1 Overview

The overall idea of the procedure is to identify operators that are suitable for line-breaking and categorise them with respect to the quality of a line-break at that position. We thereby compute a penalty for each eligible operator, that takes into account

(i) the nesting depth of the operator,

(ii) the parent structure it is contained in (e.g., a line-break in a compound expression like an integral or a sum should be less likely than one outside), and

(iii) how evenly a line-break at that position would split an expression.

The procedure consists effectively of 4 major steps, some of them are broken down into several algorithms:

1. **Canonicalisation** performs a general rewrite of a MathML, so it can be handled expressions as uniformly as possible, while identifying operators for line-breaking.
   (a) **Cleanup** rewrites "undesirable" tags, removes superfluous elements and combines elements as much as possible without disturbing potential semantic meaning of groups.
   (b) **Chunking** Introduces additional layers by identifying and combining sub-expressions that have some semantic connection (e.g., identifying sums or integrals).
   (c) **Operator Selection** finds and marks the operators that are eligible for line-breaking.
2. **Horizontal Complexity** recursively calculates an abstract complexity measure for the assumed horizontal space an expressions takes in the rendered layout.
3. **Break Quality** calculates for each target operated a quality value for line-breaking taking it contextual position into account.
   (a) **Real Depth** calculates the depth of eligible operators in the tree modulo types of expressions we want to exclude.
   (b) **Parental Complexity Factor** relates the size of a target expression against size of its context.
   (c) **Gap Complexity** represents the horizontal space between target operators on the same depth level.
4. **Clustering** discretises all target operators into k clusters of distinct line-breaking quality (e.g., a 3-clustering could result in goodbreak, badbreak, nobreak).

Definitions

We define some sets of that we will use in the remainder.

$$\begin{array}{lcl}
\mathcal{TOKEN} & = & \{\texttt{<mi>}, \texttt{<mn>}, \texttt{<mo>}, \texttt{<mtext>}, \texttt{<mspace>}, \texttt{<ms>}\} \\
\mathcal{LAYOUT} & = & \{\texttt{<mfrac>}, \texttt{<msqrt>}, \texttt{<mroot>}, \texttt{<mfenced>}, \texttt{<menclose>}\} \\
\mathcal{TABLES} & = & \{\texttt{<mtable>}, \texttt{<mlabeledtr>}, \texttt{<mtr>}, \texttt{<mtd>}, \texttt{<maligngroup>}, \texttt{<malignmark>}\} \\
\mathcal{SCRIPT} & = & \{\texttt{<msub>}, \texttt{<msup>}, \texttt{<msubsup>}, \texttt{<munder>}, \texttt{<mover>}, \texttt{<munderover>}, \texttt{<mmultiscripts>}\} \\
\mathcal{ELEMENTARY} & = & \{\texttt{<mstack>}, \texttt{<mlongdiv>}, \texttt{<msgroup>}, \texttt{<msrow>}, \texttt{<mscarries>}, \texttt{<mscarry>}, \texttt{<msline>}\} \\
\mathcal{SPACE} & = & \{\texttt{<msrow>}, \texttt{<mrow>}, \texttt{<mspace>}, \texttt{<mstyle>}, \texttt{<mphantom>}, \texttt{<mpadded>}, \texttt{<none>}\} \\
\mathcal{INFERRED} & = & \{\texttt{<msqrt>}, \texttt{<mstyle>}, \texttt{<mphantom>}, \texttt{<mpadded>}, \texttt{<merror>}, \texttt{<menclose>}, \texttt{<mtd>}, \texttt{<math>}, \texttt{<mscarry>}\}
\end{array}$$

2 Canonicalisation

The initial algorithm of the canonicalisation performs a general cleanup of our MathML element.

Algorithm 1 Cleanup on MathML expression E.

procedure CLEANUP(E)
  case `<mfenced>`:
    $F \leftarrow$ `<mfenced>`$C_0, \dots, C_m$`</mfenced>`
    if `separators` $\notin F$ **then** ▷ Case a
      $O_1 \leftarrow \cdots \leftarrow O_m \leftarrow$ `<mo>,</mo>`
    else if `separators="s"` $\in F$ **then** ▷ Case b
      $O_1 \leftarrow \cdots \leftarrow O_m \leftarrow$ `<mo>`s`</mo>`
    else if `separators="`$s_1 s_2 ... s_n$`"` $\in F$ **then** ▷ Case c
      $O_1 \leftarrow$ `<mo>`s_1`</mo>`$, \dots, O_{n-1} \leftarrow$ `<mo>`s_{n-1}`</mo>`
      $O_n \leftarrow \cdots \leftarrow O_m$`<mo>`$s_n$`</mo>`
    if `open="l"` $\in F$ **then** ▷ Case d
      $L \leftarrow$ `<mo>`l`</mo>`
    else ▷ Case e
      $L \leftarrow$ `<mo>(</mo>`
    Add `meaning="open"` to L
    if `close="r"` $\in F$ **then** ▷ Case f
      $R \leftarrow$ `<mo>`r`</mo>`
    else ▷ Case g
      $R \leftarrow$ `<mo>)</mo>`
    Insert R as last child
    Add `meaning="close"` to R
    $M \leftarrow$ `<mrow>`$L\, C_o O_1 C_1 \dots O_n C_n\, R$`</mrow>`
    Add `meaning="fenced"` to M
    Replace F by M
  end case
  case `<mfrac>`:
    if `linethickness="0"` **then**
      $F \leftarrow$ `<mfrac>`$C_1 C_2$`</mfrac>`
      $T \leftarrow$ `<mtable><mtr><mtd>`C_1`</mtd></mtr><mtr><mtd>`C_2`</mtd></mtr></mtable>`
      Replace F by T
  end case
  case `<mrow>`1: ▷ Merges unnecessary `<mrow>`s on the same level
    $R \leftarrow$ `<mrow>`$C_1 \dots C_n$`</mrow>`
    $R' \leftarrow$ `<mrow>`$D_1 \dots D_m$`</mrow>`
    if `<p>`$\dots RR' \dots$`</p>` $\wedge\, p \in \mathcal{INFERRED} \wedge$ `meaning` $\notin R \wedge$ `meaning` $\notin R'$ **then**
    ▷ I.e., mrows are direct neighbours and not strictly necessary
      Replace RR' by `<mrow>`$C_1 \dots C_n D_1 \dots D_m$`</mrow>`
  end case
  case `<mrow>`2: ▷ Removes nested, singleton `<mrow>`s
    $R \leftarrow$ `<mrow>`$C_1 \dots C_n$`</mrow>`
    $P \leftarrow$ `<t>`R`</t>`
    if $t \in \mathcal{INFERRED}$ **then**
      Replace R by $C_1 \dots C_n$
      if `meaning` $\in R$ **then**
        Add `meaning` to P
  end case

case <mspace>:
  if $F \leftarrow$ <t>$u_1, \ldots, u_n$</t>$, t \in \mathcal{TOKEN} \setminus \{$<ms>, <mspace>$\} \wedge u_1 \ldots u_n \in \mathcal{SPACE}$ **then**
    Replace F by <mspace width="U"/>, where $U = \sum_{j=1}^{n} u'_j$ ▷ Case a
  else
    if $F \leftarrow$ <t>$u_1, \ldots, u_n$</t>$, t \in \mathcal{TOKEN} \setminus \{$<ms>, <mspace>$\} \wedge u_1 \ldots u_i \in \mathcal{SPACE}, i < n$ **then**
      $S \leftarrow$ <mspace width="U"/>, where $U = \sum_{j=1}^{i} u'_j$
      $R \leftarrow$ <t>$u_{i+1} \ldots u_n$</t>
      if <p>$\ldots F \ldots$</p> $\wedge\, p \in \mathcal{INFERRED}$ **then** ▷ Case b
        Replace F by $S\,R$
      else
        Replace F by <mrow>$S\,R$</mrow> ▷ Case c
    if $F \leftarrow$ <t>$u_1, \ldots, u_n$</t>$, t \in \mathcal{TOKEN} \setminus \{$<ms>, <mspace>$\} \wedge u_i \ldots u_n \in \mathcal{SPACE}, i > 1$ **then**
      $S \leftarrow$ <mspace width="U"/>, where $U = \sum_{j=i}^{n} u'_j$
      $R \leftarrow$ <t>$u_1 \ldots u_{i-1}$</t>
      if <p>$\ldots F \ldots$</p> $\wedge\, p \in \mathcal{INFERRED}$ **then** ▷ Case d
        Replace F by $R\,S$
      else
        Replace F by <mrow>$R\,S$</mrow> ▷ Case e
end case

Note, that in the <mrow>cases we only allow to merge or remove <mrow>elements that do not contain any additional attributes. This ensures that any information on stretchy characters, etc. is retained at the correct position.

Note also, that for the final case of the algorithm above we introduced a set $\mathcal{SPACE}$ that we define as the set of all unicode horizontal non-breaking whitespace characters.

$$\begin{aligned}\mathcal{SPACE} = \{&\texttt{U+00A0}, \texttt{U+2000}, \texttt{U+2001}, \texttt{U+2002}, \texttt{U+2003}, \texttt{U+2004}, \texttt{U+2005}, \texttt{U+2006},\\ &\texttt{U+2007}, \texttt{U+2008}, \texttt{U+2009}, \texttt{U+200A}, \texttt{U+202F}, \texttt{U+205F}, \texttt{U+3000}, \texttt{U+200B},\\ &\texttt{U+200C}, \texttt{U+200D}, \texttt{U+2060}, \texttt{U+FEFF}\}\end{aligned}$$

Note that in the <mspace>case above, we strip whitespaces from the left and right of potential content. Note also, that the u'_i correspond to the MathML attribute equivalent to the actual Unicode whitespace.

The Chunking algorithm imposes a more refined hierarchical structure on the MathML expression, by recognising semantically interesting components, such as big operator applications, integrals, etc.; introducing additional mrows, as well as invisible operators and function applications.

We define some additional sets of tags and unicode characters.

$$\mathcal{MULTS} = \{\texttt{<mn>}, \texttt{<mi>}\} \cup \mathcal{LAYOUT} \cup \mathcal{SCRIPT}$$

Let $\mathcal{BIGOPS}$ be the set of all Unicode n-ary operators:

$$\begin{aligned}\mathcal{BIGOPS} = \{&\texttt{U+2140}, \texttt{U+220F}, \texttt{U+2210}, \texttt{U+2211}, \texttt{U+22C0}, \texttt{U+22C1}, \texttt{U+22C2}, \texttt{U+22C3}, \texttt{U+2A00},\\ &\texttt{U+2A01}, \texttt{U+2A02}, \texttt{U+2A03}, \texttt{U+2A04}, \texttt{U+2A05}, \texttt{U+2A06}, \texttt{U+2A07}, \texttt{U+2A08}, \texttt{U+2A09},\\ &\texttt{U+2A0A}, \texttt{U+2A0B}, \texttt{U+2AFC}, \texttt{U+2AFF}\}\end{aligned}$$

Let $\mathcal{INTEGRALS}$ be the set of all Unicode integral symbols:

$$\begin{aligned}\mathcal{INTEGRALS} = \{&\texttt{U+222B}, \texttt{U+222C}, \texttt{U+222D}, \texttt{U+222E}, \texttt{U+222F}, \texttt{U+2230}, \texttt{U+2231}, \texttt{U+2232}, \texttt{U+2233},\\ &\texttt{U+2A0C}, \texttt{U+2A0D}, \texttt{U+2A0E}, \texttt{U+2A0F}, \texttt{U+2A10}, \texttt{U+2A11}, \texttt{U+2A12}, \texttt{U+2A13}, \texttt{U+2A14},\\ &\texttt{U+2A15}, \texttt{U+2A16}, \texttt{U+2A17}, \texttt{U+2A18}, \texttt{U+2A19}, \texttt{U+2A1A}, \texttt{U+2A1B}, \texttt{U+2A1C}\}\end{aligned}$$

Algorithm 2 Chunking of MathML expression E

procedure CHUNKING(E)
  step 1: ▷ Embellished Operators
    $\mathcal{M} \leftarrow \{m \in E | m =$ `<mo>`t`</mo>`$\}$
    for $M =$ `<mo>`t`</mo>` $\in \mathcal{M}$ **do**
      while $P =$ `<p>`$Mc_1 \ldots$`</p>` $\wedge\, p \in \mathcal{SCRIPT}$ **do** ▷ M is first child of a script node.
        Add `embellished="`t`"` to P
        $M \leftarrow P$
      Add `meaning="operator"` to M
  end step
  step 2: ▷ Fence Matching
    Run balanced fence algorithm on a single (inferred) row to obtain $\mathcal{P}$ ▷ See [1] for more details.
    Let $\mathcal{P} = \{($`<t meaning="operator">`f`</t>`, `<t' meaning="operator">`f'`</t'>`$) | f, f'$ matching fences$\}$
    for Every $P \in \mathcal{P}$ **do**
      Replace `<t meaning="operator">`f`</t>`$S_1 \ldots S_n$`<t' meaning="operator">`f'`</t'>` by
      `<mrow meaning="fenced"><t meaning="open">`f`</t>`$S_1 \ldots S_n$`<t' meaning="close">`f'`</t'></mrow>`
  end step
  step 3: ▷ Chunking
    $\mathcal{T} = \{\}$
    Perform Depth-First Traversal of expression E
    $M \leftarrow$ `<p>`$C_1 \ldots C_n$`</p>`
    if $p \notin \mathcal{INFERRED}$ **then**
      recurse $C_1 \ldots C_n$
    for $i \leftarrow n$ to 1 **do**
      ▷ Big Operators
      case $(C_i =$ `<mo meaning="operator">`t`</mo>` $\vee$ `embellished="`t`"` $\in C_i) \wedge t \in \mathcal{BIGOPS}$:
        Find C_k following sibling of C_i s.t. `meaning="operator"` $\in C_k$
        Otherwise $C_k = C_n$
        Replace $C_i \cdots C_k$ with `<mrow meaning="bigop">`$C_i \cdots C_k$`</mrow>`
        Set `meaning="bigop"` in C_i
      end case
      ▷ Integrals
      case $(C_i =$ `<mo meaning="operator">`t`</mo>`$\vee$`embellished="`t`"` $\in C_i) \wedge t \in \mathcal{INTEGRALS}$:
        Find C_k following sibling of C_i s.t. $C_k =$ `<mi>`du`</mi>` for any u.
        Otherwise find C_k following sibling of C_i s.t. `meaning="operator"` $\in C_k$
        Otherwise $C_k = C_n$
        Replace $C_i \cdots C_k$ with `<mrow meaning="integral">`$C_i \cdots C_k$`</mrow>`
        Set `meaning="integral"` in C_i
      end case
      ▷ Postfix Operators
      case $(i = n \wedge$ `meaning="operator"` $\in C_i) \vee$ `meaning="postfix"` $\in C_{i+1}$:
        Set `meaning="postfix"` in C_i
      end case
      ▷ Prefix Operators
      case `meaning="operator"` $\in C_i \wedge ($`meaning="operator"` $\in C_{i-1} \vee i = 1$:
        Set `meaning="prefix"` in C_i
      end case
      ▷ Invisible Times
      case $(C_i \in \mathcal{MULTS} \vee$ `meaning="fenced"` $\in C_i) \wedge$
      $(C_{i+1} \in \mathcal{MULTS} \vee$ `meaning="fenced"` $\in C_{i+1})$:
        $N \leftarrow$ `<mo>U+2062</mo>`
        $\mathcal{T} = \mathcal{T} \cup \{N\}$

Replace C_iC_{i+1} with C_iNC_{i+1}
end case
▷ Function Application
case $C_i =$ `<mi>`$u_1 \dots u_n$`</mi>` $\wedge\, n \geq 2 \wedge$ `meaning="fenced"` $\in C_{i+1}$:
$N \leftarrow$ `<mo>U+2061</mo>`
$\mathcal{T} = \mathcal{T} \cup \{N\}$
Replace C_iC_{i+1} with `<mrow>`C_iNC_{i+1}`</mrow>`
end case ▷ Remark: Embellished functions will be treated in the previous case!
End Depth-First Traversal
for $M \in T$ **do**
Add `meaning="operator"` to M
end step

---

[1] Here suggest using Balanced Parenthesis algorithm using a stack as main datastructure. Example implementation can be found at:

- `http://www.ardendertat.com/2011/11/08/programming-interview-questions-14-check-balanced-parenthese`
- `https://dzone.com/articles/balanced-parenthesis-check`
- `http://clarkfeusier.com/2015/01/16/interview-question-balanced-parentheses-stack.html`

In our case we are not interested in a boolean decision, but to compute the actual pairs of balanced fences, hence they need to be recorded. In addition we have three more alterations:

1. We extend the standard algorithm to deal with pairs neutral fences, i.e., $|, \|, \dots$
2. We ignore any fence that do not match an opening fence. In particular we ignore neutral fences, if a closing and matching closing fence can be found. E.g., the neutral fence in $\{\dots|\dots\}$ should be ignored!
3. An exception has to be made for opening fences before tables, as these generally denote case statements.

Note: This algorithm could be further improved by using a linear programming approach to find the maximum number of matching fences. Note also, that this

Markup of elligible breaking operators. We first define a set of "stop tags", that is, operators that are children of these tags are never considered to be elligible for linebreaking.

Algorithm 3 Operator Selection on expression E; returns set of target operators T.

procedure OPERATOR SELECTION(E)
$\mathcal{S} \leftarrow \mathcal{SCRIPT} \cup \mathcal{LAYOUT} \cup \{$`<mphantom>`, `<mtable>`$\}$
$\mathcal{T} = \{\}$
if `meaning="operator"` $\in E$ **then**
$\mathcal{T} = \mathcal{T} \cup \{E\}$
else if $E =$ `<p>`$C_1 \dots C_n$`</p>` $\wedge\, p \notin \mathcal{S}$ **then**
OPERATOR SELECTION(C_1), ..., OPERATOR SELECTION(C_n)
return T

---

3 Calculating `linebreak` values

Calculating values for `linebreak` attributes consists of three procedures:

1. **Horizontal Complexity** recursively calculates an abstract complexity measure for the assumed horizontal space an expressions takes in the rendered layout.

2. **Gap Complexity** calculates the horizontal complexity between target operators on the same depth level.

3. **Break Quality** calculates for each target operated a quality value for line-breaking taking it contextual position into account.

   (a) **Real Depth** calculates the depth of eligible operators in the tree modulo types of expressions we want to exclude.

   (b) **Parental Complexity Factor** relates the size of a target expression against size of its context.

   (c) **Gap Complexity Factor** relates the size of the gap between target operators against the size of its context.

4. **Clustering** discretises all target operators into k clusters of distinct line-breaking quality (e.g., a 3-clustering could result in goodbreak, badbreak, nobreak).

3.1 Complexity measure

The complexity measure provides a heuristic estimate the width (in `em`) of subexpressions.

Note. To improve efficiency, this value can be calculated while calcluating the Gap Complexity Measure (cf. the next subsection).

3.1.1 Helpers

Algorithm 4 Auxiliary procedure for transforming alpha to numeric sizes

```
procedure MathsizeFactor(E)
    f ← 1
    mathsize ← E.getAttribute('mathsize')
    if mathsize ='small' then f ← 0.71
    else if mathsize ='big' then f ← 2
    else if mathsize = length then f ← length      ▷ We assume such lengths are given in em
    return f
```

3.1.2 Main Algorithm

Algorithm 5 Complexity measure

```
procedure Complexity
    case text = u_1 ... u_n:        ▷ u_i are either Unicode Points or Unicode graphemes
        sum ← 0
        for i ← 1, n do
            if u_i is invisible then        ▷ E.g., U+2061, U+2062, U+2063, U+2064
                sum ← sum + 0.1             ▷ Note: this avoids division by zero
            if u_i is half-point then
                sum ← sum + 0.5
            if u_i is full-point then
                sum ← sum + 1
        return sum
    end case
    case <mglyph>:
        if width="S" then
            return S                        ▷ We assume such widths are given in em
        else if src="S" then
```

return width of S ▷ Requires external call to get image dimensions
end case
case `<mspace>`, `<mpadded>`:
  if `width="S"` **then**
    return S ▷ We assume such widths are given in `em`
  else
    return 0
end case
case `<mstyle>`:
  $R \leftarrow$ `<mstyle>`$u_1 \dots u_n$`</mstyle>`
  $\mathit{level} \leftarrow 1$
  if R.getAttribute('scriptlevel') **then**
    $\mathit{level} \leftarrow R$.getAttribute('scriptlevel')
  $\mathit{multiplier} \leftarrow 1$
  if R.getAttribute('scriptsizemultiplier') **then**
    $\mathit{multiplier} \leftarrow R$.getAttribute('scriptsizemultiplier')
  $f \leftarrow R$.getAttribute('factor') $\times\ \mathit{multiplier}^{\mathit{level}} \times$ MathSizeFactor(R)
  $\mathit{sum} \leftarrow 1$ ▷ $1 = 0.5 \times 2$ as estimate for horizontal margins
  for $i \leftarrow 1, n$ **do**
    u_i.setAttribute('factor','f')
    $\mathit{sum} \leftarrow \mathit{sum} + \textsc{Complexity}(u_i)$
  return sum
end case
case $\mathcal{TOKEN}$:
  $R \leftarrow$ `<token>`$u_1 \dots u_n$`</token>`
  $\mathit{sum} \leftarrow 1$ ▷ $1 = 0.5 \times 2$ as estimate for horizontal margins
  for $i \leftarrow 1, n$ **do**
    u_i.setAttribute('factor','R.getAttribute('factor')'$\times$MathSizeFactor(R))
    $\mathit{sum} \leftarrow \mathit{sum} + \textsc{Complexity}(u_i)$
  return sum
end case
case `<msub>`, `<msup>`:
  $R \leftarrow$ `<tag>`$u_1 u_2$`</tag>`
  $\mathit{sum} \leftarrow 1$ ▷ $1 = 0.5 \times 2$ as estimate for horizontal margins
  $\mathit{sum} \leftarrow \mathit{sum} + \textsc{Complexity}(u_1)$
  u_2.setAttribute('factor', '$0.71 \times R$.getAttribute('factor')' $\times$MathSizeFactor(R))
  $\mathit{sum} \leftarrow \mathit{sum} + 0.71 \times \textsc{Complexity}(u_2)$ ▷ Default scriptsizemultiplier
  return sum
end case
case `<msubsup>`:
  $R \leftarrow$ `<tag>`$u_1 u_2 u_3$`</tag>`
  $\mathit{sum} \leftarrow 1$ ▷ $1 = 0.5 \times 2$ as estimate for horizontal margins
  u_1.setAttribute('factor', 'R.getAttribute('factor')' $\times$MathSizeFactor(R))
  $\mathit{sum} \leftarrow \mathit{sum} + \textsc{Complexity}(u_1)$
  u_2.setAttribute('factor', '$0.71 \times R$.getAttribute('factor')' $\times$MathSizeFactor(R))
  u_3.setAttribute('factor', '$0.71 \times R$.getAttribute('factor')' $\times$MathSizeFactor(R))
  $\mathit{sum} \leftarrow \mathit{sum} + 0.71 \times \max(\textsc{Complexity}(u_2), \textsc{Complexity}(u_3))$ ▷ Default scriptsizemultiplier 0.71
  return sum
end case
case `<mmultiscripts>`, `<mprescripts>`:
  $R \leftarrow$ `<tag>`$u_1 u_2 \dots u_n$`</tag>` ▷ Note: we assume "full" set of scripts
  $\mathit{sum} \leftarrow 1$ ▷ $1 = 0.5 \times 2$ as estimate for horizontal margins

u_1.setAttribute('factor', 'R.getAttribute('factor')' $\times$MathSizeFactor(R))
$sum \leftarrow sum +$ COMPLEXITY(u_1)
for $i \leftarrow 2, 4, \ldots, n$ **do**
u_i.setAttribute('factor', '$0.71 \times R$.getAttribute('factor')' $\times$MathSizeFactor(R))
u_{i+1}.setAttribute('factor', '$0.71 \times R$.getAttribute('factor')' $\times$MathSizeFactor(R))
$sum \leftarrow sum + 0.71 \times \max($COMPLEXITY$(u_i)$, COMPLEXITY$(u_{i+1}))$ ▷ Default scriptsizemultiplier 0.71
return sum
end case
case `<msqrt>`, `<menclose>`:
$R \leftarrow$ `<tag>`$u_1 \ldots u_n$`</tag>`
$sum \leftarrow 1$ ▷ $1 = 0.5 \times 2$ as estimate for horizontal margins
$sum \leftarrow sum + 1$ ▷ Estimate for surd
for $i \leftarrow 1, n$ **do**
u_i.setAttribute('factor', 'R.getAttribute('factor')' $\times$MathSizeFactor(R))
$sum \leftarrow sum +$ COMPLEXITY(u_i)
return sum
end case
case `<mroot>`:
$R \leftarrow$ `<tag>`$u_1 u_2$`</tag>`
$sum \leftarrow 1$ ▷ $1 = 0.5 \times 2$ as estimate for horizontal margins
$sum \leftarrow sum + 1$ ▷ Estimate for surd
$sum \leftarrow sum +$ COMPLEXITY(u_1) ▷ Note: assumes explicit `<mrow>`
$sum \leftarrow sum + 0.5041 \times$ COMPLEXITY(u_1) ▷ Note: default scriptsizemultiplier at scriptlevel 2 as per spec. Also, no `factor` attribute is necessary.
return sum
end case
case `<mtable>`, `<mstack>`:
$R \leftarrow$ `<tag>`$u_1 \ldots u_n$`</tag>` ▷ Note: we assume explicit table rows
for $i \leftarrow 1, n$ **do**
u_i.setAttribute('factor', 'R.getAttribute('factor')' $\times$MathSizeFactor(R))
$sum \leftarrow sum +$ COMPLEXITY(u_i)
return $1 + \max($COMPLEXITY$(u_1), \ldots,$ COMPLEXITY$(u_n))$
end case
case `<mfrac>`:
$R \leftarrow$ `<tag>`$u_1 \ldots u_n$`</tag>`
for $i \leftarrow 1, n$ **do**
u_i.setAttribute('factor', 'R.getAttribute('factor')'$\times 0.71 \times$ MathSizeFactor(R)) ▷ Default scriptsizemultiplier
$sum \leftarrow sum +$ COMPLEXITY(u_i)
return $1 + \max($COMPLEXITY$(u_1), \ldots,$ COMPLEXITY$(u_n))$
end case
case `<munder>`, `<mover>`, `<munderover>`:
$R \leftarrow$ `<tag>`$u_1 \ldots u_n$`</tag>`
u_1.setAttribute('factor', 'R.getAttribute('factor')' $\times$MathSizeFactor(R))
for $i \leftarrow 2, n$ **do**
$sum \leftarrow sum +$ COMPLEXITY(u_i)
u_i.setAttribute('factor', 'R.getAttribute('factor')'$\times 0.71 \times$ MathSizeFactor(R)) ▷ Default scriptsizemultiplier
return $1 + \max($COMPLEXITY$(u_1), \ldots,$ COMPLEXITY$(u_n))$
end case
case `<math>`, `<mrow>`, `<mphantom>`, `<merror>`, `<msrow>`, `<mtr>`, `<mtr>`, `<mtd>`:
$R \leftarrow$ `<tag>`$u_1 \ldots u_n$`</tag>`

```
        sum ← 0
        for i ← 1, n do
            u_i.setAttribute('factor', 'R.getAttribute('factor')' ×MathSizeFactor(R))
            sum ← sum + Complexity(u_i)
        return sum
    end case
    case <maction>:
        R ← <maction>u_1 ... u_n</maction>
        if actiontype="toggle", selection="S" then
            return Complexity(u_S)
            u_S.setAttribute('factor', 'R.getAttribute('factor')' ×MathSizeFactor(R))
        else
            return Complexity(u_1)
    end case
    case <annotation>, <annotation-xml>:
        return Complexity(textContent)
    end case
```

3.2 Gap Complexity

The Gap Complexity is calculated while traversing the MathML tree depth-first. It records the Horizontal Complexity (cf. the previous subsection) of the subexpression between target operators.

Algorithm 6 Gap Complexity

```
procedure Gap Complexity
    K ← max(full depth)                                   ▷ Use full depth as clustering
    H_i ← 0(i = 1, ldots, K)                              ▷ Hash table
    tag_0 ← <math>
    tag_j ← eligible(j)(j = 1, ..., n)
    case j = 0:
        return 0
    end case
    case j + 1:
        oldDepth ← full depth(<tag_j>)
        newDepth ← full depth(<tag_{j+1}>)
        dgc ← 0                                           ▷ dgc = direct-gap-complexity
        marker ← <tag_j>              ▷ The marker will travel across the tree from tag_j to tag_{j+1}
        while marker ≠ <tag_{j+1}> do
            marker ← (marker.NextSibling()) || marker.parentNode().nextSibling()
            if ancestor(marker, tag_{j+1}) then
                marker ← firstChild(marker)
            else
                dgc ← dgc + complexity(marker) × marker.getAttribute('factor')
        if oldDepth ≤ newDepth then
            H_newDepth ← dgc
            return dgc
        else
            result ← H_oldDepth + ... + H_newDepth
            H_oldDepth ... H_newDepth ← 0
            return result
    end case
```

3.3 Nesting structure

A target is preferred the less nested it is. However, we do not want to overly restrict line-breaking inside fenced expressions, while, on the other hand, we want to break up semantic units like integrals as little as possible. We therefore define skippable `<mrow>`s corresponding to fenced expressions but are not part of a big operator or function application. (Note that here mrow also includes inferred mrow elements).

We then calculate a *real depth* for each target, i.e., basically

$$\text{real depth} = \#\text{ mrow ancestors with meaning} - \#\text{ skippable mrow ancestors}$$

Algorithm 7 Full Depth computation

```
procedure FULL DEPTH(T)
    marker ← T
    realDepth ← 0
    while marker.parentNode() do
        if marker.tagName ∈ INFERRED ∧ meaning ∈ marker then
            realDepth ← 1 + realDepth
        marker ← marker.parentNode()
    return realDepth
```

Algorithm 8 Real Depth computation

```
procedure REAL DEPTH(T)
    marker ← T
    realDepth ← 0
    while marker.parentNode() do
        if marker.tagName ∈ INFERRED ∧ meaning ∈ marker ∧ meaning="fenced" ∉ marker then
            realDepth ← 1 + realDepth
        marker ← marker.parentNode()
    return realDepth
```

Converseley, we need to identify the *real parent* (mrow) for each target, i.e., the highest (in the tree) skippable mrow below the second-lowest non-skippable mrow (if that does not exist, the lowest non-skippable mrow).

Algorithm 9 Real Parent computation

```
procedure REAL PARENT MROW(T)
    marker ← T
    counter ← 1
    realParent ← T
    while marker.parentNode() do
        marker ← marker.parentNode()
        if marker.tagName ∉ INFERRED ∨ meaning ∉ marker ∨ meaning="fenced" ∈ marker then
            realParent ← marker
        if marker.tagName ∈ INFERRED ∧ meaning ∈ marker ∧ meaning="fenced" ∉ marker then
            if counter = 1 then
                counter ← 0
                realParent ← marker
            else BREAK
    return realParent
```

3.4 Parent complexity

If a target is within a very long `<mrow>`, it should be a better break-point.

$$parentalComplexityFactor = \textsc{complexity}(math)/\textsc{complexity}(RealParent(target))$$

This is solves many conundrums from the samples along the lines of "but still break in this particular nested structure"

3.5 Combine the factors to a final penalty

The idea when combining the different complexity is as follows: A large real-depth makes a break point worse. On the other hand a large parent complexity and large gap to previous point makes it a better breakpoint.

We then define the final penalty p for a target operator t as:

$$\begin{aligned} p(t) &= realDepth(t) \times parentComplexityFactor(t) \times \textsc{gapComplexityFactor} \\ &= realDepth(t) \times \frac{\textsc{complexity}(math)}{\textsc{complexity}(RealParent(t))} \times \frac{\textsc{complexity}(math)}{\textsc{gapComplexity}(t)} \end{aligned}$$

Examples:

(a) top-level target t: $realDepth(t) = 1$, $parentComplexityFactor(t) = 1$, thus $p(t) = 1$

(b) if there's a target with low depth but small parent it gets a similar penalty to a target with a higher depth but a large parent

3.6 Map to final scale

MathML allows three values 'goodbreak', 'badbreak' and 'nobreak' thus we need to map our penalty to a scale of 3 (or more, as desired).

A standard k-means clustering algorithms will suffice.

A Examples for Algorithm 1

Example 1. Examples for **Case** `<mfenced>` from algorithm 1. We consider a permutation of the main cases.

Original | Case fire | Rewritten

Original:
```
<mfenced open="[" close="]">
  <mi>a</mi>
  <mi>b</mi>
  <mi>c</mi>
  <mi>d</mi>
</mfenced>
```
Case fire: a, d, f

Rewritten:
```
<mrow meaning="fenced">
  <mo meaning="open">[</mo>
  <mi>a</mi>
  <mo>,</mo>
  <mi>b</mi>
  <mo>,</mo>
  <mi>c</mi>
  <mo>,</mo>
  <mi>d</mi>
  <mo meaning="close">]</mo>
</mrow>
```

Original:
```
<mfenced open="[" close="]" separators="+">
  <mi>a</mi>
  <mi>b</mi>
  <mi>c</mi>
  <mi>d</mi>
</mfenced>
```
Case fire: b, d, f

Rewritten:
```
<mrow meaning="fenced">
  <mo meaning="open">[</mo>
  <mi>a</mi>
  <mo>+</mo>
  <mi>b</mi>
  <mo>+</mo>
  <mi>c</mi>
  <mo>+</mo>
  <mi>d</mi>
  <mo meaning="close">]</mo>
</mrow>
```

Original:
```
<mfenced open="[" close="]" separators="-+">
  <mi>a</mi>
  <mi>b</mi>
  <mi>c</mi>
  <mi>d</mi>
</mfenced>
```
Case fire: c, d, f

Rewritten:
```
<mrow meaning="fenced">
  <mo meaning="open">[</mo>
  <mi>a</mi>
  <mo></mo>
  <mi>b</mi>
  <mo>+</mo>
  <mi>c</mi>
  <mo>+</mo>
  <mi>d</mi>
  <mo meaning="close">]</mo>
</mrow>
```

Original:
```
<mfenced open="[" close="]" separators="">
  <mi>a</mi>
  <mi>b</mi>
  <mi>c</mi>
  <mi>d</mi>
</mfenced>
```
Case fire: d, f

Rewritten:
```
<mrow meaning="fenced">
  <mo meaning="open">[</mo>
  <mi>a</mi>
  <mi>b</mi>
  <mi>c</mi>
  <mi>d</mi>
  <mo meaning="close">]</mo>
</mrow>
```

Note, that here none of the cases in the initial `if-then-else` fire.

```
<mfenced>
  <mi>a</mi>
  <mi>b</mi>
  <mi>c</mi>
  <mi>d</mi>
</mfenced>
```

a, e, g

```
<mrow meaning="fenced">
  <mo meaning="open">(</mo>
  <mi>a</mi>
  <mo>,</mo>
  <mi>b</mi>
  <mo>,</mo>
  <mi>c</mi>
  <mo>,</mo>
  <mi>d</mi>
  <mo meaning="close">)</mo>
</mrow>
```

```
<mfenced separators="+">
  <mi>a</mi>
  <mi>b</mi>
  <mi>c</mi>
  <mi>d</mi>
</mfenced>
```

b, e, g

```
<mrow meaning="fenced">
  <mo meaning="open">(</mo>
  <mi>a</mi>
  <mo>+</mo>
  <mi>b</mi>
  <mo>+</mo>
  <mi>c</mi>
  <mo>+</mo>
  <mi>d</mi>
  <mo meaning="close">)</mo>
</mrow>
```

```
<mfenced separators="-+">
  <mi>a</mi>
  <mi>b</mi>
  <mi>c</mi>
  <mi>d</mi>
</mfenced>
```

c, e, g

```
<mrow meaning="fenced">
  <mo meaning="open">(</mo>
  <mi>a</mi>
  <mo></mo>
  <mi>b</mi>
  <mo>+</mo>
  <mi>c</mi>
  <mo>+</mo>
  <mi>d</mi>
  <mo meaning="close">)</mo>
</mrow>
```

```
<mfenced separators="">
  <mi>a</mi>
  <mi>b</mi>
  <mi>c</mi>
  <mi>d</mi>
</mfenced>
```

e, g

```
<mrow meaning="fenced">
  <mo meaning="open">(</mo>
  <mi>a</mi>
  <mi>b</mi>
  <mi>c</mi>
  <mi>d</mi>
  <mo meaning="close">)</mo>
</mrow>
```

Example 2. Example for **Case** `<mfrac>` from algorithm 1:

```
<math>
  <mo>(</mo>
  <mfrac linethickness="0pt">
    <mi>n</mi>
    <mi>k</mi>
  </mfrac>
  <mo>)</mo>
</math>
```

```
<math>
  <mo>(</mo>
  <mtable>
    <mtr>
      <mtd>
        <mi>n</mi>
      </mtd>
    </mtr>
    <mtr>
      <mtd>
        <mi>k</mi>
      </mtd>
    </mtr>
  </mtable>
  <mo>)</mo>
</math>
```

Example 3. Example for the `<mrow>`cases from algorithm 1:

```
<math>
  <mrow>
  <mi>a</mi>
  <mo>+</mo>
  <mi>b</mi>
  </mrow>
  <mrow>
  <mo>+</mo>
  <mi>c</mi>
  <mo>+</mo>
  <mi>d</mi>
  </mrow>
</math>
```

```
<math>
  <mrow>
  <mi>a</mi>
  <mo>+</mo>
  <mi>b</mi>
  <mo>+</mo>
  <mi>c</mi>
  <mo>+</mo>
  <mi>d</mi>
  </mrow>
</math>
```

```
<math>
  <mrow>
  <mi>a</mi>
  <mo>+</mo>
  <mi>b</mi>
  <mo>+</mo>
  <mi>c</mi>
  <mo>+</mo>
  <mi>d</mi>
  </mrow>
</math>
```

```
<math>
  <mi>a</mi>
  <mo>+</mo>
  <mi>b</mi>
  <mo>+</mo>
  <mi>c</mi>
  <mo>+</mo>
  <mi>d</mi>
</math>
```

Example 4. Examples for the `<mspace>`case from algorithm 1:

`<mo> </mo>`	Case a	`<mspace width="1em"/>`
`<mo>  </mo>`	Case a	`<mspace width="1.25em"/>`

```
<mrow>
  ...
  <mi> abc </mi>
  ...
</mrow>
```

Case b

```
<mrow>
  ....
  <mspace width="1em"/>
  <mi>abc </mi>
  ...
</mrow>
```

```
<mrow>
  ....
  <mspace width="1em"/>
  <mi>abc </mi>
  ...
</mrow>
```

Case d

```
<mrow>
  ....
  <mspace width="1em"/>
  <mi>abc</mi>
  <mspace width=".25em"/>
  ...
</mrow>
```

```
<msup>
  ...
  <mi> abc</mi>
</msup>
```

Case c

```
<msup>
  ....
  <mrow>
    <mspace width="1em"/>
    <mi>abc</mi>
  </mrow>
</msup>
```

```
<msup>
  ...
  <mi>abc </mi>
</msup>
```

Case e

```
<msup>
  ....
  <mrow>
    <mi>abc</mi>
    <mspace width=".25em"/>
  </mrow>
</msup>
```

```
<mrow>
  ...
  <mi> ab cd </mi>
  ...
</mrow>
```

Case b + d

```
<mrow>
  ...
  <mspace width="1em"/>
  <mi>ab cd</mi>
  <mspace width=".25em"/>
  ...
</mrow>
```

B Examples for Algorithm 2

Example 5.

$$\sum_{i=1}^{n} i + \sum_{i=n+1}^{m} i = \sum_{i=1}^{m} i$$

```
<math>
  <munderover>
    <mo>&#x2211;</mo>
    <mrow>
      <mi>i</mi>
      <mo>=</mo>
      <mn>1</mn>
    </mrow>
    <mi>n</mi>
  </munderover>
  <mi>i</mi>
  <mo>+</mo>
  <munderover >
    <mo>&#x2211;</mo>
    <mrow>
      <mi>i</mi>
      <mo>=</mo>
      <mi>n</mi>
      <mo>+</mo>
      <mn>1</mn>
    </mrow>
    <mi>m</mi>
  </munderover>
  <mi>i</mi>
  <mo>=</mo>
  <munderover >
    <mo>&#x2211;</mo>
    <mrow>
      <mi>i</mi>
      <mo>=</mo>
      <mn>1</mn>
    </mrow>
    <mi>m</mi>
  </munderover>
  <mi>i</mi>
</math>
```

```
<math>
  <munderover meaning="bigop" embellished="&#x2211;">
    <mo meaning="bigop">&#x2211;</mo>
    <mrow>
      <mi>i</mi>
      <mo meaning="operator">=</mo>
      <mn>1</mn>
    </mrow>
    <mi>n</mi>
  </munderover>
  <mi>i</mi>
  <mo meaning="operator">+</mo>
  <munderover meaning="bigop" embellished="&#x2211;">
    <mo meaning="bigop">&#x2211;</mo>
    <mrow>
      <mi>i</mi>
      <mo meaning="operator">=</mo>
      <mi>n</mi>
      <mo meaning="operator">+</mo>
      <mn>1</mn>
    </mrow>
    <mi>m</mi>
  </munderover>
  <mi>i</mi>
  <mo meaning="operator">=</mo>
  <munderover meaning="bigop" embellished="&#x2211;">
    <mo meaning="bigop">&#x2211;</mo>
    <mrow>
      <mi>i</mi>
      <mo meaning="operator">=</mo>
      <mn>1</mn>
    </mrow>
    <mi>m</mi>
  </munderover>
  <mi>i</mi>
</math>
```

$$a +_2' b$$

```
<math>
  <mi>a</mi>
  <msubsup>
    <mo>+</mo>
    <mn>2</mn>
    <mo>&#x2032;</mo>
  </msubsup>
  <mi>b</mi>
</math>
```

```
<math>
  <mi>a</mi>
  <msubsup meaning="operator" embellished="+">
    <mo meaning="operator">+</mo>
    <mn>2</mn>
    <mo>&#x2032;</mo>
  </msubsup>
  <mi>b</mi>
</math>
```

Note, that the embellished operator here remains an operator (as opposed to bigops or integrals), so is still elligible for selection in algorithm 3.

Example 6.

$$\{(a,b)|a+b=a-b\}$$

```
<math>

  <mo meaning="operator">{</mo>

  <mo meaning="operator">(</mo>
  <mi>a</mi>
  <mo meaning="operator">,</mo>
  <mi>b</mi>
  <mo meaning="operator">)</mo>

  <mo meaning="operator">|</mo>
  <mi>a</mi>
  <mo meaning="operator">+</mo>
  <mi>b</mi>
  <mo meaning="operator">=</mo>
  <mi>a</mi>
  <mo meaning="operator"></mo>
  <mi>b</mi>
  <mo meaning="operator">}</mo>

</math>
```

```
<math>
 <mrow meaning="fenced">
  <mo meaning="open">{</mo>
  <mrow meaning="fenced">
   <mo meaning="open">(</mo>
   <mi>a</mi>
   <mo meaning="operator">,</mo>
   <mi>b</mi>
   <mo meaning="close">)</mo>
  </mrow>
  <mo meaning="operator">|</mo>
  <mi>a</mi>
  <mo meaning="operator">+</mo>
  <mi>b</mi>
  <mo meaning="operator">=</mo>
  <mi>a</mi>
  <mo meaning="operator"></mo>
  <mi>b</mi>
  <mo meaning="close">}</mo>
 </mrow>
</math>
```

Example 7.

$$\int_a^b f(x)\,dx = \lim_{\|\Delta x\|\to 0} \sum_{i=1}^{n} f(x_i^*)\,\Delta x_i$$

This example demonstrates four of the six cases in step 3: Big Operators, Integrals, Invisible Times, and Function Application

```
<math>

  <msubsup meaning="integral" embellished="&#x222B;">
    <mo meaning="operator">&#x222B;</mo>
    <mi>a</mi>
    <mi>b</mi>
  </msubsup>
  <mi>f</mi>

  <mo stretchy="false">(</mo>
  <mi>x</mi>
  <mo stretchy="false">)</mo>
  <mi>dx</mi>

  <mo meaning="operator">=</mo>
  <munder>
    <mo movablelimits="true" form="prefix">lim</mo>
    <mrow>
      <mrow meaning="fenced">
        <mo meaning="open" stretchy="false">&#x2016;</mo>
        <mi mathvariant="normal">&#x0394;</mi>

        <mi>x</mi>
        <mo meaning="close" stretchy="false">&#x2016;</mo>
      </mrow>
      <mo meaning="operator">&#x2192;</mo>
      <mn>0</mn>
    </mrow>
  </munder>


  <munderover meaning="bigop" embellished="&#x2211;">
    <mo meaning="operator">&#x2211;</mo>
    <mrow>
      <mi>i</mi>
      <mo meaning="operator">=</mo>
      <mn>1</mn>
    </mrow>
    <mi>n</mi>
  </munderover>
  <mi>f</mi>

  <mrow meaning="fenced">
    <mo meaning="open" stretchy="false">(</mo>
    <msubsup>
      <mi>x</mi>
      <mi>i</mi>
      <mo meaning="operator">&#x2217;</mo>
    </msubsup>
    <mo meaning="close" stretchy="false">)</mo>
  </mrow>

  <mi mathvariant="normal">&#x0394;</mi>

  <msub>
    <mi>x</mi>
    <mi>i</mi>
  </msub>

</math>
```

```
<math>
  <mrow meaning="integral">
    <msubsup meaning="integral" embellished="&#x222B;">
      <mo meaning="operator">&#x222B;</mo>
      <mi>a</mi>
      <mi>b</mi>
    </msubsup>
    <mi>f</mi>
    <mo meaning="operator">&#x2061;</mo>
    <mo stretchy="false">(</mo>
    <mi>x</mi>
    <mo stretchy="false">)</mo>
    <mi>dx</mi>
  </mrow>
  <mo meaning="operator">=</mo>
  <munder>
    <mo movablelimits="true" form="prefix">lim</mo>
    <mrow>
      <mrow meaning="fenced">
        <mo meaning="open" stretchy="false">&#x2016;</
        <mi mathvariant="normal">&#x0394;</mi>
        <mo meaning="operator">&#x2062;</mo>
        <mi>x</mi>
        <mo meaning="close" stretchy="false">&#x2016;<
      </mrow>
      <mo meaning="operator">&#x2192;</mo>
      <mn>0</mn>
    </mrow>
  </munder>
  <mo meaning="operator">&#x2062;</mo>
  <mrow meaning="bigop">
    <munderover meaning="bigop" embellished="&#x2211;">
      <mo meaning="operator">&#x2211;</mo>
      <mrow>
        <mi>i</mi>
        <mo meaning="operator">=</mo>
        <mn>1</mn>
      </mrow>
      <mi>n</mi>
    </munderover>
    <mi>f</mi>
    <mo meaning="operator">&#x2061;</mo>
    <mrow meaning="fenced">
      <mo meaning="open" stretchy="false">(</mo>
      <msubsup>
        <mi>x</mi>
        <mi>i</mi>
        <mo meaning="operator">&#x2217;</mo>
      </msubsup>
      <mo meaning="close" stretchy="false">)</mo>
    </mrow>
    <mo meaning="operator">&#x2062;</mo>
    <mi mathvariant="normal">&#x0394;</mi>
    <mo meaning="operator">&#x2062;</mo>
    <msub>
      <mi>x</mi>
      <mi>i</mi>
    </msub>
  </mrow>
</math>
```

C Example for Algorithm 3

We further elaborate example 7, marking selected operators in red.

```
<math>
  <mrow meaning="integral">
    <msubsup meaning="integral" embellished="&#x222B;">
      <mo meaning="operator">&#x222B;</mo>
      <mi>a</mi>
      <mi>b</mi>
    </msubsup>
    <mi>f</mi>
    <mo meaning="operator">&#x2061;</mo>
    <mo stretchy="false">(</mo>
    <mi>x</mi>
    <mo stretchy="false">)</mo>
    <mi>dx</mi>
  </mrow>
  <mo meaning="operator">=</mo>
  <munder>
    <mo movablelimits="true" form="prefix">lim</mo>
    <mrow>
      <mrow meaning="fenced">
        <mo meaning="open" stretchy="false">&#x2016;</mo>
        <mi mathvariant="normal">&#x0394;</mi>
        <mo meaning="operator">&#x2062;</mo>
        <mi>x</mi>
        <mo meaning="close" stretchy="false">&#x2016;</mo>
      </mrow>
      <mo meaning="operator">&#x2192;</mo>
      <mn>0</mn>
    </mrow>
  </munder>
  <mo meaning="operator">&#x2062;</mo>
  <mrow meaning="bigop">
    <munderover meaning="bigop" embellished="&#x2211;">
      <mo meaning="operator">&#x2211;</mo>
      <mrow>
        <mi>i</mi>
        <mo meaning="operator">=</mo>
        <mn>1</mn>
      </mrow>
      <mi>n</mi>
    </munderover>
    <mi>f</mi>
    <mo meaning="operator">&#x2061;</mo>
    <mrow meaning="fenced">
      <mo meaning="open" stretchy="false">(</mo>
      <msubsup>
        <mi>x</mi>
        <mi>i</mi>
        <mo meaning="operator">&#x2217;</mo>
      </msubsup>
      <mo meaning="close" stretchy="false">)</mo>
    </mrow>
    <mo meaning="operator">&#x2062;</mo>
    <mi mathvariant="normal">&#x0394;</mi>
    <mo meaning="operator">&#x2062;</mo>
    <msub>
      <mi>x</mi>
      <mi>i</mi>
    </msub>
  </mrow>
</math>
```

D Example for Procedure in Section 3

As a sample we take equation 12 of the second sample set. See also the `sample2_12.html` file for MathML:

$$\bar{u}_{Fy}^{P} = \bar{f}^{P}(p) \int_0^\infty \left\{ -\eta_F \frac{2}{\Delta} \left[\lambda \left(k^2 - \eta_P^2\right) - 2\mu\eta_P^2\right] \left(k^2 - \eta_S^2\right) \times \{\exp\{-[(y+H)\eta_F + (h-H)\eta_P]\} + \exp\{-[(H-y)\eta_F + (h-H)\eta_P]\}\} \right\} \cos(kx)\, dk$$

We present three versions of the expression:

1. After canonicalisation and chunking,
2. With selected operators marked in red,
3. With all linebreaking measures computed.

Note that the measures have been computed by hand and might not be fully accurate.

D.1 Equation after Canonicalisation and Chunking

```
<math>
  <msubsup>
    <mrow>
      <mover accent="true">
        <mi>u</mi>
        <mo meaning="operator">&OverBar;</mo>
      </mover>
    </mrow>
    <mrow>
      <mi>F</mi>
      <mo meaning="operator" new>&InvisibleTimes;</mo>
      <mi>y</mi>
    </mrow>
    <mi>P</mi>
  </msubsup>
  <mo meaning="operator">&equals;</mo>
  <msup>
    <mrow>
      <mover accent="true">
        <mi>f</mi>
        <mo meaning="operator">&OverBar;</mo>
      </mover>
    </mrow>
    <mi>P</mi>
  </msup>
  <mo meaning="operator" new>&InvisibleTimes;</mo>
  <mrow meaning="fenced">
    <mo meaning="open">(</mo>
    <mi>p</mi>
    <mo meaning="close">)</mo>
  </mrow>
  <mo meaning="operator" new>&InvisibleTimes;</mo>
  <mrow meaning="integral">
    <msubsup meaning="integral" embellished="&int;">
      <mo meaning="operator">&int;</mo>
      <mn>0</mn>
      <mi>&infin;</mi>
    </msubsup>
    <mrow>
      <mrow meaning="fenced">
        <mo meaning="open">{</mo>
        <mrow>
          <mo meaning="prefix">&minus;</mo>
          <msub>
            <mi>&eta;</mi>
            <mi>F</mi>
          </msub>
          <mfrac>
            <mn>2</mn>
            <mo meaning="operator">&Delta;</mo>
          </mfrac>
          <mo meaning="operator" new>&InvisibleTimes;</mo>
          <mrow meaning="fenced">
            <mo meaning="open">[</mo>
            <mrow>
              <mi>&lambda;</mi>
```

```
    <mo meaning="operator" new>&InvisibleTimes;</mo>
    <mrow meaning="fenced">
      <mo meaning="open">(</mo>
      <mrow>
        <msup>
          <mi>k</mi>
          <mn>2</mn>
        </msup>
        <mo meaning="operator">&minus;</mo>
        <msubsup>
          <mi>&eta;</mi>
          <mi>P</mi>
          <mn>2</mn>
        </msubsup>
      </mrow>
      <mo meaning="close">)</mo>
    </mrow>
    <mo meaning="operator">&minus;</mo>
    <mn>2</mn>
    <mo meaning="operator" new>&InvisibleTimes;</mo>
    <mi>&mu;</mi>
    <mo meaning="operator" new>&InvisibleTimes;</mo>
    <msubsup>
      <mi>&eta;</mi>
      <mi>P</mi>
      <mn>2</mn>
    </msubsup>
  </mrow>
  <mo meaning="close">]</mo>
</mrow>
<mo meaning="operator" new>&InvisibleTimes;</mo>
<mrow meaning="fenced">
  <mo meaning="open">(</mo>
  <mrow>
    <msup>
      <mi>k</mi>
      <mn>2</mn>
    </msup>
    <mo meaning="operator">&minus;</mo>
    <msubsup>
      <mi>&eta;</mi>
      <mi>S</mi>
      <mn>2</mn>
    </msubsup>
  </mrow>
  <mo meaning="close">)</mo>
</mrow>
<mo meaning="operator">&times;</mo>
<mrow meaning="fenced">
  <mo meaning="open">{</mo>
  <mrow>
    <mi>exp</mi>
    <mo meaning="operator" new>&ApplyFunction;</mo>
    <mrow meaning="fenced">
      <mo meaning="open">{</mo>
      <mrow>
        <mo meaning="prefix">&minus;</mo>
        <mrow meaning="fenced">
          <mo meaning="open">[</mo>
          <mrow>
            <mrow meaning="fenced">
              <mo meaning="open">(</mo>
              <mrow>
                <mi>y</mi>
                <mo meaning="operator">+</mo>
                <mi>H</mi>
              </mrow>
              <mo meaning="close">)</mo>
            </mrow>
            <mo meaning="operator" new>&InvisibleTimes;</mo>
            <msub>
              <mi>&eta;</mi>
              <mi>F</mi>
            </msub>
            <mo meaning="operator">+</mo>
            <mrow meaning="fenced">
              <mo meaning="open">(</mo>
              <mrow>
                <mi>h</mi>
                <mo meaning="operator">&minus;</mo>
                <mi>H</mi>
```

```
                  </mrow>
                  <mo meaning="close">)</mo>
                </mrow>
                <mo meaning="operator" new>&InvisibleTimes;</mo>
                <msub>
                  <mi>&eta;</mi>
                  <mi>P</mi>
                </msub>
              </mrow>
              <mo meaning="close">]</mo>
            </mrow>
          </mrow>
          <mo meaning="close">}</mo>
        </mrow>
        <mo meaning="operator">+</mo>
        <mi>exp</mi>
        <mo meaning="operator" new>&ApplyFunction;</mo>
        <mrow meaning="fenced">
          <mo meaning="open">{</mo>
          <mrow>
            <mo meaning="prefix">&minus;</mo>
            <mrow meaning="fenced">
              <mo meaning="open">[</mo>
              <mrow>
                <mrow meaning="fenced">
                  <mo meaning="open">(</mo>
                  <mrow>
                    <mi>H</mi>
                    <mo meaning="operator">&minus;</mo>
                    <mi>y</mi>
                  </mrow>
                  <mo meaning="close">)</mo>
                </mrow>
                <mo meaning="operator" new>&InvisibleTimes;</mo>
                <msub>
                  <mi>&eta;</mi>
                  <mi>F</mi>
                </msub>
                <mo meaning="operator">+</mo>
                <mrow meaning="fenced">
                  <mo meaning="open">(</mo>
                  <mrow>
                    <mi>h</mi>
                    <mo meaning="operator">&minus;</mo>
                    <mi>H</mi>
                  </mrow>
                  <mo meaning="close">)</mo>
                </mrow>
                <mo meaning="operator" new>&InvisibleTimes;</mo>
                <msub>
                  <mi>&eta;</mi>
                  <mi>P</mi>
                </msub>
              </mrow>
              <mo meaning="close">]</mo>
            </mrow>
          </mrow>
          <mo meaning="close">}</mo>
        </mrow>
      </mrow>
      <mo meaning="close">}</mo>
    </mrow>
  </mrow>
  <mo meaning="close">}</mo>
</mrow>
<mo meaning="operator" new>&InvisibleTimes;</mo>
<mi>cos</mi>
<mo meaning="operator" new>&ApplyFunction;</mo>
<mrow meaning="fenced">
  <mo meaning="open">(</mo>
  <mrow>
    <mi>k</mi>
    <mo meaning="operator" new>&InvisibleTimes;</mo>
    <mi>x</mi>
  </mrow>
  <mo meaning="close">)</mo>
</mrow>
<mo meaning="operator" new>&InvisibleTimes;</mo>
<mi>d</mi>
<mo meaning="operator" new>&InvisibleTimes;</mo>
<mi>k</mi>
```

```
    </mrow>
  </mrow>
</math>
```

D.2 Equation with Selected Operators

```
<math>
  <msubsup>
    <mrow>
      <mover accent="true">
        <mi>u</mi>
        <mo meaning="operator">&OverBar;</mo>
      </mover>
    </mrow>
    <mrow>
      <mi>F</mi>
      <mo meaning="operator" new>&InvisibleTimes;</mo>
      <mi>y</mi>
    </mrow>
    <mi>P</mi>
  </msubsup>
  <mo meaning="operator">&equals;</mo>
  <msup>
    <mrow>
      <mover accent="true">
        <mi>f</mi>
        <mo meaning="operator">&OverBar;</mo>
      </mover>
    </mrow>
    <mi>P</mi>
  </msup>
  <mo meaning="operator" new>&InvisibleTimes;</mo>
  <mrow meaning="fenced">
    <mo meaning="open">(</mo>
    <mi>p</mi>
    <mo meaning="close">)</mo>
  </mrow>
  <mo meaning="operator" new>&InvisibleTimes;</mo>
  <mrow meaning="integral">
    <msubsup embellished="&int;">
      <mo meaning="operator">&int;</mo>
      <mn>0</mn>
      <mi>&infin;</mi>
    </msubsup>
    <mrow>
      <mrow meaning="fenced">
        <mo meaning="open">{</mo>
        <mrow>
          <mo meaning="prefix">&minus;</mo>
          <msub>
            <mi>&eta;</mi>
            <mi>F</mi>
          </msub>
          <mfrac>
            <mn>2</mn>
            <mo meaning="operator">&Delta;</mo>
          </mfrac>
          <mo meaning="operator" new>&InvisibleTimes;</mo>
          <mrow meaning="fenced">
            <mo meaning="open">[</mo>
            <mrow>
              <mi>&lambda;</mi>
              <mo meaning="operator" new>&InvisibleTimes;</mo>
              <mrow meaning="fenced">
                <mo meaning="open">(</mo>
                <mrow>
                  <msup>
                    <mi>k</mi>
                    <mn>2</mn>
                  </msup>
                  <mo meaning="operator">&minus;</mo>
                  <msubsup>
                    <mi>&eta;</mi>
                    <mi>P</mi>
                    <mn>2</mn>
                  </msubsup>
                </mrow>
                <mo meaning="close">)</mo>
              </mrow>
              <mo meaning="operator">&minus;</mo>
```

```
    <mn>2</mn>
    <mo meaning="operator" new>&InvisibleTimes;</mo>
    <mi>&mu;</mi>
    <mo meaning="operator" new>&InvisibleTimes;</mo>
    <msubsup>
      <mi>&eta;</mi>
      <mi>P</mi>
      <mn>2</mn>
    </msubsup>
  </mrow>
  <mo meaning="close">]</mo>
</mrow>
<mo meaning="operator" new>&InvisibleTimes;</mo>
<mrow meaning="fenced">
  <mo meaning="open">(</mo>
  <mrow>
    <msup>
      <mi>k</mi>
      <mn>2</mn>
    </msup>
    <mo meaning="operator">&minus;</mo>
    <msubsup>
      <mi>&eta;</mi>
      <mi>S</mi>
      <mn>2</mn>
    </msubsup>
  </mrow>
  <mo meaning="close">)</mo>
</mrow>
<mo meaning="operator">&times;</mo>
<mrow meaning="fenced">
  <mo meaning="open">{</mo>
  <mrow>
    <mi>exp</mi>
    <mo meaning="operator" new>&ApplyFunction;</mo>
    <mrow meaning="fenced">
      <mo meaning="open">{</mo>
      <mrow>
        <mo meaning="prefix">&minus;</mo>
        <mrow meaning="fenced">
          <mo meaning="open">[</mo>
          <mrow>
            <mrow meaning="fenced">
              <mo meaning="open">(</mo>
              <mrow>
                <mi>y</mi>
                <mo meaning="operator">+</mo>
                <mi>H</mi>
              </mrow>
              <mo meaning="close">)</mo>
            </mrow>
            <mo meaning="operator" new>&InvisibleTimes;</mo>
            <msub>
              <mi>&eta;</mi>
              <mi>F</mi>
            </msub>
            <mo meaning="operator">+</mo>
            <mrow meaning="fenced">
              <mo meaning="open">(</mo>
              <mrow>
                <mi>h</mi>
                <mo meaning="operator">&minus;</mo>
                <mi>H</mi>
              </mrow>
              <mo meaning="close">)</mo>
            </mrow>
            <mo meaning="operator" new>&InvisibleTimes;</mo>
            <msub>
              <mi>&eta;</mi>
              <mi>P</mi>
            </msub>
          </mrow>
          <mo meaning="close">]</mo>
        </mrow>
      </mrow>
      <mo meaning="close">}</mo>
    </mrow>
    <mo meaning="operator">+</mo>
    <mi>exp</mi>
    <mo meaning="operator" new>&ApplyFunction;</mo>
    <mrow meaning="fenced">
```

```
                <mo meaning="open">{</mo>
                <mrow>
                  <mo meaning="prefix">&minus;</mo>
                  <mrow meaning="fenced">
                    <mo meaning="open">[</mo>
                    <mrow>
                      <mrow meaning="fenced">
                        <mo meaning="open">(</mo>
                        <mrow>
                          <mi>H</mi>
                          <mo meaning="operator">&minus;</mo>
                          <mi>y</mi>
                        </mrow>
                        <mo meaning="close">)</mo>
                      </mrow>
                      <mo meaning="operator" new>&InvisibleTimes;</mo>
                      <msub>
                        <mi>&eta;</mi>
                        <mi>F</mi>
                      </msub>
                      <mo meaning="operator">+</mo>
                      <mrow meaning="fenced">
                        <mo meaning="open">(</mo>
                        <mrow>
                          <mi>h</mi>
                          <mo meaning="operator">&minus;</mo>
                          <mi>H</mi>
                        </mrow>
                        <mo meaning="close">)</mo>
                      </mrow>
                      <mo meaning="operator" new>&InvisibleTimes;</mo>
                      <msub>
                        <mi>&eta;</mi>
                        <mi>P</mi>
                      </msub>
                    </mrow>
                    <mo meaning="close">]</mo>
                  </mrow>
                </mrow>
                <mo meaning="close">}</mo>
              </mrow>
            </mrow>
            <mo meaning="close">}</mo>
          </mrow>
        </mrow>
        <mo meaning="close">}</mo>
      </mrow>
      <mo meaning="operator" new>&InvisibleTimes;</mo>
      <mi>cos</mi>
      <mo meaning="operator" new>&ApplyFunction;</mo>
      <mrow meaning="fenced">
        <mo meaning="open">(</mo>
        <mrow>
          <mi>k</mi>
          <mo meaning="operator" new>&InvisibleTimes;</mo>
          <mi>x</mi>
        </mrow>
        <mo meaning="close">)</mo>
      </mrow>
      <mo meaning="operator" new>&InvisibleTimes;</mo>
      <mi>d</mi>
      <mo meaning="operator" new>&InvisibleTimes;</mo>
      <mi>k</mi>
    </mrow>
  </mrow>
</math>
```

D.3 Equation with Linebreaking Measures

```
<math display="block" complexity="211.24999999999997">
  <msubsup complexity="5.97">
    <mrow complexity="3.5">
      <mover accent="true" complexity="2.5">
        <mi complexity="1.5">u</mi>
        <mo meaning="operator" complexity="1.5">&oline;</mo>
      </mover>
    </mrow>
    <mrow complexity="5.5">
      <mi complexity="1.5">F</mi>
      <mo meaning="operator" new="" complexity="1.5">&#x2062;</mo>
      <mi complexity="1.5">y</mi>
    </mrow>
    <mi complexity="1.5">P</mi>
  </msubsup>
  <mo meaning="operator" complexity="1.5" gapcomplexity="5.97" penalty="35.385259631">=</mo>
  <msup complexity="5.5649999999999995">
    <mrow complexity="3.5">
      <mover accent="true" complexity="2.5">
        <mi complexity="1.5">f</mi>
        <mo meaning="operator" complexity="1.5">&oline;</mo>
      </mover>
    </mrow>
    <mi complexity="1.5">P</mi>
  </msup>
  <mo meaning="operator" new="" complexity="1.5" gapcomplexity="7.0649999999999995" penalty="29.900920028">&#x2062;</mo>
  <mrow meaning="fenced" complexity="5.5">
    <mo meaning="open" complexity="1.5">(</mo>
    <mi complexity="1.5">p</mi>
    <mo meaning="close" complexity="1.5">)</mo>
  </mrow>
  <mo meaning="operator" new="" complexity="1.5" gapcomplexity="7" penalty="30.178571429">&#x2062;</mo>
  <mrow meaning="integral" complexity="188.71499999999997" parentcomplexityfactor="1.119412871">
    <msubsup embellished="&int;" complexity="3.565">
      <mo meaning="operator" complexity="1.5">&int;</mo>
      <mn complexity="1.5">0</mn>
      <mi complexity="1.5">&infin;</mi>
    </msubsup>
    <mrow complexity="184.14999999999998">
      <mrow meaning="fenced" complexity="162.14999999999998">
        <mo meaning="open" complexity="1.5">{</mo>
        <mrow complexity="158.14999999999998">
          <mo meaning="prefix" complexity="1.5">&minus;</mo>
          <msub complexity="3.565">
            <mi complexity="1.5">&eta;</mi>
            <mi complexity="1.5">F</mi>
          </msub>
          <mfrac complexity="2.5">
            <mn complexity="1.5">2</mn>
            <mo meaning="operator" complexity="1.5">&Delta;</mo>
          </mfrac>
          <mo meaning="operator" new="" complexity="1.5" gapcomplexity="14.129999999999999" realdepth="2" depth="3" penalty="33.471474742">&
          <mrow meaning="fenced" complexity="32.695">
            <mo meaning="open" complexity="1.5">[</mo>
            <mrow complexity="28.695">
              <mi complexity="1.5">&lambda;</mi>
              <mo meaning="operator" new="" complexity="1.5" gapcomplexity="4.5" realdepth="2" depth="4" penalty="105.10043069">&#x2062;</mo
              <mrow meaning="fenced" complexity="13.629999999999999">
                <mo meaning="open" complexity="1.5">(</mo>
                <mrow complexity="9.629999999999999">
                  <msup complexity="3.565">
                    <mi complexity="1.5">k</mi>
                    <mn complexity="1.5">2</mn>
                  </msup>
                  <mo meaning="operator" complexity="1.5" gapcomplexity="6.5649999999999995" realdepth="2" depth="5" penalty="72.041422408">&
                  <msubsup complexity="3.565">
                    <mi complexity="1.5">&eta;</mi>
                    <mi complexity="1.5">P</mi>
                    <mn complexity="1.5">2</mn>
                  </msubsup>
                </mrow>
                <mo meaning="close" complexity="1.5">)</mo>
              </mrow>
              <mo meaning="operator" complexity="1.5" gapcomplexity="15.13" realdepth="2" depth="4" penalty="31.259216002">&minus;</mo>
              <mn complexity="1.5">2</mn>
              <mo meaning="operator" new="" complexity="1.5" gapcomplexity="3" realdepth="2" depth="4" penalty="157.650646036">&#x2062;</mo
              <mi complexity="1.5">&mu;</mi>
              <mo meaning="operator" new="" complexity="1.5" gapcomplexity="3" realdepth="2" depth="4" penalty="157.650646036">&#x2062;</mo
```

```
    <msubsup complexity="3.565">
      <mi complexity="1.5">&eta;</mi>
      <mi complexity="1.5">P</mi>
      <mn complexity="1.5">2</mn>
    </msubsup>
  </mrow>
  <mo meaning="close" complexity="1.5">]</mo>
</mrow>
<mo meaning="operator" new="" complexity="1.5" gapcomplexity="34.195" realdepth="2" depth="3" penalty="13.831026118">&#x2062;</mo
<mrow meaning="fenced" complexity="13.629999999999999">
  <mo meaning="open" complexity="1.5">(</mo>
  <mrow complexity="9.629999999999999">
    <msup complexity="3.565">
      <mi complexity="1.5">k</mi>
      <mn complexity="1.5">2</mn>
    </msup>
    <mo meaning="operator" complexity="1.5" gapcomplexity="6.5649999999999995" realdepth="2" depth="4" penalty="72.041422408">&minu
    <msubsup complexity="3.565">
      <mi complexity="1.5">&eta;</mi>
      <mi complexity="1.5">S</mi>
      <mn complexity="1.5">2</mn>
    </msubsup>
  </mrow>
  <mo meaning="close" complexity="1.5">)</mo>
</mrow>
<mo meaning="operator" complexity="1.5" gapcomplexity="15.13" realdepth="2" depth="3" penalty="31.259216002">&times;</mo>
<mrow meaning="fenced" complexity="98.75999999999999">
  <mo meaning="open" complexity="1.5">{</mo>
  <mrow complexity="94.75999999999999">
    <mi complexity="2.5">exp</mi>
    <mo meaning="operator" new="" complexity="1.5" gapcomplexity="5.5" realdepth="2" depth="4" penalty="85.991261474">&#x2061;</mo
    <mrow meaning="fenced" complexity="42.129999999999995">
      <mo meaning="open" complexity="1.5">{</mo>
      <mrow complexity="38.129999999999995">
        <mo meaning="prefix" complexity="1.5">&minus;</mo>
        <mrow meaning="fenced" complexity="35.629999999999995">
          <mo meaning="open" complexity="1.5">[</mo>
          <mrow complexity="31.63">
            <mrow meaning="fenced" complexity="9.5">
              <mo meaning="open" complexity="1.5">(</mo>
              <mrow complexity="5.5">
                <mi complexity="1.5">y</mi>
                <mo meaning="operator" complexity="1.5" gapcomplexity="9" realdepth="2" depth="7" penalty="52.550215346">+</mo>
                <mi complexity="1.5">H</mi>
              </mrow>
              <mo meaning="close" complexity="1.5">)</mo>
            </mrow>
            <mo meaning="operator" new="" complexity="1.5" gapcomplexity="13.5" realdepth="2" depth="6" penalty="35.033476896">&#x
            <msub complexity="3.565">
              <mi complexity="1.5">&eta;</mi>
              <mi complexity="1.5">F</mi>
            </msub>
            <mo meaning="operator" complexity="1.5" gapcomplexity="5.0649999999999995" realdepth="2" depth="6" penalty="93.37649321">
            <mrow meaning="fenced" complexity="9.5">
              <mo meaning="open" complexity="1.5">(</mo>
              <mrow complexity="5.5">
                <mi complexity="1.5">h</mi>
                <mo meaning="operator" complexity="1.5" gapcomplexity="4.5" realdepth="2" depth="7" penalty="105.10043069">&minus;<
                <mi complexity="1.5">H</mi>
              </mrow>
              <mo meaning="close" complexity="1.5">)</mo>
            </mrow>
            <mo meaning="operator" new="" complexity="1.5" gapcomplexity="11" realdepth="2" depth="6" penalty="42.995630738">&#x20
            <msub complexity="3.565">
              <mi complexity="1.5">&eta;</mi>
              <mi complexity="1.5">P</mi>
            </msub>
          </mrow>
          <mo meaning="close" complexity="1.5">]</mo>
        </mrow>
      </mrow>
      <mo meaning="close" complexity="1.5">}</mo>
    </mrow>
    <mo meaning="operator" complexity="1.5" gapcomplexity="43.63" realdepth="2" depth="4" penalty="10.840062758">+</mo>
    <mi complexity="2.5">exp</mi>
    <mo meaning="operator" new="" complexity="1.5" gapcomplexity="4" realdepth="2" depth="4" penalty="118.237984526">&#x2061;</mo>
    <mrow meaning="fenced" complexity="42.129999999999995">
      <mo meaning="open" complexity="1.5">{</mo>
      <mrow complexity="38.129999999999995">
        <mo meaning="prefix" complexity="1.5">&minus;</mo>
        <mrow meaning="fenced" complexity="35.629999999999995">
```

```
                    <mo meaning="open" complexity="1.5">[</mo>
                    <mrow complexity="31.63">
                      <mrow meaning="fenced" complexity="9.5">
                        <mo meaning="open" complexity="1.5">(</mo>
                        <mrow complexity="5.5">
                          <mi complexity="1.5">H</mi>
                          <mo meaning="operator" complexity="1.5" gapcomplexity="9" realdepth="2" depth="7" penalty="52.550215346">&minus;</m
                          <mi complexity="1.5">y</mi>
                        </mrow>
                        <mo meaning="close" complexity="1.5">)</mo>
                      </mrow>
                      <mo meaning="operator" new="" complexity="1.5" gapcomplexity="13.5" realdepth="2" depth="6" penalty="35.033476896">&#x
                      <msub complexity="3.565">
                        <mi complexity="1.5">&eta;</mi>
                        <mi complexity="1.5">F</mi>
                      </msub>
                      <mo meaning="operator" complexity="1.5" gapcomplexity="5.0649999999999995" realdepth="2" depth="6" penalty="93.37649321">
                      <mrow meaning="fenced" complexity="9.5">
                        <mo meaning="open" complexity="1.5">(</mo>
                        <mrow complexity="5.5">
                          <mi complexity="1.5">h</mi>
                          <mo meaning="operator" complexity="1.5" gapcomplexity="4.5" realdepth="2" depth="7" penalty="105.10043069">&minus;<
                          <mi complexity="1.5">H</mi>
                        </mrow>
                        <mo meaning="close" complexity="1.5">)</mo>
                      </mrow>
                      <mo meaning="operator" new="" complexity="1.5" gapcomplexity="11" realdepth="2" depth="6" penalty="42.995630738">&#x20
                      <msub complexity="3.565">
                        <mi complexity="1.5">&eta;</mi>
                        <mi complexity="1.5">P</mi>
                      </msub>
                    </mrow>
                    <mo meaning="close" complexity="1.5">]</mo>
                  </mrow>
                </mrow>
                <mo meaning="close" complexity="1.5">}</mo>
              </mrow>
            </mrow>
            <mo meaning="close" complexity="1.5">}</mo>
          </mrow>
        </mrow>
        <mo meaning="close" complexity="1.5">}</mo>
      </mrow>
      <mo meaning="operator" new="" complexity="1.5" gapcomplexity="153.085" realdepth="2" depth="3" penalty="3.089472764">&#x2062;</mo>
      <mi complexity="2.5">cos</mi>
      <mo meaning="operator" new="" complexity="1.5" gapcomplexity="4" realdepth="2" depth="3" penalty="118.237984526">&#x2061;</mo>
      <mrow meaning="fenced" complexity="9.5">
        <mo meaning="open" complexity="1.5">(</mo>
        <mrow complexity="5.5">
          <mi complexity="1.5">k</mi>
          <mo meaning="operator" new="" complexity="1.5" gapcomplexity="4.5" realdepth="2" depth="4" penalty="105.10043069">&#x2062;</mo>
          <mi complexity="1.5">x</mi>
        </mrow>
        <mo meaning="close" complexity="1.5">)</mo>
      </mrow>
      <mo meaning="operator" new="" complexity="1.5" gapcomplexity="11" realdepth="2" depth="3" penalty="42.995630738">&#x2062;</mo>
      <mi complexity="1.5">d</mi>
      <mo meaning="operator" new="" complexity="1.5" gapcomplexity="3" realdepth="2" depth="3" penalty="157.650646036">&#x2062;</mo>
      <mi complexity="1.5">k</mi>
    </mrow>
  </mrow>
</math>
```

D.4 Discussion

$$\bar{u}^P_{Fy} = \bar{f}^P(p) \int_0^\infty \left\{ -\eta_F \frac{2}{\Delta} \left[\lambda \left(k^2 - \eta_P^2 \right) - 2\mu\eta_P^2 \right] \left(k^2 - \eta_S^2 \right) \times \left\{ \exp \left\{ - \left[(y + H) \eta_F + (h - H) \eta_P \right] \right\} + \exp \left\{ - \left[(H - y) \eta_F + (h - H) \eta_P \right] \right\} \right\} \right\} \cos (kx) \, dk$$

Let's first look at a sorted list of 30 penalties in the above expression:

3.089472764	35.385259631	93.376493210
10.840062758	42.995630738	105.100430690
13.831026118	42.995630738	105.100430690
29.900920028	42.995630738	105.100430690
30.178571429	52.550215346	105.100430690
31.259216002	52.550215346	118.237984526
31.259216002	72.041422408	118.237984526
33.471474742	72.041422408	157.650646036
35.033476896	85.991261474	157.650646036
35.033476896	93.376493210	157.650646036

We note that set contains one clear outlier at the maximum value. We observe that the maximum value occurs only at implicit multiplications, in particular twice at, $2\mu\eta_P^2$ and at dk. These are places, where we really would not want to break.

Ignoring the outlier and then applying a 3-clustering (i.e., a clustering corresponding to a "good, bad, no" linebreak scenario), we get three intervals of penalties:

$$[3.089472764, 13.831026118]\,; \qquad [29.900920028, 52.550215346]\,; \qquad [72.041422408, 118.237984526]$$

We observe what breaking at all elements with penalties in the first cluster would do:

$$\begin{aligned} \bar{u}^P_{Fy} = \bar{f}^P(p) \int_0^\infty & \left\{ -\eta_F \frac{2}{\Delta} \left[\lambda \left(k^2 - \eta_P^2 \right) - 2\mu\eta_P^2 \right] \right. \\ & \left(k^2 - \eta_S^2 \right) \times \left\{ \exp \left\{ - \left[(y + H) \eta_F + (h - H) \eta_P \right] \right\} + \right. \\ & \left. \left. \exp \left\{ - \left[(H - y) \eta_F + (h - H) \eta_P \right] \right\} \right\} \right\} \\ & \cos (kx) \, dk \end{aligned}$$

The example demonstrates the influence of the parent complexity on the overall penalty, which explains why the first linebreak is at the implicit times between the fenced expressions, with penalty 13.831026118, rather than at the following explicit time $\times$, which has a penalty of 31.259216002. The idea of the parent complexity is to allow for better balanced splits in the expression and not only taking nesting depth into account.

This is also the main reason for the low penalty on the implicit times before the cos function. Intuitively, it is the least nested operator, which splits the expression *evenly*. Here evenly is relative, meaning effectively, *more evenly than splitting at the equality sign*.