AMO - Advanced Modeling and Optimization, Volume 5, Number 3, 2003

Connectivity Analysis of a Massively Parallel Architecture for Multiple Path Planning [1]

L. A. Reibling [2]
Department of Computer Science
Azusa Pacific University, Azusa, CA, USA

Abstract

A method is presented for defining a variable connectivity of a massively parallel architecture. This variable connectivity permits an increased amount of interconnection in order to reduce the convergence time to stable solutions while preserving the accuracy of the results. The method of parallel summation for a finite-difference approximation is described and results are discussed.

Keywords: multiple path planning, route planning, massively parallel architecture, connectivity analysis

I. Introduction

A massively parallel architecture for computing multiple paths for route generation systems has been developed [1]. The architecture is based upon a massively parallel approach to computing a numerical solution [2, 3, 4] to a partial differential equation (Laplace's equation) which describes the electrostatic potential field ϕ over a non-uniformly conductive medium. This equation is found in electrostatics theory [5, 6] and was shown to be a useful model for generating the multiple path solutions for the route generation problem [7].

The massively parallel architecture is an artificial neural network which computes the numerical solution to Laplace's equation. This is accomplished by assigning values to the synaptic weights of the network which correspond to the coefficients [8, 9] of a finite difference approximation. A rectangular grid is formed over the problem space. The 5-point

[1]Support provided through Independent Research and Development project funding from Smiths Industries Aerospace & Defense Systems, Inc., and additional support provided through an Accomplished Scholar's Award grant funded by Azusa Pacific University.

[2]Correspondence: Lyle A. Reibling, Department of Computer Science, Azusa Pacific University, 901 E. Alosta Ave., Azusa, CA 91702 USA, Email: reibling@apu.edu

approximation equation for the potential field ϕ at grid location P is

$$U_P = \frac{1}{4}\left[\left(1+\frac{C_x}{2}\right)U_{Q_1} + \left(1+\frac{C_y}{2}\right)U_{Q_2} + \left(1-\frac{C_x}{2}\right)U_{Q_3} + \left(1-\frac{C_y}{2}\right)U_{Q_4}\right] \quad (1)$$

where σ is the inverse cost function, $C_z = \sigma_z/\sigma$, and $Q_i, i = 1,2,3,4$ are the four nearest neighbors of the point P in the grid. For the neural network implementation, the neuron input function $u_{i,j}$ is defined as

$$u_{i,j} = \frac{1}{4}\left[(1+\frac{\sigma_x}{2\sigma})v_{i+1,j} + (1+\frac{\sigma_y}{2\sigma})v_{i,j-1} + (1-\frac{\sigma_x}{2\sigma})v_{i-1,j} + (1-\frac{\sigma_y}{2\sigma})v_{i,j-1}\right] \quad (2)$$

The non-linear neuron output function $v_{i,j}$ is defined as

$$v_{i,j} = \begin{cases} -V_{ref} & : \quad u_{i,j} < -V_{ref} \\ u_{i,j} & : \quad -V_{ref} \leq u_{i,j} \leq +V_{ref} \\ +V_{ref} & : \quad u_{i,j} > +V_{ref} \end{cases} \quad (3)$$

II. Variable Connectivity Architecture

A method to implement a variable connectivity of the architecture is desired. The concept of variable connectivity means that rather than solving the partial differential equation with each node of the grid using inputs from only four immediate neighboring nodes of the 5-point approximation, additional nodes can be connected to improve the convergence rate. The hypothesis is that increased connectivity will lower the convergence time of the architecture. This increase in architecture complexity should not outweigh the benefits obtained in convergence time, or accuracy of the results.

The Taylor-series based approach (described in Appendix B of [1]) was used to develop template definitions for increasing sizes of templates. This was an intuitive approach for increased connectivity because it had a sound mathematical basis for defining approximation templates. Given a template pattern, it seemed straightforward to enlarge the pattern from five points to templates with nine, 13, 25, or 41 points, for example. Unfortunately, this approach failed to provide the accuracy and convergence results which were anticipated. Increasing connectivity by using larger templates actually decreased the convergence rate. The next section gives the detailed results of this approach and discusses why it failed to produce the anticipated improvements in convergence rate.

With the failure of this approach, a new method was needed to achieve the desired variable connectivity in the artificial neural network. A solution was found in a *parallel summation* method which uses the 5-point finite difference approximation and expands upon the Euler method of integration into multidimensional templates. The method is described first for one dimension, both unidirectional, and bidirectional, and then extended to two dimensions.

Considering only one dimension, a first-order differential equation can be solved numerically using the Euler method. The Euler method can be viewed as a recurrence relation. In expanding the recurrence relation for each successive solution point, all of the summations arrived at by the recurrence relation can be seen. Considering the terms of the summation for each solution point, and comparing to the total summation of previous solution points, previous solution points for these terms can be substituted in a systolic fashion. There are many ways to partition these summation terms, which in effect varies the parallelism of the architecture and thus the connectivity of the neural network. The following discussion defines the equations and illustrates the architecture of this approach. A good approximation is retained over the entire problem space, because the fundamental approximation to the derivative is unchanged. The variable connectivity is accomplished by collecting computations of the numerical approximation into different groupings which may be implemented in parallel without altering the basic approximation to the derivative. The connectivity is variable in a straightforward manner, with computational speed results apparent from the inherent serial or parallel combinations of the architecture. These connection equations (summation formula) are developed from first-order one-dimensional to two-dimensional for their use in the architecture.

Consider the differential equation

$$\frac{d}{dx}f(x) = g(x), \quad \text{with initial condition} \quad f(x_0) = f_0 \tag{4}$$

Substitute the numerical approximation for the derivative and obtain

$$\frac{f(x_{i+1}) - f(x_i)}{\Delta x_i} = g(x_i) \tag{5}$$

Solving for $f(x_{i+1})$ results in the Euler forward integration equation[4]

$$f(x_{i+1}) = f(x_i) + \Delta x_i g(x_i) \tag{6}$$

Let $h = \Delta x_i$ be used as the grid size in a parallel architecture.

Recursive application of Equation 6 can be used to solve for any value $f(x_n)$:

$$\begin{aligned} f(x_0) &= f_0 \\ f(x_1) &= f(x_0) + hg(x_0) \\ f(x_2) &= f(x_1) + hg(x_1) \\ &\vdots \\ f(x_n) &= f(x_{n-1}) + hg(x_{n-1}) \end{aligned}$$

This concept is illustrated in Figure 1.

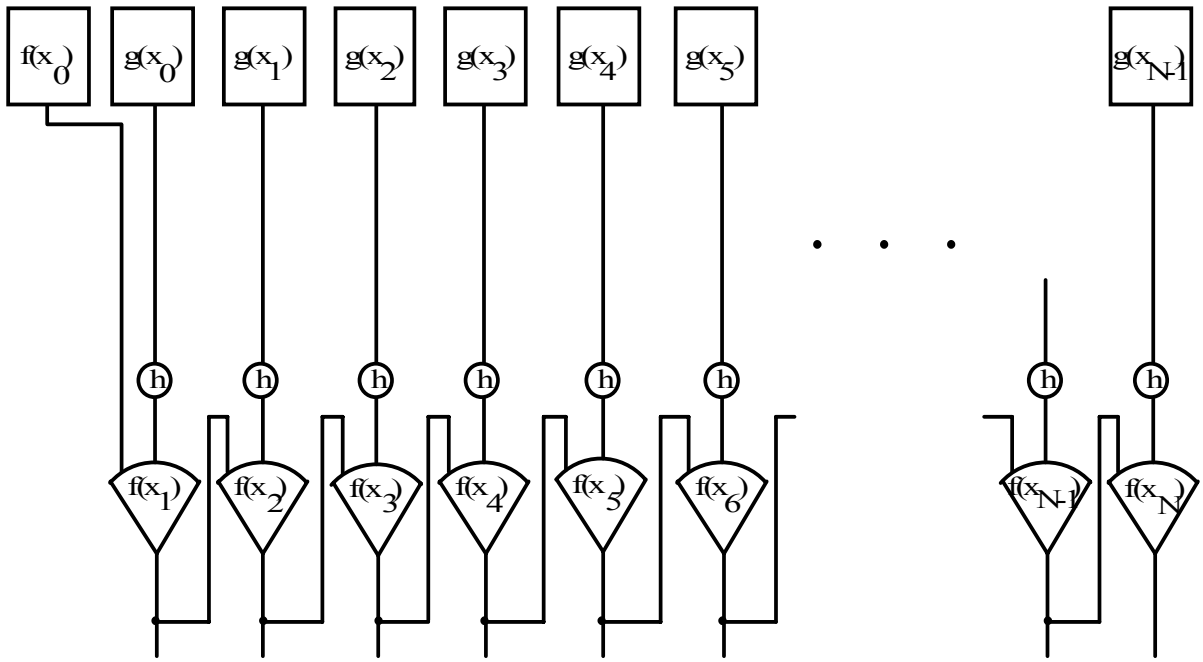

Figure 1: *Spatial Illustration of the Euler Integration Concept*

If the definition of the $f(x_i)$ terms on the right hand side of the above set of equations are expanded, this gives

$$\begin{aligned} f(x_0) &= f(0) \\ f(x_1) &= f(0) + hg(x_0) \\ f(x_2) &= f(0) + h\left[g(x_0) + g(x_1)\right] \\ f(x_3) &= f(0) + h\left[g(x_0) + g(x_1) + g(x_2)\right] \end{aligned}$$

For the nth term this is

$$f(x_n) = f(0) + \sum_{i=0}^{n-1} hg(x_i) \tag{7}$$

The summation in Equation 7 can be split into two summations:

$$f(x_n) = f(0) + \sum_{i=0}^{n-k-1} hg(x_i) + \sum_{j=n-k}^{n-1} hg(x_j) \tag{8}$$

The first two terms of Equation 8 is $f(x_{n-k})$, so this becomes

$$f(x_n) = f(x_{n-k}) + \sum_{j=n-k}^{n-1} hg(x_j) \tag{9}$$

The connectivity will be shown later to be the fan-in $K = k + 1$ to the neuron. Equation 9 gives a definition for the *variable connectivity* in the parallel summation for the Euler

integration. This variable connectivity encompasses varying degrees of parallelism via the fan-in, and has the effect of varying the amount of serial delay in computing its results. The first term of Equation 9 is the serially cascaded computation. The second term is the parallel (in addition to the serial input) computation. Of course, it will have a valid output only after the cascaded serial summation delay. The connectivity is derived as follows. Equation 9 is spatially constructed for parallel summations. So how many summations are there? The last term has $(n-1)-(n-k)+1$ addends, for a total of k inputs to the parallel summer. Including the input for the first term, this is $k+1 \equiv K$ total inputs to the summer.

Use of Equation 9 for specifying the connectivity of the network architecture will be illustrated. This illustration uses ten points of the function ($n = 10$).

Figure 2 illustrates the summation equation with a fan-in of 2, which is

$$f(x_n) = f(x_{n-1}) + hg(x_{n-1}) \tag{10}$$

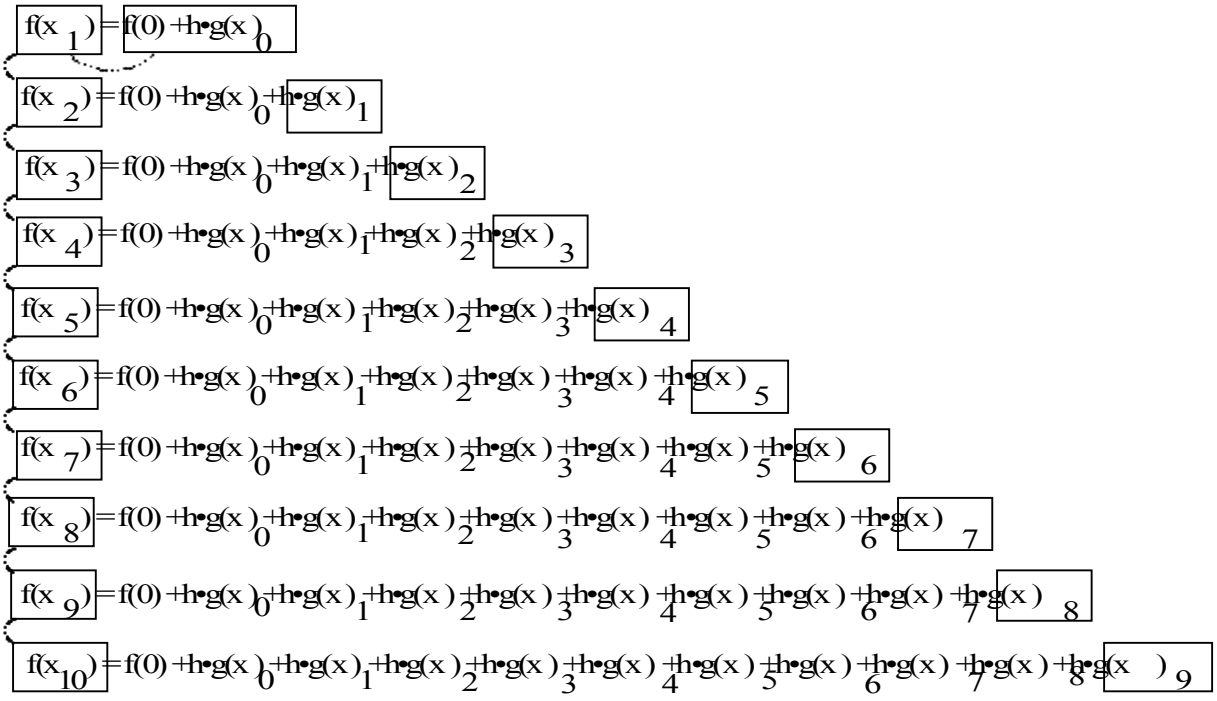

Figure 2: *K=2 Connectivity of the Parallel Summation Equations*

Each output on the left side of the equations in the figure picks up two terms, the previous output term, and the last input term on the right hand side of the equations. This architecture, as shown in Figure 3 is parallel in space, but serial in time. It has the minimum number of connections, and the maximum time to settle because of the cascaded ripple wave computations.

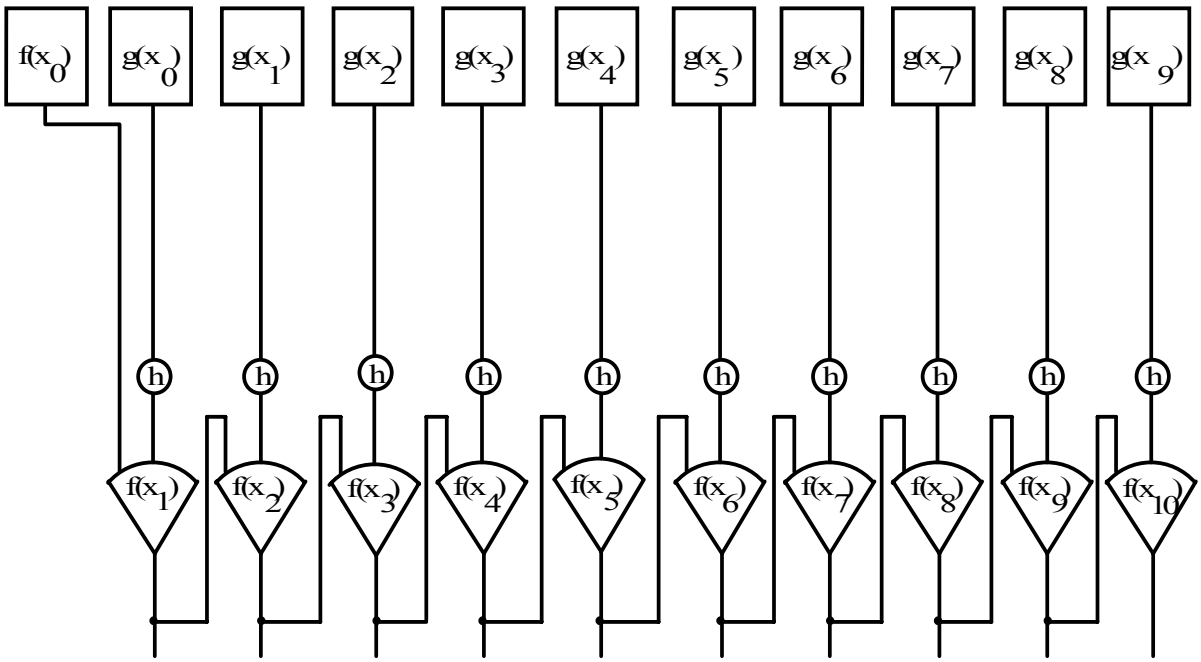

Figure 3: *K=2 Interconnections for the Parallel Summation Architecture*

The next illustration is Figure 4 for the fan-in of 3. Its equation is

$$f(x_n) = f(x_{n-2}) + \sum_{i=n-2}^{n-1} hg(x_i) \tag{11}$$

Each output on the left side of the equations of the figure picks up three terms, the second output term preceeding this output, and the last two input terms on the right hand side. This architecture shown in Figure 5 is the next step in adding more connectivity, which has a corresponding decrease in the time required for the cascaded serial computational wave.

In Figure 6 the fan-in for the complete connectivity of $n+1$ is shown. The equation describing this is

$$f(x_n) = f(0) + \sum_{i=0}^{n-1} hg(x_i) \tag{12}$$

Here no serial terms are picked up from the cascade ripple wave. All of the input terms are from the right hand side of the equation. This architecture is parallel in space, and also parallel in time. This has the maximum number of connections and minimum time because there is no ripple wave for the computation due to no cascading structure. The architecture is shown in Figure 7.

The construction of the bidirectional parallel summation is the next step. Since the objective of the parallel summation architecture is to have a variable amount of connections for a multidimensional problem space, the summation technique must not have a unidirectional

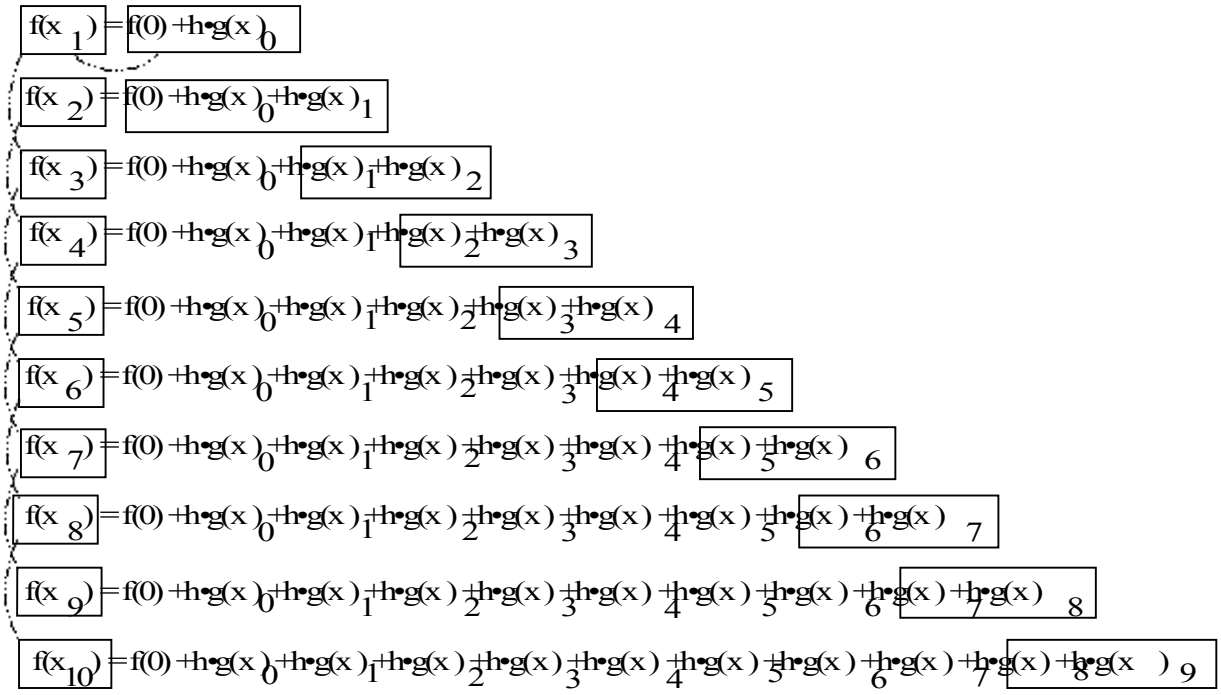

Figure 4: *K=3 Connectivity of the Parallel Summation Equations*

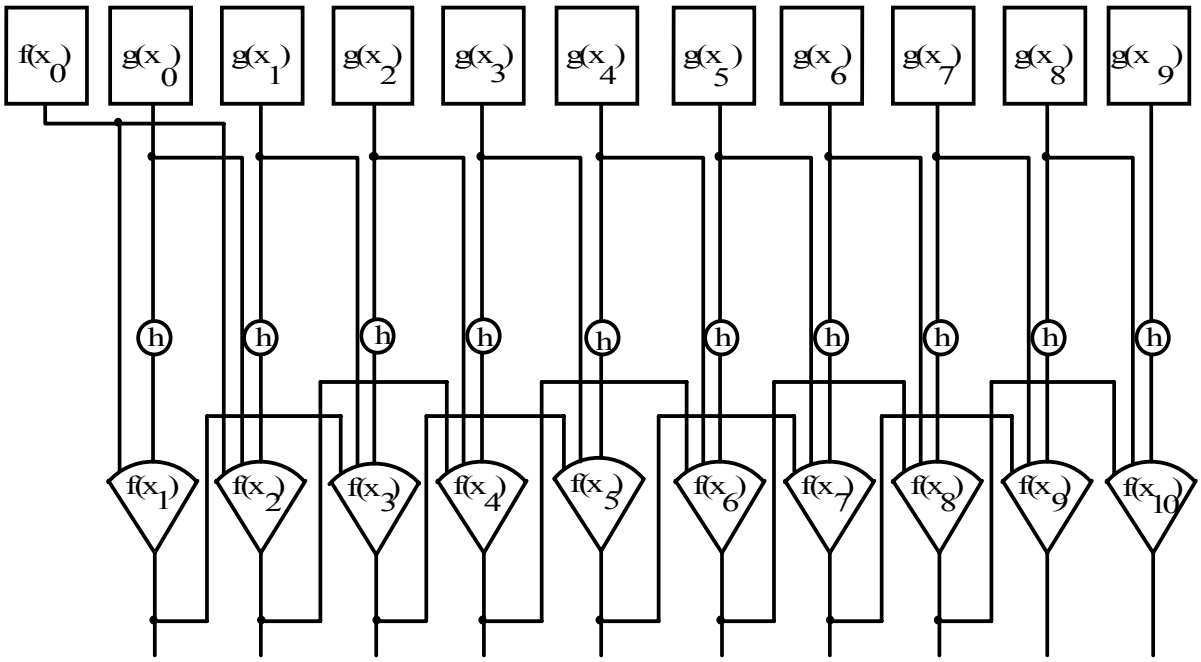

Figure 5: *K=3 Interconnections for the Parallel Summation Architecture*

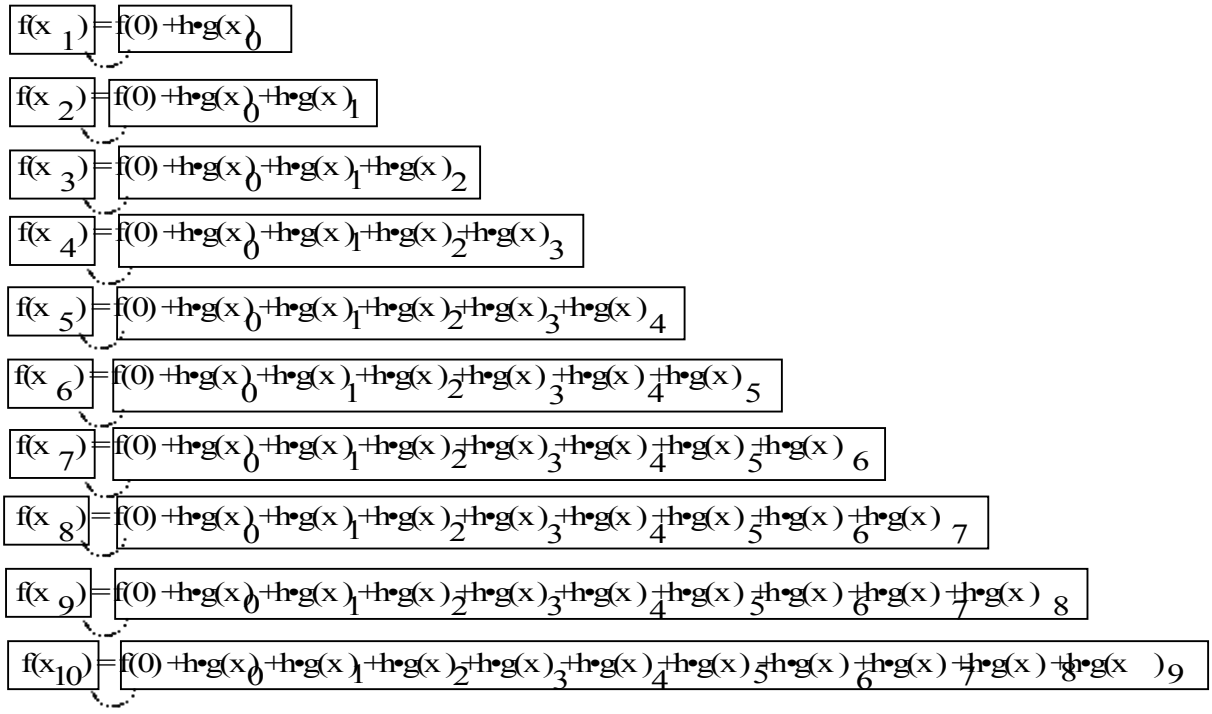

Figure 6: *K=n+1 Connected Grouping of the Parallel Summation Equations*

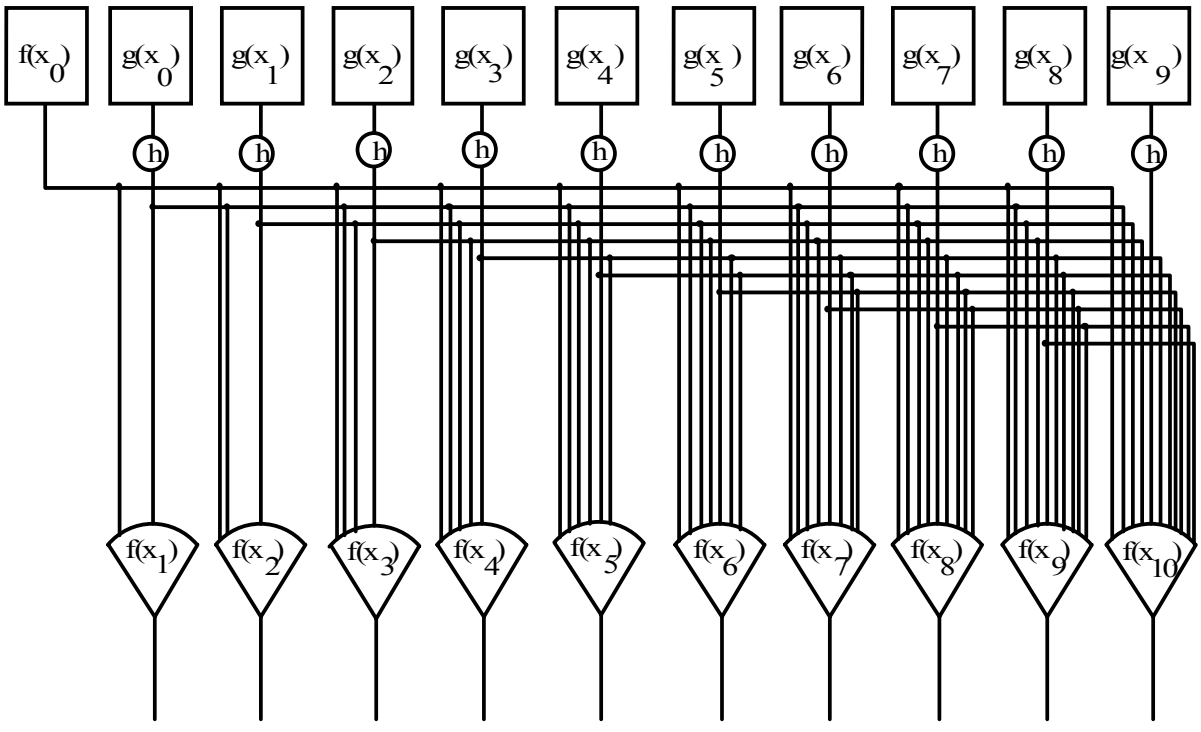

Figure 7: *K=n+1 Interconnection for the Parallel Summation Architecture*

flow to the computation. This requirement is necessary because the local connection architecture must be robust enough to support the processing of data arriving from any direction within the architecture. Therefore a bidirectional parallel summation is needed to eliminate the unidirectional computation flow of the previous architecture.

To develop a bi-directional formula, the backward difference formula first needs to be derived. Begin again with the differential equation

$$\frac{d}{dx}f(x) = g(x), \quad \text{with initial condition} \quad f(x_{N+1}) = f_{N+1} \tag{13}$$

The backward difference approximation to Equation 13 is

$$\frac{f(x_{i+1}) - f(x_i)}{\Delta x_i} = g(x_{i+1}) \tag{14}$$

Letting $h = \Delta x$ be the grid size, and solving Equation 14 for $f(x_i)$ gives

$$f(x_i) = f(x_{i+1}) - hg(x_{i+1}) \tag{15}$$

The general expression then is

$$f(x_j) = f(x_{j+k}) - \sum_{i=j+1}^{j+k} hg(x_i) \tag{16}$$

As should be expected, this architecture compared with Figure 1 is inverted along the single dimension.

Now a central difference formula is derived by combining the forward and backward difference formulas.

Recall the forward difference expression as

$$f(x_j) = f(x_{j-k}) + \sum_{i=j-k}^{j-1} hg(x_i) \tag{17}$$

and the backward difference is Equation 16. Adding these together and solving for $f(x_j)$ yields

$$f(x_j) = \frac{f(x_{j+k})}{2} + \frac{f(x_{j-k})}{2} + \sum_{i=j-k}^{j-1} \frac{h}{2}g(x_i) - \sum_{i=j+1}^{j+k} \frac{h}{2}g(x_i) \tag{18}$$

Figure 8 shows the architecture for Equation 18 with $k = 1$.

There is a pattern to the increasing parallelism (k) in the summation equations and the architectural configuration. This is illustrated as shown in Figure 9 where the right side of the figure shows the pattern after increasing the parallel connections to the example on the left. When increasing the parallelism of the summation, the input to neuron C is increased. This

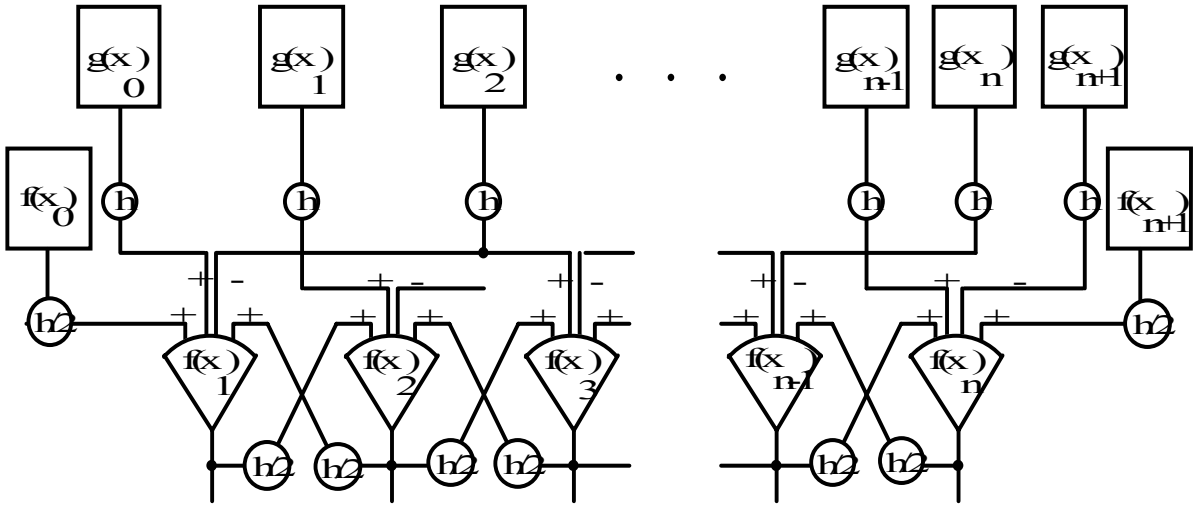

Figure 8: *Illustration of the Central Difference Parallel Summation Architecture*

is done by disconnecting the output O_B of neuron B, and adding connections from the input I_B of neuron B directly as additional inputs to neuron C. This will include connecting the output O_A of neuron A to the input of neuron C. Thus the pattern of increasing parallelism in the summation equations incrementally bypasses neurons by adding connections from their inputs directly to those neurons which they were connected to by their original output.

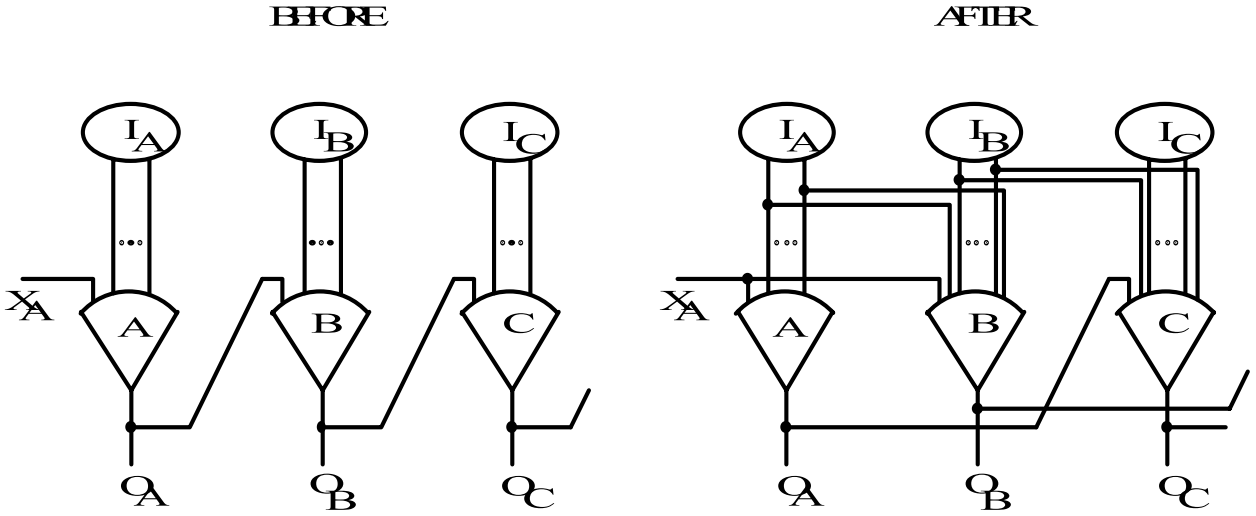

Figure 9: *Recursive Expansion of the Parallel Summation Architecture*

$$O_A = A(I_A, X_A) \tag{19}$$

$$O_B = B(I_B, O_A) \tag{20}$$

$$O_B = B(I_B, I_A, X_A) \tag{21}$$

$$O_C = C(I_C, O_B) \tag{22}$$

$$O_C = C(I_C, I_B, O_A) = C(I_C, B(I_B, O_A)) = C(I_C, O_B) \tag{23}$$

Thus, the recursive expansion by substitution. This substitution is valid since all neuron outputs converge to the linear portion of the neural output function[7]. Figure 10 illustrates the architecture of this observation for the central difference formula.

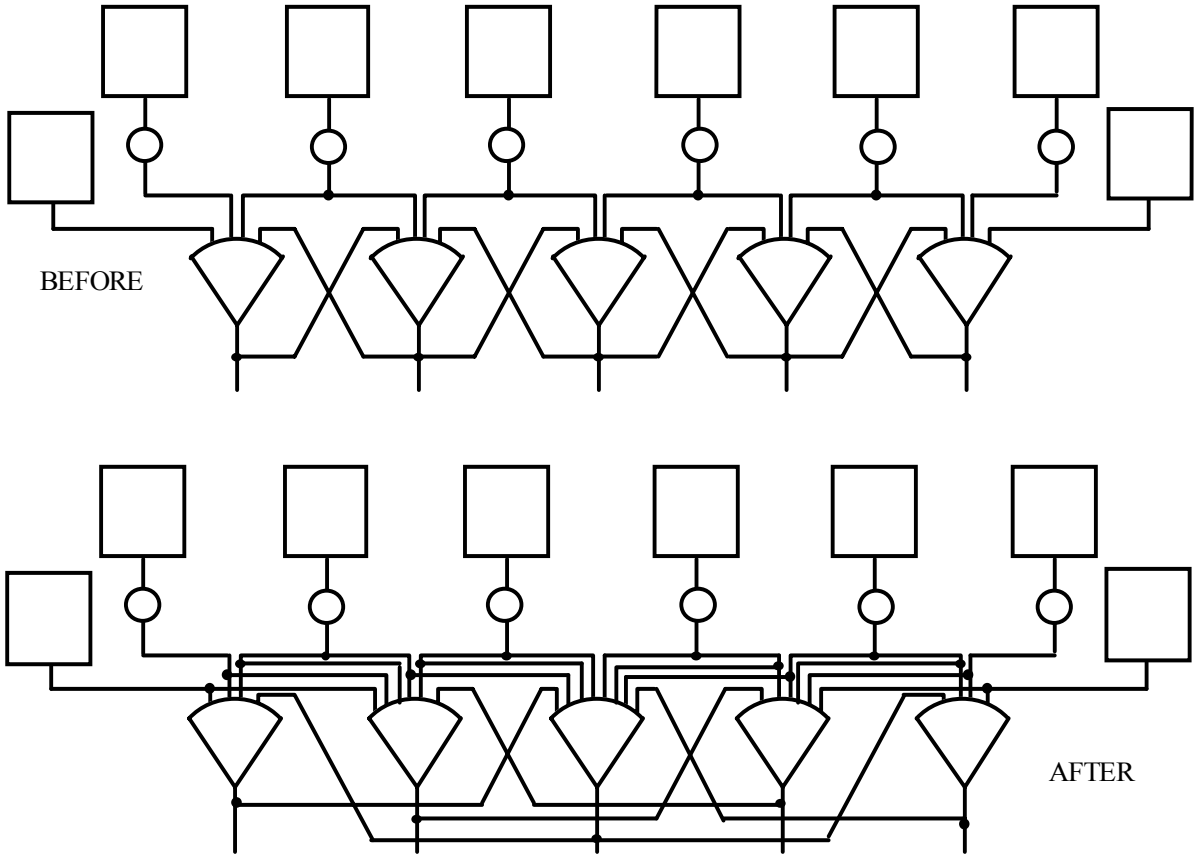

Figure 10: *Expansion of Central Difference Approximation for Parallel Summation*

With the development of the central difference formula for bidirectional parallel summation, and the observation of the recursive expansion, the final step in the construction of multidimensional parallel summation can be accomplished. Using the five point approximation template as a *kernel*, the template can be "unfolded" as many times as desired in a recurrent fashion, obtaining a larger connection neighborhood, which takes into account the local $C_x(x,y)$ and $C_y(x,y)$ in the solution. Different kernels generate different connection patterns.

The synaptic weight expressions for the model can be written as the following equations

$$w_x^+(i,j) = \frac{1}{4}\left[1 + \frac{\sigma_x}{2\sigma}(i,j)\right] \tag{24}$$

$$w_x^-(i,j) = \frac{1}{4}\left[1 - \frac{\sigma_x}{2\sigma}(i,j)\right] \tag{25}$$

$$w_y^+(i,j) = \frac{1}{4}\left[1 + \frac{\sigma_y}{2\sigma}(i,j)\right] \tag{26}$$

$$w_y^-(i,j) = \frac{1}{4}\left[1 - \frac{\sigma_y}{2\sigma}(i,j)\right] \tag{27}$$

The general equation for the neuron input function is

$$u(i,j) = w_x^+(i,j)v(i+1,j) + w_x^-(i,j)v(i-1,j) + w_y^+(i,j)v(i,j+1) + w_y^-(i,j)v(i,j-1) \tag{28}$$

which upon recursive expansion (for $k = 2$) is

$$\begin{aligned}
u(i,j) &= w_x^+(i,j)[w_x^+(i+1,j)v(i+2,j) + w_x^-(i+1,j)v(i,j) \\
&+ w_y^+(i+1,j)v(i+1,j+1) + w_y^-(i+1,j)v(i+1,j-1)] \\
&+ w_x^-(i,j)[w_x^+(i-1,j)v(i,j) + w_x^-(i-1,j)v(i-2,j) \\
&+ w_y^+(i-1,j)v(i-1,j+1) + w_y^-(i-1,j)v(i-1,j-1)] \\
&+ w_y^+(i,j)[w_x^+(i,j+1)v(i+1,j+1) + w_x^-(i,j+1)v(i-1,j+1) \\
&+ w_y^+(i,j+1)v(i,j+2) + w_y^-(i,j+1)v(i,j)] \\
&+ w_y^-(i,j)[w_x^+(i,j-1)v(i+1,j-1) + w_x^-(i,j-1)v(i-1,j-1) \\
&+ w_y^+(i,j-1)v(i,j) + w_y^-(i,j-1)v(i,j-2)]
\end{aligned} \tag{29}$$

Figure 11 shows the first two expansions for a 5-point mesh architecture. In this manner, new connection weight values are defined. This expansion continues until the desired amount of connectivity is achieved. The maximum connectivity is obtained when the expansion for each node has recursively used all other nodes. This will scale directly with the length of the largest dimension of the multidimensional space.

III. Discussion of Results

Connectivity curves show the convergence of the Taylor series based templates and the parallel summation technique templates at various degrees of connectivity in the neural network. These curves show that the parallel summation technique templates converge faster with higher connectivity.

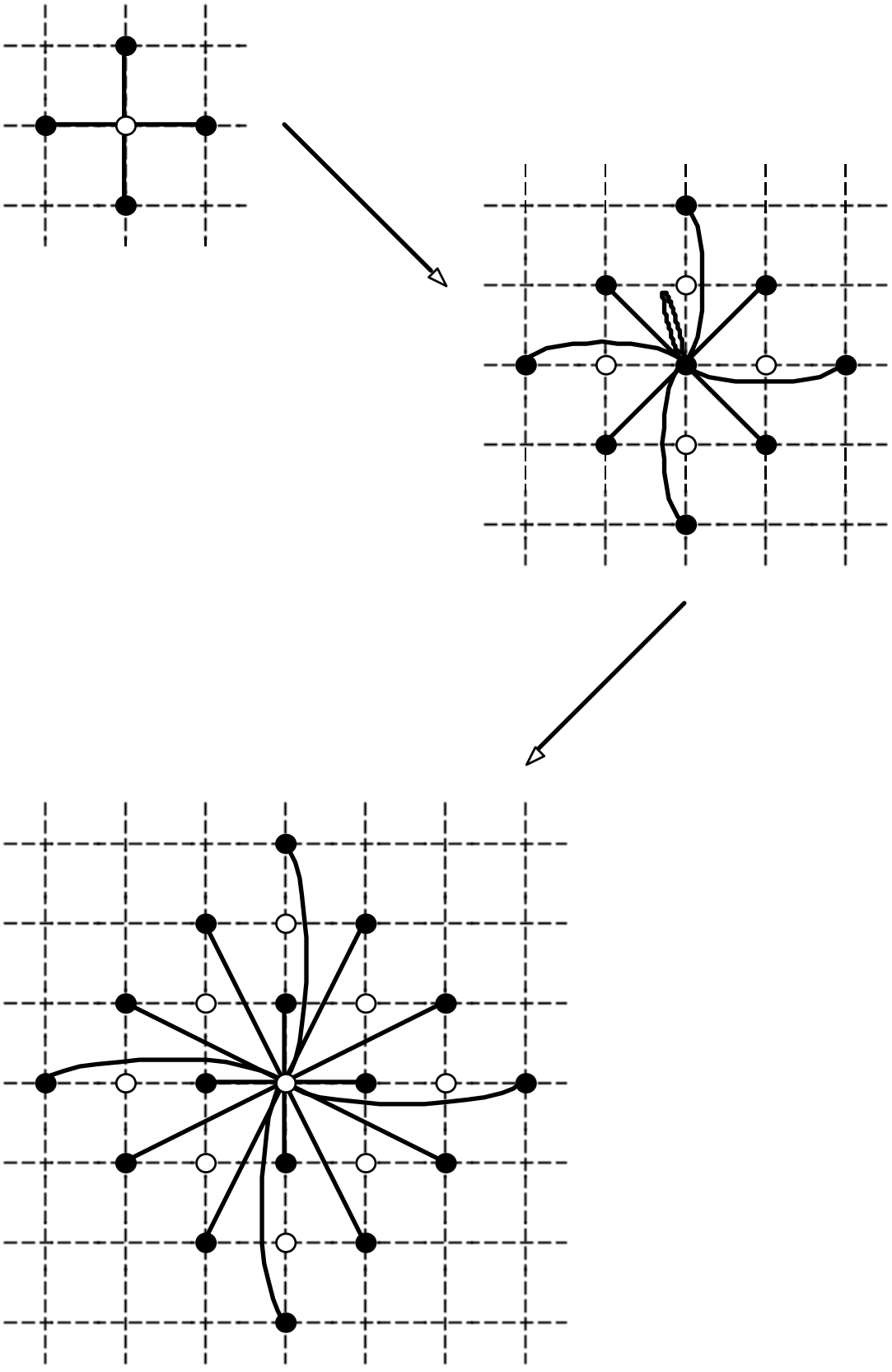

Figure 11: *Two Expansions of the 5-pt. Kernel Approximation Template*

Table 1: *Neural Network Simulation Iterations Using Taylor Series Templates*

Template Size	Cycles per Case		
	A	B	C
5 point	4,000	3,484	5,460
13 point	3,647	2,251	4,015
25 point	3,621	2,612	4,518
41 point	6,947	15,000	10,140

Several test cases were used to experiment with the architecture simulations in this research and are described in [1]. Table 1 shows the number of cycles for each simulation run of the neural network using the templates generated using the Taylor Series method. These runs were executed on a VAX 8600 until the convergence input parameter was satisfied at the end of each iteration of the simulation. The sum-squared difference of all neuron outputs is checked against the input parameter to test for convergence. For the simulation runs, the convergence input parameter was set to 0.001.

In each case, it can be seen that fewer cycles of the parallel algorithm are required for convergence using the 13 point template over the 5 point template. While this improvement continues marginally for the 25 point template of Case A, in the other cases the number of cycles to achieve convergence increased. Continued expansion to the 41 point template resulted in a further, and rather significant degradation in convergence for all cases. Table 2 contains data for the final minimum cost path value at the end of the algorithm execution. Each larger template failed to provide more accurate (smaller) cost values in each case. In fact, the cost values increase with the larger templates using the Taylor series method. The results reported here illustrate the failure of the Taylor series template method which

Table 2: *Least Cost Path Values Using Taylor Series Templates*

Template Size	Path Cost per Case		
	A	B	C
5 point	0.329	7.672	0.840
13 point	0.331	7.599	0.937
25 point	0.331	7.631	1.006
41 point	0.333	7.826	1.068

was reported in the previous section and led to the development of the parallel summation template method.

In this research, over 135 simulations were run and their results were plotted. These plots show the value of the minimum cost path found by the neural network during each cycle

of the run. This was repeated for the more connected pattern, and each run was plotted for a particular test case. A family of curves results which shows the computed cost over computation time for the varying connectivity. These plots show the impact of increased connectivity on the quality of the computation results. Figure 12 illustrates the minimum cost of successive algorithm iteration for the four increasingly complex Taylor series template connectivity patterns. Upon close inspection of the plots, the desired result of improvement from using a more complex template connectivity pattern does not occur. Indeed, as the template connectivity is increased, there is actually a decrease in the convergence, exhibited by a higher final cost of the minimum cost path.

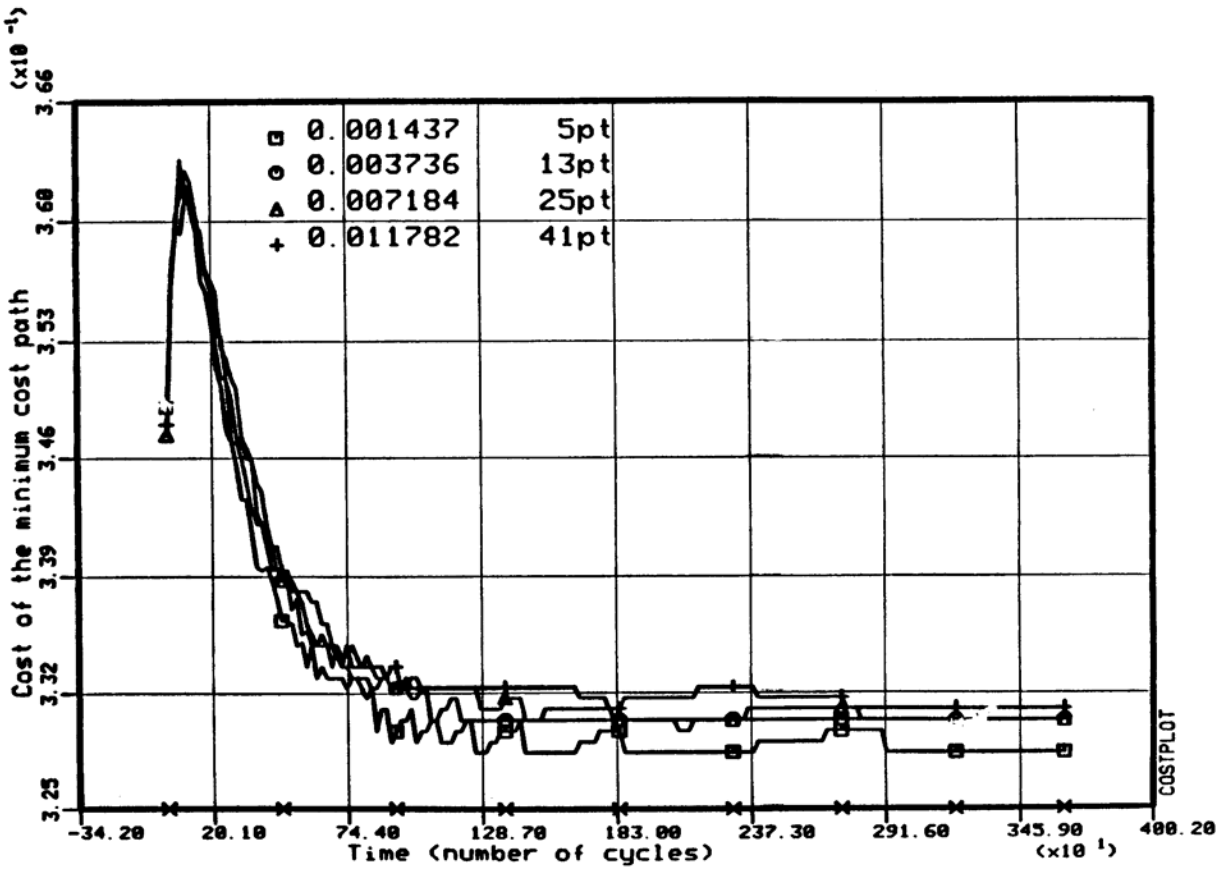

Figure 12: *Least Cost Path Connectivity Curves Using Taylor Series Templates*

However, these connection patterns are very small on the scale of connectivity. In order to get more significant results at larger connectivity, much larger templates are required. Additional constraints are added to convert the template coefficient equations from an underdetermined system. These constraints are intuitive, but yet somewhat ad hoc, trying to account for behavior farther from the approximation point at the center of the template.

As the connectivity increases with the larger templates, it was hoped that the cost plot

could be predicted to drop sooner for each increase in connectivity, reaching the minimum cost in faster time. The larger templates use more of the network node outputs. As changes to node outputs occur in a remote region of the network more distant from a local node of interest, these larger templates facilitate propagation of the remote changes to each local node's output computation. However, plots of the simulation results show that the larger templates are not providing a better approximation. An examination of Figure 12 reveals that the higher connective templates are not, in general, converging faster.

The template defines the differential operator for the approximation. Expansion of the template would seem to be counter to the definition of a derivative in the limit, which would lead to contracting the template instead of expanding the template. As the cost function varies more, less accurate approximations appear with the larger templates, and argues for smaller templates in order to have more accurate approximations of the derivatives with higher frequency cost function variation.

The parallel summation method provided a solution to the convergence and accuracy problems of the Taylor series templates of larger connectivity. The solution did not change the underlying accuracy of the five-point approximation to the derivative, but rather achieved increased connectivity through the computation architecture of the template.

Figure 13 illustrates the minimum cost of successive algorithm iteration for the increasingly complex template connectivity patterns derived using the parallel summation architecture.

The convergence pattern is similar for each template. However, unlike the previous template patterns derived from the Taylor series, these template patterns exhibit the improvement expected from using increased connectivity as originally hypothesized. As the template connectivity is increased through more parallelism in the approximation summation, the network architecture accomplished a more rapid decrease in minimum cost paths. In each plot, the increased connectivity is indicated by a larger index, k. Each test case has a similar behavior. The higher the connectivity index, the faster the minimum cost drops during the algorithm iteration.

IV. Conclusion

The electrostatic model provided a sound mathematical basis for the research [7]. This was due to the nonhomogeneous partial differential equation (Laplace) of the electrostatic potential field. With the boundary conditions applied, the flux lines could be extracted by a gradient descent over the potential surface from the source node (maximum potential boundary condition) to the sink node (minimum potential boundary condition). Therefore,

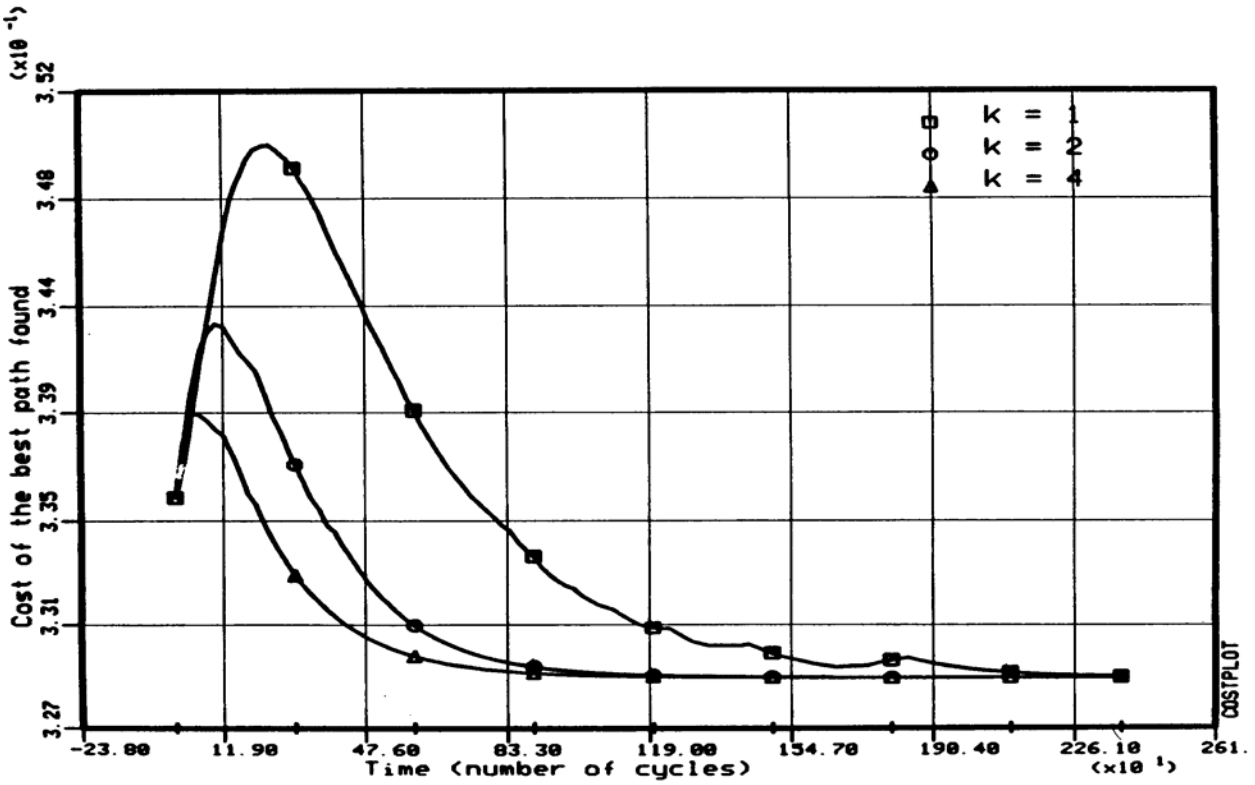

Figure 13: *Least Cost Path Connectivity Curves Using Parallel Summation Templates*

the challenge was to compute a solution to the partial differential equation using an artificial neural network.

The means to accomplish this was a finite difference approximation to the partial differential equation. A study of the finite difference approximation formulas revealed a striking similarity to the familiar equations for the artificial neural network. Thus, by an appropriate mapping of the coefficients of the finite difference approximation formulas into the weights of the artificial neural network, the neurons could compute the numerical solution to the partial differential equation.

A significant result of this approach is that the resulting design benefits from the classical numerical analysis of finite difference approximations for partial differential equations. Since the artificial neural network directly implements the finite difference approximation (in a truly parallel architecture), it retains the convergence and accuracy properties of the finite difference approximation. Additionally, the artificial neural network scales directly with the size of the problem just as is the case with the finite difference approximation. With this architecture design in place, the influence of the amount of interconnections could be investigated.

The finite difference approximation formulas map into simple interconnections between

adjacent neurons in the artificial neural network. Improvements to the convergence of the network were conjectured to result from increasing the interconnections between neurons beyond the locally adjacent neurons to larger regions of the network. This required a way to increase the number of input connections to each neuron from other neurons, and yet preserve the integrity of the finite difference approximation.

The solution was found in the parallel summation technique that was developed in Section III. This approach preserved the fundamental finite difference approximation formula as a template kernel so that the approximation of the derivative would be consistent in larger interconnection patterns. The technique consisted of recursively unfolding the kernel to rewire the input connections of each neuron directly to those neurons whose outputs were directly connected to the neuron whose outputs were the source of the input connections. Each recursive unfolding of the approximation template resulted in a higher amount of interconnections. This was shown to successfully reduce the convergence time and yet preserve the accuracy of the approximation. The benefit of the variable connectivity is to give the neural network designer tradeoffs between the convergence time, accuracy, and network costs (measured in terms of the implementation complexity through connectivity). For a given accuracy and time requirement, this tradeoff then specifies the amount of interconnectivity that is necessary in the artificial neural network. Conversely, for an embedded implementation of an artificial neural network with a given amount of connectivity, the design tradeoff can be used to project the accuracy obtainable from the network in a specified time interval.

References

[1] L. A. Reibling, "An electrostatic model of multiple path planning," *International Journal of Advanced Modeling and Optimization*, vol. 4, no. 3, pp. 41–63, 2002.

[2] L. Collatz, *The Numerical Treatment of Differential Equations.* New York, NY: Springer-Verlag, third ed., 1966.

[3] L. Lapidus and G. F. Pinder, *Numerical Solution of Partial Differential Equations in Science and Engineering.* New York, NY: John Wiley & Sons, Inc., 1982.

[4] M. G. Salvadori and M. L. Baron, *Numerical Methods in Engineering.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1961.

[5] A. Nussbaum, *ELECTROMAGNETIC THEORY for Engineers and Scientists.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1965.

[6] W. R. Smythe, *Static and Dynamic Electricity.* McGraw-Hill, Inc., 1950.

[7] L. A. Reibling, *A Massively Parallel Architecture Design for Path Planning Applications.* PhD thesis, Michigan State University, 1992.

[8] G. E. Forsythe and W. R. Wasow, *Finite-Difference Methods for Partial Differential Equations*. New York, NY: John Wiley & Sons, Inc., 1960.

[9] A. R. Mitchell and D. F. Griffiths, *The Finite Difference Method in Partial Differential Equations*. New York, NY: John Wiley & Sons, Inc., 1980.