Kangweon-Kyungki Math. Jour. **14** (2006), No. 1, pp. 35–46

VISUALIZATION OF DISCRETE CONVOLUTION STRUCTURE USING TECHNOLOGY

Keehong Song

Abstract. The concept of convolution is a fundamental mathematical concept in a wide variety of disciplines and applications including probability, image processing, physics, and many more. The visualization of convolution for the continuous case is generally predetermined. On the other hand, the convolution structure embedded in the discrete case is often subtle and its visualization is non- trivial. This paper purports to develop the CAS techniques in visualizing the logical structure in the concept of a discrete convolution.

1. Introduction

The probability distribution of the sum of independent random variables and the weighted moving average can be obtained using the idea of convolution. Smoothing of the periodic data with noise as typically seen in electrical engineering and time series analysis is also a derivative of convolution. Indeed, the concept of convolution is the mathematical basis for a wide variety of applications across the disciplines. The convolution of two functions, denoted by $f * g$, is given by the integration

$$(f * g)(x) = \int_{-\infty}^{\infty} f(u)g(x-u)du.$$

Thus, intuitively, a convolution measures the amount of overlap of one function as it is moving over another function. Basically, it is kind of superimposition as indicated in the German name of convolution, faltung or folding. Then the idea behind convolution is mostly visual

Received December 22, 2005.
2000 Mathematics Subject Classification: 94C15, 97U70.
Key words and phrases: Fourier transform, discrete convolution, Mathematica implementation, Macromedia Flash.

and thus the visualizing ideas of the concept for the continuous case is pretty much predetermined as demonstrated in the website materials for this paper. However, visualizing the ones for the discrete case requires imagination as well as the technological expertise. This paper develops and investigates the methodology of visualization techniques for the discrete convolution structures.

2. Convolution structure in polynomial

Given two polynomials, $a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots$ and $b_0 + b_1 x + b_2 x^2 + b_3 x^3 + \cdots$, it is easy to see that the coefficient list of the product of two polynomials has the convolution structure. Conversely, once the discrete convolution is defined as a computer routine, the coefficient list of the product of two polynomials can be obtained without multiplication. The list of coefficients can be obtained directly from two lists in a similar manner to the situation that one slip of tape is sliding against the other, in which the slip is the metaphor for a numeric sequence. Now the idea of the discrete convolution can be implemented as listed below.

```
convolution[f_,impulse_]:=Module[{leftsize=Length[f],
   rightsize=Length[impulse],i,j},
   Table[Sum[f[[j]]*impulse[[i-j+1]],
   {j,Max[1,i-rightsize+1],Min[i,leftsize]}],
   {i,1,leftsize+rightsize-1}]]
```

Using the Mathematica function, we can find the coefficients of the product of two polynomials without multiplication [8].

```
convolution[Array[a,5],Array[b,5]]//TableForm
```

Similarly, the coefficient list can be visually organized in terms of columns this time and then the outcome is also known as the Pascal triangle.

```
ColumnForm[NestList[convolution[{1,1},#]&,{1,1},9],
   Center]
```

```
{1, 1}
{1, 2, 1}
{1, 3, 3, 1}
```

```
                    {1, 4, 6, 4, 1}
                 {1, 5, 10, 10, 5, 1}
              {1, 6, 15, 20, 15, 6, 1}
           {1, 7, 21, 35, 35, 21, 7, 1}
        {1, 8, 28, 56, 70, 56, 28, 8, 1}
     {1, 9, 36, 84, 126, 126, 84, 36, 9, 1}
{1, 10, 45, 120, 210, 252, 210, 120, 45, 10, 1}
```

3. Convolution in probability

A typical problem in elementary probability theory involving the sum of the eyes of the dice can be solved using the idea of convolution.

```
dice = 1/6{1,1,1,1,1,1}
convolution[dice, dice]
{1/36,1/18,1/12,1/9,5/36,1/6,5/36,1/9,1/12,1/18,1/36}
```

Now we consider the following problem as an extension of the discussion above to appreciate the utility of the idea.

Example 1. Ten one dollar bills, 7 five dollar bills, 2 ten dollar bills, 1 twenty dollar bill are mixed in a bag, find the probability distribution function of the two bills drawn in sequence with replacement and its amount is added together.

This problem can also be handled in a similar manner as the dice problem, which is the idea of discrete convolution shown in the following:

```
coins=1/20{10,0,0,0,7,0,0,0,0,2,0,0,0,0,0,0,0,0,0,1};
convolution[coins,coins]
{1/4,0,0,0,7/20,0,0,0,49/400,1/10,0,0,0,7/100,0,0,0,0,
1/100,1/20,0,0,0,7/200,0,0,0,0,1/100,0,0,0,0,0,0,0,0,
0,1/400}
```

4. Fourier Transform

Now let's see how discrete case can be solved in a continuous domain, where the primary conceptual tool is going to be the Fourier transform. A Fourier series can sometimes be used to represent a function over an

interval. A generalization of the complex Fourier series, the Fourier transform of the convolution of $f(x)$ and $g(x)$ is equal to the product of the Fourier transforms of $f(x)$ and $g(x)$, $F\{f * g\} = F\{f\} * F\{g\}$ [1]. For the case of Mathematica, Fourier transform is defined as

$$\sqrt{\frac{|b|}{(2\pi)^{1-a}}} \int_{-\infty}^{\infty} f(t) e^{ib\omega t} dt.$$

The rationale behind the definition is to mainly incorporate the variations by adjusting the parameters as each discipline defines the mathematical idea differently.

As an example, the characteristic function of a probability density function defined as $\int_{-\infty}^{\infty} e^{ib\omega x} f(x) dx$ is the Fourier transform with both of parameters a and b being set to 1. Here is an example that demonstrates the case where the Fourier transform finds the probability distribution of the sum of the two and three random variables respectively and it's done with simplicity and flexibility(Figure 1, Figure 2).

```
ft=FourierTransform[UnitStep[x]-UnitStep[x-1],x,w,
FourierParameters->{1,-1}]//Expand//Simplify
InverseFourierTransform[Expand[ft^2],w,x,
FourierParameters->{1,-1}]

Plot[%,{x,0,2}]
```

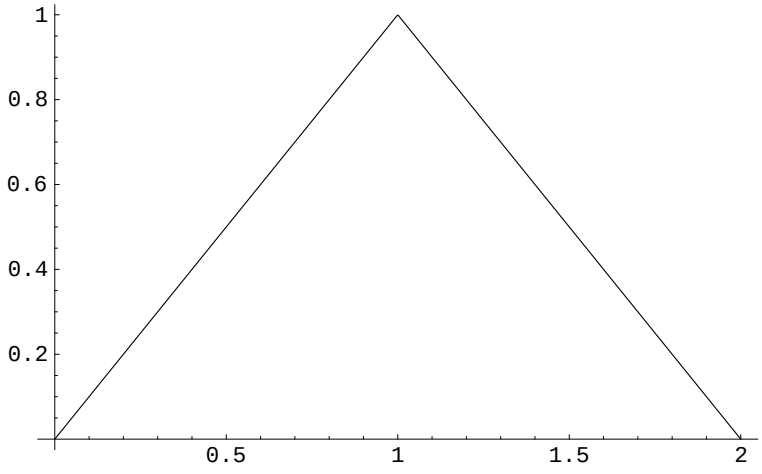

Figure 1.

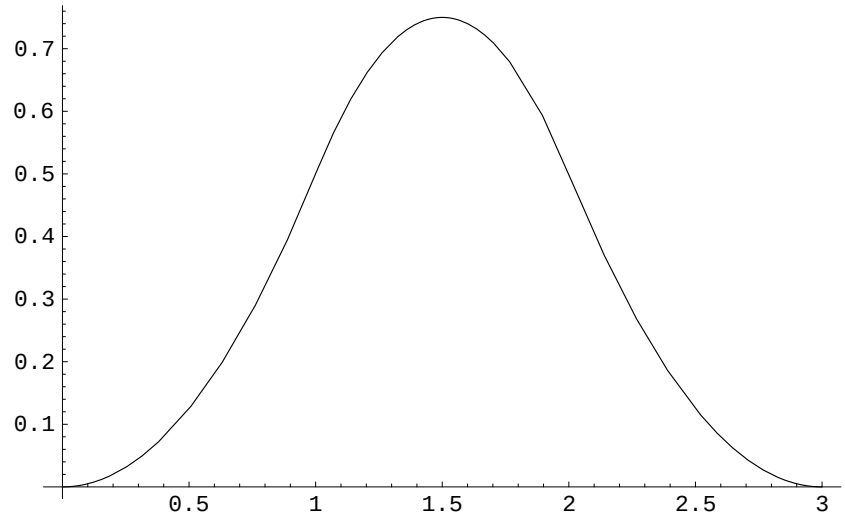

Figure 2.

Now for the case of the sum of three random variables, we are beginning get the bell-shaped distribution as in Figure 2, which will eventually become normal distribution due to the central limit theorem. At this point, a heuristic argument can be made: The convolution of a sharp-edged functions, square and a triangle, produces a smooth, bell-shaped graph, which suggests a general significance of convolution as the smoothing operator.

Revisiting Example 1 discussed earlier, this time with the Fourier transform in mind, we can set the problem up using Dirac delta and then solve using Fourier transform and the convolution theorem in the similar manner as demonstrated below:

```
f=1/20(10 DiracDelta[x-1]+7 DiracDelta[x-5]
+2 DiracDelta[x-10]+1 DiracDelta[x-20])
FT=FourierTransform[f,x,s,FourierParameters->{1,-1}]
result=InverseFourierTransform[Expand[FT^2],s,x,
FourierParameters->{1,-1}]
result /.DiracDelta[___]->1
```

5. Convolution in discrete mathematics

Now we can verify that the same probability distribution is obtained when discrete Mathematica convolution function is used. As for the

examples of the structure of discrete convolution, consider the following set of circumstances.

Case1. The number of legal expressions using the n opening and n closing parentheses.

Case2. The number of paths from (0,0) to ($2n$,0), without touching x-axis during the entire walking process.

Case 3. The number of possible combinations of order of the cars coming in and out of the dead-ended, narrow-alley parking lot.

Case 4. The number of binary trees with n nodes.

Case 5. The number of triangulations given a convex n-polygon.

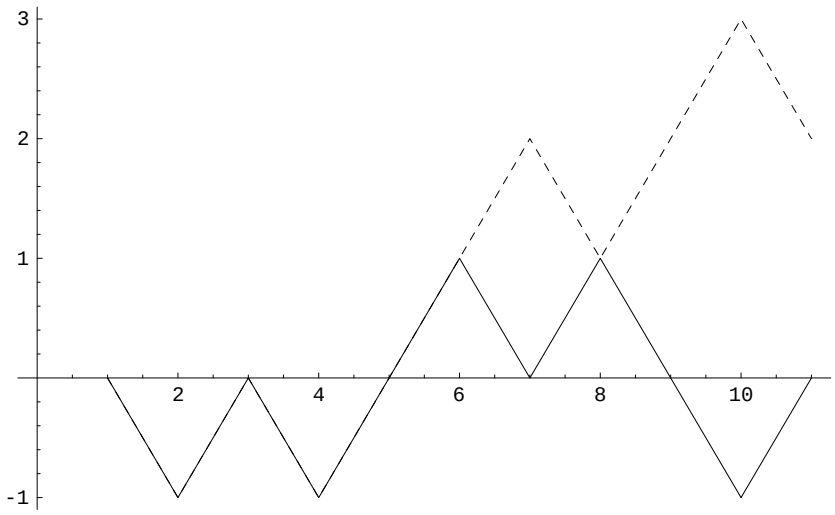

Figure 3.

It is a basic fact in discrete mathematics that the common basis for these problems described above is the idea of convolution([2], [3], [4]). In order to see the one-to-one correspondence among the cases listed above, the basic idea to start would be the reflection principle which gives a combinatoric argument for counting the discrete convolution structure. As an example, consider a sequence, -1, 1, -1, 1, 1, -1, 1, -1, -1, 1, which consists of five -1's and five 1's. As each element 1 represents the ascending step forward in the east-north direction and -1 toward the east-south, the sequence shows the solid line in the diagram shown the Figure 3. Starting from the point of single step forward right after the point where the sequence becomes illegal, the apparent

illegal sequence can be converted into a regular sequence made of $n+1$ elements of 1's and $n-1$ elements of -1's and vice versa.

Now let's first discuss the method of generating binary trees using the fact that the number of binary trees is equal to the Catalan number. The root of the tree can be considered to be a binary tree of size 1, and each of the subtrees on both left and right-hand side is also a binary tree. Thus, we establish the equality among the generating functions $B(z) = left \times root \times right = B(z) \times z \times B(z) = zB^2(z) - B(z) + 1$ which yields the generating function for the convolution $zB^2(z) - B(z) + 1 = 0$.

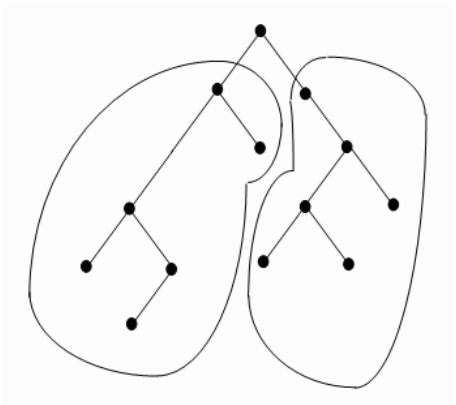

Figure 4.

The generation of the entire set of the binary trees as in Figure 4 would also be accomplished utilizing the idea in the construction of the relevant recurrence relation $b_{n+1} = b_0 b_n + b_1 b_{n-1} + \cdots + b_{n-1} b_1 + b_n b_0$.

a, b, c, d	b, **a**, c, d	b, c, **a**, d	b, c, d, **a**
a, b, d, c	b, **a**, d, c	c, b, **a**, d	b, d, c, **a**
a, c, b, d			c, b, d, **a**
a, c, d, b			c, d, b, **a**
a, d, c, b			d, c, b, **a**

Figure 5.

Now, with the letter **a** in boldface representing the root of the binary tree, the binary tree can be rearranged as the Figure 5 with respect to the position of the root, clearly establishing the relationship between Case 3 and Case 4.

The idea suggests the constructive algorithm for generating the binary trees, iterating through the each element in the set as the root moves from the first position to the last.

```
catalanTree[{}]:={{}}
catalanTree[{a_}]:={{a}}
catalanTree[l_List]:=Module[{m=Rest[l],i,
   fstElem=First[l]},
   (Insert[#,{fstElem},2]&)/@(Flatten[#,1]&)@
   Table[Distribute[{catalanTree[Take[m,i]],
   catalanTree[Take[m,{i+1,Length[m]}]]},List],
   {i,0,Length[m]}]]
```

```
catalanTree[{1,2,3}]
{{{},{1},{{},{2},{3}}},{{},{1},{{3},{2},{}}},{{2},{1},
{3}},{{{},{2},{3}},{1},{}},{{{3},{2},{}},{1},{}}}
```

Here, by flattening the trees above, we get the order in which the cars can go in and out of the narrow-alley parking lot.

```
Flatten /@ %
{1, 2, 3}
{1, 3, 2}
{2, 1, 3}
{2, 3, 1}
{3, 2, 1}
```

Based on the set of binary trees obtained in the manner described above, we can find the edges involved in a triangulation of a convex polygon by way of doing the depth-first traversal. Using the data structure used in the construction of triangulation, we can also get the binary counterpart by connecting the edges as it does the depth-first search of the catalan tree defined above.

Sketch of Triangulation Algorithm.

1. Create the matching set of binary tree for a given convex n-polygon.
2. Do depth-first traversal for each binary tree to find the edges which correspond to the triangulations of a convex n-polygon.

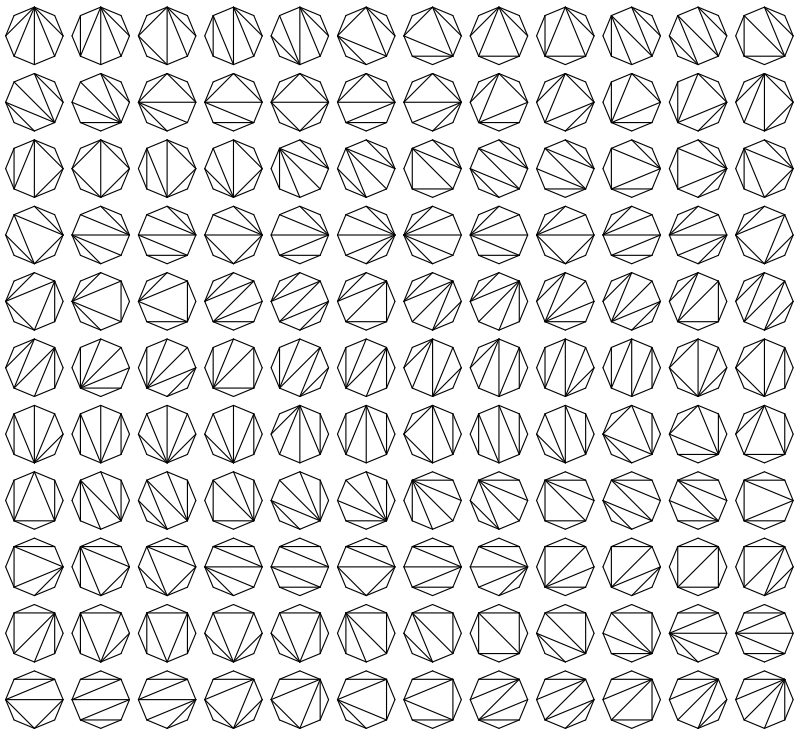

Figure 6.

As a side note, in the Mathematica implementation, the proper use of the module construct, Block instead of Module, is critical as it deals with the dynamic scoping of variables(Figure 6, Figure 7).

```
NodeQ=Function[c,c=!={}];
search[{lc_?NodeQ,{parent_},rc_?NodeQ},{l_,r_}]:=
      (*store the edge info in the array*)
      (*do the left-hand side branches*)
      (*do the right-hand side branches*)
search[{lc_?NodeQ,{parent_},{}},{l_,r_}]:=
      (*do the terminating branch*)
search[{{},{parent_},rc_?NodeQ},{l_,r}]:=
      (*do the terminating branch*)
triangleEdges[n_]:=Block[{storage=branches={},v,m,tree},
      .........
      (* process edge info *)
```

```
Table[search[tree[[m]],{2,1}];
(* do depth-first search *)
(* store branches in an array *)
{m,1,Binomial[2n,n]/(n+1)}];
.........];
```

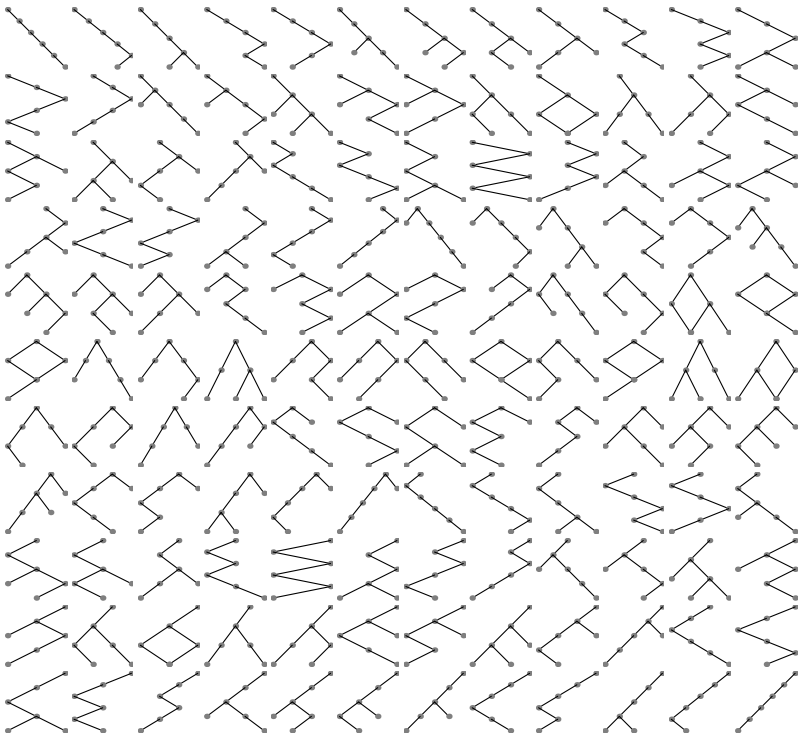

Figure 7.

6. Related Web technology and the future research

Often the visualization of mathematical concepts often requires refined control over the graphical outputs for various reasons. For that need, the computer algebra system is limited as the visual problem solving tool and the specialized multimedia solution is called for.

As seen in the example of convex n-polygon triangulation, the graphical output becomes easily too excessive for each triangulation to be identified and controlled. Figure 8 shows a Flash movie example where the individual item can be stored in a scrollable object for further ma-

nipulation. In that situation, the computer algebra system hidden in the background needs to communication with the web objects in a browser in a timely manner.

The author of this paper developed an ActiveX control that allows users to harness the computational engine of Mathematica from within the web objects such as the Flash movies and the browser components. The idea of multimedia user-interface for CAS can be easily modified to apply in more complex system configuration such as .NET and Java platform([4], [5], [6], [7]).

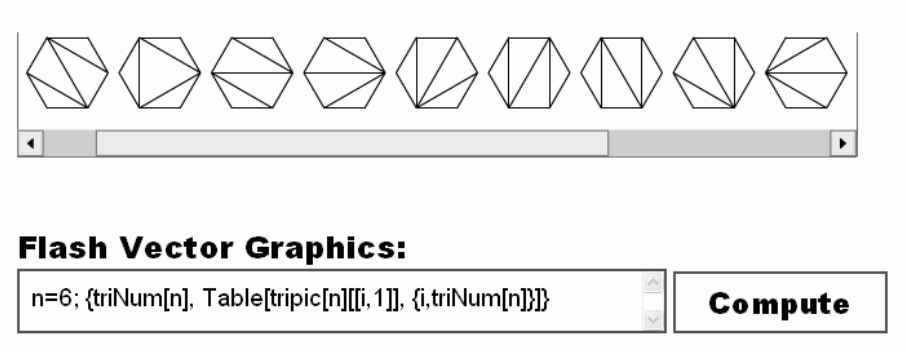

Figure 8.

Programming Files

1. The mathematica file containing the programs for this paper is downloadable from:

http://home.pusan.ac.kr/∼math/research/convolution_paper.zip

2. Visualization and animation for this paper can be viewed at the following url:

http://mathematica.co.kr/mathcomm/31_concepts/convolution

References

1. R. Bracewell, *The Fourier transform and its applications, 3rd ed.*, McGraw-Hill, 1999.
2. S. Even, *Graph Algorithm*, Computer Science Press, 1979.
3. L. Lovász, *Combinatorical Problems and Exercises, 2nd ed.*, North-Holland, 1993.
4. S. Skiena, *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Addison-Wesley, 1990.

5. K. H. Song, *Flash-Enabled User Interface for CAS*, Internet Accessible Mathematical Computation, a Workshop at ISSAC 2003, Drexel University, Philadelphia, PA, USA.
6. K. H. Song, *Developing Computational Web Animation - using Flash and .NET Technology*, Internet Accessible Mathematical Computation, a Workshop at ISSAC 2004, University of Cantabria, Santander, Spain.
7. K. H. Song, *Multimedia User-Interface for CAS*, ASCM 2005, KIAS, Korea.
8. S. Wolfram, *Mathematica: A System for Doing Mathematics by Computer*, Addison-Wesley, 2004.

Department of Mathematics Education
Pusan National University
Pusan, 609–735, Korea
E-mail: khsong@pusan.ac.kr