M&C 2017 - International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering, Jeju, Korea, April 16-20, 2017, on USB (2017)

QZ-Decomposition For Matrix-Free Sweep With High Order DG-FEM

Sebastián González-Pintor*, Anders Ålund†, Christophe Demazière‡

* *Department of Mathematical Sciences, Chalmers University of Technology, Sweden*
† *Fraunhofer Chalmers Research Centre for Industrial Mathematics, Sweden*
‡ *Department of Physics, Chalmers University of Technology, Sweden*
sebastian.gonzalez-pintor@chalmers.se, anders.alund@fcc.chalmers.se, demaz@chalmers.se

Abstract - *This work deals with the acceleration of the transport sweep for a discrete ordinates formulation of the neutron transport equation. In particular, we focus on accelerating a matrix-free version of the algorithm. The algorithmic complexity of a naive version of the transport sweep is studied, and a preprocessing technique based on a QZ-decomposion of the matrices composing the sweep is proposed. This technique is not purely matrix free, but we will see that the memory requirements for this approach is negligible in comparison with a full inversion of the streaming operator. The proposed technique is tailored for high order Finite Element Method on regular Cartesian meshes, where the local problems are dense matrices of small size, but with minor modification it can be extended to regular triangular meshes and for non-matching grids with isotropic refinements. The extension to more general geometries requires simultaneous triangularization of more than two matrices at once. The performance of the proposed acceleration is analysed in algorithmic complexity with respect to the number of iterations, and then it is used to predict the required CPU-time to solve the problem. These predictions are tested with at 2D pin-by-pin problem consisting of four assemblies, and the predictions show good agreement with the measured CPU-time for a range of polynomial degrees.*

I. INTRODUCTION

The neutron transport equation is a detailed balance in phase space, where neutrons can leave a volume in phase space by destruction, out-scattering and streaming. Mathematically the problem of modeling this balance may be written as [1]

$$\mathcal{L}\Psi = \mathcal{S}\Psi + \frac{1}{\lambda}\mathcal{F}\Psi \tag{1}$$

where the angular neutron flux, $\Psi(\mathbf{r}, \Omega, E)$, depends on the following independent variables: E for energy, Ω for direction of travel and $\vec{r}$ for the position. λ is the multiplication factor of the system. The angular discretization of the transport equation (1) chosen in this work is the Discrete Ordinates method (S_N), where one of the main features is the possibility to invert the streaming operator by means of so-called transport sweeps, as will be explained later. After discretization in energy and angle, the set of balance equations reads as follows

$$\mathcal{L}_{g,n}\Psi_{g,n} = \mathcal{M}_n \sum_{g'=1}^{G} \mathcal{S}_{g,g'}\mathcal{D}\Psi_{g'} + \frac{1}{\lambda}\mathcal{M}_n\chi_g \sum_{g'=1}^{G} \mathcal{F}_{g'}\phi_{g'}, \quad g = 1, \ldots, G;\ n = 1, \ldots, N', \tag{2}$$

where the angular flux and the scalar flux are defined as

$$\Psi_{g,n}(\mathbf{r}) = \Psi_g(\mathbf{r}, \Omega_n), \quad \text{and} \quad \phi_g := \mathcal{D}_0\Psi_g = \sum_{n=1}^{N'} \omega_n \Psi_{g,n}, \tag{3}$$

respectively, where $\{\omega_n, \Omega_n\}_{n=1}^{N'}$ is the discrete quadrature used, $g = 1, \ldots, G$ denotes the energy group, and the operators are defined as follows: $\mathcal{L}_{g,n}$ considers the streaming operator at group g with streaming direction Ω_n together with the total collision term at group g; $\mathcal{S}_{g,g'}$ is the scattering operator from group g' to g; $\mathcal{D}$ and $\mathcal{M}_n$ are the direction-to-moment and moment-to-direction operators, respectively, used for the evaluation of the anisotropic scattering; χ_g is the fission spectrum for group g; and $\mathcal{F}_{g'}$ is the fission operator for group g'.

In order to discretize the spatial variable, a discontinuous Galerkin Finite Element Method was used. Many implementations exist to approximate the S_N equations with discontinuous Galerkin methods, especially considering that this method was first implemented for this particular equation [2, 3]. This method has become one of the most robust ones in order to perform high fidelity approximations with the use of modern computer architectures. From the most recent implementations, we can highlight the one performed in [4], where the convergence of the linear discontinuous Galerkin method on conforming structured meshes was studied for the two-dimensional C5G7 problem [5] with heavy use of parallelization. Another effort worth mentioning is the one performed in [6] to approximate the same problem with continuous Finite Element Method of linear and quadratic elements, achieving very similar results to the work in [4].

The present document is structured as follows: The spatial discretization with a high order Finite Element Method is described in Section II, together with the system of equations depending on one parameter. This is the local problem, that has to be solved many times for different values of the parameter. This is the key point. Then, the algorithmic complexity of a naive version of the transport sweep is studied, and a preprocessing technique based on a QZ-decomposion of the matrices composing the sweep is proposed and analysed in Section III. In Section IV we use the C4 benchmark in order to compare the expected performance of the proposed acceleration in algorithmic complexity, with respect to the required CPU-time to solve the problem. The conclusions are summarized in Section V, together with possible extensions

and future work.

II. HIGH ORDER DG-FEM DISCRETIZATION

A source problem with the streaming term is used as the model problem to show the implementation of the high order Discontinuous Galerkin Finite Element Method for the transport equation. Here we drop the indices for energy and direction in order to simplify the notation. Let us consider the strong form of the transport equation as follows

$$\Omega \cdot \nabla\Psi + \Sigma\Psi = q, \tag{4}$$

where Ω is a given direction, and the equation is defined on a domain V, with in-flow boundary Γ_-. Fixed incoming flux boundary conditions, $\Psi(\Omega) = g(\Omega)$ in Γ_-, are assumed to simplify the notation.

The first step is to rewrite the problem defined by Eq. (4) in weak form as follows

$$(v, \Omega \cdot \nabla\Psi)_V + (v, \Sigma\Psi)_V = (v, q)_V\,, \tag{5}$$

where the inner product is defined as $(u, v)_V = \int_V uv\mathrm{d}\vec{r}$.

Then we set a partition, $\mathcal{T}_h = \{T\}$, covering the domain V, and split the previous integral in a sum of integrals over each subdomain T (where the cross-sections are assumed constant) as follows

$$\sum_{T\in\mathcal{T}_h} (v, \Omega \cdot \nabla\Psi)_T + \sum_{T\in\mathcal{T}_h} \Sigma_T\,(v, \Psi)_T = \sum_{T\in\mathcal{T}_h} (v, q)_T\,. \tag{6}$$

We then integrate by parts the first term of the equation, and using the boundary conditions we obtain

$$\begin{aligned} -\sum_{T\in\mathcal{T}_h} (\Psi, \Omega \cdot \nabla v)_T + \sum_{e\in\mathcal{E}_h} \langle \Psi^-, \Omega \cdot [\vec{n}v] \rangle_e \\ + \langle \Psi, \Omega \cdot \vec{n}v \rangle_{\Gamma_+} + \sum_{T\in\mathcal{T}_h} \Sigma_T\,(v, \Psi)_T \\ = \sum_{T\in\mathcal{T}_h} (v, q)_T + \langle g, \Omega \cdot \vec{n}v \rangle_{\Gamma_-}\,, \end{aligned} \tag{7}$$

where $\mathcal{E}_h = \{e\}$ is the set of internal edges of the triangulation $\mathcal{T}_h$, Γ_+ is the out-flow boundary, and the integral over the edges e is defined by $\langle u, v \rangle_e = \int_e uv\mathrm{ds}$. The jump $[\cdot]$ is defined by

$$[\vec{n}v] = \vec{n}^+ v^+ + \vec{n}^- v^-, \tag{8}$$

where the superscripts refer to the upwind (+) and downwind (−) values at the edge.

1. The Local Problem

In order to perform a matrix-free transport sweep, the first step is to sort the elements for a fixed direction Ω in such a way that the flux at the incoming boundary is known when solving for a particular element. In particular, we focus on the approximation of a single subdomain during the transport sweep. The discretization of the neutron transport equation for a fixed direction Ω in an element T is as follows

$$\begin{aligned} (\Psi, \Omega \cdot \nabla v)_T + \langle \Psi, \Omega \cdot \vec{n}v \rangle_{\partial_+ T} + \Sigma_T\,(v, \Psi)_T \\ = (v, q)_T + \langle \Psi^-, \Omega \cdot \vec{n}v \rangle_{\partial_- T}\,, \end{aligned} \tag{9}$$

where $\partial_- T$ and $\partial_+ T$ represent the in-flow and out-flow boundaries of T, respectively. The only requirement to solve Eq. (9) is that the incoming flux Ψ^- is known.

We notice that up to now no assumption has been made for the shape of the element T, so that it can be of triangular or quadrilateral shape. For the numerical experiments, instead, quadrilateral elements will be used, but similar results are expected for triangular elements.

The solution of problem (9) has to be performed for each source iteration, each energy group, each direction, and each element over the mesh, and thus the optimization of this operation would have an important impact on the final CPU-time required to solve the problem. For instance, the local matrices can be precalculated [7] in order to minimize the time for this local operation. Usually, the DG-FEM method for the transport problem will not have the required regularity in order to justify the use of high order FEM, but it has been reported that increasing the order of the approximation improves the convergence rate for the DG-FEM [8] by reducing the constant that multiplies the term of the order of convergence, i.e, reducing the C in Ch^α. Nevertheless, for high-order FEM, the inversion of the local matrices are increasingly expensive for two- or three-dimensional problems when increasing the polynomial order. Here we propose an alternative approach based on a reduction on the complexity of the local solver for the sweep, while keeping low memory requirements, provided that we restrict ourselves to structured geometries.

Thus, considering that all elements have the same shape and size, equation (9) can be re-written in matrix notation as follows

$$(G_w + E^o_w + s_{g,e}M)u_{i,g,w,e} = q_{i,g,w,e} - E^i_w u^i_{i,g,w,e} \tag{10}$$

where the indices i, g, w and e, respectively, correspond to the iteration, the energy group, the angular direction and the physical element. A similar break down of the cost of the local problem has been done in [9], where also more spatial discretizations schemes are considered. While the mass matrix M does not depend on anything, the local stream matrix G_w depends on the angular direction. The edge matrices E^o_w, E^i_w are for the out-flow and in-flow boundaries, the vectors u and q are the solution and the source, which are different for every index, and $s_{g,e}$ represents the cross section for a particular element and energy group. Then the local problem for $u_{i,g,w,e}$ can be written as

$$(G^o_w + s_{g,e}M)u_{i,g,w,e} = S_{i,g,w,e} \tag{11}$$

where we combine the streaming matrix together with the edge contributions as

$$G^o_w := G_w + E^o_w, \tag{12}$$

and the source term, considering also the incoming flux, is renamed as

$$S_{i,g,w,e} := q_{i,g,w,e} - E^i_w u^i_{i,g,w,e} \tag{13}$$

(not to be mistaken with the scatering, which is included in the $q_{i,g,w,e}$) for later convenience of the notation (Q will be

the unitary matrix in the QZ-decomposition). Equation (11) has the property that, for a given angular direction, the local matrix, $G^o_w + s_{g,e}M$, depends linearly on just one parameter, $s_{g,e}$, for the different groups and elements, and we are going to take advantage of this structure to simplify the solution of this system after a preprocessing step.

III. QZ-DECOMPOSITION FOR THE LOCAL PROBLEM IN THE TRANSPORT SWEEP

The time required to perform the transport sweep is one of the dominant factors of the total time required for solving the previous equation, being more relevant when the number of directions is higher. Thus, many efforts focused on reducing the number of sweeps that are required for the solution by using different acceleration techniques [10], such as the Diffusion Synthetic Acceleration (DSA) or the Non-linear Krylov Acceleration (NKA), together with strategies for parallelizing the sweeps. The combination of these accelerations generally makes the discrete ordinates method a competitive alternative to approximate the neutron transport equation.

In the following, different algorithms to apply the transport sweep are shown. We start with the standard approach, i.e., to build and solve the local systems every time it is required, and then we study different alternatives. Special attention is put on the memory required for the alternatives and the expected speed-up of the calculations (based on the number of operations needed for each algorithm).

1. On-the-fly Calculation

The on-the-fly form of the transport sweep can be written as in Algorithm 1. It is worth to notice that in Algorithm 1

Algorithm 1 On-the-fly

1: **function** TRANSPORT SWEEP
2: **for** $i < N_i$ **do**
3: **for** $g < N_g$ **do**
4: **for** $w < N_w$ **do**
5: **for** $e < N_e$ **do**
6: $S := q_{i,g,w,e} - E^i_w u^i_{i,g,w,e}$ ▷ $O(n^2)$
7: $A := G^o_w + s_{g,e}M$ ▷ $O(n^2)$
8: Solve $Au_{i,g,w,e} = S$ ▷ $O(n^3)$

we solve the local systems in line 8 with a LU-decomposition, but any other algorithm, such as Gaussian elimination, will be $O(n^3)$. Regarding the memory required for this algorithm, we observe that we use the temporal vector S to store the right hand side, and the temporal matrix A to store the local matrix, thus making the amount of required memory negligible.

The number of iterations (N_i) is defined as the number of source iterations (N_s) times the number of full sweeps (N_{fs}) needed for every source iteration ($N_i = N_s \times N_{fs}$). N_s can be reduced by using a better algorithm for the eigenvalue problem (Jacobian Free Newton Krylov, Non-linear Krylov Acceleration, Anderson Mixing, ...), while N_{fs} can be reduced by preconditioning the source iteration (DSA, Multigrid, ...). Here we focus on improving the computational effort necessary for the innermost loop, thus reducing the CPU-time for each sweep, so that this can be combined with the accelerations mentioned previously which are focused on reducing the number of sweeps.

2. Precompute Local Inverses

In order to accelerate the sweep, the simplest approach would be to precalculate the inverses of the local matrices, with a preprocessing step as costly as performing a full sweep for all the different groups. This will reduce the effort of applying the sweep for each iteration to $O(n^2)$, as we can see in Algorithm 2. While Algorithm 2 looks all right, it requires

Algorithm 2 Precompute Local Inverses

function LU-PREPROCESSING
for $g < N_g$ **do**
for $w < N_w$ **do**
for $e < N_e$ **do**
$A := G^o_w + s_{g,e}M$ ▷ $O(n^2)$
$[L_{g,w,e}, U_{g,w,e}] := lu(A)$ ▷ $O(n^3)$
function TRANSPORT SWEEP
for $i < N_i$ **do**
for $g < N_g$ **do**
for $w < N_w$ **do**
for $e < N_e$ **do**
$S := q_{i,g,w,e} - E^i_w u^i_{i,g,w,e}$ ▷ $O(n^2)$
Solve $L_{g,w,e}v = S$ ▷ $O(n^2)$
Solve $U_{g,w,e}u_{i,g,w,e} = v$ ▷ $O(n^2)$

to store all the local inverses, which in general is prohibitive because of the high memory requirements.

3. QZ preprocessing

In order to reduce the memory requirements of precomputing the inverses, we should restrict ourselves to precompute and store fewer matrices. In this direction, we suggest a QZ-decomposition [11] of the streaming matrix and the mass matrix. Because these matrices do not depend on the cross sections, meaning that they are the same for all energy groups and for all elements, the number of precomputations is highly reduced. In particular, the QZ-decomposition of a pair $\{G^o_w, M\}$ will generate four matrices as follows

$$[\hat{G}_w, \hat{M}_w, Z_w, Q_w] = qz(G^o_w, M) \tag{14}$$

such that

$$\hat{G}_w = Z_w G^o_w Q_w, \quad \hat{M}_w = Z_w M Q_w, \quad Q^T_w Q_w = I, \quad Z^T_w Z_w = I,$$

where $\hat{G}_w$, and $\hat{M}_w$ are triangular and quasi-upper triangular matrices. Then the local system (11) becomes

$$(\hat{G}_w + s_{e,g}\hat{M}_w)Q^T_w u_{i,g,w,e} = Z_w S_{i,g,w,e},$$

to be solved for $u_{i,g,w,e}$. This procedure applied to the transport sweep is shown in Algorithm 3.

There are many ways of solving the system in line 12 of Algorithm 3. One option is to perform a LU decomposition

Algorithm 3 QZ preprocessing

function QZ-PREPROCESSING
  for $w < N_w$ **do**
    $[\hat{G}_w, \hat{M}_w, Z_w, Q_w] = qz(G^o_w, M)$ ▷ $O(n^3)$
function TRANSPORT SWEEP
  for $i < N_i$ **do**
    for $g < N_g$ **do**
      for $w < N_w$ **do**
        for $e < N_e$ **do**
          $S := q_{i,g,w,e} - E^i_w u^i_{i,g,w,e}$ ▷ $O(n^2)$
          $b := Z_w S$ ▷ $O(n^2)$
          $A := \hat{G}_w + s_{g,e}\hat{M}_w$ ▷ $O(n^2)$
          Solve $Ax = b$ ▷ $O(n^2)$
          $u_{i,g,w,e} := Q^T_w x$ ▷ $O(n^2)$

of a Hessenberg system, which is $O(n^2)$, together with the backward solver of the L matrix (which is bi-diagonal due to the Hessenberg form of A), and the forward solver for the upper triangular matrix U, which is $O(n^2)$. Instead, we use the quasi-upper triangular form of A to perform a direct solve with LAPACK similar to a block-wise forward solver, with a maximum size of blocks equal to two. This reduces the complexity of the sweep from $O(n^3)$ to $O(n^2)$, while keeping low memory requirements. More information about solving quasi-upper triangular is provided in the Appendix.

A. Alternative preprocessings

The main idea is based on a generalized eigendecomposition. Consider the following problem of interest

$$(A + \sigma B)x = b,$$

with A and B symmetric dense matrices, which has to be solved for many different values of the parameter σ and different right hand sides b. We know that the eigendecomposition of the pencil (A, B) is

$$AX = BX\Lambda$$

where $X(:, i)$ is the i-th eigenvector, and Λ is a diagonal containing the real (because of the symmetry) eigenvalues, and

$$\Lambda = X^T AX, \quad I = X^T BX \quad \text{(notice } X^T X \neq I\text{)}$$

Then, for solving the linear systems, we rewrite it as

$$X^{-T}(\Lambda + \sigma I)X^{-1}x = b$$

and the solution x can be obtained by

$$y = (\Lambda + \sigma I)^{-1}X^T b; \quad x = Xx$$

where only products by the matrix X (and its transpose) and the solution of a diagonal system with $(\Lambda + \sigma I)$ are required. Here the preprocessing would be very efficient because of the simple inversion.

Because the matrices we have are not symmetric it can happen (and it actually happens) that some eigenvalues come in complex pairs, so one of the following two alternatives must be considered: a) deal with complex matrices, and be sure than the obtained solution to the system is a vector of real numbers; b) look for an alternative decomposition which involves a posterior inversion more expensive than for a diagonal matrix, but of the same order than the matrix vector product, i.e., $O(n^2)$ with n being the size of the matrices. Here we have investigated the second option with a QZ decomposition.

IV. RESULTS

We use LAPACK for the QZ decomposition, the quasi-upper triangular solver, the LU decompositions, the backward and forward triangular solvers, and the matrix-vector products, in order to obtain a fair comparison of the different methods due to different implementations. The calculations have been done on a single processor Intel i7-4770, but further analysis considering the cache sizes of this processor for different sizes of the data that each algorithm moves must be performed in order to obtain more accurate predictions of the performance. The study here is limited to a qualitative comparison between the different algorithms.

Algorithm 1 will be tested against Algorithm 3 for a simple problem, while Algorithm 2 is not considered because the memory required by this algorithm grows very fast with the size of the problem, making it impractical even for medium size problems.

1. 2D case: C4 problem

A. Problem description

The C4 problem is a semi-reflected MOX checker board. This two-dimensional problem with two energy groups, proposed in [12], consists of a 2×2 assemblies mini-core with 17×17 pins for assembly, for which the material distribution is shown in Figure 1.

It has been considered in [13] as a test for a group-wise refinement of the mesh with a Finite Element Method. Here we use the material configuration of the C4 problem to solve the neutron transport equation. The fluxes Φ_0 and Φ_1 obtained with a S_2 discrete ordinates configuration are shown in Fig. 2.

B. Prediction of the number of operations

The mesh consists of quadrilateral elements, so that in d-dimensions the size of the local matrices is defined by $n = (p + 1)^d$, where p is the polynomial degree for the approximation. If we consider that the number of operations for Algorithm 1 is

$$C^{\text{fly}}_{p,d} \approx \frac{2}{3}n^3 = \frac{2}{3}(p + 1)^{3d} = \frac{2}{3}(p + 1)^6, \tag{15}$$

and for Algorithm 3 is

$$C^{\text{qz}}_{p,d} \approx 6n^2 = 6(p + 1)^{2d} = 6(p + 1)^4, \tag{16}$$

then we can predict the number of operations required for different polynomial degrees, respectively, in 2-dimensional problems with quadrilateral elements, as shown in Fig. 3.

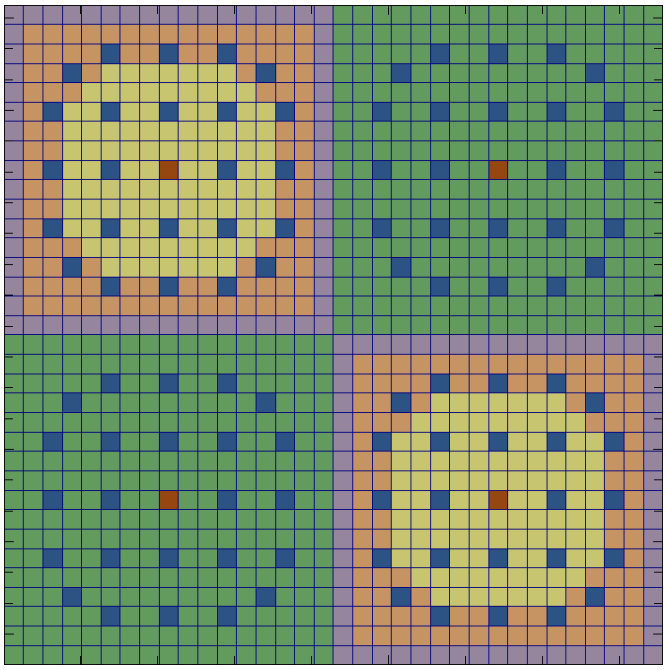


Fig. 1: C4 material composition [12]. Each color represents a given composition.

C. Measurements of CPU-time

If we measure the CPU-time needed for the different degrees, what also involves moving data through the different memories, we see that the improvement shows a similar behavior, although being more modest. This is shown in Fig. 4.

It is worth to notice here that the prediction fails mainly in the first part of the interval. This can be explained because the prediction has been made considering only the the highest order term for number of operations, and this higher order term will be dominant and representative of the asymptotic behaviour when large values of the matrix size are considered. Moreover, moving the data would also produce an extra cost that is not considered in our prediction.

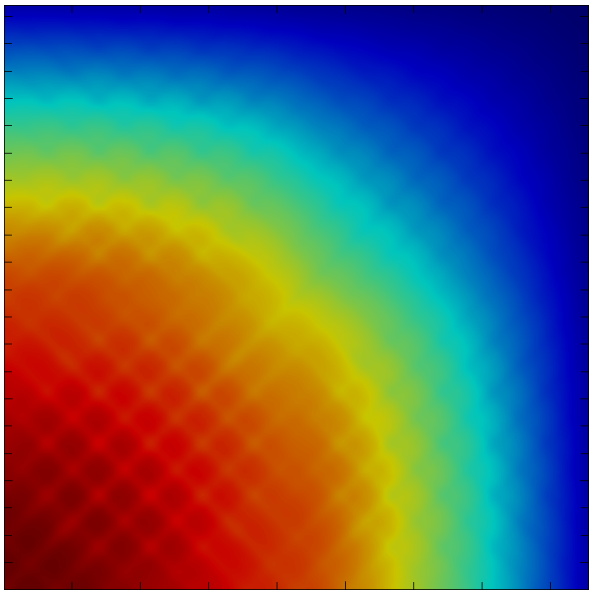
2. Expected performance in other dimensions

For completeness, we show the expected operation counts for the proposed preprocessing in one- and three-dimensional problems. We assume that the mesh consists of quadrilateral elements, so that in d-dimensions the size of the local matrices is defined by $n = (p+1)^d$, where p is the polynomial degree for the approximation. If we consider that the number of operations for Algorithm 1 and for Algorithm 3 for dimension d are

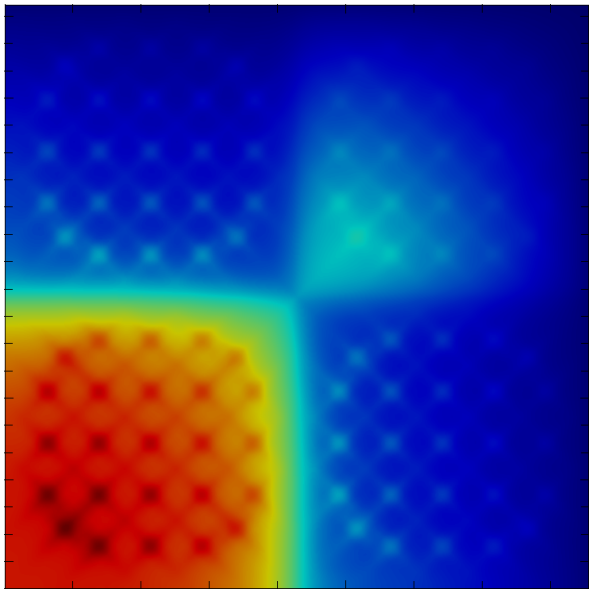
$$C^{\text{fly}}_{p,d} \approx \frac{2}{3}(p+1)^{3d}, \quad C^{\text{qz}}_{p,d} \approx 6(p+1)^{2d}, \tag{17}$$

then for one-dimensional problems we have a count of

$$C^{\text{fly}}_{p,1} \approx \frac{2}{3}(p+1)^{3}, \quad C^{\text{qz}}_{p,1} \approx 6(p+1)^{2}, \tag{18}$$



(a) Fast flux Φ_0



(b) Thermal flux Φ_1

Fig. 2: Fast and thermal flux for C4 problem

and for three-dimensional problems we have a count of

$$C^{\text{fly}}_{p,3} \approx \frac{2}{3}(p+1)^{9}, \quad C^{\text{qz}}_{p,3} \approx 6(p+1)^{6}. \tag{19}$$

In figure 5 we can see the curves for one- and three-dimensional problems for a range of polynomial degrees from 0 up to 7.

We can observe than in one-dimensional problems the lower complexity of the proposed algorithm does not compensate for the range of polynomial degrees that we have chosen. To this we must add the time spent in the preprocessing, and the fact that for lower number our estimators are not as accurate (as shown in the comparison for the two-dimensional case).

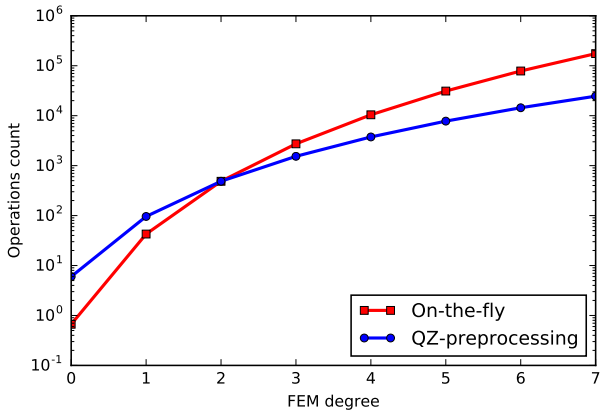


Fig. 3: Operations for different polynomial degrees

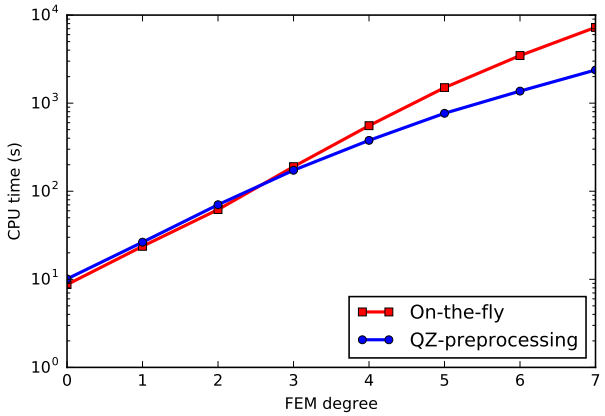


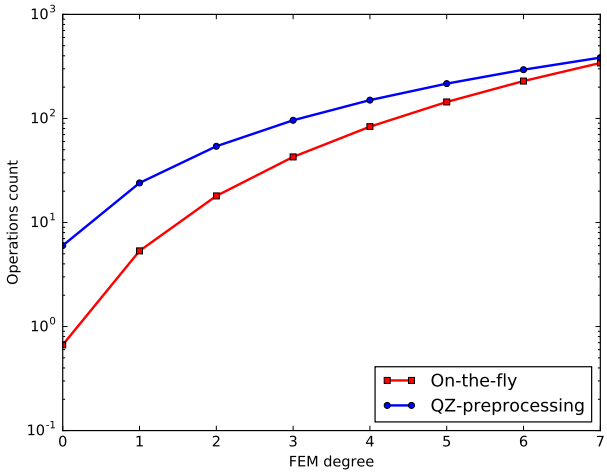
Fig. 4: CPU time for different polynomial degrees

On the other hand, for the three-dimensional case, we see the opposite effect: the higher order of the exponent makes the proposed acceleration to be worth for lower polynomial degrees, and it gets better when the polynomial degree is increased. Nevertheless, we want to notice here than even if three-dimensional calculations are performed nowadays, those are computationally very demanding, and two-dimensional computations still represent the preferred approach.

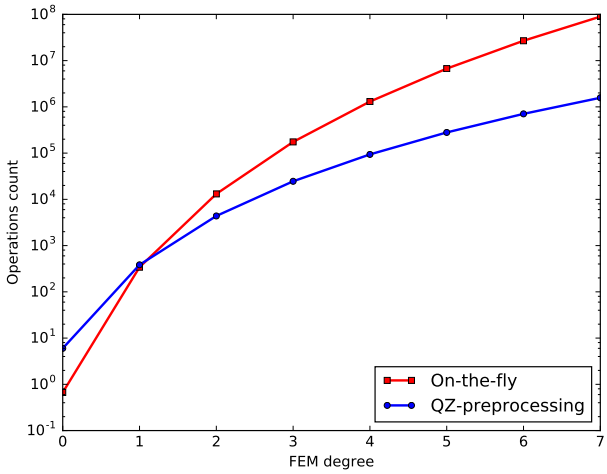
V. CONCLUSIONS

When solving the neutron transport equation with the discrete ordinates method, the transport sweep is one of the most demanding operations, increasingly demanding with respect to the number of angular directions. This operation requires the solution of small linear systems whose matrix is dense. Moreover, the size of this system grows rapidly with the polynomial degree when using a high order Finite Element Method for the spatial discretization.

The solution of these small systems, different for every cross-section and different angular directions, requires cubic complexity, for instance using LU-decomposition or Gaussian elimination. Here, for the specific case of regular meshes, we provide an alternative algorithm, based on a QZ-decomposition in a pre-processing step, allowing us to reduce



(a) Expected operation count in 1d



(b) Expected operation count in 3d

Fig. 5: Expected operation count for 1d and 3d.

to quadratic complexity for the solution of the local system with a minor expense in memory consumption.

This method is tested with a simple problem, improving the original strategy, while more significant improvements are expected with a more efficient usage of the cache.

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers of the extended summary version of this document for many suggestions for improvement and/or extensions of the current work. Additionally, this work has been partially supported by the Swedish Research Council (VR-Vetenskapsrådet) within a framework grant called DREAM4SAFER, research contract C0467701.

APPENDIX

Quasi upper triangular systems

After a QZ decomposition, the local system becomes

$$(\hat{A} + \sigma\hat{B})Q^T x = Zb,$$

where the matrix $\hat{A} + \sigma\hat{B}$ would be quasi-upper triangular. A simpler form as an upper Hessenberg matrix would have worked too, because the LU decomposition of an upper Hessenberg matrix (see [14] for example) together with the solution of the linear systems can be achieved in $O(n^2)$, which is the same as the matrix vector product operation that we already have to do. Here we have chosen the quasi-upper triangular matrix form because there is already a LAPACK routine available that solves this type of system, DLAQTR, and we have decided to rely on that implementation to minimize the damage than a naive implementation can produce in the performance.

A brief explanation of the way this routine works is as follows:

a) For solving a triangular system of size $n \times n$ we must invert every element in the diagonal. Assume we are working in with row i, and then continue with backward substitution performing products with all the elements in the same row with higher column index $i < j < n$. After finishing we move to row $i - 1$ and continue the process.

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 8 & 9 & 10 & 11 & 12 \\ 0 & 0 & 15 & 16 & 17 & 18 \\ 0 & 0 & 0 & 22 & 23 & 24 \\ 0 & 0 & 0 & 0 & 29 & 30 \\ 0 & 0 & 0 & 0 & 0 & 36 \end{pmatrix}$$

b) A quasi upper-triangular system is essentially the same as an upper-triangular system, but for each pair of conjugates complex eigenvalues of this matrix we find a 2×2 block in the diagonal that is not diagonal. Now, solving a system with this quasi upper-triangular matrix of size $n \times n$ works as in the previous case when we find an element in the diagonal that is associated with a real eigenvalue. If instead, we find a 2×2 block consisting of rows i and $i + 1$, then we solve this small system and perform the multiplication and additions blockwise with the elements for column indices $i + 1 < j \leq n$.

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 9 & 10 & 11 & 12 \\ 0 & 14 & 0 & 16 & 17 & 18 \\ 0 & 0 & 0 & 22 & 23 & 24 \\ 0 & 0 & 0 & 0 & 0 & 30 \\ 0 & 0 & 0 & 0 & 35 & 0 \end{pmatrix}$$

REFERENCES

1. W. M. STACEY, *Nuclear reactor physics*, John Wiley & Sons (2007).
2. W. H. REED and T. R. HILL, "Triangular mesh methods for the neutron transport equation," *Los Alamos Report LA-UR-73-479* (1973).
3. P. LESAINT and P. A. RAVIART, *On a Finite Element Method for Solving the Neutron Transport Equation*, Univ. Paris VI, Labo. Analyse Numérique (1974).
4. C. N. MCGRAW, M. L. ADAMS, W. D. HAWKINS, M. P. ADAMS, and T. SMITH, "Accuracy of the Linear Discontinuous Galerkin Method for Reactor Analysis with Resolved Fuel Pins," in "Physor 2014," Kyoto, Japan (2014).
5. M. A. SMITH, E. E. LEWIS, and B.-C. NA, "Benchmark on deterministic 2-D/3-D MOX fuel assembly transport calculations without spatial homogenization (C5G7 MOX Benchmark)," Tech. rep., Report NEA/NSC/DOC (2003) 16, OCDE/NEA, Paris, France (2003).
6. Y. WANG, M. D. DEHART, D. R. GASTON, F. N. GLEICHER, R. C. MARTINEAU, J. W. PETERSON, J. ORTENSI, and S. SCHUNERT, "Convergence study of RattleSnake solutions for the two-dimensional C5G7 MOX benchmark," in "Proceedings of the Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA), and the Monte Carlo (MC) Method, Nashville, TN," (2015).
7. Y. WANG and J. C. RAGUSA, "A high-order discontinuous Galerkin method for the S_N transport equations on 2D unstructured triangular meshes," *Annals of Nuclear Energy*, **36**, *7*, 931–939 (2009).
8. D. FOURNIER, R. HERBIN, and R. L. TELLIER, "Discontinuous Galerkin Discretization and hp-Refinement for the Resolution of the Neutron Transport Equation," *SIAM Journal on Scientific Computing*, **35**, *2*, A936–A956 (2013).
9. S. SCHUNERT, *Development of a Quantitative Decision Metric for Selecting the Most Suitable Discretization Method for SN Transport Problems*, Ph.D. thesis, North Carolina State University) (2013).
10. J. WILLERT, H. PARK, and D. A. KNOLL, "A comparison of acceleration methods for solving the neutron transport k-eigenvalue problem," *Journal of Computational Physics*, **274**, 681–694 (2014).
11. C. B. MOLER and G. W. STEWART, "An algorithm for generalized matrix eigenvalue problems," *SIAM Journal on Numerical Analysis*, **10**, *2*, 241–256 (1973).
12. C. CAVAREC, J. PERRON, D. VERWAERDE, and J. WEST, "Benchmark calculations of power distribution within assemblies," Tech. rep., Nuclear Energy Agency (1994).
13. Y. WANG, W. BANGERTH, and J. RAGUSA, "Three-dimensional *h*-adaptivity for the multigroup neutron diffusion equations," *Progr. Nucl. Energy*, **51**, 543–555 (2009).
14. J. J. BUONI, P. A. FARRELL, and A. RUTTAN, "Algorithms for lu decomposition on a shared memory multiprocessor," *Parallel Computing*, **19**, *8*, 925–937 (1993).