Implementation of Explicit and Implicit Runge-Kutta methods to reduce the computational cost of pollutant transport modeling

Ioannis Charis[a,*,†]

[a]*Geo engineering section, Faculty of Civil Engineering and Geosciences, Delft University of Technology, Stevinweg 1, 2618 CN Delft*
[*]*E-mail: I.Charis@student.tudelft.nl*
[†]*Student number: 4314379*

Abstract-Heimovaara et al. (2015) have developed a solute transport model (STM) so as to paint a dire portrait of the leachate generation and transport in the Braambergen landfill (located in the western portion of the Netherlands) with the ultimate goal of quantifying the chemical composition of the landfill's overall emissions. In fact, the model simulates the Chloride (Cl^-) concentration in the outflowing from the landfill leachate and the mass of Cl^- that remains in the landfill by using an ordinary differential equation (ODE) that describes dual-porosity, mobile-immobile solute transport phenomena. This environmental model, developed in MATLAB, simulates these state variables by solving the aforementioned ODE by means of the well-known ode45 solver in MATLAB. In spite of that, the computational expense of the ode45 solver does not seem to be identical (~20-30 seconds of calculations). As is widely known, various ODE solvers usually require different computing times. Namely, the selection of the method for solving such insightful equations most often affects the time efficiency of ODE-based models. In the light of these considerations, this paper exploits the robustness of several explicit and implicit Runge-Kutta methods to solve the STM's ODE in an effort to make this model more cost-effective. By virtue of the results, the fourth order explicit Runge-Kutta method along with the fourth and sixth order implicit Runge-Kutta methods have successfully managed to drastically reduce the computational cost of the STM (~4-9 seconds of calculations).

Keywords-STM, ODE, Cl^-, ode45, Explicit, Implicit, Runge-Kutta

1. Introduction and preliminaries

Computerized forecasting models utilize the predictive power of ODEs to simulate state variables that portray the dynamics of natural systems. In other words, making it simpler and more comprehensive, the model states delineate the properties of a real-world system that evolve over time and they are commonly expressed in the form of ODEs as follows (Gill 1951; Saito and Mitsui, 1996; Kloeden et al. 1999; Butcher 2007):

$$\frac{\partial y}{\partial t} = f(y(t), u(t), t) \tag{1}$$

Equation (1) is the classical definition of a first order ODE, where $\partial y(t)/\partial t$ represents the rate at which y changes over time t, $y(t) \in \mathbb{R}^n$ illustrates the state of the system at time t, n is the number of states, $u(t) \in \mathbb{R}^m$ is the input vector at time t (such as model parameters, model forcing data, etc.), m is the number of inputs , and finally, the function f contains the mathematical rules that determine how y varies over time.

In this regards, Runge-Kutta (RK) methods are attracting widespread interest due to their ability to powerfully solve ODEs. In particular, the past decade has seen a renewed importance in improving the accuracy or the flexibility (to accommodate problems of diverse nature) of the RK methods (Kaar 2006; MackCormack 2011; Durran et al. 2012; Haas 2013; Jonson 2013). Among existing RK methods, the explicit RK (ERK) and the implicit RK (IRK) methods are widely used for solving non-stiff and stiff ODEs, respectively[1] (Kwizak and Robert, 1971; Alexander 1977; Hairer et al. 1993; Verner 1996; Ascher et al. 1997; Hairer and Warrer, 2010). As a general rule, IRK methods perform better for large time steps, by contrast, ERK methods are more effective by using small time steps, which is however regarded a time-consuming process. It is also broadly recognized that IRK methods are most often quite problematic for solving non-linear ODEs (Babolian and Mordad, 2011; Pulliam 2011). Actually, the fundamental problem in the implementation of IRK methods lies in the hardness of solving implicitly non-linear system of equations.

[1]*Stiff ODEs are the equations that include a term that decays exponentially to zero as time increases.*

On another note, the advection-dispersion equation (ADE) for a single porosity porous medium is often used to delineate the transport of solutes. Nevertheless, there have been several dissenters against the ADE model (Hawsa et al., 2005; Martinez et al., 2010; Russo et al., 1989; Hu et al., 2014) that is most of the times inappropriate in describing transport in heterogeneous and fractured media. In the STM, Heimovaara, et al. (2015) selected to describe the transport of pollutants through the Wieringermeer landfill by making use of an alternative time based dual porosity model (DPM) combined with the mobile-immobile (MIM) principle. In a few words, the DPM-MIM concept assumes the existence of mobile and immobile categories of water (i.e. leachate) in the landfill. With other words, it is conceptualized that the mobile water flows easily out of the landfill whereas the stationary water is continually accumulated inside the landfill body.

In a similar manner, it can be seen that the chemical compounds that are present in the landfill are dissolved either in the mobile or in the immobile water content of the landfill. It can, therefore, be posited that a portion of these toxic particles either remains in the landfill or escapes from there. Under these assumptions, the STM exploits interdependent equations as a means of predicting a) the Cl^- concentration dissolved in the mobile contaminated water that leaves the landfill (state) and b) the Cl^- mass present in the motionless landfill water that is stored within the landfill (state). Given all these points, these model states are estimated with the way indicated below. First of all, the joint mobile-immobile mass of Cl^- is estimated using the following ODE:

$$\frac{\partial F(M_M, M_{IM})}{\partial t} = -k_{ex} \cdot \mathcal{H} \cdot F(M_M, M_{IM}) \tag{2}$$

In which, M_M (kg) represents the mass of Cl^- flowing out of the landfill and M_{IM} (kg) is the stored mass of Cl^- in the landfill. Moreover, $\mathcal{H}$ is a non-linear function that propagates M_M and M_{IM} forward in time and k_{ex} (input parameter) illustrates the interaction between the mobile and immobile Cl^- particles. Furthermore, F is a function that describes the relationship between the interdependent state variables M_M and M_{IM}. Thereafter, the mobile concentration of Cl^- [C_M(mg/L)] is obtained with ease as follows:

$$C_M = \frac{M_M}{V_M} \tag{3}$$

In equation (2), V_M (m^3/day) and V_{IM} (m^3/day) are the mobile and immobile volumes of water in the landfill that are always assumed to be known. The derivation of equation (2) is included in appendix A of this paper.

Another important thing to point out is that the amount of leachate which is produced in the Braambergen landfill and then escapes from there is collected from a drainage system that is installed across the landfill base. Thereafter, the leachate from this drainage well is daily pumped out from two pump pits installed in the north and south part of the landfill. In addition, the Cl^- concentration (mg/L) present in leachate samples (obtained from the north and south part of the landfill) was measured twice or thrice per month between the period of 1 August 2012 and 1 May 2015. Namely, this means that there is available quantitative information regarding the modeled mobile concentration of Cl^- (C_M).

In this study, in the STM, the estimation of M_M and M_{IM} is achieved by solving equation (2) with the use of a) the fourth order ERK method (ERK4), b) the second order IRK (IRK2), c) the implicit two-stage Gauss-Legendre method of order four (GL4), and finally, d) the implicit three-stage Gauss-Legendre method of order six (GL6). After that, the remaining state of interest C_M is calculated according to equation (3).

With all these in mind, it can be said that the intention of this research is primarily to implement the aforementioned ERK and IRK methods for the estimation of C_M and M_{IM} with the STM. Secondly, and more importantly, the results obtained by these RK methods are compared with each other and with the results provided by the reference method (ode45 solver) with regard to their efficiency to minimize the computational cost of this environmental model. Moreover, the current paper demonstrates the feasibility of solving non-linear ODEs (such as equation (2)) using IRK schemes. In conjunction with the above considerations, the estimates of the

mobile concentration of Cl^- (C_M) are compared with available observations so as to assess the reliability of the solutions provided by the implemented methods.

The remainder of this paper is structured as follows. The second section contains a concise description of the RK methods. Then, the third section detailedly outlines the construction and implementation of the previously mentioned RK schemes for the estimation of the STM state variables. Thereafter, the fourth section explains the results of this research, and finally, the conclusions are drawn in the fifth section.

2. Background

2.1. Introduction to RK methods

As alluded to previously, the RK methods are used to solve ODEs that are given by equation (1) by preconditioning that $y_0 = y(t_0)$, where y_0 illustrates the solution of y at the initial time t_0. As a matter of fact, the solution of equation (1) is accomplishable by means of ERK and IRK methods as follows (Gill 1951; Saito and Mitsui, 1996; Kloeden et al. 1999):

$$y_{n+1} = y_n + dt \cdot \sum_{i=1}^{s} b_i \cdot k_i \tag{4}$$

In which, y_{n+1} is the RK approximation of $y(t_{n+1})$ that is computed based on the present value y_n and the weighted average of the $k_i \cdot b_i$ increments, multiplied by the size of the time interval, dt. In ERK methods, the k values from equation (4) are calculated as follows:

$$k_1 = f(t_n, y_n) \tag{5}$$

$$k_2 = f(t_n + c_2 \cdot dt, y_n + dt \cdot (a_{21} \cdot k_1)) \tag{6}$$

Likewise, the rest of k values are obtained as such:

$$k_s = f(t_n + c_s \cdot dt, y_n + dt \cdot (a_{s1} \cdot k_1 + a_{s2} \cdot k_2 + \ldots + a_{s,s-1} \cdot k_{s-1})) \tag{7}$$

In equation (7), the integer s denotes the number of stages, the coefficient a_{ij} (for $1 \le j < i \le s$) is known as the RK matrix, and b_i (for $i = 1, 2, ..., s$) and c_i (for $i = 2, 3, ..., s$) are known as the weights and the nodes, respectively. These coefficients are rearranged in the so-called Butcher tableau (Butcher 2007). In ERK methods, the Butcher tableau is given as shown in Table 1:

Table 1
Butcher tableau in ERK methods.

0					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
⋮	⋮	⋮	⋱		
c_s	a_{s1}	a_{s2}	...	$a_{s,s-1}$	
	b_1	b_2	...	b_{s-1}	b_s

In IRK methods, the k values are computed according to the following equation:

$$k_i = f\left(t_n + c_i \cdot dt\,,\, y_n + dt \cdot \sum_{j=1}^{s} a_{ij} \cdot k_j\right),\quad j = 1, \ldots, s \tag{8}$$

Where the Butcher tableau in the IRK methods is given as indicated in Table 2:

Table 2
Butcher tableau in IRK methods.

0	a_{11}	a_{12}	...	a_{1s}
c_2	a_{21}	a_{22}	...	a_{2s}
c_3	a_{31}	a_{32}	...	a_{3s}
⋮	⋮	⋮	⋱	⋮
c_s	a_{s1}	a_{s2}	...	a_{ss}
	b_1	b_2	...	b_s

By comparing Table (1) and Table (2), one sees immediately that the Butcher tableau is constructed in a different manner in the ERK and IRK methods. Actually, the coefficient matrix a_{ij} of an ERK method is lower triangular. On the contrary, the coefficient matrix a_{ij} of an IRK method is clearly not triangular. This discrepancy yields different solutions for the k values shown in equations (7) and (8), which in turn leads to different solutions of the ODEs expressed by equation (1).

2.1.1. The explicit fourth order RK method

In the ERK4 method, equation (1) is solved as follows:

$$y_{n+1} = y_n + \frac{dt}{6} \cdot \left(k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4\right) \tag{9}$$

With, $t_{n+1} = t_n + dt$, and $k_1, \ldots, k_4$ are defined as shown below:

$$k_1 = f\left(t_n, y_n\right) \tag{10}$$

$$k_2 = f\left(t_n + \frac{dt}{2}, y_n + \frac{dt}{2} \cdot k_1\right) \tag{11}$$

$$k_3 = f\left(t_n + \frac{dt}{2}, y_n + \frac{dt}{2} \cdot k_2\right) \tag{12}$$

$$k_4 = f\left(t_n + dt\,,\, y_n + dt \cdot k_3\right) \tag{13}$$

Here, y_{n+1} is the ERK4 approximation of $y(t_{n+1})$ and is calculated as the sum of the current value y_n and the weighted average of the $k_1, \ldots, k_4$ increments, where each increment is the product of the time interval, dt, and an estimated rate of change that is expressed by function f on the right-hand side of the ODE. More analytically, these k values are defined as such:

a) k_1 is the slope at the beginning of the time interval dt, using y_n (Euler's method).

b) k_2 is the increment based on the slope at the midpoint of the interval dt, using $y_n + \frac{dt}{2} \cdot k_1$.

c) k_3 is the increment based on the slope at the midpoint, using $y_n + \frac{dt}{2} \cdot k_2$.

d) k_4 is an estimate of the slope at the endpoint, using $y_n + dt \cdot k_3$.

2.1.2. The implicit second order RK method

The IRK2 method is based on the trapezoidal rule and its Butcher tableau is defined as presented in Table 3:

Table 3
Butcher tableau in the IRK2 method.

0	0	0
1	$\frac{1}{2}$	$\frac{1}{2}$
	$\frac{1}{2}$	$\frac{1}{2}$

In this case, equation (1) is solved according to:

$$y_{n+1} = y_n + \frac{1}{2} \cdot dt \cdot k_1 + \frac{1}{2} \cdot dt \cdot k_2 \tag{14}$$

In addition, k_1 and k_2 are given by:

$$k_1 = f(t_n, y_n) \tag{15}$$

$$k_2 = f(t_n + dt, \underbrace{y_n + \frac{1}{2} \cdot dt \cdot k_1 + \frac{1}{2} \cdot dt \cdot k_2}_{y_{n+1}}) = f(t_n + dt, y_{n+1}) \tag{16}$$

2.1.3. Gauss-Legendre methods

A particular class of IRK methods is the Gauss-Legendre (GL) method that is based on the GL quadrature. A review of the recent literature suggests that there are three GL methods (Haas 2013; Jones 2013). Firstly, the implicit midpoint rule that is the one-stage GL method. Secondly, the GL4 method that is a two-stage method, and finally, the GL6 method that is known to be a three-stage GL method. As it is known up to now, there are GL schemes up to order six. The construction of GL methods higher than 6 is nearly infeasible owing to the zeros of Legendre polynomials of order 4 and above that are unduly complicated.

2.1.3.1. Fourth order Gauss-Legendre method

The GL4 has Butcher tableau of the next form:

Table 4
Butcher tableau in the GL4 method.

$\frac{1}{2} - \frac{1}{6} \cdot \sqrt{3}$	$\frac{1}{4}$	$\frac{1}{4} - \frac{1}{6} \cdot \sqrt{3}$
$\frac{1}{2} + \frac{1}{6} \cdot \sqrt{3}$	$\frac{1}{4} + \frac{1}{6} \cdot \sqrt{3}$	$\frac{1}{4}$
	$\frac{1}{2}$	$\frac{1}{2}$

With no doubt, it can be asserted that the Butcher tableau above is not diagonal. Consequently, it can be deduced that the *k* values are not independent of each other. On account of this fact, the GL4 method is considered an IRK method, or as it is frequently called a collocation method. In this GL method, the *k* values are given by equations (17) and (18).

$$k_1 = f\left(t_n + \left(\frac{1}{2} - \frac{1}{6}\cdot\sqrt{3}\right)\cdot dt\,,\, y_n + \frac{1}{4}\cdot dt\cdot k_1 + \left(\frac{1}{4} - \frac{1}{6}\cdot\sqrt{3}\right)\cdot dt\cdot k_2\right) \tag{17}$$

$$k_2 = f\left(t_n + \left(\frac{1}{2} + \frac{1}{6}\cdot\sqrt{3}\right)\cdot dt\,,\, y_n + \left(\frac{1}{4} + \frac{1}{6}\cdot\sqrt{3}\right)\cdot dt\cdot k_1 + \frac{1}{4}\cdot dt\cdot k_2\right) \tag{18}$$

It is generally agreed that the GL4 method is fairly difficult due to the fact that the *k* values from equations (17) and (18) are dependent of each other. This also implies that it is only attainable to estimate these *k* values by means of solving equations (17) and (18) with some iterative method. In doing so, an initial guess for each *k* should be assumed. Then, this initial guess is replaced into equations (17) and (18) in order to estimate the next *k* values. This operation is performed iteratively with the purpose of obtaining convergent *k* values. If this condition is satisfied, the solution of equation (1) is then obtained according to the following equation:

$$y_{n+1} = y_n + \frac{1}{2}\cdot dt\cdot k_1 + \frac{1}{2}\cdot dt\cdot k_2 \tag{19}$$

2.1.3.2. Sixth order Gauss-Legendre method

The butcher tableau of this method is illustrated in Table 5:

Table 5
Butcher tableau in the GL6 method.

$\frac{1}{2} - \frac{1}{10}\cdot\sqrt{15}$	$\frac{5}{36}$	$\frac{2}{9} - \frac{1}{15}\cdot\sqrt{15}$	$\frac{5}{36} - \frac{1}{30}\cdot\sqrt{15}$
$\frac{1}{2}$	$\frac{5}{36} + \frac{1}{24}\cdot\sqrt{15}$	$\frac{2}{9}$	$\frac{5}{36} - \frac{1}{24}\cdot\sqrt{15}$
$\frac{1}{2} + \frac{1}{10}\cdot\sqrt{15}$	$\frac{5}{36} + \frac{1}{30}\cdot\sqrt{15}$	$\frac{2}{9} + \frac{1}{15}\cdot\sqrt{15}$	$\frac{5}{36}$
	$\frac{5}{18}$	$\frac{4}{9}$	$\frac{5}{18}$

Following this, the *k* values from the GL6 method are found as follows:

$$k_1 = f\left(t_n + \left(\frac{1}{2} - \frac{1}{10}\cdot\sqrt{15}\right)\cdot dt\,,\, y_n + \frac{5}{36}\cdot dt\cdot k_1 + \left(\frac{2}{9} - \frac{1}{15}\cdot\sqrt{15}\right)\cdot dt\cdot k_2 + \left(\frac{5}{36} - \frac{1}{30}\cdot\sqrt{15}\right)\cdot dt\cdot k_3\right) \tag{20}$$

$$k_2 = f\left(t_n + \frac{1}{2}\cdot dt\,,\, y_n + \left(\frac{5}{36} + \frac{1}{24}\cdot\sqrt{15}\right)\cdot dt\cdot k_1 + \frac{2}{9}\cdot dt\cdot k_2 + \left(\frac{5}{36} - \frac{1}{24}\cdot\sqrt{15}\right)\cdot dt\cdot k_3\right) \tag{21}$$

$$k_3 = f\left(t_n + \left(\frac{1}{2} + \frac{1}{10}\cdot\sqrt{15}\right)\cdot dt\,,\, y_n + \left(\frac{5}{36} + \frac{1}{30}\cdot\sqrt{15}\right)\cdot dt\cdot k_1 + \left(\frac{2}{9} + \frac{1}{15}\cdot\sqrt{15}\right)\cdot dt\cdot k_2 + \frac{5}{36}\cdot dt\cdot k_3\right) \tag{22}$$

In the GL6 method, the *k* values are estimated similarly to the GL4 method using the afore-described iterative method. Provided that, equation (23) illustrates the GL6 solution of the ODE given by equation (1).

$$y_{n+1} = y_n + \frac{5}{18} \cdot dt \cdot k_1 + \frac{4}{9} \cdot dt \cdot k_2 + \frac{5}{18} \cdot dt \cdot k_3 \tag{23}$$

3. Methodology

This section serves as a window to an understanding of how the ERK4, IRK2, GL4 and GL6 methods are used to estimate the STM states C_M and M_{IM}. Before explaining the implementation of these methods it is worth mentioning that the *sparse* command in MATLAB has been applied to the matrix $\mathfrak{H}$ from equation (2). By doing so, any zero elements from this matrix are squeezed out resulting in less memory consumption (Shampine and Reichelt, 1997).

3.1. Implementation of ERK4

The k values from the ERK4 scheme are readily calculated without the need to use any iterative method for estimating them. In consequence, this fact renders very easy the process of estimating C_M and M_{IM}. Under these considerations, the computation of C_M and M_{IM} using the ERK4 method is detailedly described in Algorithm 1.

3.2. Implementation of IRK2

The implementation of the IRK2 method is grounded on the following simplifying considerations. First of all, the solution of equation (2) is written in a form similar to equation (14) (from the IRK2 scheme) as such:

$$F\left(M_{M_{n+1}}, M_{IM_{n+1}}\right) = F\left(M_{M_n}, M_{IM_n}\right) + \frac{1}{2} \cdot dt \cdot k_1 + \frac{1}{2} \cdot dt \cdot k_2 \tag{24}$$

In what follows, the k_1 and k_2 values are computed by substituting equations (15) and (16) into equation (2) as such:

$$k_1 = -\, dt \cdot k_{ex} \cdot \mathcal{H} \cdot F\left(M_{M_n}, M_{IM_n}\right) \tag{25}$$

$$k_2 = -\, dt \cdot k_{ex} \cdot \mathcal{H} \cdot F\left(M_{M_{n+1}}, M_{IM_{n+1}}\right) \tag{26}$$

Then, the insertion of equations (25) and (26) into equation (24) results in:

$$F\left(M_{M_{n+1}}, M_{IM_{n+1}}\right) = F\left(M_{M_n}, M_{IM_n}\right) + \frac{1}{2} \cdot\left(-dt \cdot k_{ex} \cdot \mathcal{H} \cdot F\left(M_{M_n}, M_{IM_n}\right)\right) + \frac{1}{2} \cdot\left(-\, dt \cdot k_{ex} \cdot \mathcal{H} \cdot F\left(M_{M_{n+1}}, M_{IM_{n+1}}\right)\right)$$

$$F\left(M_{M_{n+1}}, M_{IM_{n+1}}\right) = \frac{1 - \frac{1}{2} \cdot dt \cdot k_{ex} \cdot \mathcal{H}}{1 + \frac{1}{2} \cdot dt \cdot k_{ex} \cdot \mathcal{H}} \cdot F\left(M_{M_n}, M_{IM_n}\right) \tag{27}$$

Here, equation (27) represents the analytical IRK2 solution of equation (2). In this manner, M_M and M_{IM} are obtained from equation (27) and C_M is then calculated according to equation (3).

3.3. Implementation of GL4

In the GL4 method, the model states are calculated by using an iterative technique for estimating the k_1 and k_2 values as told above. The current study opted for the notorious Newton-Raphson method to calculate iteratively the k_1 and k_2 values. In the approach presented here, an initial guess for k_1 and k_2 at time t_n is obtained with Euler's method as such:

$$k_{1_n} = f(t_n, F(M_{M_n}, M_{IM_n})) \tag{28}$$

$$k_{2_n} = f(t_n, F(M_{M_n}, M_{IM_n})) \tag{29}$$

Thereafter, the computation of k_1 and k_2 values is attainable by writting equations (17) and (18) as follows:

$$k_1 = \underbrace{f(t_n + (\frac{1}{2} - \frac{1}{6}\cdot\sqrt{3})\cdot dt\,,\, y_n + \frac{1}{4}\cdot dt\cdot k_1 + (\frac{1}{4} - \frac{1}{6}\cdot\sqrt{3})\cdot dt\cdot k_2)}_{f_1(k_1,k_2)} = f_1(k_1,k_2) \tag{30}$$

$$k_2 = \underbrace{f(t_n + (\frac{1}{2} + \frac{1}{6}\cdot\sqrt{3})\cdot dt\,,\, y_n + (\frac{1}{4} + \frac{1}{6}\cdot\sqrt{3})\cdot dt\cdot k_1 + \frac{1}{4}\cdot dt\cdot k_2)}_{f_2(k_1,k_2)} = f_2(k_1,k_2) \tag{31}$$

Then, equations (30) and (31) are defined as such:

$$F_1(k_1,k_2) = k_1 - f_1(k_1,k_2) = 0 \tag{32}$$

$$F_2(k_1,k_2) = k_2 - f_2(k_1,k_2) = 0 \tag{33}$$

The k_1 and k_2 values are obtained by solving equations (32) and (33). In the Newton-Raphson method, this is achieved according to:

$$k_{1_{\ell+1}} = k_{1_\ell} - \frac{F_1(k_1,k_2)}{F'_1(k_1,k_2)} \tag{34}$$

$$k_{2_{\ell+1}} = k_{2_\ell} - \frac{F_2(k_1,k_2)}{F'_2(k_1,k_2)} \tag{35}$$

Equations (34) and (35) are iteratively computed for ℓ = 1, ... ,L number of iterations, at every time step, until the difference between subsequent k_1 and k_2 values are within some pre-specified error tolerance. The derivative terms in equations (34) and (35) are computed by using the complex-step derivative approach as indicated below (Martins et al. 2003):

$$F'_1(k_1,k_2) = imag(\frac{F_1(k_1 + i\cdot h\,,\, k_2 + i\cdot h)}{h}) \tag{36}$$

$$F'_2(k_1,k_2) = imag(\frac{F_2(k_1 + i\cdot h\,,\, k_2 + i\cdot h)}{h}) \tag{37}$$

Here, $i = \sqrt{-1}$ that represents a pure imaginary step and h is a real number that is usually set to very small values; i.e. $h \simeq$ eps. After the convergence of k_1 and k_2 values, M_M and M_{IM} are computed at every new time step t_{n+1} as follows:

$$F\left(M_{M_{n+1}}, M_{IM_{n+1}}\right) = F\left(M_{M_n}, M_{IM_n}\right) + \frac{1}{2}\cdot dt\cdot k_1 + \frac{1}{2}\cdot dt\cdot k_2 \tag{38}$$

Notice that equation (38) illustrates the GL4 solution of equation (2). Accordingly, it is easy to estimate C_M using equation (3). The afore-described implementation of the GL4 method is more rigorously illustrated in Algorithm 3.

Algorithm 1
Implementation of the ERK4 method for the calculation of the STM states.

Step 1: Set the time interval, *dt*.

Step 2: Set $M_{M_0} = M_M(t_0)$ and $M_{IM_0} = M_{IM}(t_0)$.

-FOR t_n = 1, 2, ...

Step 3: Compute k_1, k_2, k_3 and k_4 according to equations (10), (11), (12) and (13), respectively.

Step 4: Update the solution of equation (2) using equation (9) and obtain the M_M and M_{IM} estimates.

Step 5: Given that V_M is known calculate C_M by making use of equation (3).

-END FOR

Note: $M_{M_0} = M_M(t_0)$ and $M_{IM_0} = M_{IM}(t_0)$ are the solutions of M_M and M_{IM} at the initial time t_0.

Algorithm 2
Implementation of the GL4 method for the calculation of the STM states.

Step 1: Set the time interval, *dt*.

Step 2: Set $M_{M_0} = M_M(t_0)$ and $M_{IM_0} = M_{IM}(t_0)$.

Step 3: Set the initial values of k_1 and k_2 according to equations (28) and (29) that illustrate the initial iteration ℓ = 1.

Step 4: Set an initial error value that represents the difference between the k_1 and k_2 values; e.g. *error* = 1.

Step 5: Set an error tolerance value; e.g. *tol* = $1\cdot 10^{-08}$.

-FOR t_n = 1, 2, ...

-IF *error* > *tol*

Step 6: Update the k_1 and k_2 values according to equations (34) and (35), respectively.

Step 7: Update the difference; i.e. *error* between the k_1 and k_2 values that are estimated in Step 6 and from a previous iteration.

$\ell = \ell + 1$

-ELSE IF *error* ≤ *tol* go to steps 8 and 9.

-END IF

Step 8: Update the solution of equation (2) using equation (38) and obtain the M_M and M_{IM} estimates.

Step 9: Given that V_M is known calculate C_M by making use of equation (3).

-END FOR

Note: $M_{M_0} = M_M(t_0)$ and $M_{IM_0} = M_{IM}(t_0)$ are the solutions of M_M and M_{IM} at the initial time t_0.

3.4. Implementation of GL6

Finally, the GL6 approach is implemented in a manner similar to the GL4 method, with the only difference that an additional *k* increment is involved in the solution of equation (2) as outilined in section 2. In this particular case, the k_1, k_2 and k_3 values are calculated based on the above-mentioned Newton-Raphson method by solving the next equations:

$$k_{1_{\ell+1}} = k_{1_\ell} - \frac{F_1(k_1, k_2, k_3)}{F'_1(k_1, k_2, k_3)} \tag{39}$$

$$k_{2_{\ell+1}} = k_{2_\ell} - \frac{F_2(k_1, k_2, k_3)}{F'_2(k_1, k_2, k_3)} \tag{40}$$

$$k_{3_{\ell+1}} = k_{3_\ell} - \frac{F_3(k_1, k_2, k_3)}{F'_3(k_1, k_2, k_3)} \tag{41}$$

Where

$$F_1(k_1, k_2, k_3) = k_1 - f_1(k_1, k_2, k_3) = 0 \tag{42}$$

$$F_2(k_1, k_2, k_3) = k_2 - f_2(k_1, k_2, k_3) = 0 \tag{43}$$

$$F_3(k_1, k_2, k_3) = k_3 - f_3(k_1, k_2, k_3) = 0 \tag{44}$$

and

$$F'_1(k_1, k_2, k_3) = imag\left(\frac{F_1(k_1 + i \cdot h,\ k_2 + i \cdot h, k_3 + i \cdot h)}{h}\right) \tag{45}$$

$$F'_2(k_1, k_2, k_3) = imag\left(\frac{F_2(k_1 + i \cdot h,\ k_2 + i \cdot h, k_3 + i \cdot h)}{h}\right) \tag{46}$$

$$F'_3(k_1, k_2, k_3) = imag\left(\frac{F_3(k_1 + i \cdot h,\ k_2 + i \cdot h, k_3 + i \cdot h)}{h}\right) \tag{47}$$

Likewise, the initial guess for k_1, k_2 and k_3 is obtained with Euler's method at each time step t_n as follows:

$$k_{1_n} = f(t_n, F(M_{M_n}, M_{IM_n})) \tag{48}$$

$$k_{2_n} = f(t_n, F(M_{M_n}, M_{IM_n})) \tag{49}$$

$$k_{3_n} = f(t_n, F(M_{M_n}, M_{IM_n})) \tag{50}$$

After ensuring the convergence of k_1, k_2 and k_3 using the Newton-Raphson method, M_M and M_{IM} are calculated at every new time step t_{n+1} as such:

$$F(M_{M_{n+1}}, M_{IM_{n+1}}) = F(M_{M_n}, M_{IM_n}) + \frac{5}{18} \cdot dt \cdot k_1 + \frac{4}{9} \cdot dt \cdot k_2 + \frac{5}{18} \cdot dt \cdot k_3 \tag{51}$$

In the end, C_M is found according to equation (3). The entire implementation of the GL6 method for the calculation of C_M and M_{IM} is extensively indicated in Algorithm 3.

Algorithm 3
Implementation of the GL6 method for the calculation of the STM states.

Step 1: Set the time interval, *dt*.

Step 2: Set $M_{M_0} = M_M(t_0)$ and $M_{IM_0} = M_{IM}(t_0)$.

Step 3: Set the initial values of k_1, k_2 and k_3 according to equations (48), (49) and (50) that illustrate the initial iteration $\ell = 1$.

Step 4: Set an initial error value that represents the difference between the k_1, k_2 and k_3 values; e.g. *error* = 1.

Step 5: Set an error tolerance value; e.g. $tol = 1 \cdot 10^{-08}$.

-FOR t_n = 1, 2, ...

-IF *error* > *tol*

Step 6: Update the k_1, k_2 and k_3 values according to equations (39), (40) and (41), respectively.

Step 7: Update the difference; i.e. *error* between the k_1, k_2 and k_3 values that are estimated in Step 6 and from a previous iteration.

-ELSE IF *error* ≤ *tol* go to steps 8 and 9.

-END IF

Step 8: Update the solution of equation (2) using equation (51) and obtain the M_M and M_{IM} estimates.

Step 9: Given that V_M is known calculate C_M by making use of equation (3).

- END FOR

Note: $M_{M_0} = M_M(t_0)$ and $M_{IM_0} = M_{IM}(t_0)$ are the solutions of M_M and M_{IM} at the initial time t_0.

4. Results and discussion

In the present research, the ode45 reference solver along with the ERK4, IRK2, GL4 and GL6 methods are used to solve equation (2). In this manner, the STM simulates the states of interest C_M and M_{IM}. The RK methods have been implemented for three different time steps, *dt* (*dt* = 1/8, 1/4 and 1), with a view to assessing how the time step size affects the quality of the computational results and the efficiency of the simulations. Additionally, the current study compares the simulation results of this prediction model using the above-mentioned RK methods and the reference ode45 solver so as to determine to which extent the RK methods have reduced or increased the computational cost of the model. Further insight into the ode45 MATLAB solver is given in appendix B of this paper.

In this context, the Mean Absolute Error (MAE) has been selected to assess the performance of the STM with regard to the C_M estimates obtained with the implemented RK methods (y_n) and the reference ode45 solver (z_n) at time t_n = 1, … ,*N*. Given this, the MAE is defined as such:

$$\text{MAE} = |y_n - z_n| \tag{52}$$

Another important thing to highlight is that m = 3 input parameters expressed as C_{ini}, k_{ex} and N_{ex} (see their description in Table 6) have to be found prior to running the STM. In this prediction model, these parameters have been calibrated (see Heimovaara et al. 2015) through the model performance optimization that is effectuated by

comparing simulated and measured data of Cl^- concentrations in the mobile landfill water. This was accomplishable by utilizing the robustness of the DREAM (DiffeRential Evolution Adaptive Metropolis) stochastic optimization technique that is based on the Bayesian inference scheme. In particular, this method approximates the posterior distribution of parameters providing acceptable and reasonable values of parameters that describe the simulated and measured dataset. More details on this topic can be found in Laloy and Vrugt (2012).

In this study, 600000 model evaluations were invoked where DREAM provided two separate parameter sets where each one consists of nearly 500 different values of C_{ini}, k_{ex} and N_{ex} (see Figs. 1 and 2). These two parameter sets are indicative of the intricate processes that occur in the north (hereafter termed as Case 1) and the south part (hereafter termed as Case 2) of the Braambergen landfill, where from both these parts the measurements of the mobile Cl^- concentration have been obtained as highlighted in Section 1. It is, therefore, obvious that the STM focuses on estimating the state variables C_M and M_{IM} in two different locations (north and south) of the Braambergen landfill. In the present study, the average values of these calibrated parameters (C_{ini}, k_{ex} and N_{ex}) from both parameter sets (Case 1 and Case 2) are chosen to run twice the STM (see Table 6).

Table 6
Average values of the Case 1 and Case 2 DREAM-derived parameters C_{ini}, k_{ex} and N_{ex}.

Mean Values of Case	C_{ini}	k_{ex}	N_{ex}
1	3.65	-1.72	203
2	4.25	-6.15	394

Note: C_{ini} is the initial Cl^- concentration in the landfill in mol/m^3, k_{ex} is the interaction between mobile and immobile chemical solids, and N_{ex} are the days of interaction between the landfill water and the chemical compounds.

A considerable inconsistency is exhibited by the DREAM-derived parameter values between Case 1 and Case 2 as Figs. 1 and 2 show. In the first place, C_{ini} converges to different values in both cases (Case 1 and Case 2). It is also evident that k_{ex} centers around a target value in Case 1, but, however, it does not reach convergence in Case 2. On the other hand, N_{ex} does not converge in Case 1 whereas it converges to a specific value in Case 2.

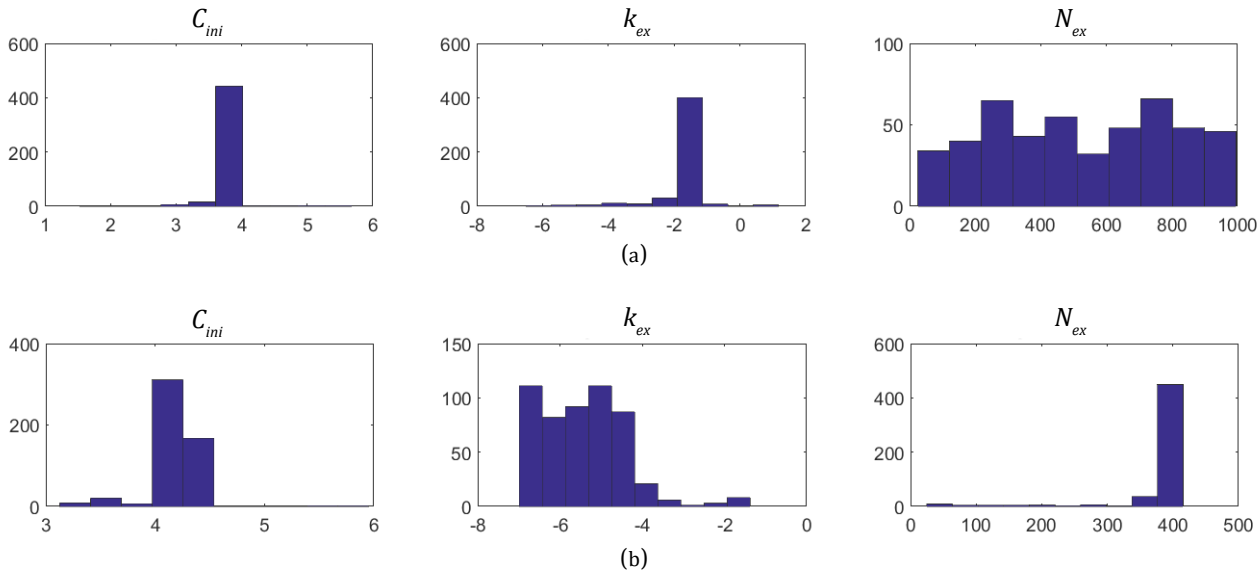


Fig. 1. Histogram of DREAM-derived parameters: a) Case 1 parameters. b) Case 2 parameters.

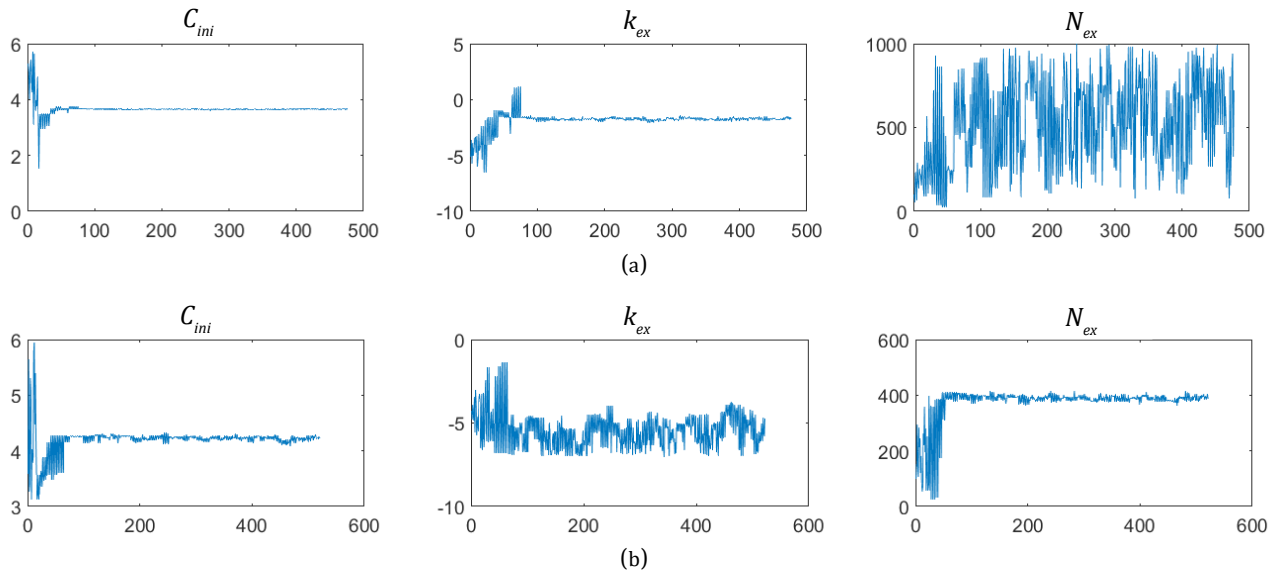


Fig. 2. Evolution of parameters throughout the implementation of the DREAM algorithm: a) Case 1 parameters. b) Case 2 parameters.

Turning now to the findings of this study, the most striking observation that emerges is the effect of parameters on the time-efficiency of the STM. Indeed, the model requires a long time to simulate its outputs irrespective of the methods used to solve equation (2) when using the mean values of the Case 2 (hereafter termed as MC2) parameters (check ST in Tables 7, 8 and 9). In contrast, the model runs fast enough (check ST in Tables 7, 8, and 9) regardless of which method solved equation (2) when using the mean values of the Case 1 (hereafter termed as MC1) parameters.

The findings also evince that the C_M and M_{IM} estimates obtained with the GL4 and GL6 methods are plainly influenced by decreasing the time step values and using the MC1 parameters. In fact, Figs. 3.a, 4.a, and 5.a illustrate the remarkable difference between the GL4 and GL6 predictions and the estimates provided by the ode45, ERK4 and IRK2 methods when using a step size of 1/8 and 1/4. This conclusion is made more precise by looking at the corresponding MAEs that relate to the mobile concentration of Cl^- (check MAEs in Tables 7, 8, and 9). Nevertheless, it seems that all the RK estimates are nearly the same using the MC1 parameters and a step size of 1. The findings also reveal that the GL4 and GL6 predictions are also similar to the ode45, ERK4 and IRK2 estimates regardless of the time step values by using the MC2 parameters (see Figs. 3.b, 4.b, and 5.b).

In addition, the findings attest that for both cases (Case 1 and Case 2) the model runs faster by using the ERK4, GL4 and GL6 methods in comparison to the ode45 solver (check ST in Tables 7, 8, and 9). However, the speed of predictions is much slower with the IRK2 method than with the reference ode45 solver (check ST in Tables 7, 8, and 9), except in the case where the MC1 parameters are used to run the model and the time step is set to 1.

In this respect, one more case that merits mentioning here is the fact that the time step size impacts on the simulation time (ST) of the STM. Truly, as the time step size increases the model simulates its outputs faster, no matter which RK method is used to solve equation (2) (check ST in Tables 7, 8, and 9). As a final remark, it is salient to mention that the C_M estimates provided by either method are not in agreement with the measurements of the Cl^- outflow as shown in Figs. 3.a and 3.b.

Table 7
ST of the STM obtained with the implementation of the reference ode45 solver and using the MC1 and MC2 parameter sets from Table 6.

	Case 1	Case 2
ST	23.43 seconds	33.8 seconds

Note: ST stands for the simulation time of the STM.

Table 8
ST of the STM obtained with the implementation of the ERK4, IRK2, GL4 and GL6 methods and MAEs between the RK (ERK4, IRK2, GL4 and GL6) and the ode45 estimates of C_M using the MC1 parameter set from Table 6.

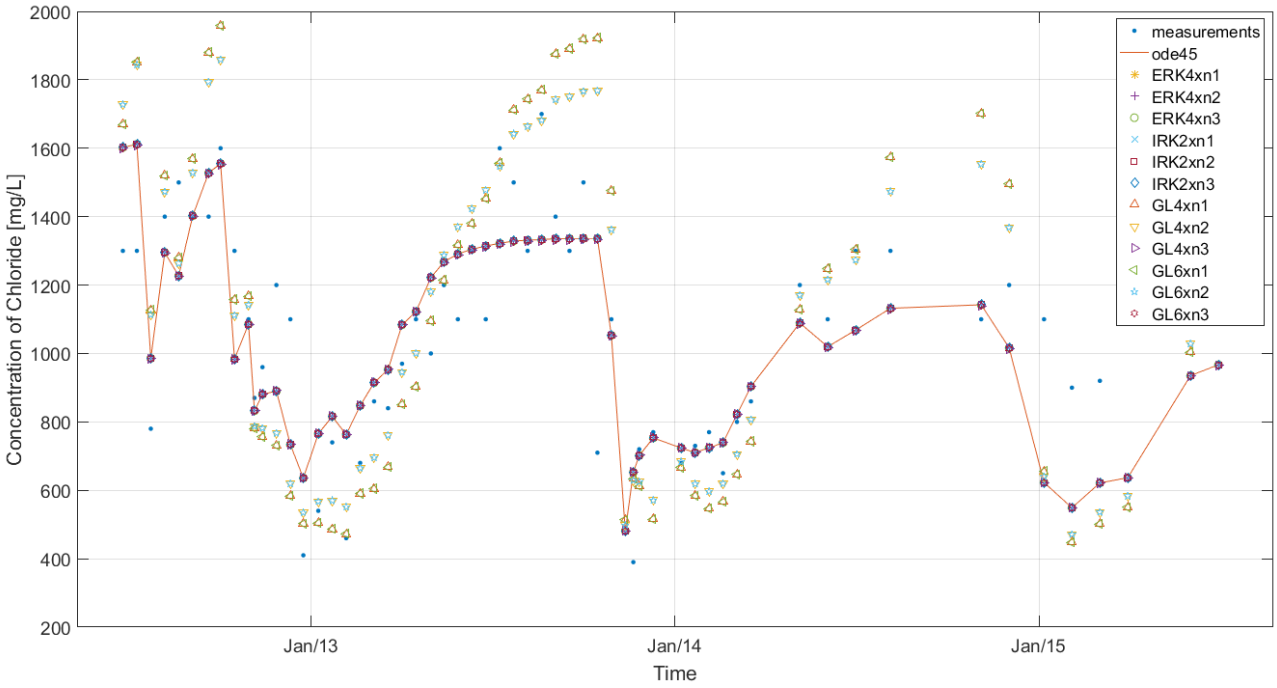
Method	dt = 1/8			dt = 1/4			dt = 1		
	ST	MAE	Acronym	ST	MAE	Acronym	ST	MAE	Acronym
ERK4	4.73	$3.06\cdot10^{-8}$	ERK4xn1	3.35	$3.13\cdot10^{-8}$	ERK4xn2	2.26	$2.30\cdot10^{-7}$	ERK4xn3
IRK2	64.56	$7.61\cdot10^{-5}$	IRK2xn1	34.33	$3.04\cdot10^{-4}$	IRK2xn2	12	$49\cdot10^{-3}$	IRK2xn3
GL4	4.66	221.41	GL4xn1	4.94	172.31	GL4xn2	4.84	1.47	GL4xn3
GL6	6.09	221.41	GL6xn1	6	172.31	GL6xn2	5.91	1.47	GL6xn3

Note: ST stands for the simulation time of the STM and is measured in seconds, MAE is measured in mg/L, and the Acronym names correspond to the legend names shown in Figs. 3.a-b, 4.a-b and 5.a-b.

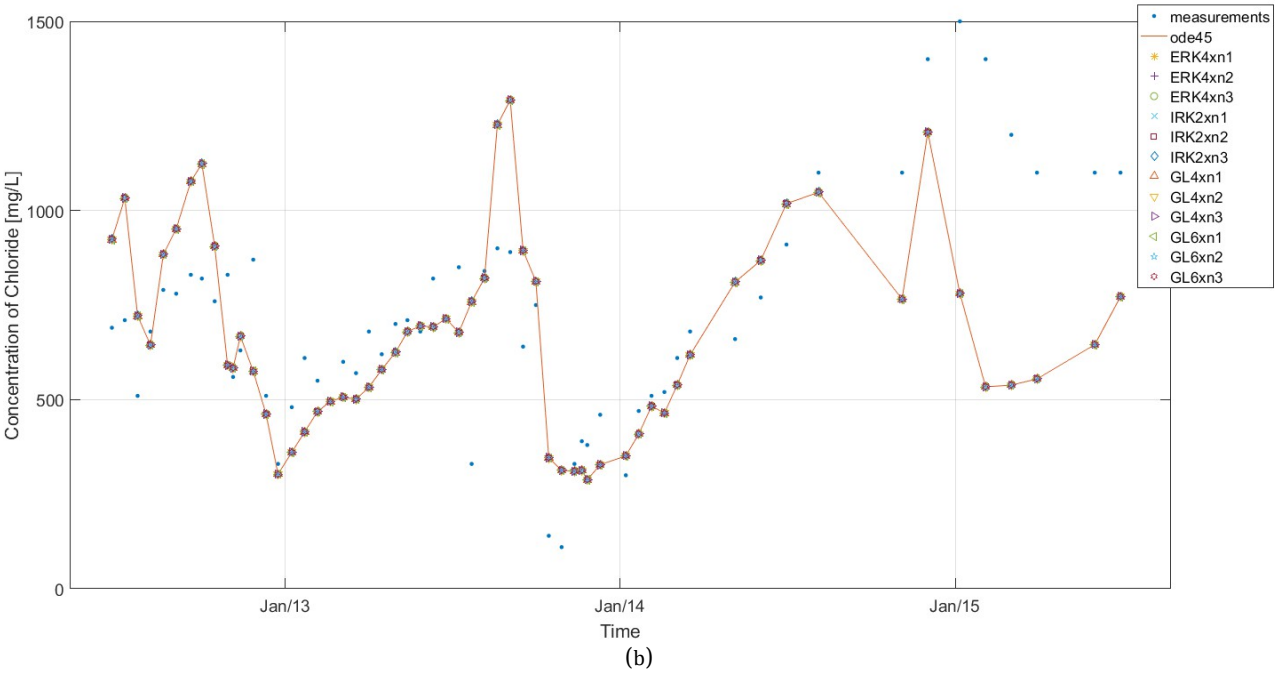
Table 9
ST of the STM obtained with the implementation of the ERK4, IRK2, GL4 and GL6 methods and MAEs between the RK (ERK4, IRK2, GL4 and GL6) and the ode45 estimates of C_M using the MC2 parameter set from Table 6.

Method	dt = 1/8			dt = 1/4			dt = 1		
	ST	MAE	Acronym	ST	MAE	Acronym	ST	MAE	Acronym
ERK4	10.63	$2.32\cdot10^{-10}$	ERK4xn1	9.27	$2.28\cdot10^{-10}$	ERK4xn2	7.72	$2.32\cdot10^{-10}$	ERK4xn3
IRK2	276.48	$4.52\cdot10^{-9}$	IRK2xn1	156.24	$3.92\cdot10^{-9}$	IRK2xn2	56.67	$8.93\cdot10^{-9}$	IRK2xn3
GL4	11.45	0.94	GL4xn1	11.37	0.81	GL4xn2	11.28	$4.22\cdot10^{-7}$	GL4xn3
GL6	12.91	0.94	GL6xn1	12.86	0.81	GL6xn2	12.69	$4.22\cdot10^{-7}$	GL6xn3

Note: ST stands for the simulation time of the STM and is measured in seconds, MAE is measured in mg/L, and the Acronym names correspond to the legend names shown in Figs. 3.a-b, 4.a-b and 5.a-b.

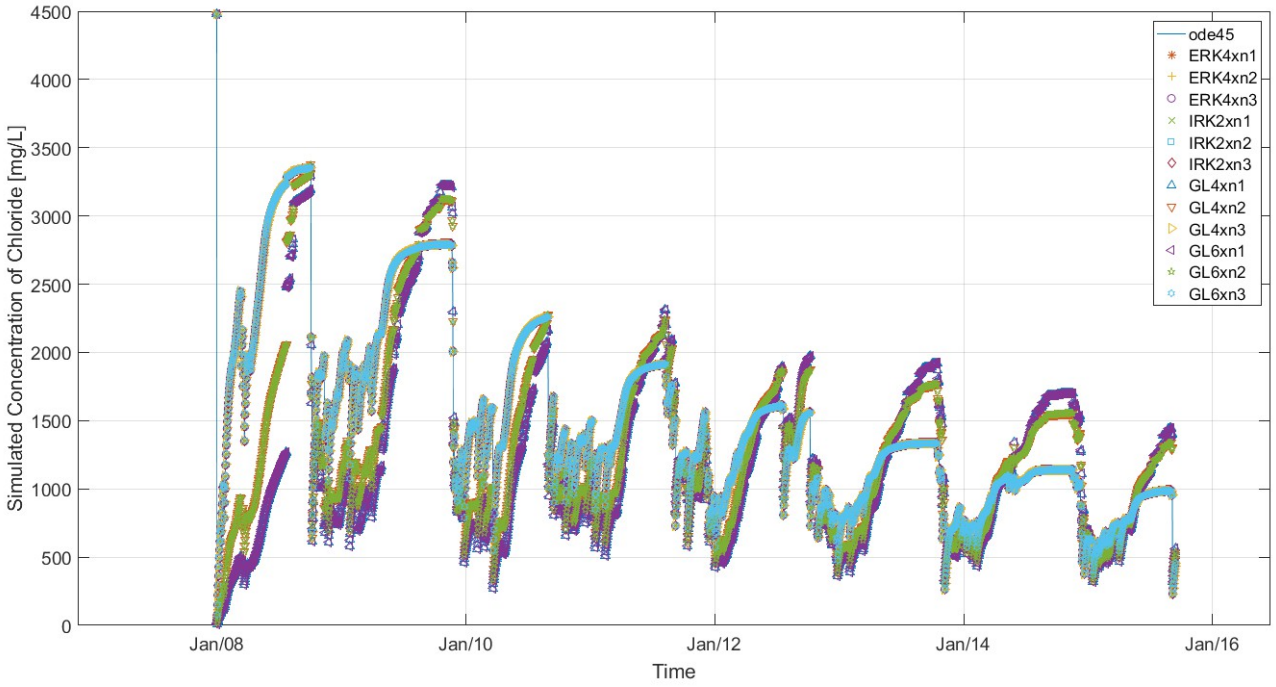


(a)

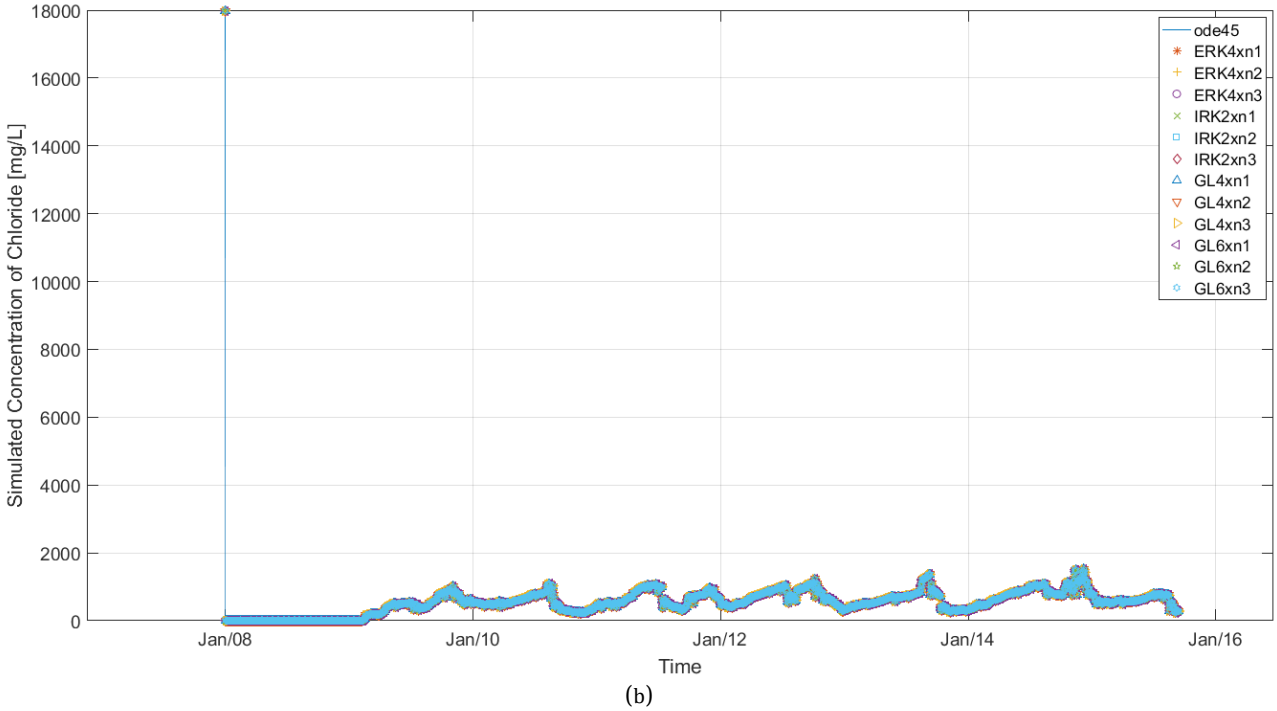


(b)

Fig. 3. Simulated and measured concentration of Cl^- for the period 1 August 2012 and 1 May 2015: a) Model estimates are obtained using the MC1 parameters from Table 6 and the measurements are taken from the north part of the Braambergen landfill. b) Model estimates are obtained using the MC2 parameters from Table 6 and the measurements are taken from the south part of the Braambergen landfill.

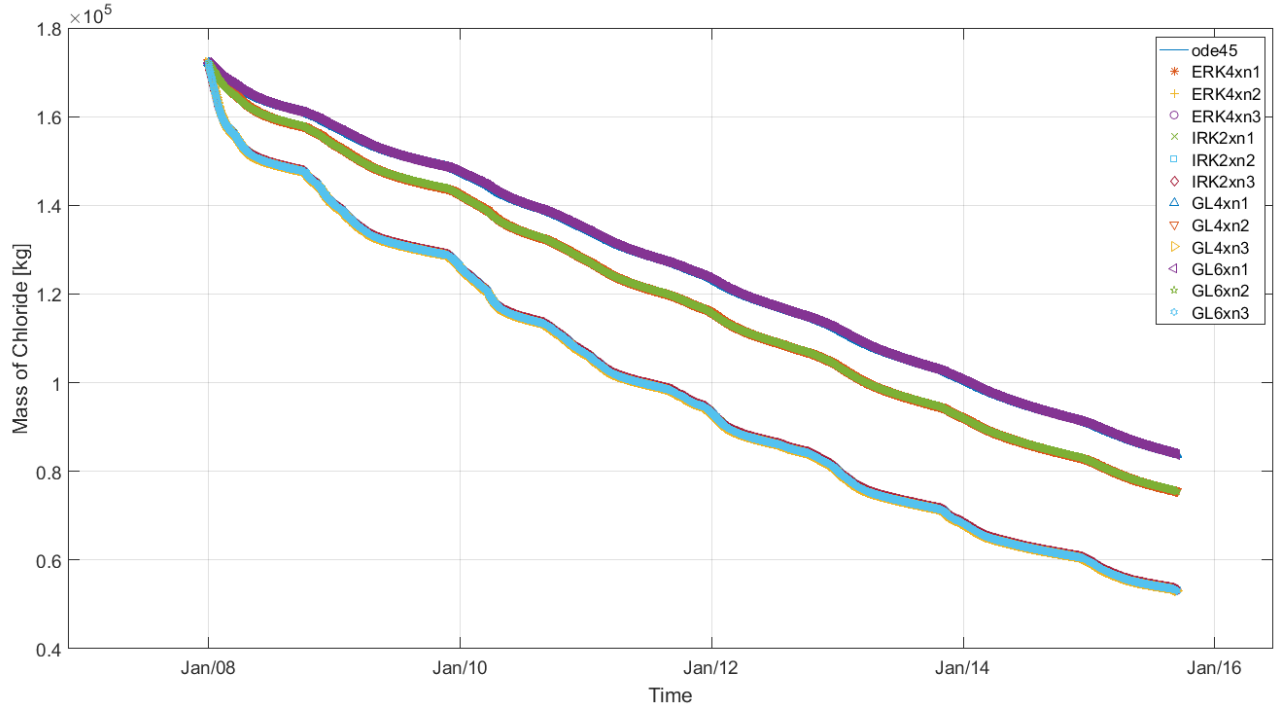


(a)

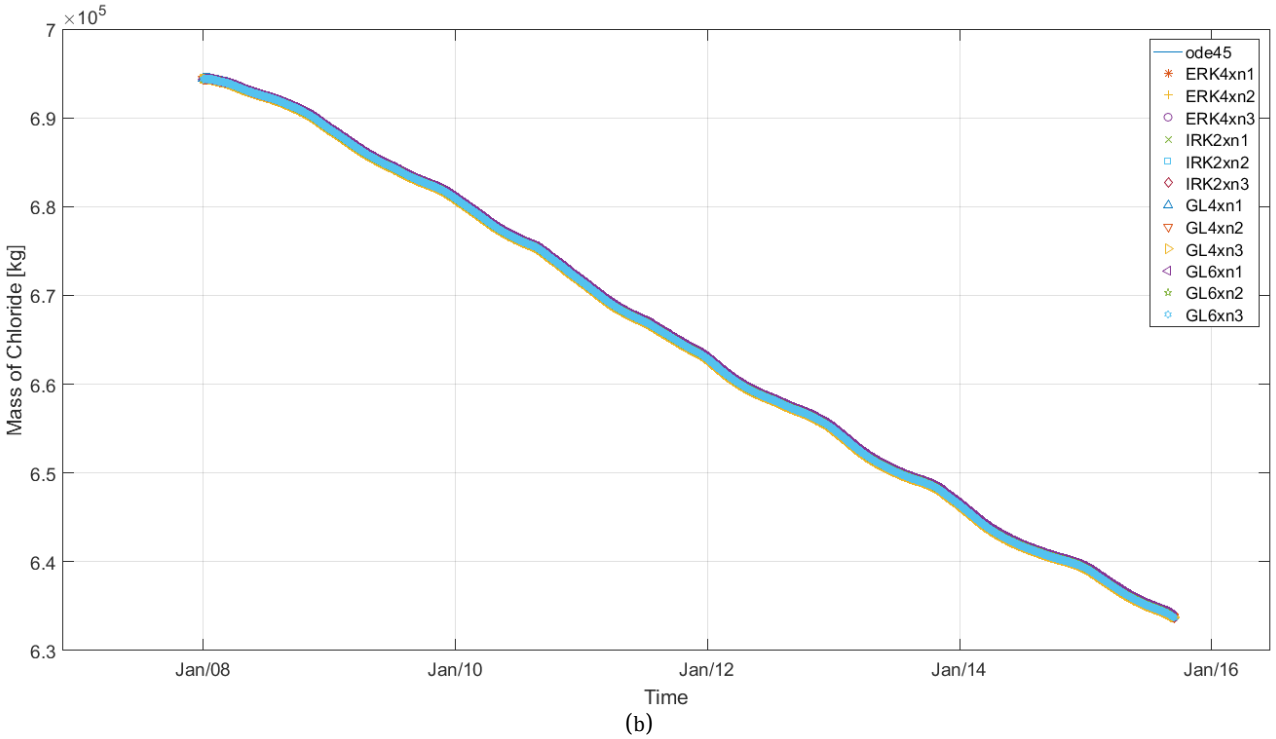


(b)

Fig. 4. Simulated concentration of Cl^- for the period 01 January 2008 and 15 September 2015: a) Model estimates are obtained using the MC1 parameters from Table 6. b) Model estimates are obtained using the MC2 parameters from Table 6.



(a)



(b)

Fig. 5. Simulated immobile mass of Cl^- for the period 31 May 2012 and 3 December 2015: a) Model estimates are obtained using the MC1 parameters from Table 6 and b) Model estimates are obtained using the MC2 parameters from Table 6.

4. Conclusions and future work

This study has implemented the ERK4, IRK2, GL4 and GL6 methods to solve the ODE given by equation (2) with the goal to reduce the simulation time of the STM. Recall that the model primarily solves equation (2) based on the ode45 solver in MATLAB. Also, it is worthwhile to remember that this computational model simulates the states under interest (C_M and M_{IM}) for the north and south part of the Braambergen landfill..

In the first place, this study has shown that it is practically feasible to apply high order IRK (such as GL4 and GL6) methods for solving non-linear ODEs such as equation (2). More importantly, the findings of this research indicate that the ERK4, GL4 and GL6 schemes have enormously contributed to enhancing the time-efficiency of this environmental model. In spite of that, the C_M estimates provided by either method used in this study are not in compliance agreement with the measurements of the Cl^- outflow.

In addition, it was shown that the performance of these RK methods was mostly affected by using different parameter sets (MC1 and MC2) and to a much lesser extent by the time step size. This might have been caused due to the inconsistently DREAM-derived parameters as discussed in Section 4. This points to the likelihood that the model parameters are associated with a certain degree of uncertainty which might impacts on the model outputs.

In consideration of the foregoing, further work needs to be done to determine whether the model parameters are correctly calibrated and examine by which means it is possible to obtain better state and parameter estimates. For this reason, future research should concentrate on the application of the notorious data assimilation methods that powerfully correct unreliable model states and parameters.

Acknowledgments

Special thanks go to Prof. Dr. Ir. T. J. Heimovaara from the Geo-Engineering section of TU Delft for his guidance and help in the achievement of this work.

References

[1] R. Alexander, (1977): Diagonally implicit Runge-Kutta methods for stiff ODE's, *SIAM Journal on Numerical Analysis*, **14**(6): 1006-1021.
[PDF URL: http://runge.math.smu.edu/Courses/Math6321_Fall10/projects/alexander-sinum1977.pdf]

[2] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri, (1997): Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations, *Applied Numerical Mathematics*, **25**(2-3): 151-167.
[Available online: http://dx.doi.org/doi:10.1016/ S0168-9274(97)00056-1]

[3] J. C. Butcher, (2007): Runge-Kutta methods, *Scholarpedia*, **2**(9): 3147.
[Available online: http://www.scholarpedia.org/ article/Runge-Kutta_methods]

[4] E. Babolian and M. Mordad, (2011): A numerical method for solving systems of linear and nonlinear integral equations of the second kind by hat basis functions, *Applied Mathematics and Computation*, **62**: 187-198.

[5] D. R. Durran and P. N. Blossey, (2012): Implicit-explicit multistep methods for fast-wave-slow-wave problems, *Monthly Weather Review*, **140**: 1307-1325.
[Available online: http://dx.doi.org/doi:10.1175/MWR-D-13-00132.1]

[6] S. Gill, (1951): A process for the step-by-step integration of differential equations in an automatic computing machine, *Proceedings of the Cambridge Philosophical Society*, **47**: 96-108.

[7] P. Haas, (2013): Implicit Runge Kutta Methods for Orbit Propagation, *CCAR University of Colorado at Boulder.*
[Available online: http://ccar.colorado.edu/asen5050/projects/projects_2013/Haas_Patrick/]

[8] E. Hairer and G. Wanner, (2010): Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems, *Springer-Verlag, Berlin*.

[9] E. Hairer, S. P. Nørsett, and G. Wanner, (1993): Solving Ordinary Differential Equations I: Nonstiff Problems, *Springer-Verlag, Berlin*.

[10] T. J. Heimovaara, A. Bun, and A. G. van Turnhout, (2015): Water balance modeling for estimation of residence time of water in a full scale landfill using a data-assimilation approach, HPM6: The 6th International

Workshop Hydro-Physico-Mechanics of Landfills, Delft, The Netherlands.

[11] B. Jones, (2013): Orbit Propagation Using Gauss-Legendre Collocation, *CCAR University of Colorado at Boulder*.
[PDF URL: http://ccar.colorado.edu/geryon/papers/Conference/jones_2012d.pdf]

[12] S. K. Kar, (2006): A semi-implicit Runge-Kutta time-difference scheme for the two-dimensional shallow-water equations, *Monthly Weather Review*, **134**: 2916-2926.
[Available online: http://dx.doi.org/10.1175/MWR3214.1]

[13] P. E. Kloeden and E. Platen, (1999): Numerical Solution of Stochastic Differential Equations, *Springer*.

[14] M. Kwizak and A. Robert, (1971): A semi-implicit scheme for grid point atmospheric models of the primitive equations, *Monthly Weather Review,* **99**: 32-36.
[Available online: http://dx.doi.org/10.1175/1520-0493(1971)099<0032:ASSFGP>2.3.CO;2]

[15] E. Laloy and J. A. Vrugt, (2012): High-dimensional posterior exploration of hydrologic models using multiple-try DREAM (ZS) and high-performance computing, *Water Resources Research*, **48**(1).

[16] R. W. MacCormack, (2011): Implicit methods for fluid dynamics, *Computers & Fluids*, **41**(1): 72-81.
[Available online: http://dx.doi.org/10.1016/j.compfluid.2010.09.017]

[17] F. S. J. Martinez, Y. A. Pachepsky, and W. J. Rawls, (2010): Single-porosity and dual-porosity modeling of water flow and solute transport in subsurface-drained fields using effective field-scale parameters, *Advances in Engineering Software*, **41**(1): 4-8.
[Available online: http://dx.doi.org/10.1016/j.advengsoft.2008.12.015]

[18] J. R. R. A. Martins, P. Sturdza, J. J. Alonso, (2003): The complex-step derivative approximation, *ACM Transactions on Mathematical Software*, **29**(3): 245-262.
[Available online: http://dx.doi.org/10.1145/838250.838251]

[19] N. W. Hawsa, P. Suresh C. Rao, Jirka Simunek, and Irene C. Poyer, (2005): Single-porosity and dual-porosity modeling of water flow and solute transport in subsurface-drained fields using effective field-scale parameters, *Journal of Hydrology*, **313**(3-4): 257-273.
[Available online: http://dx.doi.org/10.1016/j.jhydrol.2005.03.035]

[20] T. H. Pulliam, (2011): Development of implicit methods in CFD NASA Ames Research Center 1970s–1980s, *Computer & Fluids*, **41**(1): 65-71.
[Available online: http://dx.doi.org/10.1016/j.compfluid.2010.09.016]

[21] D. Russo, W. A. Jury, and G. L. Butters, (1989): Numerical analysis of solute transport during transient irrigation: 2. The effect of immobile water, *Water Resources Research*, **22**(10): 2119-2127.

[22] L. F. Shampine and M. W. Reichelt, (1997): The MATLAB ODE Suite, *SIAM Journal on Scientific Computing*, **18**: 1-22.

[23] J. H. Verner, (1996): High-order explicit Runge-Kutta pairs with low stage order. Special issue celebrating the centenary of Runge-Kutta methods, *Applied Numerical Mathematics*, **22**(1-3): 345-357.
[Available online: http://dx.doi.org/doi:10.1016/S0168-9274(96)00041-4]

[24] Y. Saito and T. Mitsui, (1996): Stability analysis of numerical schemes for stochastic differential equations, *SIAM Journal on Numerical Analysis,* **33**(6): 2254-2267.
[Available online: http://dx.doi.org/doi:10.1137/S0036142992228409]

[25] W. Hu, H. Ning, and Z. Xiaoxian, (2014): Impact of saturation on mass transfer rate between mobile and immobile waters in solute transport within aggregated soils, *Journal of Hydrology*, **519**(D): 3557-3565.
[Available online: http://dx.doi.org/10.1016/j.jhydrol.2014.10.057]

Appendix A. The derivation of formula (2)

This Appendix shows in detail the derivation of formula (2). To start with, the mass of solutes based on the DPM-MIM principle is given according to the following equation:

$$\frac{\partial F(M_M, M_{IM})}{\partial t} = -k_{ex} \cdot (C_M - C_{IM}) \tag{53}$$

It is also know that the relationship between mass and concentration of both mobile and immobile solutes is given as such:

$$C_M = \frac{M_M}{V_M} \tag{54}$$

$$C_{IM} = \frac{M_{IM}}{V_{IM}} \tag{55}$$

Then, the insertion of equations (54) and (55) into equation (53) yields:

$$\frac{\partial F(M_M, M_{IM})}{\partial t} = -k_{ex} \cdot (\frac{M_M}{V_M} - \frac{M_{IM}}{V_{IM}}) \tag{56}$$

By multiplying the right hand side of equation (56) by V_M leads to:

$$\frac{\partial F(M_M, M_{IM})}{\partial t} = -k_{ex} \cdot (V_M \cdot \frac{M_M}{V_M} - V_M \cdot \frac{M_{IM}}{V_{IM}}) = -k_{ex} \cdot (M_M - M_{IM} \cdot \frac{V_M}{V_{IM}}) \tag{57}$$

Thereafter, by rearranging Eq. (57) into matrix form, it becomes:

$$\frac{\partial F(M_M, M_{IM})}{\partial t} = -k_{ex} \cdot \begin{bmatrix} -1 & 0 & \dots & V_M / V_{IM} \\ 0 & -1 & \dots & \vdots \\ 0 & 0 & \dots & \vdots \\ 1 & 1 & \dots & \sum V_M / V_{IM} \end{bmatrix} \cdot \begin{bmatrix} M_M \\ \vdots \\ \vdots \\ \sum M_M / M_{IM} \end{bmatrix} \tag{58}$$

After that, it can be defined that:

$$\mathcal{H} = \begin{bmatrix} -1 & 0 & \dots & V_M / V_{IM} \\ 0 & -1 & \dots & \vdots \\ 0 & 0 & \dots & \vdots \\ 1 & 1 & \dots & \sum V_M / V_{IM} \end{bmatrix} \tag{59}$$

$$F(M_M, M_{IM}) = \begin{bmatrix} M_M \\ \vdots \\ \vdots \\ \sum M_M / M_{IM} \end{bmatrix} \tag{60}$$

Finally, the substitution of equations (59) and (60) into equation (58) leads to:

$$\frac{\partial F(M_M, M_{IM})}{\partial t} = -k_{ex} \cdot \mathcal{H} \cdot F(M_M, M_{IM}) \tag{61}$$

Beyond doubt, equation (61) is equivalent to equation (2).

Appendix B. Description of the ode45 MATLAB solver

The ode45 solver integrates a system of ODEs using four and five order Runge-Kutta formulas (Shampine and Reichelt, 1997). The command that is used in MATLAB to perform the ode45 solver is the following one:

$$[t\,,y]=ode45('odefun',t_{span},y_0) \tag{62}$$

In which, t_{span} = [t_0 t_{final}] integrates the ODE given by equation (1) from time t_0 to t_{final} with initial conditions y_0, *'odefun'* is a string containing the name of an ode file, and each row in solution array *y* corresponds to a time returned in column vector *t*.