1 Introduction

In scientific computing, it is common to solve a given problem by using methods from combinatorial mathematics. In particular, graphs are ubiquitous in numerical linear algebra if the underlying matrices are sparse. This thesis consists of two parts. In the first part, we discuss new coloring heuristics for the partial Jacobian computation. In the second part, we introduce a set of interactive educational modules teaching these graph problems in classroom.

This thesis is concerned with the solution of linear equations whose coefficient matrices are sparse, large, and nonsingular. Furthermore, it is assumed throughout this thesis that the coefficient matrix is a Jacobian matrix of some mathematical function. This Jacobian matrix is computed by Automatic Differentiation (AD) [1, 2] without truncation error. AD computes a product of a Jacobian and another matrix which is called the seed matrix. A careful choice of the seed matrix will reduce the computational effort as well as the storage. Different choices can be formulated as different graph problems.

These systems of linear equations are solved using iterative methods which are in practice accelerated by preconditioning techniques. Most preconditioning techniques need access to all nonzero elements of the Jacobian matrix which can lead to performance problems when standard techniques of AD are employed. Therefore, we consider an alternative approach that first applies a sparsification operator to the Jacobian matrix and then uses standard preconditioning techniques for the sparsified matrix. This process becomes complete by adding more nonzero elements to the sparsified matrix without producing more fill-in elements and without increasing efforts to set up the resulting sparsified matrix. We introduce new heuristics targeting the careful choice of these extra nonzero elements.

Our work is based on the idea of exploiting the sparsity pattern of a matrix in favor of reducing the computational efforts. The assumption here is that the sparsity pattern is known a priori, for example, from the formulation of a physical problem. Curtis, Powell, and Reid [3] were first to study the determination of a sparse Jacobian matrix based on the sparsity exploitation. Coleman and Moré [4] transformed this sparse matrix problem into the problem of vertex coloring in graph theory. The idea is to formulate the compression of the columns (similarly for rows) such that all the nonzero elements of the Jacobian matrix are determined. This is called unidirectional compression. There are other studies like [5] which consider the byproducts of some column compression schemes. Later, a bidirectional compression is introduced and analyzed in [6] and [7] which can result in larger savings in computation and storage. Several other graph models are studied further in Hossain and Steihaug [8, 9] like the pattern graph which keeps the structure of the matrix explicitly. Also, a recent graph formulation is to group together rows into blocks and partition the resulting column segments in [10]. On the other hand, rather than computing all nonzero elements, some other problems from scientific computing target the computation of only a proper subset of the nonzero elements. These problems are studied under the term of partial Jacobian computation. Gebremedhin, Manne, and Pothen [11] introduced the rules of the partial Jacobian computation and its corresponding graph problems. Several examples in the partial Jacobian computation are studied later in [12] and [13].

Lülfesmann [14] introduced for the first time the idea of combining partial Jacobian computation and ILU preconditioning. This idea defines a sparsification operator ρ which is applied to the Jacobian matrix before the ILU preconditioner is computed. The nonzero elements selected by ρ are called required elements. The remaining elements are called nonrequired elements. Lülfesmann [14] computes the seed matrix for automatic differentiation by considering a graph coloring restricted to these required elements. Then, a subset of nonrequired elements is added to the set of required elements such that the number of colors does not increase and no extra fill-in elements are generated in the ILU preconditioning. These elements are called the additionally required elements. In this thesis, we extend this idea further as follows. First, we define new coloring heuristics (both for distance-2 coloring and star bicoloring) to increase the number of additionally required elements without having a high increase in the number of colors. Then, we apply these new heuristics to an example from geoscience similar to an application from aerodynamics [15]. Later, we generalize a previous result from Lülfesmann [14] for the coloring restricted to diagonal elements. Finally, we introduce a software package to implement these new heuristics.

In the second part of this thesis, we summarize our previous publications [16, 17, 18, 19, 20] as well as discuss some new features to teach the coloring heuristics in classroom. In this part, we develop a collection of educational modules for teaching purposes. Each module illustrates side by side the matrix and graph view of a problem in scientific computing and its equivalent combinatorial problem, respectively. The student can interactively follow the steps of the algorithms in this module. We first outline the overall design of this collection. Then, we discuss the graph coloring module as well as the other available modules. We explain the new unpublished feature in which an animation of the algorithm is visualized. Finally, we explain the implementation details of this collection.

This dissertation is structured as follows. First, we discuss the known graph models from scientific computing in Chapter 2. Then, Chapter 3 discusses our new coloring heuristics. Chapter 4 introduces our interactive educational modules. Finally, the conclusion and future work are presented in Chapter 5.

2 Known graph models from scientific computing

In this chapter, we briefly discuss known graph formulations and models needed in this thesis. In each section of this chapter, we provide some references which explain further these concepts in details. We look at the ideas to determine the nonzeros of sparse Jacobian matrices in Section 2.1. We look at definitions of combining ILU preconditioning and partial Jacobian computation in Section 2.2. Throughout this thesis, we consider the natural ordering of the given matrix for ILU preconditioning.

2.1 Determining nonzeros of sparse Jacobian matrices

There are many references on exploiting the sparsity pattern of Jacobian matrices to improve the performance of automatic differentiation. Here, we look at full and partial Jacobian computation in Section 2.1.1 and Section 2.1.2.

2.1.1 Full Jacobian computation

Assume a program computes a function $f(x) : \mathbb{R}^n \to \mathbb{R}^m$ at the computational cost t. Techniques of automatic differentiation (AD) [1, 2] generate computer programs capable of evaluating the $m \times n$ Jacobian matrix J. The forward mode of automatic differentiation generates a program automatically which computes the product of the Jacobian matrix with a given seed matrix V, i.e., JV. There is a reverse mode of automatic differentiation which computes the product WJ where W is another seed matrix. These techniques of automatic differentiation compute the matrix-matrix products JV and WJ without assembling the Jacobian J.

Suppose the matrix V has c columns and the matrix W has r rows. The computational costs of these products using the forward and reverse modes is then given by ct and rt, respectively. In general, the Jacobian J is computed choosing either $c = n$ and V as the identity of order n in the forward mode or $r = m$ and W as the identity of order m in the reverse mode. However, if J is sparse and its sparsity pattern is known, the number of columns of V in the forward mode or the number of rows of W in the reverse mode can be reduced to $c < n$ or $r < m$ such that all nonzero entries of J still appear in the product JV or WJ. This way, the computational cost is decreased using either the forward mode with an appropriate linear combination of the columns of J or the reverse mode with a suitable linear combination of the rows of J; see the survey [11]. Later in this chapter,

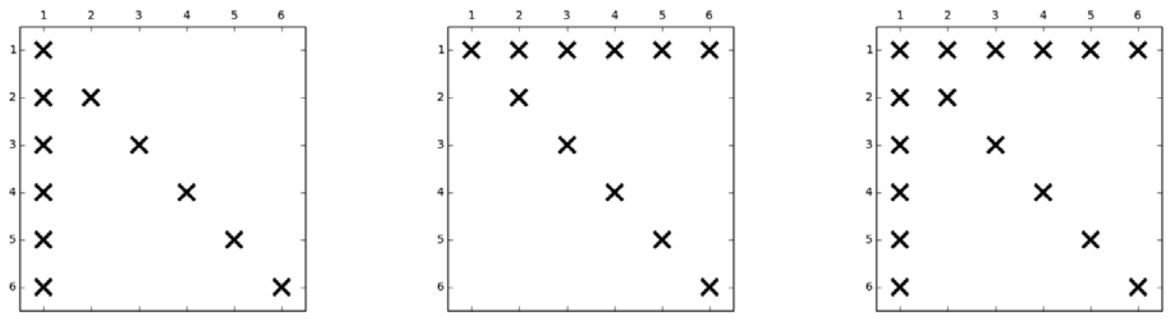

Figure 2.1: (Left) An example of a matrix compressed efficiently by columns. (Middle) An example of a matrix compressed efficiently by rows. (Right) An example of a matrix which cannot be compressed efficiently neither by columns nor by rows.

we formulate problems to compute the minimum values for c and r and the corresponding combinatorial problem.

Scientific computing problem

Here, we find a seed matrix in which the corresponding number of rows and columns is smaller than the actual Jacobian matrix which is called compression. A unidirectional compression is a compression in either rows or columns in contrast to a bidirectional compression in which both rows and columns are compressed at the same time. The key idea behind this *unidirectional compression* is now illustrated for the forward mode. First, we present a definition as follows.

Definition 1 (Structural Orthogonality) *Let $J = [c_1, c_2, \ldots, c_n]$ denote the $m \times n$ Jacobian matrix in which $c_i \in \mathbb{R}^m$ is the ith column. Two columns c_i and c_j are called* structurally orthogonal *if they do not have any nonzero element in a same row. Two columns are called* structurally non-orthogonal *if there is at least one row in which both columns, c_i and c_j, have a nonzero element. Analogously, two rows are* structurally orthogonal *if they do not have any nonzero element in a same column.*

We can compute a linear combination of a group of structurally orthogonal columns of the Jacobian matrix such that this linear combination contains all elements of these columns. The definition of the structurally orthogonal columns can be similarly adapted to rows. It follows that the number of structurally orthogonal groups represents the computational cost either for columns or rows. Figure 2.1 (Left) and Figure 2.1 (Middle) show two examples of matrices which can be compressed efficiently by columns and by rows, respectively. Now, consider the matrix in Figure 2.1 (Right) that has neither structurally orthogonal columns nor structurally orthogonal rows. Therefore, there is no unidirectional compression of the matrix, neither by columns nor rows. However, the technique of bidirectional compression, which compresses both columns and rows at the same time, will reduce the computational cost for that example. This technique uses both forward and reverse modes of automatic differentiation.

For general sparsity patterns, it is not straightforward to figure out how to linearly combine columns and rows such that the computational cost is minimized. Hence, we introduce the combinatorial optimization problems 1 and 2 for unidirectional and bidirectional compression to determine the nonzero elements of large Jacobian matrices efficiently.

Problem 1 (Minimum Unidirectional Compression) *Let J be a sparse $m \times n$ Jacobian matrix with a known sparsity pattern. Find a binary seed matrix V of dimension $n \times c$ whose number of columns is minimized such that all nonzero elements of J also appear in the matrix-matrix product JV.*

The corresponding compression problem for minimizing the number of rows in the matrix-matrix product WJ is straightforward and omitted here.

Problem 2 (Minimum Bidirectional Compression) *Let J be a sparse $m \times n$ Jacobian matrix with known sparsity pattern. Find a pair of binary seed matrices V of dimension $n \times c$ and W of dimension $r \times m$ in which the number of columns of V and the number of rows of W sum up to a minimal value, $c + r$, such that all nonzero elements of J also appear in the pair of matrix-matrix products JV and WJ.*

Combinatorial model

We reformulate the scientific computing problems which we have discussed in Section 2.1.1. The new formulation is an equivalent problem defined on a carefully chosen graph model. The survey [11] discusses different methods to exploit the sparsity involved in derivative computations. We first look at a simple graph model for the unidirectional compression.

Definition 2 (Column Intersection Graph) *The column intersection graph $G = (V, E)$ associated with an $n \times n$ Jacobian matrix J consists of a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ whose vertex v_i represents the ith column $J(:, i)$. Furthermore, there is an edge (v_i, v_j) in the set of edges E if and only if the columns $J(:, i)$ and $J(:, j)$ represented by v_i and v_j are structurally non-orthogonal.*

As we have a graph model associated with our Jacobian matrix in Definition 2, the grouping of columns can be encoded in the following well-known graph coloring problem.

Definition 3 (Coloring) *A coloring of $G = (V, E)$ is a mapping $\Phi : V \to \{1, \ldots, p\}$ with the property $\Phi(v_i) \neq \Phi(v_j)$ if $(v_i, v_j) \in E$.*

Coleman and Moré [4] then showed that Problem 1, which asks for a seed matrix with a minimal number of columns, is equivalent to the following coloring problem.

Problem 3 (Minimum Coloring) *Find a coloring Φ of the column intersection graph G associated with a sparse Jacobian J with a minimal number of colors.*

Although, this model is convincing for the unidirectional compression, the bidirectional compression can not be an instance of this model. A bidirectional compression needs the information of both rows and columns. Therefore, a bipartite graph model is defined for this purpose as in [6, 21, 7].

Definition 4 (Bipartite Graph Model) *In the bipartite graph model, the vertex set $V = V_c \cup V_r$ is decomposed into a set of vertices V_c representing columns of J and another set of vertices V_r representing rows. The set of edges E is used to represent the nonzero elements and it is defined as follows. An edge $(c_i, r_j) \in E$ connects a column vertex $c_i \in V_c$ and a row vertex $r_j \in V_r$ if there is a nonzero element in J at the position represented by c_i and r_j. The graph is bipartite indicating that all edges connect vertices from one set V_c to the other set V_r. That is, there is no edge connecting vertices within the set V_c or within V_r. Moreover, two vertices that are connected by a path of length two, are called* distance-2 neighbors*.*

The coloring problem in the column intersection graph can also be represented in this bipartite graph model. This equivalent coloring is done only in the set of column vertices. Also, *distance*-2 *neighbors* should be considered instead of adjacent vertices.

The overall idea behind transforming Problem 2, Minimum Bidirectional Compression, into an equivalent problem using the bipartite graph model is as follows. The grouping of the columns and rows is expressed by representing each group by a color. Vertices that belong to the same group of columns/rows are assigned the same color. Formally, this is represented by a coloring of a bipartite graph. Such a coloring is a mapping

$$\Phi : V_c \cup V_r \to \{0, 1, \ldots, p\}$$

that assigns to each vertex a color represented by an integer. The coloring Φ also involves a “neutral” color representing the following “don't color” situation. A vertex $v \in V_c \cup V_r$ that is not used in the grouping of columns/rows is assigned the neutral color $\Phi(v) = 0$. More precisely, if $\Phi(v) = 0$ for a column vertex v then every nonzero represented by an incident edge of v is determined by a linear combination of rows. Similarly, a nonzero entry represented by an edge that is incident to a neutrally-colored row vertex is determined by a linear combination of columns.

To represent the process of finding seed matrices using the bipartite graph model, it is necessary to consider the underlying properties, which are as follows:

1. The computational cost roughly consists of the number of groups of structurally orthogonal columns and rows. Since the overall cost is the sum of the costs associated with the forward mode and the reverse mode, the (non-neutral) colors for the forward mode and the (non-neutral) colors for the reverse mode need to be different.

2. It may happen that some nonzero elements may be computed twice, by the forward mode in JV and by the reverse mode in WJ. Therefore, an edge representing such a nonzero element connects two vertices with two different non-neutral colors. In

general, since the MINIMUM BIDIRECTIONAL COMPRESSION problem asks for computing *all* nonzero elements, at least one vertex of every edge has to be colored with a non-neutral color.

3. Suppose two columns are structurally non-orthogonal and have a nonzero element in a same row. If this row is not handled by the reverse mode, these two columns need to be in different column groups. The same argument holds for corresponding situations with row groups.

4. Consider three nonzero elements in the matrix positions (i, k), (i, ℓ), and (j, k). Suppose that the nonzero at (i, k) is computed by the reverse mode assigning some (non-neutral) color to the row vertex r_i. Then, if (j, k) is also computed via the reverse mode, a second (non-neutral) color is needed for r_j. Now, if (i, ℓ) is already determined by the reverse mode for the row i the column vertex c_ℓ is assigned the neutral color. However, if (i, ℓ) is computed by the forward mode, a third (non-neutral) color is needed for c_ℓ. A similar argument holds if (i, k) is computed by the forward mode.

Based on these considerations, the following definition captures these properties.

Definition 5 (Star Bicoloring) *Given a bipartite graph $G = (V_c \cup V_r, E)$, then a mapping $\Phi : V_c \cup V_r \to \{0, 1, \ldots, p\}$ is a star bicoloring of G if the following conditions are satisfied:*

1. *Vertices in V_c and V_r receive disjoint colors, except for the neutral color 0. That is, for every $c_i \in V_c$ and $r_j \in V_r$, either $\Phi(c_i) \neq \Phi(r_j)$ or $\Phi(c_i) = \Phi(r_j) = 0$.*

2. *At least one vertex of every edge receives a non-neutral color. That is, for every $(c_i, r_j) \in E$, the conditions $\Phi(c_i) \neq 0$ or $\Phi(r_j) \neq 0$ hold.*

3. *For every path (u, v, w) with $\Phi(v) = 0$, the condition $\Phi(u) \neq \Phi(w)$ is satisfied.*

4. *Every path of length three with four vertices uses at least three colors (possibly including the neutral color).*

Using the bipartite graph model and the definition of a star bicoloring, the problem MINIMUM BIDIRECTIONAL COMPRESSION is equivalent to the following graph problem.

Problem 4 (Minimum Star Bicoloring) *Given the bipartite graph $G = (V_r \cup V_c, E)$ associated with a sparse Jacobian matrix J, find a star bicoloring of G with a minimal number of non-neutral colors.*

A unidirectional compression is a special case of a bidirectional compression. More precisely, a unidirectional compression with respect to columns corresponds to a star bicoloring in which all the vertices in V_c are colored with a non-neutral color and all row vertices are colored with the neutral color. This way, the coloring constraint of a star bicoloring reduces to the coloring of distance-2 neighbors in the bipartite graph using different (non-neutral) colors. This distance-2 coloring in the bipartite graph model is then equivalent to a coloring in the undirected graph model in which all neighbors are colored differently. Finally, a discussion of the computational complexity of Problem 4 including recent new results is given in [22].

2.1.2 Partial Jacobian computation

Gebremedhin et al. [11] introduced the concept of partial Jacobian computation in which only a subset of the nonzero Jacobian entries, the required elements, are to be determined. Lülfesmann [14] studied this area in more details and introduced some heuristics for partial computation.

Scientific computing problem

Let R be the set representing required elements. The definition of *structural orthogonality* is adapted for partial Jacobian computation as follows.

Definition 6 (Partially Structural Orthogonality) *Two columns c_i and c_j are partially structurally orthogonal with respect to R if and only if they do not have a nonzero element in a same row where at least one of these nonzero elements is required.*

The corresponding combinatorial optimization problem for the unidirectional and bidirectional compression restricted to the required elements can be formulated as follows.

Problem 5 (Minimum Partial Unidirectional Compression) *Let J represents a sparse $m \times n$ Jacobian matrix with known sparsity pattern and R be a subset of the nonzero elements of J. Find a binary seed matrix V of dimension $n \times c$ whose number of columns is minimized such that all nonzero elements of R also appear in the matrix-matrix product JV.*

Problem 6 (Minimum Partial Bidirectional Compression) *Let J represents a sparse $m \times n$ Jacobian matrix with known sparsity pattern and R be a subset of the nonzero elements of J. Find a pair of binary seed matrices V of dimension $n \times c$ and W of dimension $r \times m$ in which the number of columns of V and the number of rows of W sum up to a minimal value, $c + r$, such that all nonzero elements of R also appear in the pair of matrix-matrix products JV and WJ.*

Now, we discuss an equivalent graph-theoretical formulation of this problem.

Combinatorial model

Based on [11, 14], the definitions of full Jacobian coloring are adapted for the restricted colorings as follows.

Definition 7 (Restricted distance-2 coloring) *Given a bipartite graph $G = (V_c \cup V_r, E)$ and a subset of required edges $E_R \subseteq E$, then a mapping $\Phi : V_c \to \{0, 1, \ldots, p\}$ is a distance-2 coloring of G restricted to E_R if all column vertices incident to at least one required edge $e \in E_R$ get a nonzero color and for every path (c_k, r_i, c_j) with $c_k, c_j \in V_c$, $r_i \in V_r$, and $(r_i, c_j) \in E_R$, $\Phi(c_k) \neq \Phi(c_j)$.*

Definition 8 (Restricted star bicoloring) *Given a bipartite graph $G = (V_c \cup V_r, E)$ and a subset of required edges $E_R \subseteq E$, then a mapping $\Phi : V_c \cup V_r \to \{0, 1, \ldots, p\}$ is a star bicoloring of G restricted to E_R if the following conditions are satisfied:*

1. *Vertices in V_c and V_r receive disjoint colors, except for the neutral color 0. That is, for every $c_i \in V_c$ and $r_j \in V_r$, either $\Phi(c_i) \neq \Phi(r_j)$ or $\Phi(c_i) = \Phi(r_j) = 0$.*

2. *At least one end point of an edge in E_R receives a nonzero color.*

3. *For every edge $(r_i, c_j) \in E_R$, $r_i, r_l \in V_r$, and $c_j, c_k \in V_c$,*
   - *if $\Phi(r_i) = 0$, then for every path (c_k, r_i, c_j), $\Phi(c_k) \neq \Phi(c_j)$*
   - *if $\Phi(c_j) = 0$, then for every path (r_i, c_j, r_l), $\Phi(r_i) \neq \Phi(r_l)$*
   - *if $\Phi(r_i) \neq 0$ and $\Phi(c_j) \neq 0$, then for every path (c_k, r_i, c_j, r_l), $\Phi(c_k) \neq \Phi(c_j)$ or $\Phi(r_i) \neq \Phi(r_l)$*

Now, the optimization problems are formulated as follows.

Problem 7 (Minimum Restricted Distance-2 Coloring) *Given the bipartite graph $G = (V_r \cup V_c, E)$ associated with a sparse Jacobian matrix J and a set of required edges E_R, find a distance-2 coloring of G restricted to E_R with a minimal number of non-neutral colors.*

Problem 8 (Minimum Restricted Star Bicoloring) *Given the bipartite graph $G = (V_r \cup V_c, E)$ associated with a sparse Jacobian matrix J and a set of required edges E_R, find a star bicoloring of G restricted to E_R with a minimal number of non-neutral colors.*

2.2 Combining partial Jacobian computation and ILU

Given a large sparse nonsingular $n \times n$ Jacobian matrix J, we are considering the solution to the following system of linear equations,

$$Jx = b,$$