of various amortisation parameters, let's check with the inbuilt finance tools available on your calculator. In many instances, such solvers are more efficient in finding the 'best' answer, or determining the consequences of changing a parameter on the amortisation time or related costs. The Finance solver can be accessed by pressing **menu** > **Finance** > **Finance Solver**.

Question 11.

Use the Finance solver to check your answer to Question 9.

N = 240

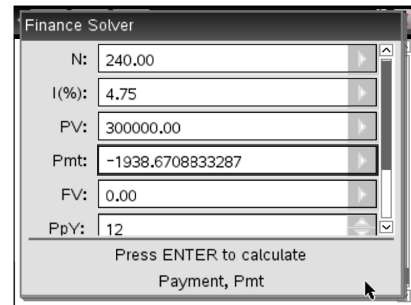
I% = 4.75

PV = \$300 000

PMT = ??

FV = 0

PpY/CpY = 12



So the least repayment option to amortise this loan over 20 years is confirmed as \$1939/month (correct to the nearest dollar).

Question 12.

In the same loan scenario as above, how much time would be saved by paying half of \$1939 (\$969.50) every fortnight, instead of \$1939 every month (the compounding period is still monthly)

N = ??

I% = 4.75

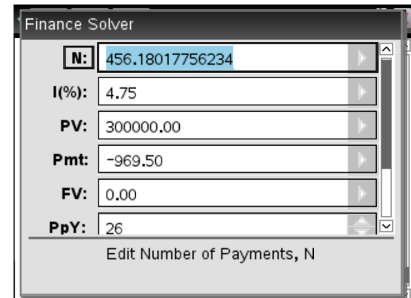
PV = \$300 000

PMT = -1939/2

FV = 0

PpY = 26

CpY = 12



By making this change, the loan can be amortised in approximately 457 fortnightly payments, or approximately 17 and a half years ($457/26 = 17.5769\dots$). So a time saving of about two and a half years.

Footnote: the 'amortTbl' command

Finally, there is a built-in function for generating amortisation tables as a matrix. It returns a matrix with columns in this order: Payment number, amount paid to interest, amount paid to principal, and balance. Its syntax is as follows:

amortTbl(NPmt,N,I,PV,Pmt,FV,PpY,CpY,PmtAt,roundValue)

For example, the command to generate the amortisation table for Question 11 is as follows (see calculator manual for more detail!)

amortTbl(240,240,4.75,300000,-1939,0,12,12,0,2)

Payment number	Interest	Principal	Balance
0	0.00	0.00	300000.00
1	-1187.00	-752.00	299248.00
2	-1185.00	-754.00	298494.00
3	-1182.00	-757.00	297737.00
4	-1179.00	-760.00	296977.00
5	-1176.00	-763.00	296214.00
6	-1173.00	-766.00	295448.00
7	-1169.00	-770.00	294678.00
8	-1166.00	-773.00	293905.00

Teacher notes

- This task attempts to address the process of amortisation from first principles using recursion in a simple case, then using a program and built in commands to make the process more efficient, less tedious and less error prone.
- The first part of the task asks students to formulate a recurrence relation. Some may need guidance with this, depending on their exposure to recurrence relations.
- The issues surrounding rounding and calculating final payments is not trivial. Hence this task tries to 'look under the hood' at the amortisation table, and the various variables as their values change over the life of the loan.
- The program **amortloan** is stored in the 'amortloan.tns' file, and has been locked so that the program code is not modified by accident (or otherwise). It can be unlocked and edited by typing the command **unlock amortloan** (pretty easy huh!).
- Once the program has been run, the variables displayed can be plotted as they just document the changes over the life of the loan. For instance overlaying 2 scatter plots of the change in interest and principal reduction over time can be generated (see right) – useful for discussion
- In the last part of the task, it is assumed that students have already done some work with the TVM Solver. If this is not the case, time will need to be spent working through its mechanics with students.

