**Insights from the Development of an Automated Framework and Test Suite for Verifying a One-dimensional Transport Model**

E. Ateljevich1, K. Zamani2, F. A. Bombardelli2 and J. Anderson1

1Bay-Delta Office, Delta Modeling Section, California Department of Water Resources (DWR), 1416 Ninth Street Sacramento, CA, P.O. BOX 942836; PH (916) 645-5455; FAX (916) 653-6077; e-mail: eli@water.ca.gov, jamiea@water.ca.gov.

2Department of Civil and Environmental Engineering, University of California, Davis, 2001 Ghausi Hall, Davis, CA 95616; PH (530) 752-0949; FAX (530) 752-7872; email: [kzamani@ucdavis.edu](mailto:kzamani@ucdavis.edu), [fabombardelli@ucdavis.edu](mailto:fabombardelli@ucdavis.edu).

**ABSTRACT**

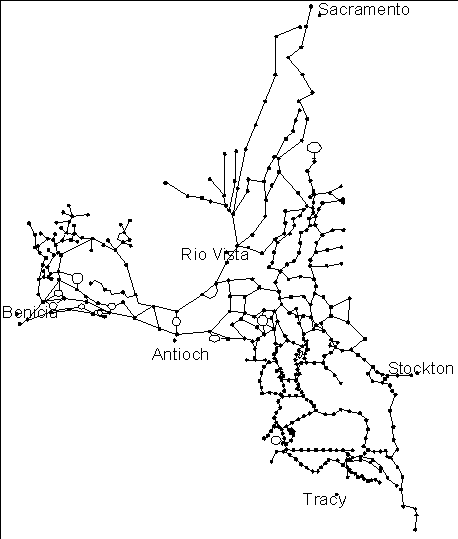
In this paper we describe a framework for software verification of a transport model. The criteria of the framework are driven by the requirements for the model, i.e., by the processes embedded in the model, but are crafted according to principles from both the software and numerical testing fields. Herein, we describe the components and implementation of the suite, emphasizing the incremental nature of the tests, quantitative criteria for testing, and the tension between the silent, automatic perspective of software testing and the verbose, graphical outputs required for public reporting of numerical verification results. Our experience might result in a useful starting point for researchers and practitioners wanting to verify codes in similar situations. WE CAN GO AS FAR AS 150 WORDS

**INTRODUCTION**

The California Department of Water Resources maintains the Delta Simulation Model 2 (DSM2), a one-dimensional (1D) hydrodynamic and transport model for rapidly simulating flow and water quality in the Sacramento-San Joaquin Delta (Fig. 1). Recently, the authors commenced work on a more flexible and more rigorously verified transport component for this suite. Our target problems include river and estuary advection, 1D approximations of common mixing mechanisms and source terms associated with sediment, radiation and non-conservative water quality kinetics. As with any high-quality software, it needs to be verified and validated. Verification and validation (V&V) are the most important steps to assess the accuracy and reliability of any numerical simulation. In simple words, in the step of verification the correctness of the solution technique used and implementation are evaluated. In turn, validation consists in evaluating whether the model is properly representing the processes involved in the problem (Fig. 2).

When faced with the need of verification and validation of our software developments, we found that rigorous testing frameworks are seldom implemented - a surprising feature considering the many pieces of code of similar characteristics developed in the last 40 years. Therefore, we embarked on the development of a new testing framework for such purpose.

In this paper, we describe our approach and experiences developing a software verification framework for a transport model. We begin by describing the motivation and requirements for testing. The criteria are driven by the requirements for the model, but are crafted according to principles from both the software and numerical testing fields. We then describe the components and implementation of the suite, emphasizing the incremental nature of the tests, quantitative criteria for testing and the tension between the silent, automatic perspective of software testing and the verbose, graphical outputs required for public reporting of numerical verification results.



**Figure 1. DSM2-Hydro, the Hydrodynamic Engine of the ADR Solver; Schematic of the Sacramento - San Joaquin Delta Grid. DWR Bay Delta Office, 22nd annual report (2001).**

**Problem definition, scaling and motivation of the testing.** Both the scaling of the problem and our choice of algorithm influence the components of our test suite. The dynamics are formulated in conservative form as follows:



where *A* is the wetted area, *C* is the scalar concentration, *u* is the flow velocity, *K* is the longitudinal dispersion coefficient, and *R* is the source term (deposition, erosion, lateral inflow and other forms of sources and sinks). Equation (1) describes the mass conservation of a given pollutant in dissolved phase, or suspended sediment away from the streambed.

The problem domain includes estuaries and river channels and even some open water areas grossly approximated as channels (Fig. 1). The main transport process is advection, and the mixing mechanisms we anticipate are turbulent diffusion, gravitational circulation, and shear dispersion. We anticipate the shear dispersion to obviously dominate over the turbulent diffusion, but we also expect the gravitational circulation to exert an important role in mixing. We additionally contemplate significant, non-linear source terms. While none of the above processes are so quickly varying as to constitute truly stiff reactions, we have CCCC

Our algorithms include an explicit scheme for advection based on the finite-volumes method (FVM) and the Lax, two-step method with van-Leer flux limiter; it also includes an implicit, time-centered Crank-Nicolson scheme for dispersion. The advection and reaction solver are coupled as a predictor corrector pair, and diffusion is implemented using operator splitting. Two features of the algorithm are particularly important. First, the scheme requires a flow field (flow discharges and flow areas) that preserves mass continuity. In some cases tests from the literature were written in non-conservative or primitive form in terms of a velocity and had to be reworked in conservative form. Second, we employ operator splitting and wanted to exercise the equations with and without known vulnerabilities (such as time-varying boundaries and nonlinear source terms) of this class of algorithm.

The target accuracy is strict second order for individual operators and near second-order for the algorithm as a whole. Second order allows coarser discretization for a modest increase in work. As computer architectures favor multiple operations with minimal movement of data, this advantage of computation over grid density is on the increase. A second-order algorithm also gives us a buffer of accuracy as details like networks of channels and coarse boundary data are added. At the time of writing this paper, our splitting is first order Godunov splitting; one goal of our testing is determine how applicable the practical observation that near second-order accuracy can be achieved with first order splitting (e.g. Leveque 1986) is to our algorithm.

Conceptual Model (Continuum)

Computational Model (Discretized)

Natural Phenomenon

(Reality)

Validation

Qualification

Verification

Programming

Simplification

Simulation

**Figure 2. Schematic of stages of CFD model qualification, verification and validation.**

**Testing principles**. Flow and transport codes inherently comprise both numerical algorithms and pieces of software. Although some well-developed testing literature exists for both, Oberkampf and Trucano (2002) describe some elements of software quality engineering in the context of numerical verification, and notes some cultural reasons why it is seldom implemented. There are myriad approaches to software testing. The principles that we want to emphasize are:

1. Testing should be automatic and continuous.
2. The approach should foster good architecture and exact specification of every unit of code
3. Tests should be designed around the specific ways developers make mistakes.
4. The approach should test ever important special case for sub-processes.
5. Tests need not have diagnostic value.

One goal of tests is that they be a continuous assessment of the code. The tests themselves stay static, and establish a gauntlet of through which future changes must be passed. Such a test is usually called a regression test, and regressions are harvested from past unit tests, system tests and mistakes. One consequence of automation and regression is that test suites must be based on binary *assertions,* true and false statements that can be tested without human intervention and that reveal whether the aspect of the code under consideration is correct.

Unit tests exercise single units of code, such as one small subroutine calculating a gradient over a mesh. Properly designed unit tests exercise all the important “corner” cases and special inputs. Examples for a gradient would be:

1. One well-behaved central case.
2. A test of edges of the mesh.
3. Tests that exercise features such as flux limiters with steep or zero gradients in both directions.

Unit tests are fast, and they allow a great number of special cases to be exercised, which might take months when the case is embedded in a larger solver. For instance, in the example of a gradient with limiter, imagine that the limiter contains a bug in one direction. Normal verification based on convergence might miss this in two ways. First, convergence is usually assessed without limiters, as they are order reducing. Second, algorithmic tests are usually constructed to show the algorithm solves a Partial Differential Equation, not that the code does this right in both directions.

\*\*\*\* unit test vs Sandia assertion the convergence tests catch everything.

Incremental 9454948223

above example, it is easy to imagine a full system test

1. Once passed, unit tests act as regression tests.

The only downside of unit tests seem to be unawareness or reluctance to code them. These problems can be exacerbated during parallel computation.

Regression tests: these tests rehearse old mistakes to prevent new coding from re-introducing errors. Regression testing is often the only type of test that can be applied to a legacy code whose architecture is not test friendly.

System or integration tests: System tests include the interaction of components in a meaningful way. A small system might represent a single operator in the algorithm. A large system might represent the whole time update. The behavior tested by a system test will make sense to a user and the criteria are is often reportable artifacts that might be used in a publication. In this regard, we have observed at least one “cultural” issue: the criterion that a test be repeatable and automatic may conflict with the desire to report results.

**NUMERICAL VERIFICATION**

One of the well-recognized and the standard verification methods of computational-fluid-dynamics (CFD) codes is based on the notion of mesh convergence. This method provides a quantitative measure of the ability of the code to respond to changes in spatial and time steps. The ratio of consecutive error norms is a perfect vehicle to catch any coding error/algorithm problem. The points needed to be considered in any mesh-convergence study include:

* Norms: L∞[[1]](#footnote-1) should be included as an ultimate diagnostic tool for local errors and worst case scenario. L2 is a more forgiving norm compared to the first error norm L1. We recommend L1 as an appropriate global metric of error.[[2]](#footnote-2)
* The convergence ratio in a very coarse grid oscillates around its main value; as the grid size is refined, convergence becomes monotonic until the mesh size reaches a point where the machine precision overtakes the truncation error of the numerical scheme. At this point error norms do not change and convergence rate is zero. Thus, convergence ratios should be checked for intermediate grid sizes (preferably at the scale of the real phenomenon).
* Although the convergence is a reliable warning of a defect, it should not be forgotten that the main goal in practice is a more accurate solver. Therefore, the superiority of methods should be assessed based on both convergence and accuracy. Accuracy metrics similarly are error norms as is discussed above, however for evaluating the accuracy error norms should be normalized by an appropriate scale of the solution.
* All of the convergence tests such as MMS, Richardson Extrapolation, could be run by a same driver. The post processing of the convergence test also could carry out with a same code for all the tests.
* Visualization of time evolution of error and results in the solution domain is a recommended strategy for debugging in cases the source of inaccuracy is obscure.

The question here is: if one wants to find accuracy and convergence ratio of a scheme in which the analytical solution is unknown (absence of analytical solution is the main motivator towards all numerical methods), what should be done? It is ideal to test a model’s correctness by comparing its numerical results with analytical solutions; however the difficulty is that there is not a general solution for the non-linear IBVP in water quality. There are some ways to deal with this problem from the simplest to the most sophisticated:

a) Comparing with a higher order code/run on a dense mesh. Downside of the method is: the benchmark code requires verification prior to the code that is subjected to the verification process.

b) Richardson Extrapolation: it is the common method for dealing with commercial packages and multidimensional complex systems (Roache and Knupp, 1993). However the drawback is that the method only checks if the solver converges and it is not able to measure where it is converging to.

Difficulties arise in Richardson EXTRAPOLATION???(BC/IC incompatibility?)

c) Method of Manufactured Solutions (MMS) (Wang and Jia, 2009), and Prescribed Solution Forcing Method (PSF) (Dee and Da Silva, 1986). The basic concept of the MMS and PSF is to compare the correctness of numerical solvers using an arbitrary manufactured function. MMS and PSF are conceptually following the same idea, although the former is more general than the latter. PSF have been used for the verification cases in which the user can not access the source code to define boundary conditions such as some groundwater codes.

**TEST SUITE DESIGN**

**General approach.** Our test suite is based on four hierarchical steps, as follows:

1) Testing of the basic components of each physical process (units) for correctness. This involves testing each sub-routine, such as the routine which coarsens the values on the mesh. One of the obvious tests is to check that the coarsen process is correctly undertaken. CCCC

2) System testing of physical processes at level 1: advection, dispersion and reaction.

3) System testing of physical processes at level 2: advection-dispersion, advection-reaction, and dispersion-reaction,

4) System testing of physical processes at level 3: advection-dispersion-reaction.

It is worth mentioning that in each category, we tested two conditions for the boundaries: a) Boundaries close to the main transport events, b) boundaries far from those events.

**Detail of tests.** We detail below the tests developed, and comment on the results.

FABIAN AND KAVEH

**Results.** FABIAN AND KAVEH

**LESSONS LEARNED AND CHALLENGES**

ELI

**CONCLUSIONS AND FINAL REMARKS**

**ACKNOWLEDGMENTS**

Mrs. Tara Smith and Dr. Francis Chung. Support by DWR.

**REFERENCES**

Abbott, M. B., Price, W. A. (Eds.), (1994), “Coastal, Estuarial, and Harbour Engineers’ Reference Book”, Chapman & Hall.

Colella, P., Puckett, E. G., (1998), *“Modern Numerical Methods for Fluid Flow”,* Class Note.

Ferziger, J. H., Peric, M., (2002), “*Computational Methods for Fluid Dynamics”,* Springer.

Fischer, B. H., List, J. E., Koh, R. C., Imberger, J., Brooks, N. H., (1979),  *“Mixing in Inland and Coastal Waters”*, Academic Press, Inc.

Fletcher, C. A. J., (1991), *“Computational Techniques for Fluid Dynamics”*, Springer.

Garcia, M. H., (Ed.), (2008), *“Sediment Engineering, Process, Measurements, Modeling, and Practice”*, Manuals and Reports on Engineering Practice No. 110, ASCE, EWRI.

Leveque, J. R., (1986), *“Intermediate Boundary Conditions for Time-Split Methods Applied to Hyperbolic Partial Differential Equations”*, Mathematics of Computation, Vol. 47 (175), pp. 37-54.

Leveque, J. R., (2002), “Finite Volume Methods for Hyperbolic Problems”, Cambridge University Press.

Oberkampf, W. L., Trucano T. G., (2002), *“Verification and Validation in Computational Fluid Dynamics”*, SANDIA REPORT, No. SAND2002-0529.

Roache, P. J., (2009), *“Fundamentals of Verification and Validation”*, Hermosa Publishers.

Roache, P. J., Knupp, P. M., (1993), *“Complete Richardson Extrapolation”*, Communication in Numerical Method in Eng. Vol. 9, pp. 365-374.

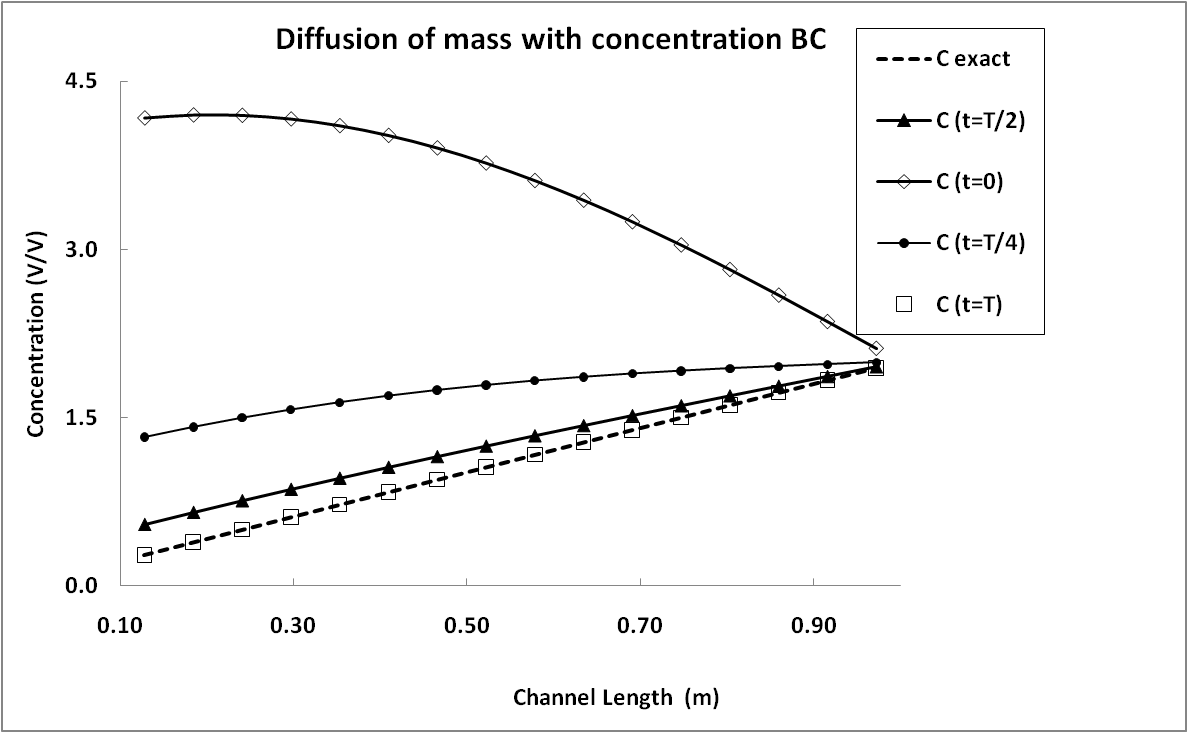
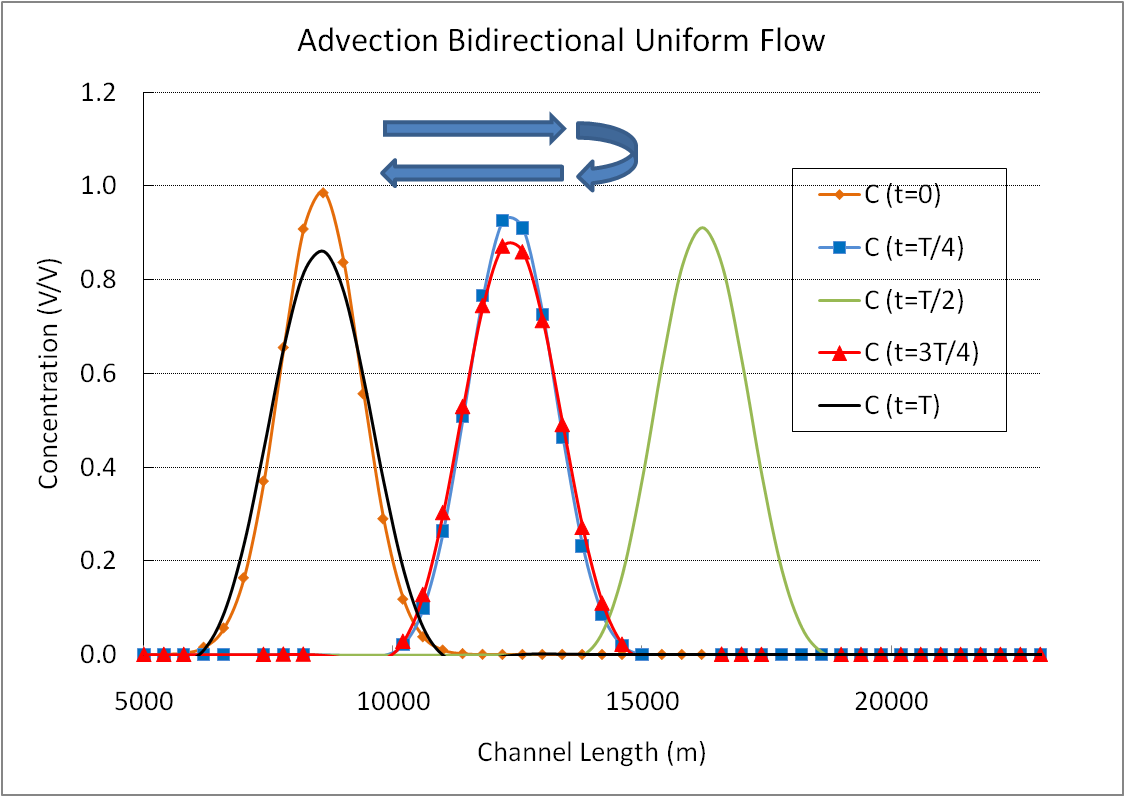
Saltzman, J., (1994), *“An Unsplit 3D Upwind Method for Hyperbolic Conservation Laws”*, J. of Computational Physics, Vol. 115, pp. 153-168.

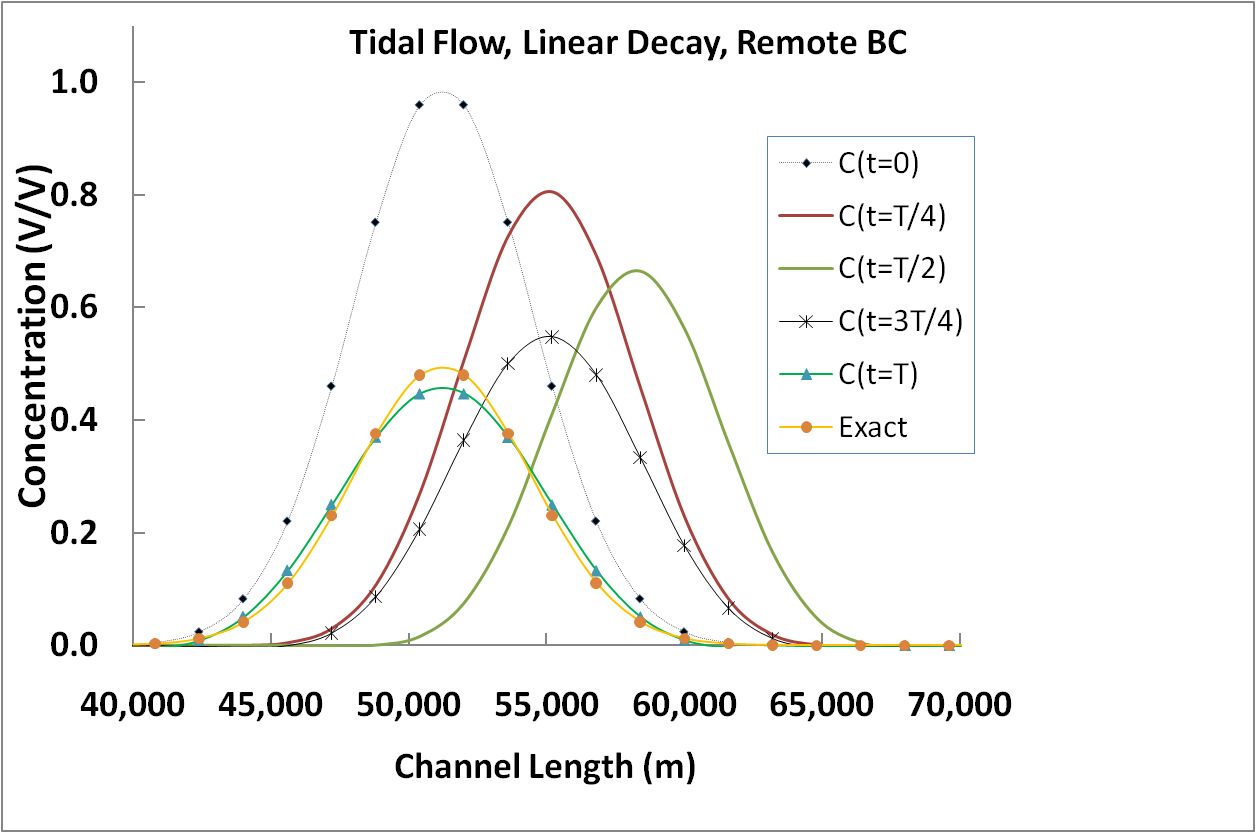
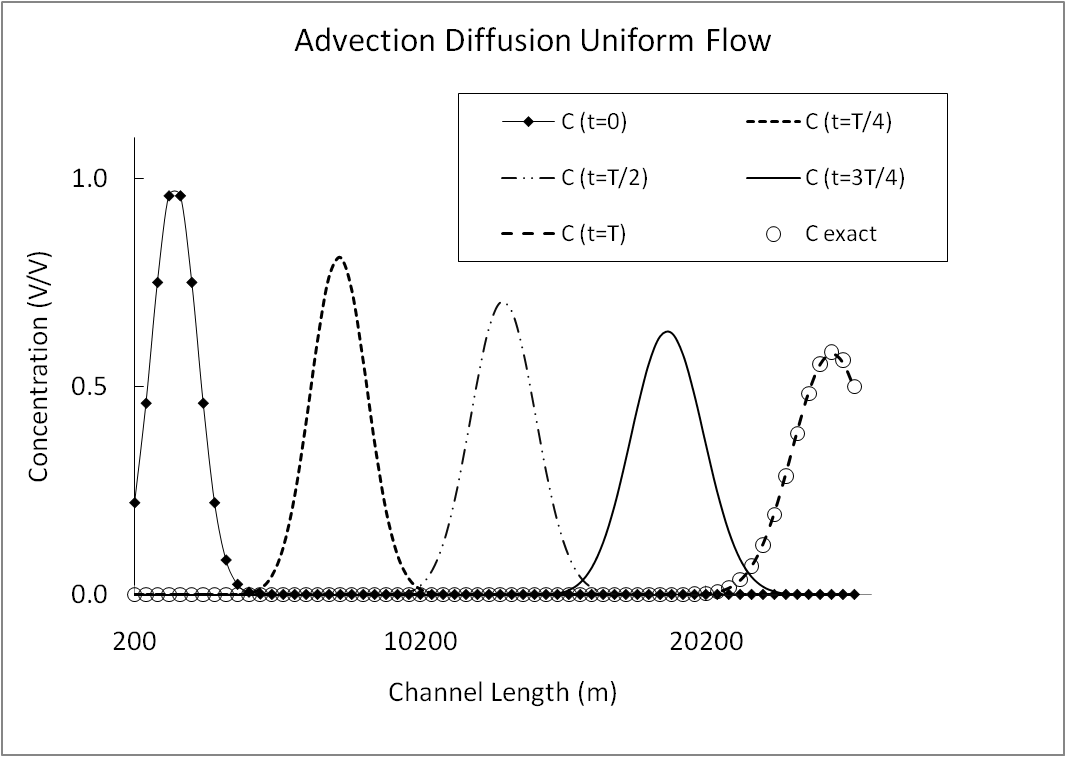
van Rijn, L. C., (1993), “*Principle of Sediment Transport in Rivers, Estuaries, and Coastal Seas”*, AQUA Publications.

Zoppou, C., Knight, J. H., (1997), “*Analytical Solution for Advection and Advection-Diffusion Equation with Spatially Variable Coefficients*”, J. Hydraulic Eng., Vol. 123(2), pp. 144-148.

**SCALING OF THE PROBLEM FOR AN ESTUARY**

The ADR solver is only working in the feasible ranges of dimensionless numbers (Peclet number and Damkohler number,) so in case the reaction rate in equation (1) should not exceed a certain limit, and generally speaking the test suit has to be designed within the natural scales of the physical problem. The assumed scales and ranges are as follows: Area~ 1000 [m2], C (0 – 0.05) [vol/vol=1], u (±0.2-2) [m/s], *u*\* isshear velocity is scaled based on where g is gravitational acceleration, *Rh* is hydraulic radius and *S* is bed slope. The longitudinal dispersion coefficient scales with:  Where *B* is width, *H* is depth and is average velocity (Abbott 1993), *K*s ~[2-150 m2/s]. Finally based on the formula suggested by Garcia and Parker (1991) reaction variation range for non-cohesive sediment in an estuary will be ( +1.0×10-3 to -3.7×10-4 [1/s]). That is to mention the length scale in the scaling process assumed the same length of spatial discretization where needed.





1. ,,, where *v*= U num - U exact [↑](#footnote-ref-1)
2. It is proven that kL∞ ≤ L2 ≤ L1 ≤ L∞ where k is a constant and 0<k<1, here norms are assumed to be scaled. [↑](#footnote-ref-2)