THE DERIVE - NEWSLETTER #91

ISSN 1990-7079

THE BULLETIN OF THE

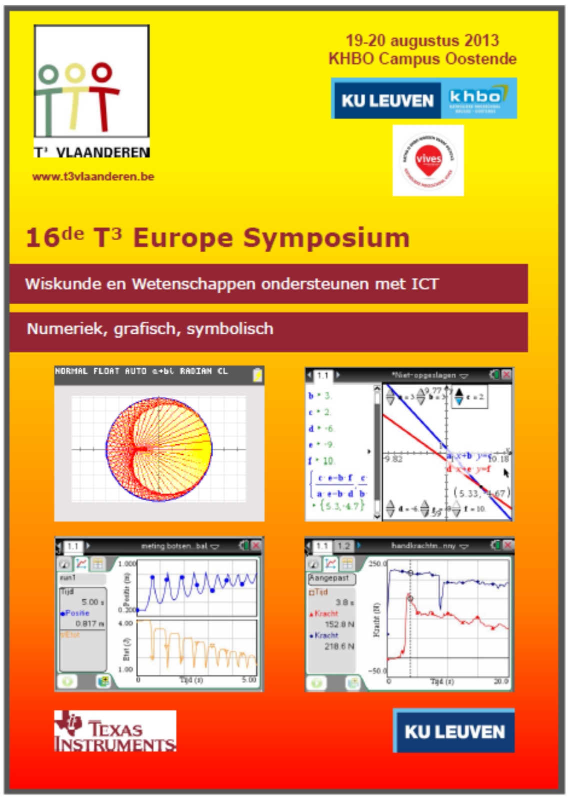
DERIVE®

USER GROUP

+ CAS-TI

Contents:

September 2013


19-20 augustus 2013
KHBO Campus Oostende

KU LEUVEN khbo

T³ VLAANDEREN
www.t3vlaanderen.be

16de T³ Europe Symposium

Wiskunde en Wetenschappen ondersteunen met ICT

Numeriek, grafisch, symbolisch

Texas Instruments KU LEUVEN

Link: http://www.t3vlaanderen.be/fileadmin/media/symposia/symposium2013/Syllabus_symposium16.pdf

Growth Process described and solved by Technology – An Example of a Sustainable Learning Spiral (Helmut Heugl)

Random Walks on Graphs, Matrices Economics and Monkey Business (Hans Schneebeli); Fermi Problems (Hans Schneebeli); Is the Calculus a Must in General Education (Hans Schneebeli)

Without Data there is no Science (Ian Galloway) and others.

fernande@quimica.unlp.edu.ar

Dear Derivians,

I am attaching a short dfw file that shows a problem that I have calculating integrals that should give gamma functions. Is there any way to overcome this undesirable output?

Greetings Francisco

#1: a :∈ Real (0, ∞)

#2: $$\int_0^\infty \mathrm{EXP}(-a\cdot x^{5/2})\,dx = \frac{\left(\frac{2}{5}\right)!}{a^{2/5}}$$

#3: $$\int_0^\infty \mathrm{EXP}\left(-\frac{4}{5}\cdot x^{5/2}\right)dx = \int_0^\infty \frac{1}{\left(e^{x^{5/2}}\right)^{4/5}}\,dx$$

Liebe DUG-Mitglieder,

Sie werden sich vielleicht fragen, warum die Datei für diesen DNL so groß geworden ist? Verantwortlich dafür sind die vielen Grafiken im ersten Beitrag. Michel, Geneviève und Frédérick haben uns ihren ACA13-Vortrag als ppt-Präsentation zur Verfügung gestellt. Ich habe daraus einen Aufsatz gemacht und die vielen Figuren aus der Präsentation übernommen.

Michel und Partner haben erfolgreich eine DERIVE-Funktionalität auf den TI-NspireCAS übertragen und uns dabei interessante Einblicke in die Funktionsweise eines CAS gegeben.

Es gibt noch einen zweiten Vortrag von ACA13 (Malaga) über das Lösen von polynomialen Gleichungssystemen mit TI-NspireCAS (Michel Beaudin, Gilles Picard und Genèvieve Savard). Er wird im nächsten DNL erscheinen.

Diese beiden Vorträge brachten mich auf eine Idee, die sie auf der letzten Seite dieses DNL finden können.

Erik van Lantschoot hat sich nochmals mit der schönen Brücke in seiner Geburtstadt beschäftigt. Dabei zeigt sich sehr deutlich, dass die Geschichte der Mathematik, der Ingenieurwissenschaft und die Geschichte überhaupt sehr vernetzt sind. Internetquellen mit den digitalisierten Texten aus vorigen Jahrhunderten machen die Beschäftigung sehr spannend. (Testen Sie Ihre „humanistische Vergangenheit" am lateinischen Text von Cornelis Alewyn!)

Auf der Info-Seite finden Sie einen Link zu den Materialien einer erfolgreichen Veranstaltung, die von unserem Mitglied Guido Herweyers in Oostende, Belgien, organisiert wurde. Einige Vorträge und Workshops sind in Englisch, daneben gibt es Beiträge in Französisch und in Holländisch.

Viele Grüße bis zum nächsten Mal

Dear DUG Members,

You might wonder about the size of the DNL-file. This is due to the many figures presented in the first contribution. Michel, Frédérick and Geneviève provided their ACA13 (Malaga) lecture in form of a ppt-presentation which I converted to a written paper. Many graphs had to be copied and pasted.

Michel and his partners were successful in transferring a DERIVE functionality to TI-NspireCAS. In their lecture they give much insight how a CAS is working internally.

There is a second ACA13 lecture treating solving polynomial systems of equations with TI-NspireCAS (Michel Beaudin, Gilles Picard and Geneviève Savard). It will be published din the next DNL.

Working through both papers I got the idea which is presented on the last page of this DNL.

Erik van Lantschoot was busy with the historic landmark in his native town once more. We can see how history of mathematics, history of engineering and history in general are connected. Internet resources offering digitized papers from earlier centuries made reading the paper really exciting. (Test your humanistic education reading Cornelis Alewyn's book written in Latin language.)

The Info page provides a link to materials of a very successful conference organized by our member Guido Herweyers in Oostende, Belgium. The keynotes and some other lectures are in English, and there are lectures in French and Dutch.

Best regards until next time

[illegible]

The *DERIVE-NEWSLETTER* is the Bulletin of the *DERIVE* & CAS-*TI User Group*. It is published at least four times a year with a content of 40 pages minimum. The goals of the *DNL* are to enable the exchange of experiences made with *DERIVE*, *TI*-CAS and other CAS as well to create a group to discuss the possibilities of new methodical and didactical manners in teaching mathematics.

Editor: Mag. Josef Böhm
D´Lust 1, A-3042 Würmla
Austria
Phone: ++43-(0)660 3136365
e-mail: nojo.boehm@pgv.at

Contributions:
Please send all contributions to the Editor. Non-English speakers are encouraged to write their contributions in English to reinforce the international touch of the *DNL*. It must be said, though, that non-English articles will be warmly welcomed nonetheless. Your contributions will be edited but not assessed. By submitting articles the author gives his consent for reprinting it in the *DNL*. The more contributions you will send, the more lively and richer in contents the *DERIVE* & CAS-*TI Newsletter* will be.

Next issue: December 2013

Preview: Contributions waiting to be published

Some simulations of Random Experiments, J. Böhm, AUT, Lorenz Kopp, GER
Wonderful World of Pedal Curves, J. Böhm, AUT
Tools for 3D-Problems, P. Lüke-Rosendahl, GER
Hill-Encryption, J. Böhm, AUT
Simulating a Graphing Calculator in *DERIVE*, J. Böhm, AUT
Do you know this? Cabri & CAS on PC and Handheld, W. Wegscheider, AUT
An Interesting Problem with a Triangle, Steiner Point, P. Lüke-Rosendahl, GER
Graphics World, Currency Change, P. Charland, CAN
Cubics, Quartics – Interesting features, T. Koller & J. Böhm, AUT
Logos of Companies as an Inspiration for Math Teaching
Exciting Surfaces in the FAZ / Pierre Charland´s Graphics Gallery
BooleanPlots.mth, P. Schofield, UK
Old traditional examples for a CAS – what´s new? J. Böhm, AUT
Truth Tables on the TI, M. R. Phillips, USA
Where oh Where is It? (GPS with CAS), C. & P. Leinbach, USA
Embroidery Patterns, H. Ludwig, GER
Mandelbrot and Newton with *DERIVE*, Roman Hašek, CZK
Tutorials for the NSpireCAS, G. Herweyers, BEL
Some Projects with Students, R. Schröder, GER
Dirac Algebra, Clifford Algebra, D. R. Lunsford, USA
Treating Differential Equations (M. Beaudin, G. Piccard, Ch. Trottier), CAN
A New Approach to Taylor Series, D. Oertel, GER
Cesar Multiplication, G. Schödl, AUT
Henon & Co; Find your very own Strange Attractor, J. Böhm, AUT
Rational Hooks, J. Lechner, AUT
Simulation of Dynamic Systems with various Tools, J. Böhm, AUT
Space Curves with adjustable Curvature and Torsion, P. Trebisz, GER
How reliable is mathematical Software? Behrooz Khavari, IRAN
Two Good Turns a.o., D. Halprin. AUS
and others

Impressum:
Medieninhaber: *DERIVE* User Group, A-3042 Würmla, D´Lust 1, AUSTRIA
Richtung: Fachzeitschrift
Herausgeber: Mag. Josef Böhm

$$\int \begin{cases} -1, & x<0 \\ x, & 0\le x<2 \\ 2-x, & 2\le x<5 \\ \sin(x), & 5\le x<\infty \end{cases} dx$$

Integration of Piecewise Continuous Functions

Michel Beaudin, Frédérick Henri,
Geneviève Savard

ÉTS, Montréal, Canada

Abstract

Piecewise functions are important in applied mathematics and engineering students need to deal with them often. In Nspire CAS, templates are an easy way to define piecewise functions; in *DERIVE*, linear combination of indicator functions can be used. Nspire CAS integrates symbolically any piecewise continuous function — and returns, as expected, an everywhere continuous antiderivative — as long as this function is not multiplied by another expression. *DERIVE* knows how to integrate sign($a\ x + b$) $f(x)$ where f is an arbitrary function, a and b real numbers and "sign" stands for the signum function: this is why products of a piecewise function with any other expression can be integrated symbolically. This will be the first part of our talk.

In the second part of this talk, we will show some implementations that will allow Nspire CAS to integrate symbolically products of piecewise functions with expressions: the starting point was the discovery of a non-documented function of Nspire CAS. Examples of various operations between two piecewise functions will be given. As a final example, we will show how we have defined a Fourier series function in Nspire CAS that performs as well as *DERIVE*'s built-in "Fourier" function.

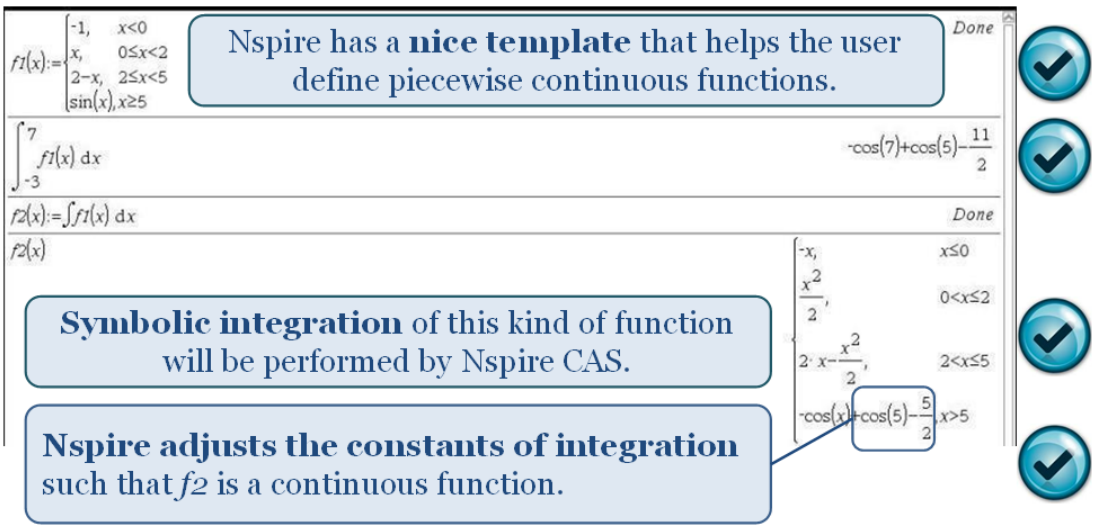
Overview

I
1. Integration of piecewise continuous functions: some problems with Nspire
2. No problem with *DERIVE*! Why?
3. Our solution: Programming new functions in Nspire

II
3. Our solution: Programming new functions in Nspire
4. Some applications: Fourier Series and deSolve
5. Conclusion

Integration of Piecewise Continuous Functions: Problems with Nspire

What Nspire does well :



$f1(x):=\begin{cases}-1, & x<0\\ x, & 0\le x<2\\ 2-x, & 2\le x<5\\ \sin(x), & x\ge 5\end{cases}$ Done

Nspire has a **nice template** that helps the user define piecewise continuous functions.

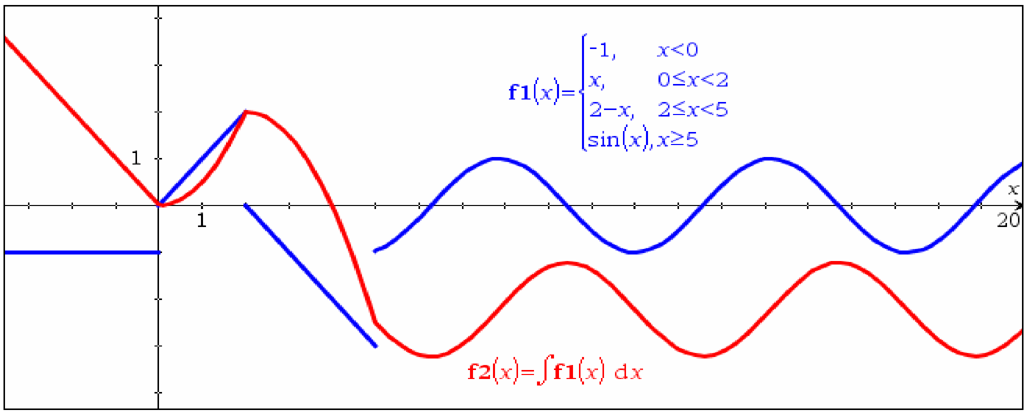
$\int_{-3}^{7} f1(x)\,dx$ $\quad -\cos(7)+\cos(5)-\frac{11}{2}$

$f2(x):=\int f1(x)\,dx$ Done

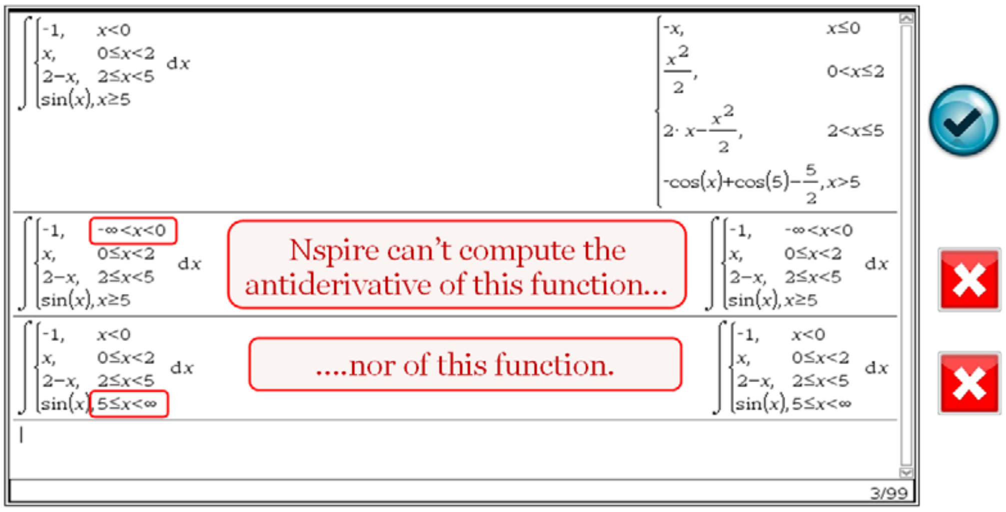
$f2(x)$ $\quad \begin{cases}-x, & x\le 0\\ \frac{x^2}{2}, & 0<x\le 2\\ 2\cdot x-\frac{x^2}{2}, & 2<x\le 5\\ -\cos(x)+\boxed{\cos(5)-\frac{5}{2}}, & x>5\end{cases}$

Symbolic integration of this kind of function will be performed by Nspire CAS.

Nspire adjusts the constants of integration such that *f2* is a continuous function.



A problem arises when ∞ appears in one of the subdomains:



$\int \begin{cases}-1, & x<0\\ x, & 0\le x<2\\ 2-x, & 2\le x<5\\ \sin(x), & x\ge 5\end{cases} dx$ $\quad \begin{cases}-x, & x\le 0\\ \frac{x^2}{2}, & 0<x\le 2\\ 2\cdot x-\frac{x^2}{2}, & 2<x\le 5\\ -\cos(x)+\cos(5)-\frac{5}{2}, & x>5\end{cases}$

$\int \begin{cases}-1, & \boxed{-\infty<x<0}\\ x, & 0\le x<2\\ 2-x, & 2\le x<5\\ \sin(x), & x\ge 5\end{cases} dx$ $\quad \int \begin{cases}-1, & -\infty<x<0\\ x, & 0\le x<2\\ 2-x, & 2\le x<5\\ \sin(x), & x\ge 5\end{cases} dx$

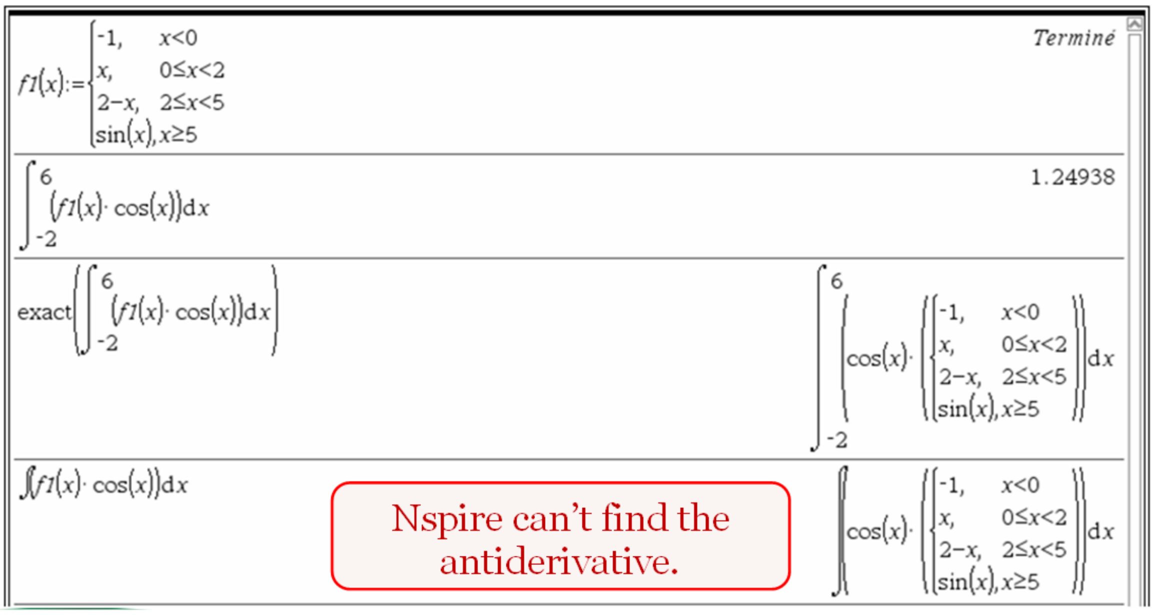
Nspire can't compute the antiderivative of this function...

$\int \begin{cases}-1, & x<0\\ x, & 0\le x<2\\ 2-x, & 2\le x<5\\ \sin(x), & \boxed{5\le x<\infty}\end{cases} dx$ $\quad \int \begin{cases}-1, & x<0\\ x, & 0\le x<2\\ 2-x, & 2\le x<5\\ \sin(x), & 5\le x<\infty\end{cases} dx$

....nor of this function.

3/99

A problem occurs when the piecewise function is multiplied by another function (even a very simple one).



$$f1(x):=\begin{cases}-1, & x<0\\ x, & 0\le x<2\\ 2-x, & 2\le x<5\\ \sin(x), & x\ge 5\end{cases}$$ Terminé

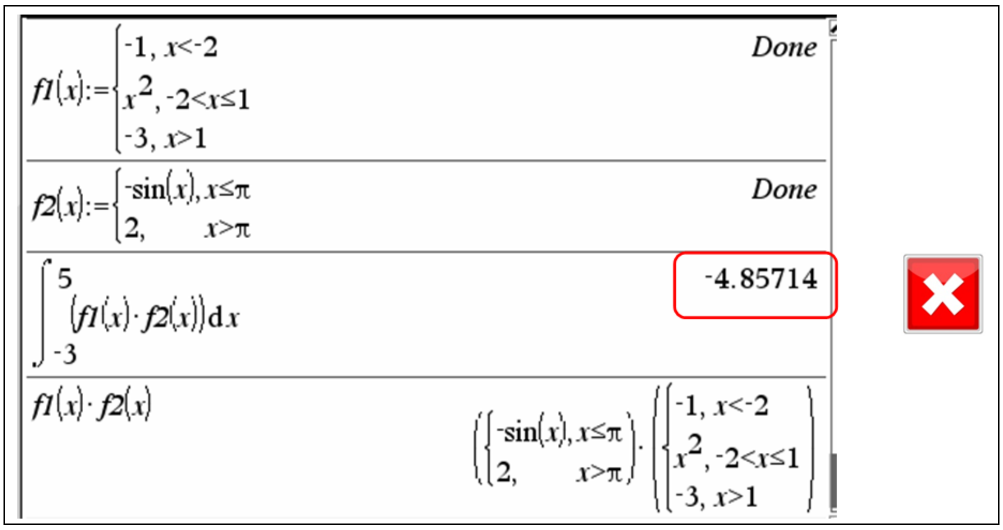
$$\int_{-2}^{6}\left(f1(x)\cdot\cos(x)\right)dx$$ 1.24938

$$\text{exact}\left(\int_{-2}^{6}\left(f1(x)\cdot\cos(x)\right)dx\right) \qquad \int_{-2}^{6}\left(\cos(x)\cdot\left(\begin{cases}-1, & x<0\\ x, & 0\le x<2\\ 2-x, & 2\le x<5\\ \sin(x), & x\ge 5\end{cases}\right)\right)dx$$

$$\int\left(f1(x)\cdot\cos(x)\right)dx \qquad \int\left(\cos(x)\cdot\left(\begin{cases}-1, & x<0\\ x, & 0\le x<2\\ 2-x, & 2\le x<5\\ \sin(x), & x\ge 5\end{cases}\right)\right)dx$$

Nspire can't find the antiderivative.

A problem occurs when we multiply two piecewise defined functions.



$$f1(x):=\begin{cases}-1, & x<-2\\ x^2, & -2<x\le 1\\ -3, & x>1\end{cases}$$ Done

$$f2(x):=\begin{cases}-\sin(x), & x\le\pi\\ 2, & x>\pi\end{cases}$$ Done

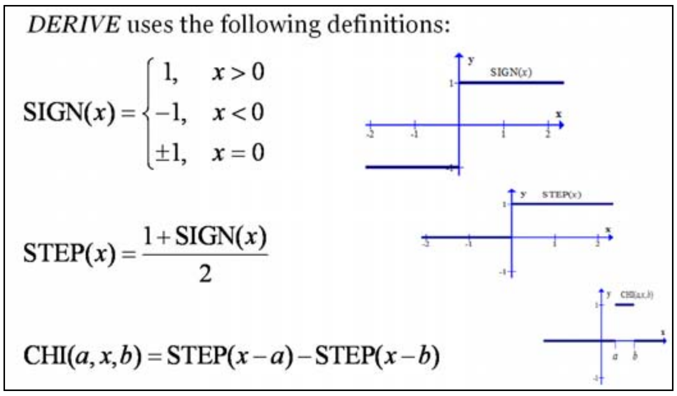
$$\int_{-3}^{5}\left(f1(x)\cdot f2(x)\right)dx$$ $\boxed{-4.85714}$

$$f1(x)\cdot f2(x) \qquad \left(\begin{cases}-\sin(x), & x\le\pi\\ 2, & x>\pi\end{cases}\right)\cdot\left(\begin{cases}-1, & x<-2\\ x^2, & -2<x\le 1\\ -3, & x>1\end{cases}\right)$$

Nspire can't compute the exact value… because it does not simplify the product into a single piecewise function.

No Problem with *DERIVE*! Why?

- Defining piecewise functions with some built-in functions (CHI, SIGN, STEP)
- A very useful integration rule
- Instead of templates, we may use indicator functions to define piecewise functions in *DERIVE*.
- In *DERIVE*, Indicator (CHI), Signum (SIGN) and Heaviside (STEP) functions are built-in; in Nspire CAS, only `sign` is implemented.



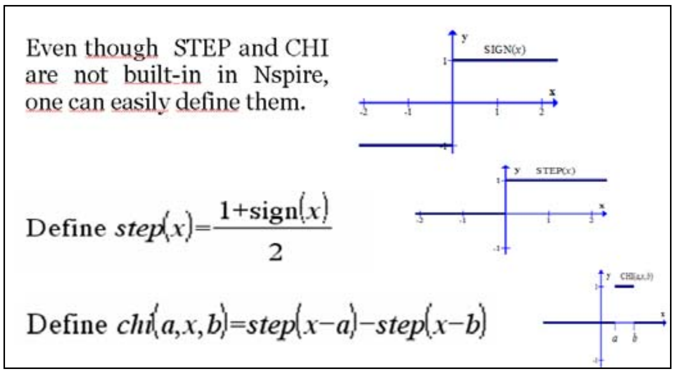
DERIVE uses the following definitions:

$$\mathrm{SIGN}(x)=\begin{cases}1, & x>0\\ -1, & x<0\\ \pm 1, & x=0\end{cases}$$

$$\mathrm{STEP}(x)=\frac{1+\mathrm{SIGN}(x)}{2}$$

$$\mathrm{CHI}(a,x,b)=\mathrm{STEP}(x-a)-\mathrm{STEP}(x-b)$$

SIGN(x)
STEP(x)
CHI(a,x,b)



Even though STEP and CHI are not built-in in Nspire, one can easily define them.

Define $step(x)=\dfrac{1+\mathrm{sign}(x)}{2}$

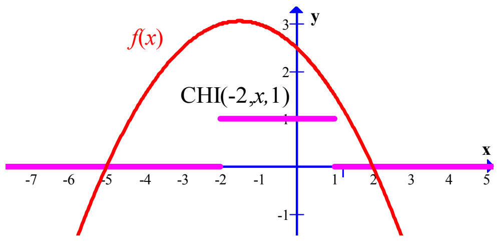
Define $chi(a,x,b)=step(x-a)-step(x-b)$

SIGN(x)
STEP(x)
CHI(a,x,b)

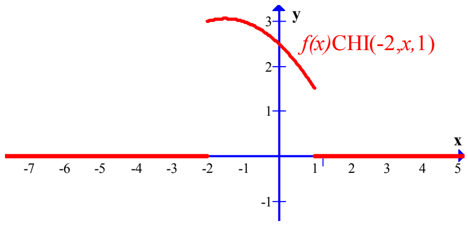
We define piecewise functions in *DERIVE* as a combination of CHI functions.

For example, if we need the piece of $f(x)$ between -2 and 1, we just multiply $f(x)$ by CHI(-2, x, 1):

For example, if we need the piece of $f(x)$ between -2 and 1, we just multiply $f(x)$ by CHI(-2, x, 1):



Values at the extremities of the subintervals are irrelevant, as far as integration is concerned.



Another example:

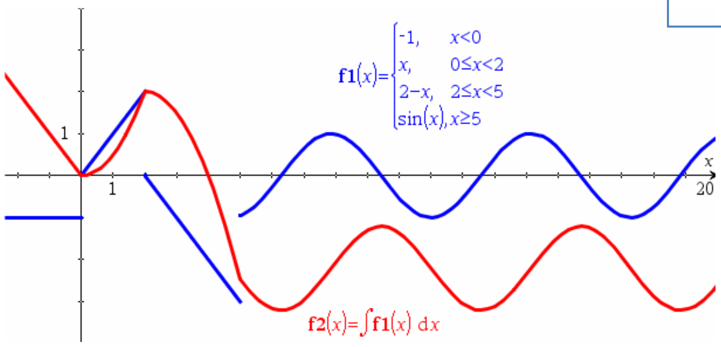
$$f1(x)=\begin{cases}-1 & x<0\\ x & 0<x<2\\ 2-x & 2<x<5\\ \sin(x) & x>5\end{cases} = \begin{matrix}-1\cdot\chi(-\infty,x,0)\\ +x\cdot\chi(0,x,2)\\ +(2-x)\cdot\chi(2,x,5)\\ +\sin(x)\cdot\chi(5,x,\infty)\end{matrix}$$

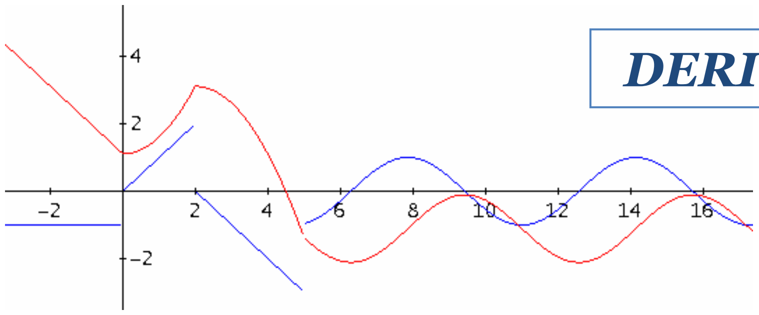
```
f1(x) := - χ(-∞, x, 0) + x·χ(0, x, 2) + (2 - x)·χ(2, x, 5) + SIN(x)·χ(5, x, ∞)
```

Nspire	**DERIVE**
$f1(x):=\begin{cases}-1, & x<0\\ x, & 0\le x<2\\ 2-x, & 2\le x<5\\ \sin(x), & x\ge 5\end{cases}$	f1(x) := - χ(-∞, x, 0) + x·χ(0, x, 2) + (2 - x)·χ(2, x, 5) + SIN(x)·χ(5, x, ∞)
$\int_{-3}^{7} f1(x)\,dx$ $\quad -\cos(7)+\cos(5)-\frac{11}{2}$	$-\text{COS}(7)+\text{COS}(5)-\frac{11}{2}$
$\int f1(x)\,dx$ $\begin{cases}-x, & x\le 0\\ \frac{x^2}{2}, & 0<x\le 2\\ 2\cdot x-\frac{x^2}{2}, & 2<x\le 5\\ -\cos(x)+\cos(5)-\frac{5}{2}, & x>5\end{cases}$	$-\text{SIGN}(x-5)\cdot\left(\frac{\text{COS}(x)}{2}-\frac{\text{COS}(5)}{2}-\frac{x^2}{4}+x+\frac{5}{4}\right)-\frac{x\cdot\lvert x-2\rvert}{2}+\frac{(x+2)\cdot\lvert x\rvert}{4}-\frac{\text{COS}(x)}{2}-\frac{x}{2}$

Both systems can integrate the piecewise function *f1(x)*.

Nspire

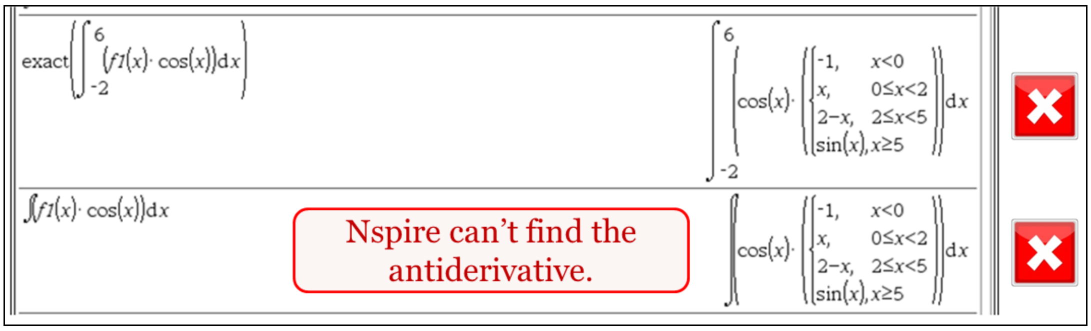




DERIVE

As you can see, the constants of integration differ.

We have seen that Nspire CAS is unable to integrate symbolically a product of a piecewise function with another expression.



$\text{exact}\left(\int_{-2}^{6}(f1(x)\cdot\cos(x))\,dx\right)$ $\quad\int_{-2}^{6}\left(\cos(x)\cdot\left(\begin{cases}-1, & x<0\\ x, & 0\le x<2\\ 2-x, & 2\le x<5\\ \sin(x), & x\ge 5\end{cases}\right)\right)dx$

$\int(f1(x)\cdot\cos(x))\,dx$ $\quad\int\left(\cos(x)\cdot\left(\begin{cases}-1, & x<0\\ x, & 0\le x<2\\ 2-x, & 2\le x<5\\ \sin(x), & x\ge 5\end{cases}\right)\right)dx$

Nspire can't find the antiderivative.

$$\int\left(\cos(x)\cdot\left(\begin{cases}-1, & x<0\\ x, & 0\le x<2\\ 2-x, & 2\le x<5\\ \sin(x), & x\ge 5\end{cases}\right)\right)dx$$

Can *DERIVE* find the antiderivative?

$$\int_{-2}^{6} \text{f1}(x)\cdot \text{COS}(x)\ dx$$

$$-\frac{\text{COS}(12)}{4}+\frac{\text{COS}(10)}{4}-\text{COS}(5)-3\cdot\text{SIN}(5)+2\cdot\text{COS}(2)+\text{SIN}(2)-1$$

and approximated:

1.249382009

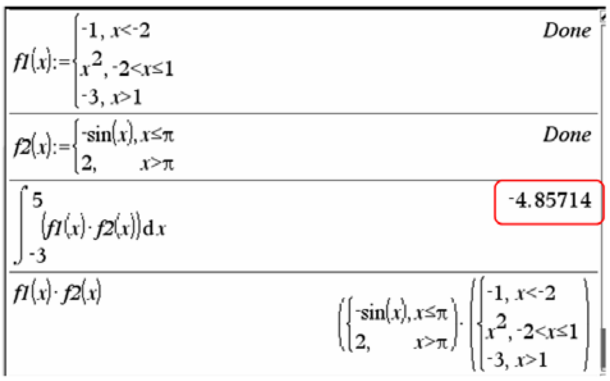
The definite integral (above) and the antiderivative (below):

$$\int \text{f1}(x)\cdot\text{COS}(x)\ dx$$

$$\text{SIGN}(x-5)\cdot\left(\frac{\text{COS}(x)}{2}+\frac{\text{SIN}(x)^2}{4}+\left(\frac{x}{2}-1\right)\cdot\text{SIN}(x)+\frac{\text{COS}(10)}{8}-\frac{\text{COS}(5)}{2}-\frac{3\cdot\text{SIN}(5)}{2}-\frac{1}{8}\right)-\text{SIGN}(x-2)\cdot(\text{COS}(x)+(x-1)\cdot\text{SIN}(x)-\text{COS}(2)-\text{SIN}(2))+\text{SIGN}(x)\cdot\left(\frac{\text{COS}(x)}{2}+\left(\frac{x}{2}+\frac{1}{2}\right)\cdot\text{SIN}(x)-\frac{1}{2}\right)+\frac{\text{SIN}(x)^2}{4}-\frac{\text{SIN}(x)}{2}$$

DERIVE is able to compute the antiderivative of $f1(x)\cos(x)$.

We have seen that Nspire CAS is unable to integrate symbolically a product of 2 piecewise defined functions.



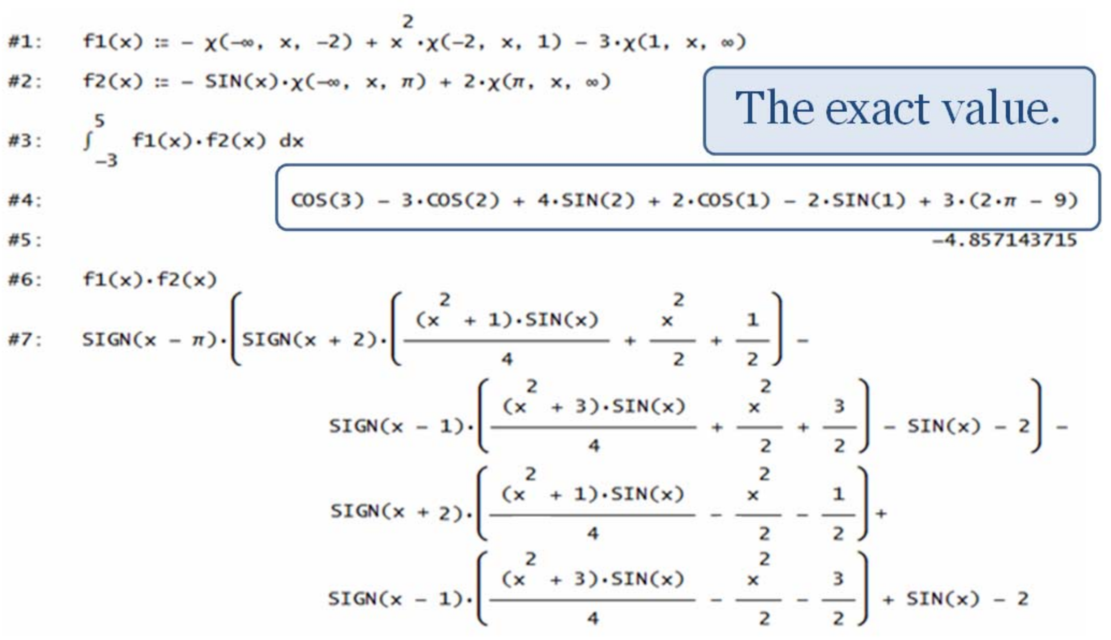
$$f1(x):=\begin{cases}-1, & x<-2\\ x^2, & -2<x\le 1\\ -3, & x>1\end{cases}$$ *Done*

$$f2(x):=\begin{cases}-\sin(x), & x\le\pi\\ 2, & x>\pi\end{cases}$$ *Done*

$$\int_{-3}^{5}\left(f1(x)\cdot f2(x)\right)dx$$ $\boxed{-4.85714}$

$$f1(x)\cdot f2(x)$$ $$\left(\begin{cases}-\sin(x), & x\le\pi\\ 2, & x>\pi\end{cases}\right)\cdot\left(\begin{cases}-1, & x<-2\\ x^2, & -2<x\le 1\\ -3, & x>1\end{cases}\right)$$

Let's compute the same integration with *DERIVE*.



#1: $f1(x) := -\chi(-\infty, x, -2) + x^2\cdot\chi(-2, x, 1) - 3\cdot\chi(1, x, \infty)$

#2: $f2(x) := -\text{SIN}(x)\cdot\chi(-\infty, x, \pi) + 2\cdot\chi(\pi, x, \infty)$

The exact value.

#3: $\int_{-3}^{5} f1(x)\cdot f2(x)\,dx$

#4: $\boxed{\text{COS}(3) - 3\cdot\text{COS}(2) + 4\cdot\text{SIN}(2) + 2\cdot\text{COS}(1) - 2\cdot\text{SIN}(1) + 3\cdot(2\cdot\pi - 9)}$

#5: -4.857143715

#6: $f1(x)\cdot f2(x)$

#7:
$$\text{SIGN}(x-\pi)\cdot\left(\text{SIGN}(x+2)\cdot\left(\frac{(x^2+1)\cdot\text{SIN}(x)}{4} + \frac{x^2}{2} + \frac{1}{2}\right) - \text{SIGN}(x-1)\cdot\left(\frac{(x^2+3)\cdot\text{SIN}(x)}{4} + \frac{x^2}{2} + \frac{3}{2}\right) - \text{SIN}(x) - 2\right) - \text{SIGN}(x+2)\cdot\left(\frac{(x^2+1)\cdot\text{SIN}(x)}{4} - \frac{x^2}{2} - \frac{1}{2}\right) + \text{SIGN}(x-1)\cdot\left(\frac{(x^2+3)\cdot\text{SIN}(x)}{4} - \frac{x^2}{2} - \frac{3}{2}\right) + \text{SIN}(x) - 2$$

DERIVE unifies the product into a combination of SIGN functions.

The reasons:

1. Piecewise functions are defined as a combination of CHI functions (and this simplifies to SIGN functions).
2. DERIVE knows the rule:

   See DERIVE's "stepwise simplification" showing the rule:

#13: $\int f(x)\cdot\text{SIGN}(a\cdot x + b)\,dx$

$$\int F(x)\cdot\text{SIGN}(a\cdot x+b)\,dx \rightarrow \text{SIGN}(a\cdot x+b)\cdot\left(\int F(x)\,dx - \text{SUBST}\left(\int F(x)\,dx, x, -\frac{b}{a}\right)\right)$$

This rule is combined with the following rule:

```
#15:   |x|

|x| → x·SIGN(x)
```

when *DERIVE* computes an integral involving an absolute value. For example,

$$\int \left|x^2 + 5\cdot x + 6\right| \, dx$$

$$\mathrm{SIGN}(x + 3)\cdot\left(\frac{(2\cdot x^3 + 15\cdot x^2 + 36\cdot x + 28)\cdot \mathrm{SIGN}(x + 2)}{6} + \frac{1}{6}\right)$$

Our Solution: Programming New Functions in Nspire

Because Nspire CAS is able to integrate symbolically a *unique* piecewise function (as long as no infinity appears in the domain!)
we thought :

a) to transform the product $f1(x)\cdot f2(x)$ into a single piecewise function,

b) to "remove" every occurrence of "infinity" in the domain,

c) and to use the built-in integrator to compute definite or indefinite integrals.

We want Nspire CAS to continue using its own – attractive – templates instead of using indicator functions.

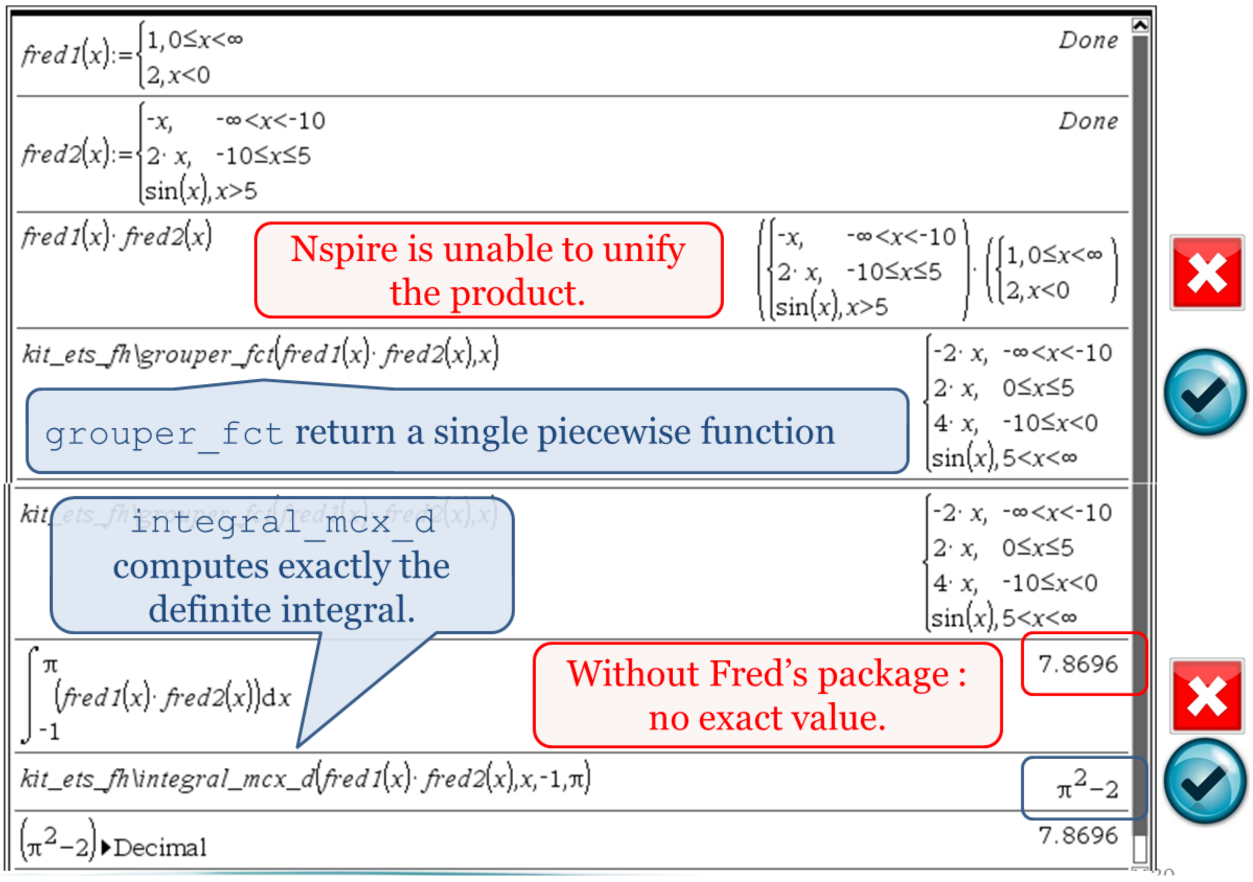
Our colleague Frédérick Henri ("Fred") has programmed some simple but quite efficient functions.

a) `grouper_fct` **groups in a single piecewise function** an expression that contains one or more piecewise subexpressions.

b) `fct_sans_infini` **removes every occurence** of ∞ or -∞ in the domain.

c) `integral_mcx` **symbolically integrates** (indefinite integral) and `integral_mcx_d` **computes exactly the definite integral** using the built-in integrator.

A few Examples:



$fred1(x):=\begin{cases}1, & 0\le x<\infty\\ 2, & x<0\end{cases}$ Done

$fred2(x):=\begin{cases}-x, & -\infty<x<-10\\ 2\cdot x, & -10\le x\le 5\\ \sin(x), & x>5\end{cases}$ Done

$fred1(x)\cdot fred2(x)$ $\left(\begin{cases}-x, & -\infty<x<-10\\ 2\cdot x, & -10\le x\le 5\\ \sin(x), & x>5\end{cases}\right)\cdot\left(\begin{cases}1, & 0\le x<\infty\\ 2, & x<0\end{cases}\right)$

Nspire is unable to unify the product.

kit_ets_fh\grouper_fct(fred1(x)·fred2(x),x) $\begin{cases}-2\cdot x, & -\infty<x<-10\\ 2\cdot x, & 0\le x\le 5\\ 4\cdot x, & -10\le x<0\\ \sin(x), & 5<x<\infty\end{cases}$

grouper_fct return a single piecewise function

kit_ets_fh\integral_mcx_d(fred1(x)·fred2(x),x) $\begin{cases}-2\cdot x, & -\infty<x<-10\\ 2\cdot x, & 0\le x\le 5\\ 4\cdot x, & -10\le x<0\\ \sin(x), & 5<x<\infty\end{cases}$

integral_mcx_d computes exactly the definite integral.

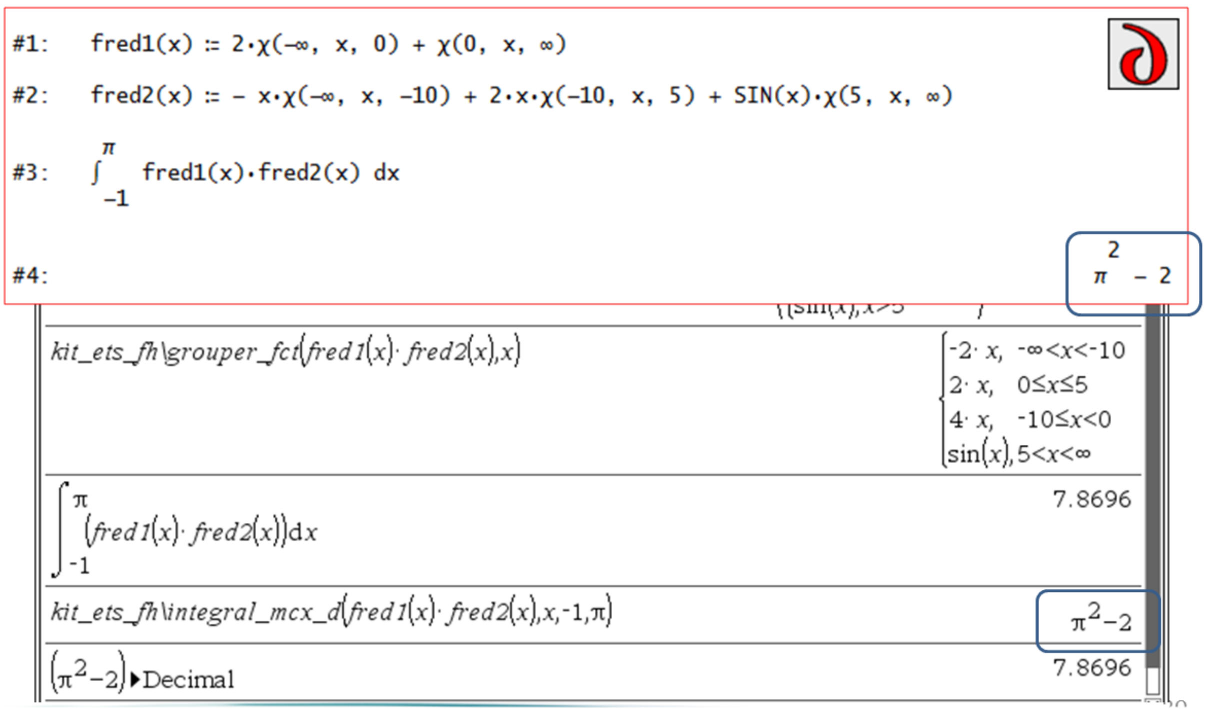
$\int_{-1}^{\pi}(fred1(x)\cdot fred2(x))dx$ 7.8696

Without Fred's package : no exact value.

kit_ets_fh\integral_mcx_d(fred1(x)·fred2(x),x,-1,π) π^2-2

(π^2-2)▸Decimal 7.8696

We observe the same result when using *DERIVE*.



```
#1:   fred1(x) := 2·χ(-∞, x, 0) + χ(0, x, ∞)

#2:   fred2(x) := - x·χ(-∞, x, -10) + 2·x·χ(-10, x, 5) + SIN(x)·χ(5, x, ∞)
```

#3: $\int_{-1}^{\pi} fred1(x)\cdot fred2(x)\ dx$

#4: $\pi^2 - 2$

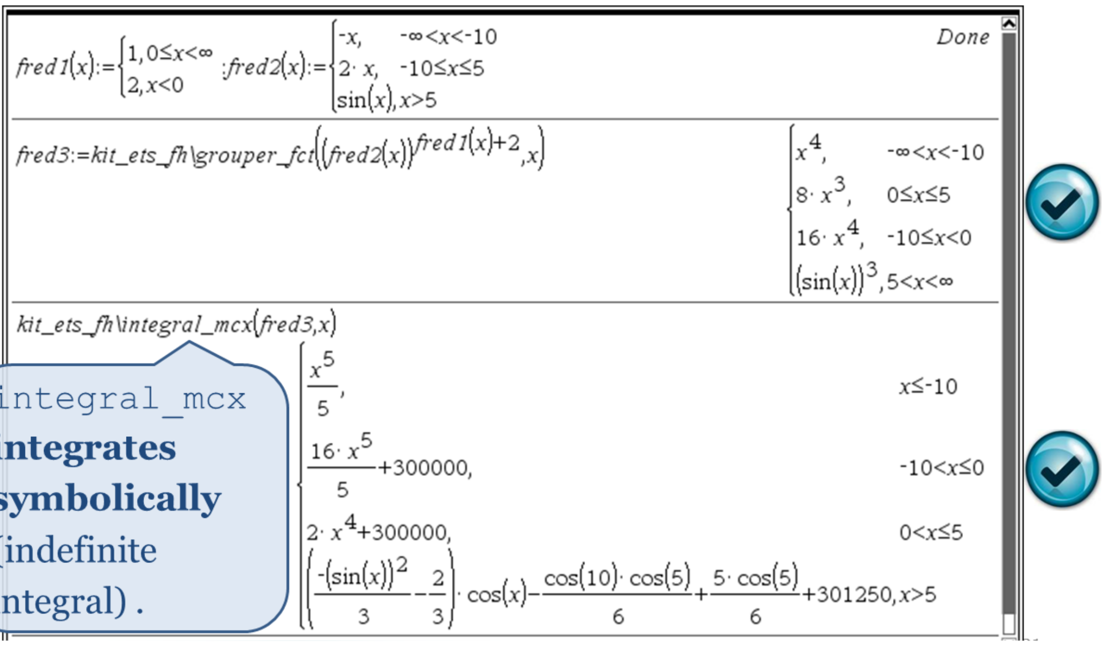
kit_ets_fh\grouper_fct(fred1(x)·fred2(x),x) $\begin{cases}-2\cdot x, & -\infty<x<-10\\ 2\cdot x, & 0\le x\le 5\\ 4\cdot x, & -10\le x<0\\ \sin(x), & 5<x<\infty\end{cases}$

$\int_{-1}^{\pi}(fred1(x)\cdot fred2(x))dx$ 7.8696

kit_ets_fh\integral_mcx_d(fred1(x)·fred2(x),x,-1,π) π^2-2

(π^2-2)▸Decimal 7.8696

`grouper_fct` also works with exponentiation:



$fred1(x):=\begin{cases}1, & 0\leq x<\infty\\ 2, & x<0\end{cases}:fred2(x):=\begin{cases}-x, & -\infty<x<-10\\ 2\cdot x, & -10\leq x\leq 5\\ \sin(x), & x>5\end{cases}$ Done

$fred3:=kit\_ets\_fh\backslash grouper\_fct\left((fred2(x))^{fred1(x)+2},x\right)$ $\begin{cases}x^4, & -\infty<x<-10\\ 8\cdot x^3, & 0\leq x\leq 5\\ 16\cdot x^4, & -10\leq x<0\\ (\sin(x))^3, & 5<x<\infty\end{cases}$

$kit\_ets\_fh\backslash integral\_mcx(fred3,x)$

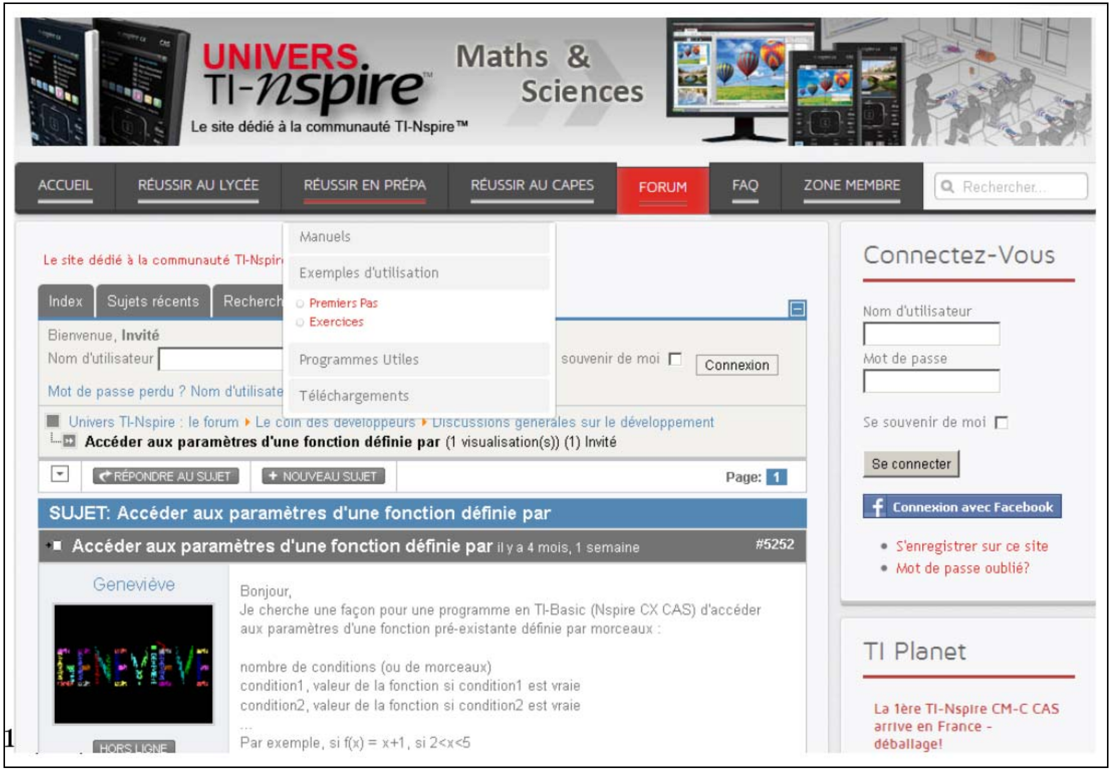
$$\begin{cases}\frac{x^5}{5}, & x\leq -10\\ \frac{16\cdot x^5}{5}+300000, & -10<x\leq 0\\ 2\cdot x^4+300000, & 0<x\leq 5\\ \left(\frac{-(\sin(x))^2}{3}-\frac{2}{3}\right)\cdot\cos(x)-\frac{\cos(10)\cdot\cos(5)}{6}+\frac{5\cdot\cos(5)}{6}+301250, & x>5\end{cases}$$

`integral_mcx` **integrates symbolically** (indefinite integral) .

Let's explain some simple algorithms and show some code.

First of all, in order to manipulate piecewise functions, we need a command to extract pieces of the function. Extraction is not documented into Nspire CAS user guide, but the following "discovery" saved us!

`part()`



Univers TI-Nspire : le forum ▸ Le coin des développeurs ▸ Discussions générales sur le développement
Accéder aux paramètres d'une fonction définie par (1 visualisation(s)) (1) Invité

SUJET: Accéder aux paramètres d'une fonction définie par

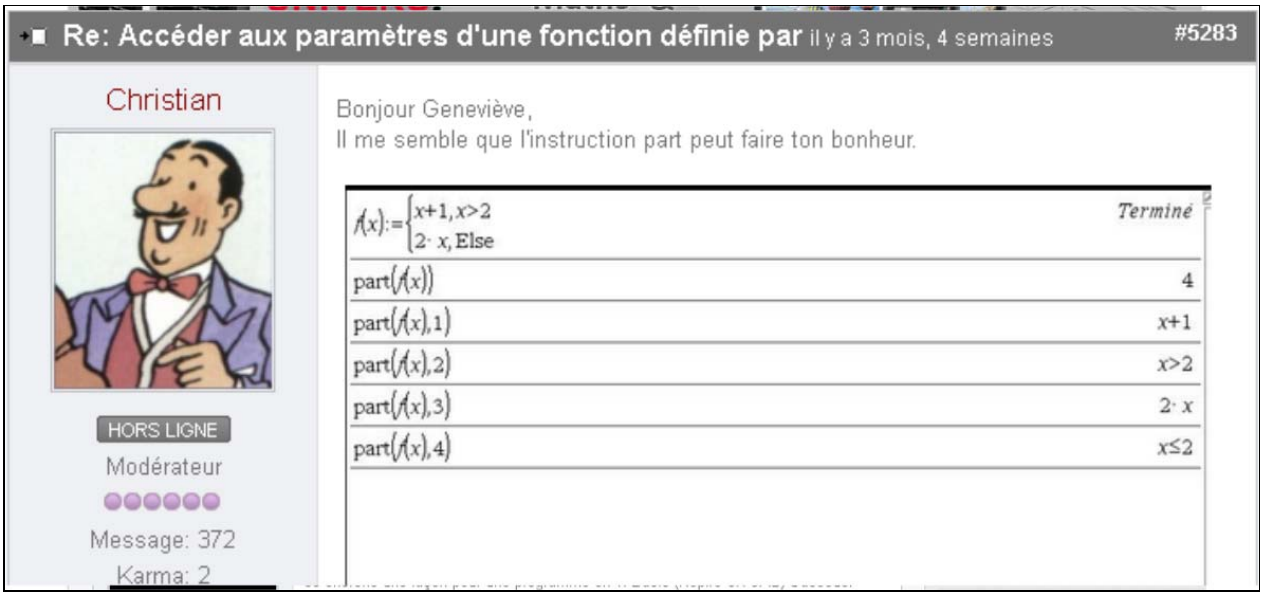
Accéder aux paramètres d'une fonction définie par il y a 4 mois, 1 semaine #5252

Geneviève

Bonjour,
Je cherche une façon pour une programme en TI-Basic (Nspire CX CAS) d'accéder aux paramètres d'une fonction pré-existante définie par morceaux :

nombre de conditions (ou de morceaux)
condition1, valeur de la fonction si condition1 est vraie
condition2, valeur de la fonction si condition2 est vraie
...
Par exemple, si f(x) = x+1, si 2<x<5

1



Re: Accéder aux paramètres d'une fonction définie par il y a 3 mois, 4 semaines #5283

Christian

HORS LIGNE

Modérateur

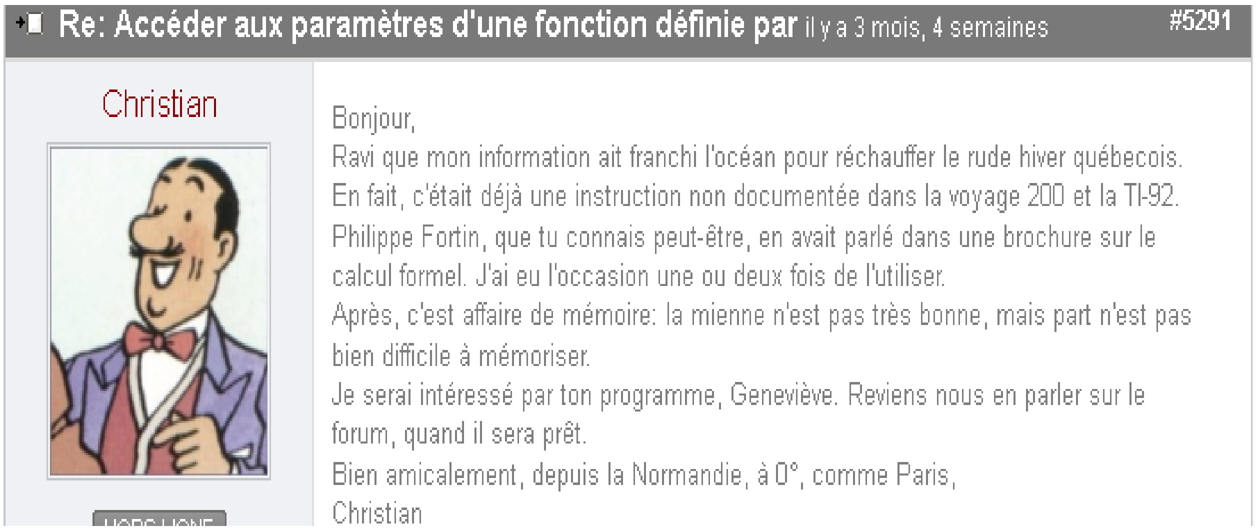
Message: 372

Karma: 2

Bonjour Geneviève,

Il me semble que l'instruction part peut faire ton bonheur.

$f(x):=\begin{cases}x+1, & x>2\\ 2\cdot x, & \text{Else}\end{cases}$	*Terminé*
part(f(x))	4
part(f(x),1)	$x+1$
part(f(x),2)	$x>2$
part(f(x),3)	$2\cdot x$
part(f(x),4)	$x\leq 2$



Re: Accéder aux paramètres d'une fonction définie par il y a 3 mois, 4 semaines #5291

Christian

HORS LIGNE

Bonjour,

Ravi que mon information ait franchi l'océan pour réchauffer le rude hiver québecois.

En fait, c'était déjà une instruction non documentée dans la voyage 200 et la TI-92.

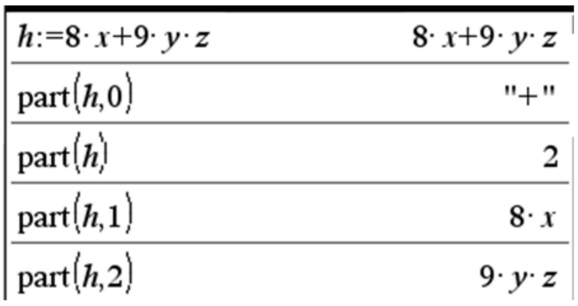
Philippe Fortin, que tu connais peut-être, en avait parlé dans une brochure sur le calcul formel. J'ai eu l'occasion une ou deux fois de l'utiliser.

Après, c'est affaire de mémoire: la mienne n'est pas très bonne, mais part n'est pas bien difficile à mémoriser.

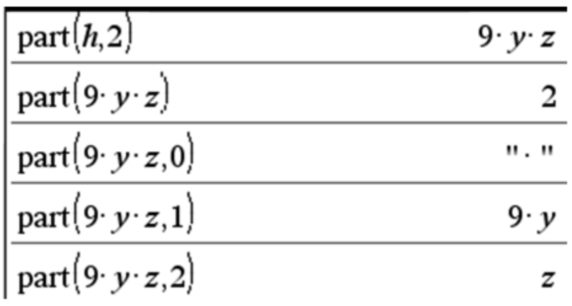
Je serai intéressé par ton programme, Geneviève. Reviens nous en parler sur le forum, quand il sera prêt.

Bien amicalement, depuis la Normandie, à 0°, comme Paris,

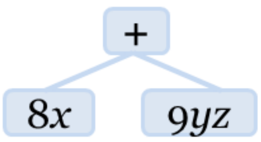
Christian

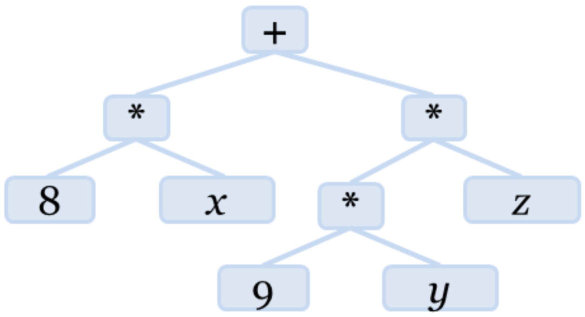


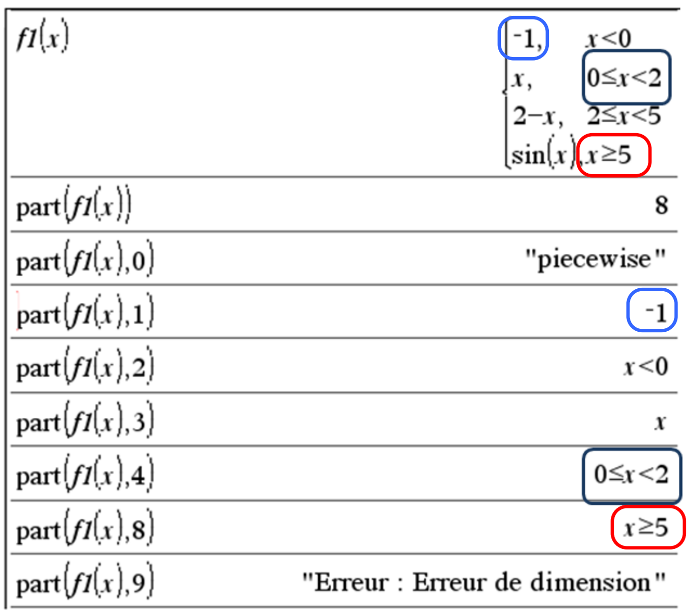
$h:=8\cdot x+9\cdot y\cdot z$	$8\cdot x+9\cdot y\cdot z$
part(h,0)	"+"
part(h)	2
part(h,1)	$8\cdot x$
part(h,2)	$9\cdot y\cdot z$



part(h,2)	$9\cdot y\cdot z$
part($9\cdot y\cdot z$)	2
part($9\cdot y\cdot z$,0)	"."
part($9\cdot y\cdot z$,1)	$9\cdot y$
part($9\cdot y\cdot z$,2)	z



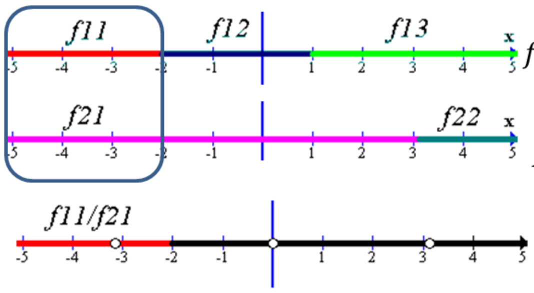




$f1(x)$	$\begin{cases} \boxed{-1}, & x<0 \\ x, & \boxed{0\le x<2} \\ 2-x, & 2\le x<5 \\ \sin(x), & \boxed{x\ge 5} \end{cases}$
part$(f1(x))$	8
part$(f1(x),0)$	"piecewise"
part$(f1(x),1)$	$\boxed{-1}$
part$(f1(x),2)$	$x<0$
part$(f1(x),3)$	x
part$(f1(x),4)$	$\boxed{0\le x<2}$
part$(f1(x),8)$	$\boxed{x\ge 5}$
part$(f1(x),9)$	"Erreur : Erreur de dimension"

`part()` is similar to Maple's `op()`

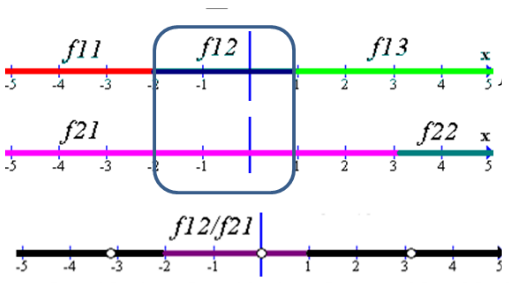
```
unifier_fcts(f1(x), "/", f2(x), x)
```



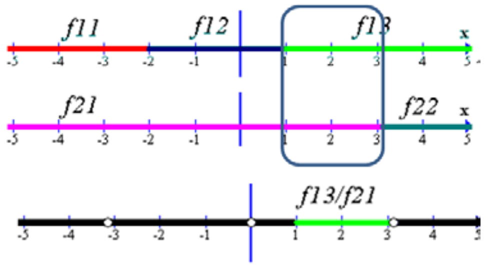
$$f1(x) = \begin{cases} -1 & x < -2 \\ x^2 & -2 \le x \le 1 \\ -3 & x > 1 \end{cases} = \begin{cases} f11 & dom(f11) \\ f12 & dom(f12) \\ f13 & dom(f13) \end{cases}$$

$$f2(x) = \begin{cases} -\sin(x) & x \le \pi \\ 2 & x > \pi \end{cases} = \begin{cases} f21 & dom(f21) \\ f22 & dom(f22) \end{cases}$$

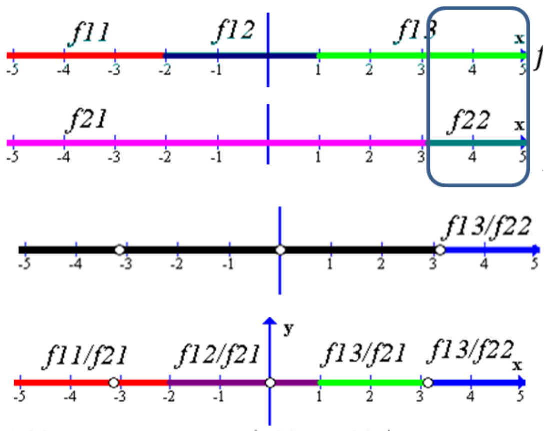
$dom(f11) \cap dom(f21) \cap domain(f11/f21)$



$dom(f11) \cap dom(f21) \cap domain(f11/f21)$
$dom(f12) \cap dom(f21) \cap domain(f12/f21)$



$dom(f11) \cap dom(f21) \cap domain(f11/f21)$
$dom(f12) \cap dom(f21) \cap domain(f12/f21)$
$dom(f13) \cap dom(f21) \cap domain(f13/f21)$

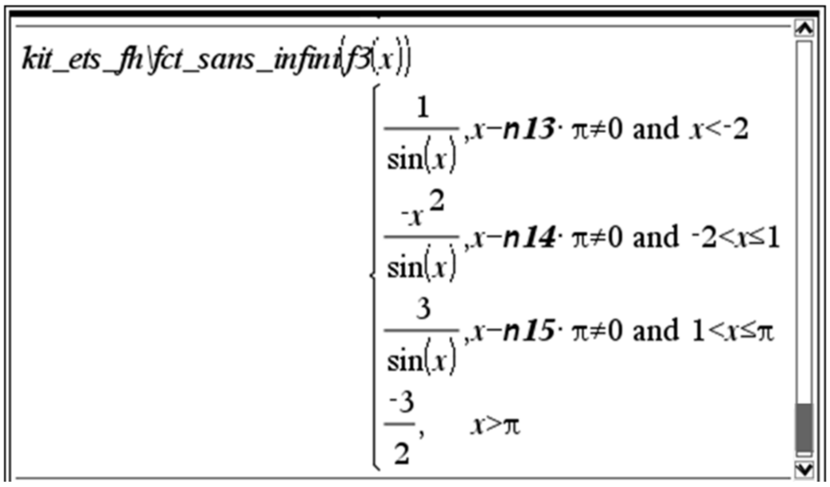


$dom(f11) \cap dom(f21) \cap domain(f11/f21)$
$dom(f12) \cap dom(f21) \cap domain(f12/f21)$
$dom(f13) \cap dom(f21) \cap domain(f13/f21)$
$dom(f11) \cap dom(f22) = \emptyset$
$dom(f12) \cap dom(f22) = \emptyset$
$dom(f13) \cap dom(f22) \cap domain(f13/f22)$

f3(x):=kit_ets_fh\unifier_fcts(f1(x),"/",f2(x),x) *Terminé*

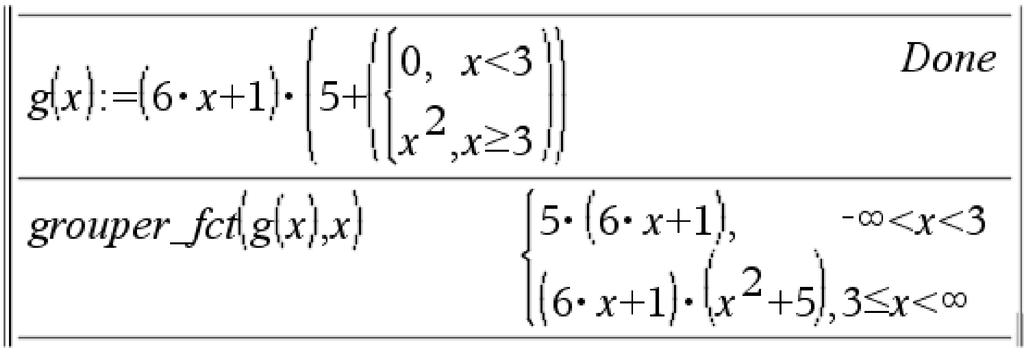
f3(x)
$$\begin{cases} \frac{1}{\sin(x)}, & x-\boldsymbol{n10}\cdot\pi\neq 0 \text{ and } x<-2 \\ \frac{-x^2}{\sin(x)}, & x-\boldsymbol{n11}\cdot\pi\neq 0 \text{ and } -2<x\le 1 \\ \frac{3}{\sin(x)}, & x-\boldsymbol{n12}\cdot\pi\neq 0 \text{ and } 1<x\le\pi \\ \frac{-3}{2}, & \pi<x<\infty \end{cases}$$

Then `fct_sans_infini(f3(x))`



kit_ets_fh\fct_sans_infini(f3(x))
$$\begin{cases} \frac{1}{\sin(x)}, & x-\boldsymbol{n13}\cdot\pi\neq 0 \text{ and } x<-2 \\ \frac{-x^2}{\sin(x)}, & x-\boldsymbol{n14}\cdot\pi\neq 0 \text{ and } -2<x\le 1 \\ \frac{3}{\sin(x)}, & x-\boldsymbol{n15}\cdot\pi\neq 0 \text{ and } 1<x\le\pi \\ \frac{-3}{2}, & x>\pi \end{cases}$$

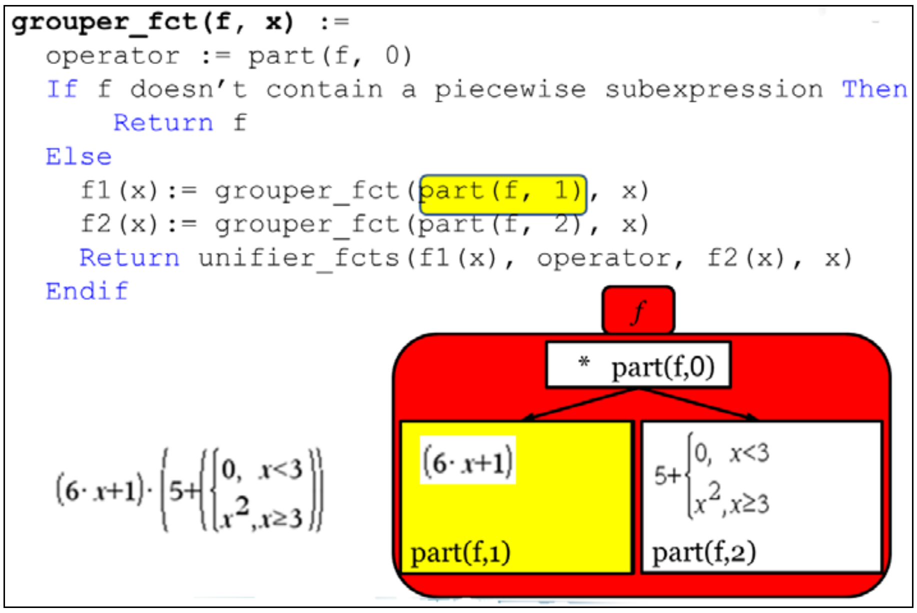
`grouper_fct(f, x)`: generalizes `unifier_fcts` by working recursively (in case of more complicated functions).



$g(x):=(6\cdot x+1)\cdot\left(5+\left(\begin{cases} 0, & x<3 \\ x^2, & x\ge 3 \end{cases}\right)\right)$ *Done*

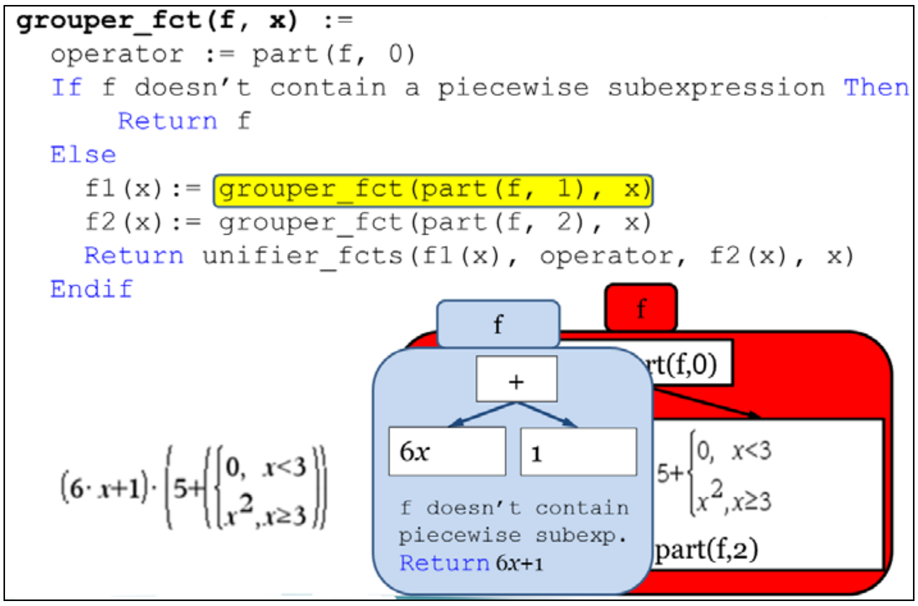
grouper_fct(g(x),x)
$$\begin{cases} 5\cdot(6\cdot x+1), & -\infty<x<3 \\ (6\cdot x+1)\cdot(x^2+5), & 3\le x<\infty \end{cases}$$

Program grouper_fct:



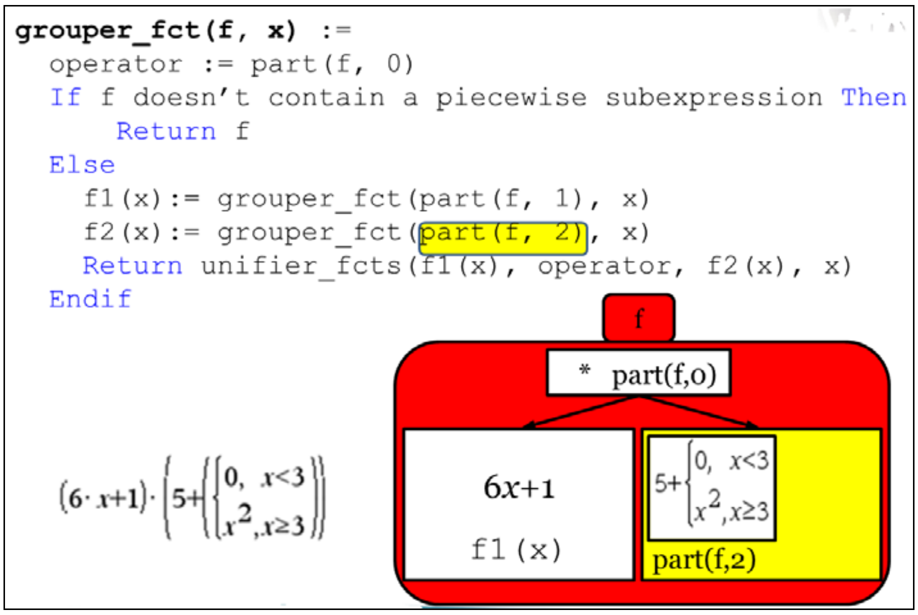
```
grouper_fct(f, x) :=
  operator := part(f, 0)
  If f doesn't contain a piecewise subexpression Then
      Return f
  Else
    f1(x):= grouper_fct(part(f, 1), x)
    f2(x):= grouper_fct(part(f, 2), x)
    Return unifier_fcts(f1(x), operator, f2(x), x)
  Endif
```

$(6\cdot x+1)\cdot\left(5+\begin{cases}0, & x<3\\ x^2, & x\geq 3\end{cases}\right)$



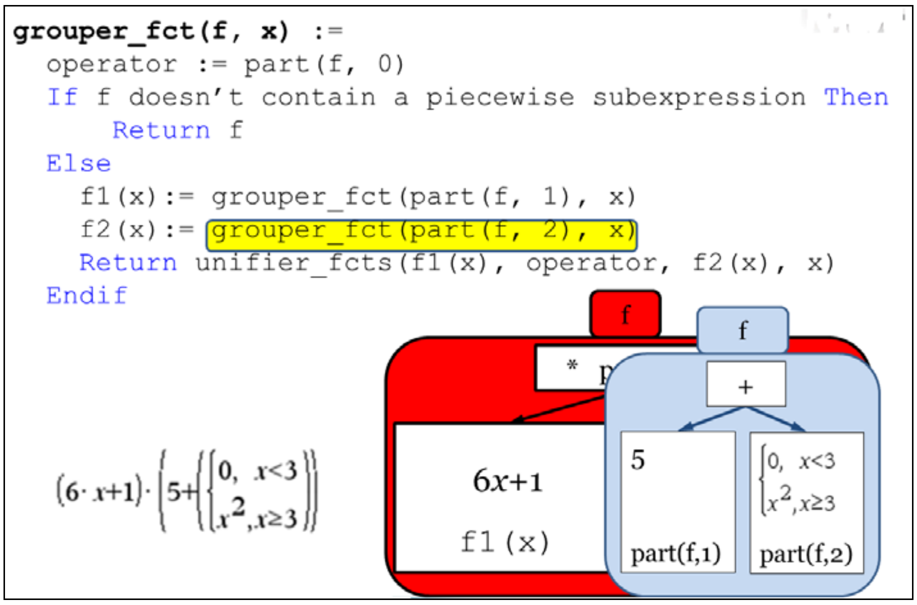
```
grouper_fct(f, x) :=
  operator := part(f, 0)
  If f doesn't contain a piecewise subexpression Then
      Return f
  Else
    f1(x):= grouper_fct(part(f, 1), x)
    f2(x):= grouper_fct(part(f, 2), x)
    Return unifier_fcts(f1(x), operator, f2(x), x)
  Endif
```

$(6\cdot x+1)\cdot\left(5+\begin{cases}0, & x<3\\ x^2, & x\geq 3\end{cases}\right)$



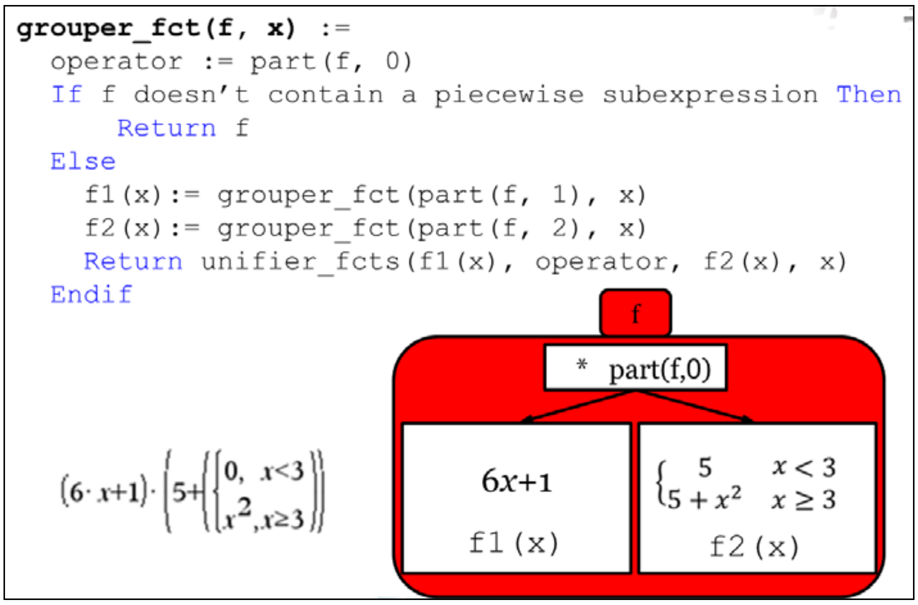
```
grouper_fct(f, x) :=
  operator := part(f, 0)
  If f doesn't contain a piecewise subexpression Then
      Return f
  Else
    f1(x):= grouper_fct(part(f, 1), x)
    f2(x):= grouper_fct(part(f, 2), x)
    Return unifier_fcts(f1(x), operator, f2(x), x)
  Endif
```

$(6\cdot x+1)\cdot\left(5+\begin{cases}0, & x<3\\ x^2, & x\geq 3\end{cases}\right)$



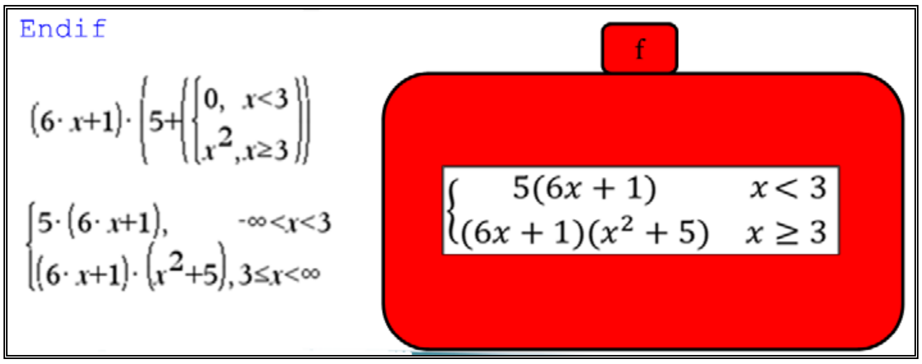
```
grouper_fct(f, x) :=
  operator := part(f, 0)
  If f doesn't contain a piecewise subexpression Then
      Return f
  Else
    f1(x):= grouper_fct(part(f, 1), x)
    f2(x):= grouper_fct(part(f, 2), x)
    Return unifier_fcts(f1(x), operator, f2(x), x)
  Endif
```

$$(6\cdot x+1)\cdot\left(5+\begin{cases}0, & x<3\\ x^2, & x\geq 3\end{cases}\right)$$



```
grouper_fct(f, x) :=
  operator := part(f, 0)
  If f doesn't contain a piecewise subexpression Then
      Return f
  Else
    f1(x):= grouper_fct(part(f, 1), x)
    f2(x):= grouper_fct(part(f, 2), x)
    Return unifier_fcts(f1(x), operator, f2(x), x)
  Endif
```

$$(6\cdot x+1)\cdot\left(5+\begin{cases}0, & x<3\\ x^2, & x\geq 3\end{cases}\right)$$



```
Endif
```

$$(6\cdot x+1)\cdot\left(5+\begin{cases}0, & x<3\\ x^2, & x\geq 3\end{cases}\right)$$

$$\begin{cases}5\cdot(6\cdot x+1), & -\infty<x<3\\ (6\cdot x+1)\cdot(x^2+5), & 3\leq x<\infty\end{cases}$$

Some Applications: Fourier Series and deSolve with Nspire

Let us recall that if an expression *f* of the variable *t* is defined over the interval $t1 < t < t2$ and extended outside the interval by periodicity (the period being $P = t2 - t1$), then the Fourier polynomial of order *n* of *f* is the following trigonometric polynomial:

$$\frac{1}{t2-t1}\int_{t1}^{t2} f\,dt + \sum_{k=1}^{n}\left(\frac{2}{t2-t1}\int_{t1}^{t2} f\cos\left(\frac{2k\pi t}{t2-t1}\right)dt\cos\left(\frac{2k\pi t}{t2-t1}\right) + \frac{2}{t2-t1}\int_{t1}^{t2} f\sin\left(\frac{2k\pi t}{t2-t1}\right)dt\sin\left(\frac{2k\pi t}{t2-t1}\right)\right)$$

$$\left(\text{that is: } \frac{a_0}{2} + \sum_{k=1}^{n} a_k\cos\left(\frac{2k\pi t}{P}\right) + b_k\sin\left(\frac{2k\pi t}{P}\right)\right)$$

And this is *DERIVE*'s definition from the library "Applications of Integration":

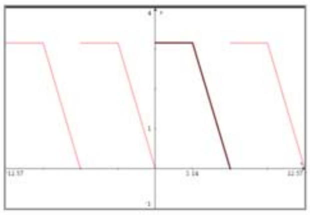
$$\text{FOURIER}(y,\ t,\ t1,\ t2,\ n) := \frac{1}{t2 - t1}\cdot\int_{t1}^{t2} y\ dt + \frac{2}{t2 - t1}\cdot\sum_{m\_=1}^{n}\left(\text{COS}\left(\frac{2\cdot\pi\cdot m\_\cdot t}{t2 - t1}\right)\cdot\int_{t1}^{t2} y\cdot\text{COS}\left(\frac{2\cdot\pi\cdot m\_\cdot t}{t2 - t1}\right) dt + \text{SIN}\left(\frac{2\cdot\pi\cdot m\_\cdot t}{t2 - t1}\right)\cdot\int_{t1}^{t2} y\cdot\text{SIN}\left(\frac{2\cdot\pi\cdot m\_\cdot t}{t2 - t1}\right) dt\right)$$

At ETS, when students need to compute the Fourier coefficients of a periodic signal, they use their TI-Nspire CX CAS handheld to compute the integrals (for the Fourier coefficients).

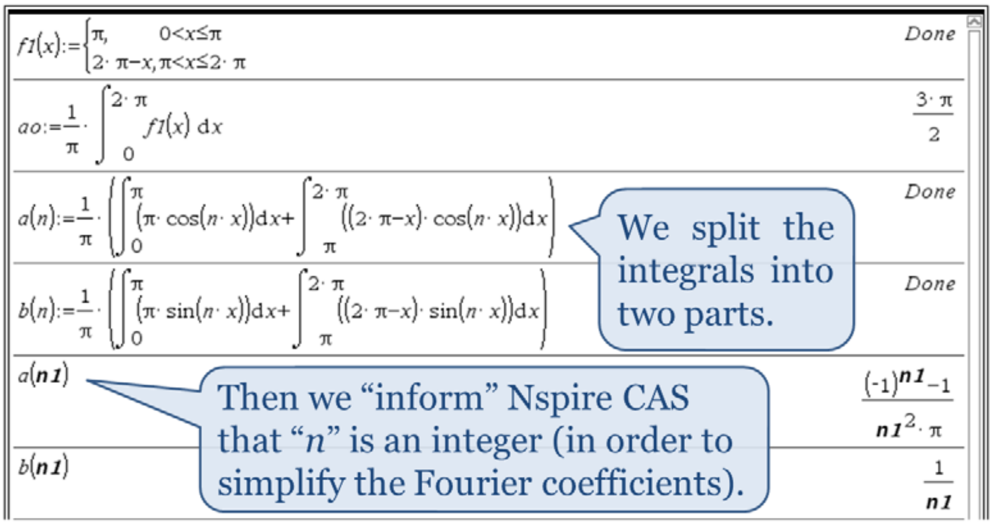
Then, they store the coefficients and are able to produce any partial sum in exact arithmetic.

Here is an example. Students are asked to find the Fourier series of the following 2π - periodic signal.

$$f(x)=\begin{cases}\pi, & 0<x\le\pi\\ -x+2\pi, & \pi<x\le 2\pi\end{cases}\qquad f(x+2\pi)=f(x)$$



The signal is neither odd nor even. So, using Nspire CAS, we compute the Fourier coefficients, splitting the integrals ourselves into two parts!



$f1(x):=\begin{cases}\pi, & 0<x\le\pi\\ 2\cdot\pi-x, & \pi<x\le 2\cdot\pi\end{cases}$ — *Done*

$ao:=\frac{1}{\pi}\cdot\int_0^{2\cdot\pi} f1(x)\,dx$ — $\frac{3\cdot\pi}{2}$

$a(n):=\frac{1}{\pi}\cdot\left(\int_0^{\pi}(\pi\cdot\cos(n\cdot x))dx+\int_{\pi}^{2\cdot\pi}((2\cdot\pi-x)\cdot\cos(n\cdot x))dx\right)$ — *Done*

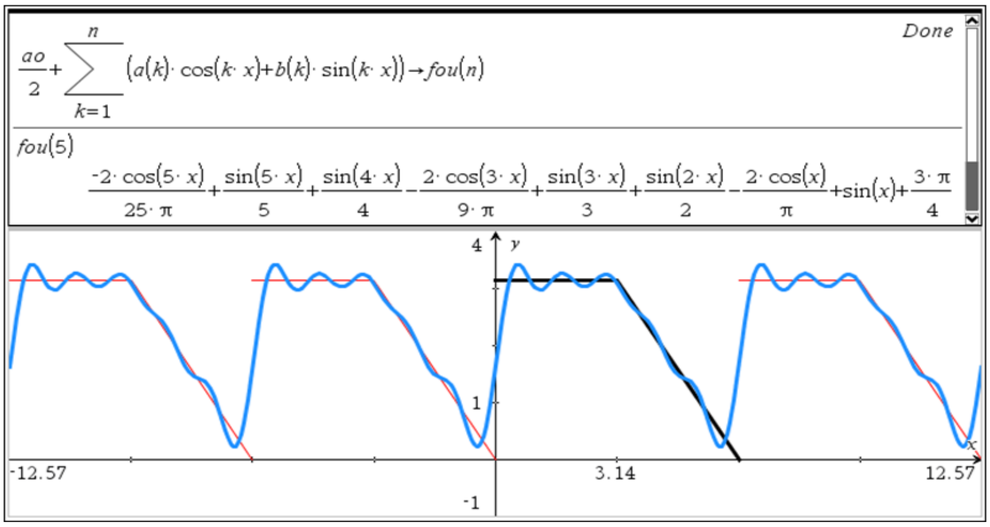
$b(n):=\frac{1}{\pi}\cdot\left(\int_0^{\pi}(\pi\cdot\sin(n\cdot x))dx+\int_{\pi}^{2\cdot\pi}((2\cdot\pi-x)\cdot\sin(n\cdot x))dx\right)$ — *Done*

$a(\mathbf{n1})$ — $\frac{(-1)^{\mathbf{n1}}-1}{\mathbf{n1}^2\cdot\pi}$

$b(\mathbf{n1})$ — $\frac{1}{\mathbf{n1}}$

We split the integrals into two parts.

Then we "inform" Nspire CAS that "*n*" is an integer (in order to simplify the Fourier coefficients).



$$\frac{ao}{2}+\sum_{k=1}^{n}\left(a(k)\cdot\cos(k\cdot x)+b(k)\cdot\sin(k\cdot x)\right)\rightarrow fou(n)$$ Done

fou(5)

$$\frac{-2\cdot\cos(5\cdot x)}{25\cdot\pi}+\frac{\sin(5\cdot x)}{5}+\frac{\sin(4\cdot x)}{4}-\frac{2\cdot\cos(3\cdot x)}{9\cdot\pi}+\frac{\sin(3\cdot x)}{3}+\frac{\sin(2\cdot x)}{2}-\frac{2\cdot\cos(x)}{\pi}+\sin(x)+\frac{3\cdot\pi}{4}$$

As mathematics teachers, we are comfortable with this procedure and don't see any reason to stop using it. But on the computer algebra side, being able to automate this procedure is something interesting.

We are still teaching some integration techniques despite the fact that the CAS has a built-in integrator! So, why not define, in Nspire CAS, a "Fourier" function like the one *DERIVE* has?

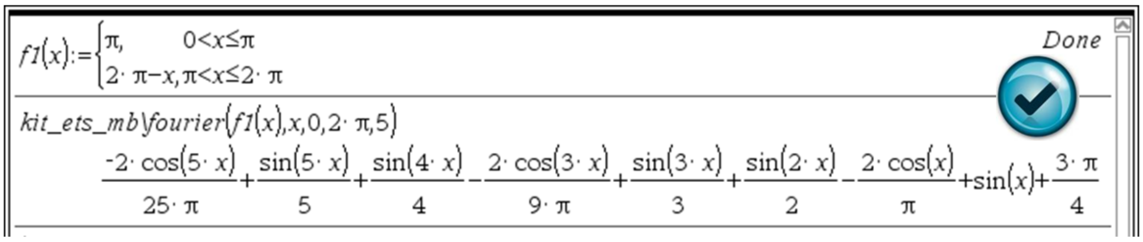
The main goal is to have a Fourier series function able to work in exact mode for piecewise signals. This is where the function `integral_mcx_d` will be useful, replacing the TI's built-in integrator.

So, we have defined a "Fourier series function" in Nspire CAS. Using the same syntax as *DERIVE* and replacing the built-in TI integrator by the `integral_mcx_d` function.

Let's check the result:

```
#1:   f(x) := π·χ(0, x, π) + (2·π - x)·χ(π, x, 2·π)

#2:   FOURIER(f(x), x, 0, 2·π, 5)
```

#3: $$-\frac{2\cdot COS(5\cdot x)}{25\cdot\pi}+\frac{SIN(5\cdot x)}{5}+\frac{SIN(4\cdot x)}{4}-\frac{2\cdot COS(3\cdot x)}{9\cdot\pi}+\frac{SIN(3\cdot x)}{3}+\frac{SIN(2\cdot x)}{2}-\frac{2\cdot COS(x)}{\pi}+SIN(x)+\frac{3\cdot\pi}{4}$$



$$f1(x):=\begin{cases}\pi, & 0<x\le\pi\\ 2\cdot\pi-x, & \pi<x\le 2\cdot\pi\end{cases}$$ Done

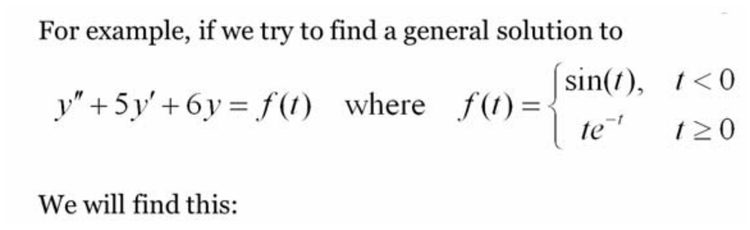
kit_ets_mb\fourier(f1(x),x,0,2·π,5)

$$\frac{-2\cdot\cos(5\cdot x)}{25\cdot\pi}+\frac{\sin(5\cdot x)}{5}+\frac{\sin(4\cdot x)}{4}-\frac{2\cdot\cos(3\cdot x)}{9\cdot\pi}+\frac{\sin(3\cdot x)}{3}+\frac{\sin(2\cdot x)}{2}-\frac{2\cdot\cos(x)}{\pi}+\sin(x)+\frac{3\cdot\pi}{4}$$

Let us mention another interesting application.

With the command “deSolve”, Nspire CAS can easily solve second order differential equations with constant coefficients … as long as the RHS of the differential equation consists of a single piece.

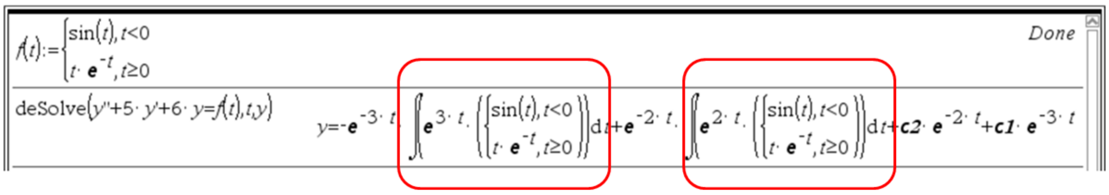
Nspire CAS solves a linear second order ODE by using the method of variation of parameters. This method involves computing integrals.



For example, if we try to find a general solution to

$$y'' + 5y' + 6y = f(t) \quad \text{where} \quad f(t) = \begin{cases} \sin(t), & t < 0 \\ te^{-t} & t \geq 0 \end{cases}$$

We will find this:



$f(t) := \begin{cases} \sin(t), & t<0 \\ t \cdot e^{-t}, & t \geq 0 \end{cases}$ Done

deSolve(y''+5·y'+6·y=f(t),t,y)

$y = -e^{-3 \cdot t} \cdot \int e^{3 \cdot t} \cdot \left(\begin{cases} \sin(t), & t<0 \\ t \cdot e^{-t}, & t \geq 0 \end{cases} \right) dt + e^{-2 \cdot t} \cdot \int e^{2 \cdot t} \cdot \left(\begin{cases} \sin(t), & t<0 \\ t \cdot e^{-t}, & t \geq 0 \end{cases} \right) dt + c2 \cdot e^{-2 \cdot t} + c1 \cdot e^{-3 \cdot t}$

Observe the two integrals that Nspire CAS can’t compute.

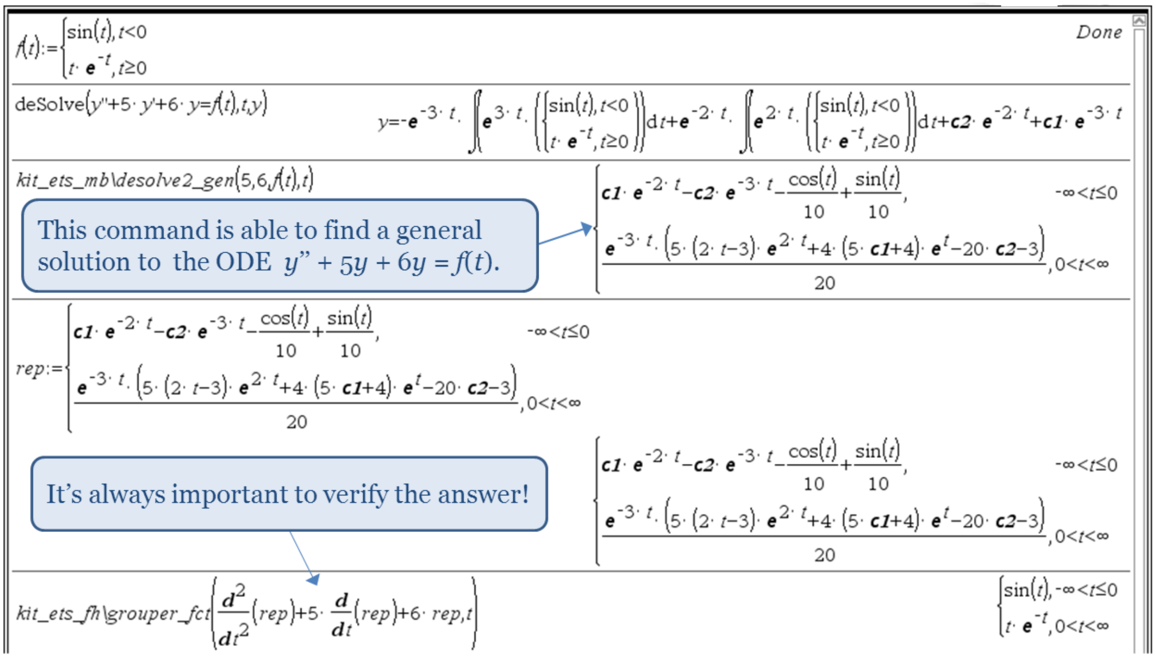
When the RHS of an ODE is piecewise, Laplace transforms are usually used. But in this example, *t* can accept negative values.

With its command “DSOLVE2”, *DERIVE* can solve the former ODE because *DERIVE* can compute the last integrals.

So, we have programmed our own `desolve2_gen` command.

Our method is still using the variation of parameters but the built-in integrator of Nspire CAS is re-placed by Fred’s function `integral_mcx_d`.

Let us do the example once more:



$f(t) := \begin{cases} \sin(t), & t<0 \\ t \cdot e^{-t}, & t \geq 0 \end{cases}$ Done

deSolve(y''+5·y'+6·y=f(t),t,y)

$y = -e^{-3 \cdot t} \cdot \int e^{3 \cdot t} \cdot \left(\begin{cases} \sin(t), & t<0 \\ t \cdot e^{-t}, & t \geq 0 \end{cases} \right) dt + e^{-2 \cdot t} \cdot \int e^{2 \cdot t} \cdot \left(\begin{cases} \sin(t), & t<0 \\ t \cdot e^{-t}, & t \geq 0 \end{cases} \right) dt + c2 \cdot e^{-2 \cdot t} + c1 \cdot e^{-3 \cdot t}$

kit_ets_mb\desolve2_gen(5,6,f(t),t)

$\begin{cases} c1 \cdot e^{-2 \cdot t} - c2 \cdot e^{-3 \cdot t} - \frac{\cos(t)}{10} + \frac{\sin(t)}{10}, & -\infty < t \leq 0 \\ \frac{e^{-3 \cdot t} \cdot \left(5 \cdot (2 \cdot t - 3) \cdot e^{2 \cdot t} + 4 \cdot (5 \cdot c1 + 4) \cdot e^{t} - 20 \cdot c2 - 3\right)}{20}, & 0 < t < \infty \end{cases}$

This command is able to find a general solution to the ODE y” + 5y + 6y = f(t).

$rep := \begin{cases} c1 \cdot e^{-2 \cdot t} - c2 \cdot e^{-3 \cdot t} - \frac{\cos(t)}{10} + \frac{\sin(t)}{10}, & -\infty < t \leq 0 \\ \frac{e^{-3 \cdot t} \cdot \left(5 \cdot (2 \cdot t - 3) \cdot e^{2 \cdot t} + 4 \cdot (5 \cdot c1 + 4) \cdot e^{t} - 20 \cdot c2 - 3\right)}{20}, & 0 < t < \infty \end{cases}$

$\begin{cases} c1 \cdot e^{-2 \cdot t} - c2 \cdot e^{-3 \cdot t} - \frac{\cos(t)}{10} + \frac{\sin(t)}{10}, & -\infty < t \leq 0 \\ \frac{e^{-3 \cdot t} \cdot \left(5 \cdot (2 \cdot t - 3) \cdot e^{2 \cdot t} + 4 \cdot (5 \cdot c1 + 4) \cdot e^{t} - 20 \cdot c2 - 3\right)}{20}, & 0 < t < \infty \end{cases}$

kit_ets_fh\grouper_fct$\left(\frac{d^2}{dt^2}(rep) + 5 \cdot \frac{d}{dt}(rep) + 6 \cdot rep, t \right)$

$\begin{cases} \sin(t), & -\infty < t \leq 0 \\ t \cdot e^{-t}, & 0 < t < \infty \end{cases}$

Conclusion

- At ÉTS, we have adopted TI technology. It started in 1999 with the TI-92 Plus; then V200 in 2002 and Nspire CAS CX since in 2011.
- The CAS system is appropriate for engineering mathematics at the undergraduate level.
- But many mathematics teachers are still using CAS software like *Maple* or *DERIVE*; as a consequence, they often want Nspire CAS to be able to perform as well as these systems!
- In the past two years, we have been asking TI to launch a new OS version of their CAS system.
- For the moment, most of their efforts have been on the side of the overall interface of Nspire CAS.
- This is correct. But, for mathematicians, the CAS engine should be ranked first.
- "If you want something done right, do it yourself".
- The mathematicians needed the help of a programmer. Frédérick Henri started to work with us.
- A new team of researchers was formed. In this talk, we showed some results of our collaboration.
- With these new functions programmed by Frédérick Henri, the built-in integrator of Nspire CAS can now be extended to products of piecewise functions.
- One consequence we showed was the definition of a "Fourier function" similar to *DERIVE*'s one.
- And when the TI built-in integrator will be able to integrate symbolically piecewise expressions, its "deSolve" command will become better.

The tns file "Kit_ETS_FH" is available for download at

http://www.seg.etsmtl.ca/nspire/COURS/Kit_ETS_FH.tns

The Website http://seg-apps.etsmtl.ca/nspire/

also contains many examples for using TI-Nspire CAS for undergraduate engineering studies.

Brussels Gate in Dendermonde, Belgium, revisited

Dr.-Ing. E. Lantschoot, Lantschoot_Weisel@web.de

The topic "**Brussels Gate**" (DNL #89) keeps me busy. First of all, I wanted to know the names of the military architects who built these three gates (Brussels, Mecheln and Ghent, the latter completey dismantled in 1916). An excellent source was the Internet site which provided the name of captain-engineer Cornelis Alewyn (https://inventaris.onroerenderfgoed.be/dibe/relict/48864).

Then, I tried to find out more about this person: curriculum, publications, etc. Alewyn lived between 1788 and 1839; in 1808 he published, in Latin, a treatise on curved lines (*De lineis spiralibus, Tentamen academicum*), substantiating his expertise in mathematics[*]. No other publications were found.

Therefore, I presumed that captain Alewyn did not invent this type of drawbridge, that he relied on existing techniques, but where did those techniques come from?

Having some knowledge about the history of civil and military engineering, I felt that France must have been the source. Indeed, french military engineering features, in the footsteps of Vauban (1633-1707), an uninterrupted succession of engineers (to name a few: Belidor (1697-1761), Monge (1746-1818), Poncelet (1788-1867), and others) who were not only prominent in their achievements, but who developed new designs and applied the most recent methods of calculus to them. In addition to that, Holland had been under the rule of Napoleon during the period 1795-1813, and it can be assumed that dutch professionals like captain-engineer Alewyn were familiar with the French publications on the subject of their activities.

A drawbridge in French is a "pont-levis". The French Wikipedia site http://fr.wikipedia.org/wiki/Pont-levis provides an overview of all different types of drawbridges, and classifies them according to their system of counterweight. The design of the Brussels gate is found in § 2.6, "*constant counterweight on a guiding rail*" and it is attributed to Bernard de Belidor (1697-1761). Note: the Wikipedia site must be read in French; had we selected another language, we would have obtained one more discussion of the topic in the new language, not a translation of the French original. The publications of Belidor can be consulted in digitalized form in http://gallica.bnf.fr/ and his derivation of the drawbridge curve can be found in "*La Science des Ingenieurs*", Paris 1729, book 4, chapt. 5. He called it a Sinusoid, although by the nomenclature of today it is an Oval of Descartes, or a subset thereof: Snail of Pascal or Cardioid. Another source for the derivation of the Belidor curve is a publication of J.-V. Poncelet, "Traîté de la construction des ponts-levis", Paris 1845, which can be consulted in digitalized form under http://www.e-rara.ch/doi/10.3931/e-rara-36; the Belidor curve is discussed on pdf-page 18, the figure appears on p. 64. An interesting sideline: where Belidor in 1729 works with "livres" (pounds) and "pieds" (feet), Poncelet complies with the (in the meantime, 1795, legalized) metric system. By the way, Poncelet is the man who invented the projective geometry, when in captivity in Russia during the campaign of Napoleon,

The Belidor counterweight system was applied occasionally (but without much success) to street-wide bridges in the USA; examples can be seen in http://bridgehunter.com/category/tag/belidor-bascule/ or in http://historicbridges.org/, search for "Belidor".

[1] You can download the Latin book from http://www.e-rara.ch/zut/content/titleinfo/1028122. (Chapters are: *Tangentis Applicationes*, *De Circulo Osculatorio*, *De Spiralium Quadratura*, ...)I found another book written by Alewyn: *Beschrijving van een ontwerp van sluizen met gekoppelde deuren, welke bij alle waterstanden geheel of gedeeltelijk geopend en wederom gesloten kunnen worden*. Josef

The Descartes Oval

This curve describes the movement of the counterweight in a Belidor drawbridge system. We are using the notations of R. Ferreol in http://mathcurve.com/courbes2d/pontlevis/pontlevis.shtml.[*]

- **p** : weight of the bridge [N]
- **q** : counterweight [N]
- **a** : length of the bridge [m]
- **b** : distance between the plane of the bridge and the center of gravity [m]
- **h** : distance of the street level from the deflection sheave O which is the origin of the coordinates [m]
- **l** : **total length** of the rope [m]
- **fr** : free rope = lenght of rope protruding at the city side when bridge is down [m]. Is also the distance OS.
- **rho**, **θ** : polar coordinates of the counterweight = point B on the Belidor curve [m, rad]. In contrast with R. Ferreol we define **θ** in the usual way, i. e. from the x-axis (**θ = 0**) and turning to the left.
- **α** : angle OHT between the bridge and the gate [rad]
- O : Deflection sheave
- H : hinge of the bridge
- T : tip of the bridge
- S : highest point of the Belidor curve
- W : a weight simulating the force of the winch.

The fundamental consideration is that the equilibrium of the point B on the Belidor curve implies that the increases and decreases in potential energy, which arise when **rho** becomes larger, do have a constant sum.

The graph depicts a most general situation: **a** and **h** are not equal, there is some free rope **fr** at the city side when the bridge is down, and a constant additional force **w** is generated in the rope by the pull of a weight **w** (we will later show to what use). In our mind, we now stretch **rho** from **rho = fr** to some arbitrary higher value, and observe four quantities:

1) Point T moves upwards, its potential energy increases by **p/2∗a∗cos(α)**,

2) Point B originally was at S and possessed a potential energy of **–q∗fr**,

3) When B comes down from S to B, there is a decrease in potential energy of **q∗rho∗sin(θ)**.

4) The length of the second rope from O to W is immaterial; if W originally was at O with 0 potential energy, it now is at **–(rho – fr) ∗w**.

[*] http://mathcurve.com/ is a great website. I recommend visiting this site (if you are interested in curves and surfaces), … Josef

By stating that the sum of those four contributions equals 0 we obtain the equation

(1) **sumpot = p/2*a*cos(α) – q* (l-√(a^2+h^2)) + q*rho*sin(θ) – (rho – l +√(a^2+h^2)) *w = 0**.

This sum is computed, in algebraic notation, by the program **oval ()**. If numerical values are desired, we call **data()** and answer the questions. **belidor()** finally applies statistical methods to the results. But yet back to **sumpot**.

Applying the cosine rule to the triangle OHT, we can eliminate cos(α). **sumpot** now appears in terms of a, h, p, q, and l, which are the design parameters of the engineer.

sumpot = **0** is a quadratic in **rho** with **θ** as a parameter. However, it is still an equation with dimension energy; what we are looking for is an equation with dimension length. We can multiply both sides of **sumpot** = **0** by any non-zero quantity, and **–h/(l*p)** is a good candidate. We thus obtain

(2) **sumdis = sumpot*h/(l*p) = 0**.

This is a quadratic in **rho**, with **θ** as a parameter. It is a physical implementation of a class of mathematical curves known as "**Oval of Descartes**". These curves are very important in mechanical and optical engineering, and no one less than the young James Clerk Maxwell started his career with an article on them (1846).

Suppose now, that we list the coefficients of the powers of **rho** of (2). This is easily done by the command **polycoeffs(sumdis,rho)**. This command delivers a three-item list **desclst** and the Oval of Descartes appears as:

(3) **desclst[1] * rho^2 + desclst[2] * rho^1 + desclst[3] * rho^0 = 0**.

Its two roots **rt1** and **rt2** are found by the command **zeros(sumdis, rho)**.When plotted, the oval of Descartes appears as two nested ovals, each one corresponding to one of the roots. The root which generates the outer oval is the one which pertains to our problem.

If we have supplied new data with the program **data()**, and if we turn to the graph, we can play around with the counterweight **q**, and we will soon observe, that the selected value **fr** is implemented as desired for values of **q** above the minimum value **qmin**, but not implemented as desired for values of **q** below **qmin**.

Thinking of it as an engineer, it makes no sense to use a counterweight lighter than **qmin**, because the equilibrium could not be achieved.

But how is this stated in mathematical terms? Suppose **rt1** is the outer oval. If not, use **rt2**. We suggest that you take the following steps:

1) **rt1**↵ on the scratchpad. Its formula appears as a numerical expression containing **sin(θ)**.
2) **DelVar q**↵: The numerical value of **q** is taken away, and replaced by **q** as a variable.
3) Now we apply a condition: **rt1|θ = –π/2** and **rt1** appears in the following form:

(4) **rt1|θ=-π/2 = factor * (|q – qmin| – q + constant)**.

If **q>qmin**, we may forget the || signs, and see that **q** disappears from the expression. Otherwise, **q** remains in the formula.

If we look at the outer oval **rt1** as a whole (modify the scale of the graph to obtain the whole outer oval) it comes to our mind, that the curve has about the shape of a degenerate form of Descartes, viz. the Snail of Pascal, which occurs when

desclst[3] = 0. The Snail of Pascal is usually written in the standardized form

(5) $\mathbf{rhop = 2 * l * (1 + 2 * (h/l) * (q/p) * \sin(\theta))}$.

and the coefficient of $\mathbf{\sin(\theta)}$ is usually referred to as **e**

(6) $\mathbf{e = 2 * (h/l) * (q/p)}$.

With foresight, the value of **e** was already calculated by **data()** and **rhop** is contained in the list of polar curves displayed on the graph, but must be activated by a check sign. We see that both curves fit very well in general, but unfortunately rather poorly in the area around S.

For the particular example of Dendermonde with $\mathbf{a = h = 3.4}$ m, $\mathbf{p = 12000}$ N, $\mathbf{fr = 0.9}$ m and $\mathbf{q = 8500}$ N the street is reached at a point $\mathbf{G = (4.80, -3.4)}$ m, which is well in accordance with the observations on the spot. But, be careful: **HG** must = **l**.

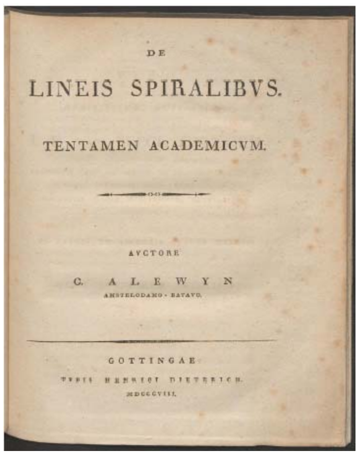
Any further study of the curves of Descartes and Pascal, although very interesting, would go far beyond the scope of this notice. Fortunately many good textbooks on plane curves are available in bookshops and libraries.

Belidor is satisfied

In the meantime, it appears that the most important design parameter is **q**, the weight of the counterweight. The engineer draws a circle with length **l** around O, sees where it intersects the Belidor curve, and then adjusts **q** so that the intersection comes down to the street level.

What we will do now is to prepare the Belidor curve for <u>regression analysis</u>. The polar coordinates for 8 points are converted to two lists **rx** and **ry** of cartesian coordinates. To these lists we apply the <u>quartic regression</u> and obtain the exact 4th-degree equation in x (*) of the Belidor curve. Then we apply the <u>quadratic regression</u> and obtain **eq2**, the best possible parabolic approach to Belidor.The differences are hard to notice, but they exist: in the first half Belidor is under the parabola, in the second half above it and then under it again. The check is easy: it suffices to plot **100 ∗ abs(eq4-eq2)** and the differences can be seen in centimeters. They are, for $\mathbf{\sqrt{(l^2-h^2)} > x > 0}$ at worst about 10 cm.

I am e-mailing a hint to the Belidor drawbridges at Dendermonde to "mathourist.canalblog.com". If you want to know what this Internet site has to offer, do not forget to enable your browser to access French sites.


DE

LINEIS SPIRALIBVS.

TENTAMEN ACADEMICVM.

AVCTORE

C. ALEWYN

AMSTELODAMO - BATAVO.

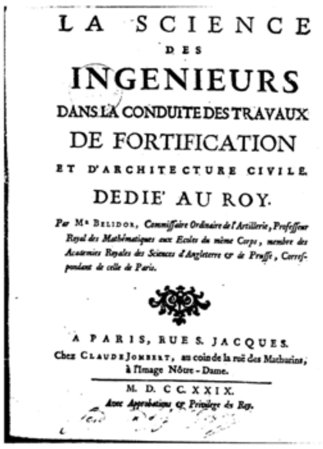
GOTTINGAE

TYPIS HENRICI DIETERICH.

MDCCCVIII.

Alewyn's *De Lineis Spiralibus*

Bernard de Belidor


LA SCIENCE

DES

INGENIEURS

DANS LA CONDUITE DES TRAVAUX

DE FORTIFICATION

ET D'ARCHITECTURE CIVILE.

DEDIE' AU ROY.

Par Mr BELIDOR, *Commiſſaire Ordinaire de l'Artillerie, Profeſſeur Royal des Mathématiques aux Ecoles du même Corps, membre des Academies Royales des Sciences d'Angleterre & de Pruſſe, Correſpondant de celle de Paris.*

A PARIS, RUE S. JACQUES.

Chez CLAUDE JOMBERT, au coin de la ruë des Mathurins, à l'Image Nôtre-Dame.

M. D. CC. XXIX.

Avec Approbation, & Privilege du Roy.

Belidor's *La Science …*

Alewyn´s problem solved mathematically

Beautiful as it is mathematically speaking, the Belidor curve is not the proper practical solution. Any climatic disturbance, rain, snow, wind would move and shatter the bridge uncontrollably. Therefore, Alewyn had to introduce a disequilibrium, which would (a) forcefully push the bridge on the bridgehead when in the horizontal, and which (b) would have to keep the bridge forcefully upright when in the vertical position.

To achieve these two goals Alewyn took advantage of the fact that (in practice) the center of gravity does not ly on the surface of the bridge but rather b m below it, and reasoned that the equilibrium of energies would have to be obtained when the energy exerted by a winch was included in **sumpot**. If the force **w** of the winch were not present, the brigde would ly flat forcefully, because the counterweight would contribute less force than required for equilibrium, and if the bridge stood upright, the force of the winch would keep it in equilibrium.

We can simulate the force **w** of the winch by a weight W moving up and down and hanging on a second rope that begins at the tip of the bridge, then goes over the deflection sheave and moves straight down to W. Mathematically speaking, this second rope can be treated as an additional force **w** in the first rope. Assuming that **a** = **h** as in Dendermonde, we now call **data()**. The program asks for **b** (center of gravity below the plane of the bridge) and proposes a sensible value for **w**, viz. the value needed for keeping the bridge in equilibrium when it is in the upright position. Higher values are tolerable (to the detriment of the men who power the winches!) but lower values would not insure the equilibrium anymore. Looking at the formulas, one sees that w plays a role similar to **q**, the counterweight (**).

Finally,

the design invented by Belidor was not very successful. To my esteem, the freedom of the designing engineer was too small: there is only one parameter - I mean **q** - he can play with, and even then only within rather narrow limits.

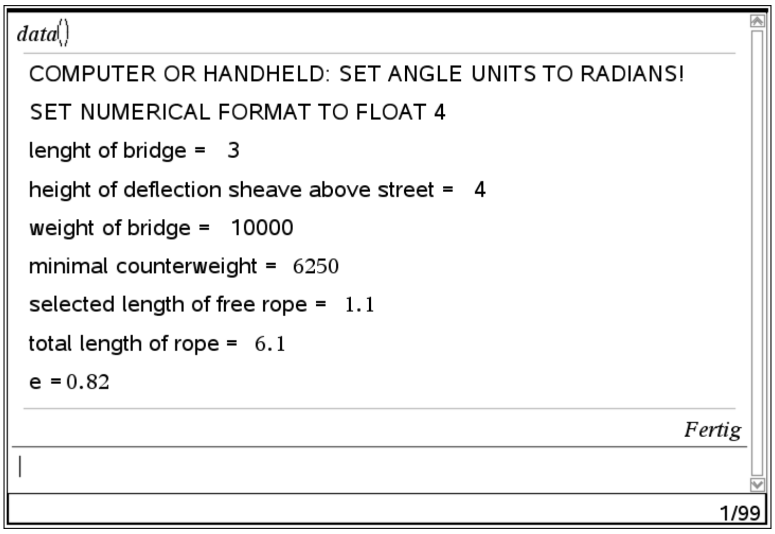
Notes

(*) It is not immedialtely obvious that Descartes (when stated in Cartesian coordinates) is a 4-th order equation in x, but if we plot that equation, we can see that the x-axis is crossed never more than 4 times.

(**) The numerical results are approximations only, because we act as if **p/2** was still located at the tip T of the plane of the bridge, but all we wanted to do was to show the principle of Alewyn's solution.

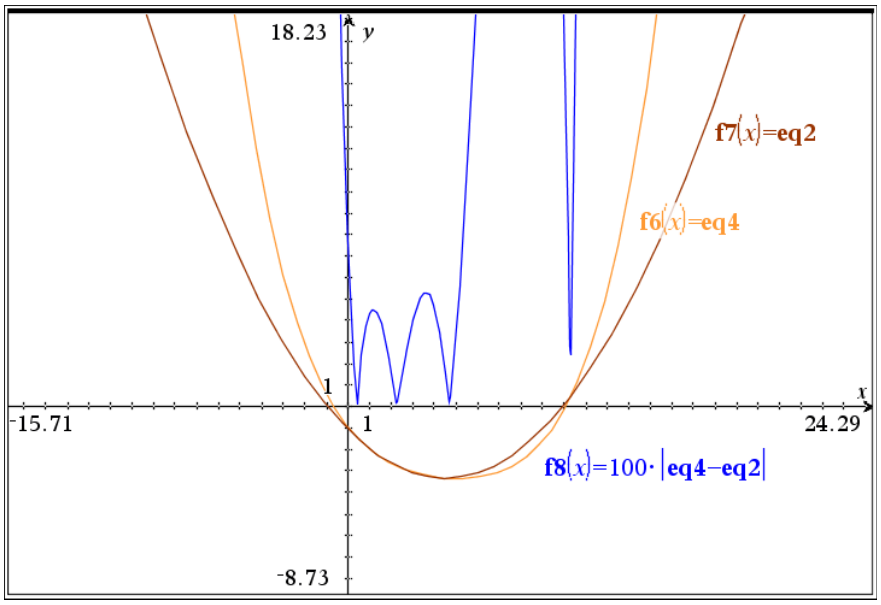
More Links (German):
http://de.wikipedia.org/Bernard_de_Bélidor
http://www.deutsches-museum.de/bibliothek/unsere-schaetze/technikgeschichte/belidor/

```
Define data()=
Prgm
:© Supplies or modifies the data
:DelVar a,b,h,fr,p,q,qmin,w
:setMode({1,4,2,1})
:© Typical values are entered
:3→a: 4→h: 0→fr: 10000→p: 0→b: 0→w
:Disp "COMPUTER OR HANDHELD: SET ANGLE UNITS TO RADIANS!"
:Disp "SET NUMERICAL FORMAT TO FLOAT 4"
:Request "lenght of bridge =  ",a
:Request "height of deflection sheave above street =  ",h
:10000→p © weight of bridge
:Request "weight of bridge =  ",p
:If a≠h Then: Goto next: EndIf
:Request "center of gravity below bridge plane =  ",b
:((p*b)/(h))→w
:Request "force of winch = ",w
:Lbl next
:© provisory value of q = value for equilibrium at θ=−((π)/(2))
:q:=((√(a^(2)+h^(2))*p)/(2*h))−w
:qmin:=q
:Disp "minimal counterweight =  ",q
:Request "length of free rope between 0 and "&string(h)&" ?",fr,0
:If fr<0 or fr>h Then: Disp "choice not appropriate":Stop
:Else
:Disp "selected length of free rope =  ",fr
:EndIf
:l:=√(a^(2)+h^(2))+fr
:Disp "total length of rope =  ",l
:e:=approx(2*((h)/(l))*((q)/(p))): Disp "e =",e
:EndPrgm
```



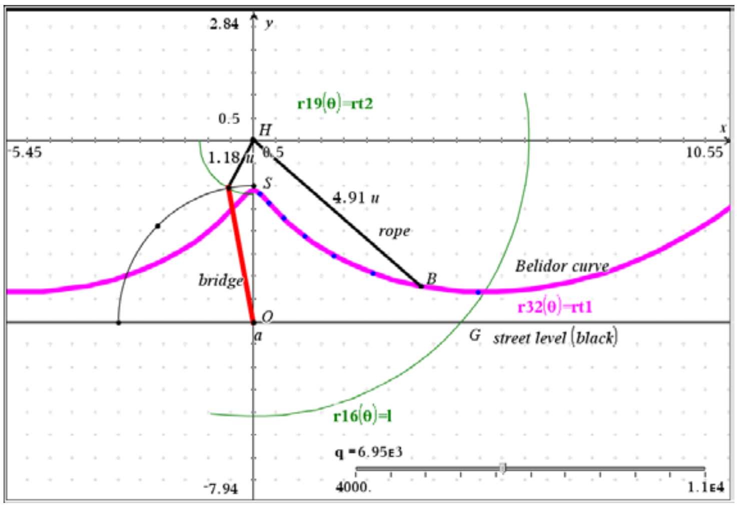
```
data()
 COMPUTER OR HANDHELD: SET ANGLE UNITS TO RADIANS!
 SET NUMERICAL FORMAT TO FLOAT 4
 lenght of bridge =   3
 height of deflection sheave above street =   4
 weight of bridge =   10000
 minimal counterweight =  6250
 selected length of free rope =  1.1
 total length of rope =  6.1
 e =0.82
                                                  Fertig
                                                    1/99
```

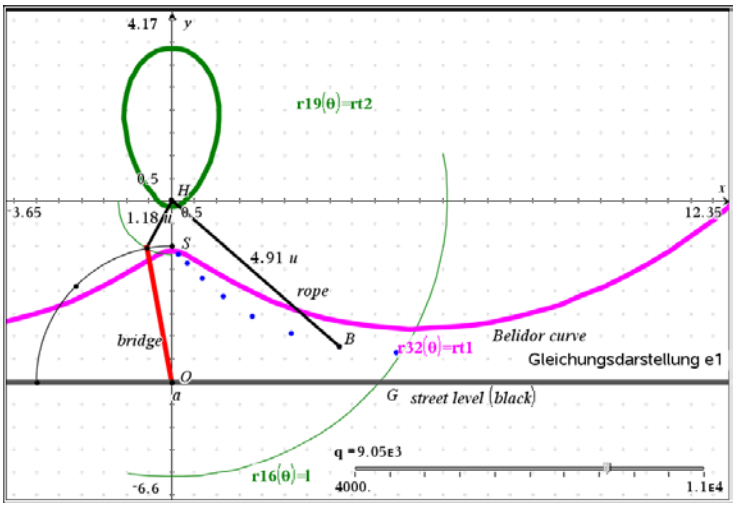
```
Define belidor()=
Prgm
:setMode(2,1)
:rx:=newList(8)
:ry:=newList(8)
:θstep:=((cos⁻¹(((h)/(l))))/(7))
:For j,1,8
:θ:=((⁻π)/(2))+j*θstep
:© select the outer oval, rt1 or rt2
:rx[j]:=rt1*cos(θ)
:ry[j]:=rt1*sin(θ)
:EndFor
:QuartReg rx,ry
:eq4:=stat.RegEqn(x)
:Disp "exact Belidor eqn = ",eq4
:QuadReg rx,ry
:eq2:=stat.RegEqn(x)
:Disp "best parabolic approximation = ",eq2
:EndPrgm^
```



```
Define oval()=
Prgm
:set*mode(2,1)
:Local ant,coa
:Unlock q
:DelVar θ,a,b,fr,h,l,p,q,w
:© cosine rule for OHT. coa stands for cos(α)
:coa:=zeros((l−rho)^(2)−a^(2)−h^(2)+2*a*h*coa,coa)[1]
:
```

```
:sumpot:=((p)/(2))*a*coa+q*(l−√(a^(2)+h^(2)))+q*rho*sin(θ)+(l−√(a^(2)+h^(2))−rho)*w
:Disp "If no interest for Alewyn, disregard the existence of w"
:Disp "sumpot = "
:Disp sumpot
:sumdis:=((⁻sumpot*4*h)/(p))
:Disp "sumdis ="
:Disp sumdis
:desclst:=polyCoeffs(sumdis,rho)
:Disp "Coefficients of Descartes´ equation with
rho as the variable and θ as a parameter
describing both ovals: = desclst"
:Disp desclst
:Disp "Inner and outer oval are ="
:rt1:=zeros(polyEval(desclst,rho),rho)[1]
:rt2:=zeros(polyEval(desclst,rho),rho)[2]
:Disp "rt1 =  ",rt1: Disp "rt2 =  ",rt2
:© computing an approximation by the snail of Pascal
:rhop:=2*l*(1+2*((h)/(l))*((q)/(p))*sin(θ))
:EndPrgm
```





Quasi-Monte-Carlo-Methods

Josef Böhm, Würmla, Austria

Instead of an abstract

A quasirandom or low discrepancy sequence, such as the Faure, Halton, Hammersley, Niederreiter or Sobol sequences, is "less random" than a pseudorandom number sequence, but more useful for such tasks as approximation of integrals in higher dimensions, and in global optimization. This is because low discrepancy sequences tend to sample space "more uniformly" than random numbers. Algorithms that use such sequences may have superior convergence. Faure sequences, in particular, seem to have become popular in mathematical finance simulations. [8]

Introduction

Monte-Carlo-methods (Monte-Carlo-simulations) are widely used for simulations based on inspecting and evaluating many randomly produced samples. The best known example – even in secondary school – is approximating π by counting "raindrops" fallen on a sqare with an inscribed quarter of a circle.

MC-methods are used in statistics, physics, medicine, ..., and in financial mathematics (option pricing, estimating risks, insurance, „,,), too. You can find a good overview and many links in

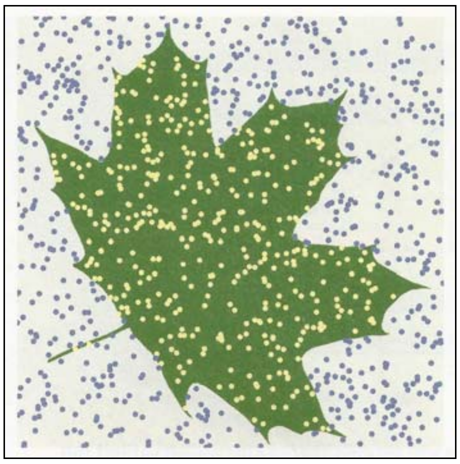
http://en.wikipedia.org/wiki/Monte_Carlo_method.

MC-methods use random numbers produced by random number generators which are in fact producing pseudo random numbers applying a less or more good algorithm. Many programs need a random seed to initiate the agorithm. Same seeds produce identical random number sequences. Some programs – like *DERIVE* – can use the internal CPU-time for starting the process, which gives new sequences whenever called.

Quasi-Monte-Carlo-methods use deterministic sequences instead of random ones. They outperform the MC-methods in convergence and error bounds. Both attributes proved to be important in financial mathematics applications.

Once more the traditional MC-method

I will start reminding on the MC-integration but not referring to the – in the meanwhile maybe boring – example of estimating π.



I was inspired by an article in "Spektrum der Wissenschaft" – which is the German edition of Scientific American – from February 2012 to work with QMC-methods.

The first figure in the journal shows a leaf and how to estimate its area. There are 1024 MC-randomly distributed points in the square and 41.89% of all points are within the area of the leaf.[See end of contribution]

I had liked to use the picture of the leaf as background picture in a *DERIVE* plot window but I was not able to reproduce the shape of the leaf by functions. So I decided to take a star for demonstration purposes. I defined six inequalities for describing an irregular hexagon, calculated the intersection points of the lines and then calculated the area of the polygon.

(I skip presenting the calculation of the vertices.)

```
[i1(x, y) := y - 5·(1 - x) ≥ 0, i2(x, y) := 2·y - 10 + x ≤ 0]

[i3(x, y) := x - y - 1 ≤ 0, i4(x, y) := 2·y + x - 4 ≥ 0]

[i5(x, y) := y - 5·(4 - x) ≤ 0, i6(x, y) := y - 2·x - 2 ≤ 0]

p(x, y) := (i1(x, y) ∧ i2(x, y) ∧ i3(x, y)) ∨ (i4(x, y) ∧ i5(x, y) ∧ i6(x, y))
```

$$a\_tr(P, Q, R) := \frac{1}{2}\cdot\left|P_1\cdot(Q_2 - R_2) + Q_1\cdot(R_2 - P_2) + R_1\cdot(P_2 - Q_2)\right|$$

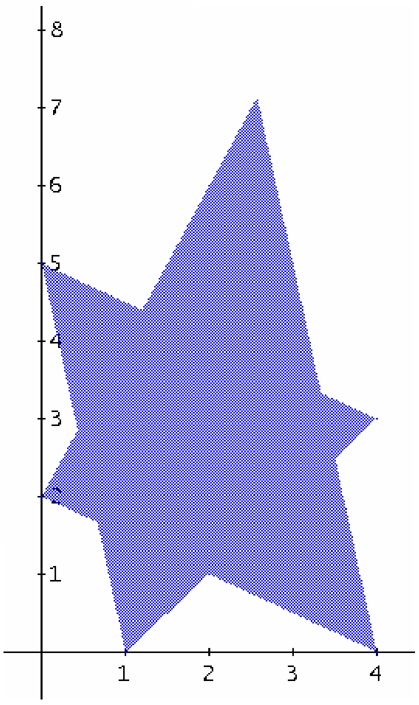
$$a\_tr\left([0, 2], [4, 0], \left[\frac{18}{7}, \frac{50}{7}\right]\right) = \frac{90}{7}$$

$$a\_tr\left(\left[\frac{3}{7}, \frac{20}{7}\right], [0, 5], \left[\frac{6}{5}, \frac{22}{5}\right]\right) = \frac{81}{70}$$

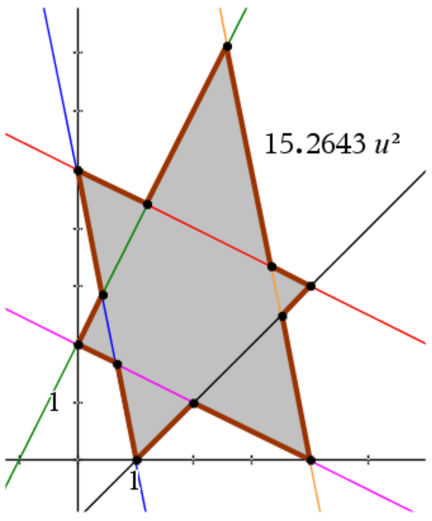
$$a\_tr\left(\left[\frac{10}{3}, \frac{10}{3}\right], [4, 3], \left[\frac{7}{2}, \frac{5}{2}\right]\right) = \frac{1}{4}$$

$$a\_tr\left(\left[\frac{2}{3}, \frac{5}{3}\right], [1, 0], [2, 1]\right) = 1$$

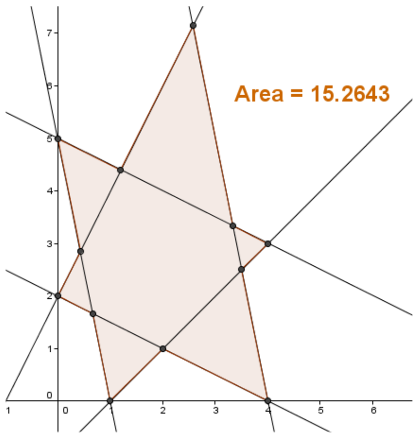
$$\frac{90}{7} + \frac{81}{70} + \frac{1}{4} + 1 = 15.26428571$$



Calculation of the area would have been easier if I had used TI-Nspire (see left below). The result is the same, fortunately!



TI-Nspire



GeoGebra

To go for sure I plotted the hexagon once more using GeoGebra – the area of the hexagon corresponds with the *DERIVE*-calculated value again.

I don't want to loose time complaining about accurracy but proceed by estimating the area of the hexagon applying the traditional Monte-Carlo-method using *DERIVE*'s random numbers.

3000 points should be sufficiently enough!

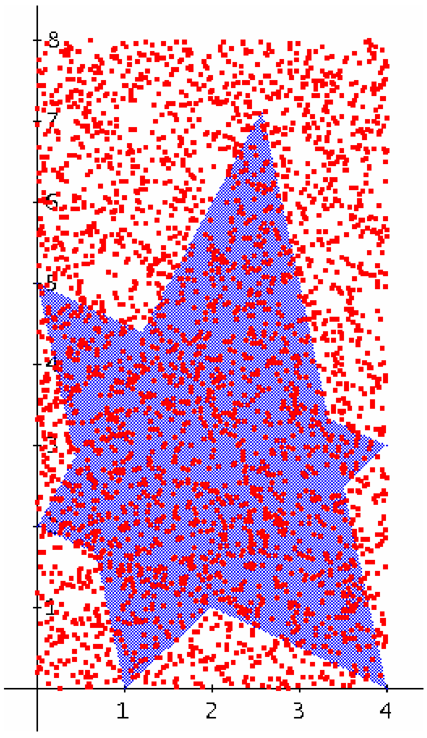
```
RANDOM(0) = 2208522468

mc_polpts := VECTOR([4·RANDOM(1), 8·RANDOM(1)], k, 3000)
```

$$\mathrm{DIM}(\mathrm{SELECT}(p(v_1, v_2), v, \mathrm{mc\_polpts})) = 1412$$

$$\mathrm{SOLVE}\left(\frac{32}{a} = \frac{3000}{1412}, a\right) = (a = 15.06133333)$$

$$\frac{\mathrm{DIM}(\mathrm{SELECT}(p(v_1, v_2), v, \mathrm{mc\_polpts}))\cdot 32}{3000} = 15.36$$

$$\frac{\mathrm{DIM}(\mathrm{SELECT}(p(v_1, v_2), v, \mathrm{mc\_polpts}))\cdot 32}{3000} = 15.22133333$$



These are the results of three simulations

$$\mathrm{VECTOR}\left(\left|\frac{15.2643 - a}{15.2643}\right|\cdot 100, a, [15.0613, 15.36, 15.2213]\right)$$

```
[1.329900486, 0.6269530866, 0.2817030587]
```

The maximum error is 1.33%. (Please compare with page 40.)

Demands on random numbers

There are three demands on random numbers:

(1) They must not be predictable.

(2) They must not be correlated, i.e. they must be independent of another.

(3) Their distribution must be well-balanced.

True random numbers fullfil all demands, pseudo random numbers follow a strict – more or less complicated – rule, quasi random numbers fullfil only one of the three demands: they are evenly distributed. A measure for the balance of the distribution is the so called *discrepancy*. The discrepancy is important for the possible error. You can find much about this on the respective websites.

The minimised error together with a very much improved convergence made quasi random numbers and the corrresponding Quasi-Monte-Carlo-methods a modern tool for financial mathematics.

The van der Corput Sequence

The Dutch mathematician *Johannes Gualtherus van der Corput* (1890 – 1975) published this low discrepancy sequence over the unit interval in 1935. Its generation is very simple:

Take the sequence of the natural numbers in their binary representation, reverse the digits including the decimal point (i.e. reflect the binary expansion about the decimal point) and then converse back into the decimal system).
The following table demonstrates how to obtain the first 10 quasi random numbers in [0, 1):

integer	binary representation	reversion	quasi random number = = van der Corput sequence
0	0	0.0	0
1	1	0.1	0.5
2	10	0.01	0.25
3	11	0.11	0.5+0.25=0.75
4	100	0.001	0.125
5	101	0.101	0.5+0.125=0.625
6	110	0.011	0.25+0.125 = 0.375
7	111	0.111	0.5+0.25+0.125=0.875
8	1000	0.0001	0.0625
9	1001	0.1001	0.5+0.0625=0.5625
10	1010	0.0101	0.25+0.0625=0.3125
…	…	…	…

To generate N points in the unit square we take x-coordinates $\frac{i}{N}$ with $i = 0, 1, 2, \ldots, N–1$ and the respective y-coordinates are the elements of the *Corput Sequence*. In case of $N = 10$ we would have the points (0,0), (0.1, 0.5), (0.2, 0.25), (0.3, 0.75), …, (0.9, 0.0625). The result is called *Hammersley Set*.

Now let's do this with our CAS:

First of all I tried to transfer the procedure given in the table step by step, i.e. conversing the integer into a binary number (in form of a string), reversing the characters and then evaluating the ones and zeros according to their position:

```
conv(x, b, r) :=
  Prog
    r := ""
    Loop
      If x < b
         RETURN APPEND(IF(MOD(x, b) < 10, STRING(MOD(x, b)), CODES_TO_NAME(55 + MOD(x, b))), r)
      r := APPEND(IF(MOD(x, b) < 10, STRING(MOD(x, b)), CODES_TO_NAME(55 + MOD(x, b))), r)
      x := FLOOR(x, b)

corput(x, b := 2, r, y) :=
  Prog
    y := REVERSE(conv(x, b))
    y := ∑(VECTOR(((NAME_TO_CODES(y))↓j - 48)·b^(-j), j, DIM(y)))
```

$$\text{VECTOR(corput(k), k, 10)} = \left[\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \frac{1}{16}, \frac{9}{16}, \frac{5}{16}\right]$$

As this is not very elegant, I did once more without using the string operations but conversing, reversing and calculating digit for digit:

```
corput_(x, b := 2, p, o) :=
  Prog
    [o := 0, p := 1/b]
    Loop
      If x = 0
         RETURN o
      o := o + MOD(x, b)·p
      x := (x - MOD(x, b))/b
      p := p/b
```

```
corput_seq(n, b := 2) := VECTOR(corput_(k, b), k, n)
```

$$\text{corput\_seq(10)} = \left[\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \frac{1}{16}, \frac{9}{16}, \frac{5}{16}\right]$$

For comparing with *DERIVE*'s (pseudo) random numbers I generate 1200 elements of the Corput-sequence in order to simulate throwing a die 1200 times and observing the outcomes:

```
#8:   dice := VECTOR(RANDOM(6) + 1, k, 1200)
```

Simplify the expression above, remove the huge expression and proceed counting:

```
#9:   VECTOR(DIM(SELECT(k = i, k, dice)), i, 1, 6)

#10:  [189, 197, 205, 212, 198, 199]

#11:  dice_qmc := corput_seq(1200)
```

Simplify again - then you can remove the huge expression, proceed counting

$$\#12:\quad \text{VECTOR}\left(\text{DIM}\left(\text{SELECT}\left(\frac{i}{6} \le k < \frac{i+1}{6}, k, \text{dice\_qmc}\right)\right), i, 0, 5\right)$$

```
#13:  [202, 200, 198, 202, 200, 198]
```

Simplifying #8 and #11 is important for fixing the sample. Otherwise `dice` and `dice_qmc` in expressions #9 and #12 would be generated new at each call within the `VECTOR`-command.

From the Sequence to Quasi Random Points (Hammersley)

There are several ways to create sets of random points in the plane, in 3D-space and – and this is the interesting part for applications in financial mathematics in many dimensions.

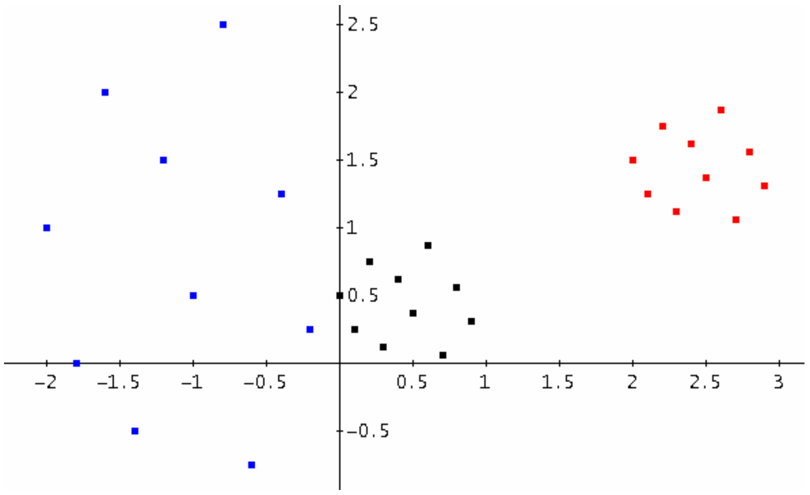
I start presenting the *Hammersley Set* described on page 33.

```
hammersley(n, lb := [0, 0], rt := [1, 1], b := 2, a_, b_, x_, y_) :=
  Prog
    [a_ := rt↓1 - lb↓1, b_ := rt↓2 - lb↓2]
    x_ := [VECTOR(lb↓1 + i/n·a_, i, 0, n - 1)]
    y_ := [VECTOR(lb↓2 + k·b_, k, corput_seq(n, b))]
    APPEND(x_, y_)`
```

(lb = left bottom vertex and rt = right top vertex of the rectangle describing the region in the plane. It is the uit square by default)

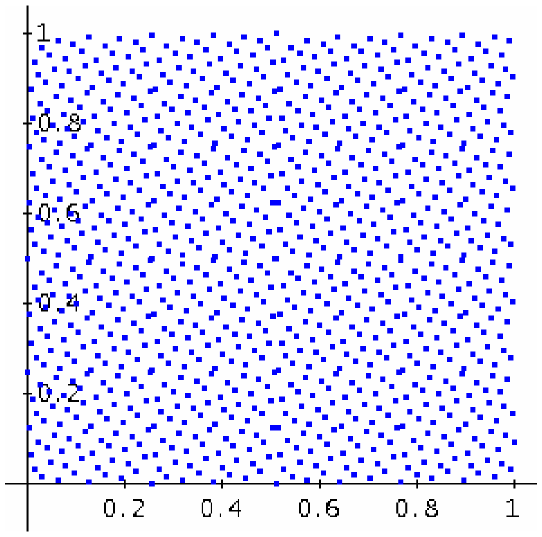
```
[hammersley(10), hammersley(10, [2, 1], [3, 2]), hammersley(10, [-2, -1], [0, 3])]
```

```
[[  0      0.5  ]  [  2     1.5   ]  [ -2      1    ]]
[[ 0.1    0.25  ]  [ 2.1   1.25   ]  [ -1.8    0    ]]
[[ 0.2    0.75  ]  [ 2.2   1.75   ]  [ -1.6    2    ]]
[[ 0.3    0.125 ]  [ 2.3   1.125  ]  [ -1.4  -0.5   ]]
[[ 0.4    0.625 ]  [ 2.4   1.625  ]  [ -1.2   1.5   ]]
[[ 0.5    0.375 ], [ 2.5   1.375  ], [  -1    0.5   ]]
[[ 0.6    0.875 ]  [ 2.6   1.875  ]  [ -0.8   2.5   ]]
[[ 0.7   0.0625 ]  [ 2.7   1.0625 ]  [ -0.6  -0.75  ]]
[[ 0.8   0.5625 ]  [ 2.8   1.5625 ]  [ -0.4   1.25  ]]
[[ 0.9   0.3125 ]  [ 2.9   1.3125 ]  [ -0.2   0.25  ]]
```

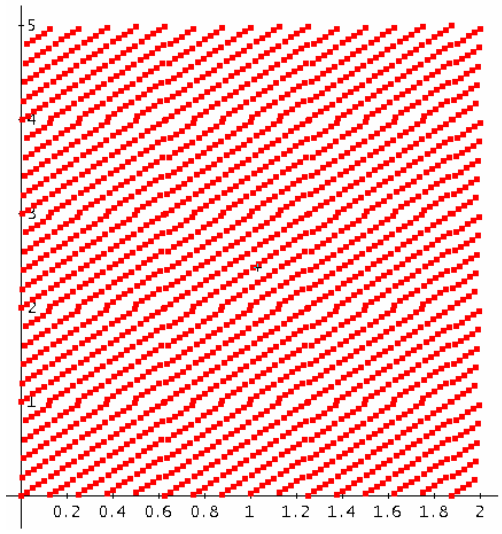


10 quasi random points distributed in three different regions in the plane. (Unit square and base 2 by default). Let’s plot 1000 and 2000 points (base 5) now:

```
hammersley(1000)
```



```
hammersley(2000, [0, 0], [2, 5], 5)
```



Generalisation: Halton Sequence and Halton Set

Why only taking base 2? The *Halton sequence* is an *n*-dimensional generalization of the van der Corput sequence, but instead of using binary representations, a different base (*prime number*) is used for each coordinate. Let's take $p_1 = 2$ and $p_2 = 3$ for *x*- and *y*-axis.

```
halton_seq(n, b) := corput_seq(n, b)
```

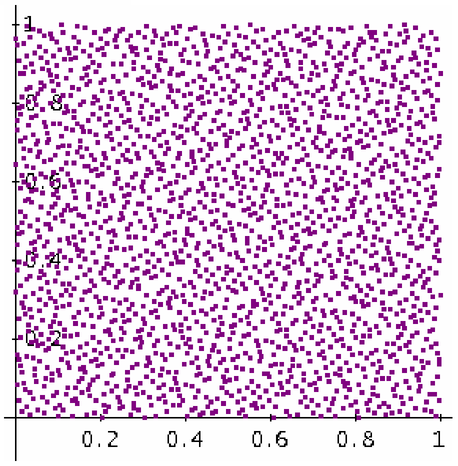
$$\text{halton\_seq}(10,\ 2) = \left[\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}, \frac{1}{16}, \frac{9}{16}, \frac{5}{16}\right]$$

$$\text{halton\_seq}(10,\ 3) = \left[\frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, \frac{8}{9}, \frac{1}{27}, \frac{10}{27}\right]$$

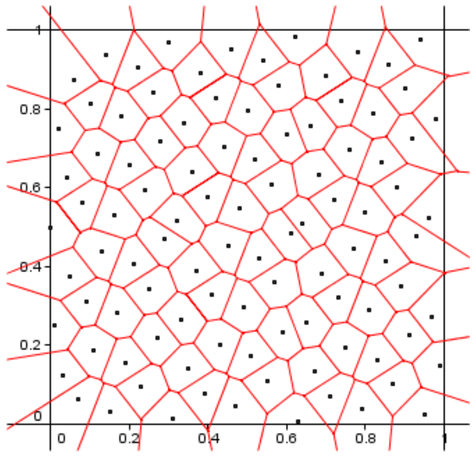
$$\text{halton\_seq}(10,\ 5) = \left[\frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, \frac{1}{25}, \frac{6}{25}, \frac{11}{25}, \frac{16}{25}, \frac{21}{25}, \frac{2}{25}\right]$$

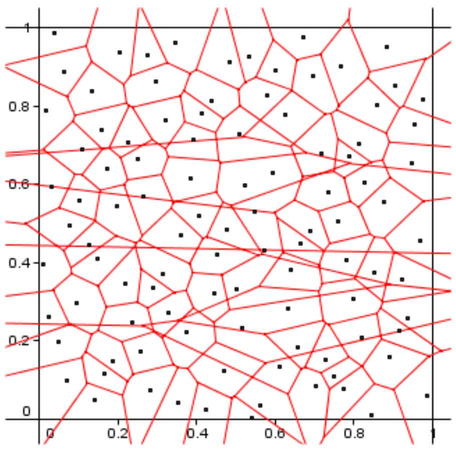
```
halton(n, lb, rt, b1, b2, a_, b_, x_, y_) :=
  Prog
    [a_ := rt↓1 - lb↓1, b_ := rt↓2 - lb↓2]
    x_ := [VECTOR(lb↓1 + k·a_, k, halton_seq(n, b1))]
    y_ := [VECTOR(lb↓2 + k·b_, k, halton_seq(n, b2))]
    APPEND(x_, y_)`
halton(2000, [0, 0], [1, 1], 2, 3)
```

Halton set:

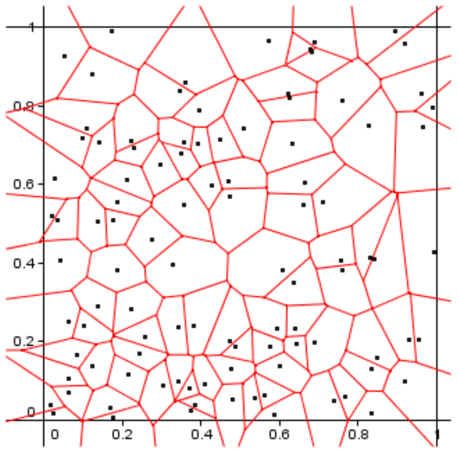


In http://planning.cs.uiuc.edu/node210.html I found a nice figure comparing the Voronoi diagrams of 196 Hammersley points and 196 Halton points and of 196 (pseudo) random points. I will try to produce the diagrams with 100 points only (Hammersley left and Halton right).





I am not so happy with the Halton-Voronoi diagram.



Then I plotted the Voronoi diagram for 100 (pseudo) random points (right) . You can observe the difference concerning the “uniform” distribution.
You may remember Enric Puig’s request on Voronoi diagrams in DNL#89. I can not give an advice how to contruct such a diagram but I can give a hint how to plot it.

I copied the points to GeoGebra which provides a Voronoi[<pointlist>] command and plotted the diagrams as you can see above and right.

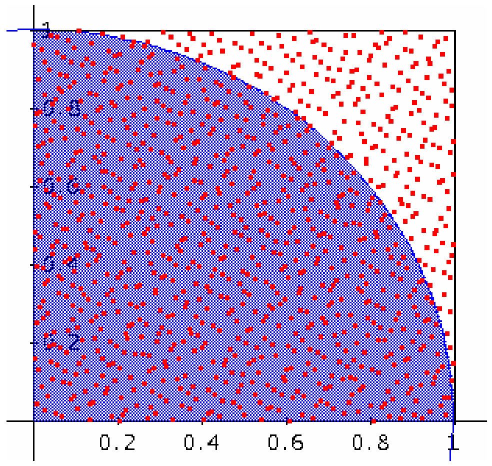
Quasi Monte Carlo vs Monte Carlo

I come back to my introduction talking about approximating π.

```
qmcpts := q_m_halton([0, 0], [1, 1], 2, 3, 1000)
```

$$x^2 + y^2 = 1$$

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$



$$0 \le x \le 1 \wedge 0 \le y \le 1 \wedge x^2 + y^2 \le 1$$

```
qmcpts := halton(1000, [0, 0], [1, 1], 2, 3)
```

$$\text{DIM}\left(\text{SELECT}\left(v_1^2 + v_2^2 \le 1,\ v,\ \text{qmcpts}\right)\right) = 787$$

$$\frac{4 \cdot 787}{1000} = \frac{787}{250}$$

$$\frac{4 \cdot 787}{1000} = 3.148$$

$$\text{DIM}\left(\text{SELECT}\left(v_1^2 + v_2^2 \le 1,\ v,\ \text{qmcpts}\right)\right) = 787$$

$$\frac{4 \cdot 787}{1000} = 3.148$$

```
mcpts := VECTOR([RANDOM(1), RANDOM(1)], i, 1000)
```

$$\text{DIM}\left(\text{SELECT}\left(v_1^2 + v_2^2 \le 1,\ v,\ \text{mcpts}\right)\right) = 800$$

$$\frac{4 \cdot 800}{1000} = 3.2$$

$$\text{DIM}\left(\text{SELECT}\left(v_1^2 + v_2^2 \le 1,\ v,\ \text{mcpts}\right)\right) = 781$$

$$\frac{4 \cdot 781}{1000} = 3.124$$

I repeated the estimation using 5000 points and received 3.1472 (QMC) vs 3.164 (MC).

In my next example I calculate the area between $y = 0$ and $f(x) = \sin(2x) + 1$ for $0 \le x \le \pi/2$.

```
f(x) := SIN(2·x) + 1
```

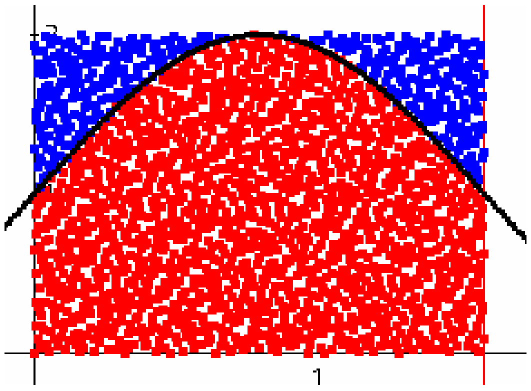
$$\text{sinpts} := \text{halton}\left(2000,\ [0,\ 0],\ \left[\frac{\pi}{2},\ 2\right],\ 2,\ 3\right)$$

$$\text{DIM(SELECT}(f(v_1) \ge v_2,\ v,\ \text{sinpts})) = 1639$$

$$\frac{1639\cdot\pi}{2000} = 2.574535179$$

$$\int_0^{\pi/2} f(x)\,dx = 2.570796326$$

$$\frac{|2.576105975 - 2.570796326|}{2.570796326} = 0.002065371319$$

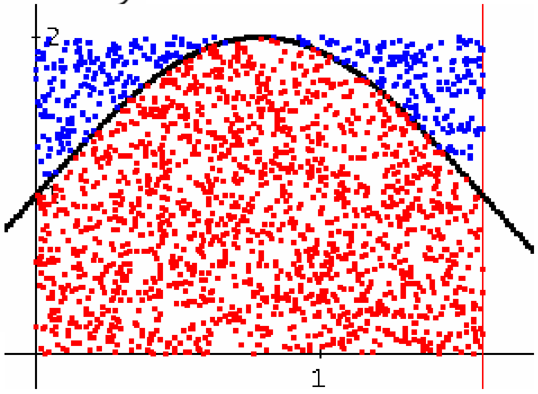


$$\text{mc\_sinpts} := \text{VECTOR}\left(\left[\frac{\pi}{2}\cdot\text{RANDOM}(1),\ 2\cdot\text{RANDOM}(1)\right],\ k,\ 2000\right)$$

$$\text{DIM(SELECT}(f(v_1) \ge v_2,\ v,\ \text{mc\_sinpts})) = 1634$$

$$\frac{1634\cdot\pi}{2000} = 2.566681197$$

$$\frac{|2.566681197 - 2.570796326|}{2.570796326} = 0.001600721519$$

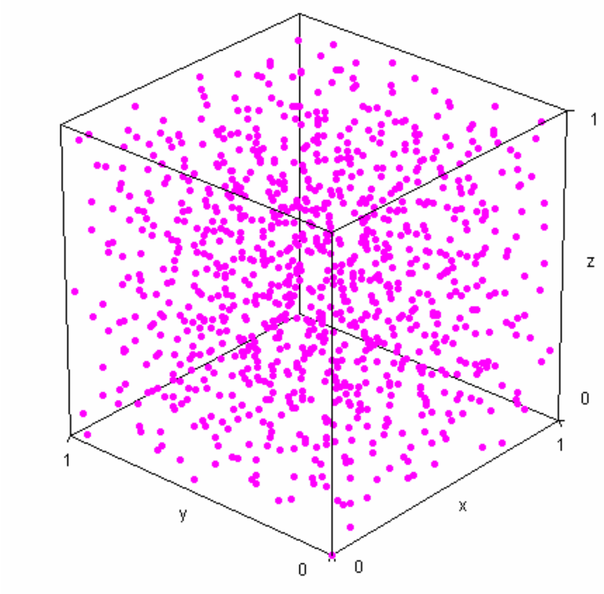


Quasi Monte Carlo in 3D Space

The applications in financial mathematics are very often evaluations of multiple integrals. So we have to leave the two dimensions and generalize for more dimensions. We will use the Halton sequences to calculate the volume under a surface.

There are QMC-methods which are better suitable if many dimensions are needed. I will present them later. `halton3d` produces a list of *n* points The region is defined by three vertices of a cuboid (lfb = left front bottom, rfb = right front bottom, lbt = left back top).

```
halton3d(n, lfb, rfb, lbt, b1, b2, b3, a_, b_, c_, x_, y_, z_) :=
  Prog
    [a_ := rfb↓1 - lfb↓1, b_ := lbt↓2 - lfb↓2, c_ := lbt↓3 - lfb↓3]
    x_ := [VECTOR(lfb↓1 + k·a_, k, halton_seq(n, b1))]
    y_ := [VECTOR(lfb↓2 + k·b_, k, halton_seq(n, b2))]
    z_ := [VECTOR(lfb↓3 + k·c_, k, halton_seq(n, b3))]
    APPEND(x_, y_, z_)`

pts3d := halton3d(1000, [0, 0, 0], [1, 0, 0], [0, 1, 1], 2, 3, 5)

plot3dlist(list) := VECTOR([v], v, list)

plot3dlist(pts3d)
```



We will estimate the volume below a surface which needs a double integral. Let's assume that we cannot integrate the function. The problem is:

$$\int_0^1\int_0^1 1-\frac{x^2}{2}-\frac{y^2}{2}\,\mathrm{d}y\,\mathrm{d}x.$$

$$g(x,\ y) := 1 - \frac{x^2}{2} - \frac{y^2}{2}$$

```
pts3d := halton3d(1000, [0, 0, 0], [1, 0, 0], [0, 1, 1], 2, 3, 5)
```

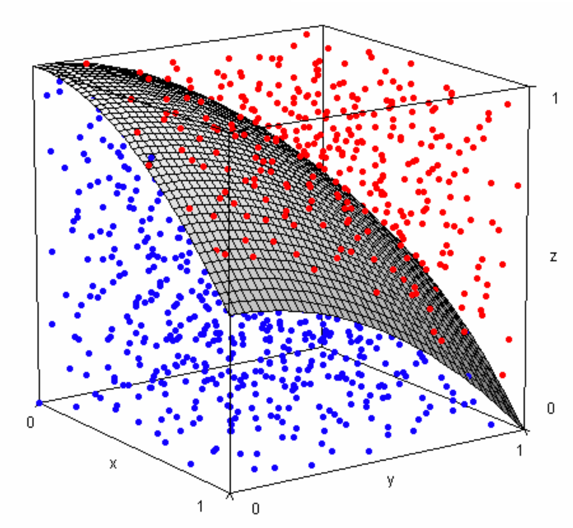
$$\mathrm{DIM(SELECT}(g(v_1,\ v_2) \ge v_3,\ v,\ \mathrm{pts3d})) = 667$$

$$\frac{667}{1000} = 0.667$$

$$\int_0^1 \int_0^1 g(x,\ y)\ dx\ dy = \frac{2}{3}$$

$$\mathrm{plot3dlist(SELECT}(g(v_1,\ v_2) \ge v_3,\ v,\ \mathrm{pts3d}))$$

$$\mathrm{plot3dlist(SELECT}(g(v_1,\ v_2) < v_3,\ v,\ \mathrm{pts3d}))$$



We count the points whose 3rd coordinate is less than the function value.

Then we plot the surface and the points under the surface (blue) and above (red).

The Hexagon from page 32

At the end of the Corput-Halton-section of my contribution I'd like to come back to the hexagon from page 32 and estimate its area with two different Halton sets (different bases):

$$\text{polpts} := \text{halton}(3000,\ [0,\ 0],\ [4,\ 8],\ 2,\ 3) \qquad \text{polpts} := \text{halton}(3000,\ [0,\ 0],\ [4,\ 8],\ 5,\ 11)$$

$$\text{DIM}(\text{SELECT}(p(v_1,\ v_2),\ v,\ \text{polpts})) = 1434 \qquad \text{DIM}(\text{SELECT}(p(v_1,\ v_2),\ v,\ \text{polpts})) = 1433$$

$$\text{SOLVE}\left(\frac{32}{a} = \frac{3000}{1434},\ a\right) = (a = 15.296) \qquad \text{SOLVE}\left(\frac{32}{a} = \frac{3000}{1433},\ a\right) = (a = 15.28533333)$$

$$\left|\frac{15.2643 - 15.296}{15.2643}\right| = 0.002076741154 \qquad \left|\frac{15.2643 - 15.2853}{15.2643}\right| = 0.001375759124$$

Errors are 0.2% and 0.1%. Convincing or not?

Fauré Sequences

When I started writing this contribution and programming the first function I only wanted to refer to the article in the *Scientific America*. Then I was ready presenting the Hammersley set. Curious as I am I did some Internet research – I shouldn't have done it. To quote our friend David Halprin:
"I opened a can of worms." (in DNL#29 at the occasion of his Super Duper Osculants). I found much materials, too many papers – and most of them filled with not so easy mathematics. They wrote about van der Corput, Halton, then much about *Fauré*- and *Sobol'* sequences. I came across a famous Austrian mathematician who was also busy in this field of mathematics, *Edmund Hlawka* (Koksma-Hlawka Inequality). Finally I was filled with ambition to include the Fauré Sequences. They are mentioned in all papers treating QMC-methods.

I must admit that I did not understand the algorithm described in the first paper. I inspected the next one, and the next one, … - surprisingly in most papers I found the same text. It seems that they all use the same source – and sometimes the writer did not even copy correctly. What do you think about this:

triangular matrix with elements where $\binom{i}{j} = \dfrac{j!}{i!(j-i)!}$:

$$\begin{bmatrix} \binom{0}{0} & \binom{0}{1} & \binom{0}{2} & \binom{0}{3}\cdots \\ 0 & \binom{1}{1} & \binom{1}{2} & \binom{1}{3}\cdots \\ 0 & 0 & \binom{2}{2} & \binom{2}{3}\cdots \\ 0 & 0 & 0 & \binom{3}{3}\cdots \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1\ldots \\ 0 & 1 & 2 & 3\ldots \\ 0 & 0 & 1 & 3\ldots \\ 0 & 0 & 0 & 1\ldots \end{bmatrix}$$

(from a master degree thesis!)

Nice binomial coefficients !!??

For creating an *s*-dimensional Fauré sequence one has to start with a Halton sequence based on the smallest prime number $p \geq s$. These are the first dimension coordinates of the *s*-dimensional points. We receive the coordinates of the next dimension by multiplying the *p*-based vectors – which finally give the decimal values – recursively by a matrix containing binomial coefficients mod *p*.

It's better to demonstrate this:

We will generate a 4-dimensional Fauré sequence. So we start with a Halton sequence based on $p = 5$. Take the 100th element: $100 = 400_5$, which is reflected 0.004_5 giving the decimal number 0.032.

$$\text{conv\_rev}(100,\ 5) = [0,\ 0,\ 4]$$

$$(\text{halton\_seq}(100,\ 5))_{100} = \frac{4}{125}$$

$$(\text{halton\_seq}(100,\ 5))_{100} = 0.032$$

We need the vector [0, 0, 4] as column vector which must be multiplied recursively by an upper triangular matrix of binomial coefficients with appropriate dimension:

$$\begin{pmatrix} C_0^0 & C_0^1 & C_0^2 & C_0^3 & \dots \\ 0 & C_1^1 & C_1^2 & C_1^3 & \dots \\ 0 & 0 & C_2^2 & C_2^3 & \dots \\ 0 & 0 & 0 & C_3^3 & \dots \\ 0 & 0 & 0 & 0 & \dots \end{pmatrix} \bmod p. \quad \text{This is in our case: } \begin{pmatrix} \binom{0}{0} & \binom{1}{0} & \binom{2}{0} \\ 0 & \binom{1}{1} & \binom{2}{1} \\ 0 & 0 & \binom{2}{2} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}.$$

So let's perform the repeated multiplication mod *p*. *DERIVE*'s ITERATES-function is an excellent tool for jobs like this:

$$\text{ITERATES}\left(\text{MOD}(\text{trm}(3)\cdot v,\ 5),\ v,\ \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix},\ 3\right) = \left[\begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix}, \begin{bmatrix} 4 \\ 3 \\ 4 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 4 \end{bmatrix}, \begin{bmatrix} 1 \\ 4 \\ 4 \end{bmatrix}\right]$$

$$\left[\frac{4}{125},\ \frac{4}{5} + \frac{3}{25} + \frac{4}{125},\ \frac{1}{5} + \frac{1}{25} + \frac{4}{125},\ \frac{1}{5} + \frac{4}{25} + \frac{4}{125}\right]$$

$$\left[\frac{4}{125},\ \frac{119}{125},\ \frac{34}{125},\ \frac{49}{125}\right]$$

```
[0.032, 0.952, 0.272, 0.392]
```

Point #100 has coordinates (0.32, 0.952, 0.272, 0.392). Later we will have the occasion to check if this calculation is correct.

I start implementing the algorithm in *DERIVE* defining two auxiliary functions:

```
       conv_rev(number, base, c) :=
         Prog
           c := []
           Loop
#1:          If number < base exit
             c := APPEND([MOD(number, base)], c)
             number := FLOOR(number, base)
           REVERSE(APPEND([number], c))

#2:    trm(k) := VECTOR(VECTOR(COMB(i, j), i, 0, k - 1), j, 0, k - 1)
```

`con_rev(number,base)` converts and reflects, giving a vector and `trm(k)` generates the appropriate upper trianglular matrix of binomial coefficients.

```
faure(n, s, p, ir, r, fau) :=
  Prog
    p := IF(PRIME?(s), s, NEXT_PRIME(s))
    fau := []
    ir := 1
    Loop
      If ir > n exit
      b := [conv_rev(ir, p)]'
      r := MOD(ITERATES(trm(DIM(v))·v, v, b, s - 1), p)
      r := [VECTOR(v'·VECTOR(p^(-k), k, 1, DIM(v)), v, r)]'↓1
      fau := APPEND(fau, r)
      ir :+ 1
    fau
```

The first test:

```
faure(11, 3)

⎡ 0.333333    0.333333   0.333333 ⎤
⎢ 0.666666    0.666666   0.666666 ⎥
⎢ 0.111111    0.444444   0.777777 ⎥
⎢ 0.444444    0.777777   0.111111 ⎥
⎢ 0.777777    0.111111   0.444444 ⎥
⎢ 0.222222    0.888888   0.555555 ⎥
⎢ 0.555555    0.222222   0.888888 ⎥
⎢ 0.888888    0.555555   0.222222 ⎥
⎢ 0.0370370   0.592592   0.481481 ⎥
⎢ 0.370370    0.925925   0.814814 ⎥
⎣ 0.703703    0.259259   0.148148 ⎦
```

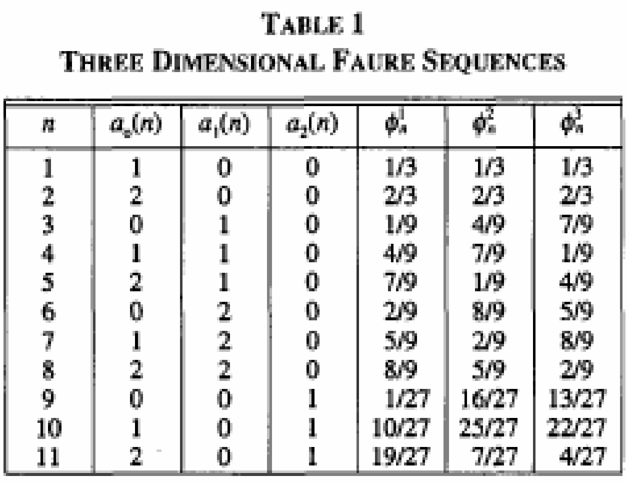


TABLE 1
THREE DIMENSIONAL FAURE SEQUENCES

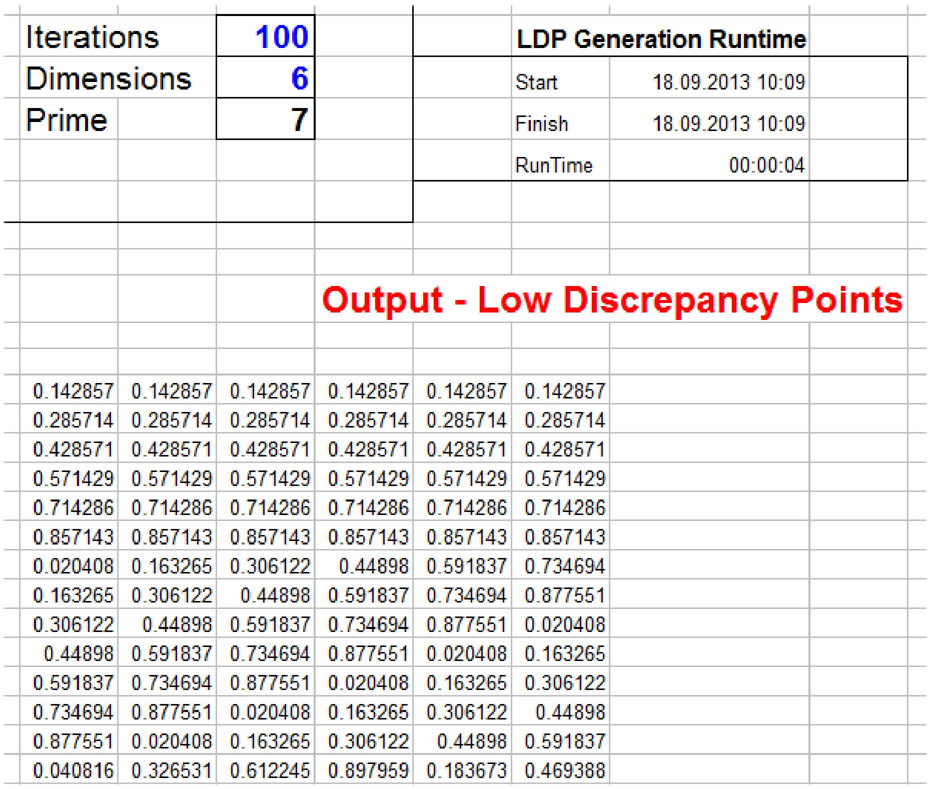
n	$a_0(n)$	$a_1(n)$	$a_2(n)$	ϕ_n^1	ϕ_n^2	ϕ_n^3
1	1	0	0	1/3	1/3	1/3
2	2	0	0	2/3	2/3	2/3
3	0	1	0	1/9	4/9	7/9
4	1	1	0	4/9	7/9	1/9
5	2	1	0	7/9	1/9	4/9
6	0	2	0	2/9	8/9	5/9
7	1	2	0	5/9	2/9	8/9
8	2	2	0	8/9	5/9	2/9
9	0	0	1	1/27	16/27	13/27
10	1	0	1	10/27	25/27	22/27
11	2	0	1	19/27	7/27	4/27

Joy a.o.: Quasi-Monte Carlo Methods in Numerical Finance

Second test:

```
faure(10, 6)

⎡ 0.1428571428    0.1428571428  0.1428571428  0.1428571428  0.1428571428    0.1428571428   ⎤
⎢ 0.2857142857    0.2857142857  0.2857142857  0.2857142857  0.2857142857    0.2857142857   ⎥
⎢ 0.4285714285    0.4285714285  0.4285714285  0.4285714285  0.4285714285    0.4285714285   ⎥
⎢ 0.5714285714    0.5714285714  0.5714285714  0.5714285714  0.5714285714    0.5714285714   ⎥
⎢ 0.7142857142    0.7142857142  0.7142857142  0.7142857142  0.7142857142    0.7142857142   ⎥
⎢ 0.8571428571    0.8571428571  0.8571428571  0.8571428571  0.8571428571    0.8571428571   ⎥
⎢ 0.02040816326   0.1632653061  0.3061224489  0.4489795918  0.5918367346    0.7346938775   ⎥
⎢ 0.1632653061    0.3061224489  0.4489795918  0.5918367346  0.7346938775    0.8775510204   ⎥
⎢ 0.3061224489    0.4489795918  0.5918367346  0.7346938775  0.8775510204    0.02040816326  ⎥
⎣ 0.4489795918    0.5918367346  0.7346938775  0.8775510204  0.02040816326   0.1632653061   ⎦
```

For my second test I use an Excel-worksheet which can be downloaded from [11].
It is an LDP (= Low Discrepancy Points) Generator.



Iterations	100
Dimensions	6
Prime	7

LDP Generation Runtime	
Start	18.09.2013 10:09
Finish	18.09.2013 10:09
RunTime	00:00:04

Output - Low Discrepancy Points

0.142857	0.142857	0.142857	0.142857	0.142857	0.142857
0.285714	0.285714	0.285714	0.285714	0.285714	0.285714
0.428571	0.428571	0.428571	0.428571	0.428571	0.428571
0.571429	0.571429	0.571429	0.571429	0.571429	0.571429
0.714286	0.714286	0.714286	0.714286	0.714286	0.714286
0.857143	0.857143	0.857143	0.857143	0.857143	0.857143
0.020408	0.163265	0.306122	0.44898	0.591837	0.734694
0.163265	0.306122	0.44898	0.591837	0.734694	0.877551
0.306122	0.44898	0.591837	0.734694	0.877551	0.020408
0.44898	0.591837	0.734694	0.877551	0.020408	0.163265
0.591837	0.734694	0.877551	0.020408	0.163265	0.306122
0.734694	0.877551	0.020408	0.163265	0.306122	0.44898
0.877551	0.020408	0.163265	0.306122	0.44898	0.591837
0.040816	0.326531	0.612245	0.897959	0.183673	0.469388

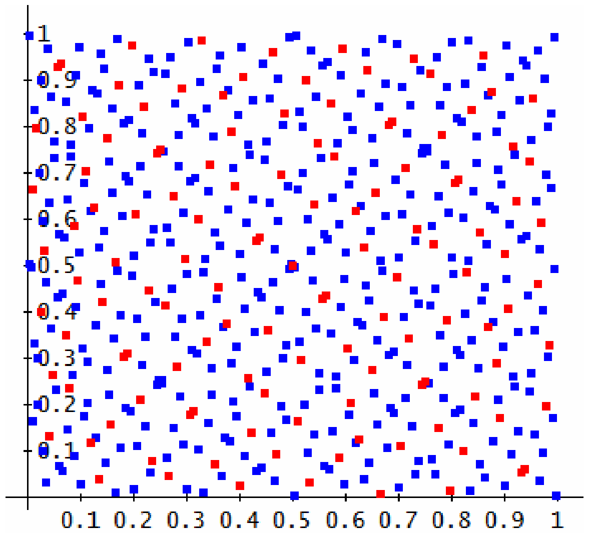
Please compare!

I use LPD for calculating the 100th point in 4D-space from above:

99	0.984	0.224	0.864	0.704
100	**0.032**	**0.952**	**0.272**	**0.392**
101	0.232	0.152	0.472	0.592

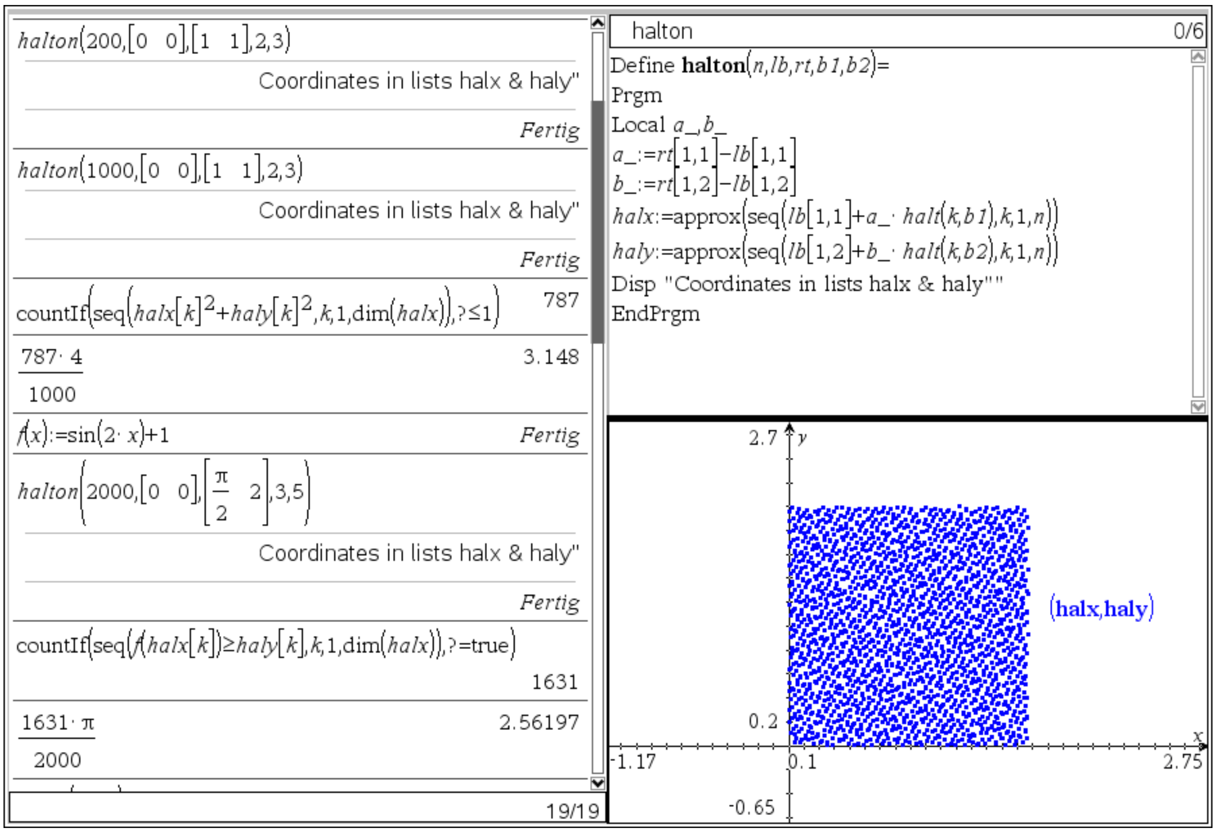
It seems that my program works properly.

I finish this contribution demonstrating the distribution of 500 points of 2D-Fauré points. The first 125 points are the red ones.



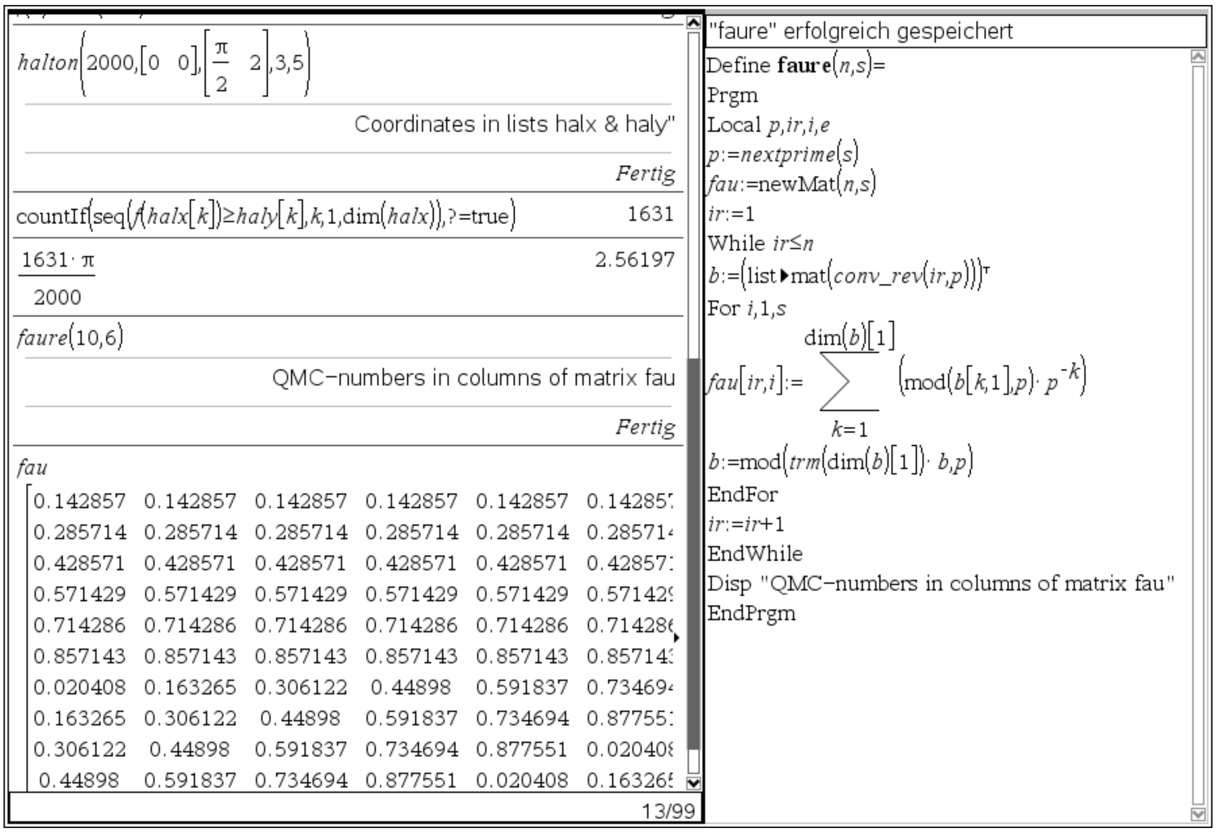
You can find a numerical approximation of a multiple integral on the next page.

QMC with TI-NspireCAS



```
halton(200,[0  0],[1  1],2,3)
                              Coordinates in lists halx & haly"
                                                         Fertig
halton(1000,[0  0],[1  1],2,3)
                              Coordinates in lists halx & haly"
                                                         Fertig
countIf(seq(halx[k]^2+haly[k]^2,k,1,dim(halx)),?≤1)         787
787·4/1000                                                3.148
f(x):=sin(2·x)+1                                         Fertig
halton(2000,[0  0],[π/2  2],3,5)
                              Coordinates in lists halx & haly"
                                                         Fertig
countIf(seq(f(halx[k])≥haly[k],k,1,dim(halx)),?=true)
                                                           1631
1631·π/2000                                             2.56197
                                                          19/19
```

```
halton                                                      0/6
Define halton(n,lb,rt,b1,b2)=
Prgm
Local a_,b_
a_:=rt[1,1]-lb[1,1]
b_:=rt[1,2]-lb[1,2]
halx:=approx(seq(lb[1,1]+a_·halt(k,b1),k,1,n))
haly:=approx(seq(lb[1,2]+b_·halt(k,b2),k,1,n))
Disp "Coordinates in lists halx & haly""
EndPrgm
```



```
halton(2000,[0  0],[π/2  2],3,5)
                              Coordinates in lists halx & haly"
                                                         Fertig
countIf(seq(f(halx[k])≥haly[k],k,1,dim(halx)),?=true)      1631
1631·π/2000                                             2.56197
faure(10,6)
                         QMC−numbers in columns of matrix fau
                                                         Fertig
fau
0.142857  0.142857  0.142857  0.142857  0.142857  0.14285
0.285714  0.285714  0.285714  0.285714  0.285714  0.28571
0.428571  0.428571  0.428571  0.428571  0.428571  0.42857
0.571429  0.571429  0.571429  0.571429  0.571429  0.57142
0.714286  0.714286  0.714286  0.714286  0.714286  0.71428
0.857143  0.857143  0.857143  0.857143  0.857143  0.85714
0.020408  0.163265  0.306122   0.44898  0.591837  0.73469
0.163265  0.306122   0.44898  0.591837  0.734694  0.87755
0.306122   0.44898  0.591837  0.734694  0.877551  0.02040
 0.44898  0.591837  0.734694  0.877551  0.020408  0.16326
                                                          13/99
```

```
"faure" erfolgreich gespeichert
Define faure(n,s)=
Prgm
Local p,ir,i,e
p:=nextprime(s)
fau:=newMat(n,s)
ir:=1
While ir≤n
b:=(list▸mat(conv_rev(ir,p)))ᵀ
For i,1,s
fau[ir,i]:= Σ(k=1, dim(b)[1]) (mod(b[k,1],p)·p^(-k))
b:=mod(trm(dim(b)[1])·b,p)
EndFor
ir:=ir+1
EndWhile
Disp "QMC−numbers in columns of matrix fau"
EndPrgm
```

You can download the respective quasi-random.tns-file for TI-Nspire CAS.

Remaining Challenges

Challenge #1: Add some introductory examples **how** QMC-methods are used in financial mathematics (Black-Scholes, option pricing, …). In many applications the quasi random numbers must be transformed to normal distributed random numbers in order to simulate a Brownian motion,

Challenge #2: *Sobol'* sequences are more complicated. Try to program the respective algorithm[9].

References

There are so many websites (my "can of worms") that I can only give a selection:

[1] http://planning.cs.uiuc.edu/node210.html, (Full book with many algorithms)

[2] http://marcoagd.usuarios.rdc.puc-rio.br/quasi_mc.html, (including Excel-files)

[3] http://www.actuaries.org/AFIR/Colloquia/Cairns/Boyle_Tan.pdf

[4] http://www.math.uwaterloo.ca/~clemieux/papers/iccs.pdf (QMC in Finance)

[5] http://www.smartquant.com/references/MonteCarlo/mc6.pdf (Applications of MC/QMC)

[6] https://quanto.inria.fr/pdf_html/mc_quasi_doc.pdf

[7] http://www.mcqmc2012.unsw.edu.au/slides/MCQMC2012_Dick_Tutorial.pdf

[8] http://people.sc.fsu.edu/~jburkardt/m_src/faure/faure.html (MATLAB-Library)

[9] http://web.maths.unsw.edu.au/~fkuo/sobol/joe-kuo-notes.pdf

[10] http://www.wpi.edu/Pubs/ETD/Available/etd-0113104-140925/unrestricted/krykova.pdf

[11] http://www.casact.org/pubs/forum/99spforum/99spf337.pdf
http://www.casact.org/pubs/forum/99spforum/ldpmaker.xls

German websites

[12] http://gcsc.uni-frankfurt.de/computational-finance/students/theses/Benny%20Xiang%20Li.pdf

[13] http://kainhofer.com/Papers/Kainhofer_Talk_QMC_Wissenswertes_20-06-2005.pdf

[14] http://itp.tugraz.at/MML/MonteCarlo/MCIntro.pdf (Grundlagen der MC-Methode)

[14] http://www.schulmediothek.de/fileadmin/pdf/Quasi-Monte_Carlo_Methode.pdf

[15] http://pauli.uni-muenster.de/~lemm/seminarSS08/horstmann_vortragqmc.pdf

and all the respective Wikipedia-sites.

Approximation of the area of the maple leaf applying a QMC-method:

1024 Hammersley QMC-points give an approximation for the Scientific American maple leaf of 41.41% of the square. The real area is 41.85% of the square

A Tribute to *DERIVE* – A Challenge for TI-NspireCAS

When I was busy with the two ACA-contributions of our Canadian friends I got the idea that it might be useful trying to keep the many Utility files of *DERIVE* alive. They are true gems for the DERIVIANs and why not making them accessible for Nspire Users in form of respective libraries. To begin with I started with LinearAlgebra.mth. There is a library for linear algebra linalgcas.tns prod-uced by PhilippeFortin which is very useful but it does not contain all tools conained in the *DERIVE* file. Examples will be given in the next DNL.

I wrote Michel Beaudin and he immediately answered proposing that we could prepare a common talk at TIME 2014. I would like to invite you all to contribute for our BIG PLAN – very well considering that LinearAlgebra.mth is an easy task compared with e.g. ZetaFunctions.mth.

It would be great to hear from you!

Row Reduction Primitives

scale_element(v,i,s) returns a copy of vector v with the ith element multiplied by expression s.
If v is a matrix then the ith row is multiplied by s.

swap_elements(v,i,j) returns a copy of vector v (or list v) with the ith and jth element interchanged.
If v is a matrix then the respective rows are interchanged.

swap_cols(A,i,j) returns a copy of matrix A (or list v) with the ith and jth column interchanged.
This function is not part of LinearAlgebra.mth.

subtr_elements(A,i,j,s) returns a copy of matrix A in which the ith row is replaced with the ith row minus s times the jth row. Is identical with Nspire's mRowAdd(-s,A,j,i).

force0(A,i,j,p) returns a copy of matrix A in which element Ai,j is forced to zero by substracting an appropriate multiple of pivot row p from row i.

pivot(A,i,j) returns a copy of matrix A in which the elements in the jth column below the ith row are forced to zero

Inverses and Adjoints by Cofactor Expansion

delete(v,i) returns a copy of vector or list v with all elements omitted whose numbers are given in the 2nd parameter. If v is a matrix then the respective rows are omitted.

minor(A,i,j) returns a copy of matrix A in which the ith row and the jth column are omitted.

cofactor(A,i,j) returns the numerator of the row j col i element of the inverse of square matrix A. The corresponding element of the inverse is this numerator divided by the determinant of A.

adjoint(A) returns adjugate of square matrix A. It is the transpose of the matrix of the cofactors of A.

Null Space (Kernel), Eigenvalues and Eigenvectors

The respective *DERIVE*-program contains some auxiliary functions which are not available in Nspire (e.g. POSITION, FIRST, REST a.o.). They must be predefined:

position(e,v) gives the position of first appearance of element e in vector v

first(v) returns the first element of a vector/list or matrix (first row).

rest(v) returns all but the first element of a vector/list or matrix (first row).

rank(A) returns the the rank of matrix A. (from the Nspire library linalgcas.tns)

null_space(A): The null space is the set of vectors x satisfying the matrix equation $A\cdot x=0$.
Ph. Fortin's function kernelvectors gives the answer as a row vector in form of a linear combination:

eigenvalues, **exact_evc** and **approx_evc** provide a better functionality.