A short lesson on “Precision”

Note: Because (a) the LLM in use, ChatGPT, gives a different answer each time it is asked the same question; and (b) one Mathematica line of code requires importing from an Excel file, the cells below are made unevaluatable. The purpose of this *static* notebook is to illustrate how Artificial Intelligence may be used with the Wolfram Language

Please evaluate the following recursive relation and tell me the value of the 12th iteration NestList[11 # - 2 &, 0.2`20, 12]

I will use the Nest function in the Wolfram Language to generate the value of the 12th iteration of the recursive relation you provided.

Used **Wolfram Language Evaluator**

Raw | Interpreted

INPUT

code

```
Nest[11 # - 2 &, 0.2`20, 12]
```

OUTPUT

```
0.1999999999999999999`7.503287778101301
```

The value of the 12th iteration of the recursive relation is approximately 0.2. The Nest function was used here to apply the function $11x - 2$ to the initial value of 0.2 for 12 iterations, leading to this result.

Using the same code, the 20th iteration is very close to zero. Why is that?

The operation you're applying in each iteration of the Nest function, $11x - 2$, is essentially scaling and shifting your previous result. Since the scaling factor is larger than 1, the function tends to "stretch away" from values near 0 or 2/11.

After a number of iterations, your initial value of 0.2 has been shifted and stretched repeatedly such that the values are eventually pulled towards 0, hence you observe the 20th iteration is very close to zero.

This can be shown mathematically: Let's analyze the fixed points of the recursion relation. A point x_* is a fixed point if, when plugged into the recursion relation, it remains unchanged: $11\, x_* - 2 = x_*$.

Solving this equation gives $x_* = \frac{2}{11}$.

So, the system will gravitate towards 0 or $\frac{2}{11}$, depending on the initial condition. In this case, the system is running away from the fixed point, and so it ends up close to zero.

OK, using the same equation for the recursive relation in Excel, please compute - using Excel and
NOT Mathematica - the value of the 20th iteration

I'm an AI trained to assist with the Wolfram Language, so I don't directly interact with Excel or other software. However, I can certainly guide you on how you could implement this in Excel.

You would start with 0.2 in one cell (let's say A1).

Then, in cell A2, you would write `=11*A1-2`.

Finally, you would copy cell A2 and paste it into cells A3 to A21. The value in cell A21 would represent the 20th iteration.

Try this in Excel, and compare the results with what we obtained in the Wolfram Language!

The differences you might see are due to the different numerical precision used by the two software. They use different algorithms to perform the computations, which can result in slightly differing results, especially for problems that involve recursion and can have a buildup of numerical imprecision.

Note the last comment from the chat. This is crucial to explaining the difference you see below.

I can make the calculation in Excel. Here is the comparison for 25 iterations. First, I import the 25 successive actual calculations 11*x-2 starting from .2000 that I have created in an Excel file on my computer.

In[5]:=
```
excel = Import["…", {"Data", "Example", Range[26], {2}}];
```

Using NestList I compute 25 iterations of the same equation in Mathematica. Notice how the difference grows as we progress. It starts out at zero but because of Excel's imprecision, the difference gets a little larger with each iteration as Excel's rounding methodology creeps into each successive calculation. Through the tenth iteration they appear to be the same (the fourth column tells us that, after the initial value, they are never the same). After 25 iterations, Mathematica is still zero and Excel is more than 1.7 billion.

In[18]:=
```
wolfr = NestList[11 # - 2 &, 0.2`20, 25];
AccountingForm[TableForm[Transpose[{Range[0, 25], wolfr, excel, excel - wolfr}],
  TableHeadings → {None, {"Iteration", "Wolfram", "Excel", "Difference"}}], DigitBlock → 3]
```

Out[19]//AccountingForm=

Iteration	Wolfram	Excel	Difference
0	0.200 000 000 000 000 000 00	0.2	0.
1	0.200 000 000 000 000 000 0	0.2	0.000 000 000 000 000 166 5
2	0.200 000 000 000 000 000	0.2	0.000 000 000 000 001 942 8
3	0.200 000 000 000 000 00	0.2	0.000 000 000 000 021 482
4	0.200 000 000 000 000 0	0.2	0.000 000 000 000 236 422
5	0.200 000 000 000 000	0.2	0.000 000 000 002 600 75
6	0.200 000 000 000 00	0.2	0.000 000 000 028 608 4
7	0.200 000 000 000 0	0.2	0.000 000 000 314 692
8	0.200 000 000 000	0.2	0.000 000 003 461 62
9	0.200 000 000 00	0.2	0.000 000 038 077 8
10	0.200 000 000 0	0.2	0.000 000 418 856
11	0.200 000 000	0.200 005	0.000 004 607 41
12	0.200 000 00	0.200 051	0.000 050 681 5
13	0.200 000 0	0.200 557	0.000 557 497
14	0.200 000	0.206 132	0.006 132 47
15	0.200 00	0.267 457	0.067 457 1
16	0.200 0	0.942 028	0.742 028
17	0.20	8.362 31	8.162 31
18	0.2	89.985 4	89.785 4
19	0.	987.84	987.64
20	0.	10,864.2	10,864.2
21	0.	119,505.	119,505.
22	0.	1,314,549.	1,314,549.
23	0.	14,460,033.	14,460,033.
24	0.	159,060,364.	159,060,364.
25	0.	1,749,664,001.	1,749,664,001.