An Operator Splitting Method for Nonlinear Reactive Transport Equations and Its Implementation Based on DLL and COM

Jiangguo Liu[1] and Richard E. Ewing[1]

Institute for Scientific Computation, Texas A&M University, College Station, TX 77843-3404 jliu@isc.tamu.edu,richard-ewing@tamu.edu

In this paper, we propose an operator splitting method for convection-dominated transport equations with nonlinear reactions, which model groundwater contaminant biodegradation and many other interesting applications. The proposed method can be efficiently implemented by applying software integration techniques such as dynamical link library (DLL) or component object model (COM). Numerical results are also included to demonstrate the performance of the method.

1 Introduction

The growing concern of groundwater contamination demands accurate description and thorough understanding of contaminant transport in porous media. The cleanup of contaminated groundwater can be effective only after complete knowledge is gained about the mechanisms in transport and chemical and biological reactions. In particular, understanding of the microbial degradation of organic contaminants in groundwater provides valuable guidelines for *in-situ* remediation strategies. Injection of oxygen or various nutrients into the subsurface might be encouraged or enhanced to stimulate the microbial activities.

The mathematical models describing contaminant transport with biodegradation are time-dependent convection-diffusion-reaction equations. The nonlinear reaction terms may describe the reactions among all the species and may be coupled to the growth equations for the bacterial populations. These models are often complicated due to the heterogeneous geological structure in porous media, through which the contaminants are transported. The nonlinearity in the reaction terms also needs to be treated carefully. Therefore, solutions to these equations present serious challenges to numerical methods.

In [WEC1995], Eulerian-Lagrangian localized adjoint methods (ELLAM) based on different operator splittings of the adjoint equation are developed

to solve linear transport equations. Then specific linearization techniques are discussed and are combined with the ELLAM schemes to solve the nonlinear, multi-species transport equations. Other operator splitting methods for convection-diffusion-reaction equations with other applications, e.g, air pollution modeling, can be found in [KR2000, KR1997, LV1999] and the references therein.

In this paper, we propose an operator splitting method for the governing equation to handle the nonlinear reactions. Roughly speaking, we split the transport equation into a convection-reaction part and a diffusion part. The diffusion problem is more or less a standard parabolic problem, whereas the convection-reaction problem becomes a nonlinear ordinary differential equation (ODE) along characteristics. We shall focus on the implementation issues of the splitting method. We shall treat a finite difference/element/volume method-based solver for conventional (linear) problems as a stand-alone object, which can be integrated with our own codes in a flexible manner. To be specific, this integration is done through dynamical link library (DLL) and component object model (COM). The CPU intensive numerics will be implemented with low-level programming languages such as C/C++/FORTRAN and code design will adhere to the object-oriented programming paradigm.

Recently, object-oriented programming (OOP) is becoming a popular practice in designing numerical (computing) software. Here we give [LM2001, BM2002] as just two interesting examples among many other excellent papers in this aspect.

2 Transport Equations with Nonlinear Reactions

A general single-species reactive transport equation can be written as

$$\begin{cases} u_t + \nabla \cdot (\mathbf{v}u - \mathbf{D}\nabla u) = R(u) + f(\mathbf{x}, t), \quad \mathbf{x} \in \Omega,\ t \in (0, T) \\ u|_{\partial\Omega} = 0 \\ u(\mathbf{x}, 0) = u_0(\mathbf{x}) \end{cases} \tag{1}$$

where $[0, T]$ is a time period, Ω a domain in $\mathbf{R}^d (d = 1, 2, \text{or}, 3)$, $u(\mathbf{x}, t)$ is the unknown (population) density or concentration of the species, $\mathbf{v}$ is a divergence-free velocity field, $\mathbf{D}$ is a diffusion tensor (it is assumed that $|\mathbf{D}| << |\mathbf{v}|$), $R(u)$ is a nonlinear reaction term, and $u_0(\mathbf{x})$ is an initial condition.

Some very interesting cases for the reactions are

- Radioactive decay with $R(u) = -au$;
- Logistic model with $R(u) = au - bu^2$;
- Biodegradation model with $R(u) = au/(u + b)$.

It should be pointed out that the third case is just a simplified model. More realistic models and detailed discussions on biodegradation can be found in [CKH1989].

3 An Operator Splitting Method for Nonlinear Reactive Transport Equations

Splitting methods come in two basic flavors: dimensionality reduction and operator decomposition. Both have been adopted in solving partial differential equations. A brief discussion on applying operator splitting techniques to solve transport equations in porous media can be found in [E2002]. In [LV1999], there is a very interesting theoretical analysis of operator splitting for convection-diffusion-reaction equations from the viewpoint of Lie algebra.

The transport equation (1) can be rewritten as

$$u_t = A(u) + B(u)$$

with

$$A(u) = -\mathbf{v} \cdot \nabla u + R(u),$$
$$B(u) = \nabla \cdot (\mathbf{D}\nabla u) + f(\mathbf{x}, t).$$

Let N be a positive integer, $\Delta t := T/N$, and $t_n = n\Delta t$ $(n = 0, 1, \cdots, N)$ be a uniform partition of the time period $[0, T]$. We split equation (1) into two in each small time period $[t_n, t_{n+1}]$:

$$u_t + \mathbf{v} \cdot \nabla u = R(u), \tag{2}$$

$$u_t = \nabla \cdot (\mathbf{D}\nabla u) + f(\mathbf{x}, t). \tag{3}$$

For any $\mathbf{x} \in \Omega$, the characteristic $\mathbf{y}(s; \mathbf{x}, t_{n+1})$ passing through $(\mathbf{x}, t_{n+1})$ is determined by

$$\begin{cases} \dfrac{d\mathbf{y}}{ds} = \mathbf{v}(\mathbf{y}, s), \quad s \in (t_n, t_{n+1}) \\ \mathbf{y}(t_{n+1}; \mathbf{x}, t_{n+1}) = \mathbf{x}. \end{cases} \tag{4}$$

Let $(\mathbf{x}^*, t_n)$ be the image of exact backtracking of $(\mathbf{x}, t_{n+1})$. If the characteristic hits the boundary, then we use $(\mathbf{x}^*, t_n^*)$, where $\mathbf{x}^* \in \partial\Omega, t_n \le t_n^* \le t_{n+1}$. Notice that exact tracking of characteristics is usually unavailable in practice and we have to resort to numerical means. All commonly used numerical methods for solving ODEs, e.g., Euler and Runge-Kutta methods, can be applied to problem (4).

Along characteristics, the convection-reaction equation becomes a nonlinear ODE

$$\begin{cases} \dfrac{du^{(1)}}{dt} = R(u^{(1)}), \quad t \in (t_n^*, t_{n+1}), \\ u^{(1)}(\mathbf{x}^*, t_n^*) = u^{(2)}(\mathbf{x}^*, t_n^*), \end{cases} \tag{5}$$

where $u^{(2)}$ is the solution for the parabolic problem and will be explained later. When $n = 0$ or $\mathbf{x}^* \in \partial\Omega$, we should replace $u^{(2)}(\mathbf{x}^*, t_n^*)$ by $u_0(\mathbf{x}^*)$ or the boundary condition.

Problem (5) can be solved numerically by, e.g., Euler and Runge-Kutta methods. This way the nonlinearity in the reaction term can be well resolved if $R(u)$ is Lipschitz with respect to u, which is true for the logistic and biodegradation models.

The other part is an initial boundary value problem for a typical parabolic equation

$$\begin{cases} u_t^{(2)} = \nabla \cdot (\mathbf{D}\nabla u^{(2)}) + f(\mathbf{x}, t), \quad \mathbf{x} \in \Omega,\ t \in (t_n, t_{n+1}), \\ u^{(2)}|_{\partial\Omega} = 0, \\ u^{(2)}(\mathbf{x}, t_n) = u^{(1)}(\mathbf{x}, t_{n+1}). \end{cases} \tag{6}$$

It is conventional and can be solved by finite difference, element, or volume methods.

Let h be the spatial mesh size in the numerical scheme (finite difference or element or volume) for solving the parabolic part (6), δt the temporal step size. Since $|\mathbf{D}| << 1$, the stability condition $|\mathbf{D}|\delta t/h^2 \leq 1/2$ are readily satisfied.

4 Techniques of Software Integration

4.1 DLL: Dynamic Link Library

A dynamic link library(DLL) is a collection of functions and data that are available for use by one or more applications running on a computer system. At run-time, the DLL is loaded into memory and made accessible to all applications. In other words, executable code modules in a DLL are loaded on demand and linked at run time, and then unloaded when they are no longer needed.

The modules calling a DLL could be an executable (EXE) or even another DLL. When a DLL is loaded, it is mapped into the address space of the calling process. So DLLs also help reduce memory overhead when several applications use the same functionality at the same time; because although each application gets its own copy of the data, they can share the code.

DLLs can define two kinds of functions: exported and internal. Exported functions can be called by other modules. Internal functions can only be called

from within the DLL where they are defined. Although a DLL can export data, its data is usually only used by its own functions.

Moreover, DLLs provide a way to modularize applications so that functionality can be updated and reused more easily. This enables the library code to be updated automatically and be transparent to applications.

The DLL standard is now supported by many high level programming languages such as C++, Fortran, and Matlab.

4.2 COM: Component Object Model

Component Object Model (COM) is the Microsoft's binary standard for object interoperability and has been widely accepted for integration of external functionality into Microsoft applications. COM is a set of object-oriented technologies and tools, which allows software developers to integrate application-specific components from different vendors into their own applications. These assembled reusable components communicate via COM. By applying COM to build applications from preexisting components, developers enjoy benefits of good maintainability and adaptability.

COM allows clients to invoke services provided by COM-compliant components (COM objects). COM objects are implemented as either Dynamic Link Libraries (DLLs) or executables (EXEs). COM objects implemented as DLLs are called in-process servers, while those implemented as EXEs are called local servers. An in-process COM server exposes its functions to outside applications, whereas a local server does not.

There are three ways in which a client can access COM objects:

- In-process server: The client can link directly to a library containing the server. The client and server execute in the same process. Communication is accomplished through function calls;
- Local Object Proxy: The client can access a server running in a different process (but on the same machine) through an inter-process communication mechanism;
- Remote Object Proxy: The client can access a remote server running on another machine. The mechanism supporting access to remote servers is called DCOM (D means Distributed).

If the client and server are in the same process, sharing data between the two is simple. However, when the server process is separated from the client process, as in a local server or remote server, COM must format and bundle the data in order to share it. This process of preparing data is called marshalling. For more details about marshalling, readers are refereed to [MsCOM].

Again, the COM standard is now supported by many high level programming languages such as C++, Fortran, and even Matlab.

5 Implementation of the Operator Splitting Method

In this section, we discuss some issues related to implementation of the operator splitting method.

Mesh Generation

Both the convection-reaction part and the parabolic part in the operator splitting method need a spatial mesh, which can be generated via applying most of those good quality mesh generators. The mesh information will be saved as data files and then read into memory later. A mesh generator can be integrated with our own program via COM and mesh data input can be done through flat file transfer.

Splitting Alternately

In our implementation, we actually adopt an alternating approach. That is, we solve the convection-reaction problem first, then the parabolic problem. For the following time step, we switch the order: solving the parabolic part first and then the convection-reaction part. Our numerical experiments indicate that this alternating approach has a better control over the splitting error.

Tracking Characteristics

Characteristic tracking is an important part of this operator splitting method. Usually, approximate characteristics instead of exact characteristics are used in numerical schemes, although the latter will be utilized whenever they are available. An approximate characteristic is understood as a chain of line segments. These line segments are adaptively formulated according to the magnitude of the velocity field. Small time steps are used where velocity magnitude is large, whereas relatively large time steps can be used when velocity is small. The number of points on each (approximate) characteristic varies. In addition, both forward and backward tracking of characteristics will be performed in practice. Therefore, doubly linked list seems to be an efficient data structure for approximate characteristics. It is this part (adaptive tracking of characteristics) where we have to develop our own code.

Solving Nonlinear ODEs along Characteristics

Now the nonlinear ODE (5) along characteristics are actually solved along approximate characteristics. Euler and Runge-Kutta methods can be applied. Note that there might be different time steps on different characteristics.

Solving Parabolic Problem via FEM

Numerically solving an initial boundary value problem to a linear parabolic equation via the finite difference/element/volume method is now more or less a conventional task. There are plenty of commercial or free softwares for doing just that. As we know, libMesh developed at The University of Texas at Austin [libMesh] and OFELI (Object Finite Element Library) developed in France [OFELI] provide free C++ source code and are easy to use. The PDE toolbox in Matlab is a basic finite element package and does not require users to have C++ programming experience.

DLL Inside and COM Outside

The operator-splitting method is actually a time-stepping procedure. Before the time-stepping loop, a spatial mesh is generated via a mesh generator. Within each time step, a convection-reaction problem and a parabolic problem are solved. After the loop, numerical solutions can be visualized by calling graphics utilities like the Visualization ToolKit (VTK). Then another spatial mesh might be needed to re-run the whole process. Taking the overhead in calling COM objects into consideration, we implement the two functionality modules, namely, solving a convection-reaction problem and solving a parabolic problem, through DLL. The mesh generator and visualization modules outside the time-stepping loop are implemented via COM.

6 Numerical Experiments

6.1 Example 1: Linear Reaction

To examine our method, we first consider a 2-dimensional problem with a linear reaction, to which we can find the exact solution so that we can compare the numerical and exact solutions. In particular, we have a rotating velocity $\mathbf{v} = (-4y, 4x)$, a constant scalar diffusion $D > 0$, a linear reaction $R(u) = Ku$ with K being a constant, and a null source/sink, i.e., $f \equiv 0$. Assume the substance is initially normally distributed, i.e., the initial condition is specified as a Gaussian hill

$$u_0(x, y) = \exp\left(-\frac{(x - x_c)^2 + (y - y_c)^2}{2\sigma^2}\right),$$

where (x_c, y_c) and $\sigma > 0$ are the center and standard deviation, respectively. Then the exact solution is given by

$$u(x, y, t) = \frac{2\sigma^2}{2\sigma^2 + 4Dt} \exp\left(Kt - \frac{(x^* - x_c)^2 + (y^* - y_c)^2)}{2\sigma^2 + 4Dt}\right),$$

where $(x^*, y^*, 0)$ is the backtracking foot of the characteristic from (x, y, t), that is,

$$\begin{cases} x^* = (\cos 4t)x + (\sin 4t)y, \\ y^* = -(\sin 4t)x + (\cos 4t)y. \end{cases}$$

For simplicity, we use a uniform triangular mesh. The 2nd order Runge-Kutta (or Heun) method is used for characteristic tracking even though exact tracking is available. The finite element solver for the parabolic part (as a DLL) is derived from the source code in OFELI.

For numerical runs, we choose $T = \pi/2$, $\Omega = [-1, 1] \times [-1, 1]$, $D = 10^{-4}$, $K = 0.1$, $(x_c, y_c) = (-0.5, -0.5)$, and $\sigma^2 = 0.01$. For the parabolic solver, we use 20 micro steps within each global time step $[t_n, t_{n+1}]$. Accordingly, we set the maximal number of time steps in characteristic tracking to 20 also. Table 1 lists some results for the numerical solution at the final time. We still obtain very good numerical solutions, even though large global time steps are used.

Table 1. Numerical results of example 1 with $\Delta t = \pi/8$

Mesh size h	L^∞-error	L^1-error	L^2-error
1/20	1.266×10^{-2}	1.247×10^{-4}	3.138×10^{-4}
1/40	1.031×10^{-2}	5.061×10^{-5}	2.085×10^{-4}
1/50	9.984×10^{-3}	4.153×10^{-5}	1.923×10^{-4}
1/60	9.796×10^{-3}	3.613×10^{-5}	1.825×10^{-4}

6.2 Example 2: Nonlinear Reaction

The second example is a simplified model for single-species biodegradation: $R(u) = au/(u + b)$. We consider a 2-dimensional problem with a constant velocity filed (V_1, V_2), a scalar diffusion $D > 0$, and no source/sink. The initial condition is a normal distribution (Gaussian hill). Again we use the parabolic solver (DLL) compiled from OFELI.

Table 2. Numerical results of example 2

Δt	h	U_{min}	U_{max}	Δt	h	U_{min}	U_{max}
0.25	1/20	0.0	1.5159	0.125	1/40	0.0	1.5251
0.25	1/40	0.0	1.5176	0.10	1/20	0.0	1.5248
0.25	1/60	0.0	1.5179	0.10	1/50	0.0	1.5268

For numerical runs, we choose $T = 1$, $\Omega = [-1, 1] \times [-1, 1]$, $(V_1, V_2) = (1, 1)$, $D = 10^{-4}$, $a = b = 1$, $(x_c, y_c) = (-0.5, -0.5)$, and $\sigma^2 = 0.01$. Table 2 lists some results of the numerical solution at the final time.

No exact solution is known for this problem. But from Table 2, we can observe that the operator splitting method is stable, and keeps positivity of the solution.

7 Concluding Remarks

In this paper, we propose an operator splitting method for transport equations with nonlinear reactions. In the implementation of the proposed method, we incorporate some existing commercial and free software components into our own program. These components include mesh generators, finite element packages, and graphics utilities. By integrating functionalities of existing software components, application developers do not have to create every thing from scratches. So less code has to be written and the time period of software development is significantly shortened. Of course, this is based on the re-usability of software components, which requires all developers to abide software development standards. Moreover, the case study presented here shows that we are moving towards to reuse of binary objects (DLL or COM), instead of merely source code. We also want to point out that customization of existing software components to some extent might still be necessary.

This paper focuses on strategies for efficient implementation of the operator splitting method. Numerical results indicate that the method works very well. Error analysis on the method will be presented in our future work. In addition, extension of the proposed operator splitting method to coupled systems is also under our investigation.

References

[BM2002] Bertolazzi, E., Manzini, G.: P2MESH: Generic objected-oriented interface between 2-d unstructured meshes and FEM/FVM-based PDE solvers. ACM Transactions on Mathematical Softwares, **28**, 101–131 (2002)

[CKH1989] Celia, M.A., Kindred, J.S., Herrera, I.: Contaminant transport and biodegradation 1. a numerical model for reactive transport in porous media. Water Resources Research, **25**, 1141–1148 (1989)

[DER1995] Dahle, H.K., Ewing, R.E., Russell, T.F.: Eulerian-Lagrangian localized adjoint methods for a nonlinear advection-diffusion equation. Comput. Meth. Appl. Mech. Eng., **122**, 223–250 (1995)

[E2002] Ewing, R.E.: Upscaling of biological processes and multiphase flow in porous media. In: Fluid Flow and Transport in Porous Media: Mathematical and Numerical Treatment, Contemp. Math. **295**, 195–215 (2002)

[E1990] Ewing, R.E.: Operator splitting and Eulerian-Lagrangian localized adjoint methods for multiphase flow. In: The mathematics of Finite Elements and Applications, VII (Uxbridge). Academic Press, London (1991)

[KR2000] Karlsen, K.H., Risebro, N.H.: Corrected operator splitting for nonlinear parabolic equations. SIAM J. Numer. Anal., **37**, 980–1003 (2000)
[KR1997] Karlsen, K.H., Risebro, N.H.: An operator splitting method for nonlinear convection-diffusion equations. Numer. Math., **77**, 365–382 (1997)
[LM2001] Langtangen, H.P., Munthe, O.: Solving systems of partial differential equations using object-oriented programming techniques with coupled heat and fluid flow as example. ACM Transactions on Mathematical Softwares, **27**, 1–26 (2001)
[LV1999] Lanser, D., Verwer, J.G.: Analysis of operator splitting for advection-diffusion-reaction problems from air pollution modelling. J. Comput. Appl. Math., **111**, 201–216 (1999)
[WEC1995] Wang, H., Ewing, R.E., Celia, M.A.: Eulerian-Lagrangian localized adjoint methods for reactive transport with biodegradation. Numer. Meth. Partial Diff. Eqn., **11** , 229–254 (1995)
[MsCOM] Website: Component Object Model: http://www.microsoft.com/com/
[libMesh] Website: libMesh: http://libmesh.sourceforge.net/
[OFELI] Website: OFELI: http://ofeli.sourceforge.net/
[VTK] Website: VTK: http://www.vtk.org/