SANGEETTEX: A LATEX ENGINE FOR TRANSCRIBING AND RENDERING INDIC MUSIC

Chandan Misra
XIM University
Bhubaneswar, India
chandan@xim.edu.in

ABSTRACT

Notation system is the building block of a particular genre of music which aids in reading, composing, and performing music in a structured manner. Though computers are being used for annotating musical scores in *Staff notation*, it has not been used for Indic music to a substantial extent. Available systems and archives either employ romanization or uses image formats for representing scores which makes it less usable in terms of recreating scores as part of other documents, retrieving musical information out of the format etc. Creating music-sheets for Indic Music in computer is a inconvenient process involving placing music symbols in its proper place according to the underlying grammar and drawing the same similar to published musical texts. Document preparation tools like Latex is suitable choice for this task making it easier for developers to print quality music-sheets. In this paper, we present SangeetTEX, a Latex based music rendering engine for *Rabindra Sangeet* (*Tagore Songs* in English), a distinguished genre of Indic music and a collection of more than 2200 songs composed and written by Bengali poet and Nobel Laureate *Rabindranath Tagore*. It allows users to create beautiful music sheets while preserving the published typesetting in its original published form and provides easy exchange of musical information in text format. SangeetTEX is available at https://github.com/cmisra/SangeetTeX.

1. INTRODUCTION

Notation system for music provides means to read, compose, and perform music in a systematic manner. Staff notation is the standard music notation for traditional western classical music. Although, *Indian Classical Music* (ICM) has been orally transmitted from teachers to students, practitioners soon realized a music notation system for aiding the learning as well as performing the traditional artform. *Pt. Vishnu Narayan Bhatkhande* and *Pt. Vishnu Digambar Paluskar* was the first to create a notation systems for *North Indian Classical Music* or *Hindustani Sangeet*, termed as *Bhatkhande* [1] and *Paluskar* [2] notation system respectively. *Rabindranath Tagore*, poet and Nobel Laureate, gave birth to a unique kind of music genre called *Rabindra Sangeet*. Asia's first Nobel Laureate also proposed a new notation system, *Akarmatrik Notation System* [3] which was used to notate all of his (more than 2200) songs. There are other prominent notation systems present, *Dandamatrik* and *Sargam* [4] for *South Indian Classical Music* or *Carnatic Sangeet*.

Since, it is very important to make Indic music in computer in terms of music education and practice, the very first step towards practicing the art in a more Indic environment (contrary to Staff notation where there are always a chance to miss certain musical information during score writing). One of the starting points for creating such systems is to provide a tool to properly render the music-sheet on a piece of paper. *Swarshala* [1] [5] from *Swar Systems*, is a prominent Indic music software that allows practice, compose and learn both Hindustani and Carnatic Sangeet. However, it uses romanization for entering musical notes to the system. Therefore, the system requires a mathematical realization of the sheet architecture in a computer and respective musical fonts for transcription. One of the attempts was to render Tagore compositions [6] in a computer that leads to a mathematical framework, Swaralipi [7], that tries to make a unification of various Indic music-sheet representation while extracting common features of different notation systems and language script to make a general notation system. However, such framework when actually implemented does not generate expected results due to the rendering challenges posed by certain music symbols - *Meend* symbol of Akarmatrik notation system and vertical line to represent *tālbibhag* symbol in Bhatkhande notation system. Additionally, it is not easy to place music symbols above and below other symbols which is why new notations are developed to curb the need for digitization of music-sheets. While these systems are quite capable of entering, storing and playing back music, publishable quality rendition of music-sheets is too hard to achieve and therefore it actually diminishes the very objective of creating the music-sheet in the first place.

In this paper, we try to develop a rendering engine, SangeetTEX, based on Latex document preparation tool that specifically allows music software developers to create print quality music-sheet implementing the grammar of ICM. Initially, we develop Latex commands to render scores of *Tagore Songs* following the model *Swaralipi* since it has

[1] SwarShala 4 by Swar Systems. Visit the following link for more details https://www.swarclassical.com/SwarShala/

a large corpora of songs and accompanied scores all published in 63 volumes [3], an ideal test bed for SangeetTEX. Later we try to include other varieties of classical music in SangeetTEX by implementing respective notation systems. The commands let the developer concentrate on methods for entering music symbols on the user interface and let them free from the complexities of music-sheet rendition. SangeetTEX defines algorithms that place various symbols on the music-sheet with the fonts for scores and lyrics. The choice of Latex is suitable since it can draw shapes whenever it is necessary to render certain symbols which would not be rendered properly if fonts were used. Since, the structure of the $t\bar{a}l$s are same in all form of traditional music, the same engine can be applied to other forms of notation systems.

2. A BRIEF ON SWARALIPI MODEL

The present Latex notation rendering engine is built upon a well-established musical framework, *Swaralipi* [7], which had been created to encode, arrange, display, and render Indic music symbols on a computer. The framework can arguably support all major Indic notation systems active at present in India. Although, the core of the framework consists of a row and a column model for rendering rows and columns of the music sheet respectively, it is greatly influenced by the annotating styles of Indic music and Indic language scripts. However, to understand the present work, explanation of the row and column models are sufficient and interested readers can therefore may refer to the original article cited for a comprehensive detail.

The framework can be visualized as a collection of 2D matrices of heterogeneous dimensions where each 2D matrix corresponds to a single line of the composition. Each such matrix consists of cells at each intersection of all the rows and columns of the matrix. These cells serve the containers of the music symbols. The row and column model define the number and alignment of the rows and columns for each 2D-matrix in the model. While row model takes only the lines of the composition as the input, the column model calculates the number of columns using three parameters: $t\bar{a}l$ and the implicit pattern carried with it, the *avartan* or the number of cycle, and the position of each music symbol in each 2D-matrix or line. We have maintained the same input parameters while designing SangeetTEX as described in Section 3. Below we describe the row and column model.

2.1 Row Model

The row model determines the number of rows that should be present in each 2D-matrix of the music sheet. In its simplest form, row model prescribes two rows for a line - the score and the lyrics. However, variations on this simplest form is possible with the introduction of other musical components like melody change during repetition, *Meend* (musical ornament), end of composition, $t\bar{a}l\bar{a}nk$ or beat markings etc. These variations led to the formation of two separate segments: *Primary Melody Segment* which consists of the musical components related to the primary melody line, and *Changed Melody Segment*, which consists of alternate musical components followed while repeating part of the primary melody line. Each segment mentioned above, consists of maximum four rows: $t\bar{a}l\bar{a}nk$ or beat marking row, score row, *Meend* row, and lyric row. However, not all the 2D-matrices in a composition require eight rows since, not all the lines use all the musical components.

Row model also implements the rendition of *Meend*, the ornament which slides from one note to another note of different pitch over a specified number of beats (similar to *Glissendo*). The form of the symbol makes it difficult to place in a particular cell of the 2D matrix. The model solves the problem by introducing three new symbols - *Meend Start*, *Meend Continue*, and *Meend End* and placing them in the *Meend Row* of the corresponding line. *Meend*s can differ in length depending on the number of beats it covers and can be implemented by repetitive use of the *Meend Continue* symbol.

Although the model provides a probable solution to the Meend issue, in a practical scenario, it fails to achieve proper rendition similar to published music piece. This includes alignment problem that gives users the freedom to pose variable spacing between adjacent cells of a 2D-matrix. Since, the continue symbol length is fixed, it is not possible to make proper justification of the *Meend*. This led us to draw the symbol in latex to render it in more neater and flexible way as described in Section 4.1.

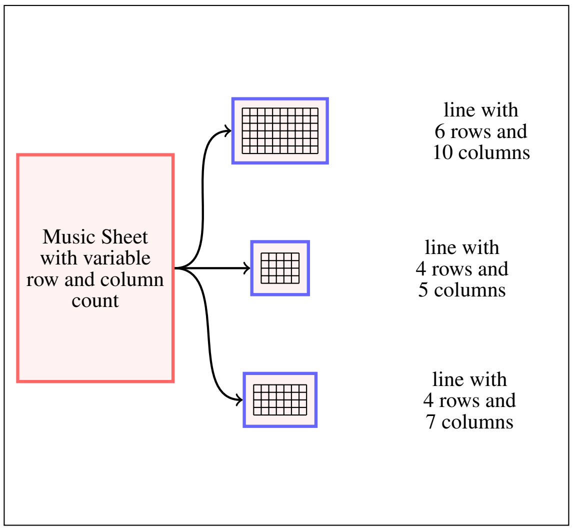

Music Sheet with variable row and column count
line with 6 rows and 10 columns
line with 4 rows and 5 columns
line with 4 rows and 7 columns

Figure 1. 2D-Matrix model of SangeetTEX.

2.2 Column Model

The column model takes $t\bar{a}l$ and *avartan* as the input and computes the number of columns present in each 2D-matrix and the position of the music symbols in it. Since every $t\bar{a}l$ is accompanied by its beat pattern, the column count can be easily computed as explained in the following example:

Figure 2 shows a 2D-matrix (bottom part of the figure) created from a single line (top part of the figure) having *Shashthi* $t\bar{a}l$ with two unequal measures having 2 beats in the first measure and 4 beats in the last, 2 *avartan*s and the lyric line is written in Bengali script, have been transformed into the architecture having 2 rows and 17 columns.

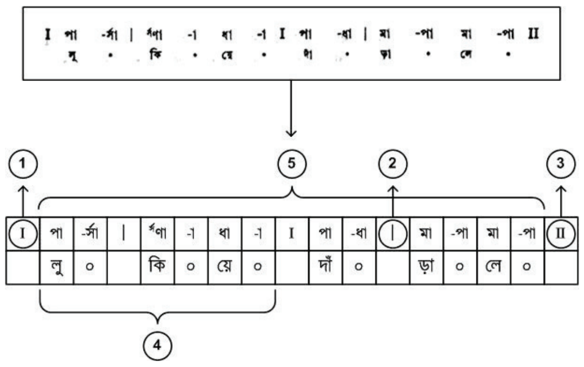


Figure 2. Determination of column number of a line of Rabindrasangeet score in the present architecture.

The calculation depends on three symbols as shown in Figure 2:

1. Symbol 1 or (I): The beginning of the $t\bar{a}l$ occupies a cell before the first beat of the $t\bar{a}l$. It is repeated once every cycle of the $t\bar{a}l$ or *avartan*.
2. Symbol 2 or (|): The $t\bar{a}lbibhag$ symbol which comes between adjacent measures of the $t\bar{a}l$ and occupies a cell by itself. Since there are only two measures per cycle of this $t\bar{a}l$, symbol 2 is used only once per cycle and can be generalized using the following equation:

$$n = (a \times m) \tag{1}$$

where n is the number of columns needed for symbol 2, a is the number of *avartan*, and m is the number of measures of the $t\bar{a}l$.

3. Symbol 3 or (II): The end of a musical phrase after which the first phrase (called the *Aasthayee*) must be sung; it also occupies a cell by itself.

Putting it altogether gives us the total column count t as:

$$t = (b \times a) + (a - 1) + n + 2 \tag{2}$$

where b is the total number of beats in the $t\bar{a}l$. The column count 17 in the above example can be easily verified from the equation.

3. STRUCTURAL ANATOMY OF SANGEETTEX

Since we are interested in creating a new document format i.e. music-sheet for score transcription and not in adding more functionalities to an existing type of document (Latex's default document class *article* for example), a *class* rather than a *package*[2] [8], have been considered a better choice for our project.

The foremost aspect of consideration to creating the structure of the music-sheet format is the symbols, which in this case belong to two classes, to be transcribed and rendered through Latex compiler. Two classes of symbols, the scores and the lyrics, are rendered using two different fonts.For lyrics, a Unicode Bangla font, named *Bangla*[3] [9] has been used and the scores use a custom font, named *Swarabitan* which was designed particularly to be used for writing scores for *Rabindra Sangeet* music-sheet. The use of Unicode font for the transcription in the TeX class file made us to select the compiler as *XeTeX* as a Unicode input and font aware engine. Another variant, LuaLaTeX engine, can also be used for this purpose.

The structure of the music-sheet is implemented in a single class file, named *sangeet* class file, and is primarily made up of two commands, namely *scoreLine* and *scoreLyricLine*, which correspond to creating a score line without and with a lyric line respectively. The scoreLine command takes three parameters: the name of the $t\bar{a}l$, the number of *avartan*, and the note sequence to form a single line of score. The *scoreLyricLine* command has an extra parameter, the lyric syllables, to be added beneath every note in the score line along with the above mentioned parameters. It is important to mention that the third parameter, note sequence, in both the commands do not include $t\bar{a}lbibhag$, *start* and *end of song* symbols.

The primary objective of the *scoreLine* command is to calculate the number of cells required to accommodate the music symbols and their respective positions on that particular line. This would obviously exclude the *Meend* symbol, since it would require different rendering altogether, to be discussed in Section 4.1. The total number of cells in a line is to be computed using equation 2 which defines the number of columns (column model) for a line in a music composition, while the position of each symbol requires more involved calculations which is discussed in Section 3.2.

3.1 Computation of Number of Columns

As already mentioned, the computation of number of columns requires two input parameters: $t\bar{a}l$ and avartan. These parameters have been given in both the classes *scoreLine* and *scoreLyricLine*. However, $t\bar{a}l$ name does not contribute to the actual calculation and therefore, require other $t\bar{a}l$ related parameters like *maatra* or measures, total beat count of a particular $t\bar{a}l$. These values are implicit to a specific $t\bar{a}l$ but must be given explicitly to obtain the count. Maatra and total beat count can be obtained from the beat pattern of the $t\bar{a}l$ and a comprehensive list of beat pattern of all $t\bar{a}l$s used in *Rabindra Sangeet* is added in the swarabitan class file. This list is included in a new latex environment named *sangeet* and the value of *maatra* and beat count are also initialized there to be used as a global variable later in both the classes.

Finally, the column count is calculated using equation 2 as follows:

[2] Latex2E for Class and Package Writer. Visit `https://www.latex-project.org/help/documentation/clsguide.pdf`

[3] Bangla, Unicode Bangla Font. Download at `https://www.omicronlab.com/bangla-fonts.html`

$$
\begin{aligned}
colcount &= ((avartan \times (maatra - 1)) \\
&\quad + (avartan \times beatcount) + (avartan - 1) \\
&\quad + 2 \\
&= (a \times (m - 1)) + (a \times b) + (a - 1) + 2
\end{aligned} \tag{3}
$$

3.2 Evaluation of the Positions of the Music Symbols

Since, both the latex commands takes only notes as the third input parameter (which is also intuitive since musical notes are the only symbols which are variable for different compositions), it is a bit tricky to determine the positions of the notes and other music symbols in any 2D-matrix. Typically, two essential symbols are present in a musical line: The beginning or end of the *tāl* (I) and *tālbibhaga* (|) as discussed in section 2.2. As previously seen, the presence of these two symbols prohibits notes to occupy consecutive positions in the 2D-matrix. Fortunately, the underlying systematic pattern of *tāl* resolves this indexing problem and therefore, implemented here with careful computation. Below we briefly explains the steps of the calculation (with a running example as given in Figure 2):

1. The key input and the tunable parameters for the computation are: *beat count* (b), *maatra* (m), *avartan* (a), beat pattern of *tāl* ($bPat$).

2. We use 1-based indexing meaning that the symbols occupy cells that start from column one till column equal to the total number of columns (given in equation 2) in that 2D-matrix. For example, in the running example the column index starts from 1 and ends at 17.

3. We create a sequence of numbers that starts from 1, increments by 1, and stops at the column count. For the running example it gives the following sequence: $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17\}$

4. We identify the indices where we place (I) symbol and put the indices in a sequence. It is placed at the beginning, at the end, and between cycles of the beats (*avartan*). If the line consists of single *avartan*, then there will be only two places where it is to be placed. The index for additional occurrences of the symbol is given by Algorithm 1. For the running example, the sequence contains three entries $\{1, 9, 17\}$.

5. We identify the indices where we place the *tāl* Bibhag Symbol (|) and insert them into a sequence. The indices to place the symbol is tricky since various *tāl* consists of unequal beat pattern as given in Table. The following algorithm (Algorithm 2) uses the number of beats in the first Bibhag or measure to calculate the indices:

   In the running example, the *tāl* bibhag sequence contains two entries $\{5, 13\}$.

6. Since, we are done with the positions of the two primary symbols, we can extract the indices of the positions of the notes by first combining the indices of step 4 and 5, and subtract it from the column count sequence created in step 3. In the running example, this would lead to the sequence:
   $\{2, 3, 4, 6, 7, 8, 10, 11, 12, 14, 15, 16\}$.

Algorithm 1: Algorithm to find the positions of the start symbol (I)

```
Data: $m, a, b, t$
Result: Sequence $seq$ that contains the indices of the
        symbol
insert $1$ to $seq$
if $a \neq 1$ then
    $index \leftarrow 1$
    for $i \leftarrow 1$ to $(a-1)$ do
        $index \leftarrow index + b + (m-1) + 1$
        insert $index$ to $seq$
    end
end
insert $t$ to $seq$
return $seq$
```

Algorithm 2: Algorithm to find the positions of the *tāl* Bibhag symbol (|)

```
Data: $m, a, b, t, bPat$
Result: Sequence $seq$ that contains the indices of the
        symbol
/* Calculate the number of times tāl bibhag
   symbol appears                                */
$talBibhag \leftarrow a \times (m-1)$
for $i \leftarrow 1$ to $talBibhag$ do
    insert $i$ to $talSeq$
end
/* get the beat count in the first Bibhag */
$first \leftarrow bPat[1]$ $N \leftarrow length(talSeq)$
for $i \leftarrow 1$ to $N$ do
    $temp \leftarrow 2 \times (talSeq[i] - 1) + 1$
    $index \leftarrow first + (b \times (talSeq[i] - 1) + temp) + 1$
    insert $index$ into $seq$
end
return $seq$
```

3.3 Evaluation of the indices for Meend

Before drawing the meend between notes, it is required to know the indices of the start and end notes. To annotate the start and finish of the meend in any line of composition, we place a backtick character (`) character just before the corresponding characters. Once we get the start and end characters we can draw the meend from start to end according to the method described in section 4.

4. RENDERING OF THE MUSIC SYMBOLS

We use tikz library to render the music sheet and once the positions of the music symbols have been evaluated, it is easy to place them using tikz. Each line of the music piece or 2D-matrix is rendered as a separate figure and each cell of such matrix is realized by a tikz node. A tikz node is

Notation Properties	English Alphabet	Rendition
Pure Notes or *Śuddh Svar*	s r g m p q n	স র গ ম প ধ ন
Flat Notes or *Komal Svar*	v t d u	ঋ জ্ঞ দ ণ
Sharp Notes or *Tīvr Svar*	k	ক্ষ
Middle Octave or *Suddh Saptak*	s r g m p q n	স র গ ম প ধ ন
Upper Octave or *Tār Saptak*	sf, rf, gf, mf, pf, qf, nf	র্স র্র র্গ র্ম র্প র্ধ র্ন
Lower Octave or *Mandra Saptak*	sh, rh, gh, mh, ph, qh, nh	স্ র্ গ্ ম্ প্ ধ্ ন্
Whole Note or *Purna Maatra*	sa ra ga ma pa qa na	সা রা গা মা পা ধা না
Half Note or *Ardh Maatra*	si ri gi mi pi qi ni	সঃরঃগঃমঃপঃধঃনঃ
Quarter Note or *Siki Maatra*	sI rI gI mI pI qI nI	স॰ র॰ গ॰ ম॰ প॰ ধ॰ ন॰
Meend	w y and x	⏝

Table 1. Notation symbols in Akarmatrik notation system and corresponding English alphabets and symbols in Sangeet-TEX.

a named placeholder which can contain text inside it. The name of a node uniquely identifies it in a figure. To render a single line we create t square tikz nodes and set the name of the node with indices 1 to t. Since, we already have the indices of various symbols, we insert the symbols as text into respective nodes. For completeness, we remove the border from each node so that it looks line a printed composition.

4.1 Rendering of Meend Symbol

For the rendition of the *meend* symbol we use the *draw* tool of tikz and create a bezier curve. The bezier curve is defined by four points: two endpoints (start and end) and two control points that determine how curve it is. We place the first two endpoints at the bottom midpoint of the start and end note of the *meend*. However, in our case, the curve operation is accomplished using angles instead of control points. We provide angles at which the curve should leave the start point and reach the target point. Moreover, we create a *cycle* by drawing another curve just above already drawn curve that reaches the start point from the end point. At last we fill the area created with color so that it looks exactly same as meend.

We have considered another important feature of *meend* symbol is that it is stretchable depending on how far start and end notes are situated. There would be a rendering issue if we try to give same angles to a wide meend as a narrow meend. This would make the wide meend fat in the middle and would provide wrong rendition. We try to adjust the angles proportional to the length of the meend so that it preserves the same height of the symbol.

4.2 Notation used in SangeetTEX

SangeetTEX uses *Swarabitan* font that chooses several English alphabets or the combinations of them to represent each symbol of the Akarmatrik Notation System. The English alphabets are chosen intelligently so that most of the notes resemble phonetically to the music symbol. The notation system uses same letter for both *Sparsh Svar* and *Svar*, but with a smaller size in case of *Sparsh Svar*. In these cases, the captial and smaller version of the same alphabet is used. A comprehensive list for the notes and corresponding English alphabets are given in Table 1. Since, we render the *Meend* in a separate way, we don't use the *Meend* keys (w, y, and x) to represent *Meend* in the commands.

5. DEMONSTRATION OF THE ENGINE

In this section, we try to show snippets of the Latex code and the corresponding rendition of the music sheet. We will also provide the meta data of the song, so that the reader can validate the transcription as required.

In the first example (Table 2), we give several rendition of the compositions with some of the prominent *tāl*s of Indian Classical Music. The simplicity of the engine lies in the fact that it only requires the name of the *tāl* and number of *avartan* of the composition and everything else has been taken care of by the rendering engine. The user can then place the musical symbols according to the notation presented in Table 1. The user only has to take care while entering the music symbols and check the number of symbols is equal to the total number of beats in the *tāl*.

The second example (Table 3) demonstrates the rendition of the variations of the *Meend* symbol in different context and therefore proved to be a better rendition engine than published music sheet. We have also included `\scoreLyricLine` command in the table to render score with lyrics in a composition.

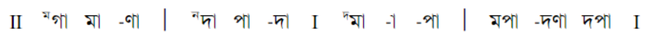
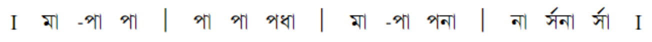
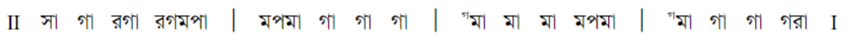
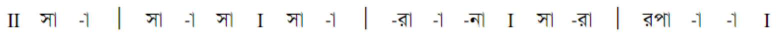
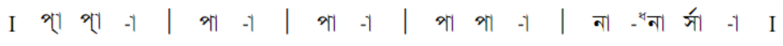
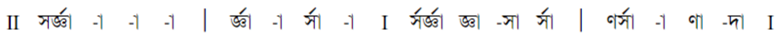
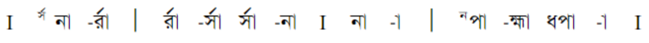
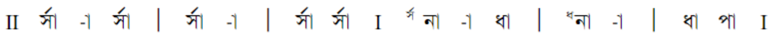
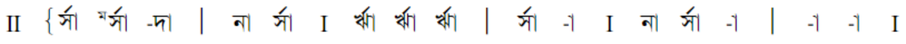
$t\bar{a}l(pattern)$ and [*avartan*]	**Latex Statement and Rendition**	**[Line Index][*Parjaay*][Song Index]**
Dadra (3-3) [2]	`\scoreNewPhrase{dadra}{2}{Mga,ma,-ua,Nda,pa,-da,Dma,-a,-pa,mpa,-dua,dpa}` 	[1][Puja][215]
Iktāl (3-3-3-3) [1]	`\scoreline{iktaal}{1}{ma,-pa,pa,pa,pa,pqa,ma,-pa,pna,na,sfna,sfa}` 	[3][Anusthanic][1]
Teentāl (4-4-4-4) [1]	`\scoreNewPhrase{teentaal}{1}{sa,ga,rga,rgmpa,mpma,ga,ga,ga, Gma,ma,ma,mpma,Gma,ga,ga,gra}` 	[1][Prem O Prakriti][24]
Jhaanp (2-3) [3]	`\scoreNewPhrase{jhap}{3}{sa,-a,sa,-a,sa,sa,-a, -ra,-a,-na,sa,-ra,rpa,-a,-a}` 	[1][Prakriti][25]
Dhamaar (3-2-2-3-4) [1]	`\scoreline{dhamaar}{1}{pha,pha,-a,pa,-a,pa,-a,pa,pa,-a,na,-Qna,sfa,-a}` 	[3][Puja O Prarthana][54]
Kahaarba (4-4) [2]	`\scoreline{kaharba}{1}{-sa,-pa,-pa,-ma,}` 	[1][Swadesh][12]
Shashthi (2-4) [2]	`\scoreline{shashthi}{2}{SFna,-rfa,rfa,-sfa,sfa,-na,na,-a,Npa,-ka,qpa,-a}` 	[4][Prakriti][23]
Teora (3-2-2) [2]	`\scoreNewPhrase{teora}{2}{sfa,-a,sfa,sfa,-a,sfa,sfa, SFna,-a,qa,Qna,-a,qa,pa}` 	[1][Swadesh][17]
Jhampak (3-2) [3]	`\scoreNewPhrase{jhampak}{3}{\{sfa,Msfa,-da,na,sfa,vfa, vfa,vfa,sfa,-a,na,sfa,-a,-a,-a}` 	[3][Puja][166]

Table 2. Rendition of SangeetTEX on different $t\bar{a}l$s having different Maatras along with the reference of the source for validation

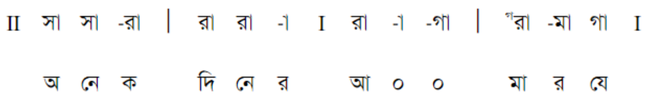
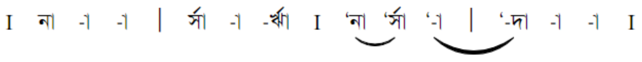
Property	**Latex Statement and Rendition**	**[Line Index][*Parjaay*][Song Index]**
Score and Lyric	`\scoreLyricNewPhrase{dadra}{2}{sa,sa,-ra,ra,ra,-a,ra,-a,-ga,Gra,-ma,ga}{অ,নে,ক,দি,নে,র,আ,০,০,মা,র,যে}` 	[1][Prem][21]
Meend	`\scoreline{dadra}{2}{na,-a,-a,sfa,-a,-vfa,‘na,‘sfa,‘-a,‘-da,-a,-a}` 	[10][Prem][43]

Table 3. Rendition of SangeetTEX on different $t\bar{a}l$s having different Maatras along with the reference of the source for validation

6. CONCLUSIONS

In this paper, we develop a music-sheet rendering engine, SangeetTEX based on Latex document processing application for the rendition of *Rabindra Sangeet* scored on *Akarmatrik Notation System*. SangeetTEX is implemented on the theoretical 2D-matrix model *Swaralipi* with the instructions to structure rows and columns of the every line of the composition. We give details of the class file along with the commands that are responsible for the rendition of score, lyric, and variations of score in the compositions. We give algorithms to calculate the number of cells for each 2D-matrix and the positions for each of the music symbols for easy rendering. We give special emphasis on the rendition of *Meend* symbol since it is one of the primary reasons to adopt such a document preparation tool to prepare print quality music-sheet. Finally, we validate the engine by annotate compositions with different $t\bar{a}l$s and score variations.

Acknowledgments

The *Swarabitan* font as been designed and developed by the author while working in a research project funded by Society for Natural Language Technology Research, Department of Information Technology and Electronics, Govt. of West Bengal, India. The font is freely downloadable from the SNLTR website[4] [10].

7. REFERENCES

[1] V. N. Bhatkhande, *Hindustani sangeet paddhati: bhatkhande kramik pustak*. Sangeet Karyalaya, 1990.

[2] D. Courtney, "North indian musical notation: An overview," https://chandrakantha.com/music-and-dance/i-class-music/n-indian-notation/, accessed: 2022-01-30.

[3] R. Tagore, *Swarabitan*. Visva-Bharati, 1949.

[4] S. Mammen, I. Krishnamurthi, A. J. Varma, and G. Sujatha, "isargam: music notation representation for indian carnatic music," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2016, no. 1, pp. 1–12, 2016.

[5] S. Systems, "Swarshala 4," https://www.swarclassical.com/SwarShala/, accessed: 2022-01-30.

[6] C. Misra, B. Bhattacharya, and A. Basu, "A new framework to preserve tagore songs," *World Digital Libraries-An international journal*, vol. 3, no. 1, pp. 63–72, 2010.

[7] C. Misra, T. Chakraborty, A. Basu, and B. Bhattacharya, "Swaralipi: A framework for transcribing and rendering indic music sheet," 2016.

[8] T. L. Project, *LaTeX2e for class and package writers*, 2006.

[9] "Bengali fonts - bangla fonts - free download," https://www.omicronlab.com/bangla-fonts.html, accessed: 2022-01-30.

[10] "Society for natural language and technology research," https://nltr.itewb.gov.in/, accessed: 2022-01-30.

[4] Swarabitan font can be downloaded by visiting the site `https://nltr.itewb.gov.in`