freeCard 1.0

Finansportalen.no's interest rate and annuity functions for credit cards

freeCard 1.0 documentation
Version 1.0 25 October 2013
Jon Erland Madsen
Finansportalen.no

Preface

freeCard is Finansportalen.no's financial functions for credit cards. freeCard performs two different computations — one adhering to a simplifyed template, made compulsary in marketing through government regulations, and one that mirrors a more realistic payback stream.

Target group

The primary target group is people who want to integrate the functions in applications (calculators etc).

Furthermore, bank employees responsbible for the accuracy of the credit card provider's product offers as far as effective interest rate and monthly payment is concerned, might be interested in the document.

Covered in the document

This document covers the following areas:

- The many price attributes of credit cards
- The Consumer Ombudsman's "template"
- The freeCard function
- Computation of effective interest rate according to the ombudsman's template
- Computation with a slightly more realistic payback stream

References

Documents relevant to this text

	Title	What	Link
1	freeCard 1.0	The functions connected to a test client	http://jiffymade.com/dev/fp/freeCard.htm
2	"Veiledning til finansavtaleloven § 46"	Guidelines issued by the consumer ombudsman	http://www.forbrukerombudet.no/id/11043105.0
3	"Opplysningsplikt ved markedsføring av kredittavtale"	The law on information when marketing credit	http://www.lovdata.no/all/hl-19990625-046.html#46
4	"Directive 2008/48/EF"	The relevant EU directive	http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2008:133:0066:01:EN:HTML
5	"Forskrift om kredittavtaler m.m"	Public regulation on interest rate computation	http://www.lovdata.no/for/sf/jd/xd-20100507-0654.html#map003

Many price attributes for credit cards

Probabaly, no other financial product have more prices than credit cards. While developing freeCard, Finansportalen identified 22 different pricing attributes applying to credit cards marketed in Norway. Each of the pricing attributes might have three different prices: A fixed fee, a percentage fee and—if the transaction was made abroad—an additional exchange fee.

	Fees per transaction	Fixed and/or Percentage and/or Exchange
1	Cash withdrawal from own bank during opening hours	F P
2	Cash withdrawal from own bank outside normal opening hours	F P
3	Cash withdrawal at own bank's counter	F P
4	Fixed fee for currency withdrawal in own bank's cash machine.	F P E
5	Cash withdrawal in a shop	F P
6	Cash withdrawal from other domestic bank during opening hours	F P
7	Cash withdrawal from other domestic bank outside normal opening hours	F P
8	Cash withdrawal at other bank's counter	F P
9	Cash withdrawal abroad, inside Europe (in addition to an eventual currency fee)	F P E
10	Cash withdrawal abroad, outside Europe	F P E
11	Purchase, domestically	F
12	Purchase abroad inside Europe	F E
13	Purchase abroad outside Europe	F E

14	Fee for transfering cash from credit card to bank account	F
15	Fee when paying bills from the credit card in the internet bank	F
16	Fee when paying with the mobile phone, charging the credit card	F
17	Fee for receiving paper bills in the snail mail	F
	Fees per time unit	
18	Annual fee for holding the credit card, always paid in advance	F P
19	Monthly fee as percentage of credit limit	P
	Other prices	
20	Maximum number of interest free days for puchases	
21	Nominal interest rate for purchases	
22	Nominal interest rate for cash withdrawals	

Even if no credit card provider has so far been observed charging at all these attributes, computing the correct effective interest rate could be a challenge.

The Consumer Ombudsman's "template"

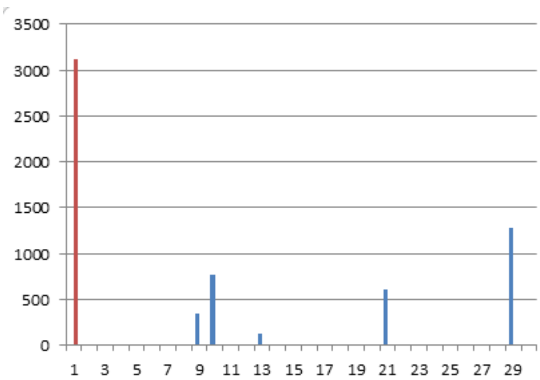
The Consumer Ombudsman enforces a simplified method for computing effective interest rate. This method is compulsory in all marketing of credit cards.

Gry Nergård, the Consumer Ombudsman in 2013.

The Consumer Ombudsman is in charge of monitoring all marketing in Norway. The ombudsman has issued a guideline for how credit cards should be presented in marketing. [The guideline](#) also describes how effective interest rate should be computed.

Although Finansportalen is not marketing loans, it is advantageous to use the same scale of mesurement as the public monitoring body and the banks themselves. Hence, freeCard returns the effective interest rate according to the ombudsman's rules as one of its results.

All charges made the first day



While charges to the card are likely to be spread out in time (the blue columns), the template assumes all are made the first day (the red column)

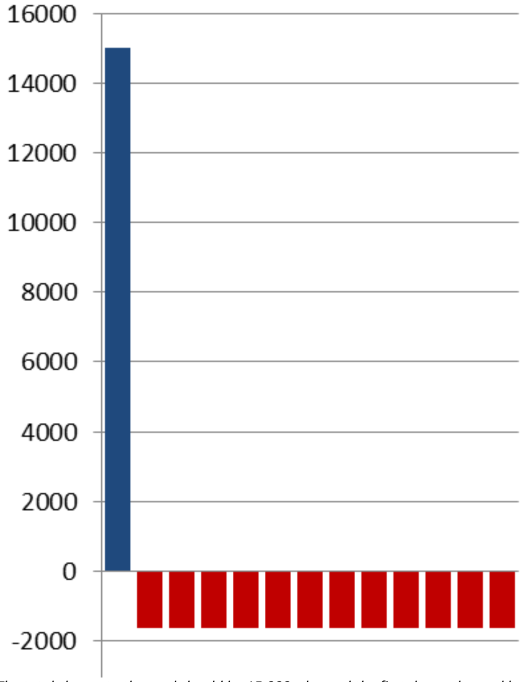
In the ombudsman's model, all charges to the card are assumed made the first day. This applies to both cash withdrawals and purchases.

Even all fees are assumed to be payed the first day.

No further use is assumed. The debt is then paid down to zero in 12 monthly installments of the same size (annuities).

An eventual interest free period or other advantages (cash-back) are to be ignored.

The total debt is 15.000



The total charge to the card should be 15.000, charged the first day and payed back in 12 equal, monthly amounts.

The guidelines are very specific with respect to usage pattern:

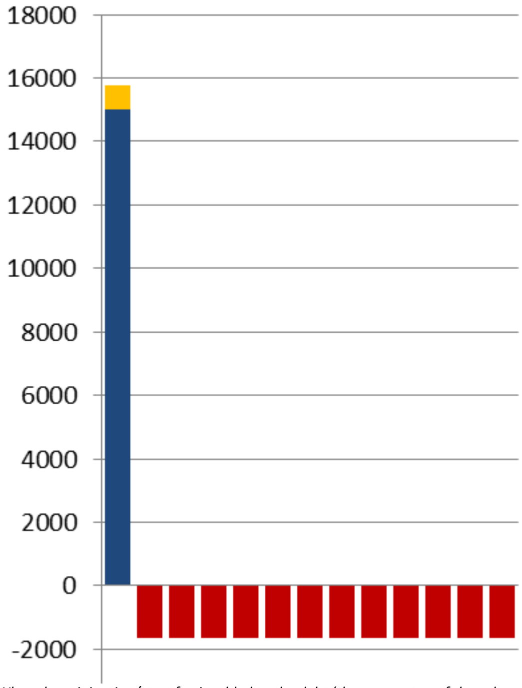
- There should be exactly 15 transactions of exactly 1.000
- 1 should be a domestic cash withdrawal in own bank's cash machine outside banking hours
- 1 should be a cash withdrawal abroad (we assume inside Europe)
- 10 should be domestic purchases
- 3 should be purchases abroad (we assume inside Europe)

This sums up to 15.000

If there is a start fee (origination fee, handling fee etc), your debt will be greater than 15.000. Otherwise, you would not be able to use 15.000.

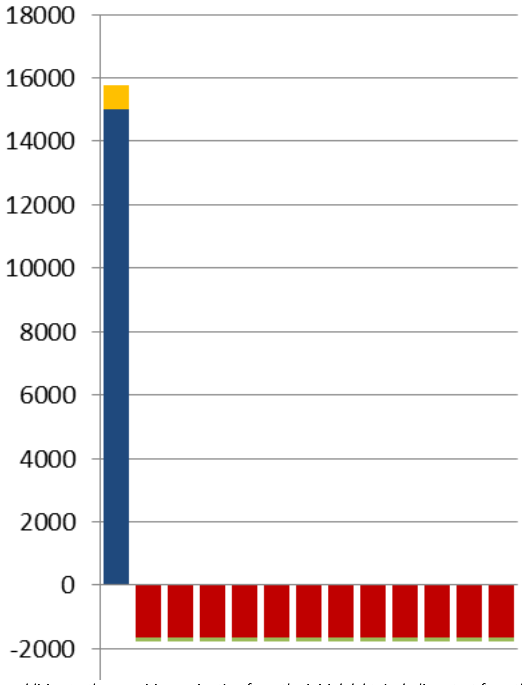
The banks compute the installments on the basis of this gross debt, included origination fee.

We do the same, we add the origination fee before computing the annuity:



When the origination/start fee is added to the debt (the orange top of the column), the monthly payments grow too.

Furthermore, the periodic, monthly term fees must be added to the annuity stream before computing the interest rate:



In addition to the annuities orginating from the initial debt, including start fees, the customer often has to pay a monthly payback fee (the green segments at the bottom of the columns) as well.

Different interest rates for purchases and cash withdrawals

freeCard supports different interest rates for cash withdrawals and purchases, as this is the practice of some banks. In the ombudsman model, this does not create problems. We only have to know what to categorize as what. Generally, freeCard assumes a certain greediness on the part of the banks. As the interest rate for cash withdrawals is normally higher than the interest rate for purchases, we assume that one time start fees are added to the cash withdrawal part of the debt.

Also, transfer from the credit card to the ordinary bank account, or paying bills in the internet bank with your credit card, is considered a cash withdrawal.

Paying with your telephone in a shop, however, is considered a purchase.

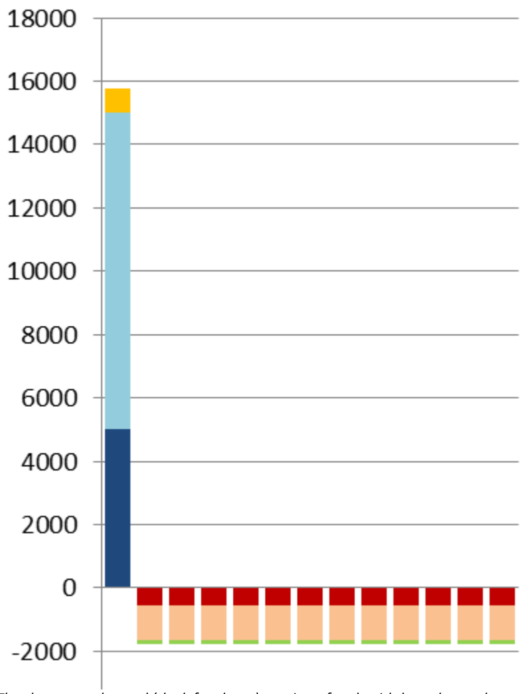
	Type of transaction	**Categorized as**
1	Cash withdrawal from own bank during opening hours	Cash
2	Cash withdrawal from own bank outside normal opening hours	Cash
3	Cash withdrawal at own bank's counter	Cash
4	Fixed fee for currency withdrawal in own bank's cash machine.	Cash
5	Cash withdrawal in a shop	Cash
6	Cash withdrawal from other domestic bank during opening hours	Cash
7	Cash withdrawal from other domestic bank outside normal opening hours	Cash
8	Cash withdrawal at other bank's counter	Cash
9	Cash withdrawal abroad, inside Europe (in addition to an eventual currency fee)	Cash
10	Cash withdrawal abroad, outside Europe	Cash
11	Purchase, domestically	Purchase
12	Purchase abroad inside Europe	Purchase
13	Purchase abroad outside Europe	Purchase
14	Transfering cash from credit card to bank account	Cash
15	Paying bills from the credit card in the internet bank	Cash
16	Paying with the mobile phone, charging the credit card	Purchase
17	Fee for receiving paper bills in the snail mail	Cash *)
	Fees per time unit	
18	Annual fee for holding the credit card, always paid in advance	Cash **)

**) This fee is normally added to the monthly payback fee and not included in the annuity computation*
***) The outlays for the annual fees are assumed borrowed and will thus increase the annuities. If the loan runs for more than one year, the present value of the annual cash stream stemming from the annual fees, will be added to the loan.*

When the two sums for cash withdrawals and purchases are known, they are assumed charged to the card the first day and bearing interest from that moment.

Hence, we simply compute the monthly annuities for each of the two with their separate nominal interest rates.

Then, we sum the two cash streams and add the monthly fees.



The charges to the card (the left column) consists of cash withdrawals, purchases and an eventual start/orgination fee (the latter added to cash). If present, we apply different interest rates to the two parts and obtain two different annuity streams, which we subsequently add.

The freeCard function

The freeCard function computes the effective, annual interes rate for credit card. It takes up to 17 parameters, provides for fees and rounding rules and gives an approximation of the effect of an interest free period.

From the source code:

```
function creditCard(
received_cash,           // EITHER.. The cash withdrawn on the card during the period, included all the transaction dependent fees.
received_purchase,       // ..AND/OR: The purchases done with the card during the period, included all the transaction dependent fees.
numberofmonths,          // OBLIGATORY.
interestfree_days,       // 0, 'false' or omitted: zero. Integer > 0: The initial interest-free period (only applied to purchases) in
round_direction,         // 0, 'false' or omitted: Annuities are rounded after normal rules  1: Rounded up 2. Rounded down
round_presision,         // 0, 'false' or omitted: Payment rounded to nearest 1/100  1: ..rounded to nearest integer
remainder_handling,      // 0, 'false' or omitted: The "global" remainder at the end of the loan period is payed / compensated with th
ignore_origination,      // 0, 'false' or omitted: Origination fee added to the loan and included in the computation. 1 or true: Compu
fee_cash_transaction,    // 0, 'false' or omitted: 0. Number >= 0: Sum of all transaction fees for cash withdrawals
fee_purc_transaction,    // 0, 'false' or omitted: 0. Number >= 0: Sum of all transaction fees for purchases
fee_origination,         // 0, 'false' or omitted: 0. Number >= 0: Processing fees: Sum of eventual inital one-time fees to be payed a
fee_annual,              // 0, 'false' or omitted: 0 .Number >= 0: Fixed fee: Annual, fixed fee.
fee_period,              // 0, 'false' or omitted: 0 .Number >= 0: Fixed fee. Loans given as a credit line might have a periodical (mc
rate_cash,               // OBLIGATORY: Interest rate for cash withdrawals might differ from interest on purchases
rate_purchase,           // OBLIGATORY: Interest rate for purchases might differ from interest on cash withdrawals
minpay_perc,             // The minimum, monthly payment as a percentage of current debt
minpay_units)            // The minimum, monthly payment in currency units (for instance NOK).


RESULT

The function returns an array; 'res':

res[0] -> Effective, annual interest rate without taking an eventual interest free period into account ("government template")
res[1] -> The monthly payment, when we apply the "government template"
res[2] -> Effective, annual interest rate when taking an eventual interest free period into account
res[3] -> The monthly payment when we take the interest free period into account
res[4] -> The remainder


The function handles only monthly payments and only annuity loans in arrears (annuity-immediate, or ordinaray annuities).


SECTIONS

1)      Preparation and adaption of data
2)      The annuities
3)      Effective interest rate
4)      Result reporting


EXTERNAL FUNCTIONS USED

function roundoff()
function rateAnnuity()
```

THE PLAN: THREE ANNUITY SERIES

The interest rate is computed by iterations. Before performing the iterations, we need to know each monthly payment.

Hence, all the relevant Payments, number of periods and remainders are first computed.

There might be different interest rates for cash withdrawals and purchases. In the case of interest free days, they usually only pert
purchases, not to cash withdrawals. The two streams might have to be treated differently.

Moreover, there are annual fees not fitting into the model with monthly annuities.

We thus make three annuitiy series:

1) The annuities paid back to the card company for cash withdrawal
2) The annuities paid back to the card company for purchases
3) The annuity series consisting of annual fees

As there is often an interest free period for purchases, but not for cash withdrawals, this model also allows us to compute a more re
interest rate, in addition to the the one produced by the model given in the ombudsman's regulation.

ERROR CODES:

When the computed payment is smaller than the minimum payment reqired in the parameters 'minpay_perc' and 'minpay_payment', the loan
paid back faster than the user has requested. It is less confusing to ask the user to select a shorter payback period than to first c
minimum payback time and then compute the effective interest rate. Hence, in these cases we return an error code:

-1 => With the chosen payback time, the annuity will fall below the required minimum payment.

Two computations

In addition to computing the interest rate and the monthly payment according to the ombudsman's rules, freeLoan also returns an interest rate and a monthly payment that to some extent takes an eventual interest free period into account.

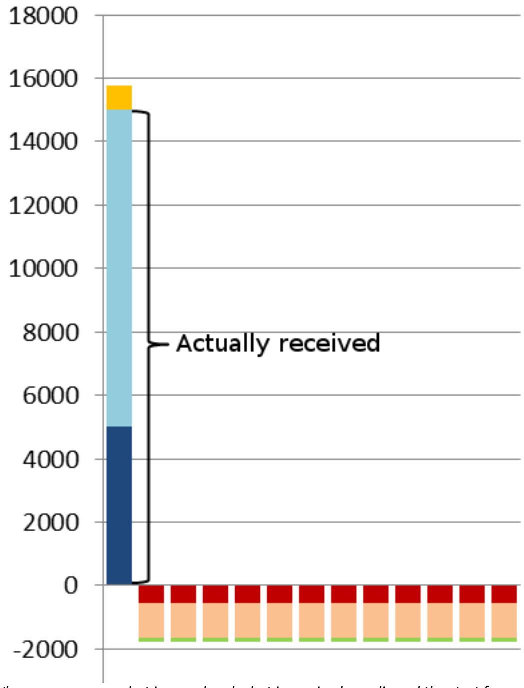
Essentially, the ombudsman' model presumes an unrealistic charge pattern and an unrealistic payback stream (it is still very well suited to comparision).

freeCard's alternative model still presumes an unrealistic charge pattern, but a more realistic payback stream. Even this computation, thus, deviates from the results that would transpire from normal usage.

The general procedure is shown below. The details are describes in freeLoan's source code.

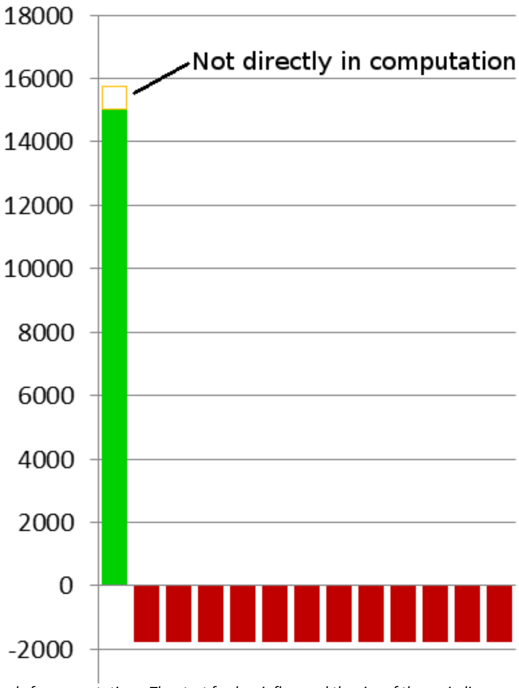
Computation of effective interest rate according to the ombudsman's template

Simply put, the effective interest rate is the relation between what we receive and what we pay. In the figure below, we pay all the outlays along the timeline fully. But we don't receive the whole column to the left. We never receive the start fee. The bank took the money and added it to our debt. So what we receive is the cash we have withdrawn and the påurchase of the goods and services we bought with the card.



When we compare what is payed and what is received, we discard the start fee.

As a matter of illustration, we colored the fees, the purchases and the cash withdrawals differently. But when we know the sizes of the different sums, money is money. We just sum them.



Ready for computations: The start fee has influenced the size of the periodic payments, which are to be compared to what we received. We did not receive the fee.

The above point is the source of many erroneous computations of effective interest rate: The bank will compute the annuities out of the whole debt, whether it is incured by fees or the customer actually using the card.

The orgination/start fee will thus increase the monthly payments.

But the effective interest rate is computed only on the basis of "visible" payments: What you actually pay to service your debt, versus what you received when you used your card.

The following is taken from comments in freeLoan's source code:

```
¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤
//////////////////////////              DERIVATION OF A FORMULA FOR ANNUITY-IMMEDIATE          ////////////////////////////////////
¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤


ANNUITY IMMEDIATE (OR ORDINARY ANNUITIES) PAID AT THE END OF EACH PERIOD

"An annuity is a terminating "stream" of fixed payments, i.e., a collection of payments to be periodically made over a specified peri
time" (Wikipedia).

The fixed payments contain both an interest part and an installment part, but as the loan is gradually repaid on each interest due da
installment share of each annuity grows and the interest share shrinks.

"If the payments are made at the end of the time periods, so that interest is accumulated before the payment, the annuity is called a
annuity-immediate, or ordinary annuity "(Wikipedia).

The derivation below applies to ordinary annuities.


NOTATION / ASSUMPTIONS


It will be easier to read the expressions when we use the short form of the variable names:


|********************************************************************************|
|                                | Symbol          | Variable name          |
|________________________________|_________________________|________________________|
| Loan, original principal *    | PV                      | loan                   |
| Nominal, term interest rate   | r                       | rate                   |
| Number of payments            | n                       | termnumber             |
| Periodical payment            | a                       | annuity                |
|                                |                         |                        |
|********************************************************************************|


* The bank computes the annuity out of your original debt to the bank—the principal at the start of the loan. This might deviate
from the amount you actually receive. Fees could make up the difference.

The present value of a loan is smaller than the sum of your payments, as a dollar paid later feels less painful than one paid now. Th
borrowers are willing to pay more back than they receive. Lenders are willing to wait for their money, provided they receive more lat

The word "annuity" implies annual payments. But monthly payments are more common now, and "math-wise" the length of the periods is ir
All that we demand, is that the periods have the same length (or, to be very precise—that the periodical interes rate is the same for
periods).

If the annual interest rate is 5%, the present value of an annuity 'a' payable in one year is PV = a/1,05
```

The present value of this annuity plus a similar annuity payable in two years will be PV = a/1,05 + a/(1,05)^2

We use '^' as notation for "in the power of", as in Excel spreadsheets.

(This math procedure, dividing each term with a growth factor, is called "to discount").

We imagine the interest rate 'r' on decimal form, so that five percent is written 0,05. Generally, the growth factor is written '(1+r

We presume annuity-immediate, so that the first payment is made at the end of the first period. Hence, we have no zero term, as you often have in Math books' examples.

ANNUITY FORMULA

We formulate the sequence of payments thus, saying that the present value is the sum of all the discounted payments. In this general case, we don't know exactly how many terms there are, so we leave a gap in the middle:

```
I       PV = a/(1+r) + a/((1+r)^2) + a/((1+r)^3) + .. + a/((1+r)^n)
```

All terms on the right hand side have a common growth factor, '1/(1+r)'. In order to make the expressions more easy to read, we intro the variable 'k':

```
k = 1/(1+r)
```

We substitue 'k' for '1/(1+r)':

```
I       PV = a*k + a*k^2 + a*k^3 + .. + a*k^n
```

We introduce a second equation, similar to the first, but where we have multiplied all terms by 'k':

```
II      PV*k = a*k^2 + a*k^3 + a*k^4 + .. + a*k^(n+1)
```

We subtract II from I. Many of the terms of the two sequences are the same and disappear. We are left with:

```
PV - PV*k = a*k - a*k^(n+1)
```

This expression can be rearranged, here with respect to the annuity 'a':

```
=>      PV (1 - k) = a (k - k^(n+1))

=>      PV = a (k - k^(n+1))/(1 - k)

=>      a = PV (1 - k)/(k - k^(n+1))
```

In javascript notation:

```
a = PV * (1 - k) / (k - Math.pow(k,n+1));
```

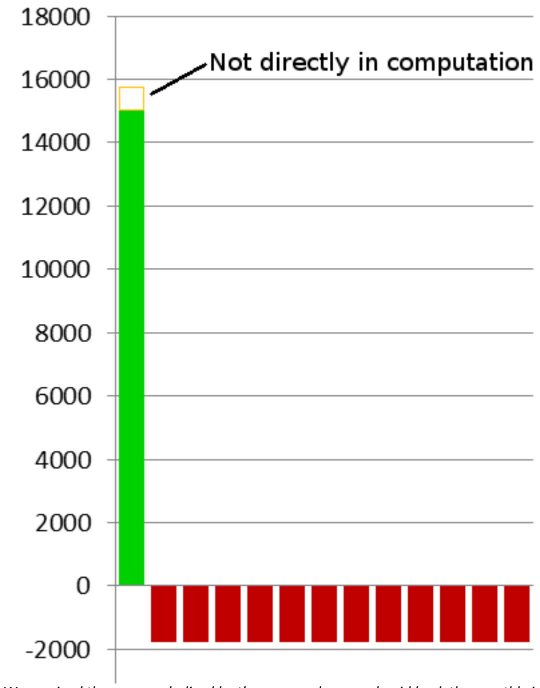
WHen we subsitute the symbols for the variable names in the table above, we get:

```
annuity = loan * (1 - k) / (k - Math.pow(k,termnumber+1));
```

```
*********************************************************************************
|                                                                               |
|       FORMULA FOR ANNUITY-IMMEDIATE:                                          |
|                                                                               |
|                                                                               |
|       annuity = loan * (1 - k) / (k - Math.pow(k,termnumber+1));              |
|                                                                               |
|                                                                               |
|       where                                                                   |
|                                                                               |
|                                                                               |
|       k = 1/(1+rate);                                                         |
|                                                                               |
|                                                                               |
*********************************************************************************
```

```
¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤
//////////////////////////              (END—DERIVATION OF A FORMULA FOR ANNUITY-IMMEDIATE)              ///////////////////////////
¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤
```

Let's look again at the graph we made:



We received the sum symbolized by the green column and paid back the monthly installments symbolized by the red.

The red columns are our annuities, 'a'. The green column is our present value, 'pv'.

Applying the formula:

```
*********************************************************************************
|                                                                               |
|       FORMULA FOR ANNUITY-IMMEDIATE:                                          |
|                                                                               |
|                                                                               |
|       annuity = loan * (1 - k) / (k - Math.pow(k,termnumber+1));              |
|                                                                               |
|                                                                               |
|       where                                                                   |
|                                                                               |
|                                                                               |
|       k = 1/(1+rate);                                                         |
|                                                                               |
|                                                                               |
*********************************************************************************
```

As the model accomodates two different interest rates, we also get two different discount factors, one for purchases and one for cash withdrawals. Let's name these discounting factors 'k_p' and 'k_c', respectively. The corresponding annuities could be called 'a_p' and 'a_c':

a_p = pv_p * (1 - k_p) / (k_p - Math.pow(k_p,termnumber+1))

a_c = pv_c * (1 - k_c) / (k_c - Math.pow(k_c,termnumber+1))

The values of the right hand symbols are all known:

k_c = 1 / (1+r_p)

k_p = 1 / (1+r_c)

r_c : The monthly interest rate for cash withdrawals in decimal form

r_p : The monthly interest rate for purchases in decimal form

pv_p : The sum of all purchases, as defined above

pv_c : The sum of all cash withdrawals, as defined above. (We also add annual fees to this sum).

termnumber : The assumption in the government model is 12 months.

Now, we only sum the two annuities. Their nominal values are now known. They are paid at the same time, with the same frequenzy etc, so we can sum them and determine the annuity. To find the monthly payment, we must add an evental monthly fee:

a = a_p + a_c + monthlyfee

We now can look back to the formula:

annuity = loan * (1 - k) / (k - Math.pow(k,termnumber+1));

With our symbols:

a = (pv_c+pv_p) * (1 - k) / (k - Math.pow(k,termnumber+1));

We know the value of all symbols but the discount factor 'k', in which the effective interest rate is hidden. Unfortunately, there is no simple way to retrive 'k' directly. It has to be done by guessing a value, and then another one and another one. It could be quite many such rounds, or iterations, but luckily we have Isaac Newton:

```
NEWTON'S METHOD - DIFFERENTIATION

No formula gives us the effective interest rate directly. We find it by trying a likely value for 'k', adjusting it until we get the
rate with the accuracy we want. Newton's method shortens this try and fail process. It postulates that our 'k' will be found close tc
tangent to the curve in the present PV/k point crosses the k-axes:

http://en.wikipedia.org/wiki/Newton%27s_method

The graph is in the PV/k space. A value of 'k' will give a value of 'PV': PV = f(k).

PV = (payment/(1-k)) * (Math.pow(k,interval_start+1) - Math.pow(k,interval_end+1));

What we have charged to the card, pluss fees, we will collect in the variable 'received'.

We are searching for a 'k' that makes 'received' equal to 'PV':

received = (payment/(1-k)) * (Math.pow(k,interval_start+1) - Math.pow(k,interval_end+1));

=> (payment/(1-k)) * (Math.pow(k,interval_start+1) - Math.pow(k,interval_end+1)) - received = 0

We want a continuous function, not only one that is valid in '0'. Hence, we call the function 'y':

y = (payment/(1-k)) * (Math.pow(k,interval_start+1) - Math.pow(k,interval_end+1)) - received

When differentiating, we see that the variable 'received' will have a differentiated that equals 0.

So we constentrate on differentiating the other terms.

For easier differentiation, we handle the two expressions on each side of the multiplication sign separately. We call them 'A' and 'E

y = A*B; where

A = payment/(1-k);

B = Math.pow(k,interval_start+1) - Math.pow(k,interval_end+1);

One can find the differentiation rules on wikipedia: http://en.wikipedia.org/wiki/Differentiation_rules

y' = A'*B + A*B'

We fist differentiate 'A' separately according to the Quotient rule:

A' = (payment' * (1-k) - payment* (1-k)') / (1-k)^2

The differntiated of the constant 'payment' is 0. The differentiated of the variable 'k' er 1:

=> A' = - payment* (1-k)' / (1-k)^2

=> A' = payment / Math.pow(1-k,2)

Then we differentiate 'B' according to the Power rule (y = x^n => y' = n*x^(n-1)):

B = Math.pow(k,interval_start+1) - Math.pow(k,interval_end+1);

B' = (interval_start+1)*Math.pow(k,interval_start) - (interval_end+1) * Math.pow(k,interval_end)

We then assemble the whole differentiated PV according to the Product rule:

y' = (payment / Math.pow(1-k,2)) * (Math.pow(k,interval_start+1) - Math.pow(k,interval_end+1)) + (payment/(1-k))*((interval_start+1)*

As there is no notation for the differentiated in the programming language, we rename the differentiated y' -> 'PV_dif':

PV_dif = y'
```

The iterations now start with a guess for the effective discount factor 'k'. The guess is derived from then nominal interest rate.

The function value 'y' is then computed. It will probably be some distance from '0', so we need a better guess for 'k'.

The best guess is obtained by calcualting the tangent inclination of the 'y' funtcion graph, follow the tangent to the axes and find the new guess for 'k' here.

The inclination of the tangent is the differentiated 'y' — y' — given the current guess for 'k'.

Normally, we only have to repeat this process four times in order to obtain a 'k' with the necessary accuracy.

This computation is shown as comments to the source code of 'function freeLoan()'.

Model with a slightly more realistic payback stream

While the Consumer Ombudsman's model ignores the interest free period often offered for purchases, freeLoan also computes an effective interest rate where this is taken into consideration.

This model is only slightly more realistic than the ombudsman's: As in the latter, all charges to the card are assumed to be made the first day. But the payback stream starts after 45/50 days, depending on the bank's payment schedule. Furthermore, if the bank refrains from charging interest on purchases during this period (as is the case for most credit cards), the model accomodates it.

How to incorporate the interest free period

In the model, the debt is not fully refunded within the interest free period. Only a part is paid in the first term, hence, interest is accrued after the first payment. But unlike the ombudsman's template, where annuities in arrears (annuity-immediate, ordinary annuities) are presumed, here no interest is accrued before the first payment. So we use the model for payments in advance instead, the formula for which is derived in the source code:

```
¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤
//////////////////////////              DERIVATION OF A FORMULA FOR ANNUITY-DUE                 ///////////////////////////////////
¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤


ANNUITY-DUE - WHERE THE ANNUITIES ARE PAID IN ADVANCE - AT THE BEGINNING OF EACH PERIOD

Important: This is NOT a formula for interest paid in advance. For definitions, see derivation of annuity-immediate. The difference f
annuity-immediate is that the first period is period zero (the moment we receive the loan). There is no interest on this first paymen


ANNUITY FORMULA


PV -> The present value of an annuity-due payment stream (when we compute effective interest rate, the present value is the initil si
a  -> The annuity in an annuity-due stream
r  -> The periodic interest rate
n  -> Number of payment periods

We formulate the sequence of payments thus, saying that the present value is the sum of all the discounted payments. In this general
case, we don't know exactly how many terms there are, so we leave a gap in the middle. The first payment shall not be discounted, as
has passed and no interest accured yet.

In order for the loan to run 'n' periods, since we start in period zero, the last payment happens in period 'n-1':


I       PV = a + a/(1+r) + a/((1+r)^2) + a/((1+r)^3) + .. + a/((1+r)^(n-1))


All terms on the right hand side have a common growth factor, '1/(1+r)'. In order to make the expressions more easy to read, we intro
the variable 'k':

k = 1/(1+r)

We substitue 'k' for '1/(1+r)':


I       PV = a + a*k + a*k^2 + a*k^3 + .. + a*k^(n-1)


We introduce a second equation, similar to the first, but where we have multiplied all terms by 'k':


II      PV*k = a*k + a*k^2 + a*k^3 + a*k^4 + .. + a*k^n


We subtract II from I. Many of the terms of the two sequences are the same and disappear. We are left with:


I - II  PV - PV*k = a - a*k^n


This expression can be rearranged:


=>      PV * (1 - k) = a * (1 - k^n)

=>      PV = a * (1 - k^n)/(1 - k)

=>      a = PV * (1 - k)/(1 - k^n)


In javascript notation:


PV = a * (1 - Math.pow(k,n))/(1 - k);

a = PV * (1 - k) / (1 - Math.pow(k,n));


*********************************************************************************
```

```
|                                                                          |
|       FORMULA FOR ANNUITY-DUE:                                           |
|                                                                          |
|                                                                          |
|       annuity = loan * (1 - k) / (1 - Math.pow(k,termnumber));    |
|                                                                          |
|                                                                          |
|       where                                                              |
|                                                                          |
|                                                                          |
|       k = 1/(1+rate);                                                    |
|                                                                          |
|                                                                          |
***************************************************************************
```

```
¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤
/////////////////////////             (END—DERIVATION OF A FORMULA FOR ANNUITY-DUE)            //////////////////////////////////
¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤¤
```

To clearify the following example, let us presume that the monthly payments in this bank are due 15 days after the end of the month. Banks normally market this as "up to 45 days of zero interest".

At this moment, we pay:

— The annuity pertaining to the interest free part of the loan, computed with the annuity due-formula and day 45 set as day zero

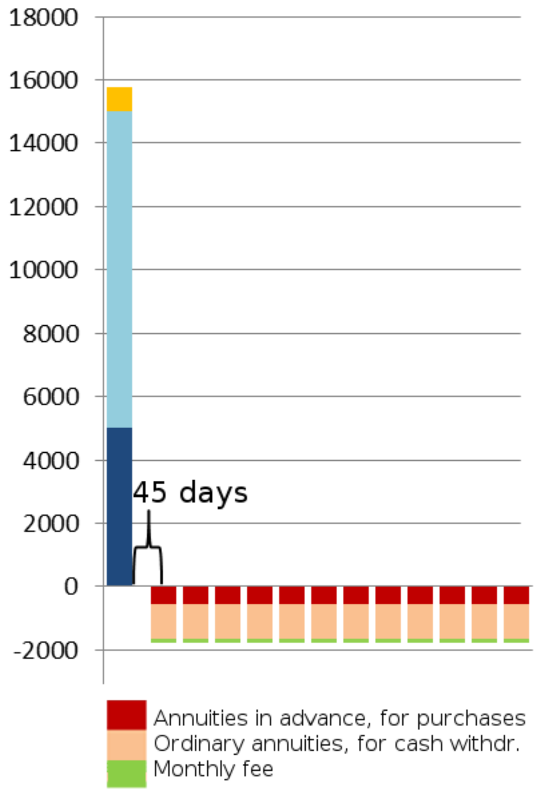
— The annuity pertaining to the interest bearing part of the loan, computed with the annuity-immedidate formula and day 15 as day zero :)

In our formula for ordinary annuities (anniuty-immediate), the first payment is due one month — 30 days — after the first day of the loan. If we compute this annuity after 15 days, we find the amount to be paid after 45 (15 + 30) days. (Remember that we also assume eventual annual fees to bear the same interest as cash withdrawals. So they are added here).

Correspondingly, we compute the payment in advance (annuity-due) amount, pertaing to the interest free amount, after 45 days.

Hence, we have obtained that the two payment streams have the same due dates during the payback period.

We add an eventual monthly fee, obtaing the nominal payment due after 45 days and thereafter every month (every 1/12 year)



Instead of payments one month after the start, the monthly payment stream starts after 45 (or sometimes 50) days. It consists of annuities in advance, pertaining to purchases and ordinary annuities pertaining to cash withdrawals.

The efficient interest rate is the rate that makes the present values of what we receive the same as the present value of what we pay.

In our model, it is easy to find the present value of what we receive, as all charges to the card are assumed to be made the first day. The present value, thus, is simply the nominal sum of the charges.

But we have no simple formula to compute the present value of a payment stream that starts in 45 days and then runs with a frequency of every 30 days.

So we make a little twist: We first use the normal present value formula for annuities in arrears with the first payment after one month.

This is the present value after 15 days. So we have to discount it by another half month / 15 days. This is rather straight forward, we only have to remember that to use the same interest rate — the effective interest rate — in both discountings.

Again, we look in freeCard's source code:

```
If something grows by 1,5% each month, how much does it grow by in one day?

If we assume daily capitalization:

        (1 + rd) = (1 + r) ^ (1/30)                     // The daily growth rate is the 30th root of the monthly growth rate

Discouting means dividing by the growth factor, or multiplying by 1 / (1 + r). We call this factor k:


        As k = 1 / (1 + r) => (1 + r) = 1/k             // The monthly growth factor

        (1 + r)^(1/30) = (1/k)^(1/30)                   // The daily growth factor


We now have the annuity stream. We first find the preliminary present value as such:

        pv_p = annu * (k - Math.pow(k,numberofmonths+1)) / (1 - k)

With its monthly scope, this is the present value one month ahead of the first payment.

But the payment stream in reality starts later.

        extra_days = due_days - 30              // The number of days we shall discount the computed present value by

We will first compute the growth factor for the extra days. It is the daily growth factor, '(1/k)^(1/30)', raised by the
number of extra days:

        extra_growth = ((1/k)^(1/30))^extra_days

I follows as normal computation rules that (a^m)^n = a^(m * n), Thus..

        extra_growth = (1/k)^(1/30 * extra_days) = (1/k)^(extra_days/30)

The discount rate is the inverse of the growth rate: 1/ (1 + r) = k

Hence, we make a new discount rate, ke,  for the 'extra_days' number of days:

        ke = 1 / ((1/k)^(extra_days/30))

Or, in Javascript:

        ke = 1 / Math.pow(1/k,extra_days/30)


We still maintain the assumption that all charges to the card are made the first day. Thus, the present value equals
the nominal, received sum.

The annuity formula gives us the payment stream that starts 30 days after we get the card. But the due date is slightly later,
normally 15-20 days into the next month. The bank will charge interest up to payment day, so we must discount the preliminary
present value we found, 'pv_p', with these extra days to obtain the present value.

received = pv_p * ke            // Discounted with the extra days. We substitute pv_p with its formula:
```

Thre rest is in the source code. It is quite straightforward: We formulate the dicounting formula with respect to the discounting factor 'k' (which in turn is a function of the effective interest rate).

We then proceed to finde the differentiated of this function, so we can apply Newtons method and perform the iterations.