A Generic Pattern for Modifying Traditional PDE Solvers to Exploit Heterogeneity in Asynchronous Behavior

Rajanikanth Jammalamadaka, Bernard P. Zeigler
Arizona Center for Integrative Modeling and Simulation,
Department of Electrical and Computer Engineering,
University of Arizona,
1230 E Speedway Blvd,
Tucson, Arizona.
{rajani, zeigler}@ece.arizona.edu

Abstract

This paper describes an activity based design pattern for solving partial differential equations. The pattern can be applied to any explicit or implicit single-step numerical method to construct an asynchronous modification with improved execution performance. For explicit methods, the modification results in no loss of accuracy while for implicit methods, we show how accuracy can be controlled by several parameters. To illustrate the approach we show how to apply the design pattern to standard single step methods for simulating the advection and diffusion problems in one dimension and demonstrate significant execution time reduction with minimal loss of accuracy compared to the original methods. We conclude by relating the concept of activity as employed in this paper to that discussed in earlier work and relate the design pattern to the existing DEVS-based quantization approaches.

1. Introduction

The concept of activity has been developed in [1, 2, and 3]. Many types of systems that are modeled as partial differential equations change significantly only in relatively small areas of the spatial domain at any given time. The work done previously introduced a way to characterize the amount of change in the solution through time and space, independently of the solution technique that is employed. In this paper we simplify the application of the concept to demonstrate some of the basic principles while still showing important consequences. In general, the activity at a cell refers to the magnitude of its partial time derivative. In contrast, in this paper we will distinguish only between zero and non-zero magnitude of the derivative. The set of cells for which the activity is not zero will be called the activity domain. We show how to restrict computation to this region as it evolves to significantly speed up the computation in cases where the activity region is relatively small and easy to track.

The flowchart in figure 1 illustrates the activity-based design pattern described in this paper. The pattern can be applied to any explicit or implicit single-step numerical method to construct an asynchronous modification with improved execution performance. The pattern augments the original method by attempting to restrict its application to the activity domain and accounting for the change in this domain as time proceeds. The resulting method works by restricting the initial space to obtain the initial activity domain. It then steps through time by following the illustrated sub-steps: 1) applying a method to minimally over-estimate the next activity domain based on properties of the partial differential equation, 2) applying the original method to the estimated activity domain, and 3) applying a method to restrict the estimated activity domain to the actual activity domain for the resulting state. The processing iterates until the simulation is done. Methods for estimating the next activity domain and restricting this domain to the true activity domain after one step iteration are part of the pattern and must be appropriately selected for each application. In the following sections, we show application of the design pattern to one-dimensional PDEs that have some general significance, advection and diffusion. We provide details of the augmentation methods required to work in these cases and show that they are simple enough to enable significant speed up in computation with near or nearly the same accuracy as provided by the original method.

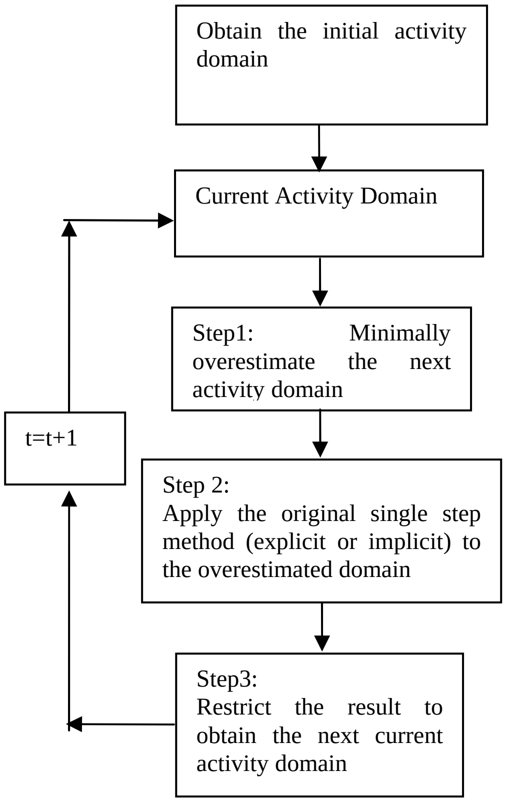

Obtain the initial activity domain
Current Activity Domain
Step1: Minimally overestimate the next activity domain
Step 2: Apply the original single step method (explicit or implicit) to the overestimated domain
Step3: Restrict the result to obtain the next current activity domain
t=t+1

Figure 1: Flowchart which illustrates the activity based design pattern

2. Estimating Speed-up

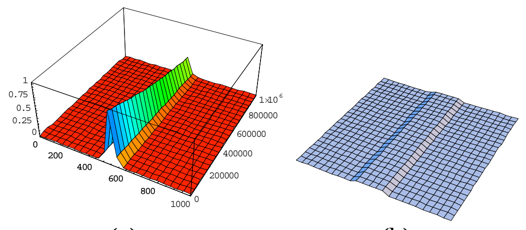
The speed up that results with the use of the activity based design pattern over a standard single-step method is estimated by

$$Ratio = \frac{Area \quad of \quad space-time}{Area \quad of \quad activity \quad in \quad space-time}$$

$$= \frac{nx * nt}{Area \quad of \quad activity \quad in \quad space-time}$$

where nx is the number of grid points in the spatial dimension and nt is the number of time steps in the simulation interval.

Figure 2(b) illustrates the effective computational region in solving 1-d diffusion used by a method to which the activity based design pattern has been applied



(a) (b)

Figure 2: Shows the effective computational region used by a method to which the activity based design pattern has been applied

3. Example 1: Forward Time Centered Space and Lax-methods for the Advection equation $\frac{\partial u(x,t)}{\partial t} = c \frac{\partial u(x,t)}{\partial x}$

The advection equation is given by $\frac{\partial u(x,t)}{\partial t} = c \frac{\partial u(x,t)}{\partial x}$ where c is the velocity of the wave. In this paper, the advection equation is solved with periodic boundary conditions when a pulse centered at $L/2$ (where L is the length of the space) is taken as the initial condition.

The activity based design pattern is applied to two commonly used explicit methods (FTCS and Lax [6]) for solving the advection equation. The flowchart in figure 3 illustrates the application of the design pattern to the FTCS and Lax methods. The initial condition is taken as follows $U(x,0) = \exp(-fac*(x^2)/2)$, where $x = ((1:N)-0.5)*h - L/2$, N is the number of grid points, L is the length and h is the spatial resolution. The width of the pulse is controlled by the fac. This width determines the size of the activity domain relative to the space and therefore, the expected increase in computation speed. As illustrated in figure 3, we first define a threshold (similar to quantum in a DEVS-based implementation [4]) and find the left-most cell, L and right-most cell, R where the pulse is greater than the threshold. If we choose a large threshold, then L and R will coincide with the boundaries of the spatial domain in which case, we just use the original method. Otherwise, we restrict the computation to the interval in which the value of the pulse is greater than the threshold**.** This is the region of activity since cells outside this region have negligible activity as determined by the threshold, i.e., cells whose value is zero and whose neighbors' values are zero have zero time derivative. This is true except for cells adjacent to the boundary of the activity region. These need to be included in the estimated activity domain for the next iteration. Thus, we have

Step 1:
Estimated activity domain = $[L-1, R+1]$ for FTCS

This is exact for the explicit FTCS method since it can expand activity at most to one neighbor on either side of the activity interval. Similarly, the Lax method employs two neighbors, therefore estimated activity domain: = $[L-2, R+2]$ for Lax.

Step 2:
Using vector notation, as in MATLAB, it is straightforward to restrict the single step transformation to the estimated activity domain:

For the FTCS method:

$$U(L-1:R+1) = U(L-1:R+1) - (c * \tau/(2*h)) * (U(L:R+2)$$

$$-U(L-2:R));$$

For the Lax method:

$$U(L-1:R+1) = 0.5*(U(L:R+2)+U(L-2:R))$$

$$-(c*\tau/2*h))*(U(L:R+2)-U(L-2:R));$$

where τ is the time step and h is the spatial cell size.

For further details the reader is referred to [6].

Step 3: Obtain next activity domain

$$if\ (U(L+1) > threshold)$$

$$L = L+1;$$

$$if\ (U(R+1) > threshold)$$

$$R = R+1;$$

We will show that applying the activity based design pattern to the FTCS and Lax methods can offer potential speed up without loss of accuracy. For the purposes of computation, both (modified and unmodified) codes have been run with 1000 grid points, c =velocity =1, L = [0, 1], τ =time step = 0.001 sec, number of time steps = 10 (therefore total time of simulation = 0.01 sec). The width factor was varied from 100 to 1000. The execution times shown here were measured by running the original method and the method to which the design pattern was applied for 1000 times and taking the average. Also, it should be noted that these numbers are relative i.e. if the programs were to be executed on a faster computer, the numbers may be smaller but the relative difference should be the same. The following table summarizes the comparison between the standard FTCS method and the FTCS method to which the design pattern was applied for solving the advection equation.

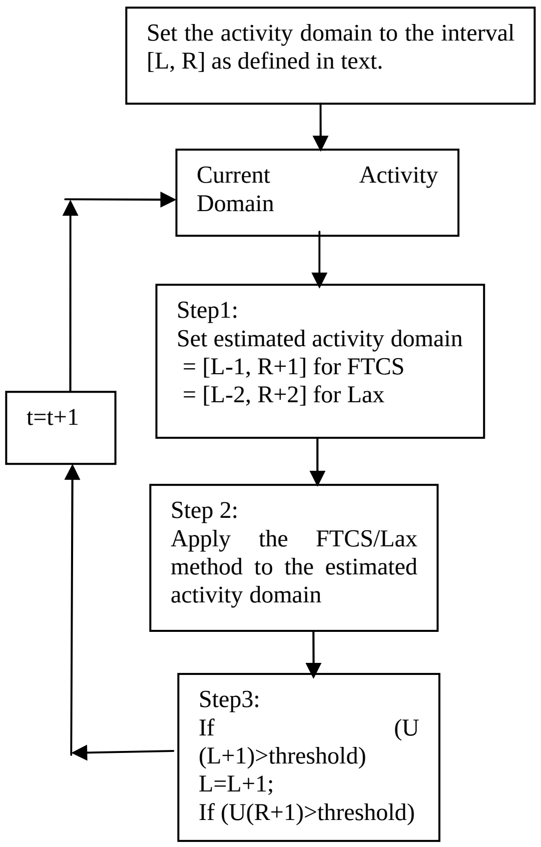

Set the activity domain to the interval [L, R] as defined in text.
Current Activity Domain
Step1:
Set estimated activity domain
= [L-1, R+1] for FTCS
= [L-2, R+2] for Lax
Step 2:
Apply the FTCS/Lax method to the estimated activity domain
Step3:
If (U (L+1)>threshold)
L=L+1;
If (U(R+1)>threshold)
t=t+1

Figure 3: Flowchart which illustrates the activity based design pattern for the Advection process

Table 1: Comparison between the standard and modified FTCS method for advection equation

fac (fac controls the pulse width)	FTCS method to which the design pattern was applied using a threshold of 10^-3 (Execution time in milli seconds)	Absolute Error for FTCS method to which the design pattern was applied (measured using 2-norm)	Standard FTCS method (Execution time in milli seconds)	Absolute Error for standard FTCS method (measured using 2-norm)
100	1.25	0.0061	2.24	0.0058
200	1.11	0.0100	2.447	0.0097
300	1.04	0.0135	2.40	0.0132
400	1.037	0.0166	2.575	0.0163
500	0.698	0.0196	2.468	0.0193
1000	0.661	0.0328	2.468	0.0326

As can be seen, as the initial pulse is given a narrower profile relative to the whole space, the modified method executes more rapidly both in absolute terms and in relation to the original method. Moreover, there is no loss in accuracy for explicit methods.

The following table summarizes the comparison between the standard Lax method and the Lax method to which the design pattern was applied for solving the advection equation.

Table 2: Comparison between the standard and modified Lax method for advection equation

fac (fac controls the pulse width)	Lax method to which the design pattern was applied, using threshold of 10^-15 (Execution time in milli seconds)	Absolute Error for modified Lax method (measured using 2-norm)	Standard Lax method (Execution time in milli seconds)	Absolute Error for standard Lax method (measured using 2-norm)
100	3.85	6.9*10^-6	3.85	6.9*10^-6
200	3.87	5.55*10^-11	3.55	5.55*10^-11
300	1.65	3.40*10^-15	3.54	2.6*10^-15
400	1.54	3.40*10^-15	3.56	2.70*10^-15
500	1.47	3.49*10^-15	3.55	2.77*10^-15
1000	1.30	4.6*10^-15	3.56	4.10*10^-15

Noting that the Lax method is second order accurate in both time and space, we see that the same results continue to hold as for the first order FTCS method. However, there is no advantage obtained with the modified method until the pulse is sufficiently narrow (at factor = 300). Indeed, the effect of the additional sub-steps is noted in the extra overhead for the case of factor = 200.

4. Example 2: Forward Time Centered Space (FTCS) and Backward Time Centered Space (BTCS)-methods for the Diffusion equation $\frac{\partial u(x,t)}{\partial t}=\alpha\frac{\partial^2 u(x,t)}{\partial x^2}$

The diffusion equation is given by $\frac{\partial u(x,t)}{\partial t}=\alpha\frac{\partial^2 u(x,t)}{\partial x^2}$ where α is the diffusion coefficient. In this paper, the diffusion equation is solved with open boundary conditions (i.e. $U(0)=0$ and $U(L)=0$) when the initial condition is taken as the same in example 1.

The activity based design pattern is applied to two commonly used explicit and implicit methods (FTCS and BTCS) for solving the advection equation. We will show that the modified methods offer potential speedups when compared with the original methods. They also offer approximately the same accuracy as the original methods.

The initial condition is taken as follows $U(x,0)=\exp\left(-fac*\left(\left(x-L/2\right)/L\right)^2\right)$ where L is the length of the space. In the following we describe the application of the design pattern to the BTCS method (the FTCS case is similar to Example 1).

In the following, we describe how the activity based design pattern was applied to the BTCS method. We follow a similar procedure to that in section 3 with some modifications that take into account the properties of the diffusion process.

Step 1: Since diffusion spreads outward, the estimated activity domain is expanded on both ends by two cells, i.e., $[L-2,R+2]$. However, for the implicit BTCS, the transition is global and can expand the activity interval in principle to the whole space. Nevertheless, we employ the same estimate and examine the resulting error production.

Step 2:

The BTCS is interesting since it is an implicit method that nominally is a global matrix inversion on the whole space. However, the LU decomposition method is commonly employed to reduce this operation from quadratic to linear in the number of cells. Using MATLAB's vector notation, the matrices representing the BTCS operator can be restricted to the estimated activity domain, as follows:

$a = (-\alpha / dx^2) * ones(R - L + 5, 1);$

$b=(1/dt)*ones(R-L+5,1)-2*a;$

$c=a;$

$b(1)=1;c(1)=0;$

$b(end) = 1; a(end) = 0;$

The LU decomposition is then computed for the restricted domain as:

$[e, f]=tridiagLU(a,b,c);$

The state vector at time t, is restricted to the activity domain as:

$$d = U(L-2:R+2,t)/dt;$$

$$d(1) = 0; d(end) = 0;$$

Finally, the state at time t+1 is computed as:

$$U(L-2:R+2,t+1)=tridiagLUSolve(d,a,e,f,U(L-2:R+2,t))$$

where tridiagLUSolve computes the result in R-L+5 iterations. For details on the standard BTCS method the reader is referred to [7].

Step 3: In the case of pulse diffusion, the activity domain first expands and then contracts. So to find the new activity domain after the state transition we employ tests for expansion and contraction. The cell adjacent to the right boundary checks whether it has become active and if so, becomes the new boundary cell:

$$if\,(U(R+1)>threshold)$$

$$R = R+1;$$

On the other hand, if the boundary cell becomes inactive, then it contracts the boundary inward:

$$elseif\,(U(R) < threshold)$$

$$R = R-1;$$

The left boundary is handled similarly.
For the BTCS, both (modified and unmodified) codes have been run with 50000 grid points, alpha=diffusion coefficient =0.1, L = [0, 1000], time step = 0.5 sec, total time of simulation = 500 sec, factor = 4000. A threshold of 10^-5 was used when the design pattern was applied to original FTCS and BTCS methods in order to modify them.

The following table summarizes the comparison between the original BTCS method and the BTCS method to which the design pattern was applied for solving the advection equation.

Table 3: Comparison between the standard and modified BTCS method for diffusion equation

fac	Modified BTCS method (Execution time in seconds) using threshold of 10^-5	Absolute Error for modified BTCS method (measured using 2-norm)	Standard BTCS method (Execution time in seconds)	Absolute Error for standard BTCS method (measured using 2-norm)
1000	25.0	3.41	75	3.41
4000	12.8	7.36	127	7.36

Note that a speed up of approximately 10 is obtained for the narrower pulse without loss in accuracy.

Since the FTCS method is explicit, both the modified and unmodified FTCS methods were run using 5000 grid points, with a time step of 0.001 sec and total time of simulation = 50 sec (rest of the parameters weren't changed). A threshold of 10^-5 was used when the design pattern was applied to original FTCS and BTCS methods in order to modify them. The following table summarizes the comparison between the original FTCS method and the FTCS method to which the design pattern was applied for solving the advection equation.
As it was mentioned before, since the FTCS method is explicit, there is no loss in accuracy when the activity based design pattern is applied to it. Note that a speed up of approximately 20 is obtained for the narrower pulse without loss in accuracy.

Table 4: Comparison between the standard and modified FTCS method for diffusion equation

fac	Modified FTCS method (Execution time in seconds) using threshold of 10^-5	Absolute Error for modified FTCS method (measured using 2-norm)	Standard FTCS method (Execution time in seconds)	Absolute Error for standard FTCS method (measured using 2-norm)
1000	10.6	0.12	31	0.12
4000	8.0	0.32	156	0.32

To verify the correctness of the modified methods, simulation runs were extended until the diffusion process terminated with all cell values under threshold. Visual inspection of the resulting plots confirmed that the correct behavior was preserved.

The unmodified Matlab codes for advection were taken from [6]. The unmodified Matlab codes for diffusion were taken from [7].

5. Discussion

Table 5 summarizes our results by comparing actual execution time ratios to estimates using the approach in section 2.

Table 5: Summary of Results

Experiment	Speed-up (Ratio of original method execution time to modified method execution time)	Speed-up estimate using activity ratio
Advection(FTCS,fac=1000)	3.7	4.0
Advection(Lax,fac=1000)	2.0	4.0
Diffusion(BTCS,fac=1000)	3.0	4.0
Diffusion(BTCS,fac=4000)	10.0	10.0
Difusion(FTCS,fac=1000)	2.9	4.0
Diffusion(FTCS,fac=4000)	19	10.0

The activity ratios were computed analytically using the given thresholds as illustrated in figure 2(b). We see that generally the estimated speed-ups are close to the actual results.

6. Conclusion

This paper described an activity based design pattern for solving partial differential equations. The pattern can be applied to any explicit or implicit single-step numerical method to construct an asynchronous modification with improved execution performance. The approach was illustrated by showing how to apply the design pattern to standard single step methods for simulating the advection and diffusion problems in one dimension. For explicit methods, the modification results in no loss of accuracy while for implicit methods, we showed how accuracy can be controlled by selecting appropriate activity estimation and restriction methods. Although the illustrated applications were admittedly relatively simple in nature, the design patterns they instantiate opens up the possibility of research into more challenging applications.

We conclude by relating the concept of activity as employed in this paper to that discussed in earlier work [1, 2, and 3]. In general, the activity at a cell refers to the magnitude of its partial time derivative. Existing DEVS-based quantization approaches exploit the heterogeneity of activity distributions by creating an asynchronous simulation model such that the time advances of its cells are inversely proportional to the magnitude of their partial time derivatives. In this way, computational resources are automatically allocated in relation to current activity, viz.; cells with small time advances (i.e., high activity) are reprocessed more rapidly than those with large advances. In contrast, in this paper we will distinguish only between zero and non-zero magnitude of the derivative. This corresponds to distributions in which cells are either passive (with infinite time advances) or active (and given the same constant time step). This results in asynchronous simulation since at any time there is a distribution of active and passive cells, although active cells are all synchronously processed. The DEVS-based quantization approaches are generalizations in the sense that they result in truly asynchronous models that can be executed in distributed simulations to exploit heterogeneity in time and space. The speed-up analysis of section 2 is generalized to quantization approaches in reference [1].

7. References

[1] R. Jammalamadaka, "Activity Characterization of Spatial Models: Application to the Discrete Event Solution of Partial Differential Equations", M.S Thesis: Fall 2003, Electrical and Computer Engineering Dept, University of Arizona.

[2] J. Nutaro, B.P. Zeigler, R. Jammalamadaka, S. Akerkar, "Discrete Event Solution of Gas Dynamics within the DEVS Framework: Exploiting Spatiotemporal Heterogeneity", ICCS, Melbourne, Australia, July 2003.

[3] Bernard P. Zeigler, R. Jammalamadaka, S. Akerkar, "Continuity and Change (Activity) Are Fundamentally Related In DEVS Simulation of Continuous Systems", AIS'04, October 4-6, Korea.

[4] Bernard P. Zeigler, Herbert Praehofer, Tag Gon Kim, Theory of Modeling and Simulation, 2nd Edition, Academic Press.

[5] Alexandre Muzy, A. Aiello, Paul-Antoine Santoni, Bernard P. Zeigler, James J. Nutaro, and Rajanikanth Jammalamadaka. Discrete event simulation of large-scale spatial continuous systems. In *International Conference on Systems, Man and Cybernetics (SMC)*, Hawaii, USA, October 2005. IEEE.

[6] Alejandro L. Garcia, Numerical Methods for Physics, Prentice Hall, Englewood Cliffs, NJ, 2000.

[7] Gerald W. Recktenwald, Finite-Difference Approximations to the Heat Equation, Class Notes, 2004.