Basic mathematics

Contents

1 Hello World (and Universe)!

```
PRINT "Hello $1!"
```

```
$ feenox hello.fee World
Hello World!
$ feenox hello.fee Universe
Hello Universe!
$
```

2 The logistic map

Plot the asymptotic behavior of the logistic map

$$x_{n+1} = r \cdot x \cdot (1 - x)$$

for a range of the parameter r.

```
DEFAULT_ARGUMENT_VALUE 1 2.6   # by default show r in [2.6:4]
DEFAULT_ARGUMENT_VALUE 2 4

steps_per_r = 2^10
steps_asymptotic = 2^8
steps_for_r = 2^10

static_steps = steps_for_r*steps_per_r

# change r every steps_per_r steps
IF mod(step_static,steps_per_r)=1
 r = quasi_random($1,$2)
ENDIF

x_init = 0.5           # start at x = 0.5
x = r*x*(1-x)          # apply the map

IF step_static-steps_per_r*floor(step_static/steps_per_r)>(steps_per_r-steps_asymptotic)
 # write the asymptotic behavior only
 PRINT r x
ENDIF
```

```
$ gnuplot
gnuplot> plot "< feenox logistic.fee" w p pt 50 ps 0.02
gnuplot> quit
$
```

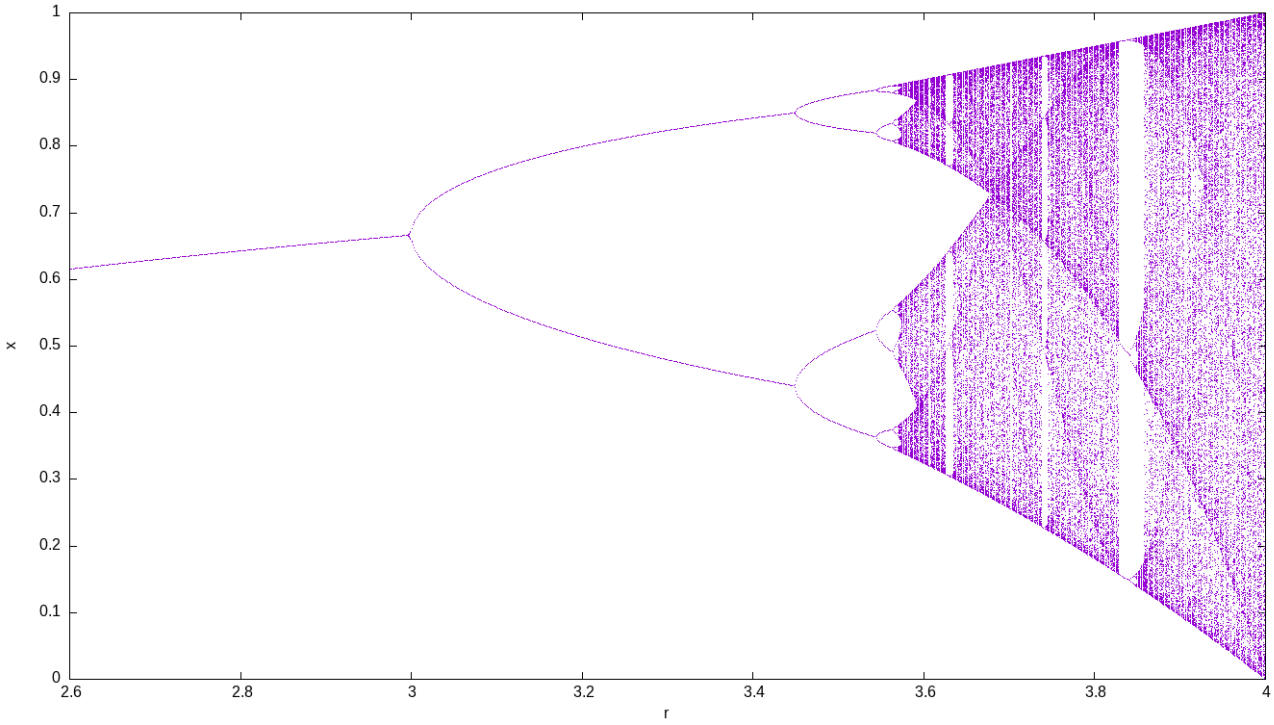


Figure 1: Asymptotic behavior of the logistic map.

3 The Fibonacci sequence

3.1 Using the closed-form formula as a function

When directly executing FeenoX, one gives a single argument to the executable with the path to the main input file. For example, the following input computes the first twenty numbers of the Fibonacci sequence using the closed-form formula

$$f(n) = \frac{\varphi^n - (1-\varphi)^n}{\sqrt{5}}$$

where $\varphi = (1+\sqrt{5})/2$ is the Golden ratio.

```
# the fibonacci sequence as function
phi = (1+sqrt(5))/2
f(n) = (phi^n - (1-phi)^n)/sqrt(5)
PRINT_FUNCTION f MIN 1 MAX 20 STEP 1
```

```
$ feenox fibo_formula.fee | tee one
1   1
2   1
3   2
4   3
```

```
5   5
6   8
7   13
8   21
9   34
10  55
11  89
12  144
13  233
14  377
15  610
16  987
17  1597
18  2584
19  4181
20  6765
$
```

3.2 Using a vector

We could also have computed these twenty numbers by using the direct definition of the sequence into a vector **f** of size 20.

```
# the fibonacci sequence as a vector
VECTOR f SIZE 20

f[i]<1:2> = 1
f[i]<3:vecsize(f)> = f[i-2] + f[i-1]

PRINT_VECTOR i f
```

```
$ feenox fibo_vector.fee > two
$
```

3.3 Solving an iterative problem

Finally, we print the sequence as an iterative problem and check that the three outputs are the same.

```
static_steps = 20
#static_iterations = 1476  # limit of doubles

IF step_static=1|step_static=2
 f_n = 1
 f_nminus1 = 1
 f_nminus2 = 1
ELSE
 f_n = f_nminus1 + f_nminus2
 f_nminus2 = f_nminus1
 f_nminus1 = f_n
ENDIF

PRINT step_static f_n
```

```
$ feenox fibo_iterative.fee > three
$ diff one two
$ diff two three
$
```

4 Computing the derivative of a function as a UNIX filter

This example illustrates how well FeenoX integrates into the UNIX philosophy. Let's say one has a function $f(t)$ as an ASCII file with two columns and one wants to compute the derivative $f'(t)$. Just pipe the function file into this example's input file `derivative.fee` used as a filter.

For example, this small input file `f.fee` writes the function of time provided in the first command-line argument from zero up to the second command-line argument:

```
end_time = $2
PRINT t $1
```

```
$ feenox f.fee "sin(t)" 1
0       0
0.0625  0.0624593
0.125   0.124675
0.1875  0.186403
0.25    0.247404
0.3125  0.307439
0.375   0.366273
0.4375  0.423676
0.5     0.479426
0.5625  0.533303
0.625   0.585097
0.6875  0.634607
0.75    0.681639
0.8125  0.726009
0.875   0.767544
0.9375  0.806081
1       0.841471
$
```

Then we can pipe the output of this command to the derivative filter. Note that

- The `derivative.fee` has the execution flag has on and a shebang line pointing to a global location of the FeenoX binary in `/usr/local/bin` e.g. after doing `sudo make install`.
- The first argument of `derivative.fee` controls the time step. This is only important to control the number of output lines. It does not have anything to do with precision, since the derivative is computed using an adaptive centered numerical differentiation scheme using the GNU Scientific Library.
- Before doing the actual differentiation, the input data is interpolated using a third-order monotonous scheme (also with GSL).
- TL;DR: this is not just "current value minus last value divided time increment."

```
#!/usr/local/bin/feenox
# read the function from stdin
FUNCTION f(t) FILE - INTERPOLATION steffen

# detect the domain range
a = vecmin(vec_f_t)
b = vecmax(vec_f_t)

# time step from arguments (or default 10 steps)
DEFAULT_ARGUMENT_VALUE 1 (b-a)/10
h = $1

# compute the derivative with a wrapper for gsl_deriv_central()
VAR t'
f'(t) = derivative(f(t'),t',t)

# write the result
PRINT_FUNCTION f' MIN a+0.5*h MAX b-0.5*h STEP h
```

```
$ chmod +x derivative.sh
$ feenox f.fee "sin(t)" 1 | ./derivative.fee 0.1 | tee f_prime.dat
0.05    0.998725
0.15    0.989041
0.25    0.968288
0.35    0.939643
0.45    0.900427
0.55    0.852504
0.65    0.796311
0.75    0.731216
0.85    0.66018
0.95    0.574296
$
```

5 On the evaluation of thermal expansion coefficients

When solving thermal-mechanical problems it is customary to use thermal expansion coefficients in order to take into account the mechanical strains induced by changes in the material temperature with respect to a reference temperature where the deformation is identically zero. These coefficients α are defined as the partial derivative of the strain ϵ with respect to temperature T such that differential relationships like

$$d\epsilon = \frac{\partial \epsilon}{\partial T}\, dT = \alpha \cdot dT$$

hold. This derivative α is called the *instantaneous* thermal expansion coefficient. For finite temperature increments, one would like to be able to write

$$\Delta\epsilon = \alpha \cdot \Delta T$$

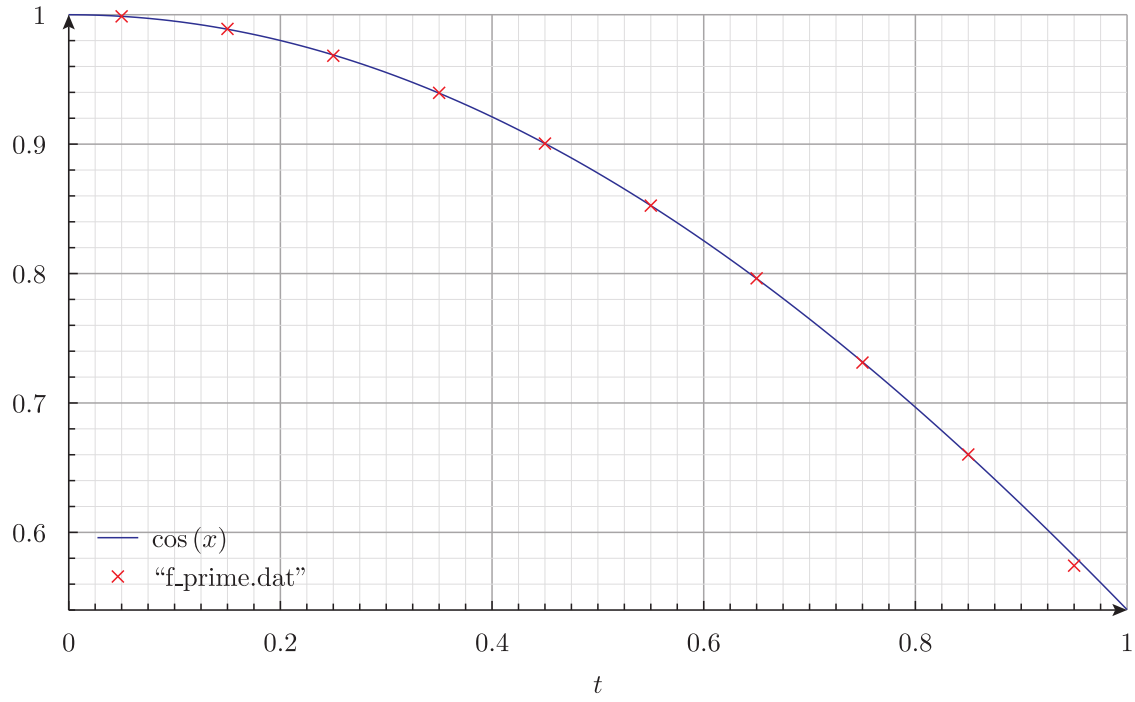


Figure 2: Numerical derivative as a UNIX filter and actual analytical result

But if the strain is not linear with respect to the temperature—which is the most common case—then α depends on T. Therefore, when dealing with finite temperature increments $\Delta T = T - T_0$ where the thermal expansion coefficient $\alpha(T)$ depends on the temperature itself then *mean* values for the thermal expansion ought to be used:

$$\Delta\epsilon = \int_{\epsilon_0}^{\epsilon} d\epsilon' = \int_{T_0}^{T} \frac{\partial \epsilon}{\partial T'}\, dT' = \int_{T_0}^{T} \alpha(T')\, dT'$$

We can multiply and divide by ΔT to obtain

$$\int_{T_0}^{T} \alpha(T')\, dT' \cdot \frac{\Delta T}{\Delta T} = \bar{\alpha}(T) \cdot \Delta T$$

where the mean expansion coefficient for the temperature range $[T_0, T]$ is

$$\bar{\alpha}(T) = \frac{\displaystyle\int_{T_0}^{T} \alpha(T')\, dT'}{T - T_0}$$

This is of course the classical calculus result of the mean value of a continuous one-dimensional function in a certain range.

Let $\epsilon(T)$ be the linear thermal expansion of a given material in a certain direction when heating a piece of such material from an initial temperature T_0 up to T so that $\epsilon(T_0) = 0$.

From our previous analysis, we can see that in fig. 3:

$$A(T) = \alpha(T) = \frac{\partial \epsilon}{\partial T}$$

$$B(T) = \bar{\alpha}(T) = \frac{\epsilon(T)}{T - T_0} = \frac{\int_{T_0}^{T} \alpha(T')\, dT'}{T - T_0}$$

$$C(T) = \epsilon(T) = \int_{T_0}^{T} \alpha(T')\, dT'$$

Therefore,

i. $A(T)$ can be computed out of $C(T)$
ii. $B(T)$ can be computed either out of $A(T)$ or $C(T)$
iii. $C(T)$ can be computed out of $A(T)$

```
# just in case we wanted to interpolate with another method (linear, splines, etc.)
DEFAULT_ARGUMENT_VALUE 1 steffen

# read columns from data file and interpolate
# A is the instantaenous coefficient of thermal expansion x 10^-6 (mm/mm/°C)
FUNCTION A(T) FILE asme-expansion-table.dat COLUMNS 1 2 INTERPOLATION $1
# B is the mean coefficient of thermal expansion x 10^-6 (mm/mm/°C) in going
# from 20°C to indicated temperature
FUNCTION B(T) FILE asme-expansion-table.dat COLUMNS 1 3 INTERPOLATION $1
# C is the linear thermal expansion (mm/m) in going from 20°C
# to indicated temperature
FUNCTION C(T) FILE asme-expansion-table.dat COLUMNS 1 4 INTERPOLATION $1

VAR T'                  # dummy variable for integration
T0 = 20                 # reference temperature
T_min = vecmin(vec_A_T) # smallest argument of function A(T)
T_max = vecmax(vec_A_T) # largest argument of function A(T)

# compute one column from another one
A_fromC(T) := 1e3*derivative(C(T'), T', T)

B_fromA(T) := integral(A(T'), T', T0, T)/(T-T0)
B_fromC(T) := 1e3*C(T)/(T-T0)   # C is in mm/m, hence the 1e3

C_fromA(T) := 1e-3*integral(A(T'), T', T0, T)

# write interpolated results
PRINT_FUNCTION A A_fromC   B B_fromA B_fromC   C C_fromA MIN T_min+1 MAX T_max-1 STEP 1
```

```
$ cat asme-expansion-table.dat
# temp  A       B       C
20      21.7    21.7    0
50      23.3    22.6    0.7
75      23.9    23.1    1.3
```

TABLE TE-2
THERMAL EXPANSION FOR ALUMINUM ALLOYS

Temperature, °C	Coefficients for Aluminum Alloys		
	A	B	C
20	21.7	21.7	0
50	23.3	22.6	0.7
75	23.9	23.1	1.3
100	24.3	23.4	1.9
125	24.7	23.7	2.5
150	25.2	23.9	3.1
175	25.7	24.2	3.7
200	26.4	24.4	4.4
225	27.0	24.7	5.1
250	27.5	25.0	5.7
275	27.7	25.2	6.4
300	27.6	25.5	7.1
325	27.1	25.6	7.8

GENERAL NOTES:

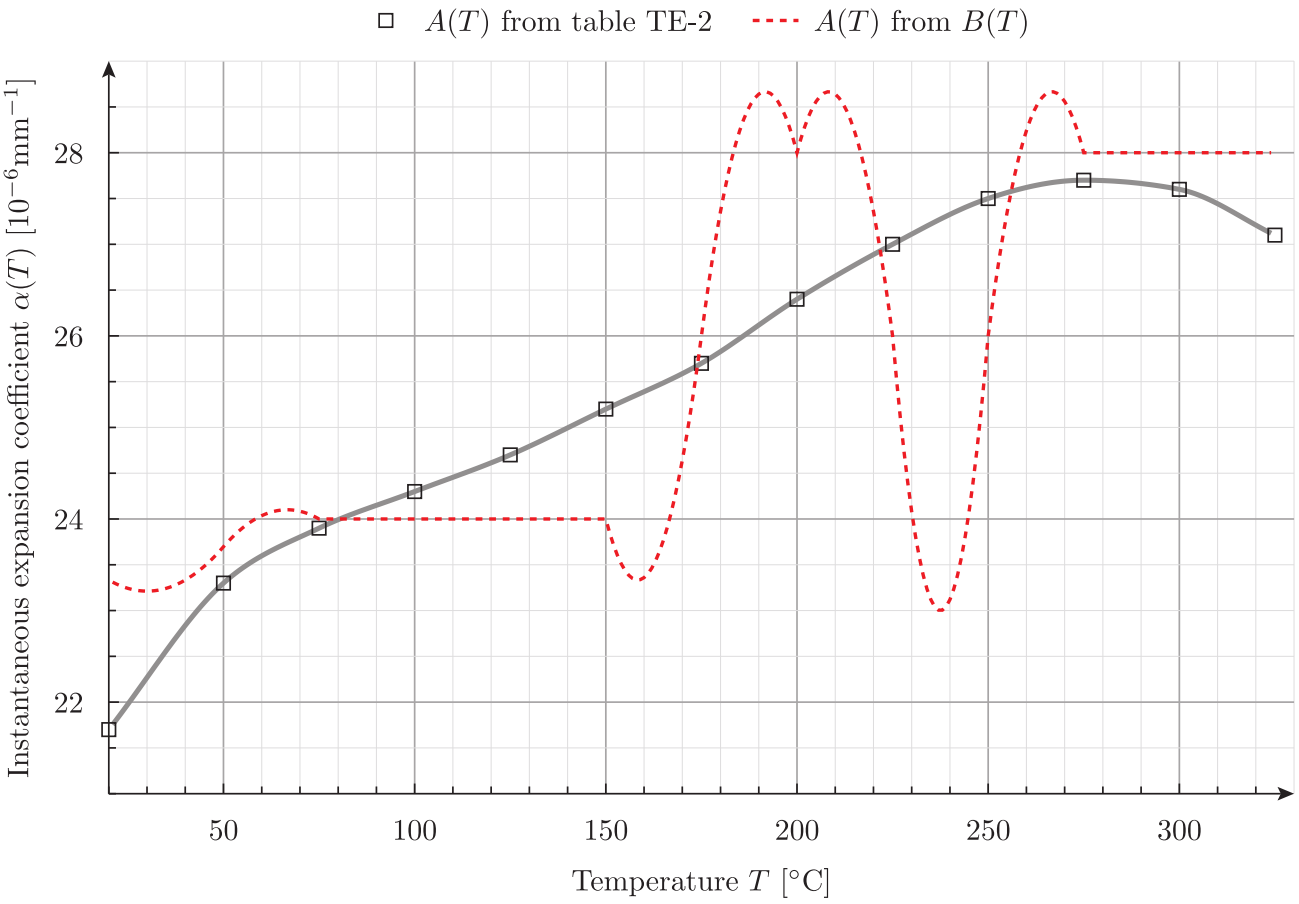
(a) Aluminum alloys represented by these thermal expansion coefficients include:

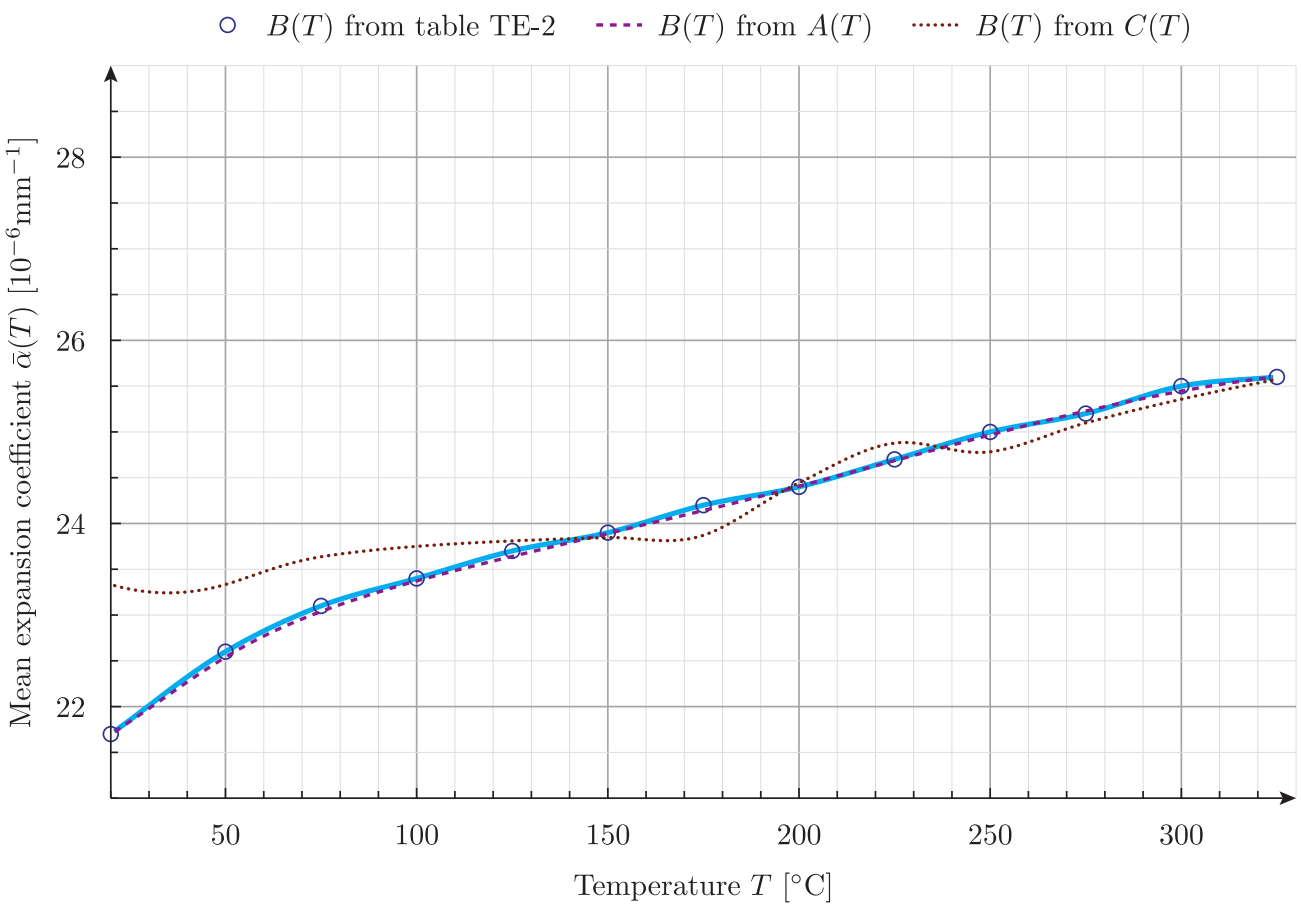
A03560	A93003	A95254
A24430	A93004	A95454
A91060	A95052	A95456
A91100	A95083	A95652
A92014	A95086	A96061
A92024	A95154	A96063

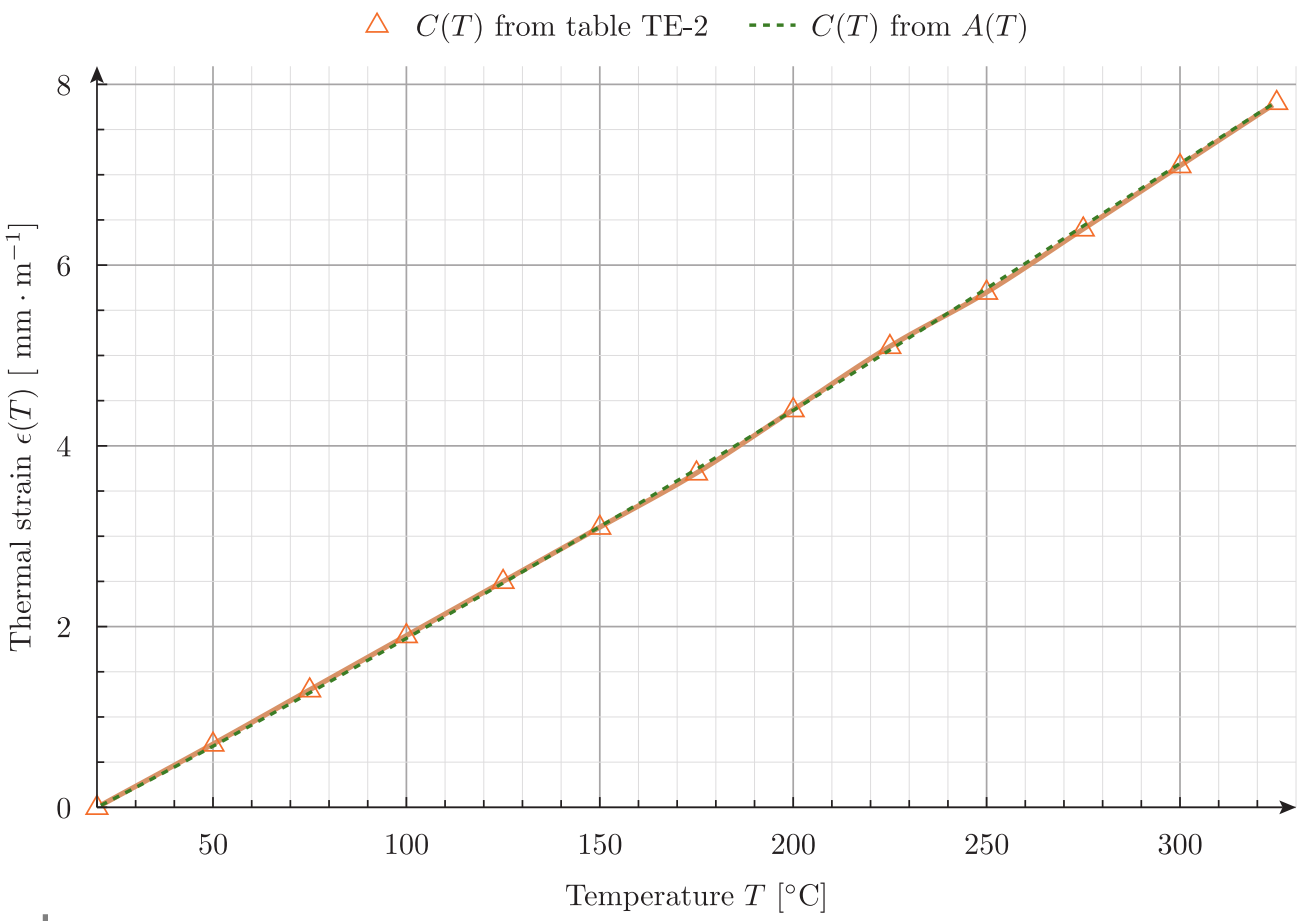
(b) Coefficient A is the instantaneous coefficient of thermal expansion $\times 10^{-6}$ (mm/mm/°C). Coefficient B is the mean coefficient of thermal expansion $\times 10^{-6}$ (mm/mm/°C) in going from 20°C to indicated temperature. Coefficient C is the linear thermal expansion (mm/m) in going from 20°C to indicated temperature.

Figure 3: Table TE2 of thermal expansion properties for Aluminum alloys from ASME II Part D (figure from this report.

```
100     24.3    23.4     1.9
125     24.7    23.7     2.5
150     25.2    23.9     3.1
175     25.7    24.2     3.7
200     26.4    24.4     4.4
225     27.0    24.7     5.1
250     27.5    25.0     5.7
275     27.7    25.2     6.4
300     27.6    25.5     7.1
325     27.1    25.6     7.8
$ feenox asme-expansion.fee > asme-expansion-interpolation.dat
$ pyxplot asme-expansion.ppl
$
```




$B(T)$ from table TE-2
$B(T)$ from $A(T)$
$B(T)$ from $C(T)$
Mean expansion coefficient $\bar{\alpha}(T)$ $[10^{-6}\mathrm{mm}^{-1}]$
22
24
26
28
50
100
150
200
250
300
Temperature T [°C]



> The conclusion (see this, this and this reports) is that values rounded to only one decimal value as presented in the ASME code section II subsection D tables are not enough to satisfy the mathematical relationships between the physical magnitudes related to thermal expansion properties of the materials listed. Therefore, care has to be taken as which of the three columns is chosen when using the data for actual thermo-mechanical numerical computations. As an exercise, the reader is encouraged to try different interpolation algorithms to see how the results change. *Spoiler alert*: they are also highly sensible to the interpolation method used to "fill in" the gaps between the table values.