TI-CF

A software Application
On TI-89, TI-92 Plus and Voyage 200

User Manual

Feb. 2005

Rev. 1.0
TI-CF 1.0

http://ti-cf.gveto.com

About TI-CF

TI-CF is an application designed to solve single variable cash flow diagram (CFD) problems in a command line, function based manner. TI-CF was written in C and assembled for the TI-89, TI-92 Plus, and Voyage 200 calculators with TIGCC. TI-CF requires at least AMS 1.01 to run. TI-CF does not require a kernel to run.

Current Capabilities

Currently, TI-CF can populate a cash flow diagram with given flow amounts and the coefficients of the variable to be solved at any period. The program has routines to fill the cash flow arrays with standard series such as uniform series (A), geometric series (g), and other common series. The cash flow diagram interest can be entered at any time as a constant interest per period, compounded *n* times per period, or compounding continuously. TI-CF operates on one cash flow diagram at a time.

Operation

TI-CF is a command line program based on functions and arguments. The generic form of function entry is:

```
CF>func(arg1, arg2, ... , argn)
```

Where *func* is the name of the function to execute, and *arg1 – argn* are the passed arguments to that function. Note that functions that do not receive arguments still must be followed by an empty parenthesis pair.

Example:

To create a new cash flow (CF) with 52 periods, use the following command:

```
new(52)
```

To print the values of the current CF, use the following command:

```
print()
```

Note that CF periods begin with period 0, not 1. So, the above *new* command would create a CF of 52 periods from period 0 to period 51. TI-CF assumes that all given cash flows are on the opposite side of the cash flow array to the variable coefficients to be solved. For a cash flow involving both receipt and disbursement given cash flows, the receipts should be entered as positive values in the given CF array and the disbursements as negative values.

Functions

The following is a list of functions used by TI-CF and the associated usage and arguments.

Key:	
`[int]`	is short for integer which simply means a whole number.
`[float]`	is a number that can have a decimal value, though it doesn't have to.
`[func]`	is short for function, and refers to the name of a TI-CF function.

`a([int] start, [int] stop, [float] a)`

Adds a uniform series in the given CF array starting at period *start* and ending at period *stop* of value *a*. Note that by as defined by a uniform series, the period *stop* is included in the series, while *start* is not.

`xa([int] start, [int] stop, [float] a)`

Adds a uniform series in the variable coefficient array starting at period *start* and ending at period *stop* of value *a*. Note that by as defined by a uniform series, the period *stop* is included in the series, while *start* is not.

`exit()`

Exits the TI-CF session.

`g([int] start, [int] stop, [float] G)`

Adds an arithmetic gradient to the given CF array starting at period *start* and ending at period *stop*. *G* is the uniform period-by-period increase or decrease.

`xg([int] start, [int] stop, [float] G)`

Adds an arithmetic gradient to the CF variable coefficient array starting at period *start* and ending at period *stop*. *G* is the uniform period-by-period increase or decrease.

`geo([int] start, [int] stop, [float] g)`

Adds a geometric gradient to the given CF array starting at period *start* and ending at period *stop*. *g* is the uniform rate of cash flow increase or decrease.

`xgeo([int] start, [int] stop, [float] g)`

Adds a geometric gradient to the CF variable coefficient array starting at period *start* and ending at period *stop*. *g* is the uniform rate of cash flow increase or decrease.

`info()`

Prints the current CF size and interest rate.

`new([int] n)`

Creates a new CF with *n* periods. The *new* function automatically deletes the old CF array (if it exists) and initializes a new one with default values of 0 for both the given cash flow and the variable coefficient array without prompting the user. Since *n* is the size of the CF, a CF allocated with *n = 5* will have periods 0 – 4. This is important since any command that refers to period 5 in this example will cause an error since there is no period 5.

`s([int] n, [float] value)`

Adds a single value of *value* into the given CF array at period *n*.

`xs([int] n, [float] value)`

Adds a single value of *value* into the variable coefficient array at period *n*.

`print()`

Prints the current cash flow values separated by commas starting at period 0.

`solve()`

Solves the current CF problem and outputs the unknown variable X. *solve()* does not affect the current CF arrays.

```
int([float] i)
int([float] r, [float] m)
int([float] r, c)
```

As indicated, there exists 3 usages for the *int* function. First, *int* called with a single argument simply sets the CF interest rate to i%. Note that the interest rate is entered is in percent, not its decimal equivalent. For instance, for an interest rate of 10%, one would enter 10 for *i* rather than .1 .

int also has support for compounding interest. When called with two arguments, the first argument is *r*, the nominal interest rate per period. The second argument is *m*, the number of compounding subperiods per period. The CF interest is then calculated and set as the effective interest rate per interest period.

Finally, to set the interest to a continuously compounding effective rate, pass a first argument of *r*, the nominal interest rate per period, and the letter 'c' to indicate compounding continuously. The CF interest rate will be set to the effective interest rate per period.

Example:

To set the interest rate to 8% per period, use the command:

```
int(8)
```

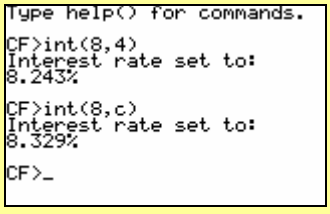
When dealing with periods that correspond to years, use the following command to set the interest rate to 8% compounded quarterly:

```
int(8,4)
```

To set an interest rate of 8% nominal compounded continuously, use the command:

```
int(8,c)
```

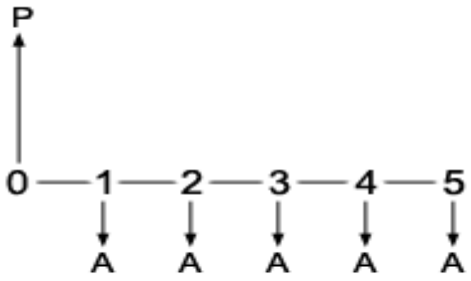
The latter two commands return the following messages, so it is easy to see how TI-CF can be used to calculate effective interest rates:



Examples

1. Consider a situation in which you borrow $5000. You will repay the loan in five equal end-of-the-year payments. The first payment is due one year after you receive the loan. Interest on the loan is 8%. What is the size of each of the five payments?

Put into a cash flow diagram, the problem is easy to solve with TI-CF:



P = $5000 n = 5 i = 8% A = unknown

First, we set the size and interest rate of the CF:

```
new(6)
int(8)
```

Let's use the *info* function to verify our entries:

```
info()
```

```
CFD of size 6 created.

CF>int(8)
Interest rate set to:
8%

CF>info()
CF size: 6
CF interest: 8%

CF>_
```

So far so good. The next step is to populate the CF with the given information. First, the single receipt:

```
s(0,5000)
```

then the variable uniform series:

```
xa(0,5,1)
```

Notice that the coefficients were all set to 1. This will be the case when all only one unknown period exists, or as here when all the unknown periods are of the same unknown value.

To be on the safe side, let's check our CF before solving:

```
print()
```

```
A series added.

CF>print()

CF:
5000,0,0,0,0,0

Variable coeficients:
0,1,1,1,1,1

CF>_
```

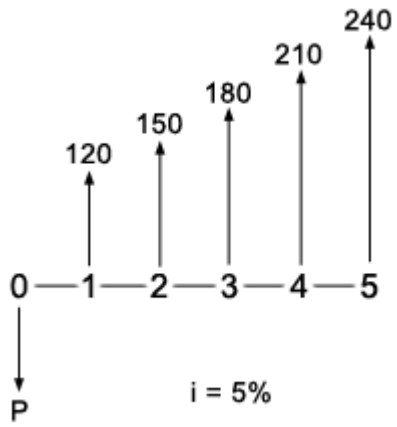
Everything's in order. Finally, use the *solve* function to solve for the unknown:

```
solve()
```

```
CF:
5000,0,0,0,0,0

Variable coeficients:
0,1,1,1,1,1

CF>solve()

X = $1252.28

CF>_
```

TI-CF gives us an answer of $1252.28.

2.



The given CF can be broken down into the sum of an arithmetic gradient series and a uniform series. Note that functions that operate on the two arrays in TI-CF add the new values in to the existing values, rather than replacing them. The problem is entered as follows:

```
new(6)
int(5)
a(0,5,120)
g(0,5,30)
xs(0,1)
solve()
```

```
CF>g(0,5,30)
6 series added.

CF>xs(0,1)
Single value added.

CF>solve()

X = $766.64

CF>_
```

The answer is given to be $766.64.

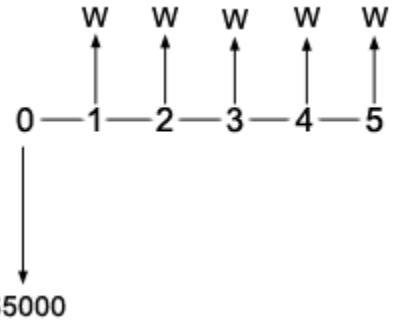
3. On January 1, a woman deposits $5000 in a credit union that pays 8% nominal annual interest, compounded quarterly. She wishes to withdraw all the money in five equal yearly sums, beginning December 31 of the first year. How much should she withdraw each year?

While this problem could be solved using 20 periods with i = 2% per period, and withdraws every 4 periods, let's make use of TI-CF's ability to use compounded interest:

```
new(6)
int(8,4)
```

```
TI-CF by Dan Hauer
Type help() for commands.

CF>new(6)
CFD of size 6 created.

CF>int(8,4)
Interest rate set to:
8.243%

CF>_
```

TI-CF has computed the effective yearly interest rate to be 8.243%. Note that internally, TI-CF has computed this value to many more decimal places than the displayed 3 for the most accurate results. The cash flow diagram is then:



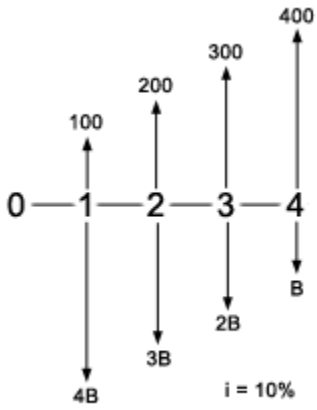
Populating the arrays and solving:

```
s(0,5000)
xa(0,5,1)
solve()
```

```
CF>s(0,5000)
Single value added.

CF>xa(0,5,1)
A series added.

CF>solve()

X = $1260.32

CF>_
```

Note that the unknown values are on top of the CFD, but this is of no consequence as long as they are on the opposite side to the given cash flows.

4.



```
new(5)
int(10)
g(-1,4,100)
xa(0,4,4)
xg(0,4,-1)
solve()
```

Note: For G, A, and g series, a *start* value of -1 is acceptable, but -1 is the lowest valid value for *start*. When entering negative number, be sure to use the sign key (−) and not the subtraction key.

Future Ambitions

Possible future additions may include more detailed in-calculator help, and the ability to solve CFD problems for the interest rate or the number of periods. A more esoteric addition may be the implementation of a graphical view of the CFD, and possibly graphical editing.

Disclaimer

TI-CF is provided in good faith, but is supplied "as is" without any warranty of any kind. Neither the programmer, nor any distributor or ISP shall be responsible for any claims attributable to errors, omissions or other inaccuracies in the programs or documentation. The entire risk as to the results and performance of the program is assumed by the user. Neither the author or any distributor or ISP make any representations or warranties, either express or implied, with respect to the software, including but not limited to, the quality, performance, or fitness for a particular purpose. In no event shall the author or any supplier be liable for direct, indirect, special, incidental, or consequential damages arising out of the use of or inability to use the software or for any loss or damage of any nature caused to any person or property or data as a result of the use of the program, even if the author or supplier have been specifically advised of the possibility of such damages.

I have used TI-CF to successfully solve many CFD problems, but I cannot guarantee it to be bug-free. It is always a good idea to archive any files you may want to keep before executing any ASM program. If you find a bug or have a general question, request, or comment, please email me at admin@gveto.com or visit http://ti-cf.gveto.com for information on updated versions.

Dan “Gveto” Hauer