ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20160520095624
Preliminary communication

Dynamic Mathematical Layout in E-Books

Maja TURČIĆ, Klaudio PAP

Abstract: The development of e-books technology, with the emphasis on content readability and accessibility, raises the issue of mathematical layout optimisation. Although typographic formatting is successfully conducted by incorporating complex mathematical content in the form of bitmap images, such content is not accessible to text-to-speech technologies nor does it enable readability on small screens. For the purpose of creating dynamic and accessible mathematical layout, various e-book formats are analysed and EPUB3 format is presented as the only format meeting the requirements of openness, fluidity, support, accessibility and semantically correct mathematical type. The paper defines the realisation of EPUB e-books with emphasis on the optimal mathematical type using the Mathematical Markup Language. The problems of display, typographic formatting and dynamic layout are resolved through JavaScript functions depending on the rendering technologies of various e-readers on different platforms.

Keywords: e-book; EPUB; mathematical layout; Mathematical Markup Language

1 INTRODUCTION

The mathematical layout has always been one of the complex tasks of graphic designers due to its own typographic rules and different default glyphs. In the digital context, LaTeX has been for a long series of years the system most designers rely on when designing mathematical content, especially for scientific and professional publication because it enables professional formatting of a static page [1]. However, at designing fluid layout and semantically correct mathematical content in e-books, designers face the problem of choosing an optimal e-book format and the means of formatting and editing mathematical content. The problem becomes even more complex if there is a need to present mathematical formulas in more than one line in relation to the platform (personal computer, e-reader, tablet/mobile phone) and the screen size at which the content is to be presented, as well as specific typographic rules ensuring readability.

The optimal e-book format includes: openness implying freedom of use and implementation, searchability, adjustment of content to all screen sizes and resolutions, content accessibility to people with disabilities (especially partially sighted and blind people) and the support of all needed glyphs.

Table 1 Comparison of e-books formats showing that only EPUB 3 meets requirements for the creation of semantic and fluid mathematical content in an open format

Format	Fluidity (adjustment to the screen size)	Being supported on various platforms and devices	Openness (free use)	Digital Rights Management (limitation to a specific device)	Accessibility (to people with disabilities)	Semantic mathematical type
Mobipocket	yes	no	yes	optional	yes	no
AZW	yes	no	no	yes	yes	no
KF8	yes	no	no	yes	no	no
EPUB 2	yes	yes	yes	optional	yes	no
EPUB 3	yes	yes	yes	optional	yes	yes
IBA	yes	no	no	yes	yes	yes
PDF	no	yes	yes	optional	partial	no

Designing e-books, the books are most commonly degraded to the digital form of printed books [2], and designers as active participants in the e-book creation process are in a dilemma when choosing from the contemporary complex ecosystem of formats and devices [3]. However, one of the greatest advantages of e-books is that the content can be adjusted to the screen size similarly to the fluid layout in web design [4]. Actually e-books technologies are based on the open Web technologies. Although PDF is still among the most popular digital formats and is supported to the highest degree, i.e. it is readable on all devices and platforms, it cannot be perceived as a serious candidate for the e-book format because it does not enable active adjustment to the screen size, which makes it extremely inaccessible to devices with small screens (primarily mobile phones), and it is the only one not based on Web technologies. Other popular formats include Mobipocket, its successors AZW and KF8, iBooks and EPUB2/3. Mobipocket is a fluid format intended for the presentation of text and images but it does not support interactivity or multimedia. It is readable on Amazon Kindle devices, Mobipocket Reader software and some Simbian mobile devices. However, there is no means to embed semantically and correctly marked up mathematical content into the Mobipocket format. That can be achieved only through using images, which eliminates the possibility of a fluid layout and accessibility to text-to-speech (TTS) technology. AZW and KF8 formats are proprietary formats owned by the company Amazon, which are readable only on Kindle devices and applications, locked by Digital Right Management (DRM) technology which prevents long-term storage and conversion of bought e-books. Also they do not support any means of semantically correct mathematical content, and KF8 does not have the possibility of text-to-speech (TTS). iBooks (IBA) is also a proprietary format owned by the company Apple and is readable only on Apple devices. Open formats are not

designed for a specific device and their specification is entirely accessible, and their advantages also include accessibility and independence. The only format implying openness, accessibility, searchability and fluid layout is EPUB3 (Tab. 1). Since 2011, EPUB 3 has been officially recommended specification by the organisation International Digital Publishing Forum (IDPF). It is based on open Web standards HTML5 (in XML version) and CSS3, it supports scripting using JavaScript, interactivity, accessibility through TTS and MathML markup language for mathematical content [5].

2 MATHEMATICAL TYPE ANALYSIS IN E-BOOKS

The main hypothesis of this paper refers to the optimal mathematical layout implying semantical correctness [6], and includes the markup language use enabling its fluidity, scalability, accessibility, searchability, updating and processing aimed at conversion into other formats. The hypothesis is that the markup language use will enable the dynamic mathematical layout by combining various mark-ups (Mathematical Markup Language - MathML) [7], styles (Cascading Style Sheets), as well as script languages through the Document Object Model (DOM).

The analysis of available examples resulted in determining two basic models of formatting and embedding mathematical formulas into e-books. The first model uses bitmap images and in that way achieves precise typographic design but the quality of image is lost, it does not enable fluidity of content and the conversion into other formats, and it also complicates and lengthens the process of potential content editing. The use of TTS (text to speech) technology, in this case, implies textual description of each individual equation which results in semantic incorrectness and a significant possibility of incorrect pronunciation which alters the content meaning. Accessibility and readability of such mathematical content is disabled because bitmap images are not scaled well, the content enlargement through the standard menu is disabled and zooming and reading of the image content is complicated and demanding. Bitmap image content is not accessible for search or conversion. The use of vector images using SVG language is possible facilitating the quality of display but the problem of fluidity, text-to-speech and conversion remains the same as with the bitmap images.

The second model uses the mathematical markup language MathML for the content description. MathML, basing its grammar on XML, is the standard and it has been recommended by W3C. Unlike images, the use of MathML enables interactivity and access to the mathematical content. When using semantically rich mathematical content, copying, searching and TTS technology use is enabled. The problem in relation to MathML is not the fact that it is too-complex for hand coding because there are quality software tools for the purpose, but its entire support. Due to the fact that the specification has not been entirely accepted by any of the browsers, and consequently by any e-reader, the creators, i.e. content designers are not able to control the layout design. Therefore, in various e-readers different problems occur such as incompatibility with certain CSS values, some MathML attributes are not supported, long formulas do not line-break and characters overlap. The only available example ("The Theory of Heat Radiation, by Dr. Max Planck. 2012-13 Infogrid Pacific Pte. Ltd") that has resolved the problem of display universally uses `<mtable>` element aimed at forming table content and matrixes, which makes such content semantically and markup wise incorrect, thus creating problems at using TTS, transfer in other formats and subsequent processing.

The complexity of mathematical type display and formatting does not result only from dynamic layout, accessibility and readability which is not possible when using images, or semantical and markup correctness at using incorrect markups, an adequate markup language not being supported, but a combination of long formulas which need to break into more lines in order to fit the screen, and mathematical typographic rules. Mathematical typographic rules, to which professionals have been accustomed, dictate readability of such content and require for the mathematical content to be presented in the same typographic quality as the surrounding text. The book "Mathematics into Type" [8] states that the mathematical type needs to be scaled together with the surrounding text; the formulas placed in the same line with the text need to have the identical baseline; if the mathematical formula is too long to be presented in one line, the break needs to be placed at the operator (e.g. "=" or "+") or at equality signs (e.g. "<"), and the following line needs to be indented and aligned to the logical spot in the line above (Fig. 1).

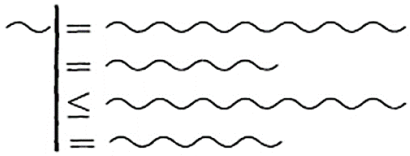

Figure 1 The rule for mathematical layout implies the break of a mathematical type at the operator or the equality sign, while the following line is indented and aligned to the first equality sign in the first line

3 MATHEMATICAL LAYOUT IN EPUB FORMAT

EPUB 3 standard e-book format is a packed set of files needed for rendering on e-readers. E-readers can be an online application (e.g. Readium – Chrome or EPUBReader - Firefox), a mobile application (e.g. iBooks – iOS, IDEAL Group Reader - Android), personal computer software (e.g. Adobe Digital Editions) or a designated device (e.g. Amazon Kindle, Barnes & NobleNook). In all e-readers, there are functions used to personalize reading experience including accessibility options such as adjustment of font size and type, image enlargement (in combination with *zoom&pan* viewing), content searching, screen rotation and text-to-speech functionality [9]. Amazon's Kindle device, the most popular e-reader nowadays, does not possess TTS technology, and neither does its application on Mac devices, and its most significant disadvantage is the fact that it does not support EPUB standard. iBooks software, although available exclusively on Apple devices, supports EPUB format and possesses the most advanced contemporary TTS technology VoiceOver [10]. E-readers formally supporting EPUB 3 standard in practice never

fully support the entire specification. The website epubtest.org maintains a systematic overview of support of EPUB 3 specification features by popular EPUB e-readers. Keeping that in mind, formatting of EPUB 3 e-books should be approached attentively because if certain features are not supported readers' access to certain content on certain devices or applications could be disabled.

EPUB 3 consists of the following 4 specifications intended for: semantics, content formatting, zip container packaging and text and audio synchronisation. It is based on HTML 5, i.e. XHTML5 standard. The use of XML syntax facilitates automated document processing. The greatest obstacle to popularisation of this format is the shortage of e-readers which support this specification fully. Because of that, IDPF and WebKit with their partners Adobe, Sony, Google, etc. started developing Readium open source system for EPUB 3 files presentation. Readium supports basic features well enough, but it still lacks support for some more advanced features [11].

When developing an EPUB 3 e-book what needs to be done first is creating XHTML5 or SVG content documents. After that a *package* document is formed, which is a special document used by e-readers in order to collect all needed information about a book. *Package* consists of a *manifest* which lists all sources needed to present content including styles, fonts, scripts, images, videos and XHTML documents. All metadata, such as information on the author, creation time, the tile, ISBN, etc. are also enumerated there. Eventually, the default order of content documents reading is stated in the section *spine*. The last step is packing all documents into one by using ZIP [12]. If content documents use MathML markup language, the value *math* needs to be specified in the section *manifest* in the attribute *properties*, and if scripted content is used, the value *scripted* needs to be specified in order for the e-reader to recognise the type of content in advance. Therefore, one chapter with scripted mathematical content has the following form:

```
<manifest>
...
<item id="p1" media-
type="application/xhtml+xml"
href="chapter1.xhtml"
properties="mathml scripted"/>
...
</manifest>
```

At creating EPUB 3 e-books, it is possible to use specialist tools such as iBooks Author or Adobe InDesign, but using these tools rarely produces semantically correct content marked up in a quality manner.

Semantical correctness and correctly marked up content is important in the data processing context, for transferring into other formats, and especially at creating digital content accessible to blind and partially sighted people. In case of EPUB 3 e-books, there is no need to create special audio books for people with disabilities while the use of TTS technology and quality screen readers enables automatic production of highly-accessible content, provided that there is a quality semantic markup [13]. The prerequisite for the creation of an EPUB e-book accessible to blind and partially sighted people via screen readers, such as VoiceOver technology of Apple devices, is the well-structured content [14].

Due to potential mistakes, the use of specialised tools is recommended for creation of automated content, such as navigation and incorporation of mandatory metadata elements. On the other hand, at creating content documents, the use of text editor typically used in creating websites is recommended. Whether a created EPUB document is in line with the specification is checked through an EPUB validator at the website validator.idpf.org [15].

4 EXPERIMENTAL MATHEMATICAL LAYOUT IN MathML LANGUAGE

Mathematical Markup Language (MathML) is intended to facilitate the use of mathematical and scientific content on the Web, as well as on other applications such as computer algebra systems, print layout and voice synthesis. MathML is an XML application and with the effective support of CSS, it enables native display of mathematical expressions. The use of MathML has been recommended by W3C since 1998, but it still lacks native support in browsers. Its purpose is description of mathematics for communication among machines, and it is not intended for editing by hand [7]. MathML 3 specification consists of presentation and content markup. Within EPUB 3 specification, both types are supported in the form of embedded content, but at the time being there is no support of content markup within e-readers. For the creation of mathematical expressions by MathML markup language, the interactive tool MathType for creating MathML or TeX mathematical content functioning in combination with other software (e.g. Microsoft Word or Adobe InDesign) has been selected. The creation is conducted through a simple interface with the existing templates without the need for the computer language use. It also offers the possibility of translation from or into the TeX notation [16]. MathML uses Unicode notation for all characters, which ensures standardised processing and display. For the use of the embedded MathML entry it is necessary to use the adequate *namespace* within `<math>` element:

```
<math
xmlns="http://www.w3.org/1998/Math/Mat
hML">...</math>
```

Although presentation elements are used for description of the mathematical entry, they are independent of media in the way that they contain enough information for good speech interpretations. Certain characters are used as operators whose display is identical to symbols of different meaning, or they are traditionally not represented at all such as `&InvisibleTimes`. These characters carry special importance in text-to-speech as well as a role in visual layout and spacing control and they need to be included in the markup. The specification also describes a detailed line break control within mathematical equations using `linebreak`, `lineleading`, `linebreakstyle` and `linebreakmultchar` attributes, which however, are still not supported in browsers.

MathJax is the project developed to resolve the problem of shortage of native support for MathML display in the web environment, as well as the problem of interaction of mathematical type with other applications and environments. MathJax also facilitates access to blind and partially sighted people through the inbuilt zoom mechanism which does not degrade display, but it also retains the accessibility of the original code to screen readers such as MathPlayer [17]. MathJax can render content in the following three ways: by using HTML and CSS, by using SVG and by using native MathML support. HTML/CSS output uses Stix web font for the display which enables optimal display regardless of fonts which a user has installed at his/her computer. SVG output sometimes has difficulty in adjusting to the screen size and its use is therefore not recommended if there is an alternative [18].

The display of MathML mathematical expressions in browsers and e-readers is done in one of the two possible ways. There is either native support to MathML language, or MathJax JavaScript library is used. This paper has proved that if there is no native support, the only possible solution is to build in MathJax library enabling display into an e-book. The experimental method of embedding MathML content into EPUB e-books has shown that e-readers whose display is based on Gecko (Firefox) or Safari rendering mechanisms have native support, and those include EPUBReader (Firefox), iBooks (iOS) and Adobe Digital Editions (OS X). Other e-readers use MathJax library for display: Readium (Chrome) and e-readers on Android devices such as IDEAL Group Reader.

The automatic embedding of JavaScript library is not advisable but the script verification of the native support is recommended, and the adjustment of the formatting to the data obtained. If an e-reader does not have native support MathJax is used which takes over the role of rendering. In that way, when native support in e-readers increases, e-books shall no longer require MathJax, and MathML shall be able to function universally on all devices.

The research has shown that the only tag within MathML specification helping to control line break is `<mrow>` used for grouping data. Wrapping parts of mathematical expressions with `<mrow>` actively controls the break point by *chunking* the expression into logical typographic units. When it comes to content in brackets, `<mrow>` prevents breaks within brackets. If an operator needs to be forced into the following line, that element also needs to be wrapped by `<mrow>` element. `<mrow>` also plays a role in formatting if the content is displayed through MathJax. MathJax recognises these elements and if content is generated in the form of SVG, it creates equivalent SVG `<g>` elements which can be manipulated through DOM. In case of HTML/CSS output, MathJax creates `<span>` elements of the class `mrow` which are in that case manipulated as needed.

Regardless of the support, it is useful to apply attribute `linebreak` with the value `'nobreak'` to the element of the first operator which prevents its transfer into the new line. The use of `linebreak='goodbreak'` is advisable at standard break points.

When using elements which are generally displayed larger than other characters, such as brackets, sum or integral symbols, it has been noticed that the display depends on an e-reader's renderer. The adjustment of their display size is conducted by using `<mstyle>` element with the attribute `dipslaystyle='true'`, which in that case uses more vertical space, and symbols are displayed in a larger version [19].

Unlike preparation of MathML content for the web, at creating content for EPUB, the named entities are not to be used since they are not allowed in XHTML5 documents, but their decimal or hexadecimal equivalents should be used. For instance, the integral symbol should not be described by HTML name*&int,* but by its hexadecimal code: ∫.

5 EXPERIMENTAL DYNAMIC MATHEMATICAL LAYOUT SOLUTIONS

Precise formatting and display control entirely depend on a particular e-reader. The aim of this paper is to accomplish the display of a mathematical formula in the way that it breaks into the new line in relation to the available screen (or window) space, the break is done at the mathematical standard points and that each line of the mathematical expression has an indent equivalent to the distance from the beginning of the expression to the first operator. Due to the fact that each e-reader has a different level of support of MathML, CSS and JavaScript and their combinations, it was necessary to identify each individual problem of support and resolve it in the way that it can be applied universally. The experimental section of the paper shows potential resolutions to the problem on a few selected examples of mathematical expressions. Figs. 2 and 3 show problems of display at individual e-readers in relation to the means of display on a certain platform.

5.1 Solution for Firefox

Firefox (Gecko) has been tested by EPUBReader e-reader having the best native MathML as well as JavaScript support. It has been noticed that in the given e-reader, JavaScript is not turned on by default and each user needs to do it manually in settings which diminishes the usefulness of this feature since users are rarely accustomed to adjusting settings. The mathematical equation is displayed and line-broken at the intended points even without the JavaScript intervention (Fig. 2), which has been enabled solely by MathML and the already described intervention of `<mrow>` elements. The only problem to be resolved in this e-reader is the application of the indent on the lines that break. The solution to this problem was achieved by creating JavaScript function that contains the loop for retrieving the first free mathematical operator `<mo>` which is found as the first child `<math>` of the parent element (Fig. 4). After that, its distance in the number of pixels from the left screen margin is retrieved by the DOM method `getBoundingClientRect().left`, in order to determine the required indent size.

Then follows the retrieval of all `<mrow>` elements created for the purpose of grouping of segments of a formula according to desired potential break points, defined as the first child of a `<math>` element and they are stored in the array `mrows[]`. Then follows the retrieval of their distance from the top of the screen which is compared to the distance of the first `<mrow>` element from the top of the screen (Fig. 5). If the distance of the first `<mrow>` element is shorter than the `<mrow>` element being checked, the left margin is actively applied to it in the size of the previously retrieved distance of the first operator:

```
el.setAttribute('style', 'margin-
left:' + uvlaka + 'px');
```

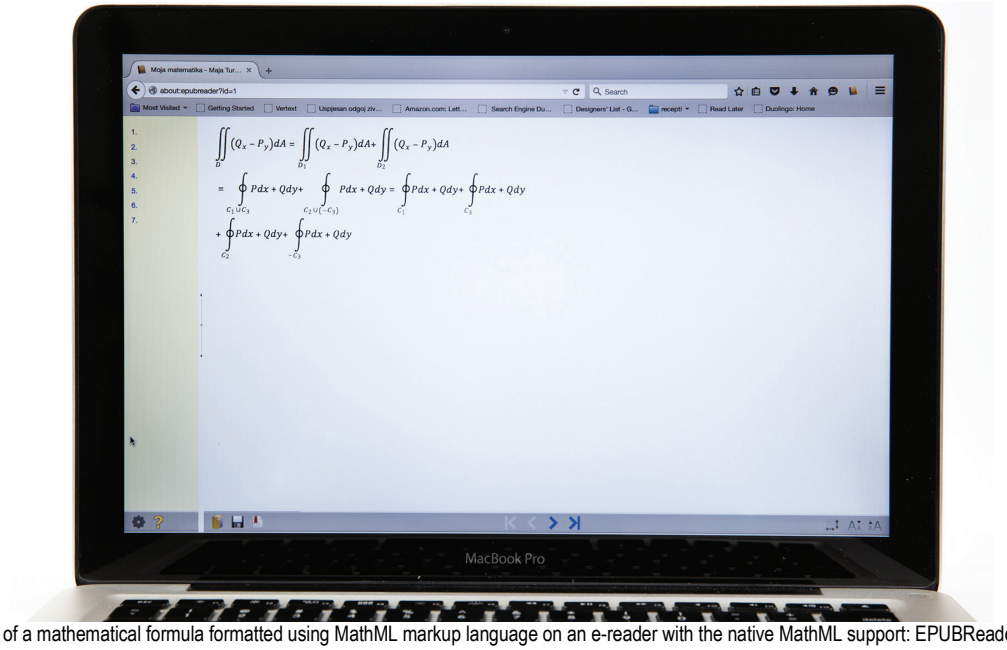

Figure 2 Display of a mathematical formula formatted using MathML markup language on an e-reader with the native MathML support: EPUBReader (Firefox) – the formula line-breaks, but it does not have an indent

Figure 3 Display of a mathematical formula formatted using MathML markup language on an e-reader using MathJax library for MathML display: IDEAL Group Reader (Android); display on an e-reader with the native MathML support: iBooks (iOS). At both devices the formula does not break and consequently a part of the content is unavailable to the reader

```
var firstOperator = [];
for (var i = 0; i < matem[0].childNodes.length ; i++) {
    var child = matem[0].childNodes[i];
    if(child.tagName == 'mo'){
        firstOperator.push(child);
    }
};
var uvlaka = firstOperator[0].getBoundingClientRect().left;
```

Figure 4 Retrieving the distance of the first operator by the JavaScript *for* loop for EPUBReader (Firefox- Gecko) for determining the required indent size

The function is fired after loading the page using the JavaScript event `onload`, as well at each change in the window size using the event `resize`. The function begins by setting all margins of `<mrow>` elements to 0. Fig. 6 shows a mathematical formula in EPUBReader on Firefox after the application of the created function.

```
var resizer = function(){

        resetOffsets();
        var currentY = mrows[0].getBoundingClientRect().top;
        var startIndex=0;
        while(startIndex < mrows.length-1){

            currentY = mrows[startIndex].getBoundingClientRect().top;
            for (i = startIndex; i < mrows.length ; i++) {
                startIndex=i;
                var el=mrows[i];
                if(currentY < el.getBoundingClientRect().top){
                    el.setAttribute('style', 'margin-left:' + uvlaka + 'px');

                    break;
                        }
                    };
                }
        };
```

Figure 5 The function for e-readers with the native MathML support for setting indent of the size of the distance of the first operator: the comparison of the distance of the `mrow` element and the first operator from the top of the screen, and setting the margin (EPUBReader, iBooks, Adobe Digital Editions)

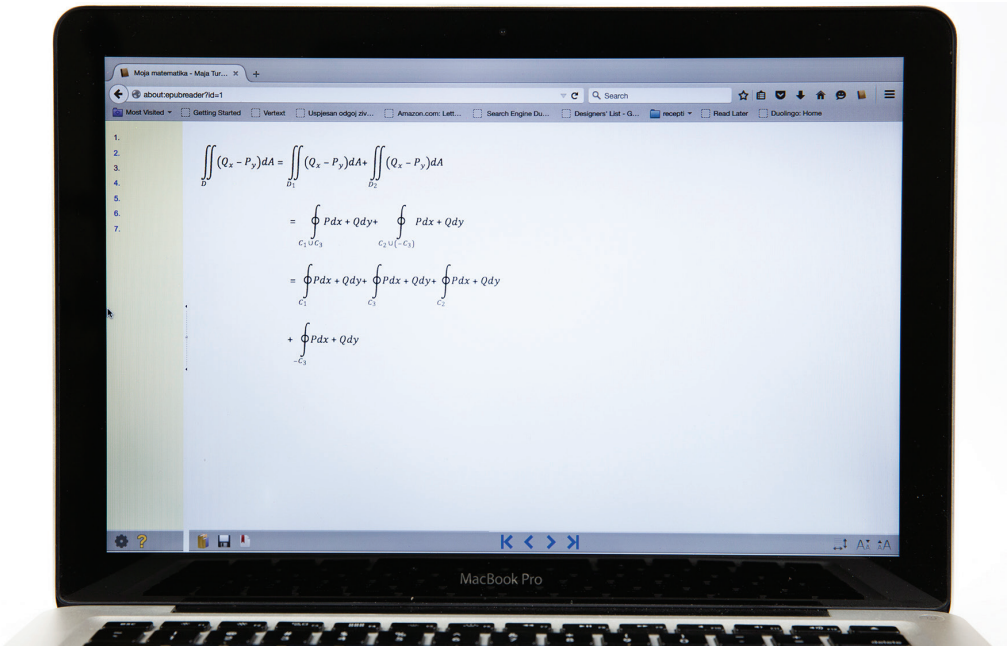

Figure 6 The solution on the e-reader EPUB Reader (Firefox). The formula is displayed and the desired indent size is applied.

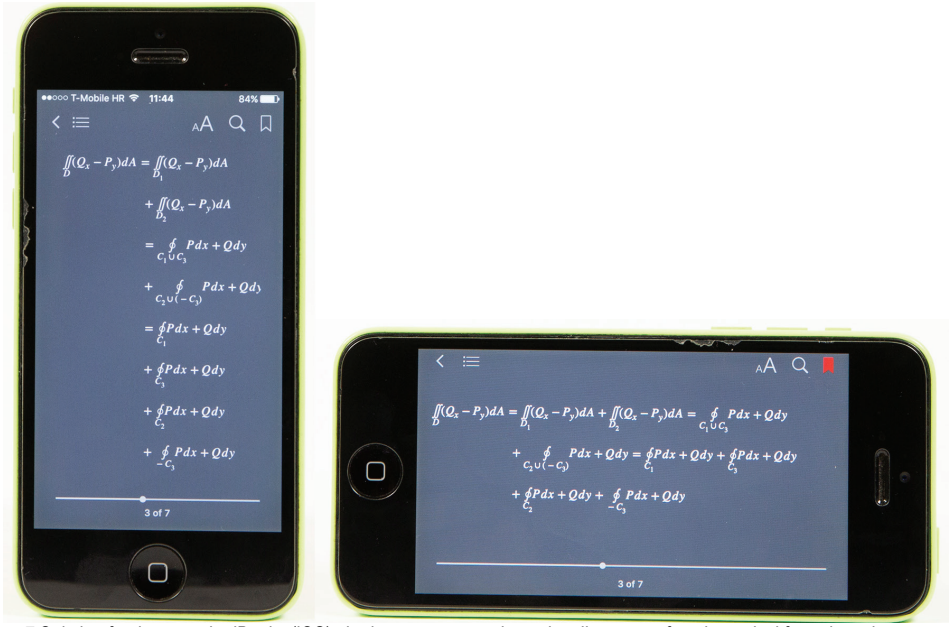

Figure 7 Solution for the e-reader iBooks (iOS); the image presents dynamic adjustment of mathematical formula to the screen size

5.2 Solution for iBooks

iBooks application on Apple devices also has the native MathML support. However, it processes and renders mathematical content differently than the Gecko e-reader. If a mathematical expression does not fit into the screen, the segment that does not fit is not shown and it cannot be accessed because it uses the CSS flex method

for layout (Fig. 3). The problem of display in the iBooks e-reader is resolved using CSS property flex-wrap which enables line breaking, i.e. the display of complete mathematical expressions:

```
math {-webkit-flex-wrap: wrap;}
```

Since CSS flex is still in its experimental phase, the prefix –webkit– needs to be added to the property. After resolving the display problem, the indent application was performed. It resulted in the need to change the function made for EPUBReader because iBooks does not comprehend the method `getBoundingClientRect()`, and consequently the method `offsetLeft` was used for retrieving the distance of the operator, and `offsetTop` for the comparison of differences of `<mrow>` elements, while the remaining part of the function remained identical. The solution to the problem of dynamic layout is presented in Fig. 7. It has been established that it is not possible to develop a universal method for EPUB Reader and iBooks because there is no universal method for retrieving the length that both e-readers can recognise. Therefore, before starting the indent function it is necessary to check whether the method is supported and apply the required one.

5.3 Solution for Adobe Digital Editions

Adobe Digital Editions also has native MathML and JavaScript support on Windows and OS X platforms, so the function created for Firefox and iBooks applies the desired indent successfully in this case as well.

By using this JavaScript function, the problem of breaking long mathematical expressions is resolved for all devices which natively support MathML and it can be used as long as MathML attributes aimed at breaking do not get support.

5.4 Solution for Readium

By opening the source code of the EPUB e-book mathematical content in Readium, it has been found that e-readers using MathJax for the MathML rendering generate a different DOM structure and the created function cannot retrieve the desired MathML elements nor create indents. The problem is universal for all e-readers which do not have the option of displaying mathematical expressions at all and the e-book author needs to build in MathJax library on its own.

Readium uses MathJax with SVG output and is therefore required to become familiar with the newly generated DOM structure and MathJax library in order to be able to intervene in the layout using JavaScript. The generated DOM structure depends on the MathML formatting using `<mrow>` elements in the way that each `<mrow>` becomes an SVG `<g>` element. First, the MathJax library is configured in order to enable an automatic line break of the mathematical content into the next line:

```
<script type="text/x-mathjax-config">
   MathJax.Hub.Config({
   SVG: { linebreaks: { automatic: true
},
       }
       });
</script>
```

MathJax uses HTML `<iframe>` element for building in of the content within which SVG is placed. In order to determine the position of the first operator initially the first operator is retrieved through its «id» atribute. All positions of elements are recorded in the form `transform: translate (x,y)` of SVG attributes and in that way the x position of the first operator is retrieved in combination with regular expressions (Fig. 8).

```
var svg = window.document.getElementById('MathJax-Element-1-Frame').getElementsByTagName('svg');

var operator = window.document.getElementById('firstOperator');
var poz = operator.getAttribute('transform');
var parts  = /translate\(\s*([^\s,)]+)[ ,]([^\s,)]+)/.exec(poz);
var xPoz = parts[1];

var drugiRed = svg[0].firstChild.lastChild;
var drugiRedovi = svg[0].firstChild.childNodes;
var govi = [];
for (var i = 0; i < drugiRedovi.length ; i++) {
    var child = drugiRedovi[i];
    if(child.tagName == 'g'){
        govi.push(child);
    }
};
```

Figure 8 Retrieving iframe with the SVG element through which all <g> elements are collected in an array, retrieval of the first operator through its «id» and its distance at the *x* axis with the regular expression

```
for (var j = 1; j < govi.length; j++) {
        var koo = govi[j].getAttribute('transform');
        var xy  = /translate\(\s*([^\s,)]+)[ ,]([^\s,)]+)/.exec(koo);
        var X = xy[1],
            Y = xy[2];
        if(X==0 && Y!=0){
            govi[j].setAttribute('transform', 'translate('+xPoz+','+Y+')');
        }
    };
```

Figure 9 Loading x and y positions of the SVG <g> elements and setting an indent if the *X* position is equal to 0, and the *Y* position is different than 0

MathJax stores each new line into its own SVG `<g>` element on which the indent is applied using `transform: translate (x,y)` attribute in the value of the retrieved distance of the first operator. It is required for the `<g>` element to have the value 0, and the *Y* value different than 0 in order for the indent to be applied (Fig. 9).

4.5 Solution for Android

The last analysis is conducted on the Android platform in IDEAL Group Reader using MathJax library with HTML output for the display of MathML language. The process again begins with configuring MathJax library for automatic line breaking. Unlike SVG output, the DOM structure is manipulated using `<span>` elements which retain the name of the original MathML element, but in the form of the class name. The retrieval of the distance of the first operator is conducted through a newly generated DOM structure. The first operator has the name class «mo» whose distance is determined using the DOM `offsetLeft` method. Lines are retrieved by iteration through specific `span` elements and the left margin is applied to them through CSS features (Fig. 10). Fig. 11 shows the solution on the Android platform for IDEAL Reader group e-reader.

```
var uvlaka = function()
{

    var mat = window.document.getElementsByClassName('math');
    var firstOperator = mat[0].getElementsByTagName('span')[3].childNodes;
    var spanovi = [];

    for (var i = 0; i < firstOperator[0].childNodes.length ; i++) {
        var child = firstOperator[0].childNodes[i];
        if(child.className == 'mo'){
            spanovi.push(child);
        }
    };
    var x = spanovi[0].offsetLeft;

    var drugiRedovi = mat[0].getElementsByTagName('span')[3].childNodes;
    var polje =[];
    for (var j = 1; j < drugiRedovi.length; j++){

        var dete = drugiRedovi[j];
        polje.push(dete);
    }

    polje.forEach(function(row){
        row.firstChild.setAttribute('style', 'margin-left:' + x + 'px');
    })


};
```

Figure 10 The function for e-readers which use MatJax with HTML output. Retrieval of the distance of the first operator through the generated DOM structure, retrieval of `<span>` elements equivalent to MathML `<mrow>` elements and setting an indent using the CSS feature for the left margin

6 EXPERIMENTAL RESULTS ANALYSIS

The solution to the problem of display and typographically adequate mathematical layout is proved to be dependent on the e-reader which is used when loading an EPUB e-book. E-readers, based on the MathML language rendering, are divided into two main groups: those that contain native MathML support and those which use MathJax JavaScript library for rendering. It has been shown that, although the MathML language contains attributes needed for the line break control, e-readers still do not recognise them and therefore they cannot be used. It has been established that e-readers on Gecko and Safari platforms natively support the MathML language and the line break at desired points is enabled by the use of adequate grouping markup at creating MathML content. The paper demonstrates the method of indentation of following lines in the size of the distance of the first operator within the mathematical equation through the JavaScript function. It has been shown that e-readers on Chrome and Android platforms use MathJax library for display that changes the DOM structure of content. After determining a new, altered DOM structure, a different method of retrieving the distance of the first operator as well as of applying indentation on following lines has been created, individually for SVG and HTML output (Tab. 2).

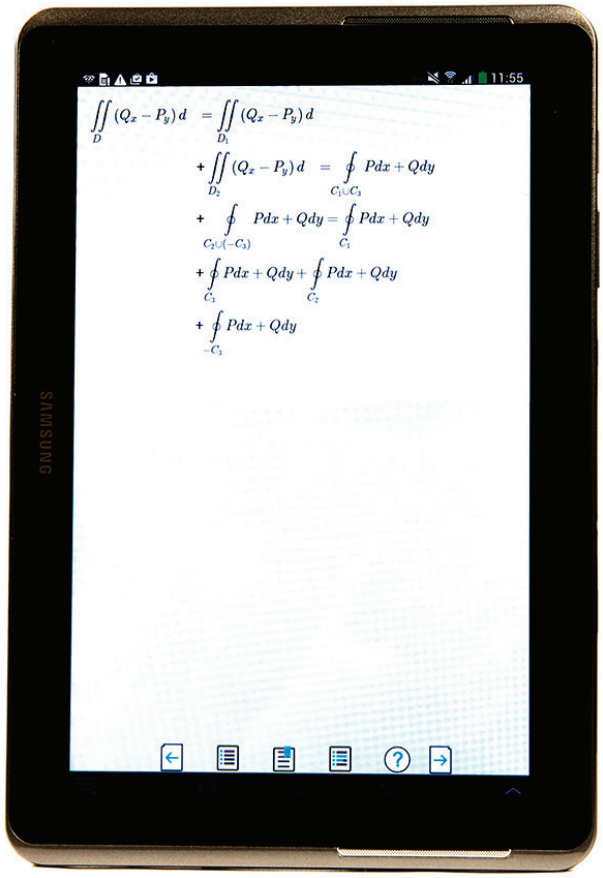

Figure 11 Solution on an IDEAL Reader group e-reader (Android). The formula is broken at the desired points, it is adjusted to the screen size and the indent is applied

Table 2 Classification of e-readers based on MathML rendering and processing

e-reader	MathML support	platform	output
EPUBReader	native	Gecko	MathML
iBooks	native	Safari	MathML
Adobe Digital Editions	native	Safari (OS X)	MathML
Readium	MathJax	Chrome	SVG
IDEAL Group Reader	MathJax	Android	HTML
Calibre	MathJax	Mac/Windows	SVG

7 CONCLUSION

This paper presents a solution for incorporation of fluid, semantically correct and typographically formatted mathematical content into e-books. Fluidity and semantical correctness ensure a high content accessibility regardless of the device used for visual reading and voice synthesis. The only e-book format that enables the named

features is EPUB 3 which is an open format, it is based on Web technologies, it has fluid layout, allows access to TTS technologies and supports the MathML language. MathML is a markup language easily created using software tools and it enables transfer into other formats, browsing and voice synthesis, which is not possible when embedding mathematical formulas in the form of bitmap images. It has been proved that the display and formatting of complex mathematical formulas depends on the support of the MathML language in various e-readers. E-readers render MathML either natively (EPUBReader, iBooks, Adobe Digital Editions) in which case the typographic formatting is achieved using the JavaScript function, or using MathJax JavaScript library (Readium, Calibre, IDEAL Reader) requiring detailed knowledge of the complex DOM structure which MathJax generates and after which a new JavaScript function has been created and applied. The solutions presented in this paper enable publishing of e-books with complex mathematical content such as scientific and academic papers and course books. The future will probably witness a decrease of the need for scripted solutions because an increase of the MathML language support in e-readers is anticipated.

8 REFERENCES

[1] Baramidze, V. (2014). LaTeX for Technical Writing. *Journal of Technical Science and Technologies, 2*(2), 45-48.

[2] Martin, C. & Jonathan A. (2011). Evolving definitions of authorship in Ebook design. *Information Services and Use, 31*(3-4), 139-146. https://doi.org/10.3233/ISU-2012-0643

[3] Warren, J. W. (2009). Innovation and Future of e-books. *The International Journal of the Book, 6*(1), 83-93. https://doi.org/10.18848/1447-9516/CGP/v06i01/36731

[4] Waller, R. (2012). Graphic Literacies for a Digital Age: The Survival of Layout. *The Information Society: An International Journal, 28*(4), 236-252 https://doi.org/10.1080/01972243.2012.689609

[5] EPUB 3.0. International Digital Publishing Forum. 2011. http://idpf.org/epub/30. (29.11.2016).

[6] Van Der Hoeven, J. (2015). Towards semantic mathematical editing. *Journal of Symbolic Computation, 71*, 1-46. https://doi.org/10.1016/j.jsc.2014.09.040

[7] Bos, B. What is MathML? Mathematical Markup Language (MathML). W3C Math Home. 2016. http://www.w3.org/Math/whatIsMathML.html. (29.11.2016)

[8] Mathematics in display. (1999). Mathematics into Type / Ellen Swanson. American Mathematical Society: Providence, Rhode Island, 45-48.

[9] Kim, C., Jeong, O., Choi, J., & Kim, W. (2013). E-book on the mobile e-reader. *Mobile Information Systems, 9*(1), 55-68. https://doi.org/10.1155/2013/509207

[10] Junus, S. G. R. & Booth, C. (2012). Ebooks and ereaders for users with print disabilities. *Library Technology Reports, 48*(7), 22-28.

[11] Bläsi, C. & Franz, R. (2013). On the interoperability of eBook formats. Johannes Gutenberg-Universität Mainz–Germany. http://wi.bwl.uni-mainz.de/publikationen/InteroperabilityReportGutenbergfinal07052013.pdf. (29.11.2016)

[12] Chapter 3: Content Documents - MathML. // EPUB 3 Best Practices / Matt Garrish and Markus Gylling. O'Reilly Media: Sebastopol, California, 2013, 72-78.

[13] Semaan, B., Tekli, J., Issa, Y. B., Tekli, G., & Chbeir, R. (2013). Toward Enhancing Web Accessibility for Blind Users through the Semantic Web. *Proceedings of 2013 International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)* / Kyoto, Japan, 247-256. https://doi.org/10.1109/SITIS.2013.50

[14] Lenzi, V. B. & Leporini, B. (2013). Investigating an Accessible and Usable ePub Book via VoiceOver: A Case Study. *Proceedings of the Human Factors in Computing and Informatics, First International Conference* / Maribor, Slovenia, 272-283. https://doi.org/10.1007/978-3-642-39062-3_17

[15] EPUB Validator (beta). International Digital Publishing Forum. 2012. http://validator.idpf.org/, (29.11.2016)

[16] Topping, P. (1999). Using MathType to create TEX and MathML equations. *Proceedings of the 1999 TEX Annual Meeting TUGBoat 20* / Vancouver, Canada, 3.

[17] Cervone, D. (2012). MathJax: A Platform for Mathematics on the Web. *Notices of the American Mathematical Society, 59*(2), 312-316. https://doi.org/10.1090/noti794

[18] MathJax Output Formats. The MathJax Consortium. 2015. http://docs.mathjax.org/en/latest/output.html. (29.11.2016)

[19] MathML Presentation Markup – mstyle. W3C. 1999. http://www.w3.org/TR/REC-MathML/chap3_3.html#sec3.3.4 (29.11.2016)

Contact information:

Maja TURČIĆ, MSc Lect.
Zagreb University of Applied Sciences
Vrbik 8, Zagreb, 10000 Croatia
mturcic@tvz.hr

Klaudio PAP, PhD Prof.
University of Zagreb, Faculty of Graphic Arts
Getaldićeva 2, 10000 Zagreb, Croatia
kpap@grf.hr