cbd

thomas & basil

November 2021

1 Assumptions & changes

In order to make the entire LaTeX generation work, we changed the `None` with `0` in the `latexify.py` on line `774`. Otherwise it would complain that there was a compare between an `int` value and a `None`. When generating the latex content the math mode isn't turned on for the array environments. Which leads to a plethora of faulty rendering when copying the output directly in LaTeX. In order to avoid this issue we changed it. Also some `\backslash left(` aren't matched with `\backslash right)`, and cause a few errors in latex output. This we ignored, as it didn't influence the LaTeX generation.

2 Derivator block

INITIAL SYSTEM:

$$\begin{cases} multIc.OUT1(i) &= multIc.IN1\,(i) \cdot multIc.IN2\,(i) \\ mult.OUT1(i) &= mult.IN1\,(i) \cdot mult.IN2\,(i) \\ inv.OUT1(i) &= 1/inv.IN1\,(i) \\ neg1.OUT1(i) &= -neg1.IN1\,(i) \\ neg2.OUT1(i) &= -neg2.IN1\,(i) \\ sum1.OUT1(i) &= sum1.IN1\,(i) + sum1.IN2\,(i) \\ sum2.OUT1(i) &= sum2.IN1\,(i) + sum2.IN2\,(i) \\ delay.OUT1(i) &= delay.IN1\,(i-1) \\ OUT1(i) &= mult.OUT1\,(i) \\ multIc.IN1(i) &= IC.OUT1\,(i) \\ multIc.IN2(i) &= delta\_t.OUT1\,(i) \\ mult.IN1(i) &= inv.OUT1\,(i) \\ mult.IN2(i) &= sum2.OUT1\,(i) \\ inv.IN1(i) &= delta\_t.OUT1\,(i) \\ neg1.IN1(i) &= multIc.OUT1\,(i) \\ neg2.IN1(i) &= delay.OUT1\,(i) \\ sum1.IN1(i) &= neg1.OUT1\,(i) \\ sum1.IN2(i) &= IN1.OUT1\,(i) \\ sum2.IN1(i) &= neg2.OUT1\,(i) \\ sum2.IN2(i) &= IN1.OUT1\,(i) \\ delay.IC(i) &= sum1.OUT1\,(i) \\ delay.IN1(i) &= IN1.OUT1\,(i) \\ delay.OUT1(0) &= delay.IC(0) \end{cases}$$

STEP 1: substituted all connections and constant values, by the outputs of the incoming connections.

$$\begin{cases} multIc.OUT1(i) &= IC.OUT1\,(i) \cdot delta\_t.OUT1\,(i) \\ mult.OUT1(i) &= inv.OUT1\,(i) \cdot sum2.OUT1\,(i) \\ inv.OUT1(i) &= 1/delta\_t.OUT1\,(i) \\ neg1.OUT1(i) &= -multIc.OUT1\,(i) \\ neg2.OUT1(i) &= -delay.OUT1\,(i) \\ sum1.OUT1(i) &= neg1.OUT1\,(i) + IN1.OUT1\,(i) \\ sum2.OUT1(i) &= neg2.OUT1\,(i) + IN1.OUT1\,(i) \\ delay.OUT1(i) &= IN1.OUT1\,(i-1) \\ OUT1(i) &= mult.OUT1\,(i) \\ delay.OUT1(0) &= sum1.OUT1(0) \end{cases}$$

STEP 2: replaced the equation $OUT1(i) = mult.OUT1\,(i)$ by $OUT1(i) = inv.OUT1\,(i) \cdot sum2.OUT1\,(i)$

$$\begin{cases} multIc.OUT1(i) &= IC.OUT1\,(i) \cdot delta\_t.OUT1\,(i) \\ inv.OUT1(i) &= 1/delta\_t.OUT1\,(i) \\ neg1.OUT1(i) &= -multIc.OUT1\,(i) \\ neg2.OUT1(i) &= -delay.OUT1\,(i) \\ sum1.OUT1(i) &= neg1.OUT1\,(i) + IN1.OUT1\,(i) \\ sum2.OUT1(i) &= neg2.OUT1\,(i) + IN1.OUT1\,(i) \\ delay.OUT1(i) &= IN1.OUT1\,(i-1) \\ OUT1(i) &= inv.OUT1\,(i) \cdot sum2.OUT1\,(i) \\ delay.OUT1(0) &= sum1.OUT1(0) \end{cases}$$

STEP 3: replace all instances of $inv.OUT1(i)$ with $1/deltat.OUT1(i)$ and replace $sum2.OUT1(i)$ with $(neg2.OUT1\,(i) + IN1.OUT1\,(i)$. This step goes wrong by forgetting a right brace.

$$\begin{cases} multIc.OUT1(i) &= IC.OUT1\,(i) \cdot delta\_t.OUT1\,(i) \\ neg1.OUT1(i) &= -multIc.OUT1\,(i) \\ neg2.OUT1(i) &= -delay.OUT1\,(i) \\ sum1.OUT1(i) &= neg1.OUT1\,(i) + IN1.OUT1\,(i) \\ delay.OUT1(i) &= IN1.OUT1\,(i-1) \\ OUT1(i) &= 1/delta\_t.OUT1\,(i) \cdot (neg2.OUT1\,(i) + IN1.OUT1\,(i) \\ delay.OUT1(0) &= sum1.OUT1(0) \end{cases}$$

STEP 4: It tries to simplify the delay away, but removes $delay.OUT1(0)$ wrongly. Therefore a cascade of 'wrong' simplifications, that is the simplifications in themselves will be correct but it won't have anything to do with the original equation, will take place. There also happens a subsitution of $neg2.OUT1(i)$, the braces are correctly placed this time.

$$\begin{cases} multIc.OUT1(i) &= IC.OUT1\,(i) \cdot delta\_t.OUT1\,(i) \\ neg1.OUT1(i) &= -multIc.OUT1\,(i) \\ sum1.OUT1(i) &= neg1.OUT1\,(i) + IN1.OUT1\,(i) \\ OUT1(i) &= 1/delta\_t.OUT1\,(i) \cdot ((-IN1.OUT1\,(i-1)) + IN1.OUT1\,(i) \end{cases}$$

STEP 5: Because $sum1.OUT1(i)$ isn't connected to $OUT1(i)$ it can be deleted from the equations.

$$\begin{cases} multIc.OUT1(i) &= IC.OUT1\,(i) \cdot delta\_t.OUT1\,(i) \\ neg1.OUT1(i) &= -multIc.OUT1\,(i) \\ OUT1(i) &= 1/delta\_t.OUT1\,(i) \cdot ((-IN1.OUT1\,(i-1)) + IN1.OUT1\,(i) \end{cases}$$

STEP 6: Because $neg1.OUT1(i)$ isn't connected to $OUT1(i)$ it can be deleted from the equations.

$$\begin{cases} multIc.OUT1(i) &= IC.OUT1\,(i) \cdot delta\_t.OUT1\,(i) \\ OUT1(i) &= 1/delta\_t.OUT1\,(i) \cdot ((-IN1.OUT1\,(i-1)) + IN1.OUT1\,(i) \end{cases}$$

STEP 7: Because $multIC.OUT1(i)$ isn't connected to $OUT1(i)$ it can be deleted from the equations. For some reason there are 2 equations created at the end. Probably because of code that generated the step and printed the equation of the last step. And the last equation is probably the return value of the function itself.

$$\left\{ \begin{array}{lcl} OUT1(i) & = & 1/delta\_t.OUT1\,(i) \cdot ((-IN1.OUT1\,(i-1)) + IN1.OUT1\,(i) \end{array} \right.$$

$$\left\{ \begin{array}{lcl} OUT1(i) & = & 1/delta\_t.OUT1\,(i) \cdot ((-IN1.OUT1\,(i-1)) + IN1.OUT1\,(i) \end{array} \right.$$

3 Integrator block

INITIAL SYSTEM:

$$\begin{cases} y0.OUT1(i) & = & 0 \\ sum.OUT1(i) & = & sum.IN1\,(i) + sum.IN2\,(i) \\ multiply.OUT1(i) & = & multiply.IN1\,(i) \cdot multiply.IN2\,(i) \\ delay.OUT1(i) & = & delay.IN1\,(i-1) \\ delayState.OUT1(i) & = & delayState.IN1\,(i-1) \\ OUT1(i) & = & sum.OUT1\,(i) \\ sum.IN1(i) & = & delayState.OUT1\,(i) \\ sum.IN2(i) & = & multiply.OUT1\,(i) \\ multiply.IN1(i) & = & delay.OUT1\,(i) \\ multiply.IN2(i) & = & delta\_t.OUT1\,(i) \\ delay.IC(i) & = & y0.OUT1\,(i) \\ delay.IN1(i) & = & IN1.OUT1\,(i) \\ delayState.IC(i) & = & IC.OUT1\,(i) \\ delayState.IN1(i) & = & sum.OUT1\,(i) \\ delay.OUT1(0) & = & delay.IC(0) \\ delayState.OUT1(0) & = & delayState.IC(0) \end{cases}$$

STEP 1: substituted all connections and constant values, by the outputs of the incoming connections.

$$\begin{cases} sum.OUT1(i) & = & delayState.OUT1\,(i) + multiply.OUT1\,(i) \\ multiply.OUT1(i) & = & delay.OUT1\,(i) \cdot delta\_t.OUT1\,(i) \\ delay.OUT1(i) & = & IN1.OUT1\,(i-1) \\ delayState.OUT1(i) & = & sum.OUT1\,(i-1) \\ OUT1(i) & = & sum.OUT1\,(i) \\ delay.OUT1(0) & = & 0 \\ delayState.OUT1(0) & = & IC.OUT1(0) \end{cases}$$

STEP 2: Substitute all instances $sum.OUT1(i)$ by $delayState.OUT1(i) + multiply.OUT1(i)$.

$$\begin{cases} multiply.OUT1(i) & = & delay.OUT1\,(i) \cdot delta\_t.OUT1\,(i) \\ delay.OUT1(i) & = & IN1.OUT1\,(i-1) \\ delayState.OUT1(i) & = & OUT1\,(i-1) \\ OUT1(i) & = & delayState.OUT1\,(i) + multiply.OUT1\,(i) \\ delay.OUT1(0) & = & 0 \\ delayState.OUT1(0) & = & IC.OUT1(0) \end{cases}$$

STEP 3: Substitute $multiply.OUT1(i)$ by $(delay.OUT1(i) delta_t.OUT1(i))$. Replace all the instances of $delayState.OUT1(i)$ by $OUT1(i-1)$. It replaces all the instances of $delay.OUT1(i)$ by $IN1.OUT1(i-1)$. It removes $delay.OUT1(0)$,

as well as $delayState.OUT1(0)$ wrongly. For some reason there are 2 equations created at the end. Probably because of code that generated the step and printed the equation of the last step. And the last equation is probably the return value of the function itself.

$$\left\{ \begin{array}{lcl} OUT1(i) & = & OUT1\left(i-1\right) + \left(IN1.OUT1\left(i-1\right) \cdot delta\_t.OUT1\left(i\right)\right) \end{array} \right.$$

$$\left\{ \begin{array}{lcl} OUT1(i) & = & OUT1\left(i-1\right) + \left(IN1.OUT1\left(i-1\right) \cdot delta\_t.OUT1\left(i\right)\right) \end{array} \right.$$

4 Factorial block

INITIAL SYSTEM:

$$\begin{cases} product.OUT1(i) & = & product.IN1\,(i) \cdot product.IN2\,(i) \\ delay.OUT1(i) & = & delay.IN1\,(i-1) \\ one.OUT1(i) & = & 1 \\ plusone.OUT1(i) & = & AddOneBlock\,(plusone.IN1\,(i)) \\ delayFac.OUT1(i) & = & delayFac.IN1\,(i-1) \\ clock - clock.OUT1(i) & = & Clock\,(clock - clock.h\,(i)) \\ clock - delta.OUT1(i) & = & 1.0 \\ n!(i) & = & product.OUT1\,(i) \\ product.IN1(i) & = & delay.OUT1\,(i) \\ product.IN2(i) & = & delayFac.OUT1\,(i) \\ delay.IC(i) & = & one.OUT1\,(i) \\ delay.IN1(i) & = & plusone.OUT1\,(i) \\ plusone.IN1(i) & = & delay.OUT1\,(i) \\ delayFac.IC(i) & = & one.OUT1\,(i) \\ delayFac.IN1(i) & = & product.OUT1\,(i) \\ clock - clock.h(i) & = & clock - delta.OUT1\,(i) \\ delay.OUT1(0) & = & delay.IC(0) \\ delayFac.OUT1(0) & = & delayFac.IC(0) \end{cases}$$

STEP 1: Substituted all connections and constant values, by the outputs of the incoming connections. $clock - delta.OUT1(i)$ removed.

$$\begin{cases} product.OUT1(i) & = & delay.OUT1\,(i) \cdot delayFac.OUT1\,(i) \\ delay.OUT1(i) & = & plusone.OUT1\,(i-1) \\ plusone.OUT1(i) & = & AddOneBlock\,(delay.OUT1\,(i)) \\ delayFac.OUT1(i) & = & product.OUT1\,(i-1) \\ clock - clock.OUT1(i) & = & Clock\,(1.0) \\ n!(i) & = & product.OUT1\,(i) \\ delay.OUT1(0) & = & 1 \\ delayFac.OUT1(0) & = & 1 \end{cases}$$

STEP 2: Replace all instances of $product.OUT1(i)$ by $delay.OUT1(i) \cdot delayFac.OUT1(i)$. Nothing goes wrong here.

$$\begin{cases} delay.OUT1(i) & = & plusone.OUT1\,(i-1) \\ plusone.OUT1(i) & = & AddOneBlock\,(delay.OUT1\,(i)) \\ delayFac.OUT1(i) & = & n!\,(i-1) \\ clock - clock.OUT1(i) & = & Clock\,(1.0) \\ n!(i) & = & delay.OUT1\,(i) \cdot delayFac.OUT1\,(i) \\ delay.OUT1(0) & = & 1 \\ delayFac.OUT1(0) & = & 1 \end{cases}$$

STEP 3: Because $clock - clock.OUT1(i)$ seems nowhere to be used, it will be removed. All instances of $plusone.OUT1(i)$ will be replaced by $AddOneBlock(delay.OUT1(i))$, which was $plusone.OUt1(i)$ initially. The instance of $delayFac.OUT1(i)$ is replaced by $n!(i-1)$, which can't be correct since this leaves ignores $delayFac.OUT1(0)$.

$$\begin{cases} delay.OUT1(i) & = & AddOneBlock\left(delay.OUT1\left(i-1\right)\right) \\ n!(i) & = & AddOneBlock\left(delay.OUT1\left(i-1\right)\right) \cdot n!\left(i-1\right) \\ delay.OUT1(0) & = & 1 \end{cases}$$

Here we replace $delay.OUT1(i-1)$ by $AddOneBlock(delay.OUT1(i2))$, which seems like a redundant step. And we define what $delay.OUT1(1)$ is. Because this is recursive this could go on forever.

$$\begin{cases} delay.OUT1(i) & = & AddOneBlock\left(delay.OUT1\left(i-1\right)\right) \\ n!(i) & = & AddOneBlock\left(AddOneBlock\left(delay.OUT1\left(i-2\right)\right)\right) \cdot n!\left(i-1\right) \\ delay.OUT1(0) & = & 1 \\ delay.OUT1(1) & = & AddOneBlock\left(1\right) \end{cases}$$

5 Forward Euler Rule

INITIAL SYSTEM: We can not find anything wrong with this system of equations.

$$\begin{cases} delay.OUT1(i) & = & delay.IN1\,(i-1) \\ multiply.OUT1(i) & = & multiply.IN1\,(i) \cdot multiply.IN2\,(i) \\ accum.OUT1(i) & = & accum.IN1\,(i) + accum.IN2\,(i) \\ OUT1(i) & = & accum.OUT1\,(i) \\ delay.IN1(i) & = & accum.OUT1\,(i) \\ delay.IC(i) & = & IC.OUT1\,(i) \\ multiply.IN2(i) & = & IN1.OUT1\,(i) \\ multiply.IN1(i) & = & delta\_t.OUT1\,(i) \\ accum.IN2(i) & = & delay.OUT1\,(i) \\ accum.IN1(i) & = & multiply.OUT1\,(i) \\ delay.OUT1(0) & = & delay.IC(0) \end{cases}$$

STEP 1: Substituted all connections and constant values. Nothing seems to go wrong here.

$$\begin{cases} delay.OUT1(i) & = & accum.OUT1\,(i-1) \\ multiply.OUT1(i) & = & delta\_t.OUT1\,(i) \cdot IN1.OUT1\,(i) \\ accum.OUT1(i) & = & multiply.OUT1\,(i) + delay.OUT1\,(i) \\ OUT1(i) & = & accum.OUT1\,(i) \\ delay.OUT1(0) & = & IC.OUT1(0) \end{cases}$$

STEP 2: Nothing goes wrong here.

$$\begin{cases} delay.OUT1(i) & = & OUT1\,(i-1) \\ multiply.OUT1(i) & = & delta\_t.OUT1\,(i) \cdot IN1.OUT1\,(i) \\ OUT1(i) & = & multiply.OUT1\,(i) + delay.OUT1\,(i) \\ delay.OUT1(0) & = & IC.OUT1(0) \end{cases}$$

STEP 3: We remove $delay.OUT1(0)$ for no reason, therefore the equation isn't complete anymore. Furthermore the braces at the end of the $OUT1(i-1)$ are absent.

$$\left\{ \begin{array}{lcl} OUT1(i) & = & delta\_t.OUT1\,(i) \cdot IN1.OUT1\,(i) + OUT1\,(i-1 \end{array} \right.$$

For reasons discussed before there is duplication of the result obtained in step 3.

$$\left\{ \begin{array}{lcl} OUT1(i) & = & delta\_t.OUT1\,(i) \cdot IN1.OUT1\,(i) + OUT1\,(i-1 \end{array} \right.$$

6 Simpson 1/3 Rule

A general remark we need to make before starting to discuss the wrongdoings of this part of the LaTeXgeneration. First off we put the equations in a `equation*` environment, and then put them in `\resizebox{1\hsize}{!}` in order to scale the equations properly over the page. At the end the equations become really small. If you zoom on the equations you can still read them properly.

INITIAL SYSTEM: This system is only for the block controlling whether it should use the simpson or the trapezoid rule and accumulating the result in a smart way. The trapezoid block is aleady explained separately so we won't repeat it here. The set of Simpson equations can be find after this set. In order to show the entire equation on the lines there needed to be inserted. There are also some subscript letters which shouldn't be there. The entire system doesn't even fit on one page, it requires be on 2 pages. The system doesn't seem able to properly print out the modulo operator, but the whole equation seems to be correct.

$$
\begin{cases}
mod.OUT1(i) &= mod.IN1\,(i)\,mod.IN2\,(i) \\
two.OUT1(i) &= 2 \\
equal1.OUT1(i) &= equal1.IN1\,(i) \leftrightarrow equal1.IN2\,(i) \\
zero.OUT1(i) &= 0 \\
choice-integral1.OUT1(i) &= MUX\,(choice-integral1.IN1\,(i)\,, choice-integral1.IN2\,(i)\,, choice-integral1.select\,(i)) \\
trapezoid.OUT1(i) &= TrapezoidBlock\,(trapezoid.IN1\,(i)\,, trapezoid.delta\_t\,(i)\,, trapezoid.IC\,(i)) \\
simpson.OUT1(i) &= SimpsonBlock\,(simpson.IN1\,(i)\,, simpson.delta\_t\,(i)\,, simpson.IC\,(i)) \\
sum.OUT1(i) &= sum.IN1\,(i) + sum.IN2\,(i) \\
delay2.OUT1(i) &= delay2.IN1\,(i-1) \\
sum\_choice.OUT1(i) &= MUX\,(sum \\
\_choice.IN1\,(i)\,, sum & \\
\_choice.IN2\,(i)\,, sum & \\
\_choice.select\,(i) & \\
xqcI94YeRySnhTKOGUdH-91.OUT1(i) &= 0 \\
equal2.OUT1(i) &= equal2.IN1\,(i) \leftrightarrow equal2.IN2\,(i) \\
choice-integral2.OUT1(i) &= MUX\,(choice-integral2.IN1\,(i)\,, choice-integral2.IN2\,(i)\,, choice-integral2.select\,(i)) \\
addone.OUT1(i) &= AddOneBlock\,(addone.IN1\,(i)) \\
delay1.OUT1(i) &= delay1.IN1\,(i-1) \\
OUT1(i) &= sum.OUT1\,(i) \\
mod.IN2(i) &= two.OUT1\,(i) \\
mod.IN1(i) &= delay1.OUT1\,(i) \\
equal1.IN1(i) &= mod.OUT1\,(i) \\
equal1.IN2(i) &= zero.OUT1\,(i) \\
choice-integral1.select(i) &= equal1.OUT1\,(i) \\
choice-integral1.IN1(i) &= trapezoid.OUT1\,(i) \\
choice-integral1.IN2(i) &= simpson.OUT1\,(i) \\
trapezoid.IC(i) &= IC.OUT1\,(i) \\
trapezoid.IN1(i) &= IN1.OUT1\,(i) \\
trapezoid.delta\_t(i) &= delta\_t.OUT1\,(i) \\
simpson.IC(i) &= IC.OUT1\,(i) \\
simpson.IN1(i) &= IN1.OUT1\,(i) \\
simpson.delta\_t(i) &= delta\_t.OUT1\,(i) \\
sum.IN1(i) &= delay2.OUT1\,(i) \\
sum.IN2(i) &= choice-integral2.OUT1\,(i) \\
delay2.IN1(i) &= sum\_choice.OUT1\,(i) \\
delay2.IC(i) &= \\
xqcI94YeRySnhTKOGUdH-91.OUT1\,(i) & \\
sum\_choice.select(i) &= equal1.OUT1\,(i) \\
sum\_choice.IN2(i) &= sum.OUT1\,(i) \\
sum\_choice.IN1(i) &= delay2.OUT1\,(i) \\
equal2.IN1(i) &= zero.OUT1\,(i) \\
equal2.IN2(i) &= delay1.OUT1\,(i) \\
choice-integral2.IN2(i) &= IC.OUT1\,(i) \\
choice-integral2.IN1(i) &= choice-integral1.OUT1\,(i) \\
choice-integral2.select(i) &= equal2.OUT1\,(i) \\
addone.IN1(i) &= delay1.OUT1\,(i) \\
delay1.IC(i) &= zero.OUT1\,(i) \\
delay1.IN1(i) &= addone.OUT1\,(i) \\
delay2.OUT1(0) &= delay2.IC(0) \\
delay1.OUT1(0) &= delay1.IC(0)
\end{cases}
$$

STEP 1: Substituted all connections and constant values. Again the mod operator is rendered wrongly. The inputs of *sum_choice* are not substituted.

$$
\begin{cases}
mod.OUT1(i) & = & delay1.OUT1\,(i)\,2 \\
equal1.OUT1(i) & = & mod.OUT1\,(i) \leftrightarrow 0 \\
choice - integral1.OUT1(i) & = & MUX\,(trapezoid.OUT1\,(i)\,, simpson.OUT1\,(i)\,, equal1.OUT1\,(i)) \\
trapezoid.OUT1(i) & = & TrapezoidBlock\,(IN1.OUT1\,(i)\,, delta\_t.OUT1\,(i)\,, IC.OUT1\,(i)) \\
simpson.OUT1(i) & = & SimpsonBlock\,(IN1.OUT1\,(i)\,, \\
delta\_t.OUT1\,(i)\,, IC.OUT1\,(i) & & \\
sum.OUT1(i) & = & delay2.OUT1\,(i) + choice - integral2.OUT1\,(i) \\
delay2.OUT1(i) & = & sum\_choice.OUT1\,(i-1) \\
sum\_choice.OUT1(i) & = & MUX\,(sum \\
{}_{c}hoice.IN1\,(i)\,, sum & & \\
{}_{c}hoice.IN2\,(i)\,, sum & & \\
{}_{c}hoice.select\,(i) & & \\
equal2.OUT1(i) & = & 0 \leftrightarrow delay1.OUT1\,(i) \\
choice - integral2.OUT1(i) & = & MUX\,(choice - integral1.OUT1\,(i)\,, IC.OUT1\,(i)\,, equal2.OUT1\,(i)) \\
addone.OUT1(i) & = & AddOneBlock\,(delay1.OUT1\,(i)) \\
delay1.OUT1(i) & = & addone.OUT1\,(i-1) \\
OUT1(i) & = & sum.OUT1\,(i) \\
delay2.OUT1(0) & = & 0 \\
delay1.OUT1(0) & = & 0
\end{cases}
$$

STEP 2: The mod operator can't be rendered. $mod.OUT1(i)$ gets substituted by nothing, and $equal1.OUT1(i)$ now equals 0 by default. This could be because you only look at the delay at step 0, and therefore the equation becomes 0. However this is not always true. This also holds for the substitution of $delay1.OUT1(i)$.

$$
\begin{cases}
mod.OUT1(i) & = & delay1.OUT1\,(i)\,2 \\
equal1.OUT1(i) & = & 0 \\
choice - integral1.OUT1(i) & = & MUX\,(trapezoid.OUT1\,(i)\,, simpson.OUT1\,(i)\,, equal1.OUT1\,(i)) \\
trapezoid.OUT1(i) & = & TrapezoidBlock\,(IN1.OUT1\,(i)\,, delta\_t.OUT1\,(i)\,, IC.OUT1\,(i)) \\
simpson.OUT1(i) & = & SimpsonBlock\,(IN1.OUT1\,(i)\,, delta\_t.OUT1\,(i)\,, IC.OUT1\,(i)) \\
delay2.OUT1(i) & = & sum\_choice.OUT1\,(i-1) \\
sum\_choice.OUT1(i) & = & MUX\,(sum \\
{}_{c}hoice.IN1\,(i)\,, sum & & \\
{}_{c}hoice.IN2\,(i)\,, sum & & \\
{}_{c}hoice.select\,(i) & & \\
equal2.OUT1(i) & = & 0 \\
choice - integral2.OUT1(i) & = & MUX\,(choice - integral1.OUT1\,(i)\,, IC.OUT1\,(i)\,, equal2.OUT1\,(i)) \\
addone.OUT1(i) & = & AddOneBlock\,(delay1.OUT1\,(i)) \\
delay1.OUT1(i) & = & addone.OUT1\,(i-1) \\
OUT1(i) & = & delay2.OUT1\,(i) + choice - integral2.OUT1\,(i) \\
delay2.OUT1(0) & = & 0 \\
delay1.OUT1(0) & = & 0
\end{cases}
$$

STEP 3: Removed the 0-th iteration of both $delay1$ and $delay2$, which is incorrect. Because of a wrongly placed $\backslash, there is no ending brace for the first \mathrm{MUX}-block.$

$$
\begin{cases}
equal1.OUT1(i) & = & 0 \\
choice - integral1.OUT1(i) & = & MUX\,(trapezoid.OUT1\,(i)\,, simpson.OUT1\,(i)\,, equal1.OUT1\,(i)) \\
trapezoid.OUT1(i) & = & TrapezoidBlock\,(IN1.OUT1\,(i)\,, delta\_t.OUT1\,(i)\,, IC.OUT1\,(i)) \\
simpson.OUT1(i) & = & SimpsonBlock\,(IN1.OUT1\,(i)\,, delta\_t.OUT1\,(i)\,, IC.OUT1\,(i)) \\
equal2.OUT1(i) & = & 0 \\
OUT1(i) & = & MUX\,(sum \\
{}_{c}hoice.IN1\,(i-1)\,, sum & & \\
{}_{c}hoice.IN2\,(i-1)\,, sum & & \\
{}_{c}hoice.select\,(i-1) + & & \\
MUX\,(choice - integral1.OUT1\,(i)\,, IC.OUT1\,(i)\,, equal2.OUT1\,(i)) & &
\end{cases}
$$

STEP 4: This is a correct step.

$$
\begin{cases}
equal1.OUT1(i) & = & 0 \\
OUT1(i) & = & MUX\,(sum \\
{}_{c}hoice.IN1\,(i-1)\,, sum & & \\
{}_{c}hoice.IN2\,(i-1)\,, sum & & \\
{}_{c}hoice.select\,(i-1) + MUX\,(MUX\,(TrapezoidBlock\,(IN1.OUT1\,(i)\,, delta\_t.OUT1\,(i)\,, IC.OUT1\,(i))\,, SimpsonBlock\,(IN1.OUT1\,(i)\,, delta\_t.OUT1\,(i)\,, IC.OUT1\,(i))\,, equal1.OUT1\,(i))\,, IC.OUT1\,(i)\,, 0) & &
\end{cases}
$$

STEP 5: Here we see the strange behaviour of the duplicate equation again.

$$\begin{cases} OUT1(i) \\ \text{„}hoice.IN1\,(i-1)\,, sum \\ \text{„}hoice.IN2\,(i-1)\,, sum \\ \text{„}hoice.select\,(i-1) + MUX\,(MUX\,(TrapezoidBlock\,(IN1.OUT1\,(i)\,, delta\_t.OUT1\,(i)\,, IC.OUT1\,(i))\,, SimpsonBlock\,(IN1.OUT1\,(i)\,, delta\_t.OUT1\,(i)\,, IC.OUT1\,(i))\,, 0)\,, IC.OUT1\,(i)\,, 0) \end{cases} = MUX\,(sum$$

$$\begin{cases} OUT1(i) \\ \text{„}hoice.IN1\,(i-1)\,, sum \\ \text{„}hoice.IN2\,(i-1)\,, sum \\ \text{„}hoice.select\,(i-1) + MUX\,(MUX\,(TrapezoidBlock\,(IN1.OUT1\,(i)\,, delta\_t.OUT1\,(i)\,, IC.OUT1\,(i))\,, SimpsonBlock\,(IN1.OUT1\,(i)\,, delta\_t.OUT1\,(i)\,, IC.OUT1\,(i))\,, 0)\,, IC.OUT1\,(i)\,, 0) \end{cases} = MUX\,(sum$$

This represents the block that is able to calculate the Simpson rule by itself. It is important to note that this block on itself is not correct, it generates wrong output in the first two ticks. Our system discards the output of this block for those values.

INITIAL SYSTEM: Once again there is a subscript error in the generated

LaTeXcode.

$$
\begin{cases}
delay2 - s.OUT1(i) &= delay2 - s.IN1\,(i-1) \\
delay3 - s.OUT1(i) &= delay3 - s.IN1\,(i-1) \\
delay1 - s.OUT1(i) &= delay1 - s.IN1\,(i-1) \\
sum3 - s.OUT1(i) &= sum3 - s.IN1\,(i) + sum3 - s.IN2\,(i) \\
div\_by\_6 - s.OUT1(i) &= div \\
{}_by & \\
{}_6 - s.IN1\,(i) \cdot div & \\
{}_by & \\
{}_6 - s.IN2\,(i) & \\
middle - s.OUT1(i) &= middle - s.IN1\,(i) \cdot middle - s.IN2\,(i) \\
invert - s.OUT1(i) &= 1/invert - s.IN1\,(i) \\
six - s.OUT1(i) &= 6 \\
xqcI94YeRySnhTKOGUdH - 169.OUT1(i) &= 4 \\
sum2 - s.OUT1(i) &= sum2 - s.IN1\,(i) + sum2 - s.IN2\,(i) \\
sum1 - s.OUT1(i) &= sum1 - s.IN1\,(i) + sum1 - s.IN2\,(i) \\
final\_product - s.OUT1(i) &= final \\
{}_product - s.IN1\,(i) \cdot final & \\
{}_product - s.IN2\,(i) & \\
OUT1(i) &= final\_product - s.OUT1\,(i) \\
delay2 - s.IC(i) &= IC.OUT1\,(i) \\
delay2 - s.IN1(i) &= IN1.OUT1\,(i) \\
delay3 - s.IC(i) &= IC.OUT1\,(i) \\
delay3 - s.IN1(i) &= delay2 - s.OUT1\,(i) \\
delay1 - s.IN1(i) &= delta\_t.OUT1\,(i) \\
delay1 - s.IC(i) &= delta\_t.OUT1\,(i) \\
sum3 - s.IN2(i) &= delta\_t.OUT1\,(i) \\
sum3 - s.IN1(i) &= delay1 - s.OUT1\,(i) \\
div\_by\_6 - s.IN1(i) &= sum3 - s.OUT1\,(i) \\
div\_by\_6 - s.IN2(i) &= invert - s.OUT1\,(i) \\
middle - s.IN2(i) &= delay2 - s.OUT1\,(i) \\
middle - s.IN1(i) &= xqcI94YeRySnhTKOGUdH - 169.OUT1\,(i) \\
invert - s.IN1(i) &= six - s.OUT1\,(i) \\
sum2 - s.IN1(i) &= IN1.OUT1\,(i) \\
sum2 - s.IN2(i) &= delay3 - s.OUT1\,(i) \\
sum1 - s.IN2(i) &= middle - s.OUT1\,(i) \\
sum1 - s.IN1(i) &= sum2 - s.OUT1\,(i) \\
final\_product - s.IN1(i) &= sum1 - s.OUT1\,(i) \\
final\_product - s.IN2(i) &= div\_by\_6 - s.OUT1\,(i) \\
delay2 - s.OUT1(0) &= delay2 - s.IC(0) \\
delay3 - s.OUT1(0) &= delay3 - s.IC(0) \\
delay1 - s.OUT1(0) &= delay1 - s.IC(0)
\end{cases}
$$

STEP 1: Substituted all connections and constant values. Did not substitute the input of the $div\_by\_6 - s$ block and the $final\_product - s$-block.

$$
\begin{cases}
delay2 - s.OUT1(i) & = & IN1.OUT1\,(i-1) \\
delay3 - s.OUT1(i) & = & delay2 - s.OUT1\,(i-1) \\
delay1 - s.OUT1(i) & = & delta\_t.OUT1\,(i-1) \\
sum3 - s.OUT1(i) & = & delay1 - s.OUT1\,(i) + delta\_t.OUT1\,(i) \\
div\_by\_6 - s.OUT1(i) & = & div \\
{}_by & & \\
{}_6 - s.IN1\,(i) \cdot div & & \\
{}_by & & \\
{}_6 - s.IN2\,(i) & & \\
middle - s.OUT1(i) & = & 4 \cdot delay2 - s.OUT1\,(i) \\
invert - s.OUT1(i) & = & 1/6 \\
sum2 - s.OUT1(i) & = & IN1.OUT1\,(i) + delay3 - s.OUT1\,(i) \\
sum1 - s.OUT1(i) & = & sum2 - s.OUT1\,(i) + middle - s.OUT1\,(i) \\
final\_product - s.OUT1(i) & = & final \\
{}_product - s.IN1\,(i) \cdot final & & \\
{}_product - s.IN2\,(i) & & \\
OUT1(i) & = & final\_product - s.OUT1\,(i) \\
delay2 - s.OUT1(0) & = & IC.OUT1(0) \\
delay3 - s.OUT1(0) & = & IC.OUT1(0) \\
delay1 - s.OUT1(0) & = & delta\_t.OUT1(0)
\end{cases}
$$

STEP 2: The equation $delay1 - s.OUT1(0)$ is wrongly removed.

$$
\begin{cases}
delay2 - s.OUT1(i) & = & IN1.OUT1\,(i-1) \\
delay3 - s.OUT1(i) & = & delay2 - s.OUT1\,(i-1) \\
delay1 - s.OUT1(i) & = & delta\_t.OUT1\,(i) \\
sum3 - s.OUT1(i) & = & delay1 - s.OUT1\,(i) + delta\_t.OUT1\,(i) \\
div\_by\_6 - s.OUT1(i) & = & div \\
{}_by & & \\
{}_6 - s.IN1\,(i) \cdot div & & \\
{}_by & & \\
{}_6 - s.IN2\,(i) & & \\
middle - s.OUT1(i) & = & 4 \cdot delay2 - s.OUT1\,(i) \\
invert - s.OUT1(i) & = & 0.16666666666666666 \\
sum2 - s.OUT1(i) & = & IN1.OUT1\,(i) + delay3 - s.OUT1\,(i) \\
sum1 - s.OUT1(i) & = & sum2 - s.OUT1\,(i) + middle - s.OUT1\,(i) \\
OUT1(i) & = & final \\
{}_product - s.IN1\,(i) \cdot final & & \\
{}_product - s.IN2\,(i) & & \\
delay2 - s.OUT1(0) & = & IC.OUT1(0) \\
delay3 - s.OUT1(0) & = & IC.OUT1(0)
\end{cases}
$$

STEP 3: Everything is correct.

$$\begin{cases} OUT1(i) & = & final \\ {}_{p}roduct - s.IN1\,(i) \cdot final \\ {}_{p}roduct - s.IN2\,(i) \end{cases}$$

$$\begin{cases} OUT1(i) & = & final \\ {}_{p}roduct - s.IN1\,(i) \cdot final \\ {}_{p}roduct - s.IN2\,(i) \end{cases}$$

7 Trapezoid Rule

INITIAL SYSTEM: When trying to render this system of equations we remark that the escaping of the character '_' is not done properly.

$$
\begin{cases}
y0.OUT1(i) & = & 0 \\
accumulator.OUT1(i) & = & accumulator.IN1\,(i) + accumulator.IN2\,(i) \\
delay\_state.OUT1(i) & = & delay \\
{}_{s}tate.IN1\,(i-1) & & \\
delay\_input.OUT1(i) & = & delay \\
{}_{i}nput.IN1\,(i-1) & & \\
mid\_adder.OUT1(i) & = & mid \\
{}_{a}dder.IN1\,(i) + mid & & \\
{}_{a}dder.IN2\,(i) & & \\
mult.OUT1(i) & = & mult.IN1\,(i) \cdot mult.IN2\,(i) \\
halver.OUT1(i) & = & 0.5 \\
delta\_halver.OUT1(i) & = & delta \\
{}_{h}alver.IN1\,(i) \cdot delta & & \\
{}_{h}alver.IN2\,(i) & & \\
OUT1(i) & = & accumulator.OUT1\,(i) \\
accumulator.IN1(i) & = & delay\_state.OUT1\,(i) \\
accumulator.IN2(i) & = & mult.OUT1\,(i) \\
delay\_state.IN1(i) & = & accumulator.OUT1\,(i) \\
delay\_state.IC(i) & = & IC.OUT1\,(i) \\
delay\_input.IN1(i) & = & IN1.OUT1\,(i) \\
delay\_input.IC(i) & = & y0.OUT1\,(i) \\
mid\_adder.IN2(i) & = & IN1.OUT1\,(i) \\
mid\_adder.IN1(i) & = & delay\_input.OUT1\,(i) \\
mult.IN1(i) & = & mid\_adder.OUT1\,(i) \\
mult.IN2(i) & = & delta\_halver.OUT1\,(i) \\
delta\_halver.IN1(i) & = & delta\_t.OUT1\,(i) \\
delta\_halver.IN2(i) & = & halver.OUT1\,(i) \\
delay\_state.OUT1(0) & = & delay \\
{}_{s}tate.IC(0) & & \\
delay\_input.OUT1(0) & = & delay \\
{}_{i}nput.IC(0) & &
\end{cases}
$$

STEP 1: substituted all connections and constant values. The constant $halver.OUT1(i)$ gets removed or inproperly substituted. All the equations that have the incorrect escaped character don't get replaced by OUT values.

$$\begin{cases} accumulator.OUT1(i) & = & delay\_state.OUT1\,(i) + mult.OUT1\,(i) \\ delay\_state.OUT1(i) & = & delay \\ {}_{s}tate.IN1\,(i-1) & & \\ delay\_input.OUT1(i) & = & delay \\ {}_{i}nput.IN1\,(i-1) & & \\ mid\_adder.OUT1(i) & = & mid \\ {}_{a}dder.IN1\,(i) + mid & & \\ {}_{a}dder.IN2\,(i) & & \\ mult.OUT1(i) & = & mid\_adder.OUT1\,(i) \cdot delta\_halver.OUT1\,(i) \\ delta\_halver.OUT1(i) & = & delta \\ {}_{h}alver.IN1\,(i) \cdot delta & & \\ {}_{h}alver.IN2\,(i) & & \\ OUT1(i) & = & accumulator.OUT1\,(i) \\ delay\_state.OUT1(0) & = & delay \\ {}_{s}tate.IC(0) & & \\ delay\_input.OUT1(0) & = & delay \\ {}_{i}nput.IC(0) & & \end{cases}$$

STEP 2: Nothing goes wrong here.

$$\begin{cases} delay\_state.OUT1(i) & = & delay \\ {}_{s}tate.IN1\,(i-1) & & \\ delay\_input.OUT1(i) & = & delay \\ {}_{i}nput.IN1\,(i-1) & & \\ mid\_adder.OUT1(i) & = & mid \\ {}_{a}dder.IN1\,(i) + mid & & \\ {}_{a}dder.IN2\,(i) & & \\ mult.OUT1(i) & = & mid\_adder.OUT1\,(i) \cdot delta\_halver.OUT1\,(i) \\ delta\_halver.OUT1(i) & = & delta \\ {}_{h}alver.IN1\,(i) \cdot delta & & \\ {}_{h}alver.IN2\,(i) & & \\ OUT1(i) & = & delay\_state.OUT1\,(i) + mult.OUT1\,(i) \\ delay\_state.OUT1(0) & = & delay \\ {}_{s}tate.IC(0) & & \\ delay\_input.OUT1(0) & = & delay \\ {}_{i}nput.IC(0) & & \end{cases}$$

STEP 3: The $delay\_state.OUT1(0)$ is removed wrongly. The substituion which runs into errors is the $OUT1(i-1)$ equation. There the parenthesis don't match up, the ones on the right hand side seem to be missing, because of wrongly placing a double '\'.

$$\begin{cases} mid\_adder.OUT1(i) & = & mid \\ {}_{a}dder.IN1\,(i) + mid & & \\ {}_{a}dder.IN2\,(i) & & \\ OUT1(i) & = & delay \\ {}_{s}tate.IN1\,(i-1) + (mid\_adder.OUT1\,(i) \cdot (delta & & \\ {}_{h}alver.IN1\,(i) \cdot delta & & \\ {}_{h}alver.IN2\,(i) & & \end{cases}$$

STEP 4: Because of the previous equation to be wrong with their parenthesis, this equation becomes even more confusing. This time the substitution of $mid\_adder.OUT1(i)$ is incorrect. The right parenthesis seem to be missing again. The same equation is written twice, as described before in this pdf file.

$$\begin{cases} OUT1(i) & = & delay \\ {}_{s}tate.IN1\,(i-1) + ((mid & & \\ {}_{a}dder.IN1\,(i) + mid & & \\ {}_{a}dder.IN2\,(i) \cdot (delta & & \\ {}_{h}alver.IN1\,(i) \cdot delta & & \\ {}_{h}alver.IN2\,(i) & & \end{cases}$$

$$\begin{cases} OUT1(i) & = & delay \\ {}_{s}tate.IN1\,(i-1) + ((mid & & \\ {}_{a}dder.IN1\,(i) + mid & & \\ {}_{a}dder.IN2\,(i) \cdot (delta & & \\ {}_{h}alver.IN1\,(i) \cdot delta & & \\ {}_{h}alver.IN2\,(i) & & \end{cases}$$