Picard’s iterative method for nonlinear multicomponent transport equations

Juergen Geiser

Abstract: In this paper, we present a Picard’s iterative method for the solution of nonlinear multicomponent transport equations. The multicomponent transport equations are important for mixture models of the ionized and neutral particles in plasma simulations. Such mixtures deal with the so-called Stefan–Maxwell approaches for the multicomponent diffusion. The underlying nonlinearities are delicate and it is not necessary to be an analytical function of the dependent variables. The proposed solver method is based on Banach’s contraction fix-point principle that allows to solve such nonlinearities without making any use to Lagrange multipliers and constrained variations. Such an improvement allows to solve delicate nonlinear problems and we test the application to model with multicomponent transport equations.

Subject(s): Algorithms & Complexity; Computer Science; Computer Science (General); Engineering & Technology; Mathematics & Statistics for Engineers; Technology

Keywords: Picard’s iterative method; variational iterative method; multicomponent transport; multilevel iterative method; plasma models; multidiffusion model; nonlinear solver methods

AMS Subject classifications: 35K25; 35K20; 74S10; 70G65

1. Introduction

We are motivated to apply nonlinear multicomponent transport equations, which are given by delicate plasma processes.
We deal with problems of normal pressure, room temperature plasma applications, which are used for medical and technical processes. Here, the increasing importance of plasma chemistry based on the multicomponent plasma is a key factor for such a trend, see for low pressure plasma (Senega & Brinkmann, 2006) and for atmospheric pressure regimes (Tanaka, 2004). Due to the fact of the influence of the mass transfer in the multicomponent mixture, also the standard conservation laws have to be improved, such improvements are well-known in fusion research, see the modeling of high-ionized plasmas (Igitkhanov, 2011), but only few works are done for weak-ionized plasma for atmospheric pressure regimes.

We concentrate on a diffusion reaction model for a mixture of $H$, $H_2$, $H_3^+$ species. The model equation results in a delicate nonlinear multidiffusion equation. Such nonlinear equations are often solved with standard fix-point iteration schemes. Here, we propose a novel higher order Picard’s iterative method, which allows to accelerate the nonlinear solver based on intermediate level. Such a treatment allows to save computational time and we obtained higher order accurate results.

The paper is outlined as follows.

In Section 2, we present our mathematical model. The Picard’s iterative methods are discussed in Section 3. The numerical experiments are presented in Section 4. In the contents, that are given in Section 5, we summarize our results.

2. Mathematical model

In the following, a model is presented due to the motivation in Senega and Brinkmann (2006), which deals with a fluid dynamical description of a plasma model based on a small Knudsen number.

The Knudsen number is the ratio of the mean free path $\lambda$ over the typical domain size $L$ of the apparatus. For small Knudsen numbers $Kn \approx 0.01 – 1.0$, we deal with a Navier–Stokes equation, where for large Knudsen numbers $Kn \geq 1.0$, we deal with the Boltzmann equations.

We deal with the following plasma model of a mixture of $H$, $H_2$, $H_3^+$. We take into account the dissociation and ionization reactions, which are given as:

\begin{align}
H_2 + e & \rightarrow H^+ + 2 e, \\
H_2 + e & \rightarrow 2H + e,
\end{align}

where the electron temperature is given as $T_e = 17,400 \text{ K}$ and the gas temperature values remain constant $T_h = 600 \text{ K}$.

Further, we have $\lambda_1 = 1.58 \times 10^{-15} T_e^{0.5} \exp \left( \frac{-15.378}{T_e} \right) = 2.082 \times 10^{-13}$ and $\lambda_2 = 1.413 \times 10^{-15} T_e^2 \exp \left( \frac{-4.68}{T_e} \right) = 4.276 \times 10^{-7}$.

The diffusion coefficients are given as in the following formula:

$$D_{ij} = \frac{3}{16} \frac{f_{ij} k_i^2 T_i T_j}{M_i \Omega_{ij}^{1.1}(T_i)},$$

where the parameters are: $f_{ij}$ is a correction factor of order unity, $M_i = \frac{m_i}{m + m_j}$ is the reduced mass, $m_i$ mass of species $i$, $m_j$ mass of species $j$, $p$ pressure, $T_i, T_j$ the temperature of the corresponding species, and $\Omega_{ij}^{1.1}$ a collision integral (Hirschfelder, Curtiss, & Bird, 1966).

We assume the following binary diffusion parameters for our experiments:
We have the following underlying model: We deal with:

\[ \sum_{j=1}^{3} N_j = 0, \] (7)

\[ \frac{\xi_2 N_1 - \xi_1 N_2}{D_{12}} + \frac{\xi_3 N_1 - \xi_1 N_3}{D_{13}} = -\nabla \xi_1, \] (9)

\[ \frac{\xi_2 N_2 - \xi_1 N_1}{D_{12}} + \frac{\xi_3 N_2 - \xi_1 N_3}{D_{23}} = -\nabla \xi_2, \] (10)

and the kinetic term \( S_i \) is given as:

\[ S_i = \sum_{j=1}^{3} \lambda_{ij} \xi_j, \] (11)

where \( \lambda_{ij} \) are the reaction rates. Further, the domain is given as \( \Omega \in \mathbb{R}^d, d \in \mathbb{N}^+ \) with \( \xi_i \in C^2 \).

We decompose the diffusion and the reaction parts and apply the following: a splitting approach to our problem; we compute \( n = 1, \ldots, N, t_0, t_1, \ldots, t_n \) time steps:

The first step is given as (Diffusion step):

\[ \partial_t \tilde{\xi}_i + \nabla \cdot N_i = 0, \quad 1 \leq i \leq 3, \] (12)

\[ \sum_{j=1}^{3} N_j = 0, \] (13)

\[ \frac{\tilde{\xi}_2 N_1 - \tilde{\xi}_1 N_2}{D_{12}} + \frac{\tilde{\xi}_3 N_1 - \tilde{\xi}_1 N_3}{D_{13}} = -\nabla \tilde{\xi}_1, \] (14)

\[ \frac{\tilde{\xi}_2 N_2 - \tilde{\xi}_1 N_1}{D_{12}} + \frac{\tilde{\xi}_3 N_2 - \tilde{\xi}_1 N_3}{D_{23}} = -\nabla \tilde{\xi}_2, \quad \text{for } t \in [t^n, t^{n+1}], \] (15)

\[ \tilde{\xi}_i(t^n) = \xi_i(t^n), \quad i = 1, 2, 3, \] (16)

and the next step (Reaction step):

\[ \partial_t \tilde{\xi}_i = S_i, \quad 1 \leq i \leq 3, \quad \text{for } t \in [t^n, t^{n+1}], \] (17)

\[ \xi_i(t^n) = \tilde{\xi}_i(t^{n+1}), \quad i = 1, 2, 3. \] (18)

**Remark 1** Based on the derived model, we discuss the application of the novel nonlinear solvers. We can also generalize the schemes for more than three species.

### 3. Iterative method

In the following, we discuss Picard’s iterative method with different levels. Picard’s iterative methods are known to solve delicate nonlinear problems (see Ramos, 2008, 2009).

In the following, we discuss the basic ideas and we develop improved Picard’s iterative methods, e.g. exponential schemes, for our special treatments.
The basic Picard's iterative method is given as:

\[ u_{k+1}(x, t) = (Pu_k)(x, t), \quad k = 0, 1, 2, \ldots, \]  

where the Picard's operator is given as:

\[ (Pu)(x, t) = u(x, t_0) + \int_{t_0}^{t} F(x, s, u(x, s), \nabla u(x, s), \Delta u(x, s)) \, ds. \]  

Picard's operator is applied for the two-level method as:

\[ u_{k+1}(x, t) = u_0(x, t) + \int_{t_0}^{t} F(x, s, u_k(x, s), \nabla u_k(x, s), \Delta u_k(x, s)) \, ds, \quad k = 0, 1, 2, \ldots. \]  

Further, the operator is applied for the three-level method as:

\[ u_{k+1}(x, t) = u_k(x, t) + \int_{t_k}^{t} F_k(s) - F_{k-1}(s) \, ds, \quad k = 0, 1, 2, \ldots, \]  

where \( F_k(s) = F(x, s, u_k(x, s), \nabla u_k(x, s), \Delta u_k(x, s)) \).

**Proof** We combine two two-level methods, given as:

\[ u_{k+1}(x, t) = u_0(x, t_0) + \int_{t_0}^{t} F_k(s) \, ds, \quad k = 0, 1, 2, \ldots, \]  

and

\[ u_k(x, t) = u_0(x, t_0) + \int_{t_k}^{t} F_{k-1}(s) \, ds, \quad k = 0, 1, 2, \ldots, \]  

where if we subtract Equation (23) with (24), we obtain:

\[ u_{k+1}(x, t) = u_k(x, t) + \int_{t_k}^{t} F_k(s) - F_{k-1}(s) \, ds, \quad k = 0, 1, 2, \ldots, \]  

3.1. Multilevel iterative method based on Picard's iterative method

The multilevel iterative method is given as:

**Theorem 1** We have the following construction formula for the \( k \)-level iterative method based on Picard's iterative method:

- Two-level method is given as:

  \[ u_i(x, t) - u(x, t_0) = \int_{t_0}^{t} F_{i-1}(s) \, ds. \]  

- \( i \)-level method with \( i \geq 3 \) is given as:

  \[ \sum_{j=0}^{k} (-1)^j \binom{k}{j} u_{i-j}(x, t) = \int_{t_0}^{t} \sum_{j=0}^{k} (-1)^j \binom{k}{j} F_{i-j-1}(s) \, ds. \]
where \( k = i - 2 \).

**Proof**  The case \( i = 2 \) is the standard Picard method and clear.

For the case \( i \geq 3 \) we have the following complete induction given as:

We start from \( i = 3 \) (and \( k = i - 2 \)) and have:

\[
\begin{align*}
\sum_{j=0}^{i} (-1)^j \binom{k}{j} u_{j+1}(x, t) &= \int_{t_0}^{t} \sum_{j=0}^{i} (-1)^j \binom{k}{j} f_{i-j-1}(s) \, ds, \quad (28) \\
u_{i}(x, t) - u_{i-1}(x, t) &= \int_{t_0}^{t} (f_{i-1}(s) - f_{i-2}(s)) \, ds. \quad (29)
\end{align*}
\]

The induction step is given as \( i \to i + 1 \), where we have to multiply \( (u_i - u_{i-1}) \) to the \( i \)-level method as:

\[
(u_i - u_{i-1}) \sum_{j=0}^{i} (-1)^j \binom{k}{j} u_{j+1}(x, t) \quad (30)
\]

\[
= \sum_{j=0}^{k} (-1)^j \binom{k}{j} u_{j+1}(x, t) + \sum_{j=0}^{k} (-1)^j \binom{k}{j} u_{j+1}(x, t) \quad (31)
\]

\[
= \int_{t_0}^{t} (f_{i-1}(s) - f_{i-2}(s)) \, ds, \quad k = 0, 1, 2, \ldots. \quad (32)
\]

\[
= \int_{t_0}^{t} (f_{i-1}(s) - f_{i-2}(s)) \, ds. \quad k = 0, 1, 2, \ldots. \quad (33)
\]

where \( \bar{k} = i - 2 \).

The same is done for the right hand side and we have proven the formula. \( \square \)

**Remark 2**  The next level methods are given as:

- **Four-level method:**

\[
u_{k}(x, t) = 2u_{k-1}(x, t) - u_{k-2}(x, t)
\]

\[
+ \int_{t_0}^{t} (f_{k-1}(s) - 2f_{k-2}(s) + f_{k-3}(s)) \, ds, \quad k = 0, 1, 2, \ldots. \quad (34)
\]

- **Five-level method:**

\[
u_{k}(x, t) = 3u_{k-1}(x, t) - 3u_{k-2}(x, t) + u_{k-3}(x, t)
\]

\[
+ \int_{t_0}^{t} (f_{k-1}(s) - 3f_{k-2}(s) + 3f_{k-3}(s) - 3f_{k-4}(s)) \, ds, \quad k = 0, 1, 2, \ldots. \quad (35)
\]

### 3.2. Exponential Picard’s iterative method

In the following, we discuss an extension of Picard’s iterative method with a linear operator (exponential idea).

The exponential Picard’s iterative method is given as:

\[
\frac{\partial u_{k+1}(x, t)}{\partial t} = Au_{k+1} + F_k(t), \quad k = 0, 1, 2, \ldots, \quad (36)
\]

where the variation of constant formula is applied and we obtain:
\[ u_{k+1}(x, t) = \exp(At)u(x, t_0) + \int_{t_0}^{t} \exp(A(t - s))F_i(s) \, ds, \quad k = 0, 1, 2, \ldots \] (37)

So the two-level method is given as:

\[ u_{k+1}(x, t) = \exp(At)u(x, t_0) + \int_{t_0}^{t} \exp(A(t - s))F(x, s, u_k(x, s), \nabla u_k(x, s), \Delta u_k(x, s)) \, ds, \quad k = 0, 1, 2, \ldots \] (38)

Further, the three-level method is given as:

\[ u_{k+1}(x, t) = u_k(x, t) + \int_{t_0}^{t} \exp(A(t - s))(F_k(s) - F_{k-1}(s)) \, ds, \quad k = 0, 1, 2, \ldots \] (39)

where \( F_k(s) = F(x, s, u_k(x, s), \nabla u_k(x, s), \Delta u_k(x, s)) \).

The multilevel iterative method is given as:

**Theorem 2** We have the following construction formula for the k-level iterative method based on Picard’s iterative method:

- Two-level method is given as:

\[ u_i(x, t) - \exp(At)u(x, t_0) = \int_{t_0}^{t} \exp(A(t - s))F_i(s) \, ds. \] (40)

- k-level method with \( k \geq 3 \) is given as:

\[ \sum_{j=0}^{k} (-1)^j \binom{k}{j} u_{j-1}(x, t) = \int_{t_0}^{t} \exp(A(t - s)) \left( \sum_{j=0}^{k} (-1)^j \binom{k}{j} F_{j-1}(s) \right) \, ds, \] (41)

where \( \tilde{k} = k - 2 \).

**Proof** The case \( k = 2 \) is the variation of constants with the standard Picard method and is clear.

For the case \( k \geq 3 \), we have the following complete induction given as in the proof of Theorem 1. □

**Remark 3** The next level methods are given as:

- Four-level method:

\[ u_k(x, t) = 2u_{k-1}(x, t) - u_{k-2}(x, t) + \int_{t_0}^{t} \exp(A(t - s))(F_{k-1}(s) - 2F_{k-2}(s) + F_{k-3}(s)) \, ds, \quad k = 0, 1, 2, \ldots \] (42)

- Five-level method:

\[ u_k(x, t) = 3u_{k-1}(x, t) - 3u_{k-2}(x, t) + u_{k-3}(x, t) + \int_{t_0}^{t} \exp(A(t - s))(F_{k-1}(s) - 3F_{k-2}(s) + 3F_{k-3}(s) - 3F_{k-4}(s)) \, ds \quad k = 0, 1, 2, \ldots \] (43)
4. Numerical experiments

In the following experiments, we discuss the improvements of our novel Picard’s iterative methods to the standard approaches. We start with test examples of blow-up and Bernoulli’s equations, where we could compare to analytical solutions. A multicomponent model is discussed in the next steps and the benefits to the higher order multilevel methods are presented.

4.1. First numerical example: blow-up equation

As a nonlinear differential example, we chose the Bernoulli’s equation:

\[
\frac{\partial u(t)}{\partial t} = \lambda (u(t)^p, \ t \in [0, T], \text{ with } u(0) = 1, \tag{44}
\]

where the analytical solution can be derived as (see also Geiser, 2008):

\[
u(t) = \left(\frac{p-1}{c-1}\right)^{\frac{1}{p}}.
\]

When we apply \(p = 2\), we have

\[
u(t) = \frac{1}{1 - t/\Delta t}
\]

We choose \(p = 2, \lambda = -0.2, T = 0.2\) and \(u(0) = 1\).

For the numerical experiments, we have the following analytical solution:

\[
u_1(t) = 1 + \lambda t,
\]

\[
u_2(t) = 1 + \lambda t + \frac{1}{2} \lambda^2 t^2 + \frac{1}{3} \lambda^3 t^3,
\]

\[
u_3(t) = 1 + \lambda t + \frac{1}{2} \lambda^2 t^2 + \frac{1}{3} \lambda^3 t^3 + \frac{5}{12} \lambda^4 t^4 + \frac{1}{3} \lambda^5 t^5 + \frac{1}{3} \lambda^6 t^6 + \frac{1}{3} \lambda^7 t^7.
\]

We apply the Picard’s iterative method, which is given in the following as a two-level method. Here, we approximate in each sub-interval \([t^n, t^{n+1}]\), \(n = 0, 1, \ldots, N\), the integral:

\[
u_{k+1,2}(t^{n+1}) = \nu_0(t^n) + \int_{t^n}^{t^{n+1}} \lambda \nu_{k,2}(s) \, ds, \quad k = 0, 1, 2, \ldots
\]

For the numerical integration, we apply the different Simpson’s rules of higher order.

- Simpson’s Rule: We apply the Simpson’s rule for the two-level method:

\[
u_{k+1,2}(t^{n+1}) = \nu_0(t^n) + \lambda \frac{\Delta t}{6} \left( u_{k,2}(t^{n+1}) + 4u_{k,2}(t^{n+1/2}) + u_{k,2}(t^n) \right), \quad k = 0, 1, 2, \ldots
\]

where \(u_{k,2}(t^{n+1}) = u_{k,2}(t^{n+1}) - \frac{1}{2} \Delta t \frac{\partial u_{k,2}}{\partial t}(t^{n+1}) = u_{k,2}(t^{n+1}) - \frac{1}{2} \Delta t u_{k,2}(t^{n+1})\).

- 3/8 Simpson’s rule: We apply the 3/8-Simpson’s rule for the two-level method:

\[
u_{k+1,2}(t^{n+1}) = \nu_0(t^n) + \lambda \frac{\Delta t}{8} \left( u_{k,2}(t^{n+1}) + 3u_{k,2}(t^{n+1/3}) + 3u_{k,2}(t^{n+2/3}) + u_{k,2}(t^n) \right), \quad k = 0, 1, 2, \ldots
\]
where

\[ u_{k,2}(t_{n+1/3}) = u_{k,2}(t_{n+1}) - \frac{2}{3} \Delta t \frac{du_{k,2}}{dt}(t_{n+1}) = u_{k,2}(t_{n+1}) - \frac{2}{3} \Delta t u_{k,2}^2(t_{n+1}), \]

\[ u_{k,2}(t_{n+2/3}) = u_{k,2}(t_{n+1}) - \frac{1}{3} \Delta t \frac{du_{k,2}}{dt}(t_{n+1}) = u_{k,2}(t_{n+1}) - \frac{1}{3} \Delta t u_{k,2}^2(t_{n+1}). \]

Further, for the next time step, we have \( u(t_{n+1}) = u_{k,2}(t_{n+1}) \) and \( K \) is the number of iterative steps, e.g. \( K = 4 \).

Further, the three-level method is given as a combination of two two-level methods’ means, and we have:

\[ \Delta u_{k+1,3}(t_{n+1}) = u_{k+1,3}(t_{n+1}) - u_{k,3}(t_{n+1}) \]

\[ = \int_{t_{n}}^{t_{n+1}} \lambda \left( u_{k+1,3}^2(s) - u_{k,3}^2(s) \right) ds, \]  \hspace{1cm} (51)

\[ u_{k+1,3}(t_{n+1}) = u_{k,3}(t_{n+1}) + \Delta u_{k+1,3}(t_{n+1}) \]  \hspace{1cm} (52)

where \( u_0(t) = u(t^0) \) and for \( u_{1,3}(t) = u_{1,2}(t) \), we have the two-level method. Further, we apply the trapezoidal or Simpson’s rule for the numerical integration of the integrals. Further, for the next time step, we have \( u(t_{n+1}) = u_{k,3}(t_{n+1}) \) and \( K \) is the number of iterative steps, e.g. \( K = 4 \).

Remark 4  The three-level method has the benefit of applying the numerical integration for the differences between \( k \) and \( k - 1 \) solutions of the right-hand side, such that we can accelerate the solver process, while we skip an additional numerical integration in each step.

Based on the reference solution, we deal with the following errors:

\[ E_{1,k,\Delta t,[0,T]} = |u_{\text{method},k,\Delta t}(t) - u_{\text{ref},\Delta t}(t)| \]

\[ = \sum_{n=0}^{N} \Delta t |u_{\text{method},k,\Delta t}(t^n) - u_{\text{ref},\Delta t}(t^n)|, \]  \hspace{1cm} (53)

where \( \text{method} \) is the different Picard’s methods and \( k \) is the number of iterative steps.

We apply the Picard’s iteration with the two-level method (see Figure 1 and Table 1).

Remark 5  Here, we see the improvement till \( K = 5 \), while here the accuracy is reached. The reduction of the errors is done with each iterative step.

We apply the Picard’s iteration with the three-level method (see Figure 2 and Table 2).

Remark 6  Here, we see the improvement till \( K = 4 \), while here the accuracy is reached. The reduction of the errors is done with each iterative step.

We apply the blow-up experiment with \( \lambda = 0.999 \) and obtain the following solutions. The Picard’s iteration with the two-Level method is given in Figure 3 for \( \lambda = 0.999 \).

The Picard’s iteration with the three-level method is given in Figure 4 for \( \lambda = 0.999 \).

Remark 7  We discussed the multilevel Picard’s method with different iterative steps to a blow-up problem. We saw that we obtain high accurate solutions with \( k \approx 3 – 5 \) iterative steps and that a three-level method can obtain faster numerical results.
Table 1. Comparative results of the two-level method with \( k \) iterative steps and the analytical solution for \( \Delta t = 10^{-2} \) and \( T = 1.0 \)

| Number of iterations | \( \Delta t \)  | \( \text{Err}_{1,1} \)  | Comp. time |
|----------------------|----------------|-------------------------|------------|
| \( K = 1 \)          | \( 10^{-1} \)  | \( 3.33 \times 10^{-2} \) | \( 2.3611 \times 10^{-4} \) |
| \( K = 2 \)          | \( 10^{-1} \)  | \( 9.0912 \times 10^{-4} \) | \( 3.1310 \times 10^{-4} \) |
| \( K = 3 \)          | \( 10^{-1} \)  | \( 1.1806 \times 10^{-5} \) | \( 3.7982 \times 10^{-6} \) |
| \( K = 4 \)          | \( 10^{-1} \)  | \( 1.5768 \times 10^{-3} \) | \( 4.3628 \times 10^{-4} \) |
| \( K = 5 \)          | \( 10^{-1} \)  | \( 1.4943 \times 10^{-2} \) | \( 4.8248 \times 10^{-4} \) |
| \( K = 6 \)          | \( 10^{-1} \)  | \( 1.4968 \times 10^{-3} \) | \( 5.2867 \times 10^{-4} \) |
| \( K = 2 \)          | \( 10^{-2} \)  | \( 3.33 \times 10^{-2} \)  | \( 2.3 \times 10^{-3} \)   |
| \( K = 3 \)          | \( 10^{-2} \)  | \( 8.7588 \times 10^{-3} \) | \( 3.1 \times 10^{-3} \)   |
| \( K = 4 \)          | \( 10^{-2} \)  | \( 1.0439 \times 10^{-2} \) | \( 3.8 \times 10^{-3} \)   |
| \( K = 5 \)          | \( 10^{-2} \)  | \( 1.5461 \times 10^{-1} \) | \( 4.5 \times 10^{-3} \)   |
| \( K = 6 \)          | \( 10^{-2} \)  | \( 1.5385 \times 10^{−1} \) | \( 4.8 \times 10^{-3} \)   |
| \( K = 2 \)          | \( 10^{-3} \)  | \( 3.33 \times 10^{-3} \)  | \( 5 \times 10^{-4} \)     |
| \( K = 3 \)          | \( 10^{-3} \)  | \( 8.7256 \times 10^{-4} \) | \( 3.1310 \times 10^{-4} \) |
| \( K = 4 \)          | \( 10^{-3} \)  | \( 1.0303 \times 10^{-4} \) | \( 3.7982 \times 10^{-6} \) |
| \( K = 5 \)          | \( 10^{-3} \)  | \( 1.5435 \times 10^{-4} \) | \( 4.3628 \times 10^{-6} \) |
| \( K = 6 \)          | \( 10^{-3} \)  | \( 1.5427 \times 10^{-4} \) | \( 4.8248 \times 10^{-6} \) |

**4.2. Second numerical example: Bernoulli’s equation**

As a nonlinear differential example, we chose the Bernoulli’s equation:

\[
\frac{\partial u(t)}{\partial t} = (\lambda_1 + \lambda_2)u(t) + (\lambda_2 + \lambda_3)(u(t))^p, \quad t \in [0, T], \quad \text{with } u(0) = 1,
\]

(54)

where the analytical solution can be derived as (see also Geiser, 2008):

\[
u(t) = \exp((\lambda_1 + \lambda_2)t) \left[ -\frac{\lambda_2 + \lambda_3}{\lambda_1 + \lambda_3} \exp((\lambda_1 + \lambda_2)(p - 1)t) + c \right]^{1/(1-p)}.
\]

Using \( u(0) = 1 \), we find that \( c = 1 + \frac{\lambda_1 + \lambda_2}{\lambda_2 + \lambda_3} \), so

\[
u(t) = \exp((\lambda_1 + \lambda_2)t) \left[ 1 + \frac{\lambda_2 + \lambda_3}{\lambda_1 + \lambda_3} \left[ 1 - \exp((\lambda_1 + \lambda_3)(p - 1)t) \right] \right]^{1/(1-p)}.
\]
Figure 2. The three-level Picard’s method compared with the analytical solution.

Figure 3. The two-level Picard’s method compared with the analytical solution.

Figure 4. The three-level Picard’s method compared with the analytical solution.

Table 2. Comparative results of the three-level method with \( k \) iterative steps and the analytical solution for \( \Delta t = 10^{-2} \) and \( T = 1.0 \)

| Three-Level method | Accuracy \( err_L \) | CPU-time (sec) |
|--------------------|----------------------|----------------|
| \( k = 1 \)       | \( 10^{-4} \)         | \( 8 \times 10^{-2} \) |
| \( k = 2 \)       | \( 10^{-1} \)         | \( 3 \times 10^{-2} \) |
| \( k = 3 \)       | \( 10^{-1} \)         | \( 3 \times 10^{-2} \) |
| \( k = 4 \)       | \( 10^{-1} \)         | \( 3 \times 10^{-2} \) |
| \( k = 5 \)       | \( 10^{-1} \)         | \( 3 \times 10^{-2} \) |
We choose \( p = 2, \lambda_1 = -1, \lambda_2 = -0.5, \lambda_3 = -1, \lambda_4 = -1, T = 0.2 \), and \( u(0) = 1 \).

We apply the Picard’s iterative method based on approximating in each sub-interval \([t^n, t^{n+1}]\), \( n = 0, 1, \ldots, N \), which is given as:

- **Two-level method** is given as:

\[
\begin{align*}
    u_{k+1}(t^{n+1}) &= \exp((\lambda_1 + \lambda_3)\Delta t)u(t^n) \\
    &+ \int_{t^n}^{t^{n+1}} \exp((\lambda_1 + \lambda_3)(t^{n+1} - s))(\lambda_2 + \lambda_4)u_k^p(s) \, ds, \quad k = 1, 2, \ldots,
\end{align*}
\]

where for \( k = 0 \); we have:

\[
\begin{align*}
    u_1(t^{n+1}) &= \exp((\lambda_1 + \lambda_3)\Delta t)u(t^n) \\
    &+ \int_{t^n}^{t^{n+1}} \exp((\lambda_1 + \lambda_3)(t^{n+1} - s))(\lambda_2 + \lambda_4)u_0^p(s) \, ds,
\end{align*}
\]

with the time step \( \Delta t = t^{n+1} - t^n \). We apply the following numerical integration rules:

- **Simpson’s rule**: We apply the Simpson’s rule for the two-level method:

\[
\begin{align*}
    u_{k+1,2}(t^{n+1}) &= \exp((\lambda_1 + \lambda_3)\Delta t)u_0(t^n) \\
    &+ (\lambda_2 + \lambda_4)\frac{\Delta t}{6}(u_{k,2}^p(t^{n+1}) + 4 \exp((\lambda_1 + \lambda_3)\Delta t/2)u_{k,2}^p(t^{n+1}/2) \\
    &+ \exp((\lambda_1 + \lambda_3)\Delta t)u_{k,2}^p(t^n)), \quad k = 0, 1, 2, \ldots,
\end{align*}
\]

where we have the intermediate solution as:

\[
\begin{align*}
    u_{k,2}(t^{n+1}/2) &= u_{k,2}(t^{n+1}) - \frac{1}{2} \Delta t \frac{du_{k,2}}{dt}(t^{n+1}) \\
    &= u_{k,2}(t^{n+1}) - \frac{1}{2} \Delta t(\lambda_1 + \lambda_3)(u_{k,2}(t^{n+1}) + (\lambda_2 + \lambda_4)(u_{k,2}(t^{n+1}))^p).
\end{align*}
\]

- **3/8 Simpson’s rule**: We apply the 3/8-Simpson’s rule for the two-level method:

\[
\begin{align*}
    u_{k+1,3}(t^{n+1}) &= \exp((\lambda_1 + \lambda_3)\Delta t)u_0(t^n) \\
    &+ (\lambda_2 + \lambda_4)\frac{\Delta t}{8}(u_{k,2}^p(t^{n+1}) + 3 \exp((\lambda_1 + \lambda_3)\Delta t/3)u_{k,2}^p(t^{n+1}/3) \\
    &+ 3 \exp((\lambda_1 + \lambda_3)2\Delta t/3)u_{k,2}^p(t^{n+1}/3) + \exp((\lambda_1 + \lambda_3)\Delta t)u_{k,2}^p(t^n)), \quad k = 0, 1, 2, \ldots,
\end{align*}
\]

where

\[
\begin{align*}
    u_{k,2}(t^{n+1}/3) &= u_{k,2}(t^{n+1}) - \frac{2}{3} \Delta t \frac{du_{k,2}}{dt}(t^{n+1}) = u_{k,2}(t^{n+1}) - \lambda_2 \frac{2}{3} \Delta t u_{k,2}^2(t^{n+1}), \\
    u_{k,2}(t^{n+1}/2) &= u_{k,2}(t^{n+1}) - \frac{1}{3} \Delta t \frac{du_{k,2}}{dt}(t^{n+1}) = u_{k,2}(t^{n+1}) - \lambda_2 \frac{1}{3} \Delta t u_{k,2}^2(t^{n+1}).
\end{align*}
\]

- **Three-level method** is given as: Further, the three-level method is given as a combination of two two-level methods’ means; we have:

\[
\begin{align*}
    \Delta u_{k+1,3}(t^{n+1}) &= u_{k+1,3}(t^{n+1}) - u_{k,3}(t^{n+1}) \\
    &= (\lambda_2 + \lambda_4)\int_{t^n}^{t^{n+1}} \exp((\lambda_1 + \lambda_3)(t - s))(u_k^p(s) - u_{k-1}^p(s)) \, ds, \\
    u_{k+1,3}(t^{n+1}) &= u_{k,3}(t^{n+1}) + \Delta u_{k+1,3}(t^{n+1}), \quad k = 0, 1, 2, \ldots,
\end{align*}
\]
where \( u_0(t) = u(t^0) \) and for \( u_{k+1}(t) = u_{k+1}(t) \), we have the two-level method. Further, we apply the trapezoidal or Simpson’s rule for the numerical integration of the integrals. Further, for the next time step, we have \( u(t^{n+1}) = u_{k+1}(t^{n+1}) \) and \( K \) is the number of iterative steps, e.g. \( K = 4 \).

\[
\begin{align*}
\mathbf{u}_{k+1}(t^{n+1}) &= u_k(t^{n+1}) \\
&+ \int_{t^n}^{t^{n+1}} \exp((\lambda_1 + \lambda_3)(t-s))((\lambda_2 + \lambda_4)u_k^p(s) - (\lambda_2 + \lambda_4)u_{k-1}^p(s)) \, ds,
\end{align*}
\]

\( k = 0, 1, 2, \ldots, \)  

where \( u_0(t) = u(t^n) \) and \( k \) is the number of iterative steps.

We apply the trapezoidal or Simpson’s rule for the numerical integration of the integrals.

\[
\begin{align*}
u_1(t) &= \exp((\lambda_1 + \lambda_3)(t-t^n))u(t^n) + \int_{t^n}^{t} \exp((\lambda_1 + \lambda_3)(t-s))((\lambda_2 + \lambda_4)u_k^p(s) - (\lambda_2 + \lambda_4)u_{k-1}^p(s)) \, ds,
\end{align*}
\]

\textbf{Remark 8} If we apply only the trapezoidal rule, we obtain less accurate results while we skip the term \( \exp((\lambda_1 + \lambda_3)(\Delta t))(\lambda_2 + \lambda_4)(u_k^p(t^n) - u_{k-1}^p(t^n)) = 0 \), such that it is important to have with \( k = 2 \) a third-order integration method, e.g. the Simpson’s rule.

Based on the reference solution, we deal with the following errors:

\[
\begin{align*}
E_{k, k, \Delta t, [0, t]} &= |u_{\text{method}, k, \Delta t}(t) - u_{\text{ref}, \Delta t}(t)| \\
&= \sum_{n=0}^{N} \Delta t|u_{\text{method}, k, \Delta t}(t^n) - u_{\text{ref}, \Delta t}(t^n)|, \quad (63)
\end{align*}
\]

where \( \text{method} \) is the different Picard’s methods and \( k \) is the number of iterative steps.

Further, we deal with different time steps \( \Delta t = \Delta t_{\text{CFL}}, \Delta t_{\text{CFL}}/2, \Delta t_{\text{CFL}}/4 \).

A restriction based on the CFL condition is not necessary, while we have a numerical integration, but for more accurate results, this can help. The CFL condition is given as:

\[
\Delta t \leq \frac{1}{(\lambda_1 + \lambda_2) + (\lambda_3 + \lambda_4)u_0^{p-1}} \quad (64)
\]

where we assume \( u^{n+1}(t) \leq u^n, \forall t \in [0, T] \) and \( u_0 \) is the initial solution.

For another accurate solution, we can integrate:

We have given:

\[
\begin{align*}
\mathbf{u}_{k+1}(t^{n+1}) &= u_k(t^{n+1}) \\
&+ \int_{t^n}^{t^{n+1}} \exp((\lambda_1 + \lambda_3)(t-s))((\lambda_2 + \lambda_4)u_k^p(s) - (\lambda_2 + \lambda_4)u_{k-1}^p(s)) \, ds \\
&= u_k(t^{n+1}) + \Delta t \phi_1(\Delta t((\lambda_1 + \lambda_3))g(t^n) + \frac{\Delta t^2}{2} \phi_2(\Delta t((\lambda_1 + \lambda_3))g'(t^n) \\
&+ O(\Delta t^3),
\end{align*}
\]

\textbf{where}

\[
\begin{align*}
g(t^n) &= (\lambda_2 + \lambda_4)(u_k^p(t^n) - u_{k-1}^p(t^n)) \\
g'(t^n) &= (\lambda_2 + \lambda_4)p((\lambda_1 + \lambda_3)(u_k^p(t^n) - u_{k-1}^p(t^n)) \\
&+ (\lambda_2 + \lambda_4)(u_k^p(t^n)u_{k-1}^p(t^n) - u_{k-2}^p(t^n)u_{k-2}^p(t^n))).
\end{align*}
\]
Further, we have:

\[ \phi_1(hA) = \frac{1}{h} \int_0^h \exp((h - \tau)A) d\tau, \quad (69) \]

\[ \phi_i(hA) = \frac{1}{h^i} \int_0^h \exp((h - \tau)A) \frac{\tau^{i-1}}{(i-1)!} d\tau, \quad i \geq 1, \quad (70) \]

and we apply to our formulas:

\[ \phi_1(\Delta t(\lambda_1 + \lambda_2)) = \frac{1}{\Delta t(\lambda_1 + \lambda_2)}(\exp(\Delta t(\lambda_1 + \lambda_2)) - 1), \quad (71) \]

\[ \phi_2(\Delta t(\lambda_1 + \lambda_2)) = \frac{1}{(\Delta t(\lambda_1 + \lambda_2))^2}(\exp(\Delta t(\lambda_1 + \lambda_2)) - 1) \]

\[ - \frac{1}{\Delta t(\lambda_1 + \lambda_2)}, \quad (72) \]

We apply the Picard’s iteration and compared to the analytical solutions, we obtain the following result in Figure 5.

**Remark 9** We saw marginal difference between the two- and three-level Picard’s methods with different iterative steps \( k = 1, 2, 5, 10 \). At least, it is important to deal with more than \( k \approx 3 \) iterative steps. Further, it is necessary to deal with accurate numerical integration methods to save the accuracy of the multilevel nonlinear methods, meaning we applied at minimum third- or fourth-order accurate methods.

**4.3. Multicomponent diffusion model**

We deal with the plasma model that we presented in Section 2. The multicomponent diffusion model, which is nonlinear in the diffusion operator, solves with the multilevel Picard’s method.

In the following, we present a three-step method, which is based on the idea of only updating the fix-point results.

To circumvent the exponential functions, we could apply for our nonlinear method a Picard’s iterative method, which is based on the following idea.

The nonlinear equation is given as:

\[ \partial_t \xi_i = -\nabla N_i(\xi_i), \quad \xi_i(0) = \xi_{i,0}, \quad (73) \]

we apply the three-step method given as:

\[ \xi_{i,k}(t) = \xi_{i,k-1}(t) + \int_0^t \left( -\nabla N_i(\xi_{i,k-1}(s)) + -\nabla N_i(\xi_{i,k-2}(s)) \right) ds, \quad (74) \]

where the initialization is given with the initial function \( \xi_{i,0}(t) = \xi_i(0) \) and the first iteration is done previously.

The explicit form of the algorithm with an explicit Euler time discretization is given as:
Algorithm 3  1.) Initialisation $k = 0$ with an explicit time-step (CFL condition is given):

\[
\begin{pmatrix}
N_1^0 \\
N_2^0
\end{pmatrix} = \begin{pmatrix}
\bar{A} & \bar{B} \\
\bar{C} & \bar{D}
\end{pmatrix} \begin{pmatrix}
-D_1 \xi_1^0 \\
-D_2 \xi_2^0
\end{pmatrix},
\]  
\hspace{10cm} \text{(75)}

where $\xi_1^0 = (\xi_{1,0}^0, \ldots, \xi_{1,J}^0)^T$, $\xi_2^0 = (\xi_{2,0}^0, \ldots, \xi_{2,J}^0)^T$ and $\xi_1^k = \xi_1^{in}(k \Delta x)$, $\xi_2^k = \xi_2^{in}(k \Delta x)$, $j = 0, \ldots, J$ and given as for the different initializations, we have:

1. Uphill example

\[
\xi_1^{in}(x) = \begin{cases}
0.8 & \text{if } 0 \leq x < 0.25, \\
1.6(0.75 - x) & \text{if } 0.25 \leq x < 0.75, \\
0.0 & \text{if } 0.75 \leq x \leq 1.0,
\end{cases}
\]  
\hspace{10cm} \text{(76)}

\[
\xi_2^{in}(x) = 0.2, \text{ for all } x \in \Omega = [0, 1],
\]  
\hspace{10cm} \text{(77)}

2. Diffusion example (Asymptotic behavior)

\[
\xi_1^{in}(x) = \begin{cases}
0.8 & \text{if } 0 \leq x \leq 0.5, \\
0.0 & \text{else},
\end{cases}
\]  
\hspace{10cm} \text{(78)}

\[
\xi_2^{in}(x) = 0.2, \text{ for all } x \in \Omega = [0, 1],
\]  
\hspace{10cm} \text{(79)}

The inverse matrices are given as:

\[
\bar{A}, \bar{B}, \bar{C}, \bar{D} \in \mathbb{R}^{J+1} \times \mathbb{R}^{J+1},
\]  
\hspace{10cm} \text{(80)}

\[
\bar{A}_{j,j} = \gamma_j \left( \frac{1}{D_{23}} + \beta \xi_1^j \right), \ j = 0, \ldots, J,
\]  
\hspace{10cm} \text{(81)}

\[
B_{j,j} = \gamma_j \alpha \xi_1^j, \ j = 0, \ldots, J,
\]  
\hspace{10cm} \text{(82)}

\[
C_{j,j} = \gamma_j \beta \xi_2^j, \ j = 0, \ldots, J,
\]  
\hspace{10cm} \text{(83)}

\[
D_{j,j} = \gamma_j \left( \frac{1}{D_{13}} + \alpha \xi_2^j \right), \ j = 0, \ldots, J,
\]  
\hspace{10cm} \text{(84)}

\[
\gamma_j = \frac{D_{13}D_{23}}{1 + \alpha D_{13} \xi_2^j + \beta D_{23} \xi_1^j}, \ j = 0, \ldots, J,
\]  
\hspace{10cm} \text{(85)}

\[
\bar{A}_{i,j} = \bar{B}_{i,j} = \bar{C}_{i,j} = \bar{D}_{i,j} = 0, \ i, j = 0, \ldots, J, \ i \neq j.
\]  
\hspace{10cm} \text{(86)}

Further the values of the first and the last grid points of $N$ are zero, means $N_{1,0}^0 = N_{1,J}^0 = N_{2,0}^0 = N_{2,J}^0 = 0$ (boundary condition).

2.) Next timesteps (till $n = N_{end}$) (iterative scheme restricted via the CFL condition based on the previous iterative solutions in the matrices):
2.1) Computation of $k = 1$ with $\xi_1^{n+1,1}$ and $\xi_2^{n+1,1}$:

\[
\begin{align*}
\xi_1^{n+1,1} &= \xi_1^n - \Delta t \, D_+ N_1^{n+1,0}, \\
\xi_2^{n+1,1} &= \xi_2^n - \Delta t \, D_+ N_2^{n+1,0},
\end{align*}
\]  

(87) \hspace{1cm} (88)

2.2) Computation of $N_1^{n+1,1}$ and $N_2^{n+1,1}$:

\[
\begin{pmatrix}
N_1^{n+1,1} \\
N_2^{n+1,1}
\end{pmatrix} =
\begin{pmatrix}
\hat{A}^{n+1,1} & \hat{B}^{n+1,1} \\
\hat{C}^{n+1,1} & \hat{D}^{n+1,1}
\end{pmatrix}
\begin{pmatrix}
-D_+ \xi_1^{n+1,1} \\
-D_+ \xi_2^{n+1,1}
\end{pmatrix},
\]

(89)

where $\xi_1^n = (\xi_1^{n,0}, \ldots, \xi_1^{n,J})^T$, $\xi_2^n = (\xi_2^{n,0}, \ldots, \xi_2^{n,J})^T$.

2.3.1) 2 step-Method: Computation of $k = 1$ (with $k = 0$) ($\xi_1^{n+1,k}$, $\xi_2^{n+1,k}$ from $\xi_1^{n+1,k-1}$, $\xi_2^{n+1,k-1}$):

\[
\begin{align*}
\xi_1^{n+1,k} &= \xi_1^n - \Delta t \, D_+ N_1^{n+1,k-1}, \\
\xi_2^{n+1,k} &= \xi_2^n - \Delta t \, D_+ N_2^{n+1,k-1},
\end{align*}
\]

(90) \hspace{1cm} (91)

2.3.2) 3 step-Method: Computation of $k = 2$ (with $k = 1$, $k = 0$) ($\xi_1^{n+1,k}$, $\xi_2^{n+1,k}$ from $\xi_1^{n+1,k-1}$, $\xi_1^{n+1,k-2}$, $\xi_1^{n+1,k-3}$, $\xi_2^{n+1,k-1}$, $\xi_2^{n+1,k-2}$):

\[
\begin{align*}
\xi_1^{n+1,k} &= \xi_1^n - \Delta t \, D_+ N_1^{n+1,k-1} + \Delta t \, D_+ N_1^{n+1,k-2}, \\
\xi_2^{n+1,k} &= \xi_2^n - \Delta t \, D_+ N_2^{n+1,k-1} + \Delta t \, D_+ N_2^{n+1,k-2},
\end{align*}
\]

(92) \hspace{1cm} (93)

2.3.3) 4 step-Method: Computation of $k \geq 3$ (with $k = 1$, $k = 2$, $k = 3$) ($\xi_1^{n+1,k}$, $\xi_2^{n+1,k}$ from $\xi_1^{n+1,k-1}$, $\xi_1^{n+1,k-2}$, $\xi_1^{n+1,k-3}$, $\xi_2^{n+1,k-1}$, $\xi_2^{n+1,k-2}$, $\xi_2^{n+1,k-3}$):

\[
\begin{align*}
\xi_1^{n+1,k} &= 2\xi_1^n - \xi_1^{n+1,k-2} - \Delta t \, D_+ N_1^{n+1,k-1} + 2\Delta t \, D_+ N_1^{n+1,k-2} - \Delta t \, D_+ N_1^{n+1,k-3}, \\
\xi_2^{n+1,k} &= 2\xi_2^n - \xi_2^{n+1,k-2} - \Delta t \, D_+ N_2^{n+1,k-1} + 2\Delta t \, D_+ N_2^{n+1,k-2} - \Delta t \, D_+ N_2^{n+1,k-3},
\end{align*}
\]

(94) \hspace{1cm} (95)

2.4) Computation of $N_1^{n+1,k}$ and $N_2^{n+1,k}$:

\[
\begin{pmatrix}
N_1^{n+1,k} \\
N_2^{n+1,k}
\end{pmatrix} =
\begin{pmatrix}
\hat{A}^{n+1,k} & \hat{B}^{n+1,k} \\
\hat{C}^{n+1,k} & \hat{D}^{n+1,k}
\end{pmatrix}
\begin{pmatrix}
-D_+ \xi_1^{n+1,k} \\
-D_+ \xi_2^{n+1,k}
\end{pmatrix},
\]

(96)

where $\xi_1^n = (\xi_1^{n,0}, \ldots, \xi_1^{n,J})^T$, $\xi_2^n = (\xi_2^{n,0}, \ldots, \xi_2^{n,J})^T$.

Further the values of the first and the last grid points of $N$ are zero, means $N_1^{n+1,0} = N_1^{n+1,J} = N_2^{n+1,0} = N_2^{n+1,J} = 0$ (boundary condition).

Further $\xi_1^{n+1,0} = (\xi_1^{n,0}, \ldots, \xi_1^{n,J})^T$, $\xi_2^{n+1,0} = (\xi_2^{n,0}, \ldots, \xi_2^{n,J})^T$ and $I_J \in \mathbb{R}^{J+1 \times J+1}$ is the start solution given with the solution at $t = t^n$.

Repeat 2.3.) and 2.4.) with $k = 2, 3, \ldots, K$ and $K$ is the maximal iteration index.

3.) Do $n = n + 1$ and goto 2.)
The computation of the inverse matrices is given as:

\[
\hat{A}_{n+1,k-1}^{\pm}, B_{n+1,k-1}^{\pm}, C_{n+1,k-1}^{\pm}, D_{n+1,k-1}^{\pm} \in \mathbb{R}^{J \times J},
\]

(97)

\[
\hat{A}_{j,j}^{n+1,k-1} = \gamma_j \left( \frac{1}{D_{23}^{j,j}} + \beta \xi_{1,j}^{n+1,k-1} \right), \quad j = 0 \ldots, J,
\]

(98)

\[
B_{j,j}^{n+1,k-1} = \gamma_j \alpha \xi_{1,j}^{n+1,k-1}, \quad j = 0 \ldots, J,
\]

(99)

\[
C_{j,j}^{n+1,k-1} = \gamma_j \beta \xi_{2,j}^{n+1,k-1}, \quad j = 0 \ldots, J,
\]

(100)

\[
D_{j,j}^{n+1,k-1} = \gamma_j \left( \frac{1}{D_{13}^{j,j}} + \alpha \xi_{2,j}^{n+1,k-1} \right), \quad j = 0 \ldots, J,
\]

(101)

\[
\gamma_j = \frac{1}{1 + \alpha D_{13}^{j,j} + \beta D_{23}^{j,j} \xi_{1,j}^{n+1,k-1}}, \quad j = 0 \ldots, J,
\]

(102)

\[
\hat{A}_{j,j}^{n+1,k-1} = B_{j,j}^{n+1,k-1} = C_{j,j}^{n+1,k-1} = D_{j,j}^{n+1,k-1} = 0,
\]

(103)

\[
i,j = 0 \ldots, J, \quad i \neq j.
\]

The numerical errors of the different schemes are given in Figure 6.

The numerical solution of the three- and four-level Picard’s fix-point schemes with different iterative steps is given in Figure 7.

The numerical errors of the four-level Picard’s fix-point schemes compared with a reference solution of fine time steps of a three-level method (see Figure 8).

The solutions of the numerical experiments are given in Figure 9.

Remark 10 We tested all different level properties, meaning three-level and four-level Picard’s fix-point schemes, with different iterative step sizes, meaning \( k = 3, 4, 5, 10 \). For all applications, we saw only marginal differences, such that a three-level method is sufficient to resolve the nonlinear problem.
5. Conclusions and discussion

We present the coupled model for a multicomponent transport model, which can be applied for solving nonlinear diffusion equations, e.g. for multicomponent plasma transport models. The Picard’s methods are flexible and we could derive multilevel methods, with exponential treatments. The benefit of the methods is to resolve the nonlinearity with more accuracy based on the intermediate
levels. Here, we present the improvements based on the blow-up problems and the Bernoulli’s equation. In multidiffusion applications, we saw the benefit of such methods in a fast resolution of the nonlinear diffusion and a decomposition on slow and fast time scales. Overall, the flexibility of such Picard’s methods is important to solve multicomponent problems.

Funding
The author received no direct funding for this research.

Author details
Juergen Geiser
E-mail: juergen.geiser@ruhr-uni-bochum.de
ORCID ID: http://orcid.org/0000-0003-1093-0001
1 Department of Electrical Engineering and Information Technology, Ruhr University of Bochum, Universitätsstrasse 150, D-44801 Bochum, Germany.

Citation information
Cite this article as: Picard’s iterative method for nonlinear multicomponent transport equations, Juergen Geiser, Cogent Mathematics (2016), 3: 1158510.

References
Geiser, J. (2008). Iterative operator-splitting methods with higher order time-integration methods and applications for parabolic partial differential equations. Journal of Computational and Applied Mathematics, 217, 227–242.

Hirschfelder, J. O., Curtiss, Ch. F., & Bird, R. B. (1966). Molecular theory of gases and liquids (Chemistry) (1st ed.). New York, NY: Wiley.
Igitkhanov, Y. (2011). Modelling of multi-component plasma for TOKES (KIT Scientific Report Nr. 7564). Karlsruhe: Institut für Hochleistungsimpuls- und Mikrowellentechnik (IHM), KIT Scientific Publishing.
Ramos, J. I. (2008). On the variational iteration method and other iterative techniques for nonlinear differential equations. Applied Mathematics and Computation, 199, 39–69.
Ramos, J. I. (2009). Picard’s iterative method for nonlinear advection reaction-diffusion equations. Applied Mathematics and Computation, 215, 1526–1536.
Senega, T. K., & Brinkmann, R. P. (2006). A multi-component transport model for non-equilibrium low-temperature low-pressure plasmas. Journal of Physics D: Applied Physics, 39, 1606–1618.
Tanaka, Y. (2004). Two-temperature chemically non-equilibrium modelling of high-power Ar-N2 inductively coupled plasmas at atmospheric pressure. Journal of Physics D: Applied Physics, 37, 1190–1205.