

The luamathalign package*

Marcel Krüger
tex@2krueger.de

May 4, 2022

1 The problem

In most cases, `amsmath` makes it simple to align multiple equations in a `align` environment. But sometimes, special requirements come up.

Maybe one of your alignment points is in an exponent, or in a radical? The first attempts for such alignments often fail. For example, assume that you want to align the following radicals like this (at the x^3 term):

$$\begin{aligned} & \sqrt{1 - 3x + 3x^2 + (x - 1)^3} \\ = & \sqrt{1 - 3x + 3x^2 + x^3 - 3x^2 + 3x - 1} \\ & = \sqrt{x^3} \end{aligned}$$

“Just adding `&` at the alignment points” doesn’t work:

```
\begin{align*}
  \sqrt{1-3x+3x^2+(\&x-1)^3}\\
  =\sqrt{1-3x+3x^2+\&x^3-3x^2+3x-1}\\
  =\sqrt{\&x^3}
\end{align*}
```

fails with

```
! Missing } inserted.
<inserted text>
}
1.73 \end{align*}
```

Another problem are nested alignments. Take this sample from [anonymous on T_EX – L^AT_EX StackExchange](#): We want alignment like

$$\begin{array}{ll} aaaa = 1 & \text{for } X \\ bbbb = 1 & \text{for } Y \\ \left. \begin{array}{l} c = 1 \\ d = 12 \end{array} \right\} & \text{for } Z \end{array}$$

but in

*This document corresponds to `luamathalign` v0.3, dated 2022-05-04.

```

\begin{align*}
aaaa &= 1 &&\text{for } \$X\$ \ \backslash\backslash
bbbb &= 1 &&\text{for } \$Y\$ \ \backslash\backslash
\left. \begin{aligned}
c &= 1 \ \backslash\backslash
d &= 12
\end{aligned} \right\} &&&\text{for } \$Z\$
\end{align*}

```

there is not obvious way to align the equal signs in the nested `aligned` with the outer signs.

2 The solution

luamathalign provides solutions for both problems under Lua^AT_EX:

`\AlignHere`

The most important new macro is `\AlignHere`: It generates an alignment point like `&`, but it can be used almost everywhere.

So problems like our first example can be implemented by just using `\AlignHere` instead of `&`:

```

\begin{align*}
\sqrt{1-3x+3x^2+(\AlignHere x-1)^3}\ \backslash\backslash
=\sqrt{1-3x+3x^2+(\AlignHere x)^3-3x^2+3x-1}\ \backslash\backslash
=\sqrt{\AlignHere x^3}
\end{align*}

```

$$\begin{aligned}
&\sqrt{1 - 3x + 3x^2 + (x - 1)^3} \\
&= \sqrt{1 - 3x + 3x^2 + x^3 - 3x^2 + 3x - 1} \\
&= \sqrt{x^3}
\end{aligned}$$

Sadly, this doesn't really help with the nested alignment problem: Even if we use `\AlignHere` in the `aligned` environment, the alignment points would be inserted in the inner and not in the outer alignment. For such cases, there is a variant which allows to specify at which level the alignment should happen:

`\SetAlignmentPoint`
`\ExecuteAlignment`

The primary command for this is `\SetAlignmentPoint⟨number⟩`. When called with a negative number it specifies the nesting level. For example when `⟨number⟩` is -1 it is the same as `\AlignHere`, while for -2 it is aligning one level higher and so on.

For example, our nested alignment above wanted to align the inner `aligned` and the outer `align*` at the same point, so `\SetAlignmentPoint-2` is used directly next to an inner alignment point (here `&`, `\AlignHere` would work too). Then the `\ExecuteAlignment` has to appear in the context of the outer `align*`, so it can be written e.g. directly before the next `&` of the outer `align*`:

```
\begin{align*}
aaaa &= 1 &&\text{for } \$X\$ \\\
bbbb &= 1 &&\text{for } \$Y\$ \\\
\left. \begin{aligned}
c &\SetAlignmentPoint-2 &= 1 \\\
d &= 12
\end{aligned} \right\} &&\text{for } \$Z\$
\end{align*}
```

$$\begin{array}{ll}
aaaa = 1 & \text{for } X \\
bbbb = 1 & \text{for } Y \\
\left. \begin{array}{l} c = 1 \\ d = 12 \end{array} \right\} & \text{for } Z
\end{array}$$

If you do not want to keep track of the right nesting level you can explicitly mark a level and refer to it. To do so, use a non-negative `⟨number⟩`. When `\SetAlignmentPoint` is used with a non-negative `⟨number⟩` then `\ExecuteAlignment⟨number⟩` must be executed afterwards with the same `⟨number⟩` at a point where adding a `&` would add a valid alignment point at the right level.

Our example above could therefore also be written as

```
\begin{align*}
aaaa &= 1 &&\text{for } \$X\$ \\\
bbbb &= 1 &&\text{for } \$Y\$ \\\
\left. \begin{aligned}
c &\SetAlignmentPoint0 &= 1 \\\
d &= 12
\end{aligned} \right\} &&\text{for } \$Z\$
\end{align*}
```

$$\begin{array}{ll}
aaaa = 1 & \text{for } X \\
bbbb = 1 & \text{for } Y \\
\left. \begin{array}{l} c = 1 \\ d = 12 \end{array} \right\} & \text{for } Z
\end{array}$$

This variant is also useful when working with custom alignment environment not prepared to work with `luamathalign`. By default `\SetAlignmentPoint⟨number⟩` with negative numbers (and therefore also `\AlignHere`) only work with `amsmath`'s `{align}`, `{aligned}` and their variants. If you have another environment which also follows similar alignment rules then you can either restrict yourself to non-negative `⟨number⟩`s in combination with `\ExecuteAlignment` or patch these environments similar to what `luamathalign` does for `amsmath`.

3 The implementation

3.1 Lua

```
1 local properties = node.get_properties_table()
2 local luacmd = require'luamathalign-luacmd'
3 local hlist = node.id'hlist'
4 local vlist = node.id'vlist'
5 local whatsit = node.id'whatsit'
6 local glue = node.id'glue'
7 local user_defined = node.subtype'user_defined'
8 local whatsit_id = luatexbase.new_whatsit'mathalign'
9 local node_cmd = token.command_id'node'
10 local ampersand = token.new(38, 4)
11
12 local mmode do
13   for k,v in next, tex.getmodevalues() do
14     if v == 'math' then mmode = k end
15   end
16   assert(mmode)
17 end
18
19 -- We might want to add y later
20 local function is_marked(mark, list)
21   for n in node.traverse(list) do
22     local id = n.id
23     if id == hlist or id == vlist then
24       if is_marked(mark, n.head) then return true end
25     elseif id == whatsit and n.subtype == user_defined
26       and n.user_id == whatsit_id and n.value == mark then
27       return true
28     end
29   end
30   return false
31 end
32 local function assert_unmarked(mark, list, ...)
33   local marked = is_marked(mark, list)
34   if marked then
35     tex.error("Multiple alignment marks", "I found multiple alignment marks \z
36       of type " .. mark .. " in an alignment where I already had an \z
37       alignment mark of that type. You should look at both of them and \z
38       decide which one is right. I will continue with the first one for now.")
39   end
40   return ...
41 end
42 local measure do
43   local vmeasure
44   local function hmeasure(mark, list)
45     local x, last = 0, list.head
46     for n in node.traverse(last) do
47       local id = n.id
48       if id == hlist then
49         local w, h, d = node.rangedimensions(list, last, n)
50         x, last = x + w, n
```

```

51     local dx = hmeasure(mark, n)
52     if dx then return assert_unmarked(mark, n.next, dx + x) end
53 elseif id == vlist then
54     local w, h, d = node.rangedimensions(list, last, n)
55     x, last = x + w, n
56     local dx = vmeasure(mark, n)
57     if dx then return assert_unmarked(mark, n.next, dx + x) end
58 elseif id == whatsit and n.subtype == user_defined
59     and n.user_id == whatsit_id and n.value == mark then
60     local w, h, d = node.rangedimensions(list, last, n)
61     local after
62     list.head, after = node.remove(list.head, n)
63     return assert_unmarked(mark, after, x + w)
64 end
65 end
66 end
67 function vmeasure(mark, list)
68     for n in node.traverse(list.head) do
69         local id = n.id
70         if id == hlist then
71             local dx = hmeasure(mark, n)
72             if dx then return assert_unmarked(mark, n.next, dx + n.shift) end
73         elseif id == vlist then
74             local dx = vmeasure(mark, n)
75             if dx then return assert_unmarked(mark, n.next, dx + n.shift) end
76         elseif id == whatsit and n.subtype == user_defined
77             and n.user_id == whatsit_id and n.value == mark then
78             local after
79             list.head, after = node.remove(list.head, n)
80             return assert_unmarked(mark, after, 0)
81         end
82     end
83 end
84 function measure(mark, head)
85     local x, last = 0, head
86     for n in node.traverse(last) do
87         local id = n.id
88         if id == hlist then
89             local w, h, d = node.dimensions(last, n)
90             x, last = x + w, n
91             local dx = hmeasure(mark, n)
92             if dx then return assert_unmarked(mark, n.next, head, dx + x) end
93         elseif id == vlist then
94             local w, h, d = node.dimensions(last, n)
95             x, last = x + w, n
96             local dx = vmeasure(mark, n)
97             if dx then return assert_unmarked(mark, n.next, head, dx + x) end
98         elseif id == whatsit and n.subtype == user_defined
99             and n.user_id == whatsit_id and n.value == mark then
100            local w, h, d = node.dimensions(last, n)
101            local after
102            head, after = node.remove(head, n)
103            return assert_unmarked(mark, after, head, x + w)
104        end

```

```

105     end
106     return head
107 end
108 end
109
110 local isolate do
111     local visolate
112     local function hisolate(list, offset)
113         local x, last = 0, list.head
114         local newhead, newtail = nil, nil
115         local n = last
116         while n do
117             local id = n.id
118             if id == hlist then
119                 local w, h, d = node.rangedimensions(list, last, n)
120                 x, last = x + w, n
121                 local inner_head, inner_tail, new_offset = hisolate(n, offset - x)
122                 if inner_head then
123                     if newhead then
124                         newtail.next, inner_head.prev = inner_head, newtail
125                     else
126                         newhead = inner_head
127                     end
128                     newtail = inner_tail
129                     offset = x + new_offset
130                 end
131                 n = n.next
132             elseif id == vlist then
133                 local w, h, d = node.rangedimensions(list, last, n)
134                 x, last = x + w, n
135                 local inner_head, inner_tail, new_offset = visolate(n, offset - x)
136                 if inner_head then
137                     if newhead then
138                         newtail.next, inner_head.prev = inner_head, newtail
139                     else
140                         newhead = inner_head
141                     end
142                     newtail = inner_tail
143                     offset = x + new_offset
144                 end
145                 n = n.next
146             elseif id == whatsit and n.subtype == user_defined
147                 and n.user_id == whatsit_id then
148                 local w, h, d = node.rangedimensions(list, last, n)
149                 x = x + w
150                 list.head, last = node.remove(list.head, n)
151                 if x ~= offset then
152                     local k = node.new(glue)
153                     k.width, offset = x - offset, x
154                     newhead, newtail = node.insert_after(newhead, newtail, k)
155                 end
156                 newhead, newtail = node.insert_after(newhead, newtail, n)
157                 n = last
158             else

```

```

159         n = n.next
160     end
161 end
162     return newhead, newtail, offset
163 end
164 function visolate(list, offset)
165     local newhead, newtail = nil, nil
166     local n = list.head
167     while n do
168         local id = n.id
169         if id == hlist then
170             if dx then return assert_unmarked(mark, n.next, dx + n.shift) end
171             local inner_head, inner_tail, new_offset = hisolate(n, offset)
172             if inner_head then
173                 if newhead then
174                     newtail.next, inner_head.prev = inner_head, newtail
175                 else
176                     newhead = inner_head
177                 end
178                 newtail = inner_tail
179                 offset = new_offset
180             end
181             n = n.next
182         elseif id == vlist then
183             if dx then return assert_unmarked(mark, n.next, dx + n.shift) end
184             local inner_head, inner_tail, new_offset = visolate(n, offset)
185             if inner_head then
186                 if newhead then
187                     newtail.next, inner_head.prev = inner_head, newtail
188                 else
189                     newhead = inner_head
190                 end
191                 newtail = inner_tail
192                 offset = new_offset
193             end
194             n = n.next
195         elseif id == whatsit and n.subtype == user_defined
196             and n.user_id == whatsit_id then
197             local after
198             list.head, after = node.remove(list.head, n)
199             if 0 ~= offset then
200                 local k = node.new(glue)
201                 k.width, offset = -offset, 0
202                 newhead, newtail = node.insert_after(newhead, newtail, k)
203             end
204             newhead, newtail = node.insert_after(newhead, newtail, n)
205             n = last
206         else
207             n = n.next
208         end
209     end
210     return newhead, newtail, offset
211 end
212 function isolate(head)

```

```

213 local x, last = 0, head
214 local newhead, newtail, offset = nil, nil, 0
215 local n = last
216 while n do
217     local id = n.id
218     if id == hlist then
219         local w, h, d = node.dimensions(last, n)
220         x, last = x + w, n
221         local inner_head, inner_tail, new_offset = hisolate(n, offset - x)
222         if inner_head then
223             if newhead then
224                 newtail.next, inner_head.prev = inner_head, newtail
225             else
226                 newhead = inner_head
227             end
228             newtail = inner_tail
229             offset = x + new_offset
230         end
231         n = n.next
232     elseif id == vlist then
233         local w, h, d = node.dimensions(last, n)
234         x, last = x + w, n
235         local inner_head, inner_tail, new_offset = visolate(n, offset - x)
236         if inner_head then
237             if newhead then
238                 newtail.next, inner_head.prev = inner_head, newtail
239             else
240                 newhead = inner_head
241             end
242             newtail = inner_tail
243             offset = x + new_offset
244         end
245         n = n.next
246     elseif id == whatsit and n.subtype == user_defined
247         and n.user_id == whatsit_id then
248         local w, h, d = node.dimensions(last, n)
249         x = x + w
250         head, last = node.remove(head, n)
251         if x ~= offset then
252             local k = node.new(glue)
253             k.width, offset = x - offset, x
254             newhead, newtail = node.insert_after(newhead, newtail, k)
255         end
256         newhead, newtail = node.insert_after(newhead, newtail, n)
257         n = last
258     else
259         n = n.next
260     end
261 end
262 return head, newhead
263 end
264 end
265
266 local function find_mmode_boundary()

```

```

267   for i=tex.nest.ptr,0,-1 do
268     local nest = tex.nest[i]
269     if nest.mode ~= mmode and nest.mode ~= -mmode then
270       return nest, i
271     end
272   end
273 end
274
275 luatexbase.add_to_callback('post_mlist_to_hlist_filter', function(n)
276   local nest = find_mmode_boundary()
277   local props = properties[nest.head]
278   local alignment = props and props.luamathalign_alignment
279   if alignment then
280     props.luamathalign_alignment = nil
281     local x
282     n, x = measure(alignment.mark, n)
283     local k = node.new'glue'
284     local off = x - n.width
285     k.width, alignment.afterkern.width = off, -off
286     node.insert_after(n.head, nil, k)
287     n.width = x
288   end
289   return n
290 end, 'luamathalign')
291

```

The glue node is referred to as a kern for historical reasons. A glue node is used since this interacts better with lua-ul.

```

292 local function get_kerntoken(newmark)
293   local nest = find_mmode_boundary()
294   local props = properties[nest.head]
295   if not props then
296     props = {}
297     properties[nest.head] = props
298   end
299   if props.luamathalign_alignment then
300     tex.error('Multiple alignment classes trying to control the same cell')
301     return token.new(0, 0)
302   else
303     local afterkern = node.new'glue'
304     props.luamathalign_alignment = {mark = newmark, afterkern = afterkern}
305     return token.new(node.direct.todirect(afterkern), node_cmd)
306   end
307 end
308
309 local function insert_whatsit(mark)
310   local n = node.new(whatsit, user_defined)
311   n.user_id, n.type, n.value = whatsit_id, string.byte'd', mark
312   node.write(n)
313 end
314 luacmd("SetAlignmentPoint", function()
315   local mark = token.scan_int()
316   if mark < 0 then
317     for i=tex.nest.ptr,0,-1 do
318       local t = tex.nest[i].head

```

```

319     local props = properties[t]
320     if props and props.luamathalign_context ~= nil then
321         mark = mark + 1
322         if mark == 0 then
323             props.luamathalign_context = true
324             return insert_whatsit(-i)
325         end
326     end
327 end
328 tex.error('No compatible alignment environment found',
329     'This either means that \\SetAlignmentPoint was used outside\n\z
330     of an alignment or the used alignment is not setup for use with\n\z
331     luamathalign. In the latter case you might want to look at\n\z
332     non-negative alignment marks.')
```

```

333 else
334     return insert_whatsit(mark)
335 end
336 end, "protected")
337
338 function handle_whatsit(mark)
339     token.put_next(ampersand, get_kerntoken(mark))
340 end
341 luacmd("ExecuteAlignment", function()
342     return handle_whatsit(token.scan_int())
343 end, "protected")
344
345 luacmd("LuaMathAlign@begin", function()
346     local t = tex.nest.top.head
347     local props = properties[t]
348     if not props then
349         props = {}
350         properties[t] = props
351     end
352     props.luamathalign_context = false
353 end, "protected")
354 luacmd("LuaMathAlign@end@early", function()
355     local t = tex.nest.top.head
356     local props = properties[t]
357     if props then
358         if props.luamathalign_context == true then
359             handle_whatsit(-tex.nest.ptr)
360         end
361         props.luamathalign_context = nil
362     end
363 end, "protected")
364 local delayed
365 luacmd("LuaMathAlign@end", function()
366     local t = tex.nest.top.head
367     local props = properties[t]
368     if props then
369         if props.luamathalign_context == true then
370             assert(not delayed)
371             delayed = {get_kerntoken(-tex.nest.ptr), ampersand}
372         end

```

```

373     props.luamathalign_context = nil
374   end
375 end, "protected")
376 luatexbase.add_to_callback("hpack_filter", function(head, groupcode)
377   if delayed and groupcode == "align_set" then
378     -- HACK: token.put_next puts the tokens into the input stream after the cell
379     -- is fully read, before the next starts. This will act as if the content was
380     -- written as the first element of the next field.
381     token.put_next(delayed)
382     delayed = nil
383   end
384   return true
385 end, "luamathalign.delayed")
386
387 luacmd("LuaMathAlign@IsolateAlignmentPoints", function()
388   local main = token.scan_int()
389   if not token.scan_keyword 'into' then
390     tex.error'Expected "into"'
391   end
392   local marks = token.scan_int()
393   local head, newhead = isolate(tex.box[main])
394   tex.box[marks] = node.direct.tonode(node.direct.hpack(
395     newhead and node.direct.todirect(newhead) or 0))
396 end, "protected")

```

3.2 LaTeX

The actual L^AT_EX package just loads the Lua module and patches amsmath:

```

397 \RequirePackage{iftex}
398 \RequireLuaTeX
399 \directlua{require'luamathalign'}
400 \IfPackageLoadedTF{amsmath}{%
401   \@firstofone
402 }{%
403   \AddToHook{package/amsmath/after}
404 }
405 {%
406   \def\align@preamble{%
407     &\hfil
408     \strut@
409     \setboxz@h{\@lign$\m@th\displaystyle{%
410       \LuaMathAlign@begin##\LuaMathAlign@end}$}%
411     \ifmeasuring@\savefieldlength@fi
412     \set@field
413     \tabskip\z@skip
414     &\setboxz@h{\@lign$\m@th\displaystyle{##}$}%
415     \ifmeasuring@\savefieldlength@fi
416     \set@field
417     \hfil
418     \tabskip\alignsep@
419   }
420   \renewcommand{\start@aligned}[2]{%
421     \RIfM@else
422       \nonmatherr@{\begin{\@currenvir}}%
423     \fi

```

```

424 \savecolumn@ % Assumption: called inside a group
425 \alignedspace@left
426 \if #1t\vtop \else \if#1b \vbox \else \vcenter \fi \fi \bgroup
427 \maxfields@#2\relax
428 \ifnum\maxfields@>\m@ne
429 \multiply\maxfields@\tw@
430 \let\math@cr@@@math@cr@@@alignedat
431 \alignsep@\z@skip
432 \else
433 \let\math@cr@@@math@cr@@@aligned
434 \alignsep@\minalignsep
435 \fi
436 \Let@ \chardef\dspbrk@context\@ne
437 \default@tag
438 \spread@equation % no-op if already called
439 \global\column@\z@
440 \ialign\bgroup
441 &\column@plus
442 \hfil
443 \strut@
444 $\m@th\displaystyle{\LuaMathAlign@begin##\LuaMathAlign@end}$%
445 \tabskip\z@skip
446 &\column@plus
447 $\m@th\displaystyle{{}##}$%
448 \hfil
449 \tabskip\alignsep@
450 \crr
451 }
452 \edef\math@cr@@@alignedat{\LuaMathAlign@end@early
453 \unexpanded\expandafter{\math@cr@@@alignedat}}
454 \edef\math@cr{\LuaMathAlign@end@early
455 \unexpanded\expandafter{\math@cr}}
456 \edef\endaligned{\LuaMathAlign@end@early
457 \unexpanded\expandafter{\endaligned}}
458 }
459 \protected\def\AlignHere{\SetAlignmentPoint\m@ne}
460 \begingroup
461 \def\patch@finph@nt\setbox\tw@\null{%
462 \LuaMathAlign@IsolateAlignmentPoints\z@ into \tw@
463 }%
464 \expanded{\endgroup%
465 \protected\def\noexpand\finph@nt{%
466 \unexpanded\expandafter\expandafter\expandafter{%
467 \expandafter\patch@finph@nt\finph@nt
468 }%
469 }}
470 \ExplSyntaxOff

```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols		M	
<code>\</code>	329	<code>\minalignsep</code>	434
A		<code>\multiply</code>	429
<code>\AddToHook</code>	403	N	
<code>\AlignHere</code>	2, 3, 459	<code>\n</code>	329, 330, 331
B		<code>\noexpand</code>	465
<code>\begin</code>	422	<code>\null</code>	461
<code>\begingroup</code>	460	P	
<code>\bgroup</code>	426, 440	<code>\protected</code>	459, 465
C		R	
<code>\chardef</code>	436	<code>\relax</code>	427
<code>\crcr</code>	450	<code>\renewcommand</code>	420
D		<code>\RequireLuaTeX</code>	398
<code>\def</code>	406, 459, 461, 465	<code>\RequirePackage</code>	397
<code>\directlua</code>	399	S	
<code>\displaystyle</code>	409, 414, 444, 447	<code>\SetAlignmentPoint</code>	3, 459
E		<code>\setbox</code>	461
<code>\edef</code>	452, 454, 456	T	
<code>\else</code>	421, 426, 432	<code>\tabskip</code>	413, 418, 445, 449
<code>\endaligned</code>	456, 457	TeX and L ^A T _E X 2 _ε commands:	
<code>\endgroup</code>	464	<code>\@currenvir</code>	422
<code>\ExecuteAlignment</code>	3	<code>\@firstofone</code>	401
<code>\expandafter</code>	453, 455, 457, 466, 467	<code>\@lign</code>	409, 414
<code>\expanded</code>	464	<code>\@one</code>	436
<code>\ExplSyntaxOff</code>	470	<code>\align@preamble</code>	406
F		<code>\alignedspace@left</code>	425
<code>\fi</code>	411, 415, 423, 426, 435	<code>\alignsep@</code>	418, 431, 434, 449
G		<code>\column@</code>	439
<code>\global</code>	439	<code>\column@plus</code>	441, 446
H		<code>\default@tag</code>	437
<code>\hfil</code>	407, 417, 442, 448	<code>\dspbrk@context</code>	436
I		<code>\finph@nt</code>	465, 467
<code>\ialign</code>	440	<code>\ifmeasuring@</code>	411, 415
<code>\if</code>	426	<code>\Let@</code>	436
<code>\ifnum</code>	428	<code>\LuaMathAlign@begin</code>	410, 444
<code>\IfPackageLoadedTF</code>	400	<code>\LuaMathAlign@end</code>	410, 444
L		<code>\LuaMathAlign@end@early</code>	452, 454, 456
<code>\let</code>	430, 433	<code>\LuaMathAlign@IsolateAlignmentPoints</code>	462
		<code>\m@ne</code>	428, 459
		<code>\m@th</code>	409, 414, 444, 447
		<code>\math@cr</code>	454, 455
		<code>\math@cr@@@</code>	430, 433
		<code>\math@cr@@@aligned</code>	433

