Journal of Computational Electronics (2024) 23:718–727
https://doi.org/10.1007/s10825-024-02178-1

# Efficiency analysis of discontinuous Galerkin approaches for the application onto quantum Liouville-type equations

**Valmir Ganiu**[1] · **Dirk Schulz**[1]

Received: 12 December 2023 / Accepted: 7 May 2024 / Published online: 4 June 2024


**Abstract**
The simulation of nanodevices is computationally inefficient with current algorithms. The discontinuous Galerkin approach has been demonstrated in the field of computational fluid dynamics to deliver high order accuracy and efficiency due to its reliance on matrix–vector multiplications. Previously, the discontinuous Galerkin approach was successfully used in conjunction with the finite volume technique to solve the Liouville–von Neumann equation in center-mass coordinates and thus simulate nanodevices. To exploit its full potential regarding high-performance computing, this work aims to substitute the aforementioned finite volume technique with the discontinuous Galerkin method. To arrive at the said formalism, a finite element method is implemented as an intermediate step.

**Keywords** Discontinuous Galerkin methods · Finite element methods · Liouville–von Neumann equation · Quantum transport · Nanodevices

## 1 Introduction

With numerous algorithms proposed to solve quantum transport equations, like for example the finite volume (FV) based techniques [1], finite difference schemes [2], and the Green's function approach [3], computational efficiency is further improvable. Specifically transient simulations are of interest, as time savings can be substantial here. Conventional Galerkin methods are a possibility to build upon and improve the accuracy of numerical approximations of quantum transport equations.

The discontinuous Galerkin (DG) method is successfully applied in computational fluid dynamics [4], but can also be adapted for conservative problems of a quantum nature. Here, the requirements are not exclusively the realization of computational efficiency, but also the adequate inclusion of quantum effects with sufficient accuracy.

The DG method operates on the basis of the following principles enabling it for the use in high-performance computing. Being based on the finite element method (FEM), transient calculations rely on matrix–vector multiplication when applying the DG approach, making it specifically suitable for the approximation of transient exponential integrators. To approximate the transient exponential operator by the use of matrix–vector multiplications, a variety of algorithms including the Krylov methods [5], Faber polynomial-based algorithms [6], or Runge–Kutta schemes [7–9] can be utilized. A further benefit of the scheme is the use of block-diagonal matrices. Thus, the computation of the appearing inverse matrices is especially efficient. At this stage, matrices are of a low order, allowing for a more efficient inclusion into the scheme.

Nevertheless, this method suffers from inherent stability issues considering time-dependent analyses which have to be resolved [7]. An appropriate choice of the numerical flux and boundary conditions can alleviate these problems [10]. Therefore, it is expedient to simulate nanodevices on the basis of quantum Liouville-type equations in center of mass coordinates with the DG algorithm.

As proposed in [11], a hybrid DG method has already been proved to be a stable algorithm and deliver accurate results when simulating nanostructures. However, the proposed method is of a hybrid nature by utilizing the FV technique to approximate the relative coordinate and the DG scheme to handle the center of mass coordinate. The use

✉ Valmir Ganiu
valmir.ganiu@tu-dortmund.de

Dirk Schulz
dirk2.schulz@tu-dortmund.de

[1] Chair for Communication Technology, TU Dortmund, Otto Hahn Str. 4, 44227 Dortmund, NRW, Germany

of the FV technique was first provided as to better assess the influence of the numerical flux introduced by the DG method on the carrier density, which is considerable and demonstrated in [12].

To arrive at a full DG-based algorithm, this paper seeks to substitute the FV technique by a second DG method, hence approximating both, the center-mass and relative, coordinates of quantum Liouville-type equations with the DG method. To do so, an intermediate step is undertaken by introducing the finite element method (FEM) instead of the FV approach since the FEM and DG methods are closely related. The use of the FEM introduces well-known stiffness and mass matrices, which are also used in the DG scheme. Finally, to arrive at the DG method, the finite elements are decoupled and linked via a numerical flux.

This work is divided into six sections. Section 2 will briefly introduce the Liouville–von Neumann equation, while also presenting the current methodology and giving some important definitions. The discretization of the relative and center of mass coordinates will be dealt with in Sects. 3 and 4, respectively. Simultaneously, Sect. 3 will present the FEM and DG concepts. Section 5 will discuss the numerical results obtained with the proposed algorithm followed by a conclusion in Sect. 6.

## 2 Fundamentals

The basis of this investigation is the Liouville–von Neumann equation (LVNE) for the one-dimensional transport problem [6]. Furthermore, the effective mass $m$ is considered to be spatially constant [13]. On the basis of these conditions, the LVNE reads as [6]

$$\frac{\partial}{\partial t}\rho(\chi,\xi,t) = \imath\frac{\hbar}{m}\frac{\partial}{\partial\chi}\frac{\partial}{\partial\xi}\rho(\chi,\xi,t) - \imath\frac{q}{\hbar}B(\chi,\xi,t)\rho(\chi,\xi,t). \tag{1}$$

$\rho$ denotes the statistical density, $q$ represents the elementary charge, and $B$ describes the static conduction band, as well as effects of the externally applied bias [13] comprised by the function $V$ and is defined according to

$$B(\chi,\xi,t) = V\left(\chi+\frac{\xi}{2},t\right) - V\left(\chi-\frac{\xi}{2},t\right).$$

### 2.1 Current methodology and definition

In [11], the FV method was utilized in the $\xi$-direction as to better assess the influence of the numerical flux, introduced by the DG scheme in the $\chi$-direction, on the statistical density. Since this was thoroughly analyzed in [10], the DG algorithm can now be introduced to discretize the $\xi$-domain according to Frensley's concept [14]. However, as mentioned above, before utilizing the DG method, an FEM approximation will be utilized as a preliminary step since it is the basis for the DG algorithm. The combination of the DG method in the $\chi$-domain and the FEM in the $\xi$-domain results in another hybrid formalism. Once the FEM is transformed into the DG scheme, the combination with the DG method in the $\chi$ -domain results in a double DG formalism. Accordingly, three notations for the algorithms shall be introduced to simplify addressing them:

- The DG method to approximate the $\chi$-coordinate and the FV technique to approximate the $\xi$-coordinate is denoted by **DGFV**
- The DG method to approximate the $\chi$-coordinate and the FEM scheme to approximate the $\xi$-coordinate is denoted by **DGFEM**
- The DG method to approximate the $\chi$-coordinate *and* the $\xi$-coordinate is denoted by **DGDG**

Generally speaking, the finite difference (FD) scheme could be used as well. However, it has been already shown in other scientific fields that the FD scheme lacks in accuracy when compared to the FEM [15, 16]. For this reason, the FD scheme will not be considered. To simplify the derivation, and because the appearing well-known stiffness and mass matrices are identical between the FEM and DG method, a generalized coordinate is introduced to cover both cases according to $\sigma = \chi, \xi$.

## 3 Discretizing the $\xi$-coordinate

### 3.1 Finite element approximation

To approximate (1) considering the $\xi$-domain, the conventional FEM technique is implemented on an equidistant grid. The computational domain $\Omega_\xi$ within the interval $D^k = [-L_\xi/2, L_\xi/2]$ is divided into a number $N_\xi$ of evenly spaced finite elements with a width of $\Delta_\xi$, whereby two neighboring finite elements share one node. A linear ansatz for the nodal functions is chosen for the approximating functions resulting in a total number of nodes with $N_{\xi p} = N_\xi + 1$. To find a discrete solution for each element, test functions $l_{i,j}(\sigma)$ must be specified. These functions are piece-wise defined polynomials of order $N = N_p - 1$ and form a $N$-dimensional finite element [11]. Next, the test functions are integrated over each element $D^k$. To approximate the partial derivative $\partial/\partial\chi$, the FEM-typical local

stiffness and mass matrices $\boldsymbol{S}$, $\boldsymbol{M}$ are introduced and their matrix elements are defined according to

$$\boldsymbol{S}^k_{i,j}(\sigma) \equiv \int_{D^k} l^k_i(\sigma)\frac{\partial}{\partial\sigma} l^k_j(\sigma)d\sigma \tag{2}$$

$$\boldsymbol{M}^k_{i,j}(\sigma) \equiv \int_{D_k} l^k_i(\sigma) l^k_j(\sigma)d\sigma, \tag{3}$$

resulting in $2\times 2$ matrices due to the above chosen linear ansatz. To preserve continuity of the statistical density $\rho$ at node junctions between finite elements, matrix elements $S^k_{i,j}$ of the local matrices are overlapped and added according to the following scheme:

$$\boldsymbol{S}' = \begin{pmatrix} S^1_{1,1} & S^1_{1,2} & 0 & \cdots & 0 \\ S^1_{2,1} & S^1_{2,2}+S^2_{1,1} & S^2_{1,2} & \cdots & 0 \\ 0 & S^2_{2,1} & S^2_{2,2}+S^3_{1,1} & S^3_{1,2} & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & S^{k-1}_{2,2}+S^k_{1,1} & S^k_{1,2} \\ 0 & \cdots & \cdots & S^k_{2,1} & S^k_{2,2} \end{pmatrix}$$

The potential values are gathered as $2\times 2$ matrices and assembled similar to the above scheme for deriving $\boldsymbol{S}'$. The nonzero elements $\boldsymbol{B}_{i,j}(\chi,t)$ are defined as

$$\boldsymbol{B}_{i,j}(\chi,t) = c_B(B(\chi,\xi_{i-\frac{1}{2}},t)+B(\chi,\xi_{i+\frac{1}{2}},t))\cdot\boldsymbol{M}^k(\chi), \tag{4}$$

whereby $c_B = \imath q/\hbar$. The values $\xi_{i\pm\frac{1}{2}}$ represent the coordinates of the interfaces for the $i$-th cell. The potential values must be further multiplied with the mass matrix $\boldsymbol{M}^k$ obtained from (3) for each finite element. Finally, the values $B_{i,j}(\chi,t)$ must be integrated over each $\xi$-element as discussed in [17]. To approximate the integral, the Gauss–Lobatto quadrature is utilized according to [17]

$$\tilde{\boldsymbol{B}}_{j,m} = \sum_{a=1}^{N_{GL}} (\tilde{V}V^{-1})_{ai}(\tilde{V}V^{-1})_{aj} B_{j,m}(\chi,t) w_a J^k. \tag{5}$$

Here, $V$ is the Vandermonde matrix, $w_a$ is the Gauss–Lobatto weights, and $J^k$ is the Jacobi matrix. The partially discretized von Neumann equation (1) derived with the FEM algorithm now reads as

$$\frac{\partial}{\partial t}\boldsymbol{\rho}(\chi,t) = \boldsymbol{S}'\frac{\partial}{\partial\chi}\boldsymbol{\rho}(\chi,t) - \tilde{\boldsymbol{B}}(\chi,t)\boldsymbol{\rho}(\chi,t) \tag{6}$$

and consists of coupled partial differential equations depending on the center-mass coordinate $\chi$. The statistical density vector $\boldsymbol{\rho}(\chi,t)$ contains the values of the density matrix $\boldsymbol{\rho}$ at the discrete locations $\xi_j$.

### 3.2 Discontinuous Galerkin scheme

The computational domain $\Omega_\xi$ is again divided into $N_\xi$ of now non-overlapping finite elements with a width of $\Delta_\xi$. This time, neighboring finite elements do not share the same node. Instead, each finite element has its individual nodes holding individual values. Therefore, the linear ansatz with $N_p = 2$ now yields a total of $N_{\xi p} = 2\cdot N_\xi$ nodes. Although the local stiffness matrix (2) is identical to the one used in the FEM approach, deriving the total stiffness matrix differs according to the following scheme

$$\boldsymbol{S}^* = \begin{pmatrix} S^1_{1,1} & S^1_{1,2} & 0 & 0 & \cdots & 0 \\ S^1_{2,1} & S^1_{2,2} & 0 & 0 & \cdots & 0 \\ 0 & 0 & S^2_{1,1} & S^2_{1,2} & \cdots & 0 \\ 0 & 0 & S^2_{2,1} & S^2_{2,2} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \cdots & \vdots \\ 0 & \cdots & \cdots & \cdots & S^k_{1,1} & S^k_{1,2} \\ 0 & \cdots & \cdots & \cdots & S^k_{2,1} & S^k_{2,2} \end{pmatrix},$$

resulting in a block-diagonal matrix. To achieve continuity, a so-called numerical flux is deployed. Although the DG method enables the free choice of the numerical flux, of which there are a variety of options available [4], one factor limits the choice for the $\xi$-domain. The inflow and outflow characteristics are unknown. Hence, a central flux has to be chosen, which equally includes both contributions. To derive the numerical flux, the $\xi$-derivative affecting $\rho(\chi,\xi,t)$ in (1) is examined closely.

First, lift operators $p_n(\sigma)$ are introduced and defined as

$$p_n(\sigma^k_l) = (1,0)^T$$
$$p_n(\sigma^k_r) = (0,1)^T$$

for the left and right edge of each element, respectively. Next, a two stage partial integration is performed on (2) to arrive at the strong formulation [12]. The double valued flux is substituted by a single valued numerical flux at each node as described in [12] resulting in

$$[l_i(\sigma)\cdot f(\sigma)]^{\sigma_r}_{\sigma_l} - [l_i(\sigma)\cdot f^*(\sigma)]^{\sigma_r}_{\sigma_l} + \int l_i(\sigma)\frac{\partial}{\partial\sigma} l_j(\sigma)d\sigma.$$

Considering that the central flux can be expressed by $f^*_{l/r} = (f^-_{l/r}+f^+_{l/r})/2$, the numerical flux for the left edge of each element reads as $f^*_{\sigma_l} = -f^k_{\sigma_l} + (f^{k-1}_{\sigma_r}+f^k_{\sigma_l})/2$, while the numerical flux at the right edge for each element equals to $f^*_{\sigma_r} = f^k_{\sigma_r} - (f^k_{\sigma_r}+f^{k+1}_{\sigma_l})/2$. Here, the superscript '$-$' refers to the interior information, whereas '+' is related to the exterior information of the element. For clarity of the notation refer to Fig. 1, where the schematic representation of the numerical flux is provided.

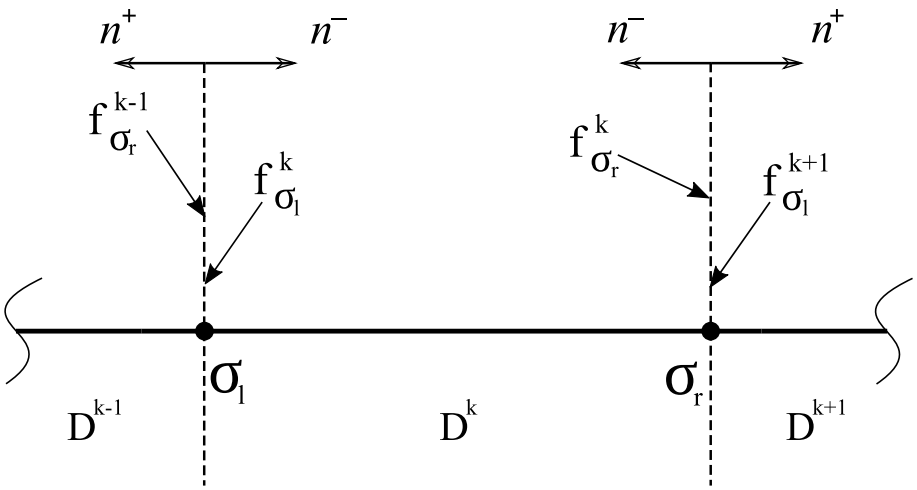


**Fig. 1** Schematic representation of the numerical flux $f$ for one $\sigma = \xi, \chi$-element

At last, the discretization matrix $\boldsymbol{S}''$ can be assembled according to the strong formulation [12]

$$\boldsymbol{S}'' = \boldsymbol{S}^* - \boldsymbol{F}. \tag{7}$$

Where $\boldsymbol{S}^*$ is the total stiffness matrix, $\boldsymbol{F}$ holds the values of the numerical flux for each element across the computational domain $\Omega_\xi$. The partially discretized LVNE (1) derived with the DG algorithm can finally be written as

$$\frac{\partial}{\partial t}\rho(\chi,t) = \boldsymbol{S}''\frac{\partial}{\partial \chi}\rho(\chi,t) - \tilde{\boldsymbol{B}}(\chi,t)\rho(\chi,t). \tag{8}$$

### 3.3 Semi-discretized quantum Liouville-type equations

To solve a semi-discretized von Neumann equation, open boundary conditions are necessary. However, these are unknown in the real space, wherefore inflow boundary conditions [14] can be incorporated. Subsequently, a distinction between forward and backward propagating waves is needed. For this purpose, the discretization matrices $\boldsymbol{S}'$ and $\boldsymbol{S}''$ obtained from the FEM and DG methods, respectively, are diagonalized with an eigenvalue decomposition according to $\boldsymbol{\Lambda} = \boldsymbol{\Phi}^\dagger \boldsymbol{S}'^{(\prime\prime)}\boldsymbol{\Phi}$. This allows an expansion of the semi-discrete density matrix $\boldsymbol{\rho}$ according to

$$\boldsymbol{\rho}(\chi,t) = \boldsymbol{\Phi}^k \cdot c^k(\chi,t) \qquad \forall \quad k = 1...N_{\xi p}, \tag{9}$$

utilizing the matrix $\boldsymbol{\Phi}^k$, containing the eigenvectors of $\boldsymbol{S}'$ or $\boldsymbol{S}''$ columnwise, and the expansion coefficients $c^k$. It is important to note that the expansion order $N_{\xi p}$ must be even to avoid singularity issues. Finally, the expansion in (9) is applied onto (6) and (8). Coupled with a basis transformation of $\boldsymbol{\Phi}^k$ and $\tilde{\boldsymbol{B}}$ in (6) and (8) from the left-hand side the discrete quantum Liouville-type equation

$$\begin{aligned}\frac{\partial}{\partial t}\boldsymbol{c}(\chi,t) =& \boldsymbol{\Phi}^\dagger \boldsymbol{S}'^{(\prime\prime)}\boldsymbol{\Phi}\frac{\partial}{\partial \chi}\boldsymbol{c}(\chi,t) \\ &- \boldsymbol{\Phi}^\dagger \tilde{\boldsymbol{B}}(\chi,t)\boldsymbol{\Phi}\cdot\boldsymbol{c}(\chi,t)\end{aligned} \tag{10}$$

for the FEM and DG scheme results. The diffusion matrices $\boldsymbol{S}'^{(\prime\prime)}$, as well as the drift matrix $\tilde{\boldsymbol{B}}$, are diagonalized by utilizing the aforementioned eigenvalue decomposition.

## 4 Discretizing the $\chi$-coordinate

### 4.1 Discontinuous Galerkin approximation

For the discretization of the $\chi$-coordinate in (10), the DG algorithm is utilized as depicted in [11] and shall be evaluated briefly in the following.

The computational domain $\Omega_\chi$ within the interval $[0, L_\chi]$ is subdivided into a number $N_\chi$ of finite elements. Unlike the $\xi$-domain, the form functions are chosen to be quadratic ($N = 2$), thus resulting in $N_p = 3$ nodes for each finite element. As depicted previously in Sect. 3B, the discrete solution for each finite element is multiplied and integrated for each $\chi \in D^k$ resulting in

$$\begin{aligned}&\int_{D^k}\frac{\partial}{\partial t}\rho(\chi,t)l_j^k(\chi)d\chi - c_A\cdot\int_{D^k}\frac{\partial}{\partial \chi}\rho(\chi,t)l_j^k(\chi)d\chi \\ &\quad + \sum_{m=1}^{N_\xi}\int_{D^k} G_{jm}^k(\chi)\lambda_m^k(\chi,t)l_j^k(\chi) = 0,\end{aligned} \tag{11}$$

where $c_A$ holds the physical constants according to $c_A = \imath\hbar/(2m\Delta\xi)$. $\Delta\xi$ is the width of a $\xi$-element. Additionally, $G_{jm}^k(\chi)$ represents the matrix elements of $\boldsymbol{\Phi}^\dagger\tilde{\boldsymbol{B}}\boldsymbol{\Phi}$ in (10). In a next step, the term containing the $\chi$ -derivative is partially integrated two times to arrive at the

strong formulation. Together with the relationship (2) and (3), (10) can be rewritten to [11]

$$\begin{aligned} &\boldsymbol{M}^k \frac{\partial}{\partial t} + [l_n(\chi)\cdot f(\chi)]_{\chi_l}^{\chi_r} - [l_n(\chi)\cdot f^*(\chi)]_{\chi_l}^{\chi_r} \\ &= \boldsymbol{S}^k + \boldsymbol{M}^k \sum_{m=1}^{N_\xi} \mathrm{diag}(\boldsymbol{G}_{jm}^k) c_m^k, \qquad \forall \quad n = 1...N_p. \end{aligned} \tag{12}$$

For the time-dependent problem, the inverse of the mass matrix $(\boldsymbol{M}^k)^{-1}$ must be multiplied to (12). For the steady-state case, it is sufficient to assume $\boldsymbol{M}^k \partial_t = 0$. Similar to the approximation in the $\xi$-direction, the partial integration introduces a numerical flux, which ensures the coupling of the discontinuous finite elements. As shown in [10], a central flux in the $\chi$-direction causes instability when aiming to perform a transient simulation. However, since the direction of the propagating waves is known due to the eigenvalues derived from the discretization matrices in 3A and 3B, the upwind flux provides an adequate approximation to account for the flux characteristic of the differential equation (12) [10].

### 4.2 Definition of the upwinding numerical flux

The upwinding numerical flux takes into account the direction from where waves are coming from. To better understand this fact consider Fig. 1, where $D^k$ represents the finite element and $\sigma_l$ as well as $\sigma_r$ denotes the left and right edge of said element, respectively. Waves propagating either to the right (positive) or to the left (negative), indicated by the normals $n^-$ and $n^+$, require the definition of a numerical flux for each case. From (12) together with the following definition for an upwinding numerical flux ($\alpha = 0$)

$$\lambda f^* = \lambda \frac{f^- + f^+}{2} + \frac{1-\alpha}{2} |\lambda| (\hat{n}^- f^- + \hat{n}^+ f^+), \tag{13}$$

a numerical flux for forward propagating waves $f^{*,+}$ and backward propagating waves $f^{*,-}$ can be defined as

$$\begin{aligned} f^{*,+} =& f^k(\sigma_r) - \left( \frac{f^k(\sigma_r) + f^{k+1}(\sigma_l)}{2} \right. \\ &\left. + \frac{1}{2}(-f^k(\sigma_r) + f^{k+1}(\sigma_l)) \right), \\ f^{*,-} =& -f^k(\sigma_l) + \left( \frac{f^k(\sigma_l) + f^{k-1}(\sigma_r)}{2} \right. \\ &\left. + \frac{1}{2}(-f^k(\sigma_l) + f^{k-1}(\sigma_r)) \right), \end{aligned}$$

respectively.

### 4.3 Boundary conditions

To complete the approximation of the computational domain $\Gamma = \Omega_\chi \times \Omega_\xi$, boundary conditions must be defined. The propagating waves within the domain move with a certain velocity, which are identified by the eigenvalues $\lambda_j$ as derived in 3A and 3B. Due to singularity issues, $\lambda_j = 0$ must not appear. Therefore, all eigenvalues are symmetrically distributed around 0. The fixed boundary conditions consist of the union of Dirichlet ($\Gamma_D$) and Neumann ($\Gamma_N$) boundaries within the computational domain $\Gamma$. The edges of the $\chi$-domain along $\chi_l = 0$ and $\chi_r = L_\chi$ are defined by the conditions $g(\chi, \xi) = (\Phi^{-1})^\dagger \hat{f}(\xi)$, where $\hat{f}(\xi)$ is the Fourier transform of the Fermi–Dirac distribution $f(k)$ [18] and defined as

$$\hat{f}(\xi) = \frac{1}{2\pi} \int \cos(\xi k) f(k) \mathrm{d}k \qquad \text{with}$$
$$f(k) = \gamma \frac{m^*}{\pi \hbar^2 \beta} \ln\left(1 + e^{\beta(-e\hbar k^2 + \mu)}\right).$$

The matrix containing the eigenvectors $\boldsymbol{\Phi}_j$ is divided into the eigenvectors corresponding either to the positive or negative eigenvalues.

The LVNE (1) demands an infinitely expanding $\xi$ -domain such that waves can propagate into infinity [19]. However, since the aim is to numerically approximate (1), a finite computational domain is needed. Therefore, the $\xi$-domain along $\xi_l = -L_\xi/2$ and $\xi_r = +L_\xi/2$ must be terminated with open boundaries as shown in [20] to accomplish the above requirement. Mathematically speaking, an openly bounded computational domain leads to zero-valued Neumann boundary conditions. From a physical perspective, the closed boundary conditions instead of open boundary conditions cause reflections of the values of the density matrix, which appear as oscillations in the final result and do not represent the physical solution [21]. Therefore, a complex absorbing potential (CAP) at both edges of $\Omega_\xi$ is deployed, which acts as a dampening layer and decays the occurring reflections [22]. Hence, the CAP is deployed along $\xi_l$ and $\xi_r$ according to

$$W(\xi) = \begin{cases} W_0(\xi - \frac{L_\xi}{2} + \delta)^2, & \text{if } \frac{L_\xi}{2} - \delta \le \xi \le \frac{L_\xi}{2} \\ W_0(\xi + \frac{L_\xi}{2} - \delta)^2, & \text{if } -\frac{L_\xi}{2} \le \xi \le -\frac{L_\xi}{2} + \delta \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

where $W_0$ represents the amplitude of the CAP and $\delta$ is the width of the CAP region, in which waves are absorbed. The complex potential $\imath W(\xi)$ appears as an additional subtrahend in $B(\chi, \xi, t)$ within (1) [22]. As shown in [10], the CAP has the added benefit of stabilizing the proposed scheme by pushing the eigenvalues of the system matrix further to the

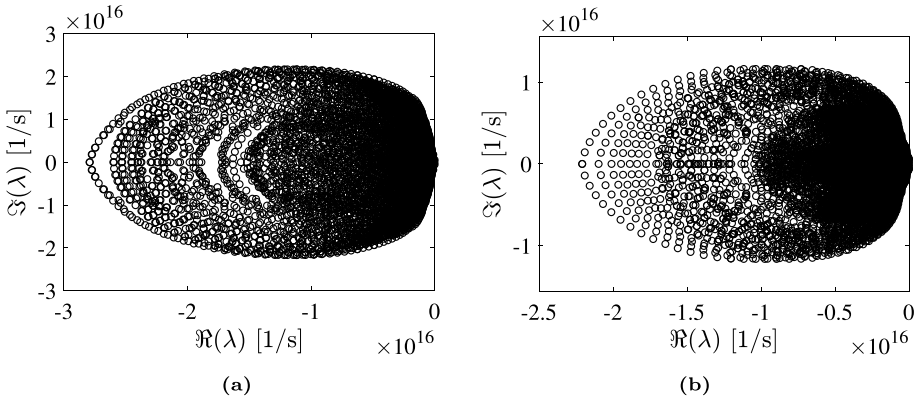


**Fig. 2** Eigenvalue spectrum of the system matrix derived with the DGFEM (a), and with the DGDG (b) algorithm, respectively

negative half of the complex plane, which is important to fulfill the stability condition for transient simulations [8].

## 5 Numerical results

To evaluate the algorithms, this chapter shall present and discuss the numerical results derived from both schemes. The stationary results from both schemes shall also be compared to the results obtained from the DGFV approach [11], as well as the quantum transmitting boundary method (QTBM), which is based on the numerical solution of the Schrodinger equation utilizing the finite element method. With its inherent inclusion of self-energy terms at the boundaries, it also takes into account the spatially open device characteristic appropriately and is, therefore, considered state of the art [23]. To apply the algorithms for a transient analysis, stability must be guaranteed. Here, the Runge–Kutta 4$^{\text{th}}$-order algorithm is chosen as it has proved an efficient performance in the past [12].

Stability is achieved when the real part of all eigenvalues derived from the system matrix is located on the negative half of the complex plane [24].

It can be concluded from Fig. 2a and b that the DGFEM scheme, as well as the DGDG algorithm, delivers an eigenvalue spectrum to allow a stable algorithm. To compare all three discretization schemes—that is, the DGFV, DGFEM, and DGDG methods—an identical simulation setup has been executed. The basis of the simulation is a GaAs/AlGaAs resonant tunneling diode with a flatband potential as indicated in Fig. 3. The barrier is set to 0.2eV. Additionally, the effective mass distribution for the steady-state case is assumed to be constant throughout the structure. Each simulation was executed utilizing the LiDO3 supercomputer located at the Technical University of Dortmund. The node is powered by two AMD EPYC 7542 CPUs at 2.49 GHz with a L3 cache of 128 MB, respectively. Both CPUs combined have 64 cores, while the RAM is 1024 GB. To maximize utilization of the one TB of RAM, the $\chi$-domain was discretized

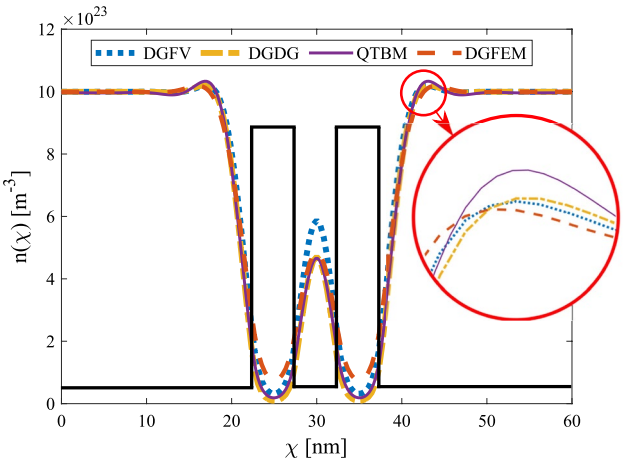


**Fig. 3** Comparison of the carrier density $n(\chi)$ between the DGFV, DGDG, DGFEM, and QTBM

**Table 1** Comparison of the carrier density $n(\chi)$ at three different $\chi$ -coordinates for the DGFV, DGDG, DGFEM, and QTBM algorithms as snapshots from Fig. 3. The units are m$^{-3}$

| $\chi$[nm] | DGFV | DGDG | DGFEM | QTBM |
|---|---|---|---|---|
| 30 | $5.87 \cdot 10^{23}$ | $4.70 \cdot 10^{23}$ | $4.85 \cdot 10^{23}$ | $4.65 \cdot 10^{23}$ |
| 35 | $3.25 \cdot 10^{22}$ | $9.94 \cdot 10^{21}$ | $7.63 \cdot 10^{22}$ | $2.01 \cdot 10^{22}$ |
| 43 | $1.02 \cdot 10^{24}$ | $1.02 \cdot 10^{24}$ | $1.02 \cdot 10^{24}$ | $1.03 \cdot 10^{24}$ |

with $N_\chi = 74$ elements, while for the $\xi$-domain $N_\xi = 340$ elements were used. Figure 3 displays the carrier density $n(\chi)$ derived with the three schemes as described in the caption. To further compare the accuracy of the three schemes, the results obtained with the state-of-the-art QTBM [23] have been included.

When observing the critical points in Fig. 3, which are the locations before and after the barriers, as well as the GaAs layer in between the barriers, the DGDG algorithm provides the most accurate result. To compare the critical points in

Fig. 3 more precisely, the carrier distribution values $n(\chi)$ are listed in Table 1.

As it can be observed from Fig. 3 and Table 1, the largest difference between the algorithms is in the center of the RTD at 30 nm. Here, the DGFV scheme leads to the highest value. At 35 nm, the DGFEM algorithm maintains a higher value compared to the other schemes. At the locations, i.e., 30nm and 43nm (see Table 1), all algorithms maintain a similar value, although being lower compared to the QTBM. In general, however, the DGDG method provides the closest carrier distribution levels to the QTBM algorithm.

To further confirm the operability of the proposed algorithms, the current density dependent on the voltage from 0V to 0.45V is shown in Fig. 4 for the DGFV, DGFEM, and DGDG algorithms. As a reference, the QTBM was also applied. However, due to a more precise approximation of the potential with the DGFEM and DGDG algorithms, the QTBM might not be the best choice as a basis for comparison. Nevertheless, to maintain consistency with Fig 3, the QTBM was included.

As is visible from Fig. 4, the DGDG and DGFEM algorithms show small deviations from each other. All three schemes display the typical progression of the IV-curve for an RTD with one resonance point within the 0.1V and 0.15V range. Only the DGFV method deviates from the two other algorithms and the reference after the first resonance point. This can be justified by the inaccuracy of the DGFV algorithm compared to both other approaches and the QTBM. Furthermore, the resonance achieves a higher value with the three DG-based algorithms when compared to the QTBM. A possibility for the deviations is that the DGDG and DGFEM algorithm approximate the spatial behavior of $B(\chi, \xi, t)$ in (1) more precisely due to the Gauss–Lobatto quadrature. It is also important to note that for the above simulations of the IV-curve an equally sized discretizing grid is used for all algorithms. Therefore, the discrepancy of the DGFV algorithm is reasonably consistent with the findings from Fig. 3.

There were not only discrepancies in the results of the three algorithms but equally so with regard to the computation time as well as to the utilized resources. To compare the RAM usage and run time of the three algorithms, four different simulation setups have been carried out to analyze the trend when varying the number of $\xi$-elements $N_\xi$. Hereby, the number of elements was increased starting from $N_\xi = 280$ to $N_\xi = 680$.

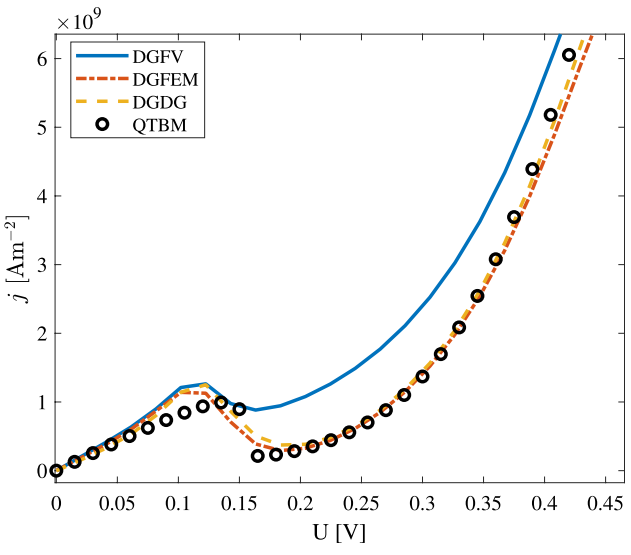


**Fig. 4** Characteristic IV-curves for a RTD derived with the DGFV, DGDG, and DGFEM. A reference curve was added derived with the QTBM to compare accuracy of the three algorithms

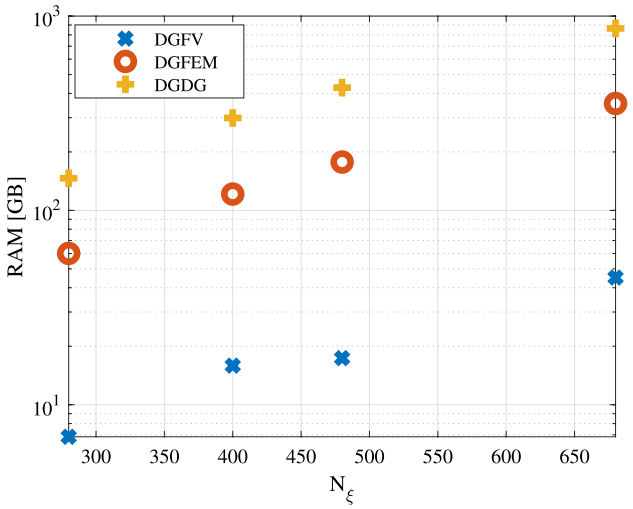


**Fig. 5** RAM usage of the DGFV, DGDG, and DGFEM concepts for varying numbers of nodes in the $\xi$-domain

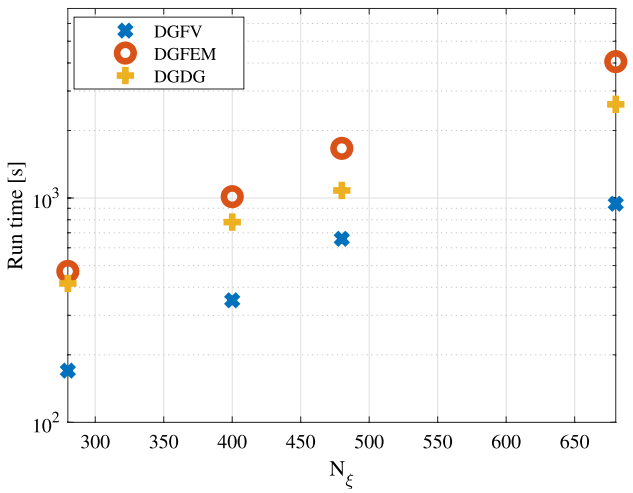


**Fig. 6** Run time of the DGFV, DGDG, and DGFEM algorithms for varying numbers of nodes in the $\xi$-domain

The results obtained are shown in Figs. 5 and 6. As it can be seen, all algorithms describe an approximately linear increase in RAM usage and computation time when refining the grid. Furthermore, the DGDG algorithm requires more RAM compared to the DGFEM scheme, while needing less time than the DGFEM method to complete the calculation. The DGFV algorithm is the most efficient approach in both cases. This is due to the fact that eigenvalues and eigenvectors are being derived analytically, while for the DGFEM and DGDG algorithm, the eigenvalues and eigenvectors are obtained from the discretization matrix as introduced in Sect. 3. Nevertheless, since $\boldsymbol{S}^{'}$ and $\boldsymbol{S}^{''}$ are Toeplitz matrices, the eigenvalues and eigenvectors can be determined analytically providing room for optimization. Additionally, the $\xi$-coordinate of the matrix containing the potential values of the DGFV scheme is approximated with the trapezoidal rule as described in [13]. With the DGFEM and DGDG algorithm, the same drift matrix is derived with the Gauss–Lobatto scheme as described in Sect. 3, which is computationally more expensive. However, this affects only the steady-state case, where the distribution matrix is computed only once.

Another major opportunity for optimization is leaving a one-script implementation behind and implementing a functional programming (FP) [25] approach. In doing so, one can exploit the advantage that matrices not needed for further calculations are removed from the workspace clearing up RAM. Accordingly, the three respective scripts for the DGDG, DGFV, and DGFEM algorithms were divided into individual functions and combined into one main script. To compare both programming styles, the simulation with the most refined discretizing grid from Figs. 5 and 6 has been integrated into the new script. The new computation times and RAM usage for the three algorithms are summarized in Table 2.

When comparing the two programming styles in Table 2, it is most noticeable that the run time across the board increases, while the RAM usage decreases. The run time for the DGFV and DGDG algorithm is almost similar. Although the DGFV scheme remains the fastest algorithm, one has to take into account the lower accuracy of the DGFV method compared to the DGFEM and DGDG algorithms. Hence, a

**Table 2** Comparison of the two programming methods for the DGFV, DGFEM, and DGDG algorithms

| | Run time [s] | | RAM usage [GB] | |
|---|---|---|---|---|
| | Single script | FP | Single script | FP |
| DGFV | 532 | 1545 | 45.13 | 37.77 |
| DGDG | 1003 | 1600 | 863.67 | 46.95 |
| DGFEM | 920 | 2361 | 354.98 | 42.02 |

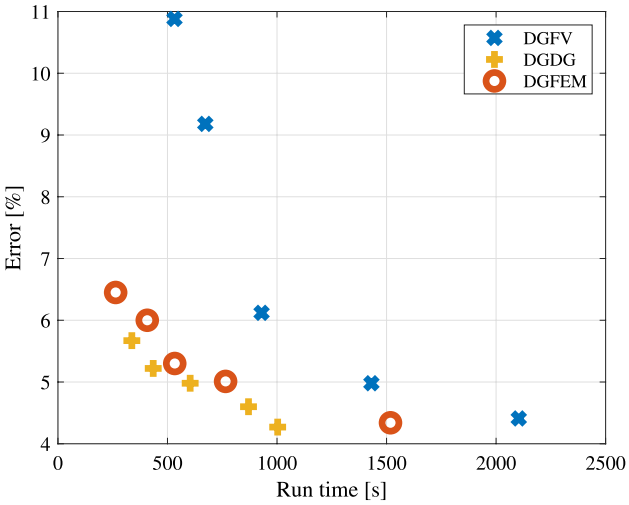


**Fig. 7** Comparison of computation time until an equal error value is reached between the DGFV, DGDG, and DGFEM concepts. The run times were determined utilizing the FP script

comparison from the perspective of an equal error between all algorithms is essential. For this purpose, the discretizing grid from Fig. 3 for the DGFV algorithm was steadily increased until said algorithm achieved the same mean error as the DGDG and DGFEM algorithms. In Fig. 7, the run time is depicted against the mean error until the same value is achieved among the three algorithms. The error was calculated according to

$$\epsilon = \frac{|n_{QTBM} - n_{DG**}|}{|n_{QTBM}|},$$

whereby $n_{DG**}$ represents the result obtained either with the DGFV, DGFEM, or DGDG scheme.

From Fig. 7, it can be concluded that the DGFV algorithm becomes computationally expensive when aiming to achieve the same error as the DGDG algorithm. The DGDG method is computationally more efficient regarding simulation time, while maintaining the same error value.

Finally, it is of larger interest to analyze the computation time for a transient simulation. Instead of utilizing the parameters from the simulations presented before, a smaller sample size has been chosen due to time and hardware limitations. The number of $\chi$-elements has been set to $N_\chi = 40$, while the number of $\xi$-elements has been reduced to $N_\xi = 280$. The order of the approximating functions was kept the same. For the discretization of the time domain, the Runge–Kutta $4^{th}$-order algorithm was chosen. The time stepping width $\Delta t$ was set to $\Delta t = 10^{-17}$s. To compare the computation time for one time step of the transient loop, each algorithm ran for a total of 100 time steps, of which the mean value was calculated. With 0.25 seconds on average, the DGFV algorithm runs the longest. Followed by the

DGDG with 0.24 seconds on average, and lastly the DGFEM with a mean value of 0.21 seconds being the fastest.

It can be concluded that the DGFEM algorithm is almost 13% faster than the DGDG scheme. An analysis of the system matrix for all algorithms revealed that the system matrix of the DGDG and DGFV scheme has 1.44% more elements when compared to the system matrix obtained from the DGFEM method. Therefore, the increased computation times of the DGDG and DGFV schemes are plausible.

Lastly, it remains to be analyzed why there is a major discrepancy in RAM usage between the algorithms. Multiple reasons affect this condition:

- The DGDG algorithm introduces block-diagonal matrices not only in the $\chi$- but also in the $\xi$-domain, resulting in matrices of a higher order.
- To derive the drift operator $\boldsymbol{G}_{jm}^{k}$ in (12), both DGDG and DGFEM algorithms utilize the Gauss–Lobatto quadrature in both domains $\Omega_{\chi}$ and $\Omega_{\xi}$ increasing the order of the involved matrices considerably. The DGFV scheme only uses the Gauss–Lobatto quadrature for the $\chi$-discretization.
- Specifically the DGDG algorithm works with matrices consisting of more nonzero elements when compared to the DGFEM scheme since the DG scheme generally does not overlap elements like the FEM does.

Still, there is one decisive advantage of the DGDG procedure over the others. Because the DG scheme produces block-diagonal matrices in both domains, in theory, the system matrix should be more 'flexible.' That is, a block-diagonal matrix can be subdivided into smaller matrices, inverted, and multiplied in parallel [26].

## 6 Conclusion

This work has developed and compared the DGFEM and DGDG algorithms to the current DGFV method by simulating a double-barrier resonant tunneling diode and analyzing the carrier density within the structure for the steady-state case. Functionality of the algorithms has further been shown by calculating the characteristic IV-curve for the aforementioned device. The comparison has shown the DGDG scheme to deliver the most accurate compliance with the QTBM reference algorithm, while maintaining the highest computational efficiency. Furthermore, only the DGDG algorithm is capable of exploiting the full potential of high-performance computing. By introducing a discretization with the DG method in both directions, the system matrix can be theoretically transformed to be globally block diagonal. This fact enables the utilization of parallelization during the inversion of the system matrix and matrix–vector multiplication. It will speed up the computation time not only for the steady-state case but also for the transient case as well. Here, a considerable time saving in each time step can be expected.

**Author Contributions** These authors contributed equally to this work.

**Funding** Open Access funding enabled and organized by Projekt DEAL. This work was supported by the Deutsche Forschungsgemeinschaft DFG under Grant SCHU 1016/10-1.

**Data availability** No datasets were generated or analyzed during the current study.

## Declarations

**Conflict of interest** The authors declare no conflict of interest.



## References

1. Khalid, K.S., Schulz, L., Schulz, D.: Self-energy concept for the numerical solution of the liouville-von neumann equation. IEEE Trans. Nanotechnol. **16**(6), 1053–1061 (2017). https://doi.org/10.1109/TNANO.2017.2747622
2. Fattebert, J.-L., Nardelli, M.B.: Finite difference methods for ab initio electronic structure and quantum transport calculations of nanostructures. Handb. Numer. Anal. **10**, 571–612 (2003). https://doi.org/10.1016/S1570-8659(03)10009-9
3. Wang, J.S., Agarwalla, B.K., Li, H., et al.: Nonequilibrium Green?s function method for quantum thermal transport. Front. Phys. **9**, 673–697 (2014). https://doi.org/10.1007/s11467-013-0340-x
4. Cockburn, B.: Discontinuous Galerkin methods for computational Fluid Dynamics, pp. 1–63. John Wiley And Sons, Ltd, New Jersey (2017). https://doi.org/10.1002/9781119176817.ecm2053
5. Tokman, M., Loffeld, J.: Efficient design of exponential-Krylov integrators for large scale computing. Procedia Comput. Sci. **1**(1), 229–237 (2010). https://doi.org/10.1016/j.procs.2010.04.026
6. Schulz, L., Inci, B., Pech, M., Schulz, D.: Subdomain-based exponential integrators for quantum Liouville-type equations. J. Comput. Electron. **20**(6), 2070–2090 (2021). https://doi.org/10.1007/s10825-021-01797-2
7. Cockburn, B., Shu, C.W.: Runge-kutta discontinuous galerkin methods for convection-dominated problems. J. Sci. Comput. **16**, 173–261 (2001). https://doi.org/10.1023/A:1012873910884

8. Musa, H., Saidu, I., Waziri, M.: A simplified derivation and analysis of fourth order runge kutta method. Int. J. Comput. Appl. (2010). https://doi.org/10.5120/1402-1891
9. Jensen, K.L., Buot, F.A.: Numerical simulation of transient response and resonant-tunneling characteristics of double-barrier semiconductor structures as a function of experimental parameter. Naval Res. Lab., (1989)
10. Ganiu, V., Schulz, D.: Application of discontinuous galerkin methods onto quantum-liouville type equations. IWCN Book of Abstracts, (2023)
11. Ganiu, V., Schulz, D.: Hybrid discontinuous galerkin approach for the solution of quantum liouville-type equations. IEEE Trans. Nanotech. **22**, 696–705 (2023). https://doi.org/10.1109/TNANO.2023.3322541
12. Ganiu, V., Schulz, D.: Discontinuous Galerkin concept for Quantum-Liouville type equations. Solid-State Electron. **200**, 108536 (2023). https://doi.org/10.1016/j.sse.2022.108536
13. Schulz, L., Schulz, D.: Formulation of a phase space exponential operator for the Wigner transport equation accounting for the spatial variation of the effective mass. J. Comput. Electron. **15**(5), 801–809 (2016). https://doi.org/10.1007/s10825-020-01551-0
14. Frensley, W.R.: Boundary conditions for open quantum systems driven far from equilibrium. Rev. Mod. Phys. **62**(3), 745–791 (1990). https://doi.org/10.1103/RevModPhys.62.745
15. Bland, M., Ct, J., Staniforth, A.: The accuracy of a finite-element vertical discretization scheme for primitive equation models: comparison with a finite-difference scheme. Mon. Wea. Rev. **111**(12), 2298–2318 (1983). https://doi.org/10.1175/1520-0493(1983)111
16. Tokhi, M.O., Mohamed, Z., Azad, A.K.M.: Finite difference and finite element approaches to dynamic modelling of a flexible manipulator. Proc. Inst. Mech. Eng. Part I J. Syst. Control Eng. **211**(2), 145–156 (1997). https://doi.org/10.1243/0959651971539966
17. Eslahchi, M.R., Masjed-Jamei, M., Babolian, E.: On numerical improvement of Gauss?Lobatto quadrature rules. Appl. Math. Comput. **164**, 707–717 (2005). https://doi.org/10.1016/j.amc.2004.04.113
18. Weinbub, J., Ferry, D.: Recent advances in wigner function approaches. Appl. Phys. Rev. **5**(4), 041104 (2018). https://doi.org/10.1063/1.5046663
19. Schulz, L., Schulz, D.: Complex absorbing potential formalism accounting for open boundary conditions within the wigner transport equation. IEEE Trans. Nanotechnol. **18**, 830–838 (2019). https://doi.org/10.1109/TNANO.2019.2933307
20. Ganiu, V., Schulz, D.: Application of a Hybrid Discontinuous Galerkin Scheme onto Quantum-Liouville-type Equations for Heterostructure Devices. In: International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), Kobe, Japan, pp. 261–264, (2023). https://doi.org/10.23919/SISPAD57422.2023.10319502.
21. Kosik, R., Cervenka, J., Kosina, H.: Numerical constraints and non-spatial open boundary conditions for the Wigner equation. J. Comput. Electron. **20**, 2052–2061 (2021). https://doi.org/10.1007/s10825-021-01800-w
22. Schulz, L., Schulz, D.: Numerical analysis of the transient behavior of the non-equilibrium quantum liouville equation. IEEE Trans. Nanotech. **17**(6), 1197–1205 (2018). https://doi.org/10.1109/TNANO.2018.2868972
23. Lent, C.S., Kirkner, D.J.: The quantum transmitting boundary method. J. Appl. Phys. **67**, 6454 (1990). https://doi.org/10.1063/1.345156
24. Fu, G., Shu, C.-W.: Optimal energy-conserving discontinuous Galerkin methods for linear symmetric hyperbolic systems. J. Comput. Phys. **394**, 329–363 (2019). https://doi.org/10.1016/j.jcp.2019.05.050
25. Chambers, J.M.: Object-oriented programming, functional programming and R. Stat. Sci. **29**(2), 167–180 (2014). https://doi.org/10.1214/13-STS452
26. Navarro, C., Hitschfeld, N., Mateu, L.: A survey on parallel computing and its applications in data-parallel problems using GPU architectures. Commun. Comput. Phys. **15**, 285–329 (2013). https://doi.org/10.4208/cicp.110113.010813a